labassignment6

August 1, 2024

1 Lab Assignment 6: Creating and Connecting to Databases

1.1 DS 6001: Practice and Application of Data Science

1.1.1 Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. Format your answers in a Jupyter notebook. To receive full credit, make sure you address every part of the problem, and make sure your document is formatted in a clean and professional way.

Please note: you will not be able to use Rivanna for this lab as Rivanna is not set up to work with Docker or with Databases. If you need help getting your local system running, please let me know.

1.1.2 Problem 0 [No points, no need to write anything here for any of the following parts, but do it anyway!]

Databases require a lot of external software. The good news is that there are excellent free and open source options to do very advanced work with databases. The bad news is that each piece of additional software comes with its own complications. This problem will guide you through the installation steps for the software you need to run database systems on your computer, document those databases, and connect to them through Python with (fingers crossed) as few problems as possible.

Part a Use `pip` to install the following Python packages on your system:

```
mysql-connector-python
psycopg
pymongo
sqlalchemy
wget
```

Part b With the exception of SQlite, database systems run as external software that must be installed and run on your computer. To make the installation steps easier, you will need some configuration files that I wrote and saved in a GitHub repository. Open your terminal and use the `cd` command to navigate to the folder in your computer where you want to work. Then type

```
git clone https://github.com/jkropko/ds6001databases
```

If this command works, it will create a new directory within your current folder called "ds6001databases".

- Check that this folder exists and contains the following files: LICENSE, README.md, compose.yaml, db_tests.ipynb, and requirements.txt
- Save the notebook file you will be using for your Lab 6 work inside the "ds6001databases" folder

Part c We will be using a system called Docker to work with databases. Docker is the most commonly used platform for working with **containers**. While we will not be delving into the topic of containerization in this course, a container is space in your computer's memory that is set apart from the rest of your computer. We can act as if the container is an entirely new computer, and inside the container we can change the operating system and install other software external to Python, such as database management systems. We can use a container to run Windows on a Mac, or vice versa, or Linux on any system. By far the easiest way to run MySQL, PostgreSQL, and MongoDB is through Docker containers.

You will need to install Docker Desktop on your computer. Go to https://www.docker.com/products/docker-desktop/ and click on the Download button, making sure the operating system listed matches the operating system of your computer.

Once Docker Desktop is installed, find the Docker Desktop program on your computer and run it.

To confirm that Docker Desktop is running, open a terminal and type `docker help`. If you see documentation that begins

```
Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers
```

then you are all set. If you see an error that Docker is not found, then the Docker Desktop client was not installed properly, so you should try downloading and installing it from the website again. If you receive a message that the Docker daemon is not running, then Docker is installed but is not running. Find the Docker Desktop executable on your computer and click it to get Docker running.

Part d Inside your "ds6001databases" folder, create a .env file. On a Mac, type `touch .env` then `open .env` to create and open the file. On Windows, open a new file on Notepad and go to "Save As", then save it in your "ds6001databases" folder – make sure to set "File As Type" to "All files" and name the file ".env".

Inside the .env file you need to choose passwords for the MySQL, PostgreSQL, and MongoDB databases, so type

```
MYSQL_ROOT_PASSWORD=redlobstercheddarbiscuits
POSTGRES_PASSWORD=outbackbloominonion
MONGO_INITDB_ROOT_PASSWORD=olivegardenunlimitedbreadsticks
MONGO_INITDB_ROOT_USERNAME=mongo
mongo_init_db = mongodb
MYSQL_DATABASE=mysql
```

Change the passwords on the first three lines to whatever you want, but DON'T USE THE @ SYMBOL as that will cause problems. Leave the fourth, fifth, and sixth lines alone, as well as the names of each environmental variable.

Part e In the terminal, make sure you are in the "ds6001databases" folder (you can check by typing `pwd`. If not, then use `cd` to navigate to the "ds6001databases" folder). Then type

```
docker compose up
```

This command launches all of the databases. If successful, you will see a long stream of output with messages that begin `ds6001databases-postgres-1`, `ds6001databases-mysql-1`, and `ds6001databases-mongo-1`. If not, we will need to debug together, but the issue likely has to do with something preventing parts a, b, c, or d from being completed successfully.

Part f To confirm that the databases are running on your system, open the "db_tests.ipynb" notebook file, which should be saved in you "ds6001databases" folder. Run everything in this notebook and make sure there are no errors.

Part g In addition to the databases, we will be using dbdocs.io to create documentation for our databases and post them online with a stable URL. But to get dbdocs running, you first need to install NodeJS on your computer: https://nodejs.org/en

Then to install dbdocs, follow the instructions here: https://dbdocs.io/docs

Part h Finally, create a notebook inside your "ds6001databases" folder for your work on this lab. Import the following libraries, and load the `.env` file where you store your passwords.

```
[ ]: import numpy as np
     import pandas as pd
     import wget
     import sqlite3
     import sqlalchemy
     import requests
     import json
     import os
     import sys
     import dotenv
     import mysql.connector
     import psycopg
     import pymongo
     from sqlalchemy import create_engine
     from dotenv import load_dotenv

     load_dotenv()
     dotenv.load_dotenv()
     POSTGRES_PASSWORD = os.getenv('POSTGRES_PASSWORD')
     MONGO_INITDB_ROOT_USERNAME = os.getenv('MONGO_INITDB_ROOT_USERNAME')
     MONGO_INITDB_ROOT_PASSWORD = os.getenv('MONGO_INITDB_ROOT_PASSWORD')
     mongo_init_db = os.getenv('mongo_init_db')
     MYSQL_ROOT_PASSWORD = os.getenv('MYSQL_ROOT_PASSWORD')
```

1.1.3 Problem 1

This problem requires you to create Markdown tables

To create a table in a markdown cell, I recommend using the markdown table generator here: https://www.tablesgenerator.com/markdown_tables. This interface allows you to choose the number of rows and columns, fill in those rows and colums, and push the “generate” button. The website will display markdown table code that looks like:

```
| Day       | Temp | Rain |
|-----------|------|------|
| Monday    | 74   | No   |
| Tuesday   | 58   | Yes  |
| Wednesday | 76   | No   |
```

Copy the markdown code and paste it into a markdown cell in your notebook. Markdown will read the code and display a table that looks like this:

Day	Temp	Rain
Monday	74	No
Tuesday	58	Yes
Wednesday	76	No

Suppose that we have (fake) data on people who were hospitalized and received at least one prescription for a medication. Here are ten records in the data:

(If this table gets cut off in the PDF, please look at the .ipynb notebook file on the module 6 page on Canvas)

patient_name	date_of_birth	prescribed_drug	prior_conditions	patient_sex	patient_insurance	drug_maker	drug_cost	attending_physician	AP_medical_school	AP_years_experience	hospital	hospital_location
Nkemdirim Arendonk	2/21/1962	Amoxicillin	[Pneumonia, Diabetes]	M	Aetna	USAntibiotics	14.62	Earnest Caro	University of California (Irvine)	14	UPMC Presbyterian Shadyside	Pittsburgh, PA
Nkemdirim Arendonk	2/21/1962	Micronase	[Pneumonia, Diabetes]	M	Aetna	Pfizer	20.55	Earnest Caro	University of California (Irvine)	14	UPMC Presbyterian Shadyside	Pittsburgh, PA
Raniero Coumans	8/15/1990	Zosyn	[Appendicitis, Crohn’s disease]	M	Cigna	Baxter International Inc	394.00	Pamela English	University of Michigan	29	Northwestern Memorial Hospital	Chicago, IL

patient_name	date_of_birth	prescribed_drug	prior_conditions	patient_sex	patient_insurance	drug_maker	drug_cost	attending_physician	AP_medical_school	AP_years_experience	hospital	hospital_location
Raniero Coumans	8/15/1990	Humira	[Appendicitis, Crohn's disease]	M	Cigna	Abbvie	7000.00	Pamela English	University of Michigan	29	Northwestern Memorial Hospital	Chicago, IL
Mizuki Debenham	3/12/1977	Inlyta	[Kidney Cancer]	F	Kaiser Permanente	Pfizer	21644.00	Lewis Conti	North Carolina State University	8	Houston Methodist Hospital	Houston, TX
Zoë De Witt	11/23/1947	Atenolol	[Cardiomyopathy, Diabetes, Sciatica]	F	Medicare	Mylan Pharmaceuticals	10.58	Theresa Dahlmann	Lake Erie College of Medicine	17	Mount Sinai Hospital	New York, NY
Zoë De Witt	11/23/1947	Micronase	[Cardiomyopathy, Diabetes, Sciatica]	F	Medicare	Pfizer	20.55	Theresa Dahlmann	Lake Erie College of Medicine	17	Mount Sinai Hospital	New York, NY
Zoë De Witt	11/23/1947	Demerol	[Cardiomyopathy, Diabetes, Sciatica]	F	Medicare	Pfizer	37.50	Theresa Dahlmann	Lake Erie College of Medicine	17	Mount Sinai Hospital	New York, NY
Bonnie Hooper	7/4/1951	Xeloda	[Pancreatic Cancer, Sciatica]	F	Blue Cross Blue Shield	Genentech	860.00	Steven Garbutt	Ohio State University	36	UCSF Medical Center	San Francisco, CA
Bonnie Hooper	7/4/1951	Demerol	[Pancreatic Cancer, Sciatica]	F	Blue Cross Blue Shield	Pfizer	37.50	Steven Garbutt	Ohio State University	36	UCSF Medical Center	San Francisco, CA

The columns in this dataset are:

- **patient_name**: The patient's name
- **date_of_birth**: The patient's date of birth
- **prescribed_drug**: The brand name of the medication that patient has been prescribed
- **prior_conditions**: A list of the conditions that the patient had been diagnosed with prior to the patient's hospitalization
- **patient_sex**: The patient's sex
- **patient_insurance**: The company responsible for the patient's health insurance coverage
- **drug_maker**: The company that manufactures the prescribed drug
- **drug_cost**: The cost of the prescribed drug

- **attending_physician**: The name of the attending physician for the patient
- **AP_medschool**: The name of the school where the attending physician got a medical degree
- **AP_years_experience**: The attending physician's number of years of experience post-residency
- **hospital**: The hospital where the attending physicial is employed
- **hospital_location**: The location of the hospital

For this problem, assume that

1. No two rows in this table share both the same patient and the same prescribed drug.
2. Some patients in the data share the same name, but no two patients in the data share the same name and date of birth.
3. No two different drugs share the same brand name.
4. No two attending physicians have the same name, and every attending physician is employed at only one hospital.
5. No two hospitals share the same name, and every hospital exists at only one location.
6. Each patient has only one attending physician. (In real-world applications we may want to design a database that allows for multiple hospitalizations for some patients, but here we'll keep it simpler by assuming each patient has one hospitalization with one attending physician.)

Part a Rearrange the data into a group of data tables that together meet the requirements of first normal form. [2 points]

patient_id	patient_name	date_of_birth	patient_sex	patient_insurance
1	Nkemdilim Arendonk	2/21/1962	M	Aetna
2	Raniero Coumans	8/15/1990	M	Cigna
3	Mizuki Debenham	3/12/1977	F	Kaiser Permanente
4	Zoë De Witt	11/23/1947	F	Medicare
5	Bonnie Hooper	7/4/1951	F	Blue Cross Blue Shield

drug_id	prescribed_drug	drug_maker	drug_cost
1	Amoxil	USAntibiotics	14.62
2	Micronase	Pfizer	20.55
3	Zosyn	Baxter International Inc	394.00
4	Humira	Abbvie	7000.00
5	Inlyta	Pfizer	21644.00
6	Atenolol	Mylan Pharmaceuticals	10.58
7	Demerol	Pfizer	37.50
8	Xeloda	Genentech	860.00

condition_id	condition
1	Pneumonia
2	Diabetes
3	Appendicitis
4	Crohn’s disease
5	Kidney Cancer
6	Cardiomyopathy
7	Sciatica
8	Pancreatic Cancer

patient_id	condition_id
1	1
1	2
2	3
2	4
3	5
4	6
4	2
4	7
5	8
5	7

physician_id	attending_physician	AP_medschool	AP_years_experience
1	Earnest Caro	University of California (Irvine)	14
2	Pamela English	University of Michigan	29
3	Lewis Conti	North Carolina State University	8
4	Theresa Dahlmans	Lake Erie College of Medicine	17
5	Steven Garbutt	Ohio State University	36

hospital_id	hospital	hospital_location
1	UPMC Presbyterian Shadyside	Pittsburgh, PA
2	Northwestern Memorial Hospital	Chicago, IL
3	Houston Methodist Hospital	Houston, TX
4	Mount Sinai Hospital	New York, NY
5	UCSF Medical Center	San Francisco, CA

prescription_id	patient_id	drug_id	physician_id	hospital_id
1	1	1	1	1
2	1	2	1	1
3	2	3	2	2

prescription_id	patient_id	drug_id	physician_id	hospital_id
4	2	4	2	2
5	3	5	3	3
6	4	6	4	4
7	4	2	4	4
8	4	7	4	4
9	5	8	5	5
10	5	7	5	5

Part b Rearrange the data on the five patients into a group of data tables that together meet the requirements of second normal form. [2 points]

patient_id	patient_name	date_of_birth	patient_sex	patient_insurance
1	Nkemdilim Arendonk	1962-02-21	M	Aetna
2	Raniero Coumans	1990-08-15	M	Cigna
3	Mizuki Debenham	1977-03-12	F	Kaiser Permanente
4	Zoë De Witt	1947-11-23	F	Medicare
5	Bonnie Hooper	1951-07-04	F	Blue Cross Blue Shield

drug_id	prescribed_drug	drug_maker	drug_cost
1	Amoxil	USAntibiotics	14.62
2	Micronase	Pfizer	20.55
3	Zosyn	Baxter International Inc	394.00
4	Humira	Abbvie	7000.00
5	Inlyta	Pfizer	21644.00
6	Atenolol	Mylan Pharmaceuticals	10.58
7	Demerol	Pfizer	37.50
8	Xeloda	Genentech	860.00

condition_id	condition
1	Pneumonia
2	Diabetes
3	Appendicitis
4	Crohn’s disease
5	Kidney Cancer
6	Cardiomyopathy
7	Sciatica
8	Pancreatic Cancer

patient_id	condition_id
1	1
1	2
2	3
2	4
3	5
4	6
4	2
4	7
5	8
5	7

physician_id	attending_physician	AP_medschool	AP_years_experience
1	Earnest Caro	University of California (Irvine)	14
2	Pamela English	University of Michigan	29
3	Lewis Conti	North Carolina State University	8
4	Theresa Dahlmans	Lake Erie College of Medicine	17
5	Steven Garbutt	Ohio State University	36

hospital_id	hospital	hospital_location
1	UPMC Presbyterian Shadyside	Pittsburgh, PA
2	Northwestern Memorial Hospital	Chicago, IL
3	Houston Methodist Hospital	Houston, TX
4	Mount Sinai Hospital	New York, NY
5	UCSF Medical Center	San Francisco, CA

prescription_id	patient_id	drug_id	physician_id	hospital_id
1	1	1	1	1
2	1	2	1	1
3	2	3	2	2
4	2	4	2	2
5	3	5	3	3
6	4	6	4	4
7	4	2	4	4
8	4	7	4	4
9	5	8	5	5
10	5	7	5	5

Part c Rearrange the data into a group of data tables that together meet the requirements of third normal form. [2 points]

patient_id	patient_name	date_of_birth	patient_sex	patient_insurance
1	Nkemdilim Arendonk	1962-02-21	M	Aetna
2	Raniero Coumans	1990-08-15	M	Cigna
3	Mizuki Debenham	1977-03-12	F	Kaiser Permanente
4	Zoë De Witt	1947-11-23	F	Medicare
5	Bonnie Hooper	1951-07-04	F	Blue Cross Blue Shield

drug_id	prescribed_drug	drug_maker	drug_cost
1	Amoxil	USAntibiotics	14.62
2	Micronase	Pfizer	20.55
3	Zosyn	Baxter International Inc	394.00
4	Humira	Abbvie	7000.00
5	Inlyta	Pfizer	21644.00
6	Atenolol	Mylan Pharmaceuticals	10.58
7	Demerol	Pfizer	37.50
8	Xeloda	Genentech	860.00

condition_id	condition
1	Pneumonia
2	Diabetes
3	Appendicitis
4	Crohn's disease
5	Kidney Cancer
6	Cardiomyopathy
7	Sciatica
8	Pancreatic Cancer

patient_id	condition_id
1	1
1	2
2	3
2	4
3	5
4	6
4	2
4	7
5	8
5	7

physician_id	attending_physician	AP_medschool_id
1	Earnest Caro	1
2	Pamela English	2
3	Lewis Conti	3
4	Theresa Dahlmans	4
5	Steven Garbutt	5

hospital_id	hospital	hospital_location
1	UPMC Presbyterian Shadyside	Pittsburgh, PA
2	Northwestern Memorial Hospital	Chicago, IL
3	Houston Methodist Hospital	Houston, TX
4	Mount Sinai Hospital	New York, NY
5	UCSF Medical Center	San Francisco, CA

prescription_id	patient_id	drug_id	physician_id	hospital_id
1	1	1	1	1
2	1	2	1	1
3	2	3	2	2
4	2	4	2	2
5	3	5	3	3
6	4	6	4	4
7	4	2	4	4
8	4	7	4	4
9	5	8	5	5
10	5	7	5	5

1.1.4 Problem 2

For this problem, create ER diagrams of the database you created in problem 1, part c using https://dbdocs.io/. Make sure you install DBDocs on your system by following these instructions: https://dbdocs.io/docs

Part a Write code using the database markup language (DBML) that represents all of the tables in this database and the connections between the tables. Paste your DBML code in a markdown cell in your notebook, contained within three backticks to begin and end the code snippet, as shown in the cell below.

Two good resources to help you:

1. The example on the Getting Started page on dbdocs.io: https://dbdocs.io/docs
2. The full syntax guide for DBML: https://dbml.dbdiagram.io/docs/#project-definition

A few notes: * Make sure to specify the data type for each column in each table. Use varchar for strings/text, int for integers, and float for numeric data with decimals. * You will probably find it useful to alias each table with one or two letters, such as: Table PRESCRIPTIONS as PR. That will allow you to use PR to refer to the PRESCRIPTIONS table, for example, in the Reference statements to link tables together. * Use the syntax [pk] after a column name and data type to designate the columns that are primary keys in each table. * To draw the lines linking one table to another, use the Ref: syntax. * If many rows from the left table match to one row in the right table, use the "many to one" symbol > * If one row from the left table matches to many rows in the right table, use the "one to many" symbol < * If one row from the left table matches to one row in the right table, use the "one to one" symbol - * If many rows from the left table match to many rows in the right table, use the "many to many" symbol <>

[2 points]

```
Project Lab6 {
  database_type: 'PostgreSQL'
  Note: '''
    # Lab6 Database
    **database schema for Lab6 project.**
  '''
}

Table patients {
  patient_id int [pk]
  patient_name varchar
  date_of_birth date
  patient_sex varchar
  patient_insurance varchar
}

Table medications {
  drug_id int [pk]
  prescribed_drug varchar
  drug_maker varchar
  drug_cost float
}

Table conditions {
  condition_id int [pk]
  condition varchar
}

Table patient_conditions {
  patient_id int
  condition_id int
}

Table physicians {
  physician_id int [pk]
  attending_physician varchar
  AP_medschool_id int
}
```

```
Table medschools {
  medschool_id int [pk]
  AP_medschool varchar
  AP_years_experience int
}

Table hospitals {
  hospital_id int [pk]
  hospital varchar
  hospital_location varchar
}

Table prescriptions {
  prescription_id int [pk]
  patient_id int
  drug_id int
  physician_id int
  hospital_id int
}

Ref: patient_conditions.patient_id > patients.patient_id
Ref: patient_conditions.condition_id > conditions.condition_id
Ref: physicians.AP_medschool_id > medschools.medschool_id
Ref: prescriptions.patient_id > patients.patient_id
Ref: prescriptions.drug_id > medications.drug_id
Ref: prescriptions.physician_id > physicians.physician_id
Ref: prescriptions.hospital_id > hospitals.hospital_id
```

1.1.5 Part b

Use the instructions on DBDocs.io (https://dbdocs.io/docs) to create a website for your ER diagram. Type the URL for your website in a markdown cell here. [1 point]

My example is here: https://dbdocs.io/jkropko/Lab6?view=relationships

ER diagram for Lab6: https://dbdocs.io/qaisme100/Lab6

1.1.6 Problem 3

For this problem, you will download the individual CSV files that comprise a relational database on album reviews from Pitchfork Magazine, collected via webscraping by Nolan B. Conaway, and use them to initialize local databases using SQlite, MySQL, and PostgreSQL.

The following code of code will download the CSV files. Please run this as is:

[ ]:
```
url = "https://github.com/nolanbconaway/pitchfork-data/raw/master/pitchfork.db"
pfork = wget.download(url)
pitchfork = sqlite3.connect(pfork)
for t in ['artists','content','genres','labels','reviews','years']:
    datatable = pd.read_sql_query("SELECT * FROM {tab}".format(tab=t),␣
↪pitchfork)
    datatable.to_csv("{tab}.csv".format(tab=t))
```

Note: this code downloaded a SQlite database and extracted the tables, saving each one as a CSV. That seems backwards, as the purpose of this exercise is to create databases. But the point here is to practice creating databases from individual data frames. Next we load the CSVs to create the data frames in Python:

[ ]:
```
reviews = pd.read_csv("reviews.csv")
artists = pd.read_csv("artists.csv")
content = pd.read_csv("content.csv")
genres = pd.read_csv("genres.csv")
labels = pd.read_csv("labels.csv")
years = pd.read_csv("years.csv")
```

Part a Initialize a new database using SQlite and the `sqlite3` library. Add the six dataframes to this database. Then issue the following query to the database

`SELECT title, artist, score FROM reviews WHERE score=10`

using two methods: first, using the `.cursor()` method, and second using `pd.read_sql_query()`. Finally, commit your changes to the database and close the database. (If you get a warning about spaces in the column names, feel free to ignore it this time.) [2 points]

[ ]:
```
import sqlite3
import pandas as pd
import mysql.connector
import wget

new_db = sqlite3.connect('pitchfork_a.db')

reviews.to_sql('reviews', new_db, if_exists='replace', index=False)
artists.to_sql('artists', new_db, if_exists='replace', index=False)
content.to_sql('content', new_db, if_exists='replace', index=False)
genres.to_sql('genres', new_db, if_exists='replace', index=False)
labels.to_sql('labels', new_db, if_exists='replace', index=False)
years.to_sql('years', new_db, if_exists='replace', index=False)
```

[ ]:
```
19108
```

[ ]:
```
cursor = new_db.cursor()
cursor.execute("SELECT title, artist, score FROM reviews WHERE score=10")
results_cursor = cursor.fetchall()

print("Results using .cursor() method:")
for row in results_cursor:
    print(row)
```

```
Results using .cursor() method:
('metal box', 'public image ltd', 10.0)
('blood on the tracks', 'bob dylan', 10.0)
('another green world', 'brian eno', 10.0)
('songs in the key of life', 'stevie wonder', 10.0)
('in concert', 'nina simone', 10.0)
("tonight's the night", 'neil young', 10.0)
('hounds of love', 'kate bush', 10.0)
('sign "o" the times', 'prince', 10.0)
('1999', 'prince', 10.0)
('purple rain', 'prince, the revolution', 10.0)
('dirty mind', 'prince', 10.0)
('off the wall', 'michael jackson', 10.0)
('"heroes"', 'david bowie', 10.0)
('low', 'david bowie', 10.0)
('a love supreme: the complete masters', 'john coltrane', 10.0)
("people's instinctive travels and the paths of rhythm", 'a tribe called quest',
10.0)
('astral weeks', 'van morrison', 10.0)
('loaded: re-loaded 45th anniversary edition', 'the velvet underground', 10.0)
('sticky fingers', 'the rolling stones', 10.0)
('it takes a nation of millions to hold us back', 'public enemy', 10.0)
('the velvet underground  45th anniversary super deluxe edition', 'the velvet
underground', 10.0)
('spiderland', 'slint', 10.0)
('the infamous', 'mobb deep', 10.0)
('white light/white heat', 'the velvet underground', 10.0)
('in utero: 20th anniversary edition', 'nirvana', 10.0)
('rumours', 'fleetwood mac', 10.0)
('illmatic', 'nas', 10.0)
('donuts (45 box set)', 'j dilla', 10.0)
('voodoo', 'dangelo', 10.0)
('the disintegration loops', 'william basinski', 10.0)
('liquid swords: chess box deluxe edition', 'gza', 10.0)
("isn't anything", 'my bloody valentine', 10.0)
('tago mago [40th anniversary edition]', 'can', 10.0)
('the smile sessions', 'the beach boys', 10.0)
('laughing stock', 'talk talk', 10.0)
('nevermind [20th anniversary edition]', 'nirvana', 10.0)
('emergency & i [vinyl reissue]', 'the dismemberment plan', 10.0)
('my beautiful dark twisted fantasy', 'kanye west', 10.0)
('disintegration [deluxe edition]', 'the cure', 10.0)
('exile on main st. [deluxe edition]', 'the rolling stones', 10.0)
('quarantine the past', 'pavement', 10.0)
("ladies and gentlemen we are floating in space [collector's editon]",
'spiritualized', 10.0)
('the stone roses', 'the stone roses', 10.0)
('the beatles', 'the beatles', 10.0)
```

```
('abbey road', 'the beatles', 10.0)
('rubber soul', 'the beatles', 10.0)
('revolver', 'the beatles', 10.0)
("sgt. pepper's lonely hearts club band", 'the beatles', 10.0)
('magical mystery tour', 'the beatles', 10.0)
('stereo box', 'the beatles', 10.0)
('kid a: special collectors edition', 'radiohead', 10.0)
('reckoning [deluxe edition]', 'r.e.m.', 10.0)
('histoire de melody nelson', 'serge gainsbourg', 10.0)
("paul's boutique", 'beastie boys', 10.0)
('murmur [deluxe edition]', 'r.e.m.', 10.0)
("otis blue: otis redding sings soul [collector's edition]", 'otis redding',
10.0)
('unknown pleasures', 'joy division', 10.0)
('daydream nation: deluxe edition', 'sonic youth', 10.0)
('pink flag', 'wire', 10.0)
('born to run: 30th anniversary edition', 'bruce springsteen', 10.0)
('in the aeroplane over the sea', 'neutral milk hotel', 10.0)
('endtroducing… [deluxe edition]', 'dj shadow', 10.0)
("crooked rain, crooked rain: la's desert origins", 'pavement', 10.0)
('london calling: 25th anniversary legacy edition', 'the clash', 10.0)
('music has the right to children', 'boards of canada', 10.0)
('live at the apollo [expanded edition]', 'james brown', 10.0)
('no thanks!: the 70s punk rebellion', 'various artists', 10.0)
('marquee moon', 'television', 10.0)
('the ascension', 'glenn branca', 10.0)
("this year's model", 'elvis costello & the attractions', 10.0)
('yankee hotel foxtrot', 'wilco', 10.0)
('source tags and codes', '…and you will know us by the trail of dead', 10.0)
('the olatunji concert: the last live recording', 'john coltrane', 10.0)
('kid a', 'radiohead', 10.0)
('animals', 'pink floyd', 10.0)
('i see a darkness', 'bonnie prince billy', 10.0)
```

[ ]:
```
query = "SELECT title, artist, score FROM reviews WHERE score=10"
results_df = pd.read_sql_query(query, new_db)
print("\nResults using pd.read_sql_query():")
print(results_df)

new_db.commit()
cursor.close()
new_db.close()
```

```
Results using pd.read_sql_query():
                                                title  \
0                                           metal box
1                                  blood on the tracks
```

```
2                                   another green world
3                              songs in the key of life
4                                            in concert
..                                                  …
71                                source tags and codes
72    the olatunji concert: the last live recording
73                                                kid a
74                                              animals
75                                    i see a darkness

                                               artist  score
0                                    public image ltd   10.0
1                                          bob dylan   10.0
2                                          brian eno   10.0
3                                      stevie wonder   10.0
4                                        nina simone   10.0
..                                               …      …
71   …and you will know us by the trail of dead   10.0
72                                      john coltrane   10.0
73                                          radiohead   10.0
74                                         pink floyd   10.0
75                                 bonnie prince billy   10.0

[76 rows x 3 columns]
```

Part b Follow the instructions in the Jupyter notebook for this module to install MySQL and `mysql.connector` on your computer. Make sure the MySQL server is running. Then import `mysql.connector` and do all of the tasks listed for part a using a MySQL database (including commiting changes and closing the database connection). Take steps to hide your password - do not let it display in your notebook. [3 points]

```
[ ]: import os
     from dotenv import load_dotenv
     import mysql.connector
     import pandas as pd
     from sqlalchemy import create_engine, exc

     load_dotenv()
     MYSQL_ROOT_PASSWORD = os.getenv('MYSQL_ROOT_PASSWORD')
     MYSQL_DATABASE = os.getenv('MYSQL_DATABASE')

     engine_string = f'mysql+mysqlconnector://root:{MYSQL_ROOT_PASSWORD}@localhost:
       ↪3306/{MYSQL_DATABASE}'
     engine = create_engine(engine_string)

     reviews = pd.read_csv('reviews.csv')
     artists = pd.read_csv('artists.csv')
```

```
content = pd.read_csv('content.csv')
genres = pd.read_csv('genres.csv')
labels = pd.read_csv('labels.csv')
years = pd.read_csv('years.csv')
```

```
[ ]: def load_data_to_mysql(df, table_name):
         connection = engine.connect()
         trans = connection.begin()
         try:
             df.to_sql(table_name, connection, if_exists='replace', index=False)
             trans.commit()
         except exc.SQLAlchemyError as e:
             print(f"Error loading {table_name}: {e}")
             trans.rollback()
         finally:
             connection.close()

     # Save dataframes to the MySQL database
     load_data_to_mysql(reviews, 'reviews')
     load_data_to_mysql(artists, 'artists')
     load_data_to_mysql(content, 'content')
     load_data_to_mysql(genres, 'genres')
     load_data_to_mysql(labels, 'labels')
     load_data_to_mysql(years, 'years')
```

```
Error loading content: (mysql.connector.errors.OperationalError) 2013 (HY000):
Lost connection to MySQL server during query
[SQL: INSERT INTO content (`Unnamed: 0`, reviewid, content) VALUES
(%(UnnamedC_0)s, %(reviewid)s, %(content)s)]
[parameters: [{'UnnamedC_0': 0, 'reviewid': 22703, 'content': ""Trip-hop"
eventually became a ’90s punchline, a music-press shorthand for "overhyped hotel
lounge music." But today, the much-maligned subgenre almo … (9090 characters
truncated) … he immediacy of its creators' tumultuous late ’90s, and fearless
enough that it still sounds like it belongs in whatever timeframe you're playing
it."}, {'UnnamedC_0': 1, 'reviewid': 22721, 'content': 'Eight years, five
albums, and two EPs in, the New York-based outfit Krallice have long since shut
up purists about their "hipster black metal." Their … (2511 characters
truncated) … ut the moment is dense and terrifying enough to make Krallice
fans wonder: What could they accomplish with an album of compositions this
compact?\xa0'}, {'UnnamedC_0': 2, 'reviewid': 22659, 'content': 'Minneapolis’
Uranium Club seem to revel in being aggressively obtuse. They sprung up last
year with their Human Exploration EP, an eight-song tape of … (3291 characters
truncated) … tation. The band clearly wants to keep everyone on edge. If the
Uranium Club is anything, it is at least very self-aware of exactly what it is
doing.'}, {'UnnamedC_0': 3, 'reviewid': 22661, 'content': "Kleenex began\xa0with
a crash. It transpired one night not long after they’d formed,\xa0in Zurich of
1978, while\xa0the germinal punk group was onsta … (7677 characters truncated)
… idea.\xa0First Songs\xa0is a testament to the\xa0freedom in limitation, or
rather, the limitless nature of freedom when logic\xa0is tossed aside.\xa0"},
{'UnnamedC_0': 4, 'reviewid': 22725, 'content': 'It is impossible to consider a
given release by a footwork artist without confronting\xa0the long shadow cast
by DJ Rashad’s catalog, particularly hi … (2918 characters truncated) …
entation). Still, New Start proves that the prowess of footwork’s first family
is intact, and Taso might just be the glue that holds it all together.'},
{'UnnamedC_0': 5, 'reviewid': 22722, 'content': 'In the pilot episode of
"Insecure," the critically lauded HBO comedy series created by Issa Rae and
Larry Wilmore, Rae’s eponymous character Issa is  … (4058 characters
truncated) … nd season and while the protagonists’ fate is surely the highest
priority for fans, a chance to devour the forthcoming score can’t be too far
behind.'}, {'UnnamedC_0': 6, 'reviewid': 22704, 'content': 'Rapper Simbi
Ajikawo, who records as Little Simz, is by all measures on an upward trajectory,
with comparisons to iconoclasts like Lauryn Hill and pr … (3541 characters
truncated) …  SiR says on "One in Rotation"; there’s some false dichotomy shit
going on here, and then the track cuts off, abruptly, as if snapped out of a
dream.'}, {'UnnamedC_0': 7, 'reviewid': 22694, 'content': 'For the last thirty
years, Israel’s electronic music scene has operated in fits and starts. When
India opened up their borders to Israeli passports i … (2308 characters
truncated) … s squalls of feedback and noise that give the track some grit, so
that a mix of the ancient and futuristic also has a bit of the messy present in
it.'}  … displaying 10 of 18393 total bound parameter sets …  {'UnnamedC_0':
18391, 'reviewid': 2413, 'content': 'Well, kids, I just went back and re-read my
review for these guys\' last album,     What Burns never Returns, and what a
laugh it was! Sometimes I ju … (2154 characters truncated) …  weirdos, I
would suggest you take a large dose of Ketamine before     enjoying.  And don\'t
forget to stick a sharp, barbed object into your bottom!'}, {'UnnamedC_0':
18392, 'reviewid': 3723, 'content': 'Neil Hamburger\'s third comedy release is a
desperate affair, even for a\n    sad sack such as America\'s Funnyman.
Hamburger-- aka Gregg Turkington … (1585 characters truncated) … et the
joke.  But there\n    aren\'t really any jokes on this album, just a series of
empty shells and\n    failed observations… classic Hamburger.'}]]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

[ ]:
```python
connection = mysql.connector.connect(
    host='localhost',
    user='root',
    password=MYSQL_ROOT_PASSWORD,
    database=MYSQL_DATABASE,
    port=3306
)
cursor = connection.cursor()

cursor.execute("SELECT title, artist, score FROM reviews WHERE score=10")
results_cursor = cursor.fetchall()
print("Results using .cursor() method:")
for row in results_cursor:
    print(row)
```

```
Results using .cursor() method:
('metal box', 'public image ltd', 10.0)
('blood on the tracks', 'bob dylan', 10.0)
('another green world', 'brian eno', 10.0)
('songs in the key of life', 'stevie wonder', 10.0)
('in concert', 'nina simone', 10.0)
("tonight's the night", 'neil young', 10.0)
('hounds of love', 'kate bush', 10.0)
('sign "o" the times', 'prince', 10.0)
('1999', 'prince', 10.0)
('purple rain', 'prince, the revolution', 10.0)
('dirty mind', 'prince', 10.0)
('off the wall', 'michael jackson', 10.0)
('"heroes"', 'david bowie', 10.0)
('low', 'david bowie', 10.0)
('a love supreme: the complete masters', 'john coltrane', 10.0)
("people's instinctive travels and the paths of rhythm", 'a tribe called quest',
10.0)
('astral weeks', 'van morrison', 10.0)
('loaded: re-loaded 45th anniversary edition', 'the velvet underground', 10.0)
('sticky fingers', 'the rolling stones', 10.0)
('it takes a nation of millions to hold us back', 'public enemy', 10.0)
('the velvet underground  45th anniversary super deluxe edition', 'the velvet
underground', 10.0)
('spiderland', 'slint', 10.0)
('the infamous', 'mobb deep', 10.0)
('white light/white heat', 'the velvet underground', 10.0)
('in utero: 20th anniversary edition', 'nirvana', 10.0)
('rumours', 'fleetwood mac', 10.0)
('illmatic', 'nas', 10.0)
('donuts (45 box set)', 'j dilla', 10.0)
('voodoo', 'dangelo', 10.0)
('the disintegration loops', 'william basinski', 10.0)
('liquid swords: chess box deluxe edition', 'gza', 10.0)
("isn't anything", 'my bloody valentine', 10.0)
('tago mago [40th anniversary edition]', 'can', 10.0)
('the smile sessions', 'the beach boys', 10.0)
('laughing stock', 'talk talk', 10.0)
('nevermind [20th anniversary edition]', 'nirvana', 10.0)
('emergency & i [vinyl reissue]', 'the dismemberment plan', 10.0)
('my beautiful dark twisted fantasy', 'kanye west', 10.0)
('disintegration [deluxe edition]', 'the cure', 10.0)
('exile on main st. [deluxe edition]', 'the rolling stones', 10.0)
('quarantine the past', 'pavement', 10.0)
("ladies and gentlemen we are floating in space [collector's editon]",
'spiritualized', 10.0)
```

```
('the stone roses', 'the stone roses', 10.0)
('the beatles', 'the beatles', 10.0)
('abbey road', 'the beatles', 10.0)
('rubber soul', 'the beatles', 10.0)
('revolver', 'the beatles', 10.0)
("sgt. pepper's lonely hearts club band", 'the beatles', 10.0)
('magical mystery tour', 'the beatles', 10.0)
('stereo box', 'the beatles', 10.0)
('kid a: special collectors edition', 'radiohead', 10.0)
('reckoning [deluxe edition]', 'r.e.m.', 10.0)
('histoire de melody nelson', 'serge gainsbourg', 10.0)
("paul's boutique", 'beastie boys', 10.0)
('murmur [deluxe edition]', 'r.e.m.', 10.0)
("otis blue: otis redding sings soul [collector's edition]", 'otis redding',
10.0)
('unknown pleasures', 'joy division', 10.0)
('daydream nation: deluxe edition', 'sonic youth', 10.0)
('pink flag', 'wire', 10.0)
('born to run: 30th anniversary edition', 'bruce springsteen', 10.0)
('in the aeroplane over the sea', 'neutral milk hotel', 10.0)
('endtroducing… [deluxe edition]', 'dj shadow', 10.0)
("crooked rain, crooked rain: la's desert origins", 'pavement', 10.0)
('london calling: 25th anniversary legacy edition', 'the clash', 10.0)
('music has the right to children', 'boards of canada', 10.0)
('live at the apollo [expanded edition]', 'james brown', 10.0)
('no thanks!: the 70s punk rebellion', 'various artists', 10.0)
('marquee moon', 'television', 10.0)
('the ascension', 'glenn branca', 10.0)
("this year's model", 'elvis costello & the attractions', 10.0)
('yankee hotel foxtrot', 'wilco', 10.0)
('source tags and codes', '…and you will know us by the trail of dead', 10.0)
('the olatunji concert: the last live recording', 'john coltrane', 10.0)
('kid a', 'radiohead', 10.0)
('animals', 'pink floyd', 10.0)
('i see a darkness', 'bonnie prince billy', 10.0)
```

[ ]:
```
query = "SELECT title, artist, score FROM reviews WHERE score=10"
results_df = pd.read_sql_query(query, engine)
print("\nResults using pd.read_sql_query():")
print(results_df)

# Commit changes and close the database
connection.commit()
cursor.close()
connection.close()
```

```
Results using pd.read_sql_query():
```

```
                                          title  \
0                                     metal box   
1                           blood on the tracks   
2                           another green world   
3                      songs in the key of life   
4                                    in concert   
..                                          ...   
71                         source tags and codes   
72  the olatunji concert: the last live recording   
73                                        kid a   
74                                      animals   
75                            i see a darkness   

                                       artist  score  
0                            public image ltd   10.0  
1                                   bob dylan   10.0  
2                                   brian eno   10.0  
3                               stevie wonder   10.0  
4                                 nina simone   10.0  
..                                        ...    ...  
71  …and you will know us by the trail of dead   10.0  
72                               john coltrane   10.0  
73                                   radiohead   10.0  
74                                  pink floyd   10.0  
75                          bonnie prince billy   10.0  

[76 rows x 3 columns]
```

Part c Follow the instructions in the Jupyter notebook for this module to install PostgreSQL and psycopg2 on your computer. Then import psycopg2 and do all of the tasks listed for part a using a PostgreSQL database (including commiting changes and closing the database connection). Take steps to hide your password - do not let it display in your notebook. [3 points]

```
[ ]: import os
from dotenv import load_dotenv
import pandas as pd
from sqlalchemy import create_engine
import psycopg2

load_dotenv()
POSTGRES_PASSWORD = os.getenv('POSTGRES_PASSWORD')
conn = psycopg2.connect(
    user='postgres',
    password=POSTGRES_PASSWORD,
    host='localhost',
    port='5432'
)
```

```
conn.autocommit = True
cursor = conn.cursor()


try:
    cursor.execute('CREATE DATABASE pitchfork_c')
except psycopg2.errors.DuplicateDatabase:
    cursor.execute('DROP DATABASE pitchfork_c')
    cursor.execute('CREATE DATABASE pitchfork_c')

cursor.close()
conn.close()
```

[ ]:
```
conn = psycopg2.connect(
    user='postgres',
    password=POSTGRES_PASSWORD,
    host='localhost',
    port='5432',
    dbname='pitchfork_c'
)
conn.autocommit = True
cursor = conn.cursor()

reviews = pd.read_csv('reviews.csv')
artists = pd.read_csv('artists.csv')
content = pd.read_csv('content.csv')
genres = pd.read_csv('genres.csv')
labels = pd.read_csv('labels.csv')
years = pd.read_csv('years.csv')

dbms = 'postgresql'
connector = 'psycopg2'
user = 'postgres'
password = POSTGRES_PASSWORD
host = 'localhost'
port = '5432'
database = 'pitchfork_c'
engine_string = f'{dbms}+{connector}://{user}:{password}@{host}:{port}/
 ↪{database}'
engine = create_engine(engine_string)
```

[ ]:
```
def load_data_to_postgresql(df, table_name, chunk_size=1000):
    connection = engine.connect()
    trans = connection.begin()
    try:
        for i in range(0, len(df), chunk_size):
            df_chunk = df.iloc[i:i+chunk_size]
            df_chunk.to_sql(table_name, connection, if_exists='append',␣
↪index=False)
        trans.commit()
    except Exception as e:
        print(f"Error loading {table_name}: {e}")
        trans.rollback()
    finally:
        connection.close()

# Save dataframes to the PostgreSQL database
load_data_to_postgresql(reviews, 'reviews')
load_data_to_postgresql(artists, 'artists')
load_data_to_postgresql(content, 'content')
load_data_to_postgresql(genres, 'genres')
load_data_to_postgresql(labels, 'labels')
load_data_to_postgresql(years, 'years')
```

[ ]:
```
cursor.execute("SELECT title, artist, score FROM reviews WHERE score=10")
results_cursor = cursor.fetchall()
print("Results using .cursor() method:")
for row in results_cursor:
    print(row)

query = "SELECT title, artist, score FROM reviews WHERE score=10"
results_df = pd.read_sql_query(query, engine)
print("\nResults using pd.read_sql_query():")
print(results_df)


cursor.close()
conn.close()
```

```
Results using .cursor() method:
('metal box', 'public image ltd', 10.0)
('blood on the tracks', 'bob dylan', 10.0)
('another green world', 'brian eno', 10.0)
('songs in the key of life', 'stevie wonder', 10.0)
('in concert', 'nina simone', 10.0)
("tonight's the night", 'neil young', 10.0)
('hounds of love', 'kate bush', 10.0)
('sign "o" the times', 'prince', 10.0)
('1999', 'prince', 10.0)
('purple rain', 'prince, the revolution', 10.0)
('dirty mind', 'prince', 10.0)
('off the wall', 'michael jackson', 10.0)
('"heroes"', 'david bowie', 10.0)
('low', 'david bowie', 10.0)
('a love supreme: the complete masters', 'john coltrane', 10.0)
```

```
("people's instinctive travels and the paths of rhythm", 'a tribe called quest',
10.0)
('astral weeks', 'van morrison', 10.0)
('loaded: re-loaded 45th anniversary edition', 'the velvet underground', 10.0)
('sticky fingers', 'the rolling stones', 10.0)
('it takes a nation of millions to hold us back', 'public enemy', 10.0)
('the velvet underground  45th anniversary super deluxe edition', 'the velvet
underground', 10.0)
('spiderland', 'slint', 10.0)
('the infamous', 'mobb deep', 10.0)
('white light/white heat', 'the velvet underground', 10.0)
('in utero: 20th anniversary edition', 'nirvana', 10.0)
('rumours', 'fleetwood mac', 10.0)
('illmatic', 'nas', 10.0)
('donuts (45 box set)', 'j dilla', 10.0)
('voodoo', 'dangelo', 10.0)
('the disintegration loops', 'william basinski', 10.0)
('liquid swords: chess box deluxe edition', 'gza', 10.0)
("isn't anything", 'my bloody valentine', 10.0)
('tago mago [40th anniversary edition]', 'can', 10.0)
('the smile sessions', 'the beach boys', 10.0)
('laughing stock', 'talk talk', 10.0)
('nevermind [20th anniversary edition]', 'nirvana', 10.0)
('emergency & i [vinyl reissue]', 'the dismemberment plan', 10.0)
('my beautiful dark twisted fantasy', 'kanye west', 10.0)
('disintegration [deluxe edition]', 'the cure', 10.0)
('exile on main st. [deluxe edition]', 'the rolling stones', 10.0)
('quarantine the past', 'pavement', 10.0)
("ladies and gentlemen we are floating in space [collector's editon]",
'spiritualized', 10.0)
('the stone roses', 'the stone roses', 10.0)
('the beatles', 'the beatles', 10.0)
('abbey road', 'the beatles', 10.0)
('rubber soul', 'the beatles', 10.0)
('revolver', 'the beatles', 10.0)
("sgt. pepper's lonely hearts club band", 'the beatles', 10.0)
('magical mystery tour', 'the beatles', 10.0)
('stereo box', 'the beatles', 10.0)
('kid a: special collectors edition', 'radiohead', 10.0)
('reckoning [deluxe edition]', 'r.e.m.', 10.0)
('histoire de melody nelson', 'serge gainsbourg', 10.0)
("paul's boutique", 'beastie boys', 10.0)
('murmur [deluxe edition]', 'r.e.m.', 10.0)
("otis blue: otis redding sings soul [collector's edition]", 'otis redding',
10.0)
('unknown pleasures', 'joy division', 10.0)
('daydream nation: deluxe edition', 'sonic youth', 10.0)
('pink flag', 'wire', 10.0)
```

```
('born to run: 30th anniversary edition', 'bruce springsteen', 10.0)
('in the aeroplane over the sea', 'neutral milk hotel', 10.0)
('endtroducing… [deluxe edition]', 'dj shadow', 10.0)
("crooked rain, crooked rain: la's desert origins", 'pavement', 10.0)
('london calling: 25th anniversary legacy edition', 'the clash', 10.0)
('music has the right to children', 'boards of canada', 10.0)
('live at the apollo [expanded edition]', 'james brown', 10.0)
('no thanks!: the 70s punk rebellion', 'various artists', 10.0)
('marquee moon', 'television', 10.0)
('the ascension', 'glenn branca', 10.0)
("this year's model", 'elvis costello & the attractions', 10.0)
('yankee hotel foxtrot', 'wilco', 10.0)
('source tags and codes', '…and you will know us by the trail of dead', 10.0)
('the olatunji concert: the last live recording', 'john coltrane', 10.0)
('kid a', 'radiohead', 10.0)
('animals', 'pink floyd', 10.0)
('i see a darkness', 'bonnie prince billy', 10.0)

Results using pd.read_sql_query():
                                            title  \
0                                       metal box
1                              blood on the tracks
2                              another green world
3                         songs in the key of life
4                                       in concert
..                                            …
71                           source tags and codes
72  the olatunji concert: the last live recording
73                                          kid a
74                                        animals
75                               i see a darkness

                                          artist  score
0                               public image ltd   10.0
1                                      bob dylan   10.0
2                                      brian eno   10.0
3                                  stevie wonder   10.0
4                                    nina simone   10.0
..                                           …     …
71  …and you will know us by the trail of dead   10.0
72                                 john coltrane   10.0
73                                     radiohead   10.0
74                                    pink floyd   10.0
75                           bonnie prince billy   10.0

[76 rows x 3 columns]
```

1.1.7 Problem 4

Colin Mitchell is a web-developer and artist who has a bunch of cool projects that play with what data can do on the internet. One of his projects is Today in History, which provides an API to access all the Wikipedia pages for historical events that happened on this day in JSON format. The records in this JSON are stored in the `['data']['events']` path. Here's the first listing for today:

```
[ ]: history = requests.get("https://history.muffinlabs.com/date")
     history_json = json.loads(history.text)
     events = history_json['data']['Events']
     events[0]
```

```
[ ]: {'year': '30 BC',
      'text': 'Octavian (later known as Augustus) enters Alexandria, Egypt, bringing
     it under the control of the Roman Republic.',
      'html': '30 BC - 30 BC - <a href="https://wikipedia.org/wiki/Augustus"
     title="Augustus">Octavian</a> (later known as Augustus) enters <a
     href="https://wikipedia.org/wiki/Alexandria" title="Alexandria">Alexandria</a>,
     <a href="https://wikipedia.org/wiki/Egypt" title="Egypt">Egypt</a>, bringing it
     under the control of the <a href="https://wikipedia.org/wiki/Roman_Republic"
     title="Roman Republic">Roman Republic</a>.',
      'no_year_html': '30 BC - <a href="https://wikipedia.org/wiki/Augustus"
     title="Augustus">Octavian</a> (later known as Augustus) enters <a
     href="https://wikipedia.org/wiki/Alexandria" title="Alexandria">Alexandria</a>,
     <a href="https://wikipedia.org/wiki/Egypt" title="Egypt">Egypt</a>, bringing it
     under the control of the <a href="https://wikipedia.org/wiki/Roman_Republic"
     title="Roman Republic">Roman Republic</a>.',
      'links': [{'title': 'Augustus',
        'link': 'https://wikipedia.org/wiki/Augustus'},
       {'title': 'Alexandria', 'link': 'https://wikipedia.org/wiki/Alexandria'},
       {'title': 'Egypt', 'link': 'https://wikipedia.org/wiki/Egypt'},
       {'title': 'Roman Republic',
        'link': 'https://wikipedia.org/wiki/Roman_Republic'}]}
```

For this problem, you will use MongoDB and the `pymongo` library to create a local document store NoSQL database containing these historical events.

Follow the instructions in the Jupyter notebook for this module to install MongoDB and `pymongo` on your computer. Make sure the local MongoDB server is running. Then import `pymongo`, connect to the local MongoDB client, create a database named “history” and a collection within that database named “today”. Insert all of the records in `events` into this collection. Then issue the following query to find all of the records whose text contain the word “Virginia”:

```
query = {
    "text":{
        "$regex": 'Virginia'
    }
}
```

If there are no results that contain the word “Virginia”, choose a different work like “England” or “China”. Display the count of the number of documents that match this query, display the output of the query, and generate a JSON formatted variable containing the output. [3 points]

[ ]:
```python
import requests
import json
import os
from pymongo import MongoClient
from bson.json_util import dumps, loads
from dotenv import load_dotenv
load_dotenv()
history = requests.get("https://history.muffinlabs.com/date")
history_json = json.loads(history.text)
events = history_json['data']['Events']

mongo_username = os.getenv('MONGO_INITDB_ROOT_USERNAME')
mongo_password = os.getenv('MONGO_INITDB_ROOT_PASSWORD')
mongo_host = os.getenv('MONGO_HOST', 'localhost')
mongo_port = os.getenv('MONGO_PORT', '27017')

mongo_url = f"mongodb://{mongo_username}:{mongo_password}@{mongo_host}:
  ↪{mongo_port}/"

client = MongoClient(mongo_url)
history_db = client["history"]
collist = history_db.list_collection_names()
if "today" in collist:
    history_db.today.drop()

today_collection = history_db["today"]
today_collection.insert_many(events)
query = {"text": {"$regex": 'China'}}
results = today_collection.find(query)


count = today_collection.count_documents(query)
print(f"Number of documents containing 'China': {count}")
for doc in results:
    print(doc)
```

```
Number of documents containing 'China': 3
{'_id': ObjectId('66ab3d1d01612a1b91f1d2b7'), 'year': '607', 'text': 'Ono no
Imoko is dispatched as envoy to the Sui court in China (Traditional Japanese
date: July 3, 607).', 'html': '607 - <a
href="https://wikipedia.org/wiki/Ono_no_Imoko" title="Ono no Imoko">Ono no
Imoko</a> is dispatched as envoy to the <a
href="https://wikipedia.org/wiki/Sui_dynasty" title="Sui dynasty">Sui</a> court
in China (Traditional <a href="https://wikipedia.org/wiki/Japanese_calendar"
title="Japanese calendar">Japanese date</a>: July 3, 607).', 'no_year_html': '<a
href="https://wikipedia.org/wiki/Ono_no_Imoko" title="Ono no Imoko">Ono no
Imoko</a> is dispatched as envoy to the <a
href="https://wikipedia.org/wiki/Sui_dynasty" title="Sui dynasty">Sui</a> court
in China (Traditional <a href="https://wikipedia.org/wiki/Japanese_calendar"
title="Japanese calendar">Japanese date</a>: July 3, 607).', 'links': [{'title':
'Ono no Imoko', 'link': 'https://wikipedia.org/wiki/Ono_no_Imoko'}, {'title':
'Sui dynasty', 'link': 'https://wikipedia.org/wiki/Sui_dynasty'}, {'title':
'Japanese calendar', 'link': 'https://wikipedia.org/wiki/Japanese_calendar'}]}
{'_id': ObjectId('66ab3d1d01612a1b91f1d2ce'), 'year': '1894', 'text': 'The
Empire of Japan and Qing China declare war on each other after a week of
fighting over Korea, formally inaugurating the First Sino-Japanese War.',
'html': '1894 - The <a href="https://wikipedia.org/wiki/Empire_of_Japan"
title="Empire of Japan">Empire of Japan</a> and <a
href="https://wikipedia.org/wiki/Qing_dynasty" title="Qing dynasty">Qing
China</a> declare war on each other after a week of fighting over Korea,
formally inaugurating the <a href="https://wikipedia.org/wiki/First_Sino-
Japanese_War" title="First Sino-Japanese War">First Sino-Japanese War</a>.',
'no_year_html': 'The <a href="https://wikipedia.org/wiki/Empire_of_Japan"
title="Empire of Japan">Empire of Japan</a> and <a
href="https://wikipedia.org/wiki/Qing_dynasty" title="Qing dynasty">Qing
China</a> declare war on each other after a week of fighting over Korea,
formally inaugurating the <a href="https://wikipedia.org/wiki/First_Sino-
Japanese_War" title="First Sino-Japanese War">First Sino-Japanese War</a>.',
'links': [{'title': 'Empire of Japan', 'link':
'https://wikipedia.org/wiki/Empire_of_Japan'}, {'title': 'Qing dynasty', 'link':
'https://wikipedia.org/wiki/Qing_dynasty'}, {'title': 'First Sino-Japanese War',
'link': 'https://wikipedia.org/wiki/First_Sino-Japanese_War'}]}
{'_id': ObjectId('66ab3d1d01612a1b91f1d2e3'), 'year': '1966', 'text': 'Purges of
intellectuals and imperialists becomes official China policy at the beginning of
the Cultural Revolution.', 'html': '1966 - Purges of intellectuals and
imperialists becomes official China policy at the beginning of the <a
href="https://wikipedia.org/wiki/Cultural_Revolution" title="Cultural
Revolution">Cultural Revolution</a>.', 'no_year_html': 'Purges of intellectuals
and imperialists becomes official China policy at the beginning of the <a
href="https://wikipedia.org/wiki/Cultural_Revolution" title="Cultural
Revolution">Cultural Revolution</a>.', 'links': [{'title': 'Cultural
Revolution', 'link': 'https://wikipedia.org/wiki/Cultural_Revolution'}]}
```

1.2 Problem 5 [No points, no need to write anything here, but do it anyway!]

When you are done working, go to the same terminal window you used to launch the databases with `docker compose up`. Here, type CONTROL + C on your keyboard to shut down the container.

Next type

```
docker compose down
```

This step removes extra database software, networks, volumes, etc. running on your computer. If you don't need them, don't clog your computer.

Whenever you need to work with databases, return to the terminal, navigate to this folder and type

```
docker compose up
```

to bring all these resources back online.

```
[ ]:
```