```
In [1]: #DATA CLEANING AND PREPARATION
        import pandas as pd

        # Create a sample DataFrame
        data = {
            'Name': ['Dharun', 'Shriya', 'Barbarian', 'Archer', 'Devil'],
            'Age': [28, 22, None, 32, 29],
            'City': ['New York', None, 'Berlin', 'London', 'Tokyo'],
            'Salary': [50000, 54000, 58000, None, 62000]
        }

        # Save the DataFrame as a CSV file
        df_to_save = pd.DataFrame(data)
        df_to_save.to_csv('data.csv', index=False)
        print("CSV File Created and Saved!\n")

        # Load the created CSV file
        df = pd.read_csv('data.csv')
        print("Loaded Data (CSV):\n", df.head())
```

```
CSV File Created and Saved!

Loaded Data (CSV):
          Name   Age      City   Salary
0     Dharun  28.0  New York  50000.0
1     Shriya  22.0       NaN  54000.0
2  Barbarian   NaN    Berlin  58000.0
3     Archer  32.0    London      NaN
4      Devil  29.0     Tokyo  62000.0
```

In [2]:
```python
# View missing data
print("Missing Data Summary:\n", df.isnull().sum())

# Drop rows with missing values
df_dropped = df.dropna()
print("Data after Dropping Missing Values:\n", df_dropped)

# Fill missing values with a specific value (e.g., 0)
df_filled_zeros = df.fillna(0)
print("Data after Filling Missing Values with 0:\n", df_filled_zeros)

# Fill missing values with the mean for numerical columns
df_filled_mean = df.fillna(df.mean(numeric_only=True))
print("Data after Filling Missing Values with Mean:\n", df_filled_mean)
```

```
Missing Data Summary:
 Name      0
Age       1
City      1
Salary    1
dtype: int64
Data after Dropping Missing Values:
      Name   Age      City   Salary
0  Dharun  28.0  New York  50000.0
4   Devil  29.0     Tokyo  62000.0
Data after Filling Missing Values with 0:
         Name   Age      City   Salary
0     Dharun  28.0  New York  50000.0
1     Shriya  22.0         0  54000.0
2  Barbarian   0.0    Berlin  58000.0
3     Archer  32.0    London      0.0
4      Devil  29.0     Tokyo  62000.0
Data after Filling Missing Values with Mean:
         Name    Age      City   Salary
0     Dharun  28.00  New York  50000.0
1     Shriya  22.00       NaN  54000.0
2  Barbarian  27.75    Berlin  58000.0
3     Archer  32.00    London  56000.0
4      Devil  29.00     Tokyo  62000.0
```

In [3]:
```python
# View missing data
print("Missing Data Summary:\n", df.isnull().sum())

# Drop rows with missing values
df_dropped = df.dropna()
print("Data after Dropping Missing Values:\n", df_dropped)

# Fill missing values with a specific value (e.g., 0)
df_filled_zeros = df.fillna(0)
print("Data after Filling Missing Values with 0:\n", df_filled_zeros)

# Fill missing values with the mean for numerical columns
df_filled_mean = df.fillna(df.mean(numeric_only=True))
print("Data after Filling Missing Values with Mean:\n", df_filled_mean)
```

```
Missing Data Summary:
 Name      0
Age       1
City      1
Salary    1
dtype: int64
Data after Dropping Missing Values:
      Name   Age      City   Salary
0  Dharun  28.0  New York  50000.0
4   Devil  29.0     Tokyo  62000.0
Data after Filling Missing Values with 0:
         Name   Age      City   Salary
0     Dharun  28.0  New York  50000.0
1     Shriya  22.0         0  54000.0
2  Barbarian   0.0    Berlin  58000.0
3     Archer  32.0    London      0.0
4      Devil  29.0     Tokyo  62000.0
Data after Filling Missing Values with Mean:
         Name    Age      City   Salary
0     Dharun  28.00  New York  50000.0
1     Shriya  22.00       NaN  54000.0
2  Barbarian  27.75    Berlin  58000.0
3     Archer  32.00    London  56000.0
4      Devil  29.00     Tokyo  62000.0
```

In [4]:
```
# View missing data
print("Missing Data Summary:\n", df.isnull().sum())

# Drop rows with missing values
df_dropped = df.dropna()
print("Data after Dropping Missing Values:\n", df_dropped)

# Fill missing values with a specific value (e.g., 0)
df_filled_zeros = df.fillna(0)
print("Data after Filling Missing Values with 0:\n", df_filled_zeros)

# Fill missing values with the mean for numerical columns
df_filled_mean = df.fillna(df.mean(numeric_only=True))
print("Data after Filling Missing Values with Mean:\n", df_filled_mean)
```

```
Missing Data Summary:
 Name      0
Age       1
City      1
Salary    1
dtype: int64
Data after Dropping Missing Values:
      Name   Age      City   Salary
0  Dharun  28.0  New York  50000.0
4   Devil  29.0     Tokyo  62000.0
Data after Filling Missing Values with 0:
         Name   Age      City   Salary
0     Dharun  28.0  New York  50000.0
1     Shriya  22.0         0  54000.0
2  Barbarian   0.0    Berlin  58000.0
3     Archer  32.0    London      0.0
4      Devil  29.0     Tokyo  62000.0
Data after Filling Missing Values with Mean:
         Name    Age      City   Salary
0     Dharun  28.00  New York  50000.0
1     Shriya  22.00       NaN  54000.0
2  Barbarian  27.75    Berlin  58000.0
3     Archer  32.00    London  56000.0
4      Devil  29.00     Tokyo  62000.0
```

In [5]:
```
# View data types of each column
print("Data Types before Conversion:\n", df.dtypes)

# Convert 'Age' to integer (after filling missing values with mean)
df['Age'] = df['Age'].fillna(df['Age'].mean()).astype(int)
print("Data Types after Conversion:\n", df.dtypes)
```

```
Data Types before Conversion:
 Name       object
Age       float64
City       object
Salary    float64
dtype: object
Data Types after Conversion:
 Name       object
Age         int32
City       object
Salary    float64
dtype: object
```

In [6]:
```
# Rename 'Salary' to 'Annual Salary'
df_renamed = df.rename(columns={'Salary': 'Annual Salary'})
print("Data after Renaming Columns:\n", df_renamed.head())
```

```
Data after Renaming Columns:
         Name  Age      City  Annual Salary
0     Dharun   28  New York        50000.0
1     Shriya   22       NaN        54000.0
2  Barbarian   27    Berlin        58000.0
3     Archer   32    London            NaN
4      Devil   29     Tokyo        62000.0
```

In [7]:
```
# Convert the 'City' column to dummy variables (one-hot encoding)
df_dummies = pd.get_dummies(df, columns=['City'], drop_first=True)
print("Data after One-Hot Encoding 'City':\n", df_dummies.head())
```

```
Data after One-Hot Encoding 'City':
         Name  Age   Salary  City_London  City_New York  City_Tokyo
0     Dharun   28  50000.0        False           True       False
1     Shriya   22  54000.0        False          False       False
2  Barbarian   27  58000.0        False          False       False
3     Archer   32      NaN         True          False       False
4      Devil   29  62000.0        False          False        True
```

In [8]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Standardization (mean = 0, variance = 1)
scaler_standard = StandardScaler()
df_scaled_standard = pd.DataFrame(scaler_standard.fit_transform(df[['Age', 'Salary']].fillna(0)), columns=['Age', 'Sal
print("Data after Standardization:\n", df_scaled_standard.head())

# Normalization (scaling values between 0 and 1)
scaler_minmax = MinMaxScaler()
df_scaled_minmax = pd.DataFrame(scaler_minmax.fit_transform(df[['Age', 'Salary']].fillna(0)), columns=['Age', 'Salary'
print("Data after Normalization:\n", df_scaled_minmax.head())
```

```
Data after Standardization:
         Age    Salary
0  0.122628  0.228528
1 -1.716790  0.404318
2 -0.183942  0.580109
3  1.348907 -1.968855
4  0.429198  0.755900
Data after Normalization:
    Age    Salary
0  0.6  0.806452
1  0.0  0.870968
2  0.5  0.935484
3  1.0  0.000000
4  0.7  1.000000
```

In [9]:
```python
# Add a new column with sample date data
df['Date_of_Joining'] = ['2020-01-15', '2019-06-01', '2018-08-09', '2021-07-21', '2022-03-11']

# Convert the 'Date_of_Joining' column to datetime format
df['Date_of_Joining'] = pd.to_datetime(df['Date_of_Joining'])
print("Data after DateTime Conversion:\n", df.head())

# Extract specific parts of the date
df['Year_Joined'] = df['Date_of_Joining'].dt.year
df['Month_Joined'] = df['Date_of_Joining'].dt.month
print("Extracted Year and Month from 'Date_of_Joining':\n", df[['Date_of_Joining', 'Year_Joined', 'Month_Joined']])
```

```
Data after DateTime Conversion:
         Name  Age      City   Salary Date_of_Joining
0     Dharun   28  New York  50000.0      2020-01-15
1     Shriya   22       NaN  54000.0      2019-06-01
2  Barbarian   27    Berlin  58000.0      2018-08-09
3     Archer   32    London      NaN      2021-07-21
4      Devil   29     Tokyo  62000.0      2022-03-11
Extracted Year and Month from 'Date_of_Joining':
   Date_of_Joining  Year_Joined  Month_Joined
0      2020-01-15         2020             1
1      2019-06-01         2019             6
2      2018-08-09         2018             8
3      2021-07-21         2021             7
4      2022-03-11         2022             3
```

In [10]:
```python
# Create a new feature 'Experience' based on 'Date_of_Joining' (years of experience from 2024)
df['Experience'] = 2024 - df['Date_of_Joining'].dt.year
print("Data with New 'Experience' Feature:\n", df[['Name', 'Date_of_Joining', 'Experience']])
```

```
Data with New 'Experience' Feature:
         Name Date_of_Joining  Experience
0     Dharun      2020-01-15           4
1     Shriya      2019-06-01           5
2  Barbarian      2018-08-09           6
3     Archer      2021-07-21           3
4      Devil      2022-03-11           2
```

```
In [ ]: 220901020 - DHARUN J
```