```
from google.colab import files
uploaded=files.upload()
```

Choose Files Mall_Customers (1).csv

- **Mall_Customers (1).csv**(text/csv) - 3981 bytes, last modified: 7/25/2025 - 100% done

```
Saving Mall_Customers (1).csv to Mall_Customers (1).csv
```

Importing Libraries

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

Loading the dataset

```
df=pd.read_csv('Mall_Customers.csv')
df
```

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

EDA

```
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
```

```
df.isnull().sum().sum()
```

```
np.int64(0)
```

```
df.duplicated().sum()
```

What can I help you build?

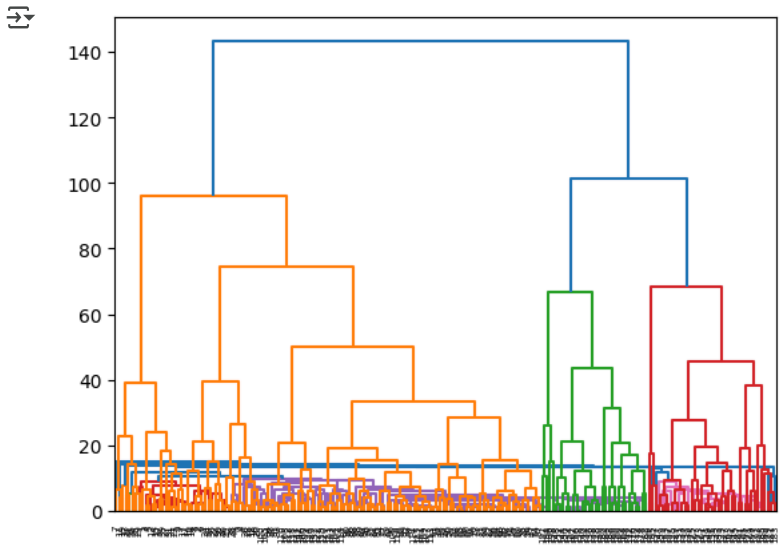
```
np.int64(0)
```

Taking input data

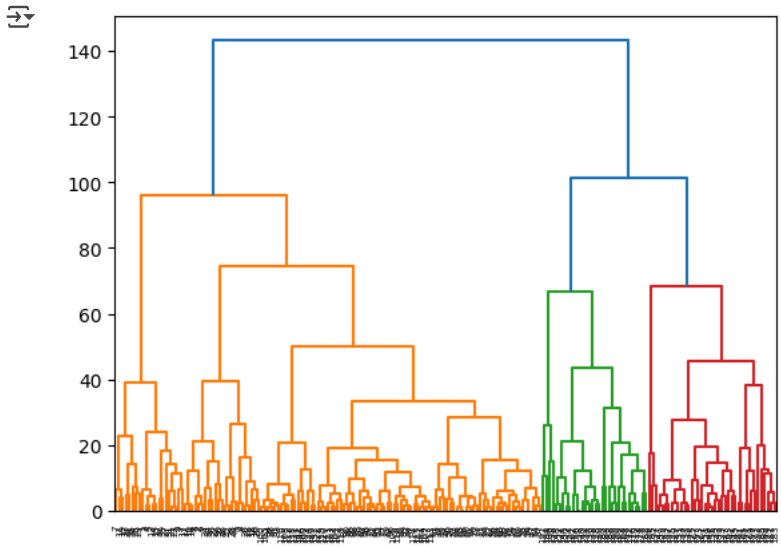
```
X=df[['Annual Income (k$)','Spending Score (1-100)']].values #Converting to 2D array
```

Choosing number of clusters

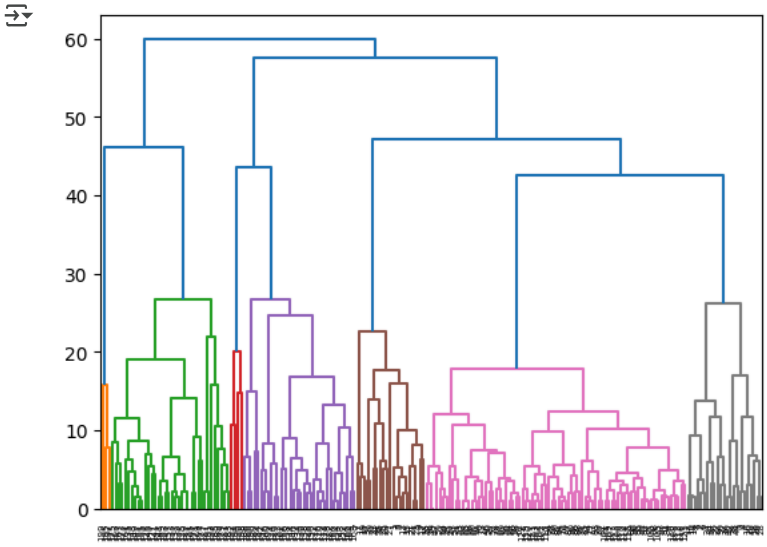
```
import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(X,method='single'))
```



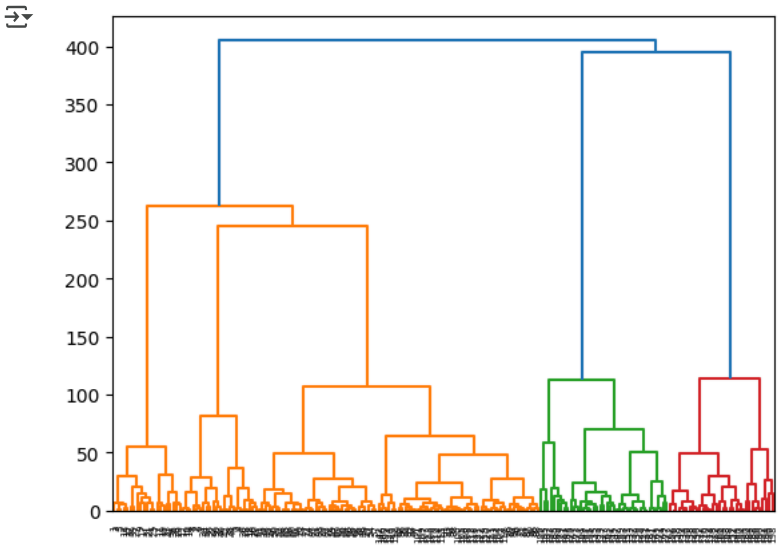

```
dendrogram = sch.dendrogram(sch.linkage(X,method='complete'))
```



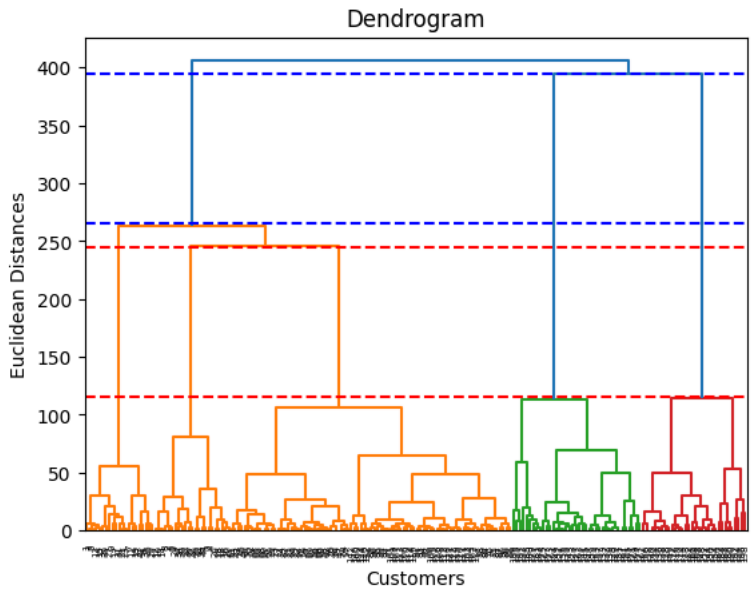

```
dendrogram = sch.dendrogram(sch.linkage(X,method='average'))
```




```
dendrogram = sch.dendrogram(sch.linkage(X,method='ward'))
```




```
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean Distances')
plt.axhline(y=394, color='b', linestyle='--')
plt.axhline(y=265, color='b', linestyle='--')
plt.axhline(y=244, color='r', linestyle='--')
plt.axhline(y=115, color='r', linestyle='--')
plt.show() # find largest vertical distance we can make without crossing any other horizontal line
```



Training the model

```
from sklearn.cluster import AgglomerativeClustering

Agg_clu=AgglomerativeClustering(n_clusters=5,metric='euclidean',linkage='ward')
Y=Agg_clu.fit_predict(X)
```

```
Y
```

```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2])
```

```
Output=pd.DataFrame(Y,columns=['Clusterid'])
Output
```

	Clusterid
0	4
1	3
2	4
3	3
4	4
...	...
195	2
196	0
197	2
198	0
199	2

200 rows × 1 columns

Next steps: Generate code with Output | View recommended plots | New interactive sheet

```
pd.concat([df,Output],axis=1)
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Clusterid
0	1	Male	19	15	39	4
1	2	Male	21	15	81	3
2	3	Female	20	16	6	4
3	4	Female	23	16	77	3
4	5	Female	31	17	40	4
...	...	...	...	...	...	...
195	196	Female	35	120	79	2
196	197	Female	45	126	28	0
197	198	Male	32	126	74	2
198	199	Male	32	137	18	0
199	200	Male	30	137	83	2

200 rows × 6 columns

Vizualization

```
plt.scatter(X[:, 0], X[:, 1], c= Y, cmap = 'rainbow')

plt.title('Customer Groups')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.show()
```