**Third problem.**

We start by running the following command (I could use a Dockerfile also, but I thought this approach would work better in this context)

The container will be named db\_container and will expose the port 3306 ( in case we’ll ever need it in the future )

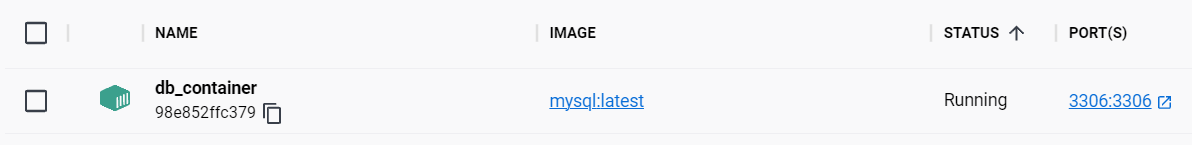
***docker run -d --name db\_container -e MYSQL\_ROOT\_PASSWORD=1234 -p 3306:3306 mysql:latest***

After that, we use the following command to bring the .sql file in the container:

***docker cp company.sql db\_container:/docker-entrypoint-initdb.d/***

The conatiner needs to be restarted to make sure it runs our script:

***docker restart db\_container***

******

We can see that the container is running accordingly. (I could use the CLI to visualize it but I prefered the GUI because it’s more “pleasing to the eye”)

We connect to the container with:

***docker exec -it db\_container mysql -uroot -p1234***A screenshot of a computer

Description automatically generated

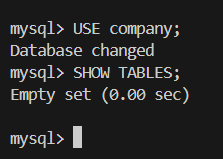
Then create the database:

***CREATE DATABASE company;***

***A black screen with white text

Description automatically generated***

After that, we need to import the company.sql tables into this DB, as we can see there are no tables associated with it (for now):



For that purpose, we’ll run:

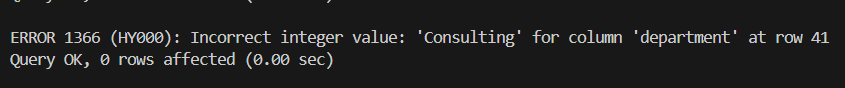
***SOURCE /docker-entrypoint-initdb.d/company.sql;***

***A screen shot of a computer

Description automatically generated***

Successfully imported.

There is also an error in this file:



The error shows because instead of the Primary key of department, there is a “Consulting” term inserted, which is a string. An integer is expected, so we’ll need to “repair” the broken record, because otherwise, the whole INSERT stataement is dropped.

We’ll drop the database and create it again after bringing the correct file inside the container:

A screenshot of a computer program

Description automatically generated

Delete the faulty file inside the container:

A screen shot of a computer

Description automatically generated

The faulty line has been modified:



Put 7 instead of ‘Consulting’.

Now we copy the file back to the container with **docker cp** command used earlier.

After this, I repeated the previous steps and got it to work this time:

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

We select each table and see we have the correct output this time:

A screenshot of a computer

Description automatically generated

**Task 3:**

We’ll use the following command to create the “ceo” user with the password “1234”:

***CREATE USER 'ceo'@'%' IDENTIFIED BY '1234';***

Where ‘%’ means that the user can connect from anywhere. For example, it can be replaced by “localhost”

Granting all privileges on company database to the new user:

***GRANT ALL PRIVILEGES ON company.\* TO 'ceo'@'%';***

***A screen shot of a computer

Description automatically generated***

I read that you should flush the privileges for them to take on immediately, so that’s what I’ve done too, by using ***FLUSH PRIVILEGES;***

**Task 4:**

First, I ran the command ***SELECT department, AVG(salary) FROM employees GROUP BY department;*** but I wasn’t satisfied because I wanted to see the name of each department instead of their ID:

A screenshot of a computer

Description automatically generated

So I replaced the command with ***SELECT department\_name, AVG(salary) FROM employees JOIN departments ON employees.department = departments.department\_id GROUP BY department;***

The final output:

A computer screen with white text

Description automatically generated

For a nicer look, we can round the result with the **ROUND** function:

***SELECT department\_name, ROUND(AVG(salary), 2) AS Avg\_salary FROM employees JOIN departments ON employees.department = departments.department\_id GROUP BY department;***

I also added “AS Avg\_salary” for this “nicer look”.

These will result into:

A screenshot of a computer screen

Description automatically generated

**Tasks Done.**