

# CA1: Database Design & Development

Module Title: Database Design & Development

Module Code: B8IT113

Module Leader: Jennifer Byrne

Student Name: Gino Ruperez

Student Code: 10577301

Contents

[CA1: Database Design & Development 1](#_Toc72496633)

[1. Project Overview/Scope 3](#_Toc72496634)

[2. Entity Relationship Diagram 4](#_Toc72496635)

[3. Assumptions Made 5](#_Toc72496636)

[4. Data Dictionary 7](#_Toc72496637)

[5. Technology Used 9](#_Toc72496638)

[6. Test Plan 10](#_Toc72496639)

[7. Reflections on Learning 16](#_Toc72496640)

[8. References 16](#_Toc72496641)

[9. SQL 16](#_Toc72496642)

# Project Overview/Scope

This project provides a technical design and provision of a new database server and the development of a new database for the Car Dealership Company. This is the initial phase of design working around the Car, Customer, Employee (Sales Person or Mechanic), Service, Service Ticket & Invoice.

Summary of requirements:

* A Salesperson can sell many cars but each car is sold only by one Salesperson.
* A Salesperson writes a single invoice for each car he or she sells.
* Customer can buy many cars, but each car is bought by only one customer.
* Customer receives sales invoice for every purchase.
* Customer can come for service or repair and will receive a service ticket for each service or repair per car. Customer is not necessarily a car buyer to avail of a service or repair.
* A car brought for service or repair will have an assigned mechanic and every mechanic can be assigned to one or many service ticket.
* GDPR compliant – customers may have an option to retain or anonymise their personal info from Car Dealership database
* Parameterized Stored Procedures (Customer, Car ,Employee and Services or Repairs)
* Data Views for Inactive Customers and Single view on all Mechanics, Tickets and Cars worked on.

Out of Scope:

This phase does not include HR related element around the Employee, this may be a new phase or indeed a new off the shelf system bought by the company. This has yet to be decided but is outside the scope of this piece of work. A Customer system may also be looked at in the future.

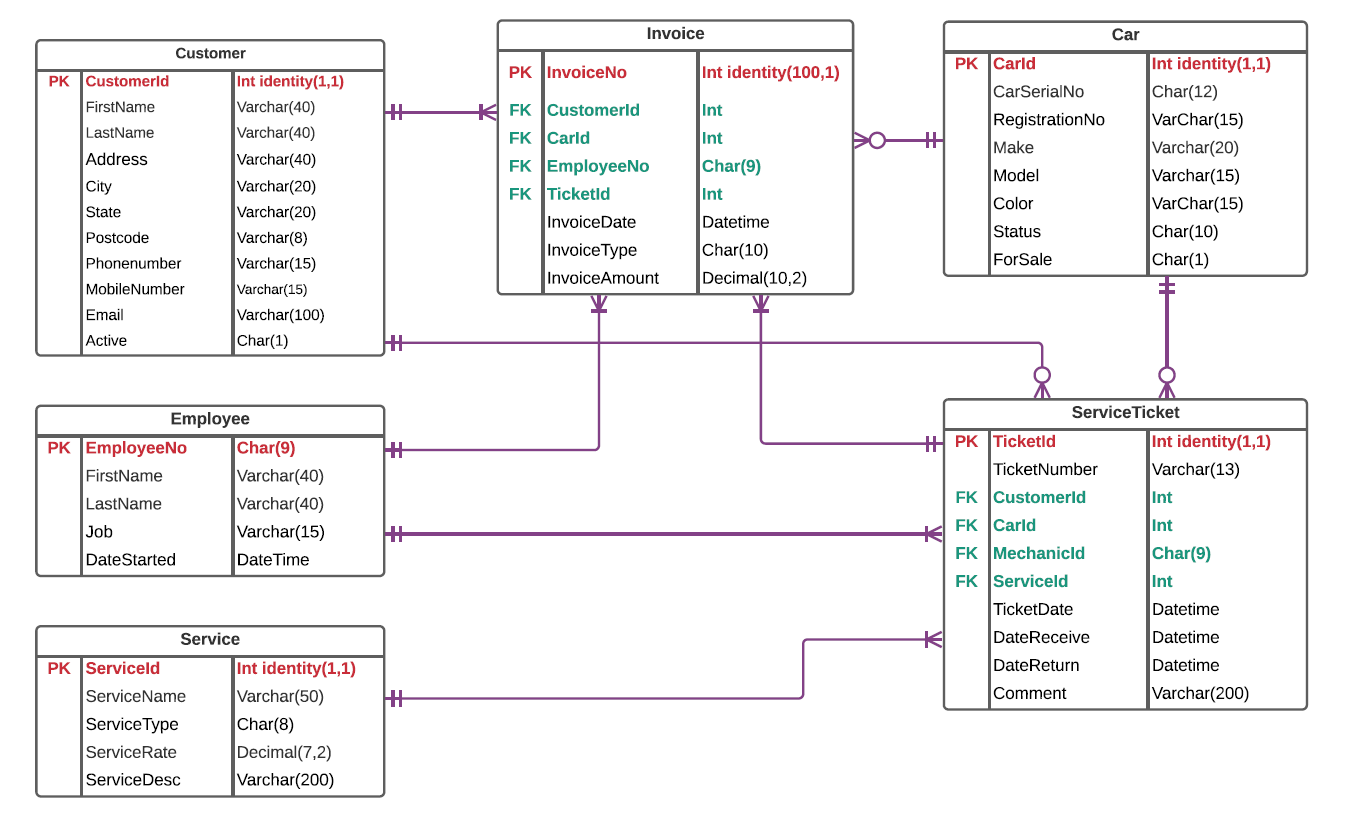
Entities

1. Car
2. Customer
3. Employee (Sales Person or Mechanic)
4. Invoice
5. Service Ticket
6. Service

Assumptions:

* Another HR system that will be link to Employee table for complete employee attributes
* Possibility of acquiring a Customer System in the future

# 2. Entity Relationship Diagram



# Assumptions Made

* Over time a car may be sold more than once.
* CarId starts from 1 auto increment by 1 and served as primary key in Car table.
* The CarSerialNo contains a unique alphanumeric value with a max size of 12 characters e.g SER101021.
* Car status column in Car table is used to identify if car is new, used or for service/repair and will have one of the following values.
  + NEW
  + USED
  + SRV – if car is currently for service or repair
* Car ForSale flag in Car table is used to identify if car can be sold or not, will have a value of either Y or N
* CustomerId starts from 1 and increment by 1, served as a primary key in Customer Table.
* Customer uses a flag called Active with value of either Y or N. Customer may request to remove their personal information and set the flag to N. When customer is anonymized the flag is set to N.
* EmployeeNo in the Employee table corresponds to the PPS number of each employee having a length of 9 characters, it is alphanumeric and served as a primary key in Employee table
* Job attribute in Employee table determine if employee is ‘Salesman’ or ‘Mechanic’
* Employee table is limited to Salesman or Mechanic entries as check constraint is added to Job. All employees is assumed to be included in Employee master table of HR system.
* InvoiceNo in Invoice table is an Integer auto increment by 1 starting from 100
* InvoiceType in Invoice table can have a value of either ‘SALESINV’ if it refers to car sale OR ‘SERVICEINV’ if it refers to car service ticket
* MechanicId in ServiceTicket table and EmployeeNo in Sales Invoice are both refer or associated to EmployeeNo in Employee Table
* TicketId from ServiceTicket table starts from 1 increment by 1, served as a primary key.
* TicketNumber from ServiceTicket table is a 13 character length and is a unique alphanumeric , autogenerate to specific format starting from SRV0000000100 increment by 1.
* The ServiceId in Service Ticket table is associated to Service table which determine the type of service/repair to be done with the car.
* Each Service Ticket will have a corresponding Invoice.
* Service table contains the list of services currently available offered by company, this allow more services to be added in the future. Services can be varied. There may have service that is inclusive of parts replacement and repair. Details of service can be put in comment field.
* ServiceId in Service table starts from 1 and auto increment by 1.
* In compliant for the GDPR instead of hard deleting the customer records, anonymization of customer records is applied by putting the text ‘GDPR’ in each customer’s attribute that refers to customer personal information only e.g customer name, address, contact details, etc.
* Marketing email is assumed to be [marketing\_dept@mail.com](mailto:marketing_dept@mail.com) where list of inactive customers can be sent for new offers or deals.
* Parameterized Stored Procedures Created
  + AddNewCustomer – Add New Customer
  + AddCar – Add New Car
  + AddNewEmployee – Add New Employee
  + AddServiceTicket – Add New Service or Repair Ticket
  + CustomerGdpr – GDPR for Customer
* Additional parameterized Stored Procedures, though it was not asked it is assumed to be needed or beneficial.
  + AddInvoice – Create new invoice
  + DeleteCustomer – Can be used to delete customer that have no transactions yet.
  + UpdateCustomer – Can be used to amend specific customer information like first name, last name, address etc. except for customer id.
  + DeleteCar – Can be used to delete car if car is not yet associated to invoice and service ticket table
  + DeleteEmployee - Can be used to delete employee record if it is not yet associated to invoice and service ticket
  + AddService - Used to add new services, with service type as ‘SERVICE’ or ‘REPAIR’
* Data Views Created
  + ViewServiceTicketLastqtr – This SQL View will return all Mechanics, Tickets and Cars worked on in the last quarter or last 3 months it uses DATEDIFF function to calculate the last 3 months from current date.
  + ViewServiceTicketAll – This SQL View will return all Mechanics, Tickets and Cars worked on without filter.
  + ViewInactiveCust – This SQL View for Inactive Customer Data View. It will return all customers who have not bought or serviced their car with them in the last 13 months, again it uses a DATEDIFF function to calculate the last 13 months. Calculated columns created as follows
* InvoiceCount – number of invoice made for customer
* LastInv# - Last invoice number made for customer
* MonthsNoTrans – number of months a customer having no transactions made
  + ViewAllCust – This SQL View will return all Customers regardless of its status, calculated columns InvoiceCount, LastInv# and MonthsNoTrans were added in the view.

# Data Dictionary

**Car**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Datatype | Optional | Description |
| CarId | Int Identity(1,1) | No | Primary key Car ID uses an IDENTITY column starting at 1 and incrementing by 1 |
| CarSerialNo | Char(12) | No | Car Serial Number in the format like “SER101021” and has a UNIQUE value |
| RegistrationNo | Varchar(15) | No | Car’s plate number |
| Make | Vachar(40) | No | Brand of car e.g. Toyota, BMW |
| Model | Varchar(40) | No | Year Model of the Car |
| Color | Varchar(30) | No | Car’s actual colour e.g. Silver Gray, Blue, etc. |
| Status | Char(10) | No | Status can be “NEW, USED, or SRV” as Check constraint |
| ForSale | Char(1) | No | A flag that indicates car is for sale or not, it can have a value of Y-Yes or N-No as Check Constraint |

**Customer**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Datatype | Optional | Description |
| CustomerId | Int Identity(1,1) | No | Primary Key Customer ID uses an IDENTITY column starting at 1 and incrementing by 1 |
| FirstName | Varchar(40) | YES | Customer Fist Name |
| LastName | Varchar(40) | YES | Customer Last Name |
| Address | Varchar(50) | YES | Customer Address |
| City | Varchar(20) | YES | Customer City |
| State | Varchar(20) | YES | Customer State |
| Postcode | Varchar(8) | YES | Customer Postcode |
| PhoneNumber | Varchar(15) | YES | Customer Phone Number |
| MobileNumber | VarChar(15) | YES | Customer Mobile Number |
| Email | VarChar(100) | YES | Customer Email Address |
| Active | Char(1) | No | Check Constraint, Active column can have value of ‘Y’ or ‘N’ to indicate active or inactive Y – Yes and N-No. |

**Employee**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Datatype | Optional | Description |
| EmployeeNo | Char(9) | No | Primary Key EmployeeNo is the PPS number of each employee an alphanumeric having a length of 9 characters. |
| FirstName | Varchar(40) | No | Employee First Name |
| LastName | Varchar(40) | No | Employee Last Name or Surname |
| Job | Varchar(15) | No | Check Constraint, Job can have a value of either SALESMAN or MECHANIC |
| DateStarted | DateTime | No | Date Employee started |

**Invoice**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Datatype | Optional | Description |
| InvoiceNo | Int Identity(1,1) | No | Primary Key which uses an IDENTITY column, starting from 100 incrementing by 1 |
| CustomerId | Int | No | Foreign key associated to Customer Table |
| CarId | Int | No | Foreign key associated to Car Table |
| EmployeeNo | Char(9) | No | Foreign key associated to Employee Table that refers to EmployeeNo |
| TicketId | Int | Yes | Foreign key associated to ServiceTicket |
| InvoiceDate | DateTime | No | Invoice date |
| InvoiceType | Char(10) | Yes | Check constraint and has a value of ‘SALESINV’ refers to car sales or ‘SERVICEINV’ refers to car service ticket |
| InvoiceAmount | Decimal(10,2) | No | Total Amount of Car sold |

**Service**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Datatype | Optional | Description |
| ServiceId | Int Identity(1,1) | No | Primary Key which uses an IDENTITY column, starting from 1 incrementing by 1 |
| ServiceName | Varchar(50) | No | Service Name for example “Tire Repair”, “Oil Change” , “Full Service”, “Basic Service”, etc. |
| ServiceType | Char(8) | No | Check Constraint and value can be REPAIR or SERVICE |
| ServiceRate | Decimal(7,2) | No | Rate of Service |
| ServiceDesc | Varchar(200) | Yes | Detailed description of the service e.g. ‘Full Service includes Oil Change, Tire Check and Car cleaning” |

**ServiceTicket**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Datatype | Optional | Description |
| TicketId | Int identity(1,1) | No | Primary key that uses identity column starting from 1 incrementing by 1 |
| TicketNumber | Varchar(13) | No | Unique Key that uses an alphanumeric number e.g. SRV000000100, SRV000000101, and so on. Auto increment by 1 based on the TicketId. |
| CustomerId | Int | No | Foreign key associated to Customer table CustomerId column |
| CarId | Int | No | Foreign key associated to Car table on CarId column |
| MechanicId | Char(9) | No | Foreign key associated to Employee table EmployeeNo column |
| ServiceId | Int | No | Foreign key associated to Service table on ServiceId column |
| TicketDate | DateTime | No | Ticket Date in the format of  MM/DD/YYY YHH:MM:SS (when entered) |
| DateReceive | DateTime | No | Date Received the car for service/repair in the format of MM/DD/YYYY HH:MM:SS (when entered) |
| DateReturn | DateTime | Yes | Date Returned the car to owner in the format of MM/DD/YYYY when entered. |
| Comment | Varchar(200) | Yes | Mechanic comment on service/repair done |

# Technology Used

* LucidChart - Tools for ERD
* Windows 10 Home Edition - Operating System
* SQL Server Management Studio v.18.8 - Transact SQL
* Microsoft Word 2010 - Documentation Tools
* Moodle - Notes/Videos recordings of lectures
* Acrobat Reader - To view Exported ERD from LucidChart into PDF

# Test Plan

| **Item Tested** | **Test Run** | **Expected Result** | **Actual Result** |
| --- | --- | --- | --- |
| Positive Test :  Complete parameters passed  AddNewCustomer | EXEC AddNewCustomer 'Amanda','Page','123 Meadows St','Celbridge','Leinster','WD3 T38','012546712','0851463657','amandapage@mail.com','Y' | One Record is added to table with autogenerated CustomerId | (1 row affected)  New customer is successfully added! |
| Negative Test :  When executing with less parameters passed  AddNewCustomer | EXEC AddNewCustomer 'Amanda','Page','123 Meadows St','Celbridge','Leinster','WD3 T38','012546712','0851463657','amandapage@mail.com' | Record is not added to table due to lack of parameter passed | Proc generated below error message :  Msg 201, Level 16, State 4, Procedure AddNewCustomer, Line 0 [Batch Start Line 4]  Procedure or function 'AddNewCustomer' expects parameter '@Active', which was not supplied. |
| Positive Test :  UpdateCustomer | EXEC UpdateCustomer 5,'Francis','Perez','','','','','','','francis.perez@mail.com','' | Customer id 5 record is updated on selected fields only FirstName, Lastname and Email | (1 row affected)  Customer ID 5 is successfully Updated! |
| Negative Test :  When non existing customer passed  UpdateCustomer | EXEC UpdateCustomer 9999,'Francis','Perez','','','','','','','francis.perez@mail.com','' | No record is updated because customer id 9999 is not existed | (0 rows affected)  An error has occured! |
| Positive Test :  Prerequisite : Customer Id is not existed in both Invoice and ServiceTicket Table  DeleteCustomer | EXEC DeleteCustomer 6 | Customer record with CustomerId = 6 is deleted from the table | (1 row affected)  Customer ID 6 is successfully deleted! |
| Negative Test:  When executing without parameter  DeleteCustomer | EXEC DeleteCustomer | An error is expected because no parameter is passed | Msg 201, Level 16, State 4, Procedure DeleteCustomer, Line 0 [Batch Start Line 14]  Procedure or function 'DeleteCustomer' expects parameter '@CustomerId', which was not supplied. |
| Negative Test:  When Customer id is not existed in table DeleteCustomer | EXEC DeleteCustomer 9999 | An error is expected because Customer id 9999 is not existed | (0 rows affected)  An error has occured! |
| Positive Test :  Complete parameter passed  AddCar | EXEC AddCar 'SERP04022015', '2015-PORS-4','Porsche','2015-Porsche Cayenne','Crimson Red','SRV','N' | 1 Record is added to the table | (1 row affected)  Car is successfully added! |
| Negative Test:  When adding an existing CarSerialNo  AddCar | EXEC AddCar 'SERP04022015', '2015-PORS-4','Porsche','2015-Porsche Cayenne','Crimson Red','SRV','N' | Error on Constraint violation of Unique key on CarSerialNo | Msg 2627, Level 14, State 1, Procedure AddCar, Line 16 [Batch Start Line 19]  Violation of UNIQUE KEY constraint 'UQ\_\_Car\_\_3ABC51DFF0E38A73'. Cannot insert duplicate key in object 'dbo.Car'. The duplicate key value is (SERH05322021).  The statement has been terminated.  An Error has occured, no record is added! |
| Negative Test:  When adding a wrong car status  AddCar | EXEC AddCar 'SERP04028015', '2015-PORS-4','Porsche','2015-Porsche Cayenne','Crimson Red','SYX','N' | An Error has occured, no record is added! | Msg 547, Level 16, State 0, Procedure AddCar, Line 16 [Batch Start Line 0]  The INSERT statement conflicted with the CHECK constraint "CK\_CarStatus". The conflict occurred in database "CarDealership", table "dbo.Car", column 'Status'.  The statement has been terminated.  An Error has occured, no record is added! |
| Positive Test :  Prerequisite :  CardIs is not existed in both Invoice and ServiceTicket tables  DeleteCar | EXEC DeleteCar 3 | 1 Record is Deteleted with car id = 3 | (1 row affected)  Car ID 3 is successfully deleted! |
| Negative Test :  When deleting that violates the foreignkey  DeleteCar | EXEC DeleteCar 5 | Car id 5 is existed in service ticket table  An Error has occured, record is not deleted! | Msg 547, Level 16, State 0, Procedure DeleteCar, Line 10 [Batch Start Line 2]  The DELETE statement conflicted with the REFERENCE constraint "FK\_\_Invoice\_\_CarId\_\_1269A02C". The conflict occurred in database "CarDelearship", table "dbo.Invoice", column 'CarId'.  The statement has been terminated.  An Error has occured, record is not deleted! |
| Positive Test:  AddEmployee | EXEC AddNewEmployee '3786SDSMX','Alex','Antonio','MECHANIC','06/28/1998' | Record is Added | (1 row affected)  New Employee is successfully added! |
| Negative Test:  When adding an existing employee No.  AddEmployee | EXEC AddNewEmployee '3786SDSMX','Alex','Antonio','MECHANIC','06/28/1998' | Violation of primary key, no record is added | Msg 2627, Level 14, State 1, Procedure AddNewEmployee, Line 15 [Batch Start Line 74]  Violation of PRIMARY KEY constraint 'PK\_\_Employee\_\_7AD0F1B64FC80875'. Cannot insert duplicate key in object 'dbo.Employee'. The duplicate key value is (3786SDSMX).  The statement has been terminated.  An Error has occured, no record is added! |
| Negative Test:  When adding a wrong Job  AddEmployee | EXEC AddNewEmployee '3186SDSMX','Alex','Antonio','MACHINE','06/28/1998' | An Error has occured, no record is added! | Msg 547, Level 16, State 0, Procedure AddNewEmployee, Line 15 [Batch Start Line 3]  The INSERT statement conflicted with the CHECK constraint "CK\_Job". The conflict occurred in database "CarDealership", table "dbo.Employee", column 'Job'.  The statement has been terminated.  An Error has occured, no record is added! |
| Positive Test :  Prerequisite : EmployeeNo not exists in both Invoice and ServiceTicket Table  DeleteEmployee | EXEC DeleteEmployee '3567882QR' | 1 Record with employee id '3567882QR' is deleted | (1 row affected)  Employee No 3567882QR is successfully deleted! |
| Negative Test  When employee not existed in table  DeleteEmployee | EXEC DeleteEmployee '3567882QR' | Error :  No Record is deleted | (0 rows affected)  An Error has occured, record is not deleted! |
| Positive Test :  AddService | EXEC AddService 'Diagnostic Computer Scan', 'SERVICE',110.50,'Services include engine inspection' | One Record is added | (1 row affected)  New service is successfully added! |
| Negative Test :  When type service type is violated  AddService | EXEC AddService 'Diagnostic Computer Scan', 'SERVREP',110.50,'Services include engine inspection' | An error has occured, no record is added! The service type ‘SERVREP’ is not acceptable should be either SERVICE or REPAIR only | Msg 547, Level 16, State 0, Procedure AddService, Line 9 [Batch Start Line 0]  The INSERT statement conflicted with the CHECK constraint "CK\_ServiceType". The conflict occurred in database "CarDealership", table "dbo.Service", column 'ServiceType'.  The statement has been terminated.  An error has occured, no record is added! |
| Positive Test :  AddServiceTicket | EXEC AddServiceTicket 7,7,'4573685DM',4,'5/4/2019','5/10/2021','5/11/2021','NEED FOR REPAIR' | One record is added with autogenerated Ticketid and TicketNumber | (1 row affected)  New Service Ticket is successfully added! |
| Negative Test  When passing additional arguments  AddServiceTicket | EXEC AddServiceTicket 1,7,7,'4573685DM',4,'5/4/2019','5/10/2021','5/11/2021','NEED FOR REPAIR' | No record is added | Msg 8144, Level 16, State 2, Procedure AddServiceTicket, Line 0 [Batch Start Line 10]  Procedure or function AddServiceTicket has too many arguments specified. |
| Positive Test :  AddInvoice | EXEC AddInvoice 7,10,'4573685DM',4,'05/10/2021','SERVICEINV',20 | One record is added | (1 row affected)  New invoice is successfully added! |
| Negative Test:  If Customer id or Car id is not yet existed  AddInvoice | EXEC AddInvoice 7,20,'4573685DM',4,'05/10/2021','SERVICEINV',20 | No record is added because car id = 20 is not yet existed | Msg 547, Level 16, State 0, Procedure AddInvoice, Line 14 [Batch Start Line 13]  The INSERT statement conflicted with the FOREIGN KEY constraint "FK\_\_Invoice\_\_CarId\_\_47D18CA4". The conflict occurred in database "CarDelearship", table "dbo.Car", column 'CarId'.  The statement has been terminated.  An error has occured, no record is added! |
| Positive Test :  CustomerGdpr | EXEC CustomerGdpr 2 | All fields except Customer Id and Active flag will be replaced with text ‘GDPR’ | (1 row affected)  Customer ID 2 is Successfully Anonimized!  Fields below will have the following values  CustomerId = 2  FirstName = GDPR  LastName = GDPR  Address = GDPR  City = GDPR  State = GDPR  Postcode = GDPR  PhoneNumber=GDPR  MobileNumber = GDPR  Email = GDPR  Active =N |
| Negative Test:  CustomerGdpr | EXEC CustomerGdpr 9999 | No Record is anonymized | (0 rows affected)  Error: Invalid Customer ID, No record is anonimized! |

# Reflections on Learning

* It’s quite challenging however I enjoyed doing it and ERD is new to me. I was able to get a good understanding of creating an entity relations diagram through Videos provided from Moodle and I learned to create using LucidChart.
* It’s literally an application of what we learned from Relational Database Management System course.
* It refreshed my previous knowledge on Transact SQL. I was able to find time looking for tips on adding different constraints on table.
* I’ve learned how to create Stored procedures , Data View and implement some basic Error checking.

# References

* Moodle Materials - videos, slides and tutorials
* Online StackOverflow.com
* Code Expert - <https://www.webcodeexpert.com/2015/08/auto-generate-auto-incremented-unique.html>
* Microsoft T-SQL doc - <https://docs.microsoft.com/en-us/sql/t-sql/functions/error-transact-sql?view=sql-server-ver15>
* https://www.w3schools.com/sql/default.asp

## SQL

Please follow the sequence of running the SQL below

***Prerequisite:***

* MS SQL Server
* **CarDealership Database has been created** 
  + Database can be created by executing a sql statement *CREATE DATABASE [databasename]* or you may follow the instruction from <https://docs.microsoft.com/en-us/sql/relational-databases/databases/create-a-database?view=sql-server-ver15>

1. CarDealerSQLTable.sql – This will create all tables
2. CarDealerSQLInsert.sql – This will setup all initial test data, Insert Statements
3. CarDealerViews.sql – This will create all the required Views
4. CarDealerStoredProc.sql – This will create all the required Stored Procedures
5. Test Plan Execution, please refer to [Test Plan](#_Test_Plan) page.

*Optional Run :*

TestPlanStoredProc.sql - This can be run as an alternate for Test Plan stated in Test plan page. If the Test Plan section is completed, there is no need to run this sql.