Marketing Campaign Testing in Python

```
[1] import pandas as pd
    import datetime
    from datetime import date, timedelta
    import plotly.graph_objects as go
    import plotly.express as px
    import plotly.io as pio
```

```
[2] control_df = pd.read_csv('/content/drive/MyDrive/control_group.csv', sep = ';')
    test_df = pd.read_csv('/content/drive/MyDrive/test_group.csv', sep = ';')
```

```
[3] control_df.head()
```

	Campaign Name	Date	Spend [USD]	# of Impressions	Reach	# of Website Clicks	# of Searches	# of View Content	# of Add to Cart	# of Purchase
0	Control Campaign	1.08.2019	2280	82702.0	56930.0	7016.0	2290.0	2159.0	1819.0	618.0
1	Control Campaign	2.08.2019	1757	121040.0	102513.0	8110.0	2033.0	1841.0	1219.0	511.0
2	Control Campaign	3.08.2019	2343	131711.0	110862.0	6508.0	1737.0	1549.0	1134.0	372.0
3	Control Campaign	4.08.2019	1940	72878.0	61235.0	3065.0	1042.0	982.0	1183.0	340.0
4	Control Campaign	5.08.2019	1835	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
[4] test_df.head()
```

	Campaign Name	Date	Spend [USD]	# of Impressions	Reach	# of Website Clicks	# of Searches	# of View Content	# of Add to Cart	# of Purchase
0	Test Campaign	1.08.2019	3008	39550	35820	3038	1946	1069	894	255
1	Test Campaign	2.08.2019	2542	100719	91236	4657	2359	1548	879	677
2	Test Campaign	3.08.2019	2365	70263	45198	7885	2572	2367	1268	578
3	Test Campaign	4.08.2019	2710	78451	25937	4216	2216	1437	566	340
4	Test Campaign	5.08.2019	2297	114295	95138	5863	2106	858	956	768

```
[5]  control_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Campaign Name         30 non-null     object 
 1   Date                  30 non-null     object 
 2   Spend [USD]           30 non-null     int64  
 3   # of Impressions      29 non-null     float64
 4   Reach                 29 non-null     float64
 5   # of Website Clicks   29 non-null     float64
 6   # of Searches         29 non-null     float64
 7   # of View Content     29 non-null     float64
 8   # of Add to Cart      29 non-null     float64
 9   # of Purchase         29 non-null     float64
dtypes: float64(7), int64(1), object(2)
memory usage: 2.5+ KB
```

```
[6]  control_df.isna().sum() #check for nulls
```

	0
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

dtype: int64

```
[7]  test_df.isna().sum() #check for nulls
```

	0
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

dtype: int64

```
[8]  # Replace null with mean value
     control_df["# of Impressions"].fillna(value=control_df["# of Impressions"].mean(),
                                           inplace=True)
     control_df["Reach"].fillna(value=control_df["Reach"].mean(),
                                inplace=True)
     control_df["# of Website Clicks"].fillna(value=control_df["# of Website Clicks"].mean(),
                                         inplace=True)
     control_df["# of Searches"].fillna(value=control_df["# of Searches"].mean(),
                                        inplace=True)
     control_df["# of View Content"].fillna(value=control_df["# of View Content"].mean(),
                                         inplace=True)
     control_df["# of Add to Cart"].fillna(value=control_df["# of Add to Cart"].mean(),
                                        inplace=True)
     control_df["# of Purchase"].fillna(value=control_df["# of Purchase"].mean(),
                                     inplace=True)
```

```
[9]  control_df.isna().sum() # Recheck for nulls
```

	0
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

dtype: int64

```
[10] # Concating both datasets with an outer join on Date

     # Adding campaign labels
     control_df["Campaign Name"] = "Control Campaign"
     test_df["Campaign Name"] = "Test Campaign"

     # Selecting and ordering columns properly
     common_columns = ["Date", "Campaign Name"] + [col for col in control_df.columns if col not in ["Date", "Campaign Name"]]

     # Concatenating both datasets in a stacked format
     merged_df = pd.concat([control_df[common_columns], test_df[common_columns]])

     # Sorting by Date and Campaign Name
     merged_df = merged_df.sort_values(["Date", "Campaign Name"]).reset_index(drop=True)


     merged_df.head()
```

	Date	Campaign Name	Spend [USD]	# of Impressions	Reach	# of Website Clicks	# of Searches	# of View Content	# of Add to Cart	# of Purchase
0	1.08.2019	Control Campaign	2280	82702.0	56930.0	7016.0	2290.0	2159.0	1819.0	618.0
1	1.08.2019	Test Campaign	3008	39550.0	35820.0	3038.0	1946.0	1069.0	894.0	255.0
2	10.08.2019	Control Campaign	2149	117624.0	91257.0	2277.0	2475.0	1984.0	1629.0	734.0
3	10.08.2019	Test Campaign	2790	95054.0	79632.0	8125.0	2312.0	1804.0	424.0	275.0
4	11.08.2019	Control Campaign	2490	115247.0	95843.0	8137.0	2941.0	2486.0	1887.0	475.0

```
[11] merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Date                  60 non-null     object 
 1   Campaign Name         60 non-null     object 
 2   Spend [USD]           60 non-null     int64  
 3   # of Impressions      60 non-null     float64
 4   Reach                 60 non-null     float64
 5   # of Website Clicks   60 non-null     float64
 6   # of Searches         60 non-null     float64
 7   # of View Content     60 non-null     float64
 8   # of Add to Cart      60 non-null     float64
 9   # of Purchase         60 non-null     float64
dtypes: float64(7), int64(1), object(2)
memory usage: 4.8+ KB
```

```
[12] # Renaming columns for readability
     merged_df = merged_df.rename(columns={
         "Spend [USD]": "Amount Spent",
         "# of Impressions": "Number of Impressions",
         "Reach": "Reach",
         "# of Website Clicks": "Website Clicks",
         "# of Searches": "Searches Received",
         "# of View Content": "Content Viewed",
         "# of Add to Cart": "Added to Cart",
         "# of Purchase": "Purchases"
     })
```

```
[13] merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Date                   60 non-null     object 
 1   Campaign Name          60 non-null     object 
 2   Amount Spent           60 non-null     int64  
 3   Number of Impressions  60 non-null     float64
 4   Reach                  60 non-null     float64
 5   Website Clicks         60 non-null     float64
 6   Searches Received      60 non-null     float64
 7   Content Viewed         60 non-null     float64
 8   Added to Cart          60 non-null     float64
 9   Purchases              60 non-null     float64
dtypes: float64(7), int64(1), object(2)
memory usage: 4.8+ KB
```

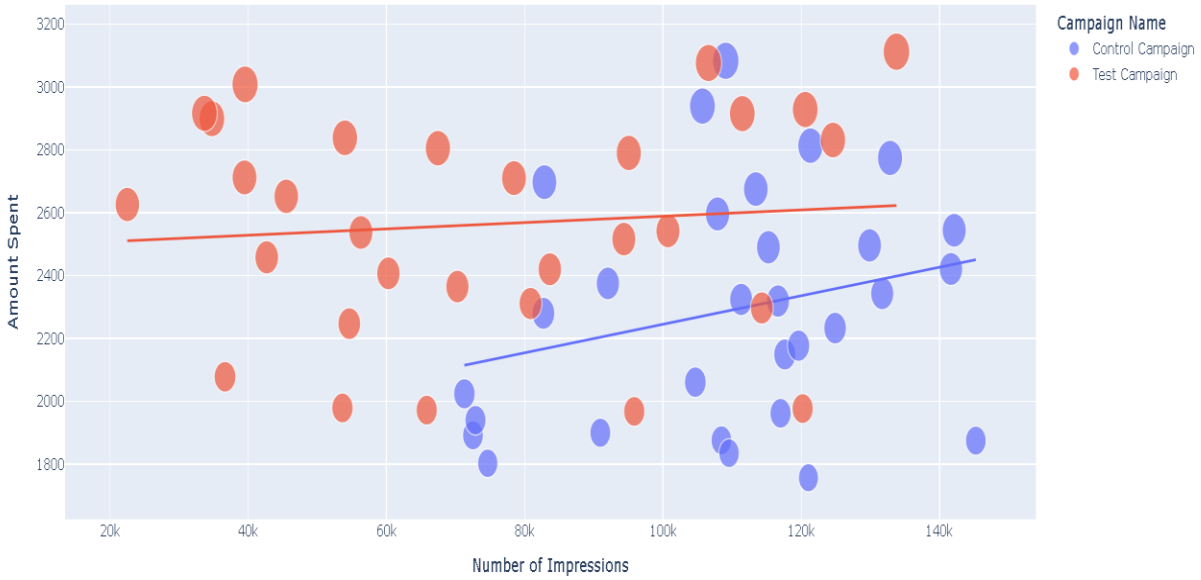
```
[14] # Check if the dataset has an equal number of samples about both campaigns
     merged_df["Campaign Name"].value_counts()
```

Campaign Name	count
Control Campaign	30
Test Campaign	30

dtype: int64

As displayed in output, the dataset has 30 samples for each campaign.

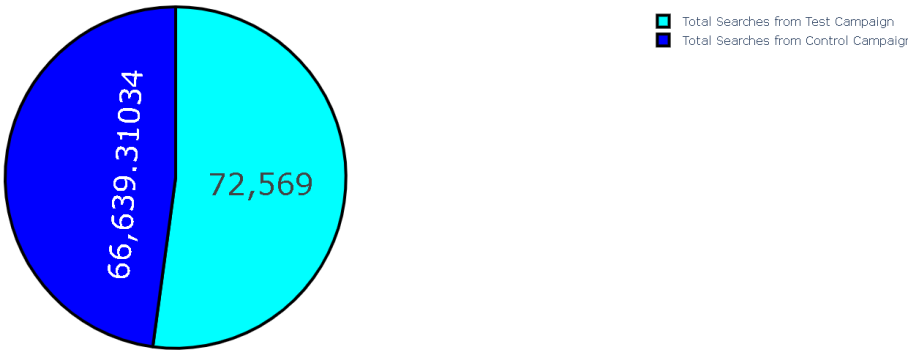
```
[15] # Analyze the relationship between the number of impressions we got from both campaigns and the amount spent on both campaigns.
     fig = px.scatter(data_frame = merged_df,
                      x="Number of Impressions",
                      y="Amount Spent",
                      size="Amount Spent",
                      color= "Campaign Name",
                      trendline="ols")
     fig.show()
```



The control campaign resulted in more impressions according to the amount spent on both campaigns.

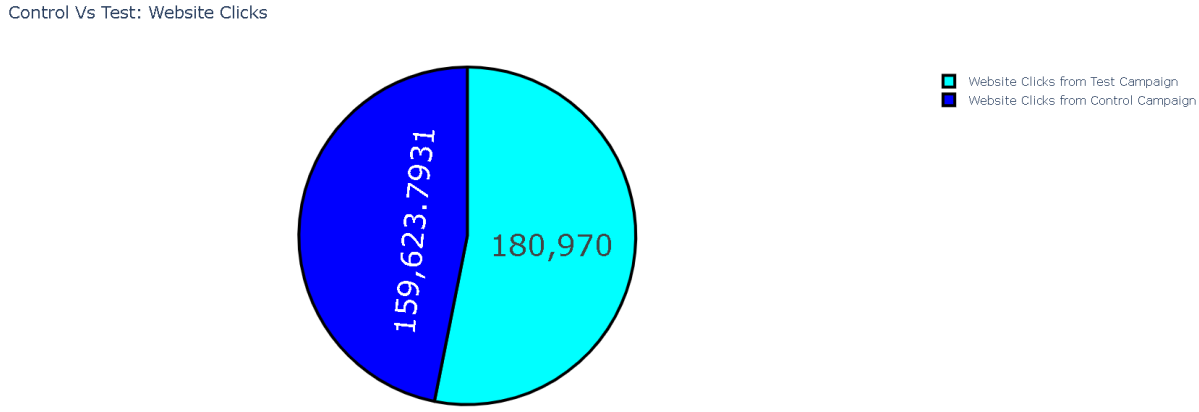
```
[16] # Check the number of searches performed on the website from both campaigns
     label = ["Total Searches from Control Campaign",
              "Total Searches from Test Campaign"]
     counts = [sum(control_df["# of Searches"]),
               sum(test_df["# of Searches"])]
     colors = ['blue','aqua']
     fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
     fig.update_layout(title_text='Control Vs Test: Searches')
     fig.update_traces(hoverinfo='label+percent', textinfo='value',
                       textfont_size=30,marker=dict(colors=colors,
                       line=dict(color='black', width=3)))
     fig.show()
```

Control Vs Test: Searches



The test campaign is searched more than control campaign on the website.

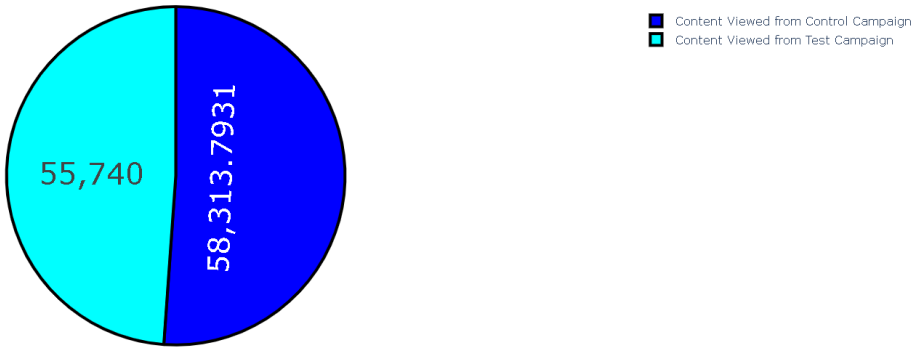
```
[17] # Check the number of website clicks from both campaigns
     label = ["Website Clicks from Control Campaign",
              "Website Clicks from Test Campaign"]
     counts = [sum(control_df["# of Website Clicks"]),
               sum(test_df["# of Website Clicks"])]
     colors = ['blue','aqua']
     fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
     fig.update_layout(title_text='Control Vs Test: Website Clicks')
     fig.update_traces(hoverinfo='label+percent', textinfo='value',
                       textfont_size=30,
                       marker=dict(colors=colors,
                                   line=dict(color='black', width=3)))
     fig.show()
```



Again, the test campaign is searched more than control campaign on the website.

```
[18] # Check the amount of content viewed after reaching the website from both campaigns
     label = ["Content Viewed from Control Campaign",
              "Content Viewed from Test Campaign"]
     counts = [sum(control_df["# of View Content"]),
               sum(test_df["# of View Content"])]
     colors = ['blue','aqua']
     fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
     fig.update_layout(title_text='Control Vs Test: Content Viewed')
     fig.update_traces(hoverinfo='label+percent', textinfo='value',
                       textfont_size=30,
                       marker=dict(colors=colors,
                                   line=dict(color='black', width=3)))
     fig.show()
```

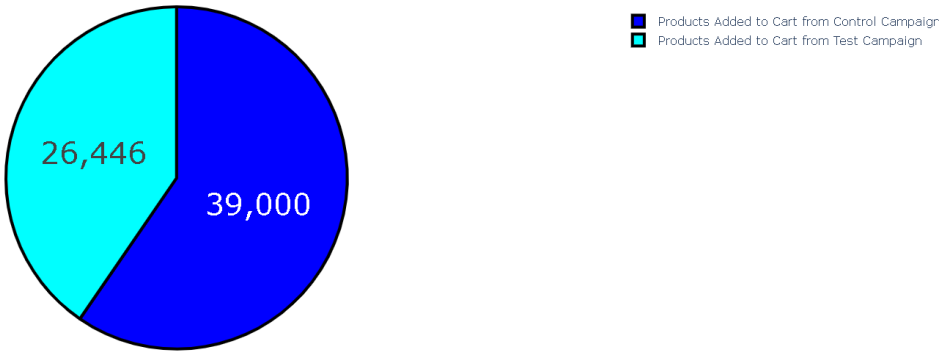
Control Vs Test: Content Viewed



The audience of the control campaign viewed more content than the test campaign. Although there is not much difference, as the website clicks of the control campaign were low, its engagement on the website is higher than the test campaign.

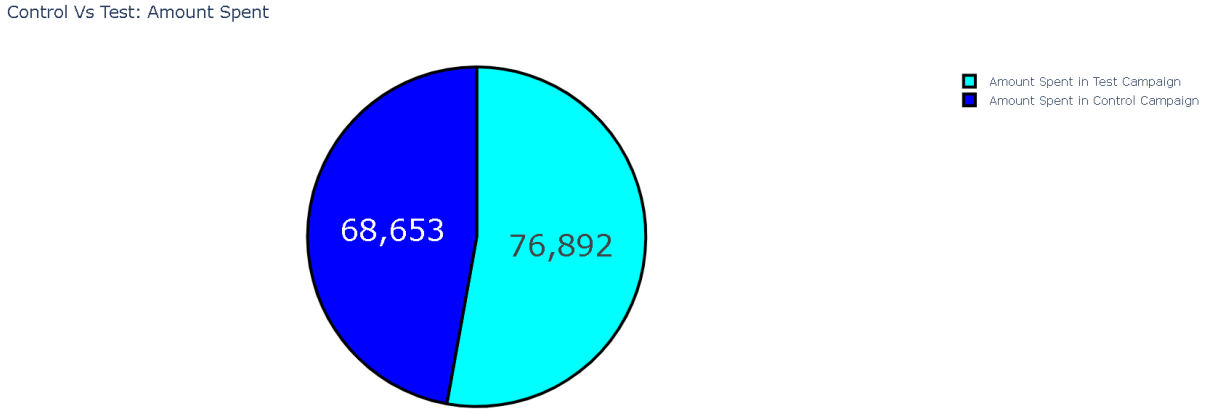
```
[19] # Check the number of products added to the cart from both campaigns
     label = ["Products Added to Cart from Control Campaign",
              "Products Added to Cart from Test Campaign"]
     counts = [sum(control_df["# of Add to Cart"]),
               sum(test_df["# of Add to Cart"])]
     colors = ['blue','aqua']
     fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
     fig.update_layout(title_text='Control Vs Test: Added to Cart')
     fig.update_traces(hoverinfo='label+percent', textinfo='value',
                       textfont_size=30,
                       marker=dict(colors=colors,
                                   line=dict(color='black', width=3)))
     fig.show()
```

Control Vs Test: Added to Cart



Despite low website clicks more products were added to the cart from the control campaign.

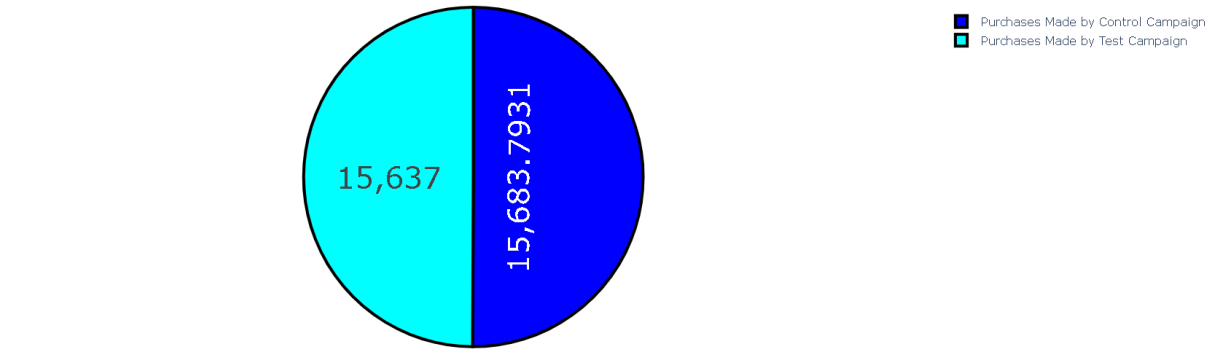
```
[20] # Check the amount spent on both campaigns
     label = ["Amount Spent in Control Campaign",
              "Amount Spent in Test Campaign"]
     counts = [sum(control_df["Spend [USD]"]),
               sum(test_df["Spend [USD]"])]
     colors = ['blue','aqua']
     fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
     fig.update_layout(title_text='Control Vs Test: Amount Spent')
     fig.update_traces(hoverinfo='label+percent', textinfo='value',
                       textfont_size=30,
                       marker=dict(colors=colors,
                                   line=dict(color='black', width=3)))
     fig.show()
```



The amount spent on the test campaign is higher than the control campaign. But as we can see that the control campaign resulted in more content views and more products in the cart, the control campaign is more efficient than the test campaign.

```
[21] # Check the purchases made by both campaigns
     label = ["Purchases Made by Control Campaign",
              "Purchases Made by Test Campaign"]
     counts = [sum(control_df["# of Purchase"]),
               sum(test_df["# of Purchase"])]
     colors = ['blue','aqua']
     fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
     fig.update_layout(title_text='Control Vs Test: Purchases')
     fig.update_traces(hoverinfo='label+percent', textinfo='value',
                       textfont_size=30,
                       marker=dict(colors=colors,
                                   line=dict(color='black', width=3)))
     fig.show()
```

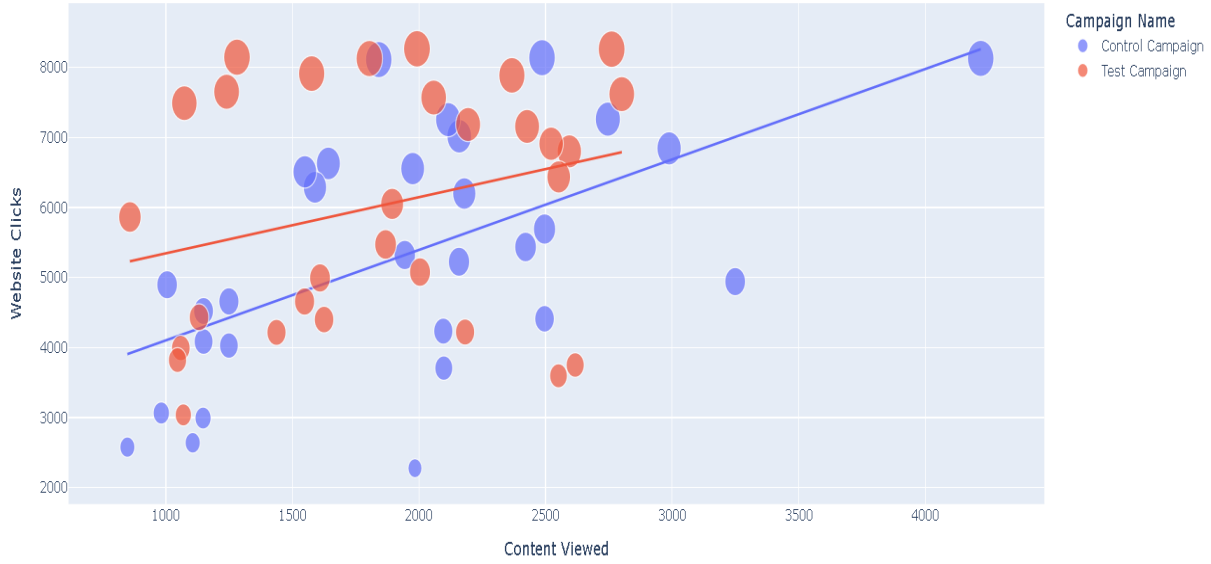
Control Vs Test: Purchases



There's only a difference of around 1% in the purchases made from both ad campaigns. As the Control campaign resulted in more sales in less amount spent on marketing, the control campaign wins here.

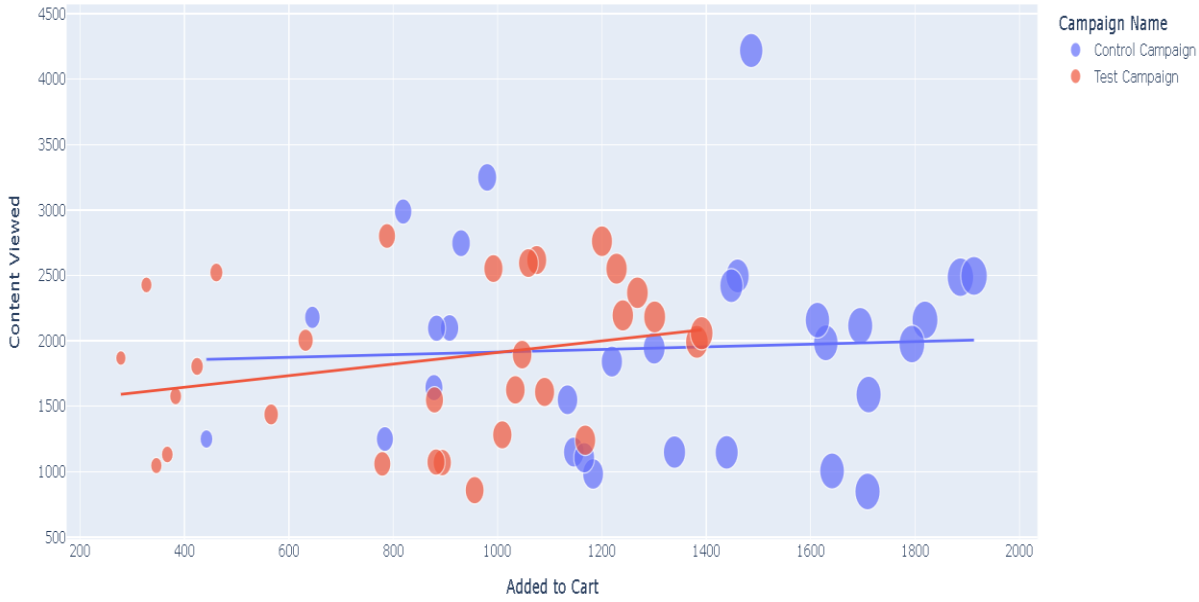
Now let's analyze some metrics to find which ad campaign converts more. First look at the relationship between the number of website clicks and content viewed from both campaigns.

```
[22] # Check the relationship between the number of website clicks and content viewed from both campaigns.
     fig = px.scatter(data_frame = merged_df,
                      x="Content Viewed",
                      y="Website Clicks",
                      size="Website Clicks",
                      color= "Campaign Name",
                      trendline="ols")
     fig.show()
```



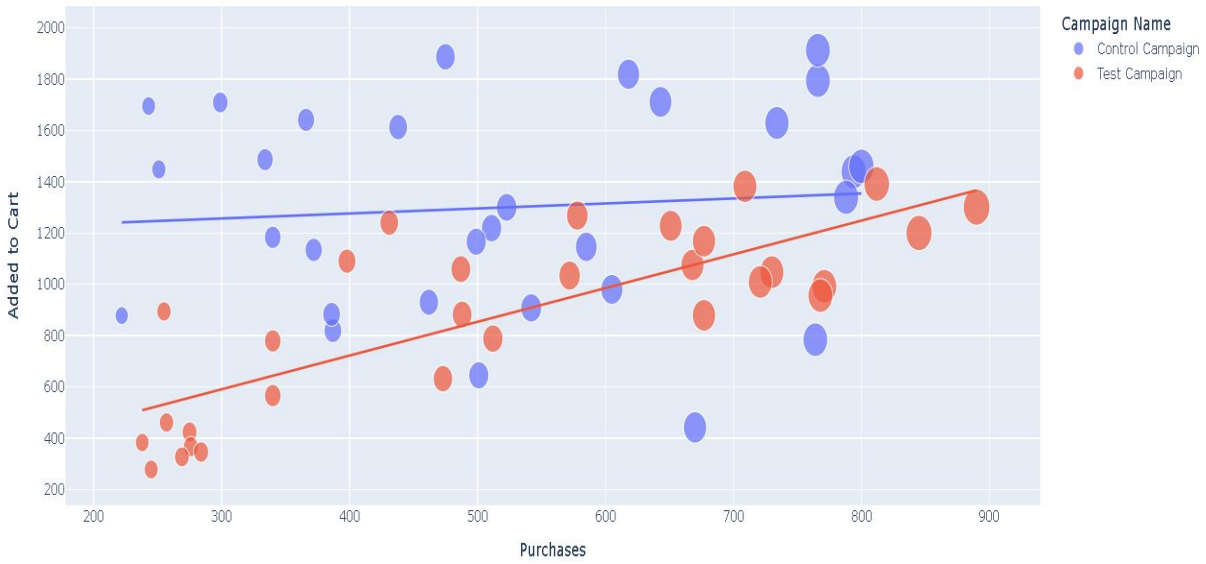
The website clicks are higher in the test campaign, but the engagement from website clicks is higher in the control campaign. So the control campaign is better.

```
[23] # Analyze the relationship between the amount of content viewed and the number of products added to the cart from both campaigns
     fig = px.scatter(data_frame = merged_df,
                      x="Added to Cart",
                      y="Content Viewed",
                      size="Added to Cart",
                      color= "Campaign Name",
                      trendline="ols")
     fig.show()
```



Again, the control campaign performs better than test campaign.

```
[24] # Check the relationship between the number of products added to the cart and the number of sales from both campaigns
     fig = px.scatter(data_frame = merged_df,
                      x="Purchases",
                      y="Added to Cart",
                      size="Purchases",
                      color= "Campaign Name",
                      trendline="ols")
     fig.show()
```



Although the control campaign resulted in more sales and more products in the cart, the conversation rate of the test campaign is higher.

Conclusion From the above A/B tests, we found that the control campaign resulted in more sales and engagement from the visitors. More products were viewed from the control campaign, resulting in more products in the cart and more sales. But the conversation rate of products in the cart is higher in the test campaign. The test campaign resulted in more sales according to the products viewed and added to the cart. And the control campaign results in more sales overall. So, the Test campaign can be used to market a specific product to a specific audience, and the Control campaign can be used to market multiple products to a wider audience.