

## Laboratory work 8

1. Create a view to show details of all flights that are departing on a specific date.

The screenshot shows a database IDE with the following components:

- Database Explorer:** Shows a database named 'airport\_bd' with tables 'baggage\_check', 'boarding\_pass', and 'booking'.
- Console:** Displays the SQL code for creating and querying the view 'flights\_dep'. The code is: 

```
CREATE OR REPLACE VIEW flights_dep AS SELECT * FROM flights WHERE DATE (sch_departure_time) = DATE '2025-09-03'; SELECT * FROM flights_dep;
```
- Database Sessions:** Shows a session for 'postgres@localhost [2]'.
- Output:** Displays the results of the query, showing 22 rows of flight data for the date 2025-09-03. The columns are: flight\_id, sch\_departure\_time, sch\_arrival\_time, departing\_airport\_id, and arriving\_airport\_id.

flight_id	sch_departure_time	sch_arrival_time	departing_airport_id	arriving_airport_id
31	2025-09-03 06:00:00.000000	2025-09-03 08:30:00.000000	1	
32	2025-09-03 06:45:00.000000	2025-09-03 09:15:00.000000	5	
33	2025-09-03 07:20:00.000000	2025-09-03 09:50:00.000000	9	
34	2025-09-03 08:10:00.000000	2025-09-03 10:40:00.000000	3	
35	2025-09-03 09:00:00.000000	2025-09-03 11:30:00.000000	7	
36	2025-09-03 09:45:00.000000	2025-09-03 12:15:00.000000	12	
37	2025-09-03 10:30:00.000000	2025-09-03 13:00:00.000000	15	
38	2025-09-03 11:15:00.000000	2025-09-03 13:45:00.000000	4	
39	2025-09-03 12:00:00.000000	2025-09-03 14:30:00.000000	6	
40	2025-09-03 12:45:00.000000	2025-09-03 15:15:00.000000	8	
41	2025-09-03 13:30:00.000000	2025-09-03 16:00:00.000000	11	
42	2025-09-03 14:15:00.000000	2025-09-03 16:45:00.000000	14	
43	2025-09-03 15:00:00.000000	2025-09-03 17:30:00.000000	17	
44	2025-09-03 15:45:00.000000	2025-09-03 18:15:00.000000	20	
45	2025-09-03 16:30:00.000000	2025-09-03 19:00:00.000000	1	

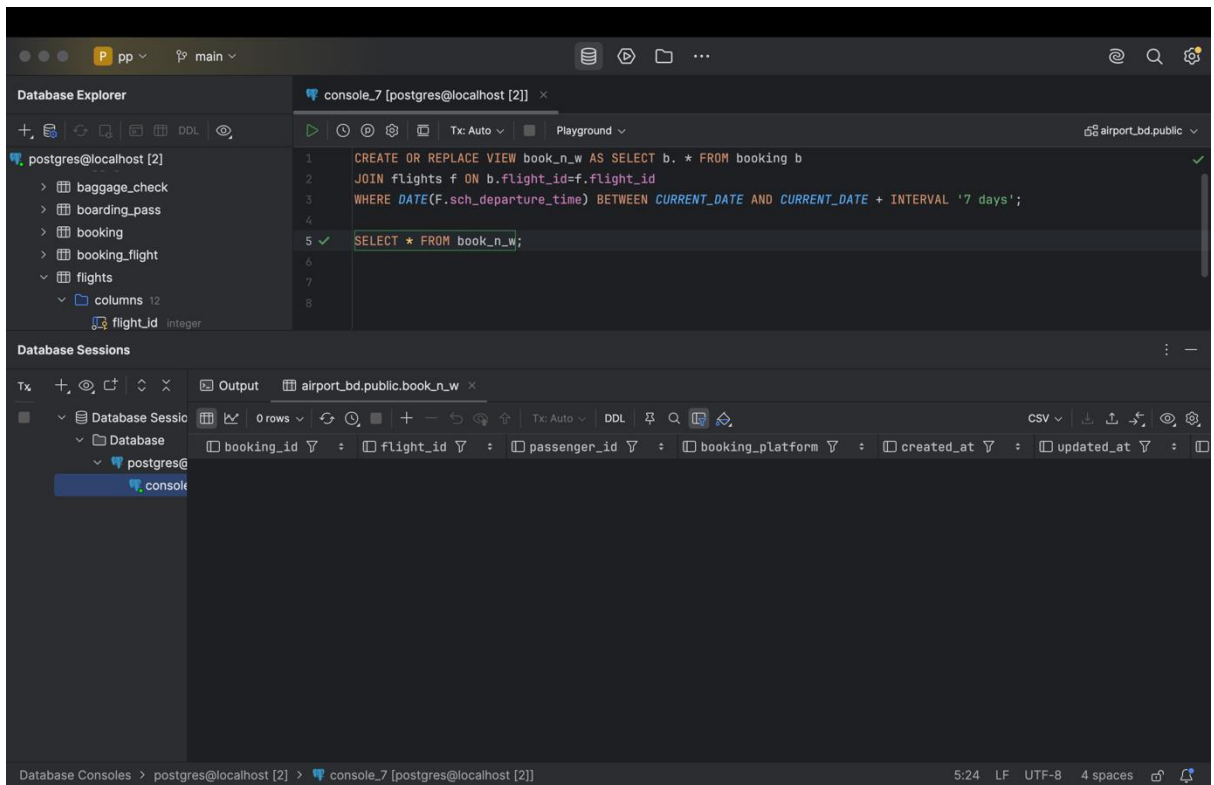
```
CREATE OR REPLACE VIEW flights_dep AS SELECT * FROM flights
WHERE DATE (sch_departure_time) = DATE '2025-09-03';

SELECT * FROM flights_dep;
```

2. Create a view that shows bookings for flights scheduled to depart within the next week.

```
CREATE OR REPLACE VIEW book_n_w AS SELECT b.* FROM booking b
JOIN flights f ON b.flight_id=f.flight_id
WHERE DATE(F.sch_departure_time) BETWEEN CURRENT_DATE AND CURRENT_DATE +
INTERVAL '7 days';

SELECT * FROM book_n_w;
```



3. Create a view to show the top 5 most popular flight routes based on the number of bookings.

```
CREATE OR REPLACE VIEW top5 AS
SELECT f.departing_airport_id,
       f.arriving_airport_id,
       COUNT(b.booking_id) AS total_booking
FROM flights f
JOIN booking b ON f.flight_id=b.flight_id
GROUP BY f.departing_airport_id, f.arriving_airport_id
ORDER BY total_booking DESC LIMIT 5;

SELECT * FROM top5;
```

The screenshot shows a database console interface with the following components:

- Database Explorer:** A tree view on the left showing the database structure. The 'flights' table is expanded, showing columns 'flight\_id' (integer), 'sch\_departure\_time' (timestamp), and 'sch\_arrival\_time' (timestamp).
- console\_7 [postgres@localhost [2]]:** The main editor area showing SQL code:
 

```
1 CREATE OR REPLACE VIEW top5 AS
2 SELECT f.departing_airport_id,
3        f.arriving_airport_id,
4        COUNT(b.booking_id) AS total_booking
5 FROM flights f
6 JOIN booking b ON f.flight_id=b.flight_id
7 GROUP BY f.departing_airport_id, f.arriving_airport_id
8 ORDER BY total_booking DESC LIMIT 5;
9
10 SELECT * FROM top5;
```
- Database Sessions:** A section at the bottom showing the current session and its output.

The output of the query is displayed in a table with 5 rows and 3 columns: departing\_airport\_id, arriving\_airport\_id, and total\_booking.

departing_airport_id	arriving_airport_id	total_booking
4	19	20
20	9	20
14	31	20
11	22	20
16	8	20

4. Create a view that lists all flights for a specific airline.

```
CREATE OR REPLACE VIEW flights_by_airline AS
SELECT * FROM flights
WHERE airline_id = 1;

SELECT * FROM flights_by_airline;
```

The screenshot shows a PostgreSQL database console interface. On the left, the 'Database Explorer' pane shows a tree structure with 'booking', 'booking\_flight', and 'flights' (containing 'columns' and 'flight\_id'). The main console area shows the following SQL commands:

```
1 CREATE OR REPLACE VIEW flights_by_airline AS
2 SELECT * FROM flights
3 WHERE airline_id = 1;
4
5 SELECT * FROM flights_by_airline;
6
7
```

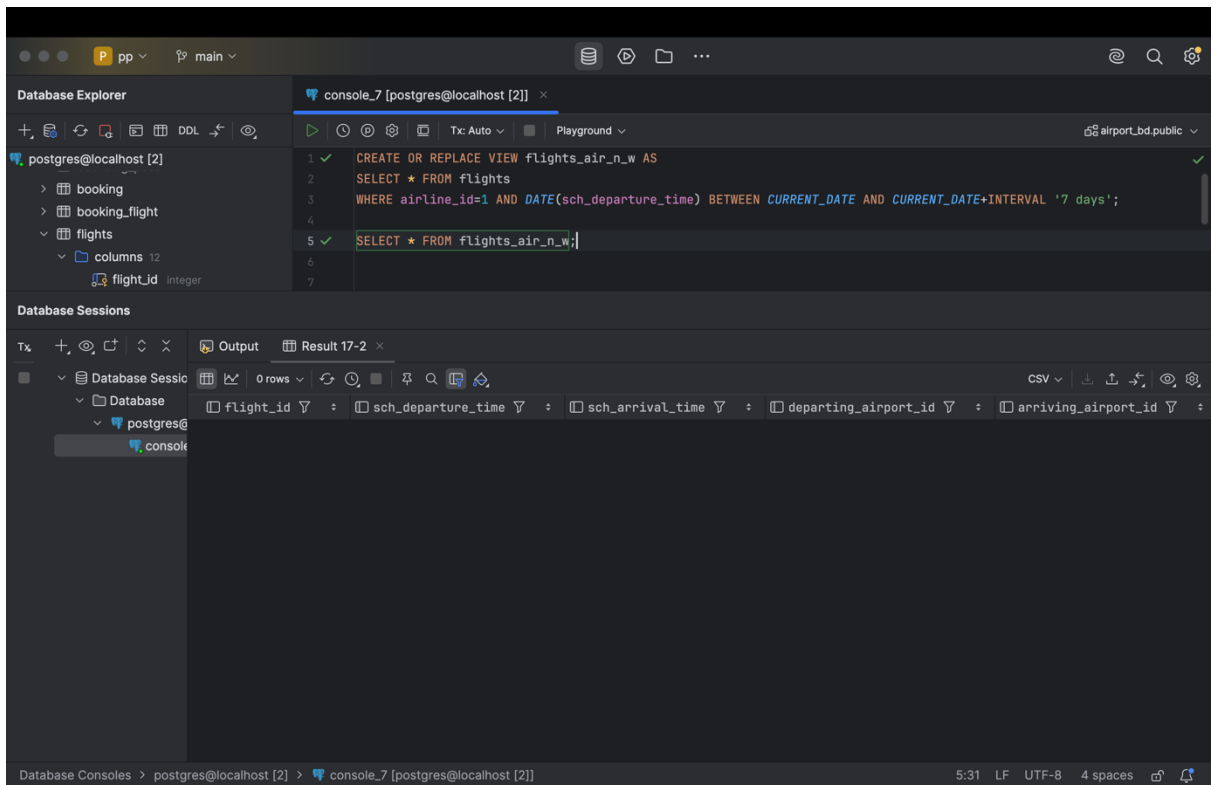
Below the console, the 'Database Sessions' pane shows a table of results for the query 'airport\_bd.public.flights\_by\_airline'. The table has 10 rows and 5 columns: flight\_id, sch\_departure\_time, sch\_arrival\_time, departing\_airport\_id, and arriving\_airport\_id.

flight_id	sch_departure_time	sch_arrival_time	departing_airport_id	arriving_airport_id
4	2025-09-01 11:30:00.000000	2025-09-01 14:00:00.000000	3	
31	2025-09-03 06:00:00.000000	2025-09-03 08:30:00.000000	1	
50	2025-09-03 20:15:00.000000	2025-09-03 22:45:00.000000	16	
70	2025-09-04 18:45:00.000000	2025-09-04 21:15:00.000000	17	
90	2025-09-05 17:15:00.000000	2025-09-05 19:45:00.000000	17	
110	2025-09-06 15:45:00.000000	2025-09-06 18:15:00.000000	17	
130	2025-09-07 14:15:00.000000	2025-09-07 16:45:00.000000	17	
150	2025-09-08 12:45:00.000000	2025-09-08 15:15:00.000000	17	
170	2025-09-09 11:15:00.000000	2025-09-09 13:45:00.000000	17	
190	2025-09-10 09:45:00.000000	2025-09-10 12:15:00.000000	17	

5. Modify the view created in task 4 to show only flights departing within the next 7 days for a specific airline.

```
CREATE OR REPLACE VIEW flights_air_n_w AS
SELECT * FROM flights
WHERE airline_id=1 AND DATE(sch_departure_time) BETWEEN CURRENT_DATE
AND CURRENT_DATE+INTERVAL '7 days';

SELECT * FROM flights_air_n_w;
```



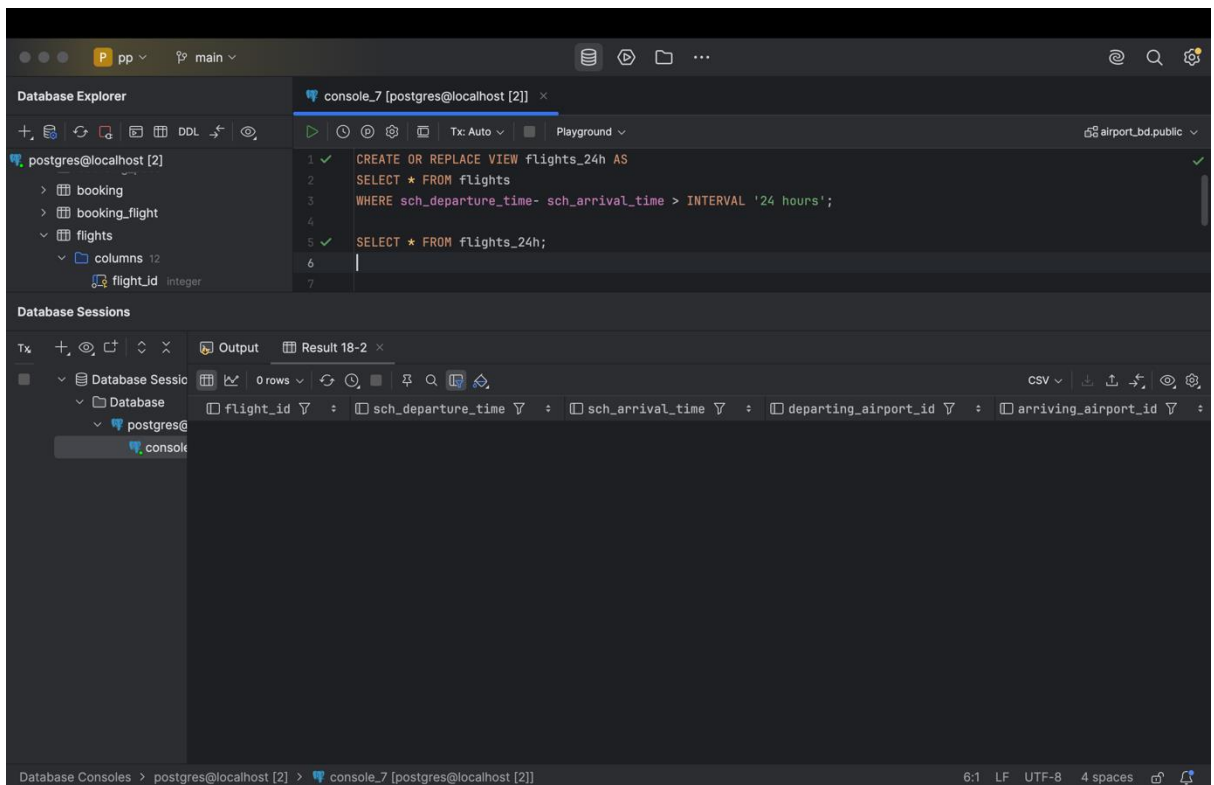
6. Create a view to show flights that are delayed by more than 24 hours.

```

CREATE OR REPLACE VIEW flights_24h AS
SELECT * FROM flights
WHERE sch_departure_time- sch_arrival_time > INTERVAL '24 hours';

SELECT * FROM flights_24h;

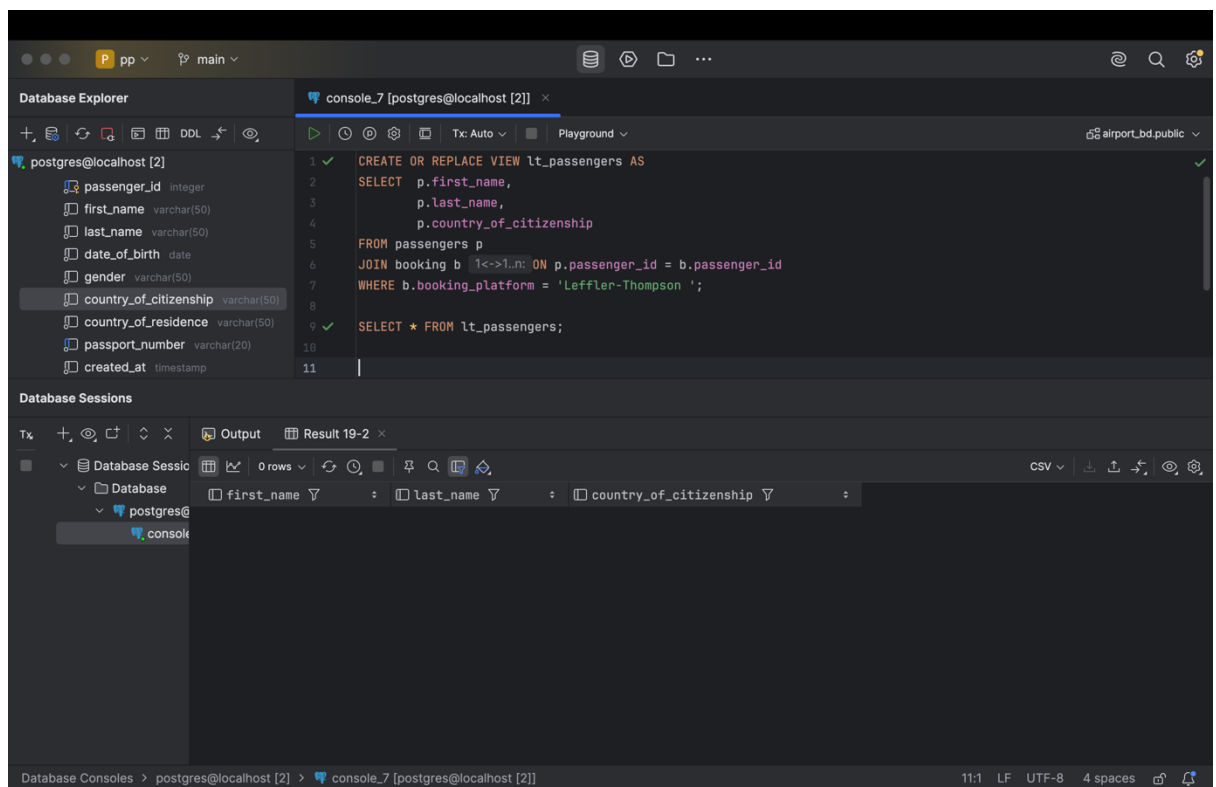
```



7. Create a view in which you can display the full name and country of origin of passengers who made bookings on Leffler-Thompson platform. Then show the list of that passengers.

```
CREATE OR REPLACE VIEW lt_passengers AS
SELECT  p.first_name,
        p.last_name,
        p.country_of_citizenship
FROM passengers p
JOIN booking b ON p.passenger_id = b.passenger_id
WHERE b.booking_platform = 'Leffler-Thompson ';

SELECT * FROM lt_passengers;
```



8. Create a view that shows top 10 most visited countries.

```
CREATE OR REPLACE VIEW top10_c AS
SELECT a.country,
       COUNT(f.flight_id) AS total_flight
FROM flights f
JOIN airport a ON f.arriving_airport_id = a.airport_id
GROUP BY a.country
ORDER BY total_flight DESC
LIMIT 10;

SELECT * FROM top10_c;
```

The screenshot shows a PostgreSQL database console interface. On the left, the 'Database Explorer' pane shows a database named 'airport' with a table 'airports' and its columns: 'airport\_id' (integer), 'airport\_name' (varchar(50)), 'country' (varchar(50)), 'state' (varchar(50)), 'city' (varchar(50)), 'created\_at' (timestamp), and 'updated\_at' (timestamp). The main console area shows the following SQL script:

```

1 CREATE OR REPLACE VIEW top10_c AS
2 SELECT a.country,
3        COUNT(f.flight_id) AS total_flight
4 FROM flights f
5 JOIN airport a ON f.arriving_airport_id = a.airport_id
6 GROUP BY a.country
7 ORDER BY total_flight DESC
8 LIMIT 10;
9
10 SELECT * FROM top10_c;
11

```

The 'Database Sessions' pane at the bottom shows the execution of the query, resulting in a table with 10 rows. The table has two columns: 'country' and 'total\_flight'.

country	total_flight
USA	67
Germany	28
Australia	11
Japan	10
China	10
Canada	10
South Korea	10
UAE	9
Brazil	9
France	9

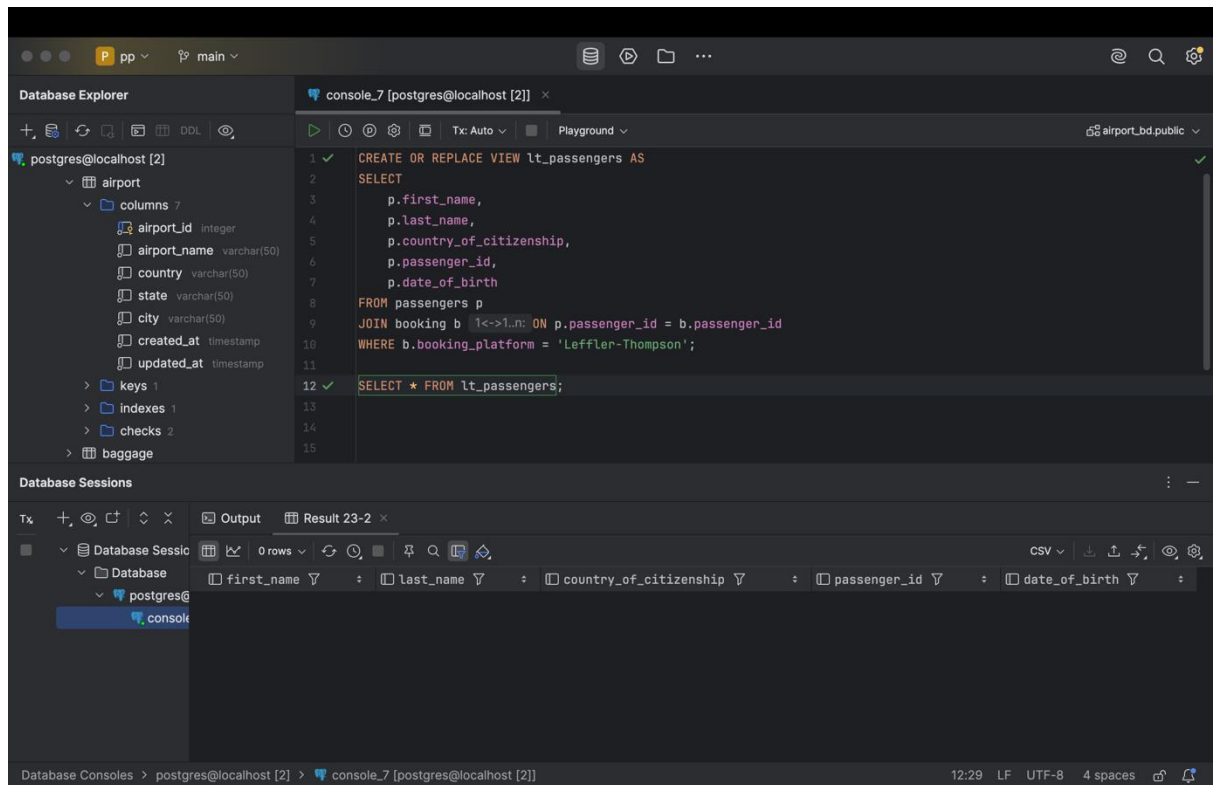
9. Update any of the created views by adding new information in the view table. Show results.

```

CREATE OR REPLACE VIEW lt_passengers AS
SELECT
    p.first_name,
    p.last_name,
    p.country_of_citizenship,
    p.passenger_id,
    p.date_of_birth
FROM passengers p
JOIN booking b ON p.passenger_id = b.passenger_id
WHERE b.booking_platform = 'Leffler-Thompson';

SELECT * FROM lt_passengers;

```



## 10. Drop all existing views.

```
DROP VIEW IF EXISTS
    flights_dep,
    book_n_w,
    top5,
    flights_by_airline,
    flights_air_n_w,
    public.flights_24h_24h,
    lt_passengers,
    top10_c
CASCADE;
```

ppmain

Database Explorer

postgres@localhost [2]

- airport
  - columns 7
    - airport\_id integer
    - airport\_name varchar(50)
    - country varchar(50)
    - state varchar(50)
    - city varchar(50)
    - created\_at timestamp
    - updated\_at timestamp

console\_7 [postgres@localhost [2]]

```
1 DROP VIEW IF EXISTS
2   flights_dep,
3   book_n_w,
4   top5,
5   flights_by_airline,
6   flights_air_n_w,
7   public.flights_24h_24h,
8   lt_passengers,
9   top10_c
10 CASCADE;
```

Database Sessions

Tx

Database Sessio

- Database
  - postgres@
    - console

Output

Result 23-2

```
[2025-11-18 22:31:42] 0 rows retrieved in 341 ms (execution: 4 ms, fetching: 337 ms)
[2025-11-18 22:35:05] airport_bd.public> DROP VIEW IF EXISTS
    flights_dep,
    book_n_w,
    top5,
    flights_by_airline,
    flights_air_n_w,
    public.flights_24h_24h,
    lt_passengers,
    top10_c
    CASCADE
view "flights_24h_24h" does not exist, skipping
[2025-11-18 22:35:05] completed in 17 ms
```

Database Consoles > postgres@localhost [2] > console\_7 [postgres@localhost [2]]

10:9 L F UTF-8 4 spaces