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

```
df = pd.read_csv("/content/data-export (1).csv")
```

```
df.head()
```

	# -----------	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	
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	En
1	Direct	2024041623	237	300	144	47.526666666666700	0.6075949367088610	4.673333333333330	
2	Organic Social	2024041719	208	267	132	32.09737827715360	0.6346153846153850	4.295880149812730	0.4943
3	Direct	2024041723	188	233	115	39.93991416309010	0.6117021276595740	4.587982832618030	0.49356
4	Organic Social	2024041718	187	256	125	32.16015625	0.6684491978609630	4.078125	

Next steps: Generate code with df New interactive sheet

```
df.columns = df.iloc[0]
df =df.drop(index = 0).reset_index(drop = True)
df.columns = ["channel group", "Datehour", "Users", "sessions", "Engaged Sessions", "Average engagement time per session",
```

```
df.head()
```

	channel group	Datehour	Users	sessions	Engaged Sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement
0	Direct	2024041623	237	300	144	47.526666666666700	0.6075949367088610	4.673333333333330	
1	Organic Social	2024041719	208	267	132	32.09737827715360	0.6346153846153850	4.295880149812730	0.4943820224719
2	Direct	2024041723	188	233	115	39.93991416309010	0.6117021276595740	4.587982832618030	0.49356223175965
3	Organic Social	2024041718	187	256	125	32.16015625	0.6684491978609630	4.078125	0.4882
4	Organic Social	2024041720	175	221	112	46.918552036199100	0.64	4.529411764705880	0.506787330316

Next steps: Generate code with df New interactive sheet

```
df["Datehour"] = pd.to_datetime(df["Datehour"], format="%Y%m%d%H", errors='coerce')
```

```
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   datetime64[ns]
 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: datetime64[ns](1), object(9)
memory usage: 248.7+ KB
```

```
numeric_cols = df.columns.drop(["channel group", "Datehour"])
df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric, errors='coerce')
df["Hour"] = df["Datehour"].dt.hour
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3182 entries, 0 to 3181
Data columns (total 11 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   channel group                         3182 non-null   object
 1   Datehour                              3182 non-null   datetime64[ns]
 2   Users                                 3182 non-null   int64
 3   sessions                              3182 non-null   int64
 4   Engaged Sessions                      3182 non-null   int64
 5   Average engagement time per session   3182 non-null   float64
 6   Engaged sessions per user             3182 non-null   float64
 7   Events per session                    3182 non-null   float64
 8   Engagement rate                       3182 non-null   float64
 9   Event count                           3182 non-null   int64
 10  Hour                                  3182 non-null   int32
dtypes: datetime64[ns](1), float64(4), int32(1), int64(4), object(1)
memory usage: 261.2+ KB
```

```
df.head()
```

	channel group	Datehour	Users	sessions	Engaged Sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count	Hour
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	2024-04-17	175	221	112	46.918552	0.640000	4.529412	0.506787	1001	20

Next steps: Generate code with df | New interactive sheet

```
df.describe()
```

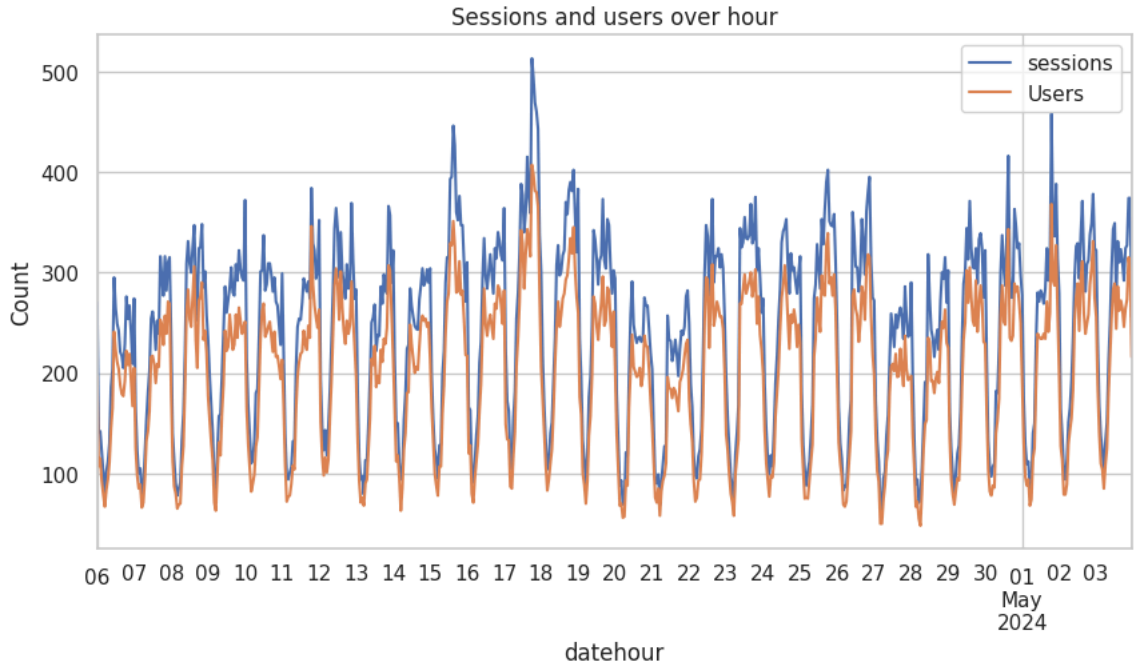
	Datehour	Users	sessions	Engaged Sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count
count	3182	3182.000000	3182.000000	3182.000000	3182.000000	3182.000000	3182.000000	3182.000000	3182.000000
mean	2024-04-20 01:17:07.278441216	41.935889	51.192646	28.325581	66.644581	0.606450	4.675969	0.503396	242.272470
min	2024-04-06 00:00:00	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000
25%	2024-04-13 02:15:00	20.000000	24.000000	13.000000	32.103034	0.561404	3.750000	0.442902	103.000000
50%	2024-04-20 02:00:00	42.000000	51.000000	27.000000	49.020202	0.666667	4.410256	0.545455	226.000000
75%	2024-04-26 22:00:00	60.000000	71.000000	41.000000	71.487069	0.750000	5.217690	0.633333	339.000000
max	2024-05-03 23:00:00	237.000000	300.000000	144.000000	4525.000000	2.000000	56.000000	1.000000	1402.000000
std	NaN	29.582258	36.919962	20.650569	127.200659	0.264023	2.795228	0.228206	184.440313

Solving business problem in some Qestion

```
# session and users over time
```

```
sns.set(style="whitegrid")
```

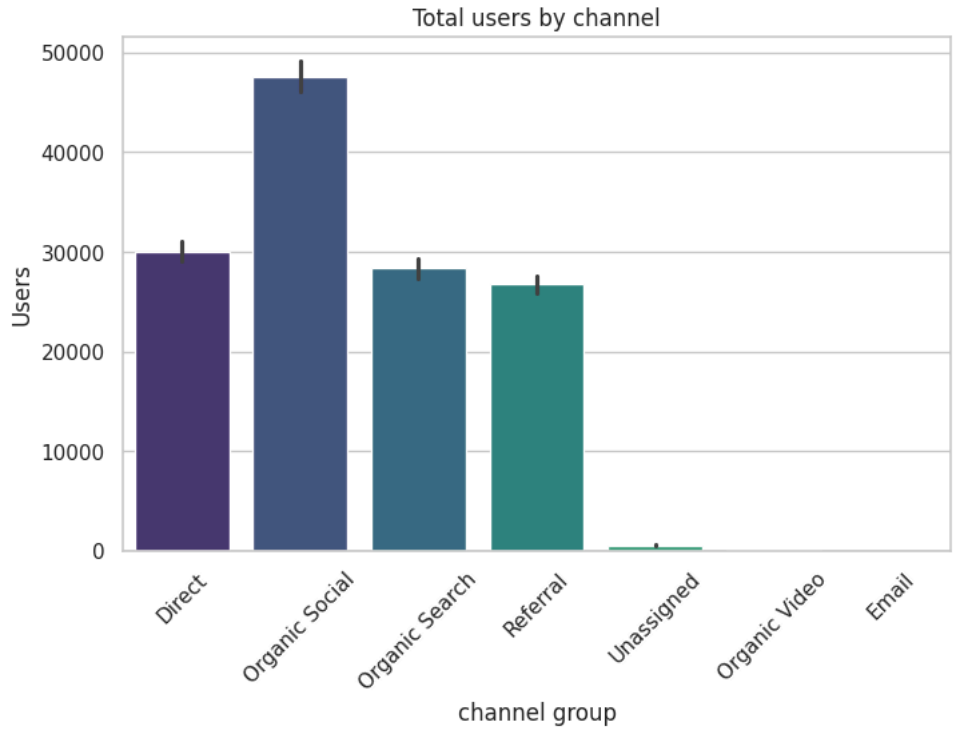
```
plt.figure(figsize=(10,5))
df.groupby("Datehour")[["sessions","Users"]].sum().plot(ax=plt.gca())
plt.title("Sessions and users over hour")
plt.xlabel("datehour")
plt.ylabel("Count")
plt.show()
```




```
#total users by channel
```

```
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()
```

```
/tmp/ipython-input-31229766.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

  sns.barplot(data=df, x="channel group", y="Users", estimator=np.sum, palette="viridis")
```




```
df.head()
```

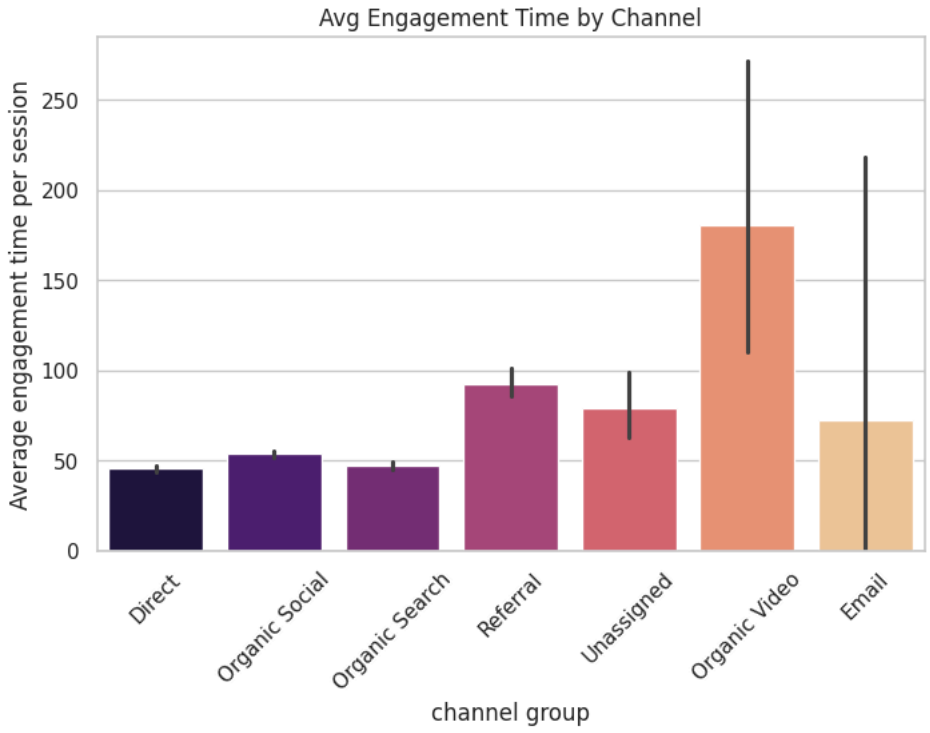
	channel group	Datehour	Users	sessions	Engaged Sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count	Hour
0	Direct	2024-04-16 23:00:00	237	300	144	47.526667	0.607595	4.673333	0.480000	1402	23
1	Organic Social	2024-04-17 19:00:00	208	267	132	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	2024-04-17	175	221	112	46.918552	0.640000	4.529412	0.506787	1001	20

Next steps: Generate code with df | New interactive sheet

```
#average engagement time by which channel
```

```
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("Avg Engagement Time by Channel")
plt.xticks(rotation=45)
plt.show()
```

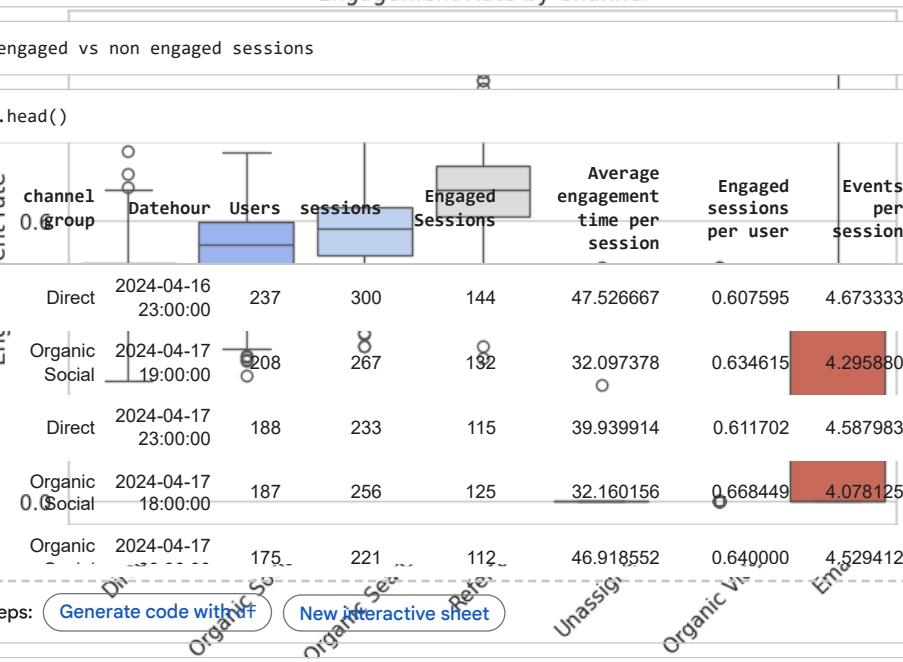
```
/tmp/ipython-input-4281777848.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

  sns.barplot(
```




```
# Engagement Rate distribution by channel
```

```
plt.figure(figsize=(8,5))

sns.boxplot(
    data=df,
    x="channel group",
    y="Engagement rate",
    palette="coolwarm"
)

plt.title("Engagement Rate by Channel")
plt.xticks(rotation=45)
plt.show()
```

```
/tmp/ipython-input-541942639.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

  sns.boxplot(
```



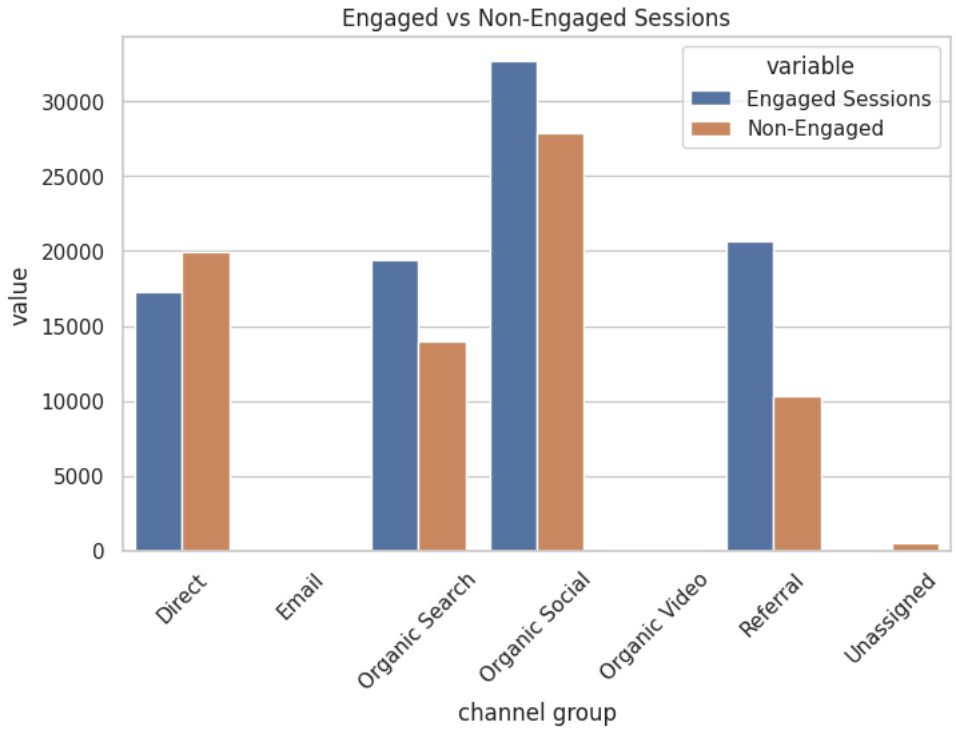

```
# engaged vs non engaged sessions
```

```
df.head()
```

	channel group	Datehour	Users	sessions	Engaged Sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count	Hour
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	2024-04-17	175	221	112	46.918552	0.640000	4.529412	0.506787	1001	20

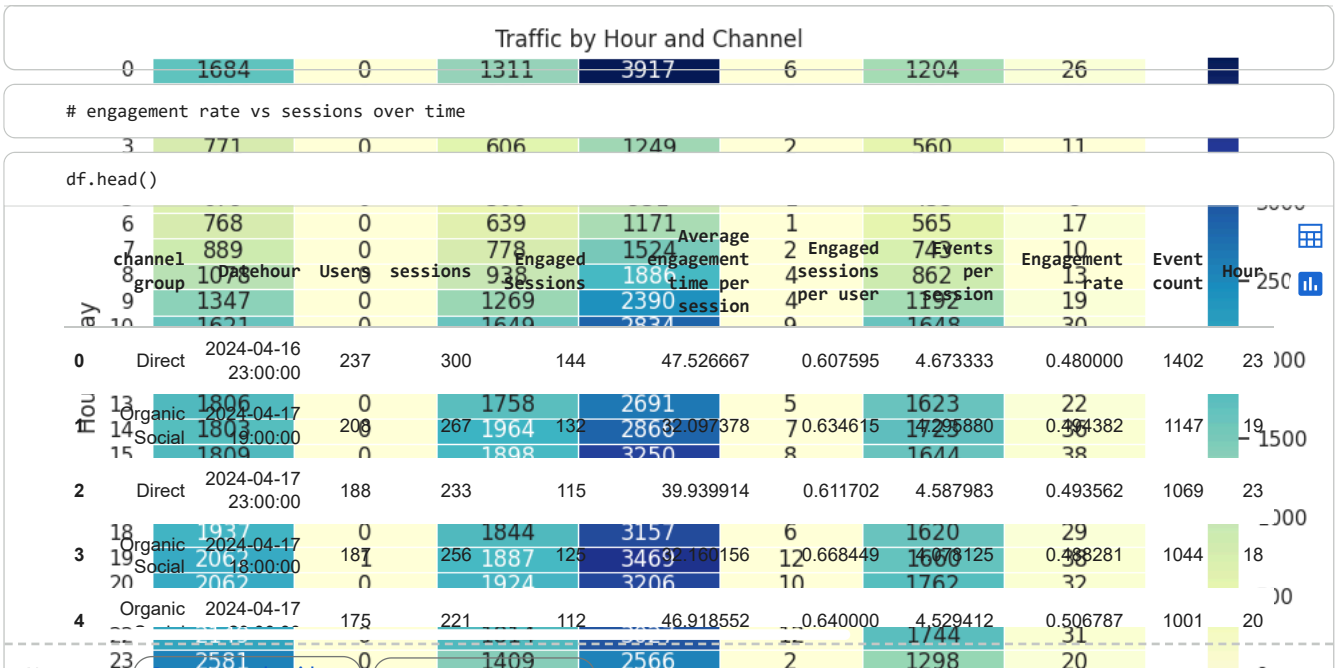
Next steps: Generate code with df New interactive sheet

```
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()
```




```
# traffic by hour and channel
```

```
heatmap_data = df.groupby(["Hour", "channel group"])["sessions"].sum().unstack().fillna(0)

plt.figure(figsize=(12, 6))
sns.heatmap(
    heatmap_data,
    cmap="YlGnBu",
    linewidths=.5,
    annot=True,
    fmt='.0f'
)

plt.title("Traffic by Hour and Channel")
plt.xlabel("channel group")
plt.ylabel("Hour of Day")
plt.show()
```




```
# engagement rate vs sessions over time
```

```
df.head()
```

	channel group	Datehour	Users	sessions	Engaged sessions	Average engagement time per session	Engaged sessions per user	Events per session	Engagement rate	Event count	Hour
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	42.097378	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	49.160156	0.668449	4.078125	0.488281	1044	18
4	Organic	2024-04-17	175	221	112	46.918552	0.640000	4.529412	0.506787	1001	20