Compute the customer behavioural segmentation on shopping mall

In [1]:
```
# use to visualize missing value
#!pip install missingno
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
## Display all the columns of the dataframe
pd.pandas.set_option('display.max_columns',None)

from scipy import stats
from scipy.stats import norm, skew
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import RobustScaler
# clustering algorithms
from sklearn.cluster import KMeans,AgglomerativeClustering,DBSCAN
from sklearn.mixture import GaussianMixture

from sklearn.metrics import silhouette_samples, silhouette_score
```

In [3]:
```
#!pip install keras
```

In [4]:
```
#!pip install tensorflow
```

In [5]:
```
from keras.layers import LSTM
```

In [6]:
```
customer_df = pd.read_csv("Mall_Customers.csv")
```

In [7]:
```
customer_df.head()
```

Out[7]:

	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 [8]:
```
customer_df.shape
```

Out[8]: (200, 5)

In [9]:
```
customer_df
```

Out[9]:

	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 [10]:
```
customer_df.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 [11]:
```
customer_df.describe(include='all')
```

Out[11]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200	200.000000	200.000000	200.000000
unique	NaN	2	NaN	NaN	NaN
top	NaN	Female	NaN	NaN	NaN
freq	NaN	112	NaN	NaN	NaN
mean	100.500000	NaN	38.850000	60.560000	50.200000
std	57.879185	NaN	13.969007	26.264721	25.823522
min	1.000000	NaN	18.000000	15.000000	1.000000
25%	50.750000	NaN	28.750000	41.500000	34.750000
50%	100.500000	NaN	36.000000	61.500000	50.000000
75%	150.250000	NaN	49.000000	78.000000	73.000000
max	200.000000	NaN	70.000000	137.000000	99.000000

In [12]:
```
customer_dtype = customer_df.dtypes
customer_dtype.value_counts()
```

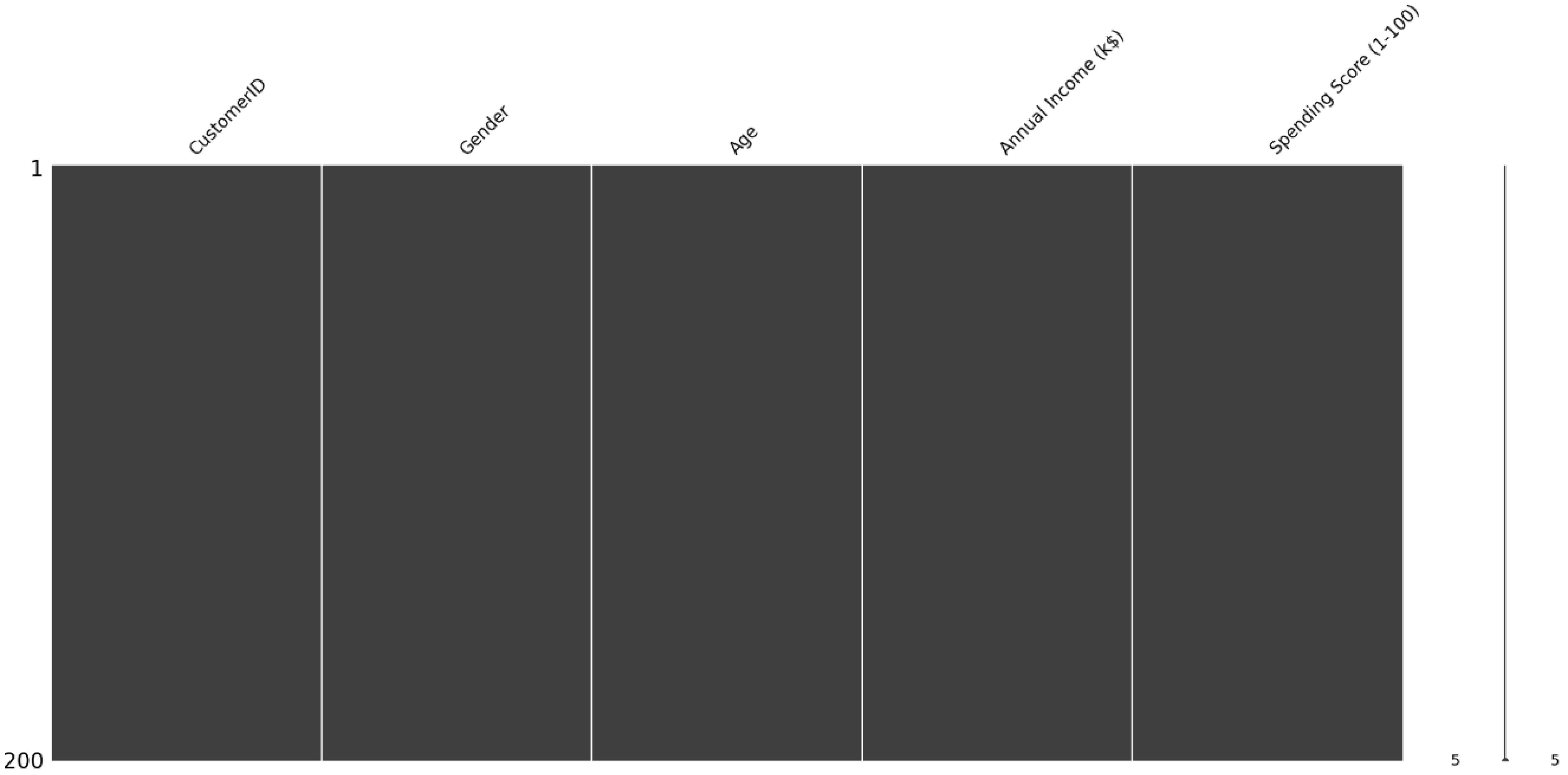
Out[12]:
```
int64     4
object    1
dtype: int64
```

In [13]:
```
customer_df.isnull().sum().sort_values(ascending = False).head()
```

Out[13]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

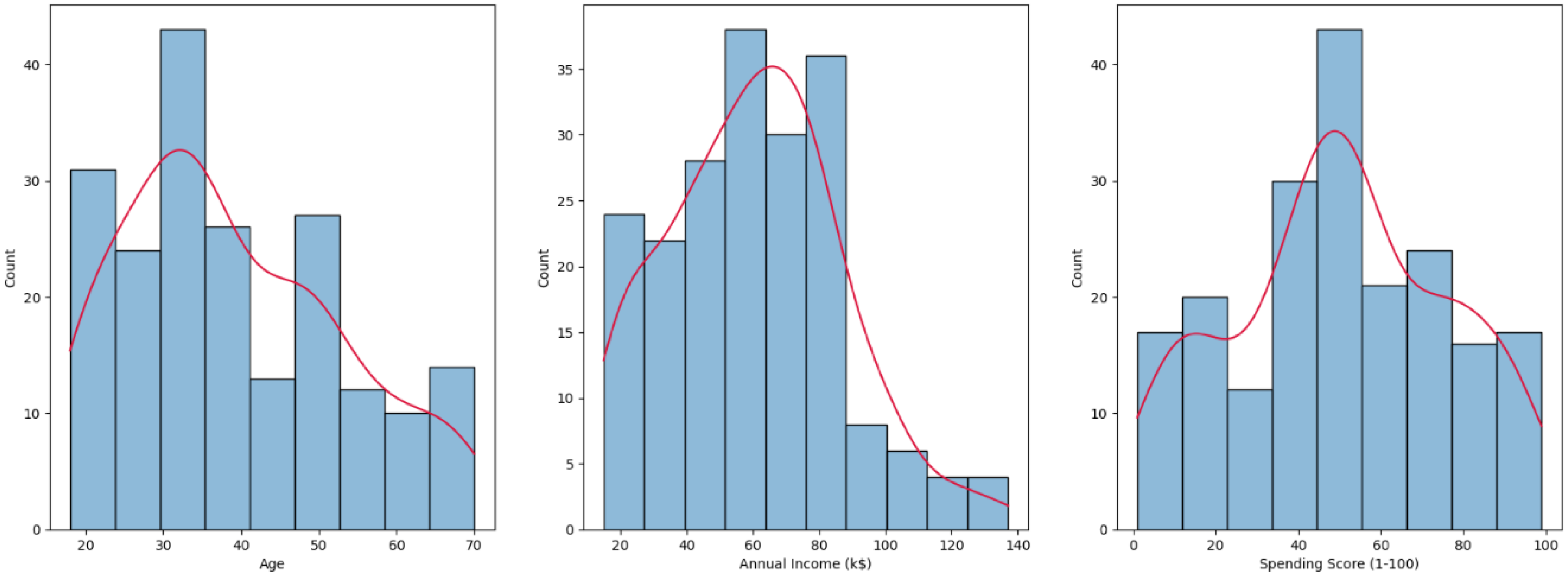
```
In [14]: msno.matrix(customer_df)
```

```
Out[14]: <AxesSubplot:>
```



In [16]:
```
f, axes = plt.subplots(2,2 , figsize=(20, 7), sharex=False)
pos = 1
for i, feature in enumerate(continuous_features):

  plt.subplot(1 , 3 , pos)
  ax = sns.histplot(data=customer_df, x = feature,kde=True,palette="husl") # ax=axes[i%2, i//2]
  ax.lines[0].set_color('crimson')
  pos = pos + 1
```



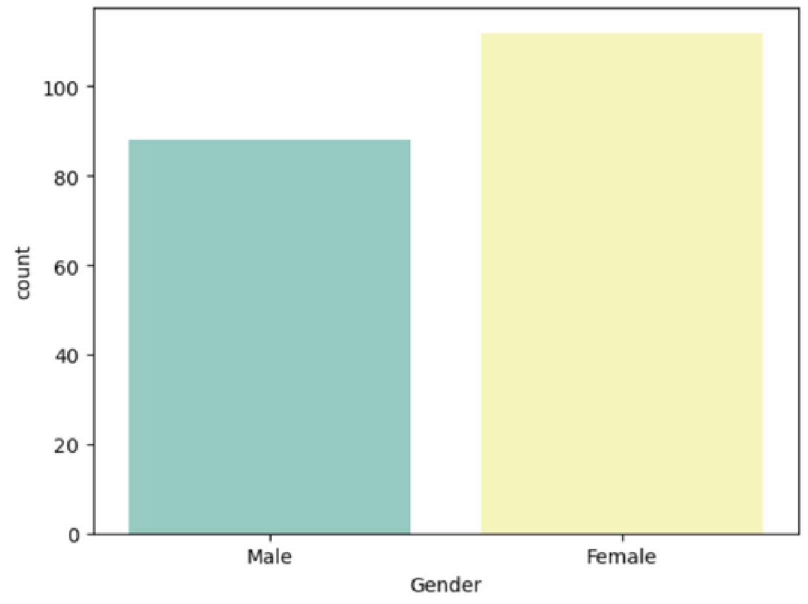
In [17]:
```
# get the features except object types
numeric_feats = customer_df.dtypes[customer_df.dtypes != 'object'].index

# check the skew of all numerical features
skewed_feats = customer_df[numeric_feats].apply(lambda x : skew(x.dropna())).sort_values(ascending = False)
print('\n Skew in numberical features: \n')
skewness_df = pd.DataFrame({'Skew' : skewed_feats})
print(skewness_df.head(10))
```

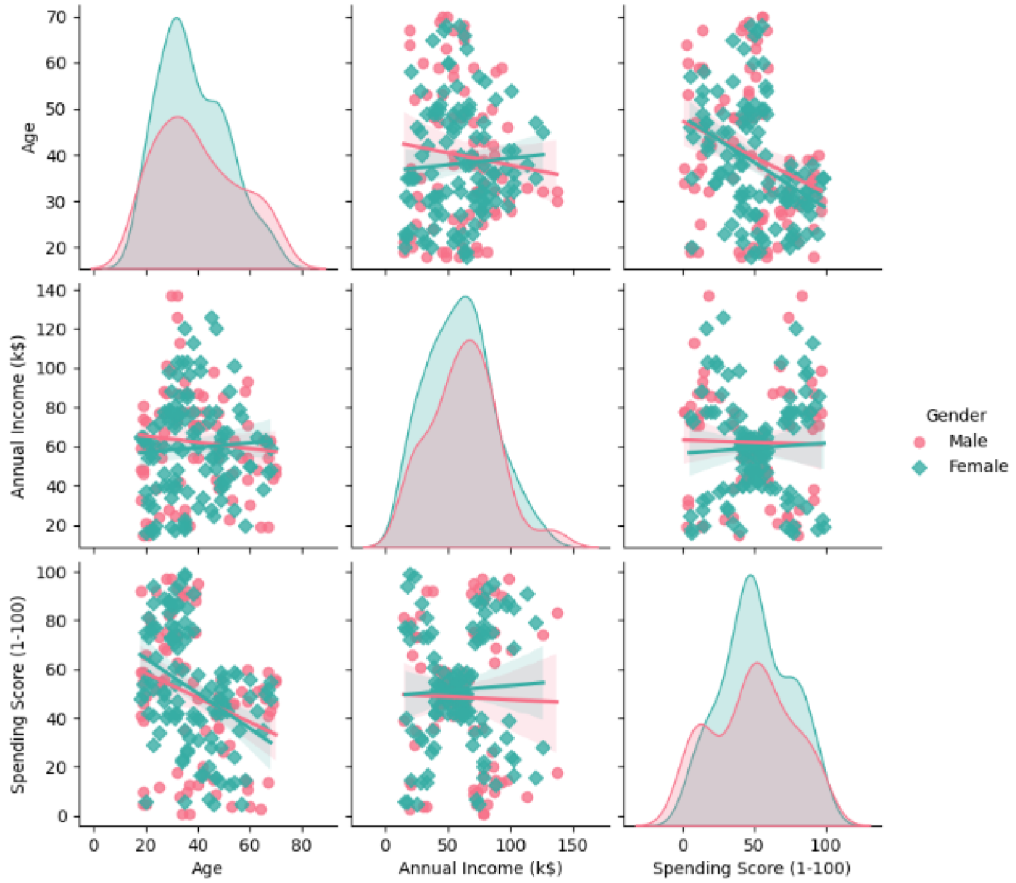
```
 Skew in numberical features:

                            Skew
Age                     0.481919
Annual Income (k$)      0.319424
CustomerID              0.000000
Spending Score (1-100) -0.046865
```

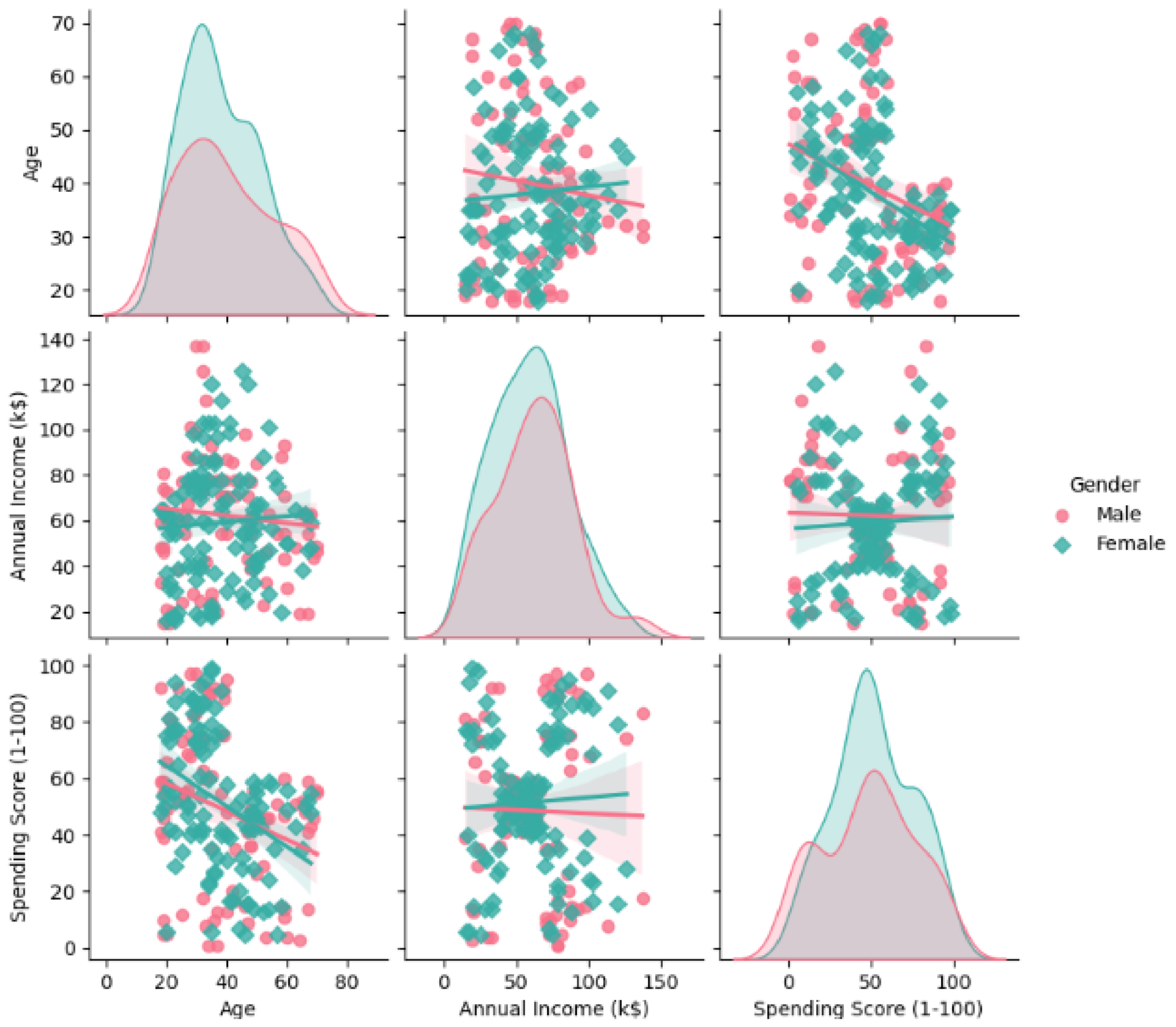
In [18]:
```python
sns.countplot(x='Gender', data=customer_df, palette="Set3")
```
Out[18]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



In [19]:
```python
vars=["Age", "Annual Income (k$)", "Spending Score (1-100)"],  kind ="reg", hue = "Gender", palette="husl", markers = ['o','D'])
```
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1e02f098070>
```

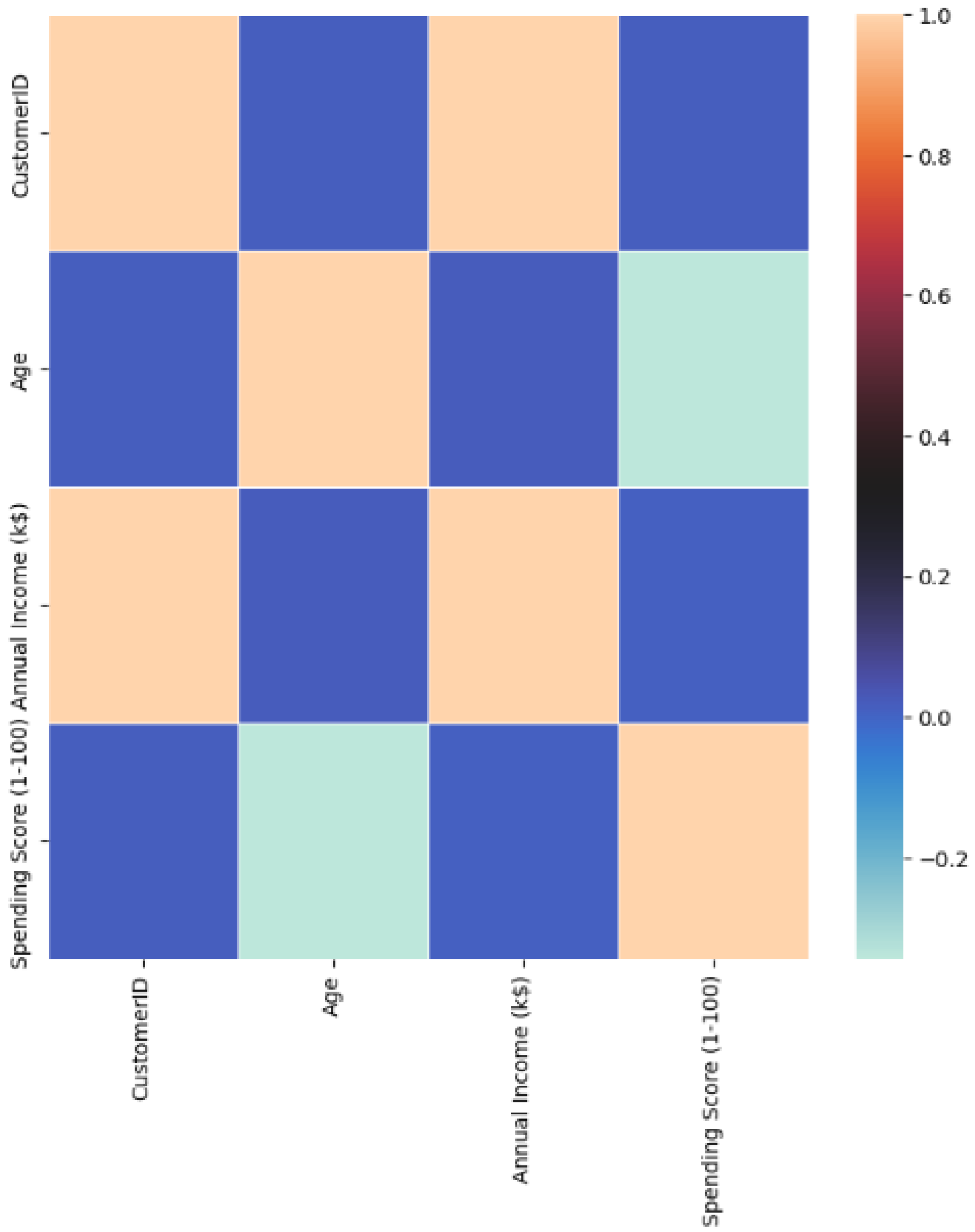


In [19]:
```python
#Pairplot
sns.pairplot(customer_df, vars=["Age", "Annual Income (k$)", "Spending Score (1-100)"],  kind ="reg", hue = "Gender", palette="hu
```
Out[19]:
```
<seaborn.axisgrid.PairGrid at 0x1e02f098070>
```



In [20]:
```
customer_corr = customer_df.corr(method='spearman')
plt.figure(figsize=(8,8))
sns.heatmap(customer_corr, cmap="icefire", linewidths=.5) #'hot', 'hot_r', 'hsv', 'hsv_r', 'icefire', 'icefire_r'
```

Out[20]: <AxesSubplot:>



In [21]:
```
customer_df.drop(columns='CustomerID',axis=1,inplace=True)
```

In [22]:
```
# Generate one-hot dummy columns
customer_df = pd.get_dummies(customer_df).reset_index(drop=True)
```

6. Model Development

In this step we'll apply various clustering algorithms and check which algorithm is best for our dataset. We'r going to use below algorithms.

- Kmeans Clustering
- Agglomerative Clustering
- GaussianMixture Model based clustering
- DBSCAN Clustering

- From the above elbow method we see that **K = 5** is the best K value for our clustering

In [23]:
```
# apply kmeans algorithm
kmeans_model=KMeans(5)
kmeans_clusters = kmeans_model.fit_predict(customer_df)
```

In [24]:
```
# apply agglomerative algorithm
agglo_model = AgglomerativeClustering(linkage="ward",n_clusters=5)
agglomerative_clusters = agglo_model.fit_predict(customer_df)
```

In [25]:
```
GaussianMixture_model = GaussianMixture(n_components=5)
gmm_clusters = GaussianMixture_model.fit_predict(customer_df)
```

In [26]:
```
model_dbscan = DBSCAN(eps=3, min_samples=17)
dbscan_clusters = model_dbscan.fit_predict(customer_df)
```

In [27]:
```
def silhouette_method(df,algo,y_pred):
  print('=================================================================================')
  print('Clustering ',algo," : silhouette score : ",silhouette_score(df,y_pred) )


silhouette_method(customer_df,' : KMeans',kmeans_clusters)
silhouette_method(customer_df,' : Agglomerative',agglomerative_clusters)
silhouette_method(customer_df,' : GaussianMixture',gmm_clusters)
print('=================================================================================')
```

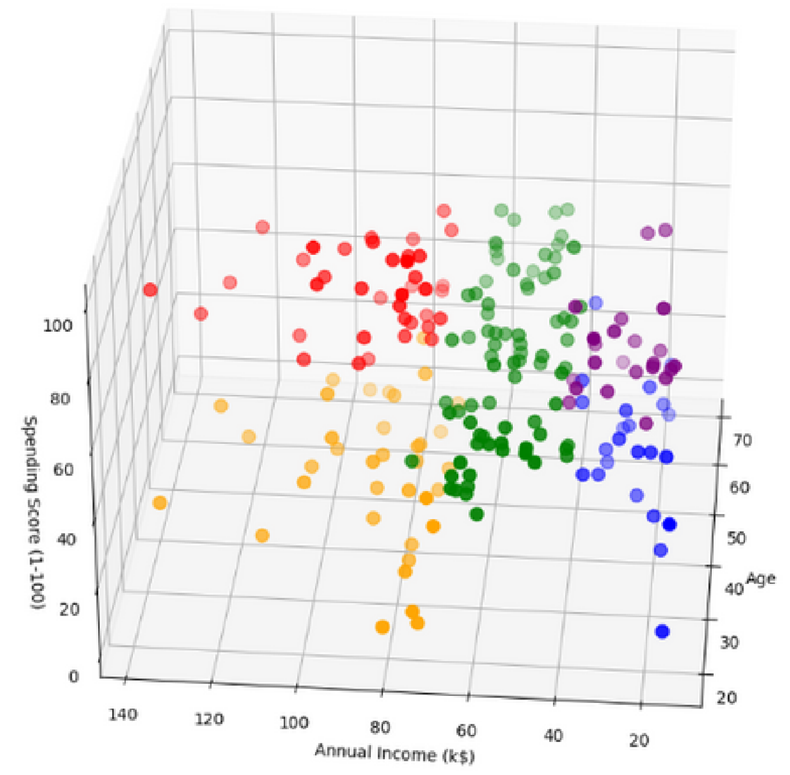
```
=================================================================================
Clustering   : KMeans  : silhouette score :  0.443849645338732
=================================================================================
Clustering   : Agglomerative  : silhouette score :  0.43976347350045475
=================================================================================
Clustering   : GaussianMixture  : silhouette score :  0.41597562753392225
---------------------------------------------------------------------------------
```

In [28]:
```
customer_df["label"] = kmeans_clusters

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(customer_df.Age[customer_df.label == 0], customer_df["Annual Income (k$)"][customer_df.label == 0], customer_df["Spen
ax.scatter(customer_df.Age[customer_df.label == 1], customer_df["Annual Income (k$)"][customer_df.label == 1], customer_df["Spen
ax.scatter(customer_df.Age[customer_df.label == 2], customer_df["Annual Income (k$)"][customer_df.label == 2], customer_df["Spen
ax.scatter(customer_df.Age[customer_df.label == 3], customer_df["Annual Income (k$)"][customer_df.label == 3], customer_df["Spen
ax.scatter(customer_df.Age[customer_df.label == 4], customer_df["Annual Income (k$)"][customer_df.label == 4], customer_df["Spen
ax.view_init(30, 185)
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```

In [28]:
```
f["Annual Income (k$)"][customer_df.label == 0], customer_df["Spending Score (1-100)"][customer_df.label == 0], c='blue', s=60)
f["Annual Income (k$)"][customer_df.label == 1], customer_df["Spending Score (1-100)"][customer_df.label == 1], c='red', s=60)
f["Annual Income (k$)"][customer_df.label == 2], customer_df["Spending Score (1-100)"][customer_df.label == 2], c='green', s=60)
f["Annual Income (k$)"][customer_df.label == 3], customer_df["Spending Score (1-100)"][customer_df.label == 3], c='orange', s=60)
f["Annual Income (k$)"][customer_df.label == 4], customer_df["Spending Score (1-100)"][customer_df.label == 4], c='purple', s=60)
```



CONCLUSION:

we minimize the amount of time that phishing pages can remain active before we protect our users from them. Even with a perfect classifier and a robust system, we recognize that our blacklist approach keeps us perpetually a step behind the phishers. We can only identify a phishing URL and normal URL using machine learning algorithm. Result we got in terms of accuracy metric.