In [2]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

#warning hadle
import warnings
warnings.filterwarnings("ignore")
```

# Download and load the dataset into the tool.

In [3]:

```
data  =  pd.read_csv('/content/Mall_Customers.csv')
data.head()
```

Out[3]:

| | 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 |

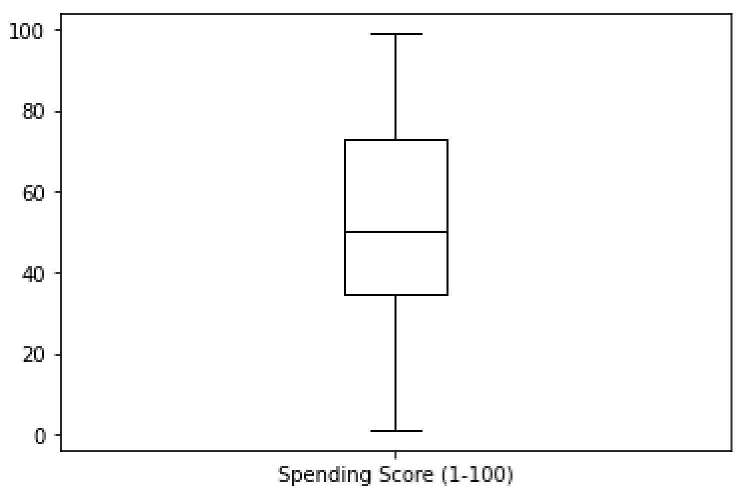
# Univariate Analysis

In [4]:
```
data['Spending Score (1-100)'].value_counts()
```

Out[4]:
```
42    8
55    7
46    6
73    6
35    5
     ..
31    1
44    1
53    1
65    1
18    1
Name: Spending Score (1-100), Length: 84, dtype: int64
```

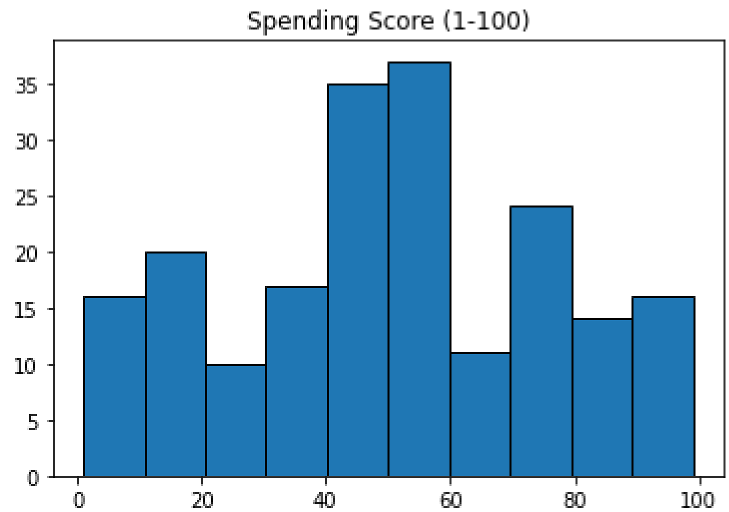
In [5]:
```
data.boxplot(column=['Spending Score (1-100)'], grid=False, color='black')
```

Out[5]:



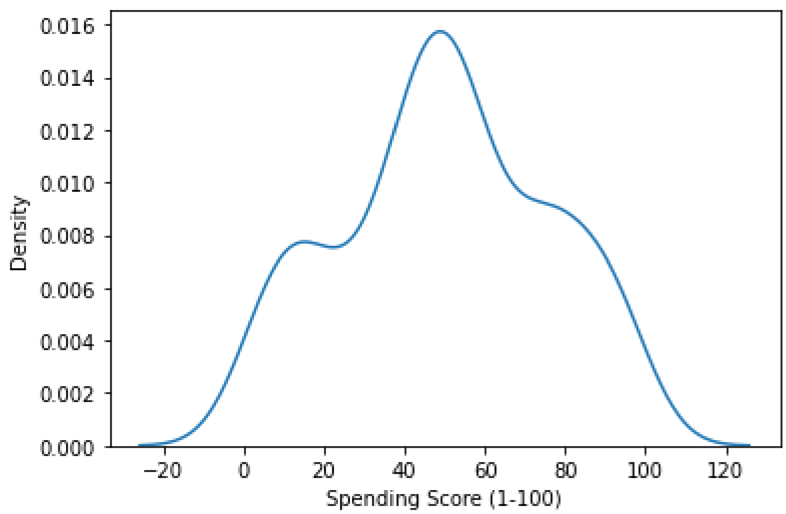

In [6]:
```
data.hist(column='Spending Score (1-100)', grid=False, edgecolor='black')
```

Out[6]:
```
array([[]],
      dtype=object)
```




In [7]:
```
sns.kdeplot(data['Spending Score (1-100)'])
```

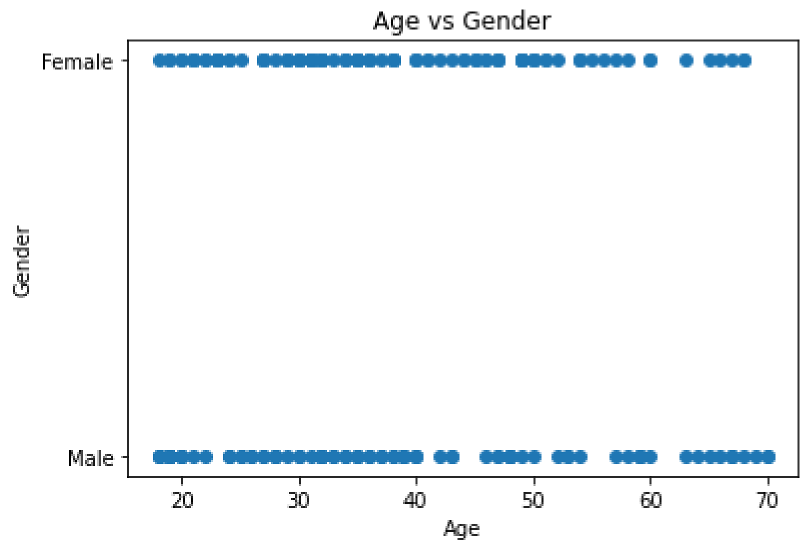
Out[7]:




# Bi- Variate Analysis

In [8]:
```
plt.scatter(data.Age,  data.Gender)
plt.title('Age vs Gender')
plt.xlabel('Age')
plt.ylabel('Gender')
```

Out[8]: Text(0, 0.5, 'Gender')




In [9]:

```
data.corr()
```

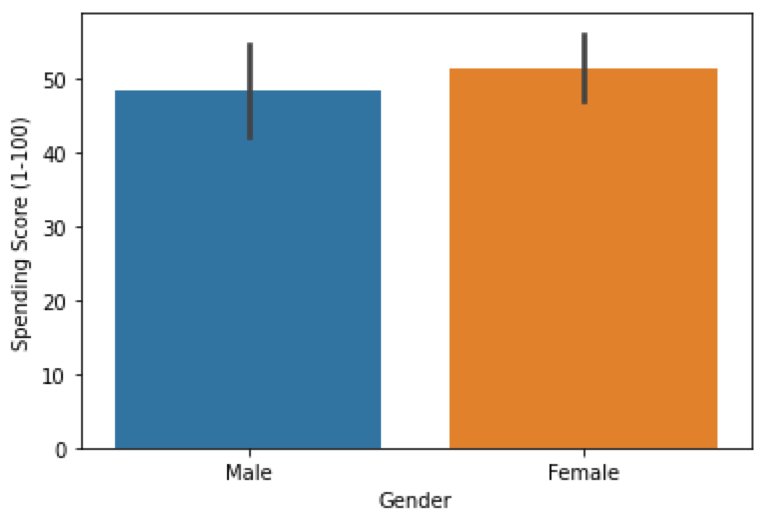
Out[9]:

| | CustomerID | Age | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **CustomerID** | 1.000000 | -0.026763 | 0.977548 | 0.013835 |
| **Age** | -0.026763 | 1.000000 | -0.012398 | -0.327227 |
| **Annual Income (k$)** | 0.977548 | -0.012398 | 1.000000 | 0.009903 |
| **Spending Score (1-100)** | 0.013835 | -0.327227 | 0.009903 | 1.000000 |

In [10]:

```
sns.barplot(x='Gender',y='Spending  Score  (1-100)',data=data)
```

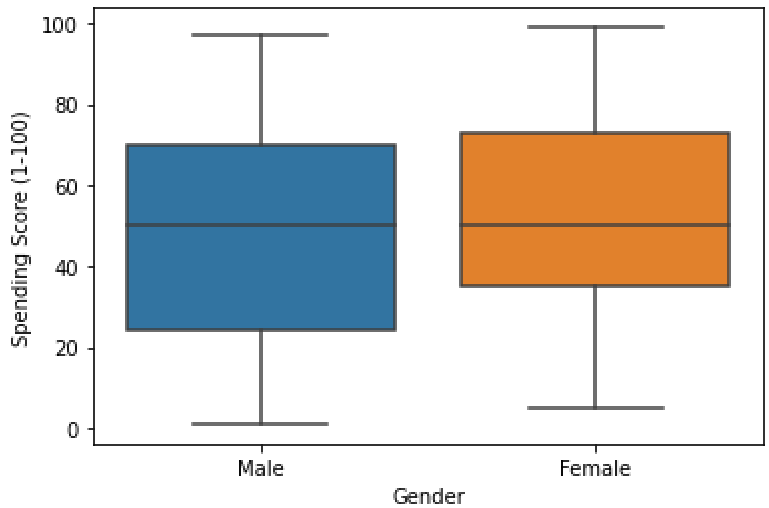
Out[10]:




In [11]:

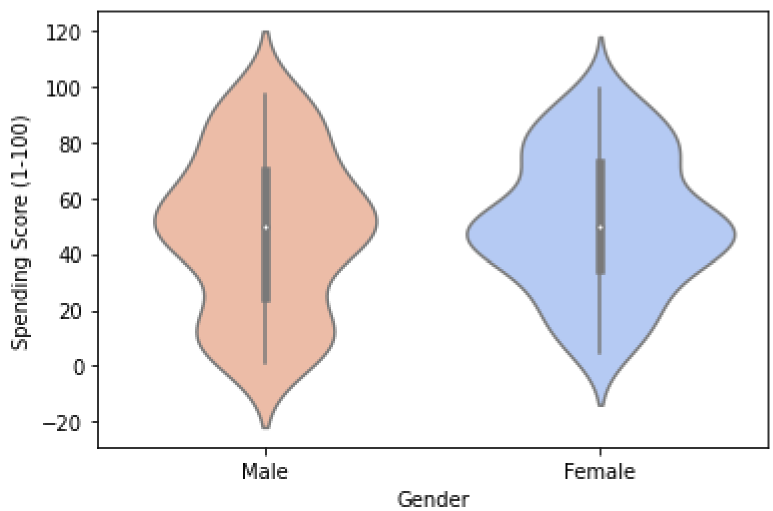
```
sns.boxplot(x='Gender',y='Spending  Score  (1-100)',data=data)
```

Out[11]:



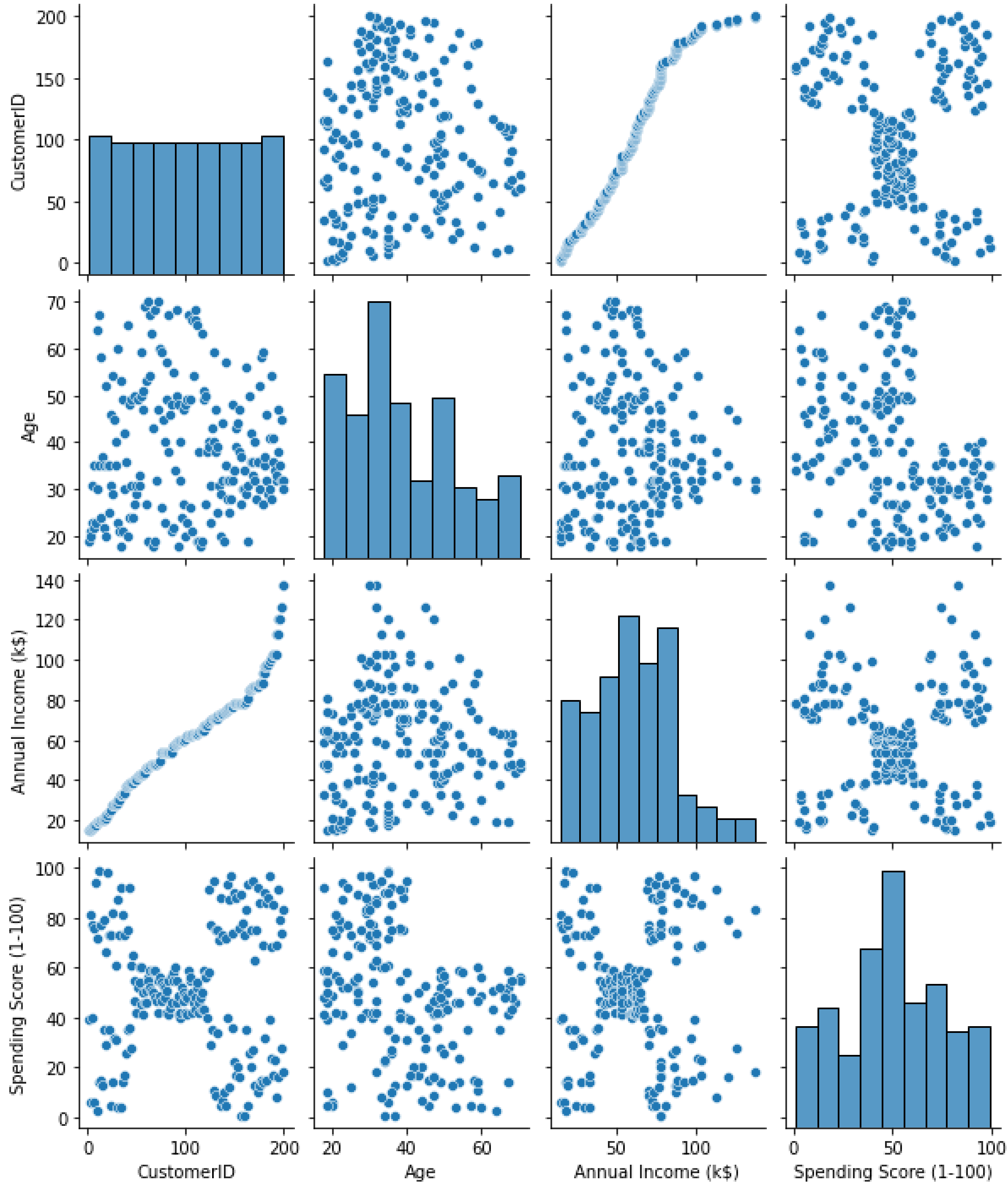

In [12]:
```
sns.violinplot(x='Gender',y='Spending  Score  (1-100)',data=data,  palette='coolwarm_r'
```

Out[12]:



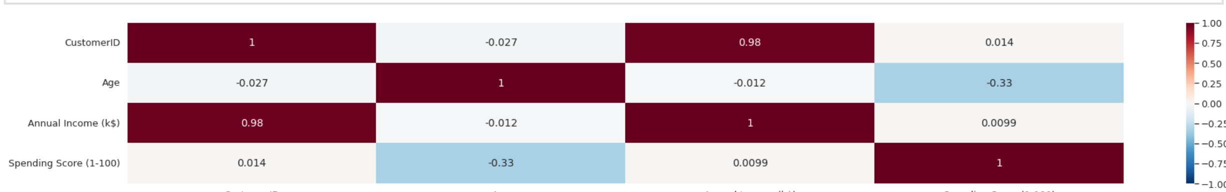

## Multi-Variate Analysis

In [13]:
```
sns.pairplot(
    data=data,
    aspect=.85);
```




In [14]:

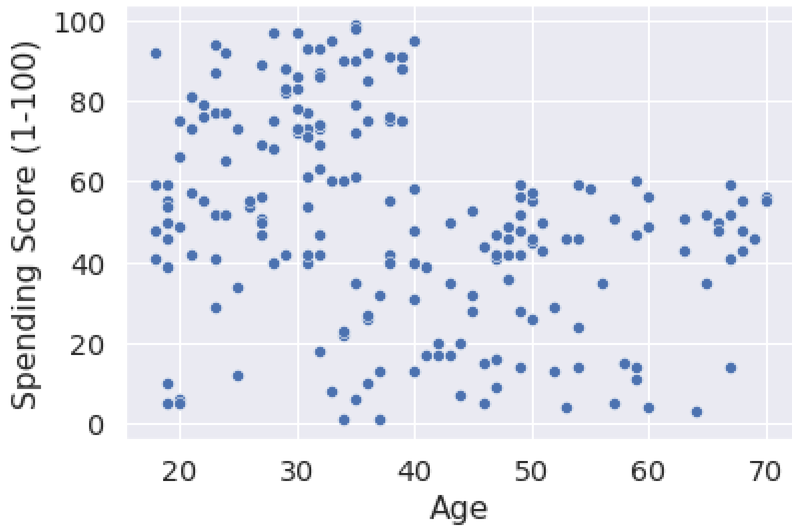
```python
sns.set(font_scale=1.15)
plt.figure(figsize=(30,4))
sns.heatmap(
    data.corr(),
    cmap='RdBu_r',
    annot=True,
    vmin=-1, vmax=1);
```



In [15]:

```
sns.set(font_scale=1.3)
sns.scatterplot(
  x='Age',y='Spending  Score  (1-100)',data=data)
plt.xlabel(
    'Age')
plt.ylabel(
    'Spending Score (1-100)')
```

Out[15]: Text(0, 0.5, 'Spending Score (1-100)')




## Descriptive statistics on the dataset

In [16]:

```
data['Spending  Score  (1-100)'].mean()
```

Out[16]: 50.2

In [17]:

```
data['Spending  Score  (1-100)'].median()
```

Out[17]: 50.0

In [18]:

```
data['Spending  Score  (1-100)'].std()
```

Out[18]: 25.823521668370173

In [19]:

```
data['Spending  Score  (1-100)'].value_counts()
```

Out[19]:

```
42    8
55    7
46    6
73    6
35    5
      ..
31    1
44    1
```

```
53    1
65    1
18    1
Name: Spending Score (1-100), Length: 84, dtype: int64
```

# Check for Missing values

In [20]:

```
data.isnull().sum().sum()
```

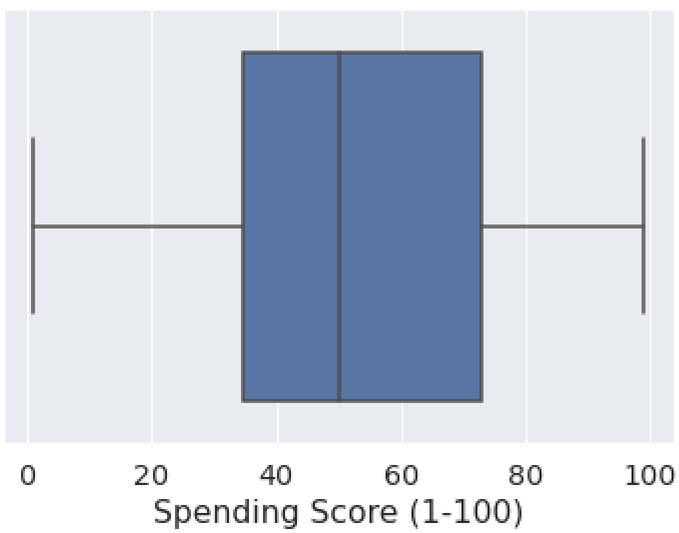
Out[20]: 0

# Find the outliers and replace them outliers

In [21]:

```
sns.boxplot(data['Spending  Score  (1-100)'],data=data)
```

Out[21]:




In [22]:

```
data['Spending  Score  (1-100)'].skew()
```

Out[22]: -0.047220201374263374

In [23]:

```
Q1=data['Spending  Score  (1-100)'].quantile(0.25)
Q3=data['Spending  Score  (1-100)'].quantile(0.75)
IQR=Q3-Q1
print(IQR)
```

```
38.25
```

In [24]:

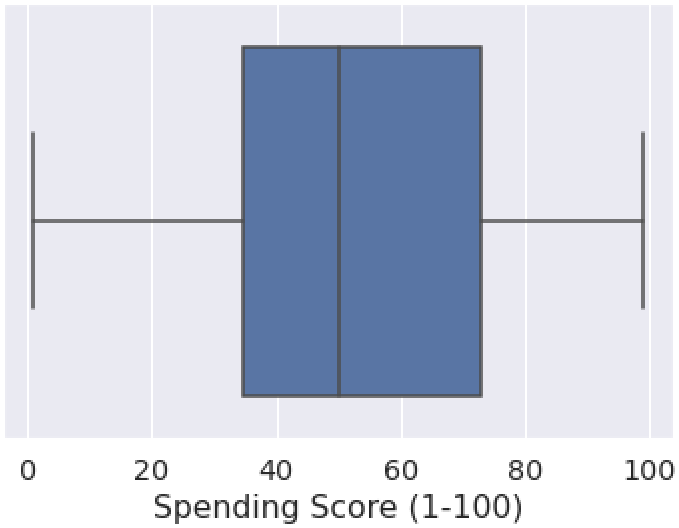
```
Q1=data['Spending  Score  (1-100)'].quantile(0.25)
Q3=data['Spending  Score  (1-100)'].quantile(0.75)
IQR=Q3-Q1
whisker_width = 1.5
lower_whisker = Q1 -(whisker_width*IQR)
```

```
upper_whisker = Q3 + (whisker_width*IQR)
data['Spending  Score  (1-100)']=np.where((data['Spending  Score  (1-100)'])>upper_whisk
```

In [25]:

```
sns.boxplot(data['Spending  Score  (1-100)'],data=data)
```

Out[25]:




# Check for Categorical columns and perform encoding

In [33]:

```
numeric_data  =  data.select_dtypes(include=[np.number])
categorical_data  =  data.select_dtypes(exclude=[np.number])
print("Number of numerical variables: ", numeric_data.shape[1])
print("Number of categorical variables: ", categorical_data.shape[1])
```

```
Number of numerical variables:  4
Number of categorical variables:  1
```

In [34]:

```
print("Number  of  categorical  variables:  ",  categorical_data.shape[1])
Categorical_variables = list(categorical_data.columns)
Categorical_variables
```

```
Number of categorical variables:  1
```

Out[34]: ['Gender']

In [36]:

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

Out[36]:

```
Female    112
Male       88
Name: Gender, dtype: int64
```

In [38]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
label  =  le.fit_transform(data['Gender'])
data["Gender"] = label
```

In [40]:
```
data['Gender'].value_counts()
```

Out[40]:
```
0    112
1     88
Name: Gender, dtype: int64
```

# Scaling the data

In [41]:
```
X = data.drop("Age",axis=1)
Y = data['Age']
```

In [42]:
```
from sklearn.preprocessing import StandardScaler
object= StandardScaler()
scale = object.fit_transform(X)
print(scale)
```

```
[[-1.7234121   1.12815215 -1.73899919 -0.43480148]
 [-1.70609137  1.12815215 -1.73899919  1.19570407]
 [-1.68877065 -0.88640526 -1.70082976 -1.71591298]
 [-1.67144992 -0.88640526 -1.70082976  1.04041783]
 [-1.6541292  -0.88640526 -1.66266033 -0.39597992]
 [-1.63680847 -0.88640526 -1.66266033  1.00159627]
 [-1.61948775 -0.88640526 -1.62449091 -1.71591298]
 [-1.60216702 -0.88640526 -1.62449091  1.70038436]
 [-1.5848463   1.12815215 -1.58632148 -1.83237767]
 [-1.56752558 -0.88640526 -1.58632148  0.84631002]
 [-1.55020485  1.12815215 -1.58632148 -1.4053405 ]
 [-1.53288413 -0.88640526 -1.58632148  1.89449216]
 [-1.5155634  -0.88640526 -1.54815205 -1.36651894]
 [-1.49824268 -0.88640526 -1.54815205  1.04041783]
 [-1.48092195  1.12815215 -1.54815205 -1.44416206]
 [-1.46360123  1.12815215 -1.54815205  1.11806095]
 [-1.4462805  -0.88640526 -1.50998262 -0.59008772]
 [-1.42895978  1.12815215 -1.50998262  0.61338066]
 [-1.41163905  1.12815215 -1.43364376 -0.82301709]
 [-1.39431833 -0.88640526 -1.43364376  1.8556706 ]
 [-1.3769976   1.12815215 -1.39547433 -0.59008772]
 [-1.35967688  1.12815215 -1.39547433  0.88513158]
 [-1.34235616 -0.88640526 -1.3573049  -1.75473454]
 [-1.32503543  1.12815215 -1.3573049   0.88513158]
 [-1.30771471 -0.88640526 -1.24279661 -1.4053405 ]
 [-1.29039398  1.12815215 -1.24279661  1.23452563]
 [-1.27307326 -0.88640526 -1.24279661 -0.7065524 ]
 [-1.25575253  1.12815215 -1.24279661  0.41927286]
 [-1.23843181 -0.88640526 -1.20462718 -0.74537397]
 [-1.22111108 -0.88640526 -1.20462718  1.42863343]
 [-1.20379036  1.12815215 -1.16645776 -1.7935561 ]
 [-1.18646963 -0.88640526 -1.16645776  0.88513158]
 [-1.16914891  1.12815215 -1.05194947 -1.7935561 ]
 [-1.15182818  1.12815215 -1.05194947  1.62274124]
 [-1.13450746 -0.88640526 -1.05194947 -1.4053405 ]
 [-1.11718674 -0.88640526 -1.05194947  1.19570407]
 [-1.09986601 -0.88640526 -1.01378004 -1.28887582]
 [-1.08254529 -0.88640526 -1.01378004  0.88513158]
```

```
 [-1.06522456 -0.88640526 -0.89927175 -0.93948177]
 [-1.04790384 -0.88640526 -0.89927175  0.96277471]
 [-1.03058311 -0.88640526 -0.86110232 -0.59008772]
 [-1.01326239  1.12815215 -0.86110232  1.62274124]
 [-0.99594166  1.12815215 -0.82293289 -0.55126616]
 [-0.97862094 -0.88640526 -0.82293289  0.41927286]
 [-0.96130021 -0.88640526 -0.82293289 -0.86183865]
 [-0.94397949 -0.88640526 -0.82293289  0.5745591 ]
 [-0.92665877 -0.88640526 -0.78476346  0.18634349]
 [-0.90933804 -0.88640526 -0.78476346 -0.12422899]
 [-0.89201732 -0.88640526 -0.78476346 -0.3183368 ]
 [-0.87469659 -0.88640526 -0.78476346 -0.3183368 ]
 [-0.85737587 -0.88640526 -0.70842461  0.06987881]
 [-0.84005514  1.12815215 -0.70842461  0.38045129]
 [-0.82273442 -0.88640526 -0.67025518  0.14752193]
 [-0.80541369  1.12815215 -0.67025518  0.38045129]
 [-0.78809297 -0.88640526 -0.67025518 -0.20187212]
 [-0.77077224  1.12815215 -0.67025518 -0.35715836]
 [-0.75345152 -0.88640526 -0.63208575 -0.00776431]
 [-0.73613079  1.12815215 -0.63208575 -0.16305055]
 [-0.71881007 -0.88640526 -0.55574689  0.03105725]
 [-0.70148935  1.12815215 -0.55574689 -0.16305055]
 [-0.68416862  1.12815215 -0.55574689  0.22516505]
 [-0.6668479   1.12815215 -0.55574689  0.18634349]
 [-0.64952717 -0.88640526 -0.51757746  0.06987881]
 [-0.63220645 -0.88640526 -0.51757746  0.34162973]
 [-0.61488572  1.12815215 -0.47940803  0.03105725]
 [-0.597565    1.12815215 -0.47940803  0.34162973]
 [-0.58024427 -0.88640526 -0.47940803 -0.00776431]
 [-0.56292355 -0.88640526 -0.47940803 -0.08540743]
 [-0.54560282  1.12815215 -0.47940803  0.34162973]
 [-0.5282821  -0.88640526 -0.47940803 -0.12422899]
 [-0.51096138  1.12815215 -0.4412386   0.18634349]
 [-0.49364065 -0.88640526 -0.4412386  -0.3183368 ]
 [-0.47631993 -0.88640526 -0.40306917 -0.04658587]
 [-0.4589992  -0.88640526 -0.40306917  0.22516505]
 [-0.44167848  1.12815215 -0.25039146 -0.12422899]
 [-0.42435775  1.12815215 -0.25039146  0.14752193]
 [-0.40703703 -0.88640526 -0.25039146  0.10870037]
 [-0.3897163   1.12815215 -0.25039146 -0.08540743]
 [-0.37239558 -0.88640526 -0.25039146  0.06987881]
 [-0.35507485 -0.88640526 -0.25039146 -0.3183368 ]
 [-0.33775413  1.12815215 -0.25039146  0.03105725]
 [-0.3204334   1.12815215 -0.25039146  0.18634349]
 [-0.30311268  1.12815215 -0.25039146 -0.35715836]
 [-0.28579196 -0.88640526 -0.25039146 -0.24069368]
 [-0.26847123 -0.88640526 -0.25039146  0.26398661]
 [-0.25115051  1.12815215 -0.25039146 -0.16305055]
 [-0.23382978 -0.88640526 -0.13588317  0.30280817]
 [-0.21650906 -0.88640526 -0.13588317  0.18634349]
 [-0.19918833 -0.88640526 -0.09771374  0.38045129]
 [-0.18186761 -0.88640526 -0.09771374 -0.16305055]
 [-0.16454688 -0.88640526 -0.05954431  0.18634349]
 [-0.14722616  1.12815215 -0.05954431 -0.35715836]
 [-0.12990543  1.12815215 -0.02137488 -0.04658587]
 [-0.11258471 -0.88640526 -0.02137488 -0.39597992]
 [-0.09526399 -0.88640526 -0.02137488 -0.3183368 ]
 [-0.07794326  1.12815215 -0.02137488  0.06987881]
 [-0.06062254 -0.88640526 -0.02137488 -0.12422899]
 [-0.04330181 -0.88640526 -0.02137488 -0.00776431]
```

```
[-0.02598109  1.12815215  0.01679455 -0.3183368 ]
[-0.00866036  1.12815215  0.01679455 -0.04658587]
[ 0.00866036 -0.88640526  0.05496398 -0.35715836]
[ 0.02598109 -0.88640526  0.05496398 -0.08540743]
[ 0.04330181  1.12815215  0.05496398  0.34162973]
[ 0.06062254  1.12815215  0.05496398  0.18634349]
[ 0.07794326  1.12815215  0.05496398  0.22516505]
[ 0.09526399 -0.88640526  0.05496398 -0.3183368 ]
[ 0.11258471 -0.88640526  0.09313341 -0.00776431]
[ 0.12990543  1.12815215  0.09313341 -0.16305055]
[ 0.14722616  1.12815215  0.09313341 -0.27951524]
[ 0.16454688  1.12815215  0.09313341 -0.08540743]
[ 0.18186761  1.12815215  0.09313341  0.06987881]
[ 0.19918833 -0.88640526  0.09313341  0.14752193]
[ 0.21650906 -0.88640526  0.13130284 -0.3183368 ]
[ 0.23382978  1.12815215  0.13130284 -0.16305055]
[ 0.25115051 -0.88640526  0.16947227 -0.08540743]
[ 0.26847123 -0.88640526  0.16947227 -0.00776431]
[ 0.28579196 -0.88640526  0.16947227 -0.27951524]
[ 0.30311268 -0.88640526  0.16947227  0.34162973]
[ 0.3204334  -0.88640526  0.24581112 -0.27951524]
[ 0.33775413 -0.88640526  0.24581112  0.26398661]
[ 0.35507485  1.12815215  0.24581112  0.22516505]
[ 0.37239558 -0.88640526  0.24581112 -0.39597992]
[ 0.3897163  -0.88640526  0.32214998  0.30280817]
[ 0.40703703  1.12815215  0.32214998  1.58391968]
[ 0.42435775 -0.88640526  0.36031941 -0.82301709]
[ 0.44167848 -0.88640526  0.36031941  1.04041783]
[ 0.4589992   1.12815215  0.39848884 -0.59008772]
[ 0.47631993  1.12815215  0.39848884  1.73920592]
[ 0.49364065  1.12815215  0.39848884 -1.52180518]
[ 0.51096138  1.12815215  0.39848884  0.96277471]
[ 0.5282821   1.12815215  0.39848884 -1.5994483 ]
[ 0.54560282  1.12815215  0.39848884  0.96277471]
[ 0.56292355 -0.88640526  0.43665827 -0.62890928]
[ 0.58024427 -0.88640526  0.43665827  0.80748846]
[ 0.597565    1.12815215  0.4748277  -1.75473454]
[ 0.61488572 -0.88640526  0.4748277   1.46745499]
[ 0.63220645 -0.88640526  0.4748277  -1.67709142]
[ 0.64952717  1.12815215  0.4748277   0.88513158]
[ 0.6668479   1.12815215  0.51299713 -1.56062674]
[ 0.68416862 -0.88640526  0.51299713  0.84631002]
[ 0.70148935 -0.88640526  0.55116656 -1.75473454]
[ 0.71881007  1.12815215  0.55116656  1.6615628 ]
[ 0.73613079 -0.88640526  0.58933599 -0.39597992]
[ 0.75345152 -0.88640526  0.58933599  1.42863343]
[ 0.77077224  1.12815215  0.62750542 -1.48298362]
[ 0.78809297  1.12815215  0.62750542  1.81684904]
[ 0.80541369  1.12815215  0.62750542 -0.55126616]
[ 0.82273442 -0.88640526  0.62750542  0.92395314]
[ 0.84005514 -0.88640526  0.66567484 -1.09476801]
[ 0.85737587  1.12815215  0.66567484  1.54509812]
[ 0.87469659  1.12815215  0.66567484 -1.28887582]
[ 0.89201732  1.12815215  0.66567484  1.46745499]
[ 0.90933804 -0.88640526  0.66567484 -1.17241113]
[ 0.92665877 -0.88640526  0.66567484  1.00159627]
[ 0.94397949 -0.88640526  0.66567484 -1.32769738]
[ 0.96130021 -0.88640526  0.66567484  1.50627656]
[ 0.97862094  1.12815215  0.66567484 -1.91002079]
[ 0.99594166 -0.88640526  0.66567484  1.07923939]
```

```
[ 1.01326239  1.12815215  0.66567484 -1.91002079]
[ 1.03058311 -0.88640526  0.66567484  0.88513158]
[ 1.04790384 -0.88640526  0.70384427 -0.59008772]
[ 1.06522456 -0.88640526  0.70384427  1.27334719]
[ 1.08254529  1.12815215  0.78018313 -1.75473454]
[ 1.09986601 -0.88640526  0.78018313  1.6615628 ]
[ 1.11718674  1.12815215  0.93286085 -0.93948177]
[ 1.13450746 -0.88640526  0.93286085  0.96277471]
[ 1.15182818  1.12815215  0.97103028 -1.17241113]
[ 1.16914891 -0.88640526  0.97103028  1.73920592]
[ 1.18646963 -0.88640526  1.00919971 -0.90066021]
[ 1.20379036  1.12815215  1.00919971  0.49691598]
[ 1.22111108  1.12815215  1.00919971 -1.44416206]
[ 1.23843181  1.12815215  1.00919971  0.96277471]
[ 1.25575253  1.12815215  1.00919971 -1.56062674]
[ 1.27307326  1.12815215  1.00919971  1.62274124]
[ 1.29039398 -0.88640526  1.04736914 -1.44416206]
[ 1.30771471 -0.88640526  1.04736914  1.38981187]
[ 1.32503543  1.12815215  1.04736914 -1.36651894]
[ 1.34235616  1.12815215  1.04736914  0.72984534]
[ 1.35967688  1.12815215  1.23821628 -1.4053405 ]
[ 1.3769976   1.12815215  1.23821628  1.54509812]
[ 1.39431833 -0.88640526  1.390894   -0.7065524 ]
[ 1.41163905 -0.88640526  1.390894    1.38981187]
[ 1.42895978  1.12815215  1.42906343 -1.36651894]
[ 1.4462805  -0.88640526  1.42906343  1.46745499]
[ 1.46360123 -0.88640526  1.46723286 -0.43480148]
[ 1.48092195  1.12815215  1.46723286  1.81684904]
[ 1.49824268 -0.88640526  1.54357172 -1.01712489]
[ 1.5155634   1.12815215  1.54357172  0.69102378]
[ 1.53288413 -0.88640526  1.61991057 -1.28887582]
[ 1.55020485 -0.88640526  1.61991057  1.35099031]
[ 1.56752558 -0.88640526  1.61991057 -1.05594645]
[ 1.5848463  -0.88640526  1.61991057  0.72984534]
[ 1.60216702  1.12815215  2.00160487 -1.63826986]
[ 1.61948775 -0.88640526  2.00160487  1.58391968]
[ 1.63680847 -0.88640526  2.26879087 -1.32769738]
[ 1.6541292  -0.88640526  2.26879087  1.11806095]
[ 1.67144992 -0.88640526  2.49780745 -0.86183865]
[ 1.68877065  1.12815215  2.49780745  0.92395314]
[ 1.70609137  1.12815215  2.91767117 -1.25005425]
[ 1.7234121   1.12815215  2.91767117  1.27334719]]
```

In [43]:

```
X_scaled  = pd.DataFrame(scale, columns = X.columns)
X_scaled
```

Out[43]:

| | CustomerID | Gender | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **0** | -1.723412 | 1.128152 | -1.738999 | -0.434801 |
| **1** | -1.706091 | 1.128152 | -1.738999 | 1.195704 |
| **2** | -1.688771 | -0.886405 | -1.700830 | -1.715913 |
| **3** | -1.671450 | -0.886405 | -1.700830 | 1.040418 |
| **4** | -1.654129 | -0.886405 | -1.662660 | -0.395980 |
| **...** | ... | ... | ... | ... |

| | CustomerID | Gender | Annual Income (k$) | Spending Score (1-100) |
|---|---|---|---|---|
| **195** | 1.654129 | -0.886405 | 2.268791 | 1.118061 |
| **196** | 1.671450 | -0.886405 | 2.497807 | -0.861839 |
| **197** | 1.688771 | 1.128152 | 2.497807 | 0.923953 |
| **198** | 1.706091 | 1.128152 | 2.917671 | -1.250054 |
| **199** | 1.723412 | 1.128152 | 2.917671 | 1.273347 |

200 rows × 4 columns

# Split the data into training and testing dataset

In [44]:

```
#train test split
from sklearn.model_selection import train_test_split
# split the dataset
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, Y, test_size=0.20, ran
```

In [45]:

```
X_train.shape
```

Out[45]: (160, 4)

In [46]:

```
X_test.shape
```

Out[46]: (40, 4)

In [47]:

```
Y_train.shape
```

Out[47]: (160,)

In [48]:

```
Y_test.shape
```

Out[48]: (40,)

# Split the data into dependent and independent variables

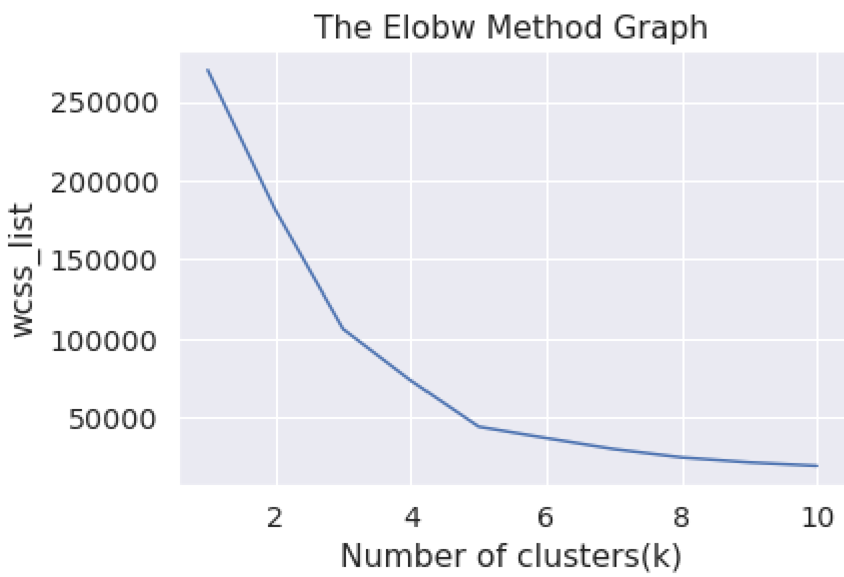
In [50]:

```
x = data.iloc[:, [3, 4]].values
```

# Build the model using any of the clustering algorithms

In [51]:
```
#finding optimal number of clusters using the elbow method
from sklearn.cluster import KMeans
wcss_list= []  #Initializing the list for the values of WCSS

#Using for loop for iterations from 1 to 10.
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss_list)
plt.title('The Elobw Method Graph')
plt.xlabel('Number  of  clusters(k)')
plt.ylabel('wcss_list')
plt.show()
```

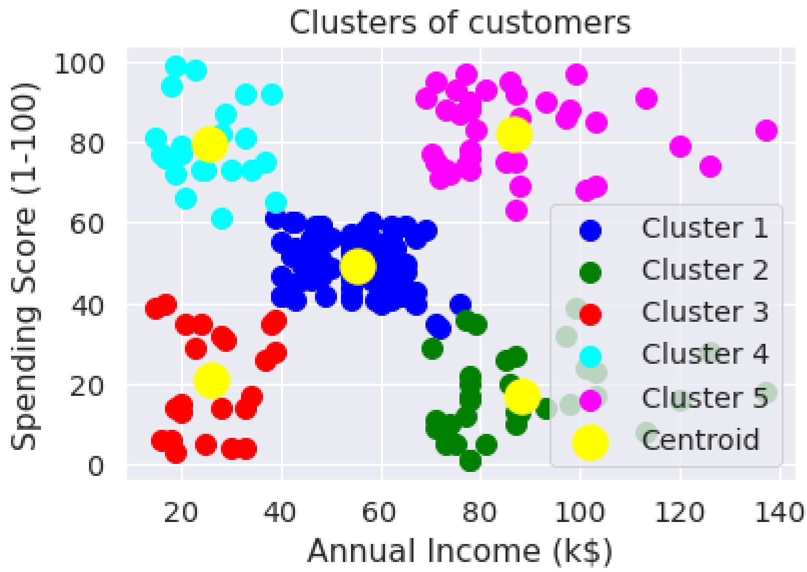



# Train and test the model

In [52]:
```
#training the K-means model on a dataset
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
```

# Add the cluster data with the primary dataset

In [53]:
```
#visulaizing the clusters
plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label =
plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label
plt.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = '
plt.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label =
plt.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', labe
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```




# Measure the performance using evaluation metrics

In [62]:

```
from sklearn.metrics import silhouette_score,calinski_harabasz_score,davies_bouldin_
sil_scores = []
calinski_score = []
davies_score = []
sil_scores.append(silhouette_score(x, y_predict))
calinski_score.append(calinski_harabasz_score(x, y_predict))
davies_score.append(davies_bouldin_score(x, y_predict))
```

In [63]:

```
print("Silhouette Coefficient: %0.3f" % silhouette_score(x, y_predict))
print("Calinski-Harabasz Index: %0.3f" % calinski_harabasz_score(x, y_predict))
print("Davies-Bouldin Index: %0.3f" % davies_bouldin_score(x, y_predict))
```

```
Silhouette Coefficient: 0.554
Calinski-Harabasz Index: 247.359
Davies-Bouldin Index: 0.573
```