NATIONAL INSTITUTE OF TECHNOLOGY, DURGAPUR

COMPUTER SCIENCE AND ENGINEERING

CSS 453- DATABASE MANAGEMENT SYSTEM LABORATORY

CAR RENTAL SYSTEM

MEMBERS:

1. 23CS8091 - BITTYAM PAUL
2. 23CS8092 - SAI SRIRAM NAYUDU
3. 23CS8093 - SRADDHA UPADHYAY
4. 23CS8094 - SOUVIK SARKAR
5. 23CS8095 - AARABHI MANOJ

INDEX

AIM OF THE PROJECT

➔ To develop a Car Rental System database that efficiently manages records of cars, customers, and rental transactions.

➔ To enable easy retrieval of information related to car availability, rental history, and customer data.

➔ To streamline the process of updating car rental status and calculating earnings.

➔ To generate valuable insights such as the most rented models, overdue rentals, and monthly reports.

PROJECT DESCRIPTION

Our project focuses on designing a structured and efficient Car Rental System using MySQL. It allows for easy storage and retrieval of car, customer, and rental data. The system includes multiple operations such as tracking available cars, updating rental statuses, and identifying high-frequency customers or popular car brands.

We implemented the project entirely using SQL queries and relational database concepts. The structured schema design and use of relational joins helped us explore practical database operations. Through this project, we gained hands-on experience with MySQL commands and understood how real-world systems manage complex data relationships.

SOFTWARE REQUIREMENTS

➔ Windows 11 OS

➔ MySQL Server 8.0

➔ SQL for database queries

➔ Google Docs for documentation

➔ DBMS concepts for relational schema design

SYSTEM IMPLEMENTATION

➔ Lenovo YOGA

➔ HP Pavilion

➔ DELL G15

SYNOPSIS

SL. NO.	TITLE	PURPOSE
1.	Cars Table	Stores details of cars including model, brand, year, daily rental price, and availability status. Key Fields: CarID (PK), Model, Brand, AvailabilityStatus
2.	Customers Table	Contains customer information such as name, email, and phone number. Key Fields: CustomerID (PK), Name, Email, Phone
3.	Rentals Table	Records rental transactions with references to cars and customers, along with rental and return dates. Key Fields: RentalID (PK), CarID (FK), CustomerID (FK), RentalDate, ReturnDate
4.	GetAvailableCars()	Stored Procedure to fetch all cars that are currently available for rent (AvailabilityStatus = 'Available')
5.	CustomerRentalHistory()	Stored Procedure to display all past rentals of a specific customer
6.	MostRentedCarModel()	Shows car models that have been rented most frequently (uses COUNT(*), GROUP BY)

7.	TotalEarnings()	Stored Procedure to calculate the total earnings from all rentals using DATEDIFF and SUM()
8.	FrequentCustomers()	Retrieves customers who have rented cars more than three times.
9.	LongestRental()	Finds the rental with the longest duration.
10.	MostRentedBrand()	Identifies the car brand with the most rentals.
11.	RentalsByMonth()	Generates a report summarizing rentals and earnings by month.
12.	OverdueRentals()	Stored Procedure to list all rentals where the ReturnDate is before the current date
13.	PreventDoubleBooking	Prevents overlapping rentals for the same car.
14.	BeforeRentalInsert	Automatically sets car status to 'Rented' before inserting a rental.
15.	AfterRentalInsertOrUpdate	Updates car status to 'Available' after a rental is inserted and has a return date.
16.	AfterRentalUpdate	Updates car status to 'Available' when a rental's return date is updated.

GitHub REPOSITORY

https://github.com/Souvik-31/Car_Rental_System

SOURCE CODE & OUTPUTS

CREATE TABLES

```
create database project;
use project;

CREATE TABLE Cars (
    CarID INT AUTO_INCREMENT PRIMARY KEY,
    Model VARCHAR(50),
    Brand VARCHAR(50),
    AvailabilityStatus ENUM('Available', 'Rented') DEFAULT
'Available',
    DailyRent INT
);

CREATE TABLE Customers (
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100),
    LicenseNumber VARCHAR(20) UNIQUE,
    Contact VARCHAR(20)
);

CREATE TABLE Rentals (
    RentalID INT AUTO_INCREMENT PRIMARY KEY,
    CarID INT,
    CustomerID INT,
    RentalDate DATE,
    ReturnDate DATE DEFAULT NULL,
    FOREIGN KEY (CarID) REFERENCES Cars(CarID),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

INSERT DATA

```
use project;

INSERT INTO Cars (Model, Brand, AvailabilityStatus, DailyRent) VALUES
('Civic', 'Honda', 'Available', 2500),
('Corolla', 'Toyota', 'Available', 2400),
('Verna', 'Hyundai', 'Available', 1500),
('Accord', 'Honda', 'Available', 2800),
('Camry', 'Toyota', 'Available', 2600),
('XUV700', 'Mahindra', 'Available', 3000),
('Scorpio-N', 'Mahindra', 'Available', 2800),
('Tiago EV', 'Tata', 'Available', 3200),
('Altroz', 'Tata', 'Available', 2200),
('Magnite', 'Nissan', 'Available', 2300),
('Elantra', 'Hyundai', 'Available', 2100),
('Sonata', 'Hyundai', 'Available', 2400),
('BMW 3 Series', 'BMW', 'Available', 7000),
('X5', 'BMW', 'Available', 8500),
('A4', 'Audi', 'Available', 6800),
('Q5', 'Audi', 'Available', 8200),
('C-Class', 'Mercedes-Benz', 'Available', 8000),
('GLC', 'Mercedes-Benz', 'Available', 9000),
('Seltos', 'Kia', 'Available', 2000),
('Tucson', 'Hyundai', 'Available', 2500);

select * from cars;
```

CarID	Model	Brand	AvailabilityStatus	DailyRent
1	Civic	Honda	Available	2500
2	Corolla	Toyota	Available	2400
3	Verna	Hyundai	Available	1500
4	Accord	Honda	Available	2800
5	Camry	Toyota	Available	2600
6	XUV700	Mahindra	Available	3000
7	Scorpio-N	Mahindra	Available	2800
8	Tiago EV	Tata	Available	3200
9	Altroz	Tata	Available	2200
10	Magnite	Nissan	Available	2300
11	Elantra	Hyundai	Available	2100
12	Sonata	Hyundai	Available	2400
13	BMW 3 S...	BMW	Available	7000
14	X5	BMW	Available	8500
15	A4	Audi	Available	6800
16	Q5	Audi	Available	8200
17	C-Class	Mercede...	Available	8000
18	GLC	Mercede...	Available	9000
19	Seltos	Kia	Available	2000
20	Tucson	Hyundai	Available	2500
NULL	NULL	NULL	NULL	NULL

```
INSERT INTO Customers (Name, LicenseNumber, Contact) VALUES
('Bittyam Paul', 'A9876543', '9876543210'),
('Sai Sriram', 'B1234567', '9887654321'),
('Sraddha Upadhyay', 'C2345678', '9898765432'),
('Souvik Sarkar', 'D3456789', '9909876543'),
('Aarabhi Manoj', 'E4567890', '9912345678'),
('Sangeeta Verma', 'F5678901', '9923456789'),
('Ravi Gupta', 'G6789012', '9934567890'),
('Anjali Joshi', 'H7890123', '9945678901'),
('Vikram Mehta', 'I8901234', '9956789012'),
('Pooja Desai', 'J9012345', '9967890123'),
('Rajesh Choudhary', 'K0123456', '9978901234'),
('Deepa Nair', 'L1234567', '9989012345'),
('Siddharth Yadav', 'M2345678', '9990123456'),
('Sneha Iyer', 'N3456789', '9812345670'),
('Karan Kapoor', 'O4567890', '9823456781'),
('Madhavi Rao', 'P5678901', '9834567892'),
('Gaurav Agarwal', 'Q6789012', '9845678903'),
('Shweta Jain', 'R7890123', '9856789014'),
('Akhil Nair', 'S8901234', '9867890125'),
('Ritu Bansal', 'T9012345', '9878901236'),
('Manoj Sharma', 'U0123456', '9889012347'),
('Neeraj Khanna', 'V1234567', '9890123458'),
('Geeta Gupta', 'W2345678', '9901234569'),
('Vishal Soni', 'X3456789', '9912345670'),
('Sonia Thakur', 'Y4567890', '9923456781'),
('Kiran Verma', 'Z5678901', '9934567892'),
('Aarti Pandey', 'AA6789012', '9945678903'),
('Nitin Mishra', 'BB7890123', '9956789014'),
('Manju Yadav', 'CC8901234', '9967890125'),
('Rajeev Kumar', 'DD9012345', '9978901236');

select * from customers;
```

CustomerID	Name	LicenseNumber	Contact
1	Bittyam Paul	A9876543	9876543210
2	Sai Sriram	B1234567	9887654321
3	Sraddha Upadhyay	C2345678	9898765432
4	Souvik Sarkar	D3456789	9909876543
5	Aarabhi Manoj	E4567890	9912345678
6	Sangeeta Verma	F5678901	9923456789
7	Ravi Gupta	G6789012	9934567890
8	Anjali Joshi	H7890123	9945678901
9	Vikram Mehta	I8901234	9956789012
10	Pooja Desai	J9012345	9967890123
11	Rajesh Choudhary	K0123456	9978901234
12	Deepa Nair	L1234567	9989012345
13	Siddharth Yadav	M2345678	9990123456
14	Sneha Iyer	N3456789	9812345670
15	Karan Kapoor	O4567890	9823456781
16	Madhavi Rao	P5678901	9834567892
17	Gaurav Agarwal	Q6789012	9845678903
18	Shweta Jain	R7890123	9856789014
19	Akhil Nair	S8901234	9867890125
20	Ritu Bansal	T9012345	9878901236
21	Manoj Sharma	U0123456	9889012347
22	Neeraj Khanna	V1234567	9890123458
23	Geeta Gupta	W2345678	9901234569
24	Vishal Soni	X3456789	9912345670
25	Sonia Thakur	Y4567890	9923456781
26	Kiran Verma	Z5678901	9934567892
27	Aarti Pandey	AA6789012	9945678903
28	Nitin Mishra	BB7890123	9956789014
29	Manju Yadav	CC8901234	9967890125
30	Rajeev Kumar	DD9012345	9978901236
NULL	NULL	NULL	NULL

```
INSERT INTO Rentals (CarID, CustomerID, RentalDate, ReturnDate)
VALUES
(1, 10, '2025-01-01', '2025-01-05'),
(2, 3, '2025-01-03', '2025-01-07'),
(3, 5, '2025-01-05', '2025-01-09'),
(15, 7, '2025-01-07', '2025-01-11'),
(5, 6, '2025-01-09', '2025-01-13'),
(18, 9, '2025-01-11', '2025-01-15'),
(7, 12, '2025-01-13', '2025-01-17'),
(20, 15, '2025-01-15', '2025-01-25'),
(9, 14, '2025-01-17', '2025-01-21'),
(10, 11, '2025-01-19', '2025-01-23'),
(11, 13, '2025-01-21', '2025-01-25'),
(12, 20, '2025-01-23', '2025-01-27'),
(13, 8, '2025-01-25', '2025-01-29'),
(14, 1, '2025-01-27', '2025-01-31'),
(5, 17, '2025-01-29', '2025-02-02'),
(1, 18, '2025-02-01', '2025-02-05'),
(17, 4, '2025-02-03', '2025-02-07'),
(18, 2, '2025-02-05', '2025-02-09'),
(19, 22, '2025-02-07', '2025-02-11'),
(20, 16, '2025-02-09', '2025-02-13'),
(1, 19, '2025-02-11', '2025-02-15'),
(7, 21, '2025-02-13', '2025-02-17'),
(2, 24, '2025-02-15', '2025-02-19'),
(9, 25, '2025-02-17', '2025-02-21'),
(5, 28, '2025-02-19', '2025-02-23'),
(12, 29, '2025-02-21', '2025-02-25'),
(8, 30, '2025-02-23', '2025-02-27'),
(6, 13, '2025-03-01', '2025-03-05'),
(18, 18, '2025-03-02', '2025-03-06'),
(3, 4, '2025-03-03', '2025-03-07'),
(17, 27, '2025-03-04', '2025-03-08'),
(10, 26, '2025-03-05', '2025-03-09'),
(20, 5, '2025-03-06', '2025-03-10'),
(3, 6, '2025-03-07', '2025-03-11'),
(5, 11, '2025-03-08', '2025-03-12'),
(6, 16, '2025-03-09', '2025-03-13'),
(7, 8, '2025-03-10', '2025-03-14'),
(6, 9, '2025-03-15', '2025-03-20'),
(1, 20, '2025-03-12', '2025-03-16'),
(5, 12, '2025-03-13', '2025-03-17'),
(4, 22, '2025-03-14', '2025-03-18'),
(20, 11, '2025-03-15', '2025-03-19'),
(19, 15, '2025-03-16', '2025-03-20'),
```

```
(12, 1, '2025-03-17', '2025-03-21'),
(20, 11, '2025-03-20', '2025-03-22'),
(19, 30, '2025-03-22', '2025-03-23'),
(7, 17, '2025-03-20', '2025-03-24'),
(9, 21, '2025-03-27', NULL),
(4, 10, '2025-03-30', '2025-04-01'),
(11, 14, '2025-04-04', NULL);

select * from rentals;
```

RentalID	CarID	CustomerID	RentalDate	ReturnDate
1	1	10	2025-01-01	2025-01-05
2	2	3	2025-01-03	2025-01-07
3	3	5	2025-01-05	2025-01-09
4	15	7	2025-01-07	2025-01-11
5	5	6	2025-01-09	2025-01-13
6	18	9	2025-01-11	2025-01-15
7	7	12	2025-01-13	2025-01-17
8	20	15	2025-01-15	2025-01-25
9	9	14	2025-01-17	2025-01-21
10	10	11	2025-01-19	2025-01-23
11	11	13	2025-01-21	2025-01-25
12	12	20	2025-01-23	2025-01-27
13	13	8	2025-01-25	2025-01-29
14	14	1	2025-01-27	2025-01-31
15	5	17	2025-01-29	2025-02-02
16	1	18	2025-02-01	2025-02-05
17	17	4	2025-02-03	2025-02-07
18	18	2	2025-02-05	2025-02-09
19	19	22	2025-02-07	2025-02-11
20	20	16	2025-02-09	2025-02-13
21	1	19	2025-02-11	2025-02-15
22	7	21	2025-02-13	2025-02-17
23	2	24	2025-02-15	2025-02-19
24	9	25	2025-02-17	2025-02-21

RentalID	CarID	CustomerID	RentalDate	ReturnDate
28	6	13	2025-03-01	2025-03-05
29	18	18	2025-03-02	2025-03-06
30	3	4	2025-03-03	2025-03-07
31	17	27	2025-03-04	2025-03-08
32	10	26	2025-03-05	2025-03-09
33	20	5	2025-03-06	2025-03-10
34	3	6	2025-03-07	2025-03-11
35	5	11	2025-03-08	2025-03-12
36	6	16	2025-03-09	2025-03-13
37	7	8	2025-03-10	2025-03-14
38	6	9	2025-03-15	2025-03-20
39	1	20	2025-03-12	2025-03-16
40	5	12	2025-03-13	2025-03-17
41	4	22	2025-03-14	2025-03-18
42	20	11	2025-03-15	2025-03-19
43	19	15	2025-03-16	2025-03-20
44	12	1	2025-03-17	2025-03-21
45	20	11	2025-03-20	2025-03-22
46	19	30	2025-03-22	2025-03-23
47	7	17	2025-03-20	2025-03-24
48	9	21	2025-03-29	NULL
49	4	10	2025-03-30	2025-04-01
50	11	14	2025-04-04	NULL
NULL	NULL	NULL	NULL	NULL

STORED PROCEDURES

```
use project;

-- Stored Procedure: Retrieve all available cars
DELIMITER $$
CREATE PROCEDURE GetAvailableCars()
BEGIN
    SELECT * FROM Cars WHERE AvailabilityStatus = 'Available';
END $$
DELIMITER ;


-- Stored Procedure: List rental history of a specific customer
```

```
DELIMITER $$
CREATE PROCEDURE CustomerRentalHistory(IN customer_id INT)
BEGIN
    SELECT Rentals.RentalID, Cars.Model, Cars.Brand,
Rentals.RentalDate, Rentals.ReturnDate
    FROM Rentals
    JOIN Cars ON Rentals.CarID = Cars.CarID
    WHERE Rentals.CustomerID = customer_id;
END $$
DELIMITER ;


-- Stored Procedure: Find the most rented car model
DELIMITER $$
CREATE PROCEDURE MostRentedCarModel()
BEGIN
    SELECT Cars.Model, COUNT(Rentals.RentalID) AS RentalCount
    FROM Rentals
    JOIN Cars ON Rentals.CarID = Cars.CarID
    GROUP BY Cars.Model
    ORDER BY RentalCount DESC
    LIMIT 1;
END $$
DELIMITER ;


-- Stored Procedure: Calculate total earnings
DELIMITER $$

CREATE PROCEDURE TotalEarnings()
BEGIN
    SELECT SUM(DATEDIFF(r.ReturnDate, r.RentalDate) * c.DailyRent) AS
TotalEarnings
    FROM Rentals r
    JOIN Cars c ON r.CarID = c.CarID
    WHERE r.ReturnDate IS NOT NULL;
END $$
DELIMITER ;


-- Stored Procedure: Retrieve customers who rented more than 3 times
DELIMITER $$
CREATE PROCEDURE FrequentCustomers()
BEGIN
```

```
    SELECT Customers.CustomerID,Customers.Name,
COUNT(Rentals.RentalID) AS RentalCount
    FROM Rentals
    JOIN Customers ON Rentals.CustomerID = Customers.CustomerID
    GROUP BY Customers.CustomerID,Customers.Name
    HAVING RentalCount > 3;
END $$
DELIMITER ;

-- Stored Procedure: Fetch the longest rental duration
DELIMITER $$
CREATE PROCEDURE LongestRental()
BEGIN
    SELECT *, DATEDIFF(ReturnDate, RentalDate) AS RentalDuration
    FROM Rentals
    WHERE ReturnDate IS NOT NULL
    ORDER BY RentalDuration DESC
    LIMIT 1;
END $$
DELIMITER ;


-- Stored Procedure: Identify the most frequently rented car brand
DELIMITER $$
CREATE PROCEDURE MostRentedBrand()
BEGIN
    SELECT Cars.Brand, COUNT(Rentals.RentalID) AS RentalCount
    FROM Rentals
    JOIN Cars ON Rentals.CarID = Cars.CarID
    GROUP BY Cars.Brand
    ORDER BY RentalCount DESC
    LIMIT 1;
END $$
DELIMITER ;


-- Stored Procedure: Generate a report on rentals by month
DELIMITER $$
CREATE PROCEDURE RentalsByMonth()
BEGIN
    SELECT
        DATE_FORMAT(r.RentalDate, '%Y-%m') AS RentalMonth,
        COUNT(*) AS RentalCount,
        SUM(DATEDIFF(r.ReturnDate, r.RentalDate) * c.DailyRent) AS
TotalEarnings
```

```
    FROM Rentals r
    JOIN Cars c ON r.CarID = c.CarID
    WHERE r.ReturnDate IS NOT NULL
    GROUP BY RentalMonth
    ORDER BY RentalMonth;
END $$
DELIMITER ;


-- Stored Procedure: Retrieve customers with overdue rentals
DELIMITER $$
CREATE PROCEDURE OverdueRentals()
BEGIN
    SELECT Customers.Name, Rentals.RentalDate, Rentals.ReturnDate
    FROM Rentals
    JOIN Customers ON Rentals.CustomerID = Customers.CustomerID
    WHERE Rentals.ReturnDate IS NULL AND RentalDate < CURDATE() -
INTERVAL 7 DAY;
END $$
DELIMITER ;
```

TRIGGERS

```
use project;

-- Trigger: In order to prevent double booking of the same vehicle

DELIMITER $$
CREATE TRIGGER PreventDoubleBooking
BEFORE INSERT ON Rentals
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT 1 FROM Rentals
        WHERE CarID = NEW.CarID
        AND (ReturnDate IS NULL OR NEW.RentalDate < ReturnDate)
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Car is already rented or has an
overlapping rental period';
    END IF;
END $$
DELIMITER ;
```

```
-- Trigger: Update car availability when a rental is created

DELIMITER $$
CREATE TRIGGER BeforeRentalInsert
BEFORE INSERT ON Rentals
FOR EACH ROW
BEGIN
    UPDATE Cars SET AvailabilityStatus = 'Rented' WHERE CarID =
NEW.CarID;
END $$
DELIMITER ;


-- Trigger: Update car availability when a rental is returned

DELIMITER $$
CREATE TRIGGER AfterRentalInsertOrUpdate
AFTER INSERT ON Rentals
FOR EACH ROW
BEGIN
    IF NEW.ReturnDate IS NOT NULL THEN
        UPDATE Cars SET AvailabilityStatus = 'Available' WHERE CarID
= NEW.CarID;
    END IF;
END $$
DELIMITER ;
DELIMITER $$
CREATE TRIGGER AfterRentalUpdate
AFTER UPDATE ON Rentals
FOR EACH ROW
BEGIN
    IF NEW.ReturnDate IS NOT NULL THEN
        UPDATE Cars SET AvailabilityStatus = 'Available' WHERE CarID
= NEW.CarID;
    END IF;
END $$
DELIMITER ;
```

QUERIES

```
use project;
```

-- Retrieve all available cars

```
SELECT * FROM Cars WHERE AvailabilityStatus = 'Available';
```

CarID	Model	Brand	AvailabilityStatus	DailyRent
1	Civic	Honda	Available	2500
2	Corolla	Toyota	Available	2400
3	Verna	Hyundai	Available	1500
4	Accord	Honda	Available	2800
5	Camry	Toyota	Available	2600
6	XUV700	Mahindra	Available	3000
7	Scorpio-N	Mahindra	Available	2800
8	Tiago EV	Tata	Available	3200
10	Magnite	Nissan	Available	2300
12	Sonata	Hyundai	Available	2400
13	BMW 3 S...	BMW	Available	7000
14	X5	BMW	Available	8500
15	A4	Audi	Available	6800
16	Q5	Audi	Available	8200
17	C-Class	Mercede...	Available	8000
18	GLC	Mercede...	Available	9000
19	Seltos	Kia	Available	2000
20	Tucson	Hyundai	Available	2500
NULL	NULL	NULL	NULL	NULL

-- List rental history of a specific customer

```
SELECT Rentals.RentalID, Cars.Model, Cars.Brand, Rentals.RentalDate,
Rentals.ReturnDate
FROM Rentals
JOIN Cars ON Rentals.CarID = Cars.CarID
WHERE Rentals.CustomerID = 1;
```

RentalID	Model	Brand	RentalDate	ReturnDate
14	X5	BMW	2025-01-27	2025-01-31
44	Sonata	Hyundai	2025-03-17	2025-03-21

-- Find the most rented car model

```
SELECT Cars.Model, COUNT(Rentals.RentalID) AS RentalCount
FROM Rentals
```

```
JOIN Cars ON Rentals.CarID = Cars.CarID
GROUP BY Cars.Model
ORDER BY RentalCount DESC
LIMIT 1;
```

	Model	RentalCount
▶	Tucson	5

-- Calculate total earnings from car rentals

```
SELECT SUM(DATEDIFF(r.ReturnDate, r.RentalDate) * c.DailyRent) AS
TotalEarnings
    FROM Rentals r
    JOIN Cars c ON r.CarID = c.CarID
    WHERE r.ReturnDate IS NOT NULL;
```

	TotalEarnings
▶	655000

-- Retrieve customers who rented more than 3 times

```
SELECT Customers.Name, COUNT(Rentals.RentalID) AS RentalCount
FROM Rentals
JOIN Customers ON Rentals.CustomerID = Customers.CustomerID
GROUP BY Customers.Name
HAVING RentalCount > 3;
```

	CustomerID	Name	RentalCount
▶	11	Rajesh Choudhary	4

-- Fetch the longest rental duration

```
SELECT *, DATEDIFF(ReturnDate, RentalDate) AS RentalDuration
FROM Rentals
WHERE ReturnDate IS NOT NULL
ORDER BY RentalDuration DESC
LIMIT 1;
```

	RentalID	CarID	CustomerID	RentalDate	ReturnDate	RentalDuration
▶	8	20	15	2025-01-15	2025-01-25	10

-- Identify the most frequently rented car brand

```
SELECT Cars.Brand, COUNT(Rentals.RentalID) AS RentalCount
FROM Rentals
JOIN Cars ON Rentals.CarID = Cars.CarID
GROUP BY Cars.Brand
ORDER BY RentalCount DESC
LIMIT 1;
```

Brand	RentalCount
Hyundai	13

-- Generate a report on rentals by month

```
SELECT DATE_FORMAT(r.RentalDate, '%Y-%m') AS RentalMonth,
        COUNT(*) AS RentalCount,
        SUM(DATEDIFF(r.ReturnDate, r.RentalDate) * c.DailyRent) AS
TotalEarnings
FROM Rentals r
JOIN Cars c ON r.CarID = c.CarID
WHERE r.ReturnDate IS NOT NULL
GROUP BY RentalMonth
ORDER BY RentalMonth;
```

RentalMonth	RentalCount	TotalEarnings
2025-01	15	243800
2025-02	12	168400
2025-03	21	242800

-- Retrieve customers with overdue rentals

```
SELECT Customers.Name, Rentals.RentalDate, Rentals.ReturnDate
FROM Rentals
JOIN Customers ON Rentals.CustomerID = Customers.CustomerID
WHERE Rentals.ReturnDate IS NULL AND RentalDate < CURDATE() -
INTERVAL 7 DAY;
```

Name	RentalDate	ReturnDate
Manoj Sharma	2025-03-27	NULL

REFERENCES

➔ CSC404: *Database Management System – Lecture Notes*
Faculty: Prof. Prasenjit Choudhury, Department of Computer Science & Engineering

➔ TutorialsPoint, *SQL - Triggers*. Retrieved from https://www.tutorialspoint.com/mysql/mysql-triggers.htm

➔ W3Schools, *SQL Stored Procedures*. Retrieved from https://www.w3schools.com/sql/sql_stored_procedures.asp

➔ GeeksforGeeks, *SQL | Triggers*. Retrieved from https://www.geeksforgeeks.org/sql-triggers/