```python
import random
import numpy as np
import pandas as pd


list = np.array([])

for _ in range(20):
    list = np.append(list, random.randint(1,5))

freq = np.array([])

for i in range(5):
    freq = np.append(freq, int(0))

for j in list:
    freq[int(j-1)] += 1

print(f"Original list :- {list}\n\n")

print("No.  Frequency")

for _ in range(5):
    print(f" {_+1}  -->  {int(freq[_])}")


r = int(input("Enter the range :- "))

odd = []
even = []

for _ in range(1, r+1):
    if _%2:
        odd.append(_)
    else:
        even.append(_)


first20 = []
for _ in range(1, 41, 2):
    first20.append(_)

prime = []
for i in first20:
    isprime = True
    for j in range(2, i):
        if i%j == 0:
            isprime = False
    if isprime and i!=1:
        prime.append(i)

print(f"Odd Numbers :- {odd}\n\n")
print(f"Even Numbers :- {even}\n\n")
print(f"Prime numbers in the first 20 odd numbers :- {prime}")

df = pd.read_csv("wine.csv")
print(df)
```

```
"""
Output:-

Original list :- [1. 5. 3. 3. 4. 3. 3. 4. 4. 5. 5. 4. 3. 5. 3. 3. 1. 2.
4. 5.]


No.  Frequency
 1  -->  2
 2  -->  1
 3  -->  7
 4  -->  5
 5  -->  5
Odd Numbers :- [1, 3, 5, 7, 9]


Even Numbers :- [2, 4, 6, 8, 10]


Prime numbers in the first 20 odd numbers :- [3, 5, 7, 11, 13, 17, 19,
23, 29, 31, 37]
    Duration  Pulse  Maxpulse  Calories
0         60    110       130     409.1
1         60    117       145     479.0
2         60    103       135     340.0
3         45    109       175     282.4
4         45    117       148     406.0
5         60    102       127     300.5
6         60    110       136     374.0
7         45    104       134     253.3
...
45        60     99       119     273.0
46        60    109       153     387.6
47        45    111       136     300.0
48        45    108       129     298.0
"""
```

```
original_tuple = ("A", "T", "H", "A", "R", "V", "A")

reversed_tuple = tuple(reversed(original_tuple))

a, b, c, d, e, f, g = original_tuple

print(f"Original Tuple: {original_tuple}\n")
print(f"Reversed Tuple: {reversed_tuple}\n")
print(f"Unpacked Variables: {a}{b}{c}{d}{e}{f}{g}")

dict1 = {"Atharva" : 32,
        "Sushant" : 28}

dict2 = {"Pragati" : 25,
        "Yukta" : 27}

merged_dict = {}

for key, value in dict1.items():
    merged_dict[key] = value

for key, value in dict2.items():
    merged_dict[key] = value


only_keys = {}

for key in merged_dict:
    only_keys[key] = None


print(f"Dictionary 1 :- {dict1}\n")
print(f"Dictionary 2 :- {dict2}\n")
print(f"Merged Dictionary :- {merged_dict}\n")
print(f"Extrated Key Values from Merged Dictionary :- {only_keys}")

"""
Output :-
Original Tuple: ('A', 'T', 'H', 'A', 'R', 'V', 'A')

Reversed Tuple: ('A', 'V', 'R', 'A', 'H', 'T', 'A')

Unpacked Variables: ATHARVA
Dictionary 1 :- {'Atharva': 32, 'Sushant': 28}

Dictionary 2 :- {'Pragati': 25, 'Yukta': 27}

Merged Dictionary :- {'Atharva': 32, 'Sushant': 28, 'Pragati': 25,
'Yukta': 27}

Extrated Key Values from Merged Dictionary :- {'Atharva': None,
'Sushant': None, 'Pragati': None, 'Yukta': None}
"""
```

```
import numpy as np

matrix1 = np.array([[1, 2, 6],[3, 4, 3],[6, 9, 6]])
matrix2 = np.array([[4, 5, 8],[3, 0, 1],[4, 4, 6]])

matrix_add = matrix1 + matrix2
matrix_sub = matrix2 - matrix1
matrix_multiply = matrix1 @ matrix2
matrix_divide = matrix2 / matrix1

print(f"Addition :- \n{matrix_add}\n")
print(f"Subtraction :- \n{matrix_sub}\n")
print(f"Multiplication :- \n{matrix_multiply}\n")
print(f"Division :- \n{matrix_divide}\n")

"""
Output :-
Addition :-
[[ 5  7 14]
 [ 6  4  4]
 [10 13 12]]

Subtraction :-
[[ 3  3  2]
 [ 0 -4 -2]
 [-2 -5  0]]

Multiplication :-
[[34 29 46]
 [36 27 46]
 [75 54 93]]

Division :-
[[4.         2.5        1.33333333]
 [1.         0.         0.33333333]
 [0.66666667 0.44444444 1.        ]]
"""
```

```
import numpy as np

matrix = np.array([[4, 5, 8],[3, 0, 1],[4, 4, 6]])

transpose_matrix = np.transpose(matrix)

print(f"Original Matrix :- \n{matrix}\n")
print(f"Transposed Matrix :- \n{transpose_matrix}\n")

A = np.array([[2, 1, -1],
              [1, 3, 2],
              [3, 2, 4]])

B = np.array([1, 2, 3])

solution = np.linalg.solve(A, B)

print(f"Coefficient Matrix :- \n{A}\n")
print(f"Solution Vector :-\n{solution}\n")

"""
Output :-
Original Matrix :-
[[4 5 8]
 [3 0 1]
 [4 4 6]]

Transposed Matrix :-
[[4 3 4]
 [5 0 4]
 [8 1 6]]

Coefficient Matrix :-
[[ 2  1 -1]
 [ 1  3  2]
 [ 3  2  4]]

Solution Vector :-
[0.44 0.36 0.24]
"""
```

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler,
MaxAbsScaler
import numpy as np

df = pd.read_csv('diabetes_unclean.csv')
df.columns
choice = 0

while choice != 7:
    print("1. Search Duplicate Records")
    print("2. To drop all Null values in the original dataframe")
    print("3. Fill NULL values with a user-given value")
    print("4. Apply MinMax Transformation")
    print("5. Apply Standardization of given data")
    print("6. Apply Maximum Absolute Scaling")
    print("7. Exit")

    ch = int(input("Enter the choice number: "))

    if ch == 1:
        if df.duplicated().sum() == 0:
            print("No duplicate values!")
        else:
            print(df[df.duplicated()])

    elif ch == 2:
        df.dropna(inplace=True)
        print(df)

    elif ch == 3:
        fill_value = float(input("Enter the value to fill NULL values
with: "))
        df.fillna(fill_value, inplace=True)
        print(df)

    elif ch == 4:
        scaler = MinMaxScaler()
        columns_to_scale = ['AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG',
'HDL', 'LDL', 'VLDL', 'BMI']
        df1 = df[columns_to_scale]
        scaled_df = pd.DataFrame(scaler.fit_transform(df1),
columns=columns_to_scale)
        print(scaled_df)

    elif ch == 5:
        scaler = StandardScaler()
        columns_to_scale = ['AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG',
'HDL', 'LDL', 'VLDL', 'BMI']
        df1 = df[columns_to_scale]
        scaled_df = pd.DataFrame(scaler.fit_transform(df1),
columns=columns_to_scale)
        print(scaled_df)

    elif ch == 6:
        scaler = MaxAbsScaler()
        columns_to_scale = ['AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG',
'HDL', 'LDL', 'VLDL', 'BMI']
```

```
        df1 = df[columns_to_scale]
        scaled_df = pd.DataFrame(scaler.fit_transform(df1), 
columns=columns_to_scale)
        print(scaled_df)

"""
Output :-
1. Search Duplicate Records
2. To drop all Null values in the original dataframe
3. Fill NULL values with a user-given value
4. Apply MinMax Transformation
5. Apply Standardization of given data
6. Apply Maximum Absolute Scaling
7. Exit
Enter the choice number: 1
No duplicate values!
1. Search Duplicate Records
2. To drop all Null values in the original dataframe
3. Fill NULL values with a user-given value
4. Apply MinMax Transformation
5. Apply Standardization of given data
6. Apply Maximum Absolute Scaling
7. Exit
Enter the choice number: 2
       ID  No_Pation Gender   AGE  Urea     Cr  HbA1c  Chol    TG  HDL  
LDL  \
0     502      17975      F  50.0   4.7   46.0    4.9   4.2  0.9  2.4  
1.4   
1     735      34221      M  26.0   4.5   62.0    4.9   3.7  1.4  1.1  
2.1   
2     420      47975      F  50.0   4.7   46.0    4.9   4.2  0.9  2.4  
1.4   
3     680      87656      F  50.0   4.7   46.0    4.9   4.2  0.9  2.4  
1.4   
4     504      34223      M  33.0   7.1   46.0    4.9   4.9  1.0  0.8  
2.0   
...   ...         ...    ...   ...   ...    ...    ...   ...  ...  ...  
...   
1000  185     454316      M  64.0   8.8  106.0    8.5   5.9  2.1  1.2  
4.0   
1002  188     454316      F  75.0  10.3  113.0    8.6   4.2  1.6  0.9  
2.6   
1003  189     454316      M  58.0   4.0   55.0    7.9   4.9  2.0  1.2  
1.4   
1007  194     454316      F  57.0   4.1   70.0    9.3   5.3  3.3  1.0  
1.4   
1008  195       4543      f  55.0   4.1   34.0   13.9   5.4  1.6  1.6  
3.1   

      VLDL   BMI CLASS  
0      0.5  24.0     N  
1      0.6  23.0     N  
2      0.5  24.0     N  
3      0.5  24.0     N  
4      0.4  21.0     N  
...    ...   ...   ...  
1000   1.2  32.0    Y  
1002   0.7  32.0    Y  
```

```
1003   1.1  35.0    Y
1007   1.3  29.0    Y
1008   0.7  33.0    Y

[994 rows x 14 columns]
1. Search Duplicate Records
2. To drop all Null values in the original dataframe
3. Fill NULL values with a user-given value
4. Apply MinMax Transformation
5. Apply Standardization of given data
6. Apply Maximum Absolute Scaling
7. Exit
Enter the choice number: 3
Enter the value to fill NULL values with: 10
       ID  No_Pation Gender   AGE  Urea     Cr  HbA1c  Chol   TG  HDL
LDL  \
0     502      17975      F  50.0   4.7   46.0    4.9   4.2  0.9  2.4
1.4
1     735      34221      M  26.0   4.5   62.0    4.9   3.7  1.4  1.1
2.1
2     420      47975      F  50.0   4.7   46.0    4.9   4.2  0.9  2.4
1.4
3     680      87656      F  50.0   4.7   46.0    4.9   4.2  0.9  2.4
1.4
4     504      34223      M  33.0   7.1   46.0    4.9   4.9  1.0  0.8
2.0
...   ...        ...    ...   ...   ...    ...    ...   ...  ...  ...
...
1000  185     454316      M  64.0   8.8  106.0    8.5   5.9  2.1  1.2
4.0
1002  188     454316      F  75.0  10.3  113.0    8.6   4.2  1.6  0.9
2.6
1003  189     454316      M  58.0   4.0   55.0    7.9   4.9  2.0  1.2
1.4
1007  194     454316      F  57.0   4.1   70.0    9.3   5.3  3.3  1.0
1.4
1008  195       4543      f  55.0   4.1   34.0   13.9   5.4  1.6  1.6
3.1

      VLDL   BMI CLASS
0      0.5  24.0     N
1      0.6  23.0     N
2      0.5  24.0     N
3      0.5  24.0     N
4      0.4  21.0     N
...    ...   ...   ...
1000   1.2  32.0    Y
1002   0.7  32.0    Y
1003   1.1  35.0    Y
1007   1.3  29.0    Y
1008   0.7  33.0    Y

[994 rows x 14 columns]
1. Search Duplicate Records
2. To drop all Null values in the original dataframe
3. Fill NULL values with a user-given value
4. Apply MinMax Transformation
5. Apply Standardization of given data
```

```
6. Apply Maximum Absolute Scaling
7. Exit
Enter the choice number: 5
          AGE      Urea        Cr     HbA1c      Chol        TG       HDL  \
0   -0.411899 -0.145760 -0.378677 -1.332538 -0.507847 -1.032054  1.803674
1   -3.153598 -0.213631 -0.112266 -1.332538 -0.892767 -0.674564 -0.160433
2   -0.411899 -0.145760 -0.378677 -1.332538 -0.507847 -1.032054  1.803674
3   -0.411899 -0.145760 -0.378677 -1.332538 -0.507847 -1.032054  1.803674
4   -2.353936  0.668693 -0.378677 -1.332538  0.031041 -0.960556 -0.613688
..        ...       ...       ...       ...       ...       ...       ...
989  1.187426  1.245598  0.620363  0.084060  0.800881 -0.174077 -0.009348
990  2.444038  1.754631  0.736917  0.123410 -0.507847 -0.531567 -0.462603
991  0.502001 -0.383309 -0.228821 -0.152040  0.031041 -0.245575 -0.009348
992  0.387764 -0.349373  0.020939  0.398860  0.338977  0.683900 -0.311518
993  0.159289 -0.349373 -0.578485  2.208958  0.415961 -0.531567  0.594993

          LDL      VLDL       BMI
0   -1.084782 -0.370087 -1.128439
1   -0.457300 -0.342856 -1.329995
2   -1.084782 -0.370087 -1.128439
3   -1.084782 -0.370087 -1.128439
4   -0.546940 -0.397318 -1.733109
..        ...       ...       ...
989  1.245863 -0.179468  0.484016
990 -0.009099 -0.315624  0.484016
991 -1.084782 -0.206700  1.088687
992 -1.084782 -0.152237 -0.120654
993  0.439102 -0.315624  0.685573

[994 rows x 10 columns]
1. Search Duplicate Records
2. To drop all Null values in the original dataframe
3. Fill NULL values with a user-given value
4. Apply MinMax Transformation
5. Apply Standardization of given data
6. Apply Maximum Absolute Scaling
7. Exit
Enter the choice number: 6
          AGE      Urea       Cr    HbA1c      Chol        TG       HDL  \
0    0.632911  0.120823  0.05750  0.30625  0.407767  0.065217  0.242424
1    0.329114  0.115681  0.07750  0.30625  0.359223  0.101449  0.111111
2    0.632911  0.120823  0.05750  0.30625  0.407767  0.065217  0.242424
3    0.632911  0.120823  0.05750  0.30625  0.407767  0.065217  0.242424
4    0.417722  0.182519  0.05750  0.30625  0.475728  0.072464  0.080808
..        ...       ...      ...      ...       ...       ...       ...
989  0.810127  0.226221  0.13250  0.53125  0.572816  0.152174  0.121212
990  0.949367  0.264781  0.14125  0.53750  0.407767  0.115942  0.090909
991  0.734177  0.102828  0.06875  0.49375  0.475728  0.144928  0.121212
992  0.721519  0.105398  0.08750  0.58125  0.514563  0.239130  0.101010
993  0.696203  0.105398  0.04250  0.86875  0.524272  0.115942  0.161616

          LDL      VLDL       BMI
0    0.141414  0.014286  0.502618
1    0.212121  0.017143  0.481675
2    0.141414  0.014286  0.502618
3    0.141414  0.014286  0.502618
```

```
4    0.202020  0.011429  0.439791
..        ...       ...       ...
989  0.404040  0.034286  0.670157
990  0.262626  0.020000  0.670157
991  0.141414  0.031429  0.732984
992  0.141414  0.037143  0.607330
993  0.313131  0.020000  0.691099

[994 rows x 10 columns]
1. Search Duplicate Records
2. To drop all Null values in the original dataframe
3. Fill NULL values with a user-given value
4. Apply MinMax Transformation
5. Apply Standardization of given data
6. Apply Maximum Absolute Scaling
7. Exit
Enter the choice number: 7
1. Search Duplicate Records
2. To drop all Null values in the original dataframe
3. Fill NULL values with a user-given value
4. Apply MinMax Transformation
5. Apply Standardization of given data
6. Apply Maximum Absolute Scaling
7. Exit
"""
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('student.csv')

print(df.head())


df['Math_Score'].describe()

percentile25 = df['Math_Score'].quantile(0.25)
percentile75 = df['Math_Score'].quantile(0.75)
print(percentile75)
print(percentile25)

iqr = percentile75 - percentile25
print(f"IQR :- {iqr}")
upper_limit = percentile75 + 1.5 * iqr
lower_limit = percentile25 - 1.5 * iqr
print("Upper limit",upper_limit)
print("Lower limit",lower_limit)
df[df['Math_Score'] > upper_limit]
df[df['Math_Score'] < lower_limit]
new_df = df[df['Math_Score'] < upper_limit]
new_df.shape
new_df_cap = df.copy()

new_df_cap['Math_Score'] = np.where(
    new_df_cap['Math_Score'] > upper_limit,
    upper_limit,
    np.where(
        new_df_cap['Math_Score'] < lower_limit,
        lower_limit,
        new_df_cap['Math_Score']
    )
)
new_df_cap.shape

"""
Output :-
Student_ID     Name  Age  Math_Score  English_Score
0           1    Alice   20          85             75
1           2      Bob   21          78             88
2           3  Charlie   22          92             80
3           4    David   19          86             92
4           5      Eve   20          98             87
92.0
75.0
IQR :- 17.0
Upper limit 117.5
Lower limit 49.5
(200, 5)
"""
```

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.stats import norm
from warnings import filterwarnings
import seaborn as sns
from scipy.stats import skew
from sklearn.preprocessing import LabelEncoder

filterwarnings('ignore')

def z_score_normalization(df, column, new_column):
    df[new_column] = (df[column] - df[column].mean()) / df[column].std()
    return df

def square_root_transformation(df, column, new_column):
    df[new_column] = np.sqrt(df[column])
    return df

def log_transformation(df, column, new_column):
    df[new_column] = np.log(df[column] + 0.001)
    return df

def plot_distribution(data, title, xlabel, ylabel):
    plt.figure(figsize=(12, 5))
    sns.distplot(data, bins=25, hist=True, kde=True, color='skyblue',
hist_kws={'edgecolor': 'black'}, kde_kws={'linewidth': 3})
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()

def main():
    while True:
        print("Menu:")
        print("1. Create a Random Data set")
        print("2. Label Encoding for 'Result'")
        print("3. Z-Score Normalization for Marks")
        print("4. Square Root Transformation for Marks")
        print("5. Log Transformation for Marks")
        print("6. Z-Score Normalization for CGPA")
        print("7. Square Root Transformation for CGPA")
        print("8. Log Transformation for CGPA")
        print("9. Quit")

        choice = input("Enter your choice: ")

        if choice == '1':
            Student_Id = np.arange(1, 26)
            Marks = np.random.randint(0, 100, 25)
            CGPA = np.random.randint(0, 10, 25)
            df = pd.DataFrame({'StudentId': Student_Id, 'Marks': Marks,
'CGPA': CGPA})
            condition1 = [
                ((df['Marks'] > 75) & (df['CGPA'] > 4)),
                ((df['Marks'] > 65) & (df['Marks'] <= 75) & (df['CGPA'] >
4)),
                ((df['Marks'] > 50) & (df['Marks'] <= 65) & (df['CGPA'] >
4))
            ]
            value1 = ['Distinction', 'First Class', 'Second Class']
            df['Result'] = np.select(condition1, value1, default='Fail')
            print(df.to_string(index=False))

        elif choice == '2':
            if 'Result' in df.columns:
                label_encoder = LabelEncoder()
                df['Result_encoded'] =
label_encoder.fit_transform(df['Result'])
                print(df.to_string(index=False))
            else:
                print("'Result' column not found in the dataset.")

        elif choice == '3':
            df = z_score_normalization(df, 'Marks', 'Standardized_Marks')
            print(df.to_string(index=False))
            plot_distribution(df['Standardized_Marks'], 'Z-Score
Normalized Marks Distribution', 'Z-Score Normalized Marks', 'Density')

        elif choice == '4':
            df = square_root_transformation(df, 'Marks', 'Marks_sqrt')
            print(df.to_string(index=False))
            plot_distribution(df['Marks_sqrt'], 'Square Root Transformed
Marks Distribution', 'Square Root Transformed Marks', 'Density')

        elif choice == '5':
            df = log_transformation(df, 'Marks', 'Log_Marks')
            print(df.to_string(index=False))
            plot_distribution(df['Log_Marks'], 'Log-Transformed Marks
Distribution', 'Log(Marks)', 'Density')

        elif choice == '6':
            df = z_score_normalization(df, 'CGPA', 'Standardized_CGPA')
            print(df.to_string(index=False))
            plot_distribution(df['Standardized_CGPA'], 'Z-Score
Normalized CGPA Distribution', 'Z-Score Normalized CGPA', 'Density')

        elif choice == '7':
            df = square_root_transformation(df, 'CGPA', 'CGPA_sqrt')
            print(df.to_string(index=False))
            plot_distribution(df['CGPA_sqrt'], 'Square Root Transformed
CGPA Distribution', 'Square Root Transformed CGPA', 'Density')

        elif choice == '8':
            df = log_transformation(df, 'CGPA', 'Log_CGPA')
            print(df.to_string(index=False))
            plot_distribution(df['Log_CGPA'], 'Log-Transformed CGPA
Distribution', 'Log(CGPA)', 'Density')

        elif choice == '9':
            break

        else:
            print("Invalid choice. Please enter a valid option.")
```

```
if __name__ == "__main__":
    main()

"""
Output :-
Menu:
1. Create a Random Data set
2. Label Encoding for 'Result'
3. Z-Score Normalization for Marks
4. Square Root Transformation for Marks
5. Log Transformation for Marks
6. Z-Score Normalization for CGPA
7. Square Root Transformation for CGPA
8. Log Transformation for CGPA
9. Quit
Enter your choice: 1
 StudentId  Marks  CGPA       Result
         1     13     1         Fail
         2     23     4         Fail
         3      3     5         Fail
         4     62     2         Fail
         5     28     2         Fail
         6     89     9  Distinction
         7     83     7  Distinction
         8     20     0         Fail
         9     54     2         Fail
        10     68     0         Fail
        11     63     2         Fail
        12     64     4         Fail
        13     29     4         Fail
        14     18     5         Fail
        15     23     7         Fail
        16     57     5 Second Class
        17     89     5  Distinction
        18     28     5         Fail
        19     17     4         Fail
        20     63     0         Fail
        21     65     1         Fail
        22      5     6         Fail
        23     41     4         Fail
        24     89     6  Distinction
        25     71     3         Fail
Menu:
1. Create a Random Data set
2. Label Encoding for 'Result'
3. Z-Score Normalization for Marks
4. Square Root Transformation for Marks
5. Log Transformation for Marks
6. Z-Score Normalization for CGPA
7. Square Root Transformation for CGPA
8. Log Transformation for CGPA
9. Quit
Enter your choice: 2
 StudentId  Marks  CGPA       Result  Result_encoded
         1     13     1         Fail               1
         2     23     4         Fail               1
         3      3     5         Fail               1
         4     62     2         Fail               1
         5     28     2         Fail               1
         6     89     9  Distinction               0
         7     83     7  Distinction               0
         8     20     0         Fail               1
         9     54     2         Fail               1
        10     68     0         Fail               1
        11     63     2         Fail               1
        12     64     4         Fail               1
        13     29     4         Fail               1
        14     18     5         Fail               1
        15     23     7         Fail               1
        16     57     5 Second Class               2
        17     89     5  Distinction               0
        18     28     5         Fail               1
        19     17     4         Fail               1
        20     63     0         Fail               1
        21     65     1         Fail               1
        22      5     6         Fail               1
        23     41     4         Fail               1
        24     89     6  Distinction               0
        25     71     3         Fail               1
Menu:
1. Create a Random Data set
2. Label Encoding for 'Result'
3. Z-Score Normalization for Marks
4. Square Root Transformation for Marks
5. Log Transformation for Marks
6. Z-Score Normalization for CGPA
7. Square Root Transformation for CGPA
8. Log Transformation for CGPA
9. Quit
Enter your choice: 3
 StudentId  Marks  CGPA       Result  Result_encoded  Standardized_Marks
         1     13     1         Fail               1           -1.205006
         2     23     4         Fail               1           -0.846373
         3      3     5         Fail               1           -1.563638
         4     62     2         Fail               1            0.552294
         5     28     2         Fail               1           -0.667057
         6     89     9  Distinction               0            1.520602
         7     83     7  Distinction               0            1.305423
         8     20     0         Fail               1           -0.953963
         9     54     2         Fail               1            0.265388
        10     68     0         Fail               1            0.767474
        11     63     2         Fail               1            0.588158
        12     64     4         Fail               1            0.624021
        13     29     4         Fail               1           -0.631193
        14     18     5         Fail               1           -1.025689
        15     23     7         Fail               1           -0.846373
        16     57     5 Second Class               2            0.372978
        17     89     5  Distinction               0            1.520602
        18     28     5         Fail               1           -0.667057
        19     17     4         Fail               1           -1.061553
        20     63     0         Fail               1            0.588158
        21     65     1         Fail               1            0.659884
        22      5     6         Fail               1           -1.491912
        23     41     4         Fail               1           -0.200834
        24     89     6  Distinction               0            1.520602
        25     71     3         Fail               1            0.875064
```

```
Menu:
1. Create a Random Data set
2. Label Encoding for 'Result'
3. Z-Score Normalization for Marks
4. Square Root Transformation for Marks
5. Log Transformation for Marks
6. Z-Score Normalization for CGPA
7. Square Root Transformation for CGPA
8. Log Transformation for CGPA
9. Quit
Enter your choice: 4
 StudentId  Marks  CGPA        Result  Result_encoded  Standardized_Marks
Marks_sqrt
         1     13     1          Fail               1           -1.205006
3.605551
         2     23     4          Fail               1           -0.846373
4.795832
         3      3     5          Fail               1           -1.563638
1.732051
         4     62     2          Fail               1            0.552294
7.874008
         5     28     2          Fail               1           -0.667057
5.291503
         6     89     9   Distinction               0            1.520602
9.433981
         7     83     7   Distinction               0            1.305423
9.110434
         8     20     0          Fail               1           -0.953963
4.472136
         9     54     2          Fail               1            0.265388
7.348469
        10     68     0          Fail               1            0.767474
8.246211
        11     63     2          Fail               1            0.588158
7.937254
        12     64     4          Fail               1            0.624021
8.000000
        13     29     4          Fail               1           -0.631193
5.385165
        14     18     5          Fail               1           -1.025689
4.242641
        15     23     7          Fail               1           -0.846373
4.795832
        16     57     5  Second Class               2            0.372978
7.549834
        17     89     5   Distinction               0            1.520602
9.433981
        18     28     5          Fail               1           -0.667057
5.291503
        19     17     4          Fail               1           -1.061553
4.123106
        20     63     0          Fail               1            0.588158
7.937254
        21     65     1          Fail               1            0.659884
8.062258
        22      5     6          Fail               1           -1.491912
2.236068
```

```
        23     41     4         Fail               1           -0.200834
6.403124
        24     89     6  Distinction               0            1.520602
9.433981
        25     71     3         Fail               1            0.875064
8.426150

Menu:
1. Create a Random Data set
2. Label Encoding for 'Result'
3. Z-Score Normalization for Marks
4. Square Root Transformation for Marks
5. Log Transformation for Marks
6. Z-Score Normalization for CGPA
7. Square Root Transformation for CGPA
8. Log Transformation for CGPA
9. Quit
Enter your choice: 5
 StudentId  Marks  CGPA       Result  Result_encoded  Standardized_Marks
Marks_sqrt  Log_Marks
         1     13     1         Fail               1           -1.205006
3.605551   2.565026
         2     23     4         Fail               1           -0.846373
4.795832   3.135538
         3      3     5         Fail               1           -1.563638
1.732051   1.098946
         4     62     2         Fail               1            0.552294
7.874008   4.127151
         5     28     2         Fail               1           -0.667057
5.291503   3.332240
         6     89     9  Distinction               0            1.520602
9.433981   4.488648
         7     83     7  Distinction               0            1.305423
9.110434   4.418853
         8     20     0         Fail               1           -0.953963
4.472136   2.995782
         9     54     2         Fail               1            0.265388
7.348469   3.989003
        10     68     0         Fail               1            0.767474
8.246211   4.219522
        11     63     2         Fail               1            0.588158
7.937254   4.143151
        12     64     4         Fail               1            0.624021
8.000000   4.158899
        13     29     4         Fail               1           -0.631193
5.385165   3.367330
        14     18     5         Fail               1           -1.025689
4.242641   2.890427
        15     23     7         Fail               1           -0.846373
4.795832   3.135538
        16     57     5 Second Class               2            0.372978
7.549834   4.043069
        17     89     5  Distinction               0            1.520602
9.433981   4.488648
        18     28     5         Fail               1           -0.667057
5.291503   3.332240
        19     17     4         Fail               1           -1.061553
4.123106   2.833272
```

```
        20     63     0         Fail               1            0.588158
7.937254   4.143151
        21     65     1         Fail               1            0.659884
8.062258   4.174403
        22      5     6         Fail               1           -1.491912
2.236068   1.609638
        23     41     4         Fail               1           -0.200834
6.403124   3.713596
        24     89     6  Distinction               0            1.520602
9.433981   4.488648
        25     71     3         Fail               1            0.875064
8.426150   4.262694

Menu:
1. Create a Random Data set
2. Label Encoding for 'Result'
3. Z-Score Normalization for Marks
4. Square Root Transformation for Marks
5. Log Transformation for Marks
6. Z-Score Normalization for CGPA
7. Square Root Transformation for CGPA
8. Log Transformation for CGPA
9. Quit
Enter your choice: 7
 StudentId  Marks  CGPA       Result  Result_encoded  Standardized_Marks
Marks_sqrt  Log_Marks  CGPA_sqrt
         1     13     1         Fail               1           -1.205006
3.605551   2.565026   1.000000
         2     23     4         Fail               1           -0.846373
4.795832   3.135538   2.000000
         3      3     5         Fail               1           -1.563638
1.732051   1.098946   2.236068
         4     62     2         Fail               1            0.552294
7.874008   4.127151   1.414214
         5     28     2         Fail               1           -0.667057
5.291503   3.332240   1.414214
         6     89     9  Distinction               0            1.520602
9.433981   4.488648   3.000000
         7     83     7  Distinction               0            1.305423
9.110434   4.418853   2.645751
         8     20     0         Fail               1           -0.953963
4.472136   2.995782   0.000000
         9     54     2         Fail               1            0.265388
7.348469   3.989003   1.414214
        10     68     0         Fail               1            0.767474
8.246211   4.219522   0.000000
        11     63     2         Fail               1            0.588158
7.937254   4.143151   1.414214
        12     64     4         Fail               1            0.624021
8.000000   4.158899   2.000000
        13     29     4         Fail               1           -0.631193
5.385165   3.367330   2.000000
        14     18     5         Fail               1           -1.025689
4.242641   2.890427   2.236068
        15     23     7         Fail               1           -0.846373
4.795832   3.135538   2.645751
        16     57     5 Second Class               2            0.372978
7.549834   4.043069   2.236068
```

```
        17     89     5  Distinction               0            1.520602
9.433981   4.488648   2.236068
        18     28     5         Fail               1           -0.667057
5.291503   3.332240   2.236068
        19     17     4         Fail               1           -1.061553
4.123106   2.833272   2.000000
        20     63     0         Fail               1            0.588158
7.937254   4.143151   0.000000
        21     65     1         Fail               1            0.659884
8.062258   4.174403   1.000000
        22      5     6         Fail               1           -1.491912
2.236068   1.609638   2.449490
        23     41     4         Fail               1           -0.200834
6.403124   3.713596   2.000000
        24     89     6  Distinction               0            1.520602
9.433981   4.488648   2.449490
        25     71     3         Fail               1            0.875064
8.426150   4.262694   1.732051

Menu:
1. Create a Random Data set
2. Label Encoding for 'Result'
3. Z-Score Normalization for Marks
4. Square Root Transformation for Marks
5. Log Transformation for Marks
6. Z-Score Normalization for CGPA
7. Square Root Transformation for CGPA
8. Log Transformation for CGPA
9. Quit
Enter your choice: 8
 StudentId  Marks  CGPA       Result  Result_encoded  Standardized_Marks
Marks_sqrt  Log_Marks  CGPA_sqrt  Log_CGPA
         1     13     1         Fail               1           -1.205006
3.605551   2.565026   1.000000  0.001000
         2     23     4         Fail               1           -0.846373
4.795832   3.135538   2.000000  1.386544
         3      3     5         Fail               1           -1.563638
1.732051   1.098946   2.236068  1.609638
         4     62     2         Fail               1            0.552294
7.874008   4.127151   1.414214  0.693647
         5     28     2         Fail               1           -0.667057
5.291503   3.332240   1.414214  0.693647
         6     89     9  Distinction               0            1.520602
9.433981   4.488648   3.000000  2.197336
         7     83     7  Distinction               0            1.305423
9.110434   4.418853   2.645751  1.946053
         8     20     0         Fail               1           -0.953963
4.472136   2.995782   0.000000 -6.907755
         9     54     2         Fail               1            0.265388
7.348469   3.989003   1.414214  0.693647
        10     68     0         Fail               1            0.767474
8.246211   4.219522   0.000000 -6.907755
        11     63     2         Fail               1            0.588158
7.937254   4.143151   1.414214  0.693647
        12     64     4         Fail               1            0.624021
8.000000   4.158899   2.000000  1.386544
        13     29     4         Fail               1           -0.631193
5.385165   3.367330   2.000000  1.386544
```

```
        14     18     5         Fail             1           -1.025689
4.242641   2.890427   2.236068  1.609638
        15     23     7         Fail             1           -0.846373
4.795832   3.135538   2.645751  1.946053
        16     57     5 Second Class             2            0.372978
7.549834   4.043069   2.236068  1.609638
        17     89     5  Distinction             0            1.520602
9.433981   4.488648   2.236068  1.609638
        18     28     5         Fail             1           -0.667057
5.291503   3.332240   2.236068  1.609638
        19     17     4         Fail             1           -1.061553
4.123106   2.833272   2.000000  1.386544
        20     63     0         Fail             1            0.588158
7.937254   4.143151   0.000000 -6.907755
        21     65     1         Fail             1            0.659884
8.062258   4.174403   1.000000  0.001000
        22      5     6         Fail             1           -1.491912
2.236068   1.609638   2.449490  1.791926
        23     41     4         Fail             1           -0.200834
6.403124   3.713596   2.000000  1.386544
        24     89     6  Distinction             0            1.520602
9.433981   4.488648   2.449490  1.791926
        25     71     3         Fail             1            0.875064
8.426150   4.262694   1.732051  1.098946

Menu:
1. Create a Random Data set
2. Label Encoding for 'Result'
3. Z-Score Normalization for Marks
4. Square Root Transformation for Marks
5. Log Transformation for Marks
6. Z-Score Normalization for CGPA
7. Square Root Transformation for CGPA
8. Log Transformation for CGPA
9. Quit
Enter your choice: 9
"""
```