Development of Gasoline Fuel Effects in the Motor Vehicle Emissions Simulator (MOVES2009)

Draft Report

United States
Environmental Protection
Agency

Development of Gasoline Fuel Effects in the Motor Vehicle Emissions Simulator (MOVES2009)

Draft Report

Assessment and Standards Division
Office of Transportation and Air Quality
U.S. Environmental Protection Agency

NOTICE

This technical report does not necessarily represent final EPA decisions or positions. It is intended to present technical analysis of issues using data that are currently available. The purpose in the release of such reports is to facilitate the exchange of technical information and to inform the public of technical developments.

United States
Environmental Protection
Agency

EPA-420-P-09-004
August 2009

Table of Contents

1. Introduction 1
2. Structure of the Draft MOVES2009 Fuel Algorithm Database 2
   2.1 Fuel Formulation Table 2
   2.2 Fuel Supply Table 3
   2.3 Fuel Adjustment Table 3
   2.4 Hydrocarbon Speciation Table 7
3. Process Used to Create MOVES Fuel Adjustments 10
   3.1 Extracting the Primary Data from NMIM 10
   3.2 MOVES Fuel Binner 11
   3.3 Predictive Model Effects 14
   3.4 Complex Model Effects 15
   3.5 Sulfur Effects 16
4. Process Used to Create Air Toxic Adjustment Factors 16
5. Use of the MOVES Fuel Algorithm in RFS2 20
Appendix A MySQL Code for the Fuel Binner 23

1. Introduction

This report discusses the development of the fuel adjustment factors to the basic emission rates which are contained in the Draft MOVES2009 version released in April, 2009, and those fuel effects generated and used in the Draft RFS2 Notice of Proposed Rulemaking (NPRM) modeling efforts of December, 2008. This report and its appendices describes the technical development of the draft MOVES model fuel parameters (version December 15, 2008) and the analysis / programming process used to develop the fuel adjustment factors for the RFS2 inventory modeling. The first section describes the structure and design of the MOVES model, and its accompanying data structures. The second section gives a brief overview of the individual emission fuel models, their use in the MOVES context, and the process used to generate fuel formulations, fuel adjustment factors and fuel supply distributions. The third section is an Appendix which contains all of the data scripts used in the MOVES process. The detail oriented reader will find all of the calculations and database manipulations documented in detail in these appendices. Because of the volume, such algorithms are not reproduced in the general text. Finally, this document contains only limited and generalized results from the model. It does not contain the specific results from regulatory analysis runs or sensitivities.

EPA is planning, and has partially completed, a completely revised fuel effects algorithm for inclusion in the proposed Final December, 2009 release of the MOVES model. This new algorithm was also used for emission inventory analysis in the Final RFS2 Rulemaking (August, 2009). In the new version, the entire fuel calculation algorithm was rewritten (substantial new MOVES coding and database structures have been added), and new fuel algorithm effects will be inserted based on recent Energy Policy Act (EPAct testing). These new effects are mostly confined to the topics of ethanol fuel effects and biodiesel effects. However, new fuel effects have also been developed for conventional gasoline vehicles with Tier2 (2004+ MY) certifications. The RFS2 rulemaking modeling also required that MOVES contain both a ‘more sensitive’ and ‘less sensitive’ case for Tier1 and later model years, whereas, Draft MOVES2009 (the algorithm in the current document) contains only the ‘less sensitive’ case. In this context, ‘more sensitive’ means the fuel adjustment effects were a function of a range of fuel parameters such as fuel sulfur level, RVP, oxygen content, distillation fractions, etc., and ‘less sensitive’ means the fuel adjustment effects were only a function of fuel sulfur level. The basic structure of the new algorithm is also different and this change will also likely cause some minor change in the overall fuel effects. A new report(s) which discusses the Final December, 2009 model has yet to be written (expected in September, 2009), but is planned prior to the release of the Final MOVES model in December, 2009. This new report will discuss the new fuel algorithm in detail and present some limited results.

2. Structure of the Draft MOVES2009 Fuel Algorithm Database

The MOVES model is a data driven model that consists primarily of a central database that is manipulated using a Java based GUI and a series of complex MySQL scripts. The MOVES program also allows the use of alternative user defined data tables in the program.

The Draft MOVES2009 Fuel Algorithm consists of four primary data tables. These are:

FuelFormulation
FuelSupply
FuelAdjustment
HCSpeciation

2.1 Fuel Formulation Table

The FuelFormulation table contains a table of fuel formulations 'keyed' using an arbitrary fuelformulationID variable, and a set of fuel. The fuel subtype designates fuel classifications such as conventional gas, reformulated gas, gasohol (i.e, 10% ethanol), and E-85/E70 ethanol-gasoline blends. The model contains several hundred individual fuelformulationIDs which cover the range of all important fuel properties.

MOVES Fuel Formulation Table Structure

fuelFormulationID	key field
fuelSubtypeID	
RVP	
sulfurLevel	
ETOHVolume	
MTBEVolume	
ETBEVolume	
TAMEVolume	
aromaticContent	
olefinContent	
benzeneContent	
e200	
e300	
volToWtPercentOxy	

2.2 Fuel Supply Table

The Fuel Supply table contains the market share data for each fuel formulation as a function of calendar year, month and county (United States). This information varies considerably by calendar year and county with new fuel formulations being phased-in over time. The Draft MOVES2009 model contains fuel supply information for the 1999 through 2012 calendar years, and allows the user to input specific data for a county / year / month combination. Market share value for gasoline, diesel and ethanol based fuels are available. The values typically vary by season. Fuel market share information is provided for all 3,222 counties and is based on in-use fuel surveys and econometric modeling projections for future calendar years *(the primary data for the future projections is taken from EPA's NMIM model which contains data used in National Emission Inventory process*[1]. The market share values are used to weight the effects of different fuelformulationID together to obtain an overall average fuel effect / fuel adjustment. The market share information is a function of county, calendar year (i.e., fuelYearID) and month. The marketShareCV is the coefficient of variation in the marketShare variable and is used for uncertainty calculation in the model. It is currently empty.

MOVES Fuel Supply Table Structure	
CountyID	key field
fuelYearID	key field
monthGroupID	key field
fuelFormulationID	key field
marketShare	
marketShareCV	

2.3 Fuel Adjustment Table

The Fuel Adjustment table contains all of the MOVES fuel adjustments. They are a function of pollutant - total hydrocarbon, carbon monoxide and nitrogen oxides (Total

[1] Document Id EPA-HQ-OAR-2005-0161-DRAFT-0920
Document Location Docket EPA-HQ-OAR-2005-0161 Phase NPRM - Notice of Proposed Rulemaking Sequence 3
Document Title: Using MOVES to Generate Inventories for the RFS2 NPRM
Memorandum to Docket from Amanda Valente, Megan Beardsley, David Brzezinksi,
Ed Glover & Prashanth Gururaja
U.S. EPA/OTAQ/ASD/AQMC
November 2008

HC, CO and NOx) – process (start, running, etc), fuel formulation, model year group and source type. Fuel Adjustment factors were developed for each of the following model year groups.

1974
19751986
19871989
19901993
1994
1995
1996
19972000
20012003
2004
2005
2006
2007
20082009
20102050

Fuel Adjustment factors were developed for the following sourcetypeIDs:

11	Motorcycle
21	Passenger Car
31	Passenger Truck
32	Light Commercial Truck
41	Intercity Bus
42	Transit Bus
43	School Bus
51	Refuse Truck
52	Single Unit Short-haul Truck
53	Single Unit Long-haul Truck
54	Motor Home
61	Combination Short-haul Truck
62	Combination Long-haul Truck

The MOVES model fuel adjustment factor is also a strong function of specific fuel properties. Most of these effects are non-linear and were derived from the EPA's - Complex model, Predictive model and MOBILE6 Sulfur model (labeled as Primary EPA Fuel Models in the remainder of this document). These effects are utilized in MOVES by linking the fuel formulation table and the fuel adjustment table using the variable fuelFormulationID.

All of the MOVES fuel adjustment factors are multiplicative correction factors to the basic emission factor in MOVES. This is shown mathematically in the simple equation (Eq 1) below.

Fuel Corrected Emissions = Fuel Adjustment Factor * Base Emissions Factor Eq 1

The Base Emission Factor variable in Eq 1 represents the base emission rates computed by MOVES. These were computed primarily from Arizona IM240 lane data, for pre-2001 model years, and computed from EPA in-use vehicle surveillance testing for model years 2001 through 2006. Base emission factors for 2007 and later model years were computed from multiplicative adjustment factors that ratio the existing post-2000 calendar year emission, and the future Tier2 emission standard bins.

The Fuel Adjustment Factors are created from emission results obtained by running the Primary EPA Fuel Models for all combinations of in-use fuel formulations. The actual Fuel Adjustment Factor for a given fuel (i) is the ratio of the emissions from fuel (i) and the emissions from the reference fuel (see Eq 2) using the Primary EPA Fuel Models.

Fuel Adjustment Factor (fuel i) = emission result(fuel i) / emission result (reference fuel) **Eq2**

Two reference fuels were used in the development of the MOVES Fuel Adjustment Factors, and their properties are shown in Table 1. The use of only two average reference fuels clearly incorporates considerable uncertainty since the emission rates are based on in-use testing of vehicles using many different and unknown fuels. The two fuels selected as reference fuels were thought to be the most representative of those in place where and when the vehicles were tested: Phoenix, AZ in summer months, 1995 through 2002.

Table 1
Reference Fuel Properties

Fuel Property	Pre-2001 Reference Fuel	2001+ Reference Fuel
Fuel Subtype	Conventional Gasoline	Conventional Gasoline
RVP (psi)	6.9	6.9
Sulfur Level (ppm)	90	30
Ethanol Volume (vol%)	0	0
MTBE Volume (vol%)	0	0
TAME Volume (vol%)	0	0
Aromatic Content (vol%)	26.1	26.1
Olefin Content (vol%)	5.6	5.6
Benzene Content (vol%)	1.0	1.0
E200 (F)	41.1	41.1
E300 (F)	83.1	83.1

The two reference fuels are identical except for the fuel sulfur levels of 30 ppms sulfur (fuelformulationid=98) and 90 ppm sulfur (fuelformulationid=99). The reference fuel with a sulfur level of 30 ppm is used to create Fuel Adjustment Factors for all 2001 and later model years, and the value of 90 ppm is used for all 2000 and earlier model years. The value of 30 ppm was chosen for the late model vehicles because it is likely that these model years are more likely to be exposed to the lower sulfur fuel than the older model years. Using a 90 ppm sulfur reference fuel on nLEV and Tier2 vehicles which were generally certified or operated on lower sulfur fuels would over-state the effects of sulfur on new and future vehicles.

The Fuel Adjustment table contains both the fuelAdjustment and the fuelAdjustmentGPA variables. The "GPA" (Geographic Phase-in Area) refers to a group of counties in the western U.S. that had special gasoline sulfur requirements under Tier 2 regulations. See 65 Fed. Reg.6755-6759 for more information on the GPA. The values are identical except for specific counties and fuelMYGroups 2004, 2005 and 2006. In those cases, the fuelAdjustmentGPA is slightly higher than the fuelAdjustment because it accounts for the irreversible effects of higher fuel sulfur levels

In MOVES, if a particular county / year / month bin has more than one fuel formulation (a common situation), the fuel adjustment factors are weighted using the market share data. See Equation 3 below. The exhaust emissions model contains no logic to account for the mixing of various fuels in a given vehicle's fuel tank (commingling). Thee Tank Fuel Generator used for TVV does adjust RVP to account for commingling.

Fuel Adjustment Factor = Adjustment1 * market share1 + Adjustment2 * market share2 + … + Adjusment i * market share i **Eq 3**

MOVES Fuel Adjustment Table Structure

polProcessID	key field
fuelMYGroupID	key field
sourceTypeID	key field
fuelFormulationID	key field
fuelAdjustment	
fuelAdjustmentCV	
fuelAdjustmentGPA	
fuelAdjustmentGPACV	

2.4 Hydrocarbon Speciation Table

The HCSpeciation table provides factors which convert the base emission factors which are in terms of total hydrocarbon emissions into hydrocarbon emissions of other terms. These are volatile organic compounds (VOC), non methane hydrocarbons (NMHC), total organic gases (TOG) and non methane organic gases (NMOG). See Table 2 below for the relationship between the various hydrocarbon speciation types. The MOVES model uses a 'chaining' method of calculation for the individual hydrocarbon speciation types. All of them start with the Total Hydrocarbon (THC) pollutant.

Table 2
Hydrocarbon Speciation Types

PollutantID	PollutantName	FID HC	Methane	Ethane	Aldehydes
1	Total Hydrocarbons	Yes	Yes	Yes	No
79	Non Methane Hydrocarbons	Yes	No	Yes	No
87	Volatile Organic Compounds	Yes	No	No	Yes
86	Total Organic Gases	Yes	Yes	Yes	Yes
80	Non Methane Organic Gases	Yes	No	Yes	Yes

The structure of the MOVES HCSpeciation table is:

MOVES HCSpeciation Table Structure

polProcessID	key field
fuelMYGroupID	key field
fuelFormulationID	key field
speciationConstant	
oxySpeciation	

The HCSpeciation table contains data for only the hydrocarbon pollutants listed above in Table 2. The fuelMYGroups and the fuelFormulationID keys have the same definitions as those used in the Fuel Adjustment table. The HC speciated compounds of VOC (volatile organic compounds) and NMOG (non methane organic gases) utilize the coefficients speciationConstant and oxySpeciation in a linear equation. The factors are a function of the oxygenate type (if present in the fuel). If a given target fuel formulation contains a mixture of oxygenate types that includes ethanol, the model will use the ethanol factors. If the mixture does not contain ethanol it will use the MTBE factors, and if it does not contain either ethanol or MTBE, it will utilize the TAME factors. Non methane hydrocarbons (NMHC) are computed by subtracting methane emissions from total hydrocarbon emissions. The methane emissions are calculated in MOVES using

independent emission factors derived from methane emission data (see MOVES Methane emission calculation report). Total organic emissions (TOG) are calculated by adding methane emissions to the NMOG emission factors. See Equations 3, 4, 5 and 6 for details.

The HCSpeciation table contains data for only the hydrocarbon pollutants listed above in Table 2. The fuelMYGroups and the fuelFormulationID keys have the same definitions as those used in the Fuel Adjustment table. The HC speciated compounds of VOC (volatile organic compounds) and NMOG (non methane organic gases) utilize the coefficients speciationConstant and oxySpeciation in a linear equation. The factors are a function of the oxygenate type (if present in the fuel). Non methane hydrocarbons (NMHC) are computed by subtracting methane emissions from total hydrocarbon emissions. The methane emissions are calculated in MOVES using independent emission factors derived from methane emission data. Total organic emissions (TOG) are calculated by adding methane emissions to the NMOG emission factors. See Equations 3, 4, 5 and 6 for details.

NMHC = THC - Methane **Eq 3**

VOC = NMHC * (speciationConstant + oxySpeciation* volToWtPercentOxy*ETOHVolume) **Eq 4a**

For fuels containing ethanol (fuelsubtypeID = 12, 13, 14, 51, 52, or 53)

VOC = NMHC * (speciationConstant + oxySpeciation* volToWtPercentOxy*MTBEVolume) **Eq 4b**

For fuels containing MTBE

VOC = NMHC * (speciationConstant + oxySpeciation* volToWtPercentOxy*ETBEVolume) **Eq 4c**

For fuels containing ETBE

VOC = NMHC * (speciationConstant + oxySpeciation* volToWtPercentOxy*TAMEVolume) **Eq 4d**

For fuels containing TAME

NMOG = NMHC * (speciationConstant + oxySpeciation* volToWtPercentOxy*ETOHVolume) **Eq 5a**

For fuels containing ethanol (fuelsubtypeID = 12, 13, 14, 51, 52, or 53)

NMOG = NMHC * (speciationConstant + oxySpeciation* volToWtPercentOxy*MTBEVolume) **Eq 5b**

For fuels containing MTBE

NMOG = NMHC * (speciationConstant + oxySpeciation* volToWtPercentOxy*ETBEVolume) **Eq 5c**

For fuels containing ETBE

NMOG = NMHC * (speciationConstant + oxySpeciation* volToWtPercentOxy*TAMEVolume) **Eq 5d**

For fuels containing TAME

TOG = NMOG + Methane **Eq 6**

3. Process Used to Create MOVES Fuel Adjustments

This section is an overview of the process and algorithms used to develop the MOVES fuel formulations, fuel supply information and fuel adjustments. Specific details of individual database manipulations and calculations can be found in the Appendix at the end of this document.

3.1 Extracting the Primary Data from NMIM

The primary fuel formulation and fuel supply data were obtained from EPA's NMIM model and national emission inventory (NEI) processs. The NMIM model is currently used to develop national emission inventories every three years. NMIM currently contains fuel formulation and fuel supply data and projections for all calendars years from 1999 through 2030. Eventually, all these data will be extracted, updated if necessary and fully processed into the MOVES model. However, as an initial product for the 2008 / 2009 MOVES model release, and for support of EPA's RFS2 Rulemaking, only fuel data from calendar years 2005 and 2022 were extracted and used. Draft MOVES2009 and the RFS2 NPRM & AQ model used different fuel supplies. Draft MOVES2009 goes only to 2012.

For calendar year 2005, a database called NMIMRFS2Fuels2005Base was created. It contains detailed fuel formulation and fuel supply data for calendar year 2005 collected from in-use fuel surveys. It contains both fuelformulation data for several thousand individual in-use fuels and fuel supply data from all 3,222 US counties on a seasonal basis. The analysis process used to create this database was done under EPA contract by Eastern Research Group and has been fully documented in the EPA 2007 Renewable Fuels Standard (RFS1) Rulemaking documentation.

Four additional databases were created for calendar year 2020 / 2022. These are:

Database: RFS2Frm2022Fuels.

Name	From Spreadsheet	Actual Database Name
Rfs2Frm2022Aeo	fuel parms for AQMC 11-26.xls	NmimRfs2Fuels2022AEO.zip
Rfs2Frm2022Rfs1	fuel parms for AQMC 12-04 - RFS1 ref case.xls	NmimRfs2Fuels2020rfs1.zip
Rfs2Frm2022E10	fuel parms for AQMC 12-01 - RFS2 policy case.xls	NmimRfs2Fuels2022e10.zip
Rfs2Frm2022E85	fuel parms for AQMC 11-19.xls	NmimRfs2Fuels2022e85.zip

E10 and E85 are for the EISA control case, but they will be run separately in MOVES and weighted together. In NMIM, which will be used for motocycles, diesel, and nonroad, the E10 case will be run. The E85 database will be prepared in NMIM, because that is the basis for creating MOVES fuels tables.

The 2022 Reference fuel database containing fuel supply projections by county, month and fuel formulation was created from two sources. These sources were the output of EPA refinery modeling under contract, which provided the bulk of the information, and projections from Advanced Energy Outlook (AEO). The AEO projections were the volume of E10 (a 90% gasoline / 10% ethanol mixture) penetration into the market. Ethanol market share information is important because the focus of the RFS2 rulemaking is ethanol production, marketing and emission effects. The 2022 Control fuel database is virtually the same as the Reference fuel database except it assumes that E10 fuels have 100 percent market share in 2022.

3.2 MOVES Fuel Binner

A real in-use gasoline fuel can vary continuously according to all of the fuel properties listed in Table 1. However, the Draft MOVES2009 model design does not currently process such fuel formulation information; the database lists it only for reference and for use in some special cases such as sulfate emissions and tank vapor venting. In general, the MOVES Fuel Binner organizes specific NMIM fuels into bins according to the eleven fuel properties listed in Table 1, and assigns average values. These average fuel property values are then processed outside the MOVES model in the Predictive, Complex and EPA Sulfur models. This process creates fuel adjustment factors and air toxic pollutant factors which are used in MOVES.

The MOVES fuel adjustment factors design was created as a compromise between utilizing detailed fuel information and the need to streamline processing in the MOVES model. Some pollutants are sensitive to certain fuel properties and detail is required for accuracy. In these cases the MOVES Fuel Binner was designed to capture these effects. On the other hand, every possible combination (i.e., greater than 10,000) of eleven fuel properties, twelve source / vehicle types, three fuel types, nine pollutants and ten model year groups quickly overwhelms the MOVES model making some aggregation necessary.

We plan to revise final MOVES2009 to perform the predictive / complex model calculations directly in the model. This will improve the performance issues associated with the current version of MOVES and allow for accurate determination of fuel factors for any reasonable fuel formulation. However, draft MOVES2009 does not contain this capability.

All 10,000+ NMIM fuel formulations were binned according to the eleven fuel properties listed in Tables 1 and 3. Table 3 shows the bins which were created and their definitions. A particular MOVES fuelformulationID has one and only one value for each of the fuel property bins. The column Avg Value does not typically represent an average value of the fuels in the bin. Instead it is more likely a 'mode' value (most of the individual data points in the bin had the Avg Value).

Table 3
Gasoline Fuel Property Bin Definitions

Fuel Property	Bin ID	Avg Value	Lower Bound (>)	Upper Bound (<=)
Fuel Subtype				
Conventional Gas	10			
RFG	11			
Ethanol 10 vol%	12			
Ethanol 8 vol%	13			
Ethanol 5 vol%	14			
Diesel	20			
RVP (psi)	1	6.9	0	7.2
RVP (psi)	2	7.5	7.2	8.2
RVP (psi)	3	8.7	8.2	9.0
RVP (psi)	4	9.2	9.0	10.0
RVP (psi)	5	10.0	10.0	11.4
RVP (psi)	6	11.5	11.4	13.4
RVP (psi)	7	13.5	13.4	14.9
RVP (psi)	8	15.0	14.9+	
Sulfur Level (ppm)	1	5.0	0	10
Sulfur Level (ppm)	2	15.0	10	25
Sulfur Level (ppm)	3	30.0	25	35
Sulfur Level (ppm)	4	50.0	35	70
Sulfur Level (ppm)	5	90.0	70	120
Sulfur Level (ppm)	6	180.0	120	210
Sulfur Level (ppm)	7	280.0	210	300
Sulfur Level (ppm)	8	400.0	300	500
Sulfur Level (ppm)	9	600.0	500+	
Ethanol Volume (vol%)	1	0	0	0
Ethanol Volume (vol%)	2	5.0	0	6.0
Ethanol Volume (vol%)	3	8.0	6.0	8.0
Ethanol Volume (vol%)	4	10.0	8.0+	
MTBE Volume (vol%)	1	0	0	0
MTBE Volume (vol%)	2	2.5	0	5.0

MTBE Volume (vol%)	3	8.0	5.0	10.0
MTBE Volume (vol%)	4	11.0	10.0+	
TAME Volume (vol%)	1	0	0	0
TAME Volume (vol%)	2	0.015	0	0.03
TAME Volume (vol%)	3	0.06	0.03+	
Aromatic Content (vol%)	1	17.5	0	20.0
Aromatic Content (vol%)	2	26.1	20.0	30.0
Aromatic Content (vol%)	3	32.0	30.0+	
Olefin Content (vol%)	1	5.6	0	8.0
Olefin Content (vol%)	2	9.2	8.0	10.0
Olefin Content (vol%)	3	11.9	10.0+	
Benzene Content (vol%)	1	0.35	0	0.40
Benzene Content (vol%)	2	0.65	0.40	0.60
Benzene Content (vol%)	3	0.65	0.60	0.80
Benzene Content (vol%)	4	1.50	0.80	2.00
Benzene Content (vol%)	5	3.50	2.00	
E200 (F)	1	41.0	0	47.0
E200 (F)	2	50.0	47.0+	
E200 (F)				
E300 (F)	1	78.6	0	80.0
E300 (F)	2	83.0	80.0	87.0
E300 (F)	3	89.1	87.0+	

After binning the 10,000+ NMIM fuel formulations into a more manageable set of less than 500 MOVES fuel formulations *(in 2022 in some RFS2 runs there were less than 200 fuel formulations*), the MOVES fuel adjustment factor process used an algorithm to create the specific fuel adjustment factors for each of the MOVES fuel / pollutant-process combinations. These fuel effects are contained in the EPA Predictive Model, the EPA Complex Model and the EPA Sulfur Model.

3.3 Predictive Model Effects

The EPA Predictive Fuel model is one of EPA's more recent fuel models used to predict the impact on HC and NOx emissions from varying gasoline fuel properties (i.e., six of the eleven fuel properties listed in Table 1). This model was developed by EPA

from vehicle and fuel testing done prior to calendar year 2001. The details of the testing and the statistical analysis of the results is quite complex and will not be reproduced in this document. The reader is encouraged to consult the EPA technical document EPA420-R-01-016, "Analysis of California's Request for Waiver of theReformulated Gasoline Oxygen Content Requirement for California Covered Areas" of June, 2001 for background on the Predictive Model. Appendix A of this current document also contains a listing of the MySQL code used to implement the EPA Predictive Fuel Model.

In MOVES the EPA Predictive Fuel Model was used only to model fuel effects of HC and NOx emissions. The MOVES Predictive Fuel Model differs from the standard model in that some of the effects of sulfur were removed from the model by normalizing to 30 ppm sulfur for 2001 and later model years and 90 ppm for pre-2001 model years. The MOVES sulfur effects are the MOBILE6.2 sulfur effects. These include both the short term and long term irreversible effects. For complete details see the EPA document EPA420-R-01-039 "fuel Sulfur Effects on Exhaust Emission – Recommendations for MOBILE6". The effect of High Emitters is also different between the EPA Predictive Model and the MOVES Predictive model. The EPA Predictive Model assumes a 50 percent weighting for High Emitters, and MOVES does not use the concept of High and Normal Emitters. For the purposes of this analysis, a weighting of 20 percent High Emitters was used to generate the MOVES fuel effects. This lower percentage of High Emitters better reflects the modern vehicle fleet. As for some of the RFS2 runs, (i.e., *less sensitive runs – fuel effects are a function of fuel sulfur level only for Tier1 and later model years).* The effects of all fuel parameters except sulfur were removed from the final fuel adjustment factors for Tier1 (1994 and later model years) vehicles. This was done to examine the effects of the assumption that the more advanced Tier1 and later vehicles are no longer sensitive to the effects of many of the individual fuel parameters.

3.4 Complex Model Effects

The MOVES fuel adjustment factors for CO emissions were developed from the EPA Complex Fuel Model. This model is an older model and is based on data from late 1980's and early 1990's. It was used in MOVES only for non-sulfur fuel effects for CO emissions, because it contains the most up-to-date assessment of these effects. The sulfur CO emission effects were taken from MOBILE6 and are based on the same document reference above in the Predictive Model section. For more details regarding EPA's complex model, the reader is referred to the website :

http://www.epa.gov/otaq/regs/fuels/rfg/58-11722.txt

and the document

"Regulation of Fuels and Fuel Additives: Standards for Reformulated Gasoline - Proposed Rule"

Appendix A of this current document also contains a listing of the MySQL code used to implement the EPA Complex Fuel Model for CO emissions.

3.5 Sulfur Effects

The sulfur effects were taken from the MOBILE6.2 document "Fuel Sulfur Effects on Exhaust Emissions – Recommendations for MOBILE6. – EPA420-R-01-039". These effects, based on more up to date testing, supercede the sulfur effects in the EPA Predictive and Complex Models. These effects include the short term sulfur effects, the long term effects and the effects of irreversibility. The report also included separate fuel sulfur effects for running and start processes.

4. Process Used to Create Air Toxic Adjustment Factors

The MOVES model reports air toxic emission inventories based on air toxic emission adjustment factors which are built into the MOVES model. The air toxic pollutants which are reported by Draft MOVES2009 are:

Benzene
Ethanol
MTBE
1,3 Butadiene
Formaldehyde
Acetealdehyde
Naphthalene
Acrolein

Other air toxic pollutants such as metals, and dioxin / furan compounds will likely be added in the future. The Draft MOVES2009 model reports air toxic pollutants as a function of gasoline, diesel and ethanol (E-85) fuels.

Equation 7 shows the general formula which is used to compute all the air toxic pollutants in Draft MOVES2009 except Naphthalene (which is a function of total particulate matter emissions). The equation is applied separately and with different factors for both running and start emissions. Currently, in MOVES and for the RFS 2 Rulemaking, the air toxic pollutants are a function total hydrocarbon emissions (i.e., the Hydrocarbon Emission Rate in Eq 7). In the future, all of them will be a function of

volatile organic compounds (VOC). The Hydrocarbon Emission Rate used in Equation 7 is a fully adjusted emission rate, adjusted for fuel, temperature, I/M and all other correction factors. The Air Toxic Factors are applied last in the chain of MOVES multiplicative calculations.

Air Toxic Pollutant Emission Rate = Hydrocarbon Emission Rate * Air Toxic Factor Eq 7

In Draft MOVES2009, the Air Toxic Factor differs by pollutant, process, model year, sourcetype (vehicle type) and fuel formulation for benzene, MTBE, 1,3 Butadiene, Formaldehyde and Acetaldehyde. All of the values used in MOVES for these pollutants were taken from the MOBILE6.2 model. The MOBILE6.2 air toxics model was developed from the 1993 EPA Complex Model, and is based primarily on data from 1990 and earlier model year vehicles. The extraction of data from the MOBILE6.2 model was an empirical process that involved running MOBILE6.2 over 20,000 times, and computing and averaging the ratio of the various air toxic pollutant emissions and the hydrocarbon emissions for both start and running emissions. To make the analysis manageable, ratios were computed only at eight years of age for all source types (vehicle types) and model years. The use of the ratio at eight years old for all vehicle ages could lead to some small differences between the MOVES air toxic ratios and those computed in the Complex model (MOBILE6.2 is based on the Complex model), but this difference is expected to be small.

The Air Toxic Factors for Naphthalene and Acrolein differ only by source type (vehicle type) and fuel type. These factors were also taken from MOBILE6.2 but are not factors of model year and age, and are mostly constants in the MOVES model.

The Air Toxic Factors for ethanol (shown in Table 4) are based on a literature search of new data and reports. The reports from three studies were used in the search / analysis. They are:

- Southwest Research Institute. 2007. Flex Fuel Vehicles (FFVs) VOC/PM Cold Temperature Characterization When Operating on Ethanol (E10, E70, E85). Prepared for U. S. Environmental Protection Agency. Available in Docket EPA-HQ-OAR-2005-0161

- Graham, L. A.; Belisle, S. L. and C. Baas. 2008. Emissions from light duty gasoline vehicles operating on low blend ethanol gasoline and E85. Atmos. Environ. 42: 4498-4516.

- Environment Canada. 2007. Comparison of Emissions from Conventional and Flexible Fuel Vehicles Operating on Gasoline and E85 Fuels. ERM Report No. 05-039, Emissions Research Division. Available in Docket EPA-HQ-OAR-2005-0161

Table 4 shows an average value for the air toxic to total hydrocarbon emissions ratio and the range of values used in MOVES for calendar year 2005. These are for

gasoline vehicles. The values represent the average of all 360 fuel formulations used in the calendar year 2005 fuel dataset. The relatively large standard deviations for Benzene and MTBE reflect the strong function of individual fuel formulation on the average results. Benzene and MTBE emissions are a strong function of the benzene volume, aromatic content or MTBE volume in the specific fuel.

Table 4
Typical Gasoline / Ethanol Air Toxic Ratios for Calendar Year 2005

	Gasoline Vehicles				Ethanol Vehicles
	Min AT Ratio	Avg AT Ratio	Max AT Ratio	Std Dev	E-85 Ratios
Benzene	0.032	0.050	0.086	0.0082	0.0041
MTBE	0.00	0.0017	0.018	0.0048	0.00
Naphthalene	0.088	0.088	0.088	0.00	0.086
1,3 Butadiene	0.0038	0.0055	0.0066	0.00063	0.00062
Formaldehyde	0.0097	0.013	0.016	0.0012	0.010
Acetaldehyde	0.0036	0.0070	0.013	0.0032	0.075
Acrolein	0.00061	0.00061	0.00061	0.00	0.00027
Ethanol - E0		0.00			
Ethanol - E10		0.024			
Ethanol - E85		0.484			

Table 5 shows the air toxic to THC ratios for diesel vehicles as a function of pollutant. These were taken from MOBILE6.2. They are constants due to a lack of test data. Specific diesel fuel properties (i.e., sulfur level) are currently assumed not to affect air toxic emission ratios.

Table 5
Diesel Vehicle Air Toxic Ratios

Pollutant	**Air Toxic Ratio**
Benzene	0.020
Ethanol	0.00
MTBE	0.00
Naphthalene	0.0037
1,3 Butadiene	0.0090
Formaldehyde	0.039
Acetaldehyde	0.012
Acrolein	0.0035

5. Use of the MOVES Fuel Algorithm in RFS2

The MOVE Fuel Algorithm was used in the RFS2 inventory development process to estimate the effects of fuel parameters on HC, CO and NOx emissions from light and heavy-duty gasoline vehicle applications. The fuel effects are a simple multiplicative adjustment factor applied to the base emission factors that account for the effects of various fuel properties and the distribution of particular gasoline fuel throughout the United States. Separate factors were generated for HC, CO, NOx and the evaporative process of permeation.

The analysis started with the development of three datasets of national fuels (fuel supply and fuelformulation data) according to individual county, year and month. Thus, each of the 3,222 individual United States counties had a set of specific fuels (with market shares) for each of the twelve months and for the two years of analysis interest. These years were the base year of 2005 and the control/reference year of 2022.

The base year of 2005 contained a set 261 fuel formulations that represent every fuel used in the United States in calendar year 2005. These fuels run the spectrum for all of individual fuel parameters (low to high sulfur, no ethanol to E-85, etc). Also, since the fuel adjustment factors are also a function of model year group and source type (vehicle type), the base year analysis contained a total of 51,285 different fuel adjustment factors. These had the following range of values by pollutant.

Table 6
Average Fuel Adjustment for RFS2 Base Year 2005

Pollutant / Process	Average Adjustment	Std deviation
Running HC	1.044	0.12
Start HC	1.041	0.11
Running CO	1.150	0.31
Start CO	1.128	0.28
Running NOx	1.211	0.20
Start NOx	1.211	0.20

The fairly large standard deviations illustrate the fairly large range of possible fuel adjustment factors. These factors range so much because in 2005 the individual fuel properties such as sulfur, conventional gas versus E-10, aromatics, etc., varied widely. For example, a low sulfur fuel has a lower adjustment factor versus a high sulfur fuel. Also, the average fuel adjustments shown in Table 6 do not include E-70 or E-85 fuel effects.

Table 7
Average Fuel Adjustment for RFS2 Control Year 2022
Sensitivity Case

Pollutant / Process	Average Adjustment	Std deviation
Running HC	0.94	0.05
Start HC	0.98	0.04
Running CO	0.97	0.25
Start CO	1.03	0.24
Running NOx	1.11	0.04
Start NOx	1.10	0.04

The Control fuel dataset consisted of a set of fuels that were composed of all E-10 (10 vol% ethanol). Other fuel properties varied so there are 114 individual fuels in this dataset. This dataset was developed to model the effect of 100 percent penetration of E-10 in the fleet plus additional areas with E-85 fuel. The average fuel adjustments shown in Table 7 do not include E-70 or E-85 fuel effects.

Table 8
Average Fuel Adjustment for RFS2 Reference Year 2022
Sensitivity Case

Pollutant / Process	Average Adjustment	Std deviation
Running HC	0.95	0.06
Start HC	0.99	0.04
Running CO	0.99	0.24
Start CO	1.05	0.23
Running NOx	1.10	0.05
Start NOx	1.09	0.05

The Reference fuel dataset consisted of a set of fuels that were composed of both E-10 (10 vol% ethanol) and conventional gasoline fuels in 2022. Other fuel properties varied so there are 139 individual fuels in this dataset. The majority (106 out of 139) of the fuels in this dataset were E-10 fuels. This dataset was developed to model a case where conventional gasoline fuels are also present in some degree. Ethanol 85 vol% (E-85) fuels are also present in this scenario. However, the fuel adjustments shown in Table 8 do not include E-70 or E-85 fuel effects.

For both the 2022 Control and Reference cases, an additional permutation was done that created two additional fuel datasets. This permutation was the creation of a Sensitivity and a Primary dataset. Tables 7 and 8 show the average effects from the Sensitivity case since it includes effects from all of the EPA fuel models (Complex model, Predictive model and the MOBILE6.2 Sulfur model). The Primary datasets disable the Complex and Predictive models (set the adjustment factor to unity), and contain only the MOBILE6.2 sulfur effects. The Primary datasets were created because EPA believes that modern Tier2 vehicles are less sensitive to fuel properties, with the exception of sulfur, than older vintage model years. This dataset was created to model this scenario. Table 9 shows the mean fuel adjustment by pollutant / process for both the Control and Reference datasets for the Primary case.

Table 9
Average Fuel Adjustment for RFS2 Reference and Control Year 2022
Primary Case

Pollutant / Process	Average Adjustment Control	Average Adjustment Reference
Running HC	0.95	0.95
Start HC	0.99	1.00
Running CO	0.97	0.99
Start CO	1.02	1.05
Running NOx	1.03	1.03
Start NOx	1.03	1.02

Appendix A MySQL Code for the Fuel Binner

This appendix includes example code only for the 2005 calendar year. The other calendar years are completely analogous and it would be redundant to include them here.

```
FLUSH TABLES;
drop Database IF EXISTS MOVESFuelRFS_Base;
Create Database MOVESFuelRFS_Base;
USE MOVESFuelRFS_Base;


--
**********************************************************************
****************
-- This script requires the successful completion of scrips ---

--                JAVA program NMIMFuelDataImporter.java

--
**********************************************************************
****************


--          Required databases are:

--                NMIMRFS2FuelsMSAT2005Base
--                RFS2005Base

--                NMIMRFS2FuelsMSAT2005Reference
--                RFS2005Base

--                MOVESDB20080828


--          Required external text files are:

--                NONE


--
**********************************************************************
*******************
-- MOVES FuelFormulationID Tables
--
**********************************************************************
*******************

--  create blank fuelformulation table.

drop TABLE if exists fuelformulation1;
CREATE TABLE fuelformulation1
SELECT fuelformulation.*      FROM RFS2005Base.fuelformulation
```

```
WHERE fuelformulationID > 1000000
GROUP BY fuelFormulationID;

ALTER TABLE fuelformulation1 MODIFY fuelformulationID INT(8);


-- read calendar year 2005 Base table.  This can be changed to any year
the user desires.

drop TABLE if exists Temp1;
CREATE TABLE Temp1
SELECT fuelformulation.* FROM RFS2005Base.fuelformulation
GROUP BY fuelFormulationID;

ALTER TABLE Temp1 MODIFY fuelformulationID INT(8);

INSERT INTO fuelformulation1
(     fuelFormulationID,
      fuelSubtypeID,
      RVP,
      sulfurLevel,
      ETOHVolume,
      MTBEVolume,
      ETBEVolume,
      TAMEVolume,
      aromaticContent,
      olefinContent,
      benzeneContent,
      e200,
      e300  )
SELECT
      fuelFormulationID,
      fuelSubtypeID,
      RVP,
      sulfurLevel,
      ETOHVolume,
      MTBEVolume,
      ETBEVolume,
      TAMEVolume,
      aromaticContent,
      olefinContent,
      benzeneContent,
      e200,
      e300
FROM Temp1
GROUP BY fuelFormulationID;


-- this MUST create a unique index or you have a problem!

CREATE UNIQUE INDEX  Index1 on fuelformulation1(fuelformulationID);

drop TABLE if exists Temp1;


Alter Table  fuelformulation1 ADD (
            RVPbin                        INT(5),
```

```
            Sulfurbin               INT(6),
            Etohbin                       INT(5),
            Mtbebin                       INT(5),
            Tamebin                       INT(5),
            Aromaticbin             INT(5),
            Olefinbin               INT(5),
            e200bin                       INT(5),
            e300bin                       INT(5),
            benzenebin              INT(5)
        );

UPDATE fuelformulation1 SET e200bin = 1         WHERE e200 <  46.99999;
UPDATE fuelformulation1 SET e200bin = 2         WHERE e200 >= 47.00000;

UPDATE fuelformulation1 SET e300bin = 1         WHERE e300 <  80.00000;
UPDATE fuelformulation1 SET e300bin = 2         WHERE e300 >= 80.00000
and e300 < 86.99999;
UPDATE fuelformulation1 SET e300bin = 3         WHERE e300 >= 87.00000;

UPDATE fuelformulation1 SET Aromaticbin = 1           WHERE
aromaticcontent <  20.0000;
UPDATE fuelformulation1 SET Aromaticbin = 2
                                      WHERE aromaticcontent >=
20.00000 and aromaticcontent < 30.00000;
UPDATE fuelformulation1 SET Aromaticbin = 3           WHERE
aromaticcontent >= 30.00000;


UPDATE fuelformulation1 SET Olefinbin = 1       WHERE olefincontent <
8.0000;
UPDATE fuelformulation1 SET Olefinbin = 2
                                      WHERE olefincontent >=
8.00000 and olefincontent < 10.00000;
UPDATE fuelformulation1 SET Olefinbin = 3       WHERE olefincontent >=
10.00000;


UPDATE fuelformulation1 SET RVPbin = 1          WHERE RVP >  0.0
and RVP <=7.2000;
UPDATE fuelformulation1 SET RVPbin = 2          WHERE RVP >  7.20001
and RVP <=8.2000;
UPDATE fuelformulation1 SET RVPbin = 3          WHERE RVP >  8.20001
and RVP <=8.9999;
UPDATE fuelformulation1 SET RVPbin = 4          WHERE RVP >  8.99999
and RVP <=9.9999;
UPDATE fuelformulation1 SET RVPbin = 5          WHERE RVP >  9.99999
and RVP <=11.400;
UPDATE fuelformulation1 SET RVPbin = 6          WHERE RVP >  11.40001
and RVP <=13.400;
UPDATE fuelformulation1 SET RVPbin = 7          WHERE RVP >  13.40001
and RVP <=14.900;
UPDATE fuelformulation1 SET RVPbin = 8          WHERE RVP >  14.90001;


UPDATE fuelformulation1 SET Sulfurbin = 1 WHERE sulfurLevel >=  0.0000
and sulfurLevel <=  10.000;
```

```
UPDATE fuelformulation1 SET Sulfurbin = 2 WHERE sulfurLevel >=  10.001
and sulfurLevel <=  25.000;
UPDATE fuelformulation1 SET Sulfurbin = 3 WHERE sulfurLevel >=  25.001
and sulfurLevel <=  35.000;
UPDATE fuelformulation1 SET Sulfurbin = 4 WHERE sulfurLevel >=  35.001
and sulfurLevel <=  70.000;
UPDATE fuelformulation1 SET Sulfurbin = 5 WHERE sulfurLevel >=  70.001
and sulfurLevel <= 120.000;
UPDATE fuelformulation1 SET Sulfurbin = 6       WHERE sulfurLevel >=
120.001 and sulfurLevel <= 210.000;
UPDATE fuelformulation1 SET Sulfurbin = 7 WHERE sulfurLevel >= 210.001
and sulfurLevel <= 300.000;
UPDATE fuelformulation1 SET Sulfurbin = 8 WHERE sulfurLevel >= 300.001
and sulfurLevel <= 500.000;
UPDATE fuelformulation1 SET Sulfurbin = 9 WHERE sulfurLevel >= 500.001;


-- Per input from Rich Cook.  These benzene bins were established.

UPDATE fuelformulation1 SET   Benzenebin = 3;

UPDATE fuelformulation1 SET   Benzenebin = 1    WHERE benzeneContent >=
0.0000 and benzeneContent <= 0.40;
UPDATE fuelformulation1 SET   Benzenebin = 2    WHERE benzeneContent >=
0.4001 and benzeneContent <= 0.80;
UPDATE fuelformulation1 SET   Benzenebin = 3    WHERE benzeneContent >=
0.8001 and benzeneContent <= 2.00;
UPDATE fuelformulation1 SET   Benzenebin = 4    WHERE benzeneContent >=
2.0001;




-- add sulfur bins for diesel fuels.  Diesel has only sulfur bins.

UPDATE fuelformulation1 SET Sulfurbin = 21
            WHERE sulfurLevel >=  0.0000 and sulfurLevel <  5.000 and
fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 22
            WHERE sulfurLevel >=  5.0000 and sulfurLevel <  15.000 and
fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 23
            WHERE sulfurLevel >=  15.0000 and sulfurLevel <  50.000 and
fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 24
            WHERE sulfurLevel >=  50.0000 and sulfurLevel <  100.000
and fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 25
            WHERE sulfurLevel >=  100.0000 and sulfurLevel < 200.000
and fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 26
```

```
            WHERE sulfurLevel >=  200.0000 and sulfurLevel < 300.000
and fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 27
            WHERE sulfurLevel >=  300.0000 and sulfurLevel < 400.000
and fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 28
            WHERE sulfurLevel >=  400.0000 and sulfurLevel < 500.000
and fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 29
            WHERE sulfurLevel >=  500.0000 and sulfurLevel < 1000.000
and fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 30
            WHERE sulfurLevel >=  1000.0000 and sulfurLevel < 2000.000
and fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 31
            WHERE sulfurLevel >=  2000.0000 and sulfurLevel < 5000.000
and fuelsubtypeid=20;

UPDATE fuelformulation1 SET Sulfurbin = 32
            WHERE sulfurLevel >=  5000.0000  and fuelsubtypeid=20;



UPDATE fuelformulation1 SET Etohbin = 1         WHERE ETOHVolume <=
0.00001;
UPDATE fuelformulation1 SET Etohbin = 2         WHERE ETOHVolume >
0.00001 and ETOHVolume <= 6.0000;
UPDATE fuelformulation1 SET Etohbin = 3         WHERE ETOHVolume >=
6.00001 and ETOHVolume <= 8.0000;
UPDATE fuelformulation1 SET Etohbin = 4         WHERE ETOHVolume >=
8.00001;


UPDATE fuelformulation1 SET Mtbebin = 1         WHERE MTBEVolume <=
0.00001;
UPDATE fuelformulation1 SET Mtbebin = 2         WHERE MTBEVolume >
0.00001 and MTBEVolume <= 5.0000;
UPDATE fuelformulation1 SET Mtbebin = 3         WHERE MTBEVolume >=
5.00001 and MTBEVolume <= 10.000;
UPDATE fuelformulation1 SET Mtbebin = 4         WHERE MTBEVolume >=
10.0001;


UPDATE fuelformulation1 SET Tamebin =  1        WHERE TAMEVolume  =
0.00;
UPDATE fuelformulation1 SET Tamebin =  2        WHERE TAMEVolume  >
0.00    and TAMEVolume < 0.03000;
UPDATE fuelformulation1 SET Tamebin =  3        WHERE TAMEVolume >=
0.03001;
```

```
--    Average values within a bin are computed for the Air Toxics fuel
parameters.
--          This is done just to give these a placeholder in the new
database.
--          Eventually, better values will have to be inserted into the
MOVES database
--          in-order to model toxic pollutants.


--    For this case DROP the diesel fuels.  They are not needed for the
RFS2 rulemaking.

drop TABLE if exists Scratch;
CREATE TABLE Scratch
SELECT      fuelformulationID,
            fuelsubtypeid,
            RVP,
            sulfurLevel,
            ETOHVolume,
            MTBEVolume,
            ETBEVolume,
            TAMEVolume,
            aromaticContent,
            olefinContent,
            benzeneContent,
            e200,
            e300,
            rvpbin,
            sulfurbin,
            tamebin,
            etohbin,
            mtbebin,
            e200bin,
            e300bin,
            aromaticbin,
            olefinbin,
            benzenebin,
            count(*) as cnt
FROM  fuelformulation1
Where             fuelsubtypeid IN (10, 11, 12, 13, 14)
Group by    fuelformulationID;



UPDATE Scratch SET      benzeneContent = NULL;
UPDATE Scratch SET      benzeneContent = 0.35         WHERE benzenebin
= 1;
UPDATE Scratch SET      benzeneContent = 0.65         WHERE benzenebin
= 2;
UPDATE Scratch SET      benzeneContent = 1.50         WHERE benzenebin
= 3;
UPDATE Scratch SET      benzeneContent = 3.50         WHERE benzenebin
= 4;


UPDATE Scratch SET  olefinContent = NULL;
```

```
UPDATE Scratch SET  olefinContent =  5.6        WHERE olefinbin = 1;
UPDATE Scratch SET  olefinContent =  9.2        WHERE olefinbin = 2;
UPDATE Scratch SET  olefinContent = 11.9        WHERE olefinbin = 3;

UPDATE Scratch SET  aromaticContent = NULL;
UPDATE Scratch SET  aromaticContent =  17.5          WHERE aromaticbin
= 1;
UPDATE Scratch SET  aromaticContent =  26.1          WHERE aromaticbin
= 2;
UPDATE Scratch SET  aromaticContent =  32.0          WHERE aromaticbin
= 3;

UPDATE Scratch SET  e200 = NULL;
UPDATE Scratch SET  e200 = 41.0          WHERE e200bin = 1;
UPDATE Scratch SET  e200 = 50.0          WHERE e200bin = 2;

UPDATE Scratch SET  e300 = NULL;
UPDATE Scratch SET  e300 = 78.6          WHERE e300bin = 1;
UPDATE Scratch SET  e300 = 83.0          WHERE e300bin = 2;
UPDATE Scratch SET  e300 = 89.1          WHERE e300bin = 3;


UPDATE Scratch SET      RVP = NULL;
UPDATE Scratch SET      RVP = 6.90       WHERE      RVPbin = 1;
UPDATE Scratch SET      RVP = 7.50       WHERE      RVPbin = 2;
UPDATE Scratch SET      RVP = 8.70       WHERE      RVPbin = 3;
UPDATE Scratch SET      RVP = 9.20       WHERE      RVPbin = 4;
UPDATE Scratch SET      RVP = 10.0       WHERE      RVPbin = 5;
UPDATE Scratch SET      RVP = 11.5       WHERE      RVPbin = 6;
UPDATE Scratch SET      RVP = 13.5       WHERE      RVPbin = 7;
UPDATE Scratch SET      RVP = 15.0       WHERE      RVPbin = 8;



UPDATE Scratch SET      sulfurLevel =     NULL;
UPDATE Scratch SET      sulfurLevel =       5.0      WHERE sulfurBin =
1;
UPDATE Scratch SET      sulfurLevel =      15.0      WHERE sulfurBin =
2;
UPDATE Scratch SET      sulfurLevel =      30.0      WHERE sulfurBin =
3;
UPDATE Scratch SET      sulfurLevel =      50.0      WHERE sulfurBin =
4;
UPDATE Scratch SET      sulfurLevel =      90.0      WHERE sulfurBin =
5;
UPDATE Scratch SET      sulfurLevel =     180.0      WHERE sulfurBin =
6;
UPDATE Scratch SET      sulfurLevel =     280.0      WHERE sulfurBin =
7;
UPDATE Scratch SET      sulfurLevel =     400.0      WHERE sulfurBin =
8;
UPDATE Scratch SET      sulfurLevel =     600.0      WHERE sulfurBin =
9;


UPDATE Scratch SET      sulfurLevel =       4.0      WHERE sulfurBin =
21;
```

```
UPDATE Scratch SET      sulfurLevel =      11.0       WHERE sulfurBin =
22;
UPDATE Scratch SET      sulfurLevel =      43.0       WHERE sulfurBin =
23;
UPDATE Scratch SET      sulfurLevel =      75.0       WHERE sulfurBin =
24;
UPDATE Scratch SET      sulfurLevel =     113.0       WHERE sulfurBin =
25;

UPDATE Scratch SET      sulfurLevel =     281.0       WHERE sulfurBin =
26;
UPDATE Scratch SET      sulfurLevel =     337.0       WHERE sulfurBin =
27;
UPDATE Scratch SET      sulfurLevel =     468.0       WHERE sulfurBin =
28;
UPDATE Scratch SET      sulfurLevel =     750.0       WHERE sulfurBin =
29;
UPDATE Scratch SET      sulfurLevel =  1500.0         WHERE sulfurBin =
30;
UPDATE Scratch SET      sulfurLevel =  3000.0         WHERE sulfurBin =
31;
UPDATE Scratch SET      sulfurLevel =  6000.0         WHERE sulfurBin =
32;


UPDATE Scratch SET      ETOHVolume =      NULL;
UPDATE Scratch SET      ETOHVolume =       0.00       WHERE Etohbin =
1;
UPDATE Scratch SET      ETOHVolume =       5.00       WHERE Etohbin =
2;
UPDATE Scratch SET      ETOHVolume =       8.00       WHERE Etohbin =
3;
UPDATE Scratch SET      ETOHVolume =      10.00       WHERE Etohbin =
4;


UPDATE Scratch SET      MTBEVolume =      NULL;
UPDATE Scratch SET      MTBEVolume =       0.00       WHERE MTBEbin =
1;
UPDATE Scratch SET      MTBEVolume =       2.50       WHERE MTBEbin =
2;
UPDATE Scratch SET      MTBEVolume =       8.00       WHERE MTBEbin =
3;
UPDATE Scratch SET      MTBEVolume =      11.00       WHERE MTBEbin =
4;


UPDATE Scratch SET      TAMEVolume =      NULL;
UPDATE Scratch SET      TAMEVolume =       0.000           WHERE
      TAMEbin =  1;
UPDATE Scratch SET      TAMEVolume =       0.015           WHERE
      TAMEbin =  2;
UPDATE Scratch SET      TAMEVolume =       0.060           WHERE
      TAMEbin =  3;
```

```
--    fuelsubtype  10 = Conventional Gasoline
--    fuelsubtype  11 = RFG
--    fuelsubtype  12 = Ethanol/Gasahol  Maximum ethanol 10% by volume
--    fuelsubtype  13   = Ethanol/Gasahol  Maximum ethanol  8% by
volume
--    fuelsubtype  14   = Ethanol/Gasahol  Maximum ethanol  5% by
volume
--    fuelsubtype  20 = Diesel



-- original data tables did not have fuelsubtypeID coded as gasahol.
--          this binner sets all gasoline fuels with ethanol > 5% as
gasahol.

UPDATE Scratch SET  fuelsubtypeID  = 12              WHERE
      ETOHVolume >= 10.0;
UPDATE Scratch SET  fuelsubtypeID  = 13              WHERE
      ETOHVolume Between 7.9 and 8.1;
UPDATE Scratch SET  fuelsubtypeID  = 14              WHERE
      ETOHVolume Between 4.9 and 5.1;


Alter Table  Scratch    ADD (
            FFIDD       BIGINT(19)
       );

UPDATE Scratch    SET   FFIDD =     fuelsubtypeID      + RVPbin*100 +
sulfurBin*1000 +
                                        Etohbin*100000  +
MTBEbin*1000000 +     TAMEbin*10000000  +
                                        e200bin*100000000 +
e300bin*1000000000 + Aromaticbin*10000000000 +
                                        olefinbin*100000000000 +
benzenebin*1000000000000
                             WHERE fuelsubtypeid IN (10, 11, 12, 13,
14);

UPDATE Scratch  SET FFIDD = -10*sulfurbin WHERE fuelsubtypeid = 20;


CREATE        INDEX  Index1 on Scratch(ffIDD);
CREATE UNIQUE INDEX  Index2 on Scratch(fuelformulationID);



drop TABLE if exists Scratch2;
CREATE TABLE Scratch2
Select FFIDD
FROM Scratch
WHERE FFIDD IS NOT NULL
GROUP BY FFIDD;

ALTER Table Scratch2 ADD FFIDDD INT Auto_Increment NOT NULL Primary
Key;
```

```
drop TABLE if exists NMIMMasterFuelFormulation;
CREATE TABLE NMIMMasterFuelFormulation
SELECT      Scratch.*,
            Scratch2.FFIDDD
FROM  Scratch LEFT JOIN Scratch2
ON          Scratch.FFIDD = Scratch2.FFIDD
WHERE Scratch.FFIDD IS NOT NULL;


-- FFID is the new fuelformulationID variable. All real values are
greater than 100.
--          Diesel fuels have values over 2000 to distinguish them.

-- FFIDD is the BIGINT value that forms a unique variable for all
combinations of the
--          individual binning variables.

-- FFIDDD is the autoincrement of FFIDD.  It runs from 101 to XXXX.
This is the current number
--          of unique fuels.


Alter Table  NMIMMasterFuelFormulation    ADD (
            FFID        INT(6)
       );

UPDATE NMIMMasterFuelFormulation    SET         FFID = FFIDDD + 2000

                              WHERE fuelsubtypeid = 20;


UPDATE NMIMMasterFuelFormulation    SET         FFID = FFIDDD + 99
                              WHERE       fuelsubtypeid IN (10, 11, 12,
13, 14);



CREATE UNIQUE INDEX  Index1 on NMIMMasterFuelFormulation
(fuelformulationID);
CREATE        INDEX  Index2 on NMIMMasterFuelFormulation (ffID);



--
**********************************************************************
******************
-- Create the MASTER fuelformulationID for the RFS rule.  This will be
the one specified in
--          the RFS 1 rulemaking.
--
**********************************************************************
******************

--          There will be two Master fuelformulationIDs in MOVES.
--          Fuel One is the 30 ppm sulfur fuel that represents the
2001+ model years.
```

```
--          Fuel Two is the 90 ppm sulfur fuel that represents pre-2001
model years.

--          fuelFormulationID = 98 is the 30 ppm sulfur master fuel.
--          fuelFormulationID = 99 is the 90 ppm sulfur master fuel.

--          Randomly grab two fuelformulationIDs and change them to the
properties that are
--          required. A diesel fuel was selected because most of the
properties are currently
--          NULL and consequently will not need to be changed.
FuelFormulationIDs = 98 and 99
--          will be the master fuelformulationIDs for all of the
subsequent scripts.  They will
--          may not actually exist in any USA county, but will
represent a fueladjustment factor
--          of unity for the respective model year groups.


drop TABLE if exists MasterFuel;
CREATE  TABLE MasterFuel
SELECT * FROM NMIMMasterFuelFormulation WHERE  FFID IN (101, 102);



--************************************************
--    Master FuelFormulationID #1         30    ppm sulfur *
--************************************************

UPDATE      MasterFuel  SET   fuelformulationID       =     99998 WHERE
FFID = 101;
UPDATE      MasterFuel  SET   fuelsubtypeID           =     10
      WHERE FFID = 101;
UPDATE      MasterFuel  SET   RVP                     =     6.9
      WHERE FFID = 101;
UPDATE      MasterFuel  SET sulfurLevel               =     30
      WHERE FFID = 101;

-- The base fuel has no oxygen so ETOHVolume, MTBEvolume, ETBEVolume
and TAMEVolume will
--                always be zero.

UPDATE      MasterFuel  SET   ETOHVolume              =     0.0
      WHERE FFID = 101;
UPDATE      MasterFuel  SET   MTBEVolume              =     0.0
      WHERE FFID = 101;
UPDATE      MasterFuel  SET   ETBEVolume              =     0.0
      WHERE FFID = 101;
UPDATE      MasterFuel  SET   TAMEVolume              =     0.0
      WHERE FFID = 101;
UPDATE      MasterFuel  SET   aromaticContent         =     26.1  WHERE
FFID = 101;
UPDATE      MasterFuel  SET   olefinContent           =     5.6
      WHERE FFID = 101;
UPDATE      MasterFuel  SET   benzeneContent          =     1.0
      WHERE FFID = 101;
```

```
-- E200     =     147.91 - 0.49 * T50
-- E300 =   155.47 - 0.22 * T90

-- Reference Fuel T50 = 218
-- Reference Fuel T90 = 329


UPDATE      MasterFuel  SET   e200  =     147.91 - 0.49 * 218.    WHERE
FFID = 101;
UPDATE      MasterFuel  SET   e300  =     155.47 - 0.22 * 329.    WHERE
FFID = 101;
UPDATE      MasterFuel  SET   FFID  =     98
      WHERE FFID = 101;


--************************************************
--    Master FuelFormulationID #2         90    ppm sulfur *
--************************************************

UPDATE      MasterFuel  SET   fuelformulationID       =     99999 WHERE
FFID = 102;
UPDATE      MasterFuel  SET   fuelsubtypeID           =     10
      WHERE FFID = 102;
UPDATE      MasterFuel  SET   RVP                     =     6.9
      WHERE FFID = 102;
UPDATE      MasterFuel  SET sulfurLevel               =     90
      WHERE FFID = 102;

-- The base fuel has no oxygen so ETOHVolume, MTBEvolume, ETBEVolume
and TAMEVolume will
--                always be zero.

UPDATE      MasterFuel  SET   ETOHVolume              =     0.0
      WHERE FFID = 102;
UPDATE      MasterFuel  SET   MTBEVolume              =     0.0
      WHERE FFID = 102;
UPDATE      MasterFuel  SET   ETBEVolume              =     0.0
      WHERE FFID = 102;
UPDATE      MasterFuel  SET   TAMEVolume              =     0.0
      WHERE FFID = 102;
UPDATE      MasterFuel  SET   aromaticContent         =     26.1  WHERE
FFID = 102;
UPDATE      MasterFuel  SET   olefinContent           =     5.6
      WHERE FFID = 102;
UPDATE      MasterFuel  SET   benzeneContent          =     1.0
      WHERE FFID = 102;

UPDATE      MasterFuel  SET   e200  =     147.91 - 0.49 * 218.    WHERE
FFID = 102;
UPDATE      MasterFuel  SET   e300  =     155.47 - 0.22 * 329.    WHERE
FFID = 102;
UPDATE      MasterFuel  SET   FFID  =     99
      WHERE FFID = 102;
```

```
--
**************************************************************************
********************
-- NOTES on the Reference fuel


--                fuelsubtypeid = 10
--                RFG = 'N' (conventional)
--                RVP = 6.9
--                Sulfur = 90
--                ETOH = 0.0
--                Tame = 0.0
--                MTBE = 0.0
--                aromaticcontent = 26.1
--                olefincontent = 5.6
--                e200 = 50.0
--                e300 = 83.0
--                benzeneContent = 0.80

--
--          This fuel best matches the fuel used in the Arizona test
program
--          in the spring,summer and fall.

--          The summer RVP is used instead of the winter one of 11.
--          This will minimize (the possibly dubious MOBILE6) RVP
correction factors in MOVES.


--
**************************************************************************
********************




drop TABLE if exists FuelFormulation;
CREATE TABLE FuelFormulation  (
      fuelFormulationID       smallint(6)       NOT NULL    default
'0',
      fuelSubTypeID                 smallint(6)       NOT NULL
      default '0',
      RVP                                 float
      default NULL,
      sulfurLevel                   float
      default NULL,
      ETOHVolume                    float
      default NULL,
      MTBEVolume                    float
      default NULL,
      ETBEVolume                    float
      default NULL,
      TAMEVolume                    float
      default NULL,
      aromaticContent               float
      default NULL,
```

```
	olefinContent			float
	default NULL,
	benzeneContent			float
	default NULL,
	e200				float
	default NULL,
	e300				float
	default NULL,
	volToWtPercentOxy	float
	default NULL,
	PRIMARY KEY			(fuelFormulationID)

);


INSERT INTO fuelformulation
(	fuelFormulationID,
	fuelSubtypeID,
	RVP,
	sulfurLevel,
	ETOHVolume,
	MTBEVolume,
	ETBEVolume,
	TAMEVolume,
	aromaticContent,
	olefinContent,
	benzeneContent,
	e200,
	e300,
	volToWtPercentOxy )
SELECT
	FFID,
	fuelSubtypeID,
	RVP,
	sulfurLevel,
	ETOHVolume,
	MTBEVolume,
	ETBEVolume,
	TAMEVolume,
	aromaticContent,
	olefinContent,
	benzeneContent,
	e200,
	e300,
	NULL as volToWtPercentOxy
FROM	MasterFuel
GROUP BY	FFID;



INSERT INTO fuelformulation
(	fuelFormulationID,
	fuelSubtypeID,
	RVP,
	sulfurLevel,
	ETOHVolume,
	MTBEVolume,
```

```
      ETBEVolume,
      TAMEVolume,
      aromaticContent,
      olefinContent,
      benzeneContent,
      e200,
      e300,
      volToWtPercentOxy )
SELECT
      FFID,
      fuelSubtypeID,
      RVP,
      sulfurLevel,
      ETOHVolume,
      MTBEVolume,
      ETBEVolume,
      TAMEVolume,
      aromaticContent,
      olefinContent,
      benzeneContent,
      e200,
      e300,
      NULL as volToWtPercentOxy
FROM        NMIMMasterFuelFormulation
GROUP BY    FFID;



-- copy all the non gasoline and non diesel fuelformulations from the
default MOVES program
--          into the fuelformulation table.

INSERT INTO fuelformulation
      ( fuelFormulationID, fuelSubtypeID, RVP, sulfurLevel, ETOHVolume,
MTBEVolume,
        ETBEVolume, TAMEVolume, aromaticContent, olefinContent,
benzeneContent, e200, e300,
        volToWtPercentOxy )
SELECT            fuelFormulationID, fuelSubtypeID, RVP, sulfurLevel,
ETOHVolume, MTBEVolume,
        ETBEVolume, TAMEVolume, aromaticContent, olefinContent,
benzeneContent, e200, e300,
        volToWtPercentOxy
FROM        MOVESDB20080828.fuelformulation
WHERE       fuelsubtypeID >= 29
GROUP BY    fuelFormulationID;

INSERT INTO fuelformulation
      ( fuelFormulationID, fuelSubtypeID, RVP, sulfurLevel, ETOHVolume,
MTBEVolume,
        ETBEVolume, TAMEVolume, aromaticContent, olefinContent,
benzeneContent, e200, e300,
        volToWtPercentOxy )
SELECT            fuelFormulationID, fuelSubtypeID, RVP, sulfurLevel,
ETOHVolume, MTBEVolume,
```

```
        ETBEVolume, TAMEVolume, aromaticContent, olefinContent,
benzeneContent, e200, e300,
        volToWtPercentOxy
FROM        MOVESDB20080828.fuelformulation
WHERE       fuelformulationID IN (10, 20)
GROUP BY    fuelFormulationID;


UPDATE fuelformulation  SET   volToWtPercentOxy = NULL;


CREATE UNIQUE INDEX  Index1 on FuelFormulation (fuelformulationID);


--drop TABLE if exists Scratch;
--drop TABLE if exists Scratch2;




--
**************************************************************************
*******************
-- Fuel Supply Tables
--
**************************************************************************
*******************

--  create blank Fuel Supply table.

drop TABLE if exists fs;
CREATE TABLE fs
SELECT fuelsupply.*
FROM RFS2005Base.fuelsupply
WHERE fuelformulationID > 1000000
GROUP BY    countyID, fuelyearID, monthgroupID, fuelformulationID;

ALTER TABLE fs MODIFY fuelformulationID INT(8);


-- read calendar year 2005 table.  This can be changed to any year the
user desires.

drop TABLE if exists Temp1;
CREATE TABLE Temp1
SELECT fuelsupply.*
FROM RFS2005Base.fuelsupply
GROUP BY    countyID, fuelyearID, monthgroupID, fuelformulationID;

ALTER TABLE Temp1 MODIFY fuelformulationID INT(8);

INSERT INTO fs
(     countyID,
      fuelYearID,
      monthGroupID,
      fuelFormulationID,
      marketShare,
```

```
      marketShareCV     )
SELECT
      countyID,
      fuelYearID,
      monthGroupID,
      fuelFormulationID,
      marketShare,
      marketShareCV
FROM Temp1
GROUP BY    countyID, fuelyearID, monthgroupID, fuelformulationID;


-- this must create a unique index or problems are apparent.

CREATE UNIQUE INDEX  Index1 on fs  (countyID, fuelyearID, monthgroupID,
fuelformulationID);
CREATE        INDEX  Index2 on fs  (fuelformulationID);




drop TABLE if exists fs1 CASCADE;
CREATE TABLE fs1
SELECT      fs.*,
            NMIMMasterFuelFormulation.FFID
FROM  fs LEFT JOIN NMIMMasterFuelFormulation
ON          fs.fuelformulationID =
NMIMMasterFuelFormulation.fuelformulationID
WHERE fs.fuelformulationID > 0
GROUP BY    countyID, fuelyearID, monthgroupID, fuelformulationID;

CREATE UNIQUE     INDEX  Index1 on fs1(countyID, fuelyearID,
monthgroupID, fuelformulationID);
CREATE            INDEX  Index2 on fs1(ffID);



drop TABLE if exists fsNEW CASCADE;
CREATE TABLE fsNEW
SELECT      fs1.countyID, fs1.fuelYearID, fs1.monthGroupID, fs1.FFID as
fuelFormulationID,
            fs1.marketShare, fs1.marketShareCV
FROM  fs1;

CREATE UNIQUE INDEX  Index1 on fsNEW(countyID, fuelyearID,
monthgroupID, fuelFormulationID);
CREATE        INDEX  Index2 on fsNEW(fuelFormulationID);




-- FuelSupply is one of the final products of the fuel binner scripts.
It goes
--          directly into the MOVES model.

drop TABLE if exists FuelSupply;
```

```
CREATE TABLE FuelSupply  (
      countyID                      int(11)                NOT NULL,
      fuelYearID                    smallint(6)       NOT NULL,
      monthGroupID                  smallint(6)       NOT NULL,
      fuelFormulationID       smallint(6)       NOT NULL,
      marketShare                   float,
      marketShareCV                 float,
      PRIMARY KEY                   (countyID, fuelYearID,
monthGroupID, fuelFormulationID)

);



INSERT INTO FuelSupply (CountyID, FuelYearID, MonthGroupID,
FuelFormulationID,
                                          marketShare, marketShareCV)
SELECT CountyID, FuelYearID, MonthGroupID, FuelFormulationID,
sum(marketshare), null
FROM  fsNEW
GROUP BY CountyID, FuelYearID, MonthGroupID, FuelFormulationID;




CREATE INDEX  Index1 on
      FuelSupply(countyID, fuelyearID, monthgroupID,
FuelFormulationID);
CREATE INDEX  Index2 on
      FuelSupply(fuelformulationID);


-- remove a Colorado county that exists only after CY 2001.  MOVES2006
cannot handle this
--                this county division.

DELETE FROM fuelsupply where countyID = 8014;


select countyid, sum(marketshare) FROM fuelsupply as A, fuelformulation
as B
      WHERE       A.fuelformulationID = B.fuelformulationID  and
                  B.fuelsubtypeID < 20
group by countyid, fuelyearid, monthgroupID
HAVING Sum(marketshare) > 1.01 or Sum(marketshare) < 0.99 limit 100;



--drop TABLE if exists fs;
--drop TABLE if exists fs1;
--drop TABLE if exists temp1;
--drop TABLE if exists FSnew;
```

Appendix B MySQL Code for the Predictive Model

This appendix contains MySQL code for the EPA Predictive Model. The RFS Project included several versions of the Predictive model based on base fuel levels and calendar years. Only the 2005 calendar year example is provided so as to save space. All of the Predictive models are structurally analogous.

```
FLUSH TABLES;
drop Database IF EXISTS RFSPredictive30_Base;
Create Database RFSPredictive30_Base;
USE RFSPredictive30_Base;


--
**************************************************************************
***************
-- This script requires the successful completion of scrips ---

--                MOVEFuelBinnerRFS_Base.sql

--
**************************************************************************
***************


--          Required databases are:

--                MOVESFuelRFS_Base
--                MOVESDB20080828


--          Required external text files are:

--                'C:\\Eds C Files\\MOVES FUELBinner\\SQL Fuel Binner
Code\\SulfurCoefficients.csv'


--
**************************************************************************
******************
----------------------------------------------------------------------------
---------------------

--
**************************************************************************
******************
--
**************************************************************************
******************
```

```
--    This is the September 16, 2008 version of this script.

--          This version of the Predictive model uses 30 ppm as the
reference sulfur.  This
--          corresponds to fuelformulationID = 98.

--          The Predictive Model is used only to model HC and NOx
pollutant effects.
--          It is used to model all the fuel effects for these two
pollutants except Sulfur.
--          Sulfur effects were removed from the EPA Predictive model
for the the MOVES exercise.
--          The sulfur effects are now at the end of this script and
are based directly on
--          MOBILE6.2 algorithms.

--          Creating a new Predictive Model Table and populating it
with real values.

--          PollutantID 1 is HC,
--          PollutantID 3 is NOx

--          The predictive model is the average result of three HC
regression models and
--          six NOx regression models.



drop TABLE if exists PredictiveModelCoeff1;
CREATE TABLE PredictiveModelCoeff1  (
      pollutantID                   smallint(6),
      predModelID                   smallint(6),
      predCoefficientID       smallint(6),
      predCoefficientName           CHAR(12),
      predCoefficient               float,
      PRIMARY KEY                   (pollutantID, predModelID,
predCoefficientID)
);


INSERT INTO PredictiveModelCoeff1
      (pollutantID, predModelID, predCoefficientID,
predCoefficientName, predCoefficient) VALUES
            (1, 1, 1,  'Intercept',       -1.5957),
            (1, 1, 2,  'RVP',             0.008474),
            (1, 1, 3,  'T50',             0.06125),
            (1, 1, 4,  'T90',             0.02084),
            (1, 1, 5,  'AROM',                  0.008729),
            (1, 1, 6,  'OLEF',                  -0.01426),
            (1, 1, 7,  'OXYGEN',          -0.01329),
            (1, 1, 8,  'SULFUR',          0.05505),
            (1, 1, 9,  'HI-EMIT',         1.6909),
            (1, 1, 10, 'T90*T90',         0.01617),
            (1, 1, 11, 'T50*T50',         0.02494),
            (1, 1, 12, 'T90*OXY',         0.01589),
            (1, 1, 13, 'SUL*HI',          -0.03174),
```

```
(1, 1, 14, 'OXY*OXY',         0.01256),
(1, 1, 15, 'T90*ARO',         0.006908),
(1, 1, 16, 'T50*HI',          0.000),
(1, 1, 17, 'OLE*OLE',         0.000),
(1, 1, 18, 'T90*OLE',         0.000),
(1, 1, 19, 'ARO*ARO',         0.000),
      (1, 2, 1,  'Intercept',       -1.5980),
      (1, 2, 2,  'RVP',             0.008971),
      (1, 2, 3,  'T50',             0.06499),
      (1, 2, 4,  'T90',             0.02104),
      (1, 2, 5,  'AROM',                  0.008729),
      (1, 2, 6,  'OLEF',                  -0.01430),
      (1, 2, 7,  'OXYGEN',          -0.01378),
      (1, 2, 8,  'SULFUR',          0.05495),
      (1, 2, 9,  'HI-EMIT',         1.6935),
      (1, 2, 10, 'T90*T90',         0.01604),
      (1, 2, 11, 'T50*T50',         0.02477),
      (1, 2, 12, 'T90*OXY',         0.01576),
      (1, 2, 13, 'SUL*HI',          -0.03172),
      (1, 2, 14, 'OXY*OXY',         0.01353),
      (1, 2, 15, 'T90*ARO',         0.007013),
      (1, 2, 16, 'T50*HI',          -0.02609),
      (1, 2, 17, 'OLE*OLE',         0.000),
      (1, 2, 18, 'T90*OLE',         0.000),
      (1, 2, 19, 'ARO*ARO',         0.000),

(1, 3, 1,  'Intercept',       -1.6012),
(1, 3, 2,  'RVP',             0.007973),
(1, 3, 3,  'T50',             0.06046),
(1, 3, 4,  'T90',             0.02133),
(1, 3, 5,  'AROM',                  0.008759),
(1, 3, 6,  'OLEF',                  -0.01457),
(1, 3, 7,  'OXYGEN',          -0.01391),
(1, 3, 8,  'SULFUR',          0.04696),
(1, 3, 9,  'HI-EMIT',         1.7091),
(1, 3, 10, 'T90*T90',         0.01633),
(1, 3, 11, 'T50*T50',         0.02469),
(1, 3, 12, 'T90*OXY',         0.01552),
(1, 3, 13, 'SUL*HI',          0.000),
(1, 3, 14, 'OXY*OXY',         0.01288),
(1, 3, 15, 'T90*ARO',         0.006814),
(1, 3, 16, 'T50*HI',          0.000),
(1, 3, 17, 'OLE*OLE',         0.000),
(1, 3, 18, 'T90*OLE',         0.000),
(1, 3, 19, 'ARO*ARO',         0.000),
      (3,   1,    1,    'Intercept',-0.6603),
      (3,   1,    2,    'RVP',            0.009093),
      (3,   1,    3,    'T50',            -0.00245),
      (3,   1,    4,    'T90',            0.00719),
      (3,   1,    5,    'AROM',           0.01587),
      (3,   1,    6,    'OLEF',           0.01988),
      (3,   1,    7,    'OXYGEN',   0.01240),
      (3,   1,    8,    'SULFUR',   0.04171),
      (3,   1,    9,    'HI-EMIT',  0.3960),
      (3,   1,    10,   'OXY*SUL',  -0.01506),
      (3,   1,    11,   'OXY*T50',  0.000),
      (3,   1,    12,   'OXY*T90',  0.000),
```

```
      (3,   1,    13,   'OXY*ARO',  0.000),
      (3,   1,    14,   'OXY*OXY',  0.000),
      (3,   1,    15,   'T50*T50',  0.000),
      (3,   1,    16,   'SUL*SUL',  0.000),
      (3,   1,    17,   'SUL*T90',  0.000),
(3,   2,    1,    'Intercept',-0.6606),
(3,   2,    2,    'RVP',           0.01172),
(3,   2,    3,    'T50',           0.000084),
(3,   2,    4,    'T90' ,          0.007879),
(3,   2,    5,    'AROM',          0.01431),
(3,   2,    6,    'OLEF',          0.01949),
(3,   2,    7,    'OXYGEN',   0.01728),
(3,   2,    8,    'SULFUR',   0.04387),
(3,   2,    9,    'HI-EMIT',  0.3963),
(3,   2,    10,   'OXY*SUL',  0.000),
(3,   2,    11,   'OXY*T50',  0.000),
(3,   2,    12,   'OXY*T90',  -0.00510),
(3,   2,    13,   'OXY*ARO',  0.000),
(3,   2,    14,   'OXY*OXY',  0.000),
(3,   2,    15,   'T50*T50',  0.000),
(3,   2,    16,   'SUL*SUL',  0.000),
(3,   2,    17,   'SUL*T90',  0.000),
      (3,   3,    1,    'Intercept',-0.6656),
      (3,   3,    2,    'RVP',           0.009694),
      (3,   3,    3,    'T50',           0.001804),
      (3,   3,    4,    'T90',           0.005543),
      (3,   3,    5,    'AROM',          0.01524),
      (3,   3,    6,    'OLEF',          0.01940),
      (3,   3,    7,    'OXYGEN',   0.01333),
      (3,   3,    8,    'SULFUR',   0.04201),
      (3,   3,    9,    'HI-EMIT',  0.3965),
      (3,   3,    10,   'OXY*SUL',  -0.01647),
      (3,   3,    11,   'OXY*T50',  0.000),
      (3,   3,    12,   'OXY*T90',  0.000),
      (3,   3,    13,   'OXY*ARO',  0.000),
      (3,   3,    14,   'OXY*OXY',  0.000),
      (3,   3,    15,   'T50*T50',  0.006974),
      (3,   3,    16,   'SUL*SUL',  0.000),
      (3,   3,    17,   'SUL*T90',  0.000),
(3,   4,    1,    'Intercept',-0.6651),
(3,   4,    2,    'RVP',           0.007673),
(3,   4,    3,    'T50',           0.001173),
(3,   4,    4,    'T90',           0.006239),
(3,   4,    5,    'AROM',          0.01407),
(3,   4,    6,    'OLEF',          0.01966),
(3,   4,    7,    'OXYGEN',   0.01371),
(3,   4,    8,    'SULFUR',   0.04201),
(3,   4,    9,    'HI-EMIT',  0.3960),
(3,   4,    10,   'OXY*SUL',  -0.01627),
(3,   4,    11,   'OXY*T50',  -0.00830),
(3,   4,    12,   'OXY*T90',  0.000),
(3,   4,    13,   'OXY*ARO',  0.000),
(3,   4,    14,   'OXY*OXY',  0.000),
(3,   4,    15,   'T50*T50',  0.000),
(3,   4,    16,   'SUL*SUL',  0.000),
(3,   4,    17,   'SUL*T90',  0.000),
      (3,   5,    1,    'Intercept',-0.6624),
```

```
                (3,   5,    2,    'RVP',           0.008390),
                (3,   5,    3,    'T50',           0.000312),
                (3,   5,    4,    'T90',           0.006213),
                (3,   5,    5,    'AROM',          0.01501),
                (3,   5,    6,    'OLEF',          0.01990),
                (3,   5,    7,    'OXYGEN',   0.01351),
                (3,   5,    8,    'SULFUR',   0.04195),
                (3,   5,    9,    'HI-EMIT',  0.3961),
                (3,   5,    10,   'OXY*SUL',  -0.01402),
                (3,   5,    11,   'OXY*T50',  0.000),
                (3,   5,    12,   'OXY*T90',  0.000),
                (3,   5,    13,   'OXY*ARO',  -0.00547),
                (3,   5,    14,   'OXY*OXY',  0.000),
                (3,   5,    15,   'T50*T50',  0.000),
                (3,   5,    16,   'SUL*SUL',  0.000),
                (3,   5,    17,   'SUL*T90',  0.000),
          (3,   6,    1,    'Intercept',-0.6737),
          (3,   6,    2,    'RVP',           0.006188),
          (3,   6,    3,    'T50',           -0.00475),
          (3,   6,    4,    'T90',           0.007587),
          (3,   6,    5,    'AROM',          0.01209),
          (3,   6,    6,    'OLEF',          0.01969),
          (3,   6,    7,    'OXYGEN',   0.008245),
          (3,   6,    8,    'SULFUR',   0.04205),
          (3,   6,    9,    'HI-EMIT',  0.3969),
          (3,   6,    10,   'OXY*SUL',  -0.01325),
          (3,   6,    11,   'OXY*T50',  0.000),
          (3,   6,    12,   'OXY*T90',  0.000),
          (3,   6,    13,   'OXY*ARO',  0.000),
          (3,   6,    14,   'OXY*OXY',  0.01120),
          (3,   6,    15,   'T50*T50',  0.000),
          (3,   6,    16,   'SUL*SUL',  0.000),
          (3,   6,    17,   'SUL*T90',  0.000);


drop TABLE if exists PredictiveModelCoeff2;
CREATE TABLE PredictiveModelCoeff2  (
      pollutantID                   smallint(6),
      predCoefficientID       smallint(6),
      predCoefficientName           CHAR(12),
      predMean                      float,
      predStdDev                    float,
      PRIMARY KEY                   (pollutantID, predCoefficientID)

);


INSERT INTO PredictiveModelCoeff2
      (pollutantID, predCoefficientID, predCoefficientName, predMean,
predStdDev) VALUES
            (1, 1,  'Intercept',1.0000,   0.000000),
            (1, 2,  'RVP',          8.51, 0.781459),
            (1, 3,  'T50',          205.62,     17.612534),
            (1, 4,  'T90',          310.65,     20.869732),
            (1, 5,  'AROM',         27.64,      6.561886),
            (1, 6,  'OLEF',         6.93, 5.143184),
```

```
            (1, 7,  'OXYGEN', 1.49, 1.249356),
            (1, 8,  'SULFUR', 183.14,     143.055894),
                  (3, 1,  'Intercept',1.0000,   0.000000),
                  (3, 2,  'RVP',          8.445335,   0.780184),
                  (3, 3,  'T50',          206.815503, 17.906267),
                  (3, 4,  'T90',          312.126198, 22.099331),
                  (3, 5,  'AROM',         28.082805,  7.383169),
                  (3, 6,  'OLEF',         6.974371,   4.932872),
                  (3, 7,  'OXYGEN', 1.347629,   1.251882),
                  (3, 8,  'SULFUR', 182.060319, 140.783197);



--------------------------------------------------------------------------
---------------------
--------------------------------------------------------------------------
---------------------

--    Bring in MOVES Fuel Parameter information.
--    Each individual MOVES Gasoline Fuel is run through the Predictive
Model.

--    Diesel fuels are not processed in any way for MOVES.
--    This information was created in the Script MOVESFuelBinnerPR.sql

drop TABLE if exists FuelFormulation;
CREATE TABLE FuelFormulation
SELECT      FuelFormulationID,
            FuelSubtypeID,
            RVP,
            SulfurLevel as Sulfur,
            ETOHVolume,
            MTBEVolume,
            ETBEVolume,
            TAMEVolume,
            aromaticContent as AROM,
            olefinContent as OLEF,
            benzeneContent as BENZ,
            e200,
            e300,
            volToWtPercentOxy
FROM MOVESFuelRFS_Base.FuelFormulation
WHERE fuelSubtypeID in (10, 11, 12, 13, 14);


CREATE UNIQUE INDEX Index1 ON FuelFormulation (FuelFormulationID);


alter TABLE FuelFormulation  ADD (
      T50                    float,
      T90                    float,
      Oxygen                 float,
      PollutantID       smallint(6)
);
```

```
--  The Predictive model requires fuel Oxygen and T50 and T90
parameters.  The MOVES
--          model contains ETOH and MTBE Volume data and e200 and e300
information.
--          These parameters must be converted.

UPDATE FuelFormulation SET    Oxygen = 0.0
                  WHERE ETOHVolume <= 0.0 and MTBEVolume <= 0.0;

UPDATE FuelFormulation SET    Oxygen = ETOHVolume / 2.900
                  WHERE ETOHVolume > 0.0 and MTBEVolume <= 0.0;

UPDATE FuelFormulation SET    Oxygen = MTBEVolume / 5.600
                  WHERE ETOHVolume <= 0.0 and MTBEVolume > 0.0;

UPDATE FuelFormulation SET    T50 = (147.91/0.49) - (e200/0.49);
UPDATE FuelFormulation SET    T90 = (155.47/0.22) - (e300/0.22);



-- FuelFormulationOriginal saves the orginal data (sulfur levels) for
future use.

drop TABLE if exists FuelFormulationOriginal;
CREATE TABLE FuelFormulationOriginal SELECT * FROM FuelFormulation;

CREATE UNIQUE INDEX Index1 ON FuelFormulationOriginal
(FuelFormulationID);




-- Set all Sulfur levels to 30.0 ppm Sulfur.  This is done to remove
the Predictive
--          Model Sulfur effects.  The Complex model sulfur effects
will be used in MOVES.


UPDATE FuelFormulation SET    Sulfur = 30.0;


--
**********************************************************************
******************
-- The first table join does HC.
--
**********************************************************************
******************

UPDATE FuelFormulation SET PollutantID    =  1;



drop TABLE if exists S1;
CREATE TABLE S1
```

```
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.RVP,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.RVP    - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as RVPdiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 2
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S1 (FuelFormulationID, PollutantID);


drop TABLE if exists S2;
CREATE TABLE S2
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.T50,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.T50    - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as T50diff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 3
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S2 (FuelFormulationID, PollutantID);


drop TABLE if exists S3;
CREATE TABLE S3
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.T90,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.T90    - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as T90diff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 4
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S3 (FuelFormulationID, PollutantID);



drop TABLE if exists S4;
CREATE TABLE S4
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
```

```
            FuelFormulation.AROM ,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.AROM   - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as AROMdiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 5
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S4 (FuelFormulationID, PollutantID);



drop TABLE if exists S5;
CREATE TABLE S5
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.OLEF ,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.OLEF   - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as OLEFdiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 6
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S5 (FuelFormulationID, PollutantID);



drop TABLE if exists S6;
CREATE TABLE S6
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.Oxygen ,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.Oxygen       -
PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as
Oxygendiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 7
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S6 (FuelFormulationID, PollutantID);



drop TABLE if exists S7;
CREATE TABLE S7
```

```
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.Sulfur ,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.Sulfur       -
PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as
Sulfurdiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 8
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S7 (FuelFormulationID, PollutantID);



-- Create the standardized Table.

drop TABLE if exists S;
CREATE TABLE S
SELECT      S1.FuelFormulationID, S1.PollutantID,
            S1.predCoefficientID,
            S1.RVPdiff as StandardCoeff
FROM  S1
GROUP BY FuelFormulationID, PollutantID, predCoefficientID;

UPDATE S SET      predCoefficientID       = 1;
UPDATE S SET      StandardCoeff           = 1.000;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S1.FuelFormulationID,
            S1.PollutantID,
            S1.predCoefficientID,
            S1.RVPdiff
FROM  S1
GROUP BY S1.FuelFormulationID, S1.PollutantID, S1.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S2.FuelFormulationID,
            S2.PollutantID,
            S2.predCoefficientID,
            S2.T50diff
FROM  S2
GROUP BY S2.FuelFormulationID, S2.PollutantID, S2.predCoefficientID;
```

```
INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S3.FuelFormulationID,
            S3.PollutantID,
            S3.predCoefficientID,
            S3.T90diff
FROM  S3
GROUP BY S3.FuelFormulationID, S3.PollutantID, S3.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S4.FuelFormulationID,
            S4.PollutantID,
            S4.predCoefficientID,
            S4.AROMdiff
FROM  S4
GROUP BY S4.FuelFormulationID, S4.PollutantID, S4.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S5.FuelFormulationID,
            S5.PollutantID,
            S5.predCoefficientID,
            S5.OLEFdiff
FROM  S5
GROUP BY S5.FuelFormulationID, S5.PollutantID, S5.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S6.FuelFormulationID,
            S6.PollutantID,
            S6.predCoefficientID,
            S6.Oxygendiff
FROM  S6
GROUP BY S6.FuelFormulationID, S6.PollutantID, S6.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
```

```
      predCoefficientID,
      StandardCoeff )
SELECT      S7.FuelFormulationID,
            S7.PollutantID,
            S7.predCoefficientID,
            S7.Sulfurdiff
FROM  S7
GROUP BY S7.FuelFormulationID, S7.PollutantID, S7.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            9 as predCoefficientID,
            1.00 as HiEmit
FROM  S7
GROUP BY FuelFormulationID, PollutantID, predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            10 as predCoefficientID,
            T90diff*T90diff
FROM  S3
GROUP BY FuelFormulationID, PollutantID, predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            11 as predCoefficientID,
            T50diff*T50diff
FROM  S2
GROUP BY FuelFormulationID, PollutantID, predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
```

```
SELECT      S3.FuelFormulationID,
            S3.PollutantID,
            12 as predCoefficientID,
            S3.T90diff*S6.Oxygendiff
FROM  S3 INNER JOIN S6
ON          S3.FuelFormulationID    =     S6.FuelFormulationID    AND
            S3.PollutantID                =     S6.PollutantID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S7.FuelFormulationID,
            S7.PollutantID,
            13 as predCoefficientID,
            S7.SulfurDiff*1.0
FROM  S7;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            14 as predCoefficientID,
            S6.Oxygendiff*S6.Oxygendiff
FROM  S6;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S3.FuelFormulationID,
            S3.PollutantID,
            15 as predCoefficientID,
            S3.T90diff*S4.AROMdiff
FROM  S3 INNER JOIN S4
ON          S3.FuelFormulationID    =     S4.FuelFormulationID    AND
            S3.PollutantID                =     S4.PollutantID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            16 as predCoefficientID,
            S2.T50diff*1.00
FROM  S2;
```

```
INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            17 as predCoefficientID,
            S5.OLEFdiff*S5.OLEFdiff
FROM  S5;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S3.FuelFormulationID,
            S3.PollutantID,
            18 as predCoefficientID,
            S3.T90diff*S5.OLEFdiff
FROM  S3 INNER JOIN S5
ON          S3.FuelFormulationID    =     S5.FuelFormulationID    AND
            S3.PollutantID                =     S5.PollutantID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            19 as predCoefficientID,
            S4.AROMdiff*S4.AROMdiff
FROM  S4;

CREATE UNIQUE INDEX Index1 ON S (FuelFormulationID, PollutantID,
predCoefficientID);




------------------------------------------------------------------------
--------------------
-- develop the three individual HC predictive Models.

drop TABLE if exists Model7;
CREATE TABLE Model7
SELECT      S.FuelFormulationID,
            S.PollutantID,
            S.predCoefficientID,
            S.StandardCoeff,
            PredictiveModelCoeff1.predCoefficient,
            S.StandardCoeff*PredictiveModelCoeff1.predCoefficient as M1
```

```
FROM  S LEFT JOIN PredictiveModelCoeff1
ON          S.PollutantID                       =
      PredictiveModelCoeff1.PollutantID AND
            S.predCoefficientID                 =
      PredictiveModelCoeff1.predCoefficientID
WHERE       PredictiveModelCoeff1.predModelID = 1
GROUP BY    S.FuelFormulationID, S.PollutantID, S.predCoefficientID;


CREATE UNIQUE INDEX Index1 ON Model7 (FuelFormulationID, PollutantID,
predCoefficientID);


drop TABLE if exists Model8;
CREATE TABLE Model8
SELECT      S.FuelFormulationID,
            S.PollutantID,
            S.predCoefficientID,
            S.StandardCoeff,
            PredictiveModelCoeff1.predCoefficient,
            S.StandardCoeff*PredictiveModelCoeff1.predCoefficient as M1
FROM  S LEFT JOIN PredictiveModelCoeff1
ON          S.PollutantID                       =
      PredictiveModelCoeff1.PollutantID AND
            S.predCoefficientID                 =
      PredictiveModelCoeff1.predCoefficientID
WHERE       PredictiveModelCoeff1.predModelID = 2
GROUP BY    S.FuelFormulationID, S.PollutantID, S.predCoefficientID;


CREATE UNIQUE INDEX Index1 ON Model8 (FuelFormulationID, PollutantID,
predCoefficientID);



drop TABLE if exists Model12;
CREATE TABLE Model12
SELECT      S.FuelFormulationID,
            S.PollutantID,
            S.predCoefficientID,
            S.StandardCoeff,
            PredictiveModelCoeff1.predCoefficient,
            S.StandardCoeff*PredictiveModelCoeff1.predCoefficient as M1
FROM  S LEFT JOIN PredictiveModelCoeff1
ON          S.PollutantID                       =
      PredictiveModelCoeff1.PollutantID AND
            S.predCoefficientID                 =
      PredictiveModelCoeff1.predCoefficientID
WHERE       PredictiveModelCoeff1.predModelID = 3
GROUP BY    S.FuelFormulationID, S.PollutantID, S.predCoefficientID;


CREATE UNIQUE INDEX Index1 ON Model12 (FuelFormulationID, PollutantID,
predCoefficientID);
```

```
drop TABLE if exists Model7b;
CREATE TABLE Model7b
SELECT FuelFormulationID, PollutantID, SUM(M1) as Coeff,
                        EXP( SUM(M1) ) as ECoeff
FROM  Model7
GROUP BY FuelFormulationID, PollutantID;

CREATE UNIQUE INDEX Index1 ON Model7b (FuelFormulationID, PollutantID);



drop TABLE if exists Model8b;
CREATE TABLE Model8b
SELECT FuelFormulationID, PollutantID, SUM(M1) as Coeff,
                        EXP( SUM(M1) ) as ECoeff
FROM  Model8
GROUP BY FuelFormulationID, PollutantID;

CREATE UNIQUE INDEX Index1 ON Model8b (FuelFormulationID, PollutantID);



drop TABLE if exists Model12b;
CREATE TABLE Model12b
SELECT FuelFormulationID, PollutantID, SUM(M1) as Coeff,
                        EXP( SUM(M1) ) as ECoeff
FROM  Model12
GROUP BY FuelFormulationID, PollutantID;

CREATE UNIQUE INDEX Index1 ON Model12b (FuelFormulationID,
PollutantID);


drop TABLE if exists ModelHC;
CREATE TABLE ModelHC
SELECT      Model7b.FuelFormulationID, Model7b.PollutantID,
            Model7b.ECoeff as E7,
            Model8b.ECoeff as E8,
            Model12b.ECoeff as E12,
            (SUM(Model7b.ECoeff + Model8b.ECoeff + Model12b.ECoeff)/3)
as E
FROM Model7b INNER JOIN Model8b     ON
      Model7b.FuelFormulationID=Model8b.FuelFormulationID and

      Model7b.PollutantID=Model8b.PollutantID
                  INNER JOIN Model12b    ON
      Model7b.FuelFormulationID=Model12b.FuelFormulationID and

      Model7b.PollutantID=Model12b.PollutantID
GROUP BY Model7b.FuelFormulationID, Model7b.PollutantID;


-- clean up intermediate tables

drop TABLE if exists  model12;
drop TABLE if exists  model12b;
drop TABLE if exists  model7;
```

```
drop TABLE if exists  model7b;
drop TABLE if exists  model8;
drop TABLE if exists  model8b;
drop TABLE if exists  s1;
drop TABLE if exists  s2;
drop TABLE if exists  s3;
drop TABLE if exists  s4;
drop TABLE if exists  s5;
drop TABLE if exists  s6;
drop TABLE if exists  s7;




--
**********************************************************************
******************
-- The first table join does NOX.
--
**********************************************************************
******************

UPDATE FuelFormulation SET PollutantID    =  3;



drop TABLE if exists S1;
CREATE TABLE S1
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.RVP,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.RVP    - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as RVPdiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 2
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S1 (FuelFormulationID, PollutantID);


drop TABLE if exists S2;
CREATE TABLE S2
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.T50,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.T50    - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as T50diff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
```

```
WHERE PredictiveModelCoeff2.predCoefficientID = 3
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S2 (FuelFormulationID, PollutantID);


drop TABLE if exists S3;
CREATE TABLE S3
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.T90,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.T90    - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as T90diff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 4
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S3 (FuelFormulationID, PollutantID);



drop TABLE if exists S4;
CREATE TABLE S4
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.AROM ,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.AROM   - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as AROMdiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 5
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S4 (FuelFormulationID, PollutantID);



drop TABLE if exists S5;
CREATE TABLE S5
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.OLEF ,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.OLEF   - PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as OLEFdiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 6
```

```
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S5 (FuelFormulationID, PollutantID);



drop TABLE if exists S6;
CREATE TABLE S6
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.Oxygen ,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.Oxygen       -
PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as
Oxygendiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 7
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S6 (FuelFormulationID, PollutantID);



drop TABLE if exists S7;
CREATE TABLE S7
SELECT      FuelFormulation.FuelFormulationID,
            FuelFormulation.PollutantID,
            FuelFormulation.Sulfur ,
            PredictiveModelCoeff2.predCoefficientID,
            (FuelFormulation.Sulfur       -
PredictiveModelCoeff2.predMean)/
                  PredictiveModelCoeff2.predStdDev          as
Sulfurdiff
FROM  FuelFormulation LEFT JOIN PredictiveModelCoeff2
ON          FuelFormulation.PollutantID =
PredictiveModelCoeff2.PollutantID
WHERE PredictiveModelCoeff2.predCoefficientID = 8
GROUP BY    FuelFormulation.FuelFormulationID,
FuelFormulation.PollutantID;

CREATE UNIQUE INDEX Index1 ON S7 (FuelFormulationID, PollutantID);



-- Create the standardized Table.

drop TABLE if exists S;
CREATE TABLE S
SELECT      S1.FuelFormulationID, S1.PollutantID,
            S1.predCoefficientID,
            S1.RVPdiff as StandardCoeff
FROM  S1
```

```
GROUP BY FuelFormulationID, PollutantID, predCoefficientID;

UPDATE S SET      predCoefficientID       = 1;
UPDATE S SET      StandardCoeff           = 1.000;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S1.FuelFormulationID,
            S1.PollutantID,
            S1.predCoefficientID,
            S1.RVPdiff
FROM  S1
GROUP BY S1.FuelFormulationID, S1.PollutantID, S1.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S2.FuelFormulationID,
            S2.PollutantID,
            S2.predCoefficientID,
            S2.T50diff
FROM  S2
GROUP BY S2.FuelFormulationID, S2.PollutantID, S2.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S3.FuelFormulationID,
            S3.PollutantID,
            S3.predCoefficientID,
            S3.T90diff
FROM  S3
GROUP BY S3.FuelFormulationID, S3.PollutantID, S3.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S4.FuelFormulationID,
            S4.PollutantID,
            S4.predCoefficientID,
            S4.AROMdiff
FROM  S4
GROUP BY S4.FuelFormulationID, S4.PollutantID, S4.predCoefficientID;
```

```
INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S5.FuelFormulationID,
            S5.PollutantID,
            S5.predCoefficientID,
            S5.OLEFdiff
FROM  S5
GROUP BY S5.FuelFormulationID, S5.PollutantID, S5.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S6.FuelFormulationID,
            S6.PollutantID,
            S6.predCoefficientID,
            S6.Oxygendiff
FROM  S6
GROUP BY S6.FuelFormulationID, S6.PollutantID, S6.predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S7.FuelFormulationID,
            S7.PollutantID,
            S7.predCoefficientID,
            S7.Sulfurdiff
FROM  S7
GROUP BY S7.FuelFormulationID, S7.PollutantID, S7.predCoefficientID;



INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            9 as predCoefficientID,
            1.00 as HiEmit
FROM  S7
GROUP BY FuelFormulationID, PollutantID, predCoefficientID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
```

```
      predCoefficientID,
      StandardCoeff )
SELECT      S6.FuelFormulationID,
            S6.PollutantID,
            10 as predCoefficientID,
            S6.Oxygendiff*S7.Sulfurdiff
FROM  S6 INNER JOIN S7
ON          S6.FuelFormulationID    =     S7.FuelFormulationID    AND
            S6.PollutantID                =     S7.PollutantID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S6.FuelFormulationID,
            S6.PollutantID,
            11 as predCoefficientID,
            S6.Oxygendiff*S2.T50diff
FROM  S6 INNER JOIN S2
ON          S6.FuelFormulationID    =     S2.FuelFormulationID    AND
            S6.PollutantID                =     S2.PollutantID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S6.FuelFormulationID,
            S6.PollutantID,
            12 as predCoefficientID,
            S6.Oxygendiff*S3.T90diff
FROM  S6 INNER JOIN S3
ON          S6.FuelFormulationID    =     S3.FuelFormulationID    AND
            S6.PollutantID                =     S3.PollutantID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S6.FuelFormulationID,
            S6.PollutantID,
            13 as predCoefficientID,
            S6.Oxygendiff*S4.AROMdiff
FROM  S6 INNER JOIN S4
ON          S6.FuelFormulationID    =     S4.FuelFormulationID    AND
            S6.PollutantID                =     S4.PollutantID;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
```

```
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            14 as predCoefficientID,
            S6.Oxygendiff*S6.Oxygendiff
FROM  S6;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            15 as predCoefficientID,
            S2.T50diff*S2.T50diff
FROM  S2;


INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      FuelFormulationID,
            PollutantID,
            16 as predCoefficientID,
            S7.Sulfurdiff*S7.Sulfurdiff
FROM  S7;



INSERT INTO S
(     FuelFormulationID,
      PollutantID,
      predCoefficientID,
      StandardCoeff )
SELECT      S7.FuelFormulationID,
            S7.PollutantID,
            17 as predCoefficientID,
            S7.Sulfurdiff*S3.T90diff
FROM  S7 INNER JOIN S3
ON          S7.FuelFormulationID    =     S3.FuelFormulationID    AND
            S7.PollutantID                =     S3.PollutantID;



CREATE UNIQUE INDEX Index1 ON S (FuelFormulationID, PollutantID,
predCoefficientID);



------------------------------------------------------------------------
---------------------
-- develop the six individual NOx predictive Models.
```

```
drop TABLE if exists Model1;
CREATE TABLE Model1
SELECT      S.FuelFormulationID,
            S.PollutantID,
            S.predCoefficientID,
            S.StandardCoeff,
            PredictiveModelCoeff1.predCoefficient,
            S.StandardCoeff*PredictiveModelCoeff1.predCoefficient as M1
FROM  S LEFT JOIN PredictiveModelCoeff1
ON          S.PollutantID                       =
      PredictiveModelCoeff1.PollutantID AND
            S.predCoefficientID                 =
      PredictiveModelCoeff1.predCoefficientID
WHERE       PredictiveModelCoeff1.predModelID = 1
GROUP BY    S.FuelFormulationID, S.PollutantID, S.predCoefficientID;


CREATE UNIQUE INDEX Index1 ON Model1 (FuelFormulationID, PollutantID,
predCoefficientID);


drop TABLE if exists Model2;
CREATE TABLE Model2
SELECT      S.FuelFormulationID,
            S.PollutantID,
            S.predCoefficientID,
            S.StandardCoeff,
            PredictiveModelCoeff1.predCoefficient,
            S.StandardCoeff*PredictiveModelCoeff1.predCoefficient as M1
FROM  S LEFT JOIN PredictiveModelCoeff1
ON          S.PollutantID                       =
      PredictiveModelCoeff1.PollutantID AND
            S.predCoefficientID                 =
      PredictiveModelCoeff1.predCoefficientID
WHERE       PredictiveModelCoeff1.predModelID = 2
GROUP BY    S.FuelFormulationID, S.PollutantID, S.predCoefficientID;


CREATE UNIQUE INDEX Index1 ON Model2 (FuelFormulationID, PollutantID,
predCoefficientID);



drop TABLE if exists Model3;
CREATE TABLE Model3
SELECT      S.FuelFormulationID,
            S.PollutantID,
            S.predCoefficientID,
            S.StandardCoeff,
            PredictiveModelCoeff1.predCoefficient,
            S.StandardCoeff*PredictiveModelCoeff1.predCoefficient as M1
FROM  S LEFT JOIN PredictiveModelCoeff1
ON          S.PollutantID                       =
      PredictiveModelCoeff1.PollutantID AND
            S.predCoefficientID                 =
      PredictiveModelCoeff1.predCoefficientID
```

```
WHERE       PredictiveModelCoeff1.predModelID = 3
GROUP BY    S.FuelFormulationID, S.PollutantID, S.predCoefficientID;


CREATE UNIQUE INDEX Index1 ON Model3 (FuelFormulationID, PollutantID,
predCoefficientID);



drop TABLE if exists Model4;
CREATE TABLE Model4
SELECT      S.FuelFormulationID,
            S.PollutantID,
            S.predCoefficientID,
            S.StandardCoeff,
            PredictiveModelCoeff1.predCoefficient,
            S.StandardCoeff*PredictiveModelCoeff1.predCoefficient as M1
FROM  S LEFT JOIN PredictiveModelCoeff1
ON          S.PollutantID                       =
      PredictiveModelCoeff1.PollutantID AND
            S.predCoefficientID                 =
      PredictiveModelCoeff1.predCoefficientID
WHERE       PredictiveModelCoeff1.predModelID = 4
GROUP BY    S.FuelFormulationID, S.PollutantID, S.predCoefficientID;


CREATE UNIQUE INDEX Index1 ON Model4 (FuelFormulationID, PollutantID,
predCoefficientID);


drop TABLE if exists Model5;
CREATE TABLE Model5
SELECT      S.FuelFormulationID,
            S.PollutantID,
            S.predCoefficientID,
            S.StandardCoeff,
            PredictiveModelCoeff1.predCoefficient,
            S.StandardCoeff*PredictiveModelCoeff1.predCoefficient as M1
FROM  S LEFT JOIN PredictiveModelCoeff1
ON          S.PollutantID                       =
      PredictiveModelCoeff1.PollutantID AND
            S.predCoefficientID                 =
      PredictiveModelCoeff1.predCoefficientID
WHERE       PredictiveModelCoeff1.predModelID = 5
GROUP BY    S.FuelFormulationID, S.PollutantID, S.predCoefficientID;


CREATE UNIQUE INDEX Index1 ON Model5 (FuelFormulationID, PollutantID,
predCoefficientID);



drop TABLE if exists Model6;
CREATE TABLE Model6
SELECT      S.FuelFormulationID,
            S.PollutantID,
```

```
            S.predCoefficientID,
            S.StandardCoeff,
            PredictiveModelCoeff1.predCoefficient,
            S.StandardCoeff*PredictiveModelCoeff1.predCoefficient as M1
FROM  S LEFT JOIN PredictiveModelCoeff1
ON          S.PollutantID                       =
      PredictiveModelCoeff1.PollutantID AND
            S.predCoefficientID                 =
      PredictiveModelCoeff1.predCoefficientID
WHERE       PredictiveModelCoeff1.predModelID = 6
GROUP BY    S.FuelFormulationID, S.PollutantID, S.predCoefficientID;


CREATE UNIQUE INDEX Index1 ON Model6 (FuelFormulationID, PollutantID,
predCoefficientID);







drop TABLE if exists Model1b;
CREATE TABLE Model1b
SELECT FuelFormulationID, PollutantID, SUM(M1) as Coeff,
                        EXP( SUM(M1) ) as ECoeff
FROM  Model1
GROUP BY FuelFormulationID, PollutantID;

CREATE UNIQUE INDEX Index1 ON Model1b (FuelFormulationID, PollutantID);



drop TABLE if exists Model2b;
CREATE TABLE Model2b
SELECT FuelFormulationID, PollutantID, SUM(M1) as Coeff,
                        EXP( SUM(M1) ) as ECoeff
FROM  Model2
GROUP BY FuelFormulationID, PollutantID;

CREATE UNIQUE INDEX Index1 ON Model2b (FuelFormulationID, PollutantID);



drop TABLE if exists Model3b;
CREATE TABLE Model3b
SELECT FuelFormulationID, PollutantID, SUM(M1) as Coeff,
                        EXP( SUM(M1) ) as ECoeff
FROM  Model3
GROUP BY FuelFormulationID, PollutantID;

CREATE UNIQUE INDEX Index1 ON Model3b (FuelFormulationID, PollutantID);



drop TABLE if exists Model4b;
```

```
CREATE TABLE Model4b
SELECT FuelFormulationID, PollutantID, SUM(M1) as Coeff,
                        EXP( SUM(M1) ) as ECoeff
FROM  Model4
GROUP BY FuelFormulationID, PollutantID;

CREATE UNIQUE INDEX Index1 ON Model4b (FuelFormulationID, PollutantID);



drop TABLE if exists Model5b;
CREATE TABLE Model5b
SELECT FuelFormulationID, PollutantID, SUM(M1) as Coeff,
                        EXP( SUM(M1) ) as ECoeff
FROM  Model5
GROUP BY FuelFormulationID, PollutantID;

CREATE UNIQUE INDEX Index1 ON Model5b (FuelFormulationID, PollutantID);



drop TABLE if exists Model6b;
CREATE TABLE Model6b
SELECT FuelFormulationID, PollutantID, SUM(M1) as Coeff,
                        EXP( SUM(M1) ) as ECoeff
FROM  Model6
GROUP BY FuelFormulationID, PollutantID;

CREATE UNIQUE INDEX Index1 ON Model6b (FuelFormulationID, PollutantID);







drop TABLE if exists ModelNOx;
CREATE TABLE ModelNOx
SELECT      Model1b.FuelFormulationID, Model1b.PollutantID,
            Model1b.ECoeff as E1,
            Model2b.ECoeff as E2,
            Model3b.ECoeff as E3,
            Model4b.ECoeff as E4,
            Model5b.ECoeff as E5,
            Model6b.ECoeff as E6,
            (SUM(Model1b.ECoeff + Model2b.ECoeff + Model3b.ECoeff +
                  Model4b.ECoeff + Model5b.ECoeff + Model6b.ECoeff)/6)
as E
FROM Model1b INNER JOIN Model2b     ON
      Model1b.FuelFormulationID=Model2b.FuelFormulationID and

      Model1b.PollutantID=Model2b.PollutantID
                  INNER JOIN Model3b           ON
      Model1b.FuelFormulationID=Model3b.FuelFormulationID and

      Model1b.PollutantID=Model3b.PollutantID
                  INNER JOIN Model4b           ON
      Model1b.FuelFormulationID=Model4b.FuelFormulationID and
```

```
      Model1b.PollutantID=Model4b.PollutantID
                  INNER JOIN Model5b           ON
      Model1b.FuelFormulationID=Model5b.FuelFormulationID and

      Model1b.PollutantID=Model5b.PollutantID
                  INNER JOIN Model6b           ON
      Model1b.FuelFormulationID=Model6b.FuelFormulationID and

      Model1b.PollutantID=Model6b.PollutantID
GROUP BY Model1b.FuelFormulationID, Model1b.PollutantID;


-- clean up intermediate tables

drop TABLE if exists  model1;
drop TABLE if exists  model2;
drop TABLE if exists  model3;
drop TABLE if exists  model4;
drop TABLE if exists  model5;
drop TABLE if exists  model6;

drop TABLE if exists  model1b;
drop TABLE if exists  model2b;
drop TABLE if exists  model3b;
drop TABLE if exists  model4b;
drop TABLE if exists  model5b;
drop TABLE if exists  model6b;



drop TABLE if exists  model12;
drop TABLE if exists  model12b;
drop TABLE if exists  model7;
drop TABLE if exists  model7b;
drop TABLE if exists  model8;
drop TABLE if exists  model8b;
drop TABLE if exists  s1;
drop TABLE if exists  s2;
drop TABLE if exists  s3;
drop TABLE if exists  s4;
drop TABLE if exists  s5;
drop TABLE if exists  s6;
drop TABLE if exists  s7;


-------------------------------------------------------------------------
----------------------
--                            End Predictive Model for HC and NOx.  No
Sulfur Effects
-------------------------------------------------------------------------
----------------------
-------------------------------------------------------------------------
----------------------
-------------------------------------------------------------------------
----------------------
```

```
-------------------------------------------------------------------------
---------------------



drop TABLE if exists P1HCNOx;
CREATE TABLE P1HCNOx
SELECT      HC.PollutantID,
            HC.FuelFormulationID,
            HC.E
FROM  ModelHC as HC
GROUP BY PollutantID, FuelFormulationID;



INSERT INTO P1HCNOx
(           PollutantID,
            FuelFormulationID,
            E     )
SELECT      NOx.PollutantID,
            NOx.FuelFormulationID,
            NOx.E
FROM  ModelNOx as NOx
GROUP BY PollutantID, FuelFormulationID;


ALTER TABLE P1HCNOx ADD PRIMARY KEY (PollutantID, FuelFormulationID);
CREATE UNIQUE INDEX Index1 ON P1HCNOx (PollutantID, FuelFormulationID);



-------------------------------------------------------------------------
--------------------
--                      Bring in the BASE or Reference
FuelFormulationID = 98
--                         This fuel is based on Maricopa County,
Arizona.

--FuelSubType                         10
      Conventional Gasoline
--OXYGEN (wt%)                        3.5
--SULFUR (ppm)                       30.0
--RVP  (psi)                          8.7
--E200 (%)                           41.0
--E300 (%)                           83.0
--AROMATICS (vol%)                   32.1
--OLEFINS  (vol%)             7.7
--FUEL BENZENE (vol%)         0.8
--T50                            218
--T90                            329
--
-------------------------------------------------------------------------
--------------------

drop TABLE if exists P1HCNOxref;
CREATE TABLE P1HCNOxref
```

```
SELECT      HC.PollutantID,
            HC.FuelFormulationID,
            HC.E
FROM  ModelHC as HC
WHERE       FuelFormulationID = 98
GROUP BY PollutantID, FuelFormulationID;



INSERT INTO P1HCNOxref
(           PollutantID,
            FuelFormulationID,
            E     )
SELECT      NOx.PollutantID,
            NOx.FuelFormulationID,
            NOx.E
FROM  ModelNOx as NOx
WHERE       FuelFormulationID = 98
GROUP BY PollutantID, FuelFormulationID;


ALTER TABLE P1HCNOxref ADD PRIMARY KEY (PollutantID,
FuelFormulationID);
CREATE UNIQUE INDEX Index1 ON P1HCNOxref (PollutantID,
FuelFormulationID);




--    Predictive Model creates the Fuel AdjustmentFactor for each
fuelformulationID and
--          pollutantID.  The adjustment is the  ratio of the
Predictive Model Output for
--          a particular Target Fuel to the Predictive Model Output of
the Reference fuel.
--    The Predictive Model is not a function of vehicle model year
(fuel model year group)
--          or SourceTypeID.


drop TABLE if exists P1;
CREATE TABLE P1
SELECT      P1HCNOx.PollutantID,
            P1HCNOx.FuelFormulationID,
            P1HCNOx.E  / P1HCNOxref.E as AdjustmentFactor
FROM P1HCNOx LEFT JOIN P1HCNOxref
ON          P1HCNOx.PollutantID                 =
      P1HCNOxref.PollutantID
GROUP BY PollutantID, FuelFormulationID;

ALTER TABLE P1 ADD PRIMARY KEY (PollutantID, FuelFormulationID);
CREATE UNIQUE INDEX Index1 ON P1 (PollutantID, FuelFormulationID);

alter TABLE P1  ADD (
      FuelMYGroupID           INT(8)
);
```

```
UPDATE P1   SET   FuelMYGroupID = 1974;


drop TABLE if exists PredictiveModelHCNOx;
CREATE TABLE PredictiveModelHCNOx
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            P1.FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;

INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            19751986 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            19871989 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            19901993 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
```

```
            P1.FuelFormulationID,
            1994 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            1995 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            1996 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            19972000 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            20012003 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;
```

```
INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            2004 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            2005 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            2006 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            2007 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
```

```
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            20082009 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      FuelFormulationID,
      FuelMYGroupID,
      AdjustmentFactor  )
SELECT      P1.PollutantID,
            P1.FuelFormulationID,
            20102050 as FuelMYGroupID,
            P1.AdjustmentFactor
FROM  P1
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


ALTER TABLE PredictiveModelHCNOx
            ADD PRIMARY KEY (PollutantID, FuelMYGroupID,
FuelFormulationID);
CREATE UNIQUE INDEX Index1 ON
            PredictiveModelHCNOx  (PollutantID, FuelMYGroupID,
FuelFormulationID);




-- Add in the other sourcetypeID categories.  These are currently
duplicate adjustment factors.

drop TABLE if exists dummy;
CREATE TABLE dummy
SELECT      PM.PollutantID,
            PM.FuelFormulationID,
            PM.FuelMYGroupID,
            PM.AdjustmentFactor
FROM PredictiveModelHCNOx as PM
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;



ALTER TABLE dummy
     ADD (
            SourceTypeID            smallint(6)
);
UPDATE dummy      SET   SourceTypeID = 11;


drop TABLE if exists PredictiveModelHCNOx;
CREATE TABLE PredictiveModelHCNOx
```

```
SELECT      dummy.PollutantID,
            dummy.SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            21 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            31 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            32 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
```

```
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            41 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            42 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            43 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            51 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;
```

```
INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            52 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            53 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            54 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            61 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
```

```
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO PredictiveModelHCNOx
(     PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            62 as SourceTypeID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;



ALTER TABLE PredictiveModelHCNOx
            ADD PRIMARY KEY (PollutantID, SourceTypeID, FuelMYGroupID,
FuelFormulationID);
CREATE UNIQUE INDEX Index1 ON
            PredictiveModelHCNOx  (PollutantID, SourceTypeID,
FuelMYGroupID, FuelFormulationID);



--    do final Table clean-up.

drop TABLE if exists s;
drop TABLE if exists dummy;
drop TABLE if exists p1;
drop TABLE if exists p1hcnox;
drop TABLE if exists p1hcnoxref;




--
**************************************************************************
******************
--
**************************************************************************
******************
--
**************************************************************************
******************
--
**************************************************************************
******************
--
**************************************************************************
******************
--                Add MOBILE6.2 Sulfur Effects into the Fuel Model for
HC and NOx
```

```
--                The current PredictiveModelHCNOx table does not
contain sulfur effects.
--
*************************************************************************
******************
--
*************************************************************************
******************
--
*************************************************************************
******************
--
*************************************************************************
******************
--
*************************************************************************
******************


-- Obtain the Sulfur levels from the MOVESFuelRFS_Base.FuelFormulation
table.  It is the original
--                data source.  The RFSPredictive.FuelFormulation table
has all of the Sulfur
--                values set to 30 ppm.

--    This first section adds the SHORT TERM Sulfur effects to the HC
and NOx pollutants.

drop TABLE if exists PredictiveSulfurHCNOx1;
CREATE TABLE PredictiveSulfurHCNOx1
SELECT
      P.PollutantID,
      P.SourceTypeID,
      P.FuelMYGroupID,
      FF.FuelFormulationID,
      P.AdjustmentFactor as OtherFactor,
      FF.SulfurLevel as Sulfur,
      FF.fuelSubtypeID
FROM  PredictiveModelHCNOx P LEFT JOIN
MOVESFuelRFS_Base.FuelFormulation FF
ON    P.FuelFormulationID     =     FF.FuelFormulationID
WHERE       FF.fuelSubtypeID in (10, 11, 12, 13, 14)
GROUP BY    PollutantID, SourceTypeID, FuelMYGroupID,
FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
            PredictiveSulfurHCNOx1  (PollutantID, SourceTypeID,
FuelMYGroupID, FuelFormulationID);



--
*************************************************************************
*******
--    These sulfur coefficients were taken from the MOBILE6 report by
Tesh Rao.  *
```

```
--    emitterType is "H" for High emitter and "N" for Normal emitter.
*
--    Some equations are log-log and others are log-linear.
*
--
**************************************************************************
*******

drop TABLE if exists SulfurCoeff;
CREATE TABLE SulfurCoeff  (
      processID                     smallint(6),
      pollutantID                   smallint(6),
      emitterType                   CHAR(1),
      sourceTypeID                  smallint(6),
      fuelMYGroupID                 INT(8),
      functionType                  CHAR(10),
      sulfurCoeff                   float,
      PRIMARY KEY                   (processID, pollutantID,
emitterType, sourceTypeID, fuelMYGroupID)

);

LOAD DATA INFILE 'C:\\Eds C Files\\MOVES FUELBinner\\SQL Fuel Binner
Code\\SulfurCoefficients.csv' REPLACE
        INTO TABLE SulfurCoeff
        FIELDS TERMINATED BY ','
        OPTIONALLY ENCLOSED BY '"'
        ESCAPED BY '\\'
        LINES TERMINATED BY '\r\n'
        IGNORE 1 LINES;

CREATE UNIQUE INDEX Index1 ON
            SulfurCoeff  (processID, PollutantID, emitterType,
SourceTypeID, FuelMYGroupID);




--    This section is only the  Normal emitters.  This process was done
piecemeal
--          rather than normals and high together that is why they are
in separate
--          sections.  Rational coding would have done this together.

drop TABLE if exists PredictiveSulfurHCNOx2;
CREATE TABLE PredictiveSulfurHCNOx2
SELECT
      C.PollutantID,
      C.FuelMYGroupID,
      C.FuelFormulationID,
      C.SourceTypeID,
      C.FuelSubtypeID,
      C.Sulfur,
      C.OtherFactor,
      SC.processID,
      SC.functionType,
      SC.emitterType,
```

```
      SC.sulfurCoeff
FROM  PredictiveSulfurHCNOx1 C LEFT JOIN SulfurCoeff SC
ON    C.PollutantID     =     SC.PollutantID                and
      C.SourceTypeID    =     SC.SourceTypeID               and
      C.FuelMYGroupID   =     SC.FuelMYGroupID
WHERE SC.emitterType = 'N' and SC.processID = 1
GROUP BY    SC.ProcessID, C.PollutantID, C.SourceTypeID,
                  C.FuelMYGroupID, C.FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
            PredictiveSulfurHCNOx2  (ProcessID, PollutantID,
SourceTypeID,
                                     FuelMYGroupID, FuelFormulationID);


drop TABLE if exists Junk;
CREATE TABLE Junk
SELECT
      C.PollutantID,
      C.FuelMYGroupID,
      C.FuelFormulationID,
      C.SourceTypeID,
      C.FuelSubtypeID,
      C.Sulfur,
      C.OtherFactor,
      SC.processID,
      SC.functionType,
      SC.emitterType,
      SC.sulfurCoeff
FROM  PredictiveSulfurHCNOx1 C LEFT JOIN SulfurCoeff SC
ON    C.PollutantID     =     SC.PollutantID                and
      C.SourceTypeID    =     SC.SourceTypeID               and
      C.FuelMYGroupID   =     SC.FuelMYGroupID
WHERE SC.emitterType = 'N' and SC.processID = 2
GROUP BY    SC.processID, C.PollutantID, C.SourceTypeID,
                  C.FuelMYGroupID, C.FuelFormulationID;


INSERT INTO PredictiveSulfurHCNOx2
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      Sulfur,
      OtherFactor,
      processID,
      functionType,
      emitterType,
      sulfurCoeff )
SELECT
      PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      Sulfur,
```

```
      OtherFactor,
      processID,
      functionType,
      emitterType,
      sulfurCoeff
FROM  Junk
GROUP BY    ProcessID, PollutantID, SourceTypeID,
                        FuelMYGroupID, FuelFormulationID;

drop TABLE if exists Junk;



Alter TABLE PredictiveSulfurHCNOx2  ADD (
      SulfAdj                       float,
      Sulf1                   float,
      Sulf30                        float,
      longTermSulfur          float,
      SulfIRR                       float,
      SulfMax                       float,
      FinalSulfAdj            float,
      IRFactor                float,
      SulfAdj2                float,
      SulfGPA                       float,
      Sulf1000                float,
      GPASulfAdj              float,
      AdjustmentFactorAll     float
);



--*************************************************************
--    Calculate Short Term Sulfur Effects for Normal Emitters   *
--*************************************************************


-- Tier0 and LEV + vehicles use LOG - LOG algorithm.

UPDATE      PredictiveSulfurHCNOx2  SET   Sulf1       =
      EXP(sulfurCoeff* LN(Sulfur));
UPDATE      PredictiveSulfurHCNOx2  SET   Sulf30            =
      EXP(sulfurCoeff* LN(30));



-- Tier1 vehicles use a LOG - Linear algorithm.

UPDATE      PredictiveSulfurHCNOx2  SET   Sulf1       =
      EXP(sulfurCoeff*Sulfur)
                        WHERE       FuelMYGroupID IN (1994, 1995, 1996,
19972000);
UPDATE      PredictiveSulfurHCNOx2  SET   Sulf30            =
      EXP(sulfurCoeff*30)
                        WHERE       FuelMYGroupID IN (1994, 1995, 1996,
19972000);
```

```
-- Used for both Tier0 and Tier1.

UPDATE      PredictiveSulfurHCNOx2  SET   SulfAdj           =     0.0
      WHERE Sulfur = 30;

UPDATE      PredictiveSulfurHCNOx2  SET   SulfAdj           =
      (Sulf1 - Sulf30) / Sulf30
                              WHERE Sulfur <> 30;



--**************************************
--    Calculate Long Term Sulfur Effects    *
--**************************************

--  Add the long term sulfur effects to the LEV and later vehicles for
sulfur levels greater
--          than 30 ppm.  Sulfur levels below 30 ppm are assumed to
have no long term effects.
--    There is no way to logically extrapolate these effects below 30
ppm.  If 30 ppm has
--          a zero long term sulfur effect then Sulfur < 30 ppm should
not have any effect
--          either.

UPDATE      PredictiveSulfurHCNOx2  SET   longTermSulfur    =     1.0;

UPDATE      PredictiveSulfurHCNOx2  SET   longTermSulfur    =     2.50
      WHERE PollutantID = 1 and
            FuelMYGroupID IN (20012003, 2004, 2005, 2006, 2007,
20082009, 20102050) and
            Sulfur > 30.0;

UPDATE      PredictiveSulfurHCNOx2  SET   longTermSulfur    =     1.47
      WHERE PollutantID = 3 and
            FuelMYGroupID IN (20012003, 2004, 2005, 2006, 2007,
20082009, 20102050) and
            Sulfur > 30.0;


UPDATE      PredictiveSulfurHCNOx2  SET   SulfAdj2          =
      SulfAdj*LongTermSulfur;



--
**********************************************************************
**********
-- Sulfur irreversibility effects for 2004+ model years and Sulfur > 30
ppm only.  *
--
**********************************************************************
**********
```

```
-- compute the normal sulfur irreversibility effects

UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(303))
                  WHERE FuelMYGroupID = 2004;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(303))
                  WHERE FuelMYGroupID = 2005;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(87))
                  WHERE FuelMYGroupID = 2006;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(87))
                  WHERE FuelMYGroupID = 2007;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(80))
                  WHERE FuelMYGroupID IN (20082009, 20102050);


-- compute the sulfur irreversibility effects if the selected sulfur
level is
--          greater than the maximum sulfur level.  Note these may not
exist in the
--          fuel supply table (real world) but the model needs
fueladjustment effects
--          anyway as placeholders.  Making the maximum sulfur level
equal to the
--          actual one assures that the sulfur irreversibility effects
are not
--          smaller than the actual effects.

UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID = 2004 and Sulfur > 303;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID = 2005 and Sulfur > 303;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID = 2006 and Sulfur > 87;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID = 2007 and Sulfur > 87;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID IN (20082009, 20102050) and
Sulfur > 80;



--
*************************************************************************
***********
-- Sulfur irreversibility effects for 2004+ model years and Sulfur > 30
ppm only.  *
```

```
--
**********************************************************************
***********

UPDATE      PredictiveSulfurHCNOx2  SET   SulfMax           =     0.0;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfMax           =
      (SulfIRR - Sulf30) / Sulf30         WHERE
                  FuelMYGroupID IN (2004, 2005, 2006, 2007, 20082009,
20102050) and Sulfur > 30.0;

UPDATE      PredictiveSulfurHCNOx2  SET   IRFactor    =     0.000;
UPDATE      PredictiveSulfurHCNOx2  SET   IRFactor    =     0.425
            WHERE FuelMYGroupID IN (2004, 2005, 2006, 2007, 20082009,
20102050) and
                    Sulfur > 30.0;


-- SulfAdj2 is the fractional increase in emissions of the sulfur
compared with the base
--          sulfur level 30 ppm.  SulfMax is the increase due to the
sulfur irreversibility
--          effects.

UPDATE      PredictiveSulfurHCNOx2
      SET   FinalSulfAdj      =     (IRFactor * SulfMax) +  (1.0 -
IRFactor) * SulfAdj2;

UPDATE      PredictiveSulfurHCNOx2  SET   FinalSulfAdj      =     1 +
FinalSulfAdj;



-- This bottoms the sulfur effects at levels around 4 ppm Sulfur.
Lower gasoline sulfur
--          levels will have no effect on the emissions.  This cap is
needed because the
--          EXTRAPOLATED relationship is log-log in nature.

-- Currently the lowest default fuel sulfur level is 10 ppm S.


UPDATE      PredictiveSulfurHCNOx2  SET   FinalSulfAdj            =
      0.50  WHERE FinalSulfAdj < 0.50;




--************************
--    Sulfur GPA Effects   *
--************************

-- Sulfur GPA is only applied to FuelMYGroupIDs of 2004, 2005 and 2006.
It is a phase-out
--          strategy for sulfur in the Rocky Mountain states.

UPDATE      PredictiveSulfurHCNOx2  SET   Sulf1000    =
      EXP(sulfurCoeff* LN(1000.0))
```

```
                  WHERE FuelMYGroupID IN (2004, 2005, 2006);

UPDATE      PredictiveSulfurHCNOx2  SET   SulfGPA           =     1.00;
UPDATE      PredictiveSulfurHCNOx2  SET   SulfGPA           =
      (Sulf1000 - Sulf30) / Sulf30  WHERE
                  FuelMYGroupID IN (2004, 2005, 2006) and Sulfur >
30.0;


UPDATE      PredictiveSulfurHCNOx2  SET   GPASulfAdj  =
      FinalSulfAdj;

UPDATE      PredictiveSulfurHCNOx2  SET   GPASulfAdj  =
                           (IRFactor * SulfGPA) +   (1.0 - IRFactor)
* SulfAdj2
                  WHERE FuelMYGroupID IN (2004, 2005, 2006) and Sulfur
> 30.0;

UPDATE      PredictiveSulfurHCNOx2  SET   GPASulfAdj  =     1.0 +
GPASulfAdj  WHERE
                  FuelMYGroupID IN (2004, 2005, 2006) and Sulfur >
30.0;









--*********************************************************************
--*********************************************************************
--*********************************************************************
--*********************************************************************
--                       Sulfur effects for High emitters only
*
--*********************************************************************
--*********************************************************************


--    Add the sulfur coefficients to the existing Predictive model
results.
--    This section is only the HIGH emitters.  This process was done
piecemeal
--          rather than normals and high together that is why they are
in separate
--          sections.  Rational coding would have done this together.

drop TABLE if exists PredictiveSulfurHCNOxHigh;
CREATE TABLE PredictiveSulfurHCNOxHigh
SELECT
      C.PollutantID,
```

```
      C.FuelMYGroupID,
      C.FuelFormulationID,
      C.SourceTypeID,
      C.FuelSubtypeID,
      C.Sulfur,
      C.OtherFactor,
      SC.processID,
      SC.functionType,
      SC.emitterType,
      SC.sulfurCoeff
FROM  PredictiveSulfurHCNOx1 C LEFT JOIN SulfurCoeff SC
ON    C.PollutantID     =     SC.PollutantID                and
      C.SourceTypeID    =     SC.SourceTypeID               and
      C.FuelMYGroupID   =     SC.FuelMYGroupID
WHERE SC.emitterType = 'H' and SC.processID = 1
GROUP BY    SC.ProcessID, C.PollutantID, C.SourceTypeID,
                  C.FuelMYGroupID, C.FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
            PredictiveSulfurHCNOxHigh  (ProcessID, PollutantID,
SourceTypeID,
                                    FuelMYGroupID, FuelFormulationID);


drop TABLE if exists Junk;
CREATE TABLE Junk
SELECT
      C.PollutantID,
      C.FuelMYGroupID,
      C.FuelFormulationID,
      C.SourceTypeID,
      C.FuelSubtypeID,
      C.Sulfur,
      C.OtherFactor,
      SC.processID,
      SC.functionType,
      SC.emitterType,
      SC.sulfurCoeff
FROM  PredictiveSulfurHCNOx1 C LEFT JOIN SulfurCoeff SC
ON    C.PollutantID     =     SC.PollutantID                and
      C.SourceTypeID    =     SC.SourceTypeID               and
      C.FuelMYGroupID   =     SC.FuelMYGroupID
WHERE SC.emitterType = 'H' and SC.processID = 2
GROUP BY    SC.processID, C.PollutantID, C.SourceTypeID,
                  C.FuelMYGroupID, C.FuelFormulationID;


INSERT INTO PredictiveSulfurHCNOxHigh
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      Sulfur,
      OtherFactor,
      processID,
      functionType,
```

```
	emitterType,
	sulfurCoeff )
SELECT
	PollutantID,
	FuelMYGroupID,
	FuelFormulationID,
	SourceTypeID,
	FuelSubtypeID,
	Sulfur,
	OtherFactor,
	processID,
	functionType,
	emitterType,
	sulfurCoeff
FROM  Junk
GROUP BY    ProcessID, PollutantID, SourceTypeID,
                        FuelMYGroupID, FuelFormulationID;

drop TABLE if exists Junk;


Alter TABLE PredictiveSulfurHCNOxHigh  ADD (
	SulfAdj                         float,
	Sulf1                   float,
	Sulf30                          float,
	longTermSulfur          float,
	SulfIRR                         float,
	SulfMax                         float,
	FinalSulfAdj            float,
	IRFactor                float,
	SulfAdj2                float,
	SulfGPA                         float,
	Sulf1000                float,
	GPASulfAdj              float,
	AdjustmentFactorAll     float
);



--*************************************************************
--    Calculate Short Term Sulfur Effects for High Emitters     *
--*************************************************************


-- Tier0 and LEV + vehicles use LOG - LOG algorithm.

UPDATE      PredictiveSulfurHCNOxHigh     SET   Sulf1       =
      EXP(sulfurCoeff* LN(Sulfur));
UPDATE      PredictiveSulfurHCNOxHigh     SET   Sulf30            =
      EXP(sulfurCoeff* LN(30));



-- Tier1 vehicles use a LOG - Linear algorithm.

UPDATE      PredictiveSulfurHCNOxHigh     SET   Sulf1       =
      EXP(sulfurCoeff*Sulfur)
```

```
                        WHERE       FuelMYGroupID IN (1994, 1995, 1996,
19972000);
UPDATE      PredictiveSulfurHCNOxHigh     SET   Sulf30            =
      EXP(sulfurCoeff*30)
                        WHERE       FuelMYGroupID IN (1994, 1995, 1996,
19972000);


-- Tier0 and Tier1 used these equations.

UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfAdj           =
      0.0         WHERE Sulfur = 30;

UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfAdj           =
      (Sulf1 - Sulf30) / Sulf30
                              WHERE Sulfur <> 30;




--****************************************
--    Calculate Long Term Sulfur Effects    *
--****************************************

--  Add the long term sulfur effects to the LEV and later vehicles for
sulfur  levels greater
--          than 30 ppm.  Sulfur levels below 30 ppm are assumed to
have no long term effects.
--    There is no way to logically extrapolate these effects below 30
ppm.  If 30 ppm has
--          a zero long term sulfur effect then Sulfur < 30 ppm should
not have any effect
--          either.

UPDATE      PredictiveSulfurHCNOxHigh     SET   longTermSulfur    =
      1.0;

UPDATE      PredictiveSulfurHCNOxHigh     SET   longTermSulfur    =
      2.50  WHERE PollutantID = 1 and
            FuelMYGroupID IN (20012003, 2004, 2005, 2006, 2007,
20082009, 20102050) and
            Sulfur > 30.0;

UPDATE      PredictiveSulfurHCNOxHigh     SET   longTermSulfur    =
      1.47  WHERE PollutantID = 3 and
            FuelMYGroupID IN (20012003, 2004, 2005, 2006, 2007,
20082009, 20102050) and
            Sulfur > 30.0;


UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfAdj2          =
      SulfAdj*LongTermSulfur;
```

```
--
**********************************************************************
***********
-- Sulfur irreversibility effects for 2004+ model years and Sulfur > 30
ppm only.  *
--
**********************************************************************
***********

-- compute the normal sulfur irreversibility effects

UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(303))
                  WHERE FuelMYGroupID = 2004;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(303))
                  WHERE FuelMYGroupID = 2005;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(87))
                  WHERE FuelMYGroupID = 2006;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(87))
                  WHERE FuelMYGroupID = 2007;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(80))
                  WHERE FuelMYGroupID IN (20082009, 20102050);


-- compute the sulfur irreversibility effects if the selected sulfur
level is
--          greater than the maximum sulfur level.  Note these may not
exist in the
--          fuel supply table (real world) but the model needs
fueladjustment effects
--          anyway as placeholders.  Making the maximum sulfur level
equal to the
--          actual one assures that the sulfur irreversibility effects
don't get
--          smaller than the actual effects.

UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID = 2004 and Sulfur > 303;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID = 2005 and Sulfur > 303;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID = 2006 and Sulfur > 87;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID = 2007 and Sulfur > 87;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfIRR           =
      EXP(sulfurCoeff* LN(sulfur))
                  WHERE FuelMYGroupID IN (20082009, 20102050) and
Sulfur > 80;
```

```
--
*****************************************************************
***********
-- Sulfur irreversibility effects for 2004+ model years and Sulfur > 30
ppm only.  *
--
*****************************************************************
***********

UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfMax           =
      0.0;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfMax           =
      (SulfIRR - Sulf30) / Sulf30         WHERE
                  FuelMYGroupID IN (2004, 2005, 2006, 2007, 20082009,
20102050) and Sulfur > 30.0;

UPDATE      PredictiveSulfurHCNOxHigh     SET   IRFactor    =
      0.000;
UPDATE      PredictiveSulfurHCNOxHigh     SET   IRFactor    =     0.425

            WHERE FuelMYGroupID IN (2004, 2005, 2006, 2007, 20082009,
20102050) and
                  Sulfur > 30.0;


-- SulfAdj2 is the fractional increase in emissions of the sulfur
compared with the base
--          sulfur level 30 ppm.  SulfMax is the increase due to the
sulfur irreversibility
--          effects.

UPDATE      PredictiveSulfurHCNOxHigh
      SET   FinalSulfAdj      =     (IRFactor * SulfMax) +  (1.0 -
IRFactor) * SulfAdj2;

UPDATE      PredictiveSulfurHCNOxHigh     SET   FinalSulfAdj      =
      1 + FinalSulfAdj;



-- This bottoms the sulfur effects at levels around 4 ppm Sulfur.
Lower gasoline sulfur
--          levels will have no effect on the emissions.  This cap is
needed because the
--          EXTRAPOLATED relationship is log-log in nature.

-- Currently the lowest default fuel sulfur level is 10 ppm S.


UPDATE      PredictiveSulfurHCNOxHigh     SET   FinalSulfAdj
      =     0.50

      WHERE FinalSulfAdj < 0.50;
```

```
--************************
--    Sulfur GPA Effects   *
--************************

-- Sulfur GPA is only applied to FuelMYGroupIDs of 2004, 2005 and 2006.
It is a phase-out
--          strategy for sulfur in the Rocky Mountain states.

UPDATE      PredictiveSulfurHCNOxHigh     SET   Sulf1000    =
      EXP(sulfurCoeff* LN(1000.0))
                  WHERE FuelMYGroupID IN (2004, 2005, 2006);

UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfGPA           =
      1.00;
UPDATE      PredictiveSulfurHCNOxHigh     SET   SulfGPA           =
      (Sulf1000 - Sulf30) / Sulf30  WHERE
                  FuelMYGroupID IN (2004, 2005, 2006) and Sulfur >
30.0;


UPDATE      PredictiveSulfurHCNOxHigh     SET   GPASulfAdj  =
      FinalSulfAdj;

UPDATE      PredictiveSulfurHCNOxHigh     SET   GPASulfAdj  =
                           (IRFactor * SulfGPA) +   (1.0 - IRFactor)
* SulfAdj2
                  WHERE FuelMYGroupID IN (2004, 2005, 2006) and Sulfur
> 30.0;

UPDATE      PredictiveSulfurHCNOxHigh     SET   GPASulfAdj  =     1.0 +
GPASulfAdj  WHERE
                  FuelMYGroupID IN (2004, 2005, 2006) and Sulfur >
30.0;




-- MOBILE6.2 sulfur effects assume a 50/50 split between Highs and
Normals in weighting.
--    This is an extremely high split for modern vehicles in terms
numbers of high emitters
--    in the fleet.  The impact of high emitters might still be 50% of
the total emissions.
--    It was utilized for consistency with MOBILE6.2.  It will reduce
the sensitivity of
--    sulfur to emissions.

drop TABLE if exists PredictiveBlend;
CREATE TABLE PredictiveBlend
SELECT
      B.ProcessID,
      B.PollutantID,
      B.FuelMYGroupID,
```

```
      B.FuelFormulationID,
      B.SourceTypeID,
      B.Sulfur,
      B.OtherFactor,
      B.FinalSulfAdj as SulfNorm,
      H.FinalSulfAdj as SulfHigh,
      (B.FinalSulfAdj*0.50+H.FinalSulfAdj*0.50) as FinalSulfAdj2,
      B.GPASulfAdj as GPANorm,
      H.GPASulfAdj as GPAHigh,
      (B.GPASulfAdj*0.50 + H.GPASulfAdj*0.50) as FinalGPA
FROM  PredictiveSulfurHCNOx2 B      INNER JOIN
      PredictiveSulfurHCNOxHigh H
ON    B.PollutantID           =     H.PollutantID           and
      B.ProcessID             =     H.ProcessID             and
      B.FuelMYGroupID         =     H.FuelMYGroupID         and
      B.FuelFormulationID     =     H.FuelFormulationID     and
      B.SourceTypeID          =     H.SourceTypeID
GROUP BY    B.ProcessID, B.PollutantID, B.SourceTypeID,
B.FuelMYGroupID, B.FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
            PredictiveBlend  (ProcessID, PollutantID, SourceTypeID,
                        FuelMYGroupID, FuelFormulationID);




--
*************************************************************************
*****************
-- Add the reference fuel sulfur effects and compute the final sulfur
adjustment ratio.  *
--          It is ratioed to Arizona's IM program and sulfur of 90 ppm.
*
--
*************************************************************************
*****************


drop TABLE if exists RefSulfur;
CREATE TABLE RefSulfur
SELECT
      ProcessID,
      PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      OtherFactor,
      Sulfur,
      FinalSulfAdj2 as RefSulfurAdj,
      FinalGPA as RefGPA
FROM  PredictiveBlend
WHERE FuelFormulationID = 98
GROUP BY    ProcessID, PollutantID, SourceTypeID, FuelMYGroupID,
FuelFormulationID;
```

```
CREATE UNIQUE INDEX Index1 ON
            RefSulfur  (ProcessID, PollutantID, SourceTypeID,
FuelMYGroupID, FuelFormulationID);




-- The Predictive name was used earlier and is recycled.

drop TABLE if exists Predictive;
CREATE TABLE Predictive
SELECT
      P.ProcessID,
      P.PollutantID,
      P.SourceTypeID,
      P.FuelMYGroupID,
      P.FuelFormulationID,
      P.OtherFactor,
      P.Sulfur,
      P.FinalGPA,
      P.FinalSulfAdj2,
      R.RefSulfurAdj,
      P.FinalSulfAdj2 / R.RefSulfurAdj    as SulfRatio,
      P.OtherFactor *  (P.FinalSulfAdj2 / R.RefSulfurAdj) as
fuelAdjustment
FROM  PredictiveBlend P LEFT JOIN RefSulfur R
ON    P.PollutantID                 =                 R.PollutantID
            and
      P.ProcessID                   =                 R.ProcessID
            and
      P.SourceTypeID                =                 R.SourceTypeID
            and
      P.FuelMYGroupID               =                 R.FuelMYGroupID
GROUP BY    ProcessID, PollutantID, SourceTypeID, FuelMYGroupID,
FuelFormulationID;


CREATE UNIQUE INDEX Index1 ON
            Predictive  (ProcessID, PollutantID, SourceTypeID,
                         FuelMYGroupID, FuelFormulationID);


drop TABLE if exists PredictiveTest;
CREATE TABLE PredictiveTest
SELECT      FuelFormulationOriginal.*,
            P.PollutantID as PID,
            P.ProcessID,
            P.SourceTypeID,
            P.FuelMYGroupID,
            P.FuelFormulationID as FID,
            P.OtherFactor,
            P.FinalGPA,
            P.FinalSulfAdj2,
            P.RefSulfurAdj,
            P.SulfRatio,
```

```
            P.FuelAdjustment
FROM  Predictive P LEFT JOIN FuelFormulationOriginal
ON          P.FuelFormulationID     =
FuelFormulationOriginal.FuelFormulationID
GROUP BY    ProcessID, P.PollutantID, SourceTypeID, FuelMYGroupID,
P.FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
            PredictiveTest  (ProcessID, PID, SourceTypeID,
                        FuelMYGroupID, FuelFormulationID);


alter TABLE Predictive  ADD (
            polProcessID            smallint(6),
            fuelAdjustmentGPA float
);

UPDATE      Predictive  SET   polProcessID            =     101
      WHERE pollutantID = 1 and ProcessID=1;
UPDATE      Predictive  SET   polProcessID            =     301
      WHERE pollutantID = 3 and ProcessID=1;

UPDATE      Predictive  SET   polProcessID            =     102
      WHERE pollutantID = 1 and ProcessID=2;
UPDATE      Predictive  SET   polProcessID            =     302
      WHERE pollutantID = 3 and ProcessID=2;

UPDATE      Predictive  SET   fuelAdjustmentGPA =     fuelAdjustment *
(FinalGPA / FinalSulfAdj2);


drop TABLE if exists FuelAdjustment;
CREATE TABLE FuelAdjustment SELECT * from
MOVESDB20080828.FuelAdjustment;


-- This assumes that polProcessID = 111 is transfered in the
ComplexPredictivemodel script.
--          It should not be duplicated.

DELETE from FuelAdjustment    WHERE polProcessID >= 0;


INSERT INTO FuelAdjustment
(     polProcessID,
      fuelMYGroupID,
      sourceTypeID,
      fuelFormulationID,
      fuelAdjustment,
      fuelAdjustmentCV,
      fuelAdjustmentGPA,
      fuelAdjustmentGPACV     )
SELECT
      polProcessID,
      fuelMYGroupID,
      sourceTypeID,
      fuelFormulationID,
```

```
      fuelAdjustment,
      NULL as fuelAdjustmentCV,
      fuelAdjustmentGPA,
      NULL as fuelAdjustmentGPACV
FROM Predictive;

delete from FuelAdjustment WHERE fuelformulationID = 10;
```

Appendix C MySQL Code for the Complex CO model and the MOBILE6.2 Sulfur Model

```
FLUSH TABLES;
drop Database IF EXISTS RFSComplexCO30_Base;
Create Database RFSComplexCO30_Base;
USE RFSComplexCO30_Base;


--
********************************************************************************
****************
-- This script requires the successful completion of scrips ---

--                 MOVEFuelBinnerRFS_Base.sql
--                 Predictivemodel_RFS30_Base.sql
--                 Predictivemodel_RFS90_Base.sql

--
********************************************************************************
****************


--          Required databases are:

--                 MOVESDB20080828
--                 MOVESFuelRFS_Base


--          Required external text files are:

--                 C:\\Eds C Files\\MOVES
FUELBinner\\ComplexModelCOCoefficients.csv
--                 C:\\Eds C Files\\MOVES FUELBinner\\SQL Fuel Binner
Code\\SulfurCoefficients.csv

--
********************************************************************************
******************
--------------------------------------------------------------------------------
--------------------
--
********************************************************************************
******************




--
********************************************************************************
******************

--    NOTES
```

```
--    This script implements the EPA Complex model for the entire table
of MOVES FuelFormulations.
--    Reference fuel sulfur = 30 ppm  FuelFormulationID = 98

--    C:\\Eds C Files\\MOVES
FUELBinner\\ComplexModelHCNoxCoefficients.csv

--    Output TABLES
--    FuelAdjustment is the final output for this script.  It contains
results for CO only.

--    It contains the Complex model adjustments for everything but
sulfur.
--    A section at the end contains the MOBILE6.2 sulfur adjustment
code.

--    These results will not match results from MOBILE6 or the official
complex model exactly.
--    TECHNOLOGY group weighting were assumed and changed for the 1996
model years.
--    For example, carbureted technology is gone.

--
**********************************************************************
******************



--    Create table containing the Complex Model Coefficients for the
Pollutant CO.
--    These coefficients are by TECH Group (1 - 10).  Group 10 is the
High Emitter Group.
--    coCoefficient numbers are not consistent with other pollutant
coefficient numbers.

drop TABLE if exists ComplexModelCOCoeff;
CREATE TABLE ComplexModelCOCoeff  (
      pollutantID             smallint(6),
      techGroupID             smallint(6),
      coCoefficientID         smallint(6),
      coefficientName         CHAR(12),
      coCoefficient           float,
      PRIMARY KEY             (pollutantID, techGroupID,
coCoefficientID)
);


LOAD DATA INFILE 'C:\\Eds C Files\\MOVES
FUELBinner\\ComplexModelCOCoefficients.csv' REPLACE
        INTO TABLE ComplexModelCOCoeff
        FIELDS TERMINATED BY ','
        OPTIONALLY ENCLOSED BY '"'
        ESCAPED BY '\\'
        LINES TERMINATED BY '\r\n'
        IGNORE 1 LINES;
```

```
CREATE UNIQUE INDEX Index1 ON ComplexModelCOCoeff (pollutantID,
techGroupID, coCoefficientID);




-- Create Table with Complex Model Centering Values.  Coefficients and
units are as follows:

--OXYGEN               (wt%)
--SULFUR               (ppm)
--RVP                        (psi)
--E200                       (%)
--E300                       (%)
--AROMATICS            (vol%)
--OLEFINS              (vol%)
--BENZENE              (vol%)


drop TABLE if exists ComplexModelCOCentering;
CREATE TABLE ComplexModelCOCentering  (
      pollutantID             smallint(6),
      techGroupID             smallint(6),
      c_Oxygen                float,
      c_Sulfur                float,
      c_RVP                   float,
      c_E200                        float,
      c_E300                        float,
      c_AROM                        float,
      c_OLEF                        float,
      c_BENZ                        float,
      PRIMARY KEY             (pollutantID, techGroupID)

);

INSERT INTO ComplexModelCOCentering
      (pollutantID, techGroupID, c_Oxygen, c_Sulfur, c_RVP, c_E200,
c_E300, c_AROM, c_OLEF, c_BENZ ) VALUES
      (2, 1, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825),
      (2, 2, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825),
      (2, 3, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825),
      (2, 4, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825),
      (2, 5, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825),
      (2, 6, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825),
      (2, 7, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825),
      (2, 8, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825),
```

```
      (2, 9, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825),
      (2, 10, 1.774834, 204.577894, 8.6114785, 46.7257717, 85.8961979,
28.2610891, 7.3187162, 1.0666825);

CREATE UNIQUE INDEX Index1 ON ComplexModelCOCentering (pollutantID,
techGroupID);




-- Bring in MOVES Fuel Parameter information.  No diesel fuels.

drop TABLE if exists FFX;
CREATE TABLE FFX
SELECT      FuelFormulationID,
            FuelSubtypeID,
            RVP as P_RVP,
            SulfurLevel as P_Sulfur,
            ETOHVolume,
            MTBEVolume,
            ETBEVolume,
            TAMEVolume,
            aromaticContent as P_AROM,
            olefinContent as P_OLEF,
            benzeneContent as P_BENZ,
            e200 as P_E200,
            e300 as P_E300
FROM MOVESFuelRFS_Base.FuelFormulation
WHERE fuelSubtypeID in (10, 11, 12, 13, 14);


alter TABLE FFX  ADD (
      P_Oxygen               float,
      pollutantID            smallint(6),
      techGroupID            smallint(6)
);

UPDATE FFX  SET   pollutantID = 2;
UPDATE FFX  SET   techGroupID = 1;


--  The Predictive model requires the fuel Oxygen parameter.  The MOVES
--          model contains ETOH and MTBE Volume data.
--          These parameters must be converted.

UPDATE FFX SET    P_Oxygen = 0.0
                  WHERE ETOHVolume <= 0.0 and MTBEVolume <= 0.0;

UPDATE FFX SET    P_Oxygen = ETOHVolume / 2.900
                  WHERE ETOHVolume > 0.0 and MTBEVolume <= 0.0;

UPDATE FFX SET    P_Oxygen = MTBEVolume / 5.600
                  WHERE ETOHVolume <= 0.0 and MTBEVolume > 0.0;
```

```
ALTER TABLE FFX DROP ETOHVolume;
ALTER TABLE FFX DROP MTBEVolume;
ALTER TABLE FFX DROP ETBEVolume;
ALTER TABLE FFX DROP TAMEVolume;

CREATE UNIQUE INDEX Index1 ON FFX (FuelFormulationID);



--
**********************************************************************
**********
-- The fuel sulfur values in the working table FFX are being changed to
30 ppm to *
--          remove the effects of sulfur.  The actual values are being
stored in      *
--          ActualFuelFormulation for later use.
*
--
**********************************************************************
**********

drop TABLE if exists ActualFuelFormulation;
CREATE TABLE ActualFuelFormulation SELECT * from FFX;

CREATE UNIQUE INDEX Index1 ON ActualFuelFormulation
(FuelFormulationID);

UPDATE      FFX   SET   P_Sulfur = 30.0;



drop TABLE if exists FuelFormulation;
CREATE TABLE FuelFormulation
SELECT      PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

INSERT INTO FuelFormulation
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
```

```
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN )
SELECT      PollutantID,
            2 as TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

INSERT INTO FuelFormulation
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN )
SELECT      PollutantID,
            3 as TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

INSERT INTO FuelFormulation
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
```

```
            P_E200,
            P_E300,
            P_OXYGEN )
SELECT      PollutantID,
            4 as TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

INSERT INTO FuelFormulation
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN )
SELECT      PollutantID,
            5 as TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

INSERT INTO FuelFormulation
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN )
```

```
SELECT      PollutantID,
            6 as TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

INSERT INTO FuelFormulation
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN )
SELECT      PollutantID,
            7 as TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

INSERT INTO FuelFormulation
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN )
SELECT      PollutantID,
            8 as TechGroupID,
            FuelFormulationID,
```

```
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

INSERT INTO FuelFormulation
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN )
SELECT      PollutantID,
            9 as TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

INSERT INTO FuelFormulation
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN )
SELECT      PollutantID,
            10 as TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
```

```
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN
FROM FFX;

CREATE UNIQUE INDEX Index1 ON FuelFormulation (FuelFormulationID,
TechGroupID);


drop TABLE if exists C1;
CREATE TABLE C1
SELECT      FuelFormulation.PollutantID,
            FuelFormulation.TechGroupID,
            FuelFormulation.FuelFormulationID,
            FuelFormulation.FuelSubtypeID,
            FuelFormulation.P_RVP,
            FuelFormulation.P_Sulfur,
            FuelFormulation.P_AROM,
            FuelFormulation.P_OLEF,
            FuelFormulation.P_BENZ,
            FuelFormulation.P_E200,
            FuelFormulation.P_E300,
            FuelFormulation.P_OXYGEN,
            ComplexModelCOCentering.c_Oxygen,
            ComplexModelCOCentering.c_Sulfur,
            ComplexModelCOCentering.c_RVP,
            ComplexModelCOCentering.c_E200,
            ComplexModelCOCentering.c_E300,
            ComplexModelCOCentering.c_AROM,
            ComplexModelCOCentering.c_OLEF,
            ComplexModelCOCentering.c_BENZ
FROM FuelFormulation LEFT JOIN ComplexModelCOCentering
ON          FuelFormulation.PollutantID         =
      ComplexModelCOCentering.PollutantID       and
            FuelFormulation.TechGroupID         =
      ComplexModelCOCentering.TechGroupID
GROUP BY PollutantID, TechGroupID, FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON C1 (PollutantID, TechGroupID,
FuelFormulationID);


drop TABLE if exists C2;
CREATE TABLE C2
SELECT      C1.*,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 1
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;
```

```
INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 2
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;
```

```
INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 3
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;


INSERT INTO C2
(           PollutantID,
            TechGroupID,
```

```
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 4
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;


INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
```

```
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 5
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;


INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
```

```
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 6
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;


INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
```

```
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 7
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;


INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
```

```
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 8
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;


INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
```

```
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 9
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;

INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
```

```
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 10
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;

INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
```

```
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 11
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;

INSERT INTO C2
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            c_Oxygen,
            c_Sulfur,
            c_RVP,
            c_E200,
            c_E300,
            c_AROM,
            c_OLEF,
            c_BENZ,
            coCoefficientID,
            CoefficientName,
            coCoefficient  )
SELECT      C1.PollutantID,
            C1.TechGroupID,
            C1.FuelFormulationID,
            C1.FuelSubtypeID,
            C1.P_RVP,
            C1.P_Sulfur,
```

```
            C1.P_AROM,
            C1.P_OLEF,
            C1.P_BENZ,
            C1.P_E200,
            C1.P_E300,
            C1.P_OXYGEN,
            C1.c_Oxygen,
            C1.c_Sulfur,
            C1.c_RVP,
            C1.c_E200,
            C1.c_E300,
            C1.c_AROM,
            C1.c_OLEF,
            C1.c_BENZ,
            ComplexModelCOCoeff.coCoefficientID,
            ComplexModelCOCoeff.CoefficientName,
            ComplexModelCOCoeff.coCoefficient
FROM C1 INNER JOIN ComplexModelCOCoeff
ON          C1.pollutantID    =     ComplexModelCOCoeff.pollutantID and
            C1.techGroupID    =     ComplexModelCOCoeff.techGroupID
WHERE ComplexModelCOCoeff.coCoefficientID = 12
GROUP BY PollutantID, TechGroupID, coCoefficientID, FuelFormulationID;



CREATE UNIQUE INDEX Index1 ON C2 (PollutantID, TechGroupID,
                        coCoefficientID, FuelFormulationID);


alter TABLE C2  ADD (
      P                 float
);



-- Perform primary Complex Model CO calculation.

UPDATE C2   SET   P           =     coCoefficient * (P_Oxygen -
C_Oxygen)   WHERE coCoefficientID = 1;
UPDATE C2   SET   P           =     coCoefficient * (P_Sulfur -
C_Sulfur)   WHERE coCoefficientID = 2;
UPDATE C2   SET   P           =     coCoefficient * (P_RVP - C_RVP)
      WHERE coCoefficientID = 3;
UPDATE C2   SET   P           =     coCoefficient * (P_E200 - C_E200)
      WHERE coCoefficientID = 4;
UPDATE C2   SET   P           =     coCoefficient * (P_E300 - C_E300)
      WHERE coCoefficientID = 5;
UPDATE C2   SET   P           =     coCoefficient * (P_AROM - C_AROM)
      WHERE coCoefficientID = 6;
UPDATE C2   SET   P           =     coCoefficient * (P_OLEF - C_OLEF)
      WHERE coCoefficientID = 7;

UPDATE C2   SET   P           =     coCoefficient * (P_RVP - C_RVP) *
(P_RVP - C_RVP)
                                                WHERE coCoefficientID =
8;
```

```
UPDATE C2   SET   P           =     coCoefficient * (P_E200 - C_E200) *
(P_E200 - C_E200)
                                            WHERE coCoefficientID =
9;
UPDATE C2   SET   P           =     coCoefficient * (P_E300 - C_E300) *
(P_E300 - C_E300)
                                            WHERE coCoefficientID =
10;
UPDATE C2   SET   P           =     coCoefficient * (P_OLEF - C_OLEF) *
(P_OLEF - C_OLEF)
                                            WHERE coCoefficientID =
11;
UPDATE C2   SET   P           =     coCoefficient * (P_E300 -
C_E300)*(P_OLEF - C_OLEF)
                                            WHERE coCoefficientID =
12;



drop TABLE if exists C3;
CREATE TABLE C3
SELECT      PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            SUM(P) as U
FROM C2
GROUP BY PollutantID, FuelFormulationID, TechGroupID;


alter TABLE C3  ADD (
      expU              float
);

UPDATE C3   SET   expU = EXP(U);


CREATE UNIQUE INDEX Index1 ON C3 (PollutantID, TechGroupID,
FuelFormulationID);



-- Reference fuelformulationID = 98       30 ppm Sulfur

drop TABLE if exists CReference;
CREATE TABLE CReference
SELECT      PollutantID,
            TechGroupID,
            FuelFormulationID,
```

```
            FuelSubtypeID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU
FROM C3
WHERE FuelFormulationID = 98
GROUP BY PollutantID, TechGroupID, FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON CReference (PollutantID, TechGroupID,
FuelFormulationID);



drop TABLE if exists C4;
CREATE TABLE C4
SELECT      C3.PollutantID,
            C3.TechGroupID,
            C3.FuelFormulationID,
            C3.FuelSubtypeID,
            C3.P_RVP,
            C3.P_Sulfur,
            C3.P_AROM,
            C3.P_OLEF,
            C3.P_BENZ,
            C3.P_E200,
            C3.P_E300,
            C3.P_OXYGEN,
            C3.expU,
            CReference.expU as RexpU
FROM  C3          LEFT JOIN   CReference
ON          C3.PollutantID          =     CReference.PollutantID
      and
            C3.TechGroupID          =     CReference.TechGroupID
GROUP BY PollutantID, FuelFormulationID, TechGroupID;

CREATE UNIQUE INDEX Index1 ON C4 (PollutantID, TechGroupID,
FuelFormulationID);

--
**********************************************************************
******************
--    Add the TechWeighting Fractions from MOBILE6.2 and the original
Complex Model.
--          The factor for High emitters was changed for all 1996 and
later groups from
--          the unrealistically high level of 55% to 20%.

--    Other Changes in the TechWeighting include:
--    1974 FuelMYGroupID            All CARB and High Emitter
Combinations only.  No significant
```

```
--                                          Electronic fuel injection
penetration at that time.
--
--    1975-93 FuelMYGroupIDs  Default values from Complex Model used.
--
--    1996+ FuelMYGroupIDs    High Emitter level reduced from 55% to
20%.  Difference
--                                          placed into Group 1
--
--    2000+       FuelMYGroupIDs    Groups 2, 3, 6, 7, 8 and 9
eliminated.
--                                          Groups 2, 3, 8 and 9 mapped
into Group 1
--                                          Groups 6 and 7 mapped into
Group 4
--                                          Group 5 left unchanged.
--
--    TechWeighting is NOT a function of SourceTypeID.  Thus,
FuelAdjustments in MOVES will
--          NOT be a function of SourceTypeID (except GPA effects).
--
--    Taken from ATGROUP.FOR
--  Group Tech Fractions:
--  GROUP(I) ---   1: PFI , 3way   , No Air, EGR
--                 2: PFI , 3way   , No Air, No EGR
--                 3: TBI , 3way   , No Air, EGR
--                 4: PFI , 3way+Ox, Air   , EGR
--                 5: PFI , 3way   , Air   , EGR
--                 6: TBI , 3way   , Air   , EGR
--                 7: TBI , 3way+Ox, Air   , EGR
--                 8: TBI , 3way   , No Air, No EGR
--                 9: CARB, 3way+Ox, Air   , EGR
--                10: All High Emitters

--    fuelmygroupid
--          1974
--      19751986
--      19871989
--      19901993
--          1994
--          1995
--          1996
--      19972000
--      20012003
--          2004
--          2005
--          2006
--          2007
--      20082009
--      20102050

--
**************************************************************************
******************

drop TABLE if exists TechWeights;
```

```
CREATE TABLE TechWeights   (
      fuelMYGroupID           INT(8),
      techGroupID             smallint(6),
      techWeighting           float,
      PRIMARY KEY             (fuelMYGroupID, techGroupID)

);
INSERT INTO TechWeights
      (fuelMYGroupID, techGroupID, techWeighting) VALUES
      (1974, 1, 0.00000),
      (1974, 2, 0.00000),
      (1974, 3, 0.00000),
      (1974, 4, 0.00000),
      (1974, 5, 0.00000),
      (1974, 6, 0.00000),
      (1974, 7, 0.00000),
      (1974, 8, 0.00000),
      (1974, 9, 0.44400),
      (1974, 10, 0.55600),
            (19751986, 1, 0.11942),
            (19751986, 2, 0.10688),
            (19751986, 3, 0.08190),
            (19751986, 4, 0.08118),
            (19751986, 5, 0.01260),
            (19751986, 6, 0.00173),
            (19751986, 7, 0.03334),
            (19751986, 8, 0.00000),
            (19751986, 9, 0.00695),
            (19751986, 10, 0.55600),
      (19871989, 1, 0.11942),
      (19871989, 2, 0.10688),
      (19871989, 3, 0.08190),
      (19871989, 4, 0.08118),
      (19871989, 5, 0.01260),
      (19871989, 6, 0.00173),
      (19871989, 7, 0.03334),
      (19871989, 8, 0.00000),
      (19871989, 9, 0.00695),
      (19871989, 10, 0.55600),
            (19901993, 1, 0.11942),
            (19901993, 2, 0.10688),
            (19901993, 3, 0.08190),
            (19901993, 4, 0.08118),
            (19901993, 5, 0.01260),
            (19901993, 6, 0.00173),
            (19901993, 7, 0.03334),
            (19901993, 8, 0.00000),
            (19901993, 9, 0.00695),
            (19901993, 10, 0.55600),
      (1994, 1, 0.47542),
      (1994, 2, 0.10688),
      (1994, 3, 0.08190),
      (1994, 4, 0.08118),
      (1994, 5, 0.01260),
      (1994, 6, 0.00173),
      (1994, 7, 0.03334),
      (1994, 8, 0.00000),
```

```
(1994, 9, 0.00695),
(1994, 10, 0.50000),
      (1995, 1, 0.47542),
      (1995, 2, 0.10688),
      (1995, 3, 0.08190),
      (1995, 4, 0.08118),
      (1995, 5, 0.01260),
      (1995, 6, 0.00173),
      (1995, 7, 0.03334),
      (1995, 8, 0.00000),
      (1995, 9, 0.00695),
      (1995, 10, 0.50000),
(1996, 1, 0.47542),
(1996, 2, 0.10688),
(1996, 3, 0.08190),
(1996, 4, 0.08118),
(1996, 5, 0.01260),
(1996, 6, 0.00173),
(1996, 7, 0.03334),
(1996, 8, 0.00000),
(1996, 9, 0.00695),
(1996, 10, 0.50000),
      (19972000, 1, 0.47542),
      (19972000, 2, 0.10688),
      (19972000, 3, 0.08190),
      (19972000, 4, 0.08118),
      (19972000, 5, 0.01260),
      (19972000, 6, 0.00173),
      (19972000, 7, 0.03334),
      (19972000, 8, 0.00000),
      (19972000, 9, 0.00695),
      (19972000, 10, 0.50000),
(20012003, 1, 0.67115),
(20012003, 2, 0.00000),
(20012003, 3, 0.00000),
(20012003, 4, 0.11625),
(20012003, 5, 0.01260),
(20012003, 6, 0.00000),
(20012003, 7, 0.00000),
(20012003, 8, 0.00000),
(20012003, 9, 0.00000),
(20012003, 10, 0.50000),
      (2004, 1, 0.67115),
      (2004, 2, 0.00000),
      (2004, 3, 0.00000),
      (2004, 4, 0.11625),
      (2004, 5, 0.01260),
      (2004, 6, 0.00000),
      (2004, 7, 0.00000),
      (2004, 8, 0.00000),
      (2004, 9, 0.00000),
      (2004, 10, 0.50000),
(2005, 1, 0.67115),
(2005, 2, 0.00000),
(2005, 3, 0.00000),
(2005, 4, 0.11625),
(2005, 5, 0.01260),
```

```
	(2005, 6, 0.00000),
	(2005, 7, 0.00000),
	(2005, 8, 0.00000),
	(2005, 9, 0.00000),
	(2005, 10, 0.50000),
		(2006, 1, 0.67115),
		(2006, 2, 0.00000),
		(2006, 3, 0.00000),
		(2006, 4, 0.11625),
		(2006, 5, 0.01260),
		(2006, 6, 0.00000),
		(2006, 7, 0.00000),
		(2006, 8, 0.00000),
		(2006, 9, 0.00000),
		(2006, 10, 0.50000),
	(2007, 1, 0.67115),
	(2007, 2, 0.00000),
	(2007, 3, 0.00000),
	(2007, 4, 0.11625),
	(2007, 5, 0.01260),
	(2007, 6, 0.00000),
	(2007, 7, 0.00000),
	(2007, 8, 0.00000),
	(2007, 9, 0.00000),
	(2007, 10, 0.50000),
		(20082009, 1, 0.67115),
		(20082009, 2, 0.00000),
		(20082009, 3, 0.00000),
		(20082009, 4, 0.11625),
		(20082009, 5, 0.01260),
		(20082009, 6, 0.00000),
		(20082009, 7, 0.00000),
		(20082009, 8, 0.00000),
		(20082009, 9, 0.00000),
		(20082009, 10, 0.50000),
	(20102050, 1, 0.67115),
	(20102050, 2, 0.00000),
	(20102050, 3, 0.00000),
	(20102050, 4, 0.11625),
	(20102050, 5, 0.01260),
	(20102050, 6, 0.00000),
	(20102050, 7, 0.00000),
	(20102050, 8, 0.00000),
	(20102050, 9, 0.00000),
	(20102050, 10, 0.50000) ;


CREATE UNIQUE INDEX Index1 ON TechWeights (fuelMYGroupID, techGroupID);



drop TABLE if exists C5;
CREATE TABLE C5
SELECT	C4.PollutantID,
		C4.TechGroupID,
		C4.FuelFormulationID,
```

```
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID = 1974
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;


INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID = 19751986
```

```
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID = 19871989
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
```

```
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID = 19901993
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;


INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
```

```
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID = 1994
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID =  1995
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
```

```
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID = 1996
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
```

```
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID = 19972000
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
```

```
WHERE TechWeights.FuelMYGroupID = 20012003
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID =  2004
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
```

```
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID =  2005
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
```

```
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID =  2006
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID =  2007
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;



INSERT INTO C5
```

```
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID = 20082009
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;


INSERT INTO C5
(           PollutantID,
            TechGroupID,
            FuelFormulationID,
            FuelSubtypeID,
            FuelMYGroupID,
            P_RVP,
            P_Sulfur,
            P_AROM,
            P_OLEF,
            P_BENZ,
            P_E200,
            P_E300,
            P_OXYGEN,
            expU,
            RexpU,
            techWeighting  )
SELECT      C4.PollutantID,
```

```
            C4.TechGroupID,
            C4.FuelFormulationID,
            C4.FuelSubtypeID,
            TechWeights.FuelMYGroupID,
            C4.P_RVP,
            C4.P_Sulfur,
            C4.P_AROM,
            C4.P_OLEF,
            C4.P_BENZ,
            C4.P_E200,
            C4.P_E300,
            C4.P_OXYGEN,
            C4.expU,
            C4.RexpU,
            TechWeights.techWeighting
FROM  C4          LEFT JOIN   TechWeights
ON          C4.TechGroupID          =     TechWeights.TechGroupID
WHERE TechWeights.FuelMYGroupID = 20102050
GROUP BY TechWeights.FuelMYGroupID, C4.PollutantID,
C4.FuelFormulationID, C4.TechGroupID;


alter TABLE C5  ADD (
      COFactorA               float
);

UPDATE C5   SET   COFactorA   =     ((expU / RexpU) - 1) *
techWeighting * 100.0;

CREATE UNIQUE INDEX Index1 ON C5 (FuelMYGroupID, PollutantID,
TechGroupID, FuelFormulationID);



drop TABLE if exists ComplexModelCO;
CREATE TABLE ComplexModelCO
SELECT      C5.PollutantID,
            C5.FuelMYGroupID,
            C5.TechGroupID,
            C5.FuelFormulationID,
            C5.FuelSubtypeID,
            C5.P_RVP as RVP,
            C5.P_Sulfur as Sulfur,
            C5.P_AROM as AROM,
            C5.P_OLEF as OLEF,
            C5.P_BENZ as BENZ,
            C5.P_E200 as E200,
            C5.P_E300 as E300,
            C5.P_OXYGEN as Oxygen,
            (SUM(COFactorA)+ 100)/100 as AdjustmentFactor
FROM C5
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;


CREATE UNIQUE INDEX Index1 ON ComplexModelCO (PollutantID,
FuelMYGroupID, FuelFormulationID);
```

```
drop TABLE if exists dummy;
CREATE TABLE dummy
SELECT      CM.PollutantID,
            CM.FuelMYGroupID,
            CM.FuelFormulationID,
            CM.FuelSubtypeID,
            CM.RVP,
            CM.Sulfur,
            CM.AROM,
            CM.OLEF,
            CM.BENZ,
            CM.E200,
            CM.E300,
            CM.Oxygen,
            CM.AdjustmentFactor
FROM ComplexModelCO as CM
GROUP BY PollutantID, FuelMYGroupID, FuelFormulationID;



ALTER TABLE dummy
     ADD (
            SourceTypeID            smallint(6)
);
UPDATE dummy      SET   SourceTypeID = 11;


drop TABLE if exists ComplexModelCO;
CREATE TABLE ComplexModelCO
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            dummy.SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
```

```
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            21 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            31 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;
```

```
INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            32 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            41 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
```

```
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            42 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
```

```
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            43 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            51 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
```

```
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            52 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            53 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
```

```
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            54 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            61 as SourceTypeID,
            dummy.FuelSubtypeID,
```

```
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;


INSERT INTO ComplexModelCO
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      RVP,
      Sulfur,
      AROM,
      OLEF,
      BENZ,
      E200,
      E300,
      Oxygen,
      AdjustmentFactor )
SELECT      dummy.PollutantID,
            dummy.FuelMYGroupID,
            dummy.FuelFormulationID,
            62 as SourceTypeID,
            dummy.FuelSubtypeID,
            dummy.RVP,
            dummy.Sulfur,
            dummy.AROM,
            dummy.OLEF,
            dummy.BENZ,
            dummy.E200,
            dummy.E300,
            dummy.Oxygen,
            dummy.AdjustmentFactor
FROM dummy
GROUP BY PollutantID, SourceTypeID, FuelMYGroupID, FuelFormulationID;

drop TABLE if exists dummy;

CREATE UNIQUE INDEX Index1 ON
            ComplexModelCO (PollutantID, SourceTypeID, FuelMYGroupID,
FuelFormulationID);



drop TABLE if exists ComplexCO;
CREATE TABLE ComplexCO
SELECT
      C.PollutantID,
```

```
      C.FuelMYGroupID,
      C.FuelFormulationID,
      C.SourceTypeID,
      C.FuelSubtypeID,
      C.RVP,
      A.P_Sulfur as Sulfur,
      C.AROM,
      C.OLEF,
      C.BENZ,
      C.E200,
      C.E300,
      C.Oxygen,
      C.AdjustmentFactor as OtherFactor
FROM  ComplexModelCO C  LEFT JOIN   ActualFuelFormulation A
ON    C.FuelFormulationID           =     A.FuelFormulationID
GROUP BY C.PollutantID, C.SourceTypeID, C.FuelMYGroupID,
C.FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
            ComplexCO (PollutantID, SourceTypeID, FuelMYGroupID,
FuelFormulationID);

drop TABLE if exists ComplexSaved;
CREATE TABLE ComplexSaved SELECT * from ComplexCO;

CREATE UNIQUE INDEX Index1 ON
            ComplexSaved (PollutantID, SourceTypeID, FuelMYGroupID,
FuelFormulationID);



--
**************************************************************************
******************
--
**************************************************************************
******************
--
**************************************************************************
******************
--
**************************************************************************
******************
--
**************************************************************************
******************
--
**************************************************************************
******************
--
**************************************************************************
******************
```

```
--    These sulfur coefficients were taken from the MOBILE6 report by
Tesh Rao.
--    emitterType is "H" for High emitter and "N" for Normal emitter.
--    Some equations are log-log and others are log-linear.

drop TABLE if exists SulfurCoeff;
CREATE TABLE SulfurCoeff  (
      processID                    smallint(6),
      pollutantID                  smallint(6),
      emitterType                  CHAR(1),
      sourceTypeID                 smallint(6),
      fuelMYGroupID                INT(8),
      functionType                 CHAR(10),
      sulfurCoeff                  float,
      PRIMARY KEY                  (processID, pollutantID,
emitterType, sourceTypeID, fuelMYGroupID)

);



LOAD DATA INFILE 'C:\\Eds C Files\\MOVES FUELBinner\\SQL Fuel Binner
Code\\SulfurCoefficients.csv' REPLACE
        INTO TABLE SulfurCoeff
        FIELDS TERMINATED BY ','
        OPTIONALLY ENCLOSED BY '"'
        ESCAPED BY '\\'
        LINES TERMINATED BY '\r\n'
        IGNORE 1 LINES;

CREATE UNIQUE INDEX Index1 ON
            SulfurCoeff  (processID, PollutantID, emitterType,
SourceTypeID, FuelMYGroupID);


--    This section is only the  Normal emitters.  This process was done
piecemeal
--          rather than normals and high together that is why they are
in separate
--          sections.  Rational coding would have done this together.

drop TABLE if exists ComplexCOB;
CREATE TABLE ComplexCOB
SELECT
      C.PollutantID,
      C.FuelMYGroupID,
      C.FuelFormulationID,
      C.SourceTypeID,
      C.FuelSubtypeID,
      C.Sulfur,
      C.OtherFactor,
      SC.processID,
      SC.functionType,
      SC.emitterType,
      SC.sulfurCoeff
FROM  ComplexCO C LEFT JOIN SulfurCoeff SC
ON    C.PollutantID     =     SC.PollutantID                and
```

```
      C.SourceTypeID    =     SC.SourceTypeID               and
      C.FuelMYGroupID   =     SC.FuelMYGroupID
WHERE SC.emitterType = 'N' and SC.processID = 1
GROUP BY    SC.processID, C.PollutantID, C.SourceTypeID,
                  C.FuelMYGroupID, C.FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
            ComplexCOB  (ProcessID, PollutantID, SourceTypeID,
                                    FuelMYGroupID, FuelFormulationID);

drop TABLE if exists Junk;
CREATE TABLE Junk
SELECT
      C.PollutantID,
      C.FuelMYGroupID,
      C.FuelFormulationID,
      C.SourceTypeID,
      C.FuelSubtypeID,
      C.Sulfur,
      C.OtherFactor,
      SC.processID,
      SC.functionType,
      SC.emitterType,
      SC.sulfurCoeff
FROM  ComplexCO C LEFT JOIN SulfurCoeff SC
ON    C.PollutantID     =     SC.PollutantID                and
      C.SourceTypeID    =     SC.SourceTypeID               and
      C.FuelMYGroupID   =     SC.FuelMYGroupID
WHERE SC.emitterType = 'N' and SC.processID = 2
GROUP BY    SC.processID, C.PollutantID, C.SourceTypeID,
                  C.FuelMYGroupID, C.FuelFormulationID;


INSERT INTO ComplexCOB
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      Sulfur,
      OtherFactor,
      processID,
      functionType,
      emitterType,
      sulfurCoeff )
SELECT
      PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      Sulfur,
      OtherFactor,
      processID,
      functionType,
      emitterType,
      sulfurCoeff
```

```
FROM  Junk
GROUP BY    ProcessID, PollutantID, SourceTypeID,
                        FuelMYGroupID, FuelFormulationID;

drop TABLE if exists Junk;



Alter TABLE ComplexCOB  ADD (
      SulfAdj                       float,
      Sulf1                   float,
      Sulf30                        float,
      longTermSulfur          float,
      SulfIRR                       float,
      SulfMax                       float,
      FinalSulfAdj            float,
      IRFactor                float,
      SulfAdj2                float,
      SulfGPA                       float,
      Sulf1000                float,
      GPASulfAdj              float,
      AdjustmentFactorAll     float
);



--*************************************************************
--    Calculate Short Term Sulfur Effects for Normal Emitters   *
--*************************************************************

-- Tier0 and LEV + vehicles use LOG - LOG algorithm.

UPDATE      ComplexCOB  SET   Sulf1       =     EXP(sulfurCoeff*
LN(Sulfur));
UPDATE      ComplexCOB  SET   Sulf30            =     EXP(sulfurCoeff*
LN(30));



-- Tier1 vehicles use a LOG - Linear algorithm.

UPDATE      ComplexCOB  SET   Sulf1       =     EXP(sulfurCoeff*Sulfur)

                        WHERE       FuelMYGroupID IN (1994, 1995, 1996,
19972000);
UPDATE      ComplexCOB  SET   Sulf30            =
      EXP(sulfurCoeff*30)
                        WHERE       FuelMYGroupID IN (1994, 1995, 1996,
19972000);



-- Used for both Tier0 and LEV

UPDATE      ComplexCOB  SET   SulfAdj           =     0.0   WHERE
      Sulfur = 30;
```

```
UPDATE      ComplexCOB  SET   SulfAdj           =     (Sulf1 - Sulf30)
/ Sulf30    WHERE Sulfur <> 30;



--*************************************
--    Calculate Long Term Sulfur Effects    *
--*************************************

--  Add the long term sulfur effects to the LEV and later vehicles for
sulfur levels greater
--          than 30 ppm.  Sulfur levels below 30 ppm are assumed to
have no long term effects.
--    There is no way to logically extrapolate these effects below 30
ppm.  If 30 ppm has
--          a zero long term sulfur effect then Sulfur < 30 ppm should
not have any effect
--          either.

UPDATE      ComplexCOB  SET   longTermSulfur    =     1.0;

UPDATE      ComplexCOB  SET   longTermSulfur    =     2.36  WHERE
PollutantID = 2 and
            FuelMYGroupID IN (20012003, 2004, 2005, 2006, 2007,
20082009, 20102050) and
            Sulfur > 30.0;



UPDATE      ComplexCOB  SET   SulfAdj2          =
      SulfAdj*LongTermSulfur;



--
**********************************************************************
***********
-- Sulfur irreversibility effects for 2004+ model years and Sulfur > 30
ppm only.  *
--
**********************************************************************
***********

-- compute the normal sulfur irreversibility effects

UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(303))
                  WHERE FuelMYGroupID = 2004;
UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(303))
                  WHERE FuelMYGroupID = 2005;
UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(87))
                  WHERE FuelMYGroupID = 2006;
```

```
UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(87))
                  WHERE FuelMYGroupID = 2007;
UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(80))
                  WHERE FuelMYGroupID IN (20082009, 20102050);


-- compute the sulfur irreversibility effects if the selected sulfur
level is
--          greater than the maximum sulfur level.  Note these may not
exist in the
--          fuel supply table (real world) but the model needs
fueladjustment effects
--          anyway as placeholders.  Making the maximum sulfur level
equal to the
--          actual one assures that the sulfur irreversibility effects
don't get
--          smaller than the actual effects.

UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(sulfur))
                  WHERE FuelMYGroupID = 2004 and Sulfur > 303;
UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(sulfur))
                  WHERE FuelMYGroupID = 2005 and Sulfur > 303;
UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(sulfur))
                  WHERE FuelMYGroupID = 2006 and Sulfur > 87;
UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(sulfur))
                  WHERE FuelMYGroupID = 2007 and Sulfur > 87;
UPDATE      ComplexCOB  SET   SulfIRR           =     EXP(sulfurCoeff*
LN(sulfur))
                  WHERE FuelMYGroupID IN (20082009, 20102050) and
Sulfur > 80;




UPDATE      ComplexCOB  SET   SulfMax           =     0.0;
UPDATE      ComplexCOB  SET   SulfMax           =     (SulfIRR -
Sulf30) / Sulf30        WHERE
                  FuelMYGroupID IN (2004, 2005, 2006, 2007, 20082009,
20102050) and Sulfur > 30.0;


UPDATE      ComplexCOB  SET   IRFactor    =     0.000;
UPDATE      ComplexCOB  SET   IRFactor    =     0.425
            WHERE FuelMYGroupID IN (2004, 2005, 2006, 2007, 20082009,
20102050) and
                    Sulfur > 30.0;


-- SulfAdj2 is the fractional increase in emissions of the sulfur
compared with the base
```

```
--          sulfur level 30 ppm.  SulfMax is the increase due to the
sulfur irreversibility
--          effects.

UPDATE      ComplexCOB
      SET   FinalSulfAdj      =     (IRFactor * SulfMax) +  (1.0 -
IRFactor) * SulfAdj2;


-- Converts the delta increase into a multiplicative factor. (i.e., 30%
increase = 1.30)

UPDATE      ComplexCOB  SET   FinalSulfAdj      =     1 + FinalSulfAdj;



-- This bottoms the sulfur effects at levels around 4 ppm Sulfur.
Lower gasoline sulfur
--          levels will have no effect on the emissions.  This cap is
needed because the
--          EXTRAPOLATED relationship is log-log in nature.

-- Currently the lowest default fuel sulfur level is 10 ppm S.


UPDATE      ComplexCOB  SET   FinalSulfAdj            =     0.50  WHERE
FinalSulfAdj < 0.50;



--************************
--    Sulfur GPA Effects   *
--************************

-- Sulfur GPA is only applied to FuelMYGroupIDs of 2004, 2005 and 2006.
It is a phase-out
--          strategy for sulfur in the Rocky Mountain states.

UPDATE      ComplexCOB  SET   Sulf1000    =     EXP(sulfurCoeff*
LN(1000.0))
                  WHERE FuelMYGroupID IN (2004, 2005, 2006);

UPDATE      ComplexCOB  SET   SulfGPA           =     1.00;
UPDATE      ComplexCOB  SET   SulfGPA           =     (Sulf1000 -
Sulf30) / Sulf30  WHERE
                  FuelMYGroupID IN (2004, 2005, 2006) and Sulfur >
30.0;


UPDATE      ComplexCOB  SET   GPASulfAdj  =     FinalSulfAdj;

UPDATE      ComplexCOB  SET   GPASulfAdj  =     (IRFactor * SulfGPA) +
(1.0 - IRFactor) * SulfAdj2
                  WHERE FuelMYGroupID IN (2004, 2005, 2006) and Sulfur
> 30.0;
```

```
UPDATE      ComplexCOB  SET   GPASulfAdj  =     1.0 + GPASulfAdj  WHERE
                  FuelMYGroupID IN (2004, 2005, 2006) and Sulfur >
30.0;









--*********************************************************************
--*********************************************************************
--*********************************************************************
--*********************************************************************
--                       Sulfur effects for High emitters only
*
--*********************************************************************
--*********************************************************************


drop TABLE if exists ComplexCOHigh;
CREATE TABLE ComplexCOHigh
SELECT
      C.PollutantID,
      C.FuelMYGroupID,
      C.FuelFormulationID,
      C.SourceTypeID,
      C.FuelSubtypeID,
      C.Sulfur,
      C.OtherFactor,
      SC.processID,
      SC.functionType,
      SC.emitterType,
      SC.sulfurCoeff
FROM  ComplexCO C LEFT JOIN SulfurCoeff SC
ON    C.PollutantID     =     SC.PollutantID               and
      C.SourceTypeID    =     SC.SourceTypeID              and
      C.FuelMYGroupID   =     SC.FuelMYGroupID
WHERE SC.emitterType = 'H' and SC.processID = 1
GROUP BY    SC.processID, C.PollutantID, C.SourceTypeID,
                  C.FuelMYGroupID, C.FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
            ComplexCOHigh  (ProcessID, PollutantID, SourceTypeID,
                                    FuelMYGroupID, FuelFormulationID);


drop TABLE if exists Junk;
CREATE TABLE Junk
SELECT
      C.PollutantID,
      C.FuelMYGroupID,
      C.FuelFormulationID,
```

```
      C.SourceTypeID,
      C.FuelSubtypeID,
      C.Sulfur,
      C.OtherFactor,
      SC.processID,
      SC.functionType,
      SC.emitterType,
      SC.sulfurCoeff
FROM  ComplexCO C LEFT JOIN SulfurCoeff SC
ON    C.PollutantID     =     SC.PollutantID               and
      C.SourceTypeID    =     SC.SourceTypeID              and
      C.FuelMYGroupID   =     SC.FuelMYGroupID
WHERE SC.emitterType = 'H' and SC.processID = 2
GROUP BY    SC.processID, C.PollutantID, C.SourceTypeID,
                  C.FuelMYGroupID, C.FuelFormulationID;


INSERT INTO ComplexCOHigh
(     PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      Sulfur,
      OtherFactor,
      processID,
      functionType,
      emitterType,
      sulfurCoeff )
SELECT
      PollutantID,
      FuelMYGroupID,
      FuelFormulationID,
      SourceTypeID,
      FuelSubtypeID,
      Sulfur,
      OtherFactor,
      processID,
      functionType,
      emitterType,
      sulfurCoeff
FROM  Junk
GROUP BY    ProcessID, PollutantID, SourceTypeID,
                        FuelMYGroupID, FuelFormulationID;

drop TABLE if exists Junk;


Alter TABLE ComplexCOHigh  ADD (
      SulfAdj                           float,
      Sulf1                       float,
      Sulf30                            float,
      longTermSulfur              float,
      SulfIRR                           float,
      SulfMax                           float,
      FinalSulfAdj                float,
      IRFactor                    float,
```

```
	SulfAdj2			float,
	SulfGPA				float,
	Sulf1000			float,
	GPASulfAdj			float,
	AdjustmentFactorAll		float
);



--*************************************************************
--    Calculate Short Term Sulfur Effects for High Emitters     *
--*************************************************************

-- All High emitting vehicles use a LOG - Linear algorithm.

UPDATE	ComplexCOHigh	SET	Sulf1	=
	EXP(sulfurCoeff*Sulfur);
UPDATE	ComplexCOHigh	SET	Sulf30		=
	EXP(sulfurCoeff*30);


UPDATE	ComplexCOHigh	SET	SulfAdj		=	0.0	WHERE
	Sulfur = 30;

UPDATE	ComplexCOHigh	SET	SulfAdj		=	(Sulf1 -
Sulf30) / Sulf30  WHERE Sulfur <> 30;




--***************************************
--    Calculate Long Term Sulfur Effects    *
--***************************************

--  Add the long term sulfur effects to the LEV and later vehicles for
sulfur levels greater
--		than 30 ppm.  Sulfur levels below 30 ppm are assumed to
have no long term effects.
--	There is no way to logically extrapolate these effects below 30
ppm.  If 30 ppm has
--		a zero long term sulfur effect then Sulfur < 30 ppm should
not have any effect
--		either.

UPDATE	ComplexCOHigh	SET	longTermSulfur	=	1.0;

UPDATE	ComplexCOHigh	SET	longTermSulfur	=	2.36  WHERE
PollutantID = 2 and
		FuelMYGroupID IN (20012003, 2004, 2005, 2006, 2007,
20082009, 20102050) and
		Sulfur > 30.0;



UPDATE	ComplexCOHigh	SET	SulfAdj2		=
	SulfAdj*LongTermSulfur;
```

```
--
*************************************************************************
***********
-- Sulfur irreversibility effects for 2004+ model years and Sulfur > 30
ppm only.  *
--
*************************************************************************
***********


-- normal sulfur irreversibility effects

UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*303)
                  WHERE FuelMYGroupID = 2004;
UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*303)
                  WHERE FuelMYGroupID = 2005;
UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*87)
                  WHERE FuelMYGroupID = 2006;
UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*87)
                  WHERE FuelMYGroupID = 2007;
UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*80)
                  WHERE FuelMYGroupID IN (20082009, 20102050);


-- compute the sulfur irreversibility effects if the selected sulfur
level is
--          greater than the maximum sulfur level.  Note these may not
exist in the
--          fuel supply table (real world) but the model needs
fueladjustment effects
--          anyway as placeholders.  Making the maximum sulfur level
equal to the
--          actual one assures that the sulfur irreversibility effects
don't get
--          smaller than the actual effects.

UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*sulfur)
                  WHERE FuelMYGroupID = 2004;
UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*sulfur)
                  WHERE FuelMYGroupID = 2005;
UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*sulfur)
                  WHERE FuelMYGroupID = 2006;
UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*sulfur)
                  WHERE FuelMYGroupID = 2007;
```

```
UPDATE      ComplexCOHigh     SET   SulfIRR           =
      EXP(sulfurCoeff*sulfur)
                  WHERE FuelMYGroupID IN (20082009, 20102050);



UPDATE      ComplexCOHigh     SET   SulfMax           =     0.0;
UPDATE      ComplexCOHigh     SET   SulfMax           =     (SulfIRR -
Sulf30) / Sulf30
      WHERE FuelMYGroupID IN (2004, 2005, 2006, 2007, 20082009,
20102050) and Sulfur > 30.0;

UPDATE      ComplexCOHigh     SET   IRFactor    =     0.000;
UPDATE      ComplexCOHigh     SET   IRFactor    =     0.425
      WHERE FuelMYGroupID IN (2004, 2005, 2006, 2007, 20082009,
20102050) and
                    Sulfur > 30.0;


-- SulfAdj2 is the fractional increase in emissions of the sulfur
compared with the base
--          sulfur level 30 ppm.  SulfMax is the increase due to the
sulfur irreversibility
--          effects.

UPDATE      ComplexCOHigh
      SET   FinalSulfAdj      =     (IRFactor * SulfMax) +  (1.0 -
IRFactor) * SulfAdj2;

UPDATE      ComplexCOHigh     SET   FinalSulfAdj      =     1 +
FinalSulfAdj;



--************************
--    Sulfur GPA Effects   *
--************************

-- Sulfur GPA is only applied to FuelMYGroupIDs of 2004, 2005 and 2006.
It is a phase-out
--          strategy for sulfur in the Rocky Mountain states.  No GPA
effects for sulfur levels
--          that are less than 30 ppm.

UPDATE      ComplexCOHigh     SET   Sulf1000    =
      EXP(sulfurCoeff*1000)
                  WHERE FuelMYGroupID IN (2004, 2005, 2006);

UPDATE      ComplexCOHigh     SET   SulfGPA           =     1.00;
UPDATE      ComplexCOHigh     SET   SulfGPA           =     (Sulf1000 -
Sulf30) / Sulf30  WHERE
                  FuelMYGroupID IN (2004, 2005, 2006) and Sulfur >
30.0;

UPDATE      ComplexCOHigh     SET   GPASulfAdj  =     FinalSulfAdj;
```

```
UPDATE      ComplexCOHigh
                  SET   GPASulfAdj  =     (IRFactor * SulfGPA) +  (1.0
- IRFactor) * SulfAdj2
                        WHERE FuelMYGroupID IN (2004, 2005, 2006) and
Sulfur > 30.0;

UPDATE      ComplexCOHigh     SET   GPASulfAdj  =     1 + GPASulfAdj
      WHERE
                  FuelMYGroupID IN (2004, 2005, 2006) and Sulfur >
30.0;




-- MOBILE6.2 sulfur effects assume a 50/50 split between Highs and
Normals in weighting.
--    This is an extremely high split for modern vehicles in terms
numbers of high emitters
--    in the fleet.  The impact of high emitters might still be 50% of
the total emissions.
--    It was utilized for consistency with MOBILE6.2.  It will reduce
the sensitivity of
--    sulfur to emissions.

drop TABLE if exists ComplexBlend;
CREATE TABLE ComplexBlend
SELECT
      B.ProcessID,
      B.PollutantID,
      B.FuelMYGroupID,
      B.FuelFormulationID,
      B.SourceTypeID,
      B.Sulfur,
      B.OtherFactor,
      B.FinalSulfAdj as SulfNorm,
      H.FinalSulfAdj as SulfHigh,
      (B.FinalSulfAdj*0.50+H.FinalSulfAdj*0.50) as FinalSulfAdj2,
      B.GPASulfAdj as GPANorm,
      H.GPASulfAdj as GPAHigh,
      (B.GPASulfAdj*0.50 + H.GPASulfAdj*0.50) as FinalGPA
FROM  ComplexCOB B      INNER JOIN  ComplexCOHigh H
ON    B.PollutantID           =     H.PollutantID           and
      B.ProcessID             =     H.ProcessID             and
      B.FuelMYGroupID         =     H.FuelMYGroupID         and
      B.FuelFormulationID     =     H.FuelFormulationID     and
      B.SourceTypeID          =     H.SourceTypeID
GROUP BY    B.ProcessID, B.PollutantID, B.SourceTypeID,
B.FuelMYGroupID, B.FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
```

```
            ComplexBlend  (ProcessID, PollutantID, SourceTypeID,
FuelMYGroupID, FuelFormulationID);



--
*********************************************************************
*****************
-- Add the reference fuel sulfur effects and compute the final sulfur
adjustment ratio.  *
--          It is ratioed to Arizona's IM program and sulfur of 30 ppm.
*
--
*********************************************************************
*****************


drop TABLE if exists RefSulfur;
CREATE TABLE RefSulfur
SELECT
      ProcessID,
      PollutantID,
      SourceTypeID,
      FuelMYGroupID,
      FuelFormulationID,
      OtherFactor,
      Sulfur,
      FinalSulfAdj2 as RefSulfurAdj,
      FinalGPA as RefGPA
FROM  ComplexBlend
WHERE FuelFormulationID = 98
GROUP BY    ProcessID, PollutantID, SourceTypeID, FuelMYGroupID,
FuelFormulationID;

CREATE UNIQUE INDEX Index1 ON
            RefSulfur  (ProcessID, PollutantID, SourceTypeID,
FuelMYGroupID, FuelFormulationID);



-- The ComplexCO name was used earlier and is recycled.

drop TABLE if exists ComplexCO;
CREATE TABLE ComplexCO
SELECT
      P.ProcessID,
      P.PollutantID,
      P.SourceTypeID,
      P.FuelMYGroupID,
      P.FuelFormulationID,
      P.OtherFactor,
      P.Sulfur,
      P.FinalGPA,
      P.FinalSulfAdj2,
      R.RefSulfurAdj,
```

```
      P.FinalSulfAdj2 / R.RefSulfurAdj    as SulfRatio,
      P.OtherFactor *  (P.FinalSulfAdj2 / R.RefSulfurAdj) as
fuelAdjustment
FROM  ComplexBlend P LEFT JOIN RefSulfur R
ON    P.PollutantID                 =                R.PollutantID
            and
      P.ProcessID                   =                R.ProcessID
            and
      P.SourceTypeID                =                R.SourceTypeID
            and
      P.FuelMYGroupID               =                R.FuelMYGroupID
GROUP BY    ProcessID, PollutantID, SourceTypeID, FuelMYGroupID,
FuelFormulationID;


CREATE UNIQUE INDEX Index1 ON
            ComplexCO  (ProcessID, PollutantID, SourceTypeID,
FuelMYGroupID, FuelFormulationID);




alter TABLE ComplexCO  ADD (
            polProcessID            smallint(6),
            fuelAdjustmentGPA float
);

UPDATE      ComplexCO   SET   polProcessID            =     201
      WHERE pollutantID = 2 and ProcessID =1;
UPDATE      ComplexCO   SET   polProcessID            =     202
      WHERE pollutantID = 2 and ProcessID =2;

UPDATE      ComplexCO   SET   fuelAdjustmentGPA =     fuelAdjustment *
(FinalGPA / FinalSulfAdj2);


drop TABLE if exists FuelAdjustment;
CREATE TABLE FuelAdjustment SELECT * from
MOVESDB20080828.FuelAdjustment;
DELETE from FuelAdjustment    WHERE polProcessID < 100000;



INSERT INTO FuelAdjustment
(     polProcessID,
      fuelMYGroupID,
      sourceTypeID,
      fuelFormulationID,
      fuelAdjustment,
      fuelAdjustmentCV,
      fuelAdjustmentGPA,
      fuelAdjustmentGPACV     )
SELECT
      polProcessID,
      fuelMYGroupID,
      sourceTypeID,
      fuelFormulationID,
```

```
      fuelAdjustment,
      NULL as fuelAdjustmentCV,
      fuelAdjustmentGPA,
      NULL as fuelAdjustmentGPACV
FROM ComplexCO;
```