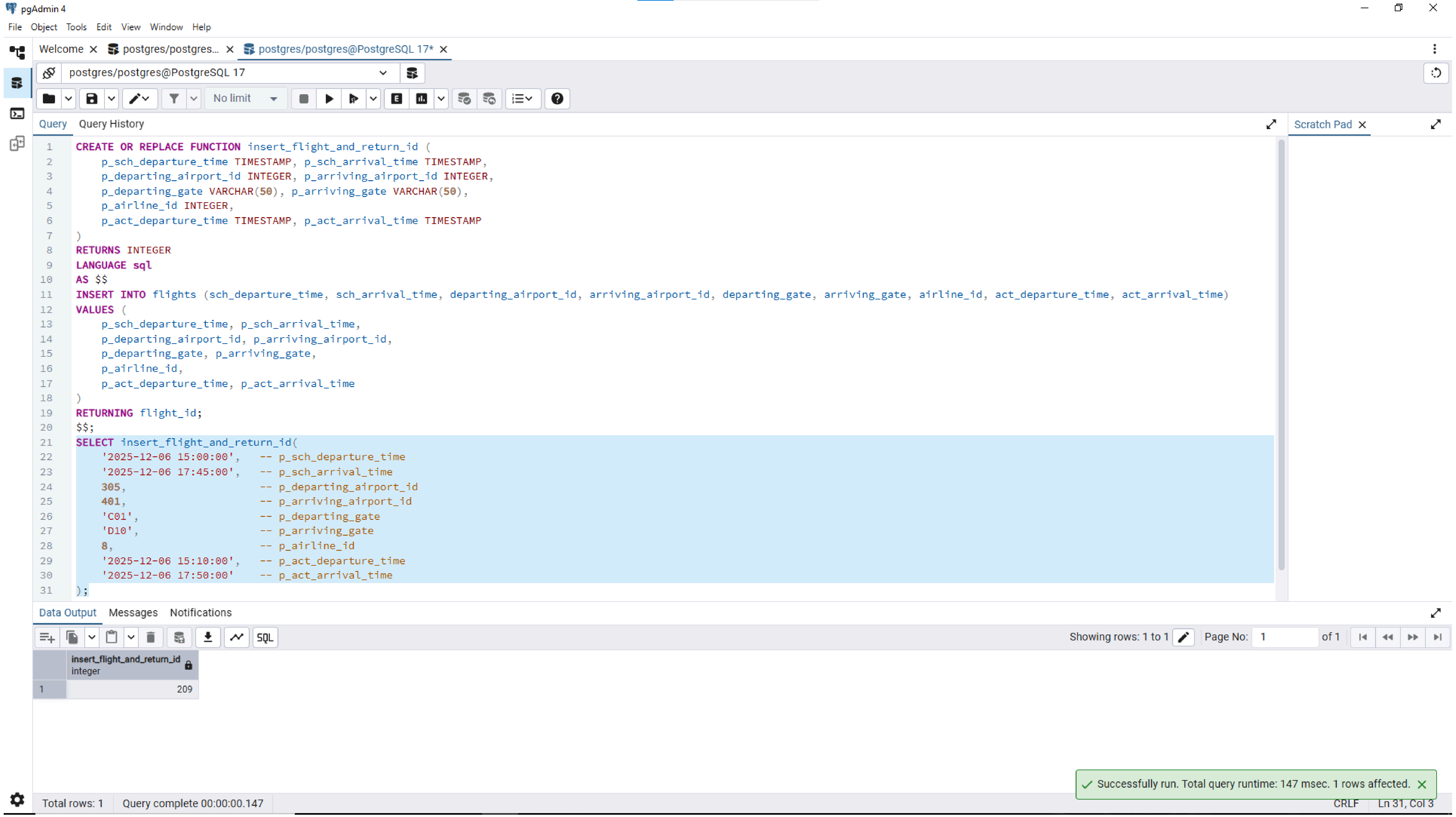


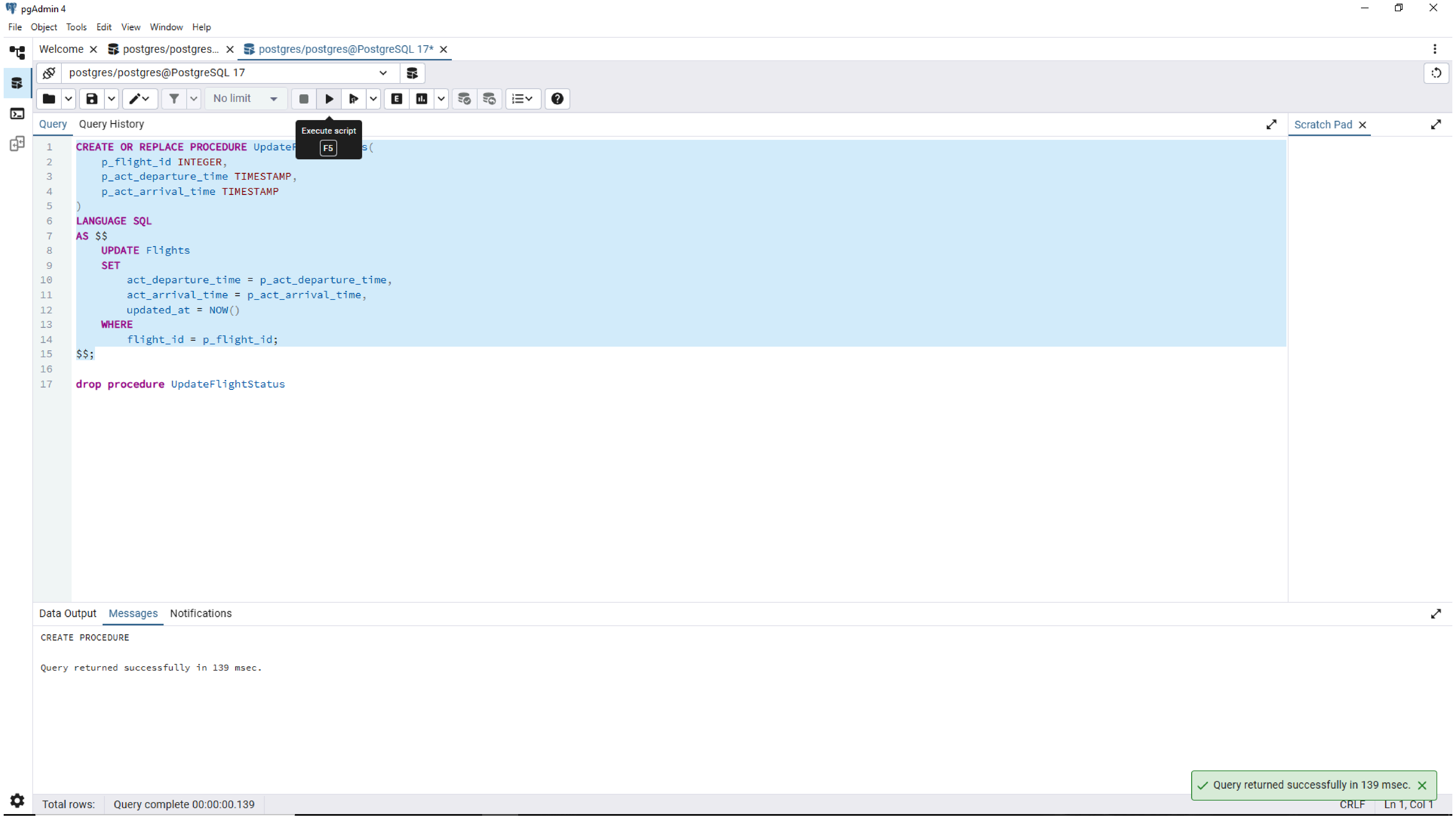
```sql
CREATE OR REPLACE FUNCTION insert_flight_and_return_id (
    p_sch_departure_time TIMESTAMP, p_sch_arrival_time TIMESTAMP,
    p_departing_airport_id INTEGER, p_arriving_airport_id INTEGER,
    p_departing_gate VARCHAR(50), p_arriving_gate VARCHAR(50),
    p_airline_id INTEGER,
    p_act_departure_time TIMESTAMP, p_act_arrival_time TIMESTAMP
)
RETURNS INTEGER
LANGUAGE sql
AS $$
INSERT INTO flights (sch_departure_time, sch_arrival_time, departing_airport_id, arriving_airport_id, departing_gate, arriving_gate, airline_id, act_departure_time, act_arrival_time)
VALUES (
    p_sch_departure_time, p_sch_arrival_time,
    p_departing_airport_id, p_arriving_airport_id,
    p_departing_gate, p_arriving_gate,
    p_airline_id,
    p_act_departure_time, p_act_arrival_time
)
RETURNING flight_id;
$$;
SELECT insert_flight_and_return_id(
    '2025-12-06 15:00:00',   -- p_sch_departure_time
    '2025-12-06 17:45:00',   -- p_sch_arrival_time
    305,                     -- p_departing_airport_id
    401,                     -- p_arriving_airport_id
    'C01',                   -- p_departing_gate
    'D10',                   -- p_arriving_gate
    8,                       -- p_airline_id
    '2025-12-06 15:10:00',   -- p_act_departure_time
    '2025-12-06 17:50:00'    -- p_act_arrival_time
);
```

	insert_flight_and_return_id integer
1	209

Successfully run. Total query runtime: 147 msec. 1 rows affected.

Total rows: 1 Query complete 00:00:00.147

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



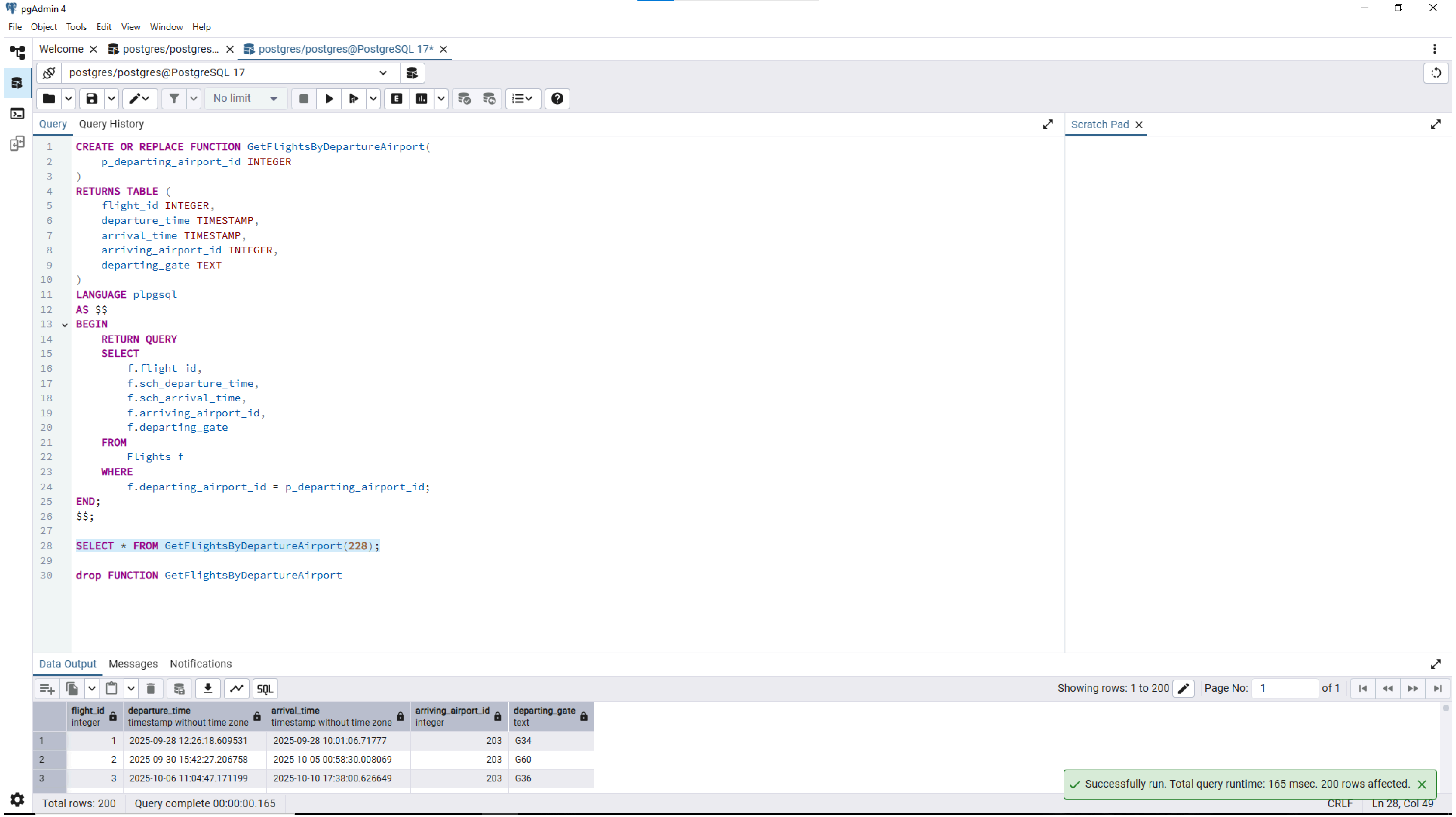
```sql
CREATE OR REPLACE PROCEDURE UpdateF[illegible]s(
    p_flight_id INTEGER,
    p_act_departure_time TIMESTAMP,
    p_act_arrival_time TIMESTAMP
)
LANGUAGE SQL
AS $$
    UPDATE Flights
    SET
        act_departure_time = p_act_departure_time,
        act_arrival_time = p_act_arrival_time,
        updated_at = NOW()
    WHERE
        flight_id = p_flight_id;
$$;

drop procedure UpdateFlightStatus
```

```
CREATE PROCEDURE

Query returned successfully in 139 msec.
```

Query returned successfully in 139 msec.

Total rows: Query complete 00:00:00.139 CRLF Ln 1, Col 1

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



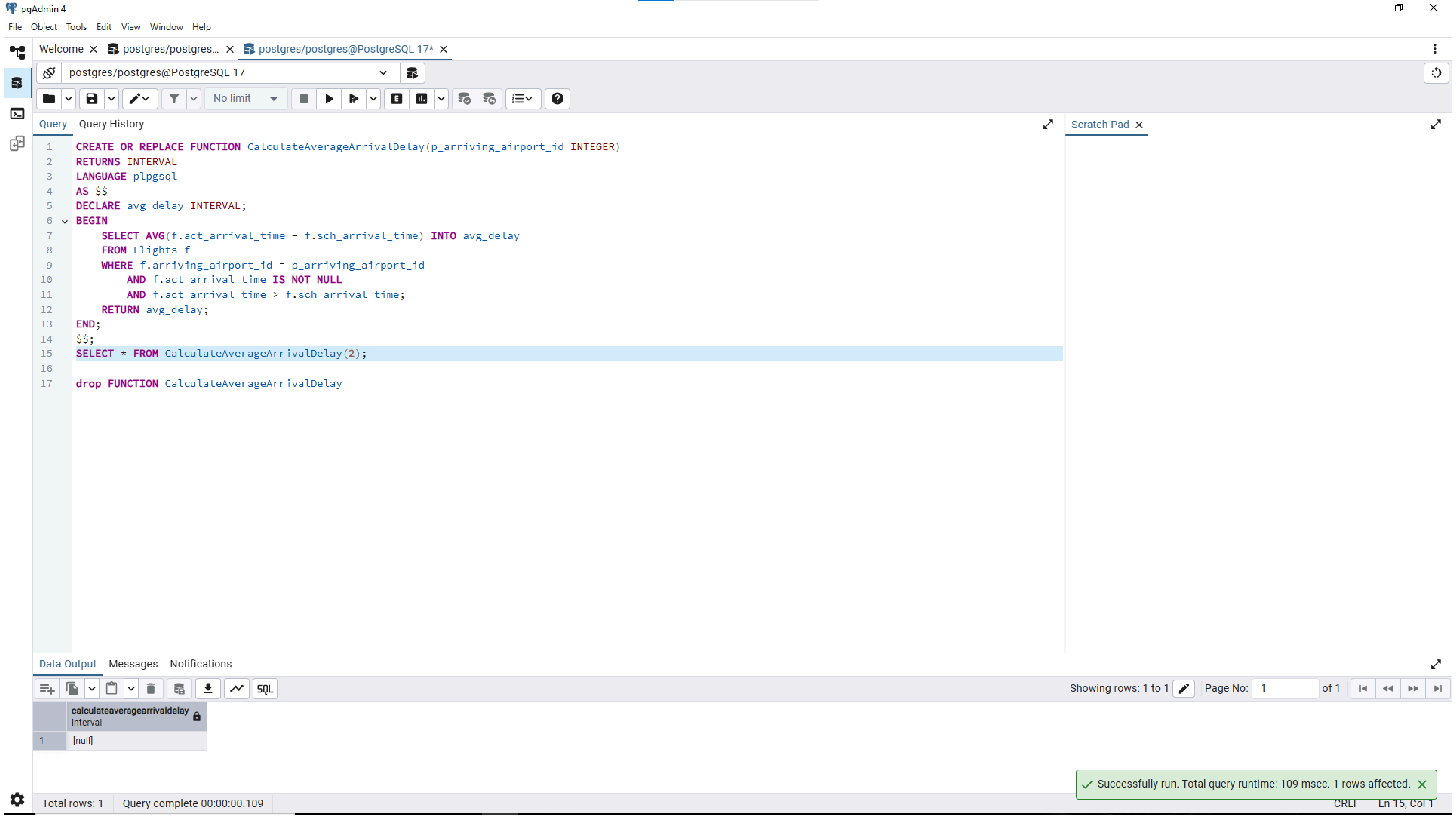
```sql
CREATE OR REPLACE FUNCTION GetFlightsByDepartureAirport(
    p_departing_airport_id INTEGER
)
RETURNS TABLE (
    flight_id INTEGER,
    departure_time TIMESTAMP,
    arrival_time TIMESTAMP,
    arriving_airport_id INTEGER,
    departing_gate TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        f.flight_id,
        f.sch_departure_time,
        f.sch_arrival_time,
        f.arriving_airport_id,
        f.departing_gate
    FROM
        Flights f
    WHERE
        f.departing_airport_id = p_departing_airport_id;
END;
$$;

SELECT * FROM GetFlightsByDepartureAirport(228);

drop FUNCTION GetFlightsByDepartureAirport
```

	flight_id integer	departure_time timestamp without time zone	arrival_time timestamp without time zone	arriving_airport_id integer	departing_gate text
1	1	2025-09-28 12:26:18.609531	2025-09-28 10:01:06.71777	203	G34
2	2	2025-09-30 15:42:27.206758	2025-10-05 00:58:30.008069	203	G60
3	3	2025-10-06 11:04:47.171199	2025-10-10 17:38:00.626649	203	G36

Successfully run. Total query runtime: 165 msec. 200 rows affected.

Total rows: 200 Query complete 00:00:00.165 CRLF Ln 28, Col 49

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

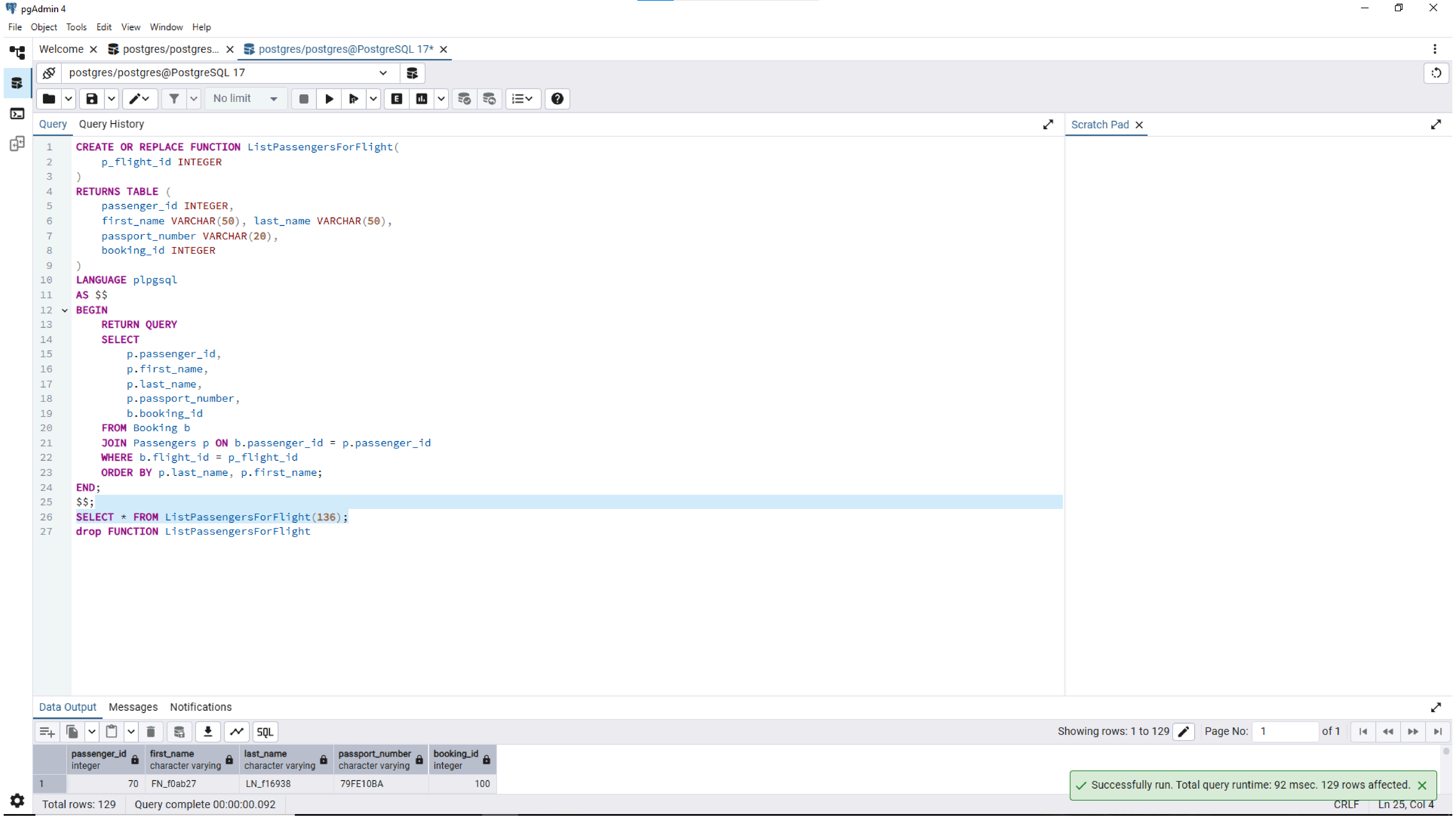


```
CREATE OR REPLACE FUNCTION CalculateAverageArrivalDelay(p_arriving_airport_id INTEGER)
RETURNS INTERVAL
LANGUAGE plpgsql
AS $$
DECLARE avg_delay INTERVAL;
BEGIN
    SELECT AVG(f.act_arrival_time - f.sch_arrival_time) INTO avg_delay
    FROM Flights f
    WHERE f.arriving_airport_id = p_arriving_airport_id
        AND f.act_arrival_time IS NOT NULL
        AND f.act_arrival_time > f.sch_arrival_time;
    RETURN avg_delay;
END;
$$;
SELECT * FROM CalculateAverageArrivalDelay(2);

drop FUNCTION CalculateAverageArrivalDelay
```

	calculateaveragearrivaldelay interval
1	[null]

```
Successfully run. Total query runtime: 109 msec. 1 rows affected.
Total rows: 1    Query complete 00:00:00.109
```

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



```
CREATE OR REPLACE FUNCTION ListPassengersForFlight(
    p_flight_id INTEGER
)
RETURNS TABLE (
    passenger_id INTEGER,
    first_name VARCHAR(50), last_name VARCHAR(50),
    passport_number VARCHAR(20),
    booking_id INTEGER
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.passenger_id,
        p.first_name,
        p.last_name,
        p.passport_number,
        b.booking_id
    FROM Booking b
    JOIN Passengers p ON b.passenger_id = p.passenger_id
    WHERE b.flight_id = p_flight_id
    ORDER BY p.last_name, p.first_name;
END;
$$;
SELECT * FROM ListPassengersForFlight(136);
drop FUNCTION ListPassengersForFlight
```

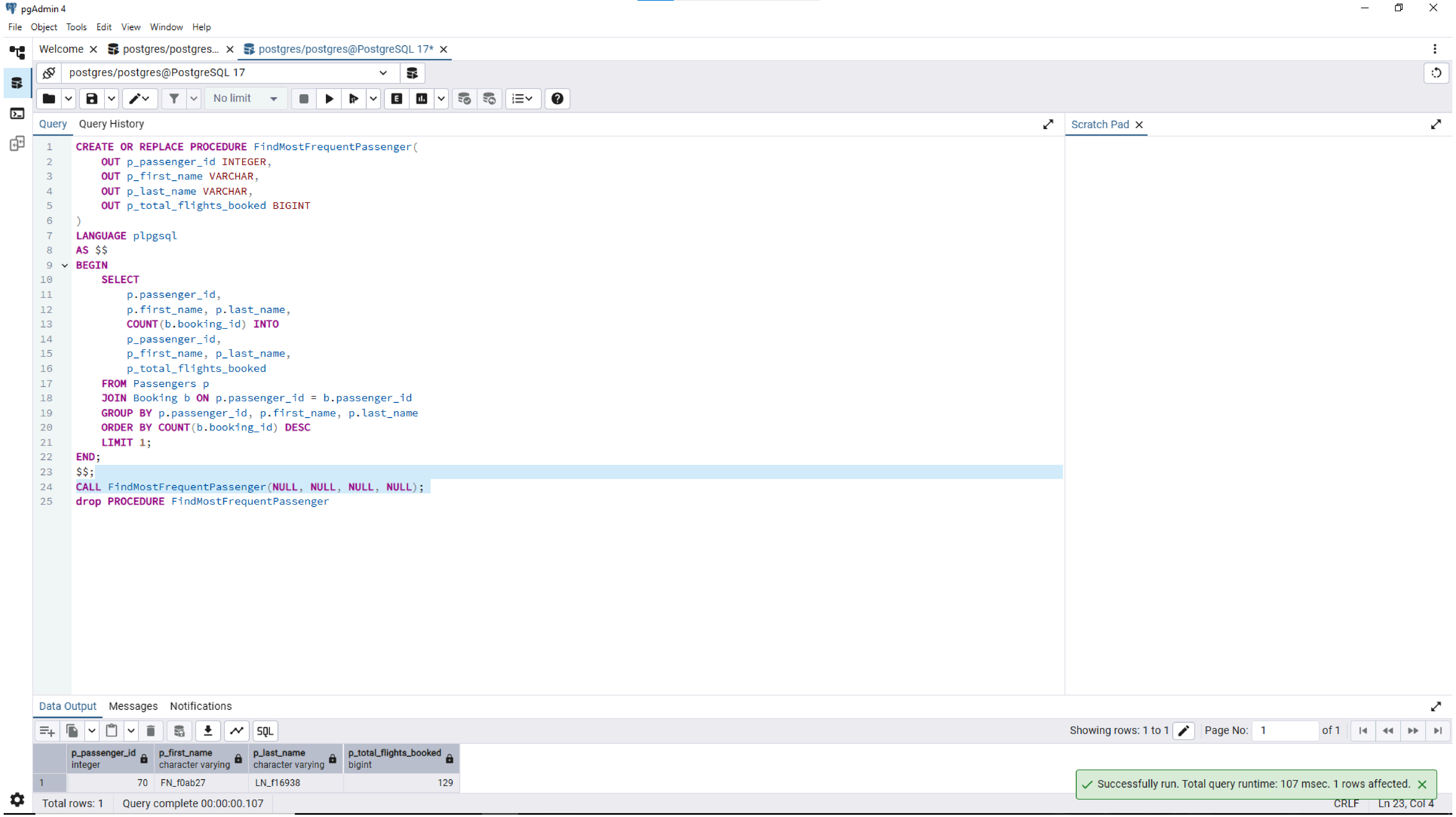
Showing rows: 1 to 129 Page No: 1 of 1

	passenger_id integer	first_name character varying	last_name character varying	passport_number character varying	booking_id integer
1	70	FN_f0ab27	LN_f16938	79FE10BA	100

Successfully run. Total query runtime: 92 msec. 129 rows affected.

Total rows: 129 Query complete 00:00:00.092

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



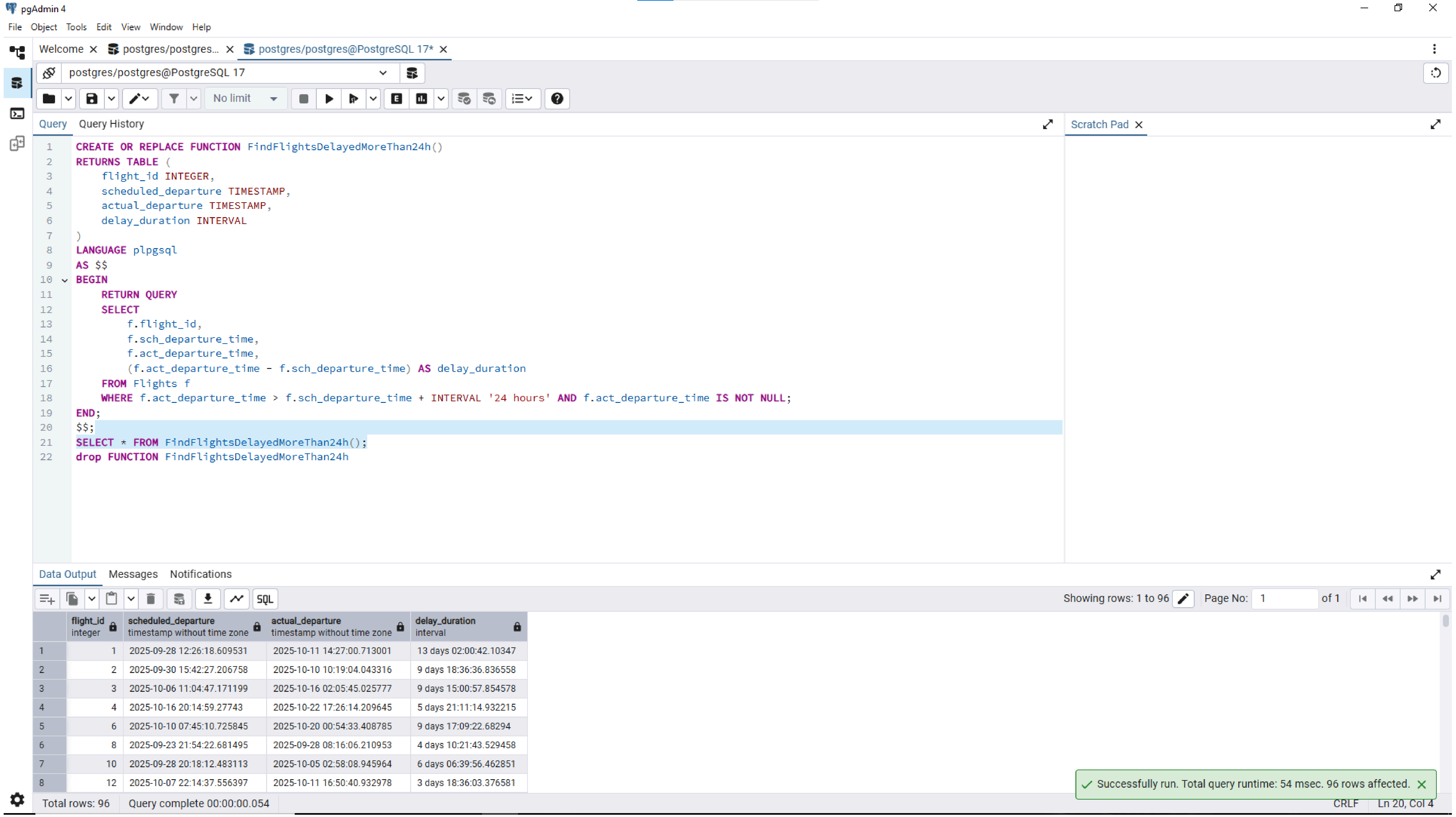
```sql
CREATE OR REPLACE PROCEDURE FindMostFrequentPassenger(
    OUT p_passenger_id INTEGER,
    OUT p_first_name VARCHAR,
    OUT p_last_name VARCHAR,
    OUT p_total_flights_booked BIGINT
)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT
        p.passenger_id,
        p.first_name, p.last_name,
        COUNT(b.booking_id) INTO
        p_passenger_id,
        p_first_name, p_last_name,
        p_total_flights_booked
    FROM Passengers p
    JOIN Booking b ON p.passenger_id = b.passenger_id
    GROUP BY p.passenger_id, p.first_name, p.last_name
    ORDER BY COUNT(b.booking_id) DESC
    LIMIT 1;
END;
$$;
CALL FindMostFrequentPassenger(NULL, NULL, NULL, NULL);
drop PROCEDURE FindMostFrequentPassenger
```

	p_passenger_id integer	p_first_name character varying	p_last_name character varying	p_total_flights_booked bigint
1	70	FN_f0ab27	LN_f16938	129

Successfully run. Total query runtime: 107 msec. 1 rows affected.

Total rows: 1 Query complete 00:00:00.107

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



```sql
CREATE OR REPLACE FUNCTION FindFlightsDelayedMoreThan24h()
RETURNS TABLE (
    flight_id INTEGER,
    scheduled_departure TIMESTAMP,
    actual_departure TIMESTAMP,
    delay_duration INTERVAL
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        f.flight_id,
        f.sch_departure_time,
        f.act_departure_time,
        (f.act_departure_time - f.sch_departure_time) AS delay_duration
    FROM Flights f
    WHERE f.act_departure_time > f.sch_departure_time + INTERVAL '24 hours' AND f.act_departure_time IS NOT NULL;
END;
$$;
SELECT * FROM FindFlightsDelayedMoreThan24h();
drop FUNCTION FindFlightsDelayedMoreThan24h
```

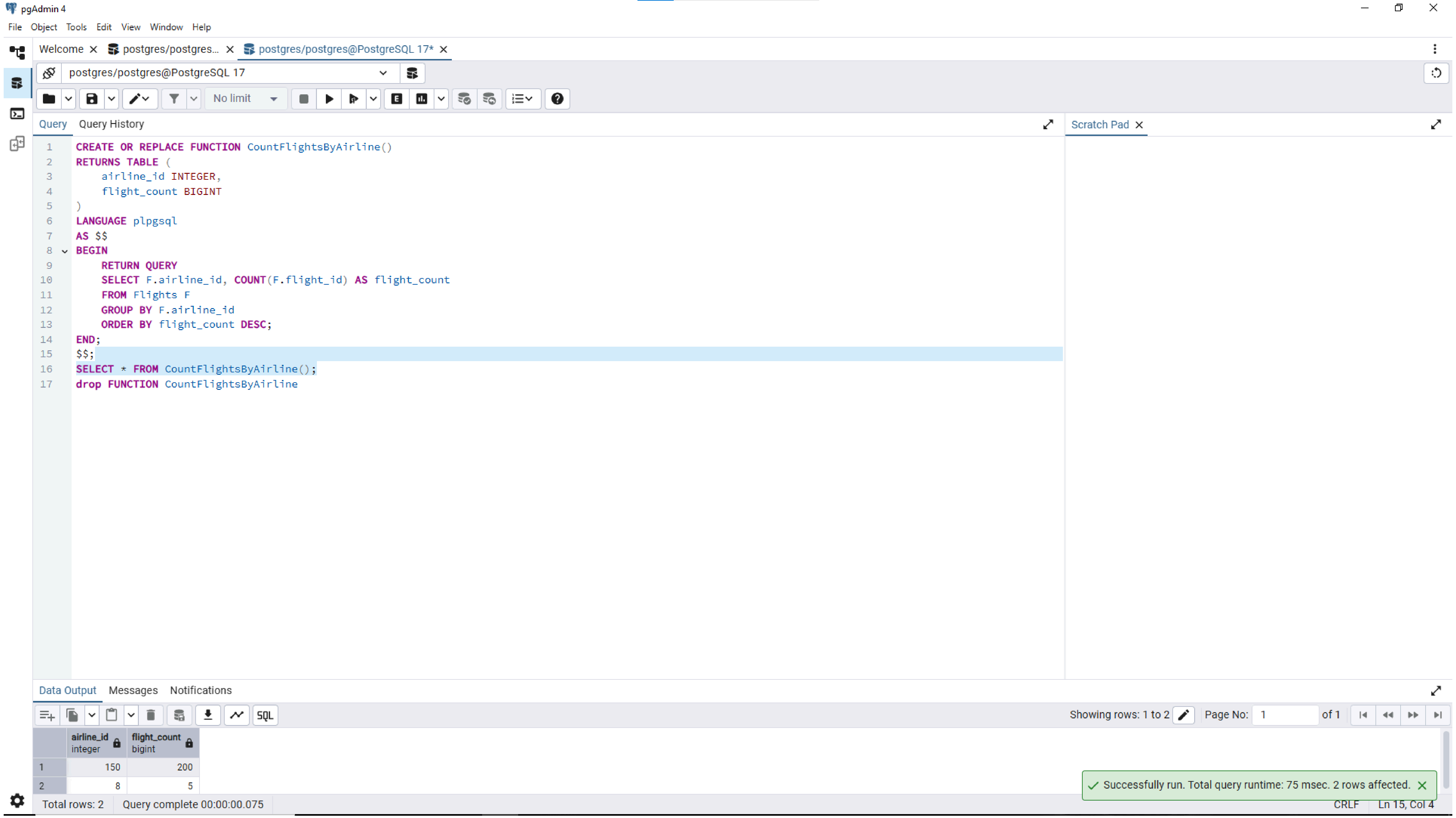
Showing rows: 1 to 96 Page No: 1 of 1

	flight_id integer	scheduled_departure timestamp without time zone	actual_departure timestamp without time zone	delay_duration interval
1	1	2025-09-28 12:26:18.609531	2025-10-11 14:27:00.713001	13 days 02:00:42.10347
2	2	2025-09-30 15:42:27.206758	2025-10-10 10:19:04.043316	9 days 18:36:36.836558
3	3	2025-10-06 11:04:47.171199	2025-10-16 02:05:45.025777	9 days 15:00:57.854578
4	4	2025-10-16 20:14:59.27743	2025-10-22 17:26:14.209645	5 days 21:11:14.932215
5	6	2025-10-10 07:45:10.725845	2025-10-20 00:54:33.408785	9 days 17:09:22.68294
6	8	2025-09-23 21:54:22.681495	2025-09-28 08:16:06.210953	4 days 10:21:43.529458
7	10	2025-09-28 20:18:12.483113	2025-10-05 02:58:08.945964	6 days 06:39:56.462851
8	12	2025-10-07 22:14:37.556397	2025-10-11 16:50:40.932978	3 days 18:36:03.376581

Successfully run. Total query runtime: 54 msec. 96 rows affected.

Total rows: 96 Query complete 00:00:00.054

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



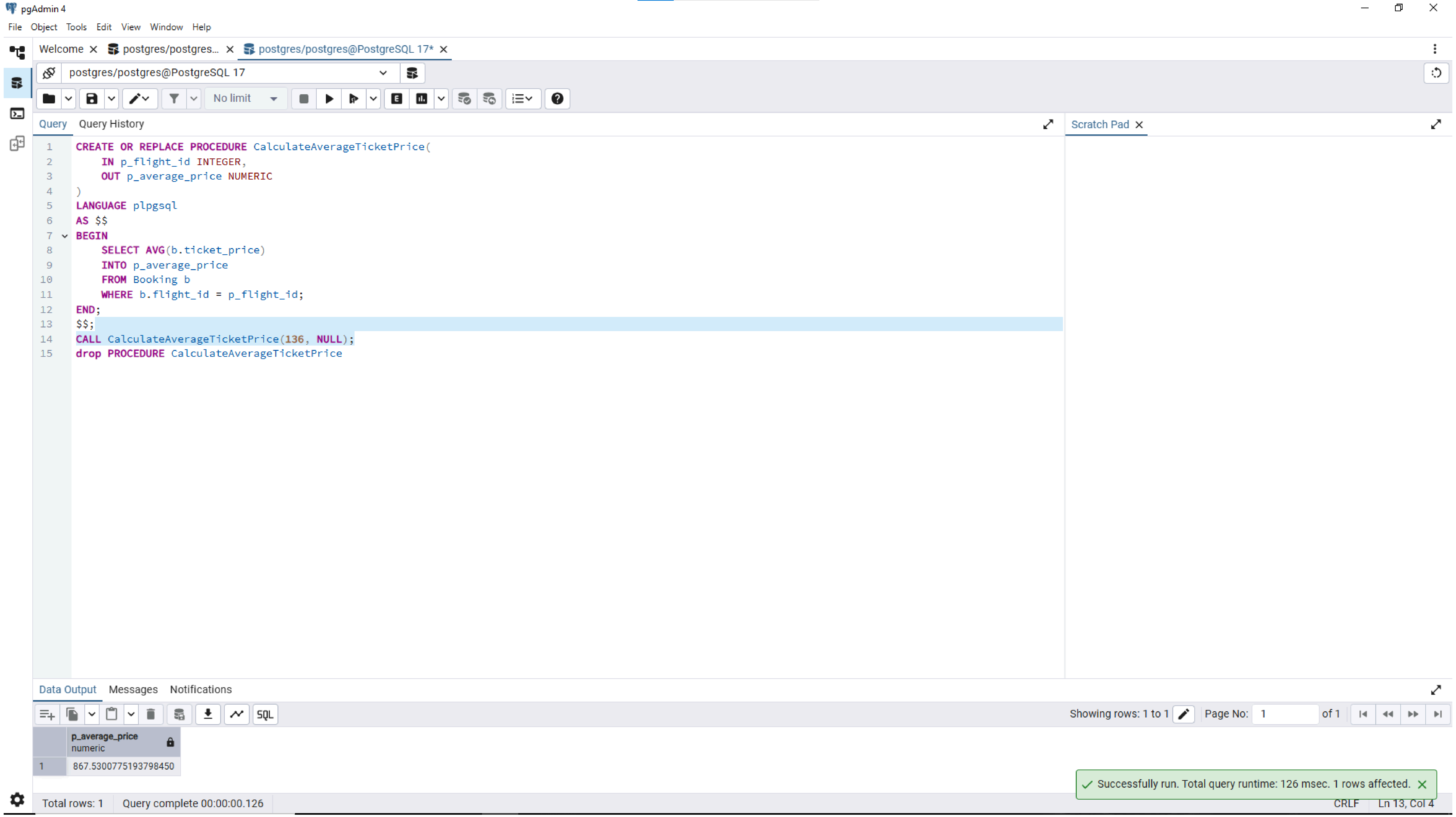
```
CREATE OR REPLACE FUNCTION CountFlightsByAirline()
RETURNS TABLE (
    airline_id INTEGER,
    flight_count BIGINT
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT F.airline_id, COUNT(F.flight_id) AS flight_count
    FROM Flights F
    GROUP BY F.airline_id
    ORDER BY flight_count DESC;
END;
$$;
SELECT * FROM CountFlightsByAirline();
drop FUNCTION CountFlightsByAirline
```

	airline_id integer	flight_count bigint
1	150	200
2	8	5

Successfully run. Total query runtime: 75 msec. 2 rows affected.

Total rows: 2   Query complete 00:00:00.075

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



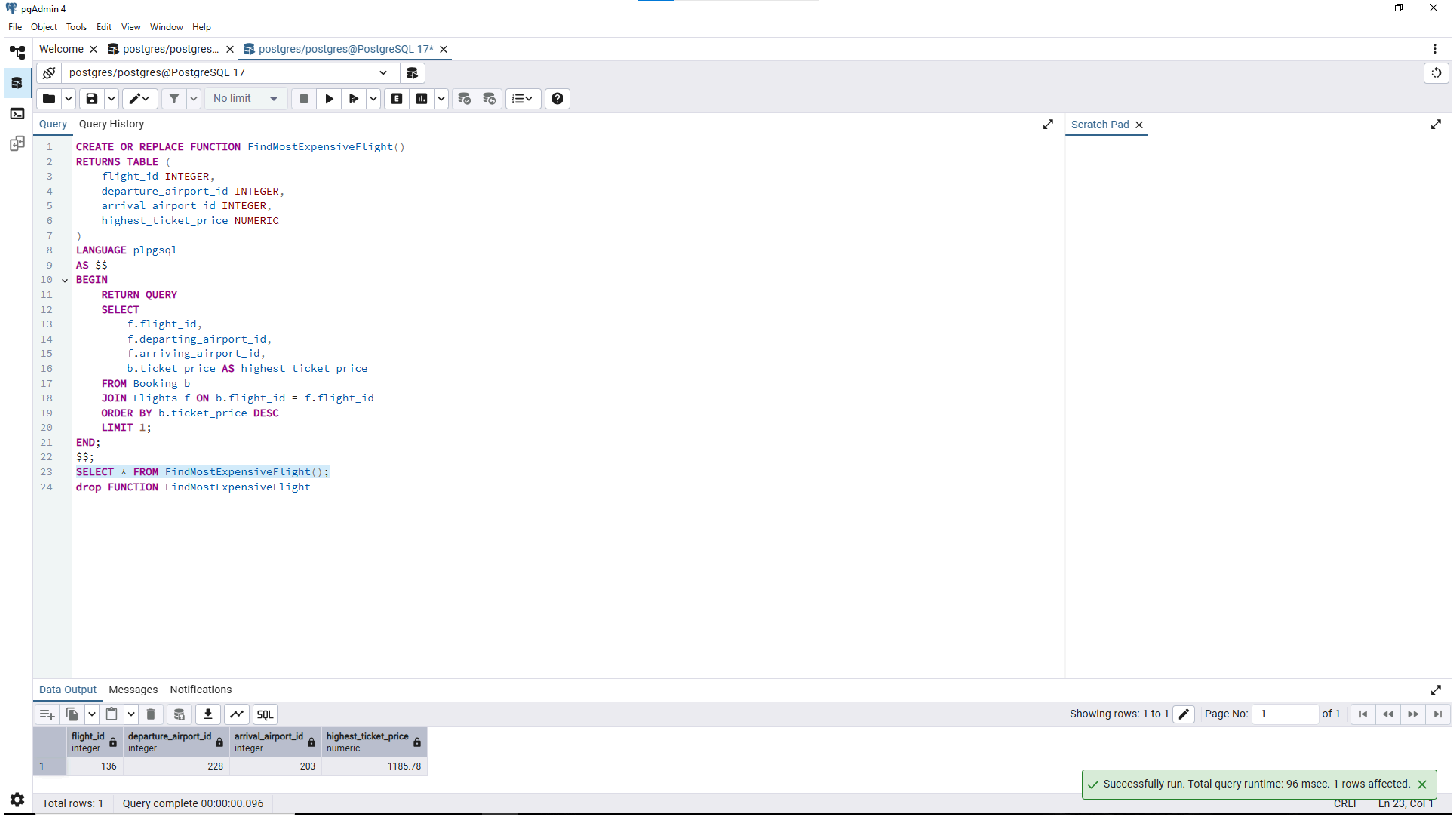
```sql
CREATE OR REPLACE PROCEDURE CalculateAverageTicketPrice(
    IN p_flight_id INTEGER,
    OUT p_average_price NUMERIC
)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT AVG(b.ticket_price)
    INTO p_average_price
    FROM Booking b
    WHERE b.flight_id = p_flight_id;
END;
$$;
CALL CalculateAverageTicketPrice(136, NULL);
drop PROCEDURE CalculateAverageTicketPrice
```

```
  p_average_price
  numeric
1 867.5300775193798450
```

Successfully run. Total query runtime: 126 msec. 1 rows affected.

Total rows: 1   Query complete 00:00:00.126

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



```sql
CREATE OR REPLACE FUNCTION FindMostExpensiveFlight()
RETURNS TABLE (
    flight_id INTEGER,
    departure_airport_id INTEGER,
    arrival_airport_id INTEGER,
    highest_ticket_price NUMERIC
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        f.flight_id,
        f.departing_airport_id,
        f.arriving_airport_id,
        b.ticket_price AS highest_ticket_price
    FROM Booking b
    JOIN Flights f ON b.flight_id = f.flight_id
    ORDER BY b.ticket_price DESC
    LIMIT 1;
END;
$$;
SELECT * FROM FindMostExpensiveFlight();
drop FUNCTION FindMostExpensiveFlight
```

Showing rows: 1 to 1 Page No: 1 of 1

	flight_id integer	departure_airport_id integer	arrival_airport_id integer	highest_ticket_price numeric
1	136	228	203	1185.78

Successfully run. Total query runtime: 96 msec. 1 rows affected.

Total rows: 1 Query complete 00:00:00.096 CRLF Ln 23, Col 1

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.