ab_testing-campaigns

March 27, 2025

Project Goal

Analyze the performance of two campaigns (control and test) based on the provided dataset. The primary objective is to determine which campaign delivers better results across key metrics.

Project Steps

Preliminary Data Processing

Load and study the dataset structure. Handle any missing values if present. Standardize data formats (e.g., dates). Check for outliers and correct them.

Calculation of Key Metrics

For each campaign, calculate: Average and median expenditure (Spend). Average impressions (Impressions), unique impressions (Reach), and website clicks (Website Clicks). Calculate cost per purchase. Conversion rates for each stage: Conversion from impressions (Impressions) to website clicks (Website Clicks). Conversion from clicks to content views (View Content). Conversion from cart additions (Add to Cart) to purchases (Purchase). Conversion from website clicks to purchases.

Comparative Analysis of Campaigns

Compare the results of the control and test campaigns across key metrics (e.g., cost per acquisition, conversion rate from website clicks to purchases). Identify strengths and weaknesses of each campaign.

Statistical Testing

Conduct an appropriate test to determine the statistical significance of differences between the campaigns. Evaluate the p-value and decide if the differences are significant.

Power Analysis

Verify if the data is sufficient to draw reliable conclusions. Assess whether the test duration was adequate.

Data Visualization

Create charts to present results: Compare expenditures, impressions, clicks, and conversions. Visualize changes across funnel stages.

Project Results

Conclusions about statistically significant differences between the campaigns. Recommendations for marketing strategy.

Dataset Features:

Campaign Name: The name of the campaign. Date: The date of record. Spend: Amount spent on the campaign in dollars. Impressions: Number of ad impressions received during the campaign. Reach: Number of unique ad impressions. Website Clicks: Number of clicks on the website received via ads. Searches: Number of users who performed searches on the site. View Content: Number of users who viewed content and products on the site. Add to Cart: Number of users who added products to the cart. Purchase: Number of purchases.

[1415]:
```
import numpy as np
import pandas as pd
import scipy as sp
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import mannwhitneyu
from statsmodels.stats.power import TTestIndPower
from statsmodels.stats.power import zt_ind_solve_power
```

[1417]:
```
control_df = pd.read_csv("control_group.csv")
test_df = pd.read_csv("test_group.csv")
```

[1419]:
```
len(control_df) == len(test_df)
```

[1419]: True

Data cleaning

[1422]:
```
df = pd.concat([control_df, test_df], ignore_index=True)
```

[1424]:
```
df
```

[1424]:

	Campaign Name	Date	Spend [USD]	# of Impressions	Reach	\
0	Control Campaign	1.08.2019	2280	82702.0	56930.0	
1	Control Campaign	2.08.2019	1757	121040.0	102513.0	
2	Control Campaign	3.08.2019	2343	131711.0	110862.0	
3	Control Campaign	4.08.2019	1940	72878.0	61235.0	
4	Control Campaign	5.08.2019	1835	NaN	NaN	
5	Control Campaign	6.08.2019	3083	109076.0	87998.0	
6	Control Campaign	7.08.2019	2544	142123.0	127852.0	
7	Control Campaign	8.08.2019	1900	90939.0	65217.0	
8	Control Campaign	9.08.2019	2813	121332.0	94896.0	
9	Control Campaign	10.08.2019	2149	117624.0	91257.0	
10	Control Campaign	11.08.2019	2490	115247.0	95843.0	
11	Control Campaign	12.08.2019	2319	116639.0	100189.0	
12	Control Campaign	13.08.2019	2697	82847.0	68214.0	
13	Control Campaign	14.08.2019	1875	145248.0	118632.0	
14	Control Campaign	15.08.2019	2774	132845.0	102479.0	
15	Control Campaign	16.08.2019	2024	71274.0	42859.0	
16	Control Campaign	17.08.2019	2177	119612.0	106518.0	
17	Control Campaign	18.08.2019	1876	108452.0	96518.0	

18	Control Campaign	19.08.2019	2596	107890.0	81268.0
19	Control Campaign	20.08.2019	2675	113430.0	78625.0
20	Control Campaign	21.08.2019	1803	74654.0	59873.0
21	Control Campaign	22.08.2019	2939	105705.0	86218.0
22	Control Campaign	23.08.2019	2496	129880.0	109413.0
23	Control Campaign	24.08.2019	1892	72515.0	51987.0
24	Control Campaign	25.08.2019	1962	117006.0	100398.0
25	Control Campaign	26.08.2019	2233	124897.0	98432.0
26	Control Campaign	27.08.2019	2061	104678.0	91579.0
27	Control Campaign	28.08.2019	2421	141654.0	125874.0
28	Control Campaign	29.08.2019	2375	92029.0	74192.0
29	Control Campaign	30.08.2019	2324	111306.0	88632.0
30	Test Campaign	1.08.2019	3008	39550.0	35820.0
31	Test Campaign	2.08.2019	2542	100719.0	91236.0
32	Test Campaign	3.08.2019	2365	70263.0	45198.0
33	Test Campaign	4.08.2019	2710	78451.0	25937.0
34	Test Campaign	5.08.2019	2297	114295.0	95138.0
35	Test Campaign	6.08.2019	2458	42684.0	31489.0
36	Test Campaign	7.08.2019	2838	53986.0	42148.0
37	Test Campaign	8.08.2019	2916	33669.0	20149.0
38	Test Campaign	9.08.2019	2652	45511.0	31598.0
39	Test Campaign	10.08.2019	2790	95054.0	79632.0
40	Test Campaign	11.08.2019	2420	83633.0	71286.0
41	Test Campaign	12.08.2019	2831	124591.0	10598.0
42	Test Campaign	13.08.2019	1972	65827.0	49531.0
43	Test Campaign	14.08.2019	2537	56304.0	25982.0
44	Test Campaign	15.08.2019	2516	94338.0	76219.0
45	Test Campaign	16.08.2019	3076	106584.0	81389.0
46	Test Campaign	17.08.2019	1968	95843.0	54389.0
47	Test Campaign	18.08.2019	1979	53632.0	43241.0
48	Test Campaign	19.08.2019	2626	22521.0	10698.0
49	Test Campaign	20.08.2019	2712	39470.0	31893.0
50	Test Campaign	21.08.2019	3112	133771.0	109834.0
51	Test Campaign	22.08.2019	2899	34752.0	27932.0
52	Test Campaign	23.08.2019	2407	60286.0	49329.0
53	Test Campaign	24.08.2019	2078	36650.0	30489.0
54	Test Campaign	25.08.2019	2928	120576.0	105978.0
55	Test Campaign	26.08.2019	2311	80841.0	61589.0
56	Test Campaign	27.08.2019	2915	111469.0	92159.0
57	Test Campaign	28.08.2019	2247	54627.0	41267.0
58	Test Campaign	29.08.2019	2805	67444.0	43219.0
59	Test Campaign	30.08.2019	1977	120203.0	89380.0

	# of Website Clicks	# of Searches	# of View Content	# of Add to Cart	\
0	7016.0	2290.0	2159.0	1819.0	
1	8110.0	2033.0	1841.0	1219.0	
2	6508.0	1737.0	1549.0	1134.0	

3	3065.0	1042.0	982.0	1183.0
4	NaN	NaN	NaN	NaN
5	4028.0	1709.0	1249.0	784.0
6	2640.0	1388.0	1106.0	1166.0
7	7260.0	3047.0	2746.0	930.0
8	6198.0	2487.0	2179.0	645.0
9	2277.0	2475.0	1984.0	1629.0
10	8137.0	2941.0	2486.0	1887.0
11	2993.0	1397.0	1147.0	1439.0
12	6554.0	2390.0	1975.0	1794.0
13	4521.0	1209.0	1149.0	1339.0
14	4896.0	1179.0	1005.0	1641.0
15	5224.0	2427.0	2158.0	1613.0
16	6628.0	1756.0	1642.0	878.0
17	7253.0	2447.0	2115.0	1695.0
18	3706.0	2483.0	2098.0	908.0
19	2578.0	1001.0	848.0	1709.0
20	5691.0	2711.0	2496.0	1460.0
21	6843.0	3102.0	2988.0	819.0
22	4410.0	2896.0	2496.0	1913.0
23	4085.0	1274.0	1149.0	1146.0
24	4234.0	2423.0	2096.0	883.0
25	5435.0	2847.0	2421.0	1448.0
26	4941.0	3549.0	3249.0	980.0
27	6287.0	1672.0	1589.0	1711.0
28	8127.0	4891.0	4219.0	1486.0
29	4658.0	1615.0	1249.0	442.0
30	3038.0	1946.0	1069.0	894.0
31	4657.0	2359.0	1548.0	879.0
32	7885.0	2572.0	2367.0	1268.0
33	4216.0	2216.0	1437.0	566.0
34	5863.0	2106.0	858.0	956.0
35	7488.0	1854.0	1073.0	882.0
36	4221.0	2733.0	2182.0	1301.0
37	7184.0	2867.0	2194.0	1240.0
38	8259.0	2899.0	2761.0	1200.0
39	8125.0	2312.0	1804.0	424.0
40	3750.0	2893.0	2617.0	1075.0
41	8264.0	2081.0	1992.0	1382.0
42	7568.0	2213.0	2058.0	1391.0
43	3993.0	1979.0	1059.0	779.0
44	4993.0	2537.0	1609.0	1090.0
45	6800.0	2661.0	2594.0	1059.0
46	7910.0	1995.0	1576.0	383.0
47	6909.0	2824.0	2522.0	461.0
48	7617.0	2924.0	2801.0	788.0
49	6050.0	2061.0	1894.0	1047.0

50	5471.0	1995.0	1868.0	278.0
51	4431.0	1983.0	1131.0	367.0
52	5077.0	2592.0	2004.0	632.0
53	7156.0	2687.0	2427.0	327.0
54	3596.0	2937.0	2551.0	1228.0
55	3820.0	2037.0	1046.0	346.0
56	6435.0	2976.0	2552.0	992.0
57	8144.0	2432.0	1281.0	1009.0
58	7651.0	1920.0	1240.0	1168.0
59	4399.0	2978.0	1625.0	1034.0

	# of Purchase
0	618.0
1	511.0
2	372.0
3	340.0
4	NaN
5	764.0
6	499.0
7	462.0
8	501.0
9	734.0
10	475.0
11	794.0
12	766.0
13	788.0
14	366.0
15	438.0
16	222.0
17	243.0
18	542.0
19	299.0
20	800.0
21	387.0
22	766.0
23	585.0
24	386.0
25	251.0
26	605.0
27	643.0
28	334.0
29	670.0
30	255.0
31	677.0
32	578.0
33	340.0
34	768.0

```
35          488.0
36          890.0
37          431.0
38          845.0
39          275.0
40          668.0
41          709.0
42          812.0
43          340.0
44          398.0
45          487.0
46          238.0
47          257.0
48          512.0
49          730.0
50          245.0
51          276.0
52          473.0
53          269.0
54          651.0
55          284.0
56          771.0
57          721.0
58          677.0
59          572.0
```

Let's transform the date data into a standard format and create a variable to store the day of the week.

[1427]:
```
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)
df['Date'] = df['Date'].dt.strftime('%Y-%m-%d')
```

[1429]:
```
df['Date'] = pd.to_datetime(df['Date'])
df['Day of week'] = df['Date'].dt.day_name()
```

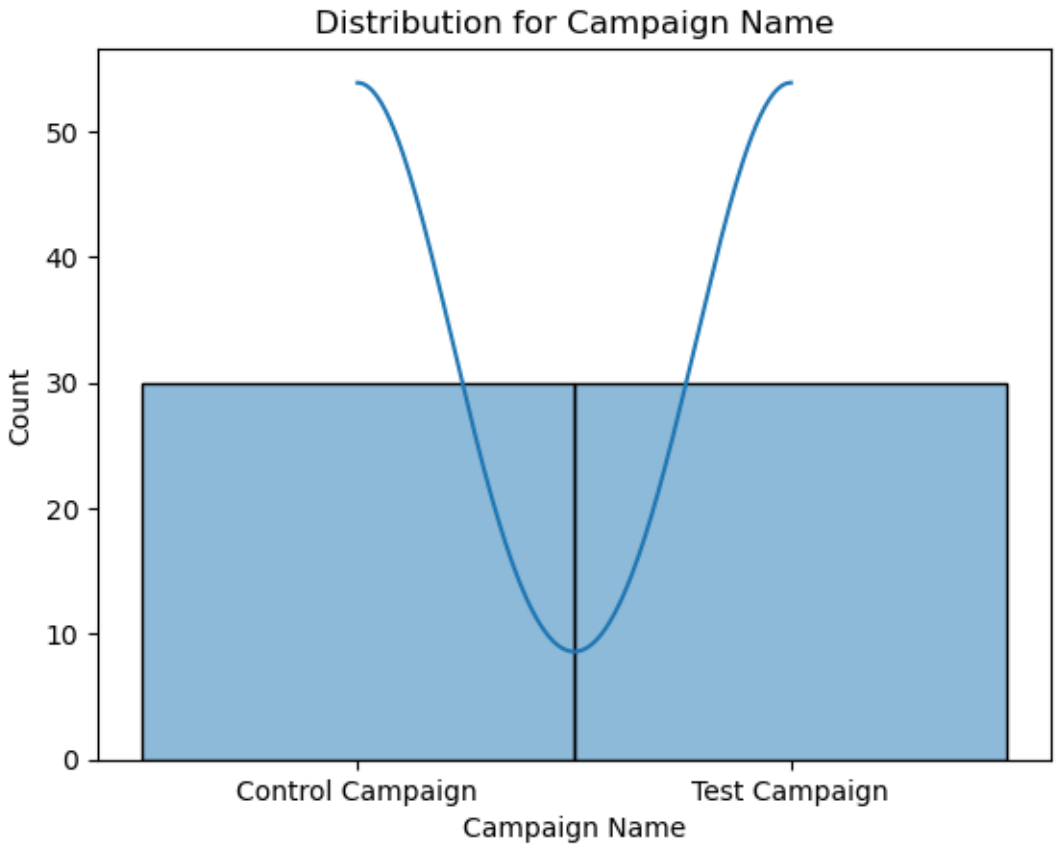
Checking the Data for Missing Values

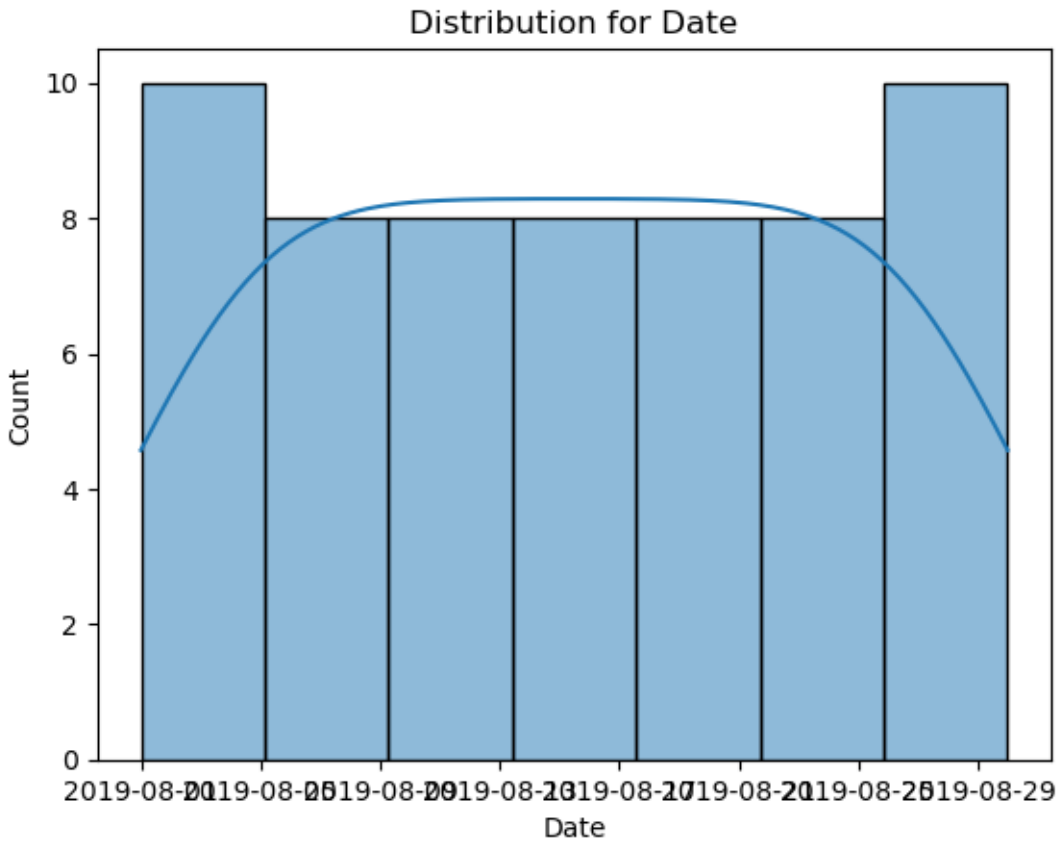
[1432]:
```
df.isnull().sum()
```

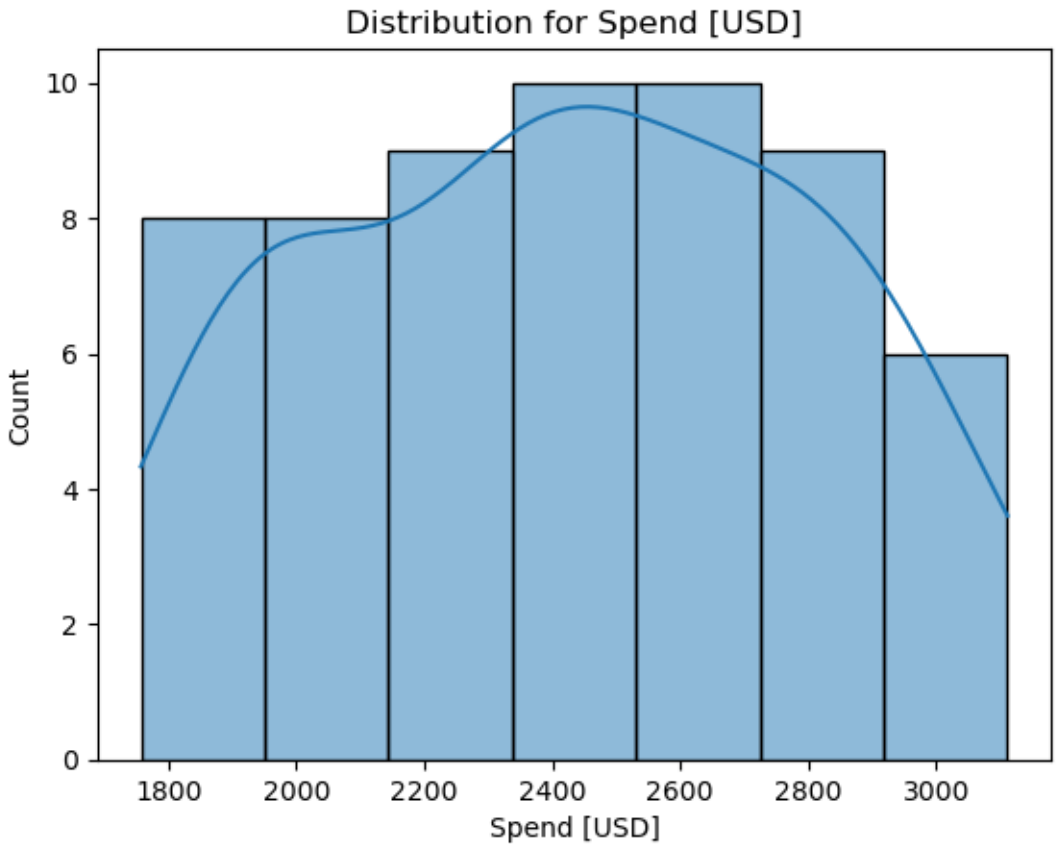
[1432]:
```
Campaign Name          0
Date                   0
Spend [USD]            0
# of Impressions       1
Reach                  1
# of Website Clicks    1
# of Searches          1
# of View Content      1
# of Add to Cart       1
```

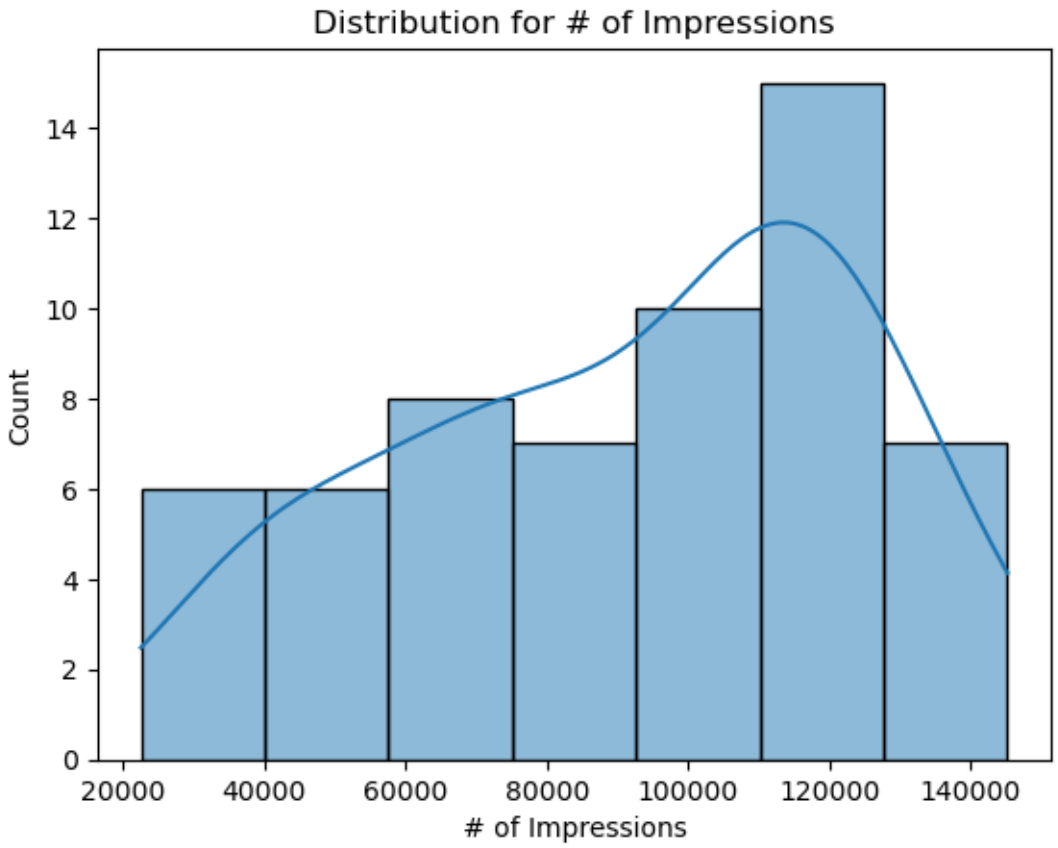
```
# of Purchase          1
Day of week            0
dtype: int64
```

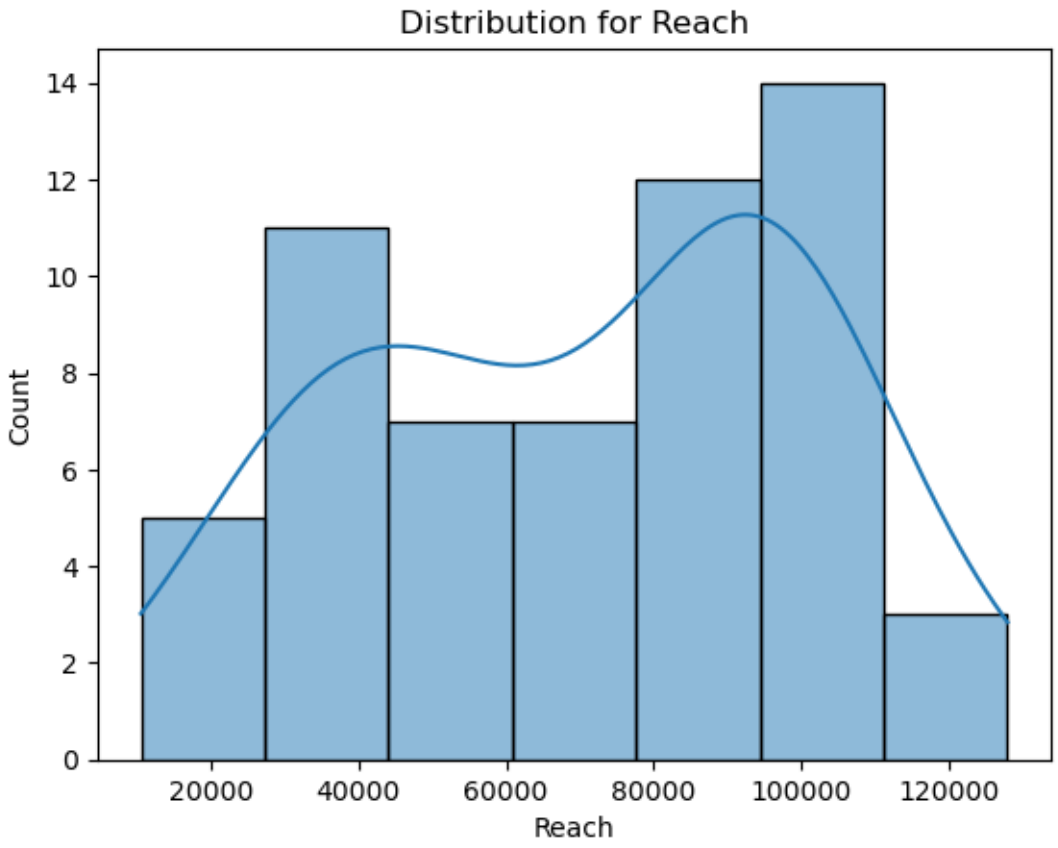
[1434]:
```
for column in df.columns:
    sns.histplot(df[column].dropna(), kde=True)
    plt.title(f'Distribution for {column}')
    plt.show()
```

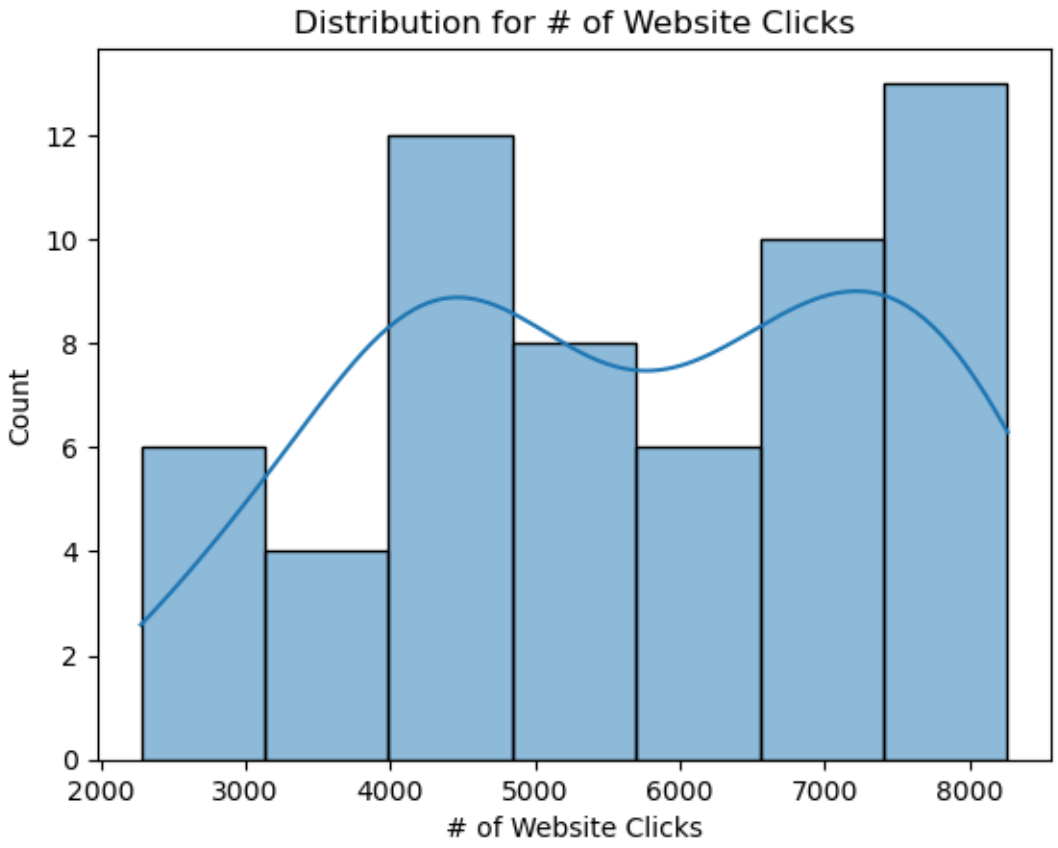


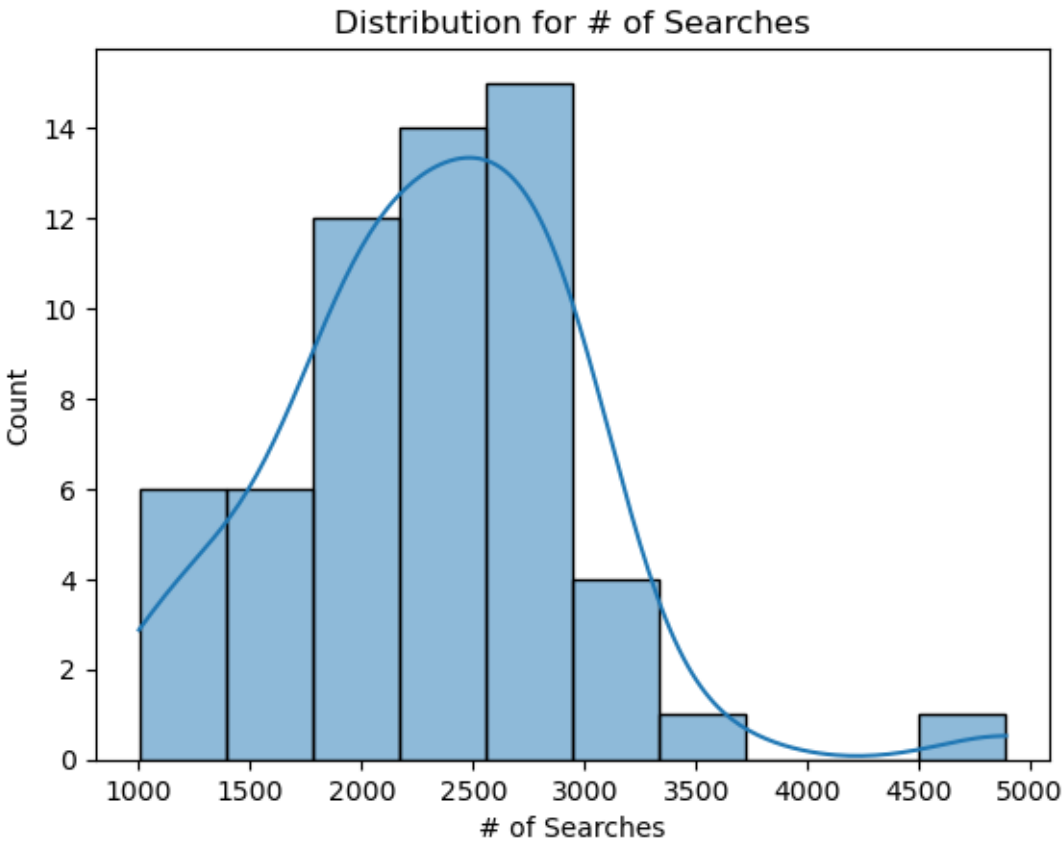

Distribution for Date
Count
10
8
6
4
2
0
2019-08-20
2019-08-25
2019-08-29
Date

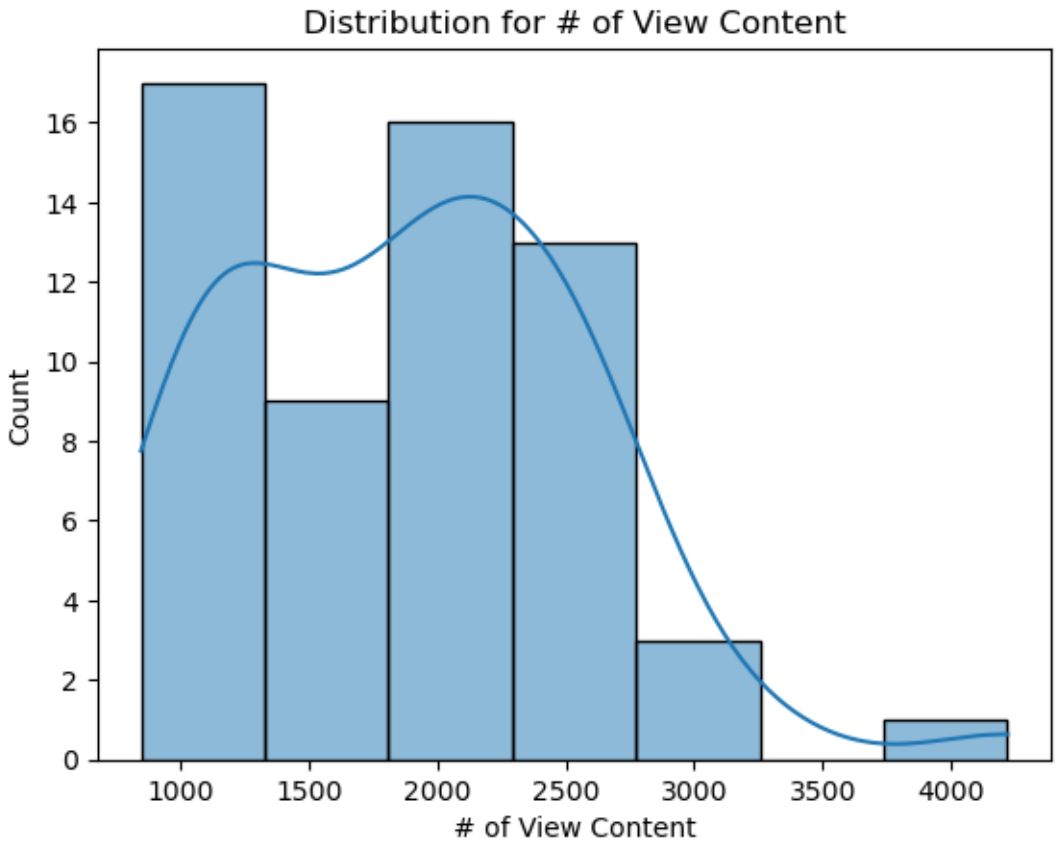

Distribution for Spend [USD]
Count
Spend [USD]
0
2
4
6
8
10
1800
2000
2200
2400
2600
2800
3000

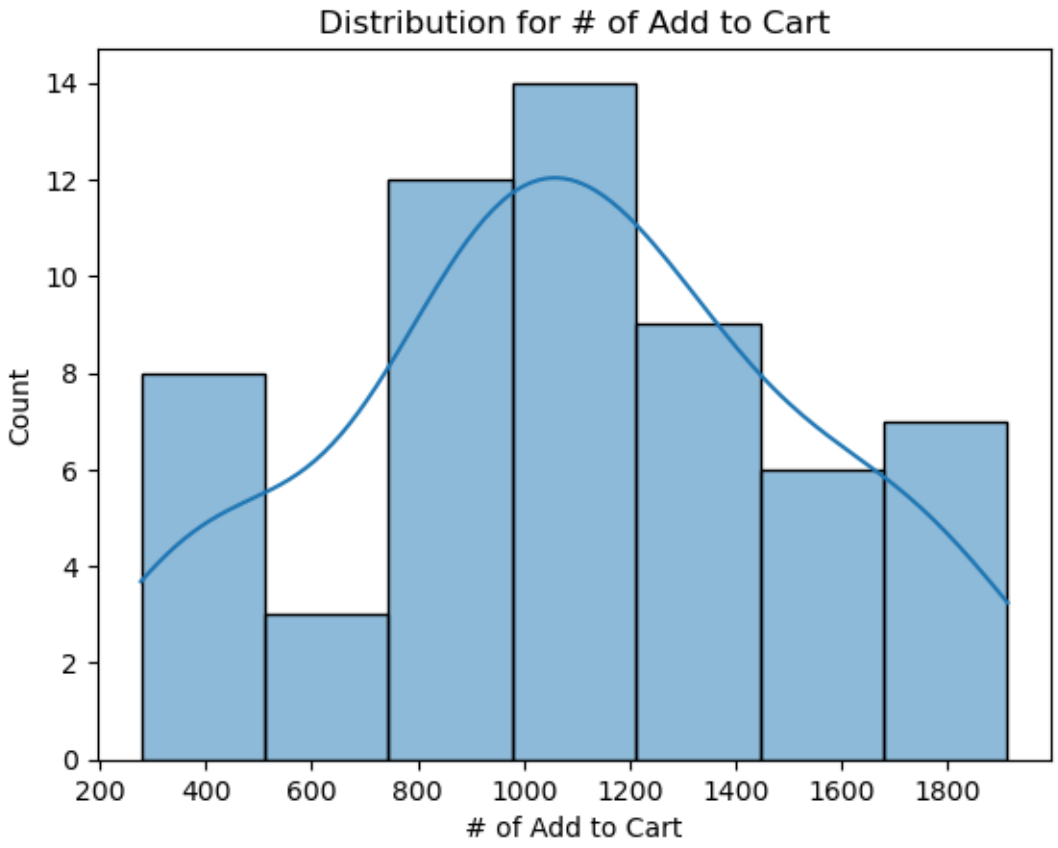

Distribution for # of Impressions
Count
# of Impressions
0
2
4
6
8
10
12
14
20000
40000
60000
80000
100000
120000
140000

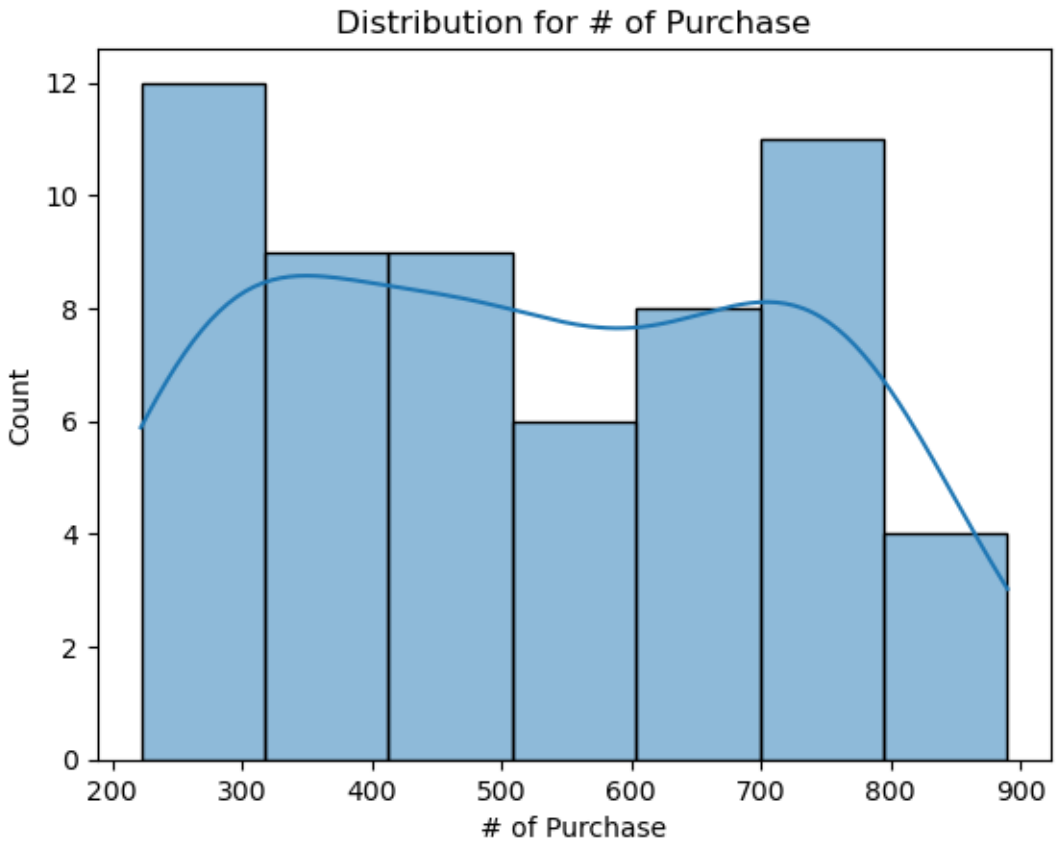

Distribution for Reach
Count
Reach
0
2
4
6
8
10
12
14
20000
40000
60000
80000
100000
120000

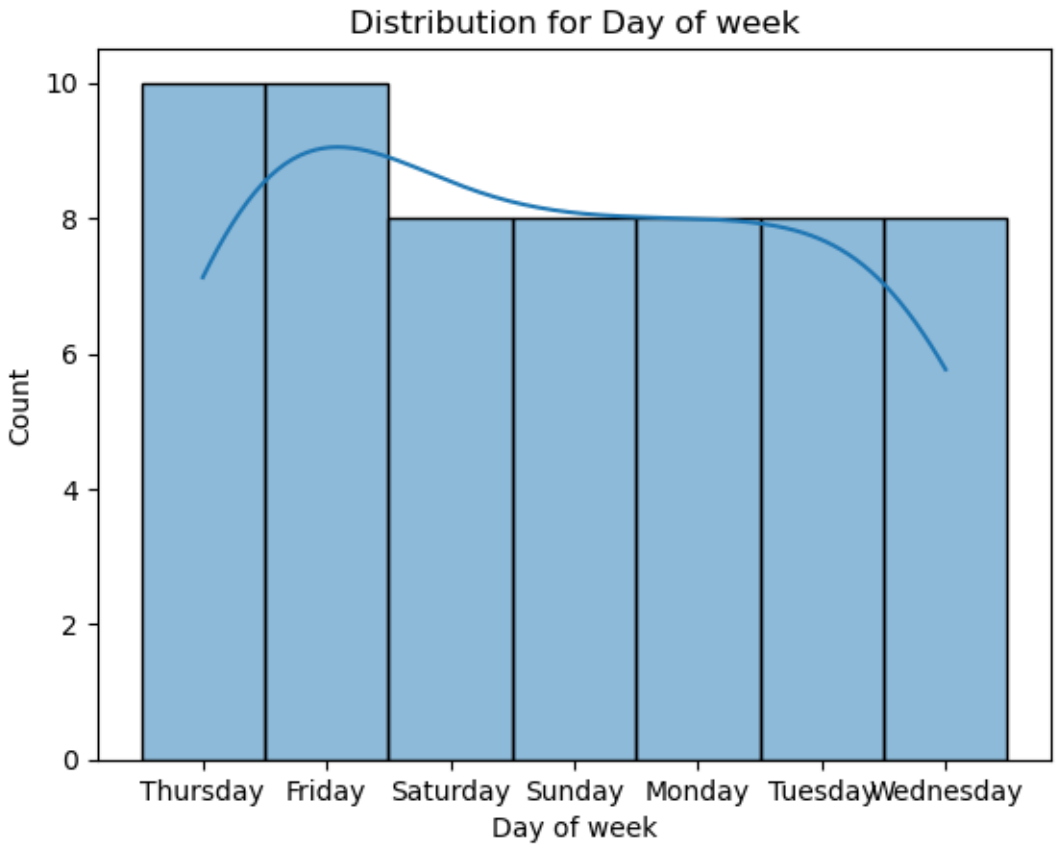

Distribution for # of Website Clicks
Count
0
2
4
6
8
10
12
2000
3000
4000
5000
6000
7000
8000
# of Website Clicks


Distribution for # of Searches
Count
# of Searches
0
2
4
6
8
10
12
14
1000
1500
2000
2500
3000
3500
4000
4500
5000

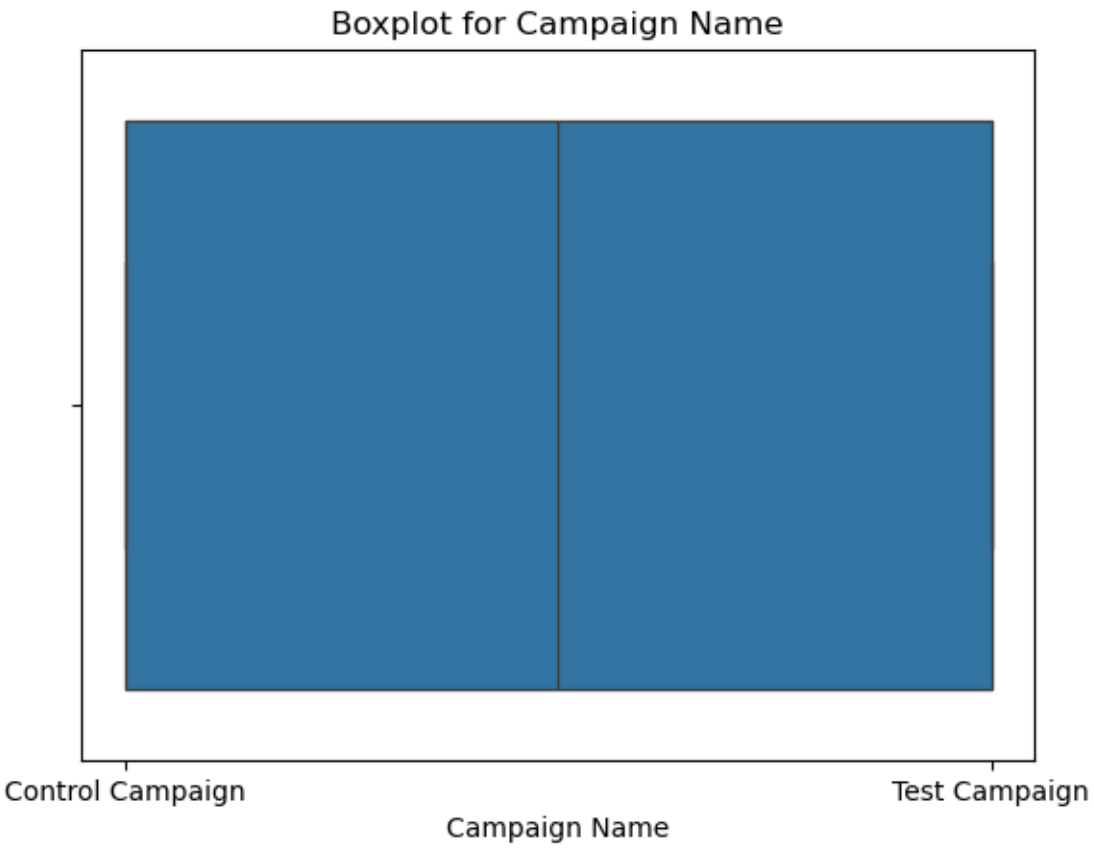

Distribution for # of View Content
Count
0
2
4
6
8
10
12
14
16
1000
1500
2000
2500
3000
3500
4000
# of View Content

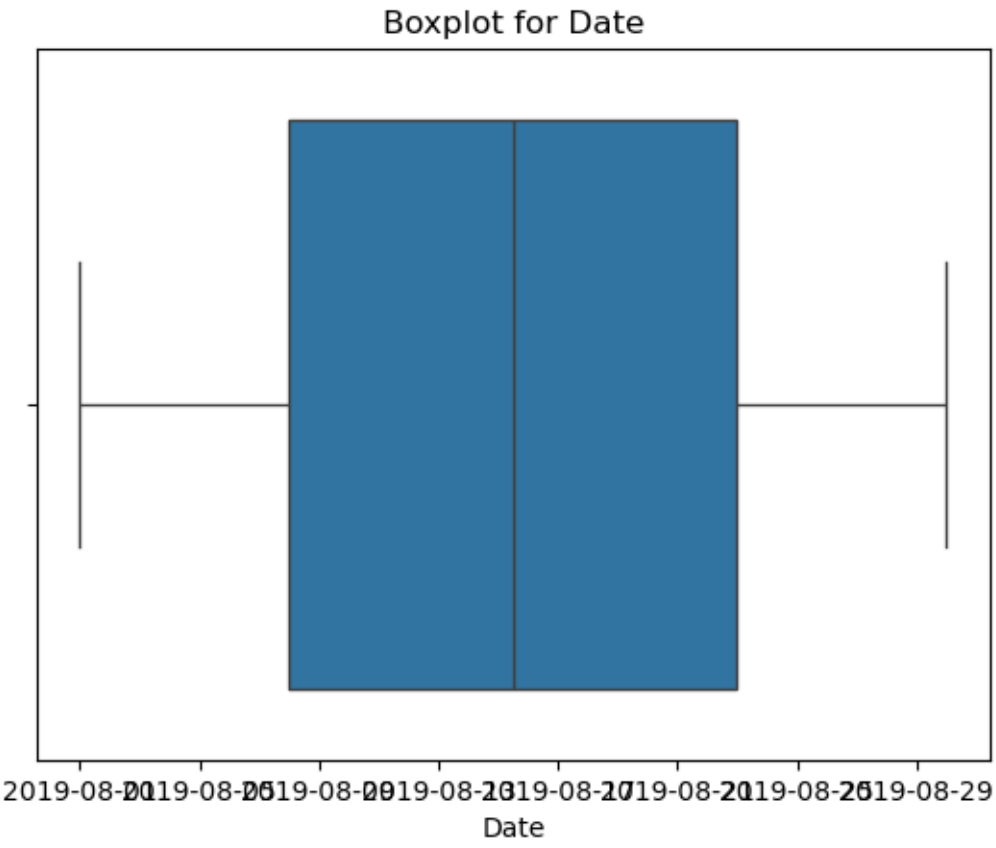

Distribution for # of Add to Cart
Count
# of Add to Cart
0
2
4
6
8
10
12
14
200
400
600
800
1000
1200
1400
1600
1800

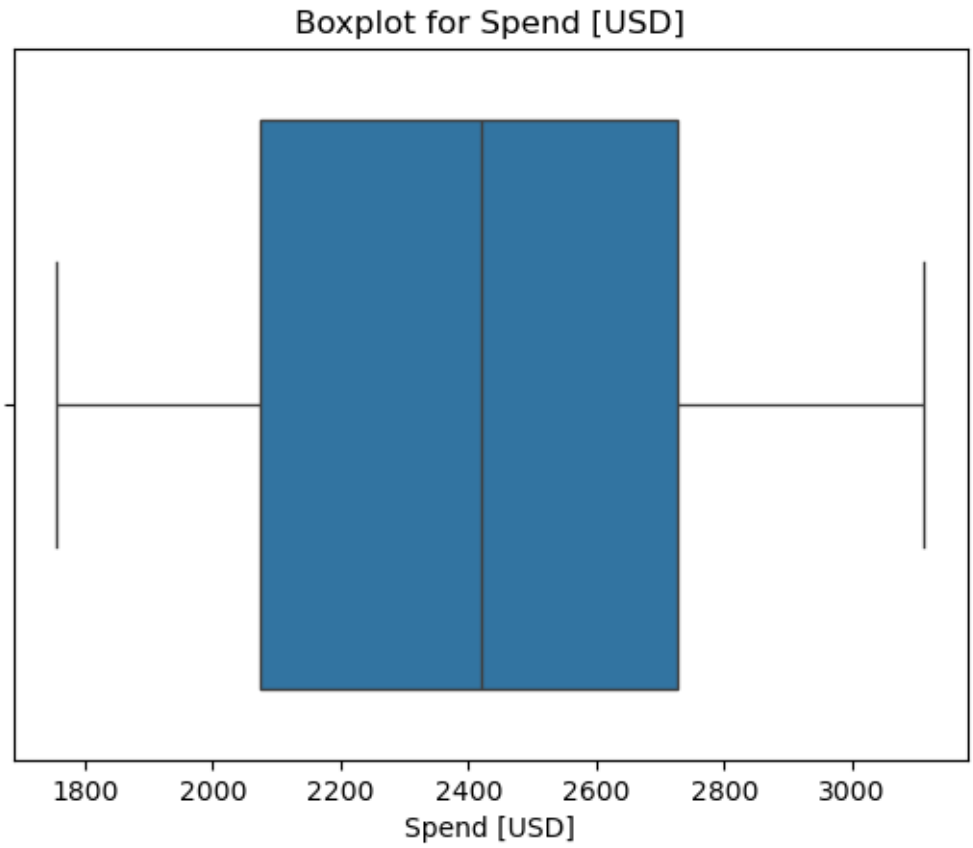

Distribution for # of Purchase
Count
# of Purchase
0
2
4
6
8
10
12
200
300
400
500
600
700
800
900

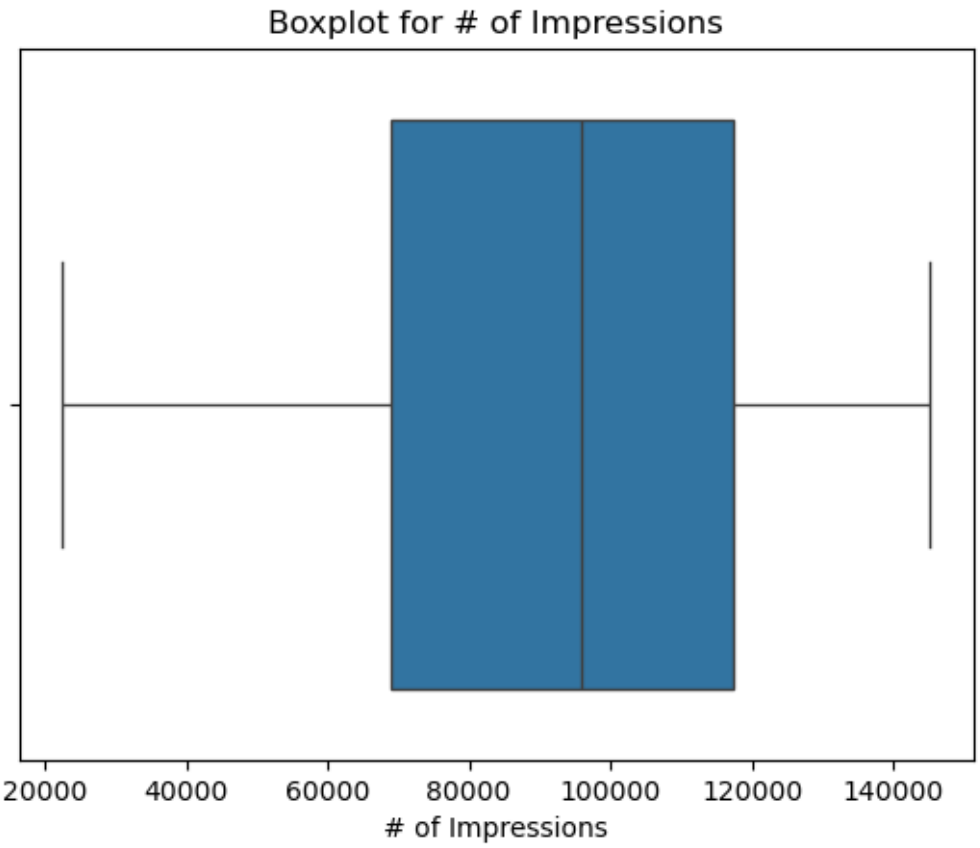


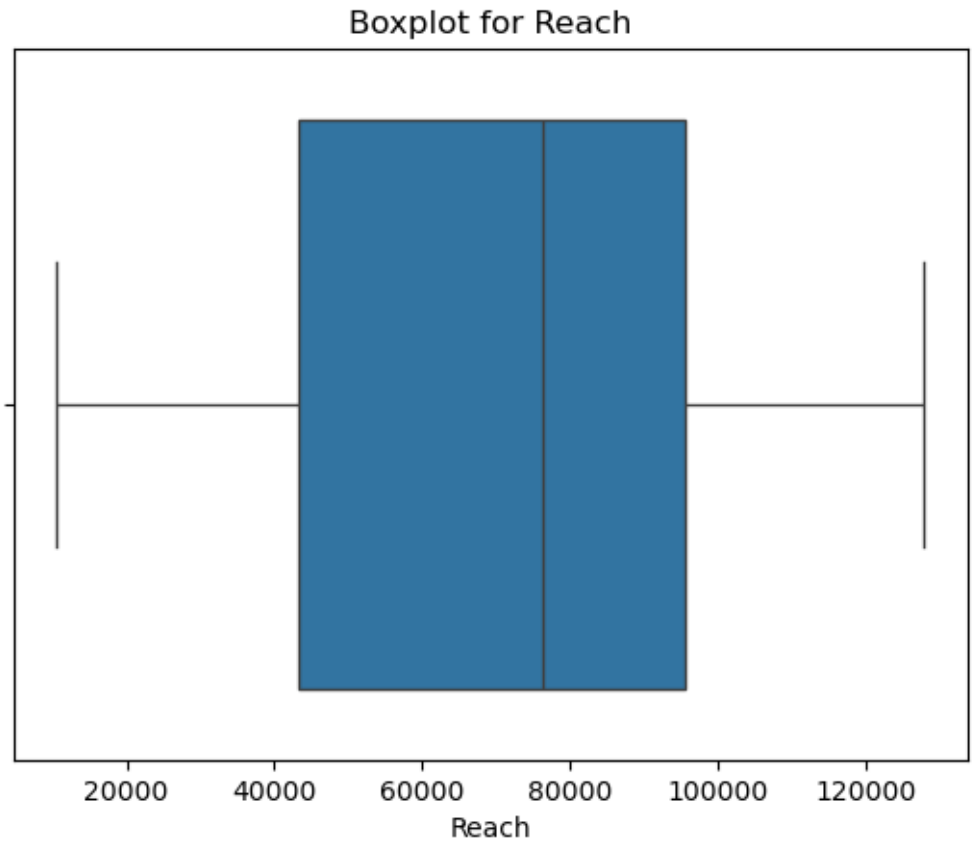
[1435]:
```
columns = df.columns
for column in df.columns:
    sns.boxplot(data=df, x=column)
    plt.title(f'Boxplot for {column}')
    plt.show()
```

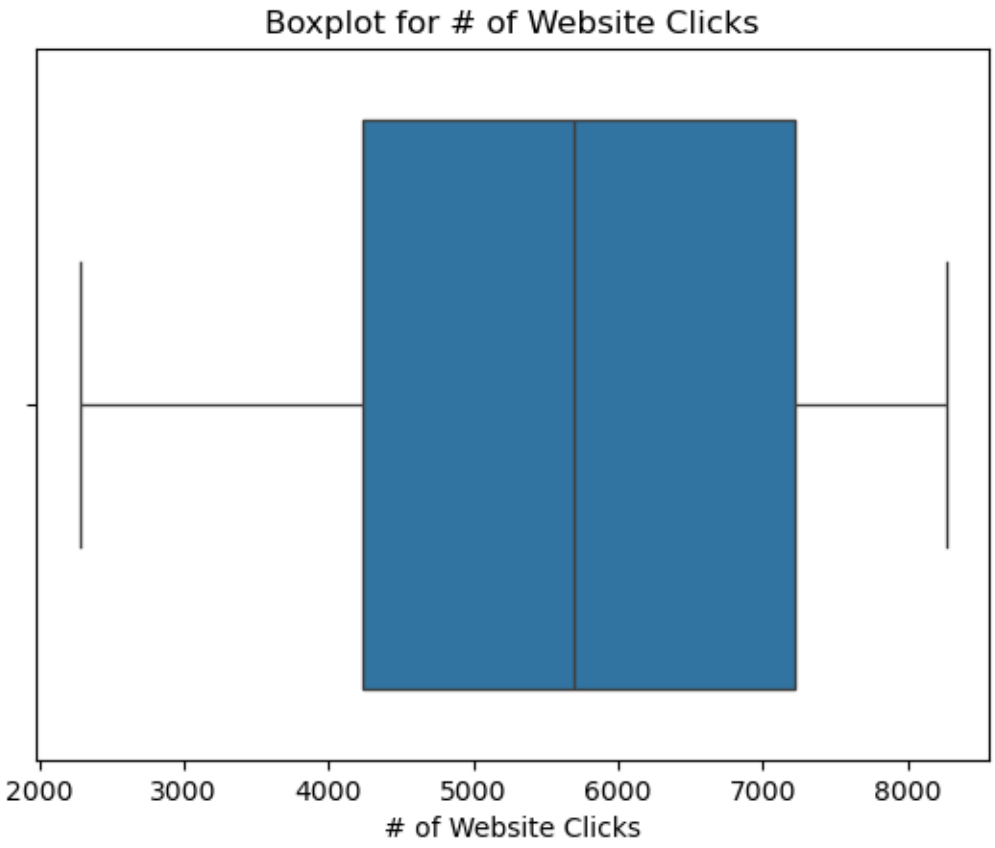

Boxplot for Campaign Name
Control Campaign
Test Campaign
Campaign Name

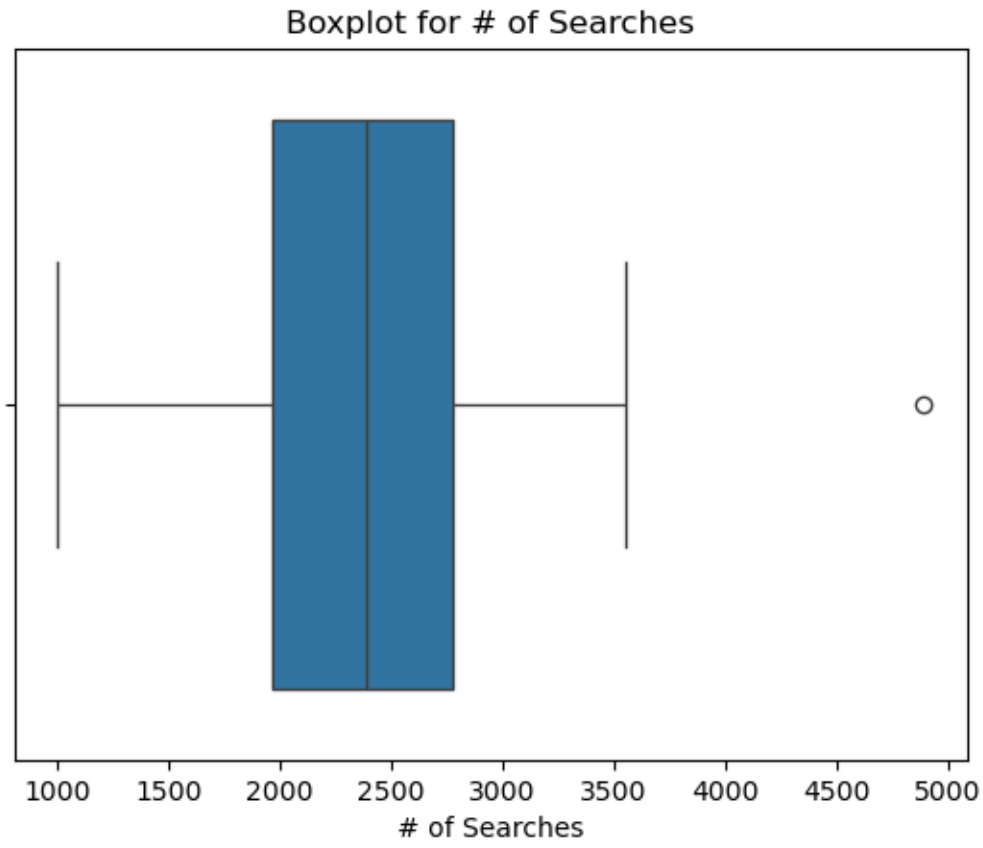

Boxplot for Date
2019-08-01
2019-08-05
2019-08-09
2019-08-13
2019-08-17
2019-08-21
2019-08-25
2019-08-29
Date

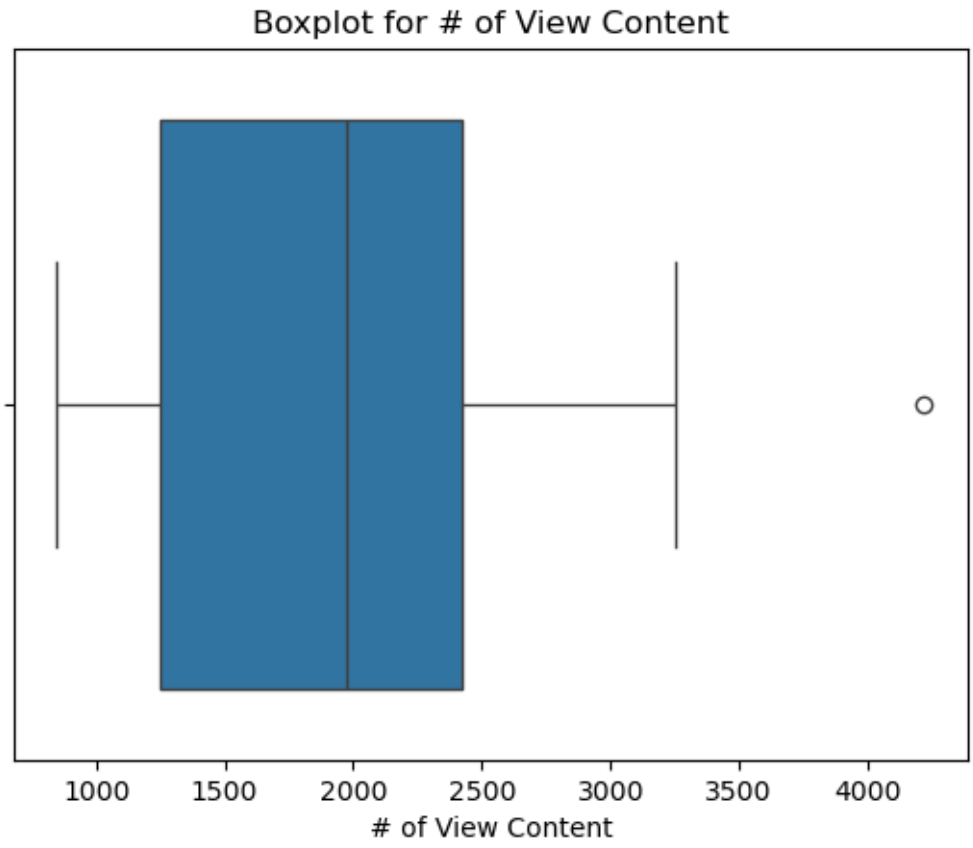

Boxplot for Spend [USD]
1800
2000
2200
2400
2600
2800
3000
Spend [USD]

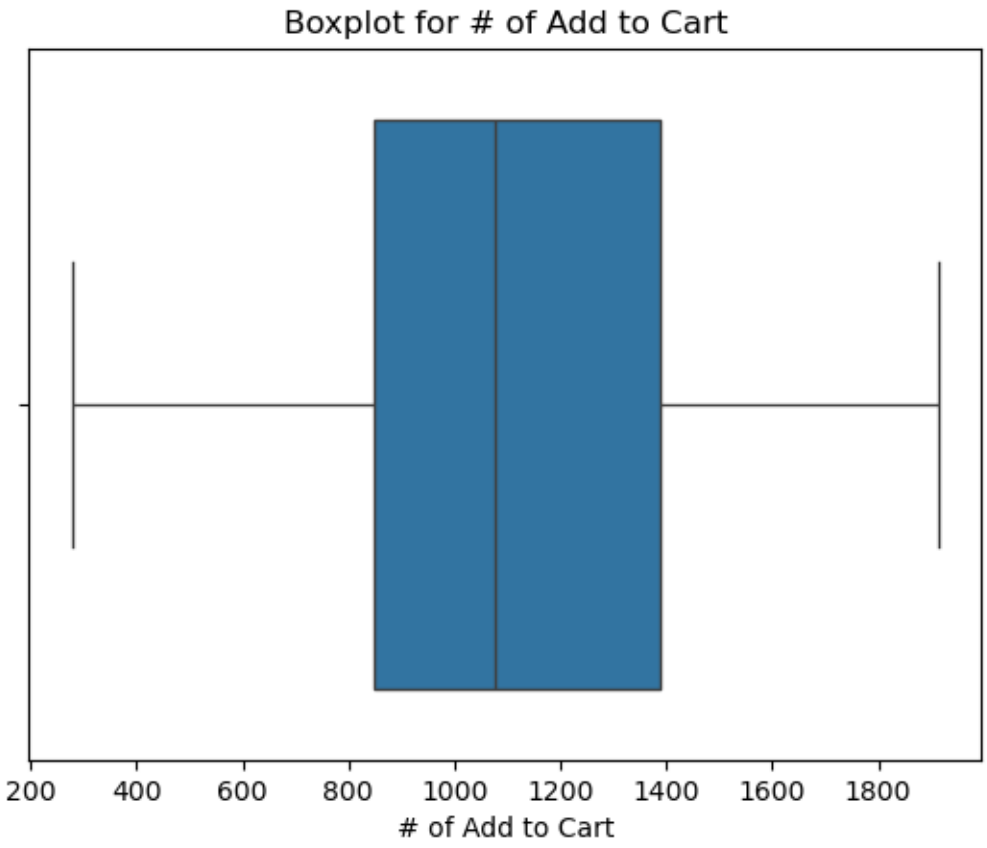

Boxplot for # of Impressions
20000
40000
60000
80000
100000
120000
140000
# of Impressions

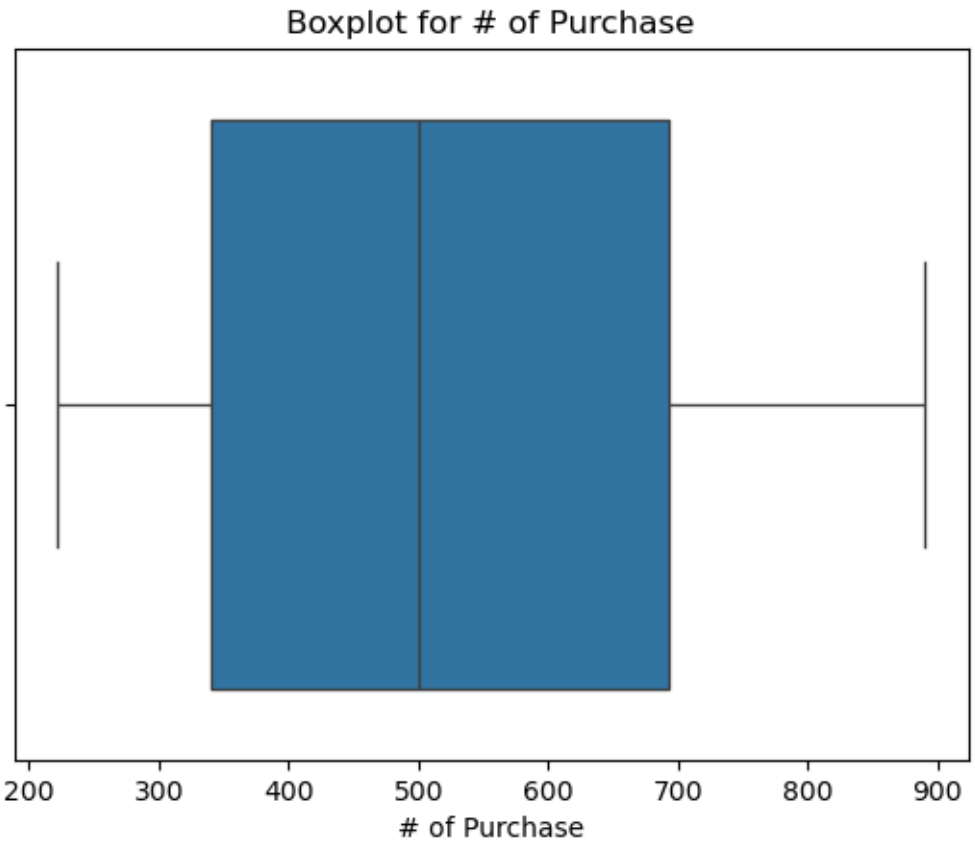

Boxplot for Reach
20000
40000
60000
80000
100000
120000
Reach

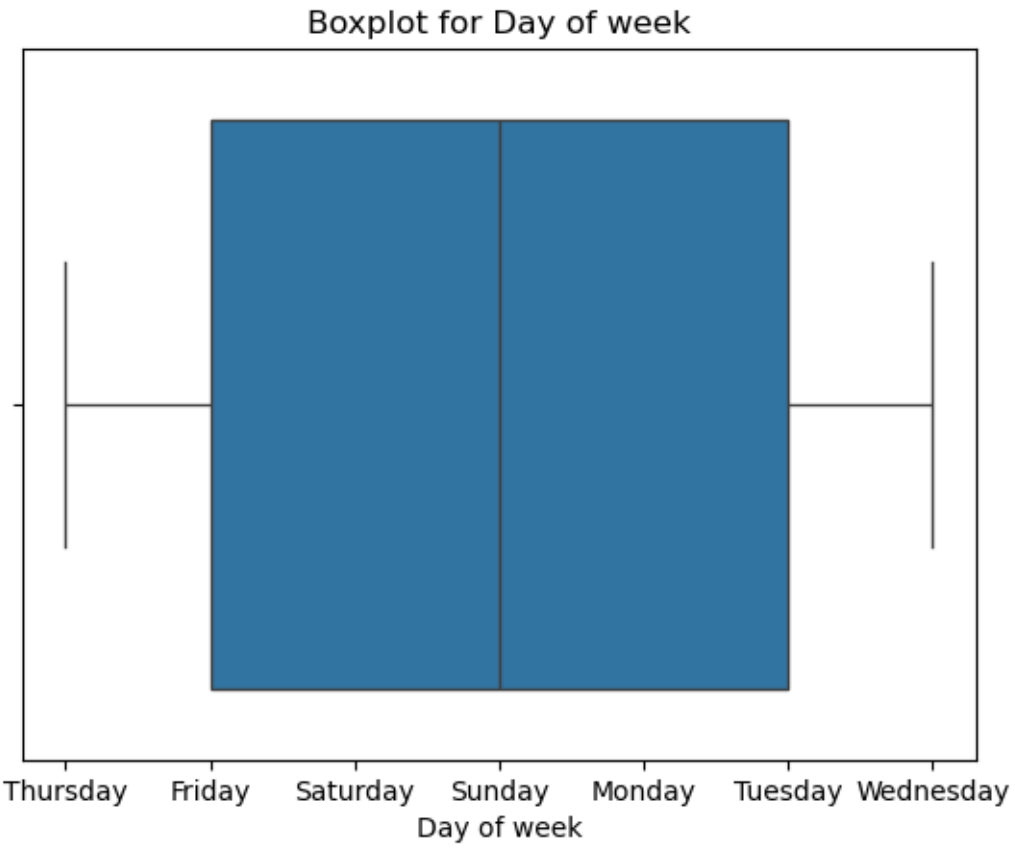

Boxplot for # of Website Clicks
2000
3000
4000
5000
6000
7000
8000
# of Website Clicks


Boxplot for # of Searches
1000
1500
2000
2500
3000
3500
4000
4500
5000
# of Searches


Boxplot for # of View Content
1000
1500
2000
2500
3000
3500
4000
# of View Content


Boxplot for # of Add to Cart
200
400
600
800
1000
1200
1400
1600
1800
# of Add to Cart


Boxplot for # of Purchase
200
300
400
500
600
700
800
900
# of Purchase



[1436]:
```
df.interpolate(method='linear', inplace=True)
```

```
/var/folders/g_/dlksrxdd3pz88bqsmz91cx540000gn/T/ipykernel_86333/2868764835.py:1
: FutureWarning: DataFrame.interpolate with object dtype is deprecated and will
raise in a future version. Call obj.infer_objects(copy=False) before
interpolating instead.
  df.interpolate(method='linear', inplace=True)
```

[1437]:
```
df.isnull().sum()
```

[1437]:
```
Campaign Name          0
Date                   0
Spend [USD]            0
# of Impressions       0
Reach                  0
# of Website Clicks    0
# of Searches          0
# of View Content      0
# of Add to Cart       0
# of Purchase          0
```

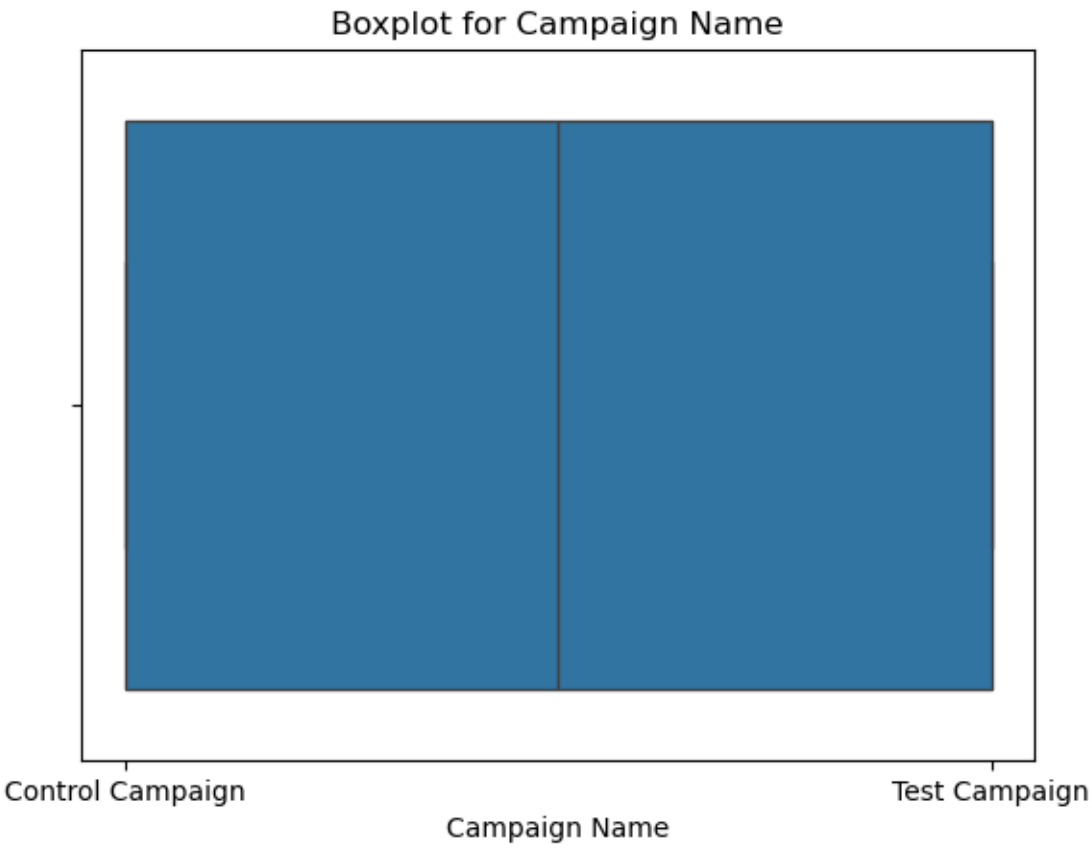
```
Day of week             0
dtype: int64
```

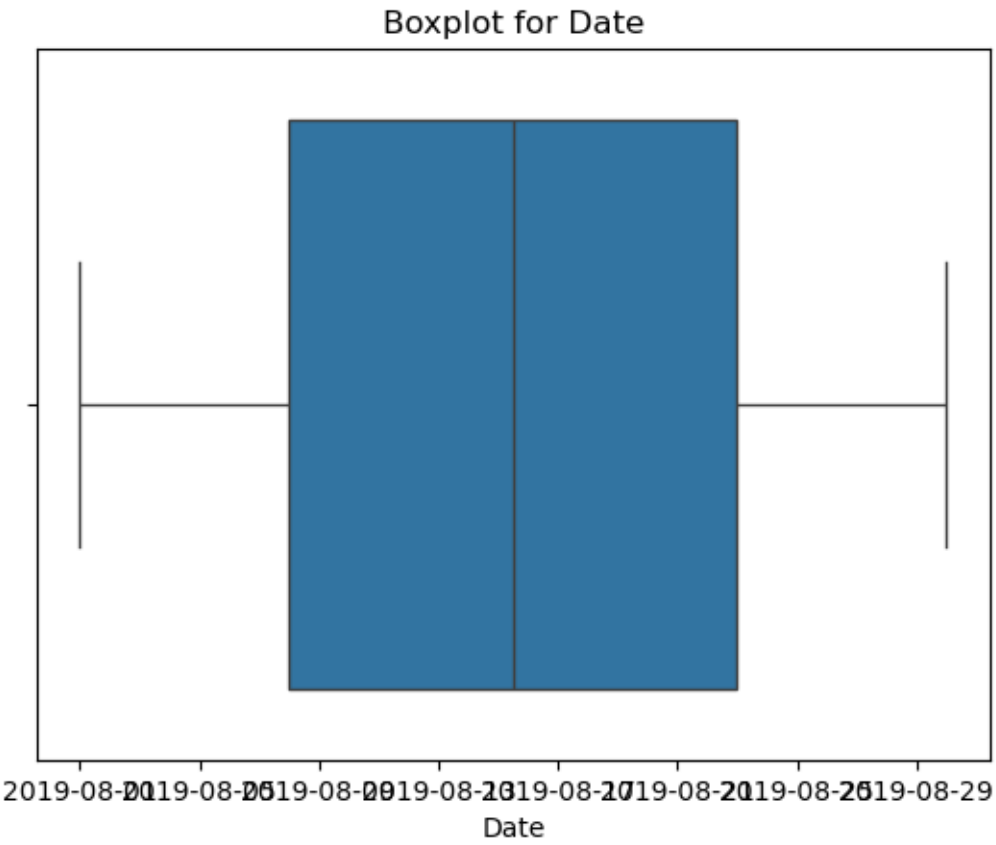
Anomalously high numbers of users viewing products on the website, searching for products, clicking on the website, and adding products to the cart, combined with low conversion rates, no extra advertising funds, and a low number of unique ad impressions, could be caused by several factors: Possible reasons for high values: Natural surge of interest: The product may have gained unexpected attention (e.g., through a viral social media post, mention in the media, or customer reviews). This increase in exploration isn't tied to advertising budgets but results in higher user engagement with the product. Returning user behavior: Loyal customers or users familiar with the brand may have suddenly become active (e.g., as a delayed response to previous ad campaigns). Technical error or automated activity: Bots or automated systems could generate fake traffic, especially if the platform is vulnerable to such activity. Competitive or research behavior from users: Users may browse the product to compare it with similar items but decide not to make a purchase. Incorrect ad targeting: If the ads were shown to an irrelevant audience, they might lead to increased exploration of the site but fail to generate conversions.

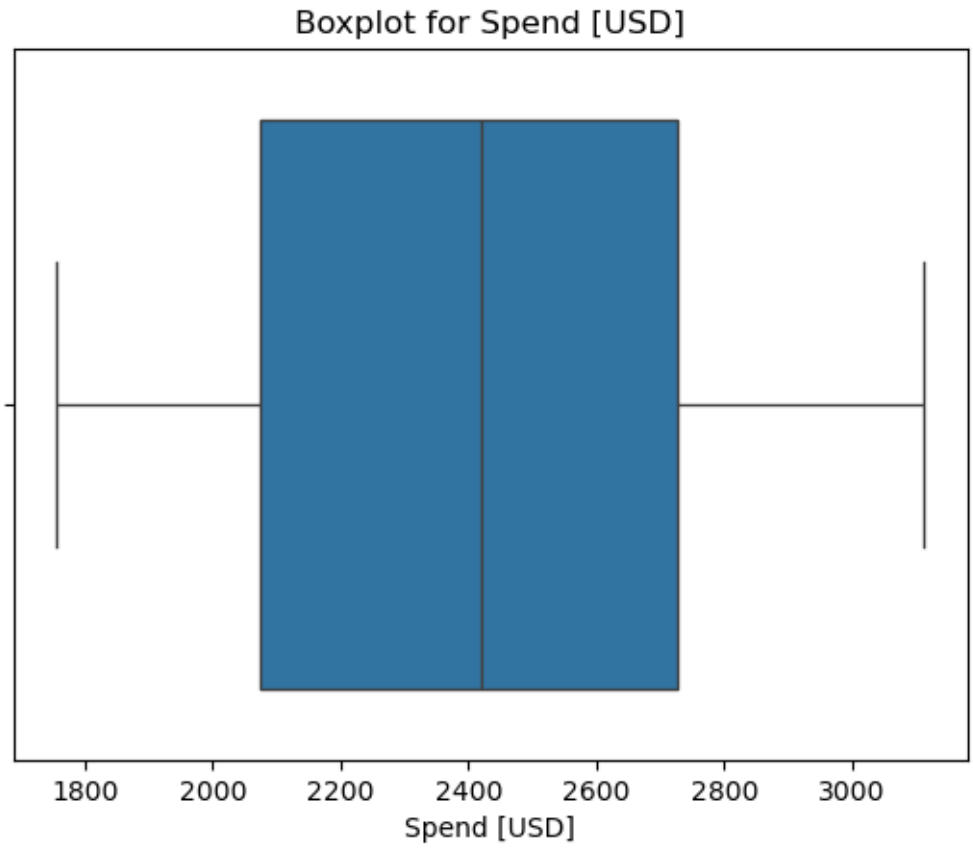
To preserve the data but limit the impact of outliers, we will replace them with the upper allowable limit.

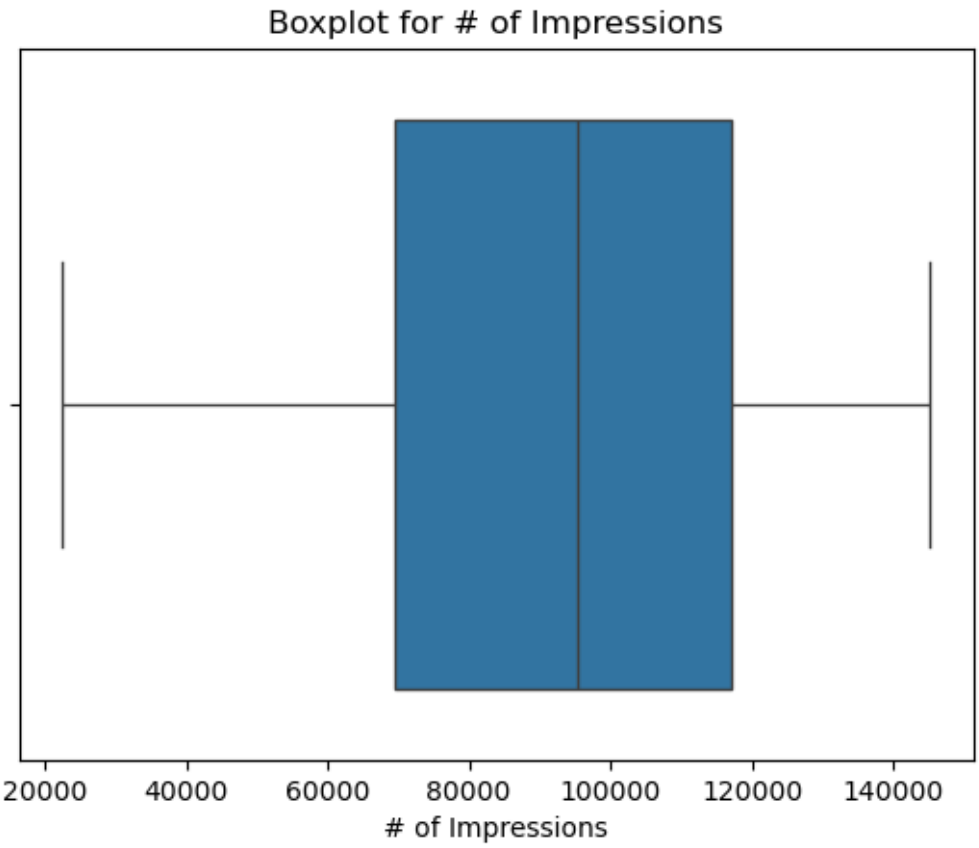
[1440]:
```
searches_top_border = df['# of Searches'].quantile(0.95)
df['# of Searches'] = df['# of Searches'].apply(lambda x: x if x <=␣
 ↪searches_top_border else searches_top_border)

view_top_border = df['# of View Content'].quantile(0.95)
df['# of View Content'] = df['# of View Content'].apply(lambda x: x if x <=␣
 ↪view_top_border else view_top_border)
```

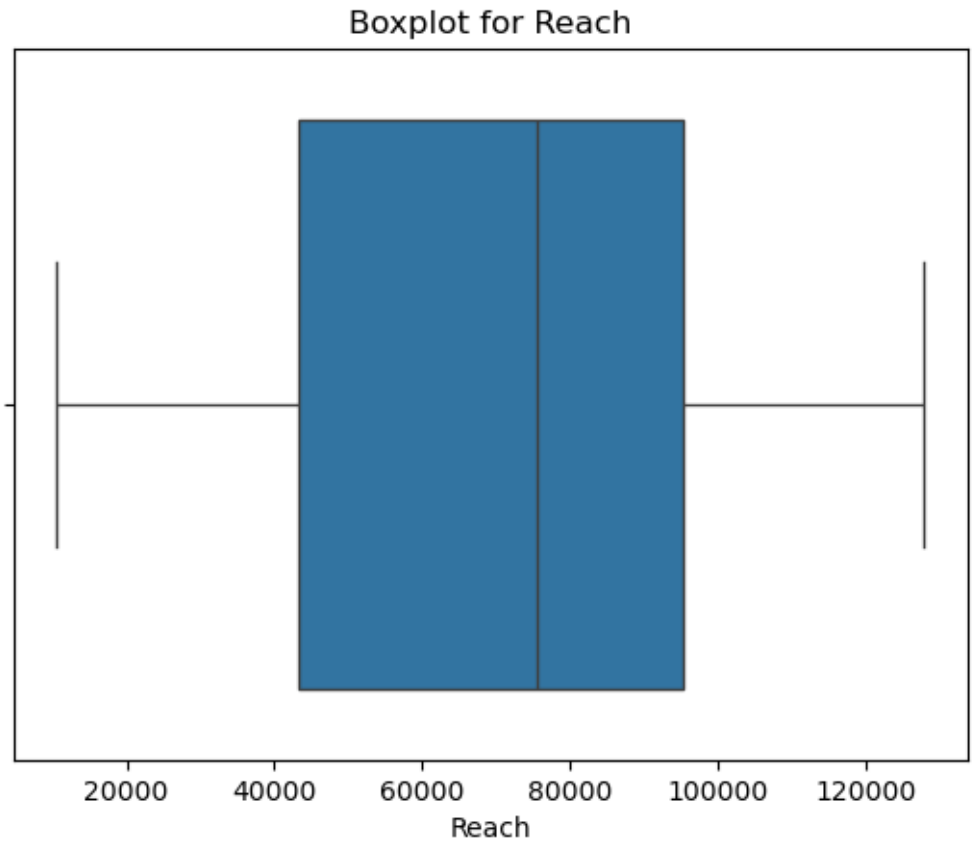
[1441]:
```
columns = df.columns
for column in df.columns:
    sns.boxplot(data=df, x=column)
    plt.title(f'Boxplot for {column}')
    plt.show()
```

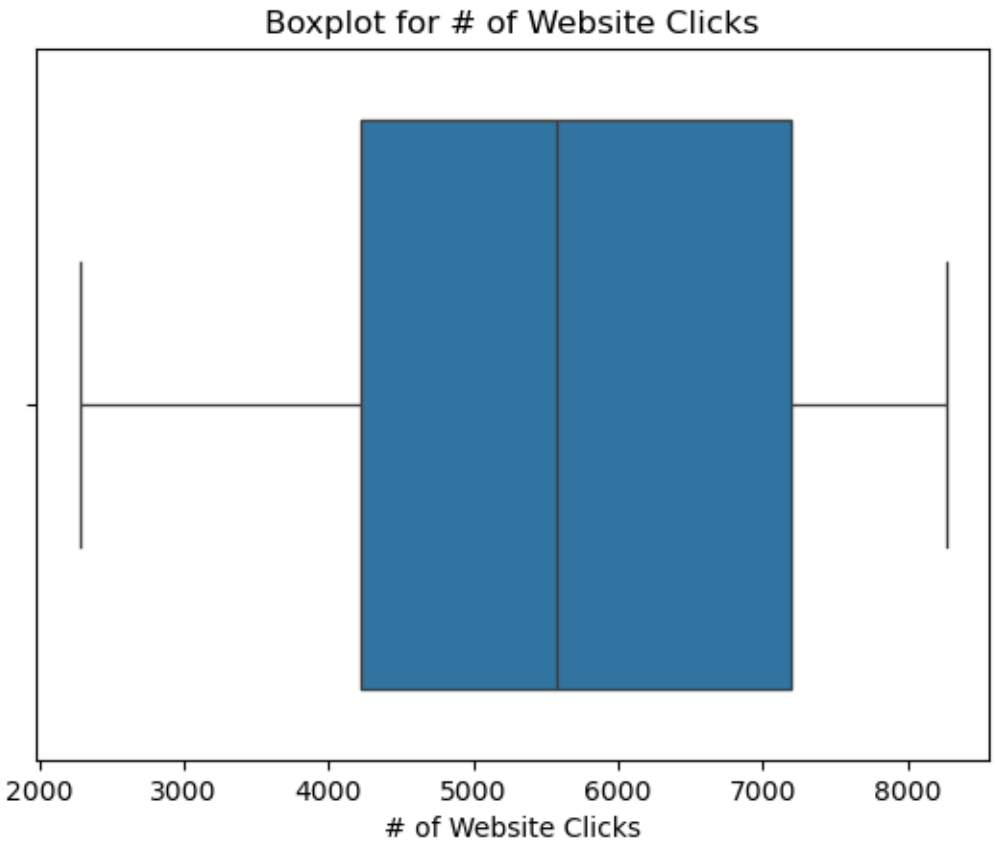

Boxplot for Campaign Name
Control Campaign
Test Campaign
Campaign Name

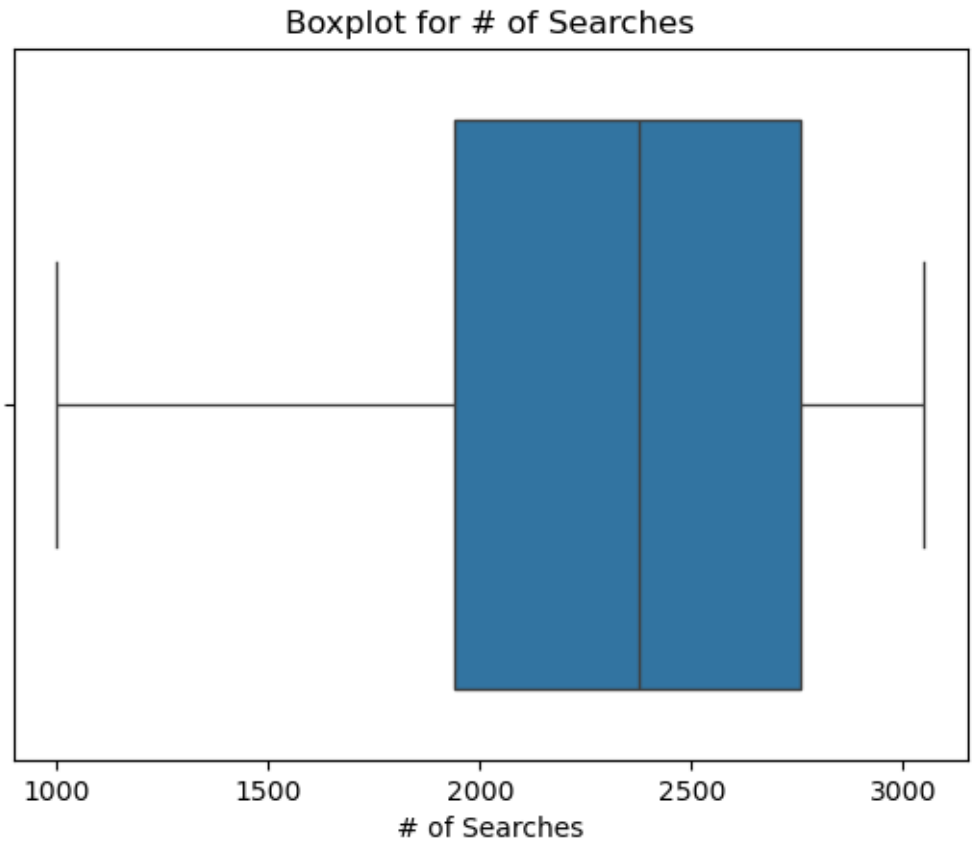

Boxplot for Date
2019-08-01
2019-08-05
2019-08-09
2019-08-13
2019-08-17
2019-08-21
2019-08-25
2019-08-29
Date

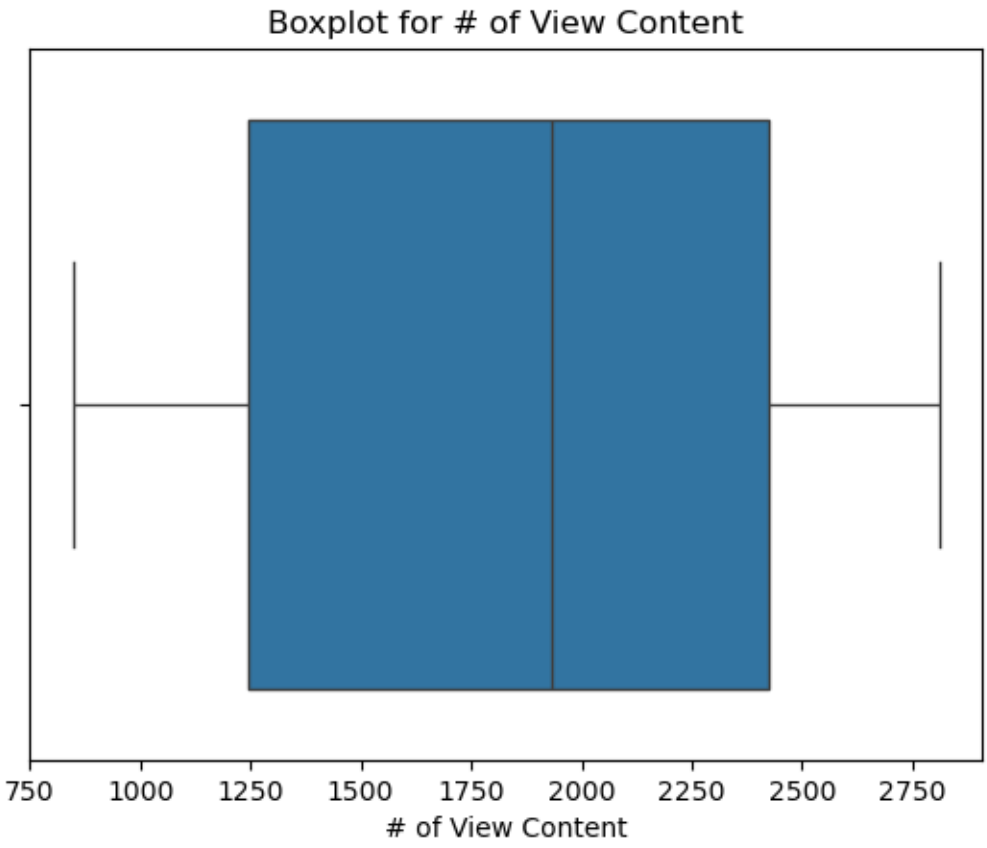

Boxplot for Spend [USD]
1800
2000
2200
2400
2600
2800
3000
Spend [USD]

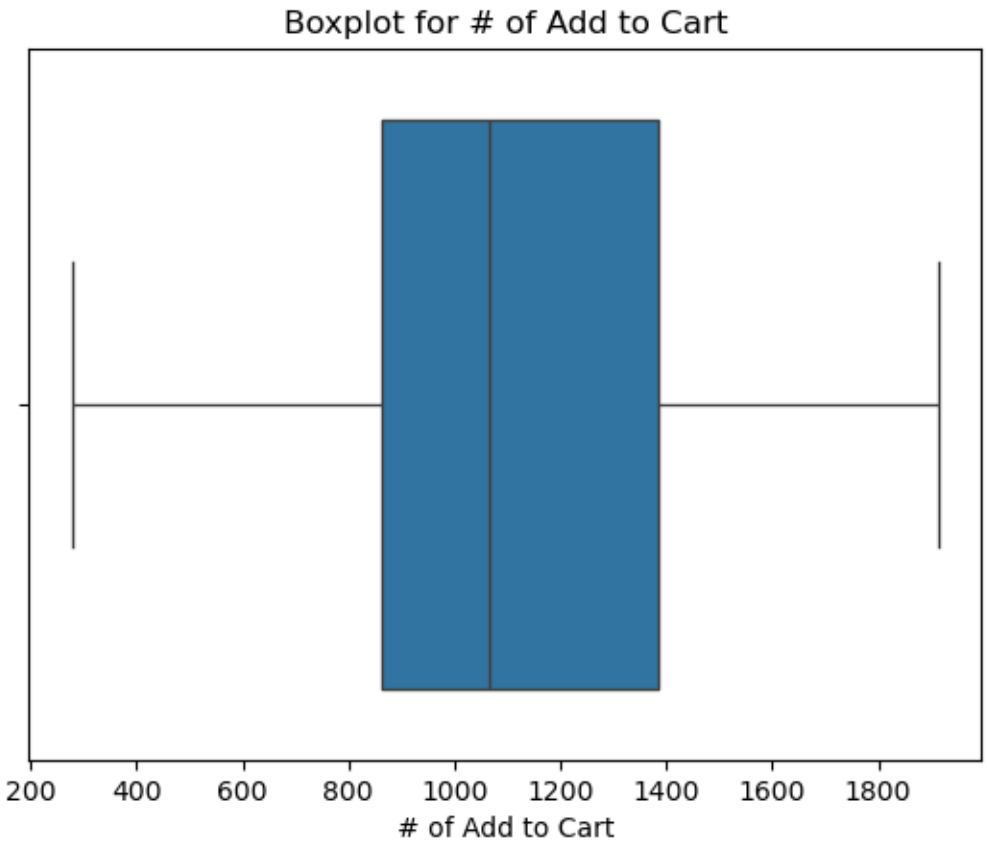

Boxplot for # of Impressions
20000
40000
60000
80000
100000
120000
140000
# of Impressions

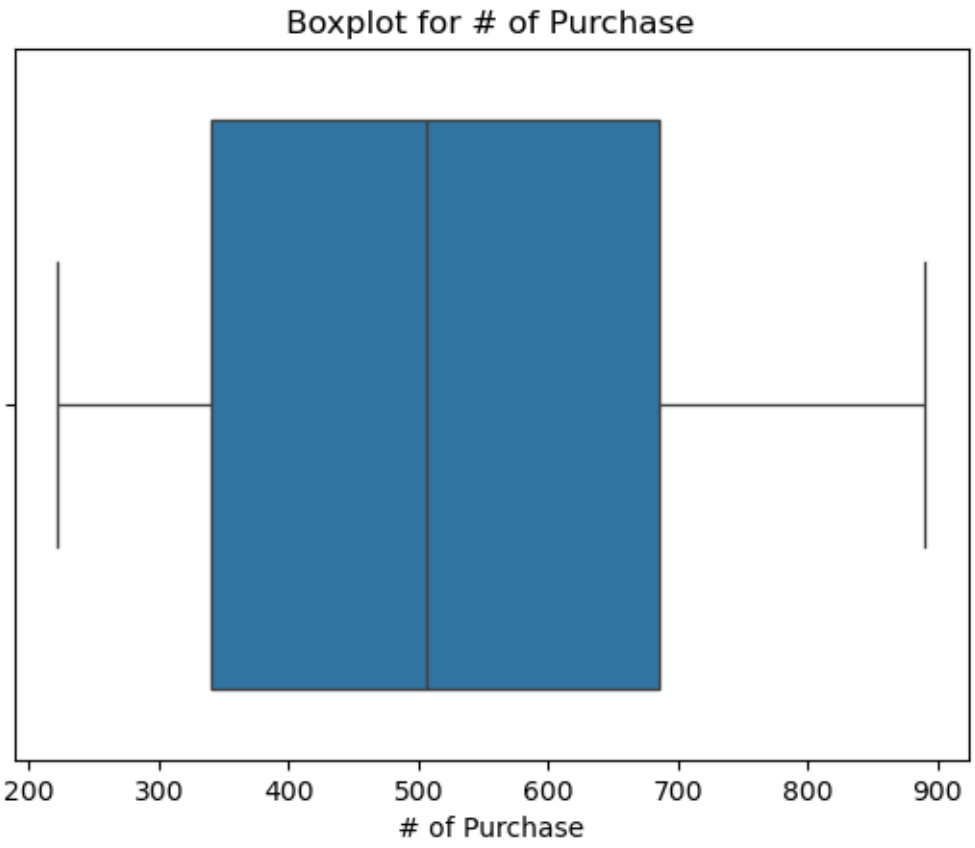

Boxplot for Reach
20000
40000
60000
80000
100000
120000
Reach

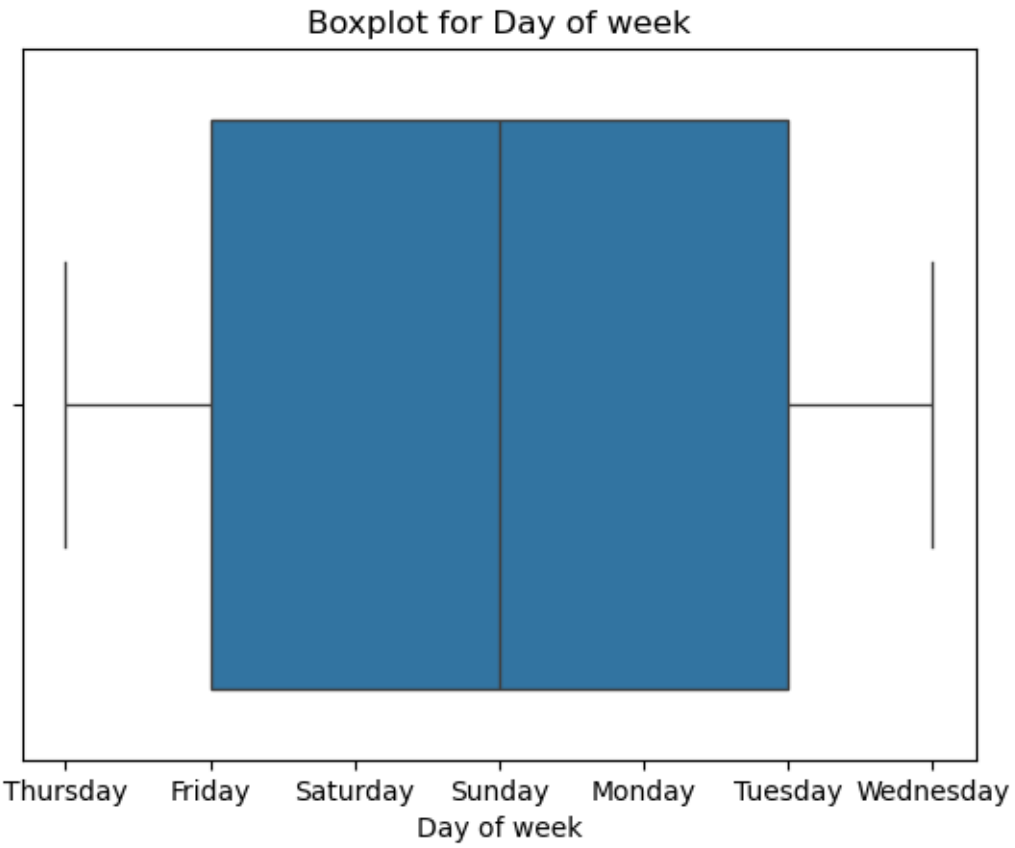

Boxplot for # of Website Clicks
2000
3000
4000
5000
6000
7000
8000
# of Website Clicks


Boxplot for # of Searches
1000
1500
2000
2500
3000
# of Searches


Boxplot for # of View Content
750
1000
1250
1500
1750
2000
2250
2500
2750
# of View Content

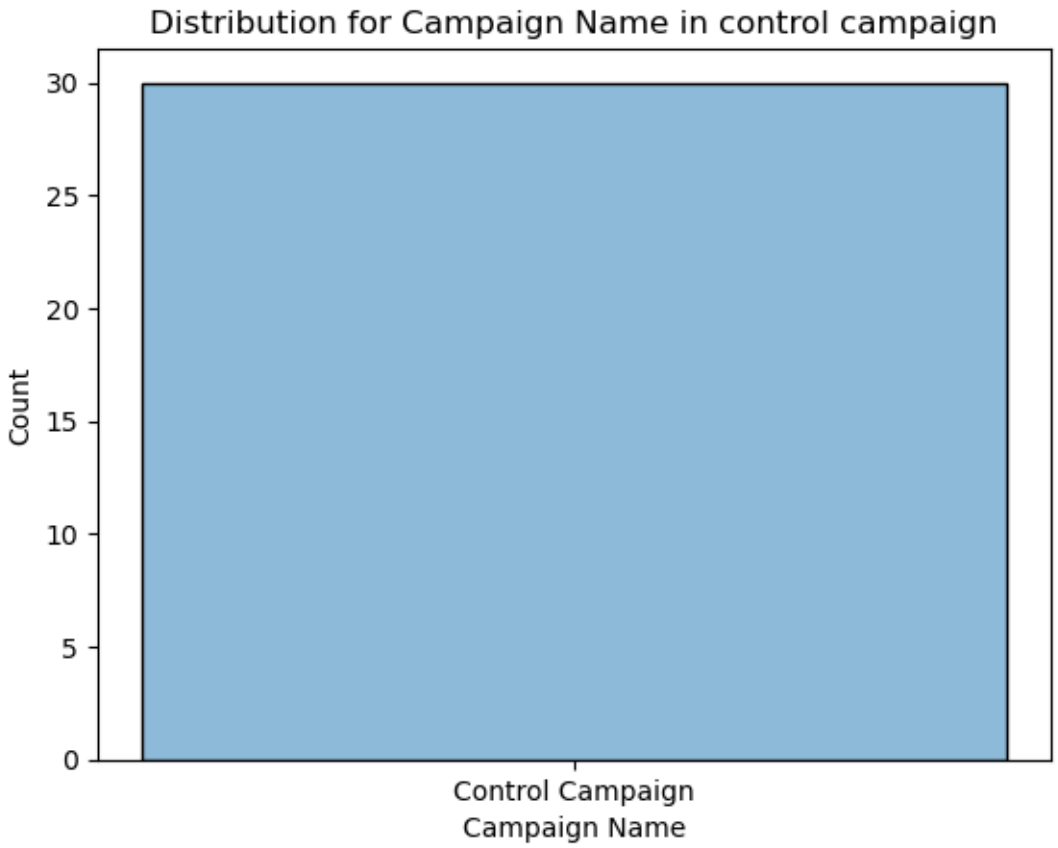

Boxplot for # of Add to Cart
200
400
600
800
1000
1200
1400
1600
1800
# of Add to Cart

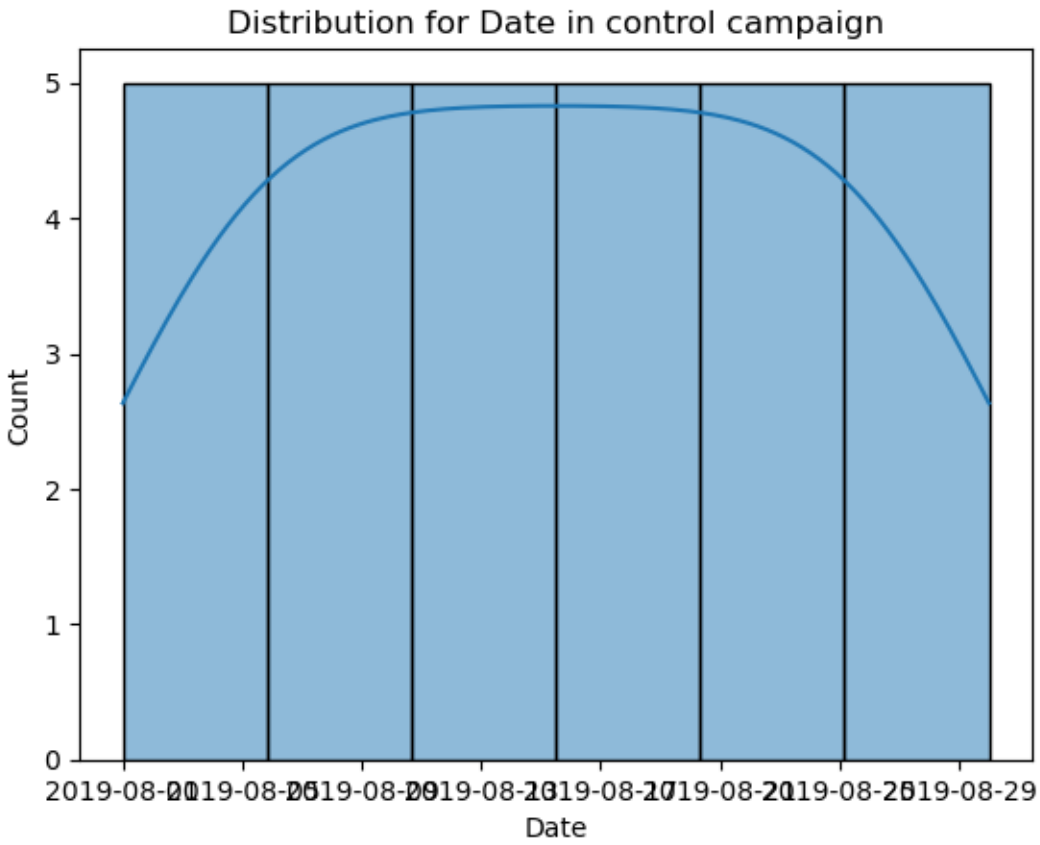

Boxplot for # of Purchase
200
300
400
500
600
700
800
900
# of Purchase

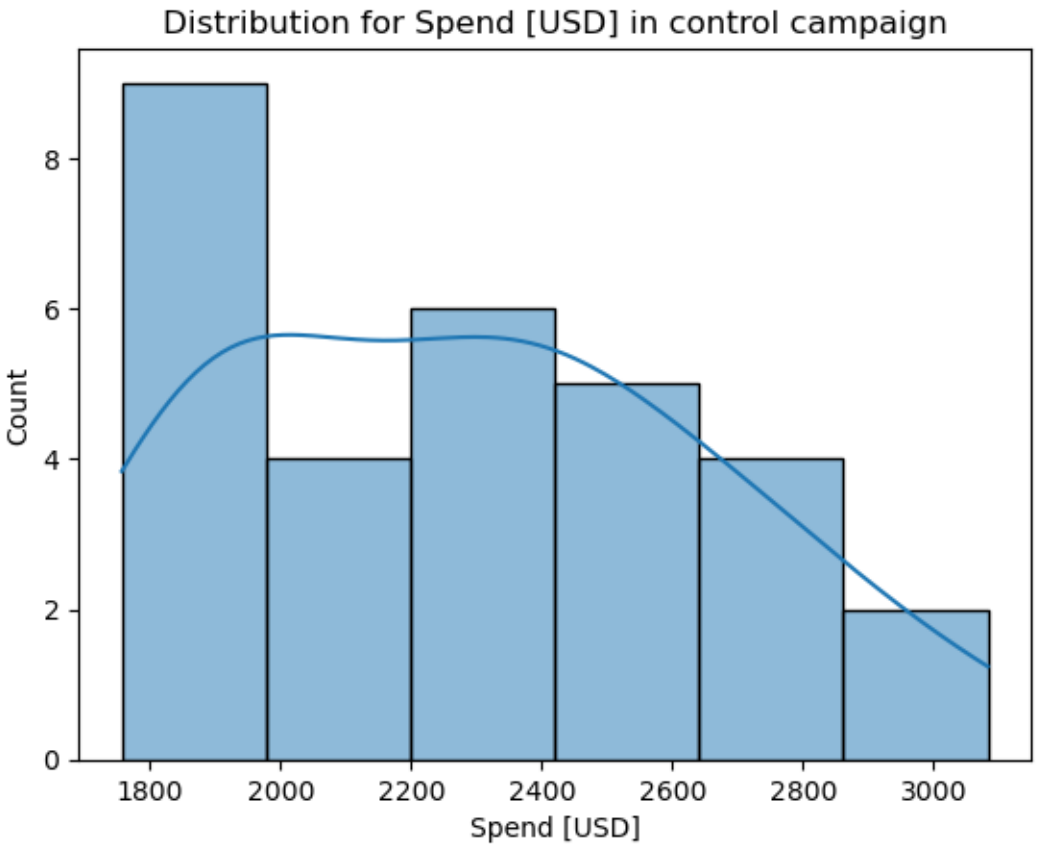


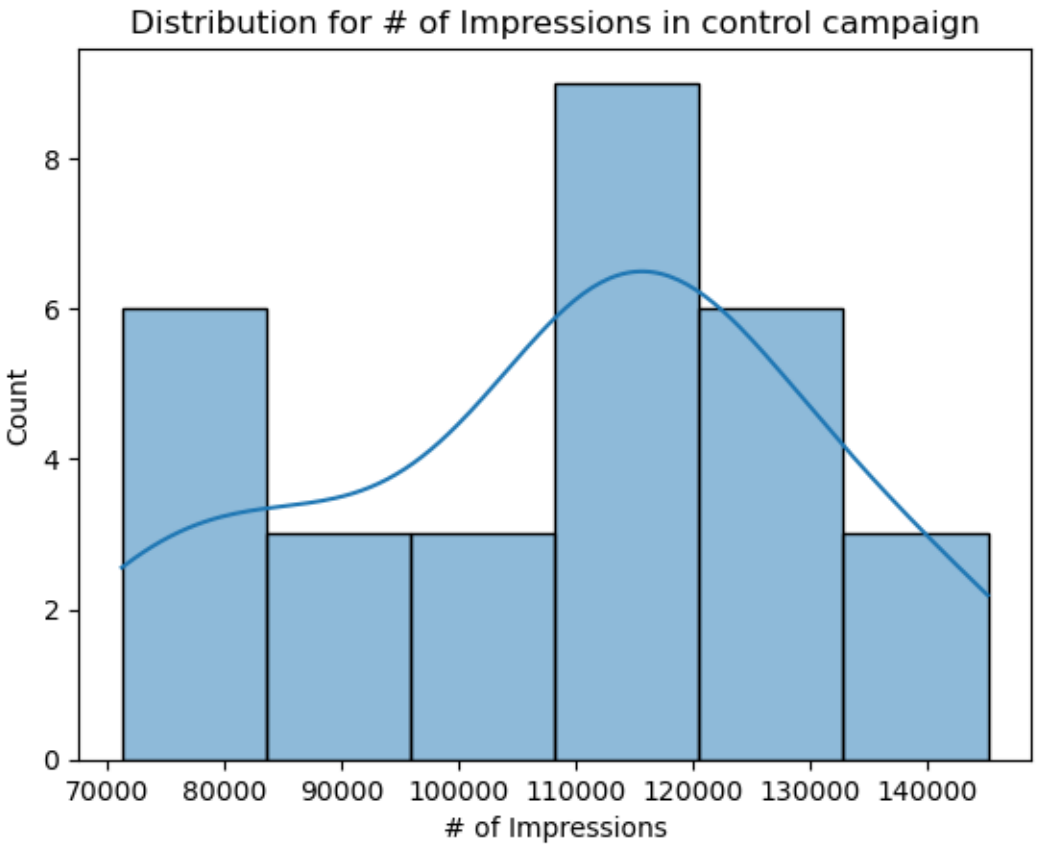
[1443]:
```
control_campaign = df[df['Campaign Name'] == 'Control Campaign']
test_campaign = df[df['Campaign Name'] == 'Test Campaign']
```

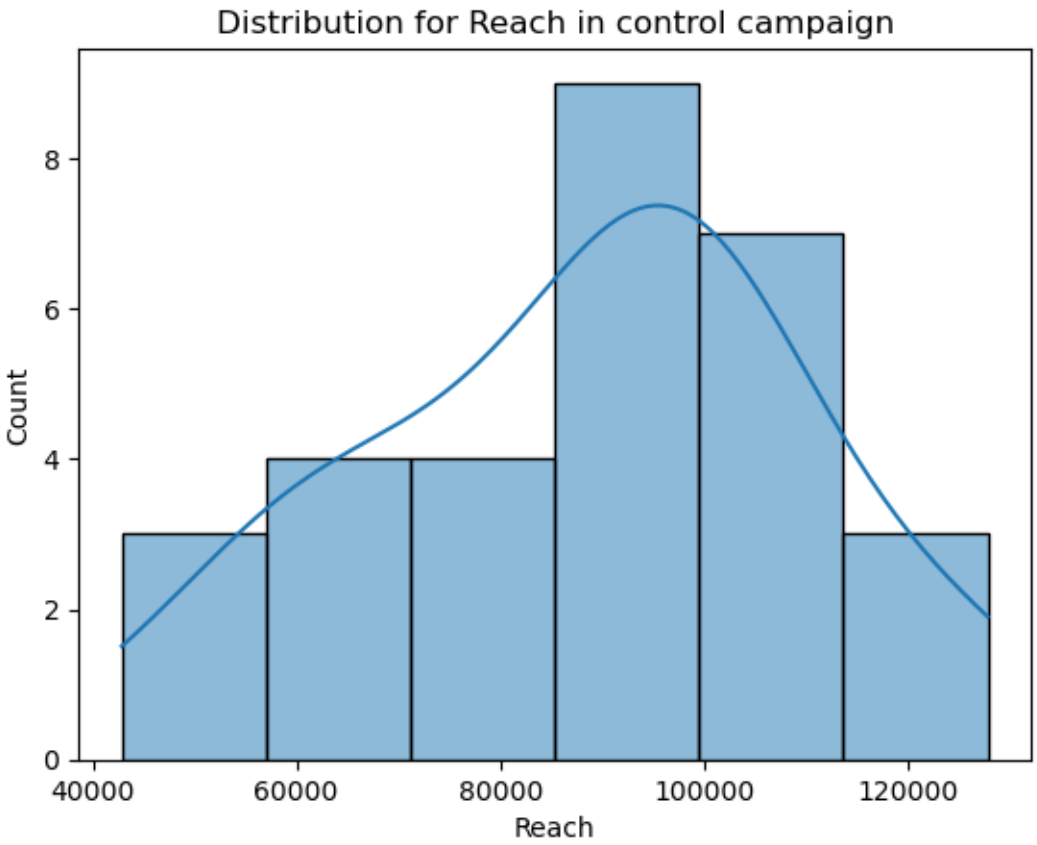
[1444]:
```
for column in control_campaign.columns:
    sns.histplot(control_campaign[column].dropna(), kde=True)
    plt.title(f'Distribution for {column} in control campaign')
    plt.show()
```

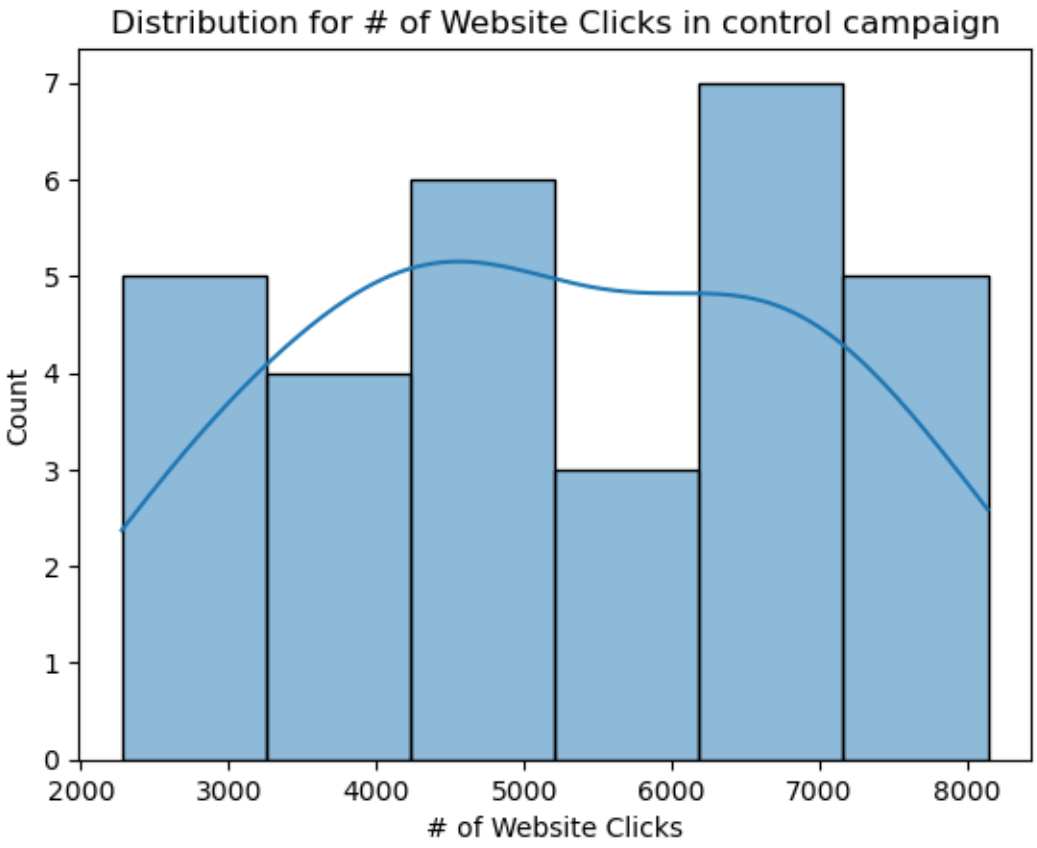

Distribution for Campaign Name in control campaign
30
25
20
15
10
5
0
Count
Control Campaign
Campaign Name

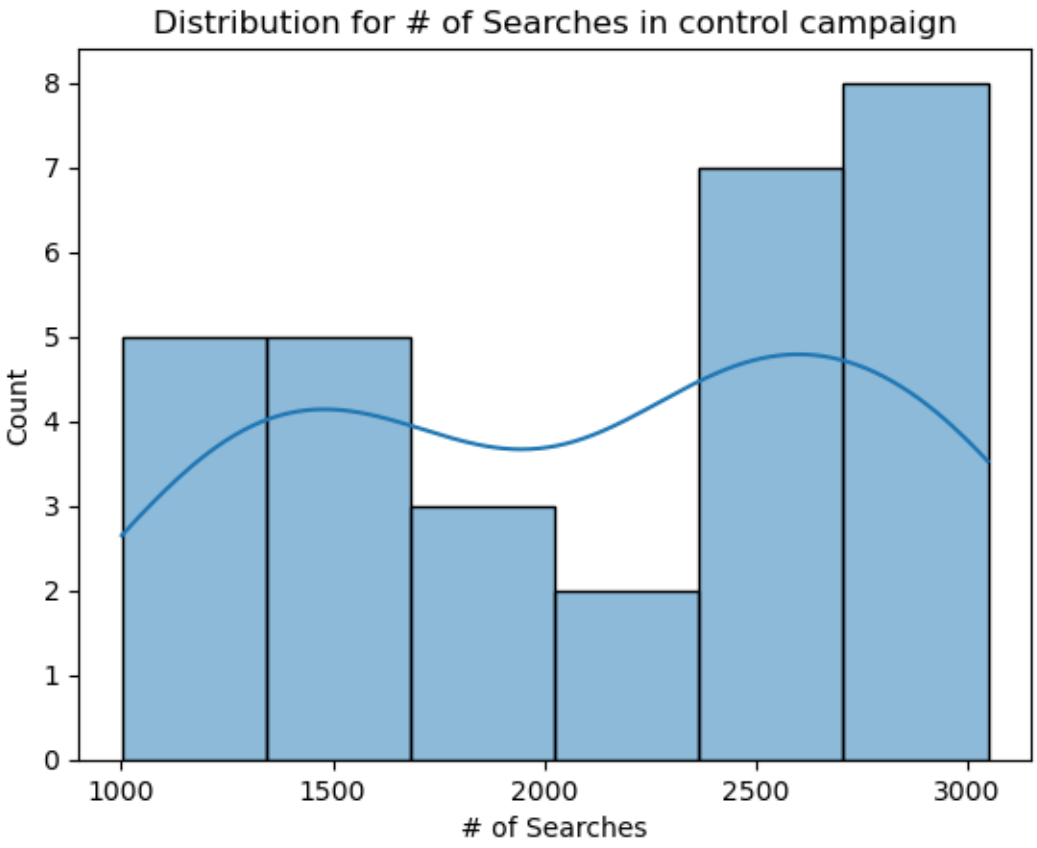

Distribution for Date in control campaign
Count
0
1
2
3
4
5
2019-08-20
2019-08-25
2019-08-29
Date

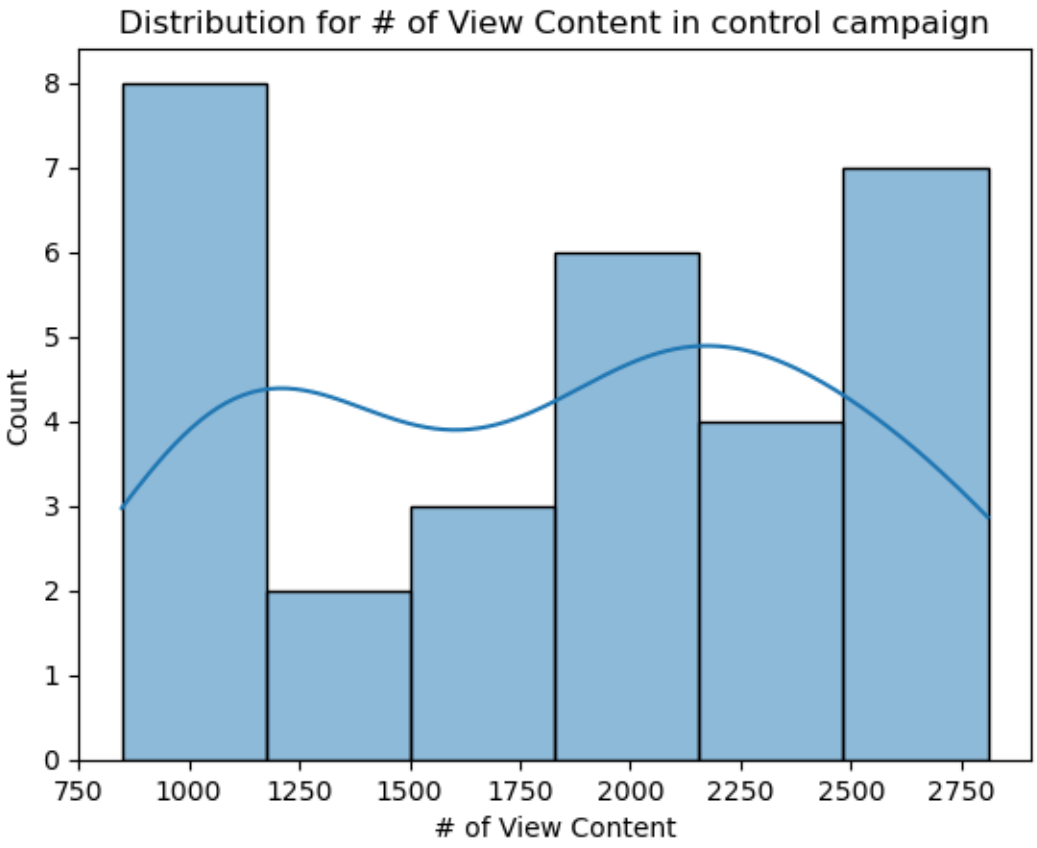

Distribution for Spend [USD] in control campaign
Count
0
2
4
6
8
1800
2000
2200
2400
2600
2800
3000
Spend [USD]

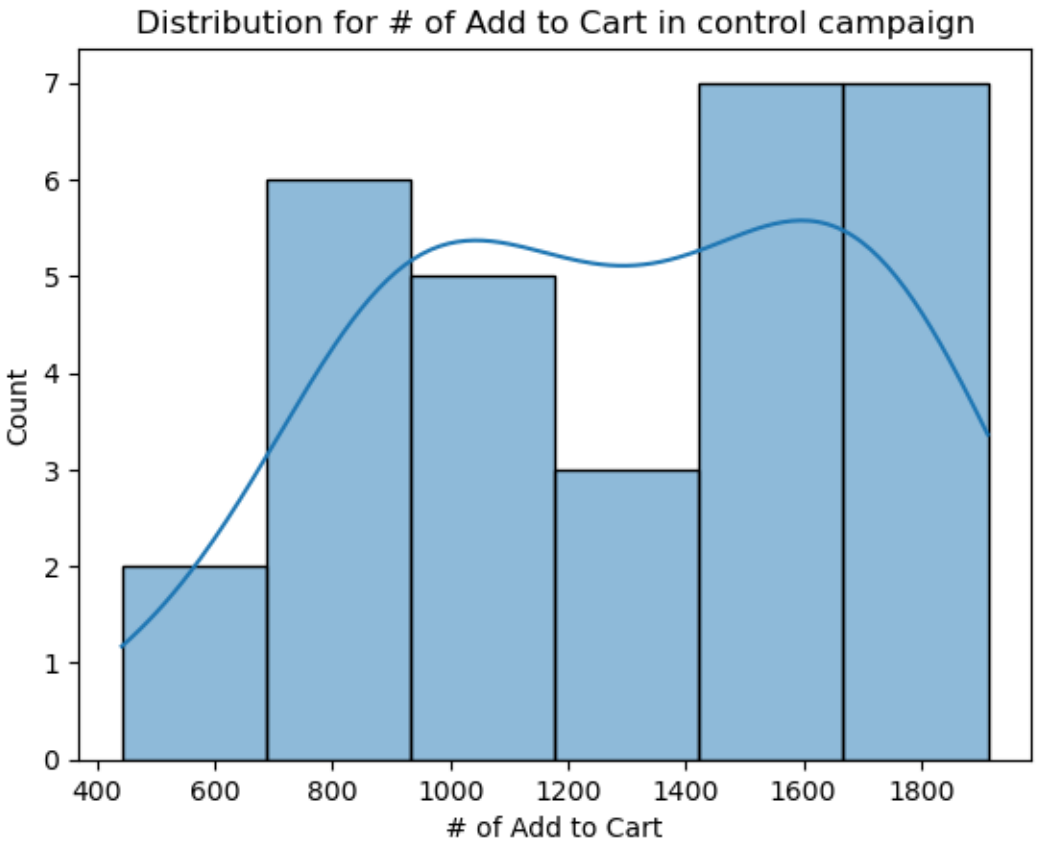

Distribution for # of Impressions in control campaign
Count
0
2
4
6
8
70000
80000
90000
100000
110000
120000
130000
140000
# of Impressions

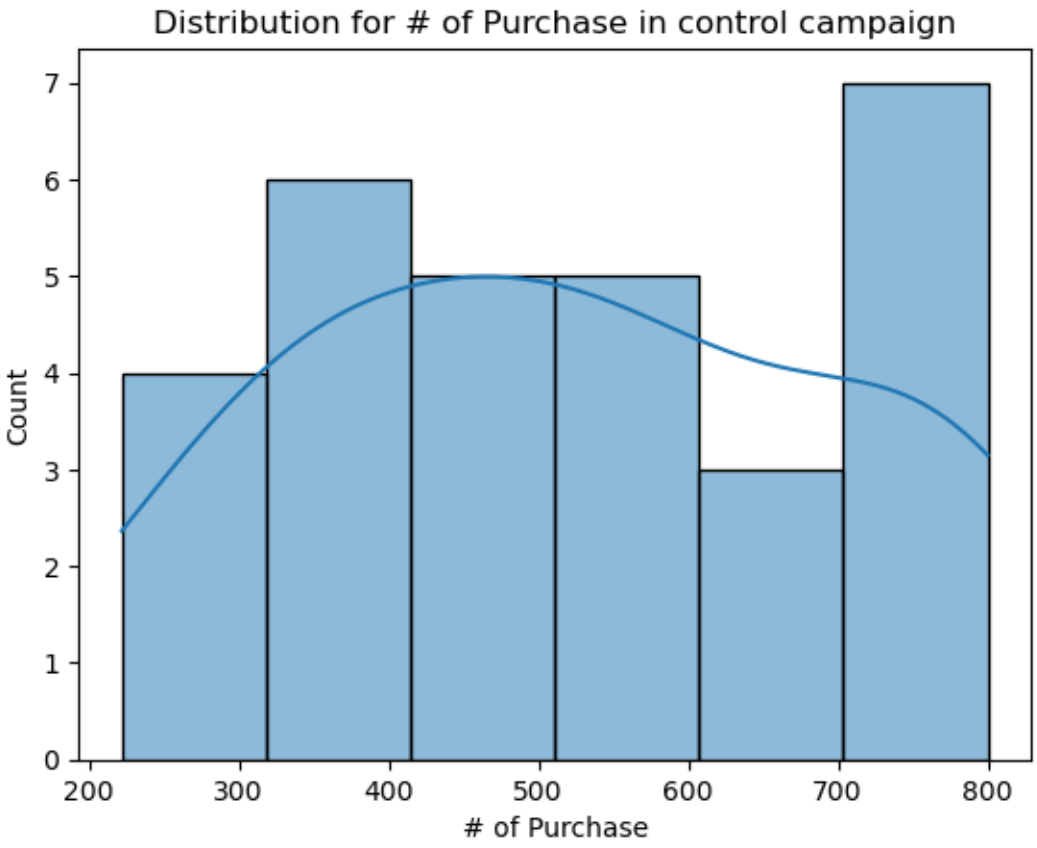

Distribution for Reach in control campaign
Count
Reach
0
2
4
6
8
40000
60000
80000
100000
120000

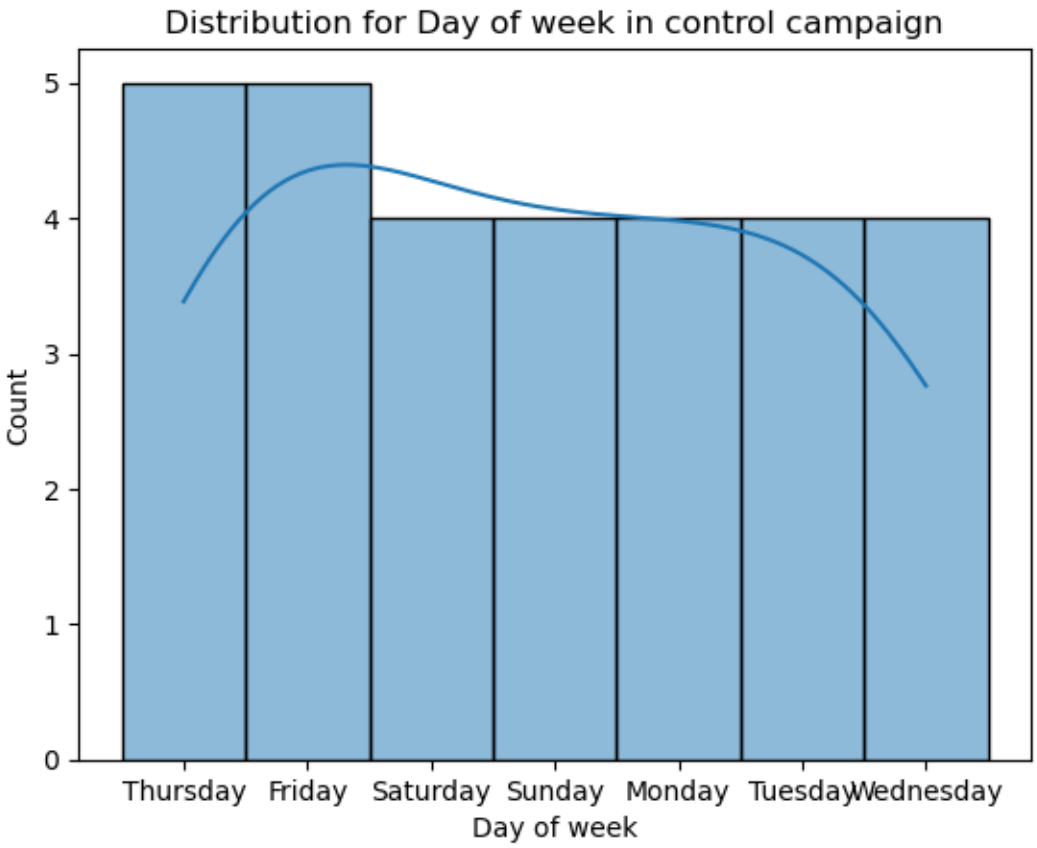

Distribution for # of Website Clicks in control campaign
Count
# of Website Clicks
0
1
2
3
4
5
6
7
2000
3000
4000
5000
6000
7000
8000


Distribution for # of Searches in control campaign
Count
# of Searches
0
1
2
3
4
5
6
7
8
1000
1500
2000
2500
3000

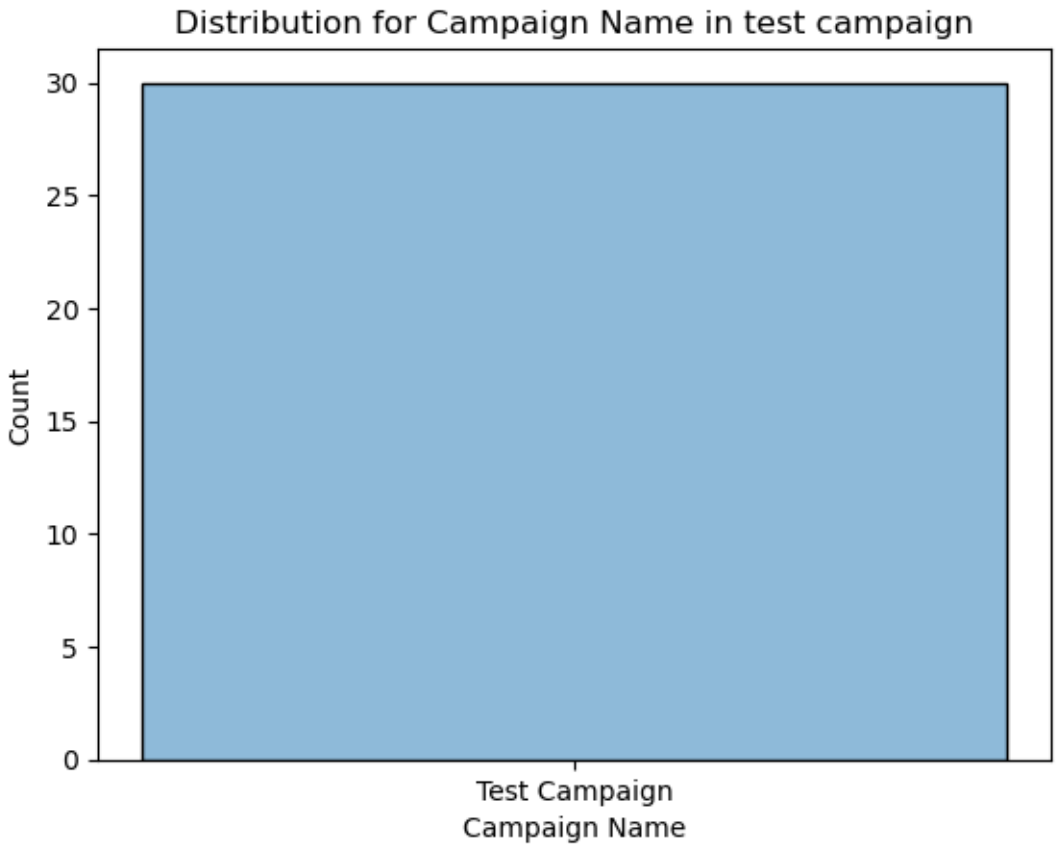

Distribution for # of View Content in control campaign
Count
# of View Content
0
1
2
3
4
5
6
7
8
750
1000
1250
1500
1750
2000
2250
2500
2750

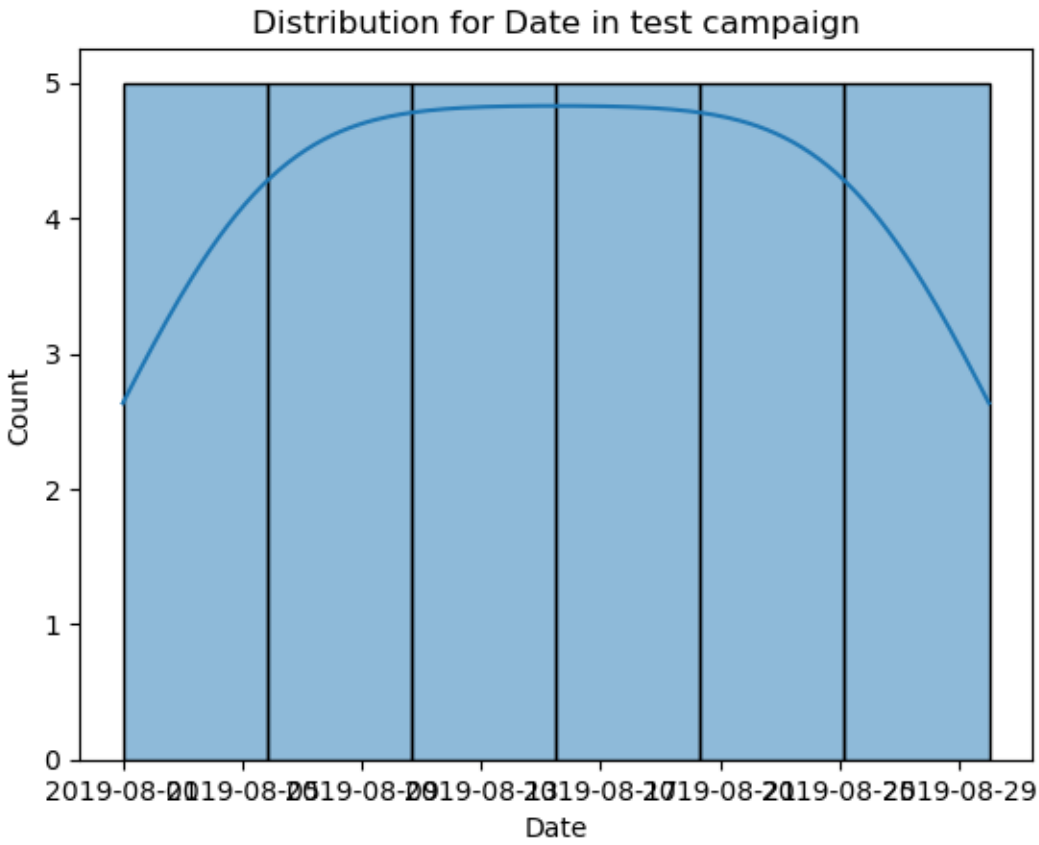

Distribution for # of Add to Cart in control campaign
Count
# of Add to Cart
0
1
2
3
4
5
6
7
400
600
800
1000
1200
1400
1600
1800

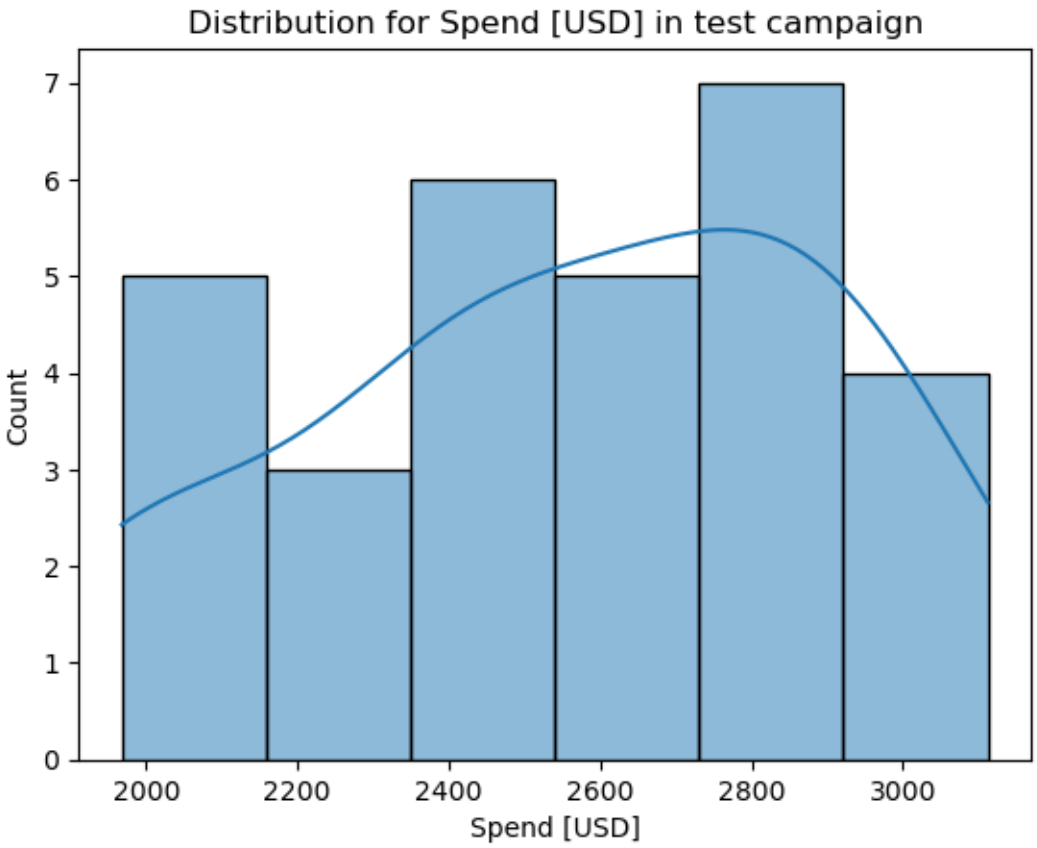

Distribution for # of Purchase in control campaign
Count
# of Purchase
0
1
2
3
4
5
6
7
200
300
400
500
600
700
800

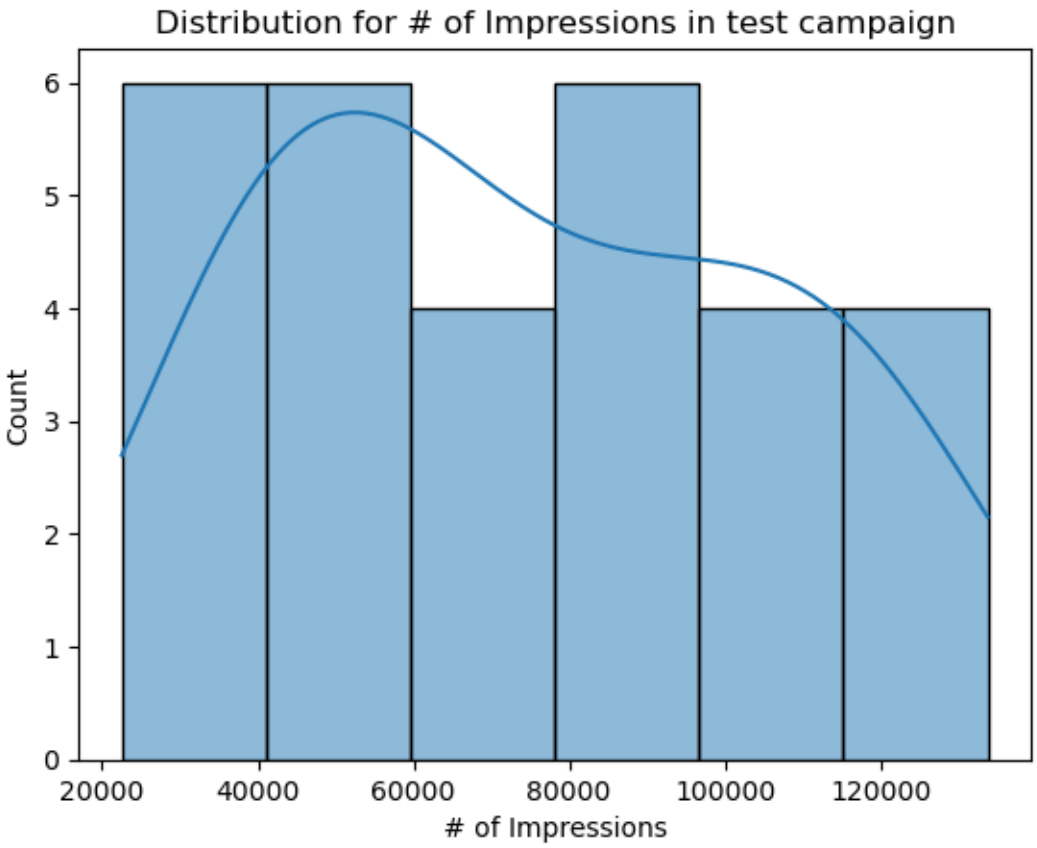


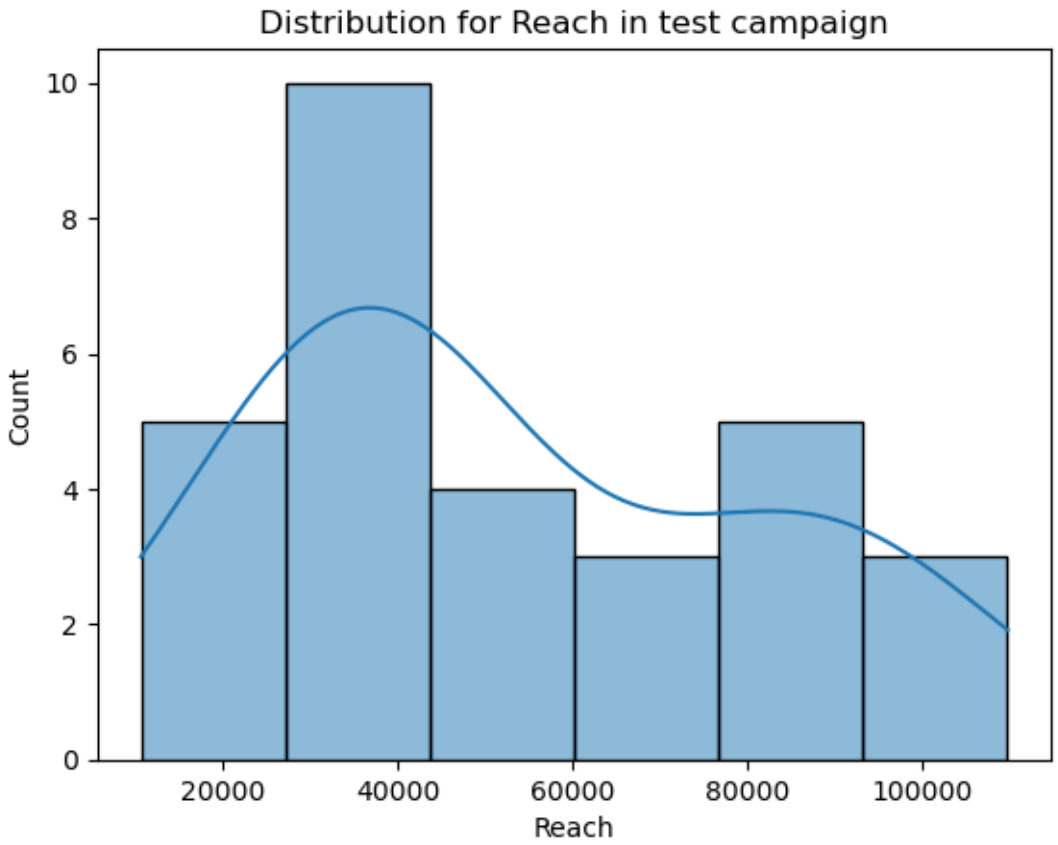
[1445]:
```
for column in test_campaign.columns:
    sns.histplot(test_campaign[column].dropna(), kde=True)
    plt.title(f'Distribution for {column} in test campaign')
    plt.show()
```

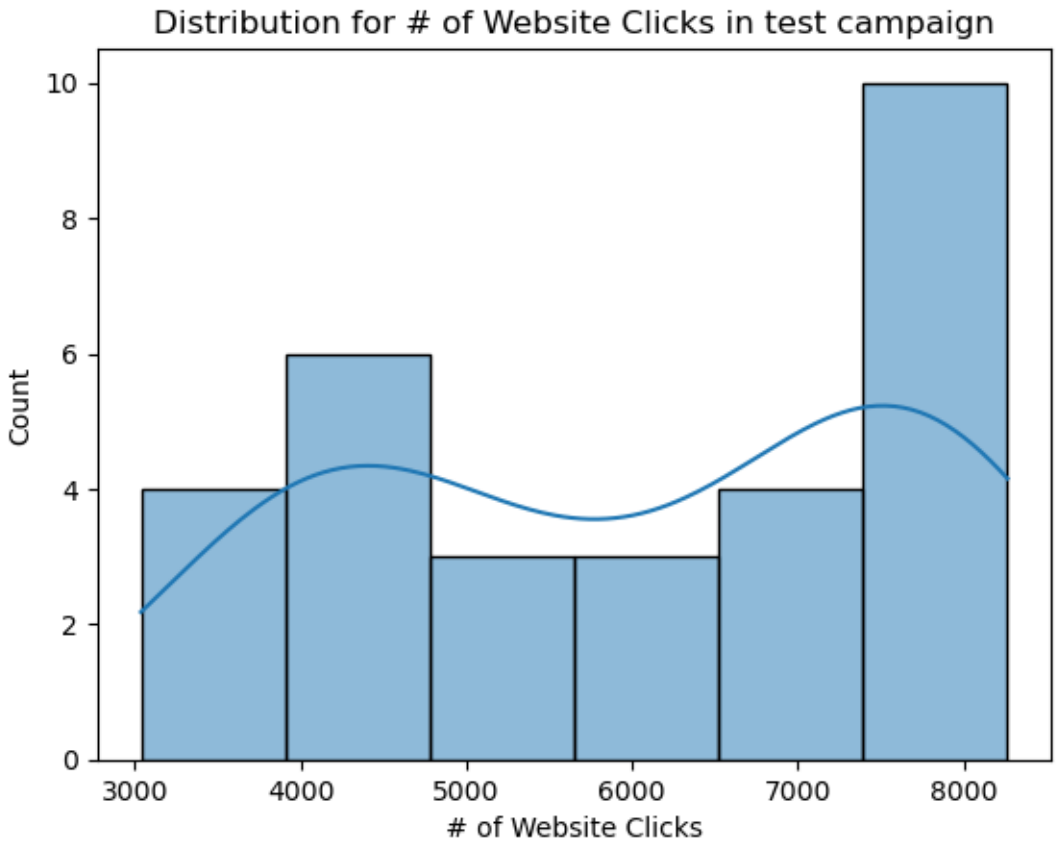

Distribution for Campaign Name in test campaign
Count
30
25
20
15
10
5
0
Test Campaign
Campaign Name

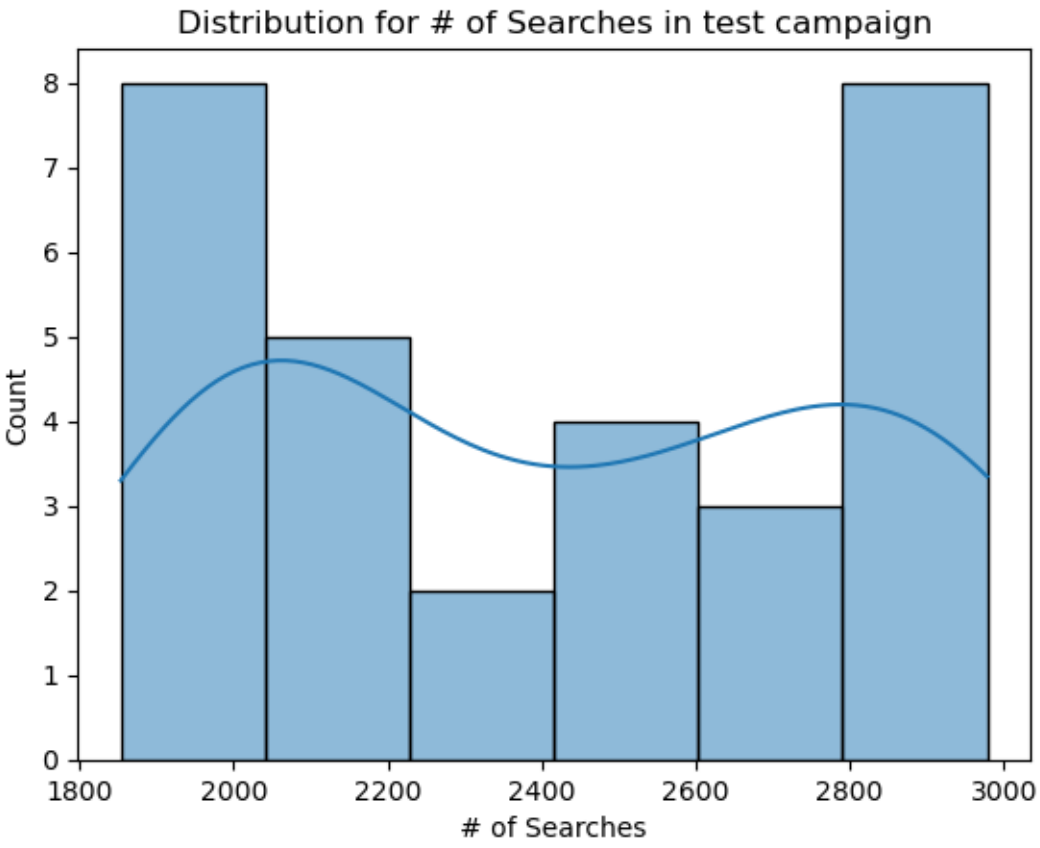

Distribution for Date in test campaign
Count
0
1
2
3
4
5
2019-08-20
2019-08-21
2019-08-25
2019-08-26
2019-08-27
2019-08-28
2019-08-29
Date

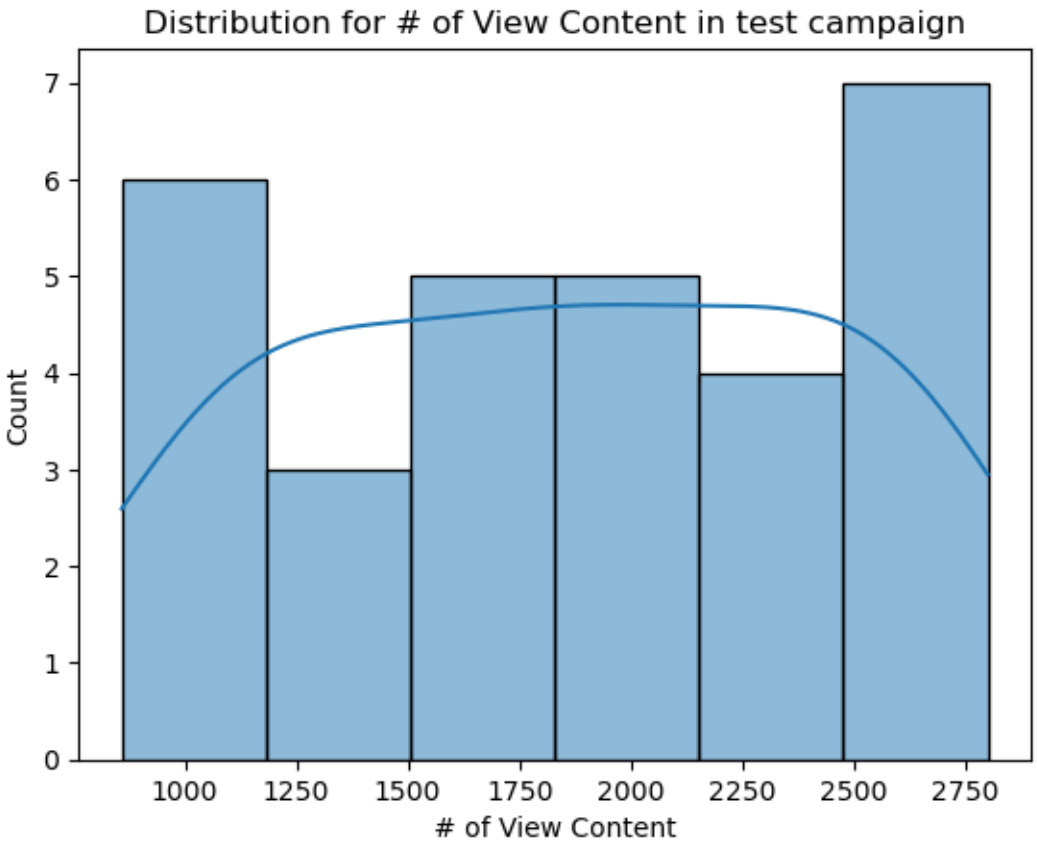

Distribution for Spend [USD] in test campaign
Count
Spend [USD]
0
1
2
3
4
5
6
7
2000
2200
2400
2600
2800
3000

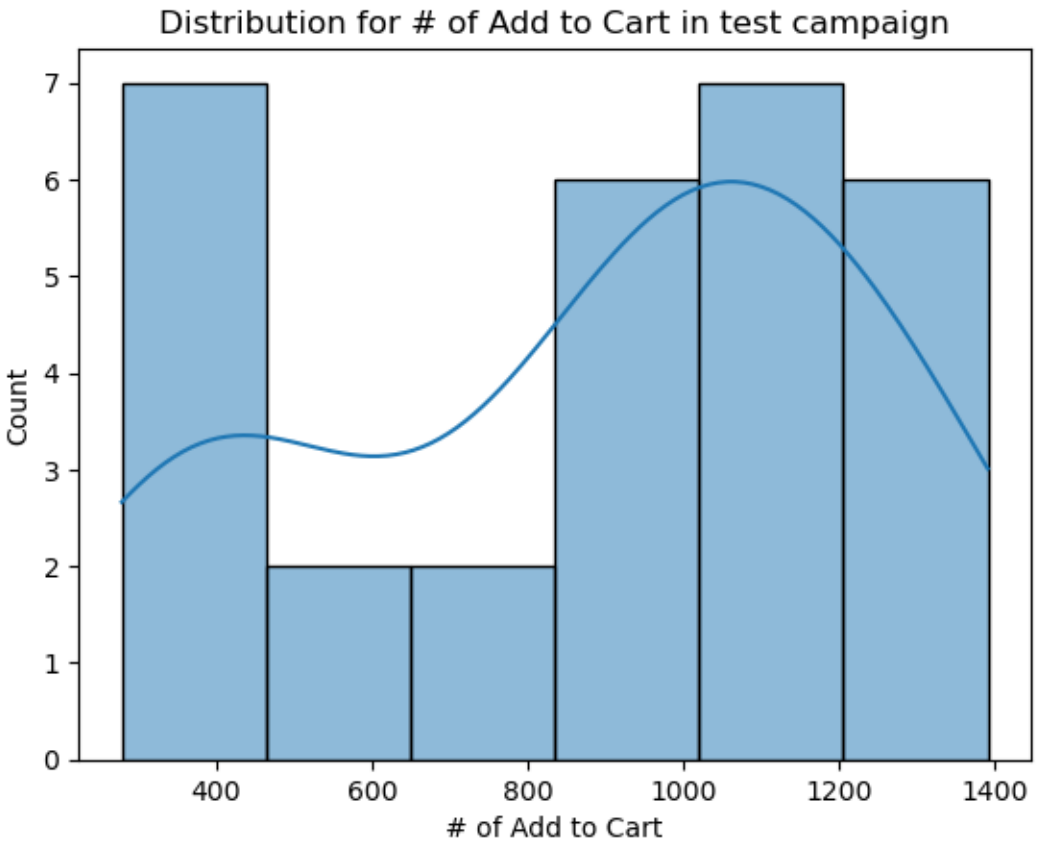

Distribution for # of Impressions in test campaign
Count
# of Impressions
0
1
2
3
4
5
6
20000
40000
60000
80000
100000
120000

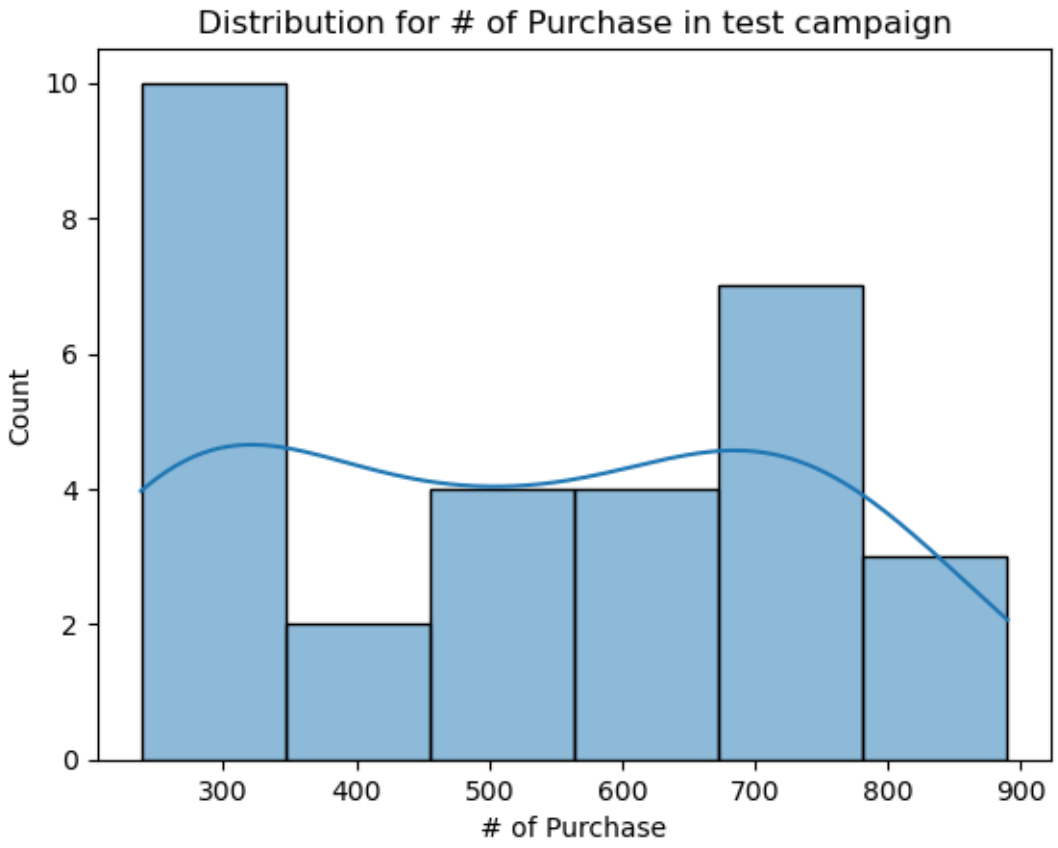

Distribution for Reach in test campaign
Count
Reach
0
2
4
6
8
10
20000
40000
60000
80000
100000

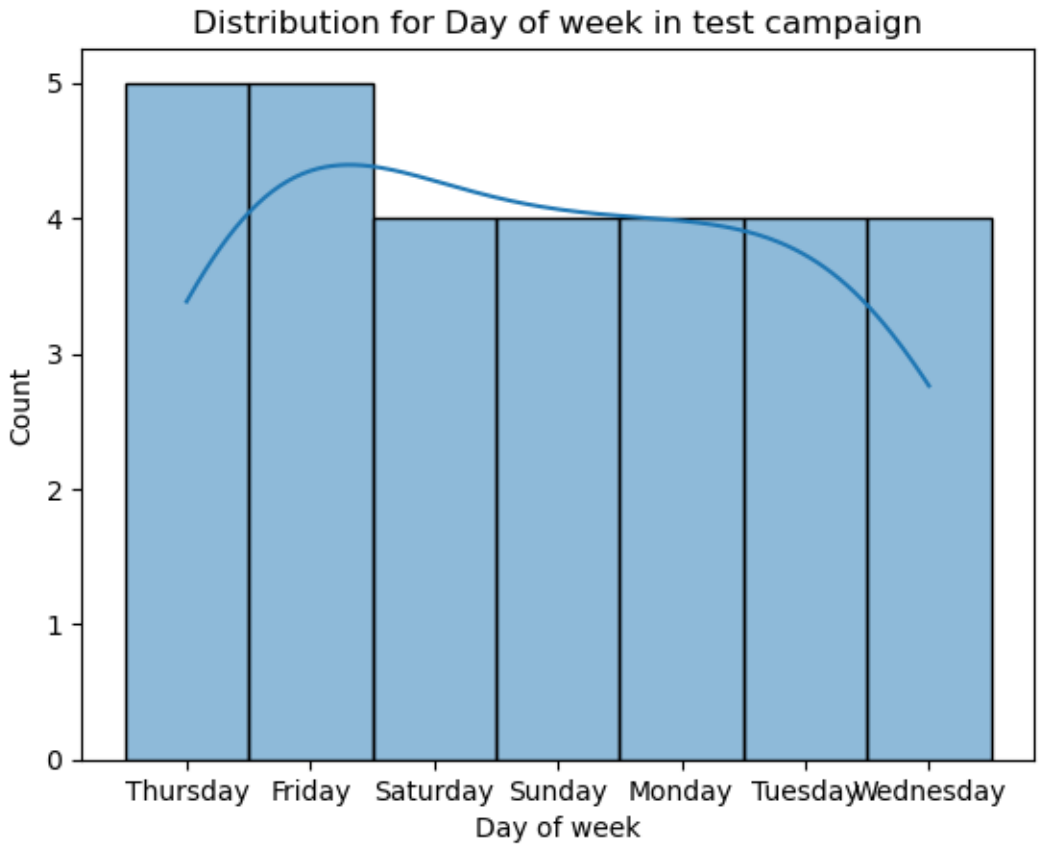

Distribution for # of Website Clicks in test campaign
Count
# of Website Clicks
0
2
4
6
8
10
3000
4000
5000
6000
7000
8000


Distribution for # of Searches in test campaign
Count
# of Searches
0
1
2
3
4
5
6
7
8
1800
2000
2200
2400
2600
2800
3000


Distribution for # of View Content in test campaign
Count
# of View Content
0
1
2
3
4
5
6
7
1000
1250
1500
1750
2000
2250
2500
2750


Distribution for # of Add to Cart in test campaign
Count
# of Add to Cart
0
1
2
3
4
5
6
7
400
600
800
1000
1200
1400


Distribution for # of Purchase in test campaign
Count
# of Purchase
0
2
4
6
8
10
300
400
500
600
700
800
900



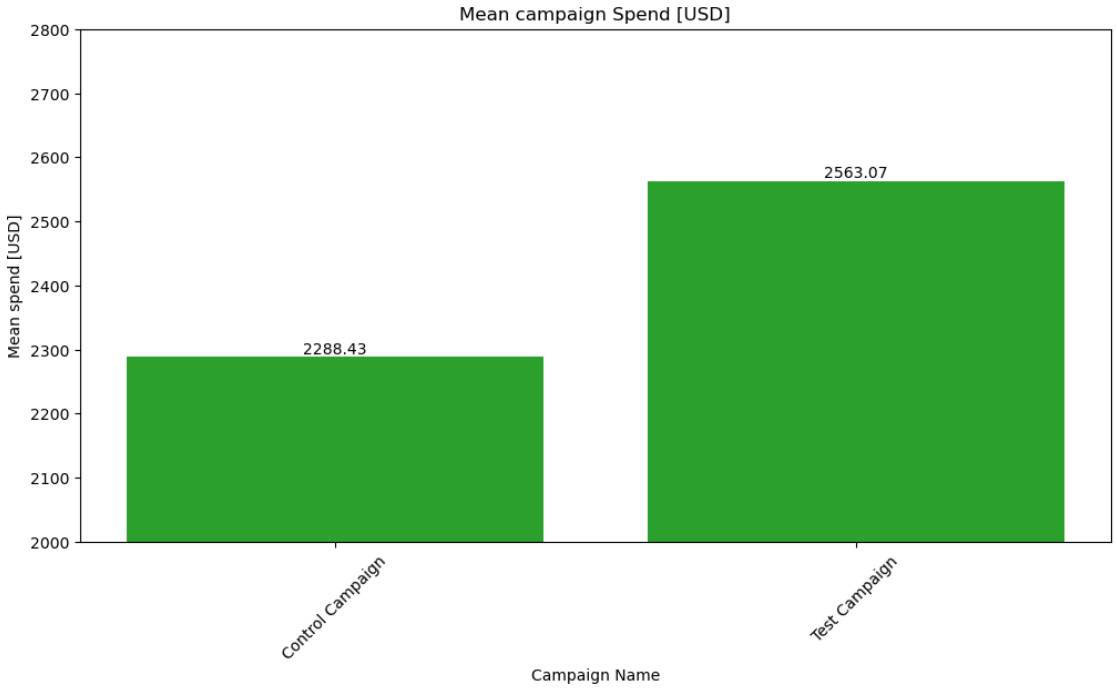
Calculation and Visualization of Key Metrics

Average and Median Expenditure for Campaigns. Total Expenditure and Purchases for Campaigns. Cost Per Purchase

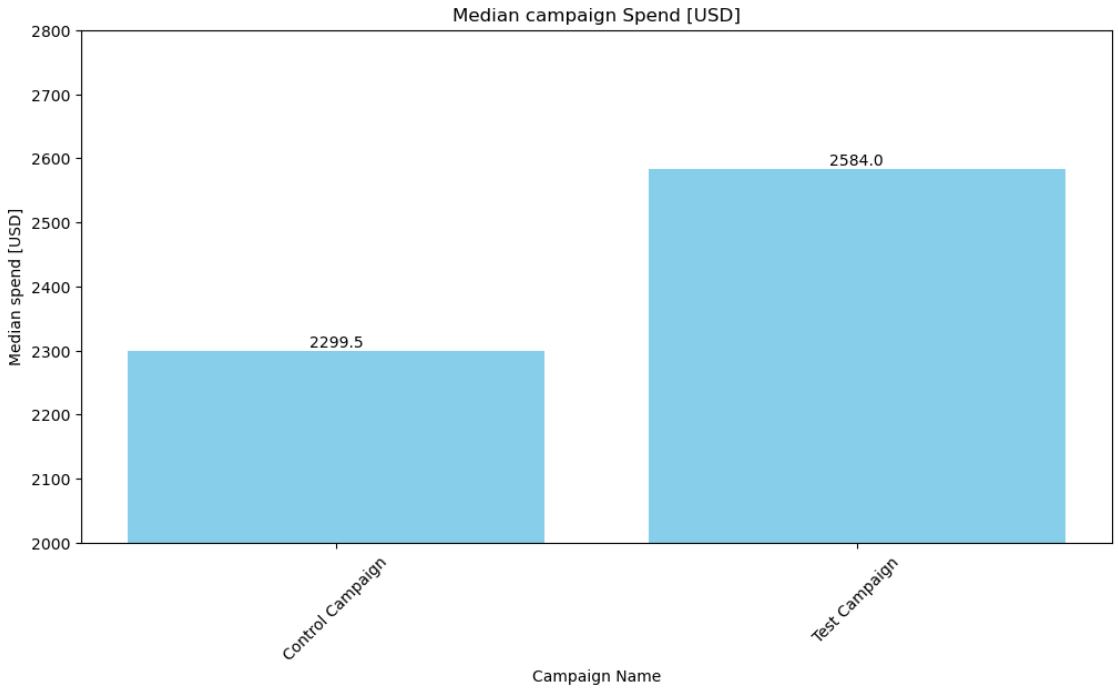
[1452]:
```
campaigns_spend = df.groupby('Campaign Name')['Spend [USD]']
mean_spend = campaigns_spend.mean()
median_spend = campaigns_spend.median()
total_spend = df.groupby('Campaign Name')['Spend [USD]'].sum()
total_purchase = df.groupby('Campaign Name')['# of Purchase'].sum()
cost_per_purchase = total_spend / total_purchase
```

[1455]:
```
campaign = mean_spend.index.tolist()
spend = mean_spend.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, spend, color='#2ca02c')
plt.xlabel('Campaign Name')
plt.ylabel('Mean spend [USD]')
plt.title('Mean campaign Spend [USD]')
plt.xticks(rotation=45)
```

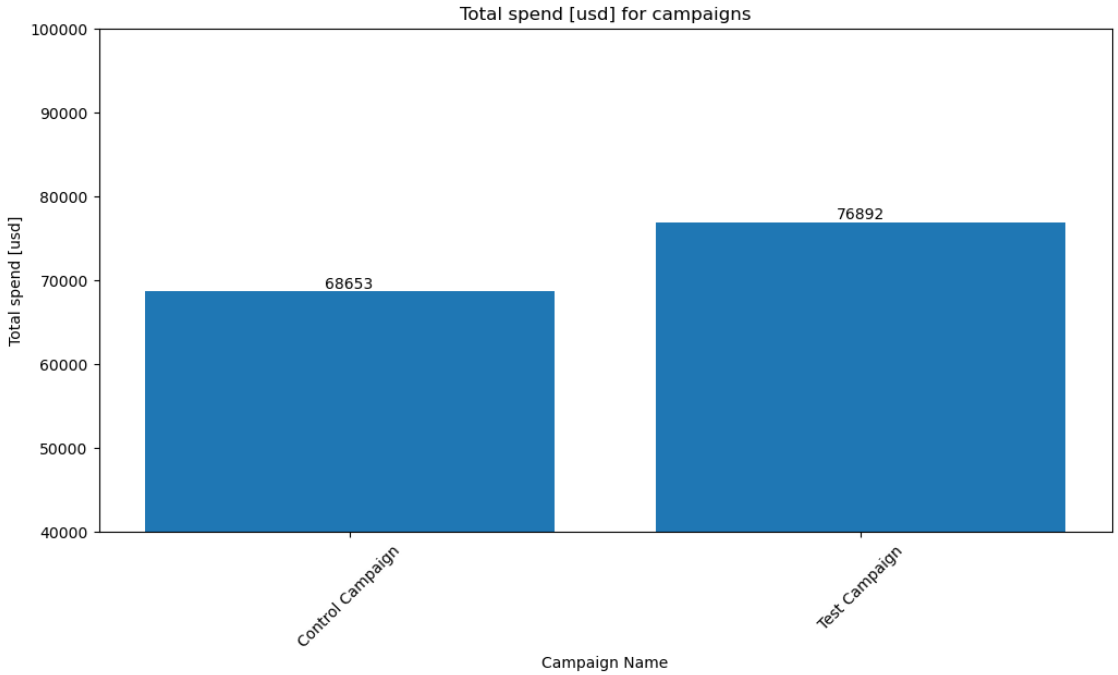
```
plt.ylim(2000, 2800)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



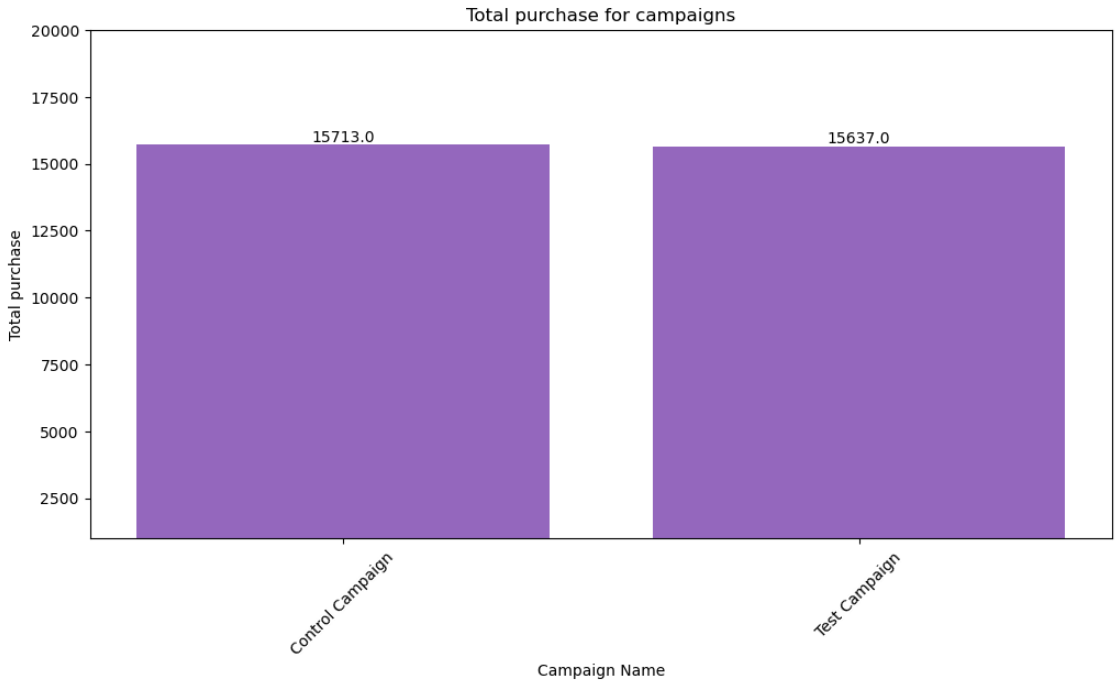
[1456]:
```
campaign = median_spend.index.tolist()
spend = median_spend.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, spend, color='skyblue')
plt.xlabel('Campaign Name')
plt.ylabel('Median spend [USD]')
plt.title('Median campaign Spend [USD]')
plt.xticks(rotation=45)
plt.ylim(2000, 2800)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



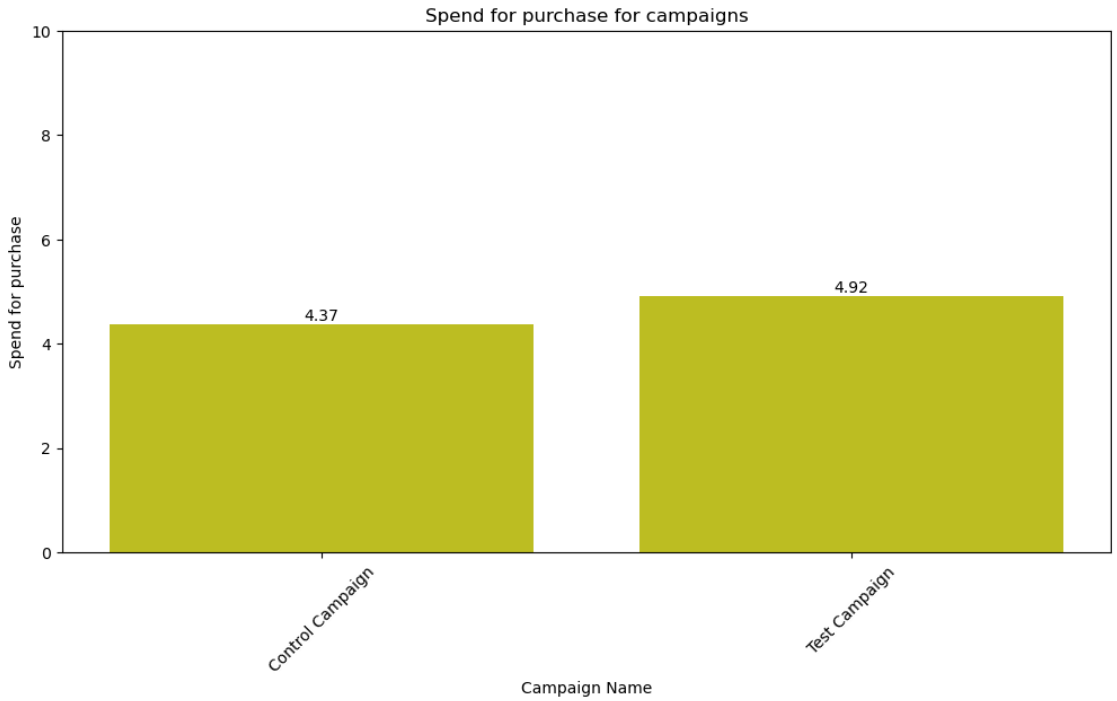
[1457]:
```
campaign = total_spend.index.tolist()
spend = total_spend.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, spend, color='#1f77b4')
plt.xlabel('Campaign Name')
plt.ylabel('Total spend [usd]')
plt.title('Total spend [usd] for campaigns')
plt.xticks(rotation=45)
plt.ylim(40000, 100000)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



[1458]:
```
campaign = total_purchase.index.tolist()
purchase = total_purchase.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, purchase, color='#9467bd')
plt.xlabel('Campaign Name')
plt.ylabel('Total purchase')
plt.title('Total purchase for campaigns')
plt.xticks(rotation=45)
plt.ylim(1000, 20000)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



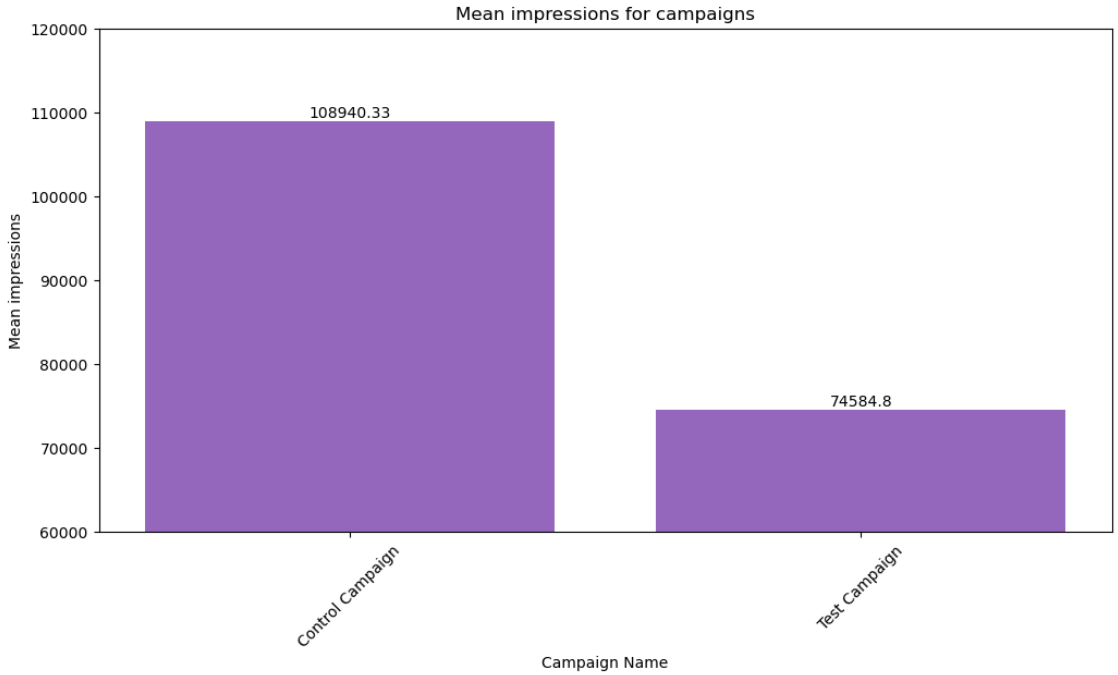
[1459]:
```
campaign = cost_per_purchase.index.tolist()
cost = cost_per_purchase.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, cost, color='#bcbd22')
plt.xlabel('Campaign Name')
plt.ylabel('Spend for purchase')
plt.title('Spend for purchase for campaigns')
plt.xticks(rotation=45)
plt.ylim(0, 10)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



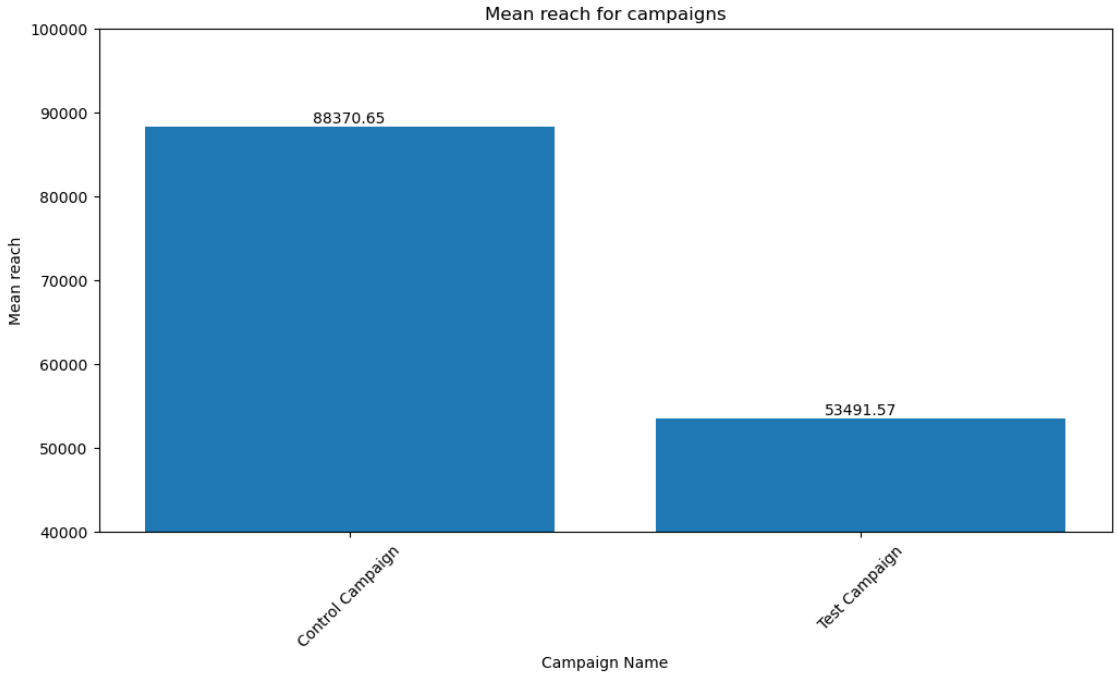
Average Number of Impressions, Unique Impressions (Reach), and Website Clicks.

[1461]:
```
average_impressions = df.groupby('Campaign Name')['# of Impressions'].mean()
average_reach = df.groupby('Campaign Name')['Reach'].mean()
average_website_clicks = df.groupby('Campaign Name')['# of Website Clicks'].
 ↪mean()
```

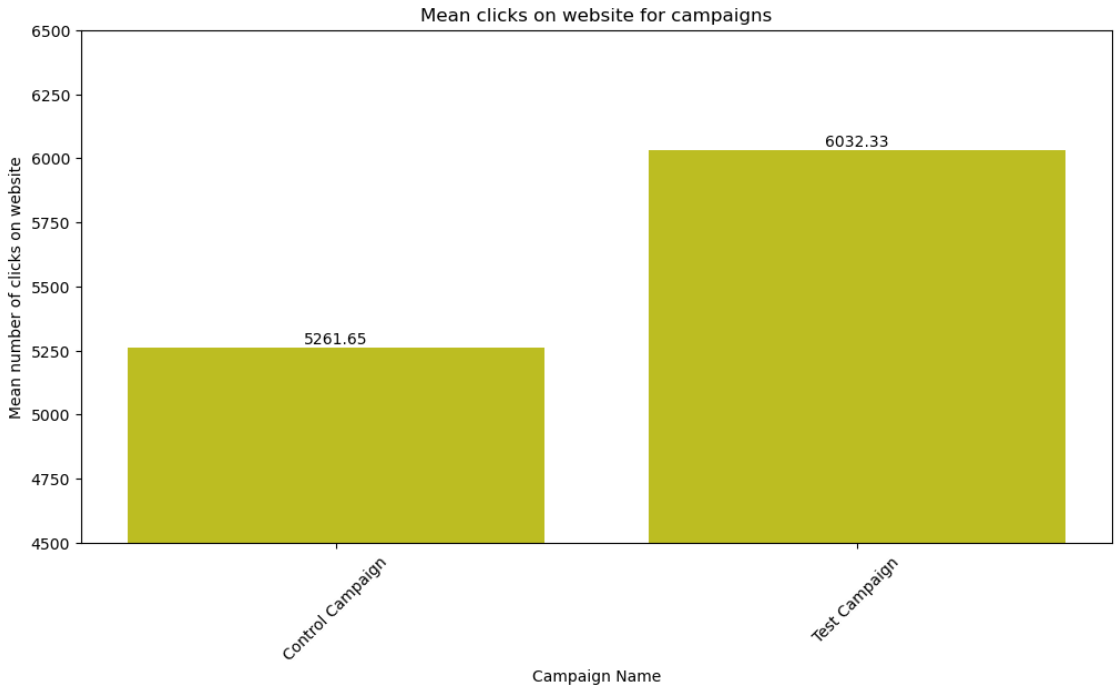
[1463]:
```
campaign = average_impressions.index.tolist()
impressions = average_impressions.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, impressions, color='#9467bd')
plt.xlabel('Campaign Name')
plt.ylabel('Mean impressions')
plt.title('Mean impressions for campaigns')
plt.xticks(rotation=45)
plt.ylim(60000, 120000)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



[1464]:
```
campaign = average_reach.index.tolist()
reach = average_reach.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, reach, color='#1f77b4')
plt.xlabel('Campaign Name')
plt.ylabel('Mean reach')
plt.title('Mean reach for campaigns')
plt.xticks(rotation=45)
plt.ylim(40000, 100000)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



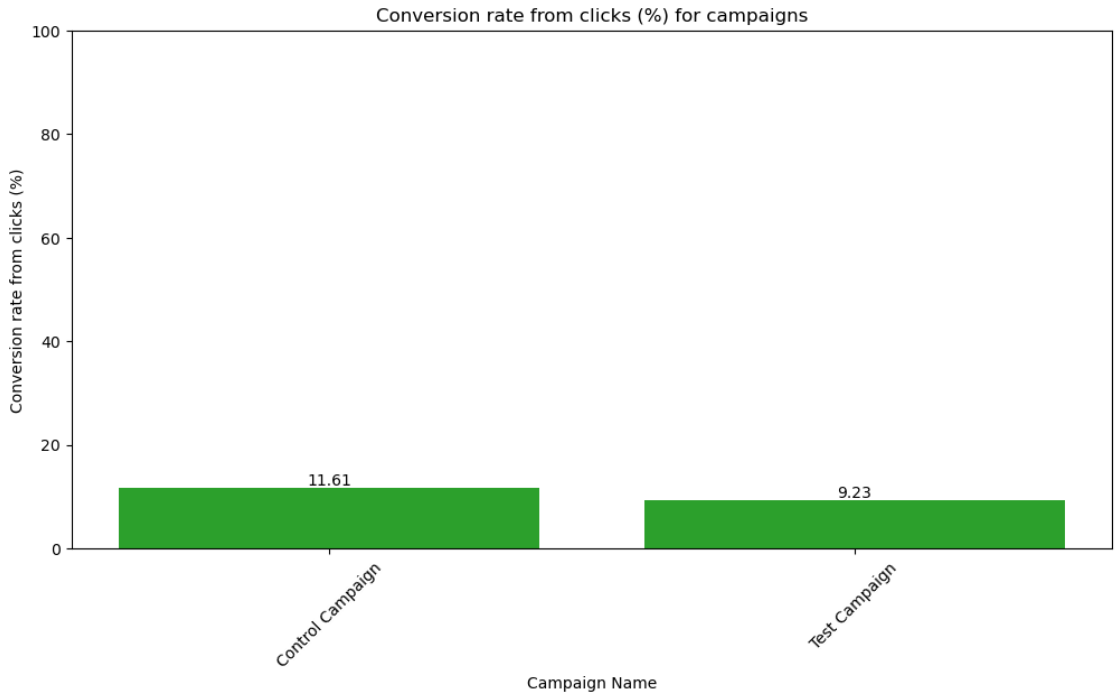
[1465]:
```
campaign = average_website_clicks.index.tolist()
clicks = average_website_clicks.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, clicks, color='#bcbd22')
plt.xlabel('Campaign Name')
plt.ylabel('Mean number of clicks on website')
plt.title('Mean clicks on website for campaigns')
plt.xticks(rotation=45)
plt.ylim(4500, 6500)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



Conversion Assessment

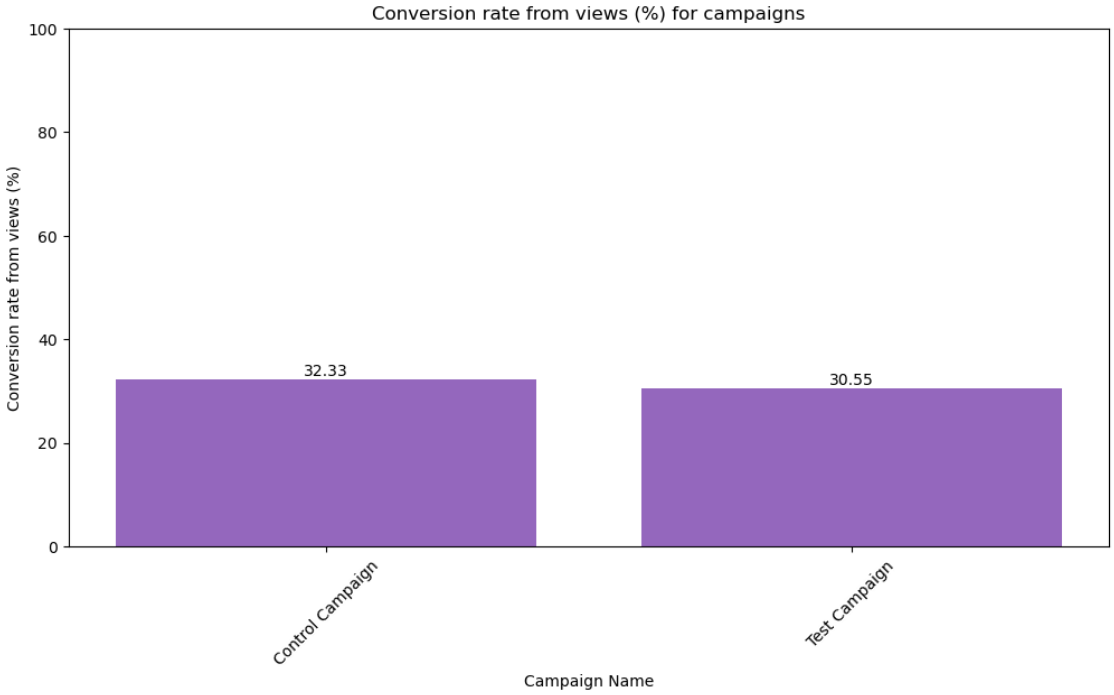
[1467]:
```
df['Conversion Rate From clicks(%)'] = (df['# of Purchase'] / df['# of Website␣
 ↪Clicks']) * 100
result = df.groupby('Campaign Name')['Conversion Rate From clicks(%)'].mean()
```

[1468]:
```
campaign = result.index.tolist()
rate = result.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, rate, color='#2ca02c')
plt.xlabel('Campaign Name')
plt.ylabel('Conversion rate from clicks (%)')
plt.title('Conversion rate from clicks (%) for campaigns')
plt.xticks(rotation=45)
plt.ylim(0, 100)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



[1471]:
```
df['Conversion Rate From views(%)'] = (df['# of Purchase'] / df['# of View␣
↪Content']) * 100
result = df.groupby('Campaign Name')['Conversion Rate From views(%)'].mean()
```

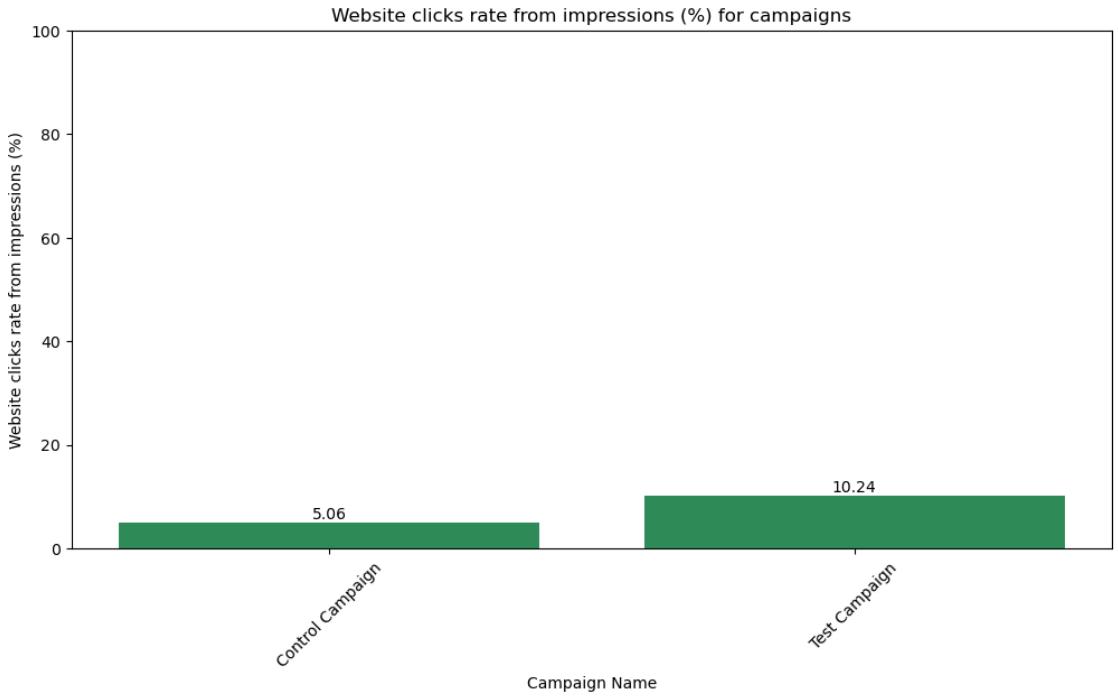
[1475]:
```
campaign = result.index.tolist()
rate = result.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, rate, color='#9467bd')
plt.xlabel('Campaign Name')
plt.ylabel('Conversion rate from views (%)')
plt.title('Conversion rate from views (%) for campaigns')
plt.xticks(rotation=45)
plt.ylim(0, 100)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
↪ha='center', va='bottom')
plt.show()
```



Conversion Funnel

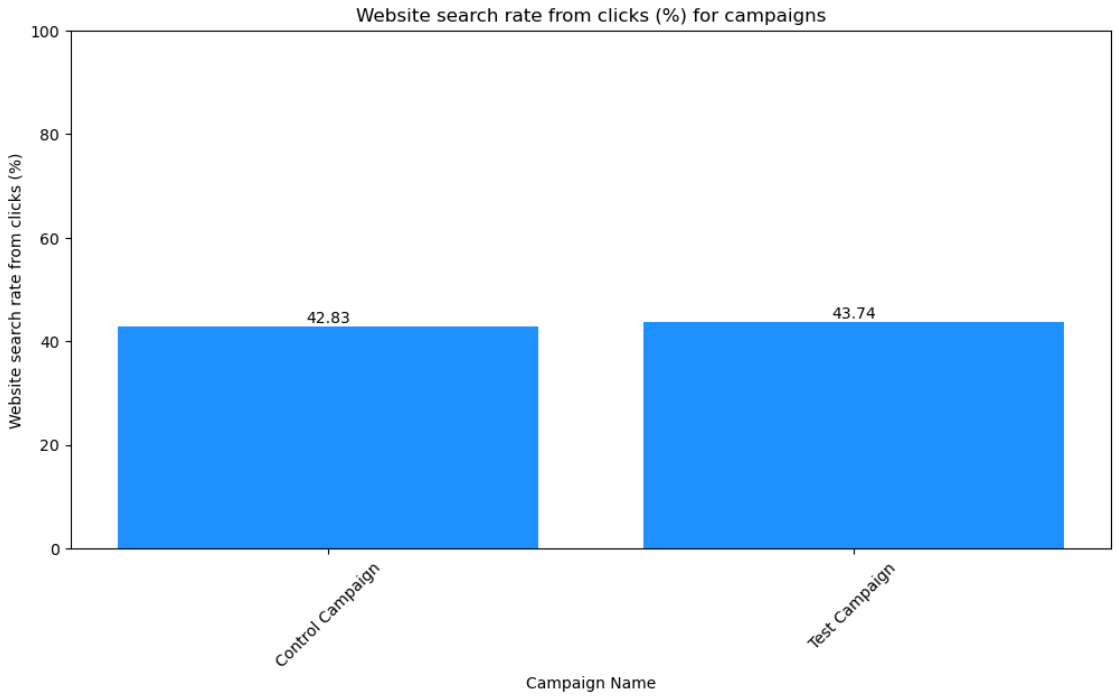
[1487]:
```
df['Website clicks Rate From Impressions(%)'] = (df['# of Website Clicks'] /␣
 ↪df['# of Impressions']) * 100
result = df.groupby('Campaign Name')['Website clicks Rate From Impressions(%)'].
 ↪mean()
```

[1489]:
```
campaign = result.index.tolist()
rate = result.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, rate, color='#2E8B57')
plt.xlabel('Campaign Name')
plt.ylabel('Website clicks rate from impressions (%)')
plt.title('Website clicks rate from impressions (%) for campaigns')
plt.xticks(rotation=45)
plt.ylim(0, 100)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



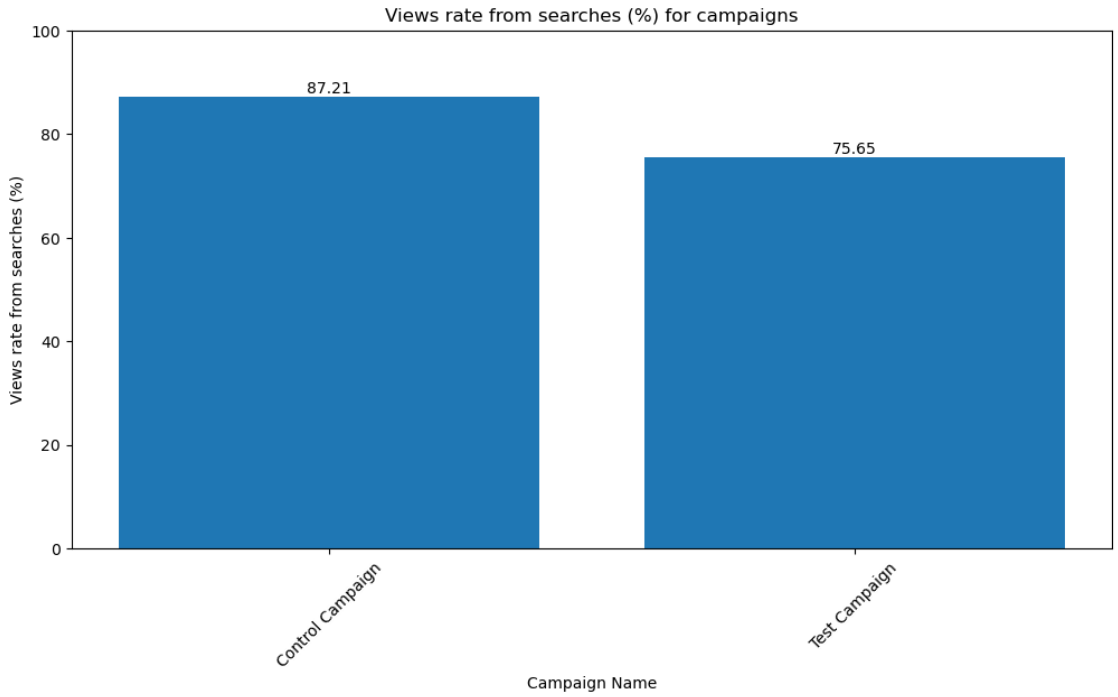
[1491]:
```
df['Searches Rate From Website clicks(%)'] = (df['# of Searches'] / df['# of␣
↪Website Clicks']) * 100
result = df.groupby('Campaign Name')['Searches Rate From Website clicks(%)'].
↪mean()
```

[1493]:
```
campaign = result.index.tolist()
rate = result.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, rate, color='#1E90FF')
plt.xlabel('Campaign Name')
plt.ylabel('Website search rate from clicks (%)')
plt.title('Website search rate from clicks (%) for campaigns')
plt.xticks(rotation=45)
plt.ylim(0, 100)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
↪ha='center', va='bottom')
plt.show()
```



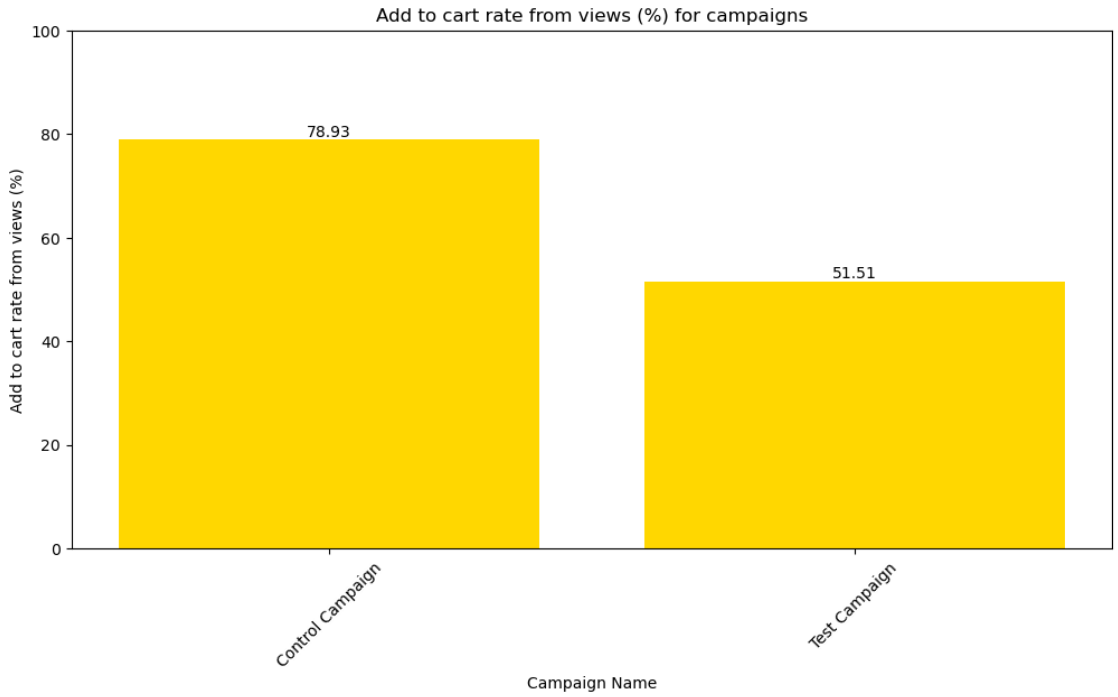
[1495]:
```
df['Views Rate From Searches(%)'] = (df['# of View Content'] / df['# of␣
 ↪Searches']) * 100
result = df.groupby('Campaign Name')['Views Rate From Searches(%)'].mean()
```

[1497]:
```
campaign = result.index.tolist()
rate = result.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, rate, color='#1f77b4')
plt.xlabel('Campaign Name')
plt.ylabel('Views rate from searches (%)')
plt.title('Views rate from searches (%) for campaigns')
plt.xticks(rotation=45)
plt.ylim(0, 100)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



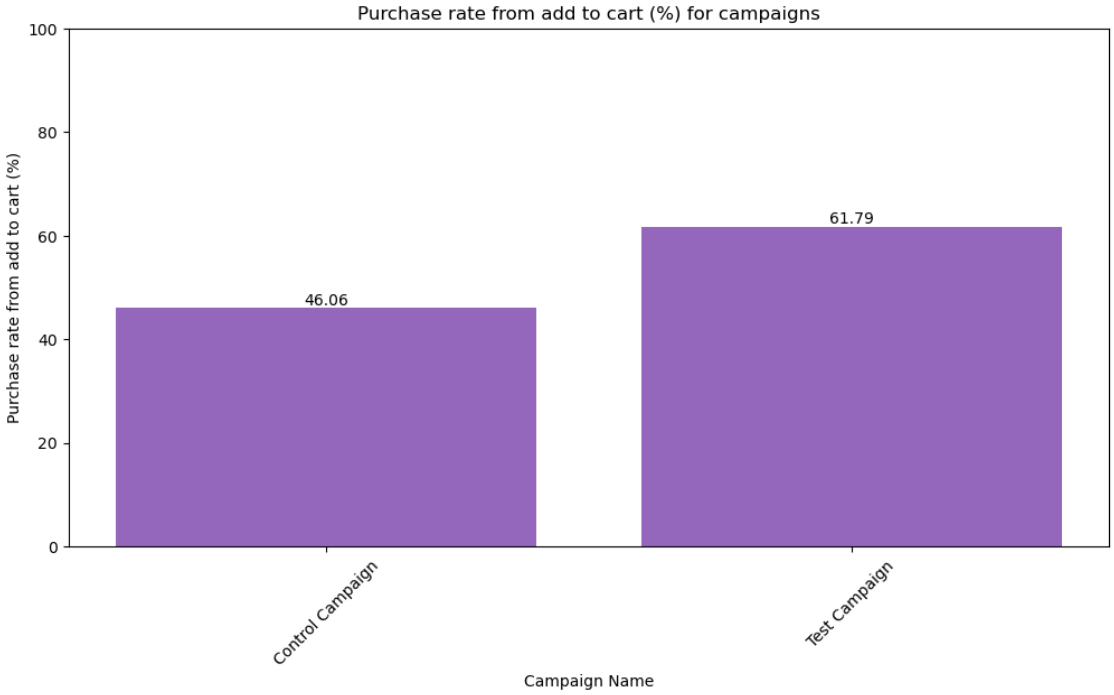
[1499]:
```
df['Add to cart Rate From views(%)'] = (df['# of Add to Cart'] / df['# of View␣
 ↪Content']) * 100
result = df.groupby('Campaign Name')['Add to cart Rate From views(%)'].mean()
```

[1501]:
```
campaign = result.index.tolist()
rate = result.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, rate, color='#FFD700')
plt.xlabel('Campaign Name')
plt.ylabel('Add to cart rate from views (%)')
plt.title('Add to cart rate from views (%) for campaigns')
plt.xticks(rotation=45)
plt.ylim(0, 100)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



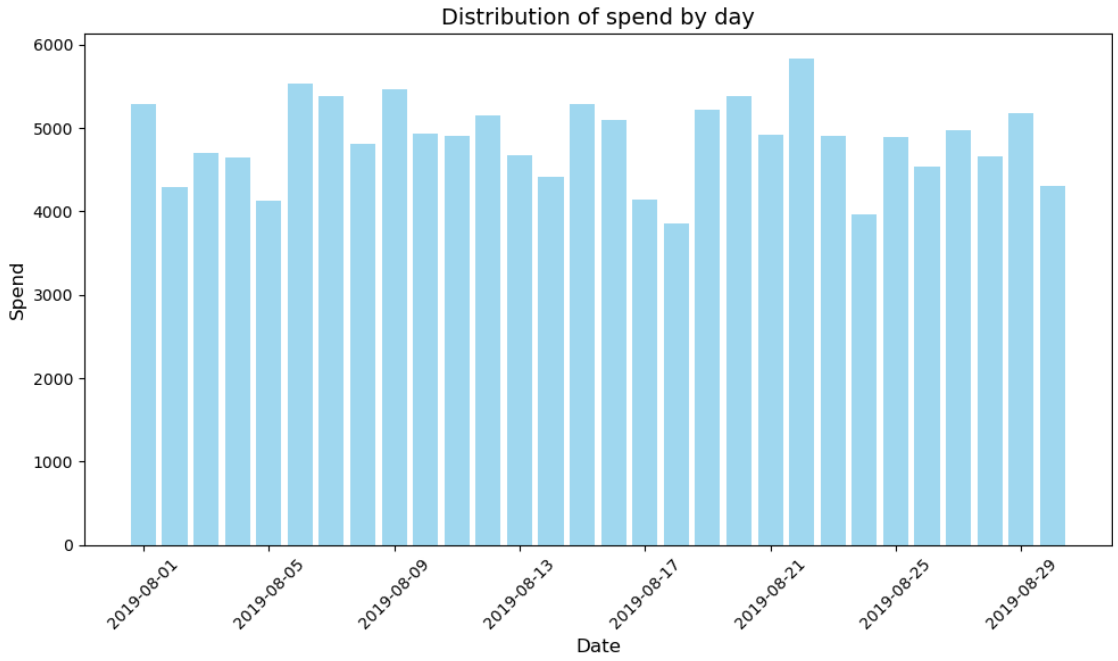
[1503]:
```
df['Purchase Rate From Add to Cart(%)'] = (df['# of Purchase'] / df['# of Add␣
 ↪to Cart']) * 100
result = df.groupby('Campaign Name')['Purchase Rate From Add to Cart(%)'].mean()
```

[1505]:
```
campaign = result.index.tolist()
rate = result.values.tolist()

plt.figure(figsize=(12, 6))
bars = plt.bar(campaign, rate, color='#9467bd')
plt.xlabel('Campaign Name')
plt.ylabel('Purchase rate from add to cart (%)')
plt.title('Purchase rate from add to cart (%) for campaigns')
plt.xticks(rotation=45)
plt.ylim(0, 100)
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2),␣
 ↪ha='center', va='bottom')
plt.show()
```



Data Distribution Over Time

[1508]:
```
daily_spend = df.groupby('Date')['Spend [USD]'].sum()

plt.figure(figsize=(10, 6))
plt.bar(daily_spend.index, daily_spend.values, color='skyblue', alpha=0.8)

plt.title('Distribution of spend by day', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Spend', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

plt.show()
```

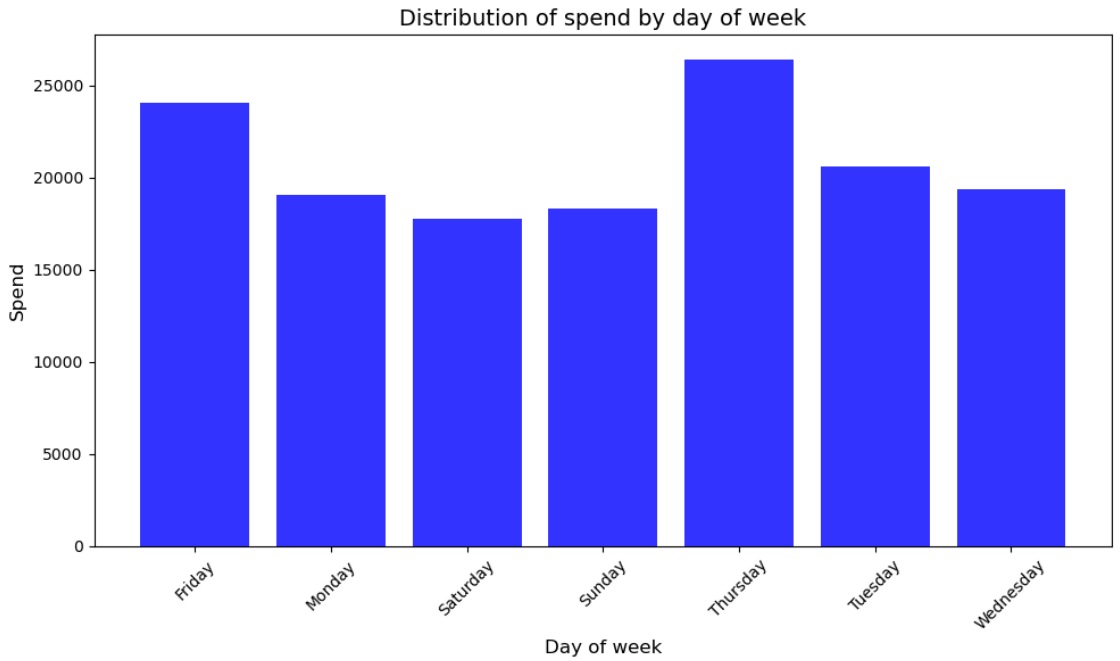


[1510]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)

indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

[1510]:
```
Index([], dtype='int64')
```

[1568]:
```
daily_spend = df.groupby('Day of week')['Spend [USD]'].sum()

plt.figure(figsize=(10, 6))
plt.bar(daily_spend.index, daily_spend.values, color='blue', alpha=0.8)

plt.title('Distribution of spend by day of week', fontsize=14)
plt.xlabel('Day of week', fontsize=12)
plt.ylabel('Spend', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
```

```
plt.tight_layout()

plt.show()
```

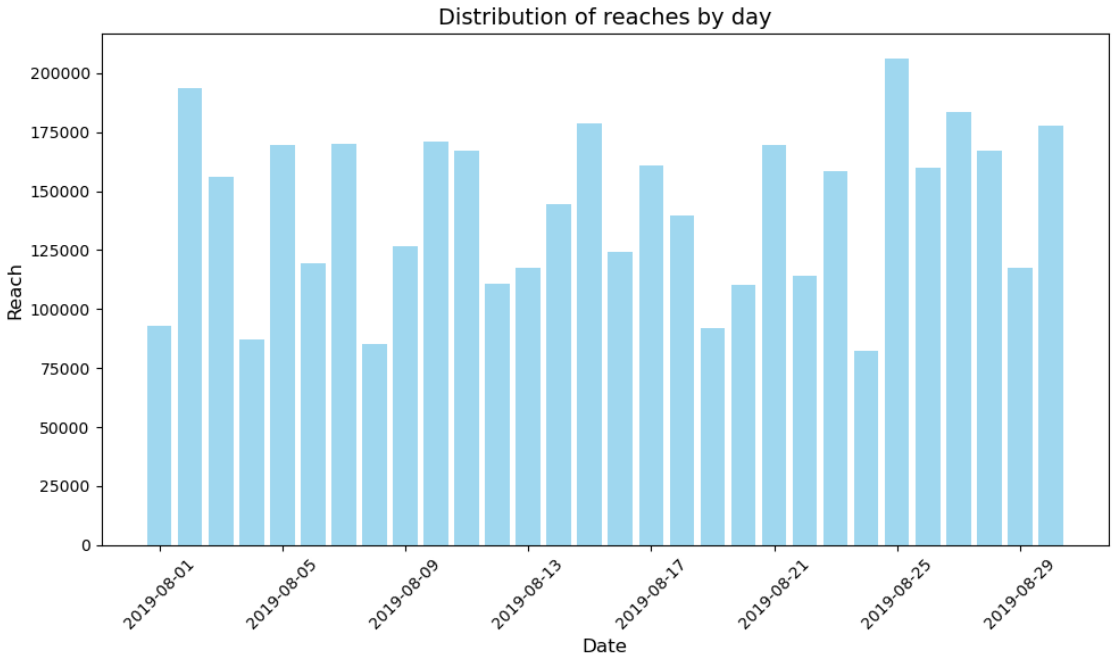


[1570]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)

indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

[1570]:
```
Index([], dtype='int64')
```

[1572]:
```
daily_reaches = df.groupby('Date')['Reach'].sum()

plt.figure(figsize=(10, 6))
plt.bar(daily_reaches.index, daily_reaches.values, color='skyblue', alpha=0.8)

plt.title('Distribution of reaches by day', fontsize=14)
```

```
plt.xlabel('Date', fontsize=12)
plt.ylabel('Reach', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

plt.show()
```

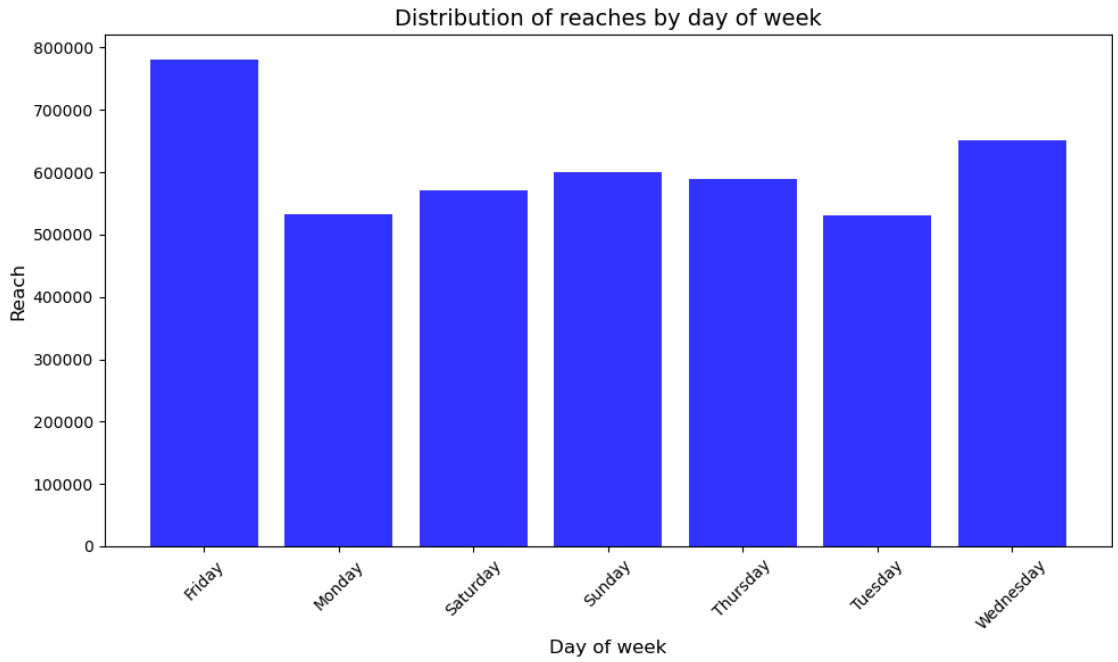


[1574]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)

indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

[1574]:
```
Index([], dtype='int64')
```

[1576]:
```
daily_reaches = df.groupby('Day of week')['Reach'].sum()
```

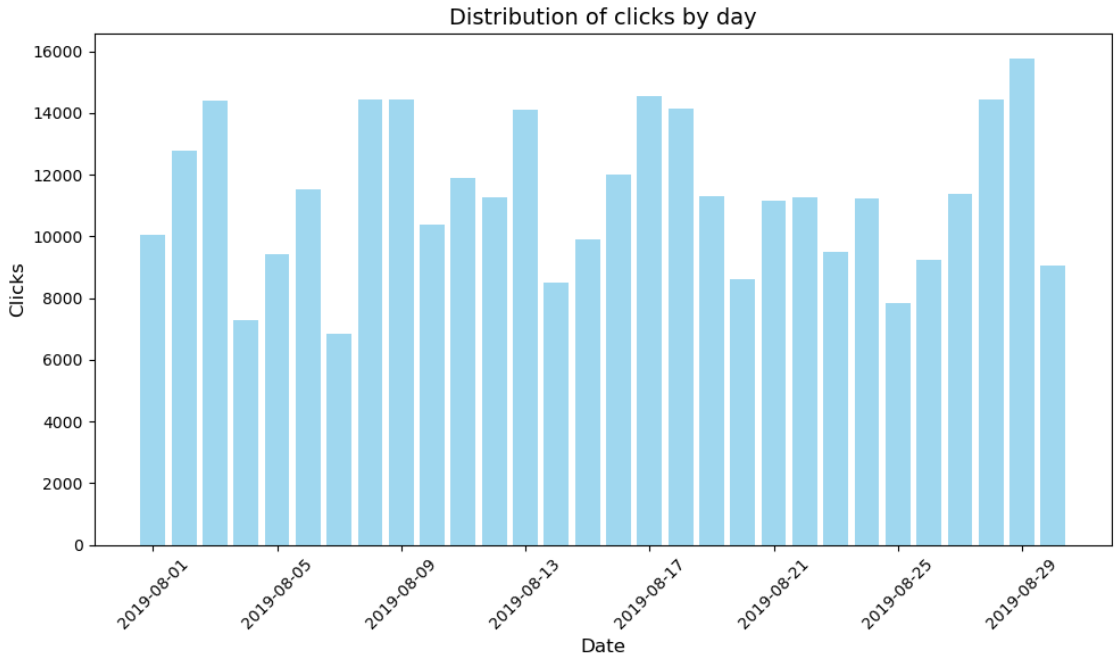
```
plt.figure(figsize=(10, 6))
plt.bar(daily_reaches.index, daily_reaches.values, color='blue', alpha=0.8)

plt.title('Distribution of reaches by day of week', fontsize=14)
plt.xlabel('Day of week', fontsize=12)
plt.ylabel('Reach', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

plt.show()
```



[1578]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)

indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

[1578]:
```
Index([], dtype='int64')
```

[1580]:
```
daily_reaches = df.groupby('Date')['# of Website Clicks'].sum()

plt.figure(figsize=(10, 6))
plt.bar(daily_reaches.index, daily_reaches.values, color='skyblue', alpha=0.8)

plt.title('Distribution of clicks by day', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Clicks', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

plt.show()
```

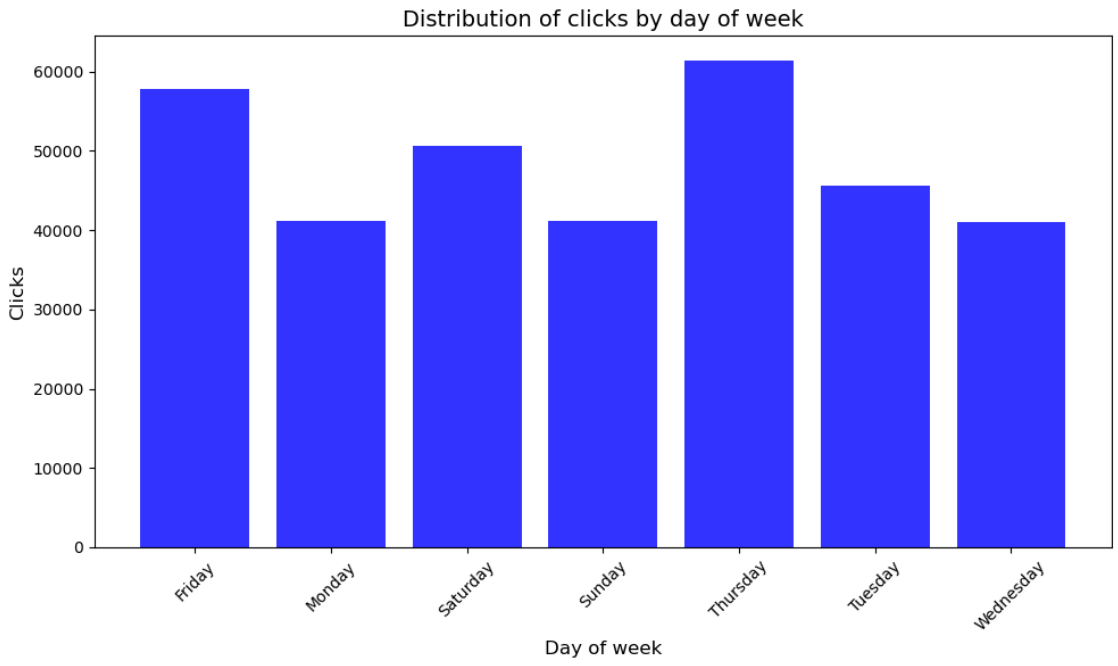


[1582]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)
```

```
indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

[1582]:
```
Index([], dtype='int64')
```

[1584]:
```
daily_reaches = df.groupby('Day of week')['# of Website Clicks'].sum()

plt.figure(figsize=(10, 6))
plt.bar(daily_reaches.index, daily_reaches.values, color='blue', alpha=0.8)

plt.title('Distribution of clicks by day of week', fontsize=14)
plt.xlabel('Day of week', fontsize=12)
plt.ylabel('Clicks', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

plt.show()
```

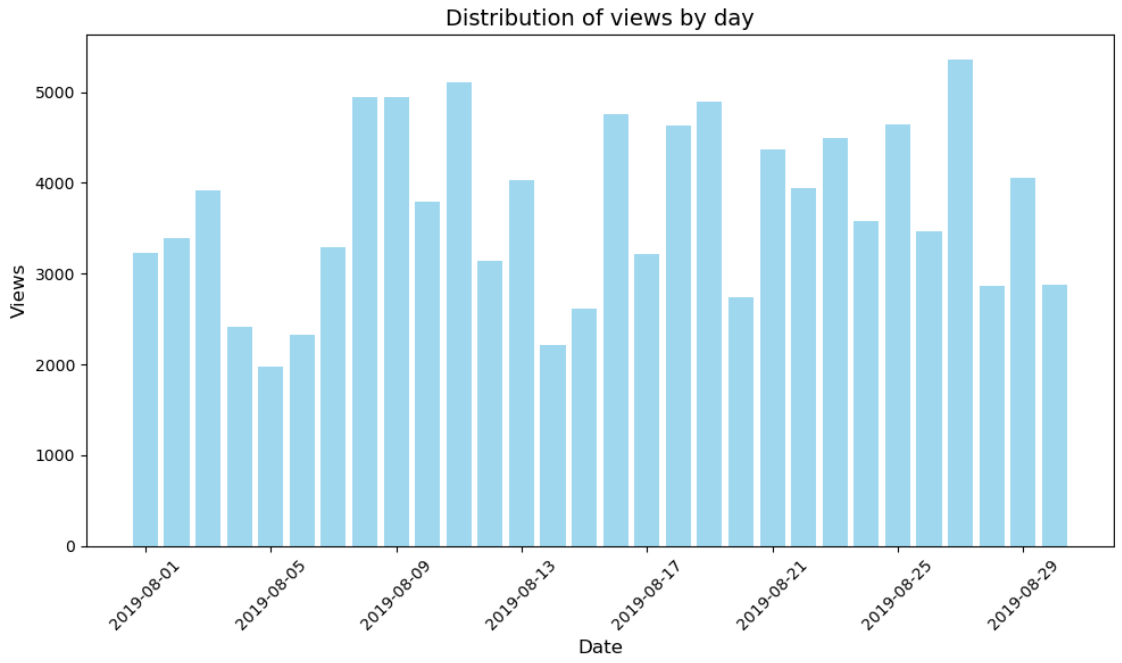


[1586]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
```

```
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)

indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

[1586]:
```
Index([], dtype='int64')
```

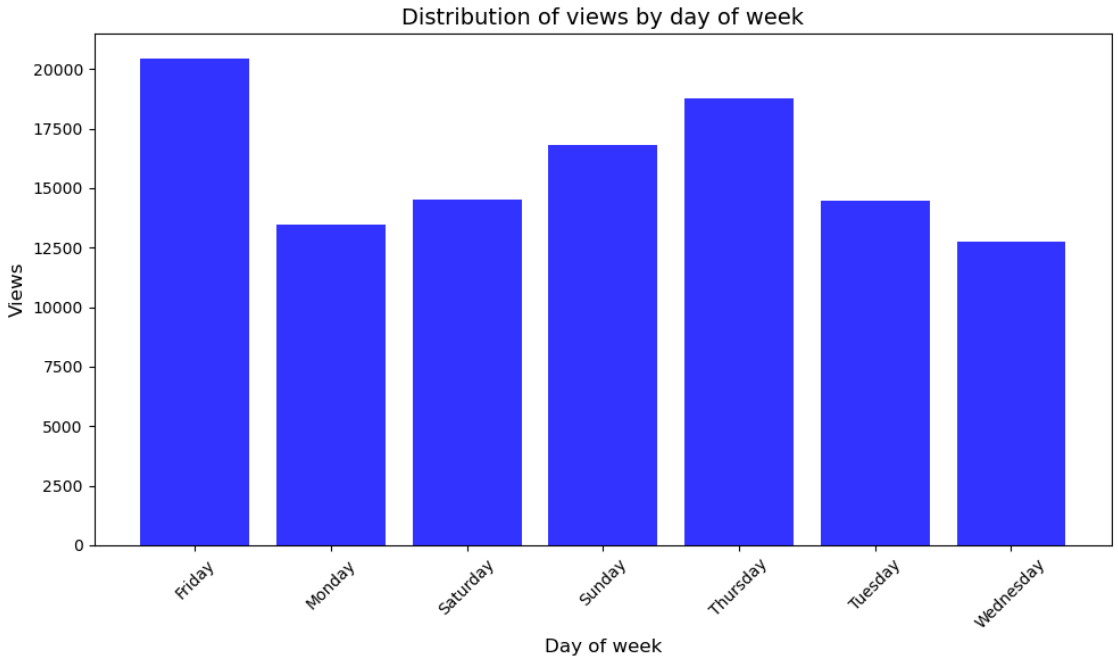
[1588]:
```
daily_reaches = df.groupby('Date')['# of View Content'].sum()

plt.figure(figsize=(10, 6))
plt.bar(daily_reaches.index, daily_reaches.values, color='skyblue', alpha=0.8)

plt.title('Distribution of views by day', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Views', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

plt.show()
```



[1590]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)

indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

[1590]:
```
Index([], dtype='int64')
```

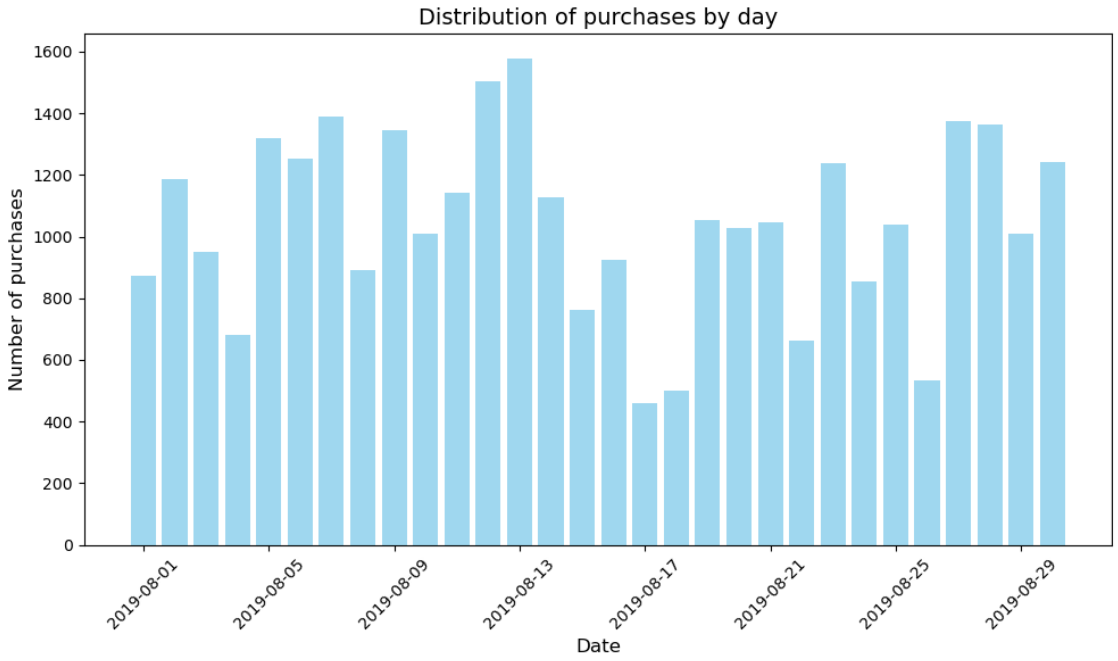
[1592]:
```
daily_reaches = df.groupby('Day of week')['# of View Content'].sum()
plt.figure(figsize=(10, 6))
plt.bar(daily_reaches.index, daily_reaches.values, color='blue', alpha=0.8)

plt.title('Distribution of views by day of week', fontsize=14)
plt.xlabel('Day of week', fontsize=12)
plt.ylabel('Views', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

plt.show()
```



[1594]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)

indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

[1594]:
```
Index([], dtype='int64')
```

[1596]:
```
daily_purchases = df.groupby('Date')['# of Purchase'].sum()

plt.figure(figsize=(10, 6))
plt.bar(daily_purchases.index, daily_purchases.values, color='skyblue', alpha=0.
 ↪8)

plt.title('Distribution of purchases by day', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Number of purchases', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()

plt.show()
```

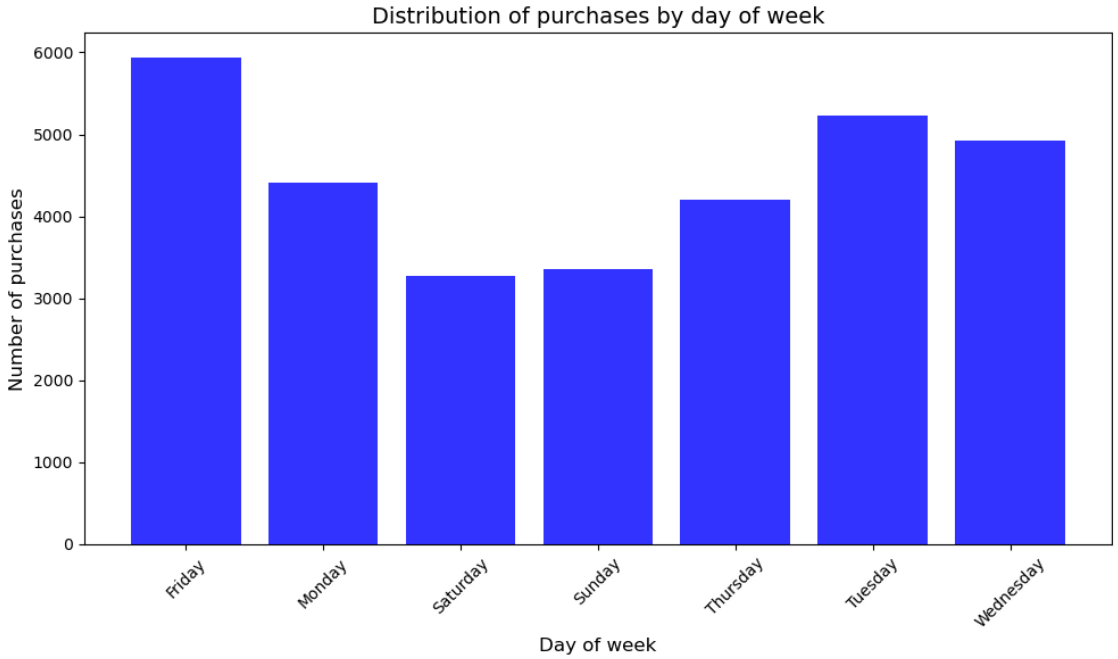


[1598]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)

indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

[1598]:
```
Index([], dtype='int64')
```

[1602]:
```
daily_purchases = df.groupby('Day of week')['# of Purchase'].sum()

plt.figure(figsize=(10, 6))
plt.bar(daily_purchases.index, daily_purchases.values, color='blue', alpha=0.8)

plt.title('Distribution of purchases by day of week', fontsize=14)
plt.xlabel('Day of week', fontsize=12)
plt.ylabel('Number of purchases', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
```

```
plt.tight_layout()

plt.show()
```



[1604]:
```
mean = daily_purchases.mean()
std_dev = daily_purchases.std()

z_score_list = []
for conversion in daily_purchases:
    z_score = (conversion - mean) / std_dev
    z_score_list.append(z_score)

z_score_series = pd.Series(z_score_list)

indexes = z_score_series[(z_score_series > 2) | (z_score_series < -2)].index
indexes
```

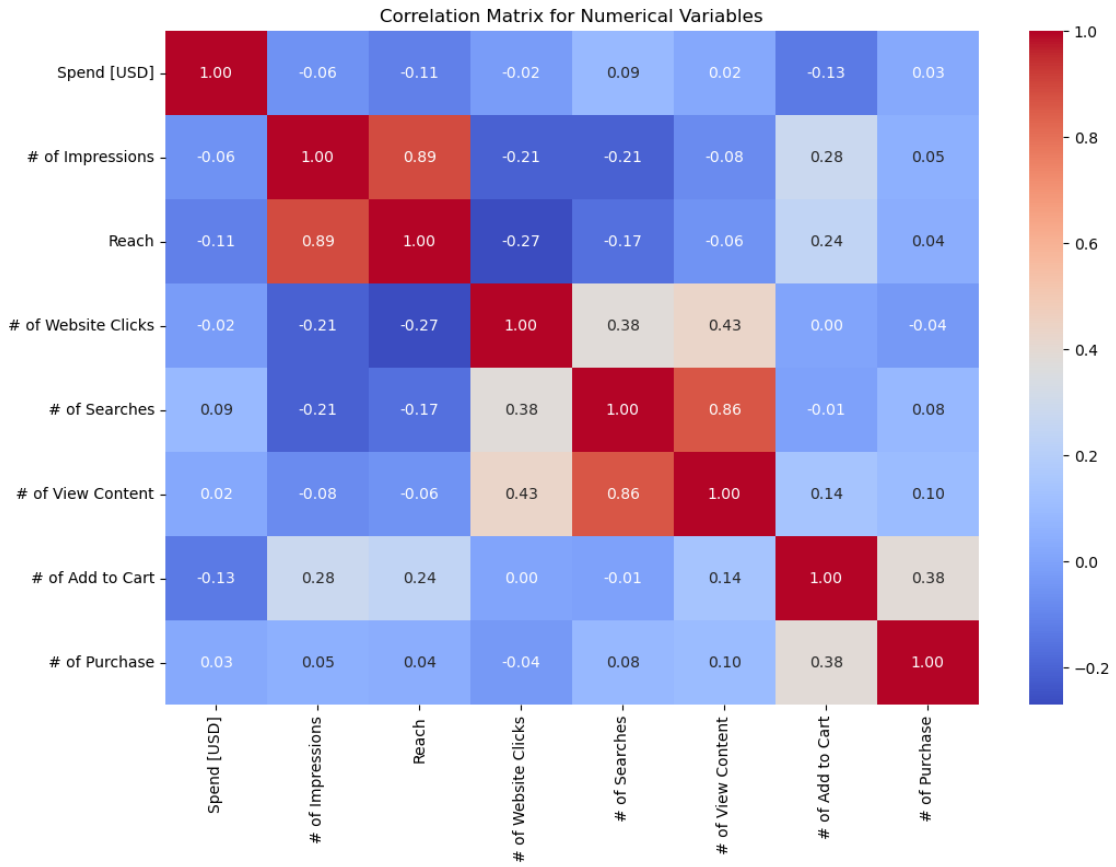
[1604]:
```
Index([], dtype='int64')
```

Thus, there are no anomalies in the data distribution over time.

Relationship Between Variables

[1608]:
```
numeric_df = df.select_dtypes(include=['number'])
numeric_df = numeric_df.iloc[:, :-7]
correlation_matrix = numeric_df.corr()
```

[1610]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Matrix for Numerical Variables")
plt.show()
```



Correlation Analysis:

Strong Positive Correlation: Ad Impressions Unique Ad Impressions (0.89): A very strong relationship. This is expected since unique impressions are a subset of all impressions. Product Searches Number of Product Views (0.86): A strong positive correlation. Users who search for products often view their details or listings. Moderate Positive Correlation: Add to Cart Purchases (0.38): A moderate relationship. Not all cart additions lead to purchases, indicating the need for an analysis of the cart stage (e.g., issues in the checkout process). Clicks Product Views (0.43): A moderate correlation. Users who click through to the site often proceed to interact with and view products. Clicks Product Searches (0.38): A moderate correlation, highlighting that not all users clicking on the site proceed to search for products.

Statistical Analysis

Using Mann-Whitney U Test

[1618]:
```
control_campaign = df[df['Campaign Name'] == 'Control Campaign']['Conversion␣
↪Rate From clicks(%)'].values
test_campaign = df[df['Campaign Name'] == 'Test Campaign']['Conversion Rate␣
↪From clicks(%)'].values

stat, p = mannwhitneyu(control_campaign, test_campaign)
alpha = 0.05
p > alpha
```

[1618]:
```
True
```

Result

No statistically significant difference in conversion rates was found for different campaigns, as the p-value is higher than the chosen threshold for Type I error.

Calculate the Power

[1624]:
```
mean_a = np.median(control_campaign)
mean_b = np.median(test_campaign)
std_a = control_campaign.std()
std_b = test_campaign.std()
nobs1 = len(control_campaign)
nobs2 = len(test_campaign)
alpha = 0.05
power = 0.8

pooled_std = np.sqrt(((std_a**2) + (std_b**2)) / 2)
effect_size = (mean_a - mean_b) / pooled_std

power_analysis = TTestIndPower()
current_power = power_analysis.solve_power(effect_size=effect_size,␣
↪nobs1=nobs1, alpha=alpha, ratio=nobs2/nobs1, alternative='two-sided')
print(f"Current Sample Power: {current_power:.2f}")
print(f"Effect size: {effect_size:.2f}")
```

```
Current Sample Power: 0.32
Effect size: 0.39
```

The power is very low, with a 68% probability that we will not detect statistically significant differences where they exist.

[1627]:
```
sample_size = zt_ind_solve_power(effect_size=effect_size, alpha=alpha,␣
↪power=power, ratio=1)
print(f"Required Sample Size: {sample_size:.0f} for one campaign")
```

```
Required Sample Size: 102 for one campaign
```

Solution

Continue collecting data for 62 days and conclude the experiment thereafter.

Intermediate Results

Costs and Efficiency:

Average and Median Costs: The test campaign shows higher average (+275 USD) and median (+285 USD) costs. The total cost overrun of the test campaign amounts to 76,892 - 68,653 = 8,249 USD. Cost per purchase in the test campaign is higher: 4.92 compared to 4.37 for the control campaign, reducing its economic efficiency. Purchases and Conversions: The control campaign recorded 76 more purchases (15,713 versus 15,637). Conversion rate is higher for the control campaign: 11.61% compared to 9.63% for the test campaign.

Advertising Metrics:

Ad Impressions: The control campaign significantly outperforms the test campaign in terms of ad impressions (108,940 versus 74,584). Unique Impressions: The control campaign also leads with 88,370 compared to 53,491. Average Clicks: The test campaign generates more website clicks: 6,032 versus 5,261, indicating higher ad interaction.

Conversions from Ad Impressions:

The control campaign's click rate is 5.06%, whereas the test campaign achieves 10.24%. This highlights the test campaign's strong ability to attract interest to the website.

Conversion Funnel:

Transitions from Clicks: Product searches from website clicks: the test campaign shows a slightly better result (43.74% versus 42.83%). Product Views: The control campaign demonstrates a higher percentage of product views (87.21% versus 75.65%). Add to Cart: The control campaign outperforms the test campaign—78.93% versus 51.51%. Purchases from Cart Additions: The test campaign performs significantly better here: 61.79% versus 46.06%.

Recommendations:

Funnel Stage Optimization: Investigate issues at the add-to-cart stage in the test campaign: users might face an inconvenient interface, unclear pricing, or a lack of promotions. Maintain high conversion rates at the final stages (purchases from the cart) by enhancing incentives (e.g., discounts, free shipping). Audience Quality: Assess the audience of the test campaign: lower unique impressions and narrower reach may suggest less effective targeting. Consider reallocating advertising budgets to broaden or fine-tune audience targeting. Economic Efficiency: Compare the Customer Acquisition Cost (CAC) with the potential Lifetime Value (LTV) for both campaigns to evaluate long-term benefits. Conduct A/B testing of different creatives or offers to reduce the cost per purchase in the test campaign. Continuation of Data Collection: Increase the data volume to achieve statistically significant results, especially for metrics with small differences. Ad Analysis: Evaluate the effectiveness of the test campaign creatives, as the high click rate indicates their appeal. They might be adapted to improve the control campaign results.

Brief Conclusion:

The test campaign shows strong activity at the early stages of the funnel (higher click rate), but it encounters challenges in the later stages, particularly at the add-to-cart stage, where conversions are significantly lower compared to the control campaign. Despite a higher number of clicks, the cost per purchase is higher, and overall profitability is lower. The control campaign is more stable, with better overall efficiency, higher conversion rates, and well-managed expenses. It is recommended to

focus on optimizing the add-to-cart stage, improving targeting in the test campaign, and continuing data collection to enhance statistical significance. This will support making well-informed decisions.