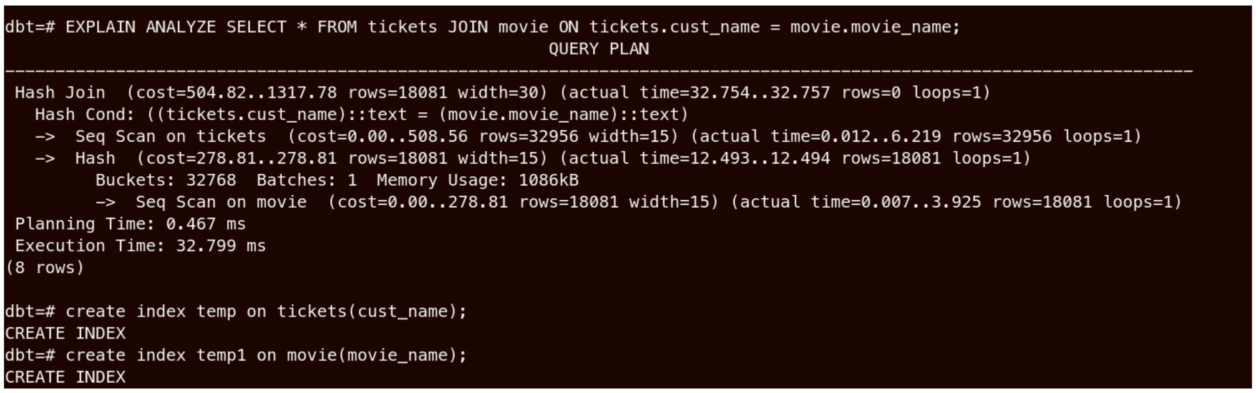
Assignment A3

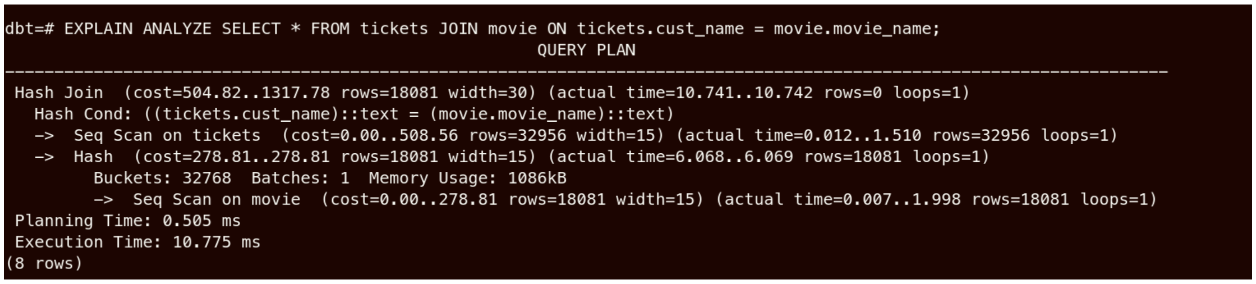
(a) Write SQL queries containing group by, order by, equi-joins, outer joins.
(b) Identify the potential performance bottlenecks by analyzing the execution plan.
(c) Rewrite the queries to improve the performance using the techniques learnt.

Name: Arvind Krishna SRN: PES1UG19CS090	Sem: 6 Sec: B

Equijoin performance improvement due to indexing



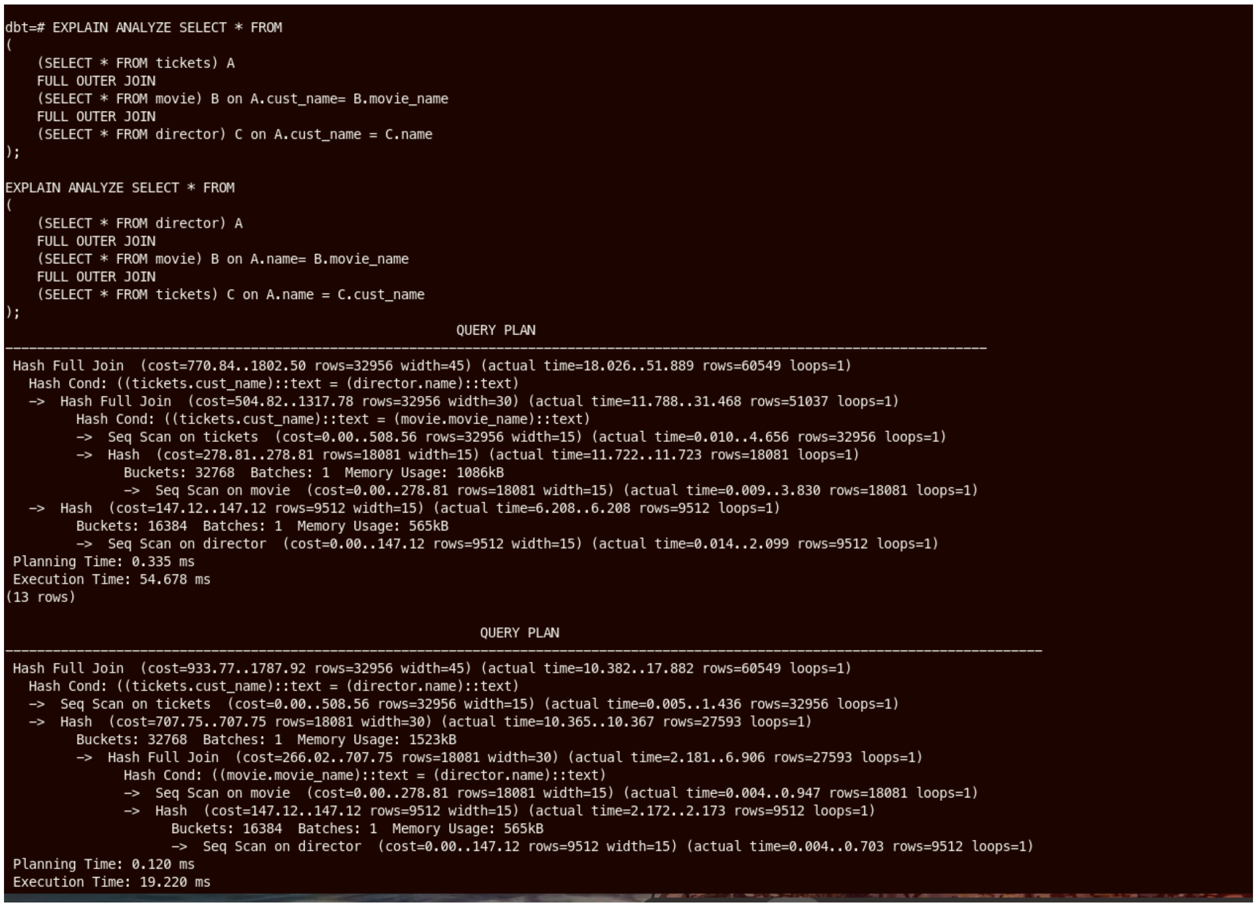
```
dbt=# EXPLAIN ANALYZE SELECT * FROM tickets JOIN movie ON tickets.cust_name = movie.movie_name;
                                                          QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=504.82..1317.78 rows=18081 width=30) (actual time=32.754..32.757 rows=0 loops=1)
   Hash Cond: ((tickets.cust_name)::text = (movie.movie_name)::text)
   ->  Seq Scan on tickets  (cost=0.00..508.56 rows=32956 width=15) (actual time=0.012..6.219 rows=32956 loops=1)
   ->  Hash  (cost=278.81..278.81 rows=18081 width=15) (actual time=12.493..12.494 rows=18081 loops=1)
         Buckets: 32768  Batches: 1  Memory Usage: 1086kB
         ->  Seq Scan on movie  (cost=0.00..278.81 rows=18081 width=15) (actual time=0.007..3.925 rows=18081 loops=1)
 Planning Time: 0.467 ms
 Execution Time: 32.799 ms
(8 rows)

dbt=# create index temp on tickets(cust_name);
CREATE INDEX
dbt=# create index temp1 on movie(movie_name);
CREATE INDEX
```



```
dbt=# EXPLAIN ANALYZE SELECT * FROM tickets JOIN movie ON tickets.cust_name = movie.movie_name;
                                                          QUERY PLAN
----------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=504.82..1317.78 rows=18081 width=30) (actual time=10.741..10.742 rows=0 loops=1)
   Hash Cond: ((tickets.cust_name)::text = (movie.movie_name)::text)
   ->  Seq Scan on tickets  (cost=0.00..508.56 rows=32956 width=15) (actual time=0.012..1.510 rows=32956 loops=1)
   ->  Hash  (cost=278.81..278.81 rows=18081 width=15) (actual time=6.068..6.069 rows=18081 loops=1)
         Buckets: 32768  Batches: 1  Memory Usage: 1086kB
         ->  Seq Scan on movie  (cost=0.00..278.81 rows=18081 width=15) (actual time=0.007..1.998 rows=18081 loops=1)
 Planning Time: 0.505 ms
 Execution Time: 10.775 ms
(8 rows)
```

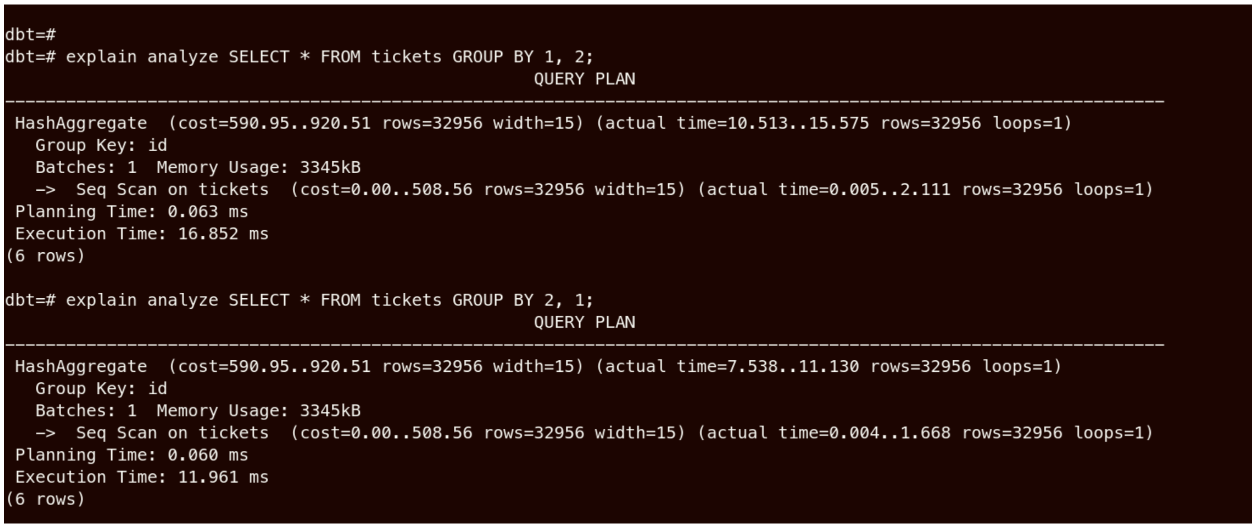
Here we can clearly see that the indexing has improved performance significantly

Changing the order of join to improve FULL OUTER JOIN



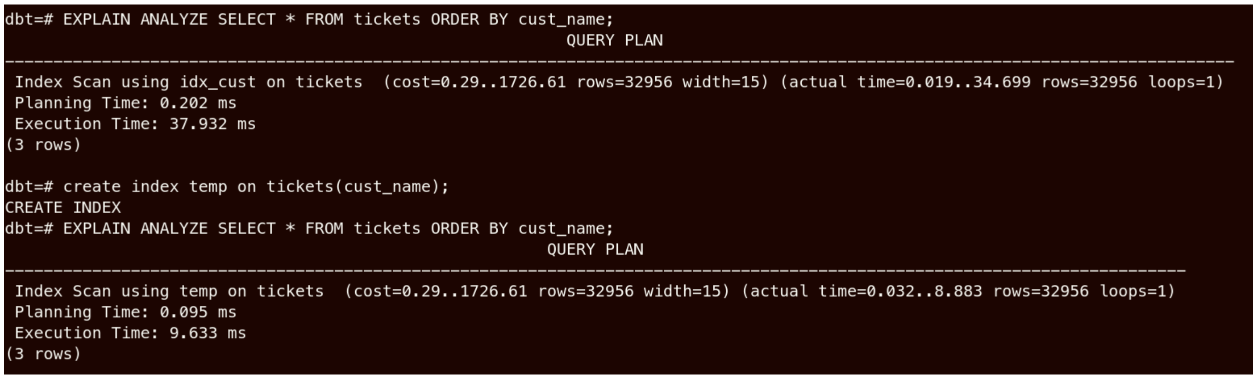
```
dbt=# EXPLAIN ANALYZE SELECT * FROM
(
    (SELECT * FROM tickets) A
    FULL OUTER JOIN
    (SELECT * FROM movie) B on A.cust_name= B.movie_name
    FULL OUTER JOIN
    (SELECT * FROM director) C on A.cust_name = C.name
);

EXPLAIN ANALYZE SELECT * FROM
(
    (SELECT * FROM director) A
    FULL OUTER JOIN
    (SELECT * FROM movie) B on A.name= B.movie_name
    FULL OUTER JOIN
    (SELECT * FROM tickets) C on A.name = C.cust_name
);
                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Hash Full Join  (cost=770.84..1802.50 rows=32956 width=45) (actual time=18.026..51.889 rows=60549 loops=1)
   Hash Cond: ((tickets.cust_name)::text = (director.name)::text)
   ->  Hash Full Join  (cost=504.82..1317.78 rows=32956 width=30) (actual time=11.788..31.468 rows=51037 loops=1)
         Hash Cond: ((tickets.cust_name)::text = (movie.movie_name)::text)
         ->  Seq Scan on tickets  (cost=0.00..508.56 rows=32956 width=15) (actual time=0.010..4.656 rows=32956 loops=1)
         ->  Hash  (cost=278.81..278.81 rows=18081 width=15) (actual time=11.722..11.723 rows=18081 loops=1)
               Buckets: 32768  Batches: 1  Memory Usage: 1086kB
               ->  Seq Scan on movie  (cost=0.00..278.81 rows=18081 width=15) (actual time=0.009..3.830 rows=18081 loops=1)
   ->  Hash  (cost=147.12..147.12 rows=9512 width=15) (actual time=6.208..6.208 rows=9512 loops=1)
         Buckets: 16384  Batches: 1  Memory Usage: 565kB
         ->  Seq Scan on director  (cost=0.00..147.12 rows=9512 width=15) (actual time=0.014..2.099 rows=9512 loops=1)
 Planning Time: 0.335 ms
 Execution Time: 54.678 ms
(13 rows)

                                                           QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------
 Hash Full Join  (cost=933.77..1787.92 rows=32956 width=45) (actual time=10.382..17.882 rows=60549 loops=1)
   Hash Cond: ((tickets.cust_name)::text = (director.name)::text)
   ->  Seq Scan on tickets  (cost=0.00..508.56 rows=32956 width=15) (actual time=0.005..1.436 rows=32956 loops=1)
   ->  Hash  (cost=707.75..707.75 rows=18081 width=30) (actual time=10.365..10.367 rows=27593 loops=1)
         Buckets: 32768  Batches: 1  Memory Usage: 1523kB
         ->  Hash Full Join  (cost=266.02..707.75 rows=18081 width=30) (actual time=2.181..6.906 rows=27593 loops=1)
               Hash Cond: ((movie.movie_name)::text = (director.name)::text)
               ->  Seq Scan on movie  (cost=0.00..278.81 rows=18081 width=15) (actual time=0.004..0.947 rows=18081 loops=1)
               ->  Hash  (cost=147.12..147.12 rows=9512 width=15) (actual time=2.172..2.173 rows=9512 loops=1)
                     Buckets: 16384  Batches: 1  Memory Usage: 565kB
                     ->  Seq Scan on director  (cost=0.00..147.12 rows=9512 width=15) (actual time=0.004..0.703 rows=9512 loops=1)
 Planning Time: 0.120 ms
 Execution Time: 19.220 ms
```

Aggregating the value with more repetition to improve GROUP BY performance



```
dbt=#
dbt=# explain analyze SELECT * FROM tickets GROUP BY 1, 2;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=590.95..920.51 rows=32956 width=15) (actual time=10.513..15.575 rows=32956 loops=1)
   Group Key: id
   Batches: 1  Memory Usage: 3345kB
   ->  Seq Scan on tickets  (cost=0.00..508.56 rows=32956 width=15) (actual time=0.005..2.111 rows=32956 loops=1)
 Planning Time: 0.063 ms
 Execution Time: 16.852 ms
(6 rows)

dbt=# explain analyze SELECT * FROM tickets GROUP BY 2, 1;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=590.95..920.51 rows=32956 width=15) (actual time=7.538..11.130 rows=32956 loops=1)
   Group Key: id
   Batches: 1  Memory Usage: 3345kB
   ->  Seq Scan on tickets  (cost=0.00..508.56 rows=32956 width=15) (actual time=0.004..1.668 rows=32956 loops=1)
 Planning Time: 0.060 ms
 Execution Time: 11.961 ms
(6 rows)
```

Indexing to improve sort (ORDER BY) performance



```
dbt=# EXPLAIN ANALYZE SELECT * FROM tickets ORDER BY cust_name;
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Index Scan using idx_cust on tickets  (cost=0.29..1726.61 rows=32956 width=15) (actual time=0.019..34.699 rows=32956 loops=1)
 Planning Time: 0.202 ms
 Execution Time: 37.932 ms
(3 rows)

dbt=# create index temp on tickets(cust_name);
CREATE INDEX
dbt=# EXPLAIN ANALYZE SELECT * FROM tickets ORDER BY cust_name;
                                                        QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------
 Index Scan using temp on tickets  (cost=0.29..1726.61 rows=32956 width=15) (actual time=0.032..8.883 rows=32956 loops=1)
 Planning Time: 0.095 ms
 Execution Time: 9.633 ms
(3 rows)
```