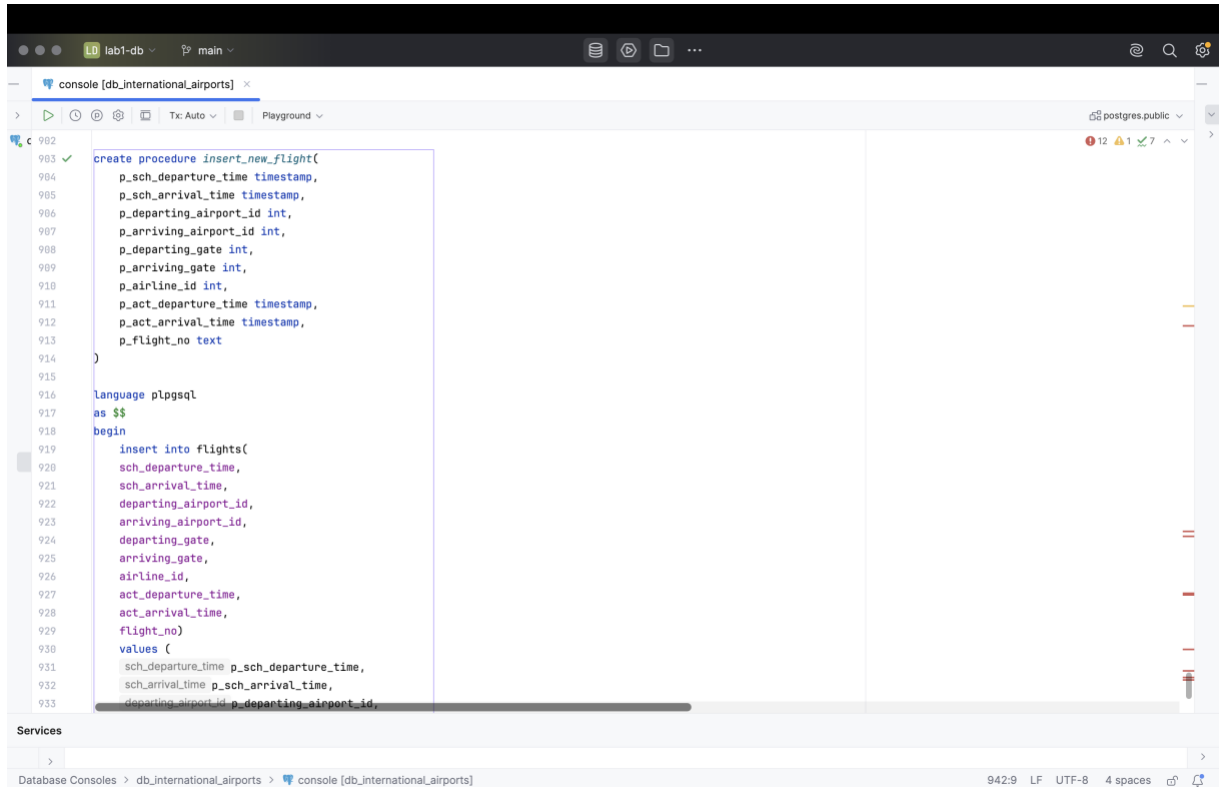


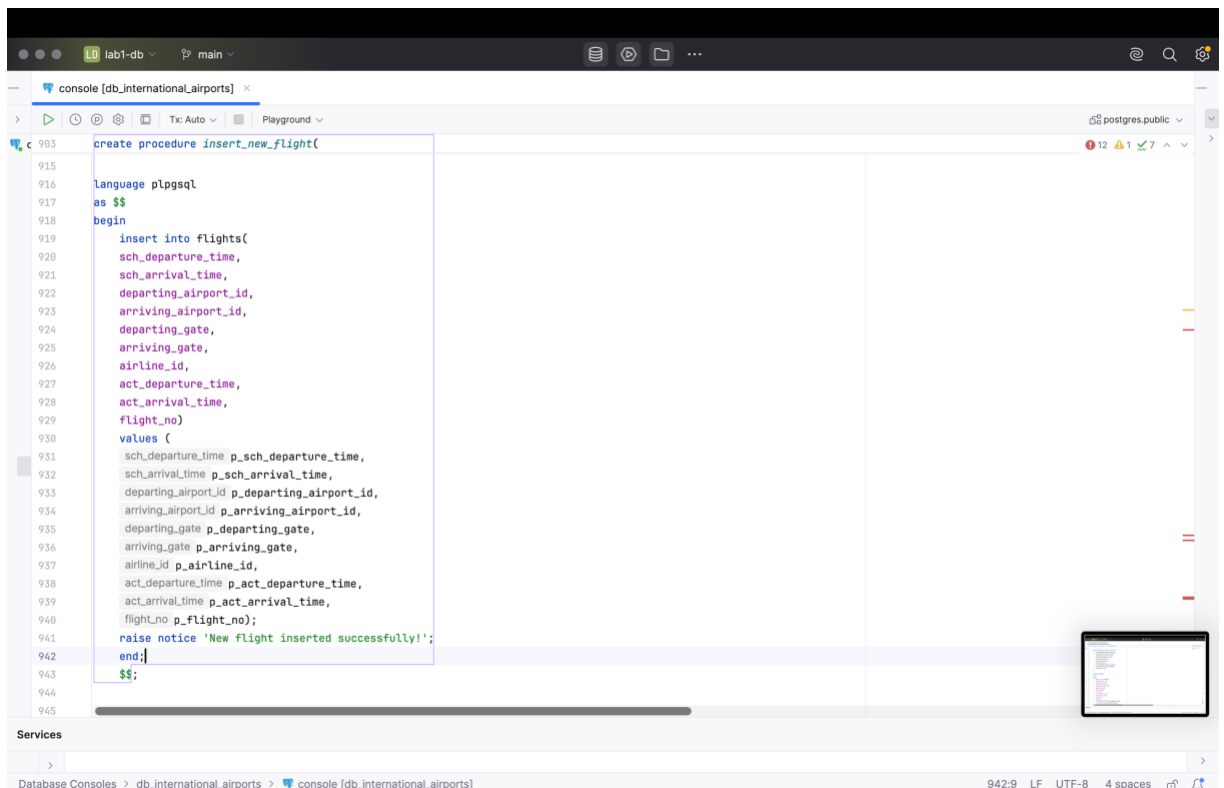
Lab 10

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



The screenshot shows a database console window titled "console [db_international_airports]". The code being entered is a PL/pgSQL stored procedure named "insert_new_flight". The procedure takes several parameters: p_sch_departure_time (timestamp), p_sch_arrival_time (timestamp), p_departing_airport_id (int), p_arriving_airport_id (int), p_departing_gate (int), p_arriving_gate (int), p_airline_id (int), p_act_departure_time (timestamp), p_act_arrival_time (timestamp), and p_flight_no (text). The procedure body starts with "begin" and contains an "insert into flights" statement with columns: sch_departure_time, sch_arrival_time, departing_airport_id, arriving_airport_id, departing_gate, arriving_gate, airline_id, act_departure_time, act_arrival_time, and flight_no. The values are provided in a "values (" block, with the first three values being p_sch_departure_time, p_sch_arrival_time, and p_departing_airport_id. The code is currently at line 933, with the next line being "departing_airport_id p_departing_airport_id,".

```
902
903 ✓ create procedure insert_new_flight(
904     p_sch_departure_time timestamp,
905     p_sch_arrival_time timestamp,
906     p_departing_airport_id int,
907     p_arriving_airport_id int,
908     p_departing_gate int,
909     p_arriving_gate int,
910     p_airline_id int,
911     p_act_departure_time timestamp,
912     p_act_arrival_time timestamp,
913     p_flight_no text
914 )
915
916 language plpgsql
917 as $$
918 begin
919     insert into flights(
920         sch_departure_time,
921         sch_arrival_time,
922         departing_airport_id,
923         arriving_airport_id,
924         departing_gate,
925         arriving_gate,
926         airline_id,
927         act_departure_time,
928         act_arrival_time,
929         flight_no)
930     values (
931         sch_departure_time p_sch_departure_time,
932         sch_arrival_time p_sch_arrival_time,
933         departing_airport_id p_departing_airport_id,
```



The screenshot shows the same database console window, but the code is now complete. The "insert_new_flight" procedure is defined with the same parameters as in the first screenshot. The "values (" block lists all the parameters: 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, and p_flight_no. The procedure ends with "end;" and "\$\$;". The code is now at line 945.

```
903 create procedure insert_new_flight(
915
916 language plpgsql
917 as $$
918 begin
919     insert into flights(
920         sch_departure_time,
921         sch_arrival_time,
922         departing_airport_id,
923         arriving_airport_id,
924         departing_gate,
925         arriving_gate,
926         airline_id,
927         act_departure_time,
928         act_arrival_time,
929         flight_no)
930     values (
931         sch_departure_time p_sch_departure_time,
932         sch_arrival_time p_sch_arrival_time,
933         departing_airport_id p_departing_airport_id,
934         arriving_airport_id p_arriving_airport_id,
935         departing_gate p_departing_gate,
936         arriving_gate p_arriving_gate,
937         airline_id p_airline_id,
938         act_departure_time p_act_departure_time,
939         act_arrival_time p_act_arrival_time,
940         flight_no p_flight_no);
941     raise notice 'New flight inserted successfully!';
942 end;
943 $$;
944
945
```

lab1-db main

console [db_international_airports] flights [db_international_airports]

postgres.public

```
call insert_new_flight(
  p_sch_departure_time '2025-12-01 13:22:47.000000',
  p_sch_arrival_time '2025-12-01 15:22:47.000000',
  p_departing_airport_id 367,
  p_arriving_airport_id 287,
  p_departing_gate 13,
  p_arriving_gate 5,
  p_airline_id 45,
  p_act_departure_time '2025-12-01 13:43:47.000000',
  p_act_arrival_time '2025-12-01 15:53:47.000000',
  p_flight_no 'F5643'
);
```

Services

13,
5,
45,
'2025-12-01 13:43:47.000000',
'2025-12-01 15:53:47.000000',
'F5643'
)

New flight inserted successfully!
[2025-12-03 01:18:17] completed in 31 ms

Database Consoles > db_international_airports > console [db_international_airports] 961:8 LF UTF-8 4 spaces

lab1-db main

console [db_international_airports] flights [db_international_airports]

postgres.public

```
call insert_new_flight(
  p_sch_departure_time '2025-12-01 13:22:47.000000',
  p_sch_arrival_time '2025-12-01 15:22:47.000000',
  p_departing_airport_id 367,
  p_arriving_airport_id 287,
  p_departing_gate 13,
  p_arriving_gate 5,
  p_airline_id 45,
  p_act_departure_time '2025-12-01 13:43:47.000000',
  p_act_arrival_time '2025-12-01 15:53:47.000000',
  p_flight_no 'F5643'
);
```

```
select* from flights where sch_departure_time = '2025-12-01 13:22:47.000000';
```

Services

Output postgres.public.flights

	sch_arrival_time	departing_airport_id	arriving_airport_id	departing_gate	arriving_gate	airline_id	act_departure
1	2025-12-01 15:22:47.000000	367	287	13	5	45	2025-12-01 13:43

1 row

Database Consoles > db_international_airports > console [db_international_airports] 963:78 LF UTF-8 4 spaces

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

The screenshot shows a database console interface with a SQL editor and a results pane. The SQL editor contains the following code:

```
965
966
967
968 ✓ create procedure update_status_flight(
969     p_flight_id int,
970     p_new_status varchar
971 )
972 language plpgsql
973 as $$
974 begin
975     update booking
976     set status = p_new_status
977     where flight_id = p_flight_id;
978
979     if not found then
980     raise exception 'No bookings found for flight_id = %', p_flight_id;
981     end if;
982 end;
983 $$;
984
```

The results pane shows a table with the following data:

	sch_arrival_time	departing_airport_id	arriving_airport_id	departing_gate	arriving_gate	airline_id	act_departure
1	2025-12-01 15:22:47.000000	367	287	13	5	45	2025-12-01 13:43

The screenshot shows the same database console interface. The SQL editor contains the following code:

```
968 create procedure update_status_flight(
973 as $$
974 begin
975     update booking
976     set status = p_new_status
977     where flight_id = p_flight_id;
978
979     if not found then
980     raise exception 'No bookings found for flight_id = %', p_flight_id;
981     end if;
982 end;
983 $$;
984
985 call update_status_flight(
986     p_flight_id 248,
987     p_new_status 'DELAYED'
988 );
989
990 ✓ select* from booking where flight_id = 248;
991
992
```

The results pane shows a table with the following data:

	booking_id	flight_id	passenger_id	booking_platform	created_at	updated_at	status	ticket
1	341	248	332	GreenLam	2025-09-30 23:51:36.708526	2025-09-30 23:51:36.708526	DELAYED	

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

The image shows a PostgreSQL console interface with two panels. The top panel displays the SQL code for creating a stored procedure named `get_flights_by_airport`. The bottom panel shows the execution of this procedure, resulting in a list of flights.

SQL Code (Top Panel):

```
create or replace procedure get_flights_by_airport(
    p_airport_id int
)
language plpgsql
as $$
declare
    r record;
begin
    for r in(
        select flight_id,
               flight_no,
               sch_departure_time,
               act_departure_time,
               sch_arrival_time,
               act_arrival_time,
               airline_id,
               created_at,
               updated_at,
               arriving_airport_id
        from flights
        where departing_airport_id = p_airport_id)
    loop
        raise notice 'Flight % | No: % | schDepTime: % | actDepTime: % | schArrTime: % | actArrTime: % | Airline: % | created at: % | upddated at: % | arrAirport: %',
            r.flight_no,
            r.sch_departure_time,
            r.act_departure_time,
            r.sch_arrival_time,
            r.act_arrival_time,
            r.airline_id,
            r.created_at,
            r.updated_at,
            r.arriving_airport_id;
    end loop;
end;
```

Execution Results (Bottom Panel):

The procedure was executed successfully, returning the following flight information:

Flight No	Scheduled Departure Time	Actual Departure Time	Scheduled Arrival Time	Actual Arrival Time	Airline
3	2007-05-07 19:10:38	2022-02-06 19:45:59	2025-05-21 13:22:47	2005-04-22 18:05:43	172
67	2002-09-18 12:36:52	2002-06-15 11:18:17	2004-11-27 13:44:52	2009-07-01 01:22:59	52
110	2023-05-15 14:04:17	2018-04-20 14:44:43	2010-11-17 13:50:28	2017-04-14 11:24:15	15

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

```
993 create or replace procedure get_flights_by_airport(  
1023     r.updated_at,  
1024     r.arriving_airport_id;  
1025     end loop;  
1026 end;  
1027 $$;  
1028  
1029  
1030  
1031 call get_flights_by_airport( p_airport_id 77);  
1032  
1033  
1034  
1035  
1036 ✓ create or replace function average_delay(arrival_airport int)  
1037 returns interval as  
1038 $$  
1039 declare  
1040     avg_delay interval;  
1041 begin  
1042     select avg(act_arrival_time - sch_arrival_time)  
1043     into avg_delay  
1044     from flights  
1045     where arriving_airport_id = arrival_airport;  
1046     return avg_delay;  
1047 end;  
1048 $$  
1049 language plpgsql;  
1050
```

```
1036 create or replace function average_delay(arrival_airport int)  
1043 into avg_delay  
1044 from flights  
1045 where arriving_airport_id = arrival_airport;  
1046 return avg_delay;  
1047 end;  
1048 $$  
1049 language plpgsql;  
1050  
1051  
1052  
1053 ✓ select average_delay( arrival_airport 1);  
1054
```

Services

Database Consoles > db_internationalAirports > console [db_internationalAirports]

1049:18 LF UTF-8 4 spaces

Output average_delay(1):interval

average_delay
1 0 years 0 mons -231 days -13 hours -37 mins -28.0 secs

1 row

Database Consoles > db_internationalAirports > console [db_internationalAirports]

1053:25 LF UTF-8 4 spaces

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

The screenshot displays a PostgreSQL console interface with two panels. The top panel shows the SQL code for creating a stored procedure, and the bottom panel shows the execution results.

SQL Code (Top Panel):

```
create or replace function average_delay(arrival_airport int)
$$
language plpgsql;

select average_delay(arrival_airport 1);

create procedure get_passengers_by_flights(
    p_flight_no varchar
)
language plpgsql
as $$
declare r record;
begin
    for r in
        select p.passenger_id,
               p.first_name,
               p.last_name
        from booking b join passengers p 1..n<->1: on p.passenger_id = b.passenger_id
        join flights f 1..n<->1: on f.flight_id = b.flight_id
        where f.flight_no = p_flight_no
    loop
        raise notice 'Passenger: % | First name: % | Last name: %', r.passenger_id, r.first_name, r.last_name;
    end loop;
end;
$$;
```

Execution Results (Bottom Panel):

```
[2025-12-03 02:24:01] completed in 4 ms
[2025-12-03 02:24:25] postgres.public> call get_passengers_by_flights(19)
[2025-12-03 02:24:25] [42883] ERROR: procedure get_passengers_by_flights(integer) does not exist
[2025-12-03 02:24:25] Hint: No procedure matches the given name and argument types. You might need to add explicit type casts.
[2025-12-03 02:24:25] Position: 6
[2025-12-03 02:24:55] postgres.public> call get_passengers_by_flights('AA9129')
Passenger: 50 | First name: Sam | Last name: Morten
Passenger: 110 | First name: Korella | Last name: Safont
Passenger: 174 | First name: Windham | Last name: Rehn
Passenger: 34 | First name: Lurleen | Last name: Arzu
Passenger: 45 | First name: Court | Last name: Portman
Passenger: 141 | First name: Maighdiln | Last name: Abbatini
Passenger: 128 | First name: Rickey | Last name: Calderon
Passenger: 123 | First name: Emmalee | Last name: Carling
Passenger: 319 | First name: Beatrix | Last name: Rudram
Passenger: 12 | First name: Denni | Last name: Liston
[2025-12-03 02:24:55] completed in 20 ms
```

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

The image shows two screenshots of a PostgreSQL console interface. The top screenshot displays the creation of a stored procedure named `max_flight_passenger()`. The procedure is written in PL/pgSQL and uses a `SELECT` statement to find the passenger with the highest number of flights, ordered by `flights_count` in descending order and limited to 1 result. A `raise notice` statement is used to output the passenger's ID, name, and total flights. The bottom screenshot shows the execution of the `max_flight_passenger()` procedure, which returns the following result:

```
[2025-12-03 02:32:22] postgres.public> create procedure max_flight_passenger()
language plpgsql
as $$
    declare r record;
    begin
        select p.passenger_id, p.first_name, p.last_name, count(b.booking_id) as flights_count
        into r
        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 flights_count desc
        limit 1;

        raise notice 'Passenger: % | Name: %, % | Total flights: %', r.passenger_id, r.first_name, r.last_name, r.flights_count;
    end;
$$

[2025-12-03 02:32:22] completed in 11 ms
[2025-12-03 02:32:44] postgres.public> call max_flight_passenger()
Passenger: 153 | Name: Livvy, Dansken | Total flights: 7
[2025-12-03 02:32:44] completed in 10 ms
```

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

The image shows two screenshots of a database console interface, likely DBeaver, demonstrating the creation and execution of a PostgreSQL stored procedure.

Top Screenshot: Creating the stored procedure

The console shows the following SQL code being executed:

```
call max_flight_passenger();

create procedure delay_morethan_24h()
language plpgsql
as $$
declare r record;
begin
for r in
select flight_id, sch_arrival_time, act_arrival_time, act_arrival_time - sch_arrival_time as delay
from flights
where act_arrival_time - sch_arrival_time > interval '24 hours'
loop
raise notice 'Flight: % | schArrTime: % | actArrTime: % | Delay: % ', r.flight_id, r.sch_arrival_time, r.act_arrival_time, r.delay;
end loop;
end;
$$
```

The Services panel shows the execution of the first command:

```
[2025-12-03 02:32:22] postgres.public> create procedure max_flight_passenger()
language plpgsql
as $$
declare r record;
begin
select p.passenger_id, p.first_name, p.last_name, count(b.booking_id) as flights_count
into r
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 flights_count desc;
```

Bottom Screenshot: Executing the stored procedure

The console shows the following SQL code being executed:

```
create or replace procedure delay_morethan_24h()
$$;

call delay_morethan_24h();
```

The Services panel shows the execution of the second command, which completed in 4 ms, followed by the output of the stored procedure:

```
[2025-12-03 02:40:11] completed in 4 ms
[2025-12-03 02:40:24] postgres.public> call delay_morethan_24h()
Flight: 4 | schArrTime: 2002-03-30 22:46:52 | actArrTime: 2014-10-20 21:51:39 | Delay: 4586 days 23:04:47
Flight: 7 | schArrTime: 2003-04-12 18:50:18 | actArrTime: 2014-04-12 16:38:56 | Delay: 4017 days 21:48:38
Flight: 9 | schArrTime: 2012-03-23 17:59:28 | actArrTime: 2016-04-16 17:26:11 | Delay: 1484 days 23:26:43
Flight: 13 | schArrTime: 2005-12-21 12:07:13 | actArrTime: 2023-12-04 17:32:12 | Delay: 6557 days 05:24:59
Flight: 15 | schArrTime: 2005-09-24 08:30:03 | actArrTime: 2014-12-17 21:10:33 | Delay: 3371 days 12:40:30
Flight: 20 | schArrTime: 2008-06-22 04:20:38 | actArrTime: 2010-11-08 18:47:08 | Delay: 869 days 14:26:30
Flight: 21 | schArrTime: 2008-03-19 04:11:05 | actArrTime: 2017-07-22 18:55:25 | Delay: 3412 days 14:44:20
Flight: 24 | schArrTime: 2001-01-30 18:04:22 | actArrTime: 2008-09-13 11:19:20 | Delay: 2782 days 17:14:58
Flight: 28 | schArrTime: 2013-10-15 15:30:30 | actArrTime: 2025-05-19 02:51:42 | Delay: 4233 days 11:21:12
Flight: 29 | schArrTime: 2012-12-11 07:09:58 | actArrTime: 2023-04-21 04:48:30 | Delay: 3782 days 21:38:32
Flight: 30 | schArrTime: 2005-11-26 17:49:53 | actArrTime: 2006-03-22 09:58:15 | Delay: 115 days 16:08:22
Flight: 31 | schArrTime: 2008-03-06 23:02:02 | actArrTime: 2021-04-07 08:36:03 | Delay: 4779 days 09:34:01
Flight: 33 | schArrTime: 2013-02-01 05:38:43 | actArrTime: 2014-01-23 04:45:51 | Delay: 355 days 23:07:08
Flight: 35 | schArrTime: 2007-11-27 00:54:54 | actArrTime: 2009-07-16 19:31:58 | Delay: 597 days 18:37:04
Flight: 36 | schArrTime: 2012-08-02 23:04:43 | actArrTime: 2020-09-14 21:59:35 | Delay: 2964 days 22:54:52
Flight: 37 | schArrTime: 2017-05-22 01:38:55 | actArrTime: 2019-01-07 03:23:40 | Delay: 595 days 01:44:45
Flight: 40 | schArrTime: 2005-03-19 21:31:48 | actArrTime: 2019-01-11 13:15:45 | Delay: 5045 days 15:43:57
Flight: 41 | schArrTime: 2009-01-19 02:13:06 | actArrTime: 2011-12-28 13:24:45 | Delay: 1073 days 11:11:39
Flight: 44 | schArrTime: 2007-04-17 16:26:03 | actArrTime: 2023-10-15 23:46:46 | Delay: 6025 days 07:20:43
Flight: 45 | schArrTime: 2013-03-22 09:37:29 | actArrTime: 2017-09-03 00:10:04 | Delay: 1625 days 14:32:35
Flight: 47 | schArrTime: 2018-12-29 00:33:42 | actArrTime: 2023-06-13 16:33:10 | Delay: 1627 days 15:59:28
Flight: 50 | schArrTime: 2021-03-26 13:53:39 | actArrTime: 2023-08-27 18:44:47 | Delay: 884 days 04:51:08
Flight: 52 | schArrTime: 2011-10-13 13:56:20 | actArrTime: 2016-08-19 19:22:37 | Delay: 1772 days 05:26:17
```


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

```
1121
1122
1123
1124 ✓ create or replace function num_flight_for_airline()
1125 returns table(
1126     airline_id int,
1127     airline_name varchar,
1128     total_flights bigint
1129 )
1130 language plpgsql
1131 as $$
1132 begin
1133     return query
1134     select a.airline_id, a.airline_name, count(f.flight_id) as total_flights
1135     from airline a left join flights f on f.airline_id = a.airline_id
1136     group by a.airline_id, a.airline_name
1137     order by total_flights desc;
1138 end;
1139 $$;
1140
1141 select* from num_flight_for_airline();
1142
1143
```

Services

Tx > Output num_flight_for_airline...total_flights:bigint) x

num_flight_for_airline

airline_id	airline_name	total_flights
1	(144,"South African Aeroways",7)	

404 rows

Database Consoles > db_internationalairports > console [db_internationalairports] 1139:4 LF UTF-8 4 spaces

Services

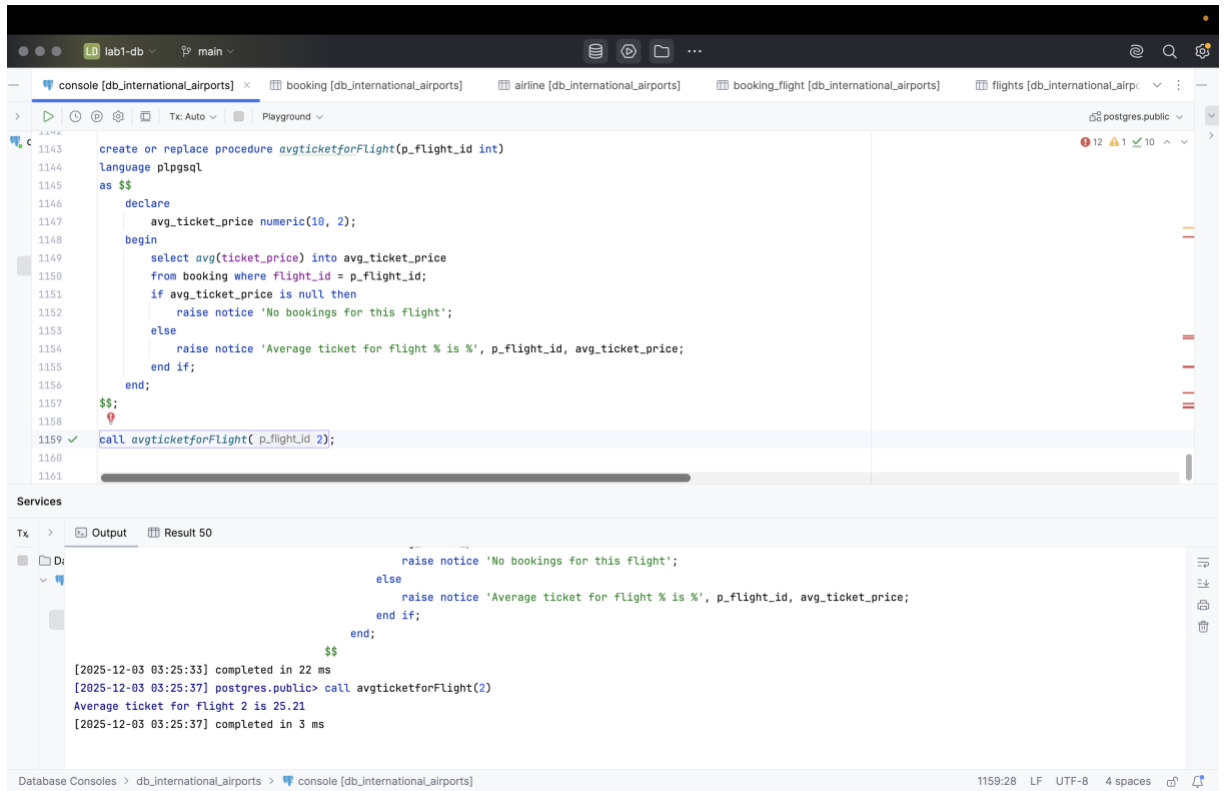
Tx > Output Result 50 x

airline_id	airline_name	total_flights
1	144 South African Aeroways	7
2	172 Avianca	5
3	135 Southwest Airlines	5
4	138 KLM Royal Dutch Airlines	4
5	113 Korean Aero	4
6	11 Southwest Airlines	4
7	37 Delta Aero Lines	4
8	158 United Airlines	4
9	6 All Nippon Aeroways	4
10	193 LATAM Airlines	4
11	63 Delta Aero Lines	4
12	103 Iberia	4
13	67 All Nippon Aeroways	4
14	98 Cathay Pacific	3
15	129 SAS Scandinavian	3
16	117 Korean Aero	3
17	366 All Nippon Aeroways	3
18	198 SAS Scandinavian	3
19	62 Korean Aero	

404 rows

Database Consoles > db_internationalairports > console [db_internationalairports] 1141:39 LF UTF-8 4 spaces

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



```
1143 create or replace procedure avgticketforFlight(p_flight_id int)
1144 language plpgsql
1145 as $$
1146 declare
1147     avg_ticket_price numeric(10, 2);
1148 begin
1149     select avg(ticket_price) into avg_ticket_price
1150     from booking where flight_id = p_flight_id;
1151     if avg_ticket_price is null then
1152         raise notice 'No bookings for this flight';
1153     else
1154         raise notice 'Average ticket for flight % is %', p_flight_id, avg_ticket_price;
1155     end if;
1156 end;
1157 $$;
1158
1159 call avgticketforFlight( p_flight_id 2);
1160
1161
```

Services

Tx > Output Result 50

```

        raise notice 'No bookings for this flight';
    else
        raise notice 'Average ticket for flight % is %', p_flight_id, avg_ticket_price;
    end if;
end;
$$
[2025-12-03 03:25:33] completed in 22 ms
[2025-12-03 03:25:37] postgres.public> call avgticketforFlight(2)
Average ticket for flight 2 is 25.21
[2025-12-03 03:25:37] completed in 3 ms
```

Database Consoles > db_international_airports > console [db_international_airports] 1159:28 LF UTF-8 4 spaces

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 a PostgreSQL console interface with a PL/pgSQL procedure named `flight_with_max_ticketprice()`. The procedure is designed to find the flight with the highest ticket price by joining the `flights`, `booking`, and `airport` tables. It uses a `SELECT` statement with `JOIN` clauses and a `GROUP BY` clause to identify the flight with the maximum ticket price. The procedure also includes a `RAISE NOTICE` statement to output the result.

```
1163
1164 create procedure flight_with_max_ticketprice()
1165 language plpgsql
1166 as $$
1167     declare flight_info record;
1168     begin
1169         select f.flight_no,
1170                dep.airport_name as departure_airport,
1171                arr.airport_name as arrival_airport,
1172                max(b.ticket_price) as max_ticket_price
1173     into flight_info
1174     from flights f join booking b on f.flight_id = b.flight_id
1175     join airport dep on f.departing_airport_id = dep.airport_id
1176     join airport arr on f.arriving_airport_id = arr.airport_id
1177     group by f.flight_no, dep.airport_name, arr.airport_name
1178     order by max_ticket_price desc
1179     limit 1;
1180
1181     raise notice 'Most expensive flight: % from % to %, ticket price: %', flight_info.flight_no, flight_info.departure_airport, flight_info.arrival_airport, flight_info.max_ticket_price;
1182 end;
1183 $$
1184
1185 call flight_with_max_ticketprice();
```

The console output shows the procedure was executed successfully, returning the following notice:

```
[2025-12-03 03:37:07] postgres.public> call flight_with_max_ticketprice()
Most expensive flight: <NULL> from Cork Airport to La Désirade Airport, ticket price: 5059.99
[2025-12-03 03:37:07] completed in 32 ms
```