Lab Session 05

5. Data Manipulation and Views

Aim:

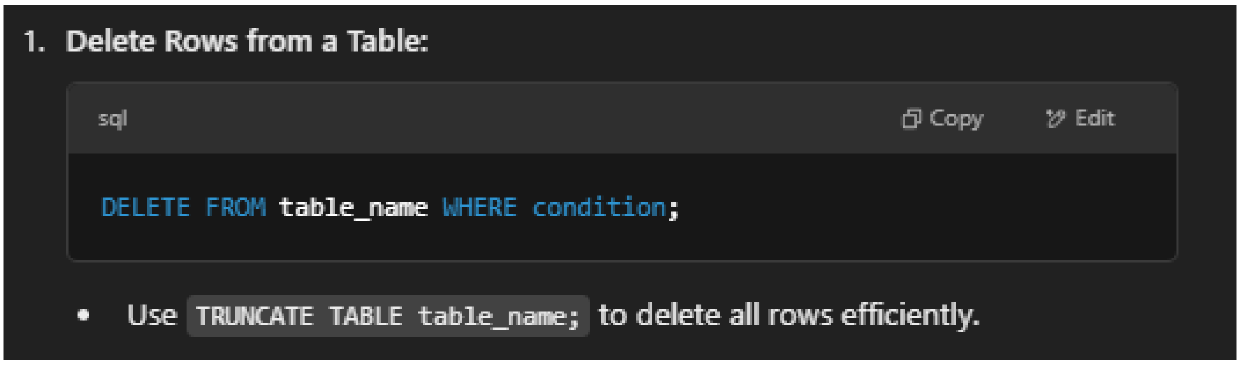
The aim of this experiment is to familiarize students with the concept of DML commands, subqueries, their syntax, and usage for retrieving specific data subsets within queries. Additionally, the lab focuses on DML commands and creating views, which are virtual tables derived from queries, to simplify complex queries and enhance data accessibility.

Description:

The lab experiment involves a procedural description where students gain hands-on experience in implementing DML commands, subqueries and creating views in a database system. The lab begins with an introduction to the DML commands like INSERT, UPDATE, DELETE, subqueries, explaining their purpose and syntax.
Students then proceed to practice writing queries with subqueries, embedding them within SELECT, FROM, and WHERE clauses to retrieve specific subsets of data based on conditions. They learn how to use subqueries to perform calculations, filtering, and data manipulation. The lab also covers the creation of views, allowing students to create virtual tables derived from queries to simplify complex queries and improve data accessibility. Students learn to create, modify, and utilize views in their queries.

Pre Lab-Task:

1) How can you delete rows from a table.

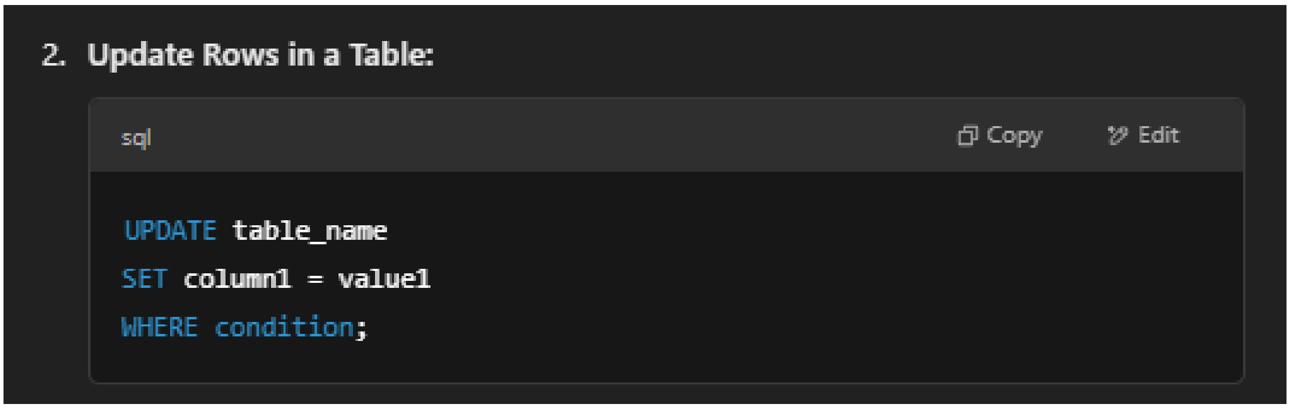


1. **Delete Rows from a Table:**

```sql
DELETE FROM table_name WHERE condition;
```

- Use `TRUNCATE TABLE table_name;` to delete all rows efficiently.

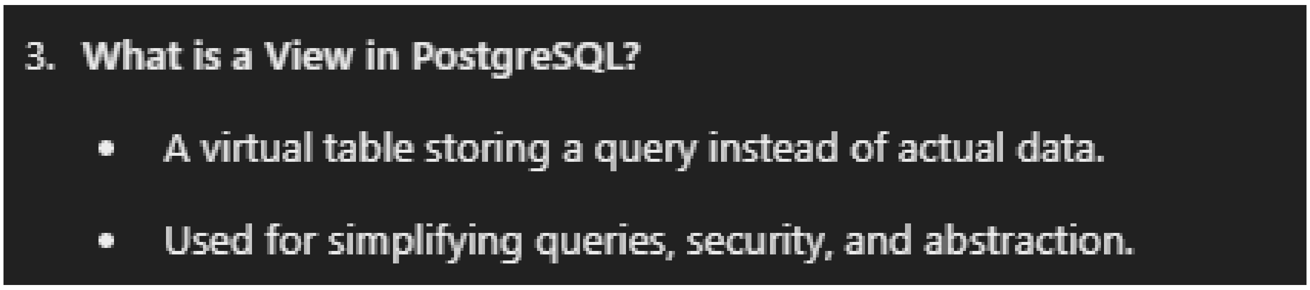
2) How to Update Rows in a Predefined Table



2. Update Rows in a Table:

```sql
UPDATE table_name
SET column1 = value1
WHERE condition;
```

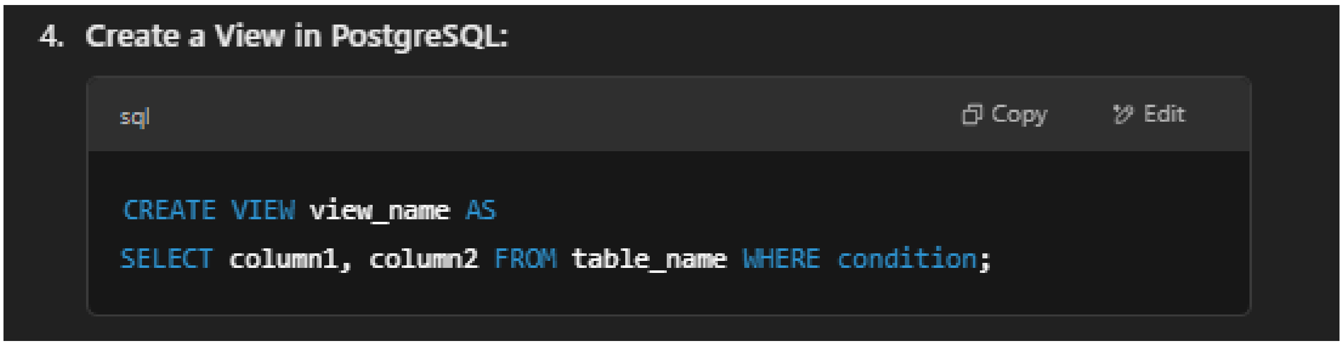
3) What is a View in PostgreSQL, and What Purpose Does it Serve?



3. What is a View in PostgreSQL?

- A virtual table storing a query instead of actual data.
- Used for simplifying queries, security, and abstraction.

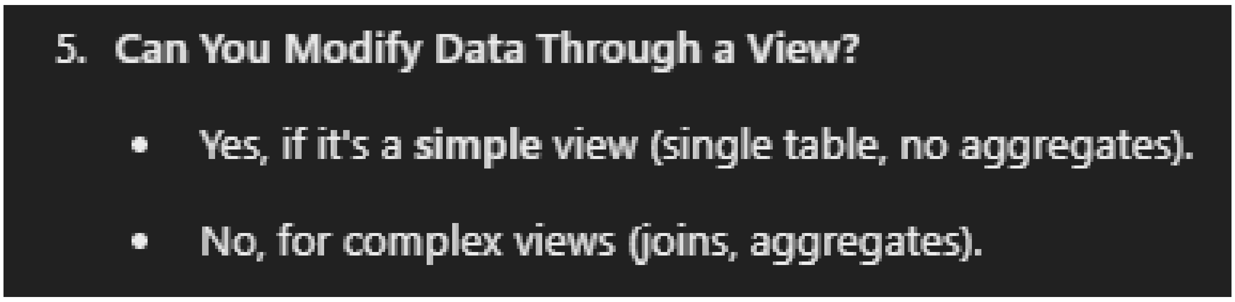
4) How to Create a View in PostgreSQL using the CREATE VIEW statement?4



4. Create a View in PostgreSQL:

```sql
CREATE VIEW view_name AS
SELECT column1, column2 FROM table_name WHERE condition;
```

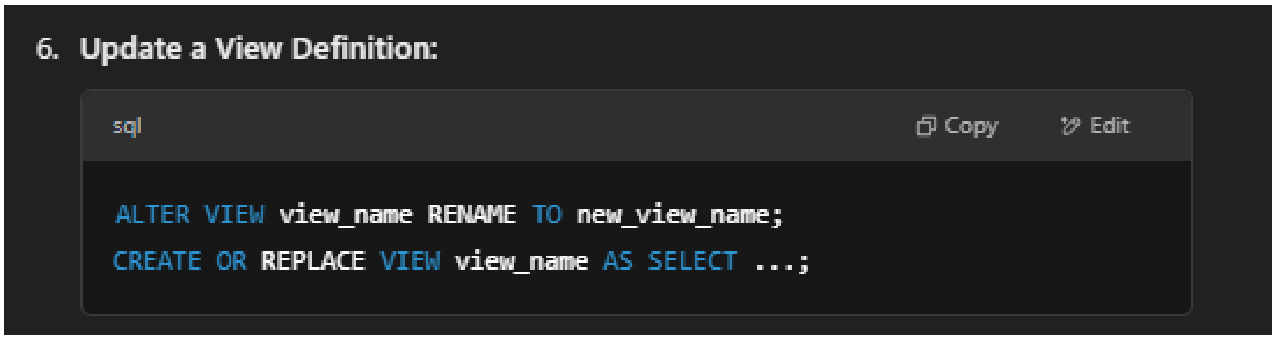
5) Can You Modify Data Through a View in PostgreSQL? Why or Why Not?



5. Can You Modify Data Through a View?

- Yes, if it's a **simple view** (single table, no aggregates).
- No, for complex views (joins, aggregates).

6) How to Update the Definition of a View in PostgreSQL Using the ALTER VIEW Statement

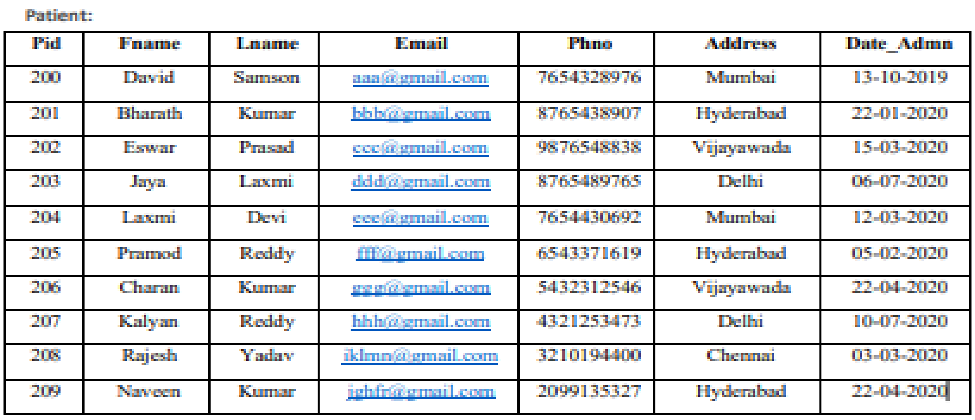


6. **Update a View Definition:**

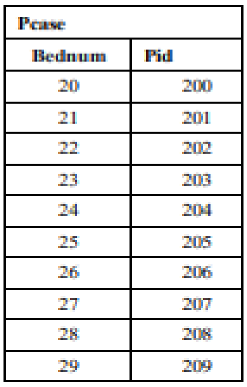
```sql
ALTER VIEW view_name RENAME TO new_view_name;
CREATE OR REPLACE VIEW view_name AS SELECT ...;
```

<u>In Lab Task</u>:

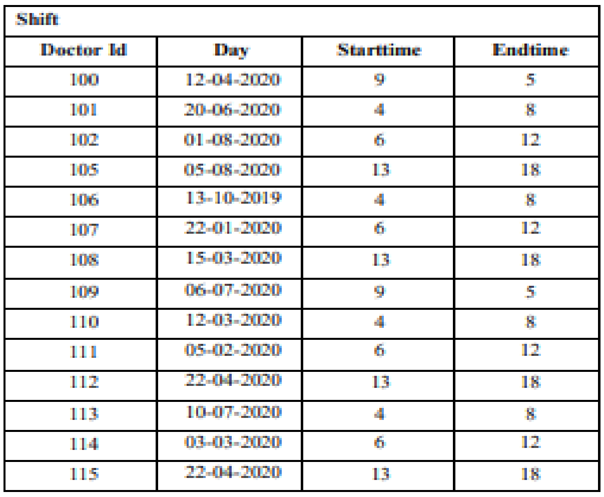
1) Write PostgreSQL queries using DML commands and nested subqueries to analyze the following data.


Patient:

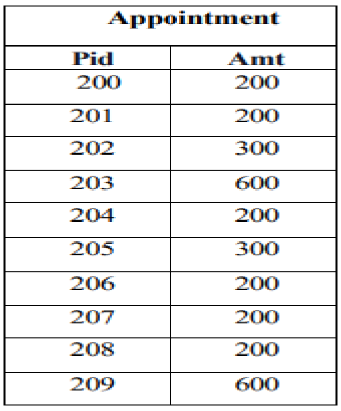
Pid	Fname	Lname	Email	Phno	Address	Date_Admn
200	David	Samson	aaa@gmail.com	7654328976	Mumbai	13-10-2019
201	Bharath	Kumar	bbb@gmail.com	8765438907	Hyderabad	22-01-2020
202	Eswar	Prasad	ccc@gmail.com	9876548838	Vijayawada	15-03-2020
203	Jaya	Laxmi	ddd@gmail.com	8765489765	Delhi	06-07-2020
204	Laxmi	Devi	eee@gmail.com	7654430692	Mumbai	12-03-2020
205	Pramod	Reddy	fff@gmail.com	6543371619	Hyderabad	05-02-2020
206	Charan	Kumar	ggg@gmail.com	5432312546	Vijayawada	22-04-2020
207	Kalyan	Reddy	hhh@gmail.com	4321253473	Delhi	10-07-2020
208	Rajesh	Yadav	iklmn@gmail.com	3210194400	Chennai	03-03-2020
209	Naveen	Kumar	ighfr@gmail.com	2099135327	Hyderabad	22-04-2020


Pcase

Bednum	Pid
20	200
21	201
22	202
23	203
24	204
25	205
26	206
27	207
28	208
29	209


Shift

Doctor Id	Day	Starttime	Endtime
100	12-04-2020	9	5
101	20-06-2020	4	8
102	01-08-2020	6	12
105	05-08-2020	13	18
106	13-10-2019	4	8
107	22-01-2020	6	12
108	15-03-2020	13	18
109	06-07-2020	9	5
110	12-03-2020	4	8
111	05-02-2020	6	12
112	22-04-2020	13	18
113	10-07-2020	4	8
114	03-03-2020	6	12
115	22-04-2020	13	18


Appointment

Pid	Amt
200	200
201	200
202	300
203	600
204	200
205	300
206	200
207	200
208	200
209	600

Questions:

i) Write a query to insert multiple rows in an appointment table as shown above.

```
CREATE TABLE Patient (
   Pid INT PRIMARY KEY,
   Fname VARCHAR(50),
   Lname VARCHAR(50),
   Email VARCHAR(100),
   Phno VARCHAR(15),
   Address VARCHAR(100),
   Date_Admn DATE
);

INSERT INTO Patient (Pid, Fname, Lname, Email, Phno, Address, Date_Admn)
VALUES
(200, 'David', 'Samson', 'aaa@gmail.com', '7654328976', 'Mumbai', '2019-10-13'),
(201, 'Bharath', 'Kumar', 'bbb@gmail.com', '8765438907', 'Hyderabad', '2020-01-22'),
(202, 'Eswar', 'Prasad', 'ccc@gmail.com', '9876548838', 'Vijayawada', '2020-03-15'),
(203, 'Jaya', 'Laxmi', 'ddd@gmail.com', '8765489765', 'Delhi', '2020-07-06'),
(204, 'Laxmi', 'Devi', 'eee@gmail.com', '7654430692', 'Mumbai', '2020-03-12'),
(205, 'Pramod', 'Reddy', 'fff@gmail.com', '6543371619', 'Hyderabad', '2020-02-05'),
(206, 'Charan', 'Kumar', 'ggg@gmail.com', '5432312546', 'Vijayawada', '2020-04-22'),
(207, 'Kalyan', 'Reddy', 'hhh@gmail.com', '4321253473', 'Delhi', '2020-07-10'),
(208, 'Rajesh', 'Yadav', 'iklmn@gmail.com', '3210194400', 'Chennai', '2020-03-03'),
(209, 'Naveen', 'Kumar', 'jphfr@gmail.com', '2099135327', 'Hyderabad', '2020-04-22');
```

```
CREATE TABLE Pcase (
   Bednum INT PRIMARY KEY,
   Pid INT REFERENCES Patient(Pid)
);

INSERT INTO Pcase (Bednum, Pid) VALUES
(20, 200),
(21, 201),
(22, 202),
(23, 203),
(24, 204),
(25, 205),
(26, 206),
(27, 207),
(28, 208),
(29, 209);
```

```
CREATE TABLE Shift (
   Doctor_Id INT PRIMARY KEY,
   Day DATE,
   Starttime INT,
   Endtime INT
);

INSERT INTO Shift (Doctor_Id, Day, Starttime, Endtime) VALUES
(100, '2020-04-12', 9, 5),
(101, '2020-06-20', 4, 8),
(102, '2020-08-08', 6, 12),
(105, '2020-08-05', 13, 18),
(106, '2019-10-13', 4, 8),
(107, '2020-01-22', 6, 12),
(108, '2020-03-15', 13, 18),
(109, '2020-07-06', 9, 5),
(110, '2020-03-12', 4, 8),
(111, '2020-02-05', 6, 12),
(112, '2020-04-22', 13, 18),
(113, '2020-07-10', 4, 8),
(114, '2020-03-03', 6, 12),
(115, '2020-04-22', 13, 18);
```

```
CREATE TABLE Appointment (
   Pid INT REFERENCES Patient(Pid),
   Amt INT
);

INSERT INTO Appointment (Pid, Amt) VALUES
(200, 200),
(201, 200),
(202, 300),
(203, 600),
(204, 200),
(205, 300),
(206, 200),
(207, 200),
(208, 200),
(209, 600);
```

ii) Write a query to update all the rows of personal emails into academic mail addresses and display a patient table.

```
UPDATE Patient
SET Email = CONCAT(SPLIT_PART(Email, '@', 1), '@university.edu')
WHERE Email LIKE '%@gmail.com'
  OR Email LIKE '%@yahoo.com'
  OR Email LIKE '%@outlook.com'
  OR Email LIKE '%@hotmail.com';
```

iii) Write a query for creating a view from patient table.

```
CREATE VIEW PatientView AS SELECT * FROM Patient;
```

iv) Write a POSTGRESQL query to update the Patient view where patientid >3 by using a replacement view statement.

```
CREATE OR REPLACE VIEW PatientView AS
SELECT * FROM Patient WHERE Pid > 3;
SELECT * FROM PatientView;
```

v) Display the patient date of joining into format “dd month yyyy’ for example as “24 March 2020”.

```
SELECT Pid, Fname, Lname,
     TO_CHAR(Date_Admn, 'DD Month YYYY') AS Date_Admn_Formatted
FROM Patient;
```

vi) Write a POSTGRESQL query to display the count of patients who come from the same city.

```
SELECT Address, COUNT(*) AS City_Patient_Count
FROM Patient
GROUP BY Address;
```

vii) Write a POSTGRESQL query to first name of patient who admitted and allotted the bed no.

```
SELECT P.Fname
FROM Patient P
JOIN Pcase PC ON P.Pid = PC.Pid;
```

viii)Write a query to retrieve the first name and last name of patients who have an appointment, and the appointment amount is greater than the average appointment amount.

```
SELECT P.Fname, P.Lname
FROM Patient P
JOIN Appointment A ON P.Pid = A.Pid
WHERE A.Amt > (SELECT AVG(Amt) FROM Appointment);
```

ix) Create a view that shows the number of appointments made by each doctor, along with their total earnings from those appointments.

```
CREATE VIEW DoctorAppointmentView AS
SELECT S.Doctor_Id, COUNT(A.Pid) AS NumAppointments, SUM(A.Amt)
AS TotalEarnings
FROM Shift S
LEFT JOIN Appointment A ON S.Doctor_Id = A.Pid
GROUP BY S.Doctor_Id;
```

OUTPUT:

x) Write a query to find the total number of patients admitted on each date.

```
SELECT Date_Admn, COUNT(*) AS NumAdmitted
FROM Patient
GROUP BY Date_Admn;
```

xi) Create a view that displays the doctor’s ID, day, and total number of appointments scheduled for each day.

```
CREATE VIEW DoctorDailyAppointments AS

SELECT S.Doctor_Id, S.Day, COUNT(A.Pid) AS TotalAppointments

FROM Shift S

LEFT JOIN Pcase P ON P.Pid IN (SELECT Pid FROM Appointment)

LEFT JOIN Appointment A ON P.Pid = A.Pid

GROUP BY S.Doctor_Id, S.Day;
```

xii) Write a query to find the patients who have not made any appointments.

```
SELECT P.Pid, P.Fname, P.Lname, P.Email, P.Address
FROM Patient P
LEFT JOIN Appointment A ON P.Pid = A.Pid
WHERE A.Pid IS NULL;
```

xiii)Create a view that shows the number of patients admitted on each date and the corresponding doctor ID who treated them.

```
CREATE VIEW PatientsAdmittedPerDay AS

SELECT S.Doctor_Id, P.Date_Admn, COUNT(P.Pid) AS TotalPatients

FROM Shift S

JOIN Pcase Pc ON Pc.Pid IN (SELECT Pid FROM Patient)

JOIN Patient P ON Pc.Pid = P.Pid

GROUP BY S.Doctor_Id, P.Date_Admn;
```

xiv)Write a query to calculate the average appointment amount for each doctor, along with the doctor's first name and last name.

```
SELECT D.Doctor_Id, D.Fname, D.Lname, AVG(A.Amt) AS
Avg_Appointment_Amount

FROM Shift S

JOIN Doctor D ON S.Doctor_Id = D.Doctor_Id

JOIN Pcase Pc ON Pc.Pid IN (SELECT Pid FROM Appointment)

JOIN Appointment A ON Pc.Pid = A.Pid

GROUP BY D.Doctor_Id, D.Fname, D.Lname;
```

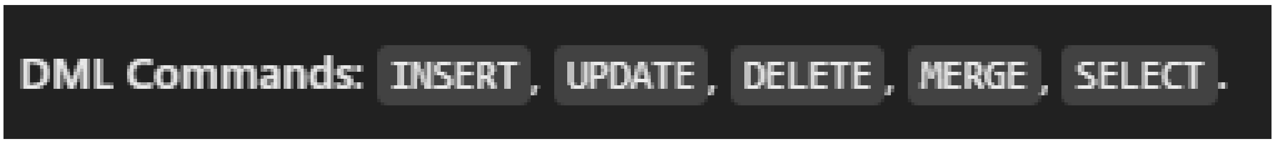
xv) Create a view that displays the details of patients who have been admitted to a particular city more than once.

```
CREATE VIEW RepeatedAdmissions AS
SELECT Address, Pid, Fname, Lname, COUNT(*) AS Admission_Count
FROM Patient
GROUP BY Address, Pid, Fname, Lname
HAVING COUNT(*) > 1;

SELECT * FROM RepeatedAdmissions;
```

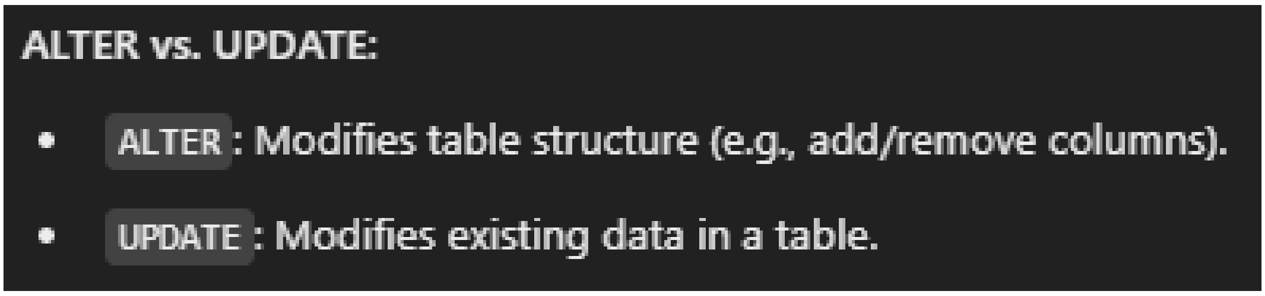
Viva-Voce Questions (In-Lab):

1) List various DML commands.



DML Commands: `INSERT`, `UPDATE`, `DELETE`, `MERGE`, `SELECT`.

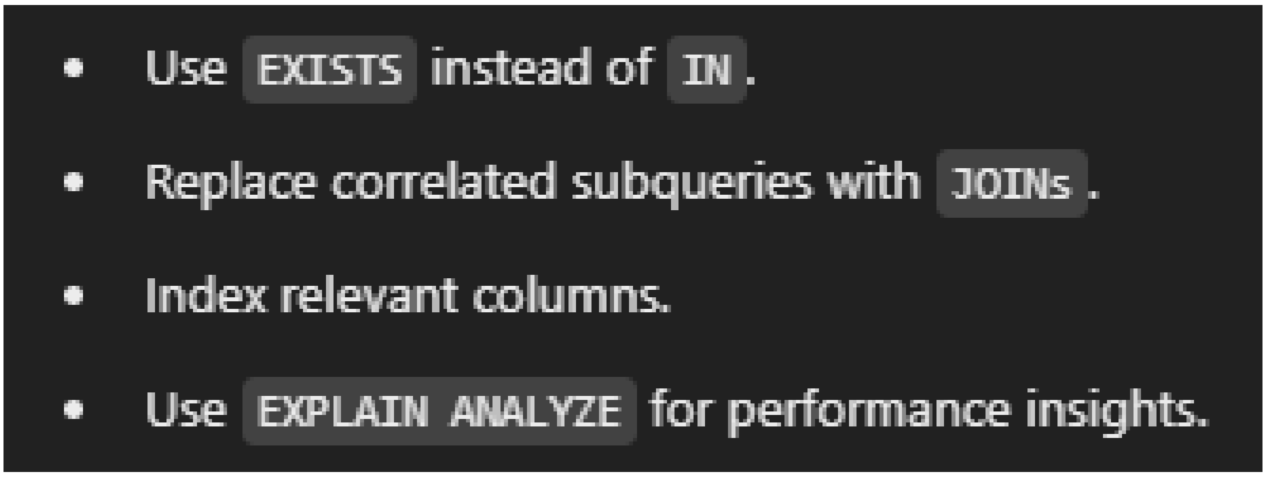
2) Differentiate the commands ALTER and UPDATE.



ALTER vs. UPDATE:

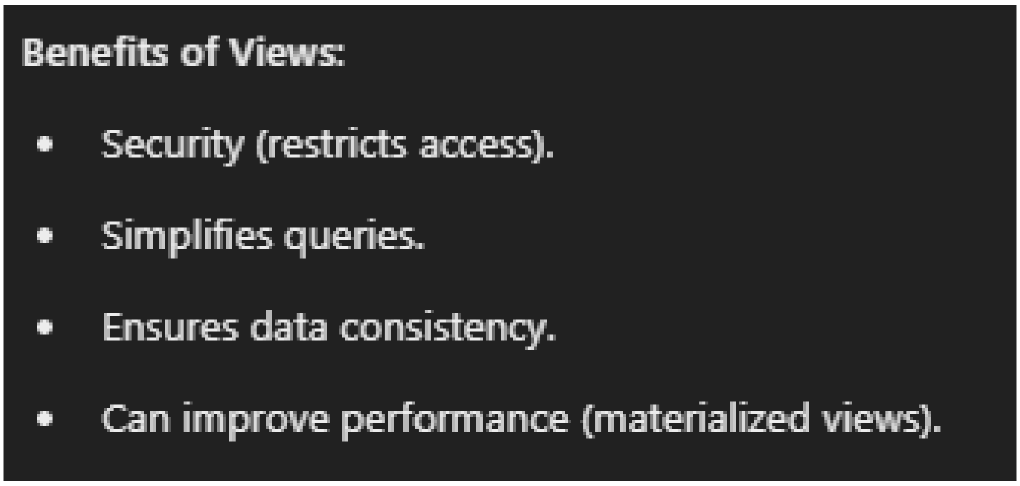
- `ALTER`: Modifies table structure (e.g., add/remove columns).
- `UPDATE`: Modifies existing data in a table.

3) How can you optimize performance when working with subqueries in PostgreSQL?



- Use `EXISTS` instead of `IN`.
- Replace correlated subqueries with `JOINs`.
- Index relevant columns.
- Use `EXPLAIN ANALYZE` for performance insights.

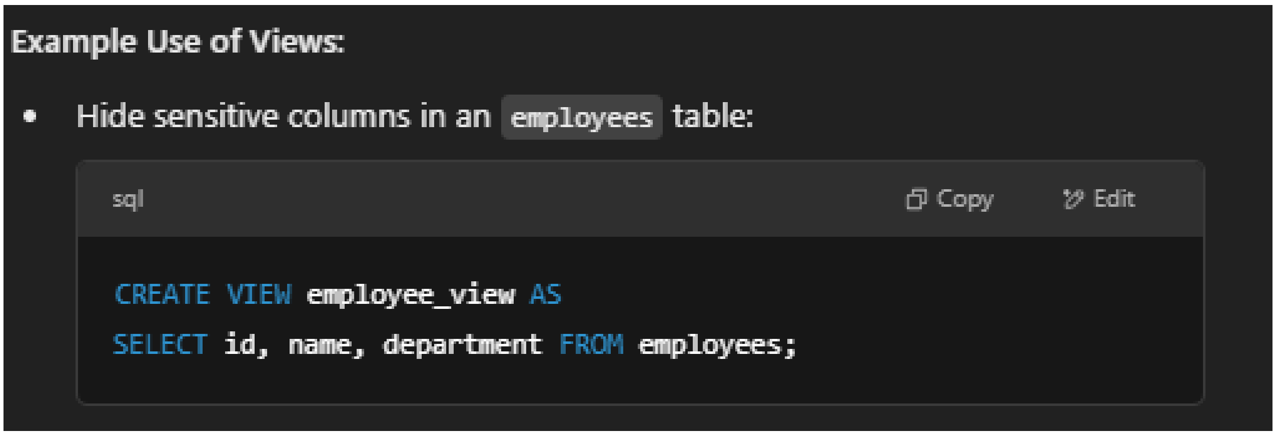
4) What are the benefits of using views in a database system?



Benefits of Views:

- Security (restricts access).
- Simplifies queries.
- Ensures data consistency.
- Can improve performance (materialized views).

5) Can you provide an example of a situation where creating a view would be beneficial?

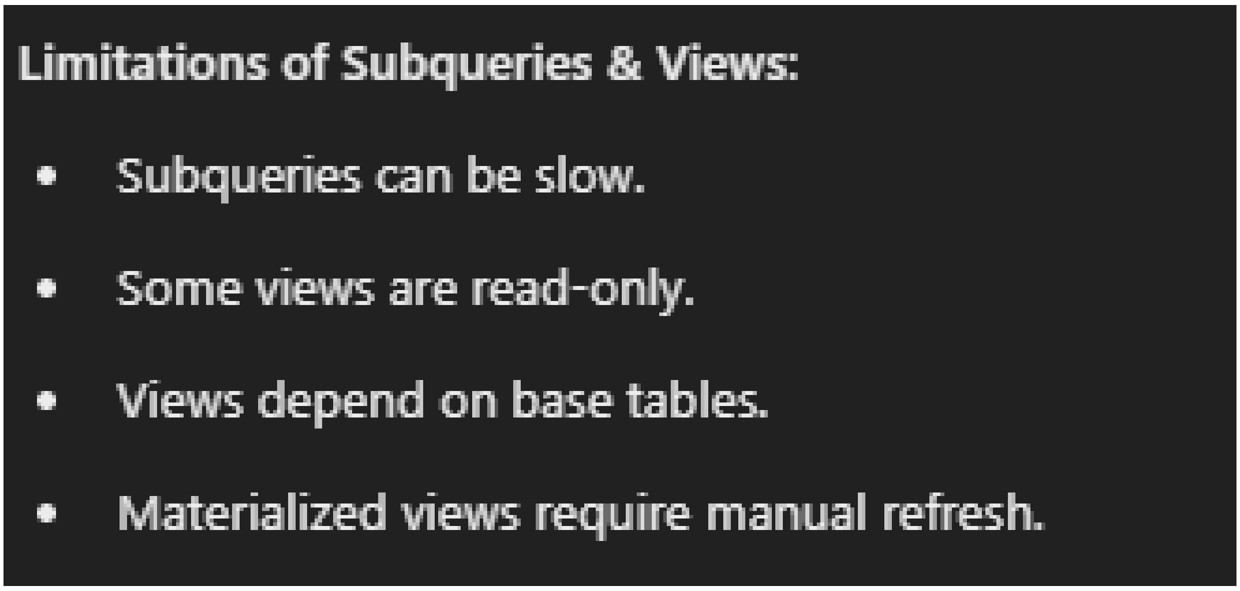


Example Use of Views:

- Hide sensitive columns in an `employees` table:

```sql
CREATE VIEW employee_view AS
SELECT id, name, department FROM employees;
```

6) Discuss the potential limitations or considerations when using subqueries or views in complex queries.



Limitations of Subqueries & Views:

- Subqueries can be slow.
- Some views are read-only.
- Views depend on base tables.
- Materialized views require manual refresh.

Post Lab Task:

1) Write PostgreSQL queries using nested subqueries to analyze the following data.

Questions:

i) Write a query to find the top 3 doctors who have earned the highest total amount from appointments.

```
SELECT doctor_id,
       (SELECT name FROM doctors d WHERE d.doctor_id = a.doctor_id) AS
doctor_name,
       SUM(amount) AS total_earnings
FROM appointments a
GROUP BY doctor_id
ORDER BY total_earnings DESC
LIMIT 3;
```

ii) Create a function that takes a doctor ID as input and returns the total number of appointments made by that doctor.

```
CREATE OR REPLACE FUNCTION
get_doctor_appointment_count(doctor_id_param INT)
RETURNS INT AS $$
DECLARE
   appointment_count INT;
BEGIN
   SELECT COUNT(*) INTO appointment_count
   FROM appointments
   WHERE doctor_id = doctor_id_param;

   RETURN appointment_count;
END;
$$ LANGUAGE plpgsql;

SELECT get_doctor_appointment_count(1);
```

iii) Write a query to find the patients who have spent the most on appointments, including their total amount spent and the doctor they visited the most.

```
WITH PatientSpending AS (
   SELECT patient_id, SUM(amount) AS total_spent
   FROM appointments
   GROUP BY patient_id
), MostVisitedDoctor AS (
   SELECT patient_id, doctor_id, COUNT(*) AS visit_count,
          RANK() OVER (PARTITION BY patient_id ORDER BY COUNT(*) DESC) AS
rank
   FROM appointments
   GROUP BY patient_id, doctor_id
)
SELECT p.patient_id,
     (SELECT name FROM patients WHERE patients.patient_id = p.patient_id) AS
patient_name,
     p.total_spent,
     d.doctor_id,
     (SELECT name FROM doctors WHERE doctors.doctor_id = d.doctor_id) AS
most_visited_doctor
FROM PatientSpending p
JOIN MostVisitedDoctor d ON p.patient_id = d.patient_id
WHERE d.rank = 1
ORDER BY p.total_spent DESC;
```

iv) Create a view that displays the average appointment amount for each doctor, categorized by the year of the appointment.

```
CREATE VIEW doctor_avg_appointment_by_year AS
SELECT doctor_id,
       EXTRACT(YEAR FROM appointment_date) AS year,
       AVG(amount) AS avg_appointment_amount
FROM appointments
GROUP BY doctor_id, year;
```

v) Write a query to find the doctor who has treated the highest number of unique patients.

```
SELECT doctor_id,
       (SELECT name FROM doctors WHERE doctors.doctor_id = a.doctor_id) AS
doctor_name,
       COUNT(DISTINCT patient_id) AS unique_patients
FROM appointments a
GROUP BY doctor_id
ORDER BY unique_patients DESC
LIMIT 1;
```

vi) Create a function that calculates the total earnings of a doctor, given their doctor ID and a date range.

```
CREATE OR REPLACE FUNCTION get_doctor_earnings(doctor_id_param INT,
start_date DATE, end_date DATE)
RETURNS NUMERIC AS $$
DECLARE
    total_earnings NUMERIC;
BEGIN
    SELECT COALESCE(SUM(amount), 0) INTO total_earnings
    FROM appointments
    WHERE doctor_id = doctor_id_param
      AND appointment_date BETWEEN start_date AND end_date;

    RETURN total_earnings;
END;
$$ LANGUAGE plpgsql;

SELECT get_doctor_earnings(1, '2024-01-01', '2024-12-31');
```

vii) Write a query to find the patients who have made appointments with doctors from multiple cities.

```
SELECT patient_id,
       (SELECT name FROM patients WHERE patients.patient_id = a.patient_id) AS
patient_name
FROM appointments a
JOIN doctors d ON a.doctor_id = d.doctor_id
GROUP BY patient_id
HAVING COUNT(DISTINCT d.city) > 1;
```

viii) Create a view that's hows the top 5 cities with the highest number of appointments.

```
CREATE VIEW top_5_cities_by_appointments AS
SELECT d.city, COUNT(a.appointment_id) AS appointment_count
FROM appointments a
JOIN doctors d ON a.doctor_id = d.doctor_id
GROUP BY d.city
ORDER BY appointment_count DESC
LIMIT 5;
```

ix) Write a query to find the doctor who has the highest average appointment amount.

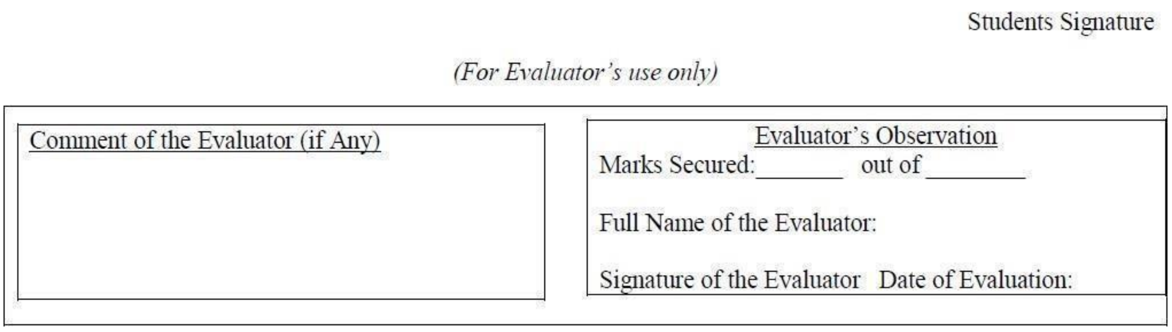
```
SELECT doctor_id,
       (SELECT name FROM doctors WHERE doctors.doctor_id = a.doctor_id) AS
doctor_name,
       AVG(amount) AS avg_appointment_amount
FROM appointments a
GROUP BY doctor_id
ORDER BY avg_appointment_amount DESC
LIMIT 1;
```

x) Create a function that takes a patient ID as input and returns the number of appointments made by that patient within a specific year.

```
CREATE OR REPLACE FUNCTION
get_patient_appointment_count(patient_id_param INT, year_param INT)
RETURNS INT AS $$
DECLARE
   appointment_count INT;
BEGIN
   SELECT COUNT(*) INTO appointment_count
   FROM appointments
   WHERE patient_id = patient_id_param
     AND EXTRACT(YEAR FROM appointment_date) = year_param;

   RETURN appointment_count;
END;
$$ LANGUAGE plpgsql;


SELECT get_patient_appointment_count(2, 2024);
```



Students Signature

(For Evaluator's use only)

Comment of the Evaluator (if Any)	Evaluator's Observation Marks Secured:_______ out of ________ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation: