In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_excel("Onyx Data - DataDNA Dataset Challenge - Social Media Content Performance Dataset - June 2025.xlsx")
```

In [3]:
```
df.head()
```

Out[3]:

	Post_ID	Platform	Content_Type	Content_Category	Post_Type	Region	Longitude	Latitude	Engagement	Views	...	Impressions	Video_Views	Live_Stream_Views	Clicks	Click_Throu
0	Post_1	TikTok	Organic	Product Promotion	Video	UK	-3.4360	55.3781	54510	599342	...	677975	465645	0	16813.0	
1	Post_2	Instagram	Organic	Product Promotion	Video	India	78.9629	20.5937	259440	1896301	...	2354474	1896301	0	NaN	
2	Post_3	X.com	Organic	Entertainment	Text	Brazil	-51.9253	-14.2350	14182	253052	...	304915	0	0	NaN	
3	Post_4	Instagram	Organic	Entertainment	Carousel	Australia	133.7751	-25.2744	49137	735640	...	829727	0	0	NaN	
4	Post_5	TikTok	Organic	Educational	Video	Brazil	-51.9253	-14.2350	20832	77358	...	93049	72707	0	1944.0	

5 rows × 24 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5600 entries, 0 to 5599
Data columns (total 24 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Post_ID             5600 non-null   object
 1   Platform            5600 non-null   object
 2   Content_Type        5600 non-null   object
 3   Content_Category    5600 non-null   object
 4   Post_Type           5600 non-null   object
 5   Region              5600 non-null   object
 6   Longitude           5600 non-null   float64
 7   Latitude            5600 non-null   float64
 8   Engagement          5600 non-null   int64
 9   Views               5600 non-null   int64
 10  Likes               5600 non-null   int64
 11  Shares              5600 non-null   int64
 12  Comments            5600 non-null   int64
 13  Engagement_Rate     5600 non-null   float64
 14  Impressions         5600 non-null   int64
 15  Video_Views         5600 non-null   int64
 16  Live_Stream_Views   5600 non-null   int64
 17  Clicks              1860 non-null   float64
 18  Click_Through_Rate  1860 non-null   float64
 19  Main_Hashtag        5600 non-null   object
 20  Post_Published_At   5600 non-null   datetime64[ns]
 21  Post_Date           5600 non-null   datetime64[ns]
 22  Post_Hour           5600 non-null   int64
 23  Engagement_Level    5600 non-null   object
dtypes: datetime64[ns](2), float64(5), int64(9), object(8)
memory usage: 1.0+ MB
```

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

Out[5]:

	Longitude	Latitude	Engagement	Views	Likes	Shares	Comments	Engagement_Rate	Impressions	Video_Views	Live_Stream_Views	Clicks
count	5600.000000	5600.000000	5.600000e+03	5.600000e+03	5600.000000	5600.000000	5600.000000	5600.000000	5.600000e+03	5.600000e+03	5.600000e+03	1860.000000
mean	9.658511	27.272488	1.154449e+05	8.582634e+05	150720.243571	47684.907679	34515.474286	0.152837	1.029742e+06	6.409677e+05	9.809161e+04	19036.116129
std	90.524261	29.330102	1.291954e+05	9.118957e+05	168758.692721	53459.056398	39801.207862	0.054360	1.096433e+06	9.587711e+05	3.198539e+05	21840.899426
min	-106.346800	-25.274400	7.300000e+01	1.000000e+03	101.000000	32.000000	20.000000	0.050049	1.125000e+03	0.000000e+00	0.000000e+00	12.000000
25%	-95.712900	20.593700	3.700425e+04	2.343490e+05	38544.000000	12104.750000	8589.500000	0.104681	2.805228e+05	0.000000e+00	0.000000e+00	4788.000000
50%	-3.436000	37.090200	5.899150e+04	3.755250e+05	67681.000000	21103.000000	15431.500000	0.154396	4.534145e+05	1.003765e+05	0.000000e+00	8227.000000
75%	78.962900	55.378100	1.534625e+05	1.441124e+06	223101.750000	71295.750000	49196.500000	0.197245	1.718588e+06	1.121694e+06	0.000000e+00	27792.000000
max	138.252900	56.130400	1.032271e+06	4.430104e+06	981669.000000	304982.000000	232658.000000	0.269302	5.282874e+06	5.474756e+06	1.998724e+06	134139.000000

In [6]:
```
df.columns
```

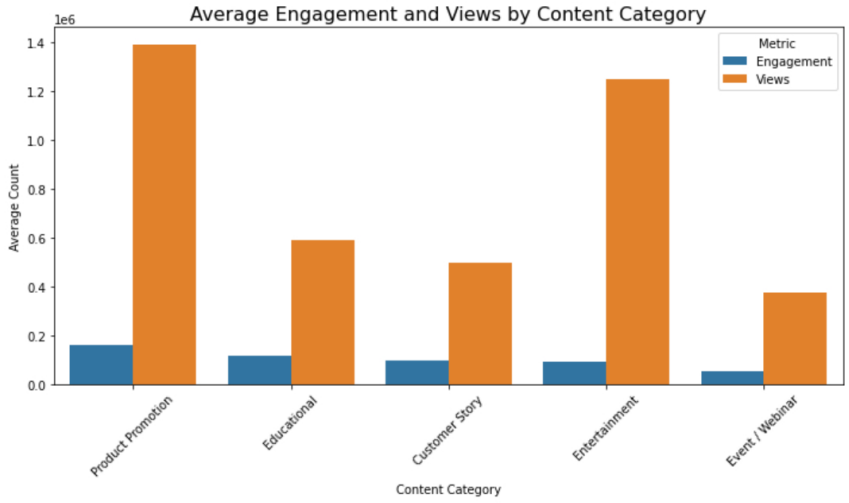
Out[6]:
```
Index(['Post_ID', 'Platform', 'Content_Type', 'Content_Category', 'Post_Type',
       'Region', 'Longitude', 'Latitude', 'Engagement', 'Views', 'Likes',
       'Shares', 'Comments', 'Engagement_Rate', 'Impressions', 'Video_Views',
       'Live_Stream_Views', 'Clicks', 'Click_Through_Rate', 'Main_Hashtag',
       'Post_Published_At', 'Post_Date', 'Post_Hour', 'Engagement_Level'],
      dtype='object')
```

In [9]:
```
null_summary = df.isnull().sum().sort_values(ascending=False)
print(null_summary)
```

```
Click_Through_Rate    3740
Clicks                3740
Post_ID                  0
Platform                 0
Post_Hour                0
Post_Date                0
Post_Published_At        0
```

```
Main_Hashtag          0
Live_Stream_Views     0
Video_Views           0
Impressions           0
Engagement_Rate       0
Comments              0
Shares                0
Likes                 0
Views                 0
Engagement            0
Latitude              0
Longitude             0
Region                0
Post_Type             0
Content_Category      0
Content_Type          0
Engagement_Level      0
dtype: int64
```

In [10]:
```
# Group and calculate average
content_category_stats = df.groupby("Content_Category")[["Engagement", "Views"]].mean().sort_values("Engagement", ascending=False).reset_index()

# Plot
plt.figure(figsize=(10,6))
sns.barplot(data=content_category_stats.melt(id_vars="Content_Category"), x="Content_Category", y="value", hue="variable")
plt.title("Average Engagement and Views by Content Category", fontsize=16)
plt.ylabel("Average Count")
plt.xlabel("Content Category")
plt.xticks(rotation=45)
plt.legend(title='Metric')
plt.tight_layout()
plt.show()
```



In [11]:
```
# By Platform
df.groupby("Platform")[["Engagement", "Views"]].mean().sort_values("Engagement", ascending=False)

# By Post Type
df.groupby("Post_Type")[["Engagement", "Views"]].mean().sort_values("Engagement", ascending=False)
```

Out[11]:

Post_Type	Engagement	Views
Video	160162.434193	1.235123e+06
Article	122780.763285	9.154202e+05
PDF	117545.562500	7.544489e+05
Carousel	111102.156863	8.297768e+05
Text	66802.054585	4.666845e+05
Image	58837.041165	3.591952e+05
Live Stream	56464.280303	3.785193e+05

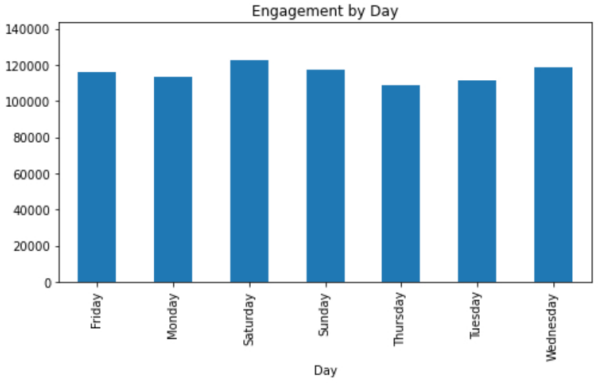
In [12]:
```
df.groupby(["Region", "Content_Category"])[["Engagement", "Views"]].mean().unstack().fillna(0)
```

Out[12]:

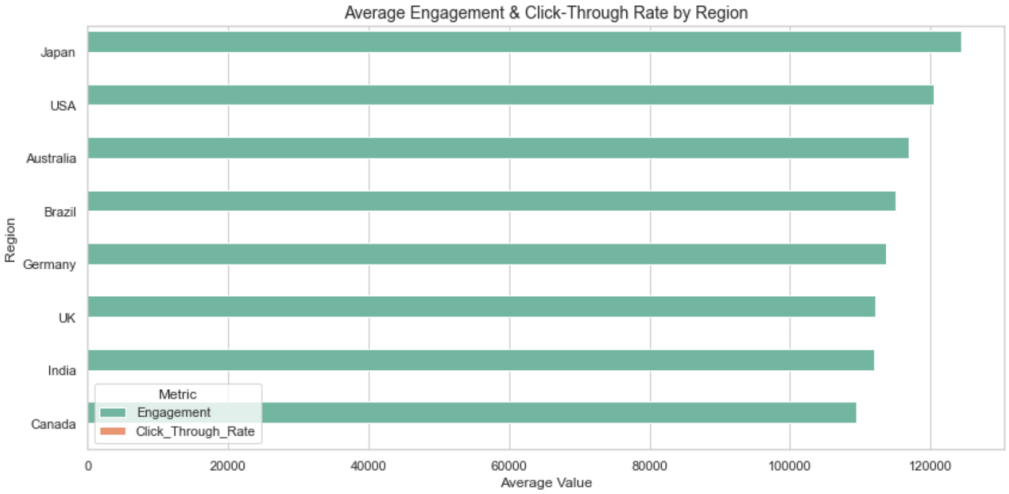
	Engagement					Views				
Content_Category	**Customer Story**	**Educational**	**Entertainment**	**Event / Webinar**	**Product Promotion**	**Customer Story**	**Educational**	**Entertainment**	**Event / Webinar**	**Product Promotion**
Region										
Australia	90371.632653	111941.530364	98431.916667	35157.633333	181439.542169	448206.163265	571667.753036	1.360602e+06	255381.316667	1.544235e+06
Brazil	85874.901099	129543.387597	92020.122642	45573.756757	149627.000000	437056.340659	639301.240310	1.221180e+06	320584.162162	1.281750e+06
Canada	94651.892857	109115.245734	96734.338983	53611.721519	154183.925466	483248.214286	547055.839590	1.278917e+06	370026.886076	1.353646e+06
Germany	116794.260870	125297.059113	83546.699115	59148.263889	141733.500000	587280.304348	636849.517241	1.136854e+06	416139.819444	1.224924e+06
India	112252.827273	117217.165289	86588.081081	47346.350000	144381.898810	561915.754545	580692.024793	1.155093e+06	339088.433333	1.262078e+06
Japan	103813.917808	133763.901639	103497.696429	49629.384615	170453.622642	536310.698630	679387.696721	1.385274e+06	358652.730769	1.486083e+06
UK	88091.178218	109204.766423	88714.320755	55721.314286	165263.044199	441810.009901	544310.868613	1.180996e+06	409351.014286	1.444019e+06
USA	102647.165138	108074.260377	96910.970874	71401.809524	181946.769634	515421.990826	550424.788679	1.283272e+06	507038.583333	1.538922e+06

In [13]:
```
# Convert post date if not already
df["Post_Date"] = pd.to_datetime(df["Post_Date"])
df["Day"] = df["Post_Date"].dt.day_name()

# By Hour
df.groupby("Post_Hour")["Engagement"].mean().plot(title="Engagement by Hour", figsize=(8,4))

# By Day
df.groupby("Day")["Engagement"].mean().plot(kind="bar", title="Engagement by Day", figsize=(8,4))
```

Out[13]:
```
<AxesSubplot:title={'center':'Engagement by Day'}, xlabel='Day'>
```



In [26]:
```
region_perf = df.groupby("Region")[["Engagement", "Click_Through_Rate"]].mean().sort_values("Engagement", ascending=False).reset_index()

# Set plot style
sns.set(style="whitegrid")
plt.figure(figsize=(12, 6))

# Melt the DataFrame for seaborn
region_melt = region_perf.melt(id_vars="Region", value_vars=["Engagement", "Click_Through_Rate"])

# Plot
sns.barplot(data=region_melt, x="value", y="Region", hue="variable", palette="Set2")
plt.title("Average Engagement & Click-Through Rate by Region", fontsize=14)
plt.xlabel("Average Value")
plt.ylabel("Region")
plt.legend(title="Metric")
plt.tight_layout()
plt.show()
```



In [15]:
```
df.groupby("Main_Hashtag")[["Impressions", "Clicks"]].mean().sort_values("Clicks", ascending=False).head(10)
```

Out[15]:

Main_Hashtag	Impressions	Clicks
#MemeMonday	1.675535e+06	43313.200000
#TrendingNow	1.952593e+06	41205.973214
#ProductDemo	2.118562e+06	37976.170833
#SaaSLaunch	1.562574e+06	34385.763889
#CustomerStory	1.206673e+06	34126.756250
#NewRelease	1.496673e+06	33577.480000
#WebinarReplay	7.085907e+05	31307.888889
#DidYouKnow	9.164976e+05	20836.184783
#Testimonial	7.296697e+05	15860.530612
#FeatureHighlight	5.601416e+05	13079.056338

In [16]:
```
df.groupby("Region")[["Video_Views", "Live_Stream_Views"]].mean().sort_values("Video_Views", ascending=False).head(10)
```

Out[16]:

Region	Video_Views	Live_Stream_Views
Japan	686559.899399	85559.307808
USA	683791.348404	99701.114362
Germany	663071.814992	127439.956938
Australia	662248.905744	94414.178203
Brazil	633370.208914	92219.667131
Canada	609116.221769	94051.881633
India	597879.325615	97045.102750
UK	596928.002732	96916.901639

In [17]:
```
df.groupby(["Content_Category", "Post_Hour"])["Engagement"].mean().unstack().fillna(0)
```

Out[17]:

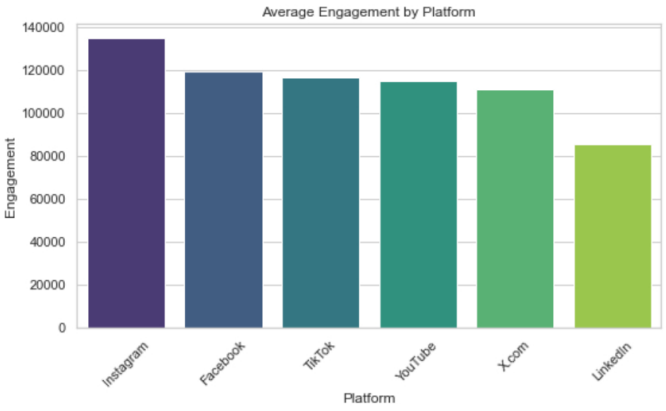
Content_Category	Post_Hour 8	9	10	11	12	13	14	15	16	17	18	
Customer Story	170956.421053	90642.700000	85770.293478	87335.488636	87980.707865	67119.318182	140518.897436	86471.702128	93053.701923	122119.519608	122635.733333	930
Educational	144383.264706	117961.254237	113402.821277	110073.402730	121856.954064	112882.985714	81824.716981	112401.067857	123609.778947	122101.072727	122558.279412	1233
Entertainment	69059.379310	108581.250000	94067.398438	91570.660714	92670.905405	108683.000000	80983.333333	87263.463636	97168.030928	102456.279279	81100.636364	831
Event / Webinar	27607.473684	42104.640000	49548.116279	53344.985915	51757.576923	40797.937500	56795.315789	59998.763158	59261.075949	56681.223684	51320.222222	513
Product Promotion	184817.016949	154708.655172	157545.163158	182119.392473	161332.237179	177832.441176	160919.406250	160231.065657	152736.522388	147366.513812	155501.153846	1606

In [18]:
```
df.groupby("Content_Type")[["Engagement", "Views", "Impressions"]].mean()
```

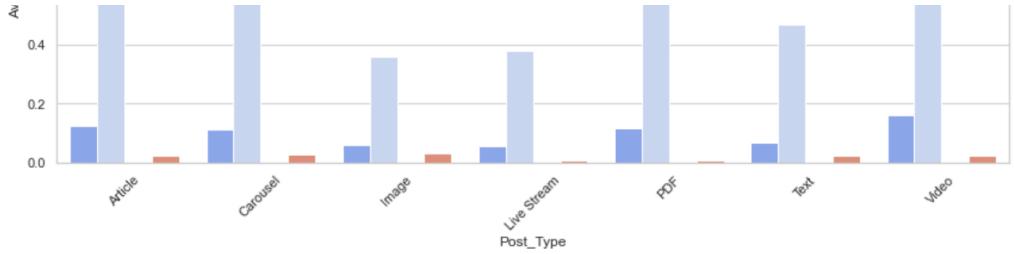
Out[18]:

Content_Type	Engagement	Views	Impressions
Organic	112689.152174	834567.159062	1.000777e+06
Sponsored	128865.451782	973664.793501	1.170801e+06

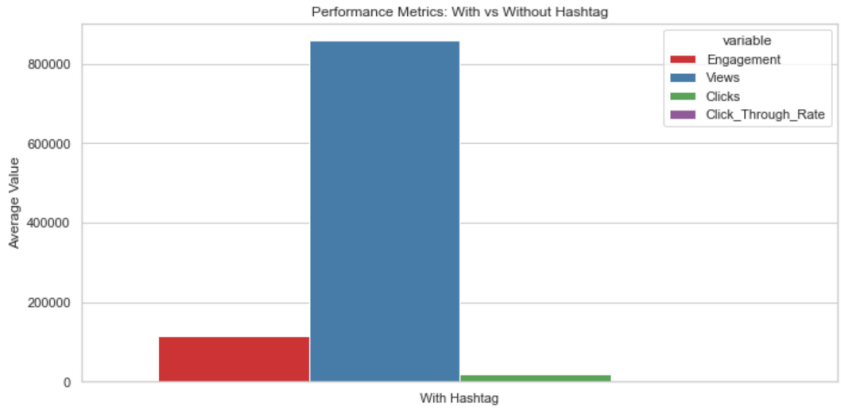
In [27]:
```
platform_engagement = df.groupby("Platform")["Engagement"].mean().reset_index().sort_values("Engagement", ascending=False)

plt.figure(figsize=(8,5))
sns.barplot(data=platform_engagement, x="Platform", y="Engagement", palette="viridis")
plt.title("Average Engagement by Platform")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



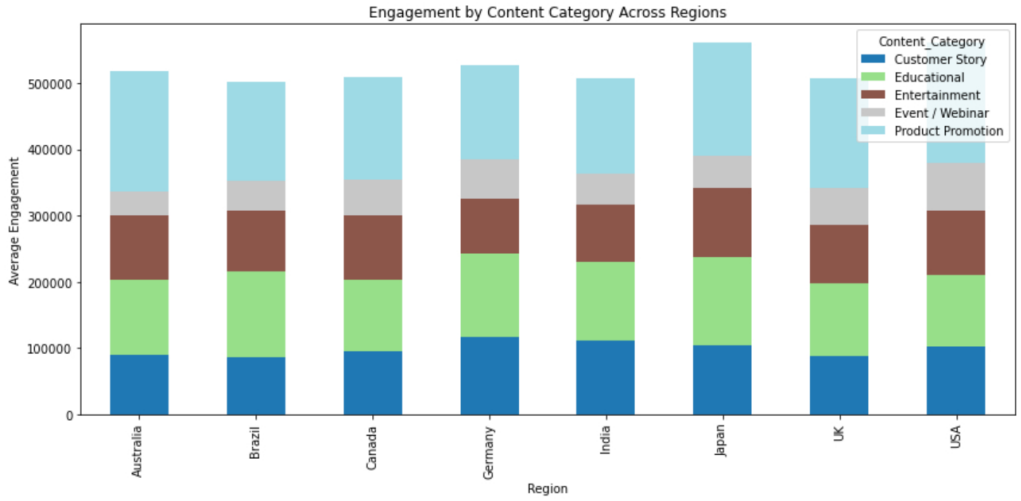
In [28]:
```
post_type_perf = df.groupby("Post_Type")[["Engagement", "Views", "Click_Through_Rate", "Clicks"]].mean().reset_index()
post_type_melt = post_type_perf.melt(id_vars="Post_Type")

plt.figure(figsize=(12,6))
sns.barplot(data=post_type_melt, x="Post_Type", y="value", hue="variable", palette="coolwarm")
plt.title("Performance Metrics by Post Type")
plt.ylabel("Average Value")
plt.xticks(rotation=45)
plt.legend(title="Metric")
plt.tight_layout()
plt.show()
```



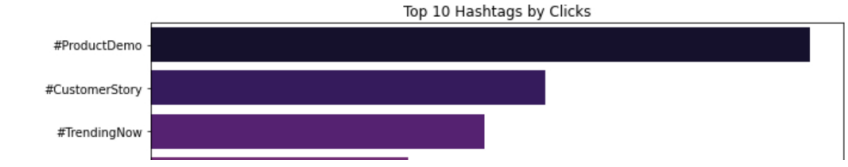
In [29]:
```
# Create flag for hashtag presence
df['Has_Hashtag'] = df['Main_Hashtag'].notnull()

hashtag_perf = df.groupby("Has_Hashtag")[["Engagement", "Views", "Clicks", "Click_Through_Rate"]].mean().reset_index()
hashtag_perf["Has_Hashtag"] = hashtag_perf["Has_Hashtag"].map({True: "With Hashtag", False: "Without Hashtag"})
hashtag_melt = hashtag_perf.melt(id_vars="Has_Hashtag")

plt.figure(figsize=(10,5))
sns.barplot(data=hashtag_melt, x="Has_Hashtag", y="value", hue="variable", palette="Set1")
plt.title("Performance Metrics: With vs Without Hashtag")
plt.ylabel("Average Value")
plt.xlabel("")
plt.tight_layout()
plt.show()
```

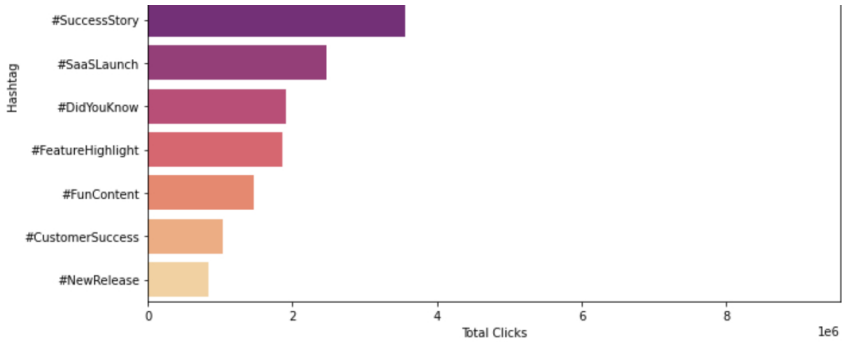


In [22]:
```
category_region = df.groupby(["Region", "Content_Category"])["Engagement"].mean().unstack().fillna(0)
category_region.plot(kind="bar", stacked=True, figsize=(12,6), colormap="tab20")
plt.title("Engagement by Content Category Across Regions")
plt.ylabel("Average Engagement")
plt.tight_layout()
plt.show()
```

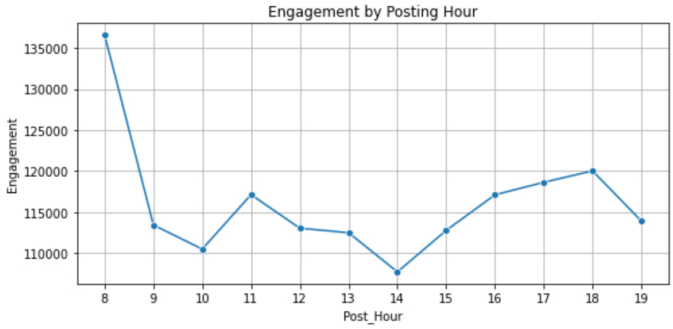


In [23]:
```
top_hashtags = df.groupby("Main_Hashtag")["Clicks"].sum().sort_values(ascending=False).head(10).reset_index()

plt.figure(figsize=(10,6))
sns.barplot(data=top_hashtags, x="Clicks", y="Main_Hashtag", palette="magma")
plt.title("Top 10 Hashtags by Clicks")
plt.xlabel("Total Clicks")
plt.ylabel("Hashtag")
plt.tight_layout()
plt.show()
```

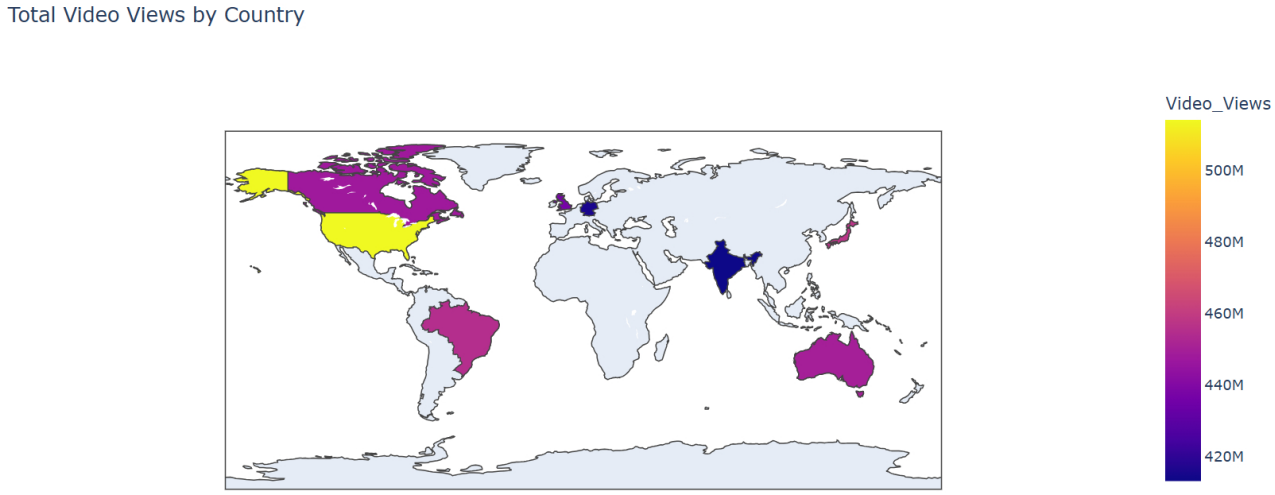




In [21]:
```
hourly_engagement = df.groupby("Post_Hour")["Engagement"].mean().reset_index()

plt.figure(figsize=(8,4))
sns.lineplot(data=hourly_engagement, x="Post_Hour", y="Engagement", marker="o")
plt.title("Engagement by Posting Hour")
plt.xticks(range(8, 20))
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [24]:
```
import plotly.express as px

country_video_views = df.groupby("Region")["Video_Views"].sum().reset_index()

fig = px.choropleth(country_video_views,
                    locations="Region",
                    locationmode="country names",
                    color="Video_Views",
                    color_continuous_scale="Plasma",
                    title="Total Video Views by Country")

fig.show()
```



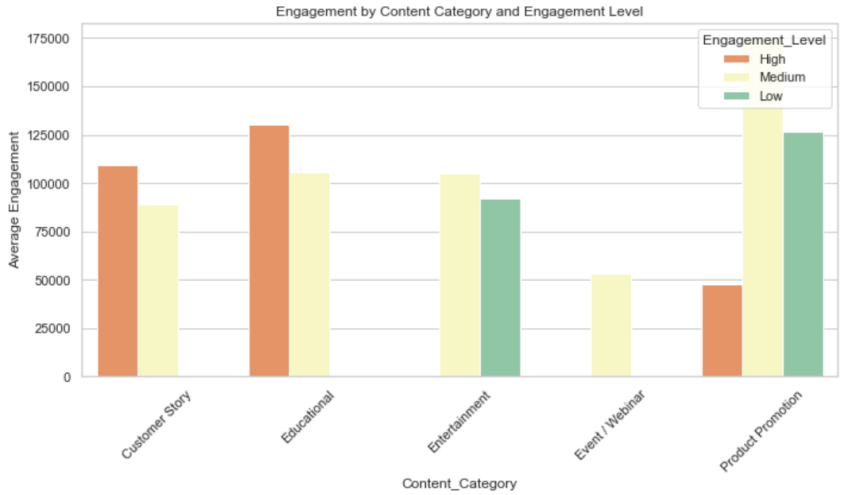
In [25]:
```
df.groupby("Region")[["Engagement", "Click_Through_Rate"]].mean().sort_values("Engagement", ascending=False)
```

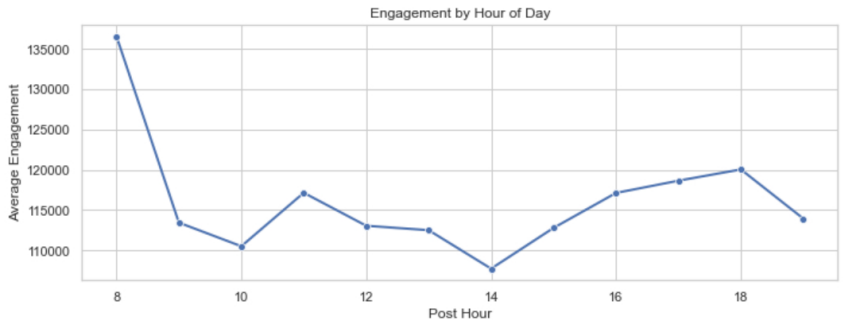
Out[25]:

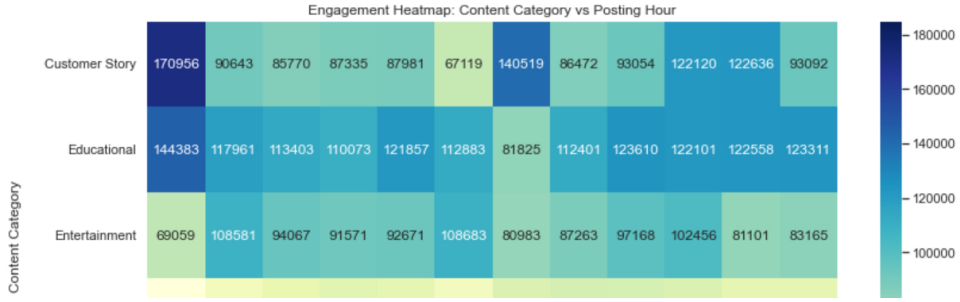
Region	Engagement	Click_Through_Rate
Japan	124296.948949	0.018149
USA	120425.046543	0.018026
Australia	116885.194404	0.018677

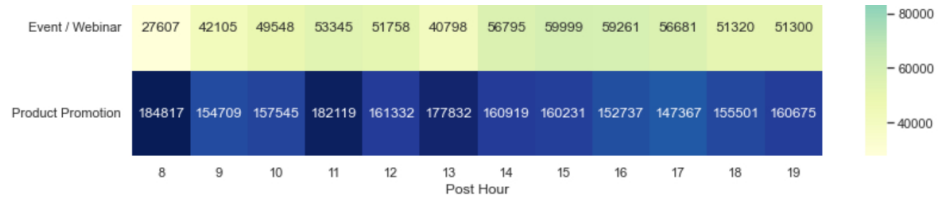
Brazil	115101.537604	0.018035
Germany	113697.374801	0.019051
UK	112071.224044	0.018205
India	112044.257598	0.018701
Canada	109381.110204	0.018161

In [30]:
```
# --- PART 1: Engagement by Content Category & Engagement Level ---
cat_level = df.groupby(["Content_Category", "Engagement_Level"])["Engagement"].mean().reset_index()

plt.figure(figsize=(10,6))
sns.barplot(data=cat_level, x="Content_Category", y="Engagement", hue="Engagement_Level", palette="Spectral")
plt.title("Engagement by Content Category and Engagement Level")
plt.ylabel("Average Engagement")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# --- PART 2: Engagement by Posting Hour ---
hourly = df.groupby("Post_Hour")["Engagement"].mean().reset_index()

plt.figure(figsize=(10,4))
sns.lineplot(data=hourly, x="Post_Hour", y="Engagement", marker='o', linewidth=2)
plt.title("Engagement by Hour of Day")
plt.xlabel("Post Hour")
plt.ylabel("Average Engagement")
plt.grid(True)
plt.tight_layout()
plt.show()

# --- PART 3: Heatmap of Content Category vs Hour ---
heat_data = df.pivot_table(values="Engagement", index="Content_Category", columns="Post_Hour", aggfunc="mean")

plt.figure(figsize=(12,6))
sns.heatmap(heat_data, annot=True, fmt=".0f", cmap="YlGnBu")
plt.title("Engagement Heatmap: Content Category vs Posting Hour")
plt.xlabel("Post Hour")
plt.ylabel("Content Category")
plt.tight_layout()
plt.show()
```

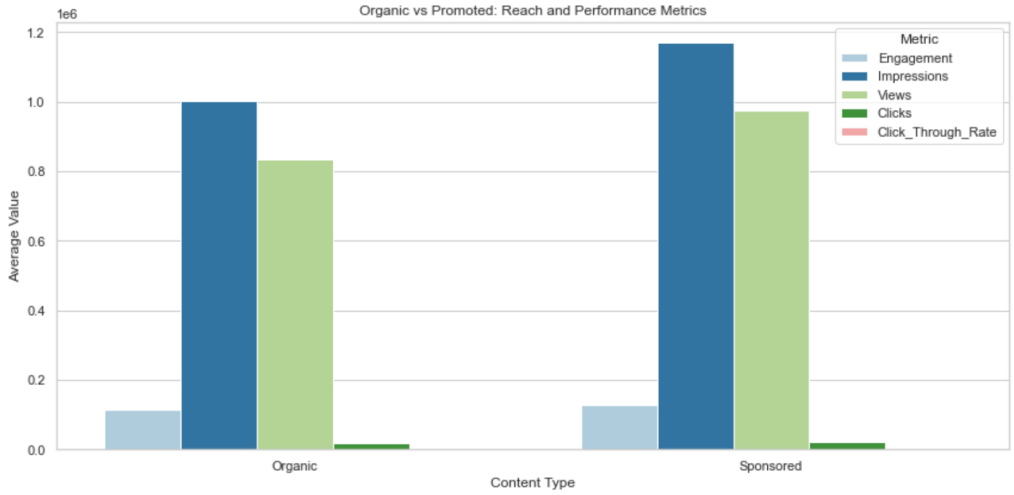








In [31]:
```
# Group by Content Type (Organic vs Promoted)
type_perf = df.groupby("Content_Type")[["Engagement", "Impressions", "Views", "Clicks", "Click_Through_Rate"]].mean().reset_index()

# Melt for Seaborn
type_melt = type_perf.melt(id_vars="Content_Type", value_vars=["Engagement", "Impressions", "Views", "Clicks", "Click_Through_Rate"])

# Plot
plt.figure(figsize=(12,6))
sns.barplot(data=type_melt, x="Content_Type", y="value", hue="variable", palette="Paired")
plt.title("Organic vs Promoted: Reach and Performance Metrics")
plt.ylabel("Average Value")
plt.xlabel("Content Type")
plt.legend(title="Metric")
plt.tight_layout()
plt.show()
```



In [ ]: