RICHARD WAREPAM

How to perform AB Testing using Python

A STEP-BY-STEP GUIDE
Improving your marketing campaign Strategies? Learn all about AB Testing using Python.

Introduction to the Ebook

Hello reader, I am guessing you reached this ebook after reading the article I wrote on **Medium** about "**A/B Testing Using Python**".

So, you might have a basic idea of what "**A/B testing**" is. Therefore, we won't be wasting our time here explaining the basic concepts of "A/B testing.".

Case Study:

A/B testing helps in finding a better approach to finding customers, marketing products, getting a higher reach, or anything that helps a business convert most of its target customers into actual customers.

Perform A/B testing to find the best campaign for the company to get more customers.

Dataset Used:

In this case study, we are using two datasets of two different types of marketing campaigns called:

- Control campaigns
- Test campaigns

Each dataset contains the same features to analyze their performance. The features are:

- Campaign name,
- Date of specific campaign,
- Amount spent on the campaign [USD],
- Number of impressions for each campaign,
- Reach of each campaign,
- Number of website clicks,
- Number of searches, number of views on the content,
- Number of times it results in "Add to Cart," and lastly,
- The number of purchases from each campaign

Datasets: **Download**

Steps for the Project:

- **Step1:** Importing the necessary libraries
- **Step 2:** Data Preparation
- **Step 3**: Calculating the Key Performance Indicators (KPIs)
- **Step 4:** Generating Insights from the KPIs
- **Step 5:** Additional analysis
- **Step 6:** Final conclusion with data-driven insights

Project Outcomes:

1. You will learn about “**How to Perform an A/B Test**" and its importance.
2. You will learn about different KPIs like: **CTR**, **CPC**, **Conversion rate**, **Cost Per Conversion**, **Added to Cart rate**, and **CPM**

Theory of the Project

Getting Started:

To get started with the project, first we need to import the necessary libraries for tasks like **data wrangling**, **data analysis**, and **data visualization**.

Data Preparation:

After importing the libraries, we load the data and initiate the data preparation to make the data better for good analysis and insights.

In data preparation, we need to perform these tasks:

1. Check if there are any **missing values** in the data.
2. If there are any, **impute** the missing values with **mean**, **median**, **mode**, or **eliminate** according to the need for the data.
3. Check for **outliers** too, and fix the outliers before the analysis for better insights.
4. And if any **feature engineering (building new features from the existing data for better analysis)** is needed, we should perform that too for better insights.

A/B Testing Concepts:

While performing the A/B test in this project, we need to calculate some KPIs, and after going through these KPIs, we can conclude which campaign is better for the business.

The KPIs are:

1. Click Through Rate (CTR):

- CTR is a metric that measures the percentage of how many target customers click on a link or a “Call to Action (CTA)" from the total number of customers that view the link or the CTA.

CTR = (Number of Website Clicks / Number of Impressions) x 100

- A high CTR means the ad links or CTA are appealing to the customers and are working well with better engagement, and a low CTR means otherwise it needs improvement.
- "*The higher the CTR, better the SEO of the website*."

2. Cost Per Click (CPC):

- CPC is a metric that measures the price that a business pays for each website click in their marketing campaign ads.

  CPC = (Amount Spend / Number of Website Clicks)

- This metric tells the business how much they pay to reach their target audience. Low CPC means good, and high CPC means otherwise.
- "*The lower the CPC, better the SEO.*"

3. Conversion Rate:

- Conversion rate is a metric that measures the percentage of visiting customers that would take the target action that the campaign aims to and become actual customers.

  Conversion rate= (No. of Purchases/No. of Website Clicks) x 100

- A high conversion rate means the marketing campaign is on the right path and is working perfectly, whereas a low conversion rate means otherwise and needs improvement.
- "*The higher the conversion rate, better the SEO.*"

4. Cost Per Conversion:

- Cost per conversion is a metric that measures the average amount that a business pays for each customer in their marketing campaign ads.

  Cost per Conversion = (Amount Spend / Number of Purchases)

- This metric tells the business how much they pay to secure a customer. Low cost per conversion means good, and high cost per conversion means otherwise.
- "*The lower the cost per conversion, the better the SEO.*"

5. Added to Cart Rate (ACR):

- Added to cart rate (ACR) is a metric that measures the percentage of visiting customers to a website who atleast add a product to their cart for further purchase.

  ACR = (No. of Items added to cart / No. of Website Clicks) x 100

- A high ACR means the ad campaign is working perfectly and the products of the business is really appealing to the customers whereas low ACR means otherwise and it needs improvement.
- "*The higher the ACR, the better the SEO.*"

6. Cost Per Impression (CPM):

- Cost per impression is also known as cost per thousand impressions. It is a metric that measures the amount that the business pays each time to their ads to get 1000 impressions.

  CPM = (Amount Spend / No. of Impressions) x 1000

- This metric tells the business how much they for 1000 impressions every time. And, low CPM means good and higher CPM means otherwise.
- "*The lower the CPM, the better their SEO.*"

Hence, these are the important measures that we need to analyze to make a better insights-driven decisions for a great marketing campaigns and improve the profitability of the business.

Additional Analysis:

Apart from all the KPIs, we also need to analyze some additional features according to the dataset.

In this dataset, we can go through features like:

1. **Reach**
2. **Number of Searches Received**
3. **Number of customers that viewed the content**

By analyzing these three features for each campaign, we can conclude our insights.

Hence, these are all the concepts needed to implement the project. Now, let’s get started with the implementation process.

abtesting

October 3, 2023

1 Importing necessary libraries:

1. "**Pandas**" for Data Wrangling (Cleaning, Exploring and Manipulating) and Data Analysis.
2. "**Numpy**" for working with Arrays, if any necessary arises.
3. "**Matplotlib**" and "**Seaborn**" for Data Visualization in analysis.
4. "**Datetime**" for manipulating data and time object data.
5. "**Warnings**" to ignore the unnecessary warnings.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import warnings
warnings.filterwarnings('ignore')
```

2 Loading Data:

- Here, in this project, we got two datasets for 2 different ad campaigns. So, first we are loading both the datasets separately, named as "**control_df**" and "**test_df**"

[2]:
```
control_df = pd.read_csv("control_group.csv", sep = ';')
test_df = pd.read_csv("test_group.csv", sep = ';')
```

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

[3]:

	Campaign Name	Date	Spend [USD]	# of Impressions	Reach	# of Website Clicks	# of Searches	# of View Content	# of Add to Cart	\
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					
4	Control Campaign	5.08.2019	1835	NaN	NaN					

```
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
```

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

[4]:
```
   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
```

3 Data Preparation:

- **Step 1:** While loading the data, we have seen that the column names of both the datasets is not optimized, for easy understandability. So, we need to rename the columns in a proper way.

[5]:
```
control_df.columns = ['Campaign Name', 'Date', 'Amount Spend', 'Number of␣
↪Impressions',
                      'Reach', 'Website Clicks', 'Searches Received', 'Content␣
↪Viewed',
                      'Added to Cart', 'Purchases']
```

```
test_df.columns = ['Campaign Name', 'Date', 'Amount Spend', 'Number of␣
 ↪Impressions',
                    'Reach', 'Website Clicks', 'Searches Received', 'Content␣
 ↪Viewed',
                    'Added to Cart', 'Purchases']
```

- **Step 2:** Now, explore the data and their datatypes with "**.info()**" function, to check if all the features are in their desired data type. Here, we have found that "Date" feature is shown as object datatype. So, we need to change the datatype to "date time" for both the datasets.

[6]:
```
test_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   Amount Spend           30 non-null     int64
 3   Number of Impressions  30 non-null     int64
 4   Reach                  30 non-null     int64
 5   Website Clicks         30 non-null     int64
 6   Searches Received      30 non-null     int64
 7   Content Viewed         30 non-null     int64
 8   Added to Cart          30 non-null     int64
 9   Purchases              30 non-null     int64
dtypes: int64(8), object(2)
memory usage: 2.5+ KB
```

[7]:
```
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   Amount Spend           30 non-null     int64
 3   Number of Impressions  29 non-null     float64
 4   Reach                  29 non-null     float64
 5   Website Clicks         29 non-null     float64
 6   Searches Received      29 non-null     float64
 7   Content Viewed         29 non-null     float64
 8   Added to Cart          29 non-null     float64
 9   Purchases              29 non-null     float64
dtypes: float64(7), int64(1), object(2)
```

```
memory usage: 2.5+ KB
```

[8]:
```
control_df['Date'] = pd.to_datetime(control_df['Date'])
test_df['Date'] = pd.to_datetime(test_df['Date'])
```

[9]:
```
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     datetime64[ns]
 2   Amount Spend           30 non-null     int64
 3   Number of Impressions  29 non-null     float64
 4   Reach                  29 non-null     float64
 5   Website Clicks         29 non-null     float64
 6   Searches Received      29 non-null     float64
 7   Content Viewed         29 non-null     float64
 8   Added to Cart          29 non-null     float64
 9   Purchases              29 non-null     float64
dtypes: datetime64[ns](1), float64(7), int64(1), object(1)
memory usage: 2.5+ KB
```

- **Step 3:** Further, We can check for missing values in the data. If we find it, we need to fix it in the best possible way for better insights. Here, in the "**control_df**", there are some missing values and after analyzing the data, we find the best way to fix it, is to "**impute mean value of the feature in the missing values**".

[10]:
```
test_df.isnull().sum()
```

[10]:
```
Campaign Name            0
Date                     0
Amount Spend             0
Number of Impressions    0
Reach                    0
Website Clicks           0
Searches Received        0
Content Viewed           0
Added to Cart            0
Purchases                0
dtype: int64
```

[11]:
```
control_df.isnull().sum()
```

[11]:
```
Campaign Name            0
Date                     0
```

```
Amount Spend              0
Number of Impressions     1
Reach                     1
Website Clicks            1
Searches Received         1
Content Viewed            1
Added to Cart             1
Purchases                 1
dtype: int64
```

[12]:
```
control_df['Number of Impressions'].fillna(value= control_df['Number of␣
↪Impressions'].mean(),
                                           inplace= True)
control_df['Reach'].fillna(value= control_df['Reach'].mean(),
                                           inplace= True)
control_df['Website Clicks'].fillna(value= control_df['Website Clicks'].mean(),
                                           inplace= True)
control_df['Searches Received'].fillna(value= control_df['Searches Received'].
↪mean(),
                                           inplace= True)
control_df['Content Viewed'].fillna(value= control_df['Content Viewed'].mean(),
                                           inplace= True)
control_df['Added to Cart'].fillna(value= control_df['Added to Cart'].mean(),
                                           inplace= True)
control_df['Purchases'].fillna(value= control_df['Purchases'].mean(),
                                           inplace= True)
```

[13]:
```
control_df.isnull().sum()
```

[13]:
```
Campaign Name             0
Date                      0
Amount Spend              0
Number of Impressions     0
Reach                     0
Website Clicks            0
Searches Received         0
Content Viewed            0
Added to Cart             0
Purchases                 0
dtype: int64
```

- "**Step 4:**" As we have fixed the missing values now, we can merge both the datasets into one with "**.merge()**" function. So, we have merged both control_df and test_df into one dataframe named as "df".

[14]:
```
df = control_df.merge(test_df,
                      how= 'outer').sort_values(['Date'])
```

```python
df = df.reset_index(drop = True)
df.head()
```

[14]:

```
        Campaign Name       Date  Amount Spend  Number of Impressions     Reach  \
0  Control Campaign 2019-01-08          2280                82702.0   56930.0
1     Test Campaign 2019-01-08          3008                39550.0   35820.0
2  Control Campaign 2019-02-08          1757               121040.0  102513.0
3     Test Campaign 2019-02-08          2542               100719.0   91236.0
4  Control Campaign 2019-03-08          2343               131711.0  110862.0

   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          8110.0             2033.0          1841.0         1219.0      511.0
3          4657.0             2359.0          1548.0          879.0      677.0
4          6508.0             1737.0          1549.0         1134.0      372.0
```

[15]:

```python
df.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     datetime64[ns]
 2   Amount Spend           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: datetime64[ns](1), float64(7), int64(1), object(1)
memory usage: 4.8+ KB
```

- **Step 5:** Lastly, we can check the descriptive statistics of the data with "**.describe()**" function and check for outliers too. Here, we need no such task to fix outliers as it seems the data is perfectly fine.

[16]:

```python
df.describe()
```

[16]:

```
       Amount Spend  Number of Impressions         Reach  Website Clicks  \
count     60.000000              60.000000     60.000000       60.000000
mean    2425.750000           92072.279310  71168.248851     5676.563218
std      381.130461           32270.541283  30847.039691     1740.469866
min     1757.000000           22521.000000  10598.000000     2277.000000
```

```
25%     2073.750000          69558.250000   43235.500000     4230.750000
50%     2420.500000          98281.000000   77422.000000     5581.000000
75%     2727.500000         117160.500000   95314.250000     7201.250000
max     3112.000000         145248.000000  127852.000000     8264.000000

       Searches Received  Content Viewed  Added to Cart   Purchases
count          60.000000       60.000000      60.000000   60.000000
mean         2320.138506     1900.896552    1090.766667  522.013218
std           663.473391      681.437956     427.427479  195.297540
min          1001.000000      848.000000     278.000000  222.000000
25%          1970.750000     1249.000000     863.250000  340.000000
50%          2374.500000     1959.396552    1082.500000  506.000000
75%          2755.750000     2422.500000    1384.250000  685.000000
max          4891.000000     4219.000000    1913.000000  890.000000
```

4 AB Testing to find best marketing strategy:

4.0.1 Calculation of metrics needed:

[17]:
```
#Click Through Rate:
df['CTR'] = (df['Website Clicks'] / df['Number of Impressions']) * 100
#Cost Per Click:
df['CPC'] = df['Amount Spend'] / df['Website Clicks']
#Conversion Rate:
df['conversion_rate'] = (df['Purchases'] / df['Website Clicks']) * 100
#Cost Per Conversion:
df['cost_per_conversion'] = df['Amount Spend'] / df['Purchases']
#Added to Cart Rate:
df['ACR'] = (df['Added to Cart'] / df['Website Clicks']) * 100
#Cost Per Impressions:
df['CPM'] = (df['Amount Spend'] / df['Number of Impressions']) * 1000
```

[18]:
```
df.head()
```

[18]:
```
      Campaign Name       Date  Amount Spend  Number of Impressions     Reach  \
0  Control Campaign 2019-01-08          2280                82702.0   56930.0
1     Test Campaign 2019-01-08          3008                39550.0   35820.0
2  Control Campaign 2019-02-08          1757               121040.0  102513.0
3     Test Campaign 2019-02-08          2542               100719.0   91236.0
4  Control Campaign 2019-03-08          2343               131711.0  110862.0

   Website Clicks  Searches Received  Content Viewed  Added to Cart  \
0          7016.0             2290.0          2159.0         1819.0
1          3038.0             1946.0          1069.0          894.0
2          8110.0             2033.0          1841.0         1219.0
3          4657.0             2359.0          1548.0          879.0
4          6508.0             1737.0          1549.0         1134.0
```

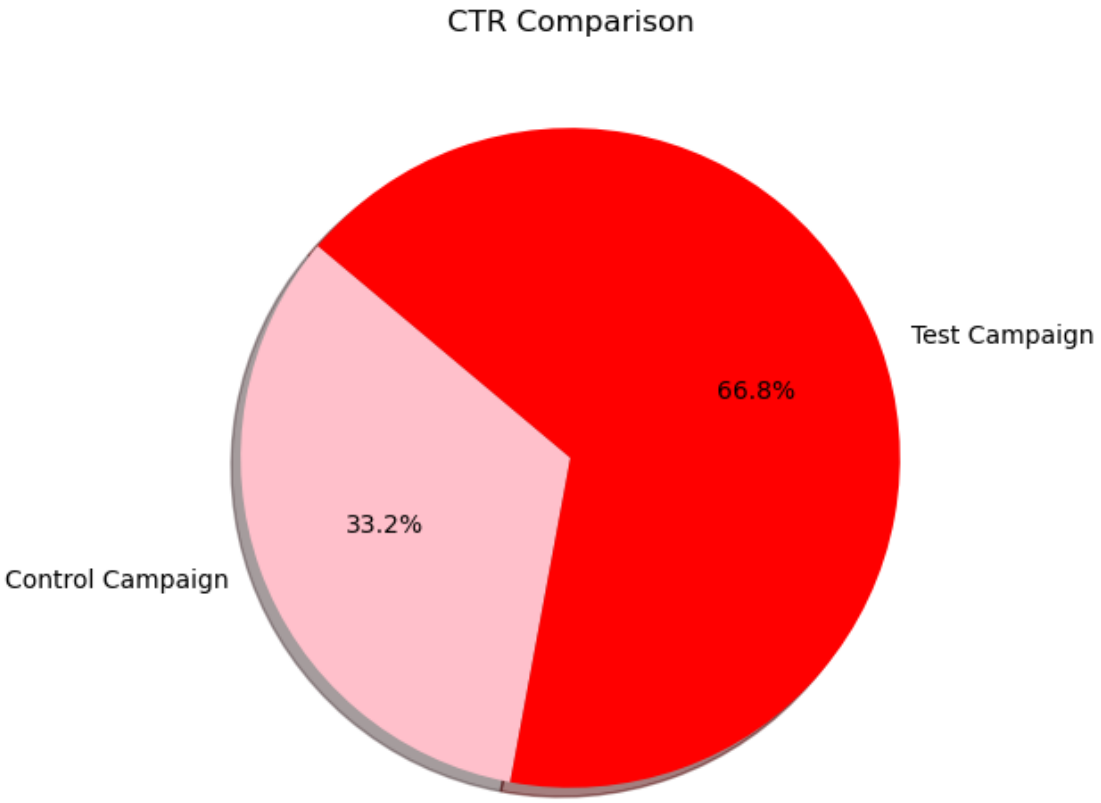
```
   Purchases       CTR       CPC  conversion_rate  cost_per_conversion  \
0      618.0  8.483471  0.324971         8.808438             3.689320
1      255.0  7.681416  0.990125         8.393680            11.796078
2      511.0  6.700264  0.216646         6.300863             3.438356
3      677.0  4.623755  0.545845        14.537256             3.754801
4      372.0  4.941121  0.360018         5.716042             6.298387

         ACR        CPM
0  25.926454  27.568862
1  29.427255  76.055626
2  15.030826  14.515863
3  18.874812  25.238535
4  17.424708  17.788947
```

4.0.2 1. Let's calculate the average Click Through Rate, grouping it by the type of ad campaigns. And, find out which campaign performs better.

[19]:
```
campaign_ctr = df.groupby('Campaign Name')['CTR'].mean().reset_index()
campaign_ctr
```

[19]:
```
      Campaign Name        CTR
0  Control Campaign   5.087893
1     Test Campaign  10.242260
```

[20]:
```
values = campaign_ctr['CTR']
labels = campaign_ctr['Campaign Name']
colors = ['Pink','Red']
plt.figure(figsize=(10,6))
plt.pie(values, labels = labels, colors = colors, autopct='%1.1f%%', shadow=␣
 ↪True, startangle= 140)
plt.title('CTR Comparison')
plt.show()
```



4.0.3 Explanation:

Here, we clearly see that the test campaign is performing better than control campaign. Hence, the test campaign seems to be more appealing to the customers and working better with higher engagement. - Test Campaign - 66.8% - Control Campaign - 33.2%

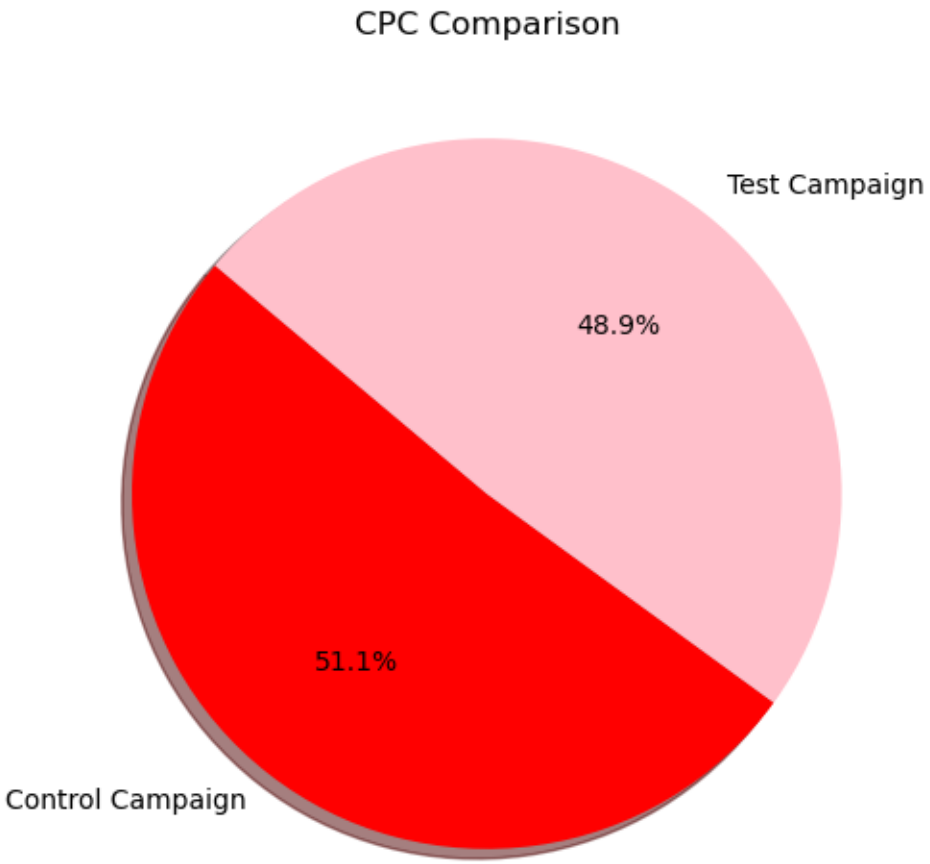
4.0.4 2. Now, let's calculate the average cost per click (CPC) for each ad campaign. And find out, which campaign costs less per click.

[21]:
```
campaign_cpc = df.groupby('Campaign Name')['CPC'].mean().reset_index()
campaign_cpc
```

[21]:
```
       Campaign Name       CPC
0  Control Campaign  0.489907
1     Test Campaign  0.468718
```

[22]:
```
values = campaign_cpc['CPC']
labels = campaign_cpc['Campaign Name']
colors = ['Red','Pink']
```

```
plt.figure(figsize=(10,6))
plt.pie(values, labels = labels, colors = colors, autopct='%1.1f%%', shadow=␣
 ↪True, startangle= 140)
plt.title('CPC Comparison')
plt.show()
```



4.0.5 Explanation:

In the above graph, we can see that both the ad campaign costs more or less the same amount per click. But, control campaign is a slightly costlier than the test campaign for 1 click. So, with low CPC, test campaign seems good for the business.

- Test Campaign - 48.9%
- Control Campaign - 51.1%

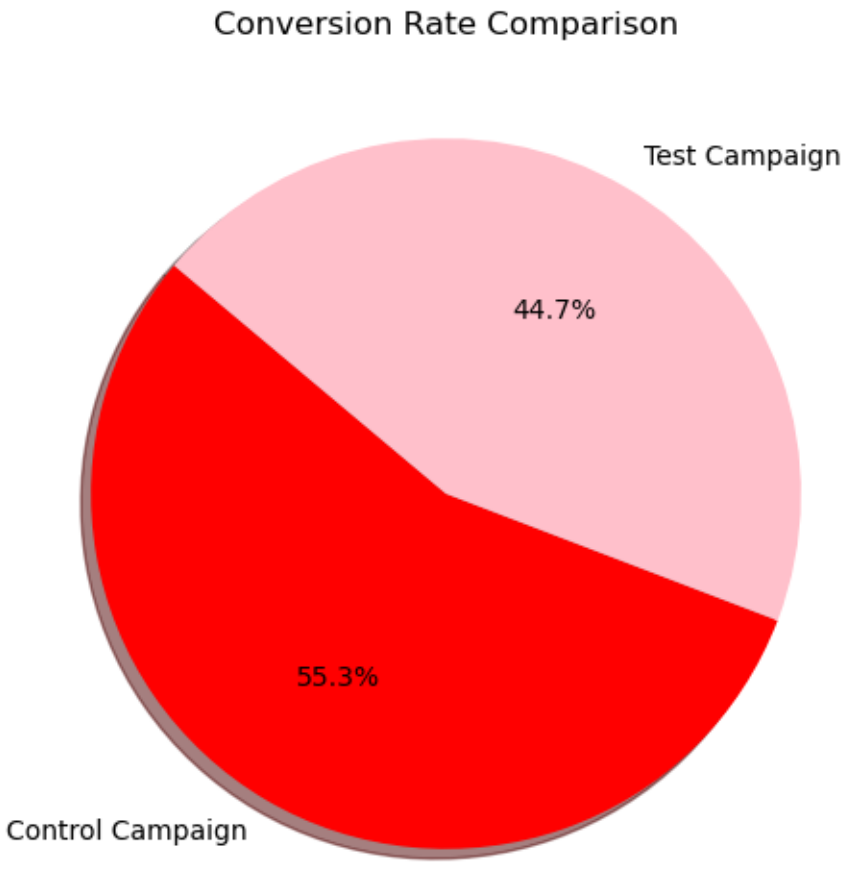
4.0.6 3. Now, Let's compare the conversion rate of both the ad campaigns and find out, which campaign is more on the right path towards the goal.

[23]:
```
campaign_conversion = df.groupby('Campaign Name')['conversion_rate'].mean().
 ↪reset_index()
campaign_conversion
```

[23]:

	Campaign Name	conversion_rate
0	Control Campaign	11.422146
1	Test Campaign	9.231182

[24]:
```
values = campaign_conversion['conversion_rate']
labels = campaign_conversion['Campaign Name']
colors = ['Red','Pink']
plt.figure(figsize=(10,6))
plt.pie(values, labels = labels, colors = colors, autopct='%1.1f%%', shadow=␣
 ↪True, startangle= 140)
plt.title('Conversion Rate Comparison')
plt.show()
```



4.0.7 Explanation:

Here, even though the CTR of control campaign was low, their conversion rate seems to be performing better than the test campaign.

- Test Campaign - 44.7%
- Control Campaign - 55.3%

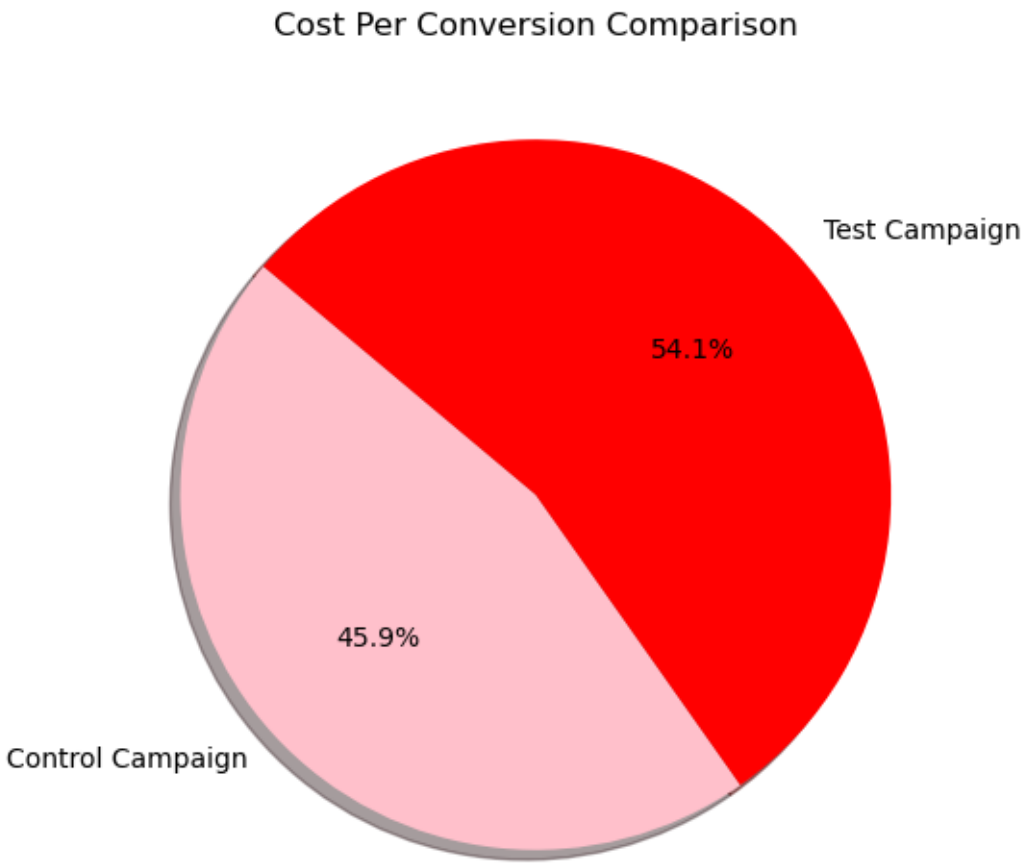
4.0.8 Let's calculate the average amount paid for each conversion of visiting customers to an actual customer of both the campaigns. And, find out which costs less between the two.

[25]:
```
campaign_cost_of_conversion = df.groupby('Campaign␣
 ↪Name')['cost_per_conversion'].mean().reset_index()
campaign_cost_of_conversion
```

[25]:

	Campaign Name	cost_per_conversion
0	Control Campaign	5.000927
1	Test Campaign	5.899589

[26]:
```
values = campaign_cost_of_conversion['cost_per_conversion']
labels = campaign_cost_of_conversion['Campaign Name']
colors = ['Pink','Red']
plt.figure(figsize=(10,6))
plt.pie(values, labels = labels, colors = colors, autopct='%1.1f%%', shadow=␣
 ↪True, startangle= 140)
plt.title('Cost Per Conversion Comparison')
plt.show()
```



4.0.9 Explanation:

In the above graph, It shows that test campaign costs more than control campaign to secure a customer. So, in this case, Control campaign seems better than test campaign.

- Test Campaign - 54.1%
- Control Campaign - 45.9%

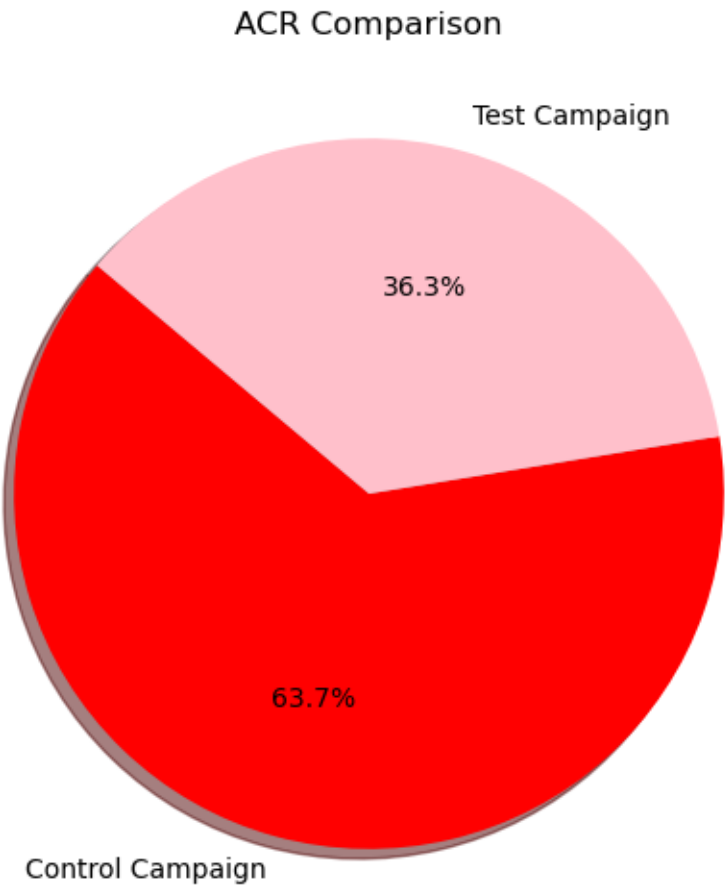
4.0.10 Now, we can compare the rate at which visiting customers add atleast one product to their cart. Let's find out which campaign shows better ACR.

[27]:
```
campaign_acr = df.groupby('Campaign Name')['ACR'].mean().reset_index()
campaign_acr
```

[27]:

	Campaign Name	ACR
0	Control Campaign	27.707909
1	Test Campaign	15.791233

[28]:
```
values = campaign_acr['ACR']
labels = campaign_acr['Campaign Name']
colors = ['Red','Pink']
plt.figure(figsize=(10,6))
plt.pie(values, labels = labels, colors = colors, autopct='%1.1f%%', shadow=␣
 ↪True, startangle= 140)
plt.title('ACR Comparison')
plt.show()
```



4.0.11 Explanation:

The graph shows that, the ACR of control campaign is far more better than test campaign. So, it tells that the control campaign is more appealing to the customers and test campaign needs some improvement.

- Test Campaign - 36.3%
- Control Campaign - 63.7%

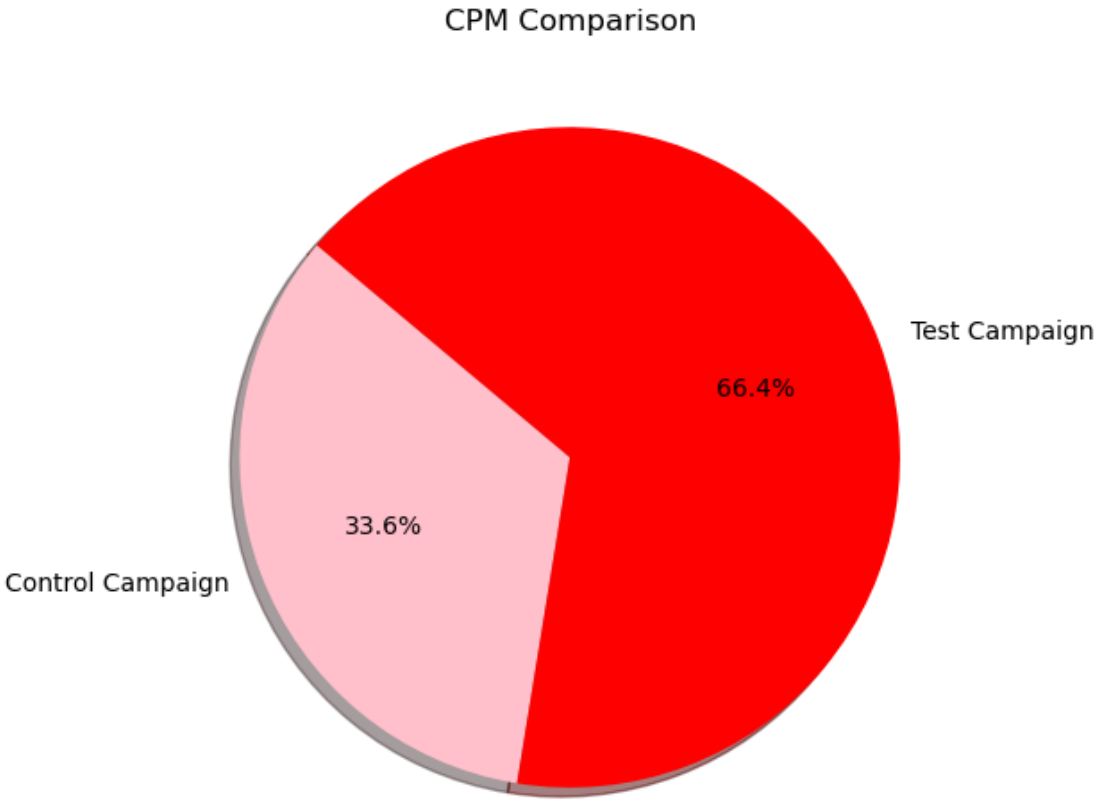
4.0.12 Lastly, let's study the Cost per 1000 impressions of both the ad campaigns.

[29]:
```
campaign_cpm = df.groupby('Campaign Name')['CPM'].mean().reset_index()
campaign_cpm
```

[29]:

	Campaign Name	CPM
0	Control Campaign	21.550149
1	Test Campaign	42.681041

[30]:
```
values = campaign_cpm['CPM']
labels = campaign_cpm['Campaign Name']
colors = ['Pink','Red']
plt.figure(figsize=(10,6))
plt.pie(values, labels = labels, colors = colors, autopct='%1.1f%%', shadow=␣
↪True, startangle= 140)
plt.title('CPM Comparison')
plt.show()
```



4.0.13 Explanation:

Here, it shows that test campaign costs the business far more for 1000 impressions each than the control campaign. So, control campaign shows better result in this metric.

- Test Campaign - 66.4%
- Control Campaign - 33.6%

5 Additional Analysis:

5.0.1 Let's study "Reach" of both the campaigns:

[31]:
```
campaigns = df['Campaign Name'].unique()
campaigns
```

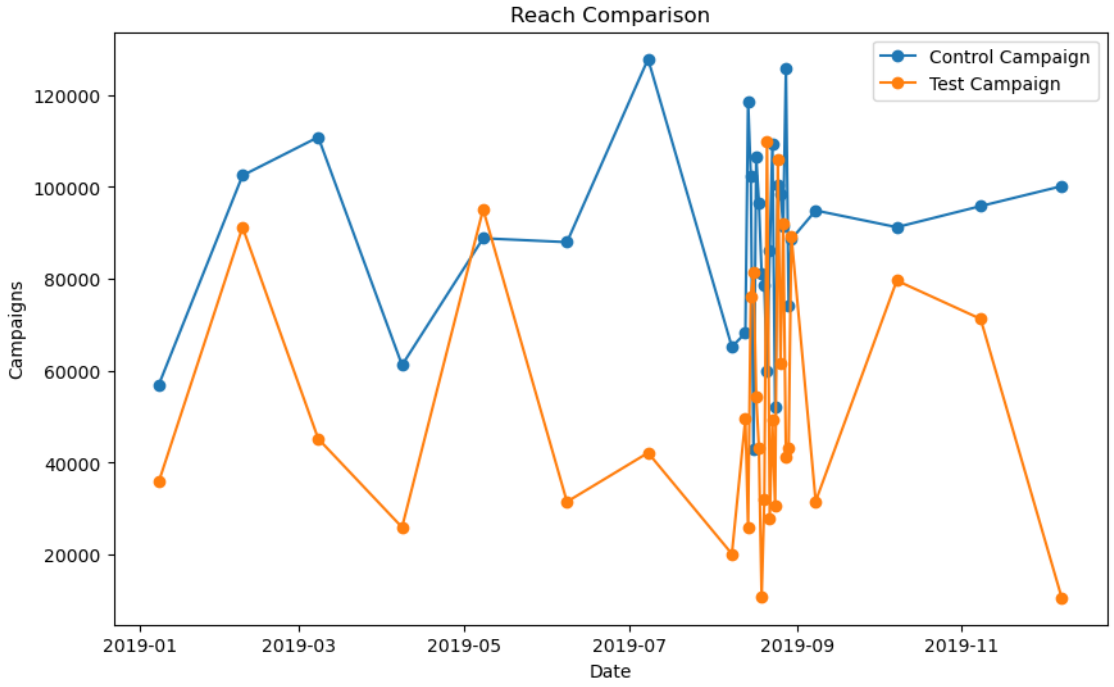
[31]:
```
array(['Control Campaign', 'Test Campaign'], dtype=object)
```

[32]:
```
campaign_reach = df.groupby('Campaign Name')['Reach'].mean().reset_index()
campaign_reach
```

[32]:

	Campaign Name	Reach
0	Control Campaign	88844.931034
1	Test Campaign	53491.566667

[33]:
```
plt.figure(figsize=(10,6))

for campaign in campaigns:
    campaign_data = df[df['Campaign Name'] == campaign]
    plt.plot(campaign_data['Date'], campaign_data['Reach'], marker = 'o',␣
 ↪linestyle = '-', label = f'{campaign}')

plt.xlabel('Date')
plt.ylabel('Campaigns')
plt.title("Reach Comparison")

plt.legend()
plt.show()
```



5.0.2 Explanation:

After the analysis, we find out that control campaign reaches more customers than the test campaign.

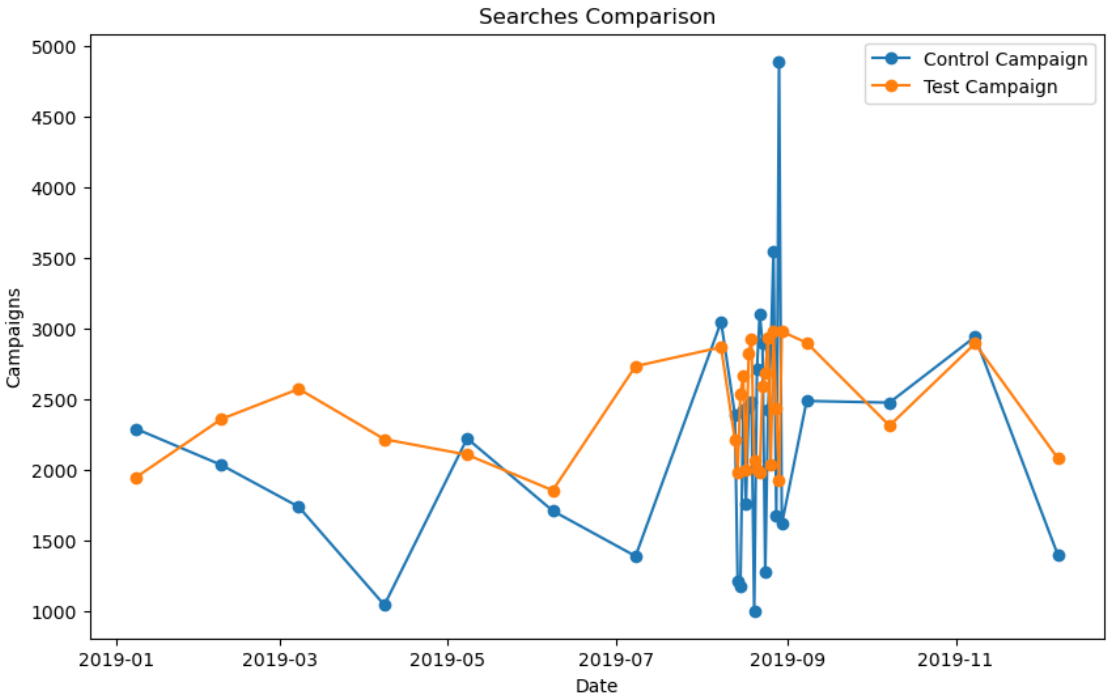
5.0.3 Let's study "No. of Searches Received" of both the campaigns:

[34]:
```
campaign_searches = df.groupby('Campaign Name')['Searches Received'].mean().
 ↪reset_index()
campaign_searches
```

[34]:
```
       Campaign Name  Searches Received
0  Control Campaign        2221.310345
1     Test Campaign        2418.966667
```

[35]:
```
plt.figure(figsize=(10,6))

for campaign in campaigns:
    campaign_data = df[df['Campaign Name'] == campaign]
    plt.plot(campaign_data['Date'], campaign_data['Searches Received'], marker␣
 ↪= 'o', linestyle = '-', label = f'{campaign}')

plt.xlabel('Date')
plt.ylabel('Campaigns')
plt.title("Searches Comparison")
```

```
plt.legend()
plt.show()
```



5.0.4 Explanation:

Both campaigns perform more or less same, in case of the "No. of Searches the campaigns received"

5.0.5 Let's study "No. of times the content is viewed" of both the campaigns:

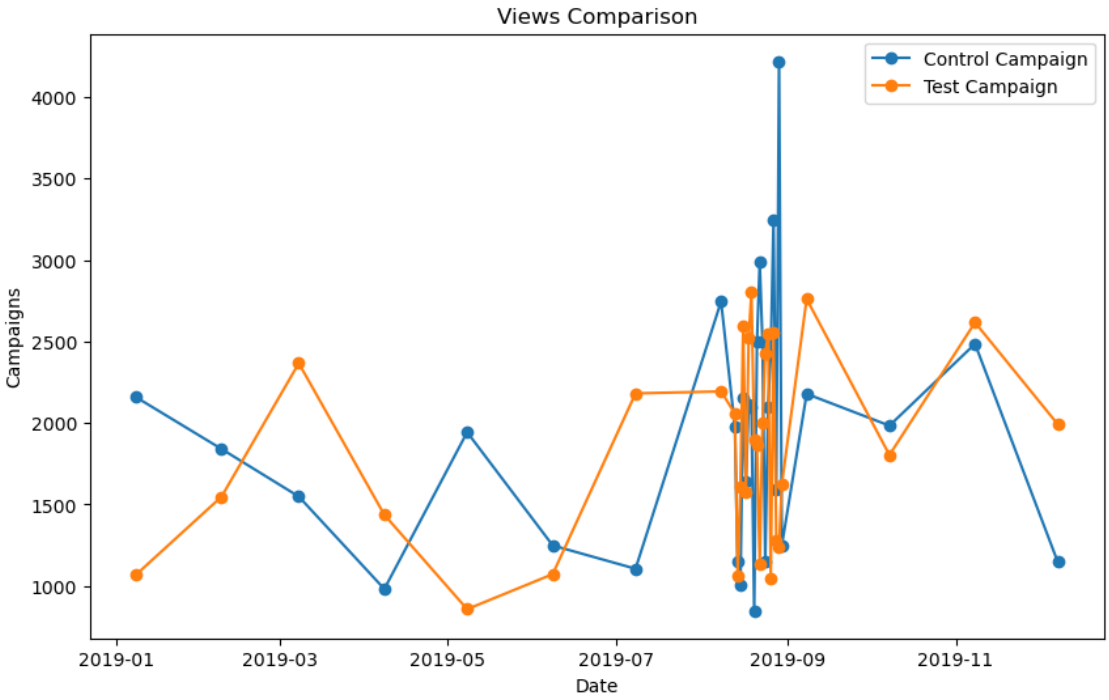
[36]:
```
campaign_views = df.groupby('Campaign Name')['Content Viewed'].mean().
 ↪reset_index()
campaign_views
```

[36]:
```
       Campaign Name  Content Viewed
0  Control Campaign     1943.793103
1     Test Campaign     1858.000000
```

[37]:
```
plt.figure(figsize=(10,6))

for campaign in campaigns:
    campaign_data = df[df['Campaign Name'] == campaign]
    plt.plot(campaign_data['Date'], campaign_data['Content Viewed'], marker =␣
 ↪'o', linestyle = '-', label = f'{campaign}')
```

```
plt.xlabel('Date')
plt.ylabel('Campaigns')
plt.title("Views Comparison")

plt.legend()
```

[37]: <matplotlib.legend.Legend at 0x7fd033a33a00>



5.0.6 Explanation:

Here too, It performs more or less the same through both the campaigns.

6 Overall Conclusion:

6.0.1 Insights:

- Control campaign has a lower average Click-Through Rate (CTR) compared to the Test campaign, indicating weaker engagement.
- Control campaign incurs a higher average Cost per Click (CPC) than the Test campaign, suggesting that it's more expensive to drive clicks in the Control group.
- Control campaign exhibits a higher average Conversion Rate compared to the Test campaign, indicating a better rate of turning clicks into purchases.

- Test campaign has a higher average Cost per Conversion compared to the Control campaign, implying that conversions are more costly in the Test group.
- Control campaign shows a higher average Add to Cart Rate compared to the Test campaign, indicating better engagement in adding products to the cart.
- Test campaign has a higher average Cost per Impressions (CPM) compared to the Control campaign, suggesting higher costs for reaching a thousand impressions.

6.0.2 Summary:

- Control campaign has lower CTR but better conversion rates and lower CPC.
- Test campaign incurs higher costs per conversion and impression but lower CTR and Add to Cart rates.

6.0.3 Which Campaign is better?

Control Campaign:

- Higher Conversion Rate, indicating better at turning clicks into purchases.
- Lower CPC, suggesting cost-effective clicks.
- Higher Add to Cart Rate, indicating better engagement in adding products to the cart.

Test Campaign:

- Higher CTR, indicating better engagement at the click-through stage.
- Higher Cost per Conversion, suggesting conversions are more costly.
- Higher Cost per Impressions (CPM), implying higher costs for reaching impressions.

6.0.4 The choice of the "better" campaign depends on the company's primary goals:

- If the company's main goal is to **maximize conversions** and **cost-efficiency** in the conversion process, the **Control Campaign may be considered better** due to its higher conversion rate and lower CPC.
- If the company's primary focus is on **generating initial user interest** and **engagement** (e.g., brand awareness), the **Test Campaign with its higher CTR might be preferred**.
- Lasly, If the company has specific budget constraints, it may need to evaluate which campaign provides a better return on investment (ROI) based on the cost per conversion and overall spend.

7 THANK YOU