

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

- Database Explorer:** The left sidebar displays the database structure under the `postgres` schema. It includes tables like `airline`, `airport`, `flights`, `airline`, and `passenger`, and views like `flights_by_date`, `bookings_next_week`, etc.
- Query Editor:** The main area shows a query being run against the `public` schema. The query is:

```
1 ✓ DROP VIEW IF EXISTS
2   flights_by_date,
3   bookings_next_week,
4   top5_routes,
5   flights_by_airline,
6   delayed_24h,
7   leffler_passengers,
8   top10_countries
9   CASCADE;
```
- Services:** The bottom section shows the status of various services. The `console` service is active and has completed a task in 23 ms.

```
CREATE OR REPLACE VIEW flights_by_date AS
SELECT f.*,
       (f.actual_arrival - f.actual_departure) AS flight_duration
FROM flights f
WHERE DATE(f.scheduled_departure) = CURRENT_DATE;
```

Services

TX	airline	435
[]	flights	48
[]	console	
[]	passenger	

[2025-11-26 00:44:35] completed in 6 ms

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

- Database Explorer:** Shows the `postgres` database with 11 tables and 0 views.
- Console Tab:** Active tab, showing a query to create a view named `top10_countries`. The query uses a self-join on the `flights` table to count visits per country and then joins with the `airport` table to get the country names. The results are grouped by country, ordered by visits in descending order, and limited to 10 rows.
- Services Tab:** Shows the `console` service running, completed in 6 ms on 2025-11-26 at 00:44:35.

```
CREATE VIEW top10_countries AS
SELECT a.country,
       COUNT(*) AS visits
  FROM flights f
 JOIN airport a ON f.arrival_airport_id = a.airport_id
 GROUP BY a.country
 ORDER BY visits DESC
LIMIT 10;
```

The screenshot shows a PostgreSQL database interface in a code editor. On the left, the Database Explorer sidebar lists the 'public' schema of the 'postgres' database, containing tables like 'airline', 'airport', 'baggage', etc., and views like 'leffler_passengers'. The main console window displays the following SQL code:

```
CREATE VIEW leffler_passengers AS
SELECT p.passenger_id,
       p.first_name || ' ' || p.last_name AS full_name,
       p.country_of_citizenship
FROM passengers p
JOIN booking b ON p.passenger_id = b.passenger_id
WHERE p.name = 'Leffler-Thompson';
```

The screenshot shows a PostgreSQL database management interface. On the left, the Database Explorer sidebar lists the 'postgres' database schema, including tables like 'airline', 'airport', 'baggage', etc., and views. The 'flights' table is currently selected. The main area is a 'console' tab where a SQL command has been run:

```
1 ✓ CREATE VIEW delayed_24h AS
2   SELECT f.*
3     FROM flights f
4    WHERE (f.actual_departure - f.scheduled_departure) > INTERVAL '24 hours';
5
```

The command was successful, indicated by the green checkmark and the number '1'. Below the console, the Services panel shows a transaction (Tx) and a history of recent queries:

```
SELECT f.*
  FROM flights f
 WHERE f.airline_id = 1
      AND f.scheduled_departure BETWEEN CURRENT_DATE
                                    AND CURRENT_DATE + INTERVAL '7 days'

[2025-11-26 00:36:34] completed in 4 ms
[2025-11-26 00:37:46] postgres.public> CREATE VIEW delayed_24h AS
          SELECT f.*
          FROM flights f
         WHERE (f.actual_departure - f.scheduled_departure) > INTERVAL '24 hours'

[2025-11-26 00:37:46] completed in 3 ms
```

The screenshot shows the DataGrip IDE interface. The top bar displays the project name "llab-2" and "Version control". The main area has tabs for "Database Explorer", "console", and "Playground". The "Database Explorer" tab is selected, showing a tree view of a database named "postgres" with various tables like "airline", "airport", "baggage", etc. The "console" tab contains a code editor with the following SQL command:

```
1 ✓ CREATE OR REPLACE VIEW flights_by_airline AS
2   SELECT f.*
3     FROM flights f
4    WHERE f.airline_id = 1
5      AND f.scheduled_departure BETWEEN CURRENT_DATE
6                                AND CURRENT_DATE + INTERVAL '7 days';
7
```

The "Services" panel on the left shows a transaction (Tx) and a database connection for "postgres" which completed in 4 ms. The bottom status bar shows the path "Database Consoles > postgres > console", and the system tray indicates it's 12:36 AM on 11/26/2025.

The screenshot shows a PostgreSQL database management interface. In the top navigation bar, there are tabs for 'llab-2' and 'Version control'. Below the tabs, the 'Database Explorer' sidebar is open, showing a tree structure of database objects under the 'postgres' schema, including 'airline', 'airport', 'baggage', 'baggage_check', 'boarding_pass', 'booking', 'booking_flight', 'flights' (which is selected), 'passengers', 'security_check', 'views', 'Database Objects', and 'Server Objects'. The main area is a 'console' tab where a SQL command is being typed:

```
CREATE VIEW flights_by_airline AS
SELECT f.*
FROM flights f
WHERE f.airline_id = 1;
```

Below the console, the 'Services' section shows a transaction (Tx) and a database connection to 'postgres'. The transaction history shows the execution of the CREATE VIEW command:

```
[2025-11-26 00:35:16] completed in 6 ms
[2025-11-26 00:36:15] postgres.public> CREATE VIEW flights_by_airline AS
    SELECT f.*
    FROM flights f
    WHERE f.airline_id = 1;
[2025-11-26 00:36:15] completed in 4 ms
```

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

- Database Explorer:** On the left, under the `postgres` database, the `flights` table is selected.
- Console:** The main area displays a SQL query for creating a view named `top5_routes`. The query selects `departing_gate`, `arriving_gate`, and the count of `booking_id` from the `flights` table, joined with the `booking_flight` table, grouped by flight ID, ordered by total bookings in descending order, and limited to 5 rows.
- Services:** On the bottom left, a transaction bar shows the query was completed in 9 ms. The transaction log shows the command and its execution details.
- Status Bar:** At the bottom right, it indicates 3.23 MB of memory used, 100% CPU usage, and 4 spaces of memory.

```
CREATE VIEW top5_routes AS
SELECT f.departing_gate,
       f.arriving_gate,
       COUNT(b.booking_id) AS total_bookings
FROM flights f
JOIN booking_flight b ON f.flight_id = b.Flight_id
GROUP BY f.flight_id
ORDER BY total_bookings DESC
LIMIT 5;
```

[2025-11-26 00:23:33] completed in 9 ms
[2025-11-26 00:35:16] postgres.public> CREATE VIEW top5_routes AS
[2025-11-26 00:35:16] postgres.public> SELECT f.departing_gate,
[2025-11-26 00:35:16] postgres.public> f.arriving_gate,
[2025-11-26 00:35:16] postgres.public> COUNT(b.booking_id) AS total_bookings
[2025-11-26 00:35:16] postgres.public> FROM flights f
[2025-11-26 00:35:16] postgres.public> JOIN booking_flight b ON f.flight_id = b.Flight_id
[2025-11-26 00:35:16] postgres.public> GROUP BY f.flight_id
[2025-11-26 00:35:16] postgres.public> ORDER BY total_bookings DESC
[2025-11-26 00:35:16] postgres.public> LIMIT 5
[2025-11-26 00:35:16] completed in 6 ms

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

Database Explorer pane (left):

- Selected database: `postgres`
- Tables listed under `public`: `airline`, `airport`, `baggage`, `baggage_check`, `boarding_pass`, `booking`, `booking_flight` (highlighted), `flights`, `passenger`, `security_check`.

console pane (right):

```
CREATE VIEW bookings_next_week AS
SELECT b.*, f.scheduled_departure
FROM booking_flight b
JOIN flights f ON b.flight_id = f.flight_id
WHERE f.scheduled_departure BETWEEN CURRENT_DATE
                                AND CURRENT_DATE + INTERVAL '7 days';
```

Services pane (bottom-left):

- Selected connection: `postgres`
- Execution history:
 - `SELECT * FROM flights WHERE DATE(scheduled_departure) = CURRENT_DATE` completed in 49 ms
 - `[2025-11-26 00:23:33] postgres.public> CREATE VIEW bookings_next_week AS SELECT b.*, f.scheduled_departure FROM booking_flight b JOIN flights f ON b.flight_id = f.flight_id WHERE f.scheduled_departure BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7 days'` completed in 9 ms

System tray and status bar (bottom-right):

- Temperature: 5°C
- Search bar: Помощь
- Date and time: 11/26/2025 12:23 AM
- File status: ENG
- Network: WiFi
- Battery: 99%

The screenshot shows the Visual Studio Code (VS Code) interface with the Database Explorer and a terminal window open.

Database Explorer: On the left, the Database Explorer sidebar shows a tree structure for the `postgres` database. Under the `public` schema, there is a `tables` node containing ten tables: `airline`, `airport`, `baggage`, `baggage_check`, `boarding_pass`, `booking`, `booking_flight`, `flights`, `passengers`, and `security_check`.

Terminal: The main area is a terminal window titled `console`. It displays the following SQL command and its execution results:

```
CREATE VIEW flights_by_date AS
SELECT *
FROM flights
WHERE DATE(scheduled_departure) = CURRENT_DATE;
```

Output from the terminal:

```
[2025-11-26 00:22:50] Connected
[2025-11-26 00:22:50] postgres> CREATE VIEW flights_by_date AS
[2025-11-26 00:22:50] postgres>     SELECT *
[2025-11-26 00:22:50] postgres>     FROM flights
[2025-11-26 00:22:50] postgres>     WHERE DATE(scheduled_departure) = CURRENT_DATE
[2025-11-26 00:22:50] completed in 49 ms
```