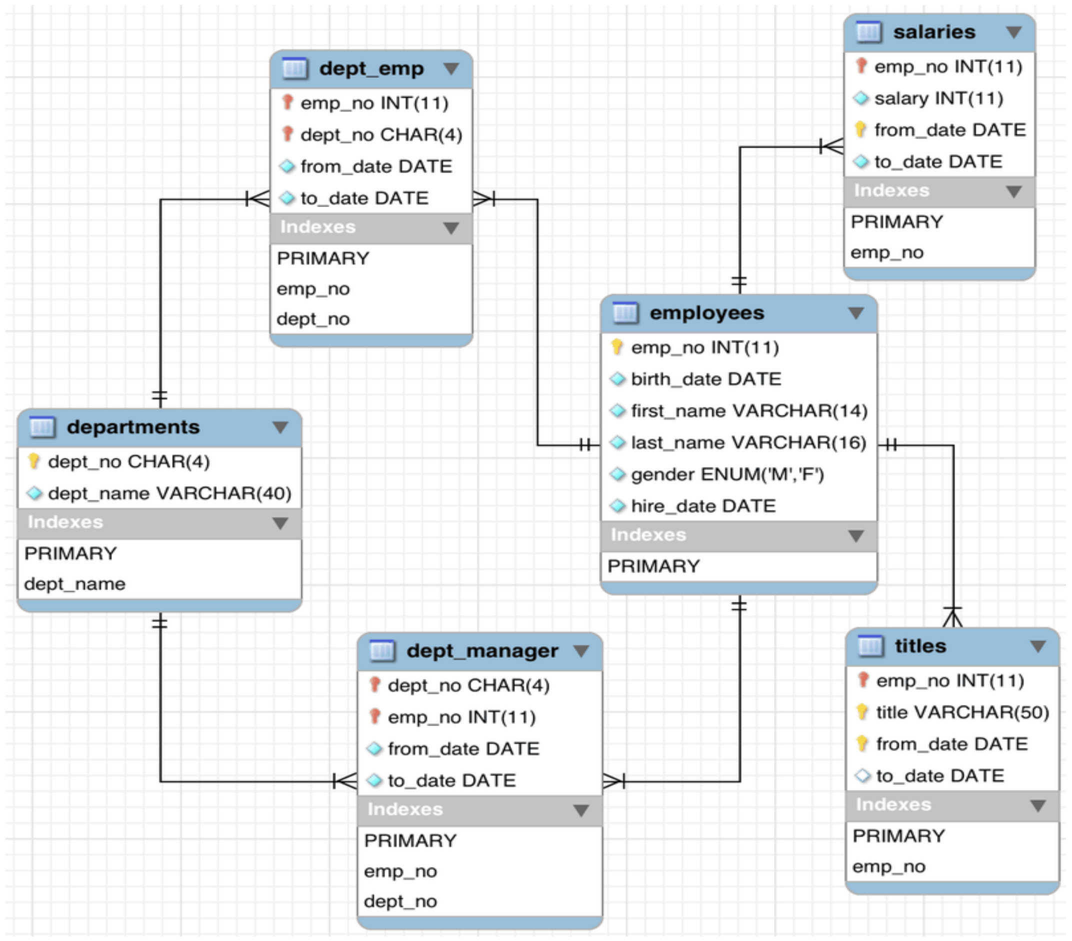
COMP90050: Advanced Database Systems
Semester 2, 2024
Lab 1: Query Execution Plan and Indexing

Database Schema:



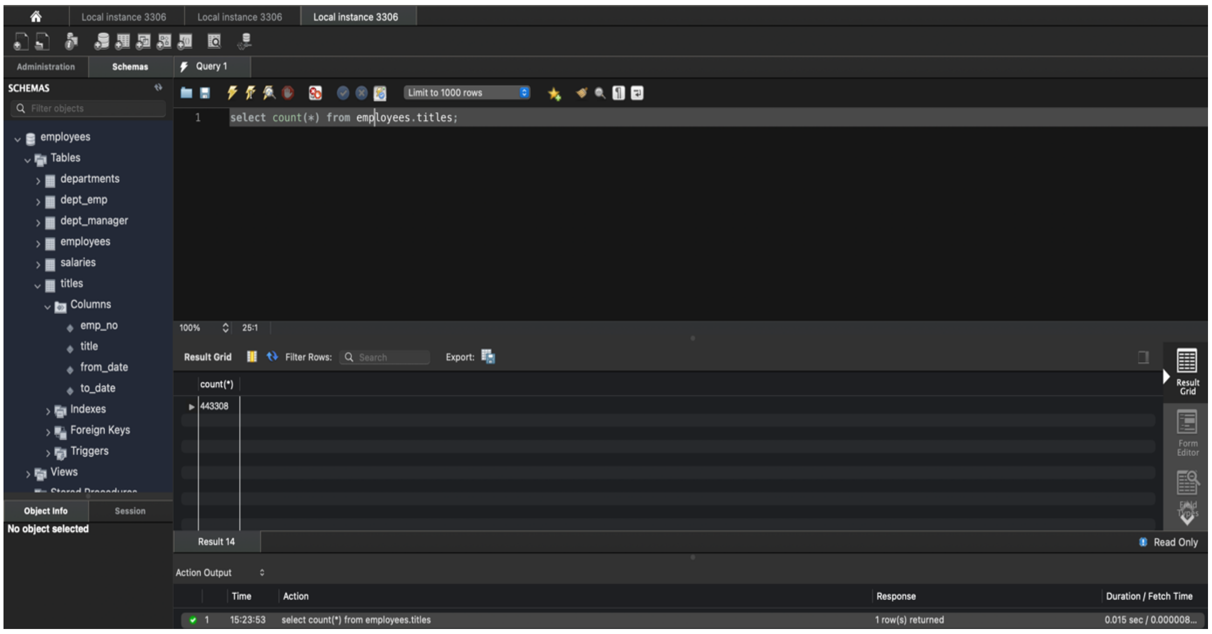
As part of this exercise, we will use the same schema as the one used in the setup instructions.

Query Execution Plan:

When we run queries in MySQL, the MySQL optimizer automatically runs in the background. Its role is to design an optimal MySQL query execution plan for every single query that is executed. We can view the execution plan(s) by using the EXPLAIN keyword (https://dev.mysql.com/doc/refman/8.0/en/explain.html) as a prefix in our queries.

For instance, in the Schema diagram above there is a “titles” table with a column called “title”. Let’s view the total number of columns as well as unique titles that are available for employees in this database.

We can retrieve the total number of rows in the title table using the following query:

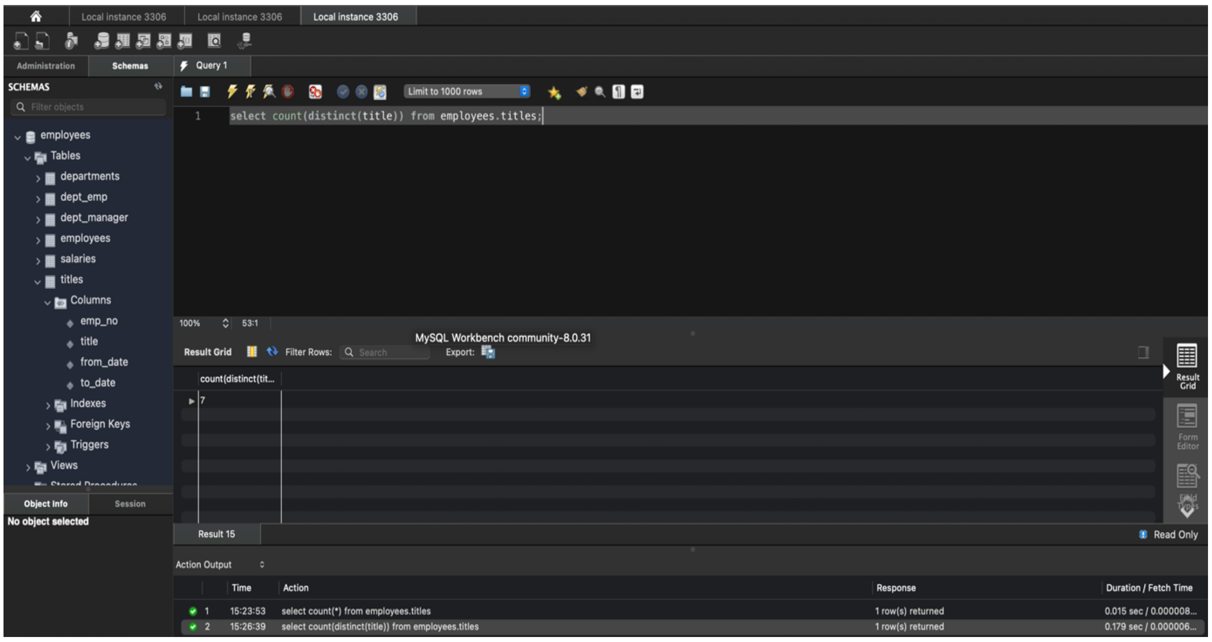


```
select count(*) from employees.titles;
```

```
count(*)
443308
```

		Time	Action	Response	Duration / Fetch Time
✓	1	15:23:53	select count(*) from employees.titles	1 row(s) returned	0.015 sec / 0.000008...

To retrieve the number of unique titles, we need to use the "distinct" keyword as follows:

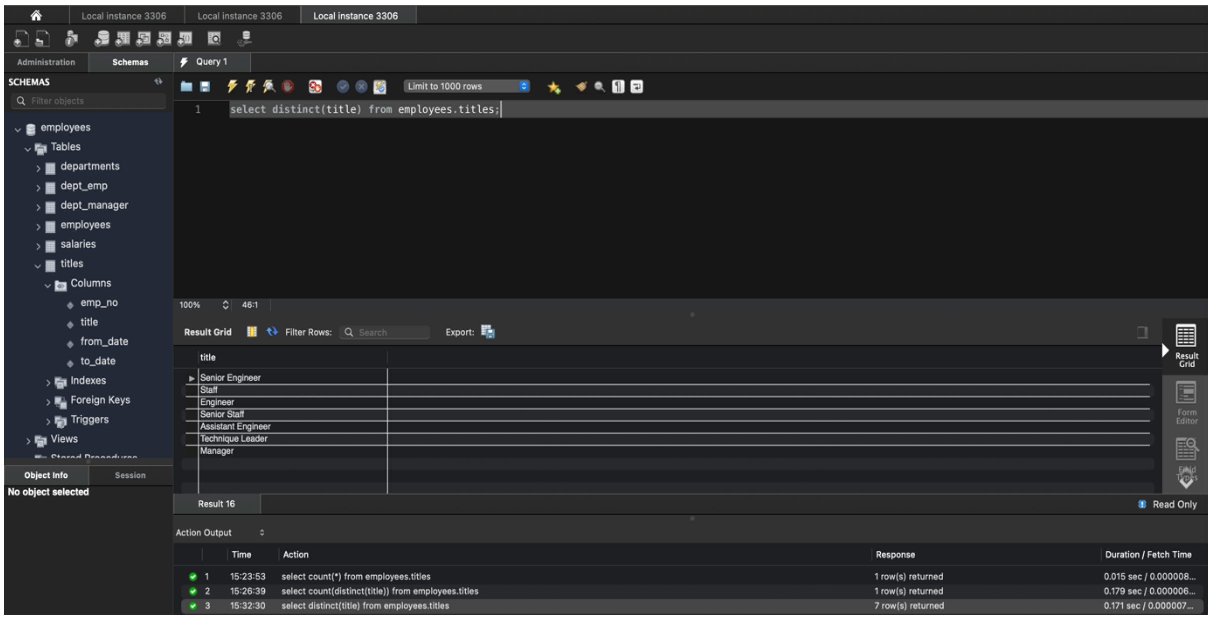


```
select count(distinct(title)) from employees.titles;
```

```
count(distinct(tit...
7
```

		Time	Action	Response	Duration / Fetch Time
✓	1	15:23:53	select count(*) from employees.titles	1 row(s) returned	0.015 sec / 0.000008...
✓	2	15:26:39	select count(distinct(title)) from employees.titles	1 row(s) returned	0.179 sec / 0.000006...

We can see that there are 7 unique titles available. If we would like to see what those titles are then we can run the following query:

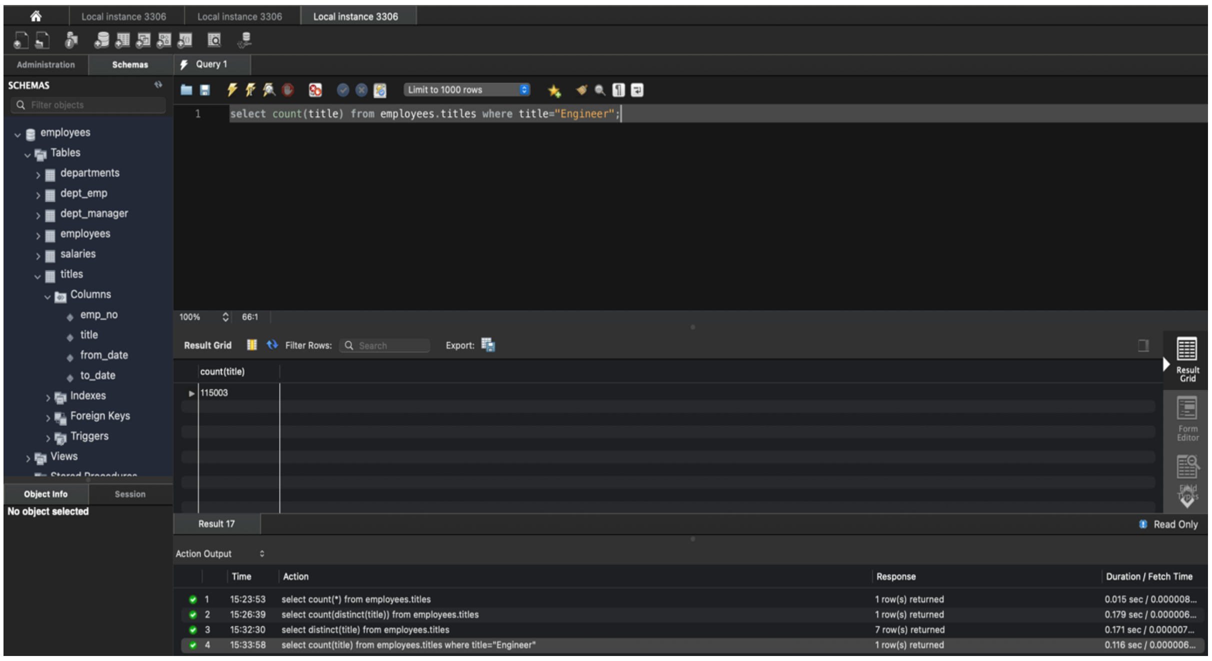


```
select distinct(title) from employees.titles;
```

```
title
Senior Engineer
Staff
Engineer
Senior Staff
Assistant Engineer
Technique Leader
Manager
```

```
    Time      Action                                              Response          Duration / Fetch Time
1   15:23:53  select count(*) from employees.titles               1 row(s) returned   0.015 sec / 0.000008...
2   15:26:39  select count(distinct(title)) from employees.titles 1 row(s) returned   0.179 sec / 0.000006...
3   15:32:30  select distinct(title) from employees.titles        7 row(s) returned   0.171 sec / 0.000007...
```

From these titles, we can try to query how many employees are there per title. In our case, we are going to retrieve the total number of employees with the title “Engineer” as follows:

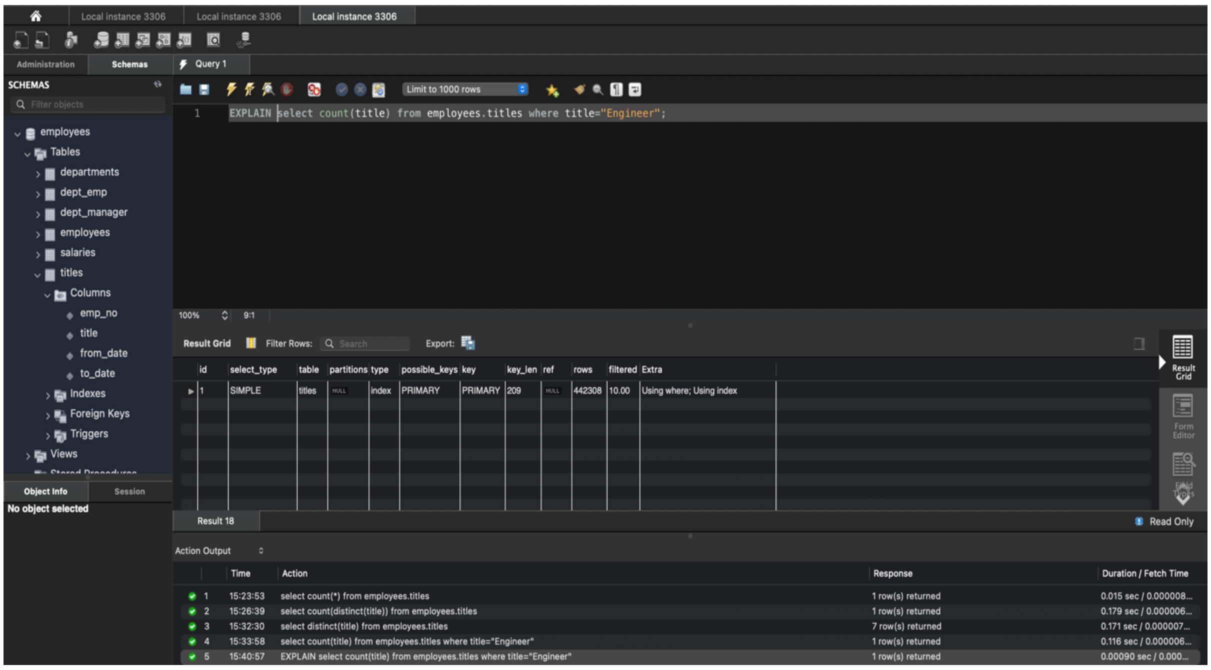


```
select count(title) from employees.titles where title="Engineer";
```

```
count(title)
115003
```

```
    Time      Action                                                          Response          Duration / Fetch Time
1   15:23:53  select count(*) from employees.titles                           1 row(s) returned   0.015 sec / 0.000008...
2   15:26:39  select count(distinct(title)) from employees.titles             1 row(s) returned   0.179 sec / 0.000006...
3   15:32:30  select distinct(title) from employees.titles                    7 row(s) returned   0.171 sec / 0.000007...
4   15:33:58  select count(title) from employees.titles where title="Engineer" 1 row(s) returned  0.116 sec / 0.000006...
```

We can observe that there are 115003 employees with the title of engineers and this query takes 0.116 seconds to execute (note that, these times may slightly vary depending on your computer). To view the breakdown of the query execution plan for this SELECT statement, we can use the “EXPLAIN” command as follows:



```
EXPLAIN select count(title) from employees.titles where title="Engineer";
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	titles	NULL	index	PRIMARY	PRIMARY	209	NULL	442308	10.00	Using where; Using index

Action Output

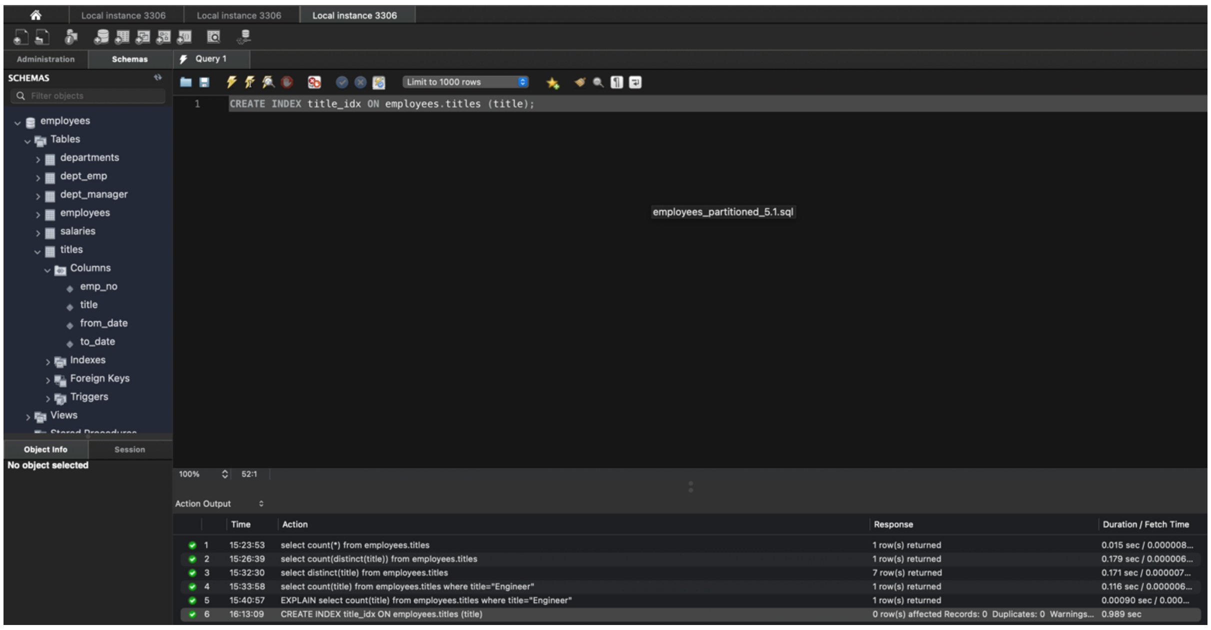
	Time	Action	Response	Duration / Fetch Time
1	15:23:53	select count(*) from employees.titles	1 row(s) returned	0.015 sec / 0.000008...
2	15:26:39	select count(distinct(title)) from employees.titles	1 row(s) returned	0.179 sec / 0.000006...
3	15:32:30	select distinct(title) from employees.titles	7 row(s) returned	0.171 sec / 0.000007...
4	15:33:58	select count(title) from employees.titles where title="Engineer"	1 row(s) returned	0.116 sec / 0.000006...
5	15:40:57	EXPLAIN select count(title) from employees.titles where title="Engineer"	1 row(s) returned	0.00090 sec / 0.000...

The EXPLAIN command provides further context around how MySQL executes the specific query. It returns a row of information detailing the tables being used, the amount of data being filtered by this query, extra information around how MySQL resolves the query and so on. A breakdown of what each column indicates as well as possible values can be found here: https://dev.mysql.com/doc/refman/8.0/en/explain-output.html

In this case, we can see that the filtered column indicates that 10% of the estimated table rows are filtered by this query. Similarly, the MySQL Execution plan is using the "where" clause and the current index (Primary Key – the *id* column) to execute the query. MySQL uses *BTree* index by default for PRIMARY.

Indexing:

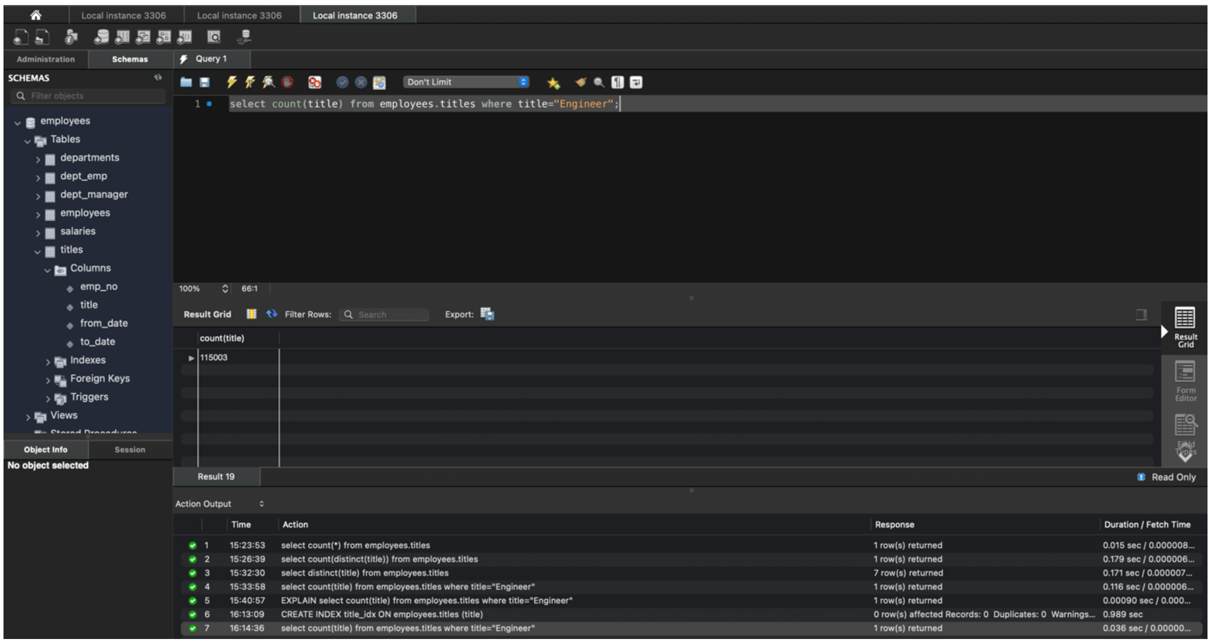
Given the current table size, the query is fast but as the number of rows continue to increase it would get slower. To reduce the execution time of the query we can perform a few optimisations, specifically – creating an index. Since we have 7 titles only and the titles are consistent across multiple rows it makes it a good candidate for creating an index on it:



```
CREATE INDEX title_idx ON employees.titles (title);
```

```
     Time      Action                                                          Response                                                  Duration / Fetch Time
1    15:23:53  select count(*) from employees.titles                           1 row(s) returned                                         0.015 sec / 0.000008...
2    15:26:39  select count(distinct(title)) from employees.titles             1 row(s) returned                                         0.179 sec / 0.000006...
3    15:32:30  select distinct(title) from employees.titles                    7 row(s) returned                                         0.171 sec / 0.000007...
4    15:33:58  select count(title) from employees.titles where title="Engineer"  1 row(s) returned                                       0.116 sec / 0.000006...
5    15:40:57  EXPLAIN select count(title) from employees.titles where title="Engineer"  1 row(s) returned                               0.00090 sec / 0.000...
6    16:13:09  CREATE INDEX title_idx ON employees.titles (title)              0 row(s) affected Records: 0  Duplicates: 0  Warnings...  0.989 sec
```

An index is an auxiliary structure used for optimisation in MySQL to increase the speed of query response time. With the newly created index on the “title” column, if we run the same SELECT query to find out the total number of employees who have the title “Engineer”, we get the following result:

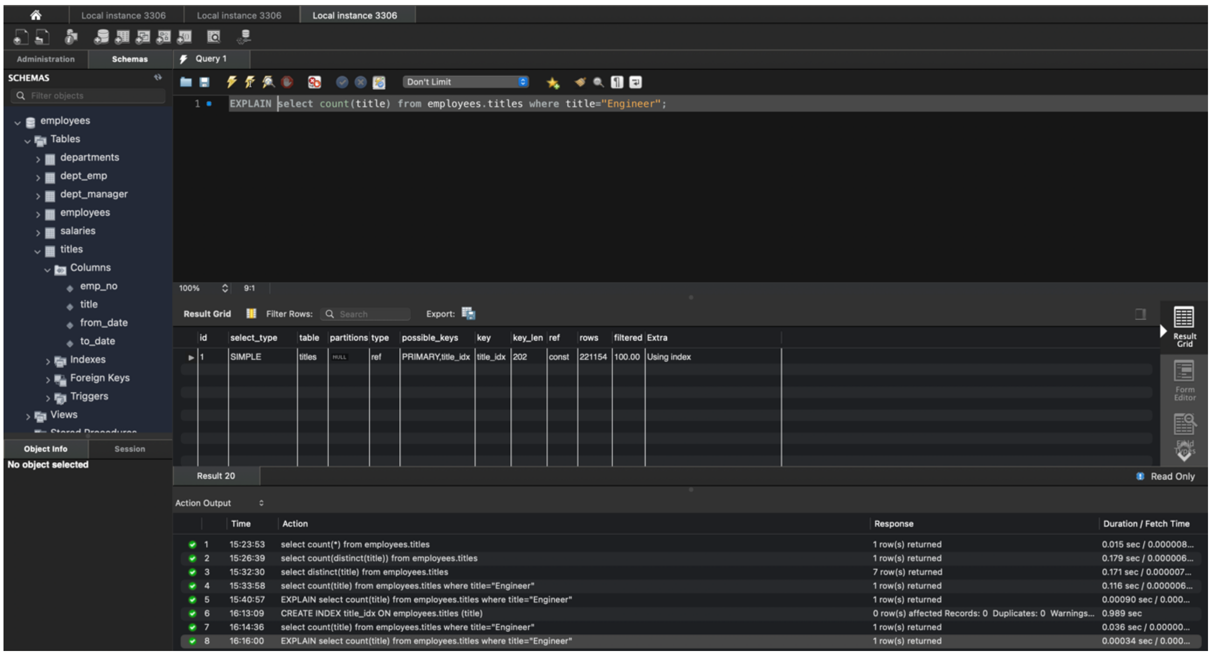


```
select count(title) from employees.titles where title="Engineer";
```

```
count(title)
115003
```

```
     Time      Action                                                          Response                                                  Duration / Fetch Time
1    15:23:53  select count(*) from employees.titles                           1 row(s) returned                                         0.015 sec / 0.000008...
2    15:26:39  select count(distinct(title)) from employees.titles             1 row(s) returned                                         0.179 sec / 0.000006...
3    15:32:30  select distinct(title) from employees.titles                    7 row(s) returned                                         0.171 sec / 0.000007...
4    15:33:58  select count(title) from employees.titles where title="Engineer"  1 row(s) returned                                       0.116 sec / 0.000006...
5    15:40:57  EXPLAIN select count(title) from employees.titles where title="Engineer"  1 row(s) returned                               0.00090 sec / 0.000...
6    16:13:09  CREATE INDEX title_idx ON employees.titles (title)              0 row(s) affected Records: 0  Duplicates: 0  Warnings...  0.989 sec
7    16:14:36  select count(title) from employees.titles where title="Engineer"  1 row(s) returned                                       0.036 sec / 0.00000...
```

We can observe that the time taken to run the same query reduced to 0.036 seconds (note that, these times may slightly vary depending on your computer), which is a lot less than the previous time of 0.116 seconds. If we use EXPLAIN on this SELECT query now, we can see the following changes:



```
EXPLAIN select count(title) from employees.titles where title="Engineer";
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	titles	NULL	ref	PRIMARY,title_idx	title_idx	202	const	221154	100.00	Using index

	Time	Action	Response	Duration / Fetch Time
1	15:23:53	select count(*) from employees.titles	1 row(s) returned	0.015 sec / 0.000008...
2	15:26:39	select count(distinct(title)) from employees.titles	1 row(s) returned	0.179 sec / 0.000006...
3	15:32:30	select distinct(title) from employees.titles	7 row(s) returned	0.171 sec / 0.000007...
4	15:33:58	select count(title) from employees.titles where title="Engineer"	1 row(s) returned	0.116 sec / 0.000006...
5	15:40:57	EXPLAIN select count(title) from employees.titles where title="Engineer"	1 row(s) returned	0.00090 sec / 0.000...
6	16:13:09	CREATE INDEX title_idx ON employees.titles (title)	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.989 sec
7	16:14:36	select count(title) from employees.titles where title="Engineer"	1 row(s) returned	0.036 sec / 0.00000...
8	16:16:00	EXPLAIN select count(title) from employees.titles where title="Engineer"	1 row(s) returned	0.00034 sec / 0.000...

We can observe major changes in the query execution plan in the filtered column as well as the new index we created is used as the primary mechanism for the query execution plan to execute this SELECT statement.

We may visually see the "Execution Plan". We can do this by running a Query and going to the "Execution Plan" tab on the bottom right scroll bar (where tabs like Result Grid are located). This presents a more visual way to capture some of the step-by-step costs associated when running queries under different types of optimizations and how/where a cost may crop in.

Questions:

1. How many rows are there for the other (unique) employee titles?
2. How long does it take for executing the SELECT queries for each employee title without the index on the "title" column of the "titles" table? You will need to remove the index on title column, which can be done by running the following query: DROP INDEX title_idx ON employees.titles;
3. How long does it take for executing the SELECT queries for each employee title with the index on the "title" column of the "titles" table?
4. What other columns in different tables of the "employees" schema can we create an index on?

Bonus Questions:

5. What are the disadvantages of creating indexes?
6. How can we identify how long does MySQL take for each part of the query? **<u>HINT:</u>** Look at the EXPLAIN ANALYZE command: https://dev.mysql.com/doc/refman/8.0/en/explain.html#explain-analyze

Tips/Guideline for the future: You can try creating indexes in other tables then run joins to see if creating different indexes makes any difference in the query performance.