Group Number: 10

RENT-A-CAR DATABASE

CS 331 Section 002

Created by:

Rohith Shankar rg665@njit.edu

Nick Russo njr35@njit.edu

The purpose of the database is to manage various aspects of the car rental process. The main goal is to keep track of car inventory, rental contracts, and billing.

The system requirements are as follows:

Cars are assigned to specific locations and each location has one or more cars. Customers can make reservations for a specific car class and location. Reservations are classified into rental agreements, but reservations can also be canceled or remain unclaimed. A rental agreement is created for a specific vehicle, which may participate in multiple rental agreements over time.Car rental rates are determined by the car class, with daily and weekly rates for each class.The car model information includes make, year, and model name.Each car is uniquely identified by a VIN. The branch location has an address and location ID.The rental process involves customers making reservations, providing their information, and then getting a rental number based on the rental agreement. Rental agreements require customer information, including driver's license and credit card details. All rentals must be associated with a reservation. Upon car return, additional information is filled in, and the rental cost is calculated based on the class rental rate.

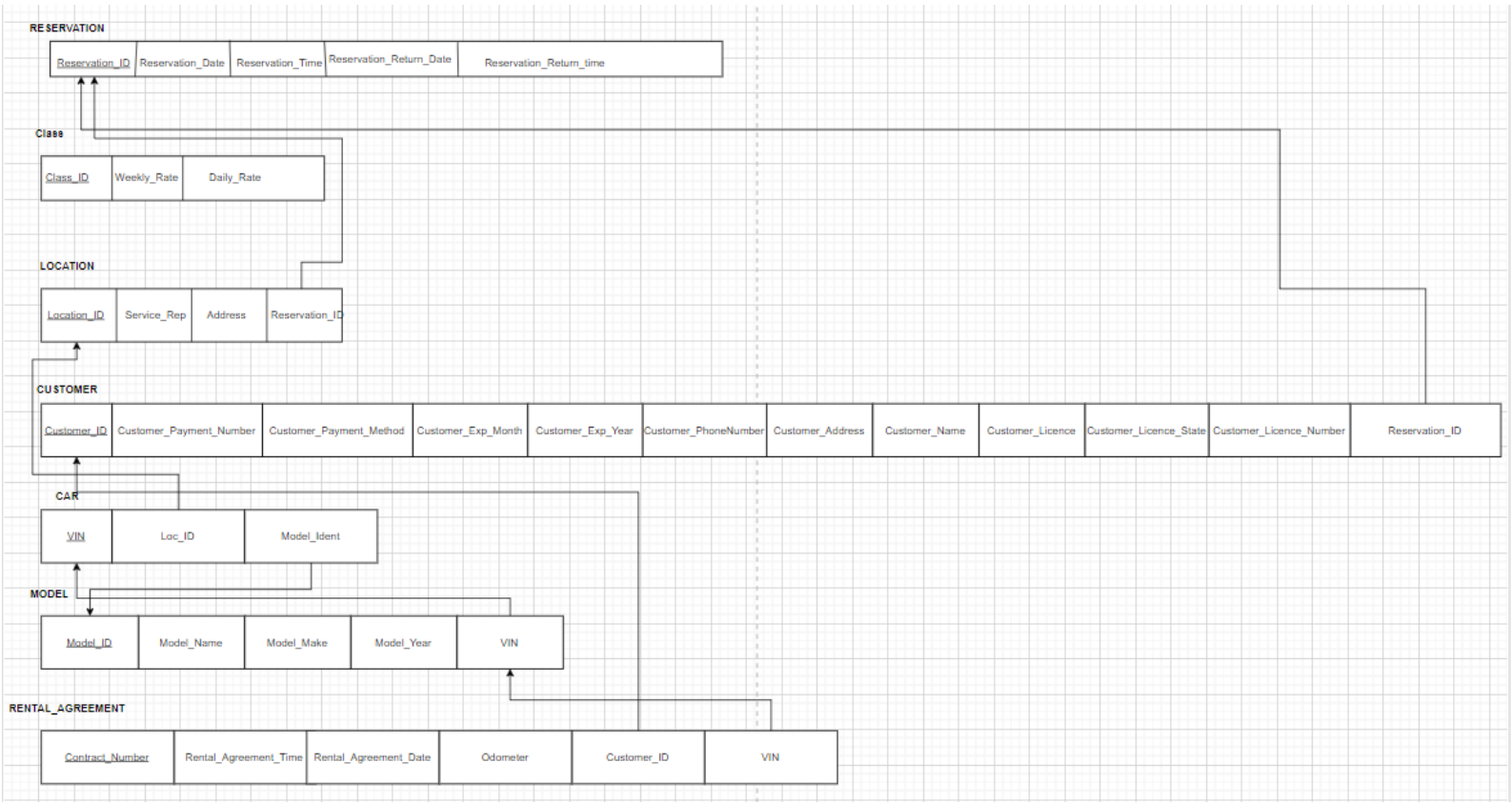
In an effort to meet all the requirements mentioned above, we have created tables to manage the following entities:

Reservation, Class, Location, Customer, Car, Model, and Rental Agreement. Foreign key constraints have been added to ensure referential integrity between tables. Sample data has been inserted using arbitrary data that we made up. Update statements have been created to modify records in each table (as per phase II). Delete statements were created to remove records from each table based on specific conditions (as per phase II). *Select queries have been created to do the following:* Return info such as the number of reservations made on each date. Return the reservations made on a specific date. Return the VIN of rental agreements with the lowest odometer reading. Return all of the customers who have made reservations with a specific service representative.

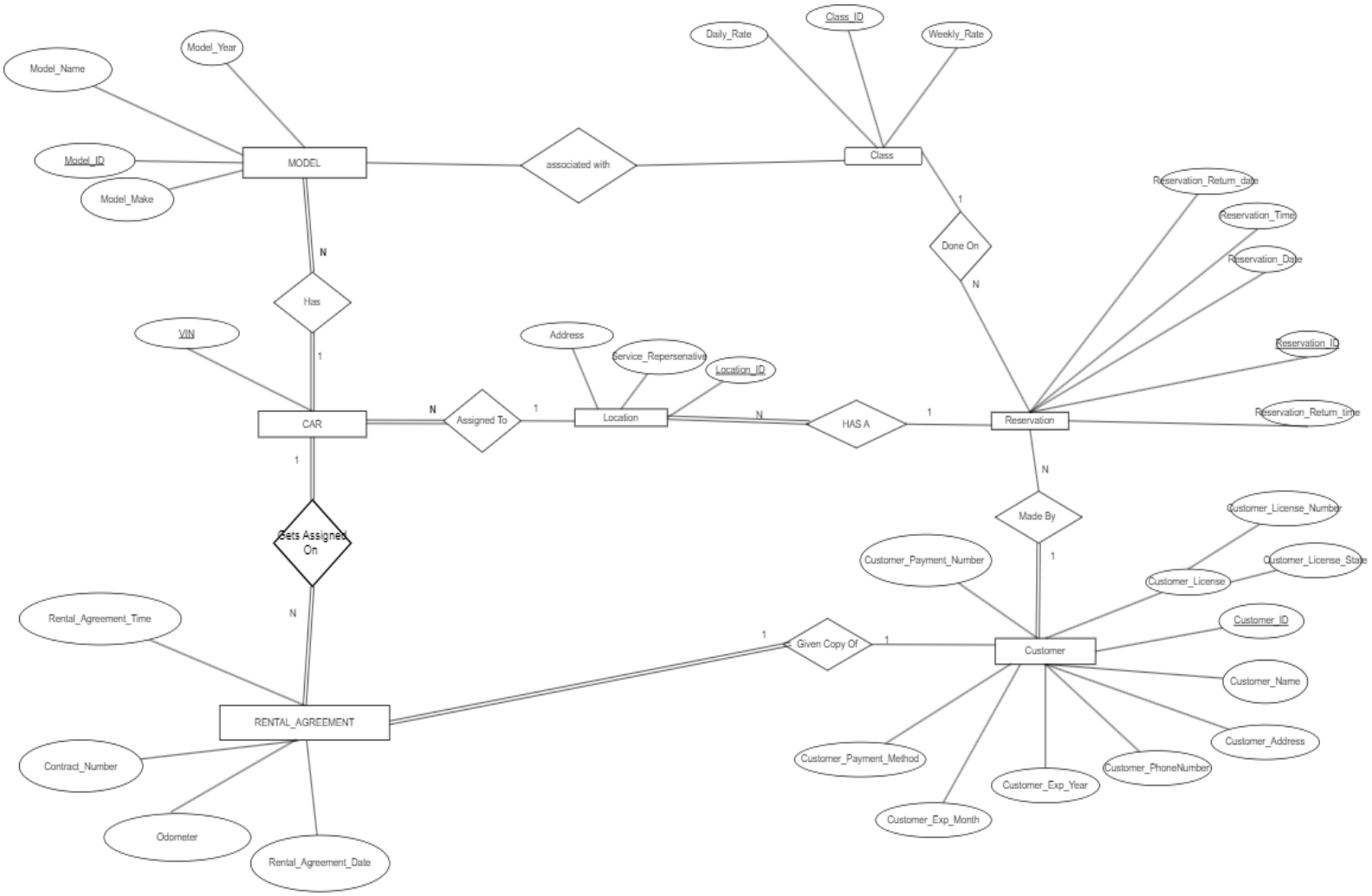
A quick rundown of the tables and their attributes in the database:

Reservation: Tracks reservations made by the customers. Class: Stores the class ID and corresponding rental rates. Location: Holds location details like service representatives and addresses. Customer: Contains customer information such as name, address, and payment method. Car: Keeps a record of cars, including their VIN, location, and model ID. Model: Includes car model details such as make, name, and year. Rental_Agreement: Stores rental agreement such as contract numbers and customer IDs.

Logical Database Design



Entity-Relationship Design



3). **Normalization**

A). RESERVATION(Reservation_ID, Reservation_DATE, Reservation_Time, Reservation_Return_Date, Reservation_Return_Time)

B).

```
SELECT * FROM Reservation;
```

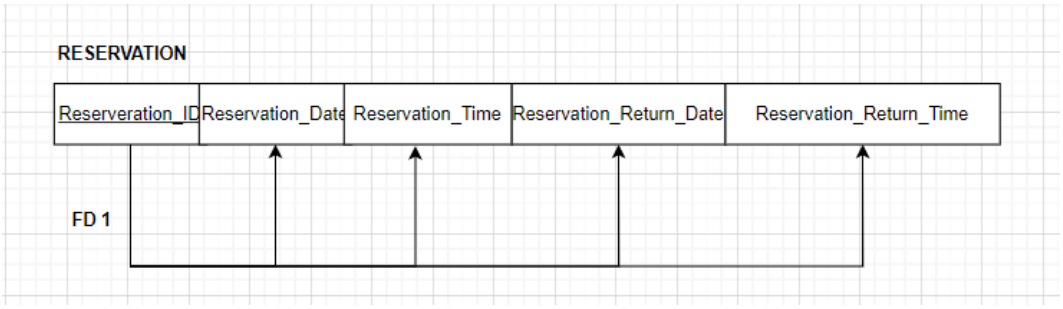
Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.007 seconds

	RESERVATION_ID	RESERVATION_DATE	RESERVATION_TIME	RESERVATION_RETURN_DATE	RESERVATION_RETURN_TIME
1	1	2023-04-15	2023-04-15 08:30:22	2023-04-16	2023-05-15 08:30:22
2	2	2023-04-16	2023-04-16 14:45:56	2023-04-17	2023-05-16 14:45:56
3	3	2023-04-17	2023-04-17 10:15:37	2023-04-18	2023-05-17 10:15:37
4	4	2023-04-18	2023-04-17 10:15:37	2023-04-19	2023-05-17 10:15:37
5	5	2023-04-19	2023-04-19 23:28:49	2023-04-20	2023-05-19 23:28:49

C). Key: Reservation_ID
Functional Dependencies: Reservation_ID->{Reservation_DATE, Reservation_Time, Reservation_Return_Date,Reservation_Return_Time}

D).



A). CLASS(Class_ID,Weekly_Rate,Daily_Rate)

B).

```
SELECT * FROM Class;
```

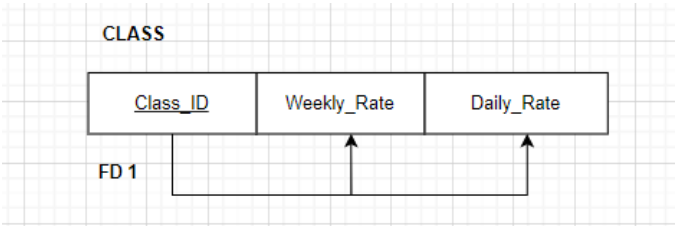
Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.006 seconds

	CLASS_ID	WEEKLY_RATE	DAILY_...
1	Premium	60	6
2	Sport	70	7
3	Luxury	80	8
4	Family	90	9
5	Outdoor	100	10

C). Key:Class_ID
Functional Dependencies: Class_ID-> {Weekly_Rate,Daily_Rate}

D).



A). CAR(VIN,Loc_ID,Model_Ident)

B).

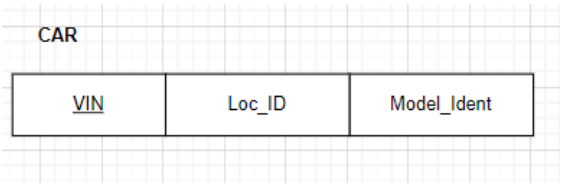
```
SELECT * FROM Car;
```

Script Output | Query Result | SQL | All Rows Fetched: 5 in 0.022 seconds

	VIN	LOC_ID	MODEL_IDENT
1	1HGCM82633A123456	1	1
2	2FMDK36C99B456789	2	2
3	1FTSW21R68E789012	3	3
4	5TDZA23C26S012345	4	4
5	JN8AZ2KR5BT678901	5	5

C). Key:VIN
Functional Dependencies: Loc_ID->{Location_ID}
Model_Ident->{Model_ID}

D).



A). LOCATION(Location_ID,Service_Rep,Address,Reservation_ID)

B).

```
SELECT * FROM Location;
```

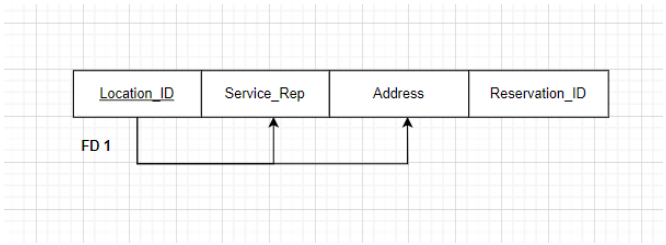
Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.009 seconds

	LOCATION_ID	SERVICE_REP	ADDRESS	RESERVATION_ID
1	1	walter	21 Sesame Street	1
2	2	Tom	31 Sesame Street	2
3	3	Mary	41 Sesame Street	3
4	4	jane	51 Sesame Street	4
5	5	watson	61 Sesame Street	5

C). Key: Location_ID
Functional Dependencies: Location_ID->{Service_Rep,Address}
Reservation_ID->Reservation_ID

D).



A).
CUSTOMER(Customer_ID,Customer_Payment_Number,Customer_Payment_Method,Customer_Exp_Month,Customer_Exp_Year,Customer_PhoneNumber,Customer_Address,Customer_Name,Customer_License,Customer_License_State,Customer_License_Number,Reservation_ID)

B).

```
SELECT * FROM Customer;
```

Script Output | Query Result | All Rows Fetched: 5 in 0.004 seconds

	CUSTOMER_ID	CUSTOMER_PAYMENT_NUMBER	CUSTOMER_PAYMENT_METHOD	CUSTOMER_EXP_MONTH	CUSTOMER_EXP_YEAR	CUSTOMER_PHONENUMBER	CUSTOMER_ADDRESS	CUSTOMER_NAME	CUSTOMER_LICENSE	CUSTOMER_LICENSE_STATE	CUSTOMER_LICENCE_NUMBER	RESERVATION_ID
1	1	1111222233334444	Credit Card	May	2025	973-123-1234	64 Sesame Street	Michael Jackson	AB123456	NY	12345678	1
2	2	2222333344445555	Credit Card	July	2024	973-234-3456	42 Sesame Street	Mr Peanut	CD789012	CA	23456789	2
3	3	3333444455556666	Debit Card	August	2023	973-234-5674	895 Sesame Street	nick russo	EF345678	TX	34567890	3
4	4	4444555566667777	Credit Card	September	2027	973-342-4563	34 Sesame Street	jack daniels	GH901234	FL	45678901	4
5	5	5555666677778888	Credit Card	October	2028	973-562-9085	89 Sesame Street	walter White	IJ567890	WA	56789012	5

C). Key: Customer_ID

<u>Functional Dependencies</u>

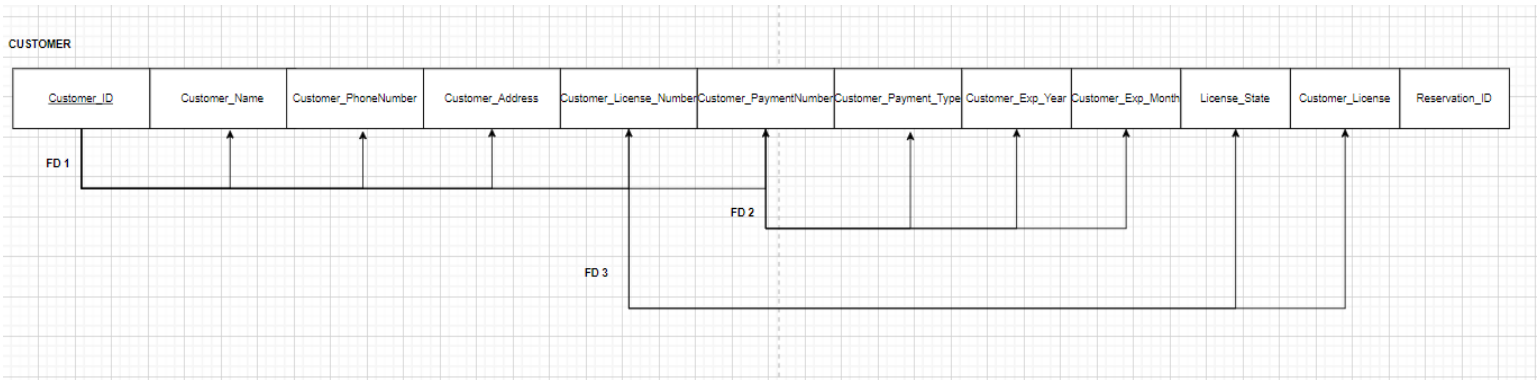
Customer_ID->{Customer_Name,Customer_PhoneNumber,Customer_Address,Customer_PaymentNumber,Customer_License_Number,}

Customer_Payment_Number->{Customer_Payment_Type,Customer_Exp_Year,Customer_Exp_Month}

Customer_License_Number->{License_State,Customer_License}

Reservation_ID ->Reservation_ID

D).



A). MODEL(Model_ID,Model_Name,Model_Make,Model_Year, VIN)

B).

```
SELECT * FROM Model;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.009 seconds

	MODEL_ID	MODEL_NAME	MODEL_MAKE	MODEL_YEAR	VIN
1	1	Civic	Honda	2023	1HGCM82633A123456
2	2	Camry	Toyota	2023	2FMDK36C99B456789
3	3	Accord	Honda	2022	1FTSW21R68E789012
4	4	Corolla	Toyota	2022	5TDZA23C26S012345
5	5	Altima	Nissan	2023	JN8AZ2KR5BT678901

C). Key: Model_ID

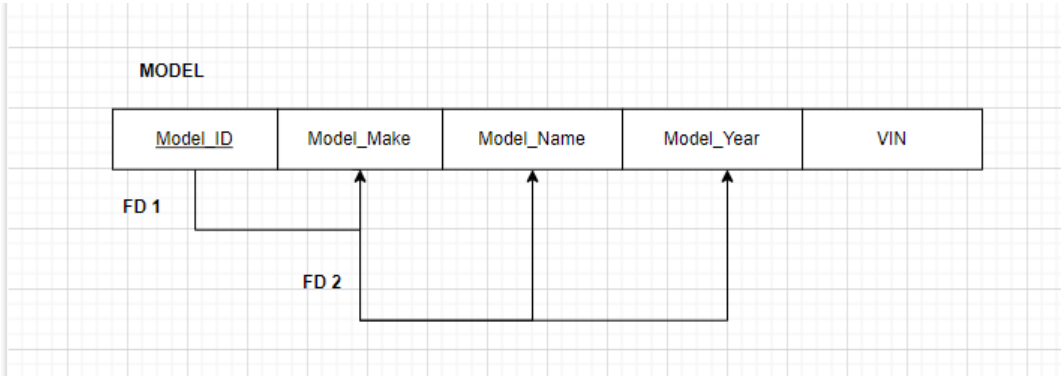
<u>Functional Dependencies</u>

Model_ID -> {Model_Make}

Model_Make->{Model_Name,Model_Year}

VIN -> VIN

D).



A).RENTAL_AGREEMENT(Rental_Agreement_Time,Rental_Agreement_Date,Odometer,Contract_Number,Customer_ID,VIN)

B).

```
SELECT * FROM Rental_Agreement;
```

Script Output | Query Result | SQL | All Rows Fetched: 5 in 0.008 seconds

	RENTAL_AGREEMENT_TIME	RENTAL_AGREEMENT_DATE	ODOMETER	CONTRACT_NUMBER	CUSTOMER_ID	VIN
1	2023-04-15 08:30:22	2023-04-16	442	RA2	1	1HGCM82633A123456
2	2023-04-16 14:45:56	2023-04-17	523	RA3	2	2FMDK36C99B456789
3	2023-04-17 10:15:37	2023-04-18	312	RA4	3	1FTSW21R68E789012
4	2023-04-18 19:52:11	2023-04-19	300	RA5	4	5TDZA23C26S012345
5	2023-04-19 23:28:49	2023-04-20	593	RA6	5	JN8AZ2KR5BT678901

C). Key: Contract_Number

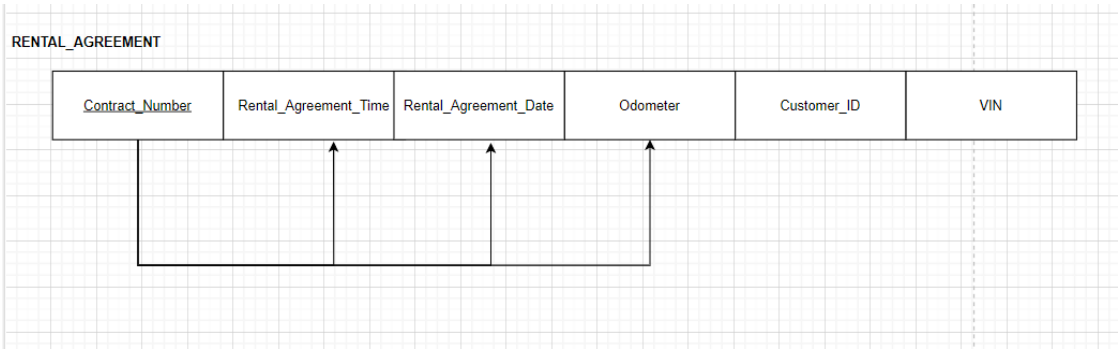
<u>Functional Dependencies</u>

Contract_Number->{Rental_Agreement_Time,Rental_Agreement_Date,Odometer}

Vin->{VIN}

Customer_ID->{Customer_ID}

D).



4.) **Write four queries in English and answer in SQL code,**

--Returns the number of reservations made on each date in the Reservation_Date column of Reservation.

- π Reservation_Date, COUNT(*) (σ Reservation)

```
SELECT  Reservation_Date,COUNT(*) as total_Reservation FROM Reservation GROUP BY Reservation_Date;
```

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.005 seconds

	RESERVATION_DATE	TOTAL_RESERVATION
1	2023-04-16	1
2	2023-04-17	1
3	2023-04-18	1
4	2023-04-19	1
5	2023-04-15	1

--Returns the number of reservations made on '2023-04-19'

– π COUNT(*) ((σ Reservation_Date='2023-04-19' (σ Reservation))

```
SELECT COUNT(*) as total_Reservation FROM Reservation GROUP BY Reservation_Date HAVING Reservation_Date= '2023-04-19';
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.004 seconds

	TOTAL_RESERVATION
1	1

-- Returns VIN of the rental agreements that have the lowest odometer reading
- π VIN ((σ Odometer ≤ γ min(Odometer)(Rental_Agreement)) Rental_Agreement)

```
SELECT VIN
FROM Rental_Agreement
WHERE Odometer <= ALL (
    SELECT Odometer
    FROM Rental_Agreement
);
```

Script Output × Query Result ×

SQL | All Rows Fetched: 1 in 0.005 secon

	VIN
1	5TDZA23C26S012345

– Returns all customers who have made reservations with the service representative named 'Tom'.
– Customer ⋈ Reservation ⋈ (σ Service_Rep = 'Tom' (Location))

```
SELECT *
FROM Customer
WHERE Reservation_ID IN (
    SELECT Reservation_ID
    FROM Location
    WHERE Service_Rep = 'Tom'
);
```

ript Output × Query Result ×

SQL | All Rows Fetched: 1 in 0.004 seconds

	CUSTOMER_ID	CUSTOMER_PAYMENT_NUMBER	CUSTOMER_PAYMENT_METHOD	CUSTOMER_EXP_MONTH	CUSTOMER_EXP_YEAR	CUSTOMER_PHONENUMBER	CUSTOMER_ADDRESS	CUSTOMER_NAME	CUSTOMER_LICENSE	CUSTOMER_LICENSE_STATE	CUSTOMER_LICENCE_NUMBER	RESERVATION_ID
1	2	2222333344445555	Credit Card	July	2024	973-234-3456	42 Sesame Street	John Peanut	CD789012	CA	23456789	2

5.) Conclusion

- After completing CS331 Rent-A-Car Database, we looked back on the experience of creating the database and we found at times that some tasks were challenging and some were simple. We found that creating the entity relationship diagram was the hardest part of the project because of how it plays a critical role in dictating both the functionality and overall logic of the database. Getting the entity relationship diagram wrong meant our implementation of the database within the sql code would subsequently reflet the errors made in the ER diagram. We were able to overcome the issues that we came across while creating the entity relationship diagram by thoroughly reading through the system requirements many times and cross checking to ensure we complied with the specific requirements. We also found the normalization of relations to be difficult at times, however, after some revision of notes and lengthy discussions, we were able to grasp a better deeper understanding of 3NF. Writing the queries was our favorite part of the project because we were able to realize our ideas from the ER diagram and implement them into the database which was both fun and a good way to practice sql. An unexpected lesson that we learned was the value of teamwork. We were able to work together very well and because of that, we were able to smoothly complete this project and be proud of the result. That helped us more deeply understand the value of teamwork as this project would have been much more challenging had we done it individually. Additionally, we learned how important it is to communicate with one another to overcome the challenges we encountered throughout the project. If we were to do this project all over again, the first thing that I would do is have a more structured plan before starting tasks. At times, we would start one of the phases without properly assigning specific roles to each other. Had we been able to do this over, we would most likely have laid out a structured plan so each of us knows exactly their role and thus, more efficiently reach our goal.