A/B Testing Analysis on

Marketing Campaigns

By:

ARJUN

A/B Testing of Marketing campaigns

Two data files describing two marketing campaigns are included in the dataset we're using here (Control Campaign and Test Campaign). To begin the A/B testing process, let's import the essential Python libraries and both datasets.

In [1]:
```
import pandas as pd
import datetime
from datetime import date, timedelta

!pip install plotly

import plotly.graph_objects as go
import plotly.express as px
import plotly.io as pio
pio.templates.default = "plotly_white"
```

```
Collecting plotly
  Downloading plotly-5.11.0-py2.py3-none-any.whl (15.3 MB)
     |████████████████████████████████| 15.3 MB 6.4 MB/s
Collecting tenacity>=6.2.0
  Downloading tenacity-8.1.0-py3-none-any.whl (23 kB)
Installing collected packages: tenacity, plotly
Successfully installed plotly-5.11.0 tenacity-8.1.0
```

In [2]:
```
# importing datasets

control_data = pd.read_csv('control_group.csv',sep=';')
test_data = pd.read_csv('test_group.csv',sep=';')
print(control_data.head(5))
print('\n')
print(test_data.head(5))
```

```
      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   

   # 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   

   # of Purchase  
0          618.0  
1          511.0  
2          372.0  
3          340.0  
4            NaN  


   Campaign Name       Date  Spend [USD]  # of Impressions  Reach  \
0  Test Campaign  1.08.2019         3008             39550  35820   
1  Test Campaign  2.08.2019         2542            100719  91236   
2  Test Campaign  3.08.2019         2365             70263  45198   
3  Test Campaign  4.08.2019         2710             78451  25937   
4  Test Campaign  5.08.2019         2297            114295  95138   

   # of Website Clicks  # of Searches  # of View Content  # of Add to Cart  \
0                 3038           1946               1069               894   
1                 4657           2359               1548               879   
2                 7885           2572               2367              1268   
3                 4216           2216               1437               566   
4                 5863           2106                858               956   

   # of Purchase  
0            255  
1            677  
2            578  
3            340  
4            768  
```

The datasets have a few mistakes in the column names. Let's introduce new column names before continuing.

In [3]:
```
control_data.columns = ["Campaign_Name", "Date", "Amount_Spent",
                        "Number_of_Impressions", "Reach", "Website_Clicks",
                        "Searches_Received", "Content_Viewed", "Added_to_Cart",
                        "Purchases"]
control_data.columns
```

Out[3]:
```
Index(['Campaign_Name', 'Date', 'Amount_Spent', 'Number_of_Impressions',
       'Reach', 'Website_Clicks', 'Searches_Received', 'Content_Viewed',
       'Added_to_Cart', 'Purchases'],
      dtype='object')
```

In [4]:
```
test_data.columns=["Campaign_Name", "Date", "Amount_Spent",
                        "Number_of_Impressions", "Reach", "Website_Clicks",
                        "Searches_Received", "Content_Viewed", "Added_to_Cart",
                        "Purchases"]
test_data.columns
```

Out[4]:
```
Index(['Campaign_Name', 'Date', 'Amount_Spent', 'Number_of_Impressions',
       'Reach', 'Website_Clicks', 'Searches_Received', 'Content_Viewed',
       'Added_to_Cart', 'Purchases'],
      dtype='object')
```

Let's check to see if the datasets include null values now:

In [5]:
```
print(control_data.isnull().sum())
print('\n')
print(test_data.isnull().sum())
```

```
Campaign_Name          0
Date                   0
Amount_Spent           0
Number_of_Impressions  1
Reach                  1
Website_Clicks         1
Searches_Received      1
Content_Viewed         1
Added_to_Cart          1
Purchases              1
dtype: int64


Campaign_Name          0
Date                   0
Amount_Spent           0
Number_of_Impressions  0
Reach                  0
Website_Clicks         0
Searches_Received      0
Content_Viewed         0
Added_to_Cart          0
Purchases              0
dtype: int64
```

The dataset for the control campaign has several rows of missing values. Fill in these blanks using the mean of each column to replace the missing values:

In [6]:

```
control_data['Number_of_Impressions'].fillna(value=control_data['Number_of_Impressions'].mean(),inplace=True)
control_data['Reach'].fillna(value=control_data['Reach'].mean(),inplace=True)
control_data['Website_Clicks'].fillna(value=control_data['Website_Clicks'].mean(),inplace=True)
control_data['Searches_Received'].fillna(value=control_data['Searches_Received'].mean(),inplace=True)
control_data['Content_Viewed'].fillna(value=control_data['Content_Viewed'].mean(),inplace=True)
control_data['Added_to_Cart'].fillna(value=control_data['Added_to_Cart'].mean(),inplace=True)
control_data['Purchases'].fillna(value=control_data['Purchases'].mean(),inplace=True)

print(control_data.isnull().sum())
```

```
Campaign_Name           0
Date                    0
Amount_Spent            0
Number_of_Impressions   0
Reach                   0
Website_Clicks          0
Searches_Received       0
Content_Viewed          0
Added_to_Cart           0
Purchases               0
dtype: int64
```

As the missing values are filled, now lets combine both datasets by outer form

In [7]:
```
cb_data= control_data.merge(test_data,how='outer').sort_values(['Date'])
cb_data = cb_data.reset_index(drop=True)
cb_data.head(5)
```

```
/opt/conda/lib/python3.9/site-packages/pandas/core/reshape/merge.py:1204: UserWarning: You are merging on int and float columns where the float values are not equal to their int representation
  warnings.warn(
```

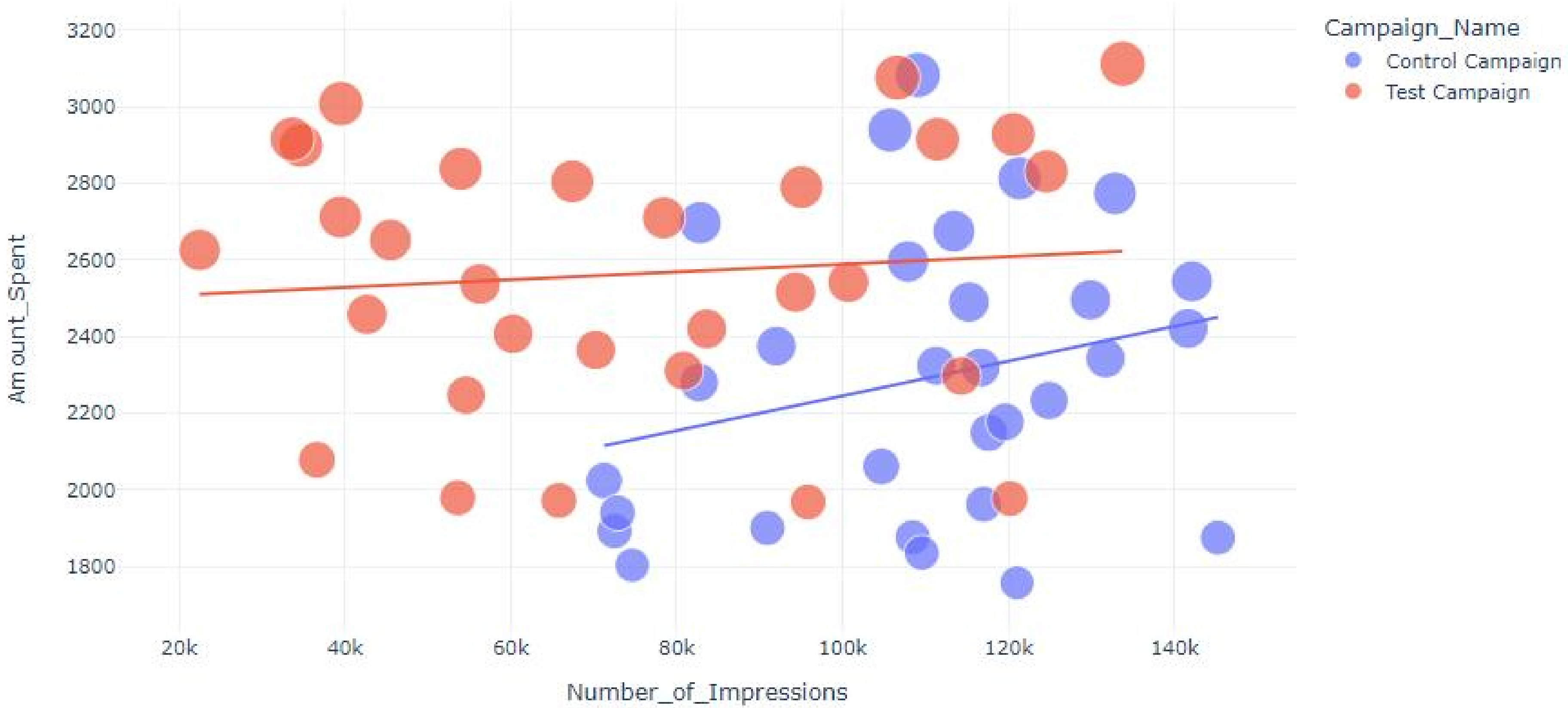
Out[7]:

	Campaign_Name	Date	Amount_Spent	Number_of_Impressions	Reach	Website_Clicks	Searches_Received	Content_Viewed	Added_to_Cart	Purchase
0	Control Campaign	1.08.2019	2280	82702.0	56930.0	7016.0	2290.0	2159.0	1819.0	618
1	Test Campaign	1.08.2019	3008	39550.0	35820.0	3038.0	1946.0	1069.0	894.0	255
2	Test Campaign	10.08.2019	2790	95054.0	79632.0	8125.0	2312.0	1804.0	424.0	275
3	Control Campaign	10.08.2019	2149	117624.0	91257.0	2277.0	2475.0	1984.0	1629.0	734
4	Test Campaign	11.08.2019	2420	83633.0	71286.0	3750.0	2893.0	2617.0	1075.0	668

In order to begin A/B testing, I will examine the link between the quantity of impressions from both ads and the cost of both campaigns.

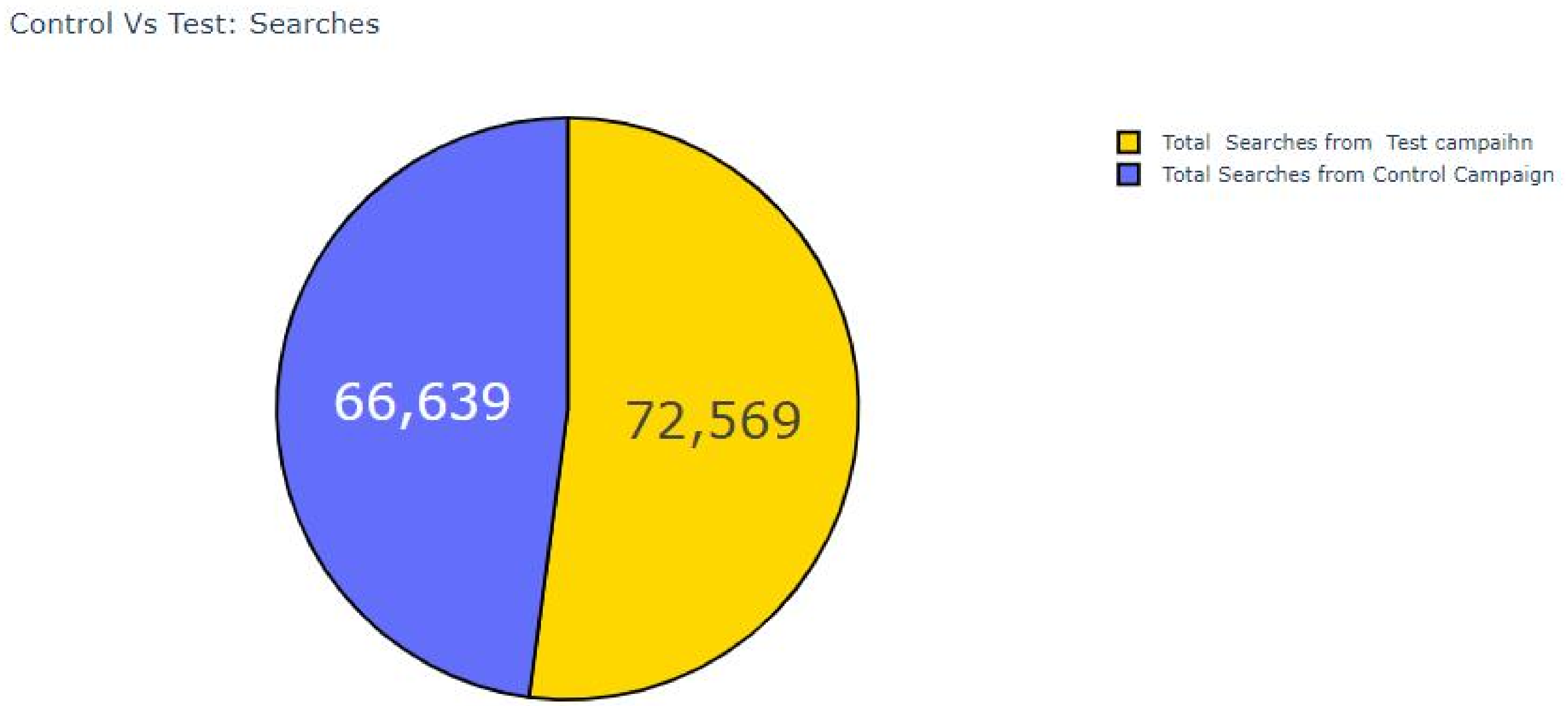
In [8]:

```
figure = px.scatter(data_frame = cb_data,
                    x="Number_of_Impressions",
                    y="Amount_Spent",
                    size="Amount_Spent",
                    color="Campaign_Name",
                    trendline="ols")
figure
```



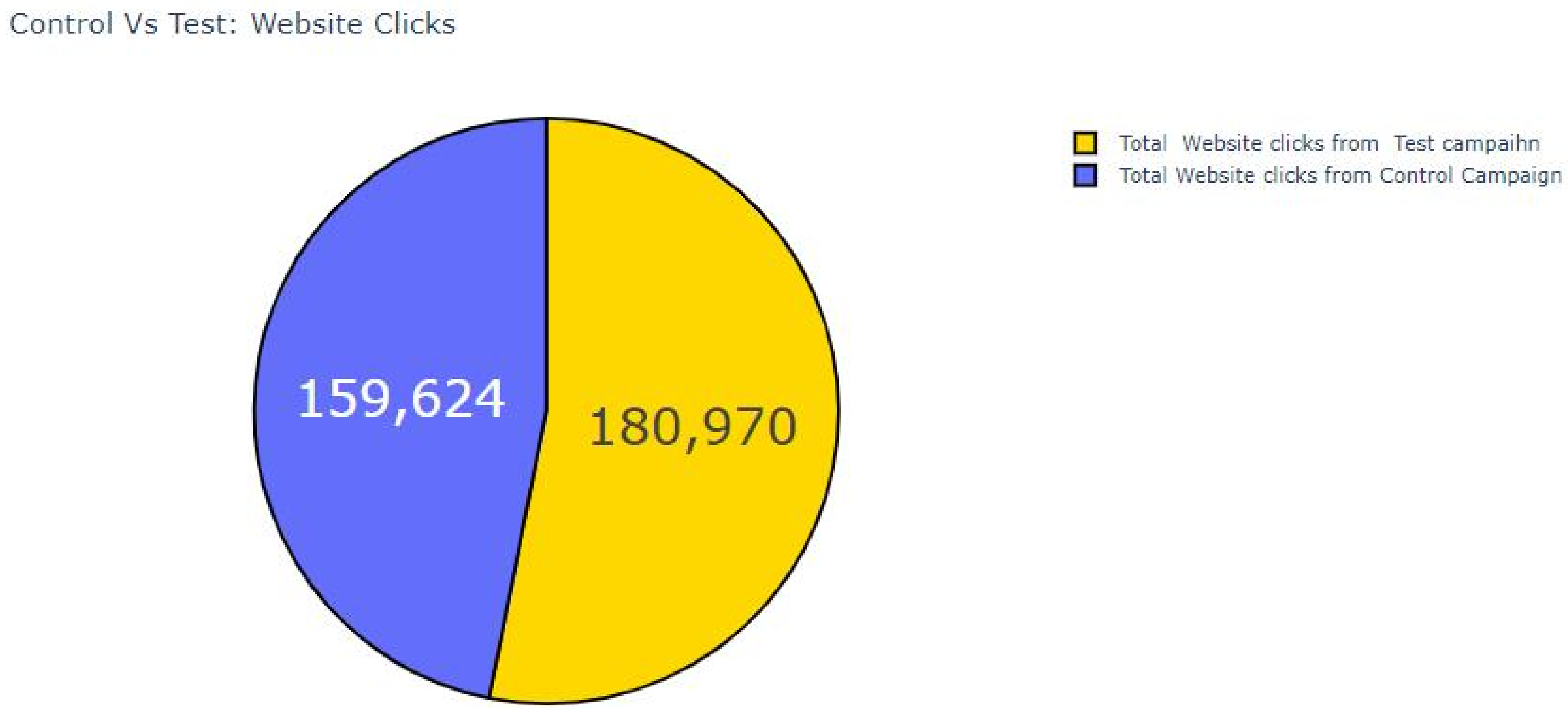
According to the amount of money spent on both campaigns, the control campaign generated more impressions. Let's now examine how many searches from each campaign were made on the website:

In [9]:
```
counts=[round(sum(control_data['Searches_Received'])),round(sum(test_data['Searches_Received']))]
label=['Total Searches from Control Campaign','Total  Searches from  Test campaihn']
colors=['purpor','gold']
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=2)))
fig.show()
```



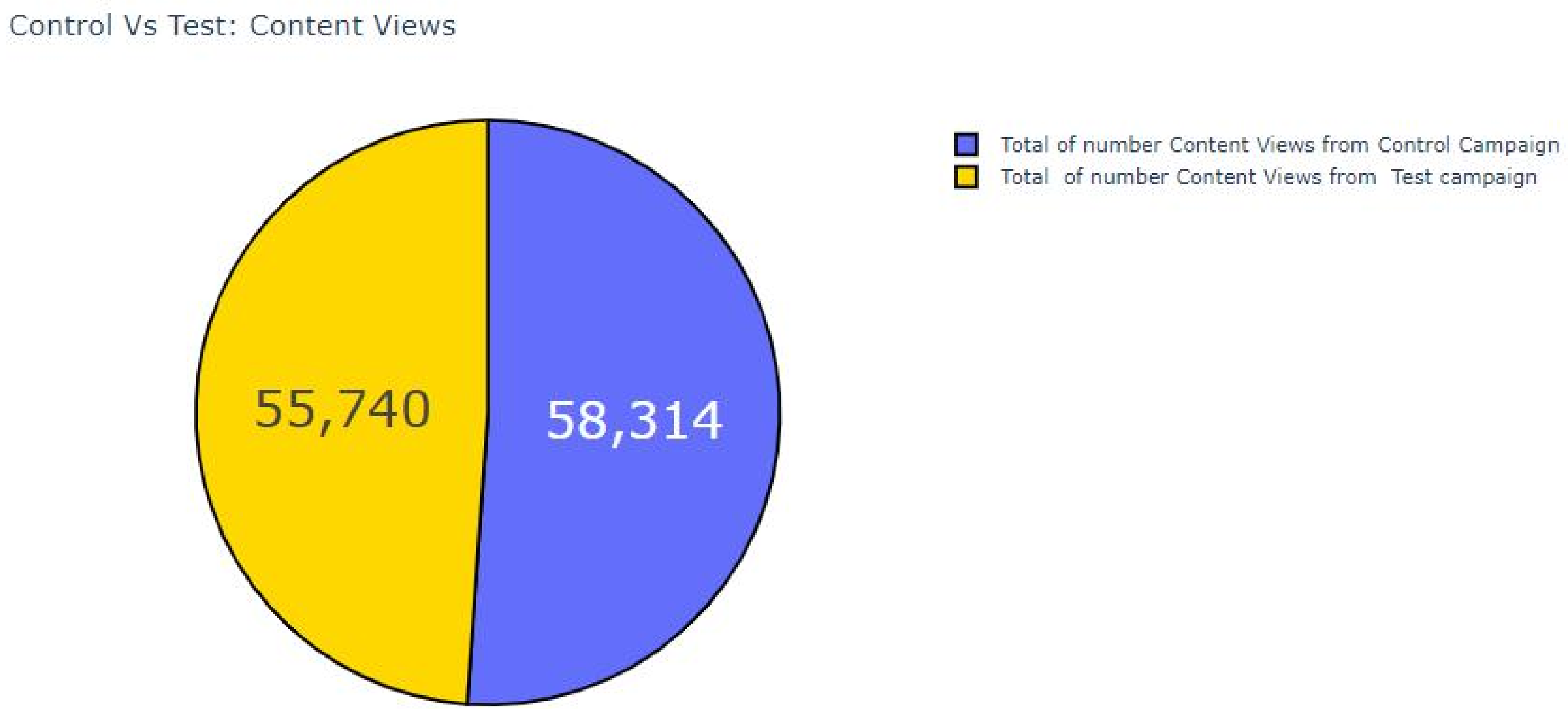
There were more website searches as a result of the test campaign. Let's now examine the number of website clicks from the two campaigns:

In [10]:
```
counts=[round(sum(control_data['Website_Clicks'])),round(sum(test_data['Website_Clicks']))]
label=['Total Website clicks from Control Campaign','Total  Website clicks from  Test campaihn']
colors=['purpor','gold']
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=2)))
fig.show()
```



In terms of website clicks, the test campaign prevails. Let's now examine how much content from each campaign was seen when users arrived at the website:

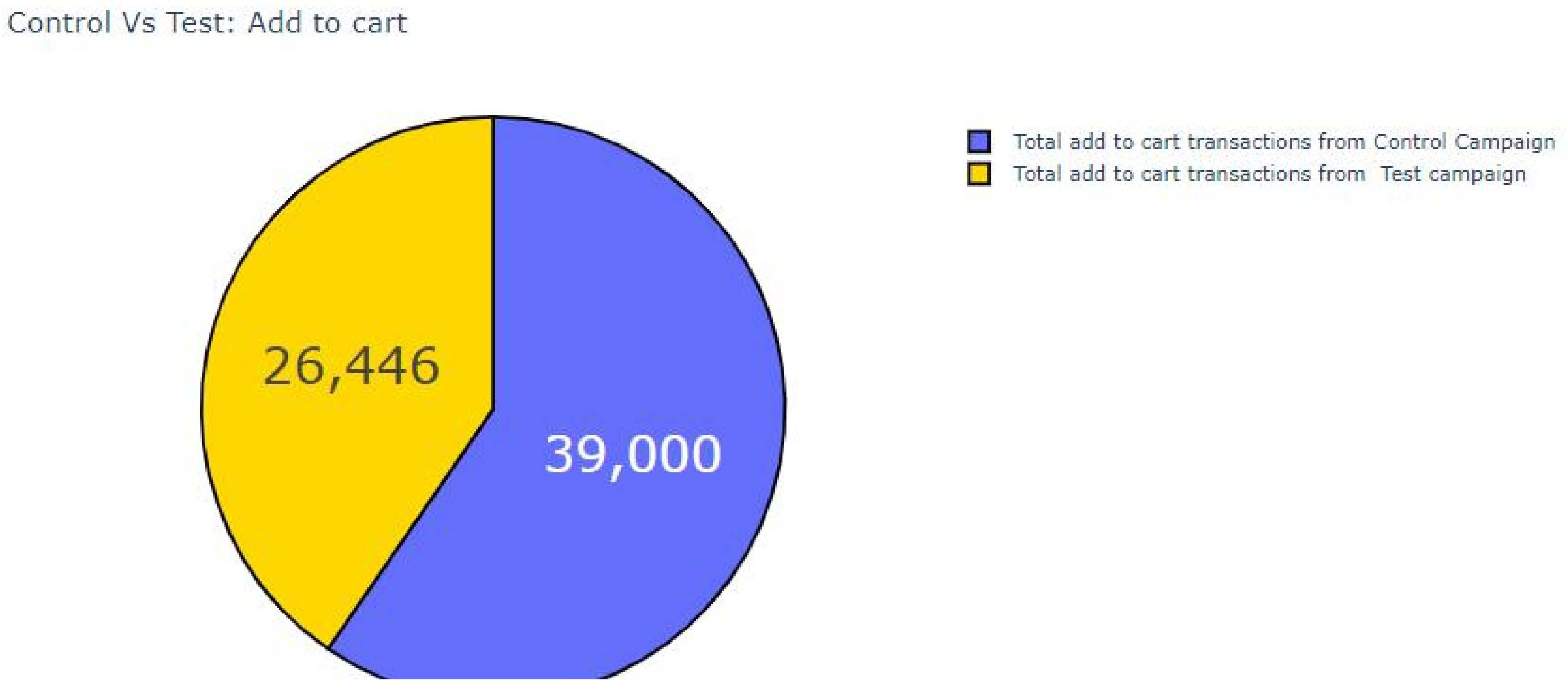
In [11]:
```
counts=[round(sum(control_data['Content_Viewed'])),round(sum(test_data['Content_Viewed']))]
label=['Total of number Content Views from Control Campaign','Total  of number Content Views from  Test campaign']
colors=['purpor','gold']
fig=go.Figure(data=[go.Pie(labels=label,values=counts)])
fig.update_layout(title_text="Control Vs Test: Content Views")
fig.update_traces(hoverinfo='label+percent',textinfo='value',
                textfont_size=30,
                marker=dict(colors=colors,
                            line=dict(color='black',width=2)))
fig.show()
```



Compared to the test campaign, more content was viewed by the control campaign's audience. The control campaign's website engagement is higher than the test campaign's, despite the fact that there is not much of a difference because the control campaign's website clicks were modest.

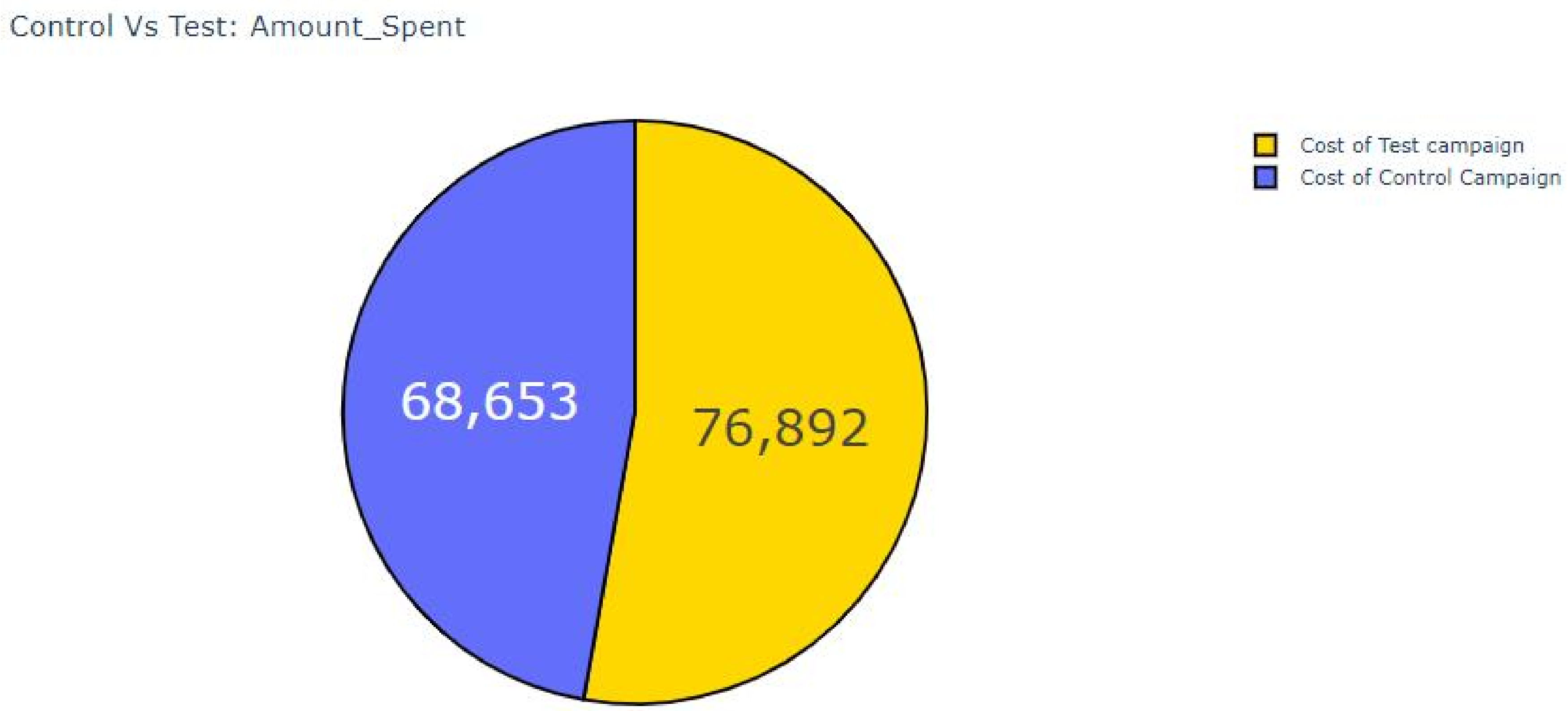
Let's now examine how many items from each campaign were added to the shopping cart:

In [12]:
```
counts=[round(sum(control_data['Added_to_Cart'])),round(sum(test_data['Added_to_Cart']))]
label=['Total add to cart transactions from Control Campaign','Total add to cart transactions from  Test campaign']
colors=['purpor','gold']
fig=go.Figure(data=[go.Pie(labels=label,values=counts)])
fig.update_layout(title_text="Control Vs Test: Add to cart")
fig.update_traces(hoverinfo='label+percent',textinfo='value',
                textfont_size=30,
                marker=dict(colors=colors,
                            line=dict(color='black',width=2)))
fig.show()
```



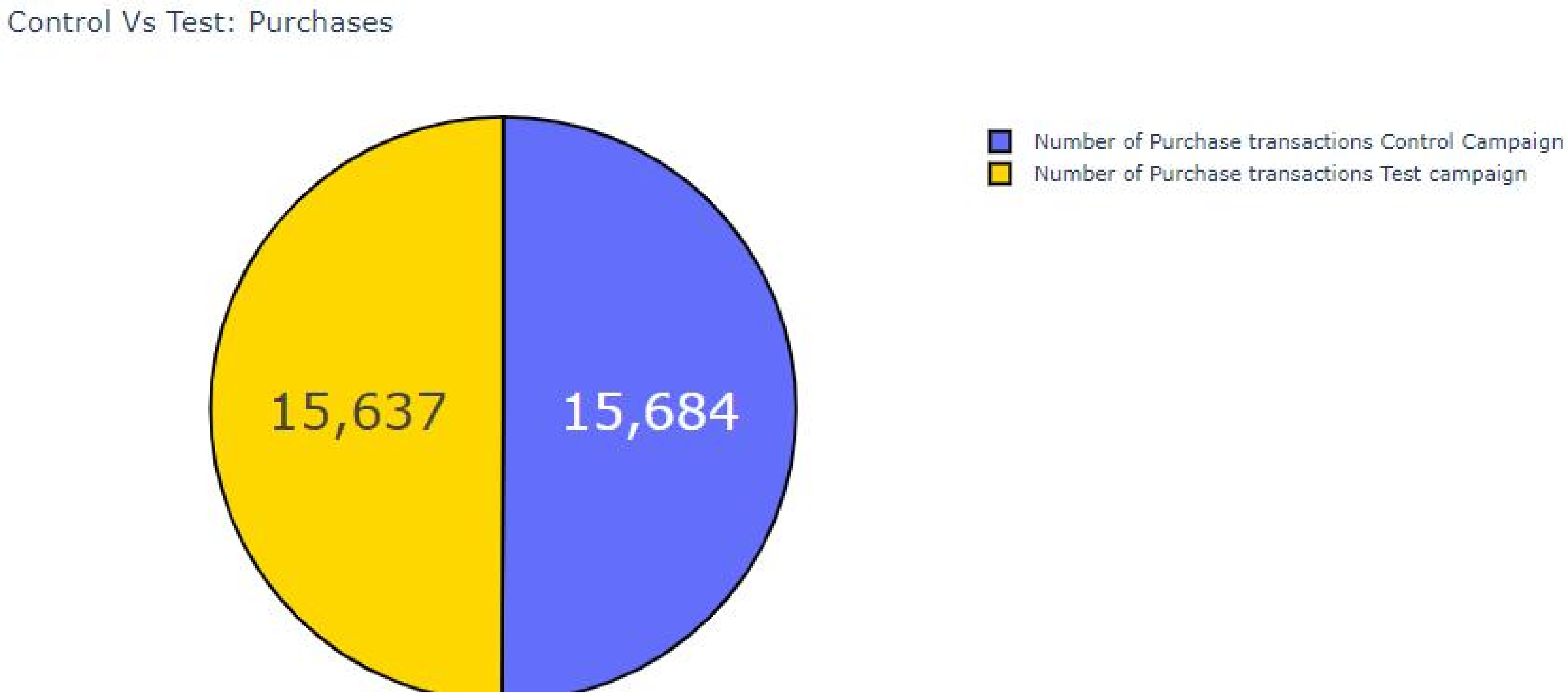
More items from the control campaign were added to the cart despite the poor number of website clicks. Let's now examine the sums spent on the two campaigns:

In [13]:
```
counts=[round(sum(control_data['Amount_Spent'])),round(sum(test_data['Amount_Spent']))]
label=['Cost of Control Campaign','Cost of Test campaign']
colors=['purpor','gold']
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=2)))
fig.show()
```



The test campaign's expenditures are more than those of the control campaign. However, the control campaign is more effective than the test campaign as evidenced by the fact that it led to more content views and more items in the shopping cart. Lets see if there is any major difference in the number of purchases.

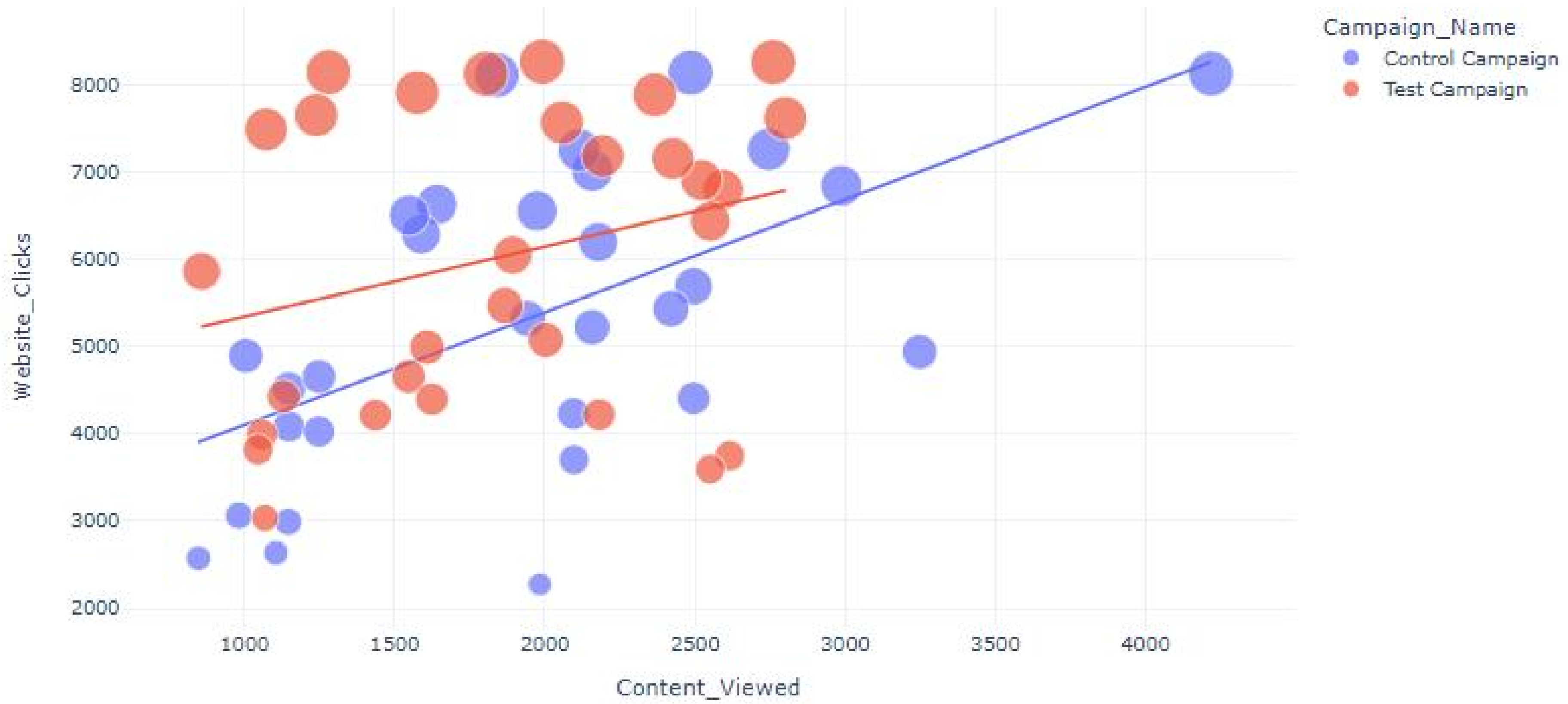
In [14]:
```
counts=[round(sum(control_data['Purchases'])),round(sum(test_data['Purchases']))]
label=['Number of Purchase transactions Control Campaign','Number of Purchase transactions Test campaign']
colors=['purpor','gold']
fig=go.Figure(data=[go.Pie(labels=label,values=counts)])
fig.update_layout(title_text="Control Vs Test: Purchases")
fig.update_traces(hoverinfo='label+percent',textinfo='value',
                  textfont_size=30,
                  marker=dict(colors=colors,
                              line=dict(color='black',width=2)))
fig.show()
```



Only 1% separates the sales made as a result of the two advertising campaigns. The control campaign prevails in this case because it increased sales while using less marketing budget.

Let's now examine some indicators to determine which advertising campaign converts better. I'll start by examining the connection between the number of website clicks and the amount of content from both campaigns that was viewed:

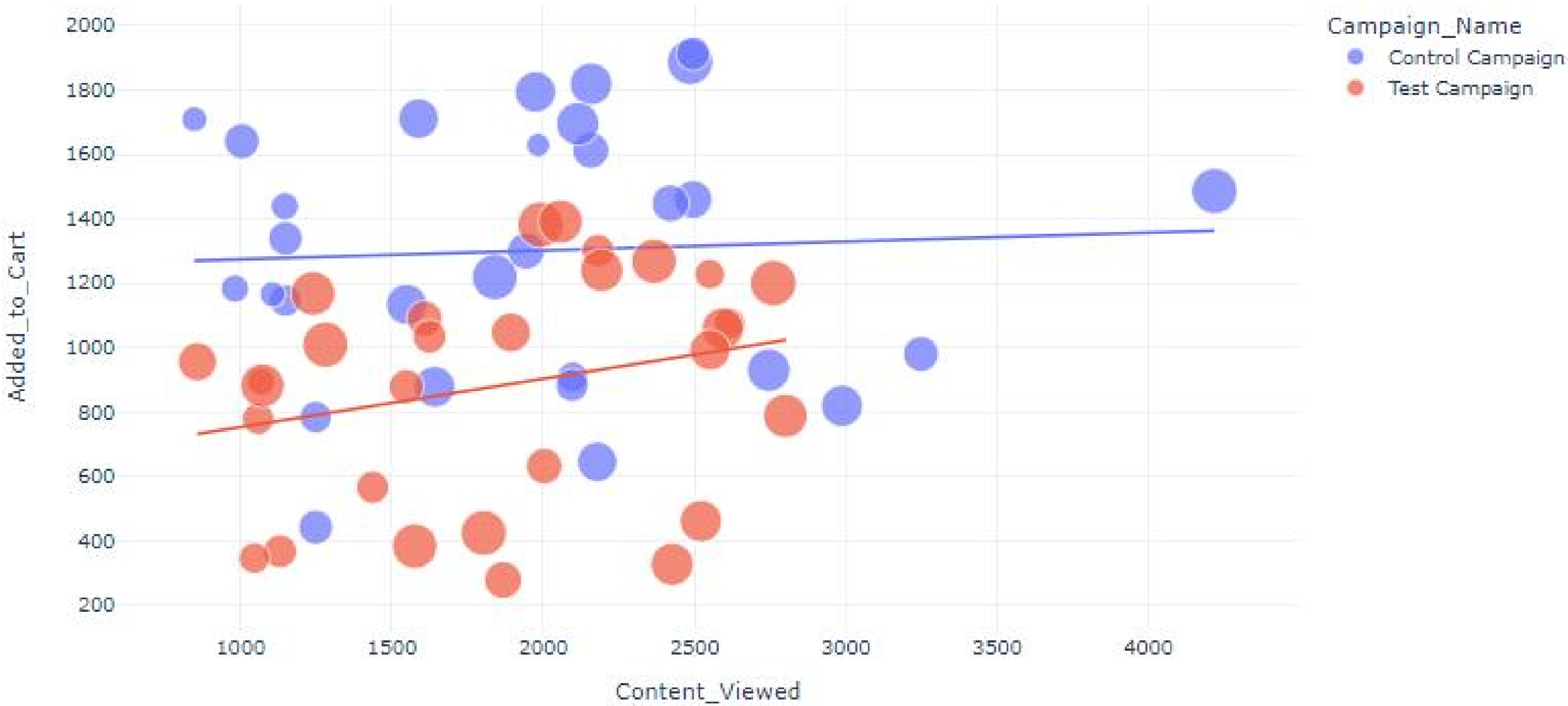
In [15]:
```
figure = px.scatter(data_frame = cb_data,
                    x="Content_Viewed",
                    y="Website_Clicks",
                    size="Website_Clicks",
                    color= "Campaign_Name",
                    trendline="ols")
figure.show()
```



The test campaign had more website clicks, whereas the control campaign has more website click engagement. The control campaign so prevails!

I'll now examine the connection between the quantity of content viewed and the quantity of items added to basket across both campaigns:

In [16]:
```
figure = px.scatter(data_frame = cb_data,
                    x="Content_Viewed",
                    y="Added_to_Cart",
                    size="Website_Clicks",
                    color= "Campaign_Name",
                    trendline="ols")
figure.show()
```



Once more, the control campaign succeeds! Let's now examine the correlation between the number of items added to the cart and the total number of sales generated by the two campaigns:

In [17]:
```
figure = px.scatter(data_frame = cb_data,
                    x="Purchases",
                    y="Added_to_Cart",
                    size="Purchases",
                    color= "Campaign_Name",
                    trendline="ols")
figure.show()
```

The test campaign has a higher conversation rate even though the control campaign generated more sales and more items in the shopping cart.

Conclusion

According to the above A/B tests, the control campaign's products were viewed more often, leading to more items being added to shopping carts and more sales. However, the test campaign had a greater conversation rate for items in the shopping basket. Additionally, the control campaign generates overall higher sales. As a result, the Control campaign can be used to market a variety of products to a larger audience while the Test campaign may be used to promote a particular product to a particular demographic.