Assignment A2

Create optimum no of indexes on different tables of the selected mini-world database. After creating indexes run Explain Plans and compare it with previous Explain Plans. Demonstrate the improvement in Cost of Query.

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

Running a random select query:

```
dbt=# EXPLAIN ANALYZE SELECT COUNT(*) FROM tickets where cust_name like '123%';
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=590.96..590.97 rows=1 width=8) (actual time=9.610..9.611 rows=1 loops=1)
   ->  Seq Scan on tickets  (cost=0.00..590.95 rows=3 width=0) (actual time=2.508..9.597 rows=9 loops=1)
         Filter: ((cust_name)::text ~~ '123%'::text)
         Rows Removed by Filter: 32947
 Planning Time: 0.178 ms
 Execution Time: 9.654 ms
(6 rows)
```

Creating an index on it:

```
dbt=# CREATE INDEX idx_cust ON tickets(cust_name);
CREATE INDEX
```

Rerunning the same query:

```
dbt=# EXPLAIN ANALYZE SELECT COUNT(*) FROM tickets where cust_name like '123%';
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=590.96..590.97 rows=1 width=8) (actual time=5.772..5.772 rows=1 loops=1)
   ->  Seq Scan on tickets  (cost=0.00..590.95 rows=3 width=0) (actual time=2.510..5.764 rows=9 loops=1)
         Filter: ((cust_name)::text ~~ '123%'::text)
         Rows Removed by Filter: 32947
 Planning Time: 0.163 ms
 Execution Time: 5.806 ms
(6 rows)
```

We can see that the execution time reduced significantly.

Similarly, following the same process:

```
dbt=# EXPLAIN ANALYZE SELECT COUNT(*) FROM tickets where id between 25000 and 100000;
                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=735.21..735.22 rows=1 width=8) (actual time=15.031..15.033 rows=1 loops=1)
   ->  Seq Scan on tickets  (cost=0.00..673.34 rows=24748 width=0) (actual time=0.017..10.870 rows=24744 loops=1)
         Filter: ((id >= 25000) AND (id <= 100000))
         Rows Removed by Filter: 8212
 Planning Time: 0.321 ms
 Execution Time: 15.077 ms
(6 rows)
```

```
dbt=# CREATE INDEX idx_tickets_id ON tickets(id);
CREATE INDEX
```

```
dbt=# EXPLAIN ANALYZE SELECT COUNT(*) FROM tickets where id between 25000 and 100000;
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=735.21..735.22 rows=1 width=8) (actual time=6.866..6.867 rows=1 loops=1)
   ->  Seq Scan on tickets  (cost=0.00..673.34 rows=24748 width=0) (actual time=0.017..4.933 rows=24744 loops=1)
         Filter: ((id >= 25000) AND (id <= 100000))
         Rows Removed by Filter: 8212
 Planning Time: 0.217 ms
 Execution Time: 6.901 ms
(6 rows)
```

Even for a simple single element search query, the performance improvement is significant

```
dbt=# drop index idx_tickets_id;
DROP INDEX
dbt=# EXPLAIN ANALYZE SELECT COUNT(*) FROM tickets where id = 23092;
CREATE INDEX idx_tickets_id ON tickets(id);
EXPLAIN ANALYZE SELECT COUNT(*) FROM tickets where id = 23092;
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4.31..4.32 rows=1 width=8) (actual time=0.052..0.053 rows=1 loops=1)
   ->  Index Only Scan using tickets_pkey on tickets  (cost=0.29..4.31 rows=1 width=0) (actual time=0.039..0.042 rows=1 loops=1)
         Index Cond: (id = 23092)
         Heap Fetches: 0
 Planning Time: 0.224 ms
 Execution Time: 0.098 ms
(6 rows)

CREATE INDEX
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=4.31..4.32 rows=1 width=8) (actual time=0.020..0.020 rows=1 loops=1)
   ->  Index Only Scan using idx_tickets_id on tickets  (cost=0.29..4.31 rows=1 width=0) (actual time=0.015..0.016 rows=1 loops=1)
         Index Cond: (id = 23092)
         Heap Fetches: 0
 Planning Time: 0.110 ms
 Execution Time: 0.034 ms
(6 rows)
```