

1)A)OLD QUERY 6 WITH NO INDEXES:

- Flags:1- set enable_seqscan = on;
2- set enable_bitmapscan = off;
3-set enable_indexscan = off;
4- set enable_indexonlyscan = off;

```
schema2=#
schema2=#
schema2=#
schema2=#
schema2=# explain analyze
schema2=#       select dnumber, count(*)
schema2=# from department, employee
schema2=# where dnumber=dno
schema2=# and
schema2=# salary > 40000
schema2=# and
schema2=# dno in (
schema2=# select dno
schema2=# from employee
schema2=# group by dno
schema2=# having count (*) > 5)
schema2=# group by dnumber;

QUERY PLAN

HashAggregate  (cost=1030.00..1032.30 rows=150 width=12) (actual time=15.404..15.424 rows=100 loops=1)
  Group Key: department.dnumber
  Batches: 1 Memory Usage: 48kB
  -> Hash Join  (cost=511.30..1012.94 rows=3572 width=4) (actual time=5.150..13.516 rows=10569 loops=1)
    Hash Cond: (employee.dno = department.dnumber)
    -> Hash Join  (cost=506.00..997.92 rows=3572 width=0) (actual time=5.098..11.285 rows=10569 loops=1)
      Hash Cond: (employee.dno = employee_1.dno)
      -> Seq Scan on employee  (cost=0.00..463.00 rows=10716 width=4) (actual time=0.006..3.663 rows=10716 loops=1)
        Filter: (salary > 40000)
        Rows Removed by Filter: 5284
      -> Hash  (cost=505.30..505.30 rows=50 width=4) (actual time=5.085..5.087 rows=100 loops=1)
        Buckets: 1024 Batches: 1 Memory Usage: 12kB
        -> HashAggregate  (cost=503.00..504.00 rows=50 width=4) (actual time=5.046..5.069 rows=100 loops=1)
          Group Key: employee_1.dno
          Filter: (count(*) > 5)
          Batches: 1 Memory Usage: 48kB
          Rows Removed by Filter: 50
          -> Seq Scan on employee_1  (cost=0.00..423.00 rows=16000 width=4) (actual time=0.005..1.182 rows=16000 loops=1)
            Filter: (count(*) > 5)
            Buckets: 1024 Batches: 1 Memory Usage: 14kB
            -> Seq Scan on department  (cost=0.00..3.50 rows=150 width=4) (actual time=0.014..0.030 rows=150 loops=1)
              Planning Time: 0.291 ms
              Execution Time: 15.518 ms
              (23 rows)
```

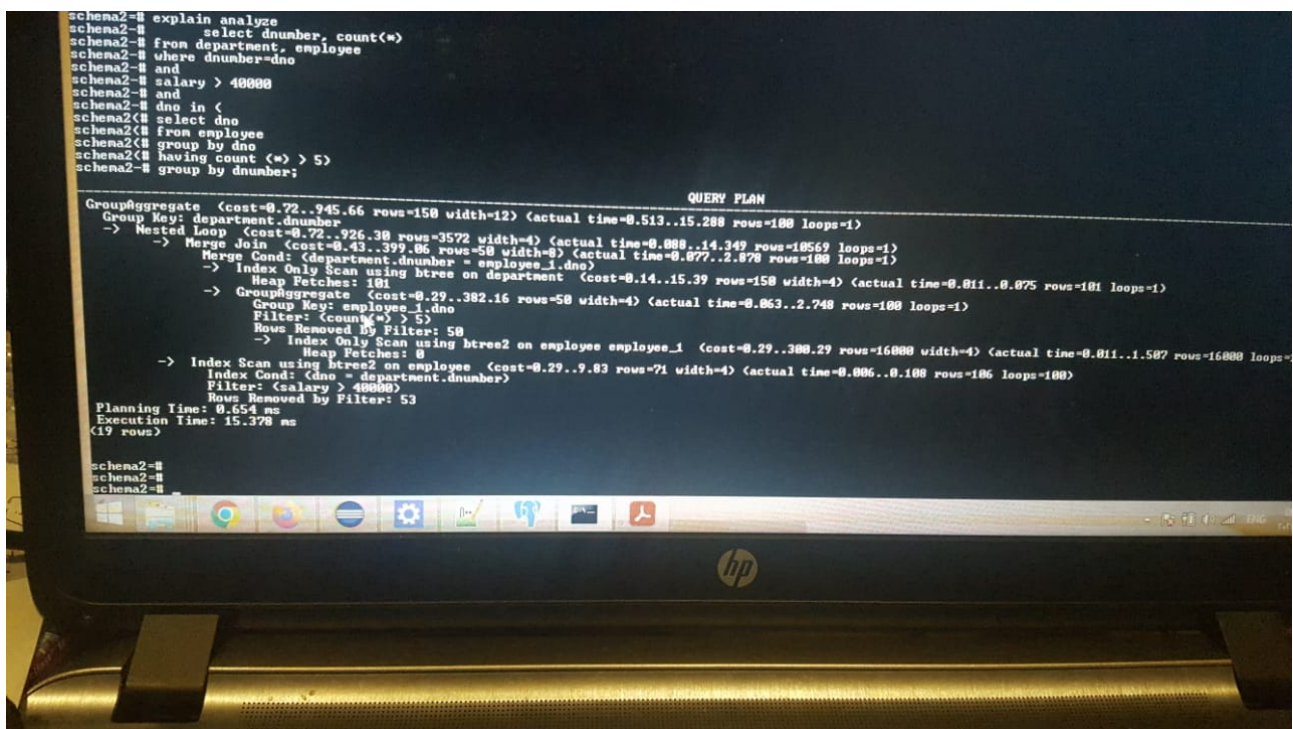
*****CONCLUSION OLD QUERY 6 NO INDEXES:*****

- 1-OLD QUERY 6 NO INDEXES COST = 1032
2-OLD QUERY 67 NO INDEXES TIME = 15.5 MS

1)B)OLD QUERY 6 WITH B+ INDEXES:

Flags:1- set enable_seqscan = off;
2- set enable_bitmapscan = on;
3- set enable_indexscan = on;
4- set enable_indexonlyscan = on;

B+ Indexes: 1- department(Dnumber).
2 - employee(Dno).
3 - employee(salary)



*****CONCLUSION OLD QUERY 6 B+ INDEXES.*****

1-OLD QUERY 6 NO INDEXES COST = 1032
2-OLD QUERY 6 NO INDEXES TIME = 15.5 MS
3-OLD QUERY 6 B+ INDEXES COST = 945
4-OLD QUERY 6 B+ INDEXES TIME = 15.3MS
=> 8.3% more cost efficient & 1.3% more time efficient

1)C)OLD QUERY 6 WITH Hash INDEXES:

Flags: 1- set enable_seqscan = off;
2- set enable_bitmapscan = on;
3- set enable_indexscan = on;
4- set enable_indexonlyscan = on;

Hash Indexes: 1- department(Dnumber).
2 - employee(Dno).
3 - employee(salary)

```
SQL Shell (psql)

schema2=#
schema2=#
schema2=#
schema2=#
schema2=#
schema2=#
schema2=#
schema2=# explain analyze
schema2=# select dnumber, count(*)
schema2=# from department, employee
schema2=# where dnumber=dno
schema2=# and
schema2=# salary > 40000
schema2=# and
schema2=# dno in (
schema2=# select dno
schema2=# from employee
schema2=# group by dno
schema2=# having count (*) > 5)
schema2=# group by dnumber;

QUERY PLAN
GroupAggregate (cost=10000001540.41..10000002292.09 rows=150 width=12) (actual time=6.657..28.812 rows=100 loops=1)
  Group Key: department.dnumber
  -> Nested Loop (cost=10000001540.41..10000002273.53 rows=3572 width=4) (actual time=6.281..19.176 rows=18569 loops=1)
    Merge Join (cost=10000001540.41..10000001677.83 rows=50 width=0) (actual time=6.166..8.523 rows=100 loops=1)
      -> Index Only Scan using department_pkey on department (cost=0.14..15.39 rows=150 width=4) (actual time=0.824..0.885 rows=101 loops=1)
      -> GroupAggregate (cost=10000001540.26..10000001562.14 rows=50 width=4) (actual time=6.128..8.370 rows=100 loops=1)
        Group Key: employee_1.dno
        Filter: (count(*) > 5)
        Rows Removed by Filter: 50
        -> Sort (cost=10000001540.26..10000001580.26 rows=16000 width=4) (actual time=6.819..6.557 rows=16000 loops=1)
          Sort Key: employee_1.dno
          Sort Method: quicksort Memory: 1175kB
          -> Seq Scan on employee employee_1 (cost=0.00..11.15 rows=71 width=4) (actual time=0.813..3.138 rows=16000 loops=1)
            Index Cond: (dno = department.dnumber)
            Filter: (salary > 40000)
            Rows Removed by Filter: 53
    Planning Time: 0.733 ms
    Execution Time: 20.367 ms
  (21 rows)
```

*****CONCLUSION OLD QUERY 6 HASH INDEXES:*****

- 1-OLD QUERY 6 NO INDEXES COST = 1032
- 2-OLD QUERY 6 NO INDEXES TIME = 15.5 MS
- 3-OLD QUERY 6 Hash INDEXES COST = 100000002292
- 4-OLD QUERY 6 HASH INDEXES TIME = 20

*** As expected much worse as the query needs the seqscan because it contains lots of aggregates and group by and the hash in this case very bad.

1)D)OLD QUERY 6 WITH BRIN INDEXES:

Flags: 1- set enable_seqscan = off;

2- set enable_bitmapscan = on;

3- set enable_indexscan = on;

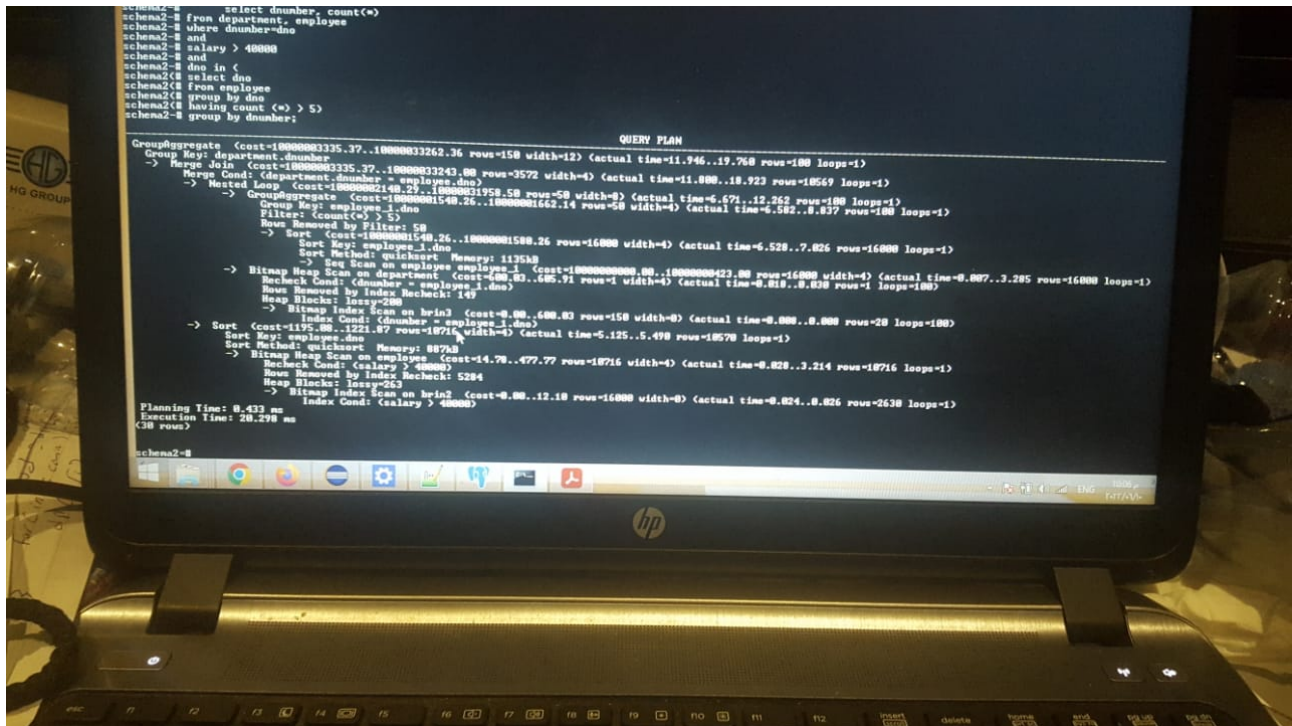
4- set enable_indexonlyscan = on;

5- update pg_index set indisvalid = false where indexrelid =
'DEPARTMENT_pkey'::regclass ;

BRIN Indexes: 1- department(Dnumber).

2 - employee(Dno).

3 - employee(salary)



*****CONCLUSION OLD QUERY 6 BRIN INDEXES:*****

1-OLD QUERY 6 NO INDEXES COST = 1032

2-OLD QUERY 6 NO INDEXES TIME = 15.5 MS

3-OLD QUERY 6 BRIN INDEXES COST = 1000000033262

4-OLD QUERY 6 BRIN INDEXES TIME = 20

*** As expected much worse as the query needs the seqscan because it contains lots of aggregates and group by and the BRIN in this case very bad.

1)E)OLD QUERY 6 WITH MY CHOICE OF INDEXES:

B+ trees and seqscan strategy(best for aggregates)same as 1)b)

Flags:1- set enable_seqscan = on;

2- set enable_bitmapscan =on;

3-set enable_indexscan = on;

4- set enable_indexonlyscan = on;

B+ Indexes: 1- department(Dnumber).

2 - employee(Dno).

3 - employee(salary)

*****CONCLUSION OLD QUERY 6 B+ INDEXES:*****

1-OLD QUERY 6 NO INDEXES COST = 1032

2-OLD QUERY 6 NO INDEXES TIME = 15.5 MS

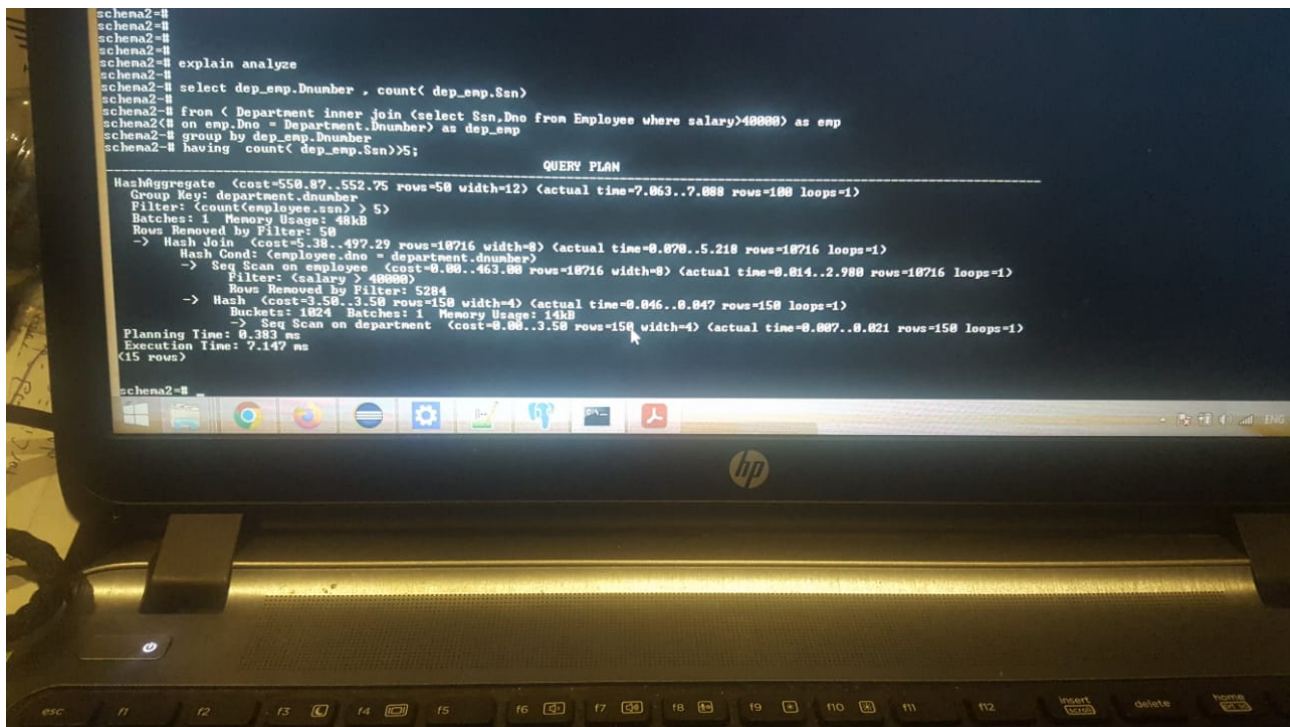
3-OLD QUERY 6 B+ INDEXES COST = 945

4-OLD QUERY 6 B+ INDEXES TIME = 15.3MS

=> 8.3% more cost efficient & 1.3% more time efficient

2)A)OPTIMIZED QUERY 6 WITH NO INDEXES:

- Flags:1- set enable_seqscan = on;
2- set enable_bitmapscan = off;
3-set enable_indexscan = off;
4- set enable_indexonlyscan = off;



*****CONCLUSION OPTIMIZED QUERY 6 NO INDEXES:*****

- 1-OLD QUERY 6 NO INDEXES COST = 1032
 - 2-OLD QUERY 67 NO INDEXES TIME = 15.5 MS
 - 3-OPTIMIZED QUERY 6 NO INDEXES COST = 550
 - 4-OPTIMIZED QUERY 67 NO INDEXES TIME = 7.1 MS
- =>47% MORE COST EFFICIENT & 55% MORE TIME EFFICIENT

2)B)OPTIMIZED QUERY 6 WITH B+ INDEXES:

Flags:1- set enable_seqscan = off;
2- set enable_bitmapscan = on;
3- set enable_indexscan = on;
4- set enable_indexonlyscan = on;

B+ Indexes: 1- department(Dnumber).
2 - employee(Dno).
3 - employee(salary)

```
schema2=#  
schema2=#  
schema2=#  
schema2=#  
schema2=#  
schema2=#  
schema2=#  
schema2=# explain analyze  
schema2=# select dep_emp.Dnumber , count( dep_emp.Ssn)  
schema2=#  
schema2=# from < Department inner join (select Ssn,Dno from Employee where salary>40000) as emp  
schema2=# on emp.Dno = Department.Dnumber) as dep_emp  
schema2=# group by dep_emp.Dnumber  
schema2=# having count( dep_emp.Ssn)>5;  
  
QUERY PLAN  
-----  
HashAggregate (cost=624.05..625.93 rows=50 width=12) (actual time=6.426..6.451 rows=100 loops=1)  
  Group Key: department.dnumber  
  Filter: (count(employee.ssn) > 5)  
  Batches: 1  Memory Usage: 48kB  
  Rows Removed by Filter: 50  
    -> Hash Join (cost=144.60..570.47 rows=10716 width=8) (actual time=0.514..4.627 rows=10716 loops=1)  
      Hash Cond: (employee.dno = department.dnumber)  
      -> Bitmap Heap Scan on employee (cost=127.33..524.28 rows=10716 width=8) (actual time=0.435..1.770 rows=10716 loops=1)  
        Recheck Cond: (salary > 40000)  
        Heap Blocks: exact=263  
        -> Bitmap Index Scan on btree3 (cost=0.00..124.65 rows=10716 width=0) (actual time=0.397..0.398 rows=10716 loops=1)  
          Index Cond: (salary > 40000)  
      -> Hash (cost=15.39..15.39 rows=150 width=4) (actual time=0.069..0.069 rows=150 loops=1)  
        Buckets: 1024 Batches: 1  Memory Usage: 14kB  
        -> Index Only Scan using btree3 on department (cost=0.14..15.39 rows=150 width=4) (actual time=0.010..0.043 rows=150 loops=1)  
          Heap Fetches: 150  
  
Planning Time: 0.343 ms  
Execution Time: 6.551 ms  
(18 rows)  
  
schema2=#  
schema2=#
```


*****CONCLUSION OPTIMIZED QUERY 6 B+ INDEXES:*****

1-OPTIMIZED QUERY 6 NO INDEXES COST = 550

2-OPTIMIZED QUERY 67 NO INDEXES TIME = 7.1 MS

3-OPTIMIZED QUERY 6 B+ INDEXES COST = 624

4-OPTIMIZED QUERY 6 B+ INDEXES TIME = 6.1MS

=> more or less same performance as seqscan but because
Of aggregates count () and group by seqscan slightly better.

2)C)OPTIMIZED QUERY 6 WITH Hash INDEXES:

Flags:1- set enable_seqscan = off;

2- set enable_bitmapscan =on;

3-set enable_indexscan = on;

4- set enable_indexonlyscan = on;

Hash Indexes: 1- department(Dnumber).

2 - employee(Dno).

3 - employee(salary)

```
schema2=# select dep_emp.Dnumber , count( dep_emp.Ssn)
schema2=#
schema2=# from ( Department inner join (select Ssn,Dno from Employee where salary>40000) as emp
schema2=# on emp.Dno = Department.Dnumber) as dep_emp
schema2=# group by dep_emp.Dnumber
schema2=# having count( dep_emp.Ssn)>5;

QUERY PLAN
GroupAggregate (cost=0.14..1854.35 rows=50 width=12) (actual time=0.290..10.985 rows=100 loops=1)
  Group Key: department.dnumber
  Filter: (count(employee.ssn) > 5)
  Rows Removed by Filter: 50
  -> Nested Loop (cost=0.14..1798.90 rows=10716 width=0) (actual time=0.028..10.062 rows=10716 loops=1)
    -> Index Only Scan using department_pkey on department (cost=0.14..15.39 rows=150 width=4) (actual time=0.011..0.055 rows=150 loops=1)
      Heap Fetches: 150
    -> Index Scan using hash on employee (cost=0.00..11.18 rows=71 width=0) (actual time=0.003..0.061 rows=71 loops=150)
      Index Cond: (dno = department.dnumber)
      Filter: (salary > 40000)
      Rows Removed by Filter: 35
Planning Time: 0.237 ms
Execution Time: 11.020 ms
(13 rows)

schema2=#
schema2=#
schema2=#
schema2=#
schema2=#
```


*****CONCLUSION OPTIMIZED QUERY 6 HASH INDEXES.*****

1-OPTIMIZED QUERY 6 NO INDEXES COST = 550

2-OPTIMIZED QUERY 67 NO INDEXES TIME = 7.1 MS

3- OPTIMIZED QUERY 6 Hash INDEXES COST = 1854

4-OPTIMIZED QUERY 6 HASH INDEXES TIME = 11

*** As expected much worse as the query needs the seqscan because it contains lots of aggregates and group by and the hash in this case very bad.

2)D)OPTIMIZED QUERY 6 WITH BRIN INDEXES:

Flags:1- set enable_seqscan = off;

2- set enable_bitmapscan = on;

3-set enable_indexscan = on;

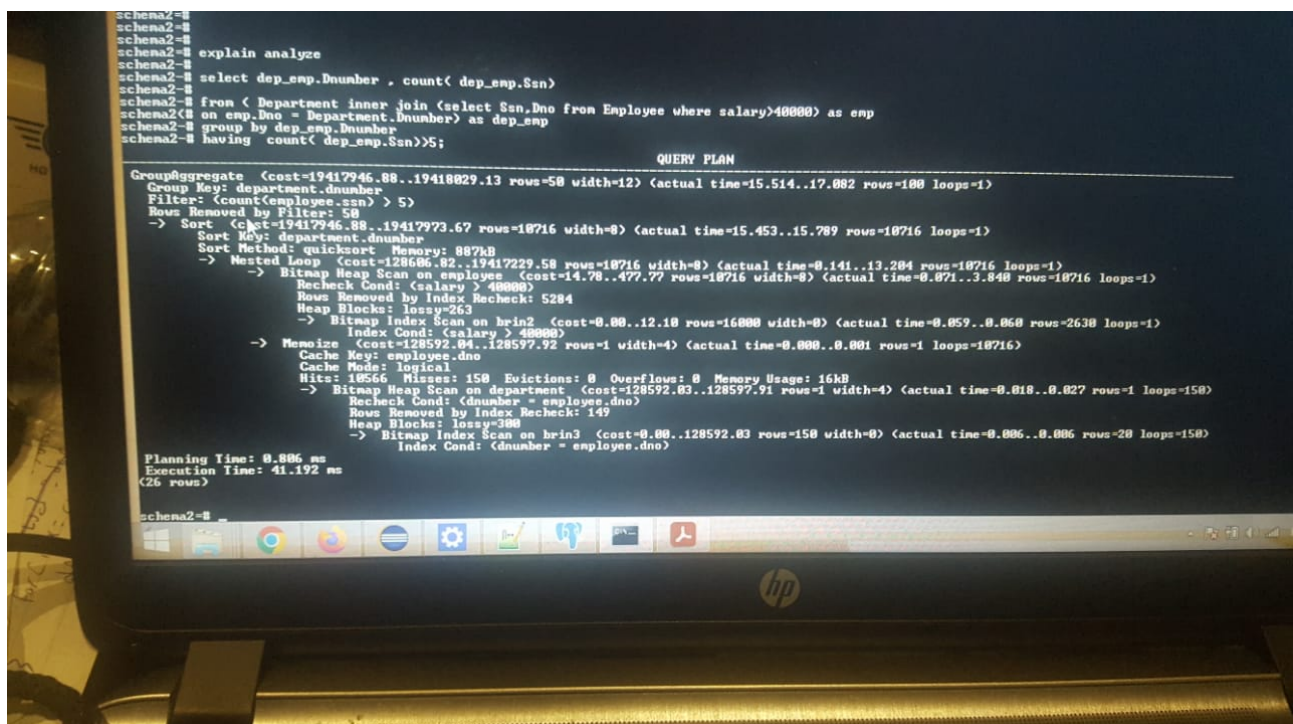
4- set enable_indexonlyscan = on;

5- update pg_index set indisvalid = false where indexrelid = 'DEPARTMENT_pkey'::regclass ;

BRIN Indexes: 1- department(Dnumber).

2 - employee(Dno).

3 - employee(salary)



*****CONCLUSION OPTIMIZED QUERY 6 BRIN INDEXES:*****

1-OPTIMIZED QUERY 6 NO INDEXES COST = 550

2-OPTIMIZED QUERY 67 NO INDEXES TIME = 7.1 MS

3-OPTIMIZED QUERY 6 BRIN INDEXES COST = 19418029

4-OPTIMIZED QUERY 6 BRIN INDEXES TIME = 41

*** As expected much worse as the query needs the seqscan because it contains lots of aggregates and group by and the BRIN in this case very bad.

1)E)OPTIMIZED QUERY 6 WITH MY CHOICE OF INDEXES:

For this optimised query I choose no indexes and full seqscans
As it contains aggregates count() and this kills the indexes also it
Is nested with multiple dependencies

Flags:1- set enable_seqscan = on;
2- set enable_bitmapscan =off;
3-set enable_indexscan = off;
4- set enable_indexonlyscan = off;

*****CONCLUSION OPTIMIZED QUERY 6 NO INDEXES:*****

1-OPTIMIZED QUERY 6 NO INDEXES COST = 550

2-OPTIMIZED QUERY 67 NO INDEXES TIME = 7.1 MS

