

1)

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer pane displays a database schema with various tables like airline, airport, baggage, etc., and their columns. The 'flights' table is selected, and its columns are listed below it. In the center, the main query editor window has a title bar 'postgres/postgres@PostgreSQL 17*'. The 'Query' tab is active, containing the following SQL code:

```
CREATE OR REPLACE PROCEDURE insert_flight(
    p_flight_number VARCHAR,
    p_departure_city VARCHAR,
    p_arrival_city VARCHAR,
    p_departure_time TIMESTAMP,
    p_arrival_time TIMESTAMP
)
LANGUAGE plpgsql
AS $$

BEGIN
    INSERT INTO flights (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, update_at)
    VALUES (p_flight_id, p_flight_number, p_sched_dep, p_departure_arrival, p_dep_airport_id, p_arrival_id, p_departing_gate, p_arriving_gate, p_airline_id, p_status, p_act_dep, p_act_arr, p_created_at, p_update_at);
END;
$$;
```

Below the code, the 'Data Output' tab shows the message 'CREATE PROCEDURE' and 'Query returned successfully in 106 msec.' At the bottom, the status bar indicates 'Total rows: 0' and 'Query complete 00:00:00.106'.

2)

The screenshot shows the pgAdmin 4 interface. The Object Explorer pane displays a different set of objects, including FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables. The 'flights' table is selected, and its columns are listed below it. In the center, the main query editor window has a title bar 'postgres/postgres@PostgreSQL 17*'. The 'Query' tab is active, containing the following SQL code:

```
CREATE OR REPLACE PROCEDURE update_flight_times(
    p_flight_id INT,
    p_act_dep_time TIMESTAMP DEFAULT NULL,
    p_act_arr_time TIMESTAMP DEFAULT NULL
)
LANGUAGE plpgsql
AS $$

BEGIN
    UPDATE flight
    SET
        act_dep_time = COALESCE(p_act_dep_time, act_dep_time),
        act_arr_time = COALESCE(p_act_arr_time, act_arr_time),
        update_at = NOW()
    WHERE flight_id = p_flight_id;
END;
$$;
```

Below the code, the 'Data Output' tab shows the message 'CREATE PROCEDURE' and 'Query returned successfully in 48 msec.' At the bottom, the status bar indicates 'Total rows: 0' and 'Query complete 00:00:00.048'.

3)

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

postgres/postgres@PostgreSQL 17*

```

CREATE OR REPLACE FUNCTION flights1_from_airport(
    p_departing_airport_id INT
)
RETURNS TABLE (
    flight_id INT,
    sch_departure_time TIMESTAMP,
    sch_arrival_time TIMESTAMP,
    departing_airport_id INT,
    arriving_airport_id INT,
    departing_gate TEXT,
    arriving_gate VARCHAR(50),
    airline_id INT,
    act_departure_time TIMESTAMP,
    act_arrival_time TIMESTAMP,
    created_at TIMESTAMP,
    updated_at TIMESTAMP
)
LANGUAGE plpgsql
AS $$

BEGIN
    RETURN QUERY
    SELECT *
    FROM flights
    WHERE departing_airport_id = p_departing_airport_id;
END;
$$;
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 47 msec.

Total rows: Query complete 00:00:00.047

✓ Query return

4)

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

postgres/postgres@PostgreSQL 17*

```

CREATE OR REPLACE FUNCTION avg_delay_for_airport(
    p_airport_id INT
)
RETURNS INTERVAL
LANGUAGE plpgsql
AS $$
DECLARE
    v_avg_delay INTERVAL;
BEGIN
    SELECT AVG(act_arrival_time - sch_arrival_time)
    INTO v_avg_delay
    FROM flights
    WHERE arriving_airport_id = p_airport_id
        AND act_arrival_time IS NOT NULL
        AND sch_arrival_time IS NOT NULL;

    RETURN v_avg_delay;
END;
$$;
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 60 msec.

Total rows: Query complete 00:00:00.060

✓ Query return

5)

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer postgres/postgres@PostgreSQL 17*

Query History Scratch Pad

Query

```
1 CREATE OR REPLACE PROCEDURE list_passengers_for_flight(
2     p_flight_id INT
3 )
4 LANGUAGE plpgsql
5 AS $$
6 BEGIN
7     SELECT p.*
8     FROM passengers p
9     JOIN booking b
10    ON b.passenger_id = p.passenger_id
11    JOIN booking_flight bf
12    ON bf.booking_id = b.booking_id
13    WHERE bf.flight_id = p_flight_id;
14
15 END;
$$;
```

Data Output Messages Notifications

CREATE PROCEDURE

Query returned successfully in 48 msec.

Total rows: Query complete 00:00:00 0.048

✓ Query returned successfully in 48 msec.

6)

The screenshot shows the pgAdmin 4 interface with the following details:

- File Object Tools Edit View Window Help**
- Object Explorer** pane on the left, expanded to show:

 - FTS Dictionaries
 - FTS Parsers
 - FTS Templates
 - Foreign Tables
 - Functions
 - Materialized Views
 - Operators
 - Procedures
 - Sequences
 - Tables (10): airline, airport, baggage, baggage_check, boarding_pass, booking, booking_flight, flights, passengers
 - Columns (10) under passengers: passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number, created_at, update_at
 - Constraints
 - Indexes
 - RLS Policies
 - Rules
 - Triggers
 - security_check
 - Trigger Functions
 - Types
 - Views
 - Subscriptions

- Query Editor** pane in the center:

 - Connection: postgres/postgres@PostgreSQL 17*
 - Toolbar icons: back, forward, search, refresh, etc.
 - Text area:

```
CREATE OR REPLACE PROCEDURE top_passenger_by_flights()
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT
        p.passenger_id,
        p.first_name,
        p.last_name,
        COUNT(bf.flight_id) AS flights_taken
    FROM passengers p
    JOIN booking b
        ON b.passenger_id = p.passenger_id
    JOIN booking_flight bf
        ON bf.booking_id = b.booking_id
    GROUP BY p.passenger_id, p.first_name, p.last_name
    ORDER BY flights_taken DESC
    LIMIT 1;
END;
$$;
```
 - Bottom tabs: Data Output, Messages (selected), Notifications
 - Message area: CREATE PROCEDURE
 - Status bar: Total rows: 0 Query complete 00:00:00.039
 - Bottom right corner: ✓ Query ret...

7)

The screenshot shows the pgAdmin 4 interface. The left sidebar is the Object Explorer, displaying a tree view of database objects. The 'Tables (10)' node is expanded, and the 'flights' table is selected, which has 14 columns listed under it. The main area is a query editor titled 'postgres/postgres@PostgreSQL 17*'. It contains a SQL script to create a procedure named 'flights_delayed_more_than_24h'. The procedure selects flights where the actual departure time is not null, the scheduled departure time is not null, and the difference between them is greater than 24 hours. The 'Messages' tab at the bottom shows the message 'CREATE PROCEDURE' and 'Query returned successfully in 39 msec.'

```
1 CREATE OR REPLACE PROCEDURE flights_delayed_more_than_24h()
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     SELECT *
6     FROM flights
7     WHERE act_departure_time IS NOT NULL
8         AND sch_departure_time IS NOT NULL
9         AND act_departure_time - sch_departure_time > INTERVAL '24 hours';
10 END;
11 $$;
```

Data Output Messages Notifications

CREATE PROCEDURE

Query returned successfully in 39 msec.

8)

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

postgres/postgres@PostgreSQL 17*

Query History

```
1 CREATE OR REPLACE FUNCTION count_flights_for_each_airline()
2   RETURNS TABLE (
3     airline_id INT,
4     flights_count BIGINT
5   )
6 LANGUAGE plpgsql
7 AS $$
```

BEGIN

```
8   RETURN QUERY
9     SELECT airline_id,
10        COUNT(*) AS flights_count
11    FROM flights
12    GROUP BY airline_id;
```

END;

```
13 $$;
```

```
14
15
16
17
18
```

```
19 SELECT * FROM count_flights_for_each_airline();
```

Data Output Messages Notifications

Total rows: Query complete 00:00:00.043

9)

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

```

> FTS Configurations
> FTS Dictionaries
> FTS Parsers
> FTS Templates
> Foreign Tables
> Functions
> Materialized Views
> Operators
> Procedures
> Sequences
Tables (10)
> airline
> airport
> baggage
> baggage_check
> boarding_pass
> booking
> booking_flight
> flights
  > Columns (14)
    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
    update_at
> Constraints
> Indexes
> RLS Policies
> Rules
> Triggers
  > nseanware

```

postgres/postgres@PostgreSQL 17* X

Query Query History

```

1 CREATE OR REPLACE PROCEDURE avg_ticket_price_for_flight(
2   p_flight_id INT
3 )
4 LANGUAGE plpgsql
5 AS $$
6 BEGIN
7   SELECT bf.flight_id,
8         AVG(b.ticket_price) AS avg_ticket_price
9   FROM booking_flight bf
10  JOIN booking b
11    ON b.booking_id = bf.booking_id
12  WHERE bf.flight_id = p_flight_id
13  GROUP BY bf.flight_id;
14
15 END;
$$;

```

Data Output Messages Notifications

CREATE PROCEDURE

Query returned successfully in 41 msec.

Total rows: 0 Query complete 00:00:00.041 ✓ Query return

10)

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

```

> Operators
> Procedures
> Sequences
Tables (10)
> airline
> airport
> baggage
> baggage_check
> boarding_pass
> booking
> booking_flight
> flights
  > Columns (14)
    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
    update_at
  > Constraints
  > Indexes
  > RLS Policies
  > Rules
  > Triggers
  > passengers
  > security_check
  > Trigger Functions
  > Types
  > Views
  > Subscriptions
  > Login/Group Roles
  > Tablespaces

```

postgres/postgres@PostgreSQL 17* X

Query Query History

```

1 CREATE OR REPLACE PROCEDURE most_expensive_flight()
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5   SELECT f.flight_id,
6         f.departing_airport_id,
7         f.arriving_airport_id,
8         MAX(b.ticket_price) AS max_ticket_price
9   FROM flights f
10  JOIN booking_flight bf
11    ON bf.flight_id = f.flight_id
12  JOIN booking b
13    ON b.booking_id = bf.booking_id
14  GROUP BY f.flight_id, f.departing_airport_id, f.arriving_airport_id
15  ORDER BY max_ticket_price DESC
16  LIMIT 1;
17
18 END;
$$;

```

Data Output Messages Notifications

CREATE PROCEDURE

Query returned successfully in 44 msec.

Total rows: 0 Query complete 00:00:00.044