9/9/25

TASK 5

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 making values in both tables

```
SELECT m.phone_id, m.brand, m.model,
s.ram, s.storage, s.battery,
FROM join phone specir
```

phone_id	brand	model	price
1	Realme	14Pro	30,000
2	Redmi	10Pro	15,000
3	Vivo	T3 Pro	25,000

```
INNER Join phone specifications
ON m.phone_id = s.phone_id.
```

phone_id	ram	storage	battery
1	16GB	256GB	5000mah
2	8GB	128GB	4100mah
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_specification, ON m.phone_id
=s.phone_id;
```

phone_id	brand	model	price
1	realme	14pro	30,000
2	Redmi	10pro	45,000
3	vivo	T3pro	25,000

ram	storage	battery
6GB	256 GB	5000 mAh
8GB	128 GB	4500 mah
12GB	256GB	5500 mah

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_specification s

ON m.phone_id = s.phone_id;

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

FULL OUTER JOIN :- Return all record 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	batter
1	realme	14pro	30,000	16GB	256GB	5000
2	Redmi	10pro	15,000	8GB	128GB	4500
3	Vivo	T3pro	25,000	12GB	256GB	5500

1) Join Queries:

CREATE TABLES

```
create tables customer (
   cust ID INT PRIMARY KEY;
   cust.NAME .VARCHAR (50) NOT NULL;
);

create table mobile (
Mobile ID INT PRIMARY KEY:
Brand VARCHAR(50) NOT NULL;
Model VARCHAR (50) NOT NULL:
 price DECIMAL (10,2) CHECK price 30,000
```

2) INSERT SAMPLE DATA:

```
Insert into mobile value (', Android items):
(101, 'Realme');
(102, 'Redmi');
(103;vivo')

INSERT into mobile value Payment values
(', 'Realme', 101),
(2, 'Redmi', 102),
(3, 'Vivo', 103),
(5, '1900', 104) - invalid PhoneID for outerjoin
```

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 values (30,000,15000,2500,

2025-08-19)

'Rao (related. Completed.);

Result: Record Inserted successfully

4) equivalent Queries

SELECT : sta Mobile Name, N. model name

FROM mobile phone

JOIN Brand ON : phone ID = M. Phone ID;

-using subQuery

SELECT Mobile Name,

(select Brand name FROM BRAND B.

WHERE N. Phone ID = S. Phone ID) as model

name.

FROM mobile phone.

5) RECURSIVE QUERY:

```
WITH RECURSIVE purchase_An
SELECT PaymentID, PhoneID
FROM Prerequisites
UNION
SELECT, PaymentID, C.PhoneID
FROM .Prerequisites
JOIN Payment Hierachy ON P.PhoneID=
)                          PaymentID
SELECT * FROM .Payment Hierachy
```

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

Result:- Thus the implementation of SQL command using joins and recursive queries are executed successfully.