In [4]:
```
import pandas as pd

df = pd.read_csv(r"C:\Users\Mummy Mo\Downloads\HNG Data set\marketing_campaign_dataset.csv")

# Display basic information about the dataset
df_info = df.info()
df_head = df.head()

df_info, df_head
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200005 entries, 0 to 200004
Data columns (total 15 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   Campaign_ID         200005 non-null  int64  
 1   Company             200005 non-null  object 
 2   Campaign_Type       200005 non-null  object 
 3   Target_Audience     200005 non-null  object 
 4   Duration            200005 non-null  object 
 5   Channel_Used        200005 non-null  object 
 6   Conversion_Rate     200005 non-null  float64
 7    Acquisition_Cost   200005 non-null  object 
 8   ROI                 200005 non-null  float64
 9   Location            200005 non-null  object 
 10  Date                200005 non-null  object 
 11  Clicks              200005 non-null  int64  
 12  Impressions         200005 non-null  int64  
 13  Engagement_Score    200005 non-null  int64  
 14  Customer_Segment    200005 non-null  object 
dtypes: float64(2), int64(4), object(9)
memory usage: 22.9+ MB
```

Out[4]:
```
(None,
    Campaign_ID              Company Campaign_Type Target_Audience Duration  \
 0            1  Innovate Industries         Email       Men 18-24  30 days   
 1            2       NexGen Systems         Email     Women 35-44  60 days   
 2            3    Alpha Innovations    Influencer       Men 25-34  30 days   
 3            4   DataTech Solutions       Display        All Ages  60 days   
 4            5       NexGen Systems         Email       Men 25-34  15 days   
 
   Channel_Used  Conversion_Rate  Acquisition_Cost    ROI     Location  \
 0   Google Ads             0.04        $16,174.00   6.29      Chicago   
 1   Google Ads             0.12        $11,566.00   5.61     New York   
 2      YouTube             0.07        $10,200.00   7.18  Los Angeles   
 3      YouTube             0.11        $12,724.00   5.55        Miami   
 4      YouTube             0.05        $16,452.00   6.50  Los Angeles   
 
          Date  Clicks  Impressions  Engagement_Score     Customer_Segment  
 0  01/01/2021     506         1922                 6    Health & Wellness  
 1  01/02/2021     116         7523                 7         Fashionistas  
 2  01/03/2021     584         7698                 1  Outdoor Adventurers  
 3  01/04/2021     217         1820                 7    Health & Wellness  
 4  01/05/2021     379         4201                 3    Health & Wellness  )
```

In [10]:
```
# Convert 'Date' to datetime format
df['Date'] = pd.to_datetime(df['Date'], format="%d/%m/%Y")

# Convert ' Acquisition_Cost ' to numeric (removing '$' and commas)
df[' Acquisition_Cost '] = df[' Acquisition_Cost '].replace(r'[\$,]', '', regex=True).astype(float)
```

In [12]:
```
# Check for missing values
missing_values = df.isnull().sum()

# Generate summary statistics for numerical columns
summary_stats = df.describe()

missing_values, summary_stats
```

Out[12]:

```
(Campaign_ID           0
 Company               0
 Campaign_Type         0
 Target_Audience       0
 Duration              0
 Channel_Used          0
 Conversion_Rate       0
  Acquisition_Cost     0
 ROI                   0
 Location              0
 Date                  0
 Clicks                0
 Impressions           0
 Engagement_Score      0
 Customer_Segment      0
 dtype: int64,
          Campaign_ID  Conversion_Rate   Acquisition_Cost             ROI  \
 count  200005.000000    200005.000000       200005.000000  200005.000000   
 mean   100003.000000         0.080069        12504.441794       5.002416   
 min         1.000000         0.010000         5000.000000       2.000000   
 25%     50002.000000         0.050000         8740.000000       3.500000   
 50%    100003.000000         0.080000        12497.000000       5.010000   
 75%    150004.000000         0.120000        16264.000000       6.510000   
 max    200005.000000         0.150000        20000.000000       8.000000   
 std     57736.614632         0.040602         4337.663210       1.734485   

                                 Date         Clicks    Impressions  \
 count                         200005  200005.000000  200005.000000   
 mean   2021-07-01 23:37:44.289392896     549.774591    5507.307107   
 min              2021-01-01 00:00:00     100.000000    1000.000000   
 25%              2021-04-02 00:00:00     325.000000    3266.000000   
 50%              2021-07-02 00:00:00     550.000000    5518.000000   
 75%              2021-10-01 00:00:00     775.000000    7753.000000   
 max              2021-12-31 00:00:00    1000.000000   10000.000000   
 std                              NaN     260.019354    2596.863794   

        Engagement_Score  
 count     200005.000000  
 mean           5.494673  
 min            1.000000  
 25%            3.000000  
 50%            5.000000  
 75%            8.000000  
 max           10.000000  
 std            2.872593  )
```

In [14]:

```python
# Check for any remaining null values in 'Date' after conversion
date_null_count = df['Date'].isnull().sum()

# Display rows where 'Date' could not be converted (if any)
invalid_dates = df[df['Date'].isnull()].head()

date_null_count, invalid_dates
```

Out[14]:

```
(0,
 Empty DataFrame
 Columns: [Campaign_ID, Company, Campaign_Type, Target_Audience, Duration, Channel_Used, Conversion_Rate,  Acqu
isition_Cost , ROI, Location, Date, Clicks, Impressions, Engagement_Score, Customer_Segment]
 Index: [])
```

In [16]:

```python
# Print column names to check for typos
print(df.columns)
```

```
Index(['Campaign_ID', 'Company', 'Campaign_Type', 'Target_Audience',
       'Duration', 'Channel_Used', 'Conversion_Rate', ' Acquisition_Cost ',
       'ROI', 'Location', 'Date', 'Clicks', 'Impressions', 'Engagement_Score',
       'Customer_Segment'],
      dtype='object')
```

In [18]:

```python
# Check for missing values again after cleaning
missing_values = df.isnull().sum()

# Generate summary statistics for numerical columns
summary_stats = df.describe()

missing_values, summary_stats
```

Out[18]:
```
(Campaign_ID           0
 Company               0
 Campaign_Type         0
 Target_Audience       0
 Duration              0
 Channel_Used          0
 Conversion_Rate       0
  Acquisition_Cost     0
 ROI                   0
 Location              0
 Date                  0
 Clicks                0
 Impressions           0
 Engagement_Score      0
 Customer_Segment      0
 dtype: int64,
          Campaign_ID  Conversion_Rate   Acquisition_Cost            ROI  \
 count  200005.000000    200005.000000      200005.000000  200005.000000
 mean   100003.000000         0.080069       12504.441794       5.002416
 min         1.000000         0.010000        5000.000000       2.000000
 25%     50002.000000         0.050000        8740.000000       3.500000
 50%    100003.000000         0.080000       12497.000000       5.010000
 75%    150004.000000         0.120000       16264.000000       6.510000
 max    200005.000000         0.150000       20000.000000       8.000000
 std     57736.614632         0.040602        4337.663210       1.734485

                                 Date         Clicks    Impressions  \
 count                         200005  200005.000000  200005.000000
 mean   2021-07-01 23:37:44.289392896     549.774591    5507.307107
 min              2021-01-01 00:00:00     100.000000    1000.000000
 25%              2021-04-02 00:00:00     325.000000    3266.000000
 50%              2021-07-02 00:00:00     550.000000    5518.000000
 75%              2021-10-01 00:00:00     775.000000    7753.000000
 max              2021-12-31 00:00:00    1000.000000   10000.000000
 std                              NaN     260.019354    2596.863794

        Engagement_Score
 count     200005.000000
 mean           5.494673
 min            1.000000
 25%            3.000000
 50%            5.000000
 75%            8.000000
 max           10.000000
 std            2.872593  )
```

In [20]:
```
# Get unique values for Target Audiences and Marketing Channels
unique_target_audiences = df["Target_Audience"].unique()
unique_marketing_channels = df["Channel_Used"].unique()

# Count of unique values
num_target_audiences = len(unique_target_audiences)
num_marketing_channels = len(unique_marketing_channels)

unique_target_audiences, num_target_audiences, unique_marketing_channels, num_marketing_channels
```

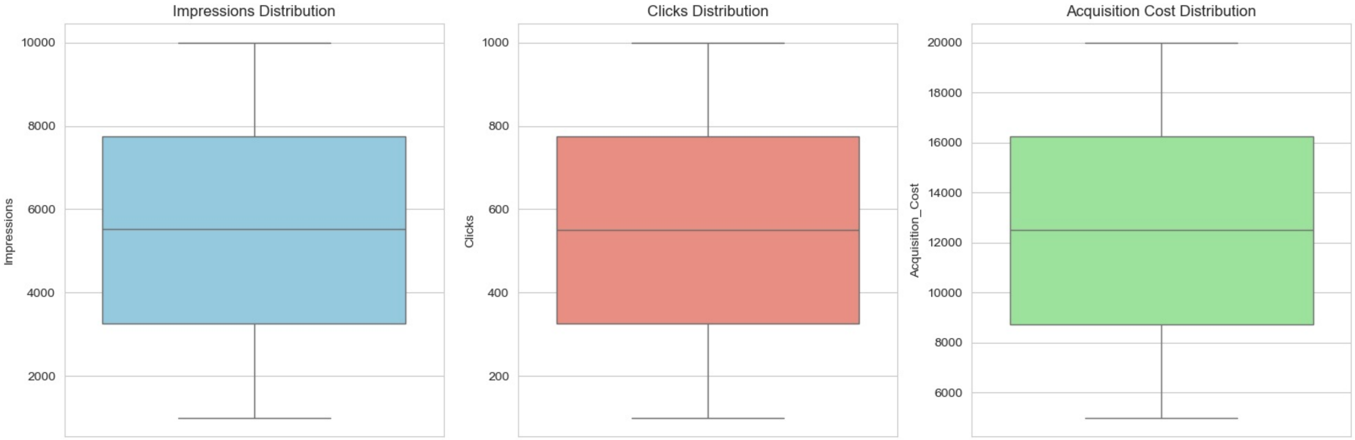
Out[20]:
```
(array(['Men 18-24', 'Women 35-44', 'Men 25-34', 'All Ages', 'Women 25-34'],
       dtype=object),
 5,
 array(['Google Ads', 'YouTube', 'Instagram', 'Website', 'Facebook',
        'Email'], dtype=object),
 6)
```

In [22]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [27]:
```
# Reload the dataset
file_path = (r"C:\Users\Mummy Mo\Downloads\HNG Data set\m_c_dataset_csv.csv")
df = pd.read_csv(file_path)

# Convert 'Acquisition_Cost' to numeric (removing '$' and commas if present)
df["Acquisition_Cost"] = df["Acquisition_Cost"].replace('[\$,]', '', regex=True).astype(float)

# Set visualization style
sns.set_style("whitegrid")

# Create boxplots for detecting outliers in key numerical metrics
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

sns.boxplot(y=df["Impressions"], ax=axes[0], color="skyblue")
axes[0].set_title("Impressions Distribution")
```

```
sns.boxplot(y=df["Clicks"], ax=axes[1], color="salmon")
axes[1].set_title("Clicks Distribution")

sns.boxplot(y=df["Acquisition_Cost"], ax=axes[2], color="lightgreen")
axes[2].set_title("Acquisition Cost Distribution")

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

```
<>:6: SyntaxWarning: invalid escape sequence '\$'
<>:6: SyntaxWarning: invalid escape sequence '\$'
C:\Users\Mummy Mo\AppData\Local\Temp\ipykernel_15148\2803953236.py:6: SyntaxWarning: invalid escape sequence '\$
'
  df["Acquisition_Cost"] = df["Acquisition_Cost"].replace('[\$,]', '', regex=True).astype(float)
```



In [29]:

```
# Function to detect outliers using IQR method
def detect_outliers(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1

    # Define lower and upper bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Filter outliers
    outliers = data[(data[column] < lower_bound) | (data[column] > upper_bound)]

    return outliers

# Identify outliers for each key numerical field
outliers_impressions = detect_outliers(df, "Impressions")
outliers_clicks = detect_outliers(df, "Clicks")
outliers_spend = detect_outliers(df, "Acquisition_Cost")

# Display counts of outliers detected
outlier_counts = {
    "Impressions Outliers": len(outliers_impressions),
    "Clicks Outliers": len(outliers_clicks),
    "Acquisition Cost Outliers": len(outliers_spend)
}

outlier_counts
```

Out[29]:

```
{'Impressions Outliers': 0,
 'Clicks Outliers': 0,
 'Acquisition Cost Outliers': 0}
```

In [10]:

```
import pandas as pd
df = pd.read_csv(r"C:\Users\Mummy Mo\Downloads\HNG Data set\m_c_dataset_csv.csv")  # Or another method to creat

# Calculate Click-Through Rate (CTR) and Cost Per Click (CPC)
df['CTR'] = df['Clicks'] / df['Impressions']
df['CPC'] = df['Acquisition_Cost'] / df['Clicks']
df['Conversion_Rate'] = df['Conversion_Rate']

# Summarize key campaign metrics
campaign_metrics = df[["CTR", "CPC", "Conversion_Rate"]].describe()

# Summary statistics for CTR and CPC
ctr_cpc_stats = df[['CTR', 'CPC', 'Conversion_Rate']].describe()

# Check for outliers in CTR and CPC using IQR method
Q1 = df[['CTR', 'CPC', 'Conversion_Rate']].quantile(0.25)
Q3 = df[['CTR', 'CPC', 'Conversion_Rate']].quantile(0.75)
IQR = Q3 - Q1
```

```
outliers = ((df[['CTR', 'CPC', 'Conversion_Rate']] < (Q1 - 1.5 * IQR)) | (df[['CTR', 'CPC', 'Conversion_Rate']]

ctr_cpc_stats, outliers
```

Out[10]:

```
(                 CTR            CPC  Conversion_Rate
 count  200005.000000  200005.000000    200005.000000
 mean        0.140405      32.008319         0.080069
 std         0.130880      26.925841         0.040602
 min         0.010054       5.021084         0.010000
 25%         0.058606      15.092037         0.050000
 50%         0.099790      22.773973         0.080000
 75%         0.169698      38.598253         0.120000
 max         0.992024     199.960000         0.150000,
 CTR                   16141
 CPC                   16045
 Conversion_Rate           0
 dtype: int64)
```

In [39]:

```
# Calculate Click-Through Rate (CTR) = (Clicks / Impressions) * 100
df["CTR"] = (df["Clicks"] / df["Impressions"]) * 100

# Calculate Cost Per Click (CPC) = Acquisition Cost / Clicks
df["CPC"] = df["Acquisition_Cost"] / df["Clicks"]

# Group by marketing channel and calculate average ROI, CTR, and CPC
channel_performance = df.groupby("Channel_Used")[["ROI", "CTR", "CPC", "Conversion_Rate"]].mean().reset_index()

# Sort by highest ROI
channel_performance = channel_performance.sort_values(by="ROI", ascending=False)

# Display the summary table
channel_performance
```

Out[39]:

	Channel_Used	ROI	CTR	CPC	Conversion_Rate
1	Facebook	5.018672	14.049724	32.129366	0.079990
4	Website	5.014114	14.096941	31.779148	0.080182
2	Google Ads	5.003126	13.918943	32.308459	0.080181
0	Email	4.996487	14.054269	31.881471	0.080282
5	YouTube	4.993720	14.119755	31.872904	0.079890
3	Instagram	4.988706	14.003691	32.080786	0.079886

In [14]:

```
# Calculate total Clicks, Impressions, and Acquisition Cost by Marketing Channel
channel_stats = df.groupby("Channel_Used").agg(
    Total_Clicks=("Clicks", "sum"),
    Total_Impressions=("Impressions", "sum"),
    Total_Acquisition_Cost=("Acquisition_Cost", "sum")
).reset_index()

# Compute CTR and CPC
channel_stats["CTR (%)"] = (channel_stats["Total_Clicks"] / channel_stats["Total_Impressions"]) * 100
channel_stats["CPC"] = channel_stats["Total_Acquisition_Cost"] / channel_stats["Total_Clicks"]

# Display the results
channel_stats
```

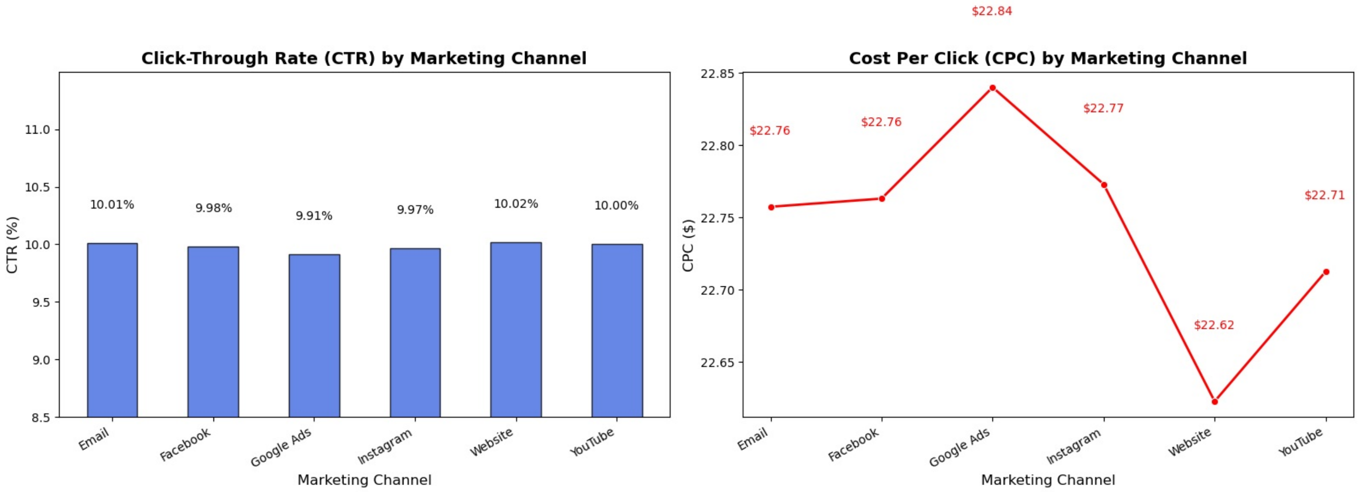
Out[14]:

	Channel_Used	Total_Clicks	Total_Impressions	Total_Acquisition_Cost	CTR (%)	CPC
0	Email	18493963	184801107	420874104	10.007496	22.757378
1	Facebook	18038175	180662496	410603426	9.984460	22.763025
2	Google Ads	18342589	185020154	418944514	9.913833	22.839988
3	Instagram	18316654	183738455	417124850	9.968873	22.772983
4	Website	18415351	183815901	416606897	10.018367	22.622805
5	YouTube	18350935	183450845	416797090	10.003189	22.712581

In [24]:

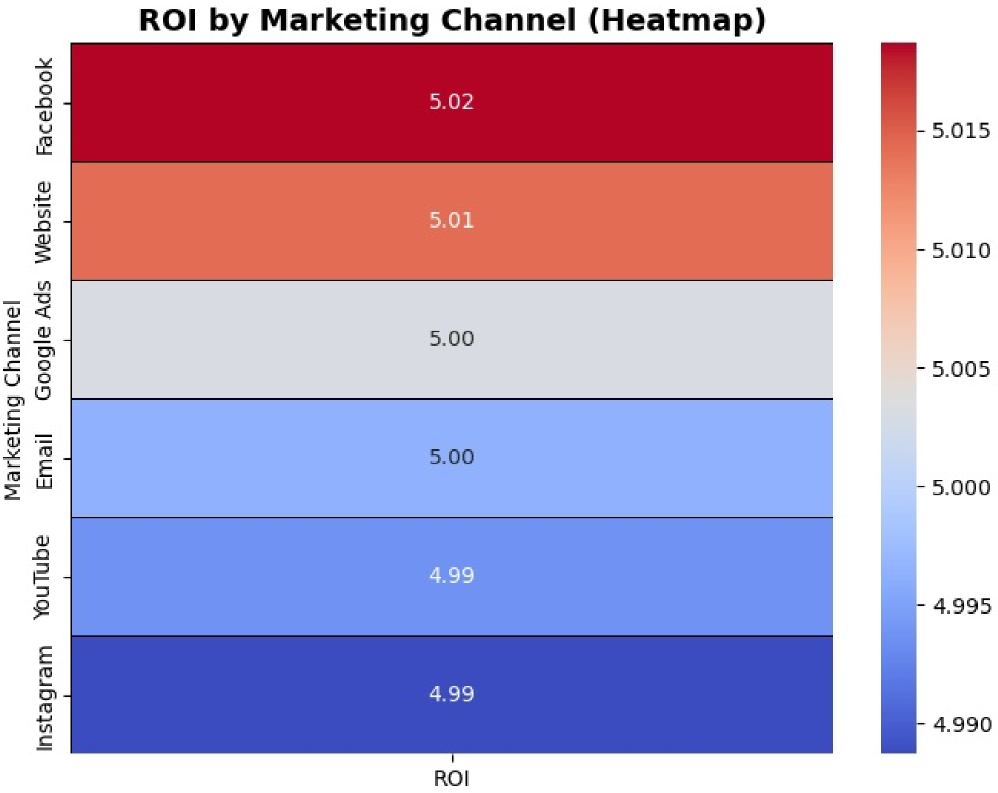
```
# Re-import NumPy to avoid NameError
import numpy as np

# Create a new figure with two subplots
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# CTR Bar Chart with Better Spacing
bars = axes[0].bar(channel_stats["Channel_Used"], channel_stats["CTR (%)"], color="royalblue", alpha=0.8, width

axes[0].set_title("Click-Through Rate (CTR) by Marketing Channel", fontsize=14, fontweight="bold")
```

```
axes[0].set_xlabel("Marketing Channel", fontsize=12)
axes[0].set_ylabel("CTR (%)", fontsize=12)

# Properly set x-axis ticks and labels
axes[0].set_xticks(range(len(channel_stats["Channel_Used"])))
axes[0].set_xticklabels(channel_stats["Channel_Used"], rotation=30, ha="right")

# Adjust CTR Y-axis to create more spacing
ctr_min = np.floor(channel_stats["CTR (%)"].min()) - 0.5  # Extra space at bottom
ctr_max = np.ceil(channel_stats["CTR (%)"].max()) + 0.5  # Extra space at top
ctr_ticks = np.arange(ctr_min, ctr_max, 0.5)  # Keep 0.5% increments
axes[0].set_yticks(ctr_ticks)
axes[0].set_ylim(ctr_min, ctr_max)

# Move labels higher to avoid overlap
for bar in bars:
    height = bar.get_height()
    axes[0].text(bar.get_x() + bar.get_width() / 2, height + 0.3, f"{height:.2f}%", ha="center", fontsize=10, c

# CPC Line Chart
sns.lineplot(x="Channel_Used", y="CPC", data=channel_stats, ax=axes[1], marker="o", color="red", linewidth=2)
axes[1].set_title("Cost Per Click (CPC) by Marketing Channel", fontsize=14, fontweight="bold")
axes[1].set_xlabel("Marketing Channel", fontsize=12)
axes[1].set_ylabel("CPC ($)", fontsize=12)

# Properly set x-axis ticks and labels for CPC chart
axes[1].set_xticks(range(len(channel_stats["Channel_Used"])))
axes[1].set_xticklabels(channel_stats["Channel_Used"], rotation=30, ha="right")

# Add data labels on CPC points
for index, row in channel_stats.iterrows():
    axes[1].text(index, row["CPC"] + 0.05, f"${row['CPC']:.2f}", ha="center", fontsize=10, color="red")

# Adjust layout for better spacing
plt.tight_layout()

# Show the improved charts without errors
plt.show()
```



In [26]:

```
# Calculate the average ROI for each marketing channel
roi_table = df.groupby("Channel_Used", as_index=False)["ROI"].mean().sort_values(by="ROI", ascending=False)

# Create a heatmap visualization
plt.figure(figsize=(8, 6))
roi_pivot = roi_table.set_index("Channel_Used")  # Set marketing channel as index

# Generate the heatmap
sns.heatmap(roi_pivot, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5, linecolor="black")

# Chart labels and title
plt.title("ROI by Marketing Channel (Heatmap)", fontsize=14, fontweight="bold")
plt.xlabel("")
plt.ylabel("Marketing Channel")

# Show the table and heatmap
plt.show()

# Display the ROI table
roi_table
```


ROI by Marketing Channel (Heatmap)

Out[26]:

	Channel_Used	ROI
1	Facebook	5.018672
4	Website	5.014114
2	Google Ads	5.003126
0	Email	4.996487
5	YouTube	4.993720
3	Instagram	4.988706

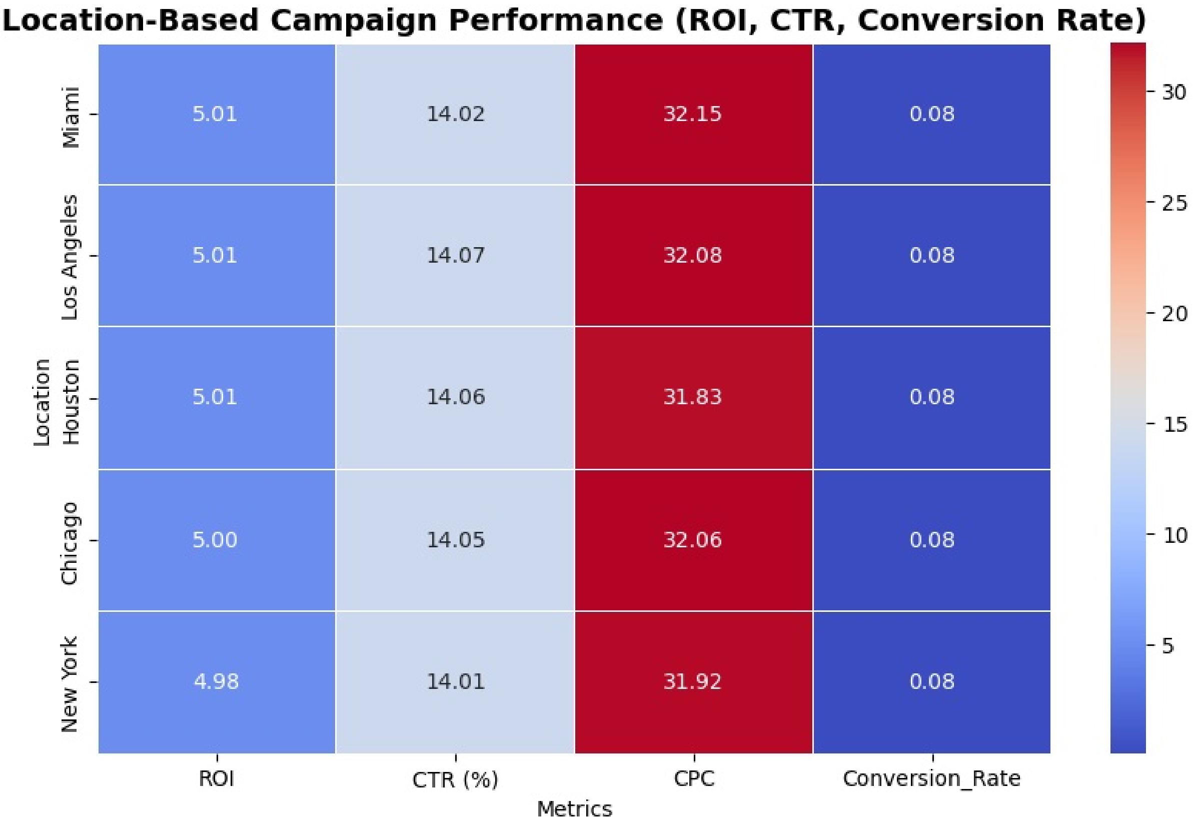
In [36]:

```
import pandas as pd

# Calculate CTR (Click-Through Rate)
df["CTR (%)"] = (df["Clicks"] / df["Impressions"]) * 100

# Create a table summarizing campaign performance by location
location_table = df.groupby("Location", as_index=False).agg({
    "ROI": "mean",
    "CTR (%)": "mean",
    "CPC": "mean",
    "Conversion_Rate": "mean"
}).sort_values(by="ROI", ascending=False)  # Sort by highest ROI

# Display the table
print(location_table)
```

```
      Location       ROI    CTR (%)        CPC  Conversion_Rate
3        Miami  5.012282  14.024957  32.152425         0.080047
2  Los Angeles  5.010876  14.067175  32.078189         0.080013
1      Houston  5.007174  14.059033  31.829355         0.079949
0      Chicago  5.001555  14.045011  32.055853         0.080131
4     New York  4.980185  14.006619  31.923819         0.080203
```

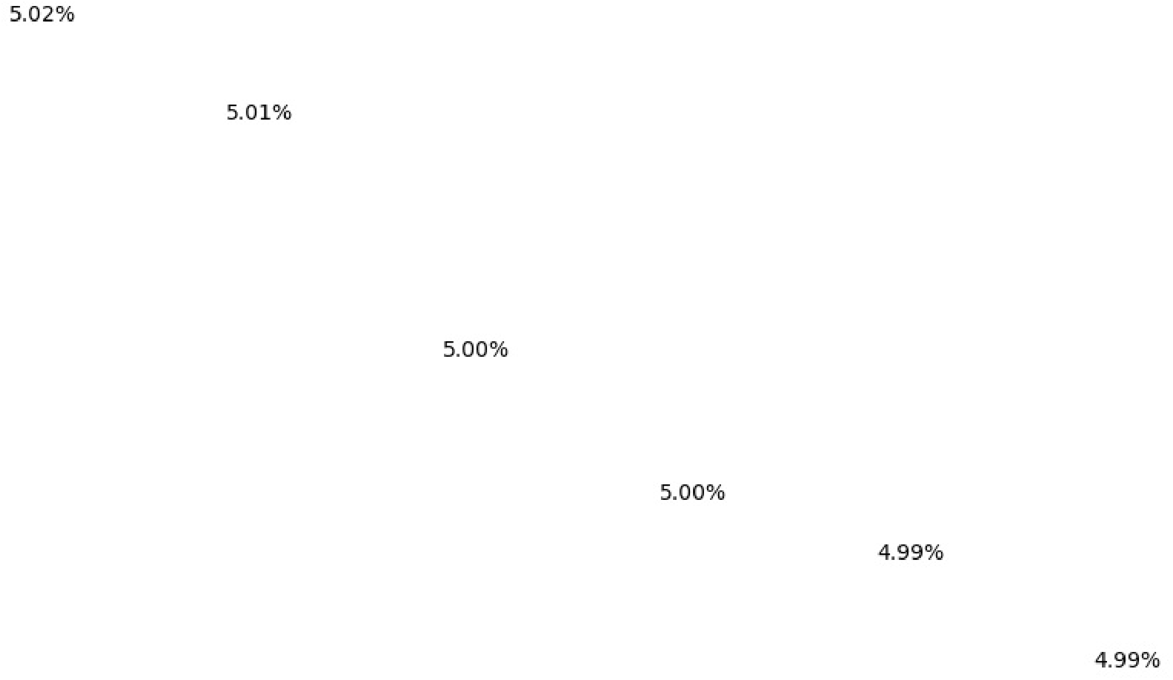
In [42]:

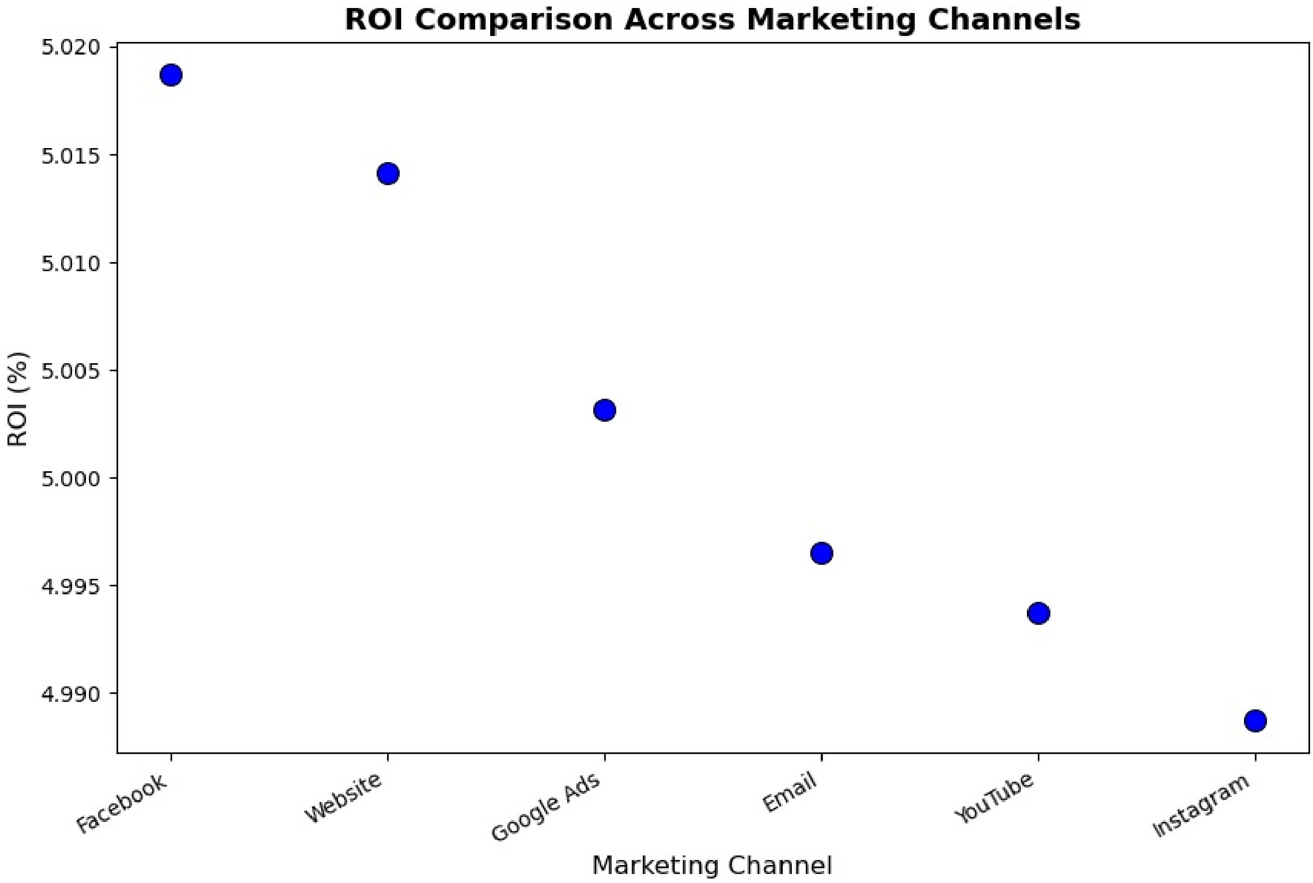
```
import seaborn as sns
import matplotlib.pyplot as plt

# Pivot data for heatmap
heatmap_data = location_table.set_index("Location")

# Create heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)

# Labels
plt.title("Location-Based Campaign Performance (ROI, CTR, Conversion Rate)", fontsize=14, fontweight="bold")
plt.xlabel("Metrics")
plt.ylabel("Location")

plt.show()
```



In [44]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Create scatter plot for ROI by Marketing Channel
plt.figure(figsize=(10, 6))
sns.scatterplot(data=roi_table, x="Channel_Used", y="ROI", s=100, color="blue", edgecolor="black")

# Add labels for each point
for index, row in roi_table.iterrows():
    plt.text(row["Channel_Used"], row["ROI"] + 0.1, f"{row['ROI']:.2f}%", ha="center", fontsize=10, color="blac

# Labels and title
plt.title("ROI Comparison Across Marketing Channels", fontsize=14, fontweight="bold")
plt.xlabel("Marketing Channel", fontsize=12)
plt.ylabel("ROI (%)", fontsize=12)
plt.xticks(rotation=30, ha="right")

# Show plot
plt.show()
```




ROI Comparison Across Marketing Channels
5.020
5.015
5.010
5.005
5.000
4.995
4.990
ROI (%)
Facebook
Website
Google Ads
Email
YouTube
Instagram
Marketing Channel

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js