Data and Dataset

- **Data encompasses a collection of numbers, words, events, facts,measurements, observations or even description of things**

- Such data is collected and stored by every event or process occurringin several disciplines.
- Engineering, Marketing, Biology , Agriculture,Social Sciences,Entertainment and Media.

- A **Dataset is a set or collection of data**. This set is normally presented ina tabular pattern. Every column describes a particular variable.

Rows/ Sample

		Gender	Item Pur	Category	Purchase Amount (USD)	Size	Season	Review Rating	Payment Method	Discount Applied	Previous Purchases	Preferred Payment Method	Frequency of Purchases
1	55	Male	Blouse	Clothing	53	L	Winter	3.1	Credit Card	Yes	14	Venmo	Fortnightly
2	19	Male	Sweater	Clothing	64	L	Winter	3.1	Bank Transfer	Yes	2	Cash	Fortnightly
3	50	Male	Jeans	Clothing	73	S	Spring	3.1	Cash	Yes	23	Credit Card	Weekly
4	21	Male	Sandals	Footwear	90	M	Spring	3.5	PayPal	Yes	49	PayPal	Weekly
5	45	Male	Blouse	Clothing	49	M	Spring	2.7	Cash	Yes	31	PayPal	Annually
6	46	Male	Sneakers	Footwear	20	M	Summer	2.9	Venmo	Yes	14	Venmo	Weekly
7	63	Male	Shirt	Clothing	85	M	Fall	3.2	Debit Card	Yes	49	Cash	Quarterly
8	27	Male	Shorts	Clothing	34	L	Winter	3.2	Debit Card	Yes	19	Credit Card	Weekly

Customer Shopping Trends (Dataset)

Types of data - Structured Data

Structured data is data that has a standardized format. It is typically tabular with rows and columns that clearly define data attributes.

- **(relational) database tables**
- **CSV/xlsx files**

Types of data -Semi-structured Data

- **Semi-structured data is a form of structured data that does not obey the tabular structure of data models associated with relational databases or other forms of data tables,**
- **but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data**
    - XML
    - JSON
    - PDF
    - Email

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<Restaurant id="41955207">
  <Name>Staghorn Steakhouse</Name>
  <ZipCode>10018</ZipCode>
  <Cuisines>Seafood, Steakhouse</Cuisines>
  <PriceLevel></PriceLevel>
  <Hours></Hours>
  <Payment></Payment>
  <DressCode></DressCode>
  <SpecialFeatures>
    <SpecialFeature>Notable Wine List</SpecialFeature>
    <SpecialFeature>Business Dining</SpecialFeature>
    <SpecialFeature>Romantic Dining</SpecialFeature>
    <SpecialFeature>Group Dining</SpecialFeature>
    <SpecialFeature>Fine Dining</SpecialFeature>
  </SpecialFeatures>
  <PromptSeating>yes</PromptSeating>
  <MakeReservation>yes</MakeReservation>
  <Romantic>yes</Romantic>
```

Semi-Structured Data - Review

Types of data -Unstructured Data

- Unstructured data (or unstructured information)is information that either does not have a pre-defined data model or is not organized in a pre-defined manner.
- Unstructured information is typically text-heavy
  - documents,
  - Web pages,
  - short texts (e.g., social media)

Unstructured data – Feedback.txt

```
The food for our event was delicious .
The food in the lounge was great and very fresh, , , salads, sandwiches etc .
As far as food, walk a few blocks toward Michigan Ave turn left or right and there are
plently of less expensive places to eat .
The Palm resturant in the hotel had some specials Sunday night, we ate there and the
food service,etc were outstanding portions are large and we shared since we are not
big eaters .
Took the charge of the minibar which we had used to keep my 2 year old sons food .
We never ate anything onsite so I can't vouch for the food options immediately
avialable .
The Lobby bar does not serve food very late at night and we couldn't find any vending
machines for soda or snacks, so stop at a nearby market before going in for the night .
In any case, I had allotted approximately $150 for "food" for 4 days 3 nights .
But the presentation was awesome and the food was so good and not having to walk in
the dark at night and scare my mother or pay for a taxi or bus, but sit around and eat
comfortably in my pajamas was well worth the cost to me .
The food was delicious and the view was fabulous .
```

Exploratory Data Analysis (EDA)

- A set of procedures for examining the data
- Summarizing the data with the help of descriptive and graphical tools.
- "No Assumption" study of data
- Determining the relationships among variables
- Handling missing values

Exploratory Data Analysis (EDA)

- **Exploratory Data Analysis** is a process of examining or understanding the data and extracting insights or main characteristics of the data.
- EDA exposes trends, patterns, and relationships within the dataset that may not be apparent.

Data Analytics

- Analytics is the use of tools and processes to combineand examine sets of data to identify patterns, relationships and trends.
- The goal of analytics is to answer specific questions, discover new insights, and help organizations make better, data-driven decisions.

Steps in EDA

- Problem definition
- Data Preparation
- Data analysis
- Development and representation of the results

Steps in EDA -Problem definition

- Before trying to extract useful insight from the data, it isessential to define the business problem to be solved
- The problem definition execution plan involves,
  - Defining main objective of the analysis
  - Defining the main deliverables
  - Outlining the main roles and responsibilities
  - Obtaining the current state of the data
  - Performing Cost/benefit analysis
  - Defining the time table

Steps in EDA -Data Preparation

The step involves in data preparation step includes

- Defining the source of data
- Defining data schemes and tables
- Understand the main characteristics of the data
- Clean the dataset
- Delete non-relevant datasets
- Transform the data
- Divide the data into required chunks for analysis

Steps in EDA -Data Analysis

This is one of the most crucial steps that deals with descriptive statistics and analysis of the data. The main task involves

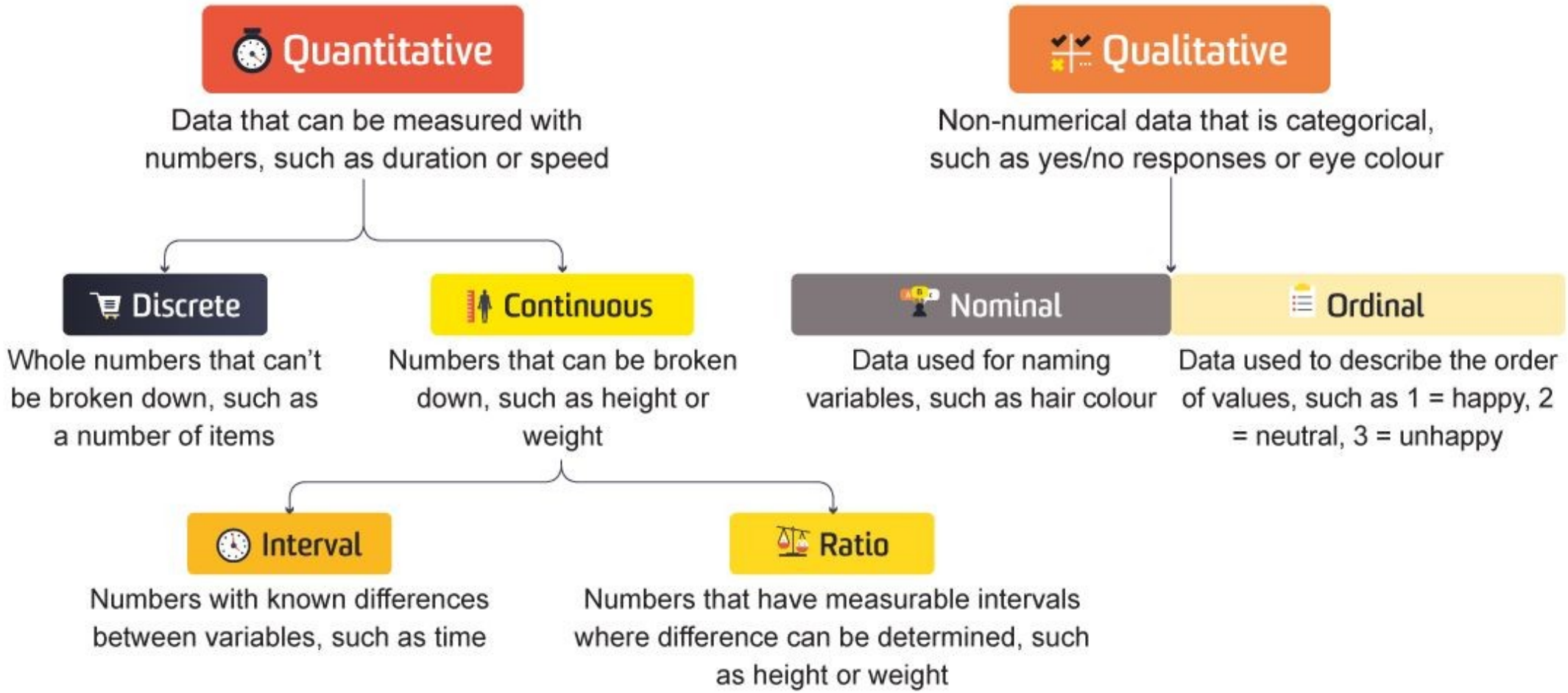
- Summarizing the data { tables, graphs, descriptive statistics,inferential statistics, correlation statistics, grouping and models}
- Finding the hidden correlation and relationships among thedata
- Developing predictive models
- Evaluating the models

Steps in EDA –Developing and representation of the result

- This step involves presenting the dataset to the targetaudience in the form of
  - Graphs
  - Summary tables
  - Maps and
  - Diagrams

- This is also an essential step as the result analyzed from the dataset should be interpretable by the business stakeholders

Types of Data



Data Transformation

Data Transformation

- Data transformation is a set of techniques used to convert data from one format or structure to another format or structure.

Python libraries

- We will be using the following
  - Pandas
    - Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.
  - NumPy
    - NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices
  - Seaborn
    - Python Seaborn library is a widely popular data visualization library that is commonly used for data science and machine learning tasks
  - Matplotlib
    - Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python

Pandas DataFrames

- A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```

Result

	calories	duration
0	420	50
1	380	40
2	390	45

- Create a simple Pandas DataFrame:

Named Indexes

- With the index argument, you can name your own indexes.
- Add a list of names to give each row a name:

```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

Result

	calories	duration
day1	420	50
day2	380	40
day3	390	45

Load Files Into a DataFrame

- If your data sets are stored in a file, Pandas can load them into a DataFrame.
- Load a comma separated file (CSV file) into a DataFrame:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df)
```

Data deduplication

- The pandas dataframe comes with a duplicated() method that returns a Boolean series stating which of the rows are duplicates:

```
0    False
1     True
2    False
3    False
4     True
5    False
6     True
dtype: bool
```

```
frame3.duplicated
()
```

The rows that say True are the ones that contain duplicated data.

Drop duplication

- We can drop these duplicates using the drop_duplicates() method:

	column 1	column 2
0	Looping	10
1	Looping	10
2	Looping	22
3	Functions	23
4	Functions	23
5	Functions	24
6	Functions	24

```
frame4 = frame3.drop_duplicates()
```

	column 1	column 2
0	Looping	10
2	Looping	22
3	Functions	23
5	Functions	24

both the duplicated and drop_duplicates methods keep the first observed value during the duplication removal process. If we pass the take_last=True argument, the methods return the last one.

You can use the duplicated() function to find duplicate values in a pandas DataFrame.

This function uses the following basic syntax:

```
#find duplicate rows across all columns
duplicateRows = df[df.duplicated()]


#find duplicate rows across specific columns
duplicateRows = df[df.duplicated(['col1', 'col2'])]
```

```python
import pandas as pd

#create DataFrame
df = pd.DataFrame({'team': ['A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'],
                   'points': [10, 10, 12, 12, 15, 17, 20, 20],
                   'assists': [5, 5, 7, 9, 12, 9, 6, 6]})

#view DataFrame
print(df)
```

```
  team  points  assists
0    A      10        5
1    A      10        5
2    A      12        7
3    A      12        9
4    B      15       12
5    B      17        9
6    B      20        6
7    B      20        6
```

Find Duplicate Rows Across All Columns

	team	points	assists
0	A	10	5
1	A	10	5
2	A	12	7
3	A	12	9
4	B	15	12
5	B	17	9
6	B	20	6
7	B	20	6

```
#identify duplicate rows
duplicateRows = df[df.duplicated()]


#view duplicate rows
duplicateRows
```

	team	points	assists
1	A	10	5
7	B	20	6

keep='last' to display the first duplicate rows instead of the last

	team	points	assists
0	A	10	5
1	A	10	5
2	A	12	7
3	A	12	9
4	B	15	12
5	B	17	9
6	B	20	6
7	B	20	6

```
#identify duplicate rows
duplicateRows = df[df.duplicated(keep='last')]

#view duplicate rows
print(duplicateRows)
```

	team	points	assists
0	A	10	5
6	B	20	6

Find Duplicate Rows Across Specific Columns

The following code shows how to find duplicate rows across just the 'team' and 'points' columns of the DataFrame:

	team	points	assists
0	A	10	5
1	A	10	5
2	A	12	7
3	A	12	9
4	B	15	12
5	B	17	9
6	B	20	6
7	B	20	6

```
#identify duplicate rows across 'team' and 'points' columns
duplicateRows = df[df.duplicated(['team', 'points'])]


#view duplicate rows
print(duplicateRows)
```

```
        team    points  assists
1       A       10      5
3       A       12      9
7       B       20      6
```

Replacing Values

	column 1	column 2
0	200.0	0
1	3000.0	1
2	-786.0	2
3	3000.0	3
4	234.0	4
5	444.0	5
6	-786.0	6
7	332.0	7
8	3332.0	8

- Use **replace method** to find and replace some values inside a dataframe

	column 1	column 2
0	200.0	0
1	3000.0	1
2	NaN	2
3	3000.0	3
4	234.0	4
5	444.0	5
6	NaN	6
7	332.0	7
8	3332.0	8

```
import numpy as np
replaceFrame = pd.DataFrame({'column 1': [200., 3000., -786.,
3000., 234., 444., -786., 332., 3332. ], 'column 2': range(9)})
replaceFrame.replace(to_replace =-786, value= np.nan)
```

Handling Missing Data

Whenever there are missing values, a NaN value is used, which indicates that there is no value specified for that particular index. There could be several reasons why a value could be NaN:

- It can happen when **data is retrieved from an external source** and there are some incomplete values in the dataset.
- It can also happen when **we join two different datasets** and some values are not matched.
- Missing values due to **data collection errors**.
- When the **shape of data changes**, there are new additional rows or columns that are not determined.

Add some missing values to our dataframe

```
dfx['store4'] = np.nan
dfx.loc['watermelon'] = np.arange(15, 19)
dfx.loc['oranges'] = np.nan
dfx['store5'] = np.nan
dfx['store4']['apple'] = 20.
```

	store1	store2	store3
apple	15	16	17
banana	18	19	20
kiwi	21	22	23
grapes	24	25	26
mango	27	28	29

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

d

The following characteristics of missing values in the preceding dataframe:

- An entire row can contain NaN values.
- An entire column can contain NaN values.
- Some (but not necessarily all) values in both a row and a column can be NaN.

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

NaN Values in Pandas Objects

isnull() function from the pandas library to identify NaN values:

```
dfx.isnull()
```

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

	store1	store2	store3	store4	store5
apple	False	False	False	False	True
banana	False	False	False	True	True
kiwi	False	False	False	True	True
grapes	False	False	False	True	True
mango	False	False	False	True	True
watermelon	False	False	False	False	True
oranges	True	True	True	True	True

True values indicate the values that are NaN

NaN Values in Pandas Objects

notnull() function from the pandas library to identify not NaN values:

```
dfx.notnull()
```

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

	store1	store2	store3	store4	store5
apple	True	True	True	True	False
banana	True	True	True	False	False
kiwi	True	True	True	False	False
grapes	True	True	True	False	False
mango	True	True	True	False	False
watermelon	True	True	True	True	False
oranges	False	False	False	False	False

True values indicate the values that are not NaN

NaN Values in Pandas Objects

sum() method to count the number of NaN values

```
dfx.isnull().sum()
```

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

```
store1 1
store2 1
store3 1
store4 5
store5 7
dtype: int64
```

NaN Values in Pandas Objects

To find the total number of missing values

```
dfx.isnull().sum().sum()
```

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

```
15
```

Dropping Missing Values

One of the ways to handle missing values is to simply remove them from dataset.

The `dropna() method` just returns a copy of the dataframe by dropping the rows with NaN. The original dataframe is not changed.

```
dataframe.dropna(axis, how, thresh, subset,
inplace)
```

Parameter	Value	Description

axis	`0`
	`1`
	`'index'`
	`'columns'`
		Optional, default 0.
		0 and 'index'removes ROWS that contains NULL values
		1 and 'columns' removes COLUMNS that contains NULL values

how	`'all'` `'any'`	Optional, default 'any'. Specifies whether to remove the row or column when ALL values are NULL, or if ANY value is NULL.
thresh	*Number*	Optional, Specifies the number of NOT NULL values required to keep the row.
subset	*List*	Optional, specifies where to look for NULL values
inplace	`True` `False`	Optional, default False. If True: the removing is done on the current DataFrame. If False: returns a copy where the removing is done.

Dropping Missing Values

- **Dropping by rows**
  - Use the how=all argument to drop only those rows entire

```
dfx.dropna(how='all')
```

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

values are entirely NaN:

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN

Dropping by columns

- Pass axis=1 to indicate a check for NaN by columns.

```
dfx.dropna(how='all',
```

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

```
axis=1)
```

	store1	store2	store3	store4
apple	15.0	16.0	17.0	20.0
banana	18.0	19.0	20.0	NaN
kiwi	21.0	22.0	23.0	NaN
grapes	24.0	25.0	26.0	NaN
mango	27.0	28.0	29.0	NaN
watermelon	15.0	16.0	17.0	18.0
oranges	NaN	NaN	NaN	NaN

Drop rows where specific columns have missing values

df.dropna(subset=['Column_Name'],inplace=True)

```
import pandas as pd
data = {
    'A': [1, 2, None, 4, 5],
    'B': ['a', 'b', 'c', None, 'e'],
    'C': [10.5, None, 30.2, 40.1, None]
}
df=pd.DataFrame(data)
print(df)
# Dropping rows where 'B' column has missing values
df_dropped_b = df.dropna(subset=['B'])

#DataFrame after dropping rows with missing values in 'B' column:")
print(df_dropped_b)
```

	A	B	C
0	1.0	a	10.5
1	2.0	b	NaN
2	NaN	c	30.2
3	4.0	None	40.1
4	5.0	e	NaN

	A	B	C
0	1.0	a	10.5
1	2.0	b	NaN
2	NaN	c	30.2
4	5.0	e	NaN

Thresh specify number of NoT NULL values to keep the row

```
df.dropna(thresh=4)
```

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
watermelon	15.0	16.0	17.0	18.0	NaN

Thresh : Drop column with minimum number of NaN exist

```
df.dropna(thresh=4, axis=1)
```

Mathematical operations with NaN

The pandas and numpy libraries handle NaN values differently for mathematical operations.

- When a NumPy function encounters NaN values, it returns NaN.

- Pandas, on the other hand, ignores the NaN values and moves ahead with processing. When performing the sum operation, NaN is treated as 0. If all the values are NaN, the result is also NaN.

Example

```
import numpy as np
import pandas as pd
ar1 = np.array([100, 200, np.nan, 300])
ser1 = pd.Series(ar1)
ar1.mean(), ser1.mean()
```

Output

```
(nan, 200.0)
```

The total quantity of fruits sold by store4
and its mean value

```
ser2 = dfx.store4
ser2.sum()
ser2.mean()
```

Output of the preceding code

```
38.0
19.0
```

	store1	store2	store3	store4	store5
apple	15.0	16.0	17.0	20.0	NaN
banana	18.0	19.0	20.0	NaN	NaN
kiwi	21.0	22.0	23.0	NaN	NaN
grapes	24.0	25.0	26.0	NaN	NaN
mango	27.0	28.0	29.0	NaN	NaN
watermelon	15.0	16.0	17.0	18.0	NaN
oranges	NaN	NaN	NaN	NaN	NaN

store4 has five NaN values. However, during the summing process, these values are treated as 0 and the result is 38.0

Replacing Missing Values

Mean: *data=data.fillna(data.mean())*

Median: *data=data.fillna(data.median())*

Standard Deviation: *data=data.fillna(data.std())*

Min: *data=data.fillna(data.min())*

Max: *data=data.fillna(data.max())*

```
# replacing missing values in quantity
# column with mean of that column
data['quantity'] = data['quantity'].fillna(data['quantity'].mean())

# replacing missing values in price column
# with median of that column
data['price'] = data['price'].fillna(data['price'].median())

# replacing missing values in bought column with
# standard deviation of that column
data['bought'] = data['bought'].fillna(data['bought'].std())

# replacing missing values in forenoon  column with
# minimum number of that column
data['forenoon'] = data['forenoon'].fillna(data['forenoon'].min())

# replacing missing values in afternoon  column with
# maximum number of that column
data['afternoon'] = data['afternoon'].fillna(data['afternoon'].max())
```

	id	item	quantity	price	bought	forenoon	afternoon
0	1	milk	2.0	67.0	672.000000	456.0	567.0
1	2	sugar	1.0	60.5	453.000000	234.0	567.0
2	3	chips	2.5	45.0	456.000000	322.0	567.0
3	4	coffee	2.0	45.0	672.000000	564.0	567.0
4	5	meat	4.0	56.0	786.000000	221.0	567.0
5	6	chocos	3.0	60.5	345.000000	221.0	213.0
6	7	juice	1.0	78.0	765.000000	221.0	344.0
7	8	jam	2.5	65.0	665.000000	221.0	333.0
8	9	bread	3.0	60.5	162.022706	221.0	567.0
9	10	butter	4.0	60.5	162.022706	221.0	322.0

Descriptive Statistics

Introduction

- Descriptive statistics are a set of techniques used to **summarize and describe the main features of a dataset.**

- These statistics provide a **concise overview** of the essential characteristics of the data, helping to **simplify large amounts of information and make it more understandable.**

- Descriptive statistics provide valuable insights into the basic properties of a dataset, helping researchers and analysts understand the **central tendencies, variability, and distribution of the data.**

- The summaries of data may be either **numerical representations or visualizations with simple graphs** for further understanding

Measures of Central Tendency

Central tendency is a statistical measure that **represents the center or middle of a distribution of data values**. It provides a summary of the "typical" or "central" value around which the data points tend to cluster

1. **Mean (Average):** The mean is calculated by summing up all the values in a dataset and then dividing by the number of observations. **It is highly sensitive to extreme values (outliers) in the dataset.**
2. **Median:** The median is the middle value in a dataset when it is ordered from smallest to largest. **If there is an even number of observations, the median is the average of the two middle values**. The median is less sensitive to extreme values than the mean and is particularly useful when the data has outliers.
3. **Mode:** The mode is the value or values that occur most frequently in a dataset. A dataset may have no mode, one mode (unimodal), or more than one mode (multimodal).

df.describe()

describe() function

To find descriptive statistics for the entire dataset

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

Measures of Variability (Dispersion)

- Multiple techniques provide the measures of dispersion in ourdataset. Some commonly used methods are
  - **Standard deviation**
  - **Variance**
  - **Minimum and maximum of values**
  - **Range**
  - **Skewness**
  - **Kurtosis**
  - **Percentiles**

Correlation

- Any dataset that we want to analyze will have different fields (that is, columns) of multiple observations (that is, variables) representing differentfacts.

Example (Attributes of House Price Prediction)

- Price
- Bedrooms
- sqft_living
- sqft_lot
- Area

Correlation

- The columns of a dataset are, most probably, related to one anotherbecause they are collected from the same event

- To examine the type of relationships these columns have and to analyze the causes and effects between them, we have to work to find the dependencies that exist among variables.

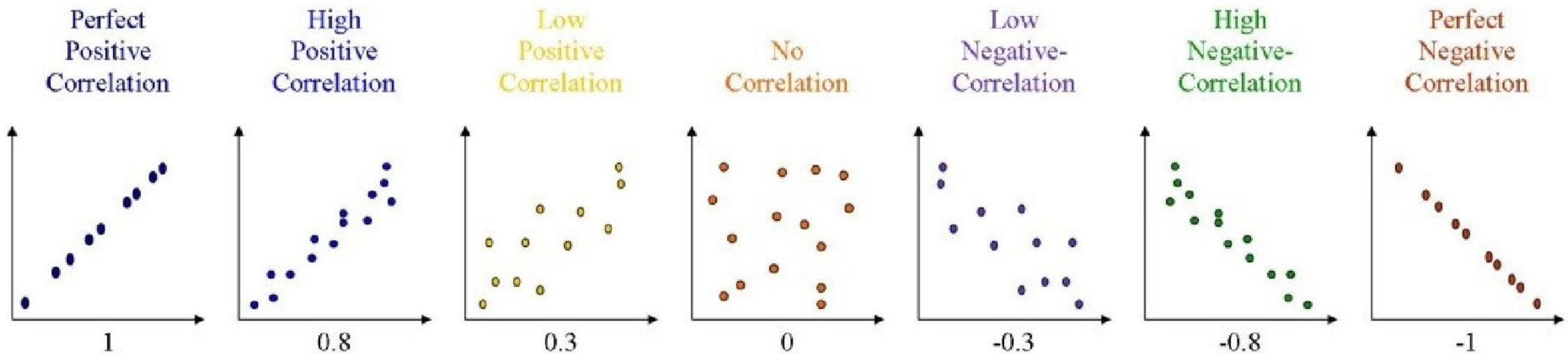
- The strength of such a relationship between two fields of a dataset is called **correlation**, which is represented by a numerical value between -1 and 1.

Correlation

- The statistical technique that examines the relationship and explains whether, and how strongly, pairs of variables are related to oneanother is known as correlation.

- Correlation answers questions such as
  - how one variable changes with respect to another.
  - If it does change, then to what degree or strength?
  - Additionally, if the relation between those variables is strong enough, then we can make predictions for future behavior.

- Correlation tells us how variables change together, both in the sameor opposite directions and in the magnitude (that is, strength) of therelationship


Perfect Positive Correlation
High Positive Correlation
Low Positive Correlation
No Correlation
Low Negative-Correlation
High Negative-Correlation
Perfect Negative Correlation
1
0.8
0.3
0
-0.3
-0.8
-1

Correlation -Analysis

Univariate analysis

- Univariate analysis is the simplest form of analyzing data.
- It means that our data has only **one type of variable** and that we perform analysis over it.
- The **main purpose** of univariate analysis is to take data, **summarize** that data, and **find patterns among the values**.
- It **doesn't deal with** causes or **relationships** between the values.
- Several techniques that describe the patterns found in univariate data include **central tendency** (that is the mean, mode, and median).

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size	fuel-system	bore	stroke	com
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130	mpfi	3.47	2.68	
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130	mpfi	3.47	2.68	
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	0.909722	52.4	2823	ohcv	six	152	mpfi	2.68	3.47	
3	2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	0.919444	54.3	2337	ohc	four	109	mpfi	3.19	3.40	
4	2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	0.922222	54.3	2824	ohc	five	136	mpfi	3.19	3.40	

Univariate analysis

#calculate mean, median and mode of datset height

```
mean = df["height"].mean()
median=df["height"].median()
mode = df["height"].mode()
print(mean , median, mode)
```

Bivariate analysis

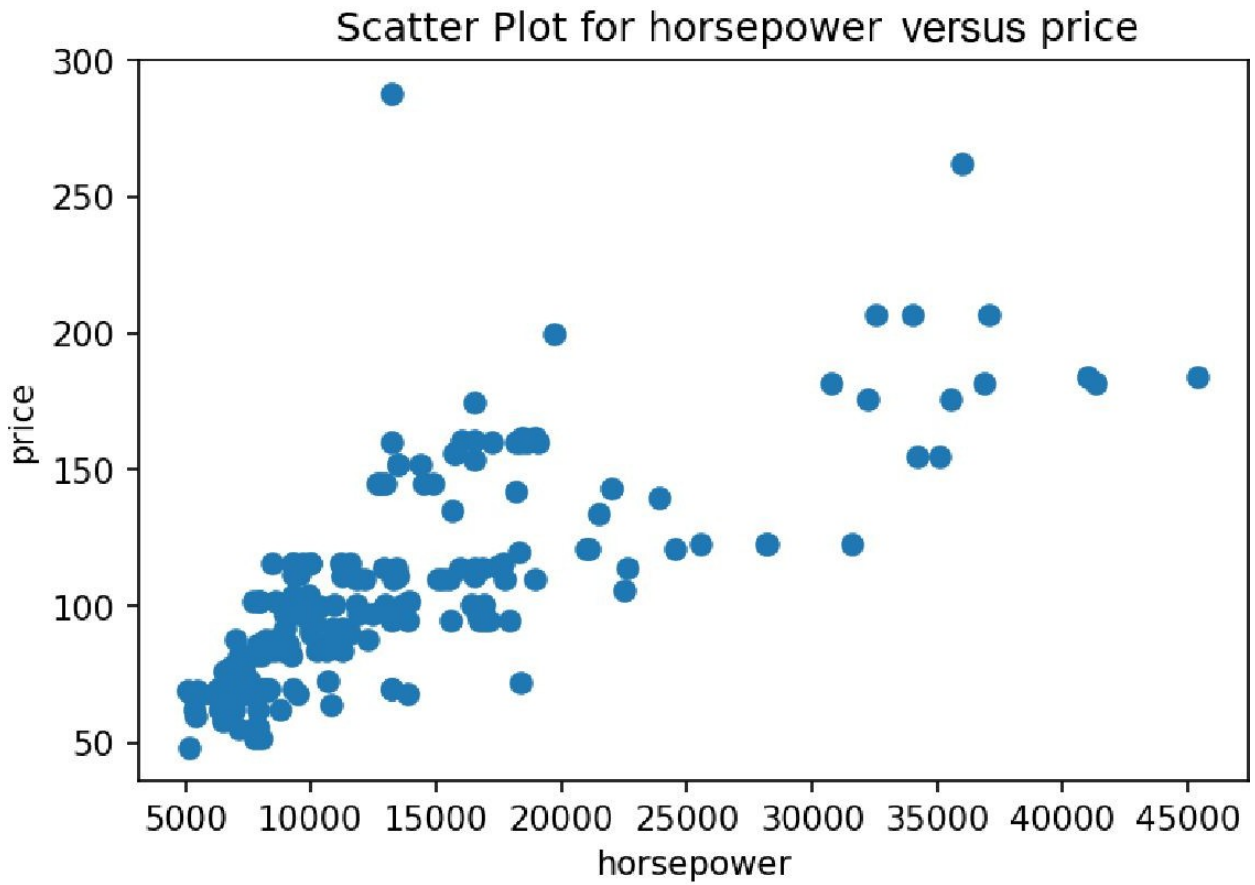
- This is the **analysis of more than one** (that is, exactly two) type of variable.
- Bivariate analysis is **used to find out** whether there is a **relationship between two different variables**.
- When we create a scatter plot by plotting one variable against another on a Cartesian plane (think of the *x* and *y* axes), it gives us a picture of what the data is trying to tell us.
- If the **data points seem to fit the line or curve**, then **there is arelationship** or correlation between the two variables.
- Generally, bivariate analysis helps us to predict a value for one variable (that is, a dependent variable) if we are aware of the value ofthe independent variable.

Bivariate analysis : Example

A scatter plot of advertising dollars and sales rates over a period oftime:

While plotting a scatter plot, we can see that the sales values are dependent on the advertising dollars; that is, as the advertising dollars increase, the sales values also increase.

```
# plot the relationship between “horsepower” and ”price”
plt.scatter(df["price"], df["horsepower"])
plt.title("Scatter Plot for horsepower vs price")
plt.xlabel("horsepower")
plt.ylabel("price")
```



Multivariate analysis

- Multivariate analysis is the **analysis of three or more variables.**
- This allows us to look at correlations (that is, how one variable changes with respect to another) and attempt to make predictions for future behavior more accurately than with bivariate analysis.

Correlation analysis using a heatmap. A heatmap is the best technique to make this look beautiful and easier to interpret