Task:5 Date: 9/9/25

WRITING JOIN QUERIES, EQUIVALENT AND/OR RECURSIVE QUERIES.

Aim:- To implement and execute JOIN queries equivalent queries, and recursive queries using Mobile database

INNER JION:-

Returns records that matching values in both tables.

```
SELECT m.phone_id, m.brand, m.model,
    S.ram, S.storage, S.battery.
FROM Mobile phones m
INNER JOIN phone specifications
```

phone_id	brand	model	price
1	Realme	11pro	30,000
2	Redmi	10pro	15,000
3	vivo	T3 pro.	25,000

```
INNER JOIN phone specifications s
ON m.phone_id = s.phone_id;
```

phone_id	ram	storage	battery
1	16GB	256GB	5000mah
2	8GB	128GB	4500mah
3	12GB	256GB	5500mah

LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table.

```
SELECT m.phone_id, m.brand, m.model,
       S.ram, S.storage, S.battery
FROM Mobile phones m
LEFT JOIN phone_specifications S ON m.phone_id
= S.phone_id;
```

phone_id	brand	model	price
1	realme	narzo	30,000
2	Redmi	note	17,000
3	vivo	T3 pro	25,000

ram	storage	battery
16GB	256GB	5000mAh
8GB	128GB	4500mAh
12GB	256GB	5000mAh

RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table.

```
SELECT m.phone_id, m.brand, m.model,
       S.ram, S.storage, S.battery
FROM Mobile phones m
RIGHT JOIN phone specifications S
ON m.phone_id = S.phone_id;
```

phone_id	brand	model	price	ram	storage	battery
1	realme	14Pro	30,000	16GB	256GB	5000 mah
2	Redmi	10Pro	15,000	8GB	128GB	4500 mah
3	vivo	T3 Pro	25,000	12GB	256GB	5100mah

FULL OUTER JOIN:- Return all records when there is a match in either left or Right table

```
SELECT:- m.phone_id, m.brand, m.model, s.ram
         s.storage, s.battery.
FROM Mobile phones m
FULL OUTER JOIN phone Specifications s ON
    m.phone_id = s.phone_id;
```

phone_id	brand	model	price	ram	storage	battery
1	realme	14Pro	30,000	16GB	256GB	5000
2	Redmi	10Pro	15,000	8GB	128GB	4500
3	vivo	T3 Pro	25,000	12GB	256GB	5100

1) JOIN Queies.

CREATE TABLES

```
Create Tables Customer(
    CustID INT PRIMARY KEY;
    CUST_NAME VARCHAR (50) NOT NULL;
);

Create table Mobile(
    Mobile ID INT PRIMARY KEY;
    Brand VARCHAR (50) NOT NULL;
```

```
    Mobile VARCHAR (50) NOT NULL;
    Price DECIMAL (10,2) CHECK (price > 30,000);
);
CREATE TABLE purchase (
    Purchase ID INT PRIMARY KEY;
    CUST ID NOT NULL;
    Mobile ID NOT NULL;
Quantity INT CHECK (Quantity 20);
Purchase Date DATE DEFAULT CURRENT DATE;
    FOREIGN KEY (CUST ID);
    REFERENCES Mobiles (Mobile ID)
);
    CREATE TABLE Payment (
        Payment ID INT PRIMARY KEY,
        Purchase ID INT UNIQUE,
        AMOUNT DECIMAL (10,2) NOT NULL,
        Payment Date DATE DEFAULT,
        CURRENT_ DATE,
    Payment method VARCHAR (20)
    CHECK (payment method IN 'UPI', 'card',
            'netbanking', 'COD');
        FOREIGN KEY (purchase ID)
    REFERENCES Purches (purchase ID)
    );
```

2. INSERT SAMPLE DATA.

```
Insert into Mobile values (1, Android items);
(101,' Realme '),
(102,' Redmi '),
(103,' vivo '),
Insert INTO Mobile value payment values
(1,' Realme ';101),
(2,' Redmi ';102),
(3,' vivo ',101),
(4,' poco ',103),
(5,' 1900 ';104); -- Invalid phoneID for
                    OUTERJOIN example
INSERT INTO Review values
('c1',' Database System';101);
('c2',' Good product & worth it';101);
('c3',' product its good';102),
('c4', afford to buy it';103);
INSERT INTO Payments values (30,000;15000,
                             25000, 2021-01-19)
' Rao (reated.    Completed ');
```

Result: Record inserted Successfully.

3. JOIN QUERIES :

(a) INNER JOIN

```
SELECT m.phone_id -, m.brand , m.model , s.ram,
  s. storage , s.battery
FROM Mobile phone m
INNER JOIN phone specification ON .m.phone_id=
      s phone_id;
```

(b) LEFT JOIN

```
SELECT 'm.phone_id , m.brand, m.model, s.ram,
    S.storage , s . battery.
FROM Mobile phones.m
LEFT JOIN phone specification ON . m.phone_id.
      = s phone_id;
```

(c) RIGHT JOIN

```
SELECT m.phone_id, m.brand , m. model,
    S.ram , S. storage , s . battery
FROM Mobile phones m
RIGHT JOIN phone specification
    ON m. phone_id = s. phone_id;
```

(d) FULL OUTER JOIN:-

```
SELECT :- m.phone_id , m.brand , m.model,
  'S.ram . S. storage , s . battery
FROM Mobile phones M
FULL OUTER JOIN .. phone . specifications . ON
      m . phone_id = s. phone_id;
```

4. Equivalent Queries

```
SELECT s.: Sta.MobileName, d.M. ModelName
FROM Mobile.phone
JOIN BRand ON s.·phoneID < M. PhoneID;
-. Using Subquery
SELECT MobileName,
    (SELECT BrandName FROM .BRand B.
    WHERE M.PhoneID = S.PhoneID) As ModelName
FROM Mobile.phone;
```

Purchases Hierarchy

5) RECURSIVE QUERY (~~Course Hierarchy~~)

```
WITH .RECURSIVE .purchases As.
SELECT paymentID, ~~purchases~~ ID
                  preReq.
FROM ~~Prerequisites~~ phone
     Phone
UNION
SELECT P. paymentID, C. preReqID
                        phone
FROM .prerequisites P.
JOIN payment Hierarchy ( ON P. preReq ID =
                            phone
                            payment ID
)
SELECT * FROM . payment Hierarchy.
```

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	-
TOTAL (20)	15
WITH DATE	M

Result:- Thus, the implementation of SQL commands using Joins and recursive queries are executed successfully.