9/9/25 Task 5.

~~Task~~

Writing join Queries .equivalent AND/OR Recursive queries.

Aim:- To implement and execute join queries equivalent queries and recursive queries using mobile phone data base.

Inner join:-

Returns Records that matching values in both tables select.mobile_id, mobile name, mobile 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.	256GB	5500mAh.

Left (outer) John:- Return all records from the table and the matched records from the right table.

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.	realme	14 pro	30,000	16 GB	256GB	5000mAh
2.	Realmi	10 pro	15,000	8GB	128GB	4500MAh
3.	vivo	T3 pro	25,000	12GB	256GB	5500MAh.

Right (outer John):- 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.

Elect m.phone_id, m.brand, m_model, s.ram,

S.storage, s.battery.
From mobile phones.m
Full outer join phone specifications on m.pho
-ne_id- s.phone_id;

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

1. ~~Joins~~ 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);
);
create table purchase (
purchase ID INT primary key;
cust ID NOT NULL;
mobile ID NOT NULL;
Quantity INT check (quantity 20);
purchase data DATE DE FAULT CURRENT DATE;
```

```
    FOREGIN KEY (CUST ID);
    References mobiles (mobile ID)
);
    CREATE TABLE Payments
    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 method INT 'ID' Net bank
          -ing 100);
    Foriegn key (Purchase ID)
    References purches (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 & worthit': '101');
('C3': 'Product its good': '102');
('C4': 'afford to buy it '103');

Insert into values (39,000, 15,000, 25000, 2025-05-19

1 Row (related completed);

Result: Reward inserted successfully.

3. ⬆ JOIN Queries:-

'a'. inner join
select m.phone id, m.brand, m.model, sram,
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.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 is battery.
From mobile phones m.
Full outer join phone specifications 'ON.
m.phone_id = s.phone_id;

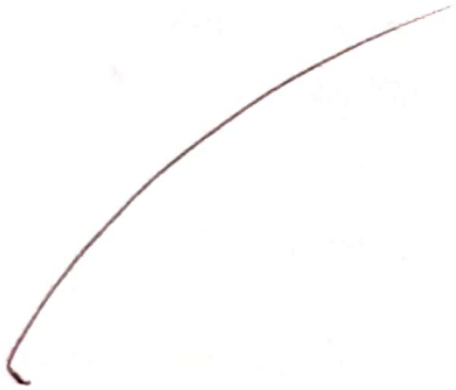
4. Equivalent Queries

Select : mobile name, model name from mobile phone.
~~John~~ join brand ID, phone ID, m.phone ID;
using subsequery.
select mobile name,
(select brandname from brand where m.phone ID
= s phone ID) model name from mobile phone;

5. Recursive Query (B Purchases):-

With recursive purchase ID
select payment ID, phone ID
From prerequisites
Union
select, Payment ID, Phone ID,

from prerequisites p
join payment archy on phone ID = pa
-ment I.
);
select from payment Hieraicky.



VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	1
RECORD (5)	1
TOTAL (20)	14
SIGN WITH DATE	9/9/24

Result:-

Thus, the implementation of ~~sql command~~ using joins and recusive queries are executed successfully.