📊 Website Performance Analysis Report

1. Dataset Overview

- **Source**: watching_customer.csv
- **Time Range**: April – May 2024 (hourly granularity)
- **Rows**: 3,182 observations
- **Key Metrics**:
    - Users, Sessions, Engaged Sessions
    - Average Engagement Time per Session
    - Events per Session, Engagement Rate, Event Count
    - Channel Group (e.g., Direct, Organic Social, Organic Search, etc.)

jupyter website_performance_analysis Last Checkpoint: a day ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df = pd.read_csv("watching_customer.csv")
```

In [4]:
```
df.head(5)
```

Out[4]:

# ---------- ---------- ---------- ---	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
0	Session primary channel group (Default channel...	Date + hour (YYYYMMDDHH)	Users	Sessions	Engaged sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count
1	Direct	2024041623	237	300	144	47.526666666666700	0.6075949367088610	4.673333333333330	0.48	1402
2	Organic Social	2024041719	208	267	132	32.09737827715360	0.6346153846153850	4.295880149812730	0.4943820224719100	1147
3	Direct	2024041723	188	233	115	39.93991416309010	0.6117021276595740	4.587982832618030	0.49356223175965700	1069
4	Organic Social	2024041718	187	256	125	32.16015625	0.6684491978609630	4.078125	0.48828125	1044

2. Data Cleaning

- Removed header row that was mistakenly included as data
- Standardized column names
- Converted DateHour (format: YYYYMMDDHH) to datetime
- Parsed numeric columns from string to float/int
- Extracted **Hour of Day** for time-based analysis

In [5]:
```
df.columns = df.iloc[0]
```

In [5]:
```
df.head()
```

Out[5]:

	Session primary channel group (Default channel group)	Date + hour (YYYYMMDDHH)	Users	Sessions	Engaged sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count
0	Session primary channel group (Default channel...	Date + hour (YYYYMMDDHH)	Users	Sessions	Engaged sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count
1	Direct	2024041623	237	300	144	47.526666666666700	0.6075949367088610	4.673333333333330	0.48	1402
2	Organic Social	2024041719	208	267	132	32.09737827715360	0.6346153846153850	4.295880149812730	0.4943820224719100	1147
3	Direct	2024041723	188	233	115	39.93991416309010	0.6117021276595740	4.587982832618030	0.49356223175965700	1069
4	Organic Social	2024041718	187	256	125	32.16015625	0.6684491978609630	4.078125	0.48828125	1044

In [4]:
```
df = df.drop(index=0).reset_index(drop=True)
```

In [5]:
```
df.columns = ["channel group","DateHour", "Users","Sessions", "Engaged sessions", "Average engagement time per session","Engaged
              "Events per session", "Engagement rate","Event count" ]
```

In [15]:
```
df.head()
```

Out[15]:

	channel group	DateHour	Users	Sessions	Engaged sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count
0	Direct	2024041623	237	300	144	47.526666666666700	0.6075949367088610	4.673333333333330	0.48	1402
1	Organic Social	2024041719	208	267	132	32.09737827715360	0.6346153846153850	4.295880149812730	0.4943820224719100	1147
2	Direct	2024041723	188	233	115	39.93991416309010	0.6117021276595740	4.587982832618030	0.49356223175965700	1069
3	Organic Social	2024041718	187	256	125	32.16015625	0.6684491978609630	4.078125	0.48828125	1044
4	Organic Social	2024041720	175	221	112	46.918552036199100	0.64	4.529411764705880	0.5067873303167420	1001

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3182 entries, 0 to 3181
Data columns (total 10 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   channel group                         3182 non-null   object
 1   DateHour                              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
```

Cleaning And Data Visualization:

Cleaning and Data Visualization

In [6]:
```
df["DateHour"]=pd.to_datetime(df["DateHour"], format="%Y%m%d%H", errors='coerce')
numeric_cols = df.columns.drop(["channel group", "DateHour"])
df[numeric_cols]= df[numeric_cols].apply(pd.to_numeric, errors='coerce')
df["Hours"] =df["DateHour"].dt.hour
```

In [19]:
```
df.head()
```

Out[19]:

	channel group	DateHour	Users	Sessions	Engaged sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count	Hours
0	Direct	2024-04-16 23:00:00	237	300	144	47.526667	0.607595	4.673333	0.480000	1402	23
1	Organic Social	2024-04-17 19:00:00	208	267	132	32.097378	0.634615	4.295880	0.494382	1147	19
2	Direct	2024-04-17 23:00:00	188	233	115	39.939914	0.611702	4.587983	0.493562	1069	23
3	Organic Social	2024-04-17 18:00:00	187	256	125	32.160156	0.668449	4.078125	0.488281	1044	18
4	Organic Social	2024-04-17 20:00:00	175	221	112	46.918552	0.640000	4.529412	0.506787	1001	20

3. Key Insights

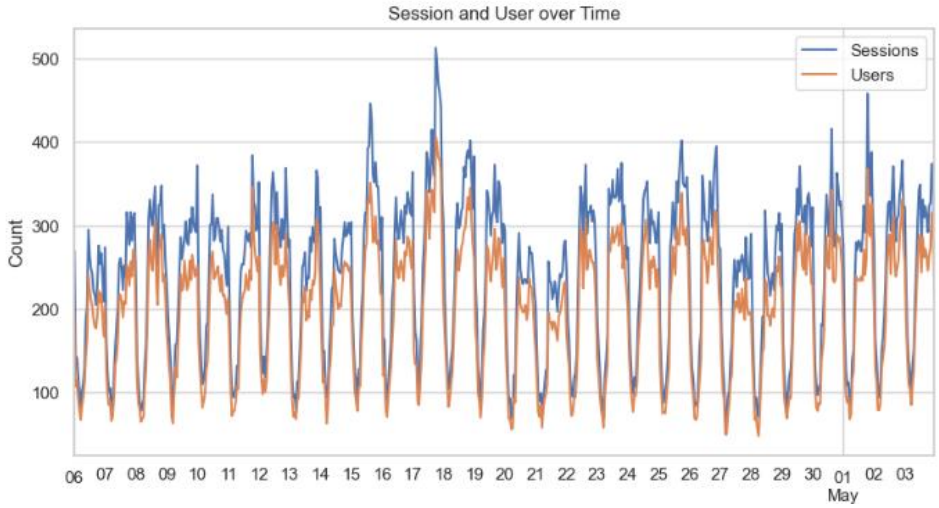
📈 Time Trends

- **Sessions and Users** show clear daily/weekly seasonality, peaking during business hours and weekdays.

Session and User over Time

```
In [24]: sns.set(style="whitegrid")
plt.figure(figsize=(10,5))
df.groupby("DateHour")[["Sessions", "Users"]].sum().plot(ax=plt.gca())
plt.title("Session and User over Time")
plt.xlabel("DateHour")
plt.ylabel("Count")

plt.show()
```



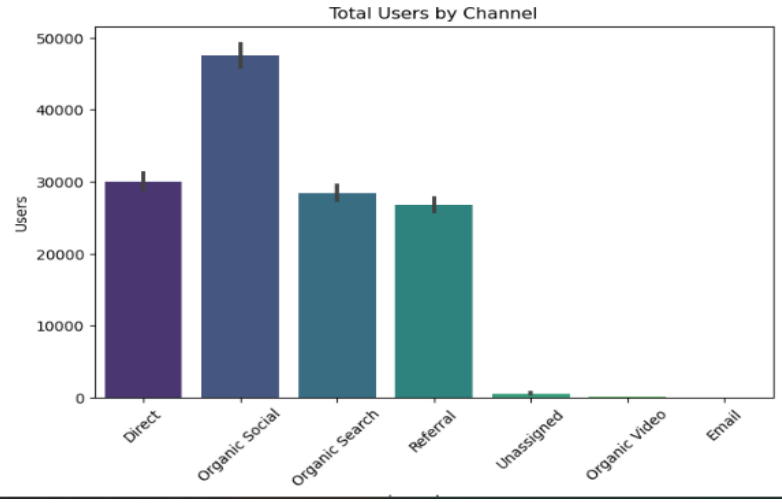
- Overall traffic appears stable with no major anomalies.

📦 Channel Performance

- **Total Users**: Organic Social and Organic Search dominate total user acquisition.

Total User by Channel

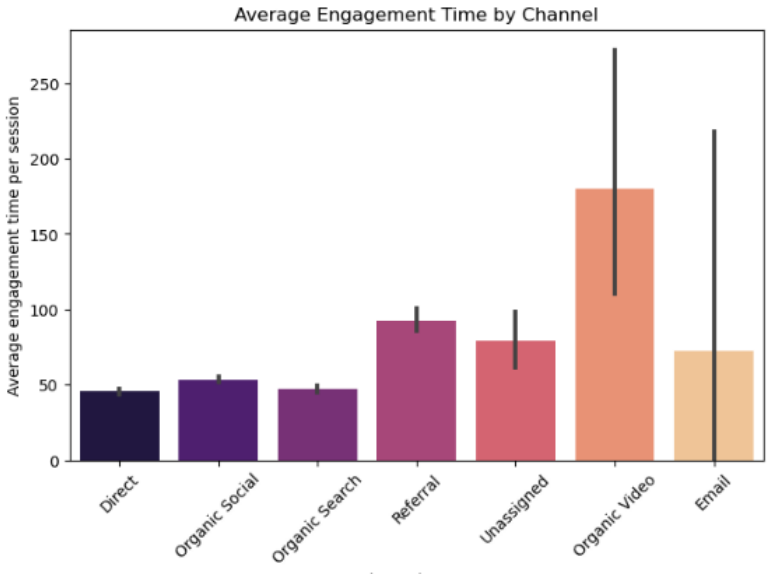
```
In [11]: plt.figure(figsize=(8,5))
sns.barplot(data=df, x="channel group", y="Users", estimator=np.sum, palette="viridis")
plt.title("Total Users by Channel")
plt.xticks(rotation=45)
plt.show()
```



- **Engagement Time**:
    - **Direct** traffic has the **highest average engagement time** (~200–250 seconds).
    - **Organic Social** shows **shorter sessions** (~30–50 seconds).

Average Engagement Time by Channel

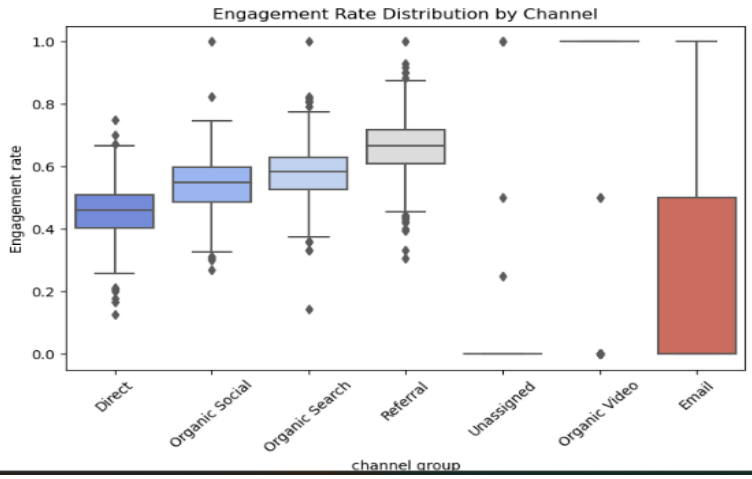
```
In [13]: plt.figure(figsize=(8,5))
sns.barplot(data=df, x="channel group", y="Average engagement time per session", estimator=np.mean, palette="magma")
plt.title("Average Engagement Time by Channel")
plt.xticks(rotation=45)
plt.show()
```



- **Engagement Rate Distribution**:
    - Most channels show engagement rates between **0.4–0.6**, with low variance.
    - **Email** and **Direct** show slightly higher median engagement.

Engagement Rate Distribution by Channel

```
In [14]: plt.figure(figsize=(8,5))
sns.boxplot(data=df, x="channel group", y="Engagement rate", palette="coolwarm")
plt.title("Engagement Rate Distribution by Channel")
plt.xticks(rotation=45)
plt.show()
```



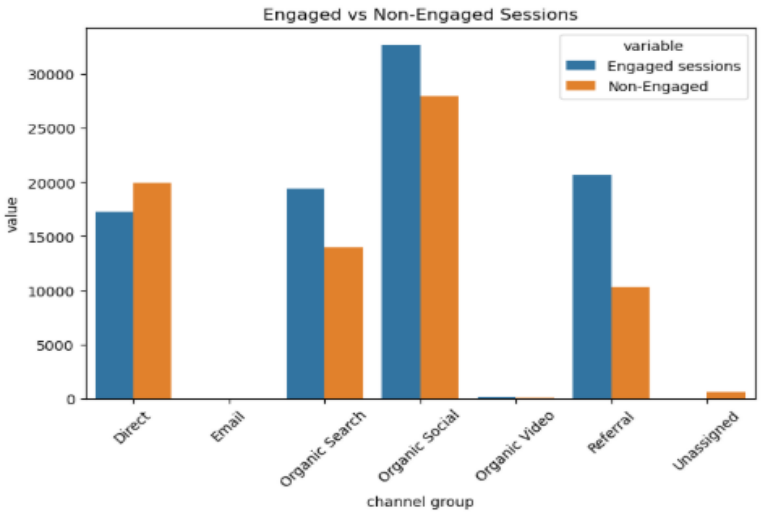
🔄 Engaged vs Non-Engaged Sessions

- Across all channels, **engaged sessions** represent a significant portion.
- **Direct** and **Organic Search** have the highest absolute number of engaged sessions.

Engaged vs Non-Engaged Session

```
In [7]: session_df  = df.groupby("channel group")[["Sessions", "Engaged sessions"]].sum().reset_index()
session_df["Non-Engaged"] = session_df["Sessions"] - session_df["Engaged sessions"]
session_df_melted = session_df.melt(id_vars ="channel group", value_vars=["Engaged sessions", "Non-Engaged"])

plt.figure(figsize=(8,5))
sns.barplot(data=session_df_melted, x="channel group", y="value", hue="variable")
plt.title("Engaged vs Non-Engaged Sessions")
plt.xticks(rotation=45)
plt.show()
```



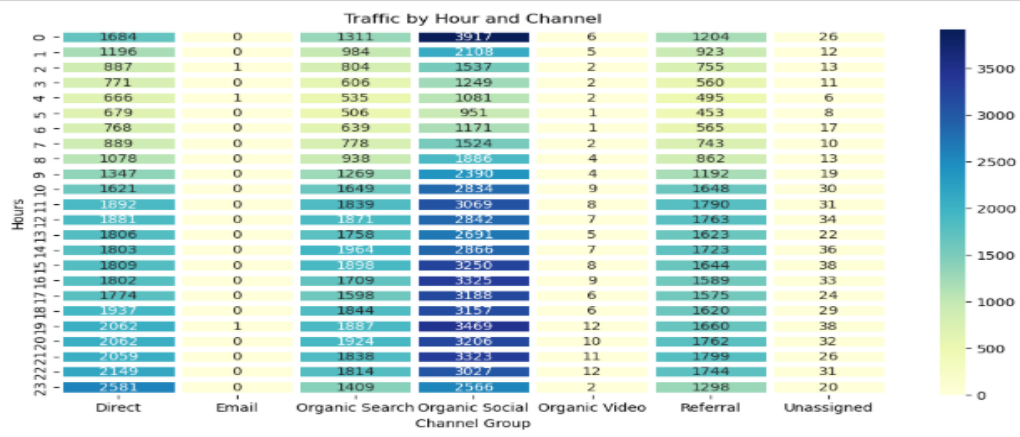
⏰ Hourly Traffic Heatmap

- Peak traffic occurs between **18:00–22:00**, especially for Organic Social and Direct.
- Minimal activity between **00:00–06:00**.
- **Email** and **Referral** show consistent but lower volume throughout the day.

Traffic by Hour and Channel

```
In [11]: heatmap_data = df.groupby(["Hours", "channel group"])["Sessions"].sum().unstack().fillna(0)

plt.figure(figsize=(12,6))
sns.heatmap(heatmap_data, cmap="YlGnBu", linewidth=5, annot=True, fmt=".0f")
plt.title("Traffic by Hour and Channel")
plt.xlabel("Channel Group")
plt.ylabel("Hours")
plt.show()
```



📉 Engagement Rate vs Sessions Over Time

- **Inverse relationship observed**: Higher session volume sometimes correlates with **lower engagement rate**, possibly due to traffic quality fluctuations (e.g., campaign spikes).
- Engagement rate generally stable around **40–50%**.

Engagement Rate vs Session Over Time

```
In [13]: df_plot = df.groupby("DateHour")[["Engagement rate", "Sessions"]].mean().reset_index()

plt.figure(figsize=(10,5))
plt.plot(df_plot["DateHour"], df_plot["Engagement rate"], label="Engagement Rate", color="green")
plt.plot(df_plot["DateHour"], df_plot["Sessions"], label="Sessions", color="blue")
plt.title("Engagement Rate vs Session Over Time")
plt.xlabel("DateHour")
plt.legend()
plt.grid("True")
plt.show()
```