9/9/25

TASK : 5

WRITING TO IN QUERIES EQUIVALENT AND/ OR RECURSIVE QUERIES

Aim: To implement and execute join Queries equivalent Queries and recursive 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.batteries.
FROM mobile phones m
INNER join phone_specify. s
```

phone id	brand	model	price
1	Realme	10 pro	30,000
2	Redmi	10 pro	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	8 GB	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, ON m.phone_id
= s.phone_id.

Phone_id	brand	model	Price
1	Realme	14 Pro	30,000
2	Redmi	10pro	15,000
3	vivo	T8 Pro	25,000

ram	Storage	battery
16 GB	256 GB	5000mah
12GB	128GB	500mach
8GB	128GB	4500 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.modul, 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 JION :- Return all records when there is a match is 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 JION phone specification 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	T3Pro	25,000	12GB	256GB	5500mah

1. JION 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);
);
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;
    FOREGIN 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 2, method INT 'UP' and );
    NEt banking; ('100');
    FOREIGN KEY (purchase ID)
REFERENCES Purchases (purchase ID)
```

2. INSERT SAMPLE DATA :-

```
Insert into mobile values (' Android items);
    (101, 'Realme'),
    (102, 'Redmi'),
    (103, 'VIVO');
```

INERST INTO mobile value payment values.

(1, 'realme'; 101),1

(2, 'Redmi': ; 102);

(3, 'VIVO'; 101),

(4, 'POLO'; 103),

(5, '1900'; 104),- - invalid phone ID for

OUTER JION Example.

INSERT INTO REVIED values.

(' (1'; 'Database system; 101);

(' (2'; 'Good product & worth if ', 101);

(' (3'; 'Product it's good' 102),

(' (4'; afford to buy if '103);

INSERT INTO Payment values (30,000 ; 15000, 25,000,
2025-08-19)

1 ROW created completed');

Result: Reward invested succesfully

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

(c) RIGHT JOIN:

```
SELELT m.phone_Id, m.brand, m.model, s.ram,
s.storage s.battery
FROM mobile_phones m.
RIGHT JOIN, phone_specification ON mobile_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.no. mobile Name, m.model.
Name,
FROM mobile_phones
JOIN Brand ONS. phone ID, m.phone
ID : using subway
```

SELECT mobile Name ;

(SELECT Brand Name FROM Brand, COUNT(*) FROM phone ID.s s. phone ID) AS model Name FROM mobile phone ;

RECURSIVE QUERY (PURCHASES Hierachy).

WITH RECURSIVE Purchases AS

SELECT Payment ID, phone ID

FROM PRE requestsites.

UNION

SELECT, payment ID, phone ID

FROM PRE request sites P

JOIN payment Hi reachy ION P. preveq ID =

phone payment ID

)

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 commands using joins recursive queries are executed successfully.