📋 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:

1. 🛒 E-Commerce Module:
    - Entities: Customer, Orders, Product, Payments
    - Relationships:
        - Customer places multiple orders.
        - Each order contains multiple products.
        - Payments are made for each order.
2. 🎓 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.
3. 🏥 Healthcare Module:
    - Entities: Patient, Doctor, Appointment, Prescriptions
    - Relationships:
        - Patients book appointments with doctors.

        - Doctors issue prescriptions to patients.

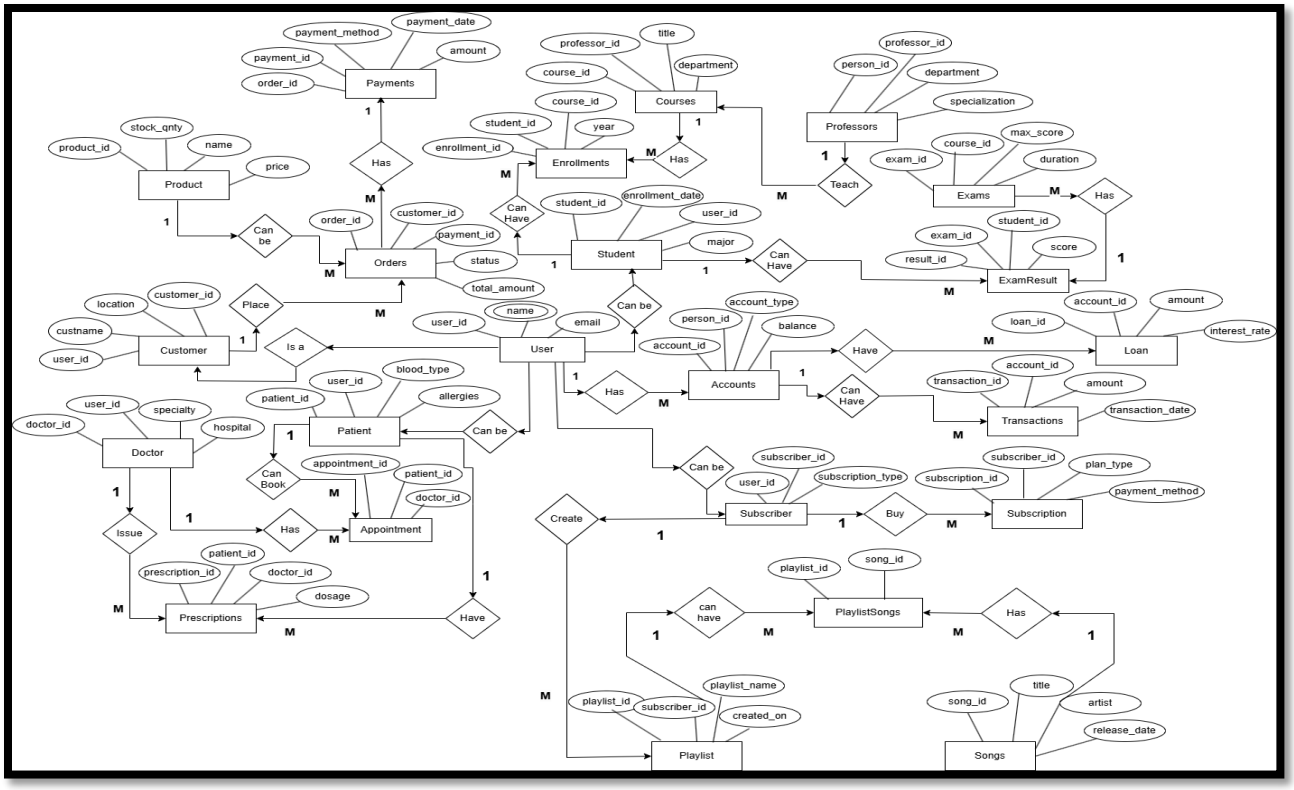
4. 🏦 Banking Module:
    - Entities: Accounts, Transactions, Loan, Branch
    - Relationships:
        - Users have accounts in different branches.
        - Accounts record transactions and loans.
5. 🎵 Entertainment / Music Streaming Module:
    - Entities: Subscriber, Subscription, Playlist, Songs, 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.

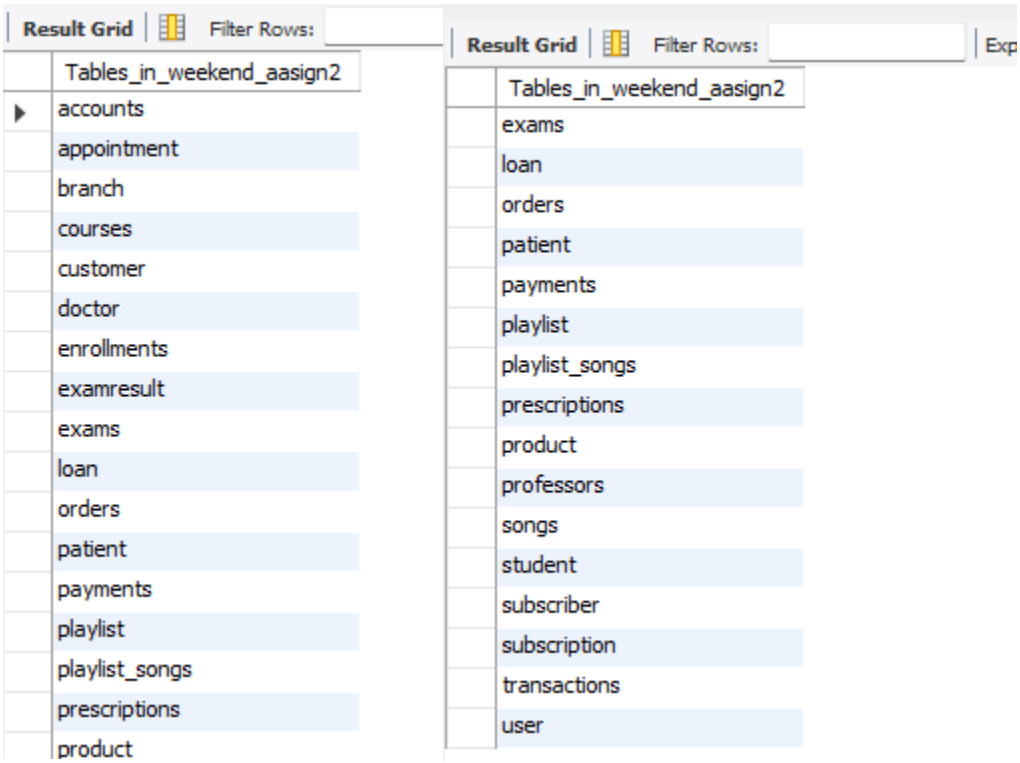
ERD Diagram



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)
);
```



Tables_in_weekend_aasign2
accounts
appointment
branch
courses
customer
doctor
enrollments
examresult
exams
loan
orders
patient
payments
playlist
playlist_songs
prescriptions
product

Tables_in_weekend_aasign2
exams
loan
orders
patient
payments
playlist
playlist_songs
prescriptions
product
professors
songs
student
subscriber
subscription
transactions
user

Phase 3: Inserting Sample Data

Sample Data - INSERT Queries

-- User Table

INSERT INTO User (user_id, name, email) VALUES

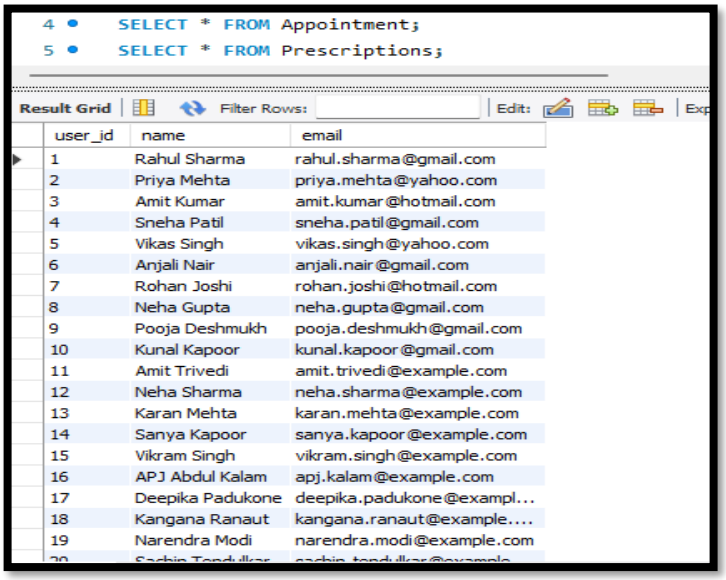
(1, 'Amit Sharma', 'amit.sharma@email.com'),

(2, 'Priya Gupta', 'priya.gupta@email.com'),

(3, 'Rohan Kumar', 'rohan.kumar@email.com'),

(4, 'Neha Verma', 'neha.verma@email.com'),

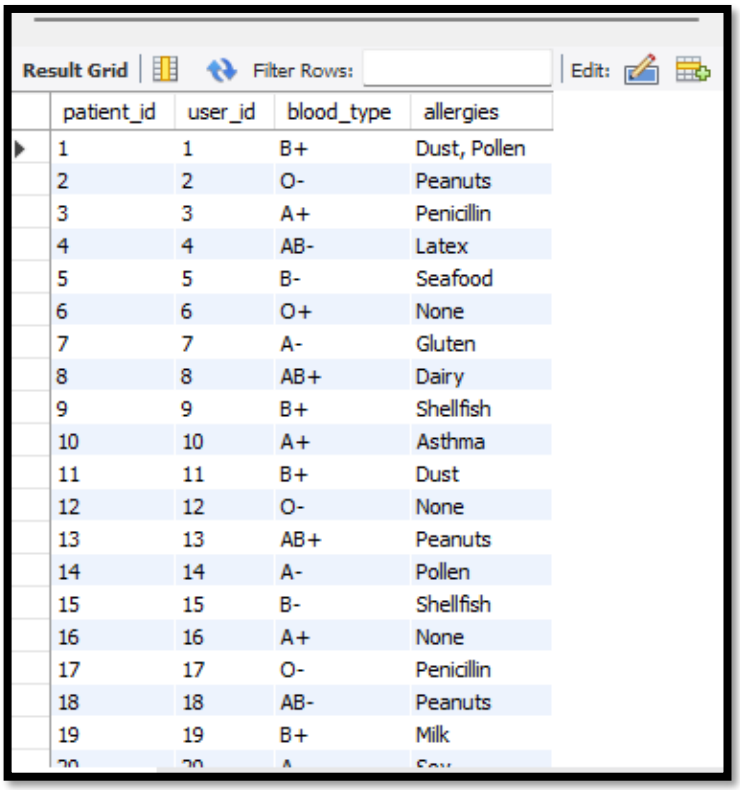
(5, 'Suresh Iyer', 'suresh.iyer@email.com');



```
4 •  SELECT * FROM Appointment;
5 •  SELECT * FROM Prescriptions;
```

user_id	name	email
1	Rahul Sharma	rahul.sharma@gmail.com
2	Priya Mehta	priya.mehta@yahoo.com
3	Amit Kumar	amit.kumar@hotmail.com
4	Sneha Patil	sneha.patil@gmail.com
5	Vikas Singh	vikas.singh@yahoo.com
6	Anjali Nair	anjali.nair@gmail.com
7	Rohan Joshi	rohan.joshi@hotmail.com
8	Neha Gupta	neha.gupta@gmail.com
9	Pooja Deshmukh	pooja.deshmukh@gmail.com
10	Kunal Kapoor	kunal.kapoor@gmail.com
11	Amit Trivedi	amit.trivedi@example.com
12	Neha Sharma	neha.sharma@example.com
13	Karan Mehta	karan.mehta@example.com
14	Sanya Kapoor	sanya.kapoor@example.com
15	Vikram Singh	vikram.singh@example.com
16	APJ Abdul Kalam	apj.kalam@example.com
17	Deepika Padukone	deepika.padukone@exampl...
18	Kangana Ranaut	kangana.ranaut@example....
19	Narendra Modi	narendra.modi@example.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');
```



Result Grid | Filter Rows: | Edit:

patient_id	user_id	blood_type	allergies
1	1	B+	Dust, Pollen
2	2	O-	Peanuts
3	3	A+	Penicillin
4	4	AB-	Latex
5	5	B-	Seafood
6	6	O+	None
7	7	A-	Gluten
8	8	AB+	Dairy
9	9	B+	Shellfish
10	10	A+	Asthma
11	11	B+	Dust
12	12	O-	None
13	13	AB+	Peanuts
14	14	A-	Pollen
15	15	B-	Shellfish
16	16	A+	None
17	17	O-	Penicillin
18	18	AB-	Peanuts
19	19	B+	Milk

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

custname	location	customer_id
Rajesh Kumar	Mumbai	1
Priya Sharma	Delhi	2
Amit Patel	Ahmedabad	3
Sneha Nair	Bangalore	4
Vikas Verma	Chennai	5
Anjali Singh	Pune	6
Rohan Gupta	Hyderabad	7
Neha Choudhary	Kolkata	8
Pooja Iyer	Lucknow	9
Kunal Joshi	Jaipur	10
Rajesh Sharma	Mumbai	11
Priya Iyer	Chennai	12
Aman Gupta	Delhi	13
Neha Singh	Pune	14
Alok Verma	Kolkata	15
Kiran Rao	Hyderabad	16
Arjun Mehta	Bangalore	17
Pooja Desai	Ahmedabad	18
Vikas Kumar	Jaipur	19

-- 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';
```

```
4
5  SELECT S.name AS student_name, C.title AS course_title
6  FROM Student S
7  JOIN Enrollments E ON S.student_id = E.student_id
8  JOIN Courses C ON E.course_id = C.course_id
9  WHERE C.title = 'Data Structures';
10
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_name	course_title
Amit Trivedi	Data Structures

2. Aggregate Functions

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

```
SELECT SUM(total_amount) AS Total_Revenue FROM Orders;
```

```
4
5 •  SELECT SUM(total_amount) AS total_revenue
6    FROM Orders;
7
8 •  SELECT P.department, AVG(A.balance) AS avg_salary
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_revenue
117879.00

-- 5. Calculate average salary per department.

SELECT department, AVG(salary) AS Avg_Salary FROM Employee

GROUP BY department;

```
3
4 •  SELECT P.department, AVG(A.balance) AS avg_salary
5    FROM Professors P
6    JOIN Accounts A ON P.user_id = A.user_id
7    GROUP BY P.department;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

department	avg_salary
Computer Science	100000.000000
Electrical Engineering	80000.000000
Information Technology	120000.000000
Electronics	222500.000000
Data Science	150000.000000
Management	175000.000000
Mechanical Engineering	205000.500000
Civil Engineering	180000.000000
Law	250000.000000
Pharmacy	275000.000000

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

SELECT name, salary FROM Employee

ORDER BY salary DESC LIMIT **3**;

```
9 •  SELECT U.name AS employee_name, A.balance AS salary
10   FROM Accounts A
11   JOIN User U ON A.user_id = U.user_id
12   ORDER BY A.balance DESC
13   LIMIT 3;
14
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

employee_name	salary
APJ Abdul Kalam	350000.00
Sachin Tendulkar	275000.00
Kangana Ranaut	250000.00

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;

```
1  SELECT C.custname, O.order_id, O.total_amount, O.status
2  FROM Customer C
3  INNER JOIN Orders O ON C.customer_id = O.customer_id;
4
5  SELECT D.doctor_id, U.name AS doctor_name, P.patient_id, U2.name AS patient_name
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

custname	order_id	total_amount	status
Rajesh Kumar	1	4999.00	Completed
Priya Sharma	2	2999.00	Pending
Amit Patel	3	1499.00	Shipped
Sneha Nair	4	799.00	Completed
Vikas Verma	5	2499.00	Pending
Anjali Singh	6	399.00	Shipped
Rohan Gupta	7	1999.00	Completed
Neha Choudhary	8	899.00	Pending
Pooja Iyer	9	699.00	Completed
Kunal Joshi	10	599.00	Shipped
Rajesh Sharma	11	39998.00	Completed
Priya Iyer	12	2999.00	Pending
Aman Gupta	13	9999.00	Completed
Neha Singh	14	14999.00	Completed
Alok Verma	15	999.00	Cancelled
Kiran Rao	16	1999.00	Completed
Arjun Mehta	17	3999.00	Pending
Pooja Desai	18	2999.00	Completed
Vikas Kumar	19	2499.00	Shipped
Sneha Nair	20	19999.00	Completed

-- 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;

```
5  SELECT D.doctor_id, U.name AS doctor_name, P.patient_id, U2.name AS patient_name
6  FROM Doctor D
7  LEFT JOIN Patient P ON D.doctor_id = P.user_id
8  LEFT JOIN User U ON D.user_id = U.user_id
9  LEFT JOIN User U2 ON P.user_id = U2.user_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

doctor_id	doctor_name	patient_id	patient_name
2	NULL	2	Priya Mehta
5	NULL	5	Vikas Singh
7	NULL	7	Rohan Joshi
9	NULL	9	Pooja Deshmukh
10	NULL	10	Kunal Kapoor
12	NULL	12	Neha Sharma
14	NULL	14	Sanya Kapoor
15	NULL	15	Vikram Singh
1	Rahul Sharma	1	Rahul Sharma
3	Priya Mehta	3	Amit Kumar
6	Amit Kumar	6	Anjali Nair
4	Sneha Patil	4	Sneha Patil
8	Rohan Joshi	8	Neha Gupta
11	Amit Trivedi	11	Amit Trivedi
13	Karan Mehta	13	Karan Mehta
16	APJ Abdul Ka...	16	APJ Abdul Kalam
17	Deepika Pad...	17	Deepika Padukone
18	Kangana Ra...	18	Kangana Ranaut
19	Narendra Modi	19	Narendra Modi
20	Sachin Tend...	20	Sachin Tendulkar

-- 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;

```
10
11 • SELECT S.title AS song_title, U.name AS user_name
12   FROM Songs S
13   RIGHT JOIN Playlist_Songs PS ON S.song_id = PS.song_id
14   RIGHT JOIN Playlist P ON PS.playlist_id = P.playlist_id
15   RIGHT JOIN Subscriber SB ON P.subscriber_id = SB.subscriber_id
16   RIGHT JOIN User U ON SB.user_id = U.user_id;
17
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

song_title	user_name
Tum Hi Ho	Rahul Sharma
Channa Mereya	Rahul Sharma
Kabira	Rahul Sharma
Dil Dhadakne Do	Priya Mehta
Raabta	Priya Mehta
Phir Bhi Tumko Chaahunga	Amit Kumar
Tera Ban Jaunga	Amit Kumar
Lag Ja Gale	Sneha Patil
Tujh Mein Rab Dikhta Hai	Sneha Patil
Maahi Ve	Vikas Singh
Tum Hi Ho	Anjali Nair
Kabira	Anjali Nair
Dil Dhadakne Do	Rohan Joshi
Raabta	Rohan Joshi

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);

Limit to 50 rows

```
1 • SELECT U.name AS employee_name, A.balance AS salary
2   FROM Accounts A
3   JOIN User U ON A.user_id = U.user_id
4   WHERE A.balance > (SELECT AVG(balance) FROM Accounts);
5
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee_name	salary
Sneha Patil	200000.00
Kunal Kapoor	175000.00
Neha Sharma	205000.50
Sanya Kapoor	230000.75
Vikram Singh	180000.00
APJ Abdul Kalam	350000.00
Kangana Ranaut	250000.00
Narendra Modi	175000.00
Sachin Tendulkar	275000.00

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

SELECT name FROM Product

WHERE product_id IN (

SELECT product_id FROM Orders GROUP BY product_id

HAVING COUNT(order_id) > 10

);

```
6  • SELECT P.name AS product_name, COUNT(O.order_id) AS order_count
7    FROM Product P
8    JOIN Orders O ON P.product_id = O.order_id
9    GROUP BY P.name
10   HAVING COUNT(O.order_id) > 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_name	order_count

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

SELECT name FROM User WHERE user_id NOT IN (

SELECT user_id FROM Orders);

```
12 • SELECT U.name
13   FROM User U
14   WHERE U.user_id NOT IN (
15       SELECT C.user_id
16       FROM Customer C
17       JOIN Orders O ON C.customer_id = O.customer_id
18   );
19
```

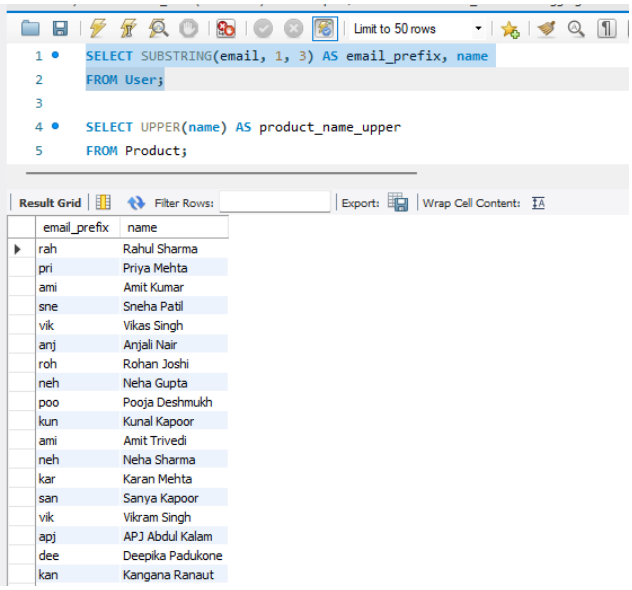
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name

5. String Functions

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

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



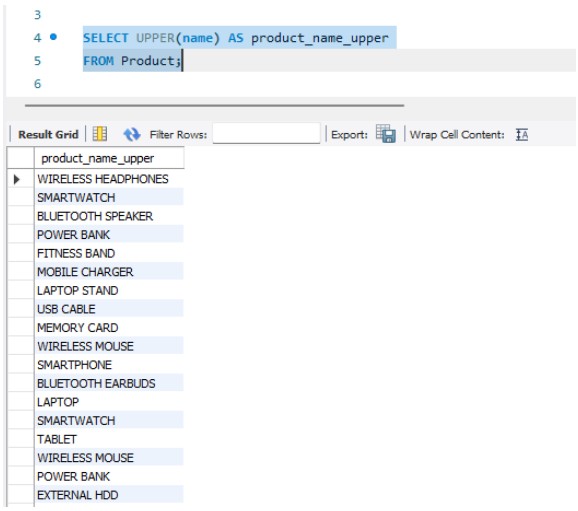
```
SELECT SUBSTRING(email, 1, 3) AS email_prefix, name
FROM User;

SELECT UPPER(name) AS product_name_upper
FROM Product;
```

email_prefix	name
rah	Rahul Sharma
pri	Priya Mehta
ami	Amit Kumar
sne	Sneha Patil
vik	Vikas Singh
anj	Anjali Nair
roh	Rohan Joshi
neh	Neha Gupta
poo	Pooja Deshmukh
kun	Kunal Kapoor
ami	Amit Trivedi
neh	Neha Sharma
kar	Karan Mehta
san	Sanya Kapoor
vik	Vikram Singh
apj	APJ Abdul Kalam
dee	Deepika Padukone
kan	Kangana Ranaut

-- 14. Convert all product names to uppercase.

SELECT UPPER(name) AS Product_Name FROM Product;



```
SELECT UPPER(name) AS product_name_upper
FROM Product;
```

product_name_upper
WIRELESS HEADPHONES
SMARTWATCH
BLUETOOTH SPEAKER
POWER BANK
FITNESS BAND
MOBILE CHARGER
LAPTOP STAND
USB CABLE
MEMORY CARD
WIRELESS MOUSE
SMARTPHONE
BLUETOOTH EARBUDS
LAPTOP
SMARTWATCH
TABLET
WIRELESS MOUSE
POWER BANK
EXTERNAL HDD