Assignment 2 - Pandas Data Management

From: Muhammad Khizar Hayat Tahir

UID: 346009027

Submitted to: Prof. Ashok Robin

Course: BANA 680

A. Introduction

Data Management is the practice of acquiring, cleaning, manipulating, storing, and analysing data. With the amount of data available to us, it is impossible to just look at raw sheets and tables and be able to make sense out of it. Luckily for us, there are several tools and packages available that make such management doable. One such package in Python is Pandas. Pandas is a software library widely used for Data Manipulation and Analysis. It offers data structures and methods for dealing with datasets. This assignment is an illustration of how effective Pandas are for managing datasets and acquiring the insights otherwise not visible.

B. Assignment Concept

In this assignment, we are going to use data related to causes of death in the United States to answer some analytical questions. We have been provided with two datasets:

1. The first one has unordered data from year 1999 to year 2016 showing how many deaths were caused by a particular cause of death in a particular state in a particular year.
2. The second one has population data for the individual states, the four census regions, and United States as a whole for the years 2010 till 2018.

We will use these datasets, and Pandas to answer questions about changes in death rate, and identify and analyse the leading causes of death, state-wise, nationally, and yearly.

C. Code and Explanation

Preliminary steps

Importing Pandas package

In [1]:
```
import pandas as pd
import warnings
warnings.simplefilter(action='ignore')
```

Reading in the first dataset (read_csv function) containing leading causes of death in US, and creating a pandas dataframe df1.

To get some sense of this dataset, we execute the commands below to know how much data we have for different variables and also print the top few rows of our dataframe.

In [2]:
```
deaths_file = "C://RIT Courses/BANA 680 - Data Mgmt for BA/jupyter test/NCHS_-_Leading_Causes_of_Death__United_States.csv"
df1 = pd.read_csv(deaths_file)
print('Unique States:', len(df1['State'].unique()))
print('Unique causes:', len(df1['Cause Name'].unique()))
print('Unique years:', len(df1['Year'].unique()))
(df1.head(2))
```

```
Unique States: 52
Unique causes: 11
Unique years: 18
```

Out[2]:

	Year	113 Cause Name	Cause Name	State	Deaths	Age-adjusted Death Rate
0	2012	Nephritis, nephrotic syndrome and nephrosis (N...	Kidney disease	Vermont	21	2.6
1	2016	Nephritis, nephrotic syndrome and nephrosis (N...	Kidney disease	Vermont	30	3.7

Reading in the second dataset (read_excel function) containing population per state and US in general, and creating a dataframe df2. We can print the head to see what kind of data we have. We can see that this dataset requires cleansing and adjustment to make it usable. We will do that below when required.

In [3]:
```
pop_file = "C://RIT Courses/BANA 680 - Data Mgmt for BA/jupyter test/nst-est2018-01.xlsx"
df2 = pd.read_excel(pop_file)
```

Part 1

This part required us to determine if Americans were facing increasing, decreasing, or steady likelyhood of death. The following points outline some important concepts about this part.

- This part was about America as a whole. We observed from the dataset that the data for individual states was already aggregated for United States. Those entries had United States under the "State" column. So we did not need to aggregate from individual states again, rather only filtering for US was required.
- Similarly, the causes were also aggregated under Cause Name -> "All causes". We chose to filter out All causes which will leave us with all individual causes.
- The first dataset had the actual total number of deaths for each category. Aggregating them and comparing is not a viable solution as the population could have varied.
- We needed to standardize the death variable to ensure fair comparison.
- We used that second dataset to extract the population in each year.
- Our goal was to calculate, for each year, the total number of deaths in the United States and divide them by the population of US that year to remove the effect of population size on our comparison.

Part 1.1:

We create a mask to obtain a list of booleans where the rows containing US as State would be true only. Then we subset that mask from df1 to create a dataframe with only US, df1_usa. A quick check to see unique states in df1_usa shows it only has US. We create mask1 to create a boolean list which is true when Cause name is All causes. We subset the negation of this mask from our dataframe to create df1_usa_filtered that only has US and no 'All causes' row. We do some testing at the end to verify our dataframe. A head command is executed to ensure the original structure is maintained.

In [4]:
```
mask = df1['State'].isin(['United States'])
mask1 = df1['Cause Name'].isin(['All causes'])
df1_usa = df1[mask] # this has individual + All causes
df1_usa_filtered = df1_usa[-mask1] # only individual causes
# Tests
print('Is US present?', df1_usa_filtered['State'].isin(['United States']).any())
print('Is "All causes" present?', df1_usa_filtered['Cause Name'].isin(['All cause
s']).any())
print('Total unique states:', len(df1_usa_filtered['State'].unique()))
print('Total unique causes:', len(df1_usa_filtered['Cause Name'].unique()))
df1_usa_filtered.head(2)
```

```
Is US present? True
Is "All causes" present? False
Total unique states: 1
Total unique causes: 10
```

Out[4]:

	Year	113 Cause Name	Cause Name	State	Deaths	Age-adjusted Death Rate
489	2000	Intentional self-harm (suicide) (*U03,X60-X84,...	Suicide	United States	29350	10.4
510	1999	Intentional self-harm (suicide) (*U03,X60-X84,...	Suicide	United States	29199	10.5

Part 1.2:

Here, we group the US dataframe by Year and aggregate the Deaths to obtain a dataframe (usa_deaths) which has a single entry for each year and shows the total number of deaths in that year for US regardless of Cause.

In [5]:
```
usa_deaths = df1_usa_filtered.groupby('Year').agg({'Deaths':sum}).reset_index()
usa_deaths.head(2)
```

Out[5]:

	Year	Deaths
0	1999	1905826
1	2000	1902194

Part 1.3:

We observed from the population dataset that each year has its own column. Our main dataframe has a colum for year and each year is a row. We create a new dataframe (usa_pop) that uses iloc to subset data from the population dataset(df2) to recreate it in long form so it is consistent with our first dataframe.
We are not too concerned about the index names as this is just a middle step.

In [6]:
```
usa_pop = pd.DataFrame({'Year': df2.iloc[2, 3:], 'Pop': df2.iloc[3, 3:]})
usa_pop.reset_index()
usa_pop.head(2)
```

Out[6]:

	Year	Pop
Unnamed: 3	2010	309326085
Unnamed: 4	2011	3.1158e+08

Part 1.4:

Here, we finally merge the two datasets, usa_deaths and usa_pop. We use the merge function of pandas, set the anchor on Year column as that is what we need data arranged on the basis of in both datasets. We know that standardization is very important for fair comparison, and also that one dataset has more years of data than the other. So we use a 'right' join to keep that data for years which we have the population size for. This is a cleansing method. We are left with multiple rows with Nan values which we drop. At the end, we are left with a clean/filtered single data frame (merge1) which has one entry per year, and for that year, it has the total deaths, and US population size for that year. We only have 7 years for which all relavant data is available, but those are the only years we can effectively compare. The other years will only disturb our analysis.

In [7]:
```
merge1 = pd.merge(usa_deaths, usa_pop, on='Year', how='right')
merge1 = merge1.dropna()
merge1.head(2)
```

Out[7]:

	Year	Deaths	Pop
0	2010.0	1852349.0	309326085
1	2011.0	1869321.0	3.1158e+08

Part 1.5:

Here, we calculate the Death per Million (dpm) for each year. We create a new column dpm where we store this standardized rate.

In [8]:
```
merge1['dpm'] = (merge1['Deaths']/merge1['Pop'])*1000000
merge1
```

Out[8]:

	Year	Deaths	Pop	dpm
0	2010.0	1852349.0	309326085	5988.34
1	2011.0	1869321.0	3.1158e+08	5999.49
2	2012.0	1876588.0	3.13874e+08	5978.79
3	2013.0	1910311.0	3.16058e+08	6044.18
4	2014.0	1938408.0	3.18386e+08	6088.22
5	2015.0	2013017.0	3.20743e+08	6276.11
6	2016.0	2034119.0	3.23071e+08	6296.19

Part 1.6 - Results:

From our final result (merge1), we can see that the Deaths per million (dpm) has been **generally** increasing from 2010 till 2016. This provides evidence that the Americans are facing **Increasing** likelyhood of death.

Part 2

This part required us to determine the four leading causes of death for Americans. We need a dataframe that has only the United States as "State" and has all causes **except** for "All causes". Then we would group the data to obtain leading four causes of death.

Part 2.1:

Utilize df1_usa_filtered which has only United States as State and no All causes

Part 2.2:

Here, we group our dataframe by Cause Name and aggregate the total number of deaths (sum function). We sort the values in descending order, and use iloc to obtain the first four entries and store them in usa_causes. We also store these causes as a List in usa_causes_list for later use.

In [9]:
```
usa_causes = df1_usa_filtered.groupby('Cause Name').agg({'Deaths':sum}).round()
usa_causes = usa_causes.sort_values('Deaths', ascending = False).reset_index()
usa_causes = usa_causes.iloc[0:4,:]
# Leading USA causes as list for later use
usa_causes_list = list(usa_causes['Cause Name'])
usa_causes
```

Out[9]:

	Cause Name	Deaths
0	Heart disease	11575183
1	Cancer	10244536
2	Stroke	2580140
3	CLRD	2434726

Part 2.3 - Results:

Our final result 'usa_causes' shows that the leading four causes of death in the US are **Heart disease, Cancer, Stroke, and CLRD** in that order.

Part 3

This part required us to determine if the individual states showed the same four leading causes of death as the US as a whole. For this part, we needed a dataframe that **Did not** have United State as State and All Causes as Cause Name. Our goal was to group data so as to obtain for each State, the top four leading causes of death.

Note: The question does not ask if the leading causes have the same order. So our understanding is that if the leading causes in a state are the same as US (in any order), it will be counted as having the same leading causes.

Part 3.1:

We use our two masks. mask had a boolean list which was true when State was United States. mask1 had a boolean list which was true when Cause name was All Causes. We then subset the negation of these two masks one by one to create a final dataframe (df1_granular) that had granular data of states and cause names, and no aggregated data such as US or All causes. We do some tests to verify our dataframe. The original df1 showed 52 unique states and 11 unique causes, this one has 51 states and 10 causes.

In [10]:
```
df1_no_usa = df1[-mask]
df1_granular = df1_no_usa[-mask1]
# Tests
print('Is US present?', df1_granular['State'].isin(['United States']).any())
print('Is "All causes" present?', df1_granular['Cause Name'].isin(['All causes']).
any())
print('Total unique states:', len(df1_granular['State'].unique()))
print('Total unique causes:', len(df1_granular['Cause Name'].unique()))
```

```
Is US present? False
Is "All causes" present? False
Total unique states: 51
Total unique causes: 10
```

Part 3.2:

We group the dataframe by State and Cause Name, and aggregate (sum) the Deaths. Then we sort the values and obtain the first four for each State. Our final dataframe (state_causes) has, for each state, the four leading causes of death and their respective number of deaths. The resulting table repeats the State name for all its constituent Causes. This has not been removed since this is just an intermediate step.

In [11]:
```
group1 = df1_granular.groupby(['State', 'Cause Name'])
state_causes_temp = group1.agg({'Deaths':sum})
state_causes = state_causes_temp['Deaths'].groupby(level=0, group_keys=False).nlar
gest(4).reset_index()
state_causes.head(5)
```

Out[11]:

	State	Cause Name	Deaths
0	Alabama	Heart disease	227433
1	Alabama	Cancer	180780
2	Alabama	Stroke	51507
3	Alabama	CLRD	47778
4	Alaska	Cancer	15032

Important secondary concept used below

When comparing lists for equality, the order matters. If the set of a list is compared with the set of another list, the order of values does not matter.

In [12]:
```
a = [1,2,3,4]
b = [4,3,2,1]
print(a == b)
print(set(a) == set(b))
```

```
False
True
```

Part 3.3:

The table from Part 3.2 is large so in this part, we will create some metrics to answer the question: if states show the same leading causes of deaths as US as a whole. We group our resulting table from 3.2 by State and store in group2. Then we create an empty list state_causes_bool. We run a for loop for group2. In each iteration, we store the four leading causes of death of that state in temp. Then we compare the set(temp) with the set(usa_causes_list) - prepared in Part 2.2. If the leading causes match (irrespective of order), we append a True to our boolean list, otherwise False. At the end, we check the length of boolean list to see how many states were compared with US. We also sum the list to get a total number of states that had the same leading causes as US. And then calculate the percentage.

In [13]:
```
group2 = state_causes.groupby(['State'])

state_causes_bool = []
for key, item in group2:
    temp = list(group2.get_group(key)['Cause Name'])
    if (set(temp) == set(usa_causes_list)):
        state_causes_bool.append(True)
    else:
        state_causes_bool.append(False)

print('Total number of states compared with USA leading causes:', len(state_causes
_bool))
print('Number of states with the same leading causes as USA:', sum(state_causes_bo
ol))
print('Rounded Percentage of states showing the same leading causes as USA:', roun
d((sum(state_causes_bool)/len(state_causes_bool))*100), '%')
```

```
Total number of states compared with USA leading causes: 51
Number of states with the same leading causes as USA: 30
Rounded Percentage of states showing the same leading causes as USA: 59 %
```

Part 3.4 - Results:

We observe and calculate that **about 59% of states had the same 4 leading causes**, others had slightly different four leading causes. Not all states show the same leading causes of death as US.

Part 4

This part required us to check if there were year-by-year changes in the four leading causes of death "Nationwide", in other words, in the US. We need a dataframe that only has State as 'United States' and **does not** have 'All Causes' as a Cause name. Our goal was to get a dataframe that shows the four leading causes of death in US for each year.

Part 4.1:

We will use the dataframe df1_usa_filtered created in Part 2.1. We group the data by Year and Cause Name, and aggregate (sum) for Deaths. We pick the four leading Causes in each Year only with their respective deaths. Resulting dataframe (year_causes) shows the required information.

In [14]:
```
group3 = df1_usa_filtered.groupby(['Year', 'Cause Name'])
year_causes_temp = group3.agg({'Deaths':sum})
year_causes = year_causes_temp['Deaths'].groupby(level=0, group_keys=False).nlargest(4).reset_index()
year_causes.head()
```

Out[14]:

	Year	Cause Name	Deaths
0	1999	Heart disease	725192
1	1999	Cancer	549838
2	1999	Stroke	167366
3	1999	CLRD	124181
4	2000	Heart disease	710760

Part 4.2:

We do the same calculations as in Part 3.3. Only difference is that here, we have grouped by Year and not State.

In [15]:
```
group4 = year_causes.groupby(['Year'])

year_causes_bool = []
for key, item in group4:
    temp = list(group4.get_group(key)['Cause Name'])
    if (set(temp) == set(usa_causes_list)):
        year_causes_bool.append(True)
    else:
        year_causes_bool.append(False)

print('Total number of years for which leading causes were compared:', len(year_causes_bool))
print('Total Number of years that had the same 4 leading causes of death as US:', sum(year_causes_bool))
print('Rounded percentage of years which had the same 4 leading causes of death:', round((sum(year_causes_bool)/len(year_causes_bool))*100), '%')
```

```
Total number of years for which leading causes were compared: 18
Total Number of years that had the same 4 leading causes of death as US: 14
Rounded percentage of years which had the same 4 leading causes of death: 78 %
```

Part 4.3 - Results:

We can see that **about 78% of the years** had the same four leading causes of death. Only 22% of the years did not. Printing the data showed that the causes began to vary only in the last few years and that too slightly. So there are minimal yearly changes in the leading causes of death.

D. Conclusion

Through this project, we saw how Pandas can be effectively used for data management and analysis. By working through the two datasets, we generated insights as to whether the death rate is increasing or decreasing, what are the leading causes of death nationwide, how similar are the leading causes of death in individual states compared to national, and finally, what changes are observed in these leading causes year by year. Pandas allowed us to answer broad questions statiscally.