

AIRLINE DATABASE QUERY USING SQL

Analyst: Oluoma Christian Chibuike (DigiTact Technology)

Database management tool: PostgreSQL 14

This project covers the entire process of learning SQL and also aims at the retrieval of relevant information from the database in the best and efficient manner.

The database is made up of eight (8) tables namely: airports, aircrafts, boarding_passes, bookings, flights, seats, tickets and ticket_flights. The following cases were acknowledged in facilitating the query;

CASE I:

List the cities in which there is no flights from Moscow.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under the 'Airlines' database, including servers, databases, and various schema objects like casts, catalogs, and extensions. The main area shows a query editor with the following SQL code:

```
1 SELECT DISTINCT city->> 'en'as city
2 FROM airports
3 WHERE city->> 'en'<> 'Moscow'
4 ORDER BY city;
```

The results pane shows a table with one column 'city' containing 100 rows of city names, ordered by their ID (90 to 100). The cities listed are Ust Ilimsk, Vladivostok, Volgograd, Vorkuta, Voronezh, Yakutia, Yakutsk, Yaroslavl, Yekaterinburg, Yoshkar-Ola, and Yuzhno-Sakhalinsk.

city
Ust Ilimsk
Vladivostok
Volgograd
Vorkuta
Voronezh
Yakutia
Yakutsk
Yaroslavl
Yekaterinburg
Yoshkar-Ola
Yuzhno-Sakhalinsk

CASE II:

Select airports with time zone in Asia / Novokuznetsk and Asia / Krasnoyarsk.

The screenshot shows the pgAdmin 4 interface with the 'Airlines' database selected. In the query editor, the following SQL code is run:

```
1 SELECT *
2 FROM airports
3 WHERE timezone IN ('Asia/Novokuznetsk', 'Asia/Krasnoyarsk');
```

The results are displayed in a table titled 'Data Output' with columns: airport_code, airport_name, city, coordinates, and timezone. The data includes:

airport_code	airport_name	city	coordinates	timezone
KEJ	'en': 'Kemerovo Airport', 'ru': 'Кемерово'	{en: 'Kemerovo', ru: 'Кемерово'}	(86.107200622558655, 27.00963989258)	Asia/Novokuznetsk
TOF	'en': 'Bogashovo Airport', 'ru': 'Богашево'	{en: 'Tomsk', ru: 'Томск'}	(85.20829772349256, 56.380298614502)	Asia/Krasnoyarsk
NSK	'en': 'Norilsk-Alykel Airport', 'ru': 'Норильск'	{en: 'Norilsk', ru: 'Норильск'}	(87.3321909667969, 69.31109619140625)	Asia/Krasnoyarsk
KJA	'en': 'Yemelyanovo Airport', 'ru': 'Емельяново'	{en: 'Krasnoyarsk', ru: 'Красноярск'}	(92.49330139160256, 56.172901153564)	Asia/Krasnoyarsk
KYZ	'en': 'Kyzyl Airport', 'ru': 'Кызыл'	{en: 'Kyzyl', ru: 'Кызыл'}	(94.4005966186523451, 66.939926147461)	Asia/Krasnoyarsk
NOZ	'en': 'Spichenkovo Airport', 'ru': 'Спиченково'	{en: 'Novokuznetsk', ru: 'Новокузнецк'}	(86.877197265625, 53.8114013671875)	Asia/Novokuznetsk
SWT	'en': 'Strezhnevoy Airport', 'ru': 'Стрежевой'	{en: 'Strezhnevoy', ru: 'Стрежевой'}	(77.66000366210001, 60.709400177)	Asia/Krasnoyarsk
RGK	'en': 'Gorno-Altaisk Airport', 'ru': 'Горно-Алтайск'	{en: 'Gorno-Altaisk', ru: 'Горно-Алтайск'}	(85.833297729551, 96.67015076)	Asia/Krasnoyarsk
ABA	'en': 'Abakan Airport', 'ru': 'Абакан'	{en: 'Abakan', ru: 'Абакан'}	(91.3850021362304753, 74.00016784668)	Asia/Krasnoyarsk
BAX	'en': 'Barnaul Airport', 'ru': 'Барнаул'	{en: 'Barnaul', ru: 'Барнаул'}	(83.5384979248046953, 53.363800048828125)	Asia/Krasnoyarsk

CASE III:

Which planes have a flight range from 3000 km to 6000 km?

The screenshot shows the pgAdmin 4 interface with the 'Airlines' database selected. In the query editor, the following SQL code is run:

```
1 SELECT *
2 FROM aircrafts
3 WHERE range BETWEEN 3000 AND 6000;
```

The results are displayed in a table titled 'Data Output' with columns: aircraft_code, model, and range. The data includes:

aircraft_code	model	range
SU9	'en': 'Sukhoi Superjet-100', 'ru': 'Сухой Суперджет-100'	3000
320	'en': 'Airbus A320-200', 'ru': 'Аэробус A320-200'	5700
321	'en': 'Airbus A321-200', 'ru': 'Аэробус A321-200'	5600
733	'en': 'Boeing 737-300', 'ru': 'Боинг 737-300'	4200

CASE IV:

Get the model, range and miles of every aircraft exist in the airlines database. Notice that miles = range/1.609 and round the result to 2 numbers after the float point.

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists the database structure under 'Airlines'. The 'Tables' section contains 'aircrafts'. The 'Query Editor' tab is active, displaying the following SQL query:

```
1 SELECT model, range, round(range/1.609,2)AS miles
2 FROM aircrafts
```

The 'Data Output' tab shows the results of the query:

model	range	miles
1 (en: 'Boeing 777-300', ru: 'Боинг 777-300')	11100	6898.69
2 (en: 'Boeing 767-300', ru: 'Боинг 767-300')	7900	4909.88
3 (en: 'Sukhoi Superjet-100', ru: 'Сухой Суперджет-100')	3000	1864.51
4 (en: 'Airbus A320-200', ru: 'Аэробус A320-200')	5700	3542.57
5 (en: 'Airbus A321-200', ru: 'Аэробус A321-200')	5600	3480.42
6 (en: 'Airbus A319-100', ru: 'Аэробус A319-100')	6700	4164.08
7 (en: 'Boeing 737-300', ru: 'Боинг 737-300')	4200	2610.32
8 (en: 'Cessna 208 Caravan', ru: 'Сессна 208 Караван')	1200	745.80
9 (en: 'Bombardier CRJ-200', ru: 'Бомбардье CRJ-200')	2700	1678.06

CASE V:

Calculate the Average ticket sales.

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists the database structure under 'Airlines'. The 'Tables' section contains 'bookings'. The 'Query Editor' tab is active, displaying the following SQL query:

```
1 SELECT AVG(total_amount)AS average_ticket_sales
2 FROM bookings
```

The 'Data Output' tab shows the results of the query:

average_ticket_sales
1 79025.605811528685

CASE VI:

Return the number of seats in the aircraft that has aircraft code ‘CN1’.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under the 'Airlines' schema, including tables like 'seats', 'aircraft', and 'flights'. The main window shows a query editor with the following SQL code:

```
1 SELECT count (*) AS CN1_seats
2 FROM seats
3 WHERE aircraft_code= 'CN1';
```

The results pane shows a single row of data:

cn1_seats
12

CASE VII:

Return the number of seats in the aircraft that has aircraft code ‘SU9’

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure under the 'Airlines' schema. The main window shows a query editor with the following SQL code:

```
1 SELECT count (*) AS SU9_seats
2 FROM seats
3 WHERE aircraft_code= 'SU9';
```

The results pane shows a single row of data:

SU9_seats
97

CASE VIII:

Write a query to return the aircraft code and the number of seats of each aircraft in ascending order.

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'Tables (8)'. The main area is a 'Query Editor' window containing the following SQL query:

```
1 SELECT aircraft_code, count(*)
2 FROM seats
3 GROUP BY aircraft_code
4 ORDER BY count;
```

The 'Data Output' tab shows the results of the query:

aircraft_code	count
CN1	12
CR2	50
SU9	97
319	116
733	130
320	140
321	170
763	222
773	402

CASE IX:

Calculate the number of seats for all aircraft models, but now taking into account the class of service; Business class and Economic class.

The screenshot shows the pgAdmin 4 interface. The left sidebar is titled 'Browser' and lists various database objects under 'Tables (8)'. The main area is a 'Query Editor' window containing the following SQL query:

```
1 SELECT aircraft_code, fare_conditions, count(*)
2 FROM seats
3 GROUP BY 1,2
4 ORDER BY 1,2;
```

The 'Data Output' tab shows the results of the query:

aircraft_code	fare_conditions	count
733	Business	12
733	Economy	118
763	Business	30
763	Economy	192
773	Business	30
773	Comfort	48
773	Economy	324
CN1	Economy	12
CR2	Economy	50
SU9	Business	12
SU9	Economy	85

CASE X:

What was the least day in tickets sales?

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects under 'Airlines' schema, including 'Tables (8)' which contains 'aircrafts', 'airports', 'boarding_passes', 'bookings', 'flights', 'seats', 'ticket_flights', and 'tickets'. The main panel shows a query editor with the following SQL code:

```
1 SELECT book_date, sum(total_amount) AS sales
2 FROM bookings
3 GROUP BY 1
4 ORDER BY 2
5 LIMIT 1
```

The results pane shows a single row of data:

book_date	sales
2017-07-26 13:19:00+01	3400.00

CASE XI:

Return all information about aircraft that has aircraft code ‘SU9’ and its range in miles.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects under 'Airlines' schema, including 'Tables (8)' which contains 'aircrafts', 'airports', 'boarding_passes', 'bookings', 'flights', 'seats', 'ticket_flights', and 'tickets'. The main panel shows a query editor with the following SQL code:

```
1 SELECT *, round(range/1.69,2) AS range_in_miles
2 FROM aircrafts
3 WHERE aircraft_code= 'SU9';
```

The results pane shows a single row of data:

aircraft_code	model	range	range_in_miles
SU9	{"en": "Sukhoi Superjet-100", "ru": "Сухой Суперджет-100"}	3000	1775.15

CASE XII:

What is the shortest flight duration for each possible flight from Moscow to St. Petersburg, and how many times was the flight delayed for more than an hour?

The screenshot shows the pgAdmin 4 interface with a query editor window. The browser pane on the left lists various database objects like Publications, Schemas, and Tables. The tables listed include aircrafts, airports, boarding_passes, bookings, flights, seats, ticket_flights, tickets, Trigger Functions, Types, Views, Subscriptions, and PostgreSQL system objects.

The query editor contains the following SQL code:

```
1 SELECT flight_no, (scheduled_arrival - scheduled_departure) AS scheduled_duration,
2 min(scheduled_arrival - scheduled_departure), max(scheduled_arrival - scheduled_departure),
3 sum(CASE WHEN actual_departure > scheduled_departure + INTERVAL '1 hour' THEN 1 ELSE 0 END) delays
4 FROM flights
5 WHERE (SELECT city->'en' FROM airports WHERE airport_code= departure_airport)='Moscow'
6 AND (SELECT city->'en' FROM airports WHERE airport_code= arrival_airport)='St. Petersburg'
7 AND status= 'Arrived'
8 GROUP BY flight_no, (scheduled_arrival - scheduled_departure);
```

The results table shows 12 rows of flight data with columns: flight_no, scheduled_duration, min_interval, max_interval, and delays.

flight_no	scheduled_duration	min_interval	max_interval	delays
PG0228	00:50:00	00:50:00	00:50:00	3
PG0229	00:50:00	00:50:00	00:50:00	2
PG0402	00:55:00	00:55:00	00:55:00	2
PG0403	00:55:00	00:55:00	00:55:00	2
PG0404	00:55:00	00:55:00	00:55:00	3
PG0405	00:55:00	00:55:00	00:55:00	1
PG0468	00:50:00	00:50:00	00:50:00	0
PG0469	00:50:00	00:50:00	00:50:00	5
PG0470	00:50:00	00:50:00	00:50:00	1
PG0471	00:50:00	00:50:00	00:50:00	1
PG0472	00:50:00	00:50:00	00:50:00	3

CASE XIII:

Write a query to arrange the range of model of aircrafts so short range is less than 2000, middle range is more than 2000 and less than 5000 and any range above 5000 is long range.

The screenshot shows the pgAdmin 4 interface with a query editor window. The browser pane on the left lists various database objects like Publications, Schemas, and Tables. The tables listed include aircrafts, airports, boarding_passes, bookings, flights, seats, ticket_flights, tickets, Trigger Functions, Types, Views, Subscriptions, and PostgreSQL system objects.

The query editor contains the following SQL code:

```
1 SELECT model, range,
2 CASE WHEN range < 2000 THEN 'Short'
3 WHEN range < 5000 THEN 'Middle'
4 ELSE 'Long'
5 END AS range
6 FROM aircrafts
7 ORDER BY model;
```

The results table shows 9 rows of aircraft data with columns: model, range, and range_text.

model	range	range_text
{en: "Airbus A319-100", ru: "Аэробус A319-100"}	6700	Long
{en: "Airbus A320-200", ru: "Аэробус A320-200"}	5700	Long
{en: "Airbus A321-200", ru: "Аэробус A321-200"}	5600	Long
{en: "Boeing 737-300", ru: "Боинг 737-300"}	4200	Middle
{en: "Boeing 767-300", ru: "Боинг 767-300"}	7900	Long
{en: "Boeing 777-300", ru: "Боинг 777-300"}	11100	Long
{en: "Bombardier CRJ-200", ru: "Бомбардье CRJ-200"}	2700	Middle
{en: "Cessna 208 Caravan", ru: "Сессна 208 Караван"}	1200	Short
{en: "Sukhoi Superjet-100", ru: "Сухой Суперджет-100"}	3000	Middle

CASE XIV:

Determine how many flights from each city to other cities, return the name of the city and count of flights more than 50. Order the data from the largest number of flights to the least.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like airports, flights, and bookings. The main area shows a query editor with the following SQL code:

```
1 SELECT (SELECT city->>'en' FROM airports WHERE airport_code= departure_airport)
2 AS departure_city, COUNT (*)
3 FROM flights
4 GROUP BY (SELECT city->>'en' FROM airports WHERE airport_code= departure_airport)
5 HAVING count(*)>=50
6 ORDER BY count DESC;
```

The results table shows the departure city and the count of flights, ordered by count in descending order. The data is as follows:

departure_city	count
Moscow	7917
St. Petersburg	1900
Novosibirsk	1055
Krasnoyarsk	707
Yekaterinburg	689
Perm	619
Rostov	617
Bryansk	610
Ulyanovsk	584
Sochi	584
Sovetskiy	549

CASE XV:

Return all flight details in the indicated day (2017-08-28) from airport whose code is 'KZN'. Include flight count in ascending order, departures count and when departures happens and arrivals count and when arrivals happen.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like airports, flights, and bookings. The main area shows a query editor with the following SQL code:

```
1 SELECT flight_no, scheduled_departure :: time AS dep_time, departure_airport AS departures, arrival_airport AS arrivals,
2 count(flight_id)AS flight_count
3 FROM flights
4 WHERE departure_airport= 'KZN'
5 AND scheduled_departure >= '2017-08-28': date
6 AND scheduled_departure < '2017-08-29': date
7 GROUP BY 1,2,3,4,scheduled_departure
8 ORDER BY flight_count
9 DESC, arrival_airport, scheduled_departure;
```

The results table shows flight details for KZN airport on 2017-08-28, ordered by flight count in ascending order. The data is as follows:

flight_no	dep_time	departures	arrivals	flight_count
PG0203	15:45:00	KZN	DME	1
PG0498	08:15:00	KZN	IKT	1
PG0039	15:25:00	KZN	KVX	1
PG0508	07:45:00	KZN	LED	1
PG0621	14:05:00	KZN	MQF	1
PG0609	09:15:00	KZN	ROV	1
PG0610	10:45:00	KZN	ROV	1

CASE XVI:

Who travelled from Moscow (SVO) to Novosibirsk (OVB) on seat 1A the day before yesterday, and when was the ticket booked?

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various objects like Schemas, Tables, and Functions. The main area shows a SQL query in the Query Editor:

```
1 /* The day before yesterday is counted from the public.now value, not from the current date. */
2
3 SELECT t.passenger_name, book_date FROM bookings b
4 JOIN tickets t
5 ON t.book_ref= b.book_ref
6 JOIN boarding_passes bp
7 ON bp.ticket_no= t.ticket_no
8 JOIN flights f
9 ON f.flight_id= bp.flight_id
10 WHERE f.departure_airport= 'SVO' AND f.arrival_airport= 'OVB'
11 AND f.scheduled_departure::date= public.now()::date - INTERVAL '2 day'
12 AND bp.seat_no= '1A';
```

The Data Output pane shows the results of the query:

passenger_name	book_date
SERGEY SCHERBAKOV	2017-07-28 18:39:00+01

CASE XVII:

Find the most disciplined passengers who checked in first for all their flights. Take into account only those passengers who took at least two flights.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema. The main area shows a SQL query in the Query Editor:

```
1 SELECT t.passenger_name, t.ticket_no FROM tickets t
2 JOIN boarding_passes bp
3 ON bp.ticket_no= t.ticket_no
4 GROUP BY t.passenger_name, t.ticket_no
5 HAVING max(bp.boarding_no)= 1 AND count(*)>1;
```

The Data Output pane shows the results of the query:

passenger_name	ticket_no
VIKTOR BELOV	0005432054255
YULIYA SOROKINA	0005432146218
NIKOLAY SCHERBAKOV	0005432195667
ANDREY FROLOV	0005432293170
ANNA ANDREEVA	0005432295838
ALEKSANDR MARTYNOV	0005432359667
ANNA MATVEEVA	0005432398234
KONSTANTIN KAZAKOV	0005432427125
SERGEY VOROBEV	0005432566574
ALEKSANDR FILIPPPOV	0005432675028
NINA LOGINOVA	0005432784253

CASE XVIII:

Calculate the number of passengers and number of flights departing from one airport (SVO) during each hour on the indicated day 2017-08-02.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like aircrafts, airports, and bookings. The main area shows a query editor with the following SQL code:

```
1 SELECT date_part('hour', f.scheduled_departure) AS hour, count(ticket_no) passengers_cnt,
2 count(DISTINCT f.flight_id) flights_cnt
3 FROM flights f
4 JOIN ticket_flights t
5 ON f.flight_id = t.flight_id
6 WHERE f.departure_airport = 'SVO'
7 AND f.scheduled_departure >= '2017-08-02':: date
8 AND f.scheduled_departure < '2017-08-03':: date
9 GROUP BY date_part('hour', f.scheduled_departure);
```

The results are displayed in a table:

hour	passengers_cnt	flights_cnt
1	7	484
2	8	381
3	9	540
4	10	534
5	11	157
6	12	217
7	14	273
8	15	421
9	16	237
10	17	30

CASE XIX:

Return unique city name, flight number, airport and time zone.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with tables like aircrafts, airports, and bookings. The main area shows a query editor with the following SQL code:

```
1 SELECT DISTINCT a.city-->'en' AS cities, f.flight_no, a.airport_name-->'en' AS airport, a.timezone
2 FROM flights f
3 JOIN airports a
4 ON a.airport_code = f.departure_airport;
```

The results are displayed in a table:

cities	flight_no	airport	timezone
Abakan	PG0070	Abakan Airport	Asia/Krasnoyarsk
Abakan	PG0313	Abakan Airport	Asia/Krasnoyarsk
Abakan	PG0490	Abakan Airport	Asia/Krasnoyarsk
Abakan	PG0520	Abakan Airport	Asia/Krasnoyarsk
Abakan	PG0585	Abakan Airport	Asia/Krasnoyarsk
Abakan	PG0586	Abakan Airport	Asia/Krasnoyarsk
Abakan	PG0702	Abakan Airport	Asia/Krasnoyarsk
Anadyr	PG0087	Ugolny Airport	Asia/Anadyr
Anadyr	PG0256	Ugolny Airport	Asia/Anadyr
Anadyr	PG0316	Ugolny Airport	Asia/Anadyr
Anadyr	PG0451	Ugolny Airport	Asia/Anadyr

CASE XX:

How many people can be included into a single booking according to the available data?

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with various objects like Publications, Schemas, Tables, and Views. The main area shows a query editor with the following SQL code:

```
1 SELECT tt.bookings_no, count(*) passengers_no
2 FROM (SELECT t.book_ref, count(*) bookings_no FROM tickets t GROUP BY t.book_ref)tt
3 GROUP BY tt.bookings_no
4 ORDER BY tt.bookings_no
```

The results table shows the count of passengers per booking number:

bookings_no	passengers_no
1	173390
2	75793
3	12686
4	896
5	23

CASE XXI:

Which combination of first and last names occur most often? What is the ratio of the passengers with such names to the total number of passengers?

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema. The main area shows a query editor with the following SQL code:

```
1 SELECT passenger_name, round(100.0+cnt/sum(cnt)OVER(),2)AS percent
2 FROM (SELECT passenger_name, count(*)cnt FROM tickets GROUP BY passenger_name)sub
3 ORDER BY percent DESC
```

The results table shows the most common first and last name combinations with their percentage of the total:

passenger_name	percent
ALEKSANDR IVANOV	0.23
ALEKSANDR KUZNECOV	0.21
SERGEY IVANOV	0.17
SERGEY KUZNECOV	0.16
VLADIMIR IVANOV	0.15
ALEKSANDR POPOV	0.13
VLADIMIR KUZNECOV	0.13
ALEKSANDR PETROV	0.12
ELENA KUZNECOVA	0.12
TATYANA IVANOVA	0.12
ALEKSANDR VASILEV	0.11

CASE XXII:

What are the maximum and minimum ticket prices in all directions?

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with 8 tables. The right pane shows a query editor with the following SQL code:

```
1 SELECT (SELECT city->'en'FROM airports WHERE airport_code= f.departure_airport)AS departure_city,
2 (SELECT city->'en'FROM airports WHERE airport_code= f.arrival_airport)AS arrival_city, max(tf.amount),min(tf.amount)
3 FROM flights f
4 JOIN ticket_flights tf
5 ON f.flight_id=tf.flight_id
6 GROUP BY 1,2
7 ORDER BY 1,2;
```

The results table shows the departure city, arrival city, maximum ticket price, and minimum ticket price for each flight direction.

	departure_city	arrival_city	max	min
1	Abakan	Moscow	101000.00	33700.00
2	Abakan	Novosibirsk	5800.00	5800.00
3	Abakan	Tomsk	4900.00	4900.00
4	Anadyr	Khabarovsk	92200.00	30700.00
5	Anadyr	Moscow	185300.00	61800.00
6	Anapa	Belgorod	18900.00	6300.00
7	Anapa	Moscow	36600.00	12200.00
8	Arkhangelsk	Khanty-Mansiysk	16400.00	14900.00
9	Arkhangelsk	Moscow	11100.00	10100.00
10	Arkhangelsk	Naryan-Mar	7300.00	6600.00
11	Arkhangelsk	Perm	11000.00	11000.00

CASE XXIII:

Get a list of airports in cities with more than one airports.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database schema with 8 tables. The right pane shows a query editor with the following SQL code:

```
1 SELECT aa.city->'en' AS city, aa.airport_code, aa.airport_name->'en' AS airport
2 FROM (SELECT city, count(*) FROM airports GROUP BY city HAVING count (>1))AS a
3 JOIN airports AS aa
4 ON a.city= aa.city
5 ORDER BY aa.city, aa.airport_name;
```

The results table shows the city name, airport code, and airport name for each airport located in a city with more than one airport.

	city	airport_code	airport
1	Moscow	DME	Domodedovo International Airport
2	Moscow	SVO	Sheremetyevo International Airport
3	Moscow	VKO	Vnukovo International Airport
4	Ulyanovsk	ULV	Ulyanovsk Baratayevka Airport
5	Ulyanovsk	ULY	Ulyanovsk East Airport

CASE XXIV:

What will be the total number of different routes that are theoretically laid between all cities?

The screenshot shows the pgAdmin 4 interface. The left sidebar (Browser) lists various database objects: Publications, Schemas (1), Tables (8), Views (1), Subscriptions, postgres, Login/Group Roles, and Tablespaces. The main area (Query Editor) contains the following SQL code:

```
1 SELECT count (*)
2 FROM (SELECT DISTINCT city FROM airports)AS a1
3 JOIN(SELECT DISTINCT city FROM airports)AS a2
4 ON a1.city <> a2.city
```

The Data Output tab shows the result of the query:

count	bigint
1	10100

CASE XXV:

Suppose our airline marketers want to know how often there are different names among the passengers?

The screenshot shows the pgAdmin 4 interface. The left sidebar (Browser) lists various database objects: Publications, Schemas (1), Tables (8), and Views (1). The main area (Query Editor) contains the following SQL code:

```
1 SELECT LEFT(passenger_name, STRPOS(passenger_name, ' ')-1)AS firstname, count(*)
2 FROM tickets
3 GROUP BY 1
4 ORDER BY 2 DESC;
```

The Data Output tab shows the result of the query:

firstname	text	count	bigint
1	ALEKSANDR	20328	
2	SERGEY	15133	
3	VLADIMIR	12806	
4	TATYANA	12058	
5	ELENA	11291	
6	OLGA	9998	
7	NATALYA	9716	
8	ALEKSEY	9555	
9	NIKOLAY	8048	
10	VALENTINA	8038	
11	DMITRIY	7840	

CASE XXVI:

What combination of first names and last names separately occur most often? What is the ratio of the passengers with such names to the total number of passengers?

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
WITH p AS (SELECT LEFT(passenger_name, POSITION(' ' IN passenger_name))AS passenger_name FROM tickets)
SELECT passenger_name, round(100.0*cnt/sum(cnt)OVER(),2)AS percent
FROM(SELECT passenger_name, count(*)cnt FROM p GROUP BY passenger_name)t
ORDER BY percent DESC;
```

The results table shows the top 11 most common passenger names and their percentages:

passenger_name	percent
ALEKSANDR	5.54
SERGEY	4.13
VLADIMIR	3.49
TATYANA	3.29
ELENA	3.08
OLGA	2.73
NATALYA	2.65
ALEKSEY	2.61
VALENTINA	2.19
NIKOLAY	2.19
DMITRIY	2.14

CASE XXVII:

For each ticket, display all the included flight segments, together with connection time. Limit the result to the tickets booked a week ago.

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
SELECT tf.ticket_no, f.departure_airport, f.arrival_airport, f.scheduled_arrival,
LEAD(f.scheduled_departure)OVER w AS next_departure,
LEAD(f.scheduled_departure)OVER w - f.scheduled_arrival AS gap
FROM bookings b
JOIN tickets t
ON t.book_ref = b.book_ref
JOIN ticket_flights tf
ON tf.ticket_no = t.ticket_no
JOIN flights f
ON tf.flight_id = f.flight_id
WHERE b.book_date = public.now()::date - INTERVAL'7day'
WINDOW w AS (PARTITION BY tf.ticket_no ORDER BY f.scheduled_departure);
```

The results table shows the flight segments and connection times for 11 different tickets:

ticket_no	departure_airport	arrival_airport	scheduled_arrival	next_departure	gap
1 0005432150393	SVO	AER	2017-08-27 16:50:00+01	2017-08-28 12:10:00+01	19:20:00
2 0005432150393	AER	KUF	2017-08-28 14:00:00+01	[null]	[null]
3 0005432150394	SVO	AER	2017-08-27 16:50:00+01	2017-08-28 12:10:00+01	19:20:00
4 0005432150394	AER	KUF	2017-08-28 14:00:00+01	[null]	[null]
5 0005432262754	DME	LED	2017-08-20 08:30:00+01	[null]	[null]
6 0005432262758	DME	LED	2017-08-20 08:30:00+01	[null]	[null]
7 0005432839725	SVO	CSY	2017-08-24 08:30:00+01	2017-08-29 10:05:00+01	5 days 01:35:00
8 0005432839725	CSY	SVO	2017-08-29 10:05:00+01	[null]	[null]
9 0005433009596	KJA	OVS	2017-08-23 07:45:00+01	2017-08-24 05:45:00+01	22:00:00
10 0005433009596	OVS	DME	2017-08-24 07:55:00+01	2017-08-28 09:15:00+01	4 days 01:20:00
11 0005433009596	DME	OVS	2017-08-28 11:25:00+01	2017-08-29 09:25:00+01	22:00:00

CASE XXVIII:

Which flights had the longest delays?

The screenshot shows the pgAdmin 4 interface. The left sidebar (Browser) displays the database structure under the 'public' schema, including tables like 'aircrafts', 'airports', 'boarding_passes', etc. The central area (Query Editor) contains a SQL query to select flights with the longest delays. The results are shown in a Data Output table.

```
1 SELECT f.flight_no, f.scheduled_departure, f.actual_departure,
2 (f.actual_departure - f.scheduled_departure)AS delay
3 FROM flights f
4 WHERE f.actual_departure IS NOT NULL
5 ORDER BY f.actual_departure - f.scheduled_departure DESC;
```

flight_no	scheduled_departure	actual_departure	delay
PG0589	2017-07-29 13:30:00+01	2017-07-29 18:07:00+01	04:37:00
PG0164	2017-07-29 13:25:00+01	2017-07-29 17:53:00+01	04:28:00
PG0364	2017-07-19 09:45:00+01	2017-07-19 14:12:00+01	04:27:00
PG0568	2017-08-13 14:15:00+01	2017-08-13 18:35:00+01	04:20:00
PG0454	2017-08-02 08:05:00+01	2017-08-02 12:23:00+01	04:18:00
PG0096	2017-08-03 14:35:00+01	2017-08-03 18:53:00+01	04:18:00
PG0166	2017-08-12 12:35:00+01	2017-08-12 16:51:00+01	04:16:00
PG0278	2017-07-16 12:20:00+01	2017-07-16 16:36:00+01	04:16:00
PG0564	2017-08-10 07:30:00+01	2017-08-10 11:44:00+01	04:14:00
PG0617	2017-07-16 12:30:00+01	2017-07-16 16:38:00+01	04:08:00
PG0669	2017-07-19 14:15:00+01	2017-07-19 18:23:00+01	04:08:00

CASE XXIX:

How many seats remained free on the flight PG0404 in the day before the last in the airlines database?

The screenshot shows the pgAdmin 4 interface. The left sidebar (Browser) displays the database structure under the 'public' schema. The central area (Query Editor) contains a SQL query to count free seats on flight PG0404 the day before the last flight. The results are shown in a Data Output table.

```
1 SELECT count(*) AS free_seats
2 FROM flights f
3 JOIN seats s
4 ON s.aircraft_code = f.aircraft_code
5 WHERE f.flight_no = 'PG0404'
6 AND f.scheduled_departure::date = public.now()::date -INTERVAL'1day'
7 AND NOT EXISTS (SELECT NULL FROM boarding_passes bp
8 WHERE bp.flight_id = f.flight_id
9 AND bp.seat_no = s.seat_no);
```

free_seats
63