

# CA1: Database Design and Development

Module Title: Databases Design and Development

Module Code: B8IT113

Module Leader: Jennifer Byrne

Student Name: Victoria Zuro

Student Code: 10584227

Contents

[CA1: Database Design and Development 1](#_Toc4657376)

[1. Project Overview/Scope 3](#_Toc4657377)

[2. Entity Relationship Diagram 4](#_Toc4657378)

[3. Assumptions Made 5](#_Toc4657379)

[4. Data Dictionary 5](#_Toc4657380)

[5. Technology Used 5](#_Toc4657381)

[6. Test Plan 5](#_Toc4657382)

[7. Reflections on Learning 5](#_Toc4657383)

[8. References 6](#_Toc4657384)

[9. SQL 6](#_Toc4657385)

# **Project Overview/Scope**

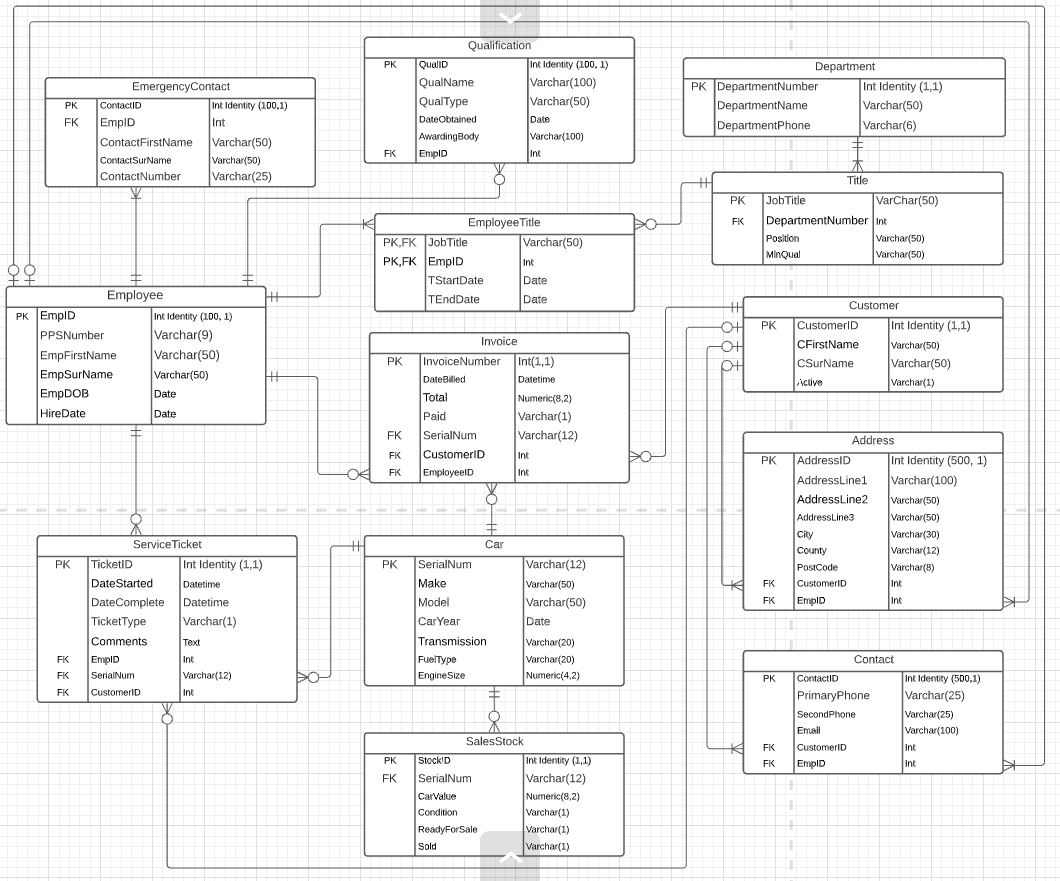
This document outlines the first phase in a multi-step plan to fully integrate all aspects of the automotive company MotorWorks in a single efficient Database System. Phase One includes the technical design and provision of the new database server. At this point in time, it is assumed that all future elements to be included are currently running on separate systems. Later phases will allow for the incorporation of additional departments and business functions at the company, including Human Resources, Marketing, Finance, etc. Future phases will also include upgrades and integration of the payment system. Please see “Assumptions” below for further details of the decisions made according current and future phases.

The current system is set up to allow future integration of the HR department. These additions have been accounted for in the first phase by the initial set up of one overarching employee entity. Phase one has accounted for the most business-critical data relevant to an employee, such as Qualifications and Emergency Contacts. For mechanic liability concerns, it was assumed that qualifications are important to have a record of within this new system. In the second phase, HR data will be integrated, including added entities such as salary. In addition, the future systems will leverage the current Department and Title tables to incorporate a managerial hierarchy system, to add features such as EmployeeReportingTo. Future phases will also include additional stored procedures to allow the company to easily add new and different Job Titles. In this phase, all the relevant/necessary job titles have been added through the included insert statements.

A payment system is also outside of the scope of the current phase. The Invoice entity contains only the most business critical information. Additional details such as subtotals, taxes owed/paid, payment method, date paid, etc. are held in a separate system. For the scope of Phase One there is simply a Y/N flag indicating if payment has been received. In addition, the payment system in relation to Service Tickets is to be implemented later. The payment for the Mechanical side of the business is currently handled separately. On implementation in the future, the invoice entity will require updating to integrate the Sales business with the Mechanic (service ticket) business. This will be required to allow for the ability of customers to drop off their cars without providing details as stated by company policies. This policy is currently represented by the relationship between Customer and ServiceTicket and will be reassessed in the future.

The implementation of Phase One will result in a fully operational Database System for MotorWorks, with all specifications as outlined in the original brief. The system has been designed with future business needs in mind. The inclusion of many different fields in each table, and tables with historical information features will allow for the business to track a significant amount of business relevant date. This can be easily put into additional views or procedures as the business deems fit.

# **Entity Relationship Diagram**



Please see the attached identical photo for convenience in enlarging. Unfortunately, a square design is not optimal for standard paper in either orientation.

# **Assumptions Made**

**Company Policy Assumptions:**

* It was assumed that all the Primary Key attributes using IDENTITY keys follow MotorWorks numbering conventions as outlined in the Data Dictionary.
* Assume that at present each invoice will have ONLY one car. This can be reassessed in the future when changes are made to Invoice to incorporate payments/service tickets.
* Employees can only belong to one department at any one time. While tasks/duties may be completed across departments, each employee will officially be linked to one and only one department as per their job title. If they switch roles within the company the department switch will update in conjunction with the new EmployeeTitle.
* Each job title is unique to a single department. Specifics such as “Sales Manager” or “Accounts Manager” will be included to differentiate all jobs within MotorWorks.
* It was assumed that being fully GDPR compliant extends to employees as well. As employees also have the right to request deletion of their personal information, a stored procedure was created to remove all personal information while preserving business critical information.
* Zero to many relationships were created between Employee and ServiceTicket/Invoice to allow for the fact that an employee is not required to open a ticket or create an invoice. Further, limitations were set to ensure that ONLY employees working within the Mechanical department are permitted to be linked to a service ticket. The requirement of a ServiceTicket and Invoice to have an Employee was captured in the one and only one relationship.
* It was assumed that while multiple mechanics may consult or work on a particular car/ticket, there will be only one employee responsible per ticket. This is to capture the main employee responsible for the ticket. It was assumed that most mechanics are competent and work independently, thus negating any importance of tracking multiple employees.
* A customer dropping off a car for a service may or may not leave personal details. For smaller jobs such as an oil change, etc. many people will not want to leave personal information. For these routine tasks, a time estimate of when the customer is to return will be given, no contact details necessary.
* Before a used car can be sold, it may or may not need the attention of a mechanic. This is captured in the ‘ReadyForSale’ column in SalesStock. These situations will go through the same service ticketing system as customer cars. In the future, upgrades can be made to the system to link ServiceTicket DateCompleted to SalesStock ReadyForSale.
* It was assumed that “qualification type” can change independently of the “qualification name”. For example, the qualification title “Apprentice” can be awarded at multiple QQI levels and equivalents.
* A Job Title may exist without an employee hired for the position. This is in anticipation of the future integration with the HR system.

**Company Needs Assumptions:**

* The Qualification table was included in Phase One (as opposed to later with the HR system implementation) because it was assumed that this is a highly important field for MotorWorks. The business may need to see how many apprentice/fully qualified mechanics there are to comply with legal apprentice restrictions. Not all employees require a qualification, however when adding a new employee if the employee is working in the Mechanical department a qualification is mandatory when using the stored procedure.
* An employee may of course change job titles, it was assumed that the company would want to track this data. This was accounted for by the addition of Start and End dates in the EmployeeTitle table. This table can now be used as a historic table for the business to extract information and trends.
* It was assumed that while a customer my ring ahead to book their car in for a service or repair the date that the booking was made is unimportant. By including only the date that the car was dropped off/the ticket was officially created, MotorWorks will be able to extract more relevant data. The current DateStarted column will allow the company to track how long cars are in the shop from start to finish.
* Assume the mechanical department has an internal system to assign tickets. The system may or may not be integrated at a later date.
* A customer may be connected to an invoice, service ticket, both, or neither. The customer information that was requested by the Marketing department takes all of these scenarios into consideration – to allow the department to best target their promotions.

# **Data Dictionary**

Complete Data Dictionary based on ERD above:

**Employee**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| EmpID | Int | NO | Primary Key which uses an INDENTITY column. Starting at 100, increasing by 1. For example “101”. |
| PPSNumber | Varchar(9) | NO | PPS Number, for example “3246571B”. |
| EmpFirstName | Varchar(50) | NO | First Name, for example “John”. |
| EmpSurName | Varchar(50) | NO | Last Name, for example “Doe”. |
| EmpDOB | Date | NO | Date of birth, for example “1/1/1990”. |
| HireDate | Date | NO | Date employee was hired by the company as per employee contract, for example “2/2/2021”. |

**Qualification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| QualID | Int | NO | Primary Key which uses an IDENTITY Column. Starts at 100 and increases by 1. For example “106”. |
| QualName | Varchar(100) | NO | Stores the name of the qualification relevant to the job. For example “Apprentice”, “Diploma in Vehicle Technology”, “Bachelor in Business – Sales and Marketing”. |
| QualType | Varchar (50) | NO | States the QQI or equivalent of the qualification, for example “QQI Level 6”. |
| DateObtained | Date | NO | The date that the qualification was awarded to the employee. For example “3/3/2005”. |
| AwardingBody | Varchar(100) | NO | The name of the awarding body granting the qualification. For example “SOLAS”, “QQI”. |
| EmpID | Int | NO | Foreign Key to Employee entity. Is an IDENTITY column in Employee starting at 100 and increasing by 1. For example “101”. |

**EmergencyContact**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| ContactID | Int | NO | Primary Key which uses an IDENTITY column, starting at 100 and increasing by 1. For example “102”. |
| EmpID | Int | NO | Foreign Key to Employee entity. Is an IDENTITY column in Employee starting at 100 and increasing by 1. For example “101”. |
| ContactFirstName | Varchar (50) | NO | First name of the emergency contact for employee. For example “Jane” |
| ContactSurName | Varchar(50) | NO | Surname of the emergency contact for employee. For example “Doe” |
| ContactNumber | Varchar (25) | NO | Main phone number of the primary emergency contact. For example “+353083947” |

**Title**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| JobTitle | Varchar(50) | NO | Primary Key stating the Job Title. For example “Sales Person”, “Sales Manager”, “Mechanic”. |
| DepartmentNumber | Int | NO | Foreign Key to Department. Is an IDENTITY column in Department starting at 1 and increasing by 1. For example “3 ”. States which Department each job belongs to. |
| Position | Varchar(50) | NO | This stores the seniority level of the position within the company. For example “Junior”, “Senior”, “Management”. |
| MinQual | Varchar(50) | YES | The minimum qualification necessary to work in the position. For example “Apprentice”, “QQI Level 7”, etc. |

**EmployeeTitle**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| JobTitle | Varchar (50) | NO | Part of a Composite Primary Key. Foreign Key to Title entity. For example “Sales Person”. |
| EmpID | Int | NO | Part of a Composite Primary Key.  Foreign Key to Employee entity. AN IDENTITY column in Employee starting at 100 and increasing by 1. For example “101”. |
| TStartDate | Date | NO | The date in which the employee commenced working in this job position. For example “1/1/2020”. |
| TEndDate | Date | YES | The date in which the employee last worked in the position. Either due to the employee leaving the company or receiving a promotion, etc. For example “12/12/2020”. |

**Department**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| DepartmentNumber | Int | NO | Primary Key. Uses an IDENTITY column starting at 1 and increasing by 1. For example “3”. |
| DepartmentName | Varchar(50) | NO | The name of the department. For example “Sales”, “Marketing”, “Servicing” |
| DepartmentPhone | Varchar(6) | NO | The extension number to directly phone the department. For example “+135”. |

**Customer**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| CustomerID | Int | NO | Primary Key. Uses an Identity column starting at 1 and increasing by 1. For example “8”. |
| CFirstName | Varchar(50) | NO | The first name of the customer for example “Mary”. |
| CSurName | Varchar(50) | NO | The surname of the customer for example “Smith”. |
| Active | Varchar(1) | NO | A “Y” / “N” flag in the system, if the customer is considered active or inactive. |

**Address**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| AddressID | Int | NO | Primary Key which uses an Identity column. Starting at 500 and increasing by 1. For example “505”. |
| AddressLine1 | Varchar(100) | NO | The primary address. For example “Apartment 2 The House” |
| AddressLine2 | Varchar(50) | YES | Additional information pertaining to the primary address. For example “St Street”. |
| AddressLine3 | Varchar(50) | YES | Additional information pertaining to the primary address. For example “Towntown”. |
| City | Varchar(30) | NO | The city of the primary residence. For example “Dublin” |
| County | Varchar(12) | NO | The county of the primary residence. For example “Dublin”. |
| PostCode | Varchar(8) | NO | The postcode of the primary residence. For example “1A2 B3C4”. |
| CustomerID | Int | YES | Foreign Key to Customer entity. Is an Identity column in Customer, starting at 1 and increasing by 1. For example “8”. |
| EmpID | Int | YES | Foreign Key to Employee entity. Is an IDENTITY column in Employee, starting at 100 and increasing by 1. For example “101”. |

**Contact**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| ContactID | Int | NO | Primary Key, uses an Identity column. Starting at 500 and increasing by 1. |
| PrimaryPhone | Varchar(25) | NO | The primary phone number to be contacted on. For example “+353832126938”. |
| SecondPhone | Varchar(25) | YES | The secondary phone number to be contacted on. For example “+353832126938”. |
| Email | Varchar(100) | NO | The primary email address. For example “marysmith@mail.com”. |
| CustomerID | Int | YES | Foreign Key to Customer entity. Is an Identity column in Customer, starting at 1 and increasing by 1. For example “8”. |
| EmpID | Int | YES | Foreign Key to Employee entity. Is an IDENTITY column in Employee, starting at 100 and increasing by 1. For example “101”. |

**Invoice**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| InvoiceNumber | Int | NO | Primary key which uses an IDENTITY column starting at 1 and increasing by 1. For example “10”. |
| DateBilled | Datetime | NO | The date and time the invoice was created. For example “2021-01-01 10:00:00”. |
| Total | Numeric(8,2) | NO | The total amount to be paid. For example “24000”. |
| Paid | Varchar(1) | NO | A “Y” / “N” flag in the system indicating if the invoice has been paid or not. |
| SerialNum | Varchar(12) | NO | Foreign Key to Car Entity. Consists of the 12-Digit VIN number specific to each car. For example “1GNEK12ZX3R2” |
| CustomerID | Int | NO | Foreign Key to Customer entity. Is an Identity column in Customer, starting at 1 and increasing by 1. For example “8”. |
| EmpID | Int | NO | Foreign Key to Employee entity. Is an IDENTITY column in Employee, starting at 100 and increasing by 1. For example “101”. |

**Car**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| SerialNum | Varchar(12) | NO | Primary Key which consists of the 17-Digit VIN number specific to each car. For example “1GNEK12ZX3R2” |
| Make | Varchar(50) | NO | The name of the make of the car. For example “Ford”. |
| Model | Varchar(50) | NO | The name of the model of the car. For example “Focus”. |
| CarYear | Date | NO | This row stores the year that the car was made. Will typically be entered by the user as “2020”, “2008” etc. Will be stored as type DATE, so will appear as “2020-01-01”, “2008-01-01”. Only the year will be used by the company. |
| Transmission | Varchar(20) | NO | The transmission type of the car. For example “Automatic” |
| FuelType | Varchar(20) | NO | The fuel type the car is. For example “Electric”, “Petrol”. |
| EngineSize | Numeric(4,2) | NO | The engine size of the car. For example. For example “1.4” |

**SalesStock**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| StockID | Int | NO | Primary Key which uses and IDENTITY column. Starting at 1 and increasing by 1. For example “6”. |
| SerialNum | Varchar(12) | NO | Foreign Key to Car Entity. Consists of the 17-Digit VIN number specific to each car. For example “1GNEK12ZX3R2” |
| CarValue | Numeric(8,2) | NO | The market value of the car. For example “50,000.00” |
| Condition | Varchar(1) | NO | A flag the show if the car is new or used. “N” for new, “U” for used. |
| ReadyForSale | Varchar(1) | NO | A flag to show if the car is ready for sale, or is waiting to be serviced first, etc. “Y” for ready-for-sale, “N” for not ready. |
| Sold | Varchar(1) | NO | A flag to show if the car has been sold to a customer. “Y” for sold, “N” for not sold. |

**ServiceTicket**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Data Type** | **Optional** | **Description** |
| TicketID | Int | NO | Primary Key which uses an IDENTITY column, starting at 1 and increasing by 1. For example “250”. |
| DateStarted | Datetime | NO | The date and time which the car was brought to the shop. For example “2021-03-03 11:00:00”. |
| DateComplete | Datetime | YES | The date and time when the work was finished and the car was ready for the customer. For example “2021-05-03 12:00:00”. |
| TicketType | Varchar(1) | NO | A flag for the type of ticket. “S” for service, “R” for repair. |
| Comments | Text | YES | Any comments the mechanic deems necessary about any of the work completed, the car, etc. |
| EmpID | Int | NO | Foreign Key to Employee entity. Is an IDENTITY column in Employee, starting at 100 and increasing by 1. For example “101”. |
| SerialNum | Varchar(12) | NO | Foreign Key to Car Entity. Consists of the 17-Digit VIN number specific to each car. For example “1GNEK12ZX3R2” |
| CustomerID | Int | YES | Foreign Key to Customer entity. Is an Identity column in Customer, starting at 1 and increasing by 1. For example “8”. |

# **Technology Used**

Various different technologies were used in the completion of this project. Lucid Chart was used for diagramming prior to the construction of the database itself. SQL Server was then used to script the Transact SQL to create the database and all functions. Microsoft Excel was used throughout the early stages to help visualize the information as it pertains to the database system. Microsoft Word was used for the creating of this final document. Throughout the entire project the internet was used to research and consult, as outlined in Section 8. References (below).

# **Test Plan**

**uspDeleteCustomer**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Type | Test Run | Expected Result | Actual Result |
| Success Test | exec uspDeleteCustomer  @CustomerID = 1,  @CustomerFirstName = ‘Anna’ | Expect all of the fields relevant to the customer in tables: Customer, Address, Contact; to be updated to null values. (Email will be updated to ‘@’ to comply with row check restrictions). Will not affect business critical info such as ‘ServiceTicket’ table. Will display ‘The customer has successfully been deleted’. | Row for relevant parameter was updated (deleted) as expected.  Message: ‘The Customer has successfully been deleted’.  AS EXPECTED |
| Negative Test | exec uspDeleteCustomer  @CustomerID = 100,  @CustomerFirstName = ‘Anna’ | Expect the message to show ‘Customer ID cannot be found’, will not proceed with update.  Will be triggered when the customer ID given is not currently in the database. | Message: ‘Customer ID cannot be found’.  No rows updated.  AS EXPECTED |
| Negative Test | exec uspDeleteCustomer  @CustomerID = null,  @CustomerFirstName = ‘Anna’ | Expect the message to show ‘Cannot delete, must input CustomerID’, will not proceed with update | Message: ‘Cannot delete, must input CustomerID’.  No rows updated.  AS EXPECTED |
| Negative Test | exec uspDeleteCustomer  @CustomerID = 2,  @CustomerFirstName = null | Expect the message to show ‘Cannot delete, must input Customer First Name for validation reasons’, will not proceed with update | Message: ‘Cannot delete, must input Customer First Name for validation reasons’  No rows updated.  AS EXPECTED |
| Negative Test | exec uspDeleteCustomer  @CustomerID = 3,  @CustomerFirstName = ‘Ben’ | Expect the message to show ‘The customerID and customer first name do not match a customer’, will not proceed with update. | Message: ‘The customerID and customer first name do not match a customer’  No rows updated.  AS EXPECTED |

**usp\_DeleteEmployee**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Type | Test Run | Expected Result | Actual Result |
| Success Test | exec uspDeleteEmployee  @EmployeeID = 101,  @EmployeeFirstName = ‘Yuri’ | Expect all of the fields relevant to the employee in tables: Employee, Address, Contact, EmergencyContact, Qualification; to be updated to null values. (Email will be updated to ‘@’ to comply with row check restrictions. PPSNumber will be updated with ‘00000000’ to comply with row check restrictions). Will not affect business critical info such as ‘ServiceTicket’ table. Will display ‘The customer has successfully been deleted’. | Row for relevant parameter was updated (deleted) as expected.  Message: ‘The Employee has successfully been deleted’.  AS EXPECTED |
| Negative Test | exec uspDeleteEmployee  @EmployeeID = 1,  @EmployeeFirstName = ‘Yuri’ | Expect the message to show ‘Employee ID cannot be found’, will not proceed with update.  Will be triggered when the Employee ID given is not currently in the database. | Message: ‘Employee ID cannot be found’.  No rows updated.  AS EXPECTED |
| Negative Test | exec uspDeleteEmployee  @EmployeeID = null,  @EmployeeFirstName = ‘Yuri’ | Expect the message to show ‘Cannot delete, must input EmployeeID’, will not proceed with update | Message: ‘Cannot delete, must input EmployeeID’.  No rows updated.  AS EXPECTED |
| Negative Test | exec uspDeleteEmployee  @EmployeeID = 101,  @EmployeeFirstName = null | Expect the message to show ‘Cannot delete, must input Employee First Name for validation reasons’, will not proceed with update | Message: ‘Cannot delete, must input Employee First Name for validation reasons’  No rows updated.  AS EXPECTED |
| Negative Test | exec uspDeleteEmployee  @EmployeeID = 3,  @EmployeeFirstName = ‘Ben’ | Expect the message to show ‘The employeeID and employee first name do not match an Employee’, will not proceed with update. | Message: ‘The employeeID and employee first name do not match an Employee’  No rows updated.  AS EXPECTED |

**usp\_AddCarToSalesStock**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Type | Test Run | Expected Result | Actual Result |
| Success Test | exec uspAddCarToSalesStock  @SerialNum = '123456789013',  @Make ='Ford',  @Model = 'Focus',  @CarYear = '2020',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.2,  @CarValue = 20000,  @Condition = 'N',  @ReadyForSale ='Y',  @Sold = 'N' | Expect the new car to be added to Car and Sales Stock, with the appropriate Foreign Key in the SalesStock Table.  Display the message ‘The car has successfully been added’ | The car was successfully added to both relevant tables.  Message: ‘The car has successfully been added’  AS EXPECTED |
| Success Test | exec uspAddCarToSalesStock  @SerialNum = '123456789s13',  @Make ='Ford',  @Model = 'Focus',  @CarYear = '2020',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.2,  @CarValue = 20000,  @Condition = 'N',  @ReadyForSale ='Y' | Expect the new car to be added to Car and Sales Stock, with the appropriate Foreign Key in the SalesStock Table.  Test to ensure  DEFAULT ‘N’ in Sold working correctly  Display the message ‘The car has successfully been added’ | The car was successfully added to both relevant tables.  Default Sold working correctly  Message: ‘The car has successfully been added’  AS EXPECTED |
| Negative Test | exec uspAddCarToSalesStock  @SerialNum = '1FUY3LYB6PP4',  @Make ='ford',  @Model = 'focus',  @CarYear = '2020',  @Transmission = 'manual',  @FuelType = 'petrol',  @EngineSize = 2.2,  @CarValue = 20000,  @Condition = 'n',  @ReadyForSale ='y',  @Sold = 'n' | Expect the car to NOT be added. Using a serial number that is already in the database (after running insert statements).  Display the message ‘PROBLEM! Rolling Back the Tran’ | Message: ‘PROBLEM! Rolling Back the Tran’  ErrorMessage ‘Violation of Primary Key constraint’.  No rows added.  AS EXPECTED |
| Negative Test | exec uspAddCarToSalesStock  @SerialNum = 'dhfjtyruehcy',  @Make ='ford',  @Model = 'focus',  @CarYear = '2020',  @Transmission = 'manual',  @FuelType = 'petrol',  @EngineSize = 2.2,  @CarValue = 20000,  @Condition = 'y',  @ReadyForSale ='y',  @Sold = 'n' | Expect the message to show ‘The condition must be entered as ‘N/U’, will not proceed to insert rows | Message: ‘The condition must be entered as ‘N/U’  No rows added.  AS EXPECTED. |
| Negative Test | exec uspAddCarToSalesStock  @SerialNum = null,  @Make ='ford',  @Model = 'focus',  @CarYear = '2020',  @Transmission = 'manual',  @FuelType = 'petrol',  @EngineSize = 2.2,  @CarValue = 20000,  @Condition = 'n',  @ReadyForSale ='y',  @Sold = 'n' | Expect the message to show ‘Cannot add car, no Serial Number given’, will not proceed to insert rows | Message: ‘Cannot add car, no Serial Number given’  No rows added.  AS EXPECTED. |
| Negative Test | exec uspAddCarToSalesStock  @SerialNum = '123456',  @Make ='ford',  @Model = 'focus',  @CarYear = '2020',  @Transmission = 'manual',  @FuelType = 'petrol',  @EngineSize = 2.2,  @CarValue = 20000,  @Condition = 'n',  @ReadyForSale ='y',  @Sold = 'n' | Expect the message to show ‘The serial number must be 12 characters’, will not proceed to insert rows | Message: The serial number must be 12 characters  No rows added.  AS EXPECTED. |
| Negative Test | exec uspAddCarToSalesStock  @SerialNum = '123456789014',  @Make ='ford',  @Model = 'focus',  @CarYear = '2020',  @Transmission = 'manual',  @FuelType = 'petrol',  @EngineSize = 2.2,  @CarValue = 20000,  @Condition = 'n',  @ReadyForSale ='w',  @Sold = 'n' | Expect the message to show ‘Ready for Sale must be entered as Y/N’, will not proceed to insert rows | Message: Ready for Sale must be entered as Y/N  No rows added.  AS EXPECTED. |
| Negative Test | exec uspAddCarToSalesStock  @SerialNum = '123456789015',  @Make ='ford',  @Model = 'focus',  @CarYear = '2020',  @Transmission = 'manual',  @FuelType = 'petrol',  @EngineSize = 2.2,  @CarValue = 20000,  @Condition = 'n',  @ReadyForSale ='y',  @Sold = 'w' | Expect the message to show ‘Sold must be entered as Y/N’, will not proceed to insert rows | Message: Sold must be entered as Y/N  No rows added.  AS EXPECTED. |
| Negative Test | exec uspAddCarToSalesStock  @SerialNum = '123456789015',  @Make ='Ford',  @Model = 'Focus',  @CarYear = '2020',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 0,  @CarValue = 20000,  @Condition = 'n',  @ReadyForSale ='y',  @Sold = 'n' | Expect the message to show ‘The engine size must be greater than 0’, will not proceed to insert rows | Message: The engine size must be greater than 0  No rows added.  AS EXPECTED. |
| Negative Test | exec uspAddCarToSalesStock  @SerialNum = '123456789015',  @Make ='ford',  @Model = 'focus',  @CarYear = '2020',  @Transmission = 'manual',  @FuelType = 'petrol',  @EngineSize = 2.2,  @CarValue = 0,  @Condition = 'n',  @ReadyForSale ='y',  @Sold = 'n' | Expect the message to show ‘The car value must be greater than 0’, will not proceed to insert rows | Message: The car value must be greater than 0  No rows added.  AS EXPECTED. |

**usp\_InsertNewEmployee**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Type | Test Run | Expected Result | Actual Result |
| Success Test | exec uspInsertNewEmployee  @PPSNumber = '4637485A',  @FirstName = 'Tina',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @HireDate = '2021-05-24',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Auto Mechanic',  @QualificationName = 'Apprentice',  @QualType = 'QQI Level 6',  @DateObtained = '2020-03-04',  @AwardingBody = 'QQI',  @Address = 'Testing House',  @AddressLine2 = '100 Testing Lane',  @AddressLine3 = 'Apartment 2',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @SecondPhoneNo = '0836475839',  @Email = 'test@email.com' | Expect all of the fields relevant to the Employee (in tables: Employee, Address, Contact, EmergencyContact, Qualification) to be updated to insert the new Employee.  Will display ‘The employee has successfully been added’ | The employee was added according to the specified parameters in all tables.  Message: ‘The employee has successfully been added’  AS EXPECTED |
| Success Test | exec uspInsertNewEmployee  @PPSNumber = '1234567B',  @FirstName = 'Tommy',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Sales Person',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @Email = 'test@email.com' | Expect all of the fields relevant to the Employee (in tables: Employee, Address, Contact, EmergencyContact, Qualification) to be updated to insert the new Employee.  Will insert all appropriate default and/or values.  Will display ‘The employee has successfully been added’ | The employee was added according to the specified parameters in all tables. Appropriate null and default values added.  Message: ‘The employee has successfully been added’  AS EXPECTED |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '1234567C',  @FirstName = 'Tanya',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Auto Mechanic',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @Email = 'test@email.com' | Expect the message to show ‘An employee working in a mechanical field must have a qualification’, will not proceed to insert rows | Message:  An employee working in a mechanical field must have a qualification  No rows added.  AS EXPECTED. |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '1234567C',  @FirstName = ‘Tammy’,  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Auto Mechanic',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @Email = 'test@email.com' | Expect the message to show ‘Cannot add Employee, must include PPS NO  ’, will not proceed to insert rows | Message:  Cannot add Employee, must include PPS NO  No rows added.  AS EXPECTED. |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '9028365R',  @FirstName = 'Tommy',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Sales Person',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @Email = 'test@email.com' | Expect the message to show ‘This PPS Number alredy exists in the system’, will not proceed to insert rows | Message:  This PPS Number alredy exists in the system  No rows added.  AS EXPECTED. |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '1234567A',  @FirstName = 'Tina',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @HireDate = '2021-05-24',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Auto Mechanic',  @QualificationName = 'Apprentice',  @QualType = 'QQI Level 6',  @DateObtained = '2020-03-04',  @AwardingBody = 'QQI',  @Address = 'Testing House',  @AddressLine2 = '100 Testing Lane',  @AddressLine3 = 'Apartment 2',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @SecondPhoneNo = '0836475839',  @Email = 'test@email.com' | Expect the message to show ‘All fields relating to qualification must be completed if a qualification name is given’, will not proceed to insert rows | Message:  All fields relating to qualification must be completed if a qualification name is given  No rows added.  AS EXPECTED. |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '1234567B',  @FirstName = 'Tommy',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Sales Person',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @Email = 'test@email.com' | Expect the message to show ‘The PPS Number must be at least 8 characters’, will not proceed to insert rows | Message:  The PPS Number must be at least 8 characters  No rows added.  AS EXPECTED. |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '1234567B',  @FirstName = 'Tommy',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Sales Person',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @Email = 'test@email.com' | Expect the message to show ‘The employee must be at least 14 years old to legally work’, will not proceed to insert rows | Message:  The employee must be at least 14 years old to legally work  No rows added.  AS EXPECTED. |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '1234567F',  @FirstName = 'Triangle',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Auto Mechanic',  @QualificationName = 'Apprentice',  @QualType = 'QQI Level 6',  @DateObtained = '2022-03-03',  @AwardingBody = 'QQI',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @Email = 'test@email.com' | Expect the message to show ‘The qualification must have been obtained before today, will not proceed to insert rows | Message:  The qualification must have been obtained before today  No rows added.  AS EXPECTED. |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '1234567G',  @FirstName = 'Taco',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Sales Person',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S',  @PhoneNumber = '0983645273',  @Email = 'test@email.com' | Expect the message to show ‘Please enter a valid postcode’, will not proceed to insert rows | Message:  Please enter a valid postcode  No rows added.  AS EXPECTED. |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '1234567H',  @FirstName = 'Tommy',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Sales Person',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @Email = 'testemail.com' | Expect the message to show ‘Invalid Email address’, will not proceed to insert rows | Message:  Invalid Email address  No rows added.  AS EXPECTED. |
| Negative Test | exec uspInsertNewEmployee  @PPSNumber = '1234567B',  @FirstName = 'Tommy',  @SurName = 'Tester',  @DateOfBirth = '1990-01-01',  @ContactFirstName = 'Terry',  @ContactSurName = 'Tester',  @ContactNumber = '0987364532',  @JobTitle = 'Sales Person',  @Address = 'Testing House',  @City = 'Dublin',  @County = 'Dublin',  @PostCode = 'D82J4S7',  @PhoneNumber = '0983645273',  @Email = 'test email.com' | Expect the message to show ‘Invalid Email address’, will not proceed to insert rows | Message:  Invalid Email address  No rows added.  AS EXPECTED. |

**usp\_NewTicketExistingCar**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Type | Test Run | Expected Result | Actual Result |
| Success Test | exec uspNewTicketExistingCar  @DateStarted = '2021-05-05',  @DateComplete = '2021-05-06',  @TicketType = 'S',  @Comment = 'Service',  @EmployeeID = 100,  @CustomerID = 9,  @SerialNum = '1FADP3K21DL2' | Expect one new ticket to be added according the parameter specifications  Will display  The ticket has successfully been added | One service ticket was added according to the specified parameters in all tables.  Message:  The ticket has successfully been added  1 row added  AS EXPECTED |
| Success Test | exec uspNewTicketExistingCar  @TicketType = 'S',  @EmployeeID = 100,  @SerialNum = '1FADP3K21DL2' | Expect one new ticket to be added according the parameter specifications. Default and null values will all be added correctly.  Will display  The ticket has successfully been added | One service ticket was added according to the specified parameters in all tables.Default and null values all added correctly.  Message:  The ticket has successfully been added  1 row added  AS EXPECTED |
| Negative Test | exec uspNewTicketExistingCar  @TicketType = 'r',  @EmployeeID = 101,  @SerialNum = '1FADP3K21DL2' | Expect the message to show ‘only employees in the mechanical department may open a ticket’, will not proceed to insert rows | Message:  only employees in the mechanical department may open a ticket  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketExistingCar  @TicketType = 'r',  @EmployeeID = 100,  @SerialNum = '1FADP3K21jL2' | Expect the message to show ‘The Serial Number is not in the system. Please try again or use uspNewTicketNewCar to add the car and create a ticket  ’, will not proceed to insert rows | Message:  The Serial Number is not in the system. Please try again or use uspNewTicketNewCar to add the car and create a ticket  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketExistingCar  @TicketType = 'r',  @EmployeeID = 100,  @SerialNum = null | Expect the message to show ‘Cannot add car, no Serial Number given’, will not proceed to insert rows | Message:  Cannot add car, no Serial Number given  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketExistingCar  @TicketType = 'r',  @EmployeeID = 100,  @SerialNum = '1FADP3K21jL' | Expect the message to show ‘The serial number must be 12 characters’, will not proceed to insert rows | Message:  The serial number must be 12 characters  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketExistingCar  @TicketType = 'r',  @EmployeeID = 100,  @SerialNum = '1FADP3K21DL2',  @CustomerID = 500 | Expect the message to show ‘The customer is not in the system’, will not proceed to insert rows | Message:  The customer is not in the system  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketExistingCar  @TicketType = 'r',  @EmployeeID = 100,  @SerialNum = '1FADP3K21DL2',  @CustomerID = 500 | Expect the message to show ‘The ticket type must be "s" for service or "r" for repair  ’, will not proceed to insert rows | Message:  The ticket type must be "s" for service or "r" for repair  No rows added.  AS EXPECTED. |

**usp\_NewTicketNewCar**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Type | Test Run | Expected Result | Actual Result |
| Success Test | exec uspNewTicketNewCar  @DateStarted = '2021-05-05',  @DateComplete = '2021-05-06',  @TicketType = 'S',  @Comment = 'Service',  @EmployeeID = 100,  @CustomerID = 9,  @SerialNum = '1FADP3K22k9d',  @Make = 'Test',  @Model = 'Test',  @CarYear = '2021',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.0 | Expect one new ticket and one new car to be added according the parameter specifications  Will display  The ticket has successfully been added | The service ticket and new car was added according to the specified parameters in all tables.  Message:  The ticket has successfully been added  1 row added  AS EXPECTED |
| Success Test | exec uspNewTicketNewCar  @TicketType = 'S',  @EmployeeID = 100,  @SerialNum = '1FADP3K21DMA',  @Make = 'Test',  @Model = 'Test',  @CarYear = '2021',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.0 | Expect one new ticket and one new car to be added according to the parameters. Default and null values will all be added correctly.  Will display  The ticket has successfully been added | The service ticket and new car was added according to the specified parameters in all tables. Default and null values all added correctly.  Message:  The ticket has successfully been added  AS EXPECTED |
| Negative Test | exec uspNewTicketNewCar  @TicketType = 'r',  @EmployeeID = 101,  @SerialNum = '1FADP3K21GH2',  @Make = 'Test',  @Model = 'Test',  @CarYear = '2021',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.0 | Expect the message to show ‘only employees in the mechanical department may open a ticket’, will not proceed to insert rows | Message: only employees in the mechanical department may open a ticket  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketNewCar  @TicketType = 'r',  @EmployeeID = 100,  @SerialNum = '1FM5K8F87EGA',  @Make = 'Test',  @Model = 'Test',  @CarYear = '2021',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.0 | Expect the message to show ‘The Serial Number is already in the system. Please check again or use uspNewTicketExistingCar to add the car and create a ticket  ’, will not proceed to insert rows | Message:  The Serial Number is already in the system. Please check again or use uspNewTicketExistingCar to add the car and create a ticket  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketNewCar  @TicketType = 'r',  @EmployeeID = 100,  @SerialNum = '1FADP3K21jL',  @Make = 'Test',  @Model = 'Test',  @CarYear = '2021',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.0 | Expect the message to show ‘The serial number must be 12 characters’, will not proceed to insert rows | Message:  The serial number must be 12 characters  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketNewCar  @TicketType = 'r',  @EmployeeID = 100,  @SerialNum = '1FADP3K21DH2',  @CustomerID = 500,  @Make = 'Test',  @Model = 'Test',  @CarYear = '2021',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.0 | Expect the message to show ‘The customer is not in the system’, will not proceed to insert rows | Message:  The customer is not in the system  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketNewCar  @TicketType = 'r',  @EmployeeID = 100,  @SerialNum = null,  @Make = 'Test',  @Model = 'Test',  @CarYear = '2021',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.0 | Expect the message to show ‘Cannot add car, no Serial Number given’, will not proceed to insert rows | Message:  Cannot add car, no Serial Number given  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketNewCar  @TicketType = 'y',  @EmployeeID = 100,  @SerialNum = '1FADP3K21DQ2',  @Make = 'Test',  @Model = 'Test',  @CarYear = '2021',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.0 | Expect the message to show ‘The ticket type must be "s" for service or "r" for repair’, will not proceed to insert rows | Message:  The ticket type must be "s" for service or "r" for repair  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewTicketNewCar  @TicketType = 'y',  @EmployeeID = 100,  @SerialNum = '1FADP3K21DQ2',  @Make = 'Test',  @Model = 'Test',  @CarYear = '2021',  @Transmission = 'Manual',  @FuelType = 'Petrol',  @EngineSize = 2.0 | Expect the message to show ‘The engine size must be greater than 0  ’, will not proceed to insert rows | Message:  The engine size must be greater than 0  No rows added.  AS EXPECTED. |

**usp\_NewCustomer**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Type | Test Run | Expected Result | Actual Result |
| Success Test | exec uspNewCustomer  @FirstName='Tilly',  @SurName='Tester',  @Active = ‘Y’,  @PhoneNumber= '0987352436',  @SecondPhoneNo= '0764534278',  @Email='TillyTeser@email.com',  @Address='Test House',  @AddressLine2 ='Testing Ave',  @AddressLine3 ='Apt 1',  @city = 'Dublin',  @county= 'Dublin',  @PostCode ='A12B3C4' | Expect all of the fields relevant to the customer (in tables: Customer, Address, Contact) to be updated to insert the new Customer.  Will display ‘You have successfully added the customer’ | The customer was added according to the specified parameters in all tables.  Message: ‘You have successfully added the customer’  AS EXPECTED |
| Success Test | exec uspNewCustomer  @FirstName='Tamar',  @SurName='Tester',  @PhoneNumber= '0987352436',  @Email='TTeser@email.com',  @Address='Test House',  @city = 'Dublin',  @county= 'Dublin',  @PostCode ='A12B3C4' | Expect all of the fields relevant to the customer (in tables: Customer, Address, Contact) to be updated to insert the new Customer. Will insert all appropriate default and/or values.  Will display ‘You have successfully added the customer’ | The customer was added according to the specified parameters in all tables. Appropriate null and default values added.  Message: ‘You have successfully added the customer’  AS EXPECTED |
| Negative Test | exec uspNewCustomer  @FirstName='Tamar',  @SurName='Tester',  @PhoneNumber= '0987352436',  @Email='TTeseremail.com',  @Address='Test House',  @city = 'Dublin',  @county= 'Dublin',  @PostCode ='A12B3C4' | Expect the message to show ‘Invalid Email address  ’, will not proceed to insert rows | Message:  Invalid Email address  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewCustomer  @FirstName='Tamar',  @SurName='Tester',  @PhoneNumber= '0987352436',  @Email='TTeseremail.com',  @Address='Test House',  @city = 'Dublin',  @county= 'Dublin',  @PostCode ='A12B3C4' | Expect the message to show ‘Invalid Email address’, will not proceed to insert rows | Message:  Invalid Email address  No rows added.  AS EXPECTED. |
| Negative Test | exec uspNewCustomer  @FirstName='Tamar',  @SurName='Tester',  @PhoneNumber= '0987352436',  @Email='TTeser@email.com',  @Address='Test House',  @city = 'Dublin',  @county= 'Dublin',  @PostCode ='A12B3C' | Expect the message to show ‘Please enter a valid postcode’, will not proceed to insert rows | Message:  Please enter a valid postcode  No rows added.  AS EXPECTED. |

**usp\_UpdateActiveCustomers**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Type | Test Run | Expected Result | Actual Result |
| Success Test | exec uspUpdateActiveCustomers | Expect after using the inserts statements that 2 rows will visually change to reflect customers who have not had a service ticket or invoice within the last 13 months. Will report more rows changed in message as stored proc rewrites each row to ensure accuracy. | Updated rows as expected. |

# 

# **Reflections on Learning**

After first receiving the project brief, I was excited to have an opportunity to put my theoretical knowledge to practice. I realized I needed to scale back my original ideas as I began to finalize my ERD. My original ideas for the project overreached my technical abilities. After reassessing the scope of my project, and removing several relationships and entities, I was finally satisfied with the ERD. As I started to work on the SQL for the database, I realized that while I had good ideas and knew what I wanted to do, I was not always sure how to complete the task using SQL. As I began to write code, research, and rewrite code, I noticed my skills starting to develop. I was able to incorporate more logic, and what I originally though were difficult concepts into my SQL. After completing this project, I feel like I have a much deeper understanding of the entire course. I can see how the concepts relate and build upon each other, and I am happy with what I have achieved.

# **References**

Reference material used for this project includes:

* Lecture Material on Moodle
* StackOverflow.com
* YouTube – Caleb Curry on Normalization
* Docs.Microsoft.com

## **9. SQL**

Please see attached SQL files. They are numbered according to the order in which to be run. The test run SQL statements are also included in a commented section at the bottom of each file. Running these statements (the successful ones) will allow a greater amount of data to be seen in the Views at the end. For example running the test queries to delete an employee will demonstrate the way that the view handles deleted employees.