Task 6: Subqueries and Nested Queries

Objective: Use subqueries in SELECT, WHERE, and FROM

Tools: MySQL Workbench

Deliverables: SQL queries with nested logic

Objectives:

1.Use scalar and correlated subqueries

2.Use subqueries inside IN, EXISTS, =

1. Creating Customers table

CREATE TABLE Customers (customer_id INT PRIMARY KEY, name VARCHAR(30), phone VARCHAR(10), city VARCHAR(20), age INT);

2. Creating Orders table

CREATE TABLE Orders (order_id INT PRIMARY KEY, customer_id INT, product VARCHAR(20), price FLOAT, FOREIGN KEY(customer_id) REFERENCES Customers(customer_id);

3. Inserting data into Customers

INSERT INTO Customers VALUES (1, 'Ravi Sharma','9745834678', 'Mumbai', 21);

INSERT INTO Customers VALUES (2, 'Priya Verma','8345587878', 'Delhi', 32);

INSERT INTO Customers VALUES (3, 'Amit Kumar','9793258778', 'Pune', 25);

INSERT INTO Customers VALUES (4, 'Neha Singh','9432885346', 'Chennai', 40);

INSERT INTO Customers VALUES (5, 'Anjali Mehta', '9874563210', 'Mumbai', 30);

INSERT INTO Customers VALUES (6, 'Rohit Desai', '9123456789', 'Mumbai', 26);

INSERT INTO Customers VALUES (7, 'Karan Gupta', '9988776655', 'Delhi', 40);

INSERT INTO Customers VALUES (8, 'Simran Kaur', '8877665544', 'Delhi', 28);

4. Inserting data into Orders

INSERT INTO Orders VALUES (101, 1, 'Laptop', 55000.00);

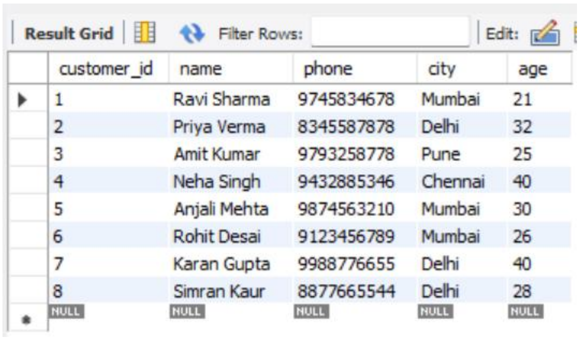
INSERT INTO Orders VALUES (102, 1, 'Keyboard', 1500.00);

INSERT INTO Orders VALUES (103, 2, 'Smartphone', 18000.00);

INSERT INTO Orders VALUES (104, 3, 'Tablet', 12000.00);

5. Displaying data from Customers table

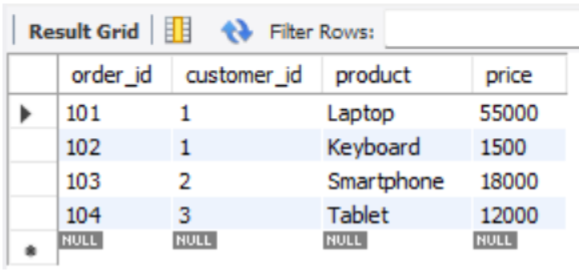
SELECT customer_id, name, phone, city FROM Customers;



customer_id	name	phone	city	age
1	Ravi Sharma	9745834678	Mumbai	21
2	Priya Verma	8345587878	Delhi	32
3	Amit Kumar	9793258778	Pune	25
4	Neha Singh	9432885346	Chennai	40
5	Anjali Mehta	9874563210	Mumbai	30
6	Rohit Desai	9123456789	Mumbai	26
7	Karan Gupta	9988776655	Delhi	40
8	Simran Kaur	8877665544	Delhi	28
NULL	NULL	NULL	NULL	NULL

6. Displaying data from Orders table

SELECT order_id, customer_id, product, price FROM Orders;



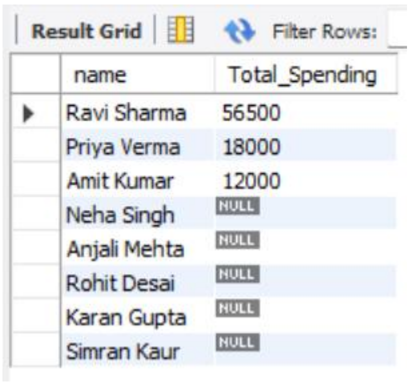
order_id	customer_id	product	price
101	1	Laptop	55000
102	1	Keyboard	1500
103	2	Smartphone	18000
104	3	Tablet	12000
NULL	NULL	NULL	NULL

7. To Find each customer's total spending

SELECT name, (SELECT SUM(price) FROM Orders

WHERE Customers.customer_id = Orders.customer_id) AS Total_Spending

FROM Customers;

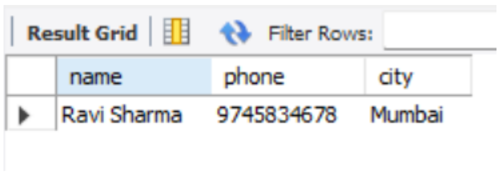


name	Total_Spending
Ravi Sharma	56500
Priya Verma	18000
Amit Kumar	12000
Neha Singh	NULL
Anjali Mehta	NULL
Rohit Desai	NULL
Karan Gupta	NULL
Simran Kaur	NULL

8. To find customers who purchased products costing more than 20000

SELECT name, phone, city FROM CUSTOMERS WHERE

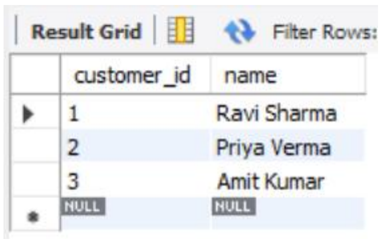
customer_id IN (SELECT customer_id FROM Orders WHERE price>20000);



name	phone	city
Ravi Sharma	9745834678	Mumbai

9. To find customers who have placed at least one order

SELECT customer_id, name FROM Customers C WHERE EXISTS (SELECT * FROM Orders O WHERE C.customer_id=O.customer_id);



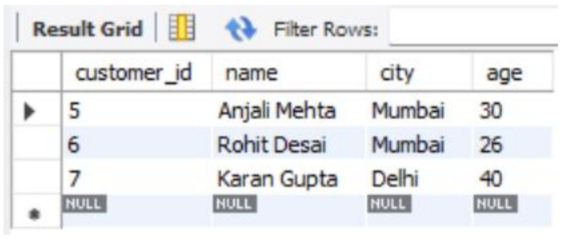
customer_id	name
1	Ravi Sharma
2	Priya Verma
3	Amit Kumar
NULL	NULL

10. To find customers whose age is greater than the average age of customers from their city

SELECT c1.customer_id, c1.name, c1.city, c1.age

FROM Customers c1

WHERE c1.age > (SELECT AVG(c2.age) FROM Customers c2 WHERE c1.city = c2.city);



customer_id	name	city	age
5	Anjali Mehta	Mumbai	30
6	Rohit Desai	Mumbai	26
7	Karan Gupta	Delhi	40
NULL	NULL	NULL	NULL

11. To find the average price of orders per customer and display only those with avg price > 20000

SELECT sub.customer_id, sub.avg_price

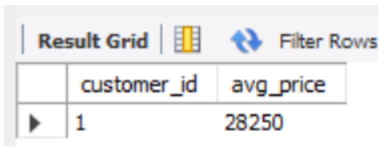
FROM (

  SELECT customer_id, AVG(price) AS avg_price

  FROM Orders

  GROUP BY customer_id

) AS sub WHERE sub.avg_price > 20000;



customer_id	avg_price
1	28250