Databases-Fall2020-FinalProject

November 23, 2021

0.1 DATA 311 - Fall 2020

0.1.1 Final Project - due Tuesday, Nov 24 by midnight

[73]:
```
import sqlite3
import pandas as pd
```

[74]:
```
conn = sqlite3.connect('Covid.db')
curs = conn.cursor()
curs.execute("PRAGMA foreign_keys=ON;")
```

[74]:
```
<sqlite3.Cursor at 0x7f9236e1b810>
```

[75]:
```
data1=pd.read_csv("./data/StayAtHome_KFF_Clean.csv")
data2=pd.read_csv("./data/states.csv")
data3=pd.read_csv("./data/PopDensity.csv")
data4=pd.read_csv("./data/CovidDailyTracking_Filtered_Nov16.csv")
data5=pd.read_csv("./data/Ballotpedia_StateEmergency.csv")
```

[76]:
```
curs.execute("DROP TABLE IF EXISTS tState;")
# Create the state table
sql = """CREATE TABLE tState (
            state TEXT PRIMARY KEY,
            st TEXT NOT NULL UNIQUE);"""
curs.execute(sql)

# Load the state data
tState = pd.read_csv('./data/states.csv')
sql = "INSERT INTO tState VALUES (?,?);"
for row in tState.values:
    curs.execute(sql, (row[0], row[1]))
```

[77]:
```
curs.execute("DROP TABLE IF EXISTS tStayAtHome;")
#Creat the stay at home table (columns are when stay at home order was␣
 ↪announced and when it was implemented: has 42 observations)
sql="""CREATE TABLE tStayAtHome (
            State TEXT REFERENCES tState(state),
            `Date Announced` DATE,
            `Effective Date` DATE);"""
curs.execute(sql)

#Load the stay at home data
tStayAtHome = pd.read_csv("./data/StayAtHome_KFF_Clean.csv")
sql = "INSERT INTO tStayAtHome VALUES (?,?,?);"
for row in tStayAtHome.values:
    curs.execute(sql, tuple(row))
```

[78]:
```
curs.execute("DROP TABLE IF EXISTS tStateInfo;")
#Create the table that has general inforamtiuon about the state like pop and␣
 ↪land size
sql="""CREATE TABLE tStateInfo (
            State TEXT REFERENCES tState(state),
            pop2019est INTEGER,
            LandSqMi INTEGER,
            LandSqKM INTEGER,
            PopPerSqMi INTEGER,
            PopPerSqKM INTEGER);"""
curs.execute(sql)

#Load the stay at home data
tStateInfo = pd.read_csv("./data/PopDensity.csv")
sql = "INSERT INTO tStateInfo VALUES (?,?,?,?,?,?);"
for row in tStateInfo.values:
    curs.execute(sql, tuple(row))
```

[79]:
```
curs.execute("DROP TABLE IF EXISTS tDailyTracker;")
#Create the daily tracker table
sql="""CREATE TABLE tDailyTracker (
            st TEXT REFERENCES tState(st),
            date_clean DATE,
            positive INTEGER,
            negative INTEGER,
            death INTEGER,
            recovered INTEGER,
            positiveIncrease INTEGER,
            negativeIncrease INTEGER,
            deathIncrease INTEGER,
            totalTestResults INTEGER,
            totalTestResultsIncrease INTEGER,
            hosptitalizedCurrently INTEGER,
            hospitalizedCumulative INTEGER,
            hospitalizedIncrease INTEGER,
            inIcuCurrently INTEGER,
            inIcuCumulative INTEGER,
            onVentilatorCurrently INTEGER,
            onVentilatorCumulative INTEGER);"""
curs.execute(sql)

#Load the daily tracker data
tDailyTracker = pd.read_csv("./data/CovidDailyTracking_Filtered_Nov16.csv")
sql = "INSERT INTO tDailyTracker VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);"
for row in tDailyTracker.values:
    curs.execute(sql, tuple(row))
```

[80]:
```
curs.execute("DROP TABLE IF EXISTS tStateEmergency;")
#Creat the stay at home table (columns are when stay at home order was␣
↪announced and when it was implemented: has 42 observations)
sql="""CREATE TABLE tStateEmergency (
            State TEXT REFERENCES tState(state),
            StateEmergency DATE,
            FirstSchoolClosure DATE);"""
curs.execute(sql)

#Load the stay at home data
tStateEmergency = pd.read_csv("./data/Ballotpedia_StateEmergency.csv")
sql = "INSERT INTO tStateEmergency VALUES (?,?,?);"
for row in tStateEmergency.values:
    curs.execute(sql, tuple(row))
```

[81]:
```
pd.read_sql("SELECT * FROM sqlite_master;", conn)
```

[81]:
```
    type                       name         tbl_name  rootpage  \
0  table                     tState           tState         2
1  index  sqlite_autoindex_tState_1           tState         3
2  index  sqlite_autoindex_tState_2           tState         4
3  table                 tStayAtHome      tStayAtHome         5
4  table                  tStateInfo       tStateInfo         6
5  table               tDailyTracker    tDailyTracker         7
6  table             tStateEmergency  tStateEmergency       198

                                                  sql
0  CREATE TABLE tState (\n              state TEXT …
1                                                None
2                                                None
3  CREATE TABLE tStayAtHome (\n             State …
4  CREATE TABLE tStateInfo (\n            State T…
5  CREATE TABLE tDailyTracker (\n              st T…
6  CREATE TABLE tStateEmergency (\n            St…
```

---

1) Which state(s) were the first to issue a state of emergency, and how many positive test cases had been reported in those state(s) at that time?

Return three columns: - State - Number of positive tests - Date of emergency declaration

```
[86]: #Need to create a view that points to each date that a state issued a state of␣
       ↪emergency, and then see the number of positive cases for those dates
      #This view is going to need a left join because not all states issued a state␣
       ↪of emergency
      curs.execute("""DROP VIEW IF EXISTS vFirstState;""")
      curs.execute("""CREATE VIEW vFirstState AS
                      SELECT * FROM tStateEmergency
                      ORDER BY StateEmergency ASC
                      LIMIT 1;""")
```

```
[86]: <sqlite3.Cursor at 0x7f9236e1b810>
```

```
[87]: #Lets Create another view where we find the info for washington on 2020-02-29␣
       ↪from tDailyTracker
      curs.execute("""DROP VIEW IF EXISTS vWashData;""")
      curs.execute("""CREATE VIEW vWashData AS
                      SELECT * FROM tDailyTracker
                      WHERE st = 'WA'
                      ;""")
```

```
[87]: <sqlite3.Cursor at 0x7f9236e1b810>
```

```
[89]: #Need to get the tDailyTracker table working so that i an get the number of␣
       ↪positives, everything else should be correct
      #Number one
      pd.read_sql("""SELECT state,StateEmergency, positive
                      FROM vFirstState
                      JOIN tState USING (state)
                      JOIN tDailyTracker USING (st)
                      WHERE date_clean = '2020-02-29'
                     ;""",conn)
```

```
[89]:         State StateEmergency  positive
      0  Washington     2020-02-29        18
```

---

2) Of states which did declare a state of emergency, which were the last, how many DEATHS had been reported in those state(s) at that time, and, if they did issue a statewide stay at home order, when?

Return 4 columns:

- State
- Number of deaths at the time state of emergency was declared
- Date the state of emergency was declared
- Date stay-at-home order issued (if it exists)

```
[90]: #This will require a view where we order the tStateEmergency table from latest␣
       ↪date to eatliest, and then pick the ones from that last date
      curs.execute("""DROP VIEW IF EXISTS vLastState;""")
      curs.execute("""CREATE VIEW vLastState AS
                      SELECT * FROM tStateEmergency
                      ORDER BY StateEmergency DESC
                      LIMIT 1;""")
      #West Virginia was the last state to declare a state of emergency, one day␣
       ↪after maine and oklahoma
```

```
[90]: <sqlite3.Cursor at 0x7f9236e1b810>
```

```
[91]: #GOING TO need to create another view that points to the day that they declared␣
       ↪the state of emergency to get the deaths number
      curs.execute("""DROP VIEW IF EXISTS vDeathCount;""")
      curs.execute("""CREATE VIEW vDeathCount AS
                      SELECT date_clean, st, death
                      FROM tDailyTracker
                      WHERE date_clean = '2020-03-16'
                      AND st = "WV";""")
```

```
[91]: <sqlite3.Cursor at 0x7f9236e1b810>
```

```
[92]: #Using this info want to state, deaths at time, emergency dat and stay at home␣
       ↪date
      #Number 2
      pd.read_sql("""SELECT state, death, StateEmergency, `Date Announced`
                      FROM vLastState
                      JOIN tState USING (state)
                      JOIN vDeathCount USING (st)
                      JOIN tStayAtHome USING (state)
                      ;""",conn)
```

```
[92]:            State  death StateEmergency Date Announced
      0  West Virginia      0     2020-03-16     2020-03-23
```

---

3) According to the data provided, which state(s) did not issue a stay-at-home order, and how many total deaths have been reported in those state(s)?

Return two columns: - State - Number of deaths (as of Nov 15)

```
[65]: #Need to do like a left join of the state data to the csv of csv of stay at␣
       ↪homes and find the ones that are null, then join
      curs.execute("""DROP VIEW IF EXISTS vNoStayAtHome;""")
      curs.execute("""CREATE VIEW vNoStayAtHome AS
                      SELECT *
```

```
                FROM tState
                LEFT JOIN tStayAtHome USING (state)
                WHERE `Date Announced` IS NULL;""")
```

[65]:
```
<sqlite3.Cursor at 0x7f1b5fc8f500>
```

[133]:
```
#Number 3
pd.read_sql("""SELECT state, death
                FROM vNoStayAtHome
                JOIN tDailyTracker USING (st)
                WHERE date_clean = '2020-11-15'


                ;""",conn)
```

[133]:
```
            state  death
0        Arkansas   2183
1            Iowa   1985
2        Nebraska    779
3    North Dakota    570
4        Oklahoma   1528
5  South Carolina   4112
6    South Dakota    644
7            Utah    718
8         Wyoming    144
```

---

4) Repeat the previous question, but this time look at states that did issue a stay-at-home order

Return three columns: - State - Number of deaths (as of Nov 15) - Date stay-at-home order announced

[218]:
```
curs.execute("""DROP VIEW IF EXISTS vYesStayAtHome;""")
curs.execute("""CREATE VIEW vYesStayAtHome AS
                SELECT *
                FROM tState
                LEFT JOIN tStayAtHome USING (state)
                WHERE `Date Announced` IS NOT NULL;""")
```

[218]:
```
<sqlite3.Cursor at 0x7f1b5f14c3b0>
```

[221]:
```
#Number 4

pd.read_sql("""SELECT state, death, `Date Announced`
                FROM tStayAtHome
                JOIN tState USING (state)
                JOIN tDailyTracker USING (st)
                WHERE date_clean = 2020-11-15'


                ;""",conn)
```

[221]:

```
                State  death Date Announced
0             Alabama   3248      2020-04-03
1              Alaska     98      2020-03-27
2             Arizona   6302      2020-03-30
3          California  18253      2020-03-19
4            Colorado   2234      2020-03-26
5         Connecticut   4737      2020-03-20
6            Delaware    736      2020-03-22
7             Florida  17734      2020-04-01
8             Georgia   8957      2020-04-02
9              Hawaii    222      2020-03-23
10              Idaho    759      2020-03-25
11           Illinois  11162      2020-03-20
12            Indiana   4910      2020-03-23
13             Kansas   1256      2020-03-28
14           Kentucky   1661      2020-03-22
15          Louisiana   6132      2020-03-22
16              Maine    165      2020-03-31
17           Maryland   4302      2020-03-30
18      Massachusetts  10329      2020-03-23
19           Michigan   8376      2020-03-23
20          Minnesota   2905      2020-03-25
21        Mississippi   3543      2020-03-31
22           Missouri   3374      2020-04-03
23            Montana    520      2020-03-26
24             Nevada   1909      2020-04-01
25      New Hampshire    499      2020-03-26
26         New Jersey  16566      2020-03-20
27         New Mexico   1208      2020-03-23
28           New York  26133      2020-03-20
29     North Carolina   4806      2020-03-27
30               Ohio   5722      2020-03-22
31             Oregon    761      2020-03-23
32       Pennsylvania   9312      2020-03-23
33       Rhode Island   1254      2020-03-28
34          Tennessee   3893      2020-03-30
35              Texas  19559      2020-03-31
36            Vermont     59      2020-03-24
37           Virginia   3800      2020-03-30
38         Washington   2519      2020-03-23
39      West Virginia    582      2020-03-23
40          Wisconsin   2751      2020-03-24
```

```
41  District of Columbia    660     2020-03-30
```

---

5) Return the following statistics for Virginia:

- Total number of postive cases reported
- Total number of deaths
- Total number of deaths per capita
- Mortality rate, estimated by: Number of deaths / number of positive cases

Hint: Beware of data types, integer conversion etc. The answers are probably not zero.

[263]:
```
#Just have to use the last set of data from the covid tracker, so the 15th and␣
↪then where st = "VA"
#I think this can work using just the tDailyTrakcer and tStateInfo (for␣
↪population data)
#LEts create a view that just points to virginia on the 15th of november, the␣
↪last day with inforation
curs.execute("""DROP VIEW IF EXISTS vVADeaths;""")
curs.execute("""CREATE VIEW vVADeaths AS
                SELECT *
                FROM tDailyTracker
                WHERE st="VA"
                AND date_clean="2020-11-15";""")
```

[263]:
```
<sqlite3.Cursor at 0x7f1b5f14c3b0>
```

[265]:
```
#NExt We are going to create a view that points at the virginia data in␣
↪tStateInfo, so that we can get the returns tat require caluclations
curs.execute("""DROP VIEW IF EXISTS vVAInfo;""")
curs.execute("""CREATE VIEW vVAInfo AS SELECT * FROM tStateInfo WHERE␣
↪state="Virginia";""")
```

[265]:
```
<sqlite3.Cursor at 0x7f1b5f14c3b0>
```

[268]:
```
pd.read_sql("""SELECT positive, death, (1.0*death/pop2019est) AS␣
↪DeathPerCapita, (1.0*death/positive) AS MortalityRate
                FROM vVADeaths
                JOIN tState USING (st)
                JOIN vVAInfo USING (State)
                ;""",conn)
```

[268]:
```
   positive  death  DeathPerCapita  MortalityRate
0    201960   3800        0.000445       0.018816
```

---

6) Which state has had the most deaths per capita as of Nov 15?

Return:

- State
- Number of deaths
- Population
- Population per square mile
- Number of deaths per capita
- Mortality rate, estimated as

Hint: I made a view first, which shortened up the SQL here quite a bit

[24]:
```
#Create a view , this info can all come from tStateInfo and tDailyTracker

curs.execute("""DROP VIEW IF EXISTS vMostDeaths;""")
curs.execute("""CREATE VIEW vMostDeaths AS
                SELECT *
                FROM tStateInfo
                JOIN tState USING (state)
                JOIN tDailyTracker USING (st)
                WHERE date_clean = "2020-11-15"
                ;""")
```

[24]:
```
<sqlite3.Cursor at 0x7f926a59cdc0>
```

[25]:
```
#death per capita, mortality rate as deaths/positive cases
pd.read_sql("""SELECT State, death, pop2019est, PopPerSqMI, (1.0*death/
 ↪pop2019est) AS DeathsPerCapita, (1.0*death/positive) AS MortalityRate
               FROM vMostDeaths
               ORDER BY DeathsPerCapita DESC
                LIMIT 1



                ;""",conn)
```

[25]:
```
        State  death  pop2019est   PopPerSqMi  DeathsPerCapita  MortalityRate
0  New Jersey  16566     8882190  1207.767785         0.001865       0.059318
```

---

7) Repeat the previous question, but this time for the state with the fewest deaths per capita as of Nov 15

[27]:
```
pd.read_sql("""SELECT State, death, pop2019est, PopPerSqMI, (1.0*death/
 ↪pop2019est) AS DeathsPerCapita, (1.0*death/positive) AS MortalityRate
               FROM vMostDeaths
               ORDER BY DeathsPerCapita ASC
                LIMIT 1;""",conn)
```

```
[27]:      State  death  pop2019est  PopPerSqMi  DeathsPerCapita  MortalityRate
      0  Vermont     59      623989   67.702291         0.000095       0.020422
```

---

8) For the entire US (i.e. the sum of all 50 states + Washington DC):

Get the daily number (not the running total) of positive cases, deaths, and tests reported

Return:

- Date
- The number of new positive tests reported per day
- The number of new deaths reported per day
- The number of new tests performed per day

Order the results by date, ascending

```
[82]: pd.read_sql("""SELECT date_clean, SUM(positiveIncrease), SUM(deathIncrease),␣
       ↪SUM(totalTestResultsIncrease)
                      FROM tDailyTracker
                      GROUP BY date_clean;""",conn)


      #May need to group by date clean
```

```
[82]:      date_clean  SUM(positiveIncrease)  SUM(deathIncrease)  \
      0    2020-01-22                      0                   0
      1    2020-01-23                      0                   0
      2    2020-01-24                      0                   0
      3    2020-01-25                      0                   0
      4    2020-01-26                      0                   0
      ..          …                      …                   …
      294  2020-11-11                 144134                1553
      295  2020-11-12                 149099                1096
      296  2020-11-13                 170051                1297
      297  2020-11-14                 162755                1314
      298  2020-11-15                 144807                 657

           SUM(totalTestResultsIncrease)
      0                                0
      1                                1
      2                                0
      3                                0
      4                                0
      ..                             …
      294                        1380904
      295                        1488194
      296                        1682170
      297                        1654691
```

```
298                           1473789

[299 rows x 4 columns]
```

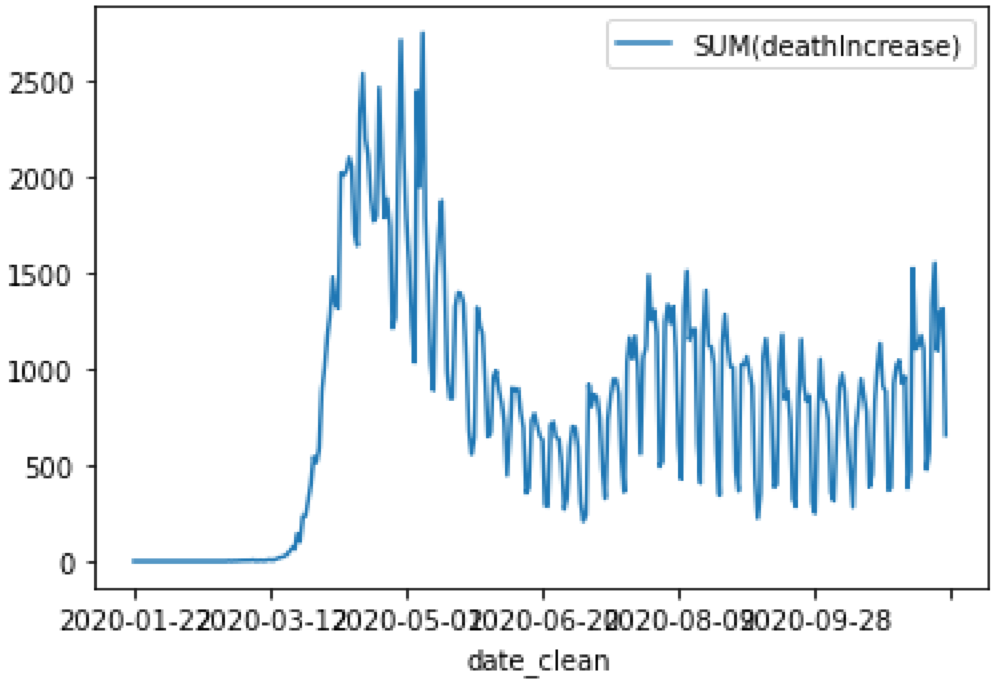
BONUS: +2 points. The previous results aren't readily intepretable as a long list of numbers.

Make a plot with date on the x-axis, and daily # of deaths on the y-axis.

```
[83]: bonus_df = pd.read_sql("""SELECT date_clean, SUM(positiveIncrease),␣
       ↪SUM(deathIncrease), SUM(totalTestResultsIncrease)
                       FROM tDailyTracker
                       GROUP BY date_clean;""",conn)
```

```
[85]: bonus_df.plot(x ='date_clean', y='SUM(deathIncrease)', kind = 'line')
```

```
[85]: <AxesSubplot:xlabel='date_clean'>
```



```
[1]: # Don't forget to close the database!
     conn.close()
```