In [1]:
```
# Importing necessary Libraries

import pandas as pd
import numpy as np
import datetime
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.cluster import KMeans
import plotly.graph_objects as go
```

In [2]:
```
# Read CSV input file

df = pd.read_csv(".//Dataset//user_profiles_for_ads.csv")
```

In [3]:
```
# Check for null values

df.isnull().sum()
```

Out[3]:
```
User ID                            0
Age                                0
Gender                             0
Location                           0
Language                           0
Education Level                    0
Likes and Reactions                0
Followed Accounts                  0
Device Usage                       0
Time Spent Online (hrs/weekday)    0
Time Spent Online (hrs/weekend)    0
Click-Through Rates (CTR)          0
Conversion Rates                   0
Ad Interaction Time (sec)          0
Income Level                       0
Top Interests                      0
dtype: int64
```

In [4]:
```
df
```

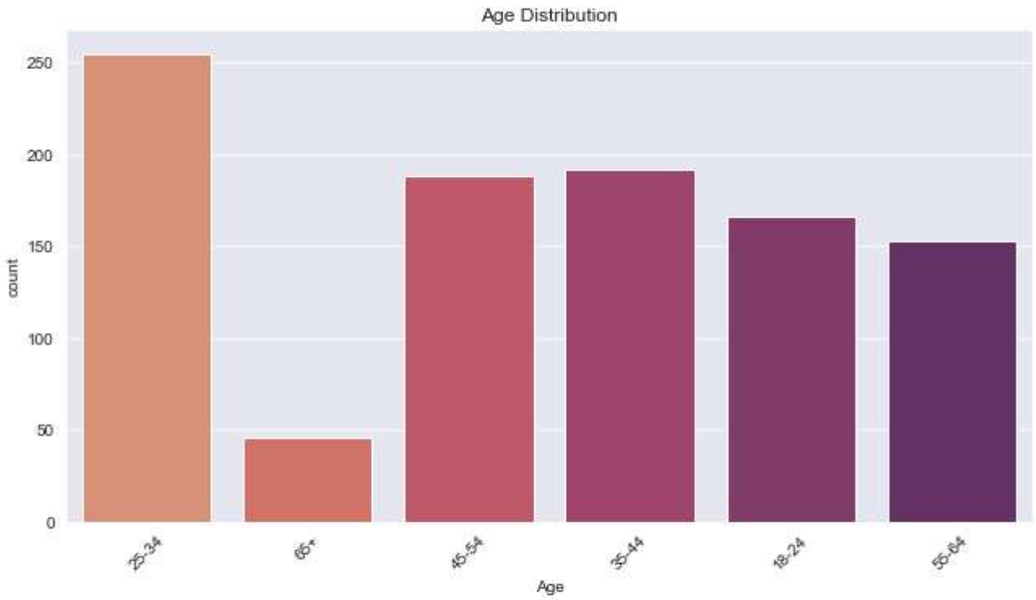
Out[4]:

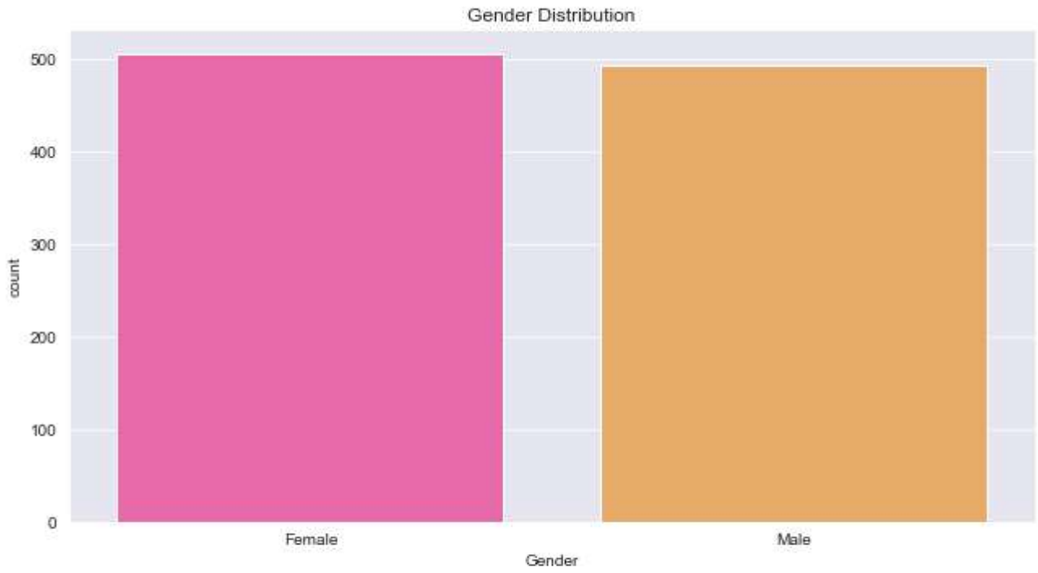
	User ID	Age	Gender	Location	Language	Education Level	Likes and Reactions	Followed Accounts	Device Usage	Time Spent Online (hrs/weekday)	Time Spent Online (hrs/weekend)	Click-Through Rates (CTR)	Conversion Rates	Ad Interaction Time (sec)	Income Level	Top Interests
0	1	25-34	Female	Suburban	Hindi	Technical	5640	190	Mobile Only	4.5	1.7	0.193	0.067	25	20k-40k	Digital Marketing
1	2	65+	Male	Urban	Hindi	PhD	9501	375	Tablet	0.5	7.7	0.114	0.044	68	0-20k	Data Science
2	3	45-54	Female	Suburban	Spanish	Technical	4775	187	Mobile Only	4.5	5.6	0.153	0.095	80	60k-80k	Fitness and Wellness
3	4	35-44	Female	Rural	Spanish	PhD	9182	152	Desktop Only	3.1	4.2	0.093	0.061	65	100k+	Gaming, DIY Crafts
4	5	25-34	Female	Urban	English	Technical	6848	371	Mobile Only	2.0	3.8	0.175	0.022	99	20k-40k	Fitness and Wellness, Investing and Finance, G...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	996	18-24	Female	Rural	Spanish	Bachelor	3144	74	Tablet	4.6	5.3	0.097	0.088	154	100k+	Data Science, Fitness and Wellness, Eco-Friend...
996	997	55-64	Female	Suburban	Hindi	PhD	9712	458	Mobile Only	4.2	5.6	0.098	0.032	78	100k+	Gardening
997	998	18-24	Male	Rural	Hindi	Technical	5736	218	Mobile + Desktop	2.1	2.4	0.154	0.070	91	100k+	Investing and Finance, Data Science, Photograp...
998	999	65+	Male	Urban	English	PhD	2992	260	Mobile + Desktop	4.1	2.7	0.031	0.025	147	60k-80k	Data Science, Eco-Friendly Living, Gaming, Tra...
999	1000	35-44	Female	Urban	Hindi	High School	5388	394	Desktop Only	2.1	5.6	0.145	0.076	98	40k-60k	Data Science, DIY Crafts, Gaming

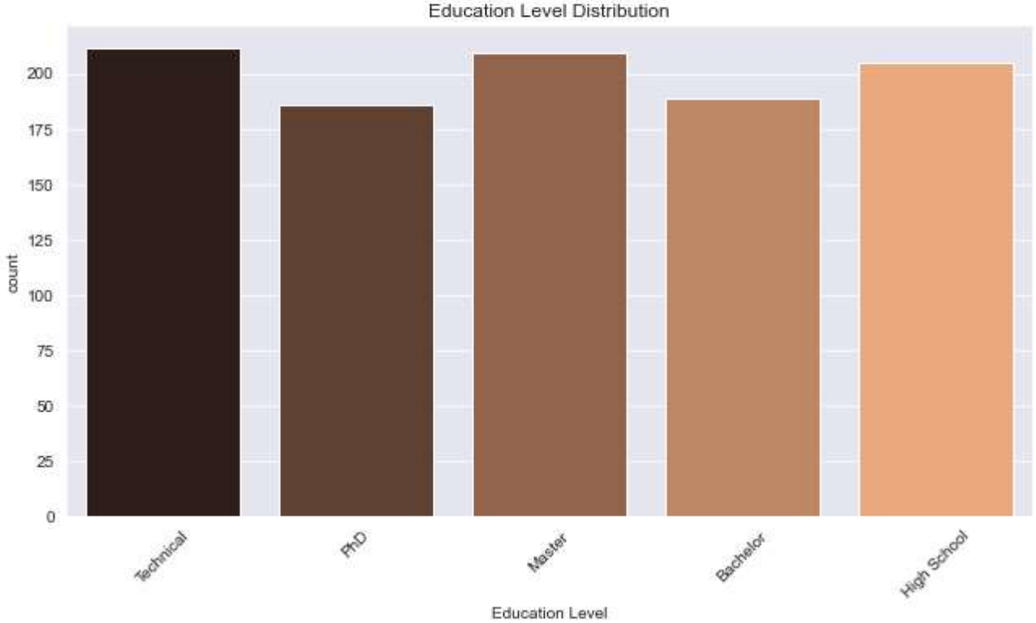
1000 rows × 16 columns

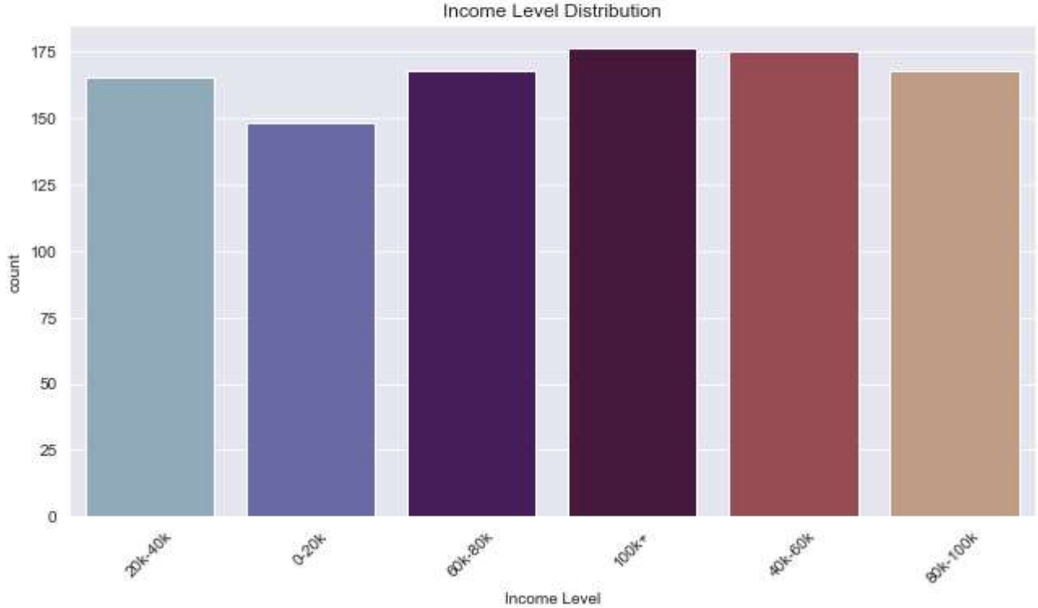
In [5]:
```
# Exploratory Data Analysis

# Visualize the data on the basis of Age, Gender , Educational level and Income level
# Set aesthetics style of the plots
sns.set_style("darkgrid")

# Distribution based on Gender, Age, Demographics


fig, axes = plt.subplots(2, 2, figsize=(18, 12))
fig.suptitle('Distribution of Key Demographic Variables')

# age distribution
sns.countplot(ax=axes[0, 0], x='Age', data=df, palette='flare')  # Example palette: Set2
axes[0, 0].set_title('Age Distribution')
axes[0, 0].tick_params(axis='x', rotation=45)

# gender distribution
sns.countplot(ax=axes[0, 1], x='Gender', data=df, palette='spring')  # Example palette: Set1
axes[0, 1].set_title('Gender Distribution')

# education level distribution
sns.countplot(ax=axes[1, 0], x='Education Level', data=df, palette='copper')  # Example palette: Pastel1
axes[1, 0].set_title('Education Level Distribution')
axes[1, 0].tick_params(axis='x', rotation=45)

# income level distribution
sns.countplot(ax=axes[1, 1], x='Income Level', data=df, palette='twilight')  # Example palette: Pastel2
axes[1, 1].set_title('Income Level Distribution')
axes[1, 1].tick_params(axis='x', rotation=45)


plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Distribution of Key Demographic Variables

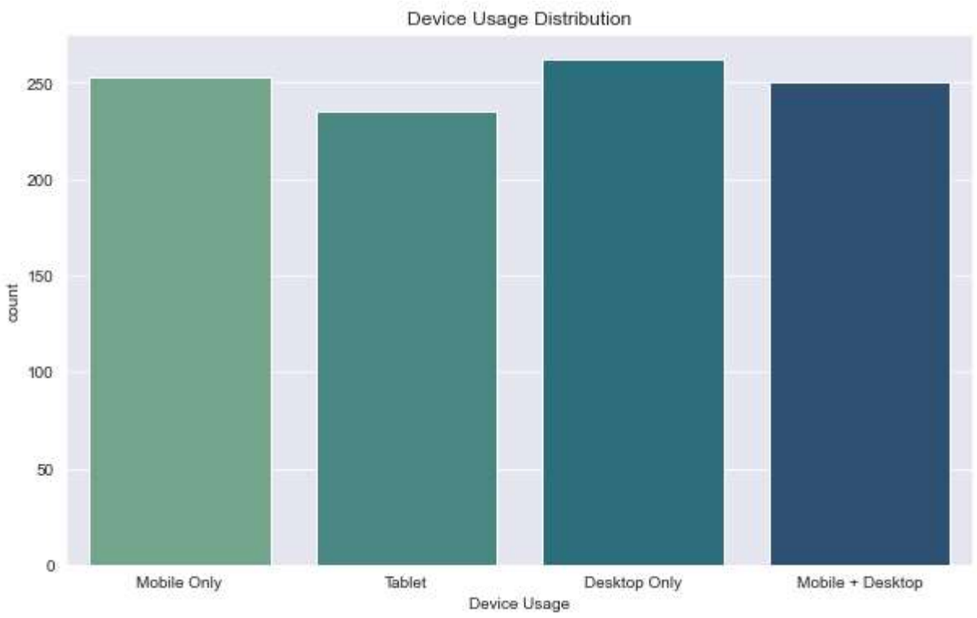






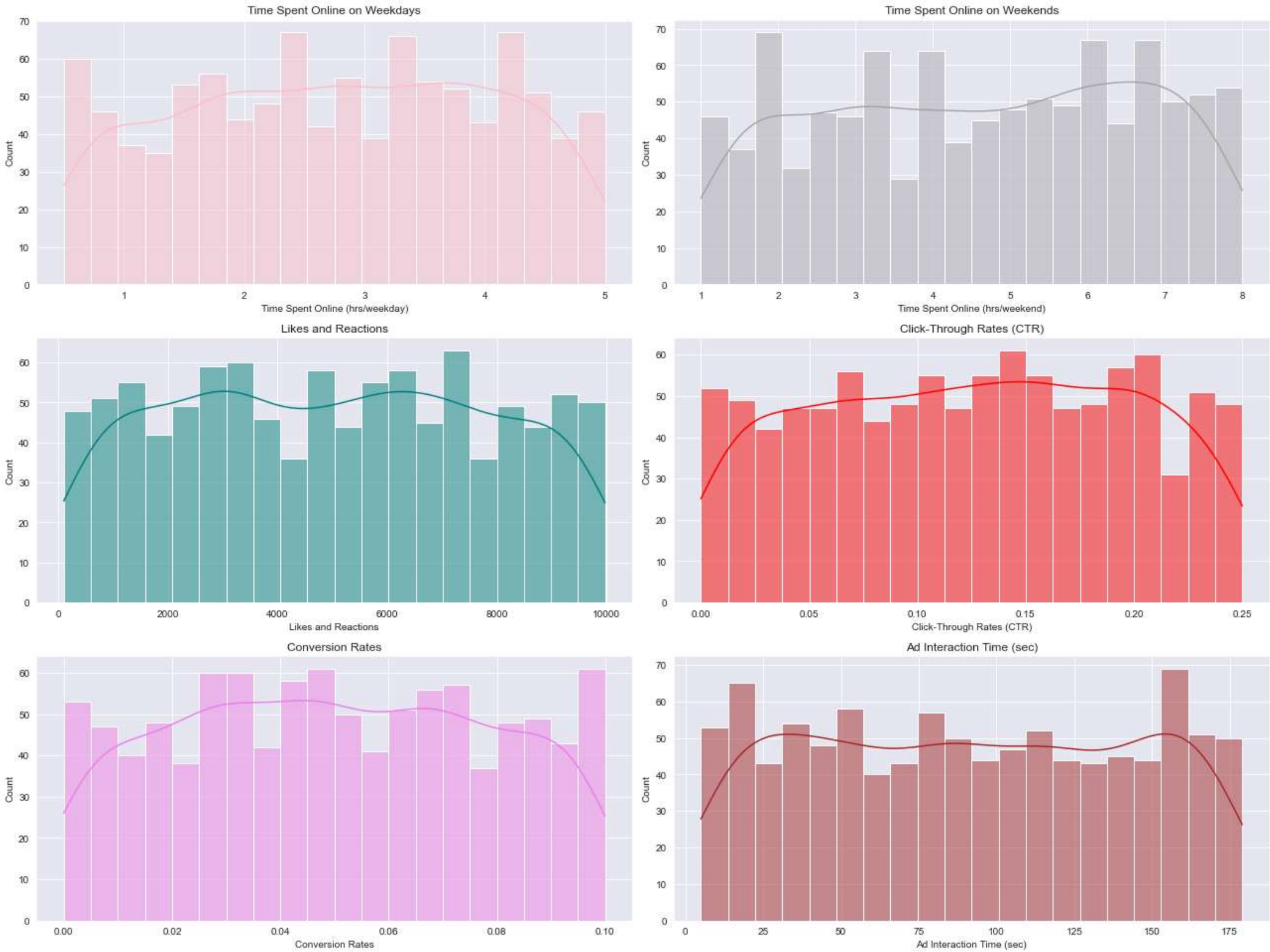


In [6]:
```
# Device Distribution
plt.figure(figsize=(10, 6))
sns.countplot(x='Device Usage', data=df, palette='crest')
plt.title('Device Usage Distribution')
plt.show()
```

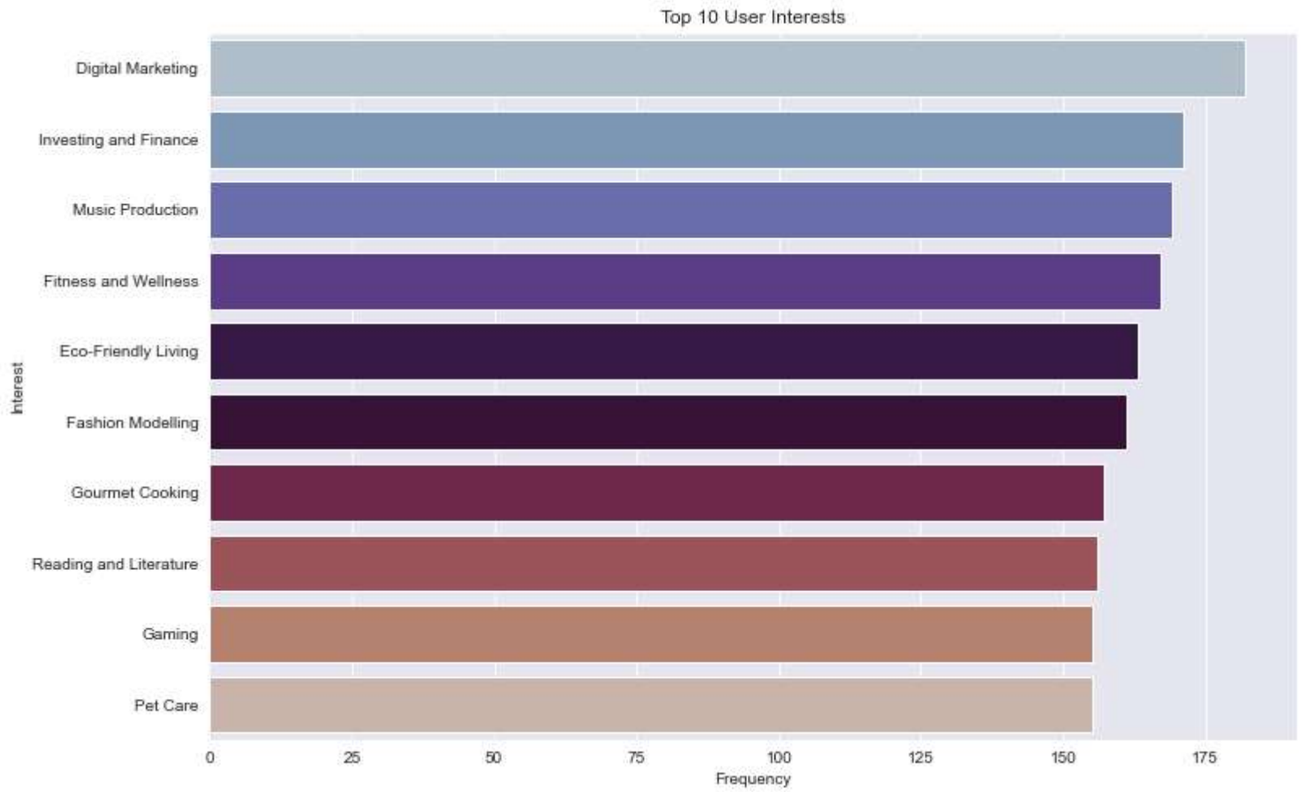


In [7]:
```
# User Online Behavior and Ad Interaction metrics

fig, axes = plt.subplots(3, 2, figsize=(18, 15))
fig.suptitle('User Online Behavior and Ad Interaction Metrics')

# time spent online on weekdays
sns.histplot(ax=axes[0, 0], x='Time Spent Online (hrs/weekday)', data=df, bins=20, kde=True, color='pink')
axes[0, 0].set_title('Time Spent Online on Weekdays')

# time spent online on weekends
sns.histplot(ax=axes[0, 1], x='Time Spent Online (hrs/weekend)', data=df, bins=20, kde=True, color='darkgray')
axes[0, 1].set_title('Time Spent Online on Weekends')

# likes and reactions
sns.histplot(ax=axes[1, 0], x='Likes and Reactions', data=df, bins=20, kde=True, color='teal')
axes[1, 0].set_title('Likes and Reactions')

# click-through rates
sns.histplot(ax=axes[1, 1], x='Click-Through Rates (CTR)', data=df, bins=20, kde=True, color='red')
axes[1, 1].set_title('Click-Through Rates (CTR)')

# conversion rates
sns.histplot(ax=axes[2, 0], x='Conversion Rates', data=df, bins=20, kde=True, color='violet')
axes[2, 0].set_title('Conversion Rates')

# ad interaction time
sns.histplot(ax=axes[2, 1], x='Ad Interaction Time (sec)', data=df, bins=20, kde=True, color='brown')
axes[2, 1].set_title('Ad Interaction Time (sec)')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

User Online Behavior and Ad Interaction Metrics



In [8]:
```
# splitting the 'Top Interests' column and creating a list of all interests
interests_list = df['Top Interests'].str.split(', ').sum()

# counting the frequency of each interest
interests_counter = Counter(interests_list)

# converting the counter object to a DataFrame for easier plotting
interests_df = pd.DataFrame(interests_counter.items(), columns=['Interest', 'Frequency']).sort_values(by='Frequency', ascending=False)

# plotting the most common interests
plt.figure(figsize=(12, 8))
sns.barplot(x='Frequency', y='Interest', data=interests_df.head(10), palette='twilight')
plt.title('Top 10 User Interests')
plt.xlabel('Frequency')
plt.ylabel('Interest')
plt.show()
```



To effectively target ad campaigns, it's crucial to segment users into distinct groups based on demographics, behavior, and interests. This segmentation can involve factors like age, gender, income level, education level, time spent online, likes and reactions, click-through rates (CTR), and conversion rates. By clustering users or creating personas using a combination of these attributes, personalized ad campaigns can be developed. This approach aims to improve user engagement and conversion rates by aligning ad content with users' interests and behaviors.

In [9]:
```
# selecting features for clustering
features = ['Age', 'Gender', 'Income Level', 'Time Spent Online (hrs/weekday)', 'Time Spent Online (hrs/weekend)', 'Likes and Reactions', 'Click-Through Rates (CTR)']

# separating the features we want to consider for clustering
X = df[features]

# defining preprocessing for numerical and categorical features
numeric_features = ['Time Spent Online (hrs/weekday)', 'Time Spent Online (hrs/weekend)', 'Likes and Reactions', 'Click-Through Rates (CTR)']
numeric_transformer = StandardScaler()

categorical_features = ['Age', 'Gender', 'Income Level']
categorical_transformer = OneHotEncoder()

# combining preprocessing steps
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# creating a preprocessing and clustering pipeline
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('cluster', KMeans(n_clusters=5, random_state=42))])

pipeline.fit(X)
cluster_labels = pipeline.named_steps['cluster'].labels_
df['Cluster'] = cluster_labels

print(df.head())
```

```
C:\Users\komal\AppData\Roaming\Python\Python38\site-packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.
4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
   User ID    Age  Gender  Location Language Education Level  \
0        1  25-34  Female  Suburban    Hindi       Technical
1        2    65+    Male     Urban    Hindi             PhD
2        3  45-54  Female  Suburban  Spanish       Technical
3        4  35-44  Female     Rural  Spanish             PhD
4        5  25-34  Female     Urban  English       Technical

   Likes and Reactions  Followed Accounts  Device Usage  \
0                 5640                190   Mobile Only
1                 9501                375        Tablet
2                 4775                187   Mobile Only
3                 9182                152  Desktop Only
4                 6848                371   Mobile Only

   Time Spent Online (hrs/weekday)  Time Spent Online (hrs/weekend)  \
0                              4.5                              1.7
1                              0.5                              7.7
2                              4.5                              5.6
3                              3.1                              4.2
4                              2.0                              3.8

   Click-Through Rates (CTR)  Conversion Rates  Ad Interaction Time (sec)  \
0                      0.193             0.067                         25
1                      0.114             0.044                         68
2                      0.153             0.095                         80
3                      0.093             0.061                         65
4                      0.175             0.022                         99

  Income Level                                  Top Interests  Cluster
0      20k-40k                               Digital Marketing        2
1        0-20k                                   Data Science        1
2      60k-80k                            Fitness and Wellness        0
3        100k+                              Gaming, DIY Crafts        3
4      20k-40k  Fitness and Wellness, Investing and Finance, G...        2
```

The clustering analysis has effectively divided user base into five separate clusters, labeled as Clusters 0 through 4. Each cluster signifies a distinct amalgamation of chosen features such as age, gender, income level, online activity, and engagement indicators. These clusters provide a foundation for crafting focused ad campaigns that cater to the specific preferences and behaviors of each segment.

In [10]:
```
# computing the mean values of numerical features for each cluster
cluster_means = df.groupby('Cluster')[numeric_features].mean()

for feature in categorical_features:
    mode_series = df.groupby('Cluster')[feature].agg(lambda x: x.mode()[0])
    cluster_means[feature] = mode_series

print(cluster_means)
```

```
         Time Spent Online (hrs/weekday)  Time Spent Online (hrs/weekend)  \
Cluster
0                               3.911111                         5.212963
1                               1.559394                         6.002424
2                               3.019737                         2.584211
3                               3.080882                         5.774510
4                               1.809626                         3.839572

         Likes and Reactions  Click-Through Rates (CTR)    Age  Gender  \
Cluster
0                2409.620370                   0.149588  25-34  Female
1                5005.121212                   0.179836  35-44    Male
2                6861.587719                   0.170614  25-34    Male
3                7457.602941                   0.067971  25-34  Female
4                3021.219251                   0.056594  45-54  Female

        Income Level
Cluster
0           80k-100k
1           80k-100k
2           20k-40k
3             100k+
4             0-20k
```

Based on the cluster analysis, each segment has been assigned a name reflecting its distinctive characteristics derived from mean numerical values and predominant categories. The segments are summarized as follows:

Cluster 0 – “Weekend Warriors”: Demonstrates high weekend online activity, moderate engagement with likes and reactions, predominantly male aged 25-34 with an income level of 80k-100k.

Cluster 1 – “Engaged Professionals”: Exhibits balanced online activity, high engagement with likes and reactions, predominantly male aged 25-34 with a high income (100k+).

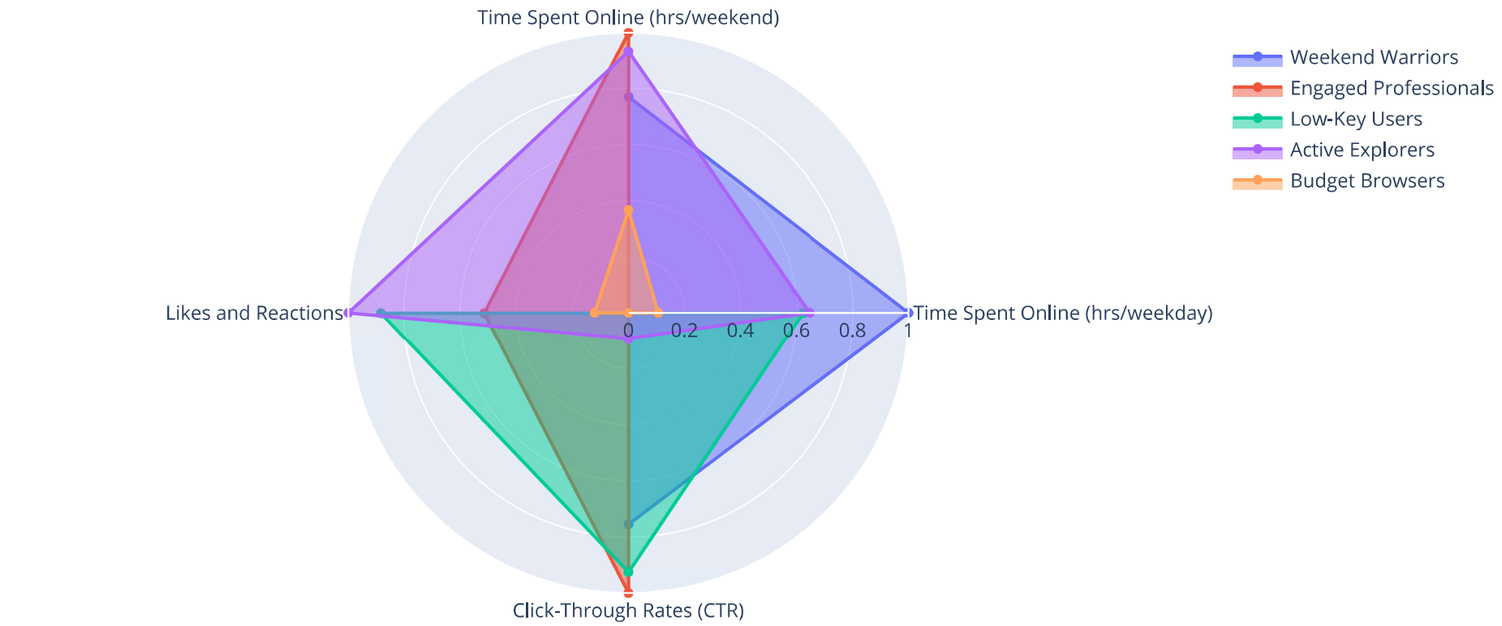
Cluster 2 – “Low-Key Users”: Displays moderate to high weekend online activity, moderate engagement with likes and reactions, predominantly male aged 25-34 with an income level of 60k-80k, and lower click-through rates (CTR).

Cluster 3 – “Active Explorers”: Shows high overall online activity, lower engagement with likes and reactions, predominantly female aged 25-34 with an income level of 60k-80k.

Cluster 4 – “Budget Browsers”: Features moderate online activity, the lowest engagement with likes and reactions, predominantly female aged 25-34 with the lowest income level (0-20k), and lower click-through rates (CTR).

In [11]:
```
# preparing data for radar chart
features_to_plot = ['Time Spent Online (hrs/weekday)', 'Time Spent Online (hrs/weekend)', 'Likes and Reactions', 'Click-Through Rates (CTR)']
labels = np.array(features_to_plot)

# creating a dataframe for the radar chart
radar_df = cluster_means[features_to_plot].reset_index()

# normalizing the data
radar_df_normalized = radar_df.copy()
for feature in features_to_plot:
    radar_df_normalized[feature] = (radar_df[feature] - radar_df[feature].min()) / (radar_df[feature].max() - radar_df[feature].min())

# adding a full circle for plotting
radar_df_normalized = radar_df_normalized.append(radar_df_normalized.iloc[0])

# assigning names to segments
segment_names = ['Weekend Warriors', 'Engaged Professionals', 'Low-Key Users', 'Active Explorers', 'Budget Browsers']
```

In [12]:
```
fig = go.Figure()

# loop through each segment to add to the radar chart
for i, segment in enumerate(segment_names):
    fig.add_trace(go.Scatterpolar(
        r=radar_df_normalized.iloc[i][features_to_plot].values.tolist() + [radar_df_normalized.iloc[i][features_to_plot].values[0]],  # Add the first value at the end to close t
        theta=labels.tolist() + [labels[0]],  # add the first label at the end to close the radar chart
        fill='toself',
        name=segment,
        hoverinfo='text',
        text=[f"{label}: {value:.2f}" for label, value in zip(features_to_plot, radar_df_normalized.iloc[i][features_to_plot])]+[f"{labels[0]}: {radar_df_normalized.iloc[i][fea
    ))

# update the layout to finalize the radar chart
fig.update_layout(
    polar=dict(
        radialaxis=dict(
            visible=True,
            range=[0, 1]
        )),
    showlegend=True,
    title='User Segments Profile'
)

fig.show()
```



The presented chart serves as a valuable tool for marketers, offering insights into various user segments' behaviors. This understanding enables marketers to refine their advertising strategies accordingly. For instance, advertisements aimed at the "Weekend Warriors" demographic could be strategically scheduled during weekends, leveraging their heightened activity during that time. Conversely, "Engaged Professionals" may exhibit more favorable responses to ads distributed evenly across the week.

In [ ]: