QUESTION 1:

SELECT

a\_IssueReport.IssueReportId,

a\_IssueReport.ProductSerialNumber,

a\_IssueReport.ReportDate,

a\_TestForm.TestCompleted,

DATEDIFF(DAY,a\_IssueReport.ReportDate,a\_TestForm.TestCompleted) 'Days in System',

a\_Unit.ModelNumber,

CASE

WHEN a\_IssueReport.CustomerId is not null

THEN a\_customer.FirstName +' ' + a\_Customer.LastName

WHEN a\_IssueReport.EmployeeId is not null

THEN a\_Employee.FirstName +' ' + a\_Employee.LastName

WHEN a\_IssueReport.DistributorId is not null

THEN a\_Distributor.Name

END 'Reporter',

CASE

WHEN a\_IssueReport.CustomerId is not null

THEN 'Customer'

WHEN a\_IssueReport.EmployeeId is not null

THEN 'Employee'

WHEN a\_IssueReport.DistributorId is not null

THEN 'Distributor'

END 'Reporter Type',

a\_IssueReportProblem.Description 'Type of Problem',

a\_IssueReport.ProblemDescription 'Description of Problem',

CASE

WHEN a\_IssueReport.InjuryOccurred = 1

THEN 'Yes'

ELSE

'No'

END 'Injury',

CASE

WHEN a\_IssueReport.InjuryOccurred = 1

THEN a\_IssueReport.InjuryDescription

ELSE

'None'

END 'Injury'

FROM

a\_IssueReport

INNER JOIN

a\_TestForm ON a\_IssueReport.IssueReportId = a\_TestForm.IssueReportId

INNER JOIN

a\_Unit ON a\_IssueReport.ProductSerialNumber = a\_Unit.SerialNumber

LEFT JOIN

a\_Customer ON a\_IssueReport.CustomerId = a\_Customer.CustomerId

LEFT JOIN

a\_Employee ON a\_IssueReport.EmployeeId = a\_Employee.EmpId

LEFT JOIN

a\_Distributor ON a\_IssueReport.DistributorId = a\_Distributor.DistributorId

INNER JOIN

a\_IssueReportProblem ON a\_IssueReportProblem.ProblemID = a\_IssueReport.IssueReportProblemId

WHERE

a\_IssueReport.IssueReportId = 5

;

GO

OUTPUT QUESTION 1: CROPPED TO FIT PAGE WIDTH





QUESTION 2:

SELECT

a\_IssueReport.IssueReportId,

a\_IssueReport.ProductSerialNumber,

a\_IssueReport.ReportDate,

a\_TestForm.TestCompleted,

a\_Employee.FirstName + ' ' + a\_Employee.LastName 'Tester',

a\_TestForm.TestDescription,

a\_TestForm.TestResults,

CASE

WHEN a\_TestForm.TestCompleted is NULL

THEN 'NO'

ELSE

'YES'

END 'Test Completed?'

FROM

a\_IssueReport

INNER JOIN

a\_TestForm ON a\_IssueReport.IssueReportId = a\_TestForm.IssueReportId

INNER JOIN

a\_Unit ON a\_IssueReport.ProductSerialNumber = a\_Unit.SerialNumber

LEFT JOIN

a\_Employee ON a\_TestForm.EmpId= a\_Employee.EmpId

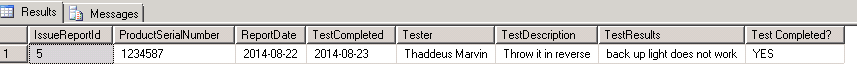
WHERE

a\_IssueReport.IssueReportId = 5

;

GO

OUTPUT QUESTION 2:



QUESTION 3:

/\*

CREATE VIEW av\_ReturnedUnits -- the view was important to merge the data from model and unit. Because

unit contained the fk to th model enitity which held the model description. We need to merge the two so that when we

needed to access the model description from enitities that only had the serial number we were able to do so via the unit entity

or I should say from this view.

\*/

CREATE VIEW av\_ReturnedUnits AS

SELECT

SerialNumber,

a\_Unit.ModelNumber,

a\_Model.[Description],

MSRP

FROM

a\_Unit

INNER JOIN a\_Model ON

a\_Unit.ModelNumber = a\_Model.ModelNumber

;

GO

/\*

CREATE VIEW av\_TestCounts was usefull because it allowed us to count the number of tests completed for each issue report.

\*/

CREATE VIEW av\_TestCounts AS

SELECT

Test.IssueReportId,

COUNT(\*) AS 'TestsConducted'

FROM

a\_TestForm Test

GROUP BY

Test.IssueReportId

;

GO

/\*

CREATE VIEW av\_Number3 was important because it utilized the two prior views and merged their data with the

additional data we needed to complete the queury for problem number 3. By using mulitple view for this problem, we were

able to modularize a large query into smaller more manageable queries. We were also able to extract data from this view

for query number 4 with a where condition.

\*/

CREATE VIEW av\_Number3 AS

SELECT

Report.IssueReportId AS 'Report ID',

Report.ReportDate AS 'Date Made',

Report.ProductSerialNumber AS 'Product Serial Number',

Unit.ModelNumber AS 'Model #',

Unit.[Description] AS 'Model Description',

Problem.[Description] AS 'ProblemType',

ISNULL(TestCounts.TestsConducted, 0) AS 'Number of Tests Performed',

CASE

WHEN Distributor.DistributorId IS NOT NULL

THEN Distributor.Name

WHEN Employee.EmpId IS NOT NULL

THEN Employee.LastName + ', ' + Employee.FirstName

WHEN Customer.CustomerId IS NOT NULL

THEN Customer.LastName + ', ' + Customer.FirstName

END AS 'Reporter Name',

CASE

WHEN Distributor.DistributorId IS NOT NULL

THEN 'Distributor'

WHEN Employee.EmpId IS NOT NULL

THEN 'Employee'

WHEN Customer.CustomerId IS NOT NULL

THEN 'Customer'

END AS 'ReporterType'

FROM

a\_IssueReport Report

INNER JOIN a\_IssueReportProblem Problem ON

Report.IssueReportProblemId = Problem.[ProblemID]

INNER JOIN av\_ReturnedUnits Unit ON

Report.ProductSerialNumber = Unit.SerialNumber

LEFT OUTER JOIN av\_TestCounts TestCounts ON

Report.IssueReportId = TestCounts.IssueReportId

LEFT OUTER JOIN a\_Distributor Distributor ON

Report.DistributorId = Distributor.DistributorId

LEFT OUTER JOIN a\_Customer Customer ON

Report.CustomerId = Customer.CustomerId

LEFT OUTER JOIN a\_Employee Employee ON

Report.EmployeeId = Employee.EmpId

;

GO

SELECT

\*

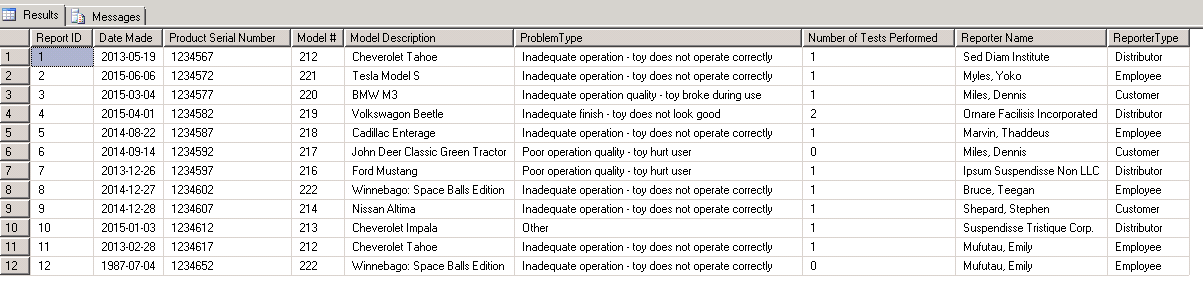
FROM

av\_Number3

;

GO

OUTPUT QUESTION 3:



QUESTION 4:

SELECT

\*

FROM

av\_Number3

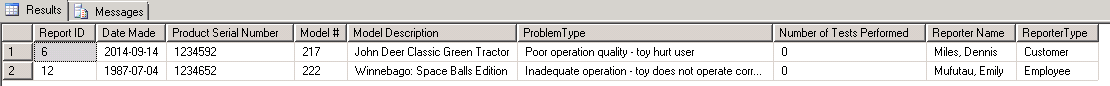
WHERE

av\_Number3.[Number of Tests Performed] = 0

;

GO

OUTPUT QUESTION 4:



QUESTION 5:

/\*

CREATE VIEW av\_Number5 wasn't really important for us to create for query number five, however looking ahead to query number six, we knew we had

to extract information from within this query for a particular condition so we decided to store the query as a view. This enabled

us to utilize the query again without having to copy and pasting the same code.

\*/

CREATE VIEW av\_Number5 AS

SELECT

Report.IssueReportId AS 'Report ID',

Report.ReportDate AS 'Date Made',

Report.ProductSerialNumber AS 'Product Serial Number',

Unit.ModelNumber AS 'Model #',

Unit.[Description] AS 'Model Description',

Problem.[Description] AS 'ProblemType',

ISNULL(TestCounts.TestsConducted, 0) AS 'Number of Tests Performed',

CASE

WHEN Distributor.DistributorId IS NOT NULL

THEN Distributor.Name

WHEN Employee.EmpId IS NOT NULL

THEN Employee.LastName + ', ' + Employee.FirstName

WHEN Customer.CustomerId IS NOT NULL

THEN Customer.LastName + ', ' + Customer.FirstName

END AS 'Reporter Name',

CASE

WHEN Distributor.DistributorId IS NOT NULL

THEN 'Distributor'

WHEN Employee.EmpId IS NOT NULL

THEN 'Employee'

WHEN Customer.CustomerId IS NOT NULL

THEN 'Customer'

END AS 'ReporterType'

FROM

a\_IssueReport Report

INNER JOIN a\_IssueReportProblem Problem ON

Report.IssueReportProblemId = Problem.[ProblemID]

INNER JOIN av\_ReturnedUnits Unit ON

Report.ProductSerialNumber = Unit.SerialNumber

LEFT OUTER JOIN av\_TestCounts TestCounts ON

Report.IssueReportId = TestCounts.IssueReportId

LEFT OUTER JOIN a\_Distributor Distributor ON

Report.DistributorId = Distributor.DistributorId

LEFT OUTER JOIN a\_Customer Customer ON

Report.CustomerId = Customer.CustomerId

LEFT OUTER JOIN a\_Employee Employee ON

Report.EmployeeId = Employee.EmpId

WHERE

Report.IssueReportId IN (

SELECT

TF.IssueReportId

FROM

a\_TestForm TF

WHERE

TF.TestCompleted IS NULL

)

;

GO

SELECT

\*

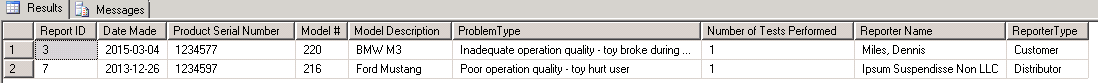
FROM

av\_Number5

;

GO

OUTPUT QUESTION 5:



QUESTION 6:

SELECT

\*

FROM

av\_Number5 OuterNumber5

WHERE

OuterNumber5.[Date Made] = (

SELECT

MIN(InnerNumber5.[Date Made])

FROM

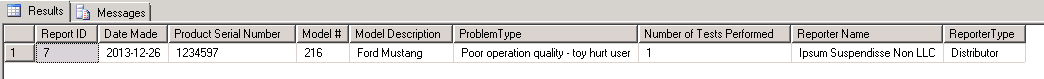
av\_Number5 InnerNumber5

)

;

GO

OUTPUT QUESTION 6:



QUESTION 7:

/\*

CREATE VIEW v\_ModelReportCount, v\_ModelTestCount, v\_ModelInjuryCount, and v\_ModelReportDate were useful because they allowed us

to summerize the data for each model number and description. By breaking up the query into multiple views we were able to gather

specific data (such as count of probs,tests,problem reports) for each vehicle model. This allowed for more logical approach to the ultimate

query and we were able to debug any errors we had in our enity attributes as we went. The views also allowed us to easily use aggregate functions

without the complexity of having to keep track of all the joins or groups.

\*/

select \* from v\_ModelReportCount

select \* from v\_ModelTestCount

CREATE VIEW v\_ModelReportCount AS

SELECT

a\_Unit.ModelNumber,

Count(\*) 'Report Count'

FROM

a\_IssueReport

LEFT JOIN a\_Unit

ON a\_Unit.SerialNumber = a\_IssueReport.ProductSerialNumber

GROUP BY a\_Unit.ModelNumber

;

GO

CREATE VIEW v\_ModelTestCount AS

SELECT

a\_Unit.ModelNumber,

Count(\*) 'Test Count'

FROM

a\_TestForm

LEFT JOIN a\_IssueReport

ON a\_IssueReport.IssueReportId = a\_TestForm.IssueReportId

LEFT JOIN a\_Unit

ON a\_IssueReport.ProductSerialNumber = a\_Unit.SerialNumber

Group BY a\_Unit.ModelNumber

;

GO

CREATE VIEW v\_ModelInjuryCount AS

SELECT

a\_Unit.ModelNumber,

COUNT(\*) 'Injury Count'

FROM

a\_IssueReport

LEFT JOIN a\_Unit

on a\_IssueReport.ProductSerialNumber = a\_Unit.SerialNumber

WHERE a\_IssueReport.InjuryOccurred = 1

Group BY a\_Unit.ModelNumber

;

GO

CREATE VIEW v\_ModelReportDate AS

SELECT

a\_IssueReport.ProductSerialNumber,

a\_Unit.ModelNumber,

a\_issueReport.ReportDate

FROM

a\_IssueReport

LEFT JOIN a\_Unit

on a\_IssueReport.ProductSerialNumber = a\_Unit.SerialNumber

;

GO

/\*

CREATE VIEW v\_ModelSummary, this view allowed us to bring all the data from the four prior views into one

qeury and summarize all the data for each model into one place. We also knew we needed to use this data for query number eight

so it made sense to create a view that we could reference down the road.

\*/

CREATE VIEW v\_ModelSummary AS

SELECT

a\_Model.ModelNumber,

a\_Model.[Description],

ISNULL(v\_ModelReportCount.[Report Count],0) 'Count of Problem Reports',

ISNULL(v\_ModelTestCount.[Test Count],0) 'Count of Tests',

ISNULL(v\_ModelInjuryCount.[Injury Count],0) 'Count of Injury Problem Reports',

(SELECT MIN(v\_ModelReportDate.ReportDate) FROM v\_ModelReportDate WHERE a\_Model.ModelNumber = v\_ModelReportDate.ModelNumber) 'Earliest Problem Report Date',

(SELECT MAX(v\_ModelReportDate.ReportDate) FROM v\_ModelReportDate WHERE a\_Model.ModelNumber = v\_ModelReportDate.ModelNumber) 'Most Recent Problem Report Date'

FROM

a\_Model

LEFT JOIN v\_ModelReportCount

ON a\_Model.ModelNumber = v\_ModelReportCount.ModelNumber

LEFT JOIN v\_ModelTestCount

ON a\_Model.ModelNumber = v\_ModelTestCount.ModelNumber

LEFT JOIN v\_ModelInjuryCount

ON a\_Model.ModelNumber = v\_ModelInjuryCount.ModelNumber

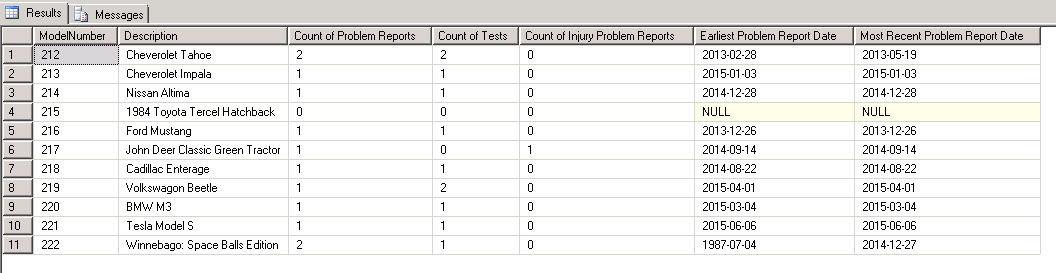
GROUP BY

a\_Model.ModelNumber,a\_Model.[Description],v\_ModelReportCount.[Report Count],v\_ModelTestCount.[Test Count],v\_ModelInjuryCount.[Injury Count]

;

GO

OUTPUT QUESTION 7:



QUESTION 8:

/\*

CREATE VIEW v\_ModelMinReport and v\_ModelMaxReport allowed us to complete this query systematically. We originally tried to

write the query for number eight without any views and it got really messy, so we opted to use the modular approach once again and do

views so we could see what the results as we went. Due to the nature of query number eight, we concluded that doing views to determine

which vehilce had the minimum amount of issues (as in v\_ModelminReport) and the maximimum amount of issues (as in v\_ModelMaxReport)

was appropriate because each of these views required a sub query to compute the min and max report vehicles. These views were helpful

because for the completion of query number eight we just had to create a select statment that took the values produced from the two

views and ORed (unioned) the results to complete the final query.

\*/

CREATE VIEW v\_ModelMinReport AS

SELECT

v\_ModelSummary.ModelNumber,

v\_ModelSummary.Description,

v\_ModelSummary.[Count of Problem Reports]

FROM

v\_ModelSummary

WHERE

v\_ModelSummary.[Count of Problem Reports] = (SELECT MIN(v\_ModelSummary.[Count of Problem Reports]) FROM v\_ModelSummary)

;

GO

CREATE VIEW v\_ModelMaxReport AS

SELECT

v\_ModelSummary.ModelNumber,

v\_ModelSummary.Description,

v\_ModelSummary.[Count of Problem Reports]

FROM

v\_ModelSummary

WHERE

v\_ModelSummary.[Count of Problem Reports] = (SELECT MAX(v\_ModelSummary.[Count of Problem Reports]) FROM v\_ModelSummary)

;

GO

SELECT

\*,

'Min' AS MinOrMax

FROM

v\_ModelMinReport

UNION

SELECT

\*,

'Max' AS MinOrMax

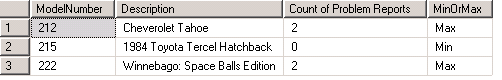
FROM

v\_ModelMaxReport

;

GO

OUTPUT QUESTION 8:



QUESTION 9:

/\*

CREATE VIEW av\_ReporterTypeFrequency was useful because it allowed us to count the number of issue reports made by

each entity (customer, distributor, employee). From this we could create a subquery that just pulled the entity that

filed the most reports. This query was useful in keeping the whole query simple. Like most other views we have used in

this project, the purpose was to be able to see our logic and program in modular fashion.

\*/

CREATE VIEW av\_ReporterTypeFrequency AS

SELECT

av\_Number3.ReporterType,

COUNT(\*) AS 'Frequency'

FROM

av\_Number3

GROUP BY

av\_Number3.ReporterType

;

GO

SELECT

RTF.ReporterType,

RTF.Frequency

FROM

av\_ReporterTypeFrequency RTF

WHERE

RTF.Frequency = (

SELECT

MAX(InnerRTF.Frequency)

FROM

av\_ReporterTypeFrequency InnerRTF

)

;

GO

OUTPUT QUESTION 9:

