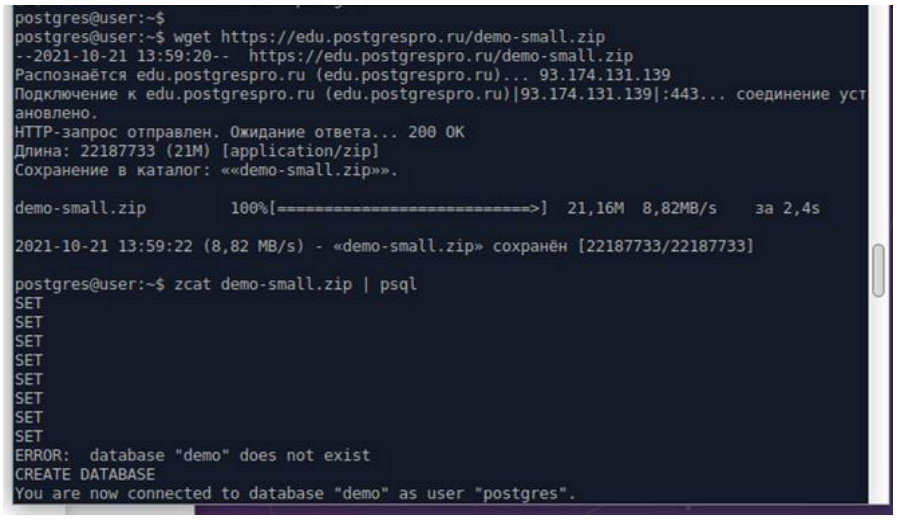
Мамонов Антон 3ИСиП-19-1

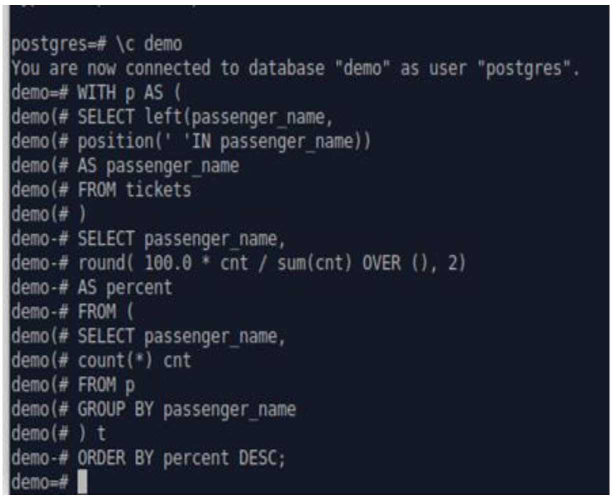
Создаем в Postgresql демонстрационную базу



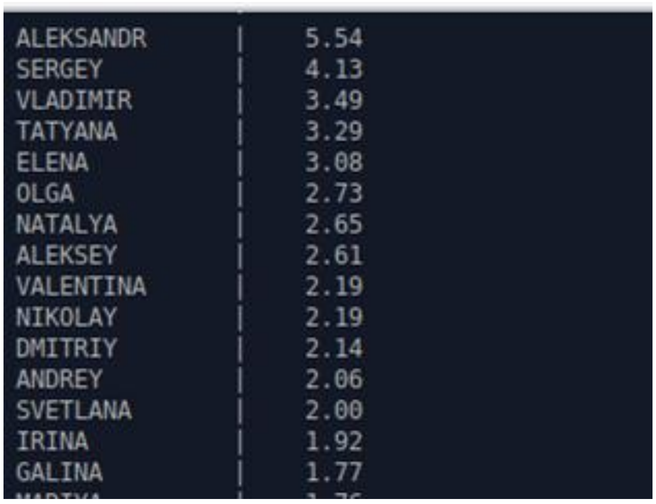
```
postgres@user:~$
postgres@user:~$ wget https://edu.postgrespro.ru/demo-small.zip
--2021-10-21 13:59:20--  https://edu.postgrespro.ru/demo-small.zip
Распознаётся edu.postgrespro.ru (edu.postgrespro.ru)... 93.174.131.139
Подключение к edu.postgrespro.ru (edu.postgrespro.ru)|93.174.131.139|:443... соединение уст
ановлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: 22187733 (21M) [application/zip]
Сохранение в каталог: ««demo-small.zip»».

demo-small.zip        100%[==========================>]  21,16M  8,82MB/s    за 2,4s

2021-10-21 13:59:22 (8,82 MB/s) - «demo-small.zip» сохранён [22187733/22187733]

postgres@user:~$ zcat demo-small.zip | psql
SET
SET
SET
SET
SET
SET
SET
SET
ERROR:  database "demo" does not exist
CREATE DATABASE
You are now connected to database "demo" as user "postgres".
```

Теперь сделаем задачки

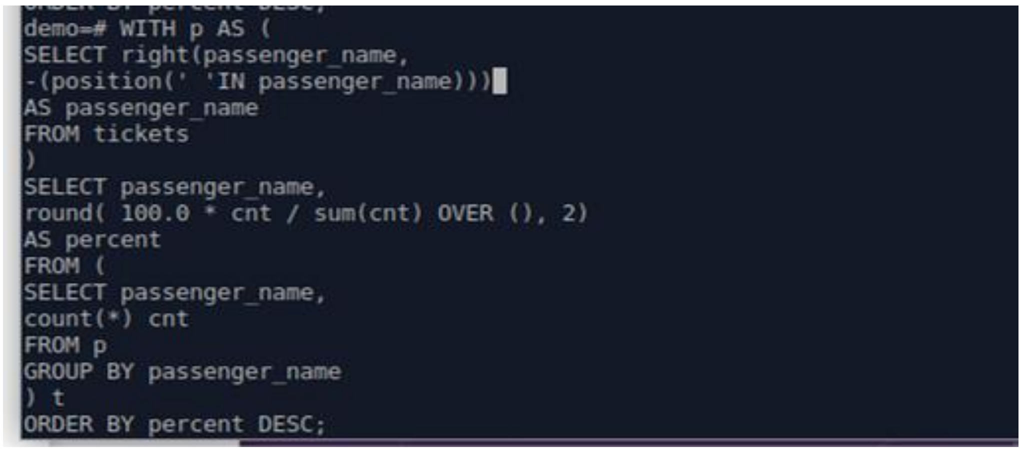
1. Какие сочетания имени и фамилии встречаются чаще всего и какую долю от числа всех пассажиров они составляют?



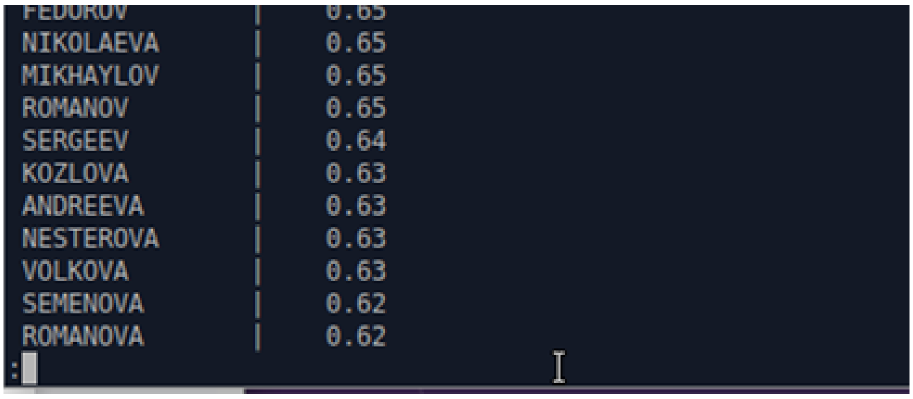
```
postgres=# \c demo
You are now connected to database "demo" as user "postgres".
demo=# WITH p AS (
demo(# SELECT left(passenger_name,
demo(# position(' 'IN passenger_name))
demo(# AS passenger_name
demo(# FROM tickets
demo(# )
demo-# SELECT passenger_name,
demo-# round( 100.0 * cnt / sum(cnt) OVER (), 2)
demo-# AS percent
demo-# FROM (
demo(# SELECT passenger_name,
demo(# count(*) cnt
demo(# FROM p
demo(# GROUP BY passenger_name
demo(# ) t
demo-# ORDER BY percent DESC;
demo=#
```



```
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
ANDREY         |     2.06
SVETLANA       |     2.00
IRINA          |     1.92
GALINA         |     1.77
```

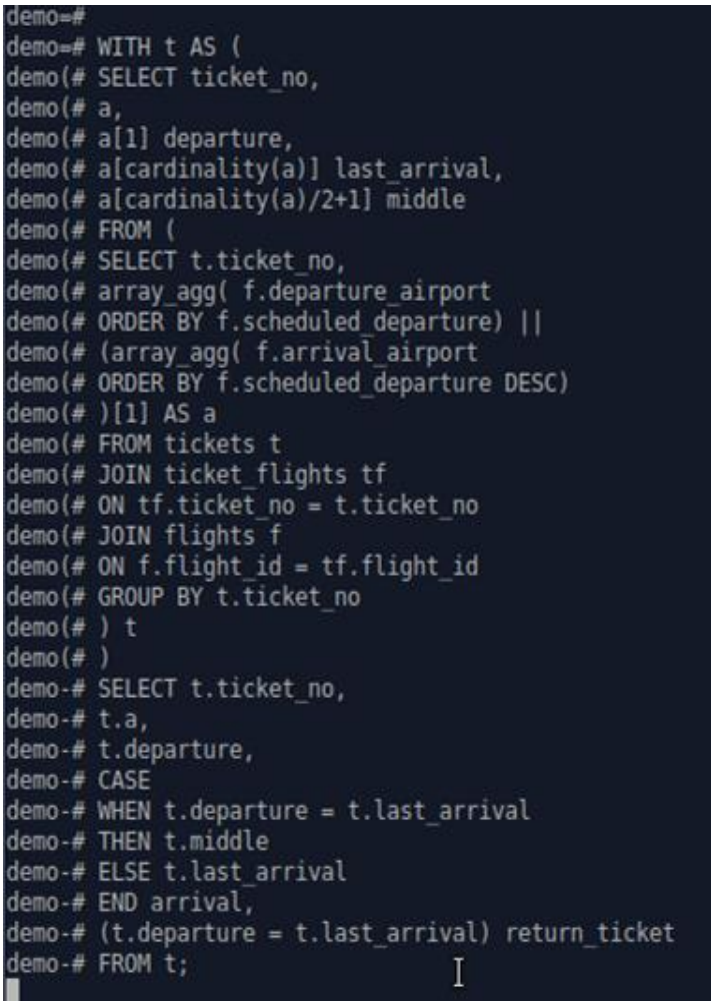


```
demo=# WITH p AS (
SELECT right(passenger_name,
-(position(' 'IN passenger_name)))
AS passenger_name
FROM tickets
)
SELECT passenger_name,
round( 100.0 * cnt / sum(cnt) OVER (), 2)
AS percent
FROM (
SELECT passenger_name,
count(*) cnt
FROM p
GROUP BY passenger_name
) t
ORDER BY percent DESC;
```

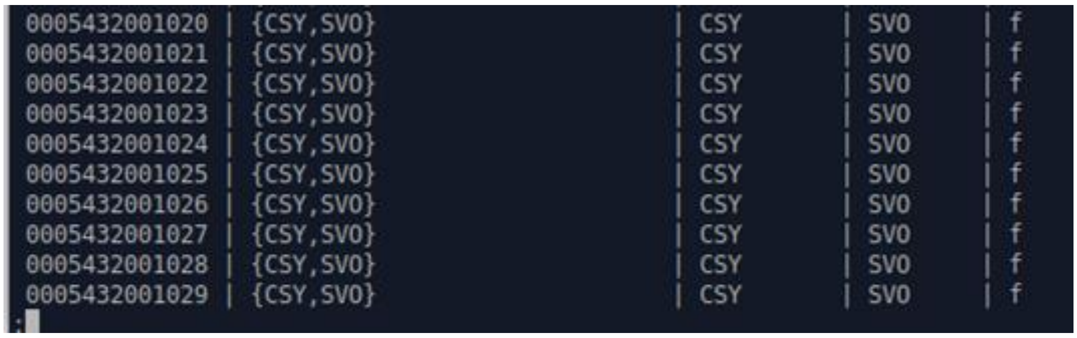


```
FEDOROV          |    0.65
NIKOLAEVA        |    0.65
MIKHAYLOV        |    0.65
ROMANOV          |    0.65
SERGEEV          |    0.64
KOZLOVA          |    0.63
ANDREEVA         |    0.63
NESTEROVA        |    0.63
VOLKOVA          |    0.63
SEMENOVA         |    0.62
ROMANOVA         |    0.62
:
```

2. В билете нет указания, в один ли он конец, или туда и обратно. Однако это можно вычислить, сравнив первый пункт отправления с последним пунктом назначения. Выведите для каждого билета аэропорты отправления и назначения без учета пересадок, и признак, взят ли билет туда и обратно.
Решение.

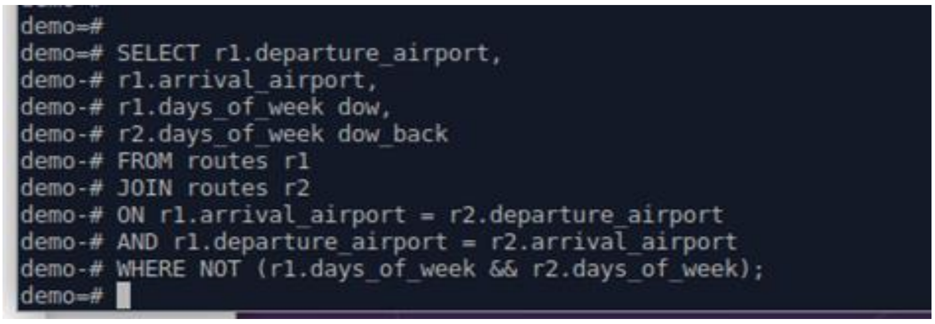


```
demo=#
demo=# WITH t AS (
demo(# SELECT ticket_no,
demo(# a,
demo(# a[1] departure,
demo(# a[cardinality(a)] last_arrival,
demo(# a[cardinality(a)/2+1] middle
demo(# FROM (
demo(# SELECT t.ticket_no,
demo(# array_agg( f.departure_airport
demo(# ORDER BY f.scheduled_departure) ||
demo(# (array_agg( f.arrival_airport
demo(# ORDER BY f.scheduled_departure DESC)
demo(# )[1] AS a
demo(# FROM tickets t
demo(# JOIN ticket_flights tf
demo(# ON tf.ticket_no = t.ticket_no
demo(# JOIN flights f
demo(# ON f.flight_id = tf.flight_id
demo(# GROUP BY t.ticket_no
demo(# ) t
demo(# )
demo-# SELECT t.ticket_no,
demo-# t.a,
demo-# t.departure,
demo-# CASE
demo-# WHEN t.departure = t.last_arrival
demo-# THEN t.middle
demo-# ELSE t.last_arrival
demo-# END arrival,
demo-# (t.departure = t.last_arrival) return_ticket
demo-# FROM t;
```

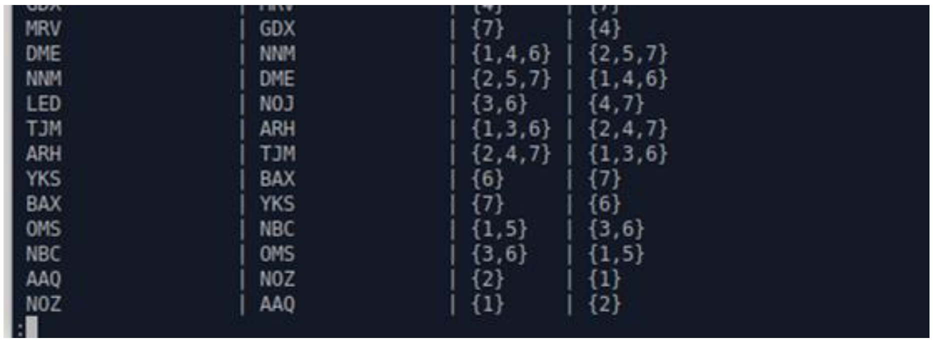


```
0005432001020 | {CSY,SVO}                     | CSY       | SVO      | f
0005432001021 | {CSY,SVO}                     | CSY       | SVO      | f
0005432001022 | {CSY,SVO}                     | CSY       | SVO      | f
0005432001023 | {CSY,SVO}                     | CSY       | SVO      | f
0005432001024 | {CSY,SVO}                     | CSY       | SVO      | f
0005432001025 | {CSY,SVO}                     | CSY       | SVO      | f
0005432001026 | {CSY,SVO}                     | CSY       | SVO      | f
0005432001027 | {CSY,SVO}                     | CSY       | SVO      | f
0005432001028 | {CSY,SVO}                     | CSY       | SVO      | f
0005432001029 | {CSY,SVO}                     | CSY       | SVO      | f
:
```

3. Найдите билеты, взятые туда и обратно, в которых путь «туда» не совпадает с путем «обратно». Найдите такие пары аэропортов, рейсы между которыми в одну и в другую стороны отправляются по разным дням недели.
Решение.

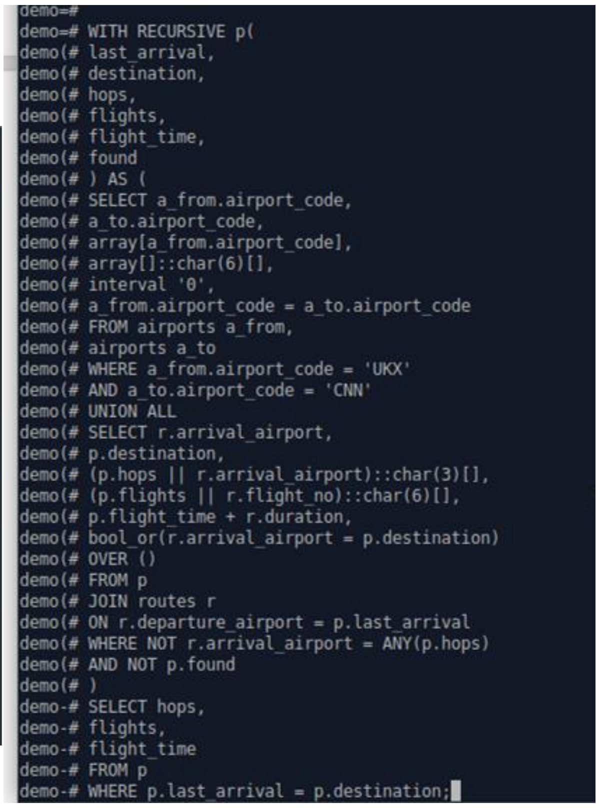


```
demo=#
demo=# SELECT r1.departure_airport,
demo-# r1.arrival_airport,
demo-# r1.days_of_week dow,
demo-# r2.days_of_week dow_back
demo-# FROM routes r1
demo-# JOIN routes r2
demo-# ON r1.arrival_airport = r2.departure_airport
demo-# AND r1.departure_airport = r2.arrival_airport
demo-# WHERE NOT (r1.days_of_week && r2.days_of_week);
demo=#
```

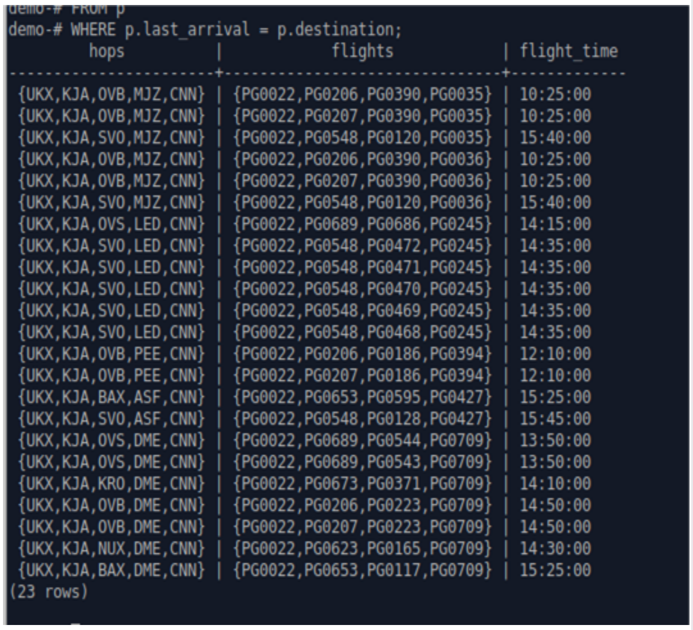


```
 GDX            | MRV            | {4}     | {7}
 MRV            | GDX            | {7}     | {4}
 DME            | NNM            | {1,4,6} | {2,5,7}
 NNM            | DME            | {2,5,7} | {1,4,6}
 LED            | NOJ            | {3,6}   | {4,7}
 TJM            | ARH            | {1,3,6} | {2,4,7}
 ARH            | TJM            | {2,4,7} | {1,3,6}
 YKS            | BAX            | {6}     | {7}
 BAX            | YKS            | {7}     | {6}
 OMS            | NBC            | {1,5}   | {3,6}
 NBC            | OMS            | {3,6}   | {1,5}
 AAQ            | NOZ            | {2}     | {1}
 NOZ            | AAQ            | {1}     | {2}
:
```

4. Как с помощью минимального числа пересадок можно долететь из Усть-Кута (UKX) в Нерюнгри (CNN), и какое время придется провести в воздухе?

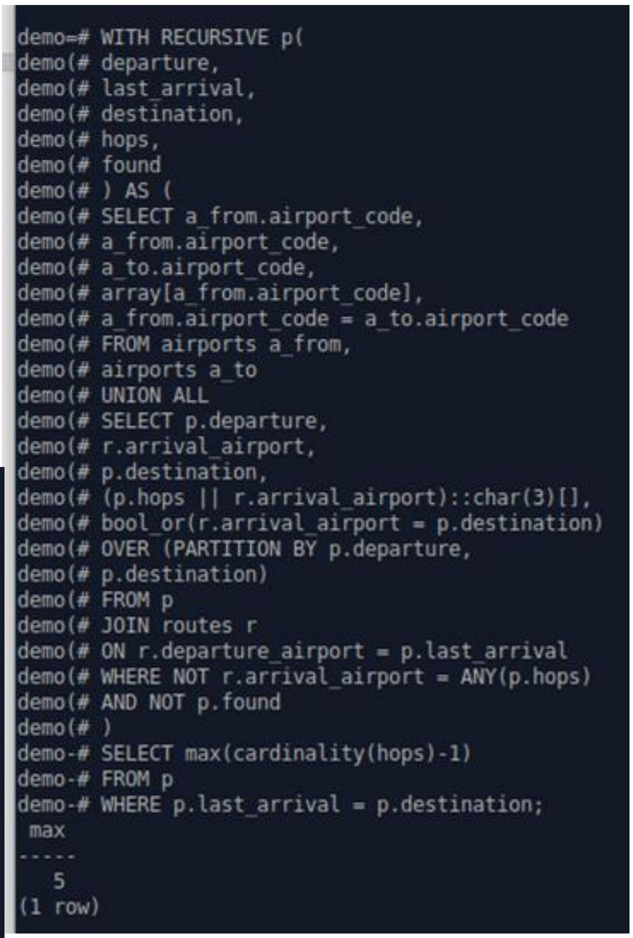


```
demo=#
demo=# WITH RECURSIVE p(
demo(# last_arrival,
demo(# destination,
demo(# hops,
demo(# flights,
demo(# flight_time,
demo(# found
demo(# ) AS (
demo(# SELECT a_from.airport_code,
demo(# a_to.airport_code,
demo(# array[a_from.airport_code],
demo(# array[]::char(6)[],
demo(# interval '0',
demo(# a_from.airport_code = a_to.airport_code
demo(# FROM airports a_from,
demo(# airports a_to
demo(# WHERE a_from.airport_code = 'UKX'
demo(# AND a_to.airport_code = 'CNN'
demo(# UNION ALL
demo(# SELECT r.arrival_airport,
demo(# p.destination,
demo(# (p.hops || r.arrival_airport)::char(3)[],
demo(# (p.flights || r.flight_no)::char(6)[],
demo(# p.flight_time + r.duration,
demo(# bool_or(r.arrival_airport = p.destination)
demo(# OVER ()
demo(# FROM p
demo(# JOIN routes r
demo(# ON r.departure_airport = p.last_arrival
demo(# WHERE NOT r.arrival_airport = ANY(p.hops)
demo(# AND NOT p.found
demo(# )
demo-# SELECT hops,
demo-# flights,
demo-# flight_time
demo-# FROM p
demo-# WHERE p.last_arrival = p.destination;
```



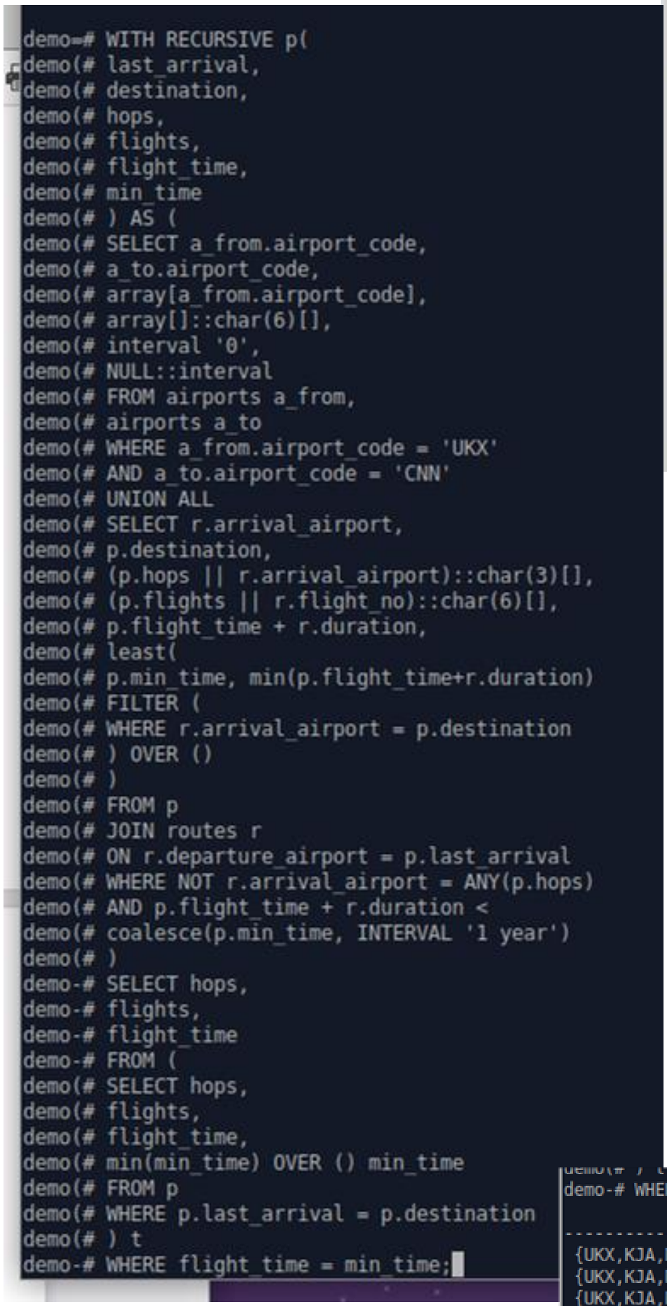
```
demo-# FROM p
demo-# WHERE p.last_arrival = p.destination;
         hops          |            flights            | flight_time
-----------------------+-------------------------------+-------------
 {UKX,KJA,OVB,MJZ,CNN} | {PG0022,PG0206,PG0390,PG0035} | 10:25:00
 {UKX,KJA,OVB,MJZ,CNN} | {PG0022,PG0207,PG0390,PG0035} | 10:25:00
 {UKX,KJA,SVO,MJZ,CNN} | {PG0022,PG0548,PG0120,PG0035} | 15:40:00
 {UKX,KJA,OVB,MJZ,CNN} | {PG0022,PG0206,PG0390,PG0036} | 10:25:00
 {UKX,KJA,OVB,MJZ,CNN} | {PG0022,PG0207,PG0390,PG0036} | 10:25:00
 {UKX,KJA,SVO,MJZ,CNN} | {PG0022,PG0548,PG0120,PG0036} | 15:40:00
 {UKX,KJA,OVS,LED,CNN} | {PG0022,PG0689,PG0686,PG0245} | 14:15:00
 {UKX,KJA,SVO,LED,CNN} | {PG0022,PG0548,PG0472,PG0245} | 14:35:00
 {UKX,KJA,SVO,LED,CNN} | {PG0022,PG0548,PG0471,PG0245} | 14:35:00
 {UKX,KJA,SVO,LED,CNN} | {PG0022,PG0548,PG0470,PG0245} | 14:35:00
 {UKX,KJA,SVO,LED,CNN} | {PG0022,PG0548,PG0469,PG0245} | 14:35:00
 {UKX,KJA,SVO,LED,CNN} | {PG0022,PG0548,PG0468,PG0245} | 14:35:00
 {UKX,KJA,OVB,PEE,CNN} | {PG0022,PG0206,PG0186,PG0394} | 12:10:00
 {UKX,KJA,OVB,PEE,CNN} | {PG0022,PG0207,PG0186,PG0394} | 12:10:00
 {UKX,KJA,BAX,ASF,CNN} | {PG0022,PG0653,PG0595,PG0427} | 15:25:00
 {UKX,KJA,SVO,ASF,CNN} | {PG0022,PG0548,PG0128,PG0427} | 15:45:00
 {UKX,KJA,OVS,DME,CNN} | {PG0022,PG0689,PG0544,PG0709} | 13:50:00
 {UKX,KJA,OVS,DME,CNN} | {PG0022,PG0689,PG0543,PG0709} | 13:50:00
 {UKX,KJA,KRO,DME,CNN} | {PG0022,PG0673,PG0371,PG0709} | 14:10:00
 {UKX,KJA,OVB,DME,CNN} | {PG0022,PG0206,PG0223,PG0709} | 14:50:00
 {UKX,KJA,OVB,DME,CNN} | {PG0022,PG0207,PG0223,PG0709} | 14:50:00
 {UKX,KJA,NUX,DME,CNN} | {PG0022,PG0623,PG0165,PG0709} | 14:30:00
 {UKX,KJA,BAX,DME,CNN} | {PG0022,PG0653,PG0117,PG0709} | 15:25:00
(23 rows)
```

5. Какое максимальное число пересадок может потребоваться, чтобы добраться из одного любого аэропорта в любой другой?

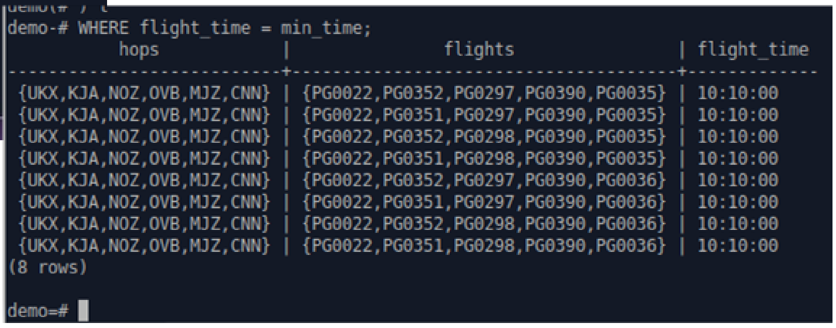


```
demo=# WITH RECURSIVE p(
demo(# departure,
demo(# last_arrival,
demo(# destination,
demo(# hops,
demo(# found
demo(# ) AS (
demo(# SELECT a_from.airport_code,
demo(# a_from.airport_code,
demo(# a_to.airport_code,
demo(# array[a_from.airport_code],
demo(# a_from.airport_code = a_to.airport_code
demo(# FROM airports a_from,
demo(# airports a_to
demo(# UNION ALL
demo(# SELECT p.departure,
demo(# r.arrival_airport,
demo(# p.destination,
demo(# (p.hops || r.arrival_airport)::char(3)[],
demo(# bool_or(r.arrival_airport = p.destination)
demo(# OVER (PARTITION BY p.departure,
demo(# p.destination)
demo(# FROM p
demo(# JOIN routes r
demo(# ON r.departure_airport = p.last_arrival
demo(# WHERE NOT r.arrival_airport = ANY(p.hops)
demo(# AND NOT p.found
demo(# )
demo-# SELECT max(cardinality(hops)-1)
demo-# FROM p
demo-# WHERE p.last_arrival = p.destination;
 max
-----
   5
(1 row)
```

6. Найдите кратчайший путь из Усть-Кута (UKX) в Нерюнгри (CNN) с точки зрения чистого времени перелетов (игнорируя время пересадок)

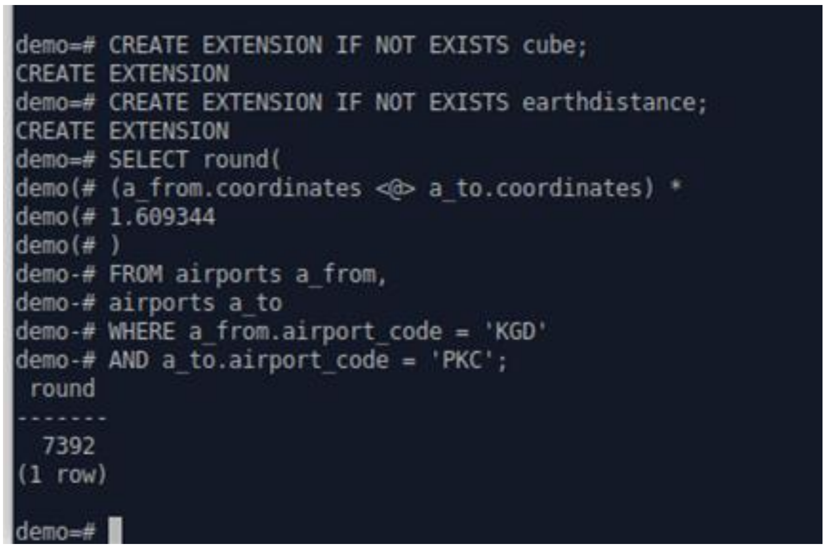


```
demo=# WITH RECURSIVE p(
demo(# last_arrival,
demo(# destination,
demo(# hops,
demo(# flights,
demo(# flight_time,
demo(# min_time
demo(# ) AS (
demo(# SELECT a_from.airport_code,
demo(# a_to.airport_code,
demo(# array[a_from.airport_code],
demo(# array[]::char(6)[],
demo(# interval '0',
demo(# NULL::interval
demo(# FROM airports a_from,
demo(# airports a_to
demo(# WHERE a_from.airport_code = 'UKX'
demo(# AND a_to.airport_code = 'CNN'
demo(# UNION ALL
demo(# SELECT r.arrival_airport,
demo(# p.destination,
demo(# (p.hops || r.arrival_airport)::char(3)[],
demo(# (p.flights || r.flight_no)::char(6)[],
demo(# p.flight_time + r.duration,
demo(# least(
demo(# p.min_time, min(p.flight_time+r.duration)
demo(# FILTER (
demo(# WHERE r.arrival_airport = p.destination
demo(# ) OVER ()
demo(# )
demo(# FROM p
demo(# JOIN routes r
demo(# ON r.departure_airport = p.last_arrival
demo(# WHERE NOT r.arrival_airport = ANY(p.hops)
demo(# AND p.flight_time + r.duration <
demo(# coalesce(p.min_time, INTERVAL '1 year')
demo(# )
demo-# SELECT hops,
demo-# flights,
demo-# flight_time
demo-# FROM (
demo(# SELECT hops,
demo(# flights,
demo(# flight_time,
demo(# min(min_time) OVER () min_time
demo(# FROM p
demo(# WHERE p.last_arrival = p.destination
demo(# ) t
demo-# WHERE flight_time = min_time;
```



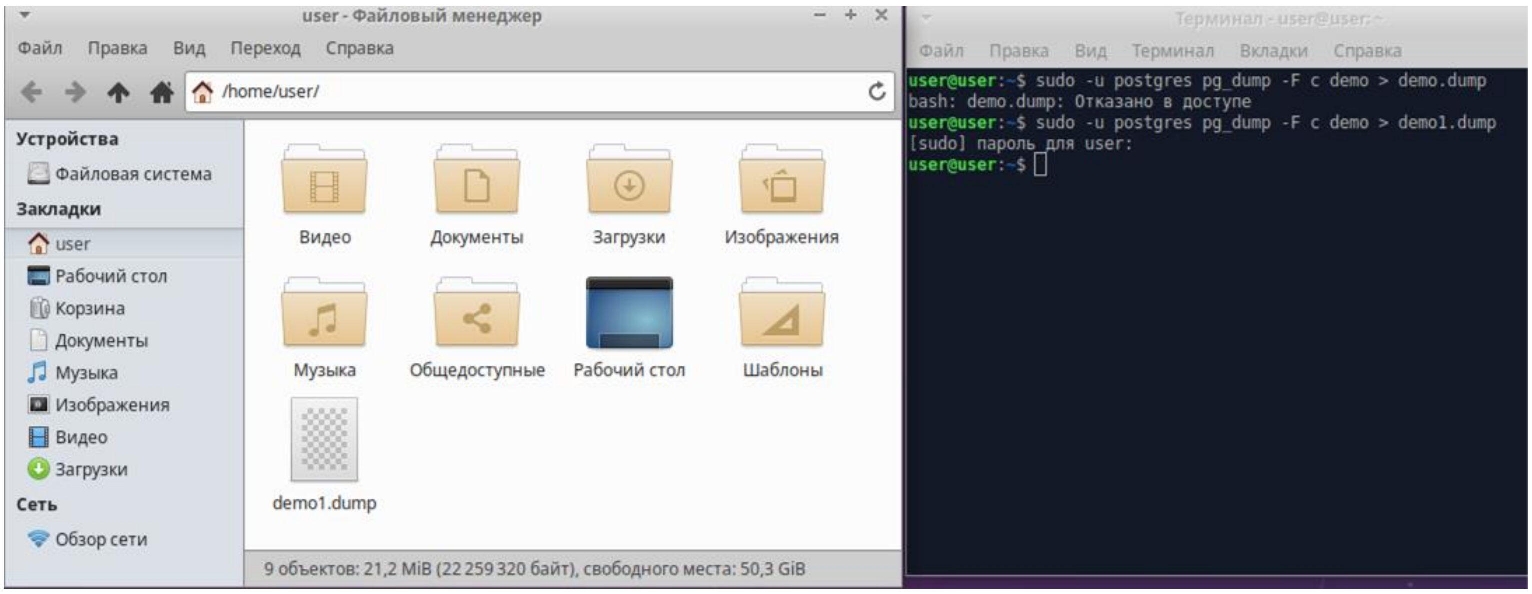
```
demo-# WHERE flight_time = min_time;
          hops           |                   flights                   | flight_time
-------------------------+---------------------------------------------+-------------
 {UKX,KJA,NOZ,OVB,MJZ,CNN} | {PG0022,PG0352,PG0297,PG0390,PG0035} | 10:10:00
 {UKX,KJA,NOZ,OVB,MJZ,CNN} | {PG0022,PG0351,PG0297,PG0390,PG0035} | 10:10:00
 {UKX,KJA,NOZ,OVB,MJZ,CNN} | {PG0022,PG0352,PG0298,PG0390,PG0035} | 10:10:00
 {UKX,KJA,NOZ,OVB,MJZ,CNN} | {PG0022,PG0351,PG0298,PG0390,PG0035} | 10:10:00
 {UKX,KJA,NOZ,OVB,MJZ,CNN} | {PG0022,PG0352,PG0297,PG0390,PG0036} | 10:10:00
 {UKX,KJA,NOZ,OVB,MJZ,CNN} | {PG0022,PG0351,PG0297,PG0390,PG0036} | 10:10:00
 {UKX,KJA,NOZ,OVB,MJZ,CNN} | {PG0022,PG0352,PG0298,PG0390,PG0036} | 10:10:00
 {UKX,KJA,NOZ,OVB,MJZ,CNN} | {PG0022,PG0351,PG0298,PG0390,PG0036} | 10:10:00
(8 rows)

demo=#
```

7. Найдите расстояние между Калининградом (KGD) и Петропавловском-Камчатским (PKC).
Решение.



```
demo=# CREATE EXTENSION IF NOT EXISTS cube;
CREATE EXTENSION
demo=# CREATE EXTENSION IF NOT EXISTS earthdistance;
CREATE EXTENSION
demo=# SELECT round(
demo(# (a_from.coordinates <@> a_to.coordinates) *
demo(# 1.609344
demo(# )
demo-# FROM airports a_from,
demo-# airports a_to
demo-# WHERE a_from.airport_code = 'KGD'
demo-# AND a_to.airport_code = 'PKC';
 round
-------
  7392
(1 row)

demo=#
```

Бэкап, созданной базы.



```
user@user:~$ sudo -u postgres pg_dump -F c demo > demo.dump
bash: demo.dump: Отказано в доступе
user@user:~$ sudo -u postgres pg_dump -F c demo > demo1.dump
[sudo] пароль для user:
user@user:~$
```