University of Sulaimani
College of Science
Computer Department
4th Stage

Data Science Management

Exploratory Data Analysis (EDA)

Pandas - Python Data Analysis Library

Theoretical and practical lectures

Class 5

Assist. Prof. Dr. Miran Taha Abdullah
2025-2026

Class Agenda

- Review of the previous lecture (Diagram)
- Introduction to Pandas (EDA)
- Main Pandas methods
- Pandas for Data Analysis
- Implementations using Python

Important Modules

1. **NumPy**: Supports large, multi-dimensional arrays and matrices, with a collection of mathematical functions for array operations.
2. **SciPy**: Provides advanced scientific computing tools like optimization, integration, and signal processing.
3. **Statsmodels**: Focuses on statistical modeling and hypothesis testing, offering tools for regression and time series analysis.
4. **Pandas**: ~~Ideal~~ used for data manipulation and analysis, particularly for tabular data using DataFrames.
5. Matplotlib: A library for creating static, interactive, and animated visualizations in Python.
6. Keras: A high-level neural networks API for building and training deep learning models, often used with TensorFlow.
7. Scikit-Learn: Provides tools for machine learning tasks, including classification, regression, clustering, and dimensionality reduction.
8. PyTorch: A deep learning framework known for its dynamic computation graph and strong GPU support.
9. Scrapy: A robust framework for web scraping and extracting data from websites.
10. BeautifulSoup: A library for parsing HTML and XML documents, commonly used for web scraping.
11. TensorFlow: An open-source platform for machine learning, particularly deep learning and neural networks.

Exploratory Data Analysis (EDA) with Pandas

- **Exploratory data analysis (EDA)** is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

The outcomes of Exploratory Data Analysis (EDA) are:

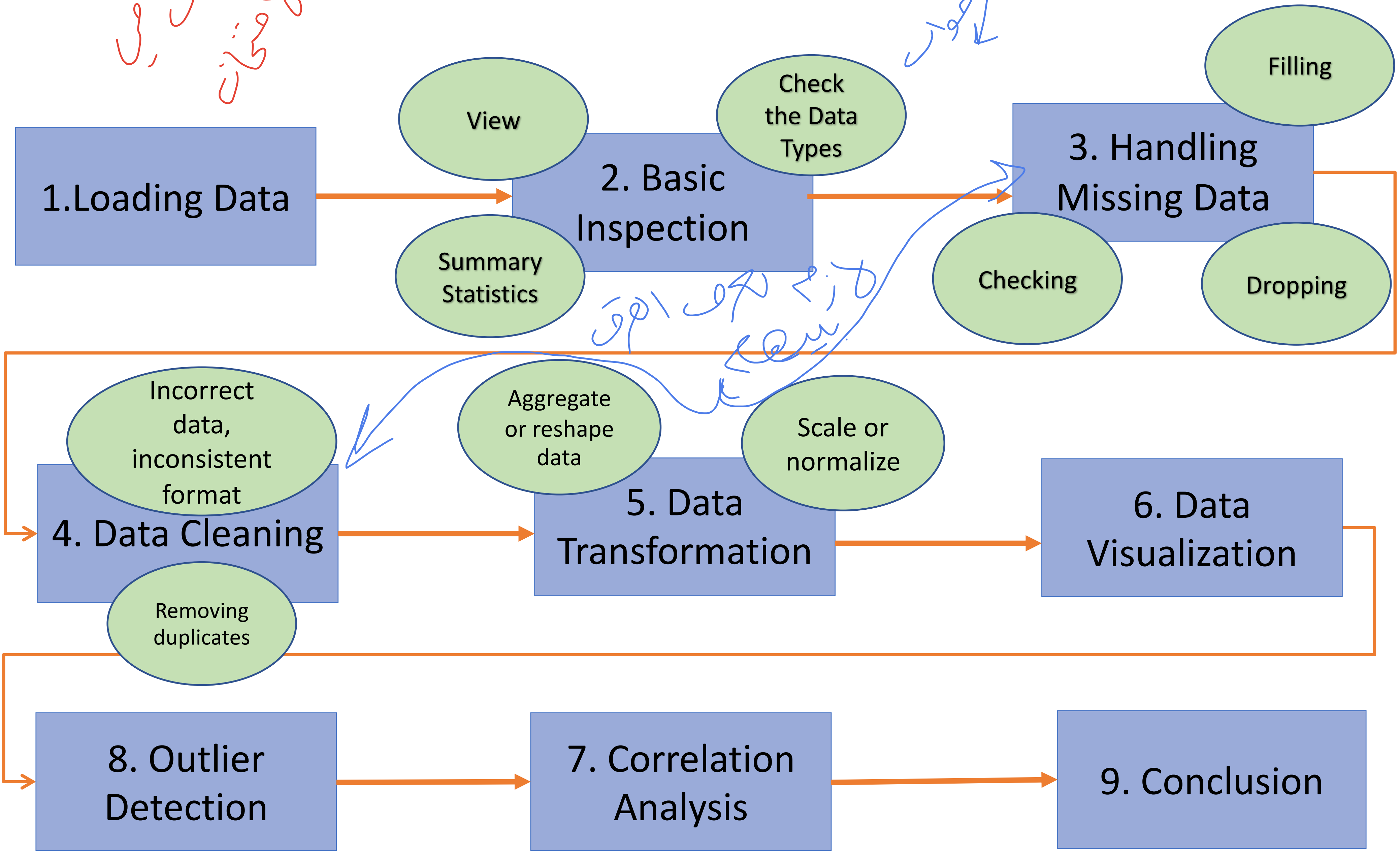
- ➢ Better understanding of the given dataset which helps clean up the data.
- ➢ It gives you a clear picture of the features and the relationships between the data.
- ➢ Providing guidelines for essential variables and non-essential variables.
- ➢ Handling missing values or human error.
- ➢ Identifying outliers.
- ➢ The EDA process would maximize insights into a dataset.
- ➢ It is also helpful when applying machine learning algorithms later on.

مهم في الامتحان

Step-by-step guide for performing **EDA**

سؤال/ اكتب الكلام سرمال

جواب



• Introduction to Pandas

Primary Features:

- Provides DataFrame and Series objects for easy data handling.
- Offers tools for cleaning, transforming, and analyzing data.
- Allows easy manipulation of large datasets.
- Integrates with other Python libraries for visualization and machine learning.

Panel

data

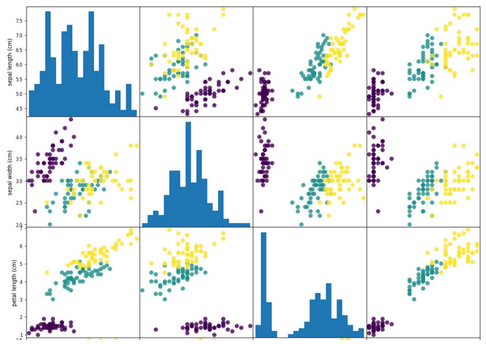
-Simplifies loading,
-Processing,
-Analyzing data.

Pandas

Integrates

Data Processing → **Visualization**

Matplotlib and Seaborn



Advantages of Pandas:

1. **Efficient Data Handling**: Easily handles large datasets with high performance.
2. **Flexible Data Structures**: Provides DataFrame and Series for convenient data manipulation.
3. **SQL-like Operations**: Supports operations like filtering, grouping, and joining, making data analysis simpler.
4. **Integration with Visualization Libraries**: Works well with Matplotlib and Seaborn for graphical data representation.
5. **Wide Range of File Formats Supported**: Supports various file formats like .csv, .xlsx, .json, .sql, and more.

Key Features of Pandas

Data Structures:

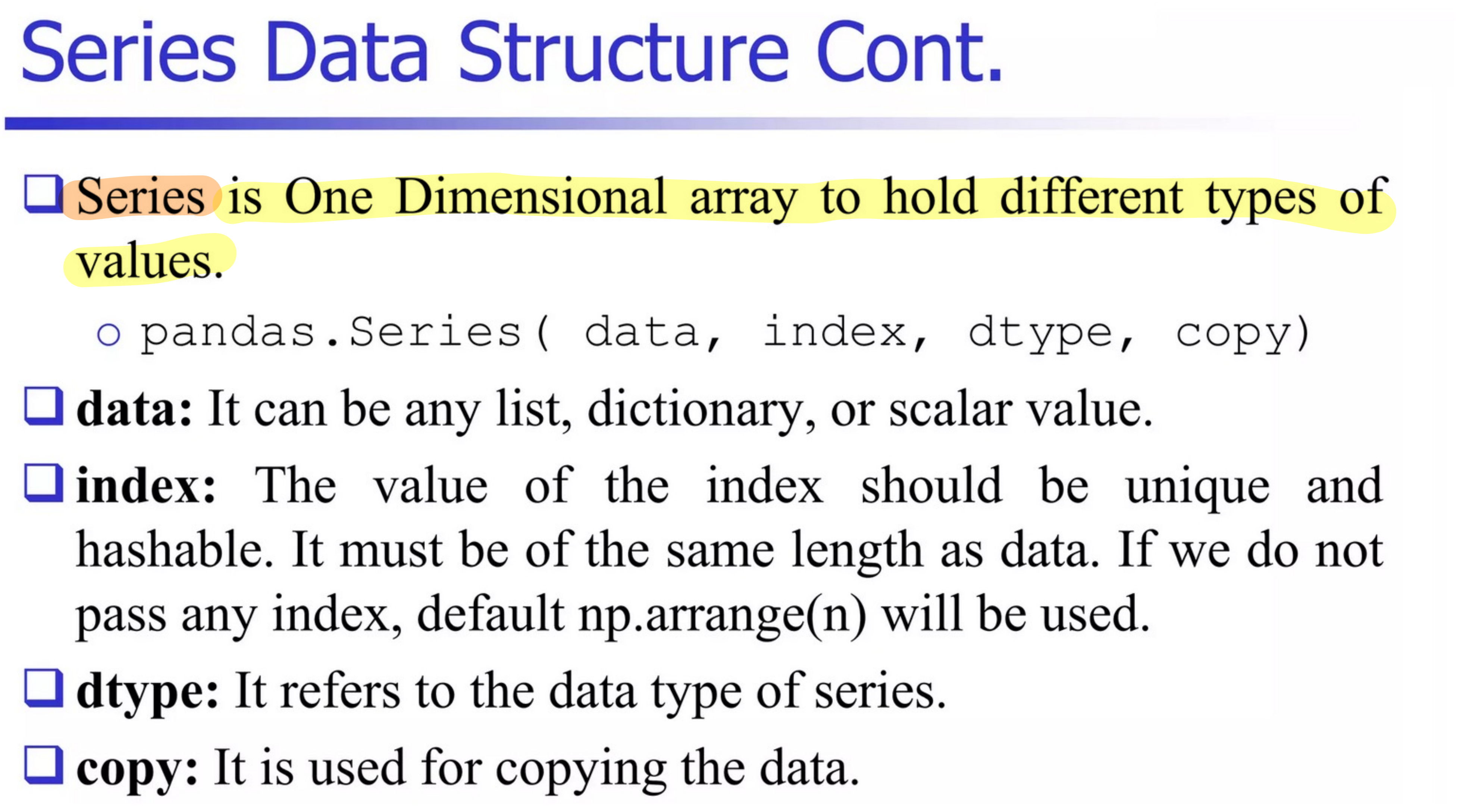
- **Series**: A one-dimensional labeled array (like a list or array).
- **DataFrame**: A two-dimensional, size-mutable, and potentially heterogeneous tabular data structure with labeled axes (rows and columns).

Functions for Data Handling:

- **Loading data** (e.g., pd.read_csv(), pd.read_excel()).
- **Inspecting data** (head(), tail(), info(), describe()).

Operations on Data:

- **Indexing and slicing.**
- **Filtering, sorting, and grouping.**
- **Handling missing data.**



Series Data Structure Cont.

- Series is One Dimensional array to hold different types of values.
  - `pandas.Series( data, index, dtype, copy)`
- **data:** It can be any list, dictionary, or scalar value.
- **index:** The value of the index should be unique and hashable. It must be of the same length as data. If we do not pass any index, default np.arrange(n) will be used.
- **dtype:** It refers to the data type of series.
- **copy:** It is used for copying the data.

pandas.Series(data=None, index=None, dtype=None, name=None, copy=False)

Creating a Series

- We can create Series by using various inputs:
  - Array
  - Dictonary
  - Scalar value

Importing Data:

- pd.read_csv(filename) : It read the data from CSV file.
- pd.read_table(filename) : It is used to read the data from delimited text file.
- pd.read_excel(filename) : It read the data from an Excel file.
- pd.read_sql(query,connection _object) : It read the data from a SQL table/database.
- pd.read_json(json _string) : It read the data from a JSON formatted string, URL or file.

Why does Pandas only display the first 5 and the last 5 rows of a large DataFrame by default, and how can you modify this behavior to view more rows?

Commands and Samples

1. View the first or last rows

df.head()	Show first 5 rows (default)
df.head(10)	Show first 10 rows
df.tail()	Show last 5 rows (default)
df.tail(10)	Show last 10 rows

2. View general information about the dataset

df.info()	Overview: number of rows, columns, non-null count, data types
df.shape	Returns (rows, columns)
df.columns	List of column names
df.dtypes	Data type of each column

3. Summary statistics

df.describe()	Summary of numerical columns: count, mean, std, min, 25%, 50%, 75%, max
df.describe(include='all’)	Summary including categorical columns

Commands and Samples

4. Access specific rows or columns

df['ColumnName']	Access a single column
df[['Col1','Col2']]	Access multiple columns
df.iloc[0]	Access first row by index
df.iloc[0:5]	Access first 5 rows by index
df.loc[0]	Access first row by label
df.loc[:, ['Col1','Col2’]]	Access all rows but specific columns

5. Check for missing or null values

df.isnull()	Returns True/False for each cell
df.isnull().sum()	Count of missing values per column
df.notnull()	Opposite of isnull

6. Check duplicates

df.duplicated()	Returns True/False if row is duplicate
df.duplicated().sum()	Count of duplicate rows

Commands and Samples

7. Check unique values

df['ColumnName'].unique()	Array of unique values
df['ColumnName'].nunique()	Number of unique values
df['ColumnName'].value_counts()	Frequency of each unique value

8. Sorting data

df.sort_values('ColumnName')	Sort by column ascending
df.sort_values('ColumnName', ascending=False)	Descending

9. Sample of data

df.sample(5)	Random 5 rows

10. Transpose data

df.T	Swap rows and columns

```
df_transposed = df.T
```

What are Null Values?

Null values represent **missing data** in a DataFrame.

Types of null values:

for → Numeric

- NaN → Missing numeric values

→ Time

- NaT → Missing datetime values

→ Generic

- None → Generic Python missing value

Na → integer

[illegible]

df.isnull() → Returns **True** if a cell is null, otherwise **False**.

df.isnull().sum()

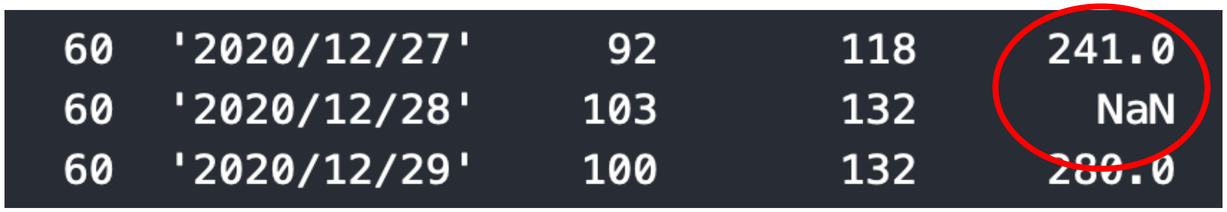
df[df.isnull().any(axis=1)]

notnull() → opposite of isnull(), shows **only non-missing values**.

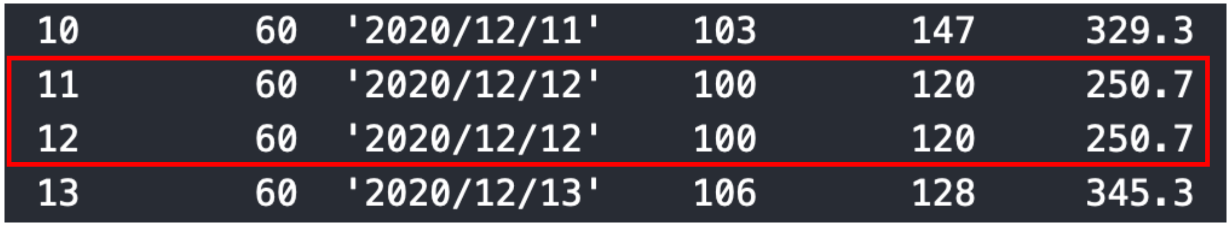
Data Cleaning

Data cleaning means fixing bad data in your data set.

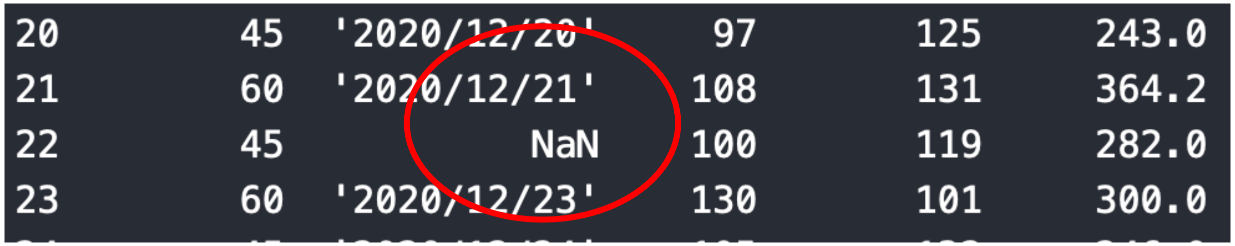
- Empty cells
- Data in wrong format
- Wrong data
- Duplicates



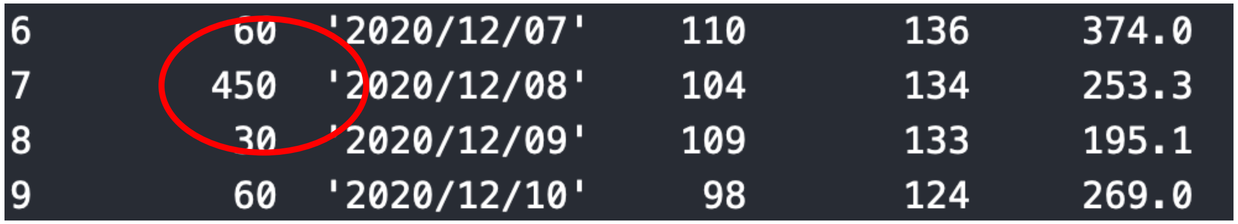
```
 60  '2020/12/27'    92       118      241.0
 60  '2020/12/28'   103       132        NaN
 60  '2020/12/29'   100       132      280.0
```



```
10        60  '2020/12/11'   103      147      329.3
11        60  '2020/12/12'   100      120      250.7
12        60  '2020/12/12'   100      120      250.7
13        60  '2020/12/13'   106      128      345.3
```



```
20        45  '2020/12/20'    97      125      243.0
21        60  '2020/12/21'   108      131      364.2
22        45           NaN   100      119      282.0
23        60  '2020/12/23'   130      101      300.0
```



```
6         60  '2020/12/07'   110      136      374.0
7        450  '2020/12/08'   104      134      253.3
8         30  '2020/12/09'   109      133      195.1
9         60  '2020/12/10'    98      124      269.0
```

Pandas - Cleaning Empty Cells

- ***Empty cells can potentially give you a wrong result when you analyze data.***

1- Remove Rows

One way to deal with empty cells is to **remove rows** that contain **empty cells**. This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

dropna() removes rows or columns with NaN values.

Use axis=0 (default) to drop rows with null values.

Use axis=1 to drop columns with null values.

```
import pandas as pd
df = pd.read_csv('data.csv')
new_df = df.dropna()
Print(new_df)
```

Delete null values from all columns/rows

- In Pandas, when performing operations that modify the original DataFrame (such as dropping rows or columns, or filling missing values), you can use the inplace=True argument to apply the changes directly to the original DataFrame without needing to reassign it to a new variable.
- **inplace=True** modifies the original DataFrame and doesn't return a new object.

  يسوي سيه ويعدل كلها

- If **inplace=False** (or if the argument is not provided), the operation will return a new DataFrame, and you need to assign the result to a variable.

```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],

        'Age': [25, 30, 35, 40],

        'City': ['NY', 'LA', 'SF', 'TX']}

df = pd.DataFrame(data)

df.drop('City', axis=1, inplace=True)

print(df)
```

```
      Name  Age
0    Alice   25
1      Bob   30
2  Charlie   35
3    David   40
```

- **Data visualization** plays a crucial role in **Exploratory Data Analysis (EDA)** as it helps data scientists to **better understand** and **interpret** the underlying patterns and relationships in the data. It is used to support data exploration, cleaning, and preparation tasks.

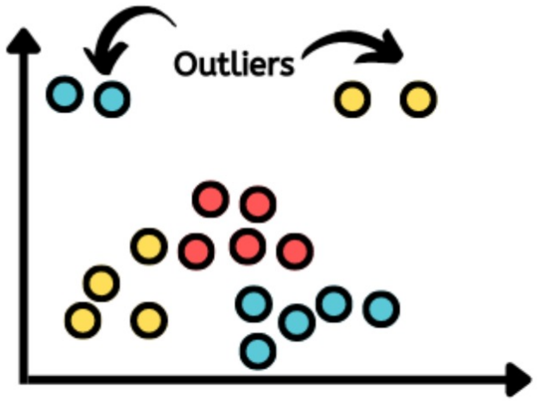
- **Why data visualization is important in the EDA process:**

1-Understanding the structure and relationships in data.

2-Detecting issues such as missing values and outliers.

3-Guiding decisions about data preprocessing, modeling, and analysis.

4-Providing insights that improve the accuracy and effectiveness of analyses.



End of Class 5