PRO-C113

Data Story -2

What is our GOAL for this MODULE?

The goal of this module is to learn about data visualization.

What did we ACHIEVE in the class TODAY?

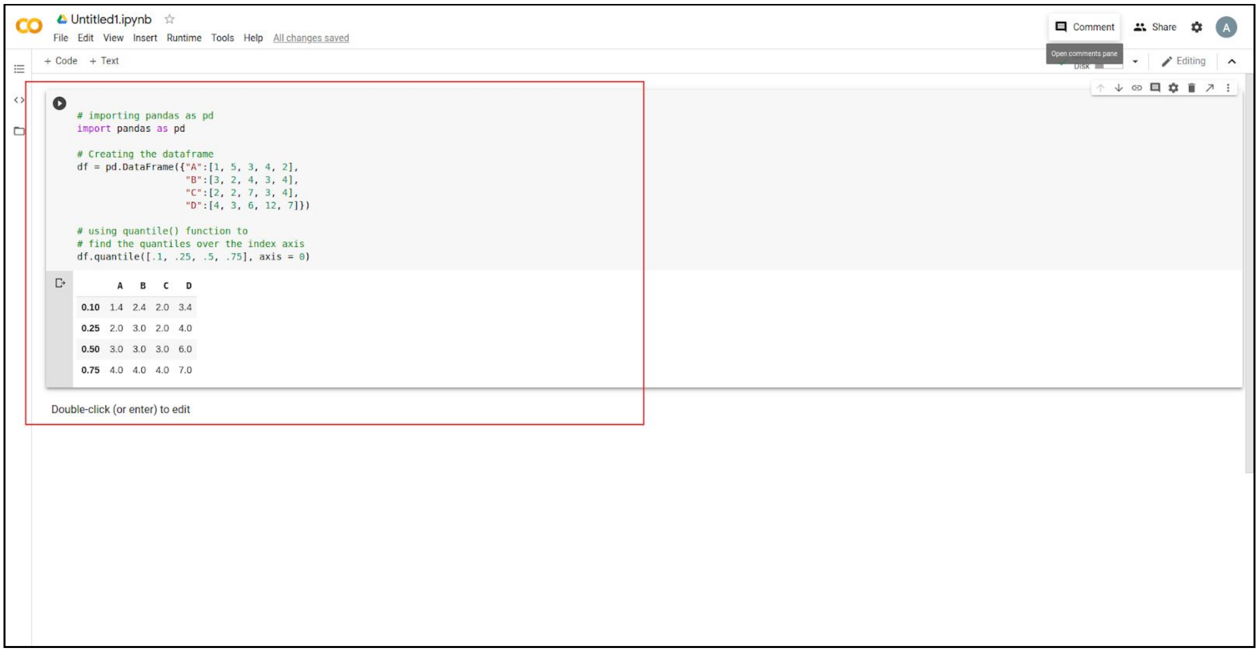
In this class we finished our data story and by looking at the outputs we gave a conclusion.

Which CONCEPTS/CODING BLOCKS did we cover today?

- Learned about IQR.
- Removed the outliers from the data.
- Calculated the z score.

How did we DO the activities?

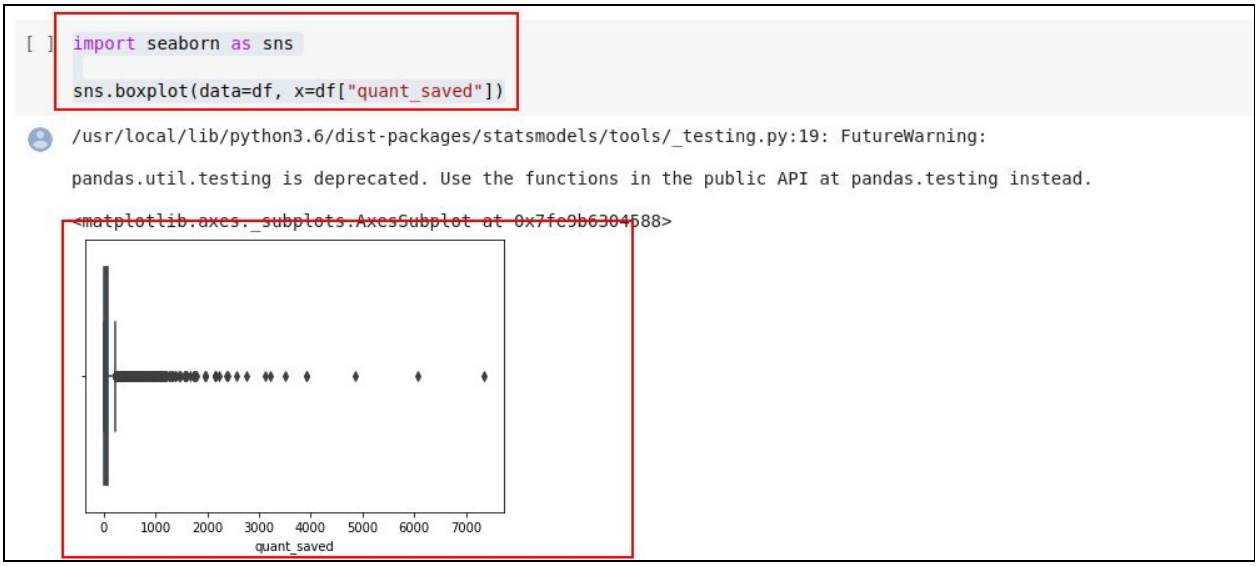
1. We found the InterQuantile Range of a sample data.


Untitled1.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment
Share
Open comments pane
Editing
Double-click (or enter) to edit

```
# importing pandas as pd
import pandas as pd

# Creating the dataframe
df = pd.DataFrame({"A":[1, 5, 3, 4, 2],
                   "B":[3, 2, 4, 3, 4],
                   "C":[2, 2, 7, 3, 4],
                   "D":[4, 3, 6, 12, 7]})

# using quantile() function to
# find the quantiles over the index axis
df.quantile([.1, .25, .5, .75], axis = 0)
```

```
        A    B    C    D
0.10  1.4  2.4  2.0  3.4
0.25  2.0  3.0  2.0  4.0
0.50  3.0  3.0  3.0  6.0
0.75  4.0  4.0  4.0  7.0
```

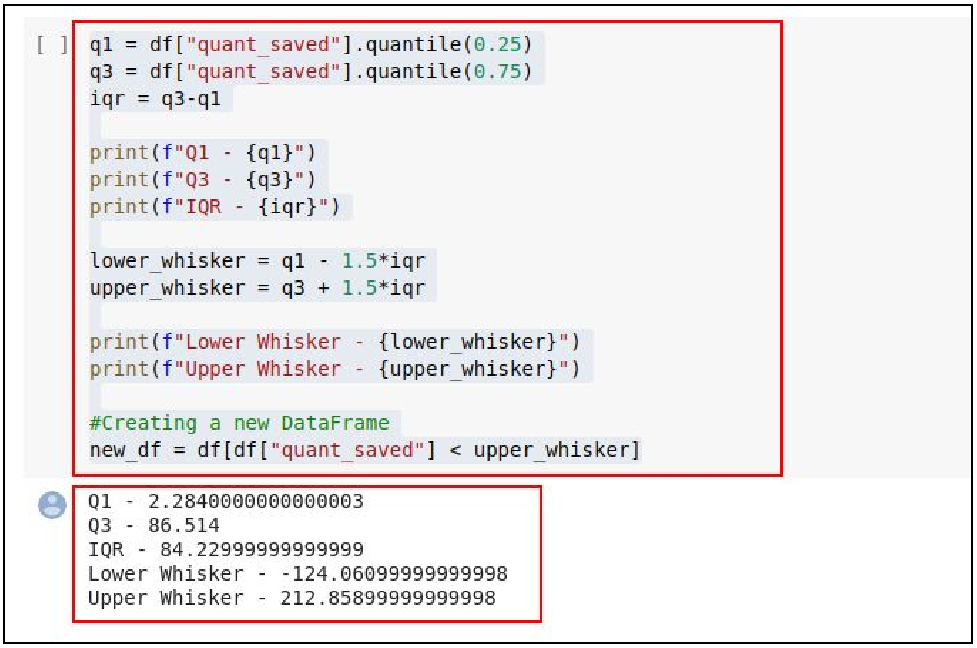
2. We used the box plot to check where the outlier lies.



```
import seaborn as sns

sns.boxplot(data=df, x=df["quant_saved"])
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
<matplotlib.axes._subplots.AxesSubplot at 0x7fe9b6304588>
```

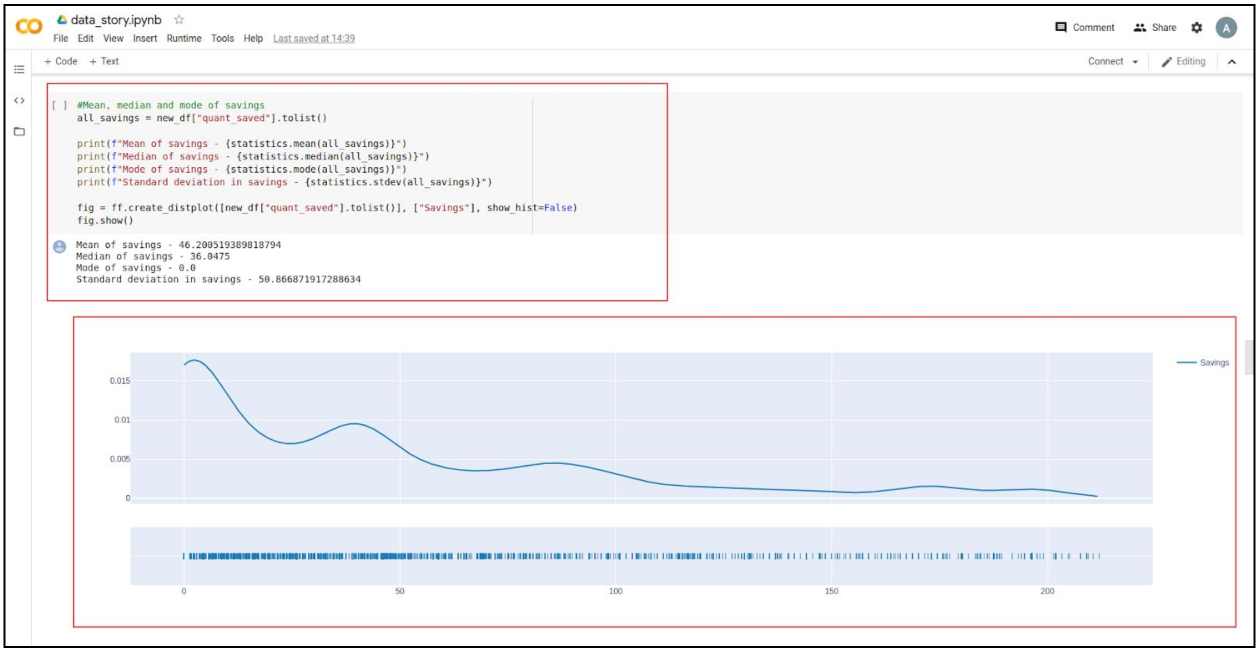
3. We found the IQR of our data.



```
[ ] q1 = df["quant_saved"].quantile(0.25)
    q3 = df["quant_saved"].quantile(0.75)
    iqr = q3-q1

    print(f"Q1 - {q1}")
    print(f"Q3 - {q3}")
    print(f"IQR - {iqr}")

    lower_whisker = q1 - 1.5*iqr
    upper_whisker = q3 + 1.5*iqr

    print(f"Lower Whisker - {lower_whisker}")
    print(f"Upper Whisker - {upper_whisker}")

    #Creating a new DataFrame
    new_df = df[df["quant_saved"] < upper_whisker]
```

```
Q1 - 2.2840000000000003
Q3 - 86.514
IQR - 84.22999999999999
Lower Whisker - -124.06099999999998
Upper Whisker - 212.85899999999998
```

4. We calculated the mean, median and mode along with the standard deviation of the new data and plotted the normal distribution.

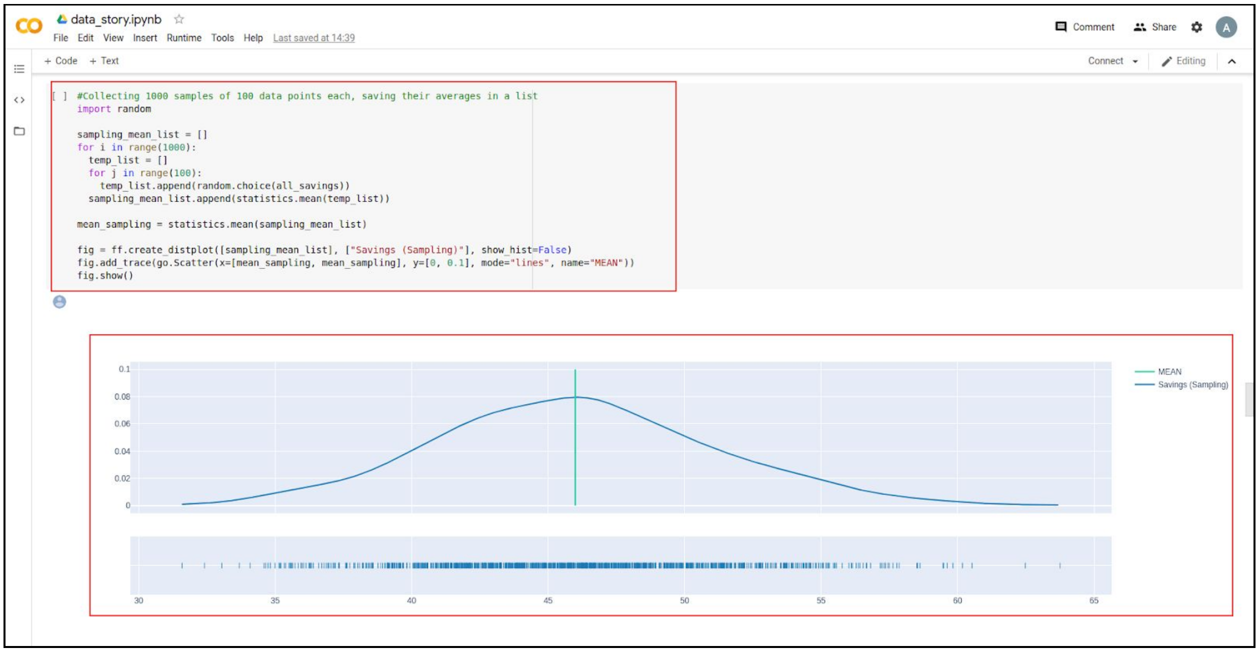


data_story.ipynb
File Edit View Insert Runtime Tools Help Last saved at 14:39
Comment Share
+ Code + Text
Connect
Editing

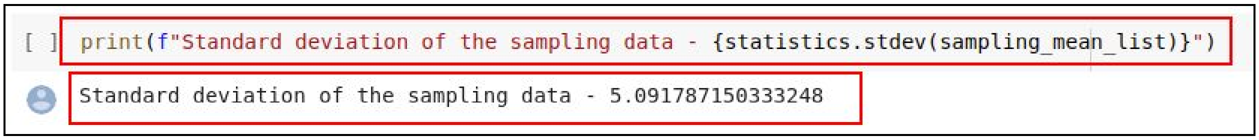
```
[ ] #Mean, median and mode of savings
    all_savings = new_df["quant_saved"].tolist()

    print(f"Mean of savings - {statistics.mean(all_savings)}")
    print(f"Median of savings - {statistics.median(all_savings)}")
    print(f"Mode of savings - {statistics.mode(all_savings)}")
    print(f"Standard deviation in savings - {statistics.stdev(all_savings)}")

    fig = ff.create_distplot([new_df["quant_saved"].tolist()], ["Savings"], show_hist=False)
    fig.show()
```

```
Mean of savings - 46.200519389818794
Median of savings - 36.0475
Mode of savings - 0.0
Standard deviation in savings - 50.866871917288634
```

5. We plotted the sampling mean on the graph and printed the standard deviation of the new data.



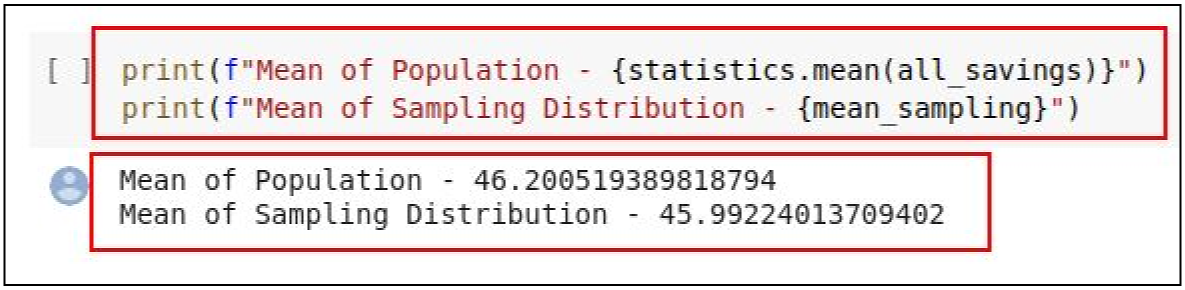
```
[ ] #Collecting 1000 samples of 100 data points each, saving their averages in a list
    import random

    sampling_mean_list = []
    for i in range(1000):
      temp_list = []
      for j in range(100):
        temp_list.append(random.choice(all_savings))
      sampling_mean_list.append(statistics.mean(temp_list))

    mean_sampling = statistics.mean(sampling_mean_list)

    fig = ff.create_distplot([sampling_mean_list], ["Savings (Sampling)"], show_hist=False)
    fig.add_trace(go.Scatter(x=[mean_sampling, mean_sampling], y=[0, 0.1], mode="lines", name="MEAN"))
    fig.show()
```



```
[ ] print(f"Standard deviation of the sampling data - {statistics.stdev(sampling_mean_list)}")
```

```
Standard deviation of the sampling data - 5.091787150333248
```

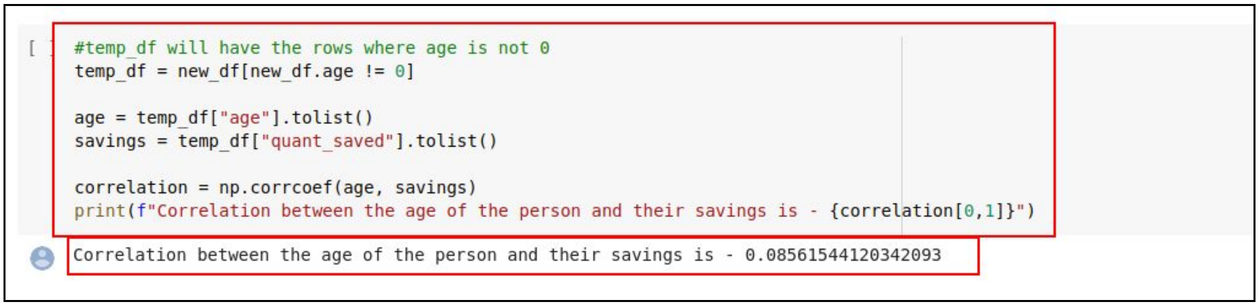
6. We calculated the mean of population and mean of sampling.



```
[ ] print(f"Mean of Population - {statistics.mean(all_savings)}")
    print(f"Mean of Sampling Distribution - {mean_sampling}")
```

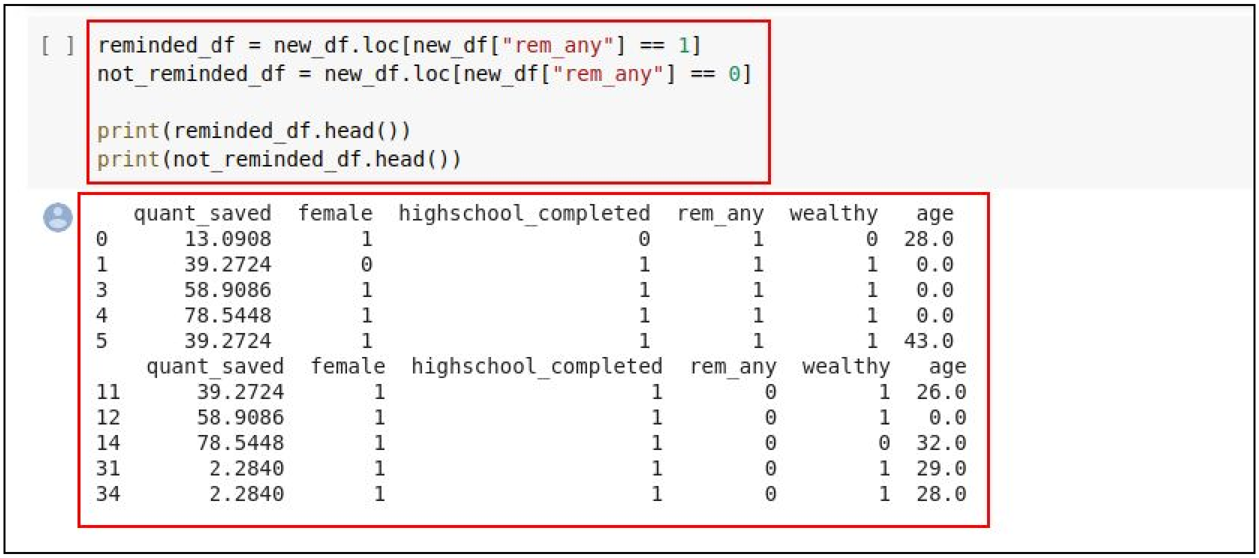
```
Mean of Population - 46.205193898l8794
Mean of Sampling Distribution - 45.99224013709402
```

7. We found that the correlation between age and saving was significant.



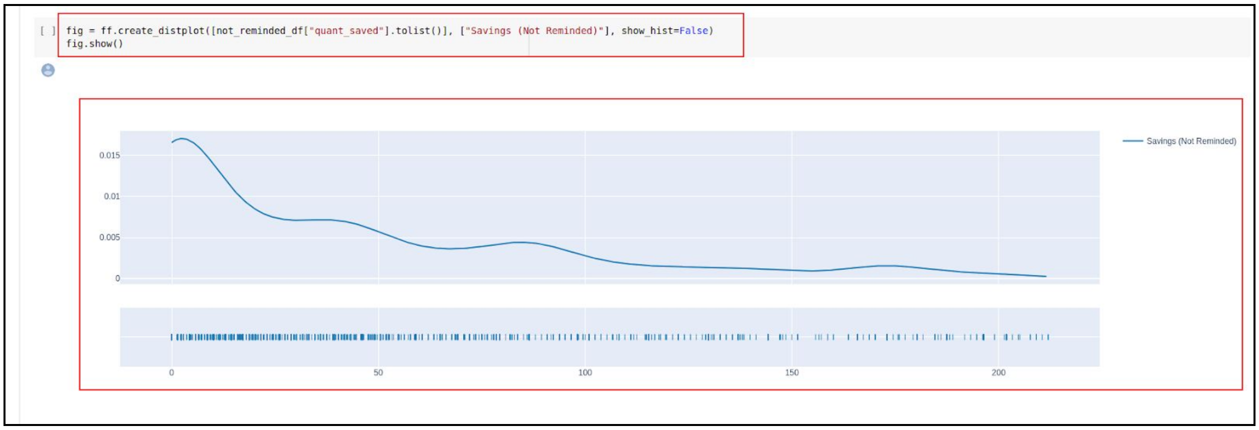
```
[ ] #temp_df will have the rows where age is not 0
    temp_df = new_df[new_df.age != 0]

    age = temp_df["age"].tolist()
    savings = temp_df["quant_saved"].tolist()

    correlation = np.corrcoef(age, savings)
    print(f"Correlation between the age of the person and their savings is - {correlation[0,1]}")

    Correlation between the age of the person and their savings is - 0.08561544120342093
```

8. We then calculated the new data for people who were given reminders and people who weren't.



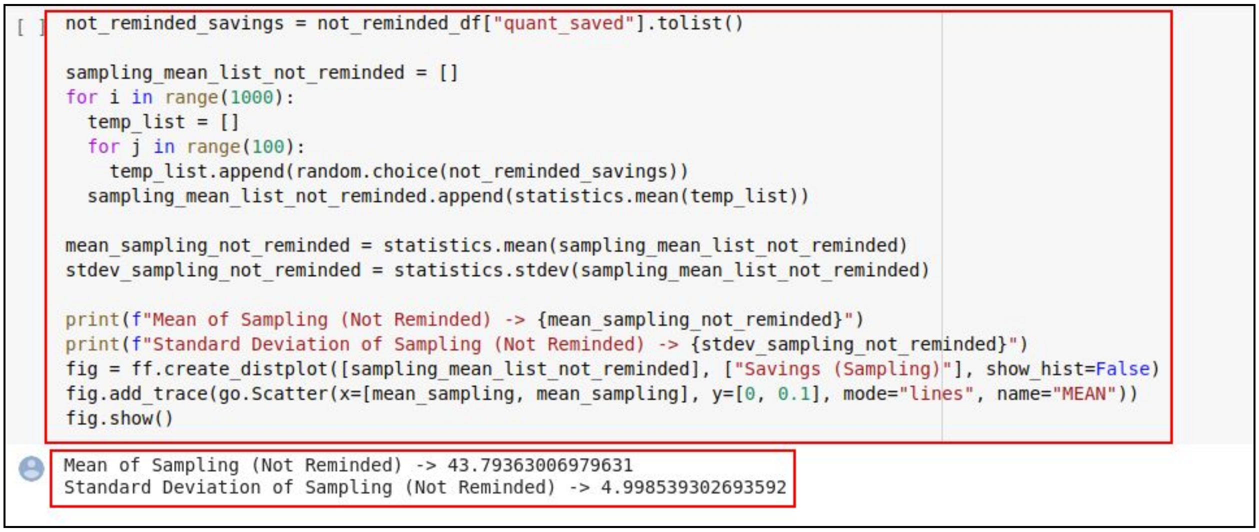
```
[ ] reminded_df = new_df.loc[new_df["rem_any"] == 1]
    not_reminded_df = new_df.loc[new_df["rem_any"] == 0]

    print(reminded_df.head())
    print(not_reminded_df.head())

   quant_saved  female  highschool_completed  rem_any  wealthy   age
0      13.0908       1                     0        1        0  28.0
1      39.2724       0                     1        1        1   0.0
3      58.9086       1                     1        1        1   0.0
4      78.5448       1                     1        1        1   0.0
5      39.2724       1                     1        1        1  43.0
    quant_saved  female  highschool_completed  rem_any  wealthy   age
11      39.2724       1                     1        0        1  26.0
12      58.9086       1                     1        0        1   0.0
14      78.5448       1                     1        0        0  32.0
31       2.2840       1                     1        0        1  29.0
34       2.2840       1                     1        0        1  28.0
```

9. We then plotted the mean of people who weren’t informed to save .



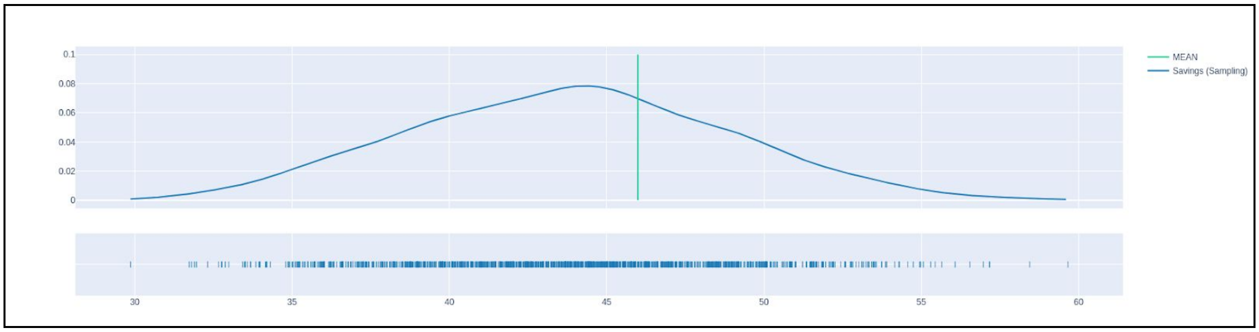
```
[ ] fig = ff.create_distplot([not_reminded_df["quant_saved"].tolist()], ["Savings (Not Reminded)"], show_hist=False)
    fig.show()
```

10. We then found the mean and standard deviation of the new sample.

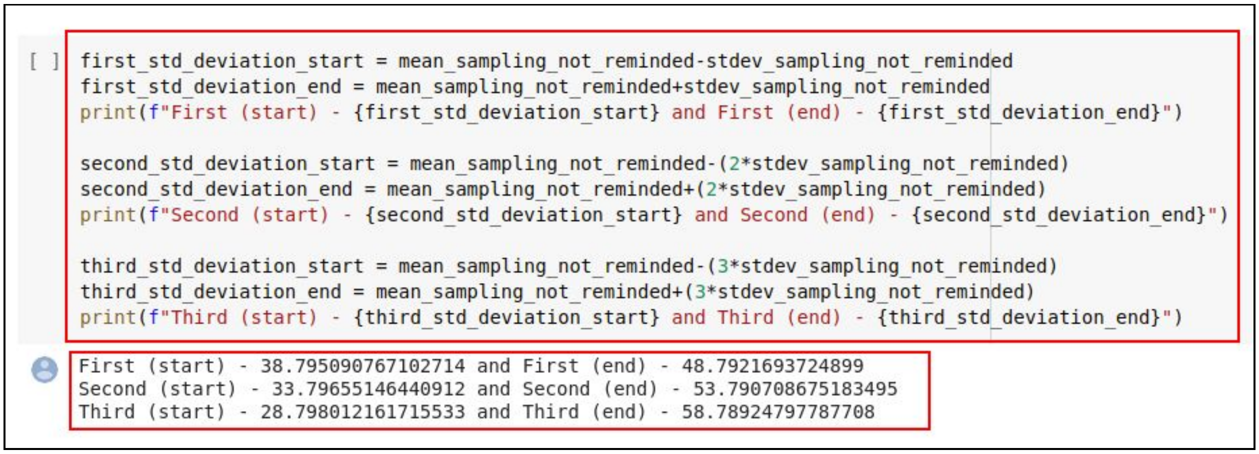


```
[ ] not_reminded_savings = not_reminded_df["quant_saved"].tolist()

    sampling_mean_list_not_reminded = []
    for i in range(1000):
      temp_list = []
      for j in range(100):
        temp_list.append(random.choice(not_reminded_savings))
      sampling_mean_list_not_reminded.append(statistics.mean(temp_list))

    mean_sampling_not_reminded = statistics.mean(sampling_mean_list_not_reminded)
    stdev_sampling_not_reminded = statistics.stdev(sampling_mean_list_not_reminded)

    print(f"Mean of Sampling (Not Reminded) -> {mean_sampling_not_reminded}")
    print(f"Standard Deviation of Sampling (Not Reminded) -> {stdev_sampling_not_reminded}")
    fig = ff.create_distplot([sampling_mean_list_not_reminded], ["Savings (Sampling)"], show_hist=False)
    fig.add_trace(go.Scatter(x=[mean_sampling, mean_sampling], y=[0, 0.1], mode="lines", name="MEAN"))
    fig.show()
```

```
Mean of Sampling (Not Reminded) -> 43.79363006979631
Standard Deviation of Sampling (Not Reminded) -> 4.998539302693592
```



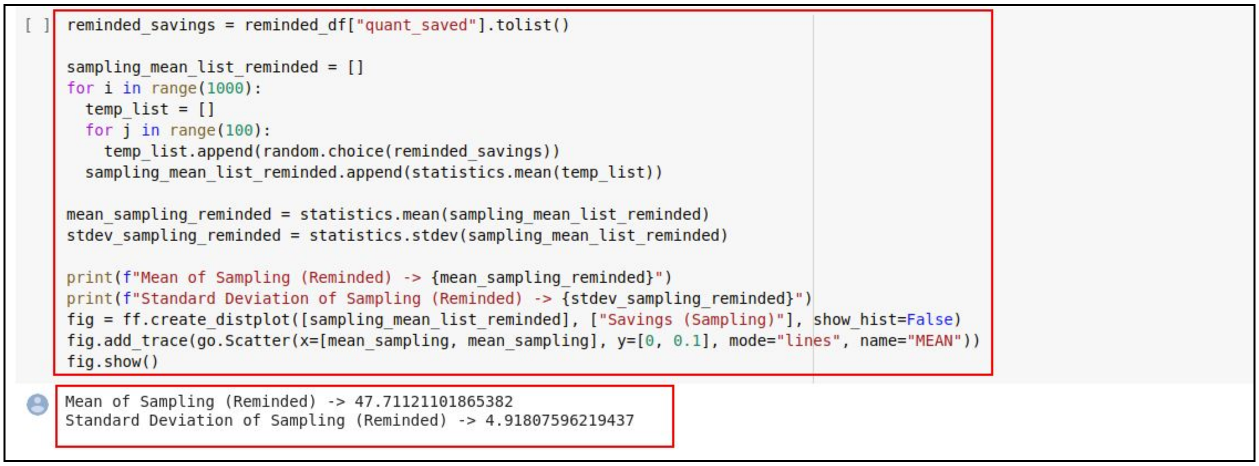
11. We then calculated the mean , median and mode of sampling data.



```
[ ] first_std_deviation_start = mean_sampling_not_reminded-stdev_sampling_not_reminded
    first_std_deviation_end = mean_sampling_not_reminded+stdev_sampling_not_reminded
    print(f"First (start) - {first_std_deviation_start} and First (end) - {first_std_deviation_end}")

    second_std_deviation_start = mean_sampling_not_reminded-(2*stdev_sampling_not_reminded)
    second_std_deviation_end = mean_sampling_not_reminded+(2*stdev_sampling_not_reminded)
    print(f"Second (start) - {second_std_deviation_start} and Second (end) - {second_std_deviation_end}")

    third_std_deviation_start = mean_sampling_not_reminded-(3*stdev_sampling_not_reminded)
    third_std_deviation_end = mean_sampling_not_reminded+(3*stdev_sampling_not_reminded)
    print(f"Third (start) - {third_std_deviation_start} and Third (end) - {third_std_deviation_end}")
```

```
First (start) - 38.795090767102714 and First (end) - 48.7921693724899
Second (start) - 33.79655146440912 and Second (end) - 53.790708675183495
Third (start) - 28.798012161715533 and Third (end) - 58.78924797787708
```

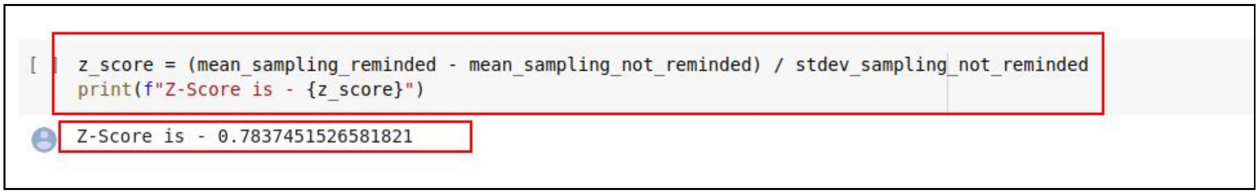
12. We calculated the mean and standard deviation of people who were reminded data.



```
[ ] reminded_savings = reminded_df["quant_saved"].tolist()

    sampling_mean_list_reminded = []
    for i in range(1000):
      temp_list = []
      for j in range(100):
        temp_list.append(random.choice(reminded_savings))
      sampling_mean_list_reminded.append(statistics.mean(temp_list))

    mean_sampling_reminded = statistics.mean(sampling_mean_list_reminded)
    stdev_sampling_reminded = statistics.stdev(sampling_mean_list_reminded)

    print(f"Mean of Sampling (Reminded) -> {mean_sampling_reminded}")
    print(f"Standard Deviation of Sampling (Reminded) -> {stdev_sampling_reminded}")
    fig = ff.create_distplot([sampling_mean_list_reminded], ["Savings (Sampling)"], show_hist=False)
    fig.add_trace(go.Scatter(x=[mean_sampling, mean_sampling], y=[0, 0.1], mode="lines", name="MEAN"))
    fig.show()
```

```
Mean of Sampling (Reminded) -> 47.71121101865382
Standard Deviation of Sampling (Reminded) -> 4.91807596219437
```

13. Finally calculated the z score.



```
[ ] z_score = (mean_sampling_reminded - mean_sampling_not_reminded) / stdev_sampling_not_reminded
    print(f"Z-Score is - {z_score}")
```

```
Z-Score is - 0.7837451526581821
```

We concluded that there was no significant difference between the money saved by the people who were reminded to save and people who weren't reminded to save

What's NEXT?

In the next class, we will learn about machine learning.

EXTEND YOUR KNOWLEDGE:

From the following link you can read about how data storytelling is a skill and why it is important:
https://www.nugit.co/what-is-data-storytelling/