

1_

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

- Database Explorer:** Shows the schema for the 'flights' table, which has 14 columns: flight_id, flight_no, scheduled_departure, scheduled_arrival, departure_airport_id, arrival_airport_id, airline_id, status, actual_departure, actual_arrival, created_at, update_at, and two foreign keys.
- Console:** Displays the SQL code for creating a stored procedure named 'Insert_flight'. The procedure uses the PL/pgSQL language and declares a variable 'p_flight_id' of type INT. It begins by selecting the maximum flight ID from the 'flights' table and inserting a new row with the next available ID into the 'flights' table. The new row includes the flight number, airline ID, departure and arrival airport IDs, status, and scheduled arrival and departure dates.
- Services:** Shows a transaction (Tx) and a connection to 'postgres@localhost'.
- Output:** Shows the execution results of the 'Insert_flight' procedure, indicating it completed in 18 ms.

2_

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

- Database Explorer:** Shows the schema for the 'flights' table, identical to the one in the first screenshot.
- Console:** Displays the SQL code for creating a stored procedure named 'update_flight_status'. The procedure uses the PL/pgSQL language and declares variables 'p_flight_id' (INT) and 'p_new_status' (VARCHAR). It begins by updating the 'status' column in the 'flights' table to 'p_new_status' where the 'flight_id' matches 'p_flight_id'.
- Services:** Shows a transaction (Tx) and a connection to 'postgres@localhost'.
- Output:** Shows the execution results of the 'update_flight_status' procedure, indicating it completed in 3 ms.

3_

The screenshot shows a PostgreSQL database interface. On the left, the Database Explorer pane displays the schema structure of the 'flights' table, including columns like flight_id, flight_no, scheduled_departure, and arrival_airport_id. The right pane is a code editor titled 'console' where a function named 'get_flights_from_airport' is being created. The function takes an integer parameter 'pairport_id' and returns a table with columns flight_id, flight_no, and scheduled_departure. The query uses a WHERE clause to filter flights where the departure_airport_id matches the input parameter.

```
CREATE OR REPLACE FUNCTION get_flights_from_airport(pairport_id INT)
RETURNS TABLE (
    flight_id INT,
    flight_no VARCHAR,
    dep DATE
)
LANGUAGE sql
AS $$
SELECT flight_id, flight_no, scheduled_departure
FROM flights
WHERE departure_airport_id = pairport_id;
$$;
```

4_

The screenshot shows a PostgreSQL database interface. The Database Explorer pane shows the 'flights' table structure. The right pane displays the creation of a function 'avg_arrival_delay' that takes an integer parameter 'p_airport_id'. The function calculates the average arrival delay for flights originating from the specified airport. It uses a WHERE clause to filter flights where the arrival_airport_id matches the input parameter and then calculates the average difference between actual arrival and scheduled arrival times.

```
CREATE OR REPLACE FUNCTION avg_arrival_delay(p_airport_id INT)
RETURNS NUMERIC
LANGUAGE plpgsql
AS $$
DECLARE
    result NUMERIC;
BEGIN
    SELECT AVG(EXTRACT(EPOCH FROM f.actual_arrival - f.scheduled_arrival) / 60)
    INTO result
    FROM flights f
    WHERE f.arrival_airport_id = p_airport_id
        AND f.actual_arrival IS NOT NULL
        AND f.actual_arrival > f.scheduled_arrival;
    RETURN COALESCE(result, 0);
END;
$$;
```

```
result NUMERIC;
BEGIN
    SELECT AVG(EXTRACT(EPOCH FROM f.actual_arrival - f.scheduled_arrival) / 60)
    INTO result
    FROM flights f
    WHERE f.arrival_airport_id = p_airport_id
        AND f.actual_arrival IS NOT NULL
        AND f.actual_arrival > f.scheduled_arrival;
    RETURN COALESCE(result, 0);
END;
$$;
```

5_

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

- Database Explorer:** Shows the schema structure of the `dbwork` database, including tables like `airline`, `airport`, `baggage`, `baggage_check`, `boarding_pass`, `booking`, `booking_flight`, `flights`, `passenger`, and `security_check`.
- Services:** Shows a single active connection to `postgres@localhost`.
- Console:** Displays the creation of a stored procedure named `show_passengers_by_flight`. The code uses PL/pgSQL and includes joins across multiple tables to retrieve passenger information based on a flight ID.
- Status Bar:** Shows the command was completed in 3 ms at [2025-12-03 10:08:28].

6_

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

- Database Explorer:** Shows the schema structure of the `dbwork` database, including tables like `airline`, `airport`, `baggage`, `baggage_check`, `boarding_pass`, `booking`, `booking_flight`, `flights`, `passenger`, and `security_check`.
- Services:** Shows a single active connection to `postgres@localhost`.
- Console:** Displays the creation of a function named `get_top_passenger`. The code uses PL/pgSQL and includes joins to count the number of bookings per passenger and select the top one.
- Status Bar:** Shows the command was completed in 3 ms at [2025-12-03 10:17:05].

7_

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

- Database Explorer:** Shows the schema structure of the `dbwork` database, including tables like `airline`, `airport`, `baggage`, `baggage_check`, `boarding_pass`, `booking`, `booking_flight`, `flights`, `passengers`, and `security_check`.
- console:** The current tab is the `flights` table.
- Code Editor:** Displays the SQL code for creating a stored procedure named `delayed_24h()`. The code uses a cursor to iterate through delayed flights and raise a notice for each.
- Output:** Shows the executed SQL code and the message "[2025-12-05 10:53:56] completed in 3 ms".
- Services:** Shows the database connection status.
- Bottom Status:** Shows the date (2025-12-05), time (10:53:56), and file encoding (UTF-8).

8_

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

- Database Explorer:** Shows the schema structure of the `postgres` database, including tables like `airline`, `airport`, `baggage`, `baggage_check`, `boarding_pass`, `booking`, `booking_flight`, and `flights`.
- console:** The current tab is the `flights` table.
- Code Editor:** Displays the SQL code for creating a function named `count_flights_in_airlines(fairline_id INT)`. The function counts flights for a given airline ID.
- Output:** Shows the executed SQL code and the message "[2025-12-05 11:05:59] completed in 2 ms".
- Services:** Shows the database connection status.
- Bottom Status:** Shows the date (2025-12-05), time (11:05:59), and file encoding (UTF-8).

9_

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

- Database Explorer:** Shows the schema structure. A table named "flights" is selected, displaying its columns: flight_id, flight_no, scheduled_departure, scheduled_arrival, departure_airport_id, arrival_airport_id, departing_gate, arriving_gate, airline_id, status, actual_departure, actual_arrival, created_at, and update_at.
- Console:** A stored procedure named "avg_ticket_price" is being created or replaced. The code is as follows:

```
CREATE OR REPLACE PROCEDURE avg_ticket_price(p_flight_id INT, OUT ans NUMERIC)
LANGUAGE plpgsql
AS $$
DECLARE ans NUMERIC;
BEGIN
    SELECT AVG(b.price) INTO ans
    FROM booking b
    JOIN booking_flight bf ON b.booking_id = bf.booking_id
    WHERE bf.flight_id = p_flight_id;
END;
$$;
```

The procedure calculates the average price for a given flight ID.

Services: Shows a transaction (Tx) for the "console" service.

Output: The result of running the stored procedure is shown:

```
AS $$
DECLARE ans NUMERIC;
BEGIN
    SELECT AVG(b.price) INTO ans
    FROM booking b
    JOIN booking_flight bf ON b.booking_id = bf.booking_id
    WHERE bf.flight_id = p_flight_id;

END;
$$
```

[2025-12-05 11:14:17] completed in 3 ms

10_

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

- Database Explorer:** Shows the schema structure. A table named "flights" is selected, displaying its columns: flight_id, flight_no, scheduled_departure, scheduled_arrival, departure_airport_id, arrival_airport_id, departing_gate, arriving_gate, airline_id, status, actual_departure, actual_arrival, created_at, and update_at.
- Console:** A stored procedure named "most_expensive_flight" is being created or replaced. The code is as follows:

```
CREATE OR REPLACE PROCEDURE most_expensive_flight(
    OUT oflight_no VARCHAR,
    OUT odepart_airport_id INT,
    OUT oarrive_airport_id INT,
    OUT oprice NUMERIC
)
LANGUAGE plpgsql
AS $$
$$
BEGIN
    SELECT f.flight_no, f.departure_airport_id, f.arrival_airport_id, b.price
    INTO oflight_no, odepart_airport_id, oarrive_airport_id, oprice
    FROM flights f
    JOIN booking_flight bf ON f.flight_id = bf.flight_id
    JOIN booking b ON bf.booking_id = b.booking_id
    ORDER BY b.price DESC
    LIMIT 1;
END;
$$;
```

The procedure selects the flight with the highest price.

Services: Shows a transaction (Tx) for the "console" service.

Output: The result of running the stored procedure is shown:

airline_id	count
5	10
6	25
7	22
8	25
9	22
10	20
11	19
12	22
13	25

50 rows