

Project Report

Bus reservation system

## Module Title: Advanced databases

## Module Code: B9IS100

## GROUP MEMBERS:

## SINGUPURAM NAVIN KUMAR | 10612366

## DIGVIJAY JATKAR | 10508676

Tuba Tosunoglu| 10625974

## Module Leader: Dr Shazia A Afzal

Table of Contents

[1. ABSTRACT 2](#_Toc123129248)

[2. SCOPE OF THE DATABASE 2](#_Toc123129249)

[3. BUSINESS RULES 9](#_Toc123129250)

[4. RELATIONAL SCHEMA FOR HYBRID IN 3NF 12](#_Toc123129251)

[4.1. XML HYBRID DATABASE SCHEMA 17](#_Toc123129252)

[5. DATA DIAGRAM 19](#_Toc123129253)

[6. ENTITY RELATIONSHIP DIAGRAM 20](#_Toc123129254)

[7. IMPLEMENTATION IN SQL SERVER 21](#_Toc123129255)

[7.1. STORED PROCEDURES 21](#_Toc123129256)

[7.2. TRIGGERS 32](#_Toc123129257)

[7.3. VIEWS 34](#_Toc123129258)

[8. Conclusion 35](#_Toc123129259)

[9. Bibiliography 36](#_Toc123129260)

[10. Appendix A: Create Table Queries 36](#_Toc123129261)

[11. Appendix B: Insert Table Queries](#_Toc123129262) 40

[12. Innovation 42](#_Toc123129263)

# TABLE OF FIGURES

[Figure 1. XML DIAGRAM 17](#_Toc123131572)

[Figure 2 : Data Diagram 19](#_Toc123131573)

[Figure 3 : Entity Relationship Diagram 20](file:///D:\DBS\Classes\Advanced%20Database\Assignment\B9IS100_Advanced%20databases.docx#_Toc123131574)

[Figure 4: Stored Procedure 1 22](#_Toc123131575)

[Figure 5: Stored Procedure 2 23](#_Toc123131576)

[Figure 6 : Stored Procedure 3 25](#_Toc123131577)

[Figure 7 : Stored Procedure 4 26](#_Toc123131578)

[Figure 8: Stored Procedure 5 28](#_Toc123131579)

[Figure 9: Stored Procedure 6 29](#_Toc123131580)

[Figure 10: Stored Procedure 7 31](#_Toc123131581)

[Figure 11 : Trigger 1 32](#_Toc123131582)

[Figure 12: Trigger 2 33](#_Toc123131583)

[Figure 13: View 1 34](#_Toc123131584)

[Figure 14: View 2 35](#_Toc123131585)

# ABSTRACT

Bus Reservation system -

Bus Reservation System focuses on providing consumers with bus rentals. Customers can view available buses, register, view profiles, and book a bus using this online system. Management of both the people and the vehicles is involved in this. The system unifies all tasks for managing the bus rental company's and its workers' workloads. All the data required for their management is already in the system. After making reservations, it can schedule your employees' full day.

# SCOPE OF THE DATABASE

To accomplish the project's goals, research must be done in a variety of disciplines, from commercial principles to the realm of computing. The region includes

Reservations for buses: This involves an investigation into the operation of the bus reservation industry, the steps involved, and potential areas for improvement. the method via which the application was created technologically Both regular clients and employees of the organization will be able to use the system efficiently. Because the system is a web platform, it will always be accessible, barring any predicted infrequent transitory server issues.

1. **Bus**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Description |
| BusId | INT(10) | Id for uniquely identifier |
| BusName | VARCHAR(25) | Name designated to identify bus |
| BusEngineNumber | VARCHAR(25) | Engine number of a specific bus |
| BusChassisNumber | VARCHAR(25) | Chassis number of a bus designated |
| BusModelNumber | VARCHAR(25) | Model number of a bus |
| BusPrice | INT(10) | Rental price of a bus |
| BusFuelTypes | VARCHAR(10) | (Gasoline/Petrol/Diesel) type of fuel |
| BusMileage | INT(10) | Average(per Km bus) |
| BusPolicy | VARCHAR(25) | Policies attained |
| BusSeatingCapacity | INT(5) | Seating capacity for the persons |
| BusRegistrationDate | VARCHAR(20) | Registrated date |
| BusRating | INT(5) | User rating |
| CreatedAt | TIMESTAMP | The date on which the bus details  added to DB |

Primary Key: BusId

Required Attributes: BusId, BusName, BusEngineNumber, BusChassisNumber, BusModelNumber, BusPrice, BusFuelTypes, BusSeatingCapacity, BusPolicy BusRegistrationDate, BusRating, CreatedAt

Optional Attributes: BusMileage, BuFuelTankCC, BusRating

1. **Customer**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data type | Description |
| CustomerId | INT(10) | Identify customer uniquely |
| CustomerFirstName | VARCHAR(50) | First name of the customer |
| CustomerLastName | VARCHAR(50) | Last name of the customer |
| CustomerAddress | TEXT | Address of the customer |
| CustomerMobileNumber | VARCHAR(20) | Mobile number of the customer |
| CustomerRole | VARCHAR(10) | Role |
| CustomerDob | VARCHAR(15) | date of birth |
| CustomerEmail | VARCHAR(25) | email |
| CustomerPassword | VARCHAR(20) | password |
| CustomerLicense | VARCHAR(20) | license number |
| CreatedAt | TIMESTAMP | The date registered |

Primary Key: CustomerId

Required Attributes: CustomerFirstName, , CustomerAddress, CustomerMobileNumber, CustomerDob, CustomerEmail , CustomerPassword, CustomerLicense, CreatedAt

Optional Attributes: CustomerLastName

**3. Bus\_type**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Description |
| BusTypeId | INT(10) | Identifies every bus |
| BusId | INT(10) | Bus Id foreign key  Bus Table |
| BusPowerSteering | Varchar(20) | Type of power steering |
| BusAirConditioner | Varchar(20) | Type of AC |
| BusPassengerAirbag | Varchar(20) | Type of Airbag passenger |
| BusDriverAirbag | Varchar(20) | Type of Airbag driver |
| BusSunRoof | Varchar(20) | Type of SunRoof |
| BusDriverSeat | Varchar(20) | Type of driver seat |
| BusFogLights | Varchar(20) | Type of bus fog lights |
| BusType | VARCHAR(20) | Type of bus |
| CreatedAt | TIMESTAMP | Date Registered |

Primary Key: BusTypeId

Foreign Key: BusId

Attributes:BusPowerSteering, BusAirConditioner, BusPassengerAirbag, BusDriverAirbag, BusDriverSeat, BusType, BusType

Optional Attributes: BusSunRoof, BusFogLights

**4. Billing**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data type | Description |
| BillingId | INT(10) | Unique Identifier |
| ReservationId | INT(10) | Id for reservation details |
| BillingMode | VARCHAR(20) | Tyes of Billing Mode |
| BusTransactioAmount | INT(20) | Total amount of booked bus |

|  |  |  |
| --- | --- | --- |
| AddPromocode | VARCHAR(20) | Promo code for special  discount |
| AddTaxCharges | VARCHAR(10) | Addition tax charges |
| BusTransactionStatus | VARCHAR(10) | Transaction status |
| BusBillingCurrency | VARCHAR(20) | Currency of billing amt |
| CreatedAt | TIMESTAMP | Creatd Date of Billing |
| UpdatedAt | TIMESTAMP | Updatd Date of Billing |

Primary Key: - BillingId

Foreign Key: - ReservationId

Required Attributes: - BillingMode, BusTransactioAmount, BusBillingCurrency, AddTaxCharges

Optional Attributes: - AddPromocode, CreatedAt, UpdatedAt

**5. Bus Rental Type**

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Description |
| BusRentalTypId | INT(10) | uniquely identify |
| BusRentalTripSelection | VARCHAR(20) | Roundtrip or Oneway Booking |
| BusRentalSeaters | INT(10) | Seaters of total sittings |
| BusRentalFuelType | VARCHAR(10) | Fuel Type describes |
| BusRentalType | VARCHAR(20) | Type defines buses |
| BusRentalAutoPilot | INT | Auto Pilot option |
| BusRentalStatus | VARCHAR(10) | Check the particular bus  available |
| BusRentalStartDt | VARCHAR(20) | Start date |
| BusRentalEndDt | VARCHAR(20) | End date |

Primary Key : BusRentalTypId

**6. Reservation**

Primary Key: - ReservationId

Foreign Key: - BusId, CustomerId, BusRentalTypeId

Attributes: - ReservationStartLocation, ReservationEndLocation, ReservationDuration, ReservationTripAmount, BusKmStart, BusKmEnd, BusFuelStart, BusFuelEnd, ReservationStatus, CreatedAt, UpdatedAt

|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Description |
| ReservationId | INT(10) | Each reservation has unique Id |
| BusId | INT(10) | Unique identifier for a Bus |
| CustomerId | INT(10) | Customer Id for customer details |
| BusRentalTypeId | INT(10) | Identity for rental type |
| ReservationStartLocation | VARCHAR(25) | Start Location |
| ReservationEndLocation | VARCHAR(25) | end Location |
| ReservationDuration | INT(5) | duration |
| ReservationTripAmount | INT(20) | Total Amount chargeable |
| BusKmStart | INT(20) | Bus’s Kilometres at the start |
| BusKmEnd | INT(20) | Bus’s Kilometres at the end |
| BusFuelStart | INT(10) | Bus’s Fuel Percentage at the  start |
| BusFuelEnd | INT(10) | Bus’s Fuel Percentage at the  end |
| ReservationStatus | VARCHAR(20) | Reservation’s status |
| CreatedAt | TIMESTAMP | creation date |
| UpdatedAt | TIMESTAMP | updating date |

**7. Reimbursement**

Primary Key: - ReimbursementId Foreign Key: - ReservationId

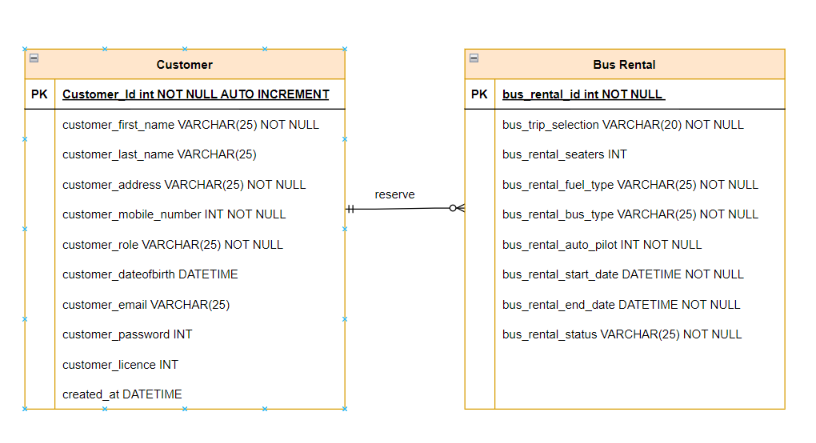
RequiredAttributes: DeductionReimbursedamount, ReimbursementStatus,

ReimbursementAmount, ReimbursementAccount, CreatedAt

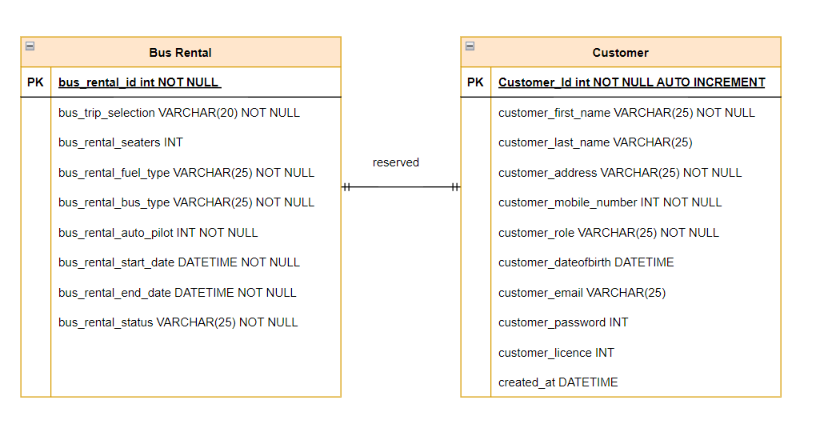
|  |  |  |
| --- | --- | --- |
| Attribute Name | Data Type | Description |
| ReimbursementId | INT(10) | Identify every unique reimbursement |
| ReservationId | INT(10) | Referred to  the reservation |
| DeductionReimbursedamou  nt | INT(10) | Reimbursed transaction amount |
| ReimbursementStatus | VARCHAR(20) | Status |
| ReimbursementAmount | INT(20) | Reimbursement amount |
| ReimbursementAccount | VARCHAR(20) | Amount of a reimbursement |
| CreatedAt | TIMESTAMP | Initiation date |

# 3. BUSINESS RULES

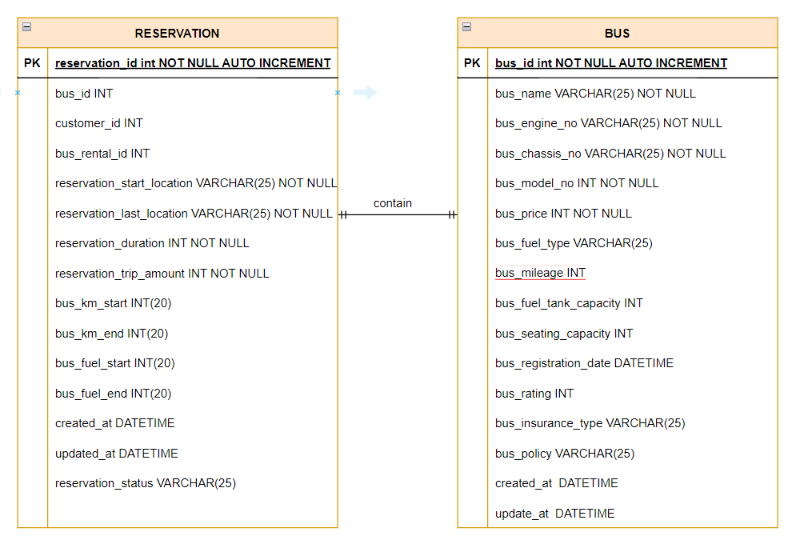
1. One to many dependencies from customer to bus rentals



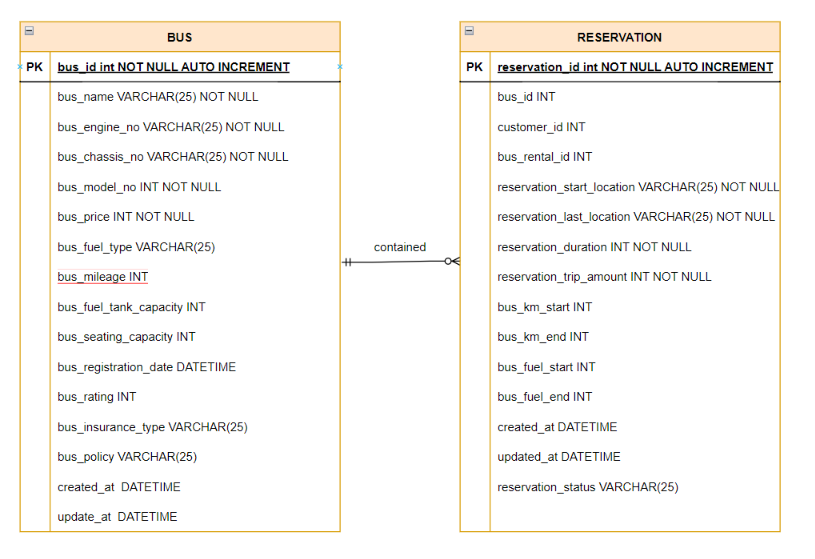
2. One to one mapping from bus rental to customer



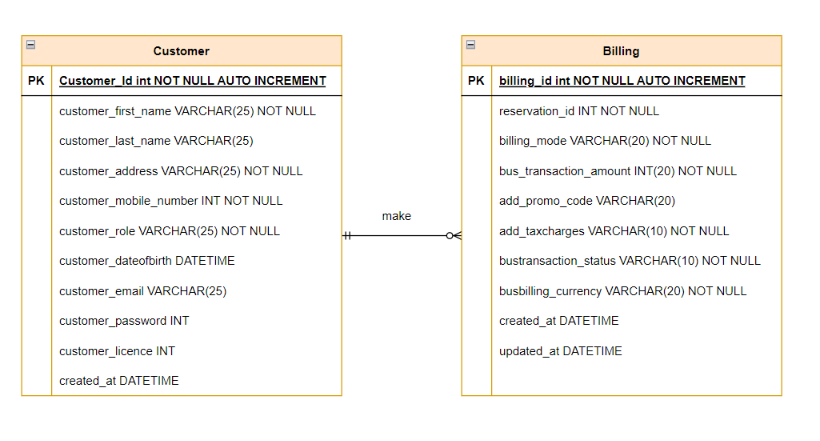
3. One reservation can contain one bus



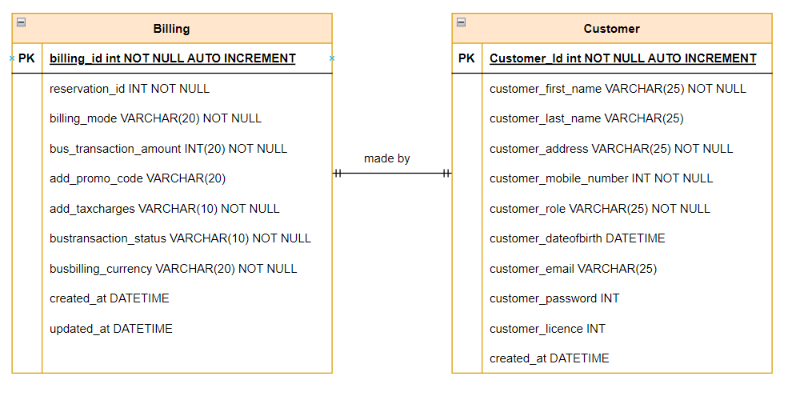
4 . One bus can be contained in many reservations



5. One customer can make many billing



6. One billing can be made by one customer



# 4. RELATIONAL SCHEMA FOR HYBRID IN 3NF

Data in a database can be organized using the normalization process. Reduced redundancy from a relation or collection of relations is achieved through normalization. Additionally, it aids in overcoming unfavorable traits like Insertion, Update, and Deletion Anomalies. The bigger table is split into smaller tables, and they are connected through relationships.

Below are the normalization forms used:

1. First Normalization Form (1NF) of an entity describes if only
   1. It holds multiple kind of values or data.
   2. Most of the values stored shold be atomic
2. Second Normalization Form (1NF) of an entity describes if only
   1. It should be in the first normalization form.
   2. Every non-key attribute needs to be reliant on the main key. (No partial reliance)
3. Third Normalization Form (1NF) of an entity describes if only
   1. It should be in the second normalization form.
   2. There should be no transitive dependencies in it.

**1. Bus Table**

* + - 1. Primary Key is defined (BusId).
      2. There is an atomic value for each column
      3. Groups should not repeatable.

Hence, the bus table is in the First Normalisation Form.

1. The table is in First Normalisation Form
2. The primary key controls every non-key attribute.

Hence, the table is in the Second Normalisation Form.

1. The table is in the Second Normalisation Form.
2. There are no transitive dependencies present.

Hence The table is in the Third Normalisation Form.

* + 1. **Customer Table**
       1. Primary Key is defined(CustomerId).
       2. There is an atomic value for each column.
       3. Groups should not repeat

Hence, The table is in the First Normalisation Form.

1. The table is in the First Normalisation Form
2. The primary key controls every non-key attribute.

Hence, the table is in the Second Normalisation Form.

1. The table is in the Second Normalisation Form.

b. There are no transitive dependencies present.

Hence The table is in the Third Normalisation Form.

* + 1. **Bus Type Table**
       1. Primary Key is defined(BusTypeId).
       2. There is an atomic value for each column.
       3. Groups should not repeat

Hence, The table is in the First Normalisation Form.

1. The table is in the First Normalisation Form
2. The primary key controls every non-key attribute.

Hence, the table is in the Second Normalisation Form.

1. The table is in the Second Normalisation Form.

b. There are no transitive dependencies present.

Hence The table is in the Third Normalisation Form.

* + 1. **Billing Table**
       1. Primary Key is defined(BillingId).
       2. There is an atomic value for each column.
       3. Groups should not repeat

Hence, The table is in the First Normalisation Form.

1. The table is in the First Normalisation Form
2. The primary key controls every non-key attribute.

Hence, the table is in the Second Normalisation Form.

1. The table is in the Second Normalisation Form.

b. There are no transitive dependencies present.

Hence The table is in the Third Normalisation Form.

* + 1. **Bus Rental Table**
       1. Primary Key is defined(BusRentalTypeId).
       2. There is an atomic value for each column.
       3. Groups should not repeat

Hence, The table is in the First Normalisation Form.

1. The table is in the First Normalisation Form
2. The primary key controls every non-key attribute.

Hence, the table is in the Second Normalisation Form.

1. The table is in the Second Normalisation Form.

b. There are no transitive dependencies present.

Hence The table is in the Third Normalisation Form.

* + 1. **Bus Reservation Table**
       1. Primary Key is defined(ReservationId).
       2. There is an atomic value for each column.
       3. Groups should not repeat

Hence, The table is in the First Normalisation Form.

1. The table is in the First Normalisation Form
2. The primary key controls every non-key attribute.

Hence, the table is in the Second Normalisation Form.

1. The table is in the Second Normalisation Form.

b. There are no transitive dependencies present.

Hence The table is in the Third Normalisation Form.

* + 1. **Reimbursement Table**
       1. Primary Key is defined(ReimbursementId).
       2. There is an atomic value for each column.
       3. Groups should not repeat

Hence, The table is in the First Normalisation Form.

a.The table is in the First Normalisation Form

b.The primary key controls every non-key attribute.

Hence, the table is in the Second Normalisation Form.

c. The table is in the Second Normalisation Form.

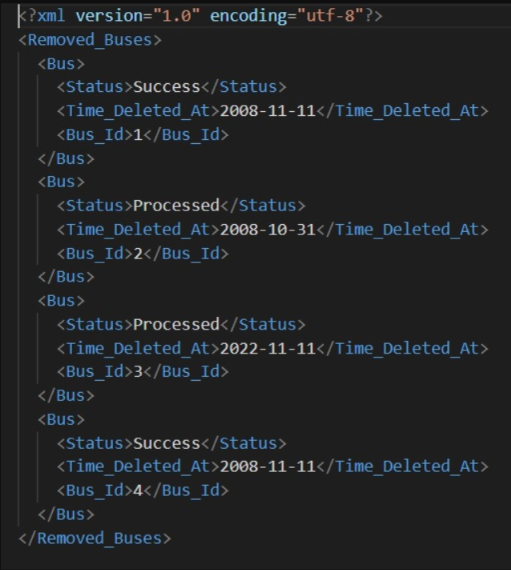
d. There are no transitive dependencies present.

Hence The table is in the Third Normalisation Form.

# 4.1. XML HYBRID DATABASE SCHEMA

In this instance, the outside system is considered. Refunds are processed, as well as a Xml data form is returned in response.

We employed insertion procedures that read the xml document below from a certain region, read it, and then entered the data into a table in order to establish a hybrid database.



**Figure 1. XML DIAGRAM**

In addition to rules governing data content and semantics, such as what fields an element can include, which sub elements it can contain, and how many items can be present, XML schema determines the shape, or structure, of an XML document. Additionally, it can specify the kind and range of values that can be assigned to each element or property. Facets are XML data restrictions that provide guidelines like minimum and maximum lengths.

**INSERT INTO Removed\_Buses (Status, Time\_Deleted\_At, Bus\_Id)**

**SELECT**

**MY\_XML.Bus.query('Status').value('.', 'VARCHAR(20)'),**

**MY\_XML.Bus.query('Time\_Deleted\_At').value('.', 'datetime'),**

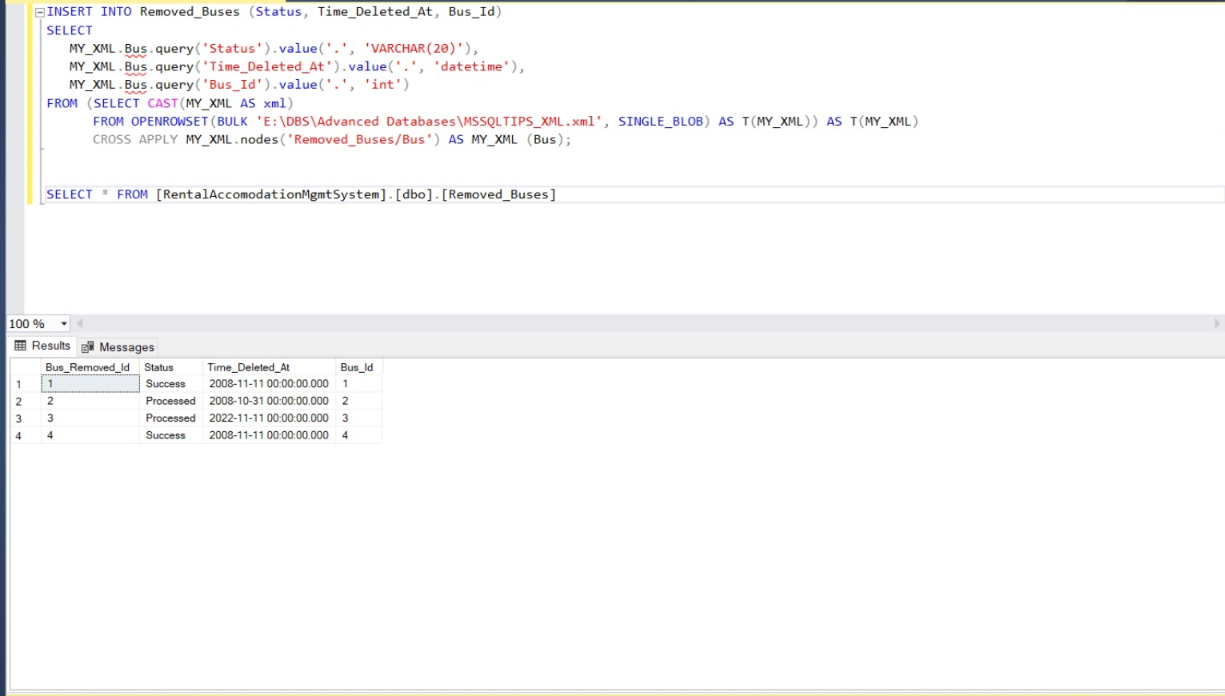
**MY\_XML.Bus.query('Bus\_Id').value('.', 'int')**

**FROM (SELECT CAST(MY\_XML AS xml)**

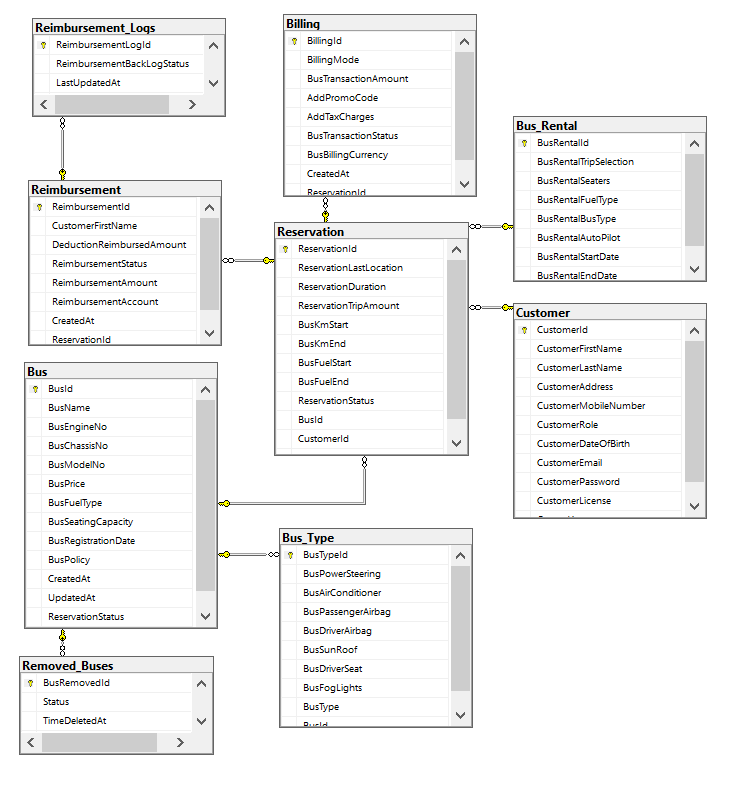
**FROM OPENROWSET(BULK 'E:\DBS\Advanced Databases\MSSQLTIPS\_XML.xml', SINGLE\_BLOB) AS T(MY\_XML)) AS T(MY\_XML)**

**CROSS APPLY MY\_XML.nodes('Removed\_Buses/Bus') AS MY\_XML (Bus);**

**SELECT \* FROM [RentalAccomodationMgmtSystem].[dbo].[Removed\_Buses]**

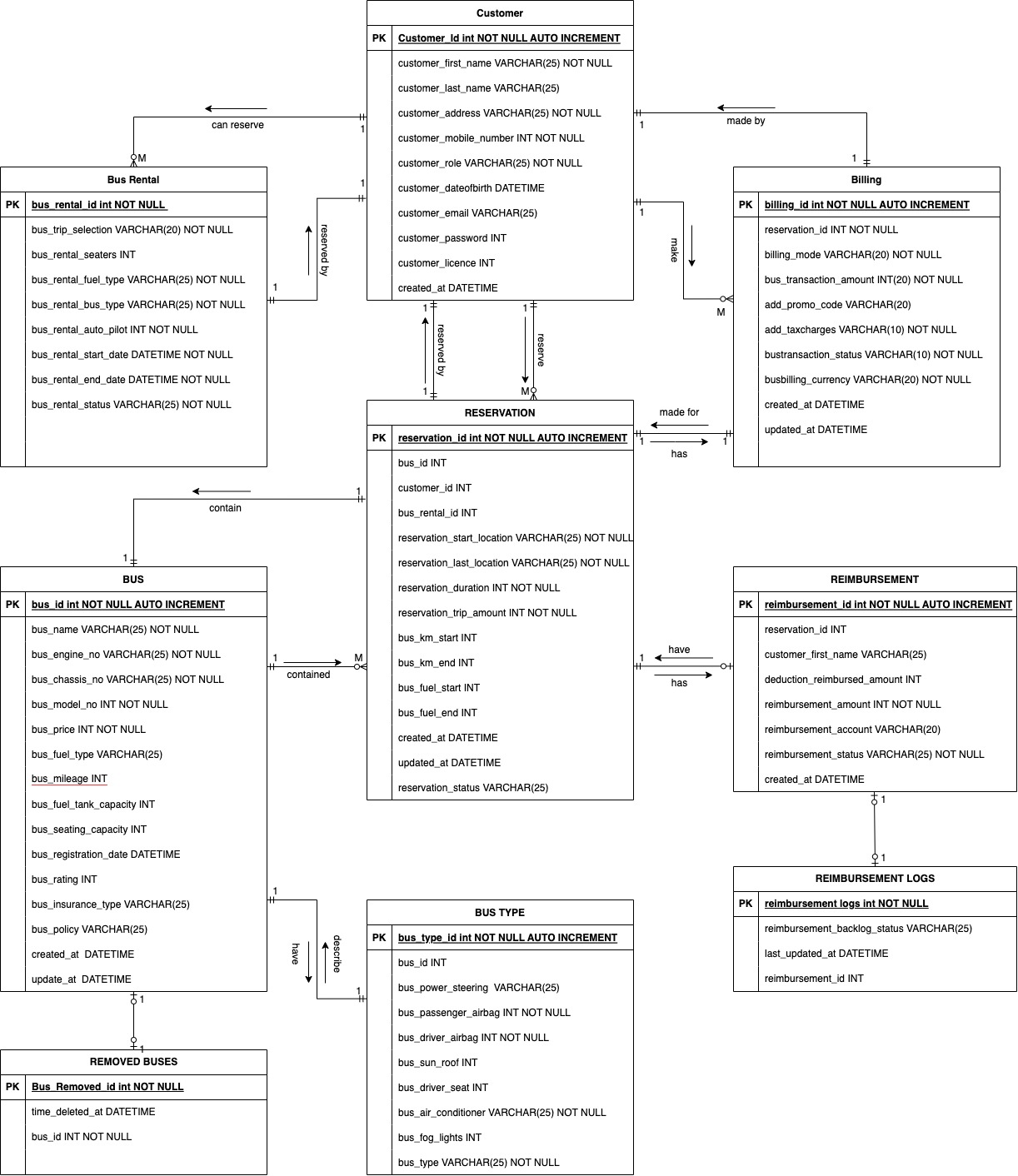


# 5. DATA DIAGRAM



**Figure 2 : Data Diagram**

# 6. ENTITY RELATIONSHIP DIAGRAM



**Figure 3 : Entity Relationship Diagram**

# 7. IMPLEMENTATION IN SQL SERVER

In the system design we have added the following things where we the stored procedures and triggers, views are used to have different database functionalities.

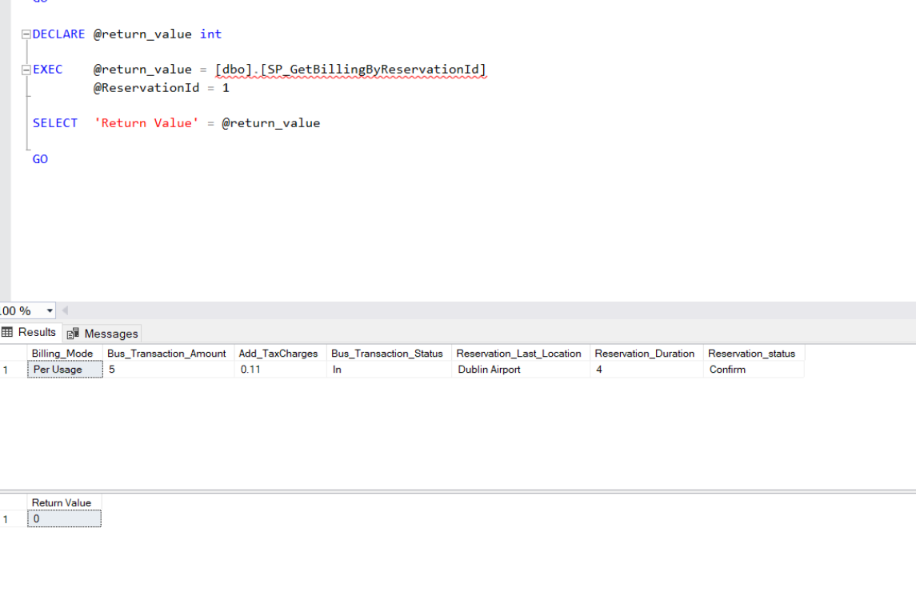
# 7.1. STORED PROCEDURES

1. Design a process that allows users to look for buses that meet their criteria for ratings.

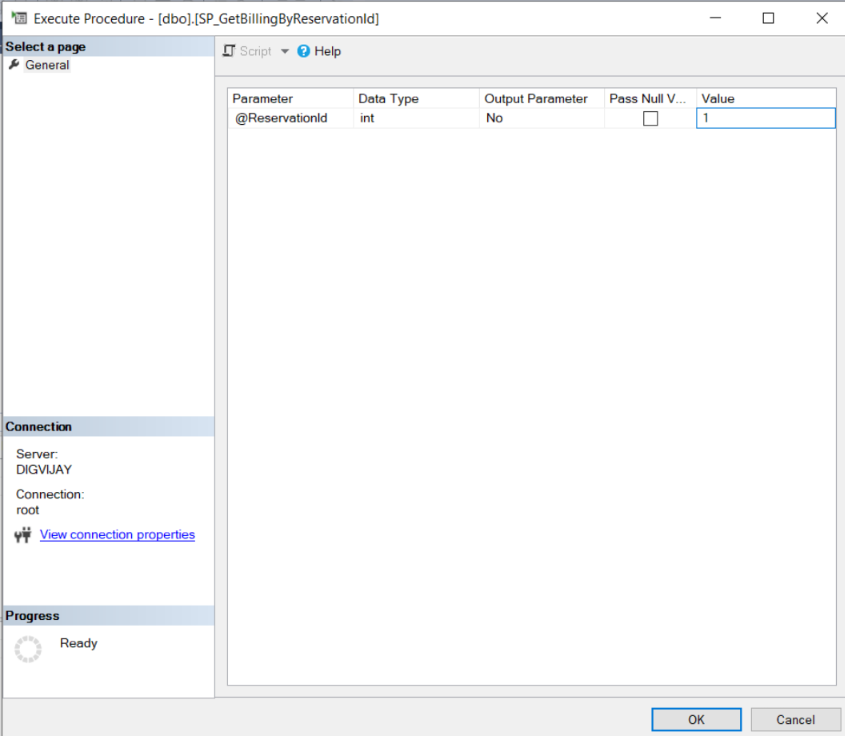
Here, the user's input is taken, and buses that meet or exceed that rating are displayed to the user:

**Stored Procedure - 1**

|  |  |
| --- | --- |
| **CREATE PROCEDURE [dbo].[SP\_GetBillingByReservationId]** | |
|  | **-- Add parameters sp** |
|  | **(@ReservationId INT)** |
|  | **AS** |
|  | **BEGIN** |
|  | **SET NOCOUNT ON;** |
|  |  |
|  | **-- Insert statements sp** |
|  | **SELECT Billing.Billing\_Mode, Billing.Bus\_Transaction\_Amount, Billing.Add\_TaxCharges, Billing.Bus\_Transaction\_Status,** |
|  | **Reservation.Reservation\_Last\_Location, Reservation.Reservation\_Duration, Reservation.Reservation\_status** |
|  | **from Billing** |
|  | **INNER JOIN Reservation** |
|  | **On Billing.Billing\_Id = Reservation.Reservation\_Id** |
|  | **where Reservation.Reservation\_Id = 1** |
|  | **order by Reservation.Reservation\_Duration** |
|  | **END** |

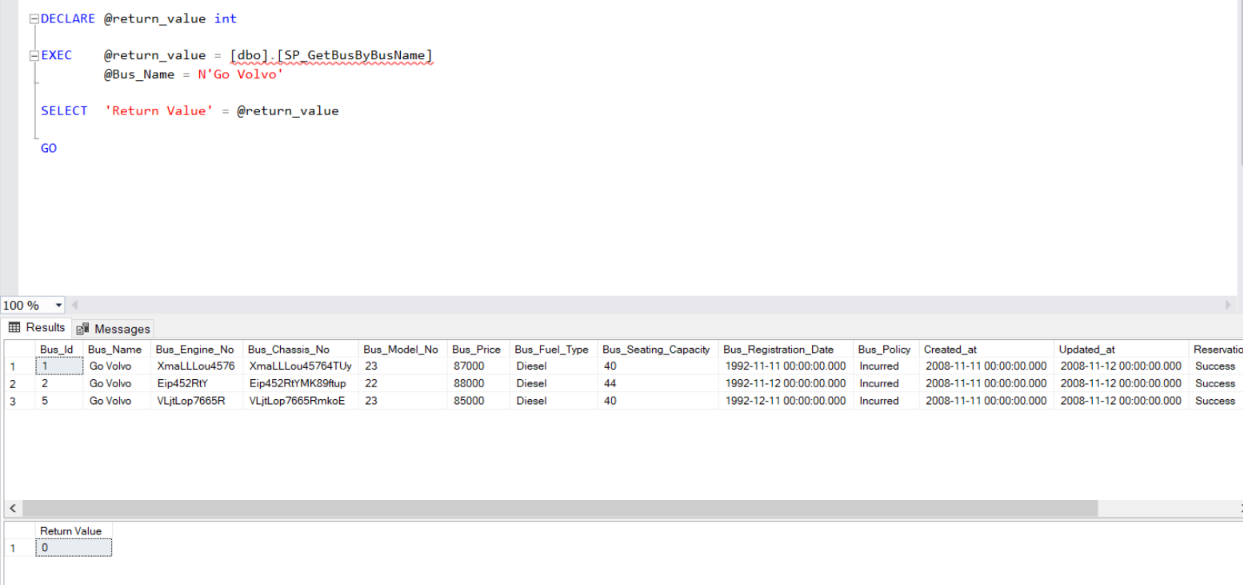


**Figure 4: Stored Procedure 1**

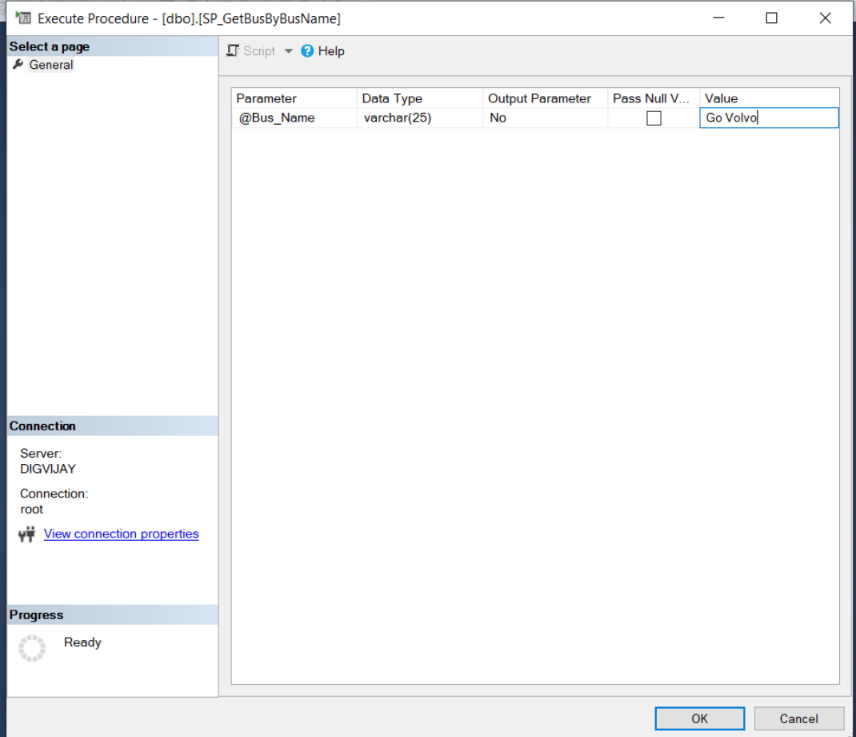


**Stored Procedure – 2**

|  |  |
| --- | --- |
| **CREATE PROCEDURE [dbo].[SP\_GetBusByBusName]** | |
|  | **-- Add parameters sp** |
|  | **(@Bus\_Name varchar(25))** |
|  |  |
|  | **AS** |
|  | **BEGIN** |
|  | **SET NOCOUNT ON;** |
|  |  |
|  | **-- Insert statements sp** |
|  | **SELECT \* from Bus** |
|  | **where Bus\_Name = @Bus\_Name** |
|  | **order by Bus\_Id asc** |
|  | **END** |

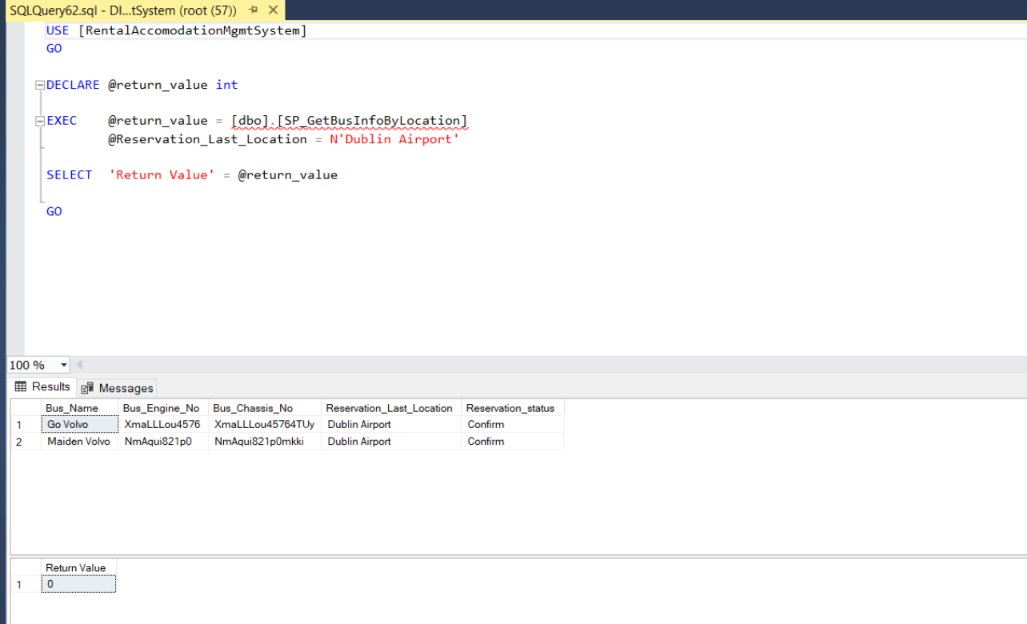


**Figure 5: Stored Procedure 2**



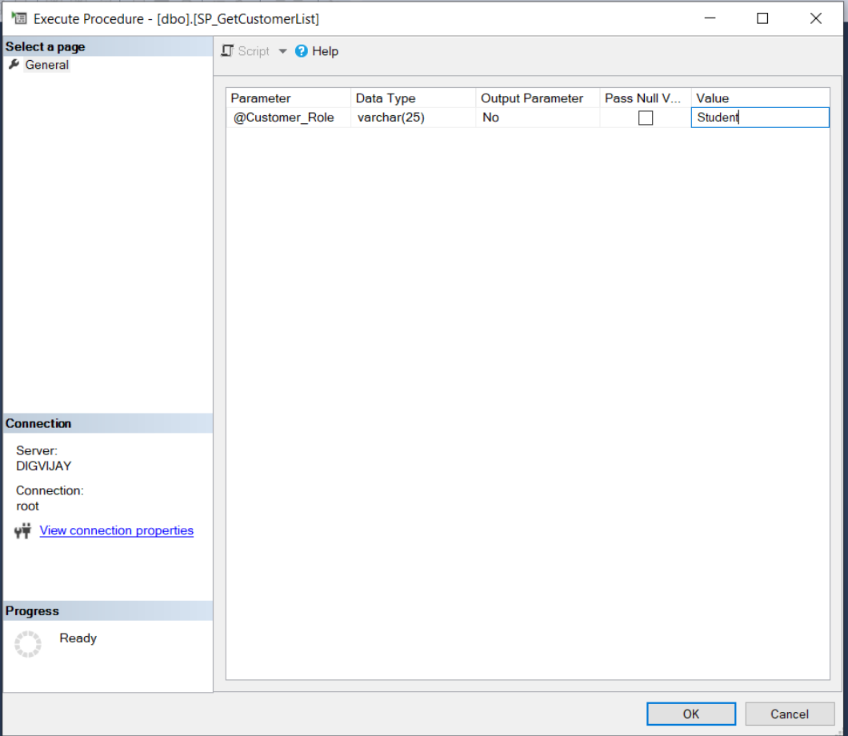
**Stored Procedure – 3**

|  |  |
| --- | --- |
| **CREATE PROCEDURE [dbo].[SP\_GetBusInfoByLocation]** | |
|  | **-- Add parameters sp** |
|  | **(@Reservation\_Last\_Location varchar(25))** |
|  | **AS** |
|  | **BEGIN** |
|  | **SET NOCOUNT ON;** |
|  |  |
|  | **-- Insert statements sp** |
|  | **SELECT Bus.Bus\_Name, Bus.Bus\_Engine\_No, Bus.Bus\_Chassis\_No, Reservation.Reservation\_Last\_Location, Reservation.Reservation\_status from Bus** |
|  | **INNER JOIN Reservation** |
|  | **on Bus.Bus\_Id = Reservation.Reservation\_Id** |
|  | **where Reservation\_Last\_Location like '%' + TRIM(@Reservation\_Last\_Location) + '%'** |
|  | **order by Bus.Bus\_Id** |
|  | **END**    **Figure 6 : Stored Procedure 3** |



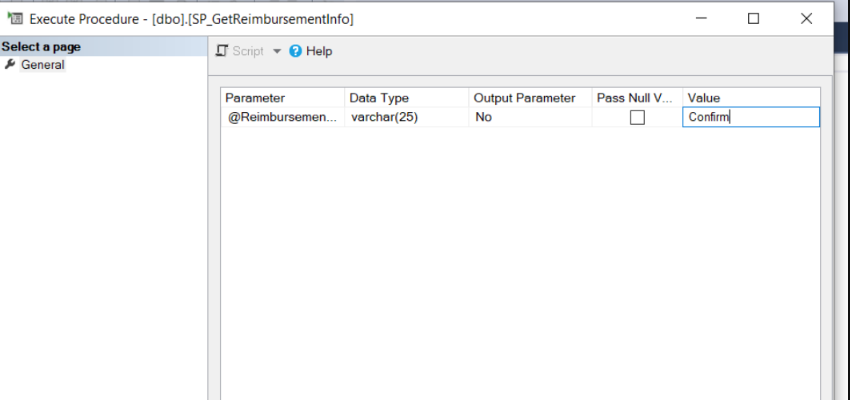
**Stored Procedure – 4**

|  |  |
| --- | --- |
| **CREATE PROCEDURE [dbo].[SP\_GetCustomerList]** | |
|  | **-- Add parameters sp** |
|  | **(@Customer\_Role varchar(25))** |
|  | **AS** |
|  | **BEGIN** |
|  | **SET NOCOUNT ON;** |
|  |  |
|  | **-- Insert statements sp** |
|  | **SELECT \* from Customer** |
|  | **WHERE Customer\_Role = @Customer\_Role** |
|  | **ORDER BY Customer.Customer\_First\_Name;** |
|  | **END**    **Figure 7 : Stored Procedure 4** |



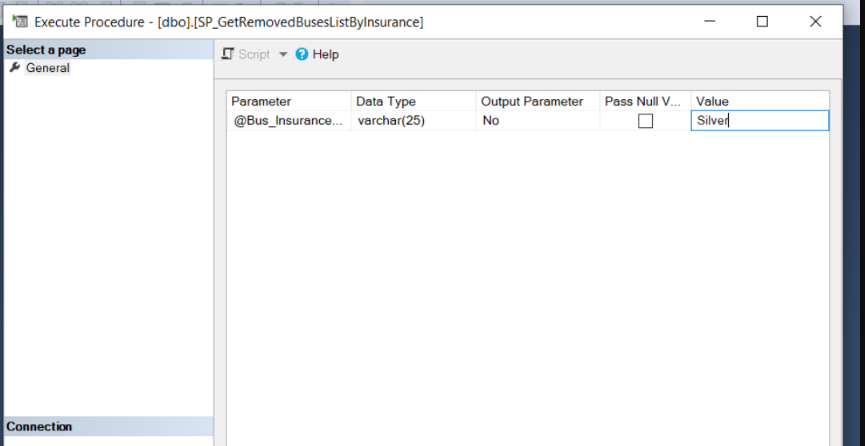
**Stored Procedure – 5**

|  |  |
| --- | --- |
| **CREATE PROCEDURE [dbo].[SP\_GetReimbursementInfo]** | |
|  | **-- Add parameters sp** |
|  | **(@Reimbursement\_Status varchar(25))** |
|  | **AS** |
|  | **BEGIN** |
|  | **SET NOCOUNT ON;** |
|  |  |
|  | **-- Insert statements sp** |
|  | **select Reimbursement.Customer\_First\_Name, Reimbursement.Reimbursement\_Amount,** |
|  | **Reimbursement\_Logs.Reimbursement\_BackLog\_Status** |
|  | **From Reimbursement** |
|  | **INNER JOIN** |
|  | **Reimbursement\_Logs on** |
|  | **Reimbursement\_Logs.Reimbursement\_Log\_Id = Reimbursement.Reimbursement\_Id** |
|  | **where Reimbursement\_Status = @Reimbursement\_Status;** |
|  | **END**    **Figure 8: Stored Procedure 5** |



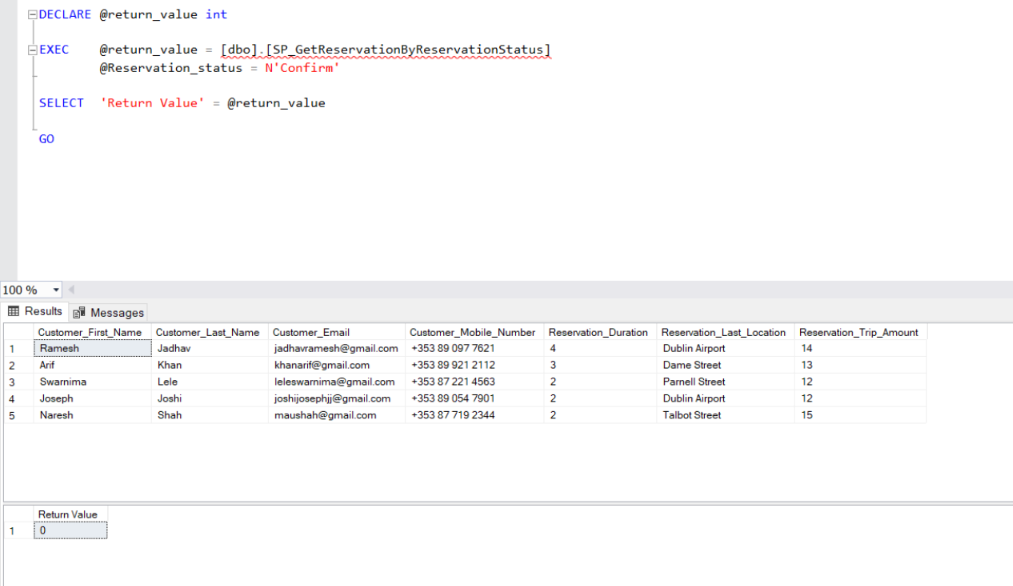
**Stored Procedure – 6**

|  |  |
| --- | --- |
| **CREATE PROCEDURE [dbo].[SP\_GetRemovedBusesListByInsurance]** | |
|  | **-- Add parameters sp** |
|  | **(@Bus\_Insurance\_Type varchar(25))** |
|  | **AS** |
|  | **BEGIN** |
|  | **SET NOCOUNT ON;** |
|  |  |
|  | **-- Insert statements sp** |
|  | **select Bus.Bus\_Name, Bus.Bus\_Engine\_No, Bus.Bus\_Chassis\_No, Bus.Bus\_Fuel\_Type, Removed\_Buses.Bus\_Removed\_Id, Removed\_Buses.Time\_Deleted\_At** |
|  | **From Bus** |
|  | **INNER JOIN** |
|  | **Removed\_Buses on** |
|  | **Bus.Bus\_Id = Removed\_Buses.Bus\_Removed\_Id** |
|  | **where Bus\_Insurance\_Type like '%' + trim(@Bus\_Insurance\_Type) + '%';** |
|  | **END**    **Figure 9: Stored Procedure 6** |

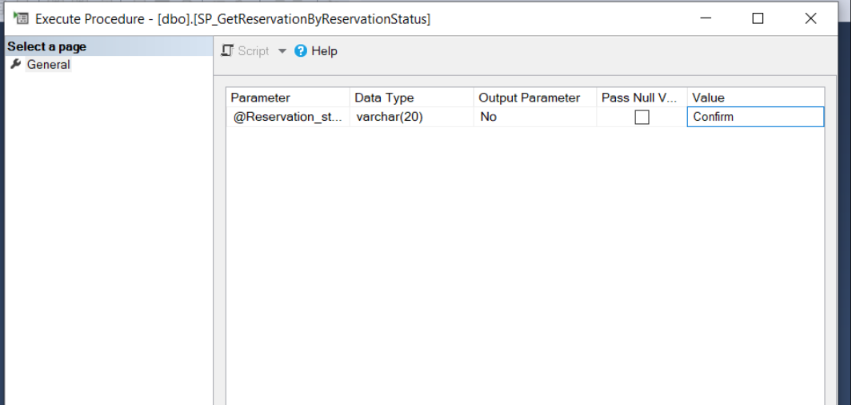


**Stored Procedure – 7**

|  |  |
| --- | --- |
| **CREATE PROCEDURE [dbo].[SP\_GetReservationByReservationStatus]** | |
|  | **-- Add parameters sp** |
|  | **(@Reservation\_status varchar(20))** |
|  | **AS** |
|  | **BEGIN** |
|  | **SET NOCOUNT ON;** |
|  |  |
|  | **-- Insert statements sp** |
|  | **SELECT Customer.Customer\_First\_Name, Customer.Customer\_Last\_Name, Customer.Customer\_Email, Customer.Customer\_Mobile\_Number,** |
|  | **Reservation.Reservation\_Duration, Reservation.Reservation\_Last\_Location, Reservation.Reservation\_Trip\_Amount** |
|  | **from Customer** |
|  | **INNER JOIN** |
|  | **Reservation ON Customer.Customer\_Id = Reservation.Reservation\_Id** |
|  | **WHERE Reservation\_status = @Reservation\_status** |
|  | **/\*'Confirm';\*/** |
|  | **END** |



**Figure 10: Stored Procedure 7**



# 7.2. TRIGGERS

1. To store the information of the customer, when they try to register, login into the system and reserve any of the vehicle and the below table works, when the customer is inserted into the table, the trigger insertcustomer data into the insert customer trigger.

**CREATE TRIGGER trInsertCustomer**

**ON Customer**

**FOR INSERT**

**AS**

**BEGIN**

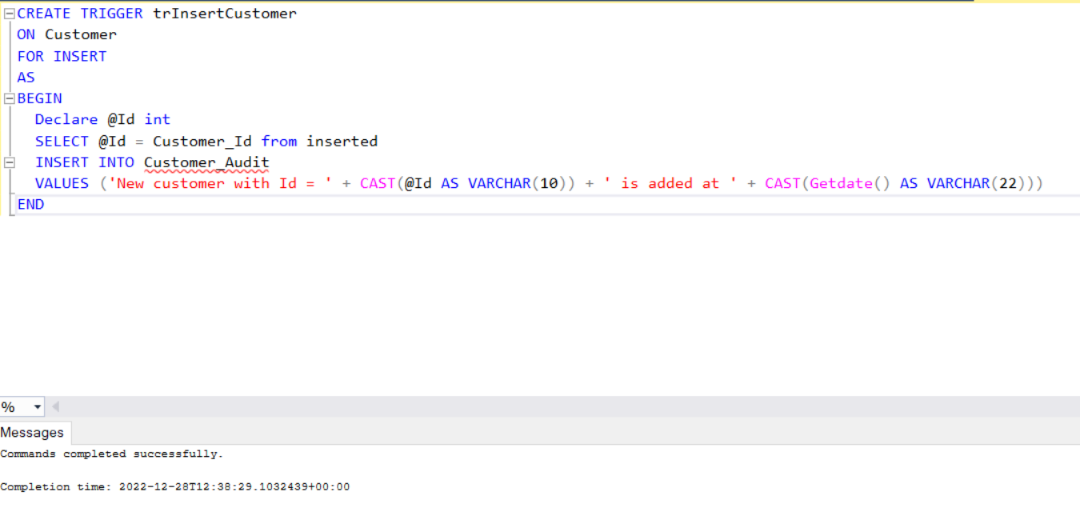
**Declare @Id int**

**SELECT @Id = Customer\_Id from inserted**

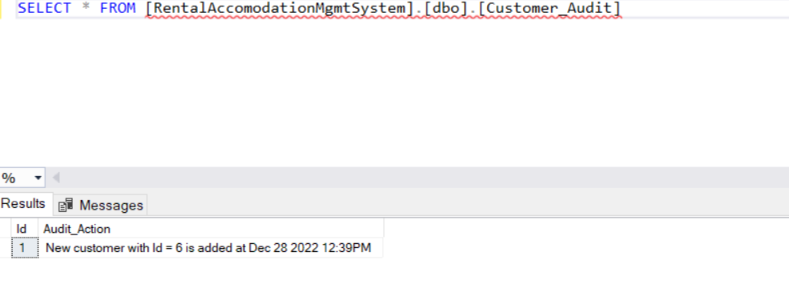
**INSERT INTO Customer\_Audit**

**VALUES ('New Customer with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is added at ' + CAST(Getdate() AS VARCHAR(22)))**

**END**



**Figure 11 : Trigger 1**



1. When a customer creates a reservation for a vehicle or a trip, insertreservation trigger is triggered and the data is inserted into the reservation\_audit table.

**CREATE TRIGGER trInsertReservation**

**ON Reservation**

**FOR INSERT**

**AS**

**BEGIN**

**Declare @Id int**

**SELECT @Id = Reservation\_Id from inserted**

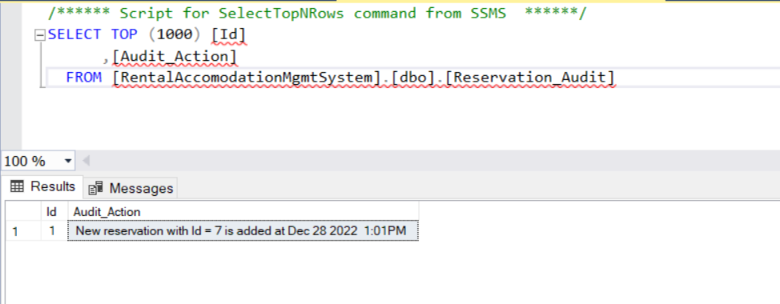
**INSERT INTO Reservation\_Audit**

**VALUES ('New reservation with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is added at ' + CAST(Getdate() AS VARCHAR(22)))**

**END**



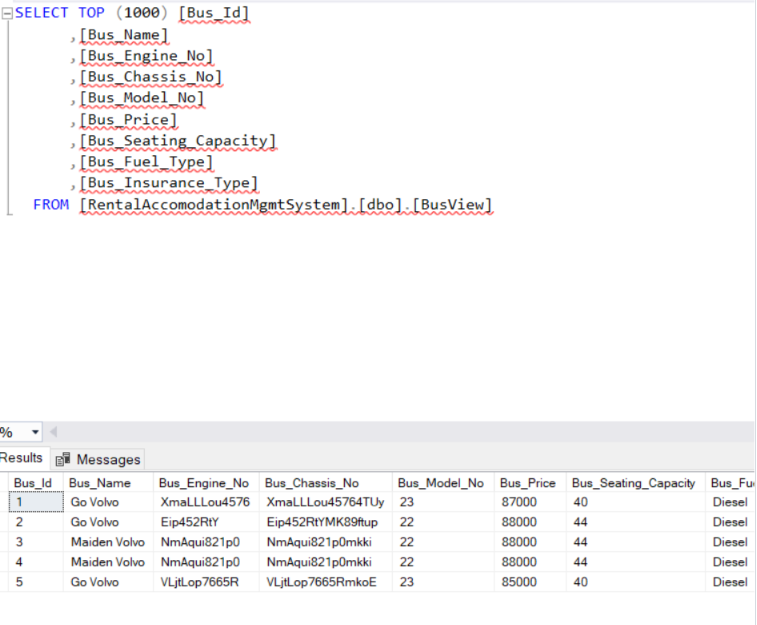
**Figure 12: Trigger 2**



# 7.3. VIEWS

1. Create View BusView AS

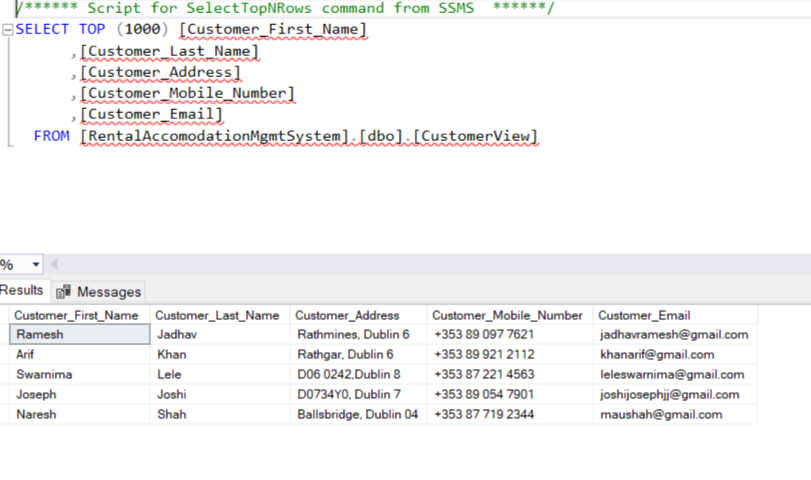
SELECT Bus\_Id, Bus\_Name, Bus\_Engine\_No, Bus\_Chassis\_No, Bus\_Model\_No, Bus\_Price, Bus\_Seating\_Capacity, Bus\_Fuel\_Type, Bus\_Insurance\_Type FROM dbo.Bus



**Figure 13: View 1**

1. CREATE VIEW CustomerView AS

SELECT Customer\_First\_Name, Customer\_Last\_Name, Customer\_Address, Customer\_Mobile\_Number, Customer\_Email FROM dbo.Customer



**Figure 14: View 2**

# 8. Conclusion

In contrast to previous practice, where all activities related to the auto rental industry were restricted to a single physical place, the bus reservation industry has evolved with a new perk. The power of the internet has changed the nature of functions and how these tasks are accomplished, even while the physical place has not entirely disappeared.

Customers can now book buses online, rent automobiles online, and, after becoming a member, either have the bus delivered to their house. The web-based approach has provided a benefit to both customers and bus reservation Websites, allowing them to run their business and successfully and efficiently meet customer needs.

# 9. Bibiliography

Ms. Supriya Khaitan1,Shivalika Sisodia2, Shivam Jaiswal3, Abhinav Kabra4, 2021. “Online Reservation System” - Volume 7 (ISSN:1583-6258) – annalsofrscb

Oloyede, M.O.1 \*, Alaya S.M.2 Adewole, K.S3, 2014. “Development of an Online Bus Ticket Reservation System for a Transportation Service” -Vol.5, No.12 – iiste

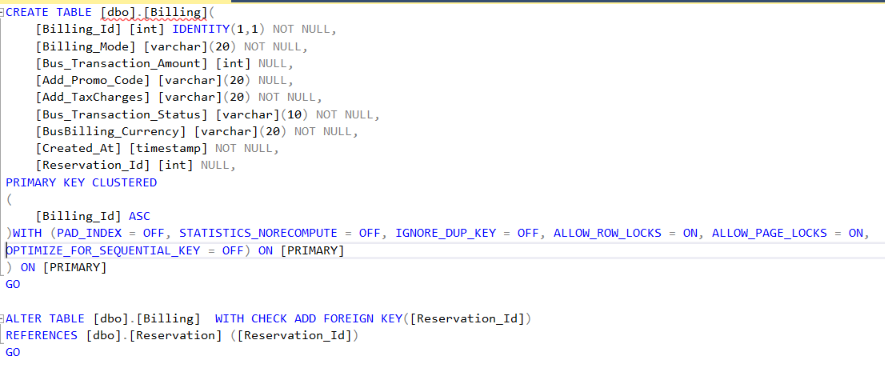
# Mateusz Grzelak, Łukasz Napierała, Łukasz Napierała, Vincent Karovič & Iryna Ivanochko, 2019 – “Bus Ticket Reservation System Agile Methods of Projects Management” – INCoS

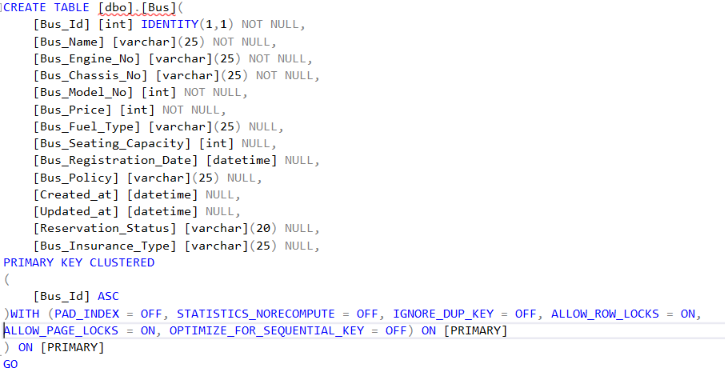
Nwakanma Ifeanyi Cosmas, Etus C, Ajere I.U. & Agomuo Uchechukwu Godswill ,2015. “Online Bus Ticket Reservation System” – Vol 1 – iiardonline

Ayman R. Mohammed, Sally S. Kassem, 2021. “UML Modeling of Online Public Bus Reservation System in Egypt” – IEEE

A. Ibrahim and A. Ta'a, "MOBILE - BASED BUS TICKETING SYSTEM IN IRAQ", *European Journal of Computer Science and Information Technology*, vol. 3, no. 5, pp. 42-55, Nov. 2015.

# 10. Appendix A: Create Table Queries





|  |  |  |
| --- | --- | --- |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
|  | |  |
| 11. Appendix B: Insert Table Queries **INSERT INTO [dbo].[Billing]**  **([BillingMode]**  **,[BusTransactionAmount]**  **,[AddPromoCode]**  **,[AddTaxCharges]**  **,[BusTransactionStatus]**  **,[BusBillingCurrency]**  **B9IS100 Advanced Databases, Dublin Business School**  **40**  **,[ReservationId])**  **VALUES**  **('Fixed',12,'FESTIVAL20',0.11,'In ','Euro',1) GO**  **INSERT INTO [dbo].[BusRental] ([BusRentalTripSelection] ,[BusRentalSeaters] ,[BusRentalFuelType] ,[BusRentalBusType] ,[BusRentalAutoPilot] ,[BusRentalStartDate] ,[BusRentalEndDate])**  **VALUES ('Confirm',5,'Diesel',1,0,'2022-01-21','2022-01-27')**  **GO**  **INSERT INTO [dbo].[Customer]**  **([CustomerFirstName]**  **,[CustomerLasName]**  **,[CustomerAddress]**  **,[CustomerMobileNumber]**  **,[CustomerRole]**  **,[CustomerDateofBirth]**  **,[CustomerEmail]**  **,[CustomerPassword]**  **,[CustomerLicense]**  **,[CreatedAt])**  **VALUES**  **('Arif'**  **,'Khan'**  **,'Rathgar, Dublin 6'**  **,'+353 89 921 2112'**  **,'Student'**  **,'1994-07-31'**  **,'khanarif@gmail.com'**  **,'Nolva#21'**  **,'20150623117899'**  **,'2015-06-23')**  **GO**  **INSERT INTO [dbo].[ReimbursementLogs]**  **([ReimbursementBackLogStatus]**  **,[LastUpdatedAt]**  **,[ReimbursementId])**  **VALUES**  **('Processed'**  **,'2022-12-25'**  **,5)**  **GO**  **INSERT INTO [dbo].[Reimbursement] ([CustomerFirstName] ,[DeductionReimbursedAmount] ,[ReimbursementStatus] ,[ReimbursementAmount] ,[ReimbursementAccount] ,[CreatedAt] ,[ReservationId])**  **VALUES ('Naresh','2','Confirm',2,2,'2022-11-12',1) GO**  **INSERT INTO [dbo].[Bus] ([BusName] ,[BusEngineNo] ,[BusChassisNo] ,[BusModelNo] ,[BusPrice] ,[BusFuelType] ,[BusSeatingCapacity] ,[BusRegistrationDate] ,[BusPolicy] ,[Createdat] ,[Updatedat] ,[ReservationStatus] B9IS100 Advanced Databases, Dublin Business School 41 ,[BusInsuranceType])**  **VALUES ('Go Volvo','XmaLLLou4576','XmaLLLou45764TUy',23,87000,'Diesel',40,'1992-11-11', 'Incurred','2008-11-11','2008-11-12','Success','Silver') GO**  **INSERT INTO [dbo].[BusType] ([BusPowerSteering] ,[BusAirConditioner] ,[BusPassengerAirbag] ,[BusDriverAirbag] ,[BusSunRoof] ,[BusDriverSeat] ,[BusFogLights] ,[BusType] ,[BusId])**  **VALUES ('HPS','Rolta',44,1,1,1,2,1,2) GO**  **INSERT INTO [dbo].[Reservation] ([ReservationLastLocation] ,[ReservationDuration] ,[ReservationTripAmount] ,[BusKmStart] ,[BusKmEnd] ,[BusFuelStart] ,[BusFuelEnd] ,[CreatedAt] ,[UpdatedAt] ,[Reservationstatus] ,[BusId] ,[CustomerId] ,[BusRentalId])**  **VALUES ('Dame Street',3,13,678,726,460,455,2022-10-08,2022-10-06,'Confirm',2,4,6) GO** |  |
| 12. Innovation  * Created stored procedures that make it easier for business owners to access the tables   they regularly use.   * Primary key and foreign key restrictions are used to handle linked tables while removing   and adding new data, in addition to processing standard data.   * Created views to simplify access to the tables that business owners utilize the most. * We do not have deletion and updating since we implemented the data in third normal form,   and have therefore developed stored methods for the anomaly.   * It is more safe and abstract since booking and payment information is stored in separate tables. | |  |
|  | | USE [RentalAccomodationMgmtSystem] |
|  | | GO |
|  | |  |
|  | | /\*\*\*\*\*\* Object: Table [dbo].[Reservation] Script Date: 28-Dec-22 1:17:31 AM \*\*\*\*\*\*/ |
|  | | SET ANSI\_NULLS ON |
|  | | GO |
|  | |  |
|  | | SET QUOTED\_IDENTIFIER ON |
|  | | GO |
|  | |  |