**CIS 133Y: Python Programming I**

# Lab 7 - Databases

**Lab 7 is an individual programming assignment - you must complete it on your own.**

## **Purpose**

Lab 7 is a continuation of [Lab 6](https://docs.google.com/document/d/1k5gQDxsMdcXajCtGZa05tuCkb-wecxQuouYS2voTAhs/edit?usp=sharing). In Lab 7, you will complete the Database class to fetch data from cisdbss.pcc.edu (a Microsoft SQL Server database server). You will modify the program to ask the user to input search terms to filter the data coming from the database. In Lab 9, you will add a user interface to the program.

**After completing this assignment, you will be able to:**

* Write a simple program that uses N-tier architecture
* Answer questions about SQL injection attacks
* Write code that uses SQL parameter binding to prevent SQL injection attacks
* Implement a data layer that fetches rows of information from a database table
* Share a database connection over multiple database commands using class methods

## **Task**

The below instructions cover creating a program that uses Name data from the US Census Bureau. You are free to use a different type of data if you wish, as long as there are at least four properties for each object, and at least five objects. In addition, you are free to include a SQLite database of your own creation and to use that, but I must be able to run your program over here, and it must work correctly.

Your program must include the following requirements:

* Your program should consist of three files: **main.py**, **Name.py** (or whatever is appropriate to hold the data you're using if you decide not to use Name data), and **Database.py**.
* The **Database.py** file should contain a class method named **readNames()** (or something that matches the type of database you provide). This method should take at least two parameters (in addition to cls). The two parameters must be used to filter the results. For example, you might pass in year and gender and return a list of the top 20 names (represented as **Name** objects) for that combination of Year and Gender from the NAMES database. Or, you might pass in name and gender and return the NameCount for that combination of name and gender from 1915-2014.
* If you wish to use the NAMES database, you can use the following code to connect to the database:

class Database:

\_\_connection = None

@classmethod

def connect(cls):

if cls.\_\_connection is None:

server = 'tcp:cisdbss.pcc.edu'

database = 'NAMES'

username = '275student'

password = '275student'

cls.\_\_connection = pyodbc.connect(

'DRIVER={SQL Server};SERVER=' + server

+ ';DATABASE=' + database

+ ';UID=' + username + ';PWD=' + password

)

* In your **readNames()** function, call **cls.connect()**. Then, **self.\_\_connection** will contain a connection to the NAMES database. This will allow you to share one database connection across all the calls to **readNames()** (this week, we only have one call to **readNames()**, but we'll have multiple calls next week).
* The two parameters you pass into **readNames()** should be used in a WHERE clause in your SQL query. The WHERE clause should filter the data, and you should use parameter binding in your query to guard against SQL injection attacks. Regardless of the parameters you choose, you should return a list of **Name** objects with all the properties filled in (Name, Year, NameCount and Gender).
* Your **Name.py** file should include a static method called **readNames()** that takes two parameters, and passes those parameters into the **readNames()** method in **Database.py**. It should return the list of names that it gets from **readNames()** in **Database.py**, just like in the previous assignment.
* Your **main** method should ask the user to enter values for both of your search parameters, and use the **readNames()** method of **Names.py** to fetch the matching **Name** objects. Then, it should write those objects to the console just as in the previous lab.
* You should validate your inputs (**Gender** should be restricted to M or F, **Year** should be a whole number between 1915 and 2014, **Name** should be non-empty, and **NameCount** should be a whole number between 5 and 200,000. You only need to have two inputs, though). Here is a sample run of an example program. My input is in **blue**:

Select a year between 1915 and 2014: **x**

Please enter a whole number!

Selecta year between 1915 and 2014: **1900**

The year must be between 1915 and 2014!

Select a year between 1915 and 2014: **1962**

Please enter a gender (M/F): **x**

Please type M or F!

Please enter a gender (M/F): **M**

20 most popular names for M babies in 1962:

Year Name Gender Count

1962 Michael M 85046

1962 David M 81328

1962 John M 78447

1962 James M 72563

1962 Robert M 70214

1962 Mark M 53522

1962 William M 44862

1962 Richard M 39424

1962 Thomas M 36536

1962 Jeffrey M 33531

1962 Scott M 30711

1962 Steven M 30703

1962 Joseph M 30093

1962 Kevin M 28356

1962 Charles M 27590

1962 Timothy M 27085

1962 Daniel M 25553

1962 Kenneth M 25247

1962 Paul M 23804

1962 Brian M 23244

## **Criteria for Success**

* Please open and compare your work with the [grading rubric](https://docs.google.com/document/d/1vjkyMxEs1zqorR7GatK_9oVah8BPL6N7LXQmcKE-564/edit?usp=sharing) before submitting.
* Remember to follow all [Python Style Guide](https://docs.google.com/document/d/14U956Z4Q0D52ULxrqeJngaqFb7GWxLFAE_kQsHSMZos/edit?usp=sharing) rules for CIS 133Y.
* Include the required [comment header](https://docs.google.com/document/d/14U956Z4Q0D52ULxrqeJngaqFb7GWxLFAE_kQsHSMZos/edit#heading=h.rzlvsh2gsndw) at the top of your code and Include your name, lab number, a brief description of the program, a list of all input accepted from the user, a list of all variables used as output in print() statements, and any sources you used.
* Programs without comments will not be graded - you will be asked to resubmit with comments for grading.
* When you have completed the project and verified that it works correctly, zip up your project folder, including main.py, Name.py, and Database.py, upload it to the Lab 7 assignment folder, and answer the submission question posted on the assignment in D2L.

## **Academic Integrity**

As a reminder, learning to code can be difficult and frustrating and takes LOTS of practice. Our academic integrity policy for this course states that when completing programming assignments, you may:

* review course materials
* look things up in the Python documentation
* google specific questions online, such as "syntax for Python while loops"
* use the sample program as a starting idea only - you must start with a blank algorithmic design planning document and Python file

You may NOT, under any circumstances, begin a programming assignment by:

* looking for completed code which you can then modify
* starting with a copy of the actual sample program and modify
* asking a more experienced friend or family member to outline a solution for you

The only way to learn to code is to do it yourself. The assignments will build from examples during the D2L lessons, so ask for clarification on the **Ask Questions!** discussion topic if something seems confusing. If you start with code from another source and change the variable names or other content to make it look original, you will receive a zero on the assignment.

I may ask you to explain your assignment verbally. If you cannot satisfactorily explain what your code does, and answer questions about why you wrote it in a particular way, then you should also expect a zero.