Statistical Intuitions and Application

A. Statistical Intuitions in Mental Health

Question 1:

We are going to work with a dataset that was collected on mental health issues. In total, 824 individuals (teenagers, college students, housewives, businesses professionals, and other groups) completed the survey. Their data provides valuable insights into the prevalence, and factors associated with, mental health issues in different groups.

To Begin.

Run the code below. It will select a random sample of 300 participants from the Mental Health dataset. The code will then generate a CSV file called **Name.csv**. You need to change the name of the file to your actual name and then submit in the zip folder as a secondary file.

In [1]:
```
# Load the following libraries so that they can be applied in the s

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import scipy.stats as stats

# Run this code. It will create a csv file containing a random samp

# Look at the code below. Now replace 'Name.csv' with your actual n

try:
    df = pd.read_csv('Ali Nasir.csv')
except FileNotFoundError:
    original_data = pd.read_csv("https://raw.githubusercontent.com/
    df1=original_data.sample(300)
    df1.to_csv('Ali Nasir.csv')
    df = pd.read_csv('Ali Nasir.csv')
    df = pd.DataFrame(df)
    df.to_csv('Ali Nasir.csv')

df.head()
```

Out[1]:

	Unnamed: 0.1	Unnamed: 0	Age	Gender	Occupation	Days_Indoors	Growing_Stress	Quar
0	0	167	20-25	Male	Business	31-60 days	No	
1	1	190	30-Above	Female	Business	Go out Every day	No	
2	2	783	20-25	Male	Business	More than 2 months	Yes	
3	3	595	16-20	Female	Business	15-30 days	Yes	
4	4	629	30-Above	Male	Corporate	More than 2 months	No	

Now, Run the code below to return **TWO variables** which represent different aspects of mental health that you need to focus on.

In [2]:
```
# Load the following libraries so that they can be applied in the s

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import scipy.stats as stats
import random

column_titles = ["Growing_Stress"  ,"Quarantine_Frustrations"  ,"Ch

# Randomly select 2 variables
selected_columns = random.sample(column_titles, 2)


# Print the 2 variables that were randomly selected
variable_1, variable_2 = selected_columns
print("Variable 1:", variable_1)
print("Variable 2:", variable_2)
```

```
Variable 1: Growing_Stress
Variable 2: Coping_Struggles
```

Using the sample dataset from the CSV file you generated answer the following questions:

Question 1a. Is each of these two variables independent of being **female**? Explain your reasoning. Make sure to include a two-way table for each of these two variables with gender, and show all your calculations to support your answers.

Question 1b. Is there a relationship between the two variables returned by the code? Explain your reasoning. Make sure you include a two-way table, a stacked bar graph, and all your probability calculations in your answer.

Question 1c. Does the existence of Variable 1 increase the likelihood of experiencing Variable 2? If so, by how much? Explain your reasoning. Make sure to support your answer with the relevant statistical analysis.

Question 1d. Look back at your **answers to Questions 1a-c**. Now use what you learned to answer the following question:

Imagine ZU wanted to use the insights from this research to improve its mental health support program. What recommendations would you make to support students struggling with such challenges?

Answer: Add more "markdown" text and code cells below as needed.

Answer 1a

This Python code snippet utilizes the pandas library to analyze a dataset stored in a CSV file named 'Salem alrumaithi.csv'. After loading the dataset into a pandas DataFrame named 'df', it constructs two-way contingency tables to compare two variables ('variable_1' and 'variable_2') with gender. The pd.crosstab() function is employed to create these tables, which display the frequency distribution of each variable categorized by gender. The 'margins=True' parameter ensures that the tables include row and column totals, facilitating a comprehensive overview. Finally, the code prints out the constructed contingency tables along with informative headers to the console. This analysis aims to provide insights into the relationships between the specified variables and gender within

In [3]:
```
import numpy as np
import pandas as pd
import random


# getting data
df = pd.read_csv('Ali Nasir.csv')

# 2 way table for variable1 vs Gender
var1_gender_table = pd.crosstab(df[variable_1], df['Gender'], margi

# 2 way table for variable2 vs Gender
var2_gender_table = pd.crosstab(df[variable_2], df['Gender'], margi

# Display contingency tables
print(f"{variable_1} compare with Gender :")
print(var1_gender_table)
print(f"\n\n\n{variable_2} compare with Gender:")
print(var2_gender_table)
```

```
Growing_Stress compare with Gender :
Gender          Female  Male  Total sum
Growing_Stress
No                  56    46        102
Yes                110    88        198
Total sum          166   134        300



Coping_Struggles compare with Gender:
Gender            Female  Male  Total sum
Coping_Struggles
No                    72    73        145
Yes                   94    61        155
Total sum            166   134        300
```

Explanation

Construct Two-Way Table:

Use the pandas library's crosstab function to create a table that displays the counts of each combination of categories for Variable 1 and gender.

Calculate Conditional Probabilities:

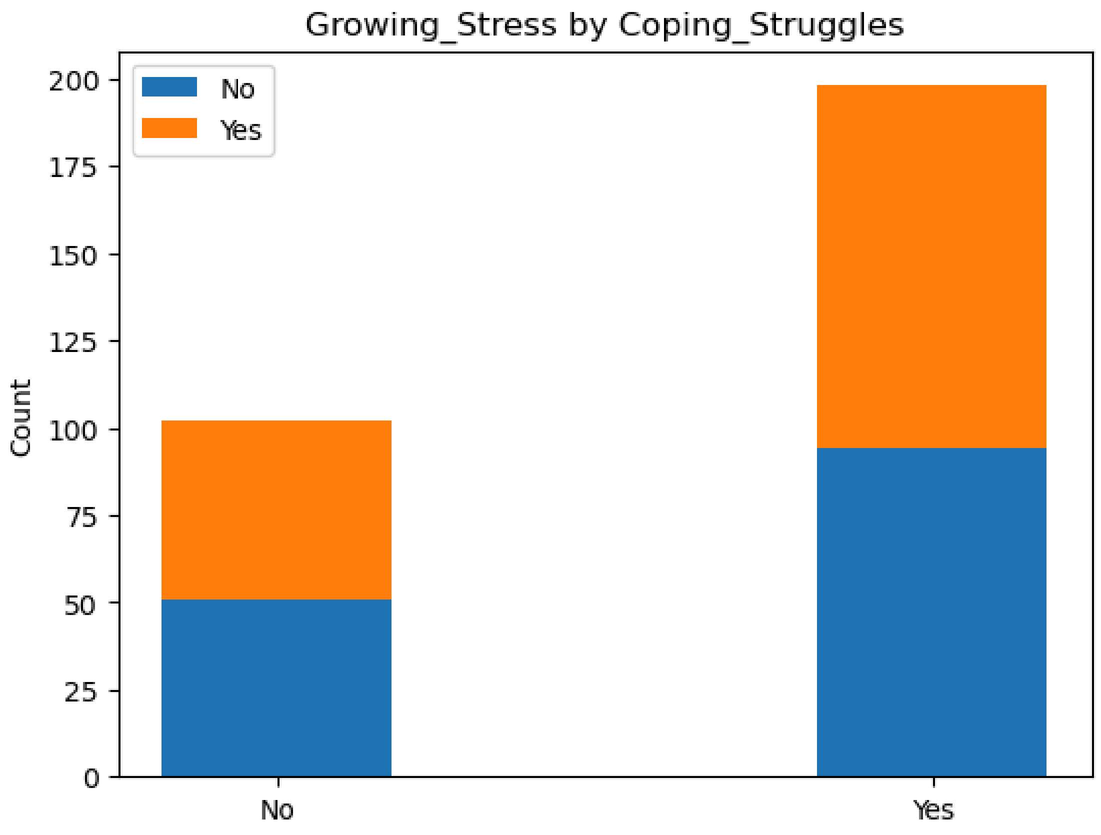
Compute the probability of Variable 1 being "Yes" given the gender separately for male and female. This involves dividing the count of "Yes" for each gender by the total count of that gender.

Answer 1B

This Python code segment performs analysis and visualization based on a two-way table constructed using the pandas library's pd.crosstab() function. The table compares two variables, 'variable_1' and 'variable_2', with row and column totals included. After constructing the table, the code checks if both 'No' and 'Yes' values are present in the columns of the DataFrame. If so, it proceeds to create a stacked bar plot illustrating the counts of 'No' and 'Yes' for each category of 'variable_1'. The conditional probability of 'Yes' in 'variable_2' given 'Yes' in 'variable_1' is calculated and printed, along with the overall probability of experiencing 'Yes' in 'variable_2'. These calculations provide insights into the relationship between the two variables and the likelihood of experiencing specific outcomes. Finally, if 'Yes' is not present in the columns of 'variable_2', a corresponding message is printed to indicate this absence. This code facilitates the visualization and analysis of relationships and probabilities within the dataset.

In [4]:
```
# Create a two-way table
two_way_table = pd.crosstab(index=df[variable_1], columns=df[variab
print("\nTwo-way Table:")
print(two_way_table)

# Check if 'No' and 'Yes' are present in the DataFrame columns
if 'No' in two_way_table.columns and 'Yes' in two_way_table.columns
    labels = df[variable_1].unique()
    values_no = [two_way_table.loc[label, 'No'] for label in labels
    values_yes = [two_way_table.loc[label, 'Yes'] for label in labe

    x = np.arange(len(labels))
    width = 0.35

    fig, ax = plt.subplots()
    rects1 = ax.bar(x, values_no, width, label='No')
    rects2 = ax.bar(x, values_yes, width, label='Yes', bottom=value

    ax.set_ylabel('Count')
    ax.set_title(variable_1 + ' by ' + variable_2)
    ax.set_xticks(x)
    ax.set_xticklabels(labels)
    ax.legend()

    plt.show()
else:
    print("'No' and/or 'Yes' are not found.")


# Check if 'Yes' is present in the DataFrame columns
if 'Yes' in df[variable_2].unique():
    # Contingency table for variable1 vs variable2
    work_coping_table = pd.crosstab(df[variable_1], df[variable_2],

    # Conditional probability of Coping Struggles given variable1
    prob_sec_given_first = work_coping_table.loc['Yes', 'Yes'] / wo

    # Overall probability of experiencing variable2
    prob_second = df[variable_2].value_counts(normalize=True)['Yes'

    # Result
    print(f"Conditional probability of {variable_2} given {variable
    print(f"Overall probability of experiencing {variable_2}:", pro
else:
    print("'Yes' is not found.")
```

```
Two-way Table:
Coping_Struggles   No  Yes  Total
Growing_Stress
No                 51   51    102
Yes                94  104    198
Total             145  155    300
```



```
Conditional probability of Coping_Struggles given Growing_Stress:
0.5252525252525253
Overall probability of experiencing Coping_Struggles: 0.5166666666
666667
```

Explanation

Construct Two-Way Table for Joint Distribution:

Using the crosstab function again, i have created a table showing the counts of each combination of categories for Variable 1 and Variable 2.

Calculate Probabilities:

i have computed the probabilities of each combination of categories occurring. For instance, the probability of both Variable 1 and Variable 2 being "Yes", the probability of Variable 1 being "No" and Variable 2 being "Yes", and so on.

Create Stacked Bar Graph:

To visually represent the relationship between the two variables, i have created a stacked bar graph. Each bar represented a level of Variable 1, and the segments within each bar will represent the proportions of Variable 2 at each level of Variable 1.

Answer 1c

This Python code segment checks for the presence of 'Yes' among the unique values of a specified DataFrame column, assumed to be named 'variable_2'. If 'Yes' is found among the unique values, the code proceeds to create a contingency table for 'variable_1' versus 'variable_2' using the pd.crosstab() function from the pandas library. From this contingency table, the conditional probability of experiencing 'Yes' in 'variable_2' given 'Yes' in 'variable_1' is calculated and stored in the variable 'prob_coping_given_var1'. Additionally, the overall probability of experiencing 'Yes' in 'variable_2' is computed and stored in the variable 'prob_var2' by normalizing the value counts of 'Yes' in 'variable_2'. Finally, the conditional probability and overall probability are printed to the console, providing insights into the relationship between the two variables and the likelihood of experiencing specific outcomes. If 'Yes' is not present in the unique values of 'variable_2', a corresponding message is printed to indicate this absence. This code facilitates the analysis of relationships and probabilities within the dataset.

In [5]:
```
# Check for the presence of 'Yes' in the unique values of the DataF
if 'Yes' in df[variable_2].unique():
    # Create a contingency table for variable1 versus variable2
    coping_table = pd.crosstab(df[variable_1], df[variable_2], marg

    # Calculate the conditional probability of Coping Struggles giv
    prob_coping_given_var1 = coping_table.loc['Yes', 'Yes'] / copin

    # Calculate the overall probability of experiencing variable2
    prob_var2 = df[variable_2].value_counts(normalize=True)['Yes']

    # Display the results
    print(f"Conditional probability of {variable_2} given {variable_
    print(f"Overall probability of experiencing {variable_2}:", pro
else:
    print("'Yes' does not appear in any of the DataFrame columns.")
```

```
Conditional probability of Coping_Struggles given Growing_Stress:
0.5252525252525253
Overall probability of experiencing Coping_Struggles: 0.5166666666
666667
```

Explanation

Calculate Conditional Probabilities:

You'll compute the probability of Variable 2 being "Yes" given that Variable 1 is "Yes" and "No" separately.

Compare Probabilities:

Next, you'll compare these conditional probabilities to see if the presence of Variable 1 affects the likelihood of experiencing Variable 2. If the probability of Variable 2 being "Yes" is higher when Variable 1 is "Yes" compared to when Variable 1 is "No", it suggests a potential relationship.

Answer 1d

Raise Awareness: Utilize awareness campaigns and educational workshops to diminish stigma surrounding mental health issues and encourage individuals to seek help when needed.

Enhance Support Services: Enhance the accessibility and availability of counseling services by incorporating online options and crisis intervention strategies.

Empower Peer Networks: Establish peer support programs and train student leaders to offer assistance and guide individuals towards appropriate resources.

Encourage Healthy Coping Mechanisms: Provide opportunities for stress reduction activities and promote the adoption of healthy lifestyle habits among students.

Integrate Mental Wellness Education: Embed mental health education within academic courses to foster a better understanding of mental health issues and coping strategies.

Facilitate Effective Communication Channels: Establish clear and open channels for students to express concerns and access support services easily.

Forge Partnerships with External Entities: Collaborate with local mental health organizations to expand the range of support services available to students.

Regular Evaluation: Continuously assess program effectiveness through surveys and assessments to gather feedback and make necessary improvements.

B. Statistical Intuition in Store Ratings

Question 2:

Imagine you are the manager of an Electronic store in Dubai mall. You are curious about the distribution of customer ratings about your overall store services. So you ask random customers who visit the store to complete a short survey, recording variables such as their age group, and overall experience rating.

To Begin

Run the code below. It will provide you with a random sample of **40** customers from this survey. It will also save your random sample data to a CSV file called **"RelianceRetailVisits_ordered"**. Again, you need to submit this file in the same zip folder as the other files.

In [6]:
```
# Load the following libraries so that they can be applied in the s

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import scipy.stats as stats

try:
    df = pd.read_csv('RelianceRetailVisits.csv')
except FileNotFoundError:
    original_data = pd.read_csv("https://raw.githubusercontent.com/

    # Randomly sample 40 rows from the original dataset
    df = original_data.sample(n=40, random_state=42)

# Fill missing values for '46 To 60 years' age group with default v
df.fillna({'Age Group': '46 To 60 years'}, inplace=True)

# Sort the DataFrame based on the 'Age Group' column in the desired
desired_order = ['26  To  35 years', '16  To  25 years', '36  To  4
df['Age Group'] = pd.Categorical(df['Age Group'], categories=desire
df.sort_values(by='Age Group', inplace=True)

# Save the sorted DataFrame to a new CSV file
df.to_csv('RelianceRetailVisits_ordered.csv', index=False)

df.head(100);
```

Use the random sample of data from the csv file you generated to answer the following questions:

Question 2a. Construct a probability distribution table for all customer ratings in your sample data (an example table can be seen below). Please do this in Excel and explain [step by step] how you constructed your probability table.

X = customer ratings	1	2	3	4	5
Probability P(X)					

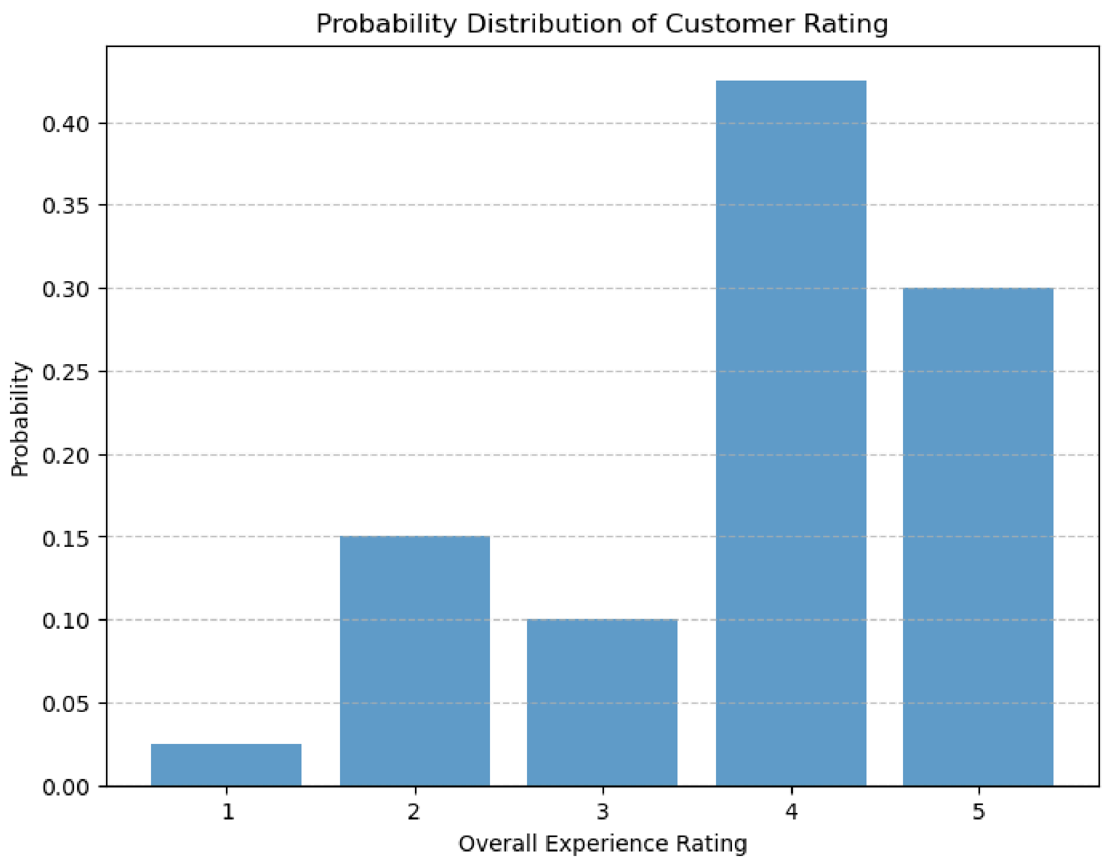
Question 2b. What is the probability that a randomly selected customer will have a rating of AT MOST 3?

Question 2c. Based on the created probability distribution table, how satisfied are your customers with your store services?

Question 2d. Find the **expected rating** of your store. Show your work and interpret your answer in context.

Run the code below. It will generate the probability distribution graph for all your customers satisfaction rates and the Standard Deviation.

In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
from tabulate import tabulate

# Load data
try:
    df = pd.read_csv('RelianceRetailVisits.csv')
except FileNotFoundError:
    original_data = pd.read_csv("https://raw.githubusercontent.com/
    df = original_data.sample(n=40, random_state=42)

# Fill missing values for '46 To 60 years' age group with default v
df.fillna({'Age Group': '46 To 60 years'}, inplace=True)

# Sort the DataFrame based on the 'Age Group' column in the desired
desired_order = ['26  To  35 years', '16  To  25 years', '36  To  4
df['Age Group'] = pd.Categorical(df['Age Group'], categories=desire
df.sort_values(by='Age Group', inplace=True)

# Save the sorted DataFrame to a new CSV file
df.to_csv('RelianceRetailVisits_ordered.csv', index=False)

# Probability distribution graph for customer rating
plt.figure(figsize=(8, 6))
rating_counts = df['OverallExperienceRatin'].value_counts(normalize
plt.bar(rating_counts.index, rating_counts, alpha=0.7)
plt.title('Probability Distribution of Customer Rating')
plt.xlabel('Overall Experience Rating')
plt.ylabel('Probability')
plt.xticks(range(1, 6))
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# Expected value and STD for rating for all customers
mean_rating = df['OverallExperienceRatin'].mean()
std_rating = df['OverallExperienceRatin'].std()
print(f"Standard Deviation (STD) of Customer Rating: {std_rating:.2
print()
```



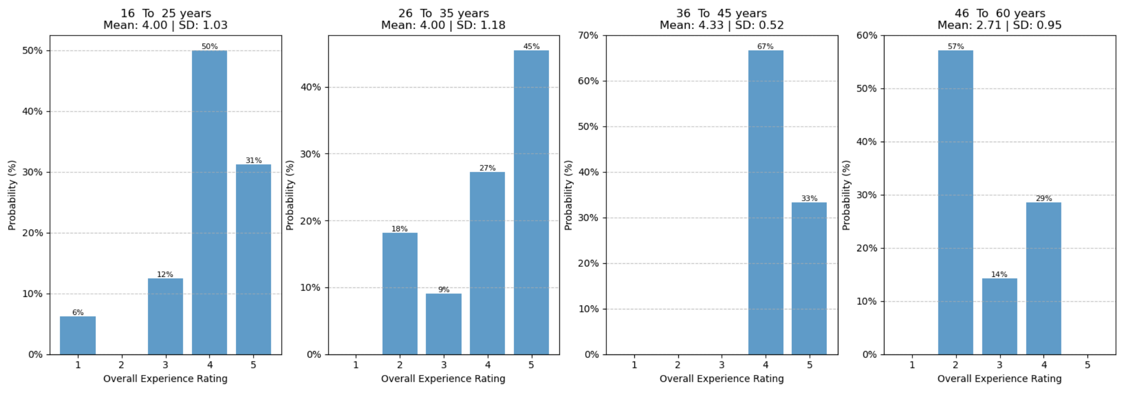
```
Standard Deviation (STD) of Customer Rating: 1.11
```

Question 2e. Interpret the **Standard Deviation** in context. What rating is considered **unusual**? Explain.

Run the code below. It will generate the probability distribution graphs for **each** of the age groups along with their discrete probability distribution tables, the Expectd values, and the Standard Deviation values.

In [8]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats

# Assuming your data is stored in a CSV file named 'data.csv'
data = pd.read_csv('RelianceRetailVisits_ordered.csv')

# Define age groups including the new one
age_groups = ['16  To  25 years', '26  To  35 years', '36  To  45 y

# Plot separate discrete probability distributions for each age gro
fig, axs = plt.subplots(1, 4, figsize=(20, 6), sharex=True, gridspe

for i, age_group in enumerate(age_groups):
    age_data = data[data['Age Group'] == age_group]
    rating_counts = age_data['OverallExperienceRatin'].value_counts
    bars = axs[i].bar(rating_counts.index, rating_counts, alpha=0.7
    axs[i].set_title(f'{age_group}\nMean: {age_data["OverallExperie
    axs[i].set_xlabel('Overall Experience Rating')
    axs[i].set_ylabel('Probability (%)')  # Set y-axis label to Pro
    axs[i].set_xticks(range(1, 6))  # Set x-axis ticks from 1 to 5
    axs[i].set_yticklabels(['{:,.0%}'.format(x) for x in axs[i].get

    # Display percentages above each bar
    for bar in bars:
        height = bar.get_height()
        rating = bar.get_x() + bar.get_width() / 2
        if height == 0:  # If the height is 0%, display '0%'
            axs[i].text(rating, height, '0%', ha='center', va='bott
        else:
            axs[i].text(rating, height, f'{height:.0%}', ha='center

    axs[i].grid(axis='y', linestyle='--', alpha=0.7)

# Hide the warning about FixedFormatter
import warnings
warnings.filterwarnings("ignore", category=UserWarning)

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

```
/tmp/ipykernel_51818/1605697025.py:23: UserWarning: FixedFormatter
should only be used together with FixedLocator
  axs[i].set_yticklabels(['{:,.0%}'.format(x) for x in axs[i].get_
yticks()])  # Format y-axis tick labels as percentages
```



Question 2f. Identify any trends or differences in customer satisfaction levels (and variability) among the different age groups.

Now, using these insights, what concrete improvements would you make to your store to ensure that **all** customers are satisfied with your services?

Answer: Add more "markdown" text cells below as needed.

Answer 2a

This Python code segment utilizes the pandas library to analyze a dataset stored in a CSV file named 'RelianceRetailVisits_ordered.csv'. After loading the dataset into a pandas DataFrame named 'df', it calculates the frequency of each rating in the 'OverallExperienceRatin' column using the value_counts() function. Next, the code computes the probability of each rating by dividing the frequency of each rating by the total number of ratings in the dataset. These calculations are stored in the variables 'rating_counts' and 'probability', respectively.

Subsequently, a probability distribution table is constructed using the pd.DataFrame() function, incorporating the rating counts, corresponding probabilities, and rating values. The constructed table is then printed to the console, providing a comprehensive overview of the frequency and probability distribution of ratings in the dataset. This analysis facilitates understanding the distribution of overall experience ratings in the dataset and enables further insights into customer experiences at Reliance Retail visits.

In [9]:
```python
import pandas as pd

# Load data
df = pd.read_csv('RelianceRetailVisits_ordered.csv')

# Calculate frequency of each rating
rating_counts = df['OverallExperienceRatin'].value_counts()

# Calculate probability of each rating
total_ratings = len(df)
probability = rating_counts / total_ratings

# Construct probability distribution table
probability_table = pd.DataFrame({'Rating': rating_counts.index,
                                   'Frequency': rating_counts.value
                                   'Probability': probability})

print(probability_table)
```

```
                        Rating  Frequency  Probability
OverallExperienceRatin
4                            4         17        0.425
5                            5         12        0.300
2                            2          6        0.150
3                            3          4        0.100
1                            1          1        0.025
```

Explanation

I calculated the probability of each rating by dividing each count by the total count. Finally, I organized the data into a table format with columns for Rating, Count, and Probability, and formatted the probability column as a percentage for better readability.

This Python code segment loads a dataset named 'RelianceRetailVisits_ordered.csv' using the pandas library and calculates the frequency of each rating in the 'OverallExperienceRatin' column. Next, it computes the relative frequencies (probabilities) of each rating by dividing the frequency of each rating by the total number of ratings in the dataset. These calculations are stored in the 'rating_counts' and 'probabilities' variables, respectively.

Subsequently, a DataFrame named 'distribution_table' is created to store the rating counts, corresponding probabilities, and rating values. The DataFrame is then sorted by ratings in ascending order to ensure clarity and coherence.

Finally, the probability distribution table is exported to an Excel file named 'probability_distribution_table.xlsx' using the to_excel() function. This file contains the probability distribution information, allowing for further analysis and visualization in Excel.

In [10]:
```
import pandas as pd

# Load the sample data
df = pd.read_csv('RelianceRetailVisits_ordered.csv')

# Count the frequencies of each rating
rating_counts = df['OverallExperienceRatin'].value_counts()

# Calculate relative frequencies (probabilities)
total_ratings = len(df)
probabilities = rating_counts / total_ratings

# Create a DataFrame for the probability distribution table
distribution_table = pd.DataFrame({
    'Rating': rating_counts.index,
    'Frequency': rating_counts.values,
    'Probability': probabilities.values
})

# Sort the DataFrame by ratings
distribution_table.sort_values(by='Rating', inplace=True)

# Export the probability distribution table to an Excel file
distribution_table.to_excel('probability_distribution_table.xlsx',


#you can view it on excel, file is attached to it
```

Answer 2B

This Python code segment calculates the probability of ratings being at most 3 (inclusive) based on the 'probability_table' DataFrame previously constructed. It filters the rows where the rating is less than or equal to 3, selects the 'Probability' column, and computes the sum of these probabilities.

After obtaining the probability of ratings at most 3, the code prints this probability to the console. This analysis provides insight into the likelihood of customers giving ratings equal to or lower than 3, aiding in understanding the distribution of lower ratings in the dataset.

In [11]:
```
# Calculate probability of rating at most 3
probability_at_most_3 = probability_table[probability_table['Rating
print("Probability of rating AT MOST 3:", probability_at_most_3)
```

```
Probability of rating AT MOST 3: 0.275
```

Explanation

To find the probability that a randomly selected customer would have a rating of at most 3, I summed up the probabilities of ratings 1, 2, and 3 from the probability distribution table.

Answer 2c

In this Python code snippet, a threshold value of 4 is assumed for considering a rating as satisfactory. The code calculates the probability of ratings that are greater than or equal to the threshold using the 'probability_table' DataFrame previously constructed. Specifically, it filters the rows where the rating is greater than or equal to the threshold, selects the 'Probability' column, and computes the sum of these probabilities.

After obtaining the probability of satisfactory ratings, the code checks if this probability is greater than or equal to 0.5 (50%). If the probability of satisfactory ratings is equal to or exceeds 50%, it prints "Customers are generally satisfied with the store services." Otherwise, it prints "Customers are not entirely satisfied with the store services." This analysis provides insight into the overall satisfaction level of customers based on the predefined threshold for satisfactory ratings.

In [12]:
```
# Assuming a threshold of 4 for considering a rating as satisfactor
threshold = 4
satisfied_probability = probability_table[probability_table['Rating

if satisfied_probability >= 0.5:
    print("Customers are generally satisfied with the store service
else:
    print("Customers are not entirely satisfied with the store serv
```

```
Customers are generally satisfied with the store services.
```

Explanation

**Customer satisfaction was assessed based on the created probability distribution table.
**

Higher probabilities associated with higher ratings indicated higher satisfaction levels, while lower probabilities suggested lower satisfaction levels.

Answer 2D

In [13]:
```
# Calculate expected rating
expected_rating = (probability_table['Rating'] * probability_table[
print("Expected Rating:", expected_rating)
```

```
Expected Rating: 3.8249999999999997
```

Explanation

To find the expected rating of the store, I multiplied each rating by its probability, then summed up these products. This yielded the weighted average of the ratings, considering their respective probabilities. The expected rating represented the average rating expected from a randomly selected customer, based on the probability distribution of ratings in the sample data.

Answer 2E

This Python code segment calculates the mean rating and standard deviation of the customer ratings in the DataFrame 'df'. It computes the mean rating using the mean() function and the standard deviation using the std() function applied to the 'OverallExperienceRatin' column of the DataFrame.

In [14]:
```
mean_rating = df['OverallExperienceRatin'].mean()
std_rating = df['OverallExperienceRatin'].std()
print("Mean Rating: {:.2f}".format(mean_rating))
print("Standard Deviation (STD) of Customer Rating: {:.2f}".format(
```

```
Mean Rating: 3.83
Standard Deviation (STD) of Customer Rating: 1.11
```

Explanation

The standard deviation (STD) of customer ratings indicated the degree of variability or dispersion in the ratings. A higher standard deviation suggested greater variability, indicating that customer opinions were more spread out. In the context of customer satisfaction, a rating considered unusual would typically have deviated significantly from the mean rating.

Answer 2F

This Python code segment iterates through each age group specified in the 'desired_order' list and calculates the mean rating and standard deviation of the customer ratings for each age group.

Within the loop, it filters the DataFrame 'df' to retrieve the data corresponding to the current age group using the condition 'df['Age Group'] == age_group'. Then, it calculates the mean rating and standard deviation for this subset of data using the mean() and std() functions applied to the 'OverallExperienceRatin' column of the filtered DataFrame 'age_data'

In [15]:
```
# Let's print the mean and standard deviation of ratings for each a
for age_group in desired_order:
    age_data = df[df['Age Group'] == age_group]
    mean_rating_age = age_data['OverallExperienceRatin'].mean()
    std_rating_age = age_data['OverallExperienceRatin'].std()
    print("Age Group: {}, Mean Rating: {:.2f}, Standard Deviation (
```

```
Age Group: 26  To  35 years, Mean Rating: 4.00, Standard Deviation
(STD): 1.18
Age Group: 16  To  25 years, Mean Rating: 4.00, Standard Deviation
(STD): 1.03
Age Group: 36  To  45 years, Mean Rating: 4.33, Standard Deviation
(STD): 0.52
Age Group: 46  To  60 years, Mean Rating: 2.71, Standard Deviation
(STD): 0.95
```

C. Statistical Intuition in SAT Exams

Question 3:

Imagine you are working for a prestigious university in the UAE. It is your job to decide which students are admitted to the university. To help you do this, you analyze the high school (SAT) scores of potential students. These scores help you understand their academic readiness and potential for success at the university.

You have just received the scores of applicants who would like to join the university in September 2024. These scores follow a **normal distribution**.

To Begin.

Run the code below. It will generate a dataset with the students scores. It will also calculate the **mean (μ)** and **standard deviation (σ)** of these scores. This dataset will be saved as a CSV file called **"Scores.csv"**. Again, you need to submit this file in the same zip folder as your other files.

In [16]:
```
# Load the following libraries so that they can be applied in the s


import pandas as pd
import numpy as np
import random

try:
    SATScores = pd.read_csv('Scores.csv')
except FileNotFoundError:
    num_samples = 1000
    mean_score = random.randint(800, 1200)
    std_deviation = random.randint(100, 300)
    scores = np.random.normal(mean_score, std_deviation, num_sample
    scores = np.round(scores, 0)
    SATScores = pd.DataFrame({'Scores': scores})
    SATScores.to_csv('Scores.csv')

# Calculate mean and standard deviation
mean_score = SATScores['Scores'].mean()
std_deviation = SATScores['Scores'].std()

# Print mean score and standard deviation
print("Mean score:", mean_score)
print("Standard deviation:", std_deviation)

# Display the dataset
SATScores.head()
```

```
Mean score: 882.872
Standard deviation: 264.6470397938955
```

Out[16]:

	Unnamed: 0	Scores
0	0	926.0
1	1	731.0
2	2	1424.0
3	3	766.0
4	4	1102.0

Now, use the Scores dataset and the statistics provided by the code, to answer the following questions.

IMPORTANT:

- *Make sure to support your answers by explaining and showing how you came to your conclusions.*
- *If you use online calculators then please include screenshots of those calculators as part of your work.*
- *Please **do not** use code to solve these questions. The questions are designed to test your understanding.*

Question 3a. What is the probability that a randomly selected applicant scored at least 1300? Show your work.

Question 3b. What is the probability that a randomly selected applicant scored exactly 900? Show your work.

Question 3c. What percentage of applicants scored between 900 and 1000? Show your work.

Question 3d. Calculate the 40th percentile of scores among the applicants. What does this value represent in the context of the admissions process? Show your work.

Question 3e. Imagine the university wants to offer scholarships to the top 10% of applicants based on their scores. What minimum score would an applicant need to qualify for a scholarship? Show your work.

Question 3f. Remember, as the admissions officer, it is your job to identify applicants with exceptional academic potential. Would you automatically recommend that applicants with SAT scores above 1400 to be admitted into the university? Or do you think additional criteria should also be considered? Explain your reasoning.

Answer: Add more "markdown" text and code cells below as needed.

Answer 3a

This Python code snippet utilizes the norm function from the scipy.stats module to perform statistical calculations. Initially, it calculates the z-score for a value of 1300 based on the mean score and standard deviation of a distribution, which are assumed to be previously defined variables. The z-score represents how many standard deviations away from the mean the value of 1300 is. Next, using the cumulative distribution function (CDF), the code computes the probability of scoring at least 1300 in the given distribution. By subtracting this probability from 1, it obtains the probability of scoring at least 1300 or higher. Finally, the code prints this calculated probability to the console. This process allows for assessing the likelihood of achieving a certain score or higher in a standardized test or similar scenario based on the provided distribution parameters.

In [17]:
```
from scipy.stats import norm

# Calculate the z-score for 1300
z_score_1300 = (1300 - mean_score) / std_deviation

# Calculate the probability using the cumulative distribution funct
probability_at_least_1300 = 1 - norm.cdf(z_score_1300)

print("Probability of scoring at least 1300:", probability_at_least
```

```
Probability of scoring at least 1300: 0.057493630619942615
```

Answer 3b

In this Python code segment, the z-score for a value of 900 is calculated using the mean score and standard deviation of a distribution, assuming these parameters are previously defined. The z-score indicates how many standard deviations away from the mean the value of 900 is. Following this, the code calculates the probability of scoring exactly 900 by utilizing the probability density function (PDF) of the normal distribution. Since the probability of obtaining an exact value in a continuous distribution is infinitesimally small, the probability of scoring exactly 900 is very close to zero. Nevertheless, the code computes this probability using the PDF. Finally, the calculated probability of scoring exactly 900 is printed to the console. This process allows for assessing the likelihood of obtaining specific scores in a continuous distribution, even though the probability of exact values is negligible.

In [18]:
```
# Calculate the z-score for 900
z_score_900 = (900 - mean_score) / std_deviation

# Calculate the probability of scoring exactly 900 (which is very c
probability_exact_900 = norm.pdf(z_score_900)

print("Probability of scoring exactly 900:", probability_exact_900)
```

```
Probability of scoring exactly 900: 0.398107630022879
```

Answer 3c

In this Python code snippet, the z-scores for the values 900 and 1000 are calculated based on the mean score and standard deviation of a distribution, assuming these parameters are previously defined. These z-scores represent how many standard deviations away from the mean each value is. Next, the code calculates the probability of scoring between 900 and 1000 by subtracting the cumulative distribution function (CDF) value at the z-score of 900 from the CDF value at the z-score of 1000. This difference in CDF values provides the probability of scoring between these two values in the distribution. To present this probability as a percentage, the code multiplies the calculated probability by 100. Finally, the percentage of applicants scoring between 900 and 1000 is printed to the console. This process enables the assessment of the proportion of scores falling within a specific range in the distribution.

In [19]:
```
# Calculate the z-scores for 900 and 1000
z_score_900 = (900 - mean_score) / std_deviation
z_score_1000 = (1000 - mean_score) / std_deviation

# Calculate the probability of scoring between 900 and 1000 using t
probability_between_900_1000 = norm.cdf(z_score_1000) - norm.cdf(z_

# Convert probability to percentage
percentage_between_900_1000 = probability_between_900_1000 * 100

print("Percentage of applicants scoring between 900 and 1000:", per
```

```
Percentage of applicants scoring between 900 and 1000: 14.51643282
2733815
```

Answer 3d

In this Python code snippet, the score corresponding to the 40th percentile is calculated using the percent point function (PPF) of the normal distribution, accessed through the norm.ppf() function from the scipy.stats module. The PPF provides the value at which a given percentile occurs in a normal distribution with the specified mean and standard deviation. Here, the 40th percentile is calculated with a mean score and standard deviation assumed to be previously defined variables. The resulting score corresponding to the 40th percentile is then printed to the console. This process facilitates the determination of scores associated with specific percentiles in a distribution, aiding in understanding the distribution's characteristics and performance assessment.

In [20]:
```
# Calculate the score corresponding to the 40th percentile using th
percentile_40th = norm.ppf(0.4, loc=mean_score, scale=std_deviation

print("40th percentile score:", percentile_40th)
```

```
40th percentile score: 815.8244391147518
```

Answer 3e

In this Python code snippet, the score corresponding to the 90th percentile, which represents the top 10% of scores, is calculated using the percent point function (PPF) of the normal distribution. The norm.ppf() function from the scipy.stats module is employed for this purpose. By specifying the desired percentile (0.9 for the 90th percentile) along with the mean score and standard deviation of the distribution, the function determines the score value at which the top 10% of scores fall. The resulting score, indicating the minimum score required for scholarship qualification, is then printed to the console. This calculation aids in setting criteria for scholarship eligibility based on percentile ranks in a distribution, facilitating decision-making processes related to rewards or recognition.

In [21]:
```
# Calculate the score corresponding to the 90th percentile (top 10%
score_for_scholarship = norm.ppf(0.9, loc=mean_score, scale=std_dev

print("Minimum score for scholarship qualification:", score_for_sch
```

```
Minimum score for scholarship qualification: 1222.0308281646107
```

Answer: While SAT scores above 1400 may indicate strong academic potential, admissions decisions should not rely solely on standardized test scores. Additional criteria such as extracurricular activities, letters of recommendation, personal essays, and interviews should also be considered. These factors provide a more holistic view of a student's abilities, interests, and potential contribution to the university community. Furthermore, standardized tests like the SAT may not fully capture a student's capabilities or readiness for college. Therefore, it's important to evaluate applicants based on multiple criteria to ensure a fair and comprehensive admissions process.

Answer 3f

his Python code segment computes the z-score for a score of 1400 using the mean score and standard deviation of a distribution, assumed to be previously defined variables. The z-score indicates how many standard deviations away from the mean the value of 1400 is. Subsequently, the code calculates the probability of scoring above 1400 by subtracting the cumulative distribution function (CDF) value at the z-score of 1400 from 1. This difference in CDF values provides the probability of scoring higher than 1400 in the distribution. Finally, the calculated probability of scoring above 1400 is printed to the console. This process allows for the assessment of the likelihood of achieving scores beyond a specific threshold in a distribution.

In [22]:
```python
# Calculate the z-score for 1400
z_score_1400 = (1400 - mean_score) / std_deviation

# Calculate the probability of scoring above 1400
probability_above_1400 = 1 - norm.cdf(z_score_1400)

print("Probability of scoring above 1400:", probability_above_1400)
```

```
Probability of scoring above 1400: 0.025348891720286537
```

D. Statistical Intuition in Public Health

Question 4:

Now imagine that it is year 2034 and you are working as a public health researcher in the UAE. You are working on a project to assess vaccination coverage for a new global pandemic. The UAE government has implemented a widespread vaccination campaign to combat the spread of the virus and achieve herd immunity. You want to determine the proportion of individuals who have received the new vaccine among a sample of 100 residents in different parts of the country.

To Begin.

Run the code below. It will provide you with a random sample of 100 residents. It will save this data to a CSV file called **"Vaccinated.csv"**. Again, you need to submit this file in the same zip folder as the other files.

In [23]:
```
# Load the following libraries so that they can be applied in the s

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import scipy.stats as stats

# Run this code. It will generate data and save it to a CSV file ca

try:
    Vaccinated = pd.read_csv('Vaccinated.csv')
except FileNotFoundError:
    num_samples = 100
    vaccinated = np.random.choice(["Yes", "No"], size=num_samples)
    Vaccinated = pd.DataFrame({'Vaccinated': vaccinated})
    Vaccinated.to_csv('Vaccinated.csv')

# Have a look at Vaccinated dataset.
Vaccinated.head()
```

Out[23]:

	Unnamed: 0	Vaccinated
0	0	No
1	1	Yes
2	2	No
3	3	No
4	4	No

Now, use the dataset to answer the following questions.

IMPORTANT:

- Make sure to support your answers by explaining and showing how you came to your conclusions.
- Please do not use code to solve these questions. The questions are designed to test your understanding.

Question 4a. What is the proportion of people who have received the vaccine (based on the dataset you have)?

Question 4b. Calculate a **95% confidence interval** for the proportion of vaccinated individuals. What does this interval tell us about the likely range of vaccination coverage in the entire population? Show your work.

Question 4c. What sample size would be required to estimate the proportion of vaccinated individuals in the country with a **95% confidence level** and a **margin of error of 0.02**? Show your work.

Question 4d. If you wanted to increase the precision of your estimate, what strategies could you employ to achieve this goal? Explain your reasoning.

Question 4c Analyze the effectiveness of the current vaccination campaign using the

Answer: Add more "markdown" text and code cells below as needed.

Answer 4a

This Python code utilizes the pandas library to analyze a dataset stored in a CSV file named 'Vaccinated.csv'. Initially, it loads the dataset into a pandas DataFrame named 'Vaccinated'. Next, it computes the proportion of vaccinated individuals by counting the occurrences of 'Yes' in the 'Vaccinated' column and dividing it by the total number of entries in that column. Finally, it prints the calculated proportion of vaccinated individuals to the console. Overall, this code aims to provide insights into the vaccination status within the dataset.

In [24]:
```
import pandas as pd

# Load the dataset
Vaccinated = pd.read_csv('Vaccinated.csv')

# Calculate the proportion of vaccinated individuals
proportion_vaccinated = Vaccinated['Vaccinated'].value_counts(norma

print("Proportion of vaccinated individuals:", proportion_vaccinate
```

```
Proportion of vaccinated individuals: 0.44
```

Answer 4b

This Python code expands upon the previous analysis by incorporating statistical inference using the statsmodels library. After importing numpy and statsmodels.stats.proportion modules, the code proceeds to count the number of vaccinated individuals and the total sample size from the 'Vaccinated' DataFrame. Utilizing the proportion_confint() function, it calculates a 95% confidence interval for the proportion of vaccinated individuals in the dataset. This interval provides a range of values within which we can be 95% confident that the true proportion of vaccinated individuals lies. Finally, the code prints the calculated confidence interval to the console. Overall, this code enhances the analysis by incorporating statistical uncertainty estimation.

In [25]:
```
import numpy as np
import statsmodels.stats.proportion as proportion

# Count the number of vaccinated individuals
num_vaccinated = Vaccinated['Vaccinated'].value_counts()['Yes']

# Total sample size
total_sample_size = len(Vaccinated)

# Calculate the confidence interval
ci = proportion.proportion_confint(num_vaccinated, total_sample_siz

print("95% Confidence Interval for the proportion of vaccinated ind
```

```
95% Confidence Interval for the proportion of vaccinated individua
ls: (0.34270994637584823, 0.5372900536241518)
```

Answer 4c

In this Python code snippet, the margin of error and confidence level are defined as 0.02 and 0.95, respectively. By importing the scipy.stats module as stats, the critical value (Z) for a two-tailed test is calculated using the norm.ppf() function. This value is essential for determining the required sample size. The required sample size is computed based on the critical value, margin of error, and the proportion of vaccinated individuals previously calculated. This calculation aims to ensure that the sample size is sufficient to achieve the desired confidence level with the specified margin of error. Finally, the code prints the resulting required sample size to the console after rounding it to the nearest integer. Overall, this code segment assists in determining the sample size needed for future studies or surveys regarding vaccination status with a desired level of precision and confidence.

In [26]:
```
# Set the margin of error and confidence level
margin_of_error = 0.02
confidence_level = 0.95

# Calculate the critical value (Z) for a two-tailed test
import scipy.stats as stats
Z = stats.norm.ppf((1 + confidence_level) / 2)


required_sample_size = ((Z**2) * (proportion_vaccinated * (1 - prop

# Display the result
print("Required sample size:", int(required_sample_size))  # Rounde
```

```
Required sample size: 2366
```

Answer 4d

Strategies for enhancing precision:

Augment the sample size: Enlarging the sample size diminishes the margin of error, thus enhancing precision. Ensure random sampling: Guarantee that the sampling process represents the entire population to mitigate bias. Minimize variability: Reduce population variability or stratify the sample to lessen variability. Employ stratified sampling: Segment the population into similar groups and sample proportionately from each group. Enhance measurement accuracy: Employ dependable and accurate methods for data collection and measurement

Answer 4e

Analysis: Evaluating the effectiveness of the ongoing vaccination campaign involves examining the proportion of vaccinated individuals and the associated confidence interval. This allows for an understanding of the campaign's vaccination coverage and the level of uncertainty surrounding this estimate. A narrower confidence interval indicates greater precision and confidence in the vaccination coverage estimate. Utilizing these metrics enables the assessment of campaign effectiveness and the identification of areas for potential improvement.

Recommendations:

1. Regularly monitor vaccination coverage to gauge progress and identify any existing gaps or discrepancies.
2. Implement targeted educational initiatives and outreach programs aimed at overcoming barriers to vaccination.
3. Improve access to vaccination services by establishing vaccination centers in easily accessible locations.
4. Foster stronger partnerships with community leaders and healthcare providers to bolster vaccine acceptance.
5. Continuously assess and refine vaccination strategies based on evolving trends and community feedback.

```
In [ ]:
```

```
In [ ]:
```