Title: Analysis of Shop Customer Data

Problem to solve: Analyse Shop Customer Data

- Perform EDA
- Perform Hypothesis Testing
- Generate insights
- Give recommendations

Data source: Kaggle

In [1]:

```
# Import libraries
import pandas as pd
import numpy as np

# Import libraries for visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Import libraries for Hypothesis Testing
from scipy.stats import ttest_ind
from scipy.stats import shapiro
from scipy.stats import kruskal
from scipy.stats import chi2_contingency
```

In [2]:

```
# Read csv file
customers = pd.read_csv('customers.csv')

# First few rows of customers dataframe
customers.head()
```

Out[2]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
0	1	Male	19	15000	39	Healthcare	1	4
1	2	Male	21	35000	81	Engineer	3	3
2	3	Female	20	86000	6	Engineer	1	1
3	4	Female	23	59000	77	Lawyer	0	2
4	5	Female	31	38000	40	Entertainment	2	6

In [3]:

```
# Last few rows of customers dataframe
customers.tail()
```

Out[3]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
1995	1996	Female	71	184387	40	Artist	8	7
1996	1997	Female	91	73158	32	Doctor	7	7
1997	1998	Male	87	90961	14	Healthcare	9	2
1998	1999	Male	77	182109	4	Executive	7	2
1999	2000	Male	90	110610	52	Entertainment	5	2

In [4]:

```
# Size of the dataframe
customers.size
```

Out[4]:

```
16000
```

In [5]:

```
# Shape of the dataframe
customers.shape
```

Out[5]:

```
(2000, 8)
```

In [6]:

```
# Index of the dataframe
customers.index
```

Out[6]:

```
RangeIndex(start=0, stop=2000, step=1)
```

In [7]:

```
# Dimensions of the dataframe
customers.ndim
```

Out[7]:

```
2
```

In [8]:

```
# Columns of the dataframe
customers.columns
```

Out[8]:

```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income ($)',
       'Spending Score (1-100)', 'Profession', 'Work Experience',
       'Family Size'],
      dtype='object')
```

In [9]:

```
# Datatypes of the dataframe
customers.dtypes
```

Out[9]:

```
CustomerID                 int64
Gender                    object
Age                        int64
Annual Income ($)          int64
Spending Score (1-100)     int64
Profession                object
Work Experience            int64
Family Size                int64
dtype: object
```

6 int64 and 2 object columns are present

In [10]:

```
# Statistical analysis of the dataframe
customers.describe()
```

Out[10]:

	CustomerID	Age	Annual Income ($)	Spending Score (1-100)	Work Experience	Family Size
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1000.500000	48.960000	110731.821500	50.962500	4.102500	3.768500
std	577.494589	28.429747	45739.536688	27.934661	3.922204	1.970749
min	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	500.750000	25.000000	74572.000000	28.000000	1.000000	2.000000
50%	1000.500000	48.000000	110045.000000	50.000000	3.000000	4.000000
75%	1500.250000	73.000000	149092.750000	75.000000	7.000000	5.000000
max	2000.000000	99.000000	189974.000000	100.000000	17.000000	9.000000

In [11]:

```
customers.describe(include='object')
```

Out[11]:

	Gender	Profession
count	2000	1965
unique	2	9
top	Female	Artist
freq	1186	612

In [12]:

```
# Info of customers dataframe
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              2000 non-null   int64 
 1   Gender                  2000 non-null   object
 2   Age                     2000 non-null   int64 
 3   Annual Income ($)       2000 non-null   int64 
 4   Spending Score (1-100)  2000 non-null   int64 
 5   Profession              1965 non-null   object
 6   Work Experience         2000 non-null   int64 
 7   Family Size             2000 non-null   int64 
dtypes: int64(6), object(2)
memory usage: 125.1+ KB
```

6 int64 columns and 2 object columns are in the dataframe

In [13]:

```
# Renaming columns
customers.rename(columns={'Annual Income ($)': 'Annual_Income',
                          'Spending Score (1-100)': 'Spending_Score',
                          'Work Experience': 'Work_Experience',
                          'Family Size': 'Family_Size'}, inplace=True)
```

In [14]:

```
# Duplicate value detection
customers[customers.duplicated()]
```

Out[14]:

CustomerID	Gender	Age	Annual_Income	Spending_Score	Profession	Work_Experience

No duplicate values detected

In [15]:

```
# Detecting missing values
customers.isna().sum()
```

Out[15]:

```
CustomerID          0
Gender              0
Age                 0
Annual_Income       0
Spending_Score      0
Profession         35
Work_Experience     0
Family_Size         0
dtype: int64
```

Profession column contains missing values

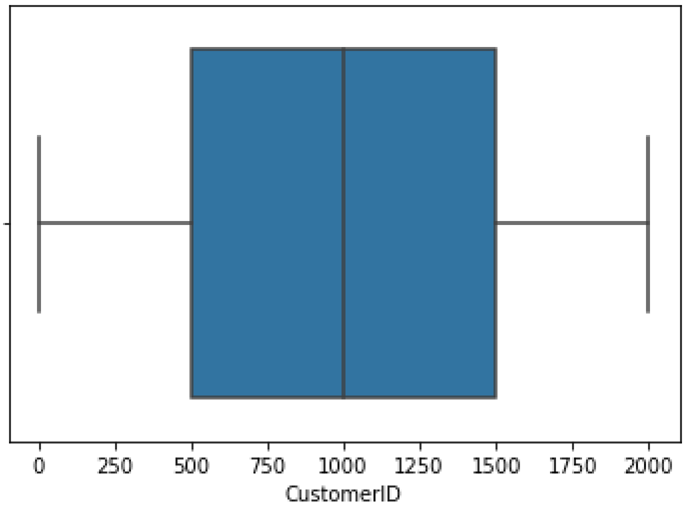
In [16]:

```
# Treating missing values
customers['Profession'].fillna('Unknown_Profession', inplace=True)
```

Checking for outliers with boxplot

In [17]:

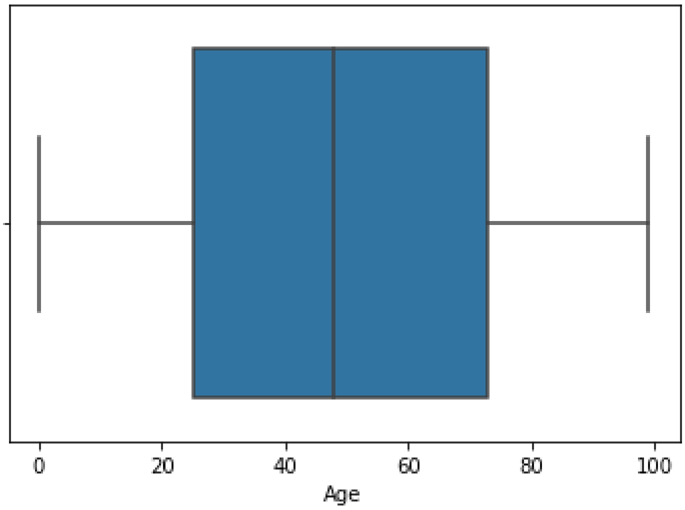
```
# Boxplot of CustomerID column
sns.boxplot(data=customers, x='CustomerID')
plt.show()
```



- No outliers detected in CustomerID column
- Median is at 1000

In [18]:

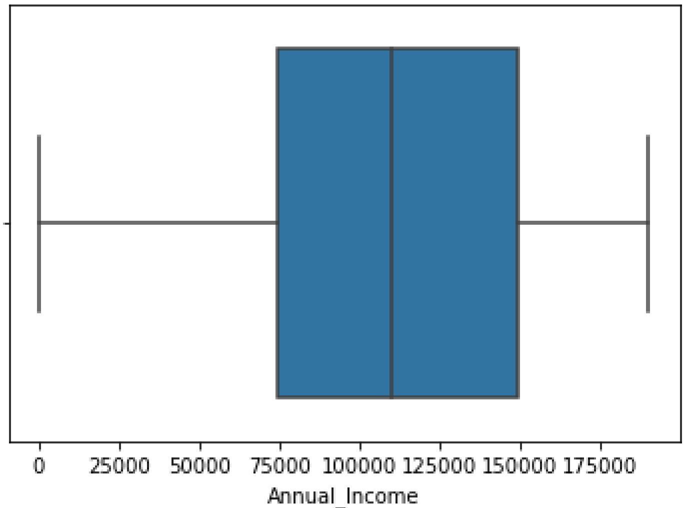
```
# Boxplot of Age column
sns.boxplot(data=customers, x='Age')
plt.show()
```



- No outliers detected in Age column
- Median is between age 40 and 60

In [19]:

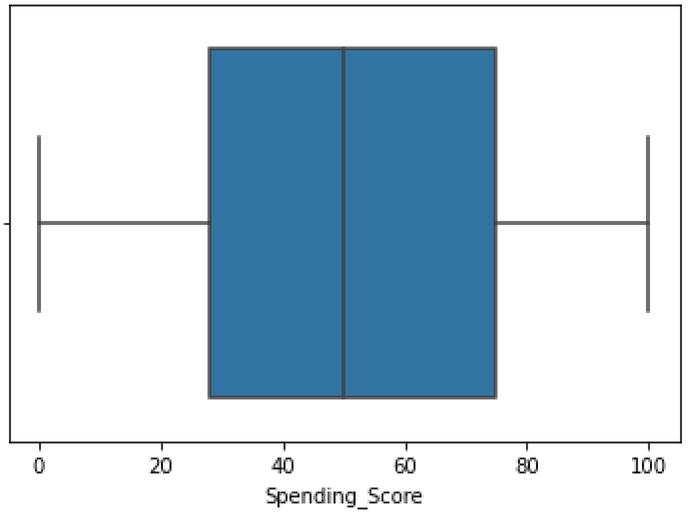
```
# Boxplot of Annual_Income column
sns.boxplot(data=customers, x='Annual_Income')
plt.show()
```



- No outliers detected in Annual_Income column
- Median is between 100000 and 125000

In [20]:

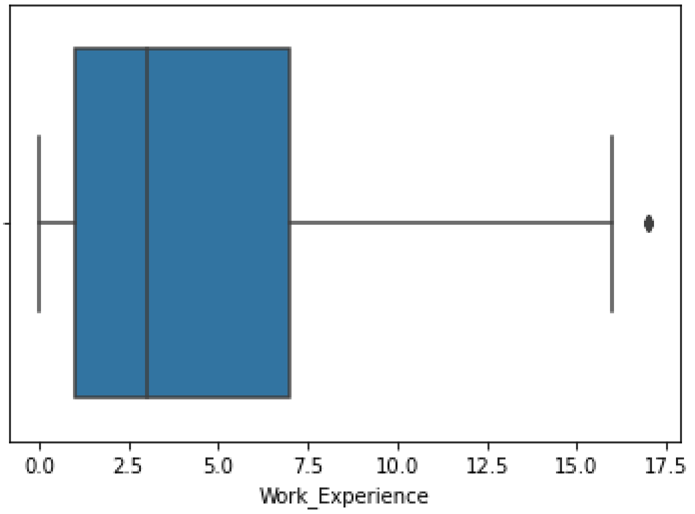
```
# Boxplot of Spending_Score column
sns.boxplot(data=customers, x='Spending_Score')
plt.show()
```



- No outliers detected in Spending_Score column
- Median is between 40 and 60

In [21]:

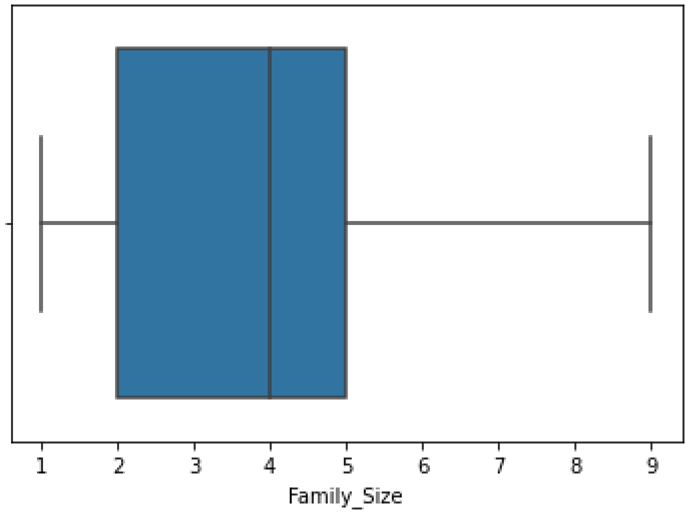
```
# Boxplot of Work_Experience column
sns.boxplot(data=customers, x='Work_Experience')
plt.show()
```



- One outlier detected in Work_Experience column
- Median is between 2.5 and 5.0

In [22]:

```
# Boxplot of Family_Size column
sns.boxplot(data=customers, x='Family_Size')
plt.show()
```



- No outliers detected in Family_Size column
- Median is 4

In [23]:

```
# Treating outlier in the Work_Experience column

quant75 = customers['Work_Experience'].quantile(0.75)
quant25 = customers['Work_Experience'].quantile(0.25)

IQR_WE = quant75 - quant25
lower_lim = quant25 - 1.5 * IQR_WE
upper_lim = quant75 + 1.5 * IQR_WE
customers = customers[(customers['Work_Experience']>lower_lim)&
                      (customers['Work_Experience']<upper_lim)]
```

Exploratory Data Analysis

In [24]:

```
customers.nunique()
```

Out[24]:

```
CustomerID         1990
Gender                2
Age                 100
Annual_Income      1776
Spending_Score      101
Profession           10
Work_Experience      16
Family_Size           9
dtype: int64
```

In [25]:

```
customers['Gender'].unique()
```

Out[25]:

```
array(['Male', 'Female'], dtype=object)
```

In [26]:

```
customers['Age'].unique()
```

Out[26]:

```
array([19, 21, 20, 23, 31, 22, 35, 64, 30, 67, 58, 24, 37, 52, 25, 46, 54,
       29, 45, 40, 60, 53, 18, 49, 42, 36, 65, 48, 50, 27, 33, 59, 47, 51,
       69, 70, 63, 43, 68, 32, 26, 57, 38, 55, 34, 66, 39, 44, 28, 56, 41,
       16, 76, 62, 80,  1,  0, 86, 79, 83, 95, 93, 78, 15,  6, 84,  4, 91,
       14, 92, 77, 89, 12,  7, 94, 96, 74, 85, 73,  9, 10, 11, 17, 90, 61,
       13, 72,  5, 75, 99, 88, 82,  8, 87,  3, 97, 81, 98,  2, 71],
      dtype=int64)
```

In [27]:

```
customers['Spending_Score'].unique()
```

Out[27]:

```
array([ 39,  81,   6,  77,  40,  76,  94,   3,  72,  14,  99,  15,  13,
        79,  35,  66,  29,  98,  73,   5,  82,  32,  61,  31,  87,   4,
        92,  17,  26,  75,  36,  28,  65,  55,  47,  42,  52,  60,  54,
        45,  41,  50,  46,  51,  56,  59,  48,  49,  53,  44,  57,  58,
        43,  91,  95,  11,   9,  34,  71,  88,   7,  10,  93,  12,  97,
        74,  22,  90,  20,  16,  89,   1,  78,  83,  27,  63,  86,  69,
        24,  68,  85,  23,   8,  18,   0,  33,  70,  37,  64,  30,  96,
         2,  38,  21,  84,  62,  80, 100,  67,  19,  25], dtype=int64)
```

In [28]:

```
customers['Profession'].unique()
```

Out[28]:

```
array(['Healthcare', 'Engineer', 'Lawyer', 'Entertainment', 'Artist',
       'Executive', 'Doctor', 'Homemaker', 'Marketing',
       'Unknown_Profession'], dtype=object)
```

In [29]:

```
customers['Work_Experience'].unique()
```

Out[29]:

```
array([ 1,  3,  0,  2,  4,  9, 12, 13,  5,  8, 14,  7,  6, 10, 11, 15],
      dtype=int64)
```

In [30]:

```
customers['Family_Size'].unique()
```

Out[30]:

```
array([4, 3, 1, 2, 6, 5, 8, 7, 9], dtype=int64)
```

In [31]:

```
customers['Gender'].value_counts()
```

Out[31]:

```
Female    1180
Male       810
Name: Gender, dtype: int64
```

In [32]:

```
customers['Age'].value_counts()
```

Out[32]:

```
31    31
52    30
32    30
54    28
63    28
      ..
10    12
77    12
42    12
71    12
98     9
Name: Age, Length: 100, dtype: int64
```

In [33]:

```
customers['Annual_Income'].value_counts()
```

Out[33]:

```
50000     7
9000      7
97000     6
85000     6
4000      6
         ..
155151    1
142723    1
142801    1
131748    1
110610    1
Name: Annual_Income, Length: 1776, dtype: int64
```

In [34]:

```
customers['Spending_Score'].value_counts()
```

Out[34]:

```
49    34
42    33
55    32
17    31
46    28
      ..
72    12
6     12
9     12
95    12
0      2
Name: Spending_Score, Length: 101, dtype: int64
```

In [35]:

```
customers['Profession'].value_counts()
```

Out[35]:

```
Artist                608
Healthcare            338
Entertainment         234
Engineer              178
Doctor                160
Executive             152
Lawyer                140
Marketing              85
Homemaker              60
Unknown_Profession     35
Name: Profession, dtype: int64
```

In [36]:

```
customers['Work_Experience'].value_counts()
```

Out[36]:

```
1     470
0     431
8     166
9     160
7     126
4     121
6     120
5     117
10     84
2      63
3      55
12     17
13     16
14     16
11     14
15     14
Name: Work_Experience, dtype: int64
```

In [37]:

```
customers['Family_Size'].value_counts()
```

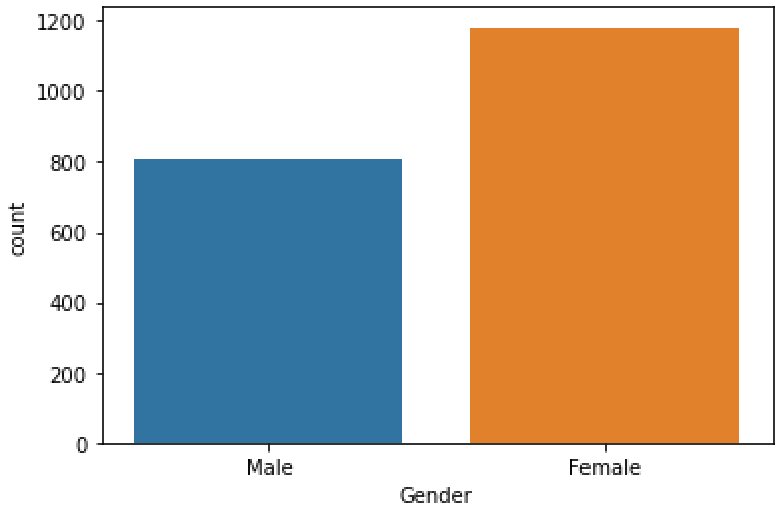
Out[37]:

```
2    360
3    310
1    297
4    287
5    256
6    241
7    234
8      4
9      1
Name: Family_Size, dtype: int64
```

Graphical Analysis of Univariate Variables

In [38]:

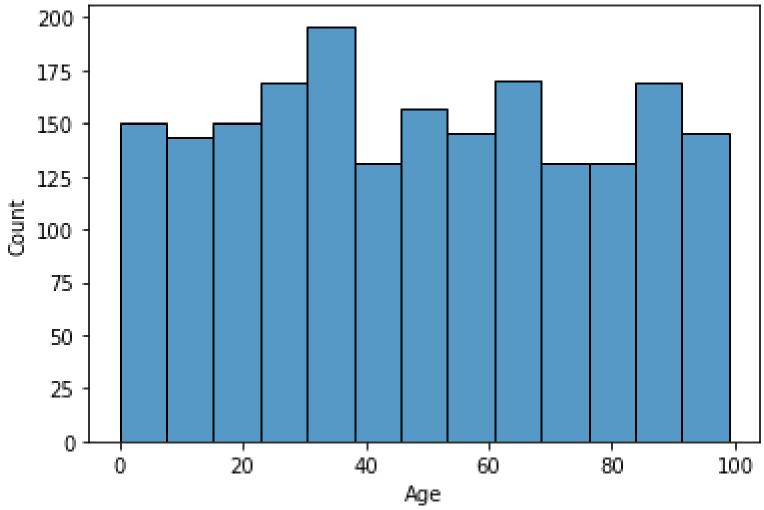
```
# count plot of Gender column
sns.countplot(data=customers, x='Gender')
plt.show()
```



More number of Female customers than Male customers in Gender column

In [39]:

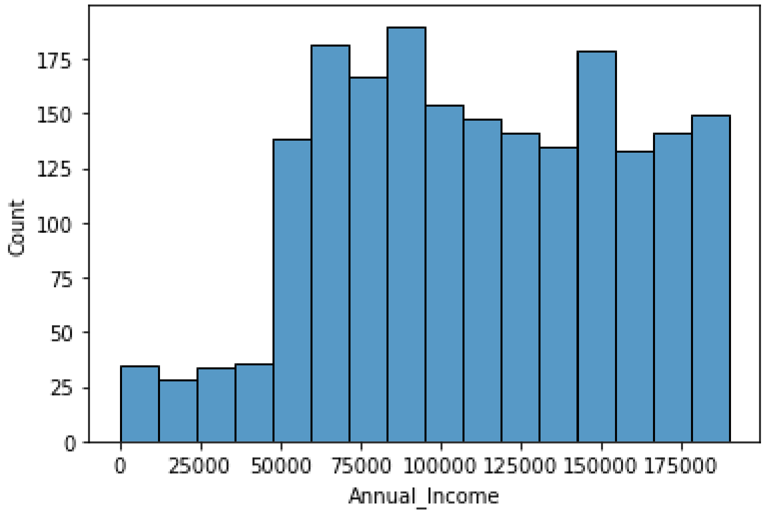
```
# histogram of Age column
sns.histplot(data=customers, x='Age')
plt.show()
```



Histogram starts to increase around 20 and then slightly decreases around 40

In [40]:

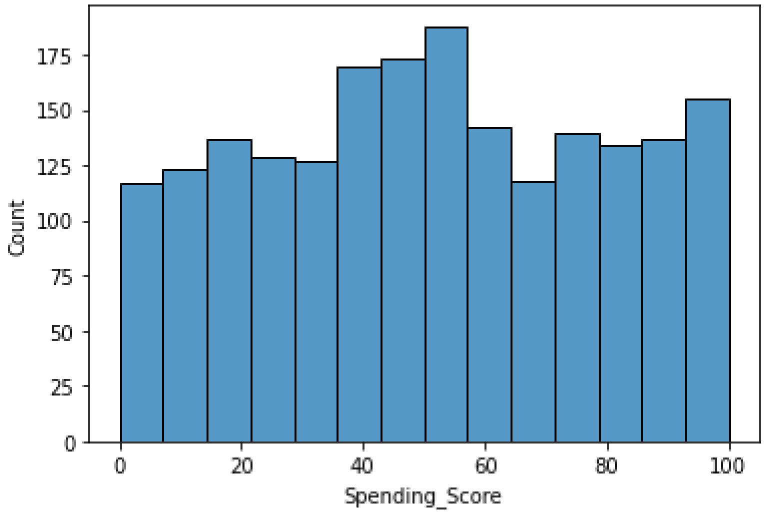
```
# histogram of Annual_Income column
sns.histplot(data=customers, x='Annual_Income')
plt.show()
```



In Annual_Income between 75000-175000 has count greater than 100

In [41]:

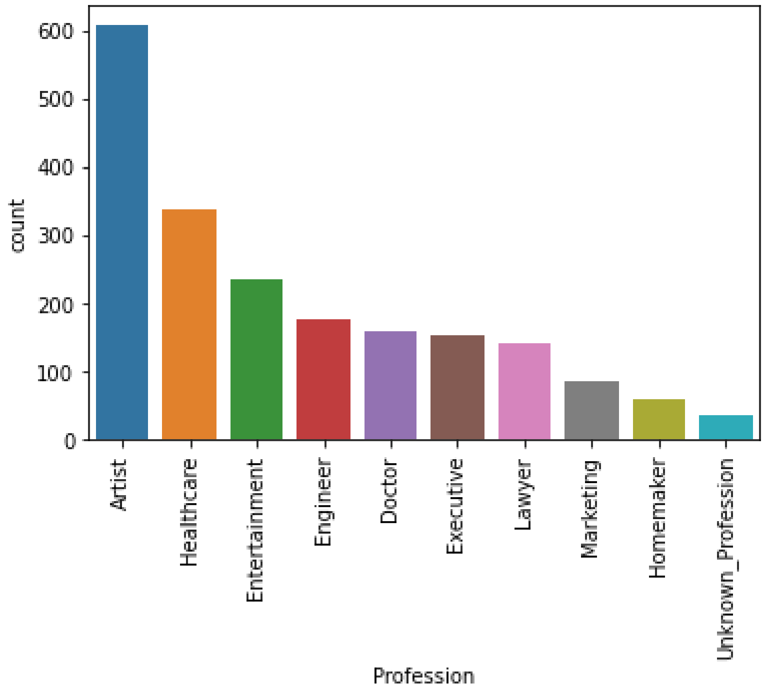
```
# histogram of Spending_Score column
sns.histplot(data=customers, x='Spending_Score')
plt.show()
```



Spending_Score between 40-60 have high count

In [42]:

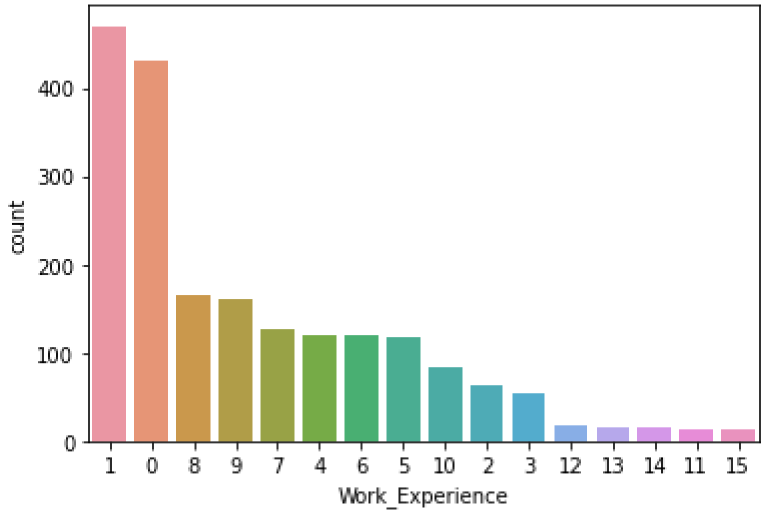
```
# count plot of Profession column
sns.countplot(data=customers, x='Profession',
              order=customers['Profession'].value_counts().index)
plt.xticks(rotation=90)
plt.show()
```



Artist has the highest count, after that Healthcare and Entertainment has the high count

In [43]:

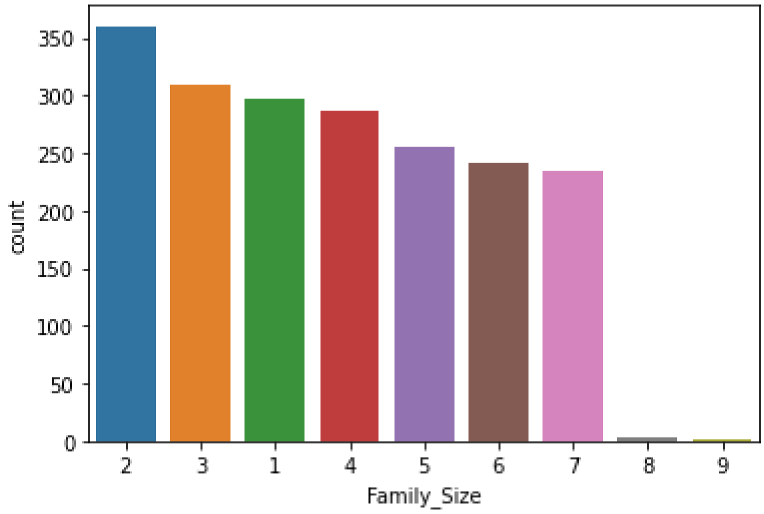
```
# count plot of Work_Experience column
sns.countplot(data=customers, x='Work_Experience',
              order=customers['Work_Experience'].value_counts().index)
plt.show()
```



1 has the highest count in the Work_Experience column

In [44]:

```
# count plot of Family_Size column
sns.countplot(data=customers, x='Family_Size',
              order=customers['Family_Size'].value_counts().index)
plt.show()
```

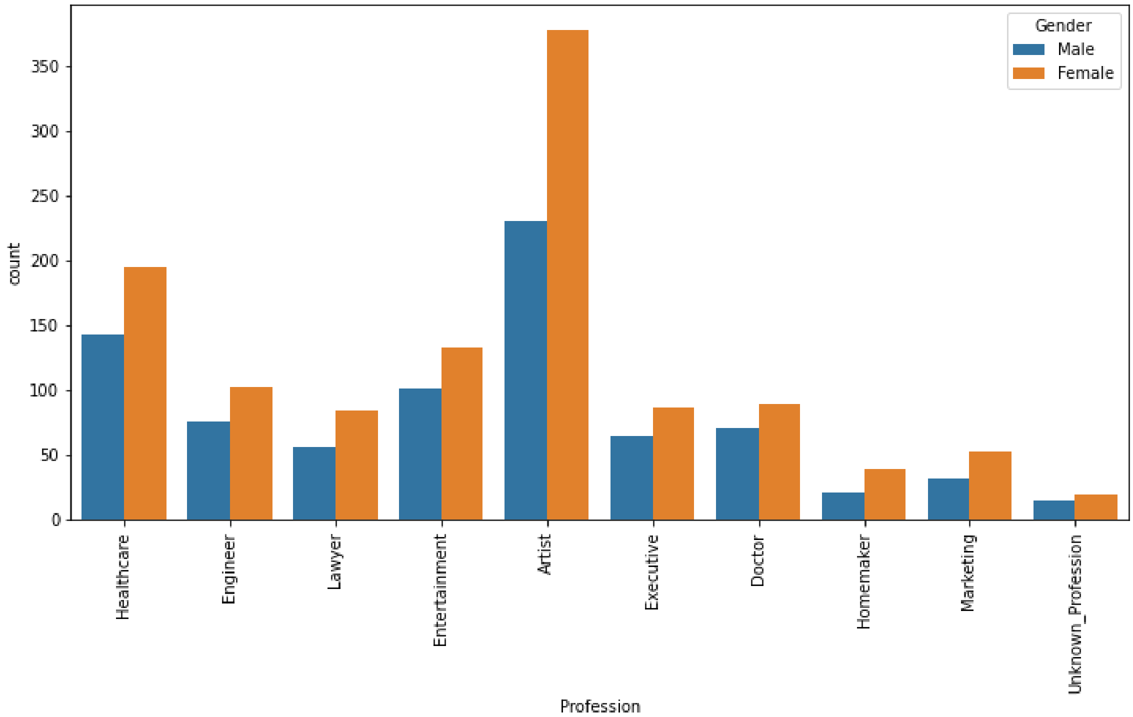


Family_Size of 2 has the highest count and lowest count of 9

Bivariate Analysis of columns from customer dataframe

In [45]:

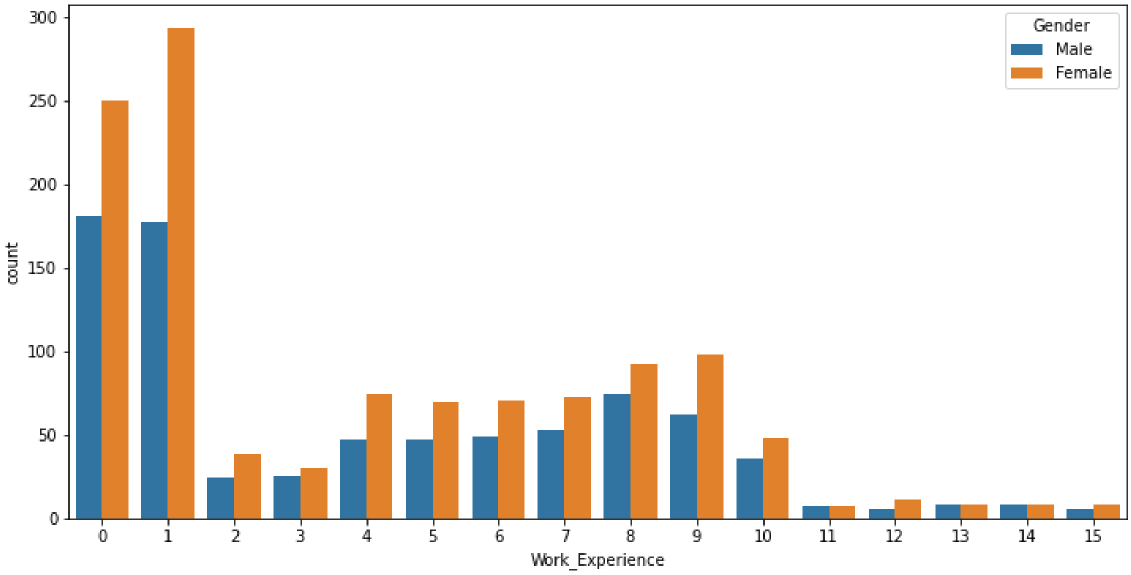
```
# count plot of Profession and Gender column
plt.figure(figsize=(12, 6))
sns.countplot(data=customers, x='Profession', hue='Gender')
plt.xticks(rotation=90)
plt.show()
```



More number of Female than Male in all Profession

In [46]:

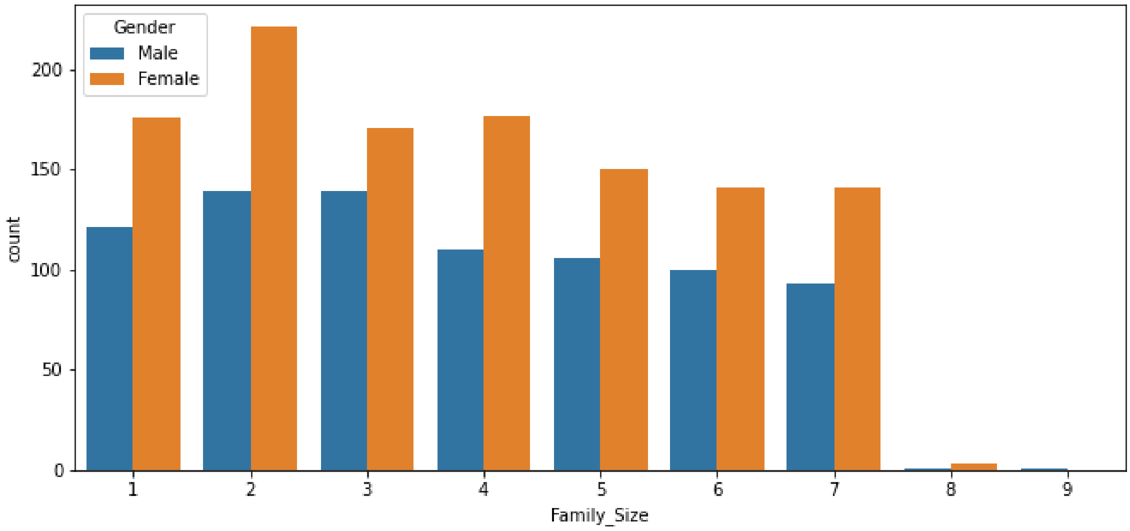
```python
# count plot of Work_Experience and Gender column
plt.figure(figsize=(12, 6))
sns.countplot(data=customers, x='Work_Experience', hue='Gender')
plt.show()
```



0 and 1 has the high count for Work_Experience for both Female and Male

In [47]:

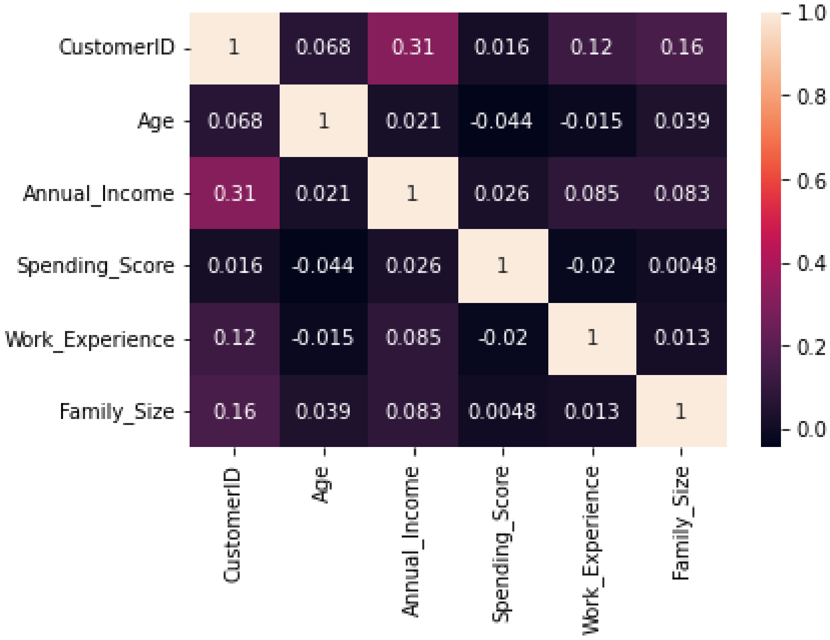
```python
# count plot of Family_Size and Gender column
plt.figure(figsize=(11, 5))
sns.countplot(data=customers, x='Family_Size', hue='Gender')
plt.show()
```



- For Female Customers, Family_Size of 2 has the highest count
- For Male Customers, Family_Size of 2 and 3 has the highest count

In [48]:

```
# Heatmap (correlation with spearman method)
sns.heatmap(customers.corr(method='spearman'), annot=True)
plt.show()
```



Spending_Score and Work_Experience has negative correlation

In [49]:

```
# Creating a function to create bins of Age column
def binage(val1):
    if val1 > 0 and val1 <= 25:
        return '0-25'
    elif val1 > 25 and val1 <= 50:
        return '26-50'
    elif val1 > 50 and val1 <= 75:
        return '51-75'
    elif val1 > 75 and val1 <= 100:
        return '76-100'

# Creating a column with bins
customers['Age_b'] = customers['Age'].apply(binage)
customers[['CustomerID', 'Age_b']]
```

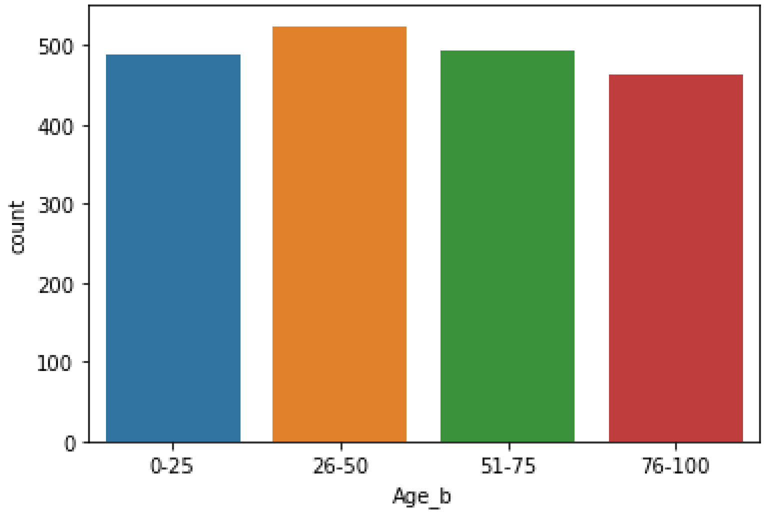
Out[49]:

	CustomerID	Age_b
0	1	0-25
1	2	0-25
2	3	0-25
3	4	0-25
4	5	26-50
...	...	...
1995	1996	51-75
1996	1997	76-100
1997	1998	76-100
1998	1999	76-100
1999	2000	76-100

1990 rows × 2 columns

In [50]:

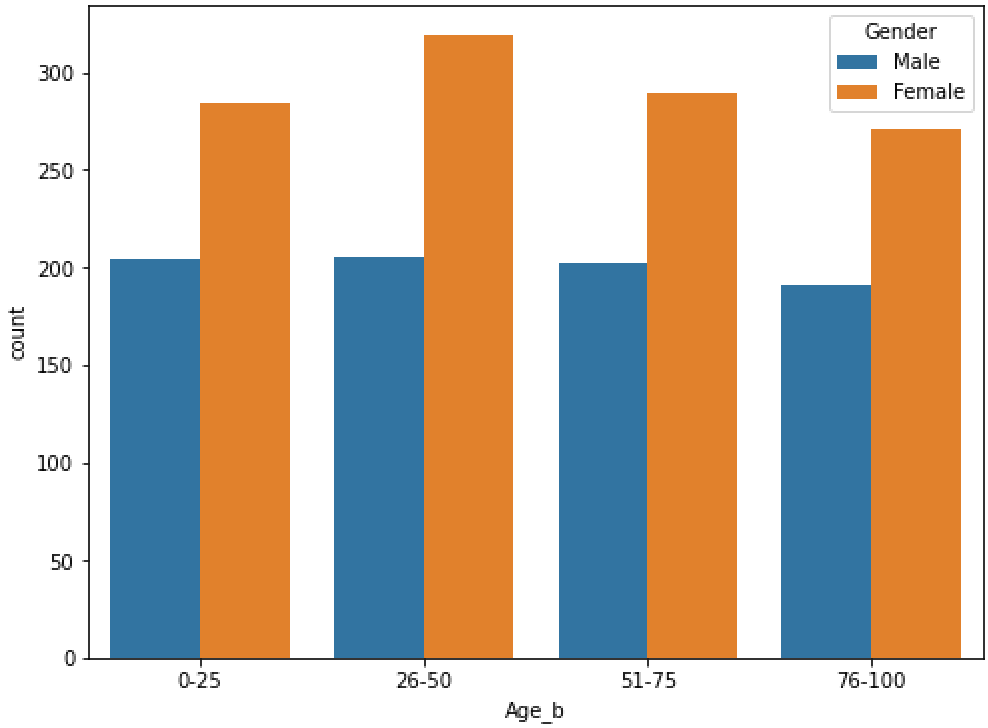
```
# count plot of Age_b column
sns.countplot(data=customers, x='Age_b')
plt.show()
```



Customers with Age between 26-50 has the highest count

In [51]:

```
# count plot of Age_b column, hue is Gender
plt.figure(figsize=(8, 6))
sns.countplot(data=customers, x='Age_b', hue='Gender')
plt.show()
```



Both Female and Male have count greater than 150

In [52]:

```
# Minimum Annual_Income and Maximum Annual_Income
np.min(customers['Annual_Income']), np.max(customers['Annual_Income'])
```

Out[52]:

```
(0, 189974)
```

In [53]:

```
# Creating a function to create bins of Annual_Income column
def binInc(val2):
    if val2 > 0 and val2 <= 50000:
        return '0-50000'
    elif val2 > 50000 and val2 <= 100000:
        return '50001-100000'
    elif val2 > 100000 and val2 <= 150000:
        return '100001-150000'
    elif val2 > 150000 and val2 <= 200000:
        return '150001-200000'


# Creating a column Annual_Income_b
customers['Annual_Income_b'] = customers['Annual_Income'].apply(binInc)
customers[['CustomerID', 'Annual_Income_b']]
```

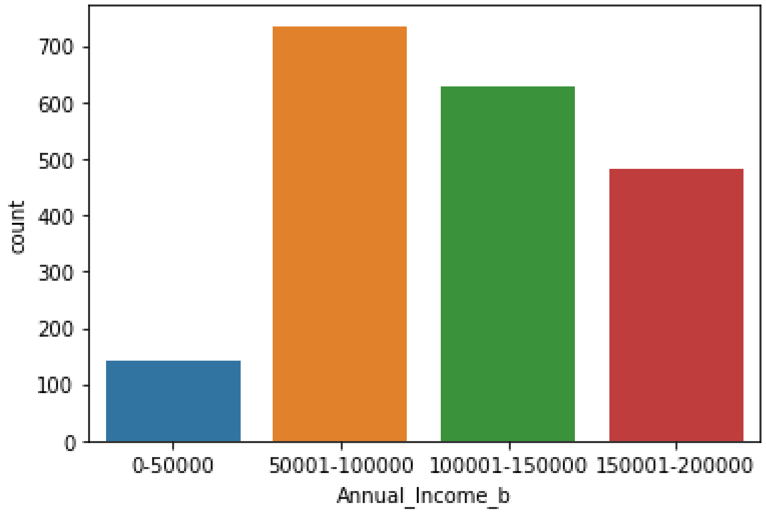
Out[53]:

	CustomerID	Annual_Income_b
0	1	0-50000
1	2	0-50000
2	3	50001-100000
3	4	50001-100000
4	5	0-50000
...	...	...
1995	1996	150001-200000
1996	1997	50001-100000
1997	1998	50001-100000
1998	1999	150001-200000
1999	2000	100001-150000

1990 rows × 2 columns

In [54]:

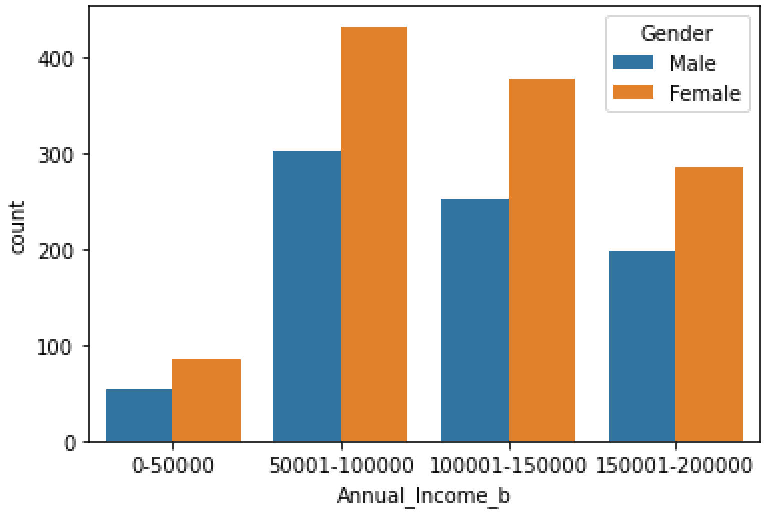
```
# count plot of Annual_Income_b column
sns.countplot(data=customers, x='Annual_Income_b')
plt.show()
```



Annual Income between 50001-100000 has the highest count

In [55]:

```
# count plot of Annual_Income_b and hue is Gender column
sns.countplot(data=customers, x='Annual_Income_b', hue='Gender')
plt.show()
```



Females have high count than Males in every Annual Income ranges

In [56]:

```python
# Creating a function to create bins of Spending_Score column
def binSco(val3):
    if val3 > 0 and val3 <= 25:
        return '0-25'
    elif val3 > 25 and val3 <= 50:
        return '26-50'
    elif val3 > 50 and val3 <= 75:
        return '51-75'
    elif val3 > 75 and val3 <= 100:
        return '76-100'

# Creating a column with Spending_Score_b
customers['Spending_Score_b'] = customers['Spending_Score'].apply(binSco)
customers[['CustomerID', 'Spending_Score_b']]
```

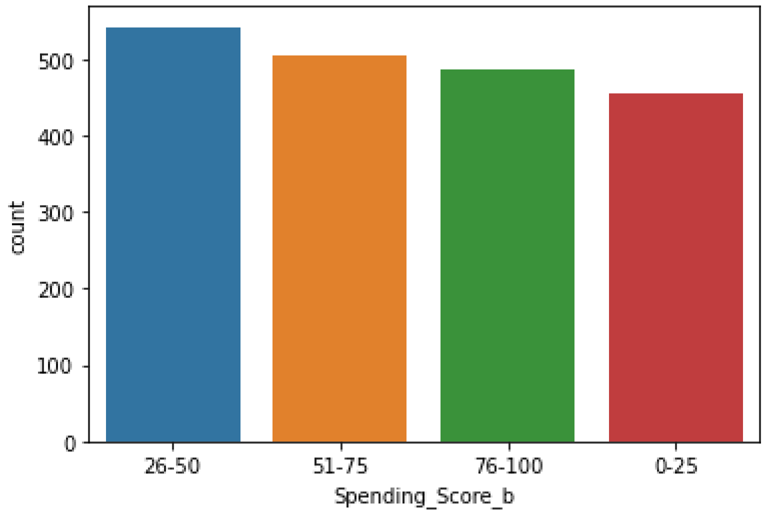
Out[56]:

	CustomerID	Spending_Score_b
0	1	26-50
1	2	76-100
2	3	0-25
3	4	76-100
4	5	26-50
...	...	...
1995	1996	26-50
1996	1997	26-50
1997	1998	0-25
1998	1999	0-25
1999	2000	51-75

1990 rows × 2 columns

In [57]:

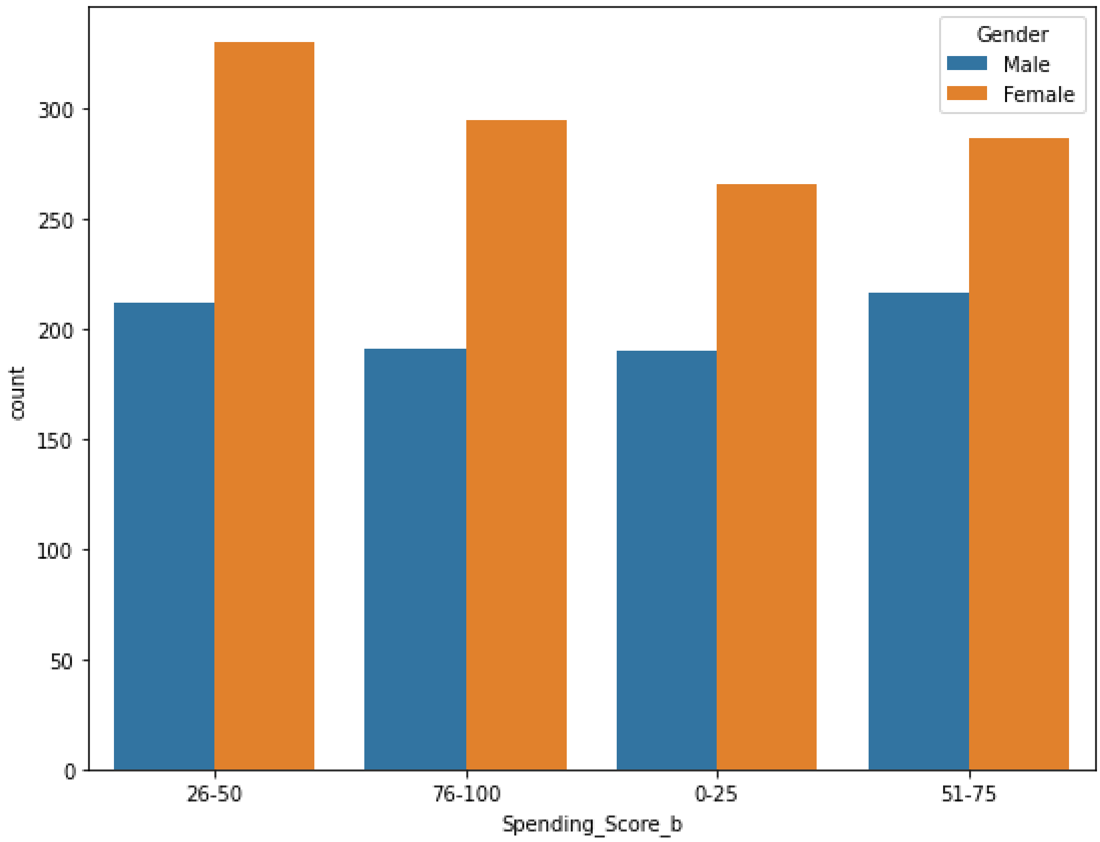
```
# count plot of Spending_Score_b column
sns.countplot(data=customers, x='Spending_Score_b',
              order=customers['Spending_Score_b'].value_counts().index)
plt.show()
```



Customers with Spending score between 26-50 has the highest count

In [58]:

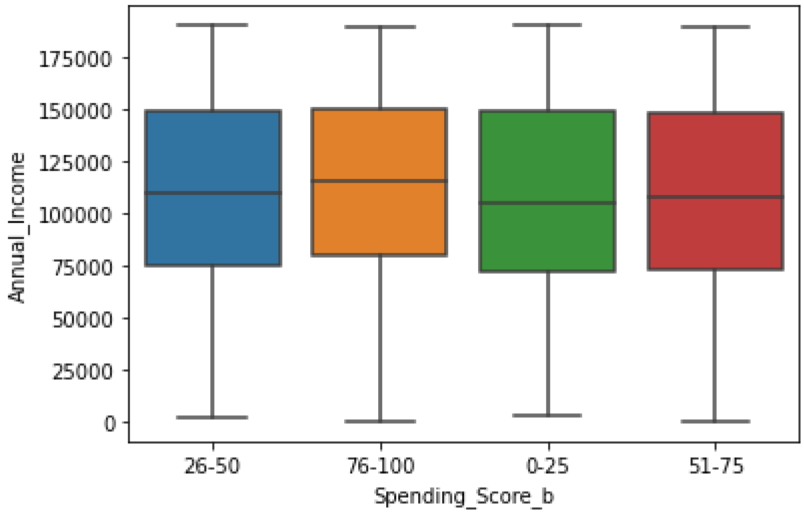
```
# count plot of Spending_Score_b and hue is Gender
plt.figure(figsize=(9, 7))
sns.countplot(data=customers, x='Spending_Score_b', hue='Gender')
plt.show()
```



Male have low count than Female for all Spending Score ranges

In [59]:

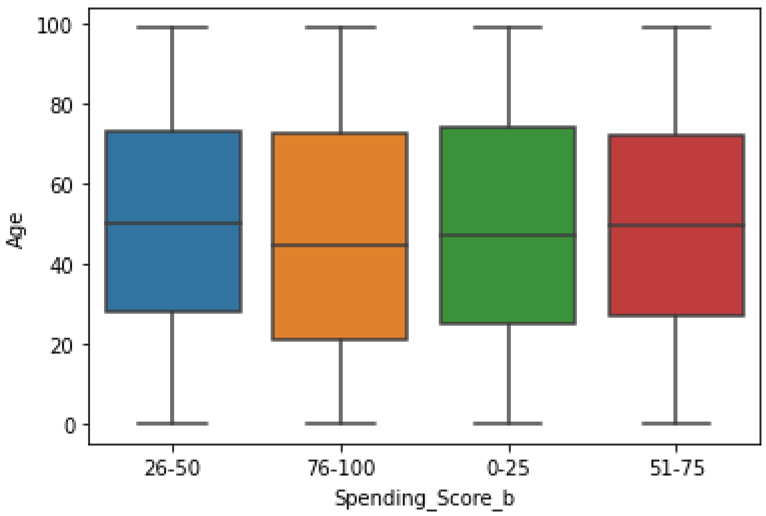
```
# Boxplot of Spending_Score_b and Annual_Income
sns.boxplot(data=customers, x='Spending_Score_b', y='Annual_Income')
plt.show()
```



Spending Score between 76-100 has the highest median Annual_Income

In [60]:

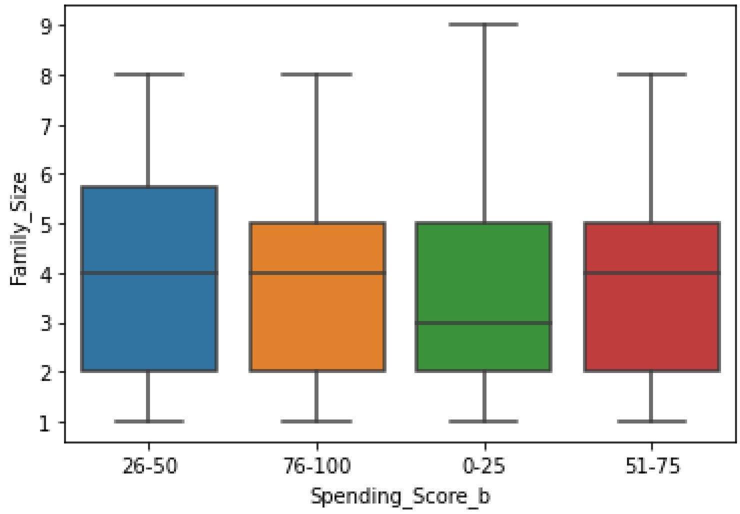
```
# Boxplot of Spending_Score_b and Age columns
sns.boxplot(data=customers, x='Spending_Score_b', y='Age')
plt.show()
```



Spending Score between 76-100 has the lowest median Age

In [61]:

```
# Boxplot of Spending_Score_b and Family_Size columns
sns.boxplot(data=customers, x='Spending_Score_b', y='Family_Size')
plt.show()
```

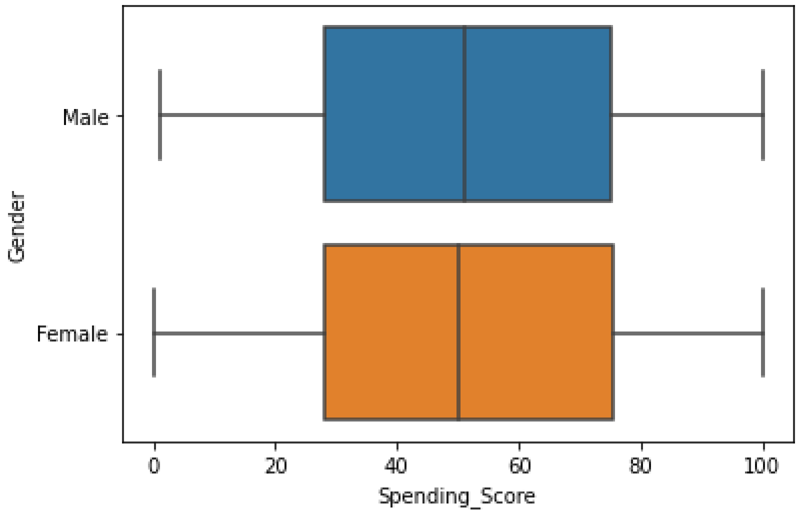


Customer with Spending Score between 0-25 has the lowest median Family_Size, which is 3

Gender and Spending_Score

In [62]:

```
# boxplot of Spending_Score and Gender
sns.boxplot(data=customers, x='Spending_Score', y='Gender')
plt.show()
```



Male and Female shows median between 40 and 60 Spending_Score

Hypothesis Testing : Does Gender has effect on Spending Score

- Null Hypothesis: Gender has no effect on Spending Score
- Alternate Hypothesis: Gender has effect on Spending Score
- Test: T-test

- alpha = 0.05

In [63]:

```
Female_sp_sc = customers[customers['Gender']=='Female']['Spending_Score']
Male_sp_sc =customers[customers['Gender']=='Male']['Spending_Score']
```

In [64]:

```
# Mean of female spending score
np.mean(Female_sp_sc)
```

Out[64]:

```
51.108474576271185
```

In [65]:

```
# Mean of male spending score
np.mean(Male_sp_sc)
```

Out[65]:

```
50.8962962962963
```

In [66]:

```
# Performing T-test
ttest, p_val = ttest_ind(Female_sp_sc, Male_sp_sc)
ttest, p_val
```

Out[66]:

```
(0.1666306488910446, 0.8676776220574317)
```

In [67]:

```
alpha = 0.05

if p_val < alpha:
    print('Reject Null Hypothesis')
    print('p_value:', p_val)
else:
    print('Fail to reject Null Hypothesis')
    print('p_value:', p_val)
```

```
Fail to reject Null Hypothesis
p_value: 0.8676776220574317
```

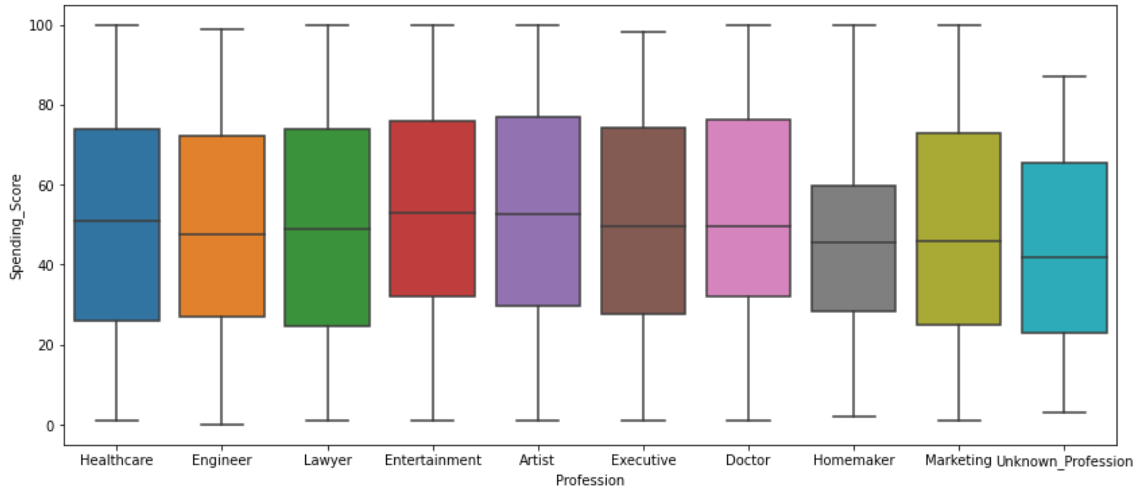
Final output:

- Gender has no effect on Spending Score

Profession and Spending_Score

In [68]:

```
# Boxplot of Profession and Spending_Score columns
plt.figure(figsize=(14, 6))
sns.boxplot(data=customers, x='Profession', y='Spending_Score')
plt.show()
```



All the Profession medians are in between 40-60(Spending_Score)

In [69]:

```
p1 = customers[customers['Profession']=='Healthcare']['Spending_Score']
p2 = customers[customers['Profession']=='Engineer']['Spending_Score']
p3 = customers[customers['Profession']=='Lawyer']['Spending_Score']
p4 = customers[customers['Profession']=='Entertainment']['Spending_Score']
p5 = customers[customers['Profession']=='Artist']['Spending_Score']
p6 = customers[customers['Profession']=='Executive']['Spending_Score']
p7 = customers[customers['Profession']=='Doctor']['Spending_Score']
p8 = customers[customers['Profession']=='Homemaker']['Spending_Score']
p9 = customers[customers['Profession']=='Marketing']['Spending_Score']
p10 = customers[customers['Profession']=='Unknown_Profession']['Spending_Score']
```

1. Shapiro test to check Normality

In [70]:

```
Spending_Score_samp = customers['Spending_Score'].sample(150)
```

Hypothesis Testing : Does Spending Score has Gaussian distribution

- Null Hypothesis: Spending Score is Gaussian
- Alternate Hypothesis: Spending Score is not Gaussian
- Test: Shapiro
- alpha = 0.05

In [71]:

```
# Performing shapiro test
test_stat, p_val = shapiro(Spending_Score_samp)
test_stat, p_val
```

Out[71]:

```
(0.9652606248855591, 0.0007712905644439161)
```

In [72]:

```
alpha = 0.05

if p_val < alpha:
    print('Reject Null Hypothesis')
    print('p_value:', p_val)
else:
    print('Fail to reject Null Hypothesis')
    print('p_value:', p_val)
```

```
Reject Null Hypothesis
p_value: 0.0007712905644439161
```

Final output:

- Spending score is not Gaussian

Spending score is not Gaussian. Assuption of ANNOVA doesn't hold, hence we will use kruskal instead of ANNOVA

Hypothesis Testing : Does Profession has effect on Spending Score

- Null Hypothesis: Profession has no effect on Spending Score
- Alternate Hypothesis: Profession has effect on Spending Score
- Test: kruskal
- alpha = 0.05

In [73]:

```
# Performing kruskal
test_stat, p_val = kruskal(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10)
test_stat, p_val
```

Out[73]:

```
(9.698200264859038, 0.37546551148583535)
```

In [74]:

```
alpha = 0.05

if p_val < alpha:
    print('Reject Null Hypothesis')
    print('p_value:', p_val)
else:
    print('Fail to reject Null Hypothesis')
    print('p_value:', p_val)
```

```
Fail to reject Null Hypothesis
p_value: 0.37546551148583535
```

Final output:

- Profession has no effect on Spending Score

Annual_Income_b and Spending_Score_b

Hypothesis Testing : Does Annual_Income_b has effect on Spending_Score_b

- Null Hypothesis: Annual_Income_b is not dependent on Spending_Score_b
- Alternate Hypothesis: Annual_Income_b is dependent on Spending_Score_b
- Test: chi2_contingency
- alpha = 0.05

In [75]:

```
SpSc_AnIn_b = pd.crosstab(index= customers['Spending_Score_b'],
                          columns= customers['Annual_Income_b'])
SpSc_AnIn_b
```

Out[75]:

Annual_Income_b	0-50000	100001-150000	150001-200000	50001-100000
Spending_Score_b				
0-25	31	133	110	182
26-50	40	171	132	199
51-75	40	156	121	186
76-100	30	169	120	166

In [76]:

```
chi_stat, p_val, df, expe_freq = chi2_contingency(SpSc_AnIn_b)
alpha = 0.05

if p_val < alpha:
    print('Reject Null Hypothesis')
    print('p_value:', p_val)
else:
    print('Fail to reject Null Hypothesis')
    print('p_value:', p_val)
```

```
Fail to reject Null Hypothesis
p_value: 0.7553476561132749
```

Final output:

- Annual_Income_b is not dependent on Spending_Score_b

Insights

- More customers are Female
- Artist has the highest count than other profession
- More number of customers with work experience of 1 year
- Most number of customers have family size of 2
- More number of customers are in between age range 26-50
- There are more customers with annual income between 50001-100000 dollars
- Gender has no effect on Spending Score
- Profession has no effect on Spending Score
- The Annual_Income_b is not dependent on Spending_Score_b

Recommendations

- Give 20 percent discount to customers with spending score greater than 80
- For customers with spending score between 60-80, give them discount of 10 percent
- Take a survey of customers with low spending score about what are their requirements
- Give a coupon which might encourage to customers with less spending score
- Shop can give member customers on their profession's day at least 5 percent discount
- Considering more customers are Female more variety of female related products to be added to the shop.
- For Male customers, shop can take a survey on what kind of products they are looking for