**Python Test: Instructions:**

Please enter your answers below to each question.

Create a Python script as needed for each question that requires one; then zip and attach the Python script(s) and output file to your email response.

**1)** Is Python interpreted or compiled?

When using CPython, Python source code is compiled to bytecode, which is then interpreted by the Python bytecode interpreter.

**2)** Write down the steps and commands on how to create a new Python3 virtual environment named py3vDev.

In a Unix-like shell, type: “python3 -m venv py3vDev”

**3)** Describe how you would write a short program which connects to a database of your choice, retrieves records from a table named “Customers” with the Last Name “Johnson”, and writes out the First Name, Last Name, and Age to a text file named “Johnson.txt”. If you prefer, you may write Python code directly, or a mixture of description and code.

import sqlite3

connection = sqlite3.connect("musicreportsdata.db")

cursor = connection.cursor()

result = cursor.execute("SELECT first\_name, last\_name, age FROM Customers WHERE last\_name='Johnson'")

rows = result.fetchall()

with open("results.txt", "w") as outfile:

for row in rows:

outfile.write(f"{row[0]} {row[1]}, age {row[2]}\n")

**4)** Using Python, create a function called “myfuncname” that takes in a name and prints out the following: 'Hello, your name is: ' followed by the name.

def myfuncname(name: str) -> None:

print(f"Hello, your name is: {name}")

**5)** Using Python create simple example program to connect **AWS S3 bucket**, get list of objects and save list of object into local drive file with name list\_of\_objects.log.

import boto3

def s3\_ls(bucket\_name: str) -> list[str]:

s3 = boto3.resource("s3")

bucket = s3.Bucket(bucket\_name)

items = []

for obj in bucket.objects.all():

items.append(obj.key)

return items

filenames = s3\_ls("paulbissex.com")

with open("list\_of\_objects.log", "w") as logfile:

logfile.writelines(name + "\n" for name in filenames)

6) Using Python, create a function called “myfuncname” that accepts a single argument for the filename path to the CSV file included in this test. The script will create a file that contains streamable **line-delimited** JSON using the contents of the CSV file. See below for further instructions:

* Expected filename of the file created from the Python script will be based on the CSV filename, i.e. basicnames.csv will produce basicnames.json. Show this in your code and should not be hardcoded.
* basicnames.csv has 10000 lines including the header. Therefore, the file created from your code should contain 9999 lines.
* To illustrate, check the sample CSV and resulting output below. Pay attention to the expected format of each JSON line in the resulting file.

From a CSV that has 4 lines, a header and 3 rows:

nconst,primaryName,birthYear,deathYear,primaryProfession,knownForTitles

nm0000001,Fred Astaire,1899,1987,"soundtrack,actor,miscellaneous","tt0072308,tt0043044,tt0050419,tt0053137"

nm0000002,Lauren Bacall,1924,2014,"actress,soundtrack","tt0071877,tt0038355,tt0117057,tt0037382"

nm0000003,Brigitte Bardot,1934,\N,"actress,soundtrack,producer","tt0057345,tt0059956,tt0049189,tt0054452"

To 3 lines of JSON, the contents of the resulting file should look like below:

{"nconst":"nm0000001","primaryName":"Fred Astaire","birthYear":1899,"deathYear":1987,"primaryProfession":"soundtrack,actor,miscellaneous","knownForTitles":"tt0072308,tt0043044,tt0050419,tt0053137"}  
{"nconst":"nm0000002","primaryName":"Lauren Bacall","birthYear":1924,"deathYear":2014,"primaryProfession":"actress,soundtrack","knownForTitles":"tt0071877,tt0038355,tt0117057,tt0037382"}  
{"nconst":"nm0000003","primaryName":"Brigitte Bardot","birthYear":1934,"deathYear":null,"primaryProfession":"actress,soundtrack,producer","knownForTitles":"tt0057345,tt0059956,tt0049189,tt0054452"}

import argparse

import csv

import json

def normalize\_years(row):

"""Years get parsed by csv module as strings, but should be ints. Non-numeric values should become None."""

for key in row:

if key.endswith("Year"):

row[key] = int(row[key]) if row[key].isnumeric() else None

return row

def myfuncname(path: str) -> None:

output\_filename = path.replace("csv", "json")

output\_text = ""

with open(path) as csv\_file:

reader = csv.DictReader(csv\_file)

for row in reader:

row = normalize\_years(row)

output\_text += json.dumps(row) + "\n"

with open(output\_filename, "w") as output\_file:

output\_file.write(output\_text)

if \_\_name\_\_ == "\_\_main\_\_":

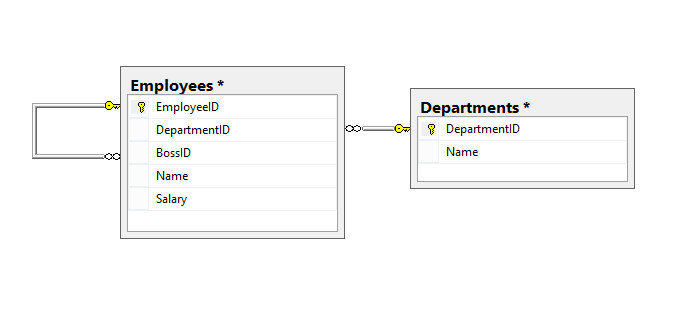
parser = argparse.ArgumentParser(description="Produce JSON from CSV")

parser.add\_argument("csv\_path", nargs=1, help="Path to CSV file")

args = parser.parse\_args()

myfuncname(args.csv\_path[0])

**SQL part Questions —** Please prepare SQL query for each question using following tables structure



1. Look for all employees with a name that includes the string 'Johnson'.

SELECT \* FROM Employees WHERE Name LIKE "%Johnson%"

1. Write a query that returns a Boolean flag if their salary is greater than or equal to $5,000.

SELECT CASE WHEN EXISTS (SELECT \* FROM Employees WHERE Name = "Pat" AND Salary > 5000) THEN TRUE ELSE FALSE END

1. Add a column to the departments table that indicates whether it is active.

ALTER TABLE Departments ADD Active BOOLEAN

1. Remove all employees that do not have a salary and their department is inactive.

DELETE FROM Employees WHERE Salary is NULL AND Employees.DepartmentID IN (SELECT DepartmentID FROM Departments WHERE Active = FALSE)

1. Select employees names who have a bigger salary than their boss

SELECT t1.Name FROM Employees AS t1 JOIN Employees AS t2 ON t1.BossID = t2.EmployeeID WHERE t1.Salary > t2.Salary

1. Return employees who have the biggest salary in their departments

SELECT Name, MAX(Salary) FROM Employees GROUP BY DepartmentID

1. Return departments that have less than 3 people

SELECT DepartmentID FROM Employees GROUP BY DepartmentID HAVING COUNT() < 3

1. Return all departments along with the number of people there

SELECT DepartmentID, count() FROM Employees GROUP BY DepartmentID

1. Return employees that don't have a boss in the same department

SELECT t1.Name FROM Employees AS t1 JOIN Employees AS t2 ON t1.BossID = t2.EmployeeID WHERE t1.DepartmentID <> t2.DepartmentID

1. Return all departments along with the total salary there

SELECT DepartmentID, SUM(Salary) FROM Employees GROUP BY DepartmentID