IT 440 – Spring 2018

Minnesota State University, Mankato

GROUP 7 – CAR RENTAL DATABASE

Nhung Do – Khuzaima Shahid – Ezana Tesfaye – Saiteja Thota

Version 1.0

April 22, 2018

TABLE OF CONTENT

[DATABASE 2](#_Toc511700133)

[Entity Relationship Diagram 2](#_Toc511700134)

[Assumptions 2](#_Toc511700135)

[Business Rules 2](#_Toc511700136)

[Normalization steps 3](#_Toc511700137)

[Unnormalized Form (UNF) 3](#_Toc511700138)

[First Normal Form (1NF) 3](#_Toc511700139)

[Second Normal Form (2NF) 4](#_Toc511700140)

[Third Normal Form (3NF) 5](#_Toc511700141)

[SQL SERVER 5](#_Toc511700142)

[Inner Join Statement 5](#_Toc511700143)

[Subquery 6](#_Toc511700144)

[CTE 6](#_Toc511700145)

[Pivot 7](#_Toc511700146)

[Clustered Index 8](#_Toc511700147)

[MYSQL 8](#_Toc511700148)

[WEB FORM 10](#_Toc511700149)

[Login Page 10](#_Toc511700150)

[Authentication 10](#_Toc511700151)

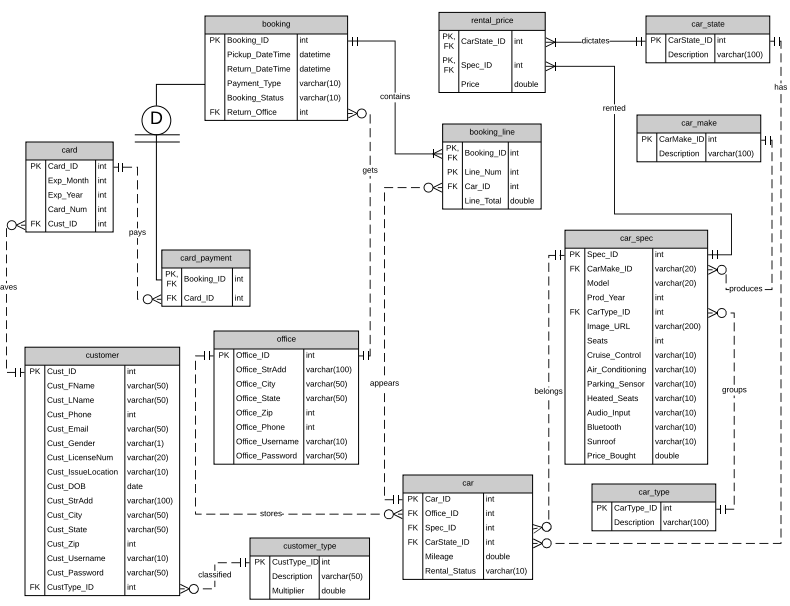
[About Page 11](#_Toc511700152)

[Dashboard 11](#_Toc511700153)

[Logout Page 12](#_Toc511700154)

# DATABASE

## Entity Relationship Diagram



## Assumptions

The business only accepts card payments.

The rented car can be returned at another office different from the original office.

The customer type determines what multiplier will be applied to the total charge. The multiplier includes discounts, insurance, fees if any.

## Business Rules

One booking must have at least one booking line.

One car may appear on many booking lines. One booking line must contain only one car.

One card may pay many card payments. One card payment must be paid by only one card.

One car make may produce many car specs. One car spec must be produced by only one car make.

One car state may have many cars in it. One car must belong to only one car state.

One car type may have many car specs. One car spec must belong to only one car type.

One customer may have many cards on profile. One card must belong to only one customer.

One customer type may include many customers. One customer must be categorized into only one customer type.

One office may store many cars. One car must be present at only one office.

One office may appear on many bookings. One booking must have only one office to return the car to.

One car spec may belong to many cars. One car must have only one car spec.

One car state may dictate many rental prices. One rental price must be dictated by only one car state.

One car spec may dictate many rental prices. One rental price must be dictated by only one car spec.

## Normalization steps

### Unnormalized Form (UNF)

In the unnormalized form, all information regarding car rentals is in one big table including the following columns:

Booking\_ID  
Line\_Num  
Car\_Make  
Model   
Prod\_Year  
Car\_Type  
Car\_Spec  
Image\_URL  
Seats  
Cruise\_Control  
Air\_Conditioning  
Parking\_Sensor  
Heated\_Seats  
Audio\_Input  
Bluetooth  
Sunroof  
Price\_Bought  
Office\_StrAdd  
Office\_City  
Office\_State  
Office\_Zip  
Office\_Phone  
Office\_Username  
Office\_Password  
Car\_State  
Mileage  
Line\_Total  
Pickup\_DateTime  
Return\_DateTime  
Payment\_Type  
Booking\_Status  
Return\_Office  
Cust\_Fname  
Cust\_Lname  
Cust\_Phone  
Cust\_Email  
Cust\_Gender  
Cust\_LicenseNum  
Cust\_IssueLocation  
Cust\_DOB  
Cust\_StrAdd  
Cust\_City  
Cust\_State  
Cust\_Zip  
Cust\_Username  
Cust\_Password  
Cust\_Type  
Multiplier

For each booking record there are details about the car to be rented, the office the car is at, the customer that is renting the car, and the booking details.

### First Normal Form (1NF)

The first step in normalizing a relation is to remove the repeating groups. The unnormalized relation is broken into 4 separate relations:

1. **Relation beween booking and booking details**

Booking\_ID Line\_Num Line\_Total Pickup\_DateTime Return\_DateTime Payment\_Type Booking\_Status Return\_Office Car\_ID Card\_ID

This relation contains all the attributes that were not repeating.

1. **Relation between booking and car**

Booking\_ID Line\_Num Car\_ID Car\_Make Model Prod\_Year Car\_Type Car\_Spec Image\_URL Seats Cruise\_Control Air\_Conditioning Parking\_Sensor Heated\_Seats Audio\_Input Bluetooth Sunroof Price\_Bought Mileage Car\_State

In this relation, one must use a concatenated key to access the repeating data regarding the car rented.

1. **Relation between booking and customer**

Booking\_ID Line\_Num Card\_ID Cust\_ID Cust\_FName Cust\_LName Cust\_Phone Cust\_Email Cust\_Gender Cust\_LicenseNum Cust\_IssueLocation Cust\_DOB Cust\_StrAdd Cust\_City Cust\_State Cust\_Zip Cust\_Username Cust\_Password Cust\_Type Multiplier

In this relation, one must use a concatenated key to access the repeating data regarding the customer.

1. **Relation between booking and office**

Booking\_ID Line\_Num Car\_ID Office\_ID Office\_StrAdd Office\_City Office\_State Office\_Zip Office\_Phone Office\_Username Office\_Password

In this relation, one must use a concatenated key to access the repeating data regarding the office.

### Second Normal Form (2NF)

In the second normal form, all the attributes will be functionally dependent on the primary key. Therefore, the next step is to remove all the partially dependent attributes and place them in another relation.

Booking\_ID Pickup\_DateTime Return\_DateTime Payment\_Type Booking\_Status Return\_Office Card\_ID

Booking\_ID Line\_Num Line\_Total Car\_ID

Car\_ID Car\_Make Model Prod\_Year Car\_Type Car\_Spec Image\_URL Seats Cruise\_Control Air\_Conditioning Parking\_Sensor Heated\_Seats Audio\_Input Bluetooth Sunroof Price\_Bought Mileage Car\_State

Cust\_ID Cust\_FName Cust\_LName Cust\_Phone Cust\_Email Cust\_Gender Cust\_LicenseNum Cust\_IssueLocation Cust\_DOB Cust\_StrAdd Cust\_City Cust\_State Cust\_Zip Cust\_Username Cust\_Password Cust\_Type Multiplier

Office\_ID Office\_StrAdd Office\_City Office\_State Office\_Zip Office\_Phone Office\_Username Office\_Password

### Third Normal Form (3NF)

A normalized relation is in the third normal form if all the nonkey attributes are fully functionally dependent on the primary key and there are no transitive (nonkey) dependencies.

Booking\_ID Pickup\_DateTime Return\_DateTime Payment\_Type Booking\_Status Return\_Office

Booking\_ID Line\_Num Car\_ID Line\_Total

Car\_ID Spec\_ID Office\_ID CarState\_ID Mileage

Spec\_ID CarMake\_ID Model Prod\_Year CarType\_ID Image\_URL Seats Cruise\_Control Air\_Conditioning Parking\_Sensor Heated\_Seats Audio\_Input Bluetooth Sunroof Price\_Bought

CarMake\_ID Description

CarState\_ID Description

CarType\_ID Description

Card\_ID Exp\_Month Exp\_Year Card\_Num Cust\_ID

Booking\_ID Card\_ID

Cust\_ID Cust\_FName Cust\_LName Cust\_Phone Cust\_Email Cust\_Gender Cust\_LicenseNum Cust\_IssueLocation Cust\_DOB Cust\_StrAdd Cust\_City Cust\_State Cust\_Zip Cust\_Username Cust\_Password CustType\_ID

CustType\_ID Description Multiplier

Office\_ID Office\_StrAdd Office\_City Office\_State Office\_Zip Office\_Phone Office\_Username Office\_Password

CarState\_ID Spec\_ID Price

# SQL SERVER

## Inner Join Statement

--list revenue collected by each office

SELECT c.Office\_ID, SUM(bl.Line\_Total) AS Revenue

FROM car\_rental.booking\_line AS bl

INNER JOIN car\_rental.booking AS b

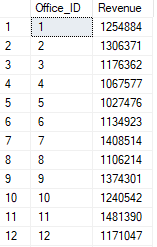
ON bl.Booking\_ID = b.Booking\_ID

INNER JOIN car\_rental.car AS c

ON c.Car\_ID = bl.Car\_ID

GROUP BY c.Office\_ID

ORDER BY Office\_ID;  
GO



## Subquery

--list all cars along with their IDs, types, models and production years

SELECT Car\_ID AS 'Car ID', CarType AS 'Type', CarMake AS 'Make', Model, Prod\_Year AS 'Production Year'

FROM (

SELECT c.Car\_ID, t.Description AS CarType, m.Description AS CarMake, s.Model, s.Prod\_Year

FROM car\_rental.car AS c

INNER JOIN car\_rental.car\_spec AS s

ON c.Spec\_ID = s.Spec\_ID

INNER JOIN car\_rental.car\_make AS m

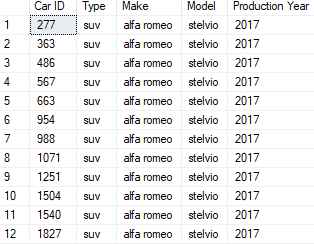
ON s.CarMake\_ID = m.CarMake\_ID

INNER JOIN car\_rental.car\_type AS t

ON s.CarType\_ID = t.CarType\_ID

) AS p

GO



## CTE

--list car models that are rented for more than 300 per day

;WITH Car\_CTE(Make, Model, Price) AS

(

SELECT m.Description AS 'Make', sp.Model, p.Price

FROM car\_rental.rental\_price AS p

INNER JOIN car\_rental.car\_spec AS sp

ON sp.Spec\_ID = p.Spec\_ID

INNER JOIN car\_rental.car\_make AS m

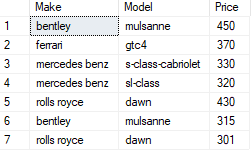
ON sp.CarMake\_ID = m.CarMake\_ID

)

SELECT \*

FROM Car\_CTE

WHERE Price > 300;  
GO



## Pivot

--pivot to list number of orders at each branch in April, May, June

SELECT Office\_ID, [4] AS April, [5] AS May, [6] AS June

FROM (

SELECT c.Office\_ID, b.Booking\_ID, DATEPART(M, b.Pickup\_DateTime) AS PickupMonth

FROM car\_rental.booking AS b

INNER JOIN car\_rental.booking\_line AS bl

ON b.Booking\_ID = bl.Booking\_ID

INNER JOIN car\_rental.car AS c

ON bl.Car\_ID = c.Car\_ID

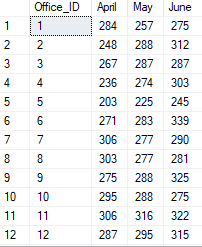
) AS piv

PIVOT (

COUNT(Booking\_ID) FOR PickupMonth IN([4], [5], [6])

) AS p

ORDER BY p.Office\_ID;  
GO



## Clustered Index

--create an index on customer table using the first name column

CREATE CLUSTERED INDEX CustomerIndex

ON car\_rental.customer (Cust\_FName DESC);

--list customers with first name Dakota along with their birthdate and customer type

SELECT Cust\_FName, Cust\_LName, Cust\_DOB, Description AS 'Type'

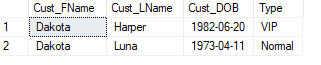
FROM car\_rental.customer AS c WITH (INDEX(CustIndex))

INNER JOIN car\_rental.customer\_type AS t

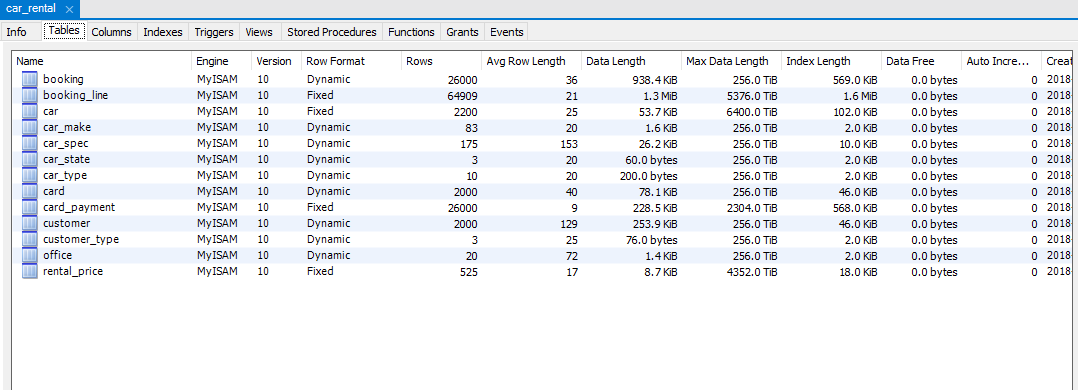
ON c.CustType\_ID = t.CustType\_ID

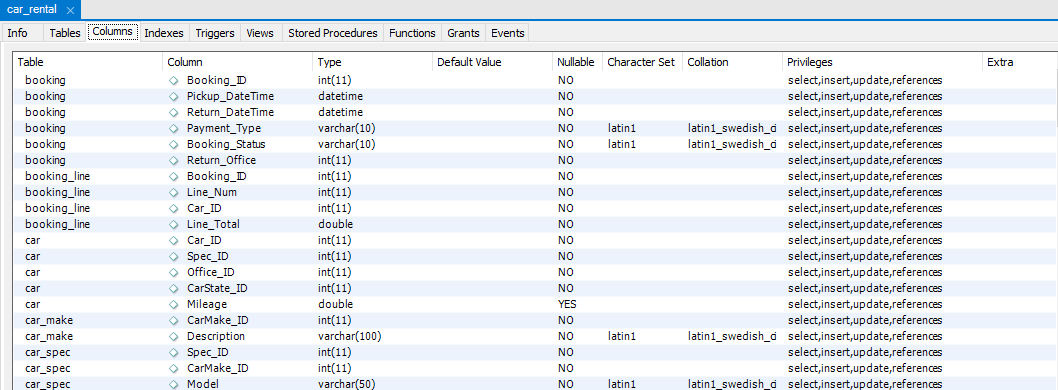
WHERE Cust\_FName = 'Dakota';

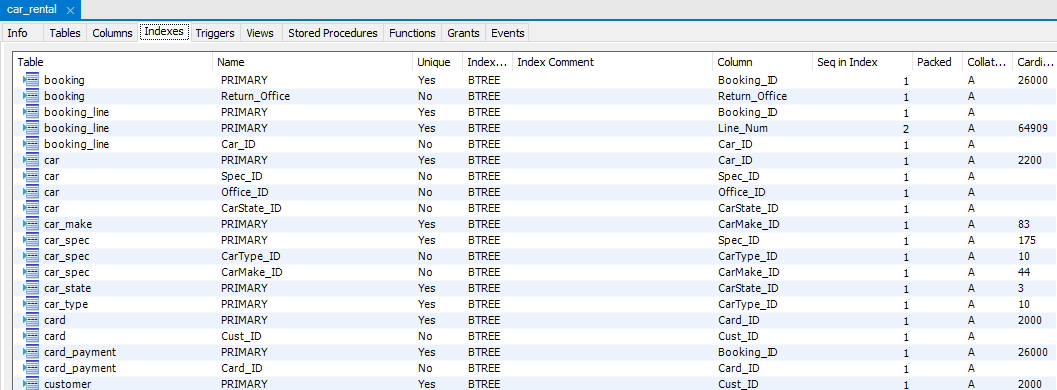
GO

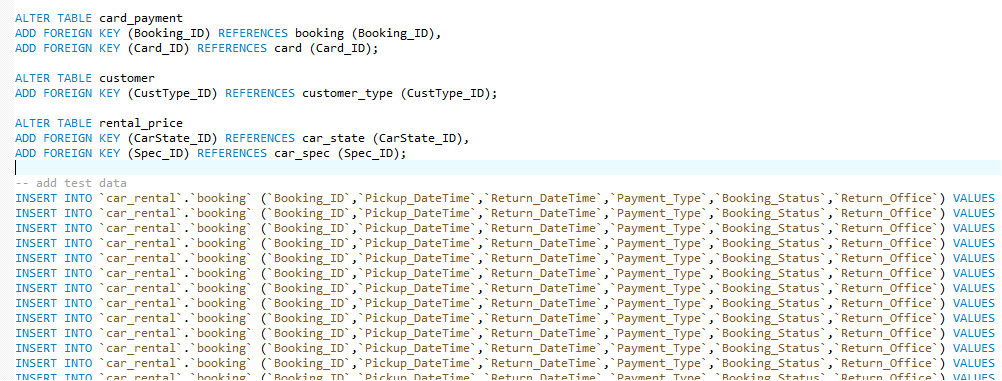


# MYSQL



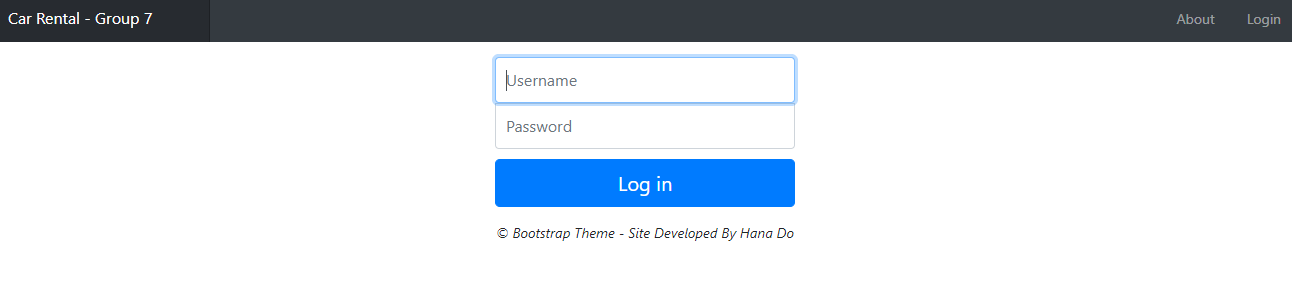




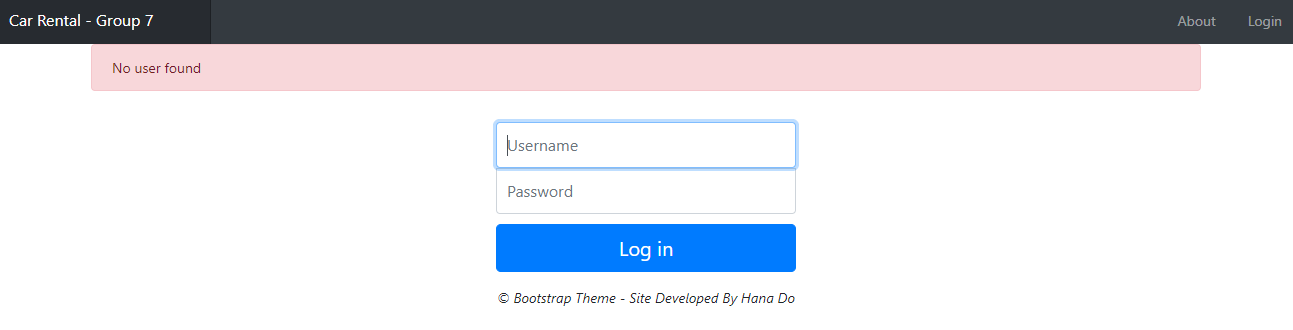


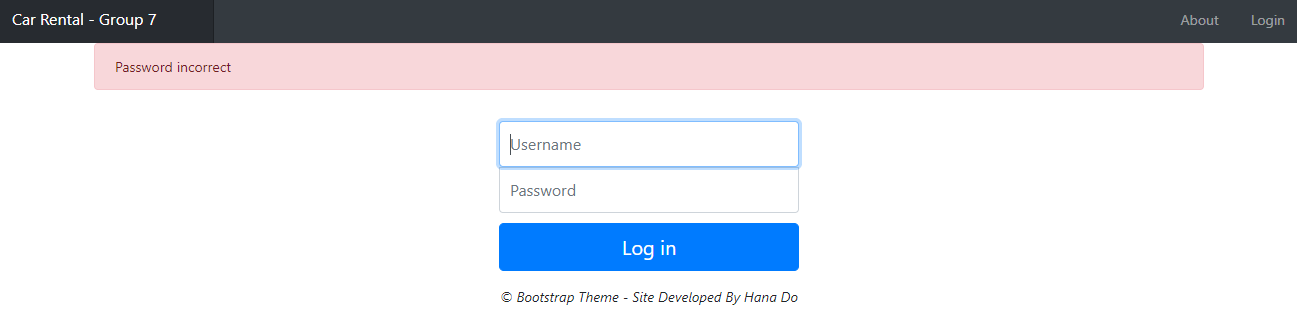
# WEB FORM

## Login Page

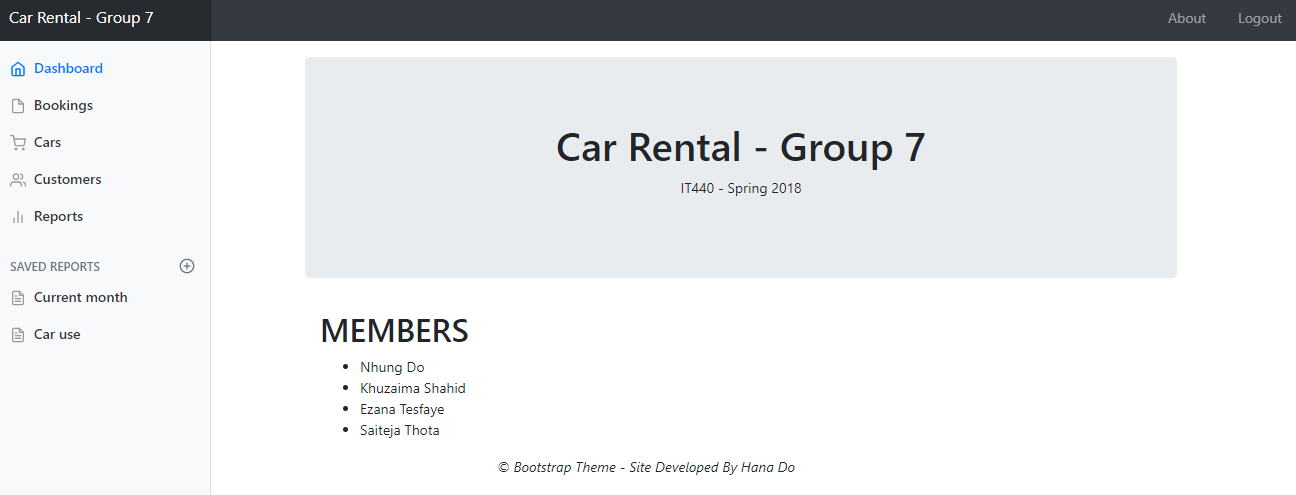


## Authentication

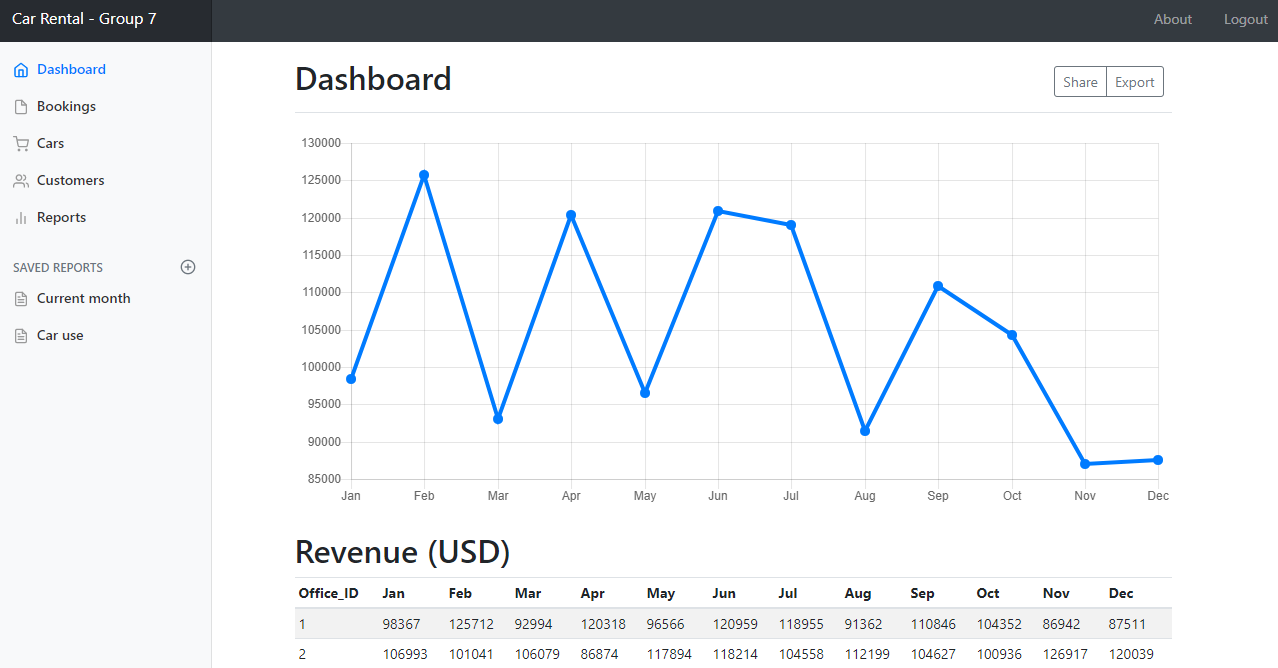




## About Page



## Dashboard



## Logout Page

