ДЗ 9. Глава 10 (упражнения 3, 6, 8).

3. Самостоятельно выполните команду EXPLAIN для запроса, содержащего общее табличное выражение (CTE). Посмотрите, на каком уровне находится узел плана, отвечающий за это выражение, как он оформляется. Учтите, что общие табличные выражения всегда материализуются, т. е. вычисляются однократно и результат их вычисления сохраняется в памяти, а затем все последующие обращения в рамках запроса направляются уже к этому материализованному результату.

```
demo=# EXPLAIN WITH add_row AS
( INSERT INTO aircrafts_tmp
SELECT * FROM aircrafts
RETURNING *
)
INSERT INTO aircrafts_log
SELECT add_row.aircraft_code, add_row.model, add_row.range,
current_timestamp, 'INSERT'
FROM add_row;
                              QUERY PLAN
-----------------------------------------------------------------------
 Insert on aircrafts_log  (cost=1.09..1.31 rows=0 width=0)
   CTE add_row
     ->  Insert on aircrafts_tmp  (cost=0.00..1.09 rows=9 width=52)
           ->  Seq Scan on aircrafts  (cost=0.00..1.09 rows=9 width=52)
   ->  CTE Scan on add_row  (cost=0.00..0.22 rows=9 width=92)
(5 rows)

demo=#
```

6. Выполните команду EXPLAIN для запроса, в котором использована какая-нибудь из оконных функций. Найдите в плане выполнения запроса узел с именем WindowAgg. Попробуйте объяснить, почему он занимает именно этот уровень в плане.

```
demo=# SELECT airport_name,
city,
round( latitude::numeric, 2 ) AS ltd,
timezone,
rank() OVER (
PARTITION BY timezone
ORDER BY latitude DESC
)
FROM airports
WHERE timezone IN ( 'Asia/Irkutsk', 'Asia/Krasnoyarsk' )
ORDER BY timezone, rank;
 airport_name  |     city      |  ltd  |     timezone     | rank
---------------+---------------+-------+------------------+------
 Усть-Илимск   | Усть-Илимск   | 58.14 | Asia/Irkutsk     |    1
 Усть-Кут      | Усть-Кут      | 56.85 | Asia/Irkutsk     |    2
 Братск        | Братск        | 56.37 | Asia/Irkutsk     |    3
 Иркутск       | Иркутск       | 52.27 | Asia/Irkutsk     |    4
 Байкал        | Улан-Удэ      | 51.81 | Asia/Irkutsk     |    5
 Норильск      | Норильск      | 69.31 | Asia/Krasnoyarsk |    1
 Стрежевой     | Стрежевой     | 60.72 | Asia/Krasnoyarsk |    2
 Богашёво      | Томск         | 56.38 | Asia/Krasnoyarsk |    3
 Емельяново    | Красноярск    | 56.18 | Asia/Krasnoyarsk |    4
 Абакан        | Абакан        | 53.74 | Asia/Krasnoyarsk |    5
 Барнаул       | Барнаул       | 53.36 | Asia/Krasnoyarsk |    6
 Горно-Алтайск | Горно-Алтайск | 51.97 | Asia/Krasnoyarsk |    7
 Кызыл         | Кызыл         | 51.67 | Asia/Krasnoyarsk |    8
(13 rows)
```

```
demo=# EXPLAIN SELECT airport_name,
city,
round( latitude::numeric, 2 ) AS ltd,
timezone,
rank() OVER (
PARTITION BY timezone
ORDER BY latitude DESC
)
FROM airports
WHERE timezone IN ( 'Asia/Irkutsk', 'Asia/Krasnoyarsk' )
ORDER BY timezone, rank;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Sort  (cost=4.11..4.14 rows=13 width=97)
   Sort Key: timezone, (rank() OVER (?))
   ->  WindowAgg  (cost=3.54..3.87 rows=13 width=97)
         ->  Sort  (cost=3.54..3.57 rows=13 width=57)
               Sort Key: timezone, latitude DESC
               ->  Seq Scan on airports  (cost=0.00..3.30 rows=13 width=57)
                     Filter: (timezone = ANY ('{Asia/Irkutsk,Asia/Krasnoyarsk}'::text[]))
(7 rows)

demo=#
```

Необходимо сначала произвести сортировку по временным зонам, а затем по ним сделать оконную функцию.

8.* Замена коррелированного подзапроса соединением таблиц является одним из способов повышения производительности.

Предположим, что мы задались вопросом: сколько маршрутов обслуживают самолеты каждого типа? При этом нужно учитывать, что может иметь место такая ситуация, когда самолеты какого-либо типа не обслуживают ни одного маршрута. Поэтому необходимо использовать не только представление «Маршруты» (`routes`), но и таблицу «Самолеты» (`aircrafts`).

Это первый вариант запроса, в нем используется коррелированный подзапрос.

```
EXPLAIN ANALYZE
  SELECT a.aircraft_code AS a_code,
         a.model,
         ( SELECT count( r.aircraft_code )
             FROM routes r
             WHERE r.aircraft_code = a.aircraft_code
         ) AS num_routes
    FROM aircrafts a
    GROUP BY 1, 2
    ORDER BY 3 DESC;
```

А в этом варианте коррелированный подзапрос раскрыт и заменен внешним соединением:

```
EXPLAIN ANALYZE
  SELECT a.aircraft_code AS a_code,
         a.model,
         count( r.aircraft_code ) AS num_routes
    FROM aircrafts a
    LEFT OUTER JOIN routes r
      ON r.aircraft_code = a.aircraft_code
    GROUP BY 1, 2
    ORDER BY 3 DESC;
```

Причина использования внешнего соединения в том, что может найтись модель самолета, не обслуживающая ни одного маршрута, и если не использовать внешнее соединение, она вообще не попадет в результирующую выборку.

Исследуйте планы выполнения обоих запросов. Попытайтесь найти объяснение различиям в эффективности их выполнения. Чтобы получить усредненную картину, выполните каждый запрос несколько раз. Поскольку таблицы, участвующие в запросах, небольшие, то различие по абсолютным затратам времени выполнения будет незначительным. Но если бы число строк в таблицах было большим, то экономия ресурсов сервера могла оказаться заметной.

Предложите аналогичную пару запросов к базе данных «Авиаперевозки». Проведите необходимые эксперименты с вашими запросами.

Запросы из упражнения:

```
demo=# EXPLAIN ANALYZE
SELECT a.aircraft_code AS a_code,
a.model,
( SELECT count( r.aircraft_code )
FROM routes r
WHERE r.aircraft_code = a.aircraft_code
) AS num_routes
FROM aircrafts a
GROUP BY 1, 2
ORDER BY 3 DESC;
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=236.31..236.34 rows=9 width=56) (actual time=0.591..0.592 rows=9 loops=1)
   Sort Key: ((SubPlan 1)) DESC
   Sort Method: quicksort  Memory: 25kB
   ->  HashAggregate  (cost=1.11..236.17 rows=9 width=56) (actual time=0.091..0.587 rows=9 loops=1)
         Group Key: a.aircraft_code
         Batches: 1  Memory Usage: 24kB
         ->  Seq Scan on aircrafts a  (cost=0.00..1.09 rows=9 width=48) (actual time=0.002..0.003 rows=9 loops=1)
         SubPlan 1
           ->  Aggregate  (cost=26.10..26.11 rows=1 width=8) (actual time=0.064..0.064 rows=1 loops=9)
                 ->  Seq Scan on routes r  (cost=0.00..25.88 rows=89 width=4) (actual time=0.010..0.058 rows=79 loops=9)
                       Filter: (aircraft_code = a.aircraft_code)
                       Rows Removed by Filter: 631
 Planning Time: 0.070 ms
 Execution Time: 0.610 ms
(14 rows)
```

```
demo=# EXPLAIN ANALYZE
SELECT a.aircraft_code AS a_code,
a.model,
count( r.aircraft_code ) AS num_routes
FROM aircrafts a
LEFT OUTER JOIN routes r
ON r.aircraft_code = a.aircraft_code
GROUP BY 1, 2
ORDER BY 3 DESC;
                                                      QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=31.83..31.85 rows=9 width=56) (actual time=0.273..0.275 rows=9 loops=1)
   Sort Key: (count(r.aircraft_code)) DESC
   Sort Method: quicksort  Memory: 25kB
   ->  HashAggregate  (cost=31.60..31.69 rows=9 width=56) (actual time=0.255..0.257 rows=9 loops=1)
         Group Key: a.aircraft_code
         Batches: 1  Memory Usage: 24kB
         ->  Hash Right Join  (cost=1.20..28.05 rows=710 width=52) (actual time=0.012..0.177 rows=711 loops=1)
               Hash Cond: (r.aircraft_code = a.aircraft_code)
               ->  Seq Scan on routes r  (cost=0.00..24.10 rows=710 width=4) (actual time=0.002..0.045 rows=710 loops=1)
               ->  Hash  (cost=1.09..1.09 rows=9 width=48) (actual time=0.007..0.008 rows=9 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                     ->  Seq Scan on aircrafts a  (cost=0.00..1.09 rows=9 width=48) (actual time=0.004..0.005 rows=9 loops=1)
 Planning Time: 0.156 ms
 Execution Time: 0.294 ms
(14 rows)
```

Коррелированный подзапрос, предположительно, выполняется быстрее из-за того, что нет несколько подсчетов как у второго подзапроса с JOIN.

Два запроса:

```
demo=# SELECT a.city AS city,
a.timezone,
(SELECT count(r.arrival_city) FROM routes r WHERE r.arrival_city = a.city) AS count
FROM airports a
WHERE a.timezone = 'Europe/Samara' GROUP BY 1, 2;
   city    |   timezone    | count
-----------+---------------+-------
 Астрахань | Europe/Samara |     4
 Ижевск    | Europe/Samara |     1
 Самара    | Europe/Samara |     3
 Ульяновск | Europe/Samara |    11
(4 rows)

demo=# SELECT a.city AS city,
a.timezone, count (r.arrival_city) as ar
FROM airports a
LEFT OUTER JOIN routes r
ON r.arrival_city = a.city
GROUP BY 1, 2 HAVING a.timezone='Europe/Samara';
   city    |   timezone    | ar
-----------+---------------+----
 Астрахань | Europe/Samara |  4
 Ульяновск | Europe/Samara | 22
 Самара    | Europe/Samara |  3
 Ижевск    | Europe/Samara |  1
(4 rows)
```

Анализ:

```
demo=# EXPLAIN ANALYZE SELECT a.city AS city,
a.timezone,
(SELECT count(r.arrival_city) FROM routes r WHERE r.arrival_city = a.city) AS count
FROM airports a
WHERE a.timezone = 'Europe/Samara' GROUP BY 1, 2;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Group  (cost=3.36..132.91 rows=5 width=40) (actual time=0.082..0.272 rows=4 loops=1)
   Group Key: a.city, a.timezone
   ->  Sort  (cost=3.36..3.37 rows=5 width=32) (actual time=0.017..0.018 rows=5 loops=1)
         Sort Key: a.city
         Sort Method: quicksort  Memory: 25kB
         ->  Seq Scan on airports a  (cost=0.00..3.30 rows=5 width=32) (actual time=0.006..0.013 rows=5 loops=1)
               Filter: (timezone = 'Europe/Samara'::text)
               Rows Removed by Filter: 99
   SubPlan 1
     ->  Aggregate  (cost=25.89..25.90 rows=1 width=8) (actual time=0.062..0.062 rows=1 loops=4)
           ->  Seq Scan on routes r  (cost=0.00..25.88 rows=7 width=17) (actual time=0.013..0.061 rows=5 loops=4)
                 Filter: (arrival_city = a.city)
                 Rows Removed by Filter: 705
 Planning Time: 0.069 ms
 Execution Time: 0.288 ms
(15 rows)
```

```
demo=# EXPLAIN ANALYZE SELECT a.city AS city,
a.timezone, count (r.arrival_city) as ar
FROM airports a
LEFT OUTER JOIN routes r
ON r.arrival_city = a.city
GROUP BY 1, 2 HAVING a.timezone='Europe/Samara';
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=32.76..32.81 rows=5 width=40) (actual time=0.156..0.157 rows=4 loops=1)
   Group Key: a.city, a.timezone
   Batches: 1  Memory Usage: 24kB
   ->  Hash Right Join  (cost=3.36..32.39 rows=49 width=49) (actual time=0.036..0.149 rows=30 loops=1)
         Hash Cond: (r.arrival_city = a.city)
         ->  Seq Scan on routes r  (cost=0.00..24.10 rows=710 width=17) (actual time=0.002..0.047 rows=710 loops=1)
         ->  Hash  (cost=3.30..3.30 rows=5 width=32) (actual time=0.029..0.029 rows=5 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 9kB
               ->  Seq Scan on airports a  (cost=0.00..3.30 rows=5 width=32) (actual time=0.006..0.027 rows=5 loops=1)
                     Filter: (timezone = 'Europe/Samara'::text)
                     Rows Removed by Filter: 99
 Planning Time: 0.194 ms
 Execution Time: 0.197 ms
(13 rows)
```

Тут тоже коррелированный подзапрос, предположительно, выполняется быстрее из-за того, что нет несколько подсчетов как у второго подзапроса с JOIN.