COVID-19 Data Analysis

In [1]:
```
import sqlite3
import numpy as np
import pandas as pd
```

Data file is in CSV format.

File size is 9.21 GB.

Since this file is too large, below, I count the number of lines and split the file into multiple files.

In [2]:
```
f = open("COVID-19_Case_Surveillance_Public_Use_Data_with_Geography.csv", "r"
```

In [3]:
```
for count, line in enumerate(f):
    pass
```

In [4]:
```
print(count)
```

```
71387132
```

In [5]:
```
chunk_size = 20000000
```

In [6]:
```
def write_chunk(part, lines):
    with open("data_part_" + str(part) + ".csv", "w") as f_out:
        f_out.write(header)
        f_out.writelines(lines)
```

In [7]:
```
with open("COVID-19_Case_Surveillance_Public_Use_Data_with_Geography.csv", "r
    count = 0
    header = f.readline()
    lines = []
    for line in f:
        count += 1
        lines.append(line)
        if count % chunk_size == 0:
            write_chunk(count // chunk_size, lines)
            lines = []
    # write remainder
    if len(lines) > 0:
        write_chunk((count // chunk_size) + 1, lines)
```

In [8]:
```
f.close()
```

Using SQL to store large dataset

- Read in each CSV data-part into a DataFrame
- Then, export each part to an SQL database

In [9]:
```
# Create SQL Engine, Connection, and Cursor
# If .db file does not exist, it will be created during connection
connection = sqlite3.connect('covid_large_dataset.db')
cursor = connection.cursor()
```

In [10]:
```
# Create table in database, if it does not exist
command1 = """CREATE TABLE IF NOT EXISTS covid_data(id INTEGER PRIMARY KEY, ca
            state_fips_code TEXT, res_county TEXT, county_fips_code TEXT, age_
            ethnicity TEXT, case_positive_specimen_interval INTEGER, case_onse
            exposure_yn TEXT, current_status TEXT, symptom_status TEXT, hosp_y
            underlying_conditions_yn TEXT)"""
cursor.execute(command1)
connection.commit()
```

In [11]:
```
# Read in first CSV part and export to SQL table in database
df = pd.read_csv("data_part_1.csv", low_memory=False)
df.to_sql('covid_data', connection, if_exists='append', index_label='id')
connection.commit()
```

In [12]:
```
# Read in second CSV part and export to SQL table in database
df = pd.read_csv("data_part_2.csv", low_memory=False)

# The first CSV part had an index (id) of range 0 to 19,999,999
# Therefore, we must reindex this part to the next range, 20,000,000 to 39,99
# Otherwise, we would get a unique index error when we try to export to SQL ta
df.index = range(20000000, 40000000)

# Export data to SQL table in database
df.to_sql('covid_data', connection, if_exists='append', index_label='id')
connection.commit()
```

In [13]:
```
# Read in third CSV part, reindex to range starting with 40,000,000,
# and export data to SQL table in database
df = pd.read_csv("data_part_3.csv", low_memory=False)
df.index = range(40000000, 60000000)
df.to_sql('covid_data', connection, if_exists='append', index_label='id')
connection.commit()
```

In [14]:
```
# Read in fourth CSV part. Get info to check number of lines.
df = pd.read_csv("data_part_4.csv", low_memory=False)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11387132 entries, 0 to 11387131
Data columns (total 19 columns):
 #   Column                           Dtype
---  ------                           -----
```

```
  0   case_month                       object
  1   res_state                        object
  2   state_fips_code                  int64
  3   res_county                       object
  4   county_fips_code                 float64
  5   age_group                        object
  6   sex                              object
  7   race                             object
  8   ethnicity                        object
  9   case_positive_specimen_interval  float64
  10  case_onset_interval              float64
  11  process                          object
  12  exposure_yn                      object
  13  current_status                   object
  14  symptom_status                   object
  15  hosp_yn                          object
  16  icu_yn                           object
  17  death_yn                         object
  18  underlying_conditions_yn         object
 dtypes: float64(3), int64(1), object(15)
```

In [15]:

```
# Then, reindex to range starting with 60,000,000,
# and export data to SQL table in database
df.index = range(60000000, (60000000+11387132))
df.to_sql('covid_data', connection, if_exists='append', index_label='id')
connection.commit()
```

In [16]:

```
# Close connection to database
connection.close()
```

Open SQL database and read in data to DataFrame for data analysis

In [17]:

```
# Create SQL Engine, Connection, and Cursor
connection = sqlite3.connect('covid_large_dataset.db')
cursor = connection.cursor()
```

In [18]:

```
# SQL command to read data from table in database
command1 = """SELECT id,
                     case_month,
                     res_state,
                     age_group,
                     sex,
                     race,
                     ethnicity,
                     case_positive_specimen_interval,
                     case_onset_interval,
                     death_yn
                FROM covid_data
               WHERE death_yn = 'Yes';"""

# Execute command and read into DataFrame
df = pd.read_sql(sql=command1, con=connection, index_col="id")
```

In [19]:

```
# Get DataFrame info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 416870 entries, 678 to 71385121
Data columns (total 9 columns):
 #   Column                           Non-Null Count   Dtype
---  ------                           --------------   -----
 0   case_month                       416870 non-null  object
 1   res_state                        416870 non-null  object
 2   age_group                        415933 non-null  object
 3   sex                              415205 non-null  object
 4   race                             396902 non-null  object
 5   ethnicity                        396191 non-null  object
 6   case_positive_specimen_interval  153374 non-null  float64
 7   case_onset_interval              160721 non-null  float64
 8   death_yn                         416870 non-null  object
dtypes: float64(2), object(7)
memory usage: 31.8+ MB
```

In [20]:

```
# Find how many covid deaths per state
deaths_by_state = df.groupby(["res_state"]).size()
print(deaths_by_state)
```

```
res_state
AK       81
AL     5100
AR     4105
AZ    21502
CA    62033
CO     5022
CT     4673
DC      676
FL    43165
GA      836
IA      907
ID     1894
IL    21804
```

```
IN     5995
KS     3804
KY     4723
LA     2472
MA    14369
MD     2821
ME      409
MI    12943
MN     6452
MO     7610
MS     1145
MT     1259
NC     5012
ND      989
NH      779
NJ    19647
NM     2123
NV     8938
NY    40393
OH    21943
OK     3825
OR     1324
PA    22002
PR     3493
RI      594
SC     4999
TN     9049
TX    19467
UT     1292
VA     5220
VT       32
WA     4460
WI     5143
WY      346
```

In [21]:

```
# Export DataFrame to csv file for use later.
df.to_csv("covid-data-with-deaths.csv")
```

In [ ]: