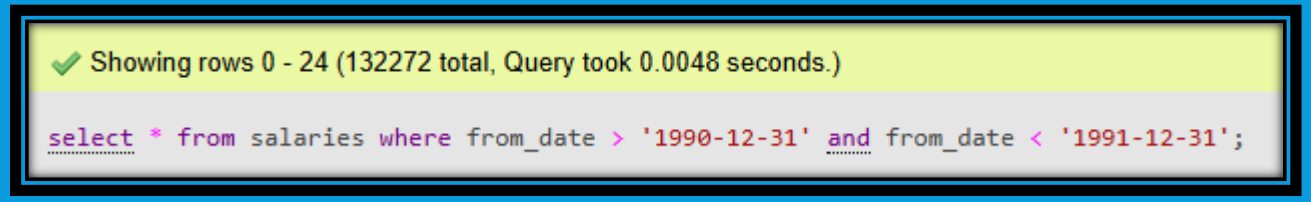
BIG DATA WITH MYSQL

Final project by:



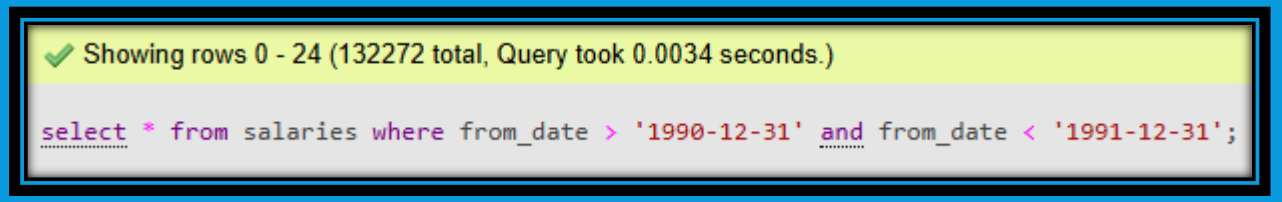
DATABASE WITH NO PARTITIONS

- Inside no_partitions.sql we have a script that will create a database called `project_no_partition` with all the tables as described in the project.jpg image.
- After that we can insert all the records from the provided `load_employees.sql, load_departments.sql, …` files, one by one, or use the `insert_all.sql` that I have created, which combines all the inserts into a single .sql file.
- Running the SELECT query on our data, we get this result:



DATABASE WITH PARTITIONS

- Inside with_partitions.sql we have a script that will create a database called `project_with_partition` with all the tables and the partitions for `salaries` and `titles`.
- After that we can insert all the records from the provided `load_employees.sql, load_departments.sql, …` files, one by one, or use the `insert_all.sql` again to insert everything at once.
- Running the SELECT query on our data, we get this result:

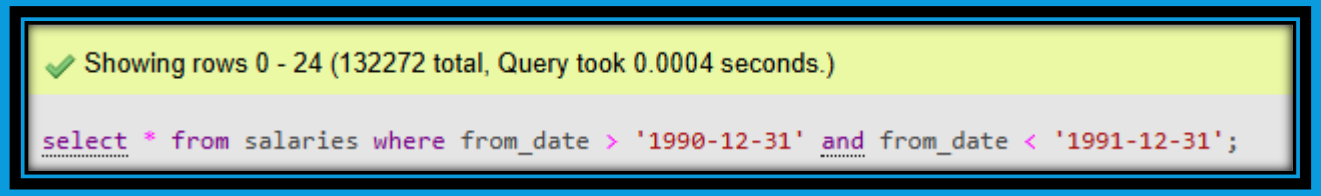


OTHER METHODS FOR OPTIMIZATION

- Adding an index is a simple way to optimize your query execution speed by a lot:

```
alter table salaries add index (from_date);
```

This single line took around 20 seconds to execute, but after the execution the select query was being executed 10 times faster:



- Another method for optimization is directly running the select on the given partition. If we have 20 partitions (1 for every year) from 2000 to 2020 year, and we want to access data from 2017, we know that it should be located on partition 17, selecting partition 17 directly would reduce the time it takes to find the correct partition.

THANKS FOR THE ATTENTION!

