#PROGRAM 1

#Develop a program to create histograms for all numerical features and analyze the distribution of each feature.

#Generate box plots for all numerical features and identify any outliers. Use California Housing dataset.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing(as_frame = True)
housing_df = data.frame
numerical_feature= housing_df.select_dtypes(include=[np.number]).columns
plt.figure(figsize=(15,10))

for i,feature in enumerate(numerical_feature):
  plt.subplot(3,3,i+1)
  sns.histplot(housing_df[feature],kde=True,bins=30,color='blue')
  plt.title(f'Distribution of {feature}')

plt.tight_layout()
plt.show()
plt.figure(figsize=(15,10))

for i,feature in enumerate(numerical_feature):
  plt.subplot(3,3,i+1)
  sns.boxplot(x=housing_df[feature],color='orange')
  plt.title(f'Box Plot of {feature}')
plt.tight_layout()
plt.show()
print("Outliers Detection: ")


outliers_summary={}
for feature in numerical_feature:
    Q1=housing_df[feature].quantile(0.25)
    Q3=housing_df[feature].quantile(0.75)
    IQR=Q3-Q1
    lower_bound=Q1-1.5*IQR
    upper_bound=Q3+1.5*IQR
    outliers=housing_df[(housing_df[feature]<lower_bound)|(housing_df[feature]>upper_bound)]
    outliers_summary[feature]=len(outliers)
    print(f"{feature}:{len(outliers)}outliers")


print("\n Dataset Summary")
print(housing_df.describe())
```

#PROGRAM 2

#Develop a program to Compute the correlation matrix to understand the relationships between pairs of

#features. Visualize the correlation matrix using a heatmap to know which variables have strong

#positive/negative correlations. Create a pair plot to visualize pairwise relationships between features. Use

#California Housing dataset.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing

california_data = fetch_california_housing(as_frame=True)
data = california_data.frame
correlation_matrix=data.corr()
plt.figure(figsize=(10,8))
sns.heatmap(correlation_matrix,annot=True,cmap='coolwarm',fmt='2f',linewidths=0.5)
plt.title('Correlation Matrix of California Housing Features')
plt.show()
sns.pairplot(data,diag_kind='kde',plot_kws={'alpha':0.5})
plt.suptitle('Pairplot of California Housing Features',y=1.02)
plt.show()
```

#Program 3

#Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the

#Iris dataset from 4 features to 2.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA

iris=load_iris()
data=iris.data
labels=iris.target
label_names=iris.target_names
label_names
iris_df=pd.DataFrame(data,columns=iris.feature_names)
iris_df
pca=PCA(n_components=2)
data_reduced=pca.fit_transform(data)
data_reduced

reduced_df=pd.DataFrame(data_reduced,columns=['Principal Component 1','Principal Component 2'])
reduced_df['label']=labels
reduced_df

plt.figure(figsize=(8,6))
colors=['r','g','b']
for i,label in enumerate(np.unique(labels)):
  plt.scatter(reduced_df[reduced_df['label']==label]['Principal Component 1'],
        reduced_df[reduced_df['label']==label]['Principal Component 2'],
        color=colors[i],label=label_names[label])

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA of Iris Dataset')
plt.legend()
plt.grid()
plt.show()
```

#Program 4

#For a given set of training data examples stored in a .CSV file, implement and demonstrate the Find-S

#algorithm to output a description of the set of all hypotheses consistent with the training examples.

```
import pandas as pd
def find_s_algorithm(file_path):
 data=pd.read_csv(file_path)
 print("Training data:")
 print(data)
 attributes=data.columns[:-1]
 class_label=data.columns[-1]
 hypothesis= None
 for index,row in data.iterrows():
  if row[class_label]=='Yes':
    if hypothesis is None:
     hypothesis = list(row[attributes])
    else:
     for i,value in enumerate(row[attributes]):
      if hypothesis[i]!=value:
        hypothesis[i]='?'
  return hypothesis
file_path=r'covid.csv'
hypothesis=find_s_algorithm(file_path)
print("\n The final hypothesis is:",hypothesis)
```