Wade Moulton

10/12/2020

CS 4720 W01

Assignment 4 Databases Report

Weblink: <http://studentweb.kennesaw.edu/~jmoulto2/>

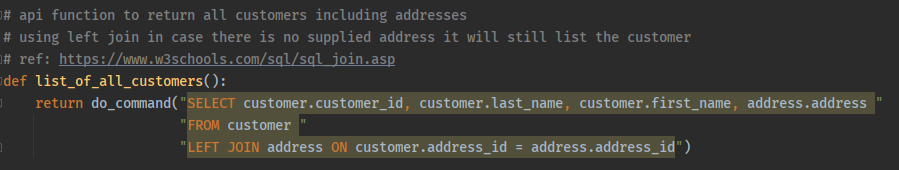
**Assignment Description**

Assignment 4 Databases focuses on implementing a database using Sqlite3. The objectives will be to send a query, retrieve data, and display that data. The important concepts covered include SQL injection attacks and possible solutions as well as creating an API. All source code can be found at my weblink and at the end of this report.

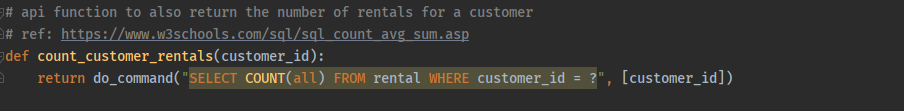
**Assignment Objectives**

The objective of this assignment is to create two new reports. The first report will return a customer id, last name, first name, and address. The second objective also adds the number of rentals for a customer.

For the first report, a new API function is used to retrieve the customer id, last name and first name. For the address, a join is needed since the addresses are stored in a separate location. The join used for this function was the left join. This ensures that even if no address exists, the customer will still be displayed.

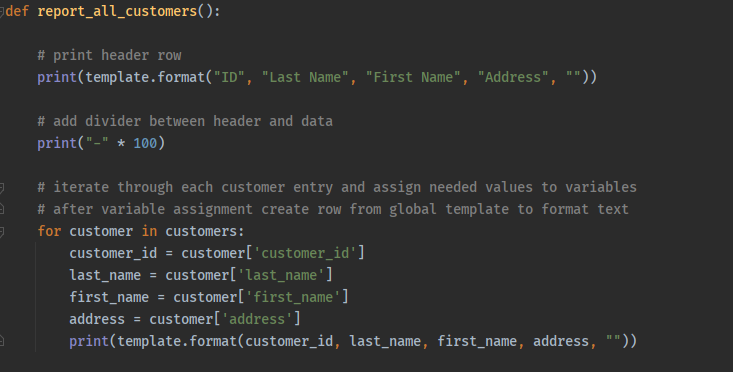


For the second report, a new function is needed to get the count of all rentals for a customer id. In this function one of the solutions to SQL injections is implemented using the ‘?’.

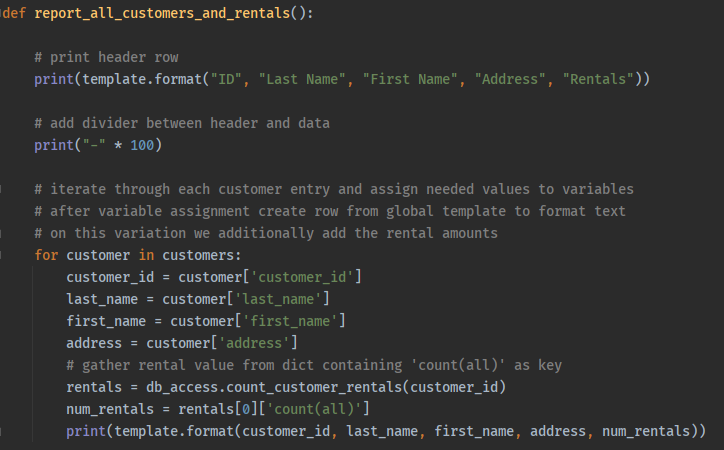


To retrieve and display this data I created the report.py file from the example given. In my implementation I did change the template styling, added two functions to separate the two different reports, and added the code for both customer data reports.

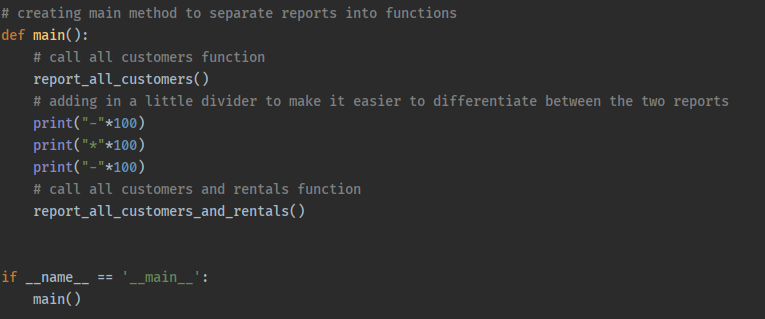
Here is a sample for report 1 from the exercise in the lecture slide:



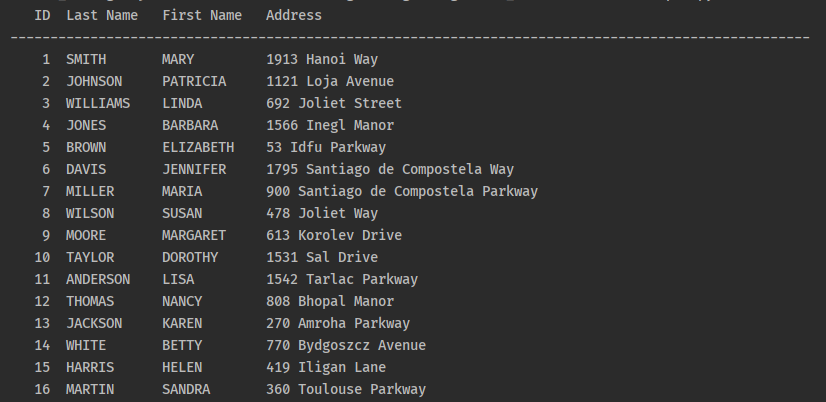
Here is a sample for report 2 from the exercise in the lecture slide:



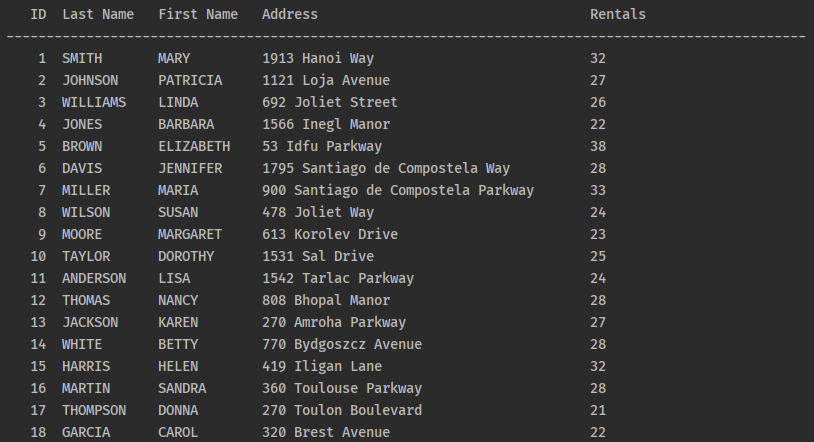
Lastly, I created a main method which included both report function calls as well as a divider for the output:



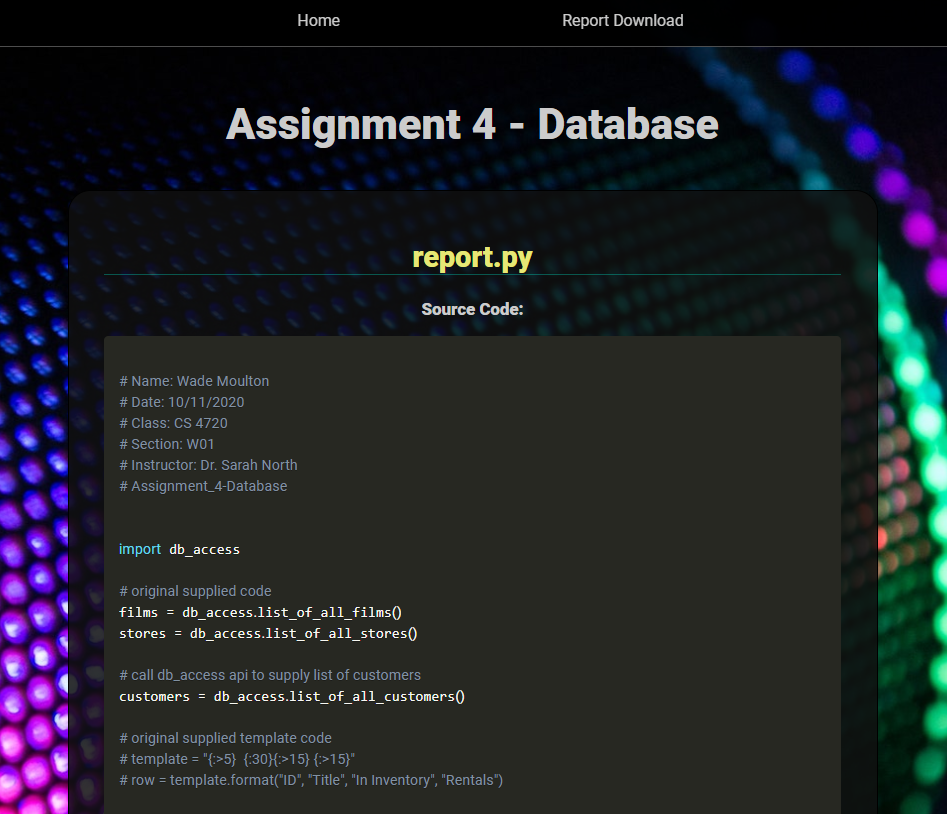
Here is the sample output for report 1:



Here is the sample output for report 2:



I did also implement these assignments into my student web page with both the full source code and outputs as well as download links at the top to download this report.



**Source Code**

# db.py

# Name: Wade Moulton  
# Date: 10/11/2020  
# Class: CS 4720  
# Section: W01  
# Instructor: Dr. Sarah North  
# Assignment\_4-Database  
  
import sqlite3  
from os.path import join, split  
  
  
# supplied function from lecture slides  
def dictionary\_factory(cursor, row):  
 col\_names = [d[0].lower() for d in cursor.description]  
 return dict(zip(col\_names, row))  
  
  
# supplied function from lecture slides  
def getConnection():  
 this\_dir = split(\_\_file\_\_)[0]  
 fname = join(this\_dir, 'sqlite-sakila.sq')  
 conn = sqlite3.connect(fname)  
 conn.row\_factory = dictionary\_factory  
 return conn  
  
  
# supplied function from lecture slides  
def do\_command(cmd, args=[]):  
 try:  
 conn = getConnection()  
 crs = conn.cursor()  
 crs.execute(cmd, args)  
 return crs.fetchall()  
 finally:  
 conn.close()

# db\_access.py

# Name: Wade Moulton  
# Date: 10/11/2020  
# Class: CS 4720  
# Section: W01  
# Instructor: Dr. Sarah North  
# Assignment\_4-Database  
  
  
from db import do\_command  
  
  
# supplied function from lecture slides  
def list\_of\_all\_stores():  
 return do\_command("select \* from store")  
  
  
# supplied function from lecture slides  
def list\_of\_all\_films():  
 return do\_command("select \* from film order by title")  
  
  
# supplied function from lecture slides  
def list\_of\_all\_inventory():  
 return do\_command("select \* from inventory")  
  
  
# supplied function from lecture slides  
def inventory\_for\_film(film\_id):  
 return do\_command("select \* from inventory where film\_id = ?", [film\_id])  
  
  
# supplied function from lecture slides  
def inventory\_for\_film\_for\_store(film\_id, store\_id):  
 return do\_command("select \* from inventory where film\_id = ? and store\_id = ?" [film\_id, store\_id])  
  
  
# supplied function from lecture slides  
def count\_rentals\_for\_film(film\_id):  
 invent = inventory\_for\_film(film\_id)  
 rentals = 0  
 for inv in invent:  
 rnt = do\_command("select count(all) as cnt from rental where inventory\_id = ?", [inv['inventory\_id']])  
 rentals += rnt[0]['cnt']  
 return rentals  
  
  
# api function to return all customers including addresses  
# using left join in case there is no supplied address it will still list the customer  
# ref: https://www.w3schools.com/sql/sql\_join.asp  
def list\_of\_all\_customers():  
 return do\_command("SELECT customer.customer\_id, customer.last\_name, customer.first\_name, address.address "  
 "FROM customer "  
 "LEFT JOIN address ON customer.address\_id = address.address\_id")  
  
  
# api function to also return the number of rentals for a customer  
# ref: https://www.w3schools.com/sql/sql\_count\_avg\_sum.asp  
def count\_customer\_rentals(customer\_id):  
 return do\_command("SELECT COUNT(all) FROM rental WHERE customer\_id = ?", [customer\_id])

# report.py

# Name: Wade Moulton  
# Date: 10/11/2020  
# Class: CS 4720  
# Section: W01  
# Instructor: Dr. Sarah North  
# Assignment\_4-Database  
  
  
import db\_access  
  
# original supplied code  
films = db\_access.list\_of\_all\_films()  
stores = db\_access.list\_of\_all\_stores()  
  
# call db\_access api to supply list of customers  
customers = db\_access.list\_of\_all\_customers()  
  
# original supplied template code  
# template = "{:>5} {:30}{:>15} {:>15}"  
# row = template.format("ID", "Title", "In Inventory", "Rentals")  
  
# creating my own template for row formatting  
template = "{:>5} {:<12}{:<12} {:<40} {:<15}"  
  
  
def report\_all\_customers():  
  
 # print header row  
 print(template.format("ID", "Last Name", "First Name", "Address", ""))  
  
 # add divider between header and data  
 print("-" \* 100)  
  
 # iterate through each customer entry and assign needed values to variables  
 # after variable assignment create row from global template to format text  
 for customer in customers:  
 customer\_id = customer['customer\_id']  
 last\_name = customer['last\_name']  
 first\_name = customer['first\_name']  
 address = customer['address']  
 print(template.format(customer\_id, last\_name, first\_name, address, ""))  
  
  
def report\_all\_customers\_and\_rentals():  
  
 # print header row  
 print(template.format("ID", "Last Name", "First Name", "Address", "Rentals"))  
  
 # add divider between header and data  
 print("-" \* 100)  
  
 # iterate through each customer entry and assign needed values to variables  
 # after variable assignment create row from global template to format text  
 # on this variation we additionally add the rental amounts  
 for customer in customers:  
 customer\_id = customer['customer\_id']  
 last\_name = customer['last\_name']  
 first\_name = customer['first\_name']  
 address = customer['address']  
 # gather rental value from dict containing 'count(all)' as key  
 rentals = db\_access.count\_customer\_rentals(customer\_id)  
 num\_rentals = rentals[0]['count(all)']  
 print(template.format(customer\_id, last\_name, first\_name, address, num\_rentals))  
  
  
# creating main method to separate reports into functions  
def main():  
 # call all customers function  
 report\_all\_customers()  
 # adding in a little divider to make it easier to differentiate between the two reports  
 print("-"\*100)  
 print("\*"\*100)  
 print("-"\*100)  
 # call all customers and rentals function  
 report\_all\_customers\_and\_rentals()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()