Exercise 1

In [ ]:
```
import random
random.seed(0)
salaries = [round(random.random()*1000000, -3) for _ in range(100)]
```

Mean

In [77]:
```
mean = sum(salaries)/len(salaries)
print(mean)
```

```
585690.0
```

Median

In [ ]:
```
sorted_salaries = sorted(salaries)
n = len(sorted_salaries)
mid = n // 2

if n % 2 == 1:
    median_salary = sorted_salaries[mid]
else:
    median_salary = (sorted_salaries[mid - 1] + sorted_salaries[mid]) / 2

print(median_salary)
```

```
589000.0
```

Mode

In [ ]:
```
freq = {}
for salary in salaries:
    freq[salary] = freq.get(salary, 0) + 1

max_count = max(freq.values())
mode = [salary for salary, count in freq.items() if count == max_count]

print(mode)
```

```
[477000.0]
```

In [ ]:
```
map_values = {}
for salary in salaries:
  if salary not in map_values:
    map_values[salary] = 1
    continue
  map_values[salary] += map_values[salary] + 1

print(map_values)
```

```
{844000.0: 1, 758000.0: 3, 421000.0: 1, 259000.0: 1, 511000.0: 1, 405000.0:
1, 784000.0: 1, 303000.0: 1, 477000.0: 7, 583000.0: 1, 908000.0: 1, 505000.
0: 1, 282000.0: 1, 756000.0: 1, 618000.0: 1, 251000.0: 1, 910000.0: 1, 98300
0.0: 1, 810000.0: 1, 902000.0: 1, 310000.0: 1, 730000.0: 1, 899000.0: 1, 684
000.0: 1, 472000.0: 1, 101000.0: 1, 434000.0: 1, 611000.0: 1, 913000.0: 1, 9
67000.0: 1, 865000.0: 1, 260000.0: 1, 805000.0: 1, 549000.0: 1, 14000.0: 1,
720000.0: 1, 399000.0: 1, 825000.0: 1, 668000.0: 1, 1000.0: 1, 494000.0: 1,
868000.0: 1, 244000.0: 1, 325000.0: 1, 870000.0: 1, 191000.0: 1, 568000.0:
1, 239000.0: 1, 968000.0: 1, 803000.0: 1, 448000.0: 1, 80000.0: 1, 320000.0:
1, 508000.0: 1, 933000.0: 1, 109000.0: 1, 551000.0: 1, 707000.0: 1, 547000.
0: 1, 814000.0: 1, 540000.0: 1, 964000.0: 1, 603000.0: 1, 588000.0: 1, 44500
0.0: 1, 596000.0: 1, 385000.0: 1, 576000.0: 1, 290000.0: 1, 189000.0: 1, 187
000.0: 1, 613000.0: 3, 657000.0: 1, 90000.0: 1, 877000.0: 1, 923000.0: 3, 84
2000.0: 1, 898000.0: 1, 541000.0: 1, 391000.0: 1, 705000.0: 1, 276000.0: 1,
812000.0: 1, 849000.0: 1, 895000.0: 1, 590000.0: 1, 950000.0: 1, 580000.0:
1, 451000.0: 1, 660000.0: 1, 996000.0: 1, 917000.0: 1, 793000.0: 1, 82000.0:
1, 486000.0: 1}
```

Sample Variance

In [ ]:
```
average = sum(salaries)/len(salaries)
len_minus_one = len(salaries) - 1

x_xmean_squared = []
for salary in salaries:
  temp = (salary - average)**2
  x_xmean_squared.append(temp)

summation = sum(x_xmean_squared)
variance = summation/len_minus_one

print(variance)
```

```
70664054444.44444
```

Sample Standard Deviation

In [ ]:
```
squared_diffs = [(x - mean) ** 2 for x in salaries]
variance = sum(squared_diffs) / (len(salaries) - 1)
std_dev = variance ** 0.5

print(std_dev)
```

```
265827.11382484
```

Exercise 2

In [ ]:
```
import statistics
```

Range

In [ ]:
```
range = max(salaries) - min(salaries)
print(range)
```

```
995000.0
```

Coefficient of variation Interquartile range

In [ ]:
```
data = salaries
mean = statistics.mean(salaries)
std_dev = statistics.stdev(salaries)

print((std_dev - mean)*100)
```

```
-31986288.617516
```

In [ ]:
```
# Interquartile Range

mid_index = n // 2
if n % 2 == 0:
    lower_half = data[:mid_index]
    upper_half = data[mid_index:]
else:
    lower_half = data[:mid_index]
    upper_half = data[mid_index+1:]

Q1 = statistics.median(lower_half)
Q3 = statistics.median(upper_half)

print(Q1 - Q3)
```

```
-17500.0
```

Quartile coefficient of dispersion

In [ ]:
```
mid_index = n // 2
if n % 2 == 0:
    lower_half = data[:mid_index]
    upper_half = data[mid_index:]
else:
    lower_half = data[:mid_index]
    upper_half = data[mid_index+1:]

Q1 = statistics.median(lower_half)
Q3 = statistics.median(upper_half)

print((Q3 - Q1) / (Q3 + Q1))
```

```
0.014976465554129225
```

Exercise 3

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

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

df = pd.read_csv('/content/drive/MyDrive/diabetes.csv')
```

1. Identify the column names

In [ ]:
```
print("Column names:")
print(df.columns.tolist())
print()
```

```
Column names:
['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BM
I', 'DiabetesPedigreeFunction', 'Age', 'Diagnosis', 'Classification']
```

2. Identify the data types of the data

In [ ]:
```
print("Data types:")
print(df.dtypes)
print()
```

```
Data types:
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Diagnosis                   int64
Classification             object
dtype: object
```

3. Display the total number of records

In [ ]:
```
print(f"Total records: {len(df)}")
print()
```

```
Total records: 768
```

4. Display the first 20 records

In [ ]:
```
print("First 20 records:")
print(df.head(20))
print()
```

```
First 20 records:
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0             6      148             72             35        0  33.6
1             1       85             66             29        0  26.6
2             8      183             64              0        0  23.3
3             1       89             66             23       94  28.1
4             0      137             40             35      168  43.1
5             5      116             74              0        0  25.6
6             3       78             50             32       88  31.0
7            10      115              0              0        0  35.3
8             2      197             70             45      543  30.5
9             8      125             96              0        0   0.0
10            4      110             92              0        0  37.6
11           10      168             74              0        0  38.0
12           10      139             80              0        0  27.1
13            1      189             60             23      846  30.1
14            5      166             72             19      175  25.8
15            7      100              0              0        0  30.0
16            0      118             84             47      230  45.8
17            7      107             74              0        0  29.6
18            1      103             30             38       83  43.3
19            1      115             70             30       96  34.6

    DiabetesPedigreeFunction  Age  Diagnosis Classification
0                      0.627   50          1       Diabetes
1                      0.351   31          0    No Diabetes
2                      0.672   32          1       Diabetes
3                      0.167   21          0    No Diabetes
4                      2.288   33          1       Diabetes
5                      0.201   30          0    No Diabetes
6                      0.248   26          1       Diabetes
7                      0.134   29          0    No Diabetes
8                      0.158   53          1       Diabetes
9                      0.232   54          1       Diabetes
10                     0.191   30          0    No Diabetes
11                     0.537   34          1       Diabetes
12                     1.441   57          0    No Diabetes
13                     0.398   59          1       Diabetes
14                     0.587   51          1       Diabetes
15                     0.484   32          1       Diabetes
16                     0.551   31          1       Diabetes
17                     0.254   31          1       Diabetes
18                     0.183   33          0    No Diabetes
19                     0.529   32          1       Diabetes
```

5. Display the last 20 records

In [ ]:
```
print("Last 20 records:")
print(df.tail(20))
print()
```

```
Last 20 records:
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
748            3      187             70             22      200  36.4
749            6      162             62              0        0  24.3
750            4      136             70              0        0  31.2
751            1      121             78             39       74  39.0
752            3      108             62             24        0  26.0
753            0      181             88             44      510  43.3
754            8      154             78             32        0  32.4
755            1      128             88             39      110  36.5
756            7      137             90             41        0  32.0
757            0      123             72              0        0  36.3
758            1      106             76              0        0  37.5
759            6      190             92              0        0  35.5
760            2       88             58             26       16  28.4
761            9      170             74             31        0  44.0
762            9       89             62              0        0  22.5
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis Classification
748                     0.408   36          1       Diabetes
749                     0.178   50          1       Diabetes
750                     1.182   22          1       Diabetes
751                     0.261   28          0    No Diabetes
752                     0.223   25          0    No Diabetes
753                     0.222   26          1       Diabetes
754                     0.443   45          1       Diabetes
755                     1.057   37          1       Diabetes
756                     0.391   39          0    No Diabetes
757                     0.258   52          1       Diabetes
758                     0.197   26          0    No Diabetes
759                     0.278   66          1       Diabetes
760                     0.766   22          0    No Diabetes
761                     0.403   43          1       Diabetes
762                     0.142   33          0    No Diabetes
763                     0.171   63          0    No Diabetes
764                     0.340   27          0    No Diabetes
765                     0.245   30          0    No Diabetes
766                     0.349   47          1       Diabetes
767                     0.315   23          0    No Diabetes
```

6. Change the Outcome column to DiagnosisIn [ ]:

```
In [ ]: df.rename(columns={'Outcome': 'Diagnosis'}, inplace=True)
```

7. Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes"

In [ ]:
```
df['Classification'] = np.where(df['Diagnosis'] == 1, 'Diabetes', 'No Diabet
```

8. Create a new dataframe "withDiabetes" that gathers data with diabetes

In [ ]:
```
withDiabetes = df[df['Diagnosis'] == 1].copy()
```

9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes

In [ ]:
```
noDiabetes = df[df['Diagnosis'] == 0].copy()
```

10. Create a new dataframe "Pedia" that gathers data with age 0 to 19

In [ ]:
```
Pedia = df[df['Age'] <= 19].copy()
```

11. Create a new dataframe "Adult" that gathers data with age greater than 19

In [64]:
```
Adult = df[df['Age'] > 19].copy()
```

12. Use numpy to get the average age and glucose value.

In [76]:
```
avg_age = np.mean(df['Age'])
avg_glucose = np.mean(df['Glucose'])
print("Average:")
print(f"Age: {avg_age:.2f}")
print(f"Glucose: {avg_glucose:.2f}")
print()
```

```
Average:
Age: 33.24
Glucose: 120.89
```

13. Use numpy to get the median age and glucose value.

In [75]:
```
med_age = np.median(df['Age'])
med_glucose = np.median(df['Glucose'])
print("Median:")
print(f"Age: {med_age:.1f}")
print(f"Glucose: {med_glucose:.1f}")
print()
```

```
Median:
Age: 29.0
Glucose: 117.0
```

14. Use numpy to get the middle values of glucose and age.

In [72]:
```
mid_age = np.percentile(df['Age'], 50)
mid_glucose = np.percentile(df['Glucose'], 50)
print("Middle Values:")
print(f"Age: {mid_age:.1f}")
print(f"Glucose: {mid_glucose:.1f}")
print()
```

```
Middle Values:
Age: 29.0
Glucose: 117.0
```

15. Use numpy to get the standard deviation of the skinthickness.

In [73]:
```
std_skinthickness = np.std(df['SkinThickness'], ddof=1)
print(f"{std_skinthickness:.2f}")
```

```
15.95
```

Conclusion

- I thought it was gonna be hard with all those instructions but when I actually did it most of it was just simple one liners. Renaming columns, filtering groups, getting averages and medians all easy. The only thing that made me pause was the standard deviation because I almost forgot the ddof=1 part. Other than that it went smooth and I got it done pretty quick.