KOREA UNIVERSITY
DATAX LAB

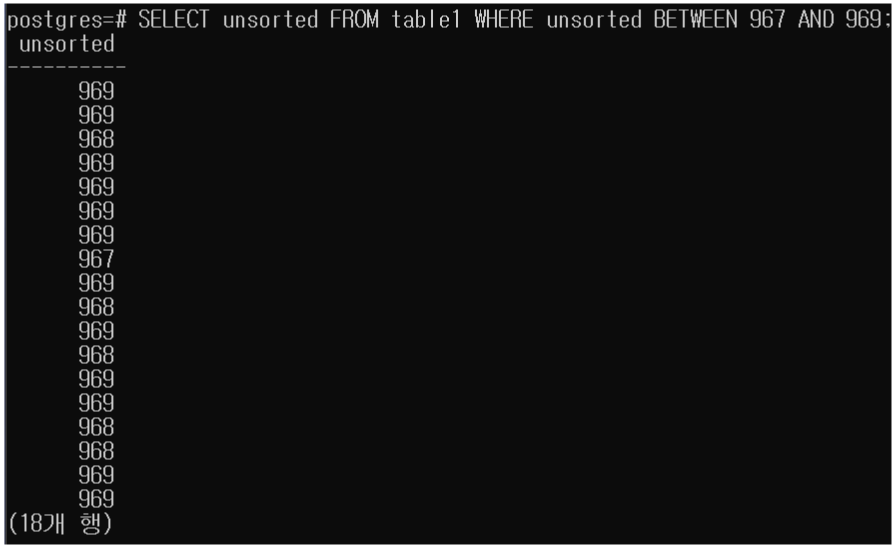
Chapter 16 – Lab Solution

Query Optimization 1

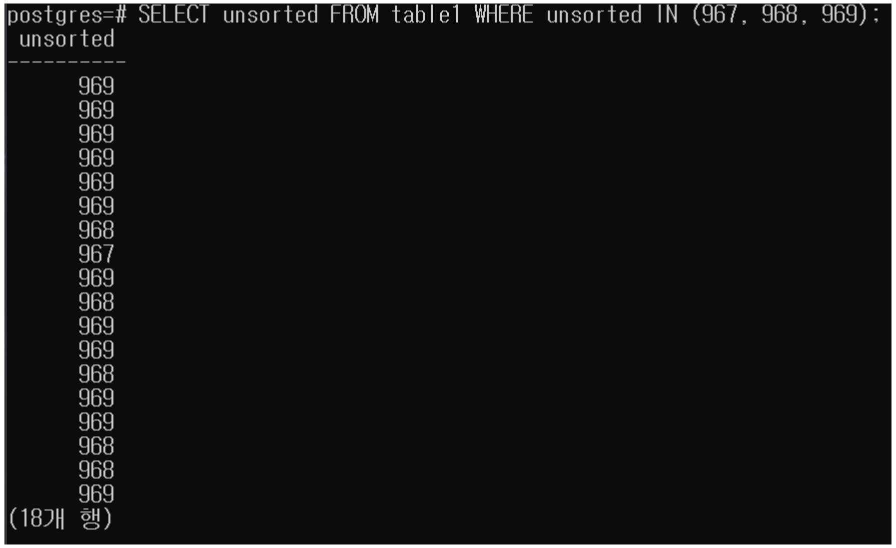
Exercise 1 Answer

a. SELECT unsorted FROM table1 WHERE unsorted BETWEEN 967 AND 969;

b. SELECT unsorted FROM table1 WHERE unsorted IN (967, 968, 969);



```
postgres=# SELECT unsorted FROM table1 WHERE unsorted BETWEEN 967 AND 969;
 unsorted
----------
      969
      969
      968
      969
      969
      969
      969
      967
      969
      968
      969
      968
      969
      969
      968
      968
      969
      969
(18개 행)
```



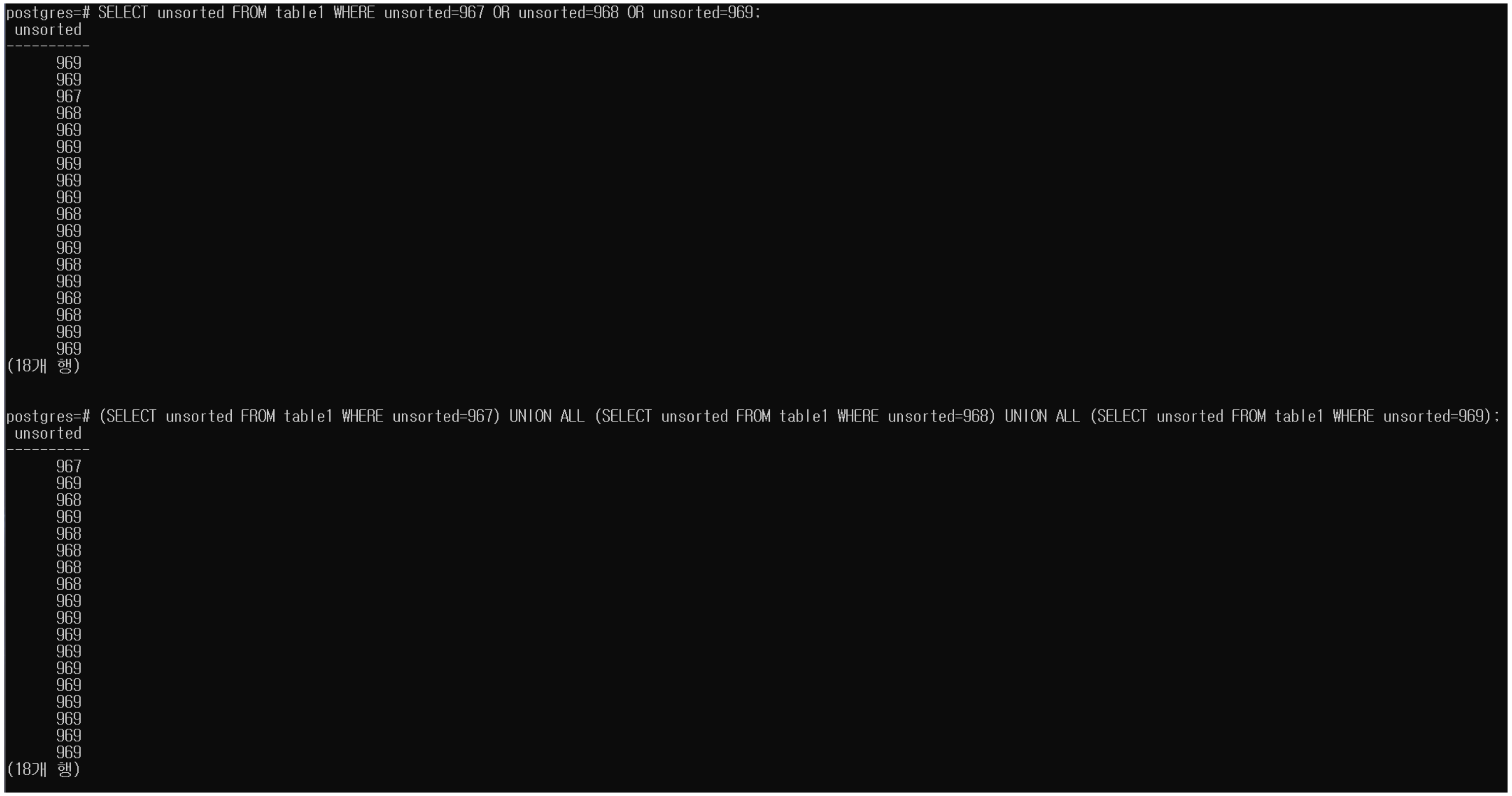
```
postgres=# SELECT unsorted FROM table1 WHERE unsorted IN (967, 968, 969);
 unsorted
----------
      969
      969
      969
      969
      969
      969
      968
      967
      969
      968
      969
      969
      968
      969
      969
      968
      968
      969
(18개 행)
```

KOREA UNIVERSITY
DATAX LAB

Exercise 1 Answer

c. SELECT unsorted FROM table1 WHERE unsorted=967 OR unsorted=968 OR unsorted=969;

d. (SELECT unsorted FROM table1 WHERE unsorted=967) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=968) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=969);



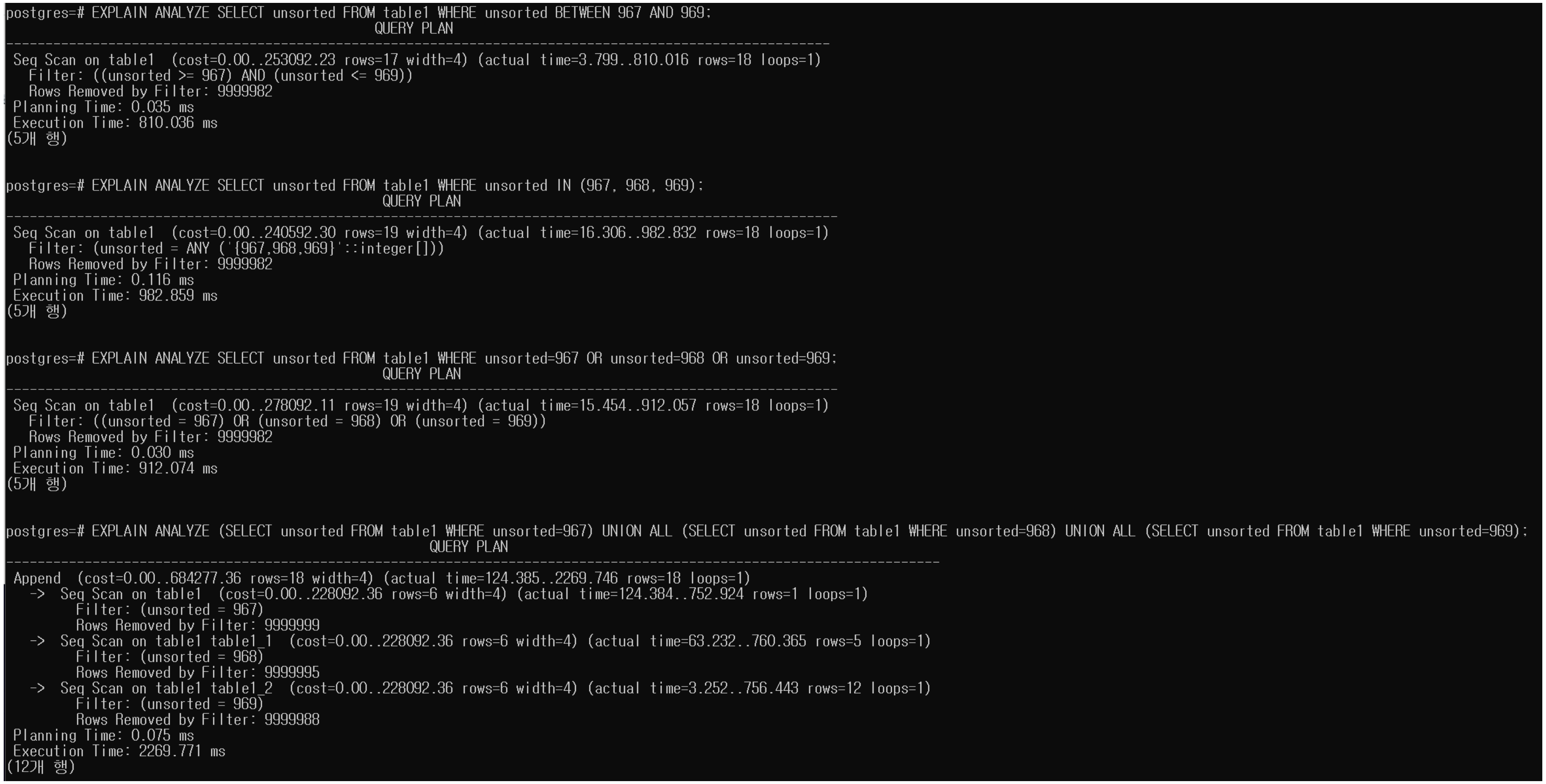
```
postgres=# SELECT unsorted FROM table1 WHERE unsorted=967 OR unsorted=968 OR unsorted=969;
 unsorted
----------
      969
      969
      967
      968
      969
      969
      969
      969
      969
      968
      969
      969
      968
      969
      968
      968
      969
      969
(18개 행)


postgres=# (SELECT unsorted FROM table1 WHERE unsorted=967) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=968) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=969);
 unsorted
----------
      967
      969
      968
      969
      968
      968
      968
      968
      969
      969
      969
      969
      969
      969
      969
      969
      969
      969
(18개 행)
```

Exercise 2.a Answer

[No index]

a. EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted BETWEEN 967 AND 969;
b. EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted IN (967, 968, 969);
c. EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted=967 OR unsorted=968 OR unsorted=969;
d. EXPLAIN ANALYZE (SELECT unsorted FROM table1 WHERE unsorted=967) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=968) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=969);

Exercise 2.a Answer



```
postgres=# EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted BETWEEN 967 AND 969;
                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Seq Scan on table1  (cost=0.00..253092.23 rows=17 width=4) (actual time=3.799..810.016 rows=18 loops=1)
   Filter: ((unsorted >= 967) AND (unsorted <= 969))
   Rows Removed by Filter: 9999982
 Planning Time: 0.035 ms
 Execution Time: 810.036 ms
(5개 행)


postgres=# EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted IN (967, 968, 969);
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on table1  (cost=0.00..240592.30 rows=19 width=4) (actual time=16.306..982.832 rows=18 loops=1)
   Filter: (unsorted = ANY ('{967,968,969}'::integer[]))
   Rows Removed by Filter: 9999982
 Planning Time: 0.116 ms
 Execution Time: 982.859 ms
(5개 행)


postgres=# EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted=967 OR unsorted=968 OR unsorted=969;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on table1  (cost=0.00..278092.11 rows=19 width=4) (actual time=15.454..912.057 rows=18 loops=1)
   Filter: ((unsorted = 967) OR (unsorted = 968) OR (unsorted = 969))
   Rows Removed by Filter: 9999982
 Planning Time: 0.030 ms
 Execution Time: 912.074 ms
(5개 행)


postgres=# EXPLAIN ANALYZE (SELECT unsorted FROM table1 WHERE unsorted=967) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=968) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=969);
                                                       QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------
 Append  (cost=0.00..684277.36 rows=18 width=4) (actual time=124.385..2269.746 rows=18 loops=1)
   ->  Seq Scan on table1  (cost=0.00..228092.36 rows=6 width=4) (actual time=124.384..752.924 rows=1 loops=1)
         Filter: (unsorted = 967)
         Rows Removed by Filter: 9999999
   ->  Seq Scan on table1 table1_1  (cost=0.00..228092.36 rows=6 width=4) (actual time=63.232..760.365 rows=5 loops=1)
         Filter: (unsorted = 968)
         Rows Removed by Filter: 9999995
   ->  Seq Scan on table1 table1_2  (cost=0.00..228092.36 rows=6 width=4) (actual time=3.252..756.443 rows=12 loops=1)
         Filter: (unsorted = 969)
         Rows Removed by Filter: 9999988
 Planning Time: 0.075 ms
 Execution Time: 2269.771 ms
(12개 행)
```

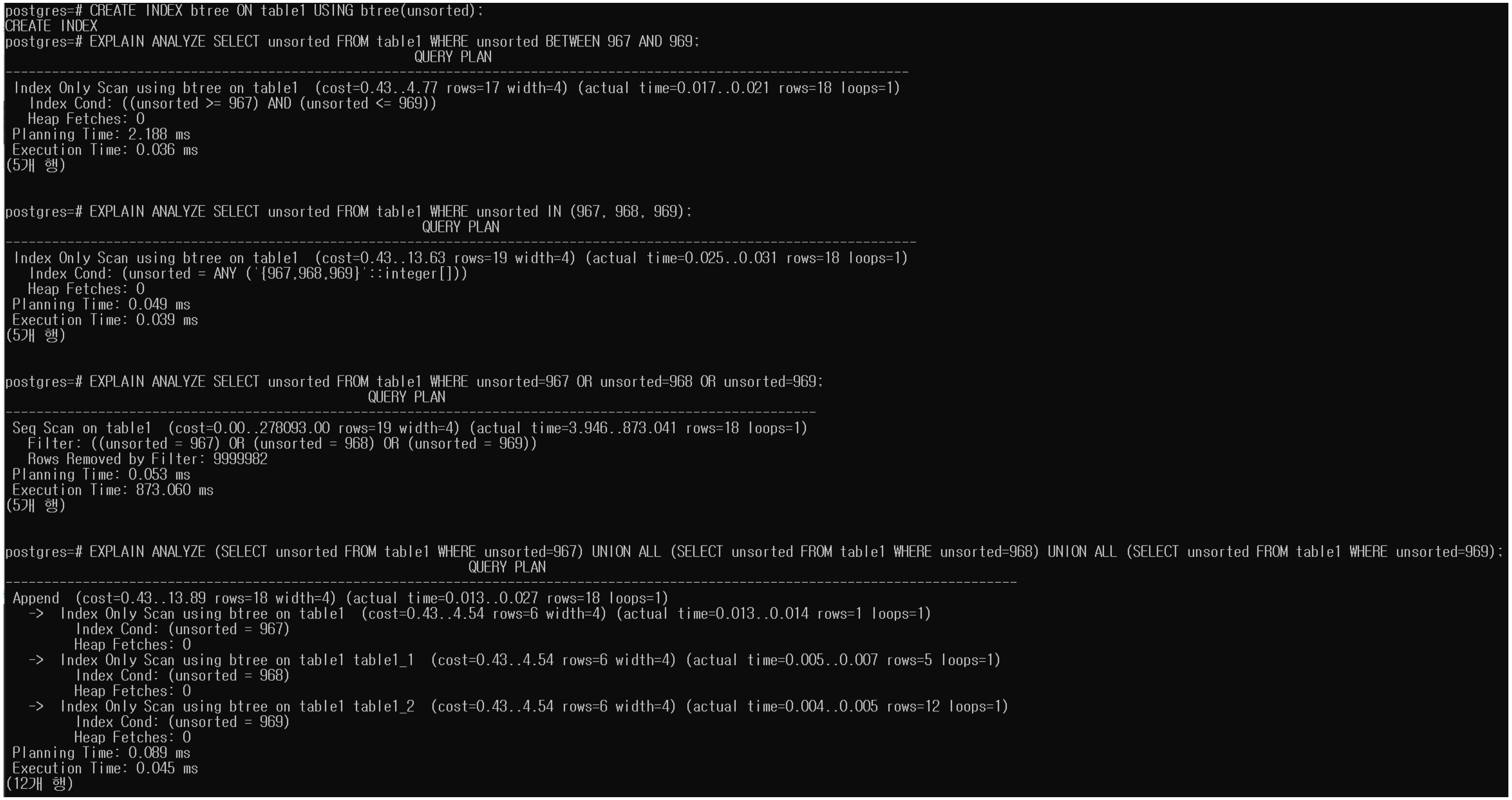
Exercise 2.b Answer

[B-tree index]

CREATE INDEX btree ON table1 USING btree(unsorted);

a. EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted BETWEEN 967 AND 969;
b. EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted IN (967, 968, 969);
c. EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted=967 OR unsorted=968 OR unsorted=969;
d. EXPLAIN ANALYZE (SELECT unsorted FROM table1 WHERE unsorted=967) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=968) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=969);

Exercise 2.b Answer



```
postgres=# CREATE INDEX btree ON table1 USING btree(unsorted);
CREATE INDEX
postgres=# EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted BETWEEN 967 AND 969;
                                                       QUERY PLAN
---------------------------------------------------------------------------------------------------------------------
 Index Only Scan using btree on table1  (cost=0.43..4.77 rows=17 width=4) (actual time=0.017..0.021 rows=18 loops=1)
   Index Cond: ((unsorted >= 967) AND (unsorted <= 969))
   Heap Fetches: 0
 Planning Time: 2.188 ms
 Execution Time: 0.036 ms
(5개 행)


postgres=# EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted IN (967, 968, 969);
                                                        QUERY PLAN
----------------------------------------------------------------------------------------------------------------------
 Index Only Scan using btree on table1  (cost=0.43..13.63 rows=19 width=4) (actual time=0.025..0.031 rows=18 loops=1)
   Index Cond: (unsorted = ANY ('{967,968,969}'::integer[]))
   Heap Fetches: 0
 Planning Time: 0.049 ms
 Execution Time: 0.039 ms
(5개 행)


postgres=# EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted=967 OR unsorted=968 OR unsorted=969;
                                                  QUERY PLAN
---------------------------------------------------------------------------------------------------------
 Seq Scan on table1  (cost=0.00..278093.00 rows=19 width=4) (actual time=3.946..873.041 rows=18 loops=1)
   Filter: ((unsorted = 967) OR (unsorted = 968) OR (unsorted = 969))
   Rows Removed by Filter: 9999982
 Planning Time: 0.053 ms
 Execution Time: 873.060 ms
(5개 행)


postgres=# EXPLAIN ANALYZE (SELECT unsorted FROM table1 WHERE unsorted=967) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=968) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=969);
                                                              QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------
 Append  (cost=0.43..13.89 rows=18 width=4) (actual time=0.013..0.027 rows=18 loops=1)
   ->  Index Only Scan using btree on table1  (cost=0.43..4.54 rows=6 width=4) (actual time=0.013..0.014 rows=1 loops=1)
         Index Cond: (unsorted = 967)
         Heap Fetches: 0
   ->  Index Only Scan using btree on table1 table1_1  (cost=0.43..4.54 rows=6 width=4) (actual time=0.005..0.007 rows=5 loops=1)
         Index Cond: (unsorted = 968)
         Heap Fetches: 0
   ->  Index Only Scan using btree on table1 table1_2  (cost=0.43..4.54 rows=6 width=4) (actual time=0.004..0.005 rows=12 loops=1)
         Index Cond: (unsorted = 969)
         Heap Fetches: 0
 Planning Time: 0.089 ms
 Execution Time: 0.045 ms
(12개 행)
```

Exercise 2.c Answer

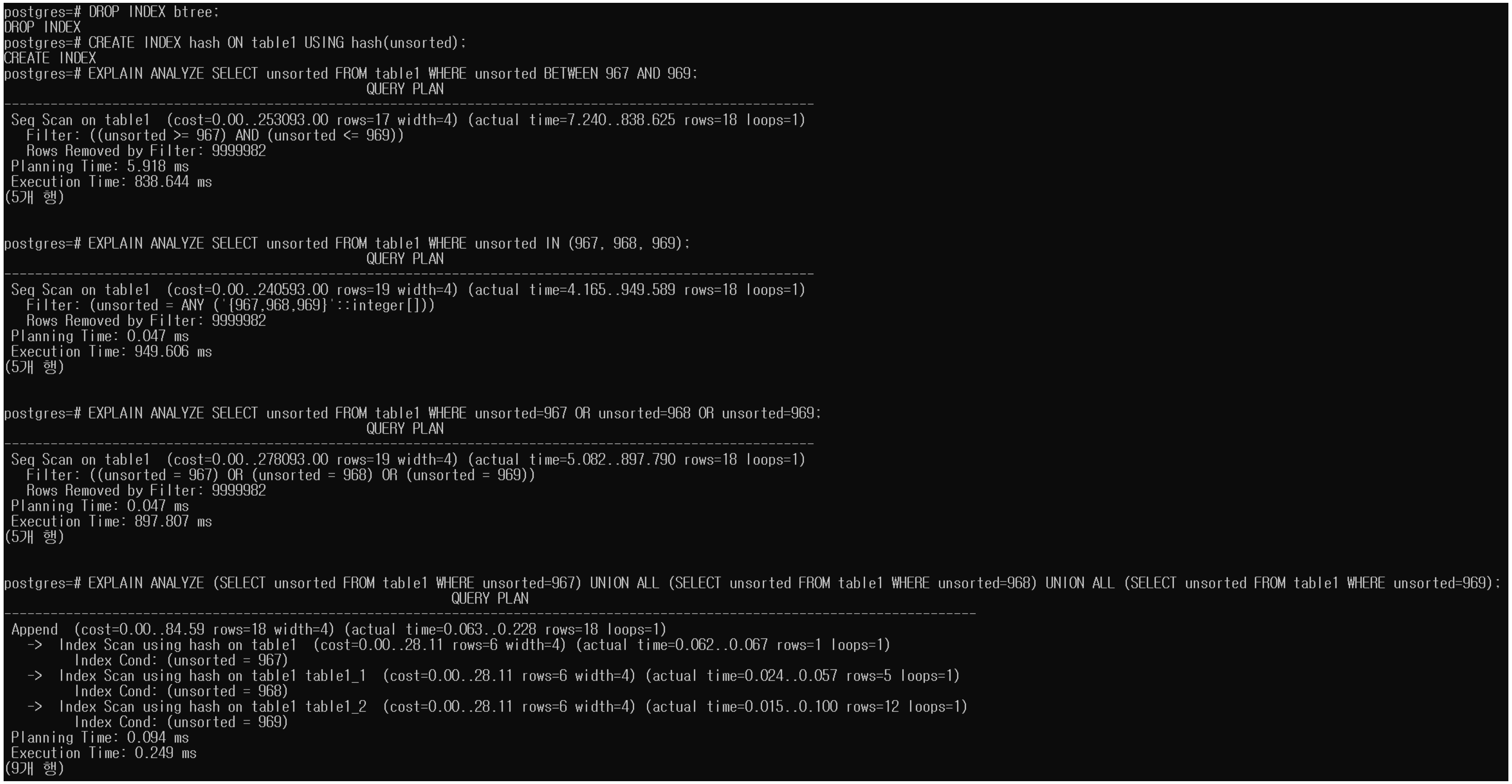
[Hash index]

DROP INDEX btree;

CREATE INDEX hash ON table1 USING hash(unsorted);

a. EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted BETWEEN 967 AND 969;
b. EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted IN (967, 968, 969);
c. EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted=967 OR unsorted=968 OR unsorted=969;
d. EXPLAIN ANALYZE (SELECT unsorted FROM table1 WHERE unsorted=967) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=968) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=969);

Exercise 2.c Answer



```
postgres=# DROP INDEX btree;
DROP INDEX
postgres=# CREATE INDEX hash ON table1 USING hash(unsorted);
CREATE INDEX
postgres=# EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted BETWEEN 967 AND 969;
                                               QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Seq Scan on table1  (cost=0.00..253093.00 rows=17 width=4) (actual time=7.240..838.625 rows=18 loops=1)
   Filter: ((unsorted >= 967) AND (unsorted <= 969))
   Rows Removed by Filter: 9999982
 Planning Time: 5.918 ms
 Execution Time: 838.644 ms
(5개 행)


postgres=# EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted IN (967, 968, 969);
                                               QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Seq Scan on table1  (cost=0.00..240593.00 rows=19 width=4) (actual time=4.165..949.589 rows=18 loops=1)
   Filter: (unsorted = ANY ('{967,968,969}'::integer[]))
   Rows Removed by Filter: 9999982
 Planning Time: 0.047 ms
 Execution Time: 949.606 ms
(5개 행)


postgres=# EXPLAIN ANALYZE SELECT unsorted FROM table1 WHERE unsorted=967 OR unsorted=968 OR unsorted=969;
                                               QUERY PLAN
----------------------------------------------------------------------------------------------------------
 Seq Scan on table1  (cost=0.00..278093.00 rows=19 width=4) (actual time=5.082..897.790 rows=18 loops=1)
   Filter: ((unsorted = 967) OR (unsorted = 968) OR (unsorted = 969))
   Rows Removed by Filter: 9999982
 Planning Time: 0.047 ms
 Execution Time: 897.807 ms
(5개 행)


postgres=# EXPLAIN ANALYZE (SELECT unsorted FROM table1 WHERE unsorted=967) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=968) UNION ALL (SELECT unsorted FROM table1 WHERE unsorted=969);
                                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Append  (cost=0.00..84.59 rows=18 width=4) (actual time=0.063..0.228 rows=18 loops=1)
   ->  Index Scan using hash on table1  (cost=0.00..28.11 rows=6 width=4) (actual time=0.062..0.067 rows=1 loops=1)
         Index Cond: (unsorted = 967)
   ->  Index Scan using hash on table1 table1_1  (cost=0.00..28.11 rows=6 width=4) (actual time=0.024..0.057 rows=5 loops=1)
         Index Cond: (unsorted = 968)
   ->  Index Scan using hash on table1 table1_2  (cost=0.00..28.11 rows=6 width=4) (actual time=0.015..0.100 rows=12 loops=1)
         Index Cond: (unsorted = 969)
 Planning Time: 0.094 ms
 Execution Time: 0.249 ms
(9개 행)
```

Exercise 2 Answer

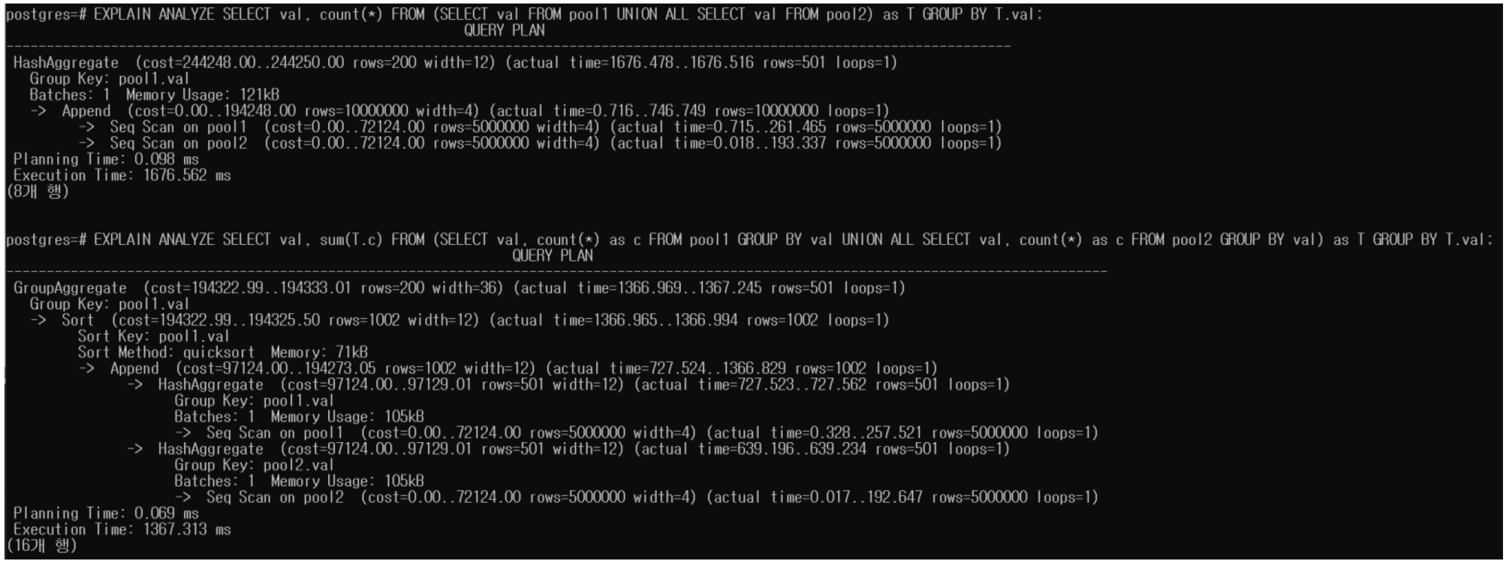
Method / Operator	No index	B-tree	Hash
a. BETWEEN	810.036 ms	0.036 ms	838.644 ms
b. IN	982.859 ms	0.039 ms	949.606 ms
c. OR	912.074 ms	873.060 ms	897.807 ms
d. UNION ALL	2269.771 ms	0.045 ms	0.249 ms

a. BETWEEN operator works well on the B-tree index. B-tree is good for range query.
b. IN operator works well on the B-tree index, but not on the Hash index.
c. OR operator does not optimize by indexes.
d. UNION ALL operator works well on both the B-tree index and the Hash index.

Exercise 3 Answer

a. EXPLAIN ANALYZE SELECT val, count(*) FROM (SELECT val FROM pool1 UNION ALL SELECT val FROM pool2) as T GROUP BY T.val;

b. EXPLAIN ANALYZE SELECT val, sum(T.c) FROM (SELECT val, count(*) as c FROM pool1 GROUP BY val UNION ALL SELECT val, count(*) as c FROM pool2 GROUP BY val) as T GROUP BY T.val;



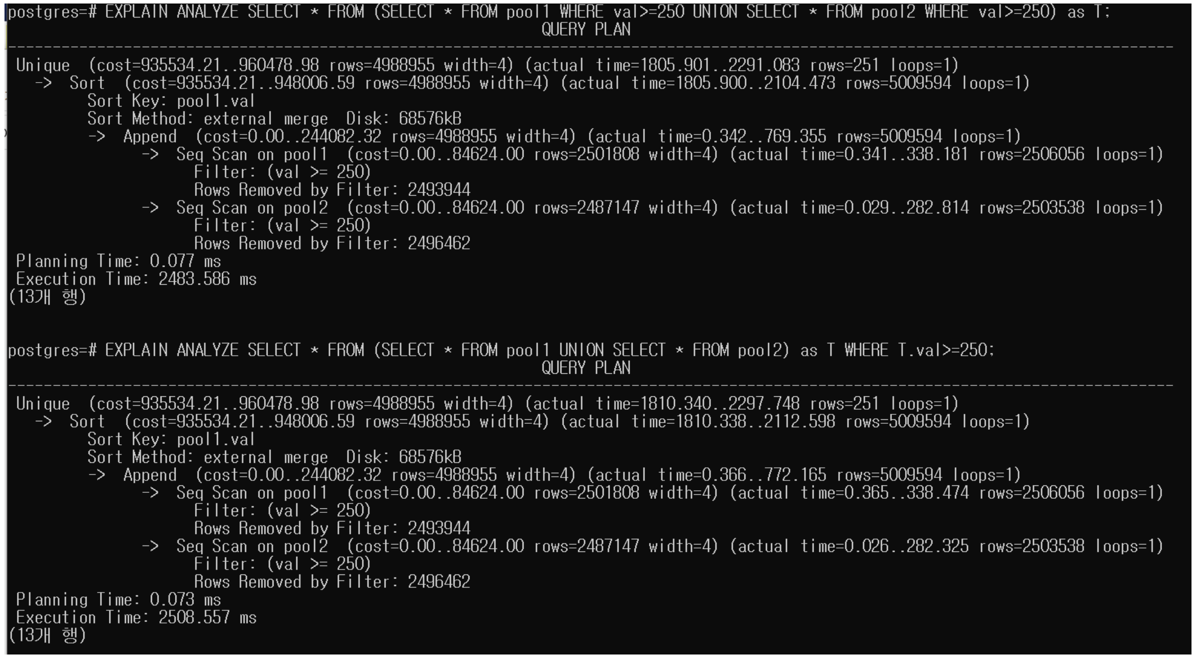
```
postgres=# EXPLAIN ANALYZE SELECT val, count(*) FROM (SELECT val FROM pool1 UNION ALL SELECT val FROM pool2) as T GROUP BY T.val;
                                                               QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=244248.00..244250.00 rows=200 width=12) (actual time=1676.478..1676.516 rows=501 loops=1)
   Group Key: pool1.val
   Batches: 1  Memory Usage: 121kB
   ->  Append  (cost=0.00..194248.00 rows=10000000 width=4) (actual time=0.716..746.749 rows=10000000 loops=1)
         ->  Seq Scan on pool1  (cost=0.00..72124.00 rows=5000000 width=4) (actual time=0.715..261.465 rows=5000000 loops=1)
         ->  Seq Scan on pool2  (cost=0.00..72124.00 rows=5000000 width=4) (actual time=0.018..193.337 rows=5000000 loops=1)
 Planning Time: 0.098 ms
 Execution Time: 1676.562 ms
(8개 행)


postgres=# EXPLAIN ANALYZE SELECT val, sum(T.c) FROM (SELECT val, count(*) as c FROM pool1 GROUP BY val UNION ALL SELECT val, count(*) as c FROM pool2 GROUP BY val) as T GROUP BY T.val;
                                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------
 GroupAggregate  (cost=194322.99..194333.01 rows=200 width=36) (actual time=1366.969..1367.245 rows=501 loops=1)
   Group Key: pool1.val
   ->  Sort  (cost=194322.99..194325.50 rows=1002 width=12) (actual time=1366.965..1366.994 rows=1002 loops=1)
         Sort Key: pool1.val
         Sort Method: quicksort  Memory: 71kB
         ->  Append  (cost=97124.00..194273.05 rows=1002 width=12) (actual time=727.524..1366.829 rows=1002 loops=1)
               ->  HashAggregate  (cost=97124.00..97129.01 rows=501 width=12) (actual time=727.523..727.562 rows=501 loops=1)
                     Group Key: pool1.val
                     Batches: 1  Memory Usage: 105kB
                     ->  Seq Scan on pool1  (cost=0.00..72124.00 rows=5000000 width=4) (actual time=0.328..257.521 rows=5000000 loops=1)
               ->  HashAggregate  (cost=97124.00..97129.01 rows=501 width=12) (actual time=639.196..639.234 rows=501 loops=1)
                     Group Key: pool2.val
                     Batches: 1  Memory Usage: 105kB
                     ->  Seq Scan on pool2  (cost=0.00..72124.00 rows=5000000 width=4) (actual time=0.017..192.647 rows=5000000 loops=1)
 Planning Time: 0.069 ms
 Execution Time: 1367.313 ms
(16개 행)
```

- The actual plan is quite different and execution time is slightly different
  - User level optimization is somewhat important

Exercise 4 Answer

a. EXPLAIN ANALYZE SELECT * FROM (SELECT * FROM pool1 WHERE val>=250 UNION SELECT * FROM pool2 WHERE val>=250) as T;

b. EXPLAIN ANALYZE SELECT * FROM (SELECT * FROM pool1 UNION SELECT * FROM pool2) as T WHERE T.val>=250;



```
postgres=# EXPLAIN ANALYZE SELECT * FROM (SELECT * FROM pool1 WHERE val>=250 UNION SELECT * FROM pool2 WHERE val>=250) as T;
                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=935534.21..960478.98 rows=4988955 width=4) (actual time=1805.901..2291.083 rows=251 loops=1)
   ->  Sort  (cost=935534.21..948006.59 rows=4988955 width=4) (actual time=1805.900..2104.473 rows=5009594 loops=1)
         Sort Key: pool1.val
         Sort Method: external merge  Disk: 68576kB
         ->  Append  (cost=0.00..244082.32 rows=4988955 width=4) (actual time=0.342..769.355 rows=5009594 loops=1)
               ->  Seq Scan on pool1  (cost=0.00..84624.00 rows=2501808 width=4) (actual time=0.341..338.181 rows=2506056 loops=1)
                     Filter: (val >= 250)
                     Rows Removed by Filter: 2493944
               ->  Seq Scan on pool2  (cost=0.00..84624.00 rows=2487147 width=4) (actual time=0.029..282.814 rows=2503538 loops=1)
                     Filter: (val >= 250)
                     Rows Removed by Filter: 2496462
 Planning Time: 0.077 ms
 Execution Time: 2483.586 ms
(13개 행)


postgres=# EXPLAIN ANALYZE SELECT * FROM (SELECT * FROM pool1 UNION SELECT * FROM pool2) as T WHERE T.val>=250;
                                                                QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=935534.21..960478.98 rows=4988955 width=4) (actual time=1810.340..2297.748 rows=251 loops=1)
   ->  Sort  (cost=935534.21..948006.59 rows=4988955 width=4) (actual time=1810.338..2112.598 rows=5009594 loops=1)
         Sort Key: pool1.val
         Sort Method: external merge  Disk: 68576kB
         ->  Append  (cost=0.00..244082.32 rows=4988955 width=4) (actual time=0.366..772.165 rows=5009594 loops=1)
               ->  Seq Scan on pool1  (cost=0.00..84624.00 rows=2501808 width=4) (actual time=0.365..338.474 rows=2506056 loops=1)
                     Filter: (val >= 250)
                     Rows Removed by Filter: 2493944
               ->  Seq Scan on pool2  (cost=0.00..84624.00 rows=2487147 width=4) (actual time=0.026..282.325 rows=2503538 loops=1)
                     Filter: (val >= 250)
                     Rows Removed by Filter: 2496462
 Planning Time: 0.073 ms
 Execution Time: 2508.557 ms
(13개 행)
```

- Two queries are executed with the same plan
  - The query optimizer tries to derive an optimized execution plan