Query 11

- Find the name of the director (first and last names) who directed a movie that casted a role for 'Eyes Wide Shut'. Empty result set not acceptable.

Note !

- all flags are set to default

Original Query

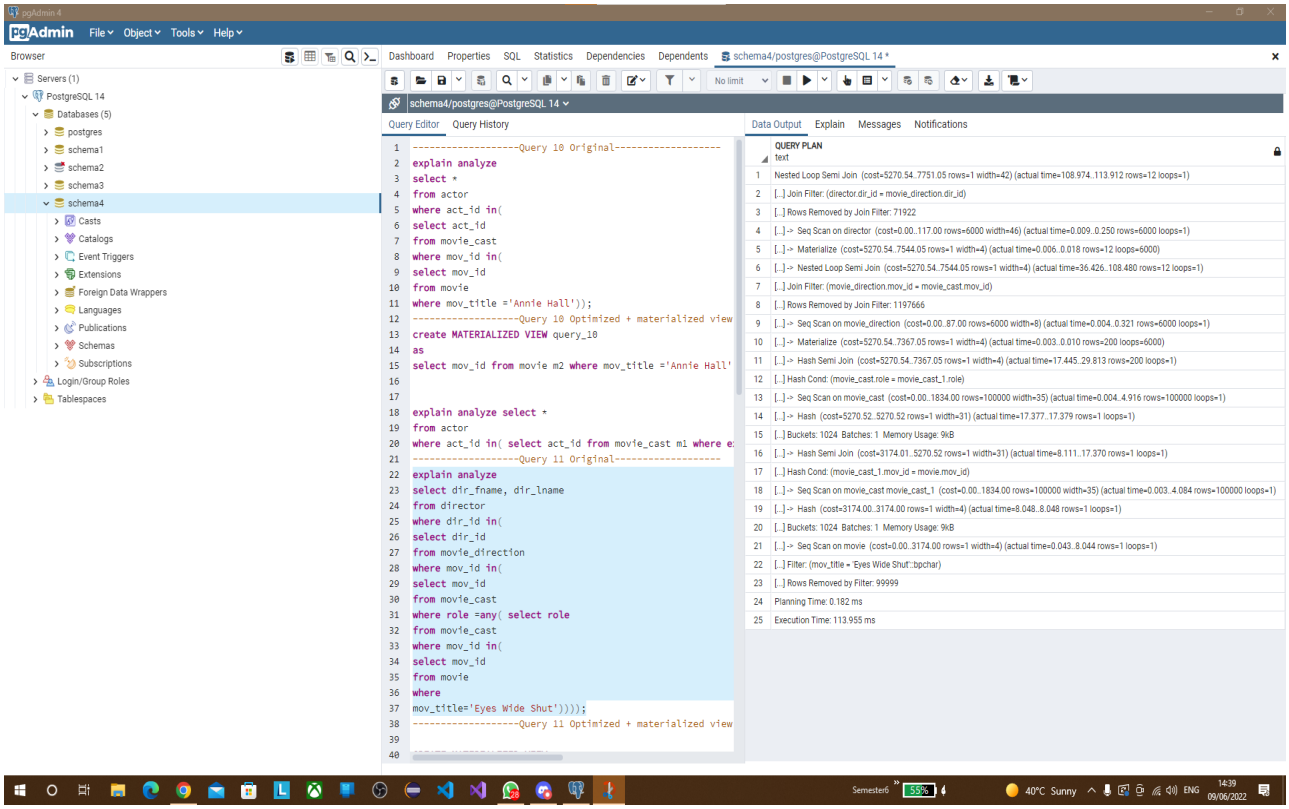
```
select dir_fname, dir_lname
from director
where dir_id in(
select dir_id
from movie_direction
where mov_id in(
select mov_id
from movie_cast
where role =any( select role
from movie_cast
where mov_id in(
select mov_id
from movie
where
mov_title='Eyes Wide Shut'))));
```

Result Set

- 9 Rows

Report

1. given query without an index :

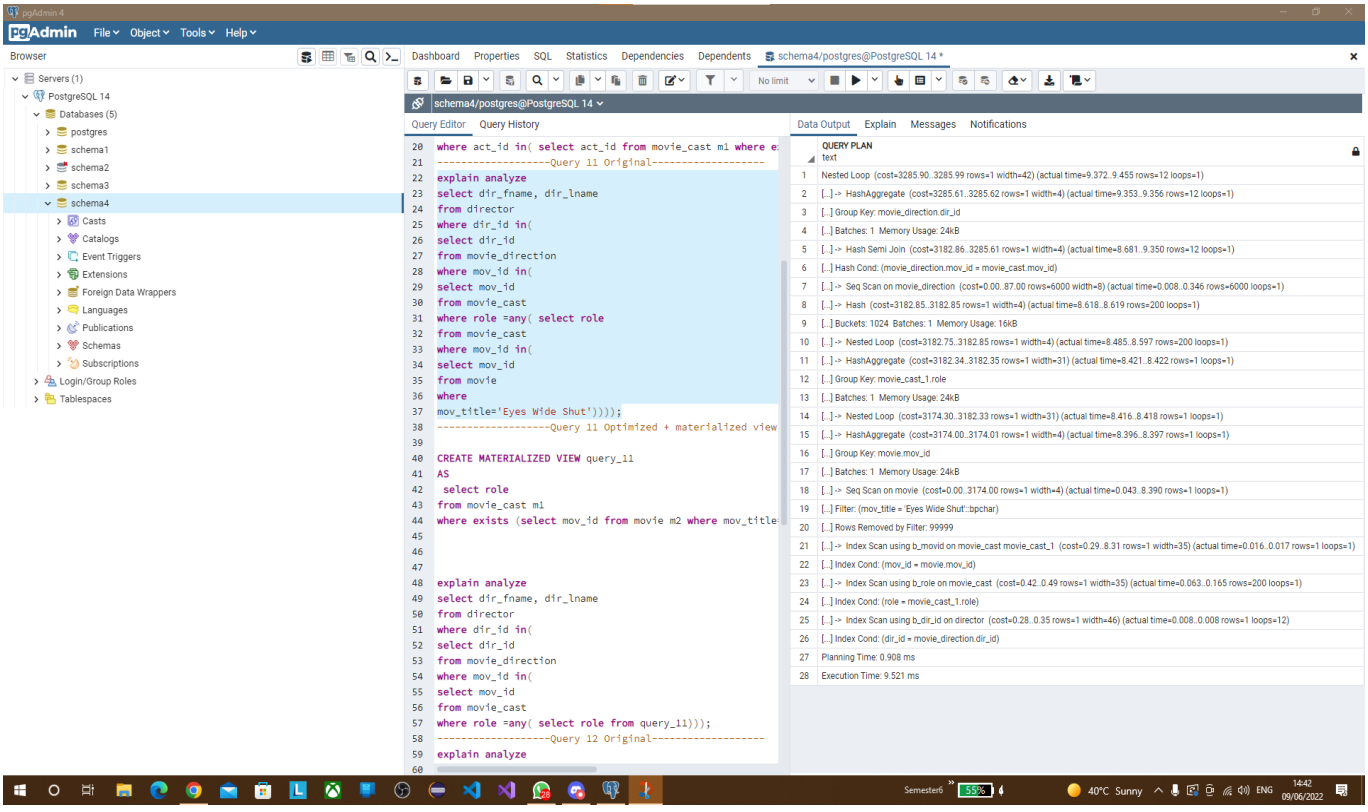


Explanation :

- Metrics :

Execution Time : 113.955 ms Total Expected Cost : 7751.05

2. given query with B+ trees indices only :



Explanation :

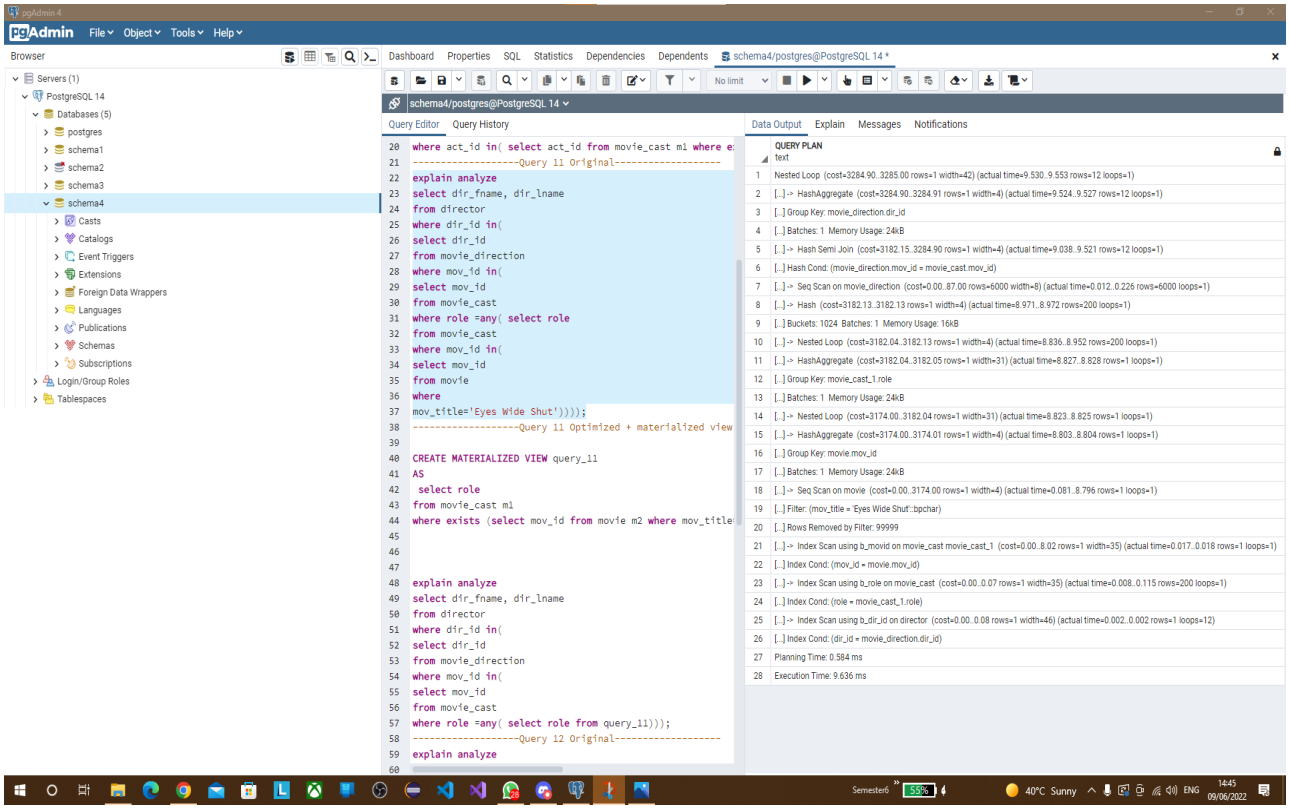
- Metrics :

  Execution Time : 9.521 ms Total Expected Cost : 3285.99

- Index created on column dir_id of table director, column role of table movie_cast, column mov_id of table movie_cast.

- The Search is done with O(Log n) for search using values of column dir_id of table director, column role of table movie_cast, column mov_id of table movie_cast so the performance is improved with respect to performance of the query without index (Total Expected Cost Was 7751.05 and became 3285.99).

3. given query with hash indices only :



Explanation :

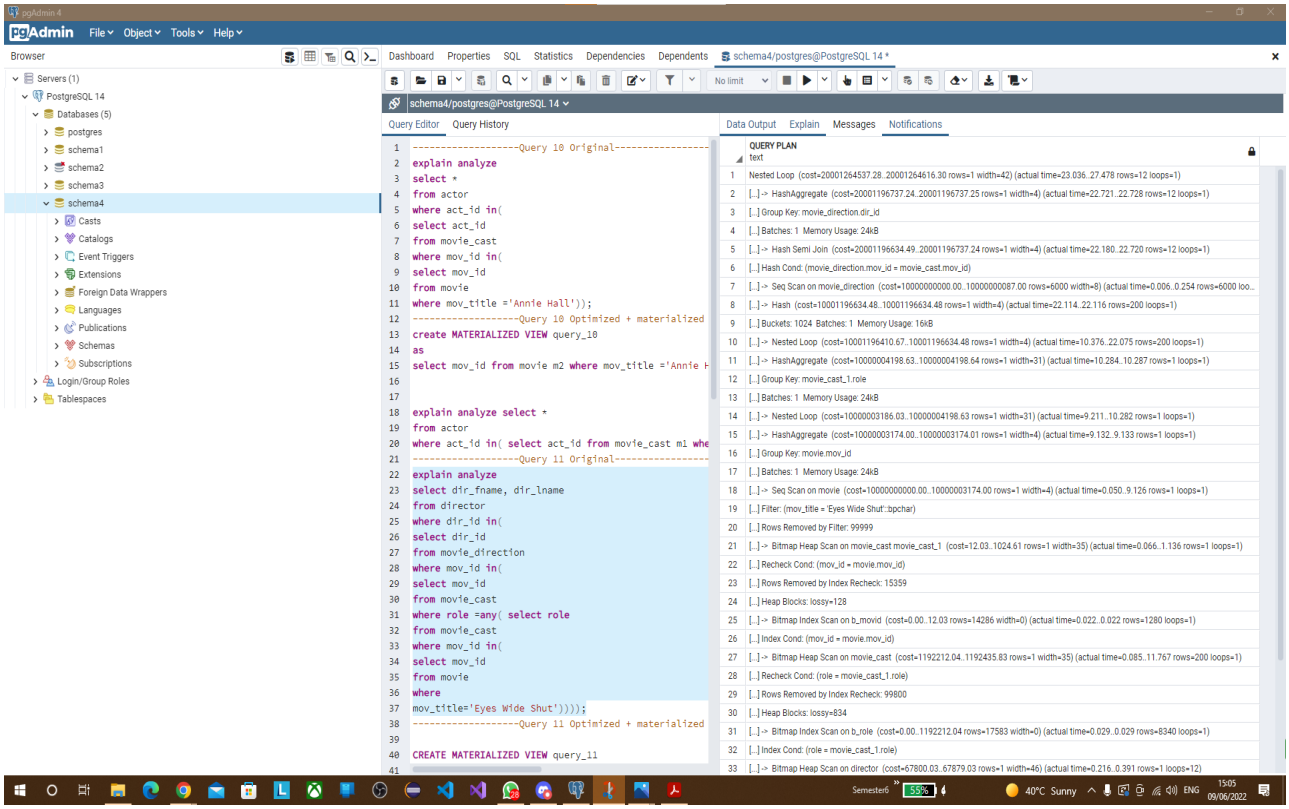
- Metrics :

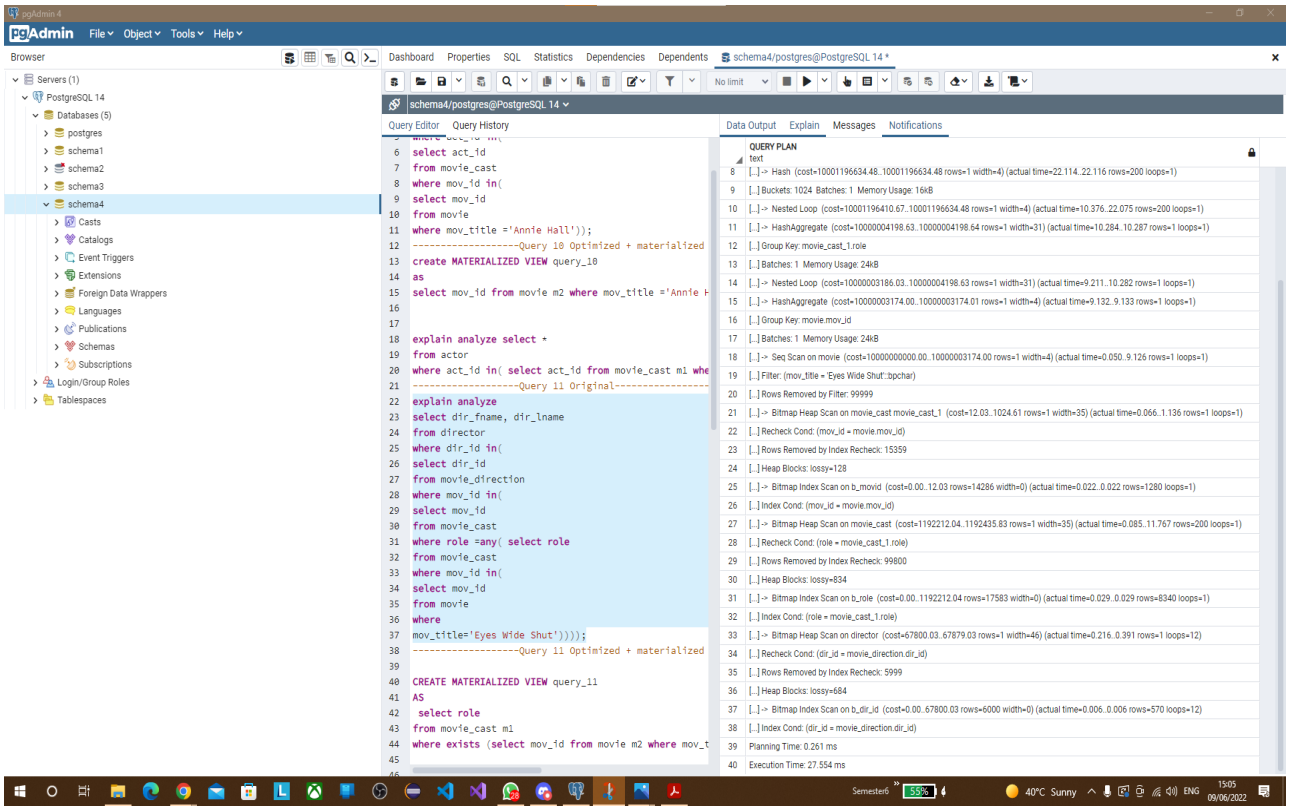
  Execution Time : 9.636 ms Total Expected Cost : 3285

- Index created on column dir_id of table director, column role of table movie_cast, column mov_id of table movie_cast.

- The Search is done with O(1) when we search for values of any column i created index on it so the performance slightly improved with respect to performance of the query with B+ Tree index and the increase in performance is small due to the small number of rows(Total Expected Cost Was 7907.55 without index and 3182.85 with B+ tree and became 3285).

4. given query with BRIN indices only :





Explanation :

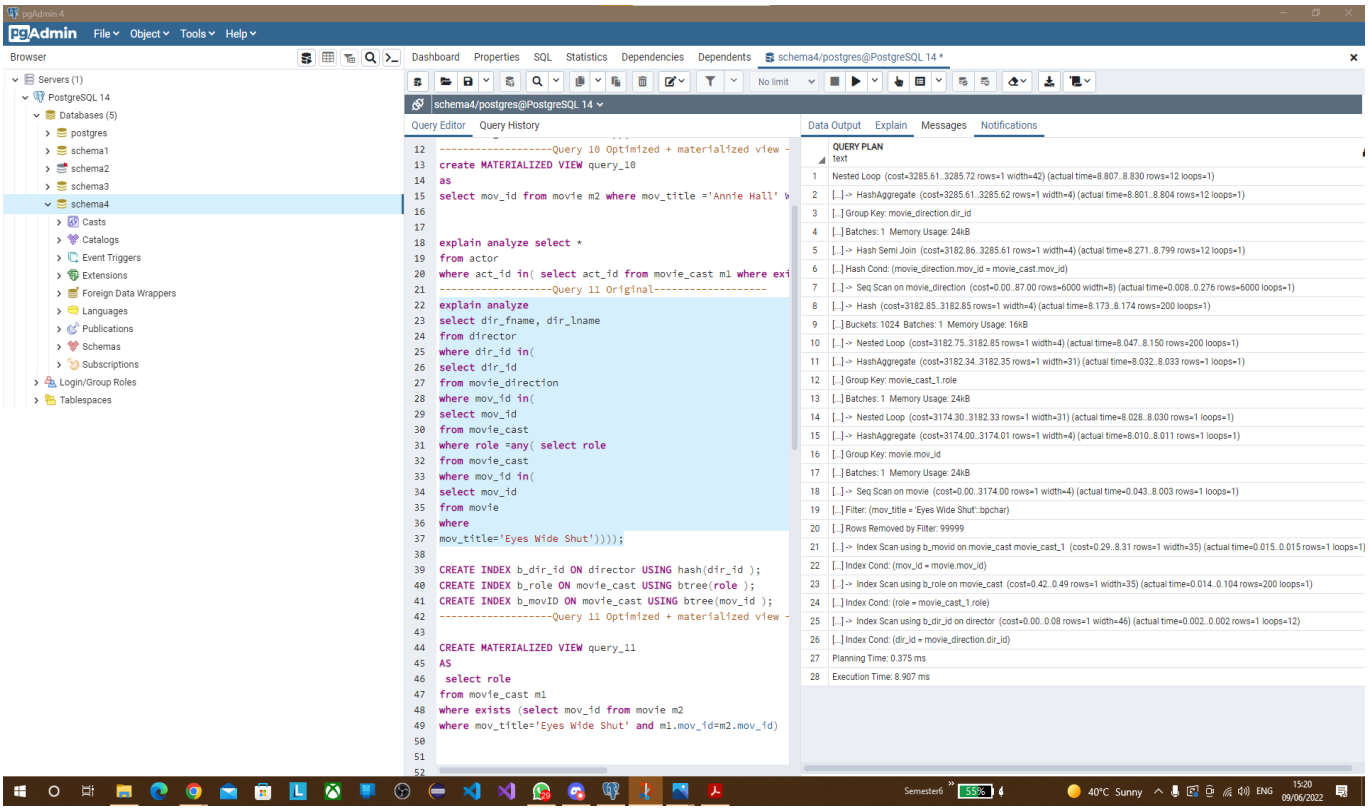
- Metrics :

Execution Time : 27.554 ms	**Total Expected Cost : 20001264616.30**

- Index created on column dir_id of table director, column role of table movie_cast, column mov_id of table movie_cast.

- Here the BRIN was not used in the original Query Plan settings because using brin in our case here is not efficient as all my query searches is not searching in small range within the range of the column "not low selectivity query" so I set flag seqscan=off .
- The Execution Time and Expected Cost became the Worst of all as a result of turning seqscan flag off.

5. given query with mixed indices (any mix of your choice) :



Explanation :

- Metrics :

  Execution Time 8.907 ms Total Expected Cost : 3285.72

- As shown in screenshot Index created on column dir_id of table director(hash), column role of table movie_cast(btree), column mov_id of table movie_cast(btree).
- Now the Performance is better than with B+ index but performance with Hash only is better as any search with values of column dir_id of table director will be in O(1) and with values of column mov_id of table movie_cast or column role of table movie_cast will be in O(Log n).

Optimized Query

```
CREATE MATERIALIZED VIEW query_11
AS
 select role
from movie_cast m1
where exists (select mov_id from movie m2
where mov_title='Eyes Wide Shut' and m1.mov_id=m2.mov_id)
```

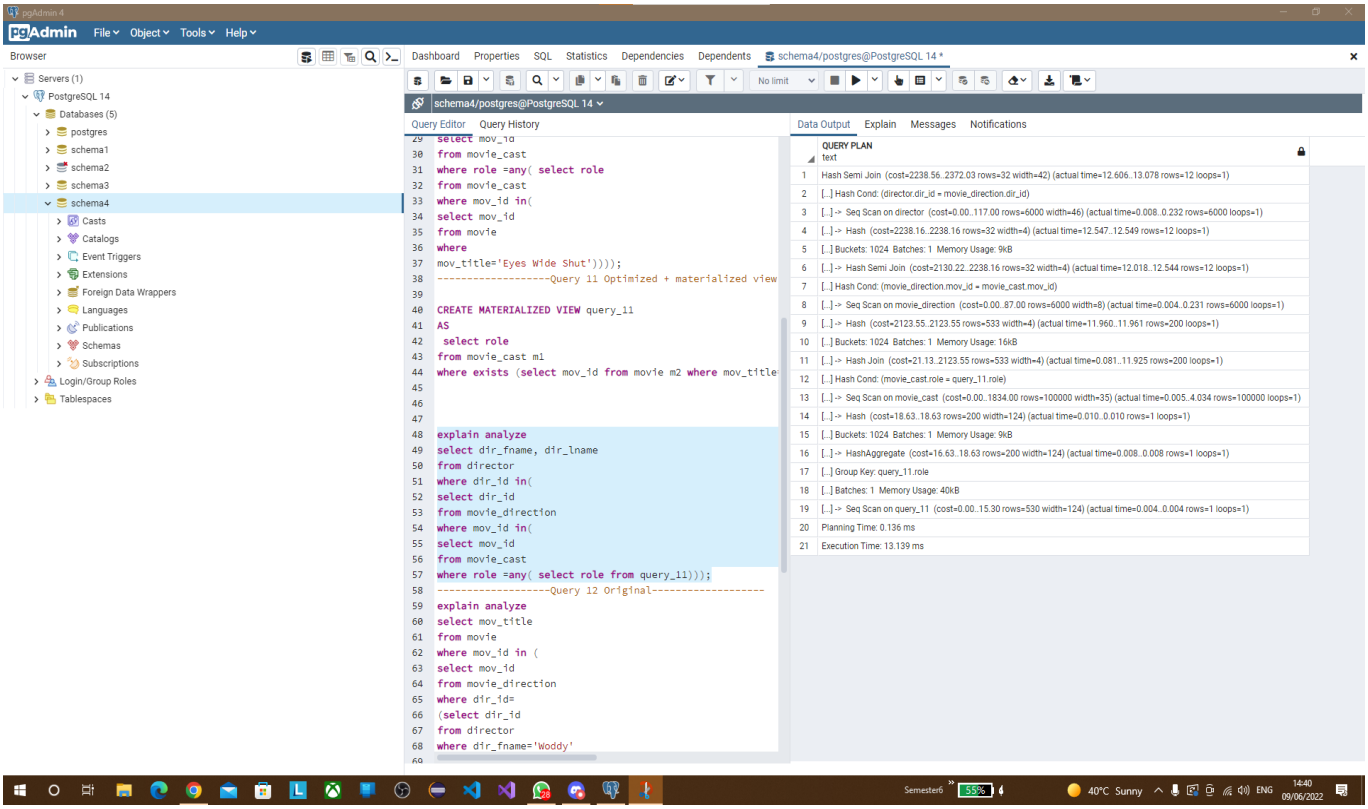
```
explain analyze
select dir_fname, dir_lname
from director
where dir_id in(
select dir_id
from movie_direction
where mov_id in(
select mov_id
from movie_cast
where role =any( select role from query_11)));
```

Result Set

- 9 Rows

Report

1. Optimized Query without an index :



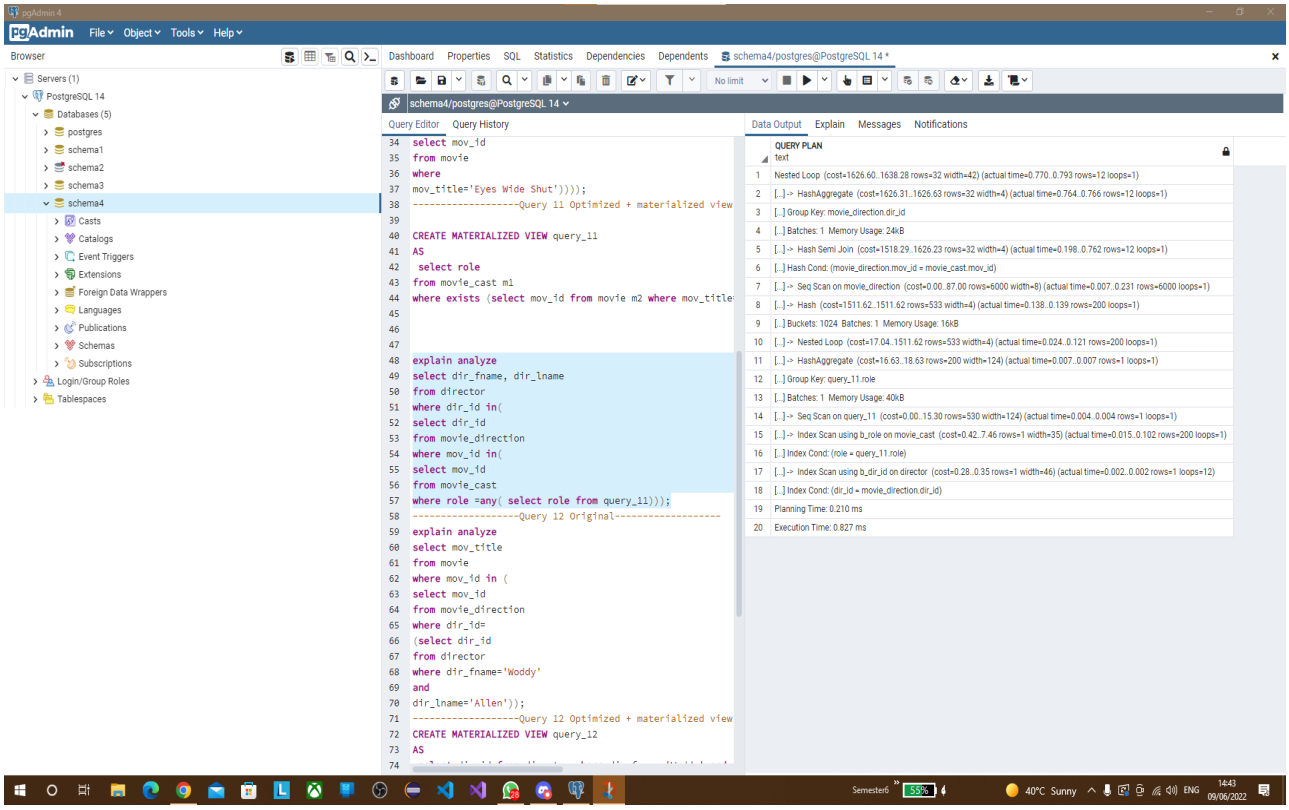
Explanation :

- Metrics :

Execution Time : 13.139 ms	Total Expected Cost : 2372.03

- Reason : Since selecting roles of movie Eyes wide shut is needed as sub query so I created materialized view storing those values with name of the view Query 11 and used it inside my query.So, the performance is improved with respect to performance of original query (Total Expected Cost Was 7751.05 and became 2372.03).

2. Optimized Query with B+ trees indices only :



Explanation :

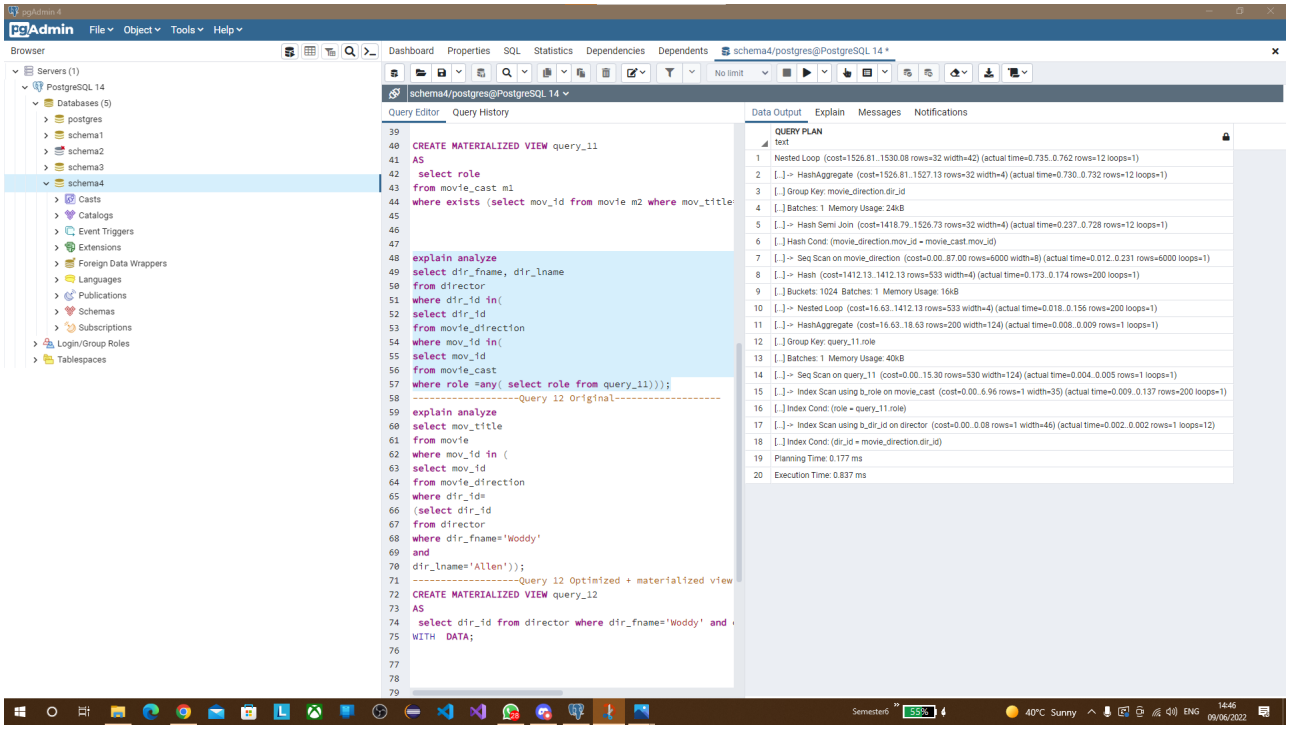
- Metrics :

Execution Time : 0.827 ms	**Total Expected Cost : 1638.28**

- Index created on column dir_id of table director, column role of table movie_cast, column mov_id of table movie_cast.

- The Search is done with O(Log n) for search using values of column dir_id of table director, column role of table movie_cast, column mov_id of table movie_cast so the performance is improved with respect to performance of the query without index (Total Expected Cost Was 2372.03 and became 1638.28).

3. Optimized Query with hash indices only :



Explanation :

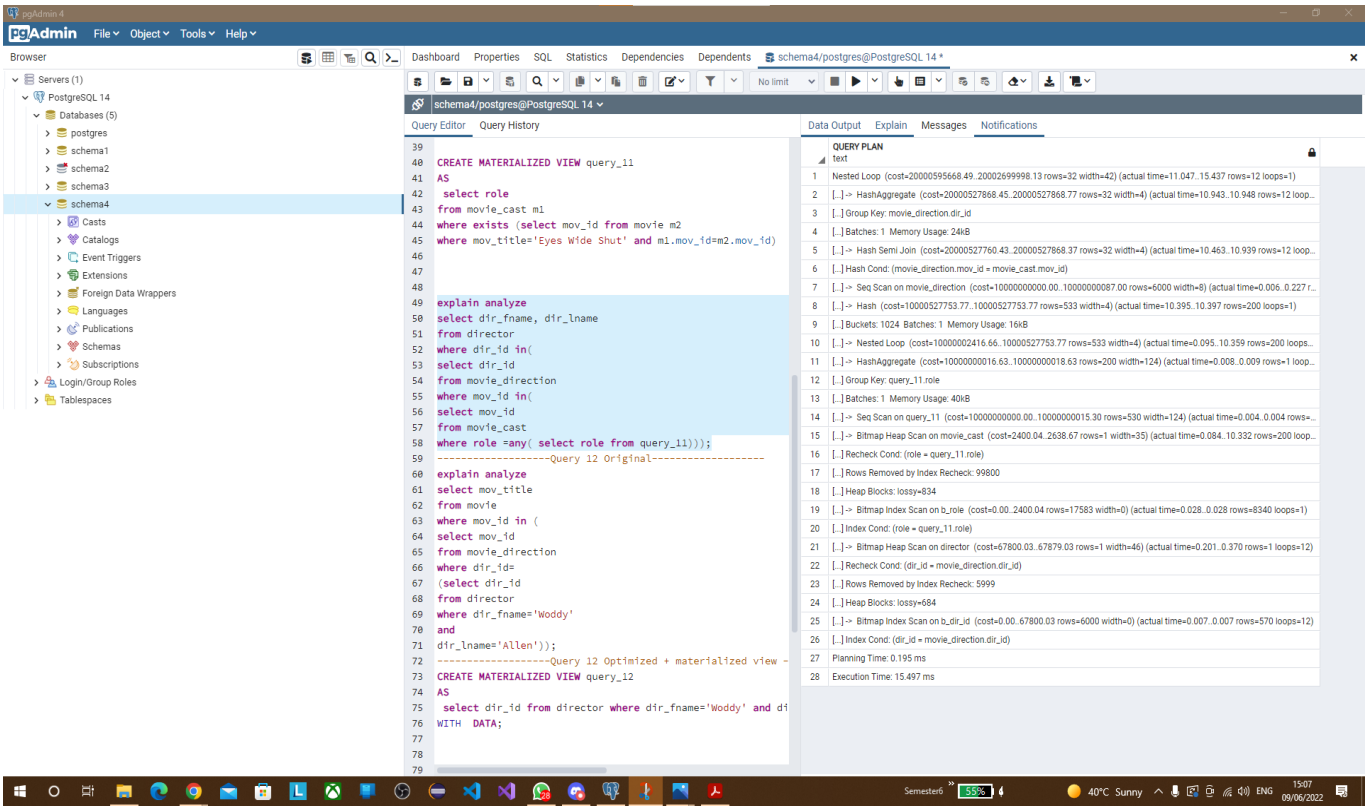
- Metrics :

  Execution Time : 0.837 ms Total Expected Cost : 1530.08

- Index created on column dir_id of table director, column role of table movie_cast, column mov_id of table movie_cast.

- The Search is done with O(1) when we search for values of any column I created hash index on it so the performance slightly improved with respect to performance of the query with B+ Tree index and the increase in performance is small due to the small number of rows(Total Expected Cost Was 2372.03 without index and 1638.28 with B+ tree and became 1530.08).

4. Optimized Query with BRIN indices only :



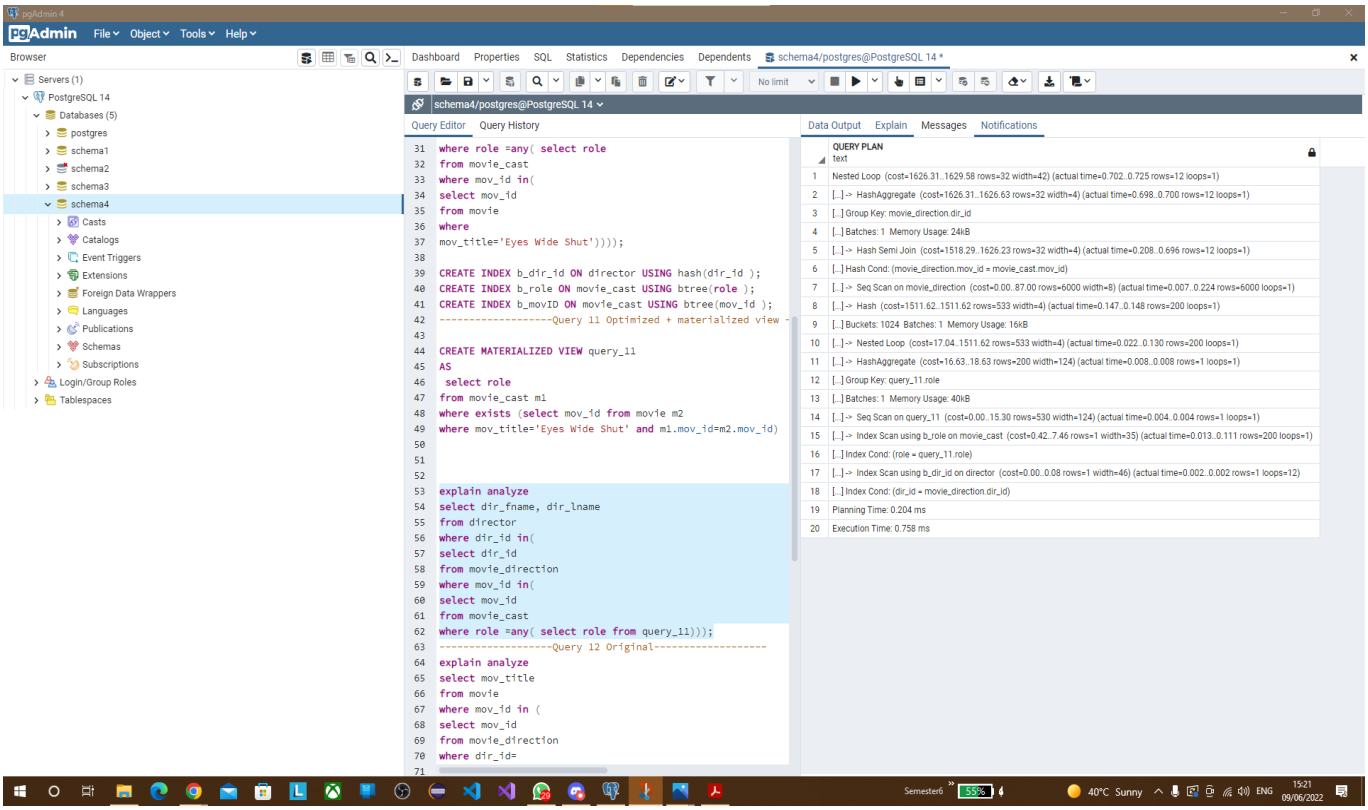
Explanation :

- Metrics :

  Execution Time : 15.497 ms Total Expected Cost : 20002699998.13

- Index created on column dir_id of table director, column role of table movie_cast, column mov_id of table movie_cast.
- Here the BRIN was not used in the original Query Plan settings because using brin in our case here is not efficient as all my query searches is not searching in small range within the range of the column "not low selectivity query" so I set flag seqscan=off.
- The Execution Time and Expected Cost became the Worst of all as a result of turning seqscan flag off.

5. Optimized Query with mixed indices (any mix of your choice) :



Explanation :

- Metrics :

  Execution Time : 0.758 ms Total Expected Cost : 1629.58

- As shown in screenshot Index created on column dir_id of table director(hash), column role of table movie_cast(btree), column mov_id of table movie_cast(btree).

- Now the Performance is better than with B+ index but performance with Hash only is better as any search with values of column dir_id of table director will be in O(1) and with values of column role of table movie_cast will be in O(Log n).