9/9/25 Task 5: Writing Join Queries, Equivalent AND/OR Recursive Queries

Aim: To implement and Execute Join queries Equivalent queries and recursive queries using Hospital Management database.

Inner Join:

Return's records that matching values in both tables

select patient_id, patient name, patient_bill

From patients.

INNER Join patient specifications

patient_id	patient Name	patient_Bill	patient address
1	Vinod	30,000	rajam
2	Virat	40,000	Kadapa
3	Vikram	20,000	Vellore
4	Sriram	50,000	Vizag

INNER JOIN patient specifications

ON. m. patient_id = s. patient-id

LEFT Outer Join: Return all records from the table, & the matched records from the right table.

select m.patient_id, m.patient_name, m patient_bill,

patient_address, patient.phone no.

From patients

LEFT Join ~~phone~~ patient specification. ON. m. patient_id = s.patient_id

Patient id	Patient name	Patient_Bill	Patient address	patient phone
1.	Vinod	30,000	rajam	7992892
2	Virat	40,000	Kadapa	6634521
3	Vikram	20,000	Nellore	99877694

Right (OUTER) JOIN : Return all records from the right table , and the matched records from the left table
select m. patient _ id , m. patient _name , patient_bill, patient_address .

From patients
Right join patients specifications .
ON.. m.patient _id = s.patient - id ;

Patient Id	patient_name	patient_bill	patient_address	patient_ph.no.
1	vinod	30,000	rajam	7992892
2	virat	40,000	kadapa	6634321
3	Vikram	20,000	Nellore	99897694

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 patients P.
Full outer JOIN ~~patient~~ doctor specializations .
m. patients_id = s.pactient _id ;

Patient -id	patient name	bill	patient_address	patient ph.no.
1	Vinod	30,000	rajam	7799289 2
2	vivek	40,000	kadapa	6634321
3	Vikram	20,000	Nellore	99897 694

JOIN Queries :
Create Tables :
create table patient (
patientId INT PRIMARY KEY;
PatientName VARCHAR [50] NOT NULL;
);
create table Data (
~~doctor~~ Id INT Primary Key;

```
specialization VARCHAR [50] NOT NULL;
salary   INT   [50] NOT NULL;
);
create table medicine (
medicine_ID INT primary key;
  medicine brand VARCHAR (50) NOT NULL;
  medine Name VARCHAR(50) NOT NULL;
    Quantity INT check (Quantity 20).
 purchase date DATE DEFAULT CURRENT DATE;
  FOREIGN KEY (patient ID);
  References medicines (medicine 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),
 Foreign key (purchase ID);
 References purchase (purchase ID)
 );
```

2) INSERT SAMPLE DATA

```
  Insert into patient values ('Desease names');
      (101, 'Dialasis');
      (102, 'B.P');
      (103, 'cancer');
 Insert into patient value Payment values
      (1, 'Dialases', 101);
```

(2, 'B.P', 102);
(3, 'cancer', 101);
(4, 'sugar', 103);
(5, 'malaria', 104);

- invalid patientID for outer Join Example

Insert INTO Review values

('C1': 'Database system: 101');
('C2': Good product & wash it ; 101);
('C3': product its' good; '102');
('C4': afford to buy it '103');

Insert into values (30,000, 15,000, 25,000, 2025-08-19)

1 Row (created completed)

Result : Reward inserted successfully

3) JOIN Queries :

a) Inner Join :
select patient_id, patient name, patient bill, address.
From patient.
Inner Join patient specification on patient_id, patient_name,

b) Left Join :
select patient_id, patient-name, patientaddress, patient-bill,
patient-age.
From patient.
Left Join patient specification on patient_id, patient_name;

c) Right Join :
Select patient-id, patient-name, patient-bill, patient
-address.
From Patient

d) Full outer Join :

```
select : patient_id, patient.name, patient_address, patient_bill.
  From patient .p.
Full outer Join patient Specification "ON"
    p.patient_id = s.patient_id;
```

4) Equivalent Queries :-

```
select : patient name , medicine model name from medicine
brand_Id ; patient.ID, m.patient ID.
using subsquery
select Medicine name
```

5) Recursive Query (purchase)

```
with recursive purchase ID
select payment ID, patient ID
     From prerequisites
UNION
select , payment ID, patient ID
From pre requisites p
Join payment activity on patient ID = patient ID
,
select * from . payment Hierarchy .
```

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	4
RECORD (5)	—
TOTAL (20)	14
SIGN WITH DATE	[illegible] 9/8/23

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