****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SQL MINI PROJECT REPORT**

**ON**

**“VEHICLE RENTAL SYSTEM”**

**SUBJECT: DATABASE MANAGEMENT SYSTEMS**

**ACADEMIC YEAR: 2024 - 2025**

**SUBMITTED BY: GROUP 8**

|  |  |
| --- | --- |
| **ROLL NO** | **NAME** |
| 23CS8036 | JIT SARKAR |
| 23CS8037 | SAYAN PRAMANIK |
| 23CS8038 | MADDU JAYASREE |
| 23CS8039 | DASARI KEERTHANA |
| 23CS8040 | NARAYAN PAUL |

**SUBMITTED TO :**

**PROF. PRASENJIT CHOUDHURY**

**📘Acknowledgement:**

We would like to express our sincere gratitude to **Prof. Prasenjit Choudhury**, our respected faculty for Database Management Systems, for his continuous guidance, encouragement, and valuable feedback throughout the course of this mini-project. We also extend our appreciation to our department, friends, and peers who directly or indirectly supported us in completing **the “Vehicle Rental System”** project. Lastly, we thank all the group members for their dedication, teamwork, and enthusiastic contribution to this successful completion.

**📖Introduction:**

A database-driven initiative, the "Vehicle Rental System" allows efficient management of rental service vehicles, customers, and rental transactions. Using structured data operations including table creation, data manipulation, stored procedures, and triggers, this project aims to show the practical use of SQL in real-world situations.  
  
Common tasks like vehicle availability, booking, and revenue tracking are handled by this system in a simplified approach. It has been created for the academic year 2024–2025 as part of the Database Management Systems course.

📊 **Tools & Technologies Used:**

* Database: MySQL
* Language: SQL
* Platform: MySQL Workbench / MySQL 8.0 Command Line Client
* OS: Windows

**GitHub Link 🔗:** [**https://github.com/narayan123git/Vehicle-Rental-System-**](https://github.com/narayan123git/Vehicle-Rental-System-)

**📈 Learning Outcomes:**

Through this project, we gained hands-on experience in:

* Designing relational database schemas.
* Writing and executing complex SQL queries.
* Implementing stored procedures and triggers.
* Understanding data integrity, relationships, and query optimization.

**TABLE CREATION**

CREATE TABLE Vehicles (

VehicleID INT PRIMARY KEY,

Model VARCHAR(50) NOT NULL,

Type VARCHAR(50) NOT NULL,

PricePerDay DECIMAL(10,2) NOT NULL,

AvailabilityStatus VARCHAR(20) NOT NULL

);

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Contact VARCHAR(50),

LicenseNumber VARCHAR(50)

);

CREATE TABLE Rentals (

RentalID INT PRIMARY KEY,

CustomerID INT,

VehicleID INT,

RentDate DATE,

ReturnDate DATE,

TotalCost DECIMAL(10,2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (VehicleID) REFERENCES Vehicles(VehicleID)

);

**INSERTION OF DATA**

INSERT INTO Vehicles (VehicleID, Model, Type, PricePerDay, AvailabilityStatus) VALUES

(1, 'Tata Nexon', 'SUV', 2500.00, 'Available'),

(2, 'Maruti Swift', 'Comfort', 1200.00, 'Available'),

(3, 'Hyundai Verna', 'Coupe', 1800.00, 'Available'),

(4, 'Mahindra Thar', 'Off-Road', 3500.00, 'Available'),

(5, 'Tata Motors', 'MPV', 3000.00, 'Available'),

(6, 'Tata Altroz', 'SUV', 2200.00, 'Not Available'),

(7, 'Honda City', 'Coupe', 2000.00, 'Available'),

(8, 'Maruti Ertiga', 'MPV', 1600.00, 'Available'),

(9, 'MG Hector', 'SUV', 2800.00, 'Not Available'),

(10, 'Hyundai i20', 'Comfort', 1400.00, 'Available');

INSERT INTO Customers (CustomerID, Name, Contact, LicenseNumber) VALUES

(101, 'Rajesh Kumar', '8765432109', 'DL-0420110034567'),

(202, 'Priya Sharma', '9876543210', 'MH-0320150076543'),

(303, 'Vikram Singh', '7654321098', 'KA-0720180089012'),

(404, 'Ananya Patel', '9988776655', 'GJ-0520190123456'),

(505, 'Suresh Reddy', '8877665544', 'AP-0620170045678'),

(606, 'Meena Verma', '9871234769', 'UP-0920160056789'),

(707, 'Arjun Kapoor', '7766554433', 'PB-0320130067890'),

(808, 'Deepika Malhotra', '9900112233', 'HR-0820140078901'),

(909, 'Rahul Gupta', '8612908747', 'TN-0220180089012'),

(1010, 'Nisha Joshi', '8899001122', 'MP-1020170090123');

INSERT INTO Rentals (RentalID, CustomerID, VehicleID, RentDate, ReturnDate, TotalCost) VALUES

(1001, 101, 1, '2025-03-01', '2025-03-05', 10000.00),

(2002, 202, 2, '2023-03-10', '2024-03-12', 2400.00),

(3003, 303, 3, '2020-03-15', '2021-03-18', 5400.00),

(4004, 404, 3, '2021-03-20', '2024-03-25', 15000.00),

(5005, 505, 4, '2022-04-01', '2025-04-03', 7000.00),

(6006, 606, 7, '2022-04-05', '2023-04-10', 10000.00),

(7007, 707, 8, '2020-04-12', '2025-04-14', 3200.00),

(7256, 101, 8, '2022-04-12', '2024-04-14', 3300.00),

(8008, 808, 10, '2021-04-18', '2023-04-22', 5600.00),

(7258, 101, 9, '2022-04-12', '2023-04-14', 3300.00),

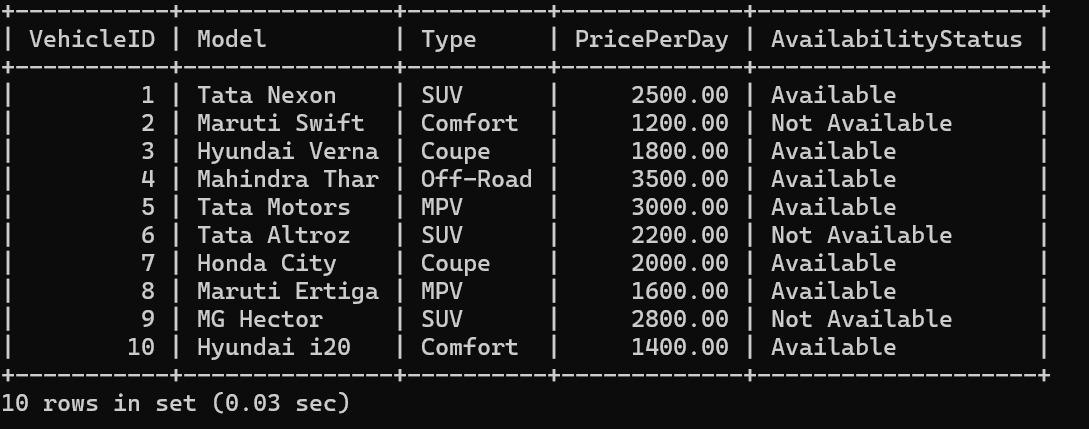
(7265, 101, 6, '2020-04-12', '2024-04-14', 3300.00),

(9009, 909, 1, '2021-05-01', '2025-05-05', 10000.00),

(10010, 1010, 3, '2025-04-01', '2025-04-03', 3600.00);

**TABLES**

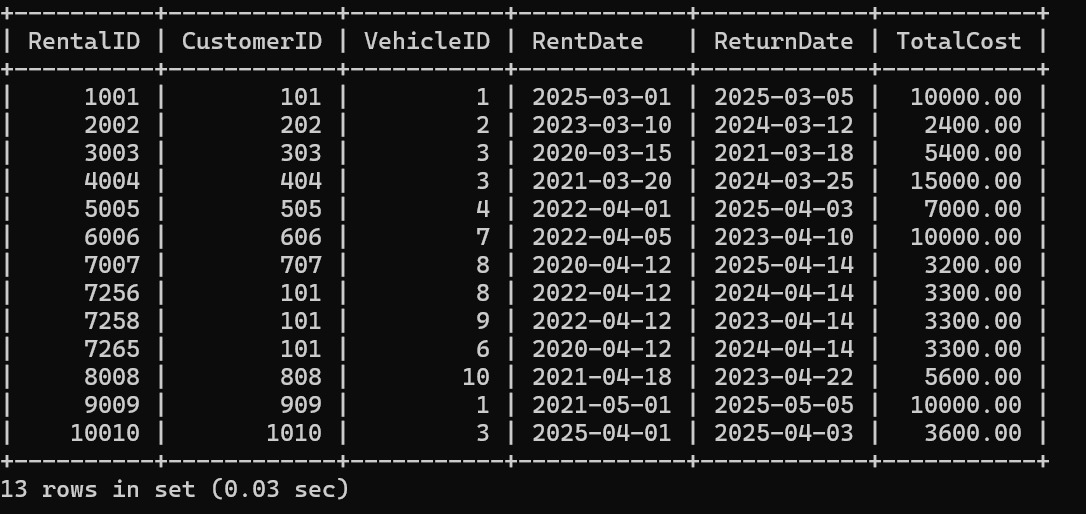
**VEHICLES**

****

**CUSTOMERS**

****

**RENTALS**

****

**STORED PROCEDURES**

**1.** DELIMITER //

CREATE PROCEDURE GetAvailableVehicles()

BEGIN

SELECT \* FROM Vehicles WHERE AvailabilityStatus = 'Available';

END;

//

DELIMITER;

**2.** DELIMITER //

CREATE PROCEDURE GetRentalsByCustomer(IN cid INT)

BEGIN

SELECT \* FROM Rentals WHERE CustomerID = cid;

END;

//

DELIMITER;

**3.** DELIMITER //

CREATE PROCEDURE TotalRevenue()

BEGIN

SELECT SUM(TotalCost) AS TotalRevenue FROM Rentals;

END;

//

DELIMITER;

**4.** DELIMITER //

CREATE PROCEDURE CustomersWithManyBookings()

BEGIN

SELECT CustomerID, COUNT(\*) AS TotalBookings

FROM Rentals

GROUP BY CustomerID

HAVING COUNT(\*) > 3;

END;

//

DELIMITER;

**5.** DELIMITER //

CREATE PROCEDURE SetVehicleUnavailable(IN vid INT)

BEGIN

UPDATE Vehicles SET AvailabilityStatus = 'Not Available' WHERE VehicleID = vid;

END;

//

DELIMITER;

**6.** DELIMITER //

CREATE PROCEDURE RecentRentals()

BEGIN

SELECT \* FROM Rentals

WHERE RentDate >= CURDATE() - INTERVAL 30 DAY;

END;

//

DELIMITER;

**7.** DELIMITER //

CREATE PROCEDURE DeleteOldRentals()

BEGIN

DELETE FROM Rentals WHERE RentDate < CURDATE() - INTERVAL 1 YEAR;

END;

//

DELIMITER;

**8.** DELIMITER //

CREATE PROCEDURE MostRentedVehicles()

BEGIN

SELECT VehicleID, COUNT(\*) AS TimesRented

FROM Rentals

GROUP BY VehicleID

ORDER BY TimesRented DESC;

END;

//

DELIMITER;

**9.** DELIMITER //

CREATE PROCEDURE TripLengths()

BEGIN

SELECT CustomerID, DATEDIFF(ReturnDate, RentDate) AS TripLength

FROM Rentals

ORDER BY TripLength DESC;

END;

//

DELIMITER;

**10.** DELIMITER //

CREATE PROCEDURE VehicleEarnings()

BEGIN

SELECT v.VehicleID, v.Model, SUM(r.TotalCost) AS TotalEarnings

FROM Vehicles v

JOIN Rentals r ON v.VehicleID = r.VehicleID

GROUP BY v.VehicleID, v.Model

ORDER BY TotalEarnings DESC;

END;

//

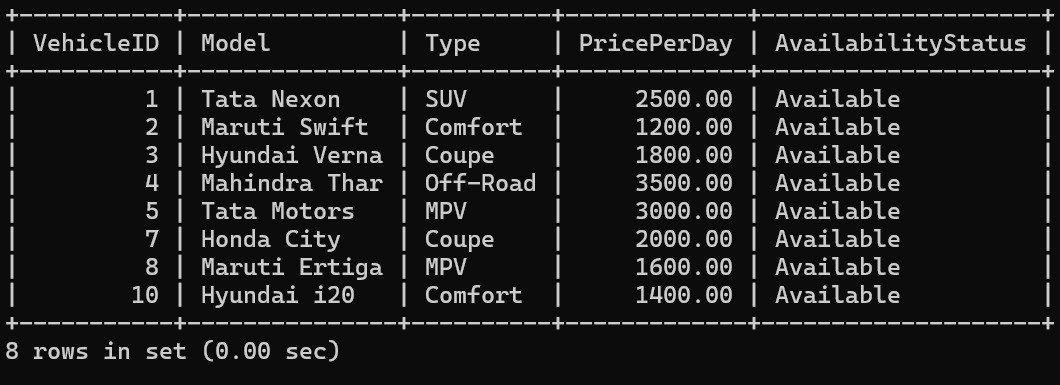
DELIMITER ;

**QUERIES**

**1.LIST OF ALL AVAILABLE VEHICLES**

CALL GetAvailableVehicles();

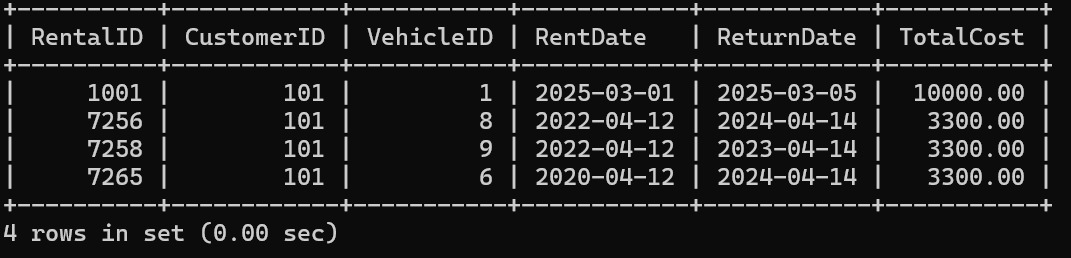
**OUTPUT:**



**2.GET RENTAL DETAILS OF A SPECIFIC CUSTOMER**

CALL GetRentalsByCustomer(101);

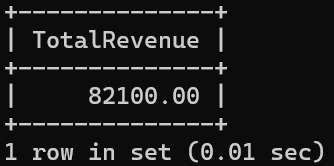
**OUTPUT:**



**3.CALCULATE TOTAL REVENUE FROM RENTALS**

CALL TotalRevenue();

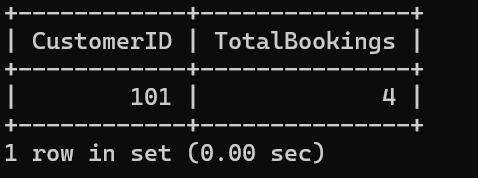
**OUTPUT:**



**4.FIND CUSTOMERS WHO RENTED MORE THAN 3 TIMES**

CALL CustomersWithManyBookings();

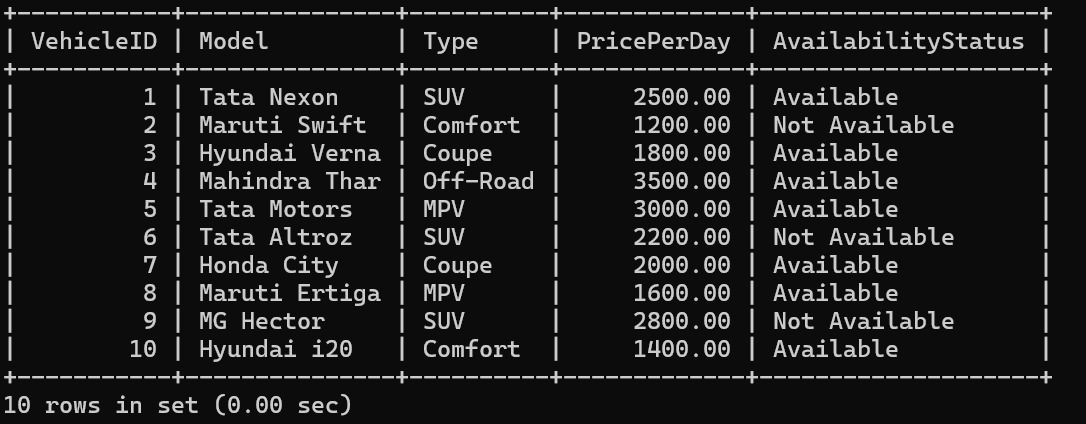
**OUTPUT:**



**5.UPDATE VEHICLE AVAILABILTY AFTER A RENTAL**

CALL SetVehicleUnavailable(2);

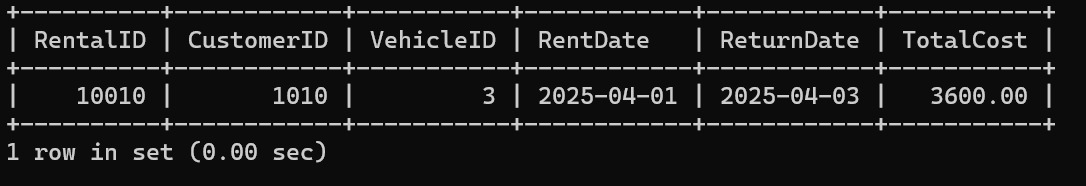
**OUTPUT:**



**6.RETRIEVE RENTAL RECORDS FOR THE LAST 30 DAYS**

CALL RecentRentals();

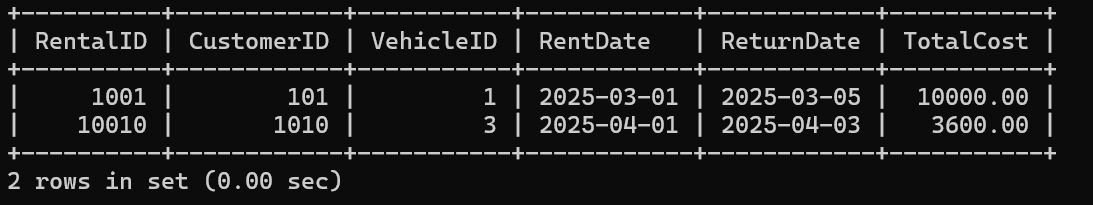
**OUTPUT:**



**7.DELETE OLD RENTAL RECORDS OLDER THAN A YEAR**

CALL DeleteOldRentals();

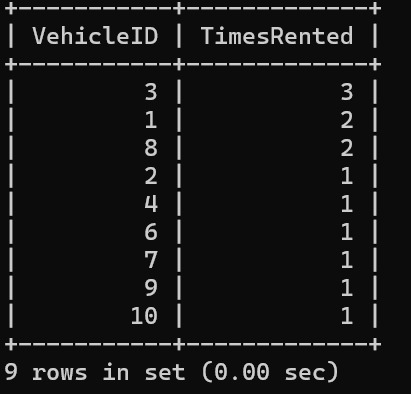
**OUTPUT:**



**8.IDENTIFY MOST FREQUENTLY RENTED VEHICLES**

CALL MostRentedVehicles();

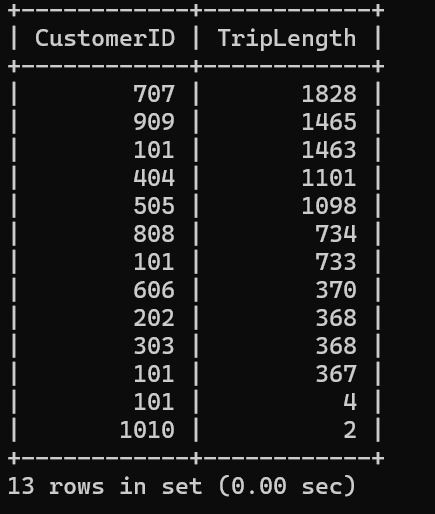
**OUTPUT:**



**9.FIND CUSTOMERS WITH THE LONGEST RENTAL DURATIONS**

CALL TripLengths();

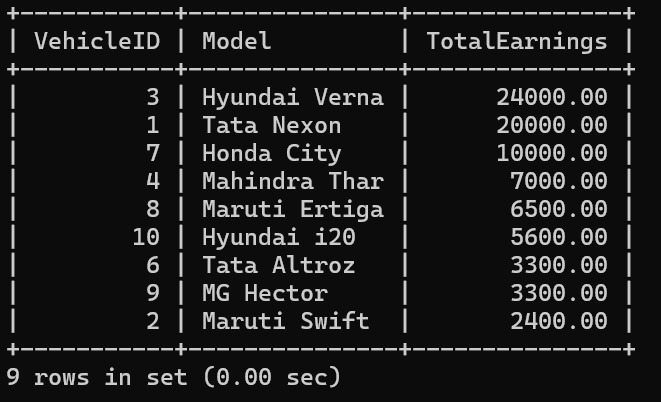
**OUTPUT:**



**10.RETRIEVE VEHICLES THAT GENERATE THE HIGHEST REVENUE**

CALL VehicleEarnings();

**OUTPUT:**



**TRIGGERS**

**-- Trigger for setting the availability status after renting a vehicle**

**1.** DELIMITER //

CREATE TRIGGER UpdateAvailabilityAfterRental

AFTER INSERT ON Rentals

FOR EACH ROW

BEGIN

UPDATE Vehicles

SET AvailabilityStatus = 'Not Available'

WHERE VehicleID = NEW.VehicleID;

END;

//

DELIMITER;

**-- Trigger for setting the availability status after returning(updating return date in Vehicles table) a vehicle**

**2.** DELIMITER //

CREATE TRIGGER UpdateAvailabilityOnReturn

AFTER UPDATE ON Rentals

FOR EACH ROW

BEGIN

IF NEW.ReturnDate IS NOT NULL THEN

UPDATE Vehicles

SET AvailabilityStatus = 'Available'

WHERE VehicleID = NEW.VehicleID;

END IF;

END;

//

DELIMITER ;

**Conclusion:**

The Vehicle Rental System project gave us practical exposure to database design and management. By working through real-world scenarios, we explored how SQL can be leveraged to create scalable and reliable systems. This experience has strengthened our understanding of DBMS concepts and their applications.