Homework 6

Due Wednesday 5/2/2018

Experimenting with Transactions - Objective

In this homework assignment you will experiment with transaction handling in Python using the MySQL connector library. This homework is worth 50 points.

Submit the following in Blackboard:

1. Your Python script
2. An image that is a screenshot of your print statements after your Python script runs to completion

<u>Requirements for all methods:</u>

1. Each method should use a try/except/finally block.
2. In each method, the connection should be closed in the finally block

Experimenting with Transactions - Part 1

Create a Python script that does the following:

1. Create a method that creates two account tables as follows:
   a. Table 1
      i. Name: “local_account”
      ii. Attribute 1: “id int”
      iii. Attribute 2: “amount decimal”
   b. Table 2
      i. Name: “remote_account”
      ii. Attribute 1: “id int”
      iii. Attribute 2: “amount decimal”
2. In that method, insert data into the tables as follows:
   a. Table 1
      i. Insert id = 1 and amount = 800.00
   b. Table 2
      i. Insert id = 2 and amount = 600.00

Experimenting with Transactions - Part 2

In the same Python script, write methods that attempt to transfer 100.00 dollars from local account with id =1 to remote account with id=2 using two separate update statements (the result of the updates should give us amounts that are equal)

1. Create a method that performs these updates with Autocommit set to “True”, print the records in both tables
2. Create a method that performs these updates with Autocommit set to “True” but raise an error after the first update statement, print the records in both tables
3. Create a method that performs these updates with Autocommit set to “False” , issue a commit after the update statements, print the records in both tables
4. Create a method that performs these updates with Autocommit set to “False”, do not issue a commit at all, print the records in both tables
5. Create a method that performs these updates with Autocommit set to “False” but raise an error after the first update statement, issue a commit after the update statements and a rollback in the except block, print the records in both tables
6. Print the default transaction isolation level used.

Methods to help you get started

```
########################################
#######GET CONNECTION
#######################################
def getConnection():
   return pymysql.connect(host='localhost',
                          user='your_username',
                          password='your_password',
                          db='your_db')
```

Methods to help you get started

```
###############################################
########CREATE TABLES
##############################################
def createTables():
   connection = getConnection()
   connection.autocommit(True)
   try:
    with connection.cursor() as cursor:
       <your code here>
   finally:
    connection.close()
```

Methods to help you get started

```
##########################################
#######DROP TABLES
#########################################
def dropTables():
   connection = getConnection()
   connection.autocommit(True)
   try:

    with connection.cursor() as cursor:
        sql="Drop table remote_account";
        cursor.execute(sql);

        sql="Drop table local_account";
        cursor.execute(sql);

   finally:
    connection.close()
```

Methods to help you get started

```
##########################################
#######Show Records
#########################################
def showRecords():
   connection = getConnection()
   try:
    with connection.cursor() as cursor:
        sql = "SELECT * from remote_account;"
        cursor.execute(sql)
        result = cursor.fetchone()
        print("REMOTE ACCOUNT: " + str(result))


    with connection.cursor() as cursor:
        sql = "SELECT * from local_account;"
        cursor.execute(sql)
        result = cursor.fetchone()
        print("LOCAL ACCOUNT: " + str(result))

   finally:
    connection.close()
```

How to raise an exception in python

```
def doSomethingAndRaiseAnError():
   Do something….
   try:
       Do something...
       raise Exception(“My Exception”)
       Do something...
   except Exception as error:
       Do something….
   finally:
       Do something...
```