Program & Output (Grouped Data)

In [119]:
```
import pandas as pd
import numpy as np
import math
```

In [92]:
```
df = pd.DataFrame({
    'No. of Defectices': ['0 - 2', '2 - 4', '4 - 6', '6 - 8', '8 - 10'],
    'frequency': [3, 4, 5, 3, 1]
})

def low_high(df, column_name):
    df['low'] = df.apply(lambda row: int(row[column_name][0]), axis=1)
    df['high'] = df.apply(lambda row: int(row[column_name].split('-')[-1]), a

low_high(df, 'No. of Defectices')
df
```

Out[92]:

	No. of Defectices	frequency	low	high
0	0 - 2	3	0	2
1	2 - 4	4	2	4
2	4 - 6	5	4	6
3	6 - 8	3	6	8
4	8 - 10	1	8	10

Mean for Grouped Data

In [93]:
```
# calculating classmark(x) and f.x
df1 = df.copy()
df1['classmark'] = (df1['low'] + df1['high'])/2
df1['f.x'] = df1.apply(lambda row: row['frequency']*row['classmark'], axis=1)
df1
```

Out[93]:

	No. of Defectices	frequency	low	high	classmark	f.x
0	0 - 2	3	0	2	1.0	3.0
1	2 - 4	4	2	4	3.0	12.0
2	4 - 6	5	4	6	5.0	25.0
3	6 - 8	3	6	8	7.0	21.0
4	8 - 10	1	8	10	9.0	9.0

In [94]:
```
# applying the mean formula
mean = df1['f.x'].sum() / df1['frequency'].sum()
print(f'Mean: {mean}')
```

```
Mean: 4.375
```

Median for Grouped Data

In [95]:
```
df2 = df.copy()

# calculating cummalative frequency
c = 0
def calculate_cf(row):
    global c
    if c == 0: c = row['frequency']
    else: c = c + row['frequency']
    return c

# calculating N/2
n_by_2 = df2['frequency'].sum()/2

df2['cf'] = df2.apply(lambda row: calculate_cf(row), axis=1)
df2
```

Out[95]:

	No. of Defectices	frequency	low	high	cf
0	0 - 2	3	0	2	3
1	2 - 4	4	2	4	7
2	4 - 6	5	4	6	12
3	6 - 8	3	6	8	15
4	8 - 10	1	8	10	16

In [96]:
```
def get_median_class():
    for index, row in df2.iterrows():
        if row['cf'] > n_by_2:
            return row, index

median_class, class_index = get_median_class()
```

In [97]:
```
# applying the median formula
L = median_class['low']
cf = int(df2.loc[class_index - 1: class_index - 1]['cf'])
f = median_class['frequency']
h = median_class['high'] - median_class['low']

median = L + ((n_by_2 - cf) / f)*h
print(f'Median: {median}')
```

```
Median: 4.4
```

Mode for Grouped Data

In [100]:
```
df3 = df.copy()

def get_modal_class():
    for index, row in df3.iterrows():
        if row['frequency'] == df3['frequency'].max():
            return row, index

modal_class, class_index_mode = get_modal_class()
```

In [101]:
```
# applying the mode formula
L = modal_class['low']
f1 = modal_class['frequency']
f0 = int(df3.loc[class_index_mode - 1: class_index_mode - 1]['frequency'])
f2 = int(df3.loc[class_index_mode + 1: class_index_mode + 1]['frequency'])
h = modal_class['high'] - modal_class['low']

mode = L + ((f1 - f0) / (2*f1 - f0 - f2))*h
print(f'Mode: {mode}')
```

```
Mode: 4.666666666666667
```

Variance & Standard Deviation for Grouped Data

In [120]:
```
df4 = df1.copy()

df4['f.(x-x_mean)^2'] = df4['frequency'] * (df4['classmark'] - mean)*(df4['cl

variance = df4['f.(x-x_mean)^2'].sum() / df4.count().max()
standard_deviation = math.sqrt(variance)
print(f'Variance: {variance}')
print(f'Standard Deviation: {standard_deviation}')
```

```
Variance: 17.15
Standard Deviation: 4.141255848169731
```

Skewness for Grouped Data

In [124]:
```
skewness = (mean - mode) / standard_deviation
print(f'Skewness: {skewness}')
if skewness > 0 and skewness < 0.4:
    print('It is little positively skewed!')
elif skewness >= 0.4 and skewness < 1:
    print('It is moderately positively skewed!')
elif skewness >= 1:
    print('It is highly positively skewed!')
elif skewness > -0.4 and skewness < 0:
    print('It is little negatively skewed!')
elif skewness > -1 and skewness <= 0.4:
    print('It is moderately negatively skewed!')
elif skewness <= 1:
    print('It is highly negatively skewed!')
else:
    print('It is not skewed!')
```

```
Skewness: -0.07042952122737646
It is little negatively skewed!
```

In [ ]: