Student Car Rental Management database design and Implementation

Final Project for Advanced Database Management System| Group 11| ISM6218.001S21

Team Members:

Sai Harsha Bodapati
Venkata Sai Thrinesh Varma Munagapati
Amrutha Sai Prathipati
Vamsidhar Reddy Roopreddy
Pooja Veeresh Angadi

Professor:

Donald Berndt

Summary of Contents

Topic	Description	Team Member	Deadline
Design	This step entails creating a logical database design that will be used in subsequent implementations.	Sai Harsha, Amrutha Sai, Vamsidhar Reddy, Pooja Veeresh	25 Sept 2021
Implementation	Creating the objects described in the design and exporting the data to tables.	Venkata Sai Thrinesh, Pooja Veeresh, Vamsidhar Reddy	15 Oct 2021
Query Writing	Creating numerous queries to successfully retrieve data for various scenarios.	Amruth Sai, Sai Harsha, Pooja Veeresh, Venkata Sai Thrinesh	03 Nov 2021
Optimization and Other aspects	Performance tuning, the creation of stored procedures to accomplish a few tasks, and the creation of various objects such as views and sequences.	Sai Harsha, Venkata Sai Thrinesh, Amruth Sai, Vamsidhar Reddy	11 Nov 2021

Motto of the project

This project is intended for use by a car rental company that specializes in providing the students with car rental service efficiently. Students may view available cars, condition of the cars, register, contact owners, and book automobiles using this online database management system.

Description

The online car rental service is comprehensive and adaptable. It is incredibly simple to use. By automating and standardizing procedures, this online automobile rental solution aids students with proper flexibility in booking a rental car. It helps to save a lot of time, money, and effort. The tracking of vehicle activities and the overall business becomes simple and requires less paperwork. The software functions as a virtual office that is open 24 hours a day, seven days a week. It improves the management's efficiency in providing high-quality services to students. It offers software development and assistance for unique features.

In this project, we will be designing a Database for Rental Car Company. There will be many components that we will be taking care of. Some of the components are - Owner of the cars, Car types, Rental information, Customers Databases, etc.

For designing the ERR model, we have used Microsoft Visio. The database is created using MySQL workbench in local environment in MySQL. We will be having many data fields in the form of CSV file which we will be importing into the database.

The main purpose of the database is to maintain the information of all the cars that are available for rentals, the owners of each car, the rental prices, the information of each rental, available cars at every point of time.

During the course of designing, there were various points at which we were required to automatically update a one or more records on creation of records in a particular table. While designing, we have created the database taking into consideration many which we will be discussing in the next section.

Entities Identified to be Tracked

- Customer
- Vehicles
- Vehicle Owner
- Vendor
- Maintenance
- Booking
- Payment

Entities with Attributes Nested

- Customer
  - CustomerID
  - LicenseNo
  - FirstName
  - LastName
  - PhoneNumber
  - EmailID
  - City
  - State
  - Zip
  - VehicleNumber
  - OwnerID

- Vehicles
  - VehicleNumber
  - Make
  - Model
  - Year
  - Type
  - SeatingCapacity

- VehicleOwner
  - OwnerID
  - FirstName
  - LastName
  - PhoneNumber
  - EmailID
  - City
  - State
  - Zip

- Vendor
  - VendorID
  - FirstName

- LastName
- PhoneNumber
- EmailID
- City
- State
- Zip

- Maintenance
  - MaintenanceType
  - VehicleNo
  - VendorID
  - CustomerID

- Booking
  - BookingID
  - BookingStartDate
  - BookingEndDate
  - BookingCost
  - VehicleNumber

- Payment
  - PaymentReferenceNumber
  - ModeofPayment
  - BookingID

Business Rules

- One Vehicle owner can have multiple vehicles.
- One customer can book multiple vehicles depending on the availability of the vehicle.
- Owner can raise multiple requests to the vendor for maintenance.
- One customer can raise multiple complaints to the owner regarding maintenance of the vehicle.

Entity Relationship Diagram -

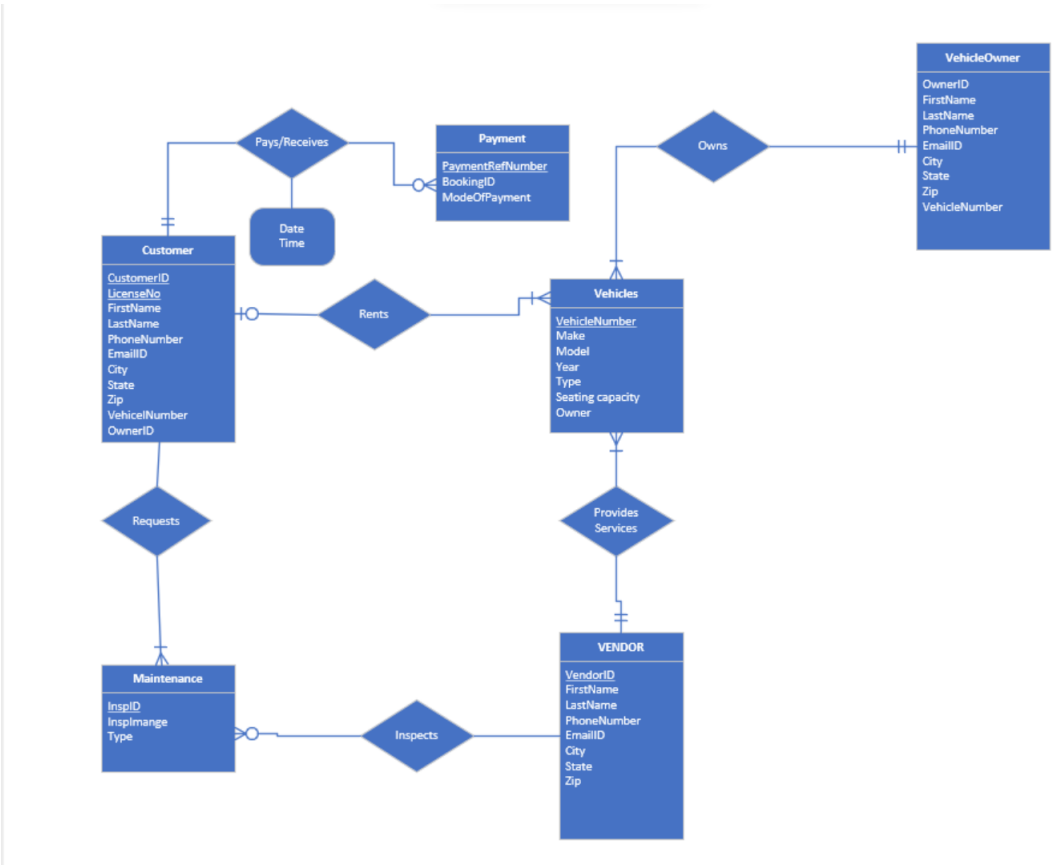


Table Views

Customer table

DESKTOP-I8DMUVO\...nt - dbo.Booking SQLQuery1.sql - DE...\

Column Name	Data Type	Allow Nulls
CustomerID	int	☐
LicenseNo	varchar(50)	☑
FirstName	varchar(50)	☑
LastName	varchar(50)	☑
PhoneNumber	varchar(50)	☑
EmailID	nvarchar(50)	☑
City	char(20)	☑
State	char(20)	☑
Zip	char(10)	☑
VehicleNumber	varchar(10)	☑
OwnerID	int	☑

This table contains all details about each customer such as CustomerID, LicenseNumber, FirstName, Last Name, MobileNumber, EmailID, address – City, State and Zip.

Vehicles table

DESKTOP-I8DMUVO\...nt - dbo.Vehicles DESKTOP-I8DMUVO...dbo.

Column Name	Data Type	Allow Nulls
VehicleNumber	varchar(10)	☐
Make	varchar(50)	☑
Model	varchar(50)	☑
Year	int	☑
Type	varchar(50)	☑
SeatingCapacity	int	☑
		☐

This table consists details of vehicles such as VehicleNumber, Make of the Vehicle, Model of the vehicle, manufacturing year of the vehicle, Type and seating capacity of the vehicle.

VehicleOwner Table

DESKTOP-I8DMUVO...dbo.VehicleOwner

Column Name	Data Type	Allow Nulls
OwnerID	int	☐
FirstName	varchar(50)	☑
LastName	varchar(50)	☑
PhoneNumber	varchar(50)	☑
EmailID	varchar(50)	☑
City	varchar(50)	☑
State	char(20)	☑
Zip	char(10)	☑
		☐

This table consists of details of the Vehicle owner such as OwnerID, FirstName, LastName, PhoneNumber, EmailID, address – City, State and Zipcode.

Vendor Table

DESKTOP-I8DMUVO\...ent - dbo.Vendor

Column Name	Data Type	Allow Nulls
VendorID	varchar(50)	☐
FirstName	varchar(50)	☑
LastName	varchar(50)	☑
PhoneNumber	varchar(50)	☑
EmailID	varchar(50)	☑
City	char(20)	☑
State	char(20)	☑
Zip	char(10)	☑
		☐

This table consists of all the Vendor details such as VendorID, First and Last Name of the Vendor, Mobile Number, EmailID, Vendor Address.

Maintenance Table

DESKTOP-I8DMUVO\...dbo.Maintenance*

Column Name	Data Type	Allow Nulls
MaintenanceType	varchar(50)	☑
VehicleNumber	varchar(10)	☑
VendorID	varchar(50)	☑
CustomerID	int	☑
		☐

This table consists of
Type of Maintenance, Vehiclenumber, VendorID and CustomerID.

Booking Table

DESKTOP-I8DMUVO\...nt - dbo.Booking

Column Name	Data Type	Allow Nulls
BookingID	varchar(10)	☐
VehicleNumber	varchar(10)	☑
BookingStartDate	date	☑
BookingEndDate	date	☑
BookingCost	money	☑
		☐

The table consists of Booking details such as BookingID, VehicleNumber, BookingStartDate, BookingEndDate and BookingCost.

Payment Table

DESKTOP-I8DMUVO\...nt - dbo.Payment

Column Name	Data Type	Allow Nulls
PaymentReferenceNumber	varchar(50)	☐
BookingID	varchar(10)	☑
ModeofPayment	varchar(50)	☑
		☐

The table consists of the payment details such as PaymentReferenceNumber, BookingId and ModeofPayment.

Data Synthesis

The data for the project was obtained using a combination of Mockaroo, an online tool, and Microsoft Excel. Some of the prominent functions that were used in Excel include,

- VLOOKUP
- INDEX
- ROWS
- RAND and
- RANDBETWEEN

Data Integrity

Data integrity refers to the data's consistency and maintenance throughout database's life cycle. In a database, data integrity can be ensured through the implementation of Integrity Constraints in a table. Integrity constraints helps us to apply business rules to the database tables. Constraints can be applied to individual columns or entire tables. The following are some of the most common constraints

- NOT NULL – It is used to Prevent a column from having a NULL value.
- PRIMARY KEY – This key Uniquely identifies each row or record in table.
- FOREIGN KEY – Uniquely identifies a column that references a PRIMARY KEY in another table.
- UNIQUE – It Prevents a column from having duplicate values.
- CHECK – Checks for values that satisfy a specific condition as defined by the user.

QUERY WRITING

1. Displaying the count of total number of vehicles booked on each day.

```
select count(BookingID) as Total_bookings,BookingStartDate as Booking_start_date
from Booking
group by BookingStartDate
```

```
select count(BookingID) as Total_bookings,BookingStartDate as Booking_start_date
from Booking
group by BookingStartDate
```

100 %

Results | Messages

	Total_bookings	Booking_start_date
1	3	2020-11-01
2	1	2020-11-02
3	1	2020-11-03
4	2	2020-11-04
5	2	2020-11-07
6	1	2020-11-09
7	1	2020-11-10
8	1	2020-11-11
9	14	2020-11-12
10	15	2020-11-13
11	13	2020-11-14
12	4	2020-11-15
13	14	2020-11-16
14	9	2020-11-17

Query executed successfully. DESKTOP-I8DMUVO\MSSQLSERVER... DESKTOP-I8DMUVO\U

2. Displaying the total booking amount made on a particular vehicle.

```
select sum(BookingCost) as Total_booking_cost,Vehiclenumber as Vehicle_number
from Booking
group by VehicleNumber
```

```
select sum(BookingCost) as Total_booking_cost,Vehiclenumber as Vehicle_number
from Booking
Group BY VehicleNumber
```

100 %

Results | Messages

	Total_booking_cost	Vehicle_number
1	462.86	A0001
2	292.58	A0002
3	110.23	A0003
4	99.57	A0004
5	178.58	A0005
6	93.756	A0007
7	172.30	A0009
8	89.00	A0010
9	82.77	A0011
10	196.99	A0012
11	91.31	A0013
12	1262.071	A0014
13	105.14	A0015
14	194.53	A0016
15	265.72	A0017
16	360.93	A0018
17	327.67	A0019

Query executed successfully. DESKTOP-I8DMUVO\MSSQLSERVER... DESKTOP-I8D

3. Obtaining maintenance information, including the vehicle's number and the vendor who serviced it.

```
select v.firstname as Vendor_First_Name, m.vehicleNumber as Vehicle_Number,
m.MaintenanceType as Maintainance_Type
from Vendor V Join Maintenance m
on v.vendorID=m.VendorID
```

```sql
select v.firstname as Vendor_First_Name, m.vehicleNumber as Vehicle_Number, m.MaintenanceType as Maintainance_Type
from Vendor V Join Maintenance m
on v.vendorID=m.VendorID
```

100 %

Results | Messages

	Vendor_First_Name	Vehicle_Number	Maintainance_Type
1	Minna	A0012	Oilchange
2	Ruthi	A0013	Oilchange
3	Georgetta	A0014	Oilchange
4	Karena	A0015	TireRotation
5	Roxy	A0016	Brakepads
6	Bellina	A0017	AirFilter
7	Jarret	A0018	TireRotation
8	Durant	A0019	Oilchange
9	Gilles	A0020	AirFilter
10	Wainwright	A0021	TireRotation
11	Judith	A0022	Battery
12	Tuck	A0023	Oilchange
13	Haze	A0024	AirFilter
14	Marthe	A0025	Battery
15	Tabbi	A0026	Brakepads
16	Anabelle	A0027	TireRotation

4. Displaying total number of customers from each state.

```sql
select Count(CustomerID) as Total_Count, State
from Customer
group by State
```

```
select Count(CustomerID) as Total_Count,State
from Customer
group by State
```

100 %

Results | Messages

	Total_Count	State
1	5	AK
2	2	AL
3	1	AR
4	3	AZ
5	6	CA
6	2	CO
7	2	CT
8	1	DC
9	8	FL
10	3	GA
11	3	IN
12	1	KY
13	1	MA
14	3	MD
15	3	MI
16	1	MN

Query executed successfully. DESKTOP-I8DMUVO\MSSQLSERVER...

5. Finding the vehicle which has been rented the most times.

```
select top 1 V.VehicleNumber, V.Make as Vehicle_make, V.Model as
Vehicle_model, count(V.VehicleNumber) as BookingCount
from Vehicles V join Booking B
on V.VehicleNumber = B.VehicleNUmber
group by V.VehicleNumber, V.Make, V.Model
order by BookingCount Desc
```

```sql
select top 1 V.VehicleNumber, V.Make as Vehicle_make, V.Model as Vehicle_model, count(V.VehicleNumber) as BookingCount
from Vehicles V join Booking B
on V.VehicleNumber = B.VehicleNUmber
group by V.VehicleNumber, V.Make, V.Model
order by BookingCount Desc
```

100 %

Results | Messages

	VehicleNumber	Vehicle_make	Vehicle_model	BookingCount
1	A0014	Honda	Accord	9

6. Finding Customer who has made maximum bookings.

```sql
Select Top 1 c.FirstName as Customer_firstname, c.LastName as Customer_lastname,
count(c.VehicleNumber) as Booking_Count
from Customer c inner join Booking B
on c.VehicleNumber=B.VehicleNumber
group by c.VehicleNumber,c.FirstName,c.LastName
Order by Booking_Count Desc
```

```sql
Select Top 1 c.FirstName as Customer_firstname, c.LastName as Customer_lastname, count(c.VehicleNumber) as Booking_Count
from Customer c inner join Booking B
on c.VehicleNumber=B.VehicleNumber
group by c.VehicleNumber,c.FirstName,c.LastName
Order by Booking_Count Desc
```

100 %

Results | Messages

	Customer_firstname	Customer_lastname	Booking_Count
1	Hugo	Scurrer	9

7. Writing a stored procedure to find the vehicles booked in a given range.

```sql
USE StudentCarRentalManagement;
GO
CREATE PROCEDURE dbo.vehiclesBookedInGivenRange
 @StartDate date,
@EndDate date
AS
SET NOCOUNT ON;
select * from Booking where BookingStartDate between @StartDate and @EndDate
GO
```

```
DECLARE @return_value int
EXEC @return_value = [dbo].[vehiclesBookedInGivenRange]
@StartDate = '2020-11-12',
@EndDate = '2020-11-20'
 GO
```

```
USE StudentCarRentalManagement;
GO
CREATE PROCEDURE dbo.vehiclesBookedInGivenRange
 @StartDate date,
@EndDate date
AS
 SET NOCOUNT ON;
select * from Booking where BookingStartDate between @StartDate and @EndDate
GO

DECLARE @return_value int
EXEC @return_value = [dbo].[vehiclesBookedInGivenRange]
@StartDate = '2020-11-12',
@EndDate = '2020-11-20'
GO
```

100 %

Results | Messages

	BookingID	VehicleNumber	BookingStartDate	BookingEndDate	BookingCost
1	EMXPM3	A0087	2020-11-12	2020-11-24	93.03
2	EMXPM363	A0012	2020-11-13	2020-11-25	196.99
3	EMXPM364	A0013	2020-11-16	2020-11-25	91.31
4	EMXPM365	A0014	2020-11-17	2020-11-22	163.02
5	EMXPM366	A0015	2020-11-16	2020-11-26	105.14
6	EMXPM367	A0016	2020-11-13	2020-11-20	194.53
7	EMXPM368	A0017	2020-11-14	2020-11-21	122.51
8	EMXPM369	A0018	2020-11-17	2020-11-26	69.61
9	EMXPM370	A0019	2020-11-14	2020-11-25	150.67

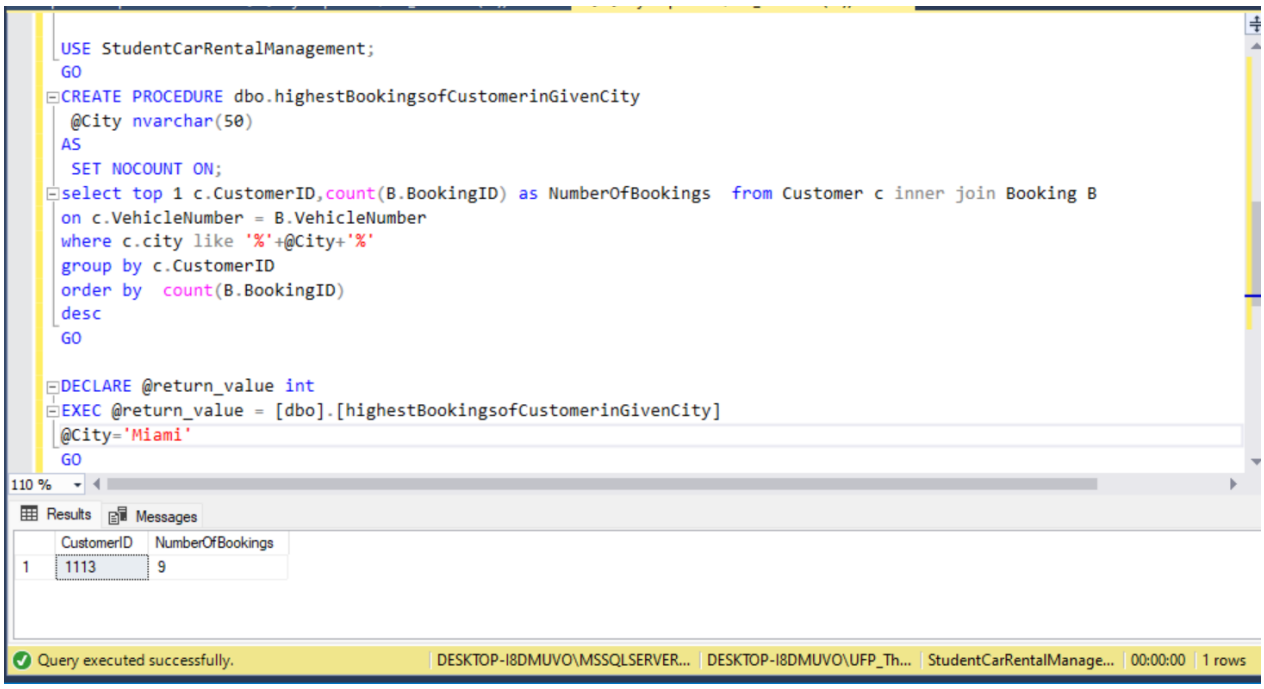
Query executed successfully. | DESKTOP-I8DMUVO\MSSQLSERVER... | DESKTOP-I8DMU

8. Writing a procedure to find highest bookings made by a customer in a given city.

```
USE StudentCarRentalManagement;
GO
CREATE PROCEDURE dbo.highestBookingsofCustomerinGivenCity
@City nvarchar(50)
AS
 SET NOCOUNT ON;
select top 1 c.CustomerID,count(B.BookingID) as NumberOfBookings  from Customer c inner join
Booking B
on c.VehicleNumber = B.VehicleNumber
where c.city like '%'+@City+'%'
```

```
group by c.CustomerID
order by  count(B.BookingID)
desc
 GO


DECLARE @return_value int
EXEC @return_value = [dbo].[highestBookingsofCustomerinGivenCity]
@City='Miami'
GO
```

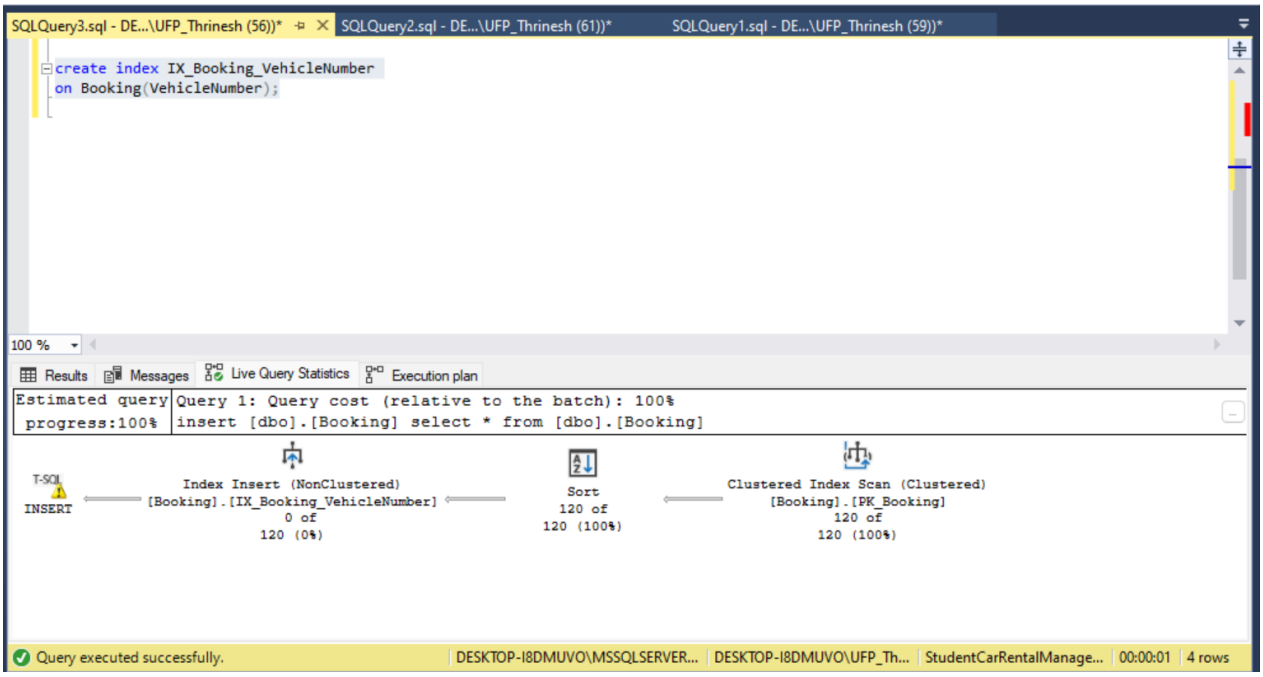


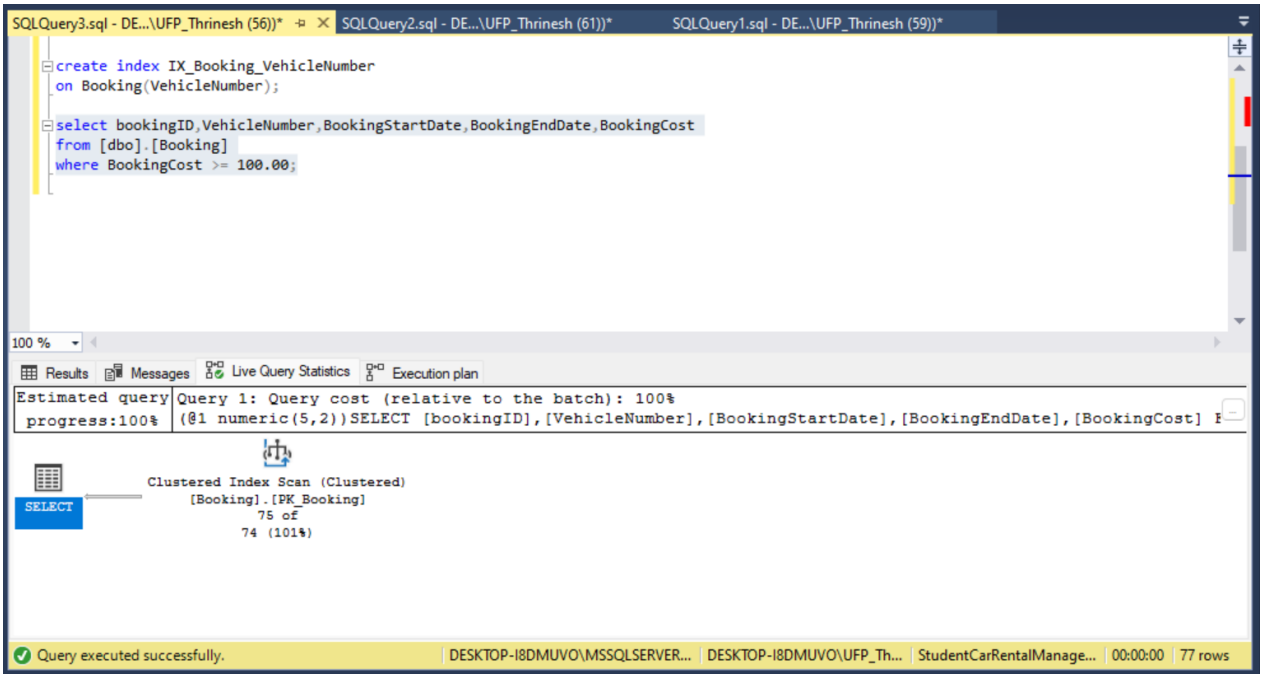
Performance Tuning

Index:

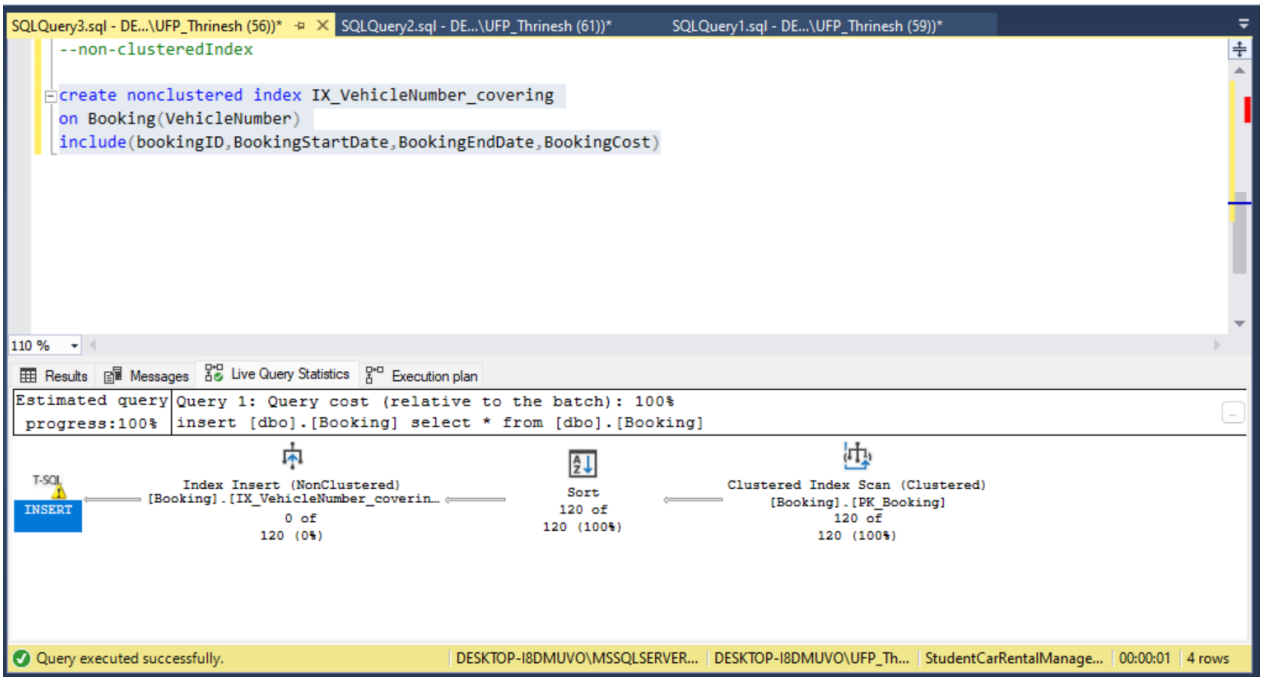
An index is used to improve query performance in general. This is accomplished through indexing, which reduces the number of data pages that must be viewed or scanned each time a query is performed. By default, the primary key creates a clustered index when we establish an index. A clustered index in SQL Server specifies the physical order of data in a table. Each table can only have one clustered index.

Covering Index using include:



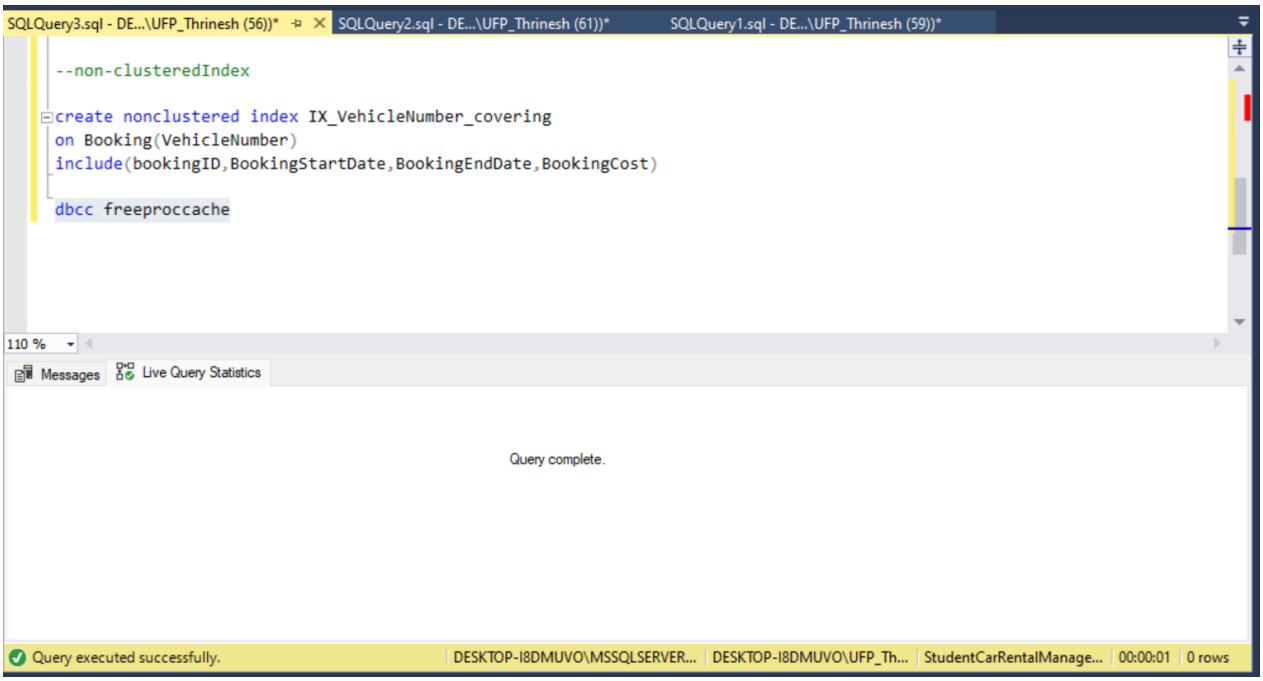


Non-Clusterd Index :

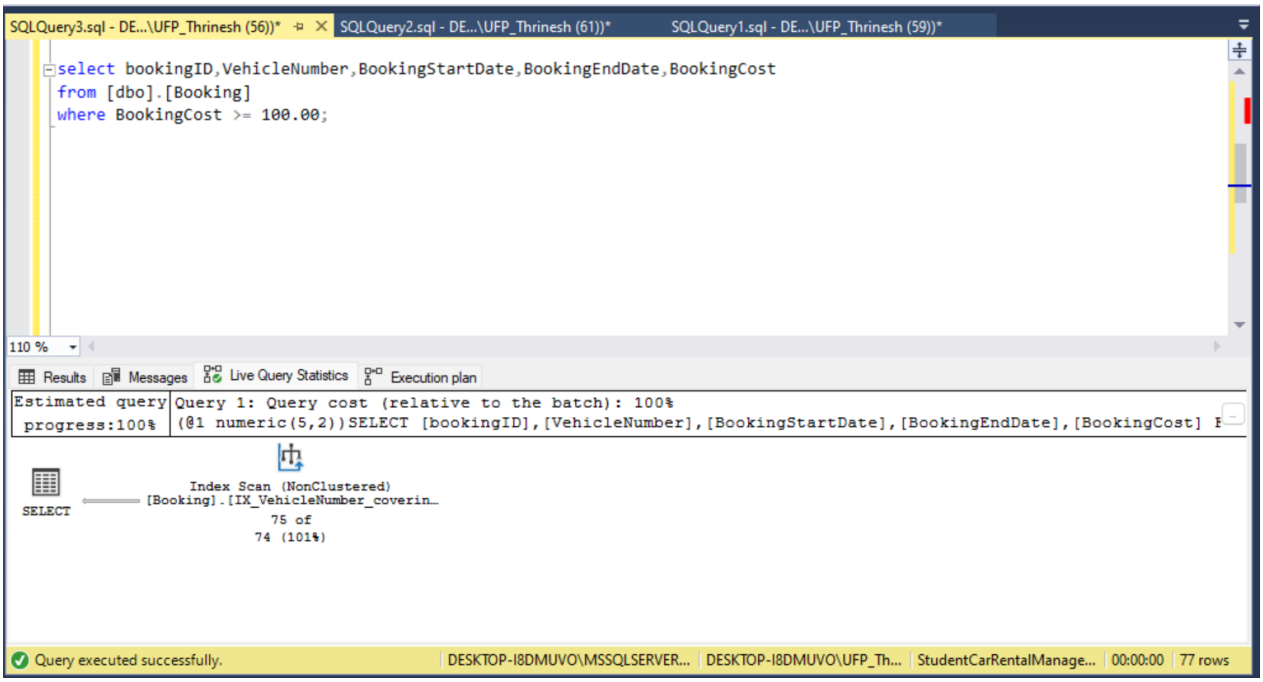


Clear Cache:

here we used DBCC to clear the cache.



Running the query again (Force use of the created index)



Comparison Performance: