DATA MENTORSHIP PROGRAM

This document teaches basic and advanced statistical problem-solving using python. Please ensure that you have downloaded Python on your computer. The datasets are saved in Google sheet, please download the sheet on your computer before uploading the dataset to Python

Owner: mdave5@horizon.csueastbay.edu Rajvi Mehta

Basic Statistics using Python

1. **Statistics in Python - why is it important?**
2. **Introduction to NumPy and Pandas and other common python libraries for basic statistics**
3. **Basic Descriptive analysis in Python (mean, median, mode, standard deviation)**
4. **Advanced Statistics in Python (Hypothesis testing, introduction to pvalue, probability, and sampling distributions)**
5. **Simple Linear regression**

Statistics in Python

Statistics refers to the *art and science* of collecting, analyzing, presenting and interpreting data. As a data professional, your knowledge of statistics will be crucial as you progress through your data career. Statistical knowledge will help you understand the outcome of your analysis/research better while helping to solve your client's problem.

This exercise will help you conduct basic and advanced statistical processes using Python. To get a better understanding of some of the most commonly used statistical terms, refer to this blog.

Choosing your Python Statistics Libraries

1. **Python Statistics:** Built-in library for descriptive statistics. Use it when your dataset is not too large.
2. **Numpy:** Third-party library for numerical computing.
3. **Scipy:** This library is based on Numpy. It offers additional functionality (including scipy. stats) for statistical analysis.
4. **Pandas:** This is a third-party library based on Numpy. Great for handling labeled one-dimensional data with series and two-dimensional data like DataFrame.
5. **Matplotlib:** Used mainly for data visualization.
6. **Seaborn:** Data visualization library built on top of Matplotlib to create statistical graphs. Used to explore data relationships and machine learning models.

Part 3 - Exercise for Basic Statistics in Python

Basic Descriptive analysis in Python (mean, median, mode, standard deviation)

Owner: mdave5@horizon.csueastbay.edu

Goal: This document shares a step-by-step guide on conducting descriptive statistics using Jupyter (Python script)

Recommended to go through this blog for people who do not have prior knowledge of statistics.

Data Set: We will be using the Top 50 Billionaires dataset for this project.

Data Dictionary: This dataset captures ranks, names, net worth, ages, financial sources, and countries of the top 50 billionaires.

- Rank - Numeric
- Name - Text
- Net Worth - Currency / Numeric
- Age - Numeric
- Financial Source - Text
- Countries - Text

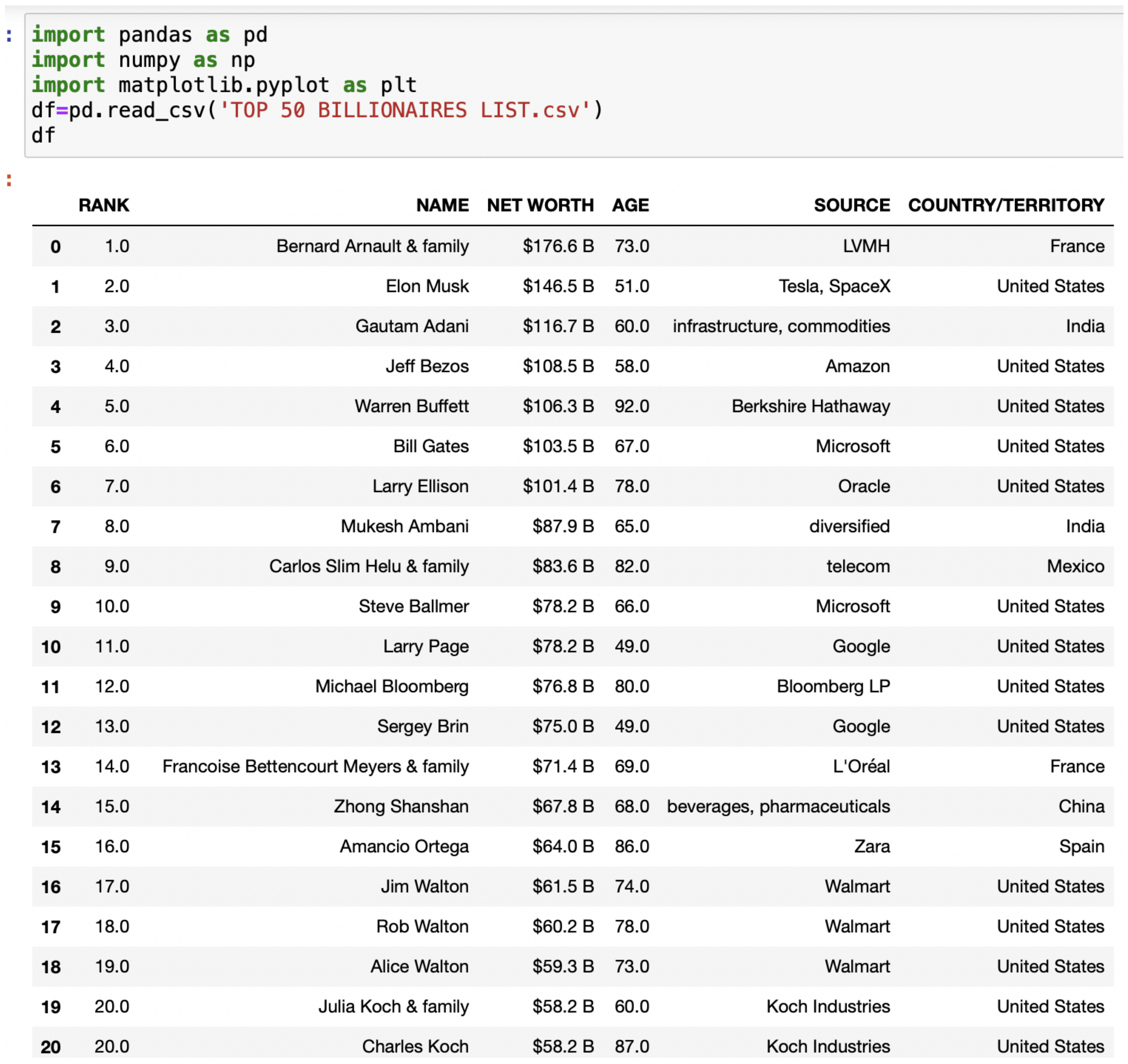
Data Questions to answer for this exercise

1. What are the average and median ages of the top 50 billionaires? (data. describe) (***basic EDA***)
2. What are the common trends among the top 10 billionaires and compare them to the bottom 10 billionaires? (bar graph, stacked bar chart)

3. Which continent have the most billionaires? (group by)
4. What is the correlation between age and net worth? Graphical Representation (scatter plot)
5. Which COUNTRY has the highest proportion of billionaires? Which country has the highest probability of billionaires whose net worth will be above average(group by sources)
6. ANOVA F-test (United States vs. other countries)

1) Basic descriptive analysis (includes pictures)

- **Reading data -**



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv('TOP 50 BILLIONAIRES LIST.csv')
df
```

	RANK	NAME	NET WORTH	AGE	SOURCE	COUNTRY/TERRITORY
0	1.0	Bernard Arnault & family	$176.6 B	73.0	LVMH	France
1	2.0	Elon Musk	$146.5 B	51.0	Tesla, SpaceX	United States
2	3.0	Gautam Adani	$116.7 B	60.0	infrastructure, commodities	India
3	4.0	Jeff Bezos	$108.5 B	58.0	Amazon	United States
4	5.0	Warren Buffett	$106.3 B	92.0	Berkshire Hathaway	United States
5	6.0	Bill Gates	$103.5 B	67.0	Microsoft	United States
6	7.0	Larry Ellison	$101.4 B	78.0	Oracle	United States
7	8.0	Mukesh Ambani	$87.9 B	65.0	diversified	India
8	9.0	Carlos Slim Helu & family	$83.6 B	82.0	telecom	Mexico
9	10.0	Steve Ballmer	$78.2 B	66.0	Microsoft	United States
10	11.0	Larry Page	$78.2 B	49.0	Google	United States
11	12.0	Michael Bloomberg	$76.8 B	80.0	Bloomberg LP	United States
12	13.0	Sergey Brin	$75.0 B	49.0	Google	United States
13	14.0	Francoise Bettencourt Meyers & family	$71.4 B	69.0	L'Oréal	France
14	15.0	Zhong Shanshan	$67.8 B	68.0	beverages, pharmaceuticals	China
15	16.0	Amancio Ortega	$64.0 B	86.0	Zara	Spain
16	17.0	Jim Walton	$61.5 B	74.0	Walmart	United States
17	18.0	Rob Walton	$60.2 B	78.0	Walmart	United States
18	19.0	Alice Walton	$59.3 B	73.0	Walmart	United States
19	20.0	Julia Koch & family	$58.2 B	60.0	Koch Industries	United States
20	20.0	Charles Koch	$58.2 B	87.0	Koch Industries	United States

Question 1: What are the average and median ages of the top 50 billionaires?

- **Basic EDA**

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   RANK               50 non-null     float64
 1   NAME               50 non-null     object 
 2   NET WORTH          50 non-null     object 
 3   AGE                49 non-null     float64
 4   SOURCE             50 non-null     object 
 5   COUNTRY/TERRITORY  50 non-null     object 
dtypes: float64(2), object(4)
memory usage: 2.5+ KB
```

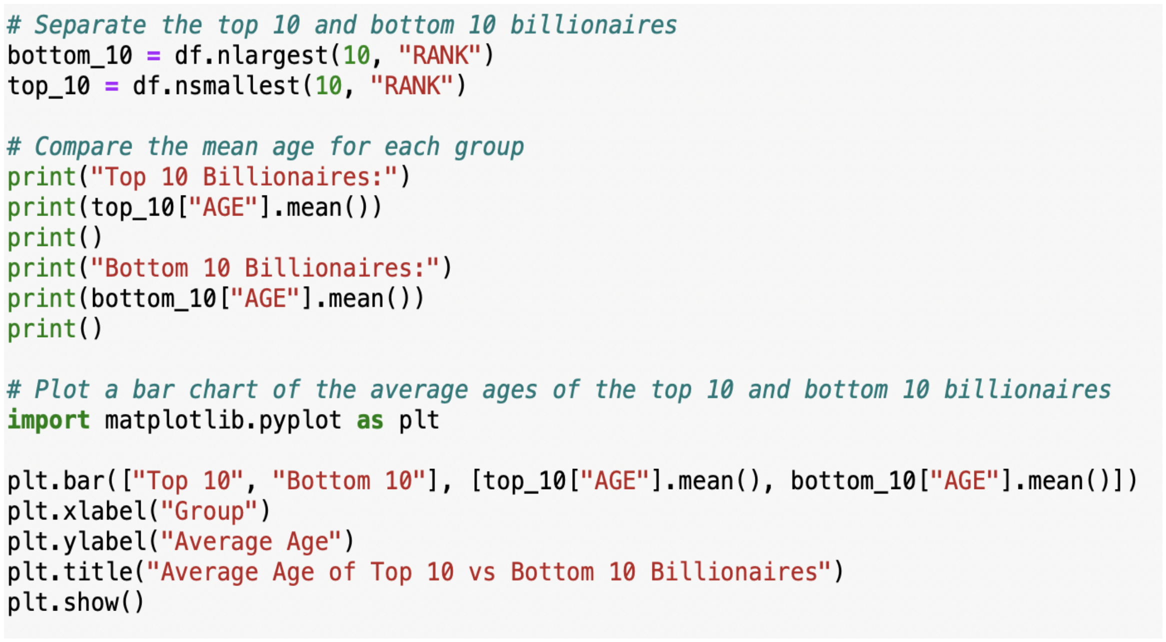
```python
df.describe()
```

	RANK	AGE
count	50.00000	49.000000
mean	25.46000	68.816327
std	14.56752	14.895012
min	1.00000	38.000000
25%	13.25000	58.000000
50%	25.50000	69.000000
75%	37.00000	82.000000
max	50.00000	94.000000

The average (mean) and the median age of the top 50 billionaires are 25.46 years and 25.5 years respectively. This implies that the dataset is roughly symmetric and normally distributed as the mean is almost equal to the median.

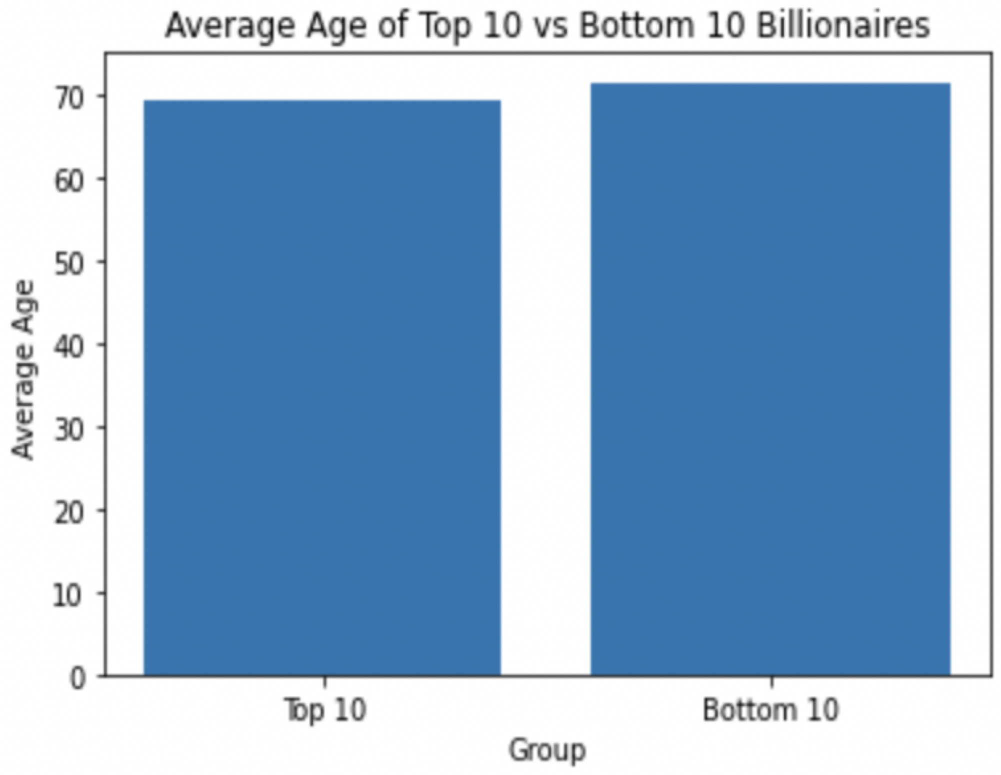
Question 2: What are the common trends among the top 10 billionaires and compare them to the bottom 10 billionaires?

- Comparison by age



```python
# Separate the top 10 and bottom 10 billionaires
bottom_10 = df.nlargest(10, "RANK")
top_10 = df.nsmallest(10, "RANK")

# Compare the mean age for each group
print("Top 10 Billionaires:")
print(top_10["AGE"].mean())
print()
print("Bottom 10 Billionaires:")
print(bottom_10["AGE"].mean())
print()

# Plot a bar chart of the average ages of the top 10 and bottom 10 billionaires
import matplotlib.pyplot as plt

plt.bar(["Top 10", "Bottom 10"], [top_10["AGE"].mean(), bottom_10["AGE"].mean()])
plt.xlabel("Group")
plt.ylabel("Average Age")
plt.title("Average Age of Top 10 vs Bottom 10 Billionaires")
plt.show()
```

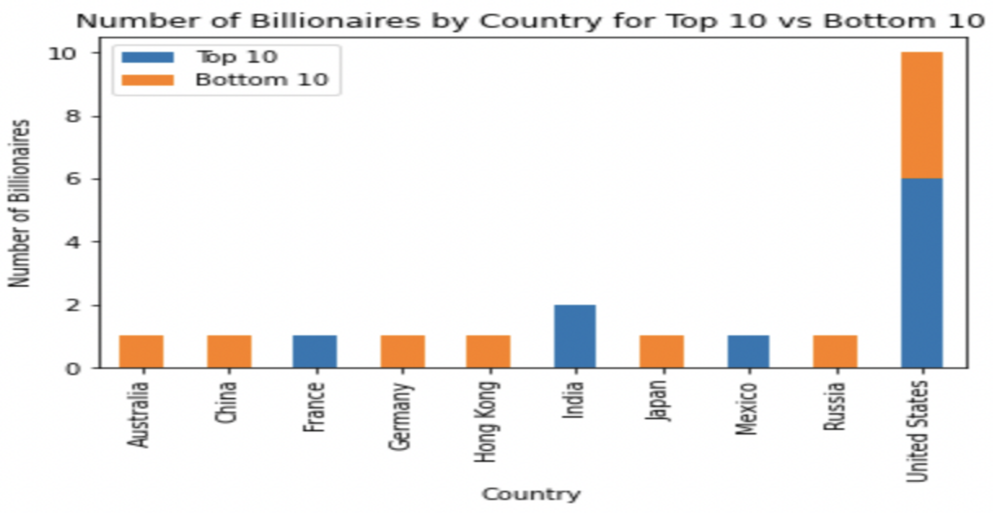
```
Top 10 Billionaires:
69.2

Bottom 10 Billionaires:
71.4
```



- Comparison by country

```python
# Get the number of billionaires by country for each group
print("Top 10 Billionaires by Country:")
print(top_10["COUNTRY/TERRITORY"].value_counts())
print()
print("Bottom 10 Billionaires by Country:")
print(bottom_10["COUNTRY/TERRITORY"].value_counts())
top_10_counts = top_10["COUNTRY/TERRITORY"].value_counts()
bottom_10_counts = bottom_10["COUNTRY/TERRITORY"].value_counts()

# Combine the two series into a single DataFrame for plotting
counts = pd.DataFrame({"Top 10": top_10_counts, "Bottom 10": bottom_10_counts})
counts = counts.fillna(0).astype(int)

# Plot the stacked bar chart
counts.plot(kind="bar", stacked=True)
plt.xlabel("Country")
plt.ylabel("Number of Billionaires")
plt.title("Number of Billionaires by Country for Top 10 vs Bottom 10")
plt.legend(loc="upper left")
plt.show()
```

```
Top 10 Billionaires by Country:
United States    6
India            2
France           1
Mexico           1
Name: COUNTRY/TERRITORY, dtype: int64

Bottom 10 Billionaires by Country:
United States    4
Russia           1
Australia        1
Germany          1
China            1
Japan            1
Hong Kong        1
Name: COUNTRY/TERRITORY, dtype: int64
```



Question 3: Which country has the most billionaires?

```python
# Group the data by country/territory and count the number of billionaires in each group
grouped = df.groupby("COUNTRY/TERRITORY")["RANK"].count()

# Find the country/territory with the most billionaires
most_billionaires_country = grouped.idxmax()

print("The country/territory with the most billionaires is:", most_billionaires_country)
```

```
The country/territory with the most billionaires is: United States
```

Question 4: What is the correlation between age and net worth?

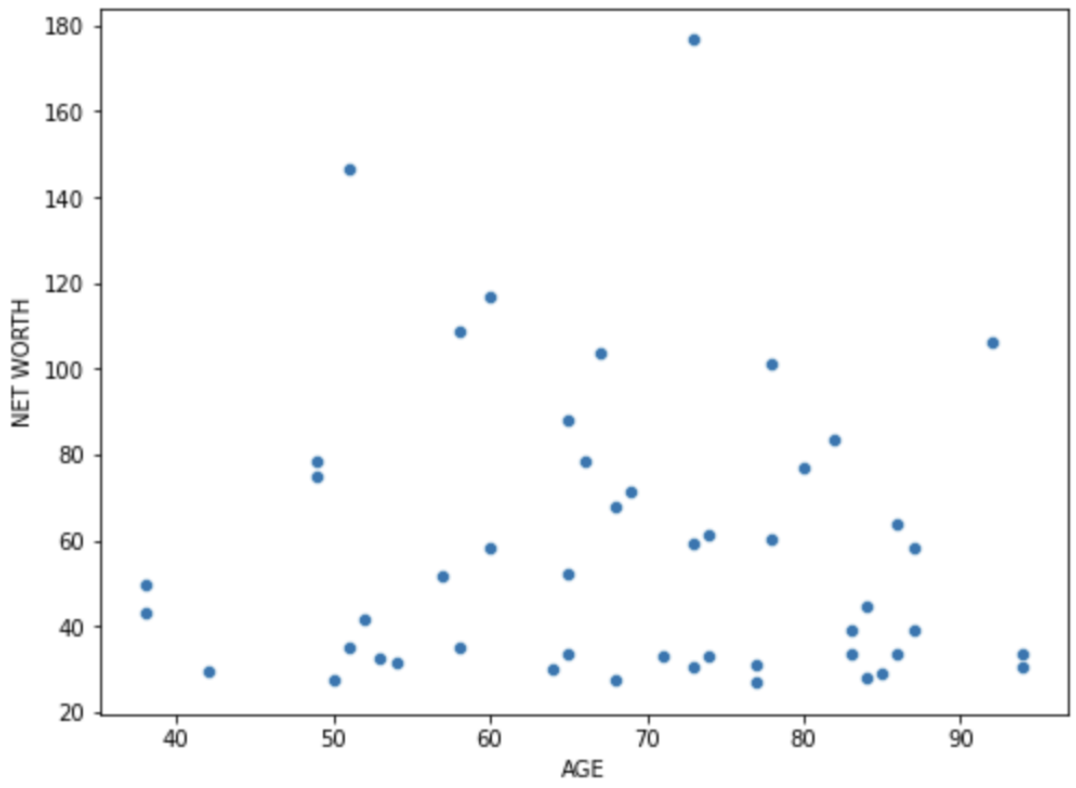
Since, net worth is a combination of currency, numbers, and strings, we first need to remove the ‘$’ and ‘B’ characters from the entire Net Worth column to convert it to float to do any computations.

```python
# replace the values and change the data type of the 'NET WORTH' column
df['NET WORTH'] = df['NET WORTH'].replace({'\$': '', 'B': ''}, regex=True).astype(float)

# Find correlation between 'AGE' and 'NET WORTH'
correlation = df['AGE'].corr(df['NET WORTH'])

print("The correlation between age and net worth of the top 50 billionaires is:", correlation)
```

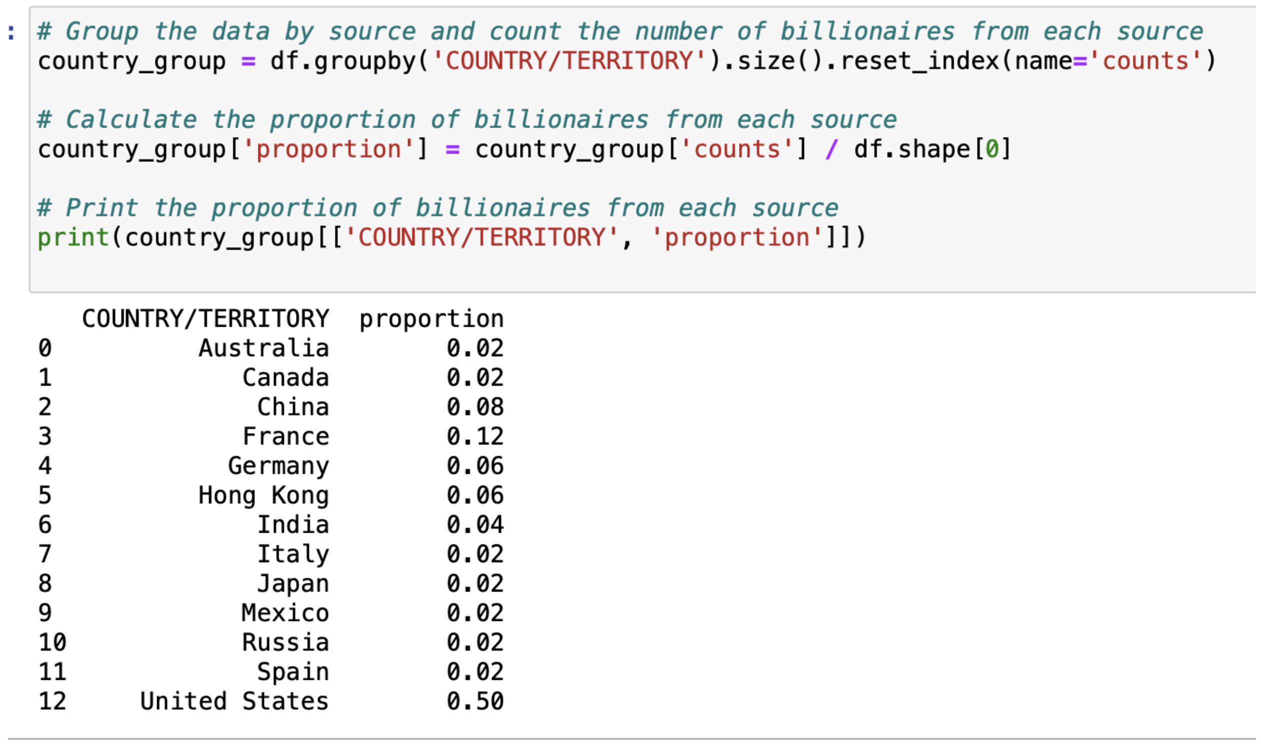
```
The correlation between age and net worth of the top 50 billionaires is: -0.05763934928266901
```

A correlation coefficient of -0.058 indicates a small negative linear relationship between age and net worth. This suggests that there is a weak linear relationship between the two variables and that changes in one variable are not strongly associated with changes in the other. In this case, the negative sign indicates that as age increases, the net worth decreases, although the magnitude of the correlation is small.

```python
import seaborn as sns

# Plot the scatterplot
plt.figure(figsize=(8, 6))
sns.scatterplot(x="AGE", y="NET WORTH", data=df)
plt.show()
```



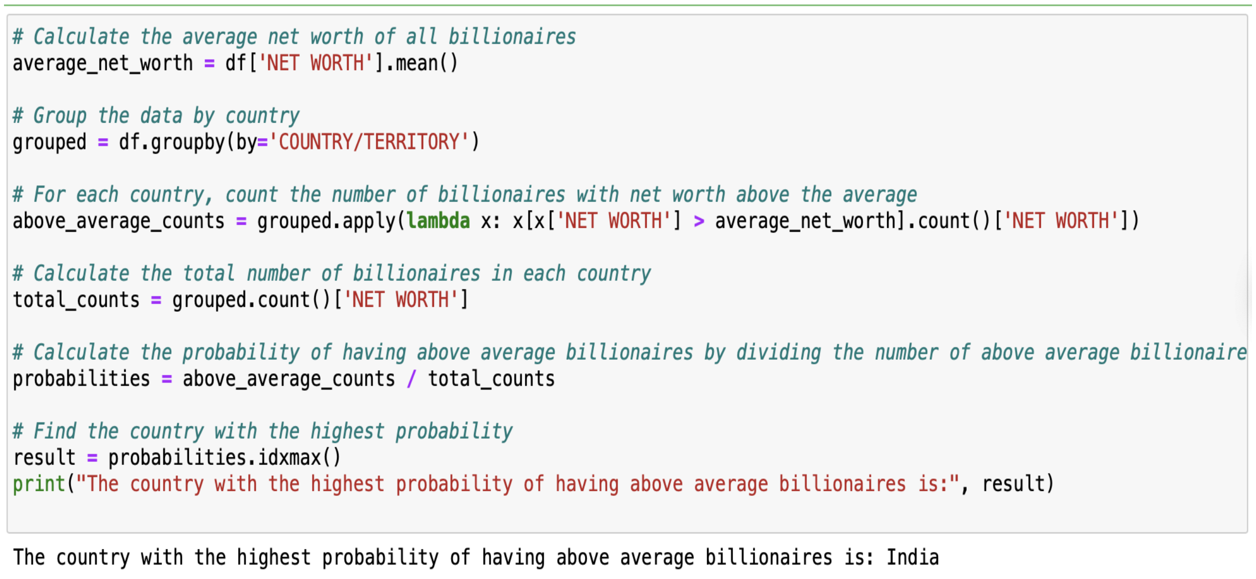
Question 5. Which country has the highest proportion of billionaires?



```
# Group the data by source and count the number of billionaires from each source
country_group = df.groupby('COUNTRY/TERRITORY').size().reset_index(name='counts')

# Calculate the proportion of billionaires from each source
country_group['proportion'] = country_group['counts'] / df.shape[0]

# Print the proportion of billionaires from each source
print(country_group[['COUNTRY/TERRITORY', 'proportion']])
```

```
   COUNTRY/TERRITORY  proportion
0          Australia        0.02
1             Canada        0.02
2              China        0.08
3             France        0.12
4            Germany        0.06
5          Hong Kong        0.06
6              India        0.04
7              Italy        0.02
8              Japan        0.02
9             Mexico        0.02
10            Russia        0.02
11             Spain        0.02
12     United States        0.50
```

Alternatively, to find which country has the highest probability of billionaires whose net worth will be above average (among the top 50), we can calculate the average net worth of all billionaires and then count the number of billionaires in each country whose net worth is above the average. Finally, divide this number by the total number of billionaires in each country to get the probability of having above-average billionaires in that country.



```
# Calculate the average net worth of all billionaires
average_net_worth = df['NET WORTH'].mean()

# Group the data by country
grouped = df.groupby(by='COUNTRY/TERRITORY')

# For each country, count the number of billionaires with net worth above the average
above_average_counts = grouped.apply(lambda x: x[x['NET WORTH'] > average_net_worth].count()['NET WORTH'])

# Calculate the total number of billionaires in each country
total_counts = grouped.count()['NET WORTH']

# Calculate the probability of having above average billionaires by dividing the number of above average billionaire
probabilities = above_average_counts / total_counts

# Find the country with the highest probability
result = probabilities.idxmax()
print("The country with the highest probability of having above average billionaires is:", result)
```

```
The country with the highest probability of having above average billionaires is: India
```

Part 4 Advanced Statistical Analysis using Python

Advanced Statistics in Python (Hypothesis testing, introduction to p-value, probability, and sampling distributions)

Owner: Rajvi Mehta

Goal: This document shares a step-by-step guide on testing hypotheses in Jupyter (Python)

Data Set: We will be using the 50 Start-Ups dataset for this project.

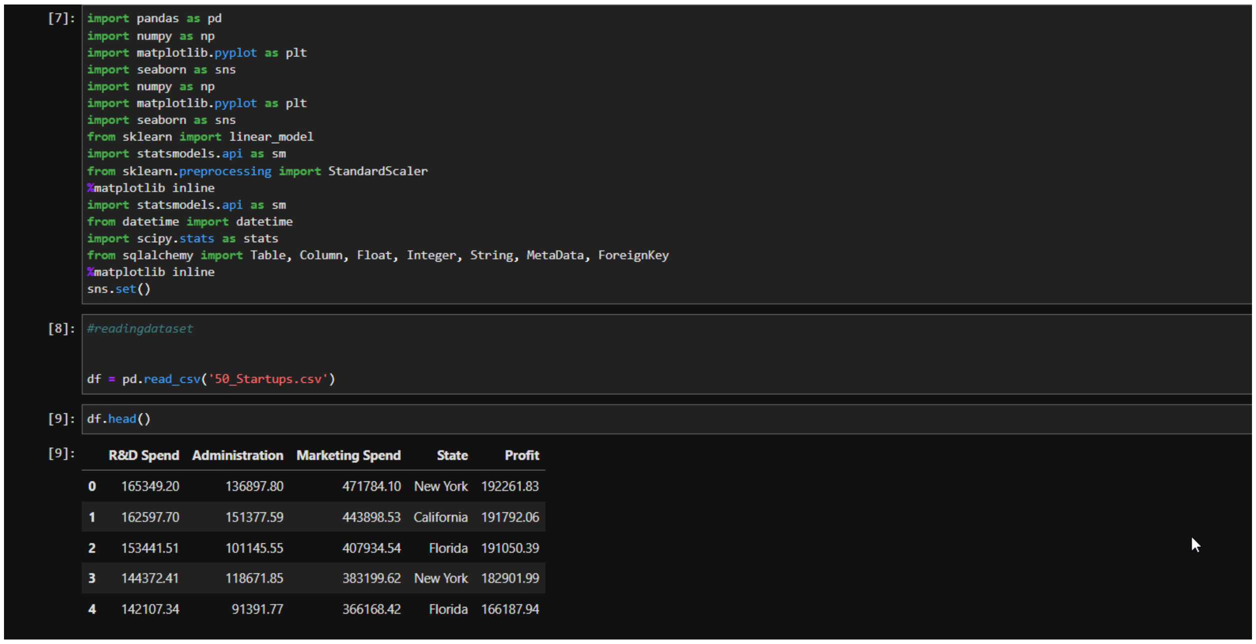
Please read this blog before proceeding.

Data Dictionary: This dataset captures R&D Spend, Administration cost, Marketing Spend, State, and Profit for 50 startups.

All data is in numeric, currencies.

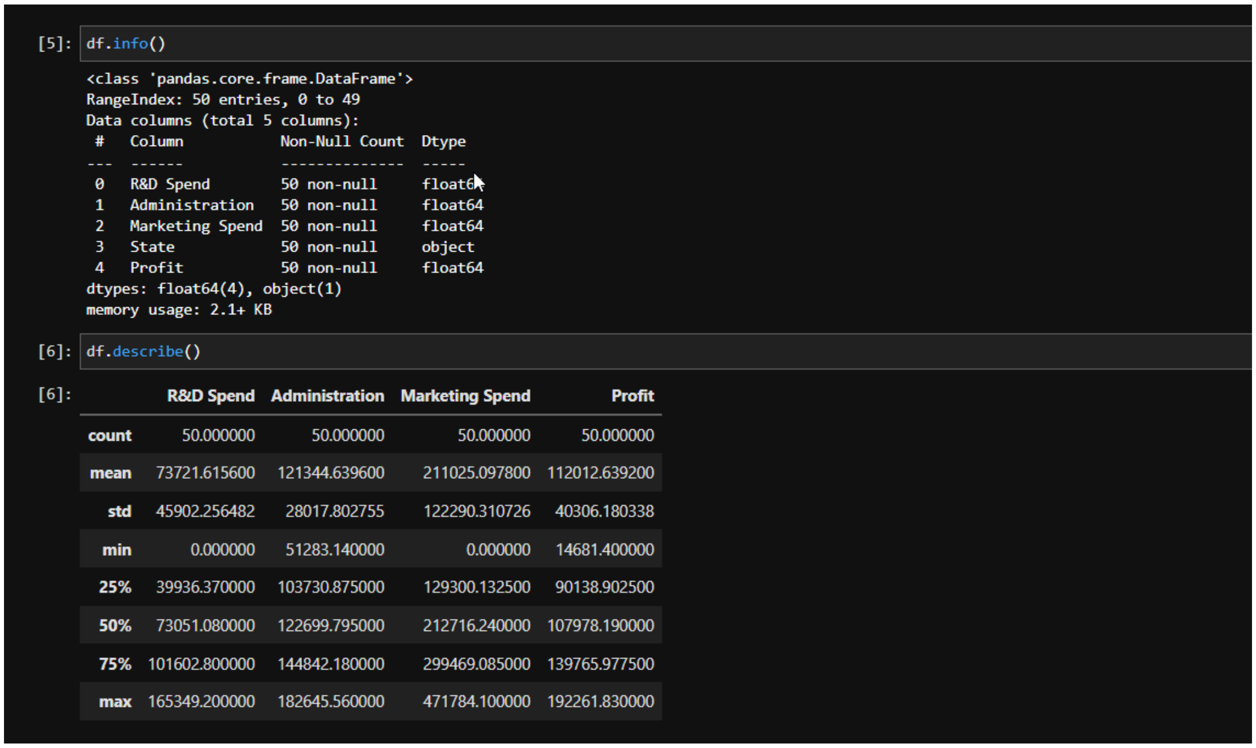
Topics covered in this: EDA, Testing for Normality, T-test, Hypothesis testing

Step 1: Upload the dataset



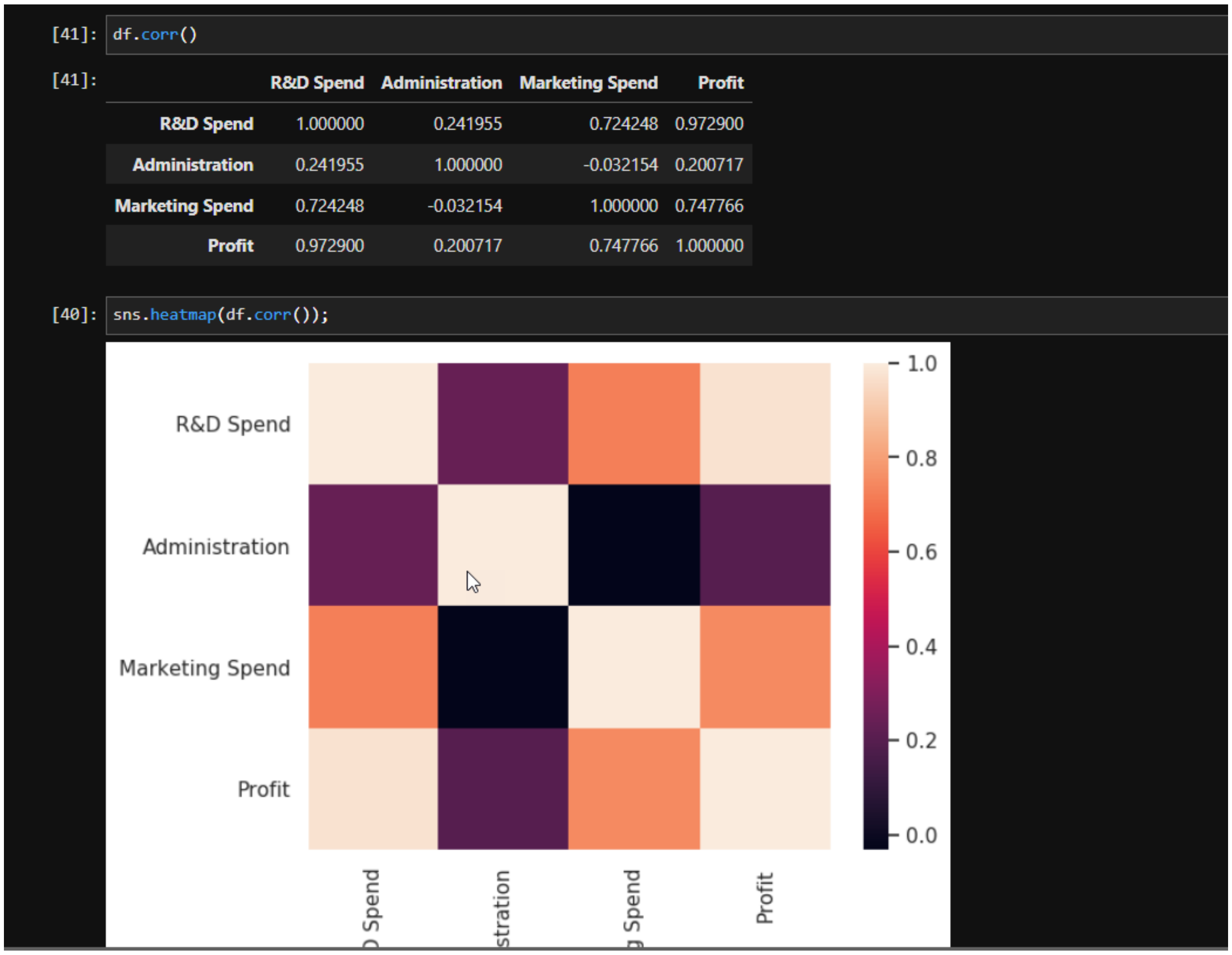
```
[7]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn import linear_model
     import statsmodels.api as sm
     from sklearn.preprocessing import StandardScaler
     %matplotlib inline
     import statsmodels.api as sm
     from datetime import datetime
     import scipy.stats as stats
     from sqlalchemy import Table, Column, Float, Integer, String, MetaData, ForeignKey
     %matplotlib inline
     sns.set()

[8]: #readingdataset


     df = pd.read_csv('50_Startups.csv')

[9]: df.head()

[9]:    R&D Spend  Administration  Marketing Spend       State     Profit
     0  165349.20       136897.80        471784.10    New York  192261.83
     1  162597.70       151377.59        443898.53  California  191792.06
     2  153441.51       101145.55        407934.54     Florida  191050.39
     3  144372.41       118671.85        383199.62    New York  182901.99
     4  142107.34        91391.77        366168.42     Florida  166187.94
```

Step 2: Conduct basic EDA



```
[5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB

[6]: df.describe()

[6]:
```

	R&D Spend	Administration	Marketing Spend	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200
std	45902.256482	28017.802755	122290.310726	40306.180338
min	0.000000	51283.140000	0.000000	14681.400000
25%	39936.370000	103730.875000	129300.132500	90138.902500
50%	73051.080000	122699.795000	212716.240000	107978.190000
75%	101602.800000	144842.180000	299469.085000	139765.977500
max	165349.200000	182645.560000	471784.100000	192261.830000



```
[41]: df.corr()
```

```
[41]:
                  R&D Spend  Administration  Marketing Spend    Profit
R&D Spend          1.000000        0.241955         0.724248  0.972900
Administration     0.241955        1.000000        -0.032154  0.200717
Marketing Spend    0.724248       -0.032154         1.000000  0.747766
Profit             0.972900        0.200717         0.747766  1.000000
```

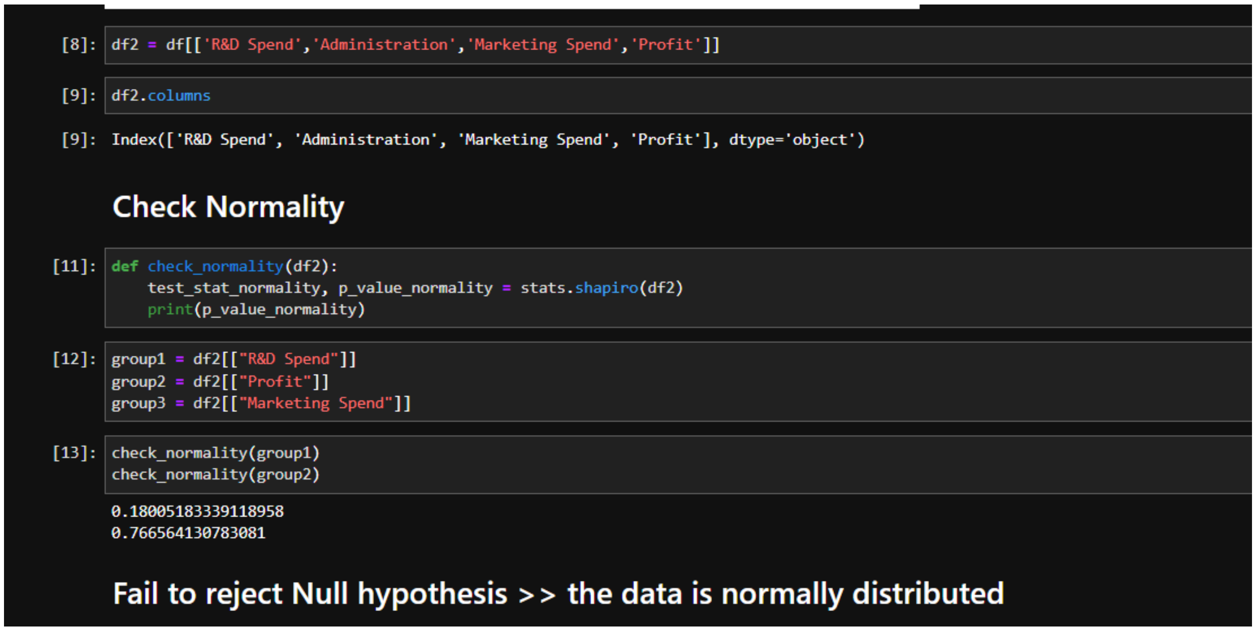
```
[40]: sns.heatmap(df.corr());
```

Step 3: Check Normality

What is Normality in statistics? The normal distribution describes the shape of the data (normally distributed data has a bell curve shape). Here, we are using a hypothesis test for checking the normality of the data.

Null Hypothesis: The data is not normally distributed.

Alternative Hypothesis: The data is normally distributed,



```
[8]: df2 = df[['R&D Spend','Administration','Marketing Spend','Profit']]

[9]: df2.columns

[9]: Index(['R&D Spend', 'Administration', 'Marketing Spend', 'Profit'], dtype='object')
```

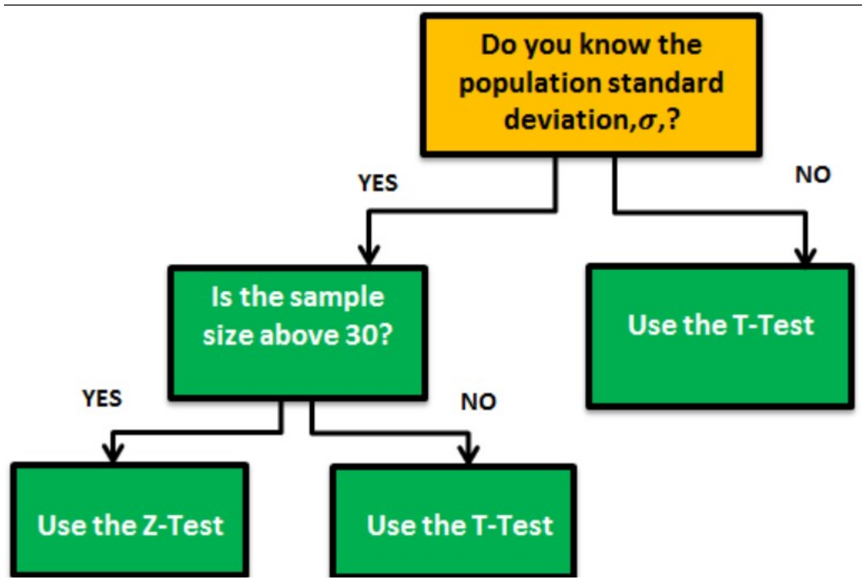
Check Normality

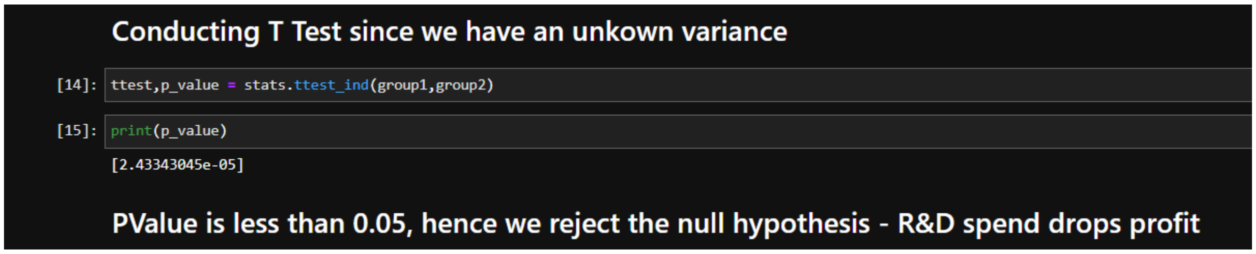
```
[11]: def check_normality(df2):
          test_stat_normality, p_value_normality = stats.shapiro(df2)
          print(p_value_normality)

[12]: group1 = df2[["R&D Spend"]]
      group2 = df2[["Profit"]]
      group3 = df2[["Marketing Spend"]]

[13]: check_normality(group1)
      check_normality(group2)

      0.18005183339118958
      0.766564130783081
```

Fail to reject Null hypothesis >> the data is normally distributed

Step 4: Conduct T Test

There are two types of tests that you can take at this point. T Test, Z test

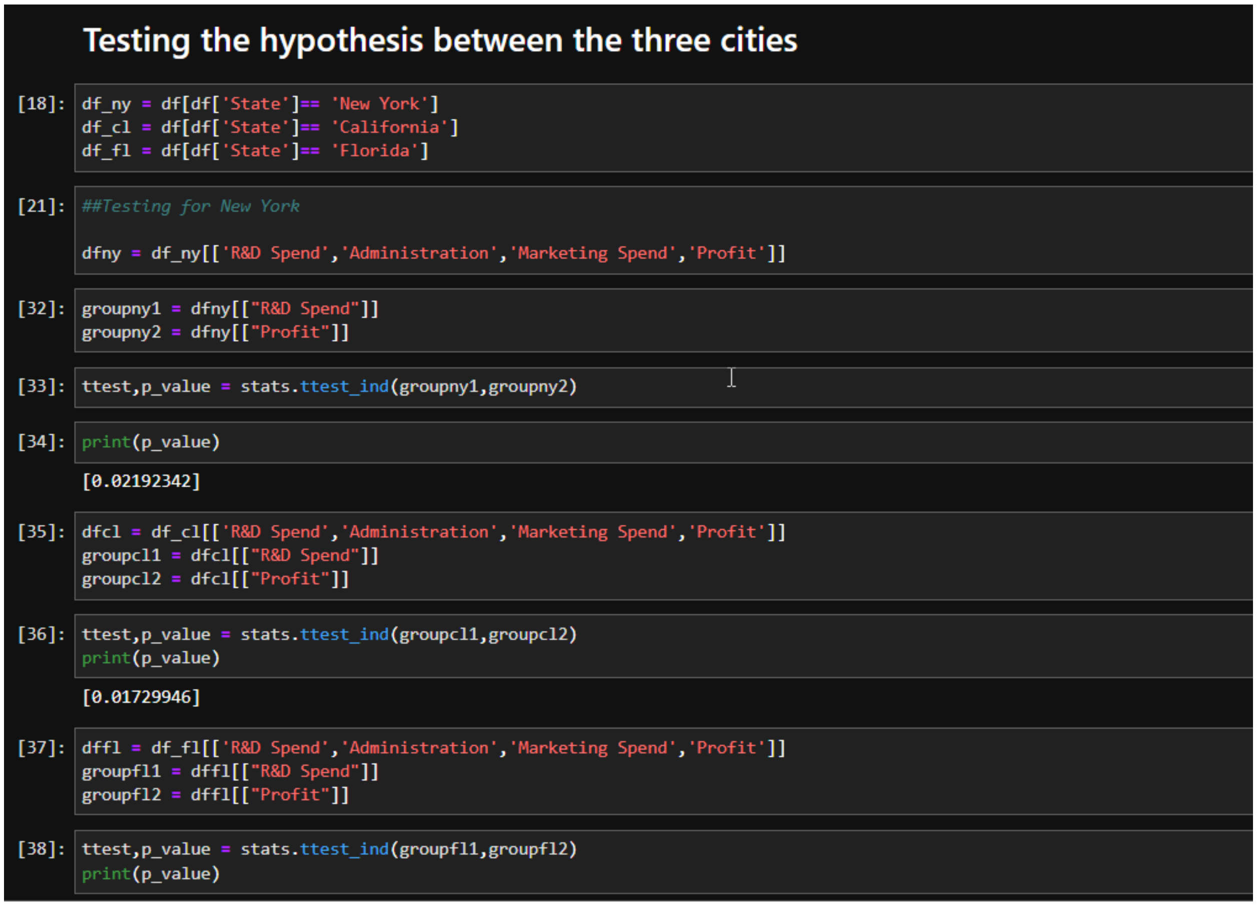

Do you know the population standard deviation,σ,?
YES
NO
Is the sample size above 30?
Use the T-Test
YES
NO
Use the Z-Test
Use the T-Test


Conducting T Test since we have an unkown variance

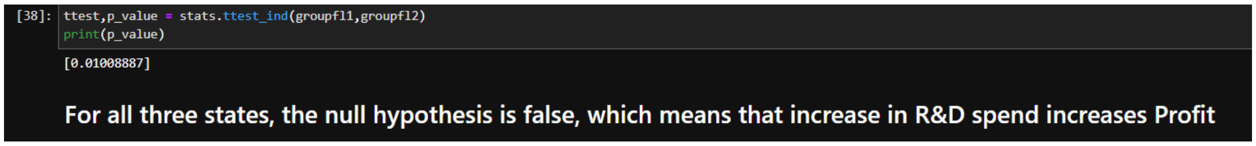
```
[14]: ttest,p_value = stats.ttest_ind(group1,group2)

[15]: print(p_value)

      [2.43343045e-05]
```

PValue is less than 0.05, hence we reject the null hypothesis - R&D spend drops profit

Step 5: Expand the Hypothesis to all three states.


Testing the hypothesis between the three cities

```
[18]: df_ny = df[df['State']== 'New York']
      df_cl = df[df['State']== 'California']
      df_fl = df[df['State']== 'Florida']

[21]: ##Testing for New York

      dfny = df_ny[['R&D Spend','Administration','Marketing Spend','Profit']]

[32]: groupny1 = dfny[["R&D Spend"]]
      groupny2 = dfny[["Profit"]]

[33]: ttest,p_value = stats.ttest_ind(groupny1,groupny2)

[34]: print(p_value)

      [0.02192342]

[35]: dfcl = df_cl[['R&D Spend','Administration','Marketing Spend','Profit']]
      groupcl1 = dfcl[["R&D Spend"]]
      groupcl2 = dfcl[["Profit"]]

[36]: ttest,p_value = stats.ttest_ind(groupcl1,groupcl2)
      print(p_value)

      [0.01729946]

[37]: dffl = df_fl[['R&D Spend','Administration','Marketing Spend','Profit']]
      groupfl1 = dffl[["R&D Spend"]]
      groupfl2 = dffl[["Profit"]]

[38]: ttest,p_value = stats.ttest_ind(groupfl1,groupfl2)
      print(p_value)
```



```
[38]: ttest,p_value = stats.ttest_ind(groupfl1,groupfl2)
      print(p_value)

      [0.01008887]
```

For all three states, the null hypothesis is false, which means that increase in R&D spend increases Profit