Credential's

Name : Mayank Anand

Registration Number : 2141001045

In [160…
```
# Importing the required libraries

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

In [161…
```
# To load the dataset

df=pd.read_csv("Heart Attack Dataset.csv")
```

Exploratory Data Analysis (EDA)

In [162…
```
# To see first 10 columns of the dataset

df.head(10)
```

Out[162]:

	Gender	Age	Blood Pressure (mmHg)	Cholesterol (mg/dL)	Has Diabetes	Smoking Status	Chest Pain Type	Treatment
0	Male	70	181	262	No	Never	Typical Angina	Lifestyle Changes
1	Female	55	103	253	Yes	Never	Atypical Angina	Angioplasty
2	Male	42	95	295	Yes	Current	Typical Angina	Angioplasty
3	Male	84	106	270	No	Never	Atypical Angina	Coronary Artery Bypass Graft (CABG)
4	Male	86	187	296	Yes	Current	Non-anginal Pain	Medication
5	Female	66	125	271	Yes	Former	Typical Angina	Coronary Artery Bypass Graft (CABG)
6	Male	33	181	262	Yes	Current	Asymptomatic	Lifestyle Changes
7	Male	84	182	288	No	Current	Non-anginal Pain	Lifestyle Changes
8	Male	73	115	286	Yes	Never	Asymptomatic	Angioplasty
9	Female	63	174	254	Yes	Former	Non-anginal Pain	Angioplasty

In [163…
```
# To see last 10 columns of the dataset

df.tail(10)
```

Out[163]:

	Gender	Age	Blood Pressure (mmHg)	Cholesterol (mg/dL)	Has Diabetes	Smoking Status	Chest Pain Type	Treatment
990	Female	85	168	208	Yes	Never	Asymptomatic	Lifestyle Changes
991	Female	72	194	181	No	Never	Non-anginal Pain	Lifestyle Changes
992	Female	77	90	276	Yes	Never	Non-anginal Pain	Coronary Artery Bypass Graft (CABG)
993	Female	77	198	268	Yes	Never	Asymptomatic	Medication
994	Female	51	107	217	Yes	Former	Non-anginal Pain	Medication
995	Male	42	125	193	Yes	Current	Typical Angina	Angioplasty
996	Male	80	186	267	Yes	Never	Atypical Angina	Coronary Artery Bypass Graft (CABG)
997	Female	64	108	174	Yes	Current	Non-anginal Pain	Coronary Artery Bypass Graft (CABG)
998	Female	84	123	195	No	Current	Asymptomatic	Lifestyle Changes
999	Male	61	155	197	No	Former	Atypical Angina	Lifestyle Changes

In [164…
```
# To know the shape of the dataset

df.shape
```

Out[164]:
```
(1000, 8)
```

In [165…
```
# To get overall preview about the dataframe

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Gender                 1000 non-null   object
 1   Age                    1000 non-null   int64
 2   Blood Pressure (mmHg)  1000 non-null   int64
 3   Cholesterol (mg/dL)    1000 non-null   int64
 4   Has Diabetes           1000 non-null   object
 5   Smoking Status         1000 non-null   object
 6   Chest Pain Type        1000 non-null   object
 7   Treatment              1000 non-null   object
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

In [166…
```
# To know all the column names of the dataset

df.columns
```

Out[166]:
```
Index(['Gender', 'Age', 'Blood Pressure (mmHg)', 'Cholesterol (mg/dL)',
       'Has Diabetes', 'Smoking Status', 'Chest Pain Type', 'Treatment'],
      dtype='object')
```

In [167…
```
# To generate descriptive statistics of the dataset

df.describe()
```

Out[167]:

	Age	Blood Pressure (mmHg)	Cholesterol (mg/dL)
count	1000.000000	1000.000000	1000.000000
mean	60.338000	145.440000	223.789000
std	17.317496	31.756525	42.787817
min	30.000000	90.000000	150.000000
25%	45.000000	118.000000	185.000000
50%	60.500000	146.000000	225.500000
75%	76.000000	173.000000	259.000000
max	89.000000	199.000000	299.000000

In [168…
```
# To know the datatype of each dataset

df.dtypes
```

Out[168]:
```
Gender                    object
Age                        int64
Blood Pressure (mmHg)      int64
Cholesterol (mg/dL)        int64
Has Diabetes              object
Smoking Status            object
Chest Pain Type           object
Treatment                 object
dtype: object
```

In [169…
```
# To know the how many male and female patients are present

genderCount=df['Gender'].value_counts()
genderCount
```

Out[169]:
```
Female    510
Male      490
Name: Gender, dtype: int64
```

In [170…
```
# To know the count of patients with Diabetes or not

hasDiabetesCount=df['Has Diabetes'].value_counts()
hasDiabetesCount
```

Out[170]:
```
Yes    517
No     483
Name: Has Diabetes, dtype: int64
```

In [171…
```
# To know the count of patients with somoking habit or not

hasSomkingHabitCount=df['Smoking Status'].value_counts()
hasSomkingHabitCount
```

Out[171]:
```
Never      352
Current    325
Former     323
Name: Smoking Status, dtype: int64
```

In [172…
```
# To know the count of type of Chest Pain of patient's

hasChestPainCount=df['Chest Pain Type'].value_counts()
hasChestPainCount
```

Out[172]:
```
Non-anginal Pain    261
Asymptomatic        255
Typical Angina      243
Atypical Angina     241
Name: Chest Pain Type, dtype: int64
```

In [173…
```
# To know count of type of treatment recieved by patient

hasTreatmentCount=df['Treatment'].value_counts()
hasTreatmentCount
```

Out[173]:
```
Lifestyle Changes                      269
Coronary Artery Bypass Graft (CABG)    252
Angioplasty                            247
Medication                             232
Name: Treatment, dtype: int64
```

Data Cleaning and Encoding Categorical Columns

In [174…
```
# to know about how may data's are null in the dataframe

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

Out[174]:
```
Gender                   0
Age                      0
Blood Pressure (mmHg)    0
Cholesterol (mg/dL)      0
Has Diabetes             0
Smoking Status           0
Chest Pain Type          0
Treatment                0
dtype: int64
```

In [175…
```
# Importing label encoder
from sklearn import preprocessing

# Creating label encoder object
label_encoder = preprocessing.LabelEncoder()
```

In [176…
```
df1= df.copy()
```

In [177…
```
# Converting the Gender Categorical column to Numerical

df1['Gender']= label_encoder.fit_transform(df1['Gender'])
```

In [178…
```
df1['Gender'].value_counts()
```

Out[178]:
```
0    510
1    490
Name: Gender, dtype: int64
```

In [179…
```
# Converting the Has Diabetes Categorical column to Numerical

df1['Has Diabetes']= label_encoder.fit_transform(df1['Has Diabetes'])
```

In [180…
```
df1['Has Diabetes'].value_counts()
```

Out[180]:
```
1    517
0    483
Name: Has Diabetes, dtype: int64
```

In [181…
```
# Converting the Smoking Status Categorical column to Numerical

df1['Smoking Status']= label_encoder.fit_transform(df1['Smoking Status'])
```

In [182…
```
df1['Smoking Status'].value_counts()
```

Out[182]:
```
2    352
0    325
1    323
Name: Smoking Status, dtype: int64
```

In [183…
```
# Converting the Chest Paint Type Categorical column to Numerical

df1['Chest Pain Type']= label_encoder.fit_transform(df1['Chest Pain Type'])
```

In [184…
```
df1['Chest Pain Type'].value_counts()
```

Out[184]:
```
2    261
0    255
3    243
1    241
Name: Chest Pain Type, dtype: int64
```

In [185…
```
# Converting the Treatment Categorical column to Numerical

df1['Treatment']= label_encoder.fit_transform(df1['Treatment'])
```

In [186…
```
df1['Treatment'].value_counts()
```

Out[186]:
```
2    269
1    252
0    247
3    232
Name: Treatment, dtype: int64
```

In [187…
```
df1.dtypes
```

Out[187]:
```
Gender                   int32
Age                      int64
Blood Pressure (mmHg)    int64
Cholesterol (mg/dL)      int64
Has Diabetes             int32
Smoking Status           int32
Chest Pain Type          int32
Treatment                int32
dtype: object
```

We can observe form above that all the 5 Categorical Column are transformed into numerical column.

In [188…
```
df1.head(10)
```

Out[188]:

	Gender	Age	Blood Pressure (mmHg)	Cholesterol (mg/dL)	Has Diabetes	Smoking Status	Chest Pain Type	Treatment
0	1	70	181	262	0	2	3	2
1	0	55	103	253	1	2	1	0
2	1	42	95	295	1	0	3	0
3	1	84	106	270	0	2	1	1
4	1	86	187	296	1	0	2	3
5	0	66	125	271	1	1	3	1
6	1	33	181	262	1	0	0	2
7	1	84	182	288	0	0	2	2
8	1	73	115	286	1	2	0	0
9	0	63	174	254	1	1	2	0

Data Normalization

From the above we can observe that if we wish we can perform Normalisation on columns Age, Blood Pressure(mmHg), Cholesterol(mg/dL)

In [189…
```
# Importing requried library for Normalisation

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [190…
```
# Performing Normalisation on Age, Blood Pressure and Cholesterol Column

df2=df1.copy()
df2[['Age', 'Blood Pressure (mmHg)','Cholesterol (mg/dL)']] = scaler.fit_transform(df2[['Age', 'Blood Pressure (mmHg)','Cholesterol (mg/dL)']])
```

In [191…
```
df2.head(10)
```

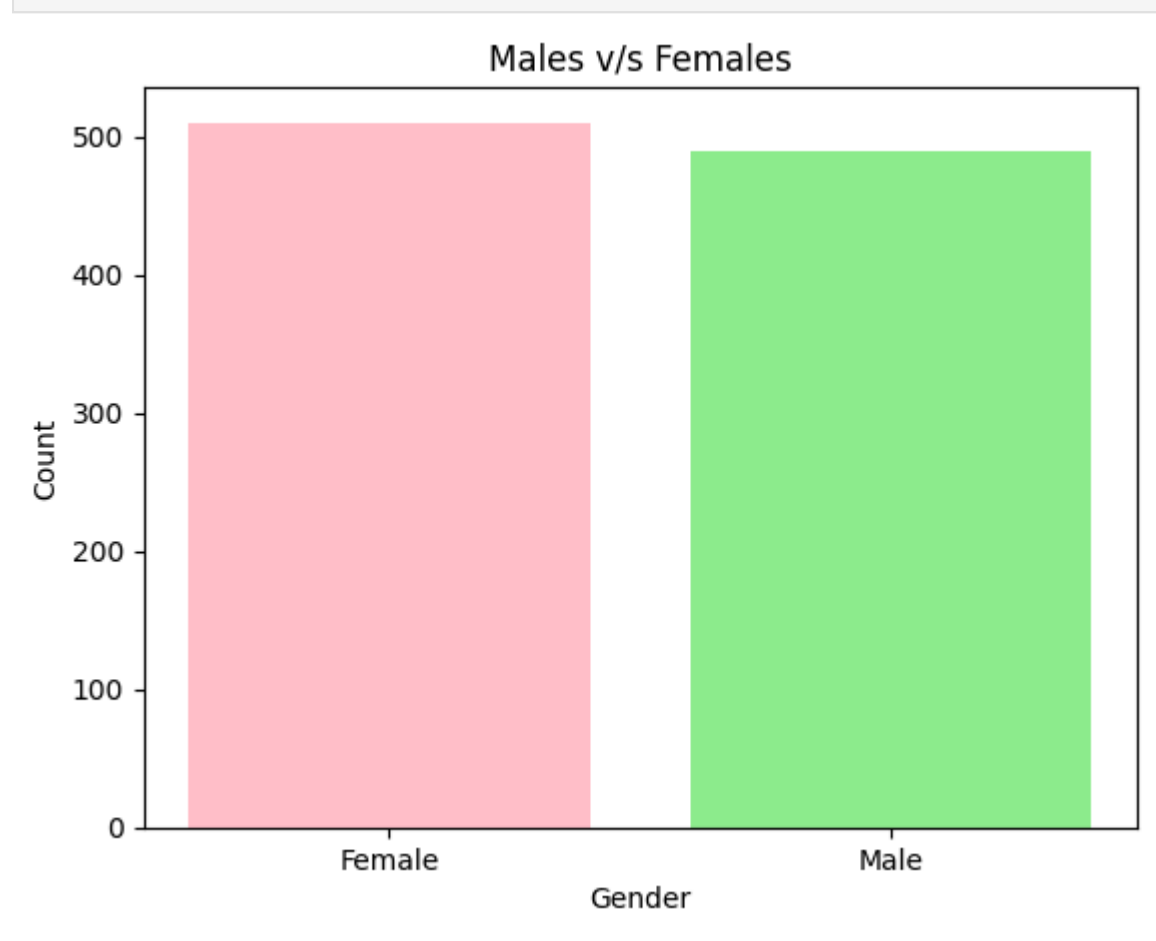
Out[191]:

	Gender	Age	Blood Pressure (mmHg)	Cholesterol (mg/dL)	Has Diabetes	Smoking Status	Chest Pain Type	Treatment
0	1	0.558212	1.120330	0.893481	0	2	3	2
1	0	-0.308397	-1.337087	0.683036	1	2	1	0
2	1	-1.059459	-1.589130	1.665115	1	0	3	0
3	1	1.367048	-1.242571	1.080544	0	2	1	1
4	1	1.482595	1.309362	1.688498	1	0	2	3
5	0	0.327116	-0.643969	1.103927	1	1	3	1
6	1	-1.579425	1.120330	0.893481	1	0	0	2
7	1	1.367048	1.151836	1.501435	0	0	2	2
8	1	0.731534	-0.959023	1.454669	1	2	0	0
9	0	0.153794	0.899793	0.706419	1	1	2	0

Data Visualization

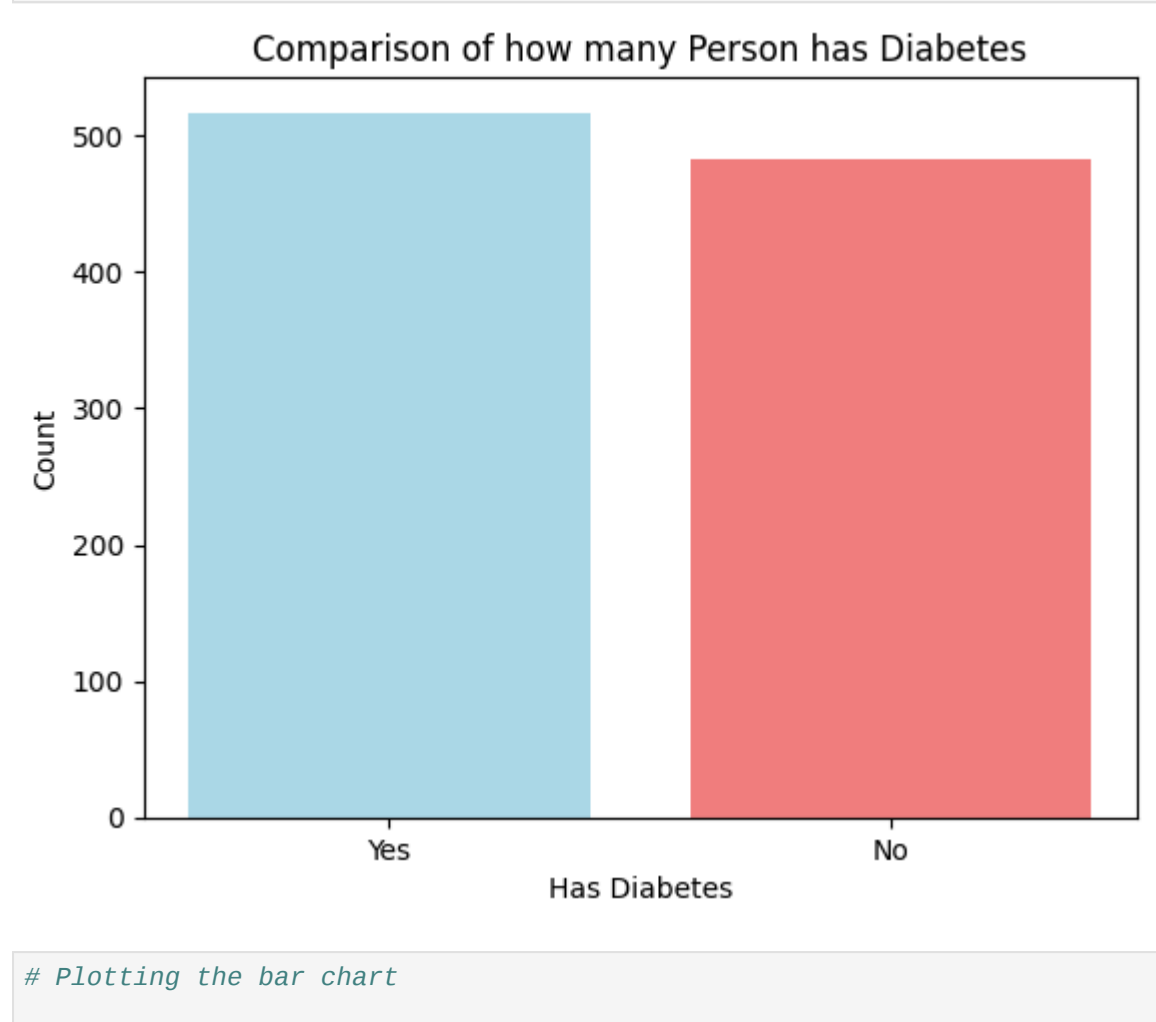
In [192…
```
# Plotting the bar chart

plt.bar(genderCount.index,genderCount.values, color=['pink', 'lightgreen'])
plt.title('Males v/s Females')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```



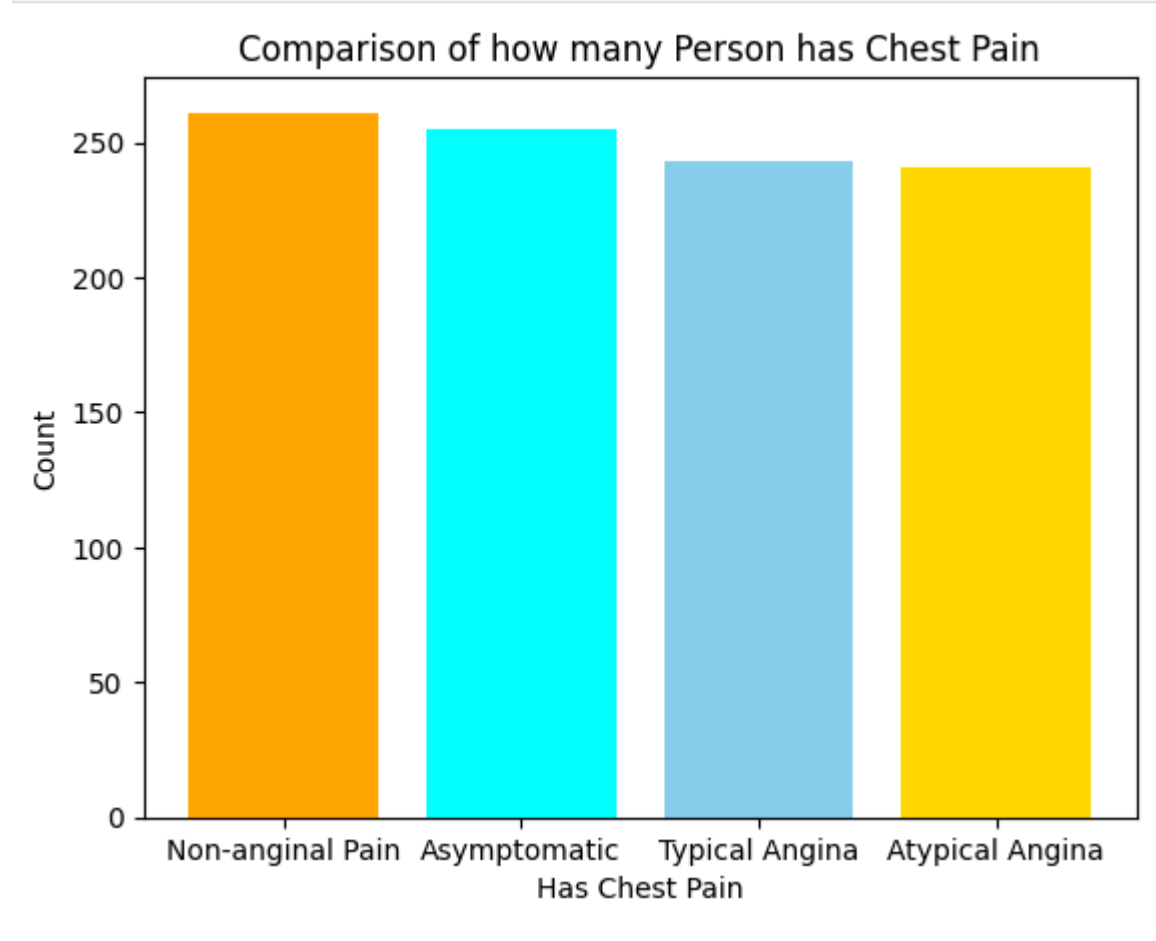
In [193…
```
# Plotting the bar chart

plt.bar(hasDiabetesCount.index,hasDiabetesCount.values, color=['lightblue', 'lightcoral'])
plt.title('Comparison of how many Person has Diabetes')
plt.xlabel('Has Diabetes')
plt.ylabel('Count')
plt.show()
```



In [194…
```
# Plotting the bar chart

plt.bar(hasChestPainCount.index,hasChestPainCount.values, color=['orange', 'cyan','skyblue','gold'])
plt.title('Comparison of how many Person has Chest Pain')
plt.xlabel('Has Chest Pain')
plt.ylabel('Count')
plt.show()
```



In [195…
```
# Plotting the bar chart

plt.bar(hasSomkingHabitCount.index,hasSomkingHabitCount.values, color=['orange', 'cyan','green'])
plt.title('Comparison of how many Person Has Somking Habit')
plt.xlabel('Has Somking Habit')
plt.ylabel('Count')
plt.show()
```