In [175]:

```
import numpy as np
```

In [176]:

```
import pandas as pd
```

In [177]:

```
import matplotlib.pyplot as plt
```

In [178]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
```

In [179]:

```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

In [180]:

```
data = pd.read_csv('C:\\Users\\dell\\mall customer.csv')
```

In [181]:

```
data
```

Out[181]:

	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 [182]:

```
data.drop('CustomerID',axis=1, inplace=True)
```

In [183]:

```
encoder = LabelEncoder()
data['Gender'] = encoder.fit_transform(data['Gender'])

gender_mappings = {index: label for index, label in enumerate(encoder.classes_)}
gender_mappings
```

Out[183]:

{0: 'Female', 1: 'Male'}

In [184]:

```
gender_mappings
```

Out[184]:

{0: 'Female', 1: 'Male'}

In [185]:

```
scaler = StandardScaler()
Scaled_data = pd.DataFrame(scaler.fit_transform(data), columns = data.columns)
```

In [186]:

```
Scaled_data
```

Out[186]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.128152	-1.424569	-1.738999	-0.434801
1	1.128152	-1.281035	-1.738999	1.195704
2	-0.886405	-1.352802	-1.700830	-1.715913
3	-0.886405	-1.137502	-1.700830	1.040418
4	-0.886405	-0.563369	-1.662660	-0.395980
...	...	...	...	...
195	-0.886405	-0.276302	2.268791	1.118061
196	-0.886405	0.441365	2.497807	-0.861839
197	1.128152	-0.491602	2.497807	0.923953
198	1.128152	-0.491602	2.917671	-1.250054
199	1.128152	-0.635135	2.917671	1.273347

200 rows × 4 columns

In [187]:

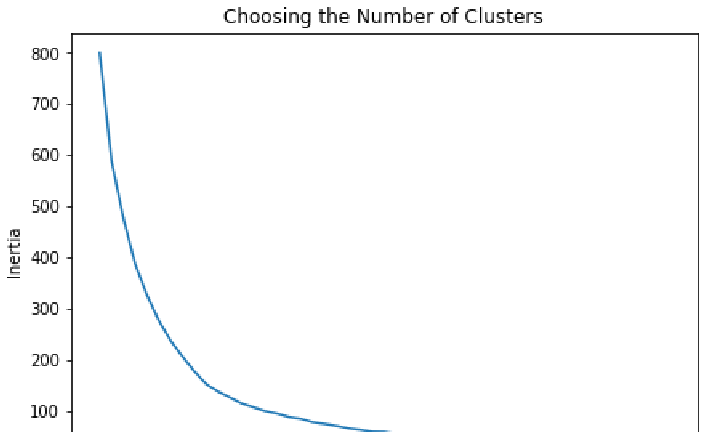
```
max_clusters = 50
```

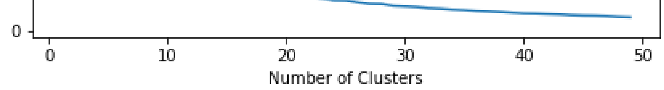
In [188]:

```
kmeans_tests = [KMeans(n_clusters=i, n_init=10) for i in range(1, max_clusters)]
inertias = [kmeans_tests[i]. fit(Scaled_data).inertia_ for i in range(len(kmeans_tests))]
```

In [189]:

```
plt.figure(figsize=(7, 5))
plt.plot(range(1, max_clusters), inertias)
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Choosing the Number of Clusters')
plt.show()
```





In [190]:

```
kmeans = KMeans(n_clusters=10, n_init=10)
kmeans.fit(Scaled_data)
```

Out[190]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=10, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [191]:

```
clusters = kmeans.predict(Scaled_data)
clusters
```

Out[191]:

```
array([5, 5, 7, 9, 7, 9, 7, 9, 3, 9, 3, 9, 7, 9, 7, 5, 7, 5, 3, 9, 5, 5,
       7, 5, 7, 5, 7, 5, 7, 9, 3, 9, 3, 5, 7, 9, 7, 9, 7, 9, 0, 5, 3, 1,
       7, 9, 0, 1, 1, 1, 0, 5, 1, 3, 0, 3, 0, 3, 1, 3, 3, 5, 0, 0, 3, 5,
       0, 0, 5, 1, 3, 0, 0, 0, 3, 5, 0, 5, 1, 0, 3, 5, 3, 0, 1, 3, 0, 1,
       1, 0, 0, 5, 3, 1, 1, 5, 0, 1, 3, 5, 1, 0, 3, 5, 3, 1, 0, 3, 3, 3,
       3, 1, 1, 5, 1, 1, 0, 0, 0, 0, 5, 1, 1, 8, 1, 4, 2, 8, 3, 8, 2, 8,
       1, 4, 2, 4, 6, 8, 2, 4, 6, 8, 1, 4, 2, 8, 2, 4, 6, 8, 2, 8, 6, 4,
       6, 4, 2, 4, 2, 4, 6, 4, 2, 4, 2, 4, 2, 4, 6, 8, 2, 8, 2, 8, 6, 4,
       2, 8, 2, 8, 6, 4, 2, 4, 6, 8, 6, 8, 6, 4, 6, 4, 2, 4, 6, 4, 6, 8,
       2, 8])
```

In [192]:

```
pca = PCA(n_components=2)
reduced_data = pd.DataFrame(pca.fit_transform(Scaled_data), columns=['PC1','PC2'])
```

In [193]:

```
reduced_data
```

Out[193]:

	PC1	PC2
0	-0.406383	-0.520714
1	-1.427673	-0.367310
2	0.050761	-1.894068
3	-1.694513	-1.631908
4	-0.313108	-1.810483
...	...	...
195	-1.179572	1.324568
196	0.672751	1.221061
197	-0.723719	2.765010
198	0.767096	2.861930
199	-1.065015	3.137256

200 rows × 2 columns

In [194]:

```
reduced_data
```

Out[194]:

	PC1	PC2
0	-0.406383	-0.520714

	PC1	PC2
1	-1.427673	-0.367310
2	0.050761	-1.894068
3	-1.694513	-1.631908
4	-0.313108	-1.810483
...	...	...
195	-1.179572	1.324568
196	0.672751	1.221061
197	-0.723719	2.765010
198	0.767096	2.861930
199	-1.065015	3.137256

200 rows × 2 columns

In [195]:

```
kmeans.cluster_centers_
```

Out[195]:

```
array([[-0.88640526,  1.09300668, -0.27940022, -0.02639866],
       [-0.88640526, -0.78153925, -0.12214217, -0.11957041],
       [ 1.12815215, -0.02700694,  0.96701244, -1.39716754],
       [ 1.12815215,  1.43505777, -0.45298304, -0.40195247],
       [-0.88640526, -0.47793198,  0.97284787,  1.22158511],
       [ 1.12815215, -0.97602698, -0.73705168,  0.41603773],
       [-0.88640526,  0.41265847,  1.21277   , -1.11029664],
       [-0.7425083 ,  0.16967696, -1.31640908, -1.1668652 ],
       [ 1.12815215, -0.39989994,  1.01344075,  1.26040667],
       [-0.88640526, -0.96084556, -1.33087991,  1.17778643]])
```

In [196]:

```
reduced_centers = pca.transform(kmeans.cluster_centers_)
```

In [197]:

```
reduced_centers
```

Out[197]:

```
array([[ 0.56402657, -0.88554419],
       [-0.662429  , -0.58044771],
       [ 1.19961046,  1.30582744],
       [ 1.5303687 ,  0.17028966],
       [-1.38150389,  0.3644368 ],
       [-0.68838314,  0.28733559],
       [ 0.83149037,  0.21501655],
       [ 0.75229959, -1.61087948],
       [-0.88272588,  1.65431318],
       [-1.6696024 , -1.35294268]])
```

In [198]:

```
reduced_data['cluster'] = clusters
```

In [199]:

```
reduced_data
```

Out[199]:

	PC1	PC2	cluster
0	-0.406383	-0.520714	5
1	-1.427673	-0.367310	5
2	0.050761	-1.894068	7
3	-1.694513	-1.631908	9
4	-0.313108	-1.810483	7
...	...	...	...

	PC1	PC2	cluster
195	-1.179572	1.324568	4
196	0.672751	1.221061	6
197	-0.723719	2.765010	8
198	0.767096	2.861930	2
199	-1.065015	3.137256	8

200 rows × 3 columns

In [200]:

```
reduced_data[reduced_data['cluster'] == 7].loc[:, 'PC1']
```

Out[200]:

```
2     0.050761
4    -0.313108
6     0.790821
12    1.685823
14    1.174436
16    0.016773
22    1.358915
24    1.513159
26    0.588833
28    0.368426
34    1.265157
36    0.839345
38    0.302432
44    0.890422
Name: PC1, dtype: float64
```

In [201]:

```
reduced_data[reduced_data['cluster']==7].loc[:, 'PC2']
```

Out[201]:

```
2    -1.894068
4    -1.810483
6    -1.947271
12   -2.023945
14   -0.612791
16   -1.743446
22   -1.828669
24   -1.764512
26   -1.625416
28   -1.563006
34   -1.581259
36   -1.487939
38   -1.319601
44   -1.349908
Name: PC2, dtype: float64
```

In [202]:

```
reduced_data[reduced_data['cluster']==0].loc[:, 'PC1']
```

Out[202]:

```
40    1.493876
46    0.219541
50    0.249710
54    0.485517
56    0.401317
62    1.137179
63    0.308720
66    0.005442
67    1.292984
71    0.416021
72    0.870906
73    0.684235
76    0.022784
79    0.513597
83    0.312158
86    0.382434
89    0.455368
90    1.103760
```

96   0.280131
101   0.351736
106   1.137431
116   1.175533
117   0.057699
118   0.582649
119   0.159938
Name: PC1, dtype: float64

In [203]:

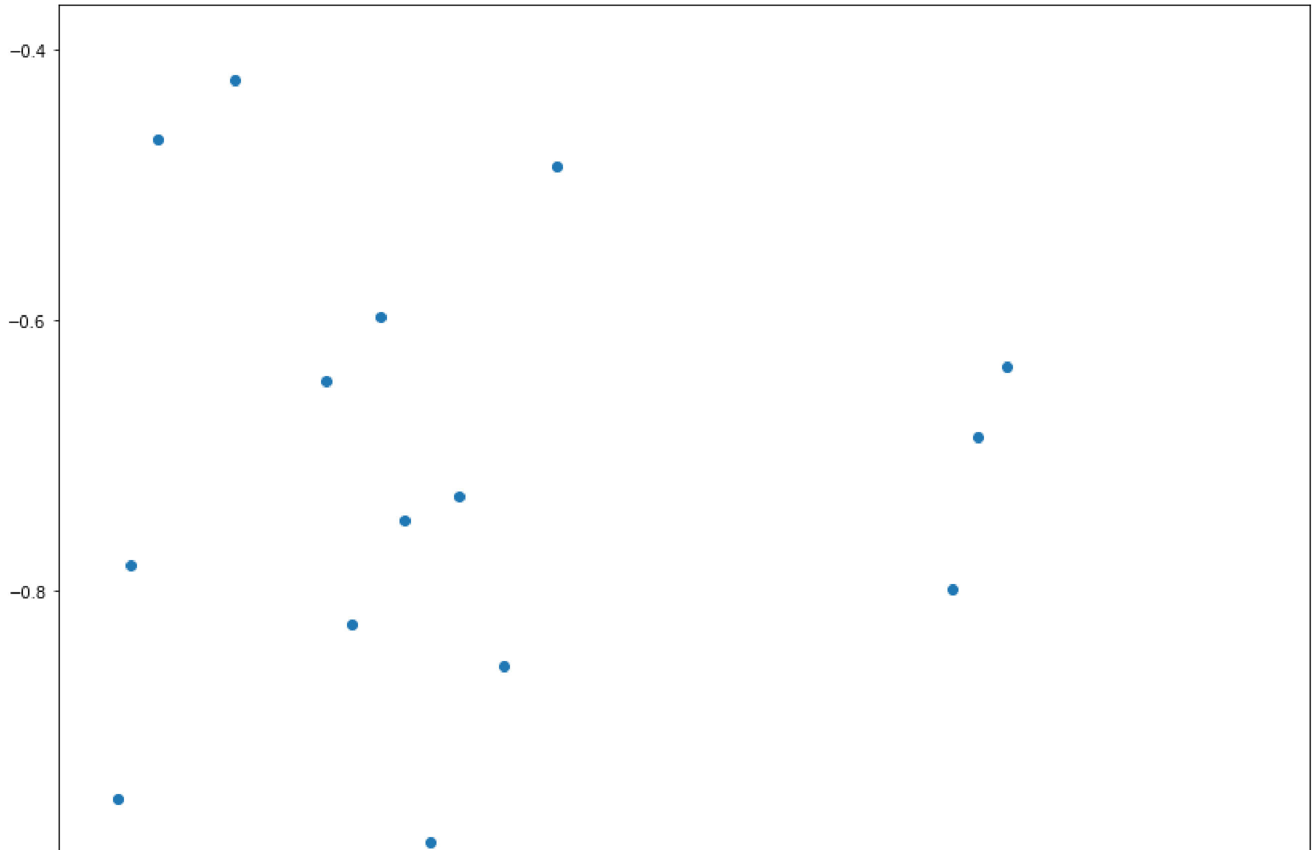
```
reduced_data[reduced_data['cluster']==0].loc[:, 'PC2']
```

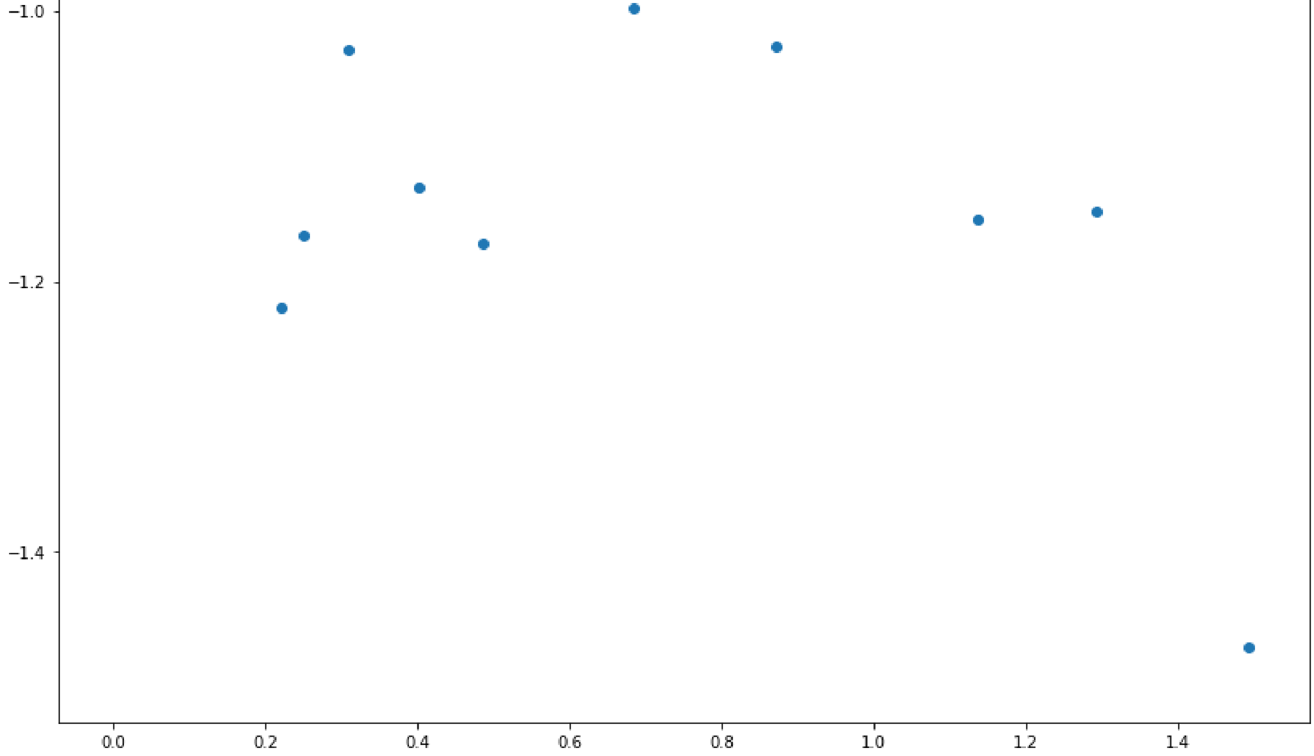
Out[203]:

40   -1.470133
46   -1.219956
50   -1.166116
54   -1.172396
56   -1.130595
62   -1.154016
63   -1.029230
66   -0.954226
67   -1.148276
71   -0.986837
72   -1.026319
73   -0.998271
76   -0.781833
79   -0.855674
83   -0.825336
86   -0.748586
89   -0.730251
90   -0.798927
96   -0.645501
101   -0.597959
106   -0.687241
116   -0.634546
117   -0.466256
118   -0.486830
119   -0.423293
Name: PC2, dtype: float64

In [204]:

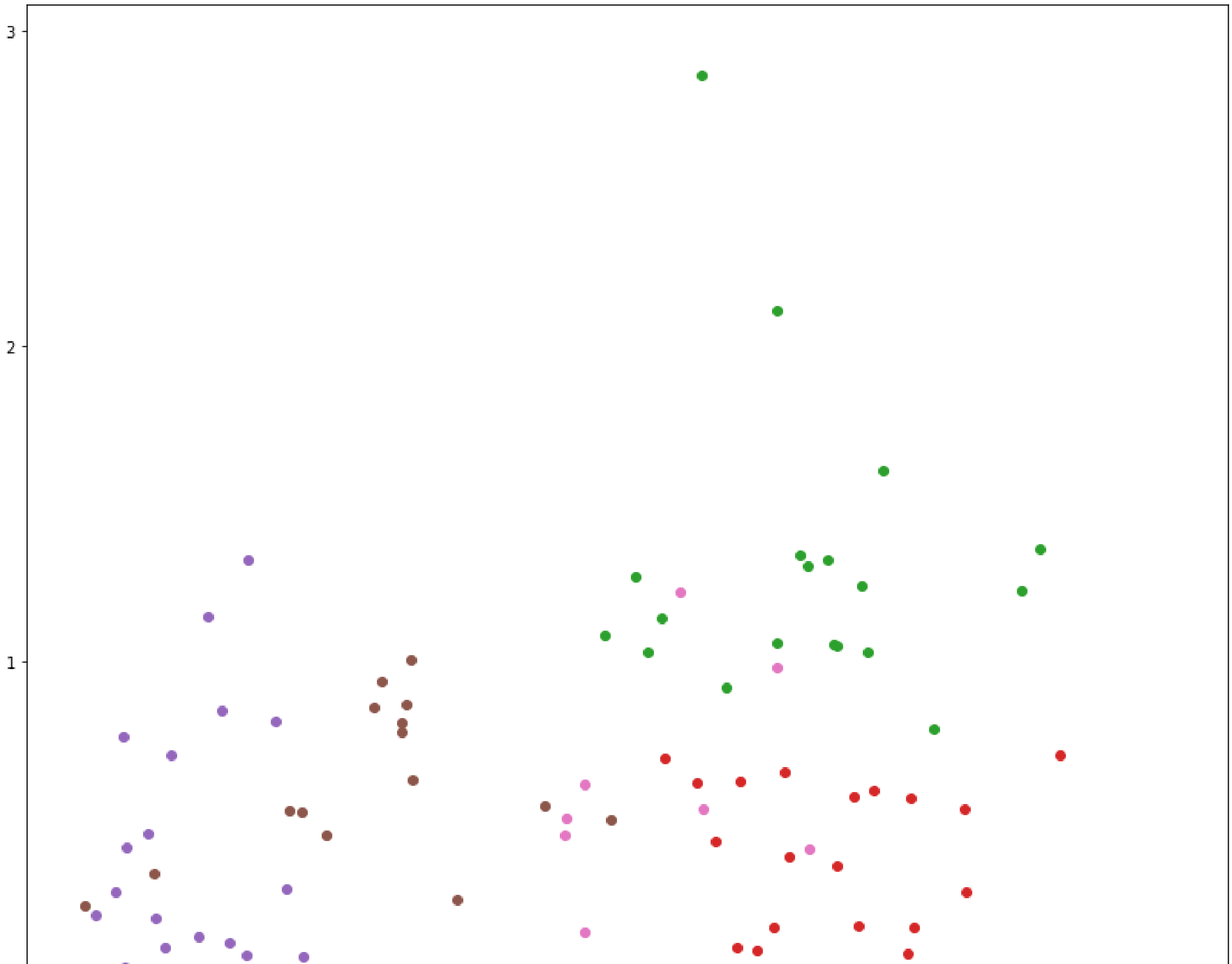
```
plt.figure(figsize=(14, 18))
plt.scatter(reduced_data[reduced_data['cluster'] == 0].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 0].loc[:,'PC2'])
plt.show()
```

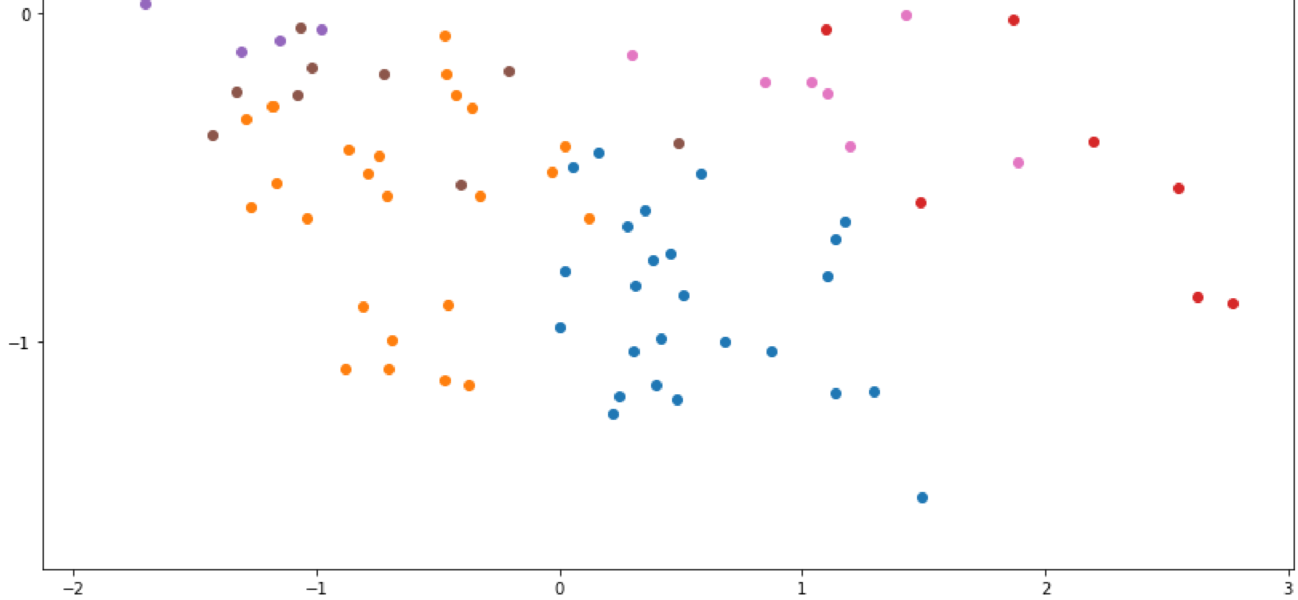




In [205]:

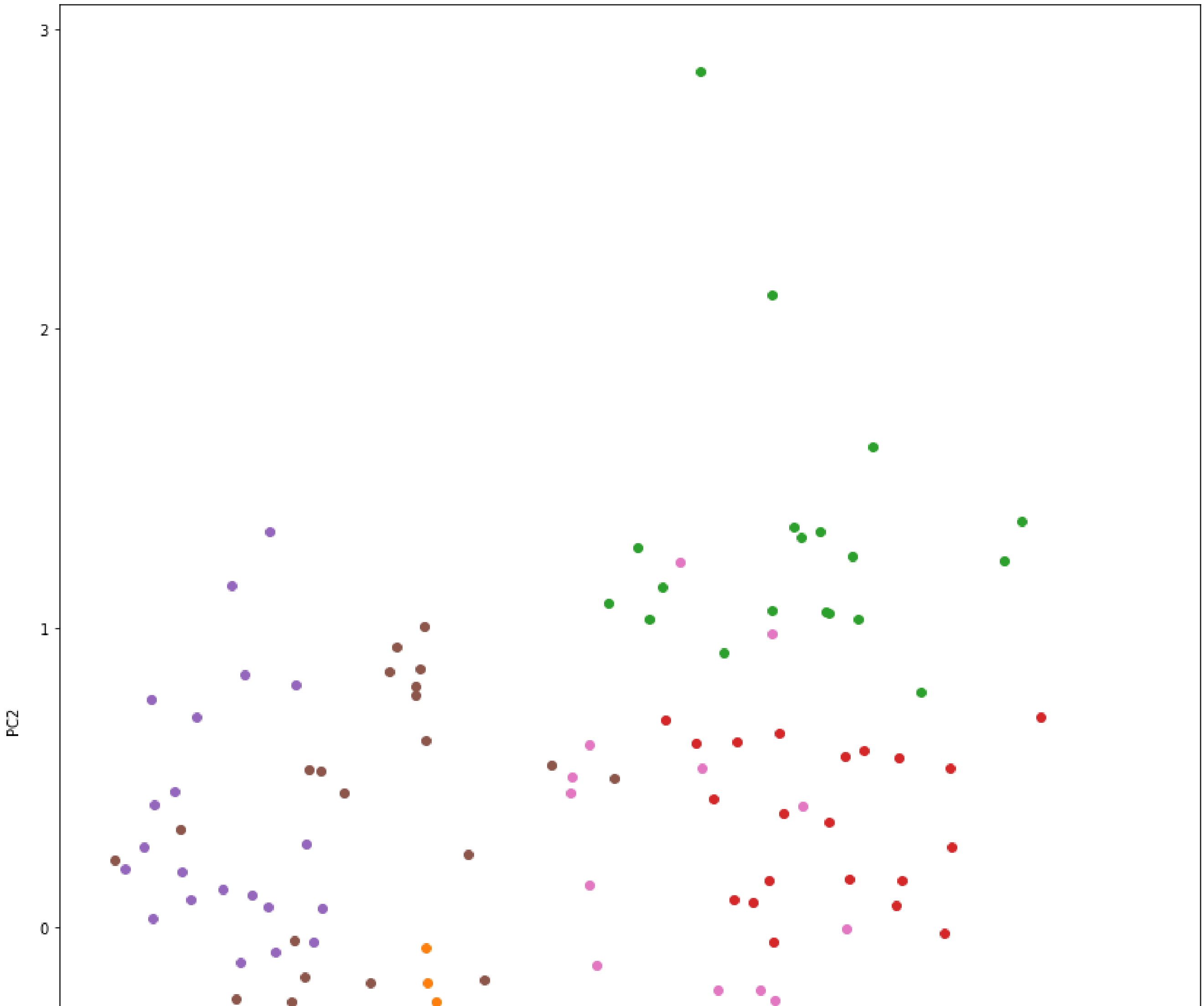
```
plt.figure(figsize=(14, 18))

plt.scatter(reduced_data[reduced_data['cluster'] == 0].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 0].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 1].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 1].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 2].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 2].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 3].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 3].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 4].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 4].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 5].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 5].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 6].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 6].loc[:,'PC2'])


plt.show()
```

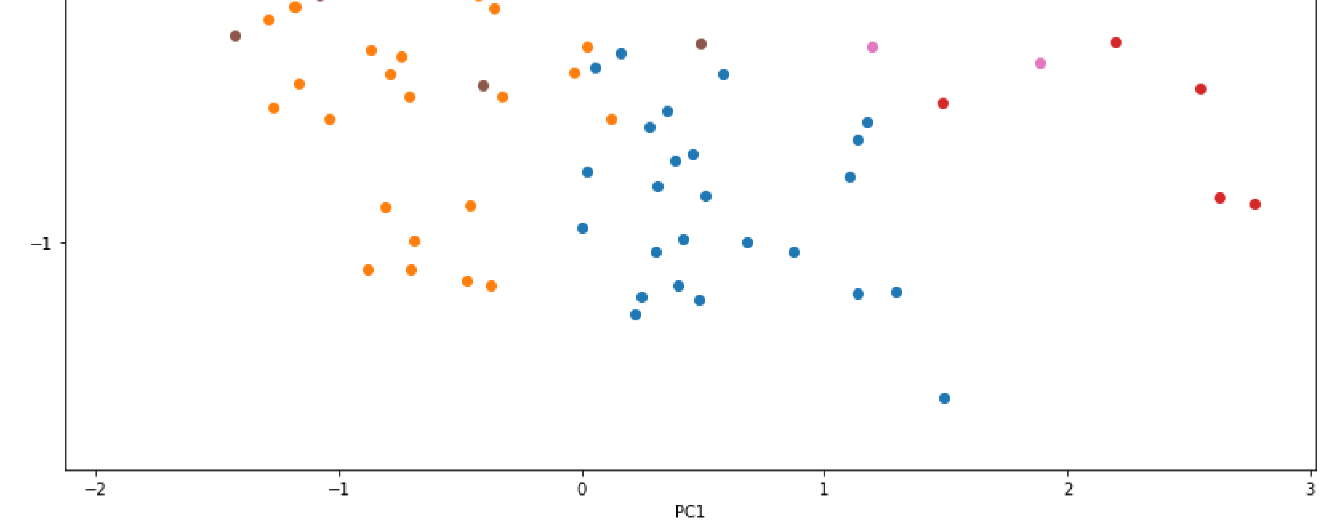




In [206]:

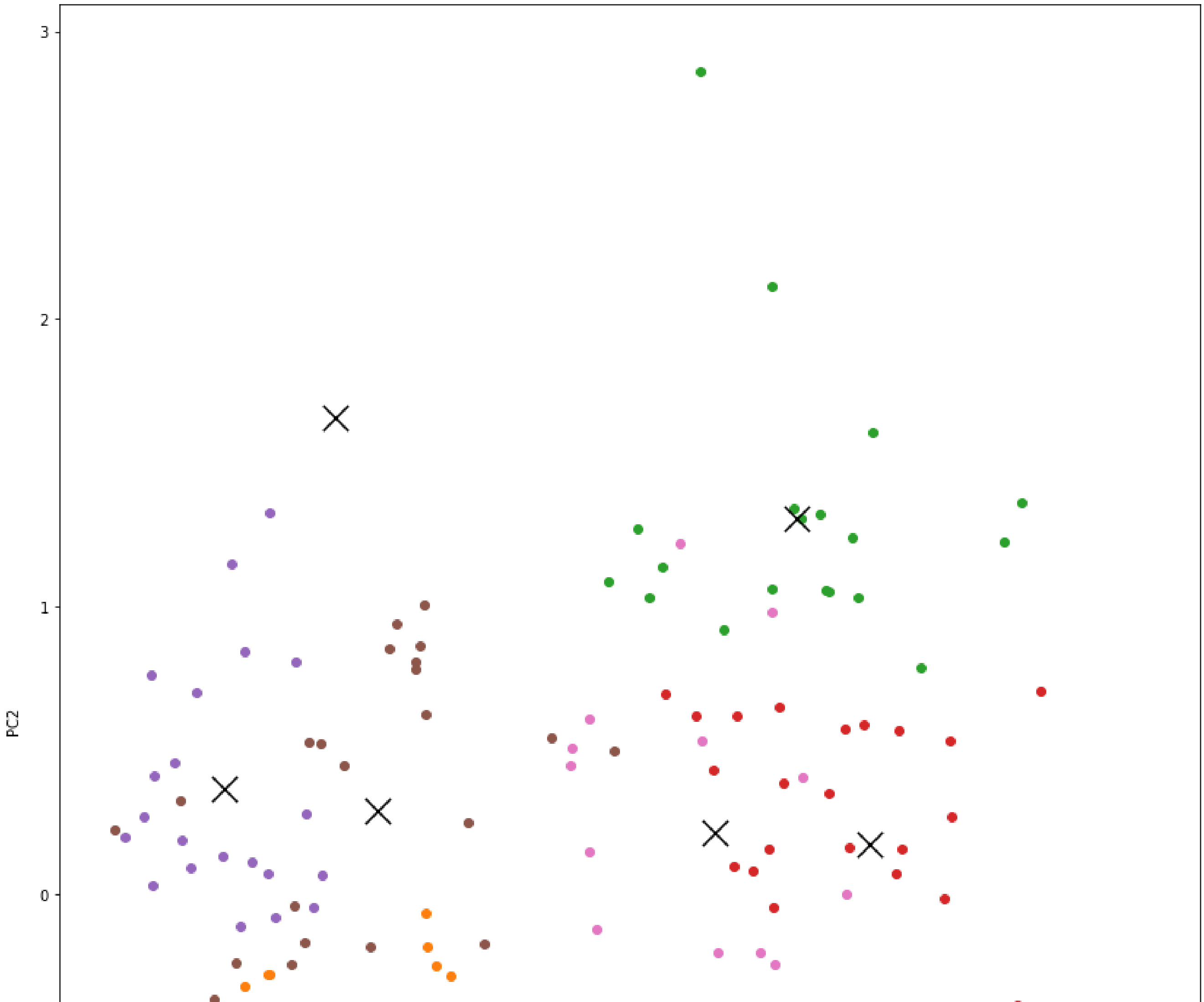
```
plt.figure(figsize=(14, 18))

plt.scatter(reduced_data[reduced_data['cluster'] == 0].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 0].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 1].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 1].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 2].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 2].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 3].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 3].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 4].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 4].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 5].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 5].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 6].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 6].loc[:,'PC2'])


plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```

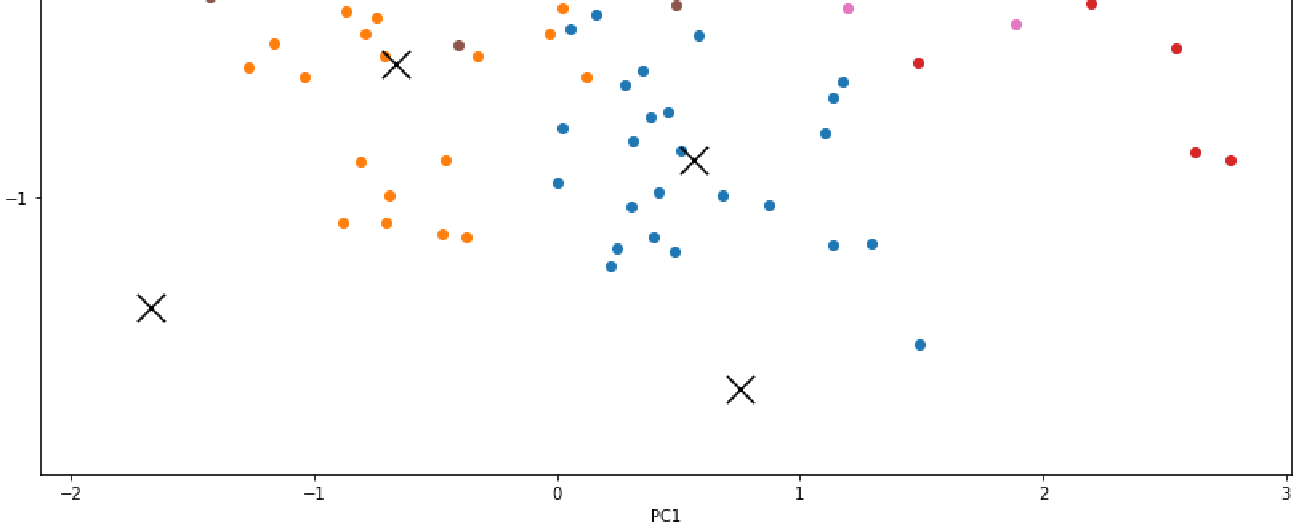




In [207]:

```
plt.figure(figsize=(14, 18))

plt.scatter(reduced_data[reduced_data['cluster'] == 0].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 0].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 1].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 1].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 2].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 2].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 3].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 3].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 4].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 4].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 5].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 5].loc[:,'PC2'])
plt.scatter(reduced_data[reduced_data['cluster'] == 6].loc[:, 'PC1'], reduced_data[reduced_data['cluster'] == 6].loc[:,'PC2'])


plt.scatter(reduced_centers[:, 0], reduced_centers[:, 1], color = 'black', marker = 'x', s=300)

plt.xlabel('PC1')
plt.ylabel('PC2')

plt.show()
```





In [208]:

```
labels = {'Female','Male'}
```

In [209]:

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

In [210]:

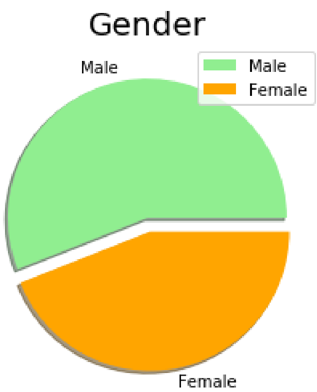
```
colors = ['lightgreen', 'orange']
```

In [211]:

```
explode = [0, 0.1]
```

In [212]:

```
plt.pie(size, colors = colors,explode = explode,labels = labels, shadow = True)
plt.title('Gender', fontsize = 20)
plt.axis('off')
plt.legend()
plt.show()
```



In [213]:

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

In [214]:

```
print(data.info)
```

```
<bound method DataFrame.info of     Gender  Age  Annual Income (k$)  Spending Score (1-100)
0        1   19                  15                      39
1        1   21                  15                      81
```

```
2      0  20          16            6
3      0  23          16           77
4      0  31          17           40
..   ... ...         ...          ...
195    0  35         120           79
196    0  45         126           28
197    1  32         126           74
198    1  32         137           18
199    1  30         137           83

[200 rows x 4 columns]>
```

In [215]:

```python
data_comparision = data[['Annual Income (k$)', 'Spending Score (1-100)']]
```

In [216]:

```python
data_comparision.head(10)
```

Out[216]:

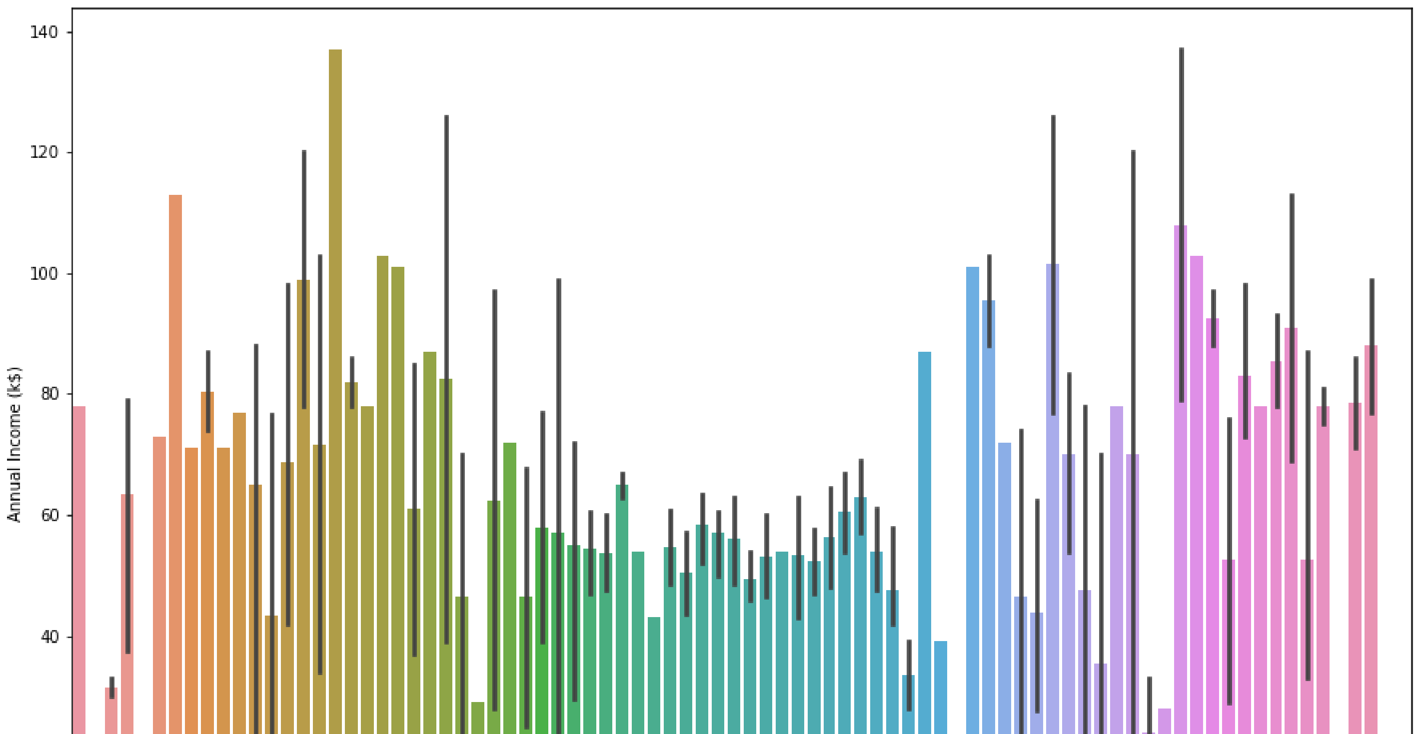
	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40
5	17	76
6	18	6
7	18	94
8	19	3
9	19	72

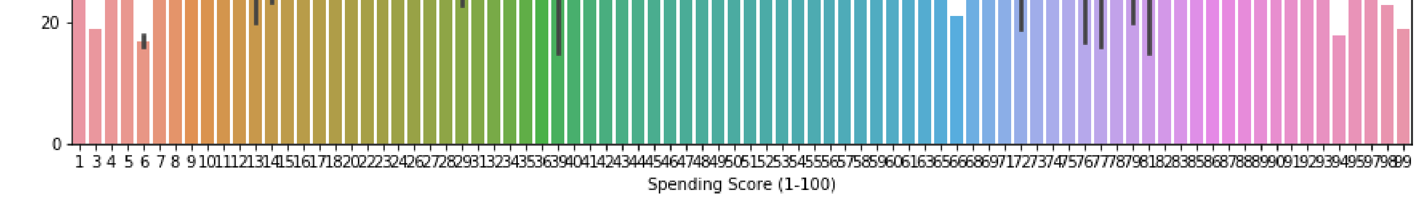
In [217]:

```python
df_top = data[['Annual Income (k$)', 'Spending Score (1-100)']]
fig, ax = plt.subplots(figsize =(15,10))
sns.barplot(y = 'Annual Income (k$)', x = 'Spending Score (1-100)', data = df_top, ax = ax)
```

Out[217]:

<matplotlib.axes._subplots.AxesSubplot at 0x1edd7acca08>



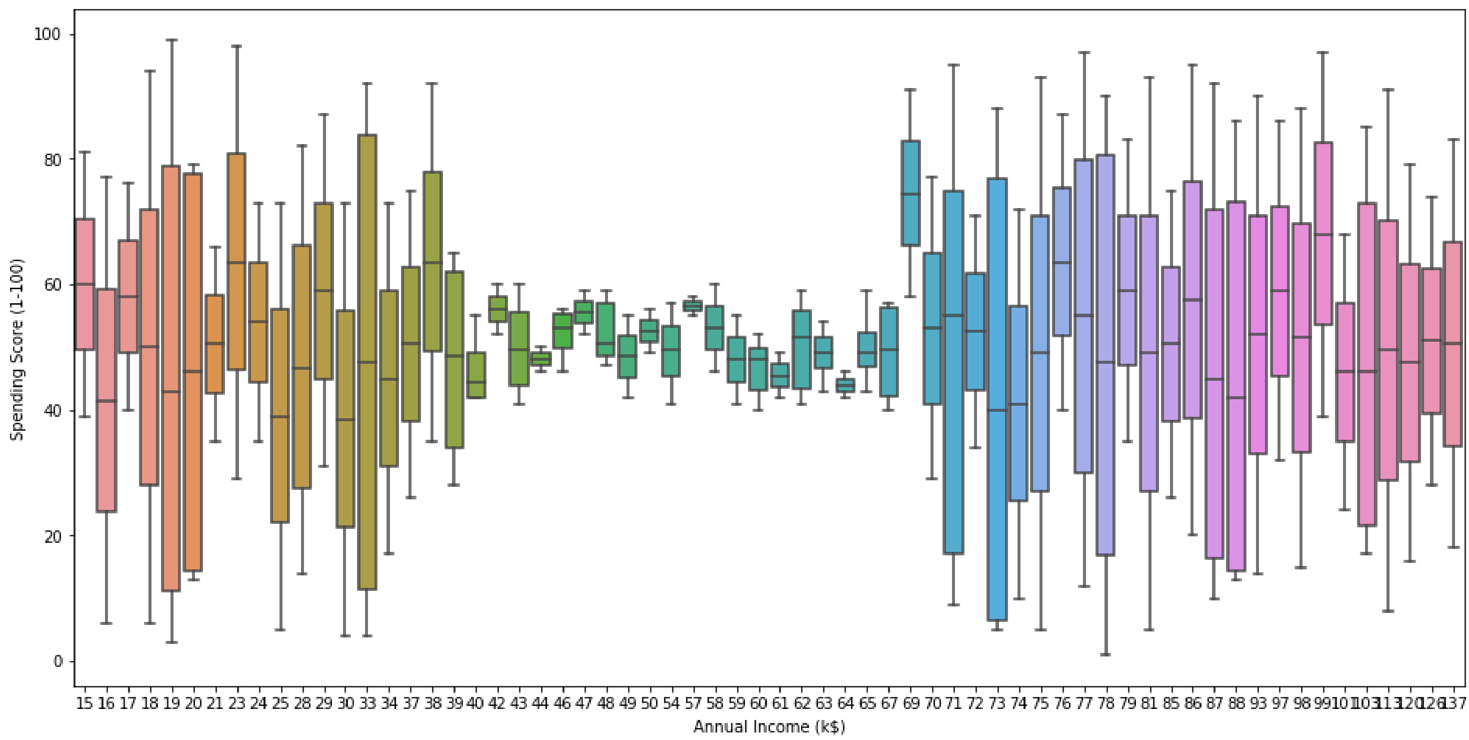


In [218]:

```
fig, ax = plt.subplots(figsize = (16,8 ))
sns.boxplot(x = 'Annual Income (k$)', y = 'Spending Score (1-100)', data = data_comparision, ax = ax)
```

Out[218]:

<matplotlib.axes._subplots.AxesSubplot at 0x1edd274e348>



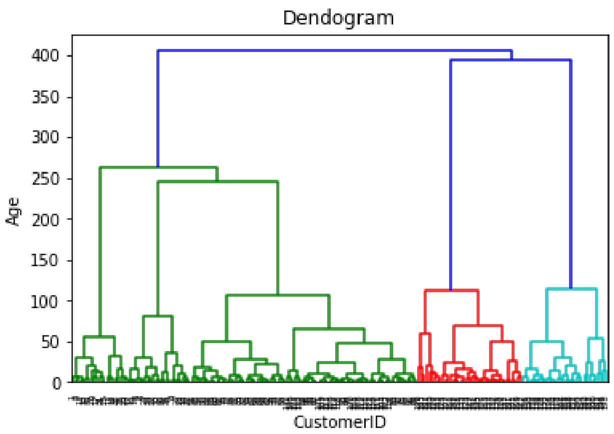
In [219]:

```
corrmat = data.corr()
fig = plt.figure(figsize=(5,5))
sns.heatmap(corrmat, vmax=1, square = True)
plt.show()
```

In [220]:

```
import scipy.cluster.hierarchy as sch
```

```
dendogram = sch.dendrogram(sch.linkage(data_comparision, method = 'ward'))
plt.title('Dendogram')
plt.xlabel('CustomerID')
plt.ylabel('Age')
plt.show()
```

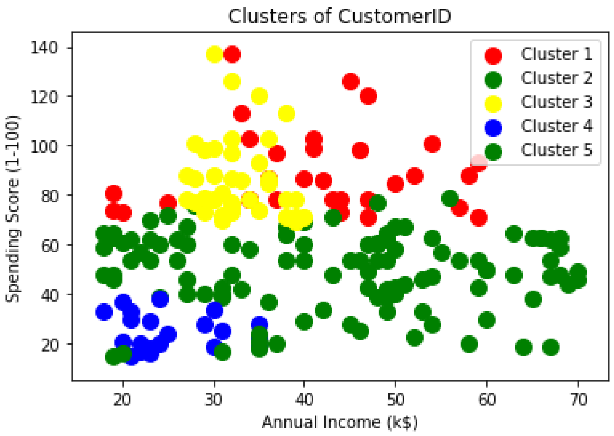


In [221]:

```
x = data.iloc[:, [1,2]].values

from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')

y_hc = hc.fit_predict(data_comparision)
```

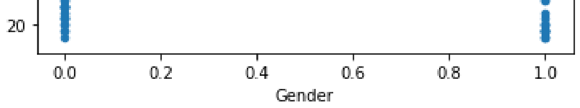
In [222]:

```
plt.scatter(x[y_hc  == 0,  0], x[y_hc == 0,  1], s=100, c='red', label = 'Cluster 1')
plt.scatter(x[y_hc  == 1,  0], x[y_hc == 1,  1], s=100, c='green', label = 'Cluster 2')
plt.scatter(x[y_hc  == 2,  0], x[y_hc == 2,  1], s=100, c='yellow', label = 'Cluster 3')
plt.scatter(x[y_hc  == 3,  0], x[y_hc == 3,  1], s=100, c='blue', label = 'Cluster 4')
plt.scatter(x[y_hc  == 4,  0], x[y_hc == 4,  1], s=100, c='green', label = 'Cluster 5')
plt.title('Clusters of CustomerID')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



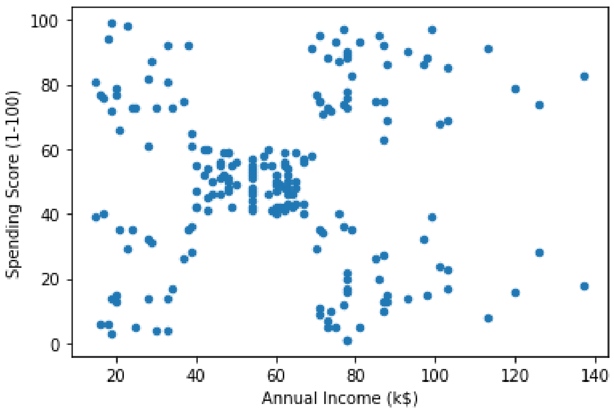
In [223]:

```
data.plot(kind='scatter',x='Gender',y='Age');
```



In [224]:

```
data.plot(kind='scatter',x='Annual Income (k$)',y='Spending Score (1-100)');
plt.show()
```

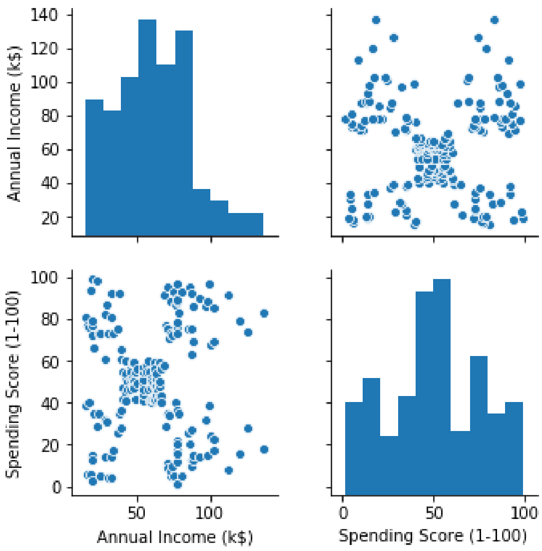


In [225]:

```
sns.pairplot(data,vars=['Annual Income (k$)','Spending Score (1-100)'])
```

Out[225]:

<seaborn.axisgrid.PairGrid at 0x1edd26d14c8>

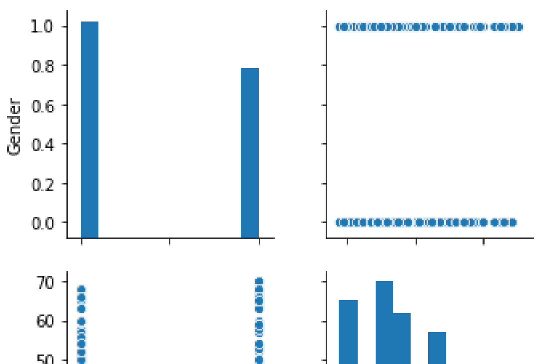


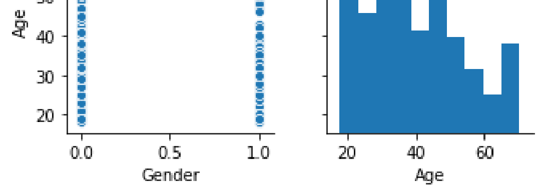
In [226]:

```
sns.pairplot(data,vars=['Gender','Age'])
```

Out[226]:

<seaborn.axisgrid.PairGrid at 0x1edd8ac8ec8>

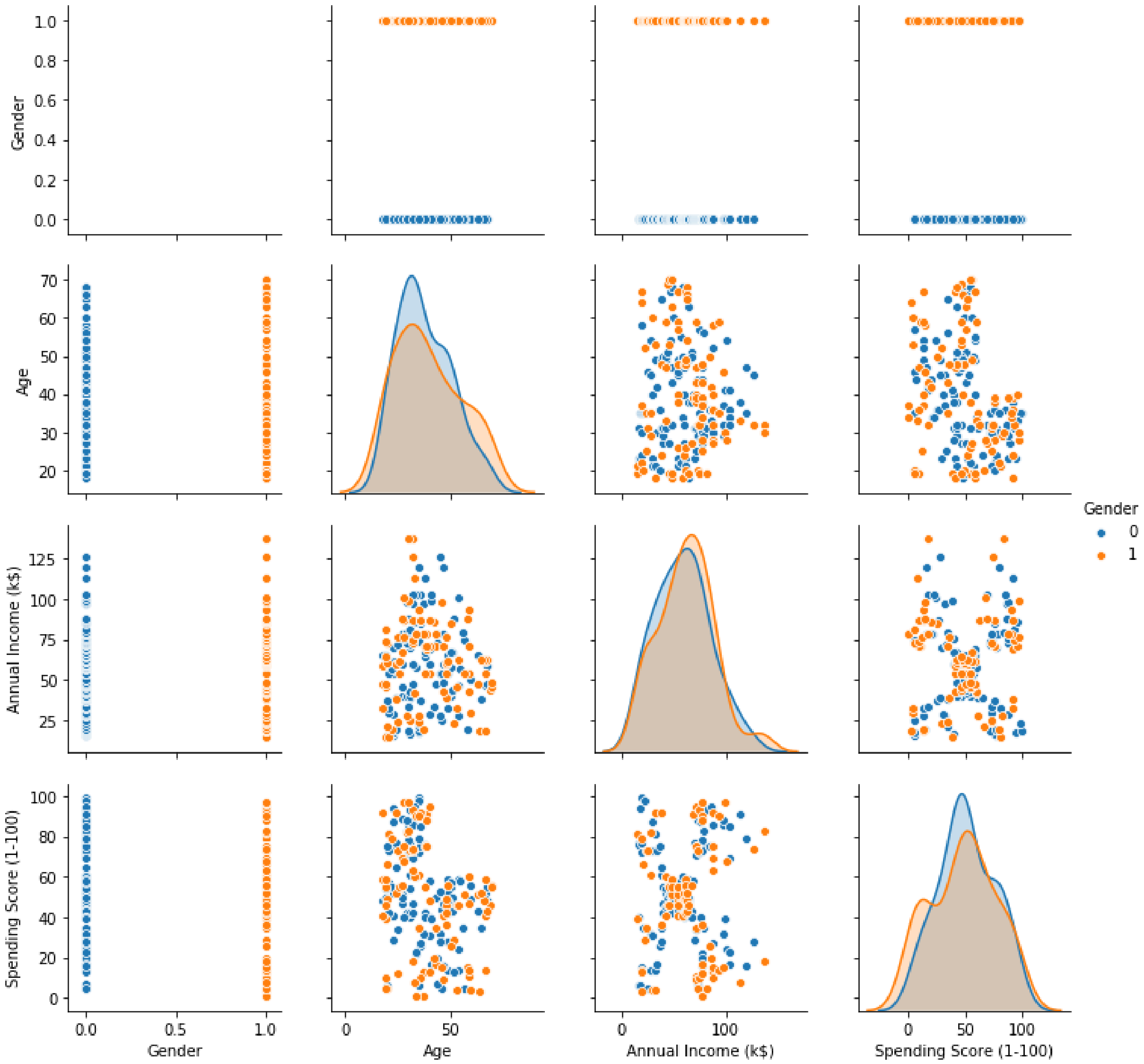




In [227]:

```
sns.pairplot(data,hue='Gender');
```

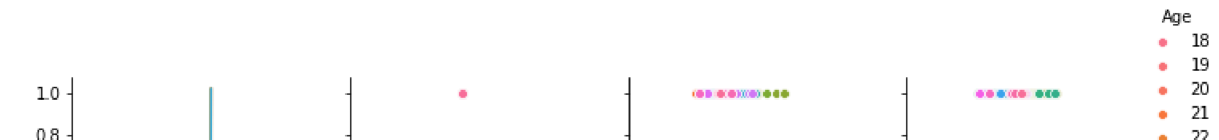
```
C:\Users\dell\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487: RuntimeWarning: invalid value encount
ered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\dell\AppData\Local\Continuum\anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.py:34: RuntimeWarning: invalid value enc
ountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```

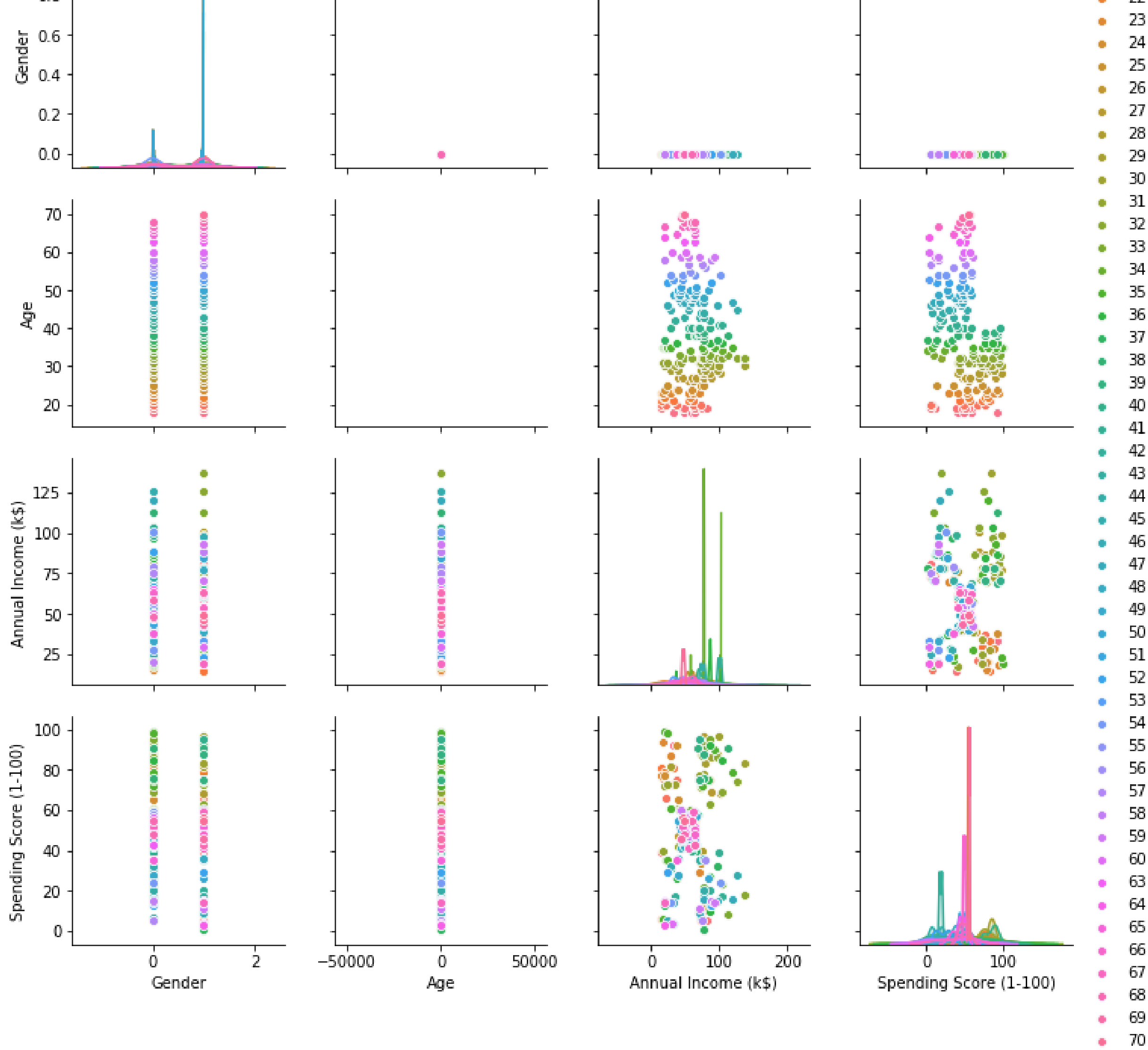


In [228]:

```
sns.pairplot(data,hue='Age');
```

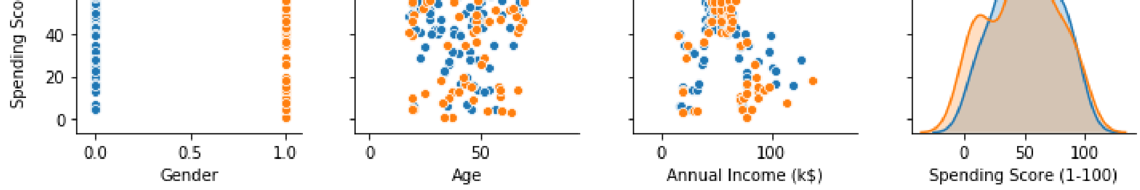
```
C:\Users\dell\AppData\Local\Continuum\anaconda3\lib\site-packages\numpy\core\_methods.py:140: RuntimeWarning: Degrees of freedom <= 0 for s
lice
  keepdims=keepdims)
C:\Users\dell\AppData\Local\Continuum\anaconda3\lib\site-packages\numpy\core\_methods.py:132: RuntimeWarning: invalid value encountered in d
ouble_scalars
  ret = ret.dtype.type(ret / rcount)
```





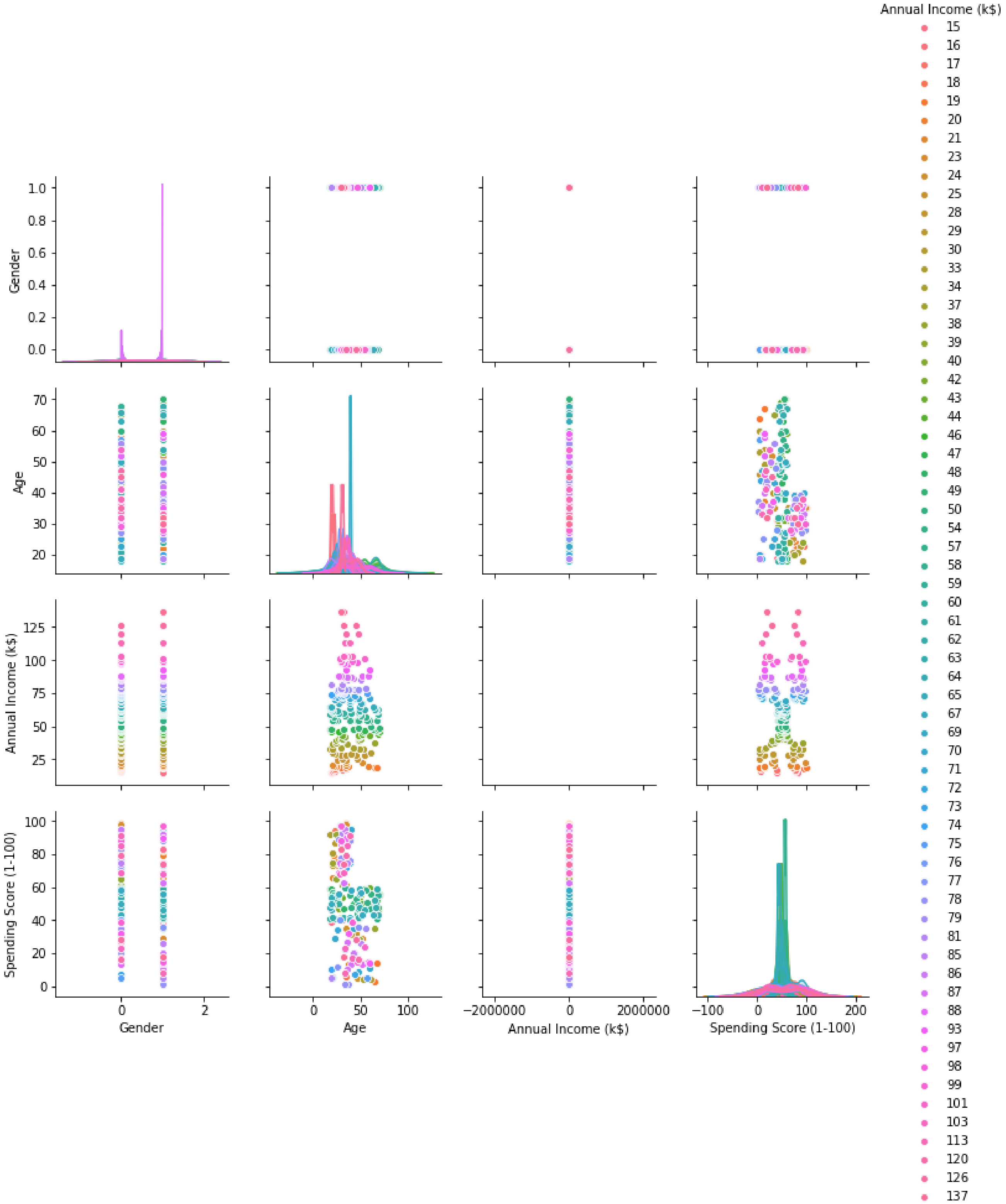
In [229]:

```
sns.pairplot(data,hue='Gender');
```



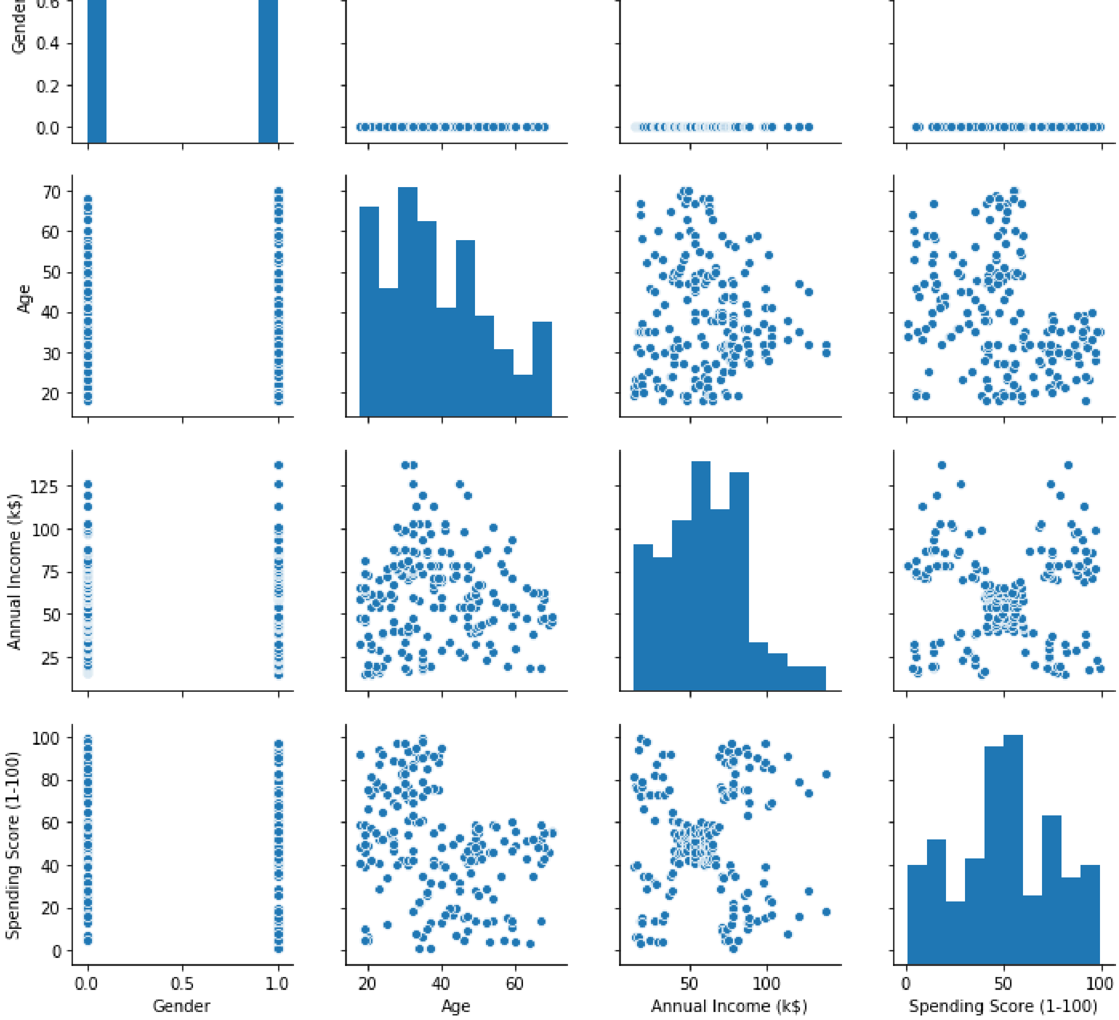
In [230]:

```python
sns.pairplot(data,hue='Annual Income (k$)');
```



In [231]:

```python
sns.pairplot(data);
```



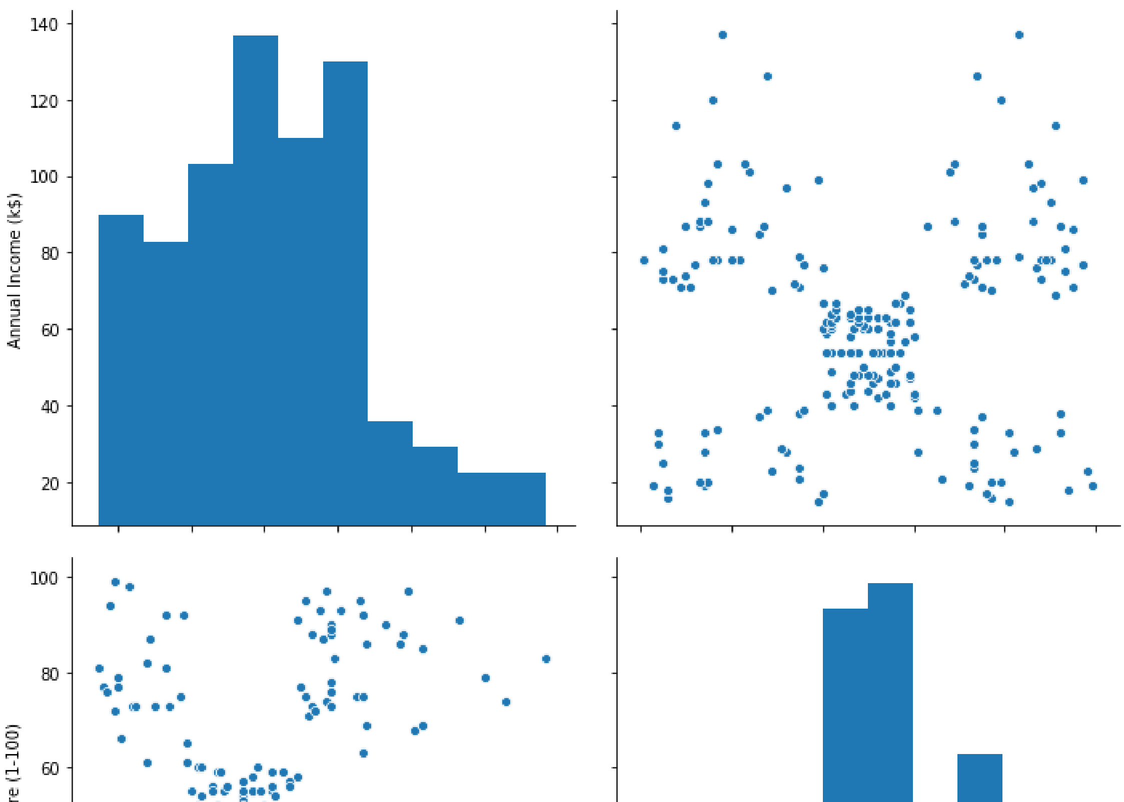
In [232]:

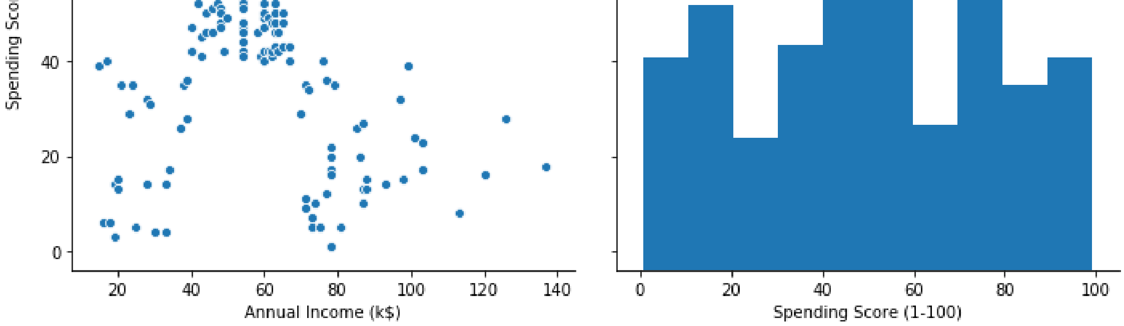
```
sns.pairplot(data,vars=['Annual Income (k$)','Spending Score (1-100)'], size=5)
```

C:\Users\dell\AppData\Local\Continuum\anaconda3\lib\site-packages\seaborn\axisgrid.py:2065: UserWarning: The `size` parameter has been renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)

Out[232]:

<seaborn.axisgrid.PairGrid at 0x1edd83b54c8>





In [233]:

```
import numpy as np
```

In [234]:

```
import pandas as pd
```

In [235]:

```
from sklearn.model_selection import train_test_split
```

In [236]:

```
from sklearn import metrics
```

In [237]:

```
from matplotlib import pyplot as plt
```

In [238]:

```
df = pd.read_csv("C:\\Users\\dell\\mall customer.csv")
```

In [239]:

```
x = df[['Age', 'Annual Income (k$)']]
y = df['CustomerID']
```

In [240]:

```
x
```

Out[240]:

	Age	Annual Income (k$)
0	19	15
1	21	15
2	20	16
3	23	16
4	31	17
...	...	...
195	35	120
196	45	126
197	32	126
198	32	137
199	30	137

200 rows × 2 columns

In [241]:

```
y
```

Out[241]:

```
0      1
1      2
2      3
3      4
4      5
      ...
195    196
196    197
197    198
198    199
199    200
Name: CustomerID, Length: 200, dtype: int64
```

In [242]:

```
from sklearn.model_selection import train_test_split
```

In [243]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

In [244]:

```
len(x_train)
```

Out[244]:

```
160
```

In [245]:

```
len(x_test)
```

Out[245]:

```
40
```

In [246]:

```
len(y_train)
```

Out[246]:

```
160
```

In [247]:

```
len(y_test)
```

Out[247]:

```
40
```

In [248]:

```
x_train
```

Out[248]:

	Age	Annual Income (k$)
48	29	40
138	19	74
31	21	30
26	45	28
124	23	70

	Age	Annual Income (k$)
...	...	...
139	35	74
136	44	73
117	49	65
67	68	48
194	47	120

160 rows × 2 columns

In [249]:

```
x_test
```

Out[249]:

	Age	Annual Income (k$)
5	22	17
181	32	97
44	49	39
93	40	60
162	19	81
81	38	54
70	70	49
97	27	60
140	57	75
119	50	67
64	63	48
88	34	58
82	67	54
25	29	28
195	35	120
111	19	63
36	42	34
61	19	46
157	30	78
129	38	71
4	31	17
118	51	67
78	23	54
196	45	126
13	24	20
45	24	39
154	47	78
30	60	30
7	23	18
189	36	103
83	46	54
185	30	99
99	20	61
156	37	78
163	31	81
86	55	57
146	48	77
171	28	87
172	36	87
176	58	88

In [250]:

```
y_train
```

Out[250]:

```
48     49
138    139
31     32
26     27
124    125
      ...
139    140
136    137
117    118
67     68
194    195
Name: CustomerID, Length: 160, dtype: int64
```

In [251]:

```
y_test
```

Out[251]:

```
5      6
181    182
44     45
93     94
162    163
81     82
70     71
97     98
140    141
119    120
64     65
88     89
82     83
25     26
195    196
111    112
36     37
61     62
157    158
129    130
4      5
118    119
78     79
196    197
13     14
45     46
154    155
30     31
7      8
189    190
83     84
185    186
99    100
156    157
163    164
86     87
146    147
171    172
172    173
176    177
Name: CustomerID, dtype: int64
```

In [252]:

```
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
```

In [253]:

```
clf.fit(x_train,y_train)
```

Out[253]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [254]:

```
clf.predict(x_test)
```

Out[254]:

```
array([  7.570234  , 179.19176315,  53.40190546,  99.10655254,
       145.48629194,  86.30525997,  73.74350716,  99.83790255,
       130.43469165, 113.61008489,  71.98500969,  95.13949605,
        84.67378687,  30.85174474, 228.52592056, 106.74486808,
        43.03420268,  70.15574554, 138.41055334, 122.89438251,
         7.06391476, 113.5538272 ,  87.14912537, 240.87715159,
        13.91462259,  54.80834779, 137.45417256,  33.4123589 ,
         9.66627764, 191.88054033,  85.85519843, 183.60888119,
       102.38400773, 138.01674949, 144.81119963,  91.80578317,
       135.24561354, 157.89378066, 157.44371911, 158.35835119])
```

In [255]:

```
y_test
```

Out[255]:

```
5        6
181    182
44      45
93      94
162    163
81      82
70      71
97      98
140    141
119    120
64      65
88      89
82      83
25      26
195    196
111    112
36      37
61      62
157    158
129    130
4        5
118    119
78      79
196    197
13      14
45      46
154    155
30      31
7        8
189    190
83      84
185    186
99     100
156    157
163    164
86      87
146    147
171    172
172    173
176    177
Name: CustomerID, dtype: int64
```