A/B Testing means analyzing two marketing strategies to choose the best marketing strategy that can convert more traffic into sales (or more traffic into your desired goal) effectively and efficiently.

In [39]:
```
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
pio.templates.default = "plotly_white"
```

Reading the two campaign datasets

In [40]:
```
control_data = pd.read_csv("control_group.csv", sep = ";")
test_data = pd.read_csv("test_group.csv", sep = ";")
```

In [4]:
```
control_data.head()
```

Out[4]:

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

In [5]:
```
control_data.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
```

In [41]:
```
test_data.head()
```

Out[41]:

	Campaign Name	Date	Spend [USD]	# of Impressions	Reach	# of Website Clicks	# of Searches	# of View Content	# of Add to Cart	# Purchas
0	Test Campaign	1.08.2019	3008	39550	35820	3038	1946	1069	894	25
1	Test Campaign	2.08.2019	2542	100719	91236	4657	2359	1548	879	67
2	Test Campaign	3.08.2019	2365	70263	45198	7885	2572	2367	1268	57
3	Test Campaign	4.08.2019	2710	78451	25937	4216	2216	1437	566	34
4	Test Campaign	5.08.2019	2297	114295	95138	5863	2106	858	956	76

In [42]:
```
test_data.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     30 non-null     int64 
 4   Reach                30 non-null     int64 
 5   # of Website Clicks  30 non-null     int64 
 6   # of Searches        30 non-null     int64 
 7   # of View Content    30 non-null     int64 
 8   # of Add to Cart     30 non-null     int64 
 9   # of Purchase        30 non-null     int64 
dtypes: int64(8), object(2)
memory usage: 2.5+ KB
```

Rearranging and naming column names

In [43]:
```
control_data.columns = ["Campaign Name", "Date", "Amount Spent(USD)",
                        "Number of Impressions", "Reach", "Website_Clicks",
                        "Searches_Received", "Content_Viewed", "Added to Cart"
                        "Purchases"]
```

In [44]:
```
test_data.columns = ["Campaign Name", "Date", "Amount Spent(USD)",
                        "Number of Impressions", "Reach", "Website_Clicks",
                        "Searches_Received", "Content_Viewed", "Added to Cart"
                        "Purchases"]
```

Replacing null values

In [45]:
```
columns_to_fill = ["Number of Impressions", "Reach", "Website_Clicks", "Search
                   "Content_Viewed", "Added to Cart", "Purchases"]

for i in columns_to_fill:
    control_data[i].fillna(value=control_data[i].mean(), inplace=True)
```

Merging two datasets by sorting values by date column

In [46]:
```
ab_data=control_data.merge(test_data,how="outer").sort_values(["Date"])
ab_data = ab_data.reset_index(drop=True)
print(ab_data.head())
```

```
      Campaign Name        Date  Amount Spent(USD)  Number of Impressions  \
0  Control Campaign   1.08.2019               2280                82702.0
1     Test Campaign   1.08.2019               3008                39550.0
2     Test Campaign  10.08.2019               2790                95054.0
3  Control Campaign  10.08.2019               2149               117624.0
4     Test Campaign  11.08.2019               2420                83633.0

     Reach  Website_Clicks  Searches_Received  Content_Viewed  Added to Cart
\
0  56930.0          7016.0             2290.0          2159.0         1819.0
1  35820.0          3038.0             1946.0          1069.0          894.0
2  79632.0          8125.0             2312.0          1804.0          424.0
3  91257.0          2277.0             2475.0          1984.0         1629.0
4  71286.0          3750.0             2893.0          2617.0         1075.0

   Purchases
0      618.0
1      255.0
2      275.0
3      734.0
4      668.0

C:\Users\Admin\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:1205:
UserWarning:

You are merging on int and float columns where the float values are not equal
to their int representation.
```

In [47]:
```
ab_data.head()
```

Out[47]:

	Campaign Name	Date	Amount Spent(USD)	Number of Impressions	Reach	Website_Clicks	Searches_Received
0	Control Campaign	1.08.2019	2280	82702.0	56930.0	7016.0	2290.0
1	Test Campaign	1.08.2019	3008	39550.0	35820.0	3038.0	1946.0
2	Test Campaign	10.08.2019	2790	95054.0	79632.0	8125.0	2312.0
3	Control Campaign	10.08.2019	2149	117624.0	91257.0	2277.0	2475.0
4	Test Campaign	11.08.2019	2420	83633.0	71286.0	3750.0	2893.0

In [48]:
```
ab_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Campaign Name         60 non-null     object 
 1   Date                  60 non-null     object 
 2   Amount Spent(USD)     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
```

To check if dataset has an equal number of samples about both campaigns

In [49]:
```
ab_data['Campaign Name'].value_counts()
```

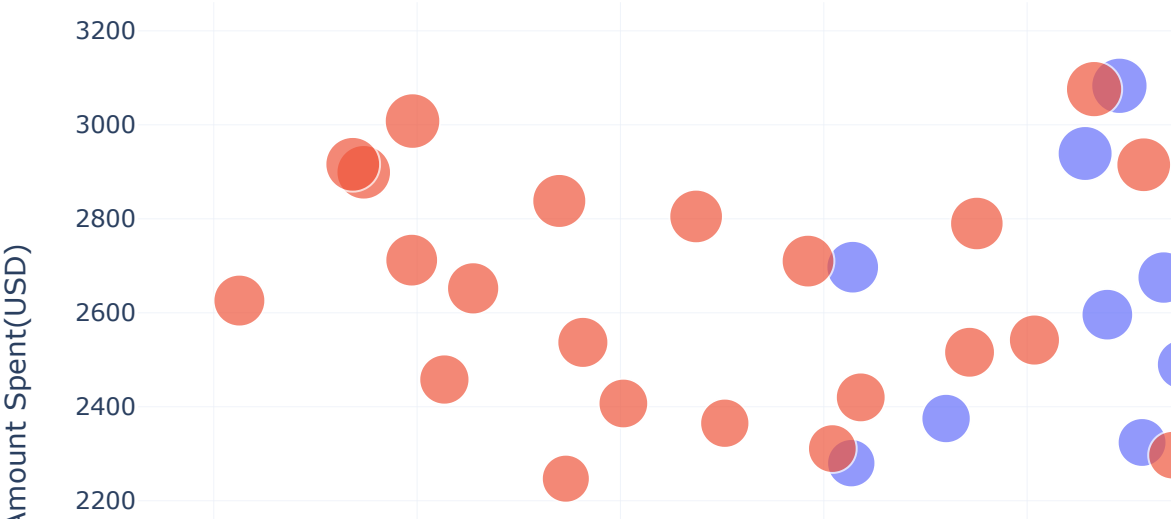
Out[49]:
```
Control Campaign    30
Test Campaign       30
Name: Campaign Name, dtype: int64
```

A/B Testing to Find the Best Marketing Strategy

I will first analyze the relationship between the number of impressions we get from both campaigns and the amount spent on both campaigns using scatter plot

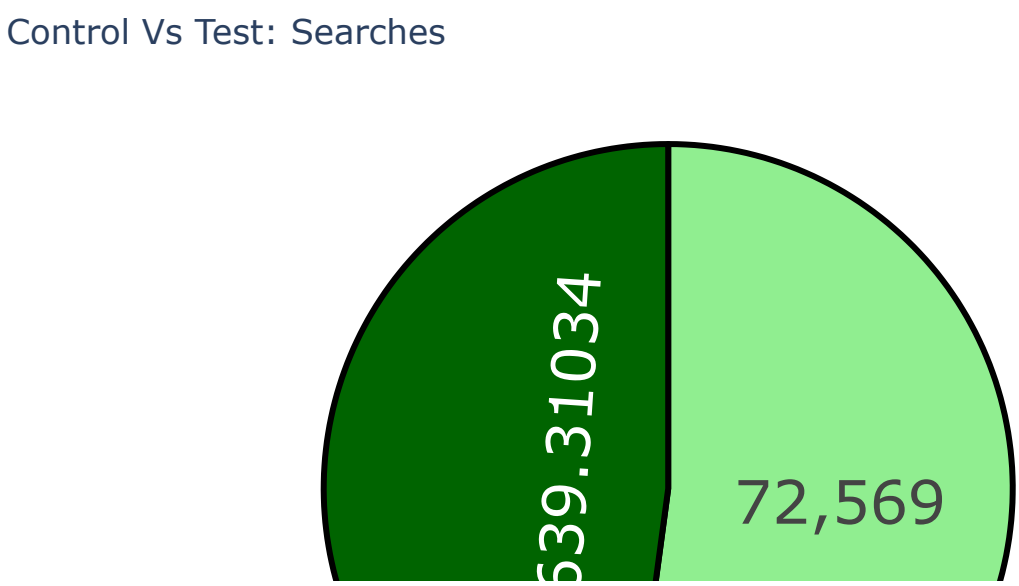
In [50]:
```
grph=px.scatter(ab_data,x='Number of Impressions',y='Amount Spent(USD)',size="
                    color= "Campaign Name",
                    )
```

In [51]:
```
grph.show()
```



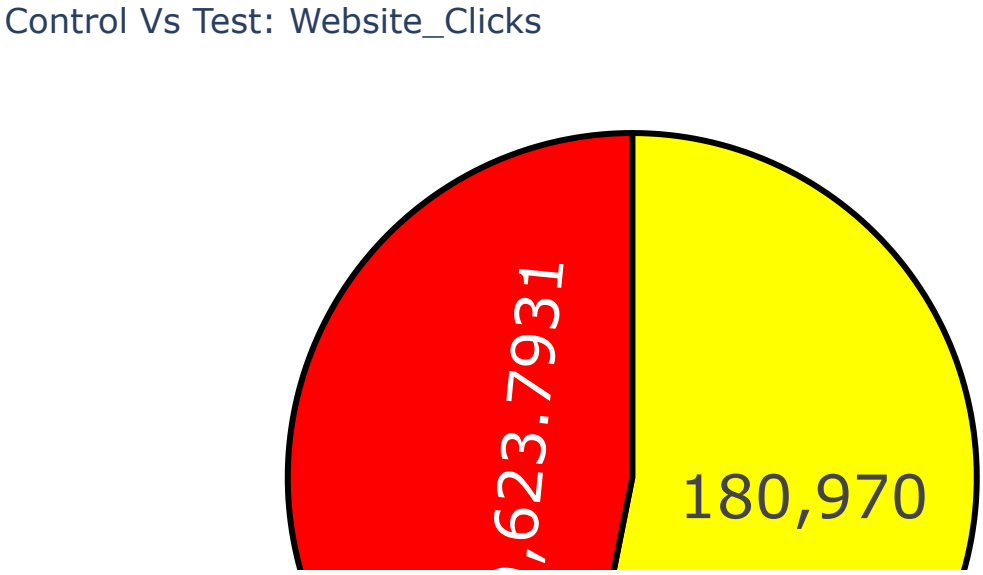
The control campaign resulted in more impressions according to the amount spent on both campaigns.

In [52]:
```
label = ["Total Searches from Control Campaign",
         "Total Searches from Test Campaign"]
counts = [sum(control_data["Searches_Received"]),
          sum(test_data["Searches_Received"])]
colors = ['Darkgreen','lightgreen']
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()
```



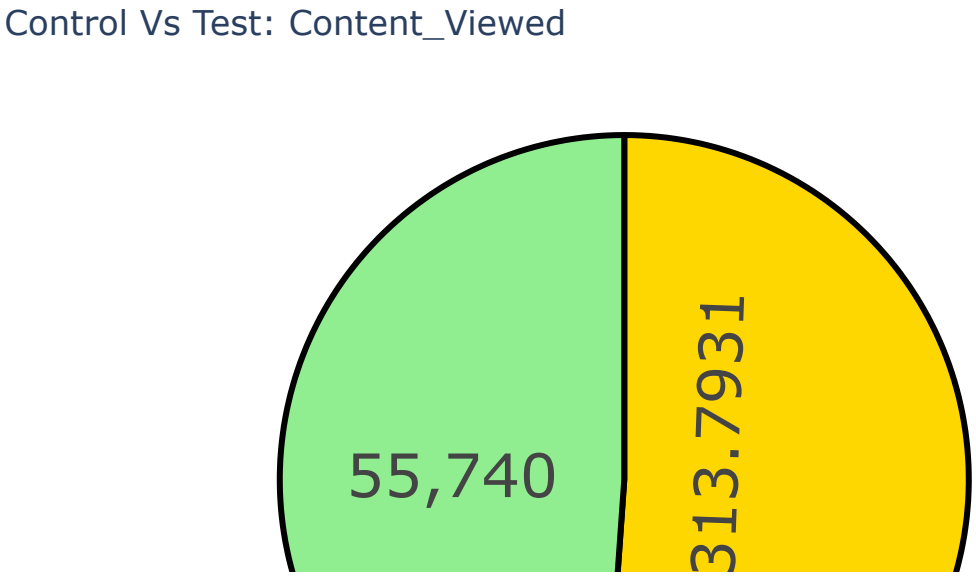
The test campaign resulted in more searches on the website.

In [55]:
```
label = ["Total Website_Clicks from Control Campaign",
         "Total Website_Clicks from Test Campaign"]
counts = [sum(control_data["Website_Clicks"]),
          sum(test_data["Website_Clicks"])]
colors = ['red','yellow']
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()
```



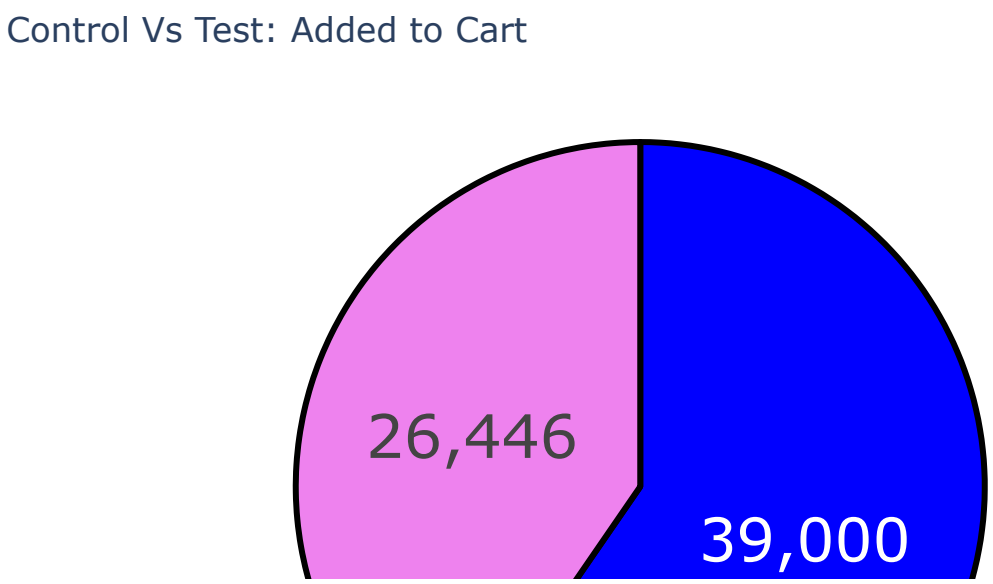
The test campaign has more number of website clicks.

In [56]:
```
label = ["Total Content_Viewed from Control Campaign",
         "Total Content_Viewed from Test Campaign"]
counts = [sum(control_data["Content_Viewed"]),
          sum(test_data["Content_Viewed"])]
colors = ['gold','lightgreen']
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()
```



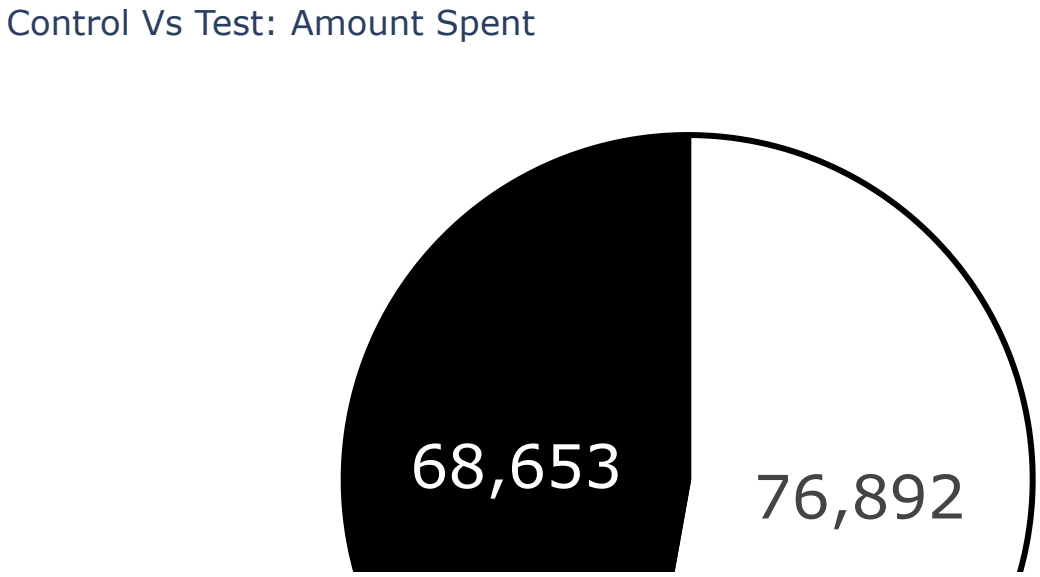
The audience of the control campaign viewed more content than the test campaign.

In [57]:
```
label = ["Total Added to Cart from Control Campaign",
         "Total Added to Cart from Test Campaign"]
counts = [sum(control_data["Added to Cart"]),
          sum(test_data["Added to Cart"])]
colors = ['blue','violet']
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()
```



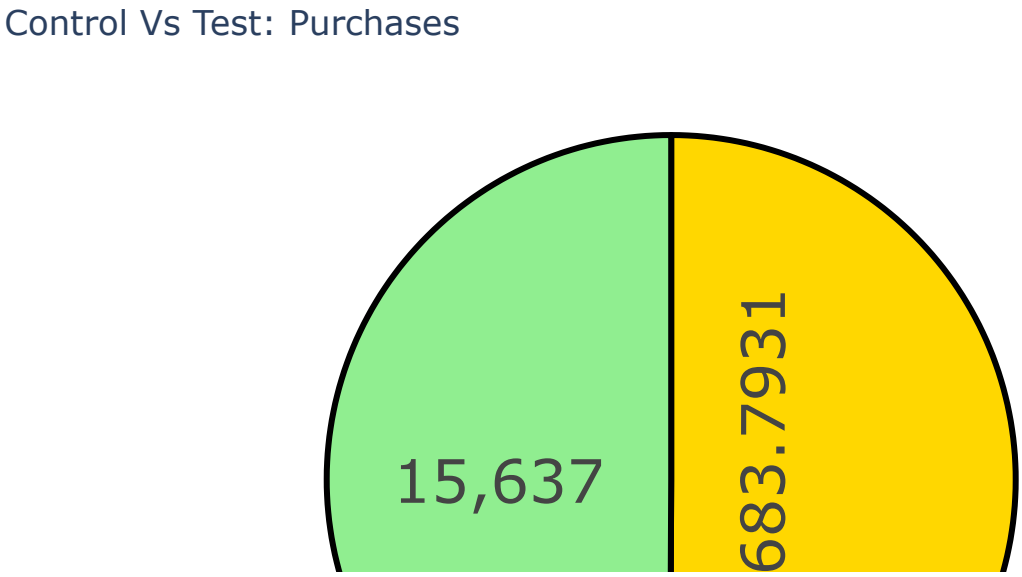
Though there were low website clicks more products were added to the cart from the control campaign.

In [58]:
```
label = ["Total Spent from Control Campaign",
         "Total Spent from Test Campaign"]
counts = [sum(control_data["Amount Spent(USD)"]),
          sum(test_data["Amount Spent(USD)"])]
colors = ['black','white']
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()
```



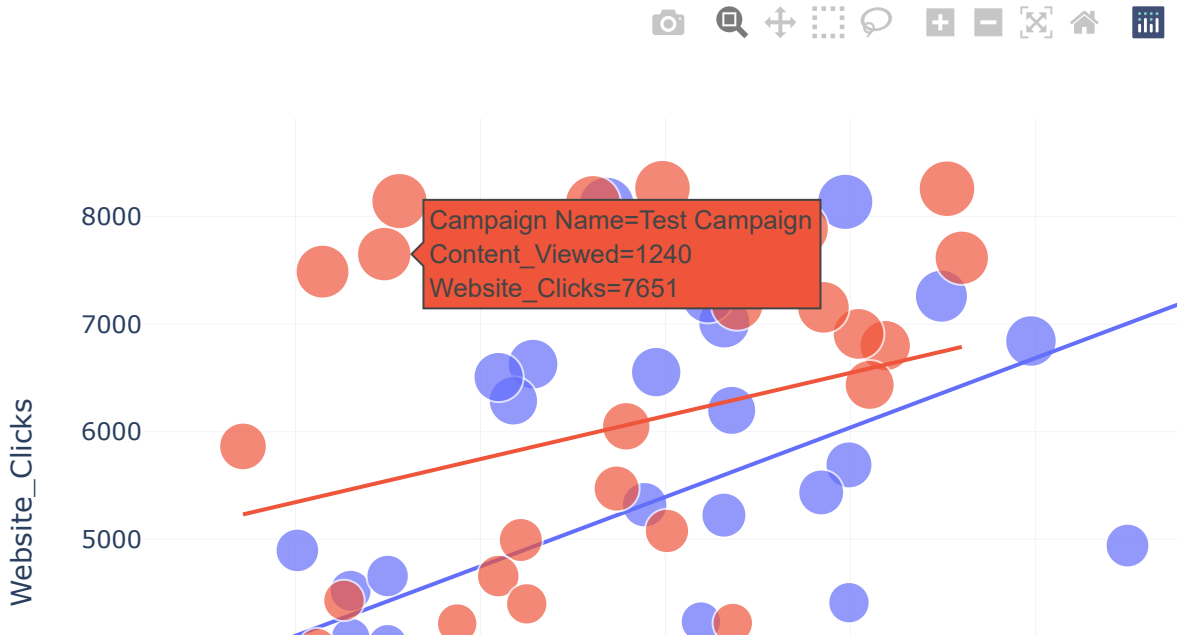
The amount spent on the test campaign is higher than the control campaign.

In [34]:
```
label = ["Purchases Made by Control Campaign",
         "Purchases Made by Test Campaign"]
counts = [sum(control_data["Purchases"]),
          sum(test_data["Purchases"])]
colors = ['gold','lightgreen']
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()
```



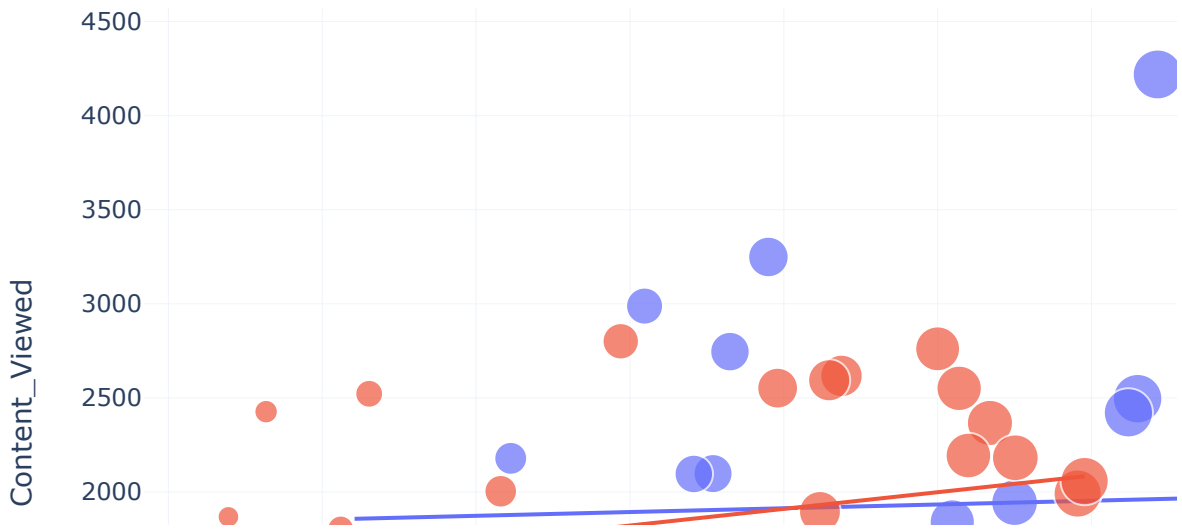
As the Control campaign resulted in more sales in less amount spent on marketing, the control campaign wins here!

In [36]:
```
figure = px.scatter(data_frame = ab_data,
                    x="Content_Viewed",
                    y="Website_Clicks",
                    size="Website_Clicks",
                    color= "Campaign Name",
                    trendline="ols")
figure.show()
```



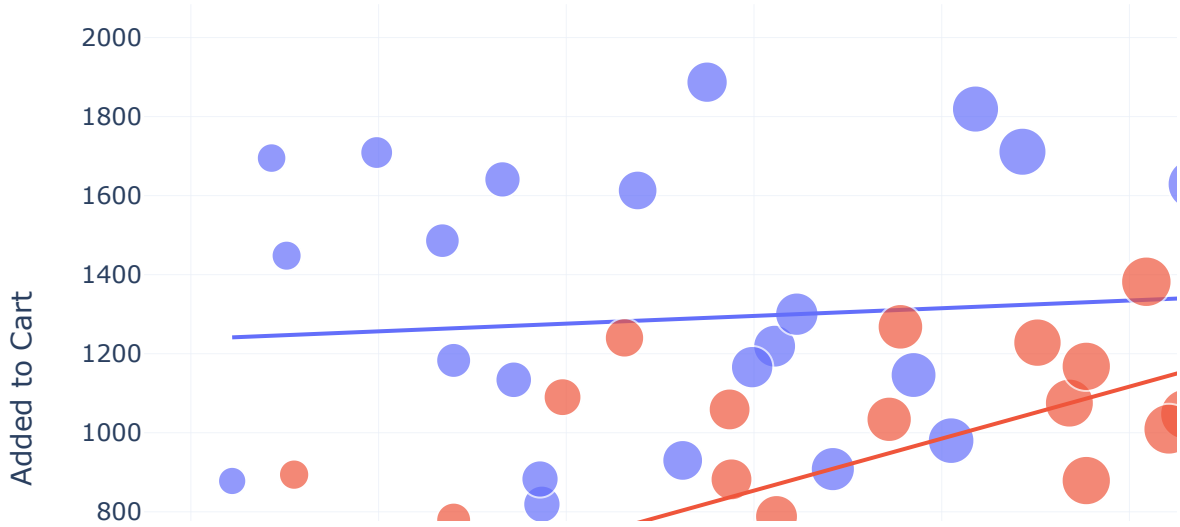
The website clicks are higher in the test campaign, but the engagement from website clicks is higher in the control campaign.

In [37]:
```
figure = px.scatter(data_frame = ab_data,
                    x="Added to Cart",
                    y="Content_Viewed",
                    size="Added to Cart",
                    color= "Campaign Name",
                    trendline="ols")
figure.show()
```



Here also control campaign has better conversion rate

In [38]:
```
figure = px.scatter(data_frame = ab_data,
                    x="Purchases",
                    y="Added to Cart",
                    size="Purchases",
                    color= "Campaign Name",
                    trendline="ols")
figure.show()
```



the conversation rate of the test campaign is higher.

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. 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.

In [ ]: