Assignment Title:

"Data Analysis and Insights for different page Optimization & How to get more user installation & Engagement from the App & Website" User and propose recommendations for improving performance"

Import required python libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Read the file

```
file ="App Analytics Report-06.05.2023.xlsx"
```

```
 excel = pd.ExcelFile(file)
```

```
excel
```

```
     <pandas.io.excel._base.ExcelFile at 0x27d299eb760>
```

```
excel.sheet_names
```

```
     ['Report Snapshot',
      'User Acquisition',
      'Traffic Aquisition',
```

```
'Event Report',
'Conversion Report',
'Pages & Screens Report',
'Retention Overview',
'User Engagement Overview',
'Demographics Report',
'Citiwise Report',
'Gender Report',
'User By Interest',
'User by Language',
'User By Age',
'Google Ads Report']
```

EDA- Exploratory data analysis

Comprehensive Analysis:

Channel Performance: The analysis shows that different marketing channels have varying levels of performance in terms of user engagement, conversion rates, and revenue generation. Organic Search and Display channels have higher engagement rates and conversions compared to Paid Search and Organic Social.

User Engagement: Users from different channels show different levels of engagement. For example, Direct and Organic Social channels have a lower number of engaged sessions per user, indicating a need to improve user engagement strategies for these channels.

Revenue Generation: The majority of revenue comes from Display and Organic Search channels, indicating their effectiveness in driving sales.

1.User Acquisition

```
User_Acqu = excel.parse('User Acquisition')
User_Acqu
```

	First user default channel group	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	Display	9957	12008	0.544457	1.206107	58.86209	204820	37434	0
1	Organic Search	7652	18141	0.813680	2.367041	534.31280	770710	109801	0
2	Paid Search	3025	4408	0.474284	1.458154	102.23780	81997	14770	0
3	Direct	1903	4975	0.318808	2.261364	1128.88100	227434	31093	0
4	Unassigned	325	1619	0.813159	4.981538	798.34150	33320	789	0
5	Organic Social	10	13	0.722222	1.300000	145.30000	248	27	0

```
User_Acqu.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 9 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   First user default channel group  6 non-null     object 
 1   New users                       6 non-null      int64  
 2   Engaged sessions                6 non-null      int64  
 3   Engagement rate                 6 non-null      float64
 4   Engaged sessions per user       6 non-null      float64
 5   Average engagement time         6 non-null      float64
 6   Event count                     6 non-null      int64  
 7   Conversions                     6 non-null      int64  
 8   Total revenue                   6 non-null      int64  
dtypes: float64(3), int64(5), object(1)
memory usage: 560.0+ bytes
```

```
User_Acqu.describe()
```

	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
count	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000	6.0
mean	3812.000000	6860.666667	0.614435	2.262367	461.322532	219754.833333	32319.000000	0.0
std	4083.739659	6894.604867	0.201367	1.420806	437.111008	284869.104021	40929.227185	0.0
min	10.000000	13.000000	0.318808	1.206107	58.862090	248.000000	27.000000	0.0
25%	719.500000	2316.250000	0.491827	1.339538	113.003350	45489.250000	4284.250000	0.0
50%	2464.000000	4691.500000	0.633339	1.859759	339.806400	143408.500000	22931.500000	0.0

```
User_Acqu.drop_duplicates()
```

	First user default channel group	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	Display	9957	12008	0.544457	1.206107	58.86209	204820	37434	0
1	Organic Search	7652	18141	0.813680	2.367041	534.31280	770710	109801	0
2	Paid Search	3025	4408	0.474284	1.458154	102.23780	81997	14770	0
3	Direct	1903	4975	0.318808	2.261364	1128.88100	227434	31093	0
4	Unassigned	325	1619	0.813159	4.981538	798.34150	33320	789	0
5	Organic Social	10	13	0.722222	1.300000	145.30000	248	27	0

2.Traffic Aquisition

```
Traffic_Aqu = excel.parse('Traffic Aquisition')
Traffic_Aqu
```

	Session default channel group	Users	Sessions	Engaged sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count	Conversions	Total revenue
0	Unassigned	20263	13448	1481	34.11704	0.073089	18.023130	0.110128	242375	114161	0
1	Display	9613	18292	10613	28.52198	1.104026	9.069320	0.580199	165896	20031	0
2	Organic Search	7689	21241	17814	195.94340	2.316816	29.302290	0.838661	622410	33612	0
3	Direct	4042	13220	7649	177.17060	1.892380	17.135850	0.578593	226536	18496	0
4	Paid Search	2909	6788	3452	36.65321	1.186662	8.989982	0.508544	61024	7595	0

```
Traffic_Aqu.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 11 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Session default channel group       6 non-null      object
 1   Users                               6 non-null      int64
 2   Sessions                            6 non-null      int64
 3   Engaged sessions                    6 non-null      int64
 4   Average engagement time per session  6 non-null      float64
 5   Engaged sessions per user           6 non-null      float64
 6   Events per session                  6 non-null      float64
 7   Engagement rate                     6 non-null      float64
 8   Event count                         6 non-null      int64
 9   Conversions                         6 non-null      int64
 10  Total revenue                       6 non-null      int64
dtypes: float64(4), int64(6), object(1)
memory usage: 656.0+ bytes
```

```
Traffic_Aqu.describe()
```

	Users	Sessions	Engaged sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count	Conversions	re
count	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000	6.000000	
mean	7421.166667	12167.500000	6836.833333	88.744788	1.277314	16.753429	0.561021	219754.833333	32319.000000	
std	7162.088199	7735.416104	6665.397735	76.756564	0.772272	7.484689	0.252797	218609.598013	41704.726979	
min	11.000000	16.000000	12.000000	28.521980	0.073089	8.989982	0.110128	288.000000	19.000000	
25%	3192.250000	8396.000000	1973.750000	34.751083	1.094188	11.085952	0.526056	87242.000000	10320.250000	
50%	5865.500000	13334.000000	5550.500000	48.357855	1.145344	17.567925	0.579396	196216.000000	19263.500000	
75%	9132.000000	17081.000000	9872.000000	147.893575	1.715950	18.017347	0.707550	238415.250000	30216.750000	
max	20263.000000	21241.000000	17814.000000	195.943400	2.316816	29.302290	0.838661	622410.000000	114161.000000	

```
Traffic_Aqu.drop_duplicates()
```

	Session default channel group	Users	Sessions	Engaged sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count	Conversions	Total revenue
0	Unassigned	20263	13448	1481	34.11704	0.073089	18.023130	0.110128	242375	114161	0
1	Display	9613	18292	10613	28.52198	1.104026	9.069320	0.580199	165896	20031	0
2	Organic Search	7689	21241	17814	195.94340	2.316816	29.302290	0.838661	622410	33612	0
3	Direct	4042	13220	7649	177.17060	1.892380	17.135850	0.578593	226536	18496	0
4	Paid Search	2909	6788	3452	36.65321	1.186662	8.989982	0.508544	61024	7595	0

```
User_Acqu.rename(columns={'First user default channel group': 'Session default channel group'}, inplace=True)

# Merge data frames based on the 'Session default channel group' column
merged_df = pd.merge(Traffic_Aqu, User_Acqu, on='Session default channel group')

# Display the merged data frame
#print(merged_df)
merged_df=pd.DataFrame(merged_df)
merged_df
```

	Session default channel group	Users	Sessions	Engaged sessions_x	Average engagement time per session	Engaged sessions per user_x	Events per session	Engagement rate_x	Event count_x	Conversions_x	Total revenue_x	u
0	Unassigned	20263	13448	1481	34.11704	0.073089	18.023130	0.110128	242375	114161	0	
1	Display	9613	18292	10613	28.52198	1.104026	9.069320	0.580199	165896	20031	0	
2	Organic Search	7689	21241	17814	195.94340	2.316816	29.302290	0.838661	622410	33612	0	
3	Direct	4042	13220	7649	177.17060	1.892380	17.135850	0.578593	226536	18496	0	
4	Paid Search	2909	6788	3452	36.65321	1.186662	8.989982	0.508544	61024	7595	0	
5	Organic Social	11	16	12	60.06250	1.090909	18.000000	0.750000	288	19	0	

```
merged_df.isnull().sum()
```

```
Session default channel group          0
Users                                  0
Sessions                               0
Engaged sessions_x                     0
Average engagement time per session    0
Engaged sessions per user_x            0
```

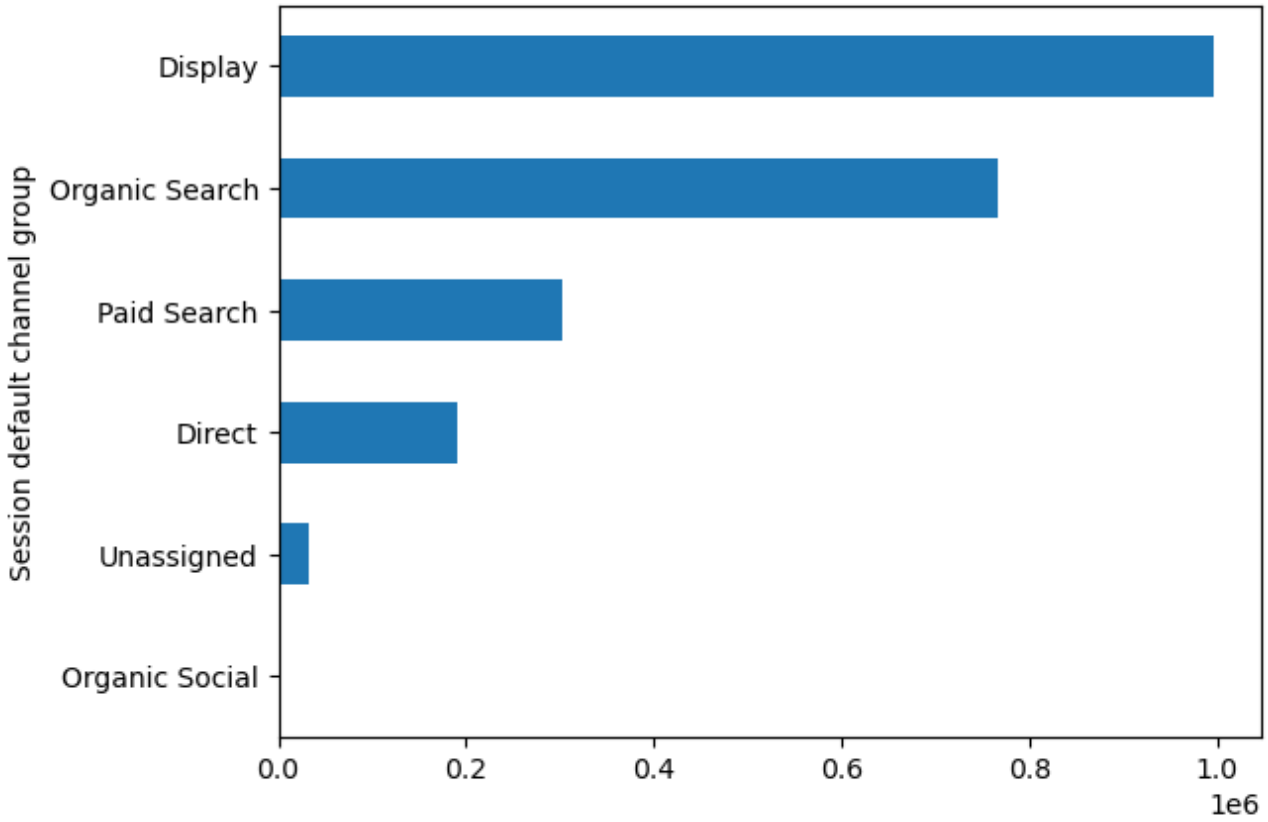
```
Events per session                  0
Engagement rate_x                   0
Event count_x                       0
Conversions_x                       0
Total revenue_x                     0
New users                           0
Engaged sessions_y                  0
Engagement rate_y                   0
Engaged sessions per user_y         0
Average engagement time             0
Event count_y                       0
Conversions_y                       0
Total revenue_y                     0
dtype: int64
```

```
merged_df.drop_duplicates()
```

	Session default channel group	Users	Sessions	Engaged sessions_x	Average engagement time per session	Engaged sessions per user_x	Events per session	Engagement rate_x	Event count_x	Conversions_x	Total revenue_x	u
0	Unassigned	20263	13448	1481	34.11704	0.073089	18.023130	0.110128	242375	114161	0	
1	Display	9613	18292	10613	28.52198	1.104026	9.069320	0.580199	165896	20031	0	
2	Organic Search	7689	21241	17814	195.94340	2.316816	29.302290	0.838661	622410	33612	0	
3	Direct	4042	13220	7649	177.17060	1.892380	17.135850	0.578593	226536	18496	0	
4	Paid Search	2909	6788	3452	36.65321	1.186662	8.989982	0.508544	61024	7595	0	
5	Organic Social	11	16	12	60.06250	1.090909	18.000000	0.750000	288	19	0	

```
(merged_df.groupby('Session default channel group')['New users'].mean()*100).sort_values().plot(kind = 'barh')
```

```
<AxesSubplot:ylabel='Session default channel group'>
```




```
(merged_df.groupby('Session default channel group')['Sessions'].mean()*100).sort_values().plot(kind = 'barh')
```

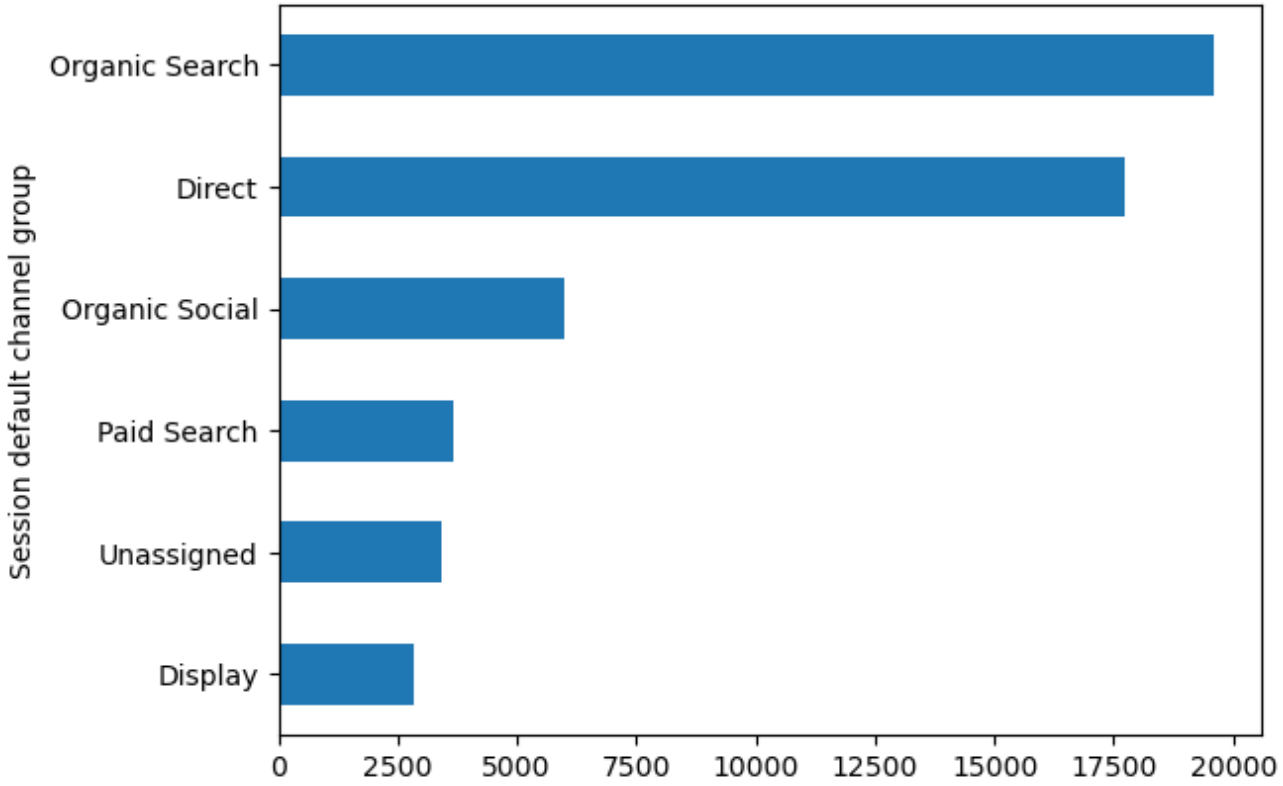
```
<AxesSubplot:ylabel='Session default channel group'>
```


```
(merged_df.groupby('Session default channel group')['Engaged sessions per user_x'].mean()*100).sort_values().plot(kind = 'barh')
```

```
<AxesSubplot:ylabel='Session default channel group'>
```

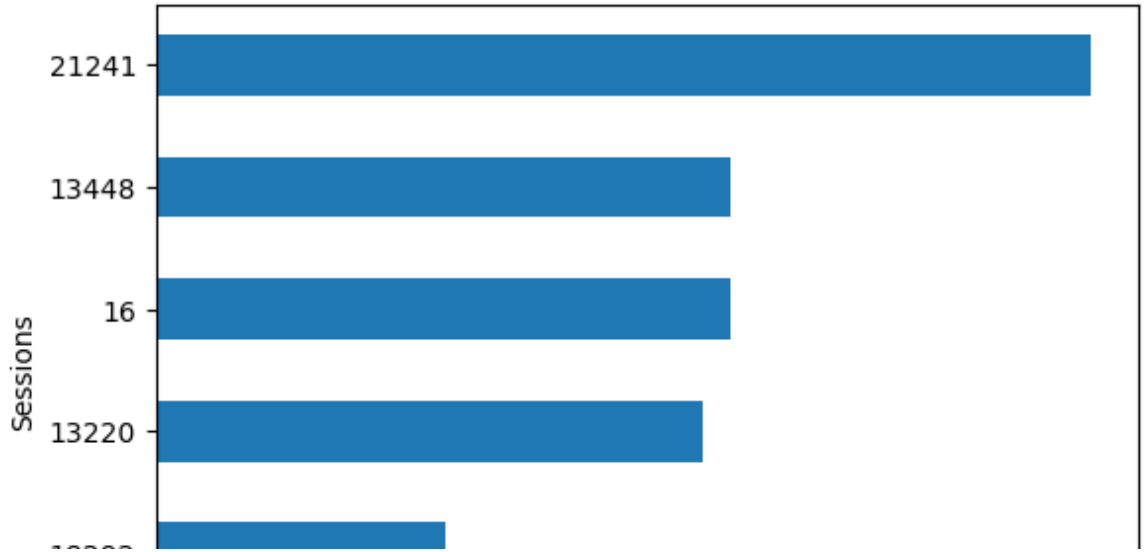
```
(merged_df.groupby('Session default channel group')['Average engagement time per session'].mean()*100).sort_values().plot(kind = 'barh'
```

```
<AxesSubplot:ylabel='Session default channel group'>
```

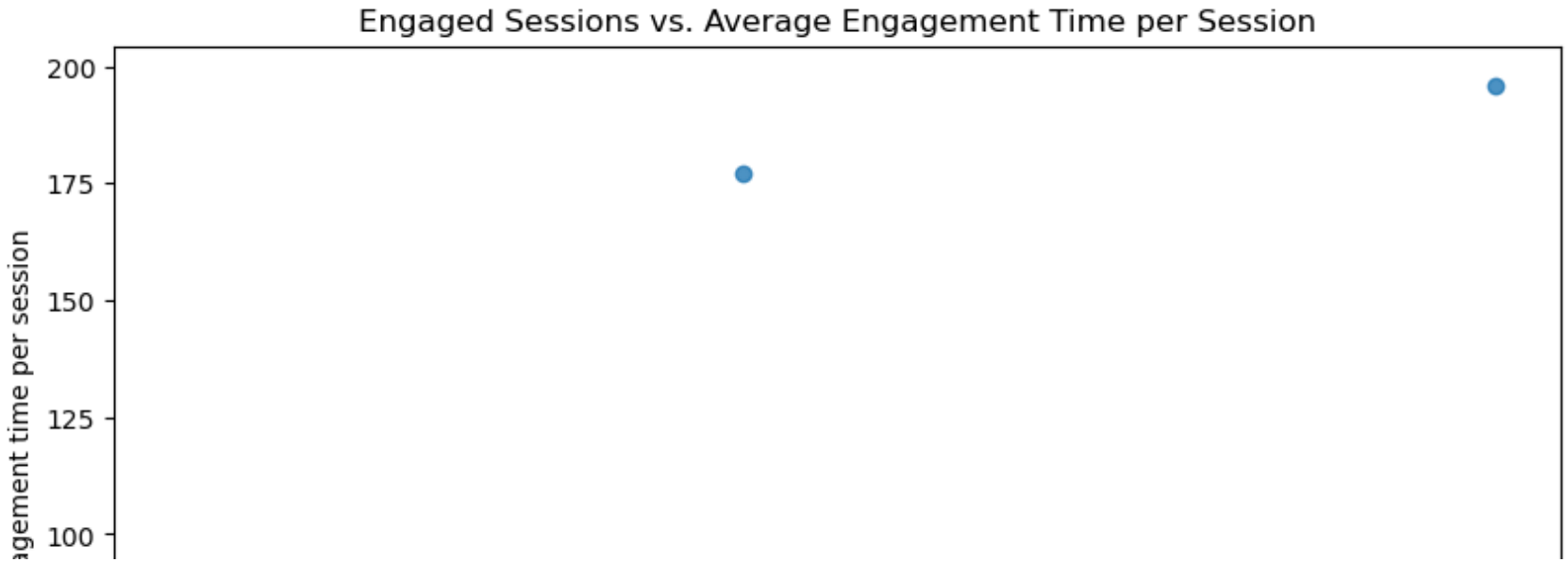



```
(merged_df.groupby('Sessions')['Events per session'].mean()*100).sort_values().plot(kind = 'barh')
```

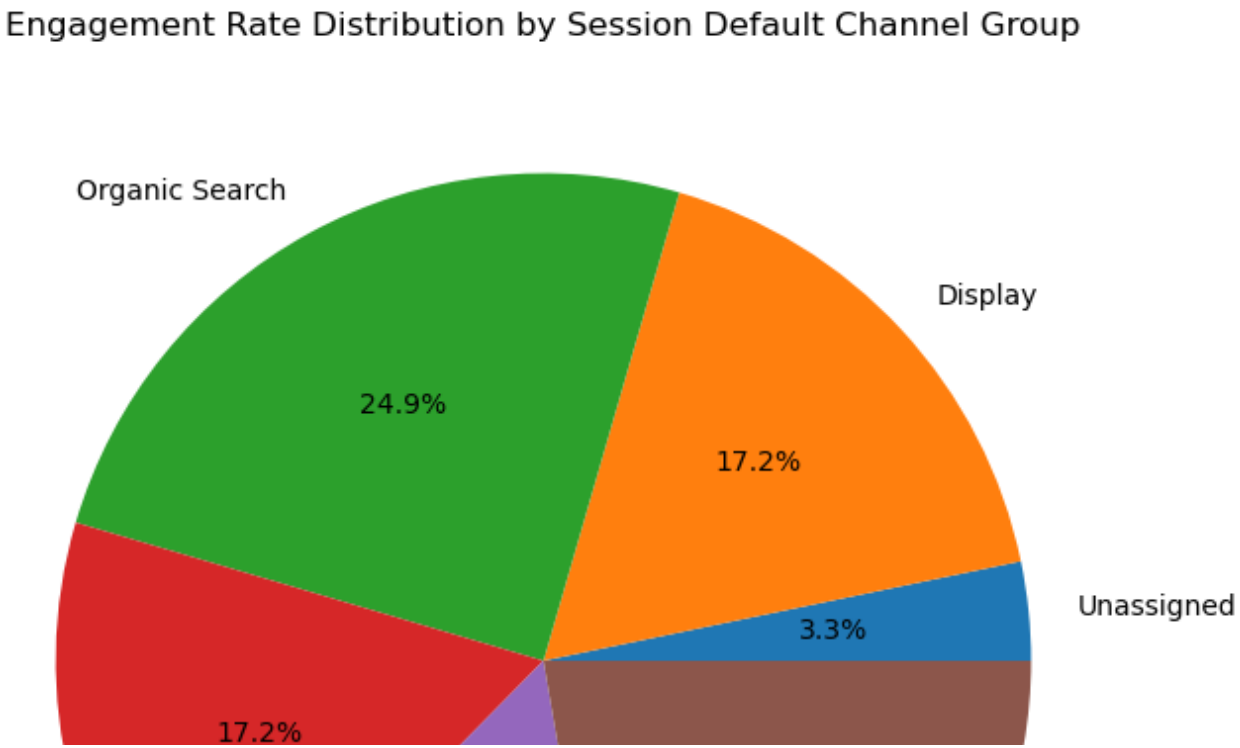
```
<AxesSubplot:ylabel='Sessions'>
```




```
# Scatter plot for Engaged sessions vs. Average engagement time per session
plt.figure(figsize=(10, 6))
plt.scatter(merged_df['Engaged sessions_x'], merged_df['Average engagement time per session'], alpha=0.8)
plt.xlabel('Engaged sessions')
plt.ylabel('Average engagement time per session')
plt.title('Engaged Sessions vs. Average Engagement Time per Session')
plt.show()
```



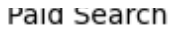

```
# Pie chart for Engagement rate distribution
plt.figure(figsize=(8, 8))
plt.pie(merged_df['Engagement rate_x'], labels=merged_df['Session default channel group'], autopct='%1.1f%%')
plt.title('Engagement Rate Distribution by Session Default Channel Group')
plt.show()
```



User Installation & Engagement Performance Analysis:

• Analyze the relationship between the User and factors such as region, customer demographics, and product attributes. • Identify the most significant factors influencing sales and their impact. • Determine any correlation or causation between variables and sales performance.

Pages & Screens Report




```
Pages_Screens_Report = excel.parse('Pages & Screens Report')
Pages_Screens_Report.head(5)
```

	Page path and screen class	Views	Users	Views per user	Average engagement time	Event count	Conversions	Total revenue
0	Flutter	156708	8726	17.958740	83.41222	203901	328	0
1	MainActivity	44326	8978	4.937180	78.29216	53374	101	0
2	feeds	18514	4358	4.248279	61.60005	37628	253	0

Demographics Report

```
Demographics_Report = excel.parse('Demographics Report')
Demographics_Report.head(5)
```

	Country	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	India	23024	22528	41479	0.593626	1.801555	334.81660	1312097	192766	0
1	United States	272	213	197	0.491272	0.724265	50.96324	3157	643	0
2	Canada	37	18	25	0.416667	0.675676	43.21622	410	121	0
3	(not set)	36	36	17	0.459459	0.472222	24.80556	241	54	0
4	United Kingdom	20	8	13	0.371429	0.650000	61.85000	289	43	0

Citiwise Report

```
Citi_Report = excel.parse('Citiwise Report')
Citi_Report.head(5)
```

	Town/City	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	Bengaluru	6097	5685	15013	0.769385	2.462359	762.20550	607200	62939	0
1	Patna	1594	1467	2127	0.440646	1.334379	98.22208	38830	6980	0
2	Hyderabad	1038	920	1578	0.569264	1.520231	243.69080	96826	34103	0
3	Indore	983	915	1241	0.426460	1.262462	67.89115	21383	4121	0
4	Lucknow	897	839	1125	0.450180	1.254181	83.40580	21041	3650	0

```
# Assuming you have resolved any issues with column names and data types

# Merge 'Pages_Screens_Report' and 'Demographics_Report' based on their indices
merged_df1_2 = Pages_Screens_Report.join(Demographics_Report, lsuffix='_Pages', rsuffix='_Demographics')

# Merge the resulting DataFrame (merged_df1_2) with 'Citi_Report' based on the index
merged_df = merged_df1_2.join(Citi_Report, lsuffix='_PagesDemographics', rsuffix='_Citi')

# Display the merged DataFrame
#print(merged_df)
merged_df = pd.DataFrame(merged_df)
merged_df.head(10)
```

	Page path and screen class	Views	Users_Pages	Views per user	Average engagement time_Pages	Event count_Pages	Conversions_Pages	Total revenue_Pages	Country
0	Flutter	156708	8726	17.958740	83.412220	203901	328	0	India
1	MainActivity	44326	8978	4.937180	78.292160	53374	101	0	United States
2	feeds	18514	4358	4.248279	61.600050	37628	253	0	Canada
3	login	16883	7291	2.315595	34.881770	40772	435	0	(not set)
4	my_rewards_screen	15381	2045	7.521271	94.179950	32910	5	0	United Kingdom

```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42 entries, 0 to 41
Data columns (total 28 columns):
 #   Column                                     Non-Null Count  Dtype  
---  ------                                     --------------  -----  
 0   Page path and screen class                 42 non-null     object 
 1   Views                                      42 non-null     int64  
 2   Users_Pages                                42 non-null     int64  
 3   Views per user                             42 non-null     float64
 4   Average engagement time_Pages              42 non-null     float64
 5   Event count_Pages                          42 non-null     int64  
 6   Conversions_Pages                          42 non-null     int64  
 7   Total revenue_Pages                        42 non-null     int64  
 8   Country                                    42 non-null     object 
 9   Users_Demographics                         42 non-null     int64  
 10  New users_PagesDemographics                42 non-null     int64  
 11  Engaged sessions_PagesDemographics         42 non-null     int64  
 12  Engagement rate_PagesDemographics          42 non-null     float64
 13  Engaged sessions per user_PagesDemographics  42 non-null     float64
 14  Average engagement time_Demographics       42 non-null     float64
 15  Event count_Demographics                   42 non-null     int64  
 16  Conversions_Demographics                   42 non-null     int64  
 17  Total revenue_Demographics                 42 non-null     int64  
```

```
 18  Town/City                                  42 non-null     object
 19  Users                                      42 non-null     int64
 20  New users_Citi                             42 non-null     int64
 21  Engaged sessions_Citi                      42 non-null     int64
 22  Engagement rate_Citi                       42 non-null     float64
 23  Engaged sessions per user_Citi             42 non-null     float64
 24  Average engagement time                    42 non-null     float64
 25  Event count                                42 non-null     int64
 26  Conversions                                42 non-null     int64
 27  Total revenue                              42 non-null     int64
dtypes: float64(8), int64(17), object(3)
memory usage: 9.3+ KB
```

```
merged_df.drop_duplicates()
```

	Page path and screen class	Views	Users_Pages	Views per user	Average engagement time_Pages	Event count_Pages	Conversions_Pages	To revenue_Pa
0	Flutter	156708	8726	17.958740	83.412220	203901	328	
1	MainActivity	44326	8978	4.937180	78.292160	53374	101	
2	feeds	18514	4358	4.248279	61.600050	37628	253	
3	login	16883	7291	2.315595	34.881770	40772	435	
4	my_rewards_screen	15381	2045	7.521271	94.179950	32910	5	
5	storyboard	8189	5244	1.561594	5.341152	15676	115	
6	SignInHubActivity	6650	3778	1.760191	0.003176	6653	0	
7	registration_screen	5501	3566	1.542625	45.075720	13496	136	
8	feedDetails	3971	1047	3.792741	69.316140	7820	84	
9	otp_screen	3291	1678	1.961263	46.864720	10833	32	
10	video_viewer_screem	2880	1521	1.893491	28.120970	5256	31	
11	FacebookActivity	2299	675	3.405926	0.524444	2310	0	
12	resume_builder	1781	828	2.150966	118.043500	3776	17	
13	CustomTabMainActivity	1301	193	6.740933	0.005181	1302	0	
14	notification_store	1062	648	1.638889	10.137350	1971	0	
15	dashboard	1058	411	2.574209	38.114360	2279	4	
16	myProfile_mediator	1056	600	1.760000	57.020000	2276	4	
17	WebViewActivity	878	400	1.701837	105.806100	1321	2	

17	WebViewActivity	878	490	1.791837	105.806100	1321	2
18	video_tutorial_view	835	722	1.156510	36.303320	1662	2
19	my_profile_learners	804	321	2.504673	127.722700	1856	4
20	FlutterViewController	758	155	4.890323	15.625810	1060	18
21	my_meetings_screen	715	273	2.619048	45.765570	1480	15
22	my_interests_screen	688	375	1.834667	29.189330	1340	7
23	discovery_screen	486	225	2.160000	70.235560	890	7
24	calculator_intro	388	281	1.380783	6.391459	680	0
25	campaign_interest	244	58	4.206897	42.379310	459	0
26	calculator_one	199	151	1.317881	41.013250	387	0
27	UIActivityViewSuccessController	99	18	5.500000	0.111111	101	0
28	UIActivityContentViewController	97	17	5.705882	32.411760	177	0
29	calculator_two	88	73	1.205479	9.287671	158	0

```
merged_df.describe()
```

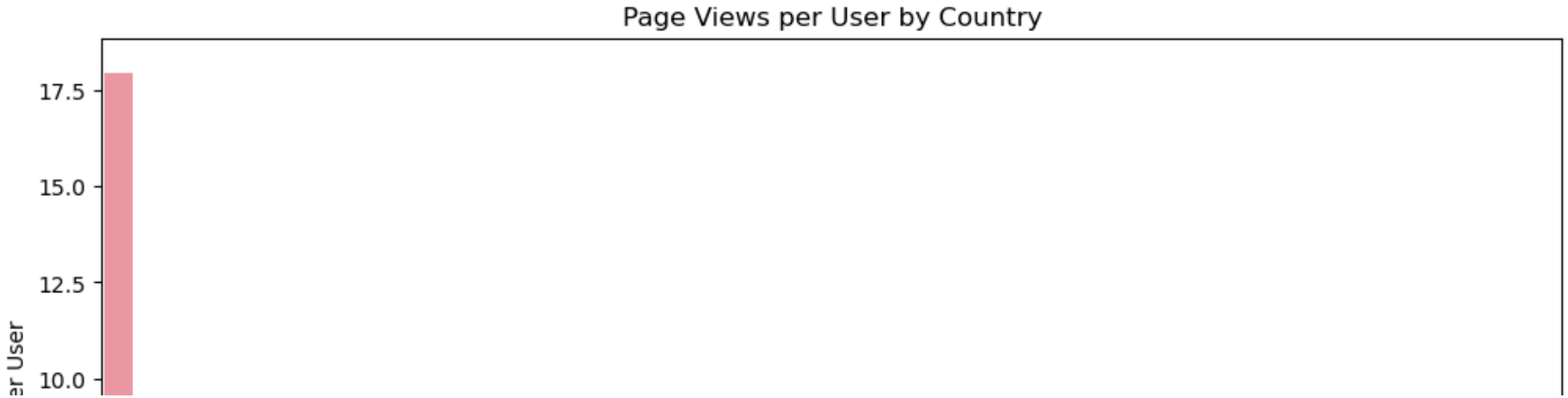
	Views	Users_Pages	Views per user	Average engagement time_Pages	Event count_Pages	Conversions_Pages	Total revenue_Pages	Users_Demographics
count	42.000000	42.000000	42.000000	42.000000	42.000000	42.000000	42.0	42.000000
mean	7081.261905	1524.761905	3.036123	48.898515	13417.547619	2145.690476	0.0	559.690476

```
merged_df.isnull().sum()
```

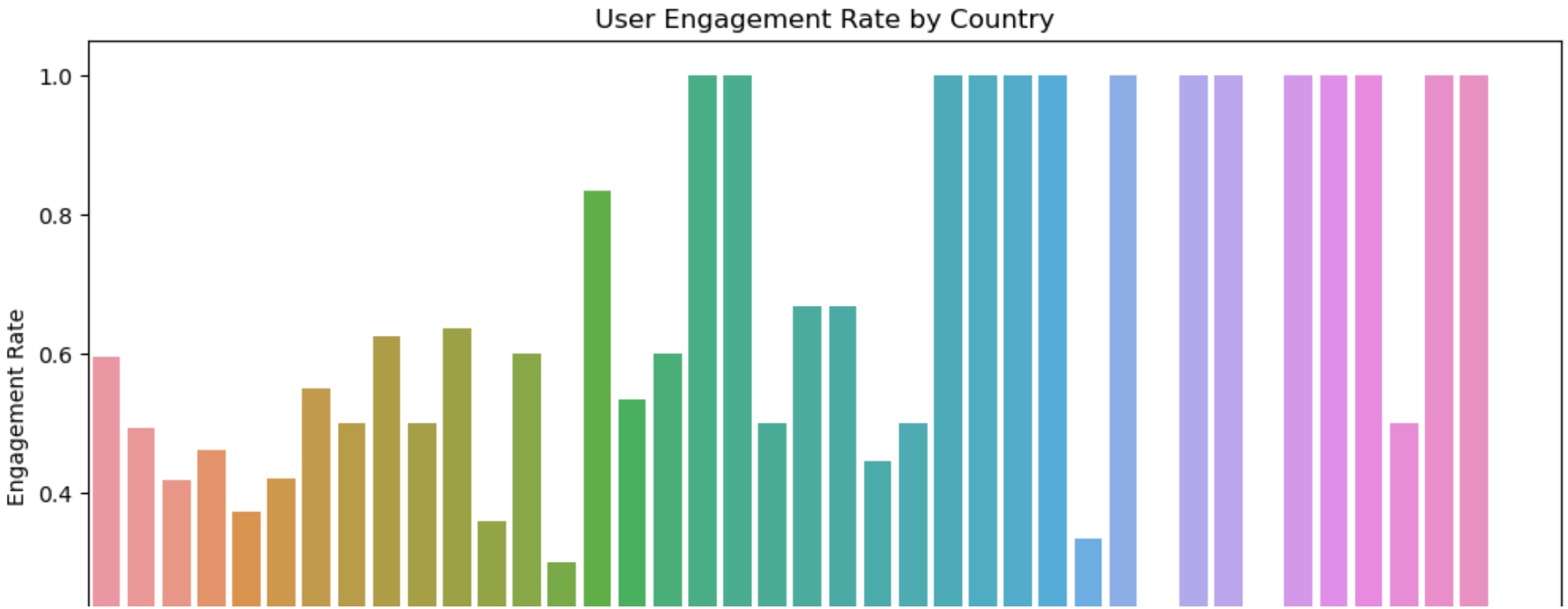
```
Page path and screen class                    0
Views                                         0
Users_Pages                                   0
Views per user                                0
Average engagement time_Pages                 0
Event count_Pages                             0
Conversions_Pages                             0
Total revenue_Pages                           0
Country                                       0
Users_Demographics                            0
New users_PagesDemographics                   0
Engaged sessions_PagesDemographics            0
Engagement rate_PagesDemographics             0
Engaged sessions per user_PagesDemographics   0
Average engagement time_Demographics          0
Event count_Demographics                      0
Conversions_Demographics                      0
Total revenue_Demographics                    0
Town/City                                     0
Users                                         0
New users_Citi                                0
Engaged sessions_Citi                         0
Engagement rate_Citi                          0
Engaged sessions per user_Citi                0
Average engagement time                       0
Event count                                   0
Conversions                                   0
Total revenue                                 0
dtype: int64
```

```
# User Engagement by Country - Bar Plot
plt.figure(figsize=(12, 6))
sns.barplot(data=merged_df, x='Country', y='Engaged sessions_PagesDemographics')
plt.title('User Engagement by Country - Engaged Sessions')
plt.xlabel('Country')
plt.ylabel('Engaged Sessions')
plt.xticks(rotation=45)
plt.show()
```

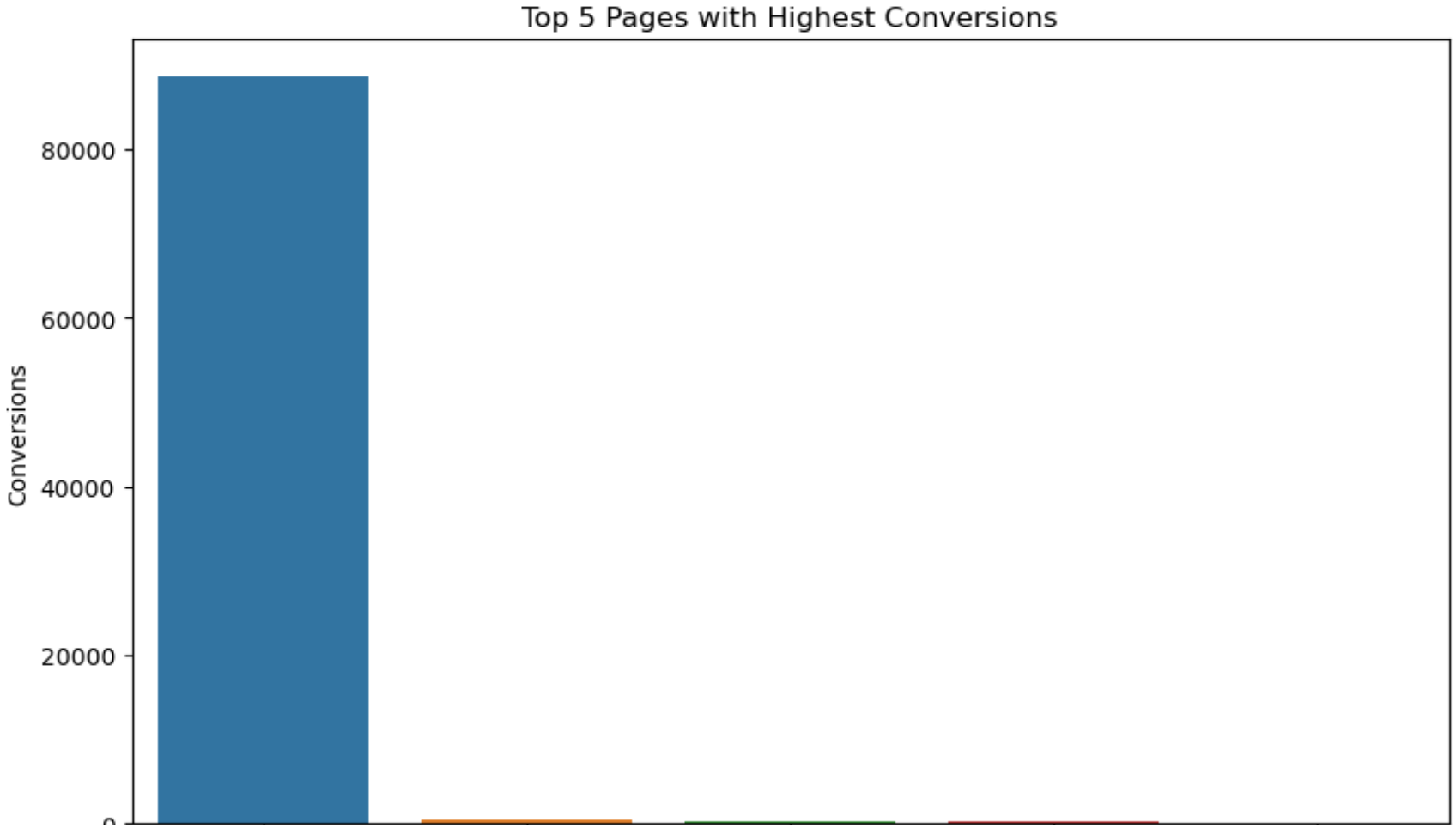

```
# Page Views per User by Country - Bar Plot
plt.figure(figsize=(12, 6))
sns.barplot(data=merged_df, x='Country', y='Views per user')
plt.title('Page Views per User by Country')
plt.xlabel('Country')
plt.ylabel('Views per User')
plt.xticks(rotation=45)
plt.show()
```




```
# User Engagement Rate by Country - Bar Plot
plt.figure(figsize=(12, 6))
sns.barplot(data=merged_df, x='Country', y='Engagement rate_PagesDemographics')
plt.title('User Engagement Rate by Country')
plt.xlabel('Country')
plt.ylabel('Engagement Rate')
plt.xticks(rotation=45)
plt.show()
```



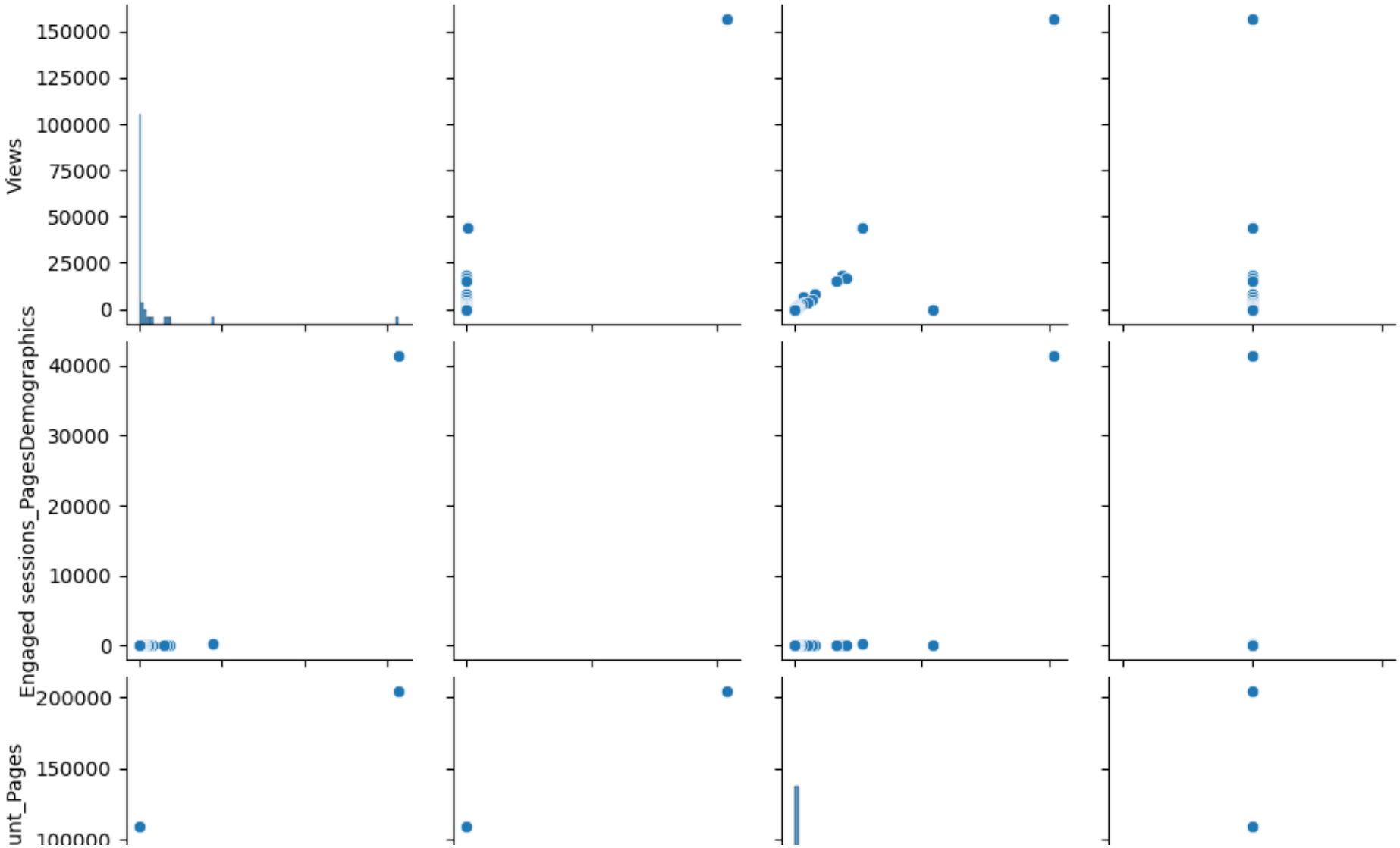

```
# Top 5 Pages with Highest Conversions - Bar Plot
plt.figure(figsize=(10, 6))
conversions_top5 = merged_df.nlargest(5, 'Conversions_Pages')
sns.barplot(data=conversions_top5, x='Page path and screen class', y='Conversions_Pages')
plt.title('Top 5 Pages with Highest Conversions')
plt.xlabel('Page')
plt.ylabel('Conversions')
plt.xticks(rotation=45)
plt.show()
```




```
# Scatter Plot - Engaged Sessions vs. Average Engagement Time
plt.figure(figsize=(10, 6))
sns.scatterplot(data=merged_df, x='Engaged sessions_PagesDemographics', y='Average engagement time_Demographics', hue='Country')
plt.title('Engaged Sessions vs. Average Engagement Time')
plt.xlabel('Engaged Sessions')
plt.ylabel('Average Engagement Time')
plt.show()
```

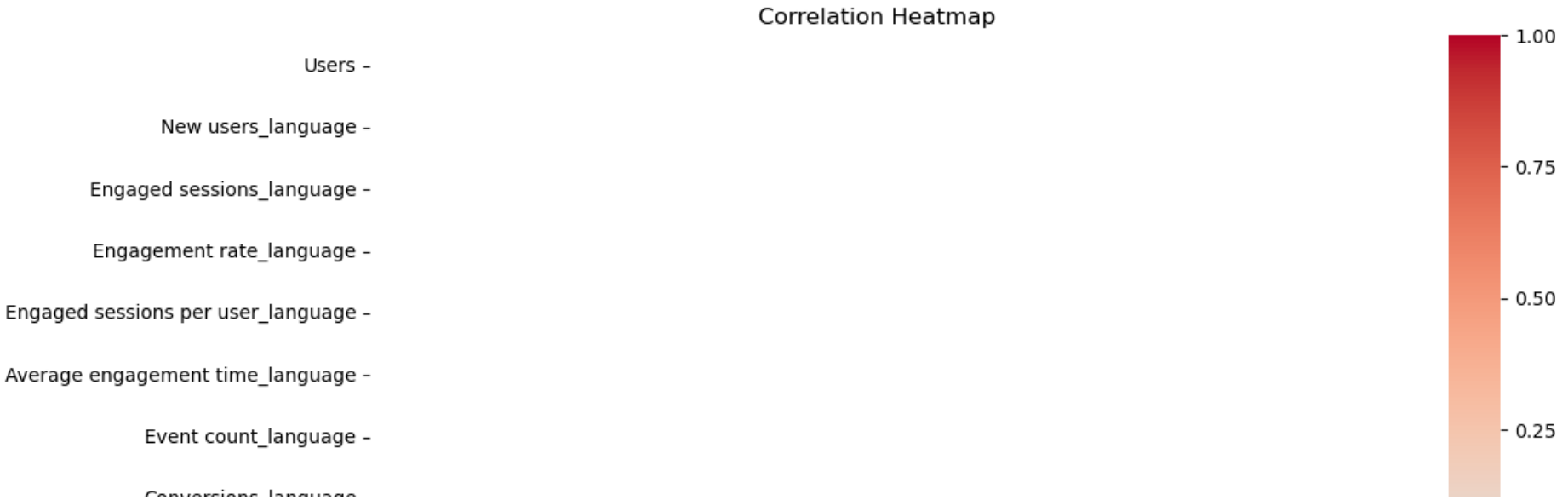

Country
India
United States
Canada
(not set)
United Kingdom
Singapore
Japan
Australia
Bangladesh
Germany
Malaysia
Nepal
Saudi Arabia
United Arab Emirates
Kuwait
Myanmar (Burma)
Qatar
China
Indonesia
Ireland
Italy
Netherlands
South Korea
Switzerland
Afghanistan
Argentina
Bahamas
Dominican Republic
France
Guernsey
Iran
Kyrgyzstan
Latvia
Engaged Sessions vs. Average Engagement Time
500
400
300
ngagement Time


```
# Pairplot to visualize multiple variables at once
sns.pairplot(merged_df[['Views', 'Engaged sessions_PagesDemographics', 'Event count_Pages', 'Total revenue_Pages']])
plt.show()
```




```
corr_matrix = merged_df.corr()

# Set up the heatmap figure size
plt.figure(figsize=(12, 10))

# Create a heatmap with a suitable color map and annotating significant correlation values
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5, vmin=-1, vmax=1)
```

```
plt.title('Correlation Heatmap')
plt.xticks(rotation=45)
plt.yticks(rotation=0)  # Remove rotation for y-axis
plt.show()
```



Event Report

```
Event_Report = excel.parse('Event Report')
Event_Report.head(5)
```

	Event name	Event count	Total users	Event count per user	Total revenue
0	screen_view	694729	23254	30.865870	0
1	notification_receive	125146	1700	138.896800	0
2	user_engagement	124836	22699	5.622230	0
3	notification_dismiss	70128	1369	144.000000	0
4	session_start	61163	23226	3.121357	0

Conversion_Report

```
Conversion_Report= excel.parse('Conversion Report')
Conversion_Report.head(5)
```

	Event name	Conversions	Total users	Total revenue
0	notification_receive	94890	1311	0
1	session_start	56203	21674	0
2	first_open	22872	23059	0
3	app_remove	12468	12538	0
4	Promilo111_otp_screen	1738	855	0

```
# Merge Event_Report with Conversion_Report on 'Event name'
event_conversion_df = pd.merge(Event_Report, Conversion_Report, on='Event name', how='outer')
event_conversion_df.head()
```

	Event name	Event count	Total users_x	Event count per user	Total revenue_x	Conversions	Total users_y	Total revenue_y
0	screen_view	694729	23254	30.865870	0	NaN	NaN	NaN
1	notification_receive	125146	1700	138.896800	0	94890.0	1311.0	0.0
2	user_engagement	124836	22699	5.622230	0	NaN	NaN	NaN
3	notification_dismiss	70128	1369	144.000000	0	NaN	NaN	NaN
4	session_start	61163	23226	3.121357	0	56203.0	21674.0	0.0

```
event_conversion_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 379 entries, 0 to 378
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
```

```
---  ------                --------------  -----
 0   Event name            378 non-null    object
 1   Event count           379 non-null    int64
 2   Total users_x         379 non-null    int64
 3   Event count per user  379 non-null    float64
 4   Total revenue_x       379 non-null    int64
 5   Conversions           18 non-null     float64
 6   Total users_y         18 non-null     float64
 7   Total revenue_y       18 non-null     float64
dtypes: float64(4), int64(3), object(1)
memory usage: 26.6+ KB
```

```
event_conversion_df.drop_duplicates()
```

	Event name	Event count	Total users_x	Event count per user	Total revenue_x	Conversions	Total users_y	Total revenue_y
0	screen_view	694729	23254	30.865870	0	NaN	NaN	NaN
1	notification_receive	125146	1700	138.896800	0	94890.0	1311.0	0.0
2	user_engagement	124836	22699	5.622230	0	NaN	NaN	NaN
3	notification_dismiss	70128	1369	144.000000	0	NaN	NaN	NaN
4	session_start	61163	23226	3.121357	0	56203.0	21674.0	0.0
...	...	...	...	...	...	...	...	...
374	Promilo119_myProfile_mediator	1	1	1.000000	0	NaN	NaN	NaN
375	Promilo_feeds	1	1	1.000000	0	NaN	NaN	NaN
376	feeds	1	1	1.000000	0	NaN	NaN	NaN
377	my_interests_screen	1	1	1.000000	0	NaN	NaN	NaN
378	(not set)	0	22269	0.000000	0	NaN	NaN	NaN

379 rows × 8 columns

```
# Display summary statistics of the merged DataFrame
event_conversion_df.describe()
```

	Event count	Total users_x	Event count per user	Total revenue_x	Conversions	Total users_y	Total revenue_y
count	379.000000	379.000000	379.000000	379.0	18.000000	18.000000	18.0
mean	3478.968338	583.583113	3.220647	0.0	10773.000000	3468.777778	0.0
std	37073.635167	2791.449046	10.403279	0.0	25160.036433	7461.269980	0.0
min	0.000000	1.000000	0.000000	0.0	20.000000	10.000000	0.0
25%	7.000000	3.500000	1.252451	0.0	127.250000	33.750000	0.0
50%	63.000000	31.000000	1.750000	0.0	620.500000	246.500000	0.0
75%	503.500000	238.000000	2.793359	0.0	1702.000000	940.500000	0.0
max	694729.000000	23254.000000	144.000000	0.0	94890.000000	23059.000000	0.0

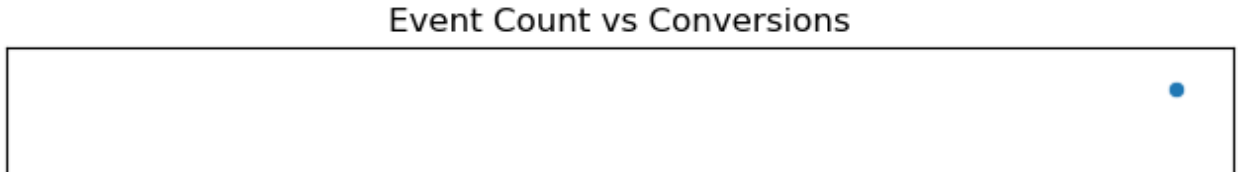
```
event_conversion_df.isnull().sum()
```

```
Event name               1
Event count              0
Total users_x            0
Event count per user     0
Total revenue_x          0
Conversions            361
Total users_y          361
Total revenue_y        361
dtype: int64
```

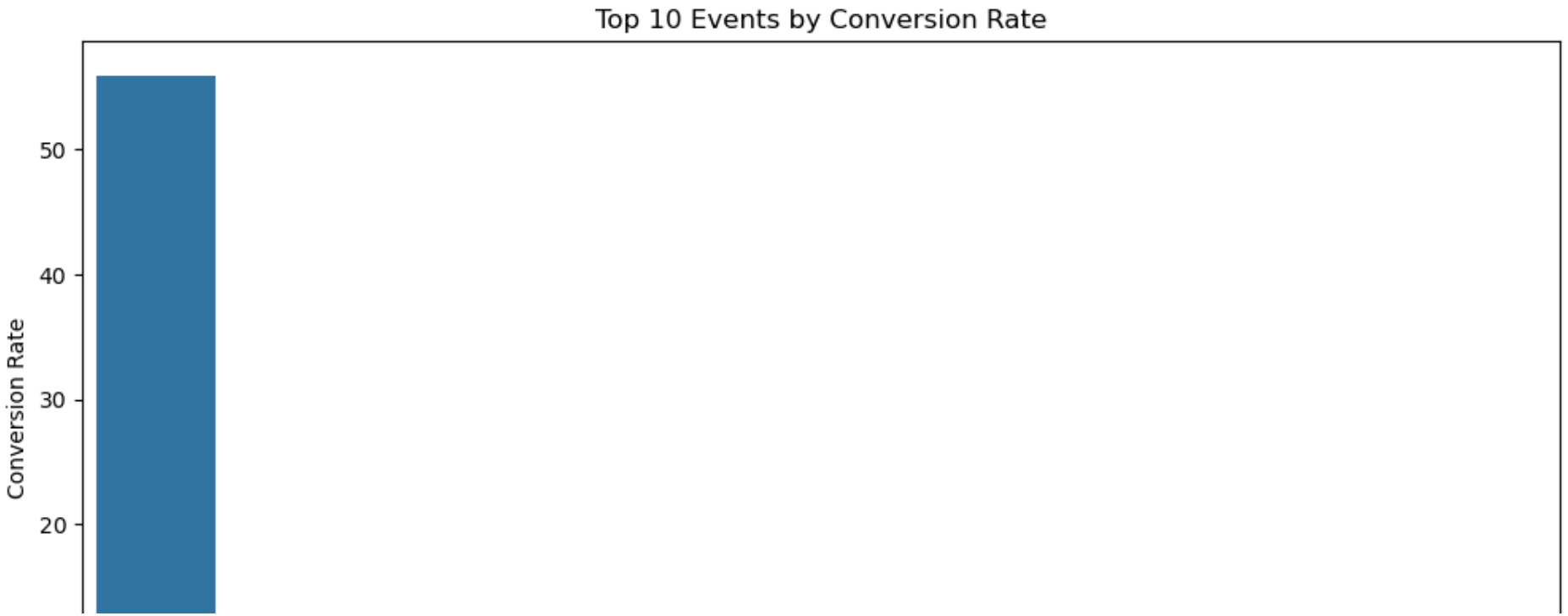
```
# Select the top N events based on event counts
N = 10
top_events = event_conversion_df.nlargest(N, 'Event count')

# Event Count for the top N events
plt.figure(figsize=(12, 6))
```

```
sns.barplot(data=top_events, x='Event name', y='Event count')
plt.title(f'Top {N} Events by Event Count')
plt.xlabel('Event name')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```


```
#  Event Count vs Conversions
plt.figure(figsize=(8, 6))
sns.scatterplot(data=event_conversion_df, x='Event count', y='Conversions')
plt.title('Event Count vs Conversions')
plt.xlabel('Event Count')
plt.ylabel('Conversions')
plt.show()
```




```
N = 10  # Number of top events to display
top_events = event_conversion_df.nlargest(N, 'Conversion Rate')
plt.figure(figsize=(12, 6))
sns.barplot(data=top_events, x='Event name', y='Conversion Rate')
plt.title(f'Top {N} Events by Conversion Rate')
plt.xlabel('Event name')
plt.ylabel('Conversion Rate')
plt.xticks(rotation=90)
plt.show()
```



Gender Report

```
Gender_Report = excel.parse('Gender Report')
Gender_Report.head(5)
```

	Gender	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	unknown	13142	12691	23161	0.564077	1.762365	439.5776	761771	93180	0
1	male	7218	5877	10467	0.543091	1.450125	128.2319	282504	65651	0
2	female	4944	4304	7877	0.637710	1.593244	208.7407	274254	35083	0

User By Interest

```
User_Interest = excel.parse('User By Interest')
User_Interest.head(5)
```

	Interests	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	Shoppers	10950	9256	15652	0.581534	1.429406	162.8347	490664	86846	0
1	Media & Entertainment/Comics & Animation Fans	10946	9247	15680	0.583008	1.432487	165.1772	491025	86845	0
2	Technology/Mobile Enthusiasts	10934	9239	15619	0.582451	1.428480	162.6945	489353	86742	0
3	Food & Dining/Cooking Enthusiasts	8410	6970	12332	0.602325	1.466350	176.9567	409713	73814	0
4	Sports & Fitness/Health &	5844	4580	8226	0.588328	1.407598	155.1451	257831	43074	0

```
User_Interest.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89 entries, 0 to 88
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Interests                 89 non-null     object
 1   Users                     89 non-null     int64
 2   New users                 89 non-null     int64
 3   Engaged sessions          89 non-null     int64
 4   Engagement rate           89 non-null     float64
 5   Engaged sessions per user 89 non-null     float64
 6   Average engagement time   89 non-null     float64
 7   Event count               89 non-null     int64
```

```
 8   Conversions                89 non-null     int64
 9   Total revenue              89 non-null     int64
dtypes: float64(3), int64(6), object(1)
memory usage: 7.1+ KB
```

```
User_Interest.drop_duplicates()
```

	Interests	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	Shoppers	10950	9256	15652	0.581534	1.429406	162.83470	490664	86846	0
1	Media & Entertainment/Comics & Animation Fans	10946	9247	15680	0.583008	1.432487	165.17720	491025	86845	0
2	Technology/Mobile Enthusiasts	10934	9239	15619	0.582451	1.428480	162.69450	489353	86742	0
3	Food & Dining/Cooking Enthusiasts	8410	6970	12332	0.602325	1.466350	176.95670	409713	73814	0
4	Sports & Fitness/Health & Fitness Buffs	5844	4580	8226	0.588328	1.407598	155.14510	257831	43074	0
...	...	...	...	...	...	...	...	...	...	...
84	Food & Dining	15	4	24	0.489796	1.600000	70.86667	460	58	0
85	Home & Garden	15	5	12	0.631579	0.800000	133.86670	453	107	0
86	Sports & Fitness/Sports Fans/Racquetball Enthu...	11	11	21	0.840000	1.909091	487.45450	736	39	0
87	Vehicles & Transportation	11	3	9	0.450000	0.818182	71.54545	161	27	0
88	Sports & Fitness/Sports Fans/Fans of American ...	10	4	18	0.782609	1.800000	201.40000	375	81	0

```
User_Interest.describe()
```

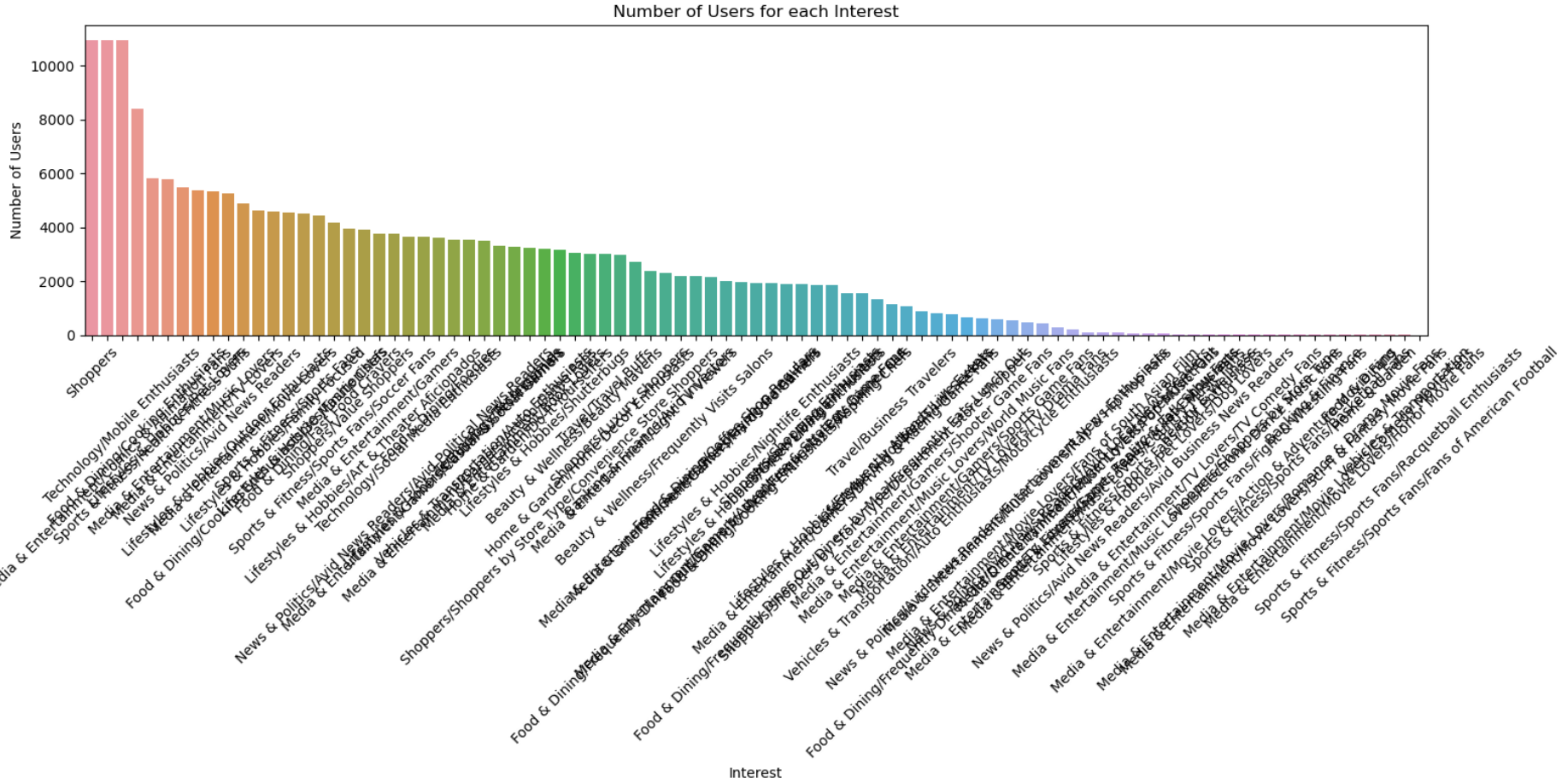
	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
count	89.000000	89.000000	89.000000	89.000000	89.000000	89.000000	89.000000	89.000000	89.0
mean	2411.044944	1911.280899	3373.932584	0.619191	1.339172	219.531871	110565.202247	19512.898876	0.0
std	2480.482385	2059.110984	3544.597874	0.081918	0.258550	366.821595	114195.875599	19401.222657	0.0
min	10.000000	3.000000	9.000000	0.450000	0.684211	54.263160	161.000000	27.000000	0.0
25%	107.000000	72.000000	142.000000	0.576499	1.218631	126.944900	5805.000000	2971.000000	0.0
50%	1942.000000	1566.000000	2608.000000	0.618416	1.345895	162.643800	90176.000000	16412.000000	0.0
75%	3652.000000	2872.000000	5443.000000	0.666153	1.459730	208.648200	174955.000000	29630.000000	0.0
max	10950.000000	9256.000000	15680.000000	0.840000	2.066351	3471.304000	491025.000000	86846.000000	0.0

```
User_Interest.isnull().sum()
```

```
Interests                   0
Users                       0
New users                   0
Engaged sessions            0
Engagement rate             0
Engaged sessions per user   0
Average engagement time     0
Event count                 0
Conversions                 0
Total revenue               0
dtype: int64
```

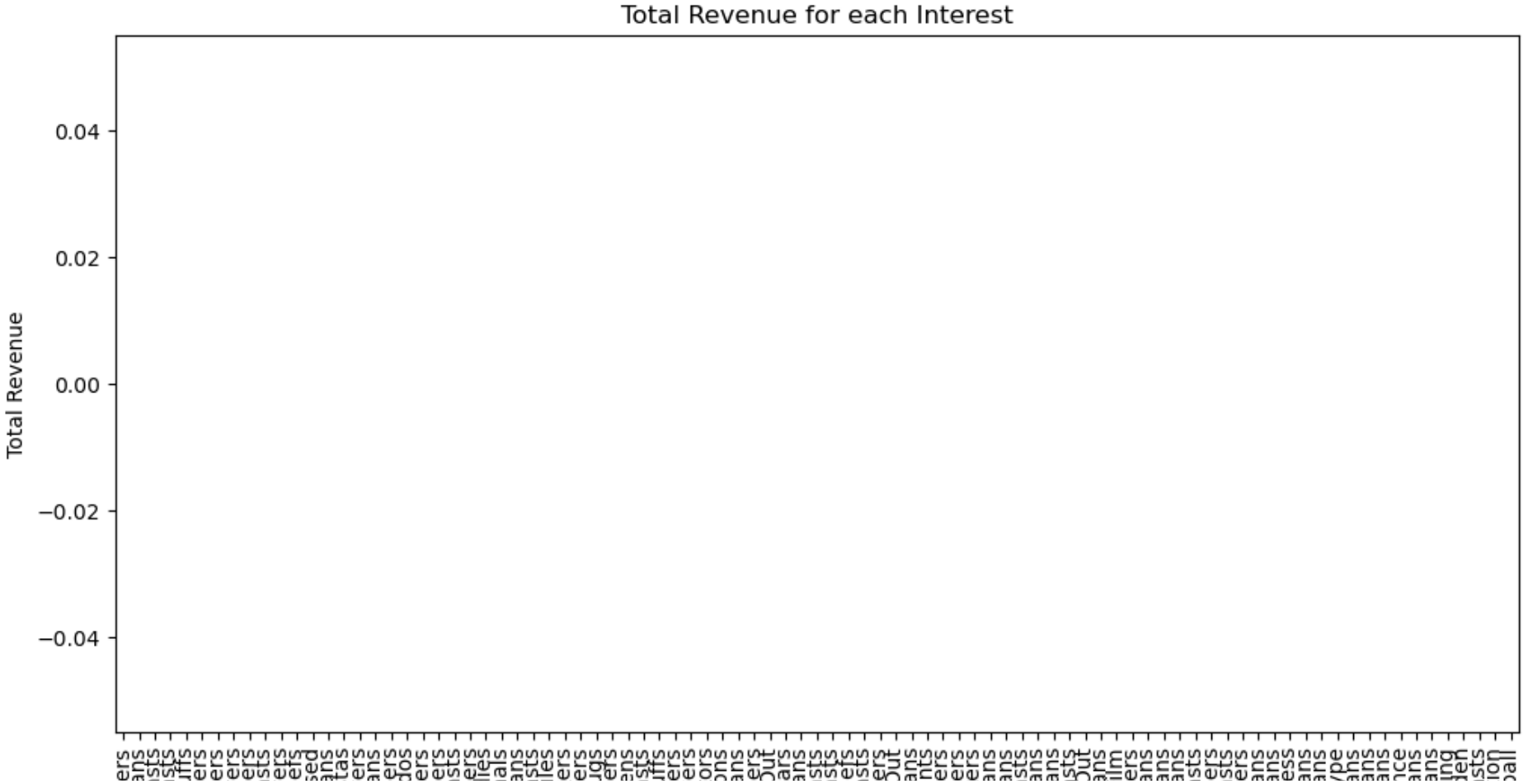
```
plt.figure(figsize=(16, 8))
sns.barplot(data=User_Interest, x='Interests', y='Users')
plt.title('Number of Users for each Interest')
plt.xlabel('Interest')
```

```
plt.ylabel('Number of Users')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

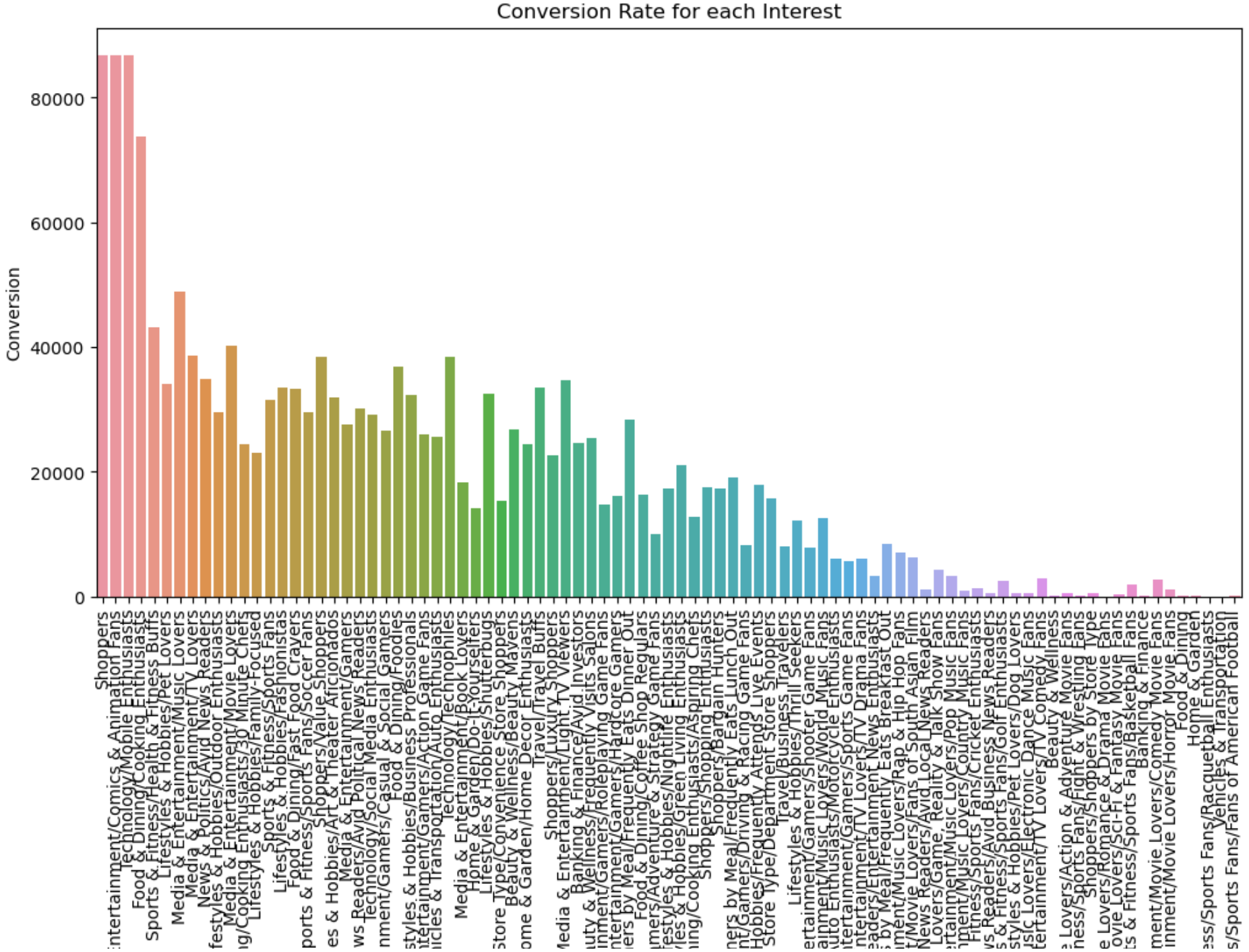



```
plt.figure(figsize=(12, 6))
sns.barplot(data=User_Interest, x='Interests', y='Total revenue')
plt.title('Total Revenue for each Interest')
```

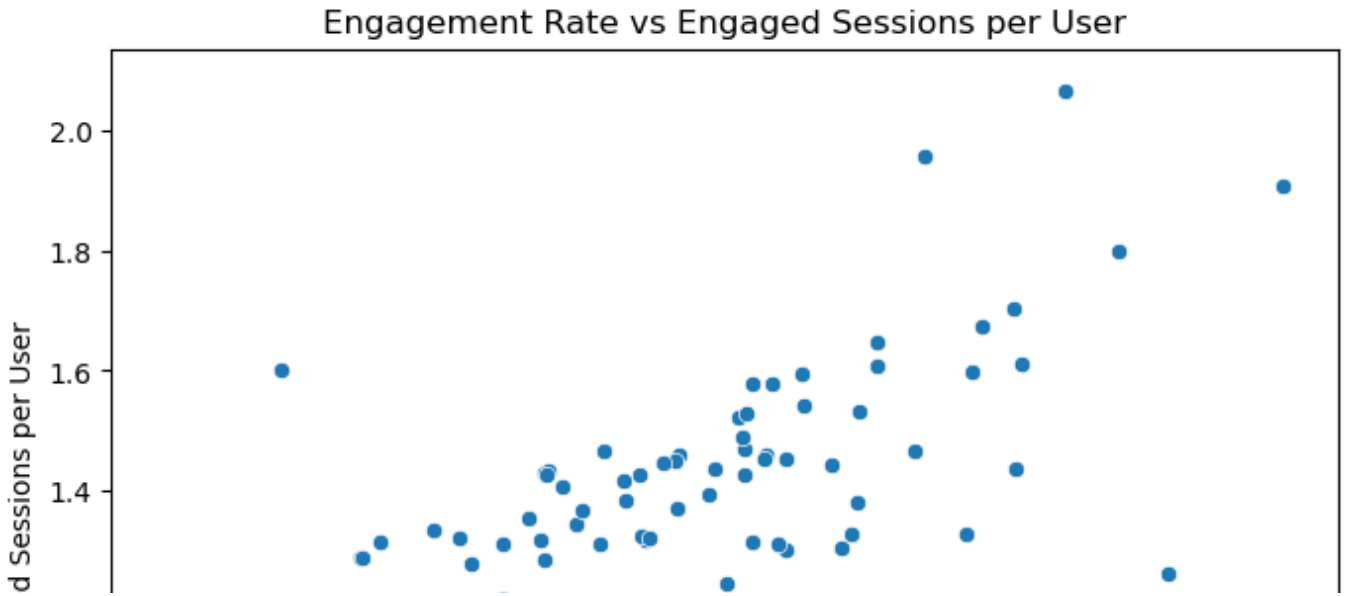
```
plt.xlabel('Interest')
plt.ylabel('Total Revenue')
plt.xticks(rotation=90)
plt.show()
```



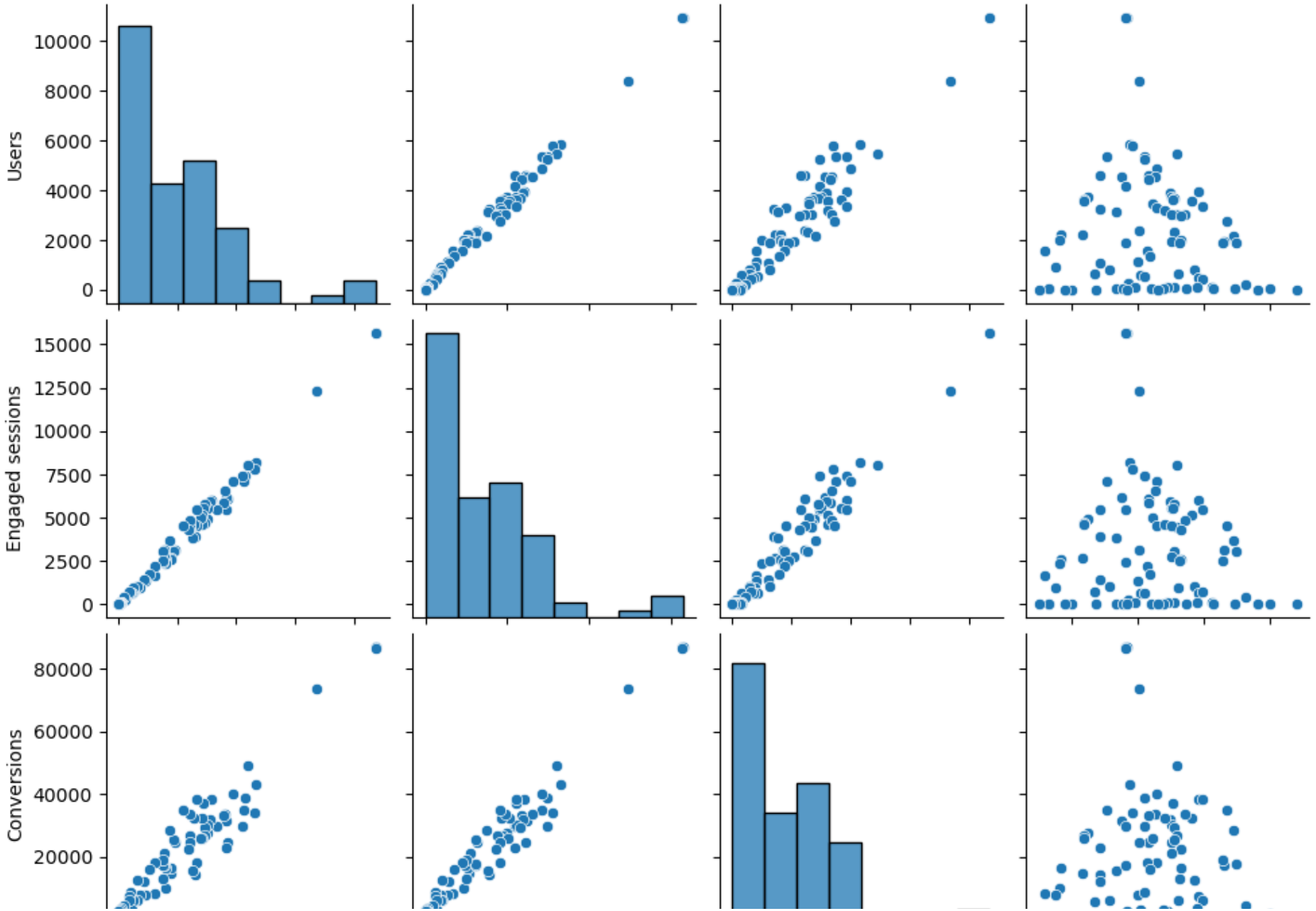

```
plt.figure(figsize=(12, 6))
sns.barplot(data=User_Interest, x='Interests', y='Conversions')
plt.title('Conversion Rate for each Interest')
plt.xlabel('Interest')
plt.ylabel('Conversion')
plt.xticks(rotation=90)
plt.show()
```


Conversion Rate for each Interest
Conversion
80000
60000
40000
20000
0

```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=User_Interest, x='Engagement rate', y='Engaged sessions per user')
plt.title('Engagement Rate vs Engaged Sessions per User')
plt.xlabel('Engagement Rate')
plt.ylabel('Engaged Sessions per User')
plt.show()
```




```
sns.pairplot(User_Interest[['Users', 'Engaged sessions', 'Conversions', 'Engagement rate']])
plt.show()
```



▾ User by Language

```
User_Language= excel.parse('User by Language')
User_Language.head()
```

	Language	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	English	22495	21990	40639	0.595147	1.806579	341.36350	1297970	189946	0
1	Hindi	586	552	798	0.406314	1.361775	60.03413	13523	2699	0
2	Marathi	85	84	98	0.426087	1.152941	38.48235	1589	323	0
3	Gujarati	78	77	100	0.448430	1.282051	46.53846	1794	327	0
4	Telugu	43	42	56	0.455285	1.302326	36.65116	812	170	0

```
User_Language.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24 entries, 0 to 23
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Language                  24 non-null     object
 1   Users                     24 non-null     int64
 2   New users                 24 non-null     int64
 3   Engaged sessions          24 non-null     int64
 4   Engagement rate           24 non-null     float64
 5   Engaged sessions per user  24 non-null     float64
 6   Average engagement time   24 non-null     float64
 7   Event count               24 non-null     int64
 8   Conversions               24 non-null     int64
 9   Total revenue             24 non-null     int64
dtypes: float64(3), int64(6), object(1)
memory usage: 2.0+ KB
```

```
User_Language.describe()
```

	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
count	24.000000	24.000000	24.000000	24.000000	24.000000	24.000000	2.400000e+01	24.000000	24.0
mean	975.916667	953.000000	1744.833333	0.535395	1.046263	65.527106	5.493871e+04	8079.750000	0.0
std	4585.079838	4482.258359	8285.991965	0.321298	0.642224	83.316618	2.647787e+05	38741.203482	0.0
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.000000e+00	1.000000	0.0
25%	1.000000	1.000000	1.000000	0.421144	0.750000	8.812500	1.075000e+01	2.000000	0.0
50%	7.000000	7.000000	7.000000	0.509036	1.000000	40.241175	1.060000e+02	21.000000	0.0
75%	21.750000	20.250000	37.750000	0.677083	1.310078	74.812500	6.312500e+02	85.000000	0.0
max	22495.000000	21990.000000	40639.000000	1.000000	2.384615	341.363500	1.297970e+06	189946.000000	0.0

```
User_Language.isnull().sum()
```

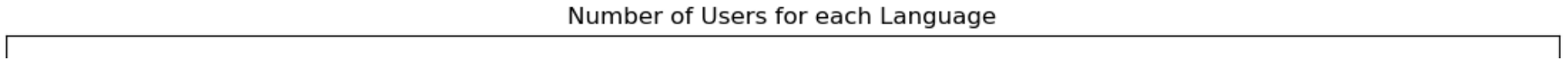
```
Language                    0
Users                       0
New users                   0
Engaged sessions            0
Engagement rate             0
Engaged sessions per user   0
Average engagement time     0
Event count                 0
Conversions                 0
Total revenue               0
dtype: int64
```

```
User_Language.drop_duplicates()
```

	Language	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	English	22495	21990	40639	0.595147	1.806579	341.36350	1297970	189946	0
1	Hindi	586	552	798	0.406314	1.361775	60.03413	13523	2699	0
2	Marathi	85	84	98	0.426087	1.152941	38.48235	1589	323	0
3	Gujarati	78	77	100	0.448430	1.282051	46.53846	1794	327	0
4	Telugu	43	42	56	0.455285	1.302326	36.65116	812	170	0
5	Tamil	36	36	43	0.518072	1.194444	45.86111	615	115	0
6	Malayalam	17	15	36	0.654545	2.117647	161.94120	548	71	0
7	Bengali	14	11	18	0.600000	1.285714	50.07143	217	39	0
8	Chinese	13	13	13	1.000000	1.000000	136.76920	138	20	0
9	Kannada	13	12	31	0.500000	2.384615	249.07690	680	75	0
10	Panjabi	9	9	17	0.708333	1.888889	92.44444	229	35	0
11	Persian	8	8	6	0.400000	0.750000	28.25000	99	23	0
12	Spanish	6	6	8	0.470588	1.333333	22.16667	113	22	0
13	Finnish	4	3	4	0.571429	1.000000	89.25000	64	11	0
14	Japanese	4	4	3	0.428571	0.750000	9.25000	49	12	0
15	Oriya	4	4	2	0.666667	0.500000	7.50000	29	10	0
16	Afrikaans	1	1	1	1.000000	1.000000	37.00000	12	2	0
17	Assamese	1	0	1	1.000000	1.000000	42.00000	6	1	0
18	German	1	1	0	0.000000	0.000000	0.00000	5	2	0
19	Malay	1	1	1	1.000000	1.000000	2.00000	7	2	0
20	Nepali	1	1	1	1.000000	1.000000	5.00000	7	2	0

21	Russian	1	1	0	0.000000	0.000000	70.000000	12	1	0

```
plt.figure(figsize=(12, 6))
sns.barplot(data=User_Language, x='Language', y='Users')
plt.title('Number of Users for each Language')
plt.xlabel('Language')
plt.ylabel('Number of Users')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

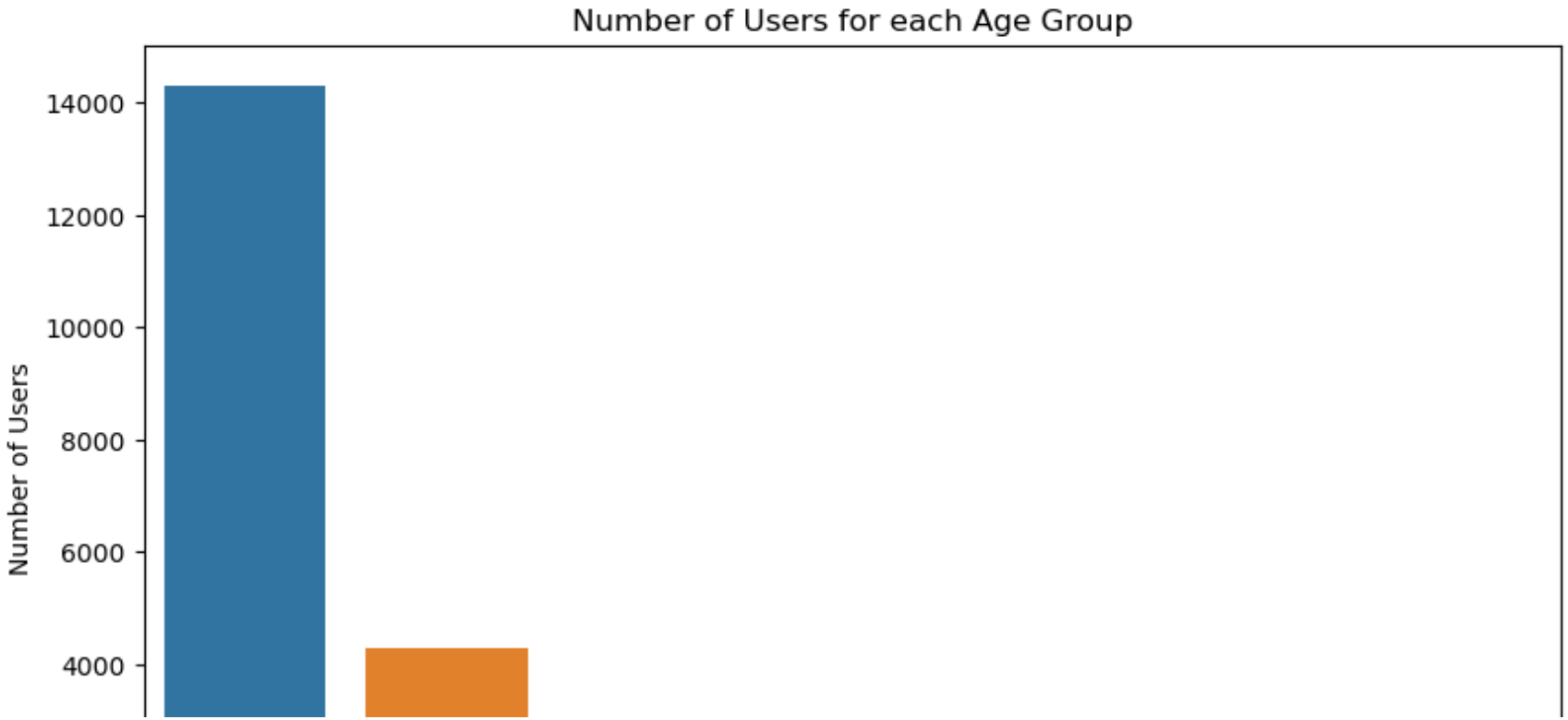


User By Age

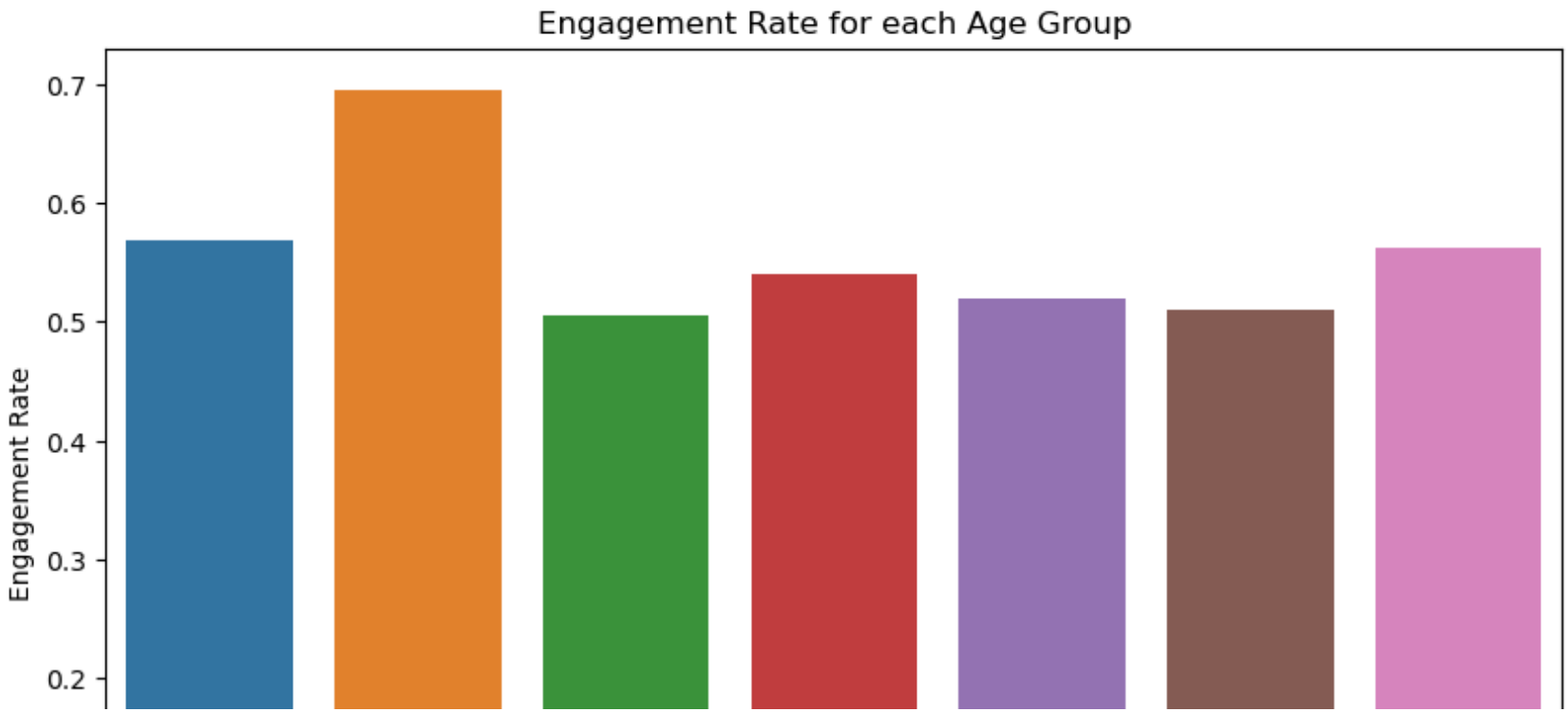
```
df11 = excel.parse('User By Age')
df11.head()
```

	Age	Users	New users	Engaged sessions	Engagement rate	Engaged sessions per user	Average engagement time	Event count	Conversions	Total revenue
0	unknown	14303	13636	24976	0.569098	1.746207	422.22330	817501	99310	0
1	18-24	4282	3678	7291	0.695308	1.702709	251.16300	309328	53661	0
2	25-34	2920	2161	3749	0.504780	1.283904	97.24144	90074	20172	0
3	65+	1422	1081	1640	0.539829	1.153305	52.30661	24780	4891	0
4	55-64	1403	979	1552	0.519411	1.106201	55.37063	25169	4823	0

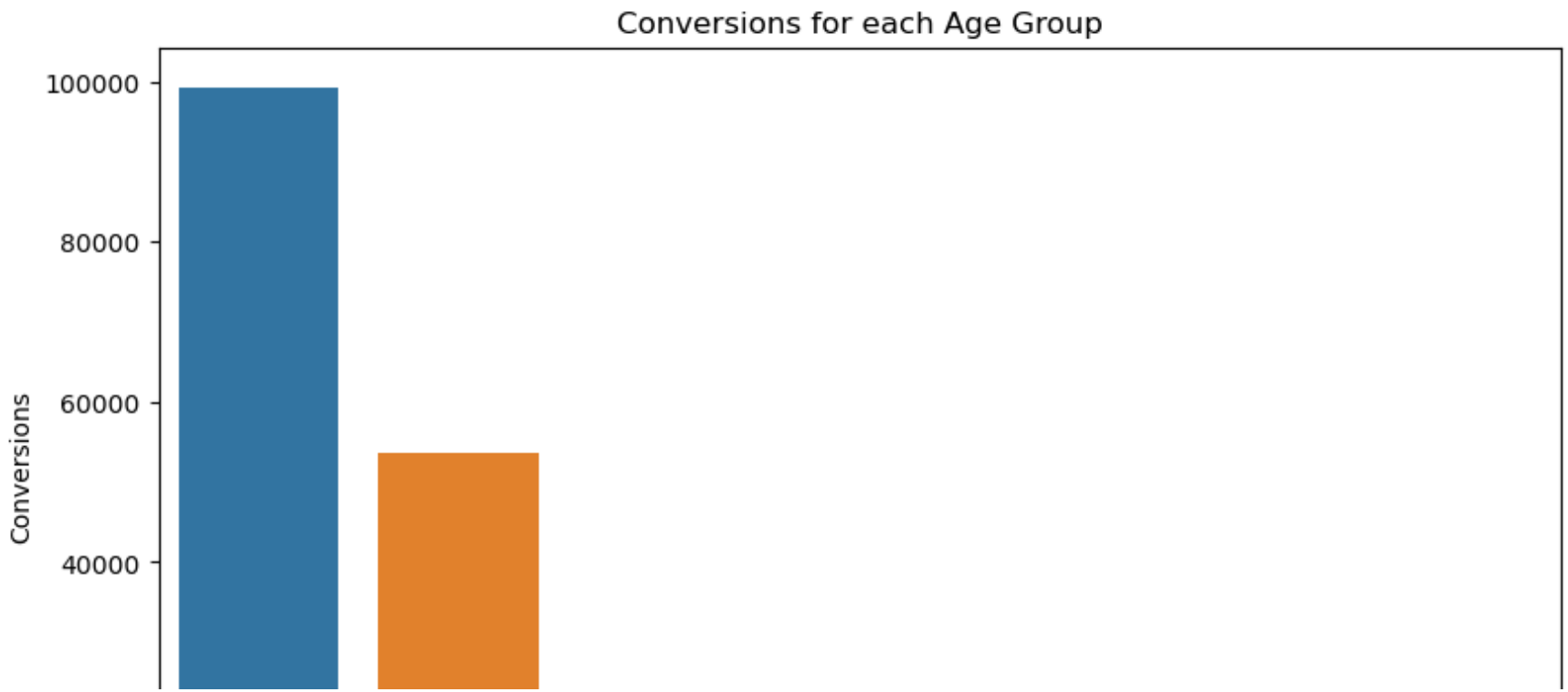
```
plt.figure(figsize=(10, 6))
sns.barplot(data=df11, x='Age', y='Users')
plt.title('Number of Users for each Age Group')
plt.xlabel('Age Group')
plt.ylabel('Number of Users')
plt.show()
```



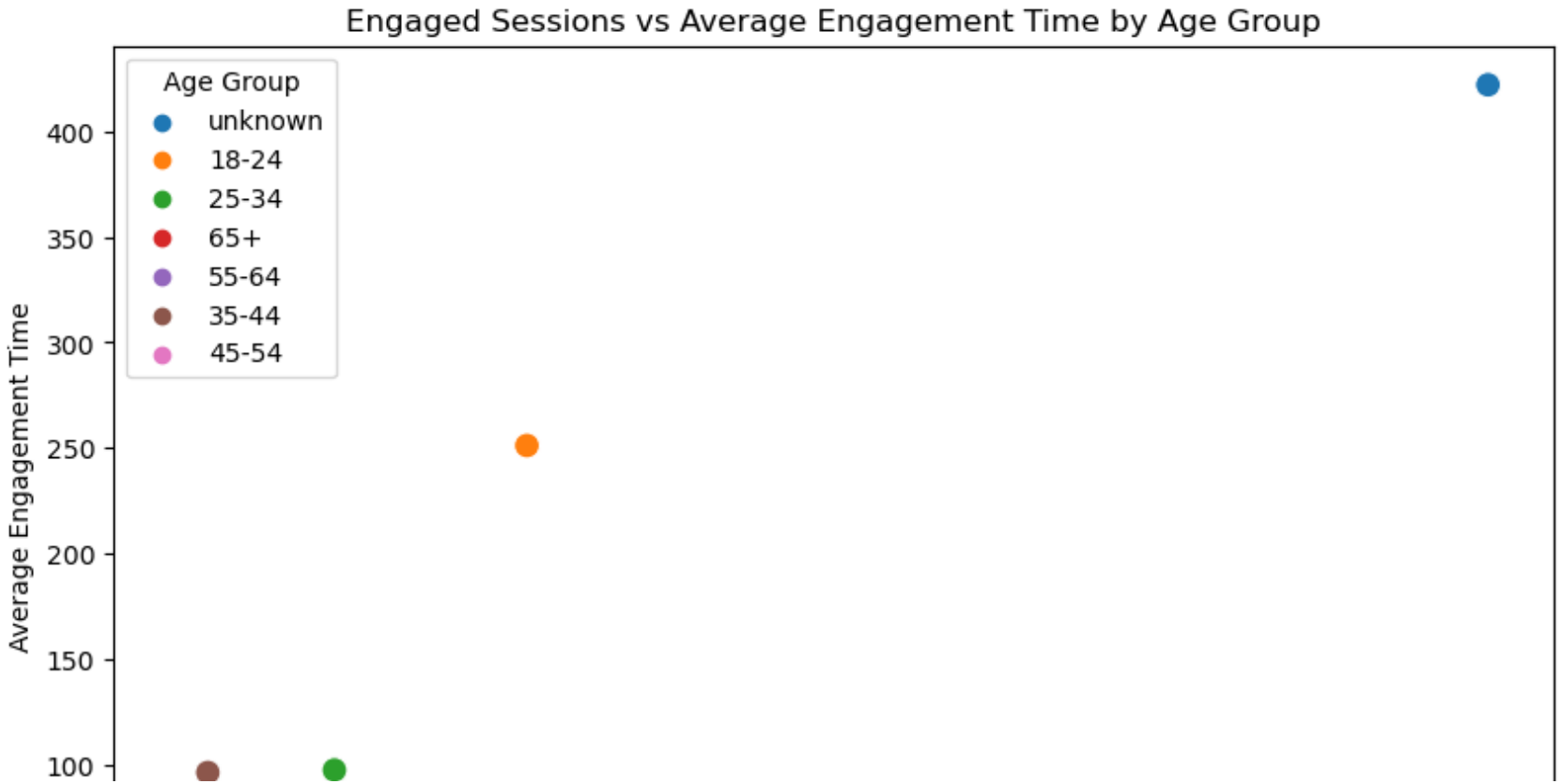

```
plt.figure(figsize=(10, 6))
sns.barplot(data=df11, x='Age', y='Engagement rate')
plt.title('Engagement Rate for each Age Group')
plt.xlabel('Age Group')
plt.ylabel('Engagement Rate')
plt.show()
```




```
#Conversions for each Age Group
plt.figure(figsize=(10, 6))
sns.barplot(data=df11, x='Age', y='Conversions')
plt.title('Conversions for each Age Group')
plt.xlabel('Age Group')
plt.ylabel('Conversions')
plt.show()
```




```
#Engaged Sessions vs Average Engagement Time by Age Group
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df11, x='Engaged sessions', y='Average engagement time', hue='Age', s=100)
plt.title('Engaged Sessions vs Average Engagement Time by Age Group')
plt.xlabel('Engaged Sessions')
plt.ylabel('Average Engagement Time')
plt.legend(title='Age Group')
plt.show()
```



Marketing Campaign Analysis:

• Evaluate the effectiveness of past marketing campaigns. • Analyze campaign metrics (e.g., conversion rates, ROI) and their impact on sales. • Identify successful campaigns and areas for improvement.

Google Ads Report

```
Google_Ads_Report = excel.parse('Google Ads Report')
Google_Ads_Report.head()
```

	Session Google Ads campaign	Users	Sessions	Engaged sessions	Google Ads clicks	Google Ads cost	Google Ads cost per click	Conversions	Cost per conversion	Event count	Total revenue	Return on ad spend
0	App Installation for May --Shahid	5429	10936	6276	147100	179175.000	1.218049	12257	14.618180	97802	0	0
1	App Install-States-A200Inst-20Jun22	842	1655	968	28742	24309.130	0.845770	1794	13.550240	15311	0	0
2	App Install-States-B100Installs-22Jun22	742	1332	780	17809	22374.580	1.256363	1422	15.734580	11640	0	0
3	App Install for April --Shahid	473	976	546	19302	20525.180	1.063370	1115	18.408230	8001	0	0

```
Google_Ads_Report.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 12 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Session Google Ads campaign  15 non-null     object 
 1   Users                        15 non-null     int64  
 2   Sessions                     15 non-null     int64  
 3   Engaged sessions             15 non-null     int64  
 4   Google Ads clicks            15 non-null     int64  
 5   Google Ads cost              15 non-null     float64
 6   Google Ads cost per click    15 non-null     float64
 7   Conversions                  15 non-null     int64  
 8   Cost per conversion          15 non-null     float64
 9   Event count                  15 non-null     int64  
```

```
 10  Total revenue                15 non-null     int64
 11  Return on ad spend           15 non-null     int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1.5+ KB
```

```
Google_Ads_Report.describe()
```

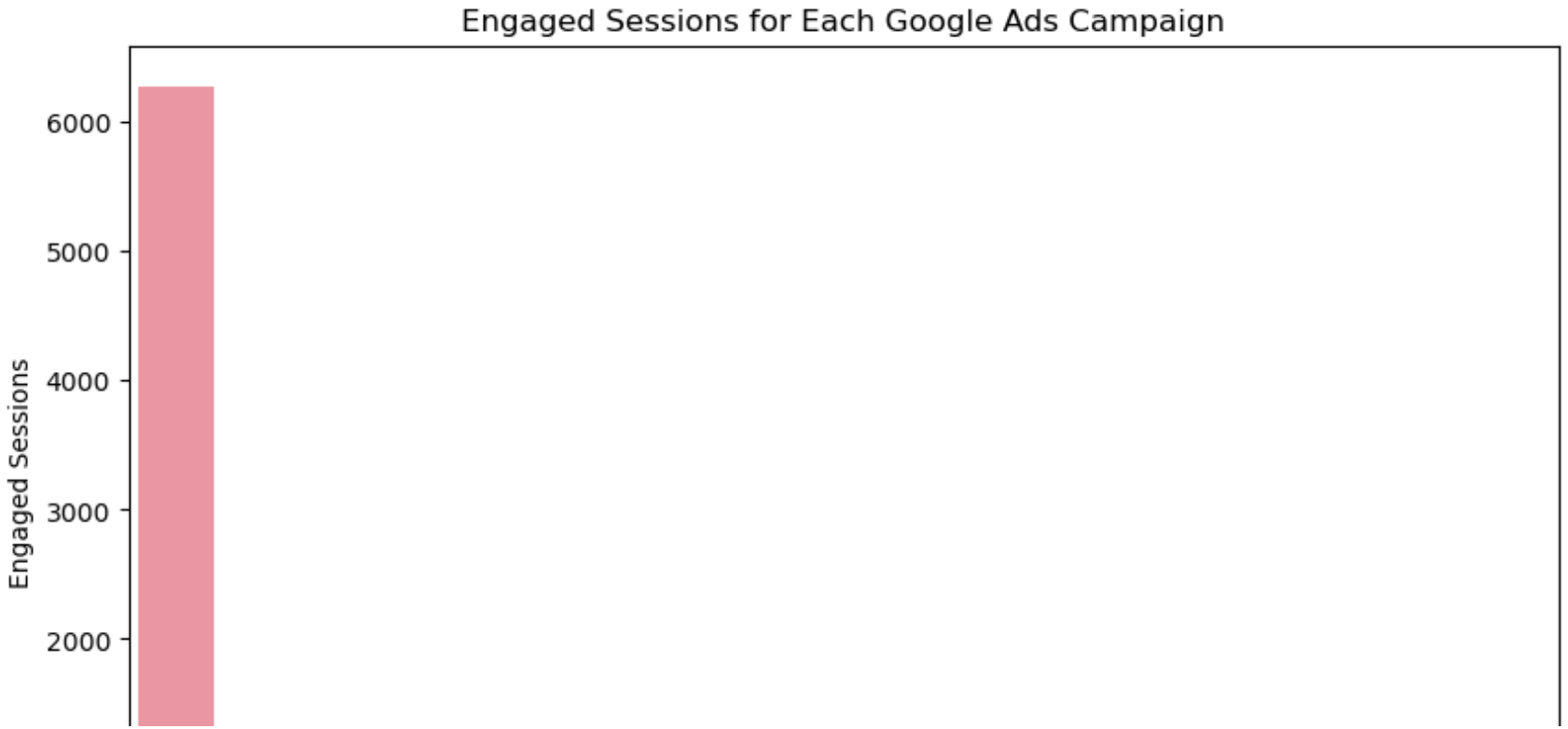
	Users	Sessions	Engaged sessions	Google Ads clicks	Google Ads cost	Google Ads cost per click	Conversions	Cost per conversion	Event count
count	15.000000	15.000000	15.000000	15.000000	15.000000	15.000000	15.000000	15.000000	15.000000
mean	666.200000	1274.133333	756.666667	17021.200000	20089.196451	1.237784	1421.533333	12.410360	12334.933333
std	1348.253696	2721.749093	1558.314184	36934.069186	44773.537357	0.635640	3047.587755	3.915273	24145.503978
min	2.000000	5.000000	5.000000	14.000000	16.623960	0.485200	5.000000	3.324793	163.000000
25%	62.000000	100.500000	65.500000	1778.500000	1399.447500	0.745567	125.500000	12.390005	1434.000000
50%	370.000000	610.000000	425.000000	4475.000000	8839.723000	1.131950	709.000000	13.106330	7504.000000
75%	621.000000	971.000000	654.500000	14202.000000	16304.610000	1.656639	1073.500000	14.614405	10844.500000
max	5429.000000	10936.000000	6276.000000	147100.000000	179175.000000	2.415885	12257.000000	18.408230	97802.000000

```
Google_Ads_Report.isnull().sum()
```

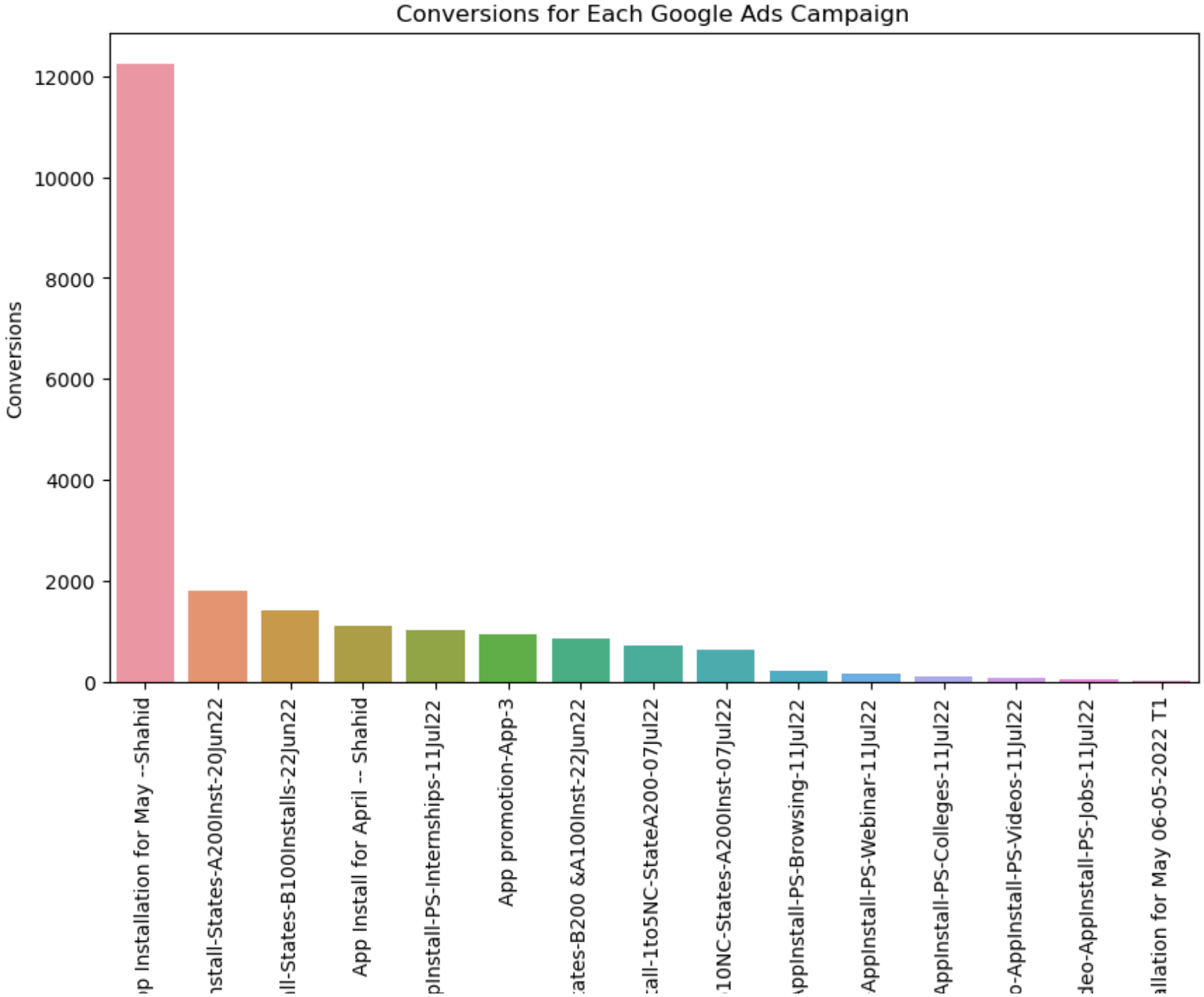
```
Session Google Ads campaign    0
Users                          0
Sessions                       0
Engaged sessions               0
Google Ads clicks              0
Google Ads cost                0
Google Ads cost per click      0
Conversions                    0
Cost per conversion            0
```

```
Event count                    0
Total revenue                  0
Return on ad spend             0
dtype: int64
```

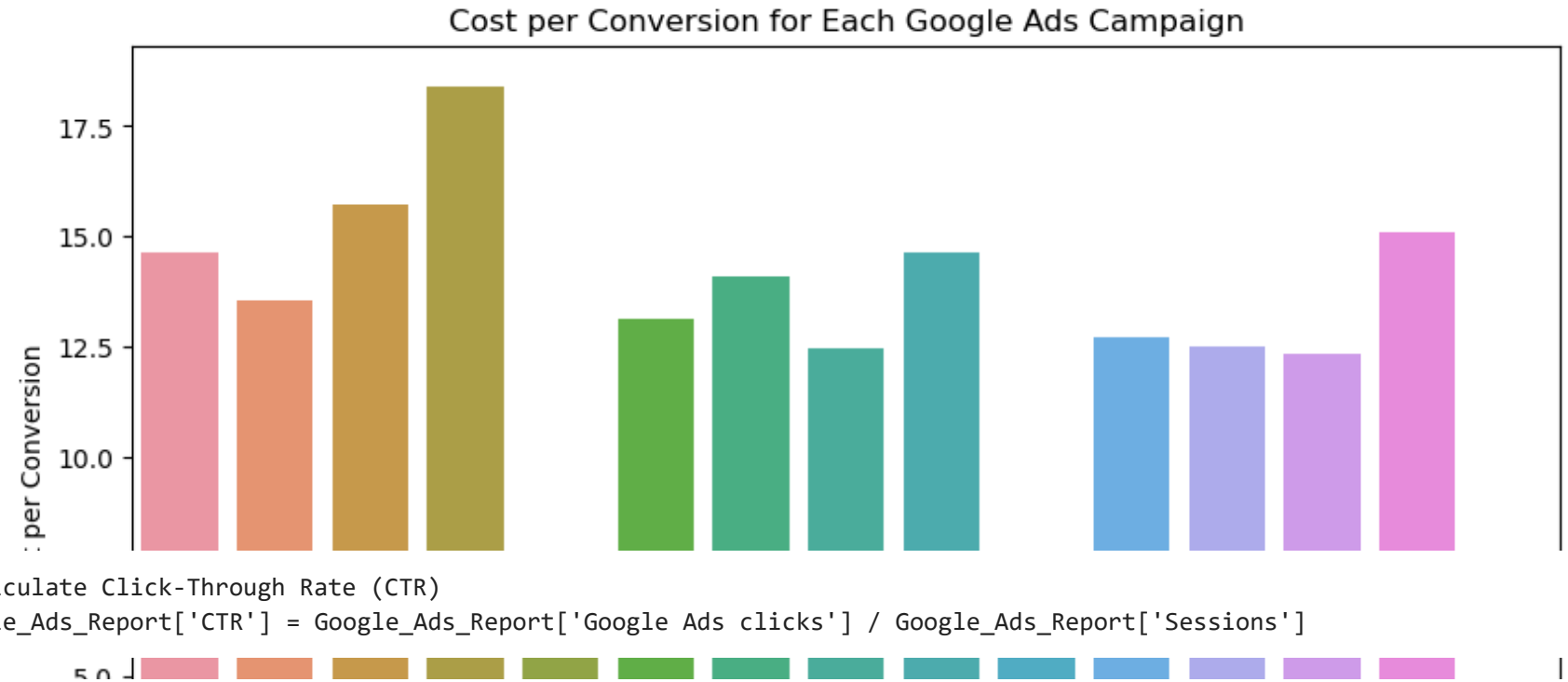
```
# Visualize Engaged Sessions
plt.figure(figsize=(10, 6))
sns.barplot(data=Google_Ads_Report, x='Session Google Ads campaign', y='Engaged sessions')
plt.title('Engaged Sessions for Each Google Ads Campaign')
plt.xlabel('Google Ads Campaign')
plt.ylabel('Engaged Sessions')
plt.xticks(rotation=90)
plt.show()
```




```
# Visualize Conversions
plt.figure(figsize=(10, 6))
sns.barplot(data=Google_Ads_Report, x='Session Google Ads campaign', y='Conversions')
plt.title('Conversions for Each Google Ads Campaign')
plt.xlabel('Google Ads Campaign')
plt.ylabel('Conversions')
plt.xticks(rotation=90)
plt.show()
```


Conversions for Each Google Ads Campaign
Conversions
0
2000
4000
6000
8000
10000
12000
p Installation for May --Shahid
nstall-States-A200Inst-20Jun22
ll-States-B100Installs-22Jun22
App Install for April -- Shahid
pInstall-PS-Internships-11Jul22
App promotion-App-3
tates-B200 &A100Inst-22Jun22
all-1to5NC-StateA200-07Jul22
10NC-States-A200Inst-07Jul22
AppInstall-PS-Browsing-11Jul22
AppInstall-PS-Webinar-11Jul22
AppInstall-PS-Colleges-11Jul22
o-AppInstall-PS-Videos-11Jul22
deo-AppInstall-PS-Jobs-11Jul22
allation for May 06-05-2022 T1

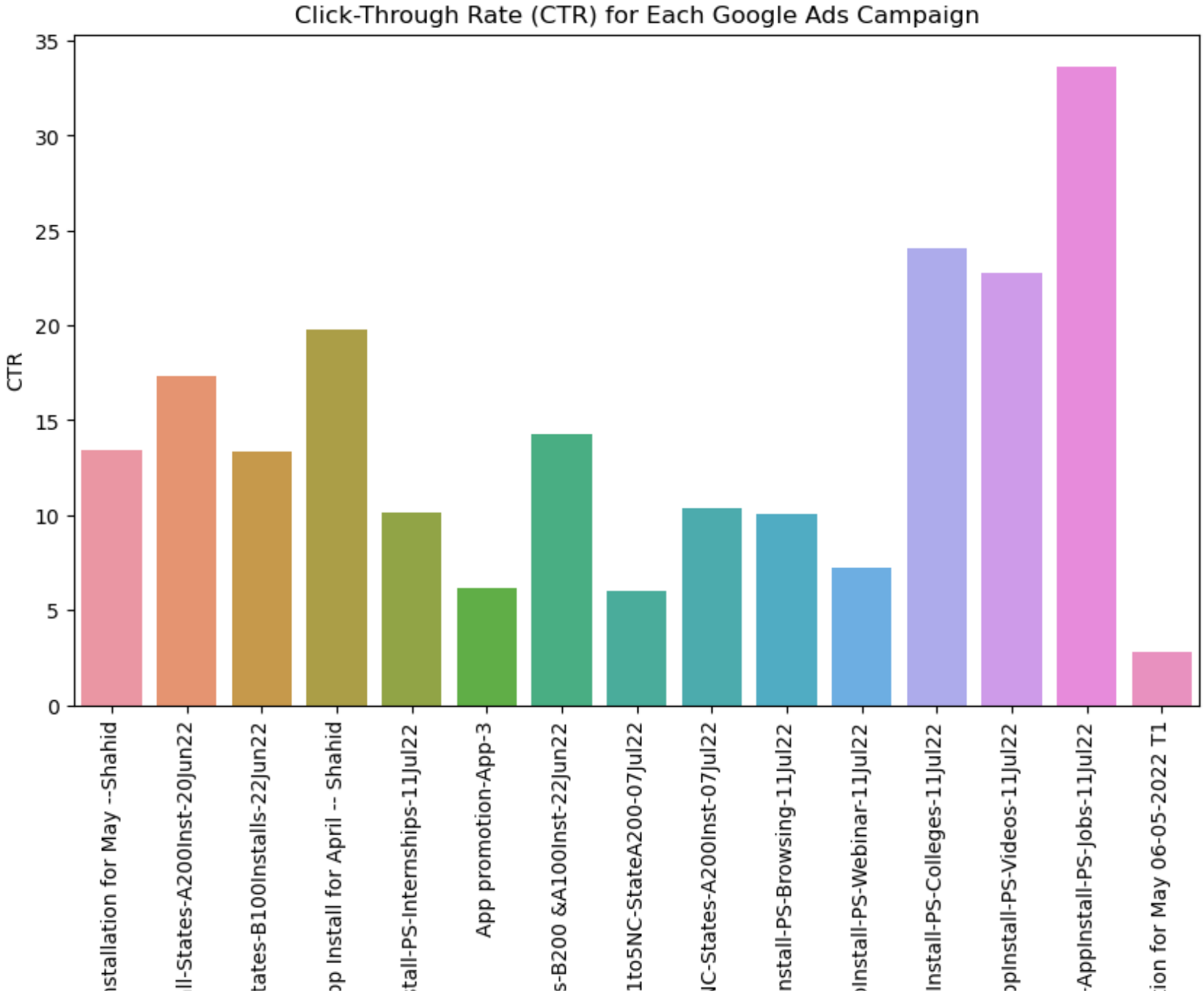
```
# Visualize Cost per Conversion
plt.figure(figsize=(10, 6))
sns.barplot(data=Google_Ads_Report, x='Session Google Ads campaign', y='Cost per conversion')
plt.title('Cost per Conversion for Each Google Ads Campaign')
plt.xlabel('Google Ads Campaign')
plt.ylabel('Cost per Conversion')
plt.xticks(rotation=90)
plt.show()
```



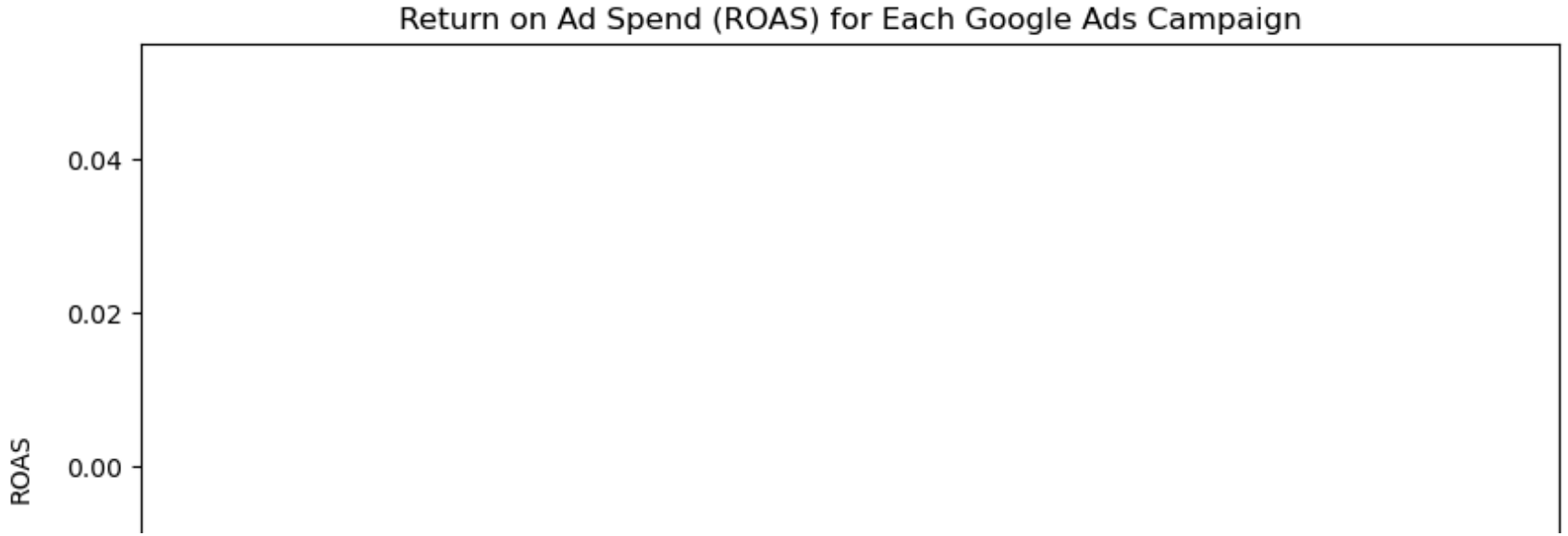

```
# Calculate Click-Through Rate (CTR)
Google_Ads_Report['CTR'] = Google_Ads_Report['Google Ads clicks'] / Google_Ads_Report['Sessions']
```



```
# Visualize Click-Through Rate (CTR)
plt.figure(figsize=(10, 6))
sns.barplot(data=Google_Ads_Report, x='Session Google Ads campaign', y='CTR')
plt.title('Click-Through Rate (CTR) for Each Google Ads Campaign')
plt.xlabel('Google Ads Campaign')
plt.ylabel('CTR')
plt.xticks(rotation=90)
plt.show()
```

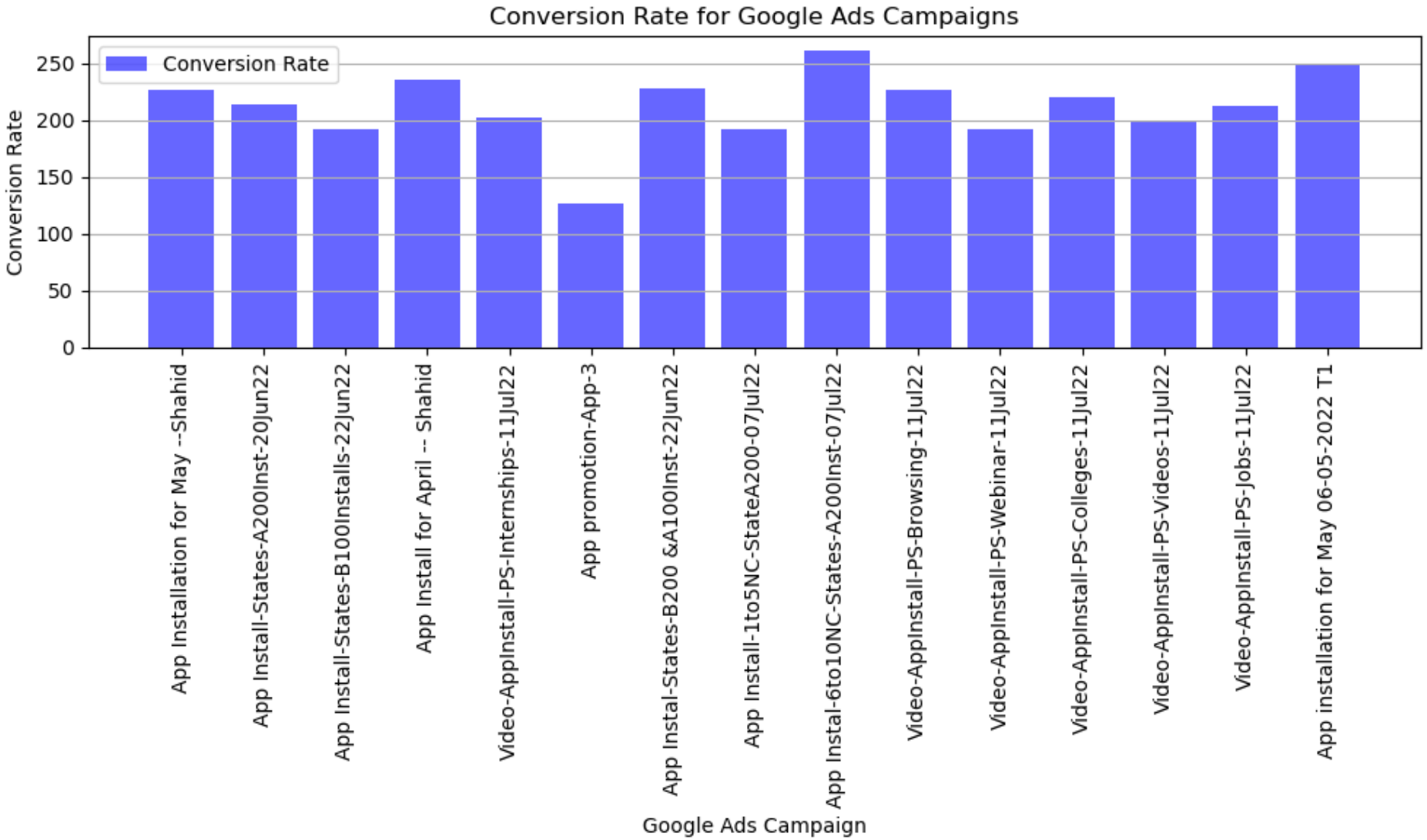

Click-Through Rate (CTR) for Each Google Ads Campaign
CTR
0
5
10
15
20
25
30
35
istallation for May --Shahid
ll-States-A200Inst-20Jun22
tates-B100Installs-22Jun22
op Install for April -- Shahid
tall-PS-Internships-11Jul22
App promotion-App-3
s-B200 &A100Inst-22Jun22
1to5NC-StateA200-07Jul22
IC-States-A200Inst-07Jul22
nstall-PS-Browsing-11Jul22
Install-PS-Webinar-11Jul22
Install-PS-Colleges-11Jul22
opInstall-PS-Videos-11Jul22
-AppInstall-PS-Jobs-11Jul22
ion for May 06-05-2022 T1

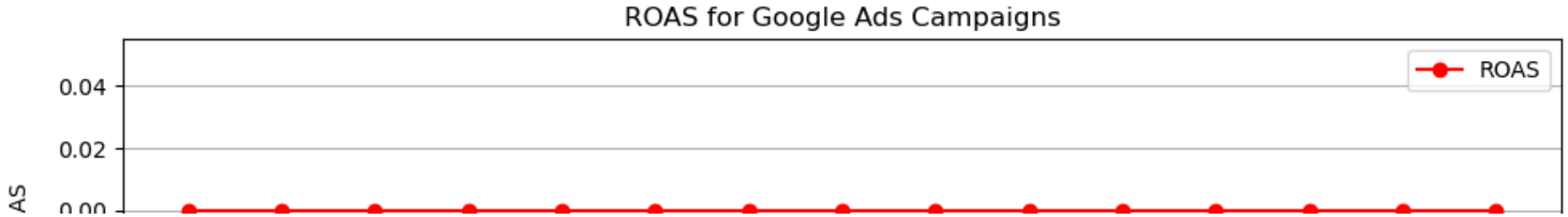
```
# Visualize Return on Ad Spend (ROAS)
plt.figure(figsize=(10, 6))
sns.barplot(data=Google_Ads_Report, x='Session Google Ads campaign', y='Return on ad spend')
plt.title('Return on Ad Spend (ROAS) for Each Google Ads Campaign')
plt.xlabel('Google Ads Campaign')
plt.ylabel('ROAS')
plt.xticks(rotation=90)
plt.show()
```



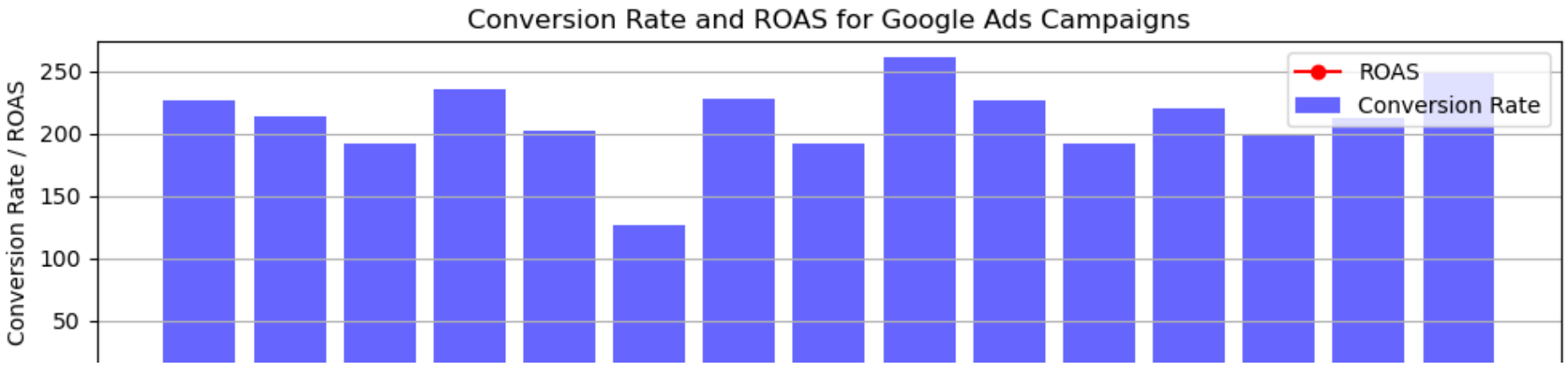

```
# Calculate Conversion Rate
Google_Ads_Report['Conversion Rate'] = (Google_Ads_Report['Conversions'] / Google_Ads_Report['Users']) * 100

# Calculate ROAS
Google_Ads_Report['ROAS'] = Google_Ads_Report['Total revenue'] / Google_Ads_Report['Google Ads cost']

import matplotlib.pyplot as plt

# Create a bar chart for Conversion Rate
plt.figure(figsize=(10, 6))
plt.bar(Google_Ads_Report['Session Google Ads campaign'], Google_Ads_Report['Conversion Rate'], color='b', alpha=0.6, label='Conversior
plt.xlabel('Google Ads Campaign')
plt.ylabel('Conversion Rate')
plt.title('Conversion Rate for Google Ads Campaigns')
plt.xticks(rotation=90)
plt.legend()
plt.grid(axis='y')
plt.tight_layout()

# Create a line plot for ROAS
plt.figure(figsize=(10, 6))
```

```
plt.plot(Google_Ads_Report['Session Google Ads campaign'], Google_Ads_Report['ROAS'], color='r', marker='o', label='ROAS')
plt.xlabel('Google Ads Campaign')
plt.ylabel('ROAS')
plt.title('ROAS for Google Ads Campaigns')
plt.xticks(rotation=90)
plt.legend()
plt.grid(axis='y')
plt.tight_layout()

plt.show()
```


Conversion Rate for Google Ads Campaigns
Conversion Rate
250
200
150
100
50
0
App Installation for May --Shahid
App Install-States-A200Inst-20Jun22
App Install-States-B100Installs-22Jun22
App Install for April -- Shahid
Video-AppInstall-PS-Internships-11Jul22
App promotion-App-3
App Instal-States-B200 &A100Inst-22Jun22
App Install-1to5NC-StateA200-07Jul22
App Instal-6to10NC-States-A200Inst-07Jul22
Video-AppInstall-PS-Browsing-11Jul22
Video-AppInstall-PS-Webinar-11Jul22
Video-AppInstall-PS-Colleges-11Jul22
Video-AppInstall-PS-Videos-11Jul22
Video-AppInstall-PS-Jobs-11Jul22
App installation for May 06-05-2022 T1
Google Ads Campaign


ROAS for Google Ads Campaigns
ROAS
0.04
0.02
AS

```
# Assuming you have loaded the dataset into a DataFrame named 'google_ads_df'
# Let's calculate the Conversion Rate and ROAS
Google_Ads_Report['Conversion Rate'] = (Google_Ads_Report['Conversions'] / Google_Ads_Report['Users']) * 100
Google_Ads_Report['ROAS'] = Google_Ads_Report['Total revenue'] / Google_Ads_Report['Google Ads cost']

# Visualize the Conversion Rate and ROAS for each campaign
plt.figure(figsize=(10, 6))
plt.bar(Google_Ads_Report['Session Google Ads campaign'], Google_Ads_Report['Conversion Rate'], color='b', alpha=0.6, label='Conversion
plt.plot(Google_Ads_Report['Session Google Ads campaign'], Google_Ads_Report['ROAS'], color='r', marker='o', label='ROAS')
plt.xlabel('Google Ads Campaign')
plt.ylabel('Conversion Rate / ROAS')
plt.title('Conversion Rate and ROAS for Google Ads Campaigns')
plt.xticks(rotation=90)
plt.legend()
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



Recommendations:

Marketing Channel Performance:

Display and Organic Search channels have the highest number of engaged sessions, making them effective for attracting and engaging users. Paid Search and Organic Social channels have relatively low engagement rates, indicating the need for improvement in their targeting and messaging strategies. Direct and Unassigned channels have moderate engagement rates, suggesting the potential for optimization to increase user engagement.

User Engagement by Country:

analyzed user engagement by country and found variations in engaged sessions. Certain countries showed higher engagement, indicating potential opportunities for targeted marketing campaigns in those regions.

Trategic Plan:

Target high-engagement countries with tailored marketing campaigns to capitalize on the already engaged user base. Allocate more resources to Display and Organic Search channels, as they have shown higher engagement rates and conversions. Optimize marketing strategies in Paid Search and Organic Social channels to improve their engagement rates and ROI.

By focusing on high-engagement countries and channels, the marketing efforts can yield better returns on investment. A/B testing will help identify the most effective marketing strategies and optimize the budget allocation accordingly. Improving the user funnel and engagement rates can lead to increased conversions and revenue generation.

Overall, the comprehensive analysis provides valuable insights into the marketing campaign's performance and user engagement. By implementing the recommended strategies

Overview:

1. **Report:** A detailed report summarizing the analysis, key findings, and recommendations. It will explain who the customers are, their interests, and how they engage with the platform. It will also suggest ways to improve sales and marketing strategies.
2. **Visualizations:** Charts and graphs to show important information visually. You will see colorful pictures that make it easier to understand the data and trends.
3. **Code/Scripts:** code used to analyze and visualize the data. It will help others see how I arrived at the results.

Overall, you'll get a clear understanding of your customers and marketing campaigns, along with actionable steps to boost sales and make informed decisions. The visuals will make everything easier to grasp!