In [2]:
```
import pandas as pd
```

In [88]:
```
import numpy as np
data = {
    'name': pd.Series(['Alice', 'Bob', 'Charlie', 'David', 'Emma', 'Frank', 'Gra
                       'Katie', 'Liam', 'Mia', 'Nate', 'Olivia', 'Peter', 'Quinn
    'division': pd.Series(['A', 'B', 'A', 'C', 'B', 'A', 'B', 'C', 'B', 'A', 'C'
                           'B', 'A', 'C']),
    'marks1': pd.Series([70, 80, 85, 90, 10, 65, 75, 60, 50, 85, np.nan, 55, 80,
    'marks2': pd.Series([60, 70, 75, 80, 25, 55, 65, 50, 40, 75, 80, 45, 365, 60
    'marks3': pd.Series([5, 60, 65, 70, 75, 45, 55, 40, 30, 65, 70, 35, 60, 50,
}
```

In [90]:
```
df=pd.DataFrame(data)
```

In [94]:
```
df.sample()
```

Out[94]:

	name	division	marks1	marks2	marks3
6	Grace	B	75.0	65.0	55.0

In [98]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   name      20 non-null     object 
 1   division  20 non-null     object 
 2   marks1    19 non-null     float64
 3   marks2    19 non-null     float64
 4   marks3    19 non-null     float64
dtypes: float64(3), object(2)
memory usage: 928.0+ bytes
```

In [102…
```
df.isnull().sum()
```

Out[102…
```
name        0
division    0
marks1      1
marks2      1
marks3      1
dtype: int64
```

In [122…
```
df['marks1'].fillna(df['marks1'].mean())
df['marks2'].fillna(df['marks2'].mean())
df['marks3'].fillna(df['marks3'].mean())
```

Out[122…
```
0      5.000000
1     60.000000
2     65.000000
3     70.000000
4     75.000000
5     45.000000
6     55.000000
7     40.000000
8     30.000000
9     65.000000
10    70.000000
11    35.000000
12    60.000000
13    50.000000
14    55.000000
15    20.000000
16    70.000000
17    60.000000
18    51.315789
19    45.000000
Name: marks3, dtype: float64
```

In [128…
```python
# Define a function for detecting outliers
def detect_outliers_iqr(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return column[(column < lower_bound) | (column > upper_bound)]

# Detect outliers
outliers_marks1 = detect_outliers_iqr(df['marks1'])
outliers_marks2 = detect_outliers_iqr(df['marks2'])
outliers_marks3 = detect_outliers_iqr(df['marks3'])

print("\nOutliers in marks1:\n", outliers_marks1)
print("\nOutliers in marks2:\n", outliers_marks2)
print("\nOutliers in marks3:\n", outliers_marks3)
```

```
Outliers in marks1:
 4     10.0
16     8.0
Name: marks1, dtype: float64

Outliers in marks2:
 12    365.0
Name: marks2, dtype: float64

Outliers in marks3:
 0    5.0
Name: marks3, dtype: float64
```

In [130…
```python
# Cap the outliers using the upper and lower bound method
def cap_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
```

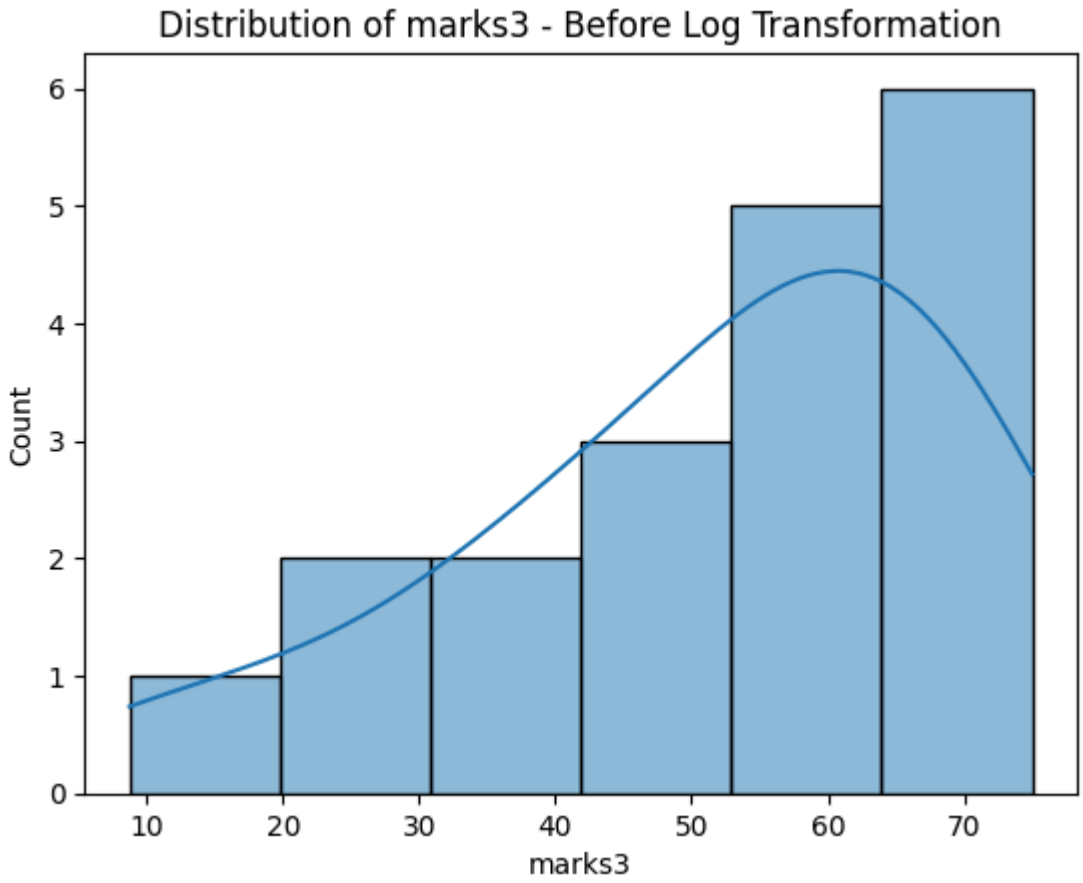
```
    return np.where(column < lower, lower,
           np.where(column > upper, upper, column))

df['marks1'] = cap_outliers(df['marks1'])
df['marks2'] = cap_outliers(df['marks2'])
df['marks3'] = cap_outliers(df['marks3'])
```

In [136…
```
print("Skewness before transformation:", df['marks3'].skew())
```

```
Skewness before transformation: -0.9001327786551979
```

In [144…
```
# Visualize original marks3
import matplotlib.pyplot as plt
sns.histplot(df['marks3'], kde=True)
plt.title("Distribution of marks3 - Before Log Transformation")
plt.show()
```

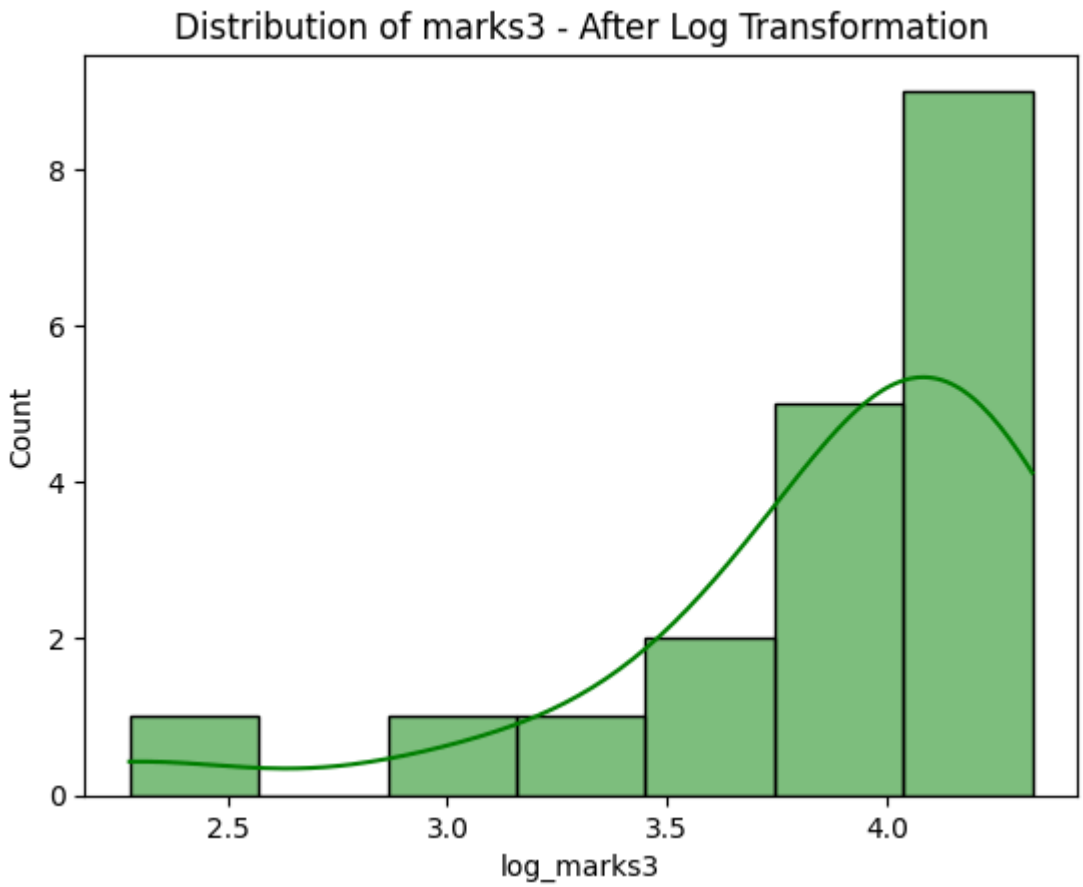


In [146…
```
df['log_marks3'] = np.log(df['marks3'] + 1)
```

In [148…
```
# Check skewness after transformation
print("Skewness after transformation:", df['log_marks3'].skew())
```

```
Skewness after transformation: -2.0575304355427595
```

In [150…
```
# Visualize transformed marks3
sns.histplot(df['log_marks3'], kde=True, color='green')
plt.title("Distribution of marks3 - After Log Transformation")
plt.show()
```


Distribution of marks3 - After Log Transformation
Count
0
2
4
6
8
2.5
3.0
3.5
4.0
log_marks3