ABDYKAMAT ADILET LAB 10

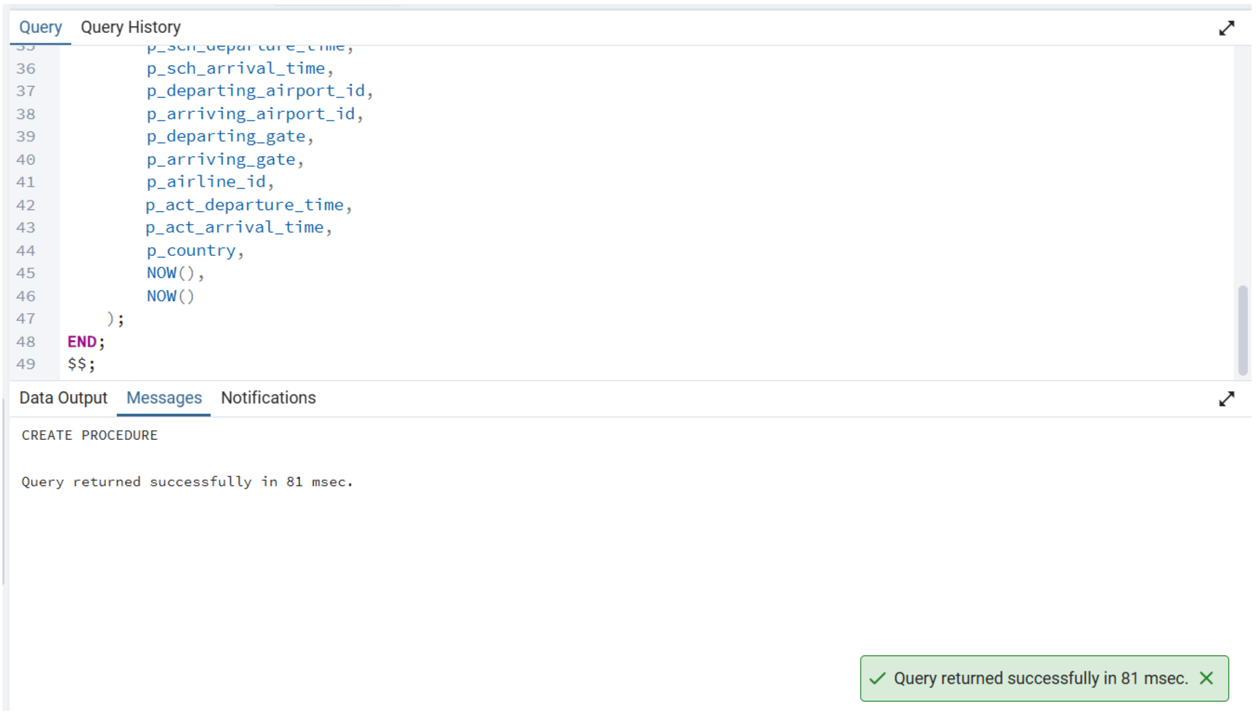
STORED PROCEDURES and FUNCTION.

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

```
CREATE OR REPLACE PROCEDURE insert_new_flight(
            p_flight_id VARCHAR(50),
    p_sch_departure_time TIMESTAMP,
    p_sch_arrival_time TIMESTAMP,
    p_departing_airport_id INT,
    p_arriving_airport_id INT,
    p_departing_gate VARCHAR(10),
    p_arriving_gate VARCHAR(10),
    p_airline_id INT,
    p_country VARCHAR(100),
            p_act_departure_time TIMESTAMP,
            p_act_arrival_time TIMESTAMP
)
LANGUAGE plpgsql
AS $$
BEGIN
            INSERT INTO flights(
                        flight_id,
        sch_departure_time,
        sch_arrival_time,
        departing_airport_id,
        arriving_airport_id,
        departing_gate,
        arriving_gate,
        airline_id,
                        act_departure_time,
                        act_arrival_time,
        country,
        created_at,
        updated_at
            )

            VALUES(
                        p_flight_id,
        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,
        p_country,
        NOW(),
        NOW()
            );
END;
$$;
```

FOR THE PROCEDURE TO WORK:

CALL insert_new_flight('FL123', '2024-01-20 10:00:00', '2024-01-20 12:00:00', 1, 2, 'A1', 'B2', 1, 'USA')
;


Query Query History

```
        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,
        p_country,
        NOW(),
        NOW()
    );
END;
$$;
```

Data Output Messages Notifications

```
CREATE PROCEDURE

Query returned successfully in 81 msec.
```

Query returned successfully in 81 msec.

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

CREATE OR REPLACE PROCEDURE uppdate_flights_status(

p_flight_id INT,

p_status VARCHAR(50)

)

LANGUAGE plpgsql

AS $$

BEGIN

UPDATE booking

SET status = p_status

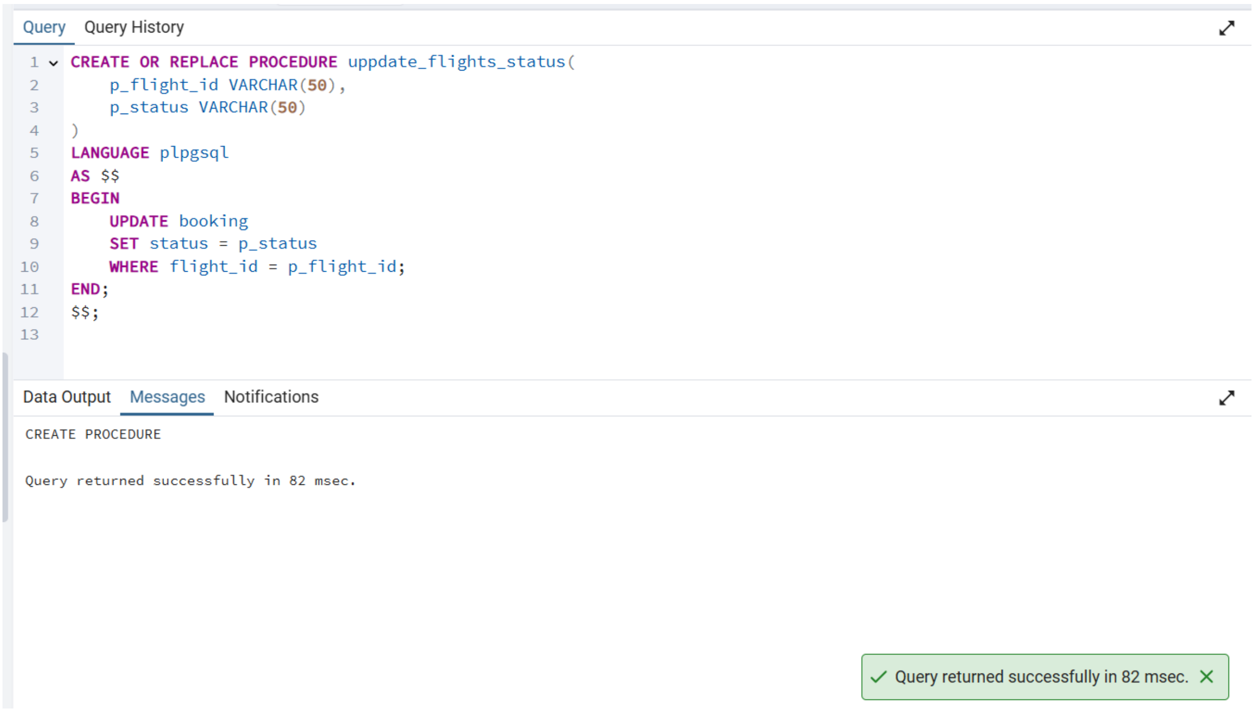
WHERE flight_id = p_flight_id;

END;

$$;

FOR THE PROCEDURE TO WORK:

CALL `uppdate_flights_status(5, 'Delayed');`


Query Query History

```
CREATE OR REPLACE PROCEDURE uppdate_flights_status(
    p_flight_id VARCHAR(50),
    p_status VARCHAR(50)
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE booking
    SET status = p_status
    WHERE flight_id = p_flight_id;
END;
$$;
```

Data Output Messages Notifications

```
CREATE PROCEDURE

Query returned successfully in 82 msec.
```

✓ Query returned successfully in 82 msec.

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

CREATE OR REPLACE FUNCTION GetFlightsByDeparture(

    p_departure_airport VARCHAR(5)

)

RETURNS TABLE (

    flight_number VARCHAR(10),

    departure_airport VARCHAR(5),

    arrival_airport VARCHAR(5),

    departure_time TIMESTAMP,

    arrival_time TIMESTAMP,

    status VARCHAR(20)

)

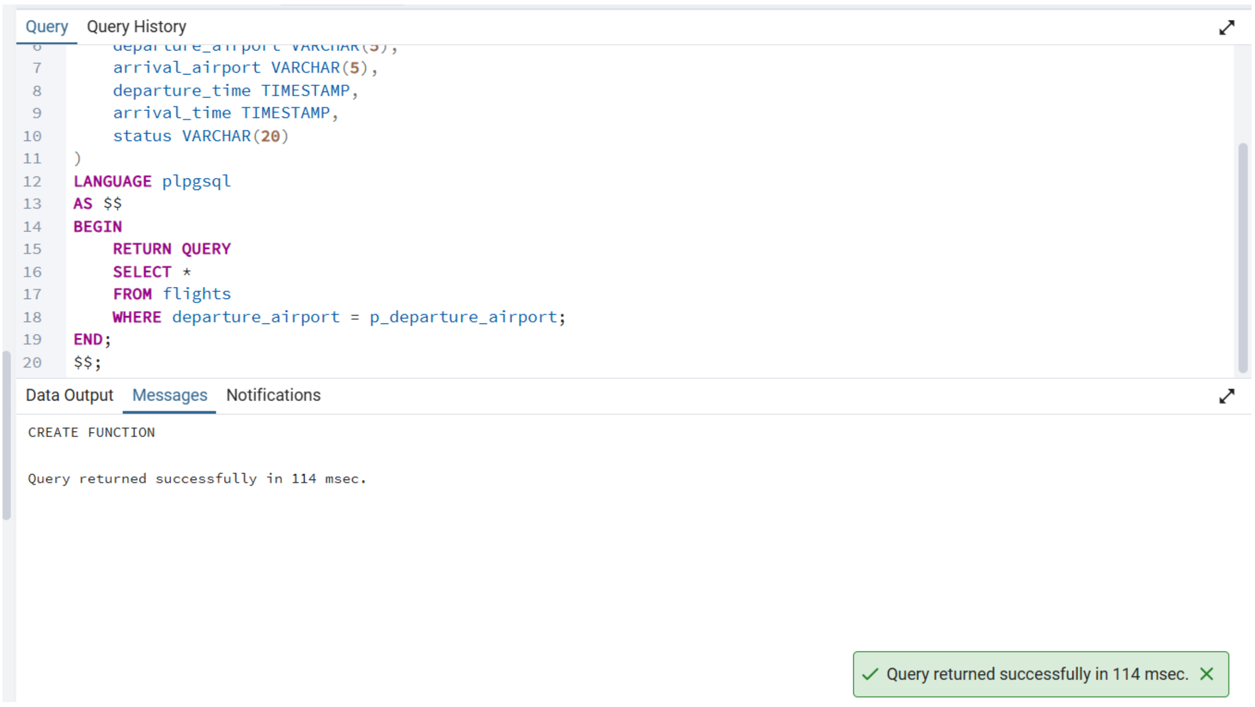
LANGUAGE plpgsql

AS $$

BEGIN

```
    RETURN QUERY
    SELECT *
    FROM flights
    WHERE departure_airport = p_departure_airport;
END;
$$;
```

SELECT * FROM GetFlightsByDeparture('SVO');



```
    arrival_airport VARCHAR(5),
    departure_time TIMESTAMP,
    arrival_time TIMESTAMP,
    status VARCHAR(20)
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT *
    FROM flights
    WHERE departure_airport = p_departure_airport;
END;
$$;
```

```
CREATE FUNCTION

Query returned successfully in 114 msec.
```

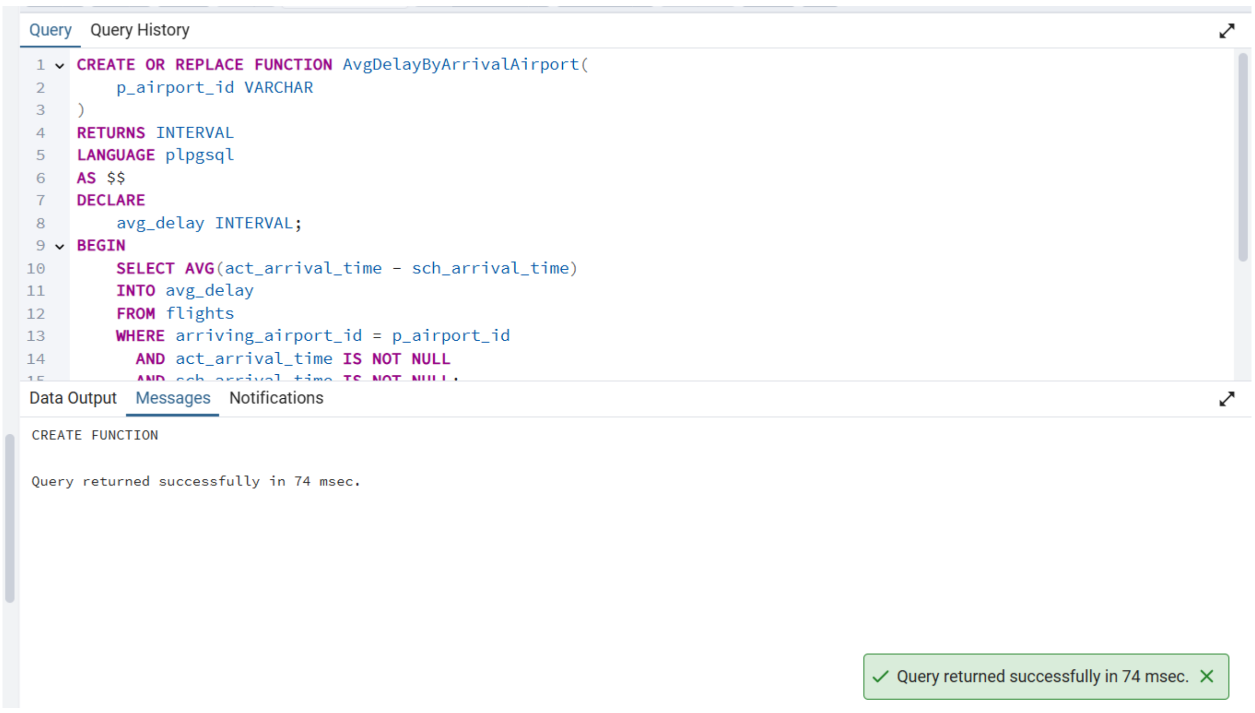
Query returned successfully in 114 msec.

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

```
CREATE OR REPLACE FUNCTION AvgDelayByArrivalAirport(
    p_airport_id INT
)
RETURNS INTERVAL
LANGUAGE plpgsql
AS $$
DECLARE
    avg_delay INTERVAL;
```

```
BEGIN
    SELECT AVG(act_arrival_time - sch_arrival_time)
    INTO 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 avg_delay;
END;
$$;
```

SELECT AvgDelayByArrivalAirport(55)



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

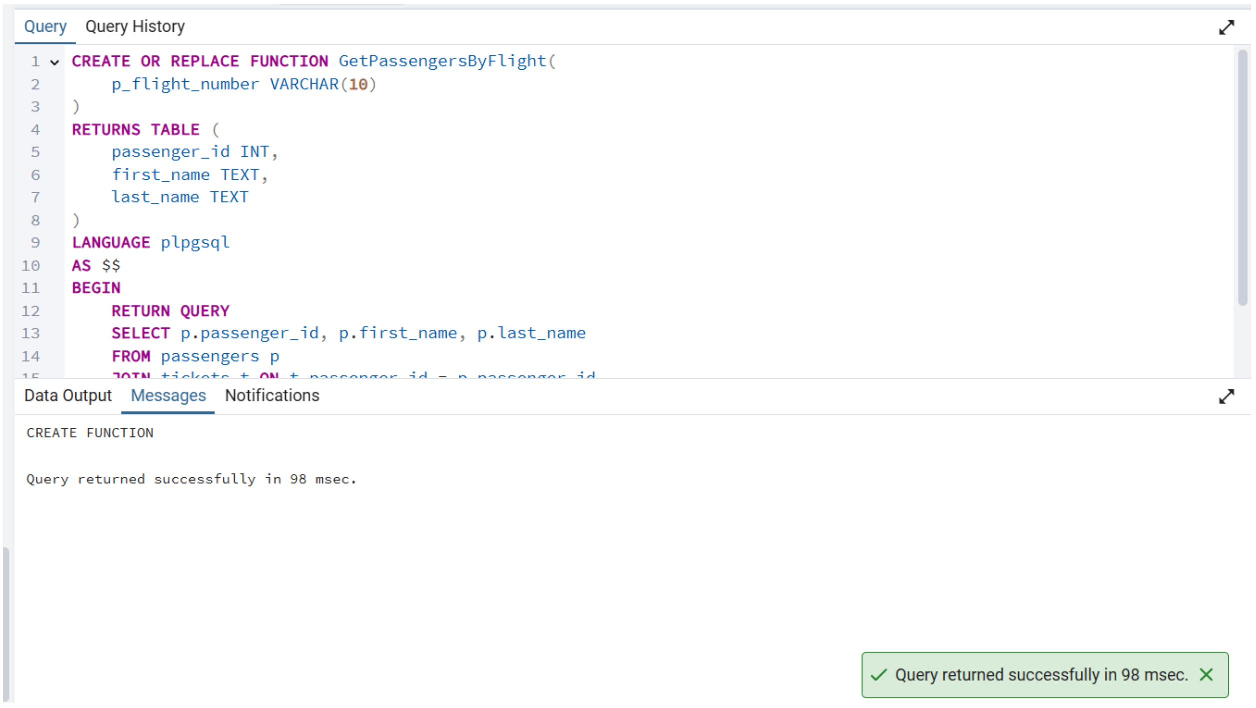
```
CREATE FUNCTION

Query returned successfully in 74 msec.
```

Query returned successfully in 74 msec.

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

```
CREATE OR REPLACE FUNCTION GetPassengersByFlight(
    p_flight_number VARCHAR(10)
)
RETURNS TABLE (
    passenger_id INT,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT p.passenger_id, p.first_name, p.last_name
    FROM passengers p
    JOIN tickets t ON t.passenger_id = p.passenger_id
    WHERE t.flight_number = p_flight_number;
END;
$$;
```

SELECT * FROM GetPassengersByFlight('SU1005');



```
CREATE OR REPLACE FUNCTION GetPassengersByFlight(
    p_flight_number VARCHAR(10)
)
RETURNS TABLE (
    passenger_id INT,
    first_name TEXT,
    last_name TEXT
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT p.passenger_id, p.first_name, p.last_name
    FROM passengers p
    JOIN tickets t ON t.passenger_id = p.passenger_id
```

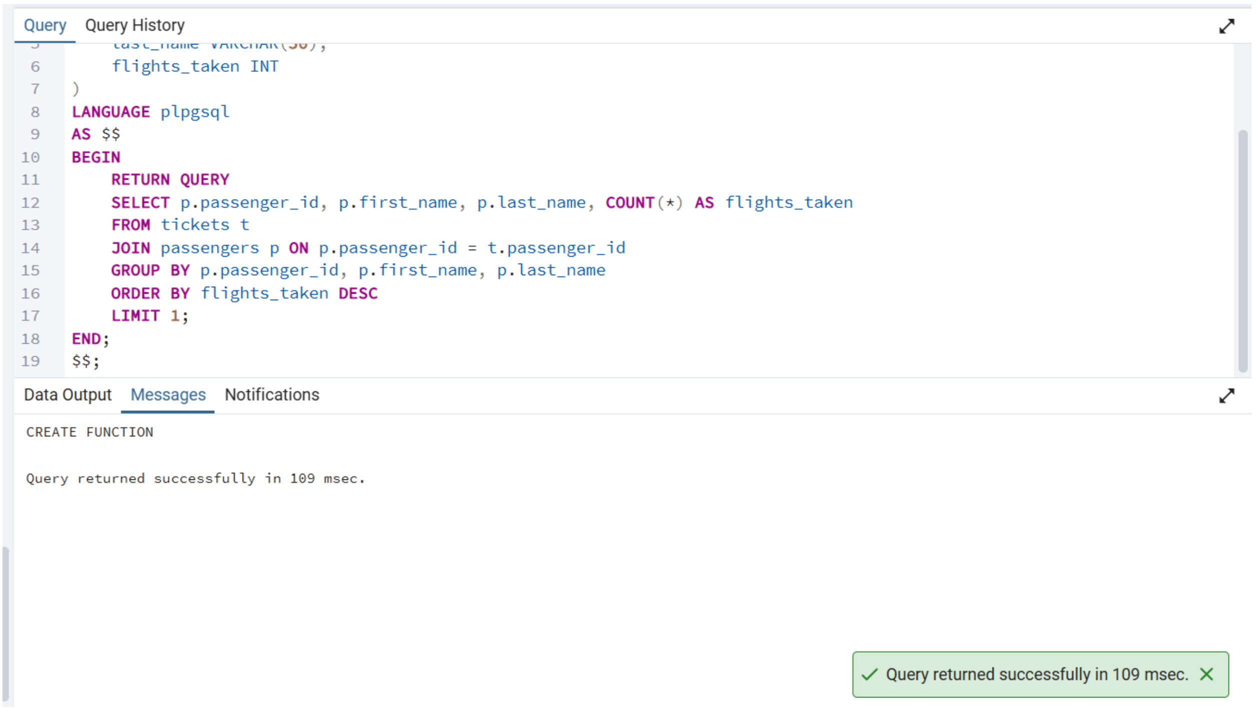
```
CREATE FUNCTION

Query returned successfully in 98 msec.
```

✓ Query returned successfully in 98 msec.

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

```
CREATE OR REPLACE FUNCTION GetTopPassenger()

RETURNS TABLE (

    passenger_id INT,

    first_name VARCHAR(50),

    last_name VARCHAR(50),

    flights_taken BIGINT

)

LANGUAGE plpgsql

AS $$

BEGIN

    RETURN QUERY

    SELECT p.passenger_id,

           p.first_name,

           p.last_name,

           COUNT(*) AS flights_taken

    FROM tickets t

    JOIN passengers p ON p.passenger_id = t.passenger_id

    GROUP BY p.passenger_id, p.first_name, p.last_name

    ORDER BY flights_taken DESC

    LIMIT 1;

END;

$$;
```

SELECT GetTopPassenger()


Query Query History

```
    last_name VARCHAR(50),
    flights_taken INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT p.passenger_id, p.first_name, p.last_name, COUNT(*) AS flights_taken
    FROM tickets t
    JOIN passengers p ON p.passenger_id = t.passenger_id
    GROUP BY p.passenger_id, p.first_name, p.last_name
    ORDER BY flights_taken DESC
    LIMIT 1;
END;
$$;
```

Data Output Messages Notifications

```
CREATE FUNCTION

Query returned successfully in 109 msec.
```

✓ Query returned successfully in 109 msec. ✕

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

CREATE OR REPLACE FUNCTION GetFlightsDelayed24h()

RETURNS TABLE (

    flight_id INT,

    delay_interval INTERVAL,

    delay_minutes NUMERIC

)

LANGUAGE plpgsql

AS $$

BEGIN

    RETURN QUERY

    SELECT

        f.flight_id,

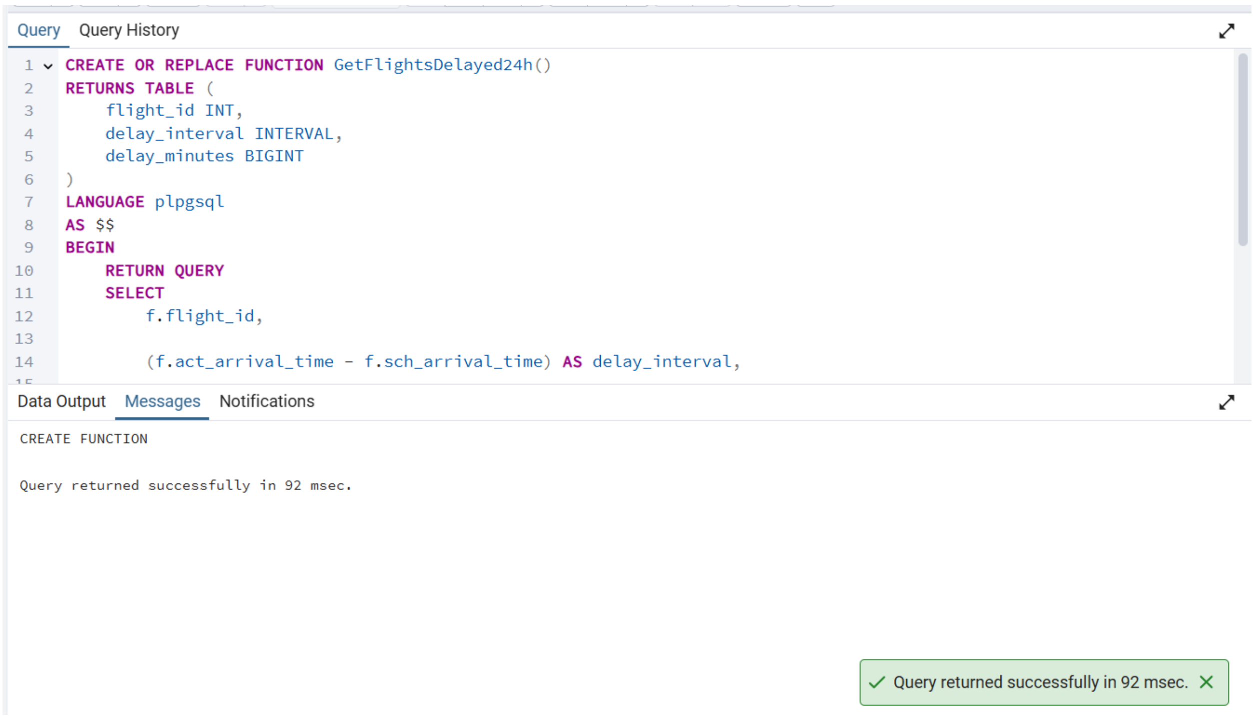
        (f.act_arrival_time - f.sch_arrival_time) AS delay_interval,

        EXTRACT(EPOCH FROM (f.act_arrival_time - f.sch_arrival_time)) / 60 AS delay_minutes

    FROM flights f

```
    WHERE f.act_arrival_time IS NOT NULL

      AND f.sch_arrival_time IS NOT NULL

      AND (EXTRACT(EPOCH FROM (f.act_arrival_time - f.sch_arrival_time)) / 60) > 1440;

END;

$$;
```

SELECT * FROM GetFlightsDelayed24h();



```
CREATE OR REPLACE FUNCTION GetFlightsDelayed24h()
RETURNS TABLE (
    flight_id INT,
    delay_interval INTERVAL,
    delay_minutes BIGINT
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        f.flight_id,

        (f.act_arrival_time - f.sch_arrival_time) AS delay_interval,
```

```
CREATE FUNCTION

Query returned successfully in 92 msec.
```

Query returned successfully in 92 msec.

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

```
CREATE OR REPLACE FUNCTION CountFlightsByAirline(

    p_airline_id INT

)

RETURNS BIGINT

LANGUAGE plpgsql

AS $$

DECLARE

    cnt BIGINT;

BEGIN

    SELECT COUNT(flight_id)
```

INTO cnt

FROM flights

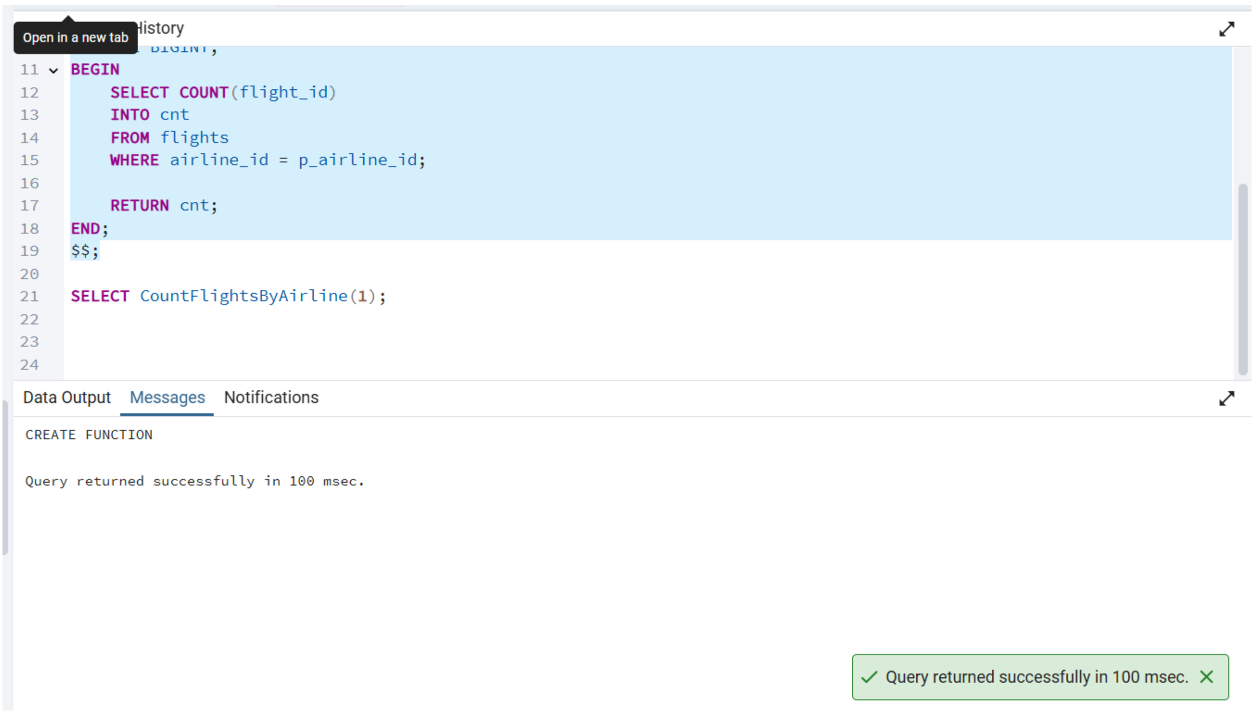
WHERE airline_id = p_airline_id;

RETURN cnt;

END;

$$;

SELECT CountFlightsByAirline(1);




```
    BIGINT;
BEGIN
    SELECT COUNT(flight_id)
    INTO cnt
    FROM flights
    WHERE airline_id = p_airline_id;

    RETURN cnt;
END;
$$;

SELECT CountFlightsByAirline(1);
```

```
CREATE FUNCTION

Query returned successfully in 100 msec.
```

Query returned successfully in 100 msec.

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

CREATE OR REPLACE FUNCTION AvgTicketPriceByFlight(

p_flight_number VARCHAR(50)

)

RETURNS NUMERIC

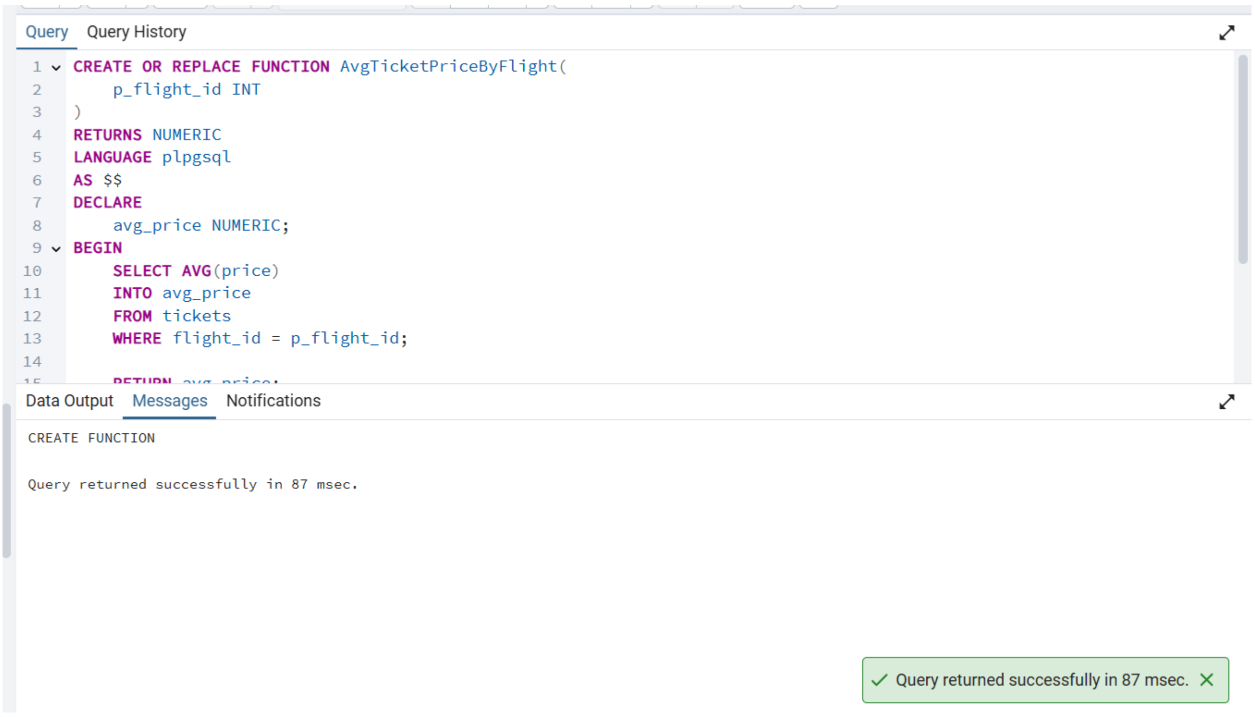
LANGUAGE plpgsql

AS $$

DECLARE

```
    avg_price NUMERIC;

BEGIN

    SELECT AVG(price)

    INTO avg_price

    FROM tickets

    WHERE flight_number = p_flight_number;



    RETURN avg_price;

END;

$$;
```

SELECT AvgTicketPriceByFlight('SU1005');



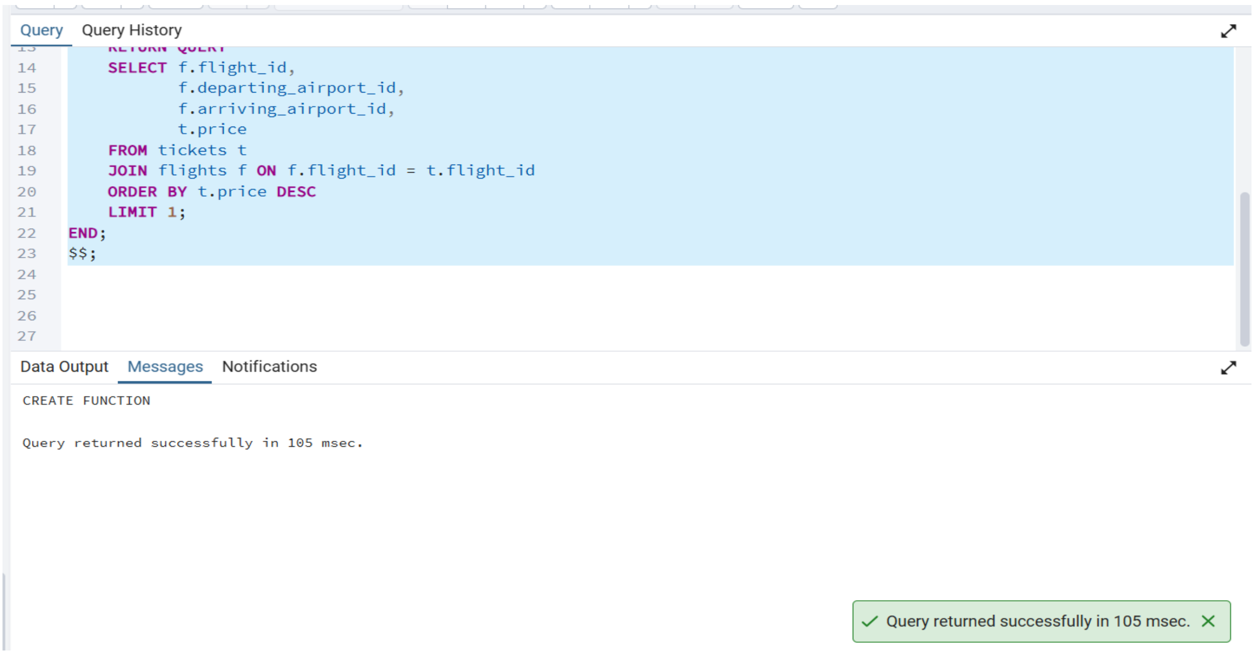
```
CREATE OR REPLACE FUNCTION AvgTicketPriceByFlight(
    p_flight_id INT
)
RETURNS NUMERIC
LANGUAGE plpgsql
AS $$
DECLARE
    avg_price NUMERIC;
BEGIN
    SELECT AVG(price)
    INTO avg_price
    FROM tickets
    WHERE flight_id = p_flight_id;

    RETURN avg_price;
```

```
CREATE FUNCTION

Query returned successfully in 87 msec.
```

Query returned successfully in 87 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 FUNCTION GetMostExpensiveFlight()

RETURNS TABLE (

    flight_id INT,
    departing_airport_id VARCHAR(50),
    arriving_airport_id VARCHAR(50),
    price NUMERIC
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT f.flight_id,
           f.departing_airport_id,
           f.arriving_airport_id,
           t.price
    FROM tickets t
    JOIN flights f ON f.flight_id = t.flight_id
    ORDER BY t.price DESC
    LIMIT 1;
END;
$$;
```



```
    SELECT f.flight_id,
           f.departing_airport_id,
           f.arriving_airport_id,
           t.price
    FROM tickets t
    JOIN flights f ON f.flight_id = t.flight_id
    ORDER BY t.price DESC
    LIMIT 1;
END;
$$;
```

```
CREATE FUNCTION

Query returned successfully in 105 msec.
```

Query returned successfully in 105 msec.