**EXPERIMENT NO: 8**

**IMPLEMENTATION OF SET OPERATORS NESTED QUERIES, AND JOIN QUERIES**

**AIM:**

To get introduced to

-UNION - JOIN

-INTERSECTION - NESTED QUERIES

-MINUS - GROUP BY & HAVING **JOINS**

**QUESTION**

**Amazon is one of the largest online stores operating in the United States of America. They are maintaining four tables in their database. The Items table, Customers table, Orders table and Delivery table. Each of these tables contains the following attributes:**

**Items: - itemid (primary key)**

**Itemname(type=varchar(50))**

**category**

**Price**

**Instock (type=int, greater than or equal to zero)**

**Customers:- custid (primary key)**

**Custname**

**Address**

**State**

**Orders:- orderid (primary key) Itemid( refers to**

**itemid of Items table) Quantity (type=int)**

**Orderdate (type=date)**

**Delivery:- deliveryid (primary key) Custid (refers to custid in**

**customers table) Orderid**

**(refers to ordered in**

**orders table)**

**Create the above tables and populate them with appropriate data.**

**.1.List the details of all customers who have placed an order**

**2.List the details of all customers whose orders have been delivered**

**3.Find the orderdate for all customers whose name starts in the letter ‘J’**

**4.Display the name and price of all items bought by the customer ‘Mickey’**

**5.List the details of all customers who have placed an order after January 2013 and not received delivery of items**

**6.Find the itemid of items which has either been ordered or not**

**delivered. (Use SET UNION)**

**7.Find the name of all customers who have placed an order and have their orders delivered.(Use SET INTERSECTION**)

8.F**ind the custname of all customers who have placed an order but not having their orders delivered. (Use SET MINUS)**

**9.Find the name of the customer who has placed the most number of orders.**

**10.Find the details of all customers who have purchased items exceeding a price of 5000 $.**

**11.Find the name and address of customers who has not ordered a ‘Samsung Galaxy S4’**

**12.Perform Left Outer Join and Right Outer Join on Customers & Orders**

**13.Find the details of all customers grouped by state .**

**14.Display the details of all items grouped by category and having a price greater than average price of all items.**

CREATE TABLE Items66 ( itemid INT PRIMARY KEY, itemname VARCHAR(50), category VARCHAR(50), price DECIMAL(10, 2), instock INT CHECK (instock >= 0) );

CREATE TABLE Customers66 ( custid INT PRIMARY KEY, custname VARCHAR(50), address VARCHAR(100), state VARCHAR(50) );

CREATE TABLE Orders66 ( orderid INT PRIMARY KEY, custid INT, itemid INT, quantity INT, orderdate DATE, FOREIGN KEY (custid) REFERENCES Customers66(custid), FOREIGN KEY (itemid) REFERENCES Items66(itemid) );

CREATE TABLE Delivery66 ( deliveryid INT PRIMARY KEY, custid INT, orderid INT, FOREIGN KEY (custid) REFERENCES Customers66(custid), FOREIGN KEY (orderid) REFERENCES Orders66(orderid) );

INSERT INTO Items66 VALUES (1, 'Laptop', 'Electronics', 1000.00, 50);

INSERT INTO Items66 VALUES (2, 'Headphones', 'Electronics', 100.00, 150);

INSERT INTO Items66 VALUES (3, 'Office Chair', 'Furniture', 200.00, 20);

INSERT INTO Items66 VALUES (4, 'Pen', 'Stationery', 2.00, 500);

INSERT INTO Items66 VALUES (5, 'Samsung Galaxy S4', 'Electronics',700.00, 30);

INSERT INTO Items66 VALUES (6, 'iphone', 'Electronics', 6000.00, 50);

INSERT INTO Customers66 VALUES (1, 'John Doe', '123 Elm St', 'California');

INSERT INTO Customers66 VALUES (2, 'Jane Smith', '456 Oak St', 'New York'); INSERT INTO Customers66 VALUES (3, 'Emily Davis', '789 Pine St', 'California');

INSERT INTO Customers66 VALUES (4, 'Mickey Mouse', 'Disneyland', 'Florida');

INSERT INTO Customers66 VALUES (5, ‘sameen', 'Disneyland', 'Florida');

INSERT INTO Orders66 VALUES (1, 1, 1, 2, '2023-08-01');

INSERT INTO Orders66 VALUES (2, 2, 2, 1, '2024-08-02');

INSERT INTO Orders66 VALUES (3, 3, 3, 5, '2024-08-03');

INSERT INTO Orders66 VALUES (4, 4, 5, 1, '2024-09-01');

INSERT INTO Orders66 VALUES (5, 5, 6, 2, '2023-08-01');

INSERT INTO Delivery66 VALUES (1, 1, 1);

INSERT INTO Delivery66 VALUES (2, 2, 2);

INSERT INTO Delivery66 VALUES (3, 3, 3);

Answers:

SELECT DISTINCT Customers66.\* FROM Customers66 JOIN Orders66 ON Customers66.custid = Orders66.custid;

SELECT DISTINCT Customers66.\* FROM Customers66 JOIN Delivery66 ON Customers66.custid = Delivery66.custid;

SELECT Customers66.custname, Orders66.orderdate FROM Customers66 JOIN Orders66 ON Customers66.custid = Orders66.custid WHERE Customers66.custname LIKE 'J%';

SELECT Items66.itemname, Items66.price FROM Items66 JOIN Orders66 ON Items66.itemid = Orders66.itemid JOIN Customers66 ON Orders66.custid = Customers66.custid WHERE Customers66.custname = 'Mickey Mouse';

SELECT Customers66.\* FROM Customers66 JOIN Orders66 ON Customers66.custid = Orders66.custid LEFT JOIN Delivery66 ON Orders66.orderid = Delivery66.orderid WHERE Orders66.orderdate > '2013-01-01' AND Delivery66.orderid IS NULL;

(SELECT itemid FROM Orders66) UNION (SELECT itemid FROM Items66 WHERE itemid NOT IN (SELECT itemid FROM Orders66));

SELECT custname FROM Customers66 WHERE custid IN ( SELECT custid FROM Orders66 INTERSECT SELECT custid FROM Delivery66 );

SELECT custname FROM Customers66 WHERE custid IN ( SELECT custid FROM Orders66 MINUS SELECT custid FROM Delivery66 );

SELECT custname FROM Customers66 WHERE custid = ( SELECT custid FROM Orders66 GROUP BY custid ORDER BY COUNT(orderid) DESC LIMIT 1 );

SELECT DISTINCT Customers66.\* FROM Customers66 JOIN Orders66 ON Customers66.custid = Orders66.custid JOIN Items66 ON Orders66.itemid = Items66.itemid WHERE Orders66.quantity \* Items66.price > 5000;

SELECT DISTINCT Customers66.custname, Customers66.address FROM Customers66 WHERE Customers66.custid NOT IN ( SELECT Orders66.custid FROM Orders66 JOIN Items66 ON Orders66.itemid = Items66.itemid WHERE Items66.itemname = 'Samsung Galaxy S4' );

SELECT Customers66.\*, Orders66.\* FROM Customers66 LEFT JOIN Orders66 ON Customers66.custid = Orders66.custid;

SELECT Customers66.\*, Orders66.\* FROM Customers66 RIGHT JOIN Orders66 ON Customers66.custid = Orders66.custid;

SELECT state, COUNT(\*) AS num\_customers FROM Customers66 GROUP BY state;

SELECT category, itemname, price FROM Items66 GROUP BY category, itemid, itemname, price HAVING price > (SELECT AVG(price) FROM Items66);