CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT
ON
Vehicle Management System

Program Name: Bachelors of Computer Application(Data Science)

Subject Name/Code: Database Management System / (24 CAP-204)

Submitted by:

Name: Varun Singh

UID: 24BCD10020

Section: 24BCD-1(A)

Submitted to:

Name: Mr. Suraj Prakash

Designation:

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

1. Aim of the Project:

To design and implement a comprehensive Database Management System for managing vehicle service operations, tracking customer vehicles, and maintaining complete service history records.

2.Abstract:

The Vehicle Service Management System (VSMS) is a relational database project designed to streamline the operations of vehicle service centers. The system efficiently manages customers, their vehicles, service records, appointments, mechanics, and billing in a centralized database. It eliminates redundant manual record-keeping by maintaining structured and interrelated data, ensuring quick retrieval and accurate tracking of vehicle maintenance history.

The database design follows a systematic approach beginning with entity identification, E–R modeling, and normalization up to the Third Normal Form (3NF) to eliminate data anomalies. The implementation uses SQL with proper constraints such as Primary Keys, Foreign Keys, Unique, and *Check* for data integrity. Queries are written using SELECT, JOIN, GROUP BY, HAVING, and VIEW statements to generate analytical reports like customer service history, total revenue, mechanic performance, and upcoming appointments.

This project demonstrates how relational database concepts can be applied to real-world scenarios to improve service management, reduce human error, and enhance decision-making in automotive service centers. The system serves as a reliable and scalable solution for organizing vehicle maintenance data effectively.

2. Objective:

- Create normalized database schema for vehicle service management
- Implement relationships between customers, vehicles, and service records
- Develop stored procedures for service operations

- Create views for service history and reporting
- Ensure data integrity through constraints and relationships

3. Tools and Technologies Used:

- SQL Server Database
- SQL for database operations
- Command Line Interface/SQL Server Management Studio

4. System Requirements:

- SQL Server 2012 or above
- Minimum 2GB RAM
- 500MB free disk space
- Windows OS

Phase 1 — Problem identification

Core requirements (from your brief):

Track customers, vehicles, and service history. Also need billing, mechanics, spare parts, appointments, and service types.

Entities (main):

- Customer — person who owns vehicles.
- Vehicle — each vehicle belonging to customer(s).
- Service — a service event (repair/maintenance) performed on a vehicle.
- Mechanic — technician who performs services.
- ServiceType — classification (Oil change, Brake repair, Inspection...).
- Part — spare parts used in services.
- Service_Part — many-to-many between Service and Part (quantity, price).

- Invoice — billing record for a Service (could be multiple services per invoice but here 1-to-1 for simplicity).
- Appointment — scheduled service slot (optional).

Relationships (short):

- Customer 1 — * N Vehicle (one customer can have many vehicles)
- Vehicle 1 — * N Service (one vehicle can have many service records)
- Service * — 1 Mechanic (a service is performed by one mechanic; a mechanic performs many services)
- Service * — 1 ServiceType (classifies service)
- Service * — * Part (via Service_Part)
- Service 1 — 0..1 Invoice (service may be billed)
- Customer 1 — * Appointment; Vehicle may be linked to Appointment.

Phase 2 — Database Design & Normalization

Relational schema (high-level):

- Customer(customer_id PK, first_name, last_name, phone UNIQUE, email UNIQUE, address, registration_date)
- Vehicle(vehicle_id PK, customer_id FK, make, model, year, vin UNIQUE, reg_no UNIQUE)
- Mechanic(mechanic_id PK, first_name, last_name, phone, email, hire_date)
- ServiceType(service_type_id PK, name UNIQUE, description, base_price)
- Service(service_id PK, vehicle_id FK, mechanic_id FK, service_type_id FK, service_date, odometer, notes, status, total_cost)
- Part(part_id PK, part_name, part_code UNIQUE, unit_price, stock_qty)

- Service_Part(service_id FK, part_id FK, quantity, unit_price, PRIMARY KEY (service_id, part_id))
- Invoice(invoice_id PK, service_id FK UNIQUE, invoice_date, total_amount, paid_amount, payment_status)
- Appointment(appointment_id PK, customer_id FK, vehicle_id FK, appointment_date, slot, status)

Normalization (brief):

- 1NF: atomic values — all attributes atomic.
- 2NF: composite PKs avoided — non-key attributes depend on whole PK (e.g., Service_Part uses composite PK (service_id, part_id)).
- 3NF: no transitive dependencies. E.g., part unit_price stored in Part and copied into Service_Part.unit_price for historical accuracy.

Phase 2 — SQL DDL (MySQL) — CODE:

```
CREATE DATABASE IF NOT EXISTS VehicleServiceManagement;
USE VehicleServiceManagement;
-- CUSTOMER
CREATE TABLE Customer (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    phone VARCHAR(15) UNIQUE,
    email VARCHAR(100) UNIQUE,
    address VARCHAR(255),
```

```
    registration_date DATE NOT NULL DEFAULT (CURRENT_DATE())
) ENGINE=InnoDB;

-- VEHICLE
CREATE TABLE Vehicle (
    vehicle_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    make VARCHAR(50) NOT NULL,
    model VARCHAR(50) NOT NULL,
    year YEAR NOT NULL CHECK (year >= 1950 AND year <= 2050),  -- fixed: no CURDATE()
    vin VARCHAR(50) UNIQUE,
    reg_no VARCHAR(30) UNIQUE,
    color VARCHAR(30),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) ON DELETE CASCADE
) ENGINE=InnoDB;

-- MECHANIC
CREATE TABLE Mechanic (
    mechanic_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    phone VARCHAR(15),
    email VARCHAR(100),
    hire_date DATE DEFAULT (CURRENT_DATE())
) ENGINE=InnoDB;
```

```
-- SERVICE TYPE
CREATE TABLE ServiceType (
    service_type_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL UNIQUE,
    description VARCHAR(255),
    base_price DECIMAL(10,2) NOT NULL DEFAULT 0.00
) ENGINE=InnoDB;

-- PART
CREATE TABLE Part (
    part_id INT AUTO_INCREMENT PRIMARY KEY,
    part_name VARCHAR(150) NOT NULL,
    part_code VARCHAR(50) UNIQUE,
    unit_price DECIMAL(10,2) NOT NULL DEFAULT 0.00,
    stock_qty INT NOT NULL DEFAULT 0 CHECK (stock_qty >= 0)
) ENGINE=InnoDB;

-- SERVICE
CREATE TABLE Service (
    service_id INT AUTO_INCREMENT PRIMARY KEY,
    vehicle_id INT NOT NULL,
    mechanic_id INT,
    service_type_id INT NOT NULL,
    service_date DATETIME NOT NULL DEFAULT
CURRENT_TIMESTAMP,
    odometer INT DEFAULT NULL,
    notes TEXT,
    status ENUM('Scheduled','In Progress','Completed','Cancelled') NOT
NULL DEFAULT 'Scheduled',
```

```
    total_cost DECIMAL(12,2) NOT NULL DEFAULT 0.00,
    FOREIGN KEY (vehicle_id) REFERENCES Vehicle(vehicle_id) ON DELETE CASCADE,
    FOREIGN KEY (mechanic_id) REFERENCES Mechanic(mechanic_id) ON DELETE SET NULL,
    FOREIGN KEY (service_type_id) REFERENCES ServiceType(service_type_id) ON DELETE RESTRICT
) ENGINE=InnoDB;

-- SERVICE_PART (Many-to-Many)
CREATE TABLE Service_Part (
    service_id INT NOT NULL,
    part_id INT NOT NULL,
    quantity INT NOT NULL CHECK (quantity > 0),
    unit_price DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (service_id, part_id),
    FOREIGN KEY (service_id) REFERENCES Service(service_id) ON DELETE CASCADE,
    FOREIGN KEY (part_id) REFERENCES Part(part_id) ON DELETE RESTRICT
) ENGINE=InnoDB;

-- INVOICE
CREATE TABLE Invoice (
    invoice_id INT AUTO_INCREMENT PRIMARY KEY,
    service_id INT NOT NULL UNIQUE,
    invoice_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    total_amount DECIMAL(12,2) NOT NULL,
    paid_amount DECIMAL(12,2) NOT NULL DEFAULT 0.00,
```

```
    payment_status ENUM('Unpaid','Partial','Paid') NOT NULL DEFAULT 'Unpaid',
    FOREIGN KEY (service_id) REFERENCES Service(service_id) ON DELETE CASCADE
) ENGINE=InnoDB;

-- APPOINTMENT
CREATE TABLE Appointment (
    appointment_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    vehicle_id INT NOT NULL,
    appointment_date DATETIME NOT NULL,
    slot VARCHAR(50),
    status ENUM('Booked','Completed','Cancelled') NOT NULL DEFAULT 'Booked',
    notes VARCHAR(255),
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) ON DELETE CASCADE,
    FOREIGN KEY (vehicle_id) REFERENCES Vehicle(vehicle_id) ON DELETE CASCADE
) ENGINE=InnoDB;


-- CUSTOMERS
INSERT INTO Customer (first_name, last_name, phone, email, address)
VALUES
('Amit', 'Kumar', '9876543210', 'amit@example.com', 'Delhi'),
('Priya', 'Sharma', '9998887776', 'priya@example.com', 'Mumbai'),
('Ravi', 'Verma', '9822334455', 'ravi@example.com', 'Pune');
```

```
-- VEHICLES
INSERT INTO Vehicle (customer_id, make, model, year, vin, reg_no, color)
VALUES
(1, 'Maruti', 'Swift', 2020, 'VIN001', 'DL01AB1234', 'Red'),
(2, 'Hyundai', 'i20', 2022, 'VIN002', 'MH02XY5678', 'White'),
(3, 'Tata', 'Nexon', 2021, 'VIN003', 'MH14GH9999', 'Blue');

-- MECHANICS
INSERT INTO Mechanic (first_name, last_name, phone, email)
VALUES
('Rohan', 'Singh', '9000011111', 'rohan@garage.com'),
('Neha', 'Patel', '9000022222', 'neha@garage.com');

-- SERVICE TYPES
INSERT INTO ServiceType (name, description, base_price)
VALUES
('Oil Change', 'Engine oil replacement and filter check', 1500.00),
('Full Service', 'Complete vehicle checkup', 3500.00),
('Brake Service', 'Brake pad replacement', 2000.00);

-- PARTS
INSERT INTO Part (part_name, part_code, unit_price, stock_qty)
VALUES
('Engine Oil', 'P001', 800.00, 100),
('Air Filter', 'P002', 500.00, 50),
('Brake Pads', 'P003', 1200.00, 30),
('Coolant', 'P004', 400.00, 40);
```

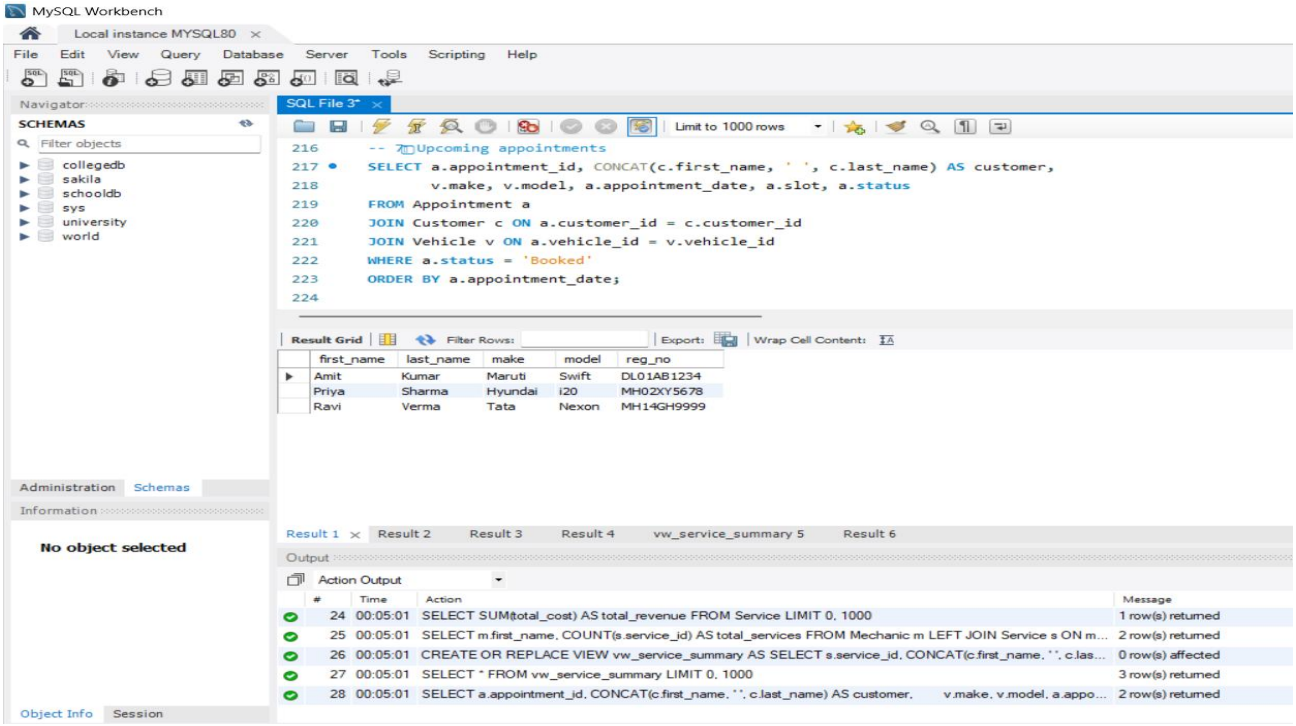
```
-- SERVICES
INSERT INTO Service (vehicle_id, mechanic_id, service_type_id, service_date, odometer, notes, status, total_cost)
VALUES
(1, 1, 1, '2025-11-01 10:00:00', 25000, 'Oil and filter change', 'Completed', 1800.00),
(2, 2, 2, '2025-11-02 11:30:00', 15000, 'Full service done', 'Completed', 3700.00),
(3, 1, 3, '2025-11-03 09:15:00', 30000, 'Brake pads replaced', 'Completed', 2500.00);

-- SERVICE_PARTS
INSERT INTO Service_Part (service_id, part_id, quantity, unit_price)
VALUES
(1, 1, 1, 800.00),
(1, 2, 1, 500.00),
(3, 3, 1, 1200.00);

-- INVOICES
INSERT INTO Invoice (service_id, total_amount, paid_amount, payment_status)
VALUES
(1, 1800.00, 1800.00, 'Paid'),
(2, 3700.00, 2000.00, 'Partial'),
(3, 2500.00, 2500.00, 'Paid');

-- APPOINTMENTS
INSERT INTO Appointment (customer_id, vehicle_id, appointment_date, slot, status, notes)
```

```
VALUES
(1, 1, '2025-11-10 09:00:00', 'Morning', 'Booked', 'Oil change'),
(2, 2, '2025-11-11 11:00:00', 'Midday', 'Booked', 'Full service');

-- 1 View all customer and vehicle details
SELECT c.first_name, c.last_name, v.make, v.model, v.reg_no
FROM Customer c
JOIN Vehicle v ON c.customer_id = v.customer_id;

-- 2 List completed services with customer and mechanic names
SELECT s.service_id, c.first_name AS customer, m.first_name AS
mechanic,
       st.name AS service_type, s.total_cost, s.status
FROM Service s
JOIN Vehicle v ON s.vehicle_id = v.vehicle_id
JOIN Customer c ON v.customer_id = c.customer_id
LEFT JOIN Mechanic m ON s.mechanic_id = m.mechanic_id
JOIN ServiceType st ON s.service_type_id = st.service_type_id
WHERE s.status = 'Completed';

-- 3 Total revenue generated
SELECT SUM(total_cost) AS total_revenue FROM Service;

-- 4 Number of services per mechanic
SELECT m.first_name, COUNT(s.service_id) AS total_services
FROM Mechanic m
LEFT JOIN Service s ON m.mechanic_id = s.mechanic_id
GROUP BY m.first_name;
```

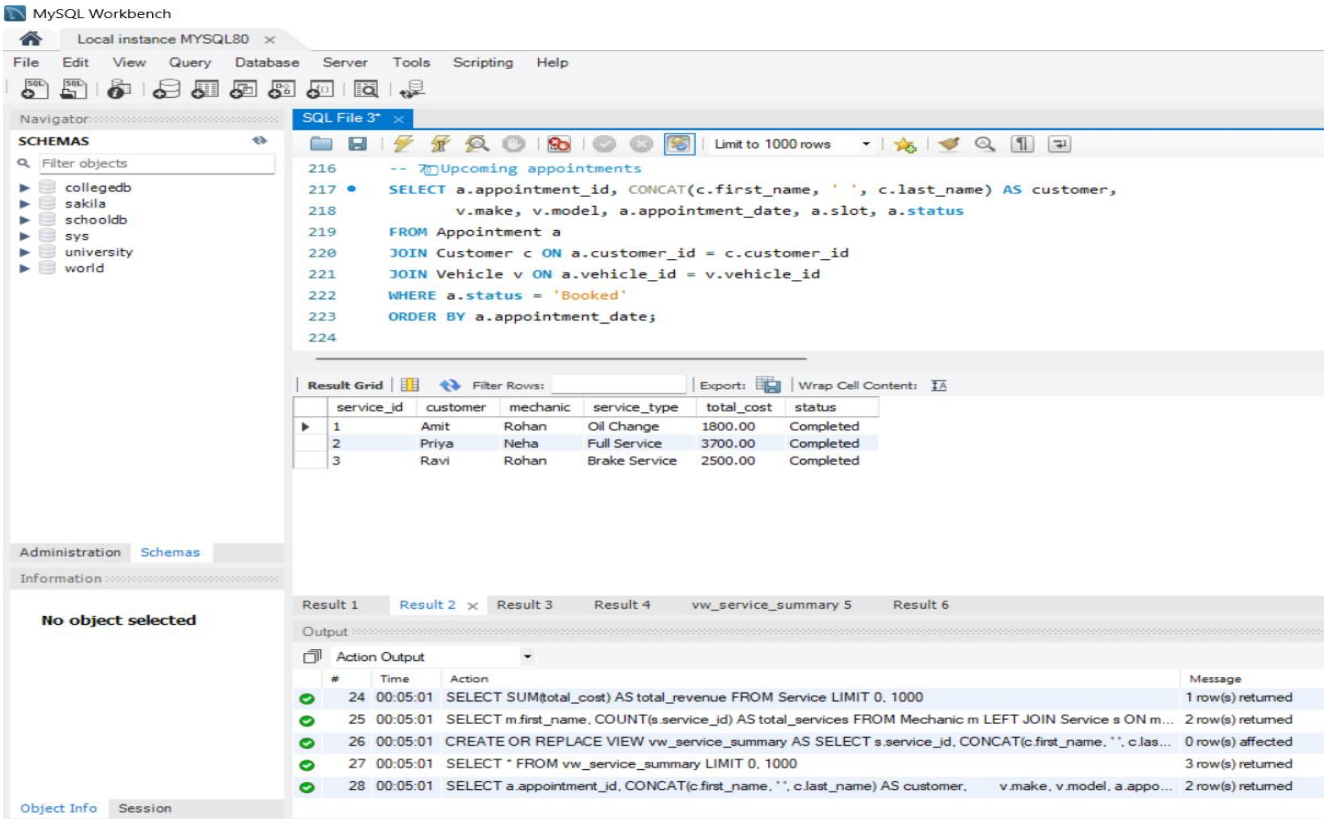
```
-- 5 Create a view for Service Summary
CREATE OR REPLACE VIEW vw_service_summary AS
SELECT s.service_id, CONCAT(c.first_name, ' ', c.last_name) AS customer,
       v.reg_no, st.name AS service_type, s.status, s.total_cost, s.service_date
FROM Service s
JOIN Vehicle v ON s.vehicle_id = v.vehicle_id
JOIN Customer c ON v.customer_id = c.customer_id
JOIN ServiceType st ON s.service_type_id = st.service_type_id;

-- 6 Retrieve from the view
SELECT * FROM vw_service_summary;

-- 7 Upcoming appointments
SELECT a.appointment_id, CONCAT(c.first_name, ' ', c.last_name) AS
customer,
       v.make, v.model, a.appointment_date, a.slot, a.status
FROM Appointment a
JOIN Customer c ON a.customer_id = c.customer_id
JOIN Vehicle v ON a.vehicle_id = v.vehicle_id
WHERE a.status = 'Booked'
ORDER BY a.appointment_date;
```

Output:

1 View All Customers and Their Vehicles:

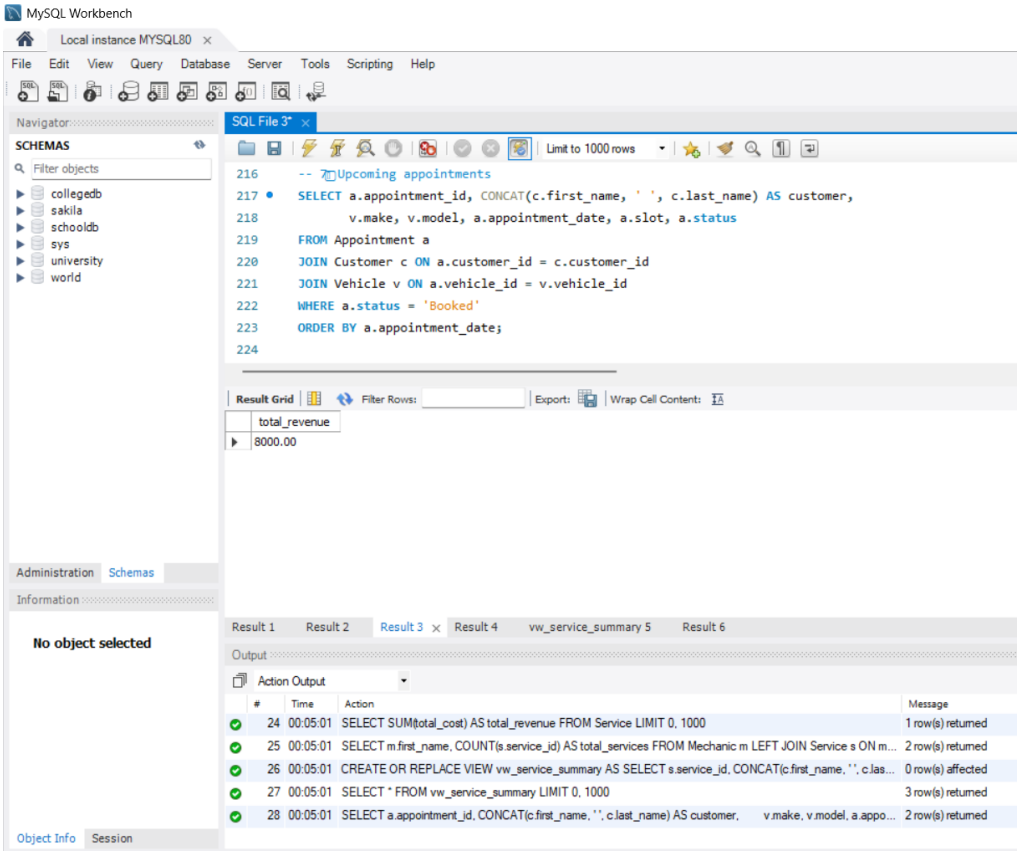


2 Completed Services (Customer + Mechanic + Cost):

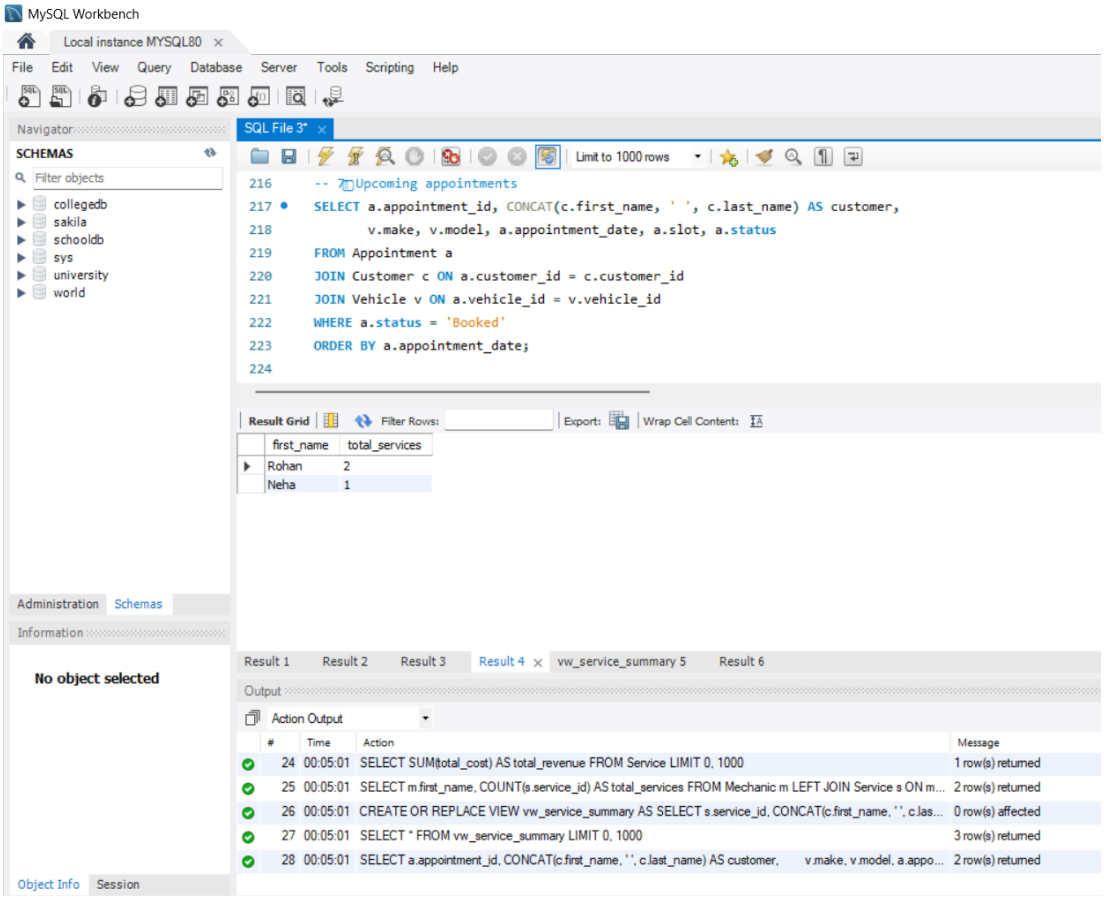


CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

3 Total Revenue Generated:



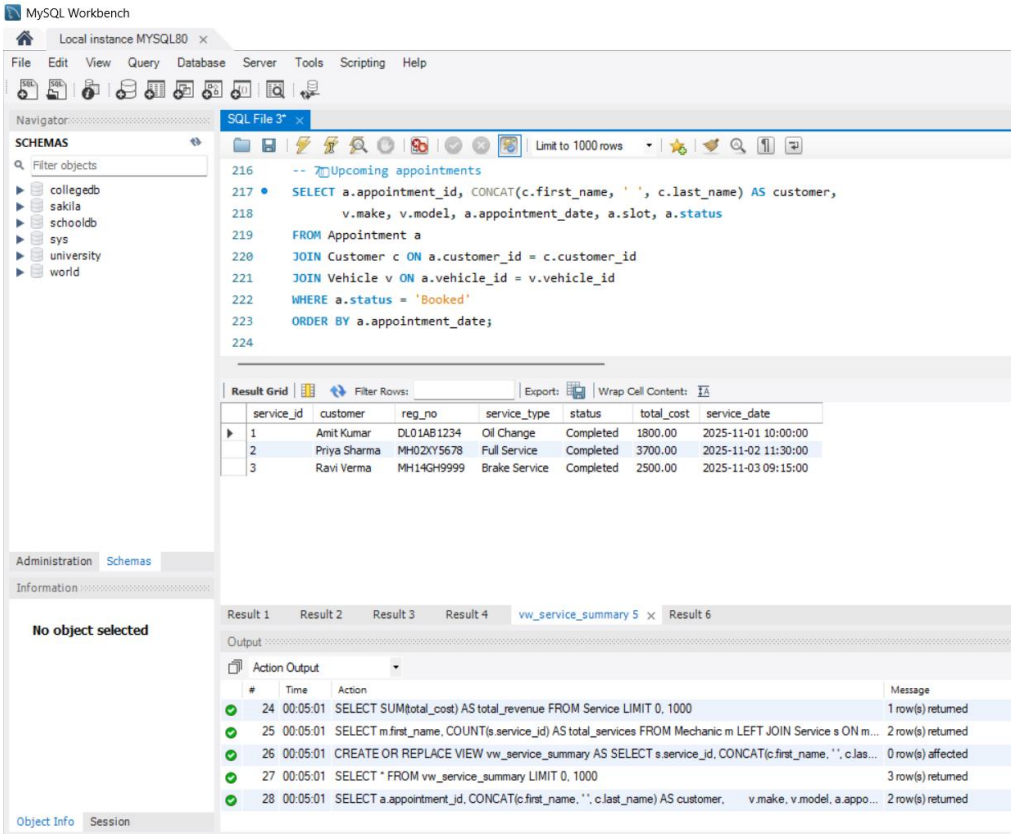
4 Number of Services per Mechanic:



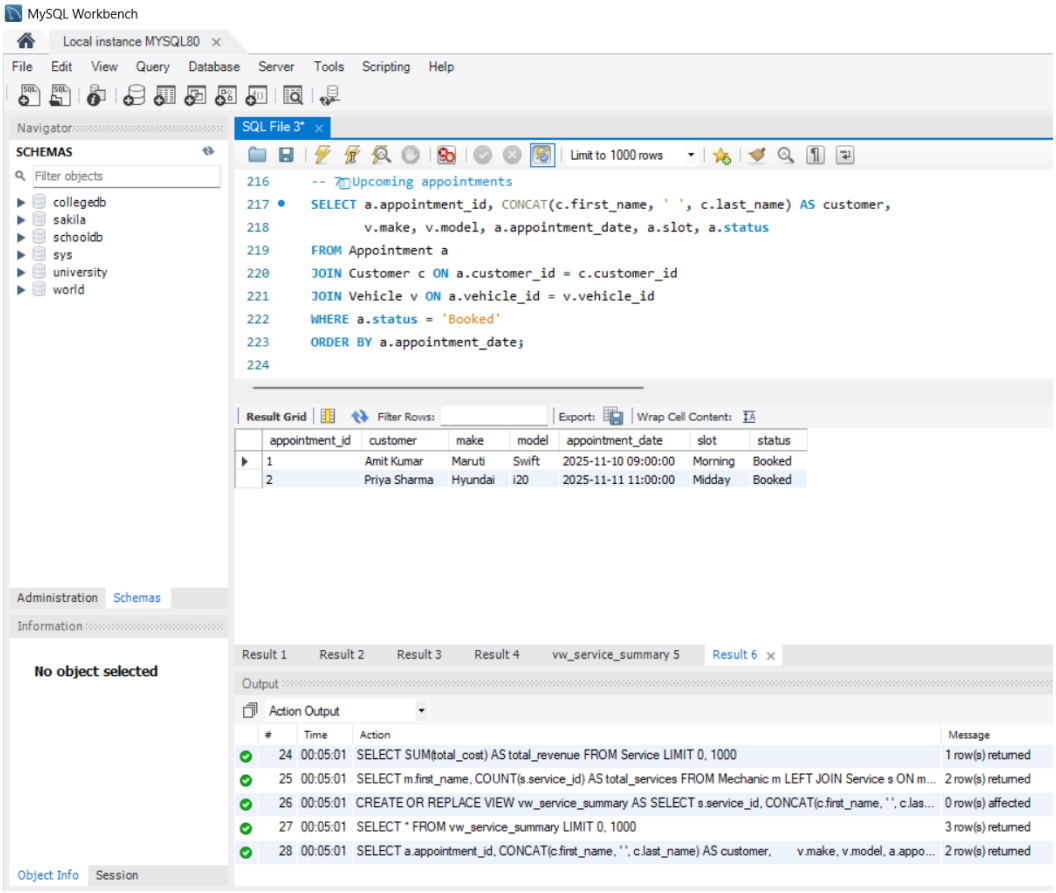
CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

5 View – Service Summary:



6 Upcoming Appointments:



6.Conclusion:

The Vehicle Service Management System (VSMS) successfully demonstrates the design and implementation of a relational database capable of managing customers, vehicles, services, mechanics, parts, and invoices efficiently. The project highlights how structured database systems can replace manual record-keeping, reduce redundancy, and ensure data consistency. Through the use of normalization, entity-relationship modeling, and SQL queries, the system ensures accurate data storage and easy retrieval of critical information such as service history, total revenue, and mechanic performance. Overall, the project achieves its objective of creating an efficient, reliable, and scalable solution for managing operations in a vehicle service center.

7.Learning Outcomes:

By completing this project, the following learning outcomes were achieved:

1. **Database Design Skills:**
   Learned how to identify entities, attributes, and relationships and convert them into an E–R model and relational schema.
2. **Normalization Techniques:**
   Applied 1NF, 2NF, and 3NF to remove redundancy and ensure data integrity.
3. **SQL Query Proficiency:**
   Gained hands-on experience with SQL operations including DDL, DML, and advanced queries using JOIN, GROUP BY, HAVING, and VIEW.
4. **Constraint Implementation:**
   Understood the role of constraints such as PRIMARY KEY, FOREIGN KEY, CHECK, NOT NULL, and UNIQUE in maintaining data accuracy.
5. **Practical Problem Solving:**
   Developed skills to design real-world database systems that improve operational efficiency and automate service management tasks.

8.References:

1. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th Edition). Pearson Education.
2. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th Edition). McGraw Hill.
3. W3Schools. (2025). *SQL Tutorial*. Retrieved from https://www.w3schools.com/sql/
4. MySQL Documentation. (2025). *MySQL 8.0 Reference Manual*. Retrieved from https://dev.mysql.com/doc/
5. TutorialsPoint. (2025). *Database Management System (DBMS) Tutorial*. Retrieved from https://www.tutorialspoint.com/dbms/