09/09/2025

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 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	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	8GB	128GB	4500mAh
3	12GB	256 GB	5500mAh

LEFT (OUTER) JOIN: 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
1	realme	14 pro	30,000	16GB
2	Redmi	10 pro	15,000	8 GB
3	vivo	T3 pro	25,000	12GB

Storage	battery
256GB	5000mAh
128GB	4500mAh
256 GB	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	T3 pro	25,000	12GB	256GB	5500MAh

Full OR 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	5000mAh
2	redmi	10pro	15,000	8GB	128GB	4500mAh
3	Vivo	T3 pro	25,000	12GB	256GB	5500mAh

1) JOIN Queries

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;
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;
   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 default;
   current:= DATE;
   Payment method VARCHAR(20)
CHECK(payment method INT 'ID' Net banking
                              100);
   Foriegn key(purchase ID)
      References purchase(purchase ID)
   );
```

2) INSERT SAMPLE DATA.

```
Insert into mobile values('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,'iqoo',104); - invalid phone ID for outer
                      Join example.
```

Insert INTO Review values

('C1' : 'Database system : 101');

('C2' : Good product & worth it ; 101);

('C3' : product it's good ; '102');

('C4' : afford to buy it '103');

Insert into values (30,000, 15,000, 25,000, 2025-05-19)

1 ROW (realed completed);

Result:- Reward inserted successfully.

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) Left Join :-

Select m.phone-id, m.brand, m.model, s.ram, s.stora
-ge, 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.stora
-ge is battery.

From mobile phones.m.

Full outer Join phone specifications > ON'
m.phone_id = s.phone_id;

4) Equivalent Queries:

Select : mobile name, model name from mobile phone.
Join brand-ID, phone ID, m phone ID
using subsequery
Select mobile name.
(Select brand name from brand where em.phone ID=
S.phone ID) model name from mobile phone.

5) Recursive query (purchase)

```
with recursive purchase ID
 select payment ID, phone ID
   From pre requsites
UNION
Select, payment ID, phone ID
From pre requisites p
Join payment ooohy on phone ID = payment ID
)
Select * from, payment Hierarchy
```

Result: Thus, the implementation of sq. commands using Joins ~~Points~~ and recursive Queries are executed successfully.

VEL TECH	
EX No.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD	
TOTAL	14
SIGNATH DATE	

15/9/25