

Lab Assignment 6: Creating and Connecting to Databases

DS 6001: Practice and Application of Data Science

Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. To receive full credit, make sure you address every part of the problem, make sure your document is formatted in a clean and professional way, and make sure the notebook is converted to a PDF and submitted to Gradescope according to these instructions:

https://docs.google.com/document/d/1B9ZkK7n_hP_hQ9lIGm31Web4S6hGnwMz9Ad7EWm3N50/ecusp=sharing.

This assignment requires you to include tables and images.

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 columns, 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

To put an image into a markdown cell in a Jupyter notebook, save the image as a .png or .jpg file in the same folder where you have saved your Jupyter notebook, and use markdown code that looks like this:

```

```


where you will need to replace `imagefile.png` with the name of your own image file.

Alternatively, if you want to control the size of the image in your notebook, type the following

code on its own line in the markdown cell:

```

```

Here the `width` option allows you to control the size of the image by making this number larger or smaller. When converting the notebook to  PDF format, make sure that the images display correctly in the PDF prior to submitting to Gradescope.

Problem 0

Import the following libraries, load the `.env` file where you store your passwords (see the notebook for module 4 for details), and turn off the error tracebacks to make errors easier to read:

```
In [1]: import numpy as np
import pandas as pd
import wget
import sqlite3
import sqlalchemy
import requests
import json
import os
import sys
import dotenv
os.chdir("/Users/jk8sd/Box Sync/Practice and Applications 1 online/Module 6 - Bu
dotenv.load_dotenv() # register the .env file where passwords are stored
sys.tracebacklimit = 0 # turn off the error tracebacks
```

Problem 1

Suppose that we have (fake) data on people who are currently being hospitalized. Here are five records in the data:

patient	conditions	dateofbirth	age	sex	attendingphysician	APmedschool	APyearsexpe
Nkemdilim Arendonk	[Pneumonia, Diabetes]	2/21/1962	58	M	Earnest Caro	University of California (Irvine)	14
Raniero Coumans	[Appendicitis, Crohn's disease]	8/15/1990	29	M	Pamela English	University of Michigan	29
Mizuki Debenham	[Kidney Cancer]	3/12/1977	43	F	Lewis Conti	North Carolina State University	8
Zoë De Witt	[Cardiomyopathy, Diabetes, Sciatica]	11/23/1947	72	F	Theresa Dahlmans	Lake Erie College of Medicine	17
Bonnie Hooper	[Pancreatic Cancer, Sciatica]	7/4/1951	68	F	Steven Garbutt	Ohio State University	36

The columns in this dataset are:

- **patient:** Patient name
- **conditions:** A list of the conditions that are relevant to the patient's hospitalization
- **dateofbirth:** The patient's date of birth
- **age:** The patient's age
- **sex:** The patient's sex
- **attendingphysician:** The name of the attending physician for the patient
- **APmedschool:** The name of the school where the attending physician got a medical degree
- **APyearsexperiece:** The attending physician's number of years of experience post-residency
- **hospital:** The hospital where the attending physicial is employed
- **hospitallocation:** The location of the hospital

For this problem, assume that

1. Some people in the data share the same name, but no two people in the data share the same name and date of birth.
2. Every attending physician is employed at only one hospital.
3. Every hospital exists at only one location.
4. There's more than one doctor with the same name, but there are no doctors with the same name that work at the same hospital.

Part a

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

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]

Part c

Rearrange the data on the five patients into a group of data tables that together meet the requirements of third normal form.

Note that the patient's age is a derived attribute from the patient's date of birth, but please don't make an extra data table just for age. In principle, if we are worried about data inconsistencies we can simply remove age from the database and calculate it when needed from date of birth. But for this exercise, leave age in the table and ignore its dependency with date of birth. [2 points]

Problem 2

For this problem, create ER diagrams of the database you created in problem 1, part c using draw.io: <https://app.diagrams.net/>. The symbols used for both Chen's notation and IE notation

are on the left-hand toolbar.

Part a

Create a conceptual ER diagram using Chen's notation. [2 points]

Part b

Create a logical ER diagram using Chen's notation. [2 points]

Part c

Create a conceptual ER diagram using IE notation. [2 points]

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.

To get the data, first set the working directory the folder on your computer to the folder where you want the CSV files to be. This should be the same folder where you saved our lab notebook and all associated files. Then change this line of code to the address for that folder:

```
In [2]: os.chdir("/Users/jk8sd/Downloads")
```

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

```
In [3]: 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 is to practice creating databases from individual data frames. Next we load the CSVs to create the data frames in Python:

```
In [4]: 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]

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 committing changes and closing the database connection). Take steps to hide your password - do not let it display in your notebook. [2 points]

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 committing changes and closing the database connection). Take steps to hide your password - do not let it display in your notebook. [2 points]

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:

```
In [5]: history = requests.get("https://history.muffinlabs.com/date")
history_json = json.loads(history.text)
events = history_json['data']['Events']
events[0]
```

```
Out[5]: {'year': '538',
'text': 'Vitiges, king of the Ostrogoths ends his siege of Rome and retreats to Ravenna, leaving the city in the hands of the victorious Byzantine general, Belisarius.',
'html': '538 - <a href="https://wikipedia.org/wiki/Vitiges" title="Vitiges">Vitiges</a>, <a href="https://wikipedia.org/wiki/Germanic_kingship" title="Germanic kingship">king</a> of the <a href="https://wikipedia.org/wiki/Ostrogoths" title="Ostrogoths">Ostrogoths</a> ends his <a href="https://wikipedia.org/wiki/Siege_of_Rome_(537%E2%80%9338)" class="mw-redirect" title="Siege of Rome (537-38)">siege of Rome</a> and retreats to <a href="https://wikipedia.org/wiki/Ravenna" title="Ravenna">Ravenna</a>, leaving the city in the hands of the victorious <a href="https://wikipedia.org/wiki/Byzantine_Empire" title="Byzantine Empire">Byzantine</a> general, <a href="https://wikipedia.org/wiki/Belisarius" title="Belisarius">Belisarius</a>.',
'no_year_html': '<a href="https://wikipedia.org/wiki/Vitiges" title="Vitiges">Vitiges</a>, <a href="https://wikipedia.org/wiki/Germanic_kingship" title="Germanic kingship">king</a> of the <a href="https://wikipedia.org/wiki/Ostrogoths" title="Ostrogoths">Ostrogoths</a> ends his <a href="https://wikipedia.org/wiki/Siege_of_Rome_(537%E2%80%9338)" class="mw-redirect" title="Siege of Rome (537-38)">siege of Rome</a> and retreats to <a href="https://wikipedia.org/wiki/Ravenna" title="Ravenna">Ravenna</a>, leaving the city in the hands of the victorious <a href="https://wikipedia.org/wiki/Byzantine_Empire" title="Byzantine Empire">Byzant
```

```

ine</a> general, <a href="https://wikipedia.org/wiki/Belisarius" title="Belisari
us">Belisarius</a>.',
'links': [{ 'title': 'Vitiges', 'link': 'https://wikipedia.org/wiki/Vitiges'},
{ 'title': 'Germanic kingship',
'link': 'https://wikipedia.org/wiki/Germanic_kingship'},
{ 'title': 'Ostrogoths', 'link': 'https://wikipedia.org/wiki/Ostrogoths'},
{ 'title': 'Siege of Rome (537-38)',
'link': 'https://wikipedia.org/wiki/Siege_of_Rome_(537%E2%80%9338)'},
{ 'title': 'Ravenna', 'link': 'https://wikipedia.org/wiki/Ravenna'},
{ 'title': 'Byzantine Empire',
'link': 'https://wikipedia.org/wiki/Byzantine_Empire'},
{ 'title': 'Belisarius', 'link': 'https://wikipedia.org/wiki/Belisarius' } ] }

```

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 word 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. [2 points]