

Laboratory work 8

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

The screenshot shows a PostgreSQL database interface with a 'Database Explorer' sidebar containing a 'postgres@localhost [2]' connection with three tables: 'baggage_check', 'boarding_pass', and 'booking'. A 'Database Sessions' tab is open, showing a 'PostgreSQL' session with a query window titled 'console_7 [postgres@localhost [2]]'. The query is:

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

Below the query window is a table viewer showing the results of the query. The table has columns: flight_id, sch_departure_time, sch_arrival_time, departing_airport_id, and arriving_airport_id. The data consists of 16 rows, each representing a flight scheduled to depart on 2025-09-03 at various times, with destination airports 1 through 15 and arrival airports 1 through 5.

```
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;
```

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the schema structure with tables like `baggage_check`, `boarding_pass`, `booking`, `booking_flight`, `flights`, and `columns`.
- Playground:** A code editor window titled `console_7 [postgres@localhost [2]]` containing the following SQL code:


```

1 CREATE OR REPLACE VIEW book_n_w AS SELECT b.* FROM booking b
2 JOIN flights f ON b.flight_id=f.flight_id
3 WHERE DATE(f.sch_departure_time) BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7 days';
4
5 ✓ SELECT * FROM book_n_w;
6
7
8
      
```
- Database Sessions:** Shows a session named `console` connected to the `book_n_w` view.
- Output:** A results grid showing the output of the query, which is currently empty.
- Bottom Status Bar:** Displays the time as 5:24, file type as LF, encoding as UTF-8, and code style as 4 spaces.

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 PostgreSQL database interface with the following details:

- Database Explorer:** Shows the schema structure with tables like `baggage_check`, `boarding_pass`, `booking`, `booking_flight`, `flights`, and `columns`.
- Playground:** A code editor window titled `console_7 [postgres@localhost [2]]` containing the following 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;
11
      
```
- Database Sessions:** Shows a session named `console` connected to the `top5` view.
- Output:** A results grid showing the output of the query, which is currently empty.
- Bottom Status Bar:** Displays the time as 11:1, file type as LF, encoding as UTF-8, and code style as 4 spaces.

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 interface with two panes. The left pane, 'Database Explorer', shows a tree structure of databases, schemas, and tables, including 'flights' with columns like 'flight_id', 'sch_departure_time', and 'sch_arrival_time'. The right pane, 'console_7 [postgres@localhost [2]]', contains the SQL code for creating the view and executing it. Below the code, a table displays 10 rows of flight data from the 'flights' table, filtered by airline_id = 1. The table has columns: flight_id, sch_departure_time, sch_arrival_time, departing_airport_id, arriving_airport_id. The data includes various flight IDs and departure/arrival times for different days.

flight_id	sch_departure_time	sch_arrival_time	departing_airport_id	arriving_airport_id
1	2025-09-01 11:30:00.000000	2025-09-01 14:00:00.000000	3	
2	2025-09-03 06:00:00.000000	2025-09-03 08:30:00.000000	1	
3	2025-09-03 20:15:00.000000	2025-09-03 22:45:00.000000	16	
4	2025-09-04 18:45:00.000000	2025-09-04 21:15:00.000000	17	
5	2025-09-05 17:15:00.000000	2025-09-05 19:45:00.000000	17	
6	2025-09-06 15:45:00.000000	2025-09-06 18:15:00.000000	17	
7	2025-09-07 14:15:00.000000	2025-09-07 16:45:00.000000	17	
8	2025-09-08 12:45:00.000000	2025-09-08 15:15:00.000000	17	
9	2025-09-09 11:15:00.000000	2025-09-09 13:45:00.000000	17	
10	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;
```

The screenshot shows a PostgreSQL database interface with a dark theme. In the top right, there are icons for search, refresh, and settings. Below them is a toolbar with buttons for file operations like new, open, save, and close. A dropdown menu is open, showing options like 'File', 'Edit', 'View', 'Tools', 'Help', and 'PostgreSQL'. On the left, the 'Database Explorer' sidebar lists databases, schemas, and tables. The 'flights' table under 'flights' schema is expanded, showing columns: flight_id (integer). The main area is a 'Playground' tab where SQL queries are run. The first query is:

```
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';
```

The second query is:

```
SELECT * FROM flights_air_n_w;
```

Below the playground, the 'Database Sessions' panel shows a single session named 'console_7'. The bottom status bar indicates the time is 5:31, and the encoding is UTF-8.

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;
```

The screenshot shows a PostgreSQL database interface with a dark theme, similar to the previous one. The 'Database Explorer' sidebar is visible on the left. The main area is a 'Playground' tab where SQL queries are run. The first query is:

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

The second query is:

```
SELECT * FROM flights_24h;
```

Below the playground, the 'Database Sessions' panel shows a single session named 'console_7'. The bottom status bar indicates the time is 6:1, and the encoding is UTF-8.

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;
```

The screenshot shows a PostgreSQL database interface. In the top right, there's a tab labeled "console_7 [postgres@localhost [2]]". Below it, the "Database Explorer" pane shows a table named "passenger" with various columns. The "Database Sessions" pane contains the SQL code for creating the view:

```
1 ✓ CREATE OR REPLACE VIEW lt_passengers AS
2   SELECT p.first_name,
3         p.last_name,
4         p.country_of_citizenship
5   FROM passengers p
6   JOIN booking b 1<->1..n: ON p.passenger_id = b.passenger_id
7   WHERE b.booking_platform = 'Leffler-Thompson';
8
9 ✓ SELECT * FROM lt_passengers;
10
11 |
```

Below the sessions pane, the "Output" tab shows the results of the query, which are currently empty ("0 rows"). The bottom status bar indicates the session details: "Database Consoles > postgres@localhost [2] > console_7 [postgres@localhost [2]]".

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 interface. In the top right, there's a search bar and a gear icon. Below it, a tab bar has 'main' selected. On the left, a 'Database Explorer' sidebar shows a tree structure for 'postgres@localhost [2]' with a 'airport' node expanded, revealing columns like 'airport_id', 'airport_name', 'country', 'state', 'city', 'created_at', and 'updated_at'. The main area is a 'console_7 [postgres@localhost [2]]' tab containing a SQL query:

```

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 1..n->1: 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

```

Below the query, an 'Output' tab displays the results of the 'SELECT *' statement:

	country	total_flight
1	USA	67
2	Germany	28
3	Australia	11
4	Japan	10
5	China	10
6	Canada	10
7	South Korea	10
8	UAE	9
9	Brazil	9
10	France	9

At the bottom, the status bar shows 'Database Consoles > postgres@localhost [2] > console_7 [postgres@localhost [2]]' and '9:1 LF UTF-8 4 spaces'.

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;

```

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the schema of the `airport` table, which includes columns: `airport_id`, `airport_name`, `country`, `state`, `city`, `created_at`, and `updated_at`.
- Console:** Displays the creation of a view named `lt_passengers` and its execution.
- Database Sessions:** Shows the current session is connected to the `postgres` database.
- Output:** Shows the results of the `SELECT * FROM lt_passengers;` query.

```
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;
```

The screenshot shows a PostgreSQL database management interface with the following details:

- Database Explorer:** Shows the `airport` schema with its columns: `airport_id`, `airport_name`, `country`, `state`, `city`, `created_at`, and `updated_at`.
- Console:** A query editor window titled `console_7 [postgres@localhost [2]]` containing the following SQL code:

```
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:** Shows a session for `postgres@localhost` with the command history:

```
[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
```
- Bottom Status Bar:** Displays the time as 10:9, file format as LF, encoding as UTF-8, and code style as 4 spaces.