## 📋 Assignment Title: "SmartCity Integrated Database System"

## Scenario:

## A SmartCity database integrates multiple services such as E-Commerce, Education, Healthcare, Banking, Transportation, HR, and Entertainment into a unified database system. Your task is to design and implement a database for this SmartCity.

## Task Breakdown

### Phase 1: Entity-Relationship Diagram (ERD)

## Step 1: Design an ERD incorporating at least 5 or all services from the list below.

## Included Services:

## 🛒 E-Commerce Module:

## Entities: Customer, Orders, Product, Payments

## Relationships:

## Customer places multiple orders.

## Each order contains multiple products.

## Payments are made for each order.

## 🎓 Education Module:

## Entities: Student, Courses, Professors, Exams, Enrollments, ExamResult

## Relationships:

## A student can enroll in multiple courses.

## Professors teach multiple courses.

## Exams are conducted for courses.

## Exam results are linked to students.

## 🏥 Healthcare Module:

## Entities: Patient, Doctor, Appointment, Prescriptions

## Relationships:

## Patients book appointments with doctors.

## Doctors issue prescriptions to patients.

## 🏦 Banking Module:

## Entities: Accounts, Transactions, Loan, Branch

## Relationships:

## Users have accounts in different branches.

## Accounts record transactions and loans.

## 🎵 Entertainment / Music Streaming Module:

## Entities: Subscriber, Subscription, Playlist, Songs, Playlist\_Songs

## Relationships:

## Subscribers can create multiple playlists.

## Playlists contain multiple songs.

## Step 2: Ensure all entities, attributes, relationships, and cardinality are defined.

## The ERD has been designed by identifying:

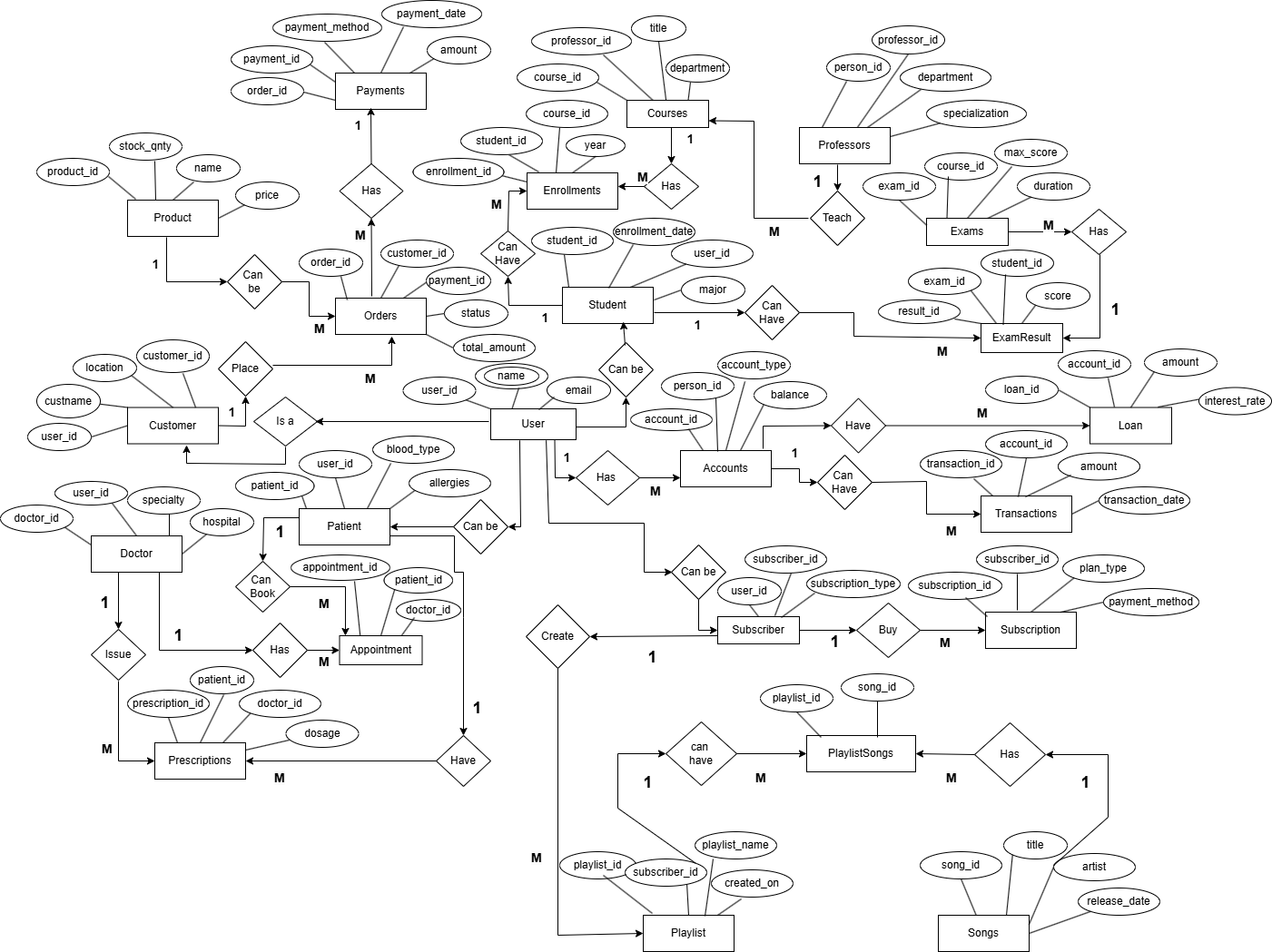
## Entities: Each service is represented by relevant entities.

## Attributes: Attributes relevant to entities are defined.

## Relationships: Relationships between entities are shown with appropriate cardinality (1:1, 1:M, M:M).

## Normalization: The database is normalized to 3NF to avoid redundancy.

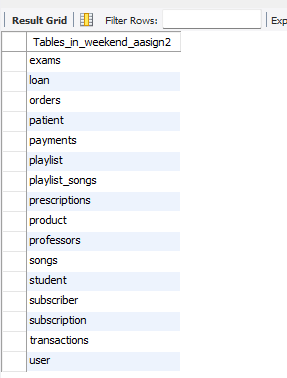
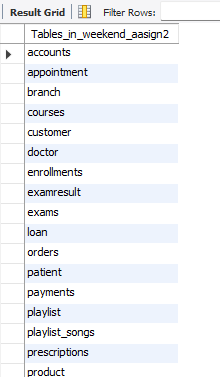
## ERD Diagram



## Phase 2: Implementing Database Schema

## Database Schema - CREATE Queries

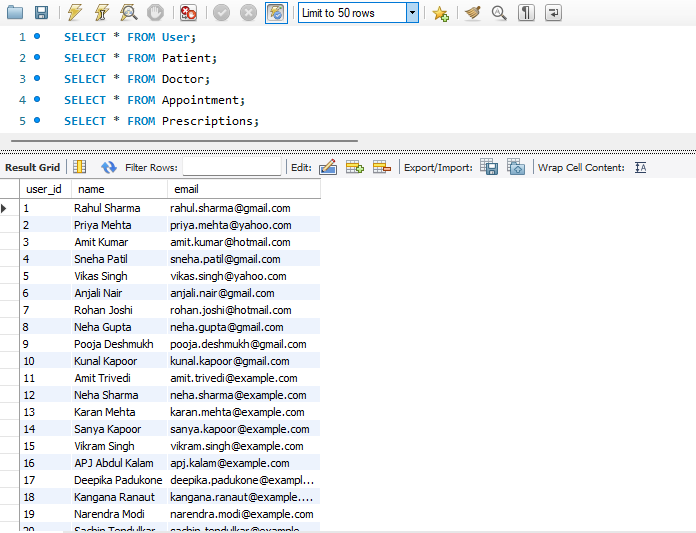
CREATE TABLE User (  
 user\_id INT PRIMARY KEY,  
 name VARCHAR(100),  
 email VARCHAR(100) UNIQUE  
);  
/\* Healthcare Module \*/  
CREATE TABLE Patient (  
 patient\_id INT PRIMARY KEY,  
 user\_id INT,  
 blood\_type VARCHAR(10),  
 allergies TEXT,  
 FOREIGN KEY (user\_id) REFERENCES User(user\_id)  
);  
CREATE TABLE Doctor (  
 doctor\_id INT PRIMARY KEY,  
 user\_id INT,  
 specialty VARCHAR(100),  
 hospital VARCHAR(100),  
 FOREIGN KEY (user\_id) REFERENCES User(user\_id)  
);



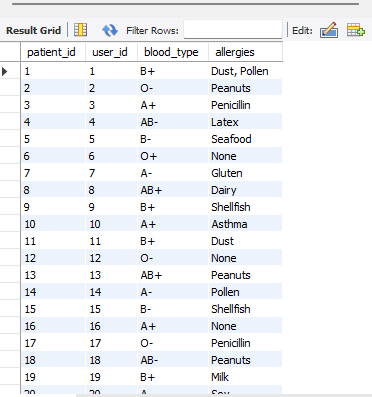
## Phase 3: Inserting Sample Data

## Sample Data - INSERT Queries

-- User Table  
INSERT INTO User (user\_id, name, email) VALUES  
(1, 'Amit Sharma', 'amit.sharma@email.com'),  
(2, 'Priya Gupta', 'priya.gupta@email.com'),  
(3, 'Rohan Kumar', 'rohan.kumar@email.com'),  
(4, 'Neha Verma', 'neha.verma@email.com'),  
(5, 'Suresh Iyer', 'suresh.iyer@email.com');

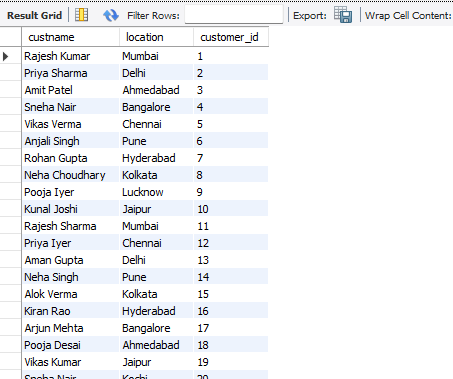


-- Patient Table  
INSERT INTO Patient (patient\_id, user\_id, blood\_type, allergies) VALUES  
(1, 1, 'O+', 'None'),  
(2, 2, 'A-', 'Dust Allergy'),  
(3, 3, 'B+', 'Pollen Allergy'),  
(4, 4, 'AB-', 'None'),  
(5, 5, 'O-', 'Asthma');

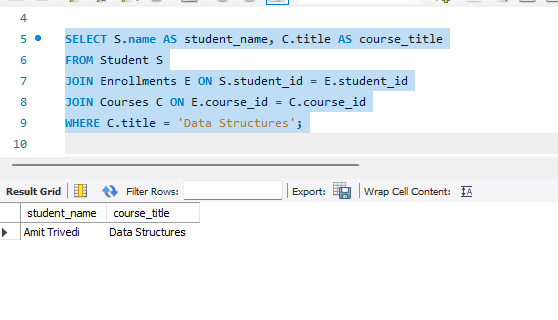


## Phase 4: Writing SQL Queries

## 1. Simple Queries -- 1. Retrieve all customers who have placed at least one order. SELECT DISTINCT c.custname, c.location FROM Customer c JOIN Orders o ON c.customer\_id = o.customer\_id;



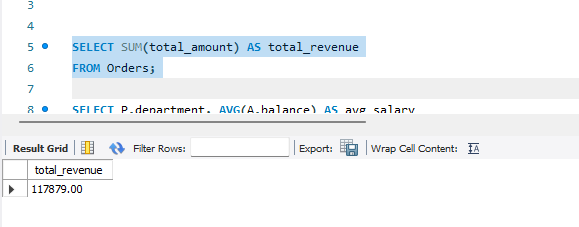
-- 2. Find students enrolled in a specific course.  
SELECT s.name, c.title   
FROM Student s  
JOIN Enrollments e ON s.student\_id = e.student\_id  
JOIN Courses c ON e.course\_id = c.course\_id  
WHERE c.title = 'Data Science';



## 2. Aggregate Functions

-- 4. Find total revenue generated from all orders.

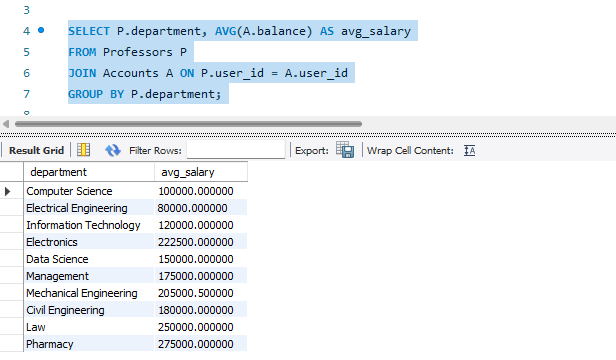
SELECT SUM(total\_amount) AS Total\_Revenue FROM Orders;



-- 5. Calculate average salary per department.

SELECT department, AVG(salary) AS Avg\_Salary FROM Employee

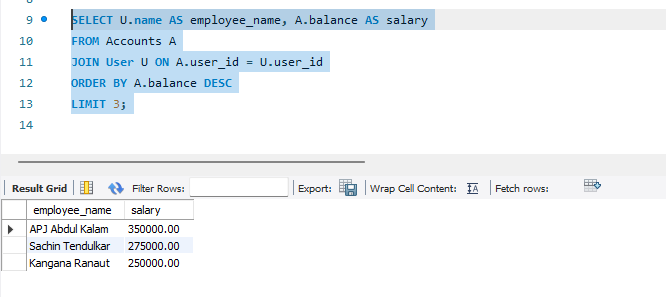
GROUP BY department;



-- 6. Retrieve top 3 highest-paid employees.

SELECT name, salary FROM Employee

ORDER BY salary DESC LIMIT **3**;



## 3. Joins

## -- 7. Get customer names along with their order details using INNER JOIN.

## SELECT c.custname, o.order\_id, o.total\_amount

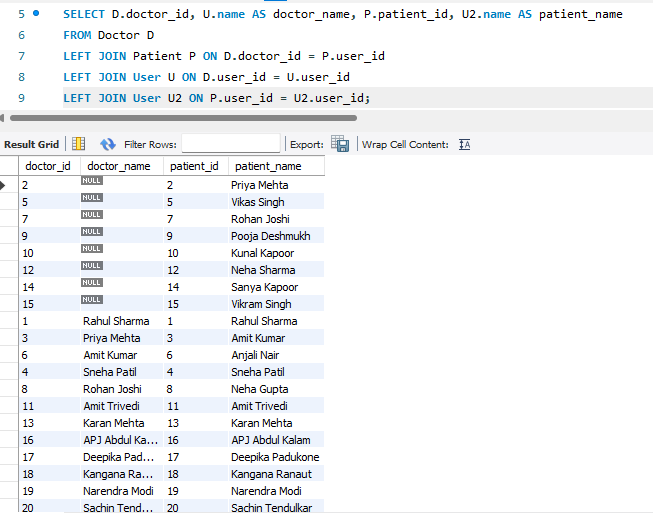
## FROM Customer c INNER JOIN Orders o ON c.customer\_id = o.customer\_id;

## 

## -- 8. Retrieve doctors and their patients using LEFT JOIN.

## SELECT d.name AS Doctor, p.name AS Patient FROM Doctor d LEFT JOIN Appointment a ON d.doctor\_id = a.doctor\_id

## LEFT JOIN Patient p ON a.patient\_id = p.patient\_id;



## -- 9. List songs and users who have added them to playlists using RIGHT JOIN.

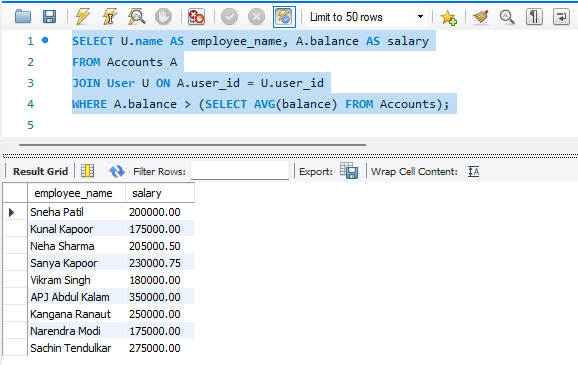
## SELECT s.title AS Song, u.name AS User FROM Songs s RIGHT JOIN Playlist\_Songs ps ON s.song\_id = ps.song\_id RIGHT JOIN Playlist pl ON ps.playlist\_id = pl.playlist\_id RIGHT JOIN Subscriber sub ON pl.subscriber\_id = sub.subscriber\_id RIGHT JOIN User u ON sub.user\_id = u.user\_id;

## 4. Subqueries

-- 10. Find employees earning more than the company average salary.

SELECT name, salary FROM Employee e

WHERE salary > (SELECT AVG(salary) FROM Employee);



-- 11. Retrieve products that have been ordered more than 10 times.

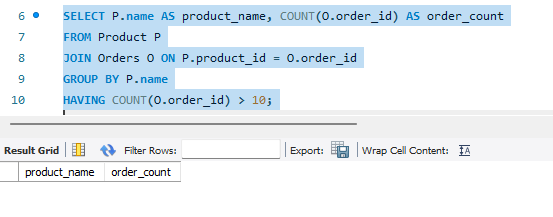
SELECT name FROM Product

WHERE product\_id IN (

SELECT product\_id FROM OrdersGROUP BY product\_id

HAVING COUNT(order\_id) > 10

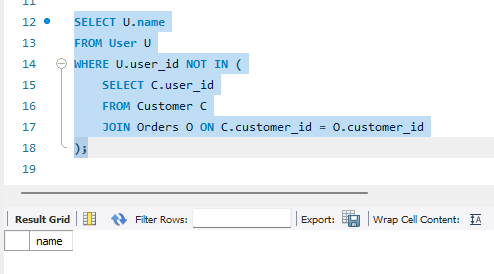
);



-- 12. Get users who have never placed an order.

SELECT name FROM User WHERE user\_id NOT IN (

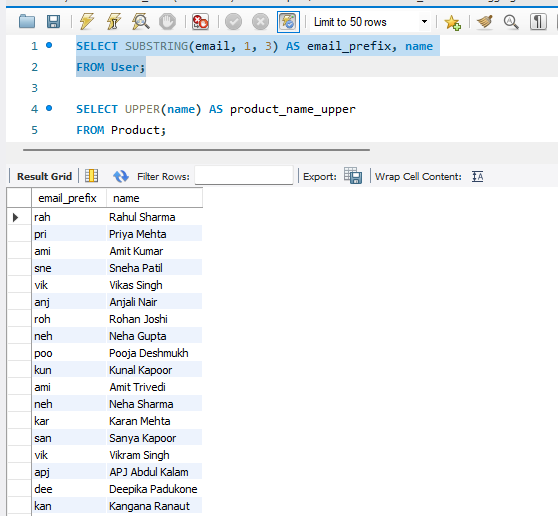
SELECT user\_id FROM Orders);

****

## 5. String Functions

## -- 13. Extract the first 3 characters of all user emails.

## SELECT LEFT(email, 3) AS EmailPrefix FROM User;



-- 14. Convert all product names to uppercase.

SELECT UPPER(name) AS Product\_Name FROM Product;

