Problem Statment

- Customer Segmentation Analysis:

Customer segmentation is the problem of uncovering information about a firm's customer base, based on their interactions with the business. In most cases this interaction is in terms of their purchase behavior and patterns. We explore some of the ways in which this can be used.

You own the mall and want to understand the customers like who can be easily converge [Target Customers] so that the sense can be given to marketing team and plan the strategy accordingly.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_csv("Mall_Customers.csv")
data.head(5)
```

Out[2]:

	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 [3]:
```
data.shape
```

Out[3]:
```
(200, 5)
```

In [4]:
```
data.describe()
```

Out[4]:

	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 [5]:
```
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
```

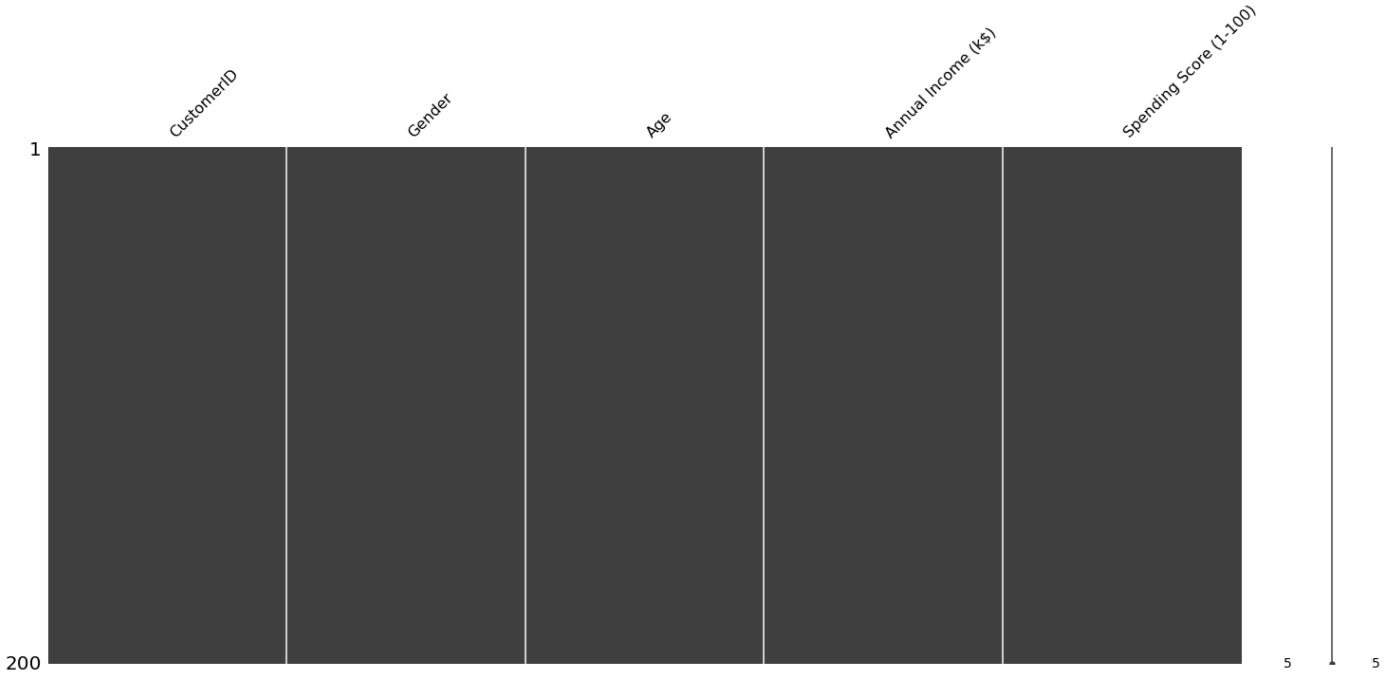
In [6]:
```
data.isnull().sum()
```

Out[6]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [7]:
```
import missingno as msno
```

In [8]:
```
msno.matrix(data)
```

Out[8]:
```
<AxesSubplot: >
```



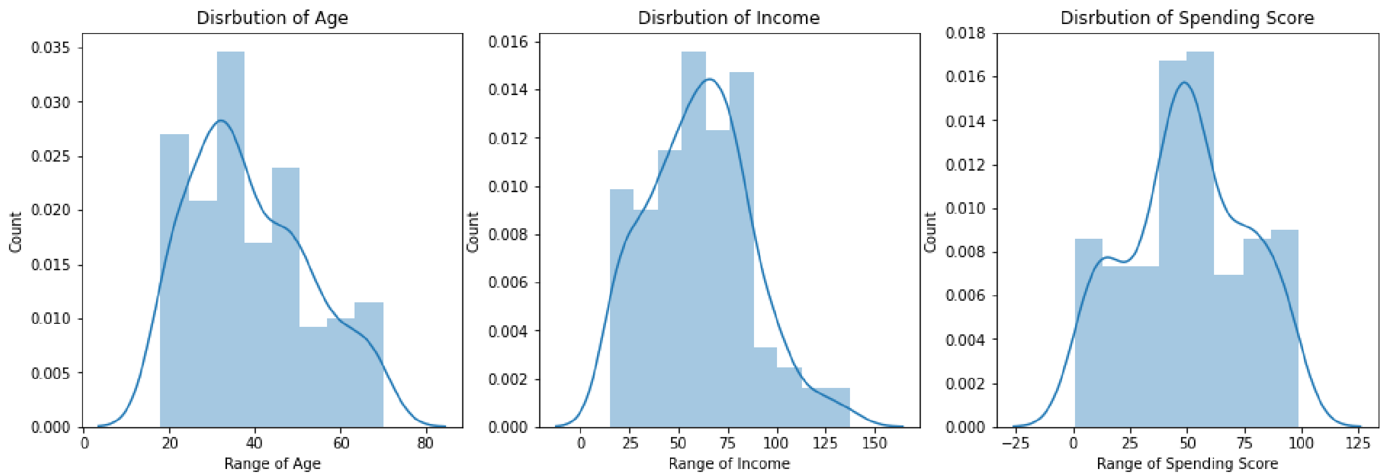
Data Visuliation & EDA

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

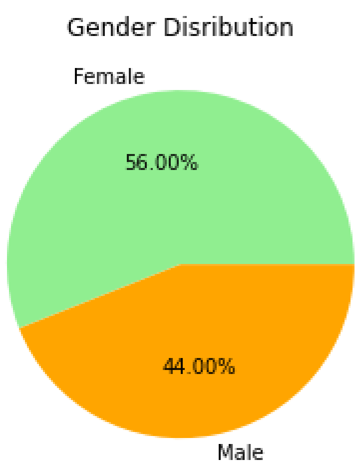
Out[9]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [10]:
```
plt.figure(figsize = (16,5))

plt.subplot(1, 3, 1)
sns.distplot(data['Age'])
plt.title("Disrbution of Age")
plt.xlabel("Range of Age")
plt.ylabel("Count")

plt.subplot(1, 3,2)
sns.distplot(data['Annual Income (k$)'])
plt.title("Disrbution of Income")
plt.xlabel("Range of Income")
plt.ylabel("Count")

plt.subplot(1, 3, 3)
sns.distplot(data['Spending Score (1-100)'])
plt.title("Disrbution of Spending Score")
plt.xlabel("Range of Spending Score")
plt.ylabel("Count")
plt.show()
```

```
C:\Application\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: 
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Application\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: 
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Application\anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: 
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

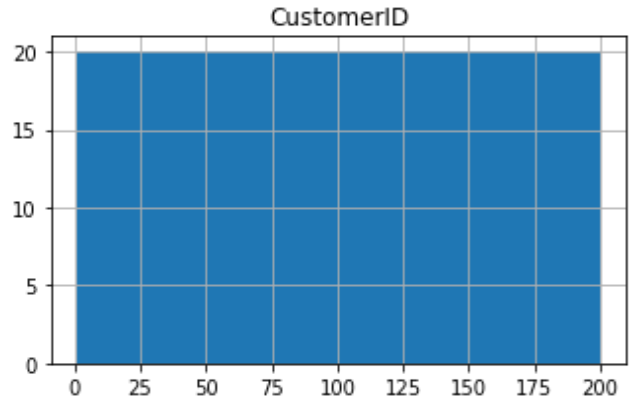


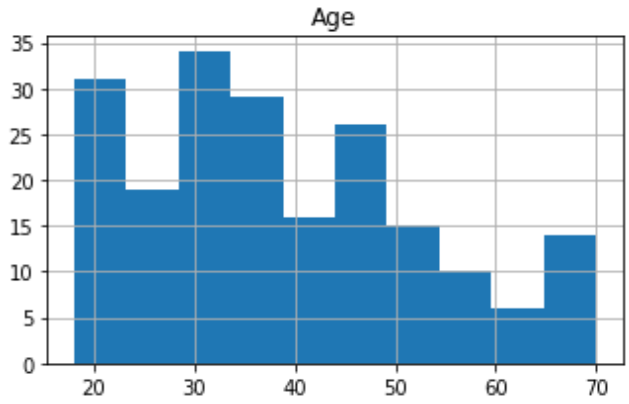
In [11]:
```
df = data.groupby('Gender').size()

df.plot(kind='pie', subplots = True,
       colors = ['lightgreen', 'orange'],
       explode = [0, 0.001],
       labels = ['Female', 'Male'],
       autopct = '%.2f%%' )
plt.title("Gender Disribution")
plt.ylabel("")
plt.show()
```

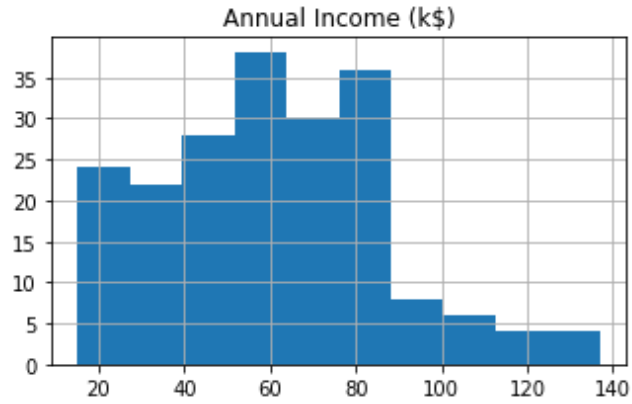


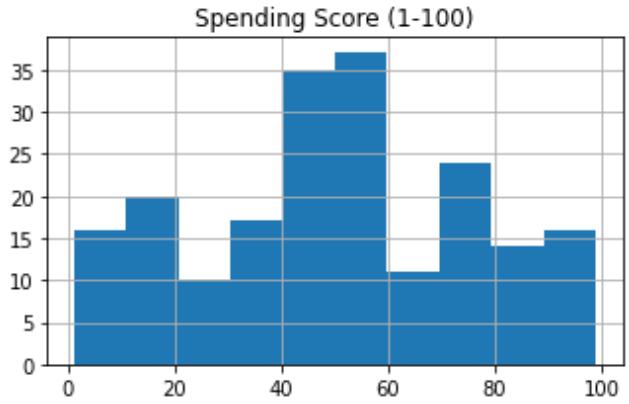
In [12]:
```
data.hist(figsize = (12, 7))
plt.show()
```

Loading [MathJax]/extensions/Safe.js

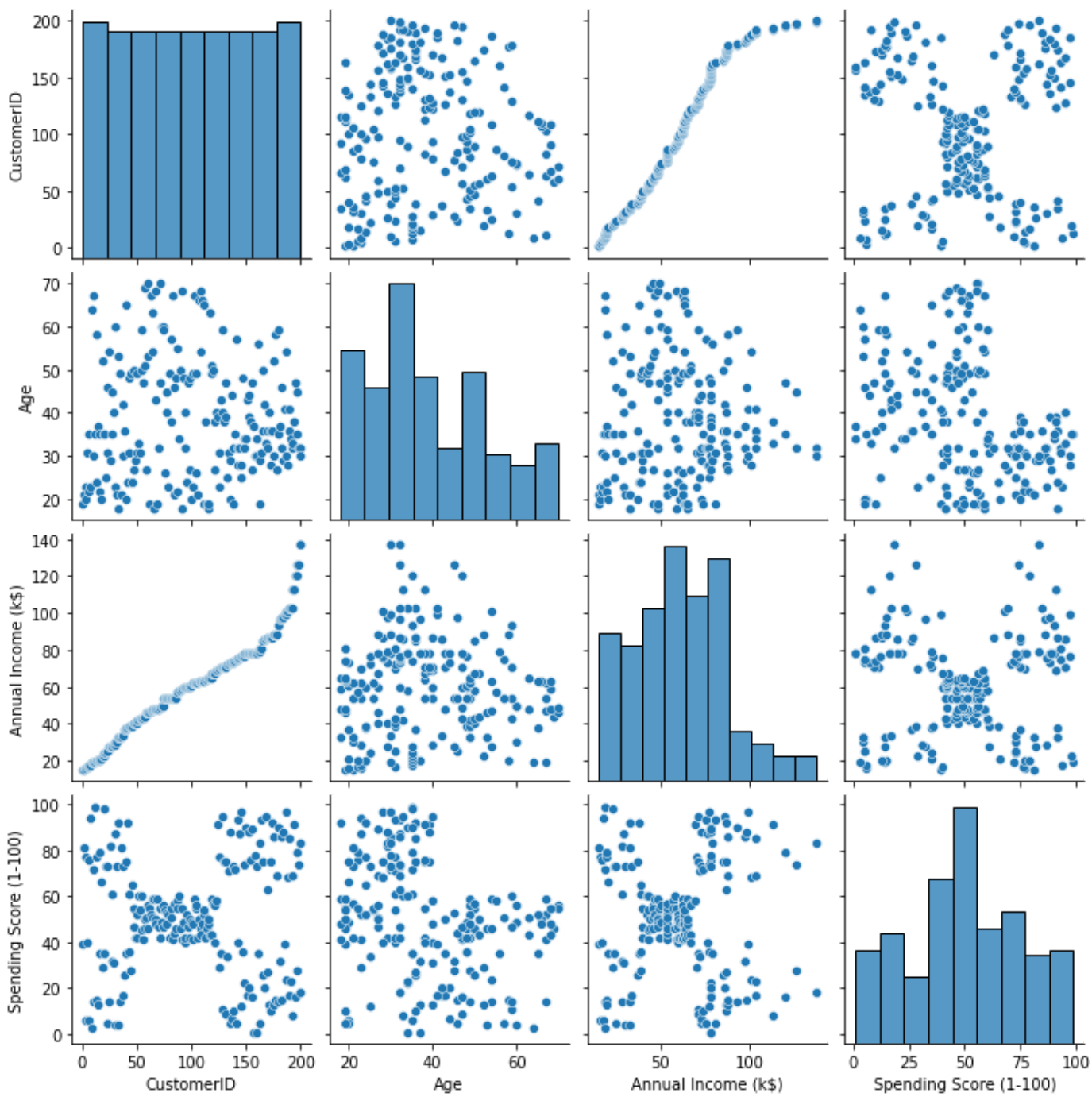






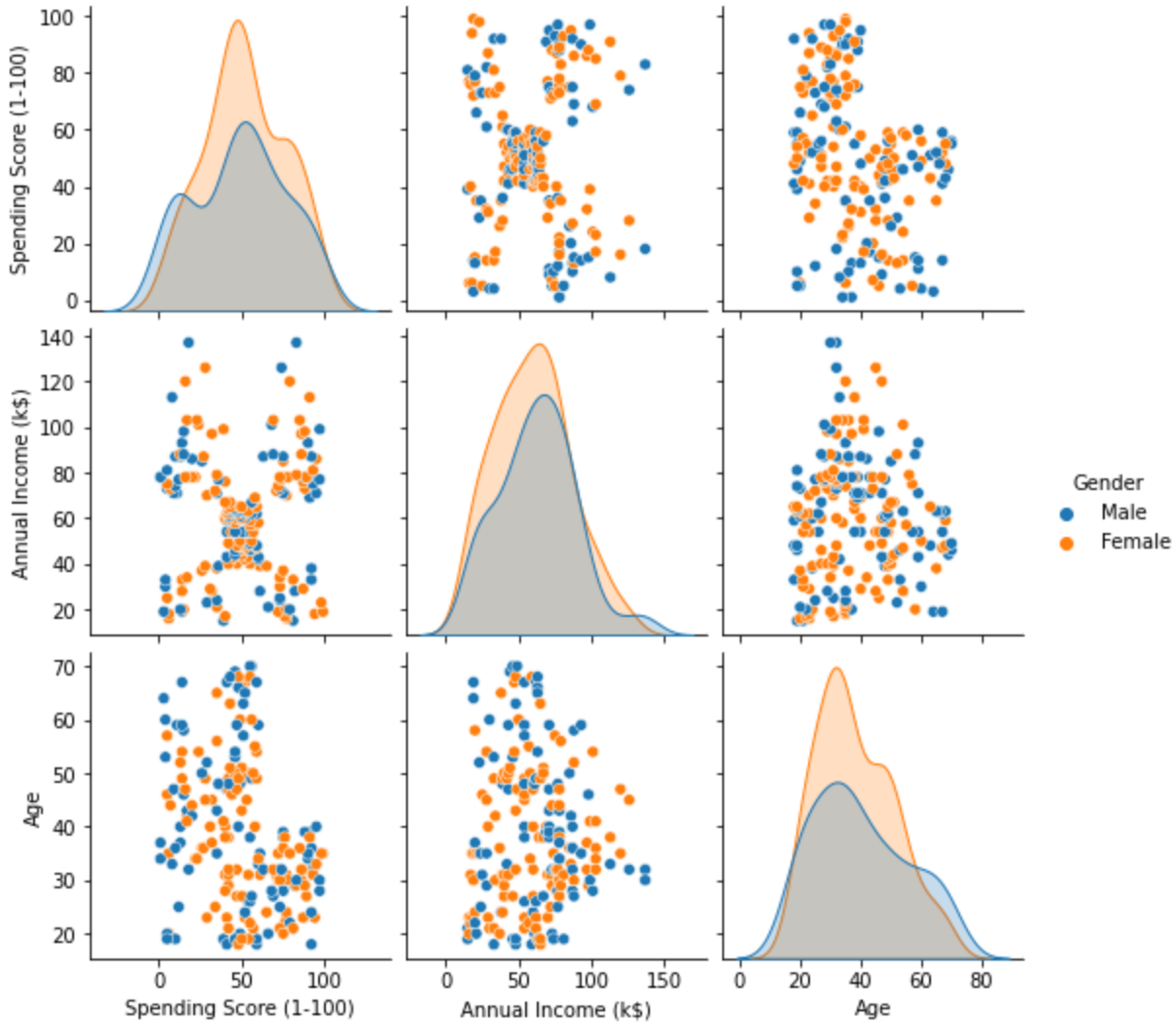


In [13]:
```
sns.pairplot(data)
plt.show()
```



In [14]:
```
sns.pairplot(data, vars = ['Spending Score (1-100)',
                           'Annual Income (k$)', 'Age'], hue = 'Gender')
```

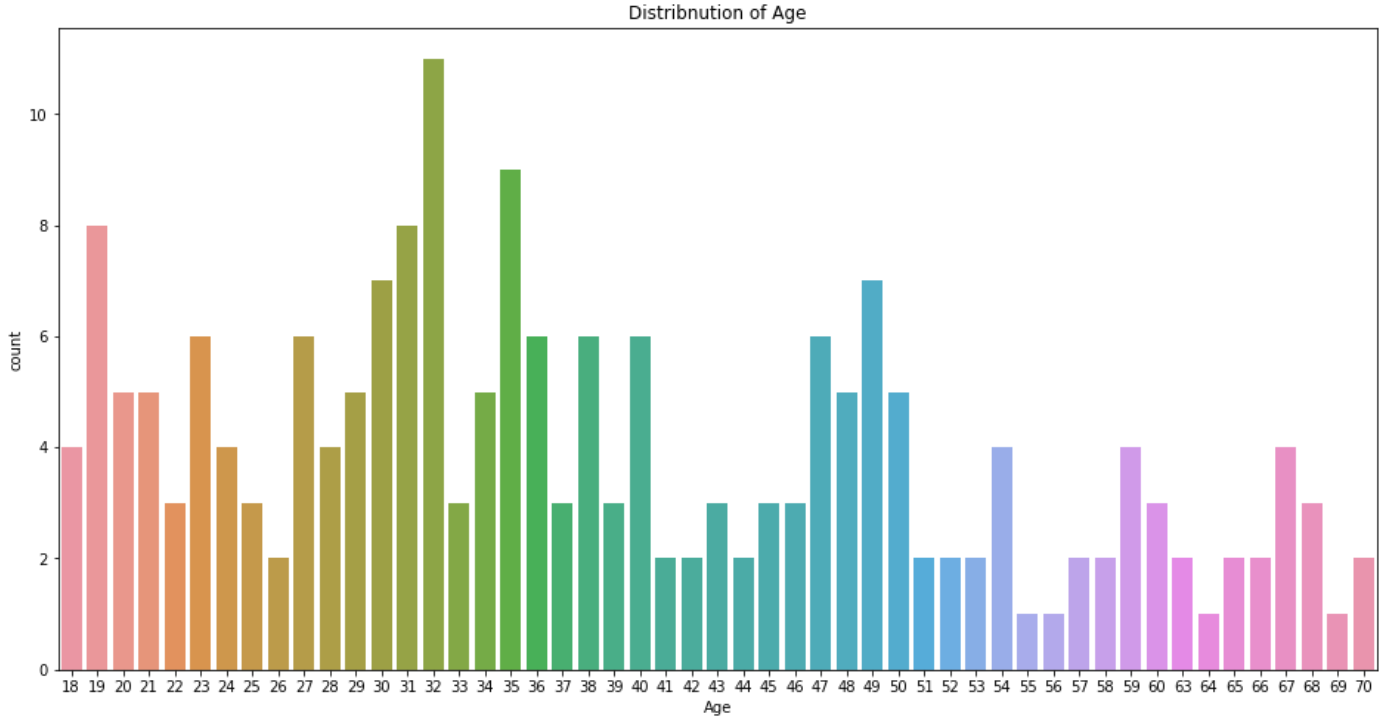
Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x16dfc453970>
```



In [15]:
```
plt.figure(figsize = (16,8))
sns.countplot(data['Age'])
plt.title("Distribnution of Age")
```

```
C:\Application\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

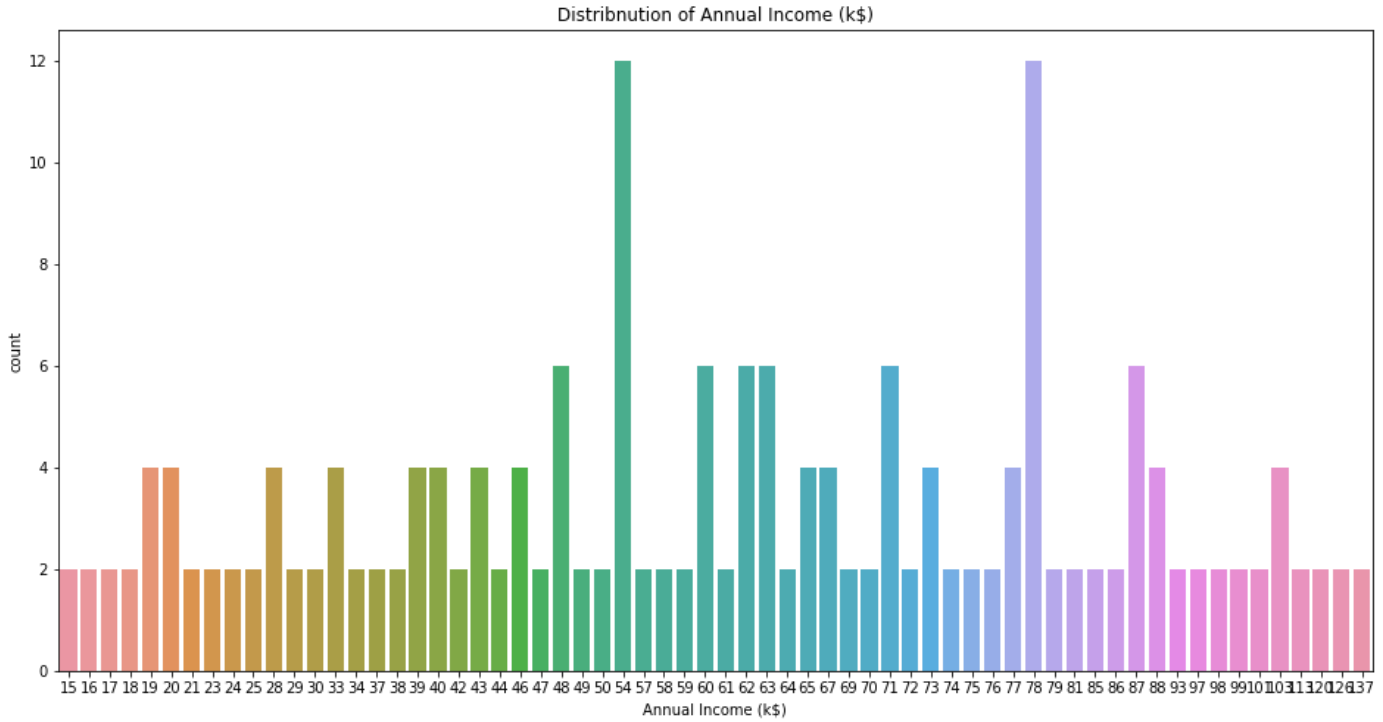
Out[15]:
```
Text(0.5, 1.0, 'Distribnution of Age')
```



In [16]:
```
plt.figure(figsize = (16,8))
sns.countplot(data['Annual Income (k$)'])
plt.title("Distribnution of Annual Income (k$)")
```

```
C:\Application\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[16]:
```
Text(0.5, 1.0, 'Distribnution of Annual Income (k$)')
```

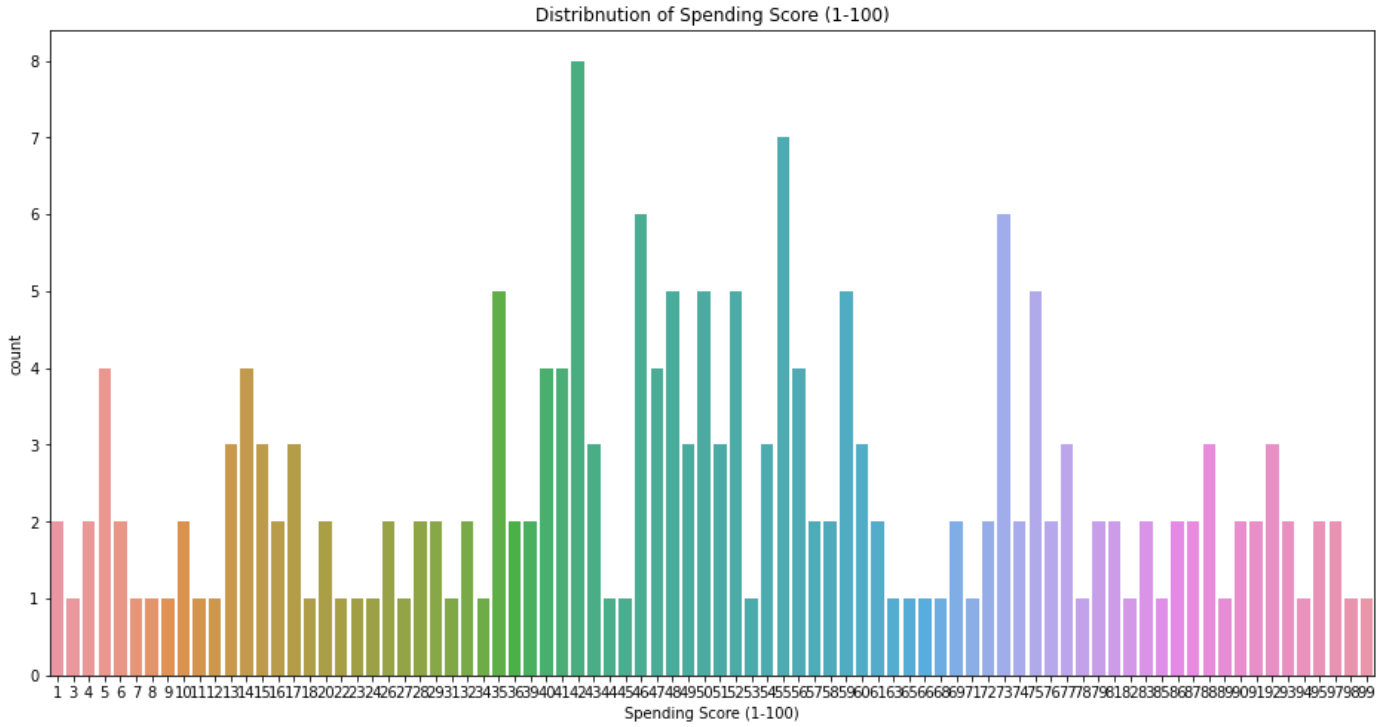


In [17]:
```
plt.figure(figsize = (16,8))
sns.countplot(data['Spending Score (1-100)'])
plt.title("Distribnution of Spending Score (1-100)")
```

```
C:\Application\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[17]:

```
Text(0.5, 1.0, 'Distribnution of Spending Score (1-100)')
```



In [18]:

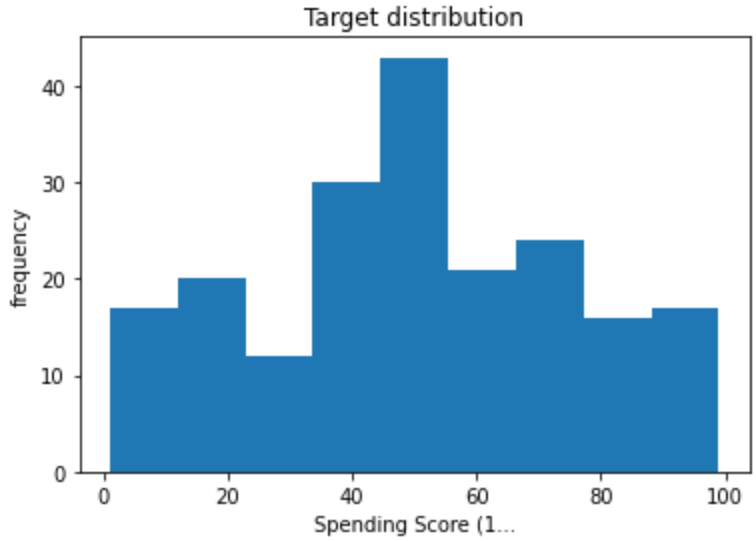
```
!pip install dabl
```

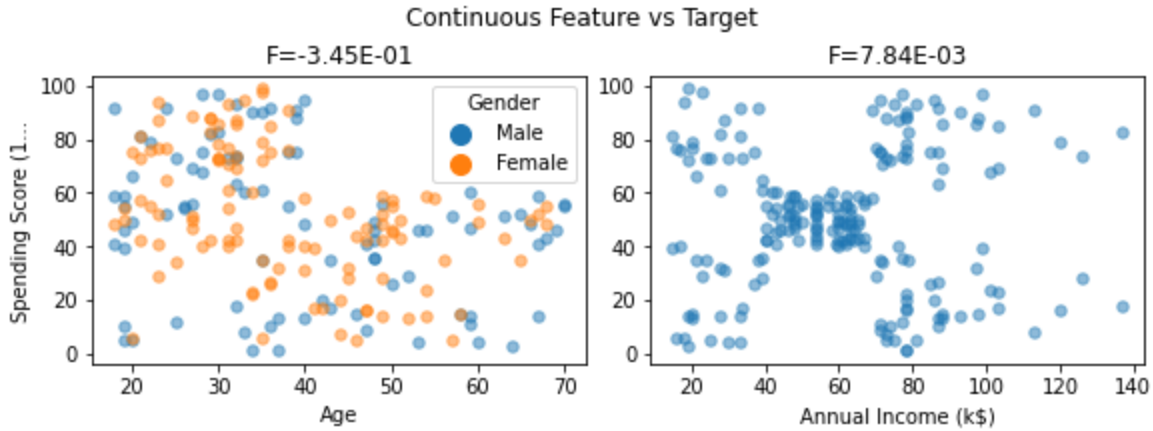
```
Requirement already satisfied: dabl in c:\application\anaconda\lib\site-packages (0.2.4)
Requirement already satisfied: scikit-learn>=1.0 in c:\application\anaconda\lib\site-pac
kages (from dabl) (1.2.0)
Requirement already satisfied: scipy in c:\application\anaconda\lib\site-packages (from
dabl) (1.6.2)
Requirement already satisfied: pandas in c:\application\anaconda\lib\site-packages (from
dabl) (1.2.4)
Requirement already satisfied: matplotlib>=3.4 in c:\application\anaconda\lib\site-packa
ges (from dabl) (3.6.2)
Requirement already satisfied: numpy in c:\application\anaconda\lib\site-packages (from
dabl) (1.19.5)
Requirement already satisfied: seaborn in c:\application\anaconda\lib\site-packages (fro
m dabl) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in c:\application\anaconda\lib\site-package
s (from matplotlib>=3.4->dabl) (8.2.0)
Requirement already satisfied: packaging>=20.0 in c:\application\anaconda\lib\site-packa
ges (from matplotlib>=3.4->dabl) (20.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\application\anaconda\lib\site-
packages (from matplotlib>=3.4->dabl) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\application\anaconda\lib\site-packages
(from matplotlib>=3.4->dabl) (0.10.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\application\anaconda\lib\site-pac
kages (from matplotlib>=3.4->dabl) (4.38.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\application\anaconda\lib\site-pac
kages (from matplotlib>=3.4->dabl) (1.3.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\application\anaconda\lib\site-pack
ages (from matplotlib>=3.4->dabl) (1.0.6)
Requirement already satisfied: pyparsing>=2.2.1 in c:\application\anaconda\lib\site-pack
ages (from matplotlib>=3.4->dabl) (2.4.7)
Requirement already satisfied: six in c:\application\anaconda\lib\site-packages (from cy
cler>=0.10->matplotlib>=3.4->dabl) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\application\anaconda\lib\site-
packages (from scikit-learn>=1.0->dabl) (2.1.0)
Requirement already satisfied: joblib>=1.1.1 in c:\application\anaconda\lib\site-package
s (from scikit-learn>=1.0->dabl) (1.2.0)
Requirement already satisfied: pytz>=2017.3 in c:\application\anaconda\lib\site-packages
(from pandas->dabl) (2021.1)
```

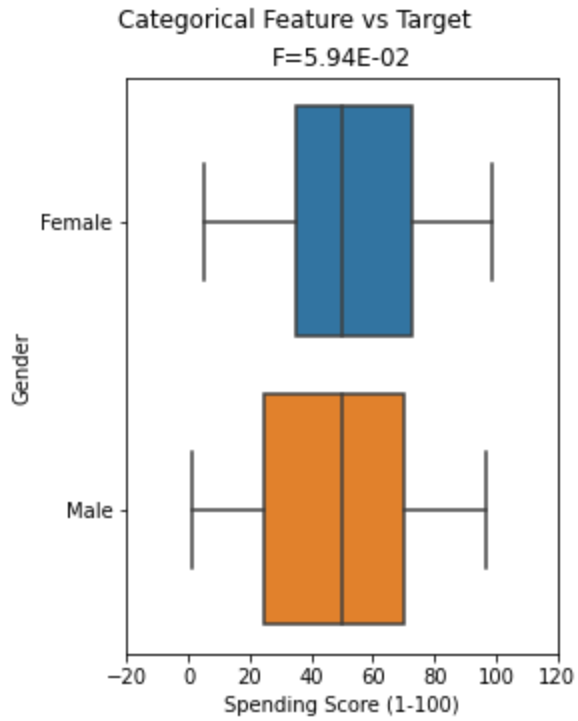
In [19]:
```
import dabl
```

In [20]:
```
dabl.plot(data, target_col = 'Spending Score (1-100)')
```

Out[20]:
```
Target looks like regression
[<AxesSubplot: title={'center': 'Target distribution'}, xlabel='Spending Score (1...', y
label='frequency'>,
 array([[<AxesSubplot: title={'center': 'F=-3.45E-01'}, xlabel='Age', ylabel='Spending S
core (1...'>,
         <AxesSubplot: title={'center': 'F=7.84E-03'}, xlabel='Annual Income (k$)'>]],
       dtype=object),
 array([[<AxesSubplot: title={'center': 'F=5.94E-02'}, xlabel='Spending Score (1-100)',
ylabel='Gender'>]],
       dtype=object)]
```



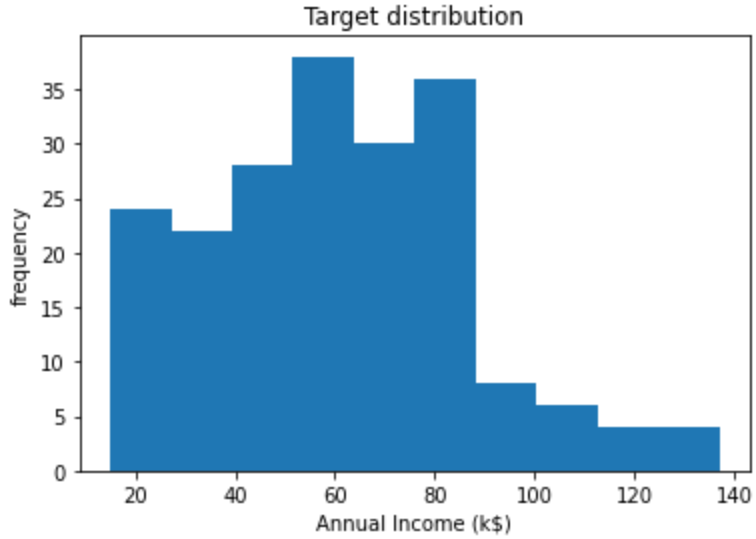


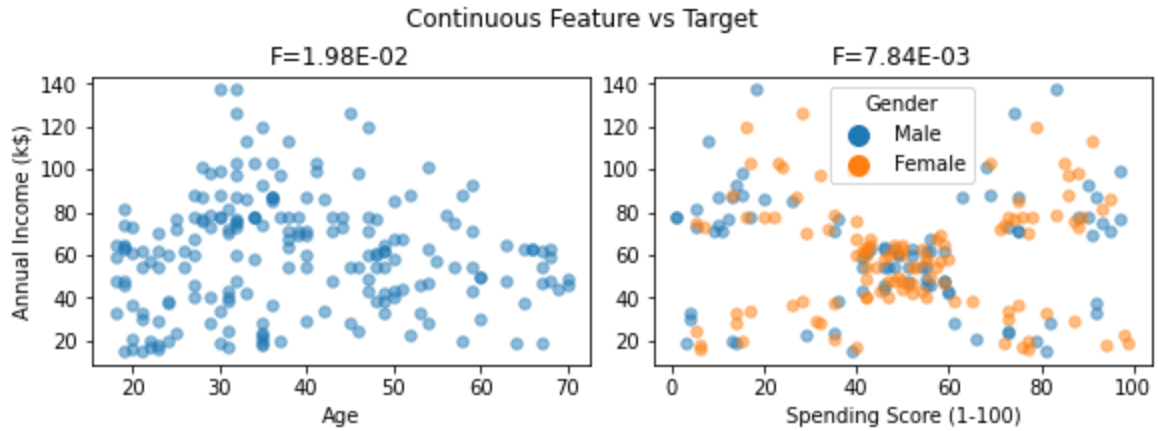


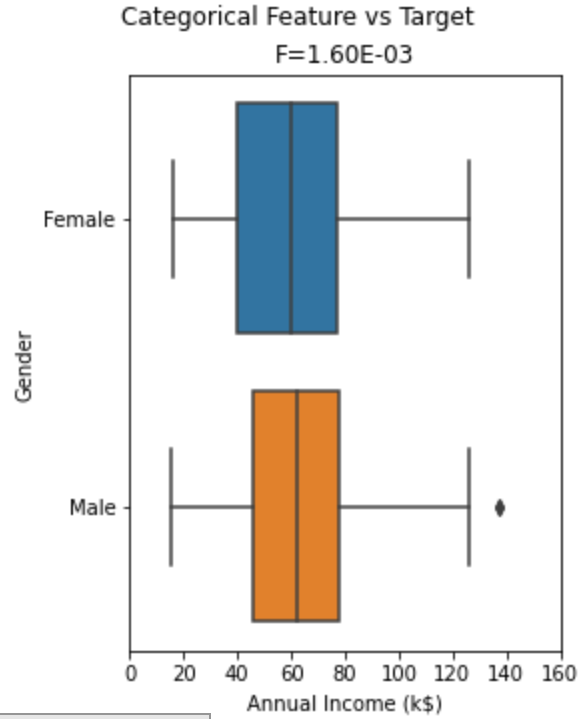
In [21]:
```
dabl.plot(data, target_col = 'Annual Income (k$)')
```

```
Target looks like regression
```

Out[21]:
```
[<AxesSubplot: title={'center': 'Target distribution'}, xlabel='Annual Income (k$)', yla
bel='frequency'>,
 array([[<AxesSubplot: title={'center': 'F=1.98E-02'}, xlabel='Age', ylabel='Annual Inco
me (k$)'>,
         <AxesSubplot: title={'center': 'F=7.84E-03'}, xlabel='Spending Score (1-10
0)'>]],
       dtype=object),
 array([[<AxesSubplot: title={'center': 'F=1.60E-03'}, xlabel='Annual Income (k$)', ylab
el='Gender'>]],
       dtype=object)]
```







In [22]:
```
sns.heatmap(data.corr(), cmap = 'Wistia', annot = True)
plt.show()
```

In [25]:
```
data.columns
```

Out[25]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [28]:
```
plt.figure(figsize = (15, 7))
for gender in ['Male', 'Female']:
    plt.scatter(x = 'Age', y = 'Annual Income (k$)',
                data = data[data['Gender'] == gender],
                s = 200, alpha = 0.5,
                label = gender)
plt.xlabel("Age")
plt.ylabel("Annual Income")
plt.title("Age vs Annual Income")
plt.legend()
plt.show()
```

In [29]:
```
plt.figure(1 , figsize = (15 , 6))
for gender in ['Male' , 'Female']:
    plt.scatter(x = 'Annual Income (k$)' , y = 'Spending Score (1-100)' ,
                data = data[data['Gender'] == gender] ,
                s = 200 , alpha = 0.5 , label = gender)
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Annual Income (k$) vsSpending Score (1-100)')
plt.legend()
plt.show()
```

K-Means Clustering

In [36]:
```
data.columns
```

Out[36]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [37]:
```
x = data.iloc[:, [3,4]].values
```

In [38]:
```
x
```

Out[38]:
```
array([[ 15,  39],
       [ 15,  81],
       [ 16,   6],
       [ 16,  77],
       [ 17,  40],
       [ 17,  76],
       [ 18,   6],
       [ 18,  94],
       [ 19,   3],
       [ 19,  72],
       [ 19,  14],
       [ 19,  99],
       [ 20,  15],
       [ 20,  77],
       [ 20,  13],
       [ 20,  79],
       [ 21,  35],
       [ 21,  66],
       [ 23,  29],
       [ 23,  98],
       [ 24,  35],
       [ 24,  73],
       [ 25,   5],
       [ 25,  73],
       [ 28,  14],
       [ 28,  82],
       [ 28,  32],
       [ 28,  61],
       [ 29,  31],
       [ 29,  87],
       [ 30,   4],
       [ 30,  73],
       [ 33,   4],
       [ 33,  92],
       [ 33,  14],
       [ 33,  81],
       [ 34,  17],
       [ 34,  73],
       [ 37,  26],
       [ 37,  75],
       [ 38,  35],
       [ 38,  92],
       [ 39,  36],
       [ 39,  61],
       [ 39,  28],
       [ 39,  65],
       [ 40,  55],
       [ 40,  47],
       [ 40,  42],
       [ 40,  42],
       [ 42,  52],
       [ 42,  60],
       [ 43,  54],
       [ 43,  60],
       [ 43,  45],
       [ 43,  41],
       [ 44,  50],
       [ 44,  46],
       [ 46,  51],
       [ 46,  46],
       [ 46,  56],
       [ 46,  55],
       [ 47,  52],
       [ 47,  59],
```

```
[ 48,  51],
[ 48,  59],
[ 48,  50],
[ 48,  48],
[ 48,  59],
[ 48,  47],
[ 49,  55],
[ 49,  42],
[ 50,  49],
[ 50,  56],
[ 54,  47],
[ 54,  54],
[ 54,  53],
[ 54,  48],
[ 54,  52],
[ 54,  42],
[ 54,  51],
[ 54,  55],
[ 54,  41],
[ 54,  44],
[ 54,  57],
[ 54,  46],
[ 57,  58],
[ 57,  55],
[ 58,  60],
[ 58,  46],
[ 59,  55],
[ 59,  41],
[ 60,  49],
[ 60,  40],
[ 60,  42],
[ 60,  52],
[ 60,  47],
[ 60,  50],
[ 61,  42],
[ 61,  49],
[ 62,  41],
[ 62,  48],
[ 62,  59],
[ 62,  55],
[ 62,  56],
[ 62,  42],
[ 63,  50],
[ 63,  46],
[ 63,  43],
[ 63,  48],
[ 63,  52],
[ 63,  54],
[ 64,  42],
[ 64,  46],
[ 65,  48],
[ 65,  50],
[ 65,  43],
[ 65,  59],
[ 67,  43],
[ 67,  57],
[ 67,  56],
[ 67,  40],
[ 69,  58],
[ 69,  91],
[ 70,  29],
[ 70,  77],
[ 71,  35],
[ 71,  95],
```

```
[ 71,  11],
[ 71,  75],
[ 71,   9],
[ 71,  75],
[ 72,  34],
[ 72,  71],
[ 73,   5],
[ 73,  88],
[ 73,   7],
[ 73,  73],
[ 74,  10],
[ 74,  72],
[ 75,   5],
[ 75,  93],
[ 76,  40],
[ 76,  87],
[ 77,  12],
[ 77,  97],
[ 77,  36],
[ 77,  74],
[ 78,  22],
[ 78,  90],
[ 78,  17],
[ 78,  88],
[ 78,  20],
[ 78,  76],
[ 78,  16],
[ 78,  89],
[ 78,   1],
[ 78,  78],
[ 78,   1],
[ 78,  73],
[ 79,  35],
[ 79,  83],
[ 81,   5],
[ 81,  93],
[ 85,  26],
[ 85,  75],
[ 86,  20],
[ 86,  95],
[ 87,  27],
[ 87,  63],
[ 87,  13],
[ 87,  75],
[ 87,  10],
[ 87,  92],
[ 88,  13],
[ 88,  86],
[ 88,  15],
[ 88,  69],
[ 93,  14],
[ 93,  90],
[ 97,  32],
[ 97,  86],
[ 98,  15],
[ 98,  88],
[ 99,  39],
[ 99,  97],
[101,  24],
[101,  68],
[103,  17],
[103,  85],
[103,  23],
[103,  69],
```

```
       [113,   8],
       [113,  91],
       [120,  16],
       [120,  79],
       [126,  28],
       [126,  74],
       [137,  18],
       [137,  83]], dtype=int64)
```

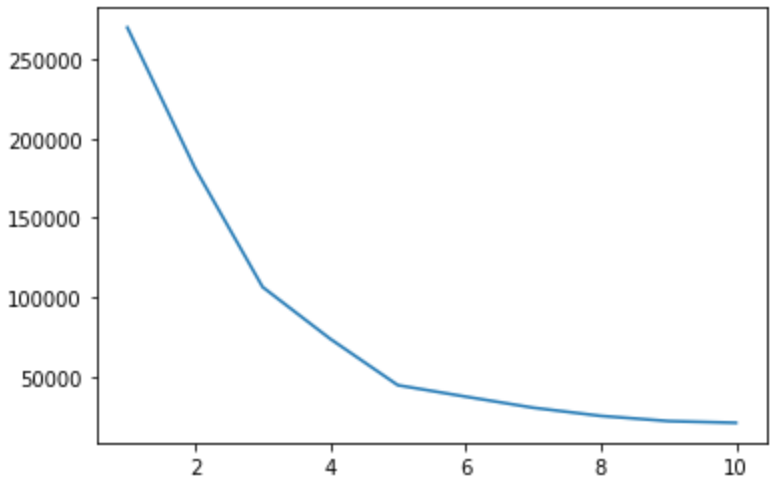
In [40]:
```
from sklearn.cluster import KMeans
```

In [44]:
```
k = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 0)
    kmeans.fit(x)
    k.append(kmeans.inertia_)
```

```
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
```

```
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
```

In [45]:
```
plt.plot(range(1, 11), k)
plt.show()
```

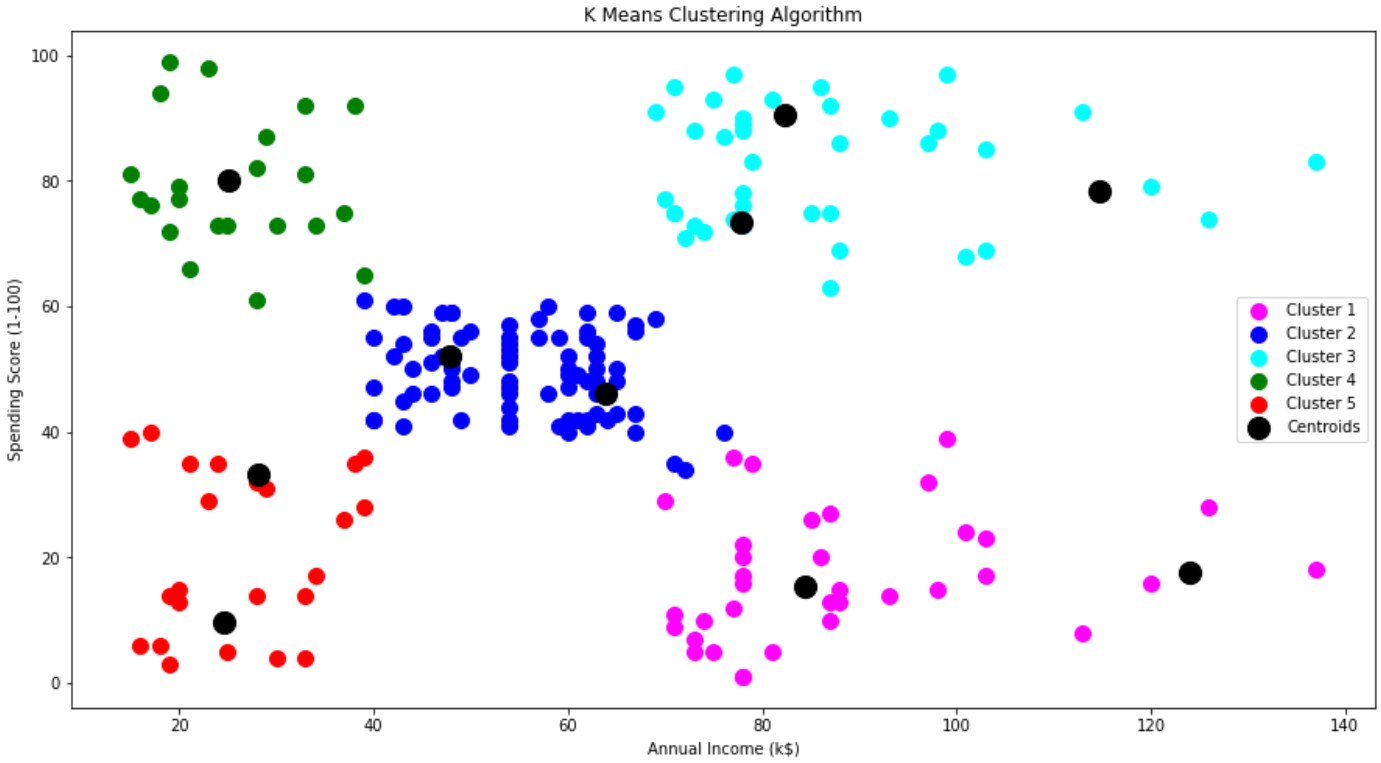


In [47]:
```
model = KMeans(n_clusters = 5, init = 'k-means++', random_state = 0)
y_kemsn = model.fit_predict(x)
```

```
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning:
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning
  warnings.warn(
C:\Application\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning:
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=1.
  warnings.warn(
```

In [48]:
```
plt.figure(1 , figsize = (15 , 8))
plt.scatter(X[y_kemsn == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'magenta', label = 'Cl
plt.scatter(X[y_kemsn == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Clust
plt.scatter(X[y_kemsn == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'cyan', label = 'Clust
plt.scatter(X[y_kemsn == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'green', label = 'Clus
```

```
plt.scatter(X[y_kemsn == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'red', label = 'Cluste
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 200, c = '
plt.title('K Means Clustering Algorithm')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



In [ ]: