

# ITC607 Data Management

## Assignment 2



Faculty of Health, Humanities and Computing

Please sign the following statement: "I declare that this assignment submission will be my own work and I will not collude with anyone else on the preparation of this Assignment."

**Name / Student ID:** Benjamin King - 2014006529 **Date:** 11 June 2019

Task	Mark	Result
1. Requirements	10	
2. Logical Design	10	
3. SQL	10	
<b>Sum</b>	<b>30</b>	

## Contents

Part 1: List and work out the requirements for the database .....	5
Tables & Queries .....	5
- Cars .....	5
- Insurance Details.....	5
- Garages .....	5
- Garage Services.....	5
- Outgoing Expenses.....	5
- Incoming Revenue.....	5
- Customers .....	5
- VIP Customers.....	5
- Casual Customers.....	5
- Bookings.....	5
Relationships.....	6
Part 2: Entity Relationship Diagram .....	7
Part 3: Creation of SQL Database.....	8
Garage - Table .....	8
SQL Code .....	8
Screenshot #1.....	8
Sample Data .....	8
Screenshot #2.....	8
Garage Services – Table .....	9
SQL Code .....	9
Screenshot #1.....	9
Sample Data .....	9
Screenshot #2.....	9
Cars - Table.....	10
SQL Code .....	10
Screenshot #1.....	10
Sample Data .....	10
Screenshot #2.....	10
Insurance Details - Table.....	11
SQL Code .....	11
Screenshot #1.....	11
Sample Data .....	11
Screenshot #2.....	11

Outgoing Expenditure - Table .....	12
SQL Code .....	12
Screenshot #1.....	12
Sample Data .....	12
Screenshot #2.....	12
Customers - Table .....	13
SQL Code .....	13
Screenshot #1.....	13
Sample Data .....	13
Screenshot #2.....	13
Customers Casual - Table .....	14
SQL Code .....	14
Screenshot #1.....	14
Sample Data .....	14
Screenshot #2.....	14
Bookings - Table .....	15
SQL Code .....	15
Screenshot #1.....	15
Sample Data .....	15
Screenshot #2.....	16
Incoming Revenue - Table.....	17
SQL Code .....	17
Screenshot #1.....	17
Sample Data .....	17
Screenshot #2.....	17
Customers VIP - Table .....	18
SQL Code .....	18
Screenshot #1.....	18
Sample Data .....	18
Screenshot #2.....	18
Cars which were purchased more than 1 year ago - View .....	19
SQL Code .....	19
Screenshot #1.....	19
Screenshot of the View .....	19
Cars that were purchased more than 1 Year Ago - Trigger .....	20
SQL Code .....	20

Screenshot #1.....	20
Example Data .....	20
Screenshot #2.....	20
Finished Database Entity Relationship Diagram .....	21
Marking Schedule .....	22

## Part 1: List and work out the requirements for the database.

### Tables & Queries

The Purpose of this database is to store information about Cheap Automobile Rentals (CAR) cars and customers, and so the database will therefore have the following tables:

- **Cars** – To store the details of the Rental Cars of CAR's
  - o Primary Key – Registration Number
  - o Foreign Key – Garage ID
- **Insurance Details** – To store the details of the Insurance for each Car of CAR's
  - o Primary Key – Registration Number (Also a Foreign Key)
- **Garages** – To store the details of the Garages and subcontractors which CAR's uses
  - o Primary Key – Garage ID
- **Garage Services** – To store the details of the costs and services each garage provides
  - o Primary Key – Garage ID (Also a Foreign Key)
- **Outgoing Expenses** – To store any details related to any outgoing expenses of CAR's
  - o Primary Key – Expense ID
  - o Foreign Key – Registration Number
- **Incoming Revenue** – To store any details related to any Incoming Revenue of CAR's
  - o Primary Key – Income ID
  - o Foreign Key – Registration Number
- **Customers** – To store the Personal Details of CAR's customers
  - o Primary Key – Customer ID
- **VIP Customers** – A Sub-Table of the Customers table which store the extra details of customers which are VIP Customers
  - o Primary Key – Customer ID (Also a Foreign Key)
  - o Foreign Key – Booking ID
- **Casual Customers** – A sub-table of the Customers table which stores the extra details of customers that are only Casual Customers
  - o Primary Key – Customer ID (Also a Foreign Key)
- **Bookings** – Stores the details of Car Rental Bookings, with the relevant relationships to the Cars and the Customers/VIP-Customers Table.
  - o Primary Key – Booking ID
  - o Foreign Key – Customer ID
  - o Foreign Key – Registration Number
  - o No End Date of a Booking can be set before the Start Date (as that doesn't make sense)
  - o A Booking can only be booked ahead within 1 month of the current date.
- A View which displays the Cars which were purchased greater than 1 year ago from the current date (because of how it is the *company policy not to keep any car for a period exceeding one year*).
- A way to alert the user inputting the data if a Car was purchased greater than 1 year ago from the current date (such as a Trigger). However, this should allow the user to enter the Car into the database, for circumstances where they might not be able to or willing to sell the car.

In this database I have decided to create a lookup query (in a View), as well as creating a Trigger for the CARS Table. I have created this because of how a lookup Query in a View easily allows the user to see what cars are older than 1 year ago (as new cars will be added to the table as records not allowing the user to easily see what cars are older than a year from the purchase date). However by also including a Trigger which checks whether the car was purchased more than a year ago, when the users enters this car into the database it will Alert the User with a message, alerting the user that it is an old car however still storing the car in the table. I have created this with how when the data is first entered into the table there might be some cars which are old, that CAR has not realised. However, they can still store the information of these cars in the database, as some of these cars might be an exception to their business rule (such as Executive Cars which might be kept for longer than a year).

### Relationships

**The Database needs to have the following Database Relationships:**

One or More Customers can be only one *VIP* Customer each. There can be more than 1 *VIP* Customer. (M:1)

One or More Customers can be only one *Casual* Customer each. There can be more than 1 *VIP* Customer (M:1).

A Customer can only be either a *VIP* or a *Casual* Customer.

Zero or more Cars can refer to one or more Outgoing Expenditures, (M:M).

Zero or more Cars can refer to one or more Incoming Expenditures. (M:M).

Zero or more Bookings can refer to only one Incoming Revenue (M:1) – Assumption that all bookings are recorded as 1 Payment (Deposit + Booking Charge are recorded together).

One or more Cars can have only one garage. – Assumption is that there each car has only one Garage that it goes to (but a garage can have more than 1 car that it goes to) (M:1).

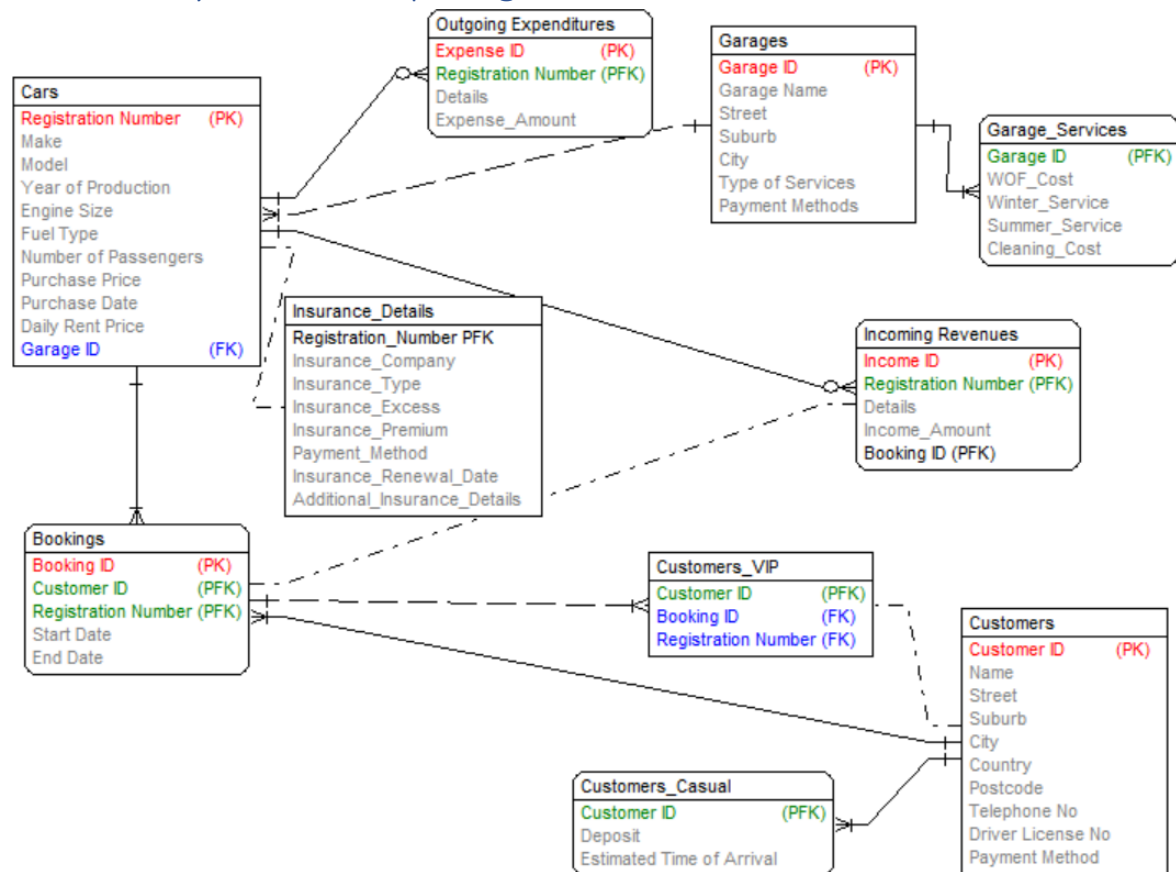
One Car can refer to One or More Bookings (not at the same time however) (1:M).

Only One Customer can refer to one or more Bookings. (1:M) – Assumption is a booking can only have 1 Customer however, a customer can book more than 1 booking.

One Car can only refer to one Insurance Detail entry – Assumption is that all Cars are insured (1:1).

One Garage can only refer to one Garage Service Entry (1:1).

## Part 2: Entity Relationship Diagram



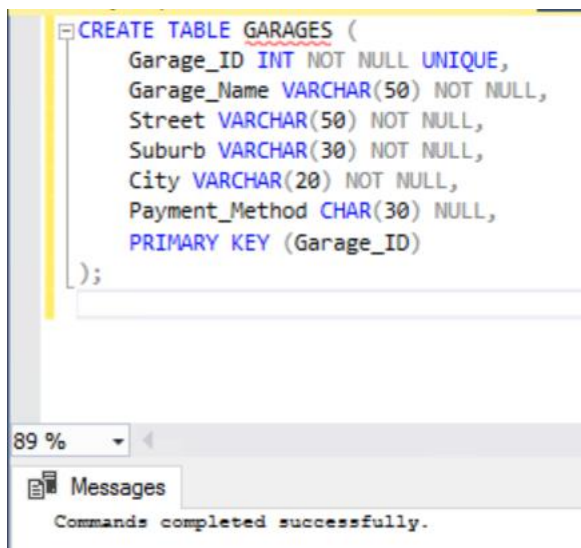
## Part 3: Creation of SQL Database

### Garage - Table

#### SQL Code

```
CREATE TABLE GARAGES (  
    Garage_ID INT NOT NULL UNIQUE,  
    Garage_Name VARCHAR(50) NOT NULL,  
    Street VARCHAR(50) NOT NULL,  
    Suburb VARCHAR(30) NOT NULL,  
    City VARCHAR(20) NOT NULL,  
    Payment_Method CHAR(30) NULL,  
    PRIMARY KEY (Garage_ID)  
);
```

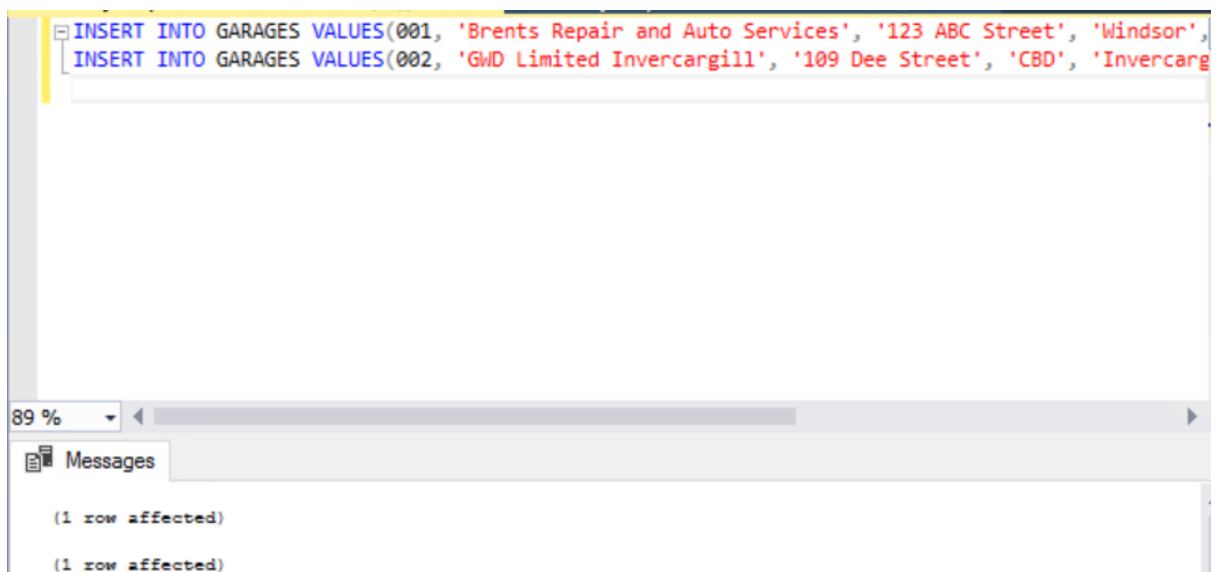
#### Screenshot #1



#### Sample Data

```
INSERT INTO GARAGES VALUES(001, 'Brents Repair and Auto Services', '123 ABC Street', 'Windsor', 'Invercargill', 'Credit');  
INSERT INTO GARAGES VALUES(002, 'GWD Limited Invercargill', '109 Dee Street', 'CBD', 'Invercargill', 'Eftpos Only');
```

#### Screenshot #2



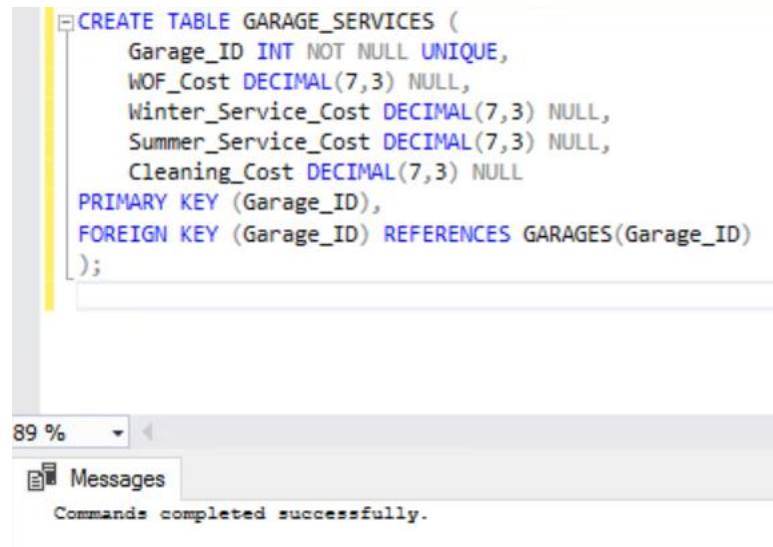


## Garage Services – Table

### SQL Code

```
CREATE TABLE GARAGE_SERVICES (  
    Garage_ID INT NOT NULL UNIQUE,  
    WOF_Cost DECIMAL(7,3) NULL,  
    Winter_Service_Cost DECIMAL(7,3) NULL,  
    Summer_Service_Cost DECIMAL(7,3) NULL,  
    Cleaning_Cost DECIMAL(7,3) NULL  
PRIMARY KEY (Garage_ID),  
FOREIGN KEY (Garage_ID) REFERENCES GARAGES(Garage_ID)  
);
```

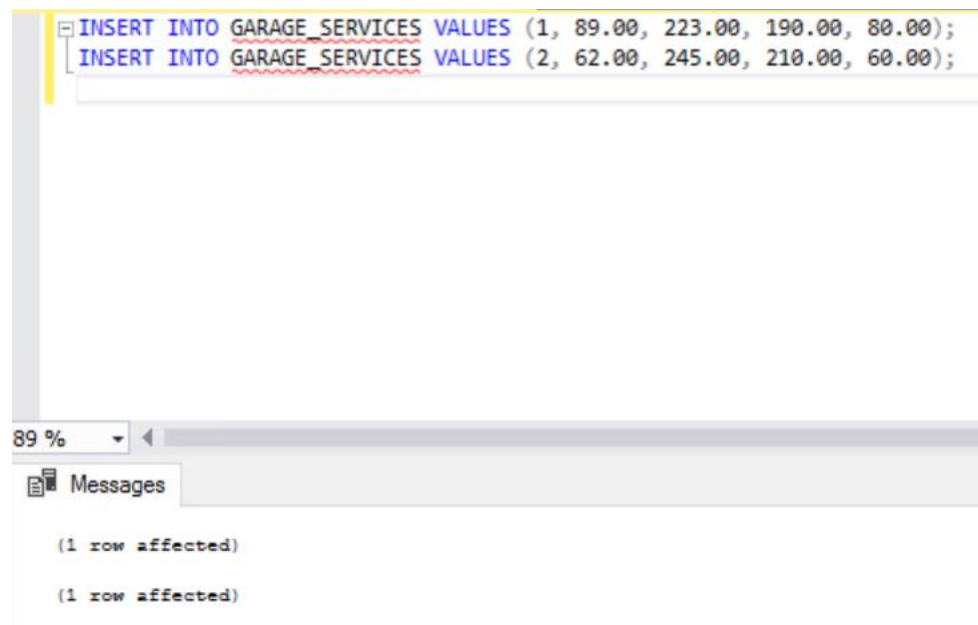
### Screenshot #1



### Sample Data

```
INSERT INTO GARAGE_SERVICES VALUES (1, 89.00, 223.00, 190.00, 80.00);  
INSERT INTO GARAGE_SERVICES VALUES (2, 62.00, 245.00, 210.00, 60.00);
```

### Screenshot #2



## Cars - Table

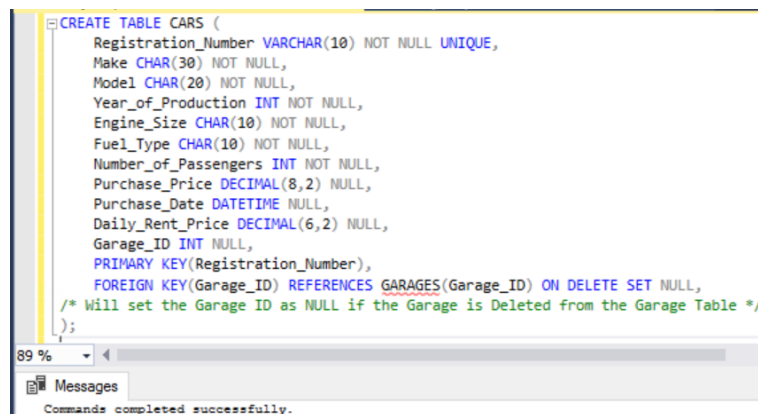
## SQL Code

```

CREATE TABLE CARS (
    Registration_Number VARCHAR(10) NOT NULL UNIQUE,
    Make CHAR(30) NOT NULL,
    Model CHAR(20) NOT NULL,
    Year_of_Production INT NOT NULL,
    Engine_Size CHAR(10) NOT NULL,
    Fuel_Type CHAR(10) NOT NULL,
    Number_of_Passengers INT NOT NULL,
    Purchase_Price DECIMAL(8,2) NULL,
    Purchase_Date DATETIME NULL,
    Daily_Rent_Price DECIMAL(6,2) NULL,
    Garage_ID INT NULL,
    PRIMARY KEY(Registration_Number),
    FOREIGN KEY(Garage_ID) REFERENCES GARAGES(Garage_ID) ON DELETE SET NULL,
    /* Will set the Garage ID as NULL if the Garage is Deleted from the Garage Table */
);

```

## Screenshot #1



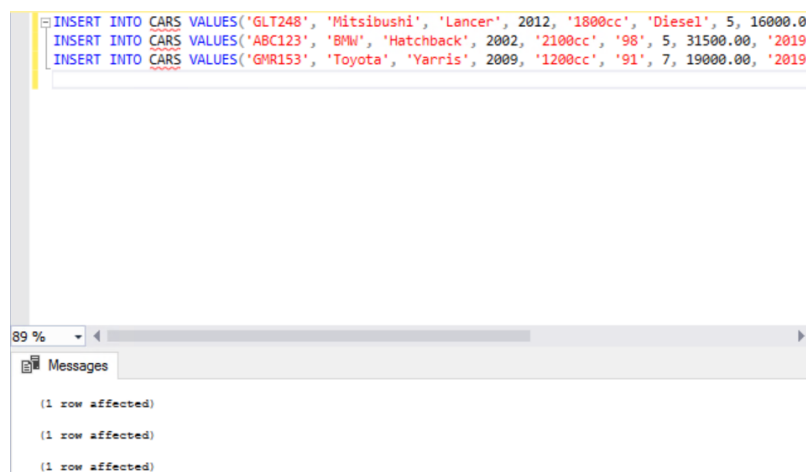
## Sample Data

```

INSERT INTO CARS VALUES('GLT248', 'Mitsubishi', 'Lancer', 2012, '1800cc', 'Diesel', 5, 16000.00, '2018-01-03', 180.00, 001);
INSERT INTO CARS VALUES('ABC123', 'BMW', 'Hatchback', 2002, '2100cc', '98', 5, 31500.00, '2019-01-03', 240.00, 002);
INSERT INTO CARS VALUES('GMR153', 'Toyota', 'Yarris', 2009, '1200cc', '91', 7, 19000.00, '2019-03-03', 180.00, 001);

```

## Screenshot #2

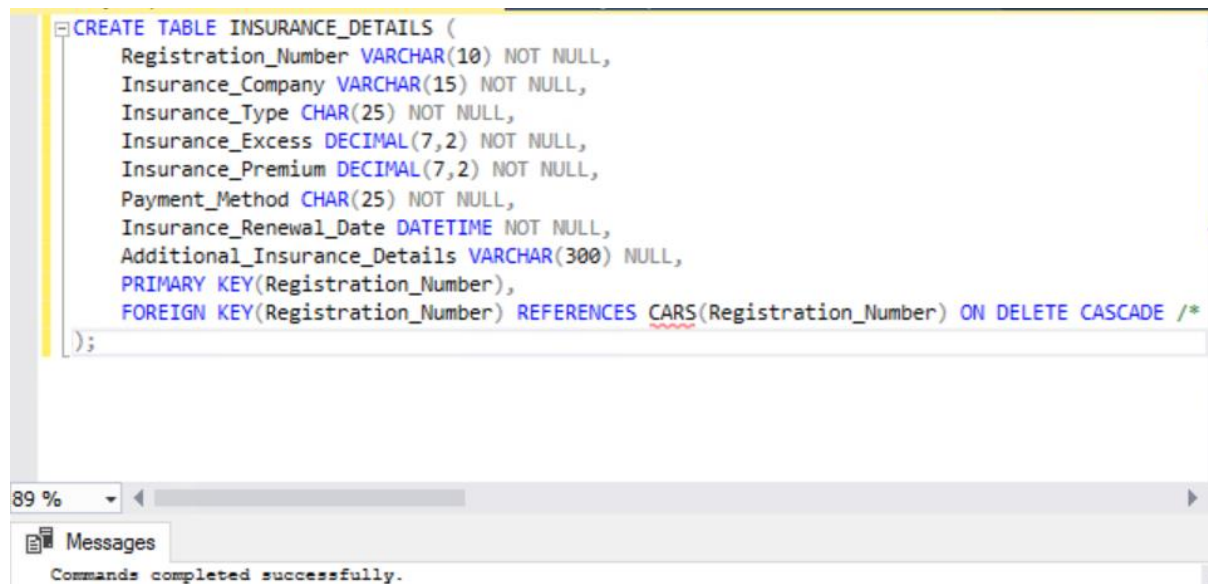


## Insurance Details - Table

## SQL Code

```
CREATE TABLE INSURANCE_DETAILS (
    Registration_Number VARCHAR(10) NOT NULL,
    Insurance_Company VARCHAR(15) NOT NULL,
    Insurance_Type CHAR(25) NOT NULL,
    Insurance_Excess DECIMAL(7,2) NOT NULL,
    Insurance_Premium DECIMAL(7,2) NOT NULL,
    Payment_Method CHAR(25) NOT NULL,
    Insurance_Renewal_Date DATETIME NOT NULL,
    Additional_Insurance_Details VARCHAR(300) NULL,
    PRIMARY KEY(Registration_Number),
    FOREIGN KEY(Registration_Number) REFERENCES CARS(Registration_Number) ON DELETE
CASCADE /* If the Car is Deleted then the Insurance Details for that car will be deleted
- ensuring that Referential Integrity is intact so that the data is deleted for the car
and not left behind */
);
```

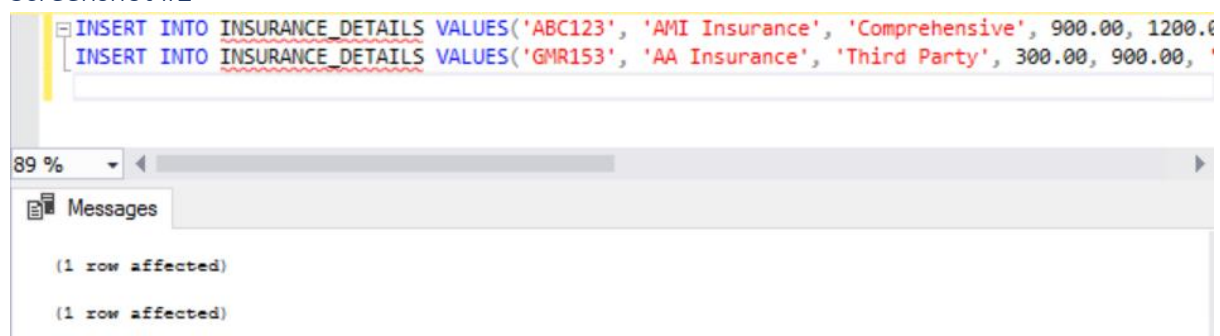
## Screenshot #1



## Sample Data

```
INSERT INTO INSURANCE_DETAILS VALUES('ABC123', 'AMI Insurance', 'Comprehensive', 900.00, 1200.00, 'Direct Credit', '2018-09-12', 'Car has had 2 Claims against it since we brought it');
INSERT INTO INSURANCE_DETAILS VALUES('GMR153', 'AA Insurance', 'Third Party', 300.00, 900.00, 'Bank Deposit', '2019-04-12', NULL);
```

## Screenshot #2

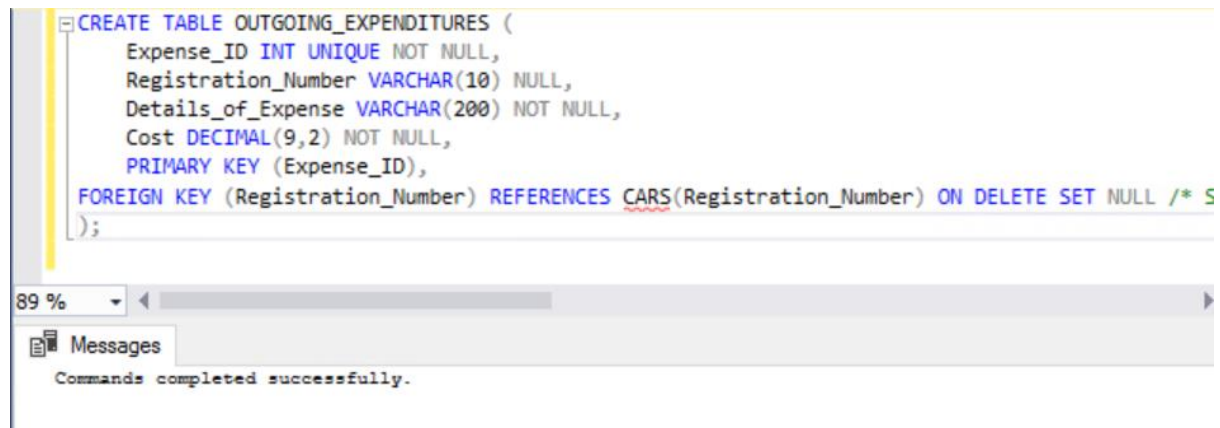


## Outgoing Expenditure - Table

### SQL Code

```
CREATE TABLE OUTGOING_EXPENDITURES (
    Expense_ID INT UNIQUE NOT NULL,
    Registration_Number VARCHAR(10) NULL,
    Details_of_Expense VARCHAR(200) NOT NULL,
    Cost DECIMAL(9,2) NOT NULL,
    PRIMARY KEY (Expense_ID),
    FOREIGN KEY (Registration_Number) REFERENCES CARS(Registration_Number) ON DELETE
    SET NULL /* Sets the Registration Number as NULL - important so that the Expenses
    are still able to be kept for Auditing Requirements etc */
);
```

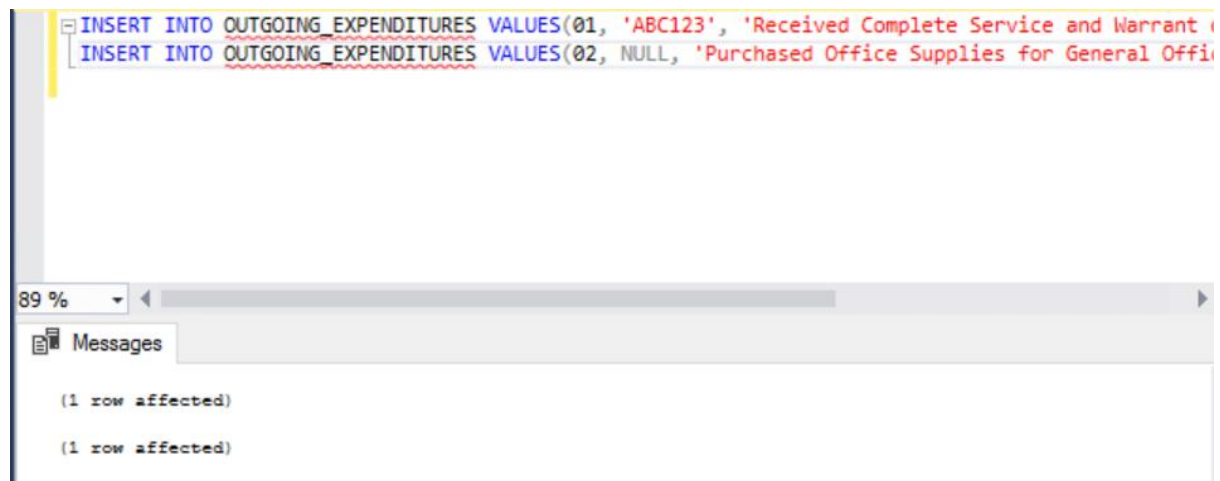
### Screenshot #1



### Sample Data

```
INSERT INTO OUTGOING_EXPENDITURES VALUES (01, 'ABC123', 'Received Complete Service and
Warrant of Fitness', 345.00);
INSERT INTO OUTGOING_EXPENDITURES VALUES (02, NULL, 'Purchased Office Supplies for
General Office', 125.9);
```

### Screenshot #2



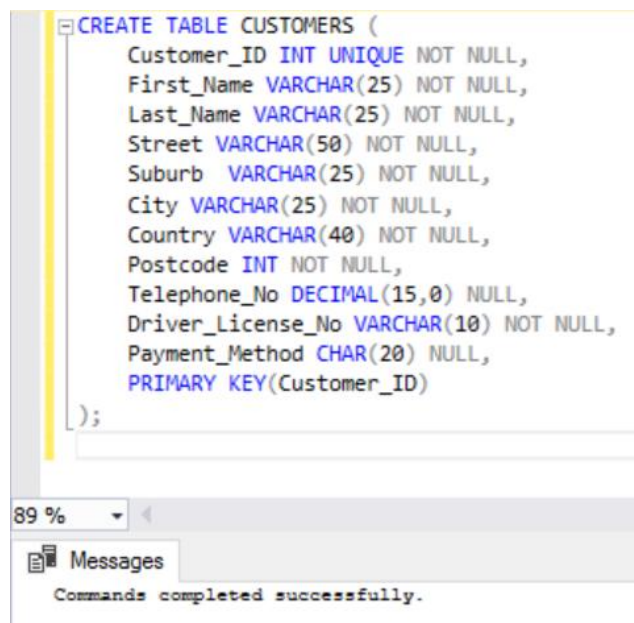
*In this table the Registration Number Field is NULL with how for expenses such as Office Expenses there will not be a Registration Number associated with that Expense*

## Customers - Table

## SQL Code

```
CREATE TABLE CUSTOMERS (
    Customer_ID INT UNIQUE NOT NULL,
    First_Name VARCHAR(25) NOT NULL,
    Last_Name VARCHAR(25) NOT NULL,
    Street VARCHAR(50) NOT NULL,
    Suburb VARCHAR(25) NOT NULL,
    City VARCHAR(25) NOT NULL,
    Country VARCHAR(40) NOT NULL,
    Postcode INT NOT NULL,
    Telephone_No DECIMAL(15,0) NULL,
    Driver_License_No VARCHAR(10) NOT NULL,
    Payment_Method CHAR(20) NULL,
    PRIMARY KEY(Customer_ID)
);
```

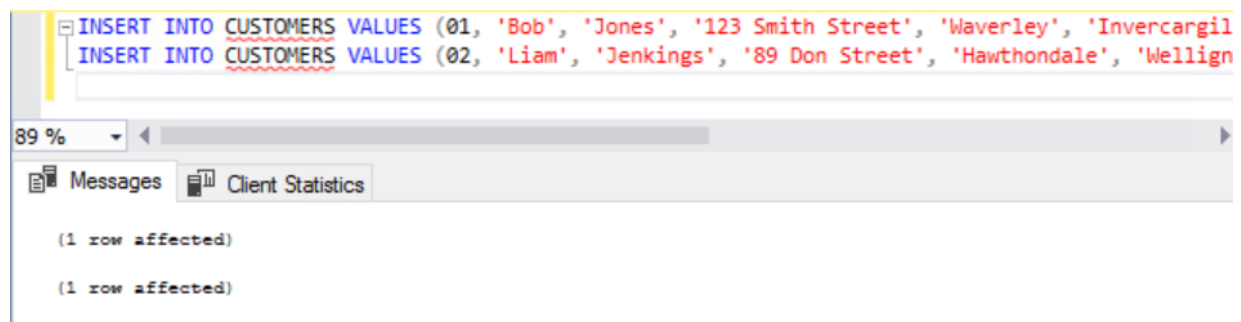
## Screenshot #1



## Sample Data

```
INSERT INTO CUSTOMERS VALUES (01, 'Bob', 'Jones', '123 Smith Street', 'Waverley', 'Invercargill', 'New Zealand', 9810, 032177821, 'DSC12345', 'Credit');
INSERT INTO CUSTOMERS VALUES (02, 'Liam', 'Jenkins', '89 Don Street', 'Hawthondale', 'Wellington', 'Australia', 1204, 022983232, 'ABC8292', 'Eftpos');
```

## Screenshot #2

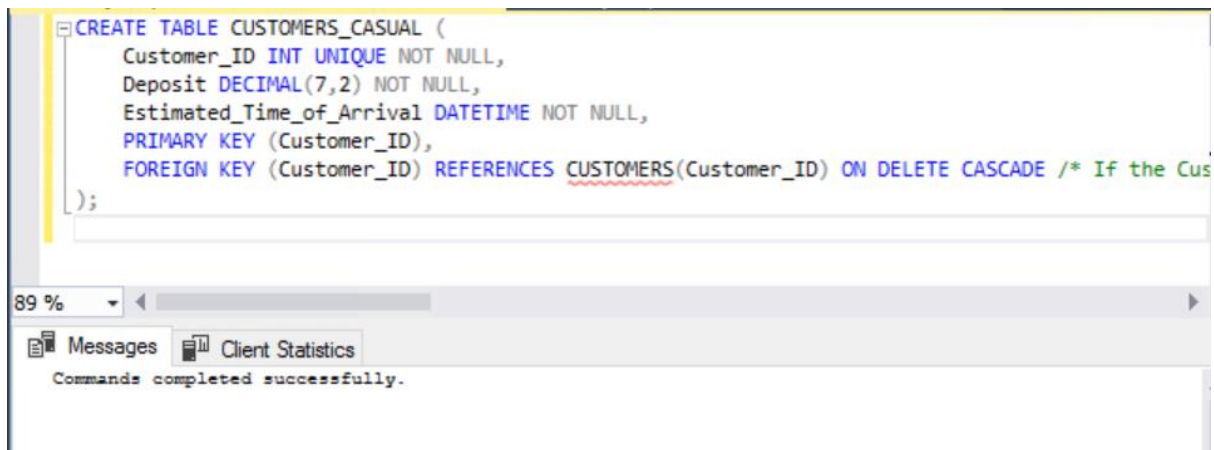


## Customers Casual - Table

## SQL Code

```
CREATE TABLE CUSTOMERS_CASUAL (  
    Customer_ID INT UNIQUE NOT NULL,  
    Deposit DECIMAL(7,2) NOT NULL,  
    Estimated_Time_of_Arrival DATETIME NOT NULL,  
    PRIMARY KEY (Customer_ID),  
    FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS(Customer_ID) ON DELETE CASCADE /*  
If the Customers details are deleted from the CUSTOMERS table it will delete them in  
the Customers_Casual Table - Important for Referential Integrity to ensure it is intact  
so that the data is deleted for the Customer and not left behind */  
);
```

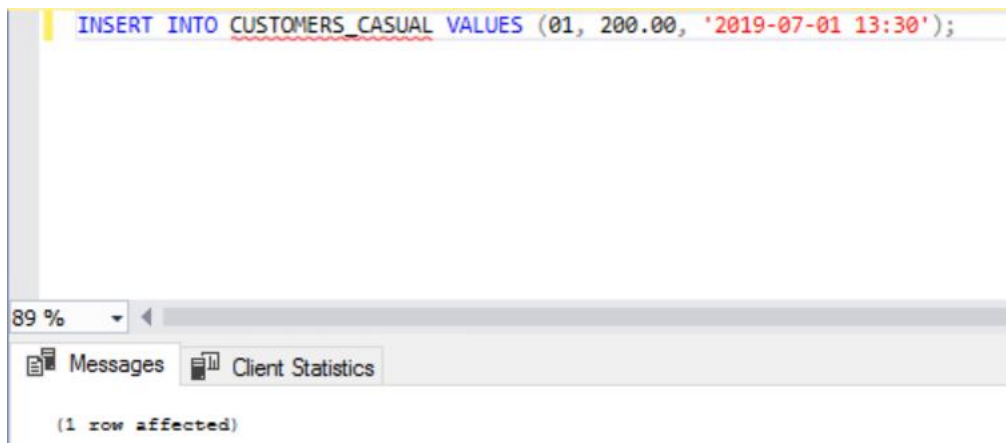
## Screenshot #1



## Sample Data

```
INSERT INTO CUSTOMERS_CASUAL VALUES (01, 200.00, '2019-07-01 13:30');
```

## Screenshot #2

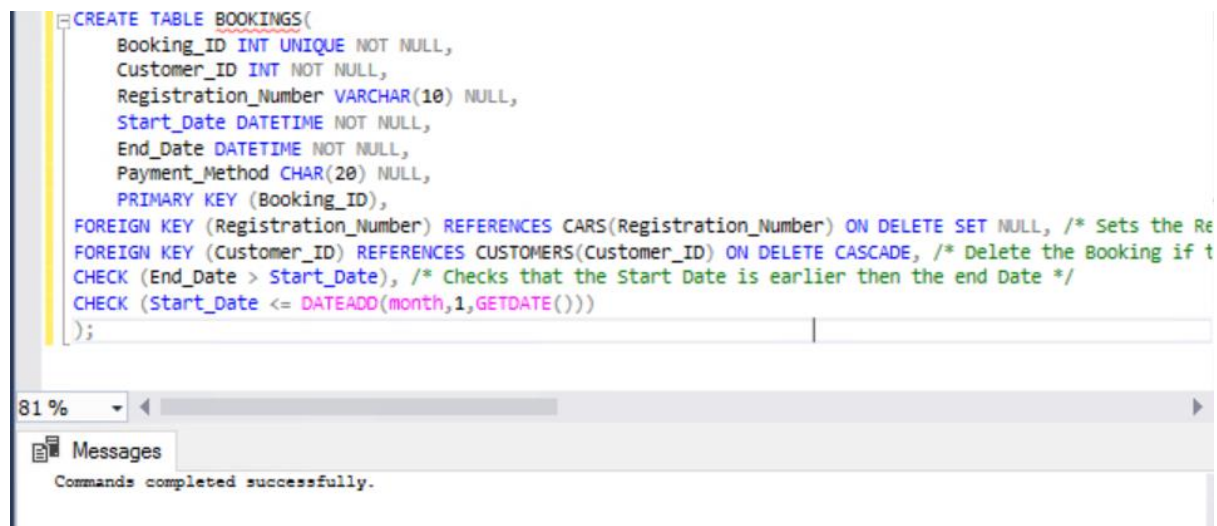


## Bookings - Table

## SQL Code

```
CREATE TABLE BOOKINGS(
    Booking_ID INT UNIQUE NOT NULL,
    Customer_ID INT NOT NULL,
    Registration_Number VARCHAR(10) NULL,
    Start_Date DATETIME NOT NULL,
    End_Date DATETIME NOT NULL,
    Payment_Method CHAR(20) NULL,
    PRIMARY KEY (Booking_ID),
    FOREIGN KEY (Registration_Number) REFERENCES CARS(Registration_Number) ON DELETE SET NULL, /* Sets the Registration Number as NULL if the Car is deleted from the CARS table - don't want to delete the Booking entry if the car gets sold etc */
    FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS(Customer_ID) ON DELETE CASCADE, /* Delete the Booking if the Customer is deleted from the CUSTOMERS Table */
    CHECK (End_Date > Start_Date), /* Checks that the Start Date is earlier then the end Date */
    CHECK (Start_Date <= DATEADD(month,1,GETDATE()))
);
```

## Screenshot #1

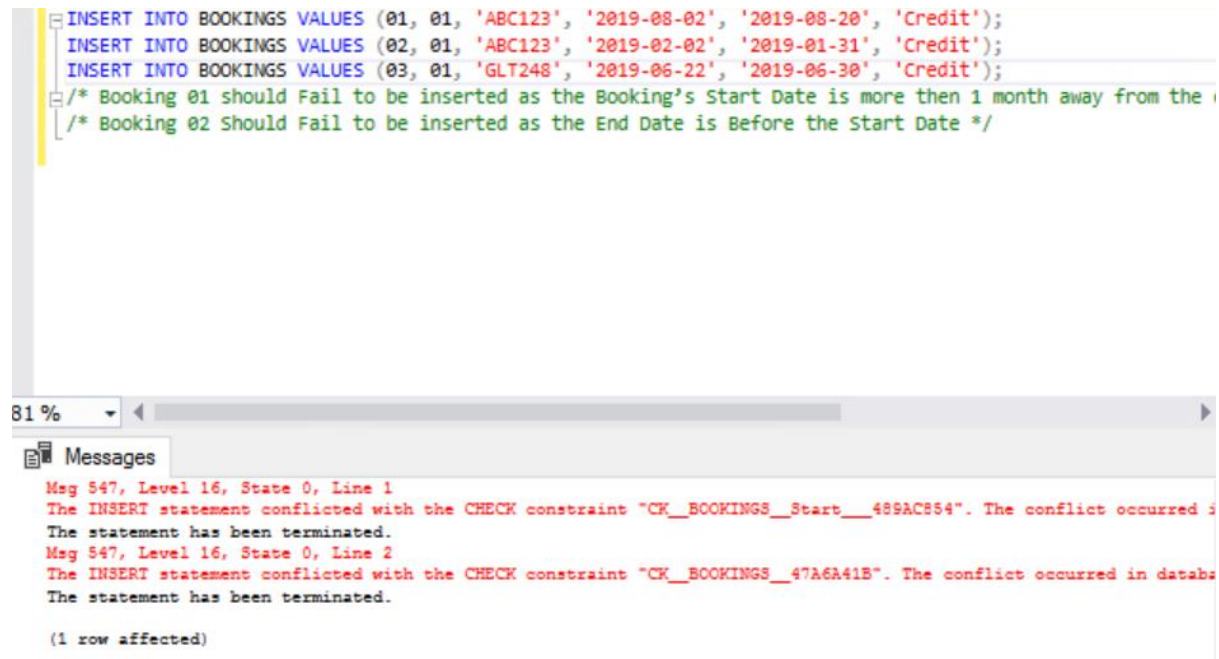


## Sample Data

```
INSERT INTO BOOKINGS VALUES (01, 01, 'ABC123', '2019-08-02', '2019-08-20', 'Credit');
INSERT INTO BOOKINGS VALUES (02, 01, 'ABC123', '2019-02-02', '2019-01-31', 'Credit');
INSERT INTO BOOKINGS VALUES (03, 01, 'GLT248', '2019-06-22', '2019-06-30', 'Credit');
/* Booking 01 should Fail to be inserted as the Booking's Start Date is more then 1 month away from the current date (13 June) */
/* Booking 02 Should Fail to be inserted as the End Date is Before the Start Date */
```



## Screenshot #2



```
INSERT INTO BOOKINGS VALUES (01, 01, 'ABC123', '2019-08-02', '2019-08-20', 'Credit');
INSERT INTO BOOKINGS VALUES (02, 01, 'ABC123', '2019-02-02', '2019-01-31', 'Credit');
INSERT INTO BOOKINGS VALUES (03, 01, 'GLT248', '2019-06-22', '2019-06-30', 'Credit');
/* Booking 01 should Fail to be inserted as the Booking's Start Date is more then 1 month away from the
/* Booking 02 Should Fail to be inserted as the End Date is Before the Start Date */
```

81 %

Messages

Msg 547, Level 16, State 0, Line 1  
The INSERT statement conflicted with the CHECK constraint "CK\_BOOKINGS\_Start\_489AC854". The conflict occurred in database "AdventureWorks". The statement has been terminated.

Msg 547, Level 16, State 0, Line 2  
The INSERT statement conflicted with the CHECK constraint "CK\_BOOKINGS\_47A6A41B". The conflict occurred in database "AdventureWorks". The statement has been terminated.

(1 row affected)

*Both of the Check Constraints work correctly as shown in the above screenshot.*

*Booking 01 Failed to be inserted due to how the Start Date of the Booking is more than 1 month away – which isn't allowed to be booked as the Business Rules only allow bookings within the next month of the current date.*

*Booking 02 Failed to be inserted due to how the Start Date of the booking was before the End Date.*

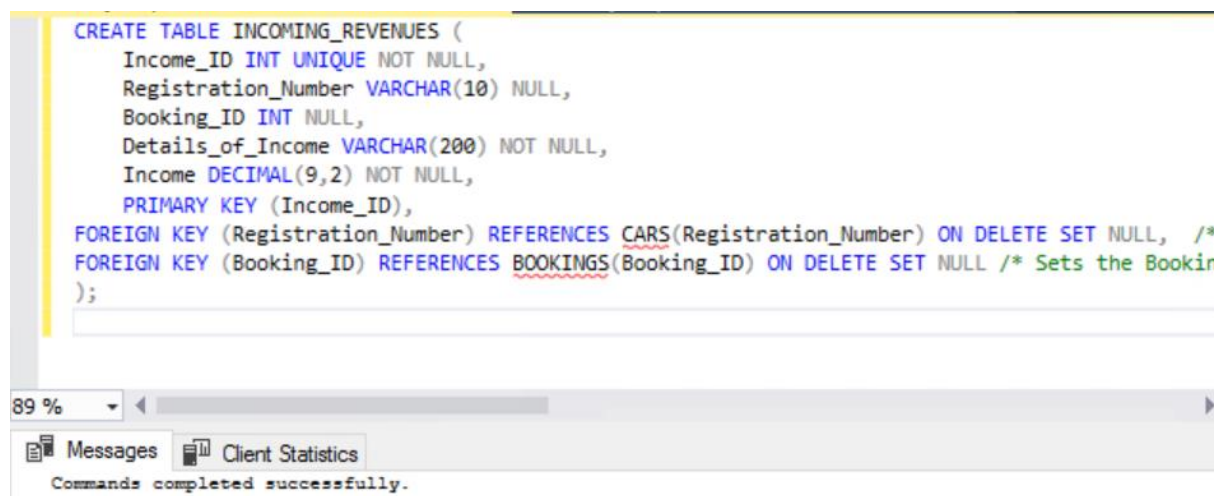


## Incoming Revenue - Table

## SQL Code

```
CREATE TABLE INCOMING_REVENUES (
    Income_ID INT UNIQUE NOT NULL,
    Registration_Number VARCHAR(10) NULL,
    Booking_ID INT NULL,
    Details_of_Income VARCHAR(200) NOT NULL,
    Income DECIMAL(9,2) NOT NULL,
    PRIMARY KEY (Income_ID),
    FOREIGN KEY (Registration_Number) REFERENCES CARS(Registration_Number) ON DELETE SET NULL, /* Sets the Registration Number as NULL - important so that the Incomes
are still able to be kept for Auditing Requirements etc */
    FOREIGN KEY (Booking_ID) REFERENCES BOOKINGS(Booking_ID) ON DELETE SET NULL /*
Sets the Booking ID as NULL - Important so that the Incomes are still able to be
kept for auditing Requirements etc */
);
```

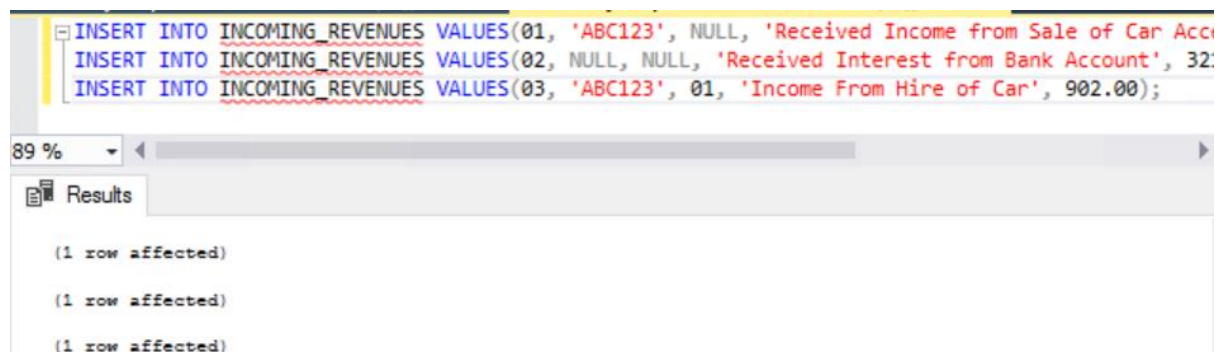
## Screenshot #1



## Sample Data

```
INSERT INTO INCOMING_REVENUES VALUES(01, 'ABC123', NULL, 'Received Income from Sale of Car Accessory', 120.00);
INSERT INTO INCOMING_REVENUES VALUES(02, NULL, NULL, 'Received Interest from Bank Account', 321.99);
INSERT INTO INCOMING_REVENUES VALUES(03, 'ABC123', 01, 'Income From Hire of Car', 902.00);
```

## Screenshot #2



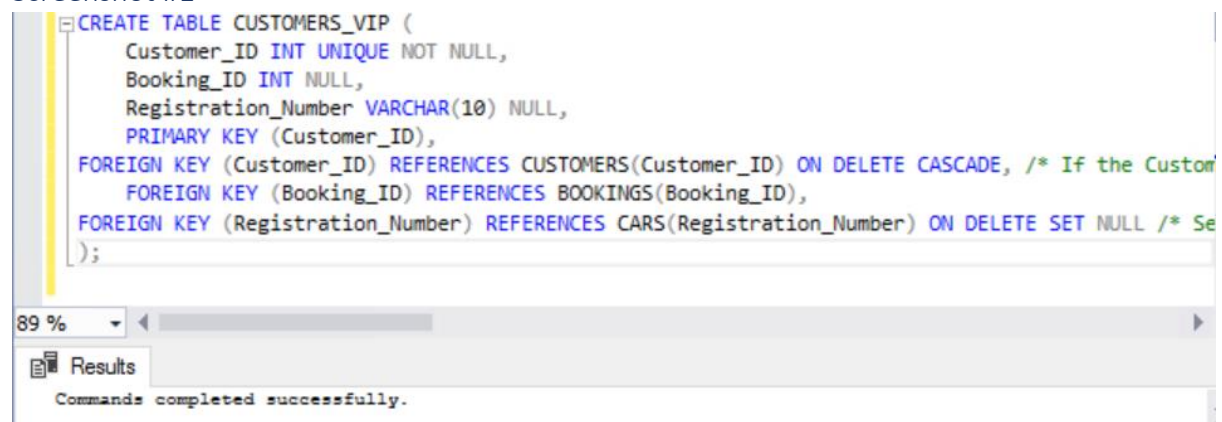
*In this table the Registration Number & Booking Number Fields are NULL with how for some Income Sources such as Interest from Investments there will not be a Registration Number or Booking Number associated with that Income*

## Customers VIP - Table

## SQL Code

```
CREATE TABLE CUSTOMERS_VIP (
    Customer_ID INT UNIQUE NOT NULL,
    Booking_ID INT NULL,
    Registration_Number VARCHAR(10) NULL,
    PRIMARY KEY (Customer_ID),
    FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS(Customer_ID) ON DELETE CASCADE, /* If
the Customers details are deleted from the CUSTOMERS_VIP table it will delete them in
the Customers_Casual Table - Important for Referential Integrity to ensure it is intact
so that the data is deleted for the Customer and not left behind */
    FOREIGN KEY (Booking_ID) REFERENCES BOOKINGS(Booking_ID),
    FOREIGN KEY (Registration_Number) REFERENCES CARS(Registration_Number) ON DELETE SET
NULL /* Set the Registration Number as Null if the Car is deleted from the database */
);
```

## Screenshot #1



```
CREATE TABLE CUSTOMERS_VIP (
    Customer_ID INT UNIQUE NOT NULL,
    Booking_ID INT NULL,
    Registration_Number VARCHAR(10) NULL,
    PRIMARY KEY (Customer_ID),
    FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS(Customer_ID) ON DELETE CASCADE, /* If the Custom
    FOREIGN KEY (Booking_ID) REFERENCES BOOKINGS(Booking_ID),
    FOREIGN KEY (Registration_Number) REFERENCES CARS(Registration_Number) ON DELETE SET NULL /* Se
);
```

89 %

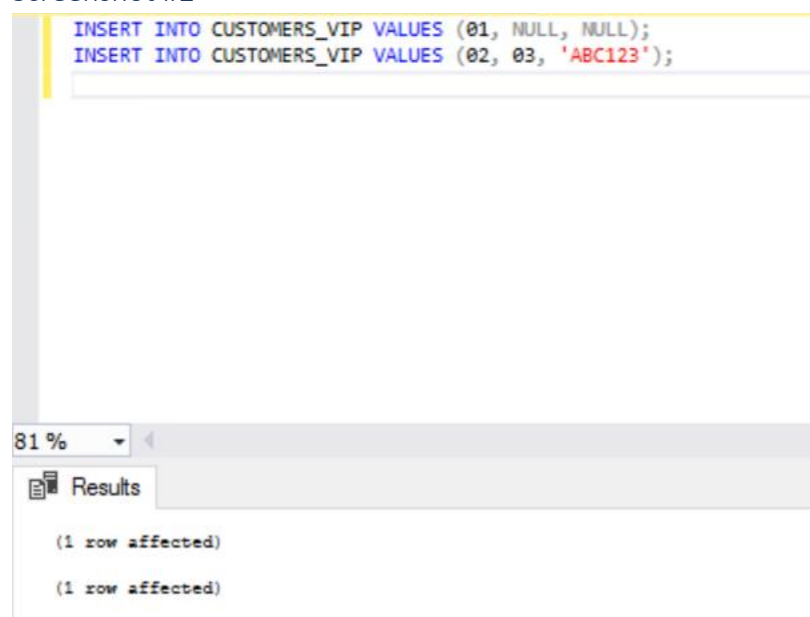
Results

Commands completed successfully.

## Sample Data

```
INSERT INTO CUSTOMERS_VIP VALUES (01, NULL, NULL);
INSERT INTO CUSTOMERS_VIP VALUES (02, 03, 'ABC123');
```

## Screenshot #2



```
INSERT INTO CUSTOMERS_VIP VALUES (01, NULL, NULL);
INSERT INTO CUSTOMERS_VIP VALUES (02, 03, 'ABC123');
```

81 %

Results

(1 row affected)

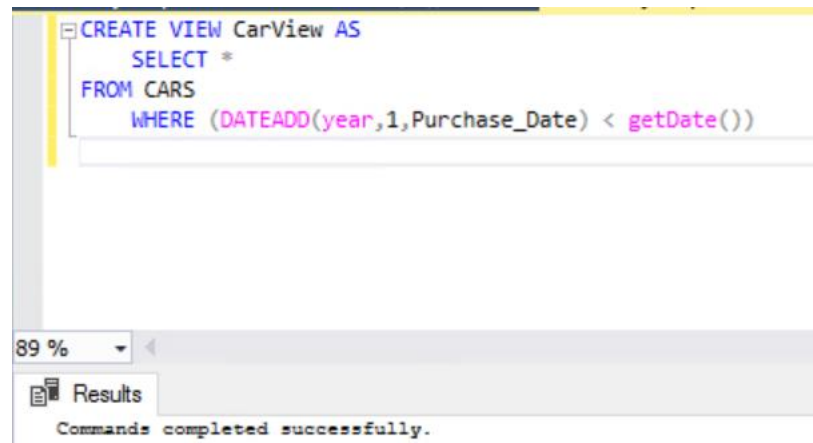
(1 row affected)

Cars which were purchased more than 1 year ago - View

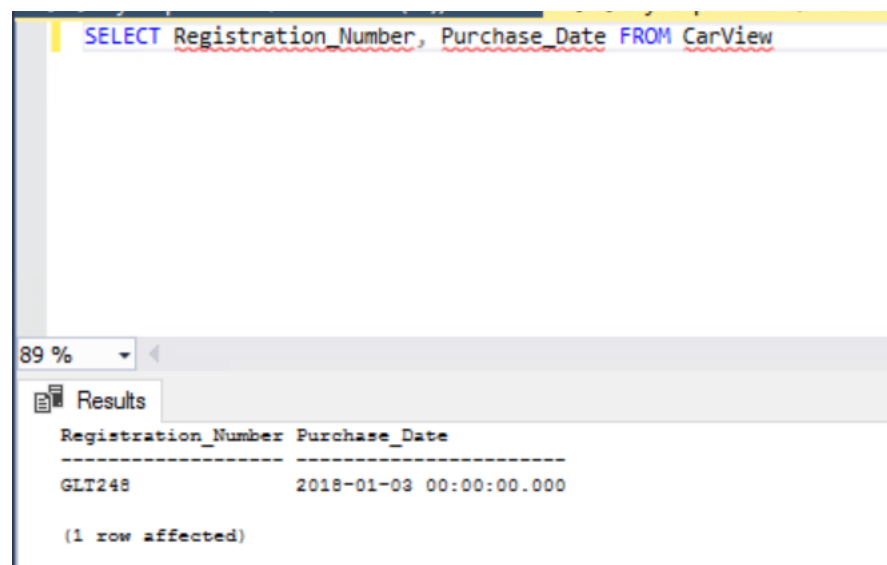
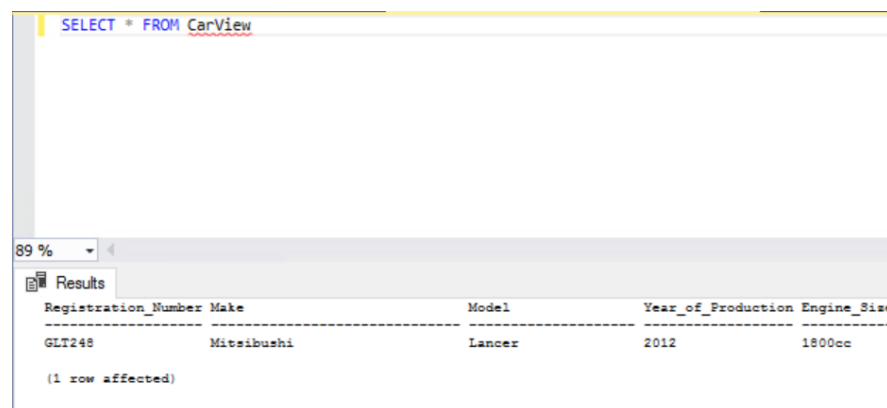
SQL Code

```
CREATE VIEW CarView AS
SELECT *
FROM CARS
WHERE (DATEADD(year,1,Purchase_Date) < getDate())
```

Screenshot #1



Screenshot of the View



## Cars that were purchased more than 1 Year Ago - Trigger

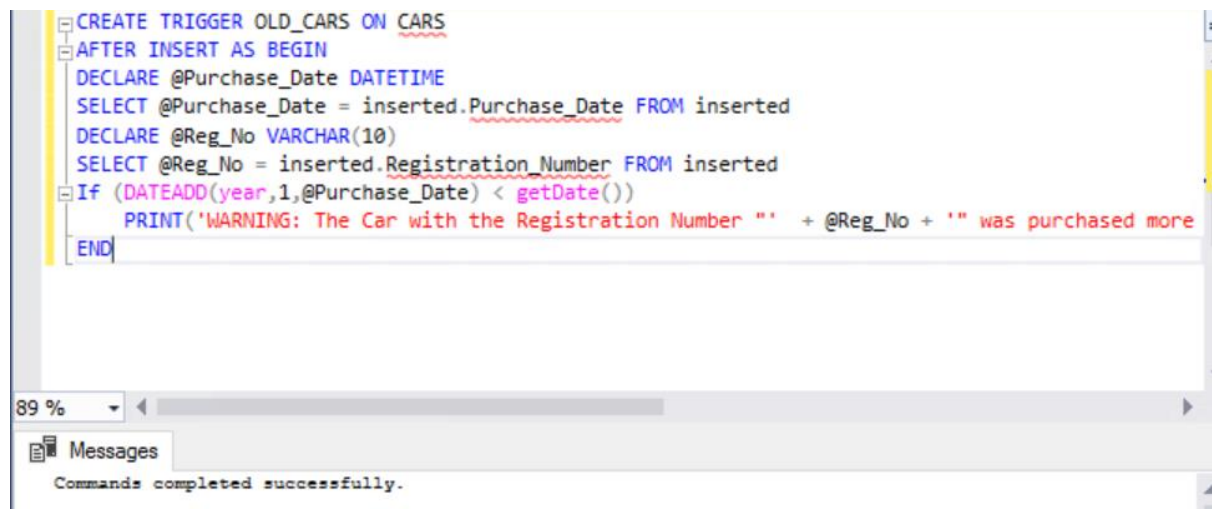
SQL Code

```

CREATE TRIGGER OLD_CARS ON CARS
AFTER INSERT AS BEGIN
DECLARE @Purchase_Date DATETIME
SELECT @Purchase_Date = inserted.Purchase_Date FROM inserted
DECLARE @Reg_No VARCHAR(10)
SELECT @Reg_No = inserted.Registration_Number FROM inserted
If (DATEADD(year,1,@Purchase_Date) < getDate())
    PRINT('WARNING: The Car with the Registration Number "' + @Reg_No + '" was
purchased more then 1 Year Ago. This Car has been added to the Database however')
END

```

Screenshot #1



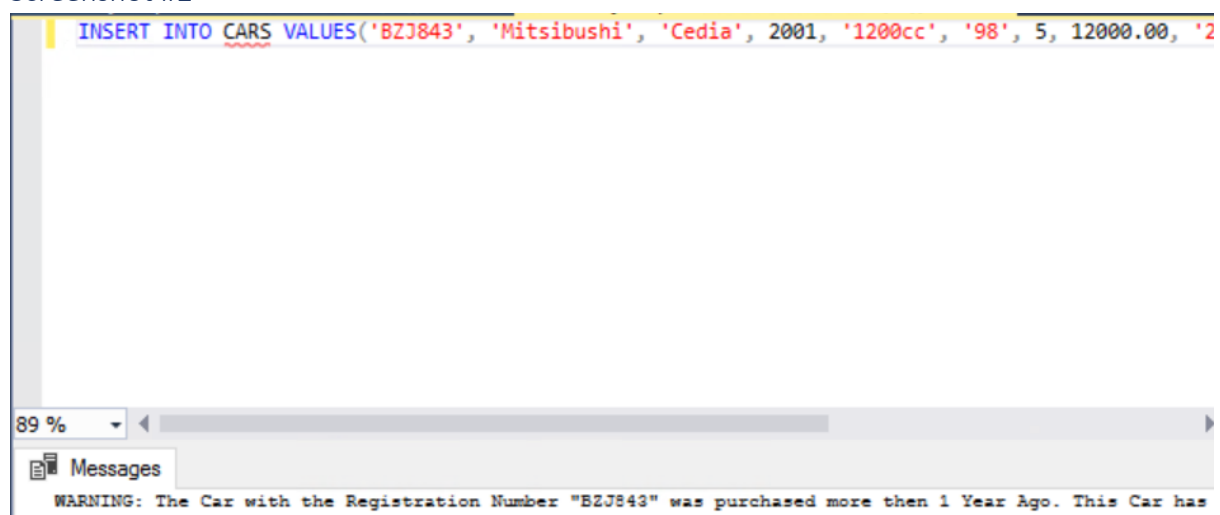
Example Data

```

INSERT INTO CARS VALUES('BZJ843', 'Mitsubishi', 'Cedia', 2001, '1200cc', '98', 5, 12000.00, '2002-01-03', 180.00, 001);

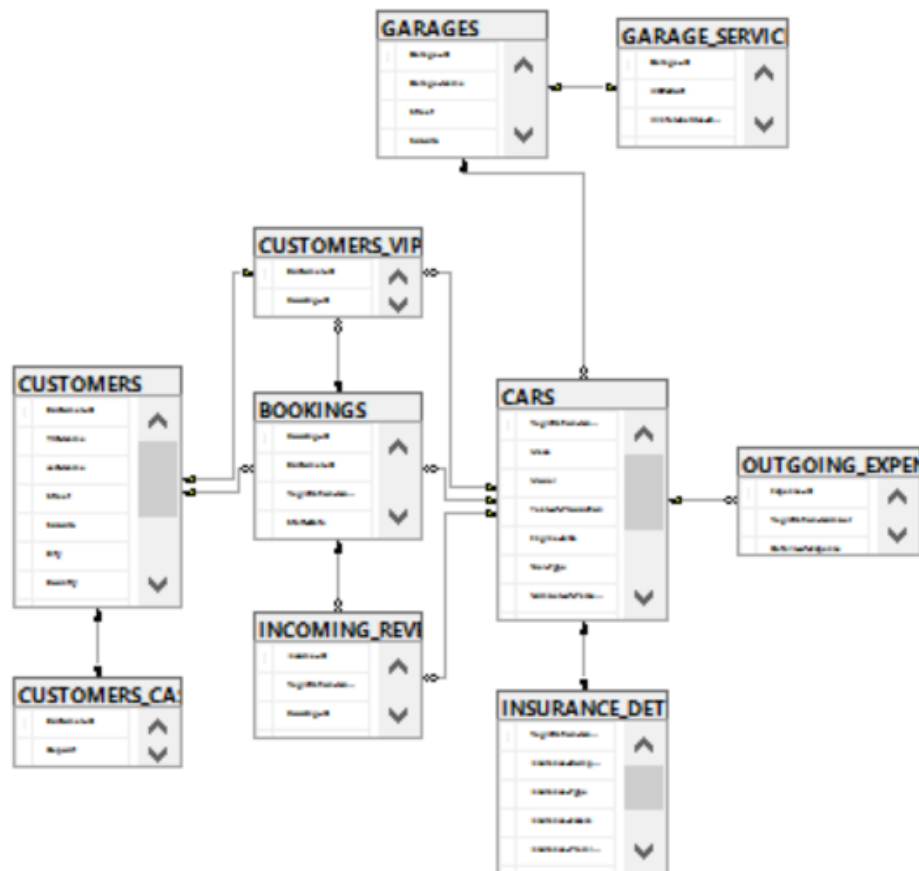
```

Screenshot #2



*This Car was brought in 2002, approximately 17 years from the current Date (10-06-2019). This Trigger performs correctly with how it shows a Warning however enters the car into the Database.*

## Finished Database Entity Relationship Diagram



## Marking Schedule

### Marking Schedule

Task	Description	Marks
Requirements	Listing Requirements.	10
Logical Design	Full ERD for the database	10
SQL	SQL creation script for the database	10
Sum		30

### *Marking Criteria per task*

0%	20%	40%	60%	80%	100%
No understanding	Little understanding of the material presented	Basic knowledge of the material presented	Adequate knowledge of the material presented	Good understanding of the knowledge presented	In depth knowledge of the subject material, gone above and beyond requirements