

Topic	Data Story-2	
Class Description	Students learn to remove the outliers from the data using the IQR method and reach a conclusion at the end.	
Class	C113	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> Remove the outliers using the IQR method. Calculate the z score. Complete the data story. 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources <ul style="list-style-type: none"> Google Colaboratory (Colab) Laptop with internet connectivity Earphones with mic Notebook and pen Student Resources <ul style="list-style-type: none"> Google Colaboratory (Colab) Laptop with internet connectivity Earphones with mic Notebook and pen 	
Class structure	Warm Up Teacher-led Activity Student-led Activity Wrap up	5 mins 15 min 15 min 5 min
CONTEXT <ul style="list-style-type: none"> Review the concepts learned in the earlier classes 		
Class Steps	Teacher Action	Student Action
Step 1: Warm Up (5 mins)	Hi <Student Name> What all did we learn in our previous classes?	ESR: -We took the data of people who were reminded to save and who weren't.

		<p>-We calculated the mean, median and mode of the people who were reminded to save and who weren't reminded to save and we saw that the difference was massive.</p> <p>- We also calculated the correlation between the age and the savings and we saw that the data was not correlated.</p>
	<p>Yes...we got to know that the data was scattered and had many outliers so our output was not proper.</p> <p>In today's class we'll learn how to calculate the spread of the data, remove the outliers and then find the z score of the data.</p> <p>Are you up for it?</p>	<p>ESR: varied</p>
	Let's get started then.	-
Teacher Initiates Screen Share		
<p align="center"><u>CHALLENGE</u></p> <ul style="list-style-type: none"> • Learn about the Inter Quartile Range (IQR) • Calculate the IQR and remove the outliers. 		
Step 2: Teacher-led Activity (15 min)	Can you tell me what you can understand from the word Interquartile Range?	<p>ESR: varied</p>

	<p>So Interquartile Range or IQR is used to measure how spread out the data points in a set are from the mean of the data set.</p> <p>The higher the IQR, the more spread out the data points; in contrast, the smaller the IQR, the more bunched up the data points are around the mean.</p>	
	<p>To calculate the IQR there are 4 simple steps.</p> <ol style="list-style-type: none"> 1. Order the data from least to greatest. 2. Find the mean. 3. Calculate the median of both the lower and upper half of the data. 4. The IQR is the difference between the upper and lower medians. <p>Pandas has a quantile function which does all this for us.</p> <p>Pandas dataframe.quantile() this function returns the data frame where the index is q, columns of self (It will take the names of the columns as they are) and values will be the quantiles.</p> <p>Let's see an example to understand better.</p> <p><i><Teacher opens the colab notebook from Teacher activity 1 and writes the</i></p>	

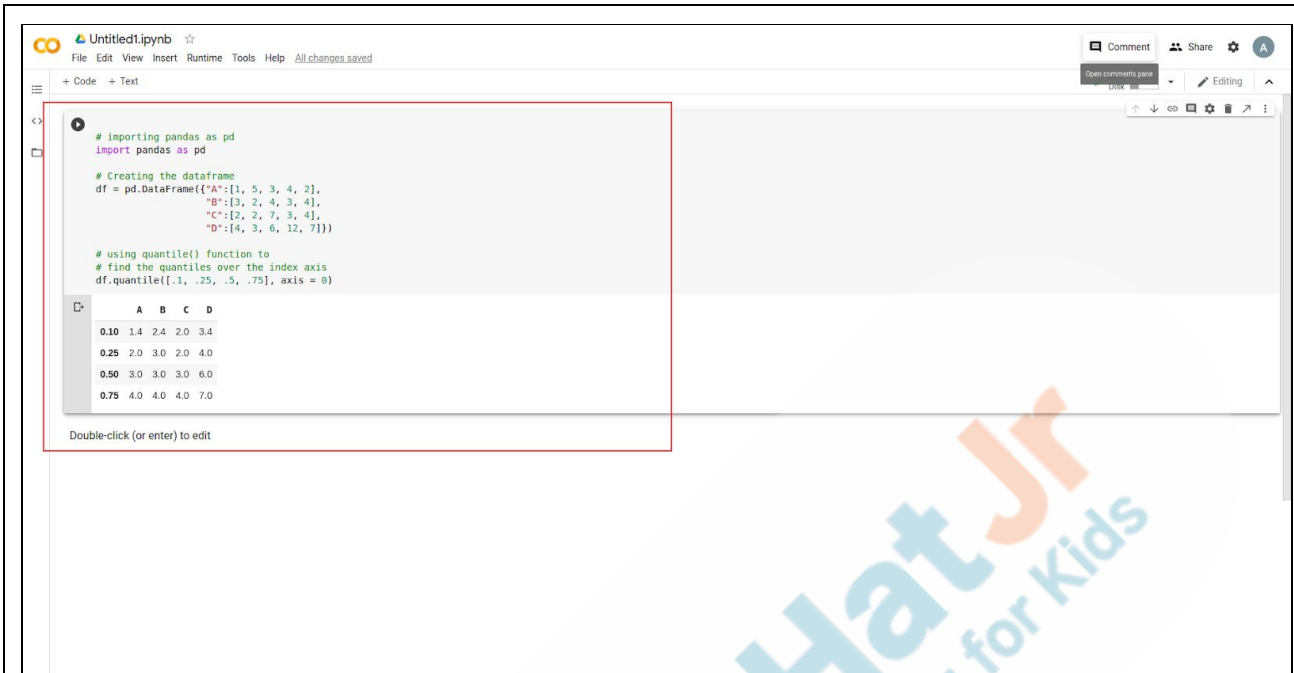
following code in the code cell and runs it>

Code:-

```
# 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)
```

As we see here we created a sample dataframe.

Using the **quantile** function we get the new data consisting of the quantiles of .1, .25, .5, .75 on the given values at axis 0.



```
# 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

Alright now let's find the IQR of our existing data and see how spread our data is.
Can you try to do that?

ESR:
Yes

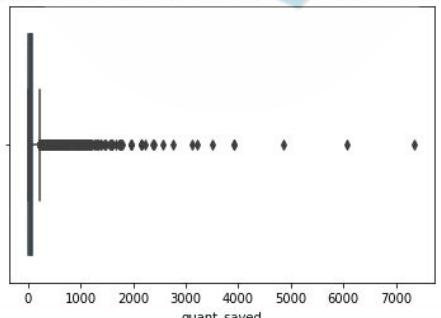
Teacher Stops Screen Share

Now it's your turn. Please share your screen with me.

- Ask Student to press ESC key to come back to panel
- Guide Student to start Screen Share
- Teacher gets into Fullscreen

ACTIVITY

- Find the IQR of the given data set.
- Remove the outliers and find the z score of the data set.

Step 3: Student-Led Activity (15 min)	<p><i>Teacher helps the student to open a new colab notebook from previous class.</i></p>	<p><i>Student opens the colab notebook from previous class. (Student Activity 1)</i></p>
	<p>Till now we have seen that in our data the mean differences were significant and there was no correlation between the age and the savings.</p> <p>To deal with this, we can remove the outliers. To remove the outliers we'll use the IQR method.</p> <p>Let's first check the outliers. To check the outliers we'll use the box plot from the seaborn library. We'll first import the library and then plot the quant_saved data on it.</p>	<p><i>Student imports the seaborn library and plots the quant_saved data on it.</i></p>
<div data-bbox="162 1270 1412 1827"> <pre>[] import seaborn as sns sns.boxplot(data=df, x=df["quant_saved"])</pre> <p>/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.</p> <p><matplotlib.axes._subplots.AxesSubplot at 0x7fe9b6304588></p>  </div>		

	<p>Here, we can see that the majority of the data is represented by a thick, long black line. We then have a thin long black line and finally we have a horizontal line, which is the rest of the data.</p> <p>Here, the thin black vertical line is the line that separates the data we should use to do our analysis. Let's remove the outliers first.</p> <p>Code:-</p> <pre> 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] </pre>	<p><i>Student codes to find the outliers.</i></p> <p><i>Student codes to find the quantile of the quant_saved data.</i></p> <p><i>Student codes to find the quantile of the 3rd quantile and then finds iqr by subtracting q3-q1 print these values.</i></p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```
[ ] 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 - 84.229999999999999
Lower Whisker - -124.06099999999998
Upper Whisker - 212.85899999999998
```

From here, we can say that anyone who saved roughly more than 213 is an outlier in this data.

Now that we've dealt with the outliers, let's see what our mean, median and modes are and also, if we have a normal distribution now. Let's also check the standard deviation now.

Teacher helps the student find the standard deviation.

Code:-

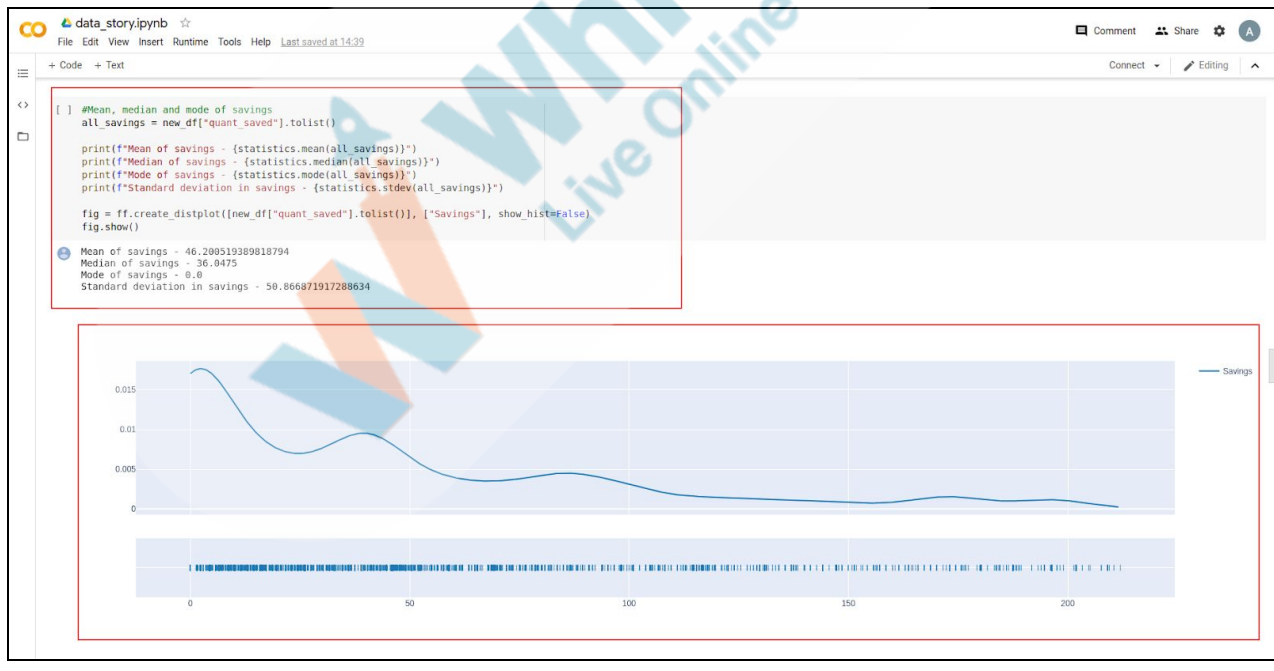
#Mean, median and mode of savings

Student codes to find the mean, median, mode and standard deviation of the quant_data. and plot it on the graph.


```
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_s
aved"].tolist()], ["Savings"],
show_hist=False)
fig.show()
```



	<p>Here, now our mean is much closer to the median than before, and the mode still remains to be 0 since there are people in our dataset who haven't saved at all!</p> <p>The standard deviation has also significantly reduced from before, but we still do not have a normal distribution curve.</p> <p>To make this a normal distribution curve now, we can simply draw random samples from the population to create a sampling (normal distribution).</p> <p>You already know how to get the 100 data points, find its mean and plot it on the graph.</p> <p>Code:-</p> <pre>#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(al l_savings))</pre>	<p><i>Student codes to find the mean of random 100 data points and plot it on the graph.</i></p>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------

```
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()
```



	<p>What do we see? Perfect now let's find the standard deviation of the sampling distribution.</p> <p>Code:- print(f"Standard deviation of the sampling data - {statistics.stdev(sampling_mean_list)})")</p>	<p>ESR:We see the normal distribution bell curve.</p> <p><i>Student codes to find the standard deviation of the sampling distribution.</i></p>
<pre>[] print(f"Standard deviation of the sampling data - {statistics.stdev(sampling_mean_list)})")</pre> <pre>Standard deviation of the sampling data - 5.091787150333248</pre>		
	<p>What do we find here? Now we'll compare the mean of the sampling and the population.</p> <p>Code:- print(f"Mean of Population - {statistics.mean(all_savings)})") print(f"Mean of Sampling Distribution - {mean_sampling}")</p>	<p>ESR: The sampling standard deviation comes out to be roughly 1/10th of the standard deviation of the population.</p> <p><i>Student codes to find the mean of the population and the sampling.</i></p>
<pre>[] print(f"Mean of Population - {statistics.mean(all_savings)})") print(f"Mean of Sampling Distribution - {mean_sampling}")</pre> <pre>Mean of Population - 46.200519389818794 Mean of Sampling Distribution - 45.99224013709402</pre>		

	<p>What do we see here?</p> <p>Let's do one final check, by trying to find the correlation between the age and the savings with the new data from which we have removed the outliers!</p>	<p>ESR:We see that they come out exactly the same.</p>
	<p>We'll find the correlation between the age and the savings data from which the outliers were removed.</p> <p>Code:- #temp_df will have the rows where age is not 0 temp_df = new_df[new_df.age != 0]</p> <p>age = temp_df["age"].tolist() savings = temp_df["quant_saved"].tolist()</p> <p>correlation = np.corrcoef(age, savings) print(f"Correlation between the age of the person and their savings is - {correlation[0,1]}")</p>	<p><i>Student codes to find the new correlation between the age and the savings data from which the outliers were removed.</i></p>

```
[
#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

	<p>What do we get?</p> <p>Now, let's get back to the real question. Does reminder have any impact on the savings of the people? How can we achieve this statistically?</p> <p>With the z-test! We will take a sample from the people who were given reminders and see if that sample fits with the population of people who were not given reminders. If it fits in well, we can say that the reminders had no significant effect.</p> <p>Let's start with first filtering out the data of people who were given notifications and those who weren't given notifications. To filter the data we just have to check if the value under that person is 0 or 1.</p> <p>Code:-</p> <pre>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())</pre>	<p>ESR:</p> <p>We get a correlation of 0.085, which is not significant.</p> <p><i>Student codes to filter the data of people who were given notifications and those who weren't given notifications.</i></p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```
[ ] 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())
```

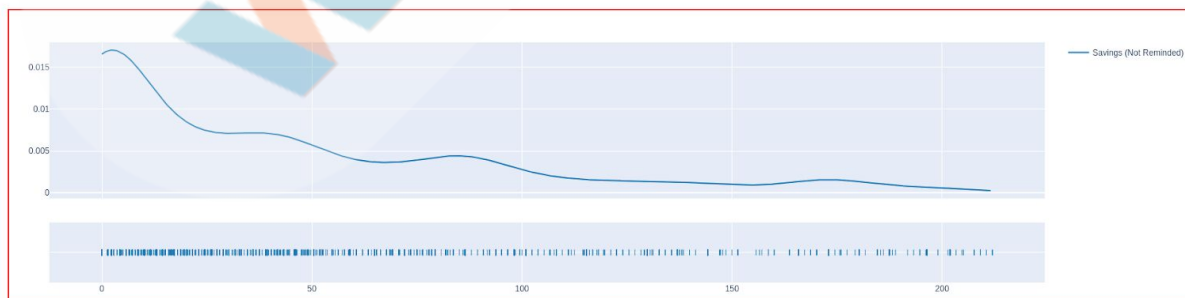
```
quant_saved  female  highschool_completed  rem_any  wealthy  age
0      13.0908      1              0          1          0    28.0
1      39.2724      0              1          1          1     0.0
3      58.9086      1              1          1          1     0.0
4      78.5448      1              1          1          1     0.0
5      39.2724      1              1          1          1    43.0
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
```

Now, let's plot the distribution graph for the people who were not reminded to save and see if it follows a normal distribution.

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

Student codes to plot the not reminded to save the money.

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



	<p>What can we observe? Correct, so we will take sampling data and plot it and also use it for z test.</p> <p>We need to find the mean and standard deviation of the sampling data. Code:-</p> <pre> 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}") </pre>	<p>ESR: This doesn't follow normal distribution.</p> <p><i>Student codes to find the mean and standard deviation of the sampling.</i></p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------


```
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()
```

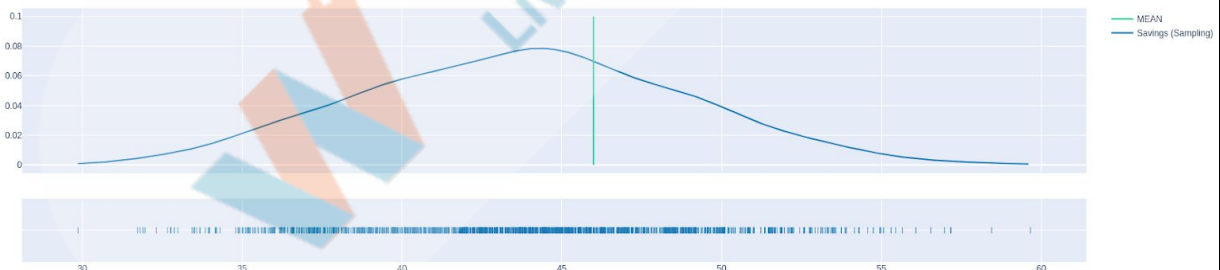
```
[ ] not_reminded_savings = not_reminded_df["quant_saved"].tolist()

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
```



Now we have a resemblance of a normal distribution, and we also have the mean and the standard deviation of the sampling data for the people who were not notified to save.

Student codes to find the first, second and the third deviation.

Now, assuming that we started notifying people to see if there was an effect in people's saving, let's see how it turned out!

First, let's calculate the first, second and third standard deviations for our sampling distribution.

Teacher helps the student with the code.

Code:-

```
first_std_deviation_start =
mean_sampling_not_reminded-std
ev_sampling_not_reminded
first_std_deviation_end =
mean_sampling_not_reminded+std
ev_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*s
tdev_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}")
```

	<pre> third_std_deviation_start = mean_sampling_not_reminded-(3*s tdev_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}") </pre>	
	<pre> [] 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}") </pre> <pre> 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 </pre>	
	<p>Now that we have everything we need for our z-test, let's quickly take a sampling distribution of the people who were reminded for saving.</p> <p><i>Teacher helps the student with the code.</i></p> <p>Code:-</p> <pre> reminded_savings = reminded_df["quant_saved"].tolist() sampling_mean_list_reminded = [] for i in range(1000): temp_list = [] </pre>	

	<pre> for j in range(100): temp_list.append(random.choice(r eminded_savings)) sampling_mean_list_reminded.app end(statistics.mean(temp_list)) mean_sampling_reminded = statistics.mean(sampling_mean_lis t_reminded) stdev_sampling_reminded = statistics.stdev(sampling_mean_lis t_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() </pre>	
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

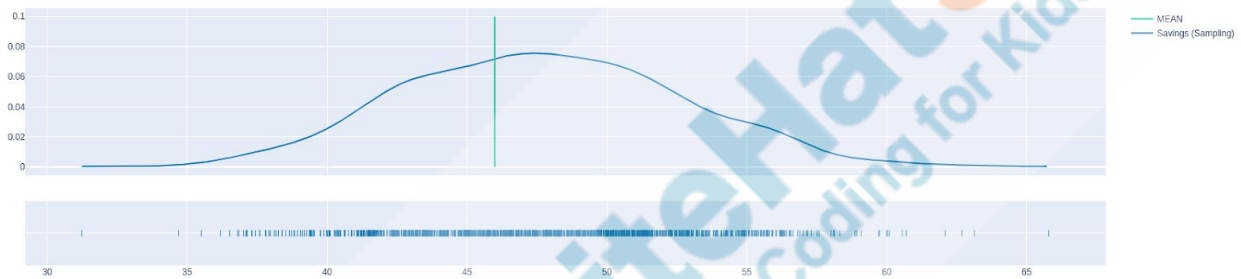
```
[ ] reminded_savings = reminded_df["quant_saved"].tolist()

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)
stddev_sampling_reminded = statistics.stdev(sampling_mean_list_reminded)

print(f"Mean of Sampling (Reminded) -> {mean_sampling_reminded}")
print(f"Standard Deviation of Sampling (Reminded) -> {stddev_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



Now we have the mean of sampling distribution of people who were given reminders and people who were not given reminders. We also have the standard deviation of the people who were not given reminders.

We find the z score to see how significantly reminders impacted savings.

Code:-

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

Student codes to find the z score of the data.

```
[ ] 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
```

From these tests what can we conclude?

ESR:

Given all the analysis that we have done, we saw that there was not much difference between the people who got reminded about saving v/s people who were not reminded of saving.

With our Z-Score test, we confirmed that there is not much difference. The Z-Score is around 0.8; it is only when the Z-Score is greater than 2 that we consider the change as significant. Although people who were reminded had a little increase in savings, there's still no significant impact.

We also saw that there was no direct correlation between the age of the people and the money that they save.

Teacher Guides Student to Stop Screen Share

FEEDBACK

- Appreciate the student for their efforts
- Identify 2 strengths and 1 area of progress for the student

Step 4: Wrap-Up (5 min)	So, in this data story class we reviewed the concepts we have learned so far. How was your experience?	ESR: varied
	Amazing. While working on this data story, we also made sure that we are at the top of all the concepts we have acquired so far. Next class, we will be learning new concepts and building new projects.	-
Project Overview	Data Story -2 Goal of the Project: In this project we will complete writing the data story of a given data set. Story: In our journey of analyzing the article's data, you also want to understand how the results are	

	<p>changing after the introduction of an intervention.</p> <p>write a program to do the z test of a given sample.</p> <p>.</p> <p>I am very excited to see your project solution and I know you will do really well.</p> <p>Bye Bye!</p>	
<div> <div>Teacher Clicks</div> <div>✕ End Class</div> </div>		
Additional Activities	<p><i>Encourage the student to write reflection notes in their reflection journal using markdown.</i></p> <p>Use these as guiding questions:</p> <ul style="list-style-type: none"> • What happened today? <ul style="list-style-type: none"> - Describe what happened - Code I wrote • How did I feel after the class? • What have I learned about programming and developing games? • What aspects of the class helped me? What did I find difficult? 	<p><i>The student uses the markdown editor to write her/his reflection in a reflection journal.</i></p>

Activity	Activity Name	Links
Teacher Activity 1	Colab notebook link	https://colab.research.google.com/n

© 2020 - WhiteHat Education Technology Private Limited.

Note: This document is the original copyright of WhiteHat Education Technology Private Limited.

Please don't share, download or copy this file without permission.

		notebooks/intro.ipynb#recent=true
Teacher Activity 2	Colab Reference (final code)	https://colab.research.google.com/drive/1IWhvt25NMx2VYMw4ILTuRzBvTL57Lt52?usp=sharing
Student Activity 1	Colab notebook link	https://colab.research.google.com/notebooks/intro.ipynb#recent=true
Student Activity 2	savings data file	https://raw.githubusercontent.com/whitehatjr/datasets/master/savings_data_final.csv