09/9/25 TASK-5

WRITING Join QUERIES EQUIVALENT AND /OR RECURSIVE QUERISE.

Aim: To implement and execute JOIN queries equivalent queries and recursive queries using mobile database

INNERTION

Return records that matching values in both tables

SELECT m.Phone-id m brand m.model,
s.ram s. storage , s . battery.
FROM Mobile Phone m
INNIER JOIN Phone SpeciR.

Phone-id	brand	model	Price
1.	Realme	14Pro	30,000
2.	Redmi	10Pro	15,000
3.	vivo	T3Pro	25,000

INNERTION 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	5500 mAh

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
   Phone.specification .ON.m .Phone - id = s.Phone-id;
```

Phone-id	brand	model	Price.
1.	realme	14 Pro	30,000
2.	Redmi	10 Pro	15,000
3.	vivo	73 Pro	25,000

ram	storage	battery
16GB	256GB	5000mAh
8GB	128GB	4500mAh
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.bran d , m . model
s.ram s.storage ,s.storage ,s.battery
FROM mobile Phones m
RIGHT JOIN Phone Specifications
ON M.Phone -id = S.Phone -id;
```

Phone-id	brand	model	Price	rom	storage	battery
1.	Realme	14Pro	30,000	12GB	256GB	5000 mah
2.	Redmi	10Pro	15,000	8GB	128GB	4500 mah
3.	vivo	73Pro	25,000	12GB	256GB	500mah

~~fa~~ 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.rom s.storage s.battery

FROM Mobile Phones M

full OUTER 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	5000
2.	Redmi	10Pro	15,000	8GB	128GB	4500
3.	vivo	73Pro	25,000	12GB	256GB	5500

3. Join QUERIES

INNER JOIN

SELECT m.Phone_id m. brand , m, model s.rom 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.rom
 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.rom , s. storage . s battery
 FROM mobile Phones m
 RIGHT mobile Phone specification
 ON m.Phone -id = s.Phone -id;
```

d) FULL OUTER JOIN:-

```
 SELECT: m.Phone -id ,m.brand , m.model
 s.rom . s.Storage is battery
 FROM mobile Phones M
 Full outer join Phones specification
 ON m. Phone -id = s .Phone -id ;
```

4. Equivalent Queries:

```
SELECT s. mobile Name , model . Name
FROM mobile Phone
JOIN Brand ONS : Phone ID= M Phone ID;
```

-using subQuery.

```
SELECT 'mobile name'
(SELECT Brand name FROM Brand B.
WHERE M.Phone ID =S.Phone ,D) AS
model name FROM mobile Phone;
```

5. RECURSIVE QUERY (Purchase Hierachy).

```
WITH RECURSIVE Purchase AS
SELECT Payment ID Phone ID
FROM Phone
UNION
SELECT Payment ID Phone ID
FROM Prerequirment .P.
JOIN Payment Hierachy ON P.Phone ID =
Payment ID
)
SELECT * FROM ,Payment Hierarchy
```

VELTECH	
EX No.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	4
VIVA VOCE (5)	5
RECORD (5)	-
TOTAL (20)	16
SIGN WITH DATE	

[illegible] 22/4/25

Result: Thus the implementation of commands using Points and Recursive Queries are executed successfully.