User Profiling and Segmentation using Python

```
User profiling refers to creating detailed profiles that represent the behaviours and preferences of users,
and segmentation divides the user base into distinct groups with common characteristics, making it easier
to target specific segments with personalized marketing, products, or services. If you want to learn how to
perform user profiling and segmentation for an advertisement campaign, this is for you.
```

```
User Profiling and Segmentation: Process We Can Follow
User profiling and segmentation are powerful techniques that enable data professionals to understand their
user base in-depth and tailor their strategies to meet diverse user needs. Below is the process we can
follow for the task of User Profiling and Segmentation:

Determine what you aim to achieve with user profiling and segmentation, such as improving customer service,
personalized marketing, or product recommendation.
Collect data from various sources, including user interactions on websites/apps, transaction histories,
social media activity, and demographic information.
Create new features that capture relevant user behaviours and preferences. It may involve aggregating
transaction data, calculating the frequency of activities, or extracting patterns from usage logs.
Select appropriate segmentation techniques.
For each segment identified, create user profiles that summarize the key characteristics and behaviours of
users in that segment.
```

In [1]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("C:\\Users\\kourg\\Downloads\\user_profiles_for_ads.csv")

print(data.head())
```

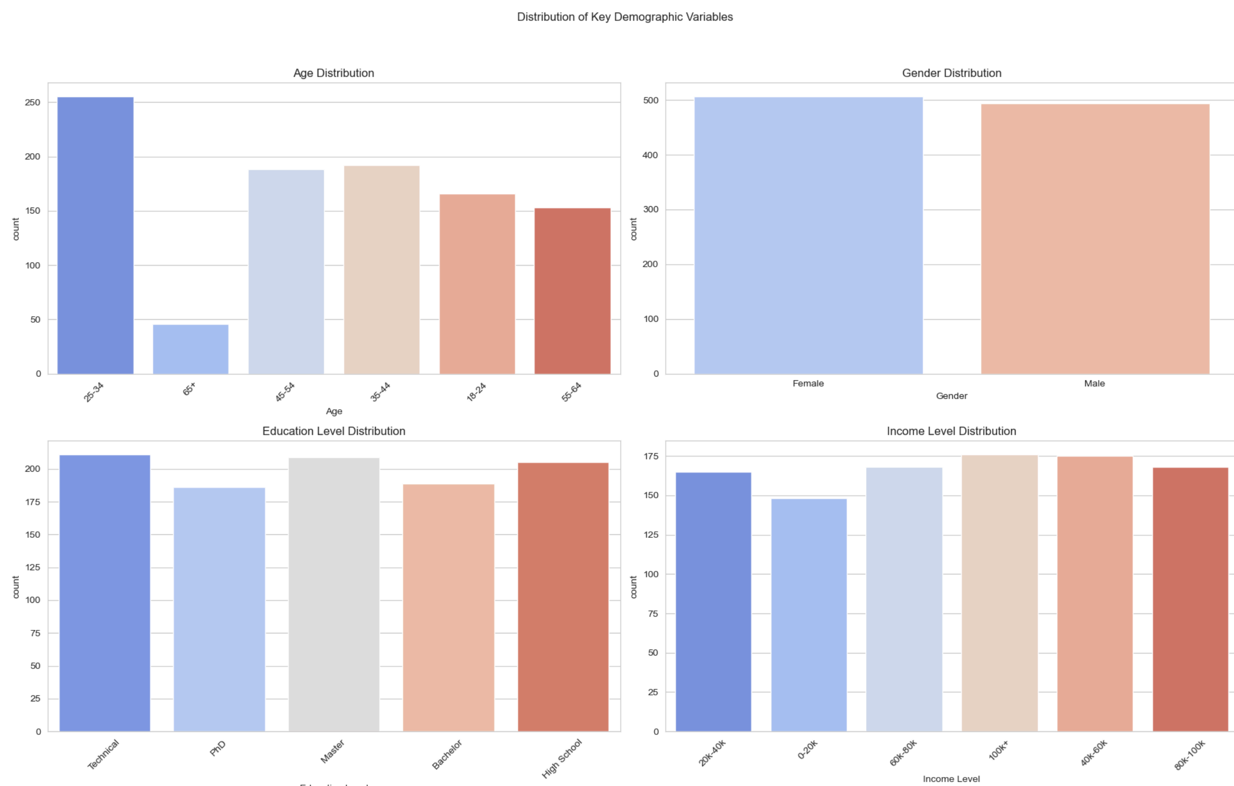
```
   User ID    Age  Gender  Location Language Education Level  \
0        1  25-34  Female  Suburban    Hindi       Technical
1        2    65+    Male     Urban    Hindi             PhD
2        3  45-54  Female  Suburban  Spanish       Technical
3        4  35-44  Female     Rural  Spanish             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
0      20k-40k                                 Digital Marketing
1         0-20k                                      Data Science
2      60k-80k                               Fitness and Wellness
3         100k+                                 Gaming, DIY Crafts
4      20k-40k  Fitness and Wellness, Investing and Finance, G...
```

In [2]:
```
print(data.isnull().sum())
```

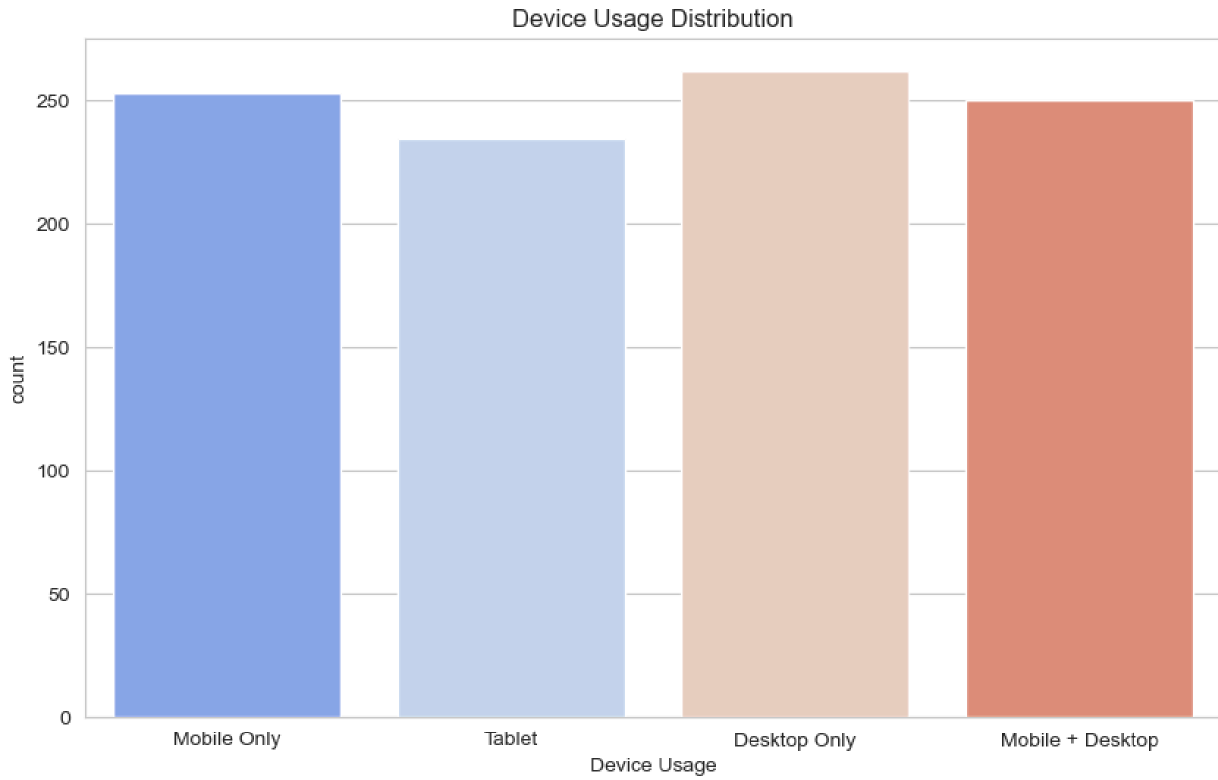
```
User ID                          0
Age                              0
Gender                           0
Location                         0
Language                         0
Education Level                  0
Likes and Reactions              0
Followed Accounts                0
Device Usage                     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 [3]:

```python
# setting the aesthetic style of the plots
sns.set_style("whitegrid")

# creating subplots for the demographic distributions
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=data, palette='coolwarm')
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=data, palette='coolwarm')
axes[0, 1].set_title('Gender Distribution')

# education level distribution
sns.countplot(ax=axes[1, 0], x='Education Level', data=data, palette='coolwarm')
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=data, palette='coolwarm')
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()
```

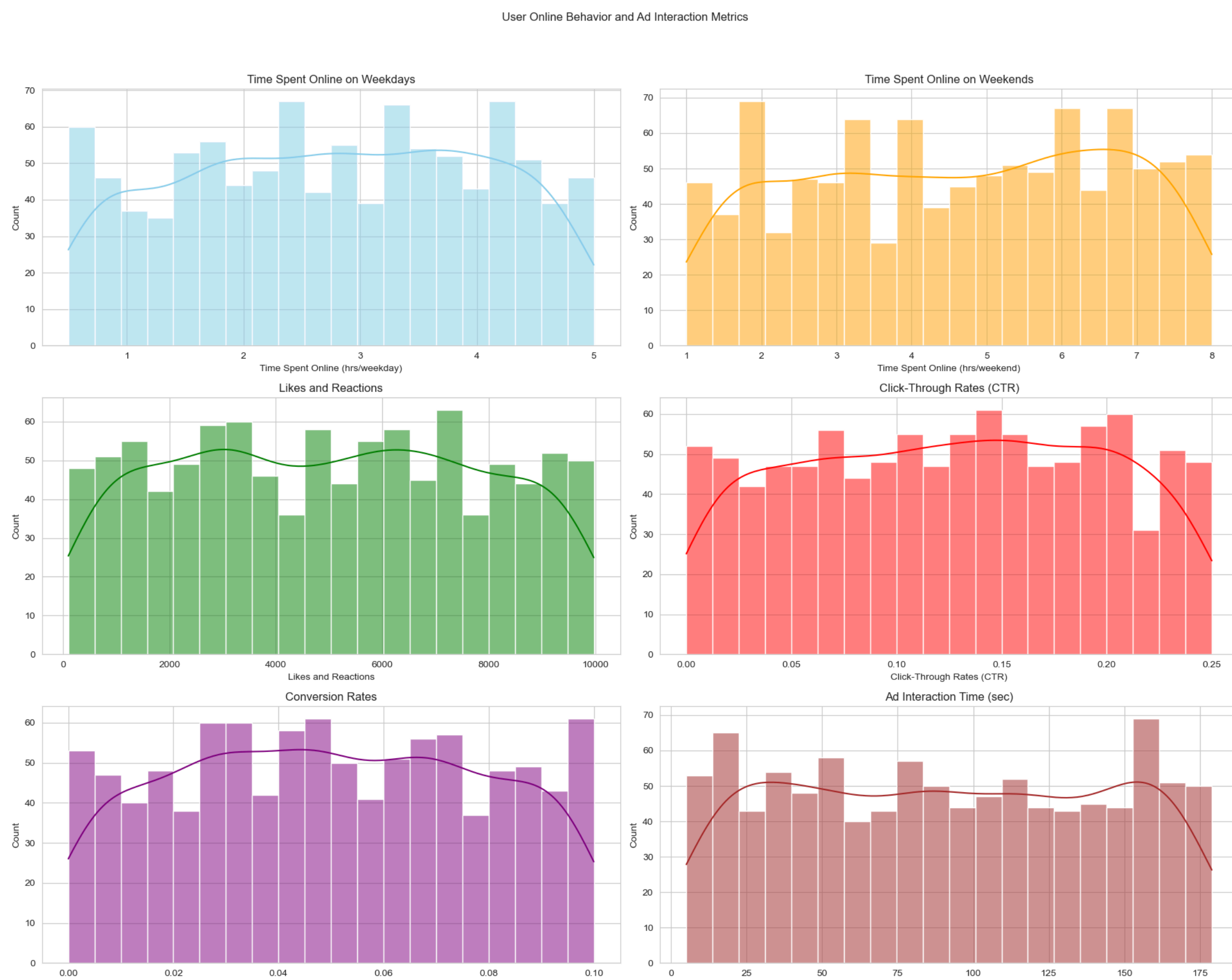


In [4]:
```
# device usage distribution
plt.figure(figsize=(10, 6))
sns.countplot(x='Device Usage', data=data, palette='coolwarm')
plt.title('Device Usage Distribution')
plt.show()
```



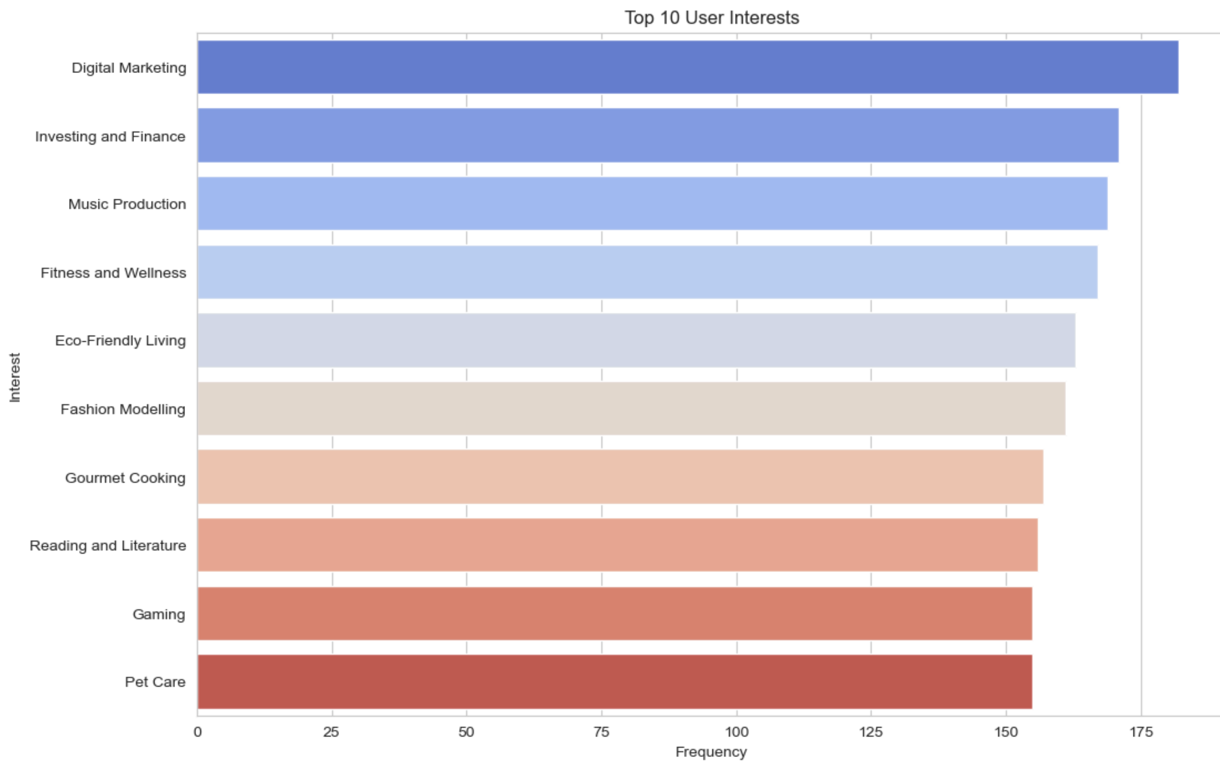
In [5]:

```
# creating subplots for 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=data, bins=20, kde=True, color='sky
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=data, bins=20, kde=True, color='ora
axes[0, 1].set_title('Time Spent Online on Weekends')

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

# click-through rates
sns.histplot(ax=axes[1, 1], x='Click-Through Rates (CTR)', data=data, 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=data, bins=20, kde=True, color='purple')
axes[2, 0].set_title('Conversion Rates')

# ad interaction time
sns.histplot(ax=axes[2, 1], x='Ad Interaction Time (sec)', data=data, 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()
```



In [6]:

```
from collections import Counter

# splitting the 'Top Interests' column and creating a list of all interests
interests_list = data['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=

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




```
User Profiling and Segmentation
We can now segment users into distinct groups for targeted ad campaigns. Segmentation can be based on
various criteria, such as:

Demographics: Age, Gender, Income Level, Education Level
Behavioural: Time Spent Online, Likes and Reactions, CTR, Conversion Rates
Interests: Aligning ad content with the top interests identified
To implement user profiling and segmentation, we can apply clustering techniques or develop personas based
on the combination of these attributes. This approach enables the creation of more personalized and
effective ad campaigns, ultimately enhancing user engagement and conversion rates.

Let’s start by selecting a subset of features that could be most indicative of user preferences and
behaviour for segmentation and apply a clustering algorithm to create user segments:
```

In [7]:

```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.cluster import KMeans

# selecting features for clustering
features = ['Age', 'Gender', 'Income Level', 'Time Spent Online (hrs/weekday)', 'Time Spent Online (hrs/w

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

# defining preprocessing for numerical and categorical features
numeric_features = ['Time Spent Online (hrs/weekday)', 'Time Spent Online (hrs/weekend)', 'Likes and Reac
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_
data['Cluster'] = cluster_labels

print(data.head())
```

```
C:\Users\kourg\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default
value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppres
s the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\kourg\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is know
n to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can
avoid it by setting the environment variable OMP_NUM_THREADS=4.
  warnings.warn(
```

```
   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 process has successfully segmented our users into five distinct groups (Clusters 0 to 4). Each cluster represents a unique combination of the features we selected, including age, gender, income level, online behaviour, and engagement metrics. These clusters can serve as the basis for creating targeted ad campaigns tailored to the preferences and behaviours of each segment.

We'll compute the mean values of the numerical features and the mode for categorical features within each cluster to get a sense of their defining characteristics:

In [8]:

```python
# computing the mean values of numerical features for each cluster
cluster_means = data.groupby('Cluster')[numeric_features].mean()

for feature in categorical_features:
    mode_series = data.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
```

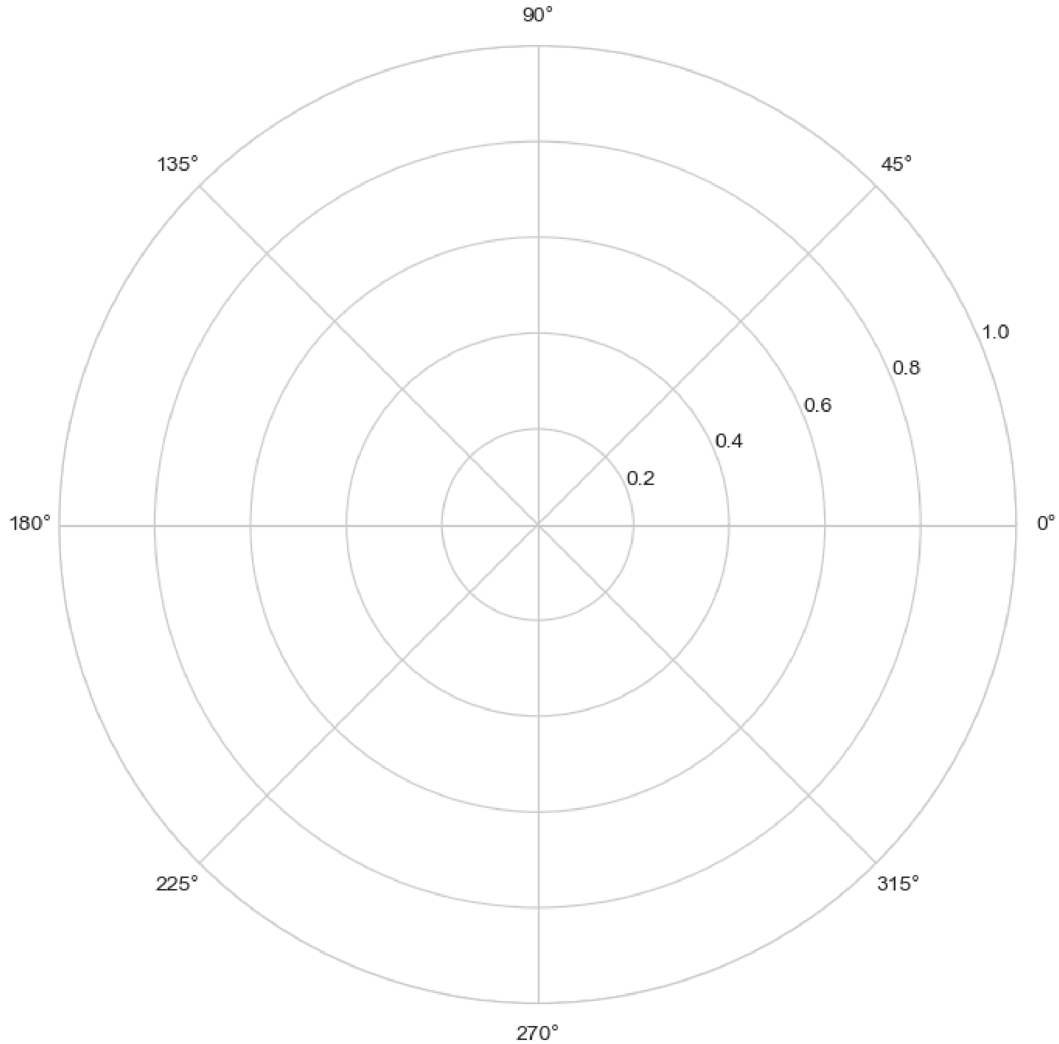
Now, we’ll assign each cluster a name that reflects its most defining characteristics based on the mean values of numerical features and the most frequent categories for categorical features. Based on the cluster analysis, we can summarize and name the segments as follows:

Cluster 0 – “Weekend Warriors”: High weekend online activity, moderate likes and reactions, predominantly male, age group 25-34, income level 80k-100k.
Cluster 1 – “Engaged Professionals”: Balanced online activity, high likes and reactions, predominantly male, age group 25-34, high income (100k+).
Cluster 2 – “Low-Key Users”: Moderate to high weekend online activity, moderate likes and reactions, predominantly male, age group 25-34, income level 60k-80k, lower CTR.
Cluster 3 – “Active Explorers”: High overall online activity, lower likes and reactions, predominantly female, age group 25-34, income level 60k-80k.
Cluster 4 – “Budget Browsers”: Moderate online activity, lowest likes and reactions, predominantly female, age group 25-34, lowest income level (0-20k), lower CTR.

In [12]:

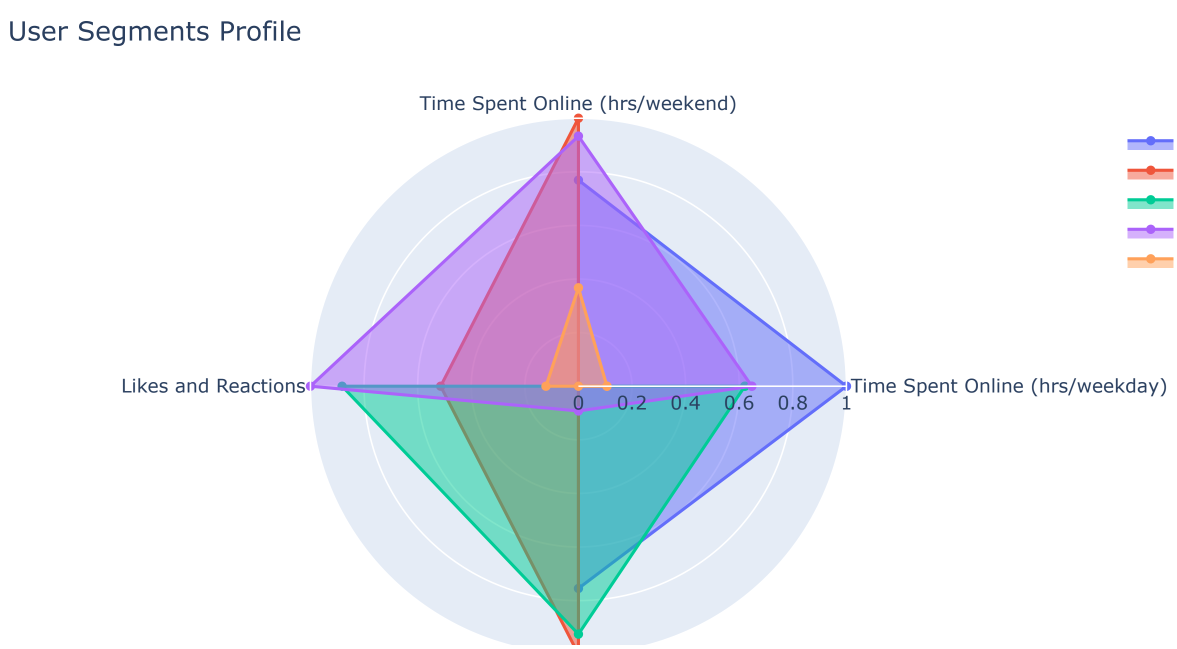
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from math import pi

# Assuming cluster_means is your DataFrame

# preparing data for radar chart
features_to_plot = ['Time Spent Online (hrs/weekday)', 'Time Spent Online (hrs/weekend)', 'Likes and Reac
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

# number of variables
num_vars = len(labels)

# calculate angle of each axis
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

# create the radar plot
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))

# add a full circle for plotting
values = radar_df_normalized.iloc[0].values.flatten().tolist()
values += values[:1]
angles += angles[:1]
ax.fill(angles, values, color='b', alpha=0.25)

# add labels
ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)
ax.set_title('Radar Chart for Clusters')

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

# add data for each cluster
for i in range(len(radar_df_normalized)):
    values = radar_df_normalized.iloc[i].values.flatten().tolist()
    values += values[:1]
    ax.plot(angles, values, label=segment_names[i])

# add legend
ax.legend(loc='upper right', bbox_to_anchor=(1.3, 1.1))

plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[12], line 33
     31 values += values[:1]
     32 angles += angles[:1]
---> 33 ax.fill(angles, values, color='b', alpha=0.25)
     35 # add labels
     36 ax.set_yticklabels([])

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:5226, in Axes.fill(self, data, *args, **kwa
rgs)
   5224 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
   5225 # _get_patches_for_fill returns a generator, convert it to a list.
-> 5226 patches = [*self._get_patches_for_fill(*args, data=data, **kwargs)]
   5227 for poly in patches:
   5228     self.add_patch(poly)

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:311, in _process_plot_var_args.__call__(sel
f, data, *args, **kwargs)
    309     this += args[0],
    310     args = args[1:]
--> 311 yield from self._plot_args(
    312     this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datakey)

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:504, in _process_plot_var_args._plot_args(s
elf, tup, kwargs, return_kwargs, ambiguous_fmt_datakey)
    501     self.axes.yaxis.update_units(y)
    503 if x.shape[0] != y.shape[0]:
--> 504     raise ValueError(f"x and y must have same first dimension, but "
    505                      f"have shapes {x.shape} and {y.shape}")
    506 if x.ndim > 2 or y.ndim > 2:
    507     raise ValueError(f"x and y can be no greater than 2D, but have "
    508                      f"shapes {x.shape} and {y.shape}")

ValueError: x and y must have same first dimension, but have shapes (5,) and (6,)
```



Now, let’s create a visualization that reflects these segments, using the cluster means for numerical features and highlighting the distinctive characteristics of each segment. We’ll create a radar chart that compares the mean values of selected features across the clusters, providing a visual representation of each segment’s profile:

In [14]:
```
# Assuming you have a list of segment names somewhere in your code
segment_names = ['Weekend Warriors', 'Engaged Professionals', 'Low-Key Users', 'Active Explorers', 'Budge

# Your Plotly code goes here
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][fe
        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[
    ))

# 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 chart above is useful for marketers to understand the behaviour of different user segments and tailor their advertising strategies accordingly. For example, ads targeting the “Weekend Warriors” could be scheduled for the weekend when they are most active, while “Engaged Professionals” might respond better to ads that are spread evenly throughout the week.

Summary

So, this is how you can perform User Profiling and Segmentation using Python. User profiling refers to creating detailed profiles that represent the behaviours and preferences of users, and segmentation divides the user base into distinct groups with common characteristics, making it easier to target specific segments with personalized marketing, products, or services.

So, this is how you can perform User Profiling and Segmentation using Python. User profiling refers to creating detailed profiles that represent the behaviours and preferences of users, and segmentation divides the user base into distinct groups with common characteristics, making it easier to target specific segments with personalized marketing, products, or services.