Task-5

WRITING JOIN QUERIES, EQUIVILENT, AND/OR. Recursive Queries.

Aim:- To implement and execute join queries equivalent queries, and recusive queries using mobile database.

INNER JOIN:-

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	14 pro	30,000
2	Redmi	10pro	15,000
3	vivo	T3 pro	25,000

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

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

LEFT (OUTER) JOINS: Return all records from the 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. on m.phone_id = s.phoneid.

phone_id	brand	model	price	ram	storage	battery
1	realme	14pro	30,000	16GB	256GB	5000mAh
2	Redmi	10pro	15,000	8GB	128GB	4500mah
3	vivo	T3pro	25,000	12GB	256GB	5500mah

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 on m.phone_id =
    s.phone_id;
```

phone id	brand	model	price	ram	storage	battery
1	realme	14pro	30,000	16GB	256GB	5000mAh
2	redmi	10pro	15,000	8GB	128GB	4500mAh
3	vivo	T3pro	25,000	12GB	256GB	5500mah

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 specification on m.phone_id =
    s.phone_id;
```

phone_id	brand	model	price	ram	storage	battery
1	realme	14pro	30,000	16GB	256GB	5000mah
2	redmi	10pro	15,000	8GB	128GB	4500mah
3	vivo	T3pro	25,000	12GB	256GB	5500mah

9 JOIN Queries

CREATE TABLES

```
create Tables customer (
  custID INT PRIMARY KEY;
  custname VARCHAR(50) NOTNULL;
);
create table mobile(
mobile ID INT PRIMARY KEY;
Brand VARCHAR (50) NOT (NULL) 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) NOTNULL,
payment date DATE DEFAULT,
CURRENT-DATE
Payment method VARCHAR (20)
CHECK (Payment method IN 'UP' and netbanking; '1000);
FOREIGN KEY (Purchase ID)
REFERENCES purchase (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, 'Redmi', 101),
(2, 'Redmi', 102);
(3, 'vivo', 101),
(4, 'poco', 103);
(5, 'Iqoo', 104); -- Invalid phone ID 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 payment values (30,000, 15000, 25000,
2025-08-19)
1Row (~~re~~rated rated completed');
```

Result : ~~Re~~cord 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 =
sphone_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.phoneid = s.phone_id;

(c) RIGHT JOIN.

SELECT m.phone_id, m.brand, m.mobl.-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 : ~~ARE~~stu mobile name, m.model name,
FROM mobile phone.
JOIN Brand ON s.phone ID, m.phone ID : using software

SELECT Mobile Name;
(SELECT Brand name FROM Brand, WHERE m.phone
ID., s.phone ID) AS model name FROM mobile phone;

(5) RECURSIVE QUERY (purchases lifecycle)

WITH RECURSIVE purchases AC.
SELECT payment ID, phone ID
FROM pre requestites
UNION.
SELECT, payment ID, phone ID

```
FROM pre_requesites p
JOIN payment_Hierachy on p.prereq_ID = phone
                         PaymentID
)
  SELECT * FROM payment_Hierachy
```

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	
RECORD (5)	4
TOTAL (20)	14
SIGN WITH DATE	

11/9

Result: Thus, the implementation DDL commands using joins and recursive queries are executed successfully.