QUESTION 1

In [11]:
```
import pandas as pd

# Create an empty DataFrame
df = pd.DataFrame()

# Create an Excel writer
excel_file = 'empty_sheet.xlsx'
writer = pd.ExcelWriter(excel_file, engine='xlsxwriter')

# Write the DataFrame to the Excel file
df.to_excel(writer, sheet_name='Sheet1', index=False)

# Close the Excel writer to save the file
writer.close()
```

```
Excel file "empty_sheet.xlsx" created successfully.
```

QUESTION 2

In [12]:
```
import pandas as pd

# Load the existing data from the Yoshops_Order_List.xlsx file
existing_data = pd.read_excel('C:/Users/hp/Downloads/Yoshops_Order_List.xlsx')

# Load the empty sheet into a DataFrame
df = pd.read_excel('empty_sheet.xlsx')

# Concatenate the existing data with the empty DataFrame
merged_df = pd.concat([df, existing_data], ignore_index=True)

# Create an Excel writer
excel_file = 'empty_sheet.xlsx'
writer = pd.ExcelWriter(excel_file, engine='xlsxwriter')

# Write the merged data to the Excel file
merged_df.to_excel(writer, sheet_name='Sheet1', index=False)

# Close the Excel writer to save the file
writer.close()

print(f'Datas imported into the file "{excel_file}" successfully.')
```

```
Datas imported into the file "empty_sheet.xlsx" successfully.
```

QUESTION 3

In [3]:
```
import pandas as pd

file_path = r'C:\Users\hp\Downloads\norway_new_car_sales_by_model.csv'
df = pd.read_csv(file_path, encoding='latin1')

excel_file_path = r'C:\Users\hp\Downloads\norway_new_car_sales_by_model.xlsx'
```

In [16]:
```
# Read the Excel file
excel_file_path = r'C:\Users\hp\Desktop\Yoshop\norway_new_car_sales_by_model.x
df = pd.read_excel(excel_file_path)

# Sort the values in the 'MODEL' column in ascending order
df_sorted = df.sort_values('Model', ascending=True)

# Save the sorted DataFrame to a new Excel file
sorted_excel_file_path = r'C:\Users\hp\Desktop\Yoshop\norway_new_car_sales_by_
df_sorted.to_excel(sorted_excel_file_path, index=False)

# Display the sorted DataFrame
```

Out[16]:

	Year	Month	Make	Model	Quantity	Pct
1621	2013	7	Audi	Audi A3	222	1.962518
178	2007	9	Audi	Audi A3	123	1.300000
1722	2013	11	Audi	Audi A3	147	1.200000
1748	2013	12	Audi	Audi A3	170	1.500000
157	2007	8	Audi	Audi A3	156	1.400000
...	...	...	...	...	...	...
2427	2016	3	Volvo	Volvo XC90	128	0.922523
2504	2016	5	Volvo	Volvo XC90	126	0.979478
2533	2016	6	Volvo	Volvo XC90	143	1.045245
2368	2016	2	Volvo	Volvo XC90	151	1.235477
1574	2013	5	Mercedes-Benz	Mercedes-Benz A-klasse	184	1.531802

2694 rows × 6 columns

QUESTION 4

In [18]:
```
import pandas as pd

# Read the Excel file
excel_file_path = r'C:\Users\hp\Downloads\Yoshops Survey_1786_Updated_13_March
df = pd.read_excel(excel_file_path)

# Define the category groups
category_groups = {
    'Chicken Biryani': 'Chicken Biryani',
    'Mutton Biryani': 'Mutton Biryani',
    'undefined': 'Undefined',
    'Veg Biryani': 'Veg Biryani',
    'No Answer': 'No Answer',
    "Don't Like Biryani": "Don't Like Biryani"
}

# Map the categories to the respective groups
df['Category Group'] = df['7. What is your Favourite food biryani essay?'].map

# Save the updated DataFrame to a new Excel file
grouped_excel_file_path = r'C:\Users\hp\Downloads\Yoshops Survey_1786_Grouped.
df.to_excel(grouped_excel_file_path, index=False)

# Display the updated DataFrame
df.head()
```

Out[18]:

	S.NO	Submitted Time	1. Name	3. Location , City Name	4. What class do you study in ?	5. Which Price range for Online Tution for LKG to PG Monthly Fees You like must	6. Laptop and Mobile which Price range you like most	7. What is your Favourite food biryani essay?	8. What is Price Rang for 1kg Biryani(2 Person Eat) you like must	
0	1	24-11-2022	Kavita Israni	No answer	No answer	undefined	No answer	Veg Biryani | undefined	149 | 199	Mut
1	2	24-11-2022	Kunal Anand	No answer	No answer	undefined	No answer	Chicken Biryani	149	(Stock
2	3	25-11-2022	Deepak parmal	No answer	Graduation	undefined	No answer	Chicken Biryani	99	F Investr
3	4	25-11-2022	Nidhi Gupta	No answer	LKG to STD 5 | STD 6 to STD 10	undefined	No answer	Chicken Biryani	149	Bank of
4	5	25-11-2022	Rohan Pandey	No answer	STD 6 to STD 10	undefined	No answer	Chicken Biryani	149	Bank of

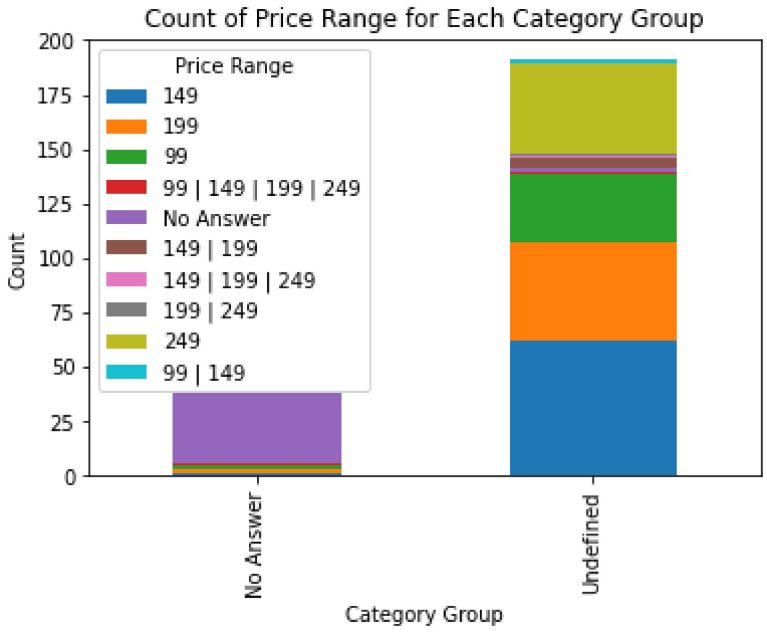
In [15]:

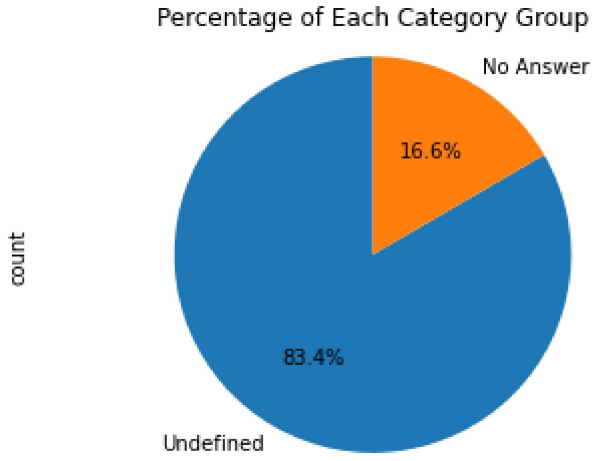
```
import matplotlib.pyplot as plt

# Read the Excel file
excel_file_path = r'C:\Users\hp\Downloads\Yoshops Survey_1786_Grouped.xlsx'
df = pd.read_excel(excel_file_path)

# Plot the bar chart
category_price_counts = df.groupby(['Category Group', '8. What is Price Rang f
category_price_counts.plot(kind='bar', stacked=True)
plt.xlabel('Category Group')
plt.ylabel('Count')
plt.title('Count of Price Range for Each Category Group')
plt.legend(title='Price Range')
plt.show()

# Plot the pie chart
category_group_counts = df['Category Group'].value_counts()
category_group_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.title('Percentage of Each Category Group')
plt.show()
```



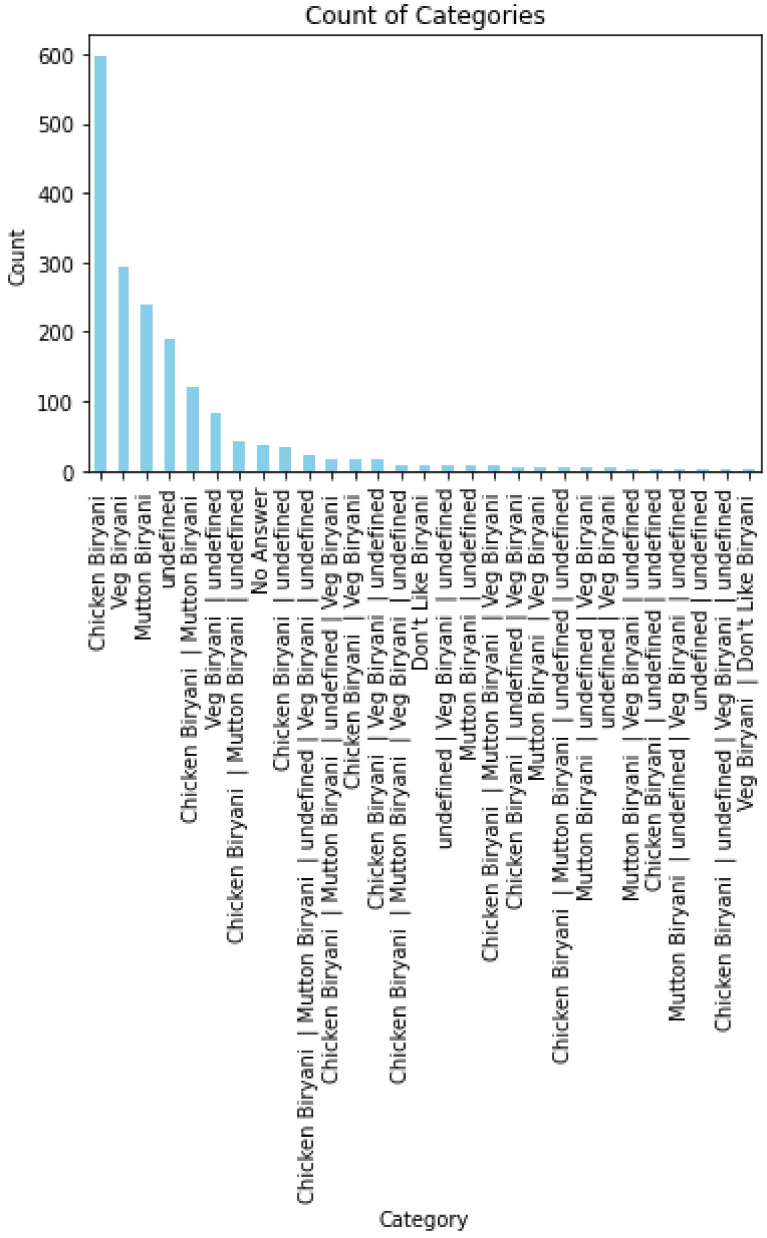

Percentage of Each Category Group
No Answer
16.6%
count
83.4%
Undefined

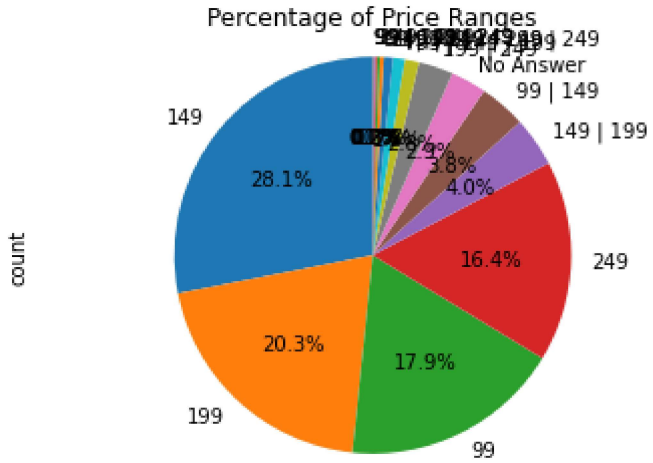
In [20]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Read the Excel file
excel_file_path = r'C:\Users\hp\Downloads\Yoshops Survey_1786_Grouped.xlsx'
df = pd.read_excel(excel_file_path)

# Plot the bar chart
category_price_counts = df['7. What is your Favourite food biryani essay?'].va
category_price_counts.plot(kind='bar', color='skyblue')
plt.xlabel('Category')
plt.ylabel('Count')
plt.title('Count of Categories')
plt.show()

# Plot the pie chart
price_range_counts = df['8. What is Price Rang for 1kg Biryani( 2 Person Eat)
price_range_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.title('Percentage of Price Ranges')
plt.show()
```


Count of Categories
Count
Category
Chicken Biryani
Veg Biryani
Mutton Biryani
undefined
Chicken Biryani | Mutton Biryani
Veg Biryani | undefined
Chicken Biryani | Mutton Biryani | undefined
No Answer
Chicken Biryani | undefined
Chicken Biryani | Veg Biryani | undefined
Chicken Biryani | Mutton Biryani | Veg Biryani | undefined
Don't Like Biryani
undefined | Veg Biryani | undefined
Mutton Biryani | undefined
Chicken Biryani | Mutton Biryani | Veg Biryani
Chicken Biryani | undefined | Veg Biryani
Mutton Biryani | Veg Biryani
Chicken Biryani | Mutton Biryani | undefined | Veg Biryani
Mutton Biryani | undefined | Veg Biryani
Mutton Biryani | Veg Biryani | undefined
Chicken Biryani | undefined | undefined
Mutton Biryani | undefined | Veg Biryani | undefined
Chicken Biryani | undefined | Veg Biryani | undefined
Veg Biryani | Don't Like Biryani


Percentage of Price Ranges
count
149
28.1%
20.3%
199
17.9%
99
16.4%
249
4.0%
149 | 199
3.8%
99 | 149
No Answer

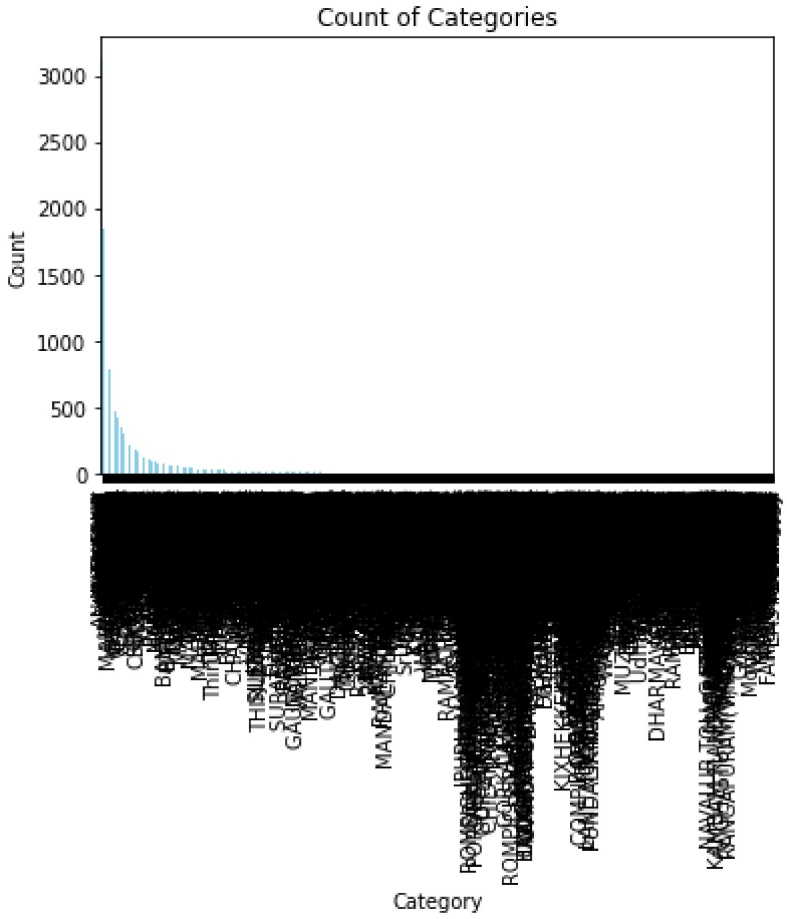
In [21]:

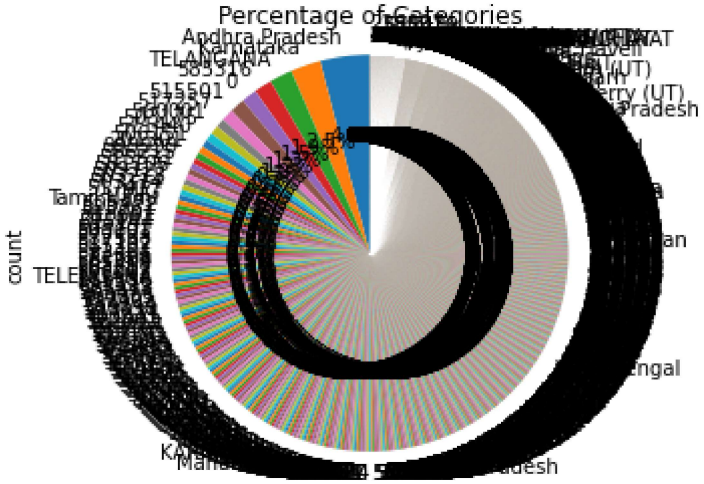
```
import matplotlib.pyplot as plt

# Read the Excel file
excel_file_path = r'C:\Users\hp\Desktop\Yoshop\empty_sheet.xlsx'
df = pd.read_excel(excel_file_path)

# Plot the bar chart
column_4_counts = df['Unnamed: 4'].value_counts()
column_4_counts.plot(kind='bar', color='skyblue')
plt.xlabel('Category')
plt.ylabel('Count')
plt.title('Count of Categories')
plt.show()

# Plot the pie chart
column_5_counts = df['Unnamed: 5'].value_counts()
column_5_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.title('Percentage of Categories')
plt.show()
```




Percentage of Categories
Andhra Pradesh
Karnataka
count

QUESTION 5

In [26]:
```
import os
import json
import re

folder_path = r'C:\Users\hp\Downloads\contact data\1657173630381_5042579018065
yoshops_contact_number = '1234567890'  # Replace with the Yoshops contact numb
number_to_exclude = '9080749858'  # Number to exclude from the results

# Get the list of files in the folder
file_list = os.listdir(folder_path)

# Iterate over the files
for file_name in file_list:
    file_path = os.path.join(folder_path, file_name)

    # Check if the file is a JSON file
    if file_name.endswith('.json'):
        with open(file_path, 'r') as file:
            json_data = json.load(file)

            # Extract phone numbers using regular expression
            phone_numbers = re.findall(r'\b(?!(?:\d{1,3}\.){3}\d{1,3}\b)(?!(?:

            # Filter out the Yoshops contact number and the number to exclude
            phone_numbers = [number for number in phone_numbers if number != y

            # Print the extracted customer contact numbers
            if phone_numbers:
                print("Customer contact numbers in", file_name)
                for number in phone_numbers:
                    print(number)
```

```
Customer contact numbers in 62c072d0-f795-11ec-9702-4165492261e7.json
8459599718
Customer contact numbers in 6bdba4c0-f74a-11ec-a0c1-0d2e72636470.json
9528692288
Customer contact numbers in 91705eb0-f7b3-11ec-9880-c1e4eddaee78.json
9123557647
Customer contact numbers in d4318e80-f773-11ec-a098-35551501145d.json
8280021014
```

In [29]:
```
import os
import json
import re

folder_path = r'C:\Users\hp\Downloads\contact data\1657173630381_5042579018065
yoshops_contact_number = '9080749858'  # Yoshops contact number to exclude

# Get the list of files in the folder
file_list = os.listdir(folder_path)

# Iterate over the files
for file_name in file_list:
    file_path = os.path.join(folder_path, file_name)

    # Check if the file is a JSON file
    if file_name.endswith('.json'):
        with open(file_path, 'r') as file:
            json_data = json.load(file)

            # Extract phone numbers using regular expression
            phone_numbers = re.findall(r'\b(?!(?:\d{1,3}\.){3}\d{1,3}\b)(?!(?:

            # Exclude Yoshops contact number
            phone_numbers = [number for number in phone_numbers if number != y

            # Print the extracted contact numbers
            if phone_numbers:
                print("Contact numbers in", file_name)
                for number in phone_numbers:
                    print(number)
```

```
Contact numbers in 05ae88c0-f871-11ec-a0b8-d9e10e7c0f2d.json
9304522101
Contact numbers in c37068b0-f80d-11ec-838e-b9a87975b9e3.json
6200028123
```

OR

In [30]:
```
import os
import json
import re

# Folder paths
folder_path_1 = r'C:\Users\hp\Downloads\contact data\1657173630381_50425790180
folder_path_2 = r'C:\Users\hp\Downloads\contact data\1657173630381_50425790180

# Yoshops contact number to exclude
yoshops_contact_number = '9080749858'

# Number to exclude from results
number_to_exclude = '9080749858'

# Function to extract contact numbers from JSON files
def extract_contact_numbers(file_path):
    with open(file_path, 'r') as file:
        json_data = json.load(file)

        # Extract phone numbers using regular expression
        phone_numbers = re.findall(r'\b(?!(?:\d{1,3}\.){3}\d{1,3}\b)(?!(?:\d{1

        # Filter out the Yoshops contact number and the number to exclude
        phone_numbers = [number for number in phone_numbers if number != yoshc

        return phone_numbers

# Process files in the first folder path
file_list_1 = os.listdir(folder_path_1)
for file_name in file_list_1:
    file_path = os.path.join(folder_path_1, file_name)

    # Check if the file is a JSON file
    if file_name.endswith('.json'):
        phone_numbers = extract_contact_numbers(file_path)

        # Print the extracted customer contact numbers
        if phone_numbers:
            print("Customer contact numbers in", file_name)
            for number in phone_numbers:
                print(number)

# Process files in the second folder path
file_list_2 = os.listdir(folder_path_2)
for file_name in file_list_2:
    file_path = os.path.join(folder_path_2, file_name)

    # Check if the file is a JSON file
    if file_name.endswith('.json'):
        phone_numbers = extract_contact_numbers(file_path)

        # Print the extracted contact numbers
        if phone_numbers:
            print("Contact numbers in", file_name)
            for number in phone_numbers:
                print(number)
```

```
Customer contact numbers in 62c072d0-f795-11ec-9702-4165492261e7.json
8459599718
Customer contact numbers in 6bdba4c0-f74a-11ec-a0c1-0d2e72636470.json
9528692288
Customer contact numbers in 91705eb0-f7b3-11ec-9880-c1e4eddaee78.json
9123557647
Customer contact numbers in d4318e80-f773-11ec-a098-35551501145d.json
8280021014
Contact numbers in 05ae88c0-f871-11ec-a0b8-d9e10e7c0f2d.json
9304522101
Contact numbers in c37068b0-f80d-11ec-838e-b9a87975b9e3.json
6200028123
```

QUESTION 6

In [37]:
```
import pandas as pd

# File path
file_path = r'C:\Users\hp\Downloads\Yoshops Survey_1786_Updated_13_March.xlsx'

# Load the Excel file into a pandas DataFrame
df = pd.read_excel(file_path)

# Define the column name
column_name = '8. What is Price Rang for 1kg Biryani( 2 Person Eat) you like m

# Extract only the first number and eliminate the second number using regular
df[column_name] = df[column_name].astype(str).str.extract(r'^(\d+)')

# Print the modified column
print(df[column_name])
```

```
0       149
1       149
2        99
3       149
4       149
       ...
1779    199
1780    199
1781    249
1782    199
1783    249
Name: 8. What is Price Rang for 1kg Biryani( 2 Person Eat) you like must , Le
ngth: 1784, dtype: object
```

In [ ]:

In [ ]:

In [*]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

while True:
    choice = input("Enter a number from 1 to 10 (or 'q' to quit): ")

    if choice == "1":

        value_counts = df['stars'].value_counts()
        plt.figure(figsize=(10, 6))
        ax = sns.barplot(x=value_counts.index, y=value_counts.values)
        plt.xlabel('Star Rating')
        plt.ylabel('Count')
        plt.title('Distribution of Star Ratings')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()
    elif choice == "2":

        # Load the Excel sheet into a DataFrame
        file_path = r"C:\Users\hp\Downloads\orders_2016-2020_Dataset.csv"
        df = pd.read_csv(file_path)

        # Perform exploratory data analysis on the 'Payment Method' column
        # Calculate value counts
        value_counts = df['Payment Method'].value_counts()

        # Plot a bar chart of the value counts
        plt.figure(figsize=(10, 6))
        ax = sns.barplot(x=value_counts.index, y=value_counts.values)
        plt.xlabel('Payment Method')
        plt.ylabel('Count')
        plt.title('Distribution of Payment Methods')
        plt.xticks(rotation=45, ha='right')  # Rotate and align x-axis labels
        plt.tight_layout()  # Adjust layout to prevent label cutoff
        plt.show()

    elif choice == "3":
                ##Load the Excel sheet into a DataFrame
        file_path = r"C:\Users\hp\Downloads\orders_2016-2020_Dataset.csv"
        df = pd.read_csv(file_path)

        ##Group the data by shipping state and count the number of unique cust
        state_customers = df['Shipping State'].value_counts()

        ##Plot a bar chart of the number of customers by state
        plt.figure(figsize=(10, 6))
        ax = sns.barplot(x=state_customers.index, y=state_customers.values)
        plt.xlabel('State')
        plt.ylabel('Number of Customers')
        plt.title('Number of Customers by State')
        plt.xticks(rotation=45, ha='right') # Rotate and align x-axis labels
        plt.tight_layout() # Adjust layout to prevent label cutoff
        plt.show()

    elif choice == "4":
```

```
        # Load the Excel sheet into a DataFrame
        file_path = r"C:\Users\hp\Downloads\orders_2016-2020_Dataset.csv"
        df = pd.read_csv(file_path)

        # Perform exploratory data analysis on the 'Shipping City' column
        # Calculate value counts
        city_counts = df['Shipping City'].value_counts().head(10)

        # Plot a bar chart of the top 10 cities with the most customers
        plt.figure(figsize=(10, 6))
        ax = sns.barplot(x=city_counts.index, y=city_counts.values)
        plt.xlabel('City')
        plt.ylabel('Number of Customers')
        plt.title('Top 10 Cities with the Most Customers')
        plt.xticks(rotation=45, ha='right')  # Rotate and align x-axis labels
        plt.tight_layout()  # Adjust layout to prevent label cutoff
        plt.show()

    elif choice == "5":

        # Load the Excel sheet into a DataFrame
        file_path = r"C:\Users\hp\Downloads\review_dataset.csv"
        df = pd.read_csv(file_path)

        # Select the top 10 categories
        top_categories = df['category'].value_counts().head(10)

        # Plot the count of each category as a count plot
        plt.figure(figsize=(10, 6))
        sns.countplot(data=df, x='category', order=top_categories.index, palet
        plt.xlabel('Category')
        plt.ylabel('Count')
        plt.title('Count of Top 10 Categories')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()

    elif choice == "6":
        # Load the Excel sheet into a DataFrame
        file_path = r"C:\Users\hp\Downloads\review_dataset.csv"
        df = pd.read_csv(file_path)

        # Calculate the value counts for each category
        category_counts = df['category'].value_counts()

        # Get the top category
        top_category = category_counts.index[0]

        # Plot a bar chart of the top category
        plt.figure(figsize=(8, 6))
        category_counts.plot(kind='bar')
        plt.xlabel('Category')
        plt.ylabel('Count')
        plt.title(f'Top Category: {top_category}')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()
```

```
elif choice == "7":
    # Load the Excel sheet into a DataFrame
    file_path = r"C:\Users\hp\Downloads\orders_2016-2020_Dataset.csv"
    df = pd.read_csv(file_path)

    # Convert 'Order Date and Time Stamp' column to datetime
    df['Order Date and Time Stamp'] = pd.to_datetime(df['Order Date and Ti

    # Extract year and month from the 'Order Date and Time Stamp' column
    df['Year'] = df['Order Date and Time Stamp'].dt.year
    df['Month'] = df['Order Date and Time Stamp'].dt.month

    # Group the data by year and month and count the number of orders
    orders_per_month = df.groupby(['Year', 'Month']).size().reset_index(na

    # Plot a line chart to visualize the number of orders per year and per
    plt.figure(figsize=(10, 6))

    # Assign different colors to each year
    colors = ['red', 'blue', 'green', 'orange', 'purple']
    years = orders_per_month['Year'].unique()

    for i, year in enumerate(years):
        year_data = orders_per_month[orders_per_month['Year'] == year]
        plt.plot(year_data['Month'], year_data['Number of Orders'], marker

    plt.xlabel('Month')
    plt.ylabel('Number of Orders')
    plt.title('Number of Orders per Year and Month')

    # Label x-axis with months
    months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep
    plt.xticks(range(1, 13), labels=months)

    plt.legend([str(year) for year in years])
    plt.grid(True)
    plt.tight_layout()
    plt.show()

elif choice == "8":
    # Load the Excel sheet into a DataFrame
    file_path = r"C:\Users\hp\Downloads\review_dataset.csv"
    df = pd.read_csv(file_path)

    # Replace rows without any values in the 'status' column with 'not rev
    df['status'].fillna('not reviewed', inplace=True)

    # Calculate the value counts for the 'status' column
    status_counts = df['status'].value_counts()

    # Plot a pie chart to visualize the distribution of reviews
    plt.figure(figsize=(8, 6))
    plt.pie(status_counts, labels=status_counts.index, autopct='%1.1f%%',
    plt.title('Review Status')
    plt.axis('equal')
    plt.show()
```

```
        # Plot a bar chart to visualize the number of reviews
        plt.figure(figsize=(8, 6))
        status_counts.plot(kind='bar')
        plt.xlabel('Review Status')
        plt.ylabel('Count')
        plt.title('Review Status')
        plt.xticks(rotation=0)
        plt.show()

        # Load the Excel sheet into a DataFrame
        file_path = r"C:\Users\hp\Downloads\review_dataset.csv"
        df = pd.read_csv(file_path)

        # Define the star ratings to analyze
        star_ratings = ['5.0 star rating', '4.9 star rating', '4.6 star rating
                        '4.0 star rating', '3.0 star rating', '2.3 star rating
                        '4.7 star rating', '3.3 star rating', '4.2 star rating

        # Filter the DataFrame for the specified star ratings
        filtered_df = df[df['stars'].isin(star_ratings)]

        # Group the data by 'product_name' and 'stars' and count the occurrenc
        rating_counts = filtered_df.groupby(['product_name', 'stars']).size().

        # Print the results for each star rating
        for rating in star_ratings:
            products = rating_counts[rating]
            top_product = products.idxmax()
            print("Product with the highest count of", rating, ":", top_produc

        # Plot the results as a pie chart
        plt.figure(figsize=(8, 8))
        explode = [0.1] * len(star_ratings)  # Explode each slice for emphasis

        # Create labels with product name and star rating
        labels = [f"{rating}\n({rating_counts[rating].idxmax()})" for rating i

        plt.pie(rating_counts.sum(), labels=labels, autopct='%1.1f%%', explode
        plt.title('Distribution of Star Ratings for Products')
        plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a
        plt.show()

    elif choice== "9":

        # Load the CSV file into a DataFrame
        file_path = r"C:\Users\hp\Downloads\orders_2016-2020_Dataset.csv"
        df = pd.read_csv(file_path)

        # Convert the 'Order Date and Time Stamp' column to datetime format
        df['Order Date and Time Stamp'] = pd.to_datetime(df['Order Date and Ti

        # Extract the date from the 'Order Date and Time Stamp' column
        df['Order Date'] = df['Order Date and Time Stamp'].dt.date

        # Group the data by 'Order Date' and count the occurrences
        order_counts = df.groupby('Order Date').size()
```

```
        # Plot the order count for each day as a bar chart
        plt.figure(figsize=(12, 6))
        order_counts.plot(kind='bar', color='skyblue')
        plt.xlabel('Order Date')
        plt.ylabel('Order Count')
        plt.title('Order Count for Each Day')
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

    elif choice == "10":
        # Calculate the average rating for each product
        product_ratings = df.groupby('product_name')['stars'].mean()

        # Get the top 10 products with the highest average rating
        top_products = product_ratings.nlargest(10)

        # Plot a bar chart of the top 10 products with the highest average rat
        plt.figure(figsize=(10, 6))
        ax = sns.barplot(x=top_products.index, y=top_products.values)
        plt.xlabel('Product')
        plt.ylabel('Average Rating')
        plt.title('Top 10 Products with the Highest Average Rating')
        plt.xticks(rotation=45, ha='right')  # Rotate and align x-axis labels
        plt.tight_layout()  # Adjust layout to prevent label cutoff
        plt.show()
        break

    else:
        print("Invalid choice. Please enter a number from 1 to 10.")
```

In [ ]: