MINOR PROJECT

TASK 1 - Exploratory Data Analysis

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

```
df=pd.read_csv("/content/HepatitisCdata.csv")
df.head(5)
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	1	0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0
1	2	0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74.0	15.6	76.5
2	3	0=Blood Donor	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86.0	33.2	79.3
3	4	0=Blood Donor	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8	75.7
4	5	0=Blood Donor	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9	68.7

<-----------------------Question 1---------------------------->

QUESTION 1

- Are there any missing values in the dataset?
- If so, how will you handle them?
- What is the distribution of different diagnosis categories in the dataset?

```
#Are there any missing values in the dataset?
df.isnull()
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
610	False	False	False	False	False	False	False	False	False	False	False	False	False	False
611	False	False	False	False	False	False	False	False	False	False	False	False	False	False
612	False	False	False	False	False	False	False	False	False	False	False	False	False	False
613	False	False	False	False	False	True	False	False	False	False	False	False	False	False
614	False	False	False	False	False	True	False	False	False	False	False	False	False	False

615 rows × 14 columns

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

```
Unnamed: 0     0
Category       0
Age            0
Sex            0
ALB            1
ALP           18
ALT            1
```

```
AST            0
BIL            0
CHE            0
CHOL          10
CREA           0
GGT            0
PROT           1
dtype: int64
```

therefore, from the above analysis we get the column names and their respective count of null values.

> In order to handle these we can fill in the missing values as "NA"

```
df=df.fillna(value="NA")
```

```
df.isnull()
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
610	False	False	False	False	False	False	False	False	False	False	False	False	False	False
611	False	False	False	False	False	False	False	False	False	False	False	False	False	False
612	False	False	False	False	False	False	False	False	False	False	False	False	False	False
613	False	False	False	False	False	False	False	False	False	False	False	False	False	False
614	False	False	False	False	False	False	False	False	False	False	False	False	False	False

615 rows × 14 columns

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

```
Unnamed: 0    0
Category      0
Age           0
Sex           0
ALB           0
ALP           0
ALT           0
AST           0
BIL           0
CHE           0
CHOL          0
CREA          0
GGT           0
PROT          0
dtype: int64
```

```
#What is the distribution of different diagnosis categories in the dataset?
```

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

```
0=Blood Donor             533
3=Cirrhosis                30
1=Hepatitis                24
2=Fibrosis                 21
0s=suspect Blood Donor      7
Name: Category, dtype: int64
```

Summarizing your analysis and observation

<-----------------------Question 2---------------------------->

Perform feature encoding or transformation on categorical variables (such as sex) in the dataset. Which encoding technique would be most suitable?

```
df.head()
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	1	0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0
1	2	0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.8	74.0	15.6	76.5
2	3	0=Blood Donor	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.2	86.0	33.2	79.3
3	4	0=Blood Donor	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8	75.7
4	5	0=Blood Donor	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9	68.7

```
encoded_data = pd.get_dummies(df, columns=["Sex"])
print(encoded_data.tail(10))
```

```
     Unnamed: 0     Category  Age   ALB    ALP    ALT    AST    BIL   CHE  \
605         606  3=Cirrhosis   42  33.0   79.0    3.7   55.7  200.0  1.72
606         607  3=Cirrhosis   49  33.0  190.7    1.2   36.3    7.0  6.92
607         608  3=Cirrhosis   52  39.0   37.0    1.3   30.4   21.0  6.33
608         609  3=Cirrhosis   58  34.0   46.4   15.0  150.0    8.0  6.26
609         610  3=Cirrhosis   59  39.0   51.3   19.6  285.8   40.0  5.77
610         611  3=Cirrhosis   62  32.0  416.6    5.9  110.3   50.0  5.57
611         612  3=Cirrhosis   64  24.0  102.8    2.9   44.4   20.0  1.54
612         613  3=Cirrhosis   64  29.0   87.3    3.5   99.0   48.0  1.66
613         614  3=Cirrhosis   46  33.0     NA   39.0   62.0   20.0  3.56
614         615  3=Cirrhosis   59  36.0     NA  100.0   80.0   12.0  9.07

     CHOL   CREA    GGT  PROT  Sex_f  Sex_m
605  5.16   89.1  146.3  69.9      1      0
606  3.82  485.9  112.0  58.5      1      0
607  3.78  158.2  142.5  82.7      1      0
608  3.98   56.0   49.7  80.6      1      0
609  4.51  136.1  101.1  70.5      1      0
610   6.3   55.7  650.9  68.5      1      0
611  3.02   63.0   35.9  71.3      1      0
612  3.63   66.7   64.2  82.0      1      0
613   4.2   52.0   50.0  71.0      1      0
614   5.3   67.0   34.0  68.0      1      0
```

```
# Add more cells if required
```

Summarizing your analysis and observations

<-----------------------Question 3----------------------------->

Analyze the distribution and range of values for each clinical and demographic feature in the dataset. Are there any outliers or extreme values?

```
df.head()
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	1	0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0
1	2	0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.8	74.0	15.6	76.5
2	3	0=Blood Donor	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.2	86.0	33.2	79.3
3	4	0=Blood Donor	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8	75.7
4	5	0=Blood Donor	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9	68.7

Summarizing your analysis and observation

```
df.shape
```

```
(615, 14)
```

```
df.describe()
```

	Unnamed: 0	Age	AST	BIL	CHE	CREA	GGT
count	615.000000	615.000000	615.000000	615.000000	615.000000	615.000000	615.000000
mean	308.000000	47.408130	34.786341	11.396748	8.196634	81.287805	39.533171
std	177.679487	10.055105	33.090690	19.673150	2.205657	49.756166	54.661071
min	1.000000	19.000000	10.600000	0.800000	1.420000	8.000000	4.500000
25%	154.500000	39.000000	21.600000	5.300000	6.935000	67.000000	15.700000
50%	308.000000	47.000000	25.900000	7.300000	8.260000	77.000000	23.300000
75%	461.500000	54.000000	32.900000	11.200000	9.590000	88.000000	40.200000
max	615.000000	77.000000	324.000000	254.000000	16.410000	1079.100000	650.900000

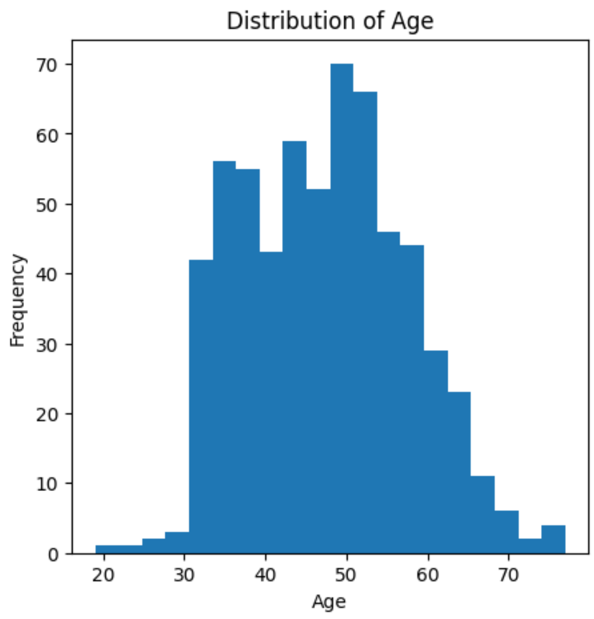
```
df.columns
```

```
Index(['Unnamed: 0', 'Category', 'Age', 'Sex', 'ALB', 'ALP', 'ALT', 'AST',
       'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT'],
      dtype='object')
```

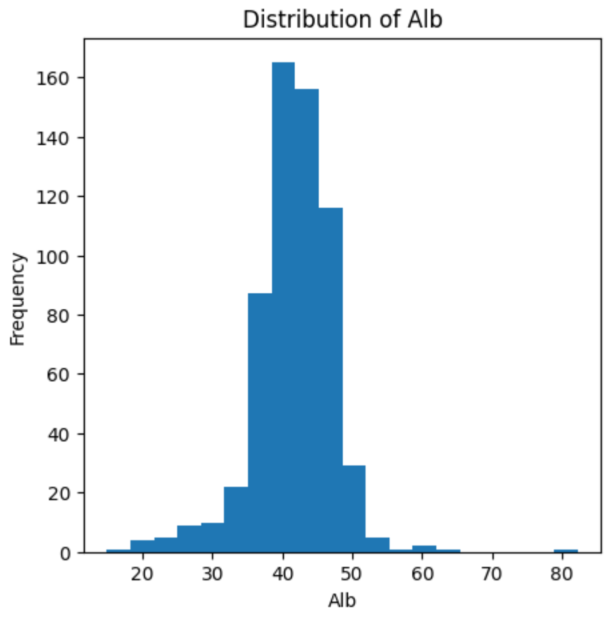
```
features=["Age","ALB","ALP","AST","BIL","CHE","CHOL","CREA","GGT","PROT"]
for feature in features:
  print(feature)
  plt.figure(figsize=(5,5))
  plt.hist(pd.to_numeric(df[feature], errors="coerce"), bins=20)
  plt.xlabel(feature.capitalize())
  plt.ylabel("Frequency")
  plt.title(f"Distribution of {feature.capitalize()}")
  plt.show()

  min=df[feature].min
  max=df[feature].max
  min_value = pd.to_numeric(df[feature], errors="coerce").min()
  max_value = pd.to_numeric(df[feature], errors="coerce").max()
  print(f"Range of values for {feature.capitalize()}: {min_value} - {max_value}")
```

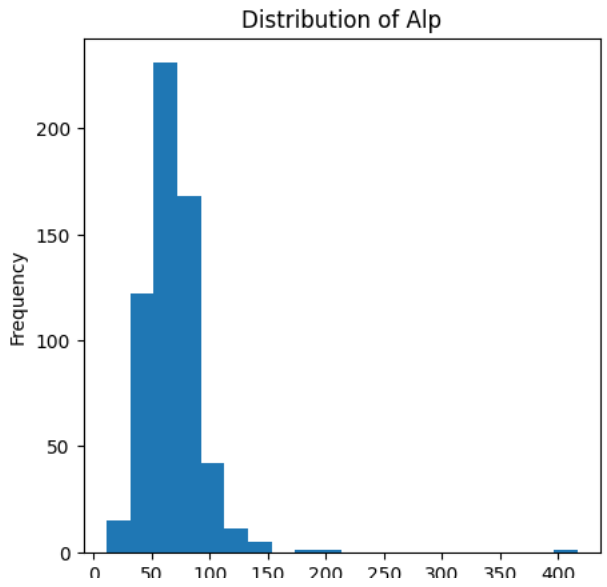
```
Age
```

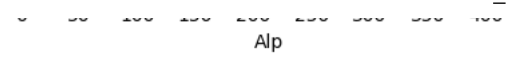


```
Range of values for Age: 19 - 77
ALB
```

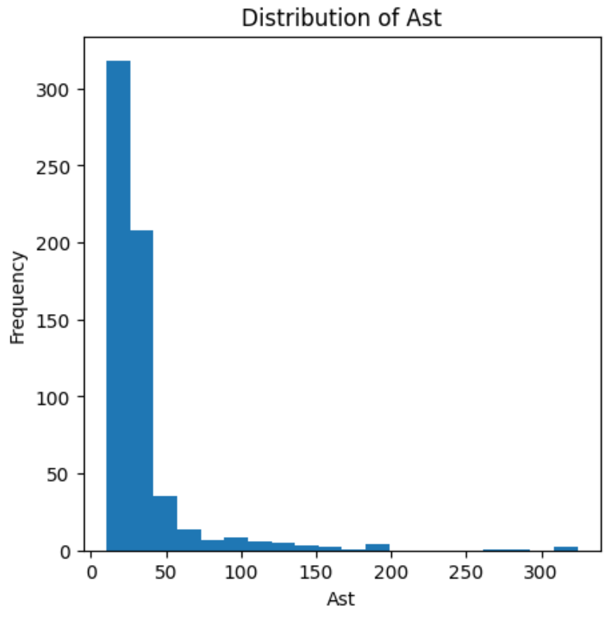


```
Range of values for Alb: 14.9 - 82.2
ALP
```

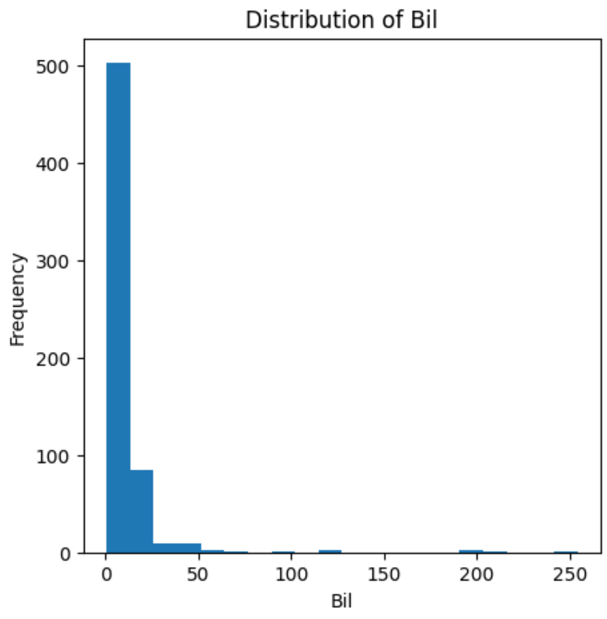




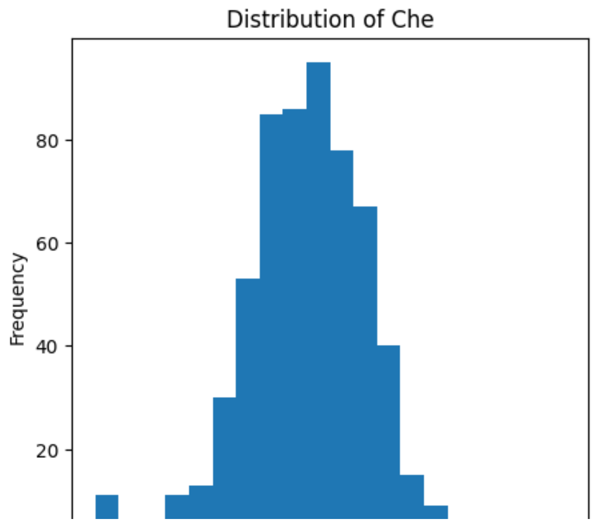
```
Range of values for Alp: 11.3 - 416.6
AST
```

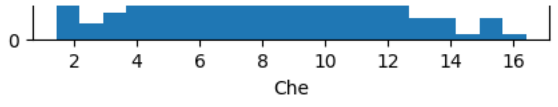


```
Range of values for Ast: 10.6 - 324.0
BIL
```

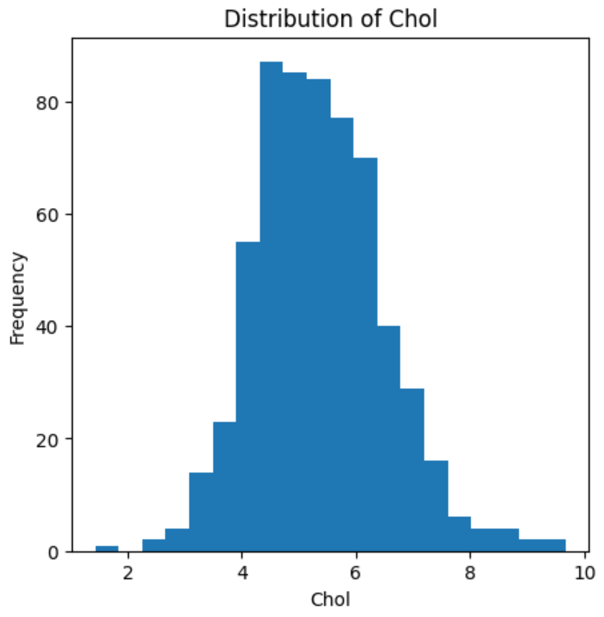


```
Range of values for Bil: 0.8 - 254.0
CHE
```

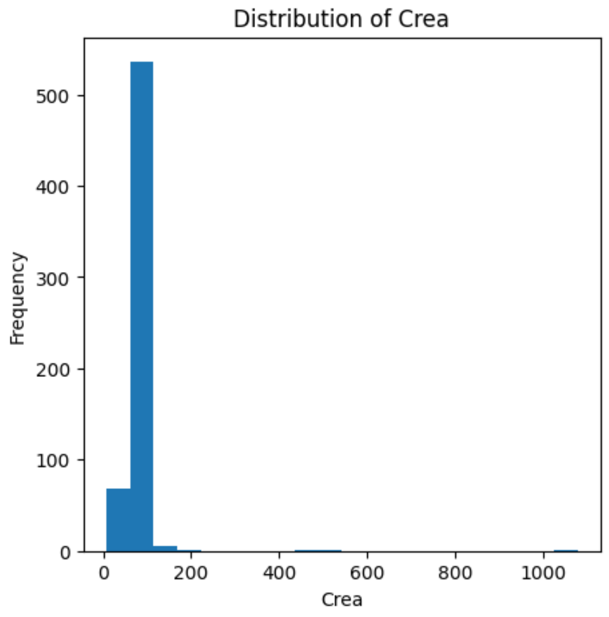




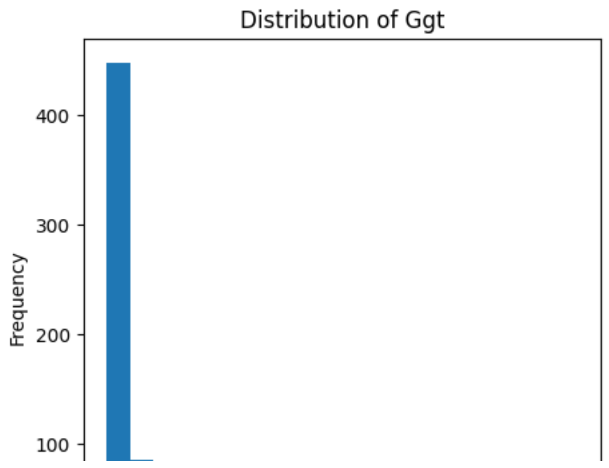
```
Range of values for Che: 1.42 - 16.41
CHOL
```

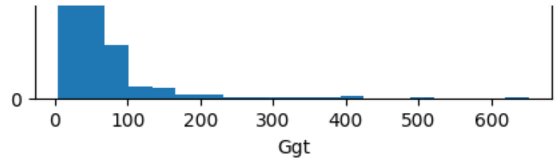


```
Range of values for Chol: 1.43 - 9.67
CREA
```

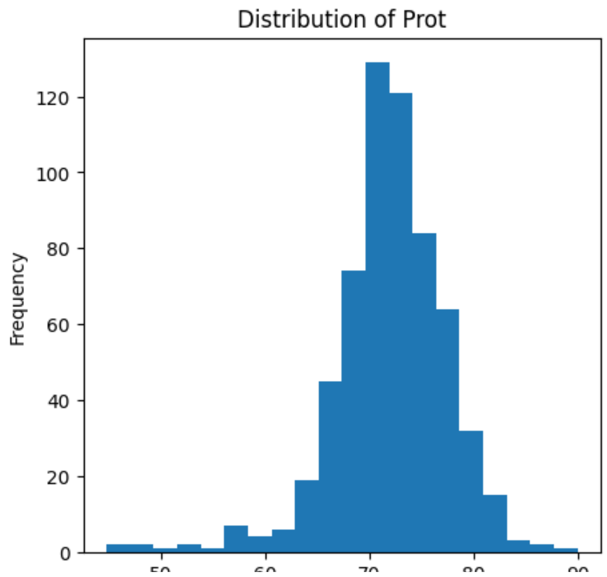


```
Range of values for Crea: 8.0 - 1079.1
GGT
```





```
Range of values for Ggt: 4.5 - 650.9
PROT
```



```
from scipy import stats

# Calculate z-scores for the "price" variable
z_scores = stats.zscore(pd.to_numeric(df["GGT"],errors="coerce"))

# Set a threshold for outliers (e.g., z-score > 3 or < -3)
threshold = 3

# Identify the outliers
outliers = df[abs(z_scores) > threshold]

# Remove the outliers from the dataset
clean_data = df[abs(z_scores) <= threshold]

# Display the outliers and cleaned dataset
print("Outliers:")
print(outliers)
print("\nCleaned Dataset:")
print(clean_data.head())
```

```
Outliers:
     Unnamed: 0               Category  Age Sex   ALB    ALP    ALT    AST  \
205         206          0=Blood Donor   50   m  42.2  145.0   27.5   37.9
533         534  0s=suspect Blood Donor   47   m  22.5  124.0   79.5   46.7
538         539  0s=suspect Blood Donor   74   m  20.3   84.0   22.8   43.0
539         540  0s=suspect Blood Donor   59   f  19.3  208.2  325.3  146.6
558         559            1=Hepatitis   56   m  37.0  114.0   27.8  324.0
559         560            1=Hepatitis   58   m  43.0   99.1   12.2   63.2
593         594            3=Cirrhosis   51   m  39.0   66.0   29.6  185.0
598         599            3=Cirrhosis   58   m  31.0  143.1    7.0  181.8
602         603            3=Cirrhosis   61   m  39.0  102.9   27.3  143.2
610         611            3=Cirrhosis   62   f  32.0  416.6    5.9  110.3

      BIL    CHE  CHOL   CREA    GGT  PROT
205   4.5  13.71   8.8  103.0  239.0  73.1
533   2.3   6.83   4.3  170.0  345.6  58.6
538   5.7   4.91  3.19   52.0  218.3  47.8
539   6.9   5.33  4.72   32.0  295.6  53.1
558  67.0   5.75  3.09   97.7  392.2  77.3
559  13.0   5.95  6.15  147.3  491.0  65.6
593  19.0   2.00   3.6   58.3  399.5  79.4
598  58.0   3.29  3.92   66.4  273.7  78.1
602  15.0   5.38  4.88   72.3  400.3  73.4
610  50.0   5.57   6.3   55.7  650.9  68.5

Cleaned Dataset:
   Unnamed: 0       Category  Age Sex   ALB   ALP   ALT   AST   BIL    CHE  \
0          1  0=Blood Donor   32   m  38.5  52.5   7.7  22.1   7.5   6.93
1          2  0=Blood Donor   32   m  38.5  70.3  18.0  24.7   3.9  11.17
2          3  0=Blood Donor   32   m  46.9  74.7  36.2  52.6   6.1   8.84
3          4  0=Blood Donor   32   m  43.2  52.0  30.6  22.6  18.9   7.33
4          5  0=Blood Donor   32   m  39.2  74.1  32.6  24.8   9.6   9.15

   CHOL   CREA   GGT  PROT
0  3.23  106.0  12.1  69.0
1   4.8   74.0  15.6  76.5
2   5.2   86.0  33.2  79.3
3  4.74   80.0  33.8  75.7
4  4.32   76.0  29.9  68.7
```

<-----------------------Question 4---------------------------->

```
#--------------code
```

```
# Add more cells if required
```

Summarizing your analysis and observations

<-----------------------Question 5---------------------------->

Can you explore the correlations between features and the diagnosis of hepatitis C? Are there any strong correlations or dependencies?

```
df.info
```

```
<bound method DataFrame.info of      Unnamed: 0       Category  Age Sex   ALB    ALP    ALT    AST   BIL  \
0             1  0=Blood Donor   32   m  38.5   52.5    7.7   22.1   7.5
1             2  0=Blood Donor   32   m  38.5   70.3   18.0   24.7   3.9
2             3  0=Blood Donor   32   m  46.9   74.7   36.2   52.6   6.1
3             4  0=Blood Donor   32   m  43.2   52.0   30.6   22.6  18.9
4             5  0=Blood Donor   32   m  39.2   74.1   32.6   24.8   9.6
..          ...            ...  ...  ..   ...    ...    ...    ...   ...
610         611    3=Cirrhosis   62   f  32.0  416.6    5.9  110.3  50.0
611         612    3=Cirrhosis   64   f  24.0  102.8    2.9   44.4  20.0
612         613    3=Cirrhosis   64   f  29.0   87.3    3.5   99.0  48.0
613         614    3=Cirrhosis   46   f  33.0     NA   39.0   62.0  20.0
614         615    3=Cirrhosis   59   f  36.0     NA  100.0   80.0  12.0

       CHE  CHOL   CREA    GGT  PROT
0     6.93  3.23  106.0   12.1  69.0
1    11.17   4.8   74.0   15.6  76.5
2     8.84   5.2   86.0   33.2  79.3
3     7.33  4.74   80.0   33.8  75.7
4     9.15  4.32   76.0   29.9  68.7
..     ...   ...    ...    ...   ...
610   5.57   6.3   55.7  650.9  68.5
611   1.54  3.02   63.0   35.9  71.3
612   1.66  3.63   66.7   64.2  82.0
613   3.56   4.2   52.0   50.0  71.0
614   9.07   5.3   67.0   34.0  68.0

[615 rows x 14 columns]>
```

```
df.head()
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	1	0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0
1	2	0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.8	74.0	15.6	76.5
2	3	0=Blood Donor	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.2	86.0	33.2	79.3
3	4	0=Blood Donor	32	m	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8	75.7
4	5	0=Blood Donor	32	m	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9	68.7

```
selected_columns = ["Age","ALB", "ALP", "ALT", "AST","BIL","CHE","CHOL","CREA","GGT","PROT"]
subset_df = df[selected_columns]
subset_df.head()
```

	Age	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	32	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0
1	32	38.5	70.3	18.0	24.7	3.9	11.17	4.8	74.0	15.6	76.5
2	32	46.9	74.7	36.2	52.6	6.1	8.84	5.2	86.0	33.2	79.3
3	32	43.2	52.0	30.6	22.6	18.9	7.33	4.74	80.0	33.8	75.7
4	32	39.2	74.1	32.6	24.8	9.6	9.15	4.32	76.0	29.9	68.7

```
correlation_matrix = subset_df.corr()
correlation_matrix
```

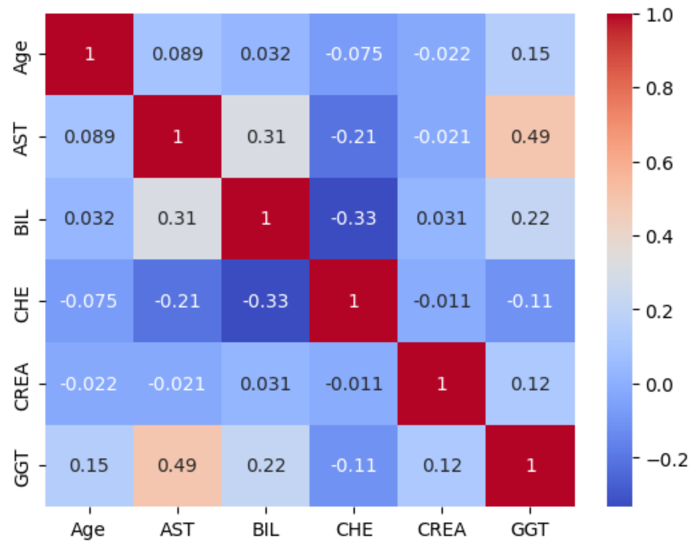
```
<ipython-input-33-bda98bf6fb1c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr i
  correlation_matrix = subset_df.corr()
```

	Age	AST	BIL	CHE	CREA	GGT
Age	1.000000	0.088666	0.032492	-0.075093	-0.022296	0.153087
AST	0.088666	1.000000	0.312231	-0.208536	-0.021387	0.491263
BIL	0.032492	0.312231	1.000000	-0.333172	0.031224	0.217024
CHE	-0.075093	-0.208536	-0.333172	1.000000	-0.011157	-0.110345
CREA	-0.022296	-0.021387	0.031224	-0.011157	1.000000	0.121003
GGT	0.153087	0.491263	0.217024	-0.110345	0.121003	1.000000

```
import seaborn as sns

# Create a heatmap of the correlation matrix
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
```

```
<Axes: >
```



Summarizing your analysis and observation

TASK 2 - Classification/Regression

Perform following steps on the same dataset which you used for EDA.

> - Data Preprocessing (as per requirement)
> - Feature Engineering
> - Split dataset in train-test (80:20 ratio)
> - Model selection
> - Model training
> - Model evaluation
> - Fine-tune the Model
> - Make predictions

Summarize your model's performance by evaluation metrices

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

```
data=pd.read_csv("/content/HepatitisCdata.csv")
data.head(5)
```

	Unnamed: 0	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	1	0=Blood Donor	32	m	38.5	52.5	7.7	22.1	7.5	6.93	3.23	106.0	12.1	69.0
1	2	0=Blood Donor	32	m	38.5	70.3	18.0	24.7	3.9	11.17	4.80	74.0	15.6	76.5
2	3	0=Blood	32	m	46.9	74.7	36.2	52.6	6.1	8.84	5.20	86.0	33.2	79.3

```
data.isnull().sum()
```

```
Unnamed: 0     0
Category       0
Age            0
Sex            0
ALB            1
ALP           18
ALT            1
AST            0
BIL            0
CHE            0
CHOL          10
CREA           0
GGT            0
PROT           1
dtype: int64
```

```
from sklearn.model_selection import train_test_split

# Separate the target variable from the features
target = data['Category']
features = data.drop('Category', axis=1)

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

```
from sklearn.ensemble import RandomForestClassifier

# Create the Random Forest Classifier
```

```
model = RandomForestClassifier()
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
import pandas as pd

# Load the dataset
data=pd.read_csv("/content/HepatitisCdata.csv")

# Convert 'Sex' column to numerical values
data = pd.get_dummies(data, columns=['Sex'], drop_first=True)

# Separate the target variable from the features
target = data['Category']
features = data.drop('Category', axis=1)

# Perform missing value imputation
imputer = SimpleImputer(strategy='mean')
imputed_features = pd.DataFrame(imputer.fit_transform(features), columns=features.columns)

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(imputed_features, target, test_size=0.2, random_state=42)

# Create the Random Forest Classifier
model = RandomForestClassifier()

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
report = classification_report(y_test, y_pred)
print(report)
```

```
                        precision    recall  f1-score   support

         0=Blood Donor       0.99      1.00      0.99        96
0s=suspect Blood Donor       1.00      0.33      0.50         3
           1=Hepatitis       0.80      0.89      0.84         9
            2=Fibrosis       0.62      0.83      0.71         6
           3=Cirrhosis       1.00      0.78      0.88         9

              accuracy                           0.95       123
             macro avg       0.88      0.77      0.79       123
          weighted avg       0.96      0.95      0.95       123
```

```
from sklearn.metrics import classification_report

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
report = classification_report(y_test, y_pred)
print(report)
```

```
                        precision    recall  f1-score   support

         0=Blood Donor       0.99      1.00      0.99        96
0s=suspect Blood Donor       1.00      0.33      0.50         3
           1=Hepatitis       0.80      0.89      0.84         9
            2=Fibrosis       0.62      0.83      0.71         6
           3=Cirrhosis       1.00      0.78      0.88         9

              accuracy                           0.95       123
             macro avg       0.88      0.77      0.79       123
          weighted avg       0.96      0.95      0.95       123
```

```
from sklearn.model_selection import GridSearchCV

# Define the hyperparameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10]
}

# Create the GridSearchCV object
grid_search = GridSearchCV(model, param_grid, cv=5)

# Perform grid search to find the best hyperparameters
grid_search.fit(X_train, y_train)

# Get the best model with tuned hyperparameters
best_model = grid_search.best_estimator_
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_split.py:700: UserWarning: The least populated class in y has only 4 me
  warnings.warn(
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Display the performance summary
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
```

```
Accuracy: 0.9512195121951219
Precision: 0.9590269047020368
Recall: 0.9512195121951219
F1-score: 0.9491240026643128
```

✓ 0s completed at 11:45 PM