**Advanced Database Management Systems**

**Final Project**

**Vehicle Rental Management database design**

**and Implementation**



Team Members: Professor:

Penchala Akshay Kumar Kandagaddala Donald Berndt

Shashank Rao Gujja

Venkata Sai Gagan Deep Alusuri

Srikumar Dundigalla

Vinay Kumar Moluguri

Naveen Yeluru

Rohith Rasamalla

Meghana Rudraraju

Lokesh Reddy Pappu

Nagarjuna Gummalla

# Summary of Contents

|  |  |  |
| --- | --- | --- |
| **S No.** | **Title** | **Page No.** |
| 1. | Motto of the Project | **3** |
| 2. | Description | **4** |
| 3. | Entities of the Database | **4** |
| 4. | Entities with Attributes Grouped | **5** |
| 5. | Business Rules | **7** |
| 6. | Entity Relationship Documentation (ERD) | **7** |
| 7. | Table Views | **8** |
| 8. | Data Synthesis | **11** |
| 9. | Data Integrity | **11** |
| 10. | Queries | **12** |
| 11. | Performance Tuning | **20** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Topic** | **Description** | **Team Member** | **Deadline** |
| **Design** | This step entails creating a logical database design that will be used in subsequent implementations. | Akshay, Gagan and Sri Kumar | 20 Nov 2022 |
| **Implementation** | Creating the objects described in the design and exporting the data to tables. | Akshay, Shashank, Vinay | 24 Nov 2022 |
| **Query Writing** | Creating numerous queries to successfully retrieve data for various scenarios. | Gagan, SriKumar, Lokesh, Nagarjuna | 2 Dec 2022 |
| **Optimization and Other aspects**  **Data Visualization** | Performance tuning, the creation of stored procedures to accomplish a few tasks, and the creation of various objects such as views and sequences.  Visualized and represented the data from different tables. | Rohith, Naveen , Meghana  Lokesh, Naveen, Shashank | 11 Dec 2022  11 Dec 2022 |

**Motto of the project**

This project is meant to be used by a vehicle rental business that specializes in effectively offering

vehicle rental services to students. Using this online database management system, people can view

available vehicles, check the condition of the vehicles, register, get in touch with owners, and reserve

them.

# Description

The online car rental service is thorough and versatile. It is extremely easy to use. This online vehicle rental solution helps students with the right flexibility when booking a rental car by automating and standardizing processes. Significant amounts of time, money, and effort are saved. It becomes easier and less paper-intensive to track vehicle activities and the business as a whole. The program performs as a virtual office that is accessible every day of the week, around-the-clock. It increases management's effectiveness in offering students top-notch services. It provides software development and help with special features.

We will be creating a database for a rental car company as part of this project. We will be taking care of a lot of different components. Owners of the cars, car types, rental information, customer databases, etc. are a few of the components.

Utilizing Microsoft Visio, we created the ERR model. Using MySQL workbench in a local environment, the database is created. We will be importing a large number of data fields in the form of a CSV file into the database.

The database's primary function is to maintain data on all the automobiles that are available for rental, including information on each vehicle's owner, rental rates, details about each rental, and vehicles that are available at any given moment.

# There were several times while designing where we needed to automatically update one or more records upon the creation of records in a specific table. We designed the database while taking a number of factors into account, which we will go over in the following section.

**The Entities in the database are as follows:**

• Customer • Maintenance • Payment • Vehicles • Booking

• Vehicle Owner • Vendor

# Entities with Attributes Grouped

* 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.
* The vendor for maintenance may receive multiple requests from the owner.
* A single customer can make numerous vehicle maintenance-related complaints to the owner.

**Entity Relationship Diagram**

Diagram

Description automatically generated

## Table Views

**Customer table**

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

**Vehicles table**

|  |  |
| --- | --- |
|  | 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**

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

**Vendor Table**

|  |  |
| --- | --- |
|  | 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**

**Graphical user interface, application

Description automatically generated**

This table consists of

Type of Maintenance, Vehiclenumber, VendorID and CustomerID.

**Booking Table**

Graphical user interface, application

Description automatically generated

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

**Payment Table**

Graphical user interface

Description automatically generated

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

# Data Synthesis

Microsoft Excel was used in conjunction with the online tool Mockaroo to collect the data for the project.

Excel has several well-known functions, some of which include

* INDEX
* ROWS
* VLOOKUP
* RANDB
* RANDBETWEEN

# Data Integrity

Data consistency and maintenance during the course of a database's life cycle are referred to as data integrity. Integrity constraints can be applied to a table in a database to ensure data integrity. Applying business rules to the database tables is made easier by integrity constraints. Individual columns or whole tables may be subject to constraints. The most typical limitations include the following.

* NOT NULL – The NOT NULL constraint **enforces a column to** NOT accept NULL **values**.
* PRIMARY KEY – This key accepts unique values for each row.
* FOREIGN KEY – A FOREIGN KEY is a field(or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
* UNIQUE – The UNIQUE constraint ensures that all values in a column are different.
* CHECK – a SQL constraint that allows database users to enter only those values which fulfill the

specified 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 bookingdata

group by bookingstartdate

Graphical user interface, text, application

Description automatically generated

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

select sum(BookingCost) as Total\_booking\_cost,Vehiclenumber as Vehicle\_number from BookingData

group by VehicleNumber

Graphical user interface, text, application

Description automatically generated

1. 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 vendordata V Join maintenancedata m on v.vendorID=m.VendorID

Graphical user interface, text, application

Description automatically generated

1. Displaying total number of customers from each state.

select Count(CustomerID) as Total\_Count, State from customerdata

group by State

Graphical user interface, text, application

Description automatically generated

1. 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

Graphical user interface, text, application

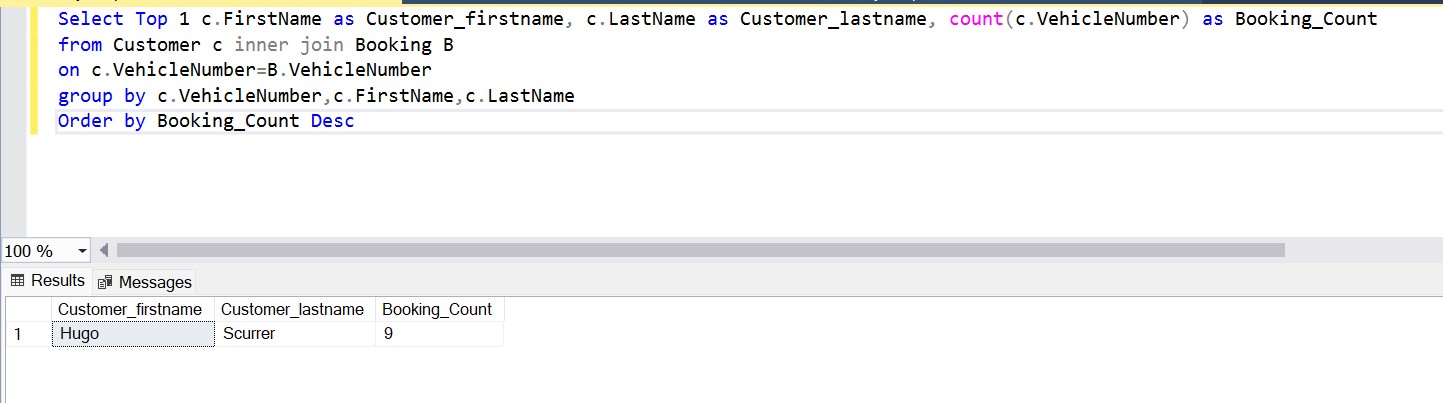
Description automatically generated

1. Finding Customer who has made maximum bookings.

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



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

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

Graphical user interface, text, application

Description automatically generated

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

USE VehicleManagementSystem; 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=’Dallas'

GO

Graphical user interface, text, application, email

Description automatically generated

# Performance Tuning

**Index:**

In general, query performance is enhanced with indexes. This is made possible via indexing,

which lowers the quantity of data pages that need to be read or scanned each time a query is run.

When we construct an index, the main key by default produces a clustered index. The physical

order of the data in a table is specified by a clustered index in SQL Server. There can only be

one clustered index per table.

**Covering Index using include:**

**Graphical user interface, text, application

Description automatically generated**

Graphical user interface, text, application, email

Description automatically generated

**Non-Clusterd Index :**

**Graphical user interface, text, application

Description automatically generated**

**Clear Cache:**

here we used DBCC to clear the cache.

Graphical user interface, text

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

**Comparison Performance:**

**Graphical user interface, application

Description automatically generated**

# THANK YOU