

1_

The screenshot shows a database console interface with a dark theme. On the left, the 'Database Explorer' pane shows a tree structure for 'postgres@localhost' with tables like 'airline', 'airport', 'baggage', 'boarding_pass', 'booking', 'booking_flight', and 'flights'. The 'flights' table is selected, showing its columns: flight_id (integer), flight_no (varchar(50)), scheduled_departure (date), scheduled_arrival (date), departure_airport_id (integer), arrival_airport_id (integer), departing_gate (varchar(50)), arriving_gate (varchar(50)), airline_id (integer), status (varchar(50)), actual_departure (date), actual_arrival (date), created_at (date), and update_at (date). The main console area displays a SQL script for creating a procedure named 'insert_flight'. The script includes parameter declarations for p_flight_number, p_airline_id, p_departure_airport_id, p_arrival_airport_id, p_status, p_scheduled_arrival, p_scheduled_departure, p_departing_gate, and p_arriving_gate. It then uses a SELECT statement to find the next flight_id and an INSERT statement to add a new flight record. The script is executed, and the 'Services' pane at the bottom shows the completion message: '[2025-12-03 09:16:39] completed in 18 ms'.

```
1 CREATE OR REPLACE PROCEDURE insert_flight(  
2   p_flight_number VARCHAR,  
3   p_airline_id INT,  
4   p_departure_airport_id INT,  
5   p_arrival_airport_id INT,  
6   p_status VARCHAR,  
7   p_scheduled_arrival DATE,  
8   p_scheduled_departure DATE,  
9   p_departing_gate VARCHAR,  
10  p_arriving_gate VARCHAR  
11 )  
12 LANGUAGE plpgsql  
13 AS  
14 $$  
15 DECLARE p_flight_id INT;  
16 BEGIN  
17   SELECT MAX(flight_id) + 1 INTO p_flight_id FROM flights;  
18   INSERT INTO flights (flight_id, flight_no, airline_id, departure_airport_id, arrival_airport_id, status, scheduled_arrival, scheduled_departure, departing_gate, arriving_gate)  
19   VALUES (p_flight_id, p_flight_number, p_airline_id, p_departure_airport_id, p_arrival_airport_id, p_status, p_scheduled_arrival, p_scheduled_departure, p_departing_gate, p_arriving_gate);  
20 END;  
21 $$
```

Database Consoles > postgres@localhost > console

2_

The screenshot shows the same database console interface as above. The main console area displays a SQL script for creating a procedure named 'update_flight_status'. The script takes p_flight_id (INT) and p_new_status (VARCHAR) as parameters and uses an UPDATE statement to change the status of the flight in the 'flights' table. The script is executed, and the 'Services' pane at the bottom shows the completion message: '[2025-12-03 09:22:07] completed in 3 ms'.

```
1 CREATE OR REPLACE PROCEDURE update_flight_status(  
2   p_flight_id INT,  
3   p_new_status VARCHAR  
4 )  
5 LANGUAGE plpgsql  
6 AS  
7 $$  
8 BEGIN  
9   UPDATE flights SET status = p_new_status WHERE flight_id = p_flight_id;  
10 END;  
11 $$
```

Database Consoles > postgres@localhost > console

3_

The screenshot shows a database console interface with a dark theme. On the left, the 'Database Explorer' pane shows a PostgreSQL database with a 'flights' table. The 'flights' table structure is visible, including columns like flight_id, flight_no, scheduled_departure, and departure_airport_id. The main console area displays the following SQL code:

```

1 CREATE OR REPLACE FUNCTION get_flights_from_airport(pairport_id INT)
2 RETURNS TABLE (
3     flight_id INT,
4     flight_no VARCHAR,
5     dep DATE
6 )
7 LANGUAGE sql
8 AS $$
9     SELECT flight_id, flight_no, scheduled_departure
10    FROM flights
11   WHERE departure_airport_id = pairport_id;
12 $$;
13

```

Below the code, the execution results are shown, indicating that the function was created successfully. The status bar at the bottom shows 'Database Consoles > postgres@localhost > console' and '13:1 CRLF UTF-8 4 spaces'.

4_

The screenshot shows a database console interface with a dark theme. On the left, the 'Database Explorer' pane shows a PostgreSQL database with a 'flights' table. The 'flights' table structure is visible, including columns like flight_id, flight_no, scheduled_departure, and departure_airport_id. The main console area displays the following SQL code:

```

1 CREATE OR REPLACE FUNCTION avg_arrival_delay(p.airport_id INT)
2 RETURNS NUMERIC
3 LANGUAGE plpgsql
4 AS
5 $$
6 DECLARE
7     result NUMERIC;
8 BEGIN
9     SELECT AVG(EXTRACT(EPOCH FROM f.actual_arrival - f.scheduled_arrival) / 60)
10    INTO result
11   FROM flights f
12  WHERE f.arrival_airport_id = p.airport_id
13     AND f.actual_arrival IS NOT NULL
14     AND f.actual_arrival > f.scheduled_arrival;
15     RETURN COALESCE(result, 0);
16 END;
17 $$;

```

Below the code, the execution results are shown, indicating that the function was created successfully. The status bar at the bottom shows 'Database Consoles > postgres@localhost > console' and '17:4 CRLF UTF-8 4 spaces'.

5_

The screenshot shows a PostgreSQL database console with the following SQL code executed:

```

1 CREATE OR REPLACE PROCEDURE show_passengers_by_flight(p_flight_id INT)
2 LANGUAGE plpgsql
3 AS $$
4 DECLARE
5     r RECORD;
6 BEGIN
7     FOR r IN
8     SELECT p.passenger_id, p.first_name, p.last_name, bp.seat, flight_id
9     FROM booking_flight bf
10    JOIN booking b ON bf.booking_id = b.booking_id
11    JOIN passengers p ON b.passenger_id = p.passenger_id
12    JOIN boarding_pass bp ON b.booking_id = bp.booking_id
13    WHERE bf.flight_id = p_flight_id
14    ORDER BY p.last_name, p.first_name
15    LOOP
16         RAISE NOTICE '% | % | % | % | %',
17             r.passenger_id, r.first_name, r.last_name, r.seat, r.flight_id;
18     END LOOP;
19 END;
20 $$;
21

```

The console output shows the procedure completed successfully in 3 ms.

6_

The screenshot shows a PostgreSQL database console with the following SQL code executed:

```

1 CREATE OR REPLACE FUNCTION get_top_passenger(OUT a INT)
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     SELECT p.passenger_id, COUNT(*)::BIGINT
6     INTO a
7     FROM passengers p
8     JOIN booking b ON p.passenger_id = b.passenger_id
9     JOIN booking_flight bf ON b.booking_id = bf.booking_id
10    GROUP BY p.passenger_id
11    ORDER BY COUNT(*) DESC
12    LIMIT 1;
13 END;
14 $$;
15

```

The console output shows the function completed successfully in 3 ms.

7_

Database Explorer

- postgres@localhost
 - pg_catalog
 - public
 - tables 10
 - airline
 - airport
 - baggage
 - baggage_check
 - boarding_pass
 - booking
 - booking_flight
 - flights
 - passengers
 - security_check
 - columns 5
 - security_check_id integer
 - check_result varchar(50)
 - created_at date
 - update_at date
 - passenger_id integer
 - keys 1
 - foreign keys 1
 - indexes 1
 - views 1
 - routes 17

```

1 CREATE OR REPLACE PROCEDURE delayed_24h()
2 LANGUAGE plpgsql
3 AS $$
4 DECLARE
5     RECORD;
6 BEGIN
7     FOR r IN
8         SELECT flight_id, flight_no, actual_arrival, scheduled_arrival
9         FROM flights
10        WHERE EXTRACT(EPOCH FROM actual_arrival - scheduled_arrival) > 86400
11    LOOP
12        RAISE NOTICE '% | % | % | %', r.flight_id, r.flight_no, r.actual_arrival, r.scheduled_arrival;
13    END LOOP;
14 END;
15 $$;
16

```

Services

Database

- postgres@localhost
 - console 23 ms
 - flights 489 ms

Output

```

BEGIN
  FOR r IN
    SELECT flight_id, flight_no, actual_arrival, scheduled_arrival
    FROM flights
    WHERE EXTRACT(EPOCH FROM actual_arrival - scheduled_arrival) > 86400
  LOOP
    RAISE NOTICE '% | % | % | %', r.flight_id, r.flight_no, r.actual_arrival, r.scheduled_arrival;
  END LOOP;
END;
$$
[2025-12-03 10:53:36] completed in 3 ms

```

Database Consoles > postgres@localhost > console

16:1 CRLF UTF-8 4 spaces

8_

Database Explorer

- postgres@localhost
 - pg_catalog
 - public
 - tables 10
 - airline
 - airport
 - baggage
 - baggage_check
 - boarding_pass
 - booking
 - booking_flight
 - flights
 - passengers
 - security_check
 - columns 5
 - security_check_id integer
 - check_result varchar(50)
 - created_at date
 - update_at date
 - passenger_id integer
 - keys 1
 - foreign keys 1
 - indexes 1
 - views 1
 - routes 17

```

1 CREATE OR REPLACE FUNCTION count_flights_in_airlines(airline_id INT)
2 RETURNS INT
3 LANGUAGE plpgsql
4 AS
5 $$
6 DECLARE
7     counter INT;
8 BEGIN
9     SELECT airline.airline_id, COUNT(*)
10    INTO counter
11    FROM flights
12   RIGHT JOIN airline ON airline.airline_id = flights.airline_id
13  GROUP BY airline.airline_id
14  ORDER BY airline.airline_id;
15 END;
16 $$;
17

```

Services

Database

- postgres@localhost
 - console 19 ms
 - flights 489 ms

Output

```

counter INT;
BEGIN
  SELECT airline.airline_id, COUNT(*)
  INTO counter
  FROM flights
  RIGHT JOIN airline ON airline.airline_id = flights.airline_id
  GROUP BY airline.airline_id
  ORDER BY airline.airline_id;
END;
$$
[2025-12-03 11:05:59] completed in 2 ms

```

Database Consoles > postgres@localhost > console

17:1 CRLF UTF-8 4 spaces

9_

The screenshot shows a PostgreSQL console window with the following SQL code:

```

1 CREATE OR REPLACE PROCEDURE avg_ticket_price(p_flight_id INT, OUT ans NUMERIC)
2 LANGUAGE plpgsql
3 AS $$
4 DECLARE ans NUMERIC;
5 BEGIN
6     SELECT AVG(b.price) INTO ans
7     FROM booking b
8     JOIN booking_flight bf ON b.booking_id = bf.booking_id
9     WHERE bf.flight_id = p_flight_id;
10
11 END;
12 $$;
13

```

The console output shows the execution of the procedure for flight_id 19:

```

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;
    $$
[2025-12-03 11:14:17] completed in 3 ms

```

The Database Explorer on the left shows the schema for the 'flights' table, including columns like 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.

10_

The screenshot shows a PostgreSQL console window with the following SQL code:

```

1 CREATE OR REPLACE PROCEDURE most_expensive_flight(
2     OUT oflight_no VARCHAR,
3     OUT odeparture_airport_id INT,
4     OUT oarrival_airport_id INT,
5     OUT oprice NUMERIC
6 )
7 LANGUAGE plpgsql
8 AS
9 $$
10 BEGIN
11     SELECT f.flight_no, f.departure_airport_id, f.arrival_airport_id, b.price
12     INTO oflight_no, odeparture_airport_id, oarrival_airport_id, oprice
13     FROM flights f
14     JOIN booking_flight bf ON f.flight_id = bf.flight_id
15     JOIN booking b ON bf.booking_id = b.booking_id
16     ORDER BY b.price DESC
17     LIMIT 1;
18 END;
19 $$;
20

```

The console output shows the execution of the procedure, returning the most expensive flight details:

airline_id	flight_no	count	price
5	5	10	
6	6	23	
7	7	22	
8	8	25	
9	9	22	
10	10	20	
11	11	19	
12	12	22	
13	13	23	

The Database Explorer on the left shows the schema for the 'flights' table, including columns like 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.