Customer Segmentation

Okino Kamali Leiba

Parameters

- Use case: Identify market segment from a larger population.
- Data: Really fake data from our really fake mall.
- Tools: Jupyter Notebook
- Cross-instrument: none
- Documentation: none

Table of Content

Introduction

Imports

- SYS module for access to variables and functions used or maintained by the interpreter
- Pandas for data analysis and manipulation
- Numpy for general array computations
- Matplotlib for creating static, animated, and interactive visualizations in Python
- Seaborn for Python data visualization based on Matplotlib
- Scikit-Learn for machine learning library that supports supervised and unsupervised learning

In [ ]:
```python
import sys, pandas as pd, numpy as np, matplotlib as plt, seaborn as sns
import warnings
warnings.filterwarnings("ignore")
ImportWarning.__suppress_context__
```

Out[ ]:
```
<member '__suppress_context__' of 'BaseException' objects>
```

In [ ]:
```python
file_path="/Users/Owner/source/vsc_repo/customer_segment_cookbook/mall_customers.cs
segment_data = pd.read_csv(file_path, delimiter=",", header=0, nrows=300, na_values
encoding="utf-8")
segment_data.columns
```

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

Exploratory Data Analysis

In [ ]:
```python
segment_data.head(5)
```

Out[ ]:

	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 [ ]:
```python
segment_data.tail(5)
```

Out[ ]:

	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 [ ]:
```python
segment_data.info
```

Out[ ]:

```
<bound method DataFrame.info of      CustomerID  Gender  Age  Annual Income (k$)  S
pending 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 x 5 columns]>
```

In [ ]:

```
if all(segment_data.isna()) or all(segment_data.isnull()):
    print("All Good")
else:
    print("Danger, Will Robinson")
```

All Good

In [ ]:

```
segment_data[["Age", "Annual Income (k$)", "Spending Score (1-100)"]].describe().ro
```

Out[ ]:

	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000	200.000	200.000
mean	38.850	60.560	50.200
std	13.969	26.265	25.824
min	18.000	15.000	1.000
25%	28.750	41.500	34.750
50%	36.000	61.500	50.000
75%	49.000	78.000	73.000
max	70.000	137.000	99.000

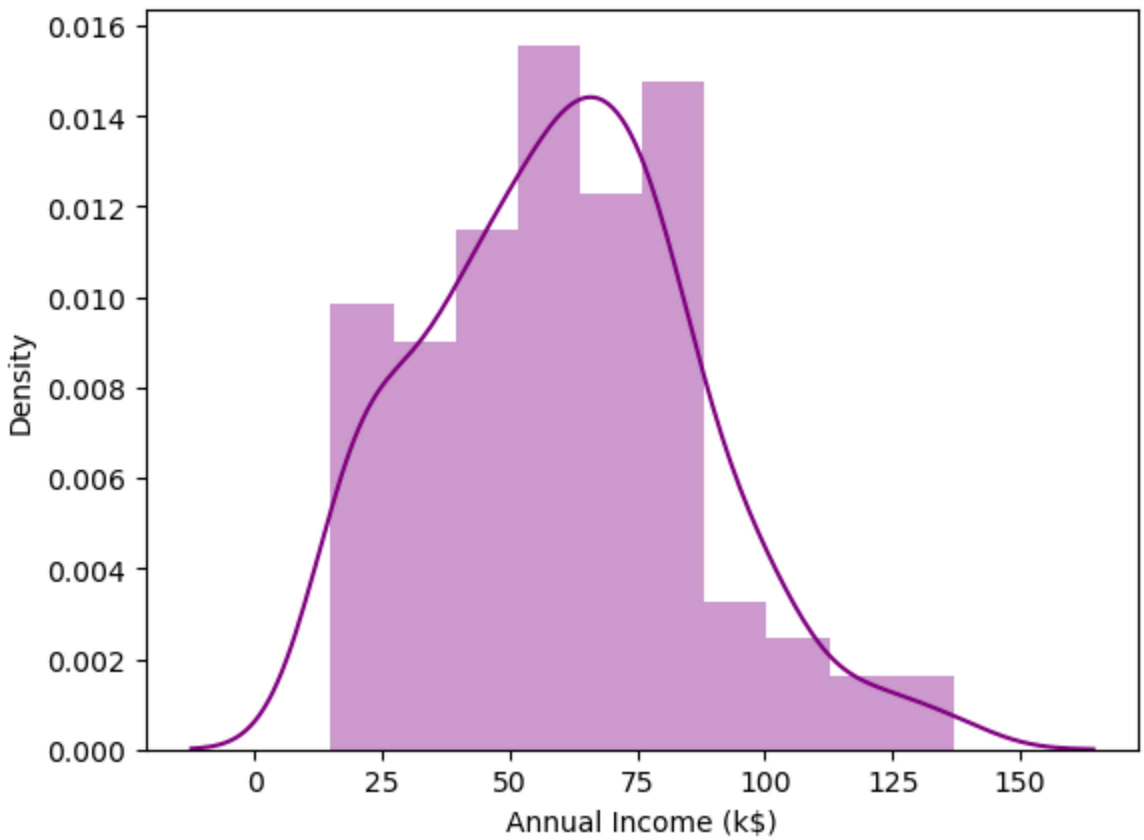
In [ ]:

```
count_data = segment_data['Spending Score (1-100)'].value_counts(normalize=True, so
count_data

# columns = ['Age', 'Annual Income (k$)','Spending Score (1-100)']
# count_data = []
# for iterable in columns:
#     count_data = segment_data[iterable].value_counts(normalize=True, sort=True, a
#     # print(count_data[iterable])
# count_data
```

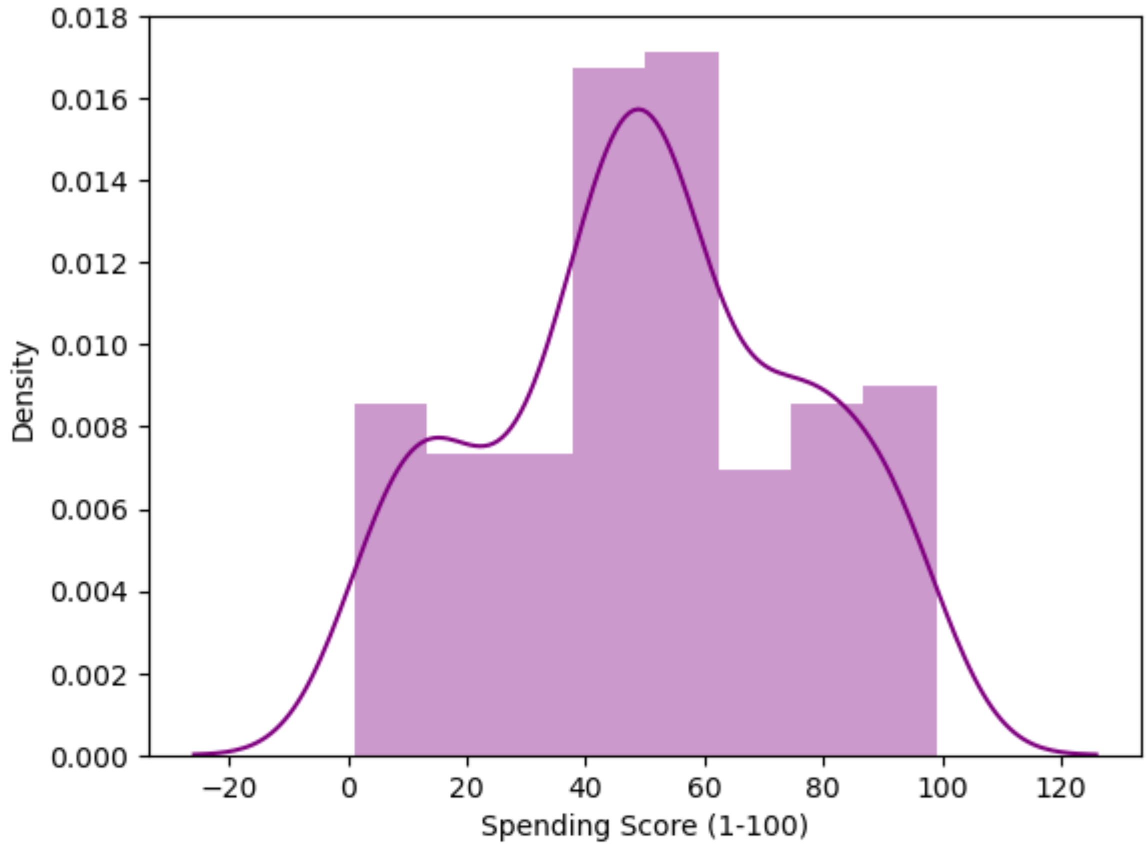
Out[ ]:
```
18    0.005
44    0.005
45    0.005
11    0.005
9     0.005
      ...
50    0.025
46    0.030
73    0.030
55    0.035
42    0.040
Name: Spending Score (1-100), Length: 84, dtype: float64
```

Univariate Analysis and Visualization

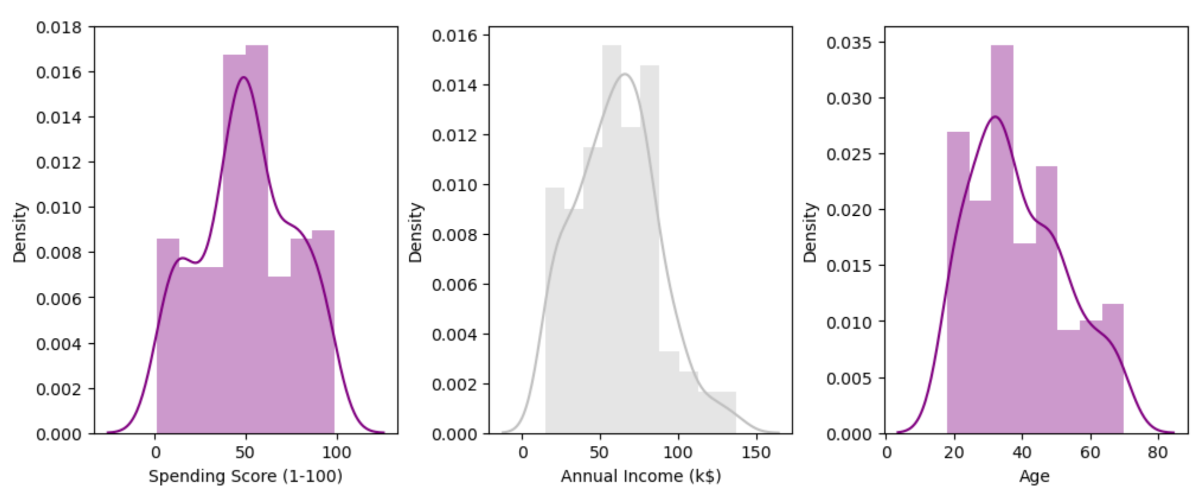
In [ ]:
```
sns.distplot(segment_data["Annual Income (k$)"], color="purple",hist=True);
```



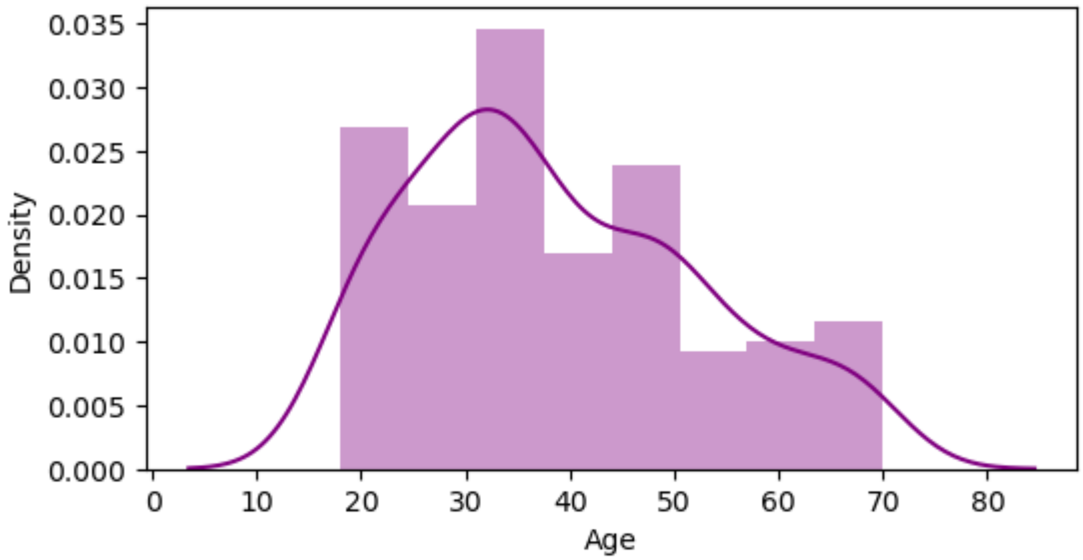
In [ ]:
```
sns.distplot(segment_data["Spending Score (1-100)"], kde=True, color="purple");
```

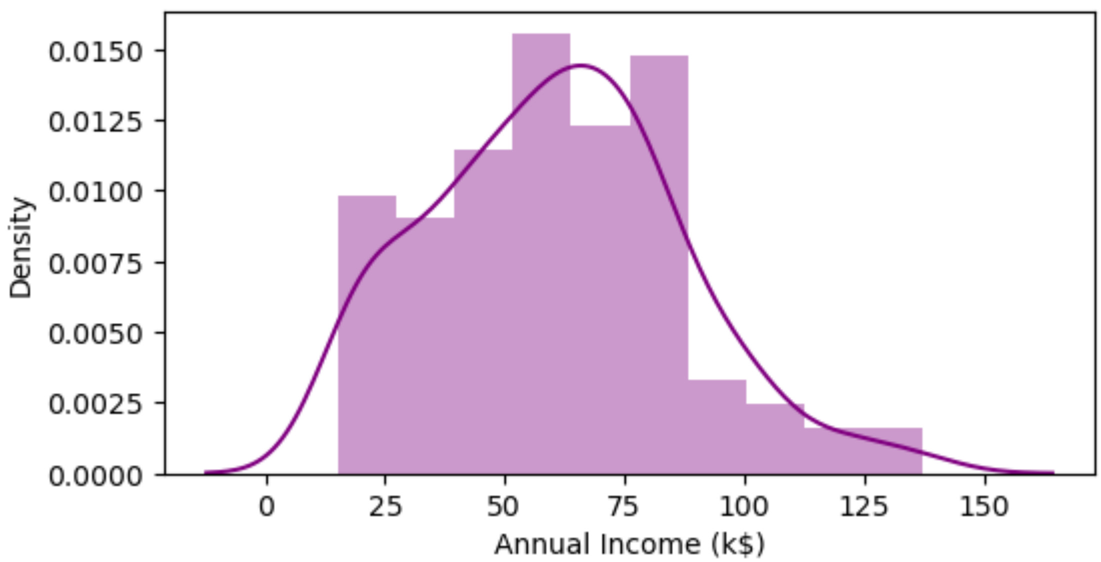


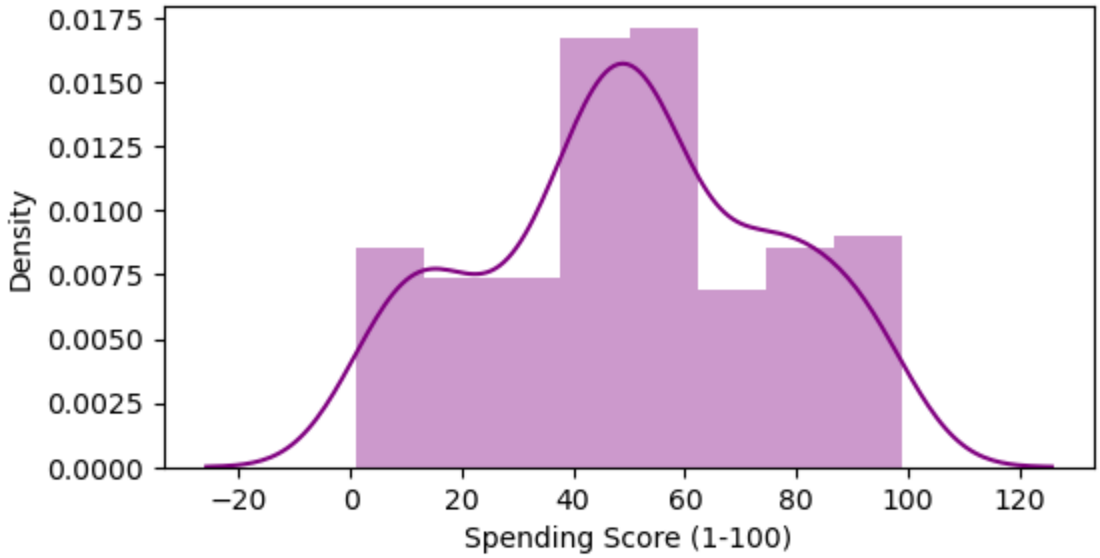
In [ ]:
```
fig, axe = plt.pyplot.subplots(nrows=1,ncols=3, constrained_layout=True, figsize=(1

sns.distplot(segment_data["Spending Score (1-100)"], kde=True, color="purple", ax=a
sns.distplot(segment_data["Annual Income (k$)"], color="silver",hist=True, ax=axe[1
sns.distplot(segment_data["Age"], color="purple",hist=True, ax=axe[2]);

# fig = plt.pyplotfigure(constrained_layout=True, figsize=(10, 4))
# subfigs = fig.subfigures(1, 2, wspace=0.07)
# sns.distplot(segment_data["Spending Score (1-100)"], kde=True, color="silver", ax
# sns.distplot(segment_data["Annual Income (k$)"], color="orange",hist=True, ax=sub
# sns.distplot(segment_data["Age"], color="orange",hist=True, ax=subfigs[2]);
```



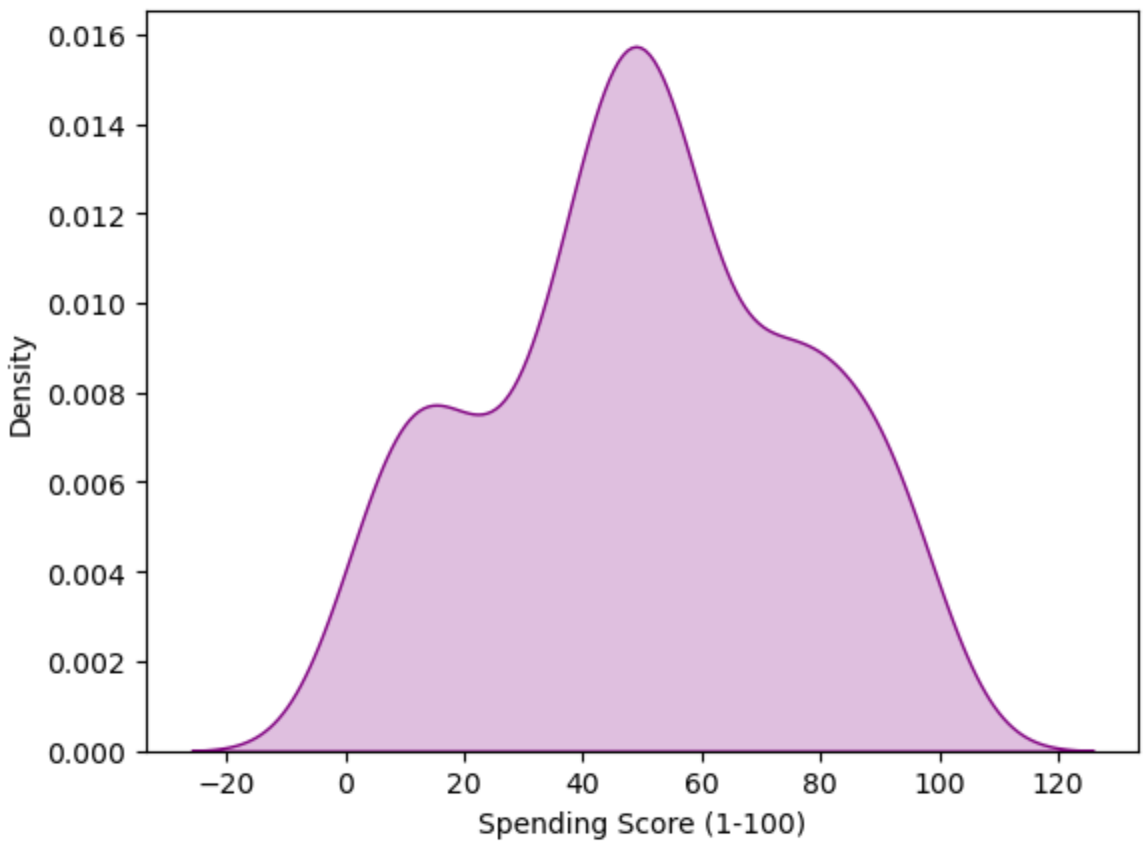
```
In [ ]: columns = ['Age', 'Annual Income (k$)','Spending Score (1-100)']
        for iterable in columns:
            fig = plt.pyplot.figure(figsize=(6, 3))
            # fig, axe = plt.pyplot.subplots(nrows=1,ncols=3)
            # fig, axe = plt.pyplot.subplots(nrows=1,ncols=3, constrained_layout=True, figs
            sns.distplot(segment_data[iterable], color="purple",hist=True);
```



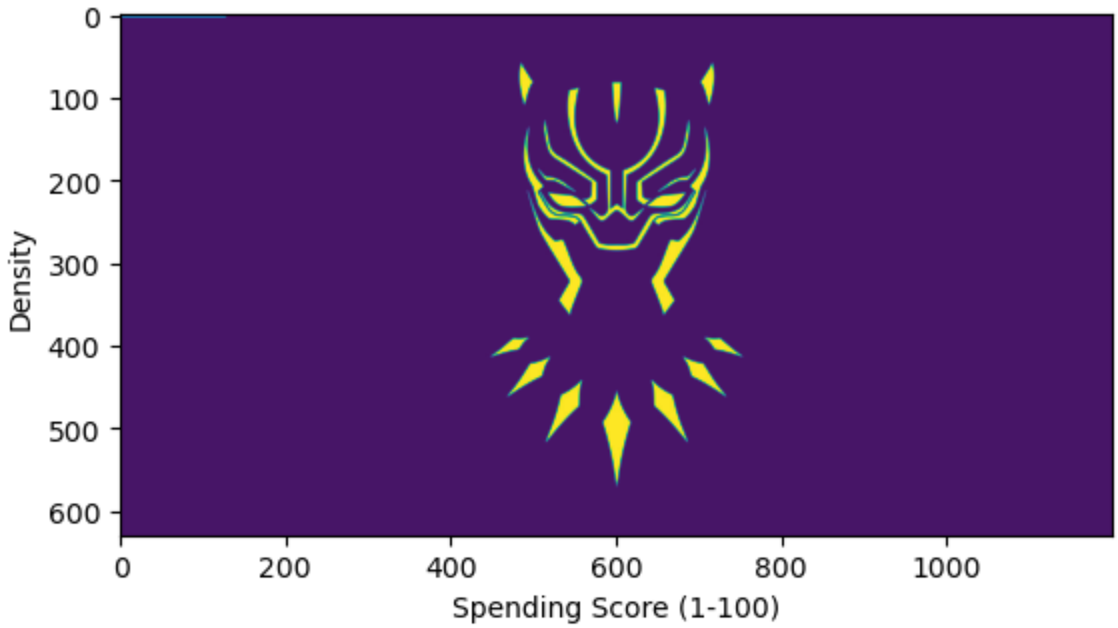




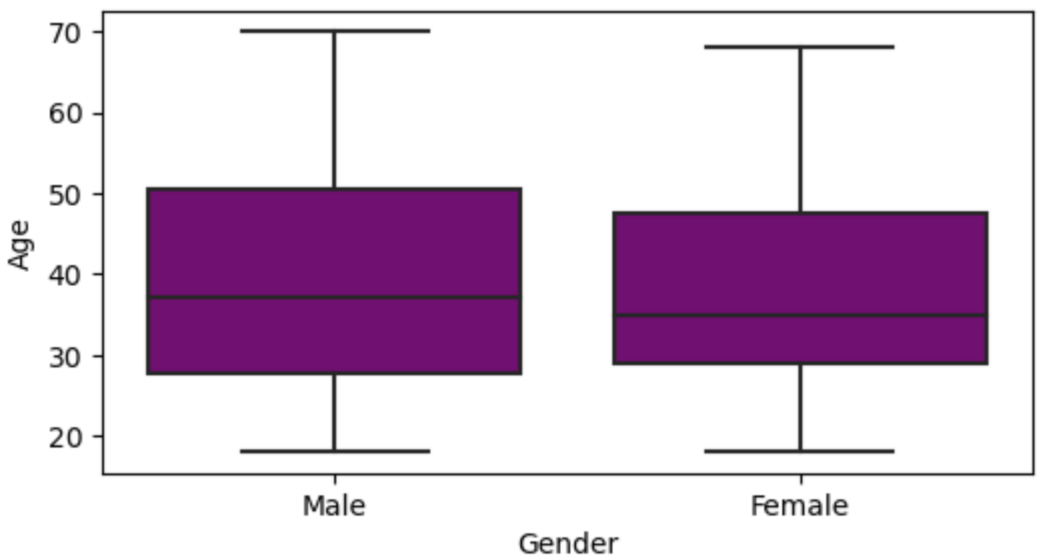
In [ ]:
```python
sns.kdeplot(segment_data["Spending Score (1-100)"], shade=True, color="purple");
```

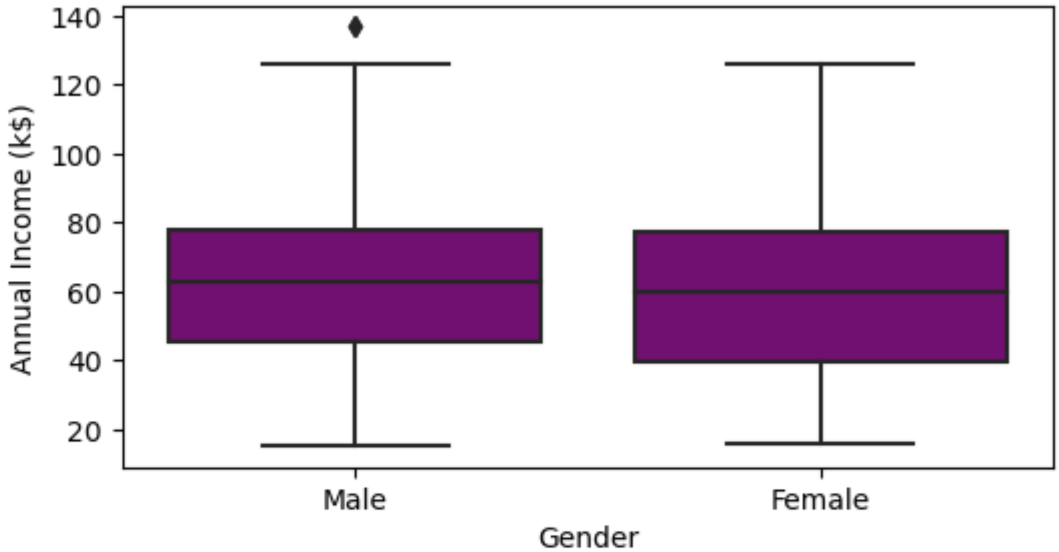


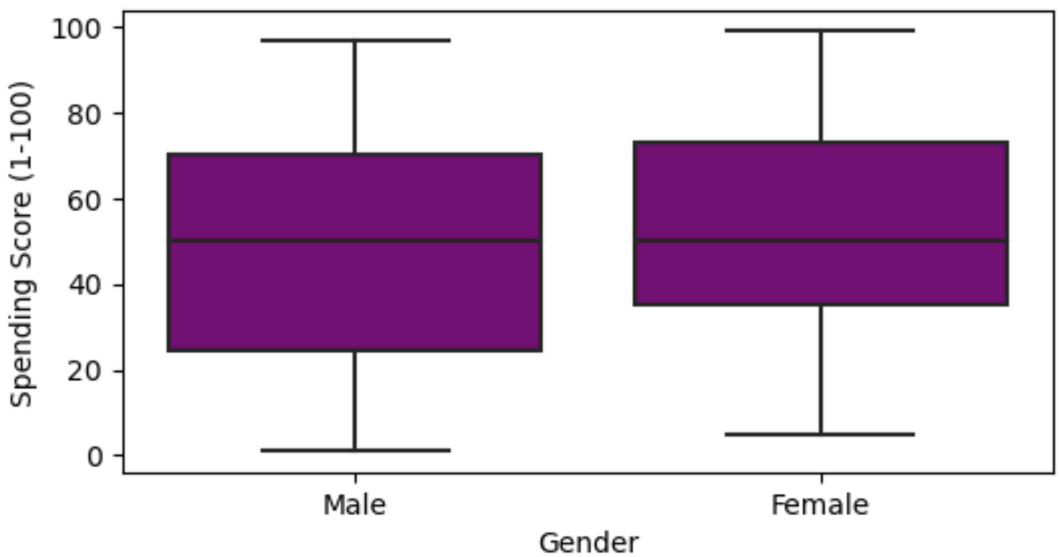
In [ ]:
```python
import matplotlib.image as mpimg
sns.kdeplot(segment_data["Spending Score (1-100)"], shade=True);
img = mpimg.imread("/Users/Owner/Pictures/Black Panther_F.jpg")
img = img[:, :, 0]
imgplot = plt.pyplot.imshow(img)
```



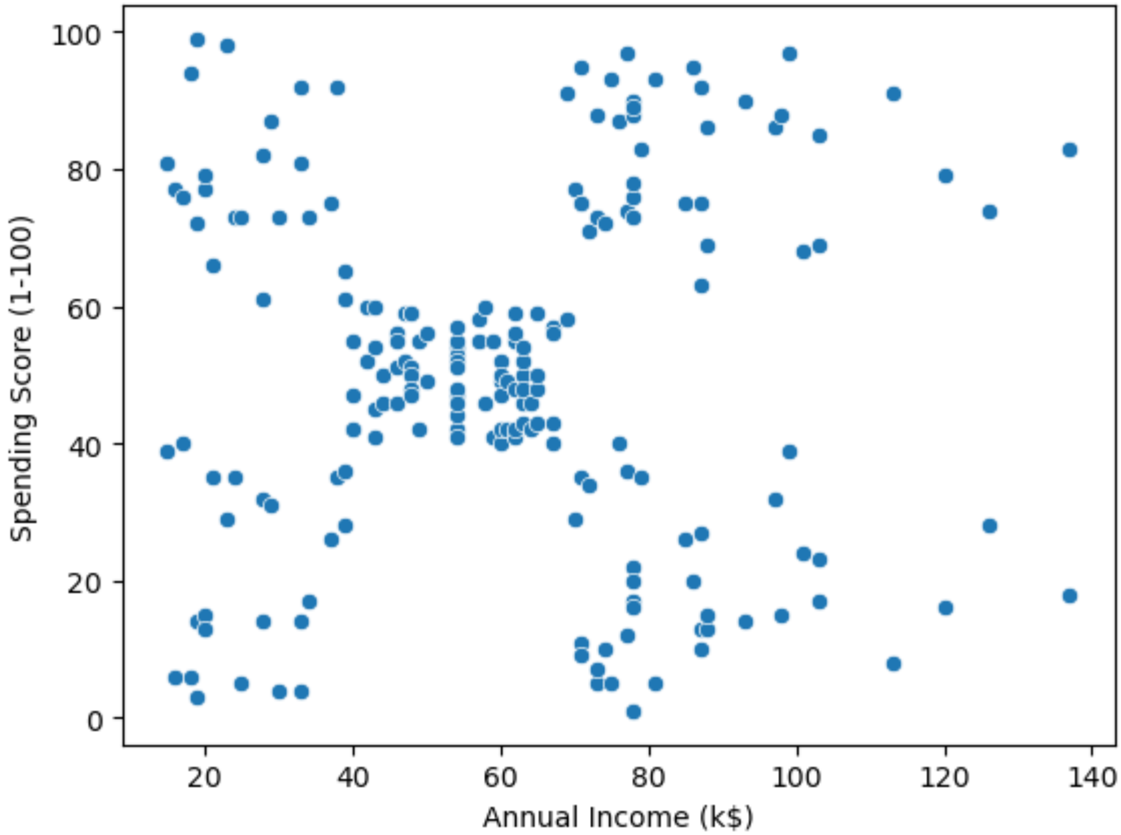
In [ ]:
```
columns = ['Age', 'Annual Income (k$)','Spending Score (1-100)']
for iterable in columns:
    fig = plt.pyplot.figure(figsize=(6, 3))
    #fig, axs = plt.pyplot.subplot()
    sns.boxplot(data=segment_data, x=segment_data["Gender"],  y=segment_data[iterab
```



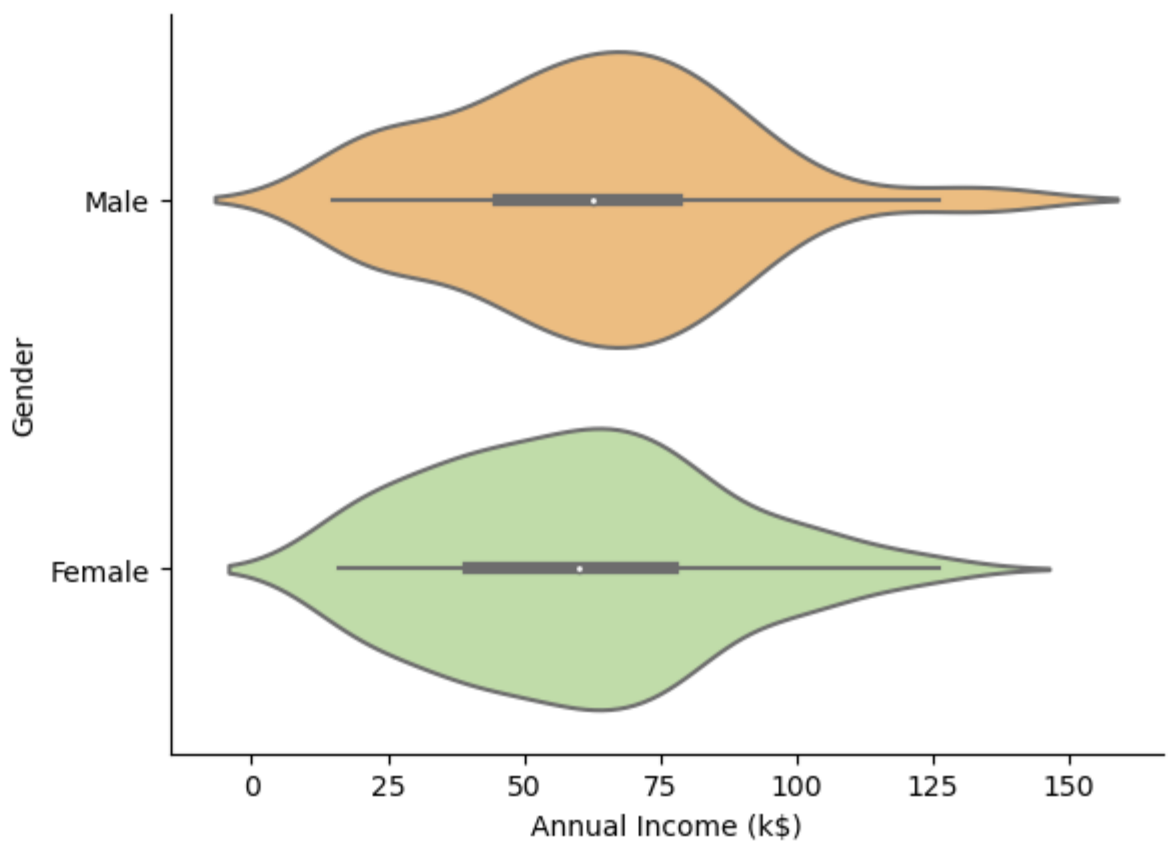




In [ ]:
```
sns.scatterplot(data=segment_data, x=segment_data["Annual Income (k$)"], y=segment_
```

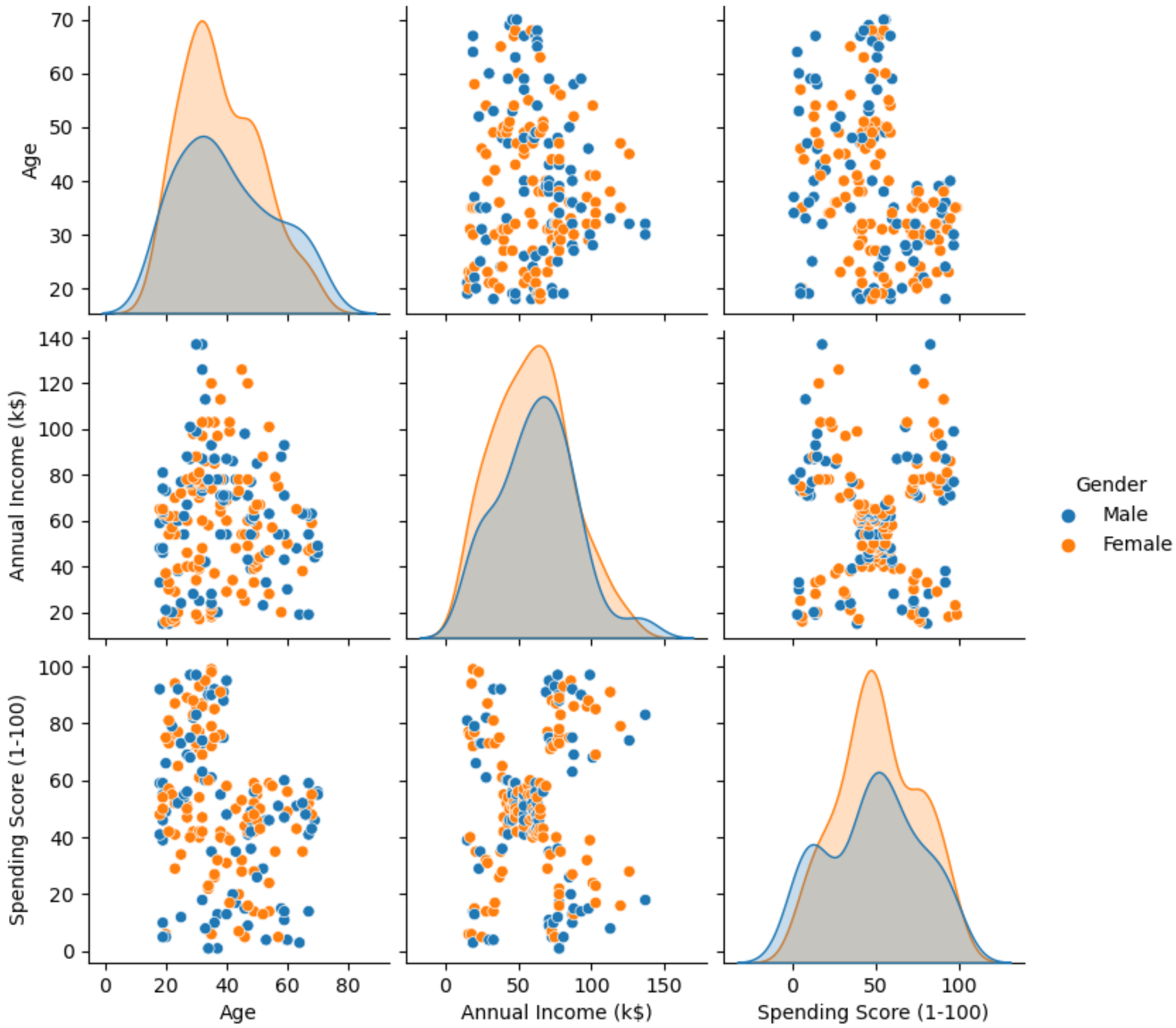


In [ ]:
```
sns.violinplot(data=segment_data, x="Annual Income (k$)", y="Gender", palette="Spec
sns.despine()
```



In [ ]:

```
drop_data = segment_data.drop("CustomerID", axis=1)
sns.pairplot(data=drop_data, hue="Gender", kind="scatter", dropna=True);
```



In [ ]:

```
mean_data = segment_data.groupby("Gender")['Age', 'Annual Income (k$)','Spending Sc
mean_data
```

Out[ ]:

	Age	Annual Income (k$)	Spending Score (1-100)
Gender			
Female	38.098	59.250	51.527
Male	39.807	62.227	48.511

In [ ]:

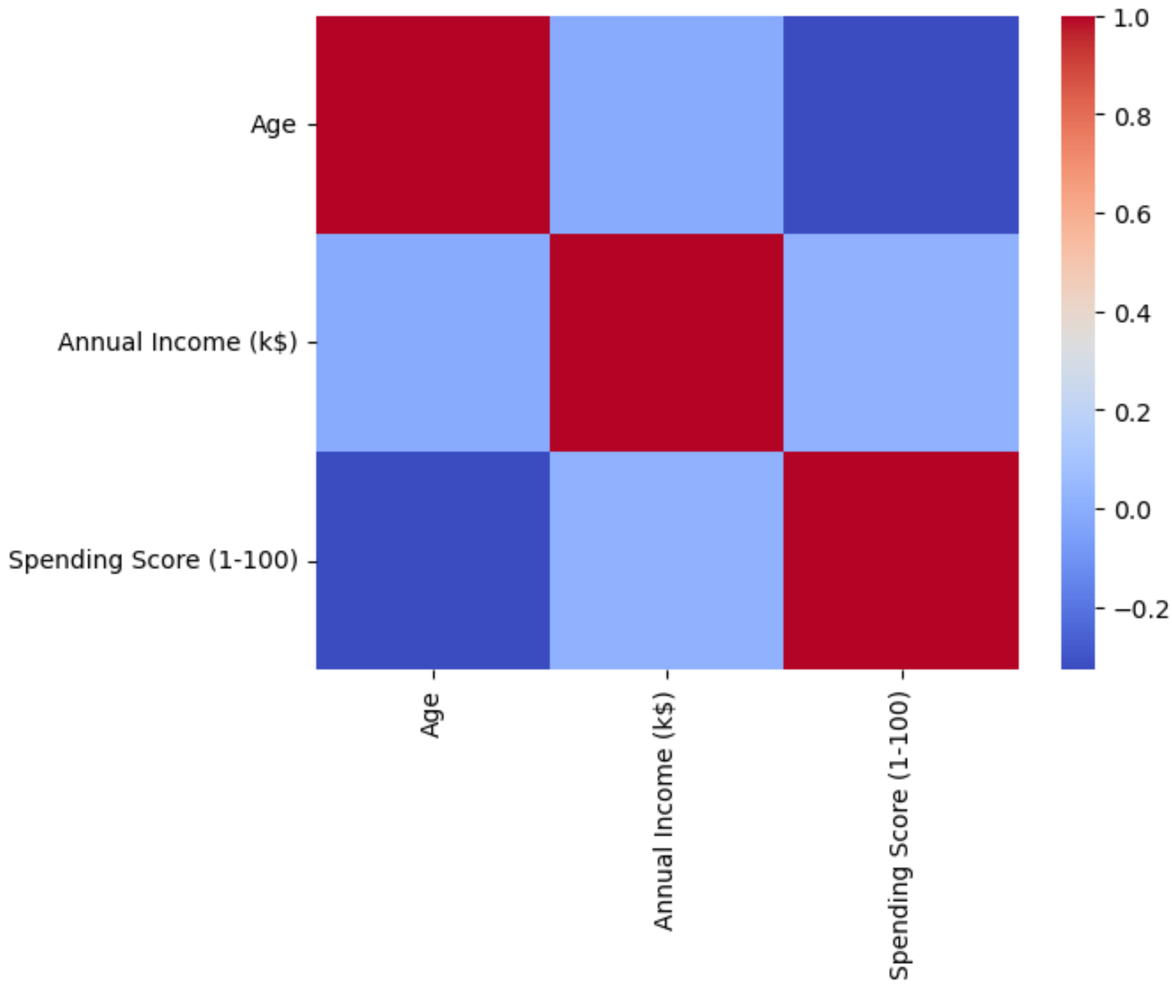
```
drop_data = segment_data.drop("CustomerID", axis=1)
drop_data.corr().round(decimals=3)
```

Out[ ]:

	Age	Annual Income (k$)	Spending Score (1-100)
Age	1.000	-0.012	-0.327
Annual Income (k$)	-0.012	1.000	0.010
Spending Score (1-100)	-0.327	0.010	1.000

```
In [ ]: sns.heatmap(drop_data.corr(),cmap="coolwarm")
```

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



Create Segments and Clusters (KMeans Clustering Algorithm)

Clustering - Univariate, Bivariate, and Multivariate

Univariate Clustering

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

```
In [ ]: uni_cluster = KMeans(init="k-means++", n_init=3, n_clusters=3)

        # segment_data["Annual Income (k$)"].array.reshape(1, -1)
        # uni_cluster = sklearn.cluster.k_means(segment_data["Annual Income (k$)"], n_clust
```

```
In [ ]: uni_cluster.fit(segment_data[["Annual Income (k$)"]])
        uni_cluster.labels_
        segment_data["Income Cluster"] = uni_cluster.labels_
        uni_cluster_data = segment_data
        uni_cluster_data.head(0)
```

Out[ ]:

CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Income Cluster

In [ ]:
```
# https://scikit-learn.org/stable/modules/clustering.html#k-means
uni_cluster.inertia_
```

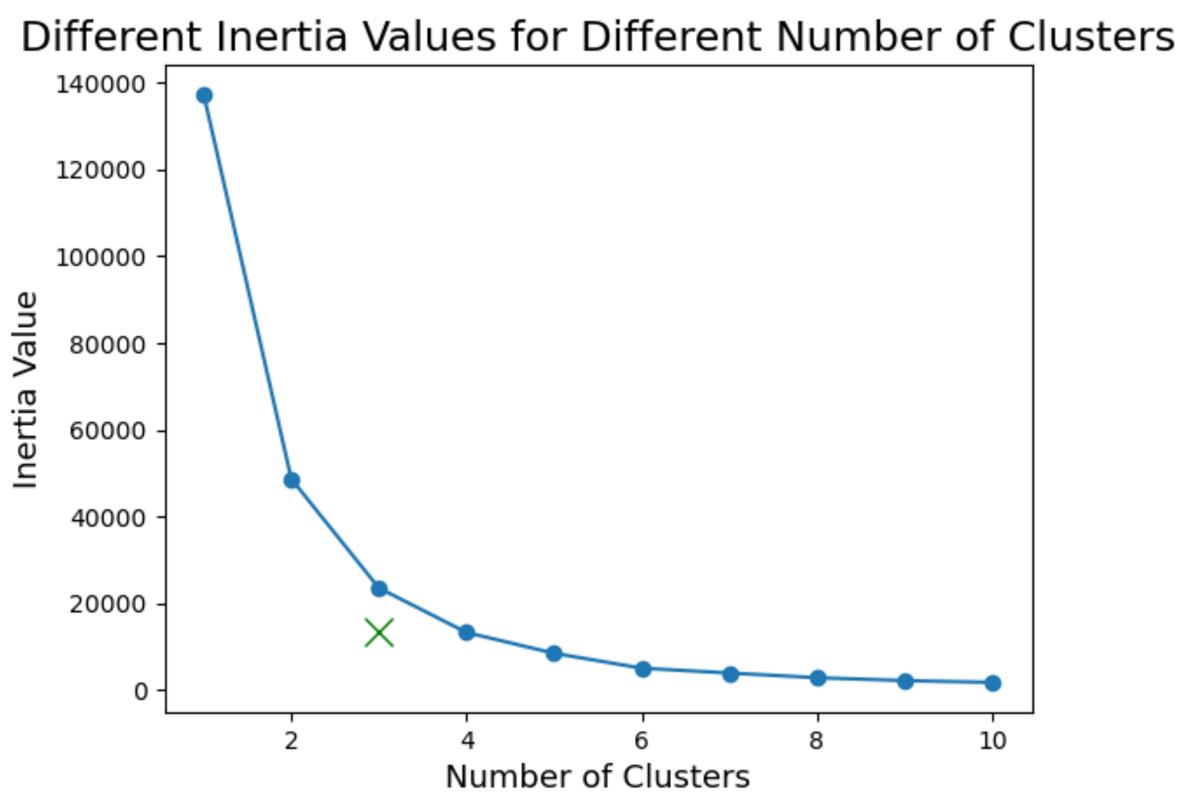
Out[ ]:
```
24361.259213759215
```

In [ ]:
```
uni_cluster.cluster_centers_
# best_fit = uni_cluster.fit_predict(segment_data[['Age', 'Annual Income (k$)','Spe
```

Out[ ]:
```
array([[108.18181818],
       [ 69.75      ],
       [ 33.48648649]])
```

In [ ]:
```
inertia_scores = []
for iterable in range(1,11):
        uni_kmeans = KMeans(n_clusters=iterable ).fit(uni_cluster_data[["Annual Inc
        inertia_scores.append(uni_kmeans.inertia_)
# inertia_scores = [iterable.fit(segment_data[["Annual Income (k$)"]]) for iterable
```

In [ ]:
```
plt.pyplot.plot(range(1,11),inertia_scores)
plt.pyplot.scatter(range(1,11),inertia_scores)
plt.pyplot.plot(3, inertia_scores[3], marker="x", color="green", linestyle="dashed"
plt.pyplot.xlabel("Number of Clusters", size=13)
plt.pyplot.ylabel("Inertia Value", size=13)
plt.pyplot.title("Different Inertia Values for Different Number of Clusters", size=
```



Univariate Analysis

In [ ]:
```
uni_mean = segment_data.groupby("Income Cluster")['Age', 'Annual Income (k$)','Spen
uni_mean
```

Out[ ]:

Income Cluster	Age	Annual Income (k$)	Spending Score (1-100)
0	37.545	108.182	52.000
1	38.663	69.750	49.798
2	39.500	33.486	50.230

In [ ]:
```
uni_count = uni_cluster_data["Income Cluster"].value_counts(normalize=True, sort=Tr
uni_count
```

Out[ ]:
```
0    0.11
2    0.37
1    0.52
Name: Income Cluster, dtype: float64
```

Bivariate Clustering

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

In [ ]:
```
bi_cluster = KMeans(init="k-means++", n_init=3, n_clusters=5)
```

In [ ]:
```
bi_cluster.fit(segment_data[["Annual Income (k$)", "Spending Score (1-100)"]])
bi_cluster.labels_
segment_data["Income and Spending Cluster"] = bi_cluster.labels_
bi_cluster_data = segment_data
bi_cluster_data.head(0)
```

Out[ ]:

CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Income Cluster	Income and Spending Cluster

In [ ]:
```
# https://scikit-learn.org/stable/modules/clustering.html#k-means
bi_cluster.inertia_
```

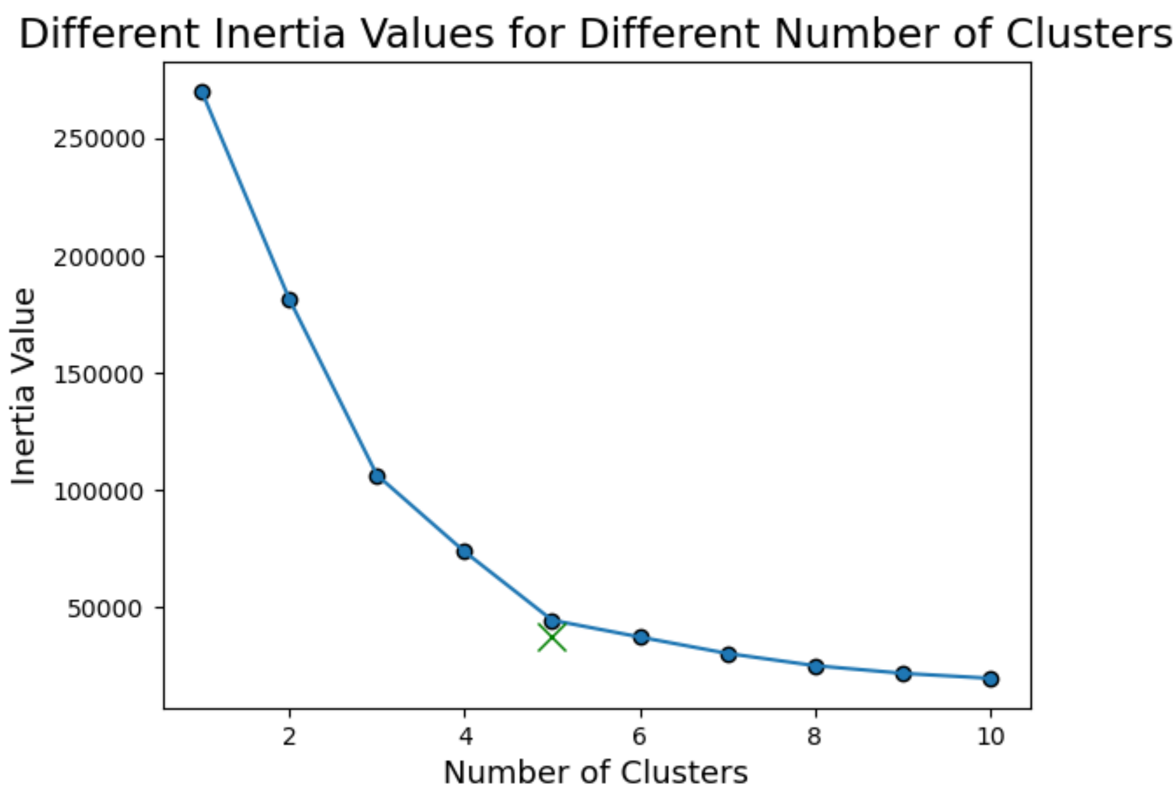
Out[ ]:
```
44448.45544793371
```

In [ ]:
```
bi_cluster.cluster_centers_
```

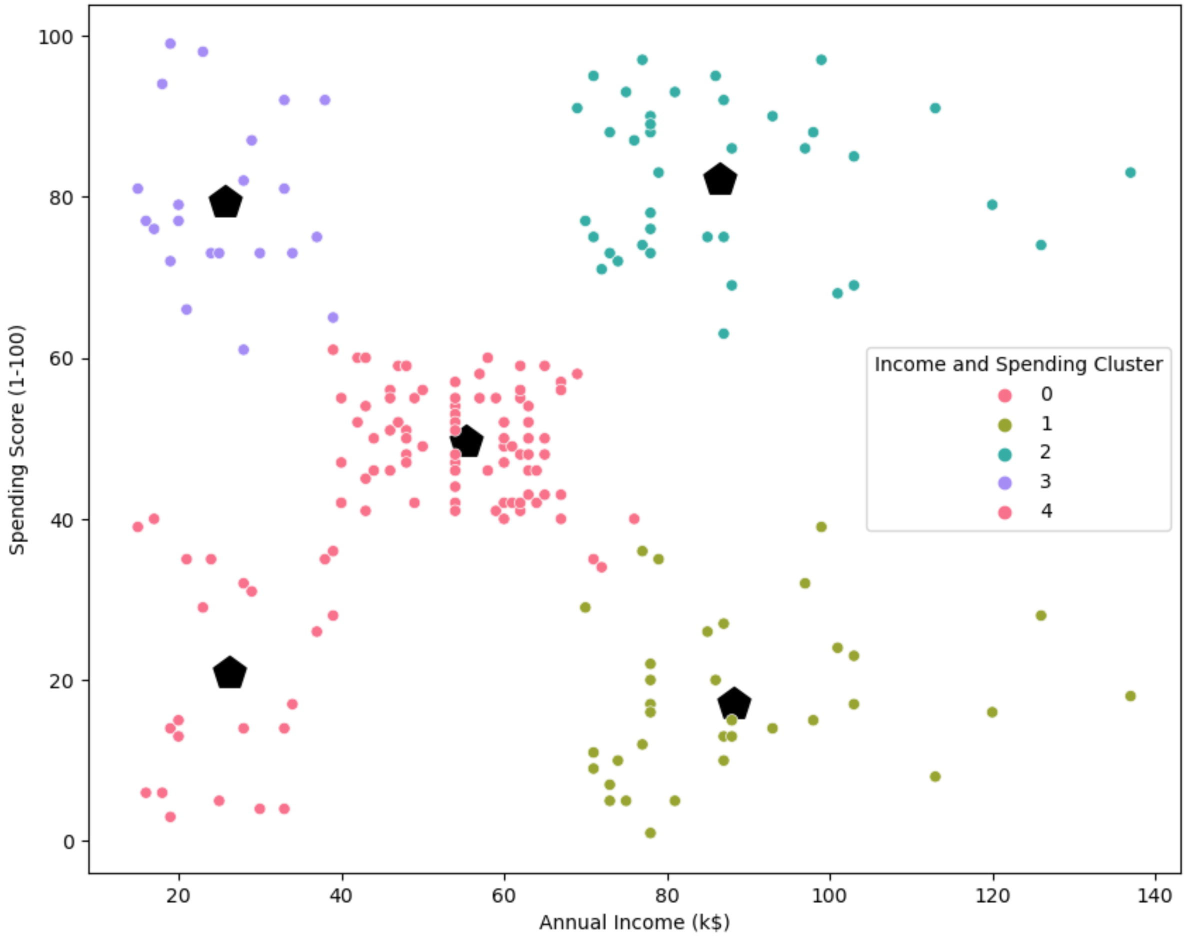
Out[ ]:
```
array([[55.2962963 , 49.51851852],
       [88.2       , 17.11428571],
       [86.53846154, 82.12820513],
       [25.72727273, 79.36363636],
       [26.30434783, 20.91304348]])
```

In [ ]:
```
bi_inertia_scores = []
for iterable in range(1,11):
        bi_kmeans = KMeans(n_clusters=iterable ).fit(bi_cluster_data[["Annual Incom
        bi_inertia_scores.append(bi_kmeans.inertia_)
```

In [ ]:
```
plt.pyplot.plot(range(1,11),bi_inertia_scores)
plt.pyplot.scatter(range(1,11),bi_inertia_scores, edgecolors="black")
plt.pyplot.plot(5, bi_inertia_scores[5], marker="x", color="green", linestyle="dash
plt.pyplot.xlabel("Number of Clusters", size=13)
plt.pyplot.ylabel("Inertia Value", size=13)
plt.pyplot.title("Different Inertia Values for Different Number of Clusters", size=
```



In [ ]:
```
cluster_centriod = pd.DataFrame(bi_cluster.cluster_centers_)
cluster_centriod.columns = ["x","y"]
plt.pyplot.figure(figsize=(10,8))
plt.pyplot.scatter(x=cluster_centriod["x"], y=cluster_centriod["y"], s=300, color="
sns.scatterplot(data=bi_cluster_data, x="Annual Income (k$)", y="Spending Score (1-
```



Bivariate Analysis

In [ ]:
```
pd.crosstab(index=segment_data["Income and Spending Cluster"], columns=segment_data
```

Out[ ]:

Gender	Female	Male
Income and Spending Cluster		
0	0.240	0.165
1	0.080	0.095
2	0.105	0.090
3	0.065	0.045
4	0.070	0.045

In [ ]:
```
pd.crosstab(index=segment_data["Income and Spending Cluster"], columns=segment_data
```

Out[ ]:

Age	18	19	20	21	22	23	24	25	26	27	...	59	60	63
Income and Spending Cluster														
0	0.015	0.025	0.005	0.010	0.005	0.010	0.005	0.005	0.01	0.02	...	0.01	0.010	0.01
1	0.000	0.010	0.005	0.000	0.000	0.005	0.000	0.005	0.00	0.00	...	0.01	0.000	0.00
2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.00	0.01	...	0.00	0.000	0.00
3	0.005	0.000	0.010	0.015	0.010	0.015	0.015	0.005	0.00	0.00	...	0.00	0.000	0.00
4	0.000	0.005	0.005	0.000	0.000	0.000	0.000	0.000	0.00	0.00	...	0.00	0.005	0.00

5 rows × 51 columns

In [ ]:
```
bi_mean = segment_data.groupby("Income and Spending Cluster")['Age', 'Annual Income
bi_mean
```

Out[ ]:

	Age	Annual Income (k$)	Spending Score (1-100)
Income and Spending Cluster			
0	42.716	55.296	49.519
1	41.114	88.200	17.114
2	32.692	86.538	82.128
3	25.273	25.727	79.364
4	45.217	26.304	20.913

In [ ]:
```
bi_count = bi_cluster_data["Income and Spending Cluster"].value_counts(normalize=Tr
bi_count
```

Out[ ]:
```
3    0.110
4    0.115
1    0.175
2    0.195
0    0.405
Name: Income and Spending Cluster, dtype: float64
```

Multivariate Clustering

In [ ]:
```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
```

In [ ]:
```
encode_cat = pd.get_dummies(segment_data, drop_first=True)
encode_cat.head()
```

Out[ ]:

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

In [ ]:

```
standard_cat = encode_cat[["Age", "Annual Income (k$)", "Spending Score (1-100)", "
standard_cat.head(5)
```

Out[ ]:

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

In [ ]:

```
standardized_data = pd.DataFrame(scale.fit_transform(standard_cat))
standardized_data = standardized_data.set_axis(["Age", "Annual Income (k$)", "Spend
standardized_data.head(5)
```

Out[ ]:

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Male
0	-1.424569	-1.738999	-0.434801	1.128152
1	-1.281035	-1.738999	1.195704	1.128152
2	-1.352802	-1.700830	-1.715913	-0.886405
3	-1.137502	-1.700830	1.040418	-0.886405
4	-0.563369	-1.662660	-0.395980	-0.886405

In [ ]:

```
from sklearn.cluster import KMeans
```

In [ ]:

```
multi_cluster = KMeans(init="k-means++", n_init=3, n_clusters=4)
```

In [ ]:

```
multi_cluster.fit(standardized_data[["Annual Income (k$)", "Spending Score (1-100)"
multi_cluster.labels_
standardized_data["Income and Spending Cluster"] = multi_cluster.labels_
multi_cluster_data = standardized_data
multi_cluster_data.head(5)
```

Out[ ]:

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Male	Income and Spending Cluster
0	-1.424569	-1.738999	-0.434801	1.128152	1
1	-1.281035	-1.738999	1.195704	1.128152	0
2	-1.352802	-1.700830	-1.715913	-0.886405	1
3	-1.137502	-1.700830	1.040418	-0.886405	0
4	-0.563369	-1.662660	-0.395980	-0.886405	1

In [ ]:

```
# https://scikit-learn.org/stable/modules/clustering.html#k-means
multi_cluster.inertia_
```

Out[ ]:

```
109.22822707921347
```

In [ ]:

```
multi_cluster.cluster_centers_
```

Out[ ]:

```
array([[-1.32954532,  1.13217788],
       [-0.47298347, -0.26414036],
       [ 1.00919971, -1.22553537],
       [ 0.99158305,  1.23950275]])
```

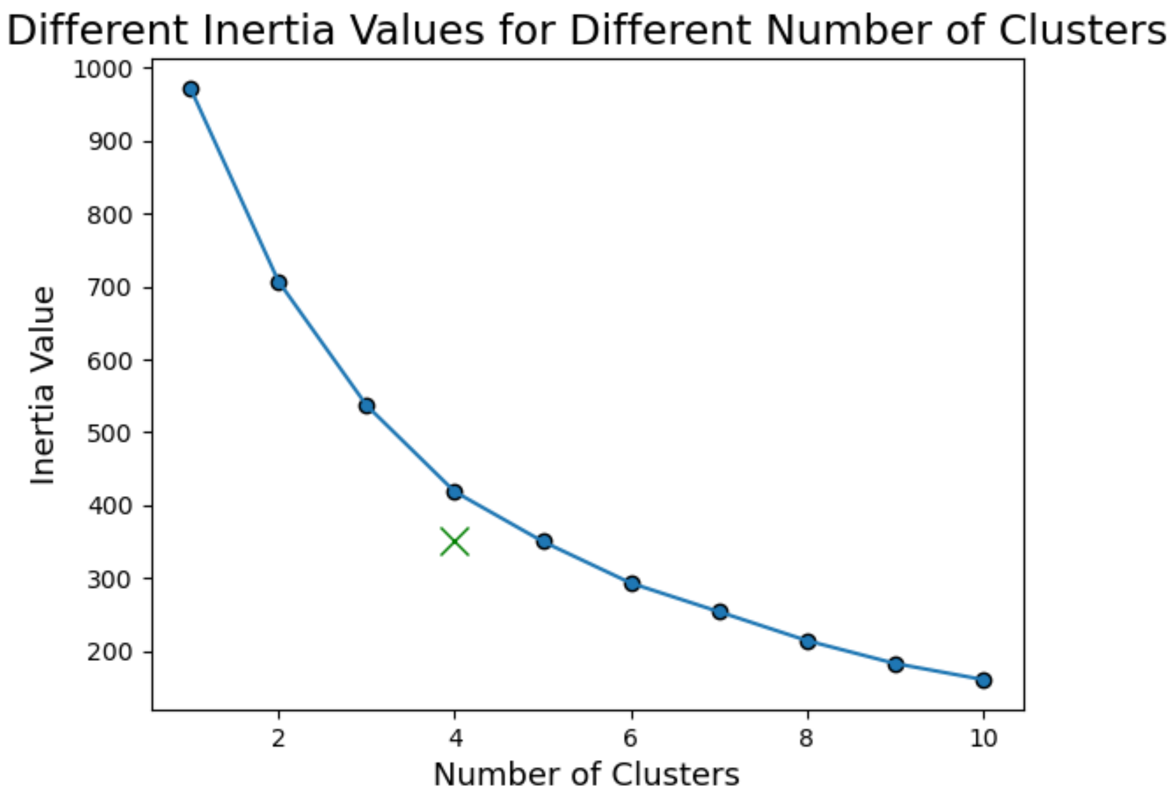
In [ ]:

```
multi_inertia_scores = []
for iterable in range(1,11):
        multi_kmeans = KMeans(n_clusters=iterable ).fit(standardized_data)
        multi_inertia_scores.append(multi_kmeans.inertia_)
```

In [ ]:

```
plt.pyplot.plot(range(1,11),multi_inertia_scores)
plt.pyplot.scatter(range(1,11),multi_inertia_scores, edgecolors="black")
plt.pyplot.plot(4, multi_inertia_scores[4], marker="x", color="green", linestyle="d
plt.pyplot.xlabel("Number of Clusters", size=13)
plt.pyplot.ylabel("Inertia Value", size=13)
plt.pyplot.title("Different Inertia Values for Different Number of Clusters", size=


# cluster_centriod = pd.DataFrame(multi_cluster.cluster_centers_)
# cluster_centriod.columns = ["w","x","y","z"]
# plt.pyplot.figure(figsize=(10,8))
# plt.pyplot.scatter(x=cluster_centriod["w"], y=cluster_centriod["x"], s=300, color
# sns.scatterplot(data=multi_cluster_data, x=cluster_centriod["w"], y=cluster_centr
```

```
The history saving thread hit an unexpected error (OperationalError('database or di
sk is full')).History will not be written to the database.
```



Mulitvariate Analysis

In [ ]:
```
from statsmodels.multivariate.manova import MANOVA
MANOVA_data = encode_cat.set_axis(["Customer_Id", "Age", "Annual_Income", "Spending
fit_data = MANOVA.from_formula("Annual_Income + Spending_Score ~ Age + Gender_Male"
print(fit_data.mv_test())
```

```
                 Multivariate linear model
==============================================================

--------------------------------------------------------------
       Intercept        Value  Num DF  Den DF   F Value  Pr > F
--------------------------------------------------------------
          Wilks' lambda 0.3893 2.0000 196.0000 153.7344 0.0000
         Pillai's trace 0.6107 2.0000 196.0000 153.7344 0.0000
 Hotelling-Lawley trace 1.5687 2.0000 196.0000 153.7344 0.0000
    Roy's greatest root 1.5687 2.0000 196.0000 153.7344 0.0000
--------------------------------------------------------------

--------------------------------------------------------------
           Age           Value  Num DF  Den DF  F Value Pr > F
--------------------------------------------------------------
           Wilks' lambda 0.8943 2.0000 196.0000 11.5775 0.0000
          Pillai's trace 0.1057 2.0000 196.0000 11.5775 0.0000
  Hotelling-Lawley trace 0.1181 2.0000 196.0000 11.5775 0.0000
     Roy's greatest root 0.1181 2.0000 196.0000 11.5775 0.0000
--------------------------------------------------------------

--------------------------------------------------------------
       Gender_Male       Value  Num DF  Den DF  F Value Pr > F
--------------------------------------------------------------
           Wilks' lambda 0.9951 2.0000 196.0000  0.4874 0.6150
          Pillai's trace 0.0049 2.0000 196.0000  0.4874 0.6150
  Hotelling-Lawley trace 0.0050 2.0000 196.0000  0.4874 0.6150
     Roy's greatest root 0.0050 2.0000 196.0000  0.4874 0.6150
==============================================================
```

Multivariate 3D Visualization

In [ ]:

```
import statsmodels.api as sm
from mpl_toolkits.mplot3d import Axes3D
x = encode_cat[["Annual Income (k$)", "Spending Score (1-100)"]]
y = encode_cat[["Age"]]
x = sm.add_constant(x)
ols_data = sm.OLS(y,x).fit()
ols_data.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Age	**R-squared:**	0.107
Model:	OLS	**Adj. R-squared:**	0.098
Method:	Least Squares	**F-statistic:**	11.82
Date:	Thu, 12 Jan 2023	**Prob (F-statistic):**	1.42e-05
Time:	23:08:19	**Log-Likelihood:**	-799.32
No. Observations:	200	**AIC:**	1605.
Df Residuals:	197	**BIC:**	1615.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	48.0284	2.974	16.148	0.000	42.163	53.894
Annual Income (k$)	-0.0049	0.036	-0.136	0.892	-0.075	0.066
Spending Score (1-100)	-0.1770	0.036	-4.859	0.000	-0.249	-0.105

Omnibus:	4.914	**Durbin-Watson:**	1.903
Prob(Omnibus):	0.086	**Jarque-Bera (JB):**	5.022
Skew:	0.371	**Prob(JB):**	0.0812
Kurtosis:	2.772	**Cond. No.**	263.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
x = encode_cat[["Annual Income (k$)", "Spending Score (1-100)"]]
y = encode_cat[["Gender_Male"]]
x = sm.add_constant(x)
ols_data = sm.OLS(y,x).fit()
ols_data.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Gender_Male	**R-squared:**	0.007
Model:	OLS	**Adj. R-squared:**	-0.003
Method:	Least Squares	**F-statistic:**	0.6568
Date:	Thu, 12 Jan 2023	**Prob (F-statistic):**	0.520
Time:	23:08:20	**Log-Likelihood:**	-143.04
No. Observations:	200	**AIC:**	292.1
Df Residuals:	197	**BIC:**	302.0
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.4314	0.112	3.860	0.000	0.211	0.652
Annual Income (k$)	0.0011	0.001	0.803	0.423	-0.002	0.004
Spending Score (1-100)	-0.0011	0.001	-0.826	0.410	-0.004	0.002

Omnibus:	1159.580	**Durbin-Watson:**	1.990
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	32.509
Skew:	0.242	**Prob(JB):**	8.72e-08
Kurtosis:	1.085	**Cond. No.**	263.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
x = encode_cat[["Age","Gender_Male"]]
y = encode_cat[["Annual Income (k$)"]]
x = sm.add_constant(x)
ols_data = sm.OLS(y,x).fit()
ols_data.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Annual Income (k$)	**R-squared:**	0.003
Model:	OLS	**Adj. R-squared:**	-0.007
Method:	Least Squares	**F-statistic:**	0.3394
Date:	Thu, 12 Jan 2023	**Prob (F-statistic):**	0.713
Time:	23:08:24	**Log-Likelihood:**	-936.59
No. Observations:	200	**AIC:**	1879.
Df Residuals:	197	**BIC:**	1889.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	60.3883	5.679	10.633	0.000	49.188	71.588
Age	-0.0299	0.134	-0.223	0.824	-0.294	0.234
Gender_Male	3.0283	3.761	0.805	0.422	-4.388	10.445

Omnibus:	3.277	**Durbin-Watson:**	0.011
Prob(Omnibus):	0.194	**Jarque-Bera (JB):**	3.300
Skew:	0.308	**Prob(JB):**	0.192
Kurtosis:	2.870	**Cond. No.**	128.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
x = encode_cat[["Age","Gender_Male"]]
y = encode_cat[["Spending Score (1-100)"]]
x = sm.add_constant(x)
ols_data = sm.OLS(y,x).fit()
ols_data.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	Spending Score (1-100)	**R-squared:**	0.109
Model:	OLS	**Adj. R-squared:**	0.099
Method:	Least Squares	**F-statistic:**	11.99
Date:	Thu, 12 Jan 2023	**Prob (F-statistic):**	1.22e-05
Time:	23:08:28	**Log-Likelihood:**	-922.05
No. Observations:	200	**AIC:**	1850.
Df Residuals:	197	**BIC:**	1860.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	74.4089	5.281	14.089	0.000	63.994	84.824
Age	-0.6006	0.125	-4.821	0.000	-0.846	-0.355
Gender_Male	-1.9892	3.497	-0.569	0.570	-8.886	4.908

Omnibus:	10.935	**Durbin-Watson:**	3.447
Prob(Omnibus):	0.004	**Jarque-Bera (JB):**	5.938
Skew:	-0.227	**Prob(JB):**	0.0514
Kurtosis:	2.289	**Cond. No.**	128.

Notes:

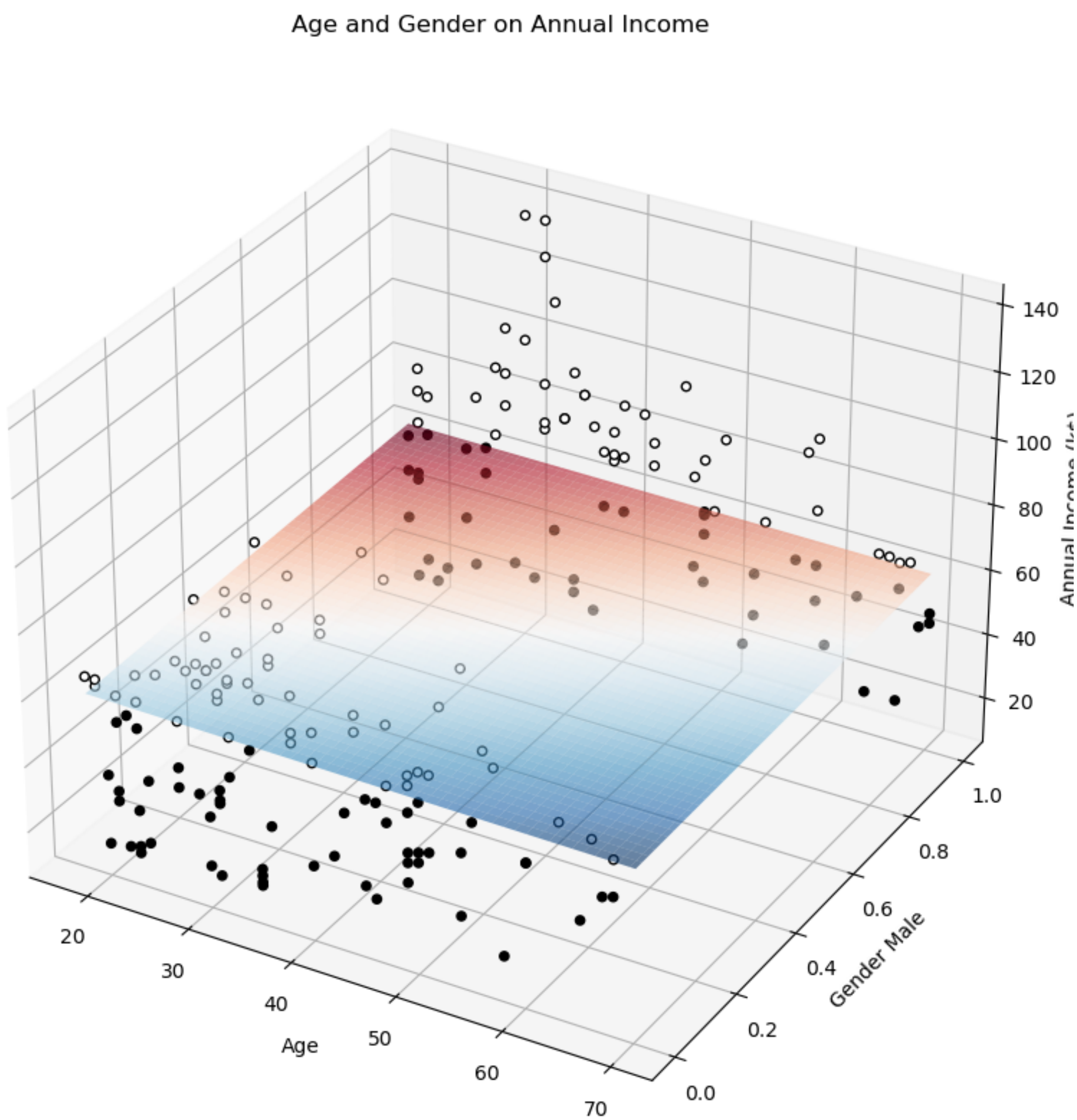
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
x = encode_cat[["Age","Gender_Male"]]
y = encode_cat[["Annual Income (k$)"]]
x = sm.add_constant(x)
ols_data = sm.OLS(y, x).fit()

# create the 3d plot
# Age/Gender_Male grid for 3d plot
xv, yv = np.meshgrid(np.linspace(x.Age.min(), x.Age.max(), num=100), np.linspace(x.

# plot the hyperplane by evaluating the parameters on the grid
z = ols_data.params[0] + ols_data.params[1] * xv + ols_data.params[2] * yv

# create matplotlib 3d axes
fig = plt.pyplot.figure(figsize=(14, 10))
ax = fig.add_subplot(111, projection='3d')
# ax = Axes3D(fig, azim=-115, elev=15)

# plot hyperplane
surface = ax.plot_surface(xv, yv, z, cmap=plt.cm.RdBu_r, alpha=0.6, linewidth=0)

# plot data points - points over the HP are white, points below are black
residual = y["Annual Income (k$)"] - ols_data.predict(x)
ax.scatter(x[residual >= 0].Age, x[residual >= 0].Gender_Male, y[residual >= 0], co
ax.scatter(x[residual < 0].Age, x[residual < 0].Gender_Male, y[residual < 0], color

# set axis labels
ax.set_xlabel("Age")
ax.set_ylabel("Gender Male")
ax.set_zlabel("Annual Income (k$)")
ax.set_title("Age and Gender on Annual Income")

# residual = y - ols_data.predict(x)
# ax.scatter(x[residual].Age, x[residual].Gender_Male, y[residual], color="black",
# ax.scatter(x[residual].Age, x[residual].Gender_Male, y[residual], color="black",
# ax.scatter(x.Age, x.Gender_Male, y, color="black", alpha=1.0, facecolor="white")
# ax.scatter(x.Age, x.Gender_Male, y, color="black", alpha=1.0)
```

Out[ ]:

```
Text(0.5, 0.92, 'Age and Gender on Annual Income')
```



Final Analysis

The third and fourth cluster of our bivariate analysis, as illustrated by scatter plot, accounts for the highest annual income, based on average age and gender, are 20, 23, and 31 and is majority female, over 70 percent. Cluster zero of our bivariate analysis with an averge age of 32 years old accounts for the highest annual income and spending score and is mostly female in gender. Individuals with the average age of 23 and gender female accounts for the highest spending score. Individuals 37 years old and older accounted for the highest annual income, but on average were in median range of the spending score, despite having the available income they spent less. This segment of the market could represent an under-engaged gap in the market and marketing stragety. An omnichannel marketing strategy could be developed based the individual psychology and behavorial profile that would motivate that individual to alter their spending pattern. It is worth noting that a high spending score could be attributed to the individual purchasing highly priced items or spending/purchasing patterns based a host of factors such as the availability of wealth and require further marketing research to determine those factors. The geographic area is fixed to our local mall, and we are only targeting demographics based on age and gender, so further marketing research is necessary to determine the psychology and behaviorial patterns of our target markets. The current analysis does provide data about what specific target markets to focus on or not focus on when allocating resources in marketing and further marketing research.