Jupyter Notebook Exercise

Part 1 - Setting up

1. Create a new directory
2. Inside the directory, create a new virtual env and activate it
3. Run the command `pip install notebook matplotlib ipython-sql pymysql`
4. Run the command `jupyter notebook`
5. A webpage will open on `localhost` with the Jupyter dashboard

Part 2 - Trying it out

1. On the Jupyter webpage, select the `new` dropdown and select `Python 3`
2. Give it a name by clicking on `Untitled`
3. Inside the textbox that says `In [ ]:` next to it, paste the following: `%matplotlib inline` (This will tell Jupyter to output any matplotlib visualisations inline)
4. Hit `Run`
5. Paste the following code into the next textbox, you should see a bar chart appear

```
 import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(18, 7))

x_data = ['A', 'B', 'C', 'D', 'E']
y_data = [5, 10, 15, 20, 25]

x_pos = np.arange(len(x_data))

plt.bar(x_pos, y_data)
plt.xticks(x_pos, x_data)
plt.ylabel('Value')
plt.title('Simple Bar Chart')

plt.show()
```

Jupyter Notebooks use a series of aliases, shortcuts and extensions that they call `magics` .

To get a list of all available to you, run `%lsmagic` .

Part 3 - Connecting to SQL

We're going to try and connect to the database we setup previously through the notebook.

It'll be easier for us if we remove some of the previous inputs, so go ahead and remove the block of code by clicking on the cell and hitting `D` twice.

1. Paste `%load_ext sql` into the second cell and hit `Run`
2. Paste `%sql mysql+pymysql://username:password@localhost:3306/database` into the third cell and hit `Run` (you will need to replace username/password/database with whatever you named them previously)
3. Paste `%sql SELECT * FROM person` and hit `Run` . Hopefully you should see some records return!
4. The magic `%sql` returns a result. The result is a list of tuples which you can manipulate with basic python. Try to parse your data so that the values for each column are in a list. This should look somewhat like id = [1, 2, 3], first_name = ["Jane", "Ali", "Joe"] ect ect.

Hint: You can access the list of tuples like this:

```
 data = %sql SELECT * FROM person
# Print the data table
print("The data table returned looks like this:\n", data)

# Print some of the values from the list of tuples
print("The first row, first column value is:", data[0][0])
print("The first row, second column value is:", data[0][1])
print("The first row, third column value is:", data[0][2])
```

Part 4 - Importing a Data Set

You'll get the opportunity to put matplotlib to the test, but first we must insert some data into our database. We will be using this open source data set about Titanic.

1. If your docker containers are still running, navigate to `localhost:8080`

2. If not, go to the directory from the databases module with the docker compose final and run `docker-compose up -d`. Retry step 1.
3. After logging in, click on `Import` on the left hand side
4. Upload the `titanic-data.sql` file in the `File upload` section, and hit `Execute`
5. You now have titanic data!

Part 5 - Playing with the data

Now that you have a table of titanic passenger data, you can go ahead and try to create a simple visualisation for the data.

Some examples:

1. Chart for number of people who survived vs. didn't survive
2. Chart of number of people for each sex who survived vs. didn't survive
3. Pie-chart of amount of people per class