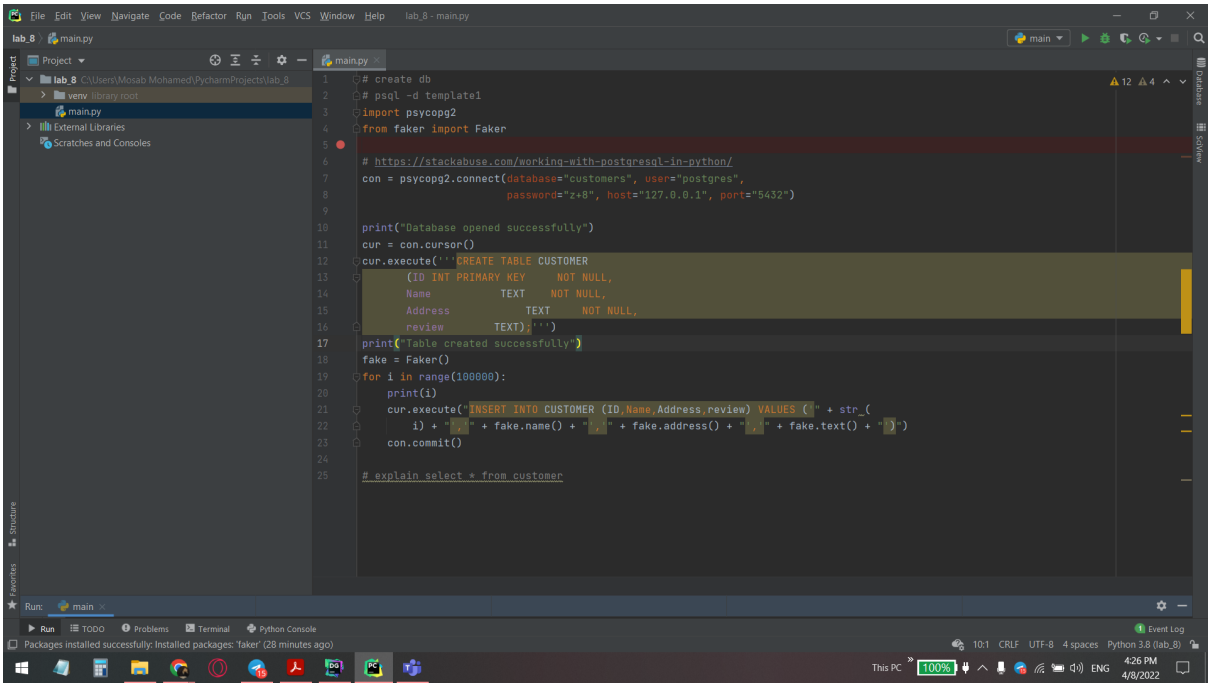
Mosab Mohamed - B20-04(Online)

Exercise 1:

1.1)

I used the following script to fill the table :



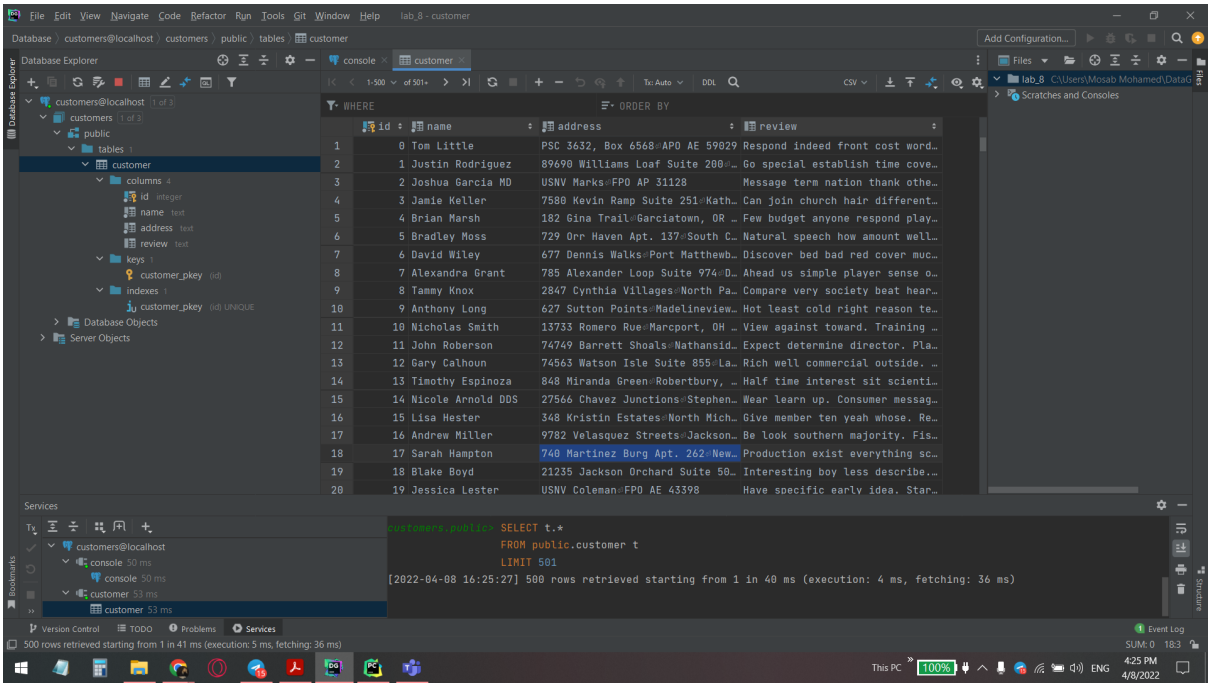
```
# create db
# psql -d template1
import psycopg2
from faker import Faker

# https://stackabuse.com/working-with-postgresql-in-python/
con = psycopg2.connect(database="customers", user="postgres",
                       password="z+8", host="127.0.0.1", port="5432")

print("Database opened successfully")
cur = con.cursor()
cur.execute('''CREATE TABLE CUSTOMER
      (ID INT PRIMARY KEY     NOT NULL,
      Name           TEXT    NOT NULL,
      Address            TEXT     NOT NULL,
      review         TEXT); ''')
print("Table created successfully")
fake = Faker()
for i in range(100000):
    print(i)
    cur.execute("INSERT INTO CUSTOMER (ID,Name,Address,review) VALUES (" + str_(
        i) + "," + fake.name() + "," + fake.address() + "," + fake.text() + ")")
    con.commit()

# explain select * from customer
```

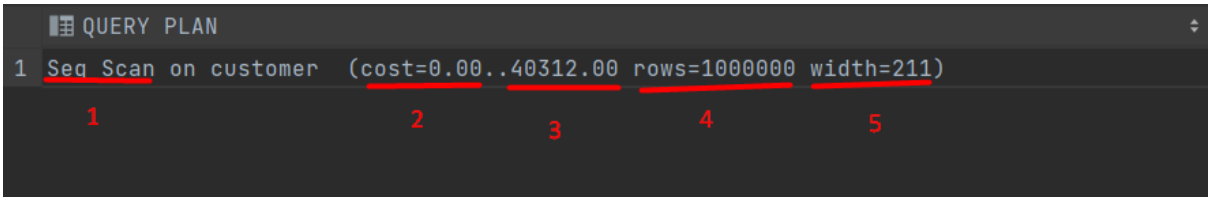
And here is the result :



1.2)

To fetch the data we use

```
EXPLAIN SELECT * FROM customer ;
```



1: Type of scan nodes
2: Estimated start-up time
3: Estimated total cost
4: Estimated number of rows
5: Estimated number of columns

1.3)

```
EXPLAIN SELECT id FROM customer WHERE name = 'Kyle Lee';
```

```
QUERY PLAN
1 Gather  (cost=1000.00..36528.45 rows=10 width=4)
2   Workers Planned: 2
3   ->  Parallel Seq Scan on customer  (cost=0.00..35527.45 rows=4 width=4)
4         Filter: (name = 'Kyle Lee'::text)
```

```
EXPLAIN SELECT id FROM customer WHERE address LIKE '1%';
```

```
QUERY PLAN
1 Seq Scan on customer  (cost=0.00..42819.28 rows=90911 width=4)
2   Filter: (address ~~ '1%'::text)
```

```
EXPLAIN SELECT review FROM customer WHERE review LIKE '%' || 'yes' || '%';
```

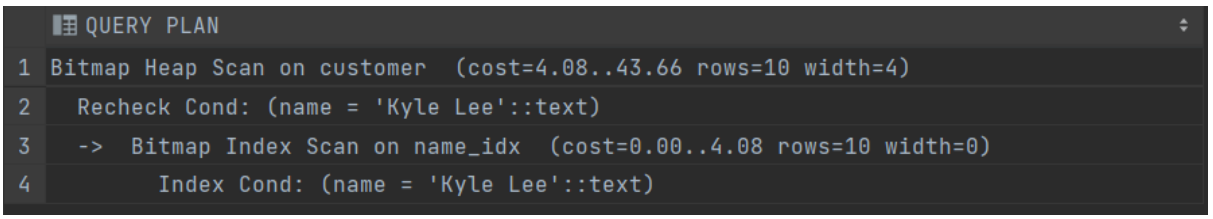
```
QUERY PLAN
1 Gather  (cost=1000.00..41578.05 rows=50506 width=149)
2   Workers Planned: 2
3   ->  Parallel Seq Scan on customer  (cost=0.00..35527.45 rows=21044 width=149)
4         Filter: (review ~~ '%yes%'::text)
```

1.4)

```
CREATE INDEX address_idx ON customer USING hash (address);
CREATE INDEX name_idx ON customer USING btree (name);
CREATE INDEX id_idx ON customer USING hash (id);
CREATE INDEX review_idx ON customer USING hash (review);
```

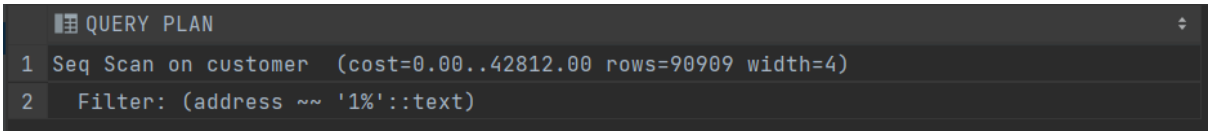
1.5)

```
EXPLAIN SELECT id FROM customer WHERE name = 'Kyle Lee';
```



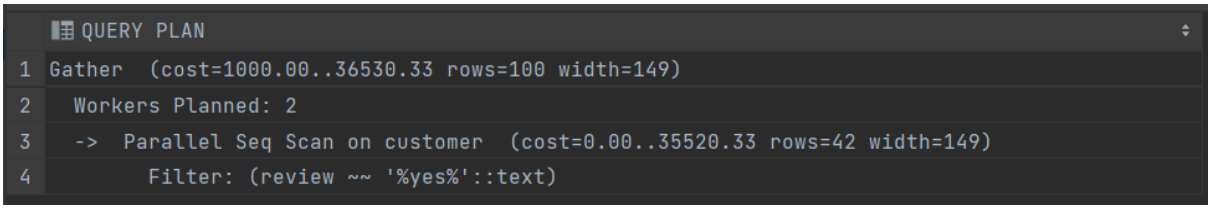
```
QUERY PLAN
1 Bitmap Heap Scan on customer  (cost=4.08..43.66 rows=10 width=4)
2   Recheck Cond: (name = 'Kyle Lee'::text)
3   ->  Bitmap Index Scan on name_idx  (cost=0.00..4.08 rows=10 width=0)
4         Index Cond: (name = 'Kyle Lee'::text)
```

```
EXPLAIN SELECT id FROM customer WHERE address LIKE '1%';
```



```
QUERY PLAN
1 Seq Scan on customer  (cost=0.00..42812.00 rows=90909 width=4)
2   Filter: (address ~~ '1%'::text)
```

```
EXPLAIN SELECT review FROM customer WHERE review LIKE '%' || 'yes' || '%';
```



```
QUERY PLAN
1 Gather  (cost=1000.00..36530.33 rows=100 width=149)
2   Workers Planned: 2
3   ->  Parallel Seq Scan on customer  (cost=0.00..35520.33 rows=42 width=149)
4         Filter: (review ~~ '%yes%'::text)
```

1.6)

You can already see that i did run the script for 1 million

Conclusion:

You can see that the estimated total cost decreases after making the Indexes