signment1-jaamie-maarsh-joy-martin

October 5, 2023

Assignment-1: Foundations of Data Analytics-IE6400

Q1:

[573]:
```
#Q1;

import numpy as np

def computing_inter_event_times(time_series_list, threshold_value):
    """
    Calculating the time intervals which is based on a time series values and a␣
  ↪threshold value.

    Input Parameters used:
    - time_series_list (list): An array representing time series data.
    - threshold_value : The minimum event size for considering an event.

    Output Return parameter:
    - inter_event_times (list): A list which returns out the inter-event times.

    Process:
    - From the input time series list, the index and the element value is being␣
  ↪taken using the enumerate function.
    with the help of the conditional parameter (if) the element from the␣
  ↪incoming list is compared against the threshold
    limit. For a positive outcome, the index value of the previous event is␣
  ↪checked and then, if successful, the difference
    is found between the index of the variable in the iteration and the index␣
  ↪of the previous event value.
    """

    # Initialization of variables
    inter_event_times = []
    last_event_time = None

    # Iterate through the time series
    for index, time_point in enumerate(time_series_list):
        if time_point >= threshold_value:
            if last_event_time is not None:
                inter_event_times.append(index - last_event_time)
                last_event_time = index
            else:
                last_event_time = index

        else:
            continue

    return inter_event_times

# Example under study:
time_series_list = [10,1,6,4,3,2,7,0,8,8,2,0,7,7,1,0]
threshold_value = 5
inter_event_times_example = computing_inter_event_times(time_series_list,␣
 ↪threshold_value)
print("The array of inter-event time between successive events is  :" ,␣
 ↪inter_event_times_example)
```

```
The array of inter-event time between successive events is  : [2, 4, 2, 1, 3, 1]
```

[534]:

```
# importing all the necessary libraries into the workspace
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

#This command is to ignore all the warnings
warnings.filterwarnings("ignore")

# loading/reading of the datasets into a dataframe (df)
df_border= pd.read_csv('/Users/jaamiemaarshj/Desktop/  DAE Course Materials/
 ↪Fundamentals of Data Analytics/Assignement-1/Border_Crossing_Entry_Data.
 ↪csv', low_memory=False)
df_data = pd.read_csv('/Users/jaamiemaarshj/Desktop/  DAE Course Materials/
 ↪Fundamentals of Data Analytics/Assignement-1/data-table.csv' ,␣
 ↪low_memory=False)

#displaying the contents of the dataframe
display(df_border)
```

```
            Port Name         State  Port Code            Border      Date  \
0             Detroit      Michigan       3801  US-Canada Border  Aug 2023
1               Alcan        Alaska       3104  US-Canada Border  Jul 2023
2              Calais         Maine        115  US-Canada Border  Jul 2023
```

```
3               Noonan  North Dakota       3420  US-Canada Border  Jul 2023
4              Warroad     Minnesota       3423  US-Canada Border  May 2023
…                 …           …        …              …       …
385766        Richford       Vermont        203  US-Canada Border  Feb 1996
385767       Vanceboro         Maine        105  US-Canada Border  Jul 1997
385768  Fort Fairfield         Maine        107  US-Canada Border  May 1996
385769         Pembina  North Dakota       3401  US-Canada Border  Jan 1996
385770        Danville    Washington       3012  US-Canada Border  Apr 1997

                       Measure  Value  Latitude  Longitude  \
0                       Trains    128    42.332    -83.048
1               Bus Passengers    696    62.615   -141.001
2                        Buses     16    45.189    -67.275
3                       Trucks    142    48.999   -103.004
4                        Buses     41    48.999    -95.377
…                        …    …       …        …
385766                   Buses      0    45.012    -72.589
385767  Truck Containers Empty    263    45.569    -67.429
385768  Rail Containers Loaded      0    46.765    -67.789
385769  Truck Containers Empty   1663    49.000    -97.237
385770  Truck Containers Empty      0    49.000   -118.504

                                Point
0         POINT (-83.047924 42.331685)
1        POINT (-141.001444 62.614961)
2         POINT (-67.275381 45.188548)
3        POINT (-103.004361 48.999333)
4            POINT (-95.376555 48.999)
…                               …
385766    POINT (-72.588559 45.01174)
385767   POINT (-67.428541 45.568761)
385768   POINT (-67.789471 46.765323)
385769   POINT (-97.237036 49.000453)
385770  POINT (-118.503722 49.000083)

[385771 rows x 10 columns]
```

Question 2:

Task 1:

[575]:

```
#Finding out the number of unique ports statewise
Unique_Ports = df_border.groupby('State')['Port Name'].nunique()
print("The Answer for Question 2 - Task 1 is found below: ")
print(" ------------------------------------------------")
print("Total count of the unique ports - State wise:" )
print(" ------------------------------------------------")
print(Unique_Ports)
print(" ------------------------------------------------")
```

```
#Figuring out the top 5 states based on the above count:
Sort_Unique_Ports = Unique_Ports.sort_values(ascending=False).head(5)
print(" Top 5 states with Maximum number of unique ports")
print(" ------------------------------------------------")
print(Sort_Unique_Ports)
```

```
The Answer for Question 2 - Task 1 is found below:
 ------------------------------------------------
Total count of the unique ports - State wise:
 ------------------------------------------------
State
Alaska           4
Arizona          6
California       7
Idaho            2
Maine           13
Michigan         4
Minnesota        8
Montana         13
New Mexico       2
New York         7
North Dakota    18
Texas           13
Vermont          5
Washington      15
Name: Port Name, dtype: int64
 ------------------------------------------------
 Top 5 states with Maximum number of unique ports
 ------------------------------------------------
State
North Dakota    18
Washington      15
Maine           13
Montana         13
Texas           13
Name: Port Name, dtype: int64
```

Task 2:

[576]:

```
#Converting the date field using the required panda utility
df_border['Date'] = pd.to_datetime(df_border['Date'])

#getting the trucks crossing data
df_border_trucks = df_border[(df_border['Measure'] == 'Trucks')]

#setting up the start and end date to extract data for years 2019 to 2022
starting_date = pd.Timestamp('2019-01-01') #setting the start date range
```

```
ending_date = pd.Timestamp('2022-12-31') #setting the end date range

#Filtering the records to display the 4 year data
df_trucks_for_4Years = df_border_trucks[(df_border_trucks['Date'] >=␣
 ↪starting_date)&(df_border_trucks['Date'] <= ending_date)]

#splitting the entire date field into 2 seperate columns as year and months for␣
 ↪easy calculation
df_trucks_for_4Years['Years'] = df_trucks_for_4Years['Date'].dt.year
df_trucks_for_4Years['Months'] = df_trucks_for_4Years['Date'].dt.month

#grouping the dataset with respect to Years and Border column
df_trucks_for_4Years = df_trucks_for_4Years.groupby(['Years' ,␣
 ↪'Border'])['Value'].agg(['sum']).reset_index()

#Calculating the value of Average Monthly trucks value from the total yearly␣
 ↪trucks
df_trucks_for_4Years['AverageMonthlyTrucks'] = df_trucks_for_4Years['sum']/12

#Renaming the "sum" column name to Total yearly trucks
df_trucks_for_4Years.rename(columns = {'sum': 'TotalYearlyTrucks'} ,inplace␣
 ↪=True)

#Picking out the necessary rows for the dataframe.
df_trucks_for_4Years = df_trucks_for_4Years[['Years' , 'Border' ,␣
 ↪'AverageMonthlyTrucks' , 'TotalYearlyTrucks']]

print("The Answer for Question 2 - Task 2 is found below: ")
print(" ------------------------------------------------")

display(df_trucks_for_4Years)
```

```
The Answer for Question 2 - Task 2 is found below:
 ------------------------------------------------

   Years            Border  AverageMonthlyTrucks  TotalYearlyTrucks
0   2019  US-Canada Border         473429.583333            5681155
1   2019  US-Mexico Border         536687.916667            6440255
2   2020  US-Canada Border         434527.750000            5214333
3   2020  US-Mexico Border         530532.000000            6366384
4   2021  US-Canada Border         464148.833333            5569786
5   2021  US-Mexico Border         579108.333333            6949300
6   2022  US-Canada Border         457769.000000            5493228
7   2022  US-Mexico Border         604866.666667            7258400
```

#Task 3:

[577]:
```
#Constructing a pivot table to display all the measure values as individual␣
 ↪columns

pivoted_df_border = df_border.pivot_table(index=['Port Name', 'State' ,␣
 ↪'Border', 'Date' ],
                            columns='Measure', values='Value', aggfunc='sum' ,␣
 ↪fill_value=0).reset_index()

print("The Answer for Question 2 - Task 3 is found below: ")
print(" ------------------------------------------------")

display(pivoted_df_border)
```

```
The Answer for Question 2 - Task 3 is found below:
 ------------------------------------------------

Measure Port Name   State            Border       Date  Bus Passengers  Buses  \
0           Alcan  Alaska  US-Canada Border 1996-01-01               9      3
1           Alcan  Alaska  US-Canada Border 1996-02-01               0      0
2           Alcan  Alaska  US-Canada Border 1996-03-01              11      3
3           Alcan  Alaska  US-Canada Border 1996-04-01              17      6
4           Alcan  Alaska  US-Canada Border 1996-05-01             638     30
…             …     …               …          …             …     …
36001      Ysleta   Texas  US-Mexico Border 2023-04-01               0      0
36002      Ysleta   Texas  US-Mexico Border 2023-05-01               0      0
36003      Ysleta   Texas  US-Mexico Border 2023-06-01               0      0
36004      Ysleta   Texas  US-Mexico Border 2023-07-01               0      0
36005      Ysleta   Texas  US-Mexico Border 2023-08-01               0      0

Measure  Pedestrians  Personal Vehicle Passengers  Personal Vehicles  \
0                  0                         2011                965
1                  0                         1800                976
2                  0                         2347               1962
3                  0                         4584               2445
4                  2                         9896               5381
…                …                           …                 …
36001         124304                       498382             284268
36002         125507                       496352             291251
36003          76505                       513539             297751
36004         107457                       549464             306408
36005         116352                       533473             307896

Measure  Rail Containers Empty  Rail Containers Loaded  Train Passengers  \
0                            0                       0                 0
1                            0                       0                 0
2                            0                       0                 0
3                            0                       0                 0
4                            0                       0                 0
```

```
…                        …                     …               …
36001                        0                       0                 0
36002                        0                       0                 0
36003                        0                       0                 0
36004                        0                       0                 0
36005                        0                       0                 0

Measure  Trains  Truck Containers Empty  Truck Containers Loaded  Trucks
0             0                       0                        0     428
1             0                       0                        0     385
2             0                       0                        0     484
3             0                       0                        0     564
4             0                       0                        0     608
…         …                     …                      …     …
36001         0                   16653                    35258   49856
36002         0                   20042                    39610   59109
36003         0                   19846                    38607   58288
36004         0                   19049                    67221   55090
36005         0                   21196                    75671   62030

[36006 rows x 16 columns]
```

1 Task 4:

[578]:
```
#creating an added column named "TotalVehicles" on to the dataframe.

#calculating the "Total Vehicles" field value based on the other "Measure" fields
pivoted_df_border['TotalVehicles'] =  pivoted_df_border['Buses'] + pivoted_df_border['Personal Vehicles'] + pivoted_df_border['Trains'] + pivoted_df_border['Trucks']

#creating a new dataframe with the necessary columns
new_pivot_df = pivoted_df_border[['State', 'Date', 'TotalVehicles']]
sort_state_wise_df = new_pivot_df.sort_values(['State'] , ascending=True)
display(sort_state_wise_df)
```

```
Measure       State        Date  TotalVehicles
0            Alaska  1996-01-01           1396
6721         Alaska  2007-02-01            511
6720         Alaska  2007-01-01            616
6719         Alaska  2006-12-01            477
6718         Alaska  2006-11-01            598
…             …        …            …
31607    Washington  2021-09-01          20164
31606    Washington  2021-08-01          20529
31605    Washington  2021-07-01          19674
```

```
31613    Washington 2022-03-01          24305
18413    Washington 2023-07-01           3344

[36006 rows x 3 columns]
```

[579]:

```python
#Sorted the dataframe by the total number of vehicles (by largest count)
sort_pivot = new_pivot_df.sort_values(['TotalVehicles'], ascending=False)

print("The Answer for Question 2 - Task 4 is found below: ")
print(" ------------------------------------------------")

display(sort_pivot)
```

```
The Answer for Question 2 - Task 4 is found below:
 ------------------------------------------------

Measure        State       Date  TotalVehicles
28441     California 2006-11-01        1756224
10223          Texas 2001-08-01        1752723
10220          Texas 2001-05-01        1597172
10218          Texas 2001-03-01        1593398
28409     California 2004-03-01        1574704
…                …          …            …
21213      Minnesota 2010-11-01              0
21212      Minnesota 2010-10-01              0
21211      Minnesota 2010-09-01              0
21210      Minnesota 2010-08-01              0
25268          Maine 2010-05-01              0

[36006 rows x 3 columns]
```

2 Task 5:

[582]:

```python
#changing the "date" field from int to the required timestamp format
df_border['Date'] = pd.to_datetime(df_border['Date'])

#pivoting the table to split the values of "Measure" from the base dataframe
pivoted_df_presidential = df_border.pivot_table(index=['Port Name', 'State' ,␣
 ↪'Border', 'Date' ],
                            columns='Measure', values='Value', aggfunc='sum' ,␣
 ↪fill_value=0).reset_index()

#splitting up the date in terms of years
pivoted_df_presidential['Years'] = pivoted_df_presidential['Date'].dt.year

#converting the date to be an integer since it converts into float when␣
 ↪splitting.
```

```
pivoted_df_presidential['Years'] = pivoted_df_presidential['Years'].astype(int)

#adding a new column for determining the summation of passengers
pivoted_df_presidential ['Individual Crossings'] =␣
 ↪pivoted_df_presidential['Pedestrians'] + pivoted_df_presidential ['Bus␣
 ↪Passengers'] + pivoted_df_presidential['Train Passengers']

display(pivoted_df_presidential)
```

```
Measure Port Name   State            Border       Date  Bus Passengers  Buses  \
0            Alcan  Alaska  US-Canada Border 1996-01-01               9      3
1            Alcan  Alaska  US-Canada Border 1996-02-01               0      0
2            Alcan  Alaska  US-Canada Border 1996-03-01              11      3
3            Alcan  Alaska  US-Canada Border 1996-04-01              17      6
4            Alcan  Alaska  US-Canada Border 1996-05-01             638     30
…           …     …               …        …             …    …
36001      Ysleta   Texas  US-Mexico Border 2023-04-01               0      0
36002      Ysleta   Texas  US-Mexico Border 2023-05-01               0      0
36003      Ysleta   Texas  US-Mexico Border 2023-06-01               0      0
36004      Ysleta   Texas  US-Mexico Border 2023-07-01               0      0
36005      Ysleta   Texas  US-Mexico Border 2023-08-01               0      0

Measure  Pedestrians  Personal Vehicle Passengers  Personal Vehicles  \
0                  0                         2011                965
1                  0                         1800                976
2                  0                         2347               1962
3                  0                         4584               2445
4                  2                         9896               5381
…              …                          …                …
36001         124304                       498382             284268
36002         125507                       496352             291251
36003          76505                       513539             297751
36004         107457                       549464             306408
36005         116352                       533473             307896

Measure  Rail Containers Empty  Rail Containers Loaded  Train Passengers  \
0                            0                       0                 0
1                            0                       0                 0
2                            0                       0                 0
3                            0                       0                 0
4                            0                       0                 0
…                        …                     …               …
36001                        0                       0                 0
36002                        0                       0                 0
36003                        0                       0                 0
36004                        0                       0                 0
36005                        0                       0                 0
```

```
Measure  Trains  Truck Containers Empty  Truck Containers Loaded  Trucks  \
0             0                       0                        0     428
1             0                       0                        0     385
2             0                       0                        0     484
3             0                       0                        0     564
4             0                       0                        0     608
…         …                     …                      …     …
36001         0                   16653                    35258   49856
36002         0                   20042                    39610   59109
36003         0                   19846                    38607   58288
36004         0                   19049                    67221   55090
36005         0                   21196                    75671   62030

Measure  Years  Individual Crossings
0         1996                     9
1         1996                     0
2         1996                    11
3         1996                    17
4         1996                   640
…        …                   …
36001     2023                124304
36002     2023                125507
36003     2023                 76505
36004     2023                107457
36005     2023                116352

[36006 rows x 18 columns]
```

```
[599]:
#defining a function to find out the presidential party using years:
def get_presidential_party_in_power(year_in_power):
#     """
#    Determining the presidential party for the required years.

#    Input Parameters used:
#    - year_in_power : Input year data.

#    Output Return parameter:
#    - Returns strings values such as 'Republican'/'Democratic'/'Unknown'

#    Process:
#    - Based on the input year given, it iterates through the years in the␣
↪condition and gives out the
#    -appropriate response.
#    """
    if 1989 <= year_in_power <= 1992 or 2001 <= year_in_power <= 2008 or 2016␣
↪<= year_in_power <= 2020:
        return 'Republican'
    elif 1993 <= year_in_power <= 2000 or 2009 <= year_in_power <= 2016 or 2021␣
  ↪<= year_in_power <= 2023:
        return 'Democratic'
    else:
        return 'Unknown'

#returning the value from the above function to create a column
pivoted_df_presidential['Party in Power'] = pivoted_df_presidential['Years'].
  ↪apply(get_presidential_party_in_power)

#after the column is created in the above step, it is then sorted by years.
pivoted_df_presidential_sorted = pivoted_df_presidential.sort_values(by='Years')

#the pivoted table choses to have the necessary columns which are required for␣
  ↪evaluation
pivoted_df_presidential_sorted_required =␣
  ↪pivoted_df_presidential_sorted[['Years', 'Party in Power' , 'Individual␣
  ↪Crossings' ]]

print("The Answer for Question 2 - Task 5 is found below: ")
print(" ------------------------------------------------")

display(pivoted_df_presidential_sorted_required)
```

```
The Answer for Question 2 - Task 5 is found below:
 ------------------------------------------------

Measure  Years Party in Power  Individual Crossings
0         1996     Democratic                     9
9576      1996     Democratic                   929
9577      1996     Democratic                   785
9578      1996     Democratic                  2028
9579      1996     Democratic                  1285
…          …             …                    …
14326     2023     Democratic                   549
14325     2023     Democratic                   363
14324     2023     Democratic                     0
14322     2023     Democratic                   534
36005     2023     Democratic                116352

[36006 rows x 3 columns]
```

3 Task 6:

[564]:

```
#renaming the columns of the homicide dataset
df_data = pd.read_csv('/Users/jaamiemaarshj/Desktop/  DAE Course Materials/
 ↪Fundamentals of Data Analytics/Assignement-1/data-table.csv' ,␣
 ↪low_memory=False)
df_homicide = df_data.rename(columns = {"YEAR": "Years"})
display(df_homicide.head())
```

```
   Years STATE  RATE DEATHS                                        URL
0   2021    AL  15.9    748     /nchs/pressroom/states/alabama/al.htm
1   2021    AK   6.4     49      /nchs/pressroom/states/alaska/ak.htm
2   2021    AZ   8.1    562     /nchs/pressroom/states/arizona/az.htm
3   2021    AR  11.7    335    /nchs/pressroom/states/arkansas/ar.htm
4   2021    CA   6.4   2495  /nchs/pressroom/states/california/ca.htm
```

[565]:

```
#In the below step, the abbrivated state values in the homicide_df are getting␣
 ↪replaced by their full names which is
#obtained from the abbrivations df above
df_homicide=df_homicide.replace(
{'AL': 'Alabama',
    'AK': 'Alaska',
    'AZ': 'Arizona',
    'AR': 'Arkansas',
    'CA': 'California',
    'CO': 'Colorado',
    'CT': 'Connecticut',
    'DE': 'Delaware',
    'FL': 'Florida',
    'GA': 'Georgia',
    'HI': 'Hawaii',
    'ID': 'Idaho',
    'IL': 'Illinois',
    'IN': 'Indiana',
    'IA': 'Iowa',
    'KS': 'Kansas',
    'KY': 'Kentucky',
    'LA': 'Louisiana',
    'ME': 'Maine',
    'MD': 'Maryland',
    'MA': 'Massachusetts',
    'MI': 'Michigan',
    'MN': 'Minnesota',
    'MS': 'Mississippi',
    'MO': 'Missouri',
    'MT': 'Montana',
    'NE': 'Nebraska',
    'NV': 'Nevada',
    'NH': 'New Hampshire',
    'NJ': 'New Jersey',
    'NM': 'New Mexico',
    'NY': 'New York',
    'NC': 'North Carolina',
    'ND': 'North Dakota',
    'OH': 'Ohio',
    'OK': 'Oklahoma',
    'OR': 'Oregon',
    'PA': 'Pennsylvania',
    'RI': 'Rhode Island',
    'SC': 'South Carolina',
    'SD': 'South Dakota',
    'TN': 'Tennessee',
    'TX': 'Texas',
    'UT': 'Utah',
    'VT': 'Vermont',
    'VA': 'Virginia',
    'WA': 'Washington',
    'WV': 'West Virginia',
    'WI': 'Wisconsin',
    'WY': 'Wyoming'
}
)
```

[505]:

```
#displaying the homicide data with the added full forms
display(df_homicide.head())
```

```
   Years       STATE  RATE DEATHS                                      URL
0   2021     Alabama  15.9    748     /nchs/pressroom/states/alabama/al.htm
1   2021      Alaska   6.4     49      /nchs/pressroom/states/alaska/ak.htm
2   2021     Arizona   8.1    562     /nchs/pressroom/states/arizona/az.htm
3   2021    Arkansas  11.7    335    /nchs/pressroom/states/arkansas/ar.htm
4   2021  California   6.4   2495  /nchs/pressroom/states/california/ca.htm
```

[566]:

```
#sorting out data based on years in the homicide dataset
df_homicide = df_homicide[(df_homicide['Years'] >= 2014) &␣
 ↪(df_homicide['Years'] <= 2021)]

#In the step, the grouping of the homicide dataset by state and yearwise
df_homicide=df_homicide.groupby(['STATE','Years'])['DEATHS'].sum().reset_index()

#Renaming columns appropriately
df_homicide=df_homicide.rename(columns={'STATE':'State','DEATHS':'Deaths'})

display(df_homicide)
```

```
       State  Years Deaths
0    Alabama   2014    374
1    Alabama   2015    473
2    Alabama   2016    544
3    Alabama   2017    602
4    Alabama   2018    568
..       …      …    …
395  Wyoming   2017     19
396  Wyoming   2018     22
397  Wyoming   2019     25
398  Wyoming   2020     25
399  Wyoming   2021     16

[400 rows x 3 columns]
```

```
[568]: #Grouping by Date and PresidentialParty and taking the sum of Value (which is␣
        ↪IndividualCrossings)
       pivoted_df_presidential['Date'] = pd.
        ↪to_datetime(pivoted_df_presidential['Date'])
       grouping_crossing =pivoted_df_presidential.
        ↪groupby([pivoted_df_presidential['Date'].dt.year ,'State'])['Individual␣
        ↪Crossings'].sum().reset_index()

       #Filtering out personnel_crossing data from the years 2014 to 2021
       grouping_crossing_df = grouping_crossing[(grouping_crossing['Date'] >=2014) &␣
        ↪(grouping_crossing['Date'] <=2021)]

       #Renaming columns appropriately
       grouping_crossing_df=grouping_crossing_df.rename(columns={'Date':'Years'})
       display(grouping_crossing_df)
```

```
     Years         State  Individual Crossings
252   2014        Alaska                273448
253   2014       Arizona               6499507
254   2014    California              18558316
255   2014         Idaho                 10419
256   2014         Maine                 58627
..     …           …                   …
359   2021      New York                 47328
360   2021  North Dakota                  8743
361   2021         Texas              11856065
362   2021       Vermont                  2131
363   2021    Washington                 20347

[112 rows x 3 columns]
```

```
[601]: #Combining the homicide and the crossing datasets - Inner joining method so␣
        ↪that all the data of the homicide dataset will be present
       inner_merged_homicide = pd.merge(df_homicide, grouping_crossing_df,  how␣
        ↪='outer', on=["Years" , 'State'])
       #display(inner_merged_homicide)
       # Sorting the values in ascending years
       inner_merged_homicide=inner_merged_homicide.sort_values(by=['Years', "State"])

       print("The Answer for Question 2 - Task 6 is found below: ")
       print(" ------------------------------------------------")

       display(inner_merged_homicide)
```

```
The Answer for Question 2 - Task 6 is found below:
 ------------------------------------------------

             State  Years Deaths  Individual Crossings
0          Alabama   2014    374                   NaN
8           Alaska   2014     37              273448.0
16         Arizona   2014    322             6499507.0
24        Arkansas   2014    217                   NaN
32      California   2014  1,813            18558316.0
..             …    …    …                   …
367       Virginia   2021    606                   NaN
375     Washington   2021    346               20347.0
383  West Virginia   2021    114                   NaN
391      Wisconsin   2021    348                   NaN
399        Wyoming   2021     16                   NaN

[400 rows x 4 columns]
```