**Business Rules**

1. A customer can only rent one vehicle at a time.
2. One customer can rent a vehicle more than once, if he/she has returned back earlier rented vehicle. They will have a unique rental ID each time.
3. A customer can rent no vehicle or many vehicles.
4. A vehicle can be rented multiple times.
5. A vehicle can only be present at one location at a time.
6. One location can have no vehicles at all or many vehicles.
7. A vehicle can be rented or not at all.
8. A valid license is required to rent a vehicle.
9. The customer picks up and drops off the vehicle on their specified dates.
10. While renting a car customer should define pickup and drop-off location in advance.
11. The customer must have insurance to rent a car.
12. The customer has a choice to use the rental company’s insurance or their own.
13. Insurance will come in 3 types-none, half, full.
14. Before renting a car , customer should pay deposit amount only through credit card ,that will act as hold for charging incase of scenarios like penalties , fines or any damages. Deposit amount vary for different vehicle. Expensive vehicle will have high deposit amount.
15. Deposit amount will be returned back once the car is returned .
16. Payment methods can be debit, credit, or cash for the total rental charge .
17. The entire payment is made up front.
18. The payment is made after returning the vehicle.
19. The final drop-off is always at the specified rental location.
20. The rental company locations are limited to the U.S.
21. Vehicle rental rate will vary depending on vehicle type.
22. When renting a vehicle, it will have a full tank to start with.
23. Customer will be charged extra depending whether the vehicle is returned with empty vs half tank vs full. $25 for half full, $50 for empty.
24. Total rental amount is calculated by (number\_of\_days\*per\_day\_rate)+insurance+tank\_level\_fine.
25. Tax rates will vary based on the state where the vehicle was rented from.
26. A unique invoice will be created for a customer with the appropriate payment information.
27. Revenue will be added to the account of the location from where the vehicle was rented.

**Entity Relationships:**

Customer and Rental: 0 to many

Vehicle and Rental: 0 to many

Customer and License: 1 to 1

Customer and Card\_Detail: 1 to many

Card\_Detail and Deposit :1 to many

Location and Rental: 0 to many

Location and Vehicle: 0 to many

Location and Tax: 1 to many

Rental and Deposit: 1 to 1

Rental and Invoice: 1 to many

Rental and Insurance: 1 to 1

**Normalization Steps**

**Unf:**

Customer\_ID, Customer\_FName, Customer\_LName, Customer\_DOB, Customer\_Street, Customer\_City, Customer\_State, Customer\_Zip\_Code, Customer\_Phone, Customer\_Email, License\_Number, License\_Validity, Card\_ID, Card\_Number, Card\_Type, Card\_Expiration\_Date, Vehicle\_ID, Vehicle\_Type, Vehicle\_Model, Vehicle\_Year, Vehicle\_Current\_Mileage, Vehicle\_Rental\_Status, Vehicle\_Number\_of\_Seats, Vehicle\_Rental\_Status, Vehicle\_Rental\_Rate, Vehicle\_Color, Location\_ID, Location\_Street, Location\_City, Location\_State, Location\_Zip\_Code, Location\_Phone, Location\_Email, Rental\_ID, Rental\_Pickup\_Date, Rental\_Dropoff\_Date, Rental\_Pickup\_Time, Rental\_Dropoff\_Time, Pickup\_Mileage, Dropoff\_Mileage, Rental\_Car\_ReturnTankLevel, Rental\_Deposit\_Paid\_Status, Rental\_Tax, Rental\_Days, Rental\_charge, Insurance\_ID, Insurance\_By, Insurance\_Type, Insurance\_Perday\_Rate, Invoice\_Number, Total\_Charge\_Incx\_Tax, Debit\_Card\_Total, Credit\_Card\_Total, Cash\_Total, Payment\_Status, Payment\_Date, Deposit\_ID, Deposit\_Amount , Tax\_ID, Tax\_Rate

**1nf:**

Customer\_ID, Customer\_FName, Customer\_LName, Customer\_DOB, Customer\_Street, Customer\_City, Customer\_State, Customer\_Zip\_Code, Customer\_Phone, Customer\_Email, Vehicle\_ID, Vehicle\_Type, Vehicle\_Model, Vehicle\_Year, Vehicle\_Current\_Mileage, Vehicle\_Number\_of\_Seats, Vehicle\_Rental\_Status, Vehicle\_Rental\_Rate, Vehicle\_Color, Location\_ID, Location\_Street, Location\_City, Location\_State, Location\_Zip\_Code, Location\_Phone, Location\_Email, Rental\_ID, Rental\_Pickup\_Date, Rental\_Dropoff\_Date, Rental\_Pickup\_Time, Rental\_Dropoff\_Time, Pickup\_Mileage, Dropoff\_Mileage, Rental\_Car\_ReturnTankLevel, Rental\_Deposit\_Paid\_Status, Rental\_Tax, Rental\_Days, Rental\_charge, Insurance\_ID, Insurance\_By, Insurance\_Type, Insurance\_Perday\_Rate ,Invoice\_Number, Total\_Charge\_Incx\_Tax, Debit\_Card\_Total, Credit\_Card\_Total, Cash\_Total, Payment\_Status, Payment\_Date, Deposit\_ID, Deposit\_Amount, License\_Number, License\_Validity, Card\_ID, Card\_Number, Card\_Type, Card\_Expiration\_Date)

**2nf:**

The 2nf table meets the 1nf requirements. There are no partial dependencies.

**Customer** (Customer\_ID, Customer\_FName, Customer\_LName, Customer\_DOB, Customer\_Street, Customer\_City, Customer\_State, Customer\_Zip\_Code, Customer\_Phone, Customer\_Email)

**Vehicles (**Vehicle\_ID, Vehicle\_Type, Vehicle\_Model, Vehicle\_Year, Vehicle\_Current\_Mileage, Vehicle\_Number\_of\_Seats, Vehicle\_Rental\_Status, Vehicle\_Rental\_Rate, Vehicle\_Color)

**Location (**Location\_ID, Location\_Street, Location\_City, Location\_State, Location\_Zip\_Code, Location\_Phone, Location\_Email)

**Rental** (Rental\_ID, Rental\_Pickup\_Date, Rental\_Dropoff\_Date, Rental\_Pickup\_Time, Rental\_Dropoff\_Time, Pickup\_Mileage, Dropoff\_Mileage, Rental\_Car\_ReturnTankLevel, Rental\_Deposit\_Paid\_Status, Rental\_Tax, Rental\_Days, Rental\_charge)

**Insurance** (Insurance\_ID, Insurance\_By, Insurance\_Type, Insurance\_Perday\_Rate)

**Invoice** (Invoice\_Number, Total\_Charge\_Incx\_Tax, Debit\_Card\_Total, Credit\_Card\_Total, Cash\_Total, Payment\_Status, Payment\_Date)

**Deposit** (Deposit\_ID, Deposit\_Amount)

**License** (License\_Number, License\_Validity)

**Card\_Detail** (Card\_ID, Card\_Number, Card\_Type, Card\_Expiration\_Date)

**Tax** (Tax\_ID, Tax\_Rate)

**3nf:**

The table is in 2nf

There are no transitive dependencies

**BCNF:**

The 3nf table meets the BCNF requirements.

**Final Database Tables**

**Customer** (Customer\_ID, Customer\_FName, Customer\_LName, Customer\_DOB, Customer\_Street, Customer\_City, Customer\_State, Customer\_Zip\_Code, Customer\_Phone, Customer\_Email, *License\_Number*)

**Vehicles (**Vehicle\_ID, Vehicle\_Type, Vehicle\_Model, Vehicle\_Year, Vehicle\_Current\_Mileage, Vehicle\_Number\_of\_Seats, Vehicle\_Rental\_Status, Vehicle\_Rental\_Rate, Vehicle\_Color, *Location\_ID*)

**Location (**Location\_ID, Location\_Street, Location\_City, Location\_State, Location\_Zip\_Code, Location\_Phone, Location\_Email)

**Rental** (Rental\_ID, Rental\_Pickup\_Date, Rental\_Dropoff\_Date, Rental\_Pickup\_Time, Rental\_Dropoff\_Time, Pickup\_Mileage, Dropoff\_Mileage, Rental\_Car\_ReturnTankLevel, Rental\_Deposit\_Paid\_Status, Rental\_Tax, Rental\_Days, Rental\_charge, *Pickup\_Location\_ID, Dropoff\_Location\_ID, Customer\_ID, Vehicle\_ID, Insurance\_ID*)

**Insurance** (Insurance\_ID, Insurance\_By, Insurance\_Type, Insurance\_Perday\_Rate)

**Invoice** (Invoice\_Number, Total\_Charge\_Incx\_Tax, Debit\_Card\_Total, Credit\_Card\_Total, Cash\_Total, Payment\_Status, Payment\_Date, *Rental\_ID*)

**Deposit** (Deposit\_ID, Deposit\_Amount, *Card\_ID,* *Rental\_ID*)

**License** (License\_Number, License\_Validity)

**Card\_Detail** (Card\_ID, Card\_Number, Card\_Type, Card\_Expiration\_Date, *Customer\_ID*)

**Tax** (Tax\_ID, Tax\_Rate, *Location\_ID*)

**SQL Queries**

Task 1: In our project customers have two options whether to have their own insurance or should opt company’s insurance .

Here we are using a **subquery** to find out the details of customers who are using their own insurance

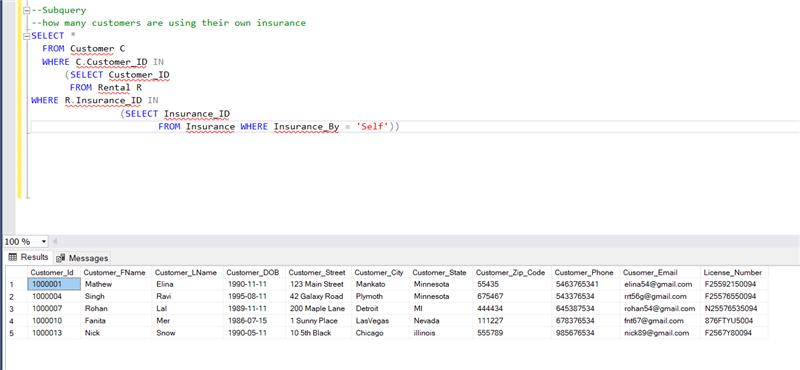
SELECT \*

FROM Customer C

WHERE C.Customer\_ID IN

(SELECT Customer\_ID FROM Rental R WHERE R.Insurance\_ID IN

(SELECT Insurance\_ID FROM Insurance WHERE Insurance\_By = 'Self')) ;



Task 2: Before renting any vehicle, the customer would like to see the details of the available vehicles he wants to rent from a specific location .

This **CTE**  gives the details of cars available at a particular location with the rented and non rented status .

WITH CARRENTALSTATUS (Vehicle\_id,vehicle\_model,Vehicle\_Type,Vehicle\_Rental\_Rate,Vehicle\_Year,

Vehicle\_Current\_Mileage,Vehicle\_Rental\_status,Location\_City,Location\_State)

AS( SELECT V.vehicle\_id,

V.vehicle\_model

,V.Vehicle\_Type

,V.Vehicle\_Rental\_Rate

,V.Vehicle\_Year,

V.Vehicle\_Current\_Mileage

,V.Vehicle\_Rental\_status

,Location\_City

,Location\_State

FROM Vehicle V

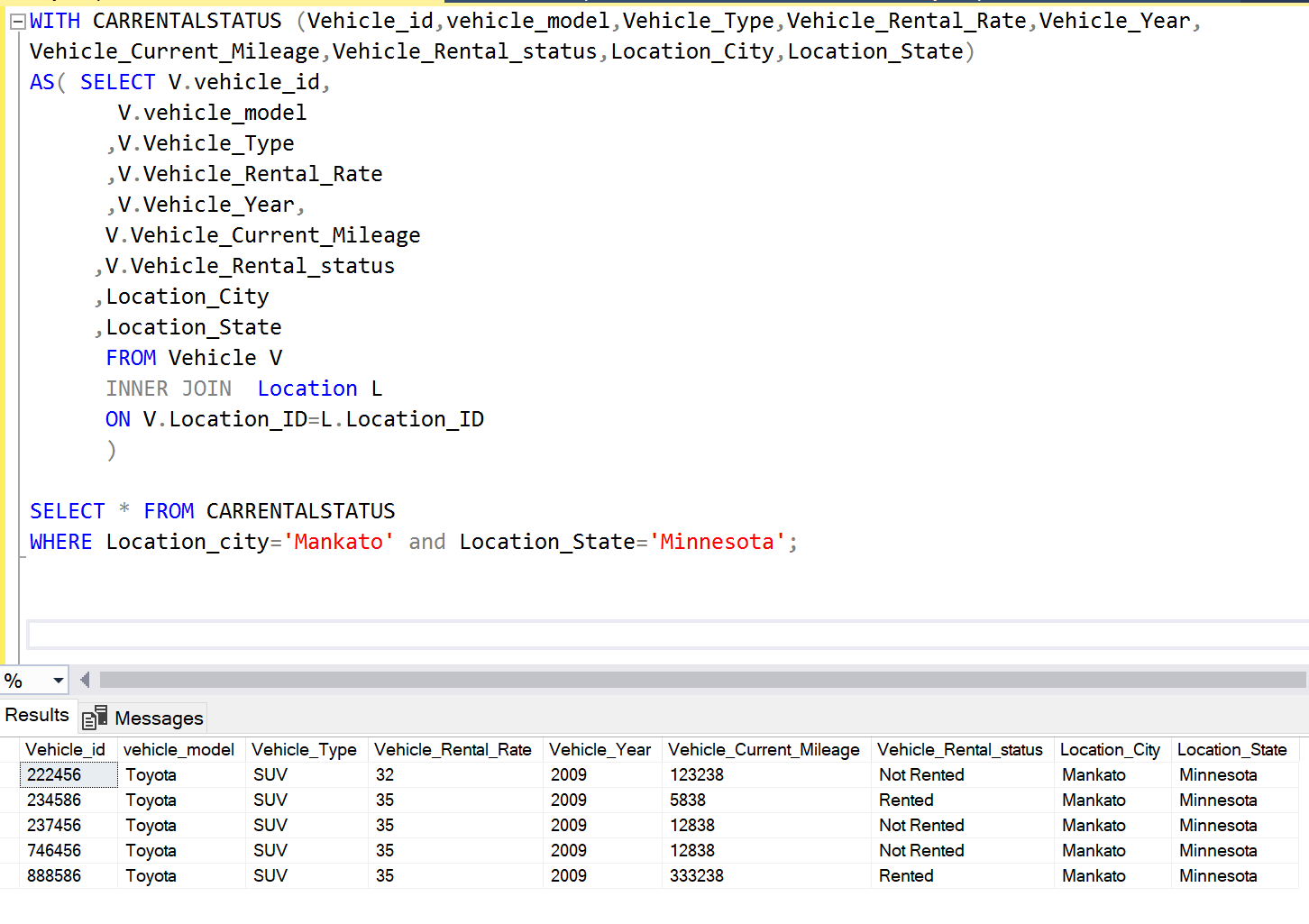
INNER JOIN Location L

ON V.Location\_ID=L.Location\_ID

)

SELECT \* FROM CARRENTALSTATUS

WHERE Location\_city='Mankato' and Location\_State='Minnesota';



Task 3: After every car rental , its current mileage need to be updated

**Subquery**: Update vehicle\_current\_mileage

UPDATE Vehicle

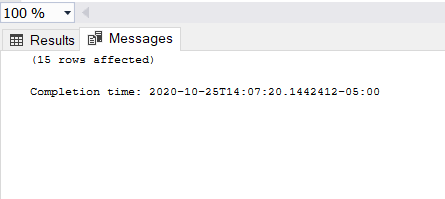
SET Vehicle\_Current\_Mileage =

(SELECT a.Dropoff\_Mileage

from (SELECT R.Dropoff\_Mileage, RANK () OVER ( PARTITION BY Vehicle\_ID ORDER BY RENTAL\_ID DESC) vrank

FROM Rental R WHERE

Vehicle.Vehicle\_ID = vehicle\_id ) a where a.vrank=1);





Task 4: To generate a bill many factors are looked on

Deposit status need to be updated, adding insurance charge if customer has taken company’s insurance , add fuel amount if customer is returning half talk or empty tank, calculating rental by multiplying no of days into car rate and then sum up all this for rental charge total

**Stored Procedure**: Updating rental table columns (Rental\_Deposit\_Paid\_Status ,rental\_charge ,Rental ) by checking various conditions and also vehicle table rental\_status column

IF OBJECT\_ID('update\_rental\_detail') IS NOT NULL

DROP PROCEDURE update\_rental\_detail;

GO

CREATE PROCEDURE update\_rental\_detail

AS

BEGIN

---UPDATING Rental\_Deposit\_Paid\_Status COLUMN OF RENTAL TABLE IF CUSTOMER HAS PAID THE DEPOSIT

UPDATE R

SET Rental\_Deposit\_Paid\_Status ='Paid'

FROM Rental R INNER JOIN Deposit D ON

R.Rental\_ID = D.Rental\_ID

WHERE Card\_id IS NOT NULL;

;

---UPDATING rental\_charge COLUMN OF RENTAL TABLE

UPDATE r

SET rental\_charge =

(r.rental\_days\* v.vehicle\_rental\_rate) +

(I.Insurance\_perday\_rate \*r.rental\_days) + ---IF CUSTOMER IS TAKING COMPANY'S INSURANCE

CASE

WHEN Rental\_Car\_ReturnTanklevel = 'HALF' THEN '15' ----CHARGE ADDED IF TANK IS NOT FULL

WHEN Rental\_Car\_ReturnTanklevel = 'EMPTY' THEN '25'

ELSE '0'

END

FROM rental r

INNER JOIN Vehicle v ON v.vehicle\_id= r.vehicle\_id

INNER JOIN Insurance I ON R.Insurance\_ID =I.Insurance\_ID;

---Updating Rental\_days column by calculating date difference

UPDATE Rental

SET Rental\_Days = datediff(d,Rental\_Pickup\_Date,Rental\_dropoff\_date);

/\*Updating VEHICLE table rental\_car\_status

If payment status is 'paid' in the invoice table , the car will be tagged as not rented in the vehicle table.

if rental table has any vehicle\_id and current date fall between rented dates ,

it will tagged as rented vehicle in vehicle table \*/

UPDATE V

SET V.Vehicle\_Rental\_Status ='Rented'

FROM Vehicle V

INNER JOIN Rental R

ON V.vehicle\_id=R.vehicle\_Id

WHERE CAST( GETDATE() AS Date ) BETWEEN R.Rental\_Pickup\_Date and R.Rental\_Dropoff\_Date ;

UPDATE V

SET V.Vehicle\_Rental\_Status ='Not Rented'

FROM Vehicle V

INNER JOIN Rental R ON

V.vehicle\_id=R.vehicle\_Id

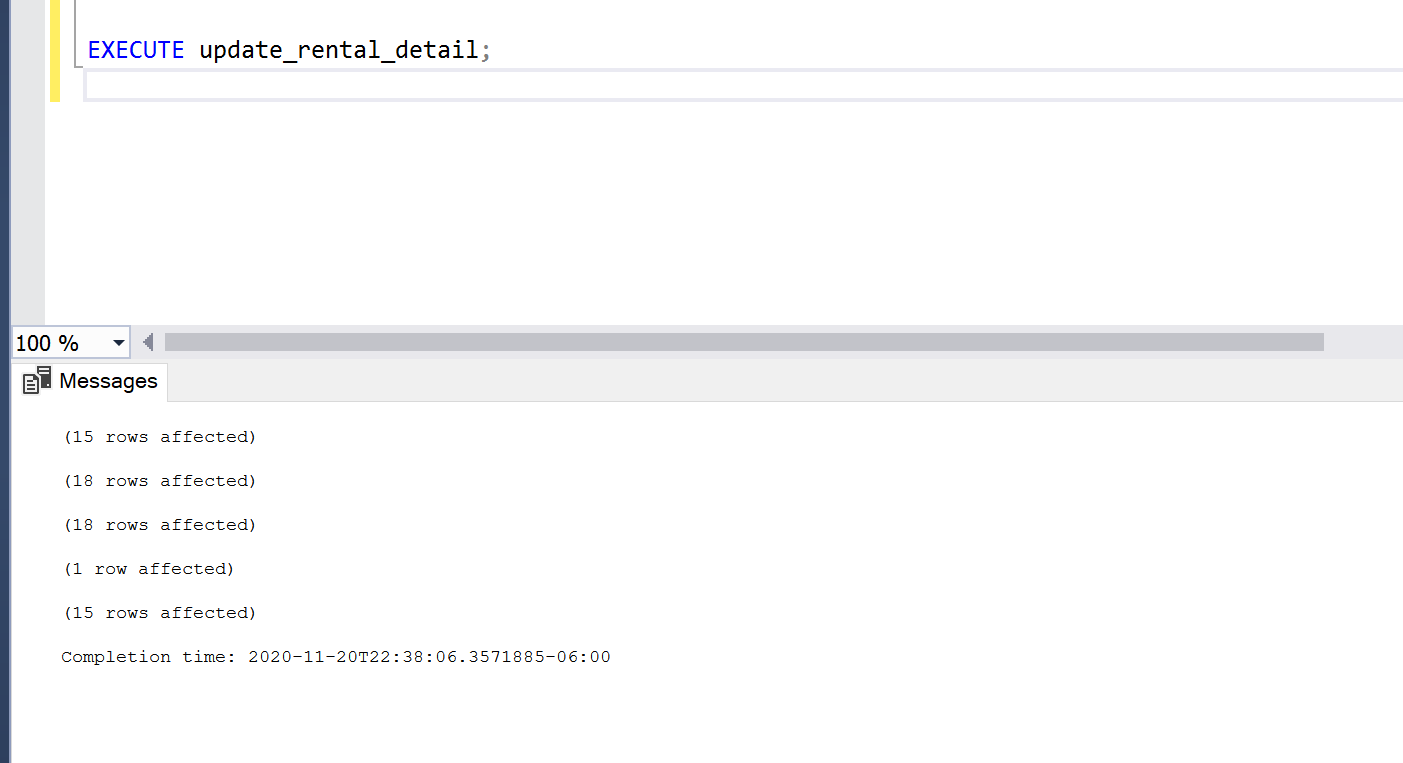
INNER JOIN Invoice I ON

I.Rental\_ID =R.Rental\_ID

WHERE Payment\_Status='PAID';

END;

EXECUTE update\_rental\_detail;



Task 5: For calculating the final invoice tax need to be added in the final rental charge .Tax depends upon the state .

Below **UDF** is Updating Total charge in invoice table by adding tax

IF OBJECT\_ID('ufnUpdateTotalChargeInInvoice') IS NOT NULL

DROP FUNCTION ufnUpdateTotalChargeInInvoice;

GO

CREATE FUNCTION ufnUpdateTotalChargeInInvoice(@rental\_charge float, @rental\_tax float)

RETURNS float

AS

BEGIN

DECLARE @total\_invoice\_charge float;

SET @total\_invoice\_charge = @rental\_charge + (@rental\_charge\*@rental\_tax)/100;

RETURN @total\_invoice\_charge;

END;

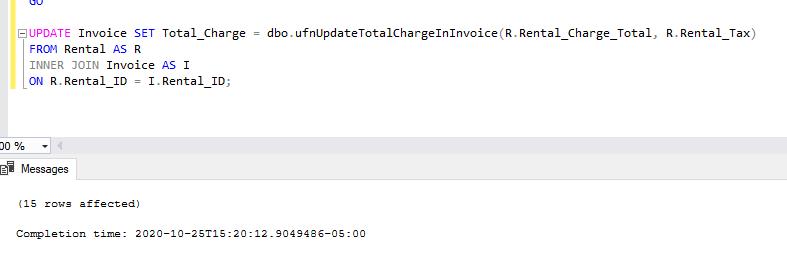
GO

UPDATE Invoice SET Total\_Charge\_Incx\_Tax= dbo.ufnUpdateTotalChargeInInvoice(R.Rental\_Charge, R.Rental\_Tax)

FROM Rental AS R

INNER JOIN Invoice AS I

ON R.Rental\_ID = I.Rental\_ID;



Task 6: Join query for total revenue at each location

\*\*note

To calculate the revenue we are using

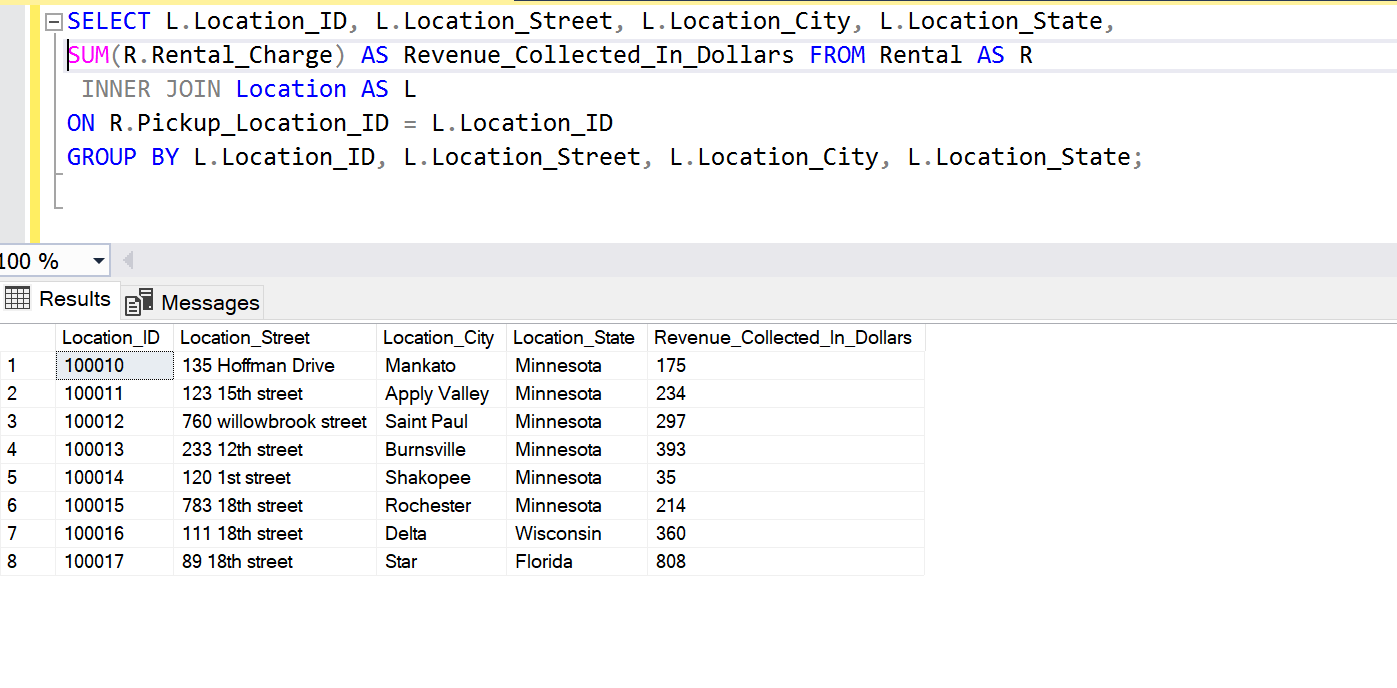
Rental table column named column rental\_charge which don’t have tax included and hence considered as revenue .Invoice table has total charge column which includes tax

SELECT L.Location\_ID, L.Location\_Street, L.Location\_City, L.Location\_State, SUM(R.Rental\_Charge) AS Revenue\_Collected\_In\_Dollars FROM Rental AS R

INNER JOIN Location AS L

ON R.Pickup\_Location\_ID = L.Location\_ID

GROUP BY L.Location\_ID, L.Location\_Street, L.Location\_City, L.Location\_State;



Task 7.

-- **TRIGGER** TO PROVIDE ERROR MESSAGE IF SOMEONE TRIES TO DELETE A RECORD FROM RENTAL TABLE

IF OBJECT\_ID('delete\_row\_msg') IS NOT NULL

DROP TRIGGER delete\_row\_msg;

GO

CREATE TRIGGER delete\_row\_msg

ON Invoice

AFTER DELETE

AS

BEGIN

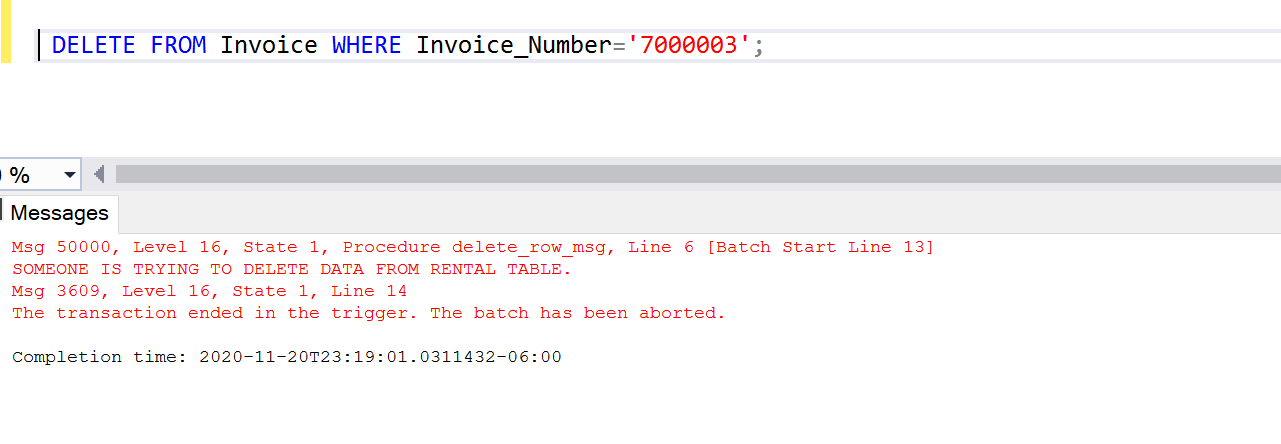
RAISERROR ('SOMEONE IS TRYING TO DELETE DATA FROM INVOICE TABLE.', 16, 1); ROLLBACK TRANSACTION;

RETURN

END;

GO

DELETE FROM Invoice WHERE Invoice\_Number='7000003';



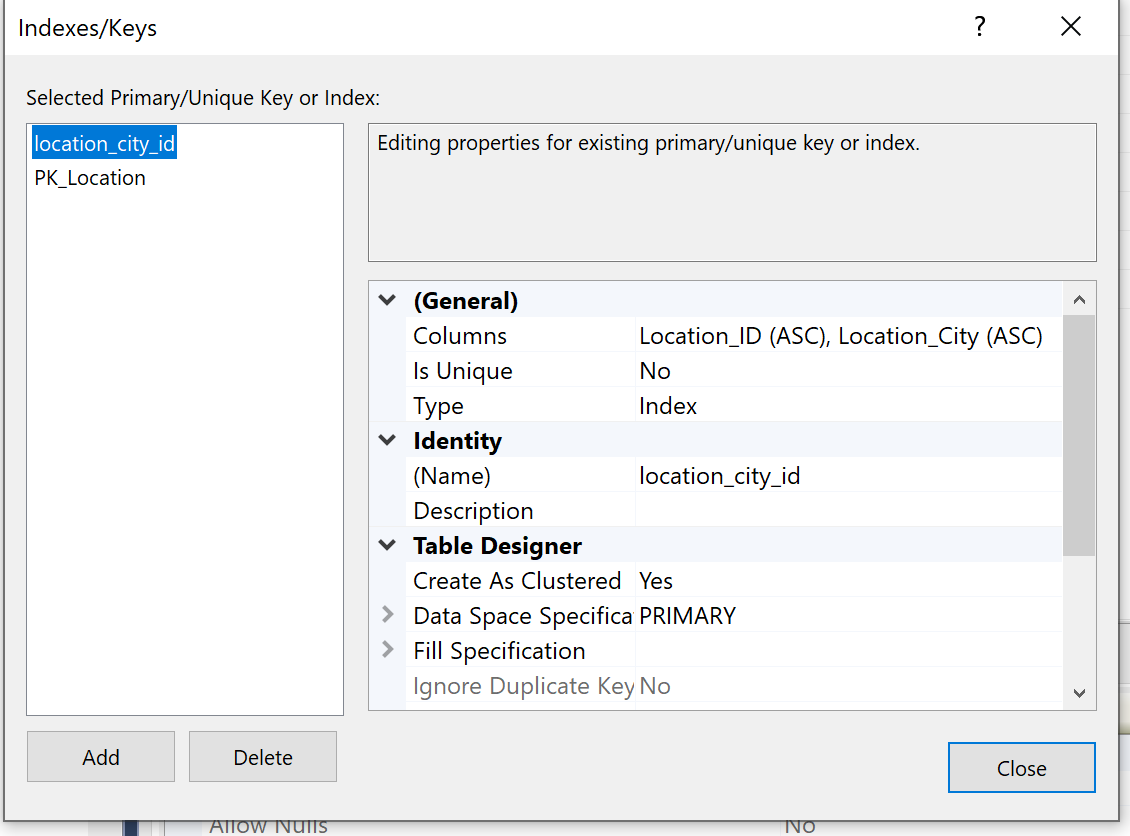
Task 8

By default a clustered index is created whenever a primary key is defined . Most of the time we are joining two tables on the basis of the primary key and all the primary keys already have a clustered index .

.

Other than primary key , **Location \_cit**y column will be used in where clause for searching cars across a specific city . Therefore we are defining clustered index on Location\_id and Location\_city

CREATE CLUSTERED INDEX location\_city\_id ON Location (Location\_id ASC,Location\_city ASC);



Task 9 : Pivot Table

Below query creates a pivot table which shows the count of no of times a particular car is rented .

select Vehicle\_ID ,[Toyota],[Suzuki],[BMW]

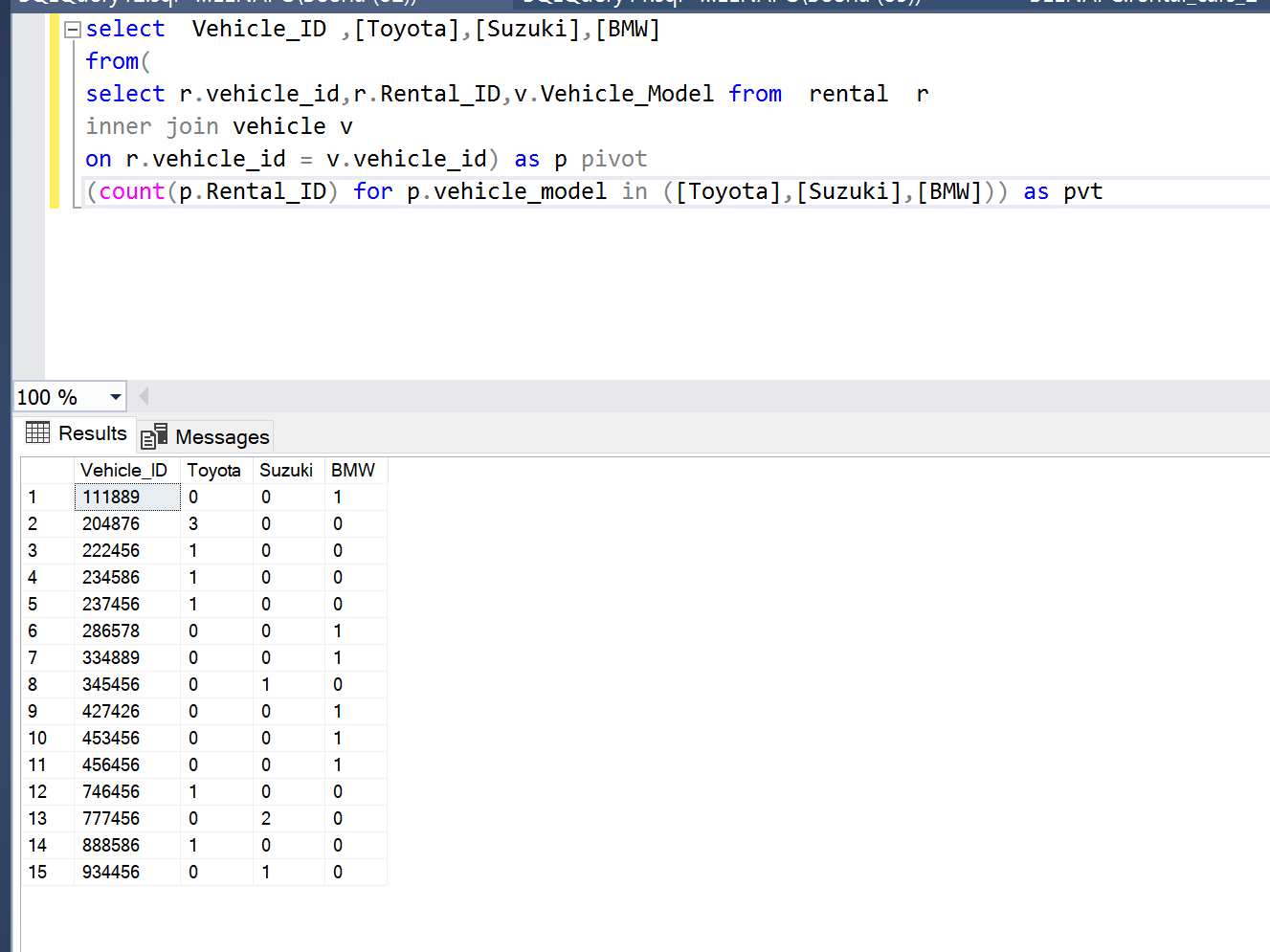
from(

select r.vehicle\_id,r.Rental\_ID,v.Vehicle\_Model from rental r

inner join vehicle v

on r.vehicle\_id = v.vehicle\_id) as p pivot

(count(p.Rental\_ID) for p.vehicle\_model in ([Toyota],[Suzuki],[BMW])) as pvt



Task 10

This cursor shows the first name of customers who rented vehicles for more than 2 days.

DECLARE @cust\_fname char(50);

DECLARE CustDdaysCursor CURSOR FOR SELECT C.Customer\_FName FROM Customer C

INNER JOIN Rental R

ON C.Customer\_Id = R.Customer\_Id

WHERE R.Rental\_Days > 2;

OPEN CustDdaysCursor;

FETCH NEXT FROM CustDdaysCursor INTO @cust\_fname;

WHILE @@FETCH\_STATUS = 0

BEGIN

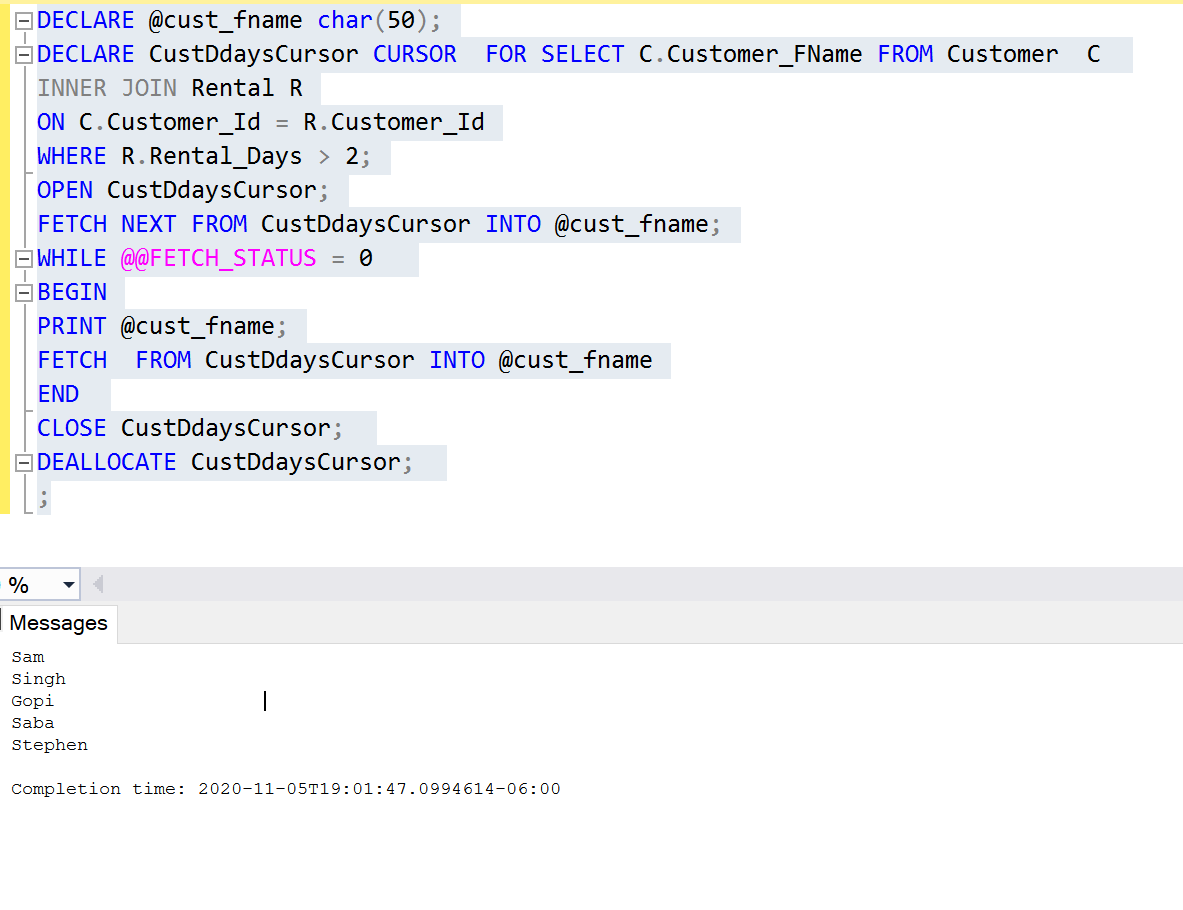
PRINT @cust\_fname;

FETCH FROM CustDdaysCursor INTO @cust\_fname

END

CLOSE CustDdaysCursor;

DEALLOCATE CustDdaysCursor;



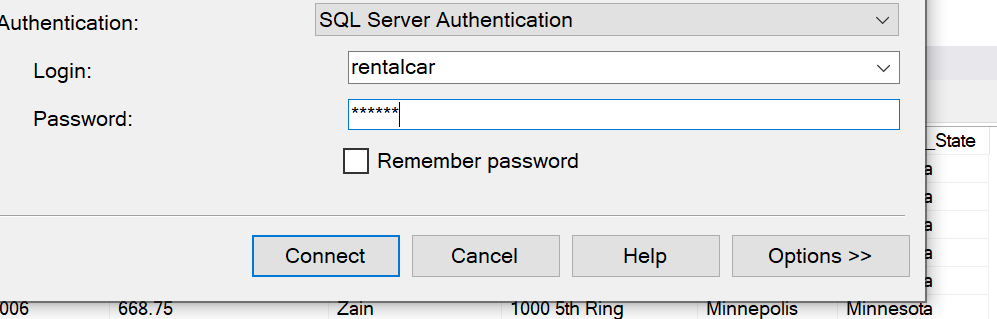
Task 11 Set up a “db\_datareader” login for your database.

1)

Created

Username : rentalcar

password “abc123”



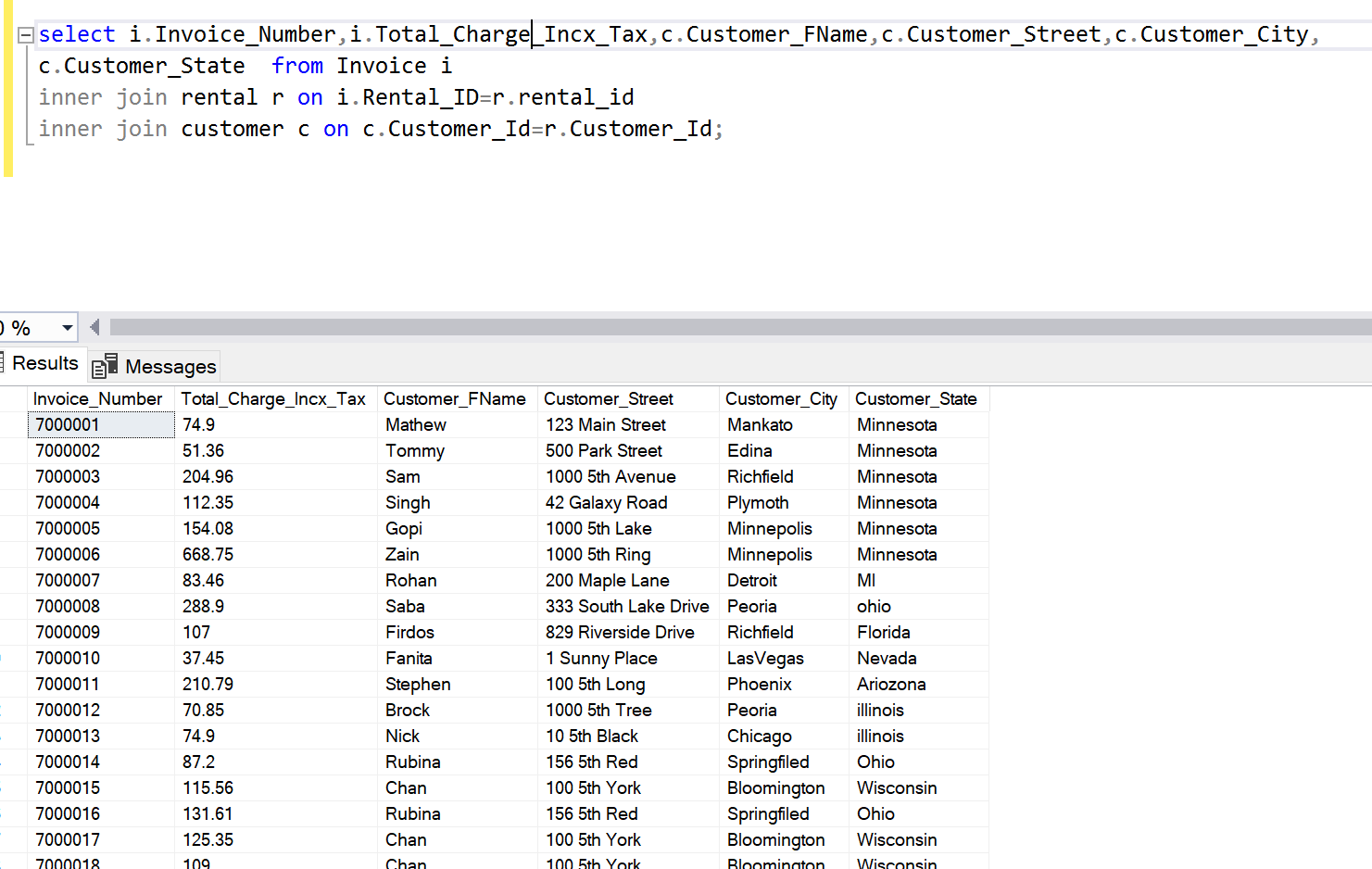
**Extra Queries**

select i.Invoice\_Number,i.Total\_Charge\_Incx\_Tax,c.Customer\_FName,c.Customer\_Street,c.Customer\_City,

c.Customer\_State from Invoice i

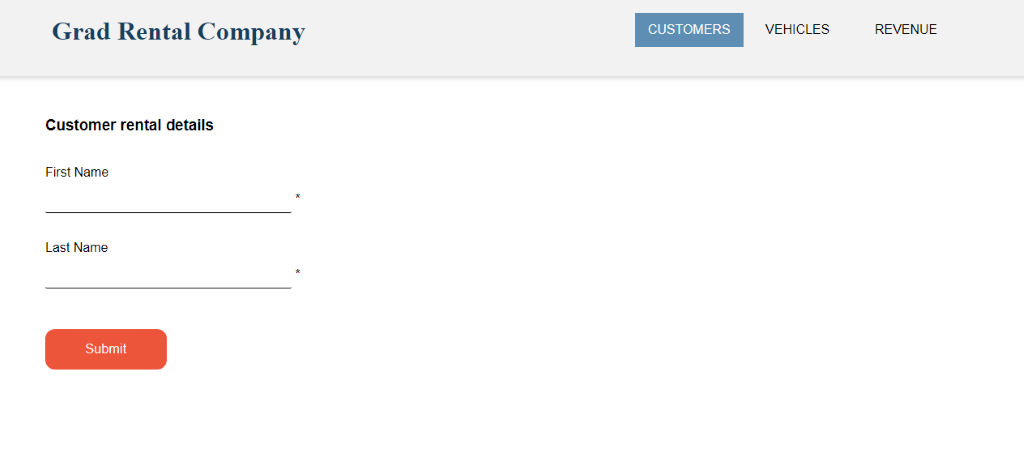
inner join rental r on i.Rental\_ID=r.rental\_id

inner join customer c on c.Customer\_Id=r.Customer\_Id;



**Web Application**

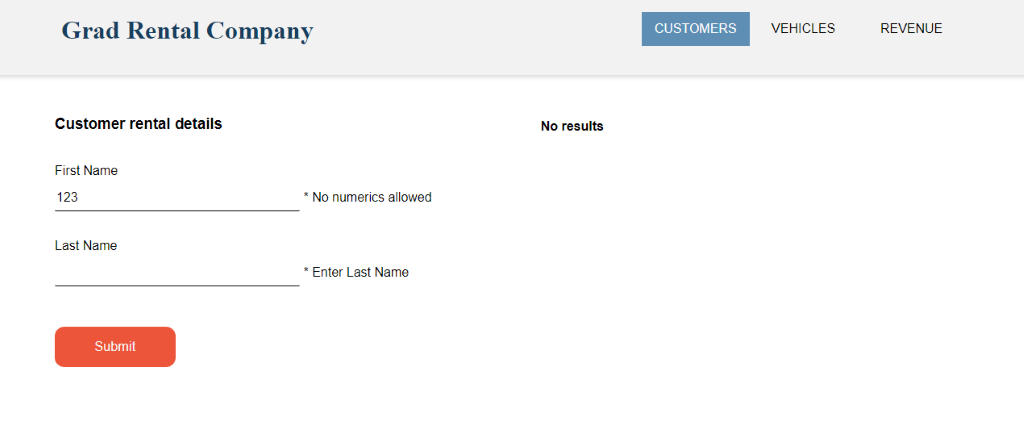
1. **Customers Page** – This page has a form to check the rental details of a particular customer. It gives the customer’s pick-up and drop-off locations, dates and number of days rented. We have also added the validations to the inputs, for example, if user doesn’t enter any value, or enters numeric, it shows error message



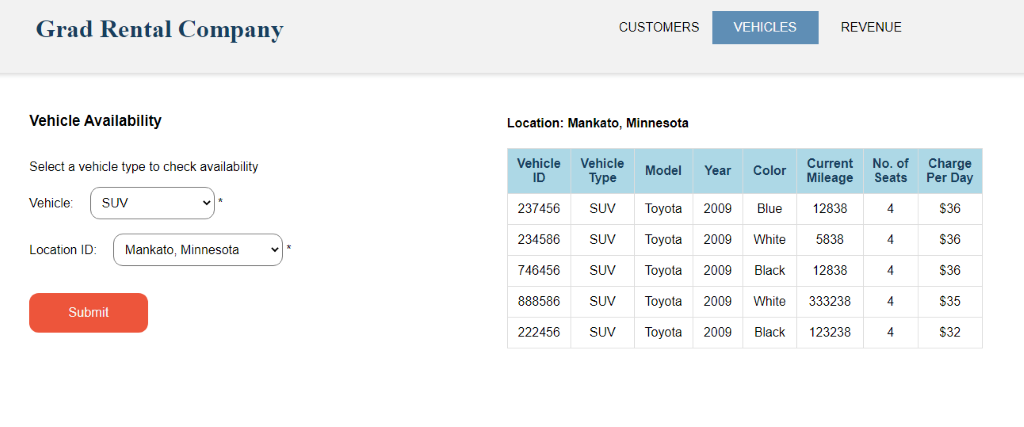
Customer rental details output



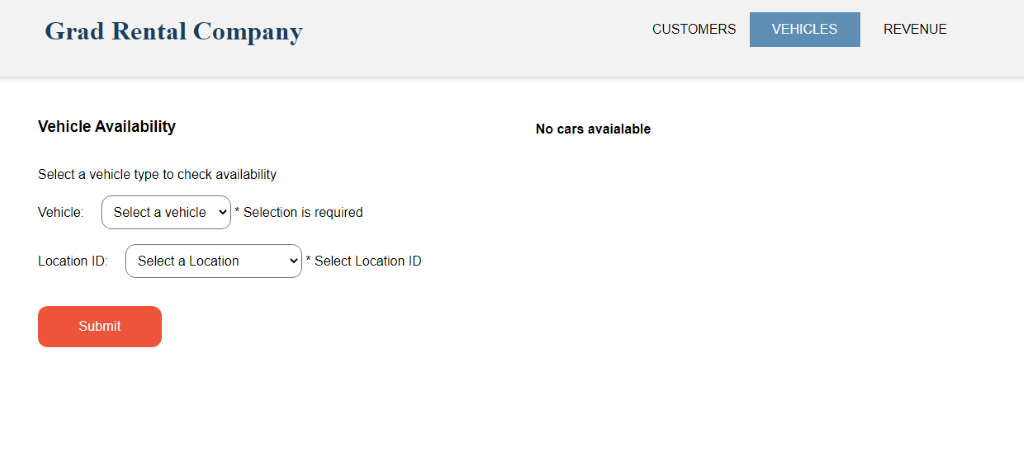
Input Validations



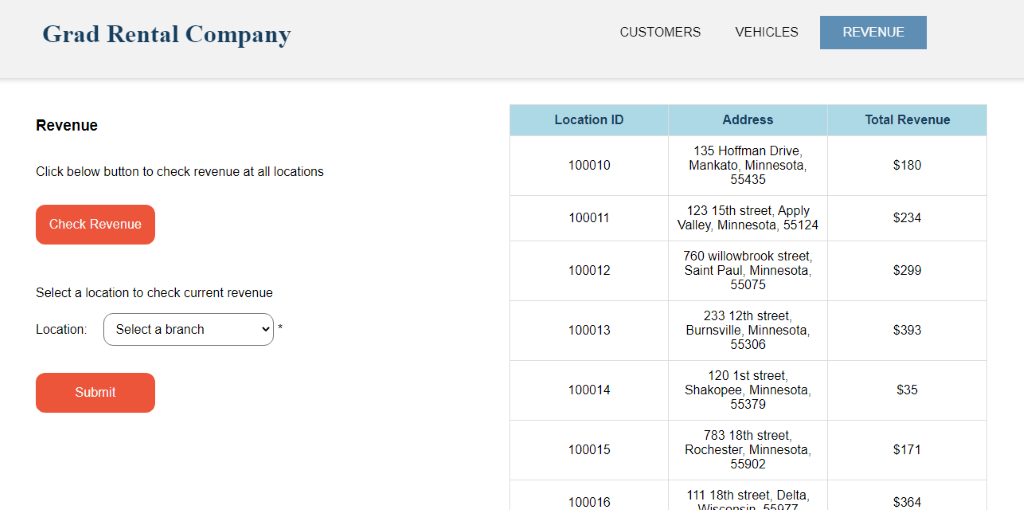
1. **Vehicles Page**– This page shows the available vehicles at a particular locations. It takes the type of Vehicle like ‘SUV’, ‘Truck’ as input and shows vehicles of the locations selected.



Validations



1. **Revenue Page** – Show the revenue collected at all the locations. It also provides an option to view Revenue of a particular locations. When the user clicks on ‘Check Revenue’ button, revenue collected at all locations is shown.



When user selects a particular location from the drop down, revenue of that location is shown.

