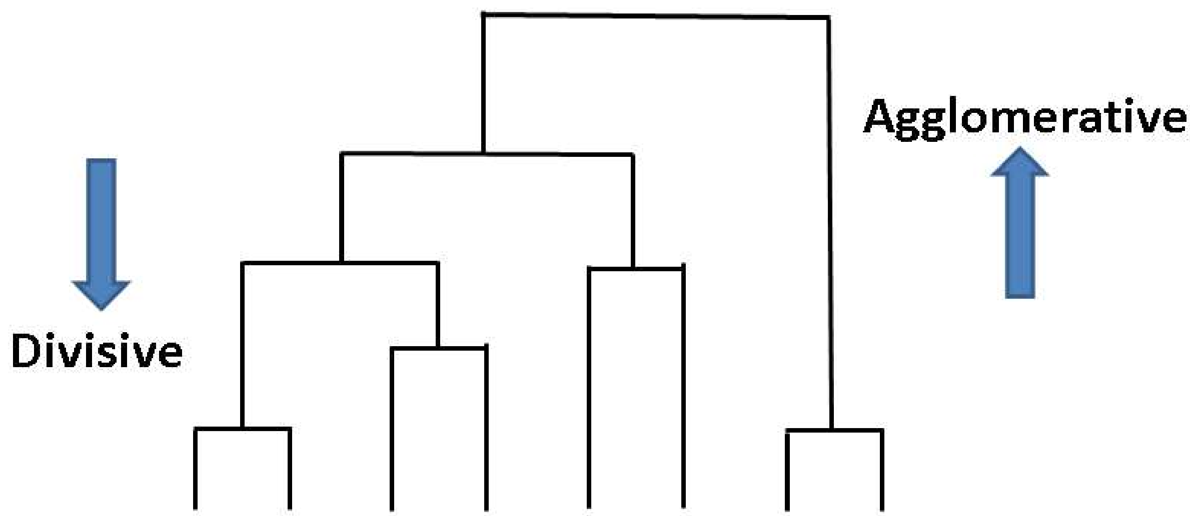
Hierarchical Clustering:

- Hierarchical clustering algorithms group similar objects into groups called clusters.

There are two types of hierarchical clustering algorithms:

1. **Agglomerative:** Bottom up approach. Start with many small clusters and merge them together to create bigger clusters.
2. **Divisive:** Top down approach. Start with a single cluster then break it up into smaller clusters.



Pros and Cons of Hierarchical Clustering

Pros:

- No assumption of a particular number of cluster, like in K-means.
- Many correspond to meaningful taxonomies.

Cons:

- Once a decision is made to combine two cluster, it can't be undone.
- Very slow for large data sets, **O(n^2 log(n))**.

Working:

1. Make each data point a cluster.
2. Take the two closest cluster and make them one cluster.
3. Repeat step 2 until there is only one cluster.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df= pd.read_csv('Mall_Customers.csv')
df.head()
```

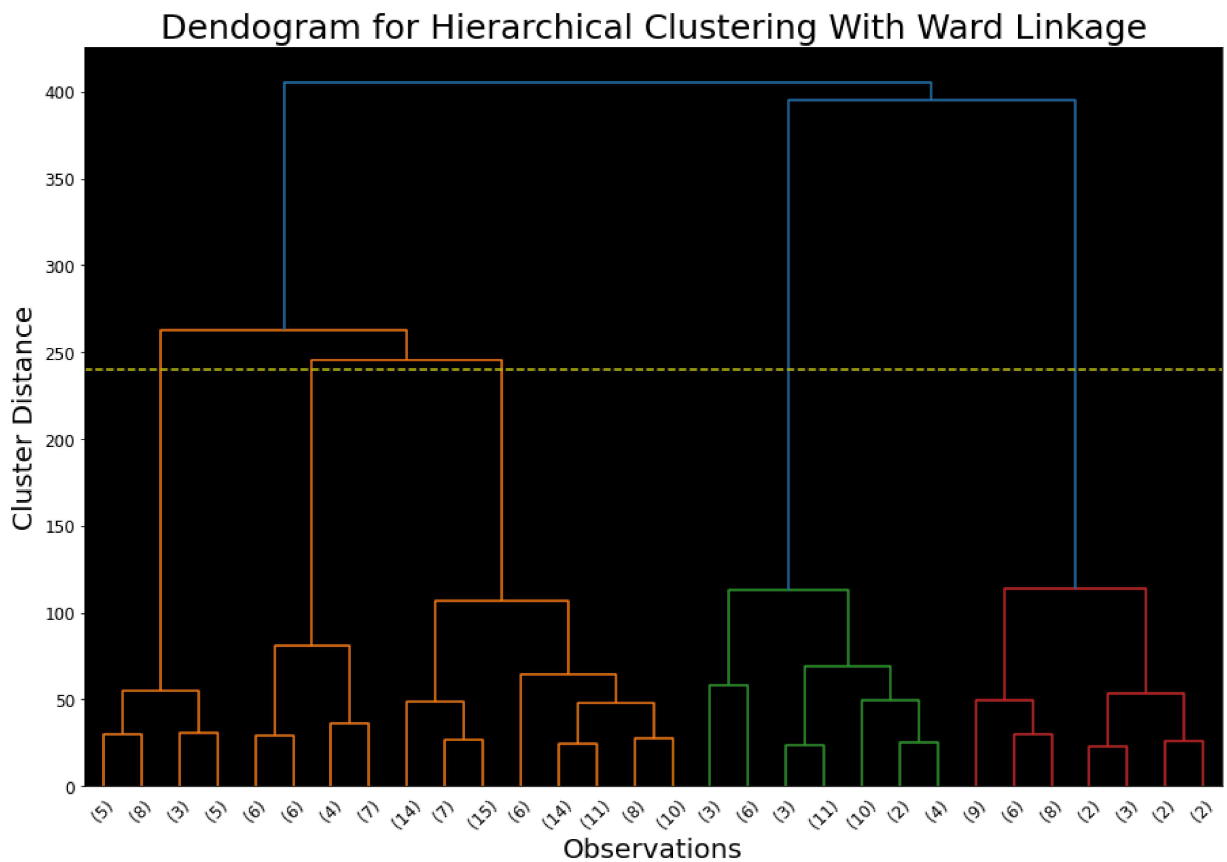
Out[2]:

	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= df.iloc[:, 3:5]
```

In [4]:
```
import scipy.cluster.hierarchy as sch
```

In [5]:
```
plt.figure(figsize= (15,10))
ax= plt.axes()
ax.set_facecolor('black')
ax= dendo= sch.dendrogram(sch.linkage(data, method= 'ward'), truncate_mode= 'last
plt.axhline(y=240, color= 'y', linestyle= '--')
plt.title('Dendogram for Hierarchical Clustering With Ward Linkage', fontsize=25)
plt.xlabel('Observations', fontsize=20)
plt.ylabel('Cluster Distance', fontsize=20)
plt.xticks(fontsize= 12);
plt.yticks(fontsize=12);
plt.show()
```



In [6]:
```
from sklearn.cluster import AgglomerativeClustering
```

In [7]:
```
agl= AgglomerativeClustering(n_clusters=5, affinity= 'euclidean', linkage= 'ward'
lables_= agl.fit_predict(data)
```

In [8]:
```
lable= list(lables_)
```

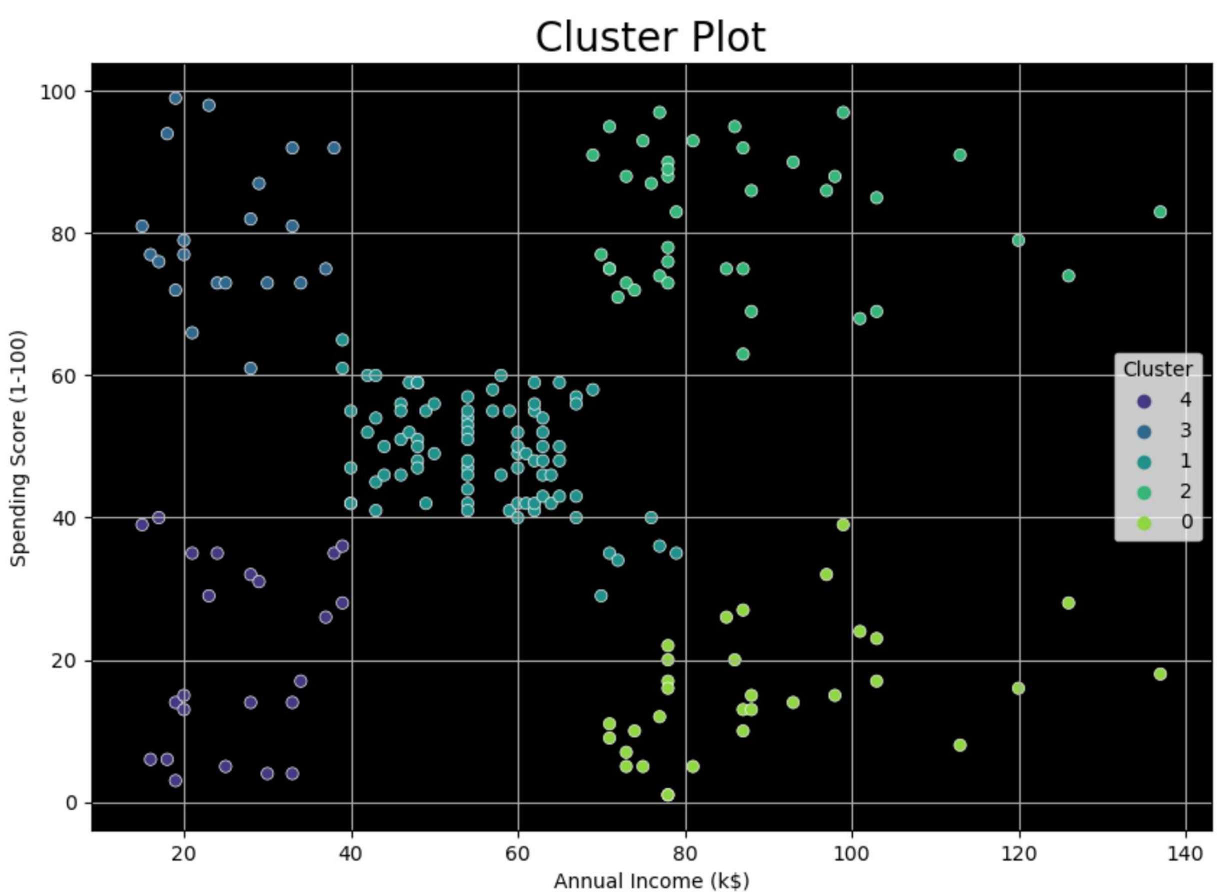
In [9]:
```
data['Cluster']= lable
data.head()
```

Out[9]:

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

In [23]:
```
plt.figure(figsize=(10,7), dpi=100)
ax= plt.axes()
ax.set_facecolor('black')
sns.scatterplot(data['Annual Income (k$)'], data['Spending Score (1-100)'], hue=
plt.grid()
plt.title('Cluster Plot', fontsize= 20)
```

Out[23]: Text(0.5, 1.0, 'Cluster Plot')



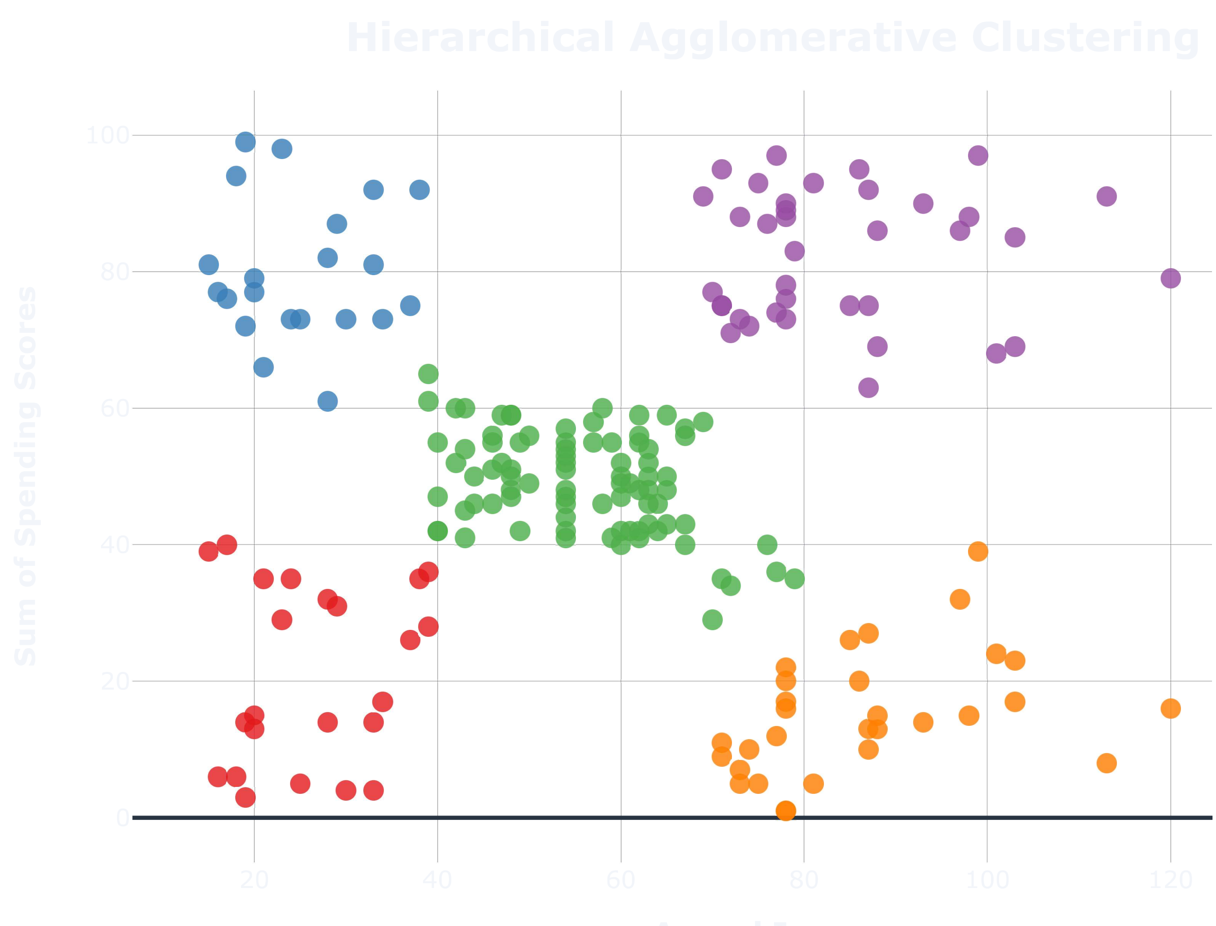
Scatter_Plot with Plotly

In [20]:
```
import plotly.express as px
data['Cluster']= data['Cluster'].astype(str)
fig= px.scatter(data, x= 'Annual Income (k$)',
               y= 'Spending Score (1-100)',
               color= 'Cluster',
               color_discrete_sequence = px.colors.qualitative.Set1)

fig.update_traces(marker= dict(size= 10, opacity= 0.80))

fig.update_layout(
        template= 'plotly_dark',
        width= 800,
        legend_title= 'Clusters',
        title= dict(
           text= '<b>Hierarchical Agglomerative Clustering</b>',
           x= 0.5,
           y= 0.95,
           font= dict(
             size= 20
           )
        ),
    xaxis_title= '<b> Annual Income</b>',
    yaxis_title= '<b>Sum of Spending Scores</b>'
        )


fig.show()
```



Cluster 0 categorizes customers having an average Annual Income(40k-60k) and an average Spending Score(40-60). **Cluster 1** represents customers that have a high Annual Income(>70k) and a low Spending Score(<40). **Cluster 2** depicts customers with a high Annual Income and a high Spending Score(>60). Customers with a low Annual Income(<40k) and a high Spending Score(>60) belong to **Cluster 3**. Lastly, **Cluster 4** represents customers with a low Annual Income(<40k) and a low Spending Score(<40).