In [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
pio.templates.default = "plotly_white"
```

In [2]:
```
control_data = pd.read_csv("control_group.csv", sep = ";")
test_data = pd.read_csv("test_group.csv", sep = ";")
```

In [3]:
```
print(control_data.head())
```

```
      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
```

Key Statistics

In [7]:
```
control_data.describe()
```

Out[7]:

	Amount Spent	Number of Impressions	Reach	Website Clicks	Searches Received	Content Viewed	Added to Cart
count	30.000000	29.000000	29.000000	29.000000	29.000000	29.000000	29.000000
mean	2288.433333	109559.758621	88844.931034	5320.793103	2221.310345	1943.793103	1300.000000
std	367.334451	21688.922908	21832.349595	1757.369003	866.089368	777.545469	407.457973
min	1757.000000	71274.000000	42859.000000	2277.000000	1001.000000	848.000000	442.000000
25%	1945.500000	92029.000000	74192.000000	4085.000000	1615.000000	1249.000000	930.000000
50%	2299.500000	113430.000000	91579.000000	5224.000000	2390.000000	1984.000000	1339.000000
75%	2532.000000	121332.000000	102479.000000	6628.000000	2711.000000	2421.000000	1641.000000

	Amount Spent	Number of Impressions	Reach	Website Clicks	Searches Received	Content Viewed	Added to Cart
max	3083.000000	145248.000000	127852.000000	8137.000000	4891.000000	4219.000000	1913.000000

In [8]:
```
test_data.describe()
```

Out[8]:

	Amount Spent	Number of Impressions	Reach	Website Clicks	Searches Received	Content Viewed	Added to Cart
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000
mean	2563.066667	74584.800000	53491.566667	6032.333333	2418.966667	1858.000000	881.533333
std	348.687681	32121.377422	28795.775752	1708.567263	388.742312	597.654669	347.584248
min	1968.000000	22521.000000	10598.000000	3038.000000	1854.000000	858.000000	278.000000
25%	2324.500000	47541.250000	31516.250000	4407.000000	2043.000000	1320.000000	582.500000
50%	2584.000000	68853.500000	44219.500000	6242.500000	2395.500000	1881.000000	974.000000
75%	2836.250000	99500.000000	78778.750000	7604.750000	2801.250000	2412.000000	1148.500000
max	3112.000000	133771.000000	109834.000000	8264.000000	2978.000000	2801.000000	1391.000000

DATA PREPARATION

In [4]:
```
#removing errors in column names

control_data.columns = ["Campaign Name", "Date", "Amount Spent",
                        "Number of Impressions", "Reach", "Website Clicks",
                        "Searches Received", "Content Viewed", "Added to Cart",
                        "Purchases"]

test_data.columns = ["Campaign Name", "Date", "Amount Spent",
                        "Number of Impressions", "Reach", "Website Clicks",
                        "Searches Received", "Content Viewed", "Added to Cart",
                        "Purchases"]
```

Looking for NULL values

In [5]:
```
print(control_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
```

In [6]:
```
print(test_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
```

Filling the missing values with MEAN

In [9]:
```
control_data["Number of Impressions"].fillna(value=control_data["Number of Impressions"
                                                 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)
```

Merging CONTROL and TEST datasets

In [10]:
```
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  Number of Impressions    Reach  \
0  Control Campaign   1.08.2019          2280                82702.0  56930.0
1     Test Campaign   1.08.2019          3008                39550.0  35820.0
2     Test Campaign  10.08.2019          2790                95054.0  79632.0
3  Control Campaign  10.08.2019          2149               117624.0  91257.0
4     Test Campaign  11.08.2019          2420                83633.0  71286.0

   Website Clicks  Searches Received  Content Viewed  Added to Cart  Purchases
0          7016.0             2290.0          2159.0         1819.0      618.0
1          3038.0             1946.0          1069.0          894.0      255.0
```

```
2          8125.0            2312.0           1804.0            424.0      275.0
3          2277.0            2475.0           1984.0           1629.0      734.0
4          3750.0            2893.0           2617.0           1075.0      668.0
C:\Users\Pavan\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:1204: UserWarnin
g: You are merging on int and float columns where the float values are not equal to thei
r int representation
  warnings.warn(
```

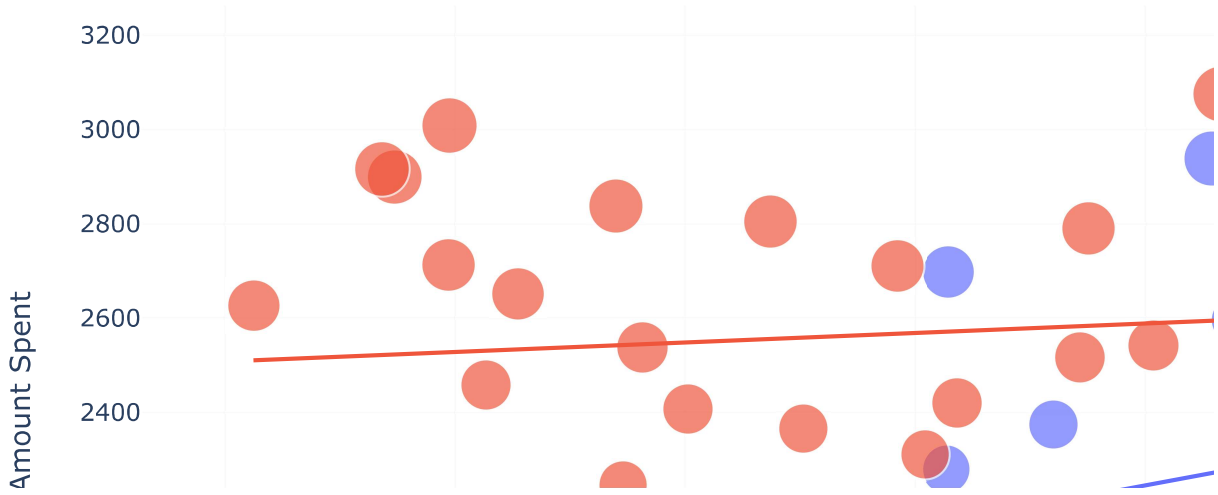
Chechking if the dataset has equal no. of samples

In [11]:
```
print(ab_data["Campaign Name"].value_counts())
```

```
Control Campaign    30
Test Campaign       30
Name: Campaign Name, dtype: int64
```

A/B testing

In [12]:
```
# Analysing relationship between NO. oF IMPRESSION and AMOUNT SPENT
figure = px.scatter(data_frame = ab_data,
                    x="Number of Impressions",
                    y="Amount Spent",
                    size="Amount Spent",
                    color= "Campaign Name",
                    trendline="ols")
figure.show()
```

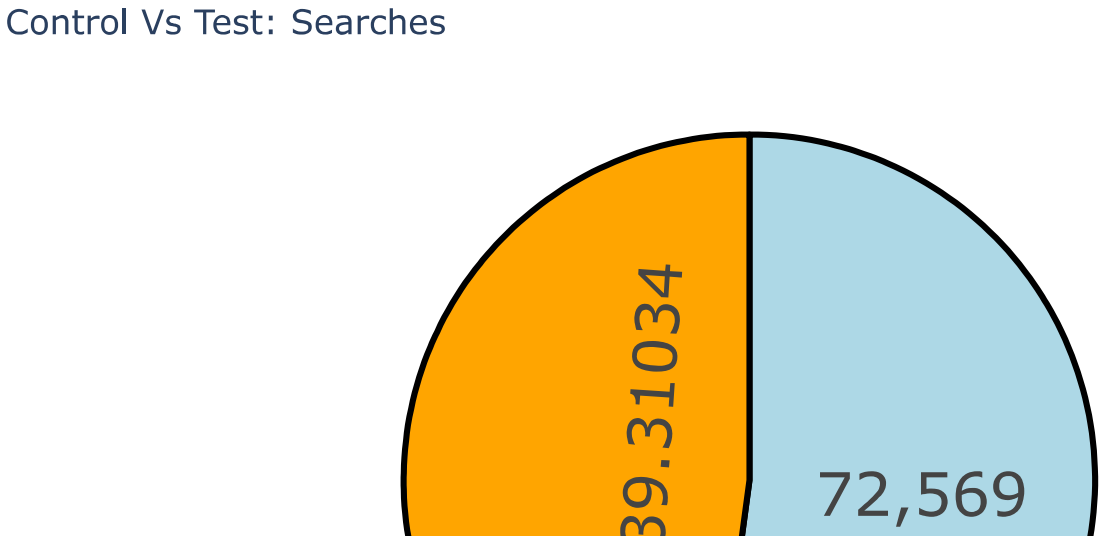


for the visualization above

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

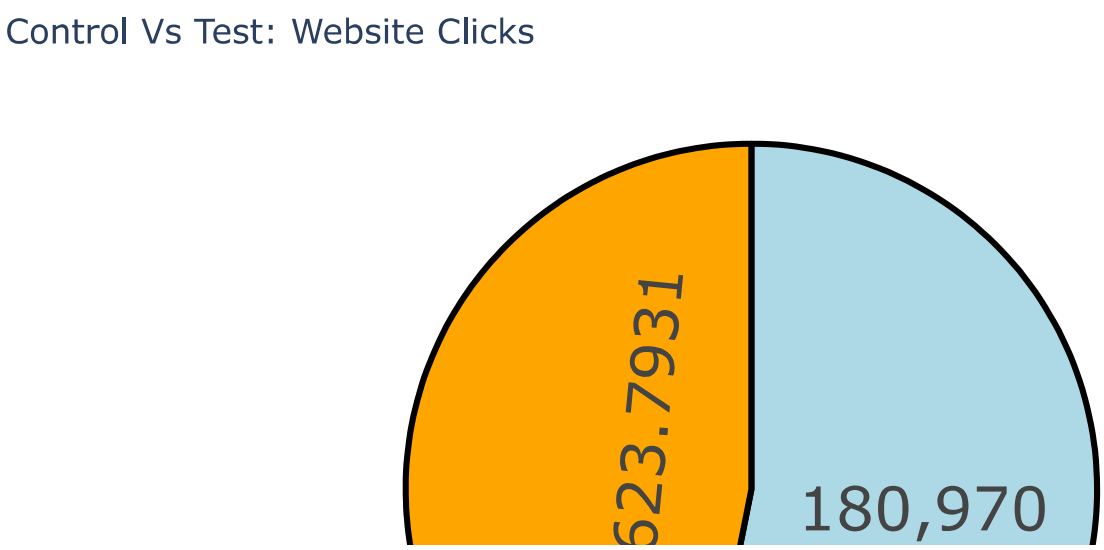
In [16]:

```
#number of searches performed on the website from both campaigns
label = ["Total Searches from Control Campaign",
         "Total Searches from Test Campaign"]
counts = [sum(control_data["Searches Received"]),
          sum(test_data["Searches Received"])]
colors = ['orange','lightblue']
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

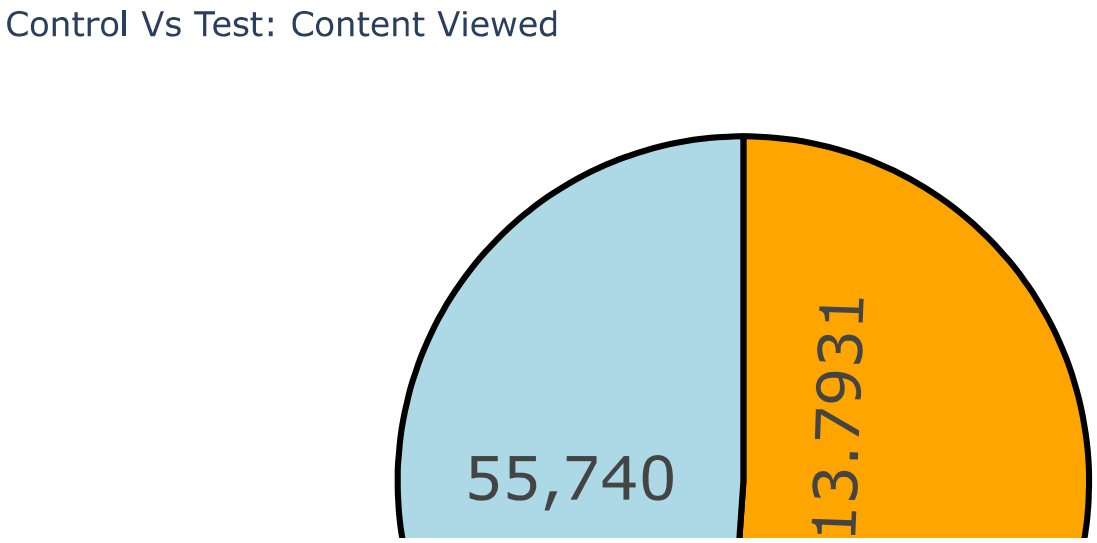
In [18]:
```
#number of website clicks from both campaigns
label = ["Website Clicks from Control Campaign",
         "Website Clicks from Test Campaign"]
counts = [sum(control_data["Website Clicks"]),
          sum(test_data["Website Clicks"])]
colors = ['orange','lightblue']
fig = go.Figure(data=[go.Pie(labels=label, 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 resulted in more searches on the website

In [19]:
```
#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_data["Content Viewed"]),
          sum(test_data["Content Viewed"])]
colors = ['orange','lightblue']
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()
```

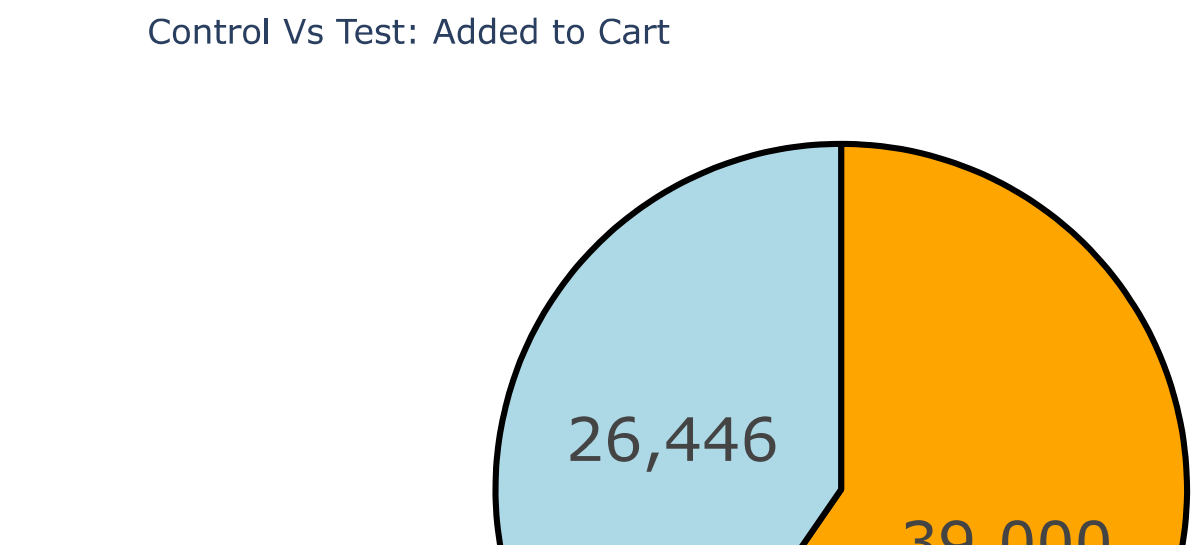


the control campaign had more views than the test campaign

In [21]:

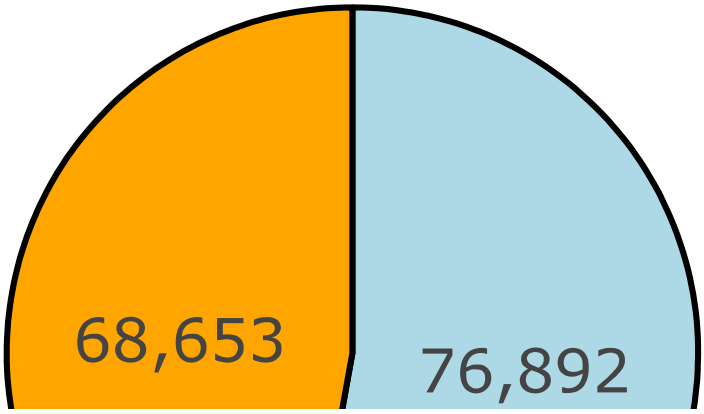
```
#number of products added to the cart from both campaigns
label = ["Products Added to Cart from Control Campaign",
         "Products Added to Cart from Test Campaign"]
counts = [sum(control_data["Added to Cart"]),
          sum(test_data["Added to Cart"])]
colors = ['orange','lightblue']
fig = go.Figure(data=[go.Pie(labels=label, 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()
```



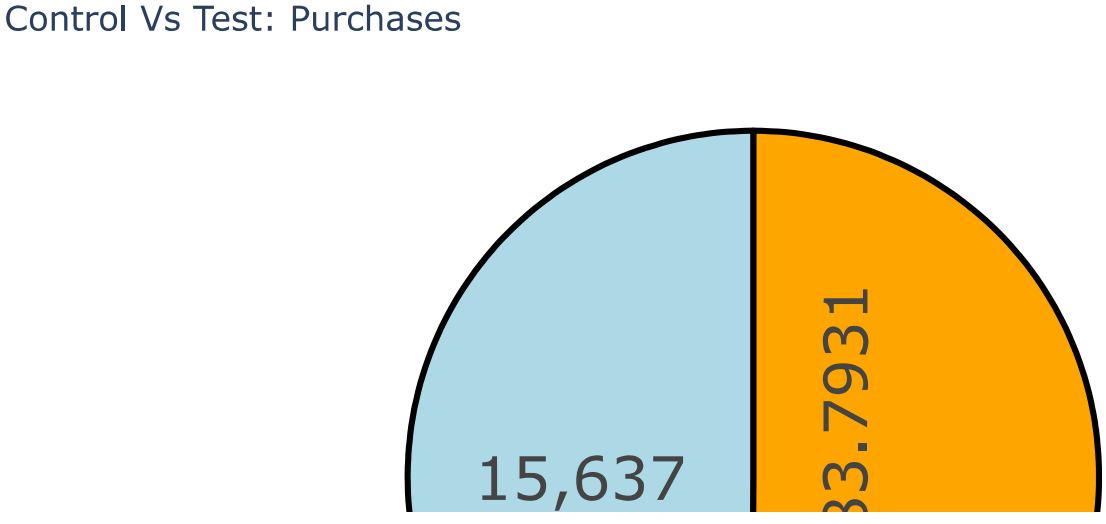
Despite low website clicks more products were added to the cart from the control campaign

In [22]:
```
#amount spent on both campaigns
label = ["Amount Spent in Control Campaign",
         "Amount Spent in Test Campaign"]
counts = [sum(control_data["Amount Spent"]),
          sum(test_data["Amount Spent"])]
colors = ['orange','lightblue']
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()
```



In [24]:

```
#purchases made by both campaigns
label = ["Purchases Made by Control Campaign",
         "Purchases Made by Test Campaign"]
counts = [sum(control_data["Purchases"]),
          sum(test_data["Purchases"])]
colors = ['orange','lightblue']
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()
```

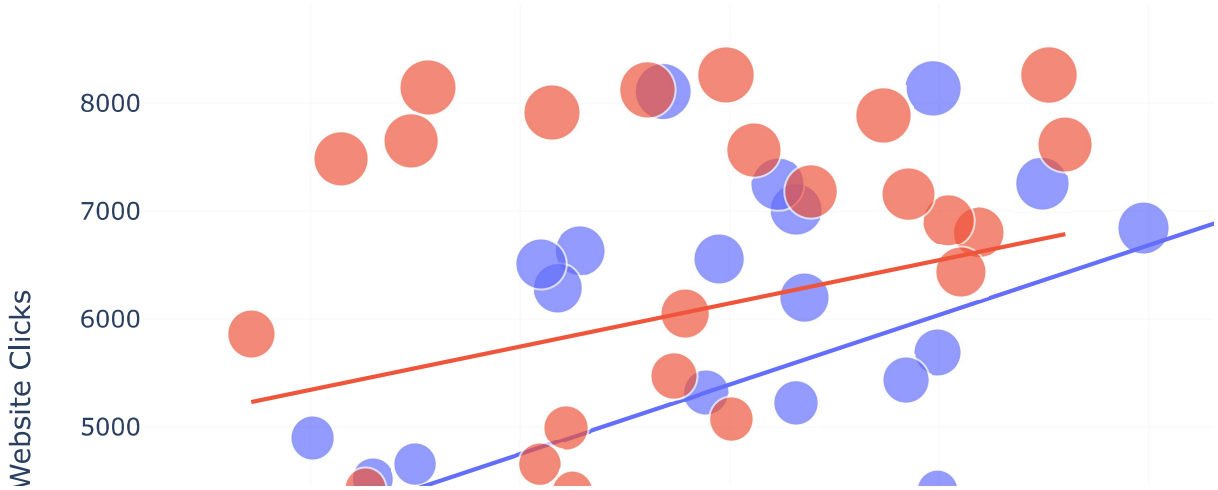


There's only a difference of around 1% in the purchases made from both ad campaigns. As the Control campaign resulted in more sales in less amount spent on marketing, the control campaign wins here!

analyzing some metrics to find which ad campaign converts more

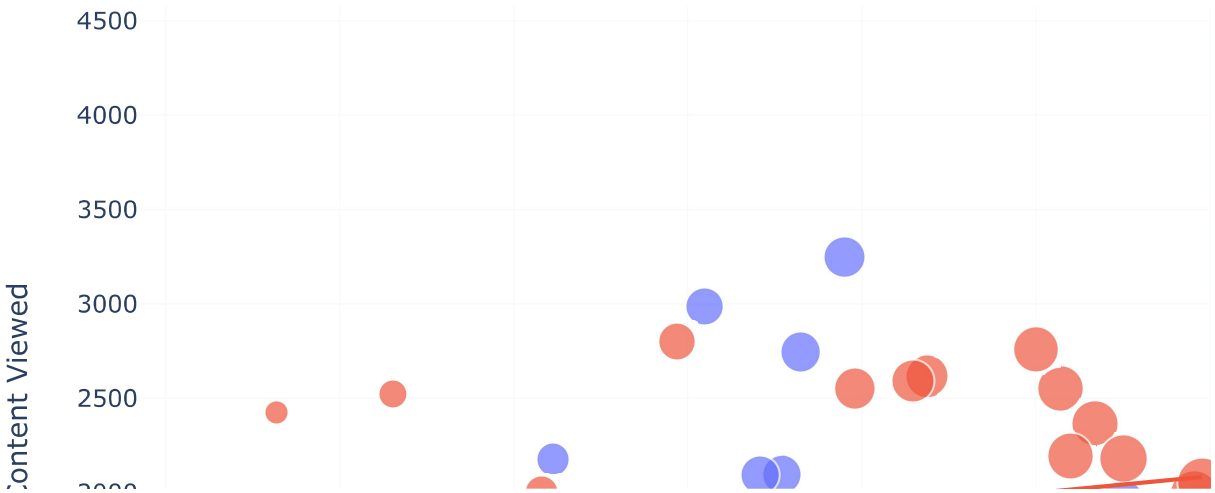
In [25]:

```
#relationship between the number of website clicks and content viewed from both campaig
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. So the control campaign wins!

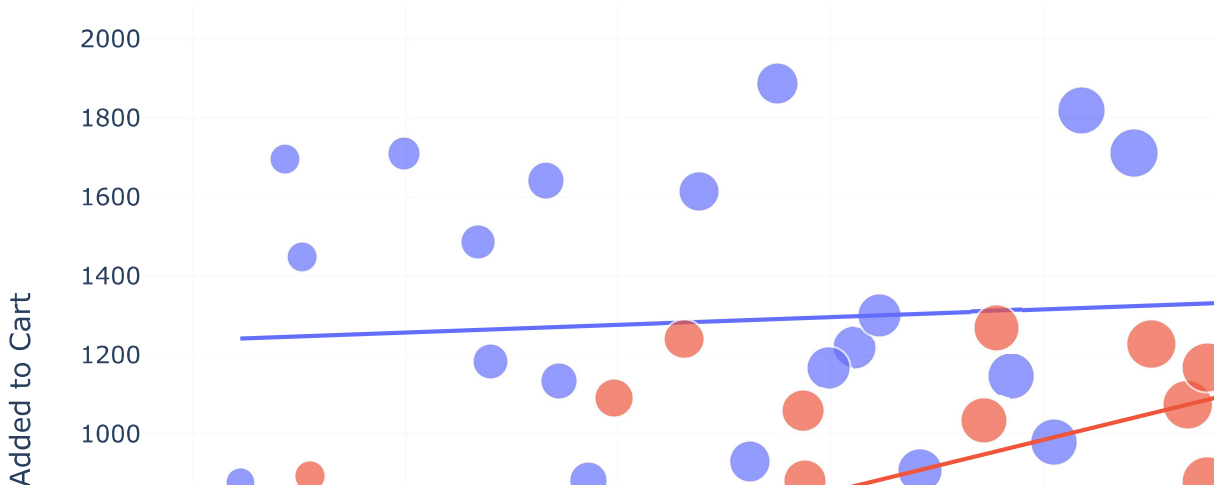
In [26]:
```
# relationship between the amount of content viewed and the number of products added to
figure = px.scatter(data_frame = ab_data,
                    x="Added to Cart",
                    y="Content Viewed",
                    size="Added to Cart",
                    color= "Campaign Name",
                    trendline="ols")
figure.show()
```



the control campaign had more products added to cart

In [27]:
```
#relationship between the number of products added to the cart and the number of sales
figure = px.scatter(data_frame = ab_data,
```

```
                    x="Purchases",
                    y="Added to Cart",
                    size="Purchases",
                    color= "Campaign Name",
                    trendline="ols")
figure.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.

In [29]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\pavan\anaconda3\lib\site-packages
(6.1.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\pavan\anaconda3\lib\site
```

```
-packages (from nbconvert) (1.4.3)
Requirement already satisfied: jinja2>=2.4 in c:\users\pavan\anaconda3\lib\site-packages
(from nbconvert) (2.11.3)
Requirement already satisfied: jupyter-core in c:\users\pavan\anaconda3\lib\site-package
s (from nbconvert) (4.8.1)
Requirement already satisfied: pygments>=2.4.1 in c:\users\pavan\anaconda3\lib\site-pack
ages (from nbconvert) (2.10.0)
Requirement already satisfied: testpath in c:\users\pavan\anaconda3\lib\site-packages (f
rom nbconvert) (0.5.0)
Requirement already satisfied: jupyterlab-pygments in c:\users\pavan\anaconda3\lib\site-
packages (from nbconvert) (0.1.2)
Requirement already satisfied: traitlets>=5.0 in c:\users\pavan\anaconda3\lib\site-packa
ges (from nbconvert) (5.1.0)
Requirement already satisfied: nbformat>=4.4 in c:\users\pavan\anaconda3\lib\site-packag
es (from nbconvert) (5.1.3)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\pavan\anaconda3\lib\site-p
ackages (from nbconvert) (0.3)
Requirement already satisfied: bleach in c:\users\pavan\anaconda3\lib\site-packages (fro
m nbconvert) (4.0.0)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\pavan\anaconda3\lib\site-pa
ckages (from nbconvert) (0.8.4)
Requirement already satisfied: defusedxml in c:\users\pavan\anaconda3\lib\site-packages
(from nbconvert) (0.7.1)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\pavan\anaconda3\lib\si
te-packages (from nbconvert) (0.5.3)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\pavan\anaconda3\lib\site-pac
kages (from jinja2>=2.4->nbconvert) (2.0.1)
Requirement already satisfied: nest-asyncio in c:\users\pavan\anaconda3\lib\site-package
s (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.5.1)
Requirement already satisfied: async-generator in c:\users\pavan\anaconda3\lib\site-pack
ages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.10)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\pavan\anaconda3\lib\sit
e-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (6.1.12)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\pavan\anaconda3\lib\site
-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: pyzmq>=13 in c:\users\pavan\anaconda3\lib\site-packages
(from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (22.2.1)
Requirement already satisfied: tornado>=4.1 in c:\users\pavan\anaconda3\lib\site-package
s (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.1)
Requirement already satisfied: pywin32>=1.0 in c:\users\pavan\anaconda3\lib\site-package
s (from jupyter-core->nbconvert) (228)
Requirement already satisfied: ipython-genutils in c:\users\pavan\anaconda3\lib\site-pac
kages (from nbformat>=4.4->nbconvert) (0.2.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\users\pavan\anaconda3\lib\s
ite-packages (from nbformat>=4.4->nbconvert) (3.2.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\pavan\anaconda3\lib\site-packag
es (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (21.2.0)
Requirement already satisfied: setuptools in c:\users\pavan\anaconda3\lib\site-packages
(from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (58.0.4)
Requirement already satisfied: six>=1.11.0 in c:\users\pavan\anaconda3\lib\site-packages
(from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (1.16.0)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\users\pavan\anaconda3\lib\site-p
ackages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (0.18.0)
Requirement already satisfied: packaging in c:\users\pavan\anaconda3\lib\site-packages
(from bleach->nbconvert) (21.0)
Requirement already satisfied: webencodings in c:\users\pavan\anaconda3\lib\site-package
s (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\pavan\anaconda3\lib\site-pac
kages (from packaging->bleach->nbconvert) (3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]: