```
from sklearn.preprocessing import LabelEncoder
from scipy.stats import chi2_contingency
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import pandas as pd
```

```
df = pd.read_csv('/content/Netflix Userbase.csv', index_col='User ID')
df.sample(7)
```

User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Dura
475	Basic	11	13-08-22	26-06-23	Germany	31	Female	Smart TV	1 M
203	Standard	13	31-07-22	24-06-23	United Kingdom	42	Female	Smart TV	1 M
874	Premium	10	23-06-22	02-07-23	Spain	31	Male	Smart TV	1 M
825	Basic	12	09-08	30-06-	Germany	30	Female	Laptop	1 M

```
df.rename(
    columns={
        'Subscription Type': 'Sub_type',
        'Monthly Revenue': 'Monthly_revenue',
        'Join Date': 'First_sub_date',
```

```
        'Last Payment Date': 'Last_sub_date',
        'Plan Duration': 'Plan_duration',
    },
    inplace=True
)

df.index.rename('ID', inplace=True)

df.sample(7)
```

ID	Sub_type	Monthly_revenue	First_sub_date	Last_sub_date	Country	Age	Gend
1092	Basic	15	21-06-22	02-07-23	Canada	41	M
405	Basic	13	06-08-22	26-06-23	Germany	49	Fem
1629	Standard	14	12-09-22	04-07-23	Canada	40	M
1408	Standard	14	03-11-22	04-07-23	United Kingdom	40	Fem
1262	Basic	12	16-10-22	03-07-23	Brazil	46	M

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

```
Sub_type          0
Monthly_revenue   0
First_sub_date    0
Last_sub_date     0
Country           0
Age               0
Gender            0
Device            0
Plan_duration     0
dtype: int64
```

```
if df.duplicated().any():
    print("The same values Found!")
else:
    print("The same values not found.")
```

```
 The same values not found.
```

```
formatted_data = []
column_name_width = 20
column_value_width = 25

for column_name, column_value in df.loc[1].items():
    column_dtype = df[column_name].dtype
    formatted_data.append(f"{column_name.ljust(column_name_width)}{str(column_value).ljust(column_value_width)}{column_dtype

sample_output = "\n".join(formatted_data)
print(sample_output)
```

```
Sub_type            Basic                   object
Monthly_revenue     10                      int64
First_sub_date      15-01-22                object
Last_sub_date       10-06-23                object
Country             United States           object
Age                 28                      int64
Gender              Male                    object
Device              Smartphone              object
Plan_duration       1 Month                 object
```

```
df['First_sub_date'] = pd.to_datetime(df['First_sub_date'], format='%d-%m-%y')
df['Last_sub_date'] = pd.to_datetime(df['Last_sub_date'], format='%d-%m-%y')

df['Sub_period'] = (df['Last_sub_date'] - df['First_sub_date'])
df['Sub_period'] = df['Sub_period'].dt.days
```

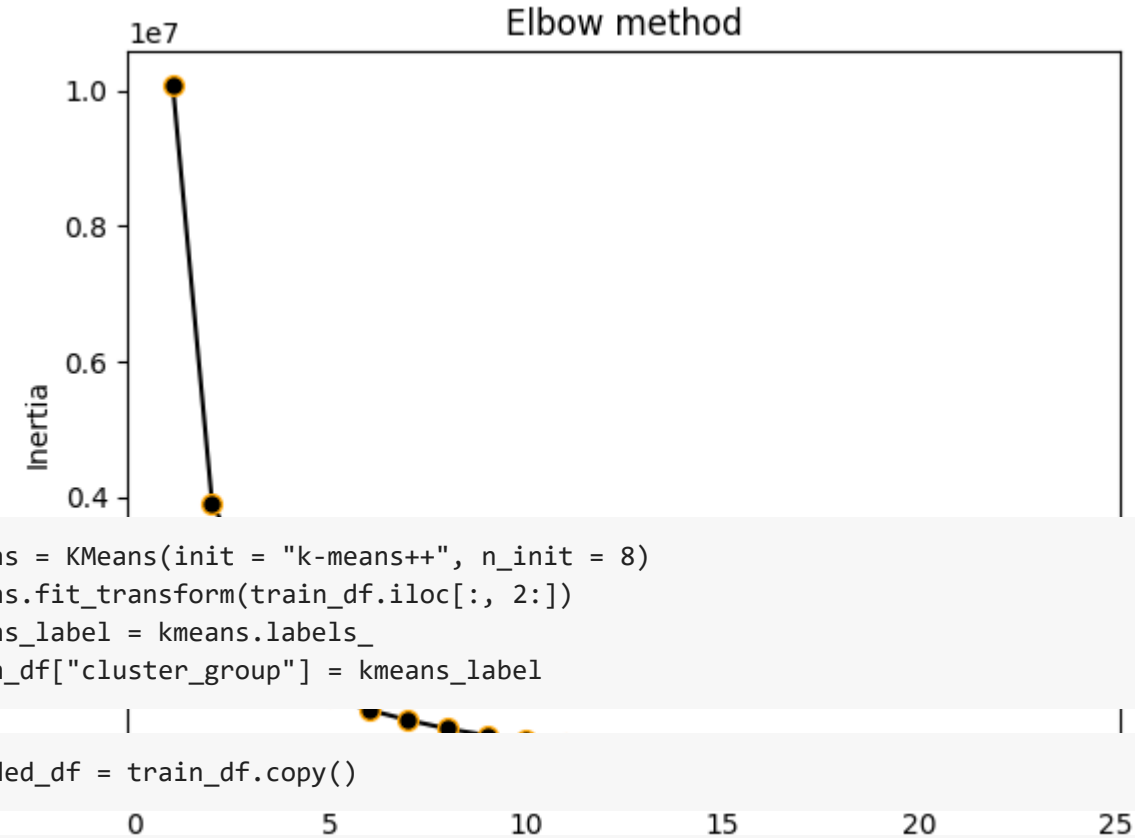
```
sub_type_encoder = LabelEncoder()
gender_encoder = LabelEncoder()
device_encoder = LabelEncoder()
```

```
train_df = df[['Sub_type', 'Age', 'Gender', 'Device', 'Sub_period']].copy()
train_df['Sub_type'] = sub_type_encoder.fit_transform(train_df['Sub_type'])
train_df['Gender'] = gender_encoder.fit_transform(train_df['Gender'])
train_df['Device'] = device_encoder.fit_transform(train_df['Device'])
train_df.sample(50)
```

143	2	30	0	3	297
860	0	50	1	1	309
139	2	40	1	2	91
708	2	39	1	0	241
1603	2	30	0	2	285
179	1	30	0	3	205
732	0	47	1	3	375
1075	0	43	1	1	331
1453	2	36	1	2	325
754	2	41	1	0	323
699	2	32	0	2	266
2143	2	46	0	2	359
1521	1	49	1	3	333
935	0	44	1	2	329
1691	1	27	0	1	242
879	2	42	0	3	376
2160	0	45	0	2	292
733	2	51	0	0	376
1642	0	43	1	3	359
245	0	47	1	1	432
2165	0	38	1	0	267
1515	2	32	1	1	366
2162	0	43	0	3	280
1532	0	44	0	3	284

824	1	47	0	2	334
1276	0	31	0	0	251
2182	2	45	0	3	243
1686	1	40	1	0	249
893	2	42	0	1	336
2456	1	39	1	2	244
189	1	47	1	1	297
55	0	39	0	1	69

```
inertias = list()
for i in range(1,25):
    kmeans = KMeans(init = "k-means++", n_clusters = i, n_init = 30)
    kmeans.fit_transform(train_df) # We are interested in all columns, excluding gender and race
    inertias.append(kmeans.inertia_)

plt.plot(range(1,25), inertias, marker='o', color = 'black', mec = 'orange', ms = 7)
plt.title('Elbow method')
plt.xlabel('No. clusters')
plt.ylabel('Inertia')
plt.show()
```




```
kmeans = KMeans(init = "k-means++", n_init = 8)
kmeans.fit_transform(train_df.iloc[:, 2:])
kmeans_label = kmeans.labels_
train_df["cluster_group"] = kmeans_label
```

```
decoded_df = train_df.copy()
```

```
decoded_df['Sub_type'] = sub_type_encoder.inverse_transform(decoded_df['Sub_type'])
decoded_df['Gender'] = gender_encoder.inverse_transform(decoded_df['Gender'])
decoded_df['Device'] = device_encoder.inverse_transform(decoded_df['Device'])
```

```
decoded_df.sample(20)
```

ID	Sub_type	Age	Gender	Device	Sub_period	cluster_group
591	Premium	35	Female	Smart TV	374	0
1082	Basic	27	Male	Tablet	369	0
56	Premium	36	Male	Tablet	469	3
314	Basic	28	Male	Laptop	431	3
1579	Standard	40	Female	Smart TV	372	0
404	Premium	28	Male	Smartphone	331	5
1691	Premium	27	Female	Smart TV	242	6
1854	Standard	46	Female	Laptop	357	0
653	Standard	28	Female	Smartphone	335	5
553	Standard	39	Male	Smart TV	305	2
1543	Standard	34	Female	Laptop	256	1
834	Standard	34	Female	Smart TV	280	1
71	Basic	27	Male	Smartphone	510	3
96	Premium	32	Male	Laptop	333	5
2238	Standard	49	Female	Smartphone	262	1

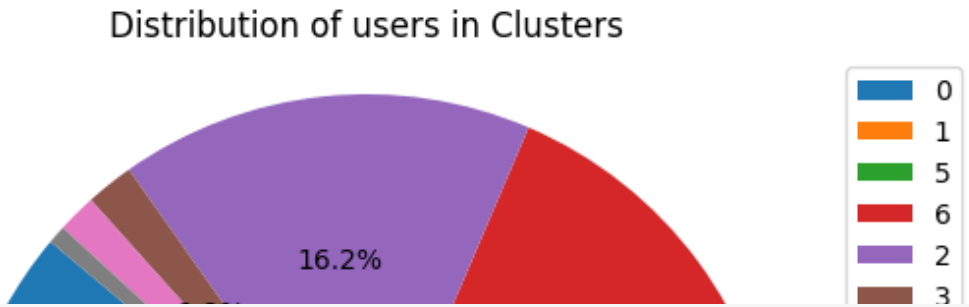
```
cluster_counts = decoded_df['cluster_group'].value_counts()

print("No. users in each cluster:")
print(cluster_counts)
```

```
No. users in each cluster:
0    645
1    481
5    445
6    421
```

```
2    405
3     47
4     38
7     18
Name: cluster_group, dtype: int64
```

```
cluster_counts = decoded_df['cluster_group'].value_counts()
```

```
plt.figure(figsize=(8, 6))
plt.pie(cluster_counts, labels=None, autopct='%1.1f%%', startangle=140)
plt.legend(labels=cluster_counts.index, loc='best')
plt.legend(labels=cluster_counts.index, loc='best')


plt.title('Distribution of users in Clusters')
plt.axis('equal')

plt.show()
```




```
gender_distribution = decoded_df.groupby(['cluster_group', 'Gender'])['Gender'].count().unstack()

print("Gender Distribution in Clusters:")
print(gender_distribution)
```

```
Gender Distribution in Clusters:
Gender         Female  Male
cluster_group
0                 321   324
1                 230   251
2                 203   202
3                  24    23
4                  22    16
5                 225   220
6                 225   196
7                   7    11
```

```
total_by_cluster = gender_distribution.sum(axis=1)
percentage_distribution = gender_distribution.divide(total_by_cluster, axis=0) * 100
```

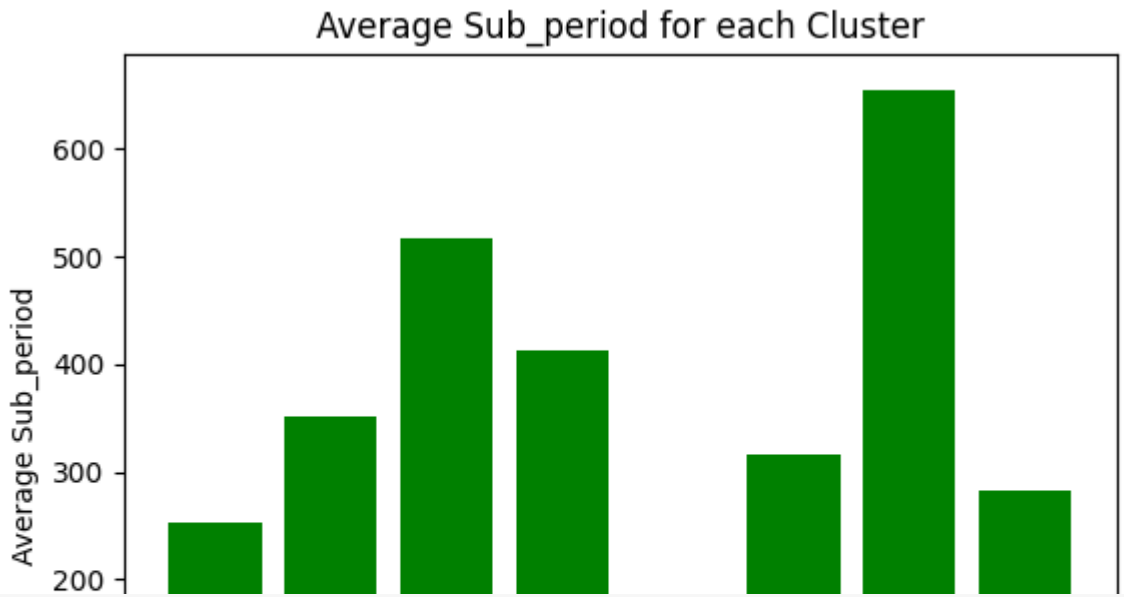
```
ax = percentage_distribution.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Gender distribution for each Clusters')
plt.xlabel('Cluster')
plt.ylabel('Percentage')
plt.legend(["Female", "Male"], loc='upper right')

for i in ax.patches:
    percentage = i.get_height()
    if 49.5 < percentage < 51: # Avoid overlapping near 50%
```

```
        percentage = 50

    ax.annotate(f'{percentage:.2f}%',
                (i.get_x() + i.get_width() / 2., i.get_height()),
                ha='center', va='center',
                fontsize=10, color='black',
                xytext=(0, 5),
                textcoords='offset points')

plt.xticks(rotation=0)
plt.show()
```


```
cluster_stats = decoded_df.groupby('cluster_group').agg({'Age': 'mean', 'Sub_period': 'mean'})

cluster_stats['Age'] = cluster_stats['Age'].astype(int)
cluster_stats['Sub_period'] = cluster_stats['Sub_period'].astype(int)

print("Cluster Statistics (Average Age and Sub_period[days]):")
print(cluster_stats)
```

```
Cluster Statistics (Average Age and Sub_period[days]):
                Age  Sub_period
cluster_group
0                39         368
1                38         265
2                38         299
3                35         455
4                37         107
5                38         333
6                38         238
7                39         580
```

```
plt.bar(decoded_df['cluster_group'], decoded_df['Sub_period'], color='green')
plt.title('Average Sub_period for each Cluster')
plt.xlabel('Cluster')
plt.ylabel('Average Sub_period')

plt.show()
```

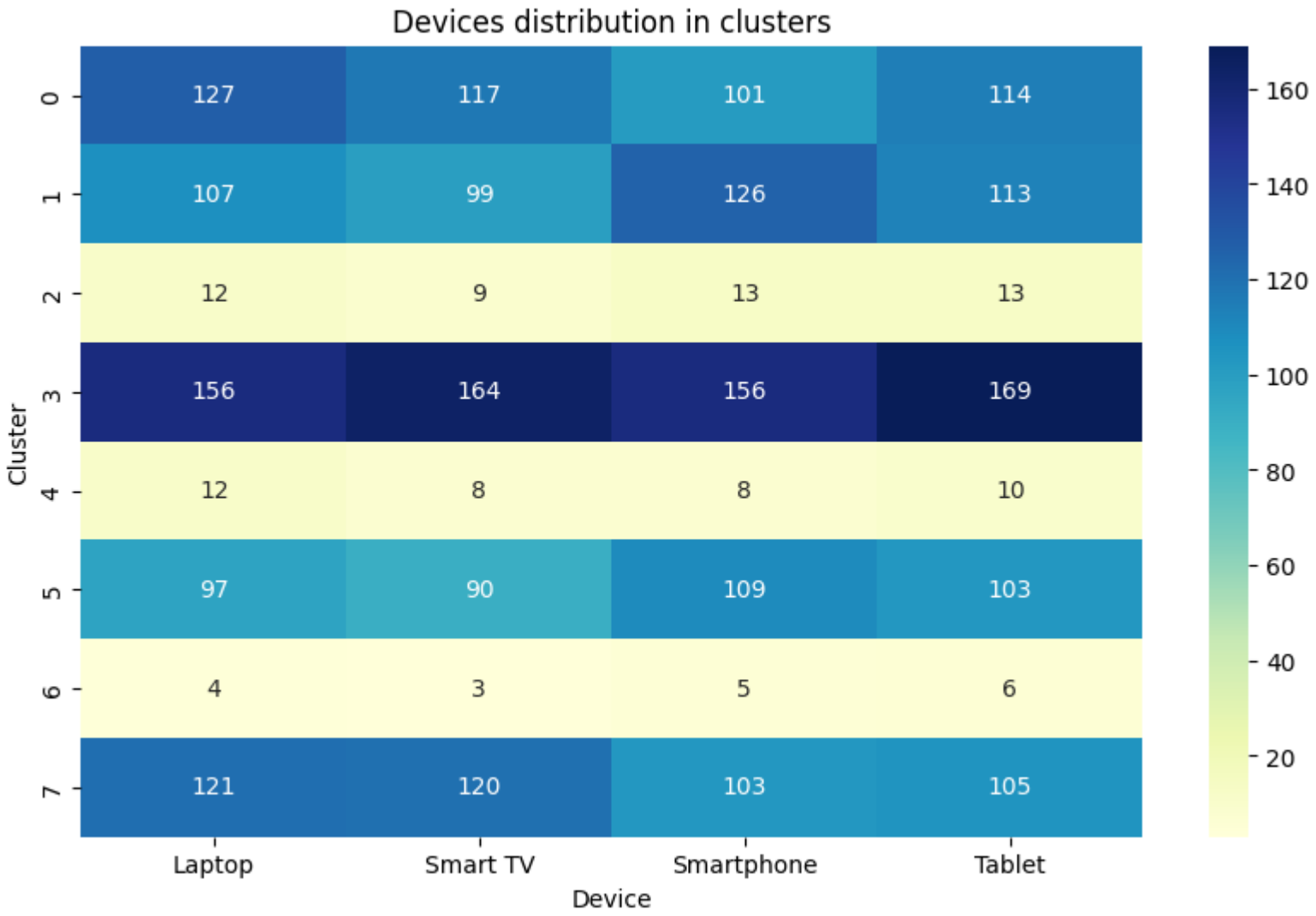



```
devices_distribution = decoded_df.groupby(['cluster_group', 'Device'])['Device'].count().unstack()

print("Devices Distribution in clusters:")
print(devices_distribution)
```

```
Devices Distribution in clusters:
Device         Laptop  Smart TV  Smartphone  Tablet
cluster_group
0                 127       117         101     114
1                 107        99         126     113
2                  12         9          13      13
3                 156       164         156     169
4                  12         8           8      10
5                  97        90         109     103
6                   4         3           5       6
7                 121       120         103     105
```

```
plt.figure(figsize=(10, 6))
sns.heatmap(devices_distribution, annot=True, fmt='d', cmap='YlGnBu')
plt.title('Devices distribution in clusters')
plt.xlabel('Device')
plt.ylabel('Cluster')
plt.show()
```

```
plt.show()
```


Devices distribution in clusters
Cluster
0
1
2
3
4
5
6
7
127
117
101
114
107
99
126
113
12
9
13
13
156
164
156
169
12
8
8
10
97
90
109
103
4
3
5
6
121
120
103
105
Laptop
Smart TV
Smartphone
Tablet
Device
160
140
120
100
80
60
40
20