

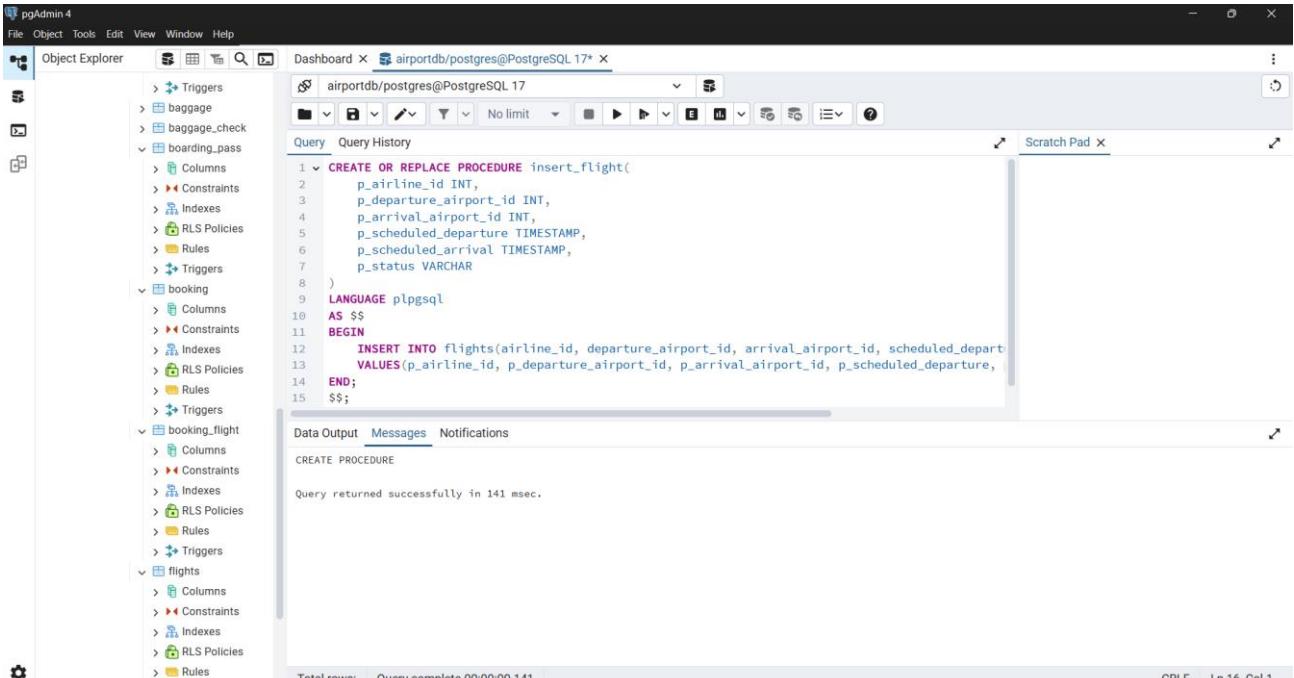
## Laboratory work 10

We continue to work with the database from the previous laboratory works.

Take a full-page screenshot that covers the code and results of each task.

### STORED PROCEDURES and FUNCTION.

1. Create a stored procedure to insert a new flight into the flights table.



The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer tree, which includes tables like 'boarding\_pass', 'booking', and 'flights'. The main window contains a query editor with the following SQL code:

```
CREATE OR REPLACE PROCEDURE insert_flight(
    p_airline_id INT,
    p_departure_airport_id INT,
    p_arrival_airport_id INT,
    p_scheduled_departure TIMESTAMP,
    p_scheduled_arrival TIMESTAMP,
    p_status VARCHAR
)
LANGUAGE plpgsql
AS $$ 
BEGIN
    INSERT INTO flights(airline_id, departure_airport_id, arrival_airport_id, scheduled_departure,
    VALUES(p_airline_id, p_departure_airport_id, p_arrival_airport_id, p_scheduled_departure,
END;
$$;
```

Below the query editor, the Data Output tab shows the result of the CREATE PROCEDURE command:

CREATE PROCEDURE  
Query returned successfully in 141 msec.

Total rows: Query complete 00:00:00.141 CRLF Ln 16, Col 1

2. Create a stored procedure to update the status of a flight.

The screenshot shows the pgAdmin 4 interface with the Object Explorer on the left and a query editor on the right. The query editor contains the following SQL code:

```

CREATE OR REPLACE PROCEDURE update_flight_status(
    p_flight_id INT,
    p_status VARCHAR
)
LANGUAGE plpgsql
AS $$

BEGIN
    UPDATE flights
    SET status = p_status
    WHERE flight_id = p_flight_id;
END;
$$;

```

The Data Output tab shows the message: "Query returned successfully in 128 msec." The status bar at the bottom right indicates "Query complete 00:00:00.128".

3. Create a stored procedure that returns a list of flights departing from a specific airport.

The screenshot shows the pgAdmin 4 interface with the Object Explorer on the left and a query editor on the right. The query editor contains the following SQL code:

```

CREATE OR REPLACE PROCEDURE flights_from_airport(
    p_airport_id INT
)
LANGUAGE plpgsql
AS $$

BEGIN
    SELECT *
    FROM flights
    WHERE departure_airport_id = p_airport_id;
END;
$$;

```

The Data Output tab shows the message: "Query returned successfully in 139 msec." The status bar at the bottom right indicates "Query complete 00:00:00.139".

4. Create a function to calculate the average delay time of flights arriving at a specific airport.

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

- Object Explorer:** Displays the database schema with tables like `boarding_pass`, `booking`, `flight`, and `flights`.
- Dashboard:** Shows the connection to `airportdb/postgres@PostgreSQL 17*`.
- Query Editor:** Contains the following SQL code:
 

```

CREATE OR REPLACE FUNCTION avg_arrival_delay(p_airport_id INT)
RETURNS INTERVAL
LANGUAGE plpgsql
AS $$
DECLARE
    avg_delay INTERVAL;
BEGIN
    SELECT AVG(actual_arrival - scheduled_arrival)
    INTO avg_delay
    FROM flights
    WHERE arrival_airport_id = p_airport_id;
    RETURN avg_delay;
END;
$$;
      
```
- Data Output:** Shows the message "Query returned successfully in 67 msec."
- Messages:** Shows the message "Query returned successfully in 67 msec." in a green box.
- Notifications:** None.
- Status Bar:** Shows "Total rows: Query complete 00:00:00.067" and "CRLF Ln 16, Col 1".

## 5. Create a stored procedure that lists all passengers for a given flight number.

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

- Object Explorer:** Displays the database schema with tables like `boarding_pass`, `booking`, `flight`, and `flights`.
- Dashboard:** Shows the connection to `airportdb/postgres@PostgreSQL 17*`.
- Query Editor:** Contains the following SQL code:
 

```

CREATE OR REPLACE PROCEDURE passengers_for_flight(p_flight_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT p.passenger_id, p.first_name, p.last_name
    FROM passengers p
    JOIN booking b ON p.passenger_id = b.passenger_id
    JOIN booking_flight bf ON b.booking_id = bf.booking_id
    WHERE bf.flight_id = p_flight_id;
END;
$$;
      
```
- Data Output:** Shows the message "Query returned successfully in 81 msec."
- Messages:** Shows the message "Query returned successfully in 81 msec." in a green box.
- Notifications:** None.
- Status Bar:** Shows "Total rows: Query complete 00:00:00.081" and "CRLF Ln 12, Col 1".

## 6. Create a stored procedure to find the passenger who has taken the greatest number of flights.

The screenshot shows the pgAdmin 4 interface with the Object Explorer on the left and a query editor on the right. The query editor contains the following SQL code:

```

CREATE OR REPLACE PROCEDURE passenger_most_flights()
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT p.passenger_id, p.first_name, p.last_name, COUNT(bf.flight_id) AS flight_count
    FROM passengers p
    JOIN booking b ON p.passenger_id = b.passenger_id
    JOIN booking_flight bf ON b.booking_id = bf.booking_id
    GROUP BY p.passenger_id
    ORDER BY flight_count DESC
    LIMIT 1;
END;
$$;

```

The Data Output tab shows the message: "Query returned successfully in 69 msec." The status bar at the bottom right indicates "Total rows: 0" and "Query complete 00:00:00.069".

7. Create a stored procedure to find all flights that are delayed by more than 24 hours.

The screenshot shows the pgAdmin 4 interface with the Object Explorer on the left and a query editor on the right. The query editor contains the following SQL code:

```

CREATE OR REPLACE PROCEDURE flights_delayed_24h()
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT *
    FROM flights
    WHERE (actual_departure - scheduled_departure) > INTERVAL '24 hours';
END;
$$;

```

The Data Output tab shows the message: "Query returned successfully in 69 msec." The status bar at the bottom right indicates "Total rows: 0" and "Query complete 00:00:00.069".

8. Create a function that counts the number of flights for each airline.

The screenshot shows the pgAdmin 4 interface with the Object Explorer on the left and a query editor on the right. The query editor contains the following SQL code:

```

CREATE OR REPLACE FUNCTION flights_count_per_airline()
RETURNS TABLE(airline_id INT, flight_count INT)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
        SELECT airline_id, COUNT(*)
        FROM flights
        GROUP BY airline_id;
END;
$$;

```

The Data Output tab shows the message "Query returned successfully in 76 msec." and the status bar indicates "Total rows: Query complete 00:00:00.076".

9. Create a stored procedure to calculate the average ticket price for a specific flight.

The screenshot shows the pgAdmin 4 interface with the Object Explorer on the left and a query editor on the right. The query editor contains the following SQL code:

```

CREATE OR REPLACE PROCEDURE avg_ticket_price(p_flight_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT AVG(price)
    FROM tickets
    WHERE flight_id = p_flight_id;
END;
$$;

```

The Data Output tab shows the message "Query returned successfully in 65 msec." and the status bar indicates "Total rows: Query complete 00:00:00.065".

10. Create a stored procedure to find the flight with the highest ticket price. The procedure should return the flight number, the departure and arrival airports, and the ticket price for the most expensive flight.

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer    Dashboard × airportdb/postgres@PostgreSQL 17\*

Queries    airportdb/postgres@PostgreSQL 17\*

No limit    Scratch Pad X

Query    Query History

```
1 CREATE OR REPLACE PROCEDURE most_expensive_flight()
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5   SELECT f.flight_id, f.departure_airport_id, f.arrival_airport_id, t.price
6   FROM flights f
7   JOIN tickets t ON f.flight_id = t.flight_id
8   ORDER BY t.price DESC
9   LIMIT 1;
10 END;
11 $$;
```

Data Output    Messages    Notifications

CREATE PROCEDURE

Query returned successfully in 97 msec.

Total rows:    Query complete 00:00:00.097

✓ Query returned successfully in 97 msec. X

CRLF    Ln 12, Col 1

A screenshot of the pgAdmin 4 interface. The left sidebar shows the 'Object Explorer' with a tree view of database objects like 'baggage', 'boarding\_pass', 'booking', 'flight', and 'tickets'. The main area is a 'Dashboard' showing a query in the 'Queries' tab. The query is a PostgreSQL procedure named 'most\_expensive\_flight'. It selects the flight ID, departure airport ID, arrival airport ID, and price from the 'flights' table, joining it with the 'tickets' table on the flight ID, ordering by price in descending order, and limiting the result to one row. The status bar at the bottom indicates the query was completed successfully in 97 msec. A message bar at the bottom right also shows the success message with a green checkmark and the same timing information.