09/09/25

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

A 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 mobiles m

INNER JOIN Phone specify

Phone id	brand	model	Price
1	Realme	14 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	5000 MAH
2	8GB	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.Phone-id

Phone-id	brand	model	Price	ram	storage	battery
1	Real me	14 Pro	30000	16GB	256GB	3000mAH
2	Redmi	10 Pro	15,000	8GB	128GB	4500mAH
3	vivo	T3 Pro	25000	12GB	256GB	5500mAH

RIGHT (outer) Join) :- Return all records from the right table and the matched records from 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	14 Pro	30000	16GB	256GB	5000mah
2	redmi	10 Pro	15000	8GB	128GB	4500MAH
3	vivo	T3 Pro	25000	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	14 Pro	30,000	16GB	256GB	5000MAH
2	redme	10 Pro	15,000	8GB	128GB	4500MAH
3	vivo	T3 Pro	25,000	12GB	256GB	5500MAH

1. JOIN Queries

CRRATE TABLES

```
create tables customer (
cust.ID INT PRIMARY KEY;
custNAME VARCHR (50) NOT NULL;
);
create table mobile (
mobile ID INT PRIMARY KEY;
Brand VARCHR (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 Defualt Current DATE;
FORIEGN 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 Default.
current_Date,
Payment method VARCHAR (20)
CHECK (Payment method INT . Netbanking ; (100));
Foriegn key (Purchase ID)
Reference Purchases (Purchase ID)
);
```

2. INSERT SAMPLE DATA

```
insert into mobile values ('Andriod item);
  (101, 'Real-me');
  (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 worthit 101);
  ('C3', 'Product its good 102);
  ('C4', 'afford to buy it' 103);
```

```
Insert INTO Payment values (30000. 15000, 25000,
                            2025-08-11)

  1 Row (· 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.phoneid;
```

b) LEFTJOIN

```
SELECT mphone-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.Mobile Name   M.Model Name
  From Mobile Phone
 JOIN Brand ON s.phone ID = M. Phone ID;
   using subquery
 SELECT Mobile Name;
     (SELECT Brand Name from Brand B.
     where M. Phone ID = S. Phone.ID) As Model Name
   from Mobile phone;
```

5) Recursive Query (purchase hierarchy)

```
 with Recursive Purchases
  SELECT Payment ID, Phone ID
   From Prerequestres
   UNION
   SELECT r. Payment ID. c.Phone ID
     FROM Prereques P
```

```
JOIN Payment Hierarchy ON . P.Phone ID = Payment ID
)
SELECT * FROM Payment Hierarchy ;
```

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	3
RECORD (5)	
TOTAL (20)	13
SIGN WITH DATE	[illegible] 10/9

Result:- The implementation of SQL Commands using joins and recursive queries are executed successfully