H. Diana McSpadden

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/edit .

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 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:

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 [ ]:
```
import numpy as np
import pandas as pd
import wget
import sqlite3
import sqlalchemy
import requests
import json
import os
import sys
import dotenv

dotenv.load_dotenv('mod.env') # 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	hospital	hospitallocation
Nkemdilim Arendonk	[Pneumonia, Diabetes]	2/21/1962	58	M	Earnest Caro	University of California (Irvine)	14	UPMC Presbyterian Shadyside	Pittsburgh, PA
Raniero Coumans	[Appendicitis, Crohn's disease]	8/15/1990	29	M	Pamela English	University of Michigan	29	Northwestern Memorial Hospital	Chicago, IL
Mizuki Debenham	[Kidney Cancer]	3/12/1977	43	F	Lewis Conti	North Carolina State University	8	Houston Methodist Hospital	Houston, TX
Zoë De Witt	[Cardiomyopathy, Diabetes, Sciatica]	11/23/1947	72	F	Theresa Dahlmans	Lake Erie College of Medicine	17	Mount Sinai Hospital	New York, NY

patient	conditions	dateofbirth	age	sex	attendingphysician	APmedschool	APyearsexpe	hospital	hospitallocation
Bonnie Hooper	[Pancreatic Cancer, Sciatica]	7/4/1951	68	F	Steven Garbutt	Ohio State University	36	UCSF Medical Center	San Francisco, CA

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 postresidency
- hospital: The hospital where the attending physicial is employed
- hospitallocation: The location of the hospital

For this problem, assume that

- Some people in the data share the same name, but no two people in the data share the same name and date of birth.
- Every attending physician is employed at only one hospital.
- Every hospital exists at only one location.
- 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]

Answer:

Criteria of First Normal Form:

1. All rows unique.
    - adding Patient_ID - I know the documentation states that the combination of Name and date of birth will be unique, but I really like ID's.
2. A column must contain atomic values, i.e. only single values. The multiple values in the **conditions** column are a problem.
    - I split FName and LName in order to have Atomic values for names (patient and physicians)
3. The values in a column must be of the same type. We don't have any problems with this.
4. Each column must have a unique name.
    - By the problem definition we know that some people share names, but not names and date's of birth.
    - While I would love to create a Person_ID right now, I think it is implied that that table will have a compound key of patient_name and dateofbirth.
5. No repeating groups.

US_State

ID	Name
PA	Pennsylvania
IL	Illinois
TX	Texas
NY	New York
CA	California

US_City

ID	Name	US_State_ID
1	Pittsburgh	PA
2	Chicago	IL
3	Houston	TX
4	New York	NY
5	San Francisco	CA

Patient

Patient_ID	patient_FName	patient_LName	dateofbirth	age	sex	attending_FName	attending_LName	APmedschool	APyearsexpe	ho
1	Nkemdilim	Arendonk	2/21/1962	58	M	Earnest	Caro	University of California (Irvine)	14	UPM Presb Shad
2	Raniero	Coumans	8/15/1990	29	M	Pamela	English	University of Michigan	29	North Mem Hosp
3	Mizuki	Debenham	3/12/1977	43	F	Lewis	Conti	North Carolina State University	8	Hous Meth Hosp
4	Zoë	De Witt	11/23/1947	72	F	Theresa	Dahlmans	Lake Erie College of Medicine	17	Mour Hosp
5	Bonnie	Hooper	7/4/1951	68	F	Steven	Garbutt	Ohio State University	36	UCSF Medi Cente

Condition

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

Patient_Condition

This is a one to one/many

Patient_ID	Condition_ID
1	1
1	2
2	3
2	4
3	5
4	6
4	2
4	7
5	8
5	7

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]

Answer

Criteria of First Normal Form:

1. Table should be in First Normal Form
2. There should be no "partial dependencies"
    A. full dependency == ID value gets you the complete row
    B. partial dependency

For **2NF** every non-key attribute is fully dependent on primary key. For the Patient table, the key is the Patient_ID, the physician information (name, yrs experience, etc), and hospital information (name, and city ID) are partially dependent, and thus break 2NF rules.

While the problem stated that "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.", I also created a separate Hospital table. This allows the model the flexibility to later store additional information about the hospital, such as number of employees, or other characteristics. I also did the same with the APMedSchool, for the same reason that we may want to store additional characteristics of the medical school. There is a very valid 2NF version where the APMedschool column is in the Physician table, and the entire Hospital table is also part of the Physician table (both the Hospital_Name and US_City_ID).

US_State

ID	Name
PA	Pennsylvania
IL	Illinois
TX	Texas
NY	New York
CA	California

US_City

ID	Name	US_State_ID
1	Pittsburgh	PA
2	Chicago	IL
3	Houston	TX
4	New York	NY
5	San Francisco	CA

Hospital

ID	Name	US_City_ID
1	UPMC Presbyterian Shadyside	1
2	Northwestern Memorial Hospital	2
3	Houston Methodist Hospital	3
4	Mount Sinai Hospital	4
5	UCSF Medical Center	5

APmedschool

ID	Name
1	University of California (Irvine)
2	University of Michigan
3	North Carolina State University
4	Lake Erie College of Medicine
5	Ohio State University

Physician

ID	FName	LName	APmedschool_ID	APYearsExpe	Hospital_ID
1	Earnest	Caro	1	14	1
2	Pamela	English	2	29	2
3	Lewis	Conti	3	8	3
4	Theresa	Dahlmans	4	17	4
5	Steven	Garbutt	5	36	5

Patient

Patient_ID	patient_FName	patient_LName	dateofbirth	age	sex	Physician_ID
1	Nkemdilim	Arendonk	2/21/1962	58	M	1

Patient_ID	patient_FName	patient_LName	dateofbirth	age	sex	Physician_ID
2	Raniero	Coumans	8/15/1990	29	M	2
3	Mizuki	Debenham	3/12/1977	43	F	3
4	Zoë	De Witt	11/23/1947	72	F	4
5	Bonnie	Hooper	7/4/1951	68	F	5

ConditionPatient

Patient_ID	Condition
1	Pneumonia
1	Diabetes
2	Appendicitis
2	Crohn's disease
3	Kidney Cancer
4	Cardiomyopathy
4	Diabetes
4	Sciatica
5	Pancreatic Cancer
5	Sciatica

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]

Answer:

My comments about about the 2NF tables and the additions of the Hospital and APMedSchool tables result in the 3NF structure seen below. There are additional comments about decisions regarding the Patient_AttendingPhysician table as well.

US_State

ID	Name
PA	Pennsylvania
IL	Illinois
TX	Texas
NY	New York
CA	California

US_City

ID	Name	US_State_ID
1	Pittsburgh	PA
2	Chicago	IL
3	Houston	TX
4	New York	NY
5	San Francisco	CA

Hospital

ID	Name	US_City_ID
1	UPMC Presbyterian Shadyside	1
2	Northwestern Memorial Hospital	2
3	Houston Methodist Hospital	3

ID	Name	US_City_ID
4	Mount Sinai Hospital	4
5	UCSF Medical Center	5

APmedschool

ID	Name
1	University of California (Irvine)
2	University of Michigan
3	North Carolina State University
4	Lake Erie College of Medicine
5	Ohio State University

Physician

ID	FName	LName	APmedschool_ID	APYearsExpe	Hospital_ID
1	Earnest	Caro	1	14	1
2	Pamela	English	2	29	2
3	Lewis	Conti	3	8	3
4	Theresa	Dahlmans	4	17	4
5	Steven	Garbutt	5	36	5

Condition

ID	Name
1	Pneumonia
2	Diabetes
3	Appendicitis
4	Crohn's disease

ID	Name
5	Kidney Cancer
6	Cardiomyopathy
7	Sciatica
8	Pancreatic Cancer

Patient

ID	FName	LName	DOB_month	DOB_day	DOB_year	age
1	Earnest	Caro	2	21	1962	58
2	Pamela	English	8	15	1990	29
3	Lewis	Conti	3	12	1977	43
4	Theresa	Dahlmans	11	23	1947	72
5	Steven	Garbutt	7	4	1951	68

Patient_Condition

This is a one to one/many

Patient_ID	Condition_ID
1	1
1	2
2	3
2	4
3	5
4	6
4	2
4	7

Patient_ID	Condition_ID
5	8
5	7

Patient_AttendingPhysician

We really need a Patient_Visit table with admit and release dates, and we need hospital units as well, but this will do for the specifications outlined in the Problem.

I do not see how one could do this with a different structure. The same patient could see the same physician again on a different date, or see a different physician on the same or different date (yes, the cities are far apart, but that wouldn't be the case for all hospitals).

This could be a many-to-many relationship, hence the primary key `PatientAttendingPhysician_ID`.

PatientAttendingPhysician_ID	Patient_ID	Physician_ID	AdmitDateTime	ReleaseDateTime
1	1	1	2020-01-01 09:53:05	2020-01-01 21:19:10
2	2	2	2020-01-01 10:16:01	2020-01-03 02:03:45
3	3	3	2020-01-01 15:03:26	2020-01-01 16:15:25
4	4	4	2020-01-01 09:01:54	2020-01-15 10:34:05
5	5	5	2020-01-01 18:45:12	2020-02-03 08:48:10

There is a simplier version of this relationship, where PatientAttendingPhysician_ID is a simplier join table where the compound key is the combination of `Patient_ID` and `Physician_ID`. That version is here. I also provide the conceptual ER diagrams for both the complicated and the simplier versions (part a). But I provide parts (b) and (c) for only the more complicated version.

Patient_ID	Physician_ID
1	1
2	2
3	3
4	4

Patient_ID	Physician_ID
5	5

Problem 2

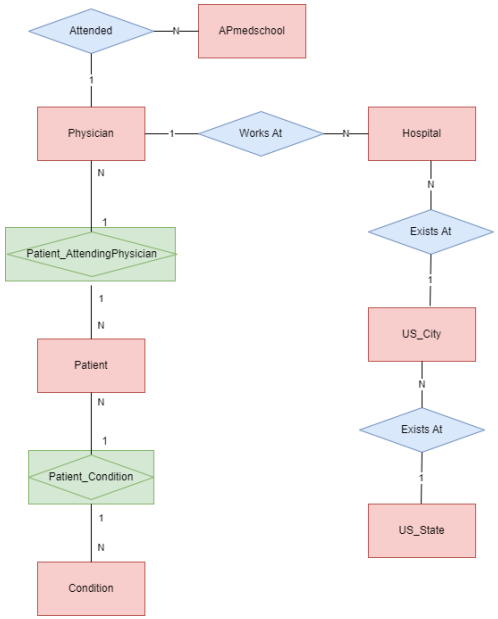
For this problem, create ER diagrams of the database you created in problem 1, part c using draw.io: https://app.diagrams.net/. The symbols used for both Chen's notation and IE notation are on the left-hand toolbar.

Part a

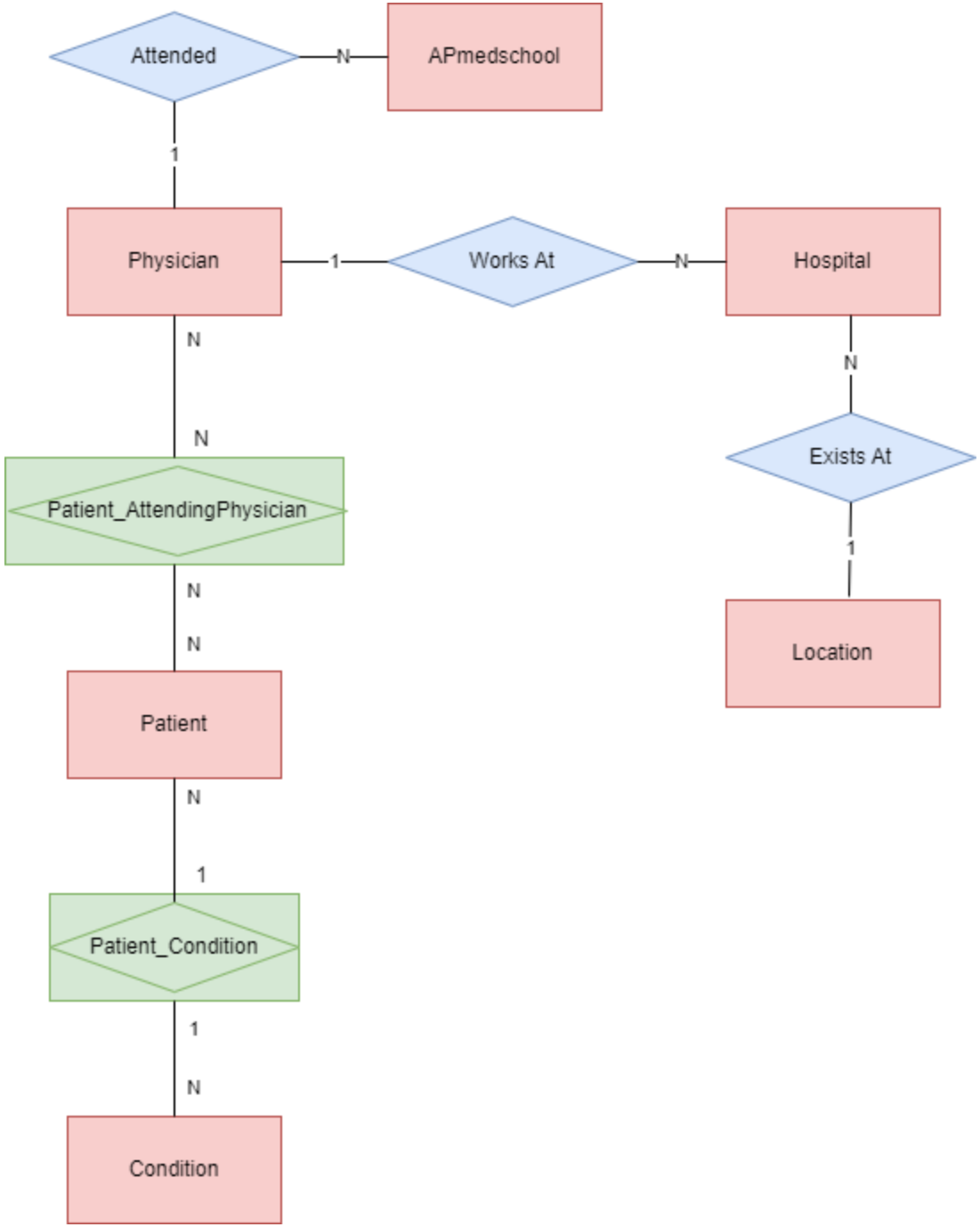
Create a conceptual ER diagram using Chen's notation. [2 points]

Answer

Here is the simplier version ER diagram in Chen's notation:

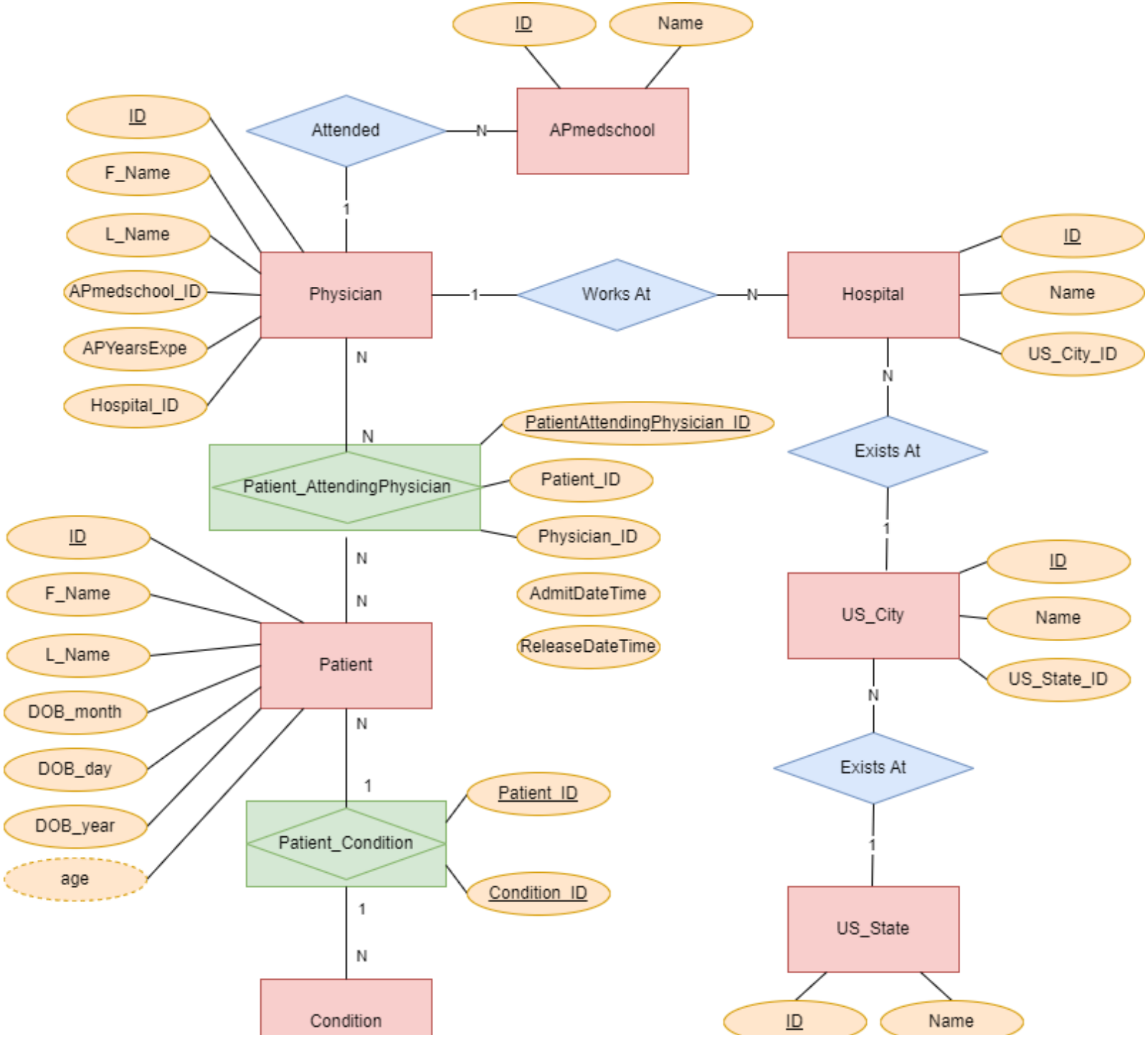


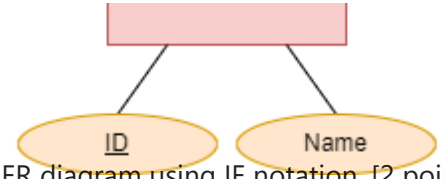
Here is the more complicated version of the ER diagram in Chen's notation:


Attended
N
APmedschool
1
Physician
1
Works At
N
Hospital
N
N
Exists At
N
1
Patient_AttendingPhysician
Location
N
N
Patient
N
1
Patient_Condition
1
N
Condition

Part b

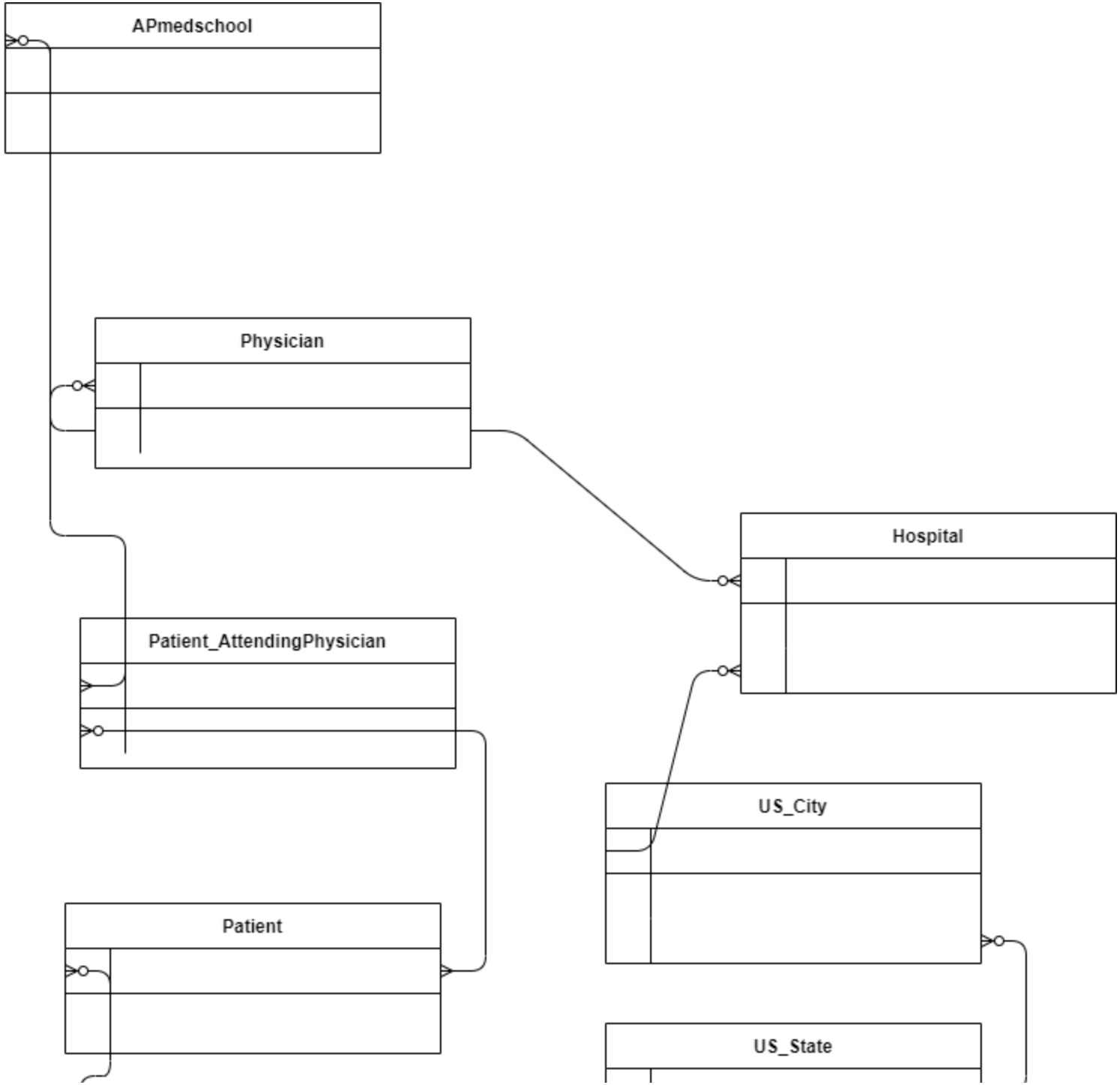
Create a logical ER diagram using Chen's notation. [2 points]

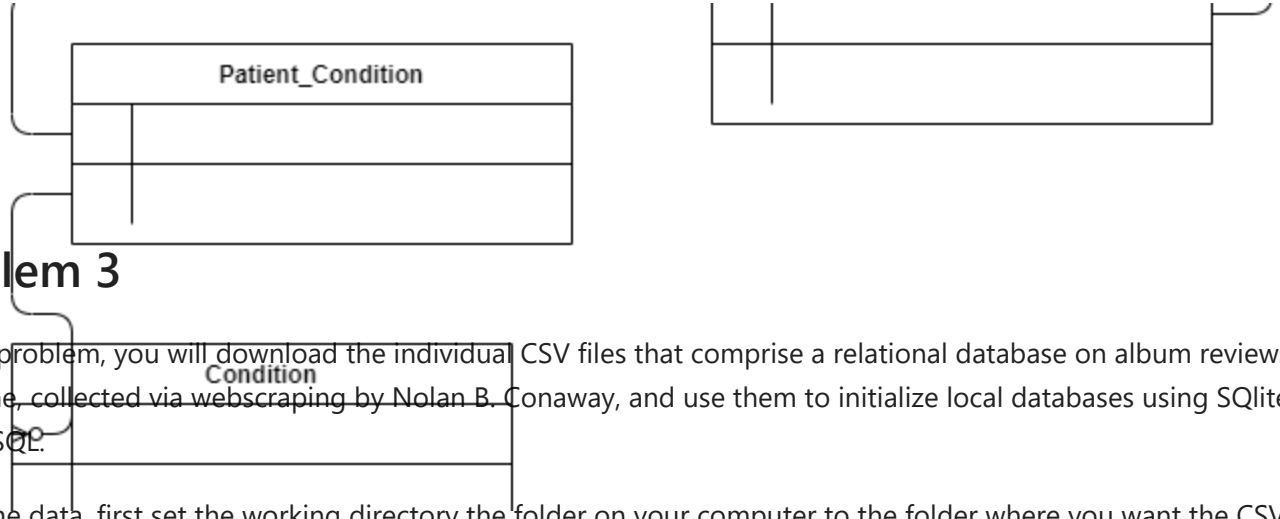

ID
Name
APmedschool
Attended
N
1
ID
F_Name
L_Name
APmedschool_ID
APYearsExpe
Hospital_ID
Physician
1
Works At
N
Hospital
ID
Name
US_City_ID
N
N
PatientAttendingPhysician_ID
Patient_AttendingPhysician
Patient_ID
Physician_ID
AdmitDateTime
ReleaseDateTime
Exists At
1
ID
F_Name
L_Name
DOB_month
DOB_day
DOB_year
age
Patient
N
N
N
1
Patient_ID
Patient_Condition
Condition_ID
1
N
Condition
US_City
ID
Name
US_State_ID
N
Exists At
1
US_State
ID
Name



Part c

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


APmedschool
Physician
Hospital
Patient_AttendingPhysician
US_City
Patient
US_State



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:

```
os.chdir("/Users/jk8sd/Downloads")
```

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

In [ ]:
```
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))
```

In [ ]:
```
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:

1. first, using the .cursor() method, and
2. 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]

In [ ]:
```
# initialize a new database using Sqlite3
db_diana = sqlite3.connect('pitchfork_diana.db')

# add the reviews, artists, content, genres, labels, and years tables to the database
reviews.to_sql('reviews', db_diana, if_exists='replace', index=False)
artists.to_sql('artists', db_diana, if_exists='replace', index=False)
content.to_sql('content', db_diana, if_exists='replace', index=False)
genres.to_sql('genres', db_diana, if_exists='replace', index=False)
labels.to_sql('labels', db_diana, if_exists='replace', index=False)
years.to_sql('years', db_diana, if_exists='replace', index=False)
```

Out[ ]:
```
19108
```

First, the .cursor() method

In [ ]:
```
cur = db_diana.cursor()
cur.execute("SELECT title, artist, score FROM reviews WHERE score=10")
rows = cur.fetchall()
# example from https://www.sqlitetutorial.net/sqlite-python/sqlite-python-select/
for row in rows:
    print(row)
```

```
('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)
```

Second, using pd.read_sql_query()

In [ ]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [ ]:
```
# from the documentation: https://pandas.pydata.org/docs/reference/api/pandas.read_sql_query.html
df = pd.read_sql_query("SELECT title, artist, score FROM reviews WHERE score=10", db_diana)
df
```

Out[ ]:

	title	artist	score
0	metal box	public image ltd	10.0
1	blood on the tracks	bob dylan	10.0
2	another green world	brian eno	10.0
3	songs in the key of life	stevie wonder	10.0
4	in concert	nina simone	10.0
5	tonight's the night	neil young	10.0
6	hounds of love	kate bush	10.0
7	sign "o" the times	prince	10.0
8	1999	prince	10.0
9	purple rain	prince, the revolution	10.0
10	dirty mind	prince	10.0
11	off the wall	michael jackson	10.0
12	"heroes"	david bowie	10.0
13	low	david bowie	10.0
14	a love supreme: the complete masters	john coltrane	10.0
15	people's instinctive travels and the paths of ...	a tribe called quest	10.0
16	astral weeks	van morrison	10.0
17	loaded: re-loaded 45th anniversary edition	the velvet underground	10.0
18	sticky fingers	the rolling stones	10.0
19	it takes a nation of millions to hold us back	public enemy	10.0
20	the velvet underground 45th anniversary super...	the velvet underground	10.0
21	spiderland	slint	10.0
22	the infamous	mobb deep	10.0
23	white light/white heat	the velvet underground	10.0

	title	artist	score
24	in utero: 20th anniversary edition	nirvana	10.0
25	rumours	fleetwood mac	10.0
26	illmatic	nas	10.0
27	donuts (45 box set)	j dilla	10.0
28	voodoo	dangelo	10.0
29	the disintegration loops	william basinski	10.0
30	liquid swords: chess box deluxe edition	gza	10.0
31	isn't anything	my bloody valentine	10.0
32	tago mago [40th anniversary edition]	can	10.0
33	the smile sessions	the beach boys	10.0
34	laughing stock	talk talk	10.0
35	nevermind [20th anniversary edition]	nirvana	10.0
36	emergency & i [vinyl reissue]	the dismemberment plan	10.0
37	my beautiful dark twisted fantasy	kanye west	10.0
38	disintegration [deluxe edition]	the cure	10.0
39	exile on main st. [deluxe edition]	the rolling stones	10.0
40	quarantine the past	pavement	10.0
41	ladies and gentlemen we are floating in space ...	spiritualized	10.0
42	the stone roses	the stone roses	10.0
43	the beatles	the beatles	10.0
44	abbey road	the beatles	10.0
45	rubber soul	the beatles	10.0
46	revolver	the beatles	10.0
47	sgt. pepper's lonely hearts club band	the beatles	10.0

	title	artist	score
48	magical mystery tour	the beatles	10.0
49	stereo box	the beatles	10.0
50	kid a: special collectors edition	radiohead	10.0
51	reckoning [deluxe edition]	r.e.m.	10.0
52	histoire de melody nelson	serge gainsbourg	10.0
53	paul's boutique	beastie boys	10.0
54	murmur [deluxe edition]	r.e.m.	10.0
55	otis blue: otis redding sings soul [collector'...	otis redding	10.0
56	unknown pleasures	joy division	10.0
57	daydream nation: deluxe edition	sonic youth	10.0
58	pink flag	wire	10.0
59	born to run: 30th anniversary edition	bruce springsteen	10.0
60	in the aeroplane over the sea	neutral milk hotel	10.0
61	endtroducing... [deluxe edition]	dj shadow	10.0
62	crooked rain, crooked rain: la's desert origins	pavement	10.0
63	london calling: 25th anniversary legacy edition	the clash	10.0
64	music has the right to children	boards of canada	10.0
65	live at the apollo [expanded edition]	james brown	10.0
66	no thanks!: the 70s punk rebellion	various artists	10.0
67	marquee moon	television	10.0
68	the ascension	glenn branca	10.0
69	this year's model	elvis costello & the attractions	10.0
70	yankee hotel foxtrot	wilco	10.0
71	source tags and codes	...and you will know us by the trail of dead	10.0

	title	artist	score
72	the olatunji concert: the last live recording	john coltrane	10.0
73	kid a	radiohead	10.0
74	animals	pink floyd	10.0
75	i see a darkness	bonnie prince billy	10.0

Commit the changes and close the connection

In [ ]:
```
# commit the changes to the database
db_diana.commit()
# close the database connection
db_diana.close()
# close the pfork connection
pitchfork.close()
```

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. [2 points]

In [ ]:
```
#pip install mysql-connector-python
```

In [ ]:
```
import mysql.connector
```

In [ ]:
```
dotenv.load_dotenv("mod6.env")
mysqlpassword = os.getenv("mysqlpassword")
#mysqlpassword
```

In [ ]:
```
dbserver = mysql.connector.connect(
    user='root',
    passwd=mysqlpassword,
    host="localhost"
```

```
)
cursor = dbserver.cursor()
```

create the new database

In [ ]:
```
try:
    cursor.execute("CREATE DATABASE pitchfork_diana")
except:
    cursor.execute("DROP DATABASE pitchfork_diana")
    cursor.execute("CREATE DATABASE pitchfork_diana")
```

now that the database exists, get a connection to it

In [ ]:
```
from sqlalchemy import create_engine
```

In [ ]:
```
db_mysql_pitchfork_diana = create_engine("mysql+mysqlconnector://{user}:{pw}@localhost/{db}"
    .format(user="root", pw=mysqlpassword, db="pitchfork_diana"))
```

In [ ]:
```
# add the reviews, artists, content, genres, labels, and years tables to the database
reviews.to_sql('reviews', con = db_mysql_pitchfork_diana, index=False, chunksize=1000, if_exists = 'replace')
artists.to_sql('artists', con = db_mysql_pitchfork_diana, index=False, chunksize=1000, if_exists = 'replace')
content.to_sql('content', con = db_mysql_pitchfork_diana, index=False, chunksize=1000, if_exists = 'replace')
genres.to_sql('genres', con = db_mysql_pitchfork_diana, index=False, chunksize=1000, if_exists = 'replace')
labels.to_sql('labels', con = db_mysql_pitchfork_diana, index=False, chunksize=1000, if_exists = 'replace')
years.to_sql('years', con = db_mysql_pitchfork_diana, index=False, chunksize=1000, if_exists = 'replace')
```

Out[ ]:
```
19108
```

In [ ]:
```
mysql_db = mysql.connector.connect(
    user='root',
    passwd=mysqlpassword,
    host="localhost",
    database="pitchfork_diana"
)

cursor_mysql = mysql_db.cursor()
cursor_mysql.execute("SELECT title, artist, score FROM reviews WHERE score=10")
reviews_df = cursor_mysql.fetchall()
```

```
colnames = [x[0] for x in cursor_mysql.description]
pd.DataFrame(reviews_df, columns=colnames)
```

Out[ ]:

	title	artist	score
0	metal box	public image ltd	10.0
1	blood on the tracks	bob dylan	10.0
2	another green world	brian eno	10.0
3	songs in the key of life	stevie wonder	10.0
4	in concert	nina simone	10.0
5	tonight's the night	neil young	10.0
6	hounds of love	kate bush	10.0
7	sign "o" the times	prince	10.0
8	1999	prince	10.0
9	purple rain	prince, the revolution	10.0
10	dirty mind	prince	10.0
11	off the wall	michael jackson	10.0
12	"heroes"	david bowie	10.0
13	low	david bowie	10.0
14	a love supreme: the complete masters	john coltrane	10.0
15	people's instinctive travels and the paths of ...	a tribe called quest	10.0
16	astral weeks	van morrison	10.0
17	loaded: re-loaded 45th anniversary edition	the velvet underground	10.0
18	sticky fingers	the rolling stones	10.0
19	it takes a nation of millions to hold us back	public enemy	10.0
20	the velvet underground 45th anniversary super...	the velvet underground	10.0
21	spiderland	slint	10.0
22	the infamous	mobb deep	10.0
23	white light/white heat	the velvet underground	10.0

	title	artist	score
24	in utero: 20th anniversary edition	nirvana	10.0
25	rumours	fleetwood mac	10.0
26	illmatic	nas	10.0
27	donuts (45 box set)	j dilla	10.0
28	voodoo	dangelo	10.0
29	the disintegration loops	william basinski	10.0
30	liquid swords: chess box deluxe edition	gza	10.0
31	isn't anything	my bloody valentine	10.0
32	tago mago [40th anniversary edition]	can	10.0
33	the smile sessions	the beach boys	10.0
34	laughing stock	talk talk	10.0
35	nevermind [20th anniversary edition]	nirvana	10.0
36	emergency & i [vinyl reissue]	the dismemberment plan	10.0
37	my beautiful dark twisted fantasy	kanye west	10.0
38	disintegration [deluxe edition]	the cure	10.0
39	exile on main st. [deluxe edition]	the rolling stones	10.0
40	quarantine the past	pavement	10.0
41	ladies and gentlemen we are floating in space ...	spiritualized	10.0
42	the stone roses	the stone roses	10.0
43	the beatles	the beatles	10.0
44	abbey road	the beatles	10.0
45	rubber soul	the beatles	10.0
46	revolver	the beatles	10.0
47	sgt. pepper's lonely hearts club band	the beatles	10.0

	title	artist	score
48	magical mystery tour	the beatles	10.0
49	stereo box	the beatles	10.0
50	kid a: special collectors edition	radiohead	10.0
51	reckoning [deluxe edition]	r.e.m.	10.0
52	histoire de melody nelson	serge gainsbourg	10.0
53	paul's boutique	beastie boys	10.0
54	murmur [deluxe edition]	r.e.m.	10.0
55	otis blue: otis redding sings soul [collector'...	otis redding	10.0
56	unknown pleasures	joy division	10.0
57	daydream nation: deluxe edition	sonic youth	10.0
58	pink flag	wire	10.0
59	born to run: 30th anniversary edition	bruce springsteen	10.0
60	in the aeroplane over the sea	neutral milk hotel	10.0
61	endtroducing... [deluxe edition]	dj shadow	10.0
62	crooked rain, crooked rain: la's desert origins	pavement	10.0
63	london calling: 25th anniversary legacy edition	the clash	10.0
64	music has the right to children	boards of canada	10.0
65	live at the apollo [expanded edition]	james brown	10.0
66	no thanks!: the 70s punk rebellion	various artists	10.0
67	marquee moon	television	10.0
68	the ascension	glenn branca	10.0
69	this year's model	elvis costello & the attractions	10.0
70	yankee hotel foxtrot	wilco	10.0
71	source tags and codes	...and you will know us by the trail of dead	10.0

	title	artist	score
72	the olatunji concert: the last live recording	john coltrane	10.0
73	kid a	radiohead	10.0
74	animals	pink floyd	10.0
75	i see a darkness	bonnie prince billy	10.0

commit and close both the db servers that I opened.

In [ ]:
```
dbserver.commit()
dbserver.close()

mysql_db.commit()
mysql_db.close()
```

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. [2 points]

In [ ]:
```
# import postgresql
import psycopg2
```

I used the same password for all the databases.

In [ ]:
```
# I used the same password as for mysql
dbserver = psycopg2.connect(
    user='postgres',
    password=mysqlpassword,
    host="localhost"
)
# autocommit == True so we can create databases
dbserver.autocommit = True
```

In [ ]:
```
cursor = dbserver.cursor()
```

create the database

In [ ]:
```
try:
    cursor.execute("CREATE DATABASE pgres_pitchfork_diana")
except:
    cursor.execute("DROP DATABASE pgres_pitchfork_diana")
    cursor.execute("CREATE DATABASE pgres_pitchfork_diana")
```

connect to the new database

In [ ]:
```
postgres_db = psycopg2.connect(
    user='postgres',
    password=mysqlpassword,
    host="localhost",
    database="pgres_pitchfork_diana"
)
```

use the sql alchemy create_engine again to create tables from dataframes

In [ ]:
```
engine = create_engine("postgresql+psycopg2://{user}:{pw}@localhost/{db}"
        .format(user="postgres", pw=mysqlpassword, db="pgres_pitchfork_diana"))
```

In [ ]:
```
reviews.to_sql('reviews', con = engine, index=False, chunksize=1000, if_exists = 'replace')
artists.to_sql('artists', con = engine, index=False, chunksize=1000, if_exists = 'replace')
content.to_sql('content', con = engine, index=False, chunksize=1000, if_exists = 'replace')
genres.to_sql('genres', con = engine, index=False, chunksize=1000, if_exists = 'replace')
labels.to_sql('labels', con = engine, index=False, chunksize=1000, if_exists = 'replace')
years.to_sql('years', con = engine, index=False, chunksize=1000, if_exists = 'replace')
```

Out[ ]:
```
19108
```

create a cursor

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

```
colnames = [x[0] for x in cursor.description]
pd.DataFrame(reviews_df, columns=colnames)
```

Out[ ]:

	title	artist	score
0	metal box	public image ltd	10.0
1	blood on the tracks	bob dylan	10.0
2	another green world	brian eno	10.0
3	songs in the key of life	stevie wonder	10.0
4	in concert	nina simone	10.0
5	tonight's the night	neil young	10.0
6	hounds of love	kate bush	10.0
7	sign "o" the times	prince	10.0
8	1999	prince	10.0
9	purple rain	prince, the revolution	10.0
10	dirty mind	prince	10.0
11	off the wall	michael jackson	10.0
12	"heroes"	david bowie	10.0
13	low	david bowie	10.0
14	a love supreme: the complete masters	john coltrane	10.0
15	people's instinctive travels and the paths of ...	a tribe called quest	10.0
16	astral weeks	van morrison	10.0
17	loaded: re-loaded 45th anniversary edition	the velvet underground	10.0
18	sticky fingers	the rolling stones	10.0
19	it takes a nation of millions to hold us back	public enemy	10.0
20	the velvet underground 45th anniversary super...	the velvet underground	10.0
21	spiderland	slint	10.0
22	the infamous	mobb deep	10.0
23	white light/white heat	the velvet underground	10.0

	title	artist	score
24	in utero: 20th anniversary edition	nirvana	10.0
25	rumours	fleetwood mac	10.0
26	illmatic	nas	10.0
27	donuts (45 box set)	j dilla	10.0
28	voodoo	dangelo	10.0
29	the disintegration loops	william basinski	10.0
30	liquid swords: chess box deluxe edition	gza	10.0
31	isn't anything	my bloody valentine	10.0
32	tago mago [40th anniversary edition]	can	10.0
33	the smile sessions	the beach boys	10.0
34	laughing stock	talk talk	10.0
35	nevermind [20th anniversary edition]	nirvana	10.0
36	emergency & i [vinyl reissue]	the dismemberment plan	10.0
37	my beautiful dark twisted fantasy	kanye west	10.0
38	disintegration [deluxe edition]	the cure	10.0
39	exile on main st. [deluxe edition]	the rolling stones	10.0
40	quarantine the past	pavement	10.0
41	ladies and gentlemen we are floating in space ...	spiritualized	10.0
42	the stone roses	the stone roses	10.0
43	the beatles	the beatles	10.0
44	abbey road	the beatles	10.0
45	rubber soul	the beatles	10.0
46	revolver	the beatles	10.0
47	sgt. pepper's lonely hearts club band	the beatles	10.0

	title	artist	score
48	magical mystery tour	the beatles	10.0
49	stereo box	the beatles	10.0
50	kid a: special collectors edition	radiohead	10.0
51	reckoning [deluxe edition]	r.e.m.	10.0
52	histoire de melody nelson	serge gainsbourg	10.0
53	paul's boutique	beastie boys	10.0
54	murmur [deluxe edition]	r.e.m.	10.0
55	otis blue: otis redding sings soul [collector'...	otis redding	10.0
56	unknown pleasures	joy division	10.0
57	daydream nation: deluxe edition	sonic youth	10.0
58	pink flag	wire	10.0
59	born to run: 30th anniversary edition	bruce springsteen	10.0
60	in the aeroplane over the sea	neutral milk hotel	10.0
61	endtroducing... [deluxe edition]	dj shadow	10.0
62	crooked rain, crooked rain: la's desert origins	pavement	10.0
63	london calling: 25th anniversary legacy edition	the clash	10.0
64	music has the right to children	boards of canada	10.0
65	live at the apollo [expanded edition]	james brown	10.0
66	no thanks!: the 70s punk rebellion	various artists	10.0
67	marquee moon	television	10.0
68	the ascension	glenn branca	10.0
69	this year's model	elvis costello & the attractions	10.0
70	yankee hotel foxtrot	wilco	10.0
71	source tags and codes	...and you will know us by the trail of dead	10.0

	title	artist	score
72	the olatunji concert: the last live recording	john coltrane	10.0
73	kid a	radiohead	10.0
74	animals	pink floyd	10.0
75	i see a darkness	bonnie prince billy	10.0

commit and close

In [ ]:
```
postgres_db.commit()
postgres_db.close()
```

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 [ ]:
```
history = requests.get("https://history.muffinlabs.com/date", verify=False) # added verify = False to avoid ssl error
history_json = json.loads(history.text)
events = history_json['data']['Events']
events[0]
```

```
c:\Users\dianam\Anaconda3\envs\ds6001\lib\site-packages\urllib3\connectionpool.py:1043: InsecureRequestWarning: Unve
rified HTTPS request is being made to host 'history.muffinlabs.com'. Adding certificate verification is strongly adv
ised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
  warnings.warn(
```

Out[ ]:

```
{'year': '452 or 453',
 'text': 'Severianus, Bishop of Scythopolis, is martyred in Palestine.',
 'html': '452 or 453 - <a href="https://wikipedia.org/wiki/452" title="452">452</a> or <a href="https://wikipedia.org/wiki/453" title="453">453</a> – <a href="https://wikipedia.org/wiki/Severianus,_Bishop_of_Scythopolis" title="Severianus, Bishop of Scythopolis">Severianus, Bishop of Scythopolis</a>, is martyred in Palestine.',
 'no_year_html': '<a href="https://wikipedia.org/wiki/452" title="452">452</a> or <a href="https://wikipedia.org/wiki/453" title="453">453</a> – <a href="https://wikipedia.org/wiki/Severianus,_Bishop_of_Scythopolis" title="Severianus, Bishop of Scythopolis">Severianus, Bishop of Scythopolis</a>, is martyred in Palestine.',
 'links': [{'title': '452', 'link': 'https://wikipedia.org/wiki/452'},
  {'title': '453', 'link': 'https://wikipedia.org/wiki/453'},
  {'title': 'Severianus, Bishop of Scythopolis',
   'link': 'https://wikipedia.org/wiki/Severianus,_Bishop_of_Scythopolis'}]}
```


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. [2 points]

In [ ]:
```
import pymongo
```

In [ ]:
```
# connect to my local MongoDB and create a new database called "history" My localhost is locationlocal:27017
mongodb_client = pymongo.MongoClient("mongodb://localhost:27017/")
```

In [ ]:
```
historyb = mongodb_client["history"]
todaycollection = historyb["today"]
```

Now make the collection within the database named "today" and insert all the records in the events into this collection.

In [ ]:
```
allevents = todaycollection.insert_many(events)
```

In [ ]:
```
todaycollection.count_documents({})
```

Out[ ]:
```
44
```

In [ ]:
```
from bson.json_util import dumps, loads
```

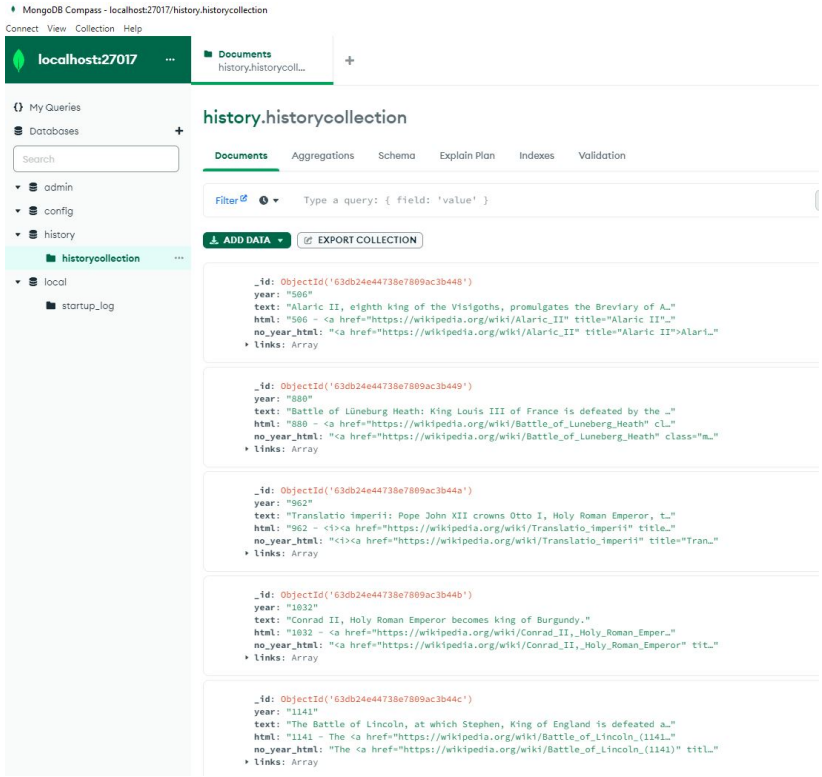
I selected the word "Roman"

In [ ]:
```
myquery = { 'text': {"$regex": 'Roman'}}
query_results = todaycollection.find(myquery)

print("Number of matching documents: ", query_results.collection.count_documents(myquery))

#myhistory_text = dumps(myhistory)
#print(myhistory_text)
```

```
Number of matching documents:  1
```

In [ ]:
```
# load the test into a JSON object
#history_json = loads(myhistory_text)
#history_json
```

The historycollection does exist. Here is a screenshot of the MongoDB GUI interface with the collection:



In [ ]:
```
# close the client
mongodb_client.close()
```