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

print("Libraries imported successfully!")
```

```
Libraries imported successfully!
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Stage2_Task - marketing_campaign_dataset_Excel.xlsx to Stage2_Task - marketing_campaign_dataset_Excel (3).xlsx
```

```
!ls /content/
```

```
 sample_data
'Stage2_Task - marketing_campaign_dataset_Excel (1).xlsx'
'Stage2_Task - marketing_campaign_dataset_Excel (2).xlsx'
'Stage2_Task - marketing_campaign_dataset_Excel (3).xlsx'
'Stage2_Task - marketing_campaign_dataset_Excel.xlsx'
```

```
import os

# Keep only one version and delete the rest
os.remove("/content/Stage2_Task - marketing_campaign_dataset_Excel (1).xlsx")
os.remove("/content/Stage2_Task - marketing_campaign_dataset_Excel (2).xlsx")
os.remove("/content/Stage2_Task - marketing_campaign_dataset_Excel (3).xlsx")

print("Extra files deleted successfully! ✅")
```

```
Extra files deleted successfully! ✅
```

```
!ls /content/
```

```
 sample_data  'Stage2_Task - marketing_campaign_dataset_Excel.xlsx'
```

```
# iMPORTING THE DATASET #
import pandas as pd

file_name = "Stage2_Task - marketing_campaign_dataset_Excel.xlsx"
df = pd.read_excel(file_name)

df.head()  # Display first few rows
```

| | Campaign_ID | Company | Campaign_Type | Target_Audience | Duration | Channel_Used | Conversion_Rate | Acquisition_Cost | ROI | Location | Date | Cl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Innovate Industries | Email | Men 18-24 | 30 days | Google Ads | 0.04 | 16174 | 6.29 | Chicago | 2021-01-01 00:00:00 | |
| **1** | 2 | NexGen Systems | Email | Women 35-44 | 60 days | Google Ads | 0.12 | 11566 | 5.61 | New York | 2021-02-01 00:00:00 | |
| **2** | 3 | Alpha Innovations | Influencer | Men 25-34 | 30 days | YouTube | 0.07 | 10200 | 7.18 | Los Angeles | 2021-03-01 00:00:00 | |
| **3** | 4 | DataTech Solutions | Display | All Ages | 60 days | YouTube | 0.11 | 12724 | 5.55 | Miami | 2021-04-01 00:00:00 | |
| **4** | 5 | NexGen Systems | Email | Men 25-34 | 15 days | YouTube | 0.05 | 16452 | 6.50 | Los Angeles | 2021-05-01 00:00:00 | |

```
# Upload the Excel File into a DataFrame #
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving Stage2_Task - marketing_campaign_dataset_Excel.xlsx to Stage2_Task - marketing_campaign_dataset_Excel.xlsx

```
# Load the Excel File into a DataFrame #

file_name = "Stage2_Task - marketing_campaign_dataset_Excel.xlsx"
df = pd.read_excel(file_name)



# Load Dataset; Count Rows and Columns

print(type(df))
print(df.shape)
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
(200005, 15)
   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             16174  6.29      Chicago
1   Google Ads             0.12             11566  5.61     New York
2      YouTube             0.07             10200  7.18  Los Angeles
3      YouTube             0.11             12724  5.55        Miami
4      YouTube             0.05             16452  6.50  Los Angeles

                   Date  Clicks  Impressions  Engagement_Score  \
0  2021-01-01 00:00:00     506         1922                 6
1  2021-02-01 00:00:00     116         7523                 7
2  2021-03-01 00:00:00     584         7698                 1
3  2021-04-01 00:00:00     217         1820                 7
4  2021-05-01 00:00:00     379         4201                 3

      Customer_Segment
0    Health & Wellness
1         Fashionistas
2  Outdoor Adventurers
3    Health & Wellness
4    Health & Wellness
```

```
# Convert Date Into Proper Date Format #

df["Date"] = pd.to_datetime(df["Date"], errors="coerce")
print(df["Date"].head())
```

```
0   2021-01-01
1   2021-02-01
2   2021-03-01
3   2021-04-01
4   2021-05-01
Name: Date, dtype: datetime64[ns]
```

```
# Count Missing Values #

print(df.isnull().sum())
```

```
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
```

```
# Count Duplicate Rows #
```

```
print(df.duplicated().sum())
```

```
0
```

```
# Check for Inconsistent or Unexpected Values #

for col in ["Company", "Campaign_Type", "Target_Audience", "Channel_Used", "Location", "Customer_Segment"]:
    print(f"{col}:\n", df[col].unique(), "\n")
```

```
Company:
 ['Innovate Industries' 'NexGen Systems' 'Alpha Innovations'
 'DataTech Solutions' 'TechCorp']

Campaign_Type:
 ['Email' 'Influencer' 'Display' 'Search' 'Social Media']

Target_Audience:
 ['Men 18-24' 'Women 35-44' 'Men 25-34' 'All Ages' 'Women 25-34']

Channel_Used:
 ['Google Ads' 'YouTube' 'Instagram' 'Website' 'Facebook' 'Email']

Location:
 ['Chicago' 'New York' 'Los Angeles' 'Miami' 'Houston']

Customer_Segment:
 ['Health & Wellness' 'Fashionistas' 'Outdoor Adventurers' 'Foodies'
 'Tech Enthusiasts']
```

```
# Check for Extra Spaces or Formatting Issues #

df["Company"] = df["Company"].str.strip()
df["Campaign_Type"] = df["Campaign_Type"].str.strip()
df["Target_Audience"] = df["Target_Audience"].str.strip()
df["Channel_Used"] = df["Channel_Used"].str.strip()
df["Location"] = df["Location"].str.strip()
df["Customer_Segment"] = df["Customer_Segment"].str.strip()
```

```
# Value Counts #
for col in ["Company", "Campaign_Type", "Target_Audience", "Channel_Used", "Location", "Customer_Segment"]:
    print(f"\n{col}:\n", df[col].value_counts())
```

```
Company:
 Company
TechCorp               40238
Alpha Innovations      40051
DataTech Solutions     40014
NexGen Systems         39991
Innovate Industries    39711
Name: count, dtype: int64

Campaign_Type:
 Campaign_Type
Influencer      40170
Search          40157
Display         39988
Email           39871
Social Media    39819
Name: count, dtype: int64

Target_Audience:
 Target_Audience
Men 18-24      40259
Men 25-34      40024
All Ages       40021
Women 25-34    40013
Women 35-44    39688
Name: count, dtype: int64

Channel_Used:
 Channel_Used
Email         33599
Google Ads    33440
YouTube       33393
Instagram     33392
Website       33361
Facebook      32820
Name: count, dtype: int64

Location:
```

```
 Location
Miami          40269
New York       40025
Chicago        40013
Los Angeles    39947
Houston        39751
Name: count, dtype: int64

Customer_Segment:
 Customer_Segment
Foodies                40210
Tech Enthusiasts       40154
Outdoor Adventurers    40011
Health & Wellness      39888
Fashionistas           39742
Name: count, dtype: int64
```

```
# Summary Statistics #
print(df.describe())
```

```
          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
```

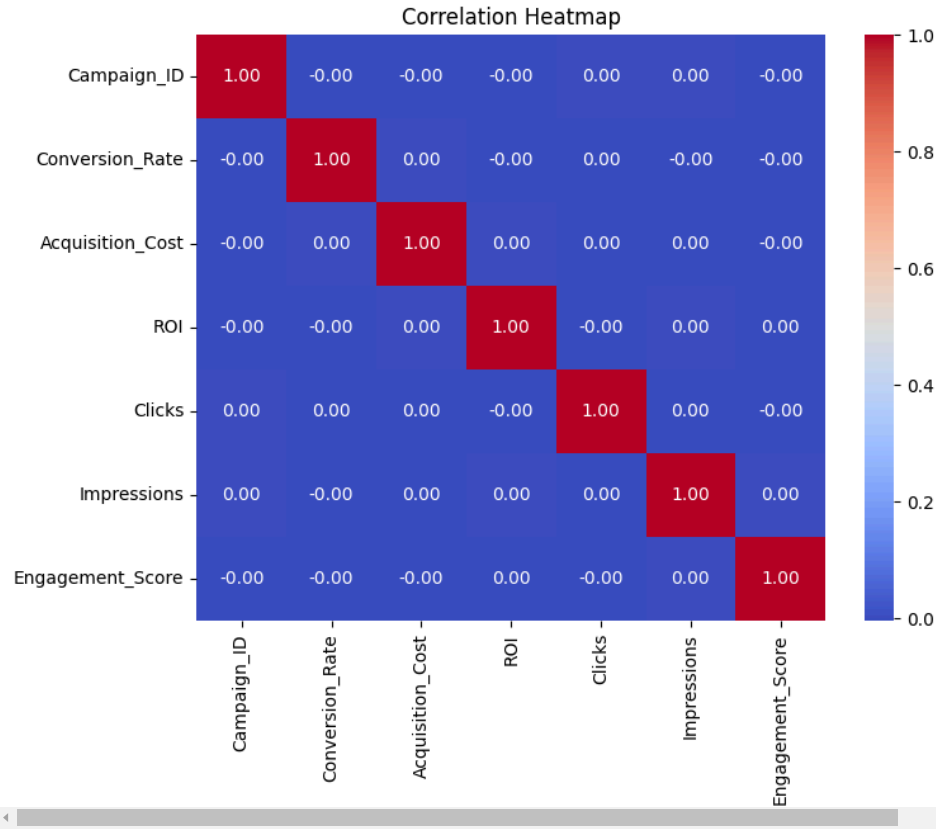
```
# Duplicate Check #

print(f"Number of duplicate rows: {df.duplicated().sum()}")
```

```
Number of duplicate rows: 0
```

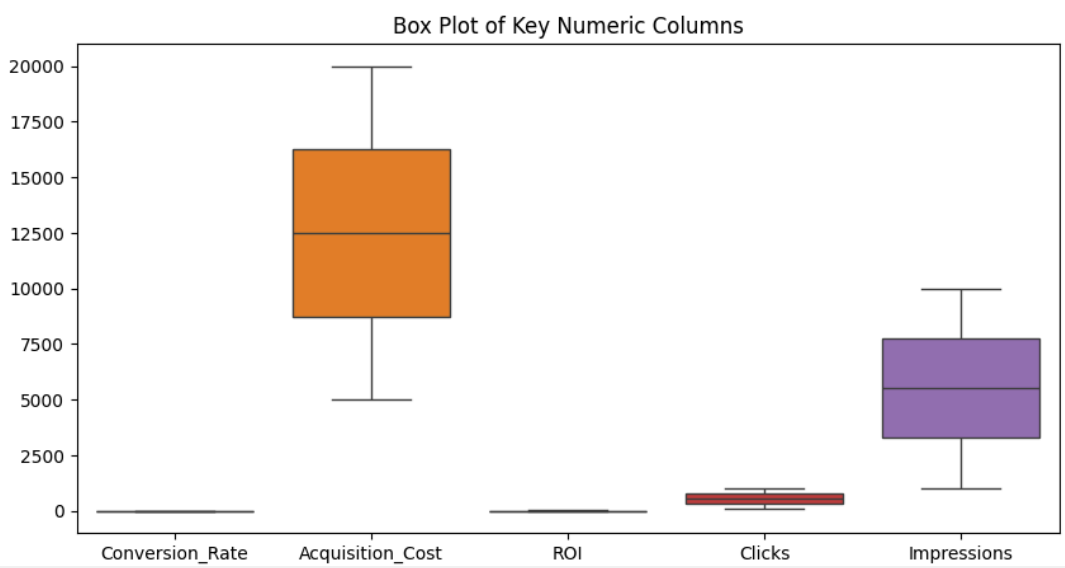
```
# Check Unique Campaign Types; Categorizing Performance Per Campaign Type #

print(df["Campaign_Type"].unique())
```

```
['Email' 'Influencer' 'Display' 'Search' 'Social Media']
```

```
# Check Correlation Between Variables #

import matplotlib.pyplot as plt
import seaborn as sns

# Select only numeric columns for correlation
numeric_df = df.select_dtypes(include=["number"])

# Plot heatmap
plt.figure(figsize=(8,6))
sns.heatmap(numeric_df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```




```
# Check for Outliers #

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
sns.boxplot(data=df[["Conversion_Rate", "Acquisition_Cost", "ROI", "Clicks", "Impressions"]])
plt.title("Box Plot of Key Numeric Columns")
plt.show()
```




```
# File Import #

from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Stage2_Task - marketing_campaign_dataset_Excel.xlsx to Stage2_Task - marketing_campaign_dataset_Excel.xlsx
```

```
# File Upload #

df = pd.read_excel("Stage2_Task - marketing_campaign_dataset_Excel.xlsx")
df.head()
```

| | Campaign_ID | Company | Campaign_Type | Target_Audience | Duration | Channel_Used | Conversion_Rate | Acquisition_Cost | ROI | Location | Date | Cl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Innovate Industries | Email | Men 18-24 | 30 days | Google Ads | 0.04 | 16174 | 6.29 | Chicago | 2021-01-01 00:00:00 | |
| **1** | 2 | NexGen Systems | Email | Women 35-44 | 60 days | Google Ads | 0.12 | 11566 | 5.61 | New York | 2021-02-01 00:00:00 | |
| **2** | 3 | Alpha Innovations | Influencer | Men 25-34 | 30 days | YouTube | 0.07 | 10200 | 7.18 | Los Angeles | 2021-03-01 00:00:00 | |
| **3** | 4 | DataTech Solutions | Display | All Ages | 60 days | YouTube | 0.11 | 12724 | 5.55 | Miami | 2021-04-01 00:00:00 | |
| **4** | 5 | NexGen Systems | Email | Men 25-34 | 15 days | YouTube | 0.05 | 16452 | 6.50 | Los Angeles | 2021-05-01 00:00:00 | |

```
# ANALYZING DATA TYPES

# Check the data types of each column
print(df.dtypes)
```

```
Campaign_ID           int64
Company              object
Campaign_Type        object
Target_Audience      object
Duration             object
Channel_Used         object
Conversion_Rate     float64
Acquisition_Cost      int64
ROI                 float64
Location             object
Date                 object
Clicks                int64
Impressions           int64
Engagement_Score      int64
Customer_Segment     object
CTR                 float64
CPC                 float64
dtype: object
```

```
#AVOIDING CONSTANT RELOADING

# Save The File:
df.to_csv("backup.csv", index=False)

# Reload Without Excel Processing
df = pd.read_csv("backup.csv")
```

```
#  IDENTIFICATION OF OUTLIEERS USING INTERQUARTILE (IQR) METHOD

# Function to detect outliers using IQR
def detect_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    return outliers

    # Check for outliers in key numeric columns
for col in ["Conversion_Rate", "Acquisition_Cost", "ROI", "Clicks", "Impressions"]:
```

```
    outliers = detect_outliers(df, col)
    print(f"{col}: {len(outliers)} outliers found")
```

```
Conversion_Rate: 0 outliers found
Acquisition_Cost: 0 outliers found
ROI: 0 outliers found
Clicks: 0 outliers found
Impressions: 0 outliers found
```

```
# UNIQUE TARGET AUDIENCES AND MARKETING CHANNELS

# Check unique values in Target_Audience and Channel_Used #

print("Unique Target Audiences:", df["Target_Audience"].unique())
print("\nUnique Marketing Channels:", df["Channel_Used"].unique())
```

```
Unique Target Audiences: ['Men 18-24' 'Women 35-44' 'Men 25-34' 'All Ages' 'Women 25-34']

Unique Marketing Channels: ['Google Ads' 'YouTube' 'Instagram' 'Website' 'Facebook' 'Email']
```

```
#  COMPARING CAMPAIGN PERFORMANCE ACROSS DIFFERENT CHANNELS

# Group by Channel and calculate mean performance metrics #

channel_performance = df.groupby("Channel_Used").agg({
    "Clicks": "sum",
    "Impressions": "sum",
    "Acquisition_Cost": "sum",
    "Conversion_Rate": "mean",
    "ROI": "mean"
}).reset_index()

# Calculate additional performance metrics #

channel_performance["CTR"] = (channel_performance["Clicks"] / channel_performance["Impressions"]) * 100  # Click-Through Rate
channel_performance["CPC"] = channel_performance["Acquisition_Cost"] / channel_performance["Clicks"]  # Cost Per Click

print(channel_performance)
```

```
  Channel_Used    Clicks  Impressions  Acquisition_Cost  Conversion_Rate  \
0        Email  18493963    184801107         420874104         0.080282
1     Facebook  18038175    180662496         410603426         0.079990
2   Google Ads  18342589    185020154         418944514         0.080181
3    Instagram  18316654    183738455         417124850         0.079886
4      Website  18415351    183815901         416606897         0.080182
5      YouTube  18350935    183450845         416797090         0.079890

         ROI        CTR        CPC
0  4.996487  10.007496  22.757378
1  5.018672   9.984460  22.763025
2  5.003126   9.913833  22.839988
3  4.988706   9.968873  22.772983
4  5.014114  10.018367  22.622805
5  4.993720  10.003189  22.712581
```

```
# UNIQUE MARKETING CHANNEL
df["Channel_Used"].unique()
```

```
array(['Google Ads', 'YouTube', 'Instagram', 'Website', 'Facebook',
       'Email'], dtype=object)
```

```
# MISSING VALUE
df[["CTR", "CPC", "Conversion_Rate"]].isnull().sum()
```

| | 0 |
|---|---|
| **CTR** | 0 |
| **CPC** | 0 |
| **Conversion_Rate** | 0 |

**dtype:** int64

```
# CALCULATING CTR, CPC, AND CONVERSION RATES TO ASSESS CAMPAIGN EFFECTIVENESS.
df.groupby("Channel_Used")[["CTR", "CPC", "Conversion_Rate"]].mean()
```

| Channel_Used | CTR | CPC | Conversion_Rate |
|---|---|---|---|
| **Email** | 14.054269 | 31.881471 | 0.080282 |
| **Facebook** | 14.049724 | 32.129366 | 0.079990 |
| **Google Ads** | 13.918943 | 32.308459 | 0.080181 |
| **Instagram** | 14.003691 | 32.080786 | 0.079886 |
| **Website** | 14.096941 | 31.779148 | 0.080182 |
| **YouTube** | 14.119755 | 31.872904 | 0.079890 |

```
# Checking for Zero Clicks or Impressions #
print(df[df["Clicks"] == 0])  # See if any channels have zero clicks
print(df[df["Impressions"] == 0])  # See if any channels have zero impressions
```

```
Empty DataFrame
Columns: [Campaign_ID, Company, Campaign_Type, Target_Audience, Duration, Channel_Used, Conversion_Rate, Acquisition_Cost, ROI, Location, Date,
Index: []
Empty DataFrame
Columns: [Campaign_ID, Company, Campaign_Type, Target_Audience, Duration, Channel_Used, Conversion_Rate, Acquisition_Cost, ROI, Location, Date,
Index: []
```

```
#  IDENTIFYING HIGH-PERFORMING & UNDERPERFORMING CAMPAIGNS #

# Define high and low performing campaigns
high_performers = df[df["ROI"] > df["ROI"].quantile(0.75)]  # Top 25% ROI
low_performers = df[df["ROI"] < df["ROI"].quantile(0.25)]  # Bottom 25% ROI

print(f"Number of High-Performing Campaigns: {len(high_performers)}")
print(f"Number of Underperforming Campaigns: {len(low_performers)}")
```

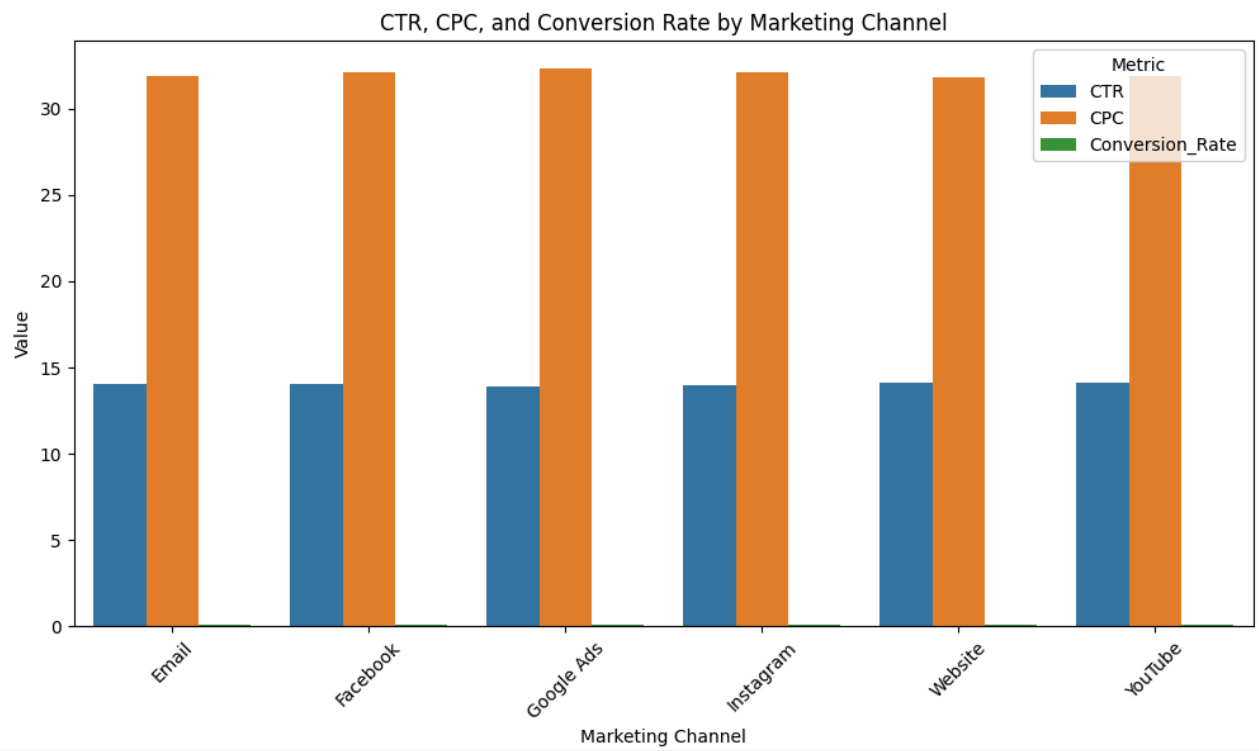
```
Number of High-Performing Campaigns: 49695
Number of Underperforming Campaigns: 49913
```

```
#EXPLORING LOCATION-BASED TRENDS

# Group by location to analyze trends
location_trends = df.groupby("Location").agg({
    "Clicks": "sum",
    "Impressions": "sum",
    "ROI": "mean",
    "Conversion_Rate": "mean"
}).reset_index()

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

print(location_trends.head(10))  # Display top 10 locations
```

```
       Location    Clicks  Impressions       ROI  Conversion_Rate
3         Miami  22056765    221347726  5.012282         0.080047
2   Los Angeles  21966553    219652325  5.010876         0.080013
1       Houston  21893075    219129799  5.007174         0.079949
0       Chicago  21980408    219999352  5.001555         0.080131
4      New York  22060866    221359756  4.980185         0.080203
```

```
# BAR CHART ASSESSING CAMPAIGN EFFECTIVENESS USING CTR, CPC, CR

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Create the data as a dictionary
data = {
    "Channel_Used": ["Email", "Facebook", "Google Ads", "Instagram", "Website", "YouTube"],
    "CTR": [14.054269, 14.049724, 13.918943, 14.003691, 14.096941, 14.119755],
    "CPC": [31.881471, 32.129366, 32.308459, 32.080786, 31.779148, 31.872904],
    "Conversion_Rate": [0.080282, 0.079990, 0.080181, 0.079886, 0.080182, 0.079890]
}

# Create DataFrame
df = pd.DataFrame(data)
```

```
# Melt the DataFrame for visualization
df_melted = df.melt(id_vars=["Channel_Used"], var_name="Metric", value_name="Value")

# Set figure size
plt.figure(figsize=(12, 6))

# Create barplot
sns.barplot(x="Channel_Used", y="Value", hue="Metric", data=df_melted)

# Add title and labels
plt.title("CTR, CPC, and Conversion Rate by Marketing Channel")
plt.xlabel("Marketing Channel")
plt.ylabel("Value")
plt.legend(title="Metric")

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




```
print(df.columns)
```

```
Index(['Channel_Used', 'CTR', 'CPC', 'Conversion_Rate'], dtype='object')
```

```
print(df.columns)  # Check available columns
print(df.head())   # See sample data
```

```
Index(['Channel_Used', 'CTR', 'CPC', 'Conversion_Rate'], dtype='object')
  Channel_Used        CTR        CPC  Conversion_Rate
0        Email  14.054269  31.881471         0.080282
1     Facebook  14.049724  32.129366         0.079990
2   Google Ads  13.918943  32.308459         0.080181
3    Instagram  14.003691  32.080786         0.079886
4      Website  14.096941  31.779148         0.080182
```

```
# File Import #

from google.colab import files
uploaded = files.upload()
```

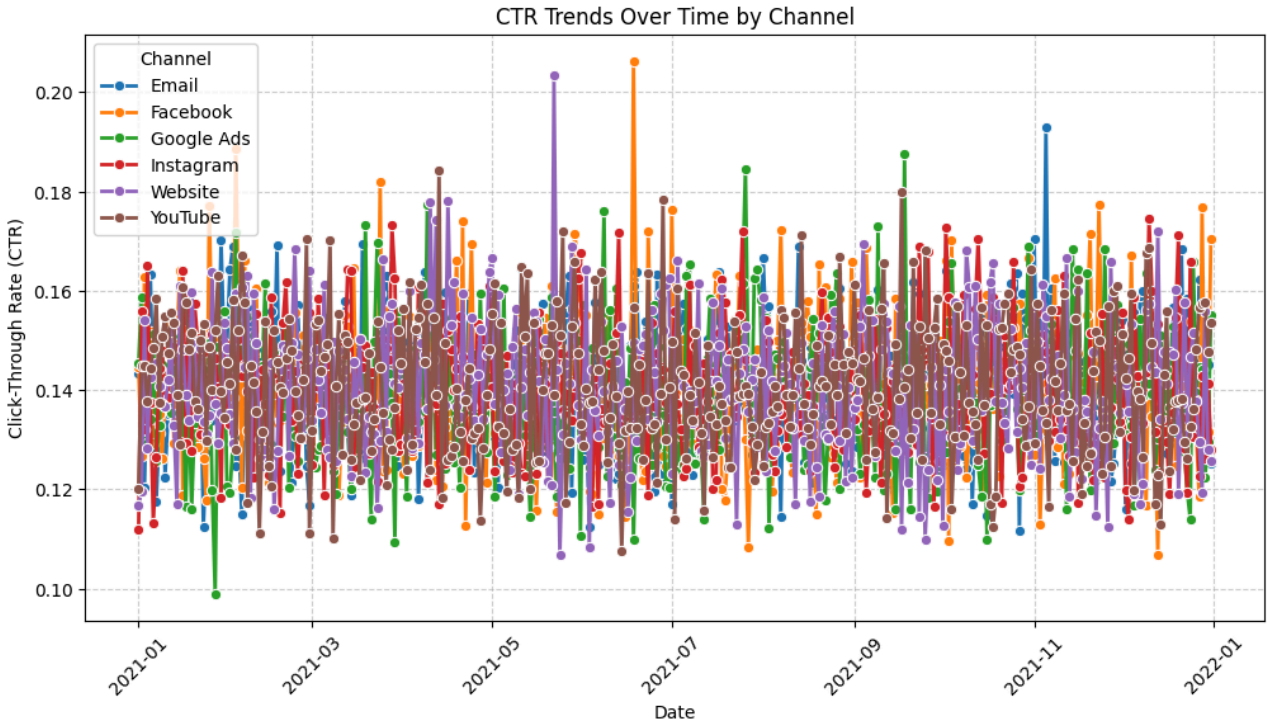
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

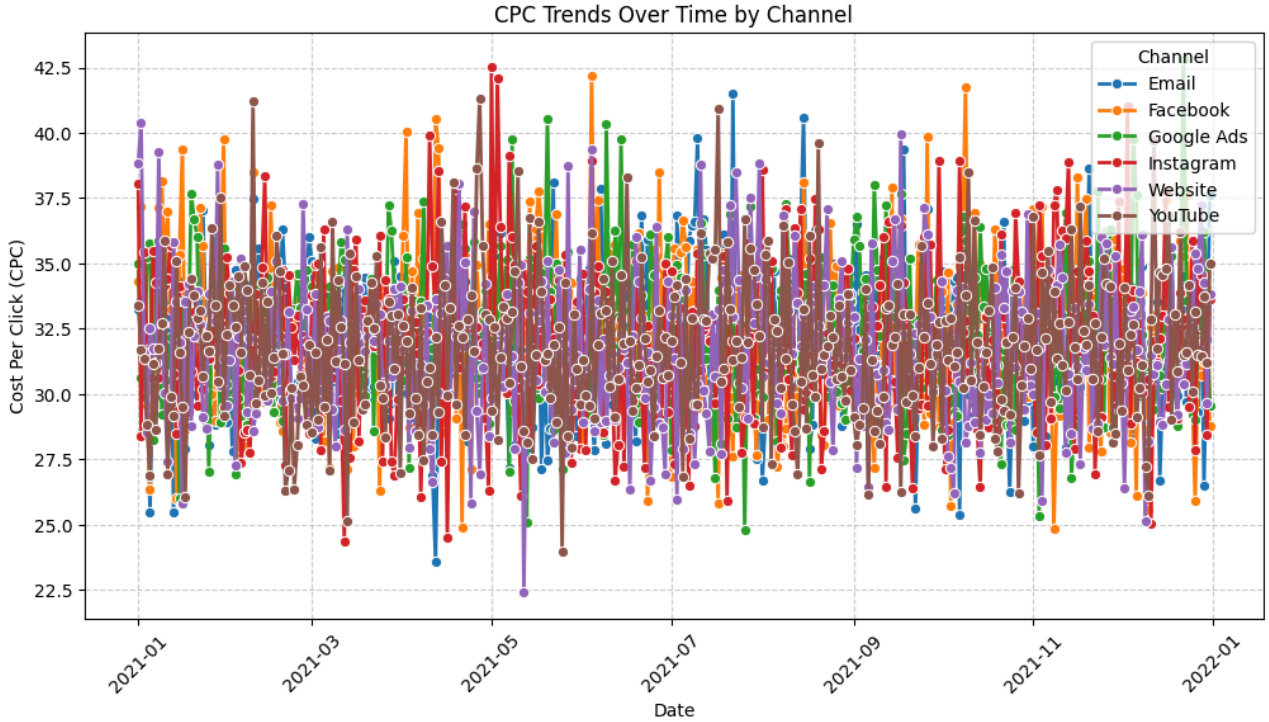
Saving Stage2 Task - marketing campaign dataset Excel.xlsx to Stage2 Task - marketing campaign dataset Excel (1).xlsx

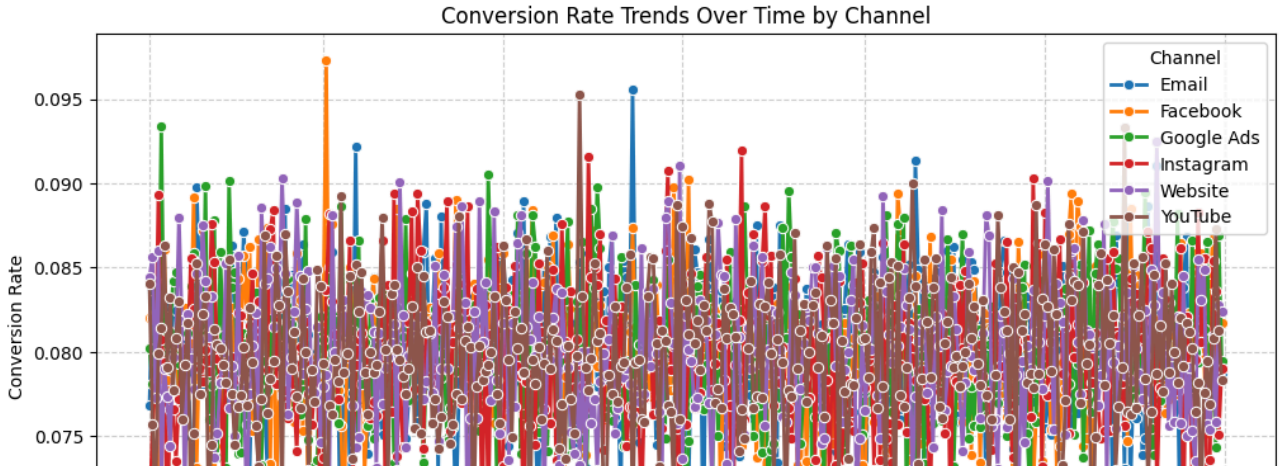
```
# File Upload #

df = pd.read_excel("Stage2_Task - marketing_campaign_dataset_Excel.xlsx")
df.head()
```

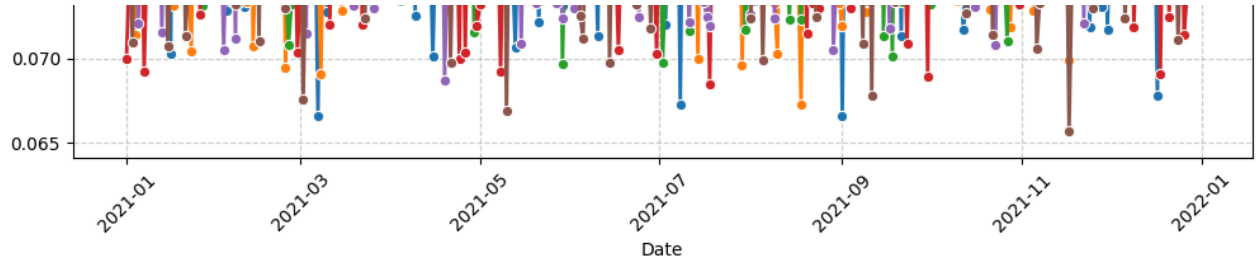
| | Campaign_ID | Company | Campaign_Type | Target_Audience | Duration | Channel_Used | Conversion_Rate | Acquisition_Cost | ROI | Location | Date | Cl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Innovate Industries | Email | Men 18-24 | 30 days | Google Ads | 0.04 | 16174 | 6.29 | Chicago | 2021-01-01 00:00:00 | |
| **1** | 2 | NexGen Systems | Email | Women 35-44 | 60 days | Google Ads | 0.12 | 11566 | 5.61 | New York | 2021-02-01 00:00:00 | |
| **2** | 3 | Alpha Innovations | Influencer | Men 25-34 | 30 days | YouTube | 0.07 | 10200 | 7.18 | Los Angeles | 2021-03-01 00:00:00 | |
| **3** | 4 | DataTech Solutions | Display | All Ages | 60 days | YouTube | 0.11 | 12724 | 5.55 | Miami | 2021-04-01 00:00:00 | |
| **4** | 5 | NexGen Systems | Email | Men 25-34 | 15 days | YouTube | 0.05 | 16452 | 6.50 | Los Angeles | 2021-05-01 00:00:00 | |

```
# LINE CHART COMPARING PERFORMANCE TRENDS ACROSS CHANNELS OVER TIME.

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load the dataset
file_path = "Stage2_Task - marketing_campaign_dataset_Excel.xlsx"
df = pd.read_excel(file_path, sheet_name="marketing_campaign_dataset")

# Convert 'Date' column to datetime format
df["Date"] = pd.to_datetime(df["Date"])

# Compute additional metrics
df["CTR"] = df["Clicks"] / df["Impressions"]
df["CPC"] = df["Acquisition_Cost"] / df["Clicks"]

# Group by Date and Channel, calculating the mean for each metric
time_series_data = df.groupby(["Date", "Channel_Used"])[["CTR", "CPC", "Conversion_Rate"]].mean().reset_index()

# Function to create line charts for different metrics
def plot_line_chart(metric, ylabel, title):
    plt.figure(figsize=(12, 6))
    sns.lineplot(data=time_series_data, x="Date", y=metric, hue="Channel_Used", marker="o", linewidth=2)
    plt.title(title)
    plt.xlabel("Date")
    plt.ylabel(ylabel)
    plt.xticks(rotation=45)
    plt.grid(True, linestyle="--", alpha=0.6)
    plt.legend(title="Channel")
    plt.show()

# Generate line charts for CTR, CPC, and Conversion Rate
plot_line_chart("CTR", "Click-Through Rate (CTR)", "CTR Trends Over Time by Channel")
plot_line_chart("CPC", "Cost Per Click (CPC)", "CPC Trends Over Time by Channel")
plot_line_chart("Conversion_Rate", "Conversion Rate", "Conversion Rate Trends Over Time by Channel")
```

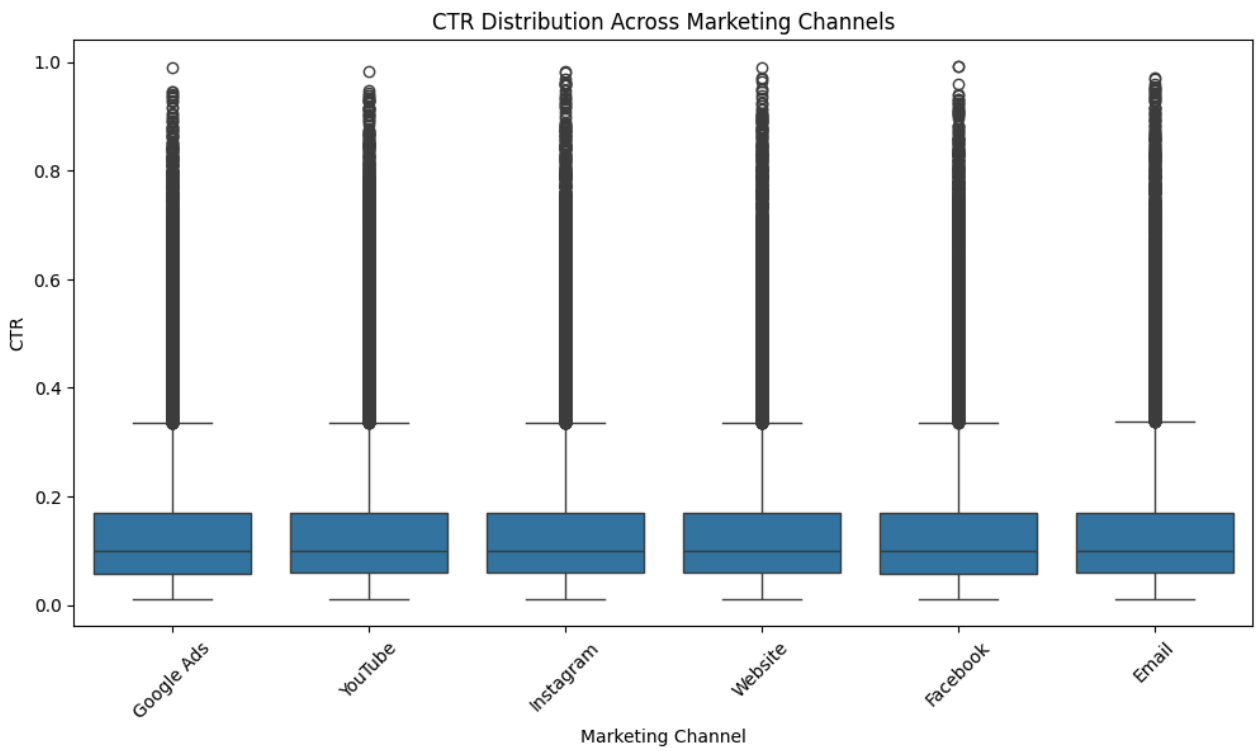

CTR Trends Over Time by Channel
Channel
Email
Facebook
Google Ads
Instagram
Website
YouTube
Click-Through Rate (CTR)
0.20
0.18
0.16
0.14
0.12
0.10
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01
Date


CPC Trends Over Time by Channel
Channel
Email
Facebook
Google Ads
Instagram
Website
YouTube
Cost Per Click (CPC)
42.5
40.0
37.5
35.0
32.5
30.0
27.5
25.0
22.5
2021-01
2021-03
2021-05
2021-07
2021-09
2021-11
2022-01
Date


Conversion Rate Trends Over Time by Channel
Channel
Email
Facebook
Google Ads
Instagram
Website
YouTube
Conversion Rate
0.095
0.090
0.085
0.080
0.075

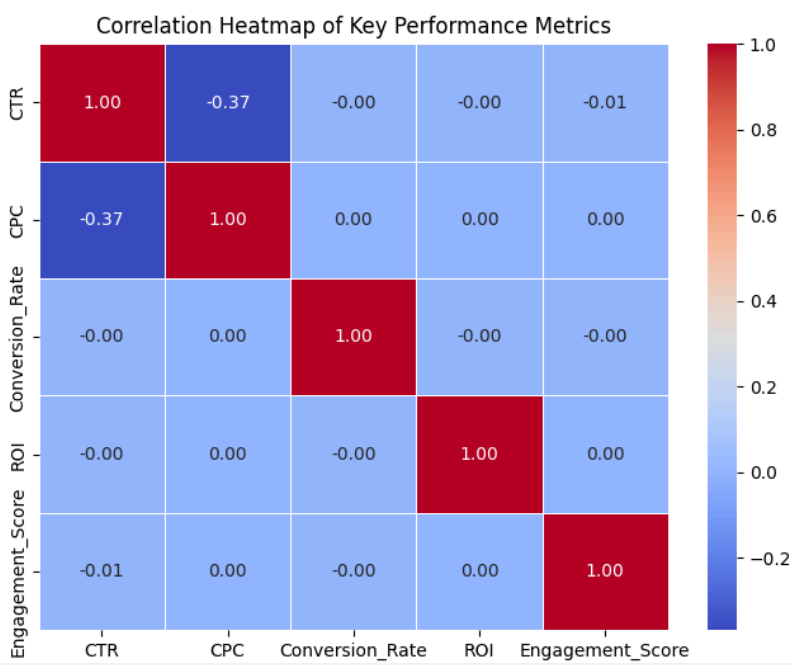



```
# BOX PLOT FOR CTR PER CHANNEL

plt.figure(figsize=(12, 6))
sns.boxplot(x="Channel_Used", y="CTR", data=df)  # Change CTR to CPC, ROI, etc. as needed
plt.title("CTR Distribution Across Marketing Channels")
plt.xlabel("Marketing Channel")
plt.ylabel("CTR")
plt.xticks(rotation=45)
plt.show()
```




```
# HEATMAP FOR CHANNELS PERFORMANCE IN DIFFERENT METRICS

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = "Stage2_Task - marketing_campaign_dataset_Excel.xlsx"
df = pd.read_excel(file_path, sheet_name="marketing_campaign_dataset")

# Convert 'Date' column to datetime format (if applicable)
df["Date"] = pd.to_datetime(df["Date"])

# Compute additional metrics if not already available
df["CTR"] = df["Clicks"] / df["Impressions"]
df["CPC"] = df["Acquisition_Cost"] / df["Clicks"]

# Select relevant numerical columns for correlation analysis
correlation_columns = ["CTR", "CPC", "Conversion_Rate", "ROI", "Engagement_Score"]
df_corr = df[correlation_columns].corr()

# Create the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(df_corr, annot=True, cmap="coolwarm", linewidths=0.5, fmt=".2f")
plt.title("Correlation Heatmap of Key Performance Metrics")
```

```
plt.show()
```




```
import matplotlib.pyplot as plt  # Import Matplotlib
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Stage2_Task - marketing_campaign_dataset_Excel.xlsx to Stage2_Task - marketing_campaign_dataset_Excel.xlsx
```

```
df = pd.read_excel("Stage2_Task - marketing_campaign_dataset_Excel.xlsx")
```

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

print("Libraries imported successfully!")
```

```
Libraries imported successfully!
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files Stage2_Tas…t_Excel.xlsx

- **Stage2_Task - marketing_campaign_dataset_Excel.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 17343447 bytes, last modified: 2/12/2025 - 100% done

```
import pandas as pd

# If you're using an Excel file:
df = pd.read_excel("Stage2_Task - marketing_campaign_dataset_Excel.xlsx")

# If you're using a CSV file:
# df = pd.read_csv("your_file.csv")
```

```
# HIGH PERFORMING & LOW PERFORMING CAMPAIGN TYPES BASED ON ROI #

# Define threshold for high and low performance
high_threshold = 6
low_threshold = 3
```

```
# Categorize campaigns
df["Performance_Category"] = df["ROI"].apply(lambda x:
    "High Performing" if x >= high_threshold else ("Underperforming" if x <= low_threshold else "Average"))

print(df["Performance_Category"].value_counts())
```

```
Performance_Category
Average            99311
High Performing    67064
Underperforming    33630
Name: count, dtype: int64
```

```
print(df["Campaign_Type"].unique())
```

```
['Email' 'Influencer' 'Display' 'Search' 'Social Media']
```

```
# COMPARING PERFORMIMG ACROSS CATEGORIES #

performance_by_type = df.groupby("Campaign_Type")["Performance_Category"].value_counts().unstack()
print(performance_by_type)
```

```
Performance_Category  Average  High Performing  Underperforming
Campaign_Type
Display                 20054             9935             9999
Email                   19976             9868            10027
Influencer              19934            10186            10050
Search                  19908            10176            10073
Social Media            19839             9868            10112
```

```
# TOP 10 PERFORMERS BASED ON ROI #
high_performers = df[df["Performance_Category"] == "High Performing"]
print(high_performers[["Campaign_ID", "Campaign_Type", "ROI"]].head(10))  # Show top 10
```

```
     Campaign_ID Campaign_Type   ROI
2              3    Influencer  7.18
8              9  Social Media  6.73
13            14         Email  7.06
20            21        Search  7.99
23            24         Email  7.31
28            29    Influencer  7.12
31            32    Influencer  6.83
34            35    Influencer  7.81
36            37       Display  7.24
39            40        Search  6.97
```

```
# BOTTOM 10 PERFORMERS BASED ON ROI #

low_performers = df[df["Performance_Category"] == "Underperforming"]
print(low_performers[["Campaign_ID", "Campaign_Type", "ROI"]].head(10))  # Show bottom 10
```

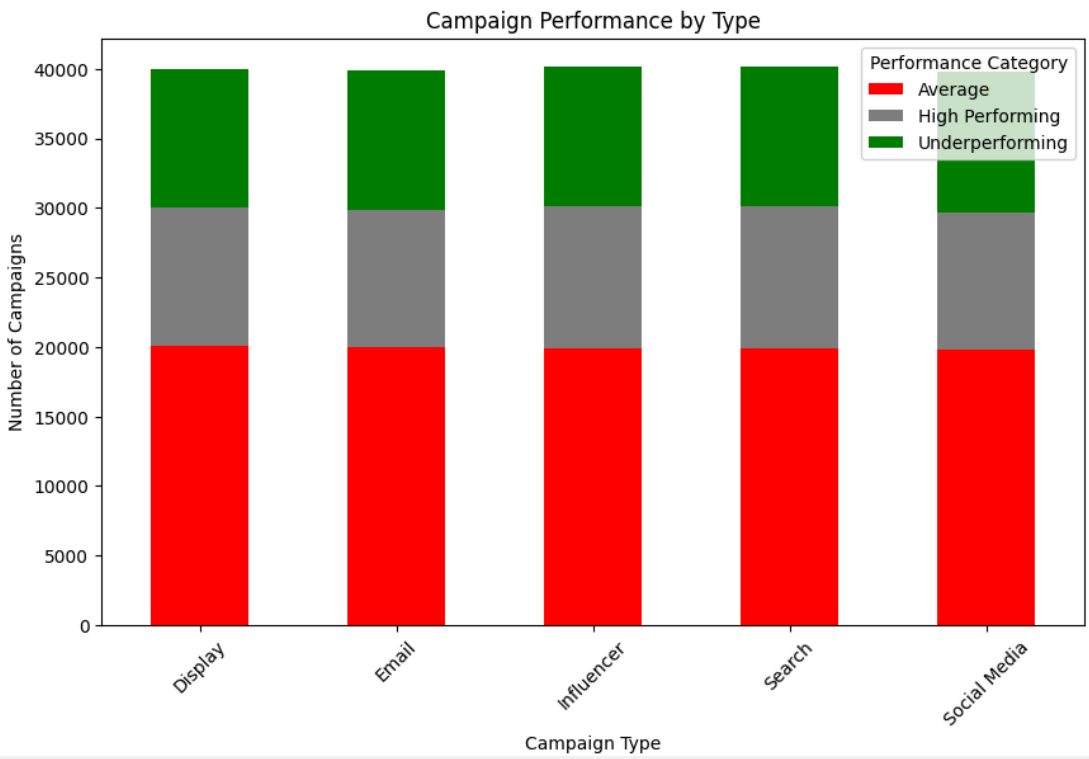
```
     Campaign_ID Campaign_Type   ROI
6              7         Email  2.86
10            11       Display  3.49
15            16  Social Media  2.91
22            23  Social Media  2.12
27            28         Email  3.29
30            31        Search  2.77
33            34    Influencer  2.71
35            36       Display  2.51
37            38    Influencer  2.62
38            39        Search  2.21
```

```
average_performers = df[df["Performance_Category"] == "Average"]
print(average_performers[["Campaign_ID", "Campaign_Type", "ROI"]].head(10))  # Show top 10
```

```
     Campaign_ID Campaign_Type   ROI
0              1         Email  6.29
1              2         Email  5.61
3              4       Display  5.55
4              5         Email  6.50
5              6       Display  4.36
7              8        Search  5.55
9             10         Email  3.78
11            12    Influencer  3.59
```

```
12           13  Social Media  4.91
14           15       Display  5.28
```

```
# CAMPAGIN PERFORMANCE BY TYPE: STACKED BAR CHART #

import matplotlib.pyplot as plt

# Group by Campaign_Type and count Performance_Category
performance_by_type = df.groupby("Campaign_Type")["Performance_Category"].value_counts().unstack()

# Plot a stacked bar chart
performance_by_type.plot(kind="bar", stacked=True, figsize=(10, 6), color=["red", "gray", "green"])

# Customize chart
plt.title("Campaign Performance by Type")
plt.xlabel("Campaign Type")
plt.ylabel("Number of Campaigns")
plt.legend(title="Performance Category")
plt.xticks(rotation=45)
plt.show()
```




```
# BEST & WORST PERFORMING CAMPAIGN TYPES #

# Count high-performing campaigns by type
high_performance_counts = df[df["Performance_Category"] == "High Performing"]["Campaign_Type"].value_counts()

# Count underperforming campaigns by type
low_performance_counts = df[df["Performance_Category"] == "Underperforming"]["Campaign_Type"].value_counts()

# Identify the best and worst performing campaign types
best_performing_campaign = high_performance_counts.idxmax()
worst_performing_campaign = low_performance_counts.idxmax()

print(f"Best Performing Campaign Type: {best_performing_campaign} ({high_performance_counts.max()} campaigns)")
print(f"Worst Performing Campaign Type: {worst_performing_campaign} ({low_performance_counts.max()} campaigns)")
```

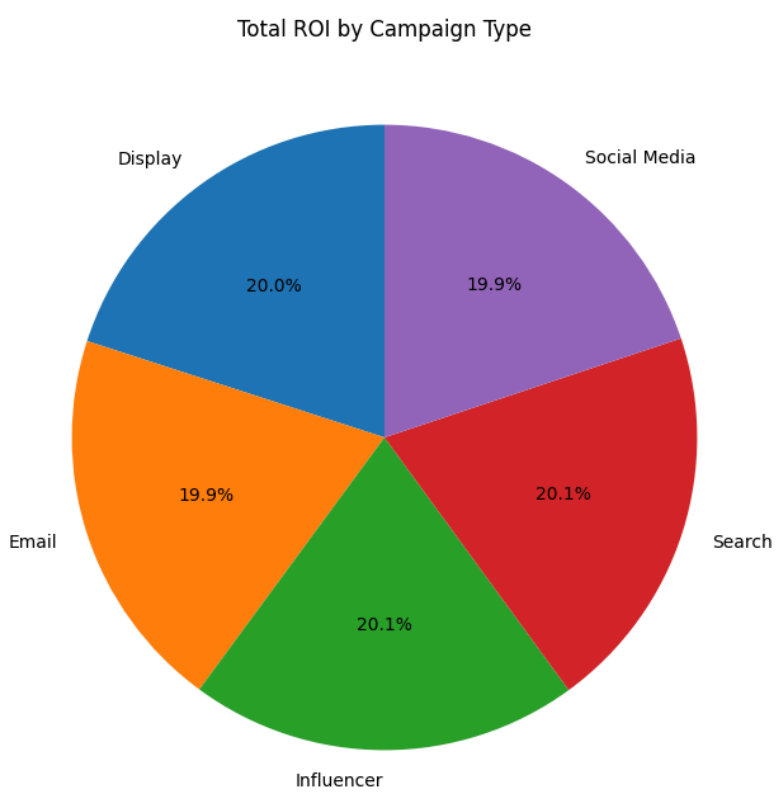
```
Best Performing Campaign Type: Influencer (10186 campaigns)
Worst Performing Campaign Type: Social Media (10112 campaigns)
```

```
# TOTAL ROI OF EACH CAMPAIGN TYPE#

# Calculate the total ROI for each campaign type
total_roi_by_campaign = df.groupby("Campaign_Type")["ROI"].sum()

# Display the results
```

```
print(total_roi_by_campaign)
```

```
Campaign_Type
Display         200199.80
Email           199126.70
Influencer      201293.46
Search          201120.60
Social Media    198767.74
Name: ROI, dtype: float64
```

```
# PIE CHART FOR CAMPAIGN TYPE PERFORMANCE

import matplotlib.pyplot as plt

# Calculate the total ROI for each campaign type
total_roi_by_campaign = df.groupby("Campaign_Type")["ROI"].sum()

# Plot a pie chart
plt.figure(figsize=(8, 8))
plt.pie(total_roi_by_campaign, labels=total_roi_by_campaign.index, autopct='%1.1f%%', startangle=90)
plt.title("Total ROI by Campaign Type")
plt.show()
```




```
# CAMPAIGN PERFORMANCE BASED ON COMPANY, CHANNEL USED & CUSTOMER SEGMENT #

import pandas as pd

# 1. Performance by Company
company_performance = df.groupby("Company").agg(
    Avg_ROI=("ROI", "mean"),
    Min_ROI=("ROI", "min"),
    Max_ROI=("ROI", "max"),
    Avg_Conversion_Rate=("Conversion_Rate", "mean"),
    Total_Campaigns=("Campaign_ID", "count")
)
print("=== Performance by Company ===")
print(company_performance)
print("-" * 50)

# 2. Performance by Channel Used
channel_performance = df.groupby("Channel_Used").agg(
    Avg_ROI=("ROI", "mean"),
    Min_ROI=("ROI", "min"),
```

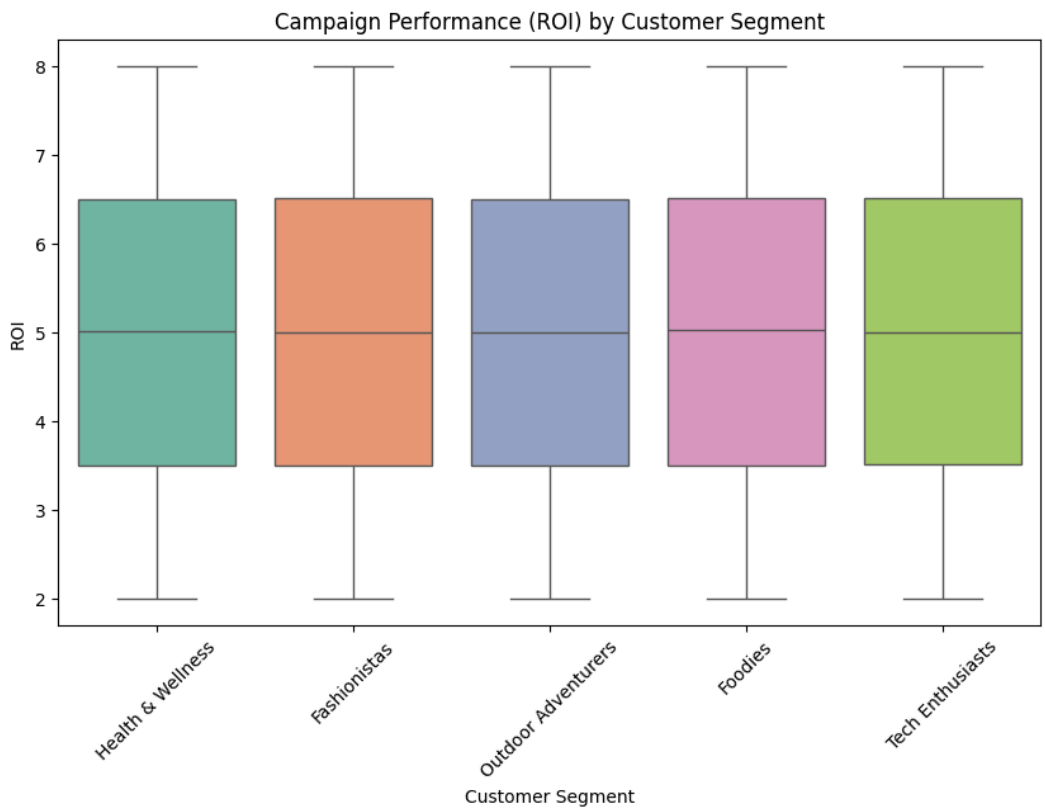
```
    Max_ROI=("ROI", "max"),
    Avg_Conversion_Rate=("Conversion_Rate", "mean"),
    Total_Campaigns=("Campaign_ID", "count")
)
print("=== Performance by Channel ===")
print(channel_performance)
print("-" * 50)

# 3. Performance by Customer Segment
segment_performance = df.groupby("Customer_Segment").agg(
    Avg_ROI=("ROI", "mean"),
    Min_ROI=("ROI", "min"),
    Max_ROI=("ROI", "max"),
    Avg_Conversion_Rate=("Conversion_Rate", "mean"),
    Total_Campaigns=("Campaign_ID", "count")
)
print("=== Performance by Customer Segment ===")
print(segment_performance)
print("-" * 50)

# (Optional) Pivot Table Examples
# -- These help you compare performance across two dimensions at once.
company_channel_pivot = pd.pivot_table(
    df,
    index="Company",
    columns="Channel_Used",
    values="ROI",
    aggfunc="mean"
)
print("=== Average ROI by Company & Channel ===")
print(company_channel_pivot)
print("-" * 50)

segment_channel_pivot = pd.pivot_table(
    df,
    index="Customer_Segment",
    columns="Channel_Used",
    values="ROI",
    aggfunc="mean"
)
print("=== Average ROI by Customer Segment & Channel ===")
print(segment_channel_pivot)
```

```
=== Performance by Company ===
                      Avg_ROI  Min_ROI  Max_ROI  Avg_Conversion_Rate  \
Company
Alpha Innovations    5.005944      2.0      8.0             0.080084
DataTech Solutions   5.005472      2.0      8.0             0.079987
Innovate Industries  5.002188      2.0      8.0             0.080383
NexGen Systems       4.991353      2.0      8.0             0.079736
TechCorp             5.007089      2.0      8.0             0.080156

                     Total_Campaigns
Company
Alpha Innovations              40051
DataTech Solutions             40014
Innovate Industries            39711
NexGen Systems                 39991
TechCorp                       40238
--------------------------------------------------
=== Performance by Channel ===
                 Avg_ROI  Min_ROI  Max_ROI  Avg_Conversion_Rate  Total_Campaigns
Channel_Used
Email         4.996487      2.0      8.0             0.080282            33599
Facebook      5.018672      2.0      8.0             0.079990            32820
Google Ads    5.003126      2.0      8.0             0.080181            33440
Instagram     4.988706      2.0      8.0             0.079886            33392
Website       5.014114      2.0      8.0             0.080182            33361
YouTube       4.993720      2.0      8.0             0.079890            33393
--------------------------------------------------
=== Performance by Customer Segment ===
                      Avg_ROI  Min_ROI  Max_ROI  Avg_Conversion_Rate  \
Customer_Segment
Fashionistas         5.000962      2.0      8.0             0.079794
Foodies              5.004326      2.0      8.0             0.080256
Health & Wellness    5.003202      2.0      8.0             0.079945
Outdoor Adventurers  4.999393      2.0      8.0             0.080180
Tech Enthusiasts     5.004177      2.0      8.0             0.080165

                     Total_Campaigns
Customer_Segment
Fashionistas                 39742
Foodies                      40210
Health & Wellness            39888
```

```
Outdoor Adventurers            40011
Tech Enthusiasts               40154
--------------------------------------------------
=== Average ROI by Company & Channel ===
Channel_Used            Email  Facebook  Google Ads  Instagram   Website  \
Company
Alpha Innovations    5.000935  5.020388    4.991662   4.976576  5.009910
DataTech Solutions   4.990800  5.014450    5.013922   4.989243  5.040937
Innovate Industries  5.022140  5.019962    4.995254   4.988053  5.009129
NexGen Systems       4.993706  5.006621    5.008727   4.970395  4.991802
TechCorp             4.974931  5.031797    5.005673   5.019418  5.019520

Channel_Used          YouTube
Company
Alpha Innovations    5.036764
DataTech Solutions   4.983959
Innovate Industries  4.978553
```

```
# CAMPAIGN PERFORMANCE BASED ON CUSTOMER SEGMENT #

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.boxplot(x="Customer_Segment", y="ROI", data=df, palette="Set2")
plt.title("Campaign Performance (ROI) by Customer Segment")
plt.xlabel("Customer Segment")
plt.ylabel("ROI")
plt.xticks(rotation=45)
plt.show()
```

```
<ipython-input-22-ede5eafee476>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False`

  sns.boxplot(x="Customer_Segment", y="ROI", data=df, palette="Set2")
```




```
# CAMPAIGN PERFORMANCE BY CHANNEL USED AND CUSTOMER SEGMENT

segment_channel_pivot = pd.pivot_table(
    df,
    index="Customer_Segment",
    columns="Channel_Used",
    values="ROI",
    aggfunc="mean"
)
```

```
print("=== Average ROI by Customer Segment & Channel ===")
print(segment_channel_pivot)
```

```
=== Average ROI by Customer Segment & Channel ===
Channel_Used             Email  Facebook  Google Ads  Instagram   Website  \
Customer_Segment
Fashionistas         4.987897  5.004506    5.029717   5.010496  5.020863
Foodies              4.991678  5.009535    4.969647   4.998445  5.033461
Health & Wellness    5.021867  5.006555    5.019812   4.968776  5.014928
Outdoor Adventurers  4.970791  5.022944    5.004235   4.993492  5.009121
Tech Enthusiasts     5.010684  5.049569    4.992254   4.972552  4.992719

Channel_Used          YouTube
Customer_Segment
Fashionistas         4.952345
Foodies              5.023932
Health & Wellness    4.987938
Outdoor Adventurers  4.996338
Tech Enthusiasts     5.007916
```

```
# CR TRENDS ACROSS LOCATIONS OVER TIME #
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Convert the Date column to datetime format (using day-first format)
df["Date"] = pd.to_datetime(df["Date"], dayfirst=True, errors="coerce")

plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x="Date", y="Conversion_Rate", hue="Location", marker="o")
plt.title("Conversion Rate Trends Across Locations Over Time")
plt.xlabel("Date")
plt.ylabel("Conversion Rate")
plt.xticks(rotation=45)
plt.legend(title="Location")
plt.tight_layout()
plt.show()
```