labassignment7

July 25, 2022

1 Lab Assignment 7: Database Queries

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.

1.1.2 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:

[1]:
```
import numpy as np
import pandas as pd
import sys
import os
import requests
import psycopg2
import pymongo
import json
from bson.json_util import dumps, loads
from sqlalchemy import create_engine
import dotenv

# change to the directory where your .env file is
os.chdir(r"C:\Users\kimbrelljm17\OneDrive - Grove City␣
 ↪College\Documents\UVA\DS6100\M07")

dotenv.load_dotenv('postgres.env') # register the .env file where passwords are␣
 ↪stored
sys.tracebacklimit = 0 # turn off the error tracebacks
```

1.1.3 Problem 1

For this problem, we will be building a PostgreSQL database that contains the collected works of Shakespeare.

The data were collected by Catherine Devlin from the repository at https://opensourceshakespeare.org/. The database will have four tables, one representing works by Shakespeare, one for characters that appear in Shakespeare's plays, one for chapters (this is, scenes within acts), and one for paragraphs (that is, lines of dialogue). The data to populate these four tables are here:

[2]:
```
works = pd.read_csv("https://github.com/jkropko/DS-6001/raw/master/localdata/
↪Works.csv")
characters = pd.read_csv("https://github.com/jkropko/DS-6001/raw/master/
↪localdata/Characters.csv")
chapters = pd.read_csv("https://github.com/jkropko/DS-6001/raw/master/localdata/
↪Chapters.csv")
paragraphs = pd.read_csv("https://github.com/jkropko/DS-6001/raw/master/
↪localdata/Paragraphs.csv")
```

In PostgreSQL, it is best practice to convert all column names to lower-case, as case sensitive column names will require extraneous double-quotes in any query. We first convert the column names in all four dataframe to lowercase:

[3]:
```
works.columns = works.columns.str.lower()
characters.columns = characters.columns.str.lower()
chapters.columns = chapters.columns.str.lower()
paragraphs.columns = paragraphs.columns.str.lower()
```

You will build a database and populate it with these data. The ER diagram for the database is:

There's no codebook, unfortunately, but the values in the columns are mostly self-explanatory:

[4]:
```
works.head()
```

[4]:
```
         workid                       title                        longtitle  \
0       12night               Twelfth Night  Twelfth Night, Or What You Will
1      allswell  All's Well That Ends Well        All's Well That Ends Well
2    antonycleo        Antony and Cleopatra             Antony and Cleopatra
3   asyoulikeit                 As You Like It                   As You Like It
4  comedyerrors            Comedy of Errors            The Comedy of Errors

   date genretype  notes     source  totalwords  totalparagraphs
0  1599         c    NaN       Moby       19837             1031
1  1602         c    NaN       Moby       22997             1025
2  1606         t    NaN       Moby       24905             1344
3  1599         c    NaN  Gutenberg       21690              872
4  1589         c    NaN       Moby       14692              661
```

[5]:
```
characters.head()
```

[5]:
```
              charid          charname             abbrev        works  \
0  1apparition-mac  First Apparition   First Apparition      macbeth
1         1citizen     First Citizen      First Citizen  romeojuliet
```

```
2     1conspirator  First Conspirator  First Conspirator   coriolanus
3   1gentleman-oth    First Gentleman    First Gentleman      othello
4            1goth         First Goth         First Goth        titus

  description  speechcount
0         NaN          1.0
1         NaN          3.0
2         NaN          3.0
3         NaN          1.0
4         NaN          4.0
```

[6]:
```
chapters.head()
```

[6]:
```
    workid  chapterid  section  chapter            description
0  12night    18704.0      1.0      1.0  DUKE ORSINO's palace.
1  12night    18705.0      1.0      2.0         The sea-coast.
2  12night    18706.0      1.0      3.0        OLIVIA'S house.
3  12night    18707.0      1.0      4.0  DUKE ORSINO's palace.
4  12night    18708.0      1.0      5.0        OLIVIA'S house.
```

[7]:
```
paragraphs.head()
```

[7]:
```
    workid paragraphid paragraphnum  charid  \
0  12night      630863            3     xxx
1  12night      630864            4  ORSINO
2  12night      630865           19   CURIO
3  12night      630866           20  ORSINO
4  12night      630867           21   CURIO

                                           plaintext  \
0  [Enter DUKE ORSINO, CURIO, and other Lords; Mu…
1  If music be the food of love, play on;\n[p]Giv…
2                     Will you go hunt, my lord?\n
3                                  What, Curio?\n
4                                     The hart.\n

                                        phonetictext  \
0         ENTR TK ORSN KR ANT OOR LRTS MSXNS ATNTNK
1  IF MSK B 0 FT OF LF PL ON JF M EKSSS OF IT OT …
2                                 WL Y K HNT M LRT
3                                            HT KR
4                                            0 HRT

                                            stemtext paragraphtype  section  \
0  enter duke orsino curio and other lord musicia…             b      1.0
1  if music be the food of love plai on give me e…             b      1.0
2                           will you go hunt my lord             b      1.0
```

```
3                                             what curio              b       1.0
4                                               the hart              b       1.0

   chapter  charcount  wordcount
0      1.0       65.0        9.0
1      1.0      646.0      114.0
2      1.0       27.0        6.0
3      1.0       13.0        2.0
4      1.0       10.0        2.0
```

Part a Connect to your local PostgreSQL server (take steps to hide your password!), create a new database for the Shakespeare data, use `create_engine()` from `sqlalchemy` to connect to the database, and create the works, characters, chapters, and paragraphs tables populated with the data from the four dataframes shown above. [2 points]

[8]:
```
dotenv.load_dotenv('postgres.env')
pgpassword = os.getenv("pgpassword")
dbserver = psycopg2.connect(
    user='postgres',
    password=pgpassword,
    host="localhost"
)
dbserver.autocommit = True
```

[9]:
```
cursor = dbserver.cursor()
try:
    cursor.execute("CREATE DATABASE shakespeare")
except:
    cursor.execute("DROP DATABASE shakespeare")
    cursor.execute("CREATE DATABASE shakespeare")
```

[10]:
```
engine = create_engine('postgresql+psycopg2://{user}:{pw}@localhost/{db}'\
                       .format(user='postgres',pw=pgpassword,db='shakespeare'))
```

[11]:
```
works.to_sql('works',con=engine,index=False,chunksize=1000, if_exists='replace')
characters.to_sql('characters',con=engine,index=False,chunksize=1000,␣
 ↪if_exists='replace')
chapters.to_sql('chapters',con=engine,index=False,chunksize=1000,␣
 ↪if_exists='replace')
paragraphs.to_sql('paragraphs',con=engine,index=False,chunksize=1000,␣
 ↪if_exists='replace')
```

Part b Write a query to display `title`, `date`, and `totalwords` from the `works` table. Rename `date` to `year`, and sort the output by `totalwords` in descending order. Also create a new column called `era` which is equal to "early" for works created before 1600, "middle" for works created between 1600 and 1607, and "late" for works created after 1607. Finally, display only the 7th through 11th rows of the output data. [1 point]

```
[12]: myquery = """
      SELECT title,date AS year,
             CASE WHEN date <  1600 THEN 'early'
                  WHEN date >= 1600 AND date <= 1607 THEN 'middle'
                  ELSE 'late'
             END AS era,
             totalwords
      FROM works
      ORDER BY works.totalwords DESC
      LIMIT 5
      OFFSET 6
      """
      pd.read_sql_query(myquery,con=engine)
```

```
[12]:                    title  year     era  totalwords
      0              King Lear  1605  middle       26119
      1  Troilus and Cressida  1601  middle       26089
      2      Henry IV, Part II  1597   early       25692
      3      Henry VI, Part II  1590   early       25411
      4      The Winter's Tale  1610    late       24914
```

Part c The `genretype` column in the "works" table designates five types of Shakespearean work:

- `t` is a tragedy, such as *Romeo and Juliet* and *Hamlet*
- `c` is a comedy, such as *A Midsummer Night's Dream* and *As You Like It*
- `h` is a history, such as *Henry V* and *Richard III*
- `s` refers to Shakespeare's sonnets
- `p` is a narrative (non-sonnet) poem, such as *Venus and Adonis* and *Passionate Pilgrim*

Write a query that generates a table that reports the average number of words in Shakepeare's works by genre type. Display the genre type and the average wordcount within genre, use appropriate aliases, and sort by the average in descending order. [1 point]

```
[13]: myquery = """
      SELECT genretype,ROUND(AVG(totalwords),0) as avgtotalwords
      FROM works
      GROUP BY works.genretype
      ORDER BY AVG(totalwords) DESC
      """
      pd.read_sql_query(myquery,con=engine)
```

```
[13]:   genretype  avgtotalwords
      0         h        24236.0
      1         t        23817.0
      2         c        20212.0
      3         s        17515.0
      4         p         6182.0
```

Part d Use a query to generate a table that contains the text of Hamlet's (the character, not just the play) longest speech, and use the `print()` function to display this text. [1 point]

```
[14]: myquery = """
SELECT plaintext
FROM characters c
INNER JOIN paragraphs p
    ON c.charid = p.charid
WHERE c.charname = 'Hamlet' AND p.wordcount != 'NaN'
ORDER BY p.wordcount DESC
LIMIT 1
"""
longest_speech = pd.read_sql_query(myquery,con=engine)
print(longest_speech.values)
```

```
[["Ay, so, God b' wi' ye!                              [Exeunt Rosencrantz and
Guildenstern\n[p]Now I am alone. \n[p]O what a rogue and peasant slave am
I!\n[p]Is it not monstrous that this player here,\n[p]But in a fiction, in a
dream of passion,\n[p]Could force his soul so to his own conceit\n[p]That, from
her working, all his visage wann'd,\n[p]Tears in his eyes, distraction in's
aspect,\n[p]A broken voice, and his whole function suiting\n[p]With forms to his
conceit? And all for nothing!\n[p]For Hecuba!\n[p]What's Hecuba to him, or he to
Hecuba,\n[p]That he should weep for her? What would he do,\n[p]Had he the motive
and the cue for passion\n[p]That I have? He would drown the stage with
tears\n[p]And cleave the general ear with horrid speech;\n[p]Make mad the guilty
and appal the free,\n[p]Confound the ignorant, and amaze indeed\n[p]The very
faculties of eyes and ears.\n[p]Yet I,\n[p]A dull and muddy-mettled rascal,
peak\n[p]Like John-a-dreams, unpregnant of my cause, \n[p]And can say nothing!
No, not for a king,\n[p]Upon whose property and most dear life\n[p]A damn'd
defeat was made. Am I a coward?\n[p]Who calls me villain? breaks my pate
across?\n[p]Plucks off my beard and blows it in my face?\n[p]Tweaks me by th'
nose? gives me the lie i' th' throat\n[p]As deep as to the lungs? Who does me
this, ha?\n[p]'Swounds, I should take it! for it cannot be\n[p]But I am pigeon-
liver'd and lack gall\n[p]To make oppression bitter, or ere this\n[p]I should
have fatted all the region kites\n[p]With this slave's offal. Bloody bawdy
villain!\n[p]Remorseless, treacherous, lecherous, kindless villain!\n[p]O,
vengeance!\n[p]Why, what an ass am I! This is most brave,\n[p]That I, the son of
a dear father murther'd,\n[p]Prompted to my revenge by heaven and hell,\n[p]Must
(like a whore) unpack my heart with words\n[p]And fall a-cursing like a very
drab,\n[p]A scullion! \n[p]Fie upon't! foh! About, my brain! Hum, I have
heard\n[p]That guilty creatures, sitting at a play,\n[p]Have by the very cunning
of the scene\n[p]Been struck so to the soul that presently\n[p]They have
proclaim'd their malefactions;\n[p]For murther, though it have no tongue, will
speak\n[p]With most miraculous organ, I'll have these Players\n[p]Play something
like the murther of my father\n[p]Before mine uncle. I'll observe his
looks;\n[p]I'll tent him to the quick. If he but blench,\n[p]I know my course.
The spirit that I have seen\n[p]May be a devil; and the devil hath power\n[p]T'
assume a pleasing shape; yea, and perhaps\n[p]Out of my weakness and my
melancholy,\n[p]As he is very potent with such spirits,\n[p]Abuses me to damn
me. I'll have grounds\n[p]More relative than this. The play's the
thing\n[p]Wherein I'll catch the conscience of the King. Exit.\n"]]
```

1.1.4 Part e

Many scenes in Shakespeare's works take place in palaces or castles. Use a query to create a table that lists all of the chapters that take place in a palace. Include the work's title, the section (renamed to "act"), the chapter (renamed to "scene"), and the description of these chapters. The setting of each scene is listed in the `description` column of the "chapters" table. [Hint: be sure to account for case sensitivity] [2 points]

[15]:
```
myquery = """
SELECT w.title,c.section as act,c.chapter as scene,c.description
FROM chapters c
INNER JOIN works w
 ON c.workid = w.workid
WHERE description LIKE '%%_alace%%'
"""
pd.read_sql_query(myquery,con=engine)
```

[15]:
```
                          title  act  scene  \
0                  Twelfth Night  2.0    4.0
1                  Twelfth Night  1.0    4.0
2                  Twelfth Night  1.0    1.0
3    All's Well That Ends Well  5.0    3.0
4    All's Well That Ends Well  5.0    2.0
..                          ...  ...    ...
120           The Winter's Tale  5.0    1.0
121           The Winter's Tale  4.0    2.0
122           The Winter's Tale  2.0    3.0
123           The Winter's Tale  2.0    1.0
124           The Winter's Tale  1.0    1.0

                            description
0                 DUKE ORSINO's palace.
1                 DUKE ORSINO's palace.
2                 DUKE ORSINO's palace.
3           Rousillon. The COUNT's palace.
4    Rousillon. Before the COUNT's palace.
..                                   ...
120           A room in LEONTES' palace.
121      Bohemia. The palace of POLIXENES.
122           A room in LEONTES' palace.
123           A room in LEONTES' palace.
124        Antechamber in LEONTES' palace.

[125 rows x 4 columns]
```

1.1.5 Part f

Create a table that lists characters, the plays that the characters appear in, the number of speeches the character gives, and the average length of the speeches that the character gives. Display the character description and the work title, not the ID values. Sort the table by average speech length, and restrict the table to only those characters that give at least 20 speeches. [Hint: you will need to use a subquery.] [2 points]

[16]:
```
myquery = """
WITH a AS (
    SELECT c.charid, c.charname, c.description, c.speechcount, AVG(p.wordcount)␣
 ↪as averagewc, c.works
    FROM characters c
    INNER JOIN paragraphs p
        ON c.charid = p.charid
    GROUP BY c.charid, c.charname, c.description, c.speechcount, c.works
    ORDER BY averagewc
)
SELECT b.title, a.charname, a.description, a.speechcount, a.averagewc
FROM a
INNER JOIN works b
    ON a.works = b.workid
WHERE a.speechcount >= 20
ORDER BY a.averagewc DESC
"""

pd.read_sql_query(myquery, con=engine)
```

[16]:
```
                        title             charname  \
0                  Richard II      King Richard II
1                  Henry VIII      Queen Katharine
2                   King John           Constance
3                  Henry VIII  Duke of Buckingham
4    Midsummer Night's Dream               Oberon
..                        …                    …
337                   Macbeth      First Murderer
338       Taming of the Shrew              Curtis
339             Julius Caesar              Lucius
340                   Henry V               Alice
341             As You Like It  (stage directions)

                              description  speechcount  averagewc
0                         king of England         98.0  61.765306
1    wife to King Henry, afterwards divorced         50.0  59.360000
2                        mother to Arthur         36.0  59.222222
3                                    None         26.0  57.307692
4                    king of the fairies         29.0  55.655172
..                                     …          …          …
```

```
337                                          None         21.0    8.666667
338                                          None         20.0    8.550000
339                             servant to Brutus         24.0    8.541667
340   a lady attending on Princess Katherine         22.0    7.454545
341                                          None        126.0    4.309517

[342 rows x 5 columns]
```

1.1.6 Part g

Which Shakepearean works do not contain any scenes in a palace or a castle? Use a query that displays the title, genre type, and publication date of works that do not contain any scenes that take place in a palace or castle. [Hint: use your work in part e as a starting point. You will need a subquery, and you will need to think carefully about the type of join that you need to perform.][2 points]

[19]:
```
myquery = """
WITH pal_cas AS(
    SELECT workid, SUM(CAST((description LIKE '%%_alace%%' OR description LIKE␣
 ↪'%%_astle%%') AS INT)) AS nopalcast
    FROM chapters
    GROUP BY workid
    HAVING SUM(CAST((description LIKE '%%_alace%%' OR description LIKE␣
 ↪'%%_astle%%') AS INT)) = 0
)
SELECT w.title, w.genretype, w.date
FROM pal_cas p
LEFT JOIN works w
    ON p.workid = w.workid
ORDER BY date, title
"""
pd.read_sql_query(myquery, con=engine)
```

[19]:
```
                     title genretype  date
0      Taming of the Shrew         c  1593
1         Venus and Adonis         p  1593
2       Love's Labour's Lost         c  1594
3          Rape of Lucrece         p  1594
4         Romeo and Juliet         t  1594
5       Merchant of Venice         c  1596
6     Much Ado about Nothing         c  1598
7       Passionate Pilgrim         p  1598
8            Julius Caesar         t  1599
9     Merry Wives of Windsor         c  1600
10    Phoenix and the Turtle         p  1601
11              Coriolanus         t  1607
12          Timon of Athens         t  1607
```

```
13      Lover's Complaint        p  1609
14                Sonnets        s  1609
15                Tempest        c  1611
```

1.1.7 Problem 2

The following file contains JSON formatted data of the official English-language translations of every constitution currently in effect in the world:

[17]:
```
const = requests.get("https://github.com/jkropko/DS-6001/raw/master/localdata/
 ↪const.json")
const_json = json.loads(const.text)
pd.DataFrame.from_records(const_json)
```

```
ERROR! Session/line number was not unique in database. History logging moved to
new session 390
```

[17]:
```
                                                    text             country  \
0    'Afghanistan 2004     Preamble    \nIn the na…          Afghanistan
1    'Albania 1998 (rev. 2012)     Preamble    \nWe…             Albania
2    'Andorra 1993     Preamble    \nThe Andorran P…             Andorra
3    'Angola 2010     Preamble    \nWe, the people …              Angola
4    'Antigua and Barbuda 1981     Preamble    \nWH…  Antigua and Barbuda
..                                                 …                   …
140  'Uzbekistan 1992 (rev. 2011)     Preamble    \…          Uzbekistan
141  'Viet Nam 1992 (rev. 2013)     Preamble    \nI…            Viet Nam
142  'Yemen 1991 (rev. 2001)     PART ONE. THE FOUN…               Yemen
143  'Zambia 1991 (rev. 2009)     Preamble    \nWE,…              Zambia
144  'Zimbabwe 2013     Preamble    \nWe the people…            Zimbabwe

    adopted  revised  reinstated  democracy
0      2004      NaN         NaN   0.372201
1      1998   2012.0         NaN   0.535111
2      1993      NaN         NaN        NaN
3      2010      NaN         NaN   0.315043
4      1981      NaN         NaN        NaN
..      …        …          …         …
140    1992   2011.0         NaN   0.195932
141    1992   2013.0         NaN   0.251461
142    1991   2001.0         NaN   0.125708
143    1991   2009.0         NaN   0.405497
144    2013      NaN         NaN   0.315359

[145 rows x 6 columns]
```

The text of the constitutions are available from the Wolfram Data Repository. I also included scores that represent the level of democractic quality in each country as of 2016. These scores are compiled by the Varieties of Democracy (V-Dem) project. Higher scores indicate greater levels of democratic openness and competition.

Part a Connect to your local MongoDB server and create a new collection for the constitution data. Use `.delete_many({})` to remove any existing data from this collection, and insert the data in `const_json` into this collection. [2 points]

[20]:
```
myclient = pymongo.MongoClient("mongodb://localhost/")
condb = myclient["condb"]
concollection = condb["concollection"]
concollection.delete_many({})
concollection.insert_many(const_json)
```

```
ERROR! Session/line number was not unique in database. History logging moved to
new session 393
```

[20]:
```
<pymongo.results.InsertManyResult at 0x20094177f08>
```

Part b Use MongoDB queries and the `dumps()` and `loads()` functions from the `bson` package to produce dataframes with the following restrictions:

- The country, adoption year, and democracy features (and not `_id`, text, revised, or reinstated) for countries with constitutions that were written after 1990
- The country, adoption year, and democracy features (and not `_id`, text, revised, or reinstated) for countries with constitutions that were written after 1990 AND have a democracy score of less than 0.5
- The country, adoption year, and democracy features (and not `_id`, text, revised, or reinstated) for countries with constitutions that were written after 1990 OR have a democracy score of less than 0.5

[1 point]

[24]:
```
cursor = concollection.find({'adopted': {'$gt':1990}},
                             {'country': 1,
                              'adopted' :1,
                             'democracy': 1,
                             '_id': 0})
qtext = dumps(cursor)
qrec = loads(qtext)
pd.DataFrame.from_records(qrec)
```

[24]:
```
         country  adopted  democracy
0    Afghanistan     2004   0.372201
1        Albania     1998   0.535111
2        Andorra     1993        NaN
3         Angola     2010   0.315043
4        Armenia     1995   0.393278
..           …        …        …
66    Uzbekistan     1992   0.195932
67      Viet Nam     1992   0.251461
68         Yemen     1991   0.125708
69        Zambia     1991   0.405497
```

```
70     Zimbabwe     2013   0.315359

[71 rows x 3 columns]
```

[27]:
```
cursor = concollection.find({'adopted': {'$gt':1990},'democracy' : {'$lt':.5}},
                             {'country': 1,
                              'adopted' :1,
                             'democracy': 1,
                             '_id': 0})
qtext = dumps(cursor)
qrec = loads(qtext)
pd.DataFrame.from_records(qrec)
```

[27]:
```
                             country  adopted  democracy
0                        Afghanistan     2004   0.372201
1                             Angola     2010   0.315043
2                            Armenia     1995   0.393278
3                            Belarus     1994   0.289968
4             Bosnia and Herzegovina     1995   0.338267
5                           Cambodia     1993   0.313738
6                              Egypt     2014   0.218600
7                  Equatorial Guinea     1991   0.217861
8                            Eritrea     1997   0.075621
9                           Ethiopia     1994   0.254865
10                              Fiji     2013   0.473559
11                            Gambia     1996   0.348132
12                              Iraq     2005   0.455402
13                        Kazakhstan     1995   0.262596
14  Lao People's Democratic Republic     1991   0.094434
15                             Libya     2011   0.294716
16                          Maldives     2008   0.386754
17                        Montenegro     2007   0.455338
18                           Myanmar     2008   0.405772
19                              Oman     1996   0.191211
20                Russian Federation     1993   0.275516
21                            Rwanda     2003   0.274476
22                      Saudi Arabia     1992   0.024049
23                            Serbia     2006   0.474443
24                           Somalia     2012   0.177772
25                       South Sudan     2011   0.183267
26                             Sudan     2005   0.311799
27                         Swaziland     2005   0.136008
28              Syrian Arab Republic     2012   0.148212
29                      Turkmenistan     2008   0.154887
30                            Uganda     1995   0.338308
31                           Ukraine     1996   0.361911
32                        Uzbekistan     1992   0.195932
```

```
33                            Viet Nam     1992    0.251461
34                               Yemen     1991    0.125708
35                              Zambia     1991    0.405497
36                            Zimbabwe     2013    0.315359
```

[30]:
```
cursor = concollection.find({'$or': [{'adopted': {'$gt':1990}}, {'democracy':␣
 ↪{'$lt': .5}}]},
                             {'country': 1,
                              'adopted' :1,
                             'democracy': 1,
                             '_id': 0})
qtext = dumps(cursor)
qrec = loads(qtext)
pd.DataFrame.from_records(qrec)
```

[30]:
```
        country  adopted  democracy
0   Afghanistan     2004   0.372201
1       Albania     1998   0.535111
2       Andorra     1993        NaN
3        Angola     2010   0.315043
4       Armenia     1995   0.393278
..          …        …         …
78   Uzbekistan     1992   0.195932
79     Viet Nam     1992   0.251461
80        Yemen     1991   0.125708
81       Zambia     1991   0.405497
82     Zimbabwe     2013   0.315359

[83 rows x 3 columns]
```

```
ERROR! Session/line number was not unique in database. History logging moved to
new session 398
```

Part c According to the Varieties of Democracy project, Hungary has become less democratic over the last few years, and can no longer be considered a democracy. Update the record for Hungary to set the democracy score at 0.4. Then query the database to extract the record for Hungary and display the data in a dataframe. [1 point]

[32]:
```
def mongo_read_query(col, q):
    qtext = dumps(col.find(q))
    qrec = loads(qtext)
    qdf = pd.DataFrame.from_records(qrec)
    return qdf
concollection.update_one({'country': 'Hungary'},
                     {'$set' : {'democracy': 0.4}})
mongo_read_query(concollection, {'country': 'Hungary'})
```

```
[32]:                          _id  \
      0  62d99a0c94b08b39b74309e0

                                                    text  country  adopted  \
      0  'Hungary 2011 (rev. 2013)     Preamble    \nGo…  Hungary     2011

         revised reinstated  democracy
      0   2013.0       None        0.4
```

Part d Set the `text` field in the database as a text index. Then query the database to find all constitutions that contain the exact phrase "freedom of speech". Display the country name, adoption year, and democracy scores in a dataframe for the constitutions that match this query. [2 points]

```
[37]: concollection.create_index([('text', 'text')])
      cursor = concollection.find({'$text': {'$search':'freedom of speech',␣
       ↪'$caseSensitive': False}},
                                  {'country': 1,
                                   'adopted' :1,
                                  'democracy': 1,
                                  '_id': 0})
      qtext = dumps(cursor)
      qrec = loads(qtext)
      pd.DataFrame.from_records(qrec)
```

```
[37]:                       country  adopted  democracy
      0                Turkmenistan     2008   0.154887
      1                      Sweden     1974   0.902575
      2                    Slovenia     1991   0.861380
      3                      Poland     1997   0.682208
      4      Bosnia and Herzegovina     1995   0.338267
      ..                        …        …         …
      140               Netherlands     1815   0.859255
      141                   Denmark     1953   0.883552
      142  United States of America     1789   0.849155
      143                 Australia     1901   0.879540
      144         Brunei Darussalam     1959        NaN

      [145 rows x 3 columns]
```

Part e Use a query to search for the terms "freedom", "liberty", "legal", "justice", and "rights". Generate a text score for all of the countries, and display the data for the countries with the top 10 relevancy scores in a dataframe. [2 points]

```
[44]: cursor = concollection.find(
                  {'$text': {'$search': 'freedom liberty legal justice rights'}},
                  {'score': {'$meta': 'textScore'}})
```

```
cursor.sort([('score', {'$meta': 'textScore'})])
qtext = dumps(cursor)
qrec = loads(qtext)
df = pd.DataFrame.from_records(qrec)
df.head(10)
```

[44]:

```
                        _id  \
0  62d99a0c94b08b39b7430a1f
1  62d99a0c94b08b39b74309d7
2  62d99a0c94b08b39b74309d4
3  62d99a0c94b08b39b74309b3
4  62d99a0c94b08b39b74309af
5  62d99a0c94b08b39b74309ce
6  62d99a0c94b08b39b7430a02
7  62d99a0c94b08b39b74309d1
8  62d99a0c94b08b39b74309da
9  62d99a0c94b08b39b7430a33

                                                text                country  \
0  'Serbia 2006     Preamble    \nConsidering the…                 Serbia
1  'Finland 1999 (rev. 2011)     Chapter 1. Funda…                Finland
2  'Estonia 1992 (rev. 2011)     Preamble     \nWi…                Estonia
3  'Armenia 1995 (rev. 2005)     Preamble     \nTh…                Armenia
4  'Albania 1998 (rev. 2012)     Preamble     \nWe…                Albania
5  'Dominican Republic 2015     Preamble    \nWe,…     Dominican Republic
6  'Moldova (Republic of) 1994 (rev. 2006)     Pr…  Moldova (Republic of)
7  'El Salvador 1983 (rev. 2014)     TITLE I      …            El Salvador
8  'Georgia 1995 (rev. 2013)     Preamble     \nWe…                Georgia
9  'Turkey 1982 (rev. 2011)     Preamble    \nAff…                 Turkey

   adopted  revised  reinstated  democracy     score
0     2006      NaN         NaN   0.474443  5.030999
1     1999   2011.0         NaN   0.856265  5.029000
2     1992   2011.0         NaN   0.909233  5.024473
3     1995   2005.0         NaN   0.393278  5.023651
4     1998   2012.0         NaN   0.535111  5.023087
5     2015      NaN         NaN   0.583654  5.019910
6     1994   2006.0         NaN   0.571357  5.017063
7     1983   2014.0         NaN   0.661989  5.016899
8     1995   2013.0         NaN   0.757486  5.015282
9     1982   2011.0         NaN   0.341745  5.014672
```

1.1.8 Question 3

Close the connections to the PostgreSQL and MongoDB databases. [1 point]

[45]:

```
dbserver.close()
myclient.close()
```