SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITL503 **SAP ID: 60003220045**

COURSE NAME: Data Warehousing and Mining **CLASS: T Y B. TECH**

NAME : Anish Sharma

LAB EXPERIMENT NO. 5

CO/LO: Interpret data for a given dataset.

Aim: To perform data exploration and preprocessing on selected data set.

Instructions:

Select any data set. (Kaggle, UCI Machine Learning Repository, Google data sets)

Exercise 1:

a) List all the attributes.
b) Identify the type of each attribute i.e. nominal, binary (symmetric or asymmetric), ordinal, numeric (interval scaled or ratio scaled).

```
[ ] """
    (a)
    Attributes:
    Rank: Numeric (Ordinal)
    Previous Rank: Numeric (Ordinal) with missing values
    Company: Nominal (Text)
    Location: Nominal (Text)
    Industry: Nominal (Text)
    Revenue: Nominal (Categorical, formatted as currency)
    Profit %: Numeric (Ratio, formatted as a percentage)
    CEO Pay Ratio: Ordinal (Categorical, formatted as ratio)
    Women on Board %: Numeric (Ratio, formatted as a percentage)
    Women in Leadership %: Numeric (Ratio, formatted as a percentage)
    Women in Workforce %: Numeric (Ratio, formatted as a percentage)
    Climate Grade: Ordinal (Categorical, letter grades)
    Sustainability Initiatives: Nominal (Text) with some missing values"""
```

c) Calculate the mean, median, mode and midrange for required attribute to measure the center of data.

```
[6]  """(c)"""
     import pandas as pd
     rank_mean = df['Rank'].mean()
     rank_median = df['Rank'].median()
     rank_mode = df['Rank'].mode()[0]

     rank_min = df['Rank'].min()
     rank_max = df['Rank'].max()
     rank_midrange = (rank_min + rank_max) / 2


     print(f"Mean of Rank: {rank_mean}")
     print(f"Median of Rank: {rank_median}")
     print(f"Mode of Rank: {rank_mode}")
     print(f"Midrange of Rank: {rank_midrange}")
```

```
Mean of Rank: 50.5
Median of Rank: 50.5
Mode of Rank: 1
Midrange of Rank: 50.5
```

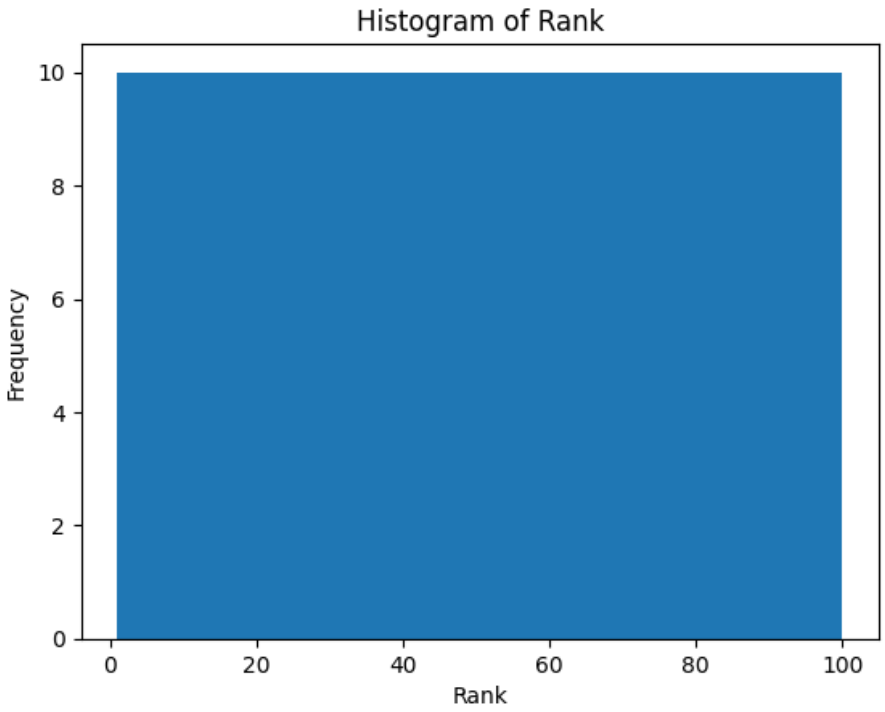
d) Calculate range, quartiles, interquartile range, variance and standard deviation for required attribute to measure the dispersion of data.

```
[7]  """(d)"""
     rank_range = df['Rank'].max() - df['Rank'].min()
     rank_quartiles = df['Rank'].quantile([0.25, 0.5, 0.75])
     rank_iqr = rank_quartiles[0.75] - rank_quartiles[0.25]
     rank_variance = df['Rank'].var()
     rank_std_dev = df['Rank'].std()

     print(f"Range of Rank: {rank_range}")
     print(f"Quartiles of Rank: {rank_quartiles}")
     print(f"Interquartile Range of Rank: {rank_iqr}")
     print(f"Variance of Rank: {rank_variance}")
     print(f"Standard Deviation of Rank: {rank_std_dev}")
```

```
Range of Rank: 99
Quartiles of Rank: 0.25    25.75
0.50    50.50
0.75    75.25
Name: Rank, dtype: float64
Interquartile Range of Rank: 49.5
Variance of Rank: 841.6666666666666
Standard Deviation of Rank: 29.011491975882016
```

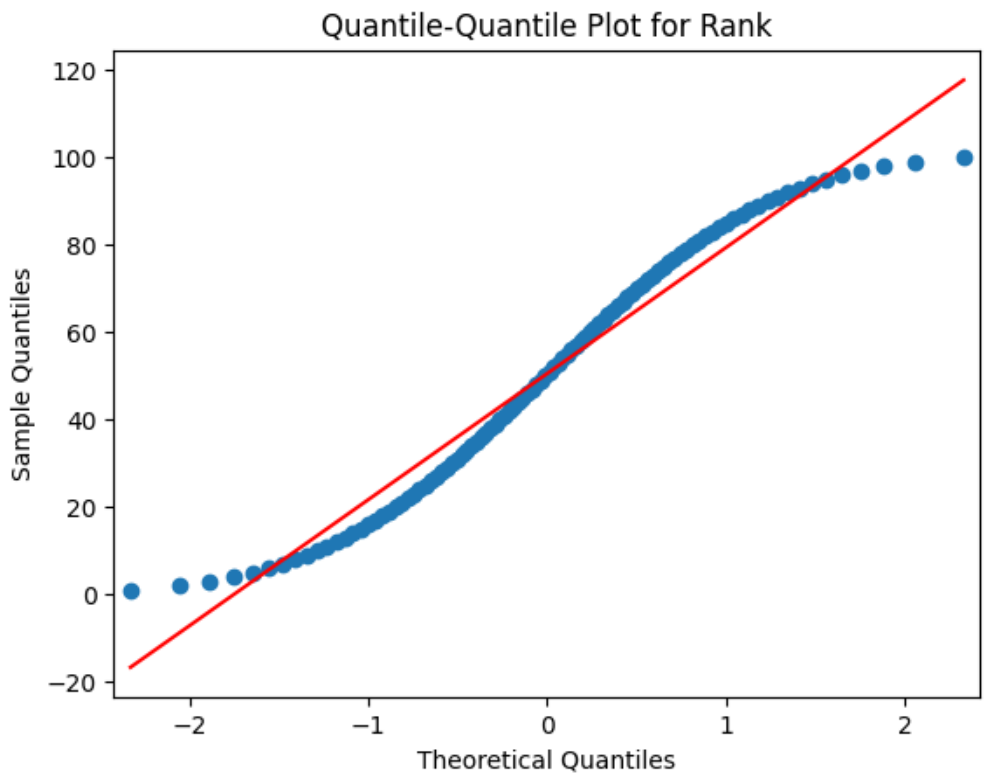
e) Identify whether the data is normally distributed or left and right skewed by plotting the required attribute using histogram.

```
[8]  """(e)"""

     import matplotlib.pyplot as plt
     plt.hist(df['Rank'], bins=10)
     plt.xlabel('Rank')
     plt.ylabel('Frequency')
     plt.title('Histogram of Rank')
     plt.show()
     """ Nominally Distributed"""
```



f) Draw the Quantile plot to check the distribution of data based on quantiles (use univariate data distribution).

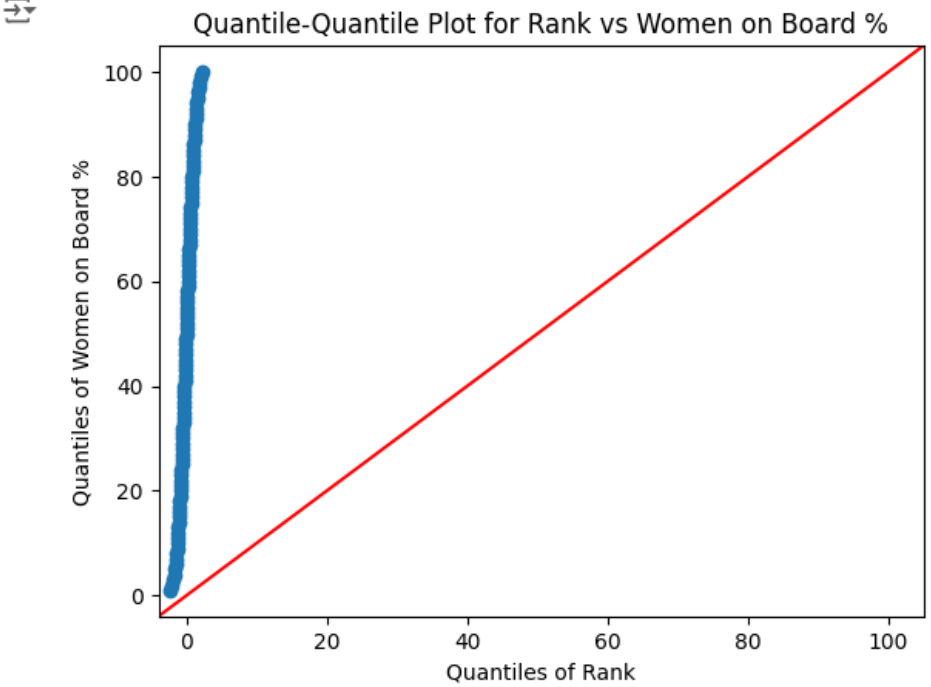
```
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm

data = df['Rank']

sm.qqplot(data, line='s')
plt.title('Quantile-Quantile Plot for Rank')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.show()
```



Draw Quantile-Quantile plot and determine whether the data follows the same distribution (use attributes from two different data sets)

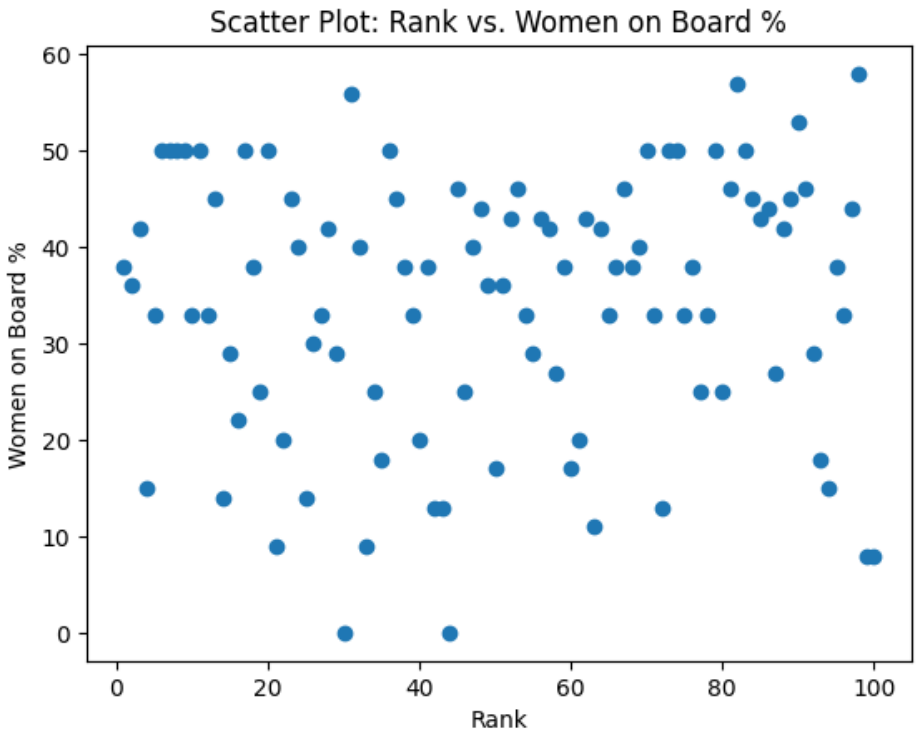
g)

```
import numpy as np
import statsmodels.api as sm

df2 = pd.read_csv("//content//most sustainable corporations.csv", encoding='latin-1')
data1 = df['Rank']
data2 = df2['Women on Board %'].str.replace('%','').astype(float) # convert data2 to numeric type

# Use data1 as sample data and compare it to a normal distribution
sm.qqplot(data1, line='45')
plt.title('Quantile-Quantile Plot for Rank vs Women on Board %')
plt.xlabel('Quantiles of Rank')
plt.ylabel('Quantiles of Women on Board %')
plt.show()
```



h) Draw a Scatter plot for any two attributes to identify whether the attributes have positive, negative or no correlation.

```
"""(h)"""
import matplotlib.pyplot as plt


df['Women on Board %'] = df['Women on Board %'].str.replace('%', '').astype(float)


plt.scatter(df['Rank'], df['Women on Board %'])
plt.xlabel('Rank')
plt.ylabel('Women on Board %')
plt.title('Scatter Plot: Rank vs. Women on Board %')
plt.show()
```



i) Calculate the dissimilarity measure for the selected dataset by selecting few records (should include nominal, binary, ordinal and numeric attributes).

[13]

```
import pandas as pd
from scipy.spatial.distance import euclidean, cityblock, cosine

selected_records = df.loc[[0, 1, 2, 3], ['Rank', 'Company', 'Location', 'Women on Board %']]

def nominal_to_binary(df, column):
  """Converts nominal column to binary representation (one-hot encoding)."""
  dummies = pd.get_dummies(df[column], prefix=column)
  return pd.concat([df, dummies], axis=1).drop(column, axis=1)


selected_records = nominal_to_binary(selected_records, 'Company')
selected_records = nominal_to_binary(selected_records, 'Location')

numeric_data = selected_records[['Rank', 'Women on Board %']].values

euclidean_distance = euclidean(numeric_data[0], numeric_data[1])

manhattan_distance = cityblock(numeric_data[0], numeric_data[1])

cosine_similarity = 1 - cosine(numeric_data[0], numeric_data[1])


print(f"Euclidean Distance between records 0 and 1: {euclidean_distance}")
print(f"Manhattan Distance between records 0 and 1: {manhattan_distance}")
print(f"Cosine Similarity between records 0 and 1: {cosine_similarity}")
```

```
Euclidean Distance between records 0 and 1: 2.23606797749979
Manhattan Distance between records 0 and 1: 3.0
Cosine Similarity between records 0 and 1: 0.9995740375718564
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Exercise 2:

I. Preprocess the selected data set

a. Remove missing values using any one suitable technique of the following:

i. Ignore the tuple

ii. Fill in the missing value manually

iii. Use a global constant to fill in the missing value

iv. Use a measure of central tendency for the attribute (e.g., the mean or median) to fill in the missing value

v. Use the attribute mean or median for all samples belonging to the same class as the given tuple

vi. Use the most probable value to fill in the missing value

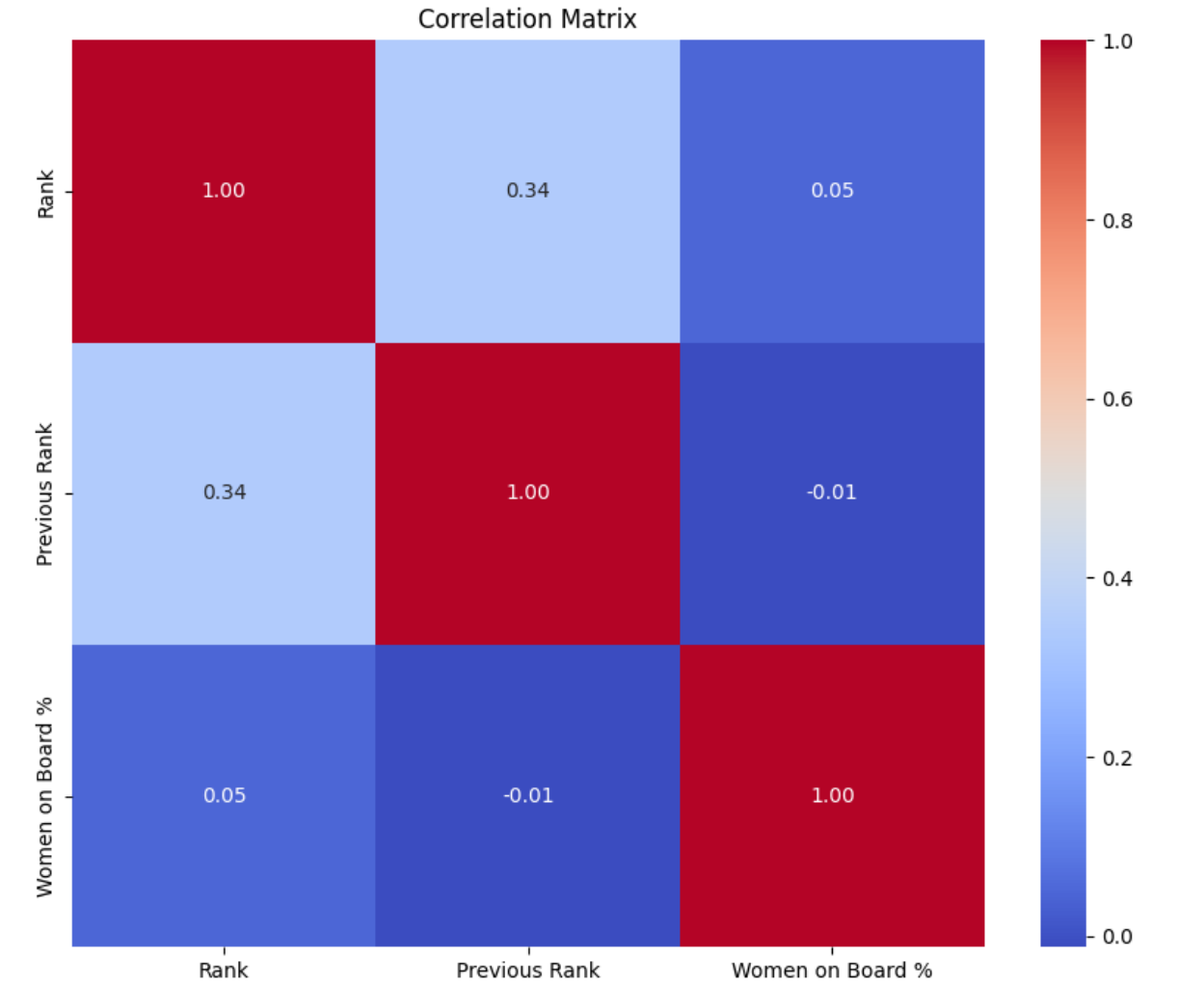
```
df_no_missing_ignore = df.dropna()
df_no_missing_global = df.fillna('Unknown')
# Convert 'Previous Rank' column to numeric type, coercing non-numeric values to NaN
df['Previous Rank'] = pd.to_numeric(df['Previous Rank'], errors='coerce')
median_previous_rank = df['Previous Rank'].median()
df['Previous Rank'].fillna(median_previous_rank, inplace=True)
df['Previous Rank'] = df.groupby('Industry')['Previous Rank'].transform(lambda x: x.fillna(x.median()))
```

b. Remove noisy data if any.

II. Perform data binning on the selected data.

III. Calculate Correlation between variables and create a correlation matrix.

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

bins = pd.cut(df['Rank'], bins=5)
print(bins.value_counts())

numeric_columns = df.select_dtypes(include=np.number).columns

correlation_matrix = df[numeric_columns].corr()

print(correlation_matrix)

import seaborn as sns
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()

def remove_outliers_iqr(df, column):
  Q1 = df[column].quantile(0.25)
  Q3 = df[column].quantile(0.75)
  IQR = Q3 - Q1
  lower_bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR

  df_no_outliers = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
  return df_no_outliers

df_no_outliers_rank = remove_outliers_iqr(df, 'Rank')

print(f"Original DataFrame shape: {df.shape}")
print(f"DataFrame shape after removing outliers for 'Rank': {df_no_outliers_rank.shape}")
```



Note: You can use different data sets for each question as per the requirement and specify the links of the dataset used for each question or select an appropriate dataset that can work for all the above.