**-- Database: Healthcare\_Analytics**

**-- DROP DATABASE IF EXISTS "Healthcare\_Analytics";**

**CREATE DATABASE "Healthcare\_Analytics"**

**WITH**

**OWNER = postgres**

**ENCODING = 'UTF8'**

**LC\_COLLATE = 'English\_India.1252'**

**LC\_CTYPE = 'English\_India.1252'**

**LOCALE\_PROVIDER = 'libc'**

**TABLESPACE = pg\_default**

**CONNECTION LIMIT = -1**

**IS\_TEMPLATE = False;**

Create table Patients(

patient\_id int unique primary key, Name varchar(20), Age int, Gender varchar(20),

Address varchar(20), Contact\_number bigint

);

select \* from patients;

select \* from Medication;

Select \* from Doctors;

Select \* from Appointments;

Select \* from Diagnosis;

Create table Medication(

Medication\_id int primary key, Diagnosis\_id int, Medication\_name text,

Dosage text, Start\_date date, End\_date date

);

Create table Doctors( Doctor\_id int primary key, Name text, Specialization text,

Exp\_years int, Contact\_number bigint

);

Create table Diagnosis(

Diagnosis\_id int primary key, Patient\_id int, Doctor\_id int, Diagnosis\_date date,

Diagnosis text, Treatment text

);

Create table Appointments(Appointment\_id int primary key, Patient\_id int,

Doctor\_id int, Appointment\_date date, Reason text, Status text);

select \* from patients;

select \* from Medication;

Select \* from Doctors;

Select \* from Appointments;

Select \* from Diagnosis;

**--Inner and Equi Joins**

**--Task: Write a query to fetch details of all completed appointments,**

**--including the patient’s name, doctor’s name, and specialization.**

**--Expected Learning: Demonstrates understanding of Inner Joins and filtering conditions.**

select a.appointment\_id, a.patient\_id, a.doctor\_id, a.appointment\_date, a.reason, a.status,

p.name, d.name, d.specialization from Appointments a

inner join patients p on p.patient\_id = a.patient\_id

inner join doctors d on d.doctor\_id = a.doctor\_id

where a.status = 'Completed';

**--Left Join with Null Handling**

**--Task: Retrieve all patients who have never had an appointment.**

**--Include their name, contact details, and address in the output.**

**--Expected Learning: Use of Left Joins and handling NULL values.**

select p.name, p.contact\_number, p.address, a.status from patients p

left join appointments a on a.patient\_id = p.patient\_id

where a.status not in ('Completed');

**--Right Join and Aggregate Functions**

**--Task: Find the total number of diagnoses for each doctor,**

**--including doctors who haven’t diagnosed any patients.**

**--Display the doctor’s name, specialization, and total diagnoses.**

**--Expected Learning: Utilization of Right Joins with aggregate functions like COUNT().**

select count(d.diagnosis\_id) as total\_diagnosis, doc.name, doc.Specialization from doctors doc

right join diagnosis d on d.doctor\_id = doc.doctor\_id

right join patients p on p.patient\_id = d.patient\_id

group by doc.specialization, doc.name

order by total\_diagnosis;

**--Full Join for Overlapping Data**

**--Task: Write a query to identify mismatches between the appointments and diagnoses tables.**

**--Include all appointments and diagnoses with their corresponding patient and doctor details.**

**--Expected Learning: Handling Full Joins for comparing data across multiple tables.**

Select \* from Appointments;

Select \* from Doctors;

Select \* from Diagnosis;

select \* from patients;

select a.appointment\_id, d.diagnosis\_id, d.diagnosis, p.name, doc.name from diagnosis d

join doctors doc on doc.doctor\_id = d.doctor\_id

join appointments a on a.patient\_id = d.patient\_id

join patients p on p.patient\_id = a.patient\_id

order by d.diagnosis;

**--Window Functions (Ranking and Aggregation)**

**--Task: For each doctor, rank their patients based on the**

**--number of appointments in descending order.**

**--Expected Learning: Application of Ranking Functions such as RANK() or DENSE\_RANK().**

select d.name, count(a.patient\_id) as No\_of\_App,

rank() over(partition by d.name) from doctors d

join appointments a on a.doctor\_id = d.doctor\_id

group by d.name;

**--Conditional Expressions**

**--Task: Write a query to categorize patients by age group**

**--(e.g., 18-30, 31-50, 51+). Count the number of patients in each age group.**

**--Expected Learning: Using CASE statements for conditional logic.**

select \* from patients;

select name, age,

case

when age between 18 and 30 then 'Seniors'

when age between 31 and 50 then 'Super\_Seniors'

else 'Senior\_Citizens'

end Category from patients

order by age;

**--Numeric and String Functions**

**--Task: Retrieve a list of patients whose contact numbers**

**--end with "1234" and display their names in uppercase.**

**--Expected Learning: Use of string functions like UPPER() and LIKE.**

alter table patients

alter column contact\_number type text;

select upper(name), patient\_id, contact\_number from patients

where contact\_number like '%1234';

**--Subqueries for Filtering**

**--Task: Find patients who have only been prescribed "Insulin" in any of their diagnoses.**

**--Expected Learning: Writing Subqueries for advanced filtering.**

select \* from patients;

Select \* from Diagnosis;

select \* from Medication;

select p.name, m.medication\_name from patients p

join diagnosis d on d.patient\_id = p.patient\_id

join medication m on d.diagnosis\_id = m.diagnosis\_id

where m.medication\_name = 'Insulin';

(select m.medication\_name from medication m

where m.medication\_name = 'Insulin')

select p.name, m.medication\_name from patients p

join diagnosis d on d.patient\_id = p.patient\_id

where m.medication\_name =

(select m.medication\_name from medication m

where m.medication\_name = 'Insulin')

**--Date and Time Functions**

**--Task: Calculate the average duration (in days) for**

**--which medications are prescribed for each diagnosis.**

**--Expected Learning: Working with date functions like DATEDIFF().**

select \* from Medication;

Select \* from Diagnosis;

SELECT (start\_date::date - end\_date::date) AS difference\_in\_days from medication;

select m.medication\_id, m.start\_date, m.end\_date, d.diagnosis, (m.start\_date::date - m.end\_date::date)

AS difference\_in\_days from medication m

join diagnosis d on d.diagnosis\_id = m.diagnosis\_id;

**--Complex Joins and Aggregation**

**--Task: Write a query to identify the doctor who has attended the most**

**--unique patients. Include the doctor’s name, specialization, and the count of unique patients.**

**--Expected Learning: Combining Joins, Grouping, and COUNT(DISTINCT).**

Select \* from Doctors;

select \* from patients;

select count(distinct patient\_id), name from patients

group by name;

select d.doctor\_id, d.name, d.specialization, count(distinct p.patient\_id) as Count\_of\_uniquepatient from doctors d

join diagnosis dia on d.doctor\_id = dia.doctor\_id

join patients p on dia.patient\_id = p.patient\_id

group by d.doctor\_id, d.name, d.specialization;