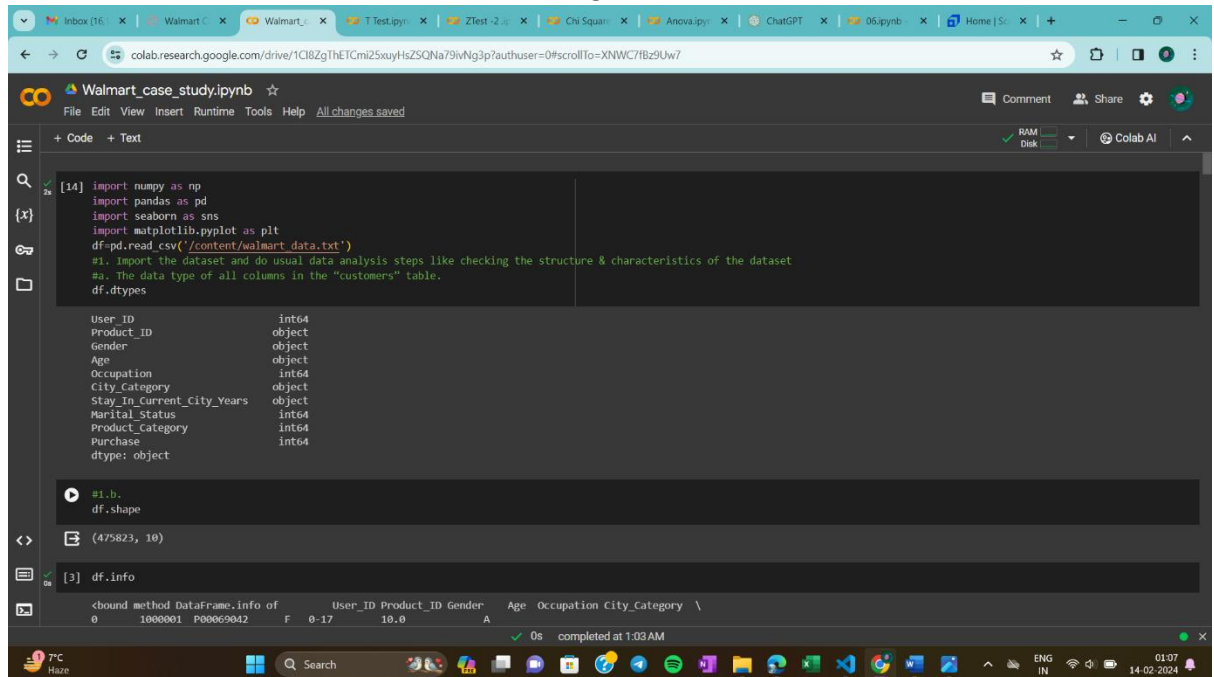


Walmart Business case study

1. Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset.
 - a. The data type of all columns in the “customers” table.
 - b. You can find the number of rows and columns given in the dataset



```
[14] import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('/content/walmart_data.txt')
#1. Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset
#1. The data type of all columns in the “customers” table.
df.dtypes

User_ID          int64
Product_ID       object
Gender           object
Age             object
Occupation       int64
City_Category    object
Stay_In_Current_City_Years  int64
Marital_Status   int64
Product_Category int64
Purchase         int64
dtype: object

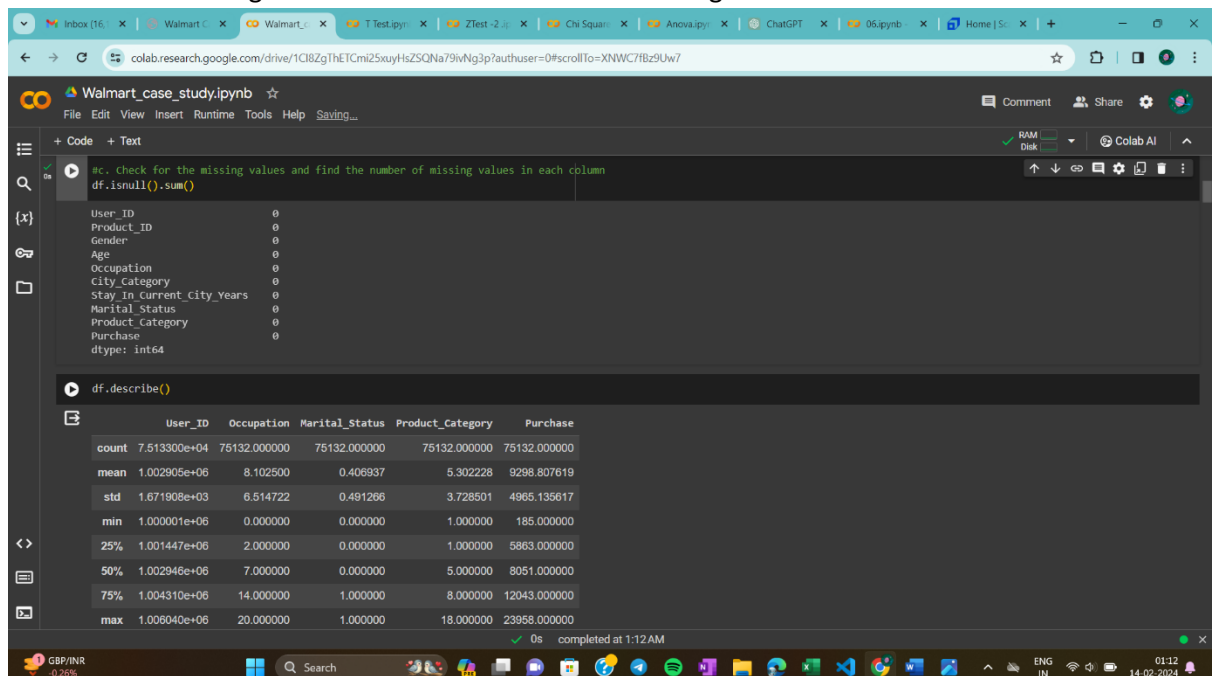
#1.b.
df.shape

(475823, 10)

[3] df.info

<bound method DataFrame.info of
0      1000001  P00069042  F  0-17  10.0  A  Age  Occupation City_Category \
completed at 1:03 AM
```

- c. Check for the missing values and find the number of missing values in each column.



```
#c. Check for the missing values and find the number of missing values in each column
df.isnull().sum()

User_ID          0
Product_ID       0
Gender           0
Age             0
Occupation       0
City_Category    0
Stay_In_Current_City_Years  0
Marital_Status   0
Product_Category 0
Purchase         0
dtype: int64

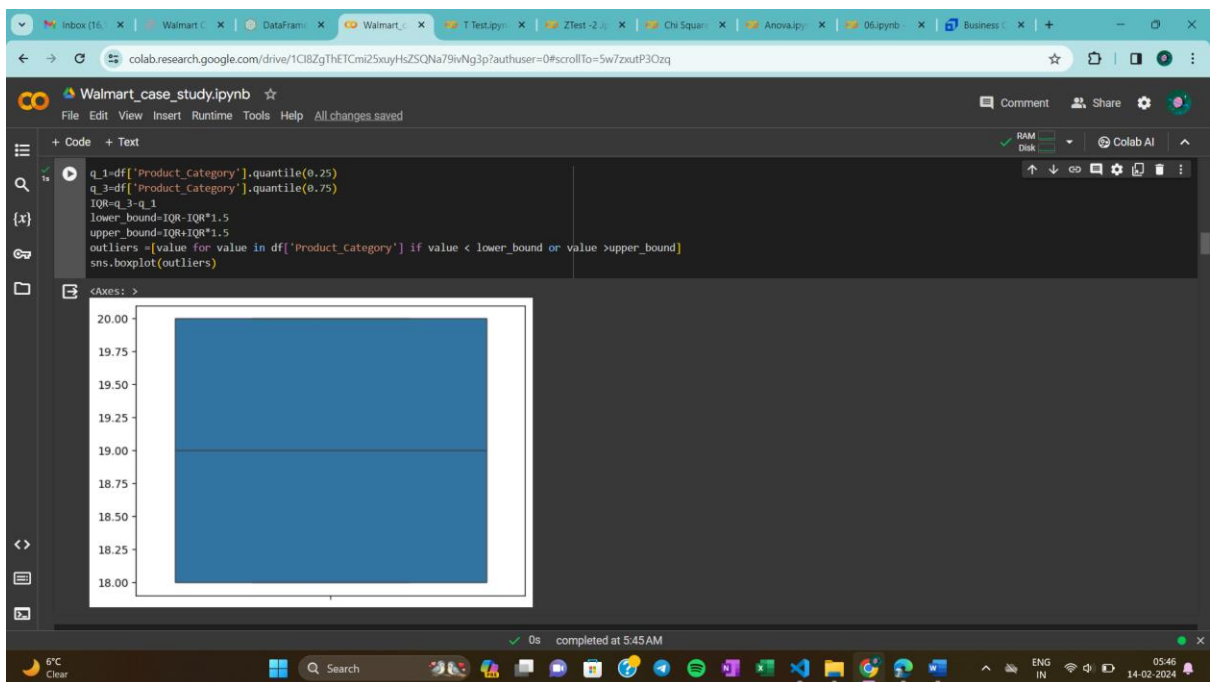
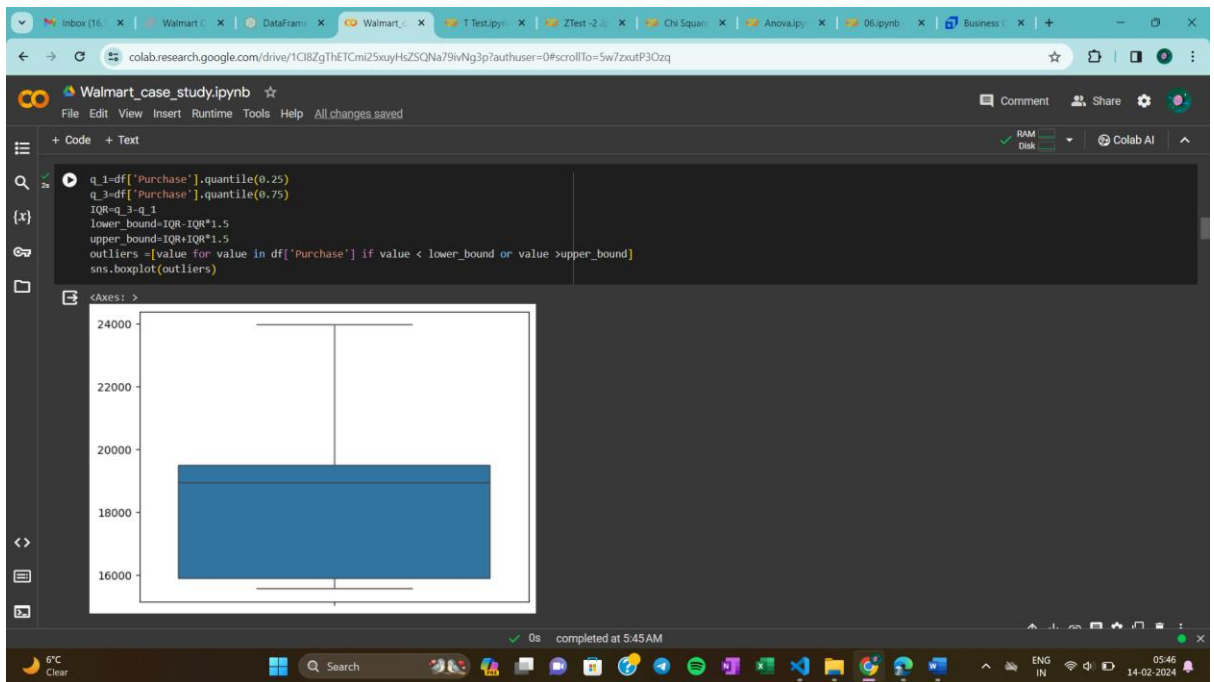
df.describe()

   count  User_ID  Occupation  Marital_Status  Product_Category  Purchase
mean  1.002905e+06  8.102500  0.406937      5.302228  9298.807619
std    1.671908e+03  6.514722  0.491266      3.728501  4965.135617
min    1.000001e+06  0.000000  0.000000      1.000000  185.000000
25%    1.001447e+06  2.000000  0.000000      1.000000  5863.000000
50%    1.002946e+06  7.000000  0.000000      5.000000  8051.000000
75%    1.004310e+06  14.000000  1.000000      8.000000  12043.000000
max    1.008040e+06  20.000000  1.000000     18.000000  23958.000000

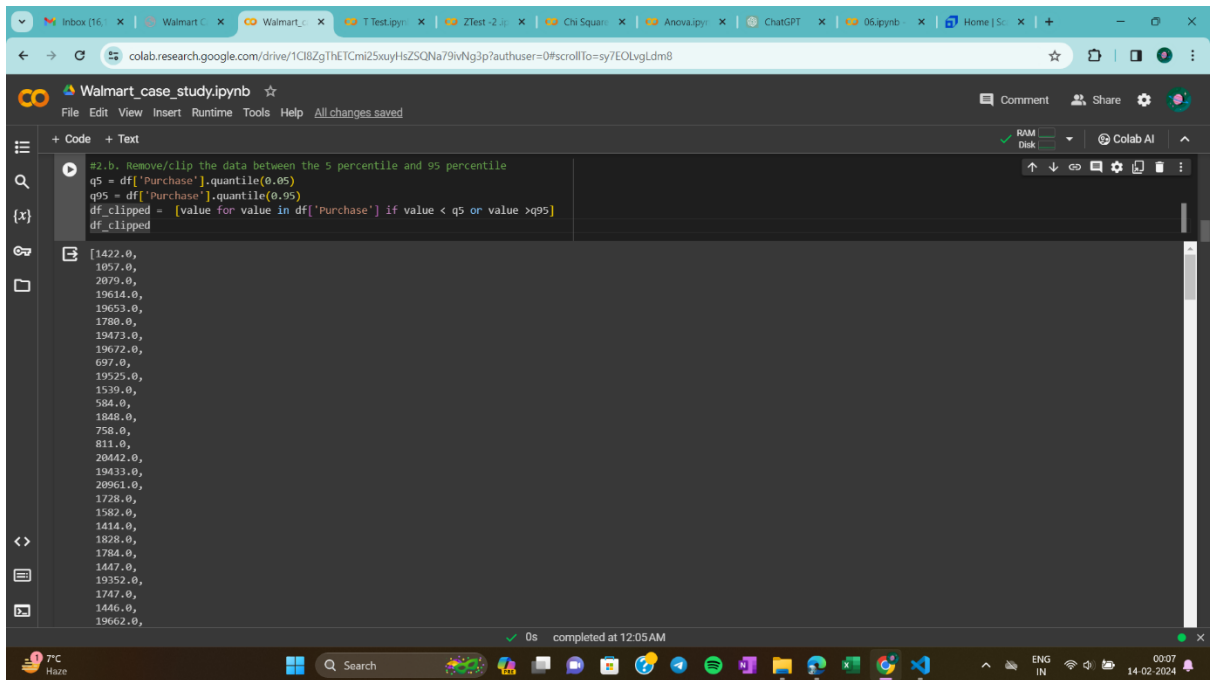
completed at 1:12 AM
```

2. Detect Null values and outliers

- a. Find the outliers for every continuous variable in the dataset.



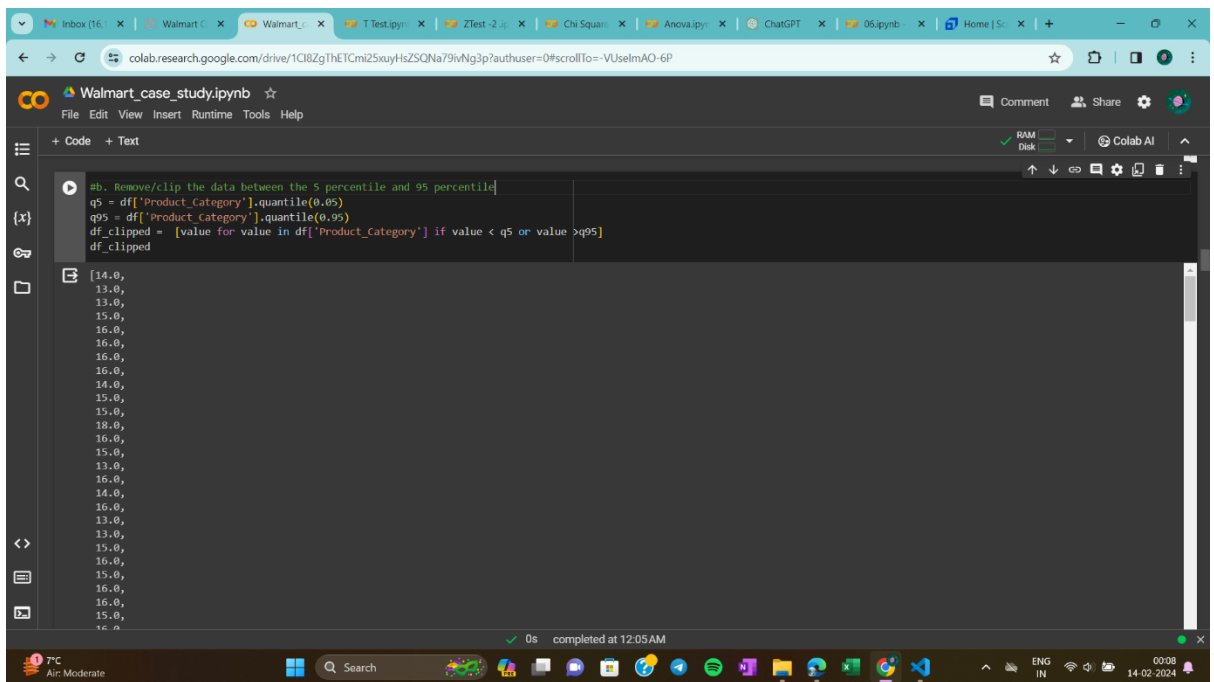
b. Remove/clip the data between the 5 percentile and 95 percentiles.



This screenshot shows a Google Colab notebook titled 'Walmart_case_study.ipynb'. The code cell contains the following Python code:

```
#2.b. Remove/clip the data between the 5 percentile and 95 percentile
q5 = df['Purchase'].quantile(0.05)
q95 = df['Purchase'].quantile(0.95)
df_clipped = [value for value in df['Purchase'] if value < q5 or value > q95]
df_clipped
```

The output of the code is a list of 20 numerical values, representing the 'Purchase' column after clipping. The values are: [1422.0, 1097.0, 2079.0, 19614.0, 19653.0, 1709.0, 19473.0, 19672.0, 697.0, 19525.0, 1539.0, 584.0, 1048.0, 758.0, 811.0, 20442.0, 19433.0, 20961.0, 1728.0, 1582.0, 1414.0, 1828.0, 1784.0, 1447.0, 19352.0, 1747.0, 1446.0, 19662.0].



This screenshot shows the same Google Colab notebook, but the code cell has been updated to clip data based on 'Product_Category' instead of 'Purchase'. The code is:

```
#b. Remove/clip the data between the 5 percentile and 95 percentile
q5 = df['Product_Category'].quantile(0.05)
q95 = df['Product_Category'].quantile(0.95)
df_clipped = [value for value in df['Product_Category'] if value < q5 or value > q95]
df_clipped
```

The output is a list of 20 numerical values representing the 'Product_Category' column after clipping. The values are: [14.0, 13.0, 13.0, 13.0, 15.0, 16.0, 16.0, 16.0, 16.0, 14.0, 15.0, 15.0, 18.0, 16.0, 15.0, 13.0, 16.0, 14.0, 16.0, 13.0, 13.0, 15.0, 16.0, 16.0, 16.0, 15.0, 16.0].

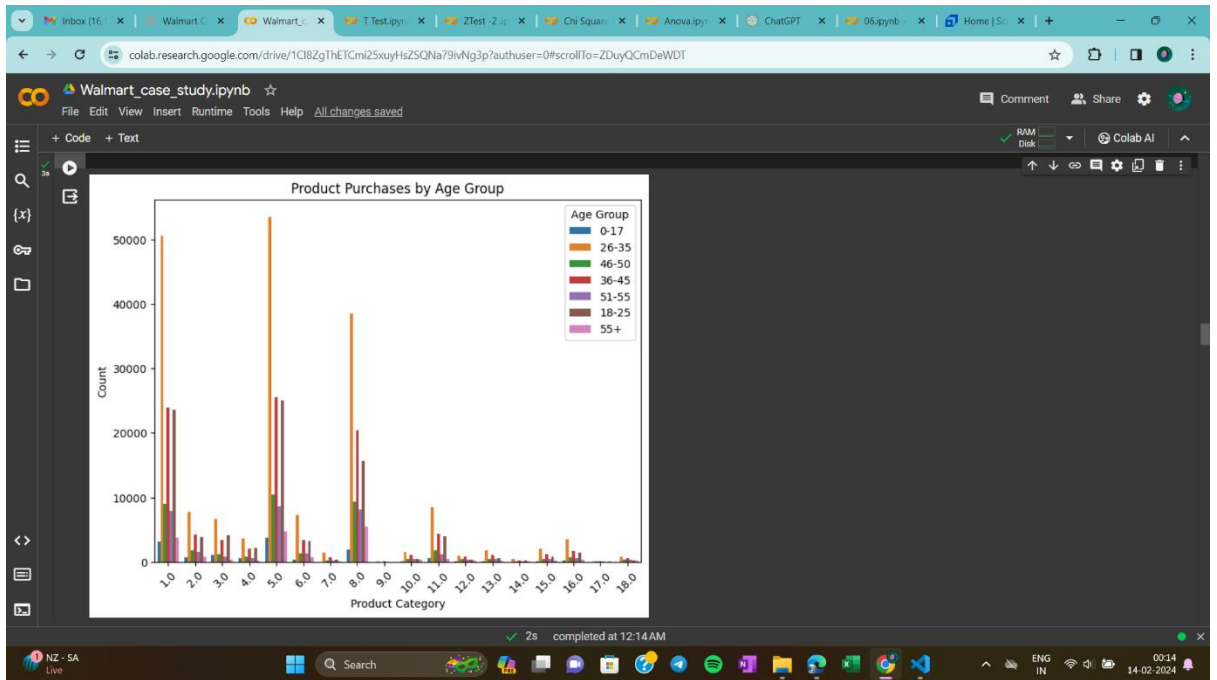
3. Data Exploration

a. What products are different age groups buying?

```
Walmart_case_study.ipynb
File Edit View Insert Runtime Tools Help All changes saved

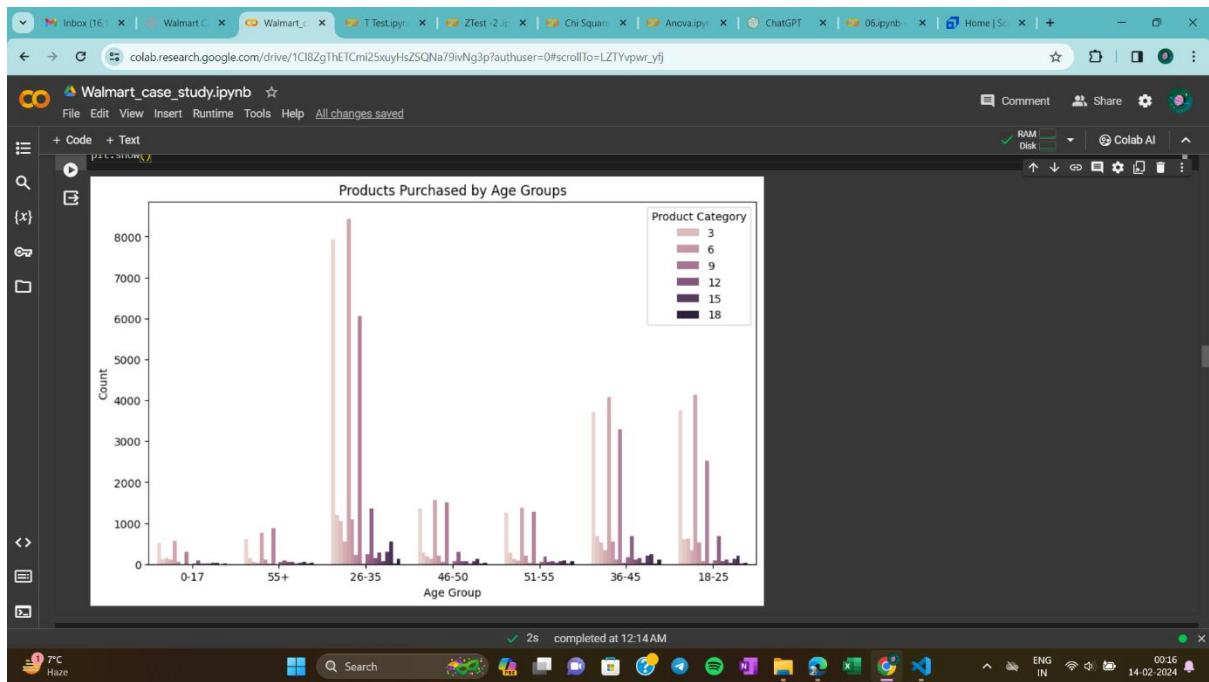
+ Code + Text
#3. Data Exploration
#a. What products are different age groups buying?
import seaborn as sns
import matplotlib.pyplot as plt

# Plot a countplot of product purchases for each age group
plt.figure(figsize=(10, 8))
sns.countplot(data=df, x='Product_Category', hue='Age')
plt.title('Product Purchases by Age Group')
plt.xlabel('Product_Category')
plt.ylabel('Count')
plt.xticks(rotation=45) # Rotate x-axis labels for better readability
plt.legend(title='Age Group')
plt.show()
```



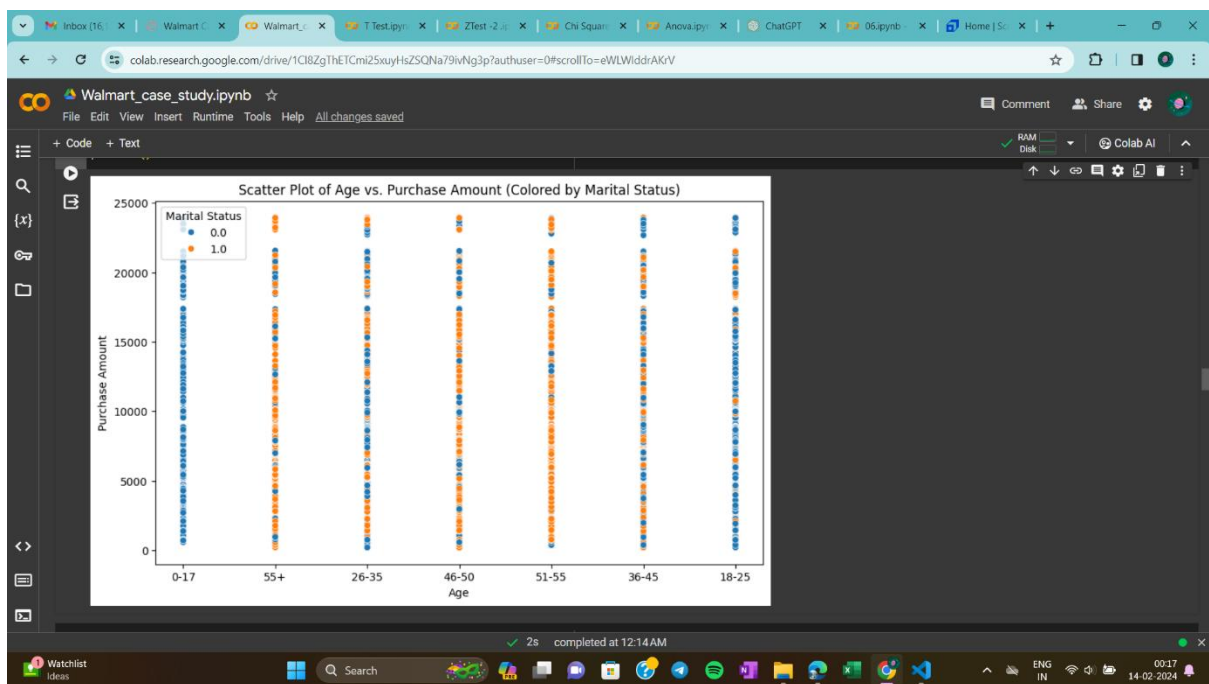
```
Walmart_case_study.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
#3. Data Exploration
#a. What products are different age groups buying?(another way)
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Age', hue='Product_Category')
plt.title('Products Purchased by Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.legend(title='Product Category')
plt.show()
```



b. Is there a relationship between age, marital status, and the amount spent?

```
#1.b. Is there a relationship between age, marital status, and the amount spent?
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Age', y='Purchase', hue='Marital_Status')
plt.title("Scatter Plot of Age vs. Purchase Amount (colored by Marital Status)")
plt.xlabel("Age")
plt.ylabel("Purchase Amount")
plt.legend(title="Marital Status")
plt.show()
```



c. Are there preferred product categories for different genders?

```
Walmart_case_study.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

#3.c. Are there preferred product categories for different genders?
# Separate data by gender
male_data = df[df['Gender'] == 'M']
female_data = df[df['Gender'] == 'F']

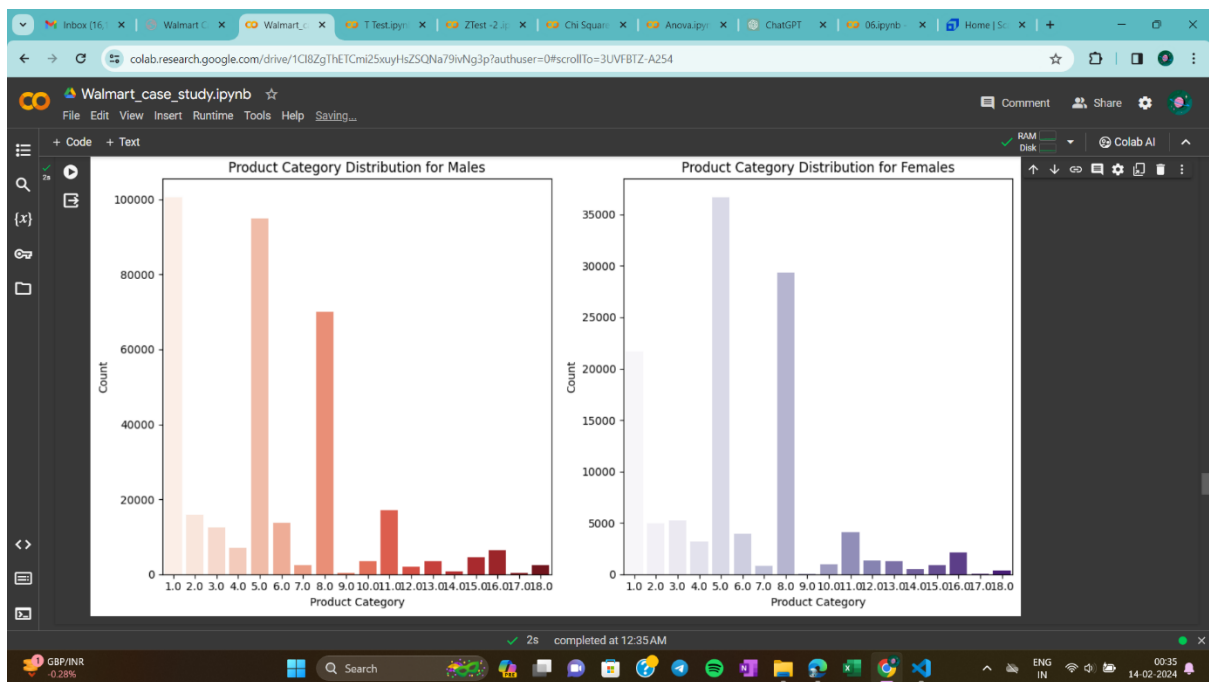
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.countplot(data=male_data, x='Product_Category', palette='Reds')
plt.title('Product Category Distribution for Males')
plt.xlabel('Product Category')
plt.ylabel('count')

plt.subplot(1, 2, 2)
sns.countplot(data=female_data, x='Product_Category', palette='Purples')
plt.title('Product Category Distribution for Females')
plt.xlabel('Product Category')
plt.ylabel('count')

plt.tight_layout()
plt.show()
```

<ipython-input-11-2a83270179d5>:8: FutureWarning:
Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'x' variable to 'hue' and set 'legend=False' for the same effect.
sns.countplot(data=male_data, x='Product_Category', palette='Reds')
<ipython-input-11-2a83270179d5>:14: FutureWarning:
Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'x' variable to 'hue' and set 'legend=False' for the same effect.
sns.countplot(data=female_data, x='Product_Category', palette='Purples')

2s completed at 12:35 AM



4. How does gender affect the amount spent?


```
Walmart_case_study.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
#4. How does gender affect the amount spent?
mean_purchase_by_gender = df.groupby('Gender')['Purchase'].mean()
std_purchase_by_gender = df.groupby('Gender')['Purchase'].std()
# Calculate confidence interval for the entire dataset
# Calculate confidence interval for each gender separately
confidence_intervals = {}
for gender in df['Gender'].unique():
    gender_data = df[df['Gender'] == gender]['Purchase']
    n = gender_data.count()
    mean_purchase = gender_data.mean()
    std_purchase = gender_data.std()
    z_score = stats.norm.ppf(0.975) # 95% confidence interval
    margin_of_error = z_score * (std_purchase / np.sqrt(n))
    confidence_intervals[gender] = (mean_purchase - margin_of_error, mean_purchase + margin_of_error)

# Print confidence intervals
for gender, interval in confidence_intervals.items():
    print(f'{gender}: {interval}')

# Bootstrap function to calculate confidence intervals
def bootstrap_confidence_interval(data, num_samples, alpha=0.05):
    bootstrapped_means = []
    for _ in range(num_samples):
        bootstrap_sample = np.random.choice(data, size=len(data), replace=True)
        bootstrapped_means.append(np.mean(bootstrap_sample))
    lower_percentile = (alpha / 2) * 100
    upper_percentile = (1 - alpha / 2) * 100
    lower_bound = np.percentile(bootstrapped_means, lower_percentile)
    upper_bound = np.percentile(bootstrapped_means, upper_percentile)
    return lower_bound, upper_bound

# Perform bootstrapping and calculate confidence intervals for different sample sizes
sample_sizes = [300, 3000, 30000]
```

```
Walmart_case_study.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text
# Bootstrap function to calculate confidence intervals
def bootstrap_confidence_interval(data, num_samples, alpha=0.05):
    bootstrapped_means = []
    for _ in range(num_samples):
        bootstrap_sample = np.random.choice(data, size=len(data), replace=True)
        bootstrapped_means.append(np.mean(bootstrap_sample))
    lower_percentile = (alpha / 2) * 100
    upper_percentile = (1 - alpha / 2) * 100
    lower_bound = np.percentile(bootstrapped_means, lower_percentile)
    upper_bound = np.percentile(bootstrapped_means, upper_percentile)
    return lower_bound, upper_bound

# Perform bootstrapping and calculate confidence intervals for different sample sizes
sample_sizes = [300, 3000, 30000]
bootstrapped_intervals = {}
for gender in df['Gender'].unique():
    data = df[df['Gender'] == gender]['Purchase']
    bootstrapped_intervals[gender] = {}
    for sample_size in sample_sizes:
        lower_bound, upper_bound = bootstrap_confidence_interval(data, num_samples=sample_size)
        bootstrapped_intervals[gender][sample_size] = (lower_bound, upper_bound)

print("Confidence Intervals for the Entire Dataset:")
for gender, interval in confidence_intervals.items():
    print(f'{gender}: {interval}')

print("\nConfidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):")
for gender, intervals in bootstrapped_intervals.items():
    print(f'{gender}:')
    for sample_size, interval in intervals.items():
        print(f"Sample Size {sample_size}: {interval}")
```

```
print(f"Confidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):")
for gender, intervals in bootstrapped_intervals.items():
    print(f"Gender: {gender}")
    for sample_size, interval in intervals.items():
        print(f"Sample Size {sample_size}: {interval}")

F: (8611.598235475956, 8848.987146921872)
M: (9327.270601912533, 9469.283972516645)
nan: (nan, nan)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3432: RuntimeWarning: Mean of empty slice.
return _methods._mean(a, axis=axis, dtype=dtype,
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:190: RuntimeWarning: invalid value encountered in double_scalars
ret = ret.dtype.type(ret / rcount)
Confidence Intervals for the Entire Dataset:
F: (8611.598235475956, 8848.987146921872)
M: (9327.270601912533, 9469.283972516645)
nan: (nan, nan)

Confidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):
F:
Sample Size 300: (8617.379192809904, 8853.219370018653)
Sample Size 3000: (8610.06420213668, 8848.664778701035)
Sample Size 30000: (8612.529226725454, 8850.247702221460)
M:
Sample Size 300: (9329.261308062072, 9476.465728355714)
Sample Size 3000: (9328.312211975548, 9468.718837191076)
Sample Size 30000: (9327.895379852658, 9469.15304091123)
nan:
Sample Size 300: (nan, nan)
Sample Size 3000: (nan, nan)
Sample Size 30000: (nan, nan)
```

- a. From the above calculated CLT answer the following questions.
- Is the confidence interval computed using the entire dataset wider for one of the genders? Why is this the case?
Yes, the ci for entire dataset is wider for the male gender compared to the female gender. This is because the std dev of purchase amounts for males is generally higher than that for females, resulting in a wider interval.
 - How is the width of the confidence interval affected by the sample size?
As the sample size increases, the width of ci generally increases .(due to CLT)
 - Do the confidence intervals for different sample sizes overlap?
Yes, the ci for different sample size overlap
 - How does the sample size affect the shape of the distributions of the means?
As the sample size increase the shape of distribution of the means become more normally distributed.

5. How does Marital Status affect the amount spent?

The image displays two screenshots of a Jupyter Notebook titled 'Walmart_case_study.ipynb' running in Google Colab. The notebook is connected to a Python 3 Google Compute Engine backend.

Top Screenshot: Shows the initial code for calculating confidence intervals for different marital statuses. The code includes:

- Grouping data by marital status to calculate mean and standard deviation of purchase amounts.
- Calculating z-scores and margins of error for 95% confidence intervals.
- Printing the resulting confidence intervals for each marital status.
- Defining a bootstrap function to calculate confidence intervals by resampling the data.

Bottom Screenshot: Shows the continuation of the code, which:

- Performs bootstrapping for different sample sizes (300, 3000, 30000).
- Calculates confidence intervals for the entire dataset and for the bootstrapped samples.
- Prints the results, showing confidence intervals for each marital status and for the bootstrapped samples at different sample sizes.

```
#5. How does Marital Status affect the amount spent?
mean_purchase_by_Marital_Status = df.groupby('Marital_Status')['Purchase'].mean()
std_purchase_by_Marital_Status = df.groupby('Marital_Status')['Purchase'].std()

confidence_intervals = {}
for Marital_Status in df['Marital_Status'].unique():
    Marital_Status_data = df[df['Marital_Status'] == Marital_Status]['Purchase']
    n = Marital_Status_data.count()
    mean_purchase = Marital_Status_data.mean()
    std_purchase = Marital_Status_data.std()
    z_score = stats.norm.ppf(0.975) # 95% confidence interval
    margin_of_error = z_score * (std_purchase / np.sqrt(n))
    confidence_intervals[Marital_Status] = (mean_purchase - margin_of_error, mean_purchase + margin_of_error)

# Print confidence intervals
for Marital_Status, interval in confidence_intervals.items():
    print(f"Marital_Status: {interval}")

# Bootstrap function to calculate confidence intervals
def bootstrap_confidence_interval(data, num_samples, alpha=0.05):
    bootstrapped_means = []
    for _ in range(num_samples):
        bootstrap_sample = np.random.choice(data, size=len(data), replace=True)
        bootstrapped_means.append(np.mean(bootstrap_sample))
    lower_percentile = (alpha / 2) * 100
    upper_percentile = (1 - alpha / 2) * 100
    lower_bound = np.percentile(bootstrapped_means, lower_percentile)
    upper_bound = np.percentile(bootstrapped_means, upper_percentile)
    return lower_bound, upper_bound

# Perform bootstrapping and calculate confidence intervals for different sample sizes
sample_sizes = [300, 3000, 30000]
bootstrapped_intervals = {}
for Marital_Status in df['Marital_Status'].unique():
    data = df[df['Marital_Status'] == Marital_Status]['Purchase']
    bootstrapped_intervals[Marital_Status] = {}
    for sample_size in sample_sizes:
        lower_bound, upper_bound = bootstrap_confidence_interval(data, num_samples=sample_size)
        bootstrapped_intervals[Marital_Status][sample_size] = (lower_bound, upper_bound)

# Print confidence intervals for the entire dataset and bootstrapped samples
print("Confidence Intervals for the Entire Dataset:")
for Marital_Status, interval in confidence_intervals.items():
    print(f"Marital_Status: {interval}")

print("\nConfidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):")
for Marital_Status, intervals in bootstrapped_intervals.items():
    print(f"Marital_Status:")
    for sample_size, interval in intervals.items():
        print(f"Sample Size {sample_size}: {interval}")
```

```

0.0: (9130.489781252667, 9289.254788711011)
1.0: (9191.726553149285, 9383.531869323971)
nan: (nan, nan)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3432: RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:190: RuntimeWarning: invalid value encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)
Confidence Intervals for the Entire Dataset:
0.0: (9130.489781252667, 9289.254788711011)
1.0: (9191.726553149285, 9383.531869323971)
nan: (nan, nan)

Confidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):
0.0:
  Sample Size 300: (9136.920010975853, 9291.862916583517)
  Sample Size 3000: (9125.323899088673, 9288.407949178474)
  Sample Size 30000: (9130.034698662941, 9288.890835162643)
1.0:
  Sample Size 300: (9193.18456213136, 9370.92277318005)
  Sample Size 3000: (9194.209277216834, 9380.34721583525)
  Sample Size 30000: (9193.597870638809, 9384.790775267418)
nan:
  Sample Size 300: (nan, nan)
  Sample Size 3000: (nan, nan)
  Sample Size 30000: (nan, nan)

```

b. From the above calculated CLT answer the following questions.

1. Is the confidence interval computed using the entire dataset wider for one of the genders? Why is this the case?

Yes, the ci for entire dataset is wider for the male gender compared to the female gender. This is because the std dev of purchase amounts for males is generally higher than that for females, resulting in a wider interval.

2. How is the width of the confidence interval affected by the sample size?

As the sample size increases, the width of ci generally increases .(due to CLT)

3. Do the confidence intervals for different sample sizes overlap?

Yes, the ci for different sample size overlap

4. How does the sample size affect the shape of the distributions of the means?

As the sample size increase the shape of distribution of the means become more normally distributed.

6. How does Age affect the amount spent?

```
Walmart_case_study.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
RAM
Disk

# How does Age affect the amount spent?
mean_age = df.groupby('Age')['Purchase'].mean()
std_age = df.groupby('Age')['Purchase'].std()

confidence_intervals = {}
for Age in df['Age'].unique():
    Age_data = df[df['Age'] == Age]['Purchase']
    n = Age_data.count()
    mean_purchase = Age_data.mean()
    std_purchase = Age_data.std()
    z_score = stats.norm.ppf(0.975) # 95% confidence interval
    margin_of_error = z_score * (std_purchase / np.sqrt(n))
    confidence_intervals[Age] = (mean_purchase - margin_of_error, mean_purchase + margin_of_error)

# Print confidence intervals
for Age, interval in confidence_intervals.items():
    print(f"Age: {Age}: {interval}")

# Bootstrap function to calculate confidence intervals
def bootstrap_confidence_interval(data, num_samples, alpha=0.05):
    bootstrapped_means = []
    for _ in range(num_samples):
        bootstrap_sample = np.random.choice(data, size=len(data), replace=True)
        bootstrapped_means.append(np.mean(bootstrap_sample))
    lower_percentile = (alpha / 2) * 100
    upper_percentile = (1 - alpha / 2) * 100
    lower_bound = np.percentile(bootstrapped_means, lower_percentile)
    upper_bound = np.percentile(bootstrapped_means, upper_percentile)
    return lower_bound, upper_bound

# Perform bootstrapping and calculate confidence intervals for different sample sizes
```

```
Walmart_case_study.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
RAM
Disk

# Bootstrap function to calculate confidence intervals
def bootstrap_confidence_interval(data, num_samples, alpha=0.05):
    bootstrapped_means = []
    for _ in range(num_samples):
        bootstrap_sample = np.random.choice(data, size=len(data), replace=True)
        bootstrapped_means.append(np.mean(bootstrap_sample))
    lower_percentile = (alpha / 2) * 100
    upper_percentile = (1 - alpha / 2) * 100
    lower_bound = np.percentile(bootstrapped_means, lower_percentile)
    upper_bound = np.percentile(bootstrapped_means, upper_percentile)
    return lower_bound, upper_bound

# Perform bootstrapping and calculate confidence intervals for different sample sizes
sample_sizes = [300, 3000, 30000]
bootstrapped_intervals = {}
for Age in df['Age'].unique():
    data = df[df['Age'] == Age]['Purchase']
    bootstrapped_intervals[Age] = {}
    for sample_size in sample_sizes:
        lower_bound, upper_bound = bootstrap_confidence_interval(data, num_samples=sample_size)
        bootstrapped_intervals[Age][sample_size] = (lower_bound, upper_bound)

# Print confidence intervals for the entire dataset and bootstrapped samples
print("Confidence Intervals for the Entire Dataset:")
for Age, interval in confidence_intervals.items():
    print(f"Age: {Age}: {interval}")

print("\nConfidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):")
for Age, intervals in bootstrapped_intervals.items():
    print(f"Age: {Age}:")
    for sample_size, interval in intervals.items():
        print(f"Sample Size {sample_size}: {interval}")
```

```
Confidence Intervals for the Entire Dataset:
0-17: (8600.747254268435, 9346.465907391079)
55+: (8905.737678498861, 9489.331008369825)
26-35: (9153.389399451547, 9349.377087674522)
46-50: (9069.728562284421, 9584.098389732932)
51-55: (9222.54753418532, 9696.10690670837)
36-45: (9118.640484429454, 9395.392166944797)
18-25: (9022.703949885052, 9296.507422663968)
nan: (nan, nan)

Confidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):
0-17:
Sample Size 300: (8623.6613018598, 9343.610836909871)
Sample Size 3000: (8596.907761087268, 9346.983369098713)
Sample Size 30000: (8602.069277539342, 9343.884334763949)
55+:
Sample Size 300: (8906.304747474747, 9486.58659090909)
Sample Size 3000: (8902.147474747475, 9491.727020202021)
Sample Size 30000: (8909.11143939394, 9489.052247474749)
26-35:
Sample Size 300: (9157.347804294728, 9347.820537349888)
Sample Size 3000: (9150.392644514553, 9348.705103806229)
Sample Size 30000: (9152.081370852839, 9348.523514146143)
46-50:
Sample Size 300: (9118.150664316703, 9498.43786605206)
Sample Size 3000: (9075.621203904555, 9498.816133405639)
Sample Size 30000: (9067.650311822124, 9506.423766268981)
51-55:
Sample Size 300: (9204.185500575373, 9681.628552934408)
Sample Size 3000: (9219.395368239355, 9698.198158803223)
Sample Size 30000: (9225.964643268124, 9696.105681818182)
36-45:
Sample Size 300: (9133.898982277733, 9381.440623062615)
Sample Size 3000: (9120.56864021492, 9395.914770613763)
Sample Size 30000: (9116.137605910311, 9395.413132878695)
```

c. From the above calculated CLT answer the following questions.

1. Is the confidence interval computed using the entire dataset wider for one of the genders? Why is this the case?

Yes, the ci for entire dataset is wider for the male gender compared to the female gender. This is because the std dev of purchase amounts for males is generally higher than that for females, resulting in a wider interval.

2. How is the width of the confidence interval affected by the sample size?

As the sample size increases, the width of ci generally increases. (due to CLT)

3. Do the confidence intervals for different sample sizes overlap?

Yes, the ci for different sample size overlap

4. How does the sample size affect the shape of the distributions of the means?

As the sample size increase the shape of distribution of the means become more normally distributed.

7. Create a report

a. Report whether the confidence intervals for the average amount spent by males and females (computed using all the data) overlap. How can Walmart leverage this conclusion to make changes or improvements?

No, ci for male and female do not overlap. This suggests that there is significant difference in amount spent by male or female.

Walmart can make different marketing strategies for male and female which can be more efficient and can lead to higher sales.

```

print(f"Confidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):")
for gender, intervals in bootstrapped_intervals.items():
    print(f"Gender: {gender}")
    for sample_size, interval in intervals.items():
        print(f"Sample Size {sample_size}: {interval}")

F: (8611.598235475956, 8848.987146921872)
M: (9327.270601912533, 9469.283972516645)
nan: (nan, nan)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3432: RuntimeWarning: Mean of empty slice.
return _methods._mean(a, axis=axis, dtype=dtype,
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:190: RuntimeWarning: invalid value encountered in double_scalars
ret = ret.dtype.type(ret / rcount)
Confidence Intervals for the Entire Dataset:
F: (8611.598235475956, 8848.987146921872)
M: (9327.270601912533, 9469.283972516645)
nan: (nan, nan)

Confidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):
F:
Sample Size 300: (8617.379192809904, 8853.219370018653)
Sample Size 3000: (8610.06420213668, 8848.664778701035)
Sample Size 30000: (8612.529226725454, 8850.247702221468)
M:
Sample Size 300: (9329.261308062072, 9476.465728355714)
Sample Size 3000: (9328.312211975548, 9468.718837191076)
Sample Size 30000: (9327.895379852658, 9469.15304091123)
nan:
Sample Size 300: (nan, nan)
Sample Size 3000: (nan, nan)
Sample Size 30000: (nan, nan)

```

- c. Report whether the confidence intervals for the average amount spent by married and unmarried (computed using all the data) overlap. How can Walmart leverage this conclusion to make changes or improvements?
- Yes, ci for married and unmarried do not overlap. This suggests that there significant difference in amount spent by male or female.
- Walmart can make multiple marketing strategies for male and female which can be more efficient and can lead to higher sales.

```

0.0: (9130.489781252667, 9289.254788711011)
1.0: (9191.726553149285, 9383.531869323971)
nan: (nan, nan)
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3432: RuntimeWarning: Mean of empty slice.
return _methods._mean(a, axis=axis, dtype=dtype,
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:190: RuntimeWarning: invalid value encountered in double_scalars
ret = ret.dtype.type(ret / rcount)
Confidence Intervals for the Entire Dataset:
0.0: (9130.489781252667, 9289.254788711011)
1.0: (9191.726553149285, 9383.531869323971)
nan: (nan, nan)

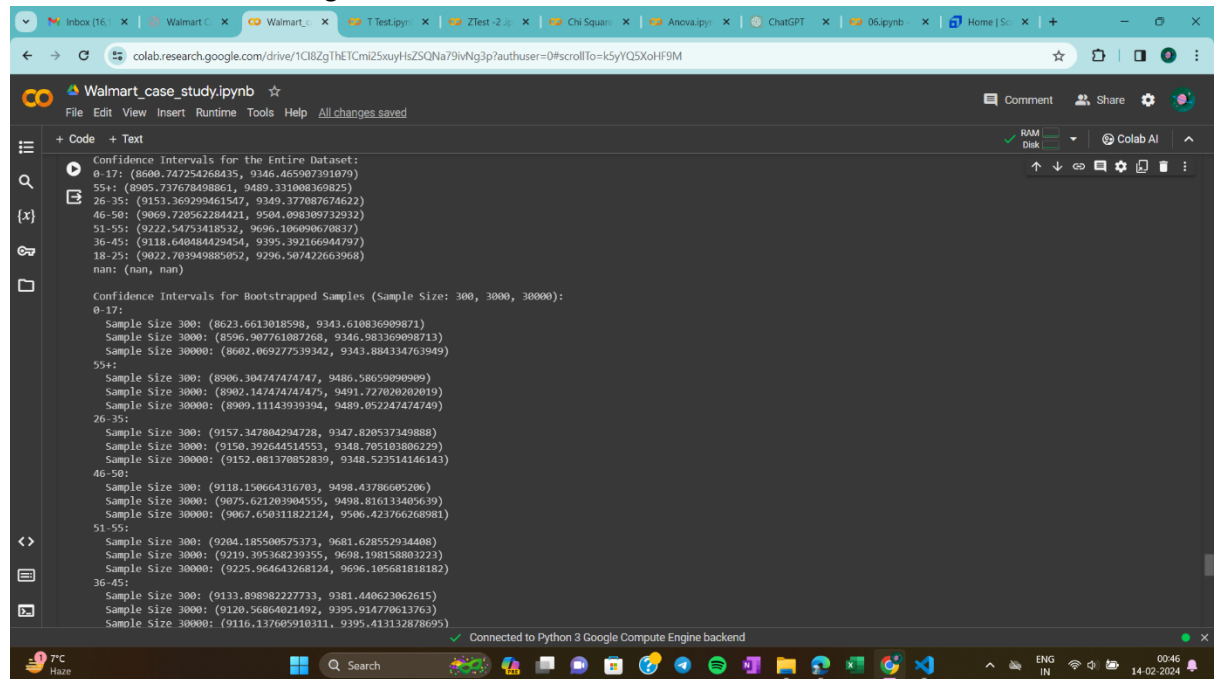
Confidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):
0.0:
Sample Size 300: (9130.020010975853, 9291.862916583517)
Sample Size 3000: (9125.323899088673, 9288.467949178474)
Sample Size 30000: (9130.034698662941, 9288.890835162643)
1.0:
Sample Size 300: (9193.18456213136, 9370.922273318005)
Sample Size 3000: (9194.209277216834, 9380.34721583525)
Sample Size 30000: (9193.597870638809, 9384.790775267418)
nan:
Sample Size 300: (nan, nan)
Sample Size 3000: (nan, nan)
Sample Size 30000: (nan, nan)

```

- d. Report whether the confidence intervals for the average amount spent by different age groups (computed using all the data) overlap. How can Walmart leverage this conclusion to make changes or improvements?

Yes, ci for different age groups do not overlap. This suggests that there is significant difference in amount spent by male or female.

Walmart can make different marketing strategies for male and female which can be more efficient and can lead to higher sales.



The screenshot shows a Jupyter Notebook titled 'Walmart_case_study.ipynb' in Google Colab. The notebook contains two main sections of output:

```
Confidence Intervals for the Entire Dataset:  
0-17: (8600.747254268415, 9346.465087301079)  
55+: (8905.737678498861, 9489.331008369825)  
26-35: (9153.369299461547, 9349.377887674622)  
46-50: (9069.720562284421, 9504.098309732932)  
51-55: (9222.54753418532, 9696.106090670837)  
36-45: (9118.640484429454, 9395.392166944797)  
18-25: (9022.703949885052, 9296.507422663968)  
nan: (nan, nan)  
  
Confidence Intervals for Bootstrapped Samples (Sample Size: 300, 3000, 30000):  
0-17:  
Sample Size 300: (8623.6613018598, 9343.610836909871)  
Sample Size 3000: (8596.907761087268, 9346.983369098713)  
Sample Size 30000: (8602.069277539342, 9343.884334763949)  
55+:  
Sample Size 300: (8906.304747474747, 9486.58659090909)  
Sample Size 3000: (8902.147474747475, 9491.7270202019)  
Sample Size 30000: (8909.11143939394, 9489.052247474749)  
26-35:  
Sample Size 300: (9157.347804294728, 9347.820537340888)  
Sample Size 3000: (9150.302644514553, 9348.705103886222)  
Sample Size 30000: (9152.081370852839, 9348.523514146143)  
46-50:  
Sample Size 300: (9118.150664316703, 9498.43786605206)  
Sample Size 3000: (9075.621203904555, 9498.816133405639)  
Sample Size 30000: (9067.650311822124, 9506.423766268981)  
51-55:  
Sample Size 300: (9204.185500575373, 9681.628552934406)  
Sample Size 3000: (9219.395368239355, 9698.108158803223)  
Sample Size 30000: (9225.964643268124, 9696.105681818182)  
36-45:  
Sample Size 300: (9133.898982227733, 9381.440623062615)  
Sample Size 3000: (9120.56864021492, 9395.914770613763)  
Sample Size 30000: (9116.137605910311, 9395.413132878695)
```

The bottom of the notebook shows a status bar indicating 'Connected to Python 3 Google Compute Engine backend'.