TASK-5 Date: 9/09/2025

Writing Join Queries, Equivalent AND/OR recursive Queries.

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

Inner join:-

Returns record's 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 specitications .

phone-id	baand	model	price
1	Realme	14pro	30,000
2	Redmi	15pro	15,000
3	vivo	y75	25,000

inner join phone specitications

on m_phone_id = sphone -id;

phoneid	Ram	Storage	battery
1	16GB	256GB	5000mAh
2	8GB	128GB	4000mAh
3	12GB	256GB	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.pho-phoid.
```

phone_id	brand	model	price	ram	storage	battery
1	realme	11pro	30,000	16GB	256GB	5000mAh
2	redmi	10pro	15,000	8GB	128GB	4500mAh
3	vivo	Y75	25,000	12GB	256GB	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 phone.m
right join phone specifications
on. m.phone-id = s.phone-id;
```

phone-id	brand	model	price	ram	storage	battery
1	realme	11pro	30000	16GB	256GB	5000mAh
2	redmi	10pro	15,000	8GB	128GB	4500mAh
3	vivo	Y75	25,000	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 phone m
full outer join phone specification 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	Y75	25,000	12GB	256GB	5500mAh

Join Queries:

```
CREATE TABLES
Create Table CUSTOMER(
CustID INT PRIMARY KEY;
CustName VARCHAR(50) NOT NULL;
);

Create table mobile(
mobileID INT PRIMARY KEY;
Brand VARCHAR(50) NOT NULL;
model VARCHAR(50) NOT NULL;
price decimal(10.2) check (price 10000);
);
```

```
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 Date default!
current - DATE;

payment method VARCHAR(20)
CHECK (payment method IN 'ID' net
                          banking)00);

Foriegn Foriegn key (purchase ID)
Reference purchase (purchase ID)
);
```

2. INSERT SIMPLE DATA:

```
insert into mobile values ('Android Item');
(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,'oppo',104);..invalid phone Id for
                outer join example
```

```
Insert INTO Review values
('C1','database system','101');
('C2','Good product worth it',101);
('C3','product not good','102');
('C4','Afford to buy it','103');
```

```
Insert into values (30,000, 15,000, 25,000,
                    2025-08-
1 Row (realed completed);
```

Result: Record Inserted Sucessfully.

5. Join Queries:-

a. Inner Join:

```
select m.phoneid, 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 phone 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.
fruom mobile phone 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.
fruom 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 sub sequery
Select mobile name
(Select brand name from brand where m.
phone-ID = sphone ID) model name from
mobile phone;

5. Recursive Query (purchases):

with recursive purchase ID.
Select payment ID, phone ID,
from prerequsties.
UNION
Select , payment ID, phone ID
from prerequisities p.
Join payment 'archy on phone ID =
payment ID
)

Select * from , payment ;

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	—
TOTAL (20)	19
SIGN WITH DATE	

Result: Thus, the implementation of [redacted] using joins and recursive queries are executed suessfully. 9/5/24