CUSTOMER SEGMENTATION PROJECT
BY

OMOGBEME ANGELA
CHIDERA ONOH
ESTHER MAKANDAH

Agenda

- Background
- Objectives
- Methodology
- Executive Summary
- Findings
- Recommendations
- Appendix

BACKGROUND

- The project delves into customer segmentation understanding customer behavior and preferences and creating clusters using demographic indicators.
- Visualizing centroids to understand their distribution and proximity

INTRODUCTION

The objective of the project is to analyze the provided dataset and identify distinct customer segments based on various demographic and behavioral attributes. By leveraging on K-means clustering

METHODOLOGY

DATA COLLECTION AND CLEANING

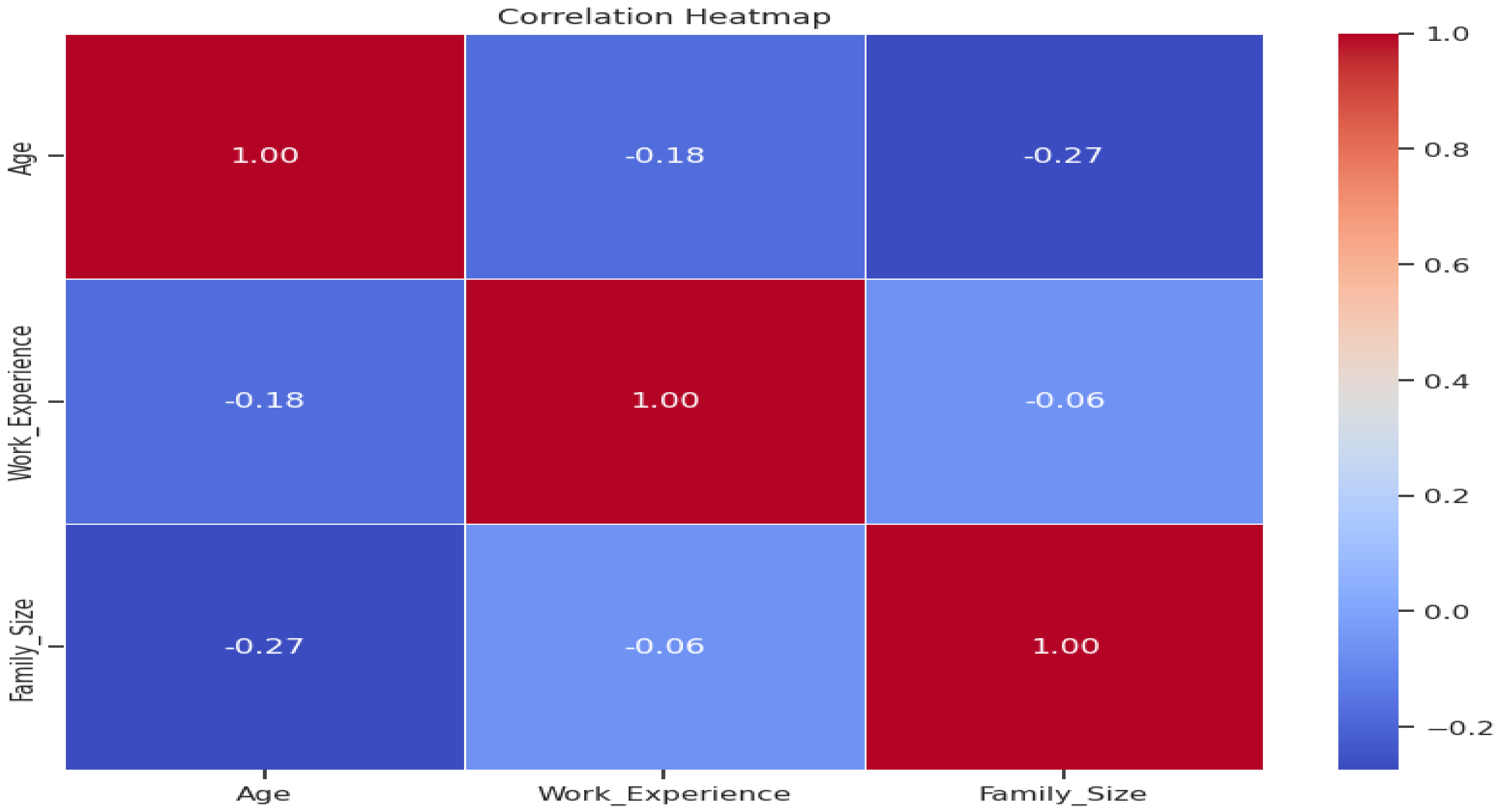
EXPLORATORY DATA ANALYSIS

CHOOSING THE NUMBERS OF CLUSTERS

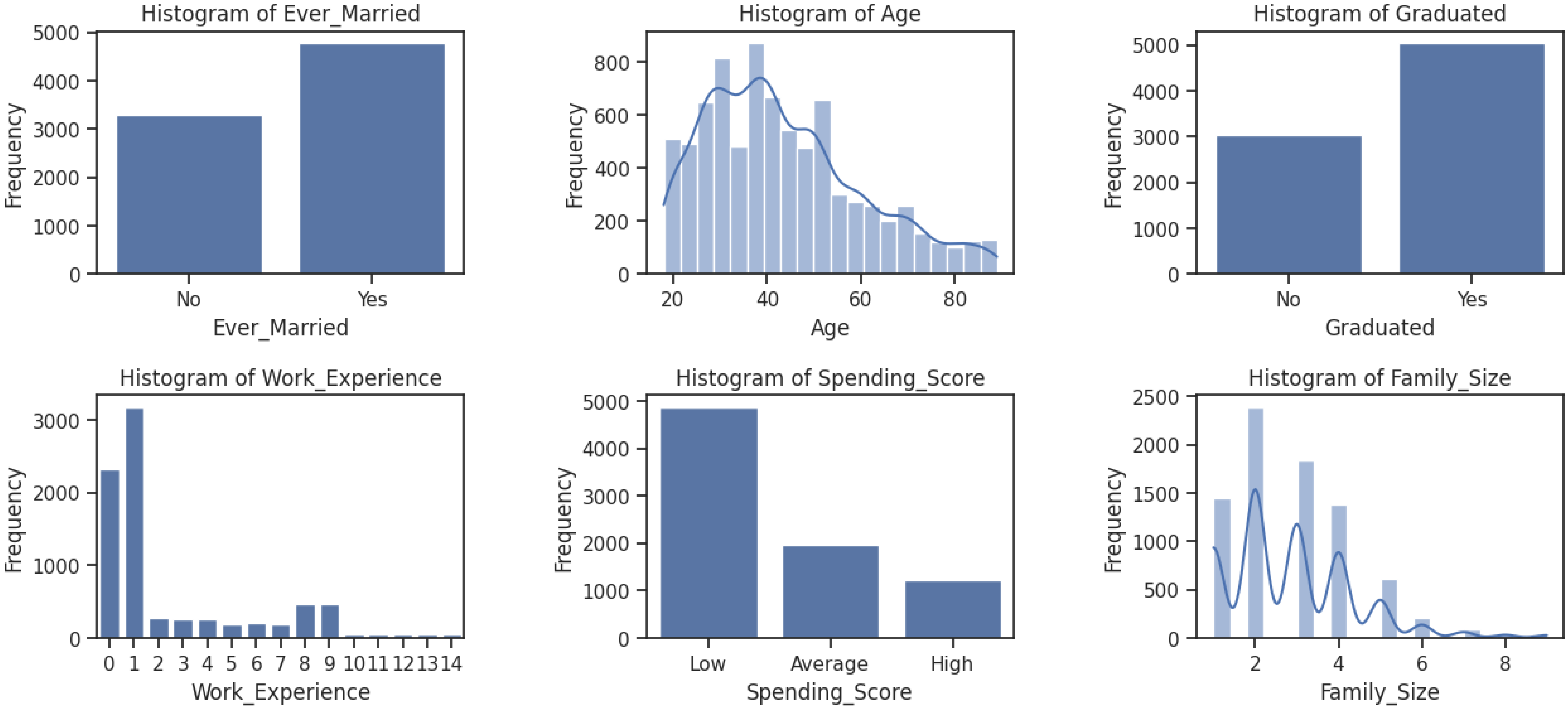
CHOOSING THE CENTROID

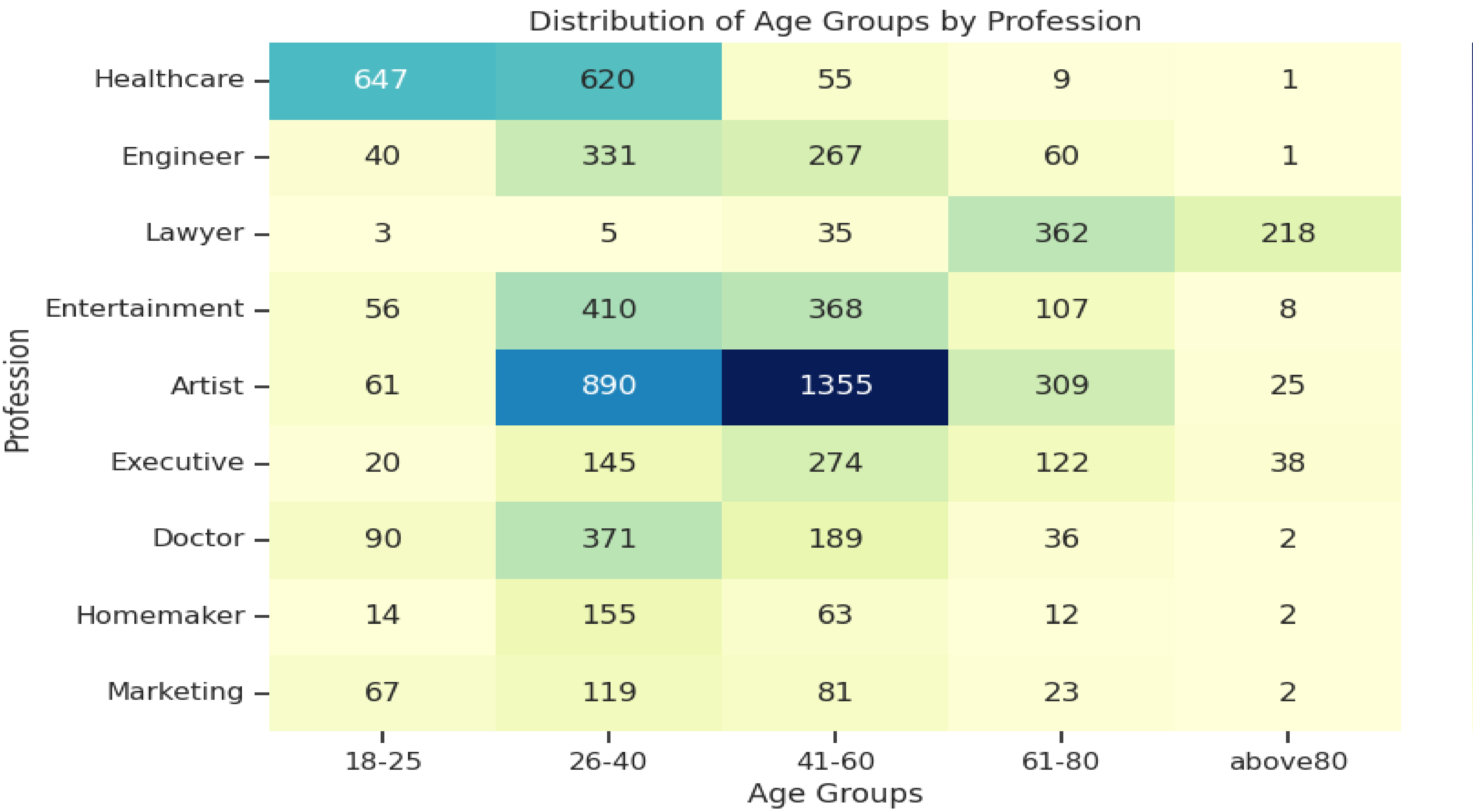
VALIDATION

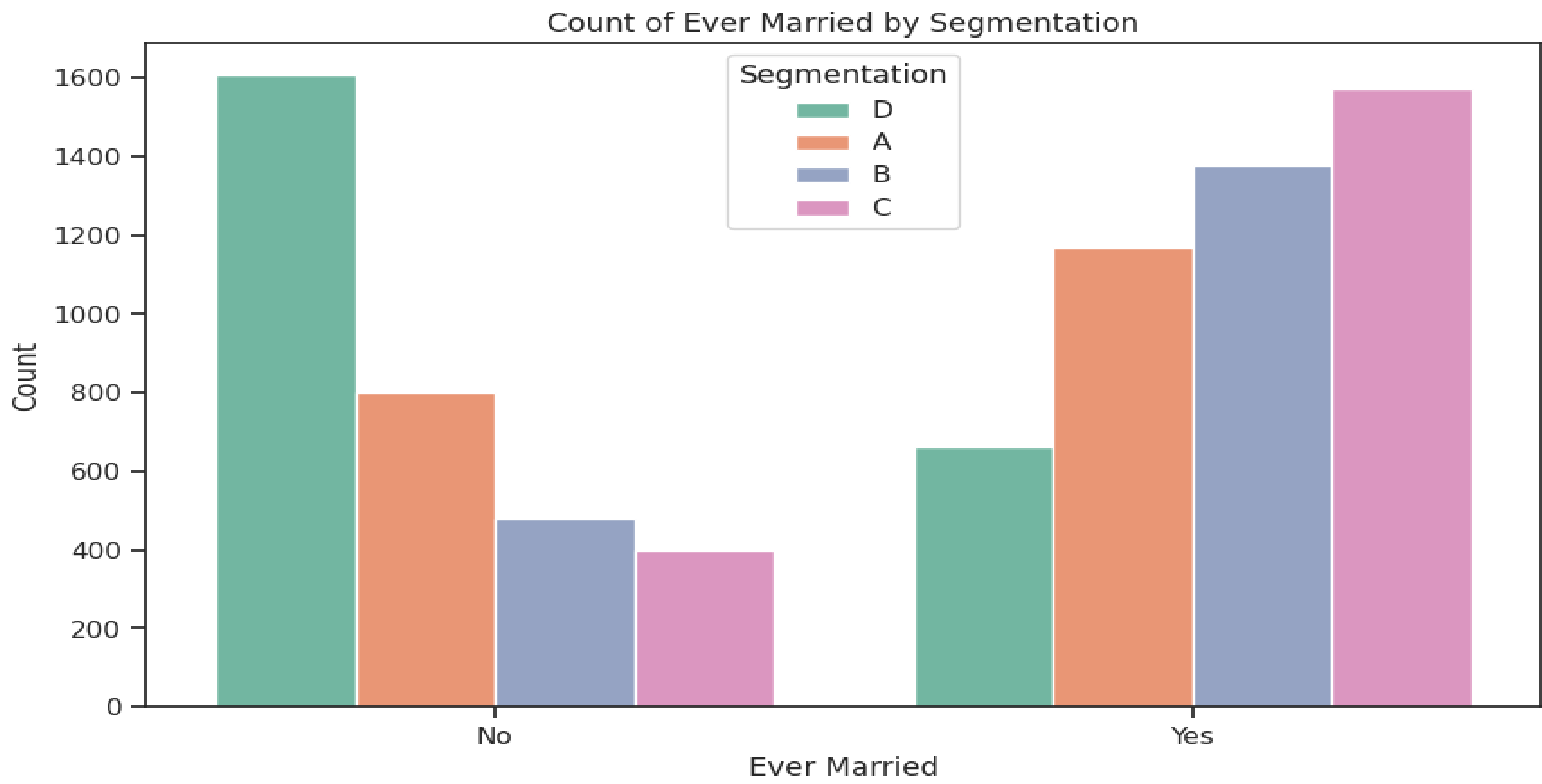
EXPLORATION DATA ANALYSIS



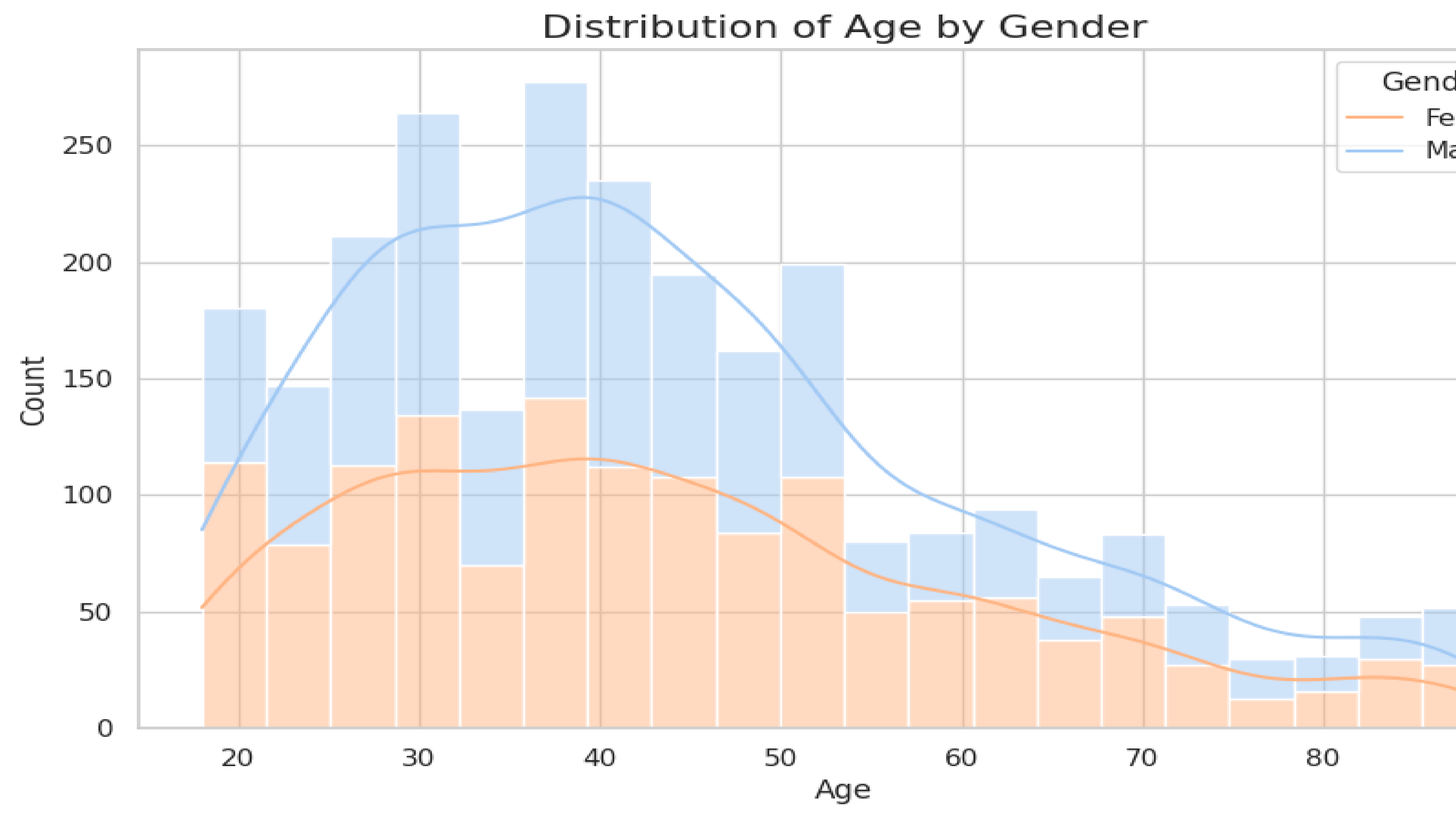
Correlation Heatmap
Age
Work_Experience
Family_Size
1.00
-0.18
-0.27
-0.18
1.00
-0.06
-0.27
-0.06
1.00
Age
Work_Experience
Family_Size
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

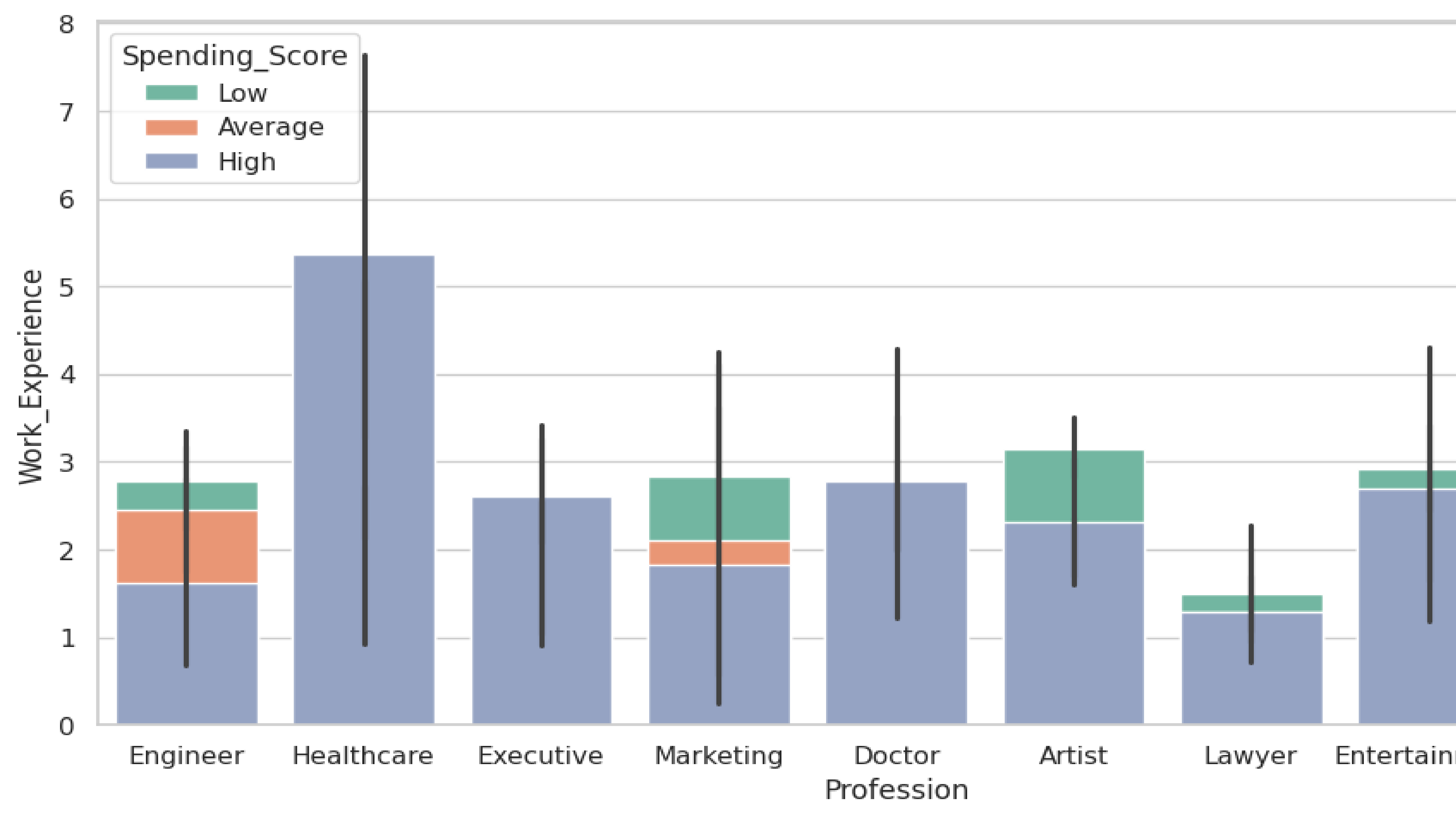

Histogram of Ever_Married
Frequency
0
1000
2000
3000
4000
5000
No
Yes
Ever_Married
Histogram of Age
Frequency
0
200
400
600
800
20
40
60
80
Age
Histogram of Graduated
Frequency
0
1000
2000
3000
4000
5000
No
Yes
Graduated
Histogram of Work_Experience
Frequency
0
1000
2000
3000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Work_Experience
Histogram of Spending_Score
Frequency
0
1000
2000
3000
4000
5000
Low
Average
High
Spending_Score
Histogram of Family_Size
Frequency
0
500
1000
1500
2000
2500
2
4
6
8
Family_Size


Distribution of Age Groups by Profession
Profession
Healthcare
Engineer
Lawyer
Entertainment
Artist
Executive
Doctor
Homemaker
Marketing
647 620 55 9 1
40 331 267 60 1
3 5 35 362 218
56 410 368 107 8
61 890 1355 309 25
20 145 274 122 38
90 371 189 36 2
14 155 63 12 2
67 119 81 23 2
18-25
26-40
41-60
61-80
above80
Age Groups

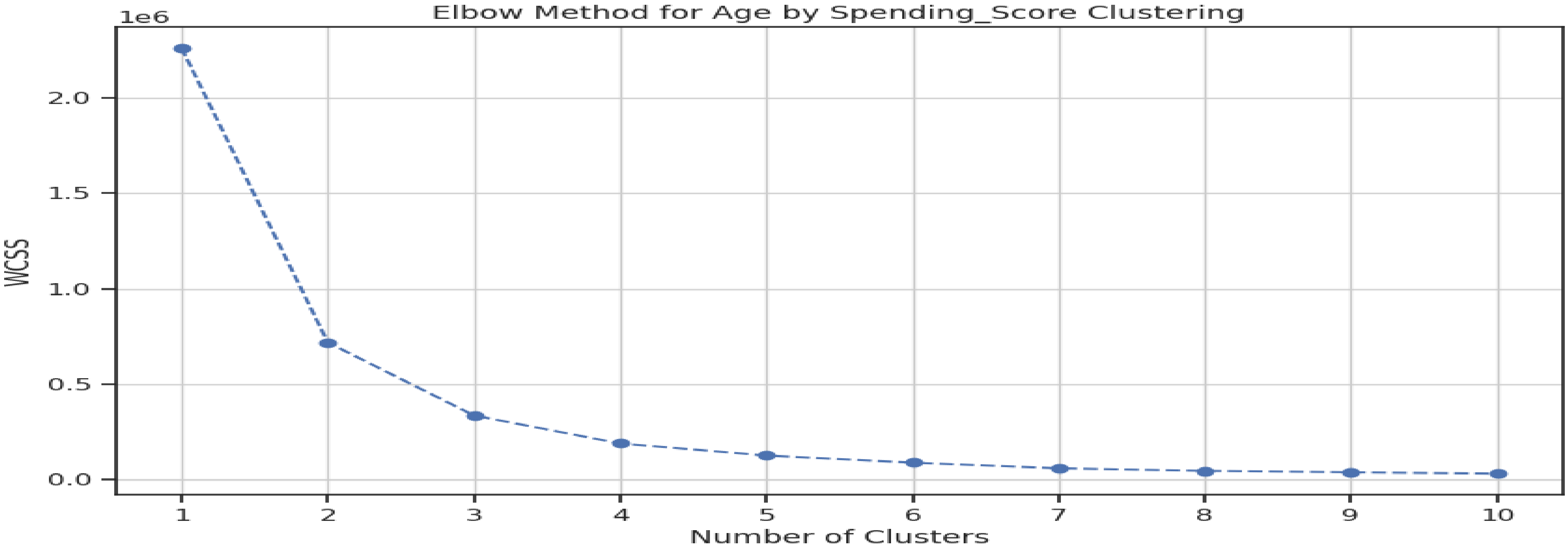


Count of Ever Married by Segmentation
Segmentation
D
A
B
C
Count
0
200
400
600
800
1000
1200
1400
1600
No
Yes
Ever Married


Distribution of Age by Gender
Count
Age
0
50
100
150
200
250
20
30
40
50
60
70
80


Spending_Score
Low
Average
High
Work_Experience
8
7
6
5
4
3
2
1
0
Engineer
Healthcare
Executive
Marketing
Doctor
Artist
Lawyer
Profession

CALCULATING THE NUMBER OF CLUSTERS



FINDINGS

- The customers population is mainly segmented by age, marital status, profession, and purchasing power.
- The data is segmented into 4 clusters.
- Cluster 0: Single people from the arts and entertainment sectors with low purchasing power.
- Cluster 1: Middle-aged, married people in the arts sector with average purchasing power.
- Cluster 2: Young, single people without higher education and with low purchasing power.
- Cluster 3: Older, married people with well-paying jobs and a high purchasing power.
- Artists within the age of 41-60 and 26-40 has the highest numbers of customers totalling(2,245) followed by healthcare workers within the ages of 18.25

RECOMMENDATIONS

While individuals in the Artists profession currently represent the largest customer segment, it is imperative to delve deeper into additional factors such as annual income and purchasing power to ascertain their impact on overall profitability. Understanding these variables will provide valuable insights into the potential value added by different customer segments,

LINES OF CODE

- **# import all the necessary libraries**
- import pandas as pd
- import numpy as np

- import matplotlib.pyplot as plt
- %matplotlib inline
- plt.style.use('ggplot')
- import seaborn as sns
- import plotly.express as px
- **# import and read files**
- test = pd.read_csv(r"/content/drive/MyDrive/Test.csv")
- train = pd.read_csv(r"/content/drive/MyDrive/Train.csv")

LINES OF CODE(CTD)

 # summary statistical analysis of the test data

- test.describe()

 # extract the categorical columns from the test data

- cat_cols = test.select_dtypes(include = ['object', 'category']).columns.tolist()
- cat_cols

Read the train dataset

- train_df = pd.read_csv(r"/content/drive/MyDrive/archive (2)/Train.csv")

- # Make a copy of the original DataFrame
- train_encoded_df = train_df.copy()
- # Define categorical columns for which you want to apply one-hot encoding
- categorical_columns = ['Gender', 'Ever_Married', 'Graduated', 'Profession', 'Spending_Score', 'Var_1']

LINES OF CODE(CTD)

```
# Perform one-hot encoding for each categorical column
for col in categorical_columns:
 # Perform one-hot encoding for the column
 encoded_df = pd.get_dummies(train_encoded_df[col], prefix=col)
# Concatenate the encoded columns to the copy of the original DataFrame
train_encoded_df = pd.concat([train_encoded_df, encoded_df], axis=1)# Drop the original categorical column
train_encoded_df.drop(columns=[col], inplace=True)

plt.figure(figsize=(15, 10))

features = ['Ever_Married', 'Age', 'Graduated','Work_Experience', 'Spending_Score', 'Family_Size']
```

LINES OF CODE(CTD)

```
for i, feature in enumerate(features,  start=1):
    plt.subplot(3, 3, i)
  plt.subplots_adjust(hspace=0.5, wspace=0.5)
   if feature == 'Age' or feature == 'Gender' or feature == 'Family_Size':
sns.histplot(df[feature], bins=20, kde=True)
else:A
   sns.countplot(data=df, x=feature)
plt.title('Histogram of {}'.format(feature))
plt.xlabel(feature)
 plt.ylabel('Frequency')
plt.show()
```

- import matplotlib.pyplot as plt
- import seaborn as sns

LINES OF CODE(CTD)

- professions = df['Profession'].unique()

for profession in professions: count_profession = [

len(df[(df.Age >= 18) & (df.Age <= 25) & (df.Profession == profession)]),

len(df[(df.Age >= 26) & (df.Age <= 40) & (df.Profession == profession)]),

- len(df[(df.Age >= 41) & (df.Age <= 60) & (df.Profession == profession)]),
- len(df[(df.Age >= 61) & (df.Age <= 80) & (df.Profession == profession)]),
- len(df[(df.Age > 80) & (df.Profession == profession)])]
- counts.append(count_profession)

Plotting

- plt.figure(figsize=(10, 6))
- sns.heatmap(counts, annot=True, fmt="d", cmap="YlGnBu", xticklabels=ages, yticklabels=professions)
- plt.xlabel('Age Groups')
- plt.ylabel('Profession')
- plt.title('Distribution of Age Groups by Profession')
- plt.show()

LINES OF CODE(CTD)

```
# Map spending score categories to numerical values
spending_score_mapping = {'Low': 0, 'Medium': 1, 'High': 2}
df['Spending_Score_Num'] = df['Spending_Score'].map(spending_score_mapping)

# Selecting the features (Age and Numerical Spending_Score)
X = df[['Age', 'Spending_Score_Num']].values

# Impute missing values with the mean
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)
# Using the Elbow Method to find the optimal number of clusters
wcss = []  # Within-cluster sum of squares

for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42)
    kmeans.fit(X_imputed)
    wcss.append(kmeans.inertia_)
```

LINES OF CODE(CTD)

```
# Plotting the Elbow Method graph
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Age by Spending_Score Clustering')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.xticks(np.arange(1, 11, 1))
plt.grid(True)
plt.show()
Define the optimal number of clusters based on the Elbow Method (e.g., 3 clusters)
n_clusters = 4
# Fit KMeans model
kmeans = KMeans(n_clusters=n_clusters, init='k-means++', random_state=42)
kmeans.fit(X_imputed)
```