Startup Product Analysis

Project Description

We are a venture capital company in Silicon Valley. Our job is to investigate user behavior for the company's app.
We want to know how users find our app and what they do with the app once they first visit their page. Most of the time users actually make a visit the page, then hang out and look at previous pages and after a lapse it completes a purchase.
Our job is to find out how long it takes users to make a purchase, what are the factors that delay the first activity app, but the averages can phase the users from the app through remarketing. Then decide to make a decision based on the results of a A/B test.
The point we'll pick after the groups are created groups that get the first time and see what groups get the later one and all the current up to the last one.
We want to know how long it takes users to make a purchase, what are the factors that delay the first activity app, but the averages can phase the users from the app through remarketing. Then decide to make a decision based on the results of a A/B test.
Our job is to find out how long it takes users to make a purchase, what are the factors that delay the first activity app, but the averages can phase the users from the app through remarketing. Then decide to make a decision based on the results of a A/B test.
The point we'll pick after the groups are created groups that get the first time and see what groups get the later one and all the current up to the last one.
We want to know how long it takes users to make a purchase, what are the factors that delay the first activity app, but the averages can phase the users from the app through remarketing. Then decide to make a decision based on the results of a A/B test.
We want to know how long it takes users to make a purchase, what are the factors that delay the first activity app, but the averages can phase the users from the app through remarketing. Then decide to make a decision based on the results of a A/B test.
The point we'll pick after the groups are created groups that get the first time and see what groups get the later one and all the current up to the last one.
We want to know how long it takes users to make a purchase, what are the factors that delay the first activity app, but the averages can phase the users from the app through remarketing. Then decide to make a decision based on the results of a A/B test.

Data Description

Each log entry is a user action or an event.

- EventName – event name
- DeviceIDHash – unique user identifier
- EventTimestamp – time of event
- ExpId – experiment number (A and UF are the control groups, 246 is the test group

Install Packages

If needed run code on the shell in spider/jupyter packages.

```
# !pip install plotly
# !pip install scipy
# !pip install seaborn
# !pip install numpy
```

Import Packages

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats as st
import math as mt
from plotly.subplots import make_subplots
import plotly.offline as offline
offline.init_notebook_mode()

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')
print('Libraries imported.')
```

```
Libraries imported.
```

Open the data file and read the general information

Create a function for initial exploration of the data

```
# Function to read and explore the data
def explore_data(path, sep=None):
    try:
        if sep:
            df = pd.read_csv('/datasets/logs_exp_us.csv', sep=sep)
        else:
            df = pd.read_csv('/datasets/logs_exp_us.csv')
    except:
        if sep:
            df = pd.read_csv('logs_exp_us.csv', sep=sep)
        else:
            df = pd.read_csv('logs_exp_us.csv')
    # Check the first few rows
    print("First 5 rows:")
    display(df.head())
    # Check the data types and non-null counts
    print("\nDataset info:")
    df.info()
    # Check for duplicate rows
    print("\nDuplicate rows:", df.duplicated().sum())
    # Check for missing values in each column
    print("\nMissing values:")
    display(df.isna().sum())
    # Check the shape of the dataset
    print("\nDataset shape:", df.shape)
    # Check the first few unique values in each column
    for col in df.columns:
        print(f"\nUnique values in {col}:")
        display(df[col].unique()[:5])
    return df

# Apply the function to your data
df = explore_data('/datasets/logs_exp_us.csv', sep='\t')
```

```
Loading file: logs_exp_us.csv
First 5 rows:
   EventName  DeviceIDHash  EventTimestamp  ExpId
0  MainScreen  4575588528974610257  1564029816  246
1  MainScreen  7416695313311560658  1564029826  246
2  PaymentScreenShown  3518123091307005509  1564029843  246
3  MainScreen  3267617425794627368  1564029854  246
4  PaymentScreenShown  6888746892508752  1564029875  246

Dataset info:
<class'pandas.core.frame.DataFrame'>
RangeIndex: 244126 entries, 0 to 244125
Data columns (total 4 columns):
 #   EventName       244126 non-null  object
 #   DeviceIDHash    244126 non-null  int64
 #   EventTimestamp  244126 non-null  int64
 #   ExpId           244126 non-null  int64
dtypes: int64(3), object(1)
memory usage: 7.5+ MB

Duplicate rows: 0

Missing values:
EventName       0
DeviceIDHash    0
EventTimestamp  0
ExpId           0
dtype: int64

Dataset shape: (244126, 4)

Unique values in EventName:
array(['MainScreen', 'PaymentScreenShown', 'CartScreenAppear',
       'OffersScreenAppear', 'Tutorial'], dtype=object)

Unique values in DeviceIDHash:
array([4575588528974610257, 7416695313311560658,
       3518123091307005509, 3267617425794627368,
       6888746892508752])

Unique values in EventTimestamp:
array([1564029816, 1564029826, 1564029843, 1564029854, 1564029875])

Unique values in ExpId:
array([246, 247, 248])
```

Clean the column names

```
# Clean column names
df.columns = df.columns.str.lower()
print("Cleaned column names:")
display(df.head())
```

```
Cleaned column names:
   eventname  deviceidhash  eventtimestamp  expid
0  MainScreen  4575588528974610257  1564029816  246
1  MainScreen  7416695313311560658  1564029826  246
2  PaymentScreenShown  3518123091307005509  1564029843  246
3  MainScreen  3267617425794627368  1564029854  246
4  PaymentScreenShown  6888746892508752  1564029875  246
```

Convert timestamp to datetime

```
# Convert timestamp to datetime
df['datetime'] = pd.to_datetime(df['eventtimestamp'], unit='s')
print("Added datetime column:")
display(df[['eventtimestamp', 'datetime']].head())
```

```
Added datetime column:
   eventtimestamp  datetime
0  1564029816  2019-07-25 04:43:36
1  1564029826  2019-07-25 04:43:46
2  1564029843  2019-07-25 04:44:03
3  1564029854  2019-07-25 04:44:14
4  1564029875  2019-07-25 04:44:35
```

Check for duplicate rows

```
# Check for duplicate rows
print("Duplicate rows:", df.duplicated().sum())
```

```
Duplicate rows: 0
```

Check for missing values

```
# Check for missing values
print("Missing values:")
display(df.isna().sum())
```

```
Missing values:
eventname       0
deviceidhash    0
eventtimestamp  0
expid           0
datetime        0
dtype: int64
```

Check the shape of the dataset

```
# Check the shape of the dataset
print("Dataset shape:", df.shape)
```

```
Dataset shape: (244126, 5)
```

Check unique values in each column

```
# Check unique values in each column
for col in df.columns:
    print(f"\nUnique values in {col}:")
    display(df[col].unique()[:5])
```

```
Unique values in eventname:
array(['MainScreen', 'PaymentScreenShown', 'CartScreenAppear',
       'OffersScreenAppear', 'Tutorial'], dtype=object)

Unique values in deviceidhash:
array([4575588528974610257, 7416695313311560658,
       3518123091307005509, 3267617425794627368,
       6888746892508752])

Unique values in eventtimestamp:
array([1564029816, 1564029826, 1564029843, 1564029854, 1564029875])

Unique values in expid:
array([246, 247, 248])

Unique values in datetime:
array(['2019-07-25 04:43:36', '2019-07-25 04:43:46',
       '2019-07-25 04:44:03', '2019-07-25 04:44:14',
       '2019-07-25 04:44:35'], dtype=object)
```

Study and check the data

Check the events

```
# Check the total number of events
print("Total number of events:", len(df))
# Check how many unique users are there
print("Unique users:", df['deviceidhash'].nunique())
# Check the date range of the data
print("Date range:", df['datetime'].min(), "to", df['datetime'].max())
```

```
Total number of events: 244126
Unique users: 7551
Date range: 2019-07-25 04:43:36 to 2019-08-07 21:15:17
```

Check the number of users and the average number of events per user

```
# Calculate average events per user
events_per_user = df.groupby('deviceidhash').size.mean()
print("Average events per user:", events_per_user)
```

```
Average events per user: 32.33210438471867
```

Check the distribution of events per user

```
# Check the distribution of events per user
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.histplot(df.groupby('deviceidhash').size, bins=50)
plt.title('Distribution of Events per User')
plt.xlabel('Number of Events')
plt.ylabel('Number of Users')
plt.show()
```

We see that the most frequent distribution appears from around 1000 percent of the data, when the Tutorial event appears really early.

Check the first and last events of each user

```
# Check the first and last events for each user
first_events = df.groupby('deviceidhash')['datetime'].min()
last_events = df.groupby('deviceidhash')['datetime'].max()
print("First events:")
display(first_events.head())
print("\nLast events:")
display(last_events.head())
```

```
First events:
deviceidhash
6888746892508752    2019-07-25 04:44:35
6909561520679493    2019-07-31 14:07:02
6922444491712511    2019-07-26 17:48:28
7035691991604509    2019-08 01 15:17:10
7048327892343828    2019-08-03 18:33:13
Name: datetime, dtype: datetime64[ns]

Last events:
deviceidhash
6888746892508752    2019-08-07 21:15:17
6909561520679493    2019-08-07 21:07:12
6922444491712511    2019-07-27 17:48:28
7035691991604509    2019-08-07 18:15:00
7048327892343828    2019-08-03 18:33:13
Name: datetime, dtype: datetime64[ns]
```

Conclusion

We have checked our dataset. We got the following information:

- The dataset contains 244126 events and 7551 users.
- The average user performs about 32 events.
- The dataset focus approximate.

In the next step we want to find the name of the columns and filters name.
We want to know how long it takes users to make a purchase, what are the factors that delay the first activity app, but the averages can phase the users from the app through remarketing. Then decide to make a decision based on the results of a A/B test.

Prepare the data for analysis

```
# Check the current data types
print("Current data types:")
display(df.dtypes)
# Convert datetime if needed
df['datetime'] = pd.to_datetime(df['eventtimestamp'], unit='s')
# Check the new data types
print("\nNew data types:")
display(df.dtypes)
```

```
Current data types:
eventname       object
deviceidhash    int64
eventtimestamp  int64
expid           int64
datetime        datetime64[ns]
dtype: object

New data types:
eventname       object
deviceidhash    int64
eventtimestamp  int64
expid           int64
datetime        datetime64[ns]
dtype: object
```

```
# Check for any inconsistencies in event names
print("Unique event names:")
display(df['eventname'].value_counts())
# Check for any inconsistencies in experiment IDs
print("\nUnique experiment IDs:")
display(df['expid'].value_counts())
```

```
Unique event names:
eventname
MainScreen           119205
OffersScreenAppear    46333
CartScreenAppear      42731
PaymentScreenShown    34116
Tutorial               1140
Name: count, dtype: int64

Unique experiment IDs:
expid
248    85582
247    84563
246    78981
Name: count, dtype: int64
```

Conclusion

In the Supplies we checked the following steps:

- Review the columns name.
- Check the number of events and users. 244126 of events - seem a really small number we will drop it.
- Check the experiment number column and compare the number of events between the groups. We want to know how many users and the time.

In the chapter we prepared the data for the analysis, ensures and ensuring the data.

Study and check the data

Check the events

```
# Check the total number of events
print("Total number of events:", len(df))
# Check how many unique users are there
print("Unique users:", df['deviceidhash'].nunique())
# Check the date range of the data
print("Date range:", df['datetime'].min(), "to", df['datetime'].max())
# Check the distribution of events per user
events_per_user = df.groupby('deviceidhash').size.mean()
print("Average events per user:", events_per_user)
```

```
Total number of events: 244126
Unique users: 7551
Date range: 2019-07-25 04:43:36 to 2019-08-07 21:15:17
Average events per user: 32.33210438471867
```

Check the number of users and the average number of events per user

```
# Calculate average events per user
events_per_user = df.groupby('deviceidhash').size.mean()
print("Average events per user:", events_per_user)
```

```
Average events per user: 32.33210438471867
```

Check the distribution of events per user

```
# Check the distribution of events per user
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.histplot(df.groupby('deviceidhash').size, bins=50)
plt.title('Distribution of Events per User')
plt.xlabel('Number of Events')
plt.ylabel('Number of Users')
plt.show
```

We see that the most frequent distribution appears from around 1000 percent of the data, when the Tutorial event appears really early.

Check the first and last events of each user

```
# Check the first and last events for each user
first_events = df.groupby('deviceidhash')['datetime'].min()
last_events = df.groupby('deviceidhash')['datetime'].max()
print("First events:")
display(first_events.head())
print("\nLast events:")
display(last_events.head())
```

```
First events:
deviceidhash
6888746892508752    2019-07-25 04:44:35
6909561520679493    2019-07-31 14:07:02
6922444491712511    2019-07-26 17:48:28
7035691991604509    2019-08 01 15:17:10
7048327892343828    2019-08-03 18:33:13
Name: datetime, dtype: datetime64[ns]

Last events:
deviceidhash
6888746892508752    2019-08-07 21:15:17
6909561520679493    2019-08-07 21:07:12
6922444491712511    2019-07-27 17:48:28
7035691991604509    2019-08-07 18:15:00
7048327892343828    2019-08-03 18:33:13
Name: datetime, dtype: datetime64[ns]
```

Conclusion

We have checked our dataset. We got the following information:

- The dataset contains 244126 events and 7551 users.
- The average user performs about 32 events.
- The dataset focus approximate.

In the next step we want to find the name of the columns and filters name.
We want to know how long it takes users to make a purchase, what are the factors that delay the first activity app, but the averages can phase the users from the app through remarketing. Then decide to make a decision based on the results of a A/B test.

Prepare the data for analysis

```
# Check the current data types
print("Current data types:")
display(df.dtypes)
# Convert datetime if needed
df['datetime'] = pd.to_datetime(df['eventtimestamp', unit='s')
# Check the new data types
print("\nNew data types:")
display(df.dtypes)
```

```
Current data types:
eventname       object
deviceidhash    int64
eventtimestamp  int64
expid           int64
datetime        datetime64[ns]
dtype: object

New data types:
eventname       object
deviceidhash    int64
eventtimestamp  int64
expid           int64
datetime        datetime64[ns]
dtype: object
```

```
# Check for any inconsistencies in event names
print("Unique event names:")
display(df['eventname'].value_counts)
# Check for any inconsistencies in experiment IDs
print("\nUnique experiment IDs:")
display(df['expid'].value_counts)
```

```
Unique event names:
eventname
MainScreen           119205
OffersScreenAppear    46333
CartScreenAppear      42731
PaymentScreenShown    34116
Tutorial               1140
Name: count, dtype: int64

Unique experiment IDs:
expid
248    85582
247    84563
246    78981
Name: count, dtype: int64
```

Conclusion

In the Supplies we checked the following steps:

- Review the columns name.
- Check the number of events and users. 244126 of events - seem a really small number we will drop it.
- Check the experiment number column and compare the number of events between the groups. We want to know how many users and the time.

In the chapter we prepared the data for the analysis, ensures and ensuring the data.

Study and check the data

Check the events

```
# Check the total number of events
print("Total number of events:", len(df))
# Check how many unique users are there
print("Unique users:", df['deviceidhash'].nunique()
# Check the date range of the data
print("Date range:", df['datetime'].min(), "to", df['datetime'].max()
# Check the distribution of events per user
events_per_user = df.groupby('deviceidhash').size.mean()
print("Average events per user:", events_per_user)
```

```
Total number of events: 244126
Unique users: 7551
Date range: 2019-07-25 04:43:36 to 2019-08-07 21:15:17
Average events per user: 32.33210438471867
```

Check the number of users and the average number of events per user

```
# Calculate average events per user
events_per_user = df.groupby('deviceidhash').size.mean()
print("Average events per user:", events_per_user)
```

```
Average events per user: 32.33210438471867
```

Check the distribution of events per user

```
# Check the distribution of events per user
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.histplot(df.groupby('deviceidhash').size, bins=50)
plt.title('Distribution of Events per User')
plt.xlabel('Number of Events')
plt.ylabel('Number of Users')
plt.show
```

We see that the most frequent distribution appears from around 1000 percent of the data, when the Tutorial event appears really early.

Check the first and last events of each user

```
# Check the first and last events for each user
first_events = df.groupby('deviceidhash')['datetime'].min()
last_events = df.groupby('deviceidhash')['datetime'].max()
print("First events:")
display(first_events.head()
print("\nLast events:")
display(last_events.head()
```

```
First events:
deviceidhash
6888746892508752    2019-07-25 04:44:35
6909561520679493    2019-07-31 14:07:02
6922444491712511    2019-07-26 17:48:28
7035691991604509    2019-08 01 15:17:10
7048327892343828    2019-08-03 18:33:13
Name: datetime, dtype: datetime64[ns]

Last events:
deviceidhash
6888746892508752    2019-08-07 21:15:17
6909561520679493    2019-08-07 21:07:12
6922444491712511    2019-07-27 17:48:28
7035691991604509    2019-08-07 18:15:00
7048327892343828    2019-08-03 18:33:13
Name: datetime, dtype: datetime64[ns]
```

Conclusion

We have checked our dataset. We got the following information:

- The dataset contains 244126 events and 7551 users.
- The average user performs about 32 events.
- The dataset focus approximate.

In the next step we want to find the name of the columns and filters name.
We want to know how long it takes users to make a purchase, what are the factors that delay the first activity app, but the averages can phase the users from the app through remarketing. Then decide to make a decision based on the results of a A/B test.

Study the results of the experiment

How many users are there in each group

```
# Check the number of users in each group
print("Number of users in each group:")
display(df.groupby('expid')['deviceidhash'].nunique()
```

```
Number of users in each group:
expid
246    2484
247    2513
248    2537
Name: deviceidhash, dtype: int64
```

We have two control groups in the A/A test, check to check our instruments and calculations. Not if there is statistically significant difference between samples 246 and 247. A/B test

Now we check the group we have for the experiment, the control groups and remove the control one to run part of the A/B test.

```
# Create a new column for control vs test groups
df['group'] = df['expid'].map({246: 'control', 247: 'control', 248: 'test'})
print("Group distribution:")
display(df['group'].value_counts)
```

```
Group distribution:
group
control    156563
test       85563
Name: count, dtype: int64
```

Calculate the number of users for each experiment

```
# Calculate number of users for each experiment
print("Number of users for each experiment:")
display(df.groupby('expid')['deviceidhash'].nunique()
```

```
Number of users for each experiment:
expid
246    2484
247    2513
248    2537
Name: deviceidhash, dtype: int64
```

Calculate the number of events for each experiment

```
# Calculate number of events for each experiment
print("Number of events for each experiment:")
display(df.groupby('expid')['eventname'].count()
```

```
Number of events for each experiment:
expid
246    78981
247    84563
248    85582
Name: eventname, dtype: int64
```

Calculate the number of events per user for each experiment

```
# Calculate events per user for each experiment
events_per_user = df.groupby('expid')['deviceidhash'].count() / df.groupby('expid')['deviceidhash'].nunique()
print("Average events per user:")
display(events_per_user)
```

```
Average events per user:
expid
246    31.799
247    33.651
248    33.734
dtype: float64
```

Conclusion

Compare the number of users

```
# Compare number of users
print("Number of users:")
display(df.groupby('expid')['deviceidhash'].nunique()
print("\nNumber of events:")
display(df.groupby('expid')['eventname'].count()
```

```
Number of users:
expid
246    2484
247    2513
248    2537
Name: deviceidhash, dtype: int64

Number of events:
expid
246    78981
247    84563
248    85582
Name: eventname, dtype: int64
```

Compare the number of events

```
# Compare number of events
print("Number of events per user:")
display(df.groupby('expid')['eventname'].count() / df.groupby('expid')['deviceidhash'].nunique()
```

```
Number of events per user:
expid
246    31.799
247    33.651
248    33.734
dtype: float64
```

Conclusion

Compare the number of users and events

```
# Compare number of users and events
print("Number of users:")
display(df.groupby('expid')['deviceidhash'].nunique()
print("\nNumber of events:")
display(df.groupby('expid')['eventname'].count()
print("\nNumber of events per user:")
display(df.groupby('expid')['eventname'].count() / df.groupby('expid')['deviceidhash'].nunique()
```

```
Number of users:
expid
246    2484
247    2513
248    2537
Name: deviceidhash, dtype: int64

Number of events:
expid
246    78981
247    84563
248    85582
Name: eventname, dtype: int64
```

```
Null Hypothesis: There is no statistically significant difference in conversion between the groups in the ad
Alternative Hypothesis: There is statistically significant difference in conversion between the groups in the
ad

The p-value we are for the test: 0.05
```

```
Computed between group SS: 0.0
```

```
1. perform hypothesis for the Mediocrebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

2. perform hypothesis for the Effectivebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

3. perform hypothesis for the Sentimentbadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

4. perform hypothesis for the Supportcommunitybadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:
```

Since the non-zero sum is not in the control group, that the number of users who performed this action. Find their more. Check whether the difference between the groups is extremely significant. Repeat the procedure for all other cases. 4.58. 4.58 > 0.05

Null hypothesis: There is no statistically significant difference in conversion between the groups

Alternative hypothesis: There is statistically significant difference in conversion between the groups

The p-value we are for the test: 0.05

Compare the control group to our test group

```
Null Hypothesis: There is no statistically significant difference in conversion between the groups
Alternative Hypothesis: There is statistically significant difference in conversion between the groups

The p-value we are for the test: 0.05
```

```
Computed between group SS: 4.58
```

```
1. perform hypothesis for the Mediocrebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

2. perform hypothesis for the Effectivebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

3. perform hypothesis for the Sentimentbadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

4. perform hypothesis for the Supportcommunitybadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:
```

```
Computed between group SS: 17.16
```

```
1. perform hypothesis for the Mediocrebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

2. perform hypothesis for the Effectivebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

3. perform hypothesis for the Sentimentbadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

4. perform hypothesis for the Supportcommunitybadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:
```

```
Computed between group SS: 17.16
```

```
1. perform hypothesis for the Mediocrebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

2. perform hypothesis for the Effectivebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

3. perform hypothesis for the Sentimentbadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

4. perform hypothesis for the Supportcommunitybadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:
```

Compare the results with the combined results for the control group

```
Computed between group SS: 17.16
```

```
# Compare the control group to the first and second ab tests:
# ANOVA
# Null Hypothesis: There is no statistically significant difference in conversion between the groups
# Alternative Hypothesis: There is statistically significant difference in conversion between the groups
# group_conversion=np.concat([df_test_group_conversion, df_test_group_conversion2])
# control_conversion=np.concat([df_control_group_conversion, df_control_group_conversion2])
# ss_total = np.sum((group_conversion - np.mean(group_conversion))**2)
# print("Computed between group SS:", ss_total)
# ss_between = 0
# for g in [test_group_conversion, control_group_conversion]:
#     ss_between += len(g) * (np.mean(g) - np.mean(group_conversion))**2
# print(ss_between)
# ss_within = ss_total - ss_between
# print("Computed within group SS:", ss_within)
# df_between = 1
# df_within = len(group_conversion) - 2
# print("Degrees of freedom between:", df_between, "Degrees of freedom within:", df_within)
# ms_between = ss_between / df_between
# ms_within = ss_within / df_within
# f_stat = ms_between / ms_within
# print("F-statistic:", f_stat)
# p_value = 1 - stats.f.cdf(f_stat, df_between, df_within)
# print("P-value:", p_value)
# if p_value < 0.05:
#     print("Reject the Null Hypothesis")
# else:
#     print("Fail to reject the Null Hypothesis")
```

```
Computed between group SS: 17.16
```

```
1. perform hypothesis for the Mediocrebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

2. perform hypothesis for the Effectivebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

3. perform hypothesis for the Sentimentbadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

4. perform hypothesis for the Supportcommunitybadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:
```

When significance test have you so far that the statistical hypotheses mentioned about? Calculate how many statistical hypothesis tests you carried out.

Since the control group and the test group are the same, the number of users who performed the action. Find their more. Check whether the difference between the groups is extremely significant. Repeat the procedure for all other cases. 17.16. 17.16 > 0.05

Null hypothesis: There is no statistically significant difference in conversion between the groups

Alternative hypothesis: There is statistically significant difference in conversion between the groups

The p-value we are for the test: 0.05

Compare the control group to our test group

```
Null Hypothesis: There is no statistically significant difference in conversion between the groups
Alternative Hypothesis: There is statistically significant difference in conversion between the groups

The p-value we are for the test: 0.05
```

```
Computed between group SS: 17.16
```

```
1. perform hypothesis for the Mediocrebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

2. perform hypothesis for the Effectivebadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

3. perform hypothesis for the Sentimentbadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:

4. perform hypothesis for the Supportcommunitybadge test:
Fobserved of variance in the first group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000
Fobserved of variance in the second group: 0.0010117180000

Pass/Fail values for this Null Hypothesis:
```

Overall Conclusion

After analyzed the data analysis and ended the Overall Hypothesis and experimented the data we came out with the following insights:

1. 40% percent of the users have other the Mediocrebadge event. Hence I want to improve the most screen of the app because other users are not interested in the app. Hence I want to improve the most screen of the app to upgrade/improve the UI.
2. Only 8% of the users have left all the events. There is a very low bounce rate.
3. If most of the events there is a statistically significant difference in conversion between the groups maybe it can show or that the users and the developers with that app is similar to the different groups or maybe there are other's experiment the group is good app.
4. Nearly small part of the users use the Sentiment or Support Community to express if or ignore the feature at all.