GBDi
Government Big Data Institute
สถาบันส่งเสริมการวิเคราะห์และบริหารข้อมูลขนาดใหญ่ภาครัฐ (สวข.)

depa

โครงการอบรมหลักสูตร Hand-on Data Science and Machine Learning

Exploratory Data Analysis (EDA) with Python

Navavit Ponganan
Data scientist
Government Big Data Institute (GBDi)

Exploratory Data Analysis (EDA)

- Digging into the data collected.
- Know its basic properties.
- Make less effort in the future analysis.
  - plot, graphics, statistical summary

What is data?

- Data are characteristics or information usually numerical, that are collected through observation. Data are a set of qualitative or quantitative variables.
- Certain data, viewed in an unrelated way, may not contain interesting information, unless they are analyzed as a whole, under a certain approach or hypothesis.

https://en.wikipedia.org/wiki/Data

What is variable?

- A variable is a dataset feature

This is variable

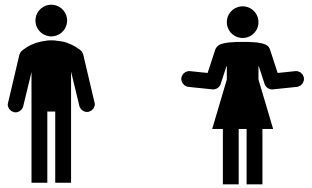
This is data

This is sample

No	Employee ID	First Name	Last Name	Age	Worked years	Salary	Status	Grade
1	1000001	John	Denver	23	1	500	Single	Elementary
2	1000002	Peter	Hank	30	3	900	Married	High School
3	1000003	Jack	Sullivan	27	2	900	Married	High School
4	1000004	Marco	Aurelio	40	8	1500	Married	Master Degree
5	1000005	Claudia	Perez	35	5	1300	Single	Master Degree
6	1000006	Sally	Royal	19	1	1400	Single	Graduate
7	1000007	Peter	Miller	33	4	600	Married	Graduate
8	1000008	Susan	Gordon	35	10	2000	Married	Master Degree

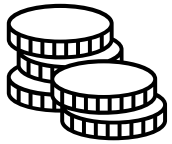
Data types

- Quantitative
  - Weight, height, age, heart rate, body mass index (BMI).
  - Household income, purchases.
- Qualitative
  - Man, Woman.
  - Low income, Average income, High income

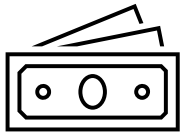


M 1

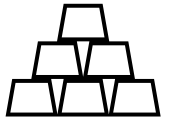
W 2



Low income 1

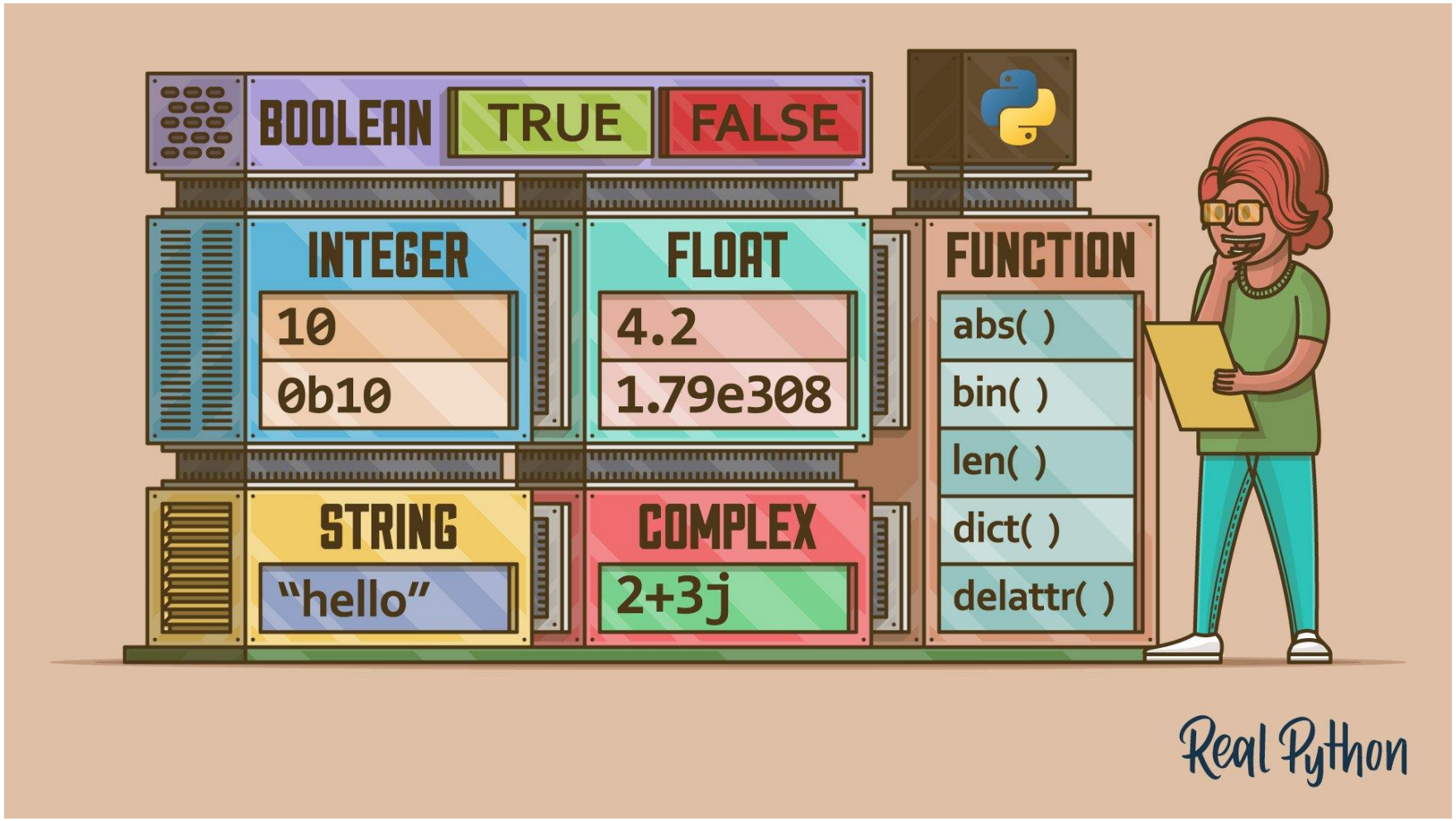


Average income 2



High income 3

Data types in Python



Lab1_Datatypes

Data types in Python

- Lists (Mutable)
  - [1, 2, 3, 4]
  - [1, 2, 3, ‘a’, ‘b’, ‘c’]
- Tuple (Immutable)
  - (1, 2, 3, 4)
  - (1, 2, 3, ‘a’, ‘b’, ‘c’)
- Sets
  - {1, 1, 2, 2, 3, 3} = {1, 2, 3}
- Dictionaries
  - {0 : “Men”, 1: “Woman”}

Key Value

Lab2_Datatypes

Data types in Python

- Dataframe

Column are variable or features

Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent
10000	Person	Cash	120000	150000	240	Canada	America
10001	Company	Cash	521400	651750	1043	Japan	Asia
10002	Company	Credit Card	451000	563750	902	Mexico	America
10003	Company	Transfer	565000	706250	1130	Spain	Europe
10004	Person	Transfer	512300	640375	1024	Argentina	America
10005	Person	Transfer	415500	519375	0	Canada	America
10006	Company	Credit Card	696300	870375	1392	EEUU	America
10007	Person	Cash	741000	926250	1482	Chile	America
10008	Company	Cash	541000	676250	1082	EEUU	America
10009	Company	Cash	83000	103750	166	EEUU	America
10010	Company	Cash	454100	567925	910	EEUU	America
10011	Person	Transfer	520033	650041	1041	EEUU	America
10012	Person	Credit Card	452000	565000	904	Canada	America
10013	Person	Transfer	352000	440000	704	Germany	Europe
10014	Company	Transfer	241010	301262	480	EEUU	America
10015	Company	Credit Card	560122	700152	1120	Mexico	America
10016	Person	Credit Card	362200	452750	0	Canada	America
10017	Person	Cash	452230	565287	903	Japan	Asia
10018	Company	Cash	521000	651250	1042	Spain	Europe

Rows are samples

Lab3_DataFrame

Quantitative Data Properties

- Trend.
- Dispersion.
- Shape.

Quantitative Data Properties

- Trend.
  - Arithmetic mean

No	Employee ID	First Name	Last Name	Age	Worked years	Salary	Status	Grade
1	1000001	John	Denver	23	1	500	Single	Elementary
2	1000002	Peter	Hank	30	3	900	Married	High School
3	1000003	Jack	Sullivan	27	2	900	Married	High School
4	1000004	Marco	Aurelio	40	8	1500	Married	Master Degree
5	1000005	Claudia	Perez	35	5	1300	Single	Master Degree

$$\bar{x} = \frac{\sum_{i=1}^{n} X_i}{n}$$

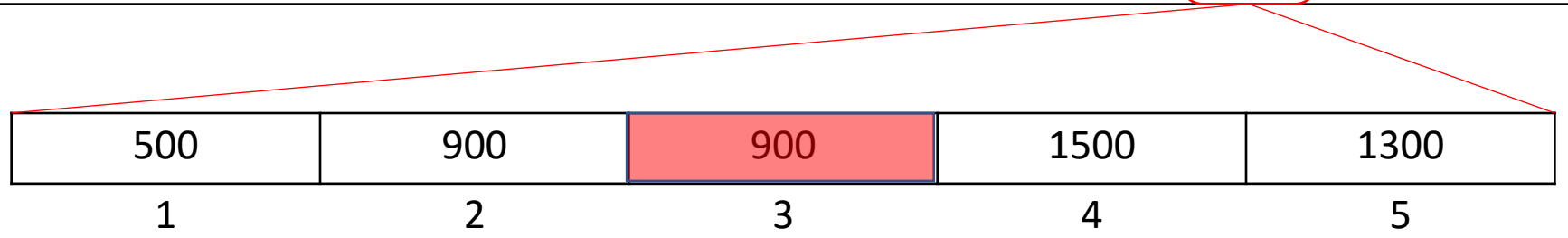
$$\text{Average salary} = \frac{500+900+900+1500+1300}{5}$$

$$= 1020$$

Quantitative Data Properties

- Trend.
  - Median (case 1)

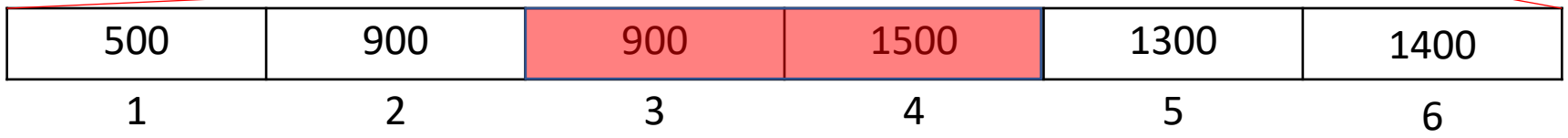
No	Employee ID	First Name	Last Name	Age	Worked years	Salary	Status	Grade
1	1000001	John	Denver	23	1	500	Single	Elementary
2	1000002	Peter	Hank	30	3	900	Married	High School
3	1000003	Jack	Sullivan	27	2	900	Married	High School
4	1000004	Marco	Aurelio	40	8	1500	Married	Master Degree
5	1000005	Claudia	Perez	35	5	1300	Single	Master Degree



Quantitative Data Properties

- Trend.
  - Median (case 2)

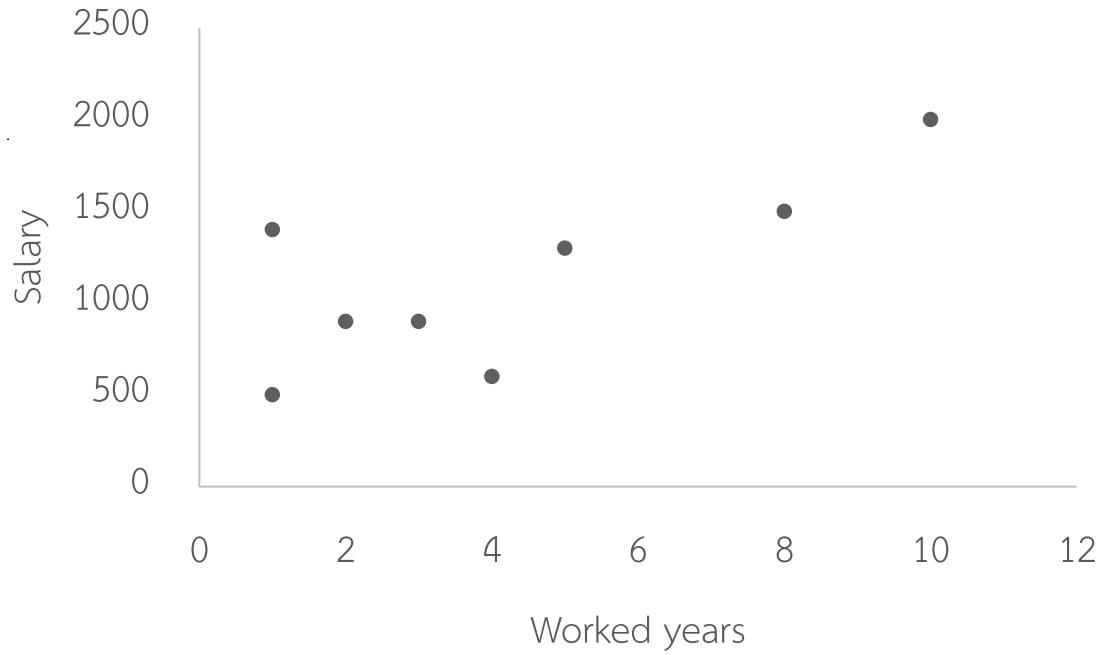
No	Employee ID	First Name	Last Name	Age	Worked years	Salary	Status	Grade
1	1000001	John	Denver	23	1	500	Single	Elementary
2	1000002	Peter	Hank	30	3	900	Married	High School
3	1000003	Jack	Sullivan	27	2	900	Married	High School
4	1000004	Marco	Aurelio	40	8	1500	Married	Master Degree
5	1000005	Claudia	Perez	35	5	1300	Single	Master Degree
6	1000006	Sally	Royal	19	1	1400	Single	Graduate



$$\text{Median} = \frac{900+1500}{2} = 1200$$

Quantitative Data Properties

- Dispersion.
  - Standard Deviation.
  - Low value indicate closeness.
  - High value indicate farness



Quantitative Data Properties

Why we should be concerned about data dispersion?

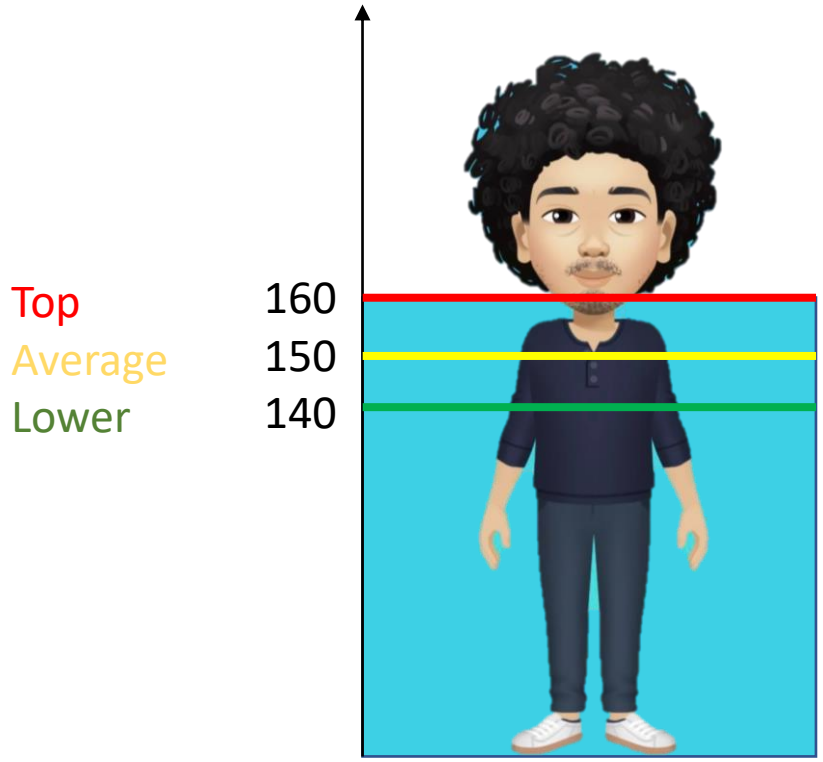
Depth average = 150 cm

Variation average = 10 cm

150+10

150-10

Between 140 – 160 cm



Quantitative Data Properties

Why we should be concerned about data dispersion?

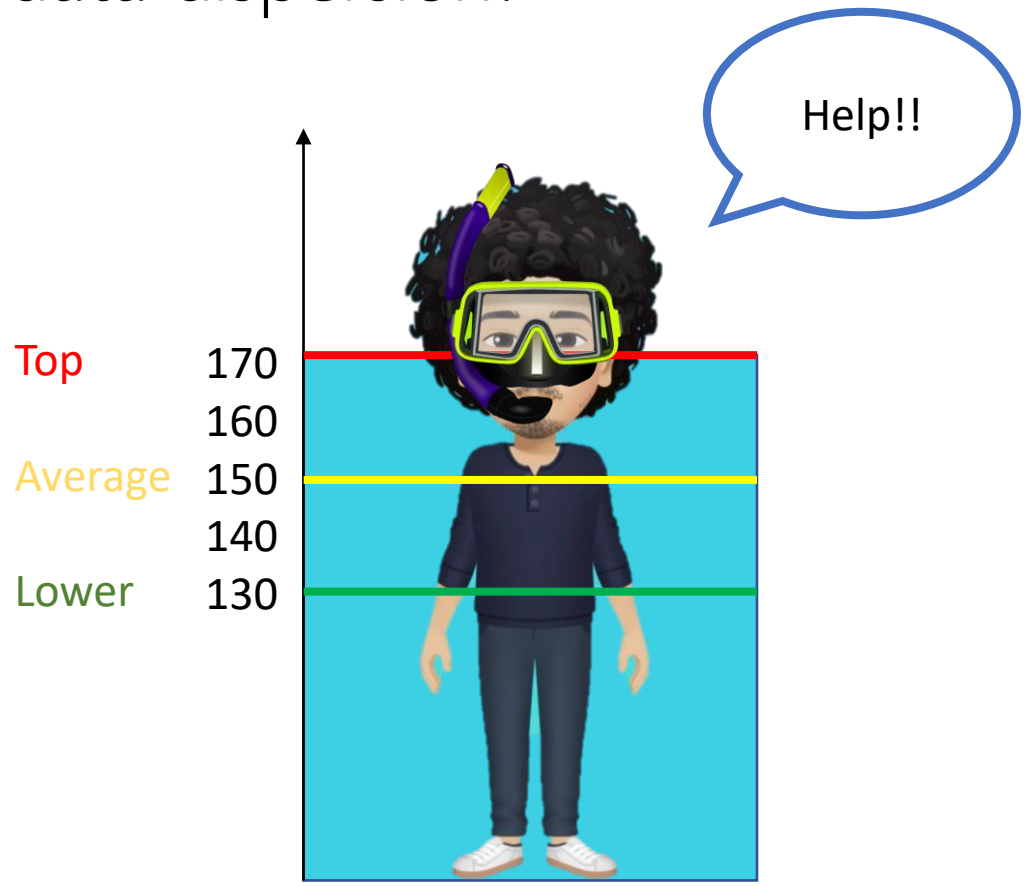
Depth average = 150 cm

Variation average = 20 cm

150+20

150-20

Between 130 – 170 cm



Quantitative Data Properties

Variance:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{X})^2}{n-1}$$

x_i	$\bar{x}$	$x_i - \bar{x}$	$(x_i - \bar{x})^2$
Salary	Average	Difference	Square Difference
500	1020	-520	270400
900	1020	-120	14400
900	1020	-120	14400
1500	1020	480	230400
1300	1020	280	78400
		0	608000

$$s^2 = \frac{608000}{5-1} = 152000$$

Quantitative Data Properties

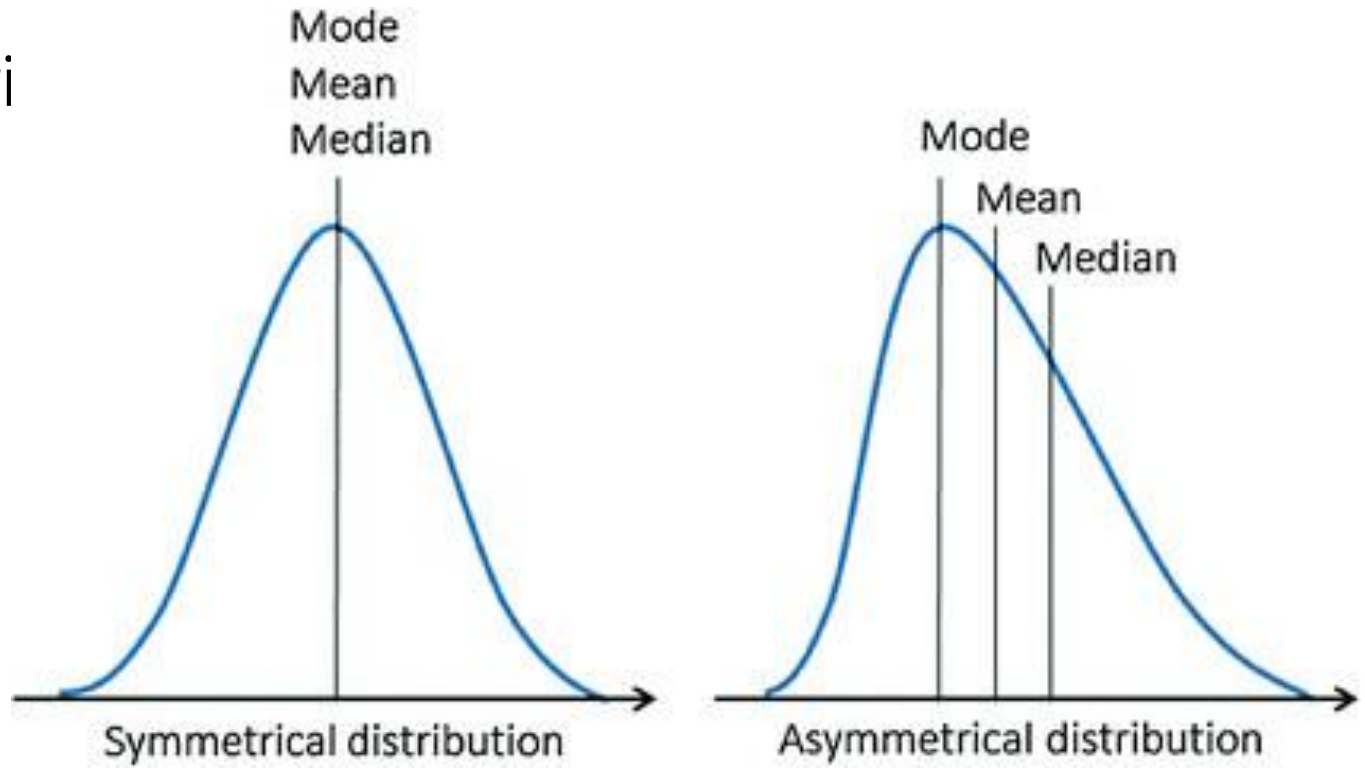
Standard Deviation:

$$S = \sqrt{s^2} \qquad S = \sqrt{152000} \approx 390$$

The salaries move between ± 390 respect to the mean.

Quantitative Data Properties

- Shape.
  - Symmetric
  - Asymmetri



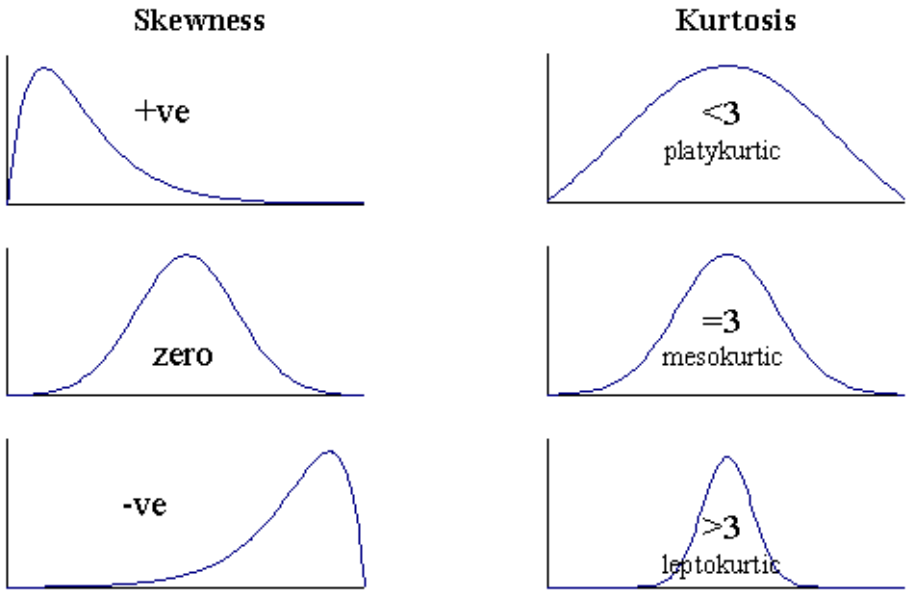
Quantitative Data Properties

- Skewness

$$\frac{1}{n} = \frac{\sum_{i=1}^{n}(x_i - \bar{X})^3}{s^3}$$

- Kurtosis

$$\frac{1}{n} = \frac{\sum_{i=1}^{n}(x_i - \bar{X})^4}{s^4}$$



Quantitative Data Properties

Moment number	Name	Measure of	Formula
1	Mean	Central tendency	$\bar{X} = \frac{\sum_{i=1}^{N} X_i}{N}$
2	Variance (Volatility)	Dispersion	$\sigma^2 = \frac{\sum_{i=1}^{N} (X_i - \bar{X})^2}{N}$
3	Skewness	Symmetry (Positive or Negative)	$Skew = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{(X_i - \bar{X})}{\sigma} \right]^3$
4	Kurtosis	Shape (Tall or flat)	$Kurt = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{(X_i - \bar{X})}{\sigma} \right]^4$

Where X is a random variable having N observations (i = 1,2,…,N).

Lab4_DataProperties

Pre-Processing Data in Python

- Missing Values.
- Data Format.
- Data Normalization.
- Grouping into Classes.
- Converting Categorical Variables to Numeric Variables.

Pre-Processing Data

- It is the process of converting or mapping data from a raw format to more manageable format for later analysis.
- Also known as “Data Cleaning”

Missing Values

- A data entry that is left empty in the data set.
- It can be reflected with “?”, Zero or a blank cell.

	Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent
0	10000	Person	Cash	120000.0	150000.0	240	Canada	America
1	10001	Company	Cash	NaN	651750.0	1043	Japan	Asia
2	10002	Company	Credit Card	451000.0	563750.0	902	Mexico	America
3	10003	Company	Transfer	565000.0	706250.0	1130	Spain	Europe
4	10004	Person	Transfer	512300.0	NaN	1024	Argentina	America

Missing Values

- Common Strategies
  - Review source data and fill in missing values.
  - Remove missing values.
    - Delete the entire variable.
    - Delete the entry with the missing data.
  - Replace missing values.
    - Replace with the average of the entire variable.
    - Replace with mode if categorical variable.
    - Replace based on collected data.
  - Stay with missing values.

Missing Values in Python

- Remove missing values in Python.

	Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent
0	10000	Person	Cash	120000.0	150000.0	240	Canada	America
1	10001	Company	Cash	NaN	651750.0	1043	Japan	Asia
2	10002	Company	Credit Card	451000.0	563750.0	902	Mexico	America
3	10003	Company	Transfer	565000.0	706250.0	1130	Spain	Europe
4	10004	Person	Transfer	512300.0	NaN	1024	Argentina	America

- dropna()
  - axis = 0
  - Delete whole row

```
df_test.dropna(subset=["Purchases"], axis=0, inplace = True)
df_test.head()
```

	Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent
0	10000	Person	Cash	120000.0	150000.0	240	Canada	America
2	10002	Company	Credit Card	451000.0	563750.0	902	Mexico	America
3	10003	Company	Transfer	565000.0	706250.0	1130	Spain	Europe
4	10004	Person	Transfer	512300.0	NaN	1024	Argentina	America
5	10005	Person	Transfer	415500.0	519375.0	0	Canada	America

  - axis = 1
  - Remove entire column

Missing Values in Python

- Remove missing values in Python.

- replace()

```
my_mean = df_test["Sales"].mean()
df_test["Sales"].replace(np.nan, int(my_mean), inplace = True)
df_test.head()
```

	Customer	Customer Type	Payment Type	Purchases	Sales
0	10000	Person	Cash	120000.0	150000.0
2	10002	Company	Credit Card	451000.0	563750.0
3	10003	Company	Transfer	565000.0	706250.0
4	10004	Person	Transfer	512300.0	NaN

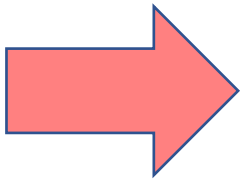
	Customer	Customer Type	Payment Type	Purchases	Sales
0	10000	Person	Cash	120000.0	150000.0
2	10002	Company	Credit Card	451000.0	563750.0
3	10003	Company	Transfer	565000.0	706250.0
4	10004	Person	Transfer	512300.0	553509.0

Data Format

- Different origins usually lead to different formats.

- A common data format makes data easy to compare and interpret.

Gender
0
1
Man
Woman
Male
Female
M
F



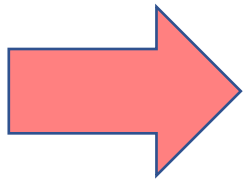
Gender
Male
Female
Male
Female
Male
Female
Male
Female

Data Format

- Converting numeric variables.

```
df_test["Purchases in thousands"] = df_test["Purchases"]/1000
df_test["Sales in thousands"] = df_test["Sales"]/1000
df_test["Refunds in thousands"] = df_test["Refunds"]/1000

df_test.head()
```

	Customer	Customer Type	Payment Type	Purchases	Sales
0	10000	Person	Cash	120000.0	150000.0
2	10002	Company	Credit Card	451000.0	563750.0
3	10003	Company	Transfer	565000.0	706250.0
4	10004	Person	Transfer	512300.0	553509.0
5	10005	Person	Transfer	415500.0	519375.0



Purchases in thousands	Sales in thousands	Refunds in thousands
120.0	150.000	0.240
451.0	563.750	0.902
565.0	706.250	1.130
512.3	553.509	1.024
415.5	519.375	0.000

Data Format

- Correcting data type.
- dtype()

```
df_test.dtypes
```

```
Customer                    int64
Customer Type              object
Payment Type               object
Purchases                 float64
Sales                     float64
Refunds                     int64
Country                    object
Continent                  object
Purchases in thousands    float64
Sales in thousands        float64
Refunds in thousands      float64
dtype: object
```

- astype()

```
df_test = df_test.astype({'Customer':'object'})
df_test.dtypes
```

```
Customer                    object
Customer Type              object
Payment Type               object
Purchases                 float64
Sales                     float64
Refunds                     int64
Country                    object
Continent                  object
Purchases in thousands    float64
Sales in thousands        float64
Refunds in thousands      float64
dtype: object
```

Normalization

- It consists of putting the data in a similar range to be able to compare them.

	No	Employee ID	First Name	Last Name	Age	Worked years	Salary	Status	Grade
0	1	1000001	John	Denver	23	1	500	Single	Elementary
1	2	1000002	Peter	Hank	30	3	900	Married	High School
2	3	1000003	Jack	Sullivan	27	2	900	Married	High School
3	4	1000004	Marco	Aurelio	40	8	1500	Married	Master Degree
4	5	1000005	Claudia	Perez	35	5	1300	Single	Master Degree
5	6	1000006	Sally	Royal	19	1	1400	Single	Graduate
6	7	1000007	Peter	Miller	33	4	600	Married	Graduate
7	8	1000008	Susan	Gordon	35	10	2000	Married	Master Degree

Normalization

- Data Normalization Methods.
  - Simple Feature Scaling.

$$x_{new} = \frac{x_{current}}{x_{maximum}} \qquad 0 \leq x_{new} \leq 1$$

  - Min-Max.

$$x_{new} = \frac{x_{current} - x_{minimum}}{x_{maximum} - x_{minimum}} \qquad 0 \leq x_{new} \leq 1$$

  - Z-score.

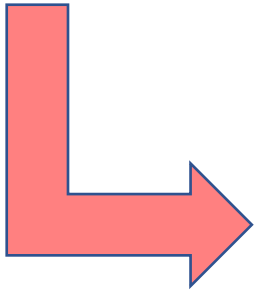
$$x_{new} = \frac{x_{current} - Mean}{Standard\ Deviation} \qquad -3 \leq x_{new} \leq 3$$

Normalization

- Applying Simple Feature Scaling method in Python.

```
df_employees[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	23	1	500
1	30	3	900
2	27	2	900
3	40	8	1500
4	35	5	1300



```
df_norm1 = df_employees

df_norm1["Age"] = df_norm1["Age"] / df_norm1["Age"].max()
df_norm1["Worked years"] = df_norm1["Worked years"] / df_norm1["Worked years"].max()
df_norm1["Salary"] = df_norm1["Salary"] / df_norm1["Salary"].max()

df_norm1[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	0.575	0.1	0.25
1	0.750	0.3	0.45
2	0.675	0.2	0.45
3	1.000	0.8	0.75
4	0.875	0.5	0.65

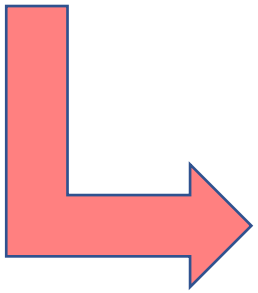
Normalization

- Applying Min-Max method in Python.

```
df_employees[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	23	1	500
1	30	3	900
2	27	2	900
3	40	8	1500
4	35	5	1300

```
df_norm2 = df_employees

df_norm2["Age"] = (df_norm2["Age"] - df_norm2["Age"].min()) / (df_norm2["Age"].max() - df_norm2["Age"].min())
df_norm2["Worked years"] = (df_norm2["Worked years"] - df_norm2["Worked years"].min()) / (df_norm2["Worked years"].max() - df_norm2["Worked years"].min())
df_norm2["Salary"] = (df_norm2["Salary"] - df_norm2["Salary"].min()) / (df_norm2["Salary"].max() - df_norm2["Salary"].min())

df_norm2[['Age','Worked years','Salary']].head()
```



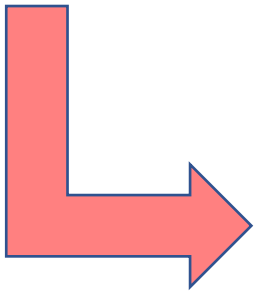
	Age	Worked years	Salary
0	0.190476	0.000000	0.000000
1	0.523810	0.222222	0.266667
2	0.380952	0.111111	0.266667
3	1.000000	0.777778	0.666667
4	0.761905	0.444444	0.533333

Normalization

- Applying Z-score method in Python.

```
df_employees[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	23	1	500
1	30	3	900
2	27	2	900
3	40	8	1500
4	35	5	1300



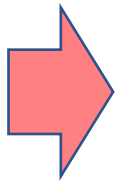
```
df_norm3 = df_employees

df_norm3["Age"] = (df_norm3["Age"] - df_norm3["Age"].mean()) / df_norm3["Age"].std()
df_norm3["Worked years"] = (df_norm3["Worked years"] - df_norm3["Worked years"].mean()) / df_norm3["Worked years"].std()
df_norm3["Salary"] = (df_norm3["Salary"] - df_norm3["Salary"].mean()) / df_norm3["Salary"].std()

df_norm3[['Age','Worked years','Salary']].head()
```

	Age	Worked years	Salary
0	-1.044119	-0.989598	-1.264654
1	-0.036004	-0.380615	-0.471146
2	-0.468053	-0.685106	-0.471146
3	1.404160	1.141844	0.719117
4	0.684078	0.228369	0.322363

Grouping into Classes

- Grouping the data into classes.
- Sales have a range that goes from 100,000 to a little more than 900,000

Sales
150000
651750
563750
706250
640375
519375
870375
926250
676250
103750
567925



Classes	Classes Range
Low	0 - 299,999
Medium	300,000 - 599,999
High	600,000 - 999,999

Grouping into Classes

- Grouping with Python.

```
my_class = np.linspace(min(df_test["Sales"]), max(df_test["Sales"]), 4)
group_names = ["Low","Medium","High"]

df_test["Sales Category"] = pd.cut(df_test["Sales"], my_class, labels = group_names, include_lowest = True)

df_test[['Sales','Sales Category']].head()
```

	Sales	Sales Category
0	150000.0	Low
2	563750.0	Medium
3	706250.0	High
4	553509.0	Medium
5	519375.0	Medium

Variable Conversion

- Converting categorical variables to numeric variables.
- get_dummies()

```
df_dummies = pd.get_dummies(df_test['Payment Type'])
df_test2 = pd.concat([df_test, df_dummies], axis = 1, sort = False)

df_test2.head()
```

Customer	Customer Type	Payment Type	Purchases	Sales	Refunds	Country	Continent	Purchases in thousands	Sales in thousands	Refunds in thousands	Sales Category	Cash	Credit Card	Transfer
10000	Person	Cash	120000.0	150000.0	240	Canada	America	120.0	150.000	0.240	Low	1	0	0
10002	Company	Credit Card	451000.0	563750.0	902	Mexico	America	451.0	563.750	0.902	Medium	0	1	0
10003	Company	Transfer	565000.0	706250.0	1130	Spain	Europe	565.0	706.250	1.130	High	0	0	1
10004	Person	Transfer	512300.0	553509.0	1024	Argentina	America	512.3	553.509	1.024	Medium	0	0	1
10005	Person	Transfer	415500.0	519375.0	0	Canada	America	415.5	519.375	0.000	Medium	0	0	1

Lab5_DataPreProcessing

Exploratory Data Analysis (EDA)

- Descriptive Analysis in Python
- Frequency Distribution
- Correlation Analysis
- Scatter Plot
- Regression Analysis

Descriptive Analysis in Python

- describe()

```
df_operations.describe()
```

	Customer	Purchases	Sales	Refunds
count	19.000000	19.000000	19.000000	19.000000
mean	10009.000000	450589.210526	563252.210526	819.210526
std	5.627314	167280.787361	209101.355900	439.467554
min	10000.000000	83000.000000	103750.000000	0.000000
25%	10004.500000	388850.000000	486062.500000	592.000000
50%	10009.000000	454100.000000	567925.000000	910.000000
75%	10013.500000	531200.000000	664000.000000	1062.500000
max	10018.000000	741000.000000	926250.000000	1482.000000

Descriptive Analysis in Python

- describe()

```
df_operations.describe(include = [np.object])
```

	Customer Type	Payment Type	Country	Continent
count	19	19	19	19
unique	2	3	8	3
top	Company	Cash	EEUU	America
freq	10	8	6	14

Descriptive Analysis in Python

- value_counts()

```
df_cus_type_counts = df_operations["Customer Type"].value_counts()
df_cus_type_counts = df_cus_type_counts.to_frame()
df_cus_type_counts
```

	Customer Type
Company	10
Person	9

Descriptive Analysis in Python

- groupby()

```
df_test = df_operations[['Customer Type','Payment Type','Sales']]
df_test.groupby(['Customer Type','Payment Type'], as_index = False).mean()
```

	Customer Type	Payment Type	Sales
0	Company	Cash	530185.000000
1	Company	Credit Card	711425.666667
2	Company	Transfer	503756.000000
3	Person	Cash	547179.000000
4	Person	Credit Card	508875.000000
5	Person	Transfer	562447.750000

Descriptive Analysis in Python

- pivot()

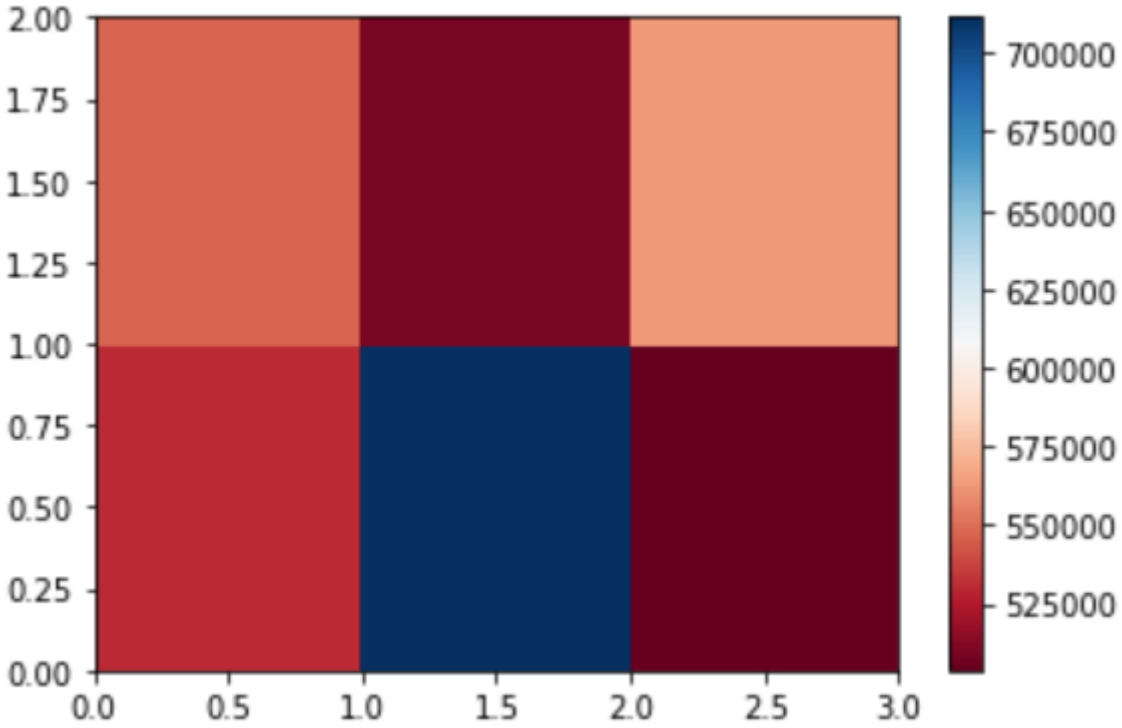
```
df_grp = df_test.groupby(['Customer Type','Payment Type'], as_index = False).mean()
df_grp.pivot(index = 'Customer Type', columns = 'Payment Type')
```

	Sales		
Payment Type	**Cash**	**Credit Card**	**Transfer**
Customer Type			
Company	530185.0	711425.666667	503756.00
Person	547179.0	508875.000000	562447.75

Descriptive Analysis in Python

- Heatmap

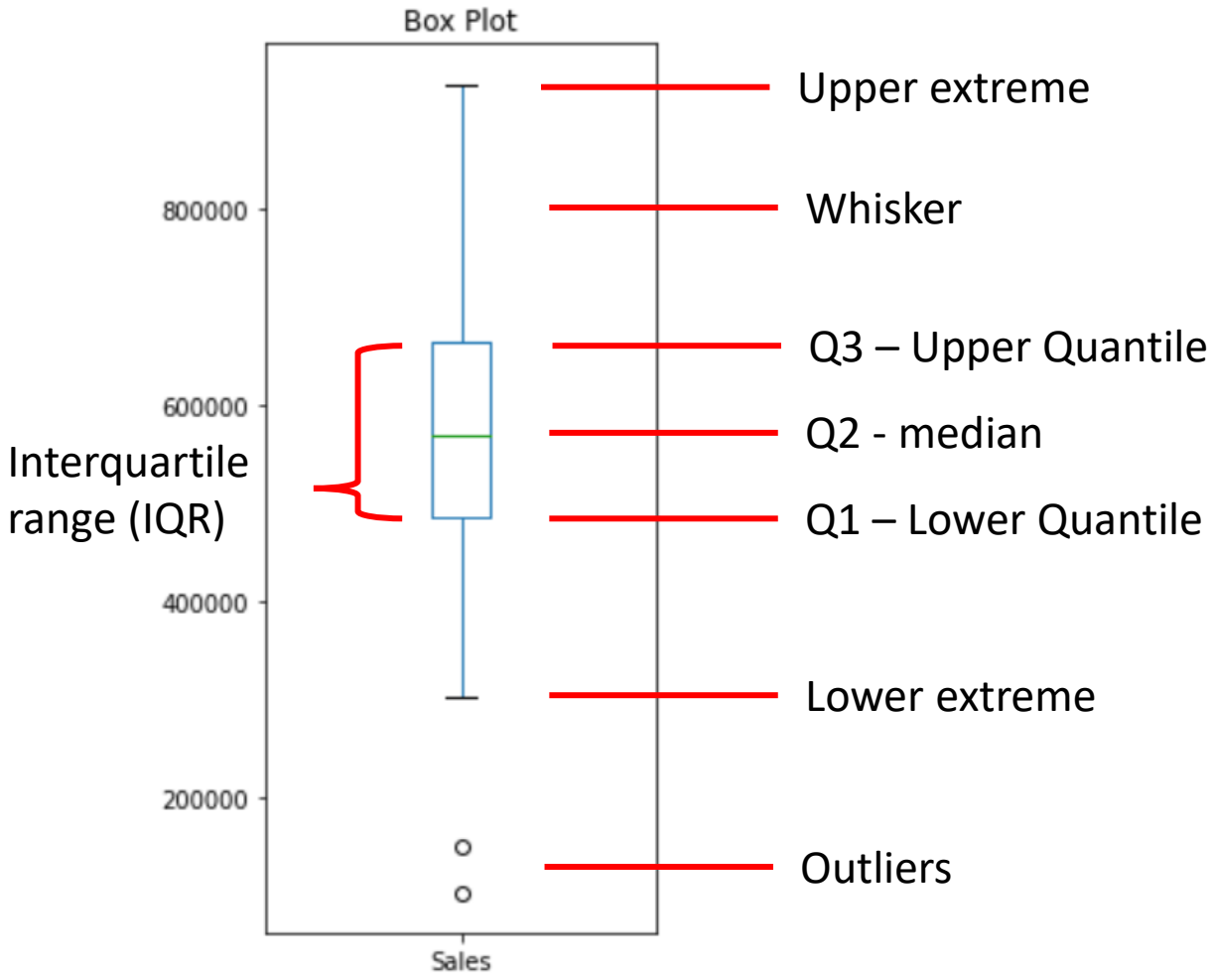
```
df_pivot = df_grp.pivot(index = 'Customer Type', columns = 'Payment Type')
plt.pcolor(df_pivot, cmap = 'RdBu')
plt.colorbar()
plt.show()
```



Descriptive Analysis in Python

- Box Plots with matplotlib library

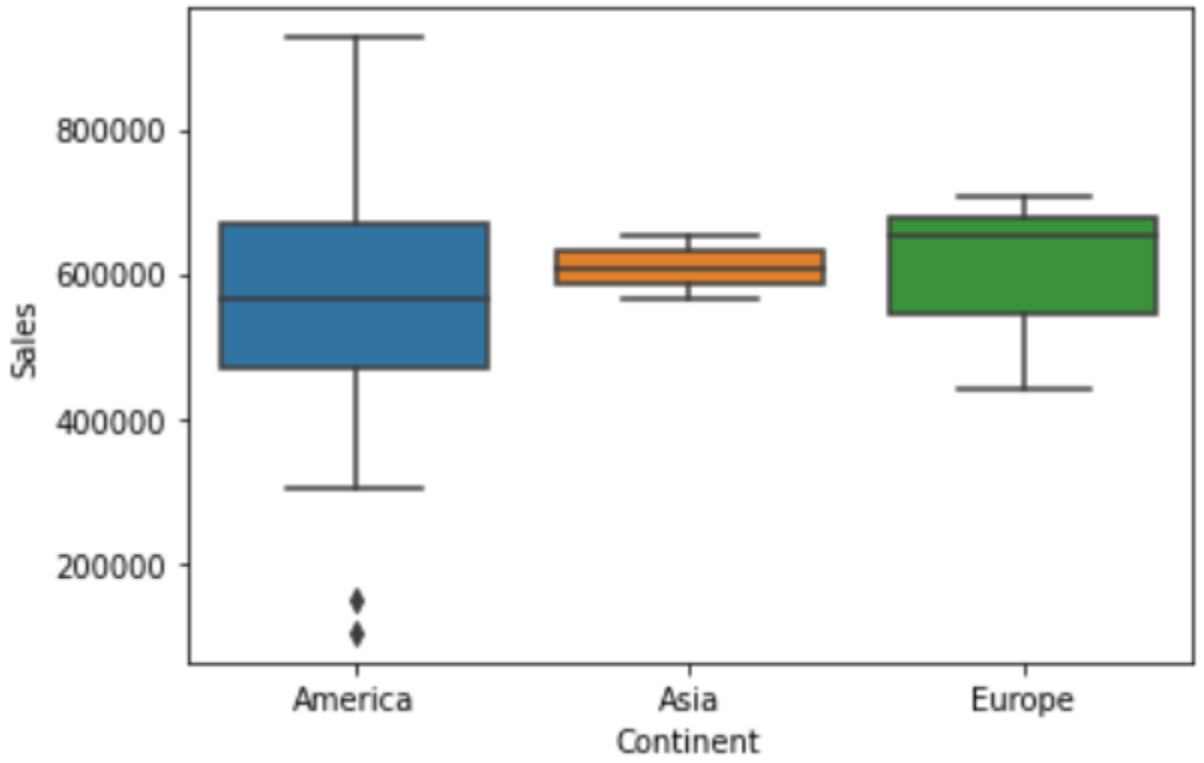
```
#We only take the variable sales.
df_oper_sales = df_operations.loc[:,'Sales']

df_oper_sales.plot(kind='box', figsize=(3,7))
plt.title('Box Plot')
plt.show()
```



Descriptive Analysis in Python

- Box Plots with seaborn library

```
sns.boxplot(x="Continent", y="Sales", data=df_operations)
```



Lab6_EDA_1

Frequency Distribution

- It consists of grouping the data into categories that show the number of cases or observations in each mutually exclusive category.

500	3000	2500	680	550
900	1400	750	850	2500
900	650	1320	700	1300
1500	2500	240	1900	750
1300	900	800	2100	2050
1400	1350	1100	750	1400
600	1900	950	800	900
2000	700	630	1000	600

Frequency Distribution

- Step 1: Establish categorical groups calls classes.
- Step 2: Distribute the data in the corresponding class.
- Step 3: Count the amount of data in each class.

Frequency Distribution

- Step 1: Establish categorical groups calls classes.
  - Define the class interval.

$$class\ interval = \frac{Maximum\ - Minimun\ value}{number\ of\ classes}$$

  - Rules of thumb of determining the number of classes.
    1) Not less than 5 and not more than 15.
    2) $2^k \geq n$ (k = 0,1,2,...) (n = number of observations).

k	2^k	n
0	1	40
1	2	40
2	4	40
3	8	40
4	16	40
5	32	40
6	64	40

$$class\ interval = \frac{3000\ - 240}{6} = 460$$

Salaries	
0	499
500	999
1000	1499
1500	1999
2000	2499
2500	3000

Frequency Distribution

- Step 2: Distribute the data in the corresponding class.

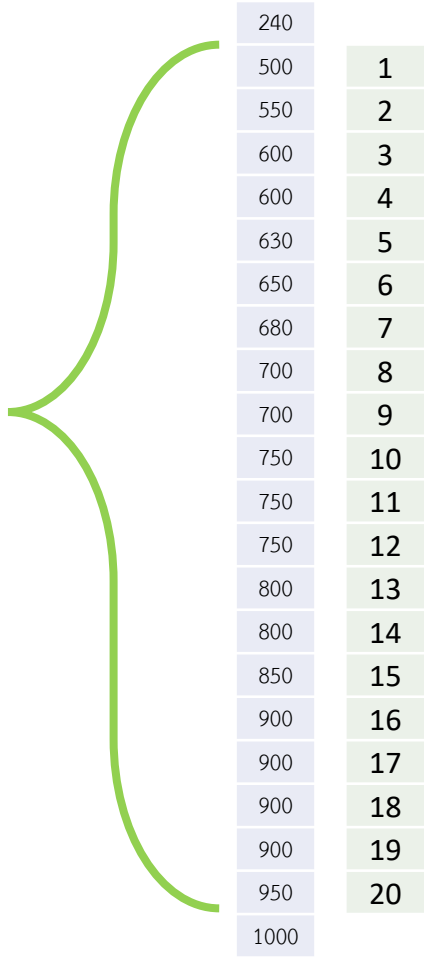
Salaries	
0	499
500	999
1000	1499
1500	1999
2000	2499
2500	3000

240
500
550
600
600
630
650
680
700
700
750
750
750
800
800
850
900
900
900
900
950
1000

Frequency Distribution

- Step 3: Count the amount of data in each class.

Salaries	
0	499
500	999
1000	1499
1500	1999
2000	2499
2500	3000



Salaries		
From	To	Frequency
0	499	1
500	999	20
1000	1499	9
1500	1999	3
2000	2499	3
2500	3000	4

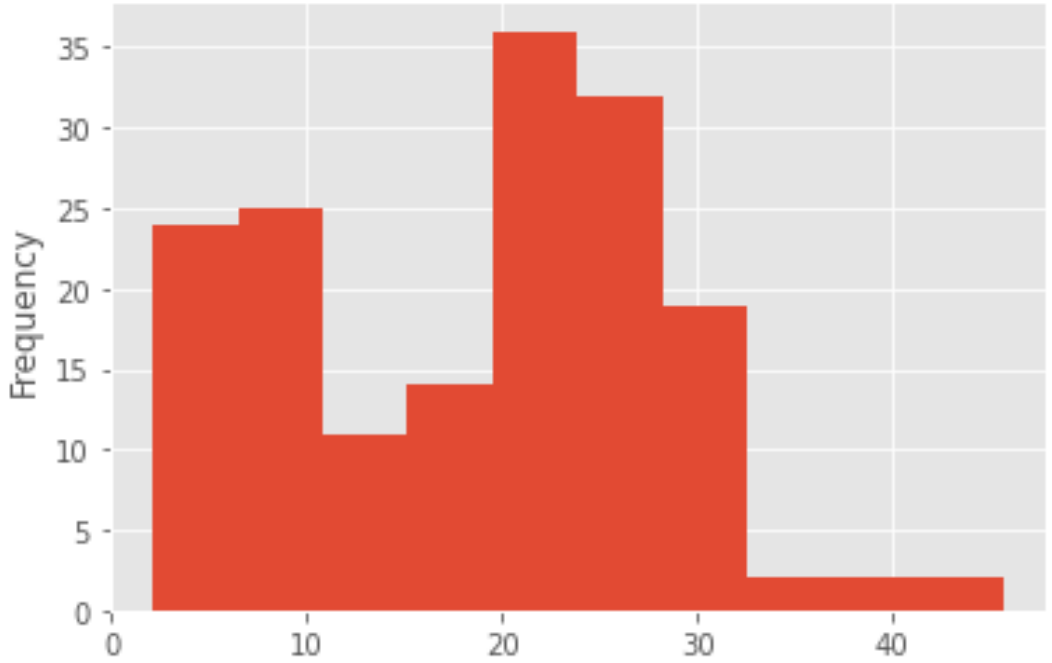
Frequency Distribution in Python

- Histrogram.
  - plot() method.

```
Country
Afghanistan              4.5
Albania                 22.3
Algeria                 26.6
Angola                   6.8
Antigua and Barbuda     19.1
Argentina               28.5
Armenia                 20.9
Australia               30.4
Austria                 21.9
Azerbaijan              19.9
Name: Obesity, dtype: float64
```

```
# An easy way to create the histogram.
df_fat['Obesity'].plot.hist()
```

```
<AxesSubplot:ylabel='Frequency'>
```



Correlation Analysis

- Measures the strength of correlation between two variables.
  - Correlation coefficient
    - Close to 1 : High positive correlation
    - Close to -1 : High negative correlation
    - Close to 0 : No correlation
  - p value
    - $p < 0.001$: High certainty in the result
    - $p < 0.05$: Moderate certainty in the result
    - $p < 0.1$: Low certainty in the result
    - $p > 0.1$: Lack of certainty in the result

Correlation Analysis

- Pearson's Correlation Coefficient.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

$$r = \frac{(20 * 4213.468) - (190.3998 * 361.6)}{\sqrt{[(20 * 2501.446) - 190.3998^2][(20 * 8568.5) - 361.6^2]}}$$

$$r = \frac{559552262.4}{23654.85706} = 0.651908$$

n = 20

Meat	Obesity			
x	y	xy	x^2	y^2
6.1244	4.5	27.5598	37.50828	20.25
8.7428	22.3	194.9644	76.43655	497.29
3.8961	26.6	103.6363	15.1796	707.56
11.0268	6.8	74.98224	121.5903	46.24
14.3202	19.1	273.5158	205.0681	364.81
19.2693	28.5	549.1751	371.3059	812.25
10.8165	20.9	226.0649	116.9967	436.81
11.6002	30.4	352.6461	134.5646	924.16
8.1099	21.9	177.6068	65.77048	479.61
11.9993	19.9	238.7861	143.9832	396.01
17.4941	32.1	561.5606	306.0435	1030.41
1.8407	3.4	6.25838	3.388176	11.56
13.1382	24.8	325.8274	172.6123	615.04
11.5636	26.6	307.5918	133.7168	707.56
5.6817	24.5	139.2017	32.28171	600.25
10.3435	22.4	231.6944	106.988	501.76
3.2849	8.2	26.93618	10.79057	67.24
21.1476	18.7	395.4601	447.221	349.69
190.3998	361.6	4213.468	2501.446	8568.5

Correlation Analysis in Python

- Correlation Coefficient with p value.

Country	Alcoholic Beverages	Animal Products	Animal fats	Aquatic Products, Other	Cereals - Excluding Beer	Eggs	Fish, Seafood	Fruits - Excluding Wine	Meat	Miscellaneous	...	Vegetable Oils	Vegetables	Obesity
Afghanistan	0.0	21.6397	6.2224	0.0	8.0353	0.6859	0.0327	0.4246	6.1244	0.0163	...	17.0831	0.3593	4.5
Albania	0.0	32.0002	3.4172	0.0	2.6734	1.6448	0.1445	0.6418	8.7428	0.0170	...	9.2443	0.6503	22.3
Algeria	0.0	14.4175	0.8972	0.0	4.2035	1.2171	0.2008	0.5772	3.8961	0.0439	...	27.3606	0.5145	26.6
Angola	0.0	15.3041	1.3130	0.0	6.5545	0.1539	1.4155	0.3488	11.0268	0.0308	...	22.4638	0.1231	6.8
Antigua and Barbuda	0.0	27.7033	4.6686	0.0	3.2153	0.3872	1.5263	1.2177	14.3202	0.0898	...	14.4436	0.2469	19.1

```
from scipy import stats
```

```
pearson_coef, p_value = stats.pearsonr(df_fat['Meat'], df_fat['Obesity'])
print("Pearson's correlation coefficient: ", pearson_coef, " p value: ", p_value)
```

```
Pearson's correlation coefficient:  0.219919023772617  p value:  0.00429447433848602
```

Correlation Analysis in Python

- Correlation Matrix.

```
df_fat[['Animal Products','Meat','Cereals - Excluding Beer','Sugar & Sweeteners','Obesity']].corr()
```

	Animal Products	Meat	Cereals - Excluding Beer	Sugar & Sweeteners	Obesity
Animal Products	1.000000	0.738702	-0.459064	-0.016549	0.417490
Meat	0.738702	1.000000	-0.270603	0.095534	0.219919
Cereals - Excluding Beer	-0.459064	-0.270603	1.000000	-0.003046	-0.488142
Sugar & Sweeteners	-0.016549	0.095534	-0.003046	1.000000	-0.163192
Obesity	0.417490	0.219919	-0.488142	-0.163192	1.000000

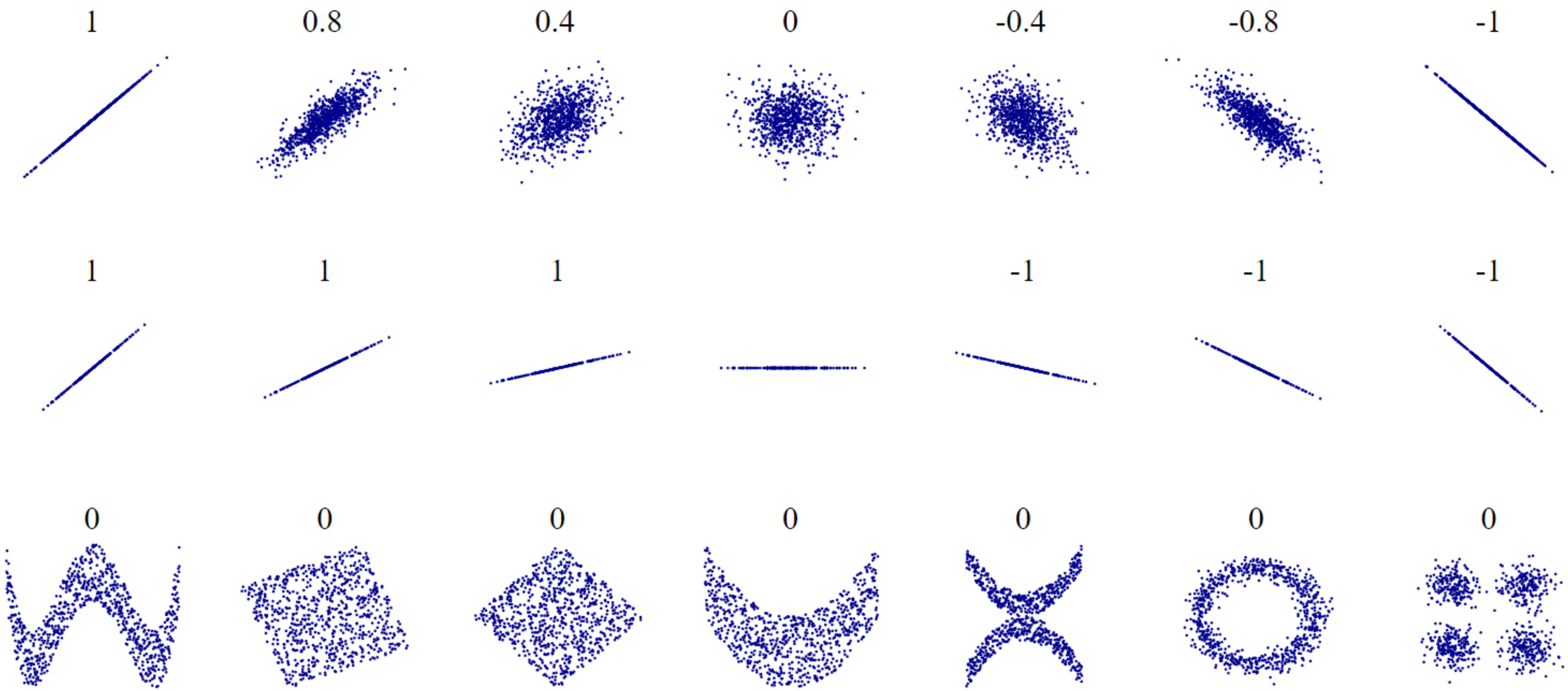
- Determination Coefficient:
  - $r^2 = (0.219919)^2 = 0.048364366561 = 0.4\%$

Scatter Plot

- A scatter plot is a graphical illustration that is uses in **regression analysis.**

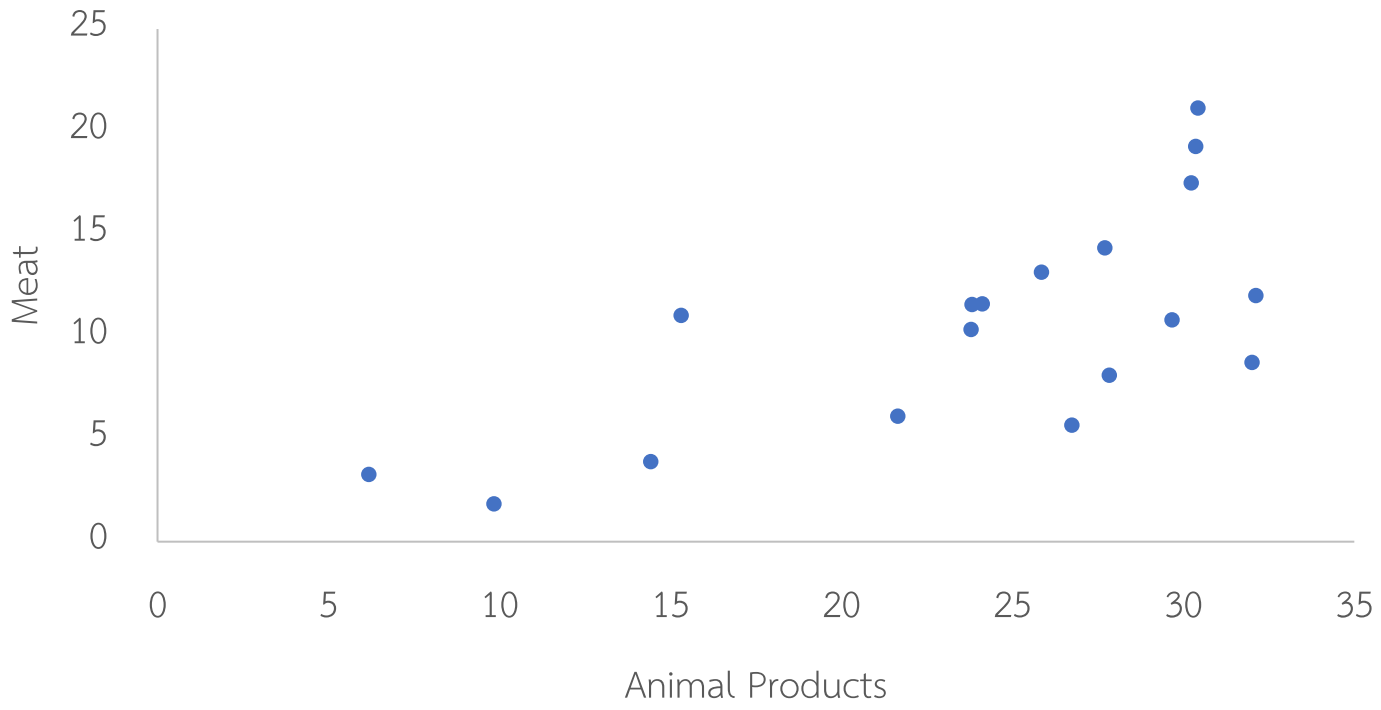
- It consists of a dispersion of point where each point represents a value of the **independent variable** (measured on the horizontal axis), and a value associated with the **dependent variable** (measured on the vertical axis).

Scatter Plot



Regression Analysis

Animal Products	Meat
x	y
21.6397	6.1244
32.0002	8.7428
14.4175	3.8961
15.3041	11.0268
27.7033	14.3202
30.3572	19.2693
29.6642	10.8165
24.1099	11.6002
27.8268	8.1099
32.1198	11.9993
30.2259	17.4941
9.8365	1.8407
25.8451	13.1382
23.8181	11.5636
26.7378	5.6817
23.7877	10.3435
6.1742	3.2849
30.4177	21.1476
431.9857	190.3998



Regression Analysis

- Least squares method

$$\hat{y} = a + bX$$

$$b = \frac{\sum X_y - n(\bar{x})(\bar{y})}{\sum X^2 - n\bar{x}^2}$$

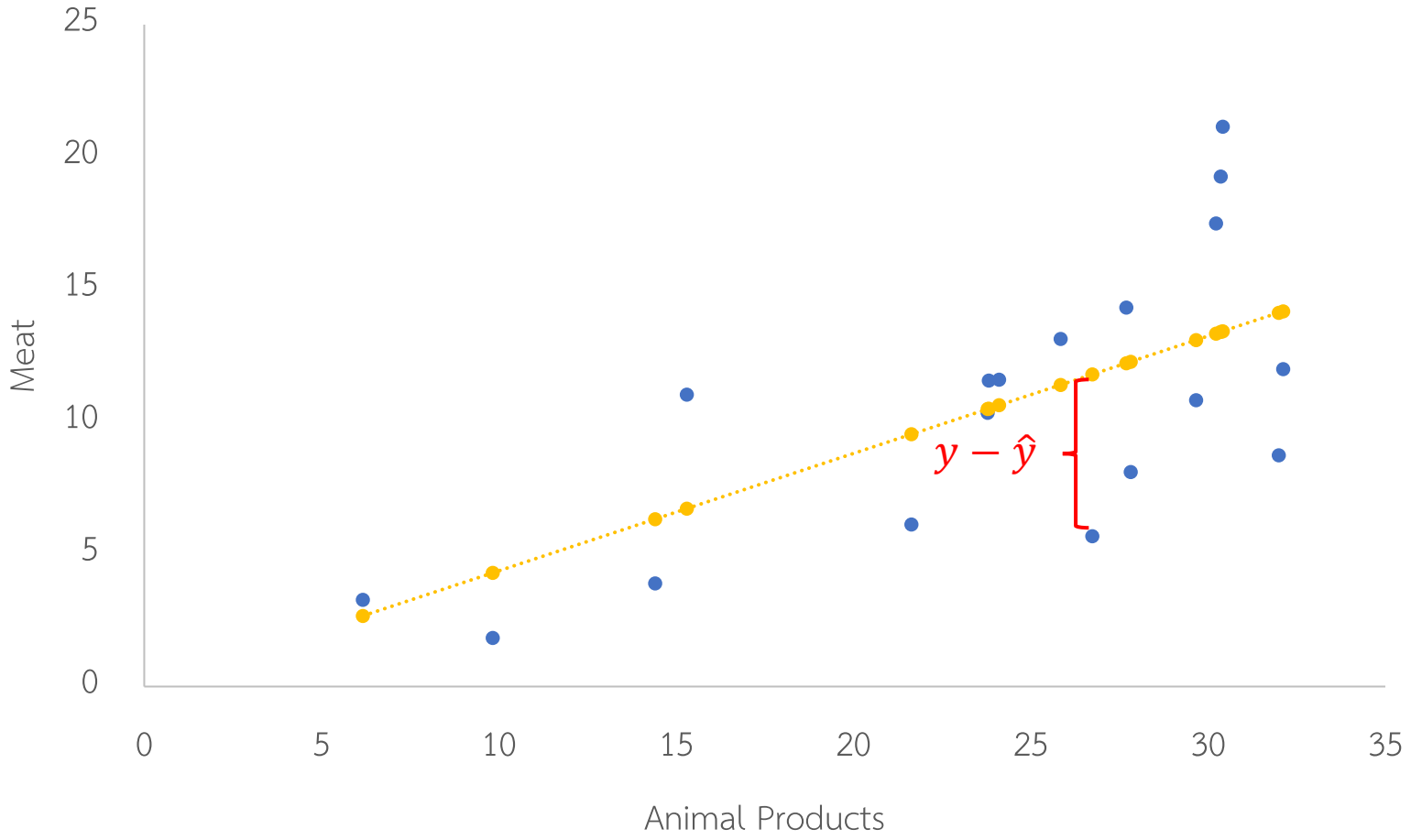
$$a = \bar{y} - b\bar{x}$$

Animal Products	Meat		
x	y	xy	x^2
21.6397	6.1244	132.530179	468.276616
32.0002	8.7428	279.771349	1024.0128
14.4175	3.8961	56.1720218	207.864306
15.3041	11.0268	168.75525	234.215477
27.7033	14.3202	396.716797	767.472831
30.3572	19.2693	584.961994	921.559592
29.6642	10.8165	320.862819	879.964762
24.1099	11.6002	279.679662	581.287278
27.8268	8.1099	225.672565	774.330798
32.1198	11.9993	385.415116	1031.68155
30.2259	17.4941	528.774917	913.605031
9.8365	1.8407	18.1060456	96.7567323
25.8451	13.1382	339.558093	667.969194
23.8181	11.5636	275.422981	567.301888
26.7378	5.6817	151.916158	714.909949
23.7877	10.3435	246.048075	565.854671
6.1742	3.2849	20.2816296	38.1207456
30.4177	21.1476	643.261353	925.236473
431.9857	190.3998	82249.9909	186611.645

$$\hat{y} = 0.443303X - 0.06116$$

Regression Analysis

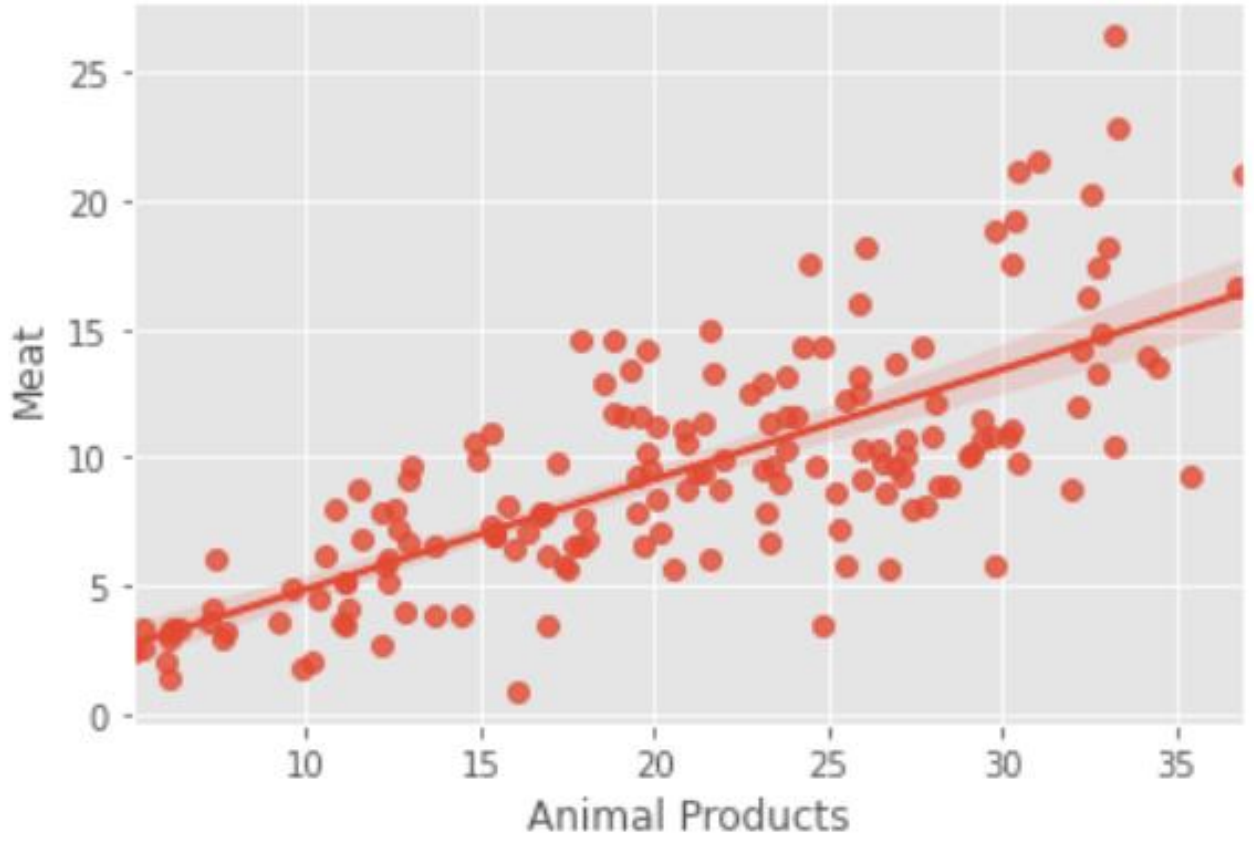
Animal Products	Meat			
x	y	xy	x^2	y^
21.6397	6.1244	132.530179	468.276616	9.531784
32.0002	8.7428	279.771349	1024.0128	14.12462
14.4175	3.8961	56.1720218	207.864306	6.330161
15.3041	11.0268	168.75525	234.215477	6.723193
27.7033	14.3202	396.716797	767.472831	12.2198
30.3572	19.2693	584.961994	921.559592	13.39628
29.6642	10.8165	320.862819	879.964762	13.08907
24.1099	11.6002	279.679662	581.287278	10.62683
27.8268	8.1099	225.672565	774.330798	12.27454
32.1198	11.9993	385.415116	1031.68155	14.17764
30.2259	17.4941	528.774917	913.605031	13.33807
9.8365	1.8407	18.1060456	96.7567323	4.29939
25.8451	13.1382	339.558093	667.969194	11.39605
23.8181	11.5636	275.422981	567.301888	10.49748
26.7378	5.6817	151.916158	714.909949	11.79179
23.7877	10.3435	246.048075	565.854671	10.484
6.1742	3.2849	20.2816296	38.1207456	2.675881
30.4177	21.1476	643.261353	925.236473	13.4231
431.9857	190.3998	82249.9909	186611.645	191.4394



Regression Analysis in Python

- regplot()

```
sns.regplot(x = "Animal Products", y = "Meat", data = df_fat)
```



Lab7_EDA_2

Follow us on

govbigdata

Twitter

Blockdit

GBDi

gbdi.depa.or.th

YouTube

Government Big Data Institute (GBDi)

Line Official

@gbdi