CUSTOMER SEGMENTATION

#USING K-MEANS CLUSTERING

```
Name: Tejas Saiprasad Havaldar
Purpose: Data Science Internship
```

In [2]:
```
import pandas as pd # Pandas (version :1.3.1 )
import numpy as np # Numpy (version :1.21.1 )
import matplotlib.pyplot as plt # Matplotlib (version :3.4.2  )
from sklearn.cluster import KMeans # Scikit Learn (version :0.24.2 )
import seaborn as sns # Seaborn (version :0.11.1 )
plt.style.use('seaborn')
```

In [3]:
```
data = pd.read_csv('Mall_Customers.csv')
```

In [4]:
```
data
```

Out[4]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

In [5]:
```
data.head()
```

Out[5]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [6]:
```
data.tail()
```

Out[6]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

In [7]:
```
len(data)
```

Out[7]:
```
200
```

In [8]:
```
data.shape
```

Out[8]:
```
(200, 5)
```

In [9]:
```
data.columns
```

Out[9]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [10]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [11]:
```
data.dtypes
```

Out[11]:
```
CustomerID                int64
Gender                   object
Age                       int64
Annual Income (k$)        int64
Spending Score (1-100)    int64
dtype: object
```

In [12]:
```
data.describe()
```

Out[12]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

In [13]:
```
data.isnull()
```

Out[13]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
195	False	False	False	False	False
196	False	False	False	False	False
197	False	False	False	False	False
198	False	False	False	False	False
199	False	False	False	False	False

200 rows × 5 columns

In [14]:
```
data.isnull().sum()
```

Out[14]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [15]:
```
#The 'customer_id' column has no relevence therefore deleting it would be better.
#Deleting 'customer_id' columnn using drop().
data = data.drop('CustomerID', axis=1)
data.head()
```

Out[15]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40

In [16]:
```
#The 'Annual income' and 'Spending score' columns have spaces in their column names, we need to rename them.
#Cleaning the data labels (Annual income and Spending Score) using rename().
data = data.rename(columns={'Annual Income (k$)':'Annual_Income','Spending Score (1-100)':'Spending_Score'})
data.head()
```

Out[16]:

	Gender	Age	Annual_Income	Spending_Score
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40

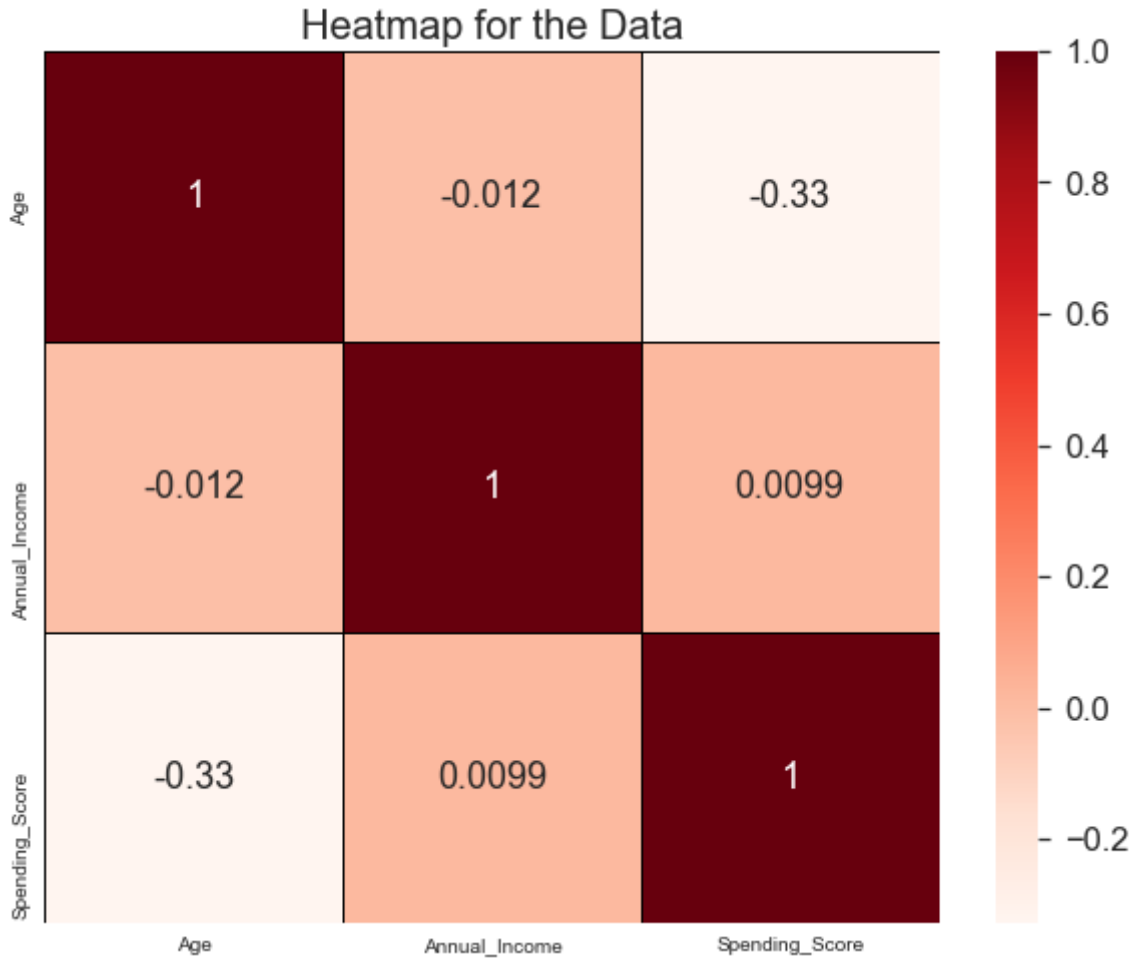
In [17]:
```
#Understanding and Visualizing Data
```

In [18]:
```
# Finding and viewing Corelations in the data and columns using corr().
corr = data.corr()
corr
```

Out[18]:

	Age	Annual_Income	Spending_Score
Age	1.000000	-0.012398	-0.327227
Annual_Income	-0.012398	1.000000	0.009903
Spending_Score	-0.327227	0.009903	1.000000

In [19]:
```
# Plotting the heatmap
fig, ax = plt.subplots(figsize=(10,8))
sns.set(font_scale=1.5)
ax = sns.heatmap(corr, cmap = 'Reds', annot = True, linewidths=0.5, linecolor='black')
plt.title('Heatmap for the Data', fontsize = 20)
plt.show()
```



In [ ]:
```
# Gender Data VisualizationGender Data Visualization
```

In [20]:
```
data['Gender'].head()
```

Out[20]:
```
0      Male
1      Male
2    Female
3    Female
4    Female
Name: Gender, dtype: object
```

In [21]:
```
data['Gender'].dtype
```

Out[21]:
```
dtype('O')
```

In [22]:
```
data['Gender'].unique()
```

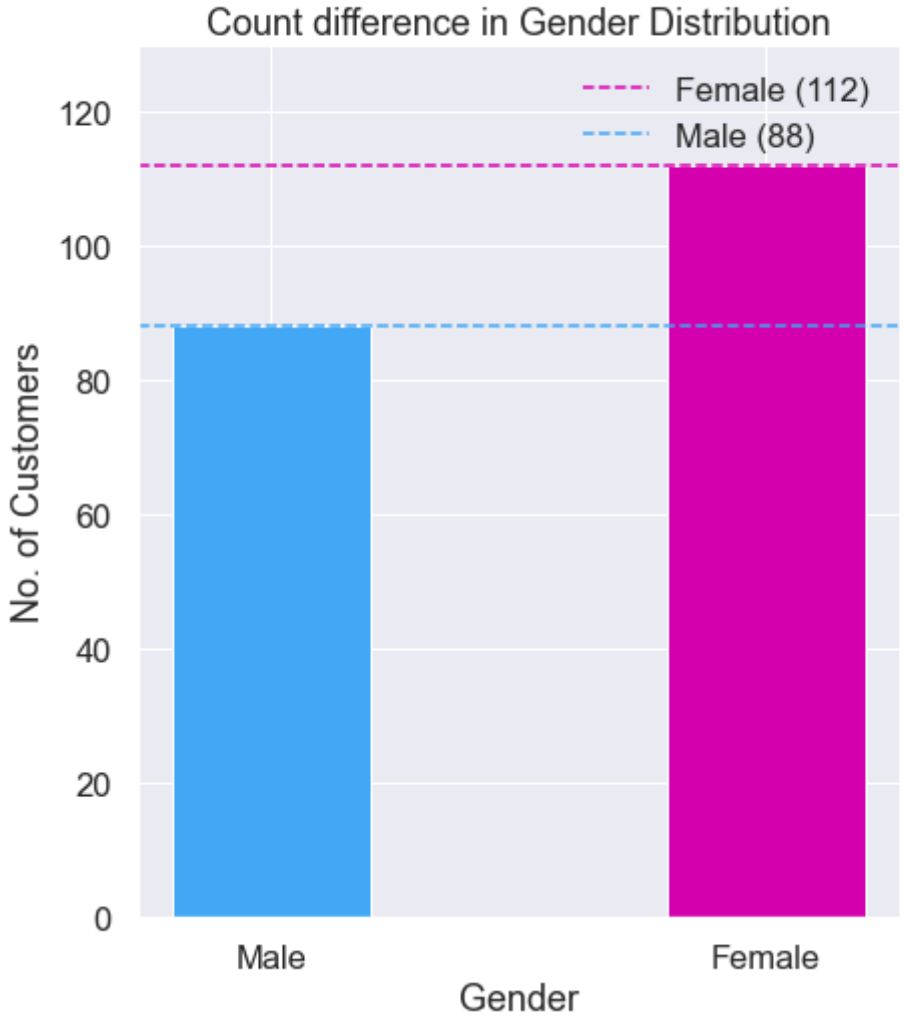
Out[22]:
```
array(['Male', 'Female'], dtype=object)
```

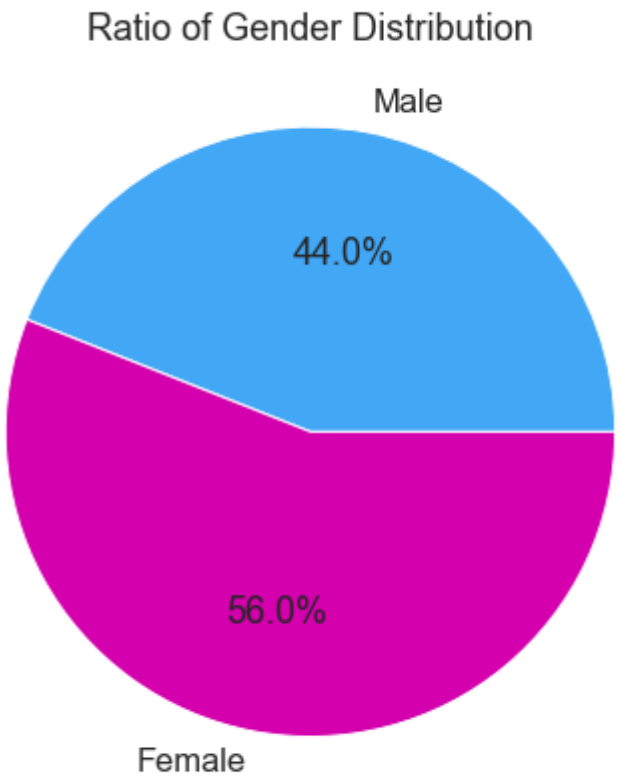
In [23]:
```
data['Gender'].value_counts()
```

Out[23]:
```
Female    112
Male       88
Name: Gender, dtype: int64
```

In [24]:
```
# Plotting Gender Distribution on Bar graph and the ratio of distribution using Pie Chart.
labels=data['Gender'].unique()
values=data['Gender'].value_counts(ascending=True)


fig, (ax0,ax1) = plt.subplots(ncols=2,figsize=(15,8))
bar = ax0.bar(x=labels, height=values, width=0.4, align='center', color=['#42a7f5','#d400ad'])
ax0.set(title='Count difference in Gender Distribution',xlabel='Gender', ylabel='No. of Customers')
ax0.set_ylim(0,130)
ax0.axhline(y=data['Gender'].value_counts()[0], color='#d400ad', linestyle='--', label=f'Female ({data.Gender.value_coun
ax0.axhline(y=data['Gender'].value_counts()[1], color='#42a7f5', linestyle='--', label=f'Male ({data.Gender.value_counts
ax0.legend()


ax1.pie(values,labels=labels,colors=['#42a7f5','#d400ad'],autopct='%1.1f%%')
ax1.set(title='Ratio of Gender Distribution')
fig.suptitle('Gender Distribution', fontsize=30);
plt.show()
```

Gender Distribution





In [ ]:
```
# Age Data Visualization
```

In [25]:
```
data['Age'].head()
```

Out[25]:
```
0    19
1    21
2    20
3    23
4    31
Name: Age, dtype: int64
```

In [26]:
```
data['Age'].dtype
```

Out[26]:
```
dtype('int64')
```

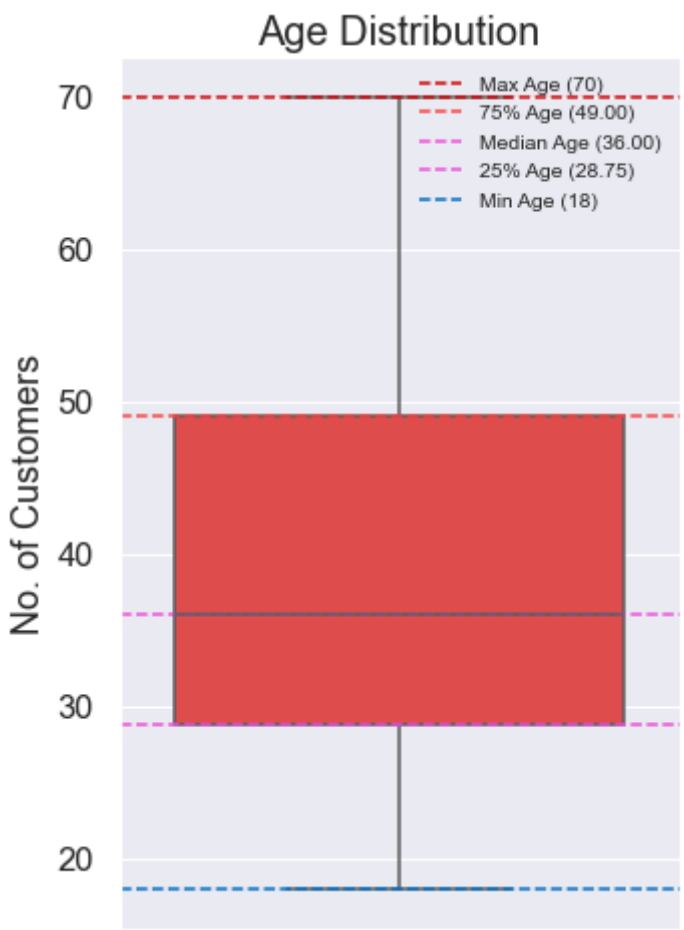
In [27]:
```
data['Age'].unique()
```

Out[27]:
```
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],
      dtype=int64)
```

In [28]:
```
data['Age'].describe()
```

Out[28]:
```
count    200.000000
mean      38.850000
std       13.969007
min       18.000000
25%       28.750000
50%       36.000000
75%       49.000000
max       70.000000
Name: Age, dtype: float64
```

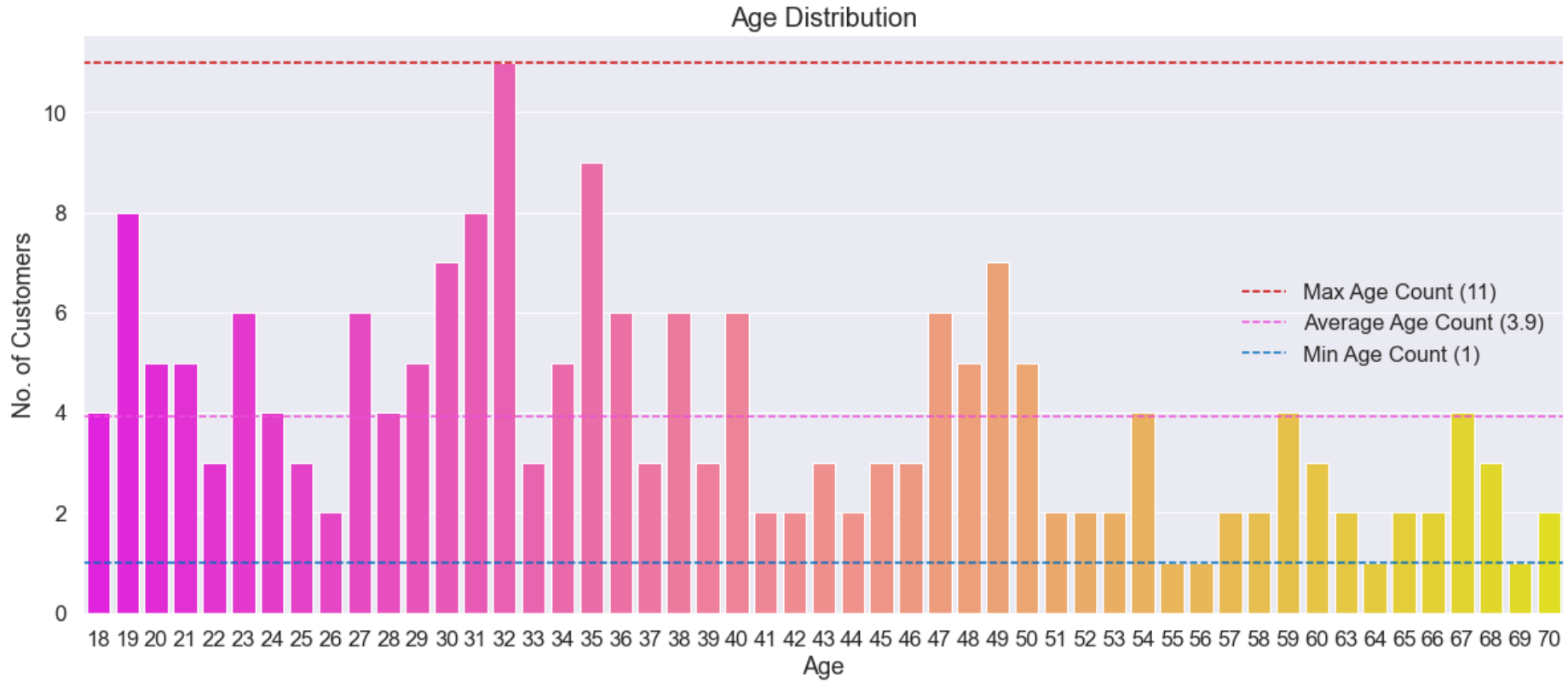
In [29]:
```
# Visualizing Statistical Description of the Age on a boxplot.
fig, ax = plt.subplots(figsize=(5,8))
sns.set(font_scale=1.5)
ax = sns.boxplot(y=data["Age"], color="#f73434")
ax.axhline(y=data['Age'].max(), linestyle='--',color='#c90404', label=f'Max Age ({data.Age.max()})')
ax.axhline(y=data['Age'].describe()[6], linestyle='--',color='#f74343', label=f'75% Age ({data.Age.describe()[6]:.2f})')
ax.axhline(y=data['Age'].median(), linestyle='--',color='#eb50db', label=f'Median Age ({data.Age.median():.2f})')
ax.axhline(y=data['Age'].describe()[4], linestyle='--',color='#eb50db', label=f'25% Age ({data.Age.describe()[4]:.2f})')
ax.axhline(y=data['Age'].min(), linestyle='--',color='#046ebf', label=f'Min Age ({data.Age.min()})')
ax.legend(fontsize='xx-small', loc='upper right')
ax.set_ylabel('No. of Customers')

plt.title('Age Distribution', fontsize = 20)
plt.show()
```



In [30]:
```
data['Age'].value_counts().head()
```

Out[30]:
```
32    11
35     9
19     8
31     8
30     7
Name: Age, dtype: int64
```

In [31]:
```
fig, ax = plt.subplots(figsize=(20,8))
sns.set(font_scale=1.5)
ax = sns.countplot(x=data['Age'], palette='spring')
ax.axhline(y=data['Age'].value_counts().max(), linestyle='--',color='#c90404', label=f'Max Age Count ({data.Age.value_co
ax.axhline(y=data['Age'].value_counts().mean(), linestyle='--',color='#eb50db', label=f'Average Age Count ({data.Age.val
ax.axhline(y=data['Age'].value_counts().min(), linestyle='--',color='#046ebf', label=f'Min Age Count ({data.Age.value_co
ax.legend(loc ='right')
ax.set_ylabel('No. of Customers')

plt.title('Age Distribution', fontsize = 20)
plt.show()
```



In [ ]:
```
# Gender wise Age Distribution
```

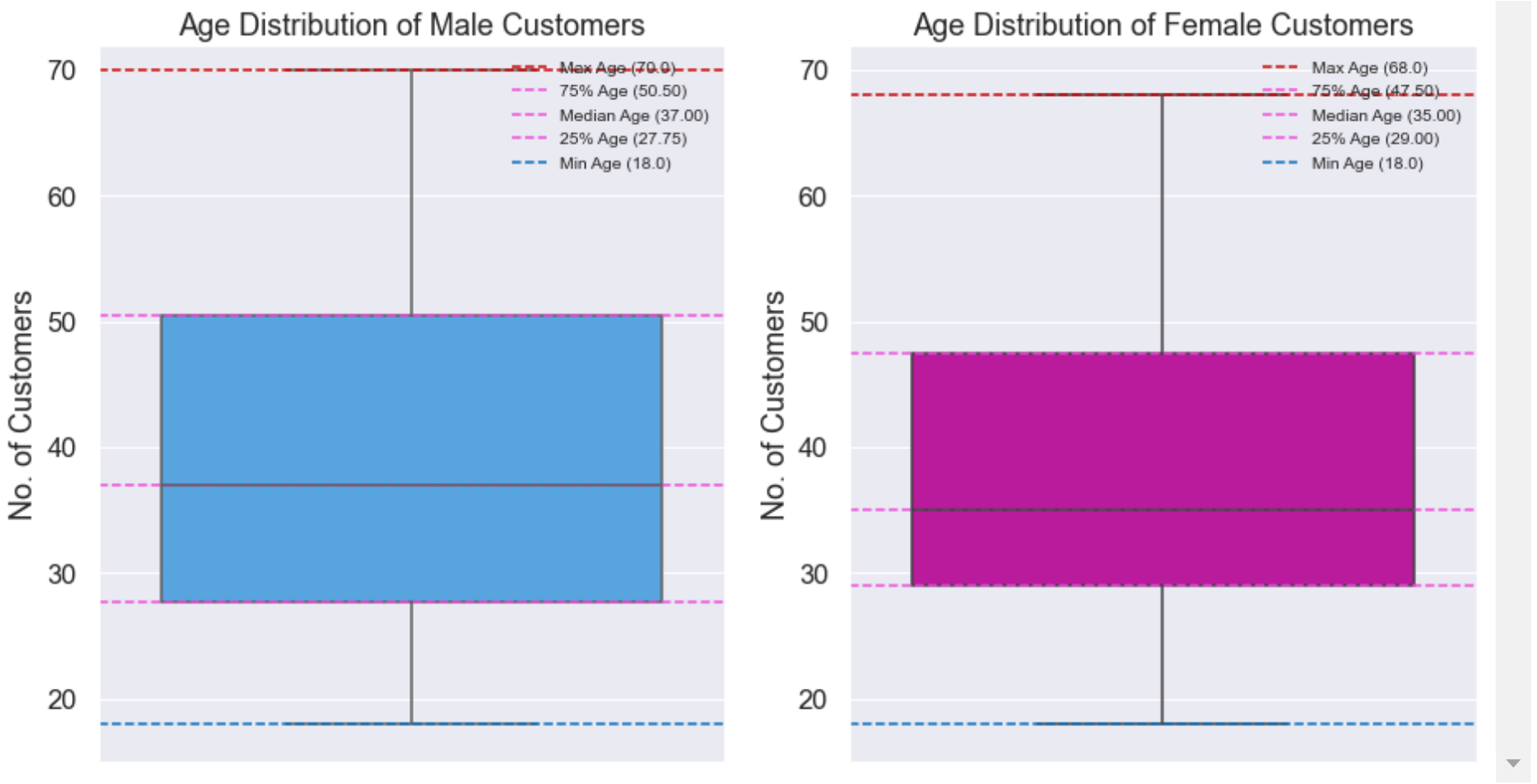
In [32]:
```
data[data['Gender']=='Male']['Age'].describe()
```

Out[32]:
```
count    88.000000
mean     39.806818
std      15.514812
min      18.000000
25%      27.750000
50%      37.000000
75%      50.500000
max      70.000000
Name: Age, dtype: float64
```

In [33]:
```
# Statistical Age Distribution of female customers.
data[data['Gender']=='Female']['Age'].describe()
```

Out[33]:
```
count    112.000000
mean      38.098214
std       12.644095
min       18.000000
25%       29.000000
50%       35.000000
75%       47.500000
max       68.000000
Name: Age, dtype: float64
```

In [34]:
```
# Visualizing Gender wise Age Distribution of Male and Female customers on a boxplot.
data_male = data[data['Gender']=='Male']['Age'].describe()
data_female = data[data['Gender']=='Female']['Age'].describe()

fig, (ax0,ax1) = plt.subplots(ncols=2,figsize=(15,8))
sns.set(font_scale=1.5)
sns.boxplot(y=data[data['Gender']=='Male']['Age'], color="#42a7f5", ax=ax0)
ax0.axhline(y=data['Age'].max(), linestyle='--',color='#c90404', label=f'Max Age ({data_male[7]})')
ax0.axhline(y=data_male[6], linestyle='--',color='#eb50db', label=f'75% Age ({data_male[6]:.2f})')
ax0.axhline(y=data_male[5], linestyle='--',color='#eb50db', label=f'Median Age ({data_male[5]:.2f})')
ax0.axhline(y=data_male[4], linestyle='--',color='#eb50db', label=f'25% Age ({data_male[4]:.2f})')
ax0.axhline(y=data_male[3], linestyle='--',color='#046ebf', label=f'Min Age ({data_male[3]})')
ax0.legend(fontsize='xx-small', loc='upper right')
ax0.set(ylabel='No. of Customers', title='Age Distribution of Male Customers')
ax0.set_ylim(15,72)

ax1 = sns.boxplot(y=data[data['Gender']=='Female']['Age'], color="#d400ad", ax=ax1)
ax1.axhline(y=data_female[7], linestyle='--',color='#c90404', label=f'Max Age ({data_female[7]})')
ax1.axhline(y=data_female[6], linestyle='--',color='#eb50db', label=f'75% Age ({data_female[6]:.2f})')
ax1.axhline(y=data_female[5], linestyle='--',color='#eb50db', label=f'Median Age ({data_female[5]:.2f})')
ax1.axhline(y=data_female[4], linestyle='--',color='#eb50db', label=f'25% Age ({data_female[4]:.2f})')
ax1.axhline(y=data_female[3], linestyle='--',color='#046ebf', label=f'Min Age ({data_female[3]})')
ax1.legend(fontsize='xx-small', loc='upper right')
ax1.set(ylabel='No. of Customers', title='Age Distribution of Female Customers')
ax1.set_ylim(15,72)

plt.show()
```



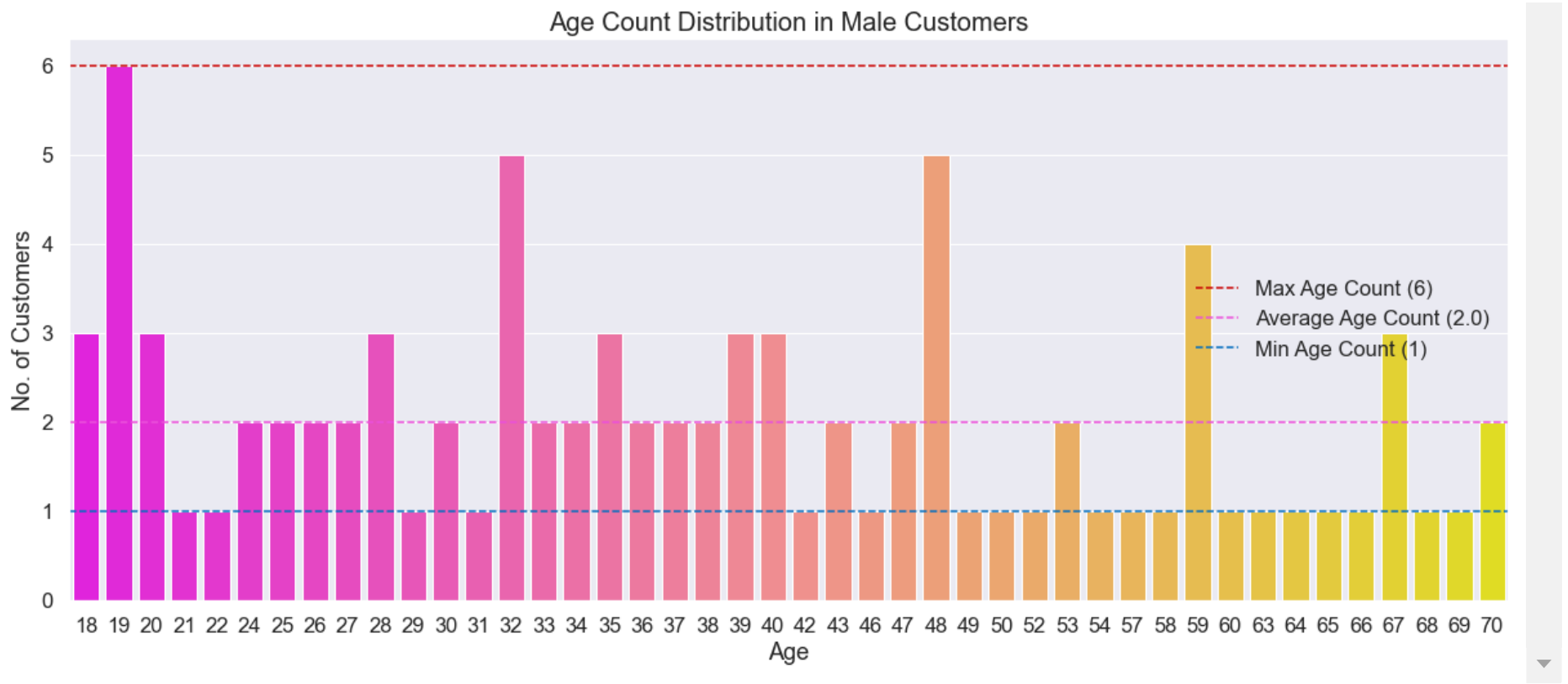
In [35]:
```
# Average Age of Male Customers.
data[data['Gender']=='Male'].Age.mean()
```

Out[35]:
```
39.80681818181818
```

In [36]:
```
data[data['Gender']=='Male'].Age.value_counts().head()
```

Out[36]:
```
19    6
32    5
48    5
59    4
28    3
Name: Age, dtype: int64
```

In [37]:
```
# Visualizing distribution of age count in Male customers using a countplot.
maxi = data[data['Gender']=='Male'].Age.value_counts().max()
mean = data[data['Gender']=='Male'].Age.value_counts().mean()
mini = data[data['Gender']=='Male'].Age.value_counts().min()


fig, ax = plt.subplots(figsize=(20,8))
sns.set(font_scale=1.5)
ax = sns.countplot(x=data[data['Gender']=='Male'].Age, palette='spring')

ax.axhline(y=maxi, linestyle='--',color='#c90404', label=f'Max Age Count ({maxi})')
ax.axhline(y=mean, linestyle='--',color='#eb50db', label=f'Average Age Count ({mean:.1f})')
ax.axhline(y=mini, linestyle='--',color='#046ebf', label=f'Min Age Count ({mini})')
ax.set_ylabel('No. of Customers')

ax.legend(loc ='right')

plt.title('Age Count Distribution in Male Customers', fontsize = 20)
plt.show()
```



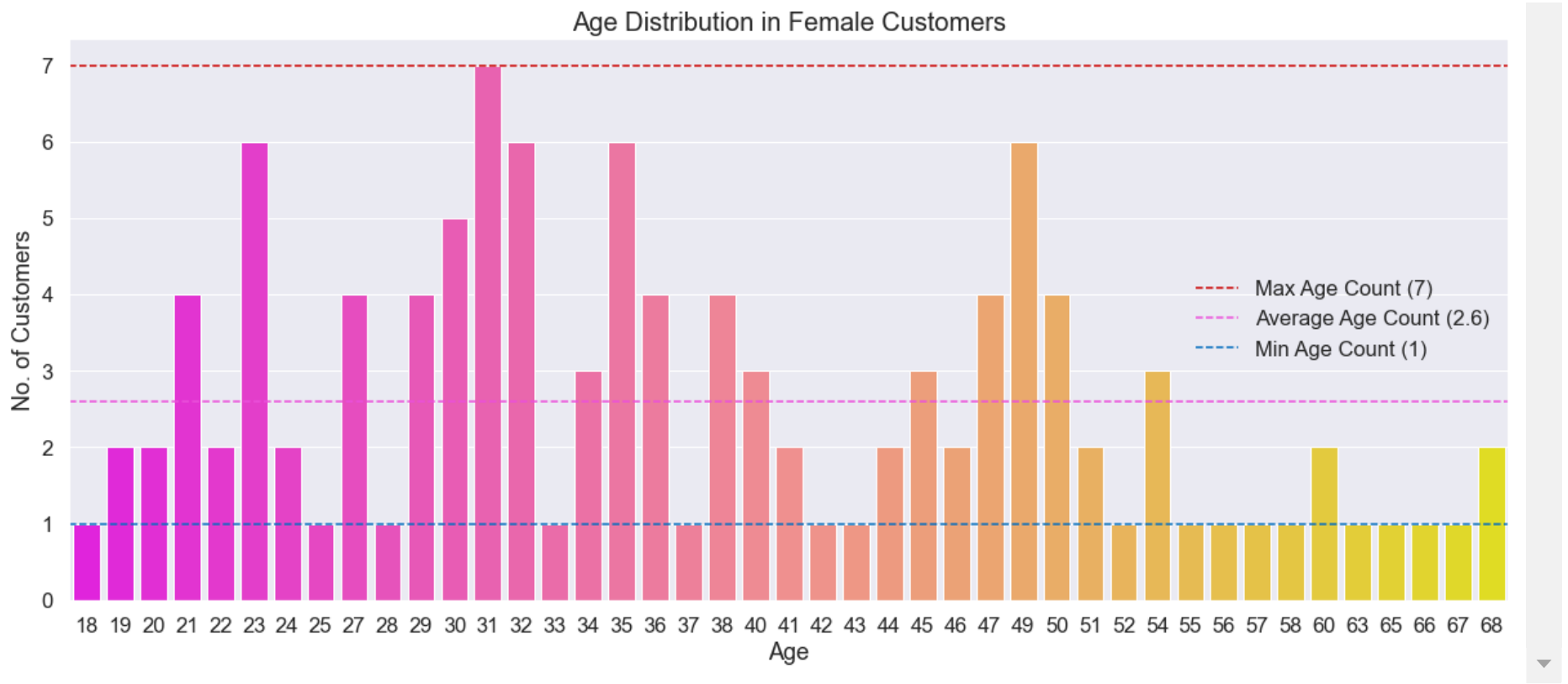
In [38]:
```
data[data['Gender']=='Female'].Age.mean()
```

Out[38]:
```
38.098214285714285
```

In [39]:
```
data[data['Gender']=='Female'].Age.value_counts().head()
```

Out[39]:
```
31    7
23    6
49    6
32    6
35    6
Name: Age, dtype: int64
```

In [40]:
```
# Visualizing distribution of age count in Female customers using a countplot.

maxi = data[data['Gender']=='Female'].Age.value_counts().max()
mean = data[data['Gender']=='Female'].Age.value_counts().mean()
mini = data[data['Gender']=='Female'].Age.value_counts().min()


fig, ax = plt.subplots(figsize=(20,8))
sns.set(font_scale=1.5)
ax = sns.countplot(x=data[data['Gender']=='Female'].Age, palette='spring')
ax.axhline(y=maxi, linestyle='--',color='#c90404', label=f'Max Age Count ({maxi})')
ax.axhline(y=mean, linestyle='--',color='#eb50db', label=f'Average Age Count ({mean:.1f})')
ax.axhline(y=mini, linestyle='--',color='#046ebf', label=f'Min Age Count ({mini})')
ax.set_ylabel('No. of Customers')
ax.legend(loc ='right')

plt.title('Age Distribution in Female Customers', fontsize = 20)
plt.show()
```



In [ ]:
```
# Analyzing Data for Modelling
# Analyzing Annual Income data
```

In [41]:
```
data['Annual_Income'].head()
```

Out[41]:
```
0    15
1    15
2    16
3    16
4    17
Name: Annual_Income, dtype: int64
```

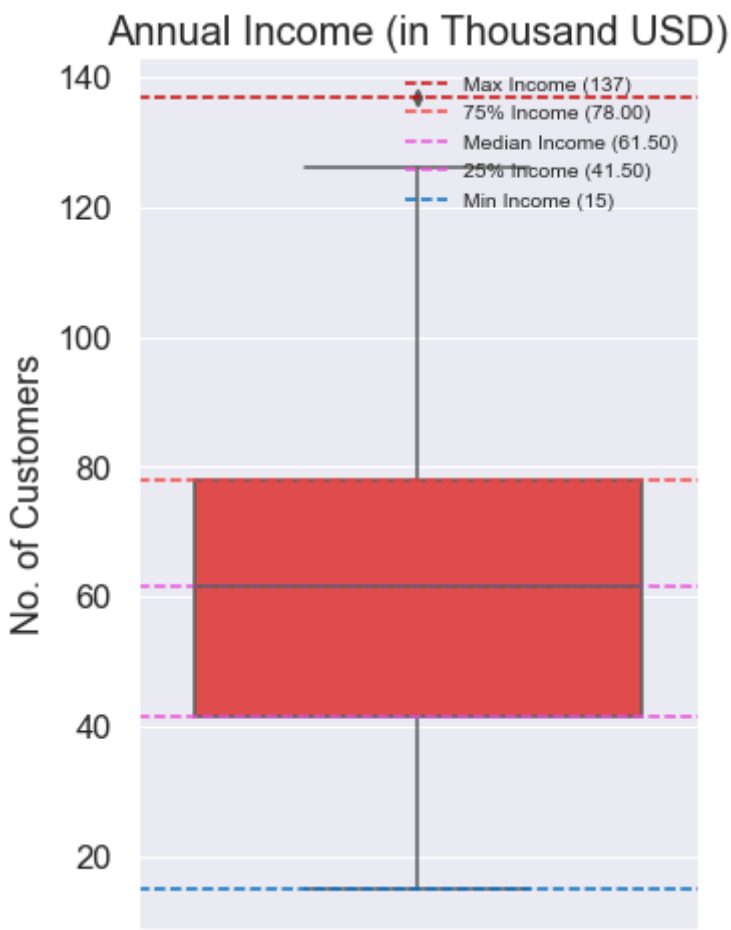
In [42]:
```
data['Annual_Income'].dtype
```

Out[42]:
```
dtype('int64')
```

In [43]:
```
data['Annual_Income'].describe()
```

Out[43]:
```
count    200.000000
mean      60.560000
std       26.264721
min       15.000000
25%       41.500000
50%       61.500000
75%       78.000000
max      137.000000
Name: Annual_Income, dtype: float64
```

In [44]:
```
# Visualizing statistical data about Annual Income column on a boxplot.
fig, ax = plt.subplots(figsize=(5,8))
sns.set(font_scale=1.5)
ax = sns.boxplot(y=data["Annual_Income"], color="#f73434")
ax.axhline(y=data["Annual_Income"].max(), linestyle='--',color='#c90404', label=f'Max Income ({data.Annual_Income.max()}
ax.axhline(y=data["Annual_Income"].describe()[6], linestyle='--',color='#f74343', label=f'75% Income ({data.Annual_Incom
ax.axhline(y=data["Annual_Income"].median(), linestyle='--',color='#eb50db', label=f'Median Income ({data.Annual_Income.
ax.axhline(y=data["Annual_Income"].describe()[4], linestyle='--',color='#eb50db', label=f'25% Income ({data.Annual_Incom
ax.axhline(y=data["Annual_Income"].min(), linestyle='--',color='#046ebf', label=f'Min Income ({data.Annual_Income.min()}
ax.legend(fontsize='xx-small', loc='upper right')
ax.set_ylabel('No. of Customers')

plt.title('Annual Income (in Thousand USD)', fontsize = 20)
plt.show()
```

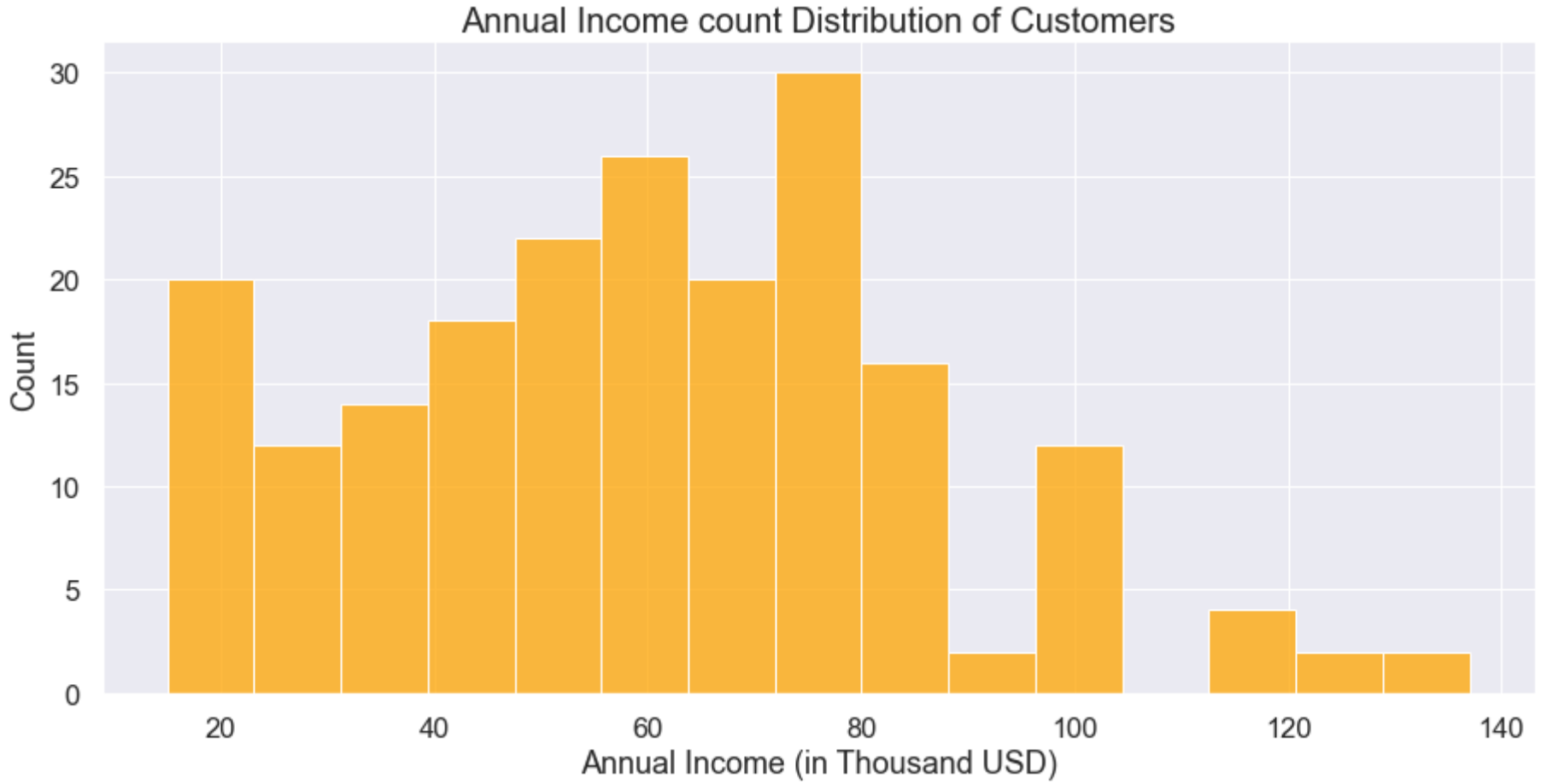


In [ ]:
```
# Distribution of Annual Income counts.
```

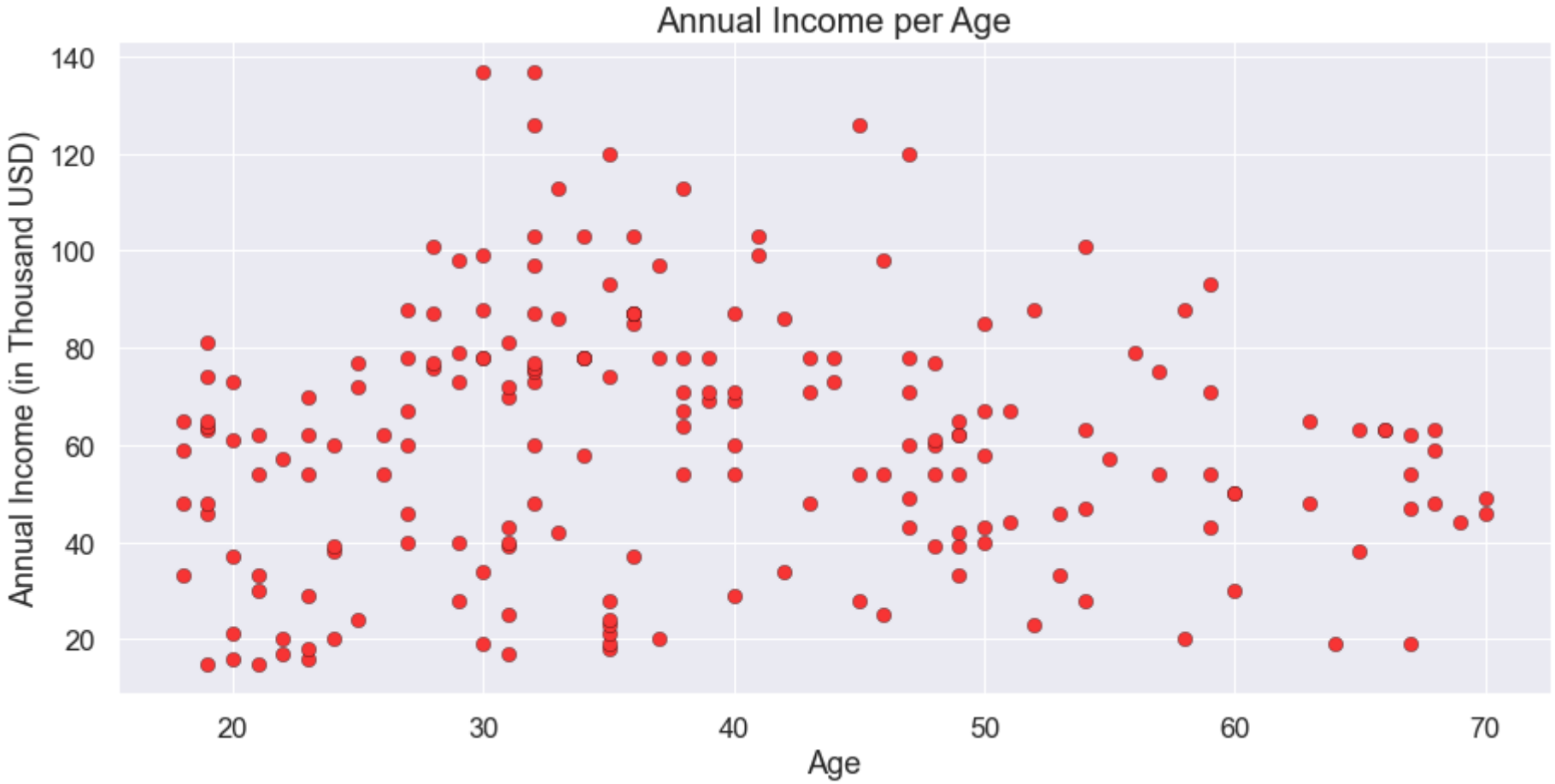
In [45]:
```
data['Annual_Income'].value_counts().head()
```

Out[45]:
```
54    12
78    12
48     6
71     6
63     6
Name: Annual_Income, dtype: int64
```

In [46]:
```
# Visualizing Annual Income count value distribution on a histogram.
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale=1.5)
ax = sns.histplot(data['Annual_Income'], bins=15, ax=ax, color=['orange'])
ax.set_xlabel('Annual Income (in Thousand USD)')
plt.title('Annual Income count Distribution of Customers', fontsize = 20)
plt.show()
```



In [47]:
```
# Visualizing Annual Income per Age on a Scatterplot.
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale=1.5)
ax = sns.scatterplot(y=data['Annual_Income'], x=data['Age'], color='#f73434', s=70,edgecolor='black', linewidth=0.3)
ax.set_ylabel('Annual Income (in Thousand USD)')

plt.title('Annual Income per Age', fontsize = 20)
plt.show()
```



In [ ]:
```
# Annual Income per Gender.
```

In [48]:
```
# Statistical data about the Annual Income of male customer.
data[data['Gender']=='Male'].Annual_Income.describe()
```

Out[48]:
```
count     88.000000
mean      62.227273
std       26.638373
min       15.000000
25%       45.500000
50%       62.500000
75%       78.000000
max      137.000000
Name: Annual_Income, dtype: float64
```

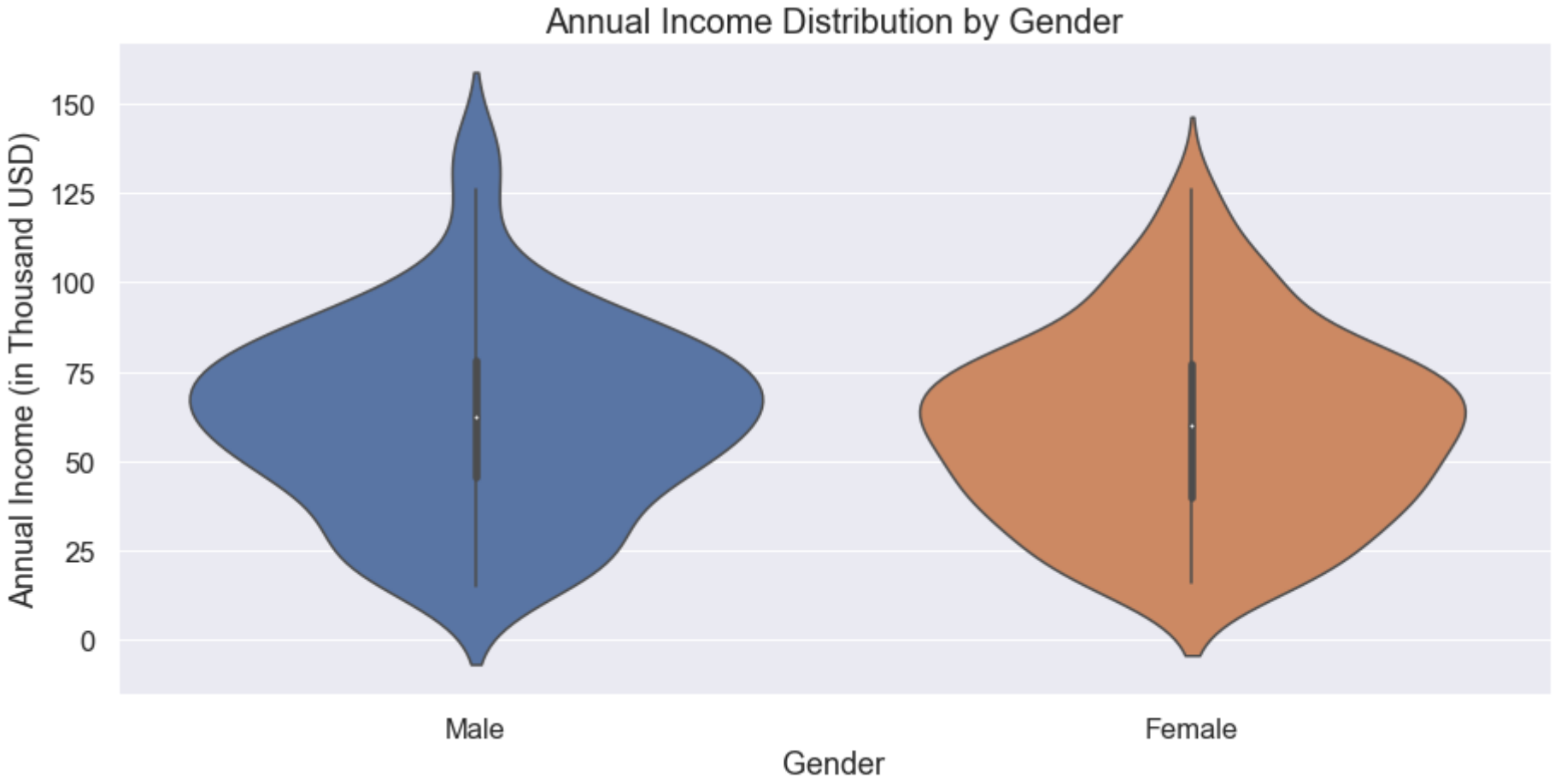
In [49]:
```
# Statistical data about the Annual Income of female customer.
data[data['Gender']=='Female'].Annual_Income.describe()
```

Out[49]:
```
count    112.000000
mean      59.250000
std       26.011952
min       16.000000
25%       39.750000
50%       60.000000
75%       77.250000
max      126.000000
Name: Annual_Income, dtype: float64
```

In [50]:
```
# Visualizing statistical difference of Annual Income between Male and Female Customers.
fig, ax = plt.subplots(figsize=(10,8))
sns.set(font_scale=1.5)
ax = sns.boxplot(x=data['Gender'], y=data["Annual_Income"], hue=data['Gender'], palette='seismic')
ax.set_ylabel('Annual Income (in Thousand USD)')

plt.title('Annual Income Distribution by Gender', fontsize = 20)
plt.show()
```

In [51]:
```
# Visualizing annual Income per Age by Gender on a scatterplot.
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale=1.5)
ax = sns.scatterplot(y=data['Annual_Income'], x=data['Age'], hue=data['Gender'], palette='seismic', s=70,edgecolor='blacl
ax.set_ylabel('Annual Income (in Thousand USD)')
ax.legend(loc ='upper right')

plt.title('Annual Income per Age by Gender', fontsize = 20)
plt.show()
```

In [52]:
```
# Visualizing difference of Annual Income between Male and Female Customers using Violin Plot.
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale=1.5)
ax = sns.violinplot(y=data['Annual_Income'],x=data['Gender'])
ax.set_ylabel('Annual Income (in Thousand USD)')

plt.title('Annual Income Distribution by Gender', fontsize = 20)
plt.show()
```



In [ ]:
```
# Analyzing Spending Score data
```

In [53]:
```
data['Spending_Score'].head()
```

Out[53]:
```
0    39
1    81
2     6
3    77
4    40
Name: Spending_Score, dtype: int64
```

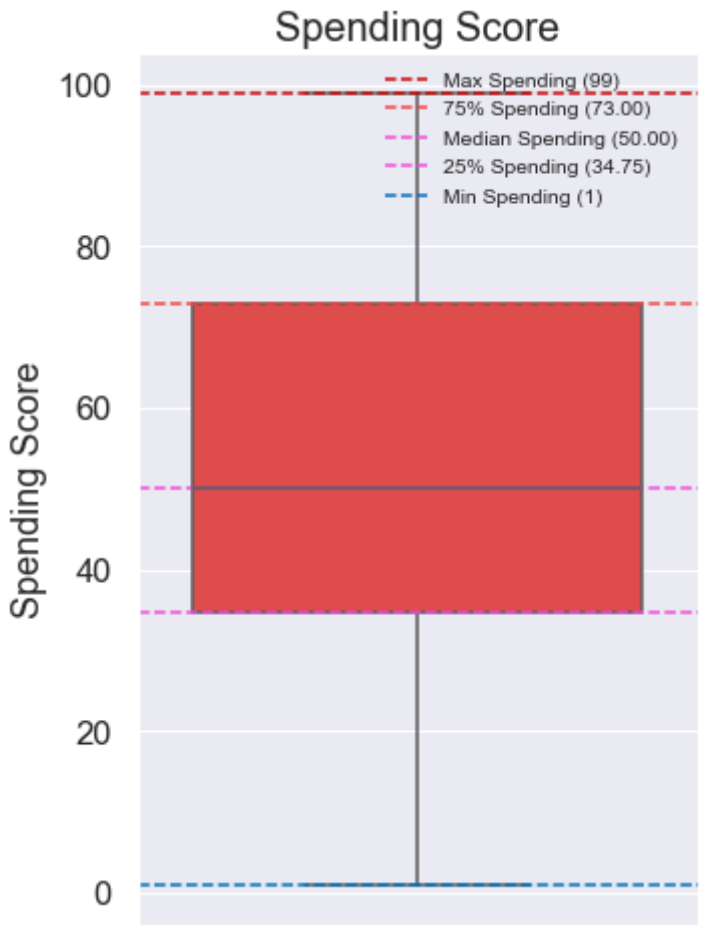
In [54]:
```
data['Spending_Score'].dtype
```

Out[54]:
```
dtype('int64')
```

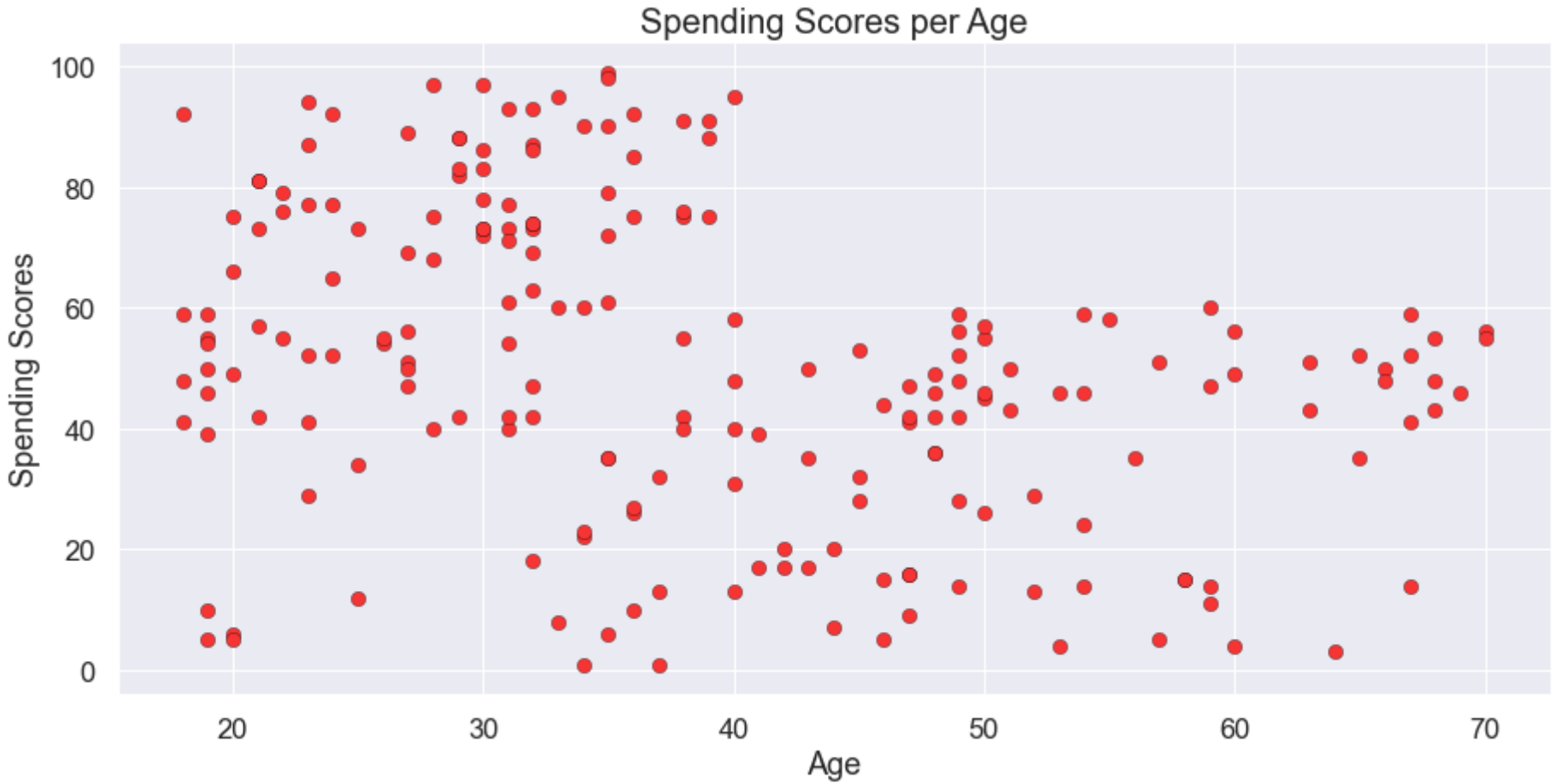
In [55]:
```
data['Spending_Score'].describe()
```

Out[55]:
```
count    200.000000
mean      50.200000
std       25.823522
min        1.000000
25%       34.750000
50%       50.000000
75%       73.000000
max       99.000000
Name: Spending_Score, dtype: float64
```

In [56]:
```
# Visualizing statistical data about Spending score column on a boxplot.

fig, ax = plt.subplots(figsize=(5,8))
sns.set(font_scale=1.5)
ax = sns.boxplot(y=data['Spending_Score'], color="#f73434")
ax.axhline(y=data['Spending_Score'].max(), linestyle='--',color='#c90404', label=f'Max Spending ({data.Spending_Score.ma
ax.axhline(y=data['Spending_Score'].describe()[6], linestyle='--',color='#f74343', label=f'75% Spending ({data.Spending_
ax.axhline(y=data['Spending_Score'].median(), linestyle='--',color='#eb50db', label=f'Median Spending ({data.Spending_Sc
ax.axhline(y=data['Spending_Score'].describe()[4], linestyle='--',color='#eb50db', label=f'25% Spending ({data.Spending_
ax.axhline(y=data['Spending_Score'].min(), linestyle='--',color='#046ebf', label=f'Min Spending ({data.Spending_Score.mi
ax.legend(fontsize='xx-small', loc='upper right')
ax.set_ylabel('Spending Score')

plt.title('Spending Score', fontsize = 20)
plt.show()
```


Spending Score
Max Spending (99)
75% Spending (73.00)
Median Spending (50.00)
25% Spending (34.75)
Min Spending (1)
Spending Score
100
80
60
40
20
0

In [57]:
```
# Visualizing Spending Scores per Age on a scatterplot.
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale=1.5)
ax = sns.scatterplot(y=data['Spending_Score'], x=data['Age'], s=70, color='#f73434', edgecolor='black', linewidth=0.3)
ax.set_ylabel('Spending Scores')

plt.title('Spending Scores per Age', fontsize = 20)
plt.show()
```



In [ ]:
```
# Spending Scores per Gender
```

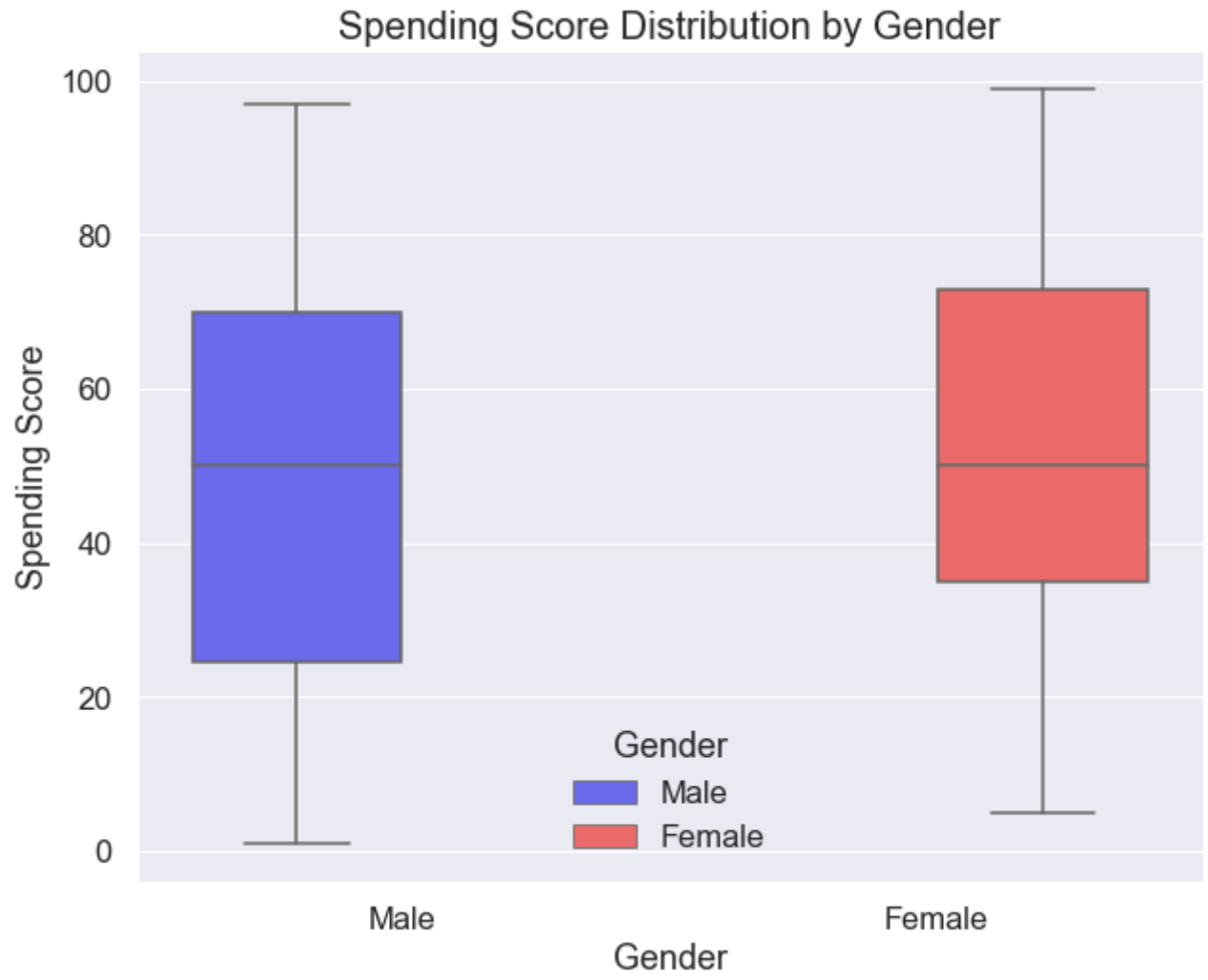
In [58]:
```
# Male customer
data[data['Gender']=='Male'].Annual_Income.describe()
```

Out[58]:
```
count     88.000000
mean      62.227273
std       26.638373
min       15.000000
25%       45.500000
50%       62.500000
75%       78.000000
max      137.000000
Name: Annual_Income, dtype: float64
```

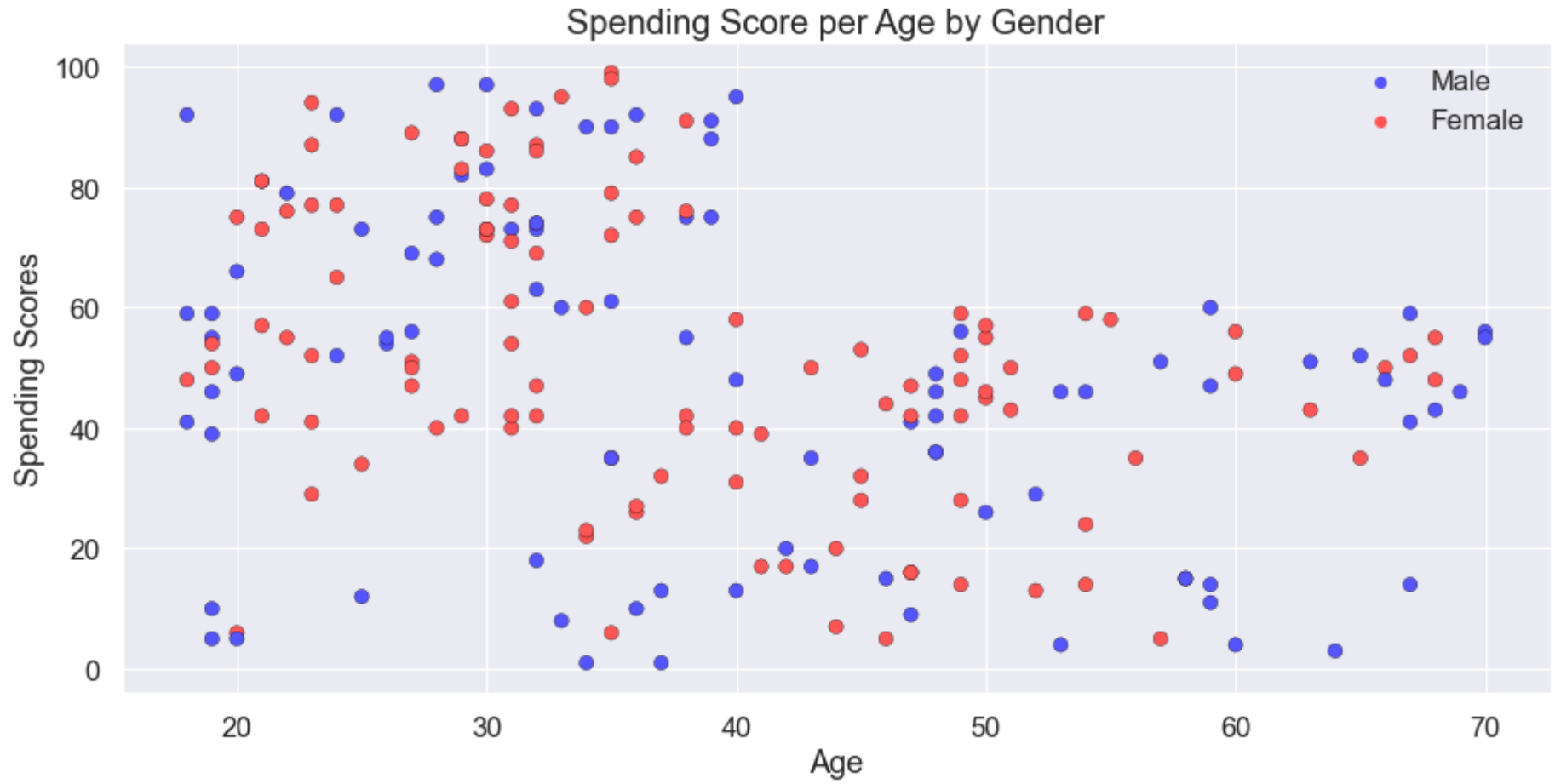
In [59]:
```
# Female customer
data[data['Gender']=='Female'].Annual_Income.describe()
```

Out[59]:
```
count    112.000000
mean      59.250000
std       26.011952
min       16.000000
25%       39.750000
50%       60.000000
75%       77.250000
max      126.000000
Name: Annual_Income, dtype: float64
```

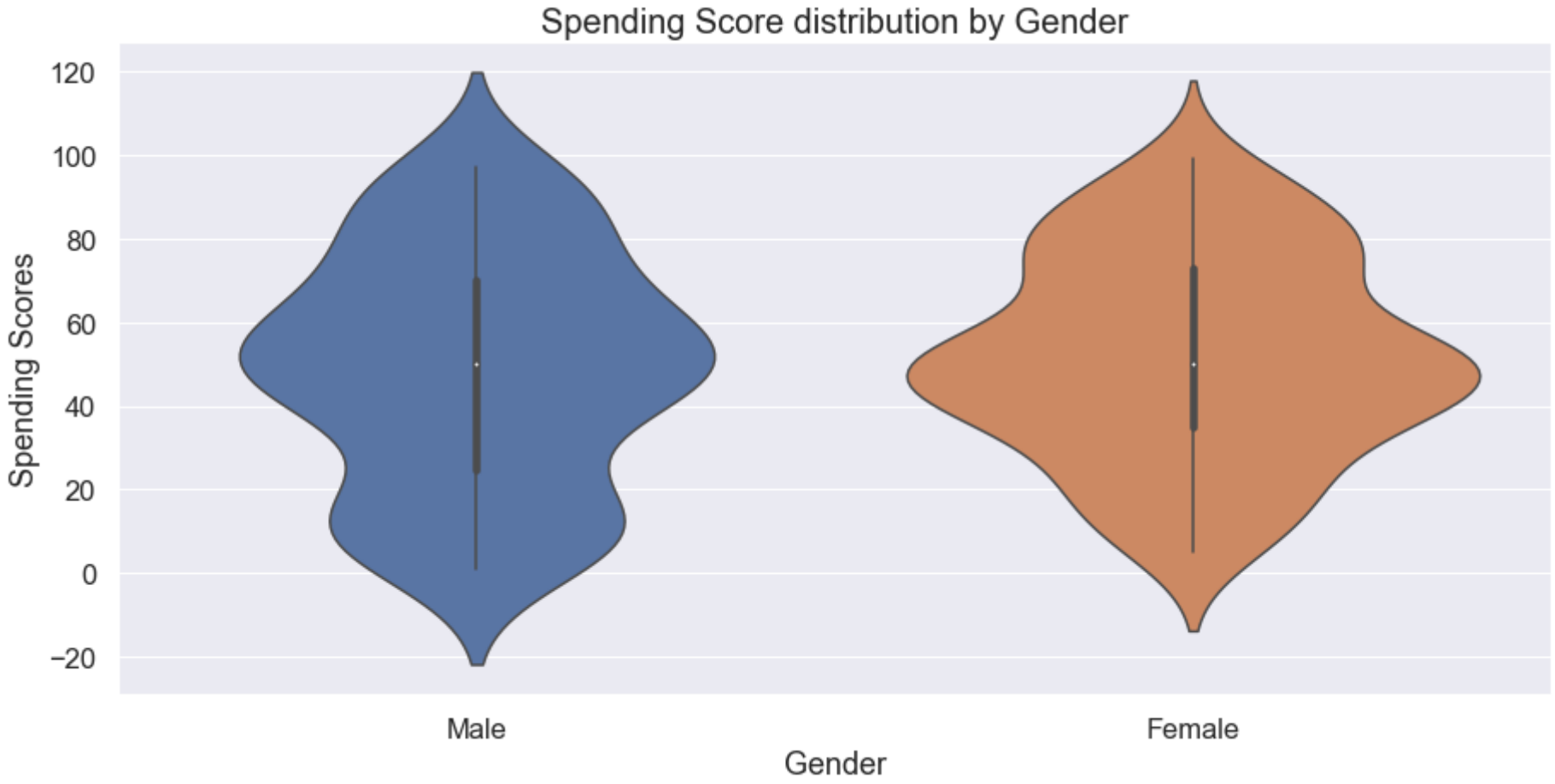
In [60]:
```
# Visualizing statistical difference of Spending Score between Male and Female Customers.
fig, ax = plt.subplots(figsize=(10,8))
sns.set(font_scale=1.5)
ax = sns.boxplot(x=data['Gender'], y=data["Spending_Score"], hue=data['Gender'], palette='seismic')
ax.set_ylabel('Spending Score')

plt.title('Spending Score Distribution by Gender', fontsize = 20)
plt.show()
```



In [61]:
```
# Visualizing Spending Score per Age by Gender on a scatterplot.
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale=1.5)
ax = sns.scatterplot(y=data['Spending_Score'], x=data['Age'], hue=data['Gender'], palette='seismic', s=70,edgecolor='bla
ax.set_ylabel('Spending Scores')
ax.legend(loc ='upper right')

plt.title('Spending Score per Age by Gender', fontsize = 20)
plt.show()
```



In [62]:
```
# Visualizing difference of Spending Score between Male and Female Customers using Violin Plot.
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale=1.5)
ax = sns.violinplot(y=data['Spending_Score'],x=data['Gender'])
ax.set_ylabel('Spending Scores')

plt.title('Spending Score distribution by Gender', fontsize = 20)
plt.show()
```



In [ ]:

```
```

In [ ]:

```
# K - Means Clustering
```

In [63]:

```
# First we need to check the data for any missing values as it can ruin our model.
data.isna().sum()
```

Out[63]:

```
Gender            0
Age               0
Annual_Income     0
Spending_Score    0
dtype: int64
```

In [64]:

```
data.head()
```

Out[64]:

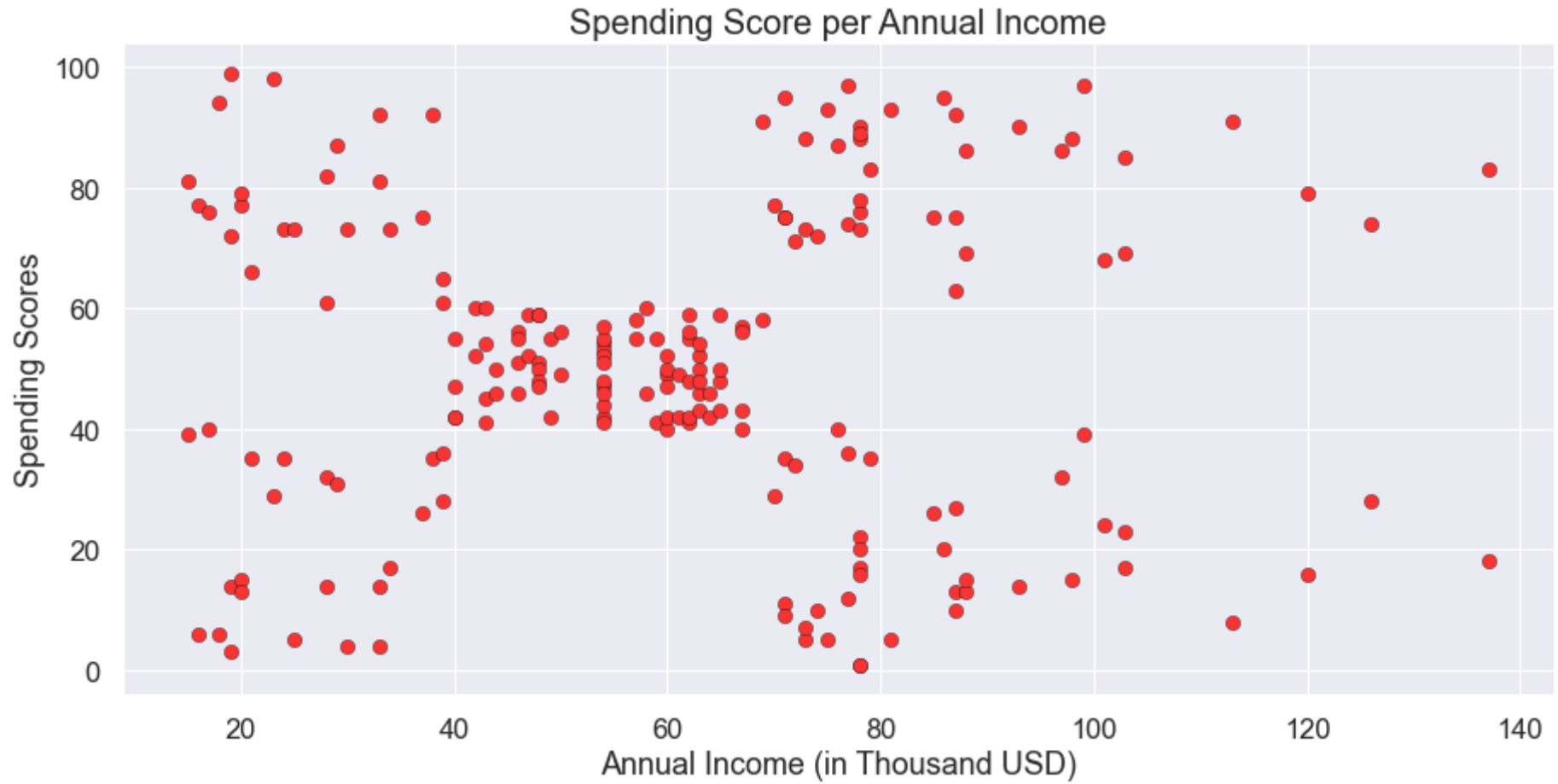
	Gender	Age	Annual_Income	Spending_Score
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40

In [65]:
```
clustering_data = data.iloc[:,[2,3]]
clustering_data.head()
```

Out[65]:

	Annual_Income	Spending_Score
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

In [66]:
```
# Now, we need to visualize the data which we are going to use for the clustering. This will give us a fair idea about th
fig, ax = plt.subplots(figsize=(15,7))
sns.set(font_scale=1.5)
ax = sns.scatterplot(y=clustering_data['Spending_Score'],x=clustering_data['Annual_Income'], s=70, color='#f73434', edge
ax.set_ylabel('Spending Scores')
ax.set_xlabel('Annual Income (in Thousand USD)')
plt.title('Spending Score per Annual Income', fontsize = 20)
plt.show()
```

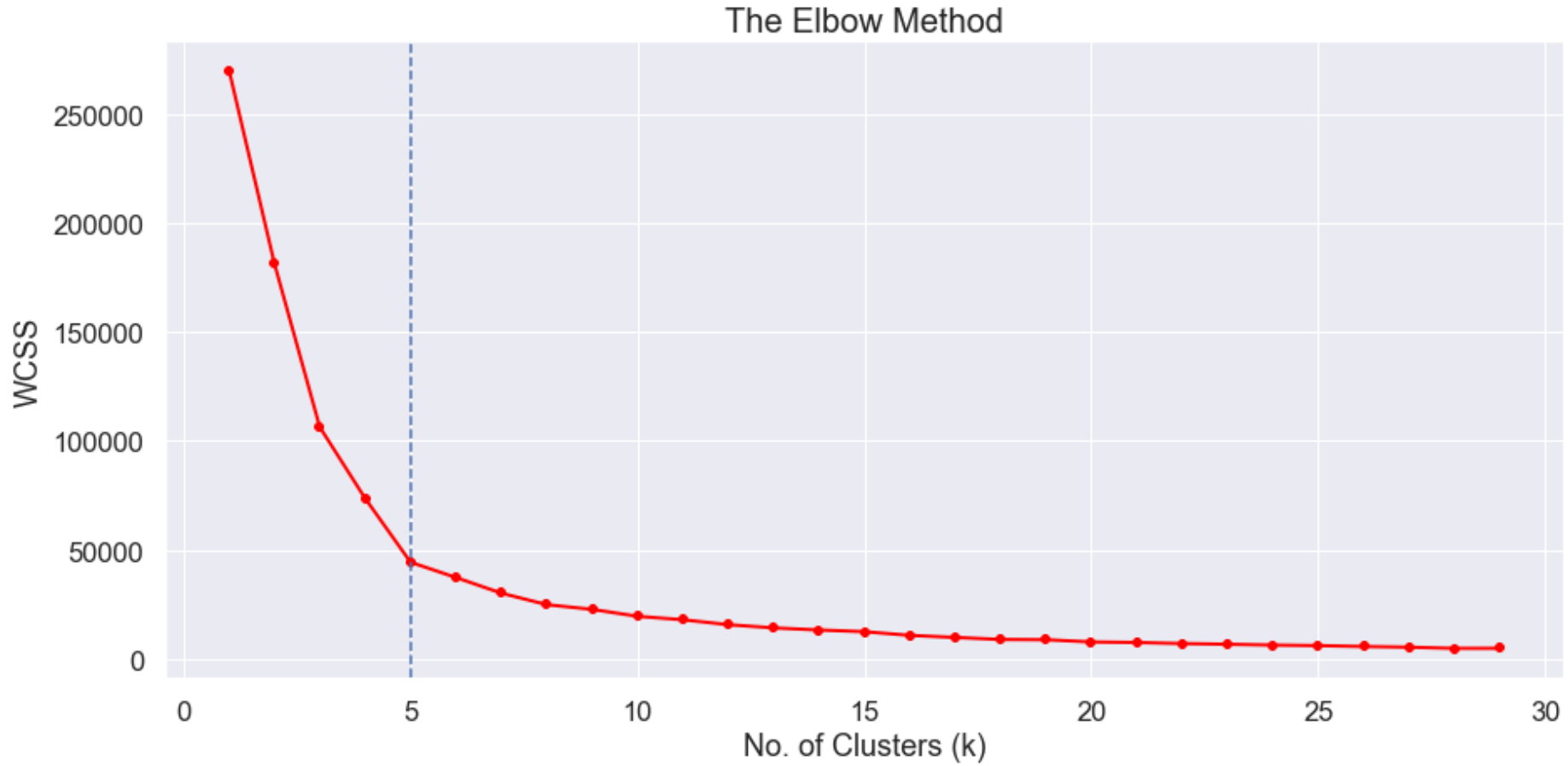


In [ ]:

In [67]:
```
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,30):
    km = KMeans(i)
    km.fit(clustering_data)
    wcss.append(km.inertia_)
np.array(wcss)
```

Out[67]:
```
array([269981.28      , 181363.5959596 , 106348.37306211,  73679.78903949,
        44448.45544793,  37442.24745038,  30259.65720729,  24999.36825861,
        22835.63880139,  19657.7836087 ,  18074.62170149,  15810.34342654,
        14343.35459279,  13371.6793251 ,  12596.98252042,  10916.91406304,
         9965.83675491,   9047.31040373,   8926.82777778,   7896.78002143,
         7654.77690088,   7164.05154249,   6847.43966083,   6474.57943432,
         6222.39308614,   5815.13748196,   5492.90720391,   4971.90329393,
         4996.52063492])
```

In [68]:
```
# Now, we visualize the Elbow Method so that we can determine the number of optimal clusters for our dataset.
fig, ax = plt.subplots(figsize=(15,7))
ax = plt.plot(range(1,30),wcss, linewidth=2, color="red", marker ="8")
plt.axvline(x=5, ls='--')
plt.ylabel('WCSS')
plt.xlabel('No. of Clusters (k)')
plt.title('The Elbow Method', fontsize = 20)
plt.show()
```



In [ ]:
```
#  Clustering
```

In [69]:
```
from sklearn.cluster import KMeans

kms = KMeans(n_clusters=5, init='k-means++')
kms.fit(clustering_data)
```

Out[69]:
```
KMeans(n_clusters=5)
```

In [70]:
```
clusters = clustering_data.copy()
clusters['Cluster_Prediction'] = kms.fit_predict(clustering_data)
clusters.head()
```

Out[70]:

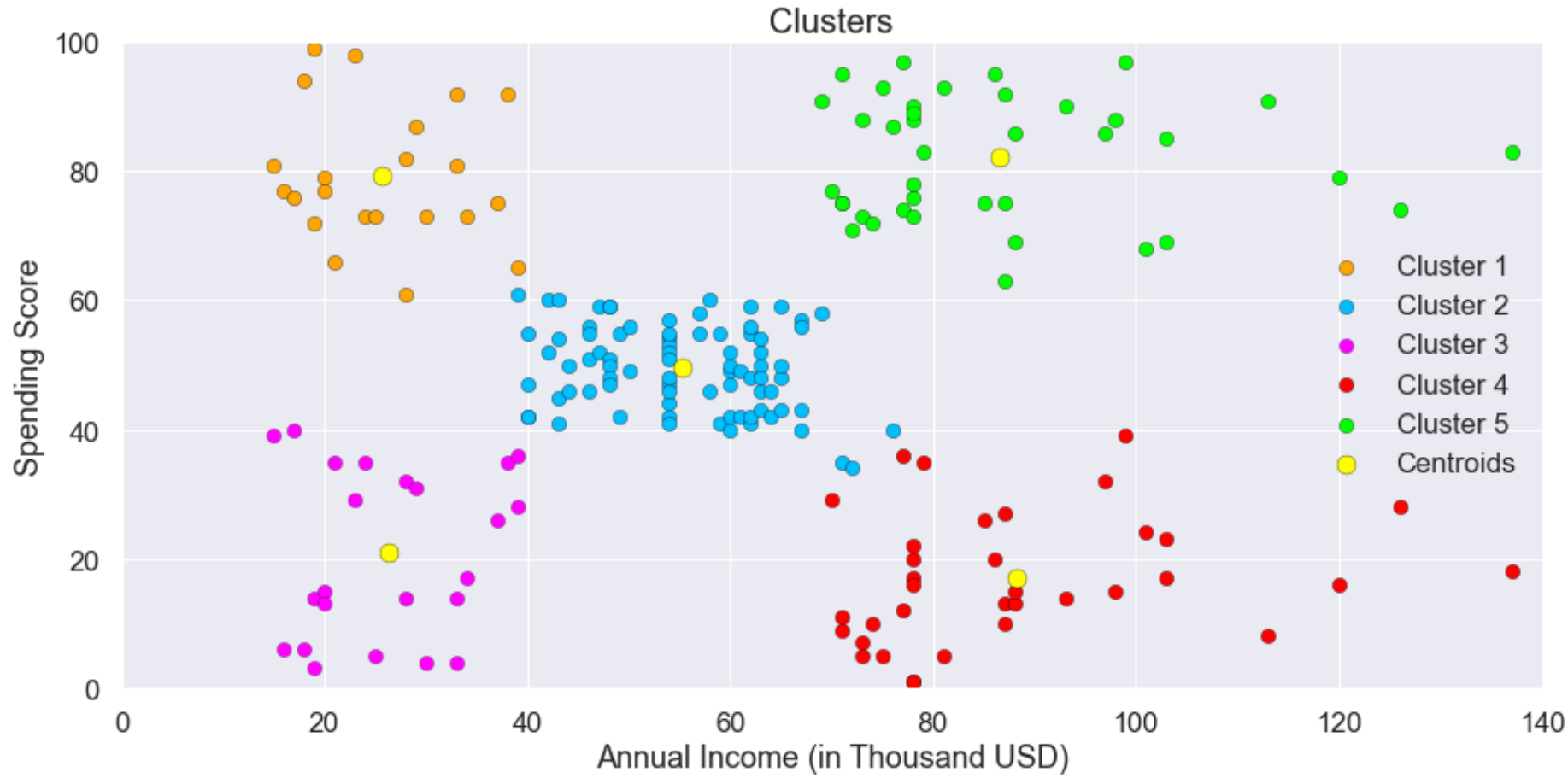
	Annual_Income	Spending_Score	Cluster_Prediction
0	15	39	2
1	15	81	4
2	16	6	2
3	16	77	4
4	17	40	2

In [71]:
```
# We can also get the centroids of the clusters by the cluster_centers_ attribute of KMeans algorithm.
kms.cluster_centers_
```

Out[71]:
```
array([[55.2962963 , 49.51851852],
       [88.2       , 17.11428571],
       [26.30434783, 20.91304348],
       [86.53846154, 82.12820513],
       [25.72727273, 79.36363636]])
```

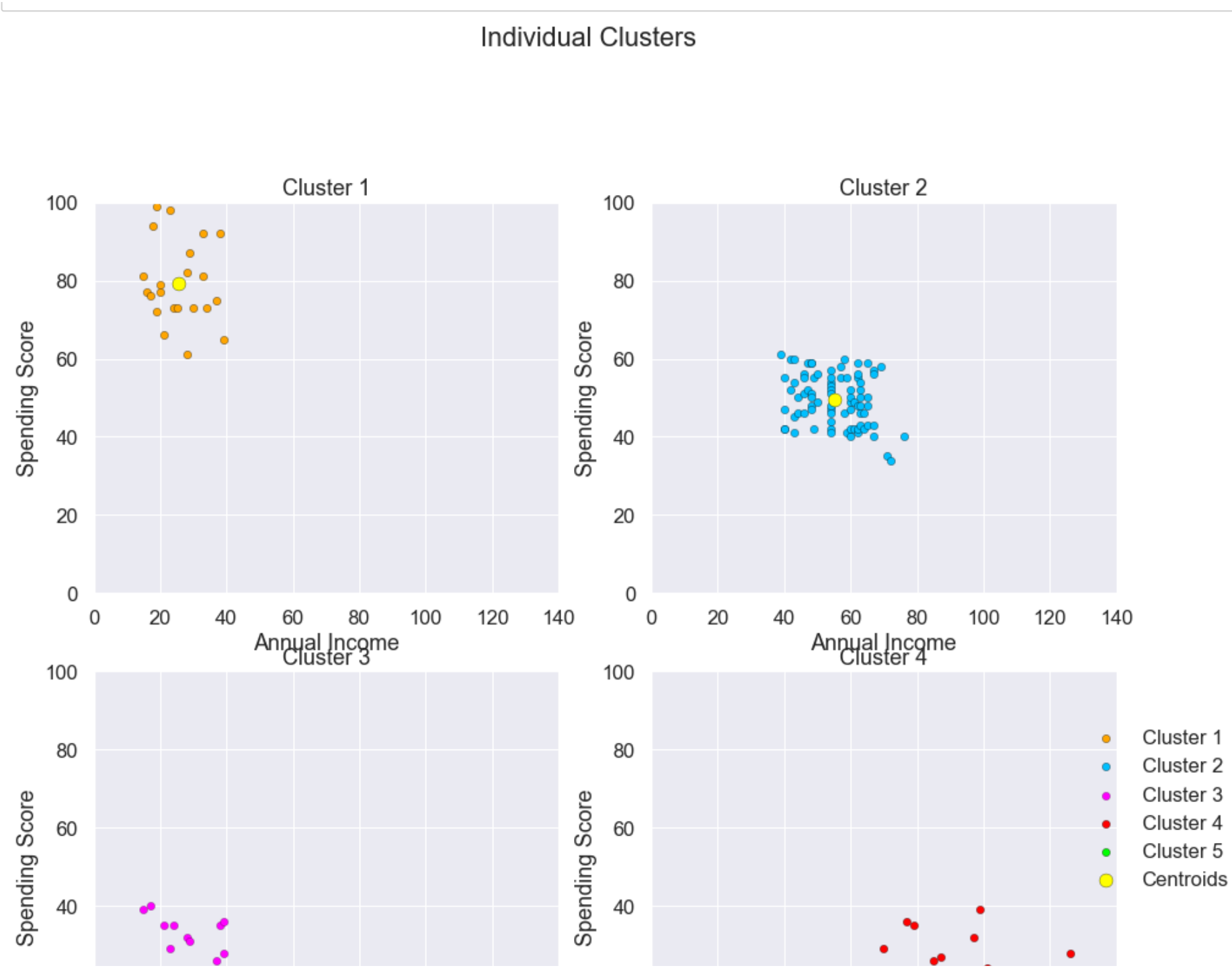
In [ ]:
```
# Now we have all the data we need, we just need to plot the data. We will plot the data using scatterplot which will al
```

In [72]:
```
fig, ax = plt.subplots(figsize=(15,7))
plt.scatter(x=clusters[clusters['Cluster_Prediction'] == 4]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 4]['Spending_Score'],
            s=70,edgecolor='black', linewidth=0.3, c='orange', label='Cluster 1')

plt.scatter(x=clusters[clusters['Cluster_Prediction'] == 0]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 0]['Spending_Score'],
            s=70,edgecolor='black', linewidth=0.3, c='deepskyblue', label='Cluster 2')

plt.scatter(x=clusters[clusters['Cluster_Prediction'] == 2]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 2]['Spending_Score'],
            s=70,edgecolor='black', linewidth=0.2, c='Magenta', label='Cluster 3')

plt.scatter(x=clusters[clusters['Cluster_Prediction'] == 1]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 1]['Spending_Score'],
            s=70,edgecolor='black', linewidth=0.3, c='red', label='Cluster 4')

plt.scatter(x=clusters[clusters['Cluster_Prediction'] == 3]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 3]['Spending_Score'],
            s=70,edgecolor='black', linewidth=0.3, c='lime', label='Cluster 5')



plt.scatter(x=kms.cluster_centers_[:, 0], y=kms.cluster_centers_[:, 1], s = 120, c = 'yellow', label = 'Centroids',edgec
plt.legend(loc='right')
plt.xlim(0,140)
plt.ylim(0,100)
plt.xlabel('Annual Income (in Thousand USD)')
plt.ylabel('Spending Score')
plt.title('Clusters', fontsize = 20)
plt.show()
```

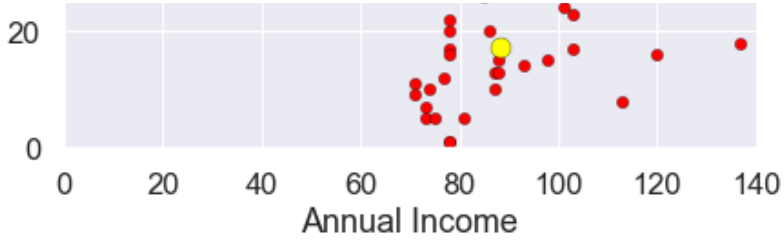

Clusters
Spending Score
Annual Income (in Thousand USD)
0
20
40
60
80
100
120
140
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids

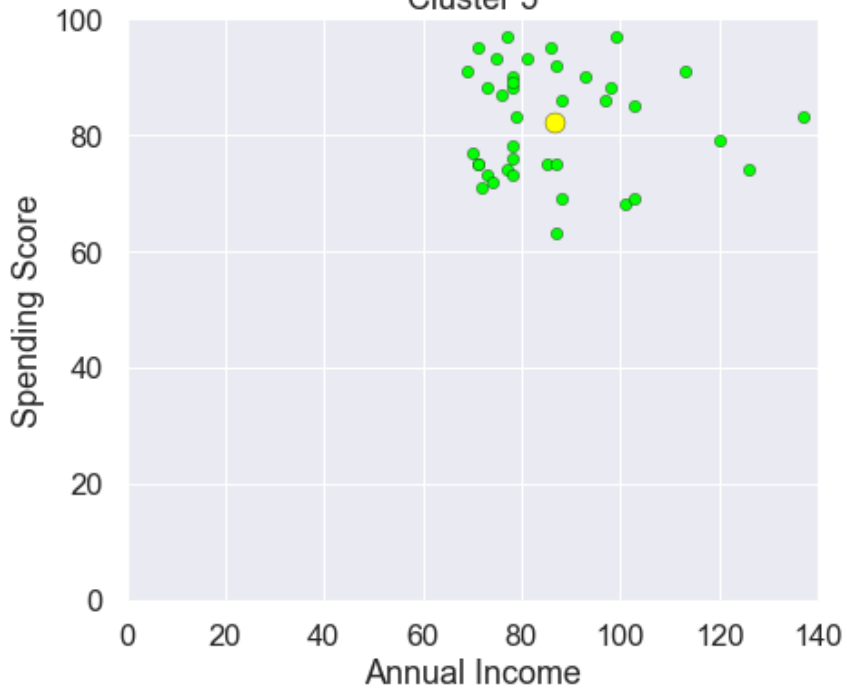
In [ ]:
```
# Analysis
```

In [73]:
```
# Visualizing all the clusters Seperately will provide us more insights.
fig, ax = plt.subplots(nrows=3, ncols=2, figsize=(15,20))
ax[0,0].scatter(x=clusters[clusters['Cluster_Prediction'] == 4]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 4]['Spending_Score'],
            s=40,edgecolor='black', linewidth=0.3, c='orange', label='Cluster 1')
ax[0,0].scatter(x=kms.cluster_centers_[4,0], y=kms.cluster_centers_[4,1],
                s = 120, c = 'yellow',edgecolor='black', linewidth=0.3)
ax[0,0].set(xlim=(0,140), ylim=(0,100), xlabel='Annual Income', ylabel='Spending Score', title='Cluster 1')

ax[0,1].scatter(x=clusters[clusters['Cluster_Prediction'] == 0]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 0]['Spending_Score'],
            s=40,edgecolor='black', linewidth=0.3, c='deepskyblue', label='Cluster 2')
ax[0,1].scatter(x=kms.cluster_centers_[0,0], y=kms.cluster_centers_[0,1],
                s = 120, c = 'yellow',edgecolor='black', linewidth=0.3)
ax[0,1].set(xlim=(0,140), ylim=(0,100), xlabel='Annual Income', ylabel='Spending Score', title='Cluster 2')

ax[1,0].scatter(x=clusters[clusters['Cluster_Prediction'] == 2]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 2]['Spending_Score'],
            s=40,edgecolor='black', linewidth=0.2, c='Magenta', label='Cluster 3')
ax[1,0].scatter(x=kms.cluster_centers_[2,0], y=kms.cluster_centers_[2,1],
                s = 120, c = 'yellow',edgecolor='black', linewidth=0.3)
ax[1,0].set(xlim=(0,140), ylim=(0,100), xlabel='Annual Income', ylabel='Spending Score', title='Cluster 3')

ax[1,1].scatter(x=clusters[clusters['Cluster_Prediction'] == 1]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 1]['Spending_Score'],
            s=40,edgecolor='black', linewidth=0.3, c='red', label='Cluster 4')
ax[1,1].scatter(x=kms.cluster_centers_[1,0], y=kms.cluster_centers_[1,1],
                s = 120, c = 'yellow',edgecolor='black', linewidth=0.3)
ax[1,1].set(xlim=(0,140), ylim=(0,100), xlabel='Annual Income', ylabel='Spending Score', title='Cluster 4')

ax[2,0].scatter(x=clusters[clusters['Cluster_Prediction'] == 3]['Annual_Income'],
            y=clusters[clusters['Cluster_Prediction'] == 3]['Spending_Score'],
            s=40,edgecolor='black', linewidth=0.3, c='lime', label='Cluster 5')
ax[2,0].scatter(x=kms.cluster_centers_[3,0], y=kms.cluster_centers_[3,1],
                s = 120, c = 'yellow',edgecolor='black', linewidth=0.3, label='Centroids')
ax[2,0].set(xlim=(0,140), ylim=(0,100), xlabel='Annual Income', ylabel='Spending Score', title='Cluster 5')

fig.delaxes(ax[2,1])
fig.legend(loc='right')
fig.suptitle('Individual Clusters')
plt.show()
```


Individual Clusters
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Spending Score
Annual Income
0
20
40
60
80
100
120
140
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids


20
0
0 20 40 60 80 100 120 140
Annual Income


20
0
0 20 40 60 80 100 120 140
Annual Income


Cluster 5
100
80
60
40
20
0
Spending Score
0 20 40 60 80 100 120 140
Annual Income

In [ ]: