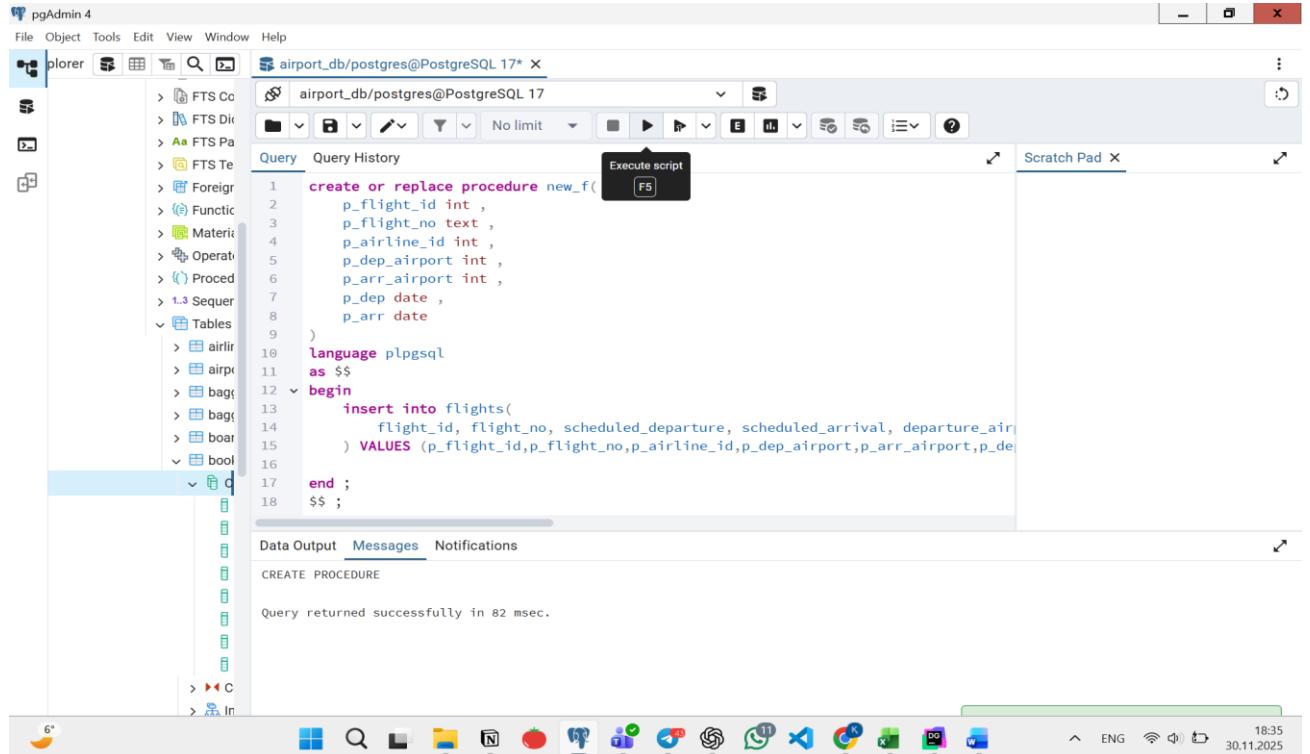


## Lab 10

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



The screenshot shows the pgAdmin 4 interface with the 'Query' tab selected. The query window contains the following PL/pgSQL code:

```
create or replace procedure new_f(
    p_flight_id int ,
    p_flight_no text ,
    p_airline_id int ,
    p_dep_airport int ,
    p_arr_airport int ,
    p_dep date ,
    p_arr date
)
language plpgsql
as $$

begin
    insert into flights(
        flight_id, flight_no, scheduled_departure, scheduled_arrival, departure_airport, arrival_airport
    ) VALUES (p_flight_id,p_flight_no,p_airline_id,p_dep_airport,p_arr_airport,p_dep);

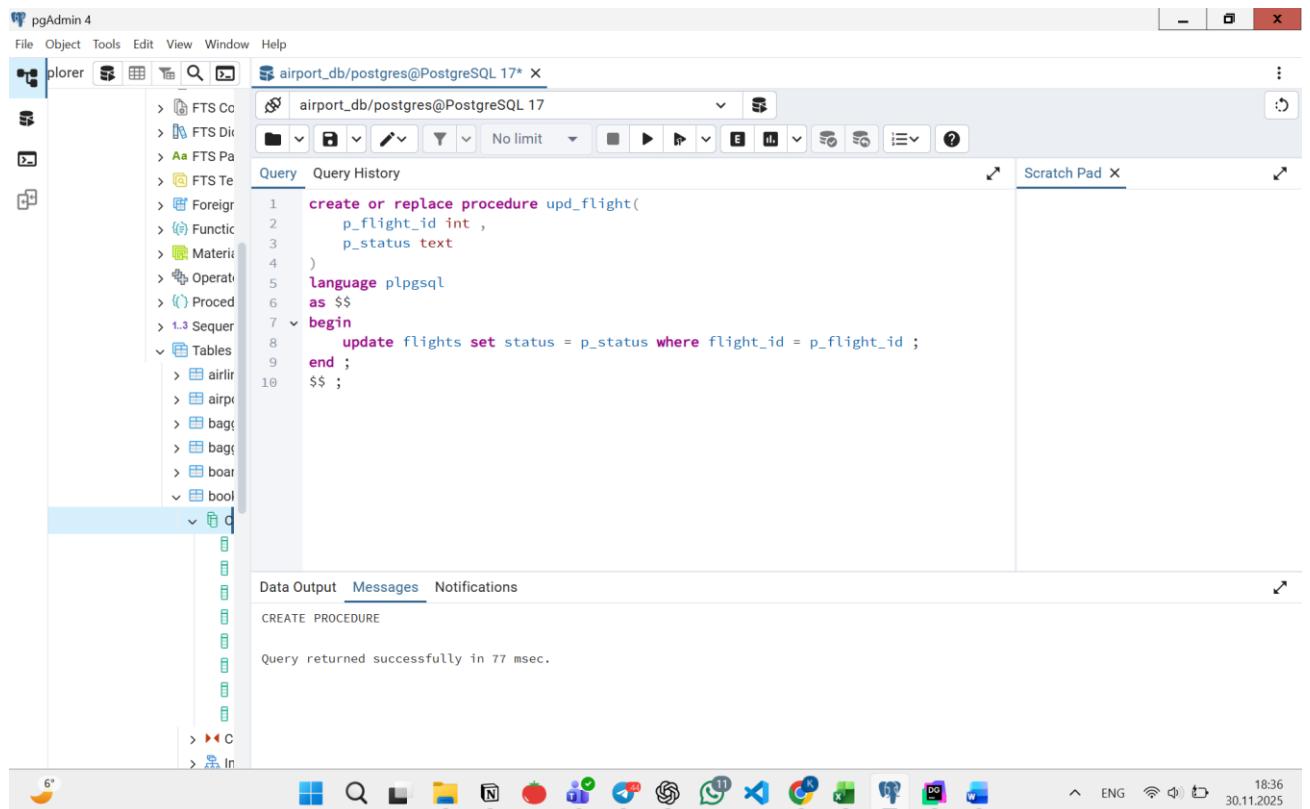
end ;
$$ ;
```

The 'Messages' tab below the query window shows the output:

CREATE PROCEDURE

Query returned successfully in 82 msec.

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



The screenshot shows the pgAdmin 4 interface with the 'Query' tab selected. The query window contains the following PL/pgSQL code:

```
create or replace procedure upd_flight(
    p_flight_id int ,
    p_status text
)
language plpgsql
as $$

begin
    update flights set status = p_status where flight_id = p_flight_id ;
end ;
$$ ;
```

The 'Messages' tab below the query window shows the output:

CREATE PROCEDURE

Query returned successfully in 77 msec.

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

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects under 'Tables'. The main window has two tabs: 'Query' and 'Query History'. The 'Query' tab contains the following SQL code:

```
1 create or replace procedure list_flights(
2     p_airport_id int
3 )
4 language plpgsql
5 as $$
6 begin
7     select *
8     from flights
9     where departure_airport_id = p_airport_id ;
10 end ;
11 $$;
```

Below the query, the 'Messages' tab shows the execution results:

CREATE PROCEDURE

Query returned successfully in 80 msec.

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

pgAdmin 4

File Object Tools Edit View Window Help

Explorer    Query    airport\_db/postgres@PostgreSQL 17\*

```

> FTS Cc
> FTS Dic
> Aa FTS Pa
> FTS Te
> Foreign
> Functc
> Materi
> Operat
> Proced
> 1.3 Sequer
> Tables
> airlin
> airp
> bag
> bagg
> boar
> bool
> C
> In

```

Query    Query History

```

1 create or replace function avg_delay_for_airport(p_airport_id int)
2 returns interval
3 language plpgsql
4 as $$*
5 declare
6     result interval;
7 begin
8     select avg(actual_arrival - scheduled_arrival)
9     into result
10    from flights
11   where arrival_airport_id = p_airport_id;
12
13    return result;
14 end;
$$;

```

Data Output    Messages    Notifications

CREATE FUNCTION

Query returned successfully in 87 msec.

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

pgAdmin 4

File Object Tools Edit View Window Help

Explorer    Query    airport\_db/postgres@PostgreSQL 17\*

```

> FTS Cc
> FTS Dic
> Aa FTS Pa
> FTS Te
> Foreign
> Functc
> Materi
> Operat
> Proced
> 1.3 Sequer
> Tables
> airlin
> airp
> bag
> bagg
> boar
> bool
> C
> In

```

Query    Query History

```

1 create or replace procedure get_passengers(p_flight_no text)
2 language plpgsql
3 as $$*
4 begin
5     select p.*
6     from passengers p
7     join booking b using (passenger_id)
8     join booking_flight bf using(bookign_id)
9     join flights f using (flight_id)
10    where f.flight_id = p_flight_no ;
11
12 end;
$$ ;

```

Data Output    Messages    Notifications

CREATE PROCEDURE

Query returned successfully in 76 msec.

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

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables' section, there is a table named 'top'. The main query editor window contains the following SQL code:

```
1 create or replace procedure top()
2 language plpgsql
3 as $$
4 begin
5     select p.passenger_id, p.first_name, p.last_name, count(bf.flight_id) as total
6         from passengers p
7     join booking b using (passenger_id)
8     join booking_flight bf using (booking_id)
9     group by p.passenger_id, p.first_name, p.last_name
10    order by total desc
11    limit 1 ;
12 end;
13 $$ ;
```

The 'Messages' tab at the bottom of the query editor shows the output:

```
CREATE PROCEDURE
```

Query returned successfully in 80 msec.

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

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables' section, there is a table named 'flights'. The main query editor window contains the following SQL code:

```
1 create or replace procedure long_delayed()
2 language plpgsql
3 as $$
4 begin
5     select *
6         from flights
7     where actual_departure - scheduled_departure > interval '1 day';
8 end;
9 $$ ;
```

The 'Messages' tab at the bottom of the query editor shows the output:

```
CREATE PROCEDURE
```

Query returned successfully in 73 msec.

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

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables' section, there is a table named 'flights'. In the main query editor, a function named 'flights\_per\_airline' is being created. The code is as follows:

```
1 create or replace function flights_per_airline(p_airline int)
2 returns int
3 language plpgsql
4 as $$
5 declare
6     result int ;
7 begin
8     select count(*)
9         into result
10        from flights
11       where airline_id = p_airline ;
12
13    return result ;
14 end;
15 $$;
```

The 'Messages' tab at the bottom shows the output of the command:

```
CREATE FUNCTION
```

Query returned successfully in 83 msec.

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

The screenshot shows the pgAdmin 4 interface. In the left sidebar, under the 'Tables' section, there is a table named 'booking'. In the main query editor, a procedure named 'avg\_price' is being created. The code is as follows:

```
1 create or replace procedure avg_price(p_flight_id int)
2 language plpgsql
3 as $$
4 begin
5     select avg(price)
6         from booking b
7        join booking_flight bf using (booking_id)
8       where bf.flight_id= p_flight_id ;
9 end;
10 $$;
```

The 'Messages' tab at the bottom shows the output of the command:

```
CREATE PROCEDURE
```

Query returned successfully in 79 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

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects under 'Tables'. The main window has two tabs: 'Query' and 'Scratch Pad'. The 'Query' tab contains the following PostgreSQL code:

```
1 create or replace procedure most_expensive()
2 language plpgsql
3 as $$
4 begin
5     select f.flight_no,f.departure_airport_id,f.arrival_airport_id,max(b.price) as max_price
6     from flights f
7     join booking_flight bf using (flight_id)
8     join booking b using(booking_id)
9     group by f.flight_no, f.departure_airport_id, f.arrival_airport_id
10    order by max_price desc
11    limit 1;
12 end;
13 $$ ;
```

Below the code, the 'Messages' tab is active, showing the output:

CREATE PROCEDURE

Query returned successfully in 83 msec.