Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Программная инженерия
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №8

Повышение производительности
тема

Преподаватель			Вожжов А.Д.
		подпись, дата	инициалы, фамилия
Студент КИ23-17/1б	032318988		Александров Е.А.
	номер зачетной книжки	подпись, дата	инициалы, фамилия

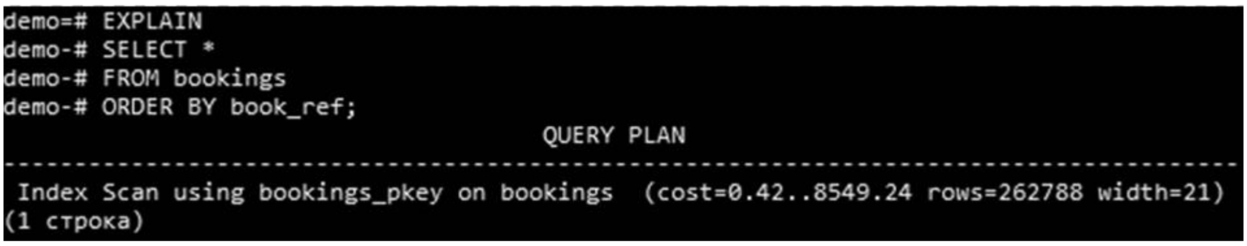
Красноярск 2025

1 Цель

Изучить основы повышения производительности. Выполнить указанные в файле задания.

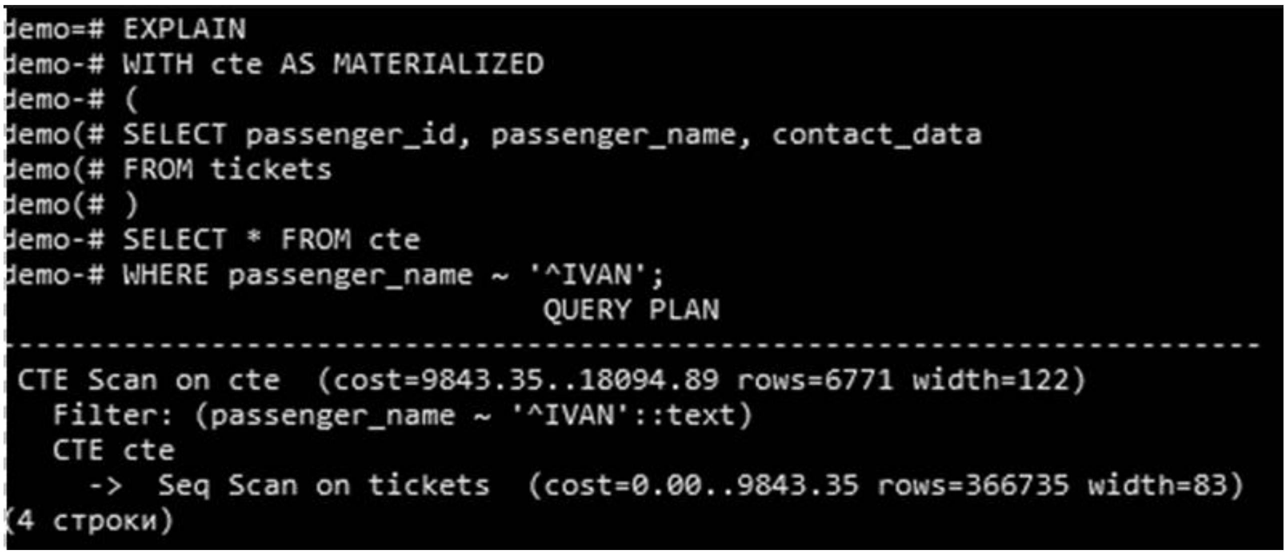
2 Ход работы

Результат выполнения заданий показан на рисунках с 1 по 23.



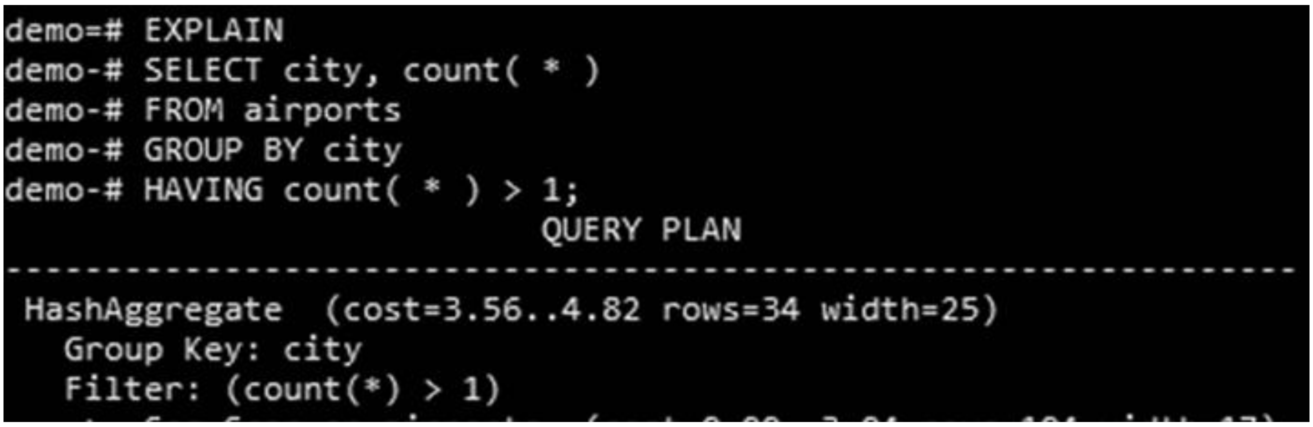
```
demo=# EXPLAIN
demo-# SELECT *
demo-# FROM bookings
demo-# ORDER BY book_ref;
                              QUERY PLAN
--------------------------------------------------------------------------------
 Index Scan using bookings_pkey on bookings  (cost=0.42..8549.24 rows=262788 width=21)
(1 строка)
```

Рисунок 1 – Задание 1



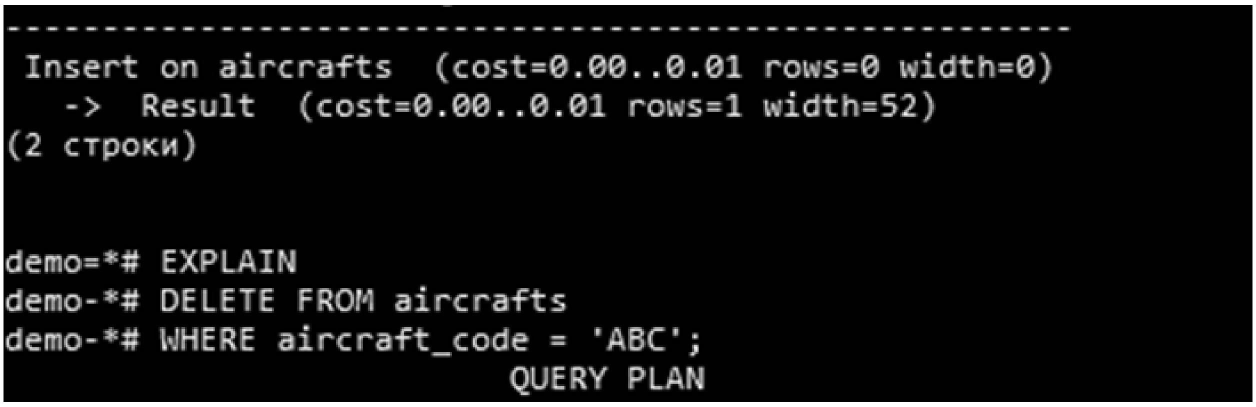
```
demo=# EXPLAIN
demo-# WITH cte AS MATERIALIZED
demo-# (
demo(# SELECT passenger_id, passenger_name, contact_data
demo(# FROM tickets
demo(# )
demo-# SELECT * FROM cte
demo-# WHERE passenger_name ~ '^IVAN';
                              QUERY PLAN
------------------------------------------------------------------------
 CTE Scan on cte  (cost=9843.35..18094.89 rows=6771 width=122)
   Filter: (passenger_name ~ '^IVAN'::text)
   CTE cte
     ->  Seq Scan on tickets  (cost=0.00..9843.35 rows=366735 width=83)
(4 строки)
```

Рисунок 2 – Задание 3



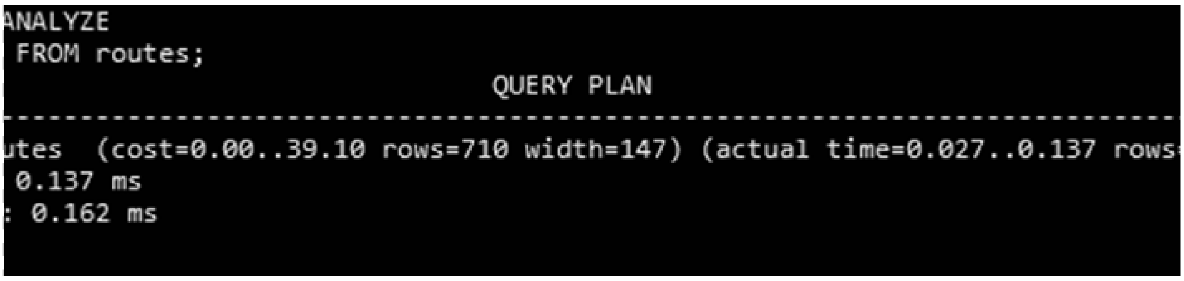
```
demo=# EXPLAIN
demo-# SELECT city, count( * )
demo-# FROM airports
demo-# GROUP BY city
demo-# HAVING count( * ) > 1;
                              QUERY PLAN
--------------------------------------------------------------------
 HashAggregate  (cost=3.56..4.82 rows=34 width=25)
   Group Key: city
   Filter: (count(*) > 1)
```

Рисунок 3 – Задание 5



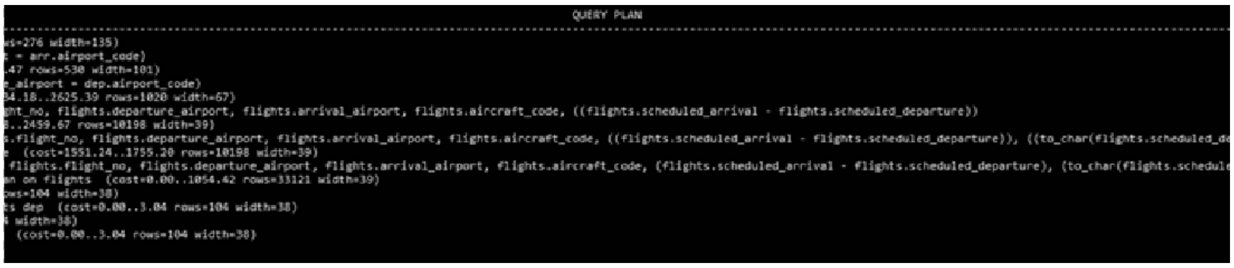
```
-------------------------------------------------------
 Insert on aircrafts  (cost=0.00..0.01 rows=0 width=0)
   ->  Result  (cost=0.00..0.01 rows=1 width=52)
(2 строки)


demo=*# EXPLAIN
demo-*# DELETE FROM aircrafts
demo-*# WHERE aircraft_code = 'ABC';
                          QUERY PLAN
```

Рисунок 4 – Задание 7



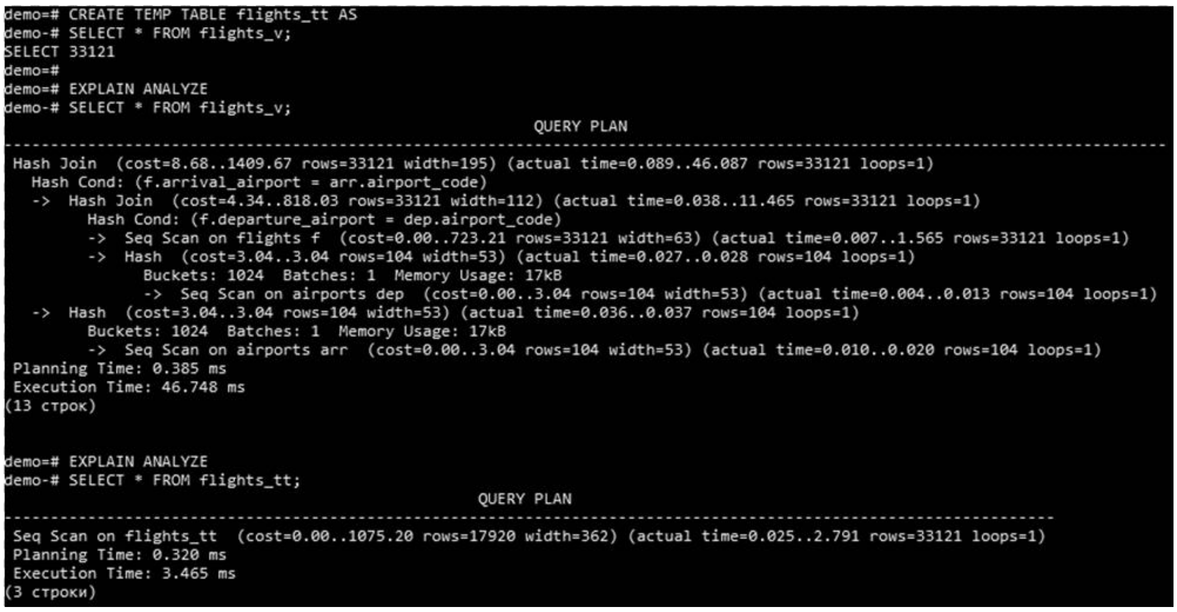
```
ANALYZE
 FROM routes;
                              QUERY PLAN
-----------------------------------------------------------------------------
utes  (cost=0.00..39.10 rows=710 width=147) (actual time=0.027..0.137 rows
 0.137 ms
: 0.162 ms
```

Рисунок 5 – Задание 9



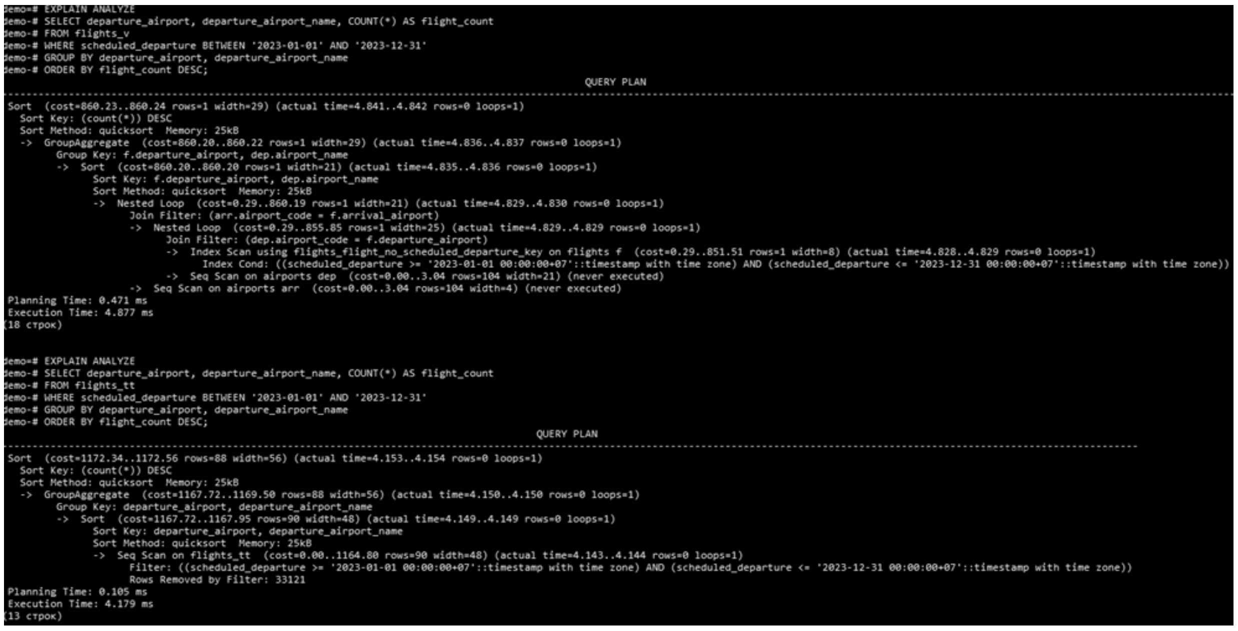
```
                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
s=276 width=135)
t = arr.airport_code)
.47 rows=530 width=101)
e_airport = dep.airport_code)
34.18..2625.39 rows=1020 width=67)
ght_no, flights.departure_airport, flights.arrival_airport, flights.aircraft_code, ((flights.scheduled_arrival - flights.scheduled_departure))
8..2459.67 rows=10198 width=39)
s.flight_no, flights.departure_airport, flights.arrival_airport, flights.aircraft_code, ((flights.scheduled_arrival - flights.scheduled_departure)), ((to_char(flights.scheduled_de
e  (cost=1551.24..1755.20 rows=10198 width=39)
 flights.flight_no, flights.departure_airport, flights.arrival_airport, flights.aircraft_code, (flights.scheduled_arrival - flights.scheduled_departure), (to_char(flights.schedule
an on flights  (cost=0.00..1054.42 rows=33121 width=39)
ows=104 width=38)
ts dep  (cost=0.00..3.04 rows=104 width=38)
4 width=38)
  (cost=0.00..3.04 rows=104 width=38)
```

Рисунок 6 – Задание 9



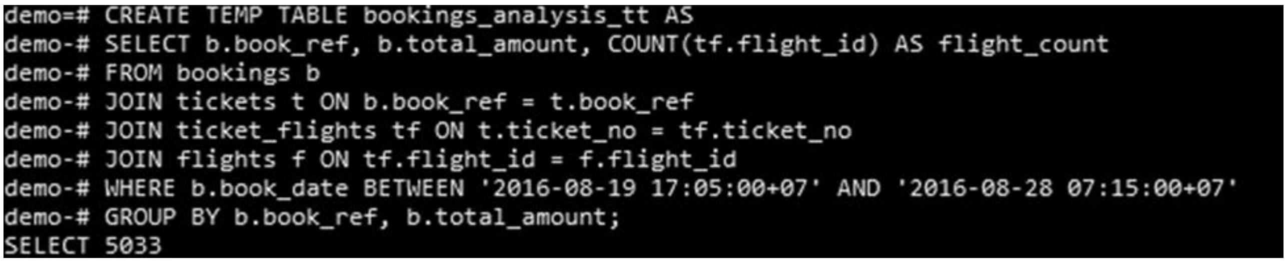
```
demo=# CREATE TEMP TABLE flights_tt AS
demo-# SELECT * FROM flights_v;
SELECT 33121
demo=#
demo=# EXPLAIN ANALYZE
demo-# SELECT * FROM flights_v;
                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=8.68..1409.67 rows=33121 width=195) (actual time=0.089..46.087 rows=33121 loops=1)
   Hash Cond: (f.arrival_airport = arr.airport_code)
   ->  Hash Join  (cost=4.34..818.03 rows=33121 width=112) (actual time=0.038..11.465 rows=33121 loops=1)
         Hash Cond: (f.departure_airport = dep.airport_code)
         ->  Seq Scan on flights f  (cost=0.00..723.21 rows=33121 width=63) (actual time=0.007..1.565 rows=33121 loops=1)
         ->  Hash  (cost=3.04..3.04 rows=104 width=53) (actual time=0.027..0.028 rows=104 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 17kB
               ->  Seq Scan on airports dep  (cost=0.00..3.04 rows=104 width=53) (actual time=0.004..0.013 rows=104 loops=1)
   ->  Hash  (cost=3.04..3.04 rows=104 width=53) (actual time=0.036..0.037 rows=104 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 17kB
         ->  Seq Scan on airports arr  (cost=0.00..3.04 rows=104 width=53) (actual time=0.010..0.020 rows=104 loops=1)
 Planning Time: 0.385 ms
 Execution Time: 46.748 ms
(13 строк)


demo=# EXPLAIN ANALYZE
demo-# SELECT * FROM flights_tt;
                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Seq Scan on flights_tt  (cost=0.00..1075.20 rows=17920 width=362) (actual time=0.025..2.791 rows=33121 loops=1)
 Planning Time: 0.320 ms
 Execution Time: 3.465 ms
(3 строки)
```

Рисунок 7 – Задание 11



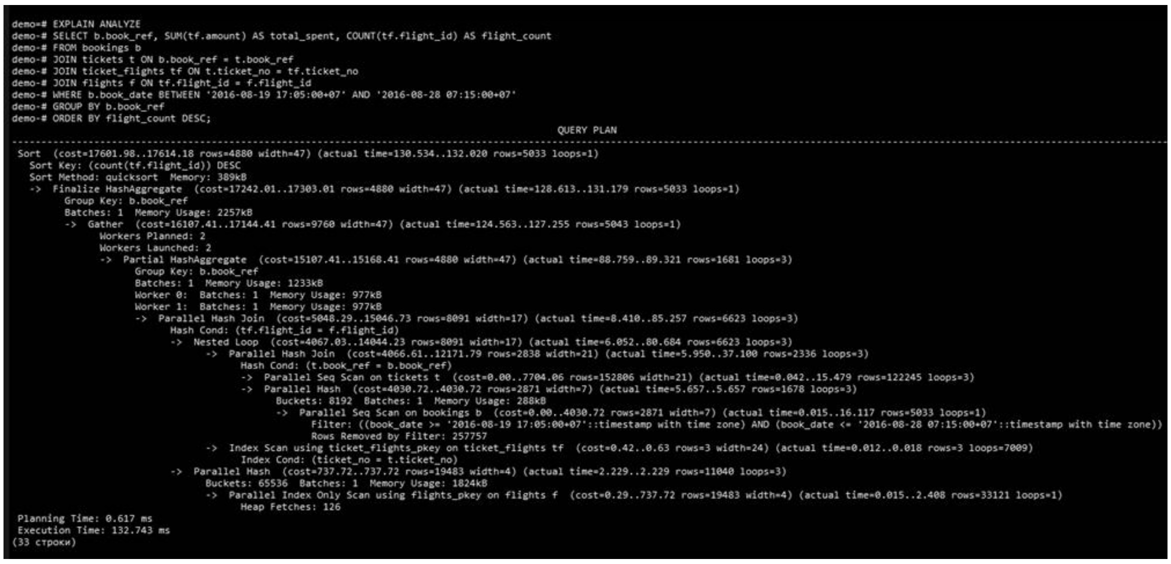
```
demo=# EXPLAIN ANALYZE
demo-# SELECT departure_airport, departure_airport_name, COUNT(*) AS flight_count
demo-# FROM flights_v
demo-# WHERE scheduled_departure BETWEEN '2023-01-01' AND '2023-12-31'
demo-# GROUP BY departure_airport, departure_airport_name
demo-# ORDER BY flight_count DESC;
                                                                      QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=860.23..860.24 rows=1 width=29) (actual time=4.841..4.842 rows=0 loops=1)
   Sort Key: (count(*)) DESC
   Sort Method: quicksort  Memory: 25kB
   ->  GroupAggregate  (cost=860.20..860.22 rows=1 width=29) (actual time=4.836..4.837 rows=0 loops=1)
         Group Key: f.departure_airport, dep.airport_name
         ->  Sort  (cost=860.20..860.20 rows=1 width=21) (actual time=4.835..4.836 rows=0 loops=1)
               Sort Key: f.departure_airport, dep.airport_name
               Sort Method: quicksort  Memory: 25kB
               ->  Nested Loop  (cost=0.29..860.19 rows=1 width=21) (actual time=4.829..4.830 rows=0 loops=1)
                     Join Filter: (arr.airport_code = f.arrival_airport)
                     ->  Nested Loop  (cost=0.29..855.85 rows=1 width=25) (actual time=4.829..4.829 rows=0 loops=1)
                           Join Filter: (dep.airport_code = f.departure_airport)
                           ->  Index Scan using flights_flight_no_scheduled_departure_key on flights f  (cost=0.29..851.51 rows=1 width=8) (actual time=4.828..4.829 rows=0 loops=1)
                                 Index Cond: ((scheduled_departure >= '2023-01-01 00:00:00+07'::timestamp with time zone) AND (scheduled_departure <= '2023-12-31 00:00:00+07'::timestamp with time zone))
                           ->  Seq Scan on airports dep  (cost=0.00..3.04 rows=104 width=21) (never executed)
                     ->  Seq Scan on airports arr  (cost=0.00..3.04 rows=104 width=4) (never executed)
 Planning Time: 0.471 ms
 Execution Time: 4.877 ms
(18 строк)


demo=# EXPLAIN ANALYZE
demo-# SELECT departure_airport, departure_airport_name, COUNT(*) AS flight_count
demo-# FROM flights_tt
demo-# WHERE scheduled_departure BETWEEN '2023-01-01' AND '2023-12-31'
demo-# GROUP BY departure_airport, departure_airport_name
demo-# ORDER BY flight_count DESC;
                                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=1172.34..1172.56 rows=88 width=56) (actual time=4.153..4.154 rows=0 loops=1)
   Sort Key: (count(*)) DESC
   Sort Method: quicksort  Memory: 25kB
   ->  GroupAggregate  (cost=1167.72..1169.50 rows=88 width=56) (actual time=4.150..4.150 rows=0 loops=1)
         Group Key: departure_airport, departure_airport_name
         ->  Sort  (cost=1167.72..1167.95 rows=90 width=48) (actual time=4.149..4.149 rows=0 loops=1)
               Sort Key: departure_airport, departure_airport_name
               Sort Method: quicksort  Memory: 25kB
               ->  Seq Scan on flights_tt  (cost=0.00..1164.80 rows=90 width=48) (actual time=4.143..4.144 rows=0 loops=1)
                     Filter: ((scheduled_departure >= '2023-01-01 00:00:00+07'::timestamp with time zone) AND (scheduled_departure <= '2023-12-31 00:00:00+07'::timestamp with time zone))
                     Rows Removed by Filter: 33121
 Planning Time: 0.105 ms
 Execution Time: 4.179 ms
(13 строк)
```

Рисунок 8 – Задание 11



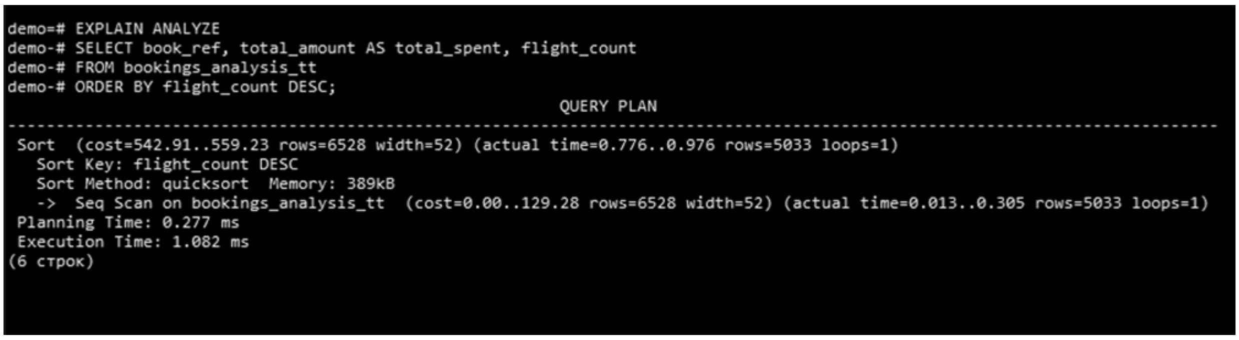
```
demo=# CREATE TEMP TABLE bookings_analysis_tt AS
demo-# SELECT b.book_ref, b.total_amount, COUNT(tf.flight_id) AS flight_count
demo-# FROM bookings b
demo-# JOIN tickets t ON b.book_ref = t.book_ref
demo-# JOIN ticket_flights tf ON t.ticket_no = tf.ticket_no
demo-# JOIN flights f ON tf.flight_id = f.flight_id
demo-# WHERE b.book_date BETWEEN '2016-08-19 17:05:00+07' AND '2016-08-28 07:15:00+07'
demo-# GROUP BY b.book_ref, b.total_amount;
SELECT 5033
```

Рисунок 9 – Задание 11



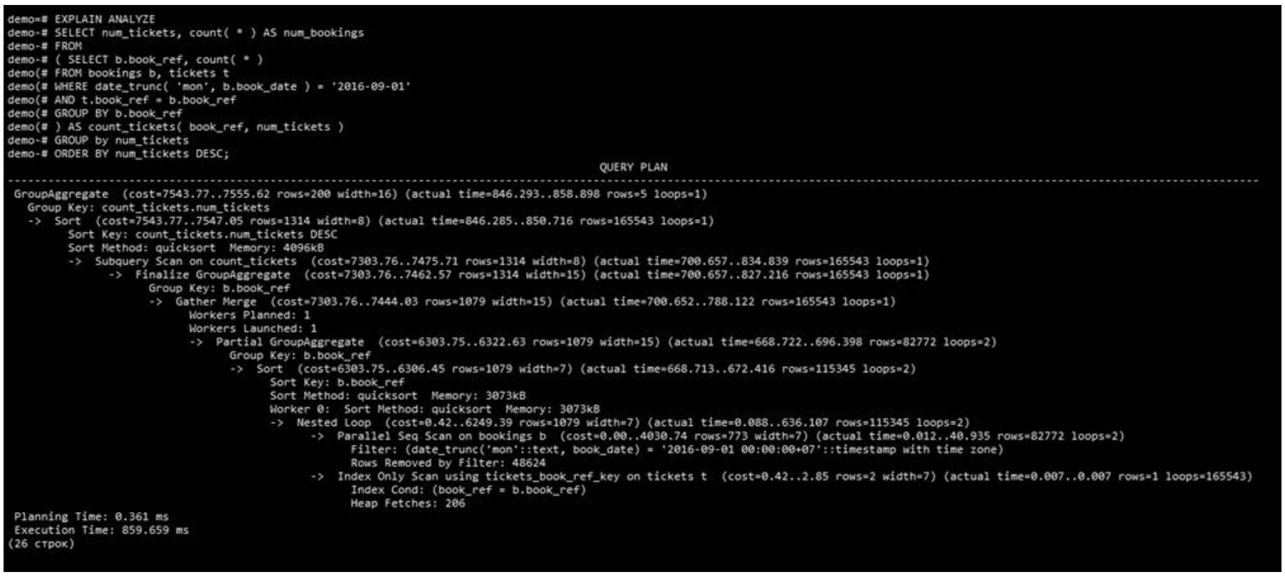
```
demo=# EXPLAIN ANALYZE
demo-# SELECT b.book_ref, SUM(tf.amount) AS total_spent, COUNT(tf.flight_id) AS flight_count
demo-# FROM bookings b
demo-# JOIN tickets t ON b.book_ref = t.book_ref
demo-# JOIN ticket_flights tf ON t.ticket_no = tf.ticket_no
demo-# JOIN flights f ON tf.flight_id = f.flight_id
demo-# WHERE b.book_date BETWEEN '2016-08-19 17:05:00+07' AND '2016-08-28 07:15:00+07'
demo-# GROUP BY b.book_ref
demo-# ORDER BY flight_count DESC;
                                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=17601.98..17614.18 rows=4880 width=47) (actual time=130.534..132.020 rows=5033 loops=1)
   Sort Key: (count(tf.flight_id)) DESC
   Sort Method: quicksort  Memory: 389kB
   ->  Finalize HashAggregate  (cost=17242.01..17303.01 rows=4880 width=47) (actual time=128.613..131.179 rows=5033 loops=1)
         Group Key: b.book_ref
         Batches: 1  Memory Usage: 2257kB
         ->  Gather  (cost=16107.41..17144.41 rows=9760 width=47) (actual time=124.563..127.255 rows=5043 loops=1)
               Workers Planned: 2
               Workers Launched: 2
               ->  Partial HashAggregate  (cost=15107.41..15168.41 rows=4880 width=47) (actual time=88.759..89.321 rows=1681 loops=3)
                     Group Key: b.book_ref
                     Batches: 1  Memory Usage: 1233kB
                     Worker 0:  Batches: 1  Memory Usage: 977kB
                     Worker 1:  Batches: 1  Memory Usage: 977kB
                     ->  Parallel Hash Join  (cost=5048.29..15046.73 rows=8091 width=17) (actual time=8.410..85.257 rows=6623 loops=3)
                           Hash Cond: (tf.flight_id = f.flight_id)
                           ->  Nested Loop  (cost=4067.03..14044.23 rows=8091 width=17) (actual time=6.052..80.684 rows=6623 loops=3)
                                 ->  Parallel Hash Join  (cost=4066.61..12171.79 rows=2838 width=21) (actual time=5.950..37.100 rows=2336 loops=3)
                                       Hash Cond: (t.book_ref = b.book_ref)
                                       ->  Parallel Seq Scan on tickets t  (cost=0.00..7704.06 rows=152806 width=21) (actual time=0.042..15.479 rows=122245 loops=3)
                                       ->  Parallel Hash  (cost=4030.72..4030.72 rows=2871 width=7) (actual time=5.657..5.657 rows=1678 loops=3)
                                             Buckets: 8192  Batches: 1  Memory Usage: 288kB
                                             ->  Parallel Seq Scan on bookings b  (cost=0.00..4030.72 rows=2871 width=7) (actual time=0.015..16.117 rows=5033 loops=1)
                                                   Filter: ((book_date >= '2016-08-19 17:05:00+07'::timestamp with time zone) AND (book_date <= '2016-08-28 07:15:00+07'::timestamp with time zone))
                                                   Rows Removed by Filter: 257757
                                 ->  Index Scan using ticket_flights_pkey on ticket_flights tf  (cost=0.42..0.63 rows=3 width=24) (actual time=0.012..0.018 rows=3 loops=7009)
                                       Index Cond: (ticket_no = t.ticket_no)
                           ->  Parallel Hash  (cost=737.72..737.72 rows=19483 width=4) (actual time=2.229..2.229 rows=11040 loops=3)
                                 Buckets: 65536  Batches: 1  Memory Usage: 1824kB
                                 ->  Parallel Index Only Scan using flights_pkey on flights f  (cost=0.29..737.72 rows=19483 width=4) (actual time=0.015..2.408 rows=33121 loops=1)
                                       Heap Fetches: 126
 Planning Time: 0.617 ms
 Execution Time: 132.743 ms
(33 строки)
```

Рисунок 10 – Задание 11



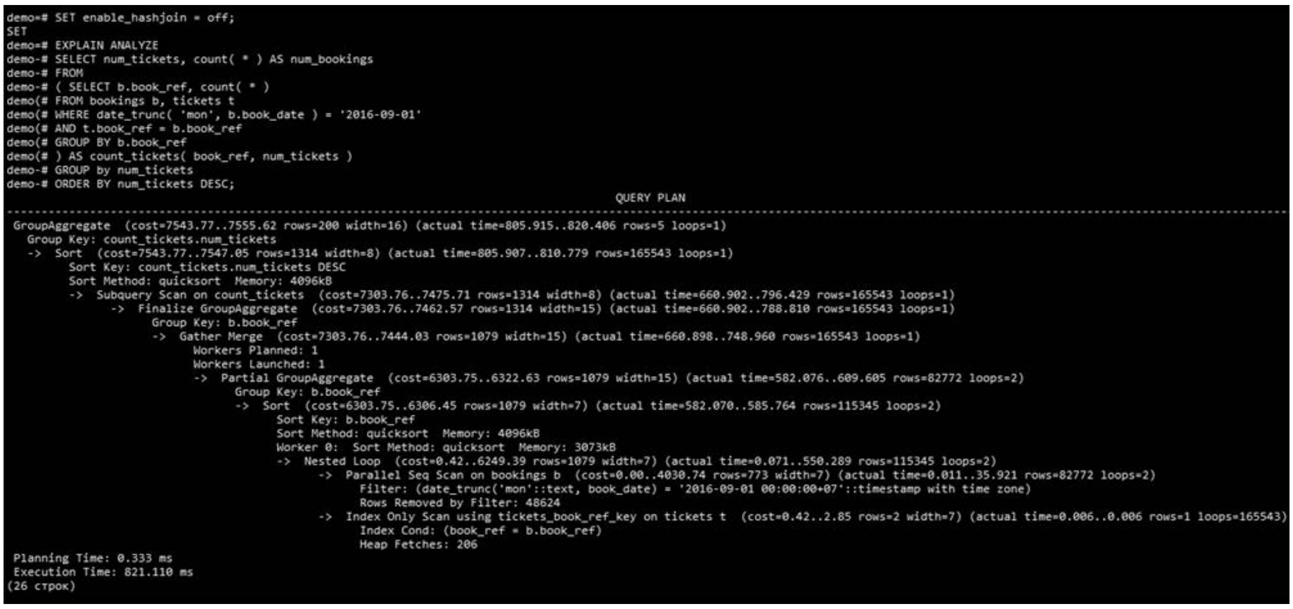
```
demo=# EXPLAIN ANALYZE
demo-# SELECT book_ref, total_amount AS total_spent, flight_count
demo-# FROM bookings_analysis_tt
demo-# ORDER BY flight_count DESC;
                                                            QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 Sort  (cost=542.91..559.23 rows=6528 width=52) (actual time=0.776..0.976 rows=5033 loops=1)
   Sort Key: flight_count DESC
   Sort Method: quicksort  Memory: 389kB
   ->  Seq Scan on bookings_analysis_tt  (cost=0.00..129.28 rows=6528 width=52) (actual time=0.013..0.305 rows=5033 loops=1)
 Planning Time: 0.277 ms
 Execution Time: 1.082 ms
(6 строк)
```

Рисунок 11 – Задание 11



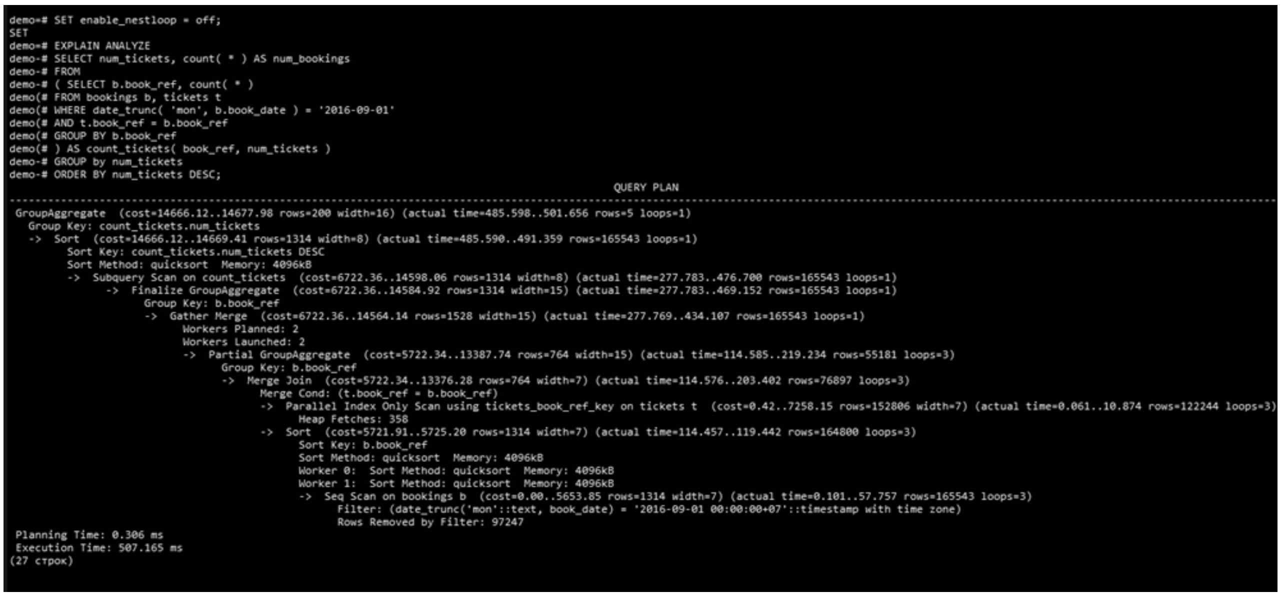
```
demo=# EXPLAIN ANALYZE
demo-# SELECT num_tickets, count( * ) AS num_bookings
demo-# FROM
demo-# ( SELECT b.book_ref, count( * )
demo(# FROM bookings b, tickets t
demo(# WHERE date_trunc( 'mon', b.book_date ) = '2016-09-01'
demo(# AND t.book_ref = b.book_ref
demo(# GROUP BY b.book_ref
demo(# ) AS count_tickets( book_ref, num_tickets )
demo-# GROUP by num_tickets
demo-# ORDER BY num_tickets DESC;
                                                            QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=7543.77..7555.62 rows=200 width=16) (actual time=846.293..858.898 rows=5 loops=1)
   Group Key: count_tickets.num_tickets
   ->  Sort  (cost=7543.77..7547.05 rows=1314 width=8) (actual time=846.285..850.716 rows=165543 loops=1)
         Sort Key: count_tickets.num_tickets DESC
         Sort Method: quicksort  Memory: 4096kB
         ->  Subquery Scan on count_tickets  (cost=7303.76..7475.71 rows=1314 width=8) (actual time=700.657..834.839 rows=165543 loops=1)
               ->  Finalize GroupAggregate  (cost=7303.76..7462.57 rows=1314 width=15) (actual time=700.657..827.216 rows=165543 loops=1)
                     Group Key: b.book_ref
                     ->  Gather Merge  (cost=7303.76..7444.03 rows=1079 width=15) (actual time=700.652..788.122 rows=165543 loops=1)
                           Workers Planned: 1
                           Workers Launched: 1
                           ->  Partial GroupAggregate  (cost=6303.75..6322.63 rows=1079 width=15) (actual time=668.722..696.398 rows=82772 loops=2)
                                 Group Key: b.book_ref
                                 ->  Sort  (cost=6303.75..6306.45 rows=1079 width=7) (actual time=668.713..672.416 rows=115345 loops=2)
                                       Sort Key: b.book_ref
                                       Sort Method: quicksort  Memory: 3073kB
                                       Worker 0:  Sort Method: quicksort  Memory: 3073kB
                                       ->  Nested Loop  (cost=0.42..6249.39 rows=1079 width=7) (actual time=0.088..636.107 rows=115345 loops=2)
                                             ->  Parallel Seq Scan on bookings b  (cost=0.00..4030.74 rows=773 width=7) (actual time=0.012..40.935 rows=82772 loops=2)
                                                   Filter: (date_trunc('mon'::text, book_date) = '2016-09-01 00:00:00+07'::timestamp with time zone)
                                                   Rows Removed by Filter: 48624
                                             ->  Index Only Scan using tickets_book_ref_key on tickets t  (cost=0.42..2.85 rows=2 width=7) (actual time=0.007..0.007 rows=1 loops=165543)
                                                   Index Cond: (book_ref = b.book_ref)
                                                   Heap Fetches: 206
 Planning Time: 0.361 ms
 Execution Time: 859.659 ms
(26 строк)
```

Рисунок 12 – Задание 13



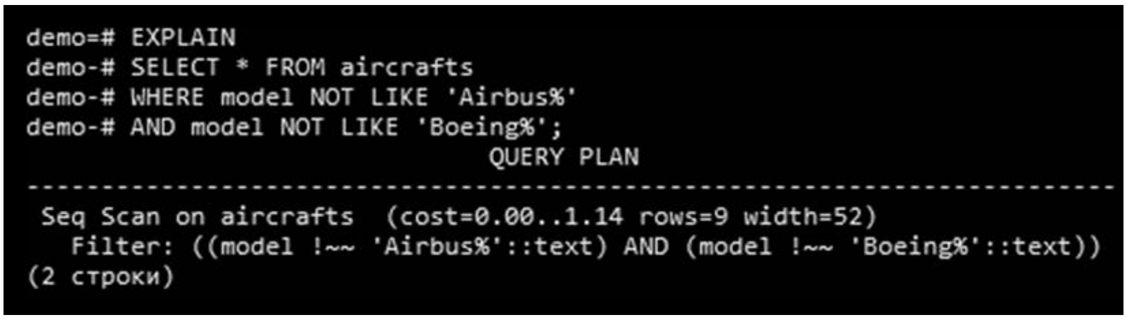
```
demo=# SET enable_hashjoin = off;
SET
demo=# EXPLAIN ANALYZE
demo-# SELECT num_tickets, count( * ) AS num_bookings
demo-# FROM
demo-# ( SELECT b.book_ref, count( * )
demo(# FROM bookings b, tickets t
demo(# WHERE date_trunc( 'mon', b.book_date ) = '2016-09-01'
demo(# AND t.book_ref = b.book_ref
demo(# GROUP BY b.book_ref
demo(# ) AS count_tickets( book_ref, num_tickets )
demo-# GROUP by num_tickets
demo-# ORDER BY num_tickets DESC;
                                                            QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=7543.77..7555.62 rows=200 width=16) (actual time=805.915..820.406 rows=5 loops=1)
   Group Key: count_tickets.num_tickets
   ->  Sort  (cost=7543.77..7547.05 rows=1314 width=8) (actual time=805.907..810.779 rows=165543 loops=1)
         Sort Key: count_tickets.num_tickets DESC
         Sort Method: quicksort  Memory: 4096kB
         ->  Subquery Scan on count_tickets  (cost=7303.76..7475.71 rows=1314 width=8) (actual time=660.902..796.429 rows=165543 loops=1)
               ->  Finalize GroupAggregate  (cost=7303.76..7462.57 rows=1314 width=15) (actual time=660.902..788.810 rows=165543 loops=1)
                     Group Key: b.book_ref
                     ->  Gather Merge  (cost=7303.76..7444.03 rows=1079 width=15) (actual time=660.898..748.960 rows=165543 loops=1)
                           Workers Planned: 1
                           Workers Launched: 1
                           ->  Partial GroupAggregate  (cost=6303.75..6322.63 rows=1079 width=15) (actual time=582.076..609.605 rows=82772 loops=2)
                                 Group Key: b.book_ref
                                 ->  Sort  (cost=6303.75..6306.45 rows=1079 width=7) (actual time=582.070..585.764 rows=115345 loops=2)
                                       Sort Key: b.book_ref
                                       Sort Method: quicksort  Memory: 4096kB
                                       Worker 0:  Sort Method: quicksort  Memory: 3073kB
                                       ->  Nested Loop  (cost=0.42..6249.39 rows=1079 width=7) (actual time=0.071..550.289 rows=115345 loops=2)
                                             ->  Parallel Seq Scan on bookings b  (cost=0.00..4030.74 rows=773 width=7) (actual time=0.011..35.921 rows=82772 loops=2)
                                                   Filter: (date_trunc('mon'::text, book_date) = '2016-09-01 00:00:00+07'::timestamp with time zone)
                                                   Rows Removed by Filter: 48624
                                             ->  Index Only Scan using tickets_book_ref_key on tickets t  (cost=0.42..2.85 rows=2 width=7) (actual time=0.006..0.006 rows=1 loops=165543)
                                                   Index Cond: (book_ref = b.book_ref)
                                                   Heap Fetches: 206
 Planning Time: 0.333 ms
 Execution Time: 821.110 ms
(26 строк)
```

Рисунок 13 – Задание 13



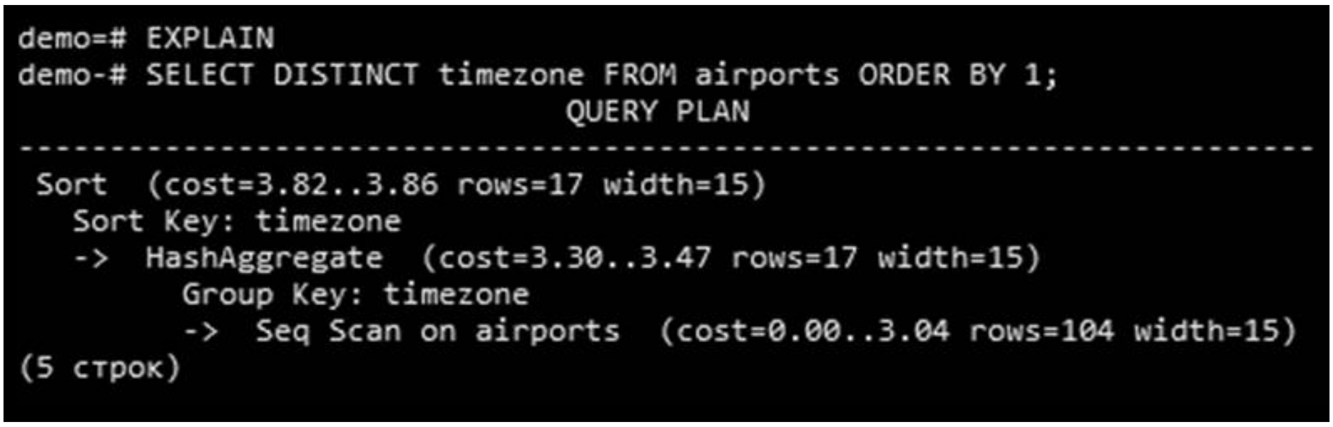
```
demo=# SET enable_nestloop = off;
SET
demo=# EXPLAIN ANALYZE
demo-# SELECT num_tickets, count( * ) AS num_bookings
demo-# FROM
demo-# ( SELECT b.book_ref, count( * )
demo(# FROM bookings b, tickets t
demo(# WHERE date_trunc( 'mon', b.book_date ) = '2016-09-01'
demo(# AND t.book_ref = b.book_ref
demo(# GROUP BY b.book_ref
demo(# ) AS count_tickets( book_ref, num_tickets )
demo-# GROUP by num_tickets
demo-# ORDER BY num_tickets DESC;
                                                                                              QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=14666.12..14677.98 rows=200 width=16) (actual time=485.598..501.656 rows=5 loops=1)
   Group Key: count_tickets.num_tickets
   ->  Sort  (cost=14666.12..14669.41 rows=1314 width=8) (actual time=485.590..491.359 rows=165543 loops=1)
         Sort Key: count_tickets.num_tickets DESC
         Sort Method: quicksort  Memory: 4096kB
         ->  Subquery Scan on count_tickets  (cost=6722.36..14598.06 rows=1314 width=8) (actual time=277.783..476.700 rows=165543 loops=1)
               ->  Finalize GroupAggregate  (cost=6722.36..14584.92 rows=1314 width=15) (actual time=277.783..469.152 rows=165543 loops=1)
                     Group Key: b.book_ref
                     ->  Gather Merge  (cost=6722.36..14564.14 rows=1528 width=15) (actual time=277.769..434.107 rows=165543 loops=1)
                           Workers Planned: 2
                           Workers Launched: 2
                           ->  Partial GroupAggregate  (cost=5722.34..13387.74 rows=764 width=15) (actual time=114.585..219.234 rows=55181 loops=3)
                                 Group Key: b.book_ref
                                 ->  Merge Join  (cost=5722.34..13376.28 rows=764 width=7) (actual time=114.576..203.402 rows=76897 loops=3)
                                       Merge Cond: (t.book_ref = b.book_ref)
                                       ->  Parallel Index Only Scan using tickets_book_ref_key on tickets t  (cost=0.42..7258.15 rows=152806 width=7) (actual time=0.061..10.874 rows=122244 loops=3)
                                             Heap Fetches: 358
                                       ->  Sort  (cost=5721.91..5725.20 rows=1314 width=7) (actual time=114.457..119.442 rows=164800 loops=3)
                                             Sort Key: b.book_ref
                                             Sort Method: quicksort  Memory: 4096kB
                                             Worker 0:  Sort Method: quicksort  Memory: 4096kB
                                             Worker 1:  Sort Method: quicksort  Memory: 4096kB
                                             ->  Seq Scan on bookings b  (cost=0.00..5653.85 rows=1314 width=7) (actual time=0.101..57.757 rows=165543 loops=3)
                                                   Filter: (date_trunc('mon'::text, book_date) = '2016-09-01 00:00:00+07'::timestamp with time zone)
                                                   Rows Removed by Filter: 97247
 Planning Time: 0.306 ms
 Execution Time: 507.165 ms
(27 строк)
```

Рисунок 14 – Задание 13



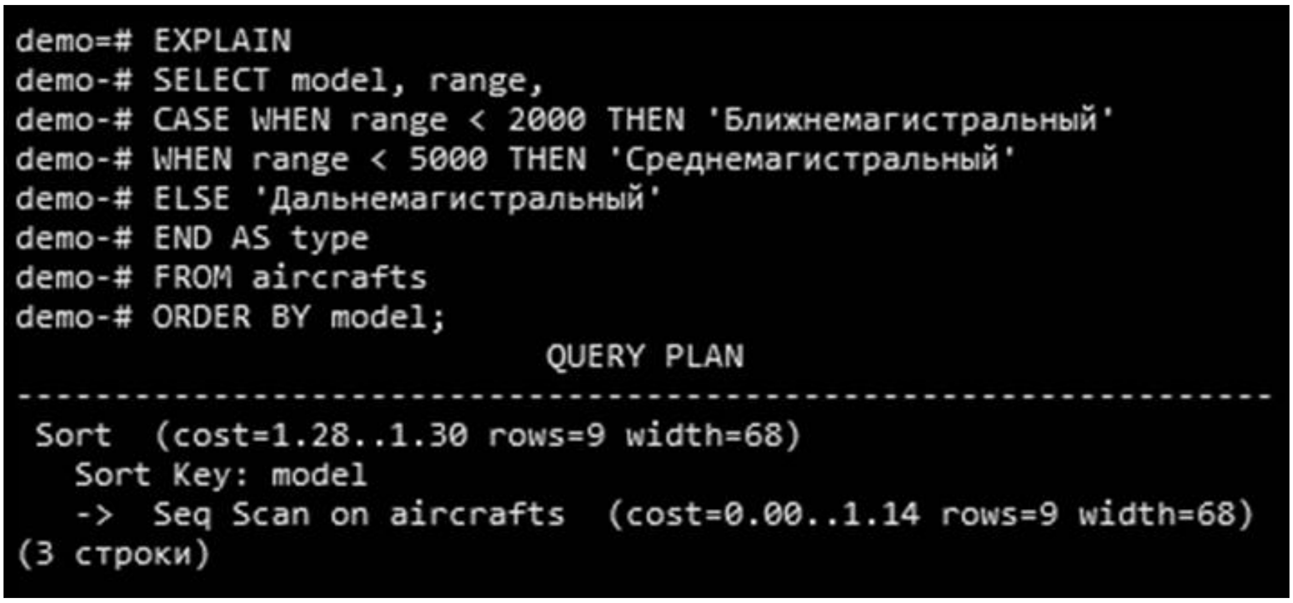
```
demo=# EXPLAIN
demo-# SELECT * FROM aircrafts
demo-# WHERE model NOT LIKE 'Airbus%'
demo-# AND model NOT LIKE 'Boeing%';
                               QUERY PLAN
-----------------------------------------------------------------------------
 Seq Scan on aircrafts  (cost=0.00..1.14 rows=9 width=52)
   Filter: ((model !~~ 'Airbus%'::text) AND (model !~~ 'Boeing%'::text))
(2 строки)
```

Рисунок 15 – Задание 15



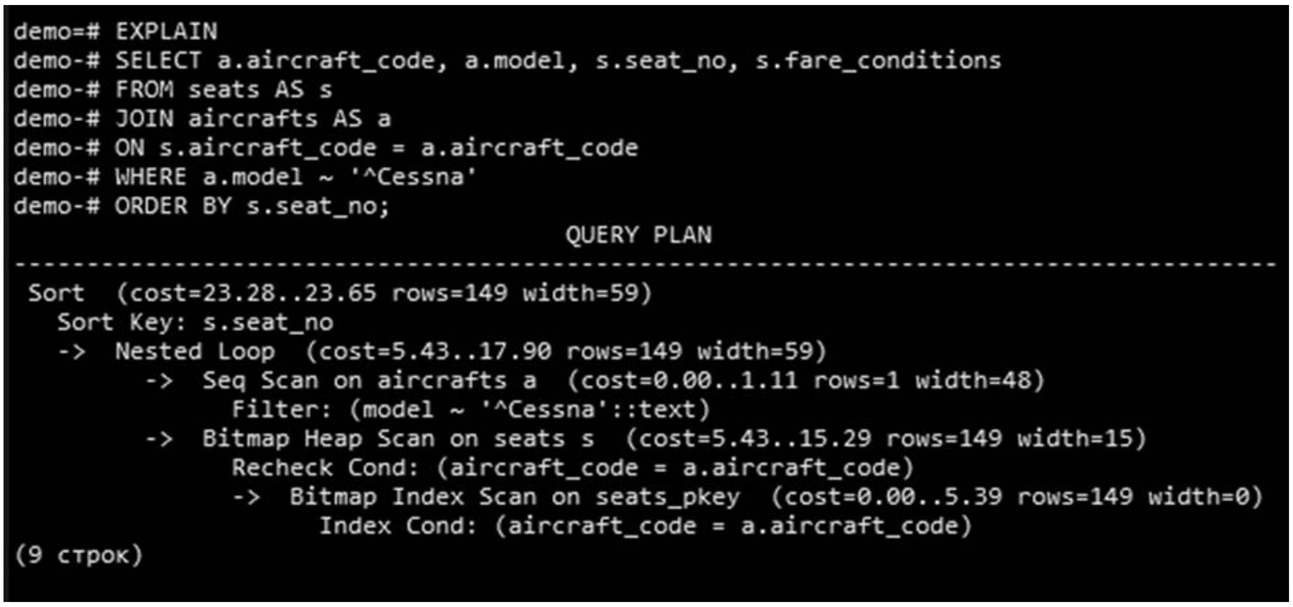
```
demo=# EXPLAIN
demo-# SELECT DISTINCT timezone FROM airports ORDER BY 1;
                               QUERY PLAN
-------------------------------------------------------------------------
 Sort  (cost=3.82..3.86 rows=17 width=15)
   Sort Key: timezone
   ->  HashAggregate  (cost=3.30..3.47 rows=17 width=15)
         Group Key: timezone
         ->  Seq Scan on airports  (cost=0.00..3.04 rows=104 width=15)
(5 строк)
```

Рисунок 16 – Задание 15



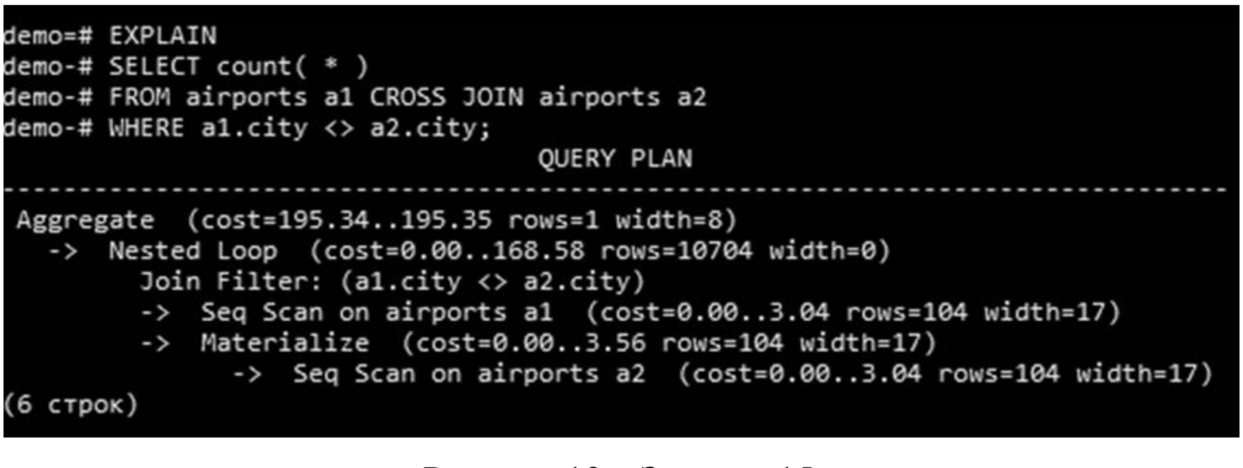
```
demo=# EXPLAIN
demo-# SELECT model, range,
demo-# CASE WHEN range < 2000 THEN 'Ближнемагистральный'
demo-# WHEN range < 5000 THEN 'Среднемагистральный'
demo-# ELSE 'Дальнемагистральный'
demo-# END AS type
demo-# FROM aircrafts
demo-# ORDER BY model;
                           QUERY PLAN
----------------------------------------------------------------
 Sort  (cost=1.28..1.30 rows=9 width=68)
   Sort Key: model
   ->  Seq Scan on aircrafts  (cost=0.00..1.14 rows=9 width=68)
(3 строки)
```

Рисунок 17 – Задание 15



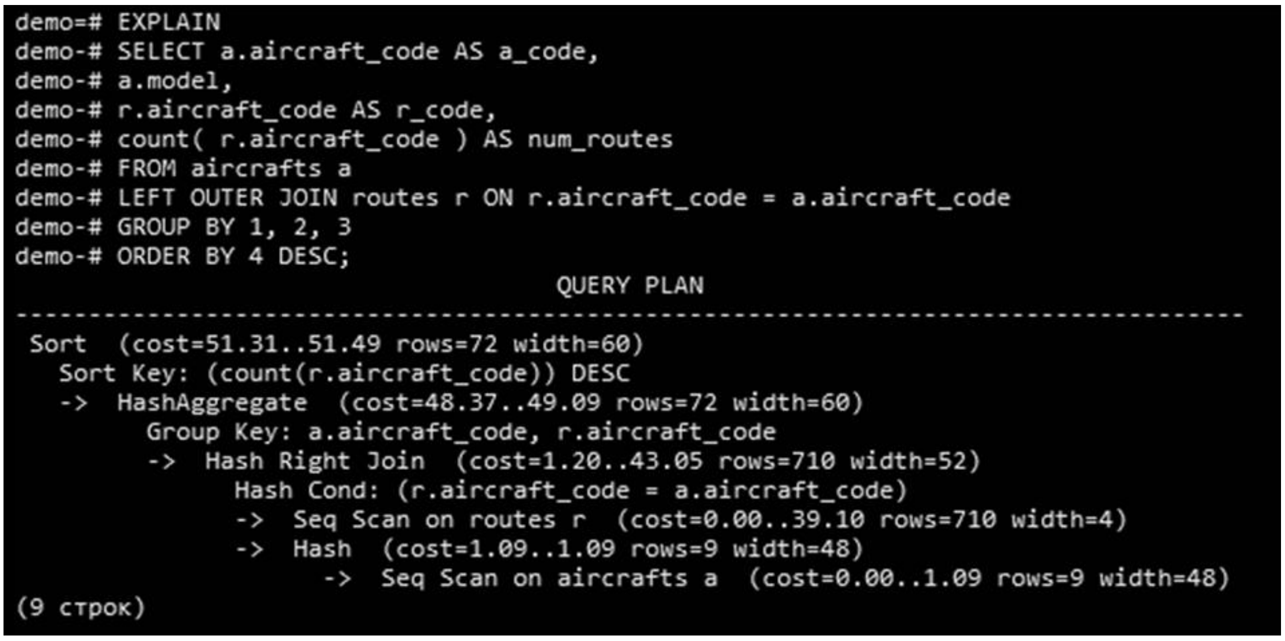
```
demo=# EXPLAIN
demo-# SELECT a.aircraft_code, a.model, s.seat_no, s.fare_conditions
demo-# FROM seats AS s
demo-# JOIN aircrafts AS a
demo-# ON s.aircraft_code = a.aircraft_code
demo-# WHERE a.model ~ '^Cessna'
demo-# ORDER BY s.seat_no;
                                      QUERY PLAN
------------------------------------------------------------------------------------
 Sort  (cost=23.28..23.65 rows=149 width=59)
   Sort Key: s.seat_no
   ->  Nested Loop  (cost=5.43..17.90 rows=149 width=59)
         ->  Seq Scan on aircrafts a  (cost=0.00..1.11 rows=1 width=48)
               Filter: (model ~ '^Cessna'::text)
         ->  Bitmap Heap Scan on seats s  (cost=5.43..15.29 rows=149 width=15)
               Recheck Cond: (aircraft_code = a.aircraft_code)
               ->  Bitmap Index Scan on seats_pkey  (cost=0.00..5.39 rows=149 width=0)
                     Index Cond: (aircraft_code = a.aircraft_code)
(9 строк)
```

Рисунок 18 – Задание 15



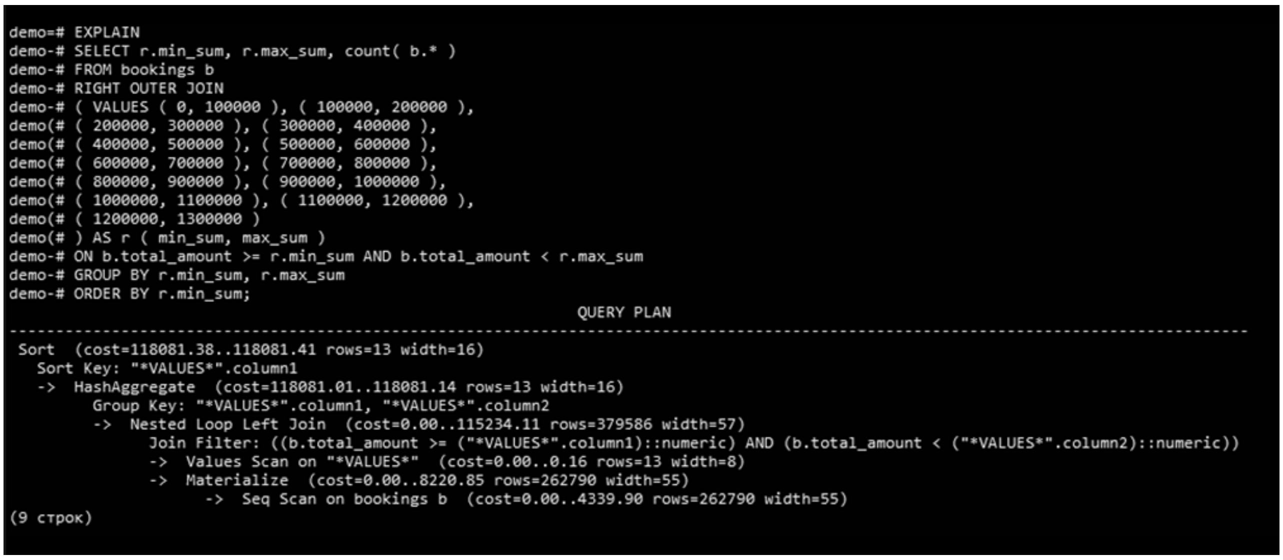
```
demo=# EXPLAIN
demo-# SELECT count( * )
demo-# FROM airports a1 CROSS JOIN airports a2
demo-# WHERE a1.city <> a2.city;
                               QUERY PLAN
------------------------------------------------------------------------------
 Aggregate  (cost=195.34..195.35 rows=1 width=8)
   ->  Nested Loop  (cost=0.00..168.58 rows=10704 width=0)
         Join Filter: (a1.city <> a2.city)
         ->  Seq Scan on airports a1  (cost=0.00..3.04 rows=104 width=17)
         ->  Materialize  (cost=0.00..3.56 rows=104 width=17)
               ->  Seq Scan on airports a2  (cost=0.00..3.04 rows=104 width=17)
(6 строк)
```

Рисунок 19 – Задание 15



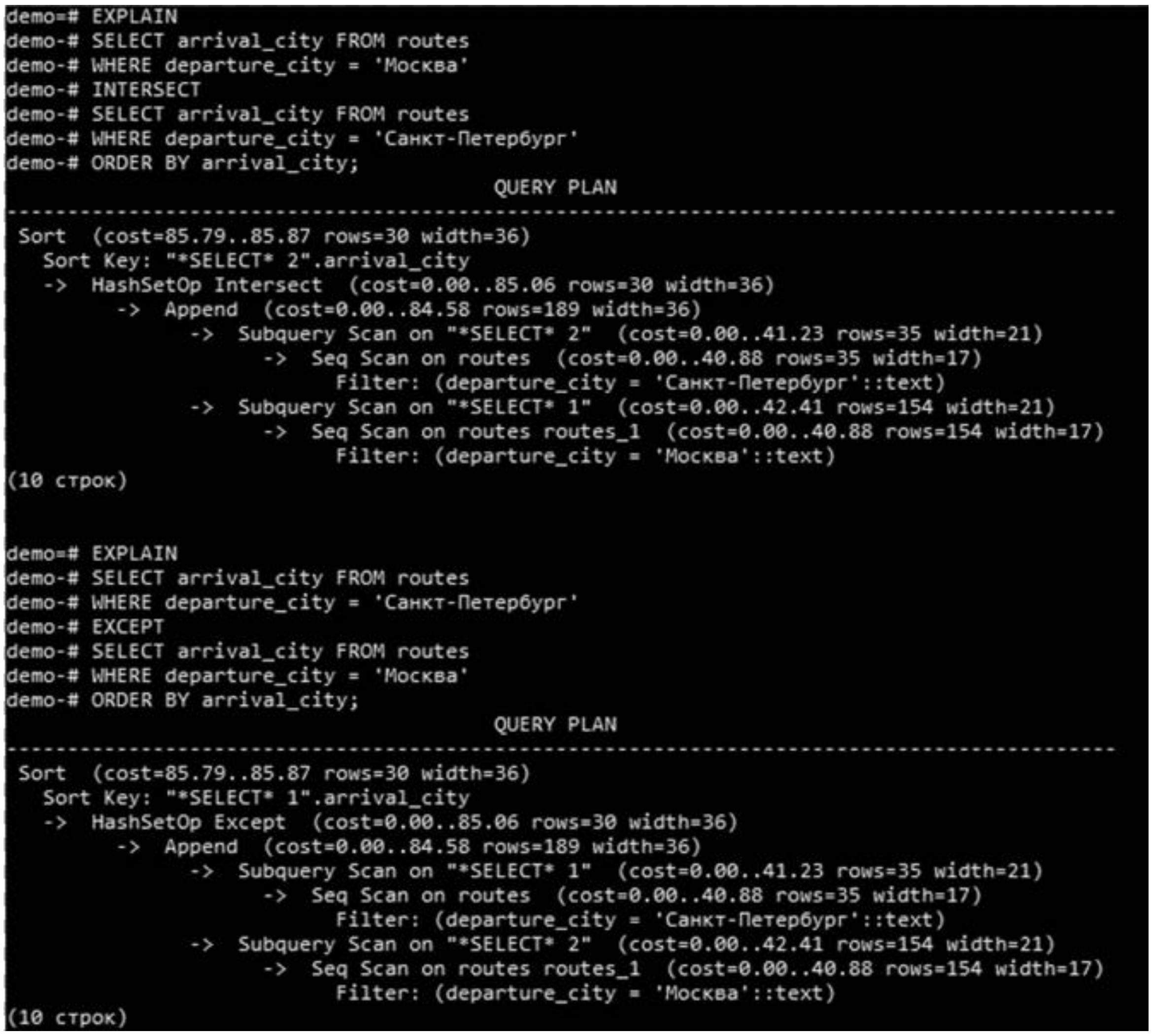
```
demo=# EXPLAIN
demo-# SELECT a.aircraft_code AS a_code,
demo-# a.model,
demo-# r.aircraft_code AS r_code,
demo-# count( r.aircraft_code ) AS num_routes
demo-# FROM aircrafts a
demo-# LEFT OUTER JOIN routes r ON r.aircraft_code = a.aircraft_code
demo-# GROUP BY 1, 2, 3
demo-# ORDER BY 4 DESC;
                                    QUERY PLAN
-----------------------------------------------------------------------------------
 Sort  (cost=51.31..51.49 rows=72 width=60)
   Sort Key: (count(r.aircraft_code)) DESC
   ->  HashAggregate  (cost=48.37..49.09 rows=72 width=60)
         Group Key: a.aircraft_code, r.aircraft_code
         ->  Hash Right Join  (cost=1.20..43.05 rows=710 width=52)
               Hash Cond: (r.aircraft_code = a.aircraft_code)
               ->  Seq Scan on routes r  (cost=0.00..39.10 rows=710 width=4)
               ->  Hash  (cost=1.09..1.09 rows=9 width=48)
                     ->  Seq Scan on aircrafts a  (cost=0.00..1.09 rows=9 width=48)
(9 строк)
```

Рисунок 20 – Задание 15



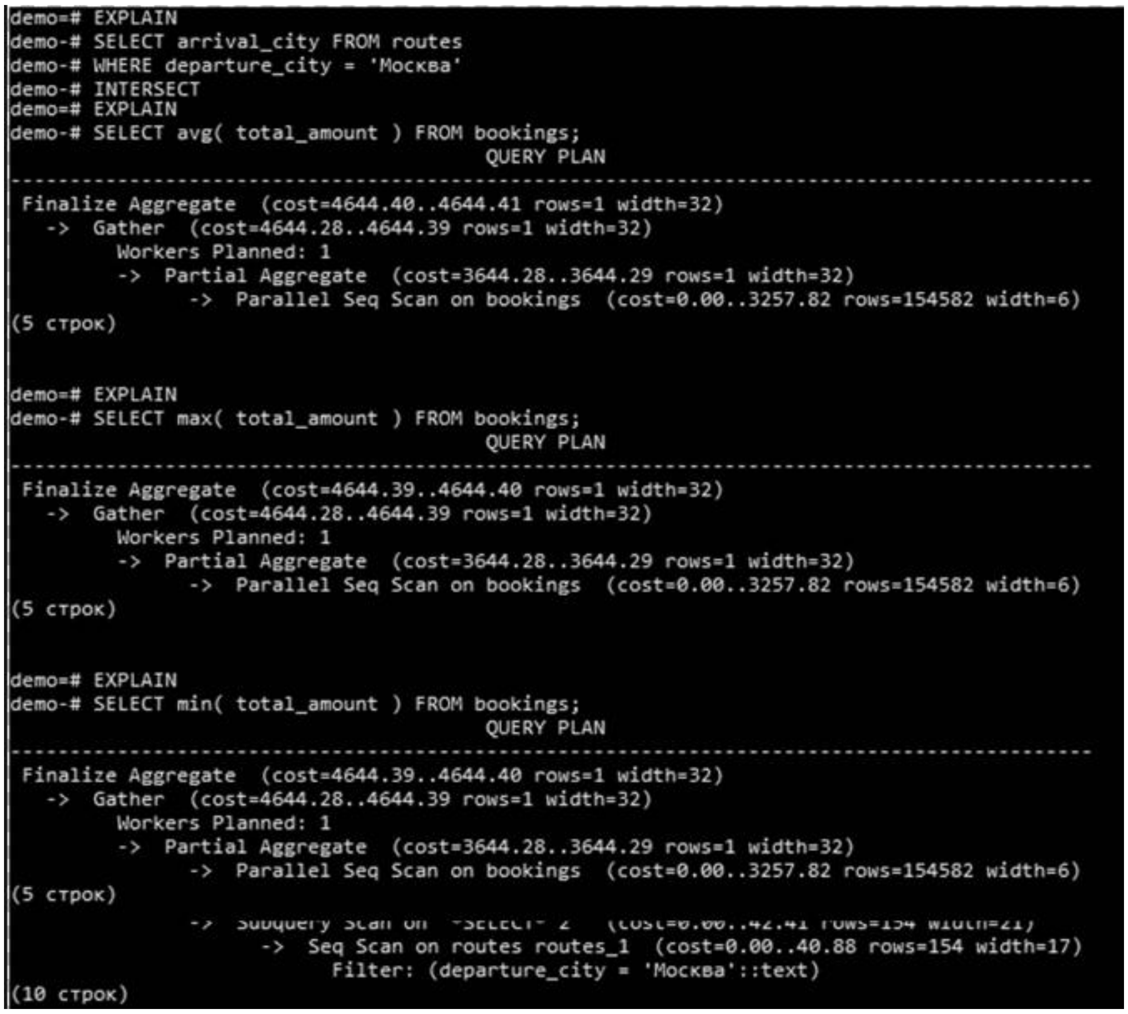
```
demo=# EXPLAIN
demo-# SELECT r.min_sum, r.max_sum, count( b.* )
demo-# FROM bookings b
demo-# RIGHT OUTER JOIN
demo-# ( VALUES ( 0, 100000 ), ( 100000, 200000 ),
demo(# ( 200000, 300000 ), ( 300000, 400000 ),
demo(# ( 400000, 500000 ), ( 500000, 600000 ),
demo(# ( 600000, 700000 ), ( 700000, 800000 ),
demo(# ( 800000, 900000 ), ( 900000, 1000000 ),
demo(# ( 1000000, 1100000 ), ( 1100000, 1200000 ),
demo(# ( 1200000, 1300000 )
demo(# ) AS r ( min_sum, max_sum )
demo-# ON b.total_amount >= r.min_sum AND b.total_amount < r.max_sum
demo-# GROUP BY r.min_sum, r.max_sum
demo-# ORDER BY r.min_sum;
                                                       QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=118081.38..118081.41 rows=13 width=16)
   Sort Key: "*VALUES*".column1
   ->  HashAggregate  (cost=118081.01..118081.14 rows=13 width=16)
         Group Key: "*VALUES*".column1, "*VALUES*".column2
         ->  Nested Loop Left Join  (cost=0.00..115234.11 rows=379586 width=57)
               Join Filter: ((b.total_amount >= ("*VALUES*".column1)::numeric) AND (b.total_amount < ("*VALUES*".column2)::numeric))
               ->  Values Scan on "*VALUES*"  (cost=0.00..0.16 rows=13 width=8)
               ->  Materialize  (cost=0.00..8220.85 rows=262790 width=55)
                     ->  Seq Scan on bookings b  (cost=0.00..4339.90 rows=262790 width=55)
(9 строк)
```

Рисунок 21 – Задание 15



```
demo=# EXPLAIN
demo-# SELECT arrival_city FROM routes
demo-# WHERE departure_city = 'Москва'
demo-# INTERSECT
demo-# SELECT arrival_city FROM routes
demo-# WHERE departure_city = 'Санкт-Петербург'
demo-# ORDER BY arrival_city;
                                      QUERY PLAN
-----------------------------------------------------------------------------------------
 Sort  (cost=85.79..85.87 rows=30 width=36)
   Sort Key: "*SELECT* 2".arrival_city
   ->  HashSetOp Intersect  (cost=0.00..85.06 rows=30 width=36)
         ->  Append  (cost=0.00..84.58 rows=189 width=36)
               ->  Subquery Scan on "*SELECT* 2"  (cost=0.00..41.23 rows=35 width=21)
                     ->  Seq Scan on routes  (cost=0.00..40.88 rows=35 width=17)
                           Filter: (departure_city = 'Санкт-Петербург'::text)
               ->  Subquery Scan on "*SELECT* 1"  (cost=0.00..42.41 rows=154 width=21)
                     ->  Seq Scan on routes routes_1  (cost=0.00..40.88 rows=154 width=17)
                           Filter: (departure_city = 'Москва'::text)
(10 строк)


demo=# EXPLAIN
demo-# SELECT arrival_city FROM routes
demo-# WHERE departure_city = 'Санкт-Петербург'
demo-# EXCEPT
demo-# SELECT arrival_city FROM routes
demo-# WHERE departure_city = 'Москва'
demo-# ORDER BY arrival_city;
                                      QUERY PLAN
-----------------------------------------------------------------------------------------
 Sort  (cost=85.79..85.87 rows=30 width=36)
   Sort Key: "*SELECT* 1".arrival_city
   ->  HashSetOp Except  (cost=0.00..85.06 rows=30 width=36)
         ->  Append  (cost=0.00..84.58 rows=189 width=36)
               ->  Subquery Scan on "*SELECT* 1"  (cost=0.00..41.23 rows=35 width=21)
                     ->  Seq Scan on routes  (cost=0.00..40.88 rows=35 width=17)
                           Filter: (departure_city = 'Санкт-Петербург'::text)
               ->  Subquery Scan on "*SELECT* 2"  (cost=0.00..42.41 rows=154 width=21)
                     ->  Seq Scan on routes routes_1  (cost=0.00..40.88 rows=154 width=17)
                           Filter: (departure_city = 'Москва'::text)
(10 строк)
```

Рисунок 22 – Задание 15



```
demo=# EXPLAIN
demo-# SELECT arrival_city FROM routes
demo-# WHERE departure_city = 'Москва'
demo-# INTERSECT
demo=# EXPLAIN
demo-# SELECT avg( total_amount ) FROM bookings;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=4644.40..4644.41 rows=1 width=32)
   ->  Gather  (cost=4644.28..4644.39 rows=1 width=32)
         Workers Planned: 1
         ->  Partial Aggregate  (cost=3644.28..3644.29 rows=1 width=32)
               ->  Parallel Seq Scan on bookings  (cost=0.00..3257.82 rows=154582 width=6)
(5 строк)


demo=# EXPLAIN
demo-# SELECT max( total_amount ) FROM bookings;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=4644.39..4644.40 rows=1 width=32)
   ->  Gather  (cost=4644.28..4644.39 rows=1 width=32)
         Workers Planned: 1
         ->  Partial Aggregate  (cost=3644.28..3644.29 rows=1 width=32)
               ->  Parallel Seq Scan on bookings  (cost=0.00..3257.82 rows=154582 width=6)
(5 строк)


demo=# EXPLAIN
demo-# SELECT min( total_amount ) FROM bookings;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=4644.39..4644.40 rows=1 width=32)
   ->  Gather  (cost=4644.28..4644.39 rows=1 width=32)
         Workers Planned: 1
         ->  Partial Aggregate  (cost=3644.28..3644.29 rows=1 width=32)
               ->  Parallel Seq Scan on bookings  (cost=0.00..3257.82 rows=154582 width=6)
(5 строк)
               ->  Subquery Scan on "*SELECT* 2"  (cost=0.00..42.41 rows=154 width=21)
                     ->  Seq Scan on routes routes_1  (cost=0.00..40.88 rows=154 width=17)
                           Filter: (departure_city = 'Москва'::text)
(10 строк)
```

Рисунок 23 – Задание 15

3 Заключение

В ходе практической работы были изучены основы повышения производительности, были выполнены указанные в файле задания.