**Project 2 Python**

Our bank wants a set of functions to clean up access to the database for our data teams and managers.

We want a set of functions and classes to help do this. Use the “bank” database from Learning SQL from Beaulieu to do this

1. A connection tool that avoids having passwords in code.

This should be a function that takes as it’s input a filename. The file has only two lines in it, the username is on the first line and the password is on the second line. We will call this a “vault” file and eventually we will want to seencode it for safety. The encoding will be on the next update.

Write a python function that will

1. Take a file name as input
2. Open the file and get the username and password
3. Make a connection to the bank database, using that username and password
4. Return the connection

Note:You will need to create a “vault” file with the test user “Pat” and Pat’s password. Set up Pat as a user with access to your postgres server and the bank database.

Your function should be in single .py file, with good documentation. Run it through pylint so it as clean as you can make it.

1. Create a Jupyter notebook to test your function (and the other functions below).
2. Make sure your .py file from step 1 is in the current working directory of the notebook
3. Import your .py file with the function to make the connection
4. Set up the “vault” file in the current working directory
5. Show that connection function from part 1 works, by creating a connection and using it to

load some data table from the “bank” table.

1. Show what happens if you input an invalid “vault” file name
2. A function that accesses the transaction database and returns a python table of all the transactions for a given month.
3. The inputs should be the connection to the bank database obtained from step 1 and the desired month and year values
4. The function should return a dataframe with the transactions from the desired month and year. The account\_id should not be included
5. If the year and month are not valid, the function should print a warning and return a data

Frame with all the columns correct, but only one row which is all -1 values

Your function should be in single .py file, with good documentation. Run it through

pylint so it as clean as you can make it.

1. Import your function from step 3 above into your test jupyter notebook.

Show that the function from step 3 works, show it for both valid and invalid dates.

You will now have sections in your test RMD file that test the connection function and the transaction function. Make sure they are labeled clearly and documented.

1. Create a class that will hold the filename, the connector and the data table as variables

Your class should be created in a .py file and should be in the same working directory as your other .py files and the jupyter notebook test system.

You will need to import your files from steps 1 and 3

1. In your class \_init\_\_ function

a1.) It should take the vault file name as it’s input, this should be stored in the class

a2.) It should then read the vault file and get the username and password using your function from step 1 and get the connector to the database. The connector should be stored in the class

1. Create a method that takes in the month and the year and returns the transactions for that

Month using your function from step 3.

1. Create a method that takes in a month and a year and generates a barplot of transactions by day within that month and year, using your function from step 3 to retrieve the data. It will look sparse because the dataset is small, but create it

Your function should be in single .py file, with good documentation. Run it through

pylint so it as clean as you can make it.

1. In your test RMD, create a section to test the class from step 5.
2. Make sure it is labeled clearly
3. Create an instance of the class
4. Test it’s function for parts b and c.

**Deliverables**

Part 1- A .py file, well documented and clean

Part 2,4,6- A pdf of the Jupyter notebook showing all the tests

Part 3. A .py file, well documented and clean

Part 5. A .py file, well documented and clean