```
import pandas as pd
data = pd.read_csv("data-export.csv")
print(data.head())
```

```
           # ----------------------------------------  \
0  Session primary channel group (Default channel...
1                                             Direct
2                                     Organic Social
3                                             Direct
4                                     Organic Social

                Unnamed: 1 Unnamed: 2 Unnamed: 3       Unnamed: 4  \
0  Date + hour (YYYYMMDDHH)      Users   Sessions  Engaged sessions
1                2024041623        237        300               144
2                2024041719        208        267               132
3                2024041723        188        233               115
4                2024041718        187        256               125

                           Unnamed: 5                Unnamed: 6  \
0  Average engagement time per session  Engaged sessions per user
1                    47.526666666666700        0.6075949367088610
2                     32.09737827715360        0.6346153846153850
3                     39.93991416309010        0.6117021276595740
4                           32.16015625        0.6684491978609630

          Unnamed: 7          Unnamed: 8   Unnamed: 9
0  Events per session      Engagement rate  Event count
1   4.673333333333330                0.48         1402
2   4.295880149812730   0.4943820224719100         1147
3   4.587982832618030  0.49356223175965700         1069
4            4.078125           0.48828125         1044
```

```
new_header = data.iloc[0]  # grab the first row for the header
data = data[1:]  # take the data less the header row
data.columns = new_header  # set the header row as the df header
data.reset_index(drop=True, inplace=True)

print(data.head())
```

```
0 Session primary channel group (Default channel group)  \
0                                             Direct
1                                     Organic Social
2                                             Direct
3                                     Organic Social
4                                     Organic Social

0 Date + hour (YYYYMMDDHH) Users Sessions Engaged sessions  \
0               2024041623   237      300              144
1               2024041719   208      267              132
2               2024041723   188      233              115
3               2024041718   187      256              125
4               2024041720   175      221              112

0 Average engagement time per session Engaged sessions per user  \
0                  47.526666666666700        0.6075949367088610
1                   32.09737827715360        0.6346153846153850
2                   39.93991416309010        0.6117021276595740
```

```
3                      32.16015625        0.6684491978609630
4                 46.918552036199100                    0.64

0 Events per session      Engagement rate Event count
0  4.673333333333330                 0.48        1402
1  4.295880149812730   0.4943820224719100        1147
2  4.587982832618030  0.49356223175965700        1069
3           4.078125           0.48828125        1044
4  4.529411764705880   0.5067873303167420        1001
```

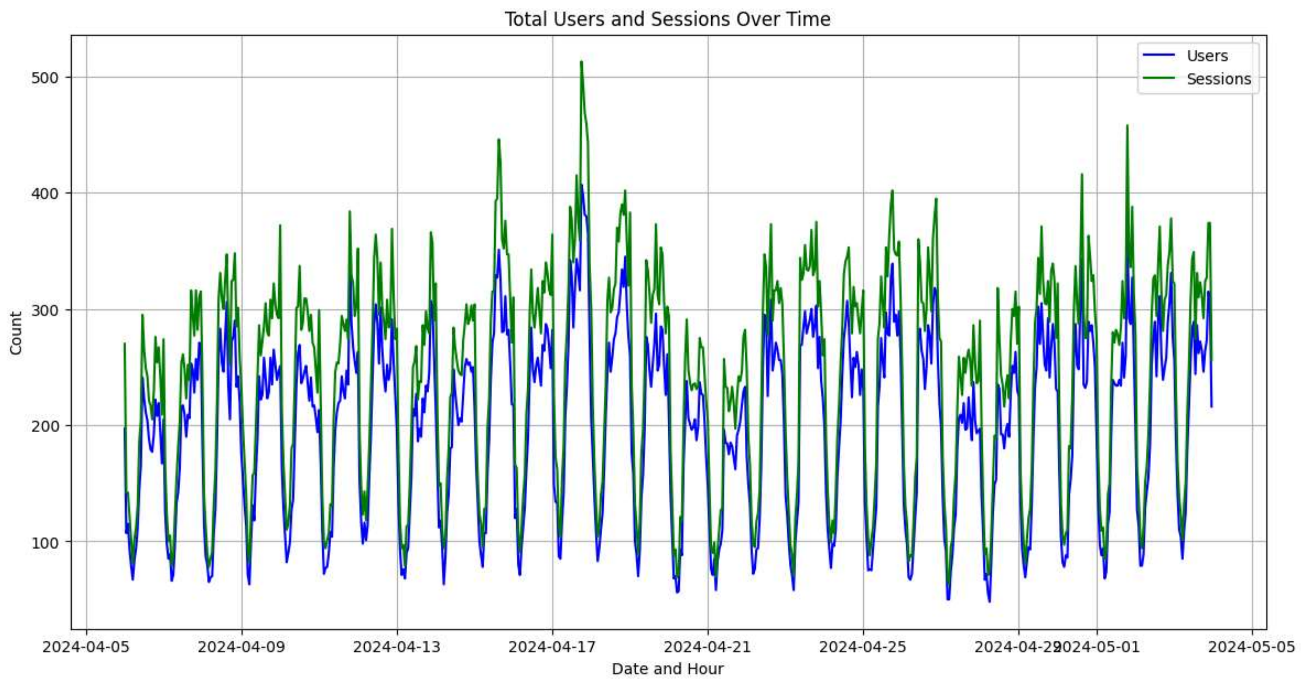
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3182 entries, 0 to 3181
Data columns (total 10 columns):
 #   Column                                              Non-Null Count  Dtype 
---  ------                                              --------------  ----- 
 0   Session primary channel group (Default channel group)  3182 non-null   object
 1   Date + hour (YYYYMMDDHH)                            3182 non-null   object
 2   Users                                               3182 non-null   object
 3   Sessions                                            3182 non-null   object
 4   Engaged sessions                                    3182 non-null   object
 5   Average engagement time per session                 3182 non-null   object
 6   Engaged sessions per user                           3182 non-null   object
 7   Events per session                                  3182 non-null   object
 8   Engagement rate                                     3182 non-null   object
 9   Event count                                         3182 non-null   object
dtypes: object(10)
memory usage: 248.7+ KB
```

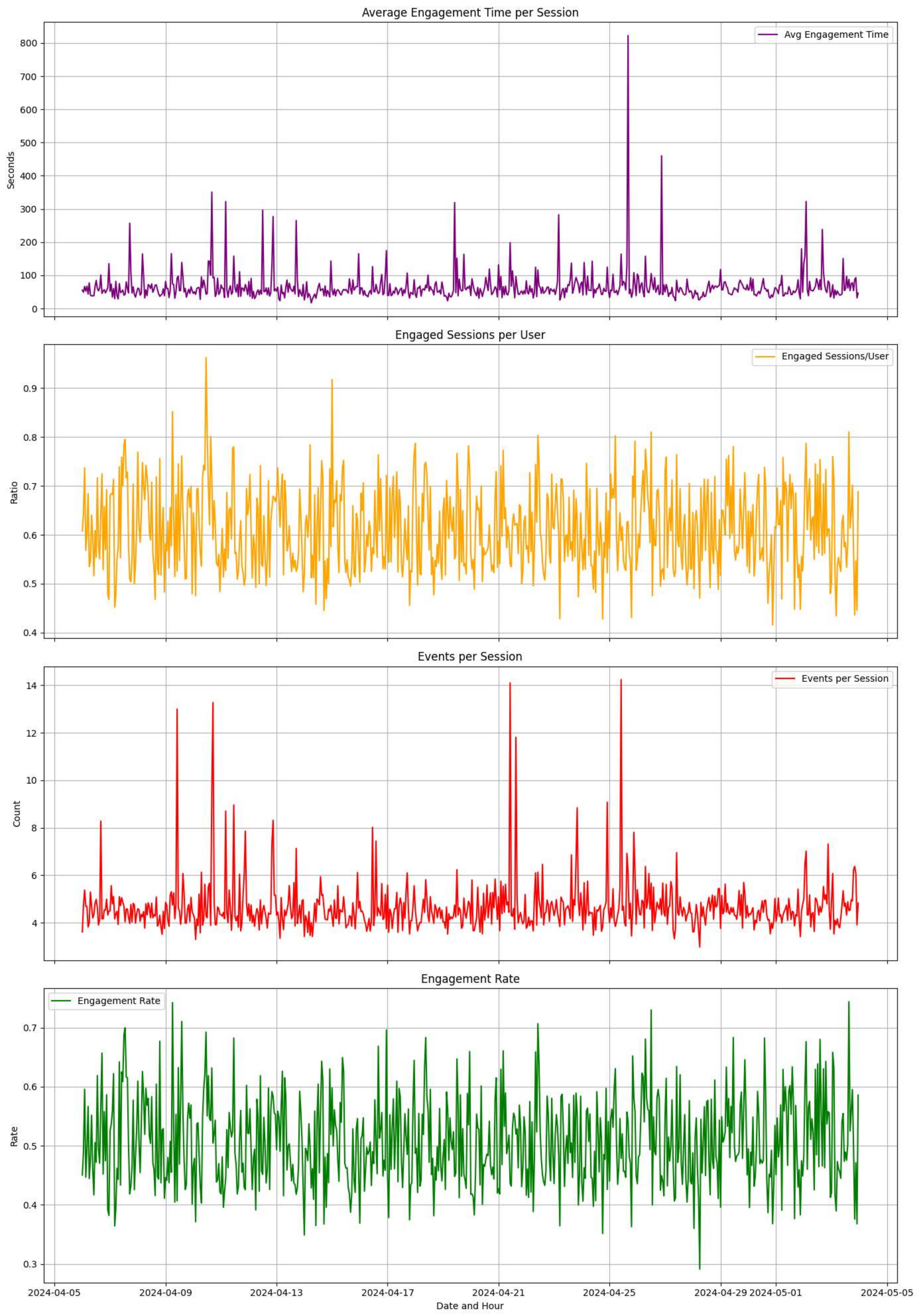
```
print(data.describe())
```

```
0      Session primary channel group (Default channel group)  \
count                                               3182   
unique                                                 7   
top                                               Direct   
freq                                                 672   

0      Date + hour (YYYYMMDDHH) Users Sessions Engaged sessions  \
count                      3182  3182     3182             3182   
unique                      672   147      180              103   
top                  2024042417     1        1                0   
freq                          6   335      340              393   

0      Average engagement time per session Engaged sessions per user  \
count                                 3182                      3182   
unique                                2823                       808   
top                                      0                         0   
freq                                   170                       393   

0      Events per session Engagement rate Event count  
count                3182            3182        3182  
unique               2025             986         678  
top                     1               0           1  
freq                  133             393         115  
```

```
data['Date + hour (YYYYMMDDHH)'] = pd.to_datetime(data['Date + hour (YYYYMMDDHH)'], forma
data['Users'] = pd.to_numeric(data['Users'])
data['Sessions'] = pd.to_numeric(data['Sessions'])

# group data by date and sum up the users and sessions
grouped_data = data.groupby(data['Date + hour (YYYYMMDDHH)']).agg({'Users': 'sum', 'Sessi


import matplotlib.pyplot as plt

# plotting the aggregated users and sessions over time
plt.figure(figsize=(14, 7))
plt.plot(grouped_data.index, grouped_data['Users'], label='Users', color='blue')
plt.plot(grouped_data.index, grouped_data['Sessions'], label='Sessions', color='green')
plt.title('Total Users and Sessions Over Time')
plt.xlabel('Date and Hour')
plt.ylabel('Count')
plt.legend()
plt.grid(True)
plt.show()
```

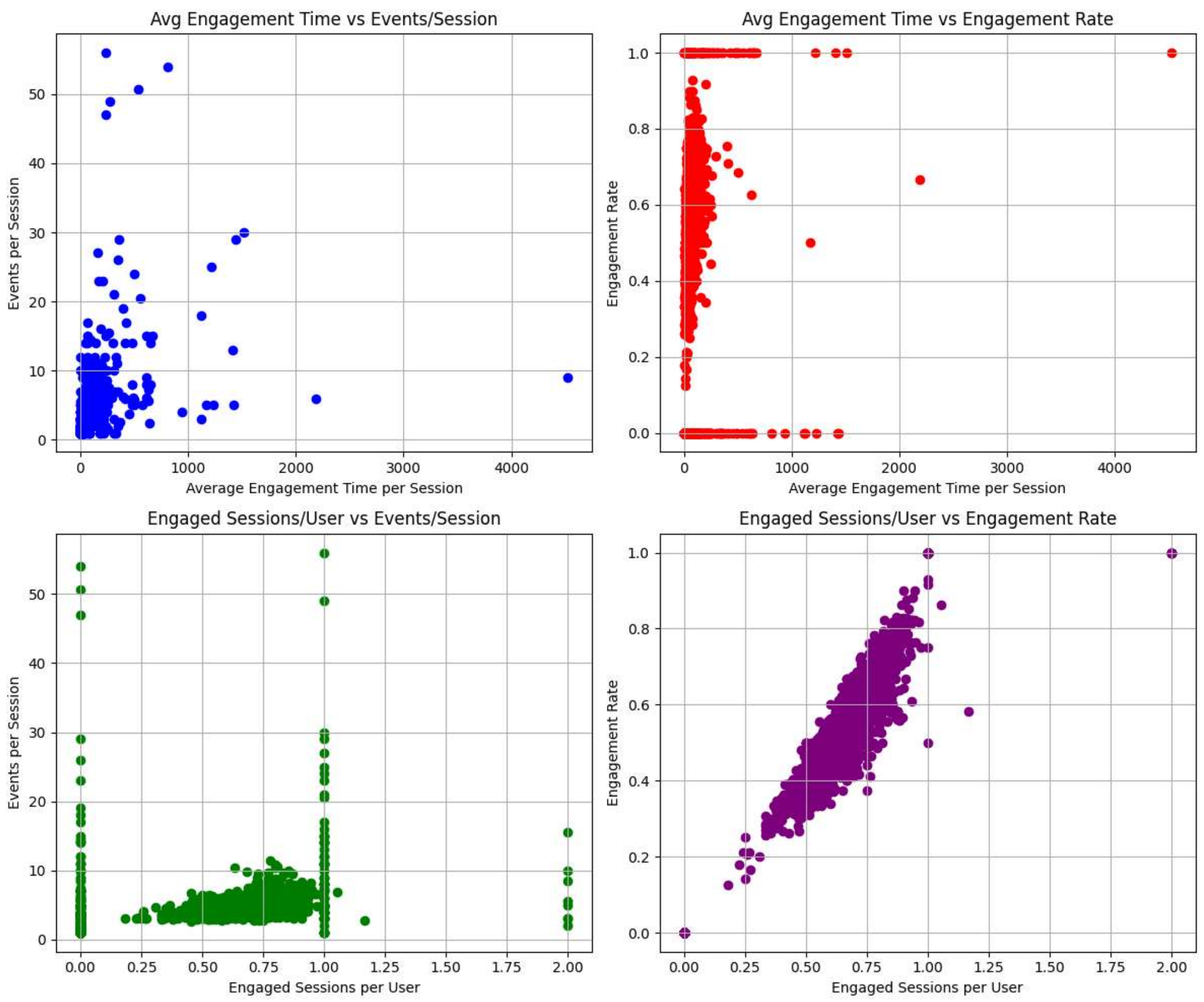


```
# convert relevant columns to numeric for engagement analysis
data['Engaged sessions'] = pd.to_numeric(data['Engaged sessions'])
data['Average engagement time per session'] = pd.to_numeric(data['Average engagement time
data['Engaged sessions per user'] = pd.to_numeric(data['Engaged sessions per user'])
data['Events per session'] = pd.to_numeric(data['Events per session'])
data['Engagement rate'] = pd.to_numeric(data['Engagement rate'])

# group data by date and calculate mean for engagement metrics
engagement_metrics = data.groupby(data['Date + hour (YYYYMMDDHH)']).agg({
    'Average engagement time per session': 'mean',
    'Engaged sessions per user': 'mean',
    'Events per session': 'mean',
    'Engagement rate': 'mean'
})

# plotting engagement metrics
fig, ax = plt.subplots(4, 1, figsize=(14, 20), sharex=True)

ax[0].plot(engagement_metrics.index, engagement_metrics['Average engagement time per sess
ax[0].set_title('Average Engagement Time per Session')
ax[0].set_ylabel('Seconds')

ax[1].plot(engagement_metrics.index, engagement_metrics['Engaged sessions per user'], lab
ax[1].set_title('Engaged Sessions per User')
ax[1].set_ylabel('Ratio')

ax[2].plot(engagement_metrics.index, engagement_metrics['Events per session'], label='Eve
ax[2].set_title('Events per Session')
ax[2].set_ylabel('Count')

ax[3].plot(engagement_metrics.index, engagement_metrics['Engagement rate'], label='Engage
ax[3].set_title('Engagement Rate')
ax[3].set_ylabel('Rate')
ax[3].set_xlabel('Date and Hour')

for a in ax:
    a.legend()
    a.grid(True)

plt.tight_layout()
plt.show()
```

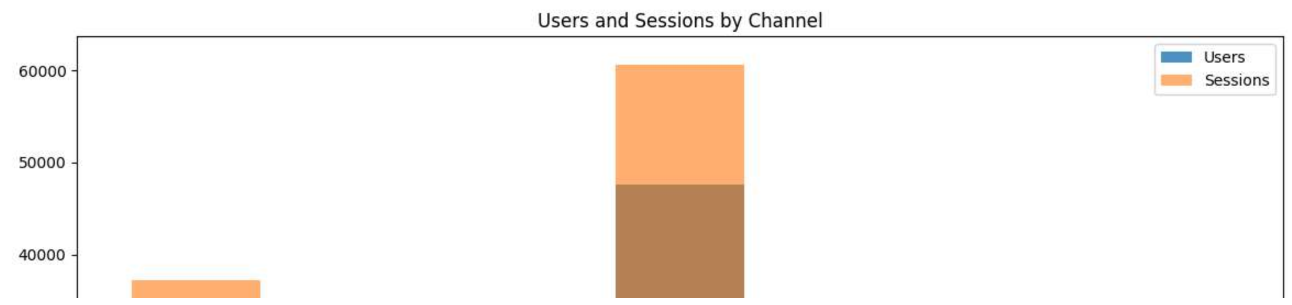


Average Engagement Time per Session
Avg Engagement Time
Seconds
Engaged Sessions per User
Engaged Sessions/User
Ratio
Events per Session
Events per Session
Count
Engagement Rate
Engagement Rate
Rate
2024-04-05
2024-04-09
2024-04-13
2024-04-17
2024-04-21
2024-04-25
2024-04-29
2024-05-01
2024-05-05
Date and Hour

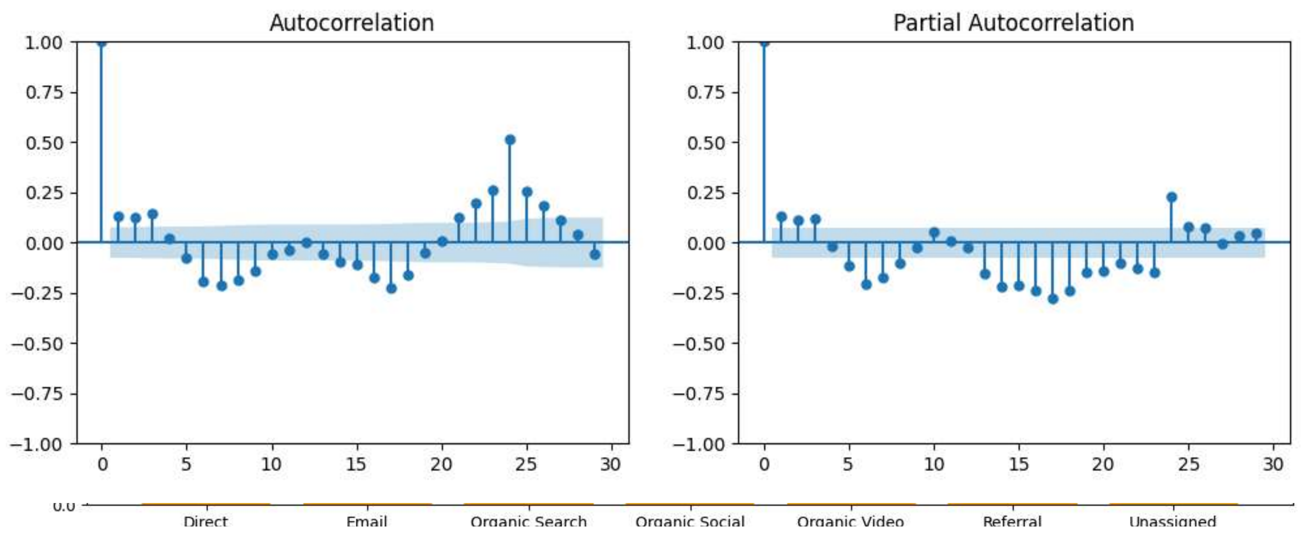
```
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# plot 1: average engagement time vs events per session
axes[0, 0].scatter(data['Average engagement time per session'], data['Events per session'
axes[0, 0].set_title('Avg Engagement Time vs Events/Session')
axes[0, 0].set_xlabel('Average Engagement Time per Session')
axes[0, 0].set_ylabel('Events per Session')
axes[0, 0].grid(True)  # enable grid

# plot 2: average engagement time vs engagement rate
axes[0, 1].scatter(data['Average engagement time per session'], data['Engagement rate'],
axes[0, 1].set_title('Avg Engagement Time vs Engagement Rate')
axes[0, 1].set_xlabel('Average Engagement Time per Session')
axes[0, 1].set_ylabel('Engagement Rate')
axes[0, 1].grid(True)

# plot 3: engaged sessions per user vs events per session
axes[1, 0].scatter(data['Engaged sessions per user'], data['Events per session'], color='
axes[1, 0].set_title('Engaged Sessions/User vs Events/Session')
axes[1, 0].set_xlabel('Engaged Sessions per User')
axes[1, 0].set_ylabel('Events per Session')
axes[1, 0].grid(True)

# plot 4: engaged sessions per user vs engagement rate
axes[1, 1].scatter(data['Engaged sessions per user'], data['Engagement rate'], color='pur
axes[1, 1].set_title('Engaged Sessions/User vs Engagement Rate')
axes[1, 1].set_xlabel('Engaged Sessions per User')
axes[1, 1].set_ylabel('Engagement Rate')
axes[1, 1].grid(True)

plt.tight_layout()
plt.show()
```


Avg Engagement Time vs Events/Session
Events per Session
Average Engagement Time per Session
Avg Engagement Time vs Engagement Rate
Engagement Rate
Average Engagement Time per Session
Engaged Sessions/User vs Events/Session
Events per Session
Engaged Sessions per User
Engaged Sessions/User vs Engagement Rate
Engagement Rate
Engaged Sessions per User

```
# group data by channel and aggregate necessary metrics
channel_performance = data.groupby('Session primary channel group (Default channel group)
    'Users': 'sum',
    'Sessions': 'sum',
    'Engaged sessions': 'sum',
    'Engagement rate': 'mean',
    'Events per session': 'mean'
})

# normalize engagement rate and events per session for comparison
channel_performance['Normalized Engagement Rate'] = channel_performance['Engagement rate'
channel_performance['Normalized Events per Session'] = channel_performance['Events per se

# plotting channel performance metrics
fig, ax = plt.subplots(3, 1, figsize=(12, 18))

# users and sessions by channel
ax[0].bar(channel_performance.index, channel_performance['Users'], label='Users', alpha=0
ax[0].bar(channel_performance.index, channel_performance['Sessions'], label='Sessions', a
ax[0].set_title('Users and Sessions by Channel')
ax[0].set_ylabel('Count')
ax[0].legend()

# normalized engagement rate by channel
ax[1].bar(channel_performance.index, channel_performance['Normalized Engagement Rate'], c
ax[1].set_title('Normalized Engagement Rate by Channel')
ax[1].set_ylabel('Normalized Rate')

# normalized events per session by channel
ax[2].bar(channel_performance.index, channel_performance['Normalized Events per Session']
ax[2].set_title('Normalized Events per Session by Channel')
ax[2].set_ylabel('Normalized Count')

plt.tight_layout()
plt.show()
```



```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
time_series_data = grouped_data['Sessions'].asfreq('H').fillna(method='ffill')
seasonal_period = 24

differenced_series = time_series_data.diff().dropna()

# plot ACF and PACF of time series
fig, axes = plt.subplots(1, 2, figsize=(12, 4))
plot_acf(differenced_series, ax=axes[0])
plot_pacf(differenced_series, ax=axes[1])
plt.show()
```



```
from statsmodels.tsa.statespace.sarimax import SARIMAX

time_series_data = grouped_data['Sessions'].asfreq('H').fillna(method='ffill')
seasonal_period = 24


sarima_model = SARIMAX(time_series_data,
                       order=(1, 1, 1),
                       seasonal_order=(1, 1, 1, seasonal_period))
```