MySQL Workbench Tutorial

1 CONTENTS

2 MYSQL WORKBENCH. INTRODUCTION

This section presents a tutorial for working with MySQL Workbench: connecting to the database server, working with schemas, tables and data with a visual interface/GUI.

2.1.1 Considering that you have just finished the installation, now the GUI will open (MySQL Workbench) with the configuration already set. (Figure 1)

This will be your visual tool to connect to the database server, used for working with databases and tables, viewing and adding data and writing SQL queries. Click on "Local Instance MySQL80" in the "MySQL Connections" list to open a connection to the database server.

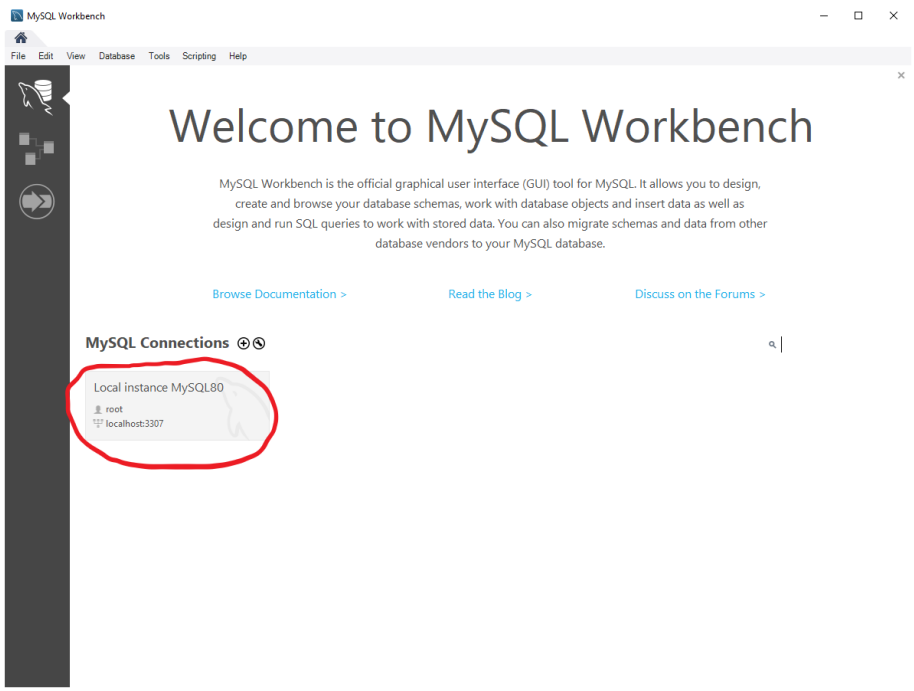

Figure 1 Welcome to MySQL Workbench

Note: if you don't see the connection in the list, then you might have skipped a step in the installation guide. It's no problem, you can create a new connection by clicking on the "+"

icon next to “MySQL Connections” as in (Figure 2). Then click “Ok” and move on to the next step.

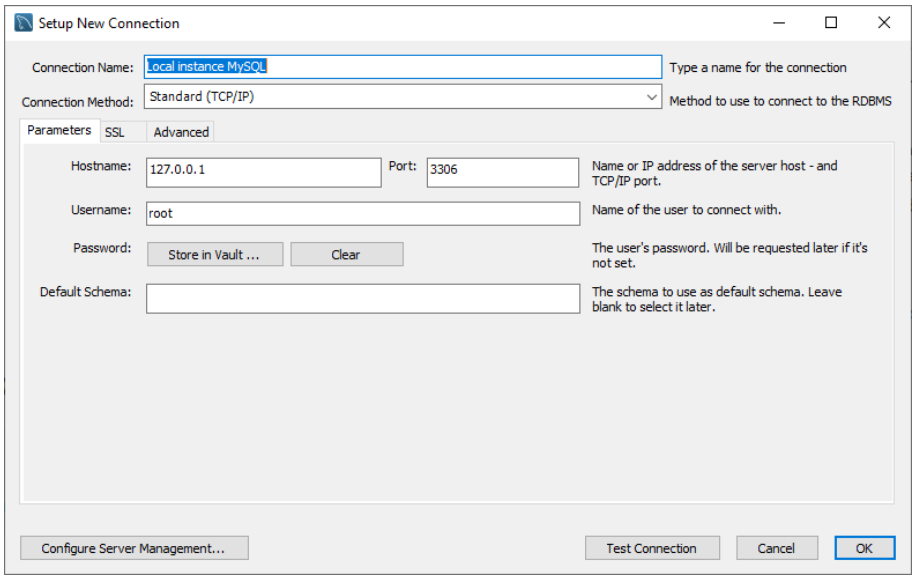

Figure 2 Setup New Connection

Connect to the MySQL server using the root password provided during the setup (e.g. **ewis2020**). (Figure 3)

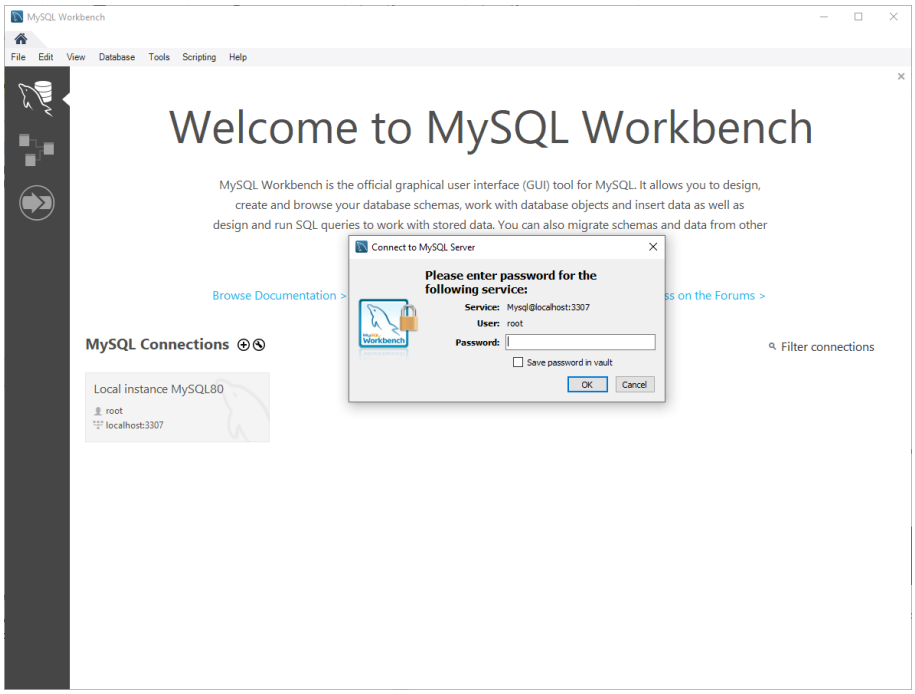

Figure 3 Connect using root password

2.1.2 Now you are connected to the MySQL database server and you can start creating a database. Select the “Schemas” tab to view a list of databases running on the local server. (Figure 4)

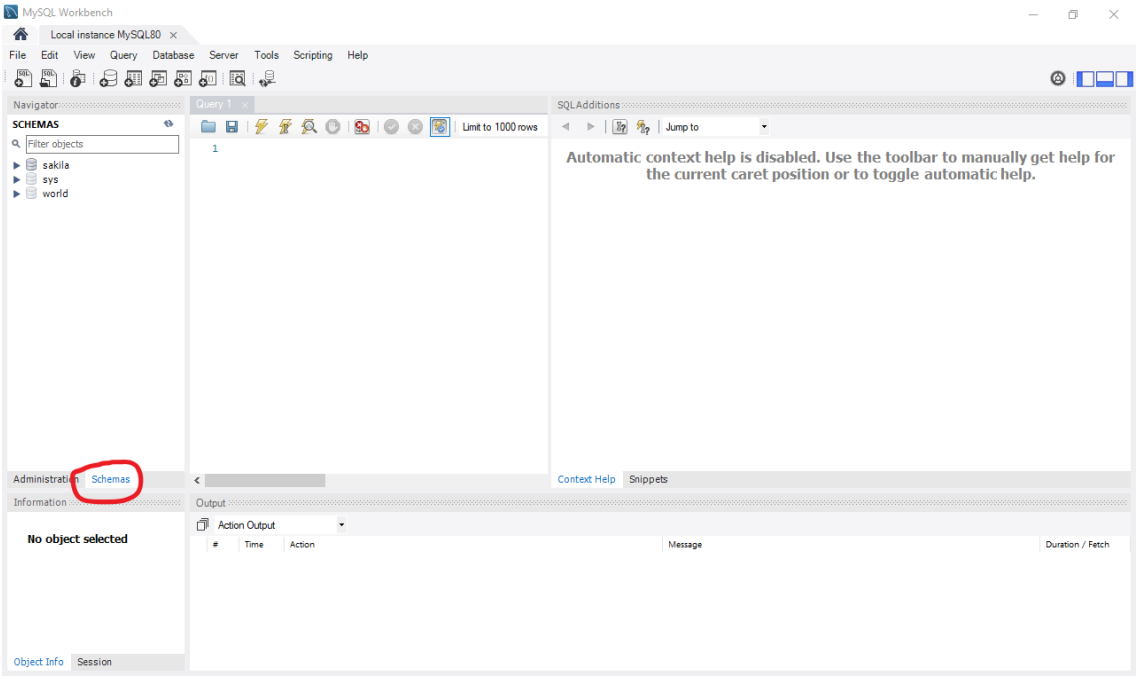

Figure 4 MySQL Workbench. Schemas

2.1.3 There will be some example schemas installed. You can expand the schema and the “Tables” section to view the tables in the database (Figure 5)

Note: The example schemas will be shown if you followed all steps in the setup tutorial. If not, then you may skip this part and keep it as reference for when you will create your own schemas and tables.

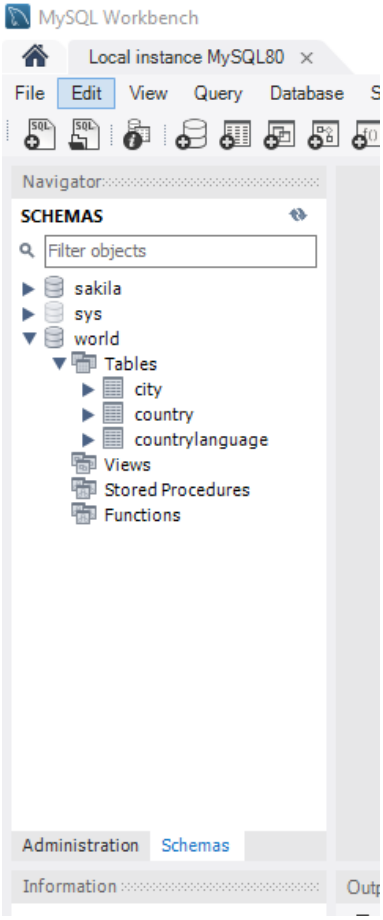

Figure 5 Expanding database schema and tables

2.1.4 Right click on a table name (e.g. city) and click on “Select rows – limit 1000”. This will show you a list of the first 1000 rows in the table, using a simple SELECT SQL query that you can view in the top window. (Figure 6)

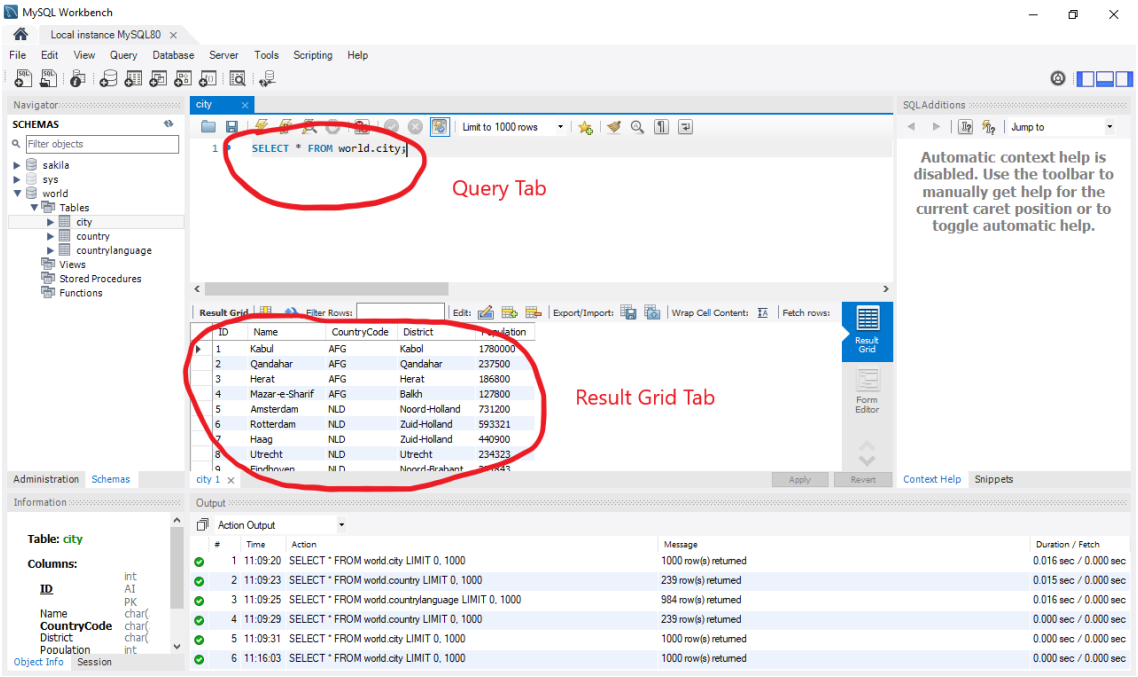

Figure 6 Select rows – limit 1000

2.1.5 Right click on a table name (e.g. city) and click on “Alter Table”. This allows you to view and modify the table definition (table name, columns, constraints, indexes, foreign keys, etc.). (Figure 7)

Note: On small screens (e.g. laptop) some tabs will not show properly. If this is the case, you may drag the bottom tab (Action Output) to fill less space, so that you can view the (more important) column definitions.

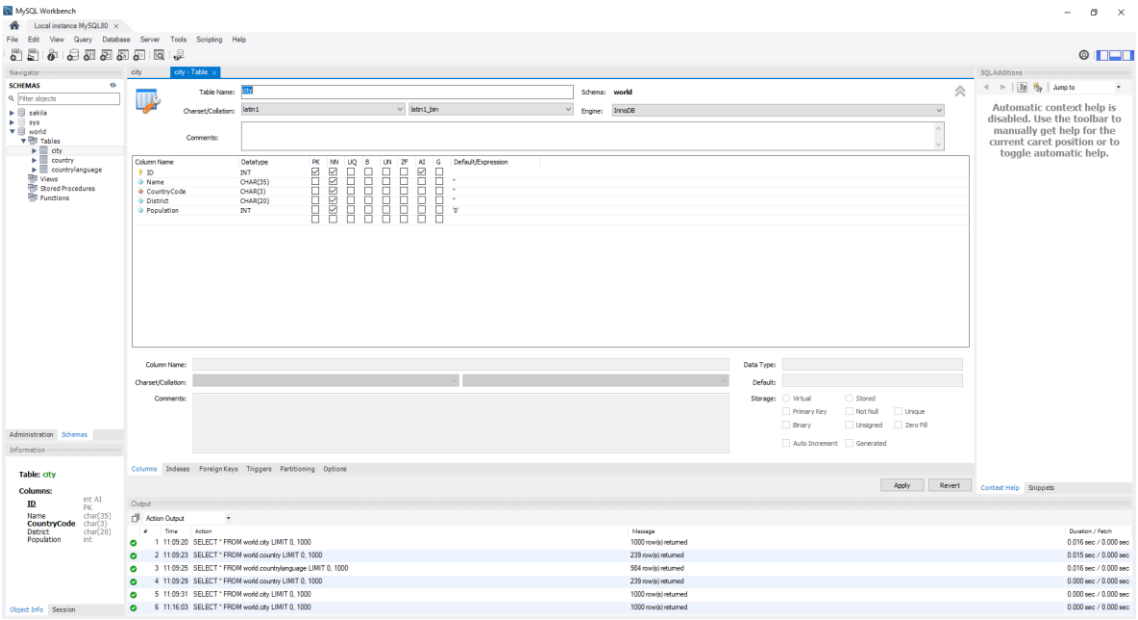

Figure 7 Alter Table

3 CREATING A DATABASE WITH MYSQL WORKBENCH

This section describes how to create a simple database shown in Figure 8 using MySQL Workbench, then adding data and writing simple queries. While this example is quite simple, the tutorial is valid for creating and working with any MySQL database in MySQL Workbench.

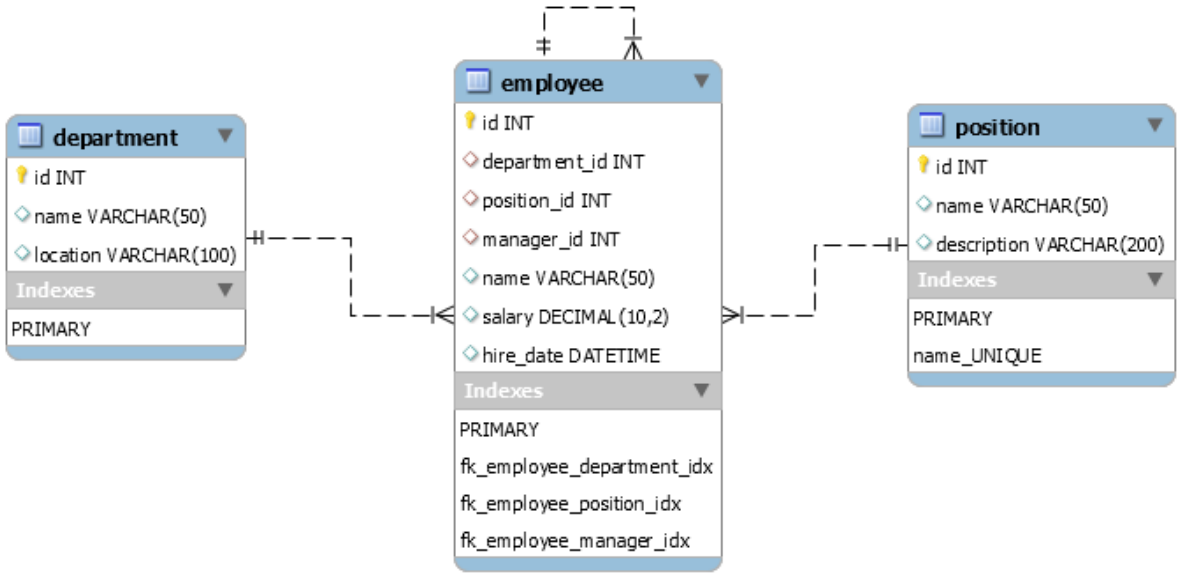

Figure 8 Example schema. Company database

3.1 Creating the database schema and tables (DDL)

The database schema and tables in this tutorial are created according to the representation in Figure 8. It is common to define the schema first and then to implement it into a DBMS. Now, this design step is already done for the simple example, and we are just implementing the schema into MySQL using MySQL Workbench.

3.1.1 Creating the database schema

Select the “Schemas” tab and right click in the blank area under the “SCHEMAS” list view. Click on “Create Schema”. A new database is about to be created. Change the “Name” field to *company* then click “Apply”. A popup window will open. Click “Apply” and then “Finish”.

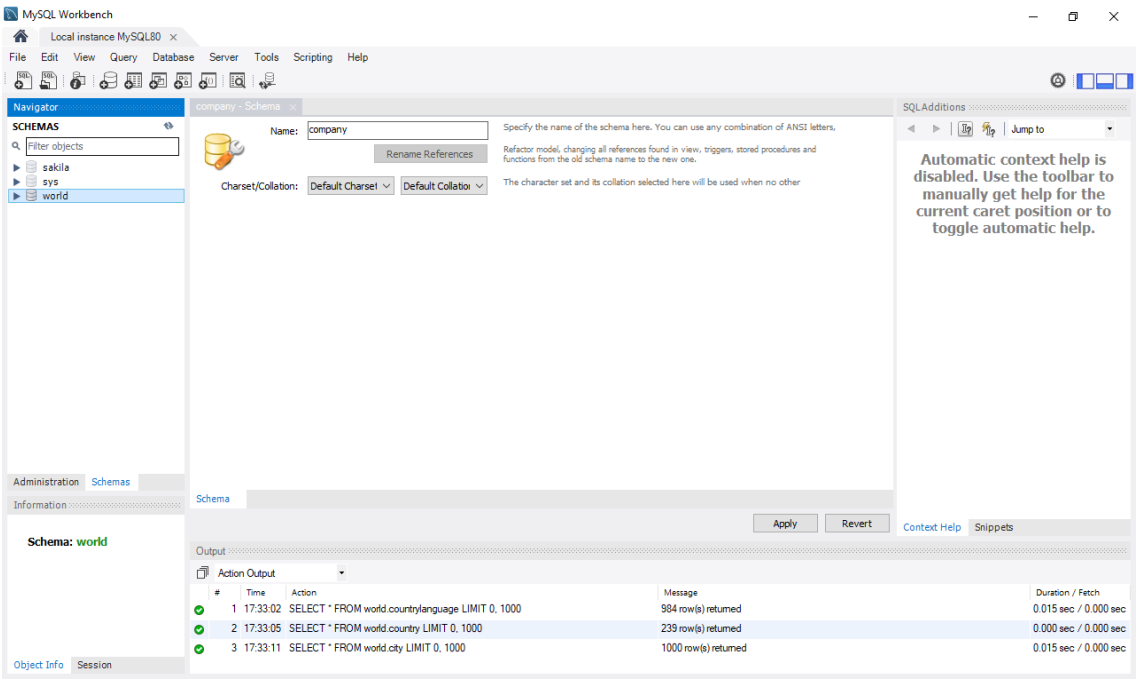

Figure 9 Creating a new database schema

3.1.2 Creating the database tables

Now that the database is created, we need to add the tables: *department*, *employee*, *position*.

Click on the arrow next to the *company* schema, then right click on “Tables”, and click on “Create Table”. A new table is about to be created. Change the “Table Name” field to *department.*

3.1.2.1 *Adding columns and constraints*

To add a new column to a table, click on the first blank row under "Column Name". Then, change the name to *id* to define the first attribute for the *department* table [1]. The *id* attribute will be of type INT (leave INT in the "Datatype" field) and will be the PRIMARY KEY for the *department* table. Select AI (Auto Increment) constraint to make the id auto generated.

Then, continue to add the other column definitions as shown in Figure 10. After adding all columns, click "Apply". A popup window will open, where you can view the DDL command generated by MySQL Workbench:

```
CREATE TABLE `company`.`department` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(50) NULL,
  `location` VARCHAR(100) NULL,
  PRIMARY KEY (`id`));
```

Click on "Apply" to execute the DDL command and create the table into the database, then click on "Finish".

[1] The *id* column makes it easier to uniquely identify records in the created tables, otherwise the PRIMARY KEY can also be a combination of attributes (in this case *name* and *location* can uniquely identify a department)

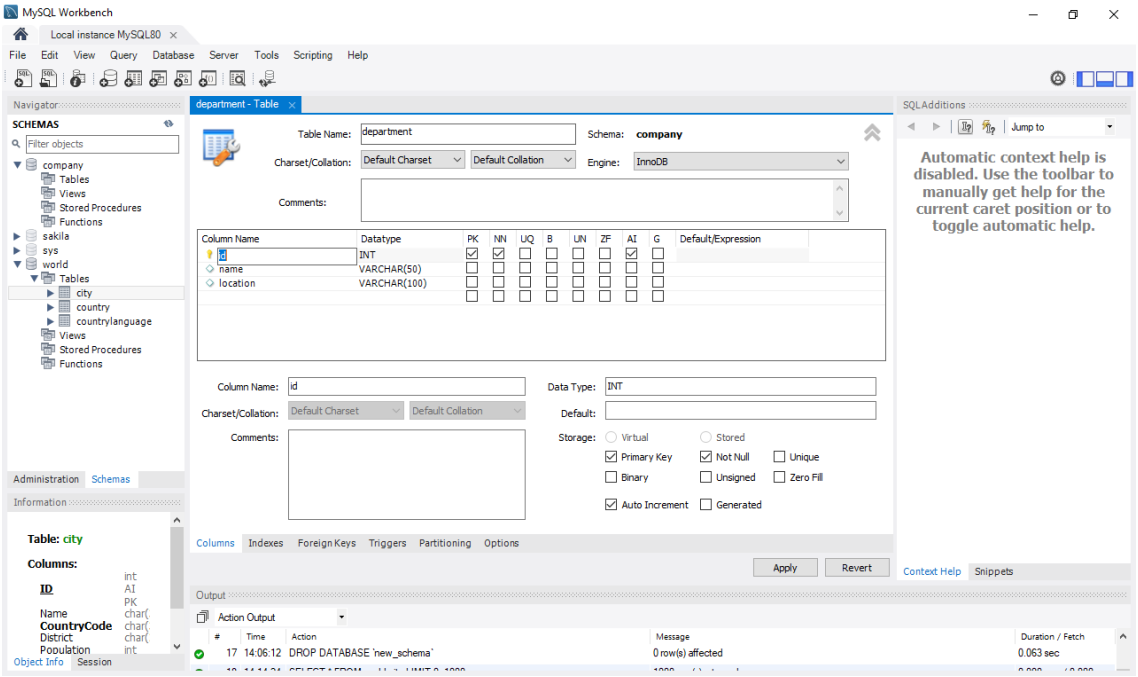

Figure 10 *department* table definition

3.1.2.2 *Then, proceed to add the other tables in the same way: employee, position.*

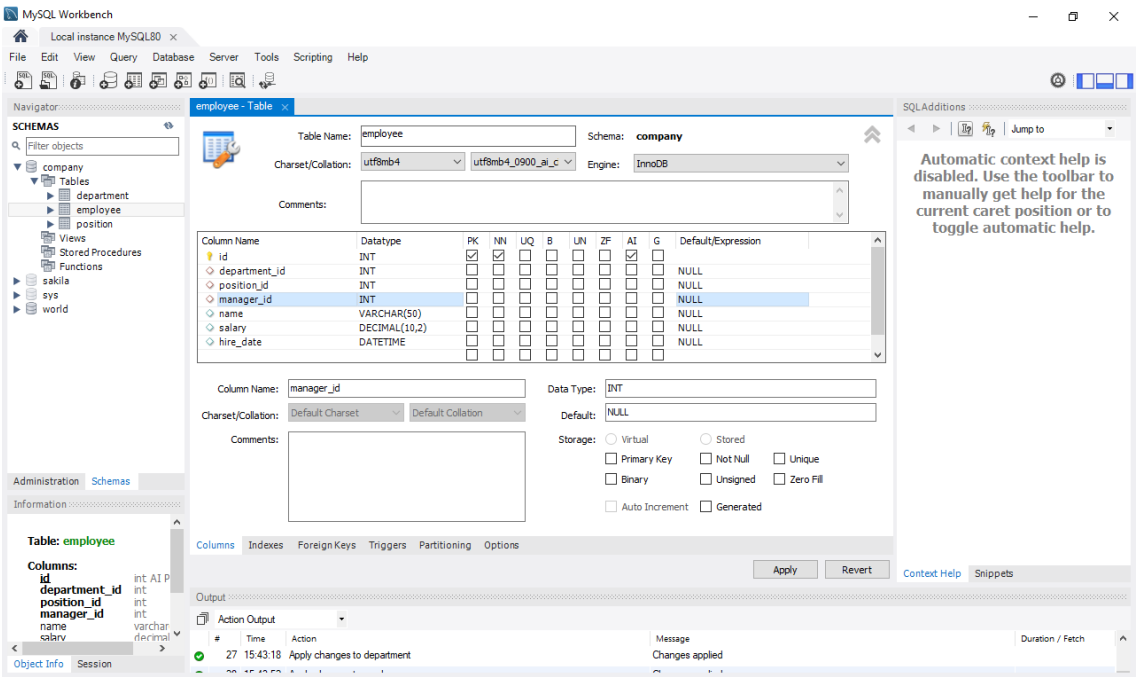

Figure 11 *employee* table definition

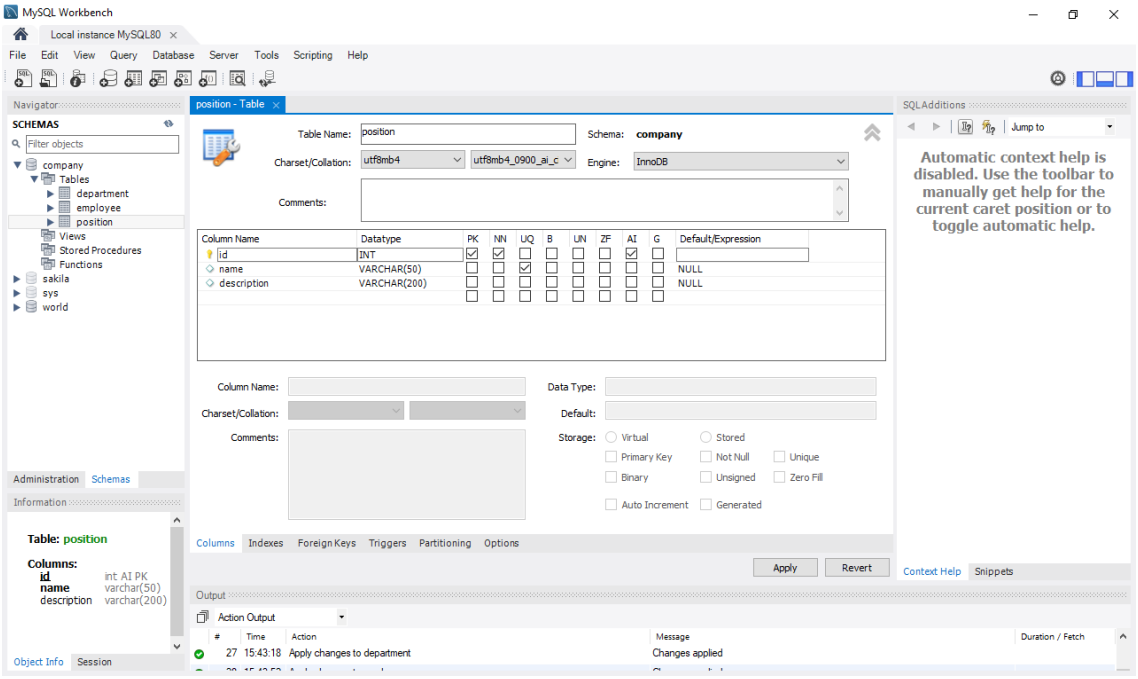

Figure 12 *position* table definition

After creating the tables, they will show in the schemas navigator as shown in Figure 13.

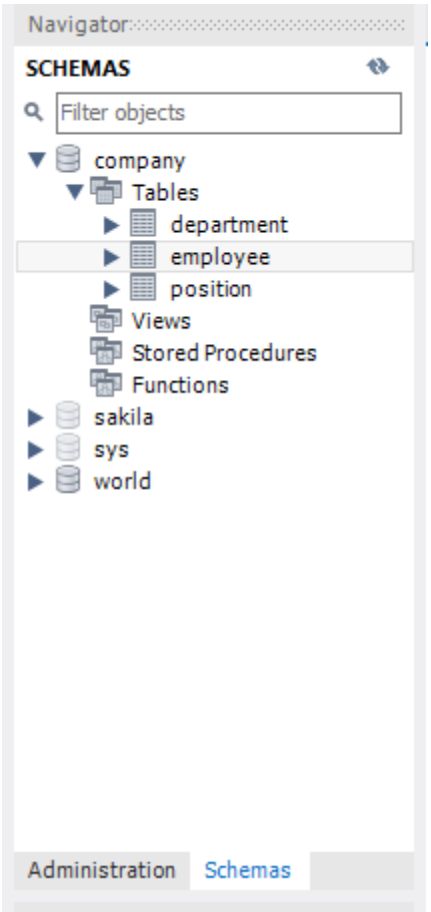

Figure 13 The tables are now visible in schemas navigator

3.1.2.3 *Defining the relationships between tables*

After the tables are created, we can add all the relationships as defined in the example schema (Figure 8). **For this, we need to modify the existing tables by right clicking on a table name and then clicking on "Alter Table".**

Department – Employee

Relationship description

- A department can have one or more employees
- An employee can belong to a single department
- Implementation: *one-to-many* relationship

Implementation

We need to define a FOREIGN KEY in the *employee* table to reference the *department* table. Right click on *employee* table, then click on "Alter Table". Switch to "Foreign Keys" tab and create the FOREIGN KEY constraint as shown in Figure 14. You have to define a name, the referenced table, the column that will be set as foreign key, the referenced column and the ON UPDATE and ON DELETE rules (we will use CASCADE for all foreign keys in this example).

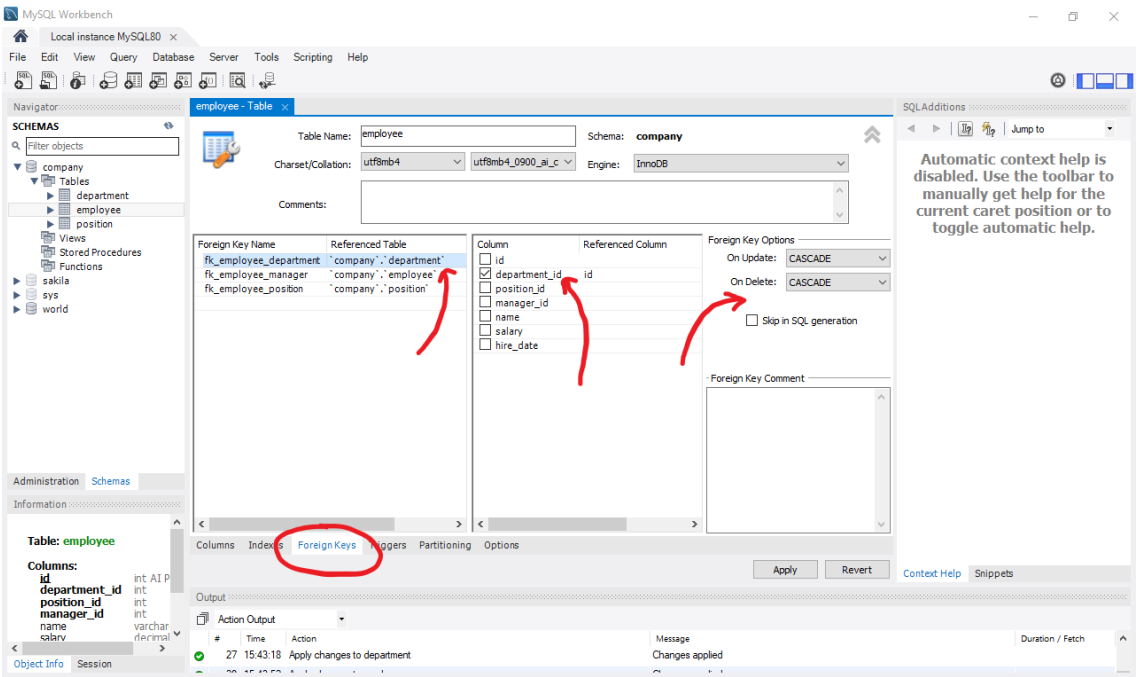

Figure 14 Adding foreign keys to the *employee* table. Department - Employee

Employee – Position

Relationship description

- An employee can have a single position in the company
- There can be more employees with the same position in the company (A position may be assigned to one or more employees)
- Implementation: *one-to-many* relationship

Implementation

We need to define a FOREIGN KEY in the *employee* table to reference the *position* table. In the "Foreign Keys" tab in *employee* table, add the FOREIGN KEY constraint as shown in Figure 15.

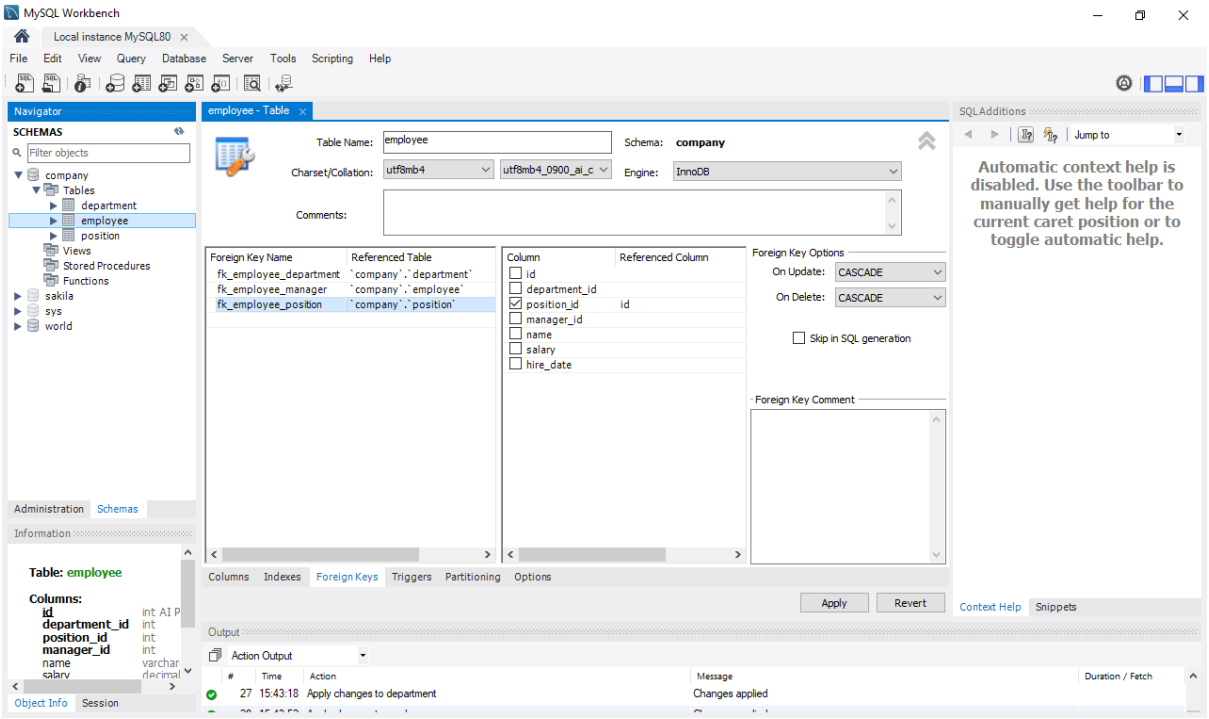

Figure 15 Adding foreign keys to the *employee* table. Employee - Position

Employee – Employee

Relationship description

- An employee can have a single manager (which is also an employee of the company)
- There can be more employees that have the same manager (A manager may be assigned to one or more employees)

- Implementation: *one-to-many/self-referencing* relationship

Implementation

We need to define a FOREIGN KEY in the *employee* table to reference the *employee* table. In the "Foreign Keys" tab in *employee* table, add the FOREIGN KEY constraint as shown in Figure 16.

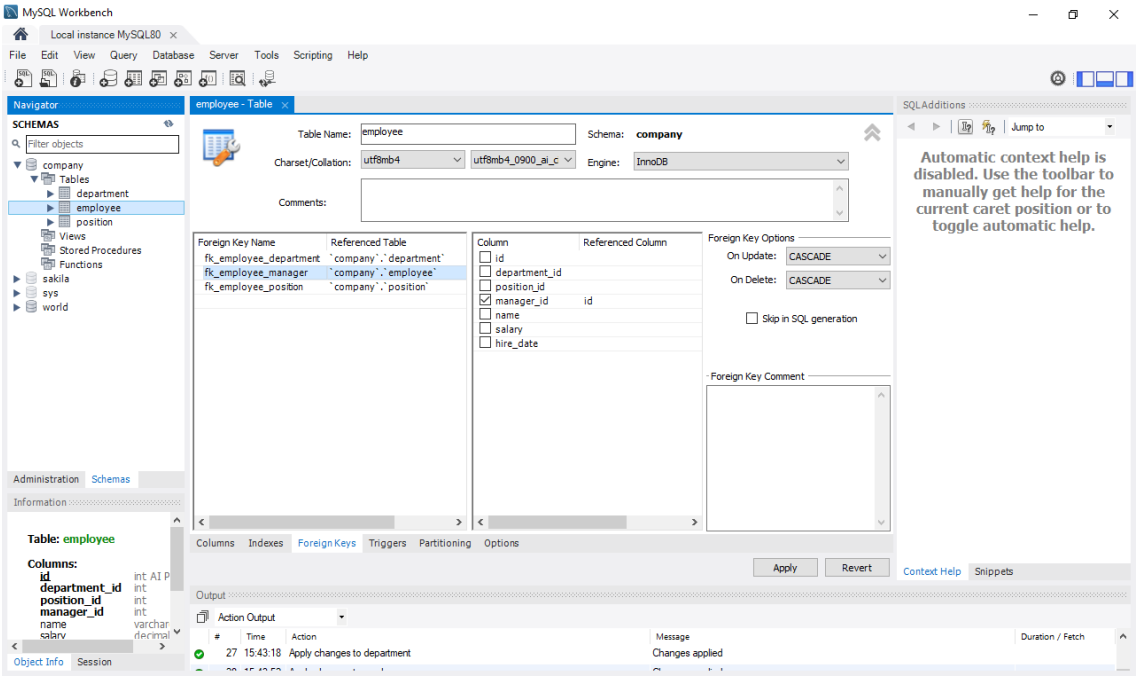

Figure 16 Adding foreign keys to the *employee* table. Employee - Employee

4 ADDING DATA INTO A DATABASE WITH MYSQL WORKBENCH

This section shows how you can add/update/remove data to existing tables using MySQL Workbench.

4.1 Using the GUI

To add data, Right click on a table and click on "Select Rows – Limit 1000". A view will open as in Figure 17 where you can view the existing data and you can add or remove data. To add a new row, click on an empty row and edit the fields. You can add multiple rows at once. After adding the rows to the table, you have to click on "Apply" to save them into the database. A popup window will open with the SQL commands that will be applied. Click on "Apply" and then "Finish".

To remove an existing row, Right click on the button to the left side and click on “Delete Row(s)”. As before, you have to click on “Apply” to remove the rows from the database. Alternatively, you can click on “Revert” and the operation will be cancelled (the rows will be restored to the table view).

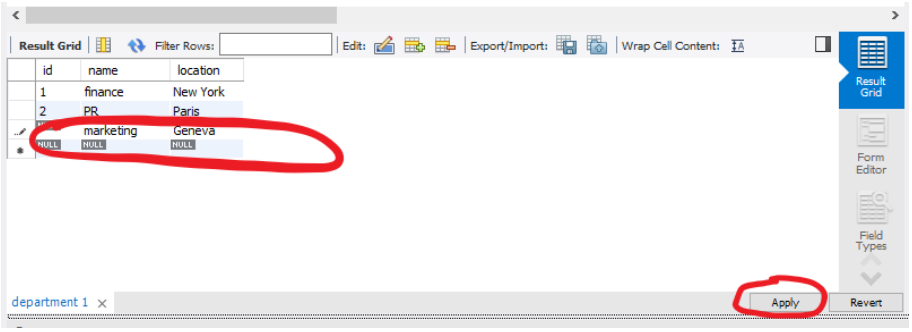

Figure 17 Adding rows to a table

4.2 Running an SQL query that contains the data

To open a new query tab in Workbench, click on “File” then click on “New Query Tab”. A blank tab will open where you can write your SQL query. There is one more step before running the query. You have to select the target database. Double click on the *company* database. The name will turn bold. Now you can apply the query by clicking on the yellow bolt icon. See Figure 18.

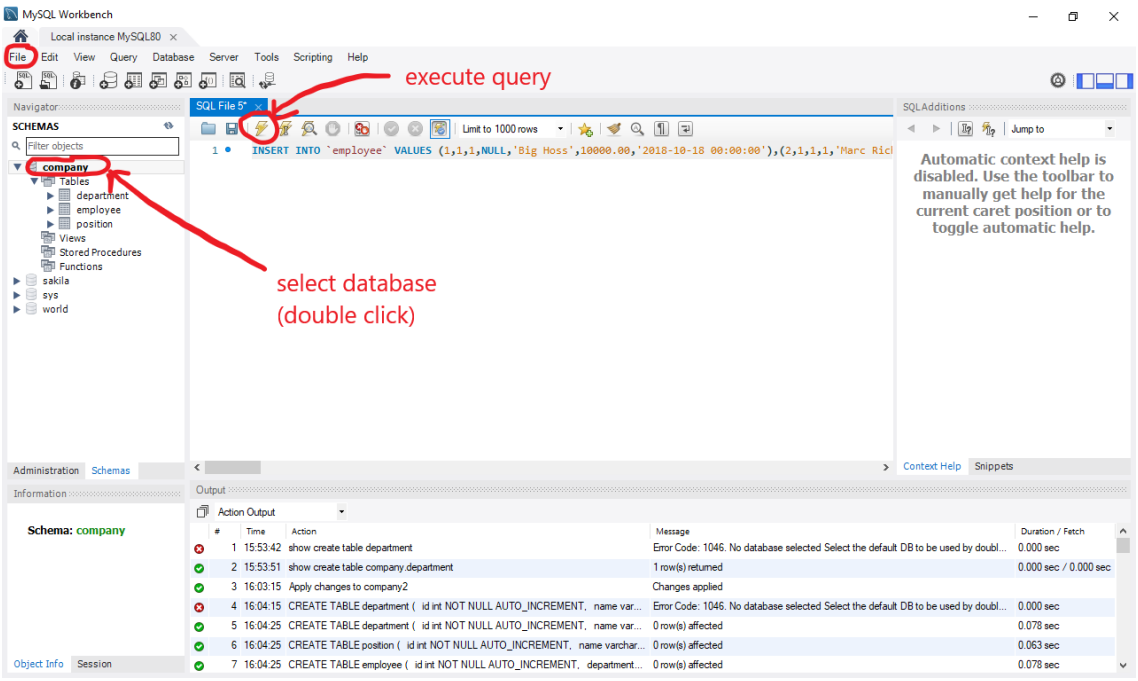

Figure 18 MySQL Workbench – SQL Editor

5 EXPORTING DATA FROM A TABLE

To save the result of a query, you can export as *csv* file as shown in Figure 19. Give the file a name and then open the file with a text editor. You will see the content in *csv* format that you can use later to process the data.

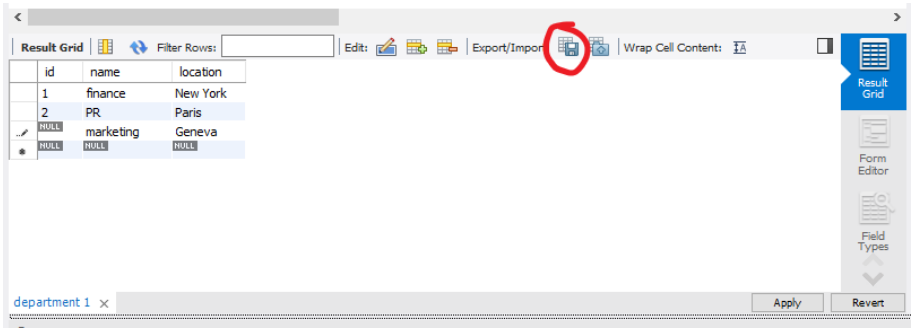

Figure 19 Exporting data to a *csv* file

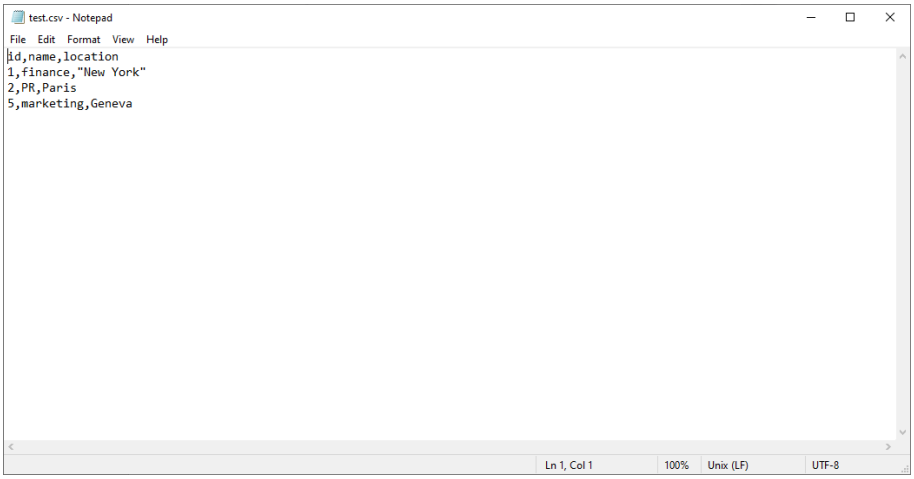

Figure 20 Data exported as csv file

6 WRITING SQL QUERIES

This section presents some SQL query examples on the *company* database. To write an SQL query in MySQL Workbench, see Figure 18.

Q1

```
-- Write a query in SQL to display all the information of the employees --
SELECT * FROM employee;
```

Q2

```sql
-- List the employees name and salary. Show only those having the salary
between 5000 and 7000 EUR and order them descending by salary --
SELECT name, salary FROM employee
WHERE salary BETWEEN 5000 AND 7000
ORDER BY salary DESC;
```

Q3

```sql
-- List the employees name and the name of their department --
SELECT emp.name AS employee, dept.name AS department FROM employee AS emp
INNER JOIN department AS dept
ON dept.id = emp.department_id;
```

Q4

```sql
-- Get the number of employees in the company --
SELECT COUNT(id) AS employee_count FROM employee;
```

Q5

```sql
-- Get the number of employees from department id 4 --
SELECT department_id, COUNT(id) AS employee_count FROM employee
WHERE department_id = 4
GROUP BY department_id;
```

Q6

```sql
-- Get the number of employees for each department --
SELECT department_id, COUNT(id) AS employee_count FROM employee
GROUP BY department_id;
```

Q7

```sql
-- Get the average salary for each department --
SELECT department_id, AVG(salary) AS average_salary FROM employee
GROUP BY department_id;
```

Q8

```
-- Get the number of analysts from department id 3 --
SELECT department_id, position_id, COUNT(id) AS employee_count FROM employee
WHERE department_id = 3 AND position_id = 3;
```

Q9

```
-- Get the number of analysts for each department --
SELECT department_id, position_id, COUNT(id) AS employee_count FROM employee
WHERE position_id = 3
GROUP BY department_id;
```

Q10

```
-- List the employees name and the name of their manager --
SELECT manager.name AS manager_name, emp.name AS employee_name FROM employee
AS emp
INNER JOIN employee AS manager ON emp.manager_id = manager.id
ORDER BY manager_name ASC;
```

Q11

```
-- List the employees name, salary, the name of their position --
SELECT emp.name AS employee, pos.name AS POSITION, emp.salary FROM employee
AS emp
INNER JOIN POSITION AS pos
ON pos.id = emp.position_id;
```

Q12

```
-- List the employees name, salary, the name of their department and position
--
SELECT emp.name AS employee, dept.name AS department, pos.name AS POSITION,
emp.salary FROM employee AS emp
INNER JOIN department AS dept
ON dept.id = emp.department_id
INNER JOIN POSITION AS pos
ON pos.id = emp.position_id;
```