Opening Remarks

This document is intended to assist with creating a test environment for you. This test environment is intended for use in testing the UWA, an application designed for the QA Manual Test training course.

The document assumes you do not have any of the tools necessary for a test environment. If you have some or all the following software already installed, adjust your effort accordingly. Your test environment should contain the following:

1. Python interpreter software and libraries
2. MySQL Database environment
3. Python IDE

Test Environment Implementation

In your browser, go to https://www.python.org/downloads/. From there you will see the latest version and if you scroll down, you can find other earlier versions of the software. It is common for people to choose the latest version of Python. At the time I set up my test environment for this course, I was using Python 3.8.3. I found that it was compatible with the other software included in my test environment. So, please be aware if you choose the latest version of Python, it might not be supported by the other software. Do some research or check with your instructor.

- Download and install Python. Make sure to check the **Add Python to PATH** option on the installation setup screen.

- Download and install MySQL Community Server and MySQL Workbench. You can skip this step if you already have a MySQL server installed.

- Install an IDE: ATOM, VS Code, or PyCharm. Use a Search Engine to find the download of your choice. Skip if you already have an IDE installed.

- From your IDE, open the Terminal command line and execute the following PIP commands (***If you experience install errors you should try to proceed to completion***):

    - NOTE →Checkout Python Flask to learn more about Flask
    - pip install Flask
    - Might need to upgrade C++: https://visualstudio.microsoft.com/visual-cpp-build-tools/ before proceeding
    - pip install mysqlclient
        - (An alternate class can be **mysql-connector-python, if mysqlclient fails**)
        - If this class is needed, use program main_mysql_conn.py instead of main.py
    - pip install flask-mysqldb
    - pip install WTForms
    - pip install Flask Flask-WTF
    - pip install flask.ext.login
    - pip install Flask-Mail

o   pip install flask-mysql-connector (if mysql-connector-python is used)
- Install Flask-MySQLdb with the command: **pip install flask-mysqldb**

Environment Installation Requirements

Follow the instructions below on how to get started with MySQL after the install.

- Open **MySQL Workbench** from your START menu

- Enter your MySQL user details after clicking the "Local Instance" connection button

- Click the **Test Connection** button near bottom of window. If successful, you can click **OK**

- Open your connection

- Execute the following SQL statements as one script or as four individual commands:

    o   Editable SQL statements to add in the Query window of the workbench

    o   An example follows the SQL code

SQL CODE

ONLY use the following SQL data if you are working with Assessment 2 first. Assessment 1 requires the use of the SQLData file at the https://github.com/wallytauriac/UWA site.
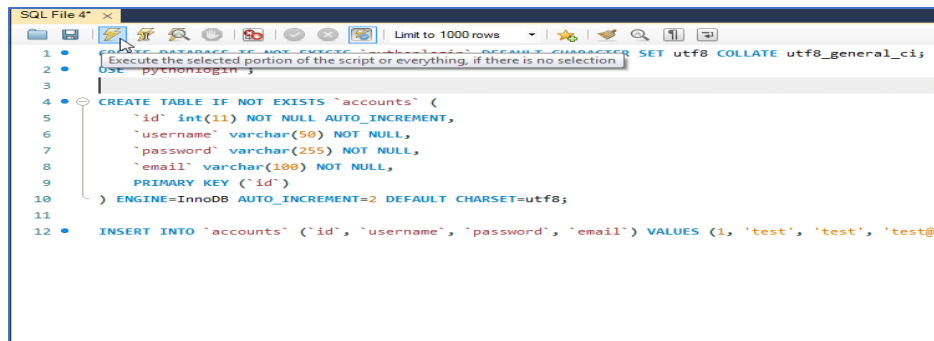
```
CREATE DATABASE IF NOT EXISTS `pythonlogin_advanced` DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci;

USE `pythonlogin_advanced`;

CREATE TABLE IF NOT EXISTS `accounts` (
        `id` int(11) NOT NULL AUTO_INCREMENT,
        `username` varchar(50) NOT NULL,
        `password` varchar(255) NOT NULL,
        `email` varchar(100) NOT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

INSERT INTO `accounts` (`id`, `username`, `password`, `email`) VALUES (1, 'test',
'0ef15de6149819f2d10fc25b8c994b574245f193', 'test@test.com');
```

EXAMPLE

The above SQL statements will create your database **pythonlogin_advanced** with the table **accounts**, along with a test account that can be used for testing purposes. The test account will have an encrypted password of "test".

When Exceptions Are Encountered

Normally installation of software, such as Python, MySQL, and an IDE goes smoothly. However, beneath the surface there may be exceptions awaiting you. My first student encountered a compatibility issue. I am including this information to help others. As other issues are encountered, I plan to include some helpful hints in this section of the document.

Compatibility Issue #1

The Problem

In the process of executing the PIP commands, an error occurred for the "MYSQLCLIENT" pip install. This can happen when the Python version is not supported by the class or library. The MYSQLCLIENT library did not yet have support for the latest version of Python. This library also requires C++ libraries which was causing errors.

Simply put, release 12 of Python was found not yet supported by all MySQL connection solutions.

The Solution

Internet searches for the errors (Example - could not build wheels for mysqlclient) revealed that compatibility was at the core of the problem. So, I knew that Python 3.8 was not having issues with the MYSQLCLIENT library. That means the uninstallation of Python and re-installation of Python 3.8.3 should solve the problem. Unfortunately, going backwards with the programming language can cause new problems. So, I did some additional research to find alternatives. Here is what I found.

There are four ways to connect Python code to MySQL. In other words, MYSQLCLIENT is not the only way. There is a website that helps with the details: https://learningactors.com/4-ways-you-can-connect-python-to-mysql/.

Tuples and Dictionaries

I have included this section because of an unusual and unexpected happening when I created an alternative python module for use with the class known as **mysql-connector-python**. It is a class that is useful if you are unable to PIP install the **mysqlclient** class.

The is a subtle difference between the use of these classes to access a mysql database table row. The results data is not formatted the same way. In other words, the **mysqlclient** class results in the data formatted into a tuple array with dictionary formatted rows (see figure 1 below). The python code in main.py is designed to handle that in functions get-settings() and login(). I had to create another version on main.py which I labelled main_mysql-conn.py. So, this module supports the use of the **mysql-connector-python** class.

Figure 1

The **mysql-connector-python** class generates a table row results with a python list format. And the rows are in tuple array format. A slight change in the get-settings() function allows the calling functions to continue working normally. (See figure 2)

Figure 2