PROJECT TITLE: MARKETING CAMPAIGN PERFORMANCE INSIGHTS

Coding Mentor: Janani Lakshmi Narayanan

Student Name : Parameswari Manthiramoorthi

DECEMBER 14, 2024

ENTRI

B4-Batch

PROJECT TITLE: MARKETING CAMPAIGN PERFORMANCE INSIGHTS

CONTENTS

PROBLEM STATEMENT:

In the highly competitive landscape of digital marketing, evaluating the success of marketing campaigns is crucial for optimizing return on investment (ROI) and improving overall performance. Despite having access to extensive data, a comprehensive analysis of key metrics—such as conversion rates, acquisition costs, and ROI—across various campaign types, channels, and audience segments remains a challenge. This project aims to analyze these metrics to uncover actionable insights, identify factors driving campaign success, and provide recommendations to enhance future marketing strategies.

DATASET DETAILS:

- **Source:** Marketing Campaign Dataset
- **Data Dictionary:**

Campaign_ID	**Unique identifier for each campaign.**
Company	Organization running the campaign.
Campaign_Type	Type of marketing effort (e.g., email, social media, influencer).
Target_Audience	Specific demographic targeted.
Duration	Campaign duration in days.
Channels_Used	Platforms or mediums used (e.g., email, social media).
Conversion_Rate	Percentage of successful actions.
Acquisition_Cost	Cost per customer acquisition.
ROI	Return on investment.
Location	Geographical area of execution.
Language	Language of campaign content.
Clicks	Total user interactions.
Impressions	Total views or displays.
Engagement_Score	A score (1–10) representing campaign interaction.
Customer_Segment	Group or category of customers targeted.
Date	Date on which the campaign occurred.

PROJECT STEPS AND OBJECTIVES:

1. LOAD THE DATASET:

1.1. Import the dataset from the provided CSV file using pandas.

```
# 1) Load the Dataset
# ➢ Read the marketing campaign data from the CSV file into a pandas DataFrame.
data = pd.read_csv('marketing_campaign.csv')
data
```

	Campaign_ID	Company	Campaign_Type	Target_Audience	Duration	Channel_Used	Conversion_Rate	Acquisition_Cost	ROI	Location	Language	Clicks	Impressions	Engagement_Score	Customer_Segment	Date
0	1	TechCorp	Email	Women 25-34	30 days	Facebook	5.294193574	9344	62.94	Houston	English	3045	67836	5	Tech Enthusiasts	01/01/2023
1	2	Innovate Ind	Influencer	Women 35-44	45 days	Google Ads	3.326375241	8783	10.67	Washington	German	1944	66361	4	Foodies	01/01/2023
2	3	NexGen Syst	Social Media	Women 25-34	45 days	Instagram	4.056375241	9111	73.2	Miami	Spanish	3156	86240	8	Fashionistas	01/01/2023
3	4	Innovate Ind	Email	Women 25-34	45 days	Instagram	4.496375241	7420	60.92	Seattle	Spanish	2388	58251	6	Foodies	01/01/2023
4	5	Data Tech So	Influencer	Men 25-34	30 days	Google Ads	4.405929621	2146	138.82	Chicago	English	1025	34407	5	Tech Enthusiasts	01/01/2023
5	6	NexGen Syst	Social Media	Women 35-44	15 days	Google Ads	5.481448637	9416	-20.35	Washington	English	1500	52838	8	Outdoor Adventurers	01/01/2023
6	7	Alpha Innova	Display	Women 35-44	30 days	Website	3.916375241	1065	1122.07	Washington	Spanish	2603	73970	5	Outdoor Adventurers	01/01/2023
7	8	NexGen Syst	Influencer	Women 25-34	30 days	Facebook	4.516375241	5634	10.49	Los Angeles	Mandarin	1245	30219	5	Foodies	01/01/2023
8	9	Innovate Ind	Email	Women 25-34	45 days	Email	5.860946252	4110	-11.68	Washington	English	726	33932	5	Tech Enthusiasts	01/01/2023
9	10	NexGen Syst	Email	Men 25-34	60 days	Google Ads	4.746375241	7923	-38.03	Washington	Mandarin	982	22593	4	Tech Enthusiasts	01/01/2023
10	11	Data Tech So	Display	Women 25-34	45 days	Email	5.286375241	2090	497.61	Atlanta	Mandarin	2498	51097	5	Health & Wellness	01/01/2023

2. DESCRIPTIVE ANALYSIS:

2.1. Print the first few rows to get an overview of the data.

```
# ➢ Print the first few rows of the dataset to get an overview of the data.
print("Print the first 10 rows from the dataset")
data.head(10)
```

2.2. Determine the number of rows and columns.

```
[12]: # ➢ Obtain the number of rows and columns in the dataset.
      print("Number of rows and columns in the dataset (Rows , Columns): ",data.shape)

      Number of rows and columns in the dataset (Rows , Columns):  (22029, 16)
```

2.3. Summarize the dataset with data types, non-null counts, and descriptive statistics.

```
[14]: # ➢ Get a concise summary of the dataset, including the data types and non-null values.
      print("concise summary of the dataset")
      print("******************************")
      data.info()
```

```
concise summary of the dataset
******************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22029 entries, 0 to 22028
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Campaign_ID       22029 non-null  int64
 1   Company           22029 non-null  object
 2   Campaign_Type     22029 non-null  object
 3   Target_Audience   22029 non-null  object
 4   Duration          22029 non-null  object
 5   Channel_Used      22029 non-null  object
 6   Conversion_Rate   22029 non-null  float64
 7   Acquisition_Cost  22029 non-null  int64
 8   ROI               22029 non-null  float64
 9   Location          22029 non-null  object
 10  Language          22029 non-null  object
 11  Clicks            22029 non-null  int64
 12  Impressions       22029 non-null  int64
 13  Engagement_Score  22029 non-null  int64
 14  Customer_Segment  22029 non-null  object
 15  Date              22029 non-null  object
dtypes: float64(2), int64(5), object(9)
memory usage: 2.7+ MB
```

```
[16]: # ➢ Generate descriptive statistics for numerical columns.
      print("Descriptive statistics for numerical columns")
      print("*******************************************")
      data.describe()
```

```
Descriptive statistics for numerical columns
*******************************************
```

[16]:

	Campaign_ID	Conversion_Rate	Acquisition_Cost	ROI	Clicks	Impressions	Engagement_Score
count	22029.000000	22029.000000	22029.000000	22029.000000	22029.000000	22029.000000	22029.000000
mean	11015.000000	4.757232	5522.740842	182.863648	2223.807572	50610.402787	6.582323
std	6359.368876	0.960393	2597.666260	301.619721	1394.166380	28542.979123	1.458804
min	1.000000	2.015723	1000.000000	-98.300000	30.000000	1001.000000	4.000000
25%	5508.000000	4.130705	3286.000000	-4.080000	1067.000000	25804.000000	5.000000
50%	11015.000000	4.761527	5525.000000	93.650000	2088.000000	50858.000000	7.000000
75%	16522.000000	5.429335	7766.000000	247.310000	3212.000000	75165.000000	8.000000
max	22029.000000	7.469907	9999.000000	3109.790000	6887.000000	99999.000000	9.000000

2.4. Count unique values in critical columns (e.g., Campaign_ID, Location).

Data exploration:

[19]:
```
# ➢ Print the number of unique Campaign_ID values in the dataset.
print("Number of unique Campaign_ID values in the dataset: ",data['Campaign_ID'].nunique())
```

```
Number of unique Campaign_ID values in the dataset:  22029
```

2.5. List the unique values of the Location and Customer_Segment columns.

[21]:
```
#➢ List the unique values of the Location and Customer_Segment columns.
print("Unique values of the Location column: \n",data['Location'].unique())
print("*****************************************")
print("Unique values of the Customer_Segment column:\n ",data['Customer_Segment'].unique())
print("*****************************************")
```

```
Unique values of the Location column:
 ['Houston' 'Washington, D.C.' 'Miami' 'Seattle' 'Chicago' 'Los Angeles'
 'Atlanta' 'Dallas' 'New York' 'San Francisco']
*****************************************
Unique values of the Customer_Segment column:
  ['Tech Enthusiasts' 'Foodies' 'Fashionistas' 'Outdoor Adventurers'
 'Health & Wellness']
*****************************************
```

2.6. Count the occurrences of each category in the Campaign_Type and Channel_Used and columns.

[23]:
```
#➢ Count the occurrences of each category in the Campaign_Type and
# Channel_Used and columns
print("Count the occurrences of each category in the Campaign_Type and Channel_Used columns:")
print("*********************************************************************************")
print(data['Campaign_Type'].value_counts())
print("*********************************************************************************")
print(data['Channel_Used'].value_counts())
```

```
Count the occurrences of each category in the Campaign_Type and Channel_Used columns:
*********************************************************************************
Campaign_Type
Display         4450
Search          4441
Social Media    4412
Email           4388
Influencer      4338
Name: count, dtype: int64
*********************************************************************************
Channel_Used
Facebook      3742
Google Ads    3694
Website       3688
Instagram     3649
YouTube       3632
Email         3624
Name: count, dtype: int64
```

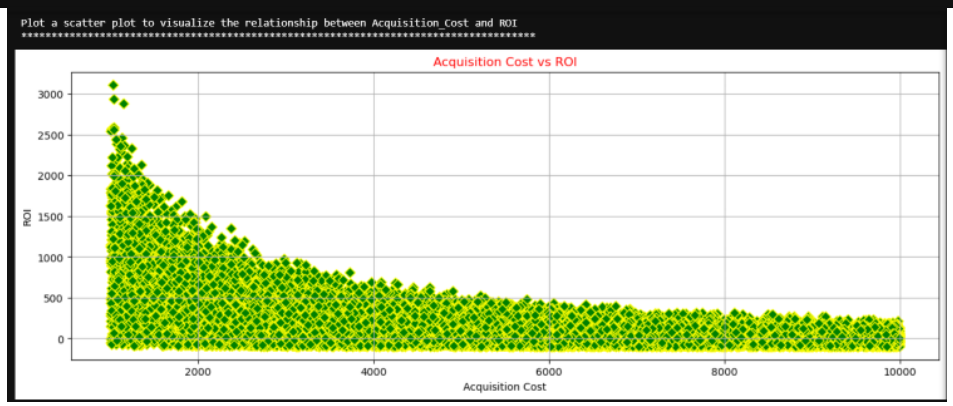
3. EXPLORATORY DATA ANALYSIS (EDA) AND VISUALIZATION:

3.1 CAMPAIGN PERFORMANCE:

3.1.1 Scatter plot: Acquisition_Cost vs. ROI.

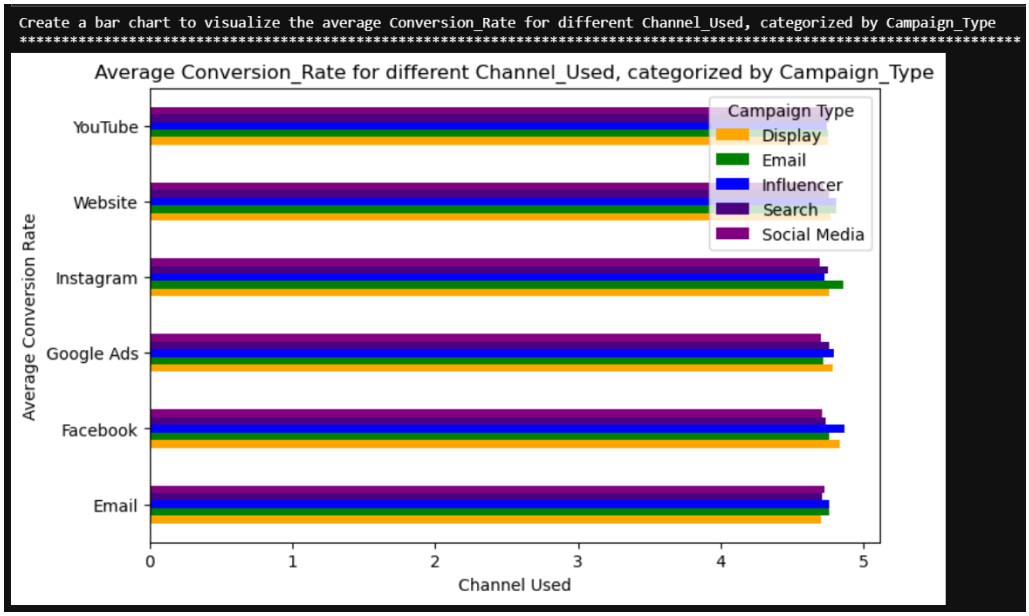
```
[27]: # ➢ Plot a scatter plot to visualize the relationship between
      # Acquisition_Cost and ROI.
      print("Plot a scatter plot to visualize the relationship between Acquisition_Cost and ROI")
      print("*********************************************************************************")
      plt.figure(figsize=(15, 5))
      plt.grid(True)
      plt.title('Acquisition Cost vs ROI',color = "RED" )
      plt.scatter(data['Acquisition_Cost'], data['ROI'],marker = 'D',s=50 ,edgecolors='yellow', facecolor ='green' )
      plt.xlabel('Acquisition Cost')
      plt.ylabel('ROI')
      plt.show()
```

```
Plot a scatter plot to visualize the relationship between Acquisition_Cost and ROI
*********************************************************************************
```



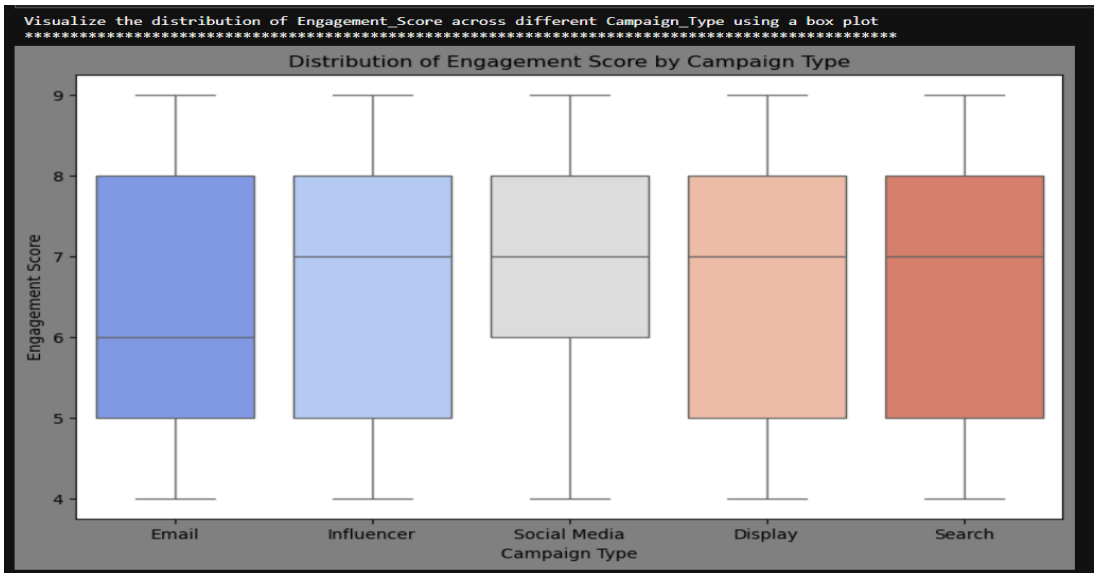
3.1.2 Bar chart: Average Conversion_Rate by Channels_Used, grouped by Campaign_Type.

```
[29]: # ➢ Create a bar chart to visualize the average Conversion_Rate for different Channel_Used, categorized by Campaign_Type.
      print("Create a bar chart to visualize the average Conversion_Rate for different Channel_Used, categorized by Campaign_Type")
      print("*********************************************************************************************************************")
      avg_conversion = data.groupby(['Channel_Used', 'Campaign_Type'])['Conversion_Rate'].mean().unstack()
      avg_conversion.plot(kind='barh', figsize=(10, 6), color =['orange', 'green', 'blue', 'indigo', 'purple'],
                          title="Average Conversion_Rate for different Channel_Used, categorized by Campaign_Type")
      plt.xlabel('Channel Used')
      plt.ylabel('Average Conversion Rate')
      plt.legend(title="Campaign Type")
      plt.show()
```



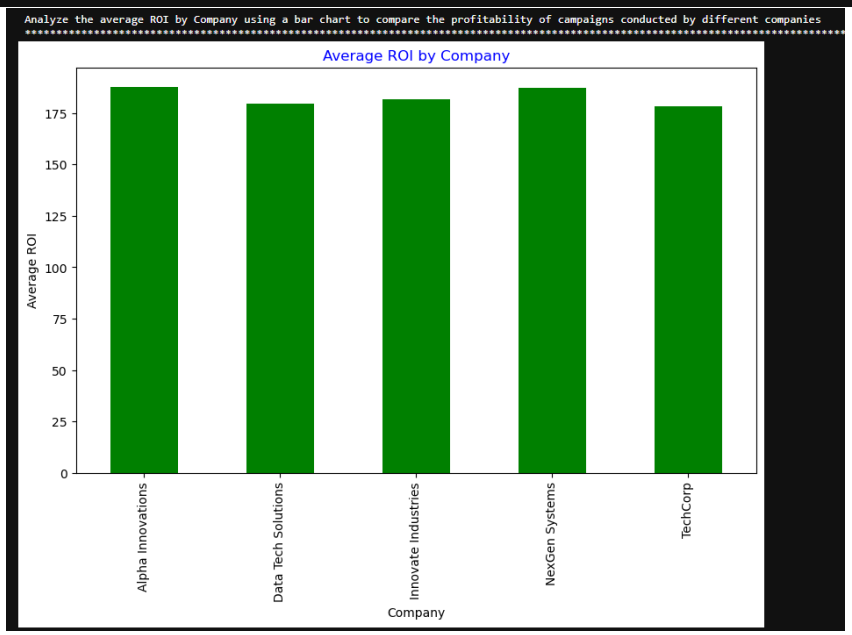
3.1.3 Box plot: Engagement_Score distribution by Campaign_Type.

```
[39]: # ➢ Visualize the distribution of Engagement_Score across different Campaign_Type using a box plot.
print("Visualize the distribution of Engagement_Score across different Campaign_Type using a box plot")
print("*****************************************************************************************")
plt.figure(figsize=(10, 6),facecolor='gray')
sns.boxplot(x='Campaign_Type', y='Engagement_Score', data=data,palette='coolwarm',hue='Campaign_Type')
plt.xlabel('Campaign Type')
plt.ylabel('Engagement Score')
plt.title('Distribution of Engagement Score by Campaign Type')
plt.show()
```



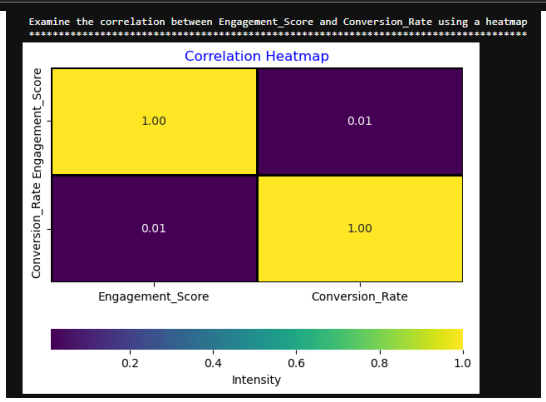
3.1.4 Bar chart: Average ROI by Company.

```
•[41]: # ➢ Analyze the average ROI by Company using a bar chart to compare the profitability of campaigns conducted by different companies.
print("Analyze the average ROI by Company using a bar chart to compare the profitability of campaigns conducted by different companies")
print("**********************************************************************************************************************************")
avg_roi_by_company = data.groupby('Company')['ROI'].mean()
plt.figure(figsize=(10, 6))
avg_roi_by_company.plot(kind='bar', )
plt.xlabel('Company')
plt.ylabel('Average ROI')
plt.title('Average ROI by Company',color='blue')
plt.show()
```



3.1.5 Heatmap: Correlation between Engagement_Score and Conversion_Rate.

```
•[43]: # ➢ Examine the correlation between Engagement_Score and Conversion_Rate using a heatmap.
print("Examine the correlation between Engagement_Score and Conversion_Rate using a heatmap")
print("**************************************************************************************")
correlation_matrix = data[['Engagement_Score', 'Conversion_Rate']].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='viridis', fmt=".2f",linewidths=2,linecolor='black',
            cbar_kws={'label': 'Intensity', 'orientation': 'horizontal'})
plt.title('Correlation Heatmap',color = 'blue')
plt.show()
```

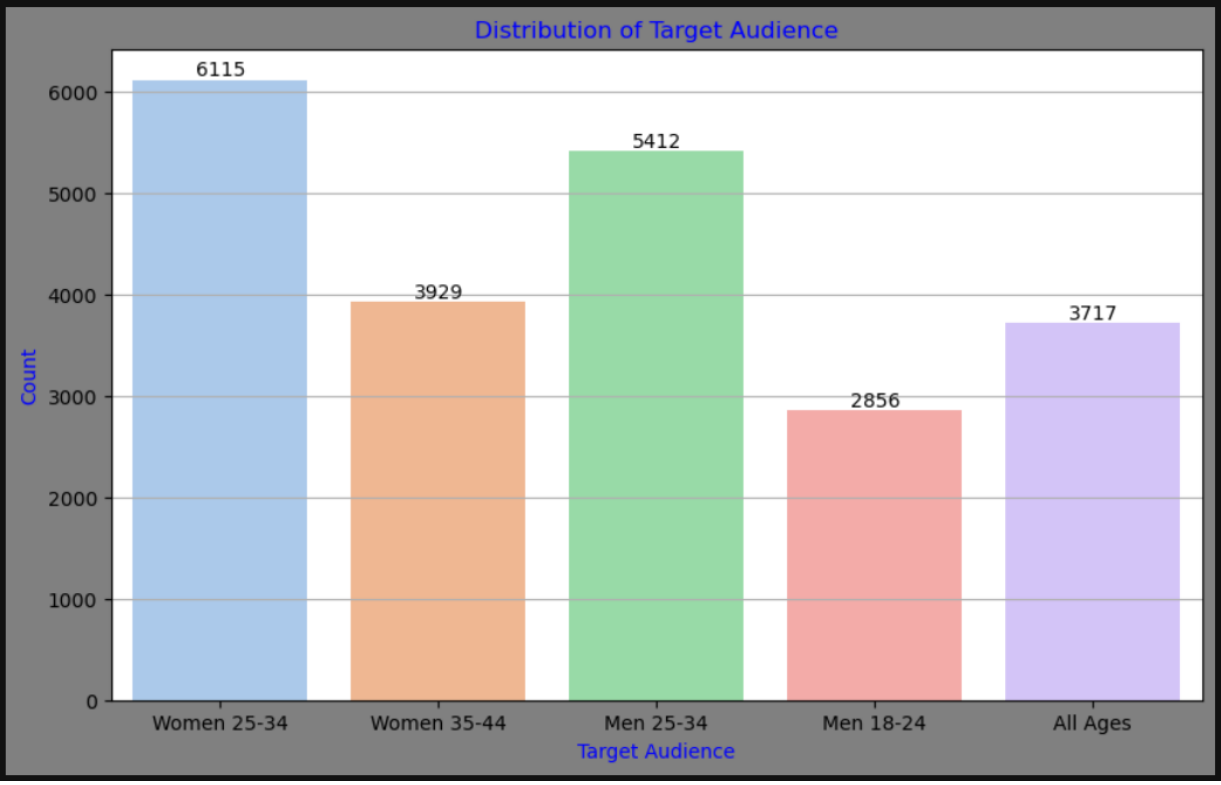


3.2 CUSTOMER SEGMENTATION:

3.2.1 Count plot: Distribution of Target_Audience.

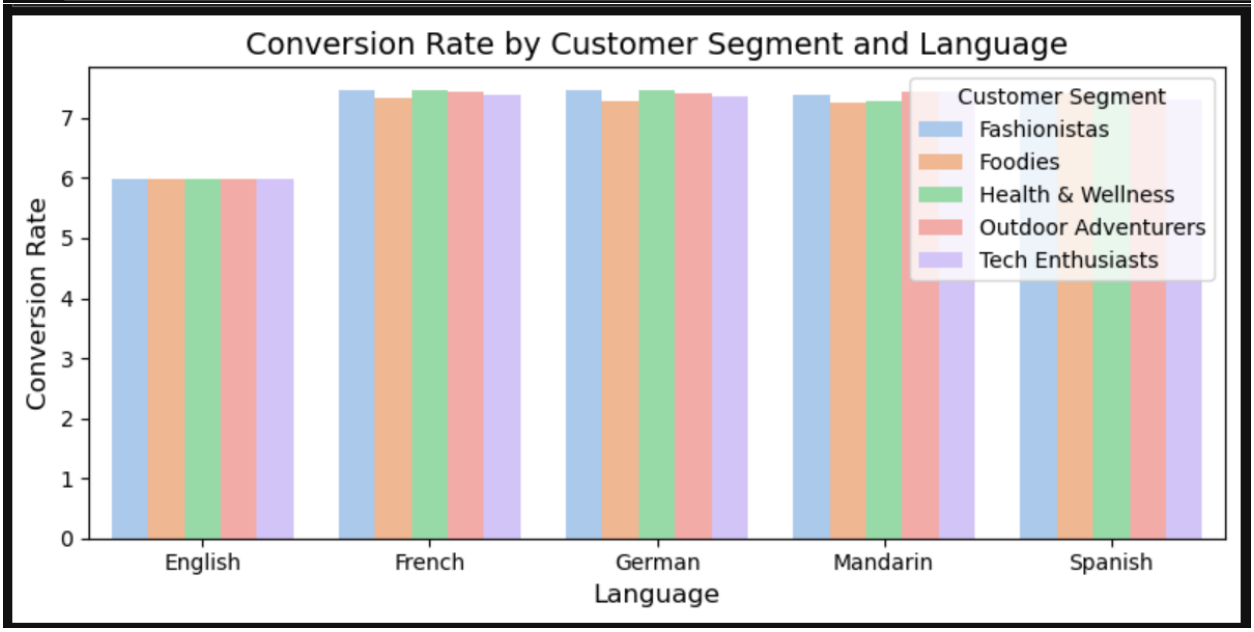
```
[46]: # ➢ Create a count plot to visualize the distribution of Target_Audience.
print("Create a count plot to visualize the distribution of Target_Audience")
print("*****************************************************************")
plt.figure(figsize=(10, 6),facecolor='gray')
plt.grid(True)
ax=sns.countplot(data=data, x='Target_Audience',hue='Target_Audience',palette='pastel')
 # Iterate count value through all bar containers
for container in ax.containers:
    ax.bar_label(container)
plt.title('Distribution of Target Audience',color = 'blue')
plt.xlabel('Target Audience',color= 'blue')
plt.ylabel('Count',color= 'blue')
plt.show()
```

```
Create a count plot to visualize the distribution of Target_Audience
*****************************************************************
```



3.2.2 Bar chart: Customer_Segment with the highest Conversion_Rate by Language.

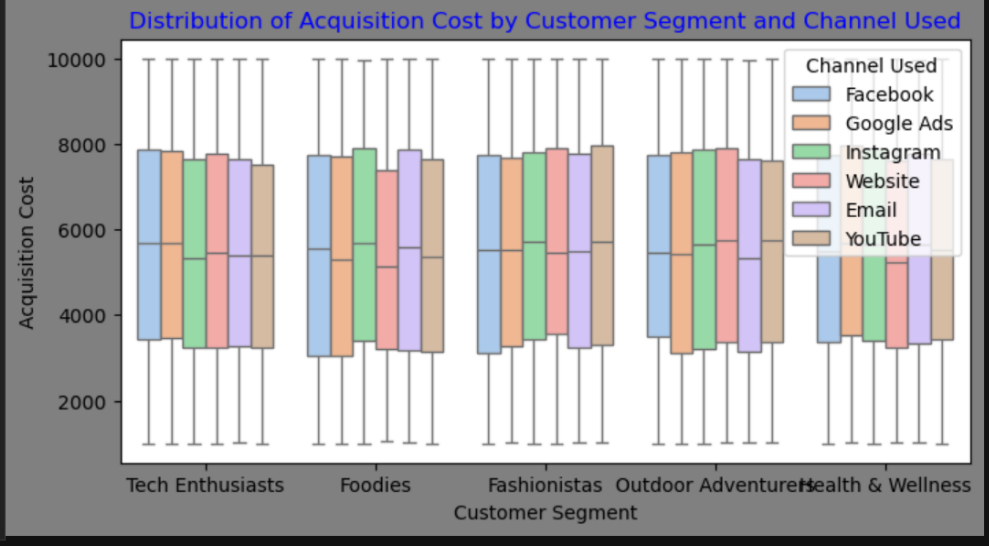
```
[53]: # ➢ Identify which Customer_Segment has the highest Conversion_Rate for each Language using a bar chart.
      # Grouping by 'Language' and 'Customer_Segment' and finding the maximum conversion rate
      conversion_rate_by_segment = data.groupby(['Language', 'Customer_Segment'])['Conversion_Rate'].max().reset_index()

      plt.figure(figsize=(8, 6), facecolor='white')
      sns.barplot(
          data=conversion_rate_by_segment,
          x='Language',
          y='Conversion_Rate',
          hue='Customer_Segment',
          palette='pastel')
      plt.xlabel('Language', fontsize=12)
      plt.ylabel('Conversion Rate', fontsize=12)
      plt.title('Conversion Rate by Customer Segment and Language', fontsize=14)
      plt.legend(title='Customer Segment')
      plt.tight_layout()
      plt.show()
```



3.2.3 Box plot: Acquisition_Cost by Customer_Segment, categorized by Channels_Used.

```
[61]: # ➢ Visualize the distribution of Acquisition_Cost across each Customer_Segment, categorized by Channel_Used, using a box plot.
      print("Visualize the distribution of Acquisition_Cost across each Customer_Segment, categorized by Channel_Used, using a box plot")
      print("*****************************************************************************************************************")
      plt.figure(figsize=(8, 4),facecolor='gray')
      sns.boxplot(x='Customer_Segment', y='Acquisition_Cost', hue='Channel_Used', data=data,palette='pastel')
      plt.xlabel('Customer Segment')
      plt.ylabel('Acquisition Cost')
      plt.title('Distribution of Acquisition Cost by Customer Segment and Channel Used',color='blue')
      plt.legend(title='Channel Used')
      plt.show()
```

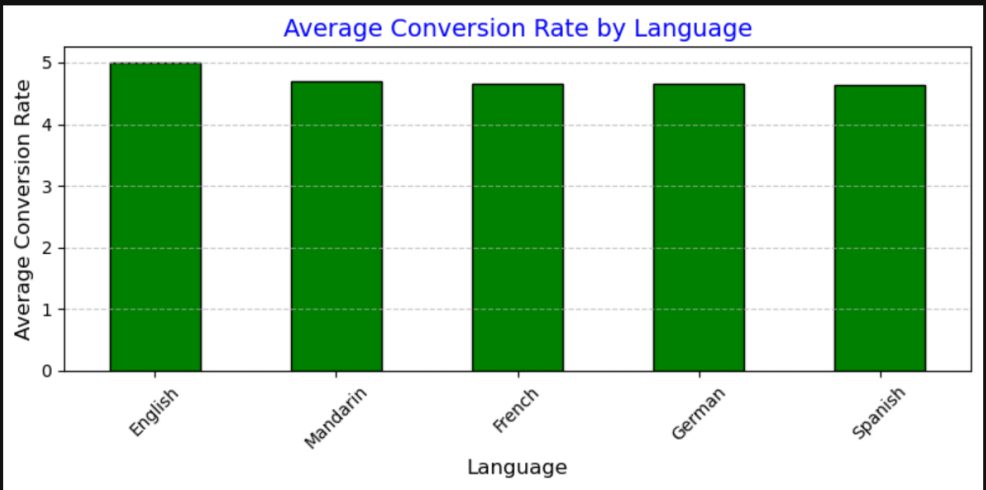
```
Visualize the distribution of Acquisition_Cost across each Customer_Segment, categorized by Channel_Used, using a box plot
**********
```



3.2.4 Bar chart: Average Conversion_Rate by Language.

```
[65]: # ➢ Analyze average Conversion_Rate by Language using a bar chart to compare the effectiveness of campaigns conducted in different languages.
print("Analyze average Conversion_Rate by Language using a bar chart to compare the effectiveness of campaigns conducted in different languages")
print("**********")
# Grouping data to calculate the average Conversion Rate by Language
avg_conversion_rate_by_language = data.groupby('Language')['Conversion_Rate'].mean()
# Creating a bar chart
plt.figure(figsize=(8, 4), facecolor='white')
avg_conversion_rate_by_language.sort_values(ascending=False).plot(kind='bar',color='green',edgecolor='black')
plt.xlabel('Language', fontsize=12)
plt.ylabel('Average Conversion Rate', fontsize=12)
plt.title('Average Conversion Rate by Language', fontsize=14, color='blue')
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()  # Adjusts layout to fit elements properly
plt.show()
```

```
Analyze average Conversion_Rate by Language using a bar chart to compare the effectiveness of campaigns conducted in different languages
**********
```

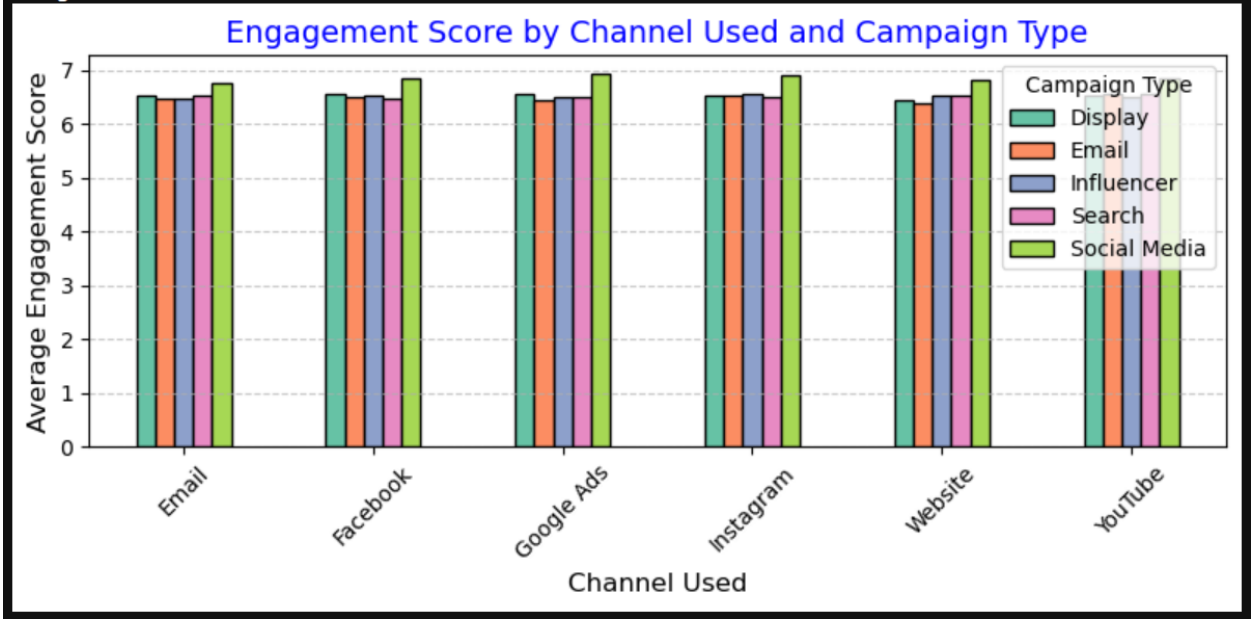


3.3 CHANNEL EFFECTIVENESS:

3.3.1 Bar chart: Engagement_Score by Channels_Used, grouped by Campaign_Type.

```
[67]: # ➢ Compare the Engagement_Score for different Channels_Used, segmented by Campaign_Type, using a bar chart.
print("Compare the Engagement_Score for different Channels_Used, segmented by Campaign_Type, using a bar chart")
print("*****************************************************************************************************")
# Grouping Engagement_Score by Channel_Used and Campaign_Type, and calculating the mean
engagement_by_channel = data.groupby(['Channel_Used', 'Campaign_Type'])['Engagement_Score'].mean().unstack()

# Setting a professional color palette
palette = sns.color_palette("Set2", n_colors=len(engagement_by_channel.columns))

# Plotting the data
plt.figure(figsize=(8, 4), facecolor='white')
engagement_by_channel.plot(kind='bar',figsize=(8, 4),color=palette,edgecolor='black')

# Adding labels, title, and legend
plt.title('Engagement Score by Channel Used and Campaign Type', fontsize=14, color='blue')
plt.xlabel('Channel Used', fontsize=12)
plt.ylabel('Average Engagement Score', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.legend(title="Campaign Type", fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

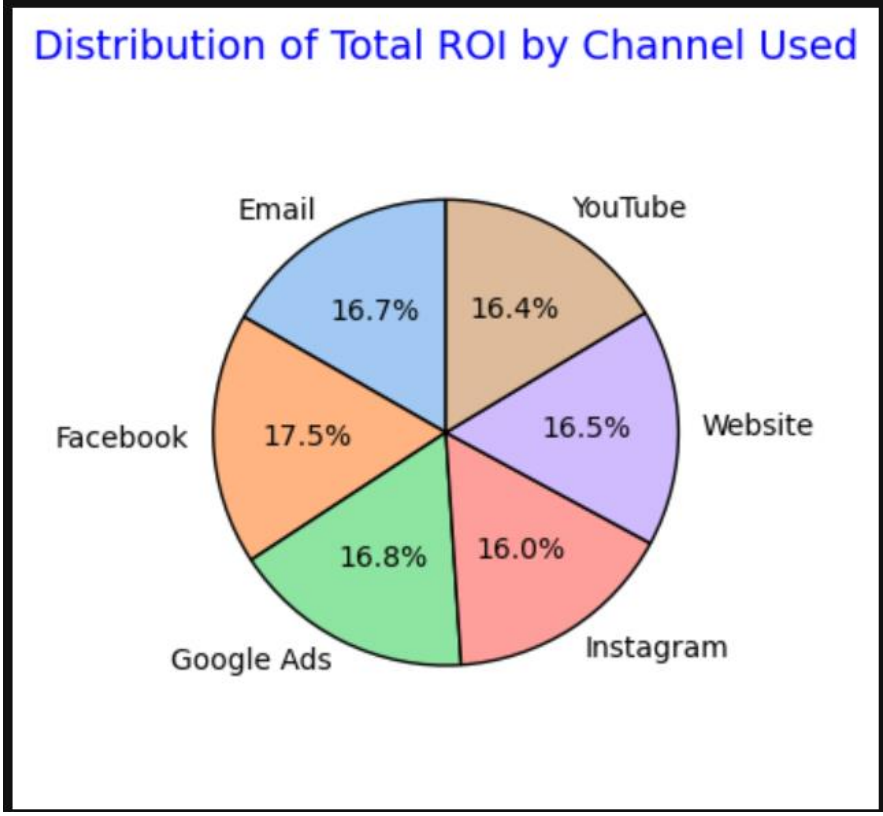
```
Compare the Engagement_Score for different Channels_Used, segmented by Campaign_Type, using a bar chart
*****************************************************************************************************
<Figure size 800x400 with 0 Axes>
```



3.3.2 Pie chart: Total ROI across Channels_Used.

```
•[69]: # ➢ Show the distribution of total ROI across different Channels_Used using a pie chart.
print("Show the distribution of total ROI across different Channels_Used using a pie chart")
print("*********************************************************************************")
# Grouping ROI by Channel_Used and calculating the total
total_roi_by_channel = data.groupby('Channel_Used')['ROI'].sum()

# Setting up the figure
plt.figure(figsize=(4, 4), facecolor='white')

# Pie chart with enhanced aesthetics
plt.pie(total_roi_by_channel,labels=total_roi_by_channel.index,
    autopct='%1.1f%%',  # Percentage format
    startangle=90,
    colors=sns.color_palette('pastel'),
    wedgeprops={'edgecolor': 'black'},  # Adding borders for clarity
    textprops={'fontsize': 10}  # Font size for labels
        )
# Adding a title
plt.title('Distribution of Total ROI by Channel Used', fontsize=14, color='blue')

# Equal aspect ratio to ensure the pie is circular
plt.axis('equal')

# Display the chart
plt.tight_layout()
plt.show()
```

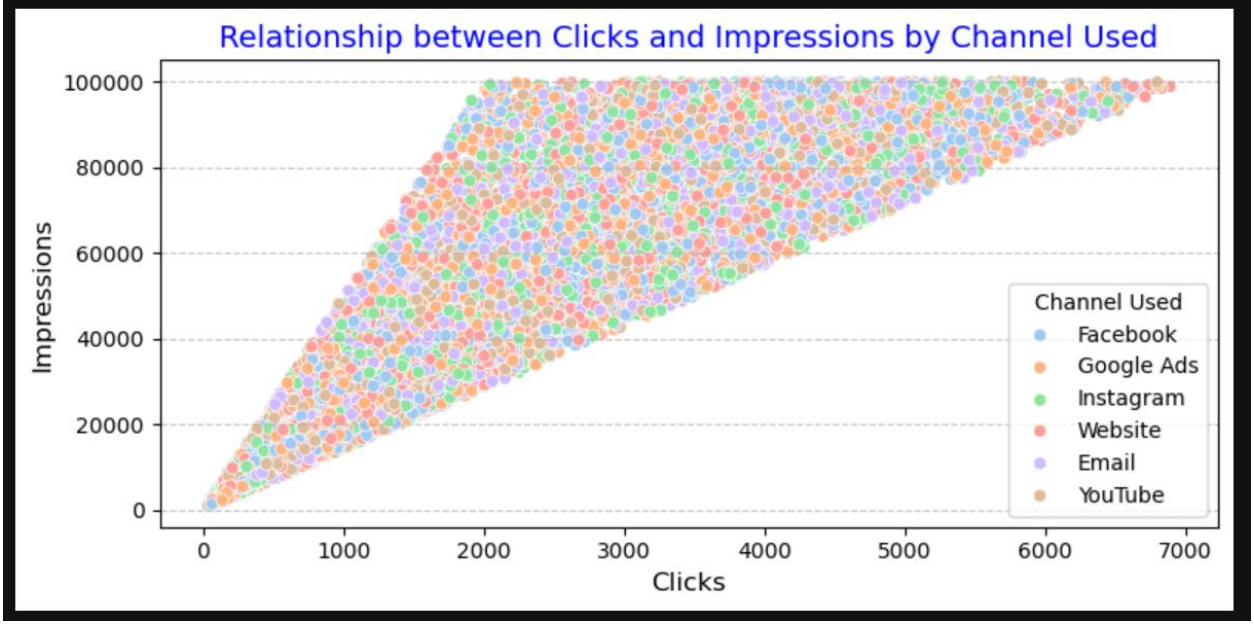
```
Show the distribution of total ROI across different Channels_Used using a pie chart
*********************************************************************************
```



3.3.3 Scatter plot: Clicks vs. Impressions by Channels_Used.

[71]:
```
# ➢ Plot a scatter plot to show the relationship between Clicks and Impressions for each Channel_Used.
print("Plot a scatter plot to show the relationship between Clicks and Impressions for each Channel_Used")
print("*****************************************************************************************")
plt.figure(figsize=(10, 6), facecolor='white')
sns.scatterplot(data=data, x='Clicks', y='Impressions', hue='Channel_Used', palette='pastel')
plt.xlabel('Clicks', fontsize=12)
plt.ylabel('Impressions', fontsize=12)
plt.title('Relationship between Clicks and Impressions by Channel Used', fontsize=14, color='blue')
plt.legend(title='Channel Used', fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
Plot a scatter plot to show the relationship between Clicks and Impressions for each Channel_Used
*****************************************************************************************
```

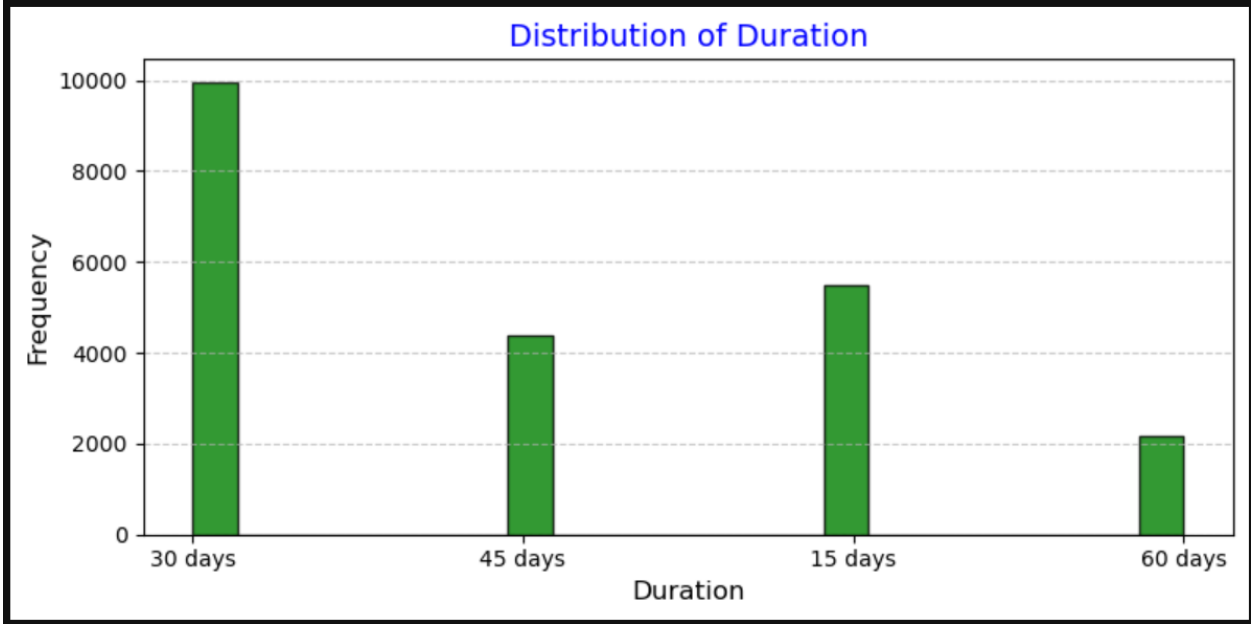


3.4 TIME-BASED ANALYSIS:

3.4.1 Histogram: Distribution of Duration.

[76]:
```
# ➢ Plot the distribution of Duration using a histogram.
print("Plot the distribution of Duration using a histogram")
print("***********************************************")
# Histogram with enhanced styling
plt.figure(figsize=(10, 6), facecolor='white')

# Plotting the histogram
plt.hist(data['Duration'], bins='auto', color='green', edgecolor='black', alpha=0.8)

# Adding labels and title
plt.xlabel('Duration', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.title('Distribution of Duration', fontsize=14, color='blue')

# Adding grid for y-axis
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Adjust layout
plt.tight_layout()

# Display the histogram
plt.show()
```

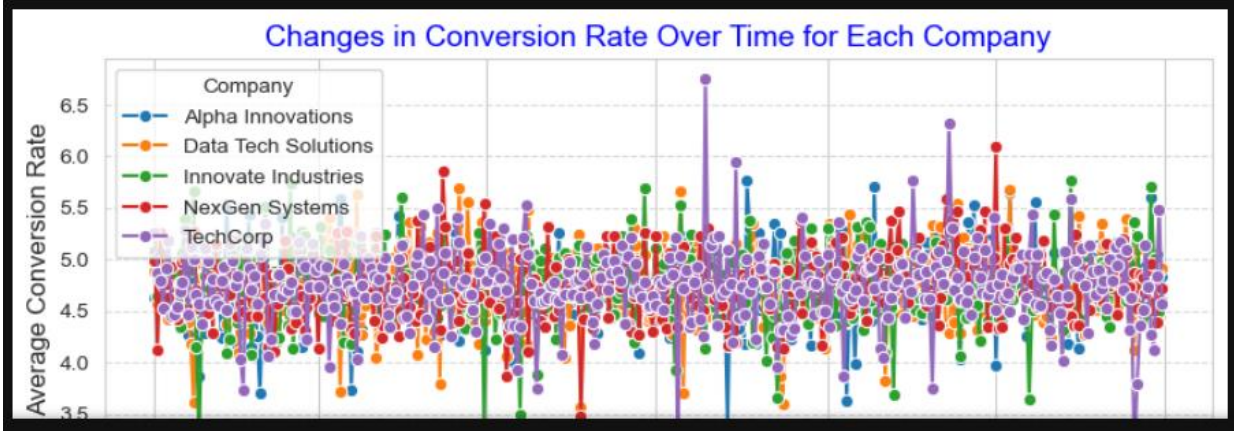
```
Plot the distribution of Duration using a histogram
***********************************************
```



3.4.2 Line chart: Conversion_Rate trends over time for each Company.

```
[80]: # ➢ Analyze how the overall Conversion_Rate has changed over Date for each Company using a line chart.
print("Analyze how the overall Conversion_Rate has changed over Date for each Company using a line chart")
print("**********************************************************************************************************")
# Convert the 'Date' column to datetime using the correct format
data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')

# Grouping data by 'Date' and 'Company' to calculate the mean Conversion_Rate
conversion_rate_by_date = data.groupby(['Date', 'Company'])['Conversion_Rate'].mean().reset_index()

# Setting the plot size and style
plt.figure(figsize=(8, 4), facecolor='white')
sns.set_style("whitegrid")

# Line plot for Conversion_Rate over Date
sns.lineplot(data=conversion_rate_by_date,x='Date',y='Conversion_Rate',hue='Company',palette='tab10',marker='o')

# Adding labels and title
plt.xlabel('Date', fontsize=12)
plt.ylabel('Average Conversion Rate', fontsize=12)
plt.title('Changes in Conversion Rate Over Time for Each Company', fontsize=14, color='blue')

# Rotating x-axis labels for better readability
plt.xticks(rotation=45)

# Adding grid lines
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Adjusting the layout
plt.tight_layout()

# Displaying the plot
plt.show()
```

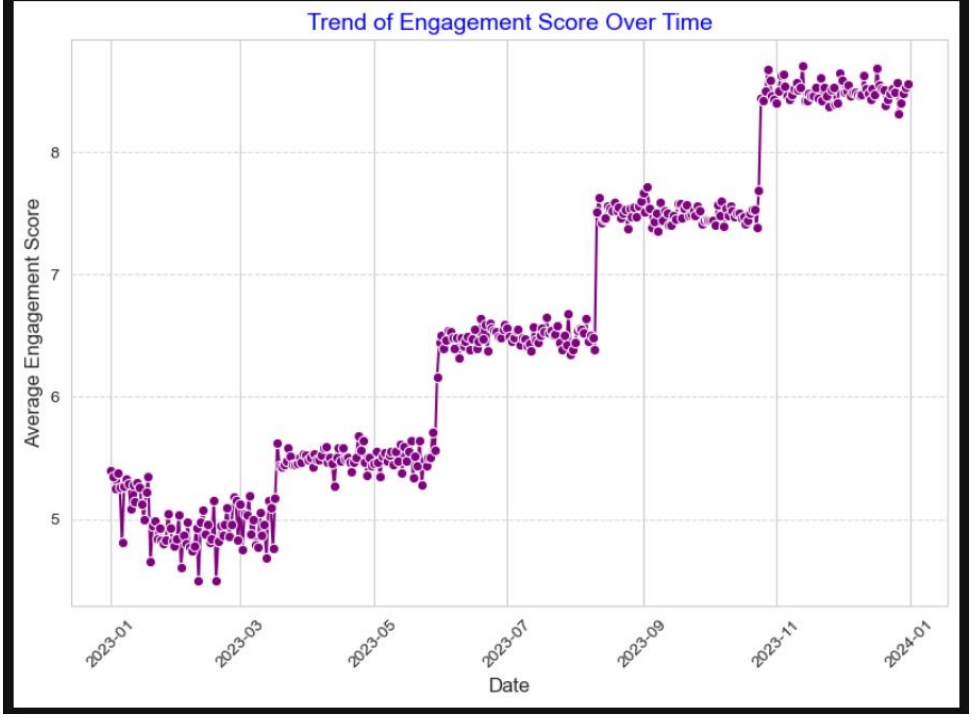
```
Analyze how the overall Conversion_Rate has changed over Date for each Company using a line chart
**********************************************************************************************************
```



3.4.3 Line chart: Engagement_Score trends over time.

```
# ➢ Examine the trend of Engagement_Score over Date with a line chart.
print("Examine the trend of Engagement_Score over Date with a line chart")
print("*****************************************************************")
# Convert the 'Date' column to datetime format, if not already done
data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')

# Group the data by Date and calculate the average Engagement_Score for each date
engagement_score_by_date = data.groupby('Date')['Engagement_Score'].mean().reset_index()

# Set the figure size and style for the plot
plt.figure(figsize=(8, 6), facecolor='white')
sns.set_style("whitegrid")

# Line plot for Engagement_Score over Date
sns.lineplot(data=engagement_score_by_date,x='Date',y='Engagement_Score',marker='o',color='purple')

# Add labels and title
plt.xlabel('Date', fontsize=12)
plt.ylabel('Average Engagement Score', fontsize=12)
plt.title('Trend of Engagement Score Over Time', fontsize=14, color='blue')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Add grid lines for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Adjust the layout to ensure no overlap
plt.tight_layout()

# Display the plot
plt.show()
```

```
Examine the trend of Engagement_Score over Date with a line chart
*****************************************************************
```

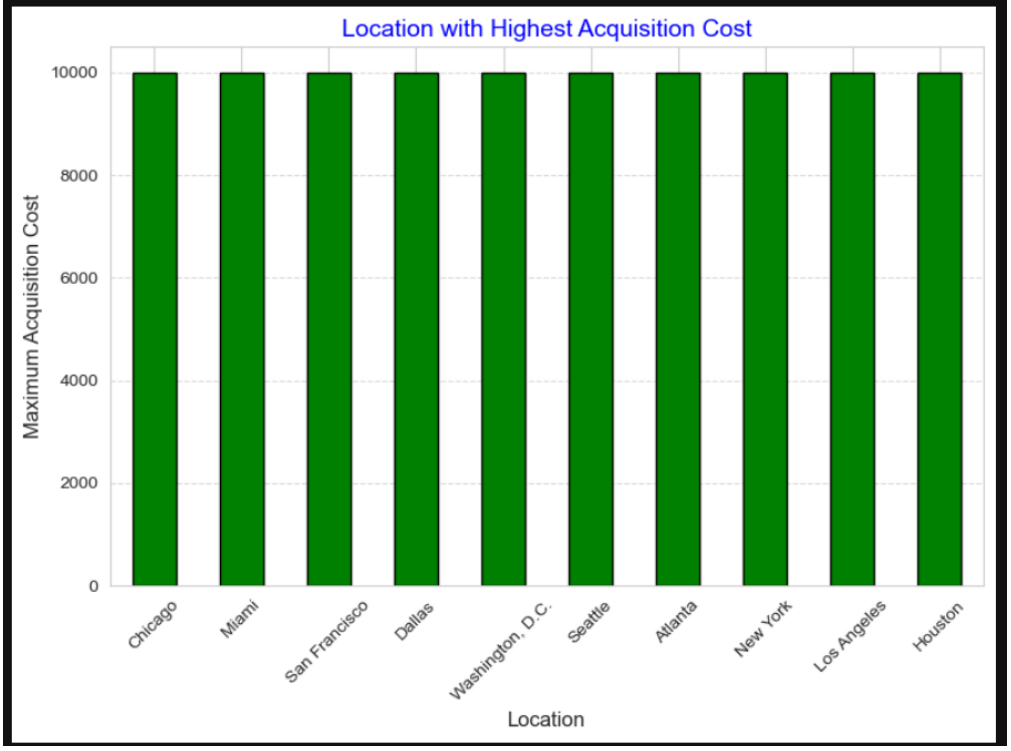


3.5 GEOGRAPHICAL ANALYSIS:

3.5.1 Bar chart: Location with the highest Acquisition_Cost.

```
# ➢ Determine which location has the highest Acquisition_Cost using a bar chart.
print("Determine which location has the highest Acquisition_Cost using a bar chart")
print("*****************************************************************************")
# Grouping data by 'Location' and calculating the maximum Acquisition_Cost for each location
location_acquisition_cost = data.groupby('Location')['Acquisition_Cost'].max()

# Setting up the figure
plt.figure(figsize=(10, 6), facecolor='white')

# Sorting the data in descending order and plotting the bar chart
location_acquisition_cost.sort_values(ascending=False).plot(kind='bar', color='green', edgecolor='black')

# Customizing the plot with labels and title
plt.xlabel('Location', fontsize=12)
plt.ylabel('Maximum Acquisition Cost', fontsize=12)
plt.title('Location with Highest Acquisition Cost', fontsize=14, color='blue')

# Rotating x-axis labels for readability
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)

# Adding gridlines for better readability of the y-axis
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Adjusting layout to avoid overlapping
plt.tight_layout()

# Display the plot
plt.show()
```

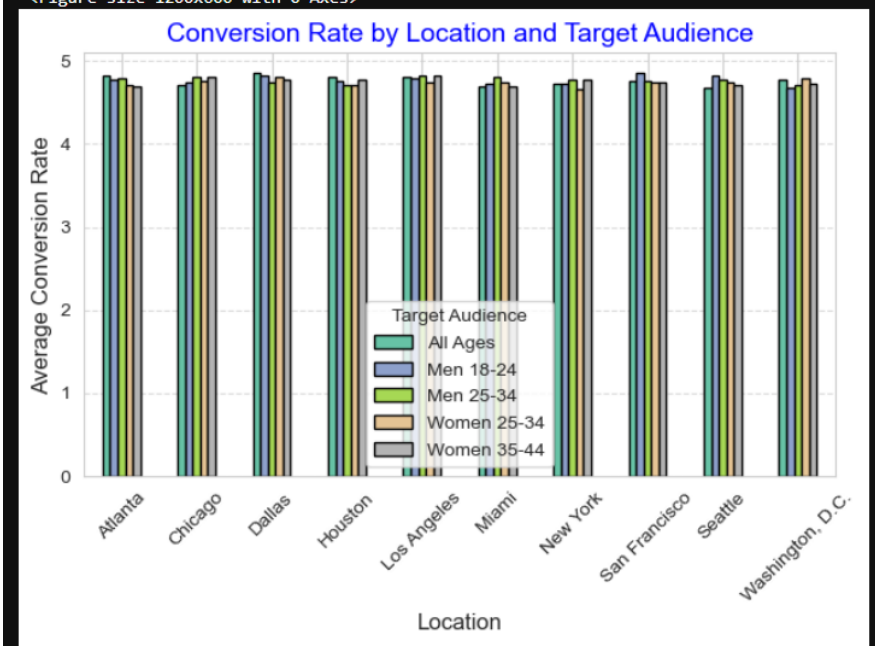
```
Determine which location has the highest Acquisition_Cost using a bar chart
*****************************************************************************
```



3.5.2 Bar chart: Conversion_Rate by Location, grouped by Target_Audience.

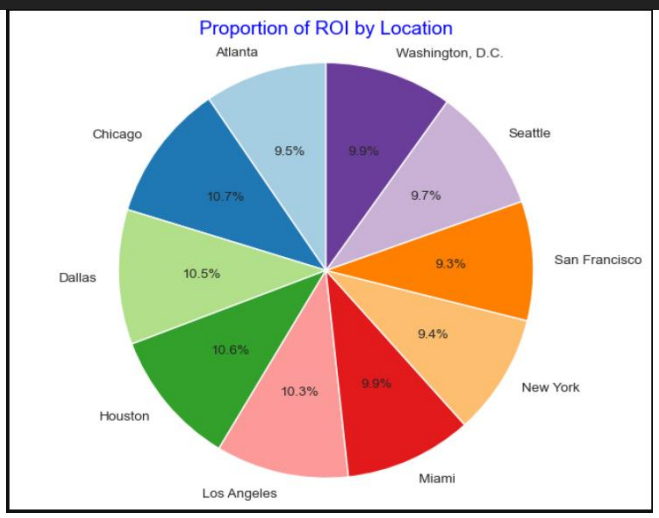
```
[95]: # ➢ Visualize the Conversion_Rate by different Location, categorized by Target_Audience, using a bar chart.
print("Visualize the Conversion_Rate by different Location, categorized by Target_Audience, using a bar chart")
print("**********************************************************************************************************************")
# Grouping data by 'Location' and 'Target_Audience' to calculate the mean Conversion_Rate
conversion_rate_by_location = data.groupby(['Location', 'Target_Audience'])['Conversion_Rate'].mean().unstack()

# Setting up the figure
plt.figure(figsize=(12, 6), facecolor='white')

# Plotting the bar chart with different colors for each Target_Audience
conversion_rate_by_location.plot(kind='bar', edgecolor='black', colormap='Set2')

# Customizing the plot with labels and title
plt.xlabel('Location', fontsize=12)
plt.ylabel('Average Conversion Rate', fontsize=12)
plt.title('Conversion Rate by Location and Target Audience', fontsize=14, color='blue')

# Rotating x-axis labels for readability
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)

# Adding a legend to identify the Target_Audience categories
plt.legend(title='Target Audience', fontsize=10)

# Adding gridlines for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Adjusting layout to avoid overlapping
plt.tight_layout()

# Display the plot
plt.show()
```

```
Visualize the Conversion_Rate by different Location, categorized by Target_Audience, using a bar chart
**********************************************************************************************************************
<Figure size 1200x600 with 0 Axes>
```



3.5.3 Pie chart: ROI proportion by Location.

```
[99]: # Grouping the data by 'Location' and summing up the ROI
roi_by_location = data.groupby('Location')['ROI'].sum()

# Setting up the figure
plt.figure(figsize=(8, 6), facecolor='white')

# Plotting the pie chart with labels and percentages
plt.pie(roi_by_location, labels=roi_by_location.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)

# Adding a title to the chart
plt.title('Proportion of ROI by Location', fontsize=14, color='blue')

# Equalizing the axis to make the pie chart circular
plt.axis('equal')

# Display the pie chart
plt.show()
```



4. INSIGHTS AND RECOMMENDATIONS:

4.1 Summarize findings for each analysis aspect.

- **Conversion Rate**: The conversion rates vary significantly across different locations and target audiences. Some locations perform exceptionally well, while others lag behind.
- **Engagement Score**: Higher engagement scores are observed in specific locations or for certain campaign types. However, some locations show lower engagement, indicating room for improvement.
- **Acquisition Cost**: The acquisition cost is higher in certain regions, leading to inefficiencies. Some regions have a much lower acquisition cost, making them more cost-effective.
- **ROI**: ROI is closely tied to both conversion rate and acquisition cost. Locations with higher conversion rates and lower acquisition costs yield better ROI, while others result in lower returns.

4.2 Provide actionable insights for optimizing campaign performance.

- **Target High-Conversion Segments**: Focus on the locations and target audiences that exhibit high conversion rates to maximize returns.
- **Refine Campaigns for Low-Engagement Locations**: Analyze the content and medium of campaigns in underperforming regions. Consider experimenting with new content formats or targeted outreach.
- **Optimize Spending**: Shift marketing budget towards regions with lower acquisition costs while reviewing and improving the strategy for high-cost locations to reduce inefficiencies.
- **Prioritize High-ROI Locations**: Allocate more resources to regions with a higher ROI. Continuously monitor ROI and adjust campaigns accordingly.

4.3 Suggest strategies to improve audience engagement and ROI.

- **Audience Segmentation and Personalization**: Customize campaigns based on regional and demographic preferences to increase conversion rates and engagement.
- **Enhance Content Interaction**: For regions with low engagement scores, integrate more interactive content (like polls, quizzes, or feedback surveys) to improve user interaction and engagement.
- **Leverage Cost-Efficient Channels**: Focus on using marketing channels that provide high engagement and ROI at a lower cost, such as organic social media or email marketing.
- **Regular Monitoring and Adjustments**: Track performance metrics regularly to identify which strategies are working. Adjust campaigns based on real-time data to maintain or improve ROI.

TOOLS AND TECHNOLOGIES:

- Python (pandas, matplotlib, seaborn, numpy)
- Jupyter Notebook
- CSV data format

CONCLUSION:

This project will provide actionable insights into marketing campaign performance, enabling businesses to refine their strategies, reduce costs, and maximize ROI. By leveraging data-driven approaches, the analysis will highlight factors contributing to success and guide future decision-making in digital marketing.