Student’s Full Name: Ansh Kumar Dev
Course Title: SQL/NoSQL Databases for Data and Information Sciences
Term name and year: Spring 2024
Submission Week: Week 16 Final Project Report
Instructor’s Name: Dr. Nayem Rahman
Date of Submission: 5/08/2024

1. Come up with a business plan for which you need to collect and store data

A revolutionary self-service moving and transportation solution tailored for the Indian market, MoveEZ aims to simplify the moving process for individuals and businesses alike. Leveraging an online plaXorm, MoveEZ offers a variety of vehicles for rent, catering to different moving needs - from small personal items to large household moves. Customers can easily book vehicles through the MoveEZ app or website, select from a range of rental durations, and enjoy competitive pricing. The service emphasizes convenience, affordability, and reliability, featuring realtime tracking of vehicles, 24/7 customer support, and flexible rental terms. MoveEZ is designed to fill the gap in the market for an efficient, user-friendly moving service, making it an ideal choice for those relocating homes, offices, or needing to transport goods across short and long distances within India.

2. Come up with the data about your business. Example: like the one I gave you for the assignment.

I will be creating my own dataset as discussed.

Database Structure and Data Insertion Overview

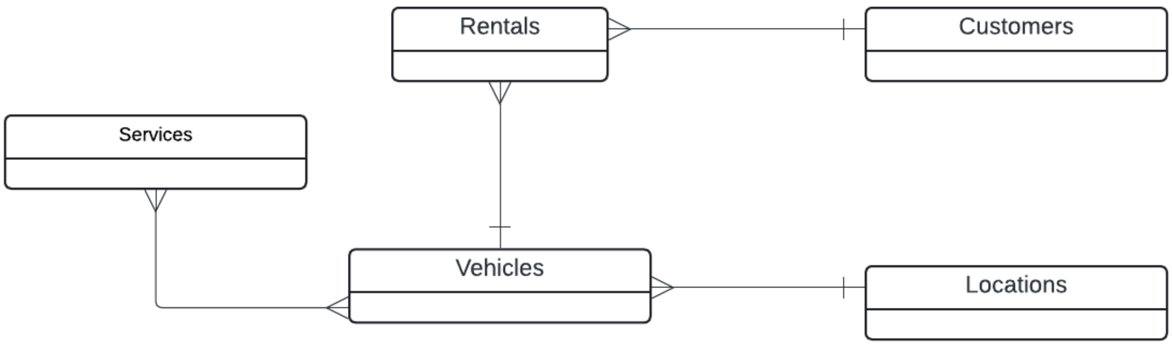
1. **Locations Table:**
    - **Data Inserted**: Three location entries representing Mumbai, New Delhi, and Bangalore. Each location has a specific address and zip code, indicating strategic positioning in major cities to cover a broad service area.
    - **Usage**: Helps in managing fleet logistics and assigning vehicles based on geographical demand and availability.
2. **Customers Table:**
    - **Data Inserted**: Five customers with names and contact details, reflecting a diverse clientele.
    - **Usage**: Customer data is essential for communication, booking management, and marketing strategies to enhance customer engagement and satisfaction.
3. **Vehicles Table:**
    - **Data Inserted**: Five different vehicles ranging in size and price, with status indicating availability.

    - **Usage**: Vehicle data supports dynamic fleet management, pricing strategies, and provides customers with suitable options based on their specific moving needs.

4. **Services Table:**
    - **Data Inserted**: Five different maintenance and service options, detailing the type of service and cost.
    - **Usage**: Critical for maintaining the fleet's operational efficiency and safety, ensuring that vehicles are always ready for customer use.
5. **Rentals Table:**
    - **Data Inserted**: Five rental transactions linking customers and vehicles, including rental dates, duration, and total cost.
    - **Usage**: Tracks rental activity, revenue per rental, and the effectiveness of pricing strategies. Also, essential for understanding customer preferences and usage patterns.
6. **Vehicle_Services Table:**
    - **Data Inserted**: Multiple entries for vehicle maintenance, showing the relationship between specific services and vehicles over time.
    - **Usage**: Provides a comprehensive history of each vehicle's maintenance, helping to plan future services and maintain high standards of vehicle reliability and performance.

3. Develop a Conceptual Model. Consider 4 or 5 entities. Make sure you have at least one many-to-many relationship. Explain with data why it's a many-to-many relationship.

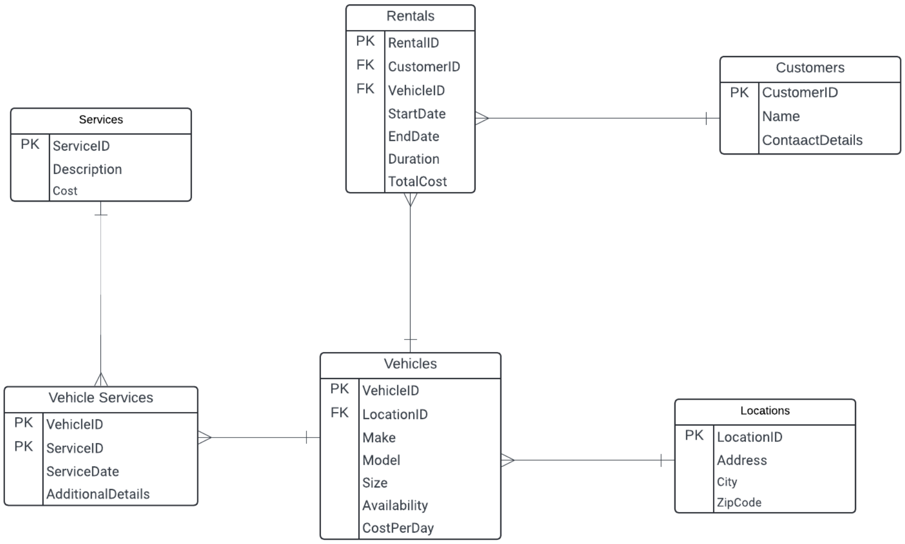
Conceptual Model



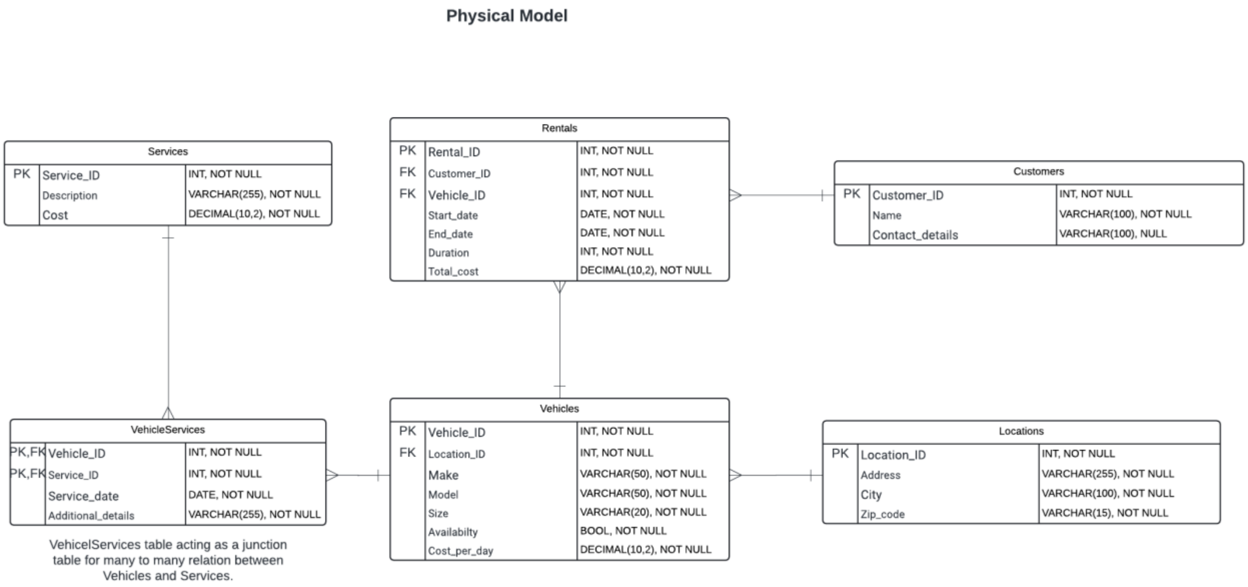
A many-to-many relationship in the context of the MoveEZ database is clearly illustrated between the Vehicles and Services tables. Each vehicle in the fleet can undergo various services, while each type of service can be applied to multiple vehicles, indicating the need for a many-to-many association. This relationship is efficiently managed through the use of a junction table, Vehicle_Services, which records each instance of a service provided to a vehicle.

4. Develop a Logical Model using the Conceptual Model. Make sure you come up with a junction entity to resolve the many-to-many relationship.

Logical Model



5. Develop the physical model based on the Logical Model



Physical Model

Services

PK	Service_ID	INT, NOT NULL
	Description	VARCHAR(255), NOT NULL
	Cost	DECIMAL(10,2), NOT NULL

Rentals

PK	Rental_ID	INT, NOT NULL
FK	Customer_ID	INT, NOT NULL
FK	Vehicle_ID	INT, NOT NULL
	Start_date	DATE, NOT NULL
	End_date	DATE, NOT NULL
	Duration	INT, NOT NULL
	Total_cost	DECIMAL(10,2), NOT NULL

Customers

PK	Customer_ID	INT, NOT NULL
	Name	VARCHAR(100), NOT NULL
	Contact_details	VARCHAR(100), NULL

VehicleServices

PK,FK	Vehicle_ID	INT, NOT NULL
PK,FK	Service_ID	INT, NOT NULL
	Service_date	DATE, NOT NULL
	Additional_details	VARCHAR(255), NOT NULL

Vehicles

PK	Vehicle_ID	INT, NOT NULL
FK	Location_ID	INT, NOT NULL
	Make	VARCHAR(50), NOT NULL
	Model	VARCHAR(50), NOT NULL
	Size	VARCHAR(20), NOT NULL
	Availabilty	BOOL, NOT NULL
	Cost_per_day	DECIMAL(10,2), NOT NULL

Locations

PK	Location_ID	INT, NOT NULL
	Address	VARCHAR(255), NOT NULL
	City	VARCHAR(100), NOT NULL
	Zip_code	VARCHAR(15), NOT NULL

VehicelServices table acting as a junction table for many to many relation between Vehicles and Services.

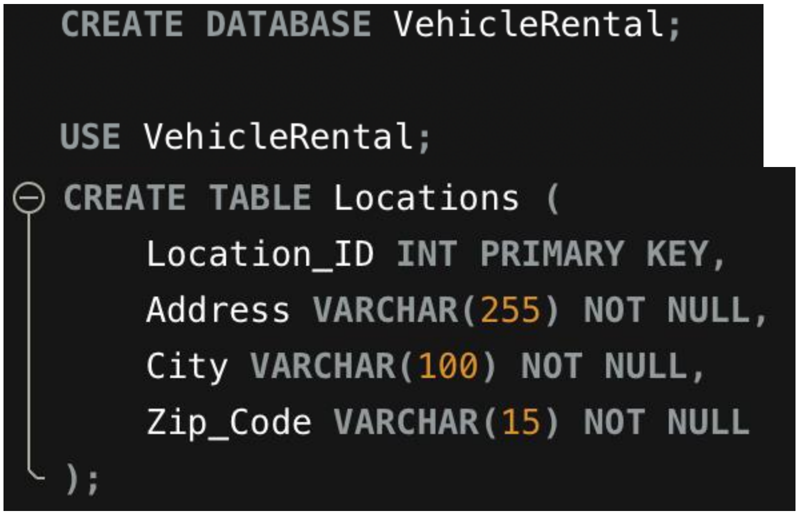
6. Create tables using a database system. Insert data into the database tables. You must provide the DDL (CREATE TABLE statements), INSERT statements, and SELECT statements.

Details: Create the tables that you have come up with (the table must be based on the Physical Model).

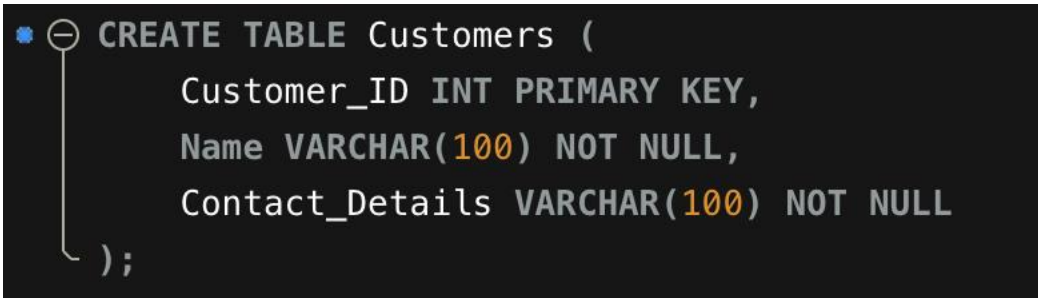
(a) Columns, Primary Key (PK), Data Type and length, and NULL/NOT NULL need to be implemented, per the Physical Model.

(b) Show the table definition (DDL) that you implemented (not in a graphical view).

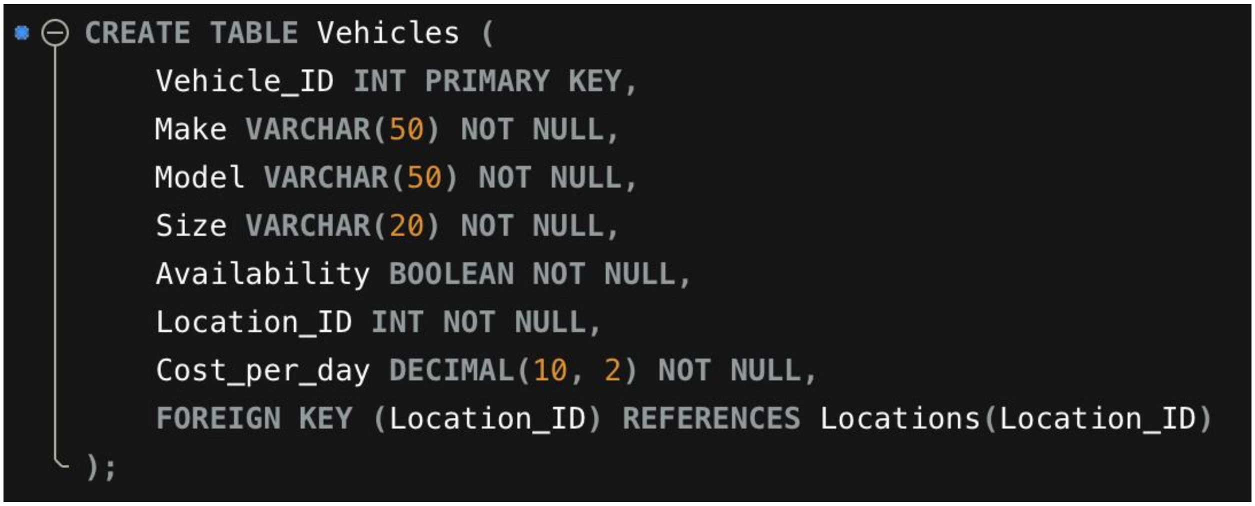
(c) Insert the complete set of data that you have come up with and show the insert statements used.



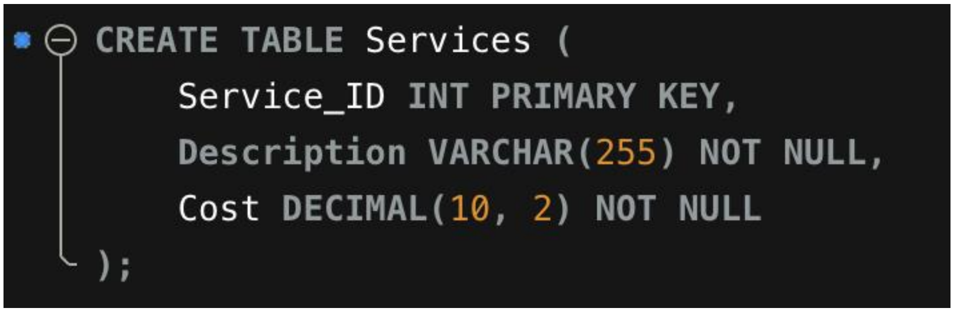
```
CREATE DATABASE VehicleRental;

USE VehicleRental;
CREATE TABLE Locations (
    Location_ID INT PRIMARY KEY,
    Address VARCHAR(255) NOT NULL,
    City VARCHAR(100) NOT NULL,
    Zip_Code VARCHAR(15) NOT NULL
);
```



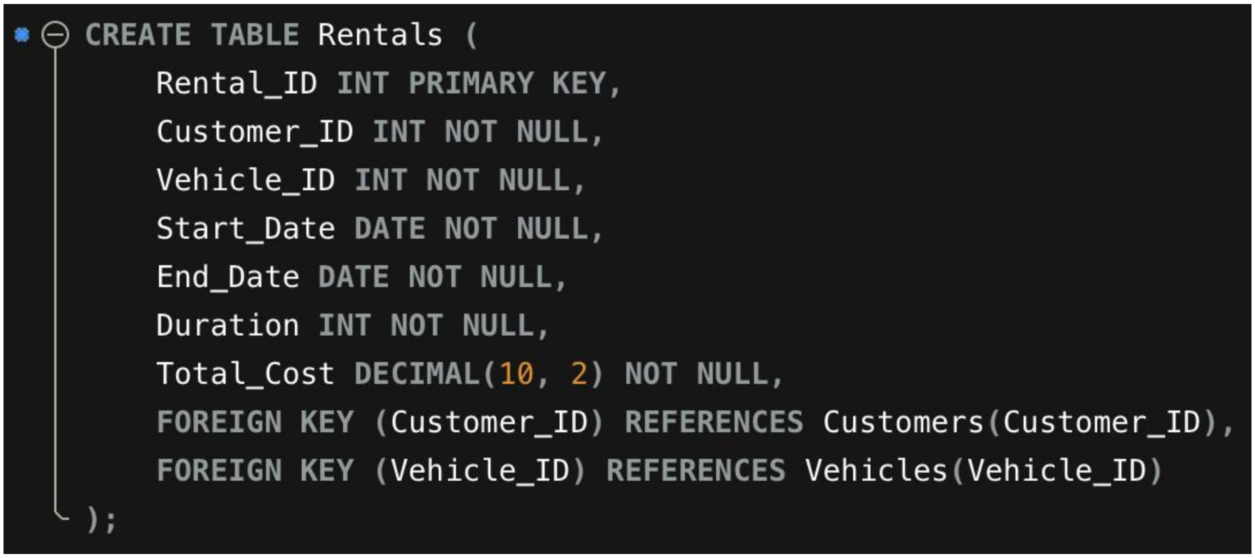
```sql
CREATE TABLE Customers (
    Customer_ID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Contact_Details VARCHAR(100) NOT NULL
);
```



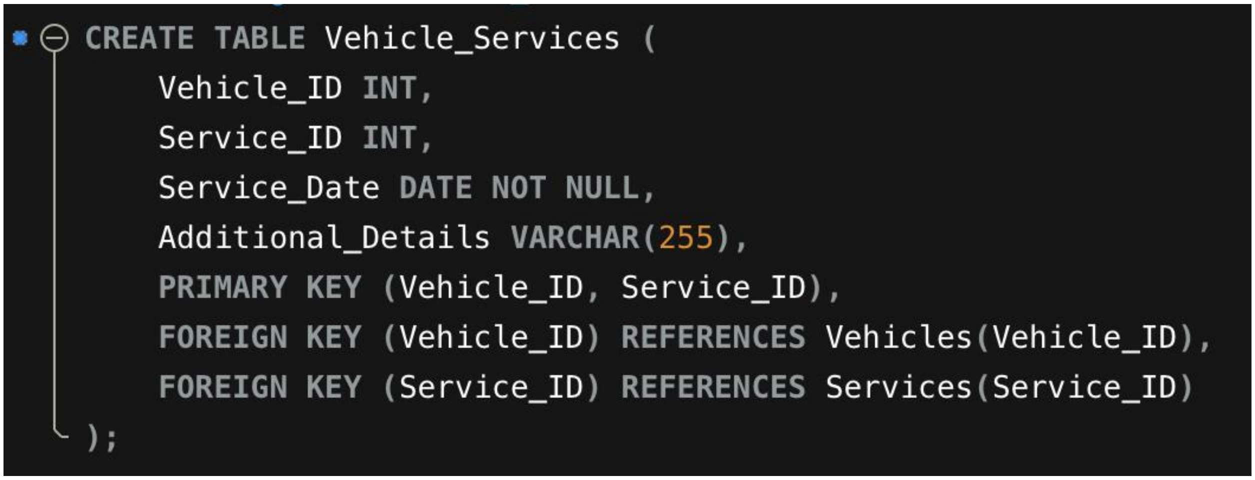
```sql
CREATE TABLE Vehicles (
    Vehicle_ID INT PRIMARY KEY,
    Make VARCHAR(50) NOT NULL,
    Model VARCHAR(50) NOT NULL,
    Size VARCHAR(20) NOT NULL,
    Availability BOOLEAN NOT NULL,
    Location_ID INT NOT NULL,
    Cost_per_day DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (Location_ID) REFERENCES Locations(Location_ID)
);
```



```sql
CREATE TABLE Services (
    Service_ID INT PRIMARY KEY,
    Description VARCHAR(255) NOT NULL,
    Cost DECIMAL(10, 2) NOT NULL
);
```

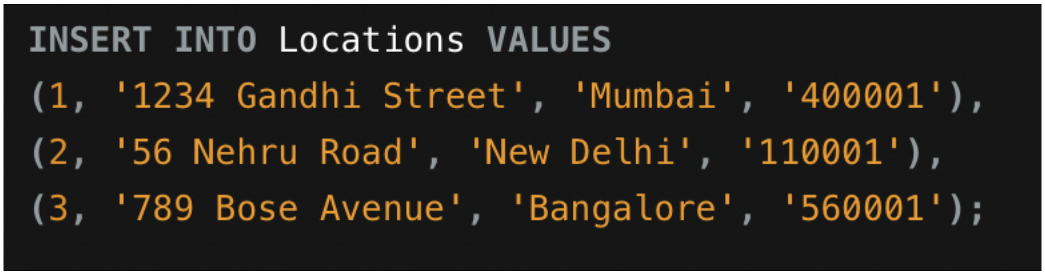


```sql
CREATE TABLE Rentals (
    Rental_ID INT PRIMARY KEY,
    Customer_ID INT NOT NULL,
    Vehicle_ID INT NOT NULL,
    Start_Date DATE NOT NULL,
    End_Date DATE NOT NULL,
    Duration INT NOT NULL,
    Total_Cost DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (Customer_ID) REFERENCES Customers(Customer_ID),
    FOREIGN KEY (Vehicle_ID) REFERENCES Vehicles(Vehicle_ID)
);
```

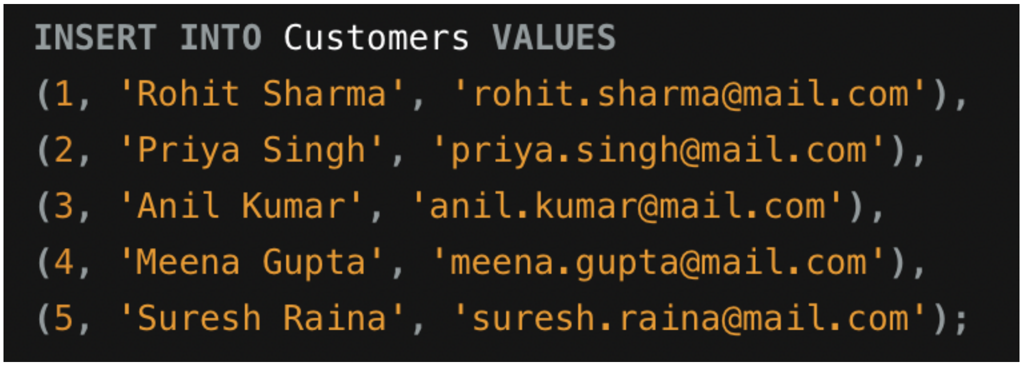


```sql
CREATE TABLE Vehicle_Services (
    Vehicle_ID INT,
    Service_ID INT,
    Service_Date DATE NOT NULL,
    Additional_Details VARCHAR(255),
    PRIMARY KEY (Vehicle_ID, Service_ID),
    FOREIGN KEY (Vehicle_ID) REFERENCES Vehicles(Vehicle_ID),
    FOREIGN KEY (Service_ID) REFERENCES Services(Service_ID)
);
```

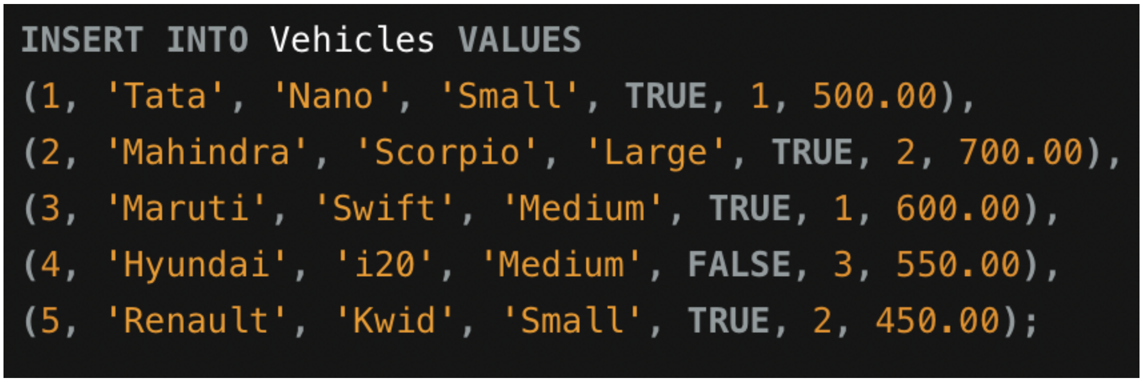
Inserting Data into tables.



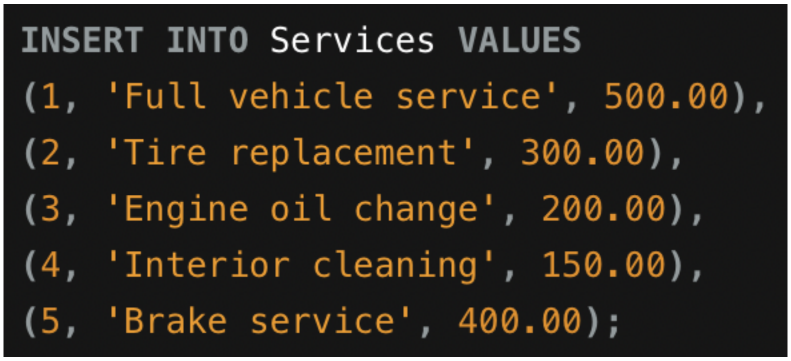
```sql
INSERT INTO Locations VALUES
(1, '1234 Gandhi Street', 'Mumbai', '400001'),
(2, '56 Nehru Road', 'New Delhi', '110001'),
(3, '789 Bose Avenue', 'Bangalore', '560001');
```



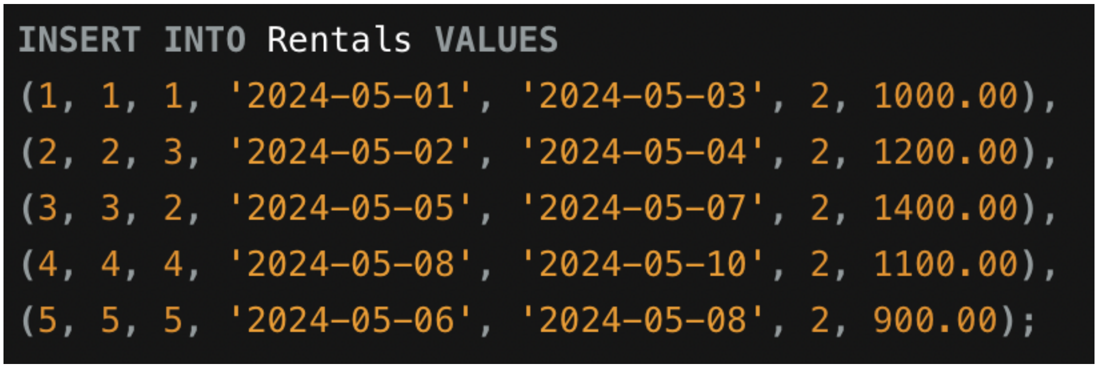
```
INSERT INTO Customers VALUES
(1, 'Rohit Sharma', 'rohit.sharma@mail.com'),
(2, 'Priya Singh', 'priya.singh@mail.com'),
(3, 'Anil Kumar', 'anil.kumar@mail.com'),
(4, 'Meena Gupta', 'meena.gupta@mail.com'),
(5, 'Suresh Raina', 'suresh.raina@mail.com');
```



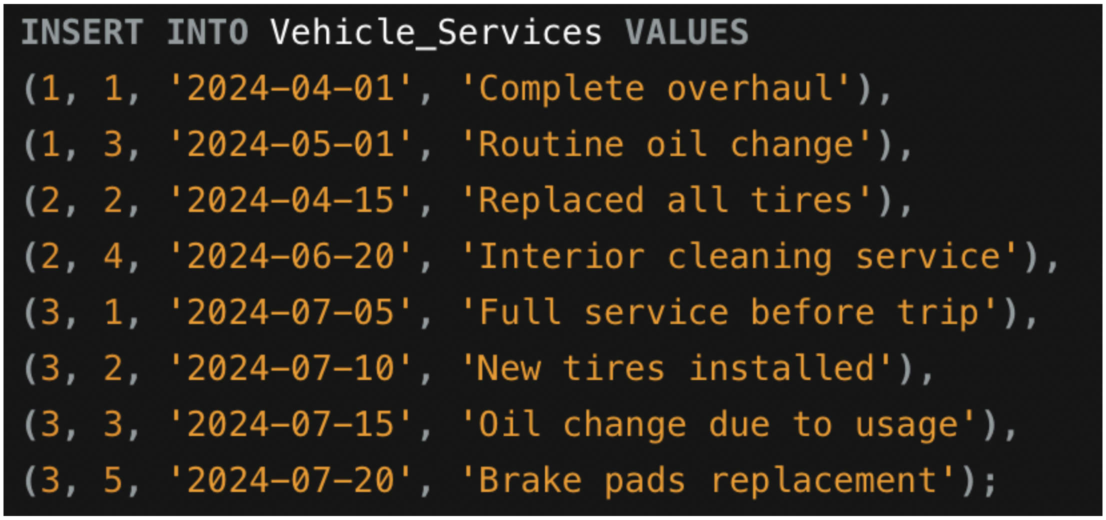
```
INSERT INTO Vehicles VALUES
(1, 'Tata', 'Nano', 'Small', TRUE, 1, 500.00),
(2, 'Mahindra', 'Scorpio', 'Large', TRUE, 2, 700.00),
(3, 'Maruti', 'Swift', 'Medium', TRUE, 1, 600.00),
(4, 'Hyundai', 'i20', 'Medium', FALSE, 3, 550.00),
(5, 'Renault', 'Kwid', 'Small', TRUE, 2, 450.00);
```



```
INSERT INTO Services VALUES
(1, 'Full vehicle service', 500.00),
(2, 'Tire replacement', 300.00),
(3, 'Engine oil change', 200.00),
(4, 'Interior cleaning', 150.00),
(5, 'Brake service', 400.00);
```



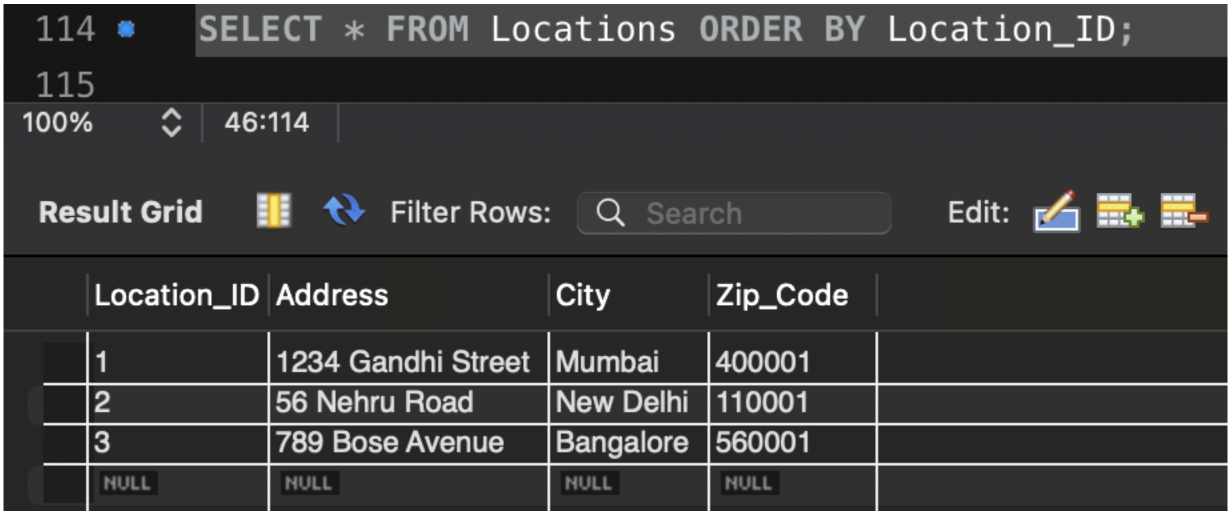
```
INSERT INTO Rentals VALUES
(1, 1, 1, '2024-05-01', '2024-05-03', 2, 1000.00),
(2, 2, 3, '2024-05-02', '2024-05-04', 2, 1200.00),
(3, 3, 2, '2024-05-05', '2024-05-07', 2, 1400.00),
(4, 4, 4, '2024-05-08', '2024-05-10', 2, 1100.00),
(5, 5, 5, '2024-05-06', '2024-05-08', 2, 900.00);
```



```sql
INSERT INTO Vehicle_Services VALUES
(1, 1, '2024-04-01', 'Complete overhaul'),
(1, 3, '2024-05-01', 'Routine oil change'),
(2, 2, '2024-04-15', 'Replaced all tires'),
(2, 4, '2024-06-20', 'Interior cleaning service'),
(3, 1, '2024-07-05', 'Full service before trip'),
(3, 2, '2024-07-10', 'New tires installed'),
(3, 3, '2024-07-15', 'Oil change due to usage'),
(3, 5, '2024-07-20', 'Brake pads replacement');
```

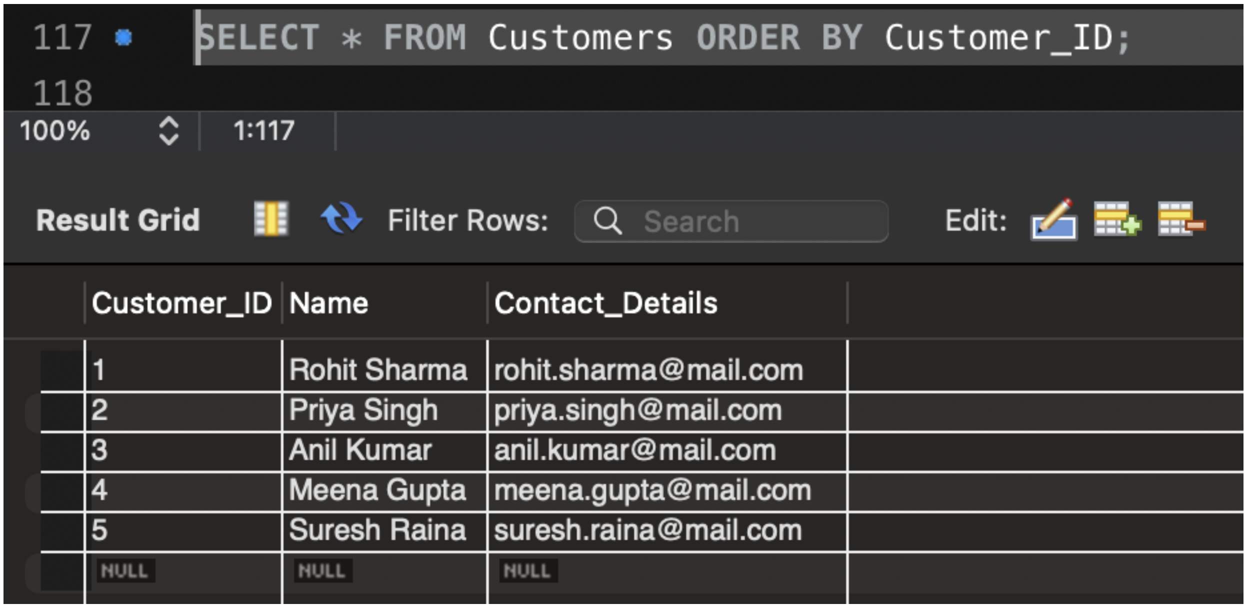
7. Create a variety of SQL queries to retrieve data from one or many tables:

1. Retrieve the data from each table by using the SELECT * statement and order by PK column(s).
Show the output. Make sure you show the print screen of the complete set of rows and columns.
The rows must be ordered by PK column(s).



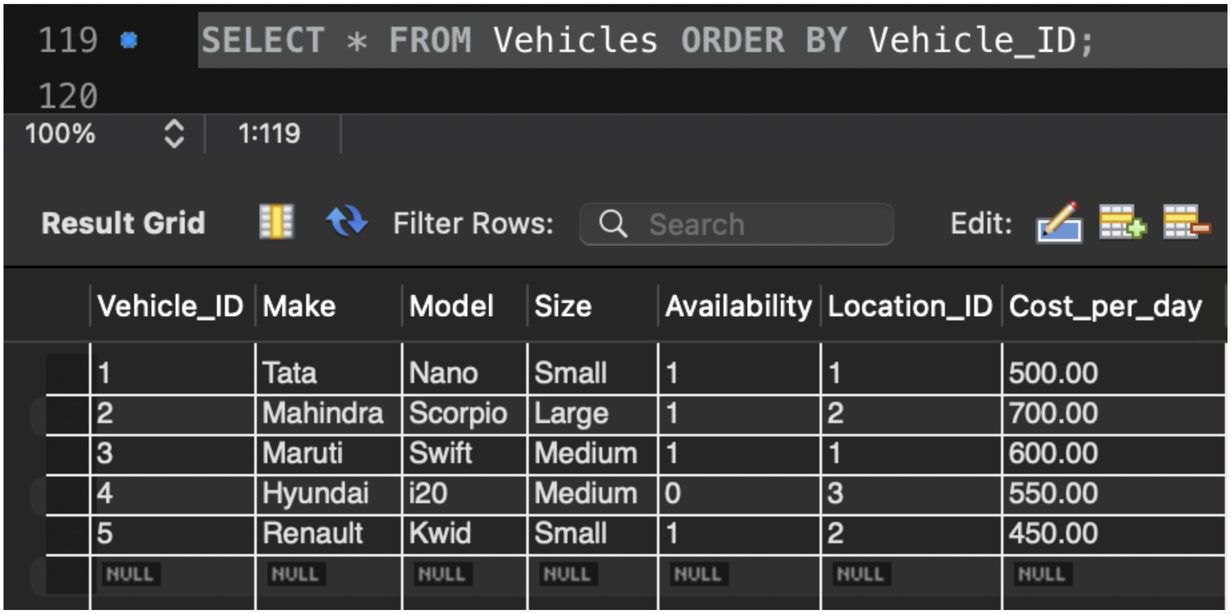
```sql
SELECT * FROM Locations ORDER BY Location_ID;
```

Location_ID	Address	City	Zip_Code
1	1234 Gandhi Street	Mumbai	400001
2	56 Nehru Road	New Delhi	110001
3	789 Bose Avenue	Bangalore	560001
NULL	NULL	NULL	NULL



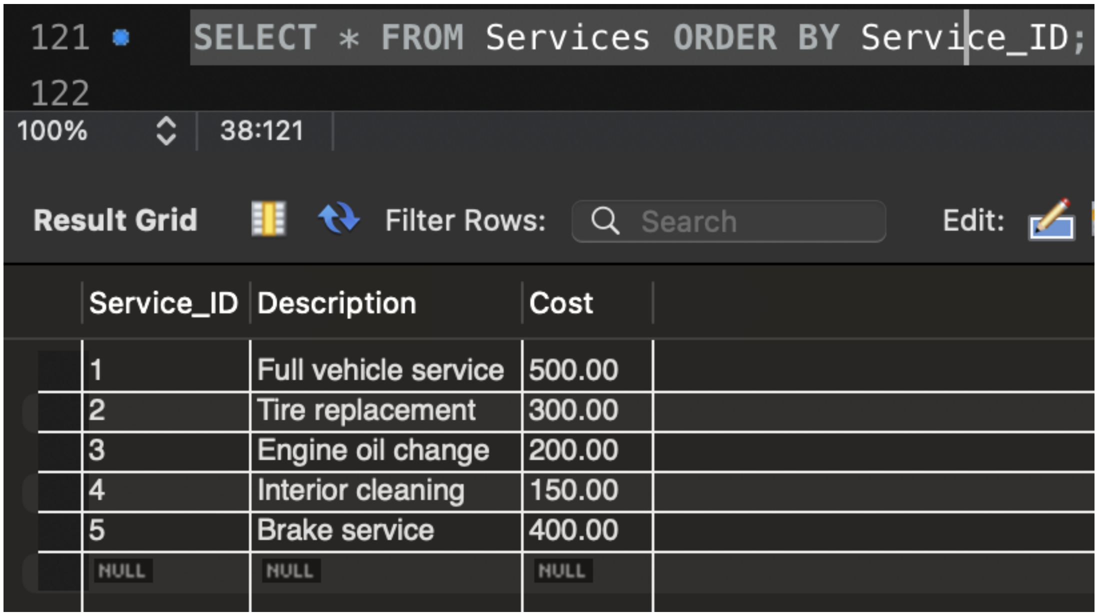
```sql
SELECT * FROM Customers ORDER BY Customer_ID;
```

Customer_ID	Name	Contact_Details
1	Rohit Sharma	rohit.sharma@mail.com
2	Priya Singh	priya.singh@mail.com
3	Anil Kumar	anil.kumar@mail.com
4	Meena Gupta	meena.gupta@mail.com
5	Suresh Raina	suresh.raina@mail.com
NULL	NULL	NULL



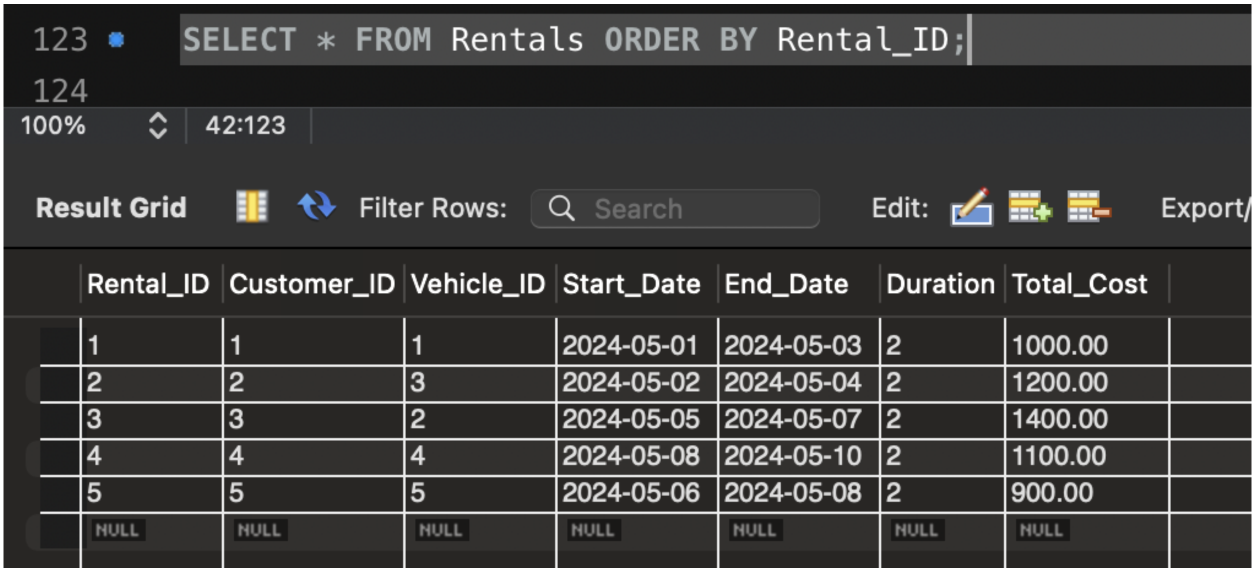
```sql
SELECT * FROM Vehicles ORDER BY Vehicle_ID;
```

Vehicle_ID	Make	Model	Size	Availability	Location_ID	Cost_per_day
1	Tata	Nano	Small	1	1	500.00
2	Mahindra	Scorpio	Large	1	2	700.00
3	Maruti	Swift	Medium	1	1	600.00
4	Hyundai	i20	Medium	0	3	550.00
5	Renault	Kwid	Small	1	2	450.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL



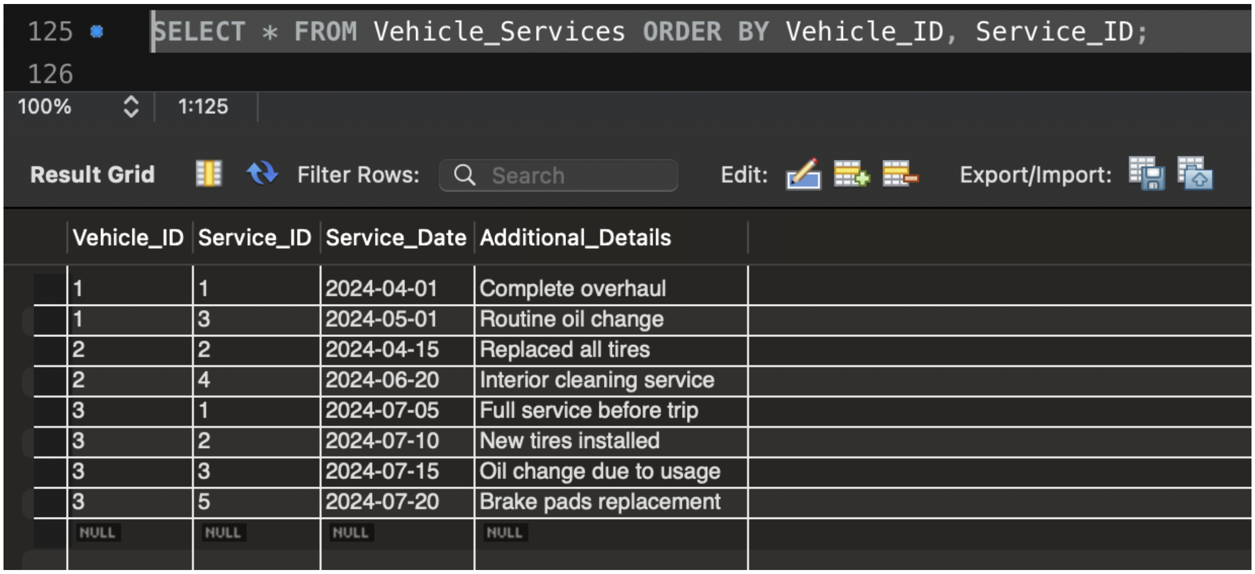
```sql
SELECT * FROM Services ORDER BY Service_ID;
```

Service_ID	Description	Cost
1	Full vehicle service	500.00
2	Tire replacement	300.00
3	Engine oil change	200.00
4	Interior cleaning	150.00
5	Brake service	400.00
NULL	NULL	NULL



```sql
SELECT * FROM Rentals ORDER BY Rental_ID;
```

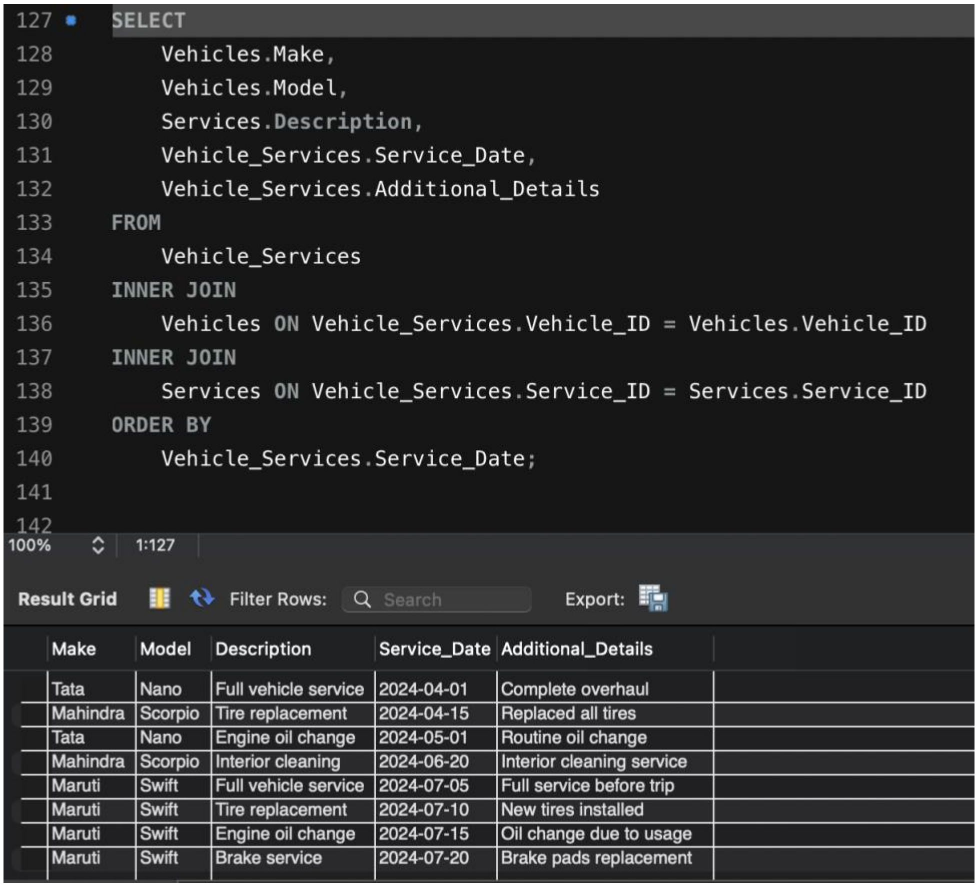
Rental_ID	Customer_ID	Vehicle_ID	Start_Date	End_Date	Duration	Total_Cost
1	1	1	2024-05-01	2024-05-03	2	1000.00
2	2	3	2024-05-02	2024-05-04	2	1200.00
3	3	2	2024-05-05	2024-05-07	2	1400.00
4	4	4	2024-05-08	2024-05-10	2	1100.00
5	5	5	2024-05-06	2024-05-08	2	900.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL



```sql
SELECT * FROM Vehicle_Services ORDER BY Vehicle_ID, Service_ID;
```

Vehicle_ID	Service_ID	Service_Date	Additional_Details
1	1	2024-04-01	Complete overhaul
1	3	2024-05-01	Routine oil change
2	2	2024-04-15	Replaced all tires
2	4	2024-06-20	Interior cleaning service
3	1	2024-07-05	Full service before trip
3	2	2024-07-10	New tires installed
3	3	2024-07-15	Oil change due to usage
3	5	2024-07-20	Brake pads replacement
NULL	NULL	NULL	NULL

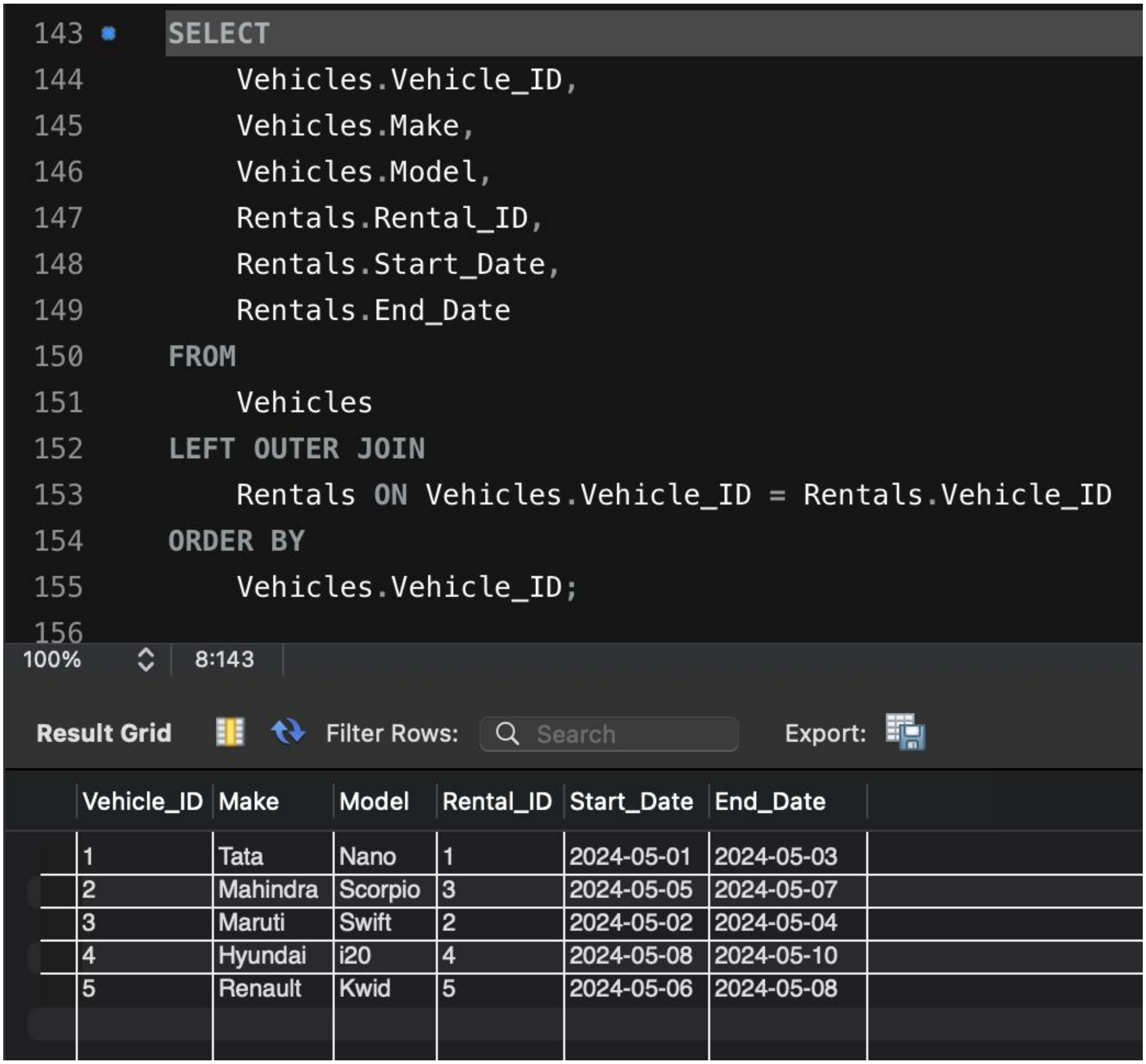
2. Write an SQL involving the junction table and two other related tables. You must use the INNER JOIN to connect with all three tables. The database that you created must be included in your SQL queries.



```sql
SELECT
    Vehicles.Make,
    Vehicles.Model,
    Services.Description,
    Vehicle_Services.Service_Date,
    Vehicle_Services.Additional_Details
FROM
    Vehicle_Services
INNER JOIN
    Vehicles ON Vehicle_Services.Vehicle_ID = Vehicles.Vehicle_ID
INNER JOIN
    Services ON Vehicle_Services.Service_ID = Services.Service_ID
ORDER BY
    Vehicle_Services.Service_Date;
```

Make	Model	Description	Service_Date	Additional_Details
Tata	Nano	Full vehicle service	2024-04-01	Complete overhaul
Mahindra	Scorpio	Tire replacement	2024-04-15	Replaced all tires
Tata	Nano	Engine oil change	2024-05-01	Routine oil change
Mahindra	Scorpio	Interior cleaning	2024-06-20	Interior cleaning service
Maruti	Swift	Full vehicle service	2024-07-05	Full service before trip
Maruti	Swift	Tire replacement	2024-07-10	New tires installed
Maruti	Swift	Engine oil change	2024-07-15	Oil change due to usage
Maruti	Swift	Brake service	2024-07-20	Brake pads replacement

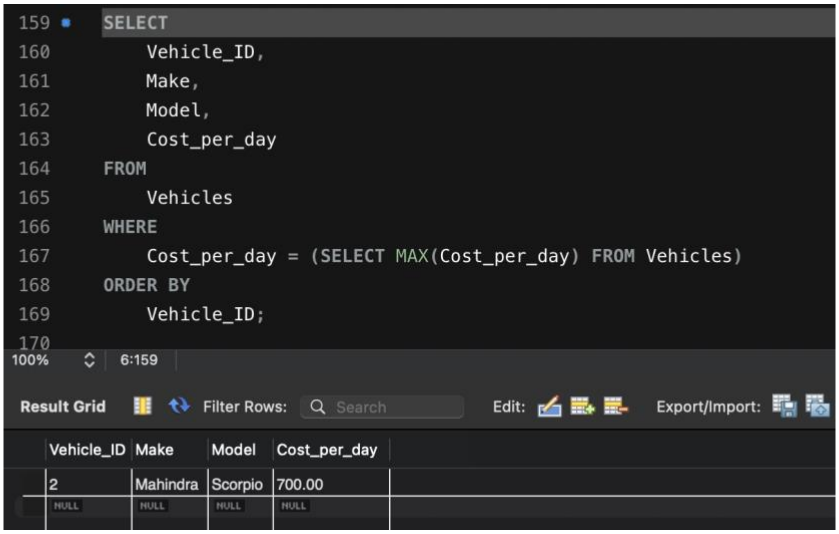
3. Write an SQL by including two or more tables and using the LEFT OUTER JOIN. Show the results and sort the results by key field(s). Interpret the results compared to what an INNER JOIN does.



```
SELECT
    Vehicles.Vehicle_ID,
    Vehicles.Make,
    Vehicles.Model,
    Rentals.Rental_ID,
    Rentals.Start_Date,
    Rentals.End_Date
FROM
    Vehicles
LEFT OUTER JOIN
    Rentals ON Vehicles.Vehicle_ID = Rentals.Vehicle_ID
ORDER BY
    Vehicles.Vehicle_ID;
```

Vehicle_ID	Make	Model	Rental_ID	Start_Date	End_Date
1	Tata	Nano	1	2024-05-01	2024-05-03
2	Mahindra	Scorpio	3	2024-05-05	2024-05-07
3	Maruti	Swift	2	2024-05-02	2024-05-04
4	Hyundai	i20	4	2024-05-08	2024-05-10
5	Renault	Kwid	5	2024-05-06	2024-05-08

4. Write a single-row subquery. Show the results and sort the results by key field(s). Interpret the output.

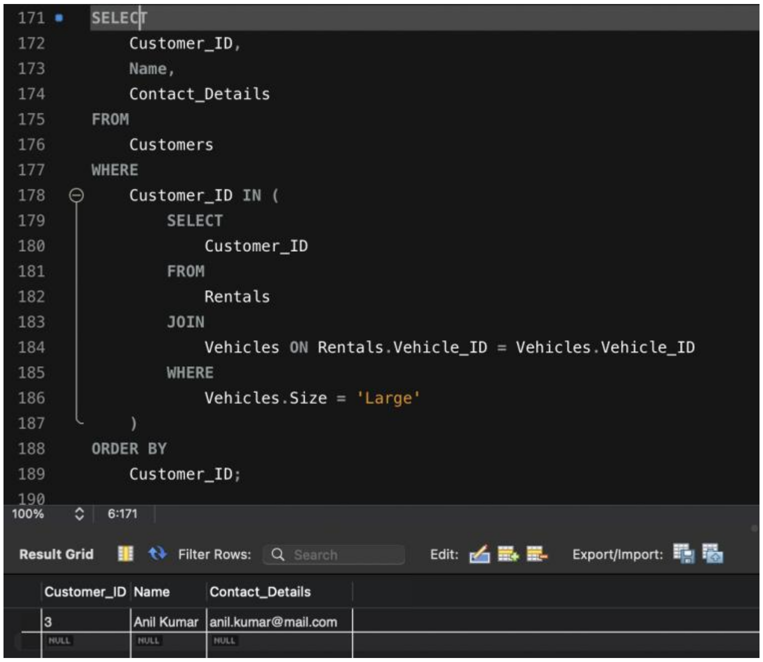


```
SELECT
    Vehicle_ID,
    Make,
    Model,
    Cost_per_day
FROM
    Vehicles
WHERE
    Cost_per_day = (SELECT MAX(Cost_per_day) FROM Vehicles)
ORDER BY
    Vehicle_ID;
```

Vehicle_ID	Make	Model	Cost_per_day
2	Mahindra	Scorpio	700.00
NULL	NULL	NULL	NULL

The SQL query result shows that the most expensive vehicle in the MoveEZ fleet is a Mahindra Scorpio, priced at ₹700.00 per day.

5. Write a multiple-row subquery. Show the results and sort the results by key field(s). Interpret the output.

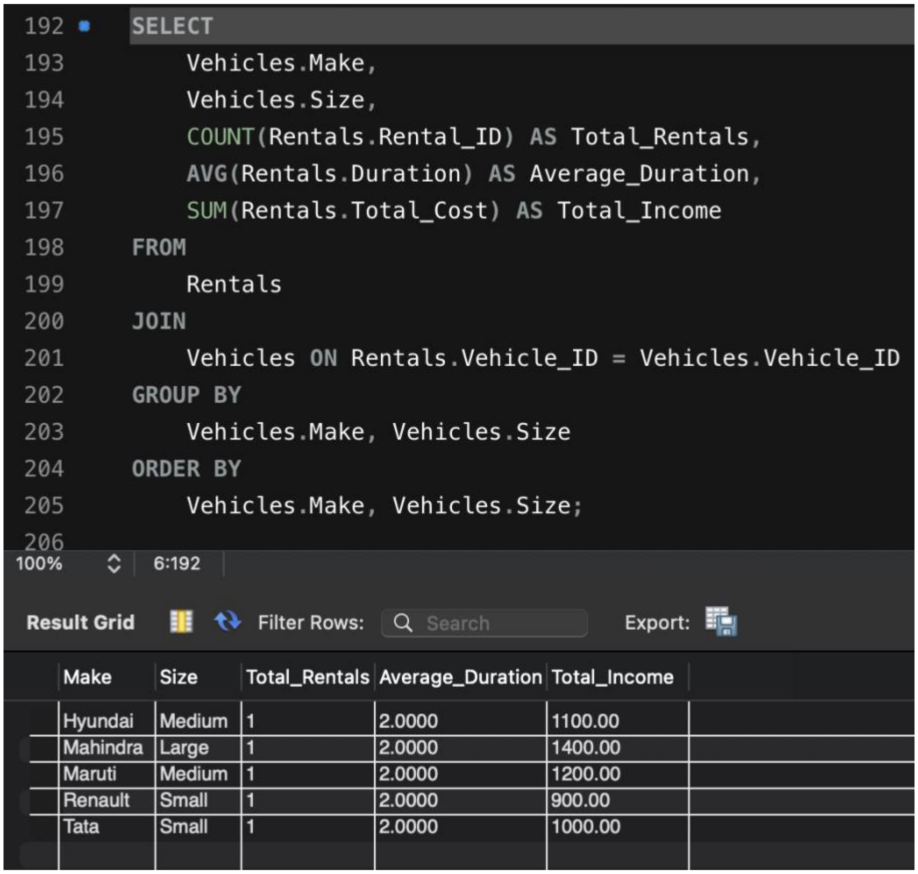


```
SELECT
    Customer_ID,
    Name,
    Contact_Details
FROM
    Customers
WHERE
    Customer_ID IN (
        SELECT
            Customer_ID
        FROM
            Rentals
        JOIN
            Vehicles ON Rentals.Vehicle_ID = Vehicles.Vehicle_ID
        WHERE
            Vehicles.Size = 'Large'
    )
ORDER BY
    Customer_ID;
```

Customer_ID	Name	Contact_Details
3	Anil Kumar	anil.kumar@mail.com
NULL	NULL	NULL

The SQL query result identifies Anil Kumar as a customer who has rented a large-sized vehicle from MoveEZ.

6. Write an SQL to aggregate the results by using multiple columns in the SELECT clause. Interpret the output.

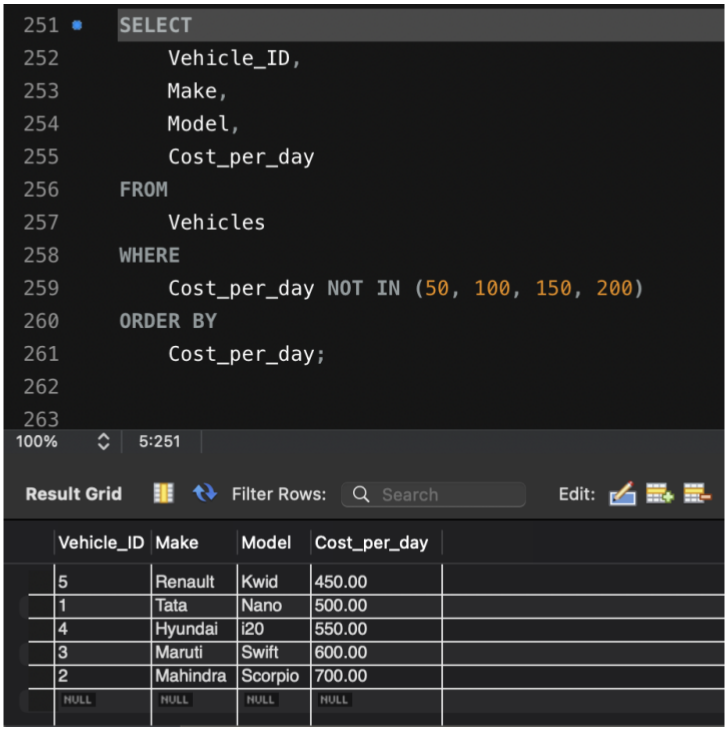


```sql
SELECT
    Vehicles.Make,
    Vehicles.Size,
    COUNT(Rentals.Rental_ID) AS Total_Rentals,
    AVG(Rentals.Duration) AS Average_Duration,
    SUM(Rentals.Total_Cost) AS Total_Income
FROM
    Rentals
JOIN
    Vehicles ON Rentals.Vehicle_ID = Vehicles.Vehicle_ID
GROUP BY
    Vehicles.Make, Vehicles.Size
ORDER BY
    Vehicles.Make, Vehicles.Size;
```

Make	Size	Total_Rentals	Average_Duration	Total_Income
Hyundai	Medium	1	2.0000	1100.00
Mahindra	Large	1	2.0000	1400.00
Maruti	Medium	1	2.0000	1200.00
Renault	Small	1	2.0000	900.00
Tata	Small	1	2.0000	1000.00

The SQL query result displays rental statistics for various vehicle makes and sizes, showing that each vehicle type has been rented once, with an average rental duration of 2 days. The total income from rentals varies by vehicle, with the Mahindra (large size) generating the highest income at ₹1400, indicating a profitable segment in the fleet.

7. Write a subquery using the NOT IN operator. Show the results and sort the results by key field(s). Interpret the output.

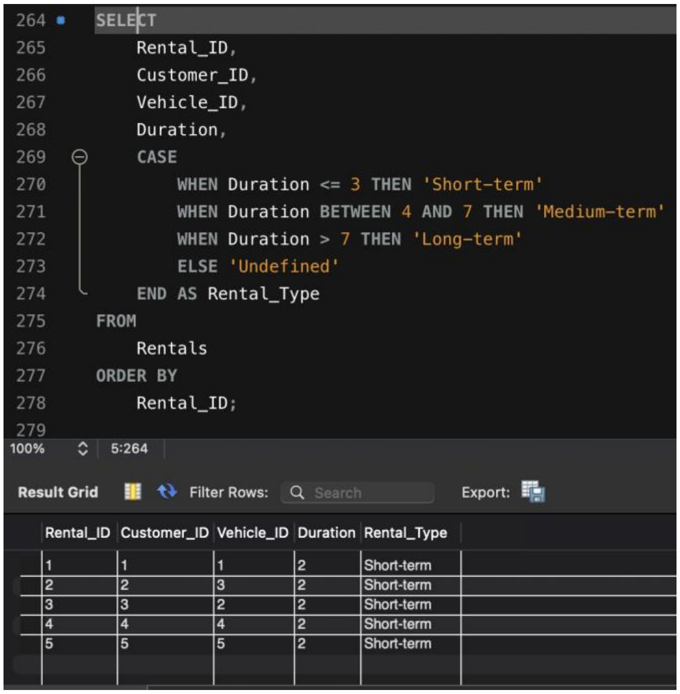


```sql
SELECT
    Vehicle_ID,
    Make,
    Model,
    Cost_per_day
FROM
    Vehicles
WHERE
    Cost_per_day NOT IN (50, 100, 150, 200)
ORDER BY
    Cost_per_day;
```

Vehicle_ID	Make	Model	Cost_per_day
5	Renault	Kwid	450.00
1	Tata	Nano	500.00
4	Hyundai	i20	550.00
3	Maruti	Swift	600.00
2	Mahindra	Scorpio	700.00
NULL	NULL	NULL	NULL

The SQL query output displays vehicles with rental prices that are not 50, 100, 150, or 200, sorted by cost per day. It highlights five vehicles with daily rental rates ranging from ₹450 for the Renault Kwid to ₹700 for the Mahindra Scorpio, suggesting a diverse pricing strategy tailored to different vehicle sizes and customer preferences.

8. Write a query using a CASE statement. Show the results and sort the results by key field(s). Interpret the output.

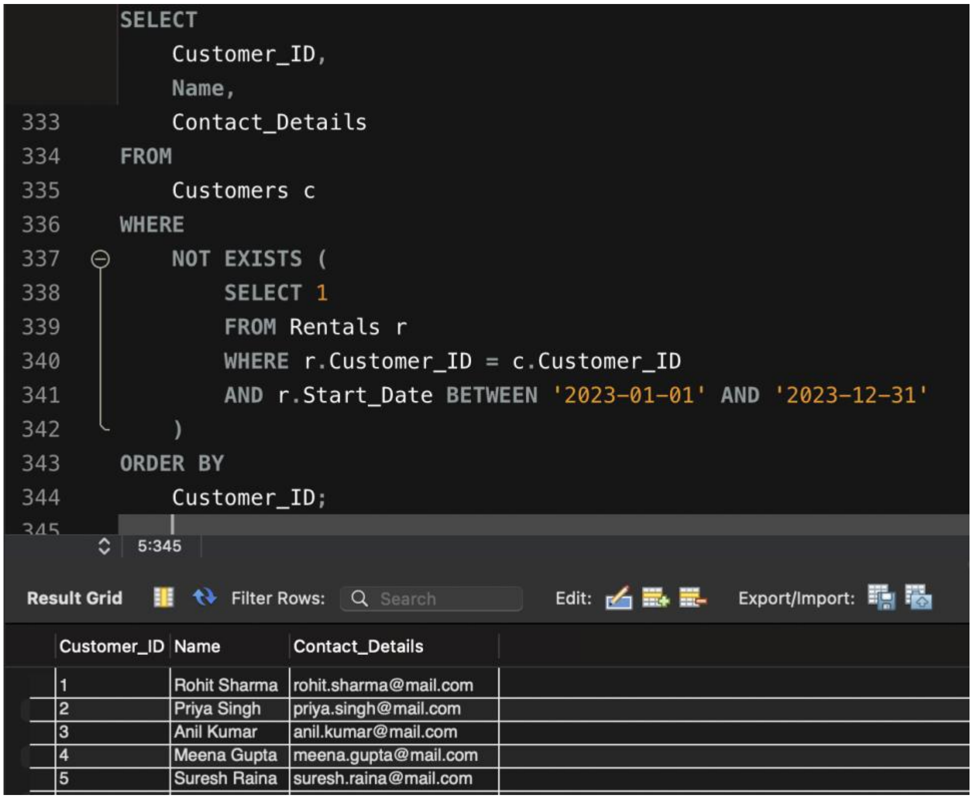


```
SELECT
    Rental_ID,
    Customer_ID,
    Vehicle_ID,
    Duration,
    CASE
        WHEN Duration <= 3 THEN 'Short-term'
        WHEN Duration BETWEEN 4 AND 7 THEN 'Medium-term'
        WHEN Duration > 7 THEN 'Long-term'
        ELSE 'Undefined'
    END AS Rental_Type
FROM
    Rentals
ORDER BY
    Rental_ID;
```

Rental_ID	Customer_ID	Vehicle_ID	Duration	Rental_Type
1	1	1	2	Short-term
2	2	3	2	Short-term
3	3	2	2	Short-term
4	4	4	2	Short-term
5	5	5	2	Short-term

The SQL query output categorizes rentals based on their duration, revealing that all listed rentals fall into the "Short-term" category, which indicates a rental duration of 3 days or fewer. This suggests that shorter rentals are predominant for these specific transactions, possibly indicating a trend or preference among the customer base for brief use of the service.

9. Write a query using the NOT EXISTS operator. Show the results and sort the results by key field(s). Interpret the output.

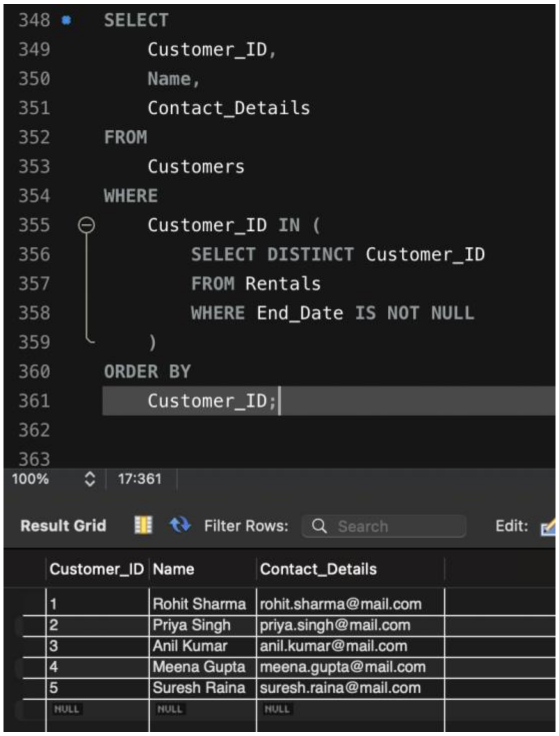


```sql
SELECT
    Customer_ID,
    Name,
    Contact_Details
FROM
    Customers c
WHERE
    NOT EXISTS (
        SELECT 1
        FROM Rentals r
        WHERE r.Customer_ID = c.Customer_ID
        AND r.Start_Date BETWEEN '2023-01-01' AND '2023-12-31'
    )
ORDER BY
    Customer_ID;
```

Customer_ID	Name	Contact_Details
1	Rohit Sharma	rohit.sharma@mail.com
2	Priya Singh	priya.singh@mail.com
3	Anil Kumar	anil.kumar@mail.com
4	Meena Gupta	meena.gupta@mail.com
5	Suresh Raina	suresh.raina@mail.com

The SQL query result identifies all customers who did not make any vehicle rentals in the year 2023, listing their IDs, names, and contact details. It showcases five customers, indicating that these individuals have accounts with MoveEZ but did not engage in rental activities during that specified period, potentially highlighting a segment of inactive or unengaged users.

10. Write a subquery using the NOT NULL operator in the inner query. Show the results and sort the results by key field(s). Interpret the output.



```
SELECT
    Customer_ID,
    Name,
    Contact_Details
FROM
    Customers
WHERE
    Customer_ID IN (
        SELECT DISTINCT Customer_ID
        FROM Rentals
        WHERE End_Date IS NOT NULL
    )
ORDER BY
    Customer_ID;
```

Customer_ID	Name	Contact_Details
1	Rohit Sharma	rohit.sharma@mail.com
2	Priya Singh	priya.singh@mail.com
3	Anil Kumar	anil.kumar@mail.com
4	Meena Gupta	meena.gupta@mail.com
5	Suresh Raina	suresh.raina@mail.com
NULL	NULL	NULL

The SQL query output lists all customers who have completed at least one rental, as indicated by having an End_Date that is not null in the Rentals table. This includes five customers, displaying their IDs, names, and contact details, suggesting these individuals are active users who have engaged with MoveEZ services by successfully completing vehicle rentals.

Summary

The MoveEZ project focused on developing a comprehensive database system to manage a self-service moving and transportation solution in India. The system's design involved creating a structured database to support various operational aspects, including vehicle management, customer interactions, and transaction processing. A significant part of the project was dedicated to designing relational tables to handle data concerning locations, customers, vehicles, services, and rentals, ensuring all necessary business functions were supported by accurate and accessible data.

Through the project, I successfully integrated data on vehicles, customers, and their interactions through rentals and services, reflecting real business scenarios. The project not only demonstrated my ability to design and implement a relational database but also highlighted my skills in writing complex SQL queries to extract meaningful information.

These queries helped in identifying usage patterns, customer behaviors, and operational efficiencies. Overall, the MoveEZ database project provided a robust framework for supporting business operations and strategic decision-making, showcasing the potential to enhance service delivery and customer satisfaction in the competitive market of transportation and moving services in India.