

EECS116/CS122A

Winter 2021

Mini-Project 2 – Python + MySQL

In this project, we will create a database and tables using MySQL, and import the data into tables. Then you are required to write a Python program that implements a non-graphical user interface for MySQL. Your Python program should interact with the user and provide complete sentences to answer a selection of queries. Due by 5PM on Tuesday 3/9. Turn in the .py source code file to “Mini Project 2” on Canvas.

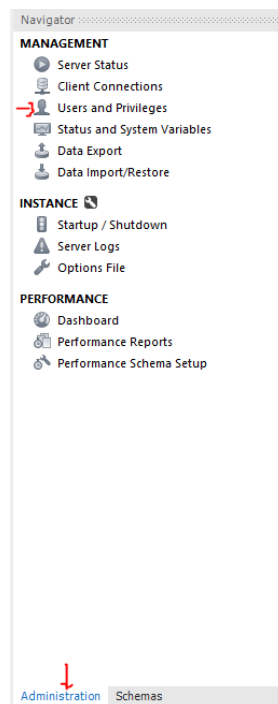
Deliverable

A .py source code file for this project.

STEP 1 – Set Up MySQL

For testing and grading purposes, you must follow the guide below to create and use a new MySQL account.

1. Open MySQL Workbench, select the “Administration” tab on the left then click “Users and Privileges”.



2. Click “Add Account” button, enter “mp2” for Login Name and “eecs116” for password. Click “Apply”. You can ignore the following warning:

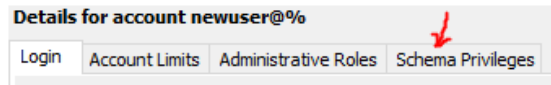
Password: Type a password to

Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.



m Password: Enter password again

Expire Password


3. Keep the newly created account selected, switch to the “Schema Privileges” tab.




4. Click the “Add Entry” button, select “All Schema” then click OK. Click “Select ‘ALL’”, then Apply.

5. Click the  on the top left to return to the homepage. Click  to create a new connection to the “mp2” account. Enter the user name and password.

6. Once logged in, create a schema named “flights” and **make it the default schema**.

7. Click  button and open the create_table.sql file. Click execute to create tables.

Then click  again to open the import_data.sql file. Execute to import data. The flight database has 9 relations:

The relation Airport contains basic information about an airport

The relation Flight contains basic information about a complete flight route

The relation Flight Leg contains information about a flight segment within that route

The relation Leg instance contains information on a flight leg from a particular day

The relation Fare contains information on a fare for a flight

The relation Airplane type has information about different airplane models

The relation Can land has information about whether a certain type of airplane can land at a particular airport

The relation Airplane has information about a single airplane

The relation Seat reservation has information about a particular reservation for a flight

8. Now your MySQL is set up and ready to use.

STEP 2 - Install the Library

We will use PyMySQL library to connect Python to MySQL. Before we start, make sure you have MySQL 5.6 or above and Python 3.6 or above. This is the minimum requirement, you need to upgrade MySQL and Python if you do not meet this requirement.

To install PyMySQL, open your command prompt(Windows) or Terminal(macOS) and type:

(If you installed both Python2 and Python3)
python3 -m pip install PyMySQL

OR

(If you only installed Python3)
python -m pip install PyMySQL

We provide some code snippets below. You can start to modify the code to do this project and provide the requested functions described at the end of this document.

- Sample Code -

In the beginning of the py file, import the library you need:

```
import pymysql
```

To Create a Connection with MySQL Server for the database “sample_python”:

```
db = pymysql.connect(host='localhost',  
                     user='mp2',  
                     passwd='eecs116',  
                     db= 'flight')  
cur = db.cursor()
```

In the host part, ‘localhost’ is the address of the server (127.0.0.1 means the localhost which is your own computer; if you want to connect to other computers just change the address), and “flight” is the database name. This code snippet uses user id “mp2” and password “eecs116” to login.

To Perform SQL Queries:

```
sql="SELECT * FROM customer"  
cur.execute(sql)
```

To Get the Results:

```
for row in cur.fetchall():      # cur.fetchone() gets one result at a time  
    print(row)
```

To Add a Tuple into a Table:

```
sql = ("INSERT INTO bank.customer(customer_ID, customer_name)  
VALUES(default, %s)")    # %s is a place holder for inserting a variable here  
val = (cust_name)      # customer name is stored in variable cust_name  
cur.execute(sql, val)  
db.commit() #use commit to save the changes you made to the database
```

Note that:

- “default” means to use the default value for customer_ID, which is auto-incremented.

Use `db.close()` to end the connection.

STEP 3 – Write the Python Program

You are about to build a demo program for an AI database capable of interacting with the user through English rather than SQL. Your Python program should interact with the user through the Python console. Your demo program will handle at least the following six queries/tasks with corresponding answers:

1. Query: Find the cheapest flight given airports and a date.
Please enter the airport code for the departure airport:

- Please enter the airport code for the destination airport:
What is the date of the flight in yyyy-mm-dd?
Result: The cheapest flight is \$FlightNumber, and the cost is \$FareAmount.
2. Query: Find the flight and seat information for a customer.
Please enter the customer's name:
Result: The flight number is \$FlightNumber, and the seat number is \$SeatNumber.
3. Query: Find all non-stop flights for an airline.
What is the name of the airline?
Result: The non-stop flights are: \$FlightNumber1, \$FlightNumber2, \$FlightNumber3...
4. Task: Add a new airplane.
Please enter the total number of seats:
Please enter the airplane type:
Result: The new airplane has been added with id: \$AirplaneID
5. Increase low-cost fares(≤ 200) by a factor.
Please enter a factor (e.g. 0.2 will increase all fares by 20%):
Result: \$NumAffected fares are affected. (e.g. "3 fares are affected")
6. Remove a seat reservation.
Please enter the flight number:
Please enter the customer name:
Result: Seat \$SeatNumber is released.

The "\$" symbol indicates a variable that you need to replace with an answer. To simplify the project, you may assume each question leads to a non-empty result(i.e., at least one tuple is returned for each question). A non-stop flights are flights with only one leg. A flight leg is a segment of a flight. For example, a flight from LAX to YYZ may have two flight legs, the first one from LAX to PHX and the second from PHX to YYZ.

Requirements:

1. Your program should show a list of queries and tasks to the user and ask the user to choose one to start with.
2. After the user has selected a query/task, collect more information from the user if needed.
3. Your Python program must communicate with the database for each query/task.
4. After reporting the result, your program should ask the user to choose another query/task or quit the program.
5. If your query involves insert, update, and delete of the database, you should commit the changes to the database.
6. Your python source code file must be named as "**mp2-xxxxxxx.py**", xxxxxxxx being your student ID.

Reference: <https://gist.github.com/onzfonz/eb8025de13e67c783795>