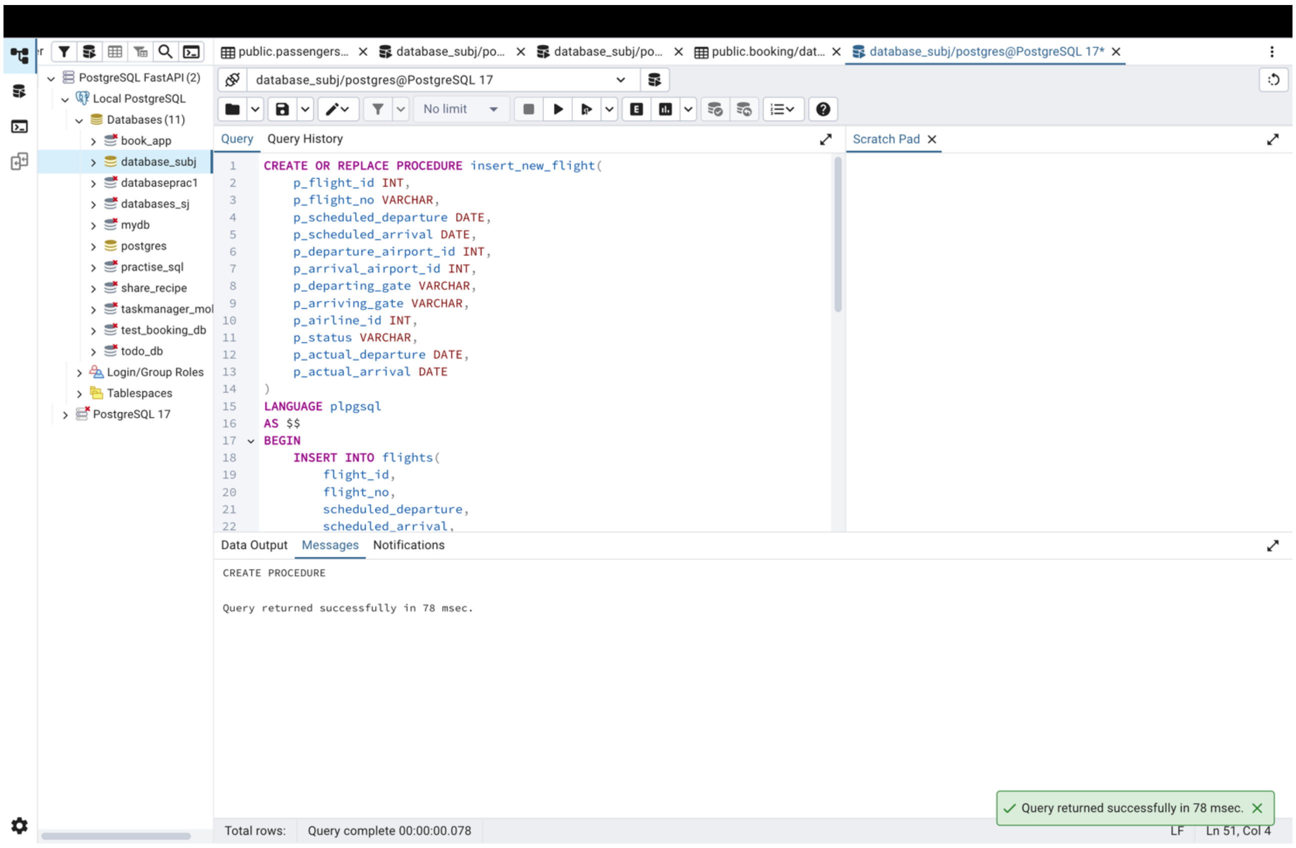
Laboratory work 10

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.



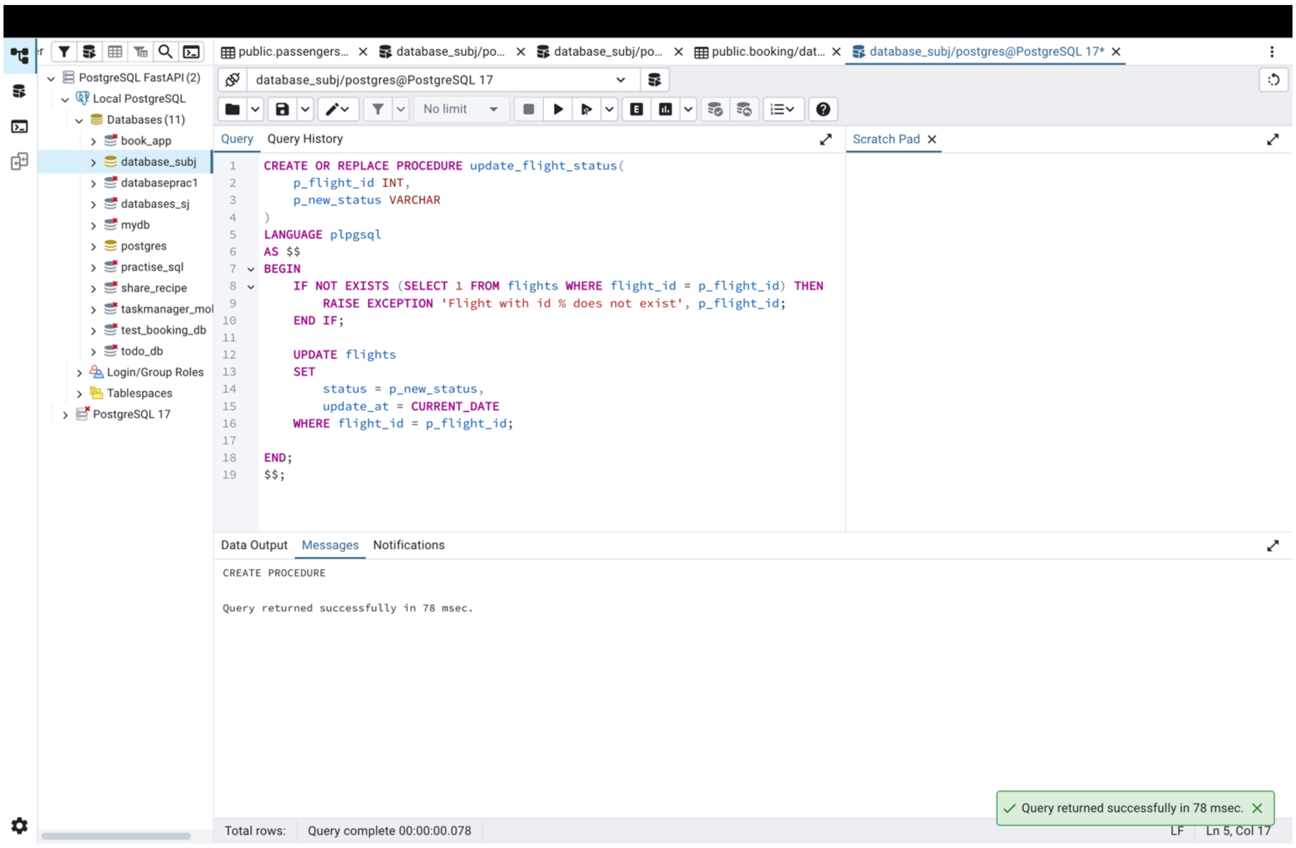
```sql
CREATE OR REPLACE PROCEDURE insert_new_flight(
    p_flight_id INT,
    p_flight_no VARCHAR,
    p_scheduled_departure DATE,
    p_scheduled_arrival DATE,
    p_departure_airport_id INT,
    p_arrival_airport_id INT,
    p_departing_gate VARCHAR,
    p_arriving_gate VARCHAR,
    p_airline_id INT,
    p_status VARCHAR,
    p_actual_departure DATE,
    p_actual_arrival DATE
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO flights(
        flight_id,
        flight_no,
        scheduled_departure,
        scheduled_arrival,
```

```
CREATE PROCEDURE

Query returned successfully in 78 msec.
```

Query returned successfully in 78 msec.

Total rows: Query complete 00:00:00.078

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

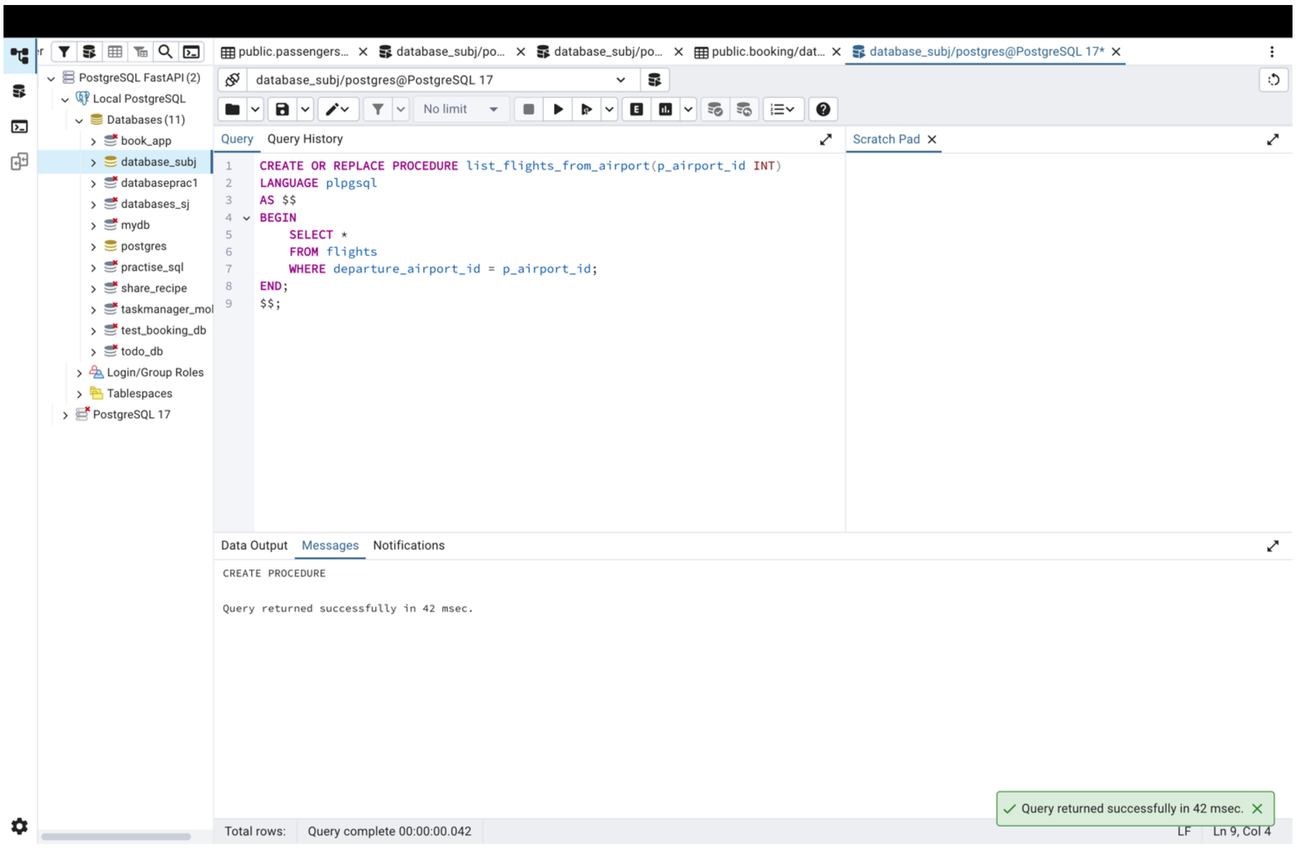


```sql
CREATE OR REPLACE PROCEDURE update_flight_status(
    p_flight_id INT,
    p_new_status VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM flights WHERE flight_id = p_flight_id) THEN
        RAISE EXCEPTION 'Flight with id % does not exist', p_flight_id;
    END IF;

    UPDATE flights
    SET
        status = p_new_status,
        update_at = CURRENT_DATE
    WHERE flight_id = p_flight_id;

END;
$$;
```

```
CREATE PROCEDURE

Query returned successfully in 78 msec.
```

Total rows: Query complete 00:00:00.078

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

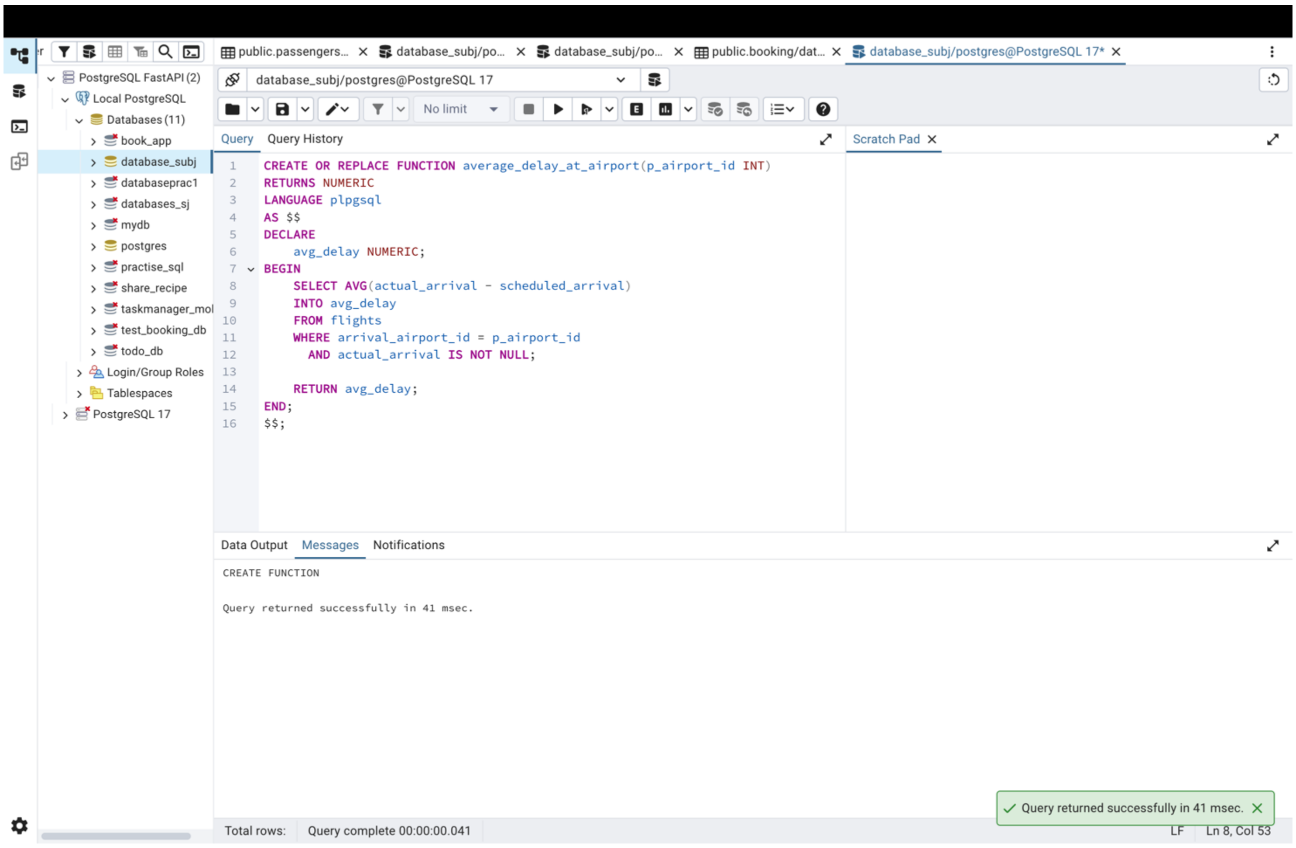


```sql
CREATE OR REPLACE PROCEDURE list_flights_from_airport(p_airport_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT *
    FROM flights
    WHERE departure_airport_id = p_airport_id;
END;
$$;
```

```
CREATE PROCEDURE

Query returned successfully in 42 msec.
```

Total rows: Query complete 00:00:00.042

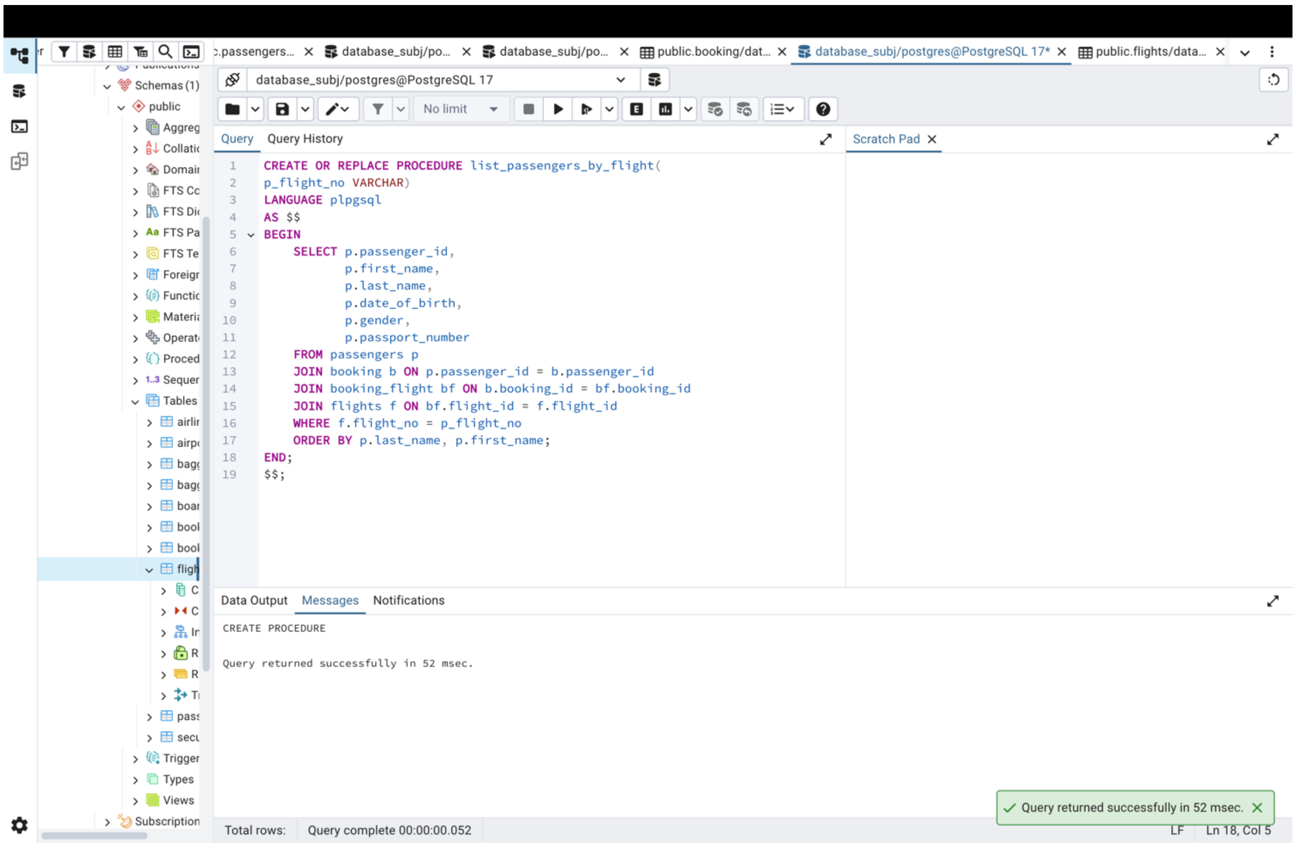
4. Create a function to calculate the average delay time of flights arriving at a specific airport.



```
CREATE OR REPLACE FUNCTION average_delay_at_airport(p_airport_id INT)
RETURNS NUMERIC
LANGUAGE plpgsql
AS $$
DECLARE
    avg_delay NUMERIC;
BEGIN
    SELECT AVG(actual_arrival - scheduled_arrival)
    INTO avg_delay
    FROM flights
    WHERE arrival_airport_id = p_airport_id
      AND actual_arrival IS NOT NULL;

    RETURN avg_delay;
END;
$$;
```

```
CREATE FUNCTION

Query returned successfully in 41 msec.
```

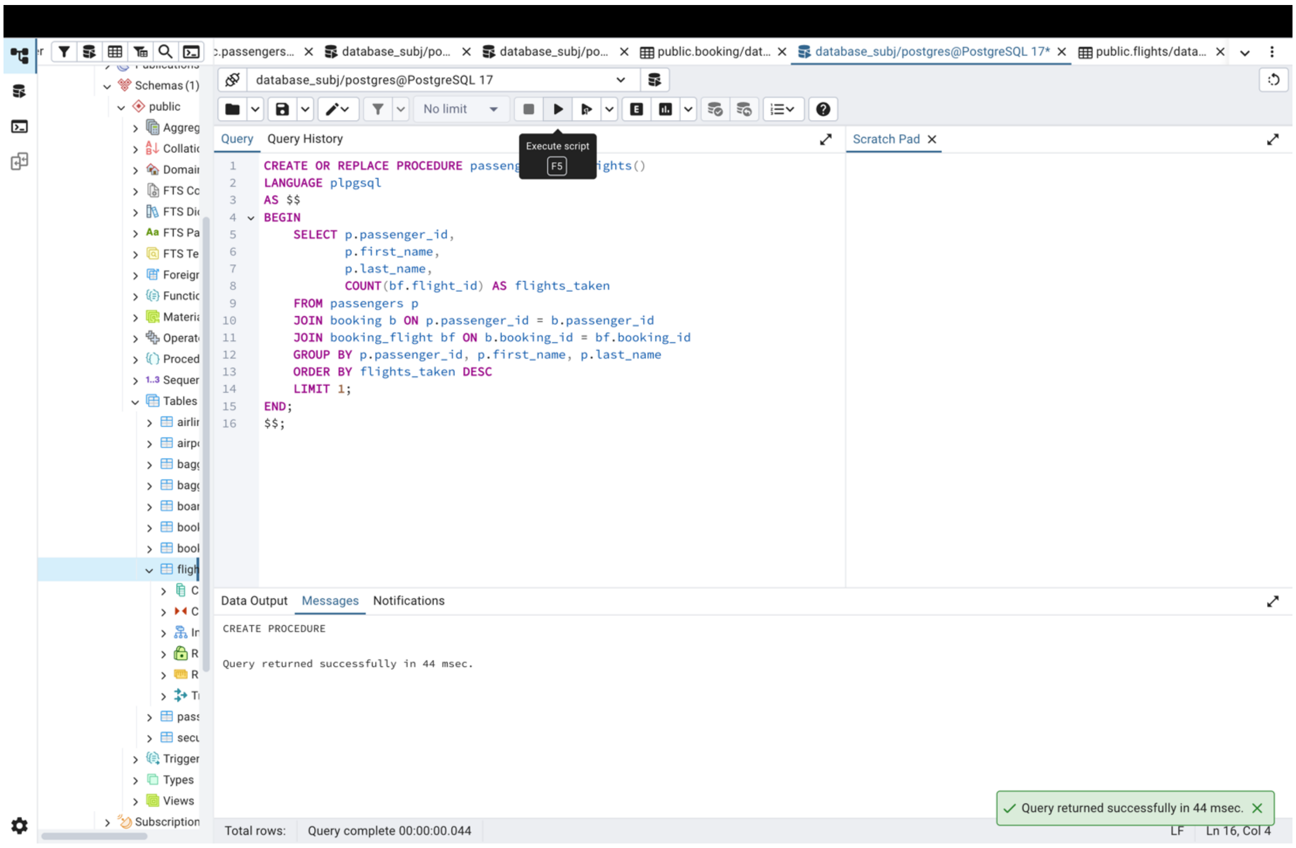
5. Create a stored procedure that lists all passengers for a given flight number.



```
CREATE OR REPLACE PROCEDURE list_passengers_by_flight(
p_flight_no VARCHAR)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT p.passenger_id,
           p.first_name,
           p.last_name,
           p.date_of_birth,
           p.gender,
           p.passport_number
    FROM passengers p
    JOIN booking b ON p.passenger_id = b.passenger_id
    JOIN booking_flight bf ON b.booking_id = bf.booking_id
    JOIN flights f ON bf.flight_id = f.flight_id
    WHERE f.flight_no = p_flight_no
    ORDER BY p.last_name, p.first_name;
END;
$$;
```

```
CREATE PROCEDURE

Query returned successfully in 52 msec.
```

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



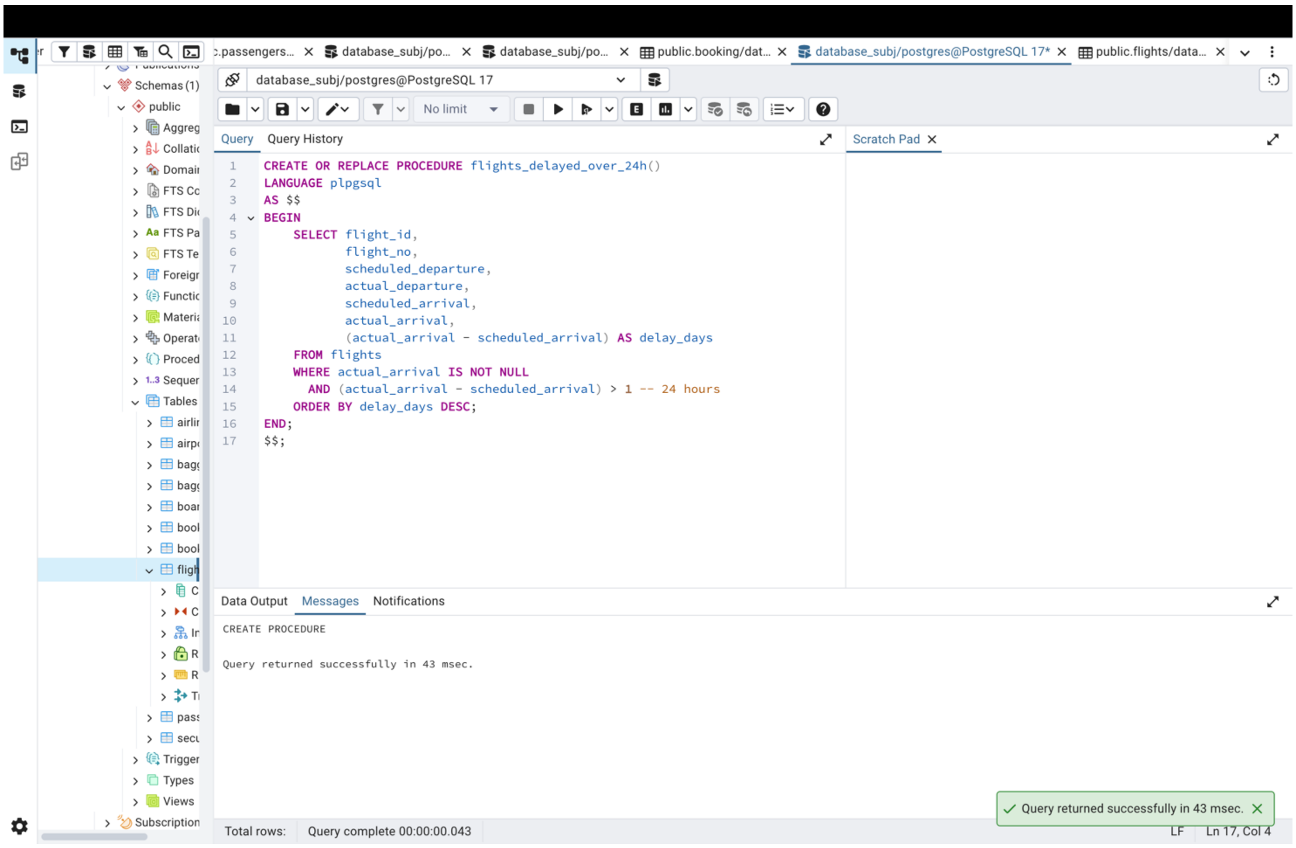
```
CREATE OR REPLACE PROCEDURE passeng        ights()
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT p.passenger_id,
           p.first_name,
           p.last_name,
           COUNT(bf.flight_id) AS flights_taken
    FROM passengers p
    JOIN booking b ON p.passenger_id = b.passenger_id
    JOIN booking_flight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

```
CREATE PROCEDURE

Query returned successfully in 44 msec.
```

Query returned successfully in 44 msec.

Total rows: Query complete 00:00:00.044

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



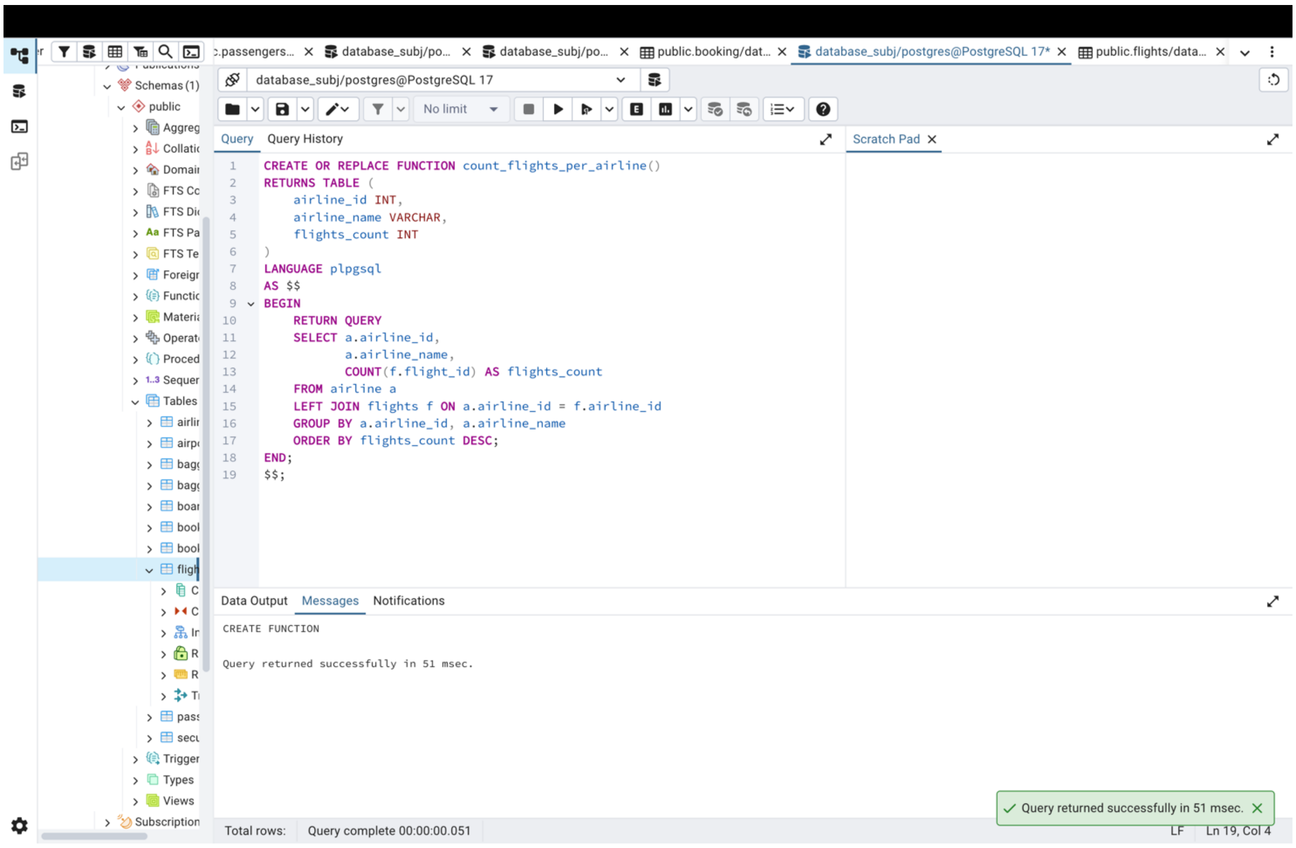
```
CREATE OR REPLACE PROCEDURE flights_delayed_over_24h()
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT flight_id,
           flight_no,
           scheduled_departure,
           actual_departure,
           scheduled_arrival,
           actual_arrival,
           (actual_arrival - scheduled_arrival) AS delay_days
    FROM flights
    WHERE actual_arrival IS NOT NULL
      AND (actual_arrival - scheduled_arrival) > 1 -- 24 hours
    ORDER BY delay_days DESC;
END;
$$;
```

```
CREATE PROCEDURE

Query returned successfully in 43 msec.
```

Query returned successfully in 43 msec.

Total rows: Query complete 00:00:00.043

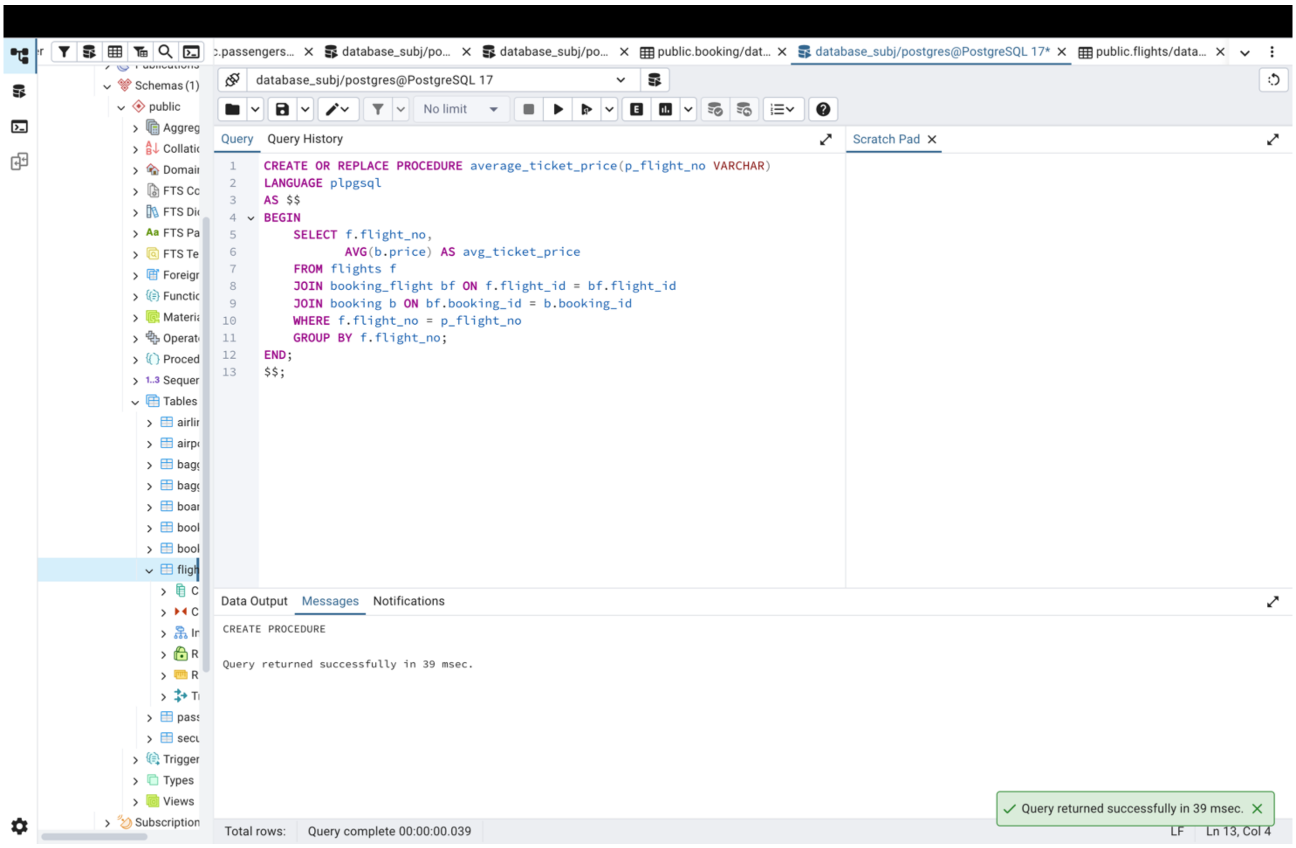
8. Create a function that counts the number of flights for each airline.



```
CREATE OR REPLACE FUNCTION count_flights_per_airline()
RETURNS TABLE (
    airline_id INT,
    airline_name VARCHAR,
    flights_count INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT a.airline_id,
           a.airline_name,
           COUNT(f.flight_id) AS flights_count
    FROM airline a
    LEFT JOIN flights f ON a.airline_id = f.airline_id
    GROUP BY a.airline_id, a.airline_name
    ORDER BY flights_count DESC;
END;
$$;
```

```
CREATE FUNCTION

Query returned successfully in 51 msec.
```

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

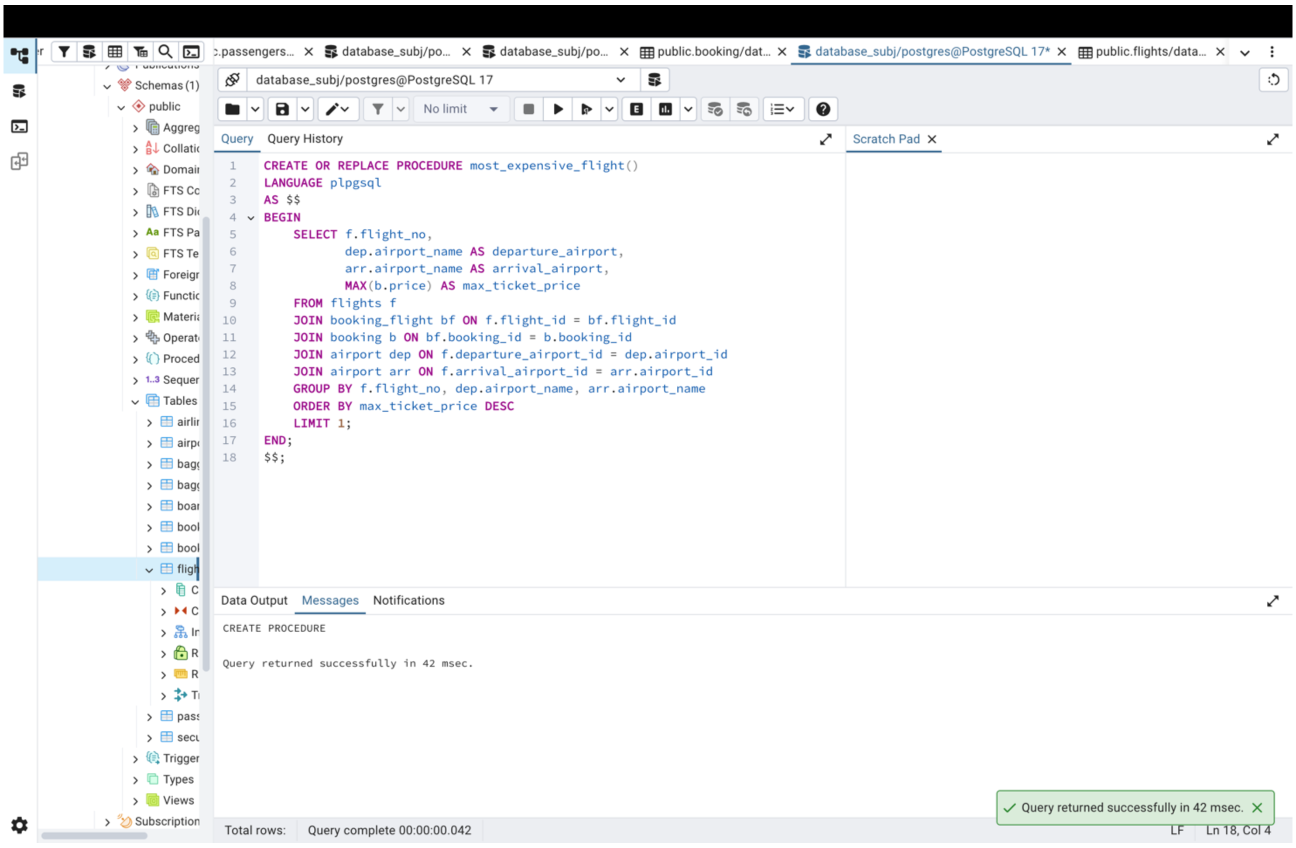


```
CREATE OR REPLACE PROCEDURE average_ticket_price(p_flight_no VARCHAR)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT f.flight_no,
           AVG(b.price) AS avg_ticket_price
    FROM flights f
    JOIN booking_flight bf ON f.flight_id = bf.flight_id
    JOIN booking b ON bf.booking_id = b.booking_id
    WHERE f.flight_no = p_flight_no
    GROUP BY f.flight_no;
END;
$$;
```

```
CREATE PROCEDURE

Query returned successfully in 39 msec.
```

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.



```
CREATE OR REPLACE PROCEDURE most_expensive_flight()
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT f.flight_no,
           dep.airport_name AS departure_airport,
           arr.airport_name AS arrival_airport,
           MAX(b.price) AS max_ticket_price
    FROM flights f
    JOIN booking_flight bf ON f.flight_id = bf.flight_id
    JOIN booking b ON bf.booking_id = b.booking_id
    JOIN airport dep ON f.departure_airport_id = dep.airport_id
    JOIN airport arr ON f.arrival_airport_id = arr.airport_id
    GROUP BY f.flight_no, dep.airport_name, arr.airport_name
    ORDER BY max_ticket_price DESC
    LIMIT 1;
END;
$$;
```

```
CREATE PROCEDURE

Query returned successfully in 42 msec.
```

Query returned successfully in 42 msec.

Total rows: Query complete 00:00:00.042 LF Ln 18, Col 4