Experiment / assignment / tutorial No. 4

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: : DML – select, insert, update and delete

1. Group by, having clause, aggregate functions, Set Operations
2. Nested queries : AND,OR,NOT, IN, NOT IN, Exists, Not Exists, Between, Like, Alias, ANY,ALL,DISTINCT
3. Update
4. Delete

Objective: To perform various DML Operations and executing nested queries with various clauses.

Expected Outcome of Experiment:
CO 3: Use SQL for Relational database creation, maintenance and query processing

Books/ Journals/ Websites referred:

1. Dr. P.S. Deshpande, SQL and PL/SQL for Oracle 10g.Black book, Dreamtech Press
2. www.db-book.com
3. Korth, Slberchatz, Sudarshan : “Database Systems Concept”, 5th Edition , McGraw Hill
4. Elmasri and Navathe,”Fundamentals of database Systems”, 4th Edition PEARSON Education.

Resources used: Postgres

Theory: Select: The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax

The basic syntax of the SELECT statement is as follows –

SELECT column1, column2, columnN FROM table_name;

Here, column1, column2... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

SELECT * FROM table_name;

The following code is an example, which would fetch the ID, Name and Salary fields of the customers available in CUSTOMERS table.

SQL> SELECT ID, NAME, SALARY FROM CUSTOMERS;

Insert: The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below.

INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)

VALUES (value1, value2, value3,...valueN);

Example

The following statements would create record in the CUSTOMERS table.

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

Update: The SQL **UPDATE** Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

Syntax:

The basic syntax of the UPDATE query with a WHERE clause is as follows −

UPDATE table_name

SET column1 = value1, column2 = value2. .., columnN = valueN

WHERE [condition];

You can combine N number of conditions using the AND or the OR operators.

The following query will update the ADDRESS for a customer whose ID number is 6 in the table.

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune'
WHERE ID = 6;
```

Delete: The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

Syntax

The basic syntax of the DELETE query with the WHERE clause is as follows −

DELETE FROM table_name

WHERE [condition];

The following code has a query, which will DELETE a customer, whose ID is 6.

```
SQL> DELETE FROM CUSTOMERS
WHERE ID = 6;
```

Clauses and Operators

1. **Group by clause:** These are circumstances where we would like to apply the aggregate functions to a single set of tuples but also to a group of sets of tuples we would like to specify this wish in

SQL using the group by clause. The attributes or attributes given by the group by clause are used to form groups. Tuples with the same value on all attributes in the group by clause placed in one group.
Example:.

Select<attribute_name,avg(<attribute_name>)as
<new_attribute_name>l From <table_name>
Group by <attribute_name>

Example: select designation, sum(salary) as total_salary from employee group by Designation;

2. Having clause: A having clause is like a where clause but only applies only to groups as a whole whereas the where clause applies to the individual rows. A query can contain both where clause and a having clause. In that case
a. The where clause is applied first to the individual rows in the tables or table structures objects in the diagram pane. Only the rows that meet the conditions in the where clause are grouped.
b. The having clause is then applied to the rows in the result set that are produced by grouping. Only the groups that meet the having conditions appear in the query output.
Example:

select dept_no from EMPLOYEE group_by dept_no
having avg (salary) >=all (select avg (salary)
from EMPLOYEE group by dept_no);

3. Aggregate functions: Aggregate functions such as SUM, AVG, count, count (*), MAX and MIN generate summary values in query result sets. An aggregate functions (with the exception of count (*) processes all the selected values in a single column to produce a single result value

Example: select dept_no,count (*)
from EMPLOYEE group by dept_no;

Example: select max (salary)as maximum from EMPLOYEE;

Example: select sum (salary) as total_salary from EMPLOYEE;

Example: Select min (salary) as minsal from EMPLOYEE;

4. Exists and Not Exists: Subqueries introduced with exists and not queries can be used for two set theory operations: Intersection and Difference. The intersection of two sets contains all elements that belong to both of the original sets. The difference contains elements that belong to only first of the two sets.

Example:

```
Select *from DEPARTMENT
where exists(select * from PROJECT
            where DEPARTMENT.dept_no = PROJECT.dept_no) ;
```

5. IN and Not In: SQL allows testing tuples for membership in a relation. The "in‟ connective tests for set membership where the set is a collection of values produced by select clause. The "not in‟ connective tests for the absence of set membership. The in and not in connectives can also be used on enumerated sets.

Example:

```
1. Select fname, mname, lname from employee where designation In
("ceo‟,‟manager‟,‟hod‟,‟assistant‟)

2. Select fullname from department where relationship not in("brother");
```

6. Between: The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive. Begin and end values are included.
Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Example:
SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;

7. LIKE: The LIKE **operator** is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

Syntax: SELECT *column1*, *column2, ...*
FROM *table_name*
WHERE *columnN* LIKE *pattern*

Examples:

1. selects all customers with a CustomerName starting with "a":

SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';

2. selects all customers with a CustomerName that have "r" in the second position:

SELECT * FROM Customers
WHERE CustomerName LIKE '_r%';

8. Alias: The use of table aliases is to rename a table in a specific SQL statement. The renaming is a temporary change and the actual table name does not change in the database. The column aliases are used to rename a table's columns for the purpose of a particular SQL query.

The basic syntax of a **table** alias is as follows.

SELECT column1, column2....

FROM table_name AS alias_name

WHERE [condition];

The basic syntax of a **column** alias is as follows.

SELECT column_name AS alias_name

FROM table_name

WHERE [condition];

Example:

```
SELECT C.ID, C.NAME, C.AGE, O.AMOUNT

FROM CUSTOMERS AS C, ORDERS AS O

WHERE  C.ID = O.CUSTOMER_ID;
```

9. Distinct: The SELECT DISTINCT statement is used to return only distinct (different) values.

Syntax: SELECT DISTINCT *column1*, *column2, ...*
FROM *table_name*;

Example: SELECT DISTINCT Country FROM Customers;

10. Set Operations: 4 different types of SET operations, along with example:

1. UNION
2. UNION ALL
3. INTERSECT
4. MINUS

UNION Operation

UNION is used to combine the results of two or more SELECT statements. However it will eliminate duplicate rows from its resultset. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.

Query: SELECT * FROM First

UNION

SELECT * FROM Second;

UNION ALL

This operation is similar to Union. But it also shows the duplicate rows.

Query: SELECT * FROM First

UNION ALL

SELECT * FROM Second;

INTERSECT

Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and datatype must be same.

Query: SELECT * FROM First

INTERSECT

SELECT * FROM Second;

MINUS

The Minus operation combines results of two SELECT statements and return only those in the final result, which belongs to the first set of the result.

Query: SELECT * FROM First

MINUS

SELECT * FROM Second;

11. ANY and ALL: The ANY and ALL operators are used with a WHERE or HAVING clause. The ANY operator returns true if any of the subquery values meet the condition. The ALL operator returns true if all of the subquery values meet the condition.

ANY

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* ANY
(SELECT *column_name* FROM *table_name* WHERE *condition*);

Example: The following SQL statement returns TRUE and lists the productnames if it finds ANY records in the OrderDetails table that quantity = 10:

SELECT ProductName
FROM Products
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);

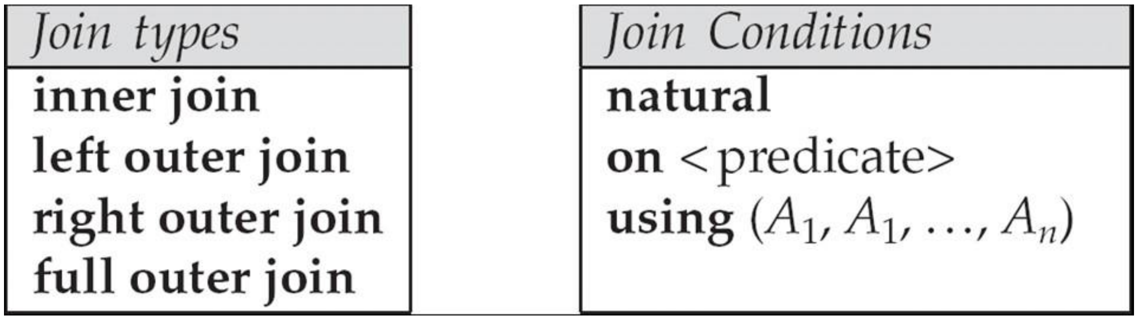
ALL

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* ALL
(SELECT *column_name* FROM *table_name* WHERE *condition*);

Example: The following SQL statement returns TRUE and lists the product names if ALL the records in the OrderDetails table has quantity = 10:

SELECT ProductName
FROM Products
WHERE ProductID = ALL (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);

JOIN OPERATIONS:



Join types
inner join
left outer join
right outer join
full outer join

Join Conditions
natural
on < predicate>
using $(A_1, A_1, \ldots, A_n)$

Join operations take two relations and return as a result another relation.

These additional operations are typically used as subquery expressions in the **from** clause

Join condition – defines which tuples in the two relations match, and what attributes are present in the result of the join.

Join type – defines how tuples in each relation that do not match any tuple in the other relation (based on the join condition) are treated

loan **join** *borrower* **on**

loan.loan_number = *borrower.loan_number*

CREATE [TEMP | TEMPORARY] VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE [condition];

Ex

CREATE VIEW COMPANY_VIEW AS

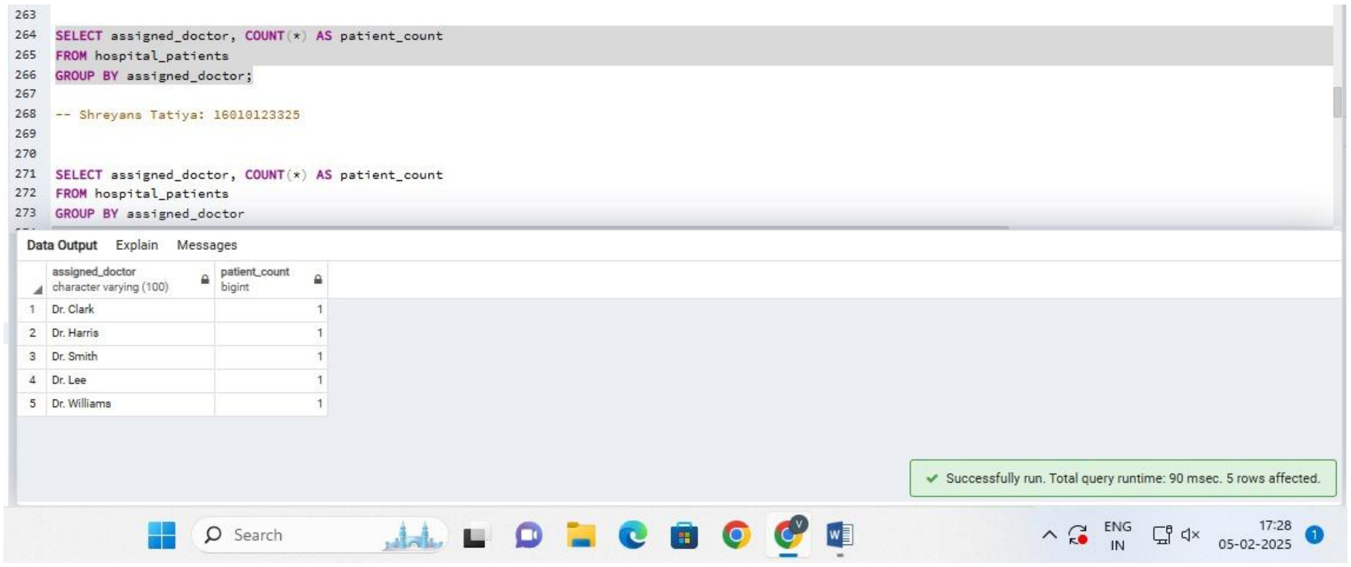
```
SELECT ID, NAME, AGE

FROM  COMPANY;
```

Dropping Views

Syntax: DROP VIEW view_name;

Implementation details

Simple question based on your application, queries and screen shots for each type:

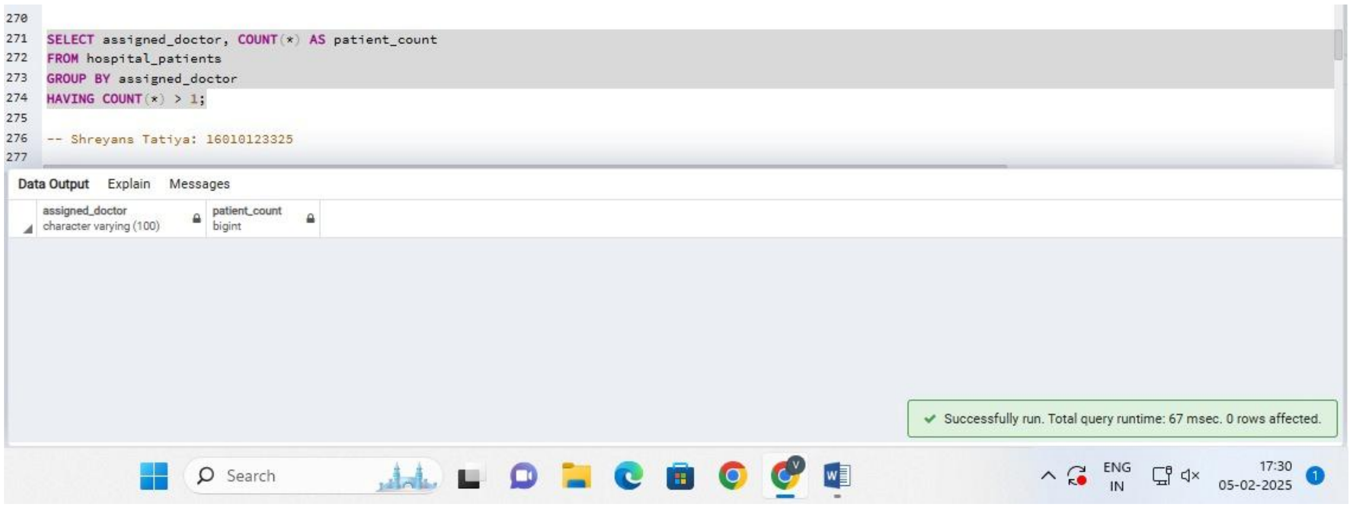


```
SELECT assigned_doctor, COUNT(*) AS patient_count
FROM hospital_patients
GROUP BY assigned_doctor;

-- Shreyans Tatiya: 16010123325


SELECT assigned_doctor, COUNT(*) AS patient_count
FROM hospital_patients
GROUP BY assigned_doctor
```

Data Output Explain Messages

	assigned_doctor character varying (100)	patient_count bigint
1	Dr. Clark	1
2	Dr. Harris	1
3	Dr. Smith	1
4	Dr. Lee	1
5	Dr. Williams	1

Successfully run. Total query runtime: 90 msec. 5 rows affected.



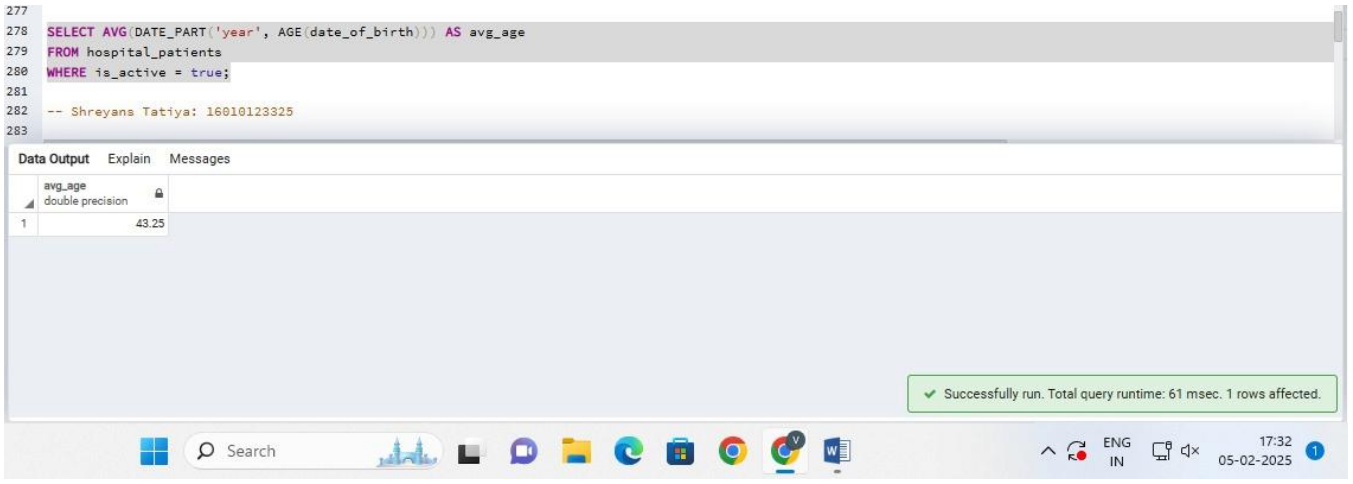
```
SELECT assigned_doctor, COUNT(*) AS patient_count
FROM hospital_patients
GROUP BY assigned_doctor
HAVING COUNT(*) > 1;

-- Shreyans Tatiya: 16010123325
```

Data Output Explain Messages

assigned_doctor character varying (100)	patient_count bigint

Successfully run. Total query runtime: 67 msec. 0 rows affected.

AVG

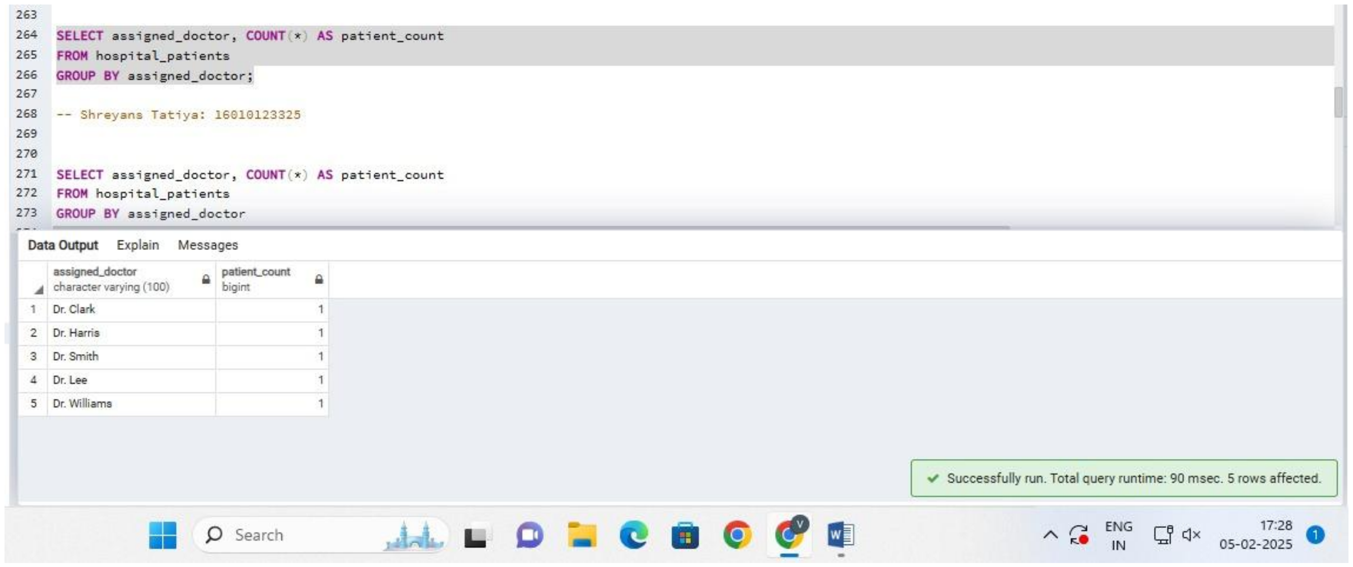


```
SELECT AVG(DATE_PART('year', AGE(date_of_birth))) AS avg_age
FROM hospital_patients
WHERE is_active = true;

-- Shreyans Tatiya: 16010123325
```

	avg_age double precision
1	43.25

Successfully run. Total query runtime: 61 msec. 1 rows affected.

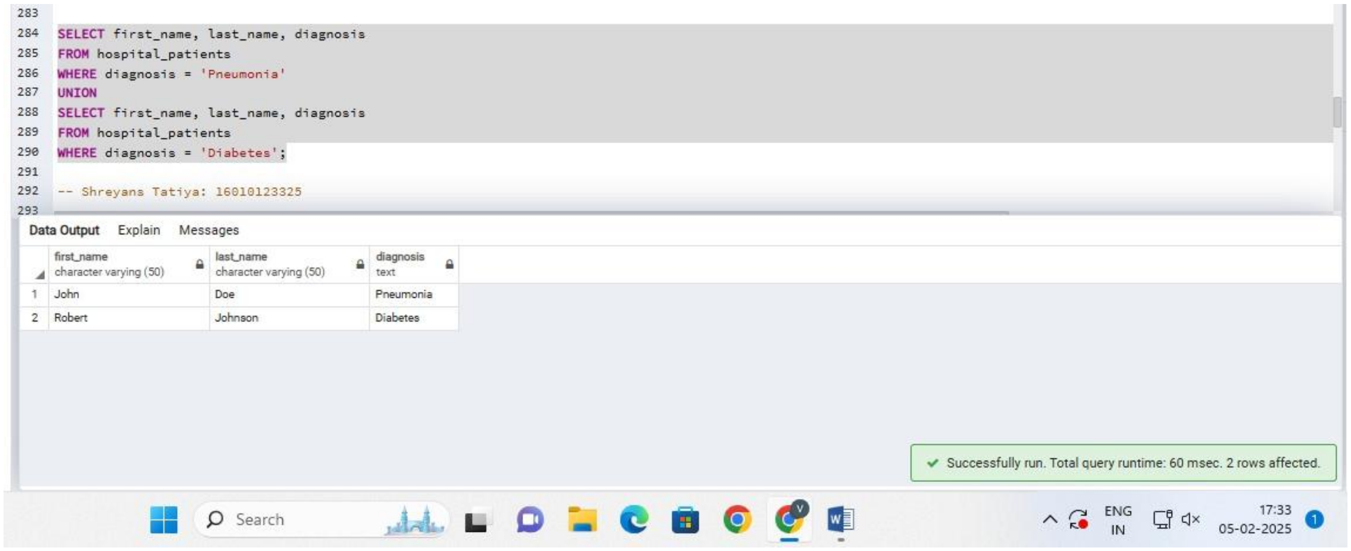
COUNT



```
SELECT assigned_doctor, COUNT(*) AS patient_count
FROM hospital_patients
GROUP BY assigned_doctor;

-- Shreyans Tatiya: 16010123325


SELECT assigned_doctor, COUNT(*) AS patient_count
FROM hospital_patients
GROUP BY assigned_doctor
```

	assigned_doctor character varying (100)	patient_count bigint
1	Dr. Clark	1
2	Dr. Harris	1
3	Dr. Smith	1
4	Dr. Lee	1
5	Dr. Williams	1

Successfully run. Total query runtime: 90 msec. 5 rows affected.

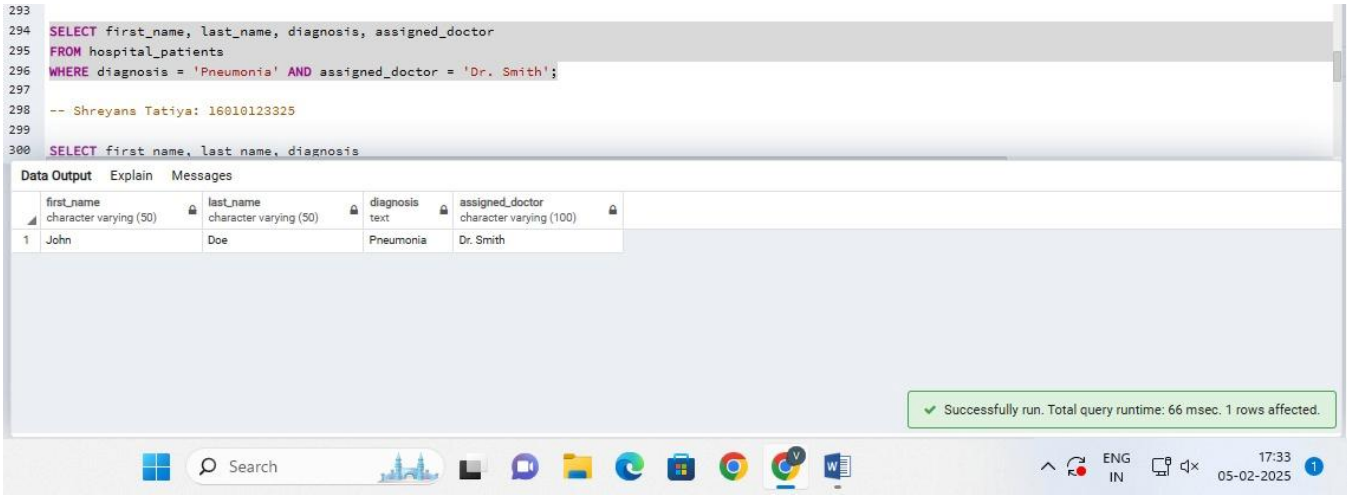


```
SELECT first_name, last_name, diagnosis
FROM hospital_patients
WHERE diagnosis = 'Pneumonia'
UNION
SELECT first_name, last_name, diagnosis
FROM hospital_patients
WHERE diagnosis = 'Diabetes';

-- Shreyans Tatiya: 16010123325
```

Data Output Explain Messages

	first_name character varying (50)	last_name character varying (50)	diagnosis text
1	John	Doe	Pneumonia
2	Robert	Johnson	Diabetes

Successfully run. Total query runtime: 60 msec. 2 rows affected.

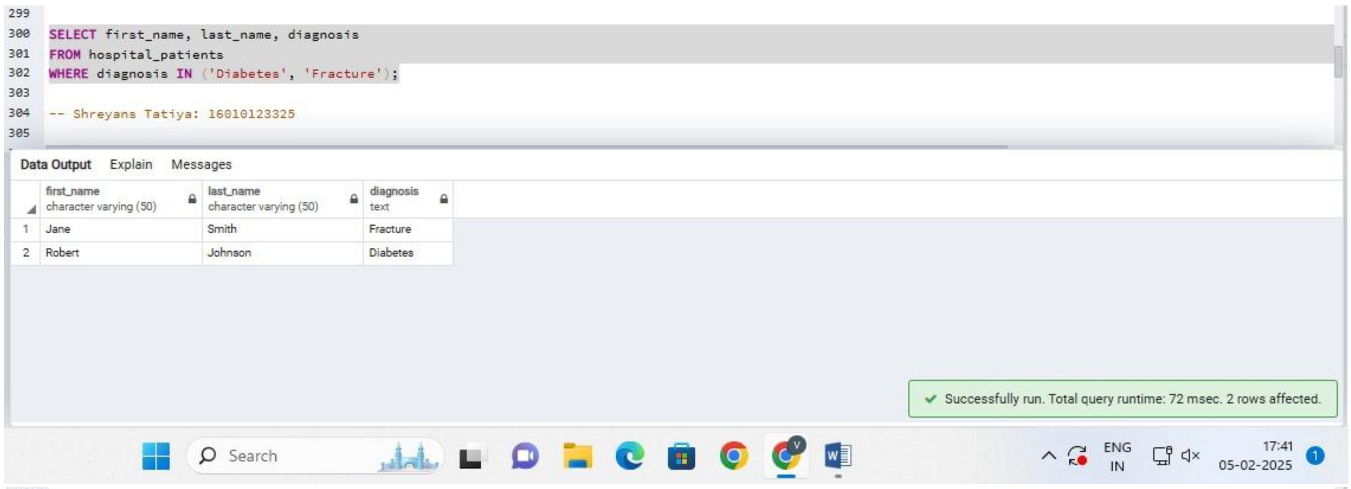


```
SELECT first_name, last_name, diagnosis, assigned_doctor
FROM hospital_patients
WHERE diagnosis = 'Pneumonia' AND assigned_doctor = 'Dr. Smith';

-- Shreyans Tatiya: 16010123325

SELECT first_name, last_name, diagnosis
```

Data Output Explain Messages

	first_name character varying (50)	last_name character varying (50)	diagnosis text	assigned_doctor character varying (100)
1	John	Doe	Pneumonia	Dr. Smith

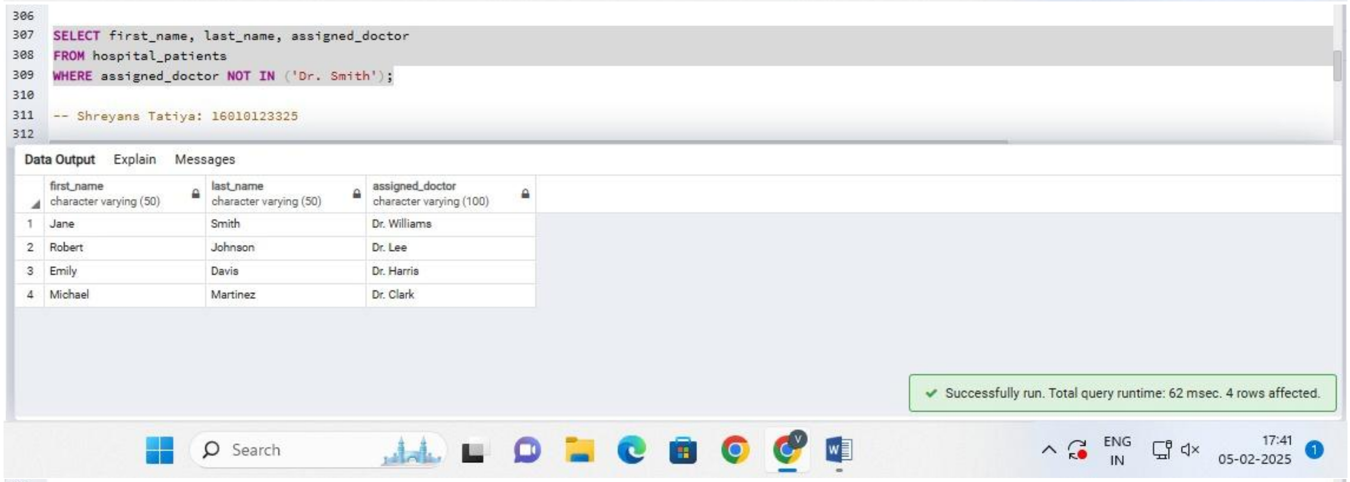
Successfully run. Total query runtime: 66 msec. 1 rows affected.



```
SELECT first_name, last_name, diagnosis
FROM hospital_patients
WHERE diagnosis IN ('Diabetes', 'Fracture');

-- Shreyans Tatiya: 16010123325
```

	first_name character varying (50)	last_name character varying (50)	diagnosis text
1	Jane	Smith	Fracture
2	Robert	Johnson	Diabetes

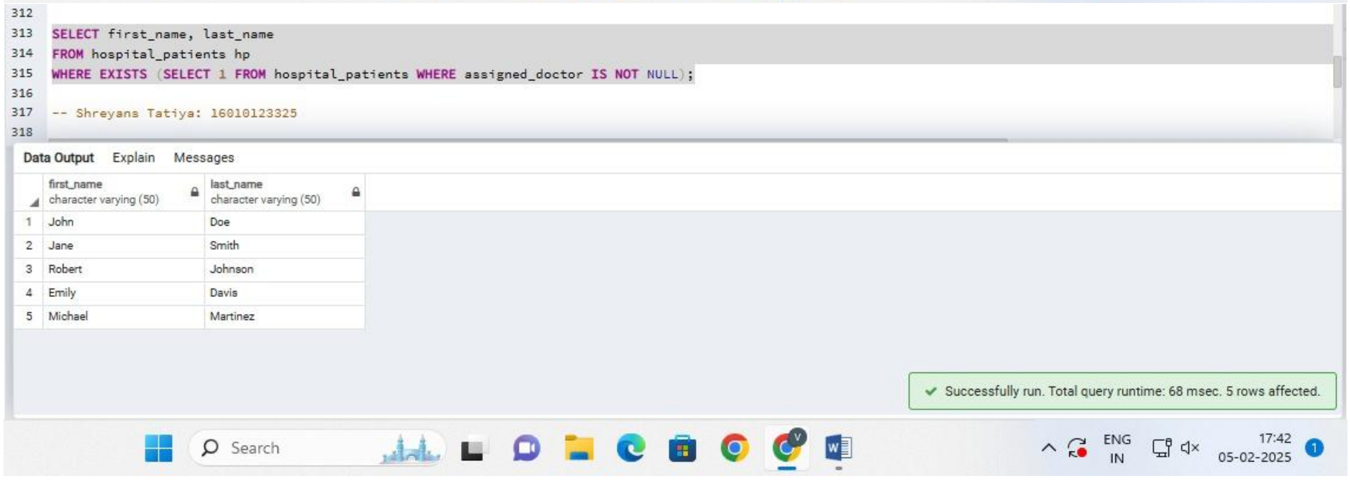
Successfully run. Total query runtime: 72 msec. 2 rows affected.



```
SELECT first_name, last_name, assigned_doctor
FROM hospital_patients
WHERE assigned_doctor NOT IN ('Dr. Smith');

-- Shreyans Tatiya: 16010123325
```

	first_name character varying (50)	last_name character varying (50)	assigned_doctor character varying (100)
1	Jane	Smith	Dr. Williams
2	Robert	Johnson	Dr. Lee
3	Emily	Davis	Dr. Harris
4	Michael	Martinez	Dr. Clark

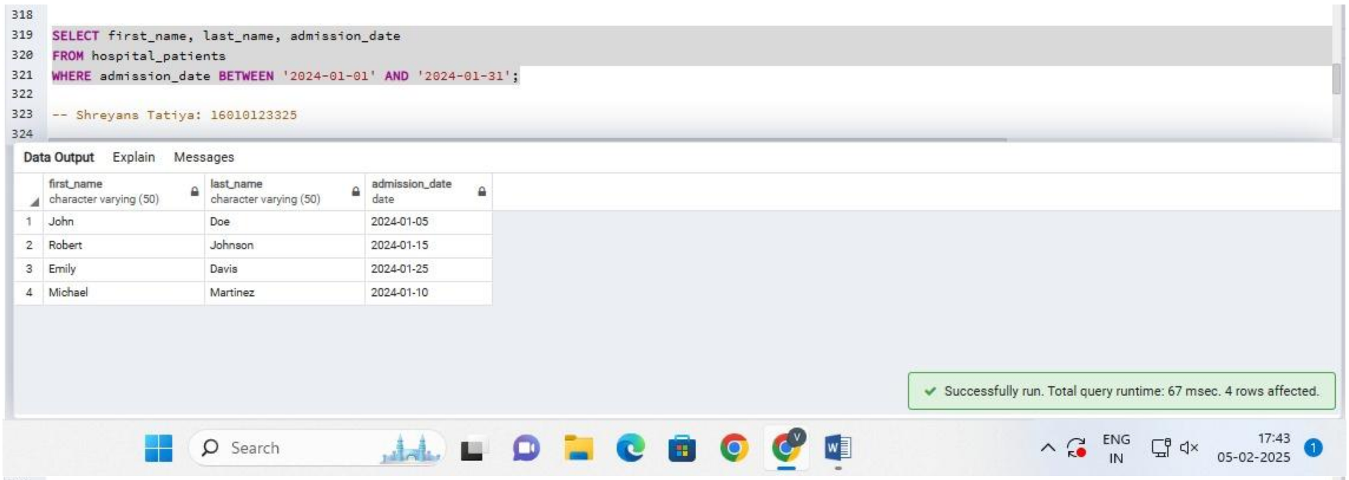
Successfully run. Total query runtime: 62 msec. 4 rows affected.



```
SELECT first_name, last_name
FROM hospital_patients hp
WHERE EXISTS (SELECT 1 FROM hospital_patients WHERE assigned_doctor IS NOT NULL);

-- Shreyans Tatiya: 16010123325
```

	first_name character varying (50)	last_name character varying (50)
1	John	Doe
2	Jane	Smith
3	Robert	Johnson
4	Emily	Davis
5	Michael	Martinez

Successfully run. Total query runtime: 68 msec. 5 rows affected.



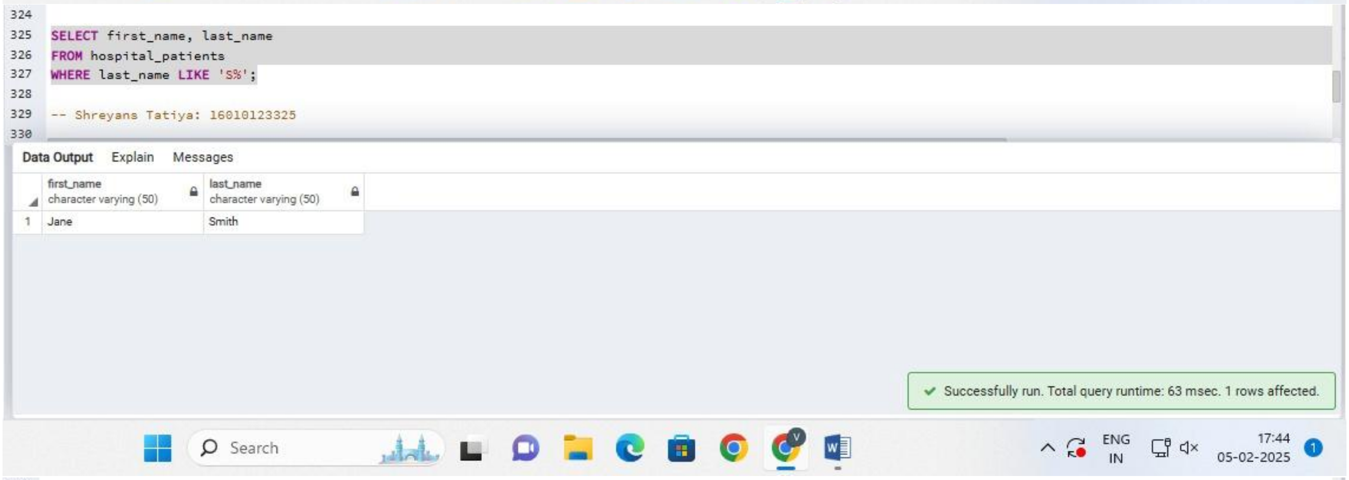
```
SELECT first_name, last_name, admission_date
FROM hospital_patients
WHERE admission_date BETWEEN '2024-01-01' AND '2024-01-31';

-- Shreyans Tatiya: 16010123325
```

Data Output   Explain   Messages

	first_name character varying (50)	last_name character varying (50)	admission_date date
1	John	Doe	2024-01-05
2	Robert	Johnson	2024-01-15
3	Emily	Davis	2024-01-25
4	Michael	Martinez	2024-01-10

Successfully run. Total query runtime: 67 msec. 4 rows affected.

17:43
05-02-2025



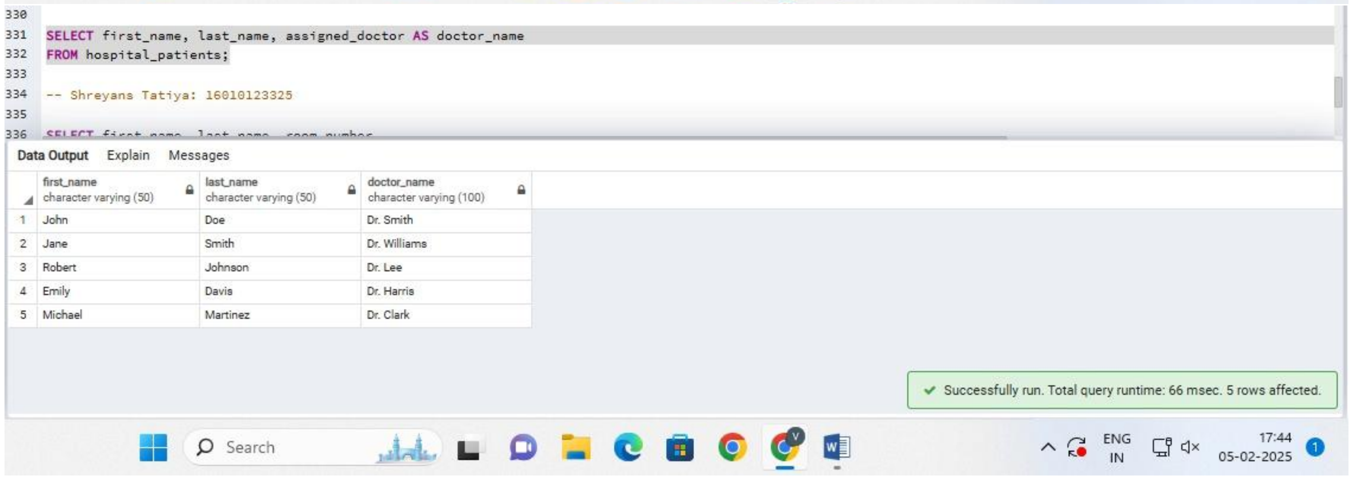
```
SELECT first_name, last_name
FROM hospital_patients
WHERE last_name LIKE 'S%';

-- Shreyans Tatiya: 16010123325
```

Data Output   Explain   Messages

	first_name character varying (50)	last_name character varying (50)
1	Jane	Smith

Successfully run. Total query runtime: 63 msec. 1 rows affected.

17:44
05-02-2025



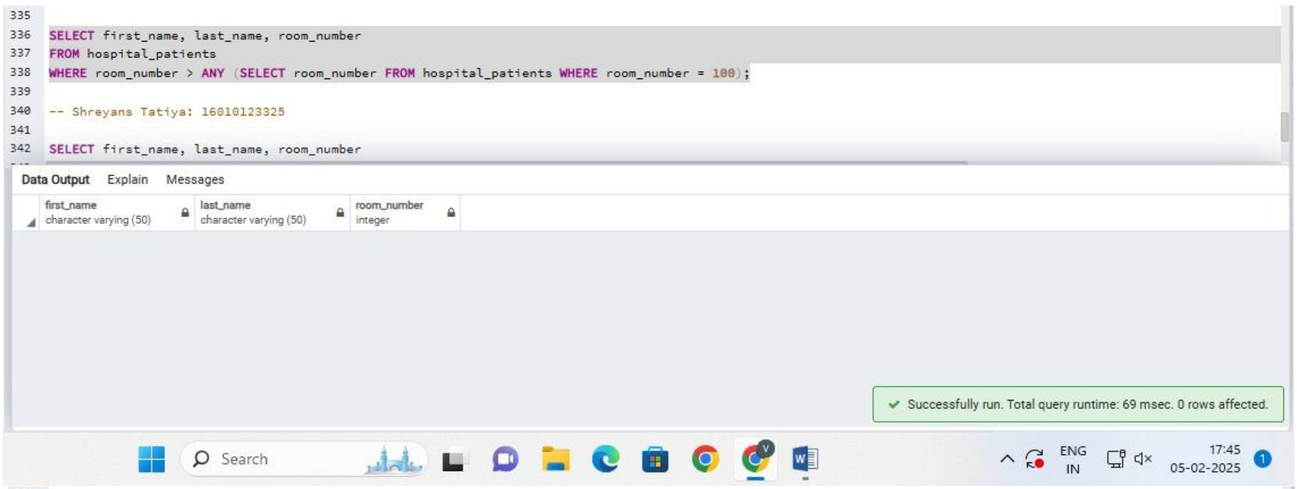
```
SELECT first_name, last_name, assigned_doctor AS doctor_name
FROM hospital_patients;

-- Shreyans Tatiya: 16010123325

SELECT first_name, last_name, room_number
```

Data Output   Explain   Messages

	first_name character varying (50)	last_name character varying (50)	doctor_name character varying (100)
1	John	Doe	Dr. Smith
2	Jane	Smith	Dr. Williams
3	Robert	Johnson	Dr. Lee
4	Emily	Davis	Dr. Harris
5	Michael	Martinez	Dr. Clark

Successfully run. Total query runtime: 66 msec. 5 rows affected.

17:44
05-02-2025

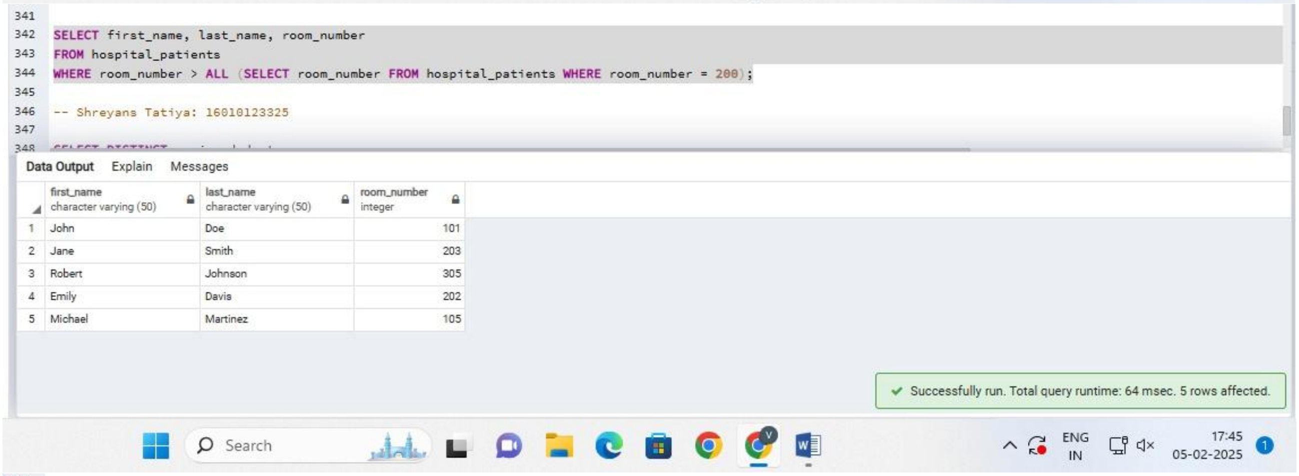


```
SELECT first_name, last_name, room_number
FROM hospital_patients
WHERE room_number > ANY (SELECT room_number FROM hospital_patients WHERE room_number = 100);

-- Shreyans Tatiya: 16010123325

SELECT first_name, last_name, room_number
```

Data Output | Explain | Messages

first_name character varying (50)	last_name character varying (50)	room_number integer

Successfully run. Total query runtime: 69 msec. 0 rows affected.

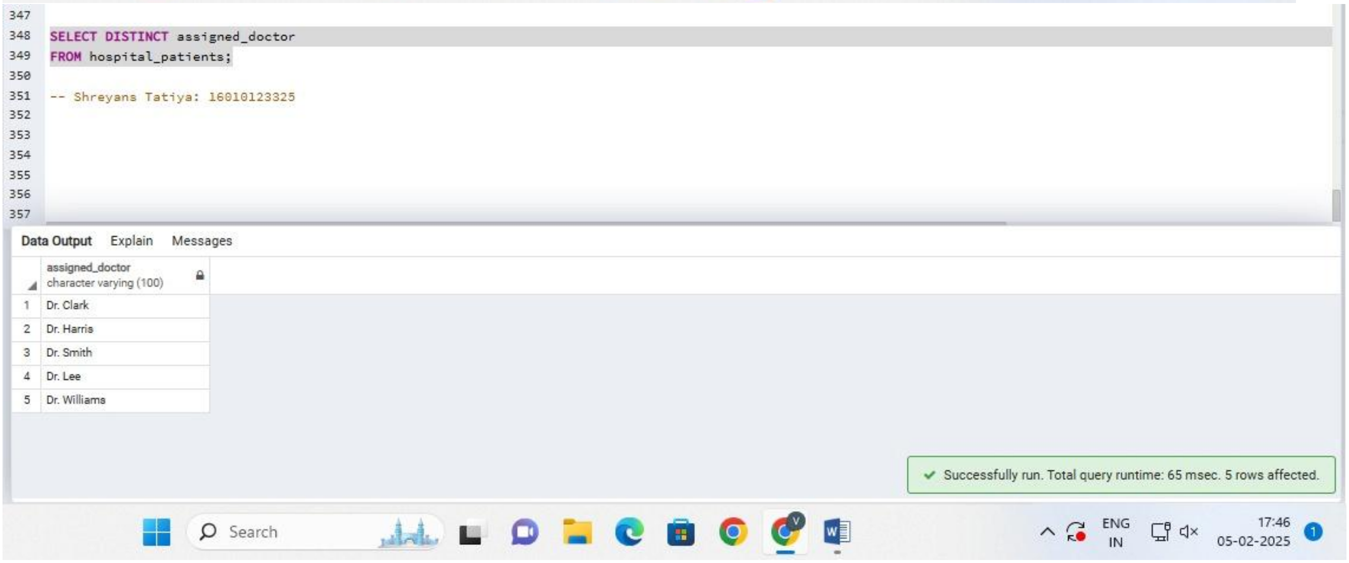


```
SELECT first_name, last_name, room_number
FROM hospital_patients
WHERE room_number > ALL (SELECT room_number FROM hospital_patients WHERE room_number = 200);

-- Shreyans Tatiya: 16010123325
```

Data Output | Explain | Messages

	first_name character varying (50)	last_name character varying (50)	room_number integer
1	John	Doe	101
2	Jane	Smith	203
3	Robert	Johnson	305
4	Emily	Davis	202
5	Michael	Martinez	105

Successfully run. Total query runtime: 64 msec. 5 rows affected.



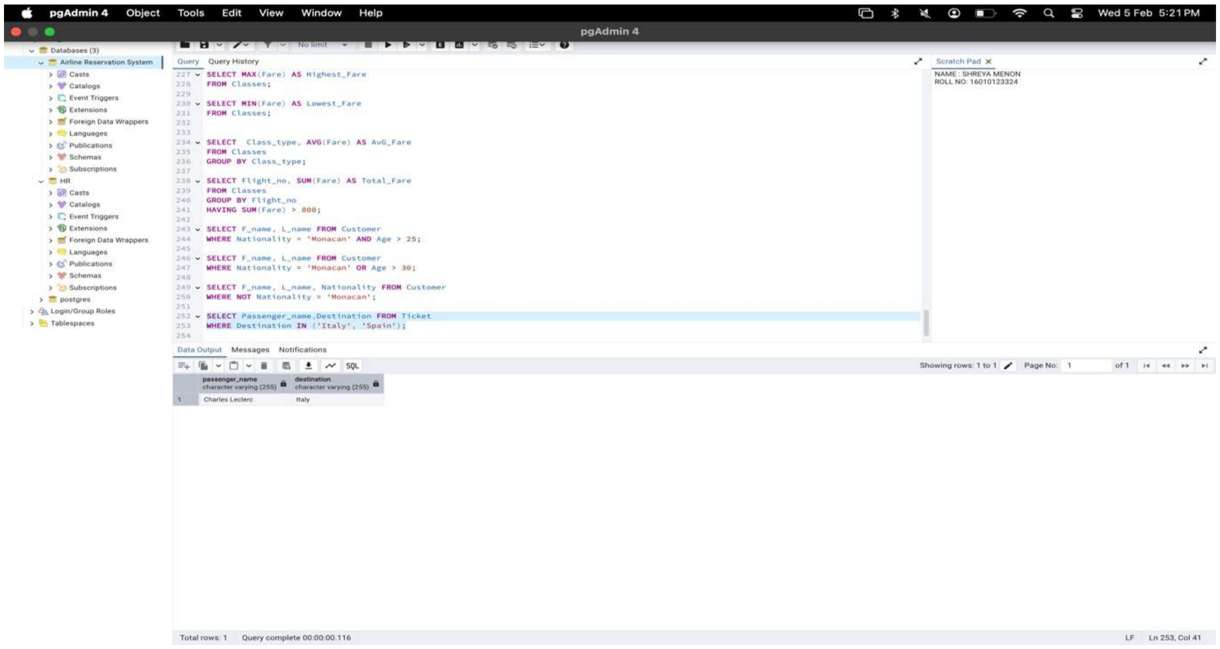
```
SELECT DISTINCT assigned_doctor
FROM hospital_patients;

-- Shreyans Tatiya: 16010123325
```

Data Output | Explain | Messages

	assigned_doctor character varying (100)
1	Dr. Clark
2	Dr. Harris
3	Dr. Smith
4	Dr. Lee
5	Dr. Williams

Successfully run. Total query runtime: 65 msec. 5 rows affected.

IN / NOT IN



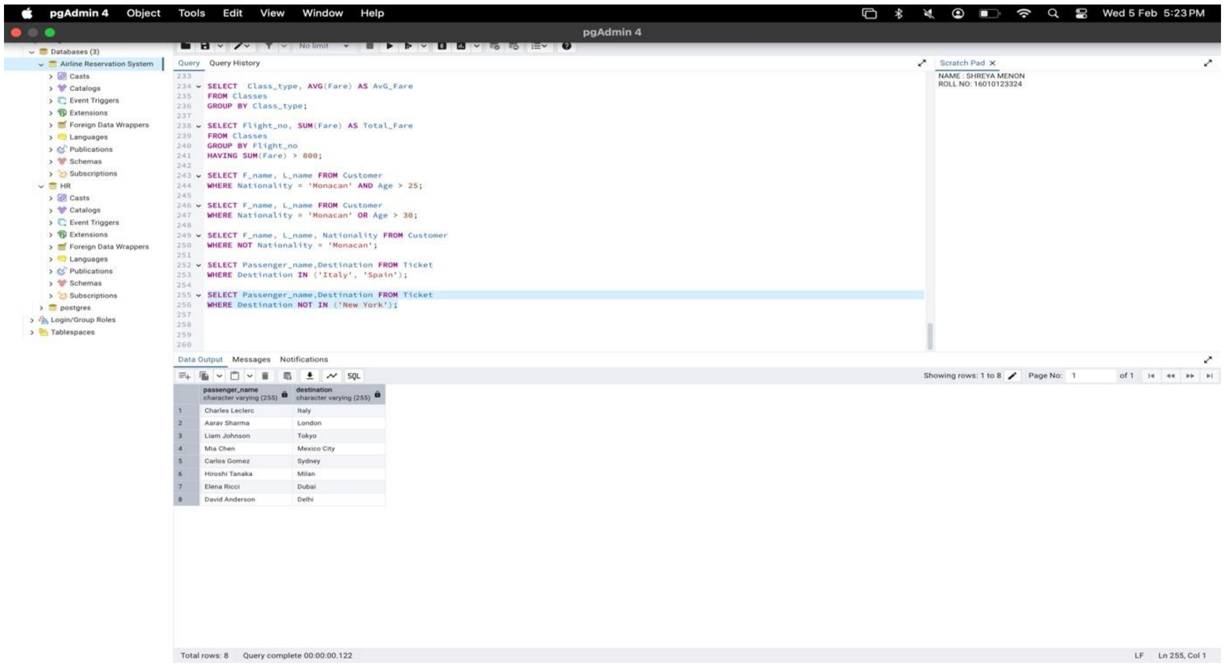
```
SELECT MAX(Fare) AS Highest_Fare
FROM Classes;

SELECT MIN(Fare) AS Lowest_Fare
FROM Classes;


SELECT  Class_type, AVG(Fare) AS AvG_Fare
FROM Classes
GROUP BY Class_type;

SELECT Flight_no, SUM(Fare) AS Total_Fare
FROM Classes
GROUP BY Flight_no
HAVING SUM(Fare) > 800;

SELECT F_name, L_name FROM Customer
WHERE Nationality = 'Monacan' AND Age > 25;

SELECT F_name, L_name FROM Customer
WHERE Nationality = 'Monacan' OR Age > 30;

SELECT F_name, L_name, Nationality FROM Customer
WHERE NOT Nationality = 'Monacan';

SELECT Passenger_name,Destination FROM Ticket
WHERE Destination IN ('Italy', 'Spain');
```

NAME : SHREYA MENON
ROLL NO: 16010123324

	passenger_name character varying (255)	destination character varying (255)
1	Charles Leclerc	Italy

Total rows: 1 Query complete 00:00:00.116

Department of Computer Engineering



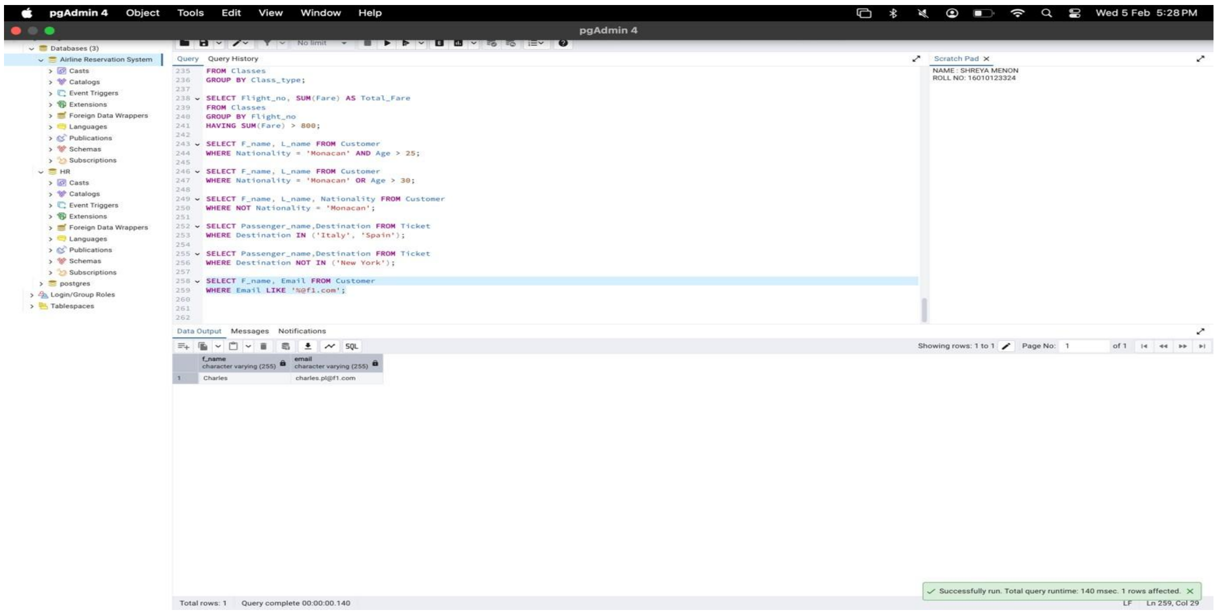
Scratch Pad: NAME : SHREYA MENON ROLL NO: 16010123324

```sql
SELECT  Class_type, AVG(Fare) AS AvG_Fare
FROM Classes
GROUP BY Class_type;

SELECT Flight_no, SUM(Fare) AS Total_Fare
FROM Classes
GROUP BY Flight_no
HAVING SUM(Fare) > 800;

SELECT F_name, L_name FROM Customer
WHERE Nationality = 'Monacan' AND Age > 25;

SELECT F_name, L_name FROM Customer
WHERE Nationality = 'Monacan' OR Age > 30;

SELECT F_name, L_name, Nationality FROM Customer
WHERE NOT Nationality = 'Monacan';

SELECT Passenger_name,Destination FROM Ticket
WHERE Destination IN ('Italy', 'Spain');

SELECT Passenger_name,Destination FROM Ticket
WHERE Destination NOT IN ('New York');
```

	passenger_name character varying (255)	destination character varying (255)
1	Charles Leclerc	Italy
2	Aarav Sharma	London
3	Liam Johnson	Tokyo
4	Mia Chen	Mexico City
5	Carlos Gomez	Sydney
6	Hiroshi Tanaka	Milan
7	Elena Ricci	Dubai
8	David Anderson	Delhi

Total rows: 8 Query complete 00:00:00.122

LIKE



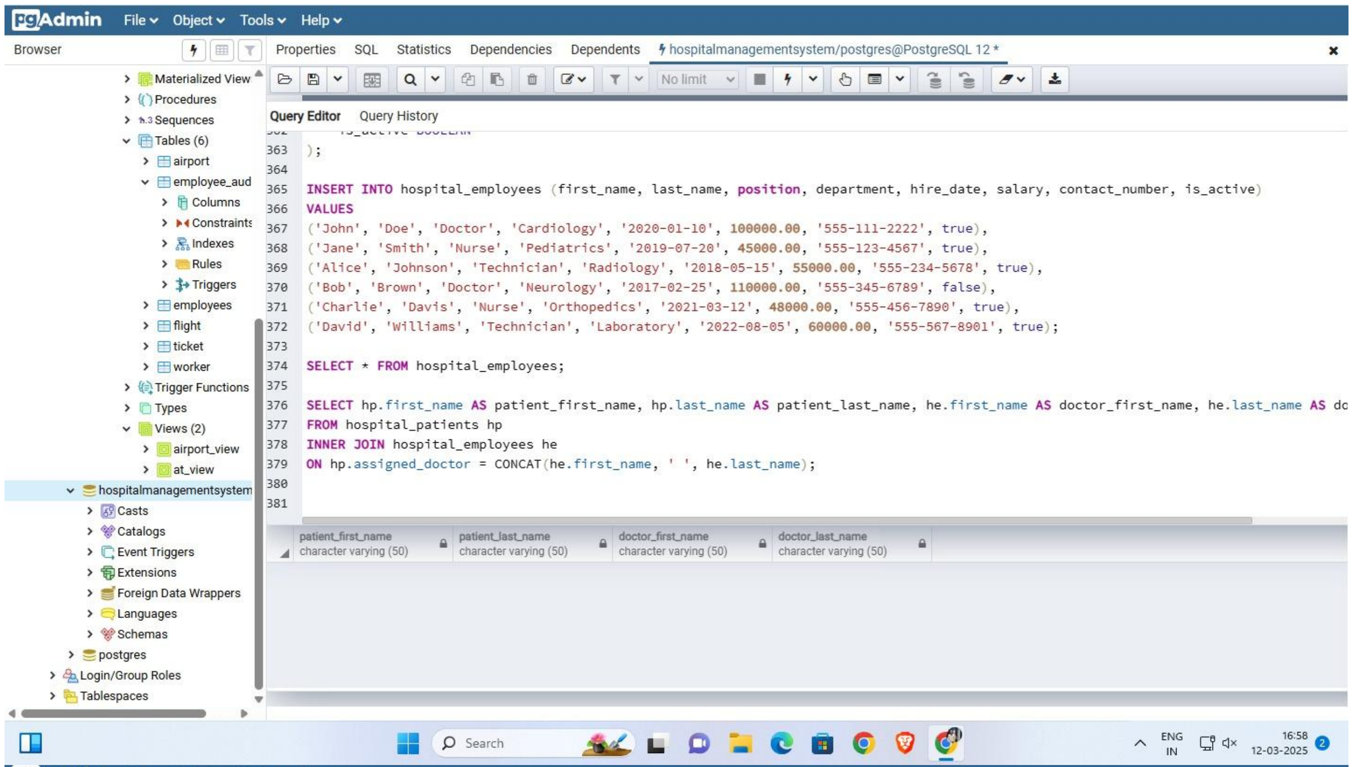
Scratch Pad: NAME : SHREYA MENON ROLL NO: 16010123324

```sql
FROM Classes
GROUP BY Class_type;

SELECT Flight_no, SUM(Fare) AS Total_Fare
FROM Classes
GROUP BY Flight_no
HAVING SUM(Fare) > 800;

SELECT F_name, L_name FROM Customer
WHERE Nationality = 'Monacan' AND Age > 25;

SELECT F_name, L_name FROM Customer
WHERE Nationality = 'Monacan' OR Age > 30;

SELECT F_name, L_name, Nationality FROM Customer
WHERE NOT Nationality = 'Monacan';

SELECT Passenger_name,Destination FROM Ticket
WHERE Destination IN ('Italy', 'Spain');

SELECT Passenger_name,Destination FROM Ticket
WHERE Destination NOT IN ('New York');

SELECT F_name, Email FROM Customer
WHERE Email LIKE '%@f1.com';
```

	f_name character varying (255)	email character varying (255)
1	Charles	charles.pl@f1.com

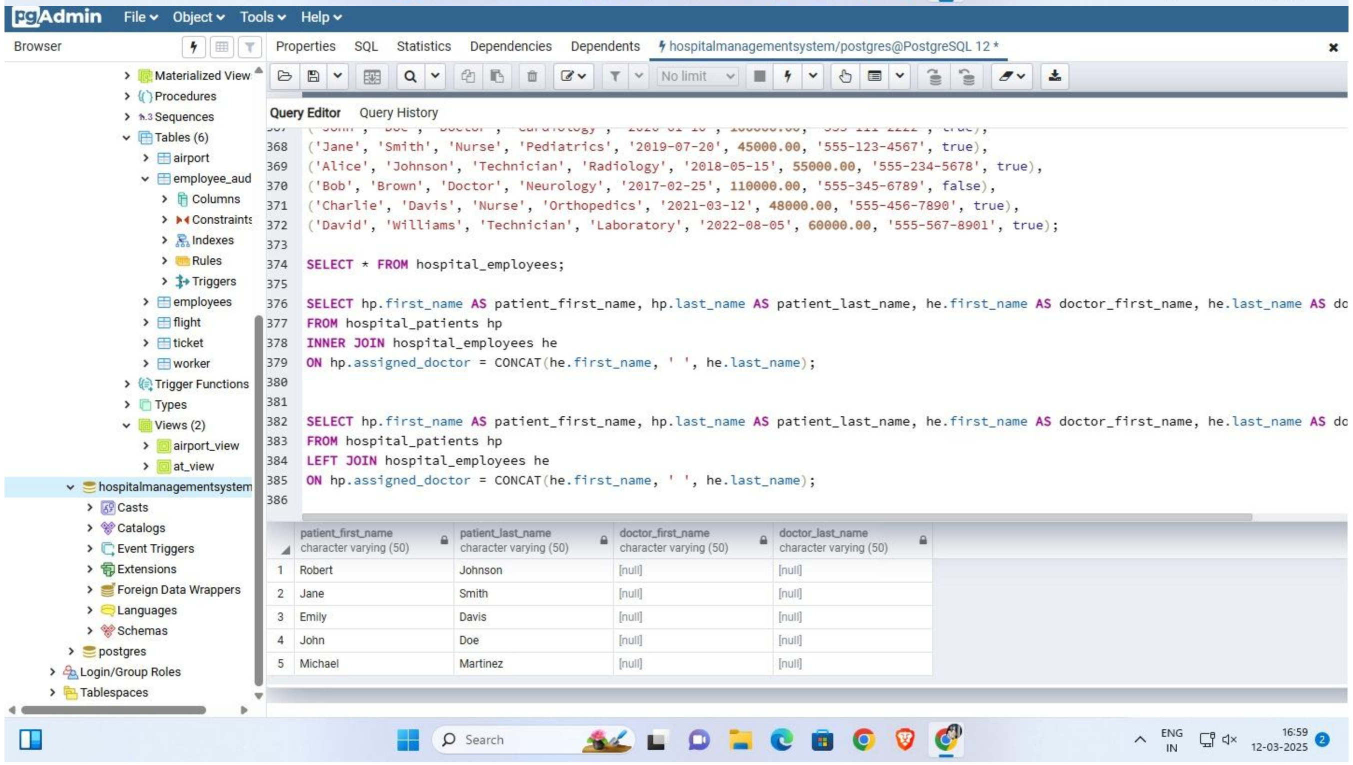
Successfully run. Total query runtime: 140 msec. 1 rows affected.

Total rows: 1 Query complete 00:00:00.140



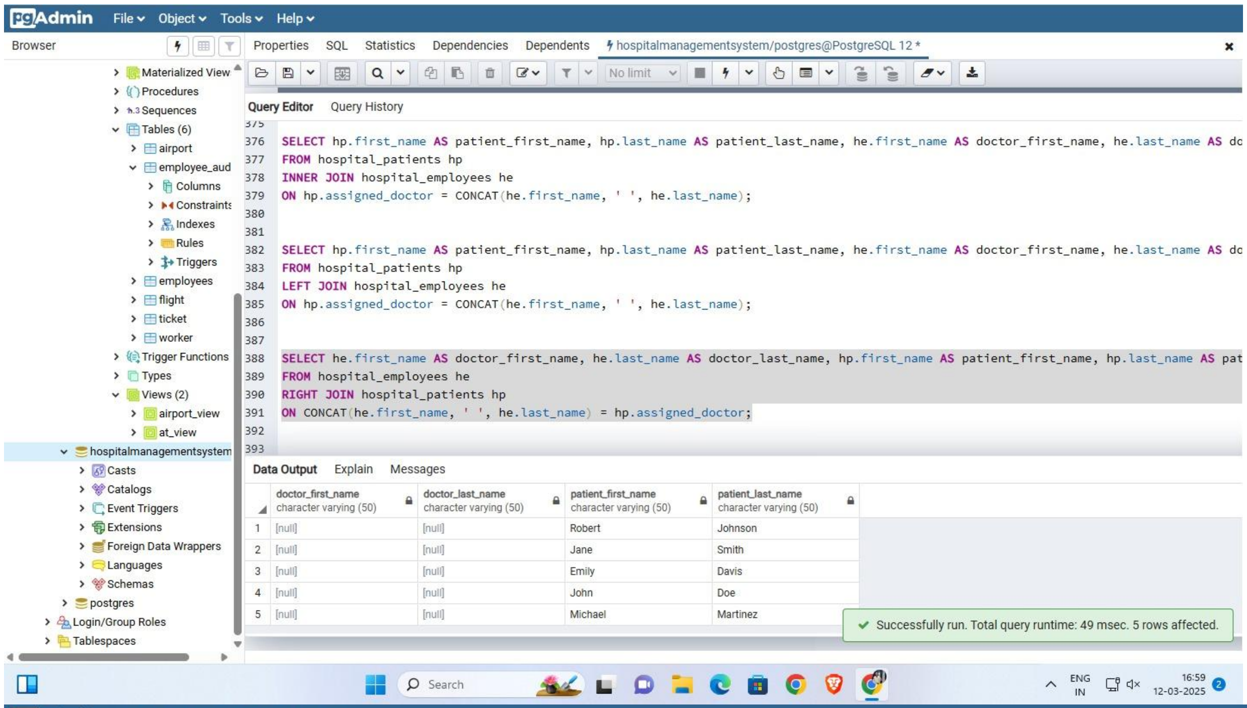
```
    is_active BOOLEAN
);

INSERT INTO hospital_employees (first_name, last_name, position, department, hire_date, salary, contact_number, is_active)
VALUES
('John', 'Doe', 'Doctor', 'Cardiology', '2020-01-10', 100000.00, '555-111-2222', true),
('Jane', 'Smith', 'Nurse', 'Pediatrics', '2019-07-20', 45000.00, '555-123-4567', true),
('Alice', 'Johnson', 'Technician', 'Radiology', '2018-05-15', 55000.00, '555-234-5678', true),
('Bob', 'Brown', 'Doctor', 'Neurology', '2017-02-25', 110000.00, '555-345-6789', false),
('Charlie', 'Davis', 'Nurse', 'Orthopedics', '2021-03-12', 48000.00, '555-456-7890', true),
('David', 'Williams', 'Technician', 'Laboratory', '2022-08-05', 60000.00, '555-567-8901', true);

SELECT * FROM hospital_employees;

SELECT hp.first_name AS patient_first_name, hp.last_name AS patient_last_name, he.first_name AS doctor_first_name, he.last_name AS do
FROM hospital_patients hp
INNER JOIN hospital_employees he
ON hp.assigned_doctor = CONCAT(he.first_name, ' ', he.last_name);
```

patient_first_name character varying (50)	patient_last_name character varying (50)	doctor_first_name character varying (50)	doctor_last_name character varying (50)



```
('John', 'Doe', 'Doctor', 'Cardiology', '2020-01-10', 100000.00, '555-111-2222', true),
('Jane', 'Smith', 'Nurse', 'Pediatrics', '2019-07-20', 45000.00, '555-123-4567', true),
('Alice', 'Johnson', 'Technician', 'Radiology', '2018-05-15', 55000.00, '555-234-5678', true),
('Bob', 'Brown', 'Doctor', 'Neurology', '2017-02-25', 110000.00, '555-345-6789', false),
('Charlie', 'Davis', 'Nurse', 'Orthopedics', '2021-03-12', 48000.00, '555-456-7890', true),
('David', 'Williams', 'Technician', 'Laboratory', '2022-08-05', 60000.00, '555-567-8901', true);

SELECT * FROM hospital_employees;

SELECT hp.first_name AS patient_first_name, hp.last_name AS patient_last_name, he.first_name AS doctor_first_name, he.last_name AS do
FROM hospital_patients hp
INNER JOIN hospital_employees he
ON hp.assigned_doctor = CONCAT(he.first_name, ' ', he.last_name);


SELECT hp.first_name AS patient_first_name, hp.last_name AS patient_last_name, he.first_name AS doctor_first_name, he.last_name AS do
FROM hospital_patients hp
LEFT JOIN hospital_employees he
ON hp.assigned_doctor = CONCAT(he.first_name, ' ', he.last_name);
```

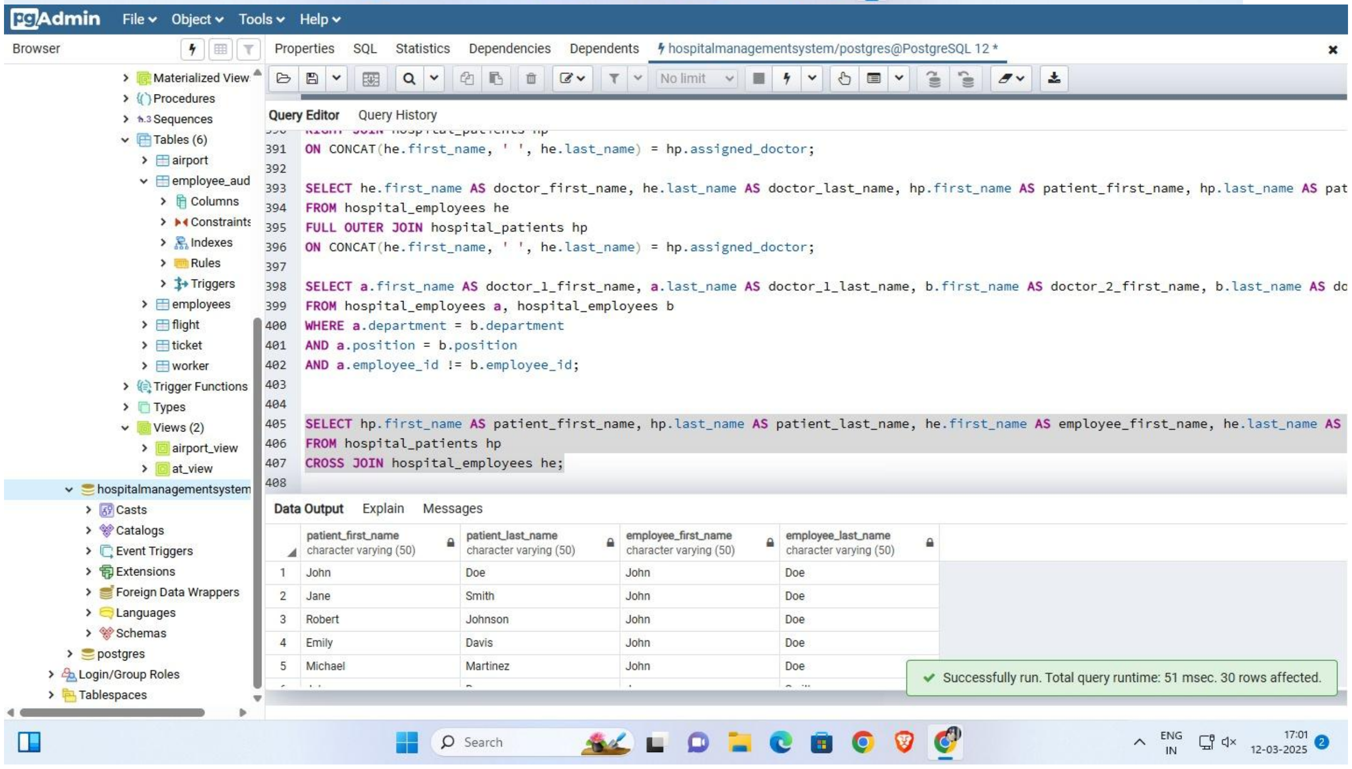
	patient_first_name character varying (50)	patient_last_name character varying (50)	doctor_first_name character varying (50)	doctor_last_name character varying (50)
1	Robert	Johnson	[null]	[null]
2	Jane	Smith	[null]	[null]
3	Emily	Davis	[null]	[null]
4	John	Doe	[null]	[null]
5	Michael	Martinez	[null]	[null]



```
SELECT hp.first_name AS patient_first_name, hp.last_name AS patient_last_name, he.first_name AS doctor_first_name, he.last_name AS do
FROM hospital_patients hp
INNER JOIN hospital_employees he
ON hp.assigned_doctor = CONCAT(he.first_name, ' ', he.last_name);


SELECT hp.first_name AS patient_first_name, hp.last_name AS patient_last_name, he.first_name AS doctor_first_name, he.last_name AS do
FROM hospital_patients hp
LEFT JOIN hospital_employees he
ON hp.assigned_doctor = CONCAT(he.first_name, ' ', he.last_name);


SELECT he.first_name AS doctor_first_name, he.last_name AS doctor_last_name, hp.first_name AS patient_first_name, hp.last_name AS pat
FROM hospital_employees he
RIGHT JOIN hospital_patients hp
ON CONCAT(he.first_name, ' ', he.last_name) = hp.assigned_doctor;
```

	doctor_first_name character varying (50)	doctor_last_name character varying (50)	patient_first_name character varying (50)	patient_last_name character varying (50)
1	[null]	[null]	Robert	Johnson
2	[null]	[null]	Jane	Smith
3	[null]	[null]	Emily	Davis
4	[null]	[null]	John	Doe
5	[null]	[null]	Michael	Martinez

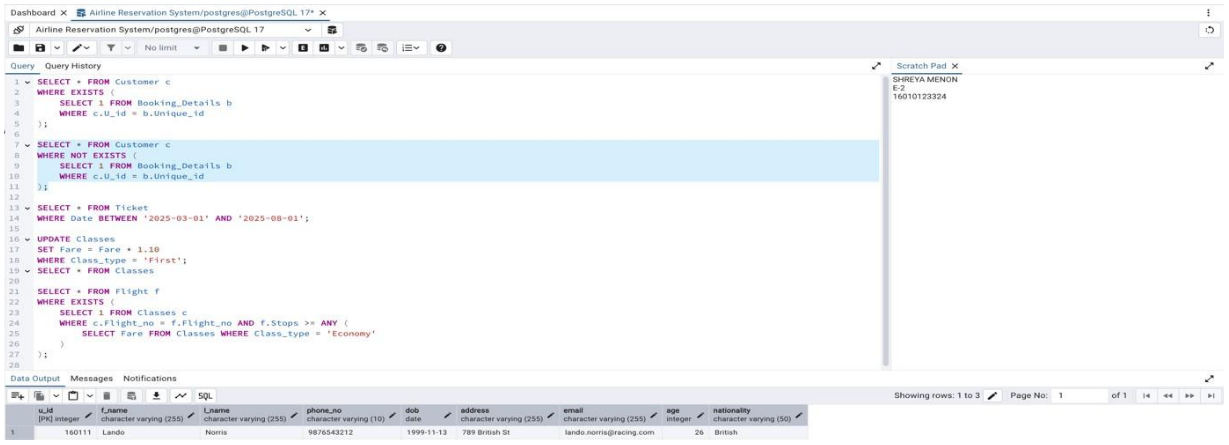
Successfully run. Total query runtime: 49 msec. 5 rows affected.



```
ON CONCAT(he.first_name, ' ', he.last_name) = hp.assigned_doctor;

SELECT he.first_name AS doctor_first_name, he.last_name AS doctor_last_name, hp.first_name AS patient_first_name, hp.last_name AS pat
FROM hospital_employees he
FULL OUTER JOIN hospital_patients hp
ON CONCAT(he.first_name, ' ', he.last_name) = hp.assigned_doctor;

SELECT a.first_name AS doctor_1_first_name, a.last_name AS doctor_1_last_name, b.first_name AS doctor_2_first_name, b.last_name AS dc
FROM hospital_employees a, hospital_employees b
WHERE a.department = b.department
AND a.position = b.position
AND a.employee_id != b.employee_id;


SELECT hp.first_name AS patient_first_name, hp.last_name AS patient_last_name, he.first_name AS employee_first_name, he.last_name AS
FROM hospital_patients hp
CROSS JOIN hospital_employees he;
```

	patient_first_name character varying (50)	patient_last_name character varying (50)	employee_first_name character varying (50)	employee_last_name character varying (50)
1	John	Doe	John	Doe
2	Jane	Smith	John	Doe
3	Robert	Johnson	John	Doe
4	Emily	Davis	John	Doe
5	Michael	Martinez	John	Doe

Successfully run. Total query runtime: 51 msec. 30 rows affected.



Dashboard × Airline Reservation System/postgres@PostgreSQL 17* ×

Airline Reservation System/postgres@PostgreSQL 17

```sql
SELECT * FROM Customer c
WHERE EXISTS (
    SELECT 1 FROM Booking_Details b
    WHERE c.U_id = b.Unique_id
);

SELECT * FROM Customer c
WHERE NOT EXISTS (
    SELECT 1 FROM Booking_Details b
    WHERE c.U_id = b.Unique_id
);

SELECT * FROM Ticket
WHERE Date BETWEEN '2025-03-01' AND '2025-08-01';

UPDATE Classes
SET Fare = Fare * 1.10
WHERE Class_type = 'First';
SELECT * FROM Classes

SELECT * FROM Flight f
WHERE EXISTS (
    SELECT 1 FROM Classes c
    WHERE c.Flight_no = f.Flight_no AND f.Stops >= ANY (
        SELECT Fare FROM Classes WHERE Class_type = 'Economy'
    )
);
```

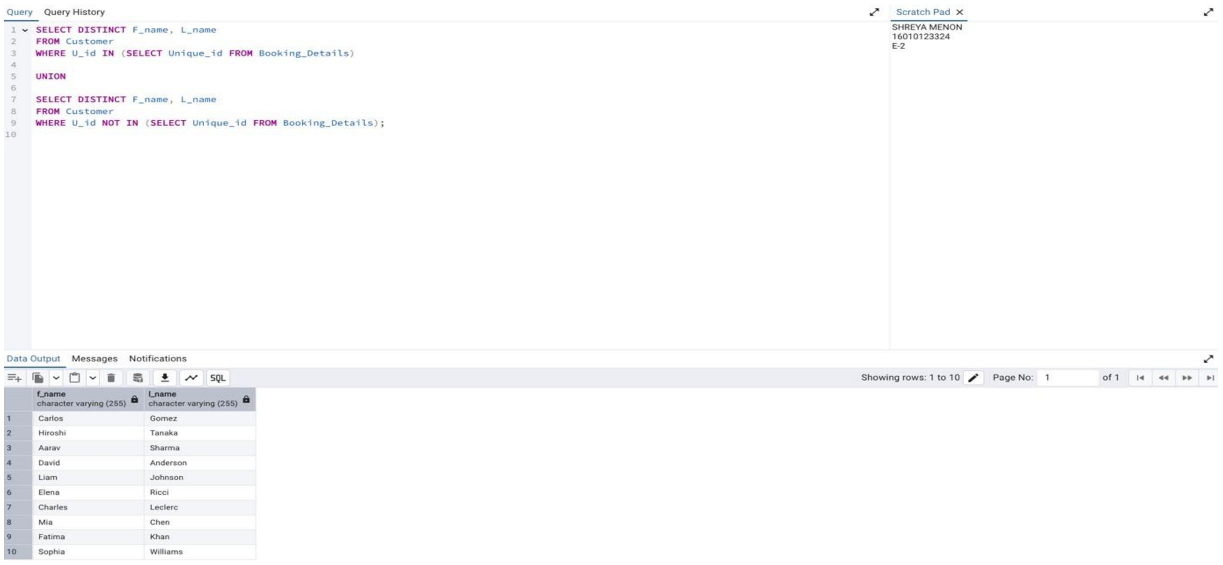
Scratch Pad: SHREYA MENON
E-2
16010123324

Showing rows: 1 to 3 Page No: 1 of 1

	u_id [PK] integer	f_name character varying (255)	l_name character varying (255)	phone_no character varying (10)	dob date	address character varying (255)	email character varying (255)	age integer	nationality character varying (50)
1	160111	Lando	Norris	9876543212	1999-11-13	789 British St	lando.norris@racing.com	26	British

Set Operations

UNION



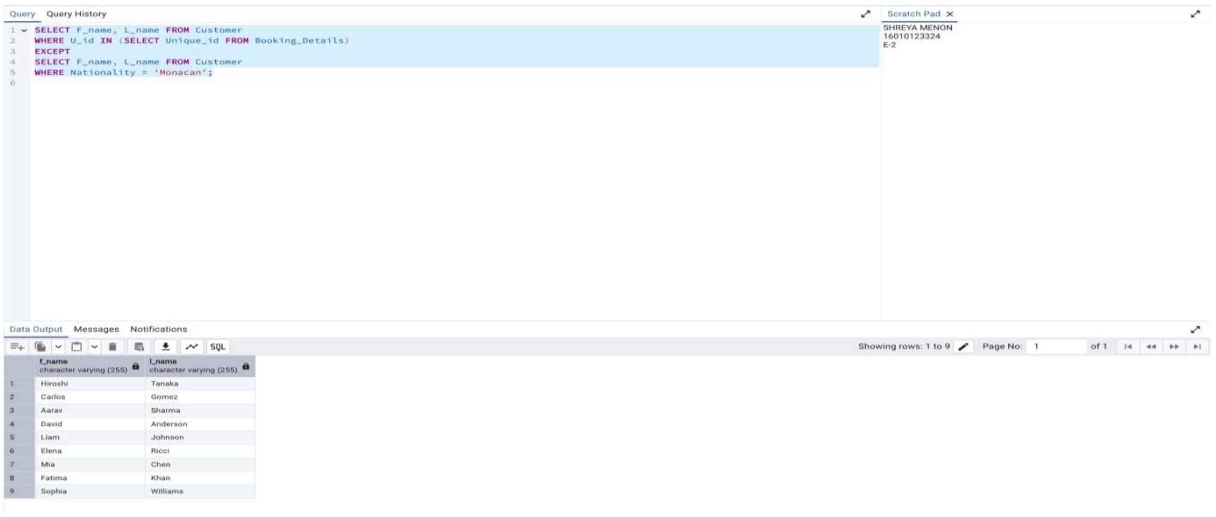
```sql
SELECT DISTINCT F_name, L_name
FROM Customer
WHERE U_id IN (SELECT Unique_id FROM Booking_Details)

UNION

SELECT DISTINCT F_name, L_name
FROM Customer
WHERE U_id NOT IN (SELECT Unique_id FROM Booking_Details);
```

Scratch Pad: SHREYA MENON
16010123324
E-2

Showing rows: 1 to 10 Page No: 1 of 1

	f_name character varying (255)	l_name character varying (255)
1	Carlos	Gomez
2	Hiroshi	Tanaka
3	Aarav	Sharma
4	David	Anderson
5	Liam	Johnson
6	Elena	Ricci
7	Charles	Leclerc
8	Mia	Chen
9	Fatima	Khan
10	Sophia	Williams

MINUS/EXCEPT



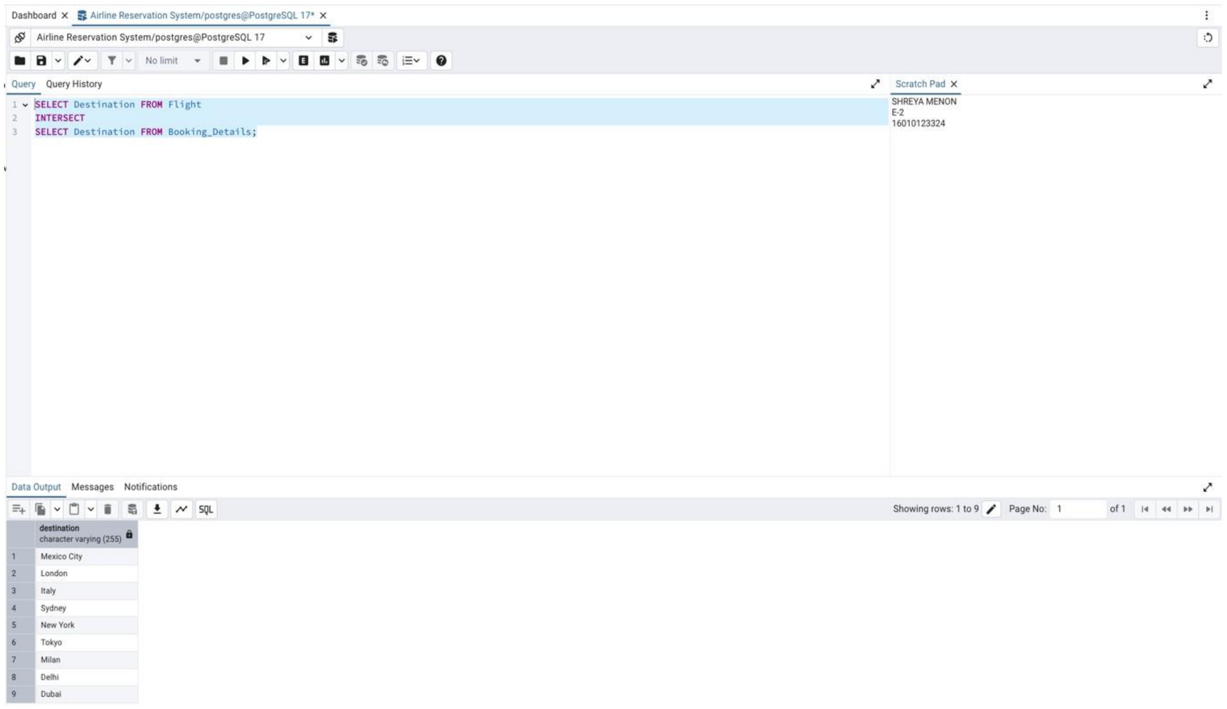
```
SELECT F_name, L_name FROM Customer
WHERE U_id IN (SELECT Unique_id FROM Booking_Details)
EXCEPT
SELECT F_name, L_name FROM Customer
WHERE Nationality = 'Monacan';
```

SHREYA MENON
16010123324
E-2

Showing rows: 1 to 9 Page No: 1 of 1

	f_name character varying (255)	l_name character varying (255)
1	Hiroshi	Tanaka
2	Carlos	Gomez
3	Aarav	Sharma
4	David	Anderson
5	Liam	Johnson
6	Elena	Ricci
7	Mia	Chen
8	Fatima	Khan
9	Sophia	Williams

INTERSECT



Airline Reservation System/postgres@PostgreSQL 17

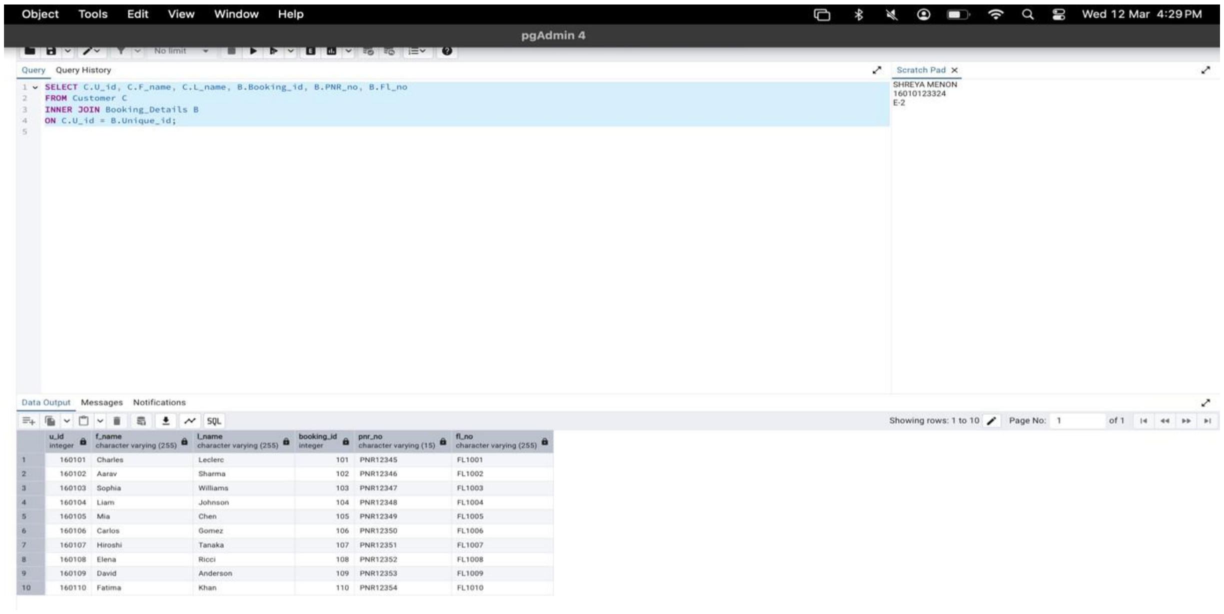
```
SELECT Destination FROM Flight
INTERSECT
SELECT Destination FROM Booking_Details;
```

SHREYA MENON
E-2
16010123324

Showing rows: 1 to 9 Page No: 1 of 1

	destination character varying (255)
1	Mexico City
2	London
3	Italy
4	Sydney
5	New York
6	Tokyo
7	Milan
8	Delhi
9	Dubai

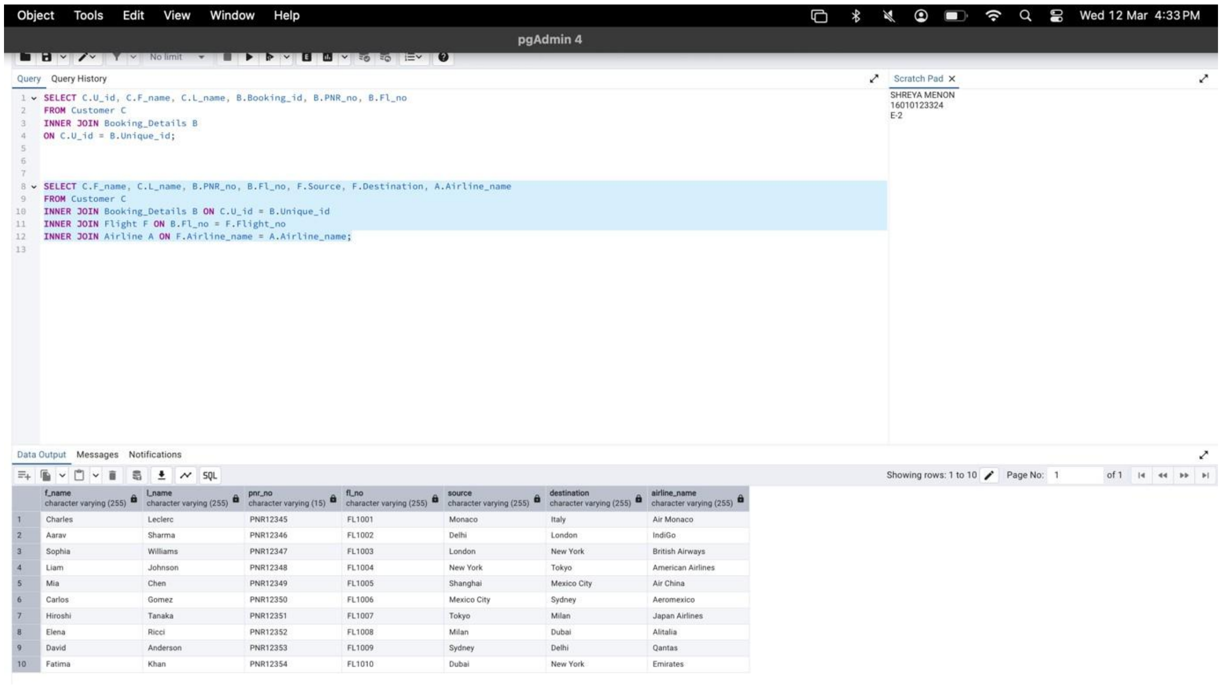
INNER JOIN & Multiple INNER JOIN



```sql
SELECT C.U_id, C.F_name, C.L_name, B.Booking_id, B.PNR_no, B.Fl_no
FROM Customer C
INNER JOIN Booking_Details B
ON C.U_id = B.Unique_id;
```

SHREYA MENON
16010123324
E-2

	u_id integer	f_name character varying (255)	l_name character varying (255)	booking_id integer	pnr_no character varying (15)	fl_no character varying (255)
1	160101	Charles	Leclerc	101	PNR12345	FL1001
2	160102	Aarav	Sharma	102	PNR12346	FL1002
3	160103	Sophia	Williams	103	PNR12347	FL1003
4	160104	Liam	Johnson	104	PNR12348	FL1004
5	160105	Mia	Chen	105	PNR12349	FL1005
6	160106	Carlos	Gomez	106	PNR12350	FL1006
7	160107	Hiroshi	Tanaka	107	PNR12351	FL1007
8	160108	Elena	Ricci	108	PNR12352	FL1008
9	160109	David	Anderson	109	PNR12353	FL1009
10	160110	Fatima	Khan	110	PNR12354	FL1010

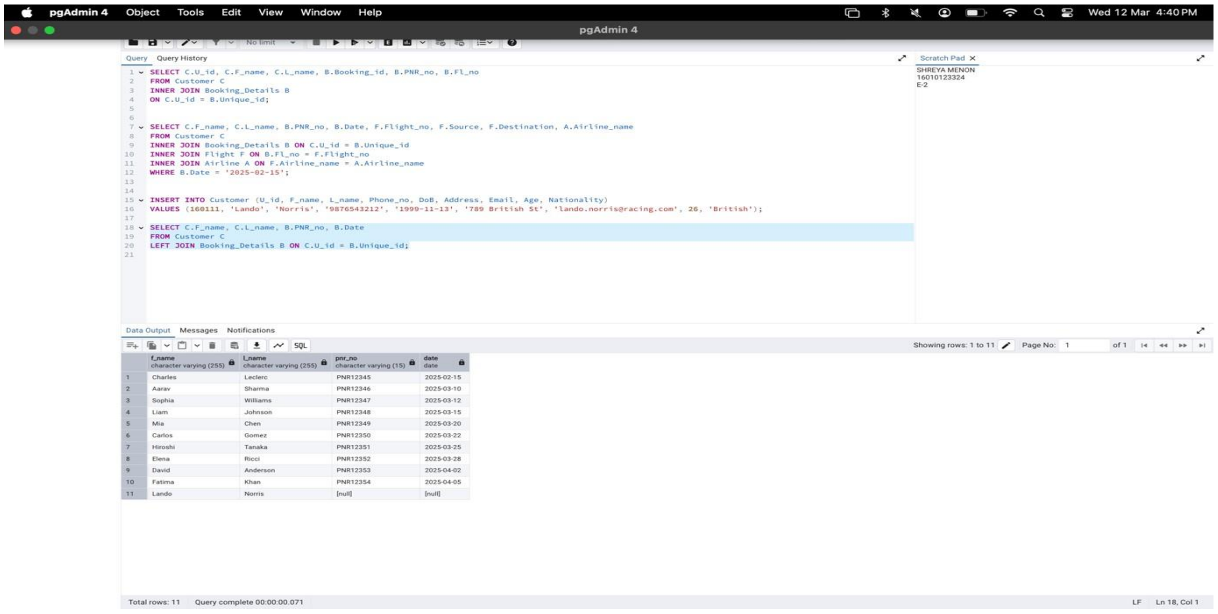


```sql
SELECT C.U_id, C.F_name, C.L_name, B.Booking_id, B.PNR_no, B.Fl_no
FROM Customer C
INNER JOIN Booking_Details B
ON C.U_id = B.Unique_id;



SELECT C.F_name, C.L_name, B.PNR_no, B.Fl_no, F.Source, F.Destination, A.Airline_name
FROM Customer C
INNER JOIN Booking_Details B ON C.U_id = B.Unique_id
INNER JOIN Flight F ON B.Fl_no = F.Flight_no
INNER JOIN Airline A ON F.Airline_name = A.Airline_name;
```

SHREYA MENON
16010123324
E-2

	f_name character varying (255)	l_name character varying (255)	pnr_no character varying (15)	fl_no character varying (255)	source character varying (255)	destination character varying (255)	airline_name character varying (255)
1	Charles	Leclerc	PNR12345	FL1001	Monaco	Italy	Air Monaco
2	Aarav	Sharma	PNR12346	FL1002	Delhi	London	IndiGo
3	Sophia	Williams	PNR12347	FL1003	London	New York	British Airways
4	Liam	Johnson	PNR12348	FL1004	New York	Tokyo	American Airlines
5	Mia	Chen	PNR12349	FL1005	Shanghai	Mexico City	Air China
6	Carlos	Gomez	PNR12350	FL1006	Mexico City	Sydney	Aeromexico
7	Hiroshi	Tanaka	PNR12351	FL1007	Tokyo	Milan	Japan Airlines
8	Elena	Ricci	PNR12352	FL1008	Milan	Dubai	Alitalia
9	David	Anderson	PNR12353	FL1009	Sydney	Delhi	Qantas
10	Fatima	Khan	PNR12354	FL1010	Dubai	New York	Emirates

LEFT JOIN



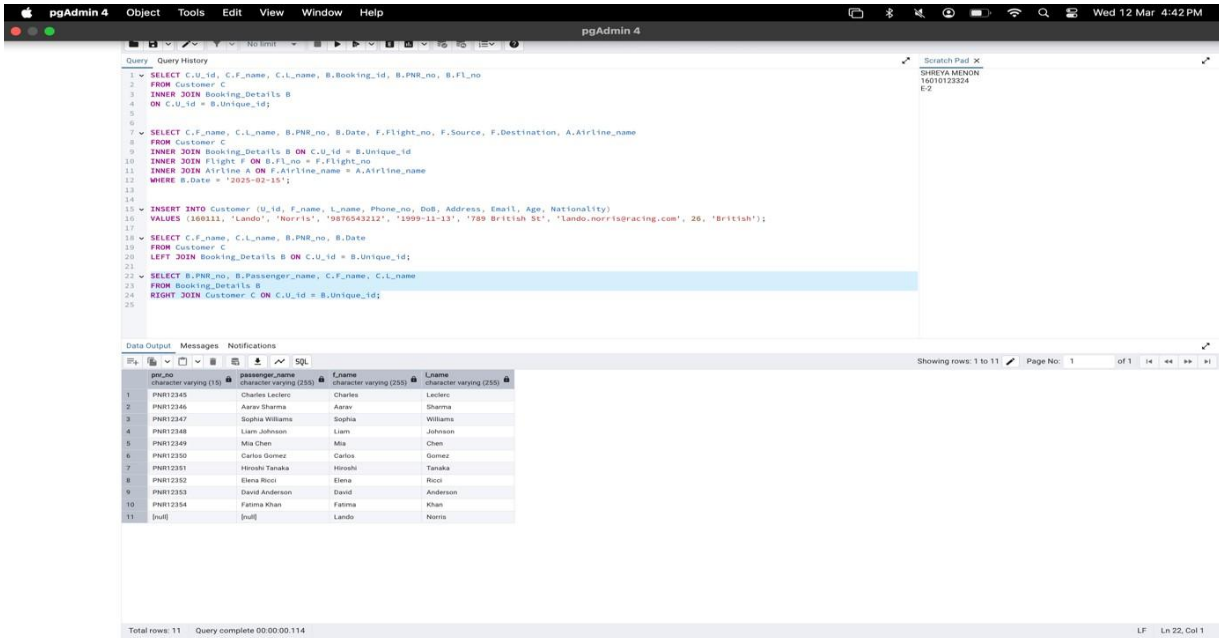
```sql
SELECT C.U_id, C.F_name, C.L_name, B.Booking_id, B.PNR_no, B.Fl_no
FROM Customer C
INNER JOIN Booking_Details B
ON C.U_id = B.Unique_id;


SELECT C.F_name, C.L_name, B.PNR_no, B.Date, F.Flight_no, F.Source, F.Destination, A.Airline_name
FROM Customer C
INNER JOIN Booking_Details B ON C.U_id = B.Unique_id
INNER JOIN Flight F ON B.Fl_no = F.Flight_no
INNER JOIN Airline A ON F.Airline_name = A.Airline_name
WHERE B.Date = '2025-02-15';


INSERT INTO Customer (U_id, F_name, L_name, Phone_no, DoB, Address, Email, Age, Nationality)
VALUES (160111, 'Lando', 'Norris', '9876543212', '1999-11-13', '789 British St', 'lando.norris@racing.com', 26, 'British');

SELECT C.F_name, C.L_name, B.PNR_no, B.Date
FROM Customer C
LEFT JOIN Booking_Details B ON C.U_id = B.Unique_id;
```

SHREYA MENON
16010123324
E-2

	f_name character varying (255)	l_name character varying (255)	pnr_no character varying (15)	date date
1	Charles	Leclerc	PNR12345	2025-02-15
2	Aarav	Sharma	PNR12346	2025-03-10
3	Sophia	Williams	PNR12347	2025-03-12
4	Liam	Johnson	PNR12348	2025-03-15
5	Mia	Chen	PNR12349	2025-03-20
6	Carlos	Gomez	PNR12350	2025-03-22
7	Hiroshi	Tanaka	PNR12351	2025-03-25
8	Elena	Ricci	PNR12352	2025-03-28
9	David	Anderson	PNR12353	2025-04-02
10	Fatima	Khan	PNR12354	2025-04-05
11	Lando	Norris	[null]	[null]

Total rows: 11 Query complete 00:00:00.071

RIGHT JOIN



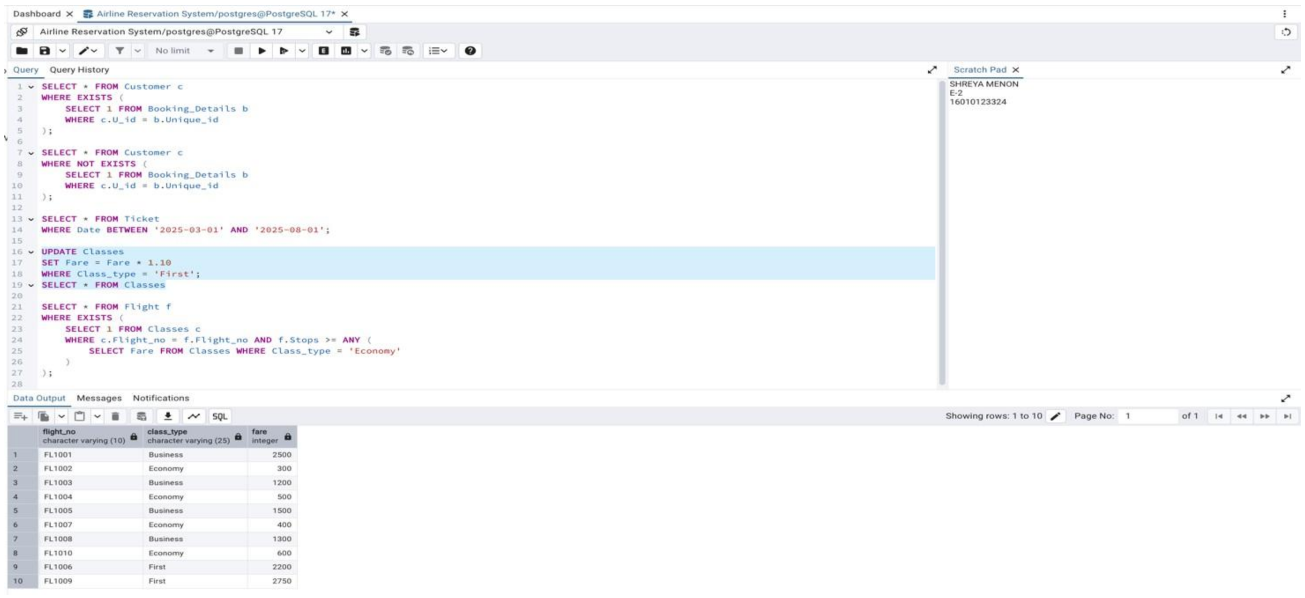
```sql
SELECT C.U_id, C.F_name, C.L_name, B.Booking_id, B.PNR_no, B.Fl_no
FROM Customer C
INNER JOIN Booking_Details B
ON C.U_id = B.Unique_id;


SELECT C.F_name, C.L_name, B.PNR_no, B.Date, F.Flight_no, F.Source, F.Destination, A.Airline_name
FROM Customer C
INNER JOIN Booking_Details B ON C.U_id = B.Unique_id
INNER JOIN Flight F ON B.Fl_no = F.Flight_no
INNER JOIN Airline A ON F.Airline_name = A.Airline_name
WHERE B.Date = '2025-02-15';


INSERT INTO Customer (U_id, F_name, L_name, Phone_no, DoB, Address, Email, Age, Nationality)
VALUES (160111, 'Lando', 'Norris', '9876543212', '1999-11-13', '789 British St', 'lando.norris@racing.com', 26, 'British');

SELECT C.F_name, C.L_name, B.PNR_no, B.Date
FROM Customer C
LEFT JOIN Booking_Details B ON C.U_id = B.Unique_id;

SELECT B.PNR_no, B.Passenger_name, C.F_name, C.L_name
FROM Booking_Details B
RIGHT JOIN Customer C ON C.U_id = B.Unique_id;
```

SHREYA MENON
16010123324
E-2

	pnr_no character varying (15)	passenger_name character varying (255)	f_name character varying (255)	l_name character varying (255)
1	PNR12345	Charles Leclerc	Charles	Leclerc
2	PNR12346	Aarav Sharma	Aarav	Sharma
3	PNR12347	Sophia Williams	Sophia	Williams
4	PNR12348	Liam Johnson	Liam	Johnson
5	PNR12349	Mia Chen	Mia	Chen
6	PNR12350	Carlos Gomez	Carlos	Gomez
7	PNR12351	Hiroshi Tanaka	Hiroshi	Tanaka
8	PNR12352	Elena Ricci	Elena	Ricci
9	PNR12353	David Anderson	David	Anderson
10	PNR12354	Fatima Khan	Fatima	Khan
11	[null]	[null]	Lando	Norris

Total rows: 11 Query complete 00:00:00.114

UPDATE



SHREYA MENON
E-2
16010123324

```sql
SELECT * FROM Customer c
WHERE EXISTS (
    SELECT 1 FROM Booking_Details b
    WHERE c.U_id = b.Unique_id
);

SELECT * FROM Customer c
WHERE NOT EXISTS (
    SELECT 1 FROM Booking_Details b
    WHERE c.U_id = b.Unique_id
);

SELECT * FROM Ticket
WHERE Date BETWEEN '2025-03-01' AND '2025-08-01';

UPDATE Classes
SET Fare = Fare * 1.10
WHERE Class_type = 'First';
SELECT * FROM Classes

SELECT * FROM Flight f
WHERE EXISTS (
    SELECT 1 FROM Classes c
    WHERE c.Flight_no = f.Flight_no AND f.Stops >= ANY (
        SELECT Fare FROM Classes WHERE Class_type = 'Economy'
    )
);
```

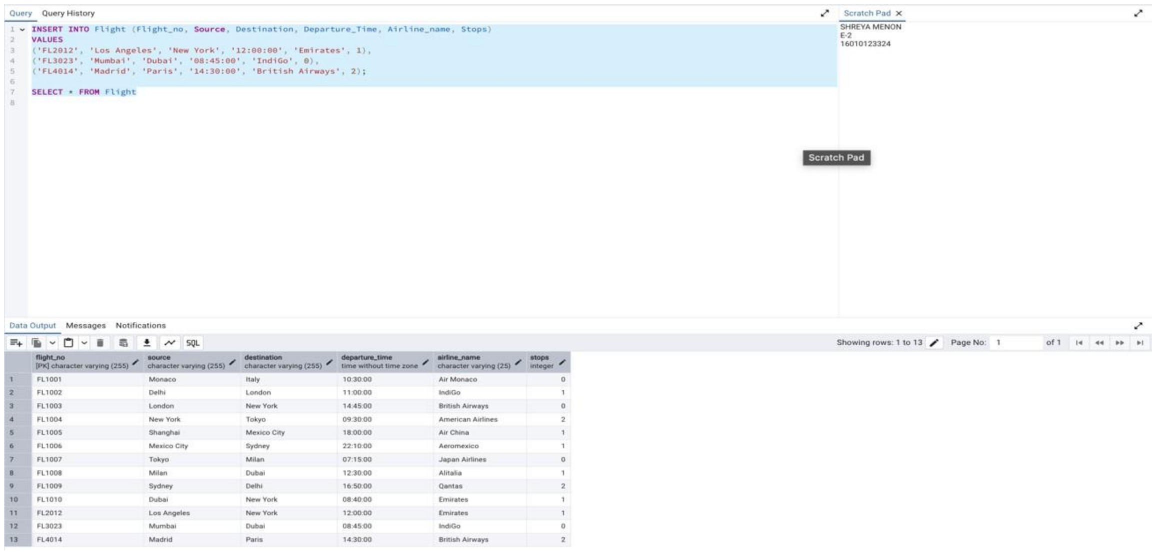
	flight_no character varying (10)	class_type character varying (25)	fare integer
1	FL1001	Business	2500
2	FL1002	Economy	300
3	FL1003	Business	1200
4	FL1004	Economy	500
5	FL1005	Business	1500
6	FL1007	Economy	400
7	FL1008	Business	1300
8	FL1010	Economy	600
9	FL1006	First	2200
10	FL1009	First	2750

Conclusion:

The above experiment highlights DML operations like use of aggregate functions, set operations and joins on our database.

Post lab queries:

1. **W.r.t your table give SQL query to insert more than one record at a time**



```
INSERT INTO Flight (Flight_no, Source, Destination, Departure_Time, Airline_name, Stops)
VALUES
('FL2012', 'Los Angeles', 'New York', '12:00:00', 'Emirates', 1),
('FL3023', 'Mumbai', 'Dubai', '08:45:00', 'IndiGo', 0),
('FL4014', 'Madrid', 'Paris', '14:30:00', 'British Airways', 2);

SELECT * FROM Flight
```

SHREYA MENON
E-2
16010123324

	flight_no [PK] character varying (255)	source character varying (255)	destination character varying (255)	departure_time time without time zone	airline_name character varying (25)	stops integer
1	FL1001	Monaco	Italy	10:30:00	Air Monaco	0
2	FL1002	Delhi	London	11:00:00	IndiGo	1
3	FL1003	London	New York	14:45:00	British Airways	0
4	FL1004	New York	Tokyo	09:30:00	American Airlines	2
5	FL1005	Shanghai	Mexico City	18:00:00	Air China	1
6	FL1006	Mexico City	Sydney	22:10:00	Aeromexico	1
7	FL1007	Tokyo	Milan	07:15:00	Japan Airlines	0
8	FL1008	Milan	Dubai	12:30:00	Alitalia	1
9	FL1009	Sydney	Delhi	16:50:00	Qantas	2
10	FL1010	Dubai	New York	08:40:00	Emirates	1
11	FL2012	Los Angeles	New York	12:00:00	Emirates	1
12	FL3023	Mumbai	Dubai	08:45:00	IndiGo	0
13	FL4014	Madrid	Paris	14:30:00	British Airways	2

2. **What is the difference between Join and full outer join operation**

A **JOIN** (by default, an **INNER JOIN**) returns only the records that have matching values in both tables. If a record in one table doesn't have a corresponding match in the other, it is excluded from the result.

A **FULL OUTER JOIN**, on the other hand, returns all records from both tables. If there is a match, it displays the matching data. If no match is found, NULL values are returned for the missing data from the other table.

- **JOIN (INNER JOIN)** → Only matching rows.
- **FULL OUTER JOIN** → All rows from both tables, with NULLs where no match is found.