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

In [2]:
```python
from mpl_toolkits.mplot3d import Axes3D
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

Load the Dataset

In [11]:
```python
data=pd.read_csv('Mall_Customers.csv')
```

In [10]:
```python
data
```

Out[10]:

	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 [12]:
```python
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 [14]:
```python
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
```

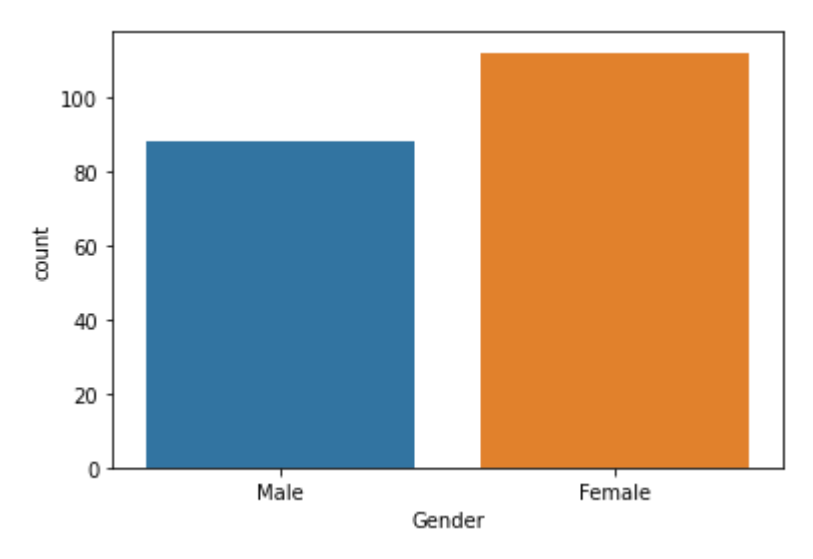
In [16]:
```python
data.isnull().sum()
```

Out[16]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Exploratory Data Analysis

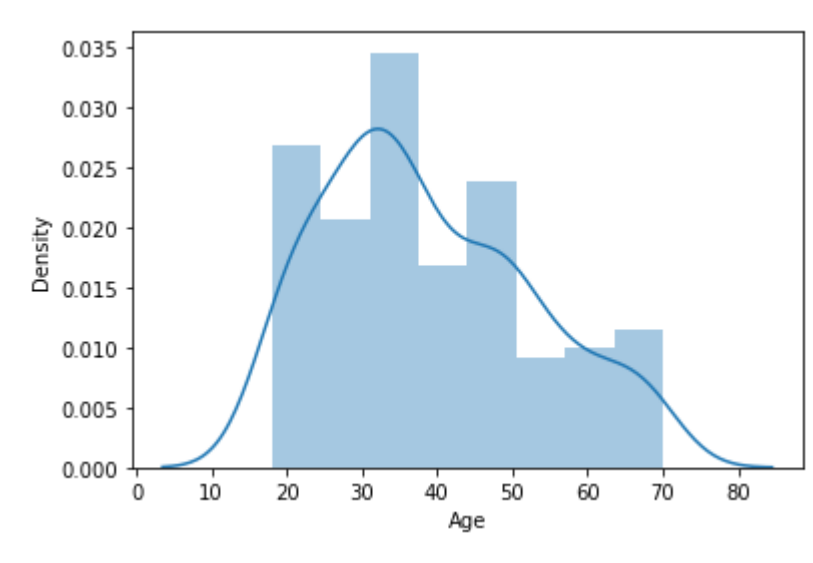
In [19]:
```python
sns.countplot(data['Gender'])
```

Out[19]: <AxesSubplot:xlabel='Gender', ylabel='count'>



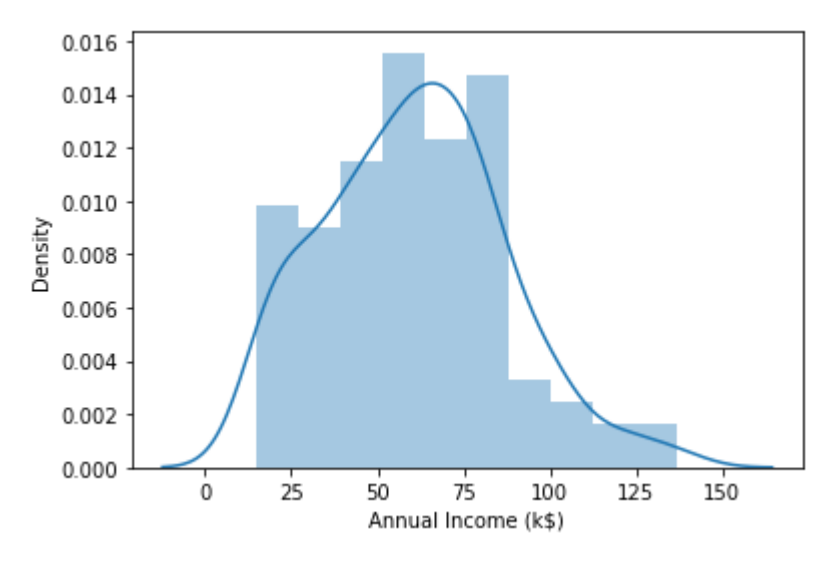
In [21]:
```python
sns.distplot(data['Age'])
```

Out[21]: <AxesSubplot:xlabel='Age', ylabel='Density'>



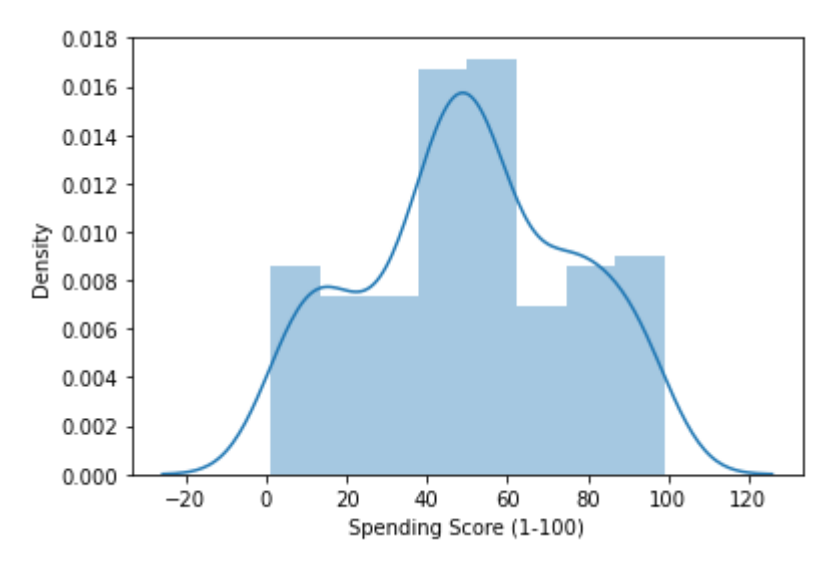
In [23]:
```python
sns.distplot(data['Annual Income (k$)'])
```

Out[23]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Density'>



In [26]:
```python
sns.distplot(data['Spending Score (1-100)'])
```

Out[26]: <AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Density'>



Correlation Matrix

In [27]:
```python
corr = data.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[27]: <AxesSubplot:>

Clustering

In [28]:
```python
data.head()
```

Out[28]:

	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

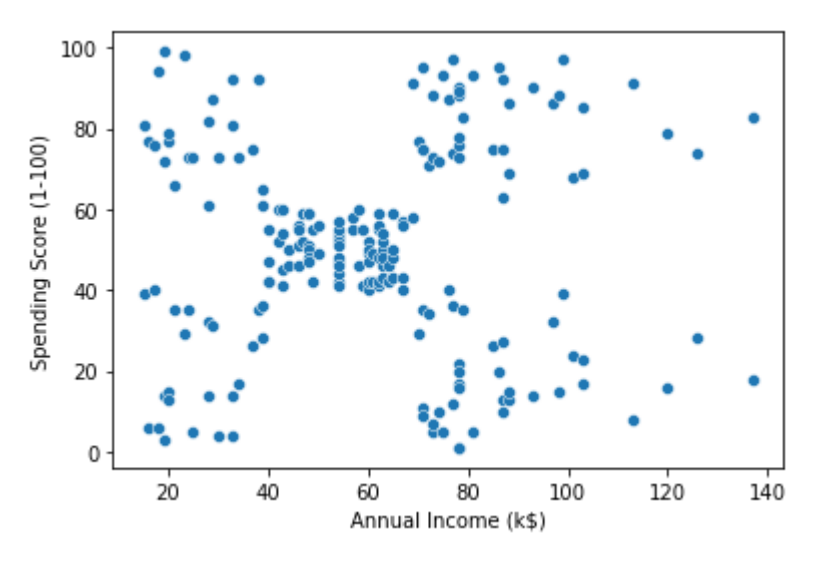
In [32]:
```python
#cluster on 2 features
df1 = data[['Annual Income (k$)', 'Spending Score (1-100)']]
df1.head()
```

Out[32]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40

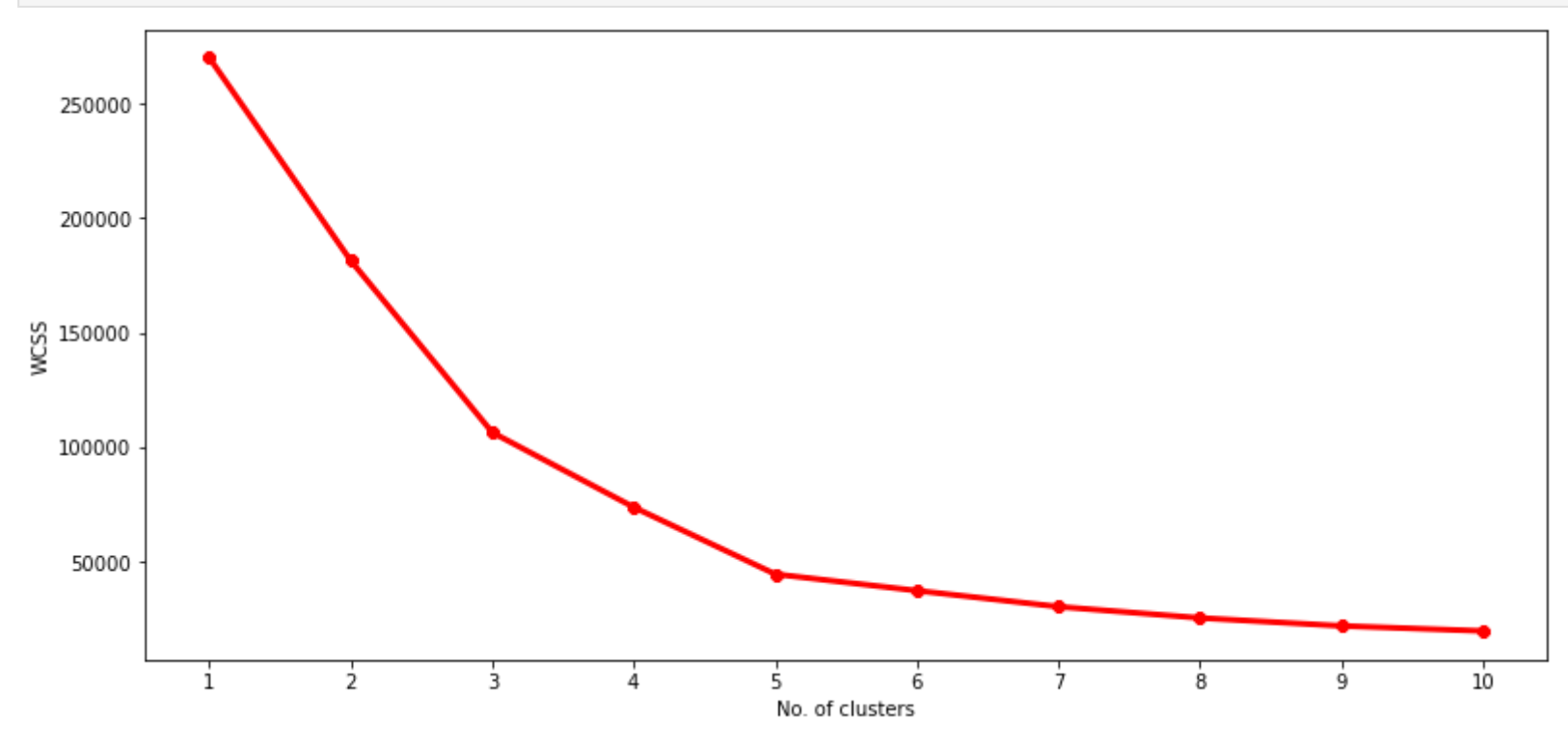
In [33]:
```python
# scatter plot
sns.scatterplot(df1['Annual Income (k$)'], df1['Spending Score (1-100)'])
```

Out[33]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>



In [34]:
```python
from sklearn.cluster import KMeans
errors = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(df1)
    errors.append(kmeans.inertia_)
```

In [35]:
```python
# plot the results for elbow method
plt.figure(figsize=(13,6))
plt.plot(range(1,11), errors)
plt.plot(range(1,11), errors, linewidth=3, color='red', marker='8')
plt.xlabel('No. of clusters')
plt.ylabel('WCSS')
plt.xticks(np.arange(1,11,1))
plt.show()
```



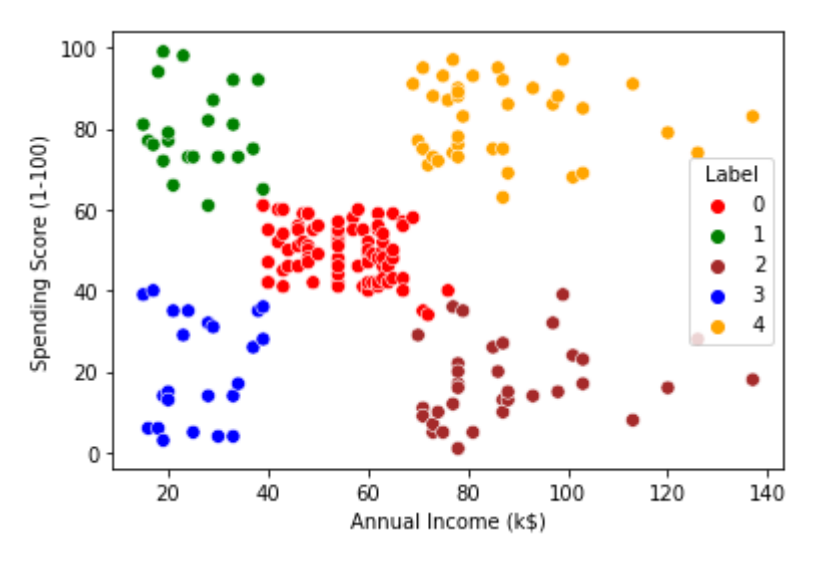
In [36]:
```python
km = KMeans(n_clusters=5)
km.fit(df1)
y = km.predict(df1)
df1['Label'] = y
df1.head()
```

Out[36]:

	Annual Income (k$)	Spending Score (1-100)	Label
0	15	39	3
1	15	81	1
2	16	6	3
3	16	77	1
4	17	40	3

In [37]:
```python
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)', data=df1, hue='Label', s=50, palette=['red', 'green', 'brown', 'blue', 'orange'
```

Out[37]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>



In [39]:
```python
# cluster on 3 features
df2 = data[['Annual Income (k$)', 'Spending Score (1-100)', 'Age']]
df2.head()
```

Out[39]:

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

In [40]:
```python
errors = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(df2)
    errors.append(kmeans.inertia_)
```

In [42]:
```python
# plot the results for elbow method
plt.figure(figsize=(13,6))
plt.plot(range(1,11), errors)
plt.plot(range(1,11), errors, linewidth=3, color='red', marker='8')
plt.xlabel('No. of clusters')
plt.ylabel('WCSS')
plt.xticks(np.arange(1,11,1))
plt.show()
```

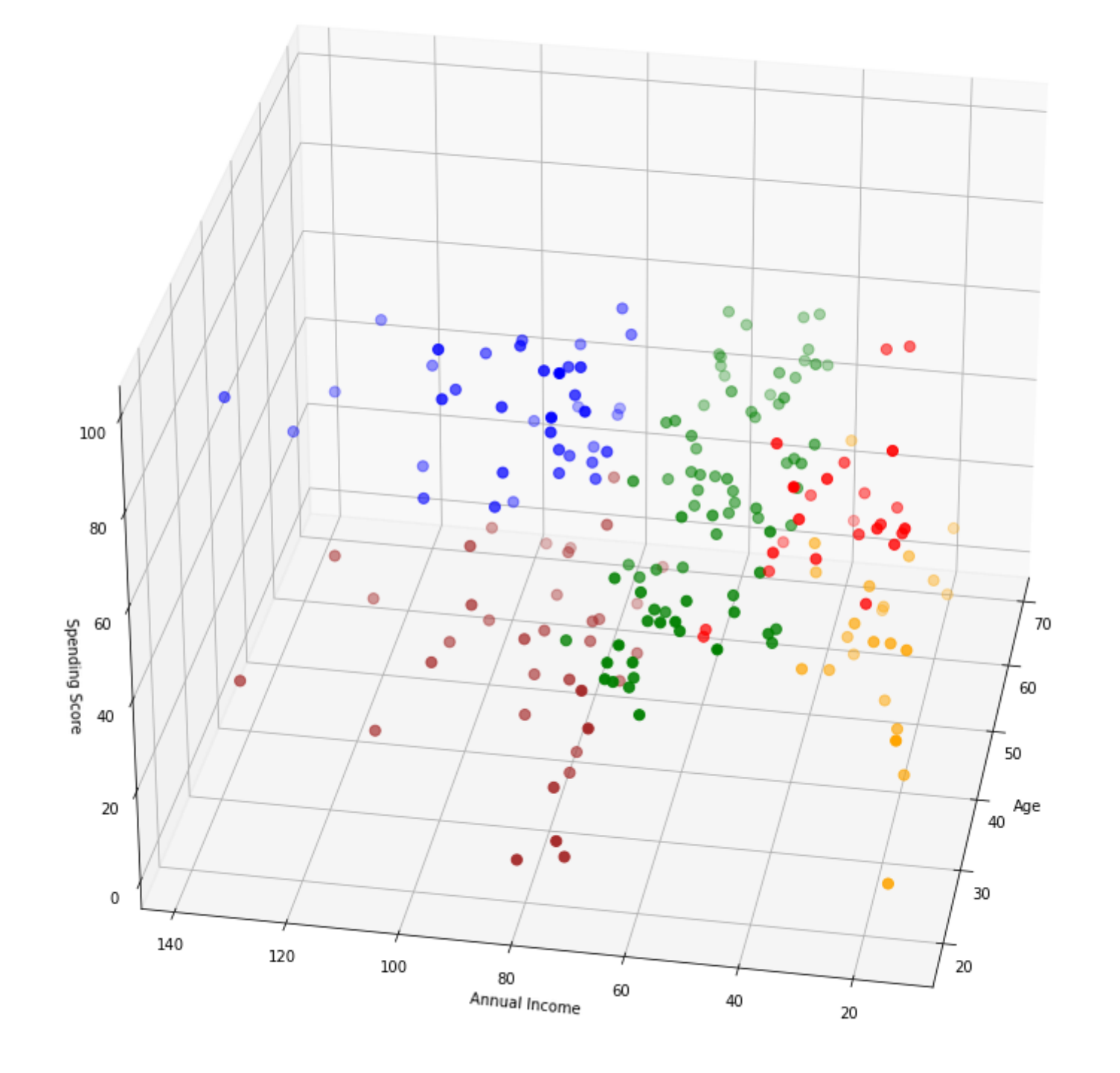
In [43]:
```python
km = KMeans(n_clusters=5)
km.fit(df2)
y = km.predict(df2)
df2['Label'] = y
df2.head()
```

Out[43]:

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

In [44]:
```python
# 3d scatter plot
fig = plt.figure(figsize=(20,15))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df2['Age'][df2['Label']==0], df2['Annual Income (k$)'][df2['Label']==0], df2['Spending Score (1-100)'][df2['Label']==0], c='red', s=50)
ax.scatter(df2['Age'][df2['Label']==1], df2['Annual Income (k$)'][df2['Label']==1], df2['Spending Score (1-100)'][df2['Label']==1], c='green', s=50
ax.scatter(df2['Age'][df2['Label']==2], df2['Annual Income (k$)'][df2['Label']==2], df2['Spending Score (1-100)'][df2['Label']==2], c='blue', s=50)
ax.scatter(df2['Age'][df2['Label']==3], df2['Annual Income (k$)'][df2['Label']==3], df2['Spending Score (1-100)'][df2['Label']==3], c='brown', s=50
ax.scatter(df2['Age'][df2['Label']==4], df2['Annual Income (k$)'][df2['Label']==4], df2['Spending Score (1-100)'][df2['Label']==4], c='orange', s=5
ax.view_init(30, 190)
ax.set_xlabel('Age')
ax.set_ylabel('Annual Income')
ax.set_zlabel('Spending Score')
plt.show()
```



In [ ]: