

**Mosab Mohamed - B20-04(Online)**

### Exercise 1:

1.1)

I used the following script to fill the table :

The screenshot shows a PyCharm IDE with a Python script named 'main.py' in the 'lab\_8' project. The script is as follows:

```

1 # Create db
2 # psql -d template1
3 import psycopg2
4 from faker import Faker
5
6 # https://stackabuse.com/working-with-postgresql-in-python/
7 con = psycopg2.connect(database="customers", user="postgres",
8                        password="z48", host="127.0.0.1", port="5432")
9
10 print("Database opened successfully")
11 cur = con.cursor()
12 cur.execute("""CREATE TABLE CUSTOMER
13             (ID INT PRIMARY KEY     NOT NULL,
14              Name           TEXT     NOT NULL,
15              Address        TEXT     NOT NULL,
16              review         TEXT);""")
17 print("Table created successfully")
18 fake = Faker()
19 for i in range(100000):
20     print(i)
21     cur.execute("INSERT INTO CUSTOMER (ID,Name,Address,review) VALUES (" + str(
22         i) + "," + fake.name() + "," + fake.address() + "," + fake.text() + ")")
23     con.commit()
24
25 # explain select * from customer

```

The bottom status bar shows 'Python 3.8 (lab\_8)' and '4:26 PM 4/8/2022'.

And here is the result :

The screenshot shows a web-based database management tool interface. On the left is a 'Database Explorer' pane showing a tree structure of a database named 'customers@localhost'. The tree includes 'customers' (1 of 3), 'public' (1 of 3), 'tables' (1), 'customer' (1), 'columns' (4), 'keys' (1), 'indexes' (1), and 'Database Objects'. The 'customer' table is selected, showing its columns: 'id' (integer), 'name' (text), 'address' (text), and 'review' (text). Below the tree is a 'Services' pane showing a list of services, including 'customers@localhost', 'console 50 ms', 'console 53 ms', and 'customer 53 ms'. The main area is a SQL editor with a query: 

```
SELECT t.*
FROM public.customer t
LIMIT 501
```

 Below the query, a status message reads: '[2022-04-08 16:25:27] 500 rows retrieved starting from 1 in 40 ms (execution: 4 ms, fetching: 36 ms)'. On the right is a table of results with 19 rows and 4 columns: 'id', 'name', 'address', and 'review'. The table contains data for customers 0 through 18. The status bar at the bottom indicates '500 rows retrieved starting from 1 in 41 ms (execution: 5 ms, fetching: 36 ms)'.

id	name	address	review
0	Tom Little	PSC 3632, Box 6568-APO AE 59029	Respond indeed front cost word...
1	Justin Rodriguez	89690 Williams Loaf Suite 200...	Go special establish time cov...
2	Joshua Garcia ND	USNV Marks-FPO AP 31128	Message term nation thank othe...
3	Jamie Keller	7580 Kevin Ramp Suite 251-Kath...	Can join church hair different...
4	Brian Marsh	182 Gina Trail-Garciatown, OR	Few budget anyone respond play...
5	Bradley Moss	729 Orr Haven Apt. 137-South C...	Natural speech how amount well...
6	David Wiley	677 Dennis Walks-Port Matthewb...	Discover bed bad red cover muc...
7	Alexandra Grant	785 Alexander Loop Suite 974-D...	Ahead us simple player sense o...
8	Tammy Knox	2847 Cynthia Villages-North Pa...	Compare very society beat hear...
9	Anthony Long	627 Sutton Points-Madelineview...	Hot least cold right reason te...
10	Nicholas Smith	13733 Romero Rue-Marcport, OH	View against toward. Training...
11	John Roberson	74749 Barrett Shoals-Nathansid...	Expect determine director. Pla...
12	Gary Calhoun	74563 Watson Isle Suite 855-La...	Rich well commercial outside...
13	Timothy Espinoza	848 Miranda Green-Robertbury...	Half time interest sit scienti...
14	Nicole Arnold DDS	27566 Chavez Junctions-Stephan...	Wear learn up. Consumer messag...
15	Lisa Hester	348 Kristin Estates-North Mich...	Give member ten yeah whose. Re...
16	Andrew Miller	9782 Velasquez Streets-Jackson...	Be look southern majority. Fis...
17	Sarah Hampton	740 Martinez Burg Apt. 262-New...	Production exist everything sc...
18	Blake Boyd	21235 Jackson Orchard Suite 50...	Interesting boy never describe...
19	Jessica Lester	USNV Coleman-FPO AE 43398	Have specific early idea. Stan...

1.2)

To fetch the data we use

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

QUERY PLAN				
1	Seq Scan on customer	(cost=0.00..40312.00	rows=1000000	width=211)
1	2	3	4	5

- 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