Task - 1 : Prediction using supervised ML

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

path =  "/content/student_scores - student_scores.csv"
Data = pd.read_csv(path)
print("Data")
Data
```

```
Data
```

```
{"summary":"{\n  \"name\": \"Data\",\n  \"rows\": 25,\n  \"fields\":
[\n    {\n      \"column\": \"Hours\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2.5250940576540906,\n
\"min\": 1.1,\n        \"max\": 9.2,\n        \"num_unique_values\":
23,\n        \"samples\": [\n          8.9,\n          2.7,\n
2.5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Scores\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 25,\n        \"min\": 17,\n        \"max\": 95,\n
\"num_unique_values\": 23,\n        \"samples\": [\n          95,\n
25,\n          21\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"Data"}
```

```
Data.head()
```

```
{"summary":"{\n  \"name\": \"Data\",\n  \"rows\": 25,\n  \"fields\":
[\n    {\n      \"column\": \"Hours\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2.5250940576540906,\n
\"min\": 1.1,\n        \"max\": 9.2,\n        \"num_unique_values\":
23,\n        \"samples\": [\n          8.9,\n          2.7,\n
2.5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Scores\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 25,\n        \"min\": 17,\n        \"max\": 95,\n
\"num_unique_values\": 23,\n        \"samples\": [\n          95,\n
25,\n          21\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"Data"}
```

```
Data.tail()
```

```
{"summary":"{\n  \"name\": \"Data\",\n  \"rows\": 5,\n  \"fields\": [\
n    {\n      \"column\": \"Hours\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2.1224985276791126,\n
```

\"min\": 2.7,\n   \"max\": 7.8,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   4.8,\n   7.8,\n   3.8\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Scores\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 24,\n   \"min\": 30,\n   \"max\": 86,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   54,\n   86,\n   35\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

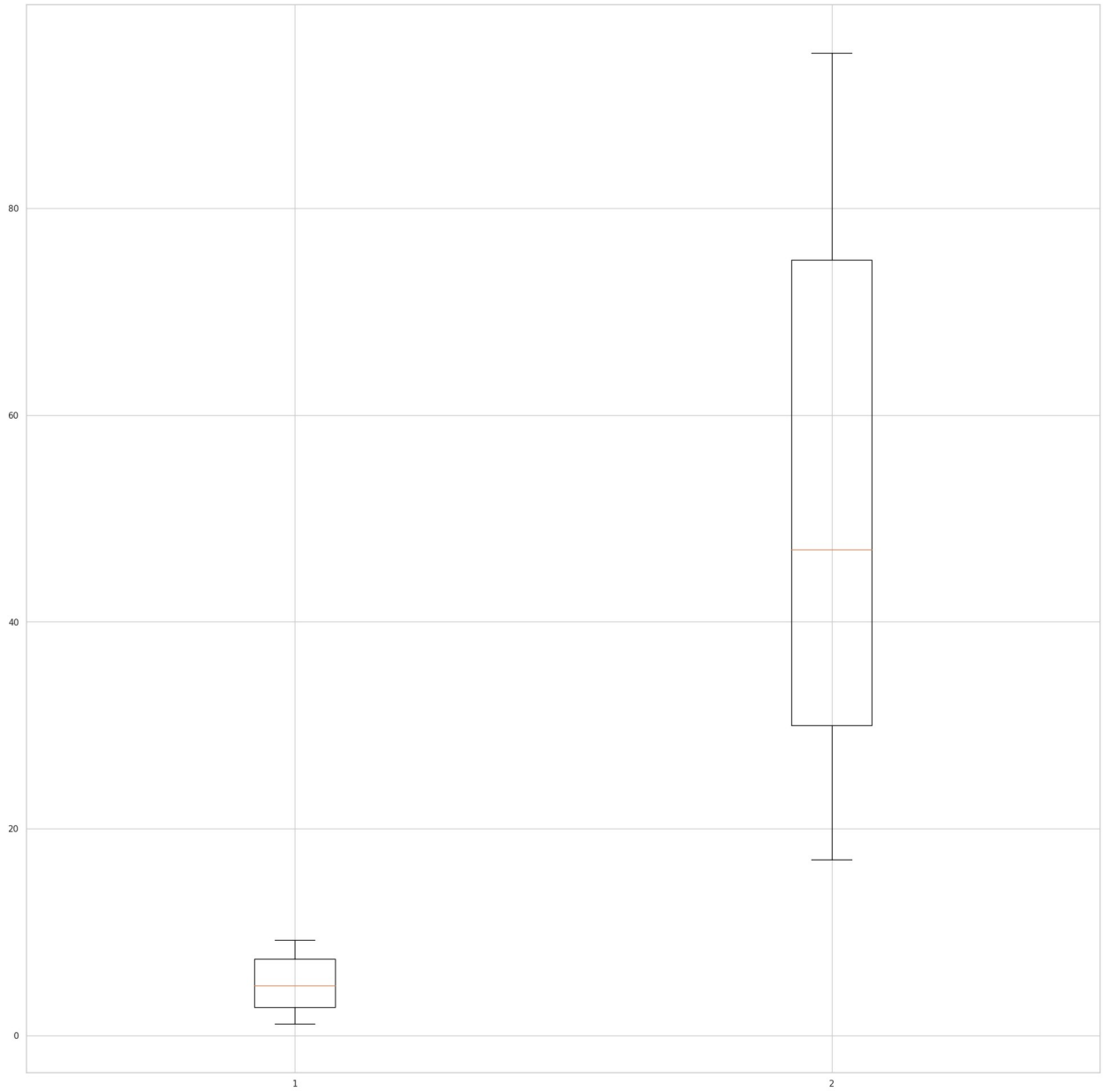
```
Data.describe()
```

{"summary":"{\n \"name\": \"Data\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"Hours\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 7.660098285663746,\n   \"min\": 1.1,\n   \"max\": 25.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   5.012,\n   4.8,\n   25.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Scores\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 27.358571628410314,\n   \"min\": 17.0,\n   \"max\": 95.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   51.48,\n   47.0,\n   25.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

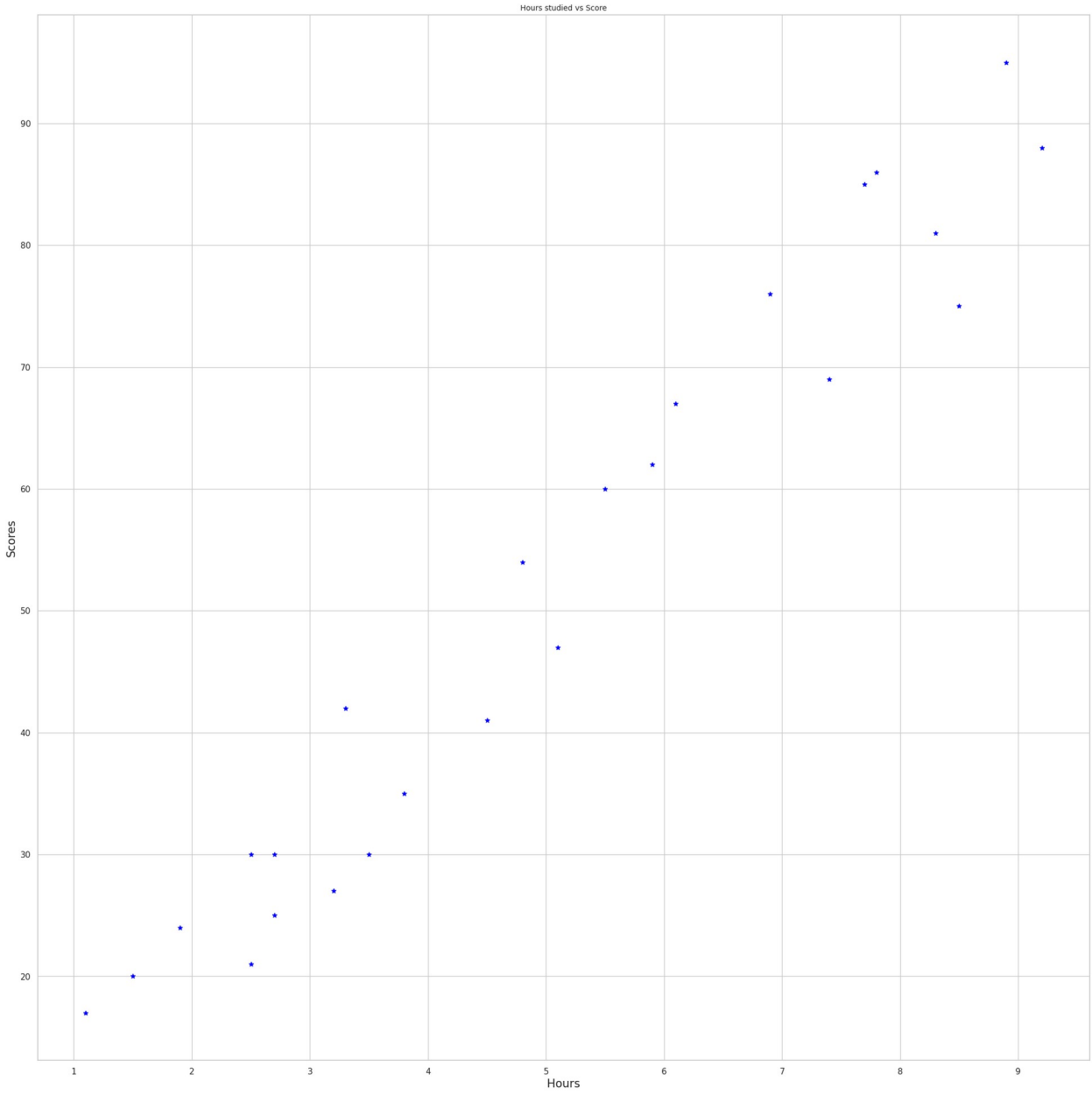
```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

```
import seaborn as sns
plt.boxplot(Data)
plt.show()
```




```
plt.xlabel('Hours',fontsize=15)
plt.ylabel('Scores',fontsize=15)
plt.title('Hours studied vs Score', fontsize=10)
plt.scatter(Data.Hours,Data.Scores,color='blue',marker='*')
plt.show()
```




```
X = Data.iloc[:,:-1].values
Y = Data.iloc[:,1].values
X
```

```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
```

```
       [8.3],
       [2.7],
       [7.7],
       [5.9],
       [4.5],
       [3.3],
       [1.1],
       [8.9],
       [2.5],
       [1.9],
       [6.1],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

```
Y
```

```
array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 17, 95,
30,
       24, 67, 69, 30, 54, 35, 76, 86])
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,random_state =
0,test_size=0.2)
```

```
print("X train.shape =", X_train.shape)
print("Y train.shape =", Y_train.shape)
print("X test.shape  =", X_test.shape)
print("Y test.shape  =", Y_test.shape)
```

```
X train.shape = (20, 1)
Y train.shape = (20,)
X test.shape  = (5, 1)
Y test.shape  = (5,)
```

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

```
linreg.fit(X_train,Y_train)
print("Training our algorithm is finished")
```

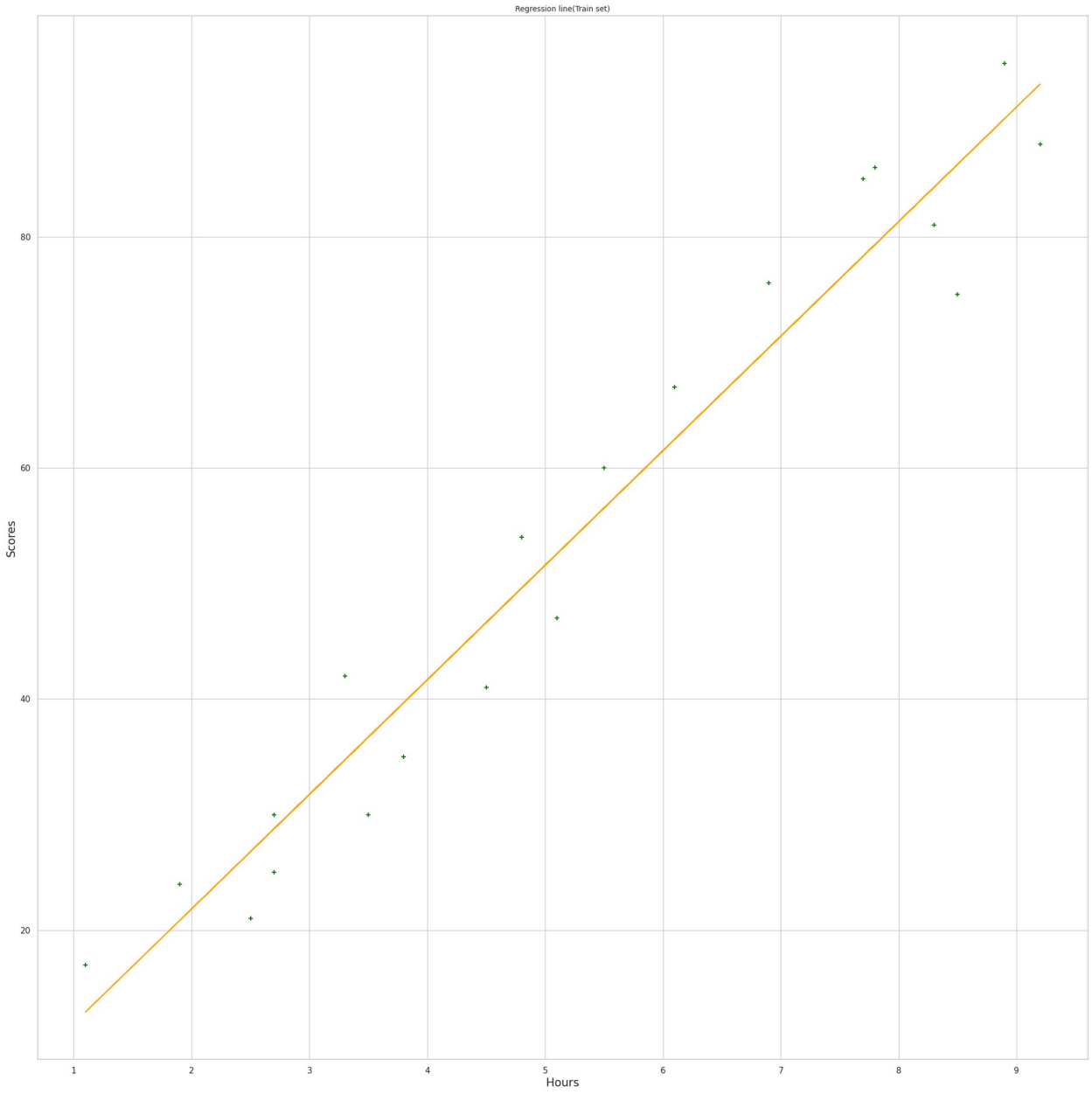
```
Training our algorithm is finished
```

```
print("B0 =",linreg.intercept_,"\nB1 =",linreg.coef_)
```

```
B0 = 2.018160041434662
B1 = [9.91065648]
```

```
Y0 = linreg.intercept_ + linreg.coef_*X_train
```

```
plt.scatter(X_train,Y_train,color='green',marker='+')
plt.plot(X_train,Y0,color='orange')
plt.xlabel("Hours",fontsize=15)
plt.ylabel("Scores",fontsize=15)
plt.title("Regression line(Train set)",fontsize=10)
plt.show()
```

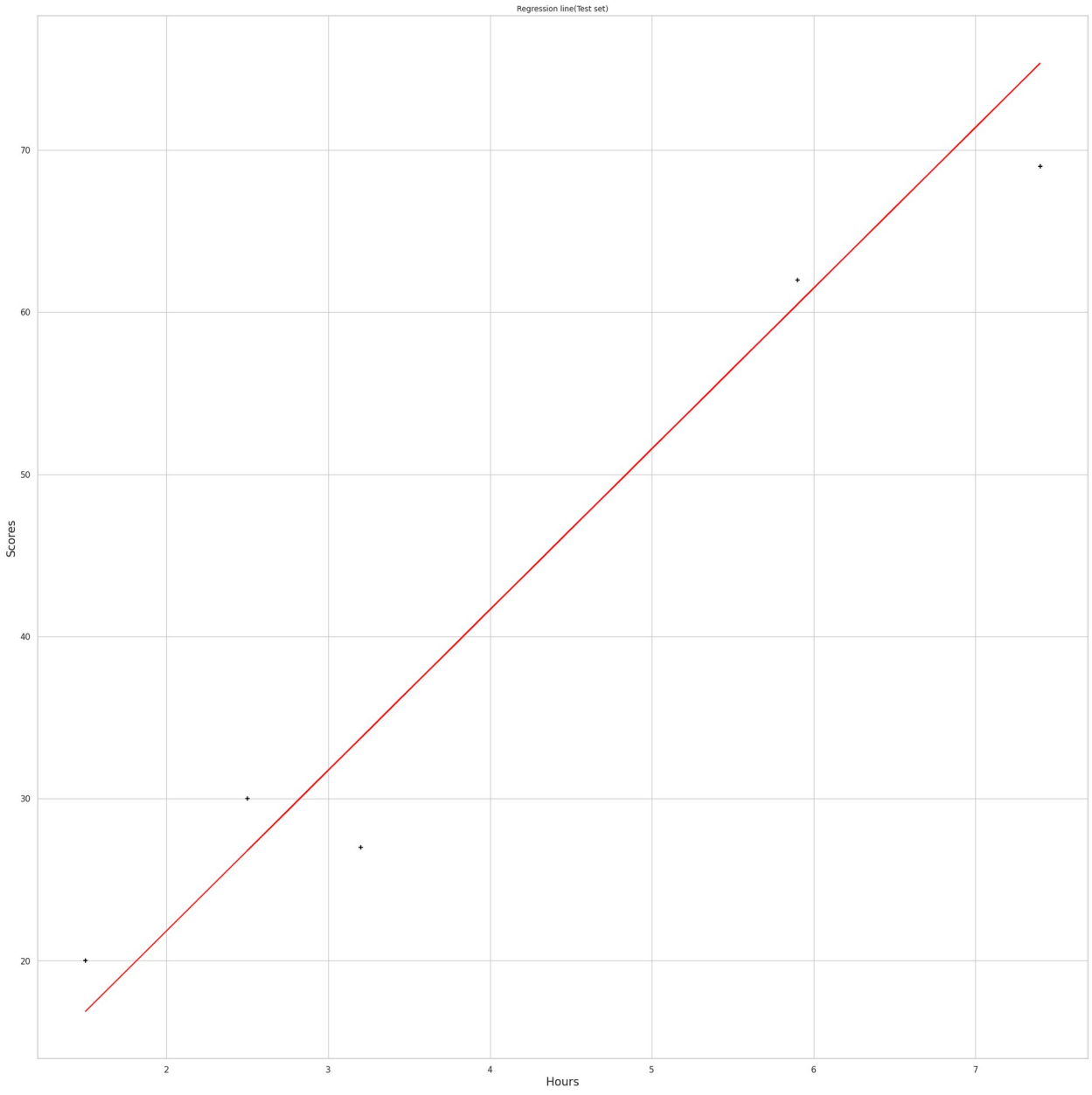



```
Y_pred=linreg.predict(X_test)##predicting the Scores for test data
print(Y_pred)

[16.88414476 33.73226078 75.357018   26.79480124 60.49103328]
```

```
Y_test
```

```
array([20, 27, 69, 30, 62])
```

```
plt.plot(X_test,Y_pred,color='red')
plt.scatter(X_test,Y_test,color='black',marker='+')
plt.xlabel("Hours",fontsize=15)
plt.ylabel("Scores",fontsize=15)
plt.title("Regression line(Test set)",fontsize=10)
plt.show()
```



```
Y_test1 = list(Y_test)
prediction=list(Y_pred)
df_compare = pd.DataFrame({ 'Actual':Y_test1,'Result':prediction})
df_compare
```

```
{"summary":"{\n  \"name\": \"df_compare\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"Actual\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
22,\n        \"min\": 20,\n        \"max\": 69,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          27,\n
62,\n          69\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Result\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 24.407192455008023,\n        \"min\": 16.884144762398023,\n
\"max\": 75.35701799818725,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          33.732260779489835,\n
60.491033277223885,\n          75.35701799818725\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df_compare"}
```

```
from sklearn import metrics
metrics.r2_score(Y_test,Y_pred)
```

```
0.9454906892105354
```

```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

```
MSE = metrics.mean_squared_error(Y_test,Y_pred)
root_E = np.sqrt(metrics.mean_squared_error(Y_test,Y_pred))
Abs_E = np.sqrt(metrics.mean_squared_error(Y_test,Y_pred))
print("Mean Squared Error      = ",MSE)
print("Root Mean Squared Error = ",root_E)
print("Mean Absolute Error     = ",Abs_E)
```

```
Mean Squared Error      =  21.598769307217456
Root Mean Squared Error =  4.647447612100373
Mean Absolute Error     =  4.647447612100373
```

```
Prediction_score = linreg.predict([[9.25]])
print("predicted score for a student studying 9.25
hours :",Prediction_score)
```

```
predicted score for a student studying 9.25 hours : [93.69173249]
```

Task - 2 : Prediction using Unsupervised ML

```
from sklearn.cluster import KMeans
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
%matplotlib inline
```

```
df=pd.read_csv('/content/Iris (1).csv')
df.drop(['Id'],axis=1,inplace=True)

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 150,\n  \"fields\": [\
n    {\n      \"column\": \"SepalLengthCm\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0.828066127977863,\n
\"min\": 4.3,\n        \"max\": 7.9,\n        \"num_unique_values\":
35,\n        \"samples\": [\n          6.2,\n          4.5,\n
5.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SepalWidthCm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.4335943113621737,\n        \"min\":
2.0,\n        \"max\": 4.4,\n        \"num_unique_values\": 23,\n
\"samples\": [\n          2.3,\n          4.0,\n          3.5\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"PetalLengthCm\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.7644204199522626,\n        \"min\": 1.0,\n        \"max\": 6.9,\n
\"num_unique_values\": 43,\n        \"samples\": [\n          6.7,\n
3.8,\n          3.7\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PetalWidthCm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.7631607417008411,\n        \"min\":
0.1,\n        \"max\": 2.5,\n        \"num_unique_values\": 22,\n
\"samples\": [\n          0.2,\n          1.2,\n          1.3\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Species\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n          \"Iris-
setosa\",\n          \"Iris-versicolor\",\n          \"Iris-
virginica\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
df.shape
```

```
(150, 5)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
```

```
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
df.describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n
{\n      \"column\": \"SepalLengthCm\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 51.24711349471842,\n
\"min\": 0.828066127977863,\n        \"max\": 150.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
5.843333333333334,\n          5.8,\n          150.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SepalWidthCm\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
52.08647211421483,\n        \"min\": 0.4335943113621737,\n
\"max\": 150.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          3.0540000000000003,\n          3.0,\n
150.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PetalLengthCm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 51.835227940958106,\n        \"min\":
1.0,\n        \"max\": 150.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          3.758666666666666,\n          4.35,\n
150.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PetalWidthCm\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 52.636634243409915,\n        \"min\":
0.1,\n        \"max\": 150.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          1.1986666666666668,\n          1.3,\n
150.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


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

```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
df.drop_duplicates(inplace=True)
```

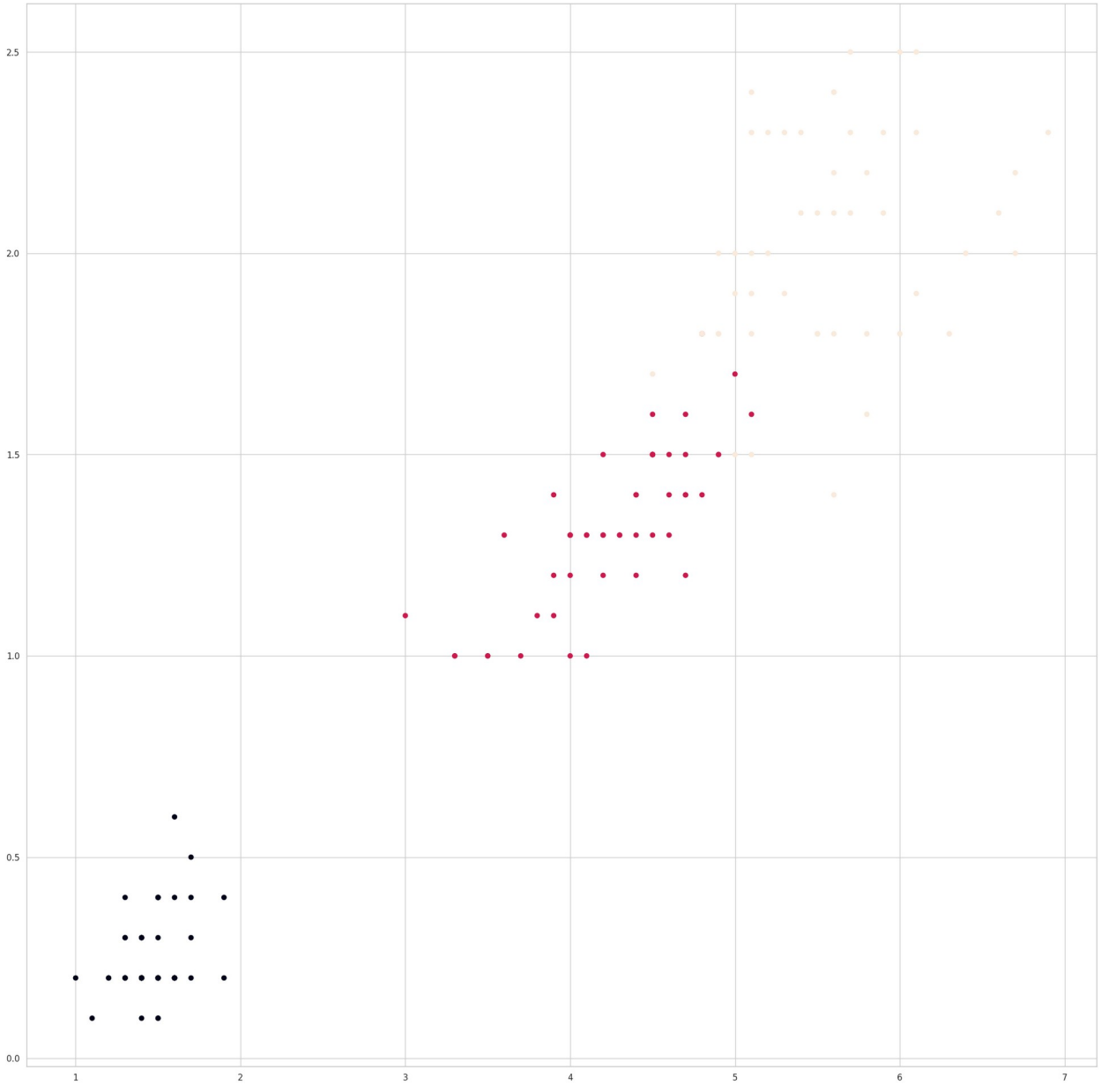
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Species']=le.fit_transform(df['Species'])
df['Species'].value_counts()
```

```
Species
1    50
```

```
2    49
0    48
Name: count, dtype: int64
```

```
plt.scatter(df['PetalLengthCm'],df['PetalWidthCm'],c=df.Species.values
)
```

```
<matplotlib.collections.PathCollection at 0x7e339666fa00>
```




```
df.corr()
```

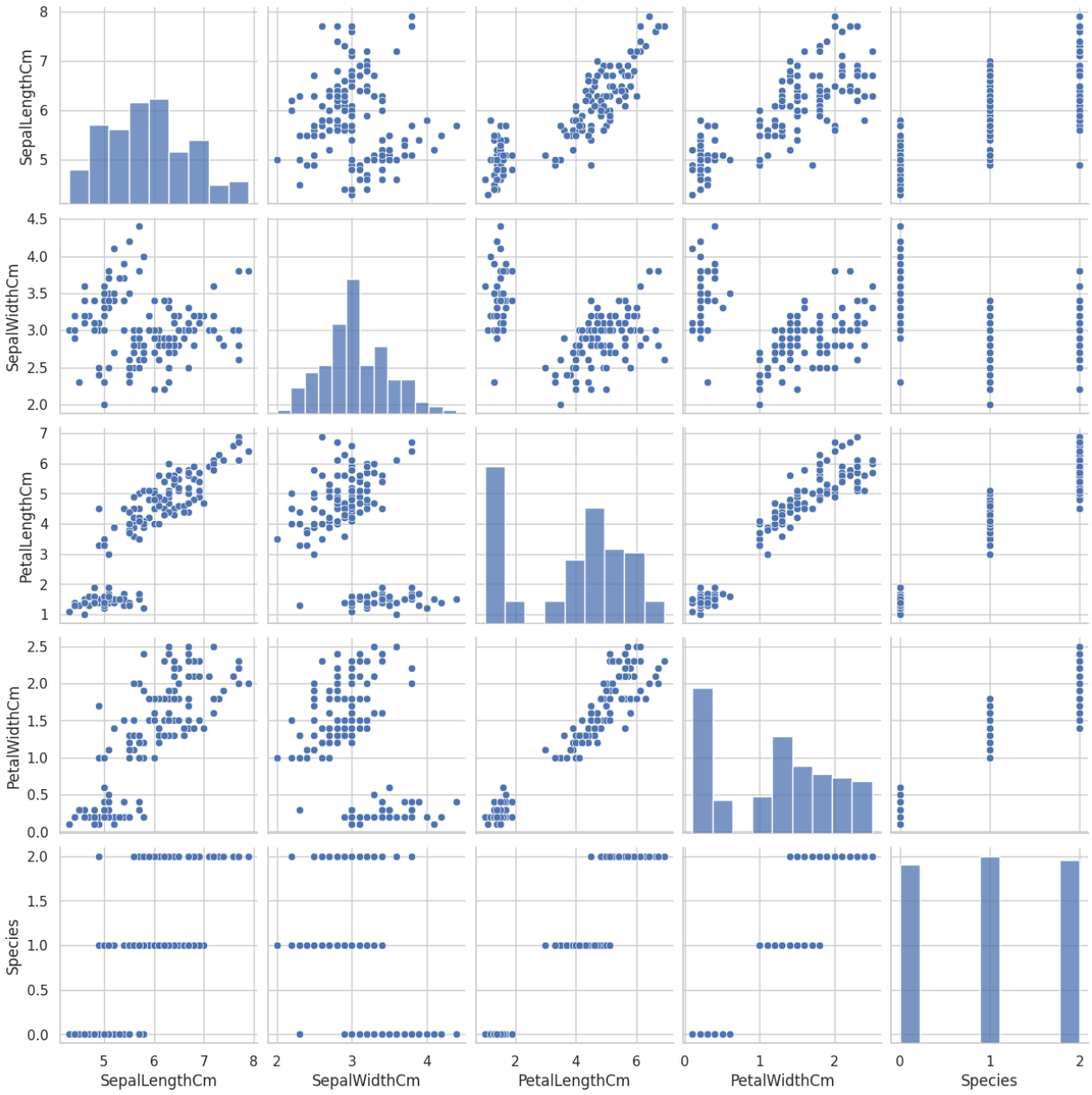
{"summary":"{\n \"name\": \"df\",\n \"rows\": 5,\n \"fields\": [\n{\n   \"column\": \"SepalLengthCm\",\n   \"properties\": {\n\"dtype\": \"number\",\n   \"std\": 0.444721620820587,\n\"min\": -0.1093208232854695,\n   \"max\": 1.0,\n\"num_unique_values\": 5,\n   \"samples\": [\n   -0.1093208232854695,\n   0.7829035213482042,\n   0.8713045563091528\n   ],\n   \"semantic_type\": \"\",\n\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"SepalWidthCm\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.606762266931114,\n   \"min\": -0.42105739409782933,\n   \"max\": 1.0,\n\"num_unique_values\": 5,\n   \"samples\": [\n   1.0,\n   -0.41834785749138836,\n   -0.42105739409782933\n   ],\n\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"PetalLengthCm\",\n\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.6128775795331841,\n   \"min\": -0.42105739409782933,\n\"max\": 1.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   -0.42105739409782933,\n   0.9483385082992367,\n   1.0\n   ],\n   \"semantic_type\": \"\",\n\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"PetalWidthCm\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.5810831767696716,\n   \"min\": -0.3563761603697159,\n   \"max\": 1.0,\n\"num_unique_values\": 5,\n   \"samples\": [\n   -0.3563761603697159,\n   0.9556932449750069,\n   0.9618827517033721\n   ],\n   \"semantic_type\": \"\",\n\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Species\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.6049619032913631,\n   \"min\": -0.41834785749138836,\n   \"max\": 1.0,\n\"num_unique_values\": 5,\n   \"samples\": [\n   -0.41834785749138836,\n   1.0,\n   0.9483385082992367\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
fig=plt.figure(figsize=(15,12))
sns.heatmap(df.corr(),linewidths=1,annot=True)
```

```
<Axes: >
```


```
sns.pairplot(df)
```

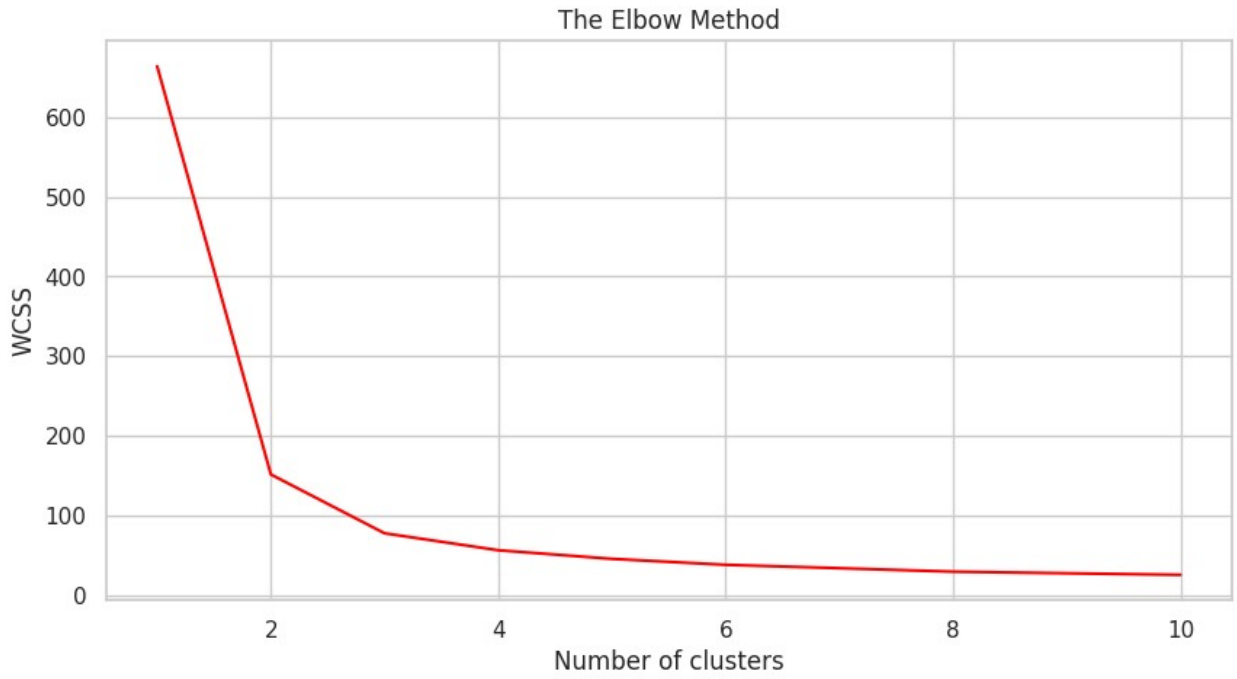
```
<seaborn.axisgrid.PairGrid at 0x7e339656fb50>
```




```
df=df.iloc[:,[0,1,2,3]].values

from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state =
42)
    kmeans.fit(df)
    # inertia method returns wcss for that model
    wcss.append(kmeans.inertia_)

wcss
```

```
[663.895238095238,
 151.77145833333333,
 77.91989035087718,
 56.64237065018315,
 45.81642192982456,
 38.380978808131445,
 34.117070947570944,
 29.758015809726338,
 27.74499792038028,
 25.769652682285034]
```

```
plt.figure(figsize=(10,5))
sns.set(style='whitegrid')
sns.lineplot(x=range(1, 11), y=wcss, markers = 'o', color='red')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```




```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 5)
y_kmeans = kmeans.fit_predict(df)
y_kmeans
```

```
array([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, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2,
2,
       2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2,
0,
       2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 2, 0], dtype=int32)
```

```
fig = plt.figure(figsize=(10, 7))
plt.title('Clusters with Centroids',fontweight ='bold', fontsize=20)
plt.scatter(df[y_kmeans == 0, 2], df[y_kmeans == 0, 3], s = 100, c =
'seagreen', label = 'Iris-versicolour')
plt.scatter(df[y_kmeans == 1, 2], df[y_kmeans == 1, 3], s = 100, c =
'purple', label = 'Iris-setosa')
plt.scatter(df[y_kmeans == 2, 2], df[y_kmeans == 2, 3],s = 100, c =
'yellow', label = 'Iris-virginica')
plt.scatter(kmeans.cluster_centers_[:, 2],
kmeans.cluster_centers_[:,3], s = 300, c = 'red',marker='*',
            label = 'Centroids')
plt.title('Iris Flower Clusters')
plt.ylabel('Petal Width in cm')
plt.xlabel('Petal Length in cm')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7e3395daae60>
```

Task - 3 : Exploratory Data Analysis - Retail

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('/content/Retail(Dataset).csv')  #loading dataset
df.head()

{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 9994,\n  \"fields\":
[\n    {\n      \"column\": \"Ship Mode\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"Standard Class\",\n          \"Same
Day\",\n          \"Second Class\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Segment\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
```

```
3,\n        \"samples\": [\n          \"Consumer\",\n
\"Corporate\",\n          \"Home Office\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Country\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
1,\n        \"samples\": [\n          \"United States\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"City\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 531,\n
\"samples\": [\n          \"Laurel\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"State\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 49,\
n        \"samples\": [\n          \"Delaware\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Postal Code\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
32063,\n        \"min\": 1040,\n        \"max\": 99301,\n
\"num_unique_values\": 631,\n        \"samples\": [\n          30062\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Region\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
4,\n        \"samples\": [\n          \"West\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Category\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"Furniture\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sub-Category\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 17,\n        \"samples\": [\n
\"Bookcases\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Sales\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 623.2451005086807,\n        \"min\": 0.444,\n        \"max\":
22638.48,\n        \"num_unique_values\": 5825,\n        \"samples\":
[\n          2624.985\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Quantity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2,\n        \"min\": 1,\n
\"max\": 14,\n        \"num_unique_values\": 14,\n        \"samples\":
[\n          14\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Discount\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.20645196782571615,\n        \"min\":
0.0,\n        \"max\": 0.8,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          0.4\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Profit\",\n      \"properties\": {\n        \"dtype\":
```

\"number\",\n   \"std\": 234.2601076909573,\n   \"min\": -6599.978,\n   \"max\": 8399.976,\n   \"num_unique_values\": 7287,\n   \"samples\": [\n   -183.6324\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df.tail()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 5,\n \"fields\": [\n{\n   \"column\": \"Ship Mode\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"Standard Class\",\n   \"Second Class\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Segment\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Consumer\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Country\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"United States\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"City\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"Miami\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"State\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"California\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Postal Code\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 26591,\n   \"min\": 33180,\n   \"max\": 92683,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   33180\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Region\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"West\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Category\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"Furniture\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sub-Category\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 4,\n   \"samples\": [\n   \"Phones\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sales\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 113.83840794389212,\n   \"min\": 25.248,\n

\"max\": 258.576,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   91.96\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Quantity\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 0,\n
\"min\": 2,\n   \"max\": 4,\n   \"num_unique_values\": 3,\n
\"samples\": [\n   3\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Discount\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.10954451150103324,\n   \"min\":
0.0,\n   \"max\": 0.2,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   0.0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Profit\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 27.34624136308315,\n   \"min\":
4.1028,\n   \"max\": 72.948,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   15.6332\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

df.shape

```
(9994, 13)
```

df.describe()

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n
{\n   \"column\": \"Postal Code\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 35860.31406157158,\n
\"min\": 1040.0,\n   \"max\": 99301.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
55190.3794276566,\n   56430.5,\n   9994.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Sales\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 8197.010918685499,\n
\"min\": 0.444,\n   \"max\": 22638.48,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
229.85800083049833,\n   54.489999999999995,\n   9994.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Quantity\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
3531.848471644344,\n   \"min\": 1.0,\n   \"max\": 9994.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
3.789573744246548,\n   3.0,\n   9994.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Discount\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
3533.3336684667293,\n   \"min\": 0.0,\n   \"max\": 9994.0,\n
\"num_unique_values\": 6,\n   \"samples\": [\n   9994.0,\n
0.15620272163297977,\n   0.8\n   ],\n

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Profit\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 5288.326642672474,\n   \"min\": -6599.978,\n   \"max\": 9994.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n 28.65689630778467,\n   8.6665,\n   9994.0\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
df.isnull().sum()       #checking null values
```

```
Ship Mode       0
Segment         0
Country         0
City            0
State           0
Postal Code     0
Region          0
Category        0
Sub-Category    0
Sales           0
Quantity        0
Discount        0
Profit          0
dtype: int64
```

```
df.info() #information about dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Ship Mode     9994 non-null   object 
 1   Segment       9994 non-null   object 
 2   Country       9994 non-null   object 
 3   City          9994 non-null   object 
 4   State         9994 non-null   object 
 5   Postal Code   9994 non-null   int64  
 6   Region        9994 non-null   object 
 7   Category      9994 non-null   object 
 8   Sub-Category  9994 non-null   object 
 9   Sales         9994 non-null   float64
 10  Quantity      9994 non-null   int64  
 11  Discount      9994 non-null   float64
 12  Profit        9994 non-null   float64
dtypes: float64(3), int64(2), object(8)
memory usage: 1015.1+ KB
```

```
df.columns
```

```
Index(['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Postal
Code',
       'Region', 'Category', 'Sub-Category', 'Sales', 'Quantity',
'Discount',
       'Profit'],
      dtype='object')
```

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

```
17
```

```
df.nunique()
```

```
Ship Mode          4
Segment            3
Country            1
City             531
State             49
Postal Code      631
Region             4
Category           3
Sub-Category      17
Sales           5825
Quantity          14
Discount          12
Profit          7287
dtype: int64
```

```
df['Postal Code'] = df['Postal Code'].astype('object')
```

```
df.drop_duplicates(subset=None,keep='first',inplace=True)
df.duplicated().sum()
```

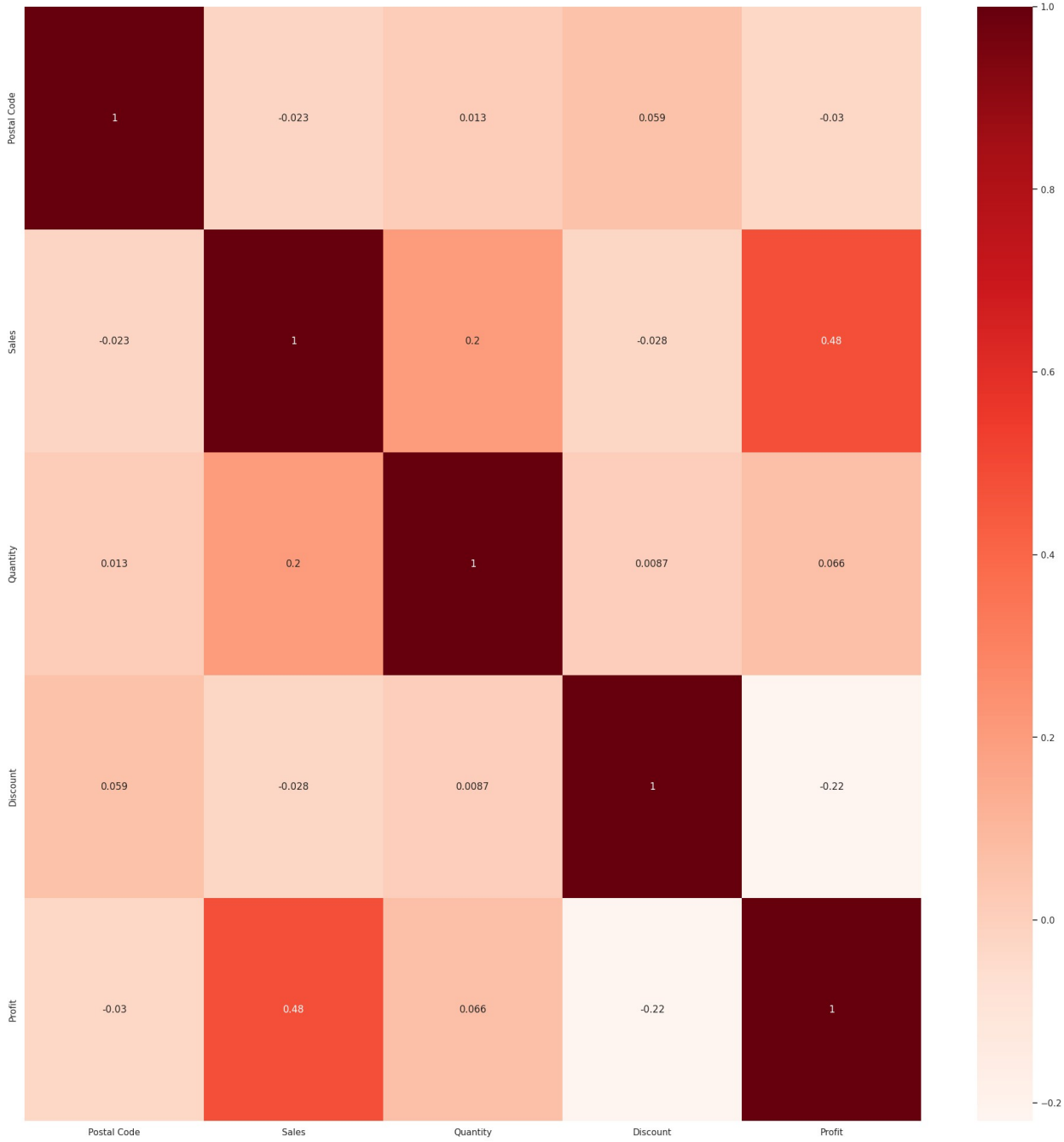
```
0
```

```
df['Postal Code'] = df['Postal Code'].astype(float)
```

```
df_numeric = df.select_dtypes(include=['number'])
corr = df_numeric.corr()
```

```
corr = df_numeric.corr()
sns.heatmap(corr,annot=True,cmap='Reds')
```

```
<Axes: >
```




```
df = df.drop(['Postal Code'],axis = 1)

sns.pairplot(df, hue = 'Ship Mode')
```

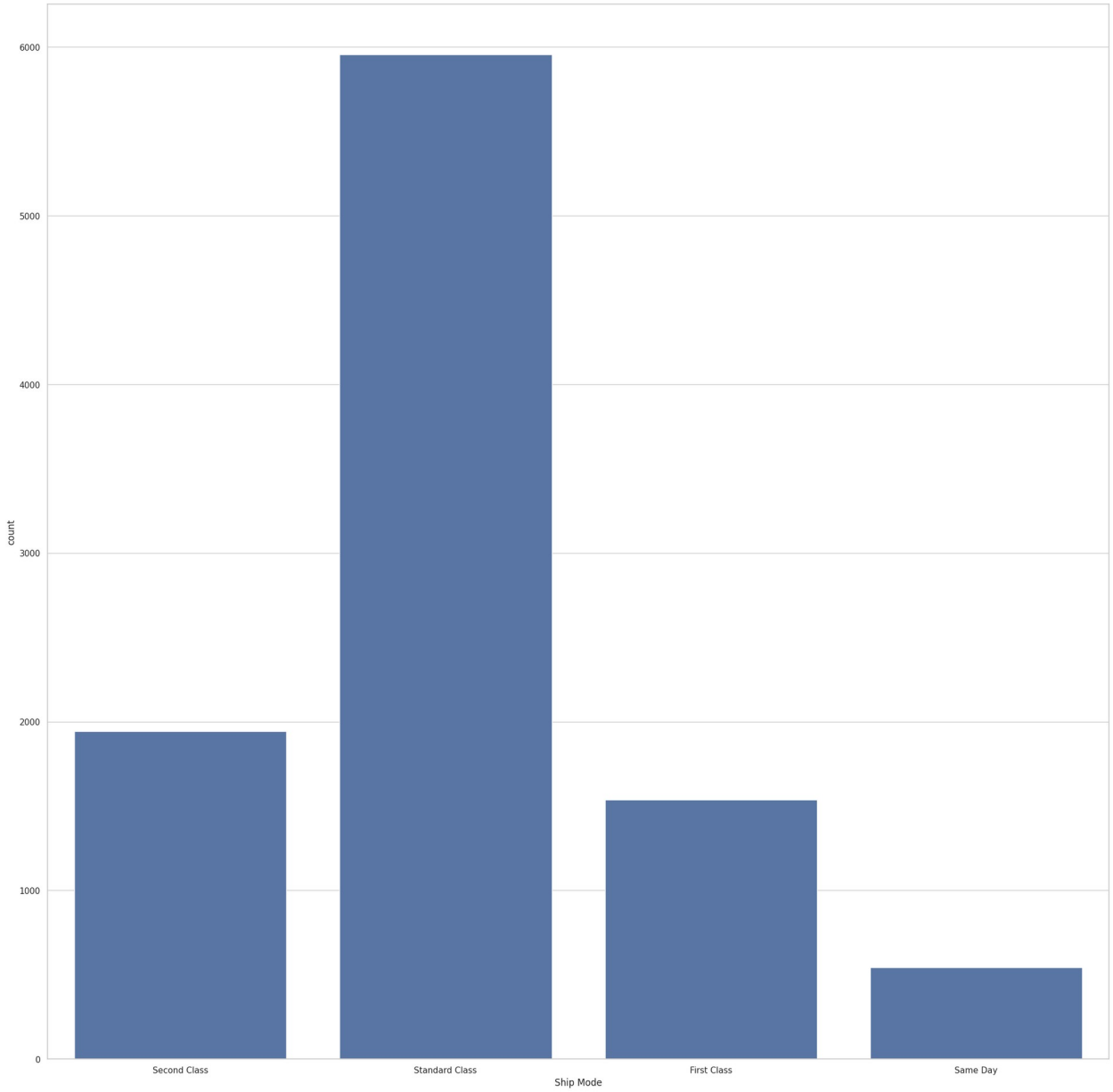
```
<seaborn.axisgrid.PairGrid at 0x7e3395cd7c70>
```


```
df['Ship Mode'].value_counts()
```

```
Ship Mode
Standard Class    5955
Second Class      1943
First Class       1537
Same Day           542
Name: count, dtype: int64
```

```
sns.countplot(x=df['Ship Mode'])
```

```
<Axes: xlabel='Ship Mode', ylabel='count'>
```

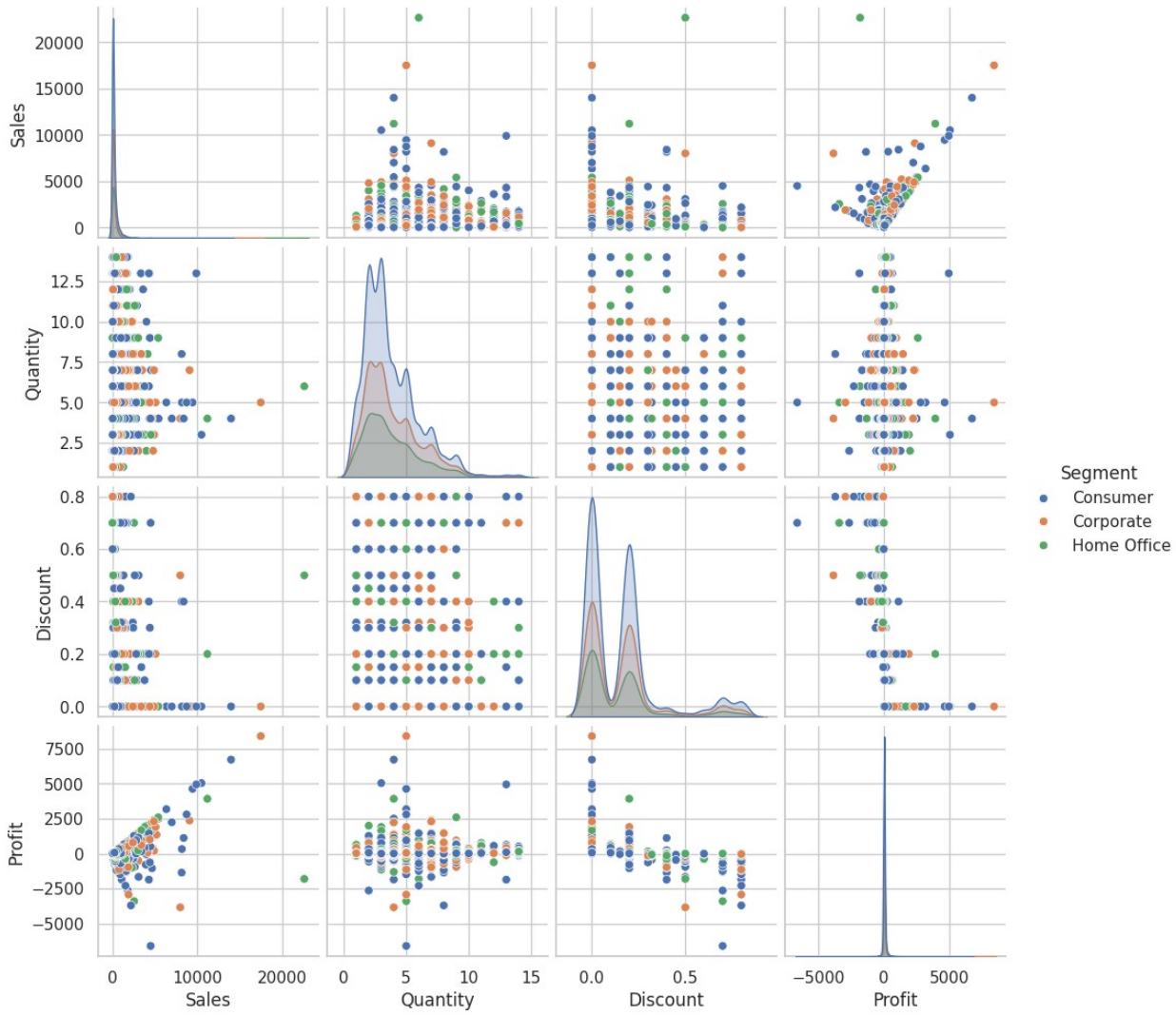



```
df['Segment'].value_counts()
```

```
Segment
Consumer       5183
Corporate      3015
Home Office    1779
Name: count, dtype: int64
```

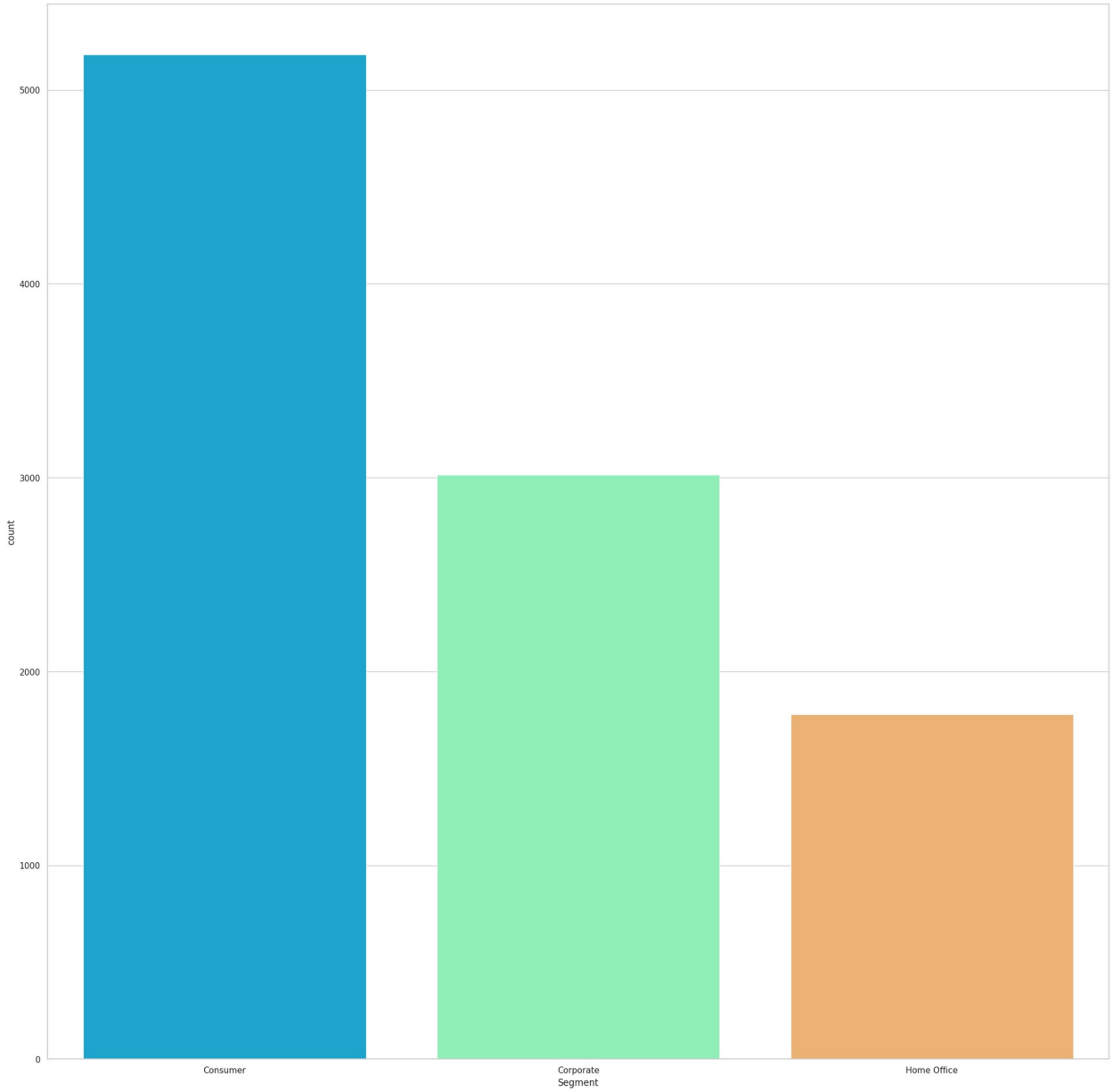
```
sns.pairplot(df,hue = 'Segment')
```

```
<seaborn.axisgrid.PairGrid at 0x7e33a45e0af0>
```




```
sns.countplot(x = 'Segment',data = df, palette = 'rainbow')
```

```
<Axes: xlabel='Segment', ylabel='count'>
```

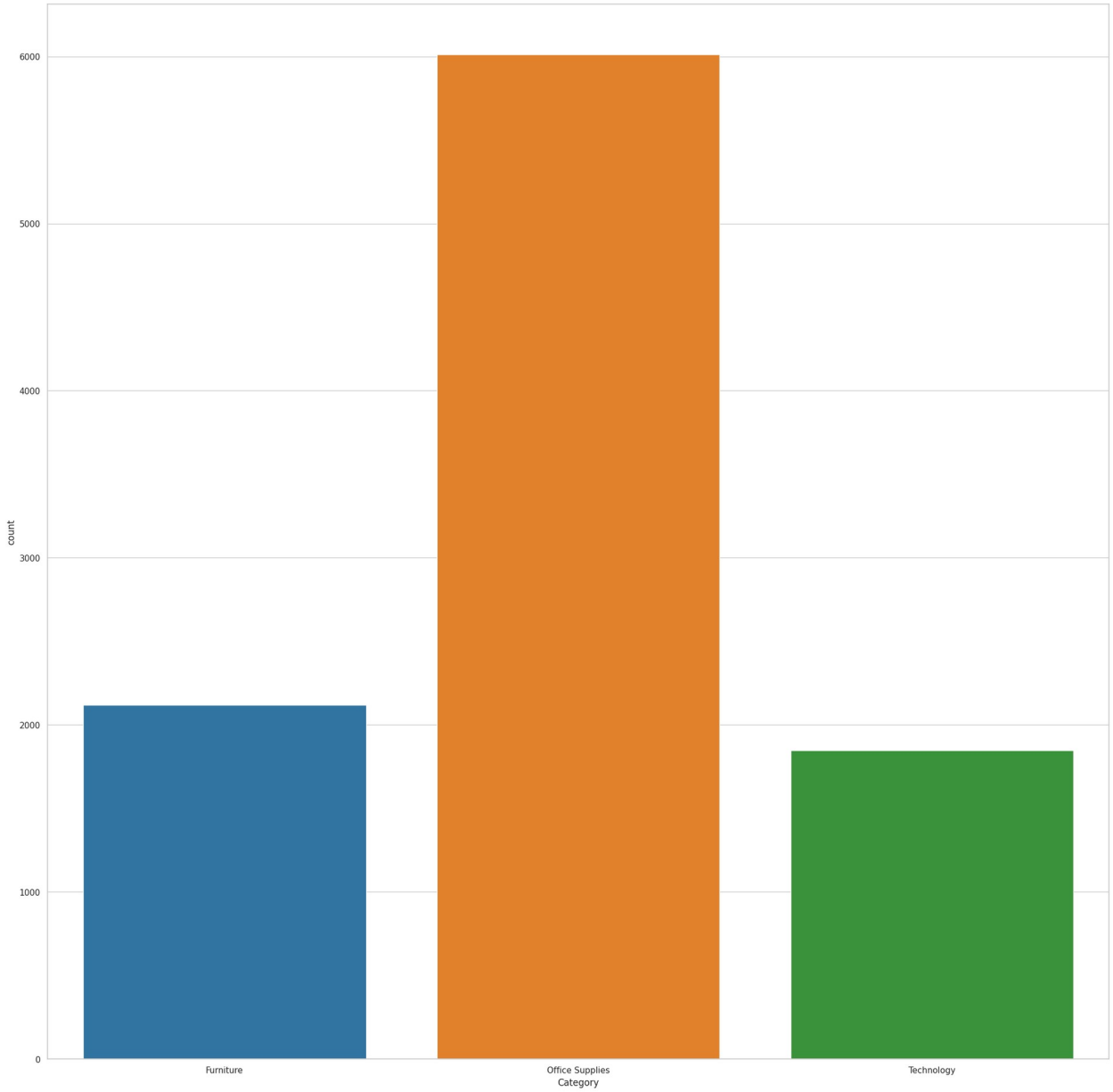



```
df['Category'].value_counts()
```

```
Category
Office Supplies    6012
Furniture          2118
Technology         1847
Name: count, dtype: int64
```

```
sns.countplot(x='Category',data=df,palette='tab10')
```

```
<Axes: xlabel='Category', ylabel='count'>
```




```
sns.pairplot(df,hue='Category')
```

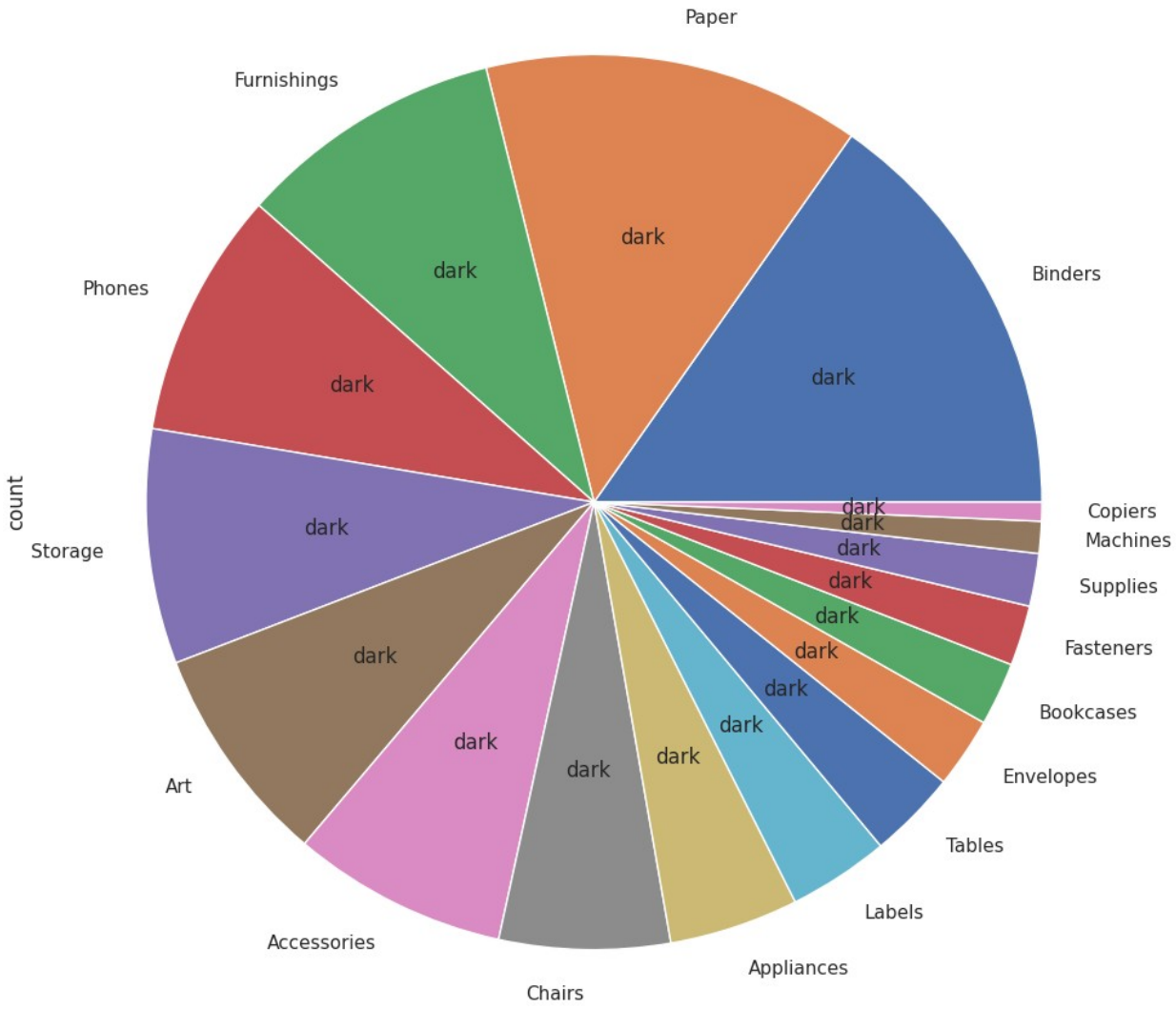
```
<seaborn.axisgrid.PairGrid at 0x7e3394bd9630>
```


```
df['Sub-Category'].value_counts()
```

```
Sub-Category
Binders        1522
Paper          1359
Furnishings     956
Phones          889
Storage         846
Art             795
Accessories     775
Chairs          615
Appliances      466
Labels          363
Tables          319
Envelopes       254
Bookcases       228
Fasteners       217
Supplies        190
```

```
Machines        115
Copiers          68
Name: count, dtype: int64
```

```
plt.figure(figsize=(15,12))
df['Sub-Category'].value_counts().plot.pie(autopct='dark')
plt.show()
```

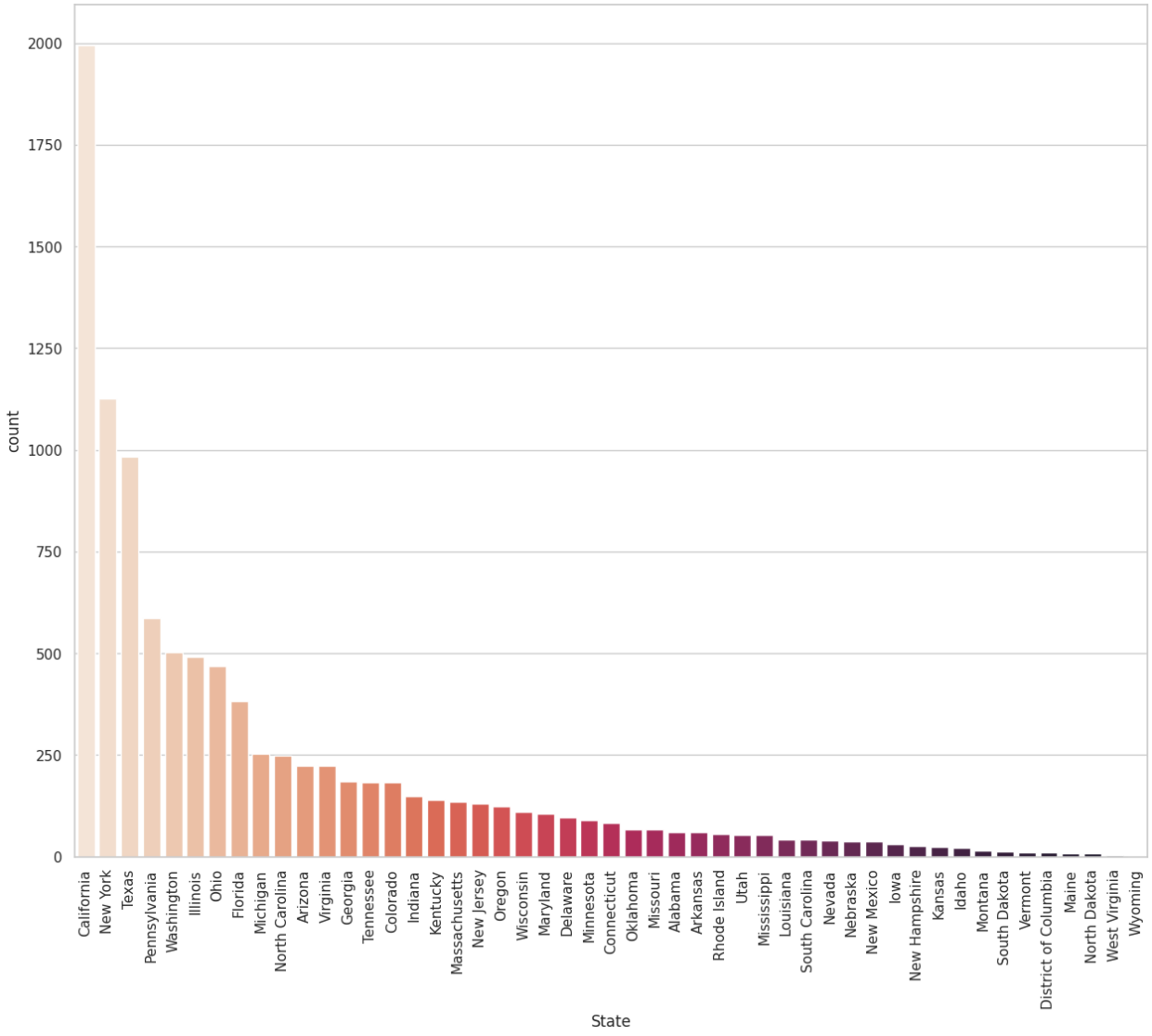



```
df['State'].value_counts()
```

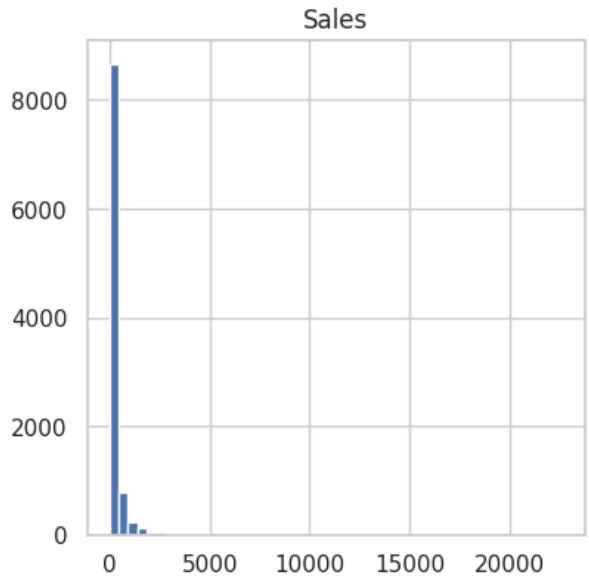
```
State
California                1996
```

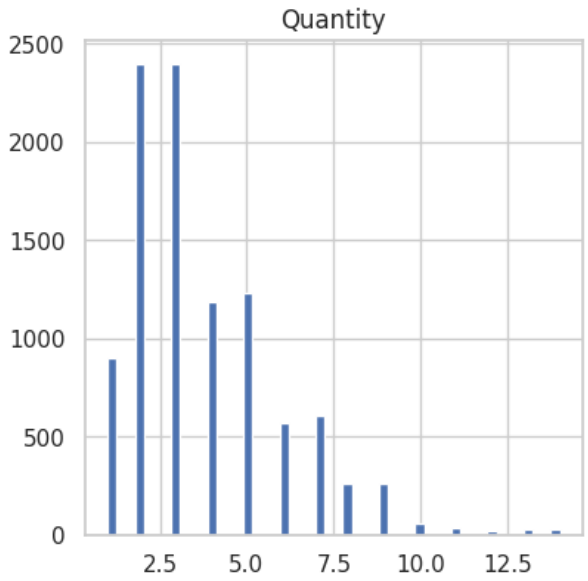
```
New York                1127
Texas                    983
Pennsylvania             586
Washington               502
Illinois                 491
Ohio                     468
Florida                  383
Michigan                 254
North Carolina           249
Arizona                  224
Virginia                 224
Georgia                  184
Tennessee                183
Colorado                 182
Indiana                  149
Kentucky                 139
Massachusetts            135
New Jersey               130
Oregon                   123
Wisconsin                110
Maryland                 105
Delaware                  96
Minnesota                 89
Connecticut               82
Oklahoma                  66
Missouri                  66
Alabama                   61
Arkansas                  60
Rhode Island              56
Utah                      53
Mississippi               53
Louisiana                 42
South Carolina            42
Nevada                    39
Nebraska                  38
New Mexico                37
Iowa                      30
New Hampshire             27
Kansas                    24
Idaho                     21
Montana                   15
South Dakota              12
Vermont                   11
District of Columbia      10
Maine                      8
North Dakota               7
West Virginia              4
Wyoming                    1
Name: count, dtype: int64
```

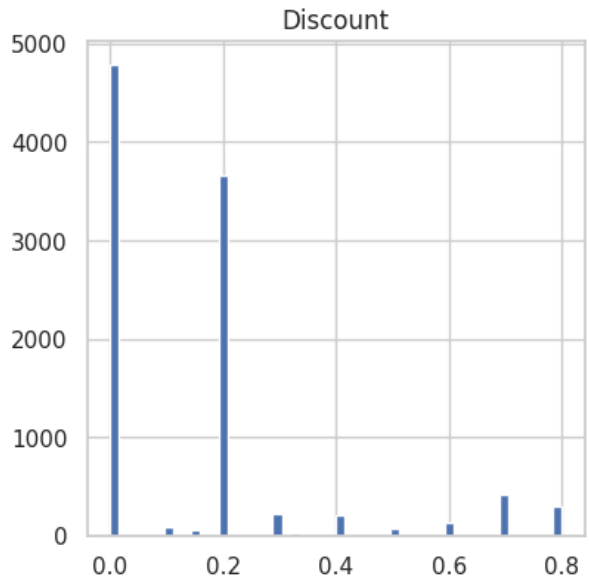
```
plt.figure(figsize=(15,12))
sns.countplot(x='State',data=df,palette='rocket_r',order=df['State'].v
alue_counts().index)
plt.xticks(rotation=90)
plt.show()
```

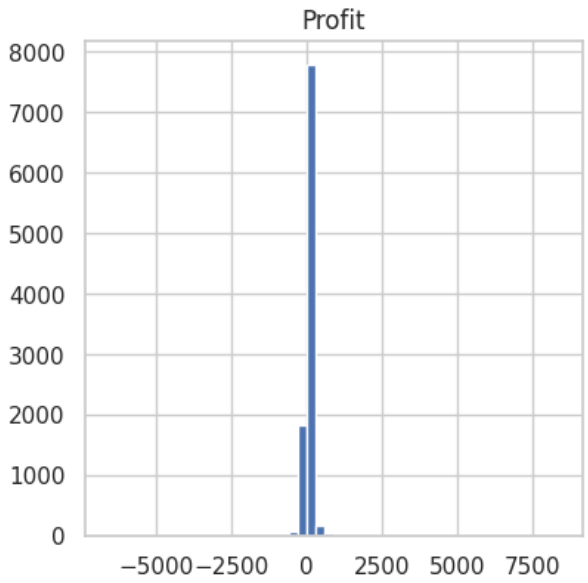



```
df.hist(figsize=(10,10),bins=50)
plt.show()
```

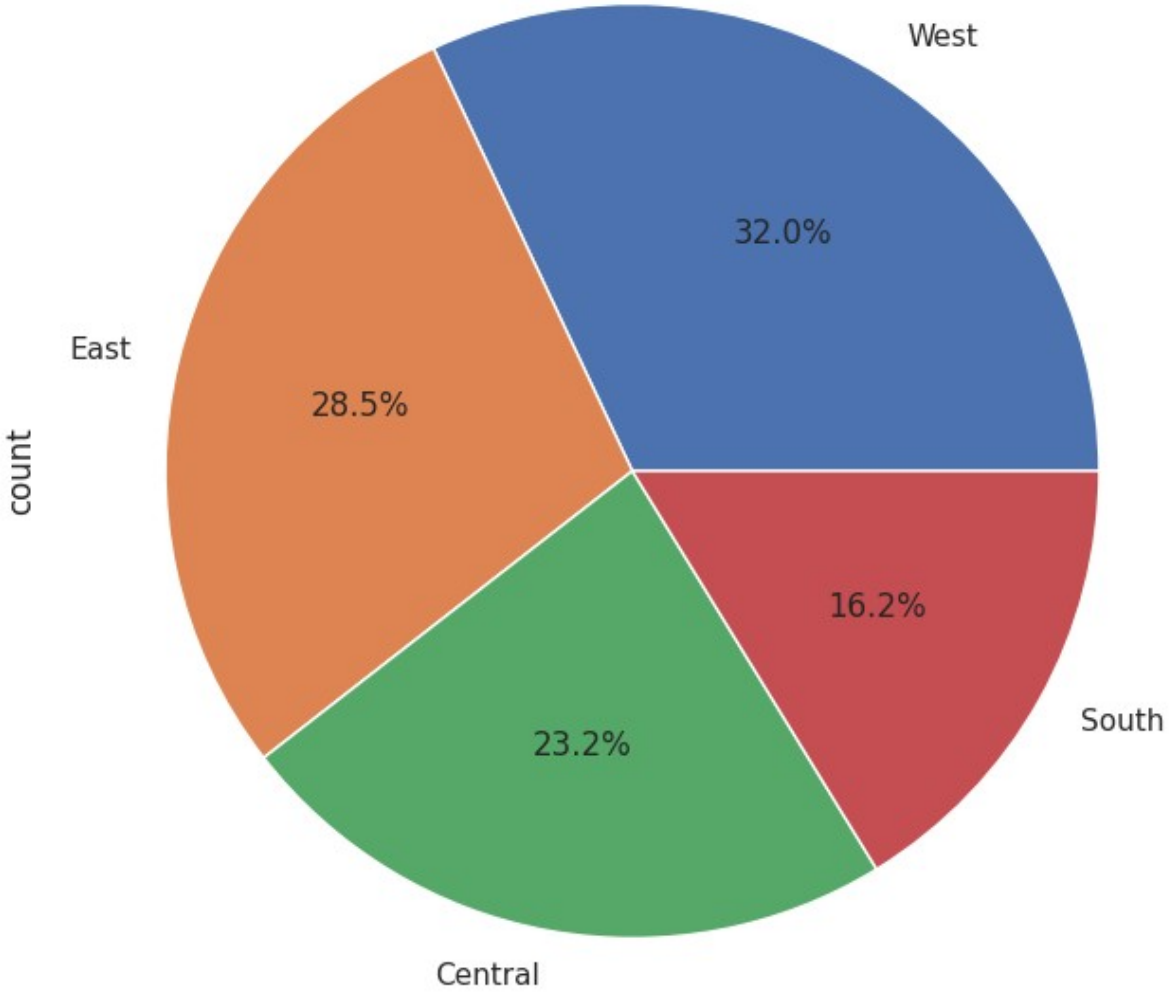




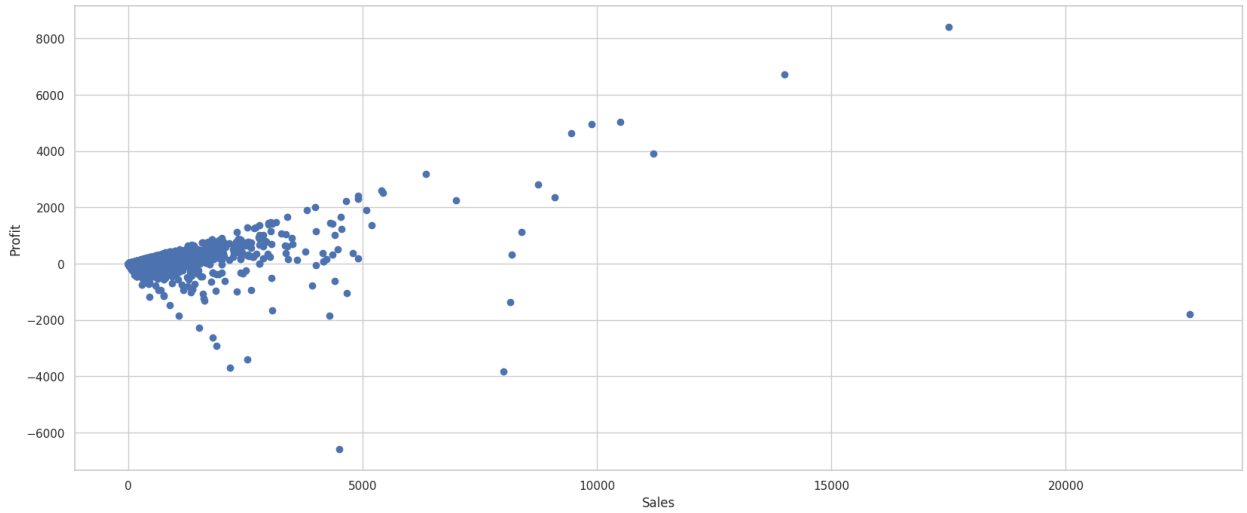




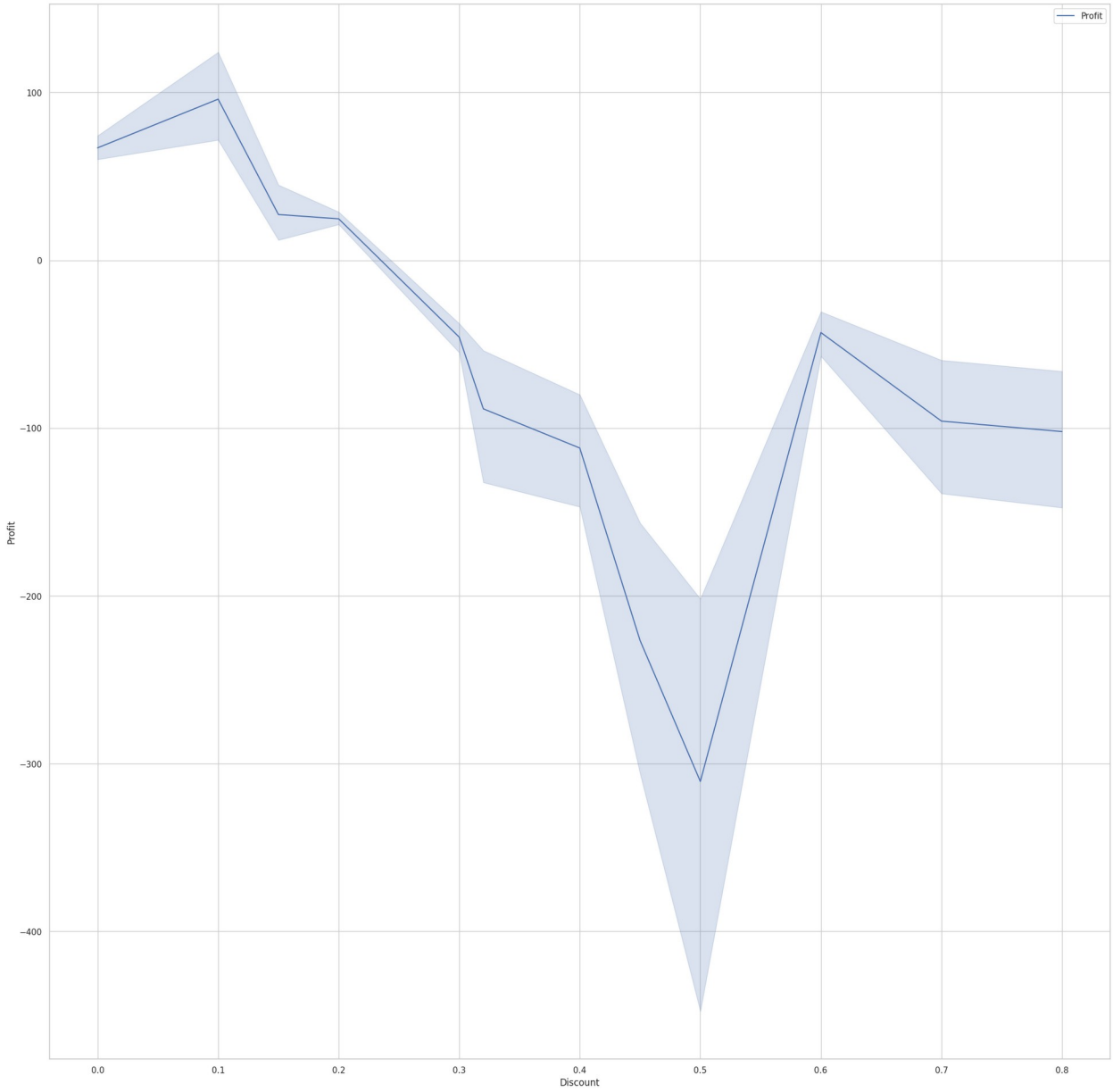

```
plt.figure(figsize=(10,8))
df['Region'].value_counts().plot.pie(autopct = '%1.1f%%')
plt.show()
```



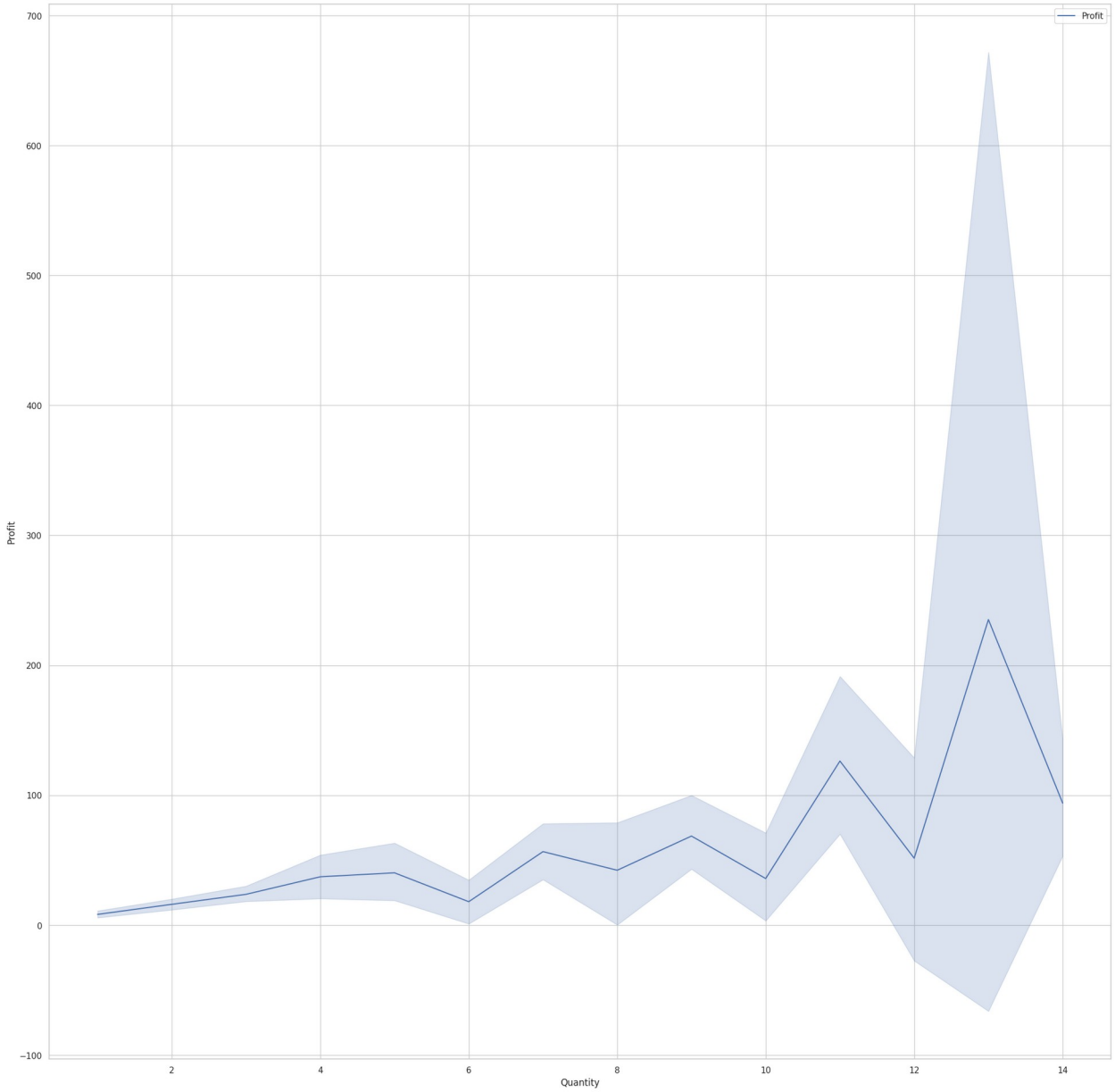

```
fig,ax=plt.subplots(figsize=(20,8))
ax.scatter(df['Sales'],df['Profit'])
ax.set_xlabel('Sales')
ax.set_ylabel('Profit')
plt.show()
```



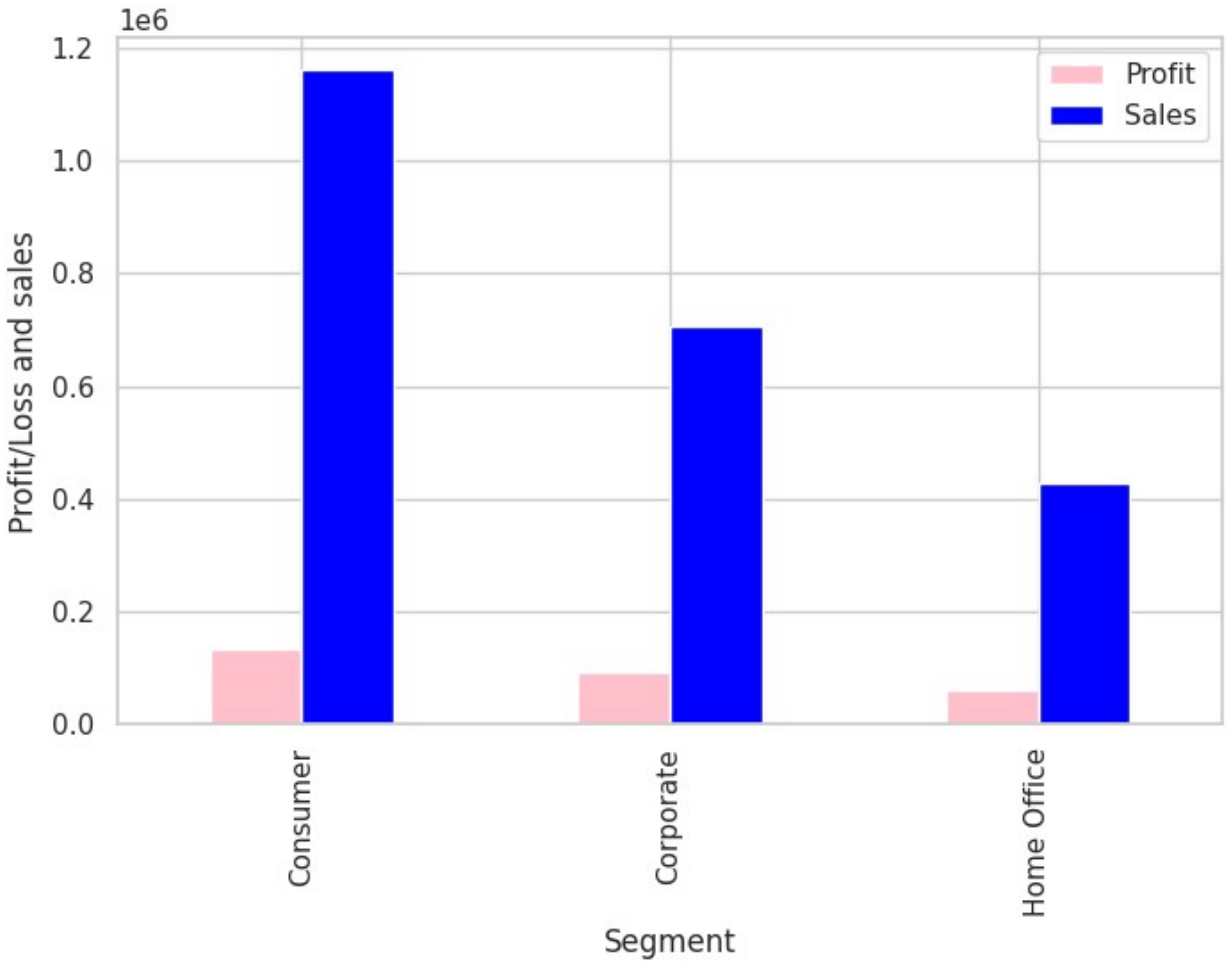

```
sns.lineplot(x='Discount',y='Profit',label='Profit',data=df)
plt.legend()
plt.show()
```



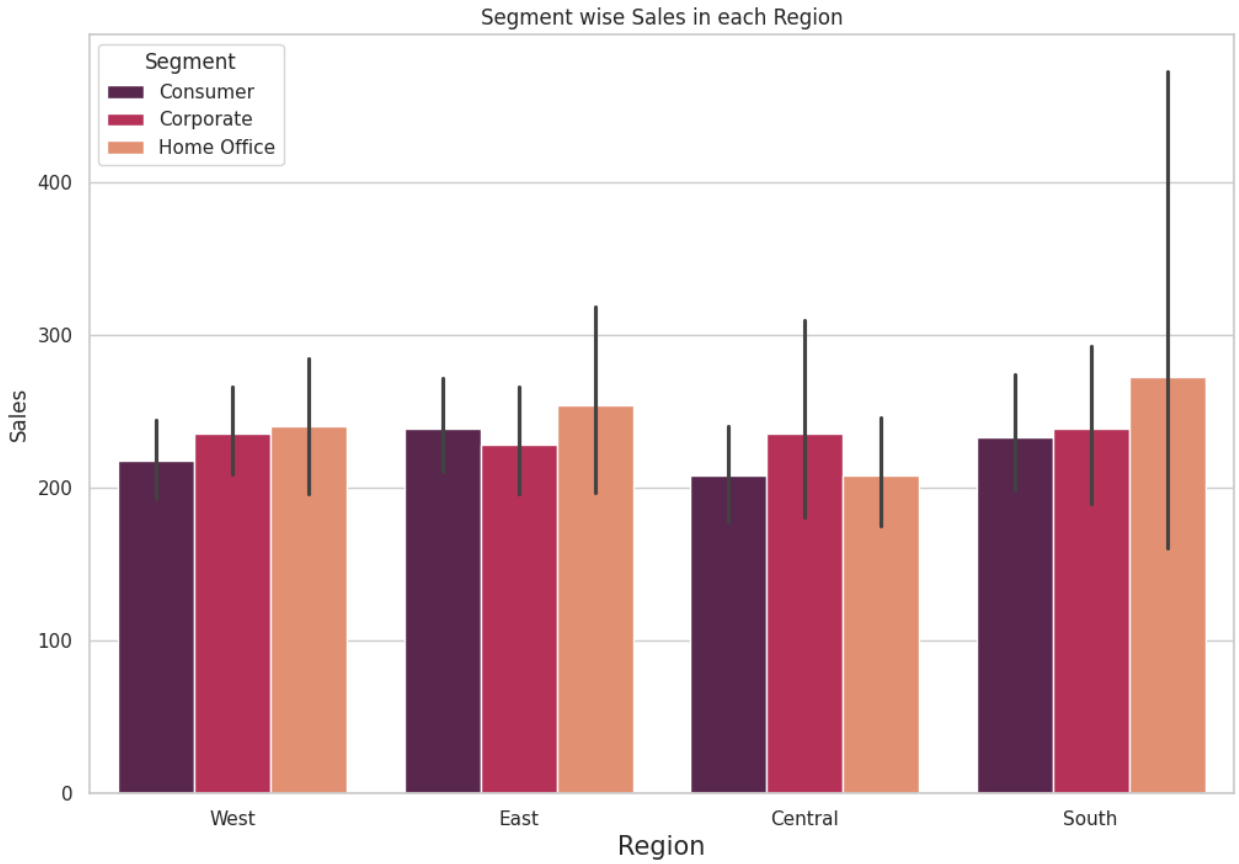

```
sns.lineplot(x='Quantity',y='Profit',label='Profit',data=df)
plt.legend()
plt.show()
```



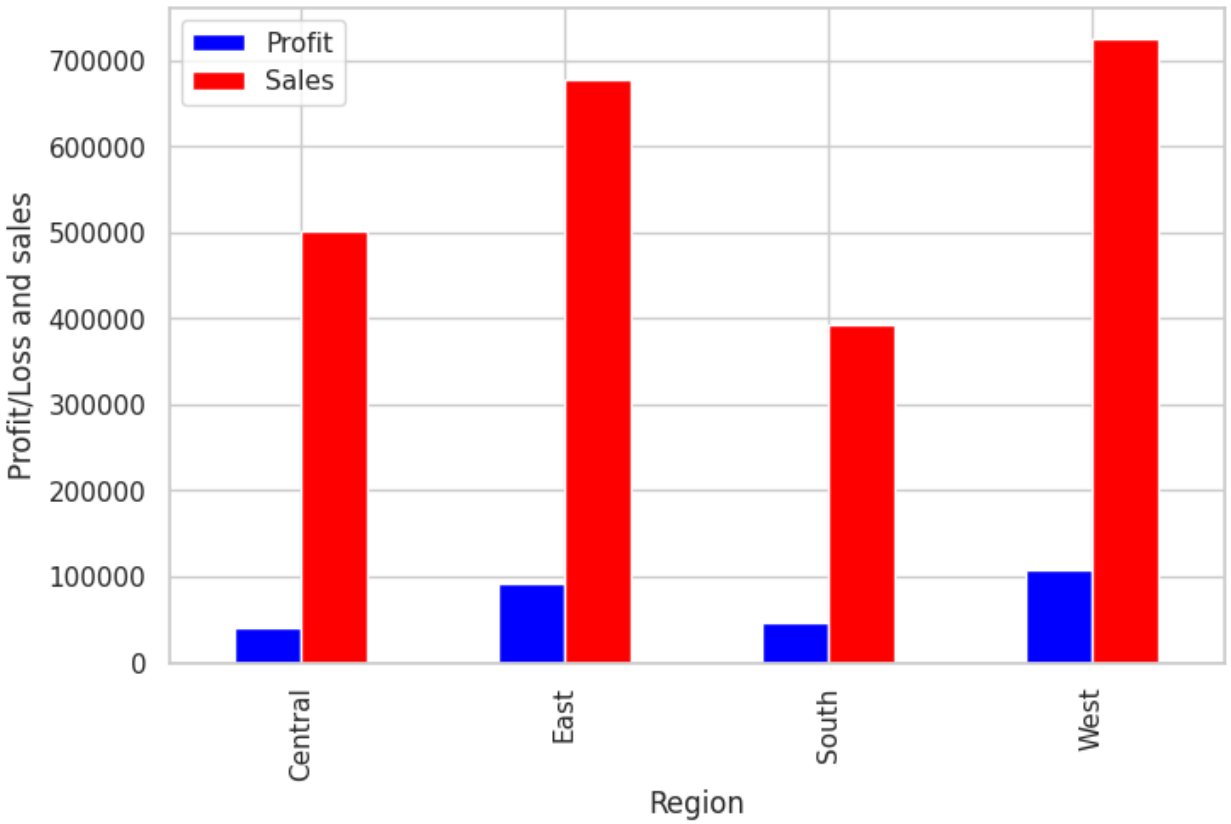

```
df.groupby('Segment')
[['Profit','Sales']].sum().plot.bar(color=['pink','blue'],figsize=(8,5
))
plt.ylabel('Profit/Loss and sales')
plt.show()
```



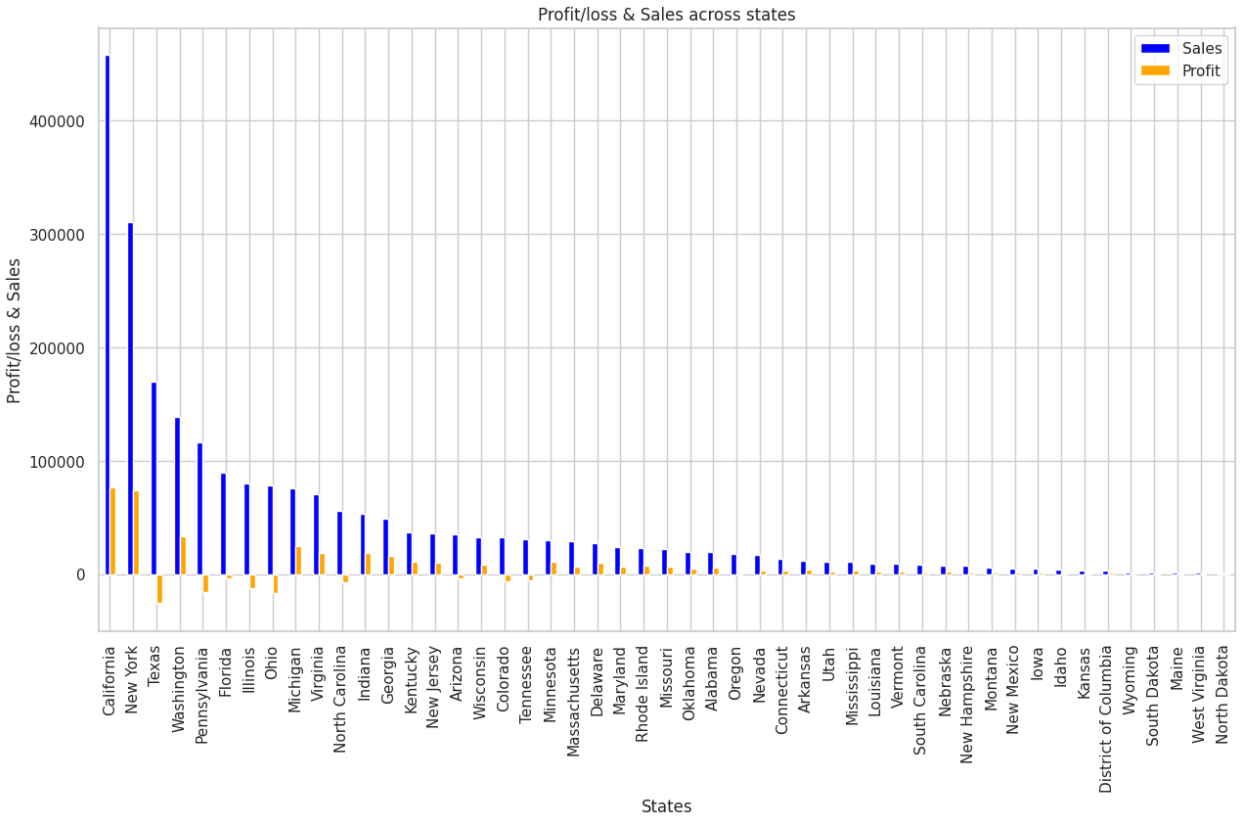

```
plt.figure(figsize=(12,8))
plt.title('Segment wise Sales in each Region')
sns.barplot(x='Region',y='Sales',data=df,hue='Segment',order=df['Regio
n'].value_counts().index,palette='rocket')
plt.xlabel('Region',fontsize=15)
plt.show()
```




```
df.groupby('Region')
[['Profit','Sales']].sum().plot.bar(color=['blue','red'],figsize=(8,5)
)
plt.ylabel('Profit/Loss and sales')
plt.show()
```



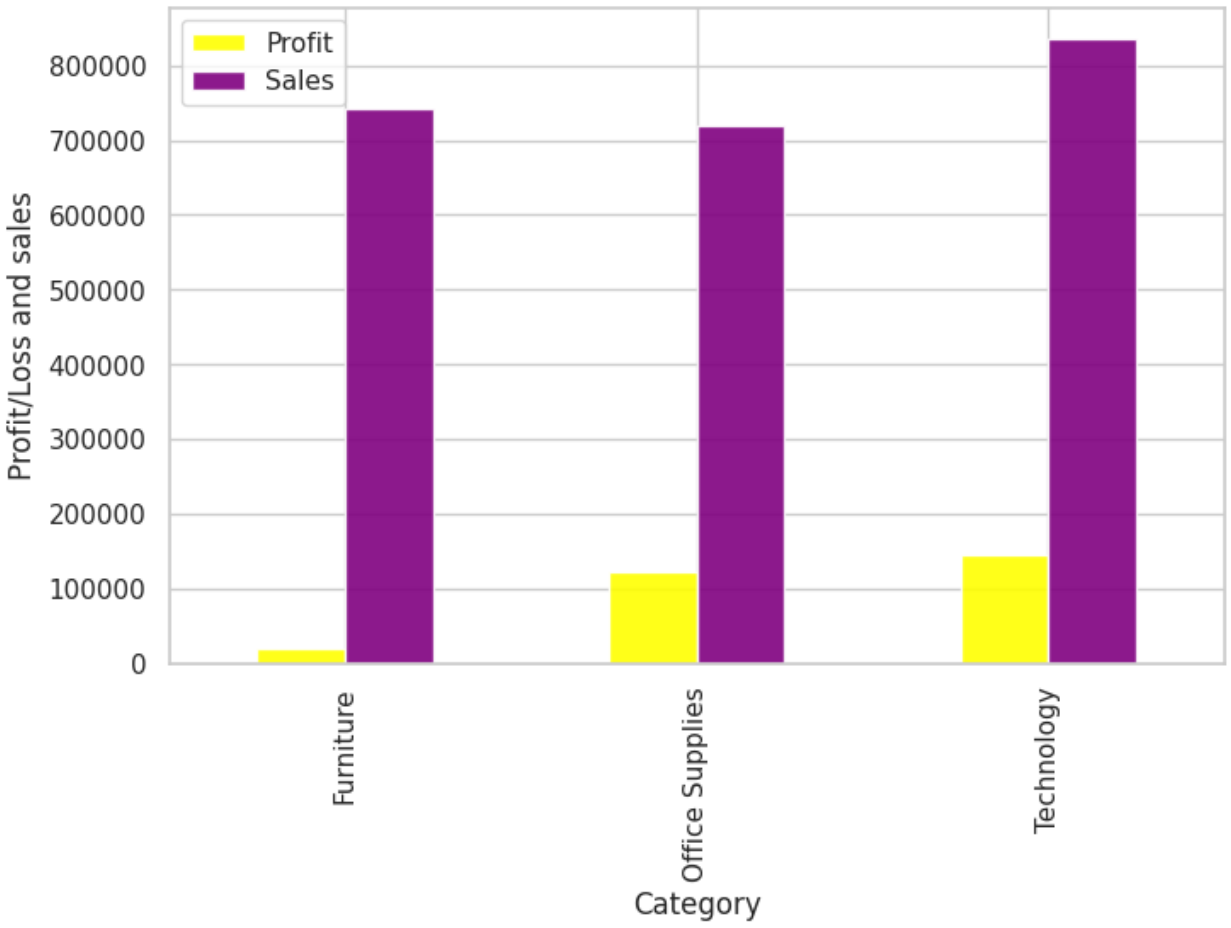

```
ps = df.groupby('State')
[['Sales','Profit']].sum().sort_values(by='Sales',ascending=False)
ps[:].plot.bar(color=['blue','orange'],figsize=(15,8))
plt.title('Profit/loss & Sales across states')
plt.xlabel('States')
plt.ylabel('Profit/loss & Sales')
plt.show()
```



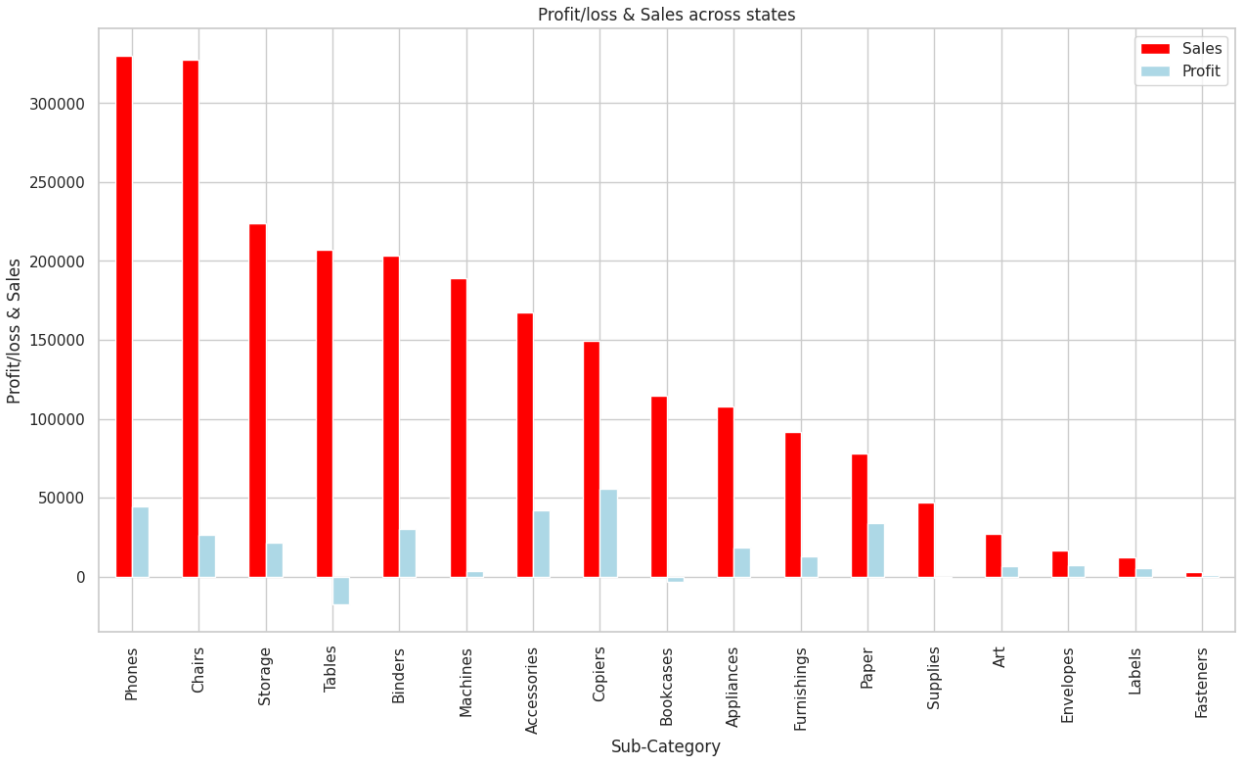

```
t_states = df['State'].value_counts().nlargest(10)
t_states
```

```
State
California        1996
New York          1127
Texas              983
Pennsylvania       586
Washington         502
Illinois           491
Ohio               468
Florida            383
Michigan           254
North Carolina     249
Name: count, dtype: int64
```

```
df.groupby('Category')
[['Profit','Sales']].sum().plot.bar(color=['yellow','purple'],alpha=0.
9,figsize=(8,5))
plt.ylabel('Profit/Loss and sales')
plt.show()
```




```
ps = df.groupby('Sub-Category')
[['Sales','Profit']].sum().sort_values(by='Sales',ascending=False)
ps[:].plot.bar(color=['red','lightblue'],figsize=(15,8))
plt.title('Profit/loss & Sales across states')
plt.xlabel('Sub-Category')
plt.ylabel('Profit/loss & Sales')
plt.show()
```



Task - 4 : Exploratory Data Analysis

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
!py -m pip install folium
```

```
/bin/bash: line 1: py: command not found
```

```
import folium
from folium.plugins import MarkerCluster
df =
pd.read_csv("/content/globalterrorismdb_0718dist.csv",encoding='latin1
')
df
```

```
{"type":"dataframe","variable_name":"df"}
```

```
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181691 entries, 0 to 181690
Columns: 135 entries, eventid to related
```

```
dtypes: float64(55), int64(22), object(58)
memory usage: 187.1+ MB
```

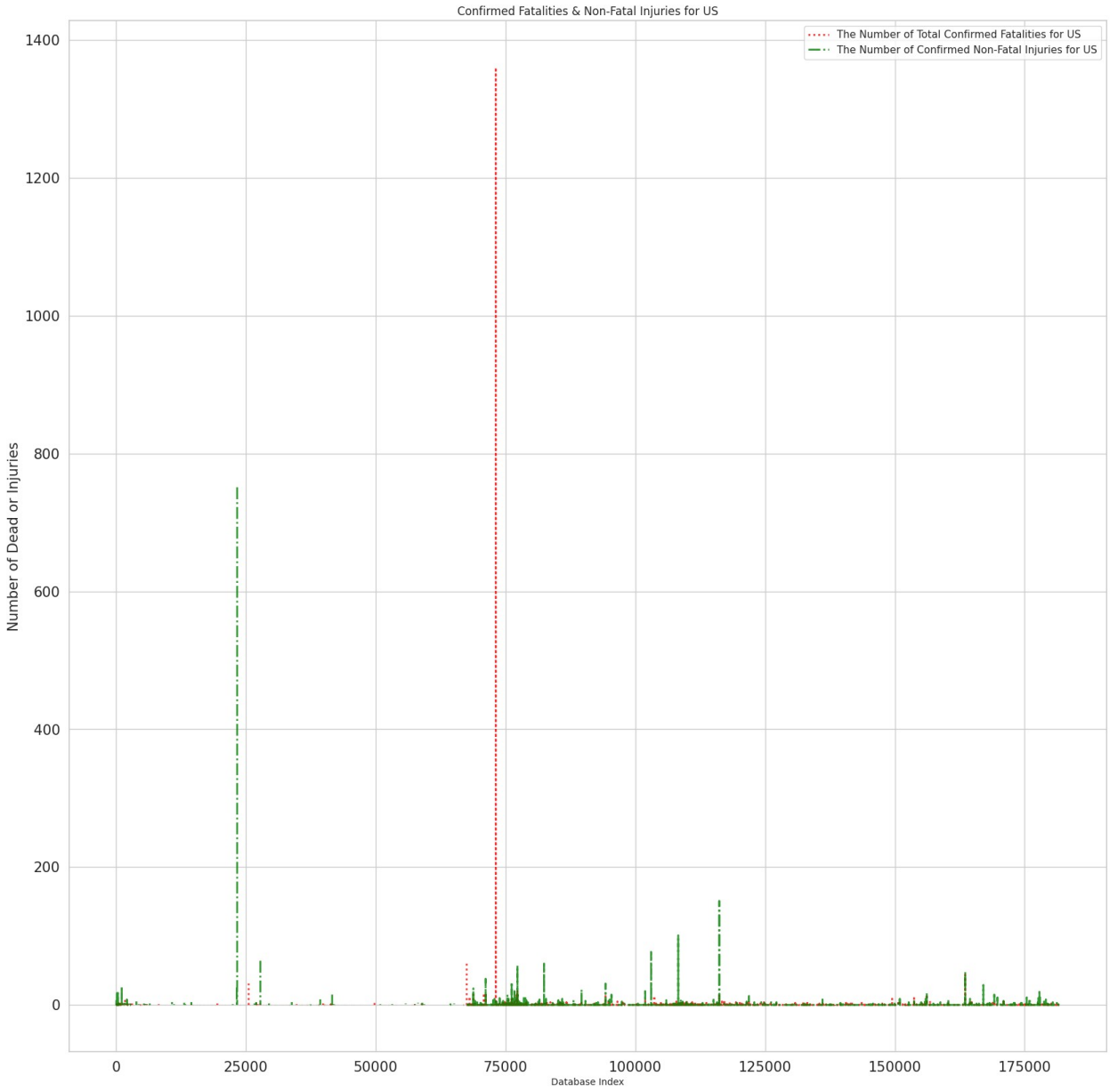
```
df.describe()
```

```
{"type":"dataframe"}
```

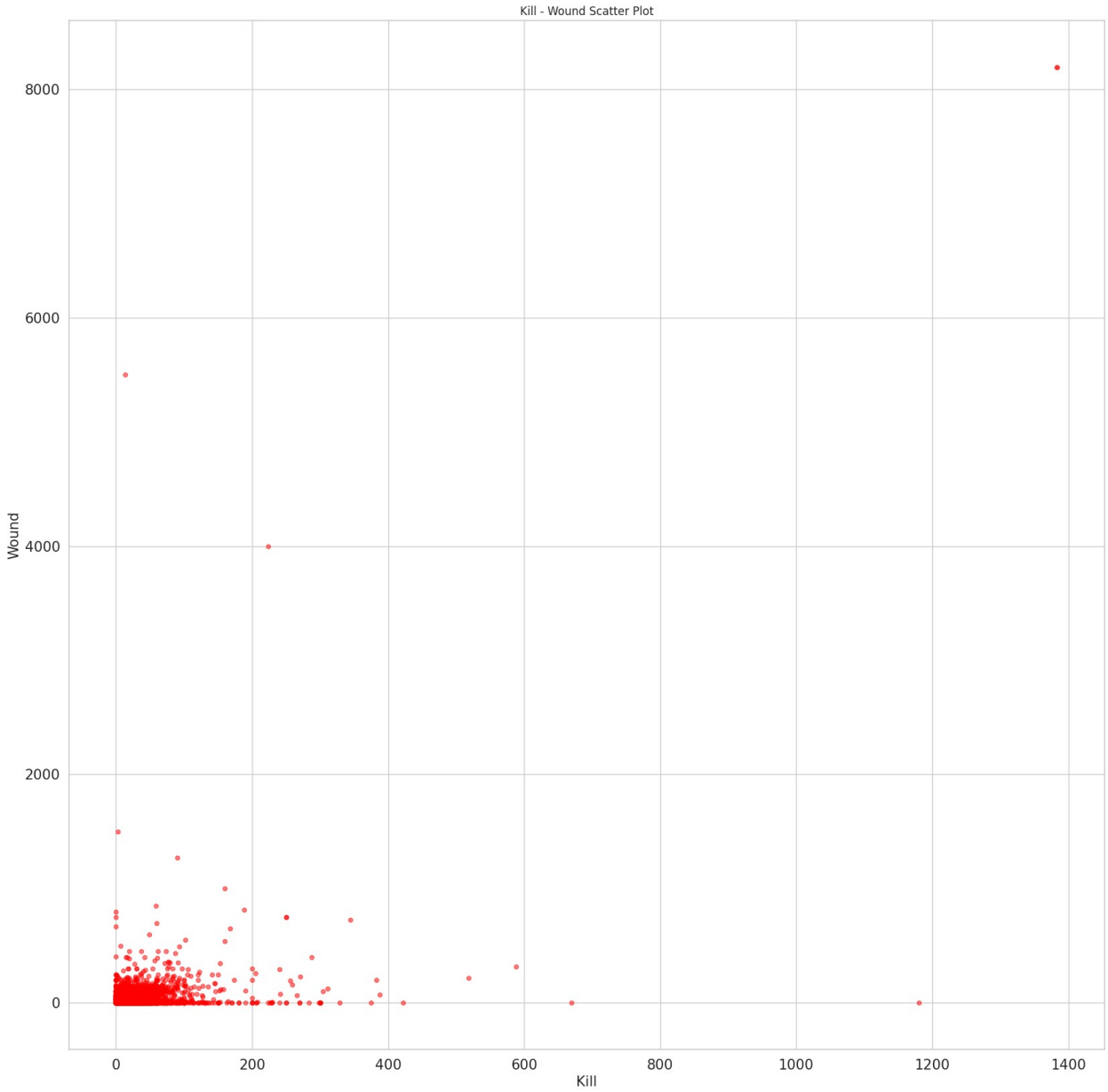
```
df.columns
```

```
Index(['eventid', 'iyear', 'imonth', 'iday', 'approxdate', 'extended',
       'resolution', 'country', 'country_txt', 'region',
       ...
       'addnotes', 'scite1', 'scite2', 'scite3', 'dbsource',
'INT_LOG',
       'INT_IDEO', 'INT_MISC', 'INT_ANY', 'related'],
      dtype='object', length=135)
```

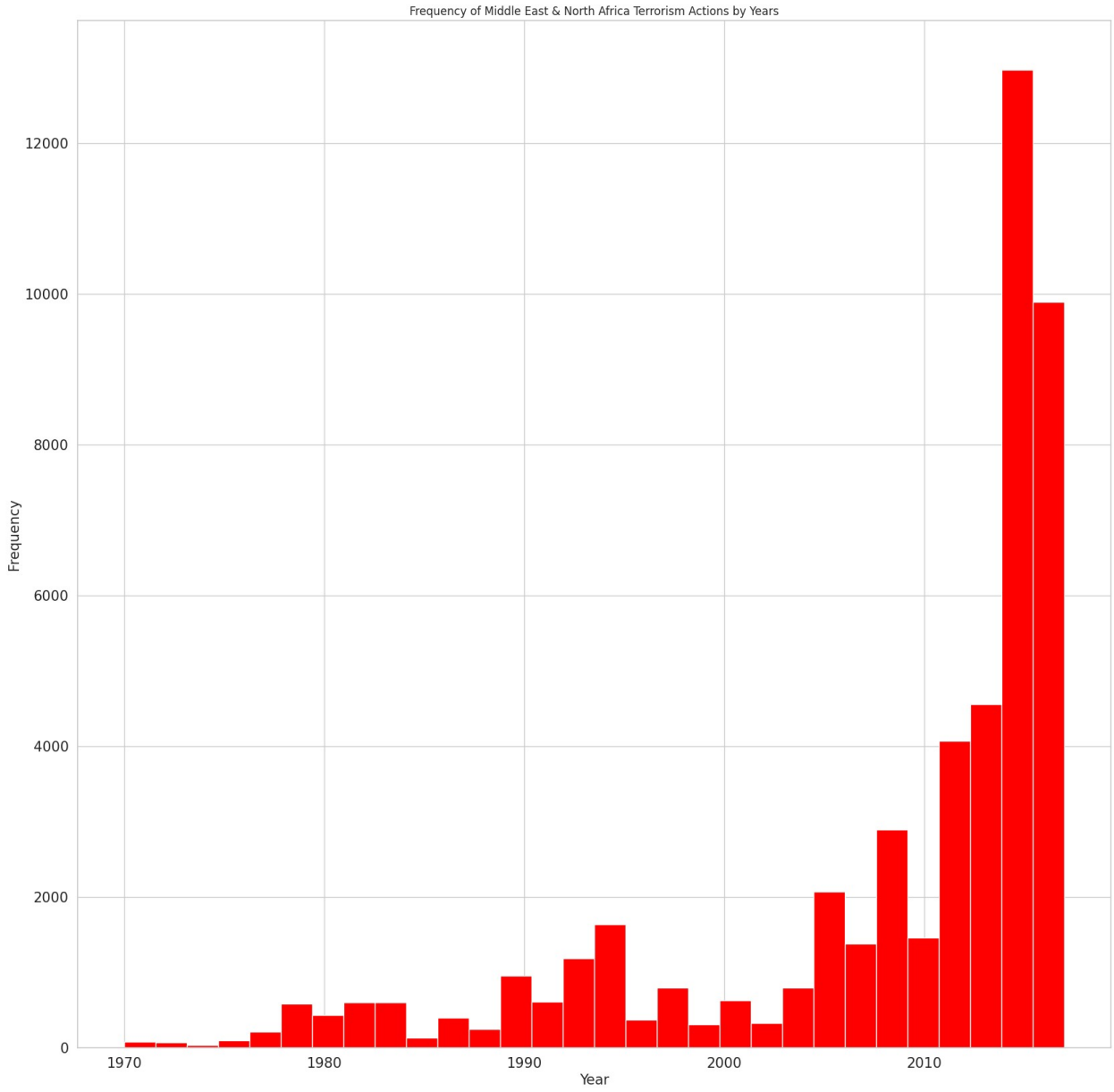
```
df.nkillus.plot(kind = 'line', color = 'red', label = 'The Number of
Total Confirmed Fatalities for US', linewidth = 2, alpha = 0.8, grid =
True,
                linestyle = ':', figsize = (20,20), fontsize=15)
df.nwoundus.plot(color = "green", label = 'The Number of Confirmed
Non-Fatal Injuries for US', linewidth = 2, alpha = 0.8, grid = True,
                linestyle = '-.', figsize = (20,20), fontsize=15)

plt.legend(loc='upper right')     # legend = puts label into plot
plt.xlabel('Database Index', fontsize=10)              # label = name
of label
plt.ylabel('Number of Dead or Injuries', fontsize=15)

plt.title('Confirmed Fatalities & Non-Fatal Injuries for US')
#plot title
plt.show()
```




```
# Death and Injuries at all time
df.plot(kind = 'scatter', x = 'nkill', y = 'nwound', alpha = 0.5,
color = 'red', figsize = (20,20), fontsize=15)
plt.xlabel('Kill', fontsize=15)
plt.ylabel('Wound', fontsize=15)
plt.title('Kill - Wound Scatter Plot')
plt.show()
```

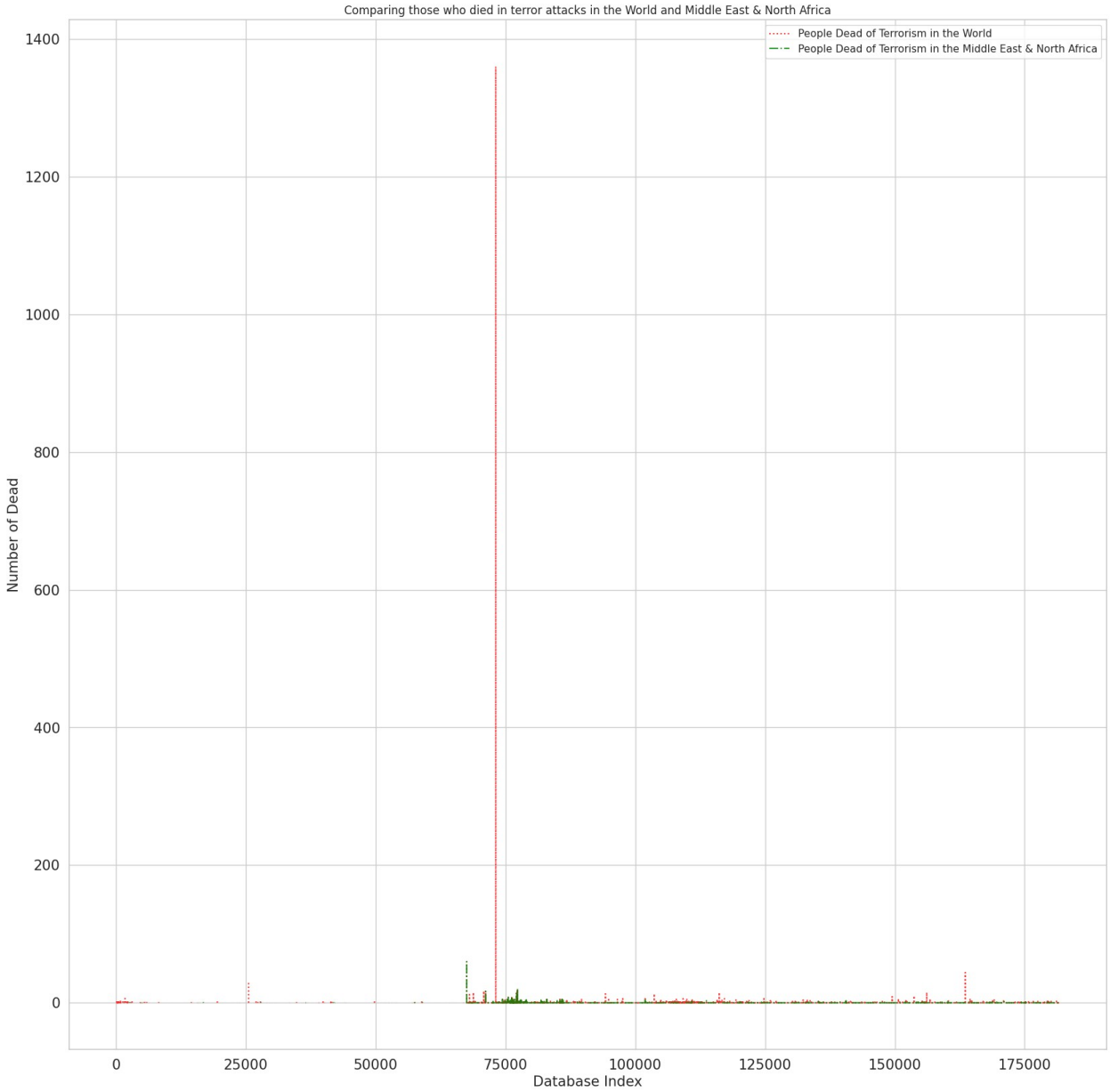



```
# Analysis of middle east and north africa
middleEastData = df[df['region'] == 10]
middleEastData.iyear.plot(kind = 'hist', bins = 30, figsize = (20,20),
color = 'red', fontsize=15)
plt.xlabel('Year', fontsize=15)
plt.ylabel('Frequency', fontsize=15)
plt.title('Frequency of Middle East & North Africa Terrorism Actions
by Years')
plt.show()
```




```
df.nkillus.plot(kind = 'line', color = 'red', label = 'People Dead of
Terrorism in the World', linewidth = 1.5, alpha = 0.8, grid = True,
                 linestyle = ':', figsize = (20,20), fontsize=15)
middleEastData.nkillus.plot(color = "green", label = 'People Dead of
Terrorism in the Middle East & North Africa', linewidth = 1.5, alpha =
0.8,
                            grid = True, linestyle = '-.', figsize =
(20,20), fontsize=15)

plt.legend(loc='upper right')     # legend = puts label into plot
plt.xlabel('Database Index', fontsize=15)              # label = name
of label
plt.ylabel('Number of Dead', fontsize=15)
```

```
plt.title('Comparing those who died in terror attacks in the World and
Middle East & North Africa')            #plot title
plt.show()
```




```
# Terrorist Attacks of a Particular year and their Locations.
filterYear = df['iyear'] == 1970

filterData = df[filterYear] # filter data
# filterData.info()
reqFilterData = filterData.loc[:,'city':'longitude'] #We are getting
the required fields
```

```
reqFilterData = reqFilterData.dropna() # drop NaN values in latitude
and longitude
reqFilterDataList = reqFilterData.values.tolist()
# reqFilterDataList

map = folium.Map(location = [0, 30], tiles='CartoDB positron',
zoom_start=2)
# clustered marker
markerCluster = folium.plugins.MarkerCluster().add_to(map)
for point in range(0, len(reqFilterDataList)):
    folium.Marker(location=[reqFilterDataList[point]
[1],reqFilterDataList[point][2]], popup = reqFilterDataList[point]
[0]).add_to(markerCluster)
map
```

```
<folium.folium.Map at 0x7e338b2e1000>
```

```
killData = df.loc[:,'nkill']
print('Number of people killed by terror attack:',
int(sum(killData.dropna())))
# drop the NaN values
```

```
Number of people killed by terror attack: 411868
```

```
# what types of attacks these deaths were made of
attackData = df.loc[:,'attacktype1':'attacktype1_txt']
# attackData
typeKillData = pd.concat([attackData, killData], axis=1)

typeKillFormatData =
typeKillData.pivot_table(columns='attacktype1_txt', values='nkill',
aggfunc='sum')
typeKillFormatData
```


```
{"summary":"{\n  \"name\": \"typeKillFormatData\",\n  \"rows\": 1,\n
\"fields\": [\n    {\n      \"column\": \"Armed Assault\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
null,\n        \"min\": 160297.0,\n        \"max\": 160297.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          160297.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Assassination\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": null,\n        \"min\": 24920.0,\n
\"max\": 24920.0,\n        \"num_unique_values\": 1,\n
\"samples\": [\n          24920.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Bombing/Explosion\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
null,\n        \"min\": 157321.0,\n        \"max\": 157321.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          157321.0\
n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Facility/Infrastructure Attack\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": null,\n   \"min\":
3642.0,\n   \"max\": 3642.0,\n   \"num_unique_values\": 1,\n
\"samples\": [\n   3642.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Hijacking\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
null,\n   \"min\": 3718.0,\n   \"max\": 3718.0,\n
\"num_unique_values\": 1,\n   \"samples\": [\n   3718.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Hostage Taking (Barricade
Incident)\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": null,\n   \"min\": 4478.0,\n   \"max\":
4478.0,\n   \"num_unique_values\": 1,\n   \"samples\": [\n
4478.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Hostage Taking (Kidnapping)\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": null,\n   \"min\":
24231.0,\n   \"max\": 24231.0,\n   \"num_unique_values\":
1,\n   \"samples\": [\n   24231.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Unarmed Assault\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
null,\n   \"min\": 880.0,\n   \"max\": 880.0,\n
\"num_unique_values\": 1,\n   \"samples\": [\n   880.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Unknown\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
null,\n   \"min\": 32381.0,\n   \"max\": 32381.0,\n
\"num_unique_values\": 1,\n   \"samples\": [\n   32381.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\
n}","type":"dataframe","variable_name":"typeKillFormatData"}

```
typeKillFormatData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1 entries, nkill to nkill
Data columns (total 9 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Armed Assault                         1 non-null      float64
 1   Assassination                         1 non-null      float64
 2   Bombing/Explosion                     1 non-null      float64
 3   Facility/Infrastructure Attack        1 non-null      float64
 4   Hijacking                             1 non-null      float64
 5   Hostage Taking (Barricade Incident)   1 non-null      float64
 6   Hostage Taking (Kidnapping)           1 non-null      float64
 7   Unarmed Assault                       1 non-null      float64
```

```
 8   Unknown                                 1 non-null      float64
dtypes: float64(9)
memory usage: 80.0+ bytes
```

```
# Flatten the values list
flattened_values = [item for sublist in values for item in sublist]

# Create the pie chart using the flattened values
fig, ax = plt.subplots(figsize=(20, 20),
subplot_kw=dict(aspect="equal"))
plt.pie(flattened_values, startangle=90, autopct='%.2f%%')
plt.title('Types of terrorist attacks that cause deaths')
plt.legend(labels, loc='upper right', bbox_to_anchor = (1.3, 0.9),
fontsize=15) # location legend
plt.show()
```



```
countryData = df.loc[:,'country':'country_txt']
# countyData
countryKillData = pd.concat([countryData, killData], axis=1)
```

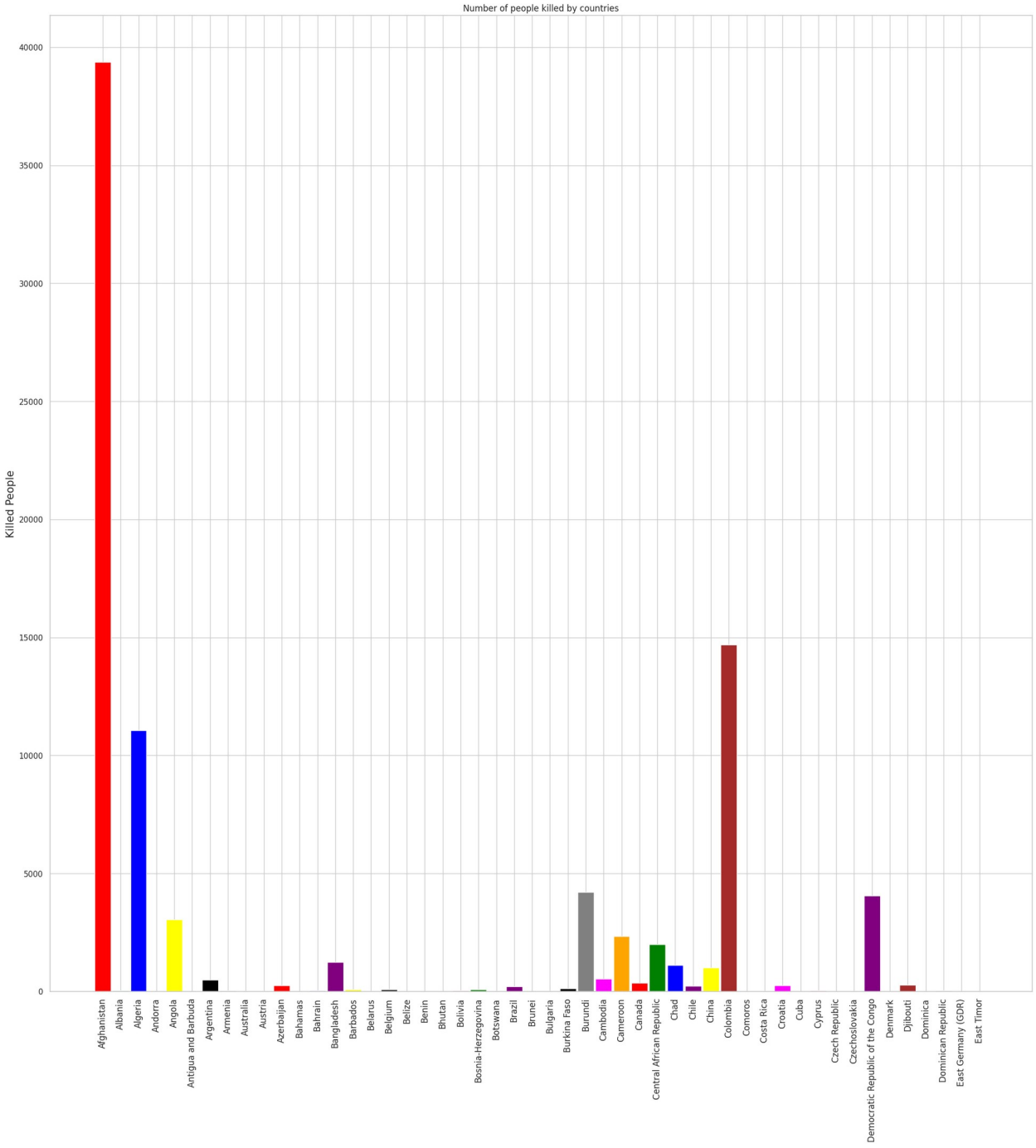
```
countryKillFormatData =
countryKillData.pivot_table(columns='country_txt', values='nkill',
aggfunc='sum')
countryKillFormatData
```

```
{"type":"dataframe","variable_name":"countryKillFormatData"}
```

```
countryKillFormatData.info()
```

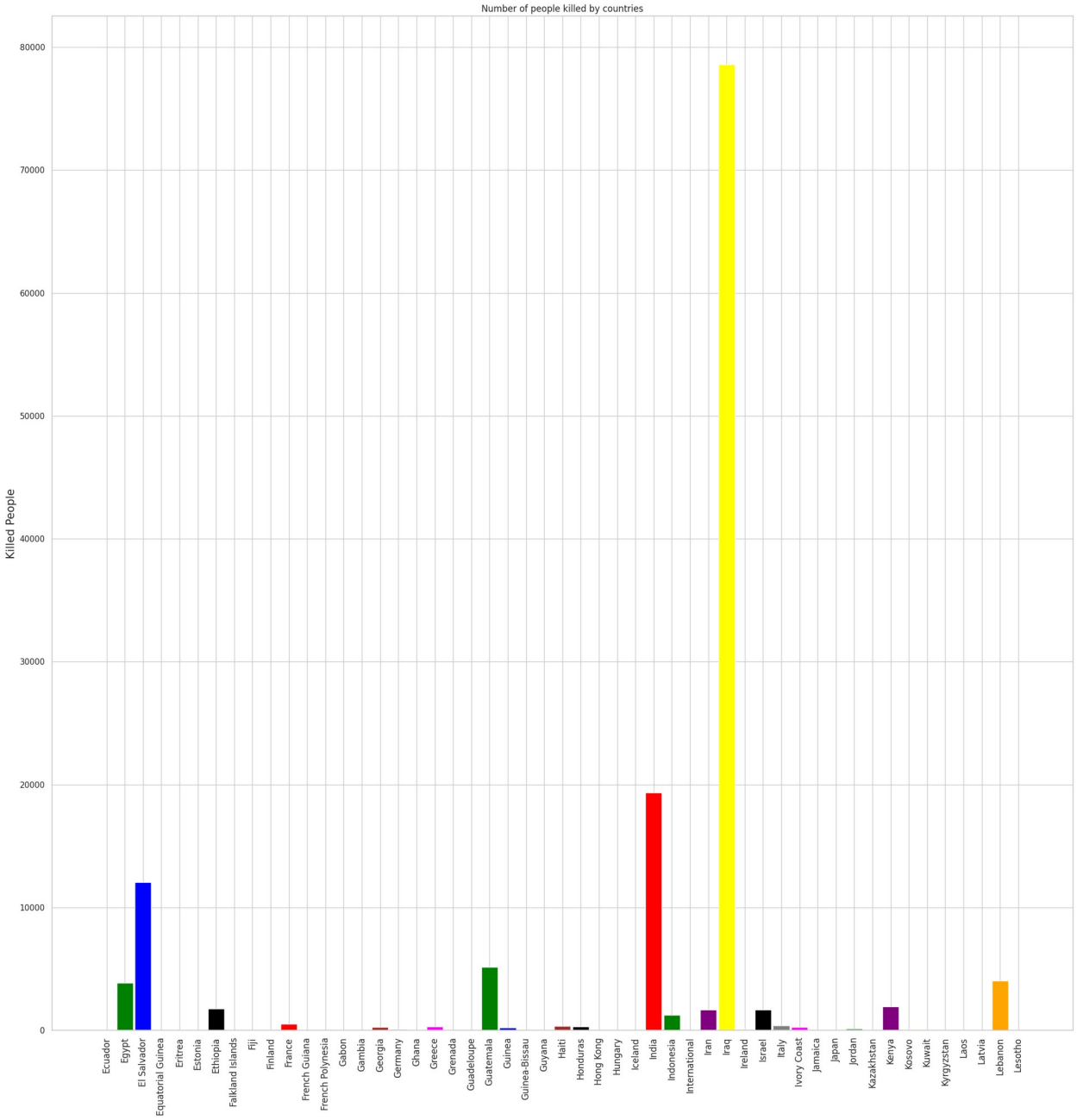
```
<class 'pandas.core.frame.DataFrame'>
Index: 1 entries, nkill to nkill
Columns: 205 entries, Afghanistan to Zimbabwe
dtypes: float64(205)
memory usage: 1.6+ KB
```

```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0]=25
fig_size[1]=25
plt.rcParams["figure.figsize"] = fig_size

labels = countryKillFormatData.columns.tolist()
labels = labels[:50] #50 bar provides nice view
index = np.arange(len(labels))
transpoze = countryKillFormatData.T
values = transpoze.values.tolist()
values = values[:50]
values = [int(i[0]) for i in values] # convert float to int
colors = ['red', 'green', 'blue', 'purple', 'yellow', 'brown',
'black', 'gray', 'magenta', 'orange'] # color list for bar chart bar
color
fig, ax = plt.subplots(1, 1)
ax.yaxis.grid(True)
fig_size = plt.rcParams["figure.figsize"]
fig_size[0]=25
fig_size[1]=25
plt.rcParams["figure.figsize"] = fig_size
plt.bar(index, values, color = colors, width = 0.9)
plt.ylabel('Killed People', fontsize=15)
plt.xticks(index, labels, fontsize=12, rotation=90)
plt.title('Number of people killed by countries')
# print(fig_size)
plt.show()
```



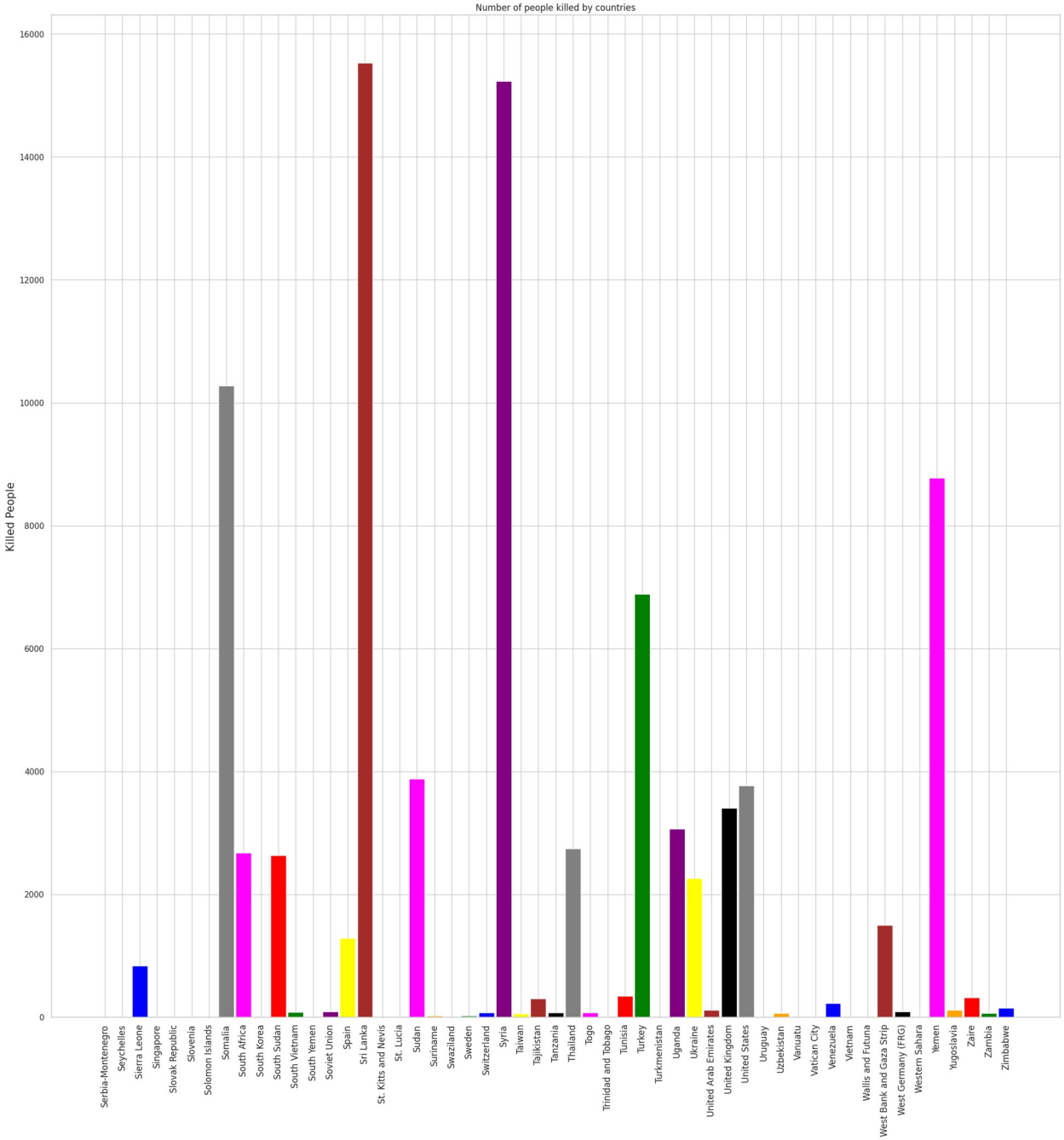

```
labels = countryKillFormatData.columns.tolist()
labels = labels[50:101]
index = np.arange(len(labels))
transpoze = countryKillFormatData.T
values = transpoze.values.tolist()
values = values[50:101]
values = [int(i[0]) for i in values]
colors = ['red', 'green', 'blue', 'purple', 'yellow', 'brown',
```

```
'black', 'gray', 'magenta', 'orange']
fig, ax = plt.subplots(1, 1)
ax.yaxis.grid(True)
fig_size = plt.rcParams["figure.figsize"]
fig_size[0]=25
fig_size[1]=25
plt.rcParams["figure.figsize"] = fig_size
plt.bar(index, values, color = colors, width = 0.9)
plt.ylabel('Killed People', fontsize=15)
plt.xticks(index, labels, fontsize=12, rotation=90)
plt.title('Number of people killed by countries')
plt.show()
```




```
labels = countryKillFormatData.columns.tolist()
labels = labels[152:206]
index = np.arange(len(labels))
transpoze = countryKillFormatData.T
values = transpoze.values.tolist()
values = values[152:206]
values = [int(i[0]) for i in values]
colors = ['red', 'green', 'blue', 'purple', 'yellow', 'brown',
'black', 'gray', 'magenta', 'orange']
fig, ax = plt.subplots(1, 1)
ax.yaxis.grid(True)
```

```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0]=25
fig_size[1]=25
plt.rcParams["figure.figsize"] = fig_size
plt.bar(index, values, color = colors, width = 0.9)
plt.ylabel('Killed People', fontsize=15)
plt.xticks(index, labels, fontsize=12, rotation=90)
plt.title('Number of people killed by countries')
plt.show()
```



```
aaaaaaaaaaa

aaaaaaaaaa
```