

Lab 10

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

The screenshot shows the pgAdmin 4 interface with a database console titled "console [db_international_airports]". The code editor contains the following PostgreSQL stored procedure definition:

```
create procedure insert_new_flight(
    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,
    p_flight_no text
)
language plpgsql
as $$begin
    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,
        flight_no)
    values (
        sch_departure_time p_sch_departure_time,
        sch_arrival_time p_sch_arrival_time,
        departing_airport_id p_departing_airport_id,
```

The code is highlighted in blue and green, indicating syntax. The status bar at the bottom right shows "942:9 LF UTF-8 4 spaces".

The screenshot shows the pgAdmin 4 interface with the same database console. The stored procedure definition has been completed with a "raise notice" statement:

```
create procedure insert_new_flight(
    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,
    p_flight_no text
)
language plpgsql
as $$begin
    insert into flights(
        sch_departure_time,
        sch_arrival_time,
        departing_airport_id,
        arriving_airport_id,
        departing_gate,
        arriving_gate,
        airline_id,
        act_departure_time,
        act_arrival_time,
        flight_no)
    values (
        sch_departure_time p_sch_departure_time,
        sch_arrival_time p_sch_arrival_time,
        departing_airport_id p_departing_airport_id,
        arriving_airport_id p_arriving_airport_id,
        departing_gate p_departing_gate,
        arriving_gate p_arriving_gate,
        airline_id p_airline_id,
        act_departure_time p_act_departure_time,
        act_arrival_time p_act_arrival_time,
        flight_no p_flight_no);
    raise notice 'New flight inserted successfully!';
end;
$$;
```

A small window in the bottom right corner shows a preview of the code. The status bar at the bottom right shows "942:9 LF UTF-8 4 spaces".

Lab1-db main

console [db_international_airports] flights [db_international_airports]

```
946
947
948
949
950 ✓ call insert_new_flight(
951     p_sch_departure_time '2025-12-01 13:22:47.000000',
952     p_sch_arrival_time '2025-12-01 15:22:47.000000',
953     p_departing_airport_id 367,
954     p_arriving_airport_id 287,
955     p_departing_gate 13,
956     p_arriving_gate 5,
957     p_airline_id 45,
958     p_act_departure_time '2025-12-01 13:43:47.000000',
959     p_act_arrival_time '2025-12-01 15:53:47.000000',
960     p_flight_no 'F5643'
961 );
962
```

Services

Tx > Di

```
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]

```
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963 ✓ select* from flights where sch_departure_time = '2025-12-01 13:22:47.000000';
964
```

Services

Tx > Output postgres.public.flights

Output

	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.

Screenshot of a PostgreSQL database interface showing the creation of a stored procedure and its execution.

Top Window (console [db_international_airports]):

```

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

```

Bottom Window (Output [postgres.public.flights]):

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

Second Window (console [db_international_airports]):

```

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

```

Bottom Window (Output [postgres.public.booking]):

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.

```

 993  create or replace procedure get_flights_by_airport(
 994      p_airport_id int
 995  )
 996  language plpgsql
 997  as $$
 998      declare
 999          r record;
1000      begin
1001          for r in(
1002              select flight_id,
1003                  flight_no,
1004                  sch_departure_time,
1005                  act_departure_time,
1006                  sch_arrival_time,
1007                  act_arrival_time,
1008                  airline_id,
1009                  created_at,
1010                  updated_at,
1011                  arriving_airport_id
1012          from flights
1013          where departing_airport_id = p_airport_id)
1014      loop
1015          raise notice 'Flight % | No: % | schDepTime: % | actDepTime: % | schArrTime: % | actArrTime: % | Airline: % | created at: % | updated at: % | arrAirport: %',
1016              r.flight_no,
1017              r.sch_departure_time,
1018              r.act_departure_time,
1019              r.sch_arrival_time,
1020              r.act_arrival_time,
1021              r.airline_id,
1022              r.created_at,
1023              r.updated_at,
1024              r.arriving_airport_id;
1025      end loop;
1026  end;
1027  $$;
1028
1029
1030
1031  call get_flights_by_airport( p_airport_id 77);
1032
1033
1034

```

The screenshot shows two code editors side-by-side. The top editor contains the full PL/pgSQL code for the stored procedure. The bottom editor shows the execution of the procedure with the parameter value 77, followed by the output of three flight records. The output includes flight details like flight number, departure and arrival times, airline, and creation/updating information, along with the arriving airport ID.

```

[2025-12-03 02:08:15] completed in 4 ms
[2025-12-03 02:08:19] postgres.public> call get_flights_by_airport(77)
Flight 3 | No: <NULL> | schDepTime: 2007-05-07 19:10:38 | actDepTime: 2022-02-06 19:45:59 | schArrTime: 2025-05-21 13:22:47 | actArrTime: 2005-04-22 18:05:43 | Airline: 172
Flight 67 | No: <NULL> | schDepTime: 2002-09-18 12:36:52 | actDepTime: 2002-06-15 11:18:17 | schArrTime: 2004-11-27 13:44:52 | actArrTime: 2009-07-01 01:22:59 | Airline: 52
Flight 110 | No: <NULL> | schDepTime: 2023-05-15 14:04:17 | actDepTime: 2018-04-20 14:44:43 | schArrTime: 2010-11-17 13:50:28 | actArrTime: 2017-04-14 11:24:15 | Airline: 15
[2025-12-03 02:08:19] completed in 6 ms

```

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

```

The screenshot shows the execution of the function and its output:

```

Tx > Output [average_delay(1):interval]
average_delay
1 0 years 0 mons -231 days -13 hours -37 mins -28.0 secs

```

Output: 1 row

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

```

 1036 create or replace function average_delay(arrival_airport int)
1048     $$language plpgsql;
1050
1051
1052
1053 select average_delay( arrival_airport 1);
1054
1055
1056
1057 ✓ 1057 create procedure get_passengers_by_flights(
1058     p_flight_no varchar
1059 )
1060 language plpgsql
1061 as $$*
1062     declare r record;
1063 begin
1064     for r in
1065         select p.passenger_id,
1066             p.first_name,
1067             p.last_name
1068     from booking b join passengers p 1..n->1: on p.passenger_id = b.passenger_id
1069     join flights f 1..n->1: on f.flight_id = b.flight_id
1070     where f.flight_no = p_flight_no
1071
1072     loop
1073         raise notice 'Passenger: % | First name: % | Last name: %', r.passenger_id, r.first_name, r.last_name;
1074     end loop;
1075 end;
1076
1077 $$;

```

```

 1057 create procedure get_passengers_by_flights(
1073     end loop;
1074     end;
1075     $$;
1076
1077
1078
1079 ✓ 1079 call get_passengers_by_flights( p_flight_no 'AA9129');

raise notice 'Passenger: % | First name: % | Last name: %', r.passenger_id, r.first_name, r.last_name;
end loop;
end;
$$

[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: Maighdin | 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.

```

 1057 create procedure get_passengers_by_flights()
 1058 $$
 1059
 1060
 1061
 1062
 1063
 1064 ✓ create procedure max_flight_passenger()
 1065 language plpgsql
 1066 as $$*
 1067     declare r record;
 1068     begin
 1069         select p.passenger_id, p.first_name, p.last_name, count(b.booking_id) as flights_count
 1070         into r
 1071         from passengers p join booking b 1<->1..n on p.passenger_id = b.passenger_id
 1072         group by p.passenger_id, p.first_name, p.last_name
 1073         order by flights_count desc
 1074         limit 1;
 1075
 1076         raise notice 'Passenger: % | Name: %, % | Total flights: %', r.passenger_id, r.first_name, r.last_name, r.flights_count;
 1077
 1078     end;
 1079
 1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1099
 1100
 1101
 1102

```



```

1094 create procedure max_flight_passenger()
1095
1096     raise notice 'Passenger: % | Name: %, % | Total flights: %', r.passenger_id, r.first_name, r.last_name, r.flights_count;
1097
1098 end;
1099
1100
1101
1102 ✓ call max_flight_passenger();

```



```

Tx > [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;

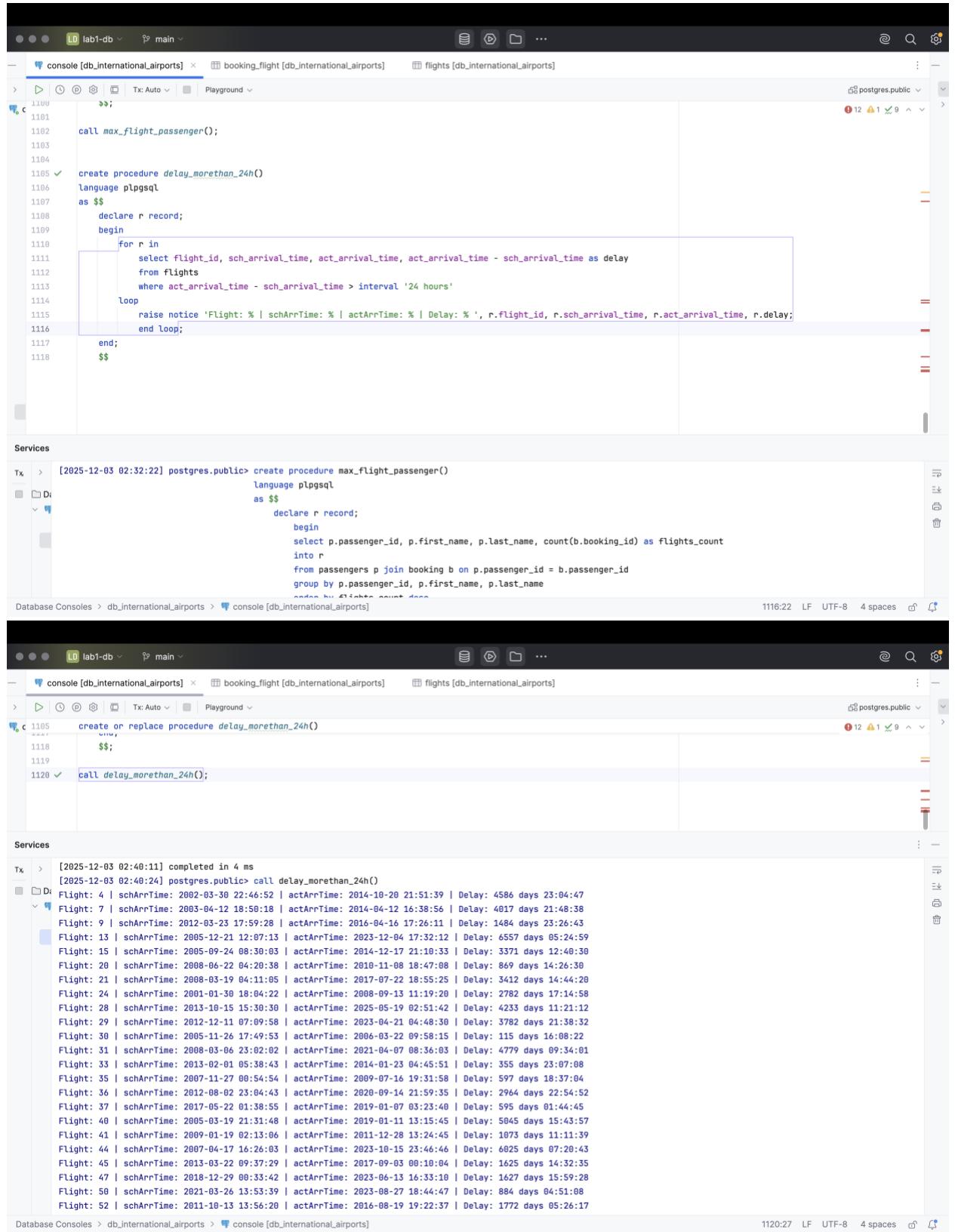
        raise notice 'Passenger: % | Name: %, % | Total flights: %', r.passenger_id, r.first_name, r.last_name, r.flights_count;

    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, Dansk | 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.



```

 1101
 1102     call max_flight_passenger();
 1103
 1104
 1105 ✓  create procedure delay_morethan_24h()
 1106     language plpgsql
 1107     as $$*
 1108         declare r record;
 1109         begin
 1110             for r in
 1111                 select flight_id, sch_arrival_time, act_arrival_time, act_arrival_time - sch_arrival_time as delay
 1112                 from flights
 1113                 where act_arrival_time - sch_arrival_time > interval '24 hours'
 1114             loop
 1115                 raise notice 'Flight: % | schArrTime: % | actArrTime: % | Delay: %', r.flight_id, r.sch_arrival_time, r.act_arrival_time, r.delay;
 1116             end loop;
 1117         end;
 1118     $$*

```

Services

```

Tx > [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

```

Database Consoles > db_international_airports > console [db_international_airports]

1116:22 LF UTF-8 4 spaces ↴


```

 1105 create or replace procedure delay_morethan_24h()
 1106     $$*
 1107
 1108
 1109 ✓  call delay_morethan_24h();

```

Services

```

Tx > [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: 2002-03-22 22:46:52 | actArrTime: 2014-10-22 21:51:39 | Delay: 4586 days 23:04:47
Flight: 7 | schArrTime: 2003-04-12 18:58:18 | 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 21:58:32
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

```

Database Consoles > db_international_airports > console [db_international_airports]

1120:27 LF UTF-8 4 spaces ↴

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

The screenshot shows a PostgreSQL IDE interface with two main panes. The top pane displays the SQL code for creating a function named `num_flight_for_airline`. The bottom pane shows the results of executing this function, which is a table listing various airlines and their total flight counts.

```

create or replace function num_flight_for_airline()
returns table(
    airline_id int,
    airline_name varchar,
    total_flights bigint
)
language plpgsql
as $$
begin
    return query
    select a.airline_id, a.airline_name, count(f.flight_id) as total_flights
    from airline a left join flights f 1<>0..n: on f.airline_id = a.airline_id
    group by a.airline_id, a.airline_name
    order by total_flights desc;
end;
$$;

select* from num_flight_for_airline();

```

Services

airline_id	airline_name	total_flights
1	South African Aeroways	7
2	Avianca	5
3	Southwest Aerolines	5
4	KLM Royal Dutch Aerolines	4
5	Korean Aero	4
6	Southwest Aerolines	4
7	Delta Aero Lines	4
8	United Aerolines	4
9	All Nippon Aeroways	4
10	LATAM Aerolines	4
11	Delta Aero Lines	4
12	Iberia	4
13	All Nippon Aeroways	4
14	Cathay Pacific	3
15	SAS Scandinavian	3
16	Korean Aero	3
17	All Nippon Aeroways	3
18	SAS Scandinavian	3
19	Korean Aero	3

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

The screenshot shows a PostgreSQL database console interface. The top bar indicates the connection is to 'lab1-db' and the schema is 'main'. The left sidebar lists several databases: 'console [db_international_airports]', 'booking [db_international_airports]', 'airline [db_international_airports]', 'booking_flight [db_international_airports]', 'flights [db_international_airports]', and 'postgres.public'. The main area contains the following SQL code:

```
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
```

The code at line 1159 is highlighted with a green checkmark. Below the code, the 'Services' section shows the transaction status and the output pane displays the results of the query:

```
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

The bottom status bar shows the time as 1159:28, the encoding as LF, the character set as UTF-8, and the number of spaces as 4.

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 database interface with a code editor and a terminal window. The code editor contains a PL/pgSQL stored procedure named `flight_with_max_ticketprice()`. The procedure selects the most expensive flight from the `booking` table, joining it with `flights` and `airport` tables to get departure and arrival airport names. It then groups by flight number and sorts by ticket price in descending order, limiting the result to one flight. A `raise notice` statement is included to output the results. The terminal window shows the procedure was created and a call to it resulted in a notice about the most expensive flight from Cork Airport to La Désirade Airport at a price of 5059.99.

```
1163
1164 create procedure flight_with_max_ticketprice()
1165 language plpgsql
1166 as $$
1167     declare flight_info record;
1168     begin
1169         select f.flight_no,
1170               dep.airport_name as departure_airport,
1171               arr.airport_name as arrival_airport,
1172               max(b.ticket_price) as max_ticket_price
1173         into flight_info
1174         from flights f join booking b 1<->1..n: on f.flight_id = b.flight_id
1175         join airport dep 1..n->1: on f.departing_airport_id = dep.airport_id
1176         join airport arr 1..n->1: on f.arriving_airport_id = arr.airport_id
1177         group by f.flight_no, dep.airport_name, arr.airport_name
1178         order by max_ticket_price desc
1179         limit 1;
1180
1181         raise notice 'Most expensive flight: % from % to %, ticket price: %', flight_info.flight_no, flight_info.departure_airport, flight_info.arrival_airport, flight_info.max_ticket_price;
1182     end;
1183
1184 $$;
1185
1186 call flight_with_max_ticketprice();
```

Services

Tx	Output	Result 50
[2025-12-03 03:30:51] completed in 17 ms		
D	[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	

Database Consoles > db_international_airports > console [db_international_airports]

1185:36 LF UTF-8 4 spaces ⌂ ⌂