Assignment 3:

In this assignment we imported the database and performed indexing on the data.

Firstly, we find a large dataset so that we can see the difference in the execution time.

For importing a .sql file we will first create a database and then import the file using the query below:

>>>mysql -u username -p password database_name < file_name.sql

Query used in the assignment:

>>>mysql -u root spin < load_salaries1.sql (dataset used for Clustering)

>>>mysql -u root sakila < data_sakila.sql (dataset used for Primary Indexing)

Types of indexing:

1. Single level indexing
    - Primary Indexing
    - Clustering
    - Secondary Indexing
2. Multi level indexing

For checking the number of records in a table we used the COUNT function.

- Clustering: When we apply indexing on a non-prime attribute it is called clustering.
  Checking for the number of records in the table. This is done by using the “count “ function.

  Query is :
  >>select count(*) from salaries;

```
MariaDB [spin]> select count(*) from salaries;
+----------+
| count(*) |
+----------+
|   967330 |
+----------+
1 row in set (2.001 sec)
```

1. Retrieving the salaries whose id is equal to 90000 with and without indexing

   Without Indexing Execution time is 1.881s

```
MariaDB [spin]> select sal from salaries where eid = 90000;
+-------+
| sal   |
+-------+
| 40000 |
| 42979 |
| 43840 |
| 44422 |
| 45251 |
| 48836 |
| 50857 |
| 54562 |
| 58477 |
+-------+
9 rows in set (1.881 sec)
```

   If we index the eid attribute which is non-prime the new execution time Is 0.158s. This happens because the pointer will not search in the entire column for these eids sequentially but directly point the eid saving time

```
MariaDB [spin]> select sal from salaries where eid = 90000;
+-------+
| sal   |
+-------+
| 40000 |
| 42979 |
| 43840 |
| 44422 |
| 45251 |
| 48836 |
| 50857 |
| 54562 |
| 58477 |
+-------+
9 rows in set (0.158 sec)
```

2. Retrieving the eids from the table whose salary is equal to 100000 with and without indexing
   In this question we shall index the salary attribute and then execute the queries

   Query for indexing the ‘sal ‘ attribute:

   >>Create index I on salaries(sal);

With indexing the execution time is 0.066s

Query is:
>>select eid from salaries where sal = 100000;

Without Indexing the execution time is2.218s

Query is:
>>select eid from salaries where sal = 100000;

```
MariaDB [spin]> select eid from salaries where sal = 100000;
+-------+
| eid   |
+-------+
| 14321 |
| 21546 |
+-------+
2 rows in set (0.066 sec)

MariaDB [spin]> drop index i on salaries;
Query OK, 0 rows affected (0.392 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [spin]> select eid from salaries where sal = 100000;
+-------+
| eid   |
+-------+
| 14321 |
| 21546 |
+-------+
2 rows in set (2.218 sec)
```

3. Retrieving eids and salaries from the table by applying a self join with and without creating an index
   In this question we declare index to two attributes ‘ID’ and ‘sal’

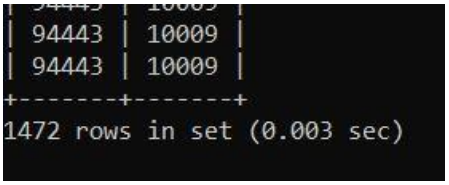
   Query for creating index to two colums:
   >>create index I on salaries(ID, sal);

   Query for dropping the index:
   >>drop index i on salaries;

   Since we have a single table we shall self join to check for the execution times
   Here we shall select the eids and salaries of those employees whose eid is less than 10010.

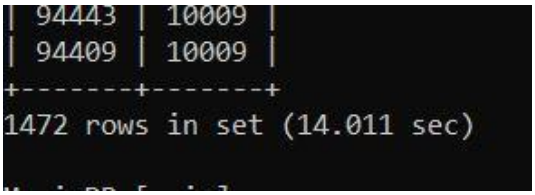
With indexing the execution time is 0.003s

Query:
>>select e.sal,e.eid from salaries e, salaries f where e.eid=f.eid and e.eid < 10010;
The execution time :



Without Indexing the execution time is 14.011s

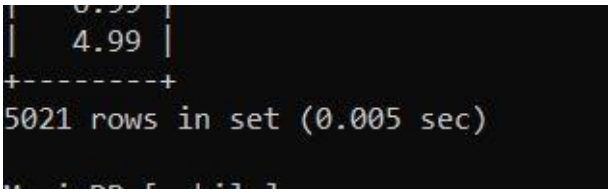
Query:
>>select e.sal,e.eid from salaries e, salaries f where e.eid=f.eid and e.eid < 10010;
with index - 14.011s
The execution time:



- Primary Indexing: When we apply indexing on a key in a dataset which is primary that is unique and not null, it is called Primary indexing.

4. Retrieve the amount where the payment ID is more than 5000 with and without indexing.
   In this we index the 'payment_id' attribute which is the primary key

   With Indexing execution time 0.005s
   Query:
   >> select amount from payment where payment_id > 50000;
   The execution time:



Without Indexing execution time is 0.036s
Query:
>> select amount from payment where payment_id > 50000;
The execution time:

```
|   4.99 |
+--------+
5021 rows in set (0.036 sec)
```

5. Retrieve the payment and rental id from the table where rental is null with and without indexing.
   This is an example of secondary indexing since here we are applying indexing on both 'payment_id' and 'rental_id', that is a mixture of primary and clustering. In this we will self join

   Without Indexing execution time is 0.012s

   Query:
   >> select e.payment_id,e.rental_id from payment e,payment f where e.payment_id=f.payment_id and e.rental_id is null;

```
MariaDB [sakila]> select e.payment_id,e.rental_id from payment e,payment f where e
.payment_id=f.payment_id
    -> and e.rental_id is null;
+------------+-----------+
| payment_id | rental_id |
+------------+-----------+
|        424 |      NULL |
|       7011 |      NULL |
+------------+-----------+
2 rows in set (0.012 sec)
```

   With Indexing execution time is 0.007s

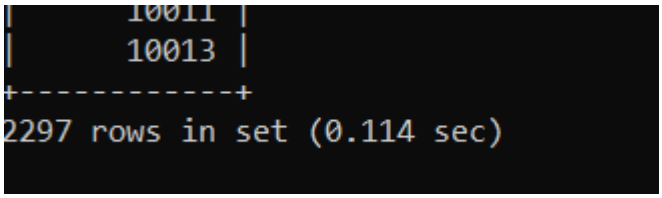
   Query:
   >> select e.payment_id,e.rental_id from payment e,payment f where e.payment_id=f.payment_id and e.rental_id is null;

```
MariaDB [sakila]> select e.payment_id,e.rental_id from payment e,payment f where e
.payment_id=f.payment_id
    -> and e.rental_id is null;
+------------+-----------+
| payment_id | rental_id |
+------------+-----------+
|        424 |      NULL |
|       7011 |      NULL |
+------------+-----------+
2 rows in set (0.007 sec)
```

6. Select the payment ID where amount is in the range of 0 to 2.2 with and without indexing.
   Here we will index the non primary attribute which is amount (clustering)

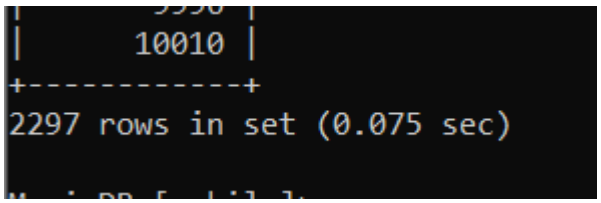
   Without indexing execution time 0.114s

   Query:
   >> select payment_id from payment where amount between 0.0 and 2.2;
   Execution time:



   With Indexing execution time is 0.075s

   Query:
   >> select payment_id from payment where amount between 0.0 and 2.2;
   Execution time:



7. Retrieve average amount of the employees with payment id 1000 with and without indexing

```
MariaDB [sakila]> select avg(amount) from payment where payment_id = 10000;
+-------------+
| avg(amount) |
+-------------+
|    6.990000 |
+-------------+
1 row in set (0.100 sec)

MariaDB [sakila]> drop index i on payment;
Query OK, 0 rows affected (0.443 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [sakila]> select avg(amount) from payment where payment_id = 10000;
+-------------+
| avg(amount) |
+-------------+
|    6.990000 |
+-------------+
1 row in set (0.001 sec)
```

In this question we indexed the attribute ‘payment_id’. The execution time with indexing is more than the execution time without indexing.

Table:

Query Number	Execution time (with indexing)	Execution time (without indexing)
1	0.158s	1.881s
2	0.066s	2.218s
3	0.003s	14.011s
4	0.005s	0.036s
5	0.007s	0.012s
6	0.075s	0.114s
7	0.100s	0.001s

Conclusion: In this assignment we studied that in most of the cases the execution time for indexing is less than that without indexing.