

## Laboratory work 10

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

The screenshot shows the SQL Developer interface. On the left, the Database Explorer shows the 'public' schema with a table named 'flights'. The main editor displays the following PL/SQL code:

```
1 CREATE OR REPLACE PROCEDURE ins_new_flight(in flight_id int, in flight_no varchar(50), in scheduled_departure date,  
2 in scheduled_arrival date, in departure_airport_id int, in arrival_airport_id int, in departing_gate varchar(50),  
3 in arriving_gate varchar(50), in airline_id int, in status varchar(50), in actual_departure date, in actual_arrival date,  
4 in created_at date, in update_at date)  
5 language plpgsql as $$  
6 BEGIN  
7 INSERT INTO flights(flight_id, flight_no, scheduled_departure, scheduled_arrival, departure_airport_id, arrival_airport_id, departing_gate,  
8 arriving_gate, airline_id, status, actual_departure, actual_arrival, created_at, update_at)  
9 VALUES ( flight_id, flight_no, scheduled_departure, scheduled_departure, scheduled_arrival, scheduled_arrival, departing_gate, arriving_gate,  
10 airline_id, airline_id, status, status, actual_departure, actual_arrival, created_at, update_at );  
11 END;  
12 $$;
```

The Services pane at the bottom shows the execution of 'console\_3' with a message: '[2024-11-29 23:40:39] completed in 30 ms'.

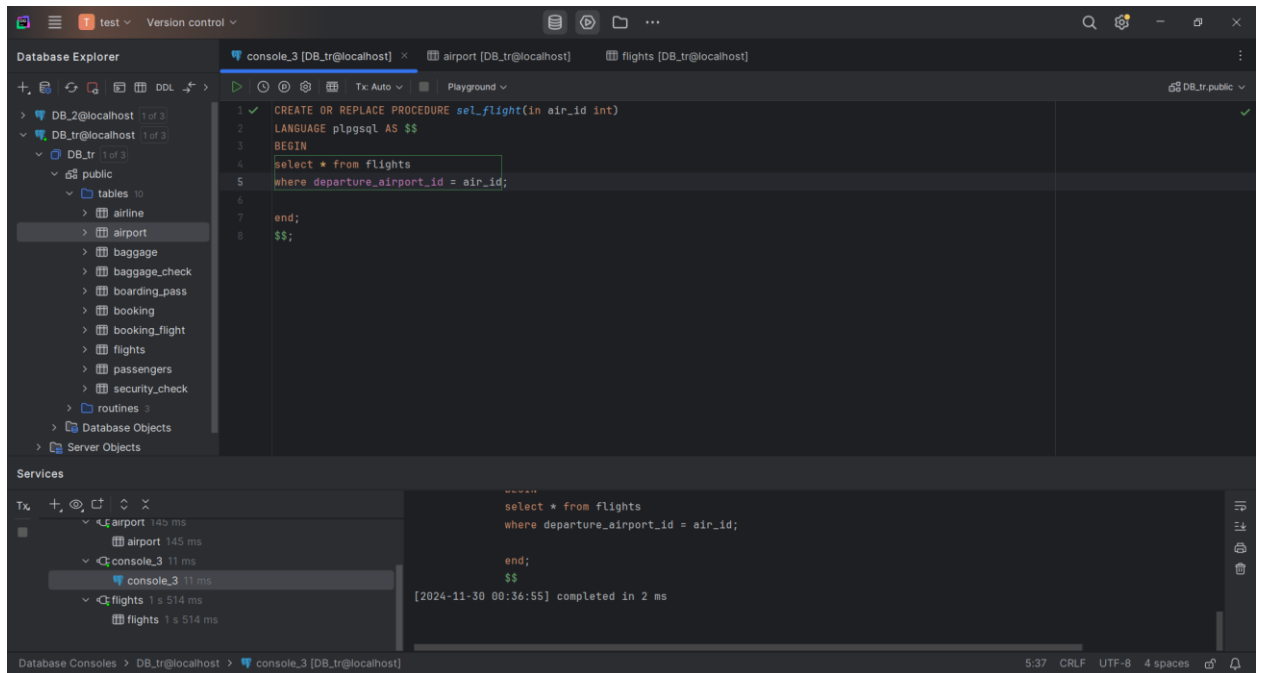
2. Create a stored procedure to update the status of a flight.

The screenshot shows the SQL Developer interface. The main editor displays the following PL/SQL code:

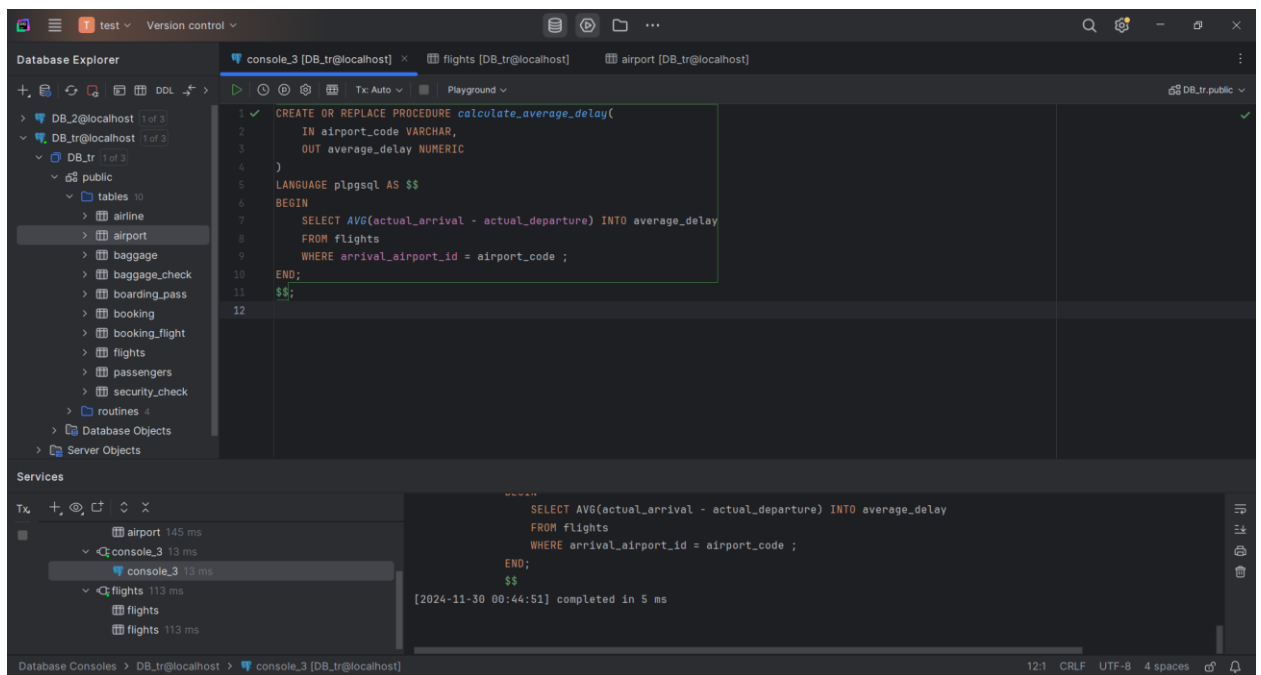
```
1 CREATE OR REPLACE PROCEDURE up_flight_status(in flight_num int, in new_status varchar(50))  
2 LANGUAGE plpgsql AS $$  
3 BEGIN  
4 update flights  
5 set status = new_status  
6 where flight_no = flight_num;  
7 end;  
8 $$;
```

The Services pane at the bottom shows the execution of 'console\_3' with a message: '[2024-11-29 23:44:38] completed in 8 ms'.

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



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



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

The screenshot shows a database IDE with a SQL query in the console. The query is a stored procedure named `sel_pas_spec_flight_num` that takes a flight number as input and returns the passenger's full name who has taken that flight. The query uses joins to connect the `passengers`, `booking`, `booking_flight`, and `flights` tables.

```

1 CREATE OR REPLACE PROCEDURE sel_pas_spec_flight_num(in flight_num varchar(50))
2 LANGUAGE plpgsql AS $$
3 BEGIN
4     SELECT p.first_name || ' ' || p.last_name as full_name from passengers p
5     join booking b 1<->1 on b.passenger_id = p.passenger_id
6     join booking_flight bf 1<->1 on bf.booking_id = b.booking_id
7     join flights f 1<->1 on f.flight_id = bf.flight_id
8     where flight_no = flight_num ;
9
10
11
12 END;
13 $$;
14

```

The output window shows the results of the query, displaying the full names of the passengers who have taken the specified flight:

Full Name
Alphonse Philippou
Elfrida Schukert

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

The screenshot shows a database IDE with a SQL query in the console. The query is a stored procedure named `top_pas` that returns the passenger's full name and the count of flights they have taken, ordered by the count in descending order.

```

1 CREATE OR REPLACE PROCEDURE top_pas(
2 out pass_name text,
3 out pass_count int
4 )
5 language plpgsql as $$
6 begin
7     select p.first_name || ' ' || p.last_name as full_name, count(f.flight_id) into pass_name, pass_count
8     from passengers p
9     join booking b 1<->1 on b.passenger_id = p.passenger_id
10    join booking_flight bf 1<->1 on bf.booking_id = b.booking_id
11    join flights f 1<->1 on f.flight_id = bf.flight_id
12    group by p.first_name, p.last_name
13    order by count(f.flight_id) desc limit 1;
14
15
16
17 end;
18
19 $$;
20

```

The output window shows the results of the query, displaying the full name of the passenger who has taken the greatest number of flights:

Full Name	Pass Count
Alphonse Philippou	2
Elfrida Schukert	1

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

The screenshot shows a database IDE with a 'Database Explorer' on the left and a 'console\_3 [DB\_tr@localhost]' window in the center. The Explorer shows a tree structure with 'DB\_tr' expanded, containing 'tables' and 'routines'. The 'routines' folder is selected, showing a list of routines including 'find\_delayed\_flights'. The console window displays the following SQL code:

```

1 CREATE OR REPLACE PROCEDURE find_delayed_flights(
2   OUT delayed_flights TEXT
3 )
4 LANGUAGE plpgsql AS $$
5 BEGIN
6   SELECT array_agg(flight_no) INTO delayed_flights
7   FROM flights
8   WHERE (actual_arrival - actual_departure) > 24;
9 END;
10 $$;

```

Below the console, the 'Services' panel shows a list of transactions. The 'Output' panel displays the results of the procedure:

p.first_name    ' '    p.last_name: text
Alphonso Philippou
Elfrida Schukert

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

The screenshot shows a database IDE with a 'Database Explorer' on the left and a 'console\_5 [DB\_tr@localhost]' window in the center. The Explorer shows a tree structure with 'DB\_tr' expanded, containing 'tables' and 'routines'. The 'routines' folder is selected, showing a list of routines including 'count\_flights\_by\_airline'. The console window displays the following SQL code:

```

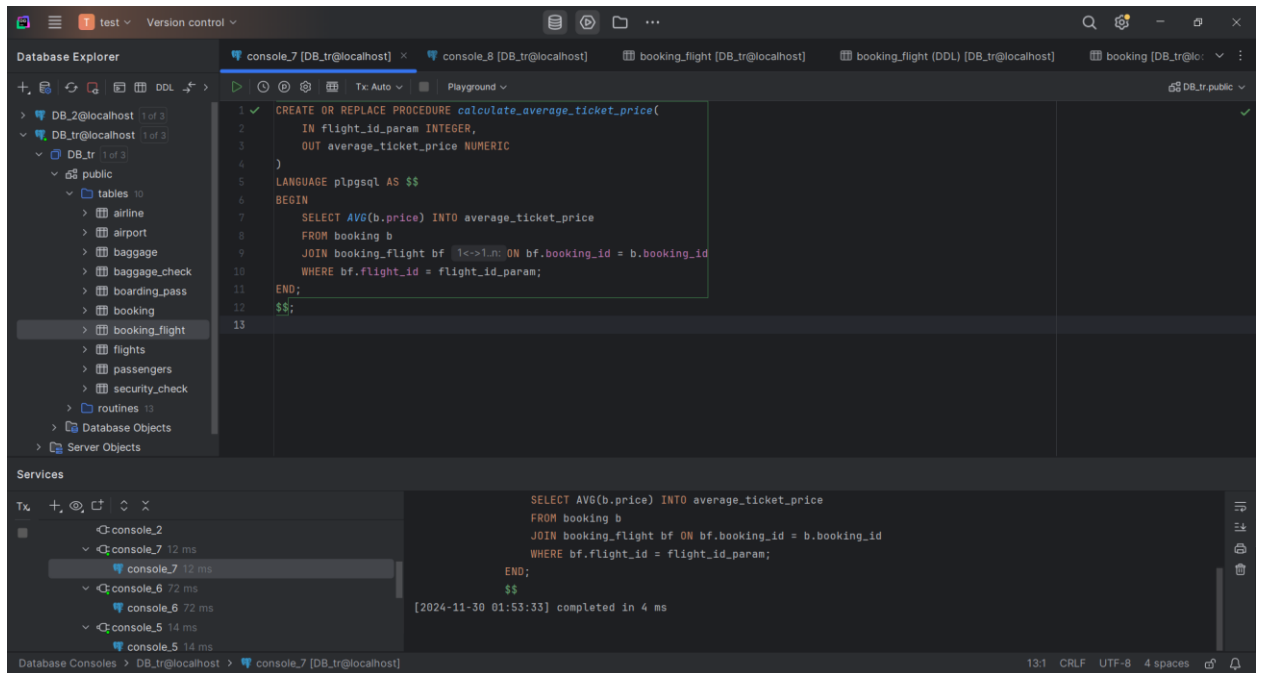
1 CREATE OR REPLACE PROCEDURE count_flights_by_airline()
2 LANGUAGE plpgsql AS $$
3 DECLARE
4   rec RECORD;
5 BEGIN
6   FOR rec IN
7     SELECT a.airline_name, COUNT(f.flight_id) AS flight_count
8     FROM flights f
9     JOIN airline a ON a.airline_id = f.airline_id
10    GROUP BY a.airline_name
11    ORDER BY flight_count DESC
12  LOOP
13    RAISE NOTICE 'Airline: %, Flights: %', rec.airline_name, rec.flight_count;
14  END LOOP;
15 END;
16 $$;

```

Below the console, the 'Services' panel shows a list of transactions. The 'Output' panel displays the results of the procedure:

air_name	air_count
IPC	33

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



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.

