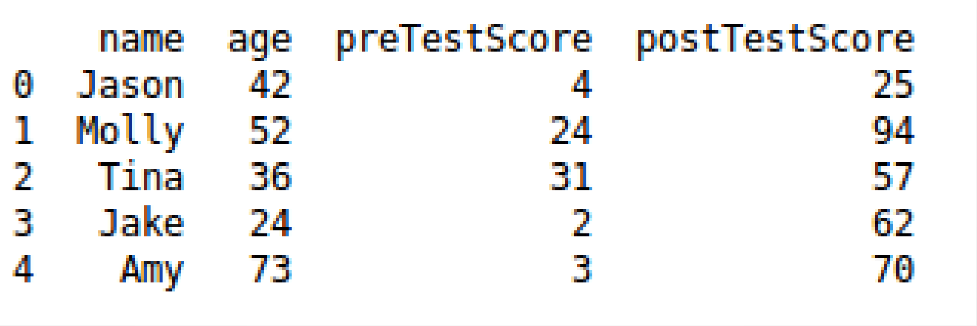
Statistic with pandas

Presented to you by

Dipl. Inform.(FH) Jony Sugianto, M. Comp. Sc.
WA:0812-130-86659
Github: https://github.com/jonysugianto

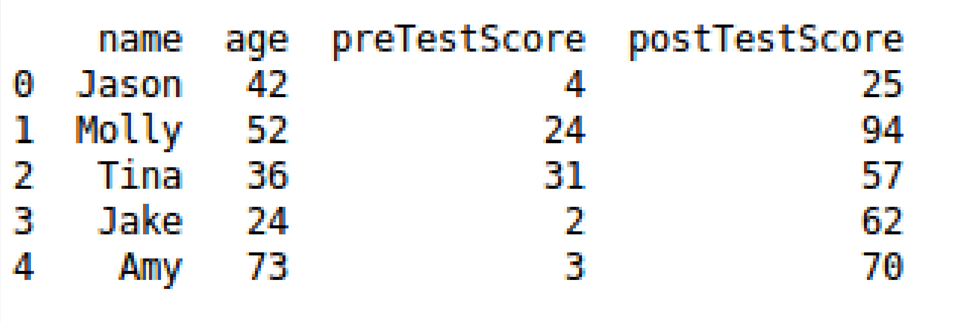
Sample Data

```
import pandas as pd

data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
        'age': [42, 52, 36, 24, 73],
        'preTestScore': [4, 24, 31, 2, 3],
        'postTestScore': [25, 94, 57, 62, 70]}
df = pd.DataFrame(data, columns = ['name', 'age', 'preTestScore', 'postTestScore'],
index=[0,1,2,3,4])
df=df.sort_index()

print()
print(df)
```



```
    name  age  preTestScore  postTestScore
0  Jason   42             4             25
1  Molly   52            24             94
2   Tina   36            31             57
3   Jake   24             2             62
4    Amy   73             3             70
```

Sum, Cumulative Sum

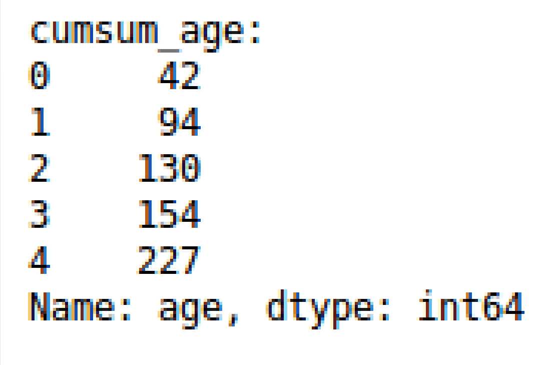


```
    name  age  preTestScore  postTestScore
0  Jason   42             4             25
1  Molly   52            24             94
2   Tina   36            31             57
3   Jake   24             2             62
4    Amy   73             3             70
```

```
sum_age=df['age'].sum()
print(‘sum_age=’, sum_age)
```

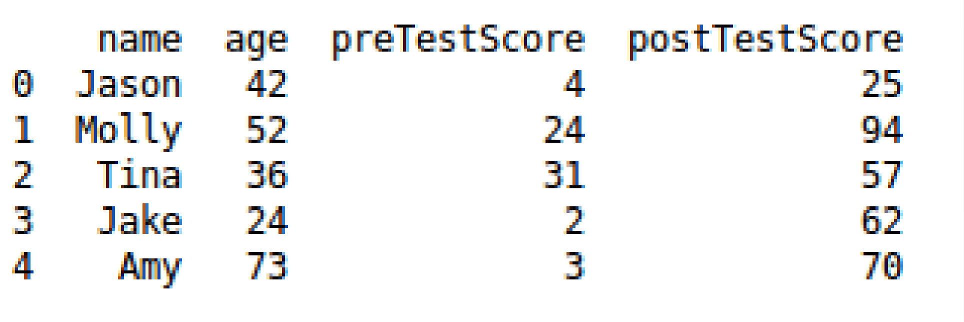
sum_age=227

```
cumsum_age=df['age'].cumsum()
print('cumsum_age:')
print(cumsum_age)
```



```
cumsum_age:
0     42
1     94
2    130
3    154
4    227
Name: age, dtype: int64
```

Min,Max,Mean



```
    name  age  preTestScore  postTestScore
0  Jason   42             4             25
1  Molly   52            24             94
2   Tina   36            31             57
3   Jake   24             2             62
4    Amy   73             3             70
```

```
min_age=df['age'].min()
print('min_age:', min_age)
```

min_age=24

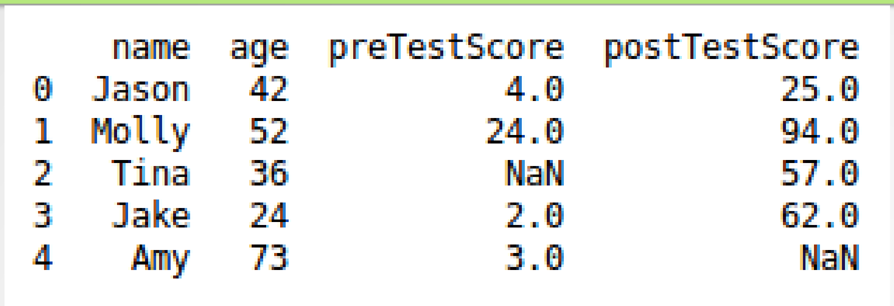
```
max_age=df['age'].max()
print('max_age:', max_age)
```

max_age=73

```
mean_age=df['age'].mean()
print('mean_age:', mean_age)
```

mean_age=45.4

Shape Dataframe, Count



```
    name  age  preTestScore  postTestScore
0  Jason   42           4.0           25.0
1  Molly   52          24.0           94.0
2   Tina   36           NaN           57.0
3   Jake   24           2.0           62.0
4    Amy   73           3.0            NaN
```

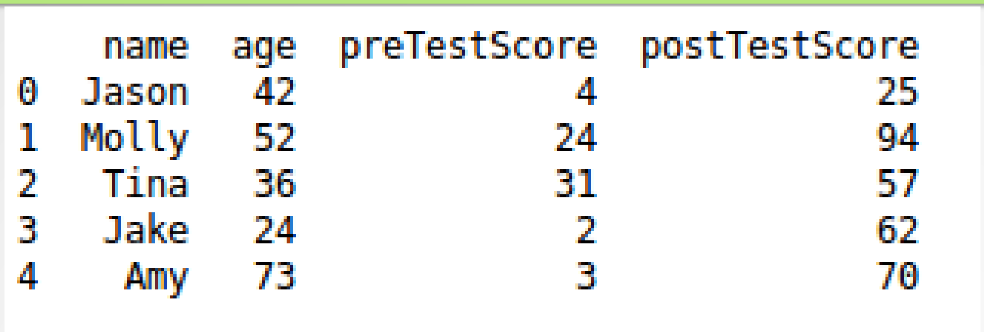
```
print('shape:',df.shape)
```

shape=(5,4)

```
count_preTestScore=df['preTestScore'].count()
print('count_preTestScore:',count_preTestScore)
```

count_preTestScore: 4

Median

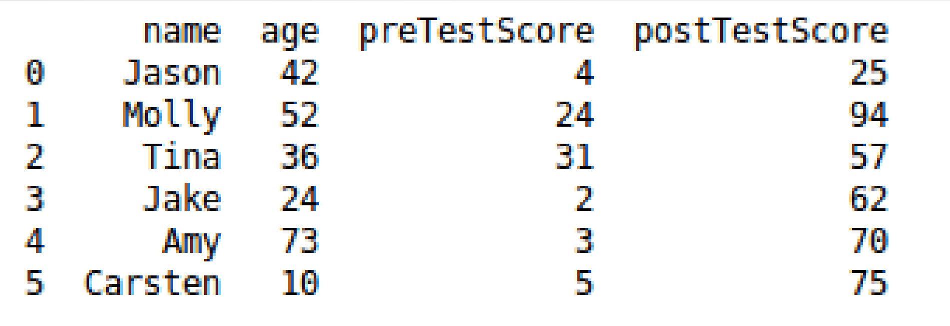


```
    name  age  preTestScore  postTestScore
0  Jason   42             4             25
1  Molly   52            24             94
2   Tina   36            31             57
3   Jake   24             2             62
4    Amy   73             3             70
```

```
median_age=df['age'].median()
print('median_age:',median_age)
```

median_age: 42.0

Median



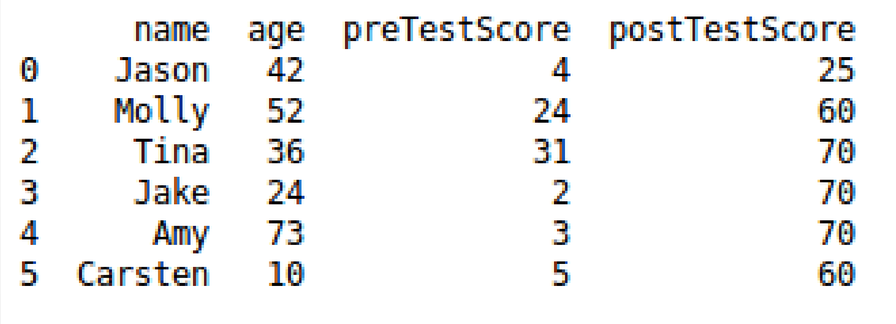
```
      name  age  preTestScore  postTestScore
0    Jason   42             4             25
1    Molly   52            24             94
2     Tina   36            31             57
3     Jake   24             2             62
4      Amy   73             3             70
5  Carsten   10             5             75
```

```
median_age=df['age'].median()
print('median_age:',median_age)
```

median_age: 39.0

Mode

```
import pandas as pd

data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy', 'Carsten'],
        'age': [42, 52, 36, 24, 73,10],
        'preTestScore': [4, 24, 31, 2, 3,5],
        'postTestScore': [25, 60, 70, 70, 70, 60]}
df = pd.DataFrame(data, columns = ['name', 'age', 'preTestScore', 'postTestScore'],
index=[0,1,2,3,4,5])

print()
print(df)
```

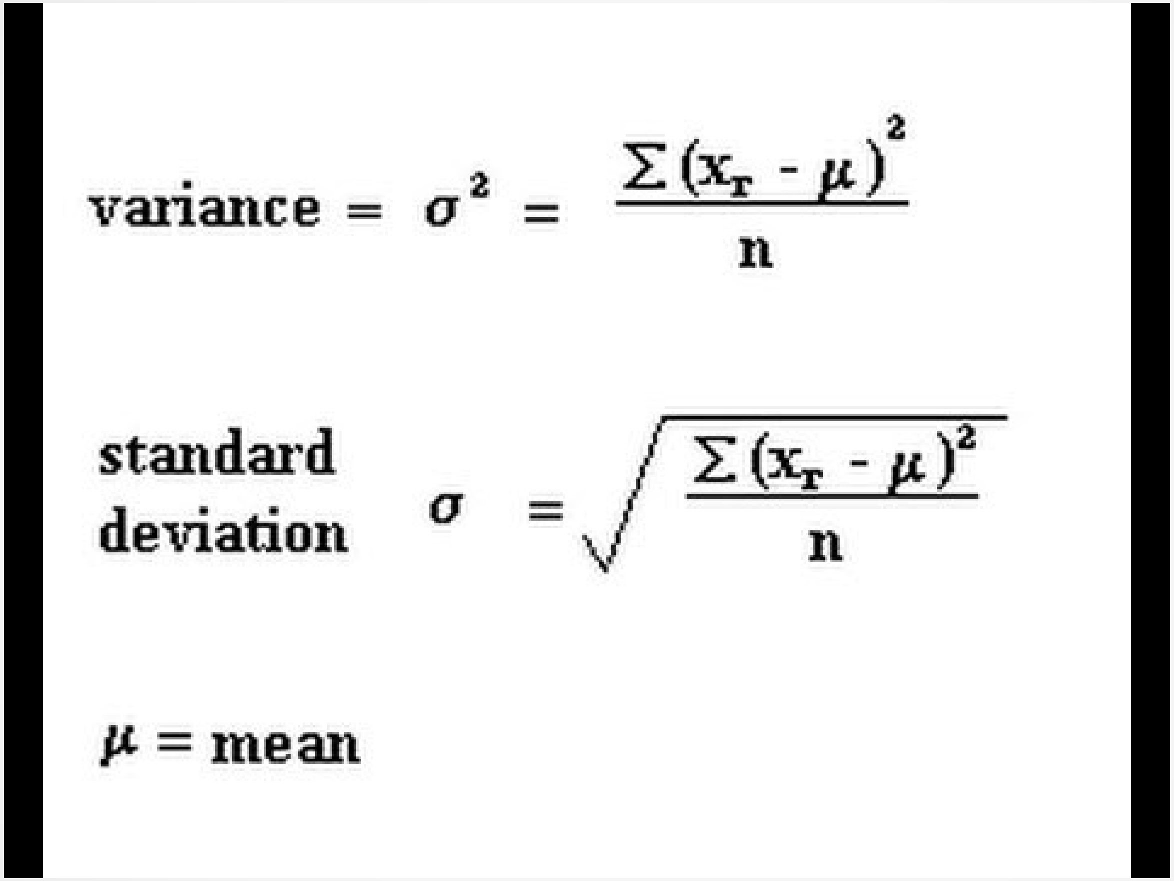


```
      name  age  preTestScore  postTestScore
0    Jason   42             4             25
1    Molly   52            24             60
2     Tina   36            31             70
3     Jake   24             2             70
4      Amy   73             3             70
5  Carsten   10             5             60
```

```
mode_postTestScore=df['postTestScore'].mode()
```

```
mode:
0    70
dtype: int64
```

Variance, Standard Deviation



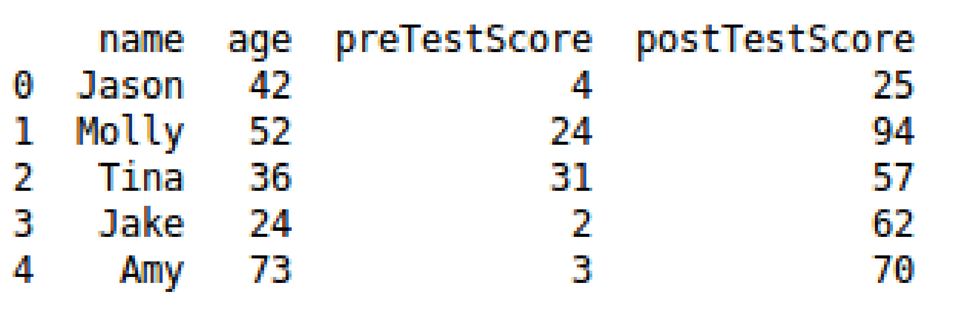
$$\text{variance} = \sigma^2 = \frac{\sum (x_r - \mu)^2}{n}$$

$$\text{standard deviation} \quad \sigma = \sqrt{\frac{\sum (x_r - \mu)^2}{n}}$$

μ = mean

Variance, Standard Deviation

```
import pandas as pd

data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
        'age': [42, 52, 36, 24, 73],
        'preTestScore': [4, 24, 31, 2, 3],
        'postTestScore': [25, 94, 57, 62, 70]}
df = pd.DataFrame(data, columns = ['name', 'age', 'preTestScore', 'postTestScore'],
index=[0,1,2,3,4])
df=df.sort_index()

print()
print(df)
```



```
    name  age  preTestScore  postTestScore
0  Jason   42             4             25
1  Molly   52            24             94
2   Tina   36            31             57
3   Jake   24             2             62
4    Amy   73             3             70
```

```
var_postTestScore=df['postTestScore'].var()
print('var postTestScore=', var_postTestScore)
```

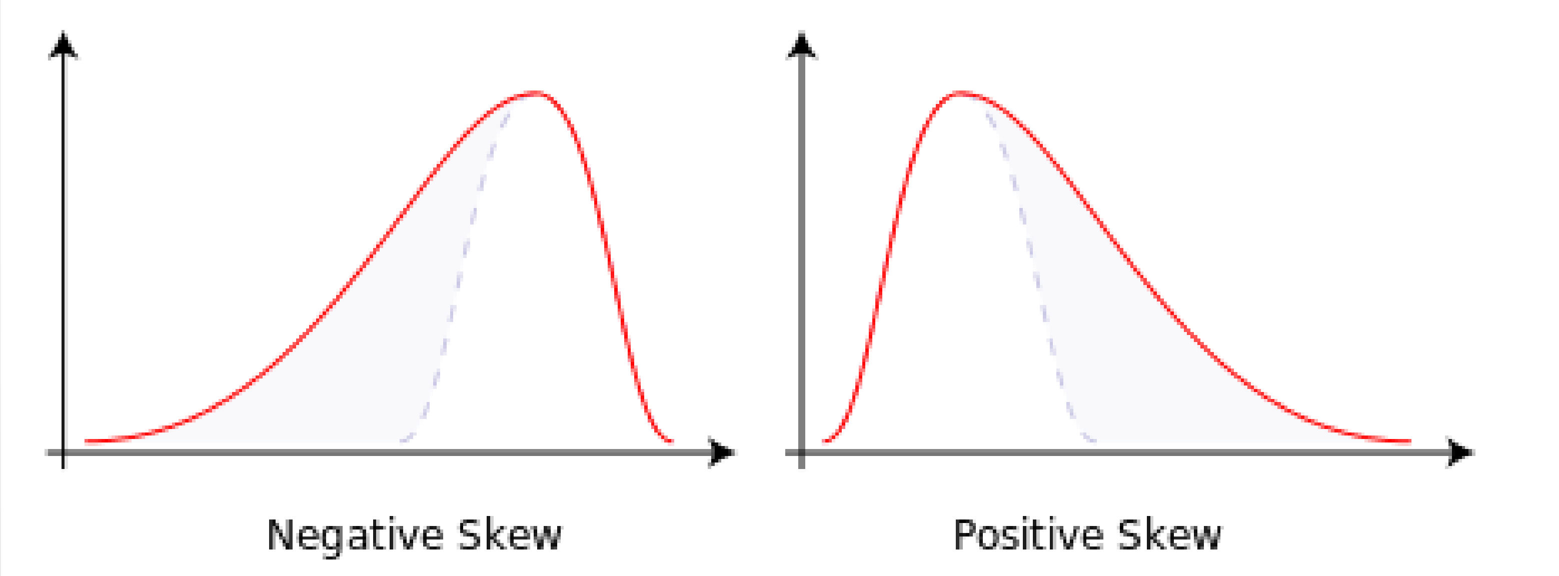
```
var postTestScore= 620.3
```

```
std_postTestScore=df['postTestScore'].std()
print('std postTestScore=', std_postTestScore)
```

```
std postTestScore= 24.90582261239327
```

Skewness

$$\gamma_1 = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$$



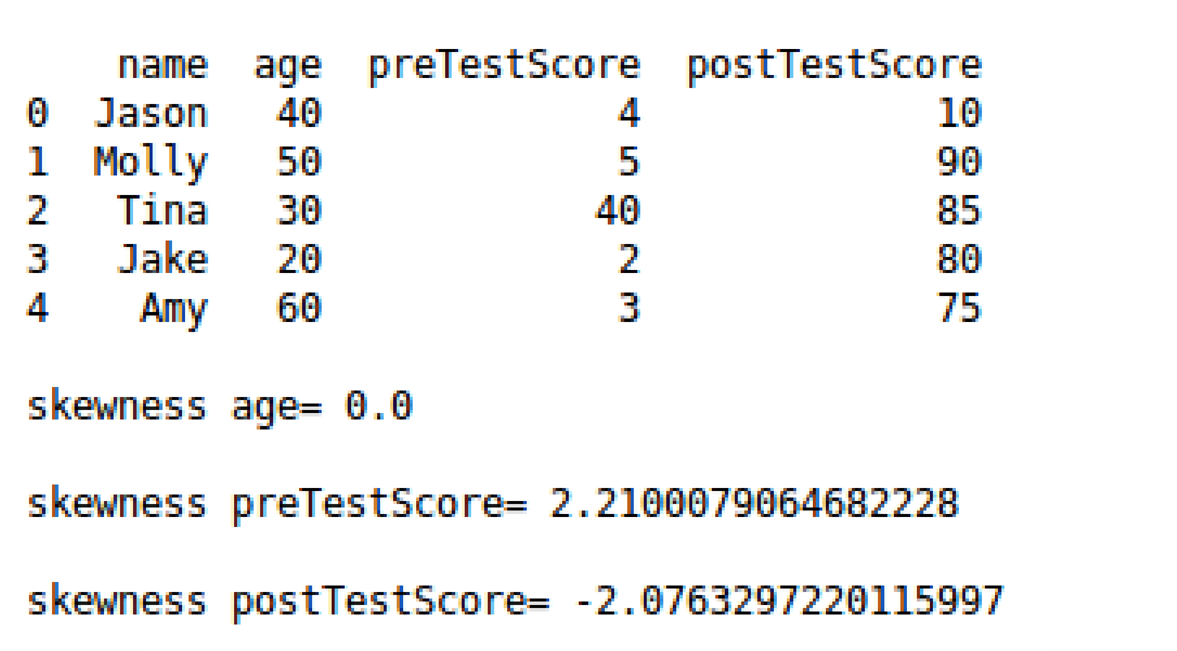
Negative Skew

Positive Skew

Skewness

```
data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
        'age': [40, 50, 30, 20, 60],
        'preTestScore': [4, 5, 40, 2, 3],
        'postTestScore': [10, 90, 85, 80, 75]}
df = pd.DataFrame(data, columns = ['name', 'age', 'preTestScore', 'postTestScore'],
index=[0,1,2,3,4])
df=df.sort_index()

print()
print(df)
print()
sk_age=df['age'].skew()
print('skewness age=', sk_age)
print()
sk_preTestScore=df['preTestScore'].skew()
print('skewness preTestScore=', sk_preTestScore)
print()
sk_postTestScore=df['postTestScore'].skew()
print('skewness postTestScore=', sk_postTestScore)
print()
```

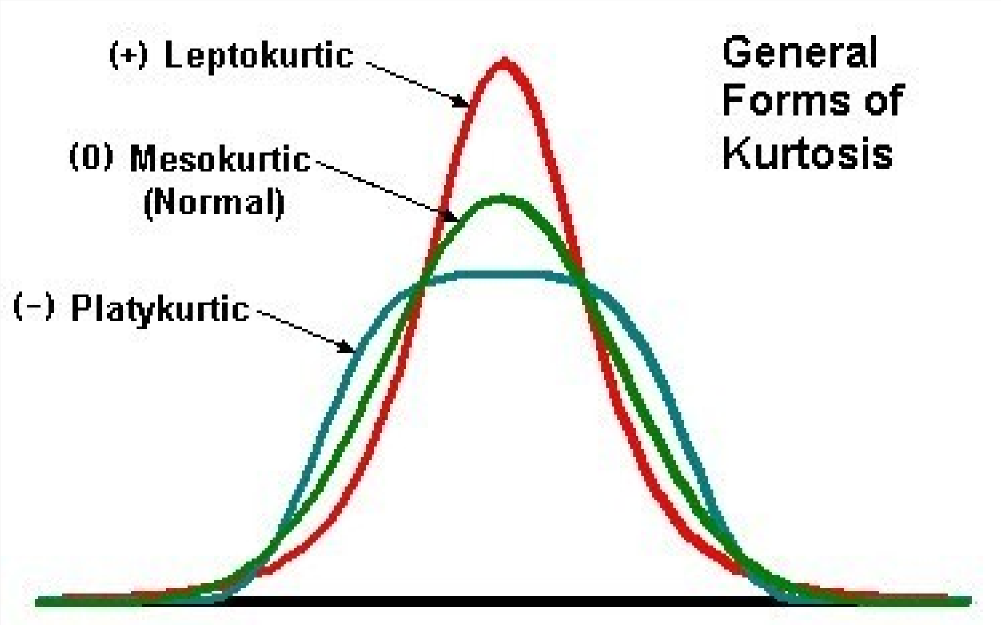
Skewness



```
    name  age  preTestScore  postTestScore
0  Jason   40             4             10
1  Molly   50             5             90
2   Tina   30            40             85
3   Jake   20             2             80
4    Amy   60             3             75

skewness age= 0.0

skewness preTestScore= 2.2100079064682228

skewness postTestScore= -2.0763297220115997
```

Kurtosis

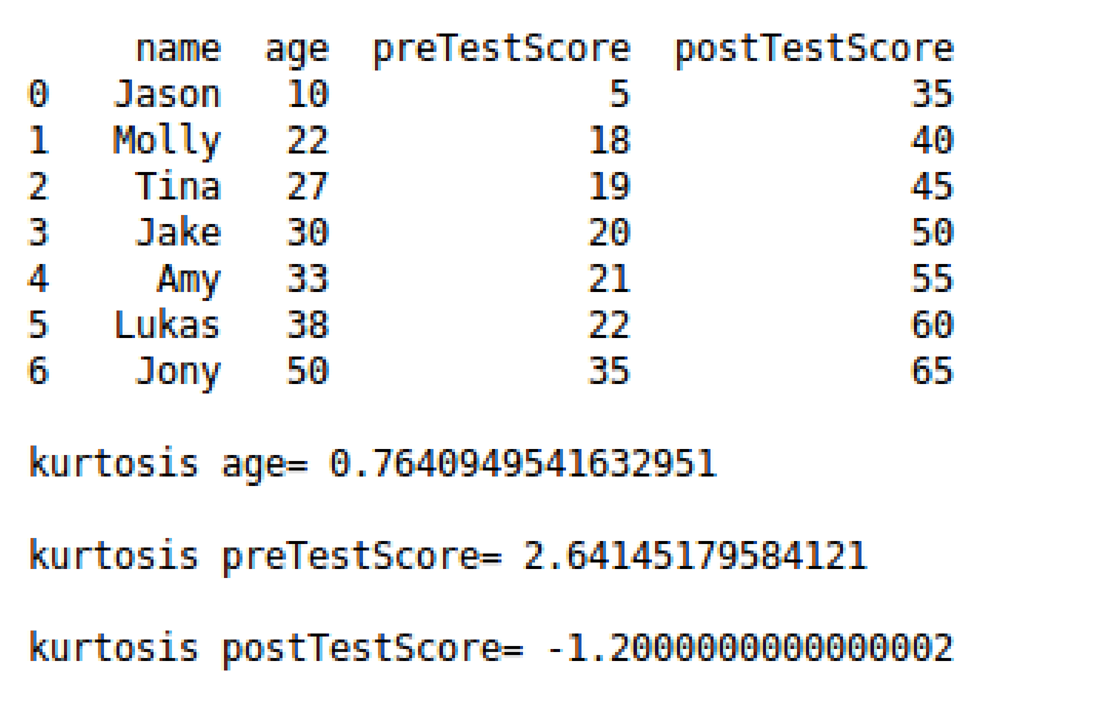
$$\mathrm{Kurt}[X] = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right]$$



Kurtosis

```
import pandas as pd

data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy',' Lukas', 'Jony'],
        'age': [10, 22, 27, 30, 33, 38, 50],
        'preTestScore': [5,18, 19, 20, 21, 22,35],
        'postTestScore': [35, 40, 45, 50, 55, 60, 65]}
df = pd.DataFrame(data, columns = ['name', 'age', 'preTestScore', 'postTestScore'],
index=[0,1,2,3,4,5,6])
df=df.sort_index()

print()
print(df)
print()
kurt_age=df['age'].kurtosis()
print('kurtosis age=', kurt_age)
print()
kurt_preTestScore=df['preTestScore'].kurtosis()
print('kurtosis preTestScore=', kurt_preTestScore)
print()
kurt_postTestScore=df['postTestScore'].kurtosis()
print('kurtosis postTestScore=', kurt_postTestScore)
print()
```

Kurtosis



```
     name  age  preTestScore  postTestScore
0   Jason   10             5             35
1   Molly   22            18             40
2    Tina   27            19             45
3    Jake   30            20             50
4     Amy   33            21             55
5   Lukas   38            22             60
6    Jony   50            35             65

kurtosis age= 0.7640949541632951

kurtosis preTestScore= 2.64145179584121

kurtosis postTestScore= -1.2000000000000002
```

Correlation

Correlation coefficients are used in statistics to measure how strong a relationship is between two variables.

Pearson Correlation Coefficient

$$r = \frac{\mathrm{cov}(x, y)}{\sigma_x . \sigma_y}$$

$$r = \frac{\sum x.y}{\sqrt{\sum x^2 . \sum y^2}}$$

Where

$$x = X - \bar{X}$$

and

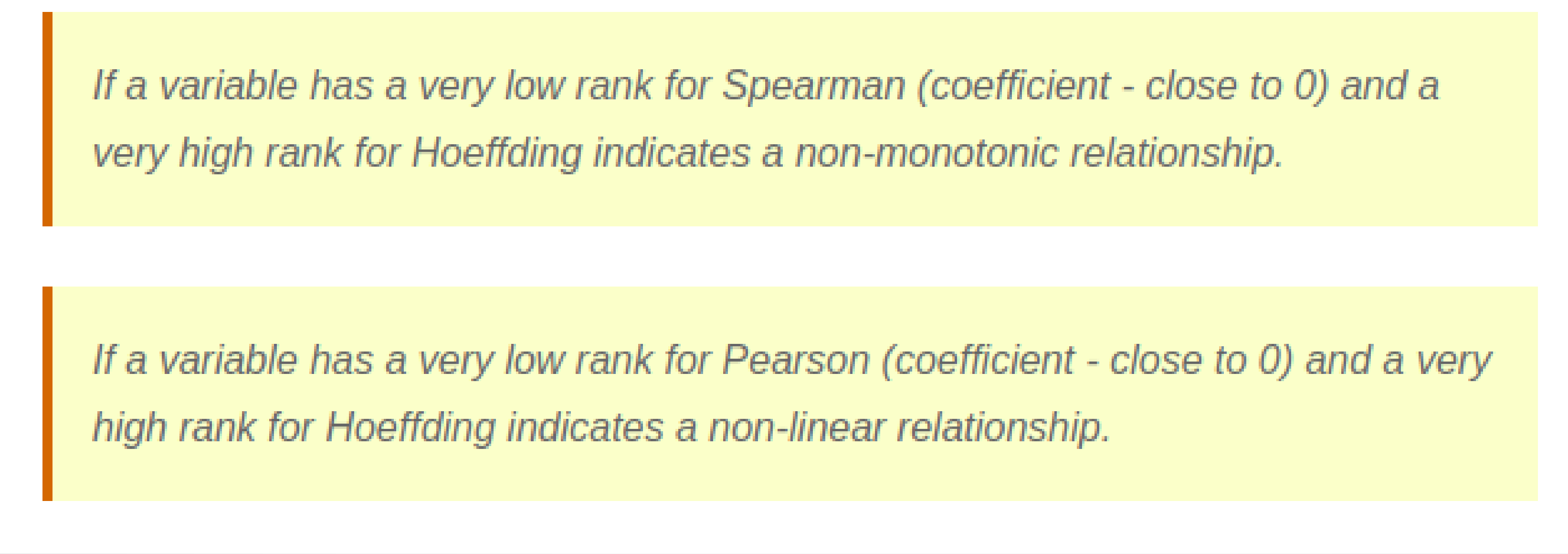
$$y = Y - \bar{Y}$$

Spearman's correlation

Spearman's correlation is a measure of monotonic relationship. It can be used for ordinal variables. It is less sensitive to outliers. If spearman correlation coefficient of a variable is close to 0, it means there is no monotonic relationship between variables.

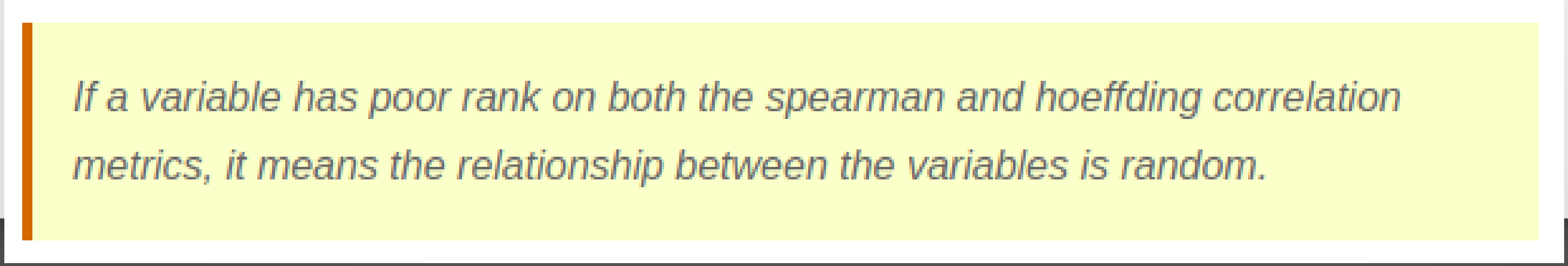
Hoeffding’s D Correlation

Hoeffding’s D correlation is a measure of linear, monotonic and non-monotonic relationship. It has values between –0.5 to 1. The signs of Hoeffding coefficient has no interpretation.



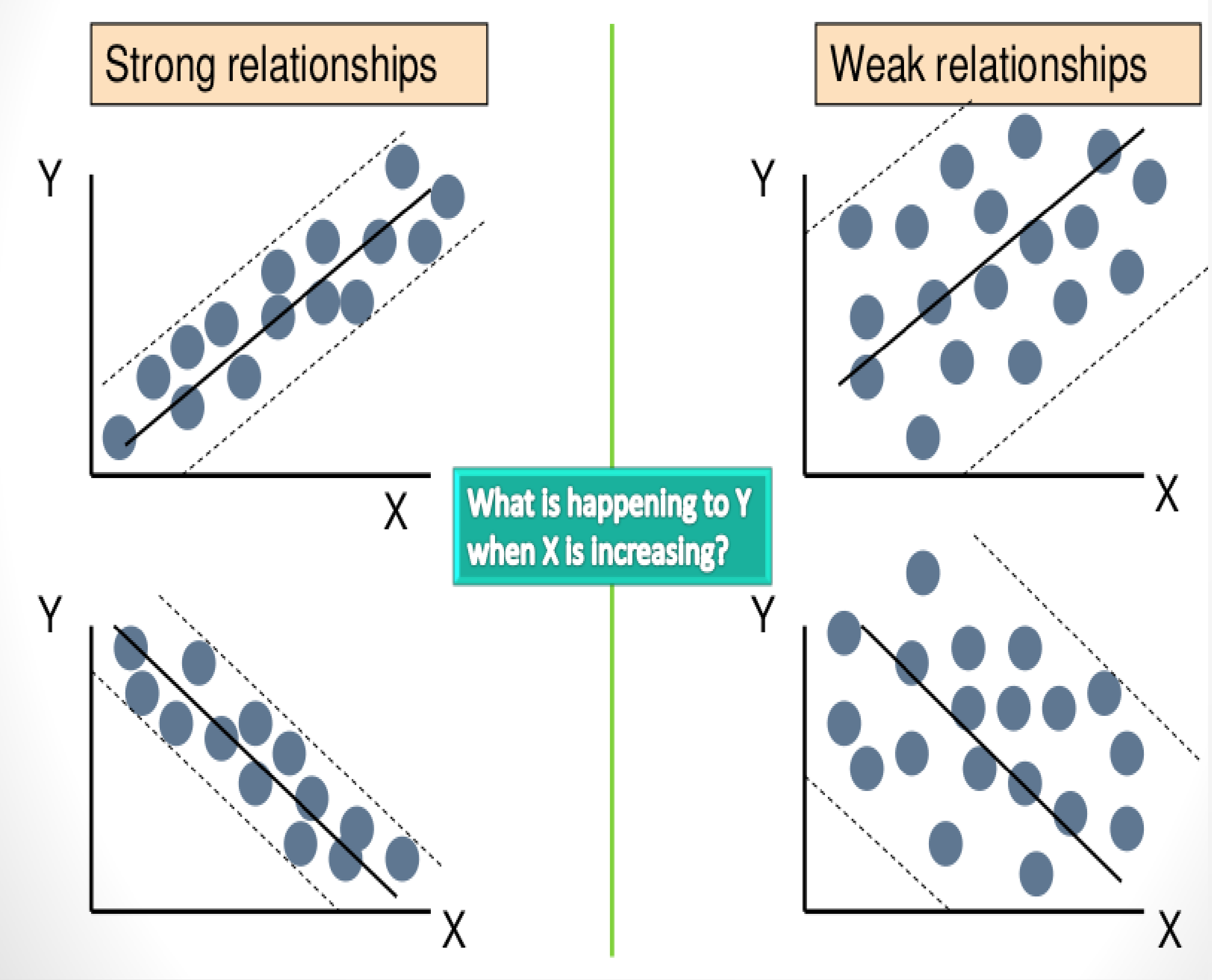
> *If a variable has a very low rank for Spearman (coefficient - close to 0) and a very high rank for Hoeffding indicates a non-monotonic relationship.*

> *If a variable has a very low rank for Pearson (coefficient - close to 0) and a very high rank for Hoeffding indicates a non-linear relationship.*

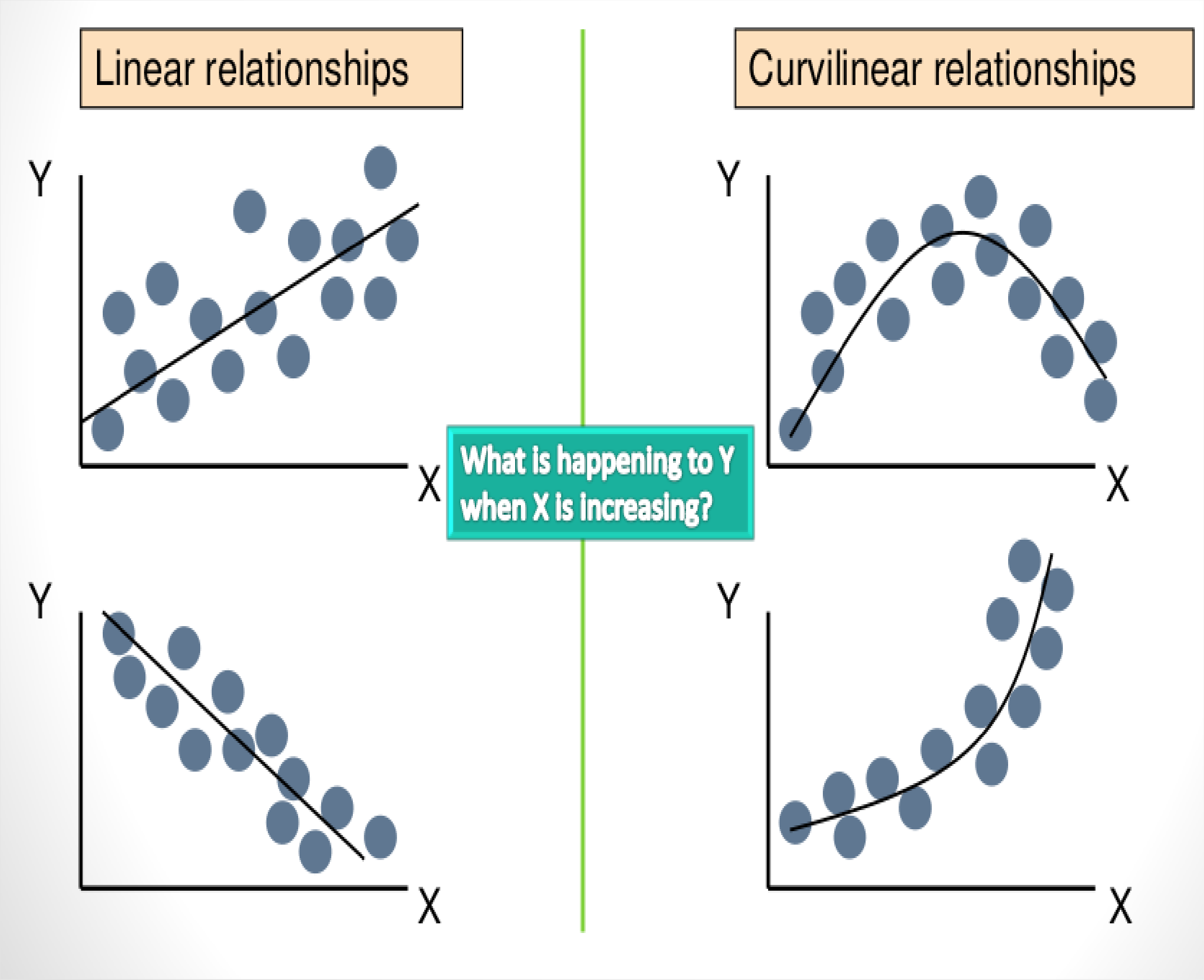


> *If a variable has poor rank on both the spearman and hoeffding correlation metrics, it means the relationship between the variables is random.*

Type of relationship



Type of relationship



Correlation

High positive correlation

Zero correlation

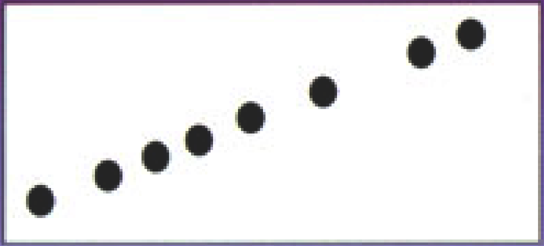
High negative correlation

stronger ↕ weaker

weaker ↕ stronger

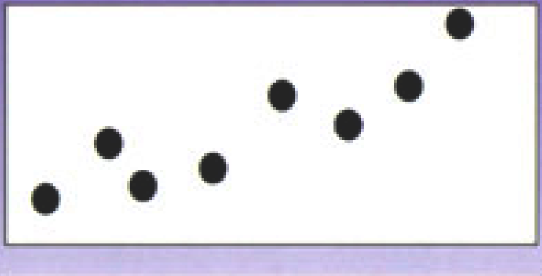
+1.00 **perfect positive**

as one event increases, the second exactly increases



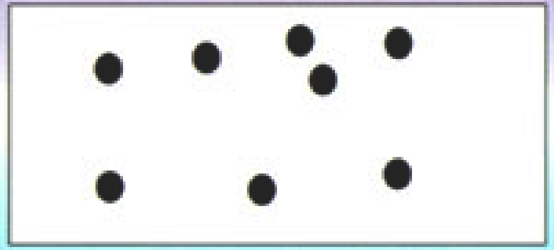
+.50 **positive**

as one event increases, the second sometimes increases



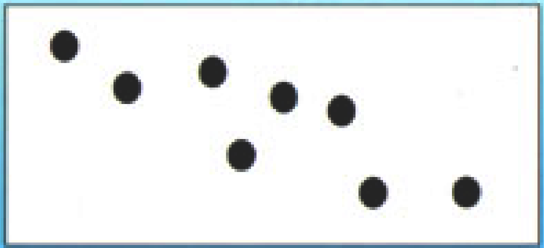
0 **zero correlation**

no relationship between the events



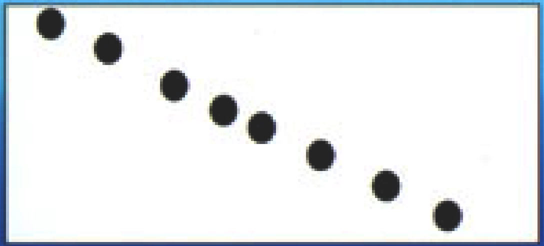
-.50 **negative**

as one event increases, the second sometimes decreases



-1.00 **perfect negative**

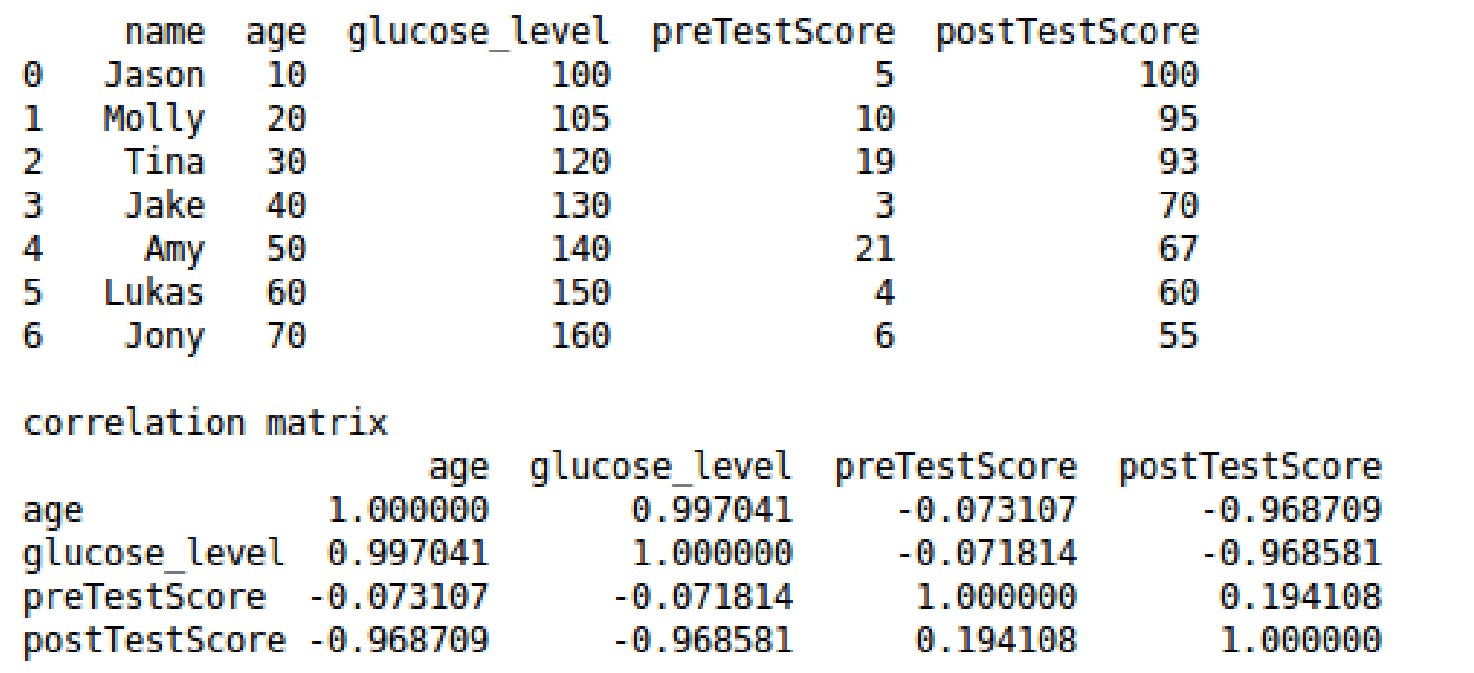
as one event increases, the second exactly decreases



Correlation matrix example code

```
import pandas as pd

data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy',' Lukas', 'Jony'],
        'age': [10, 20, 30, 40, 50, 60, 70],
        'glucose_level':[100,105,120,130,140,150,160],
        'preTestScore': [5, 10, 19, 3, 21, 4, 6],
        'postTestScore': [100, 95, 93, 70, 67, 60, 55]}
df = pd.DataFrame(data,
                  columns = ['name', 'age', 'glucose_level', 'preTestScore', 'postTestScore'],
                  index=[0,1,2,3,4,5,6])
df=df.sort_index()

print()
print(df)
print()
corr_coeffs=df.corr(method='pearson')
print('correlation matrix')
print(corr_coeffs)
```

Correlation matrix example



```
    name  age  glucose_level  preTestScore  postTestScore
0  Jason   10            100             5            100
1  Molly   20            105            10             95
2   Tina   30            120            19             93
3   Jake   40            130             3             70
4    Amy   50            140            21             67
5  Lukas   60            150             4             60
6   Jony   70            160             6             55

correlation matrix
                    age  glucose_level  preTestScore  postTestScore
age            1.000000       0.997041     -0.073107      -0.968709
glucose_level  0.997041       1.000000     -0.071814      -0.968581
preTestScore  -0.073107      -0.071814      1.000000       0.194108
postTestScore -0.968709      -0.968581      0.194108       1.000000
```