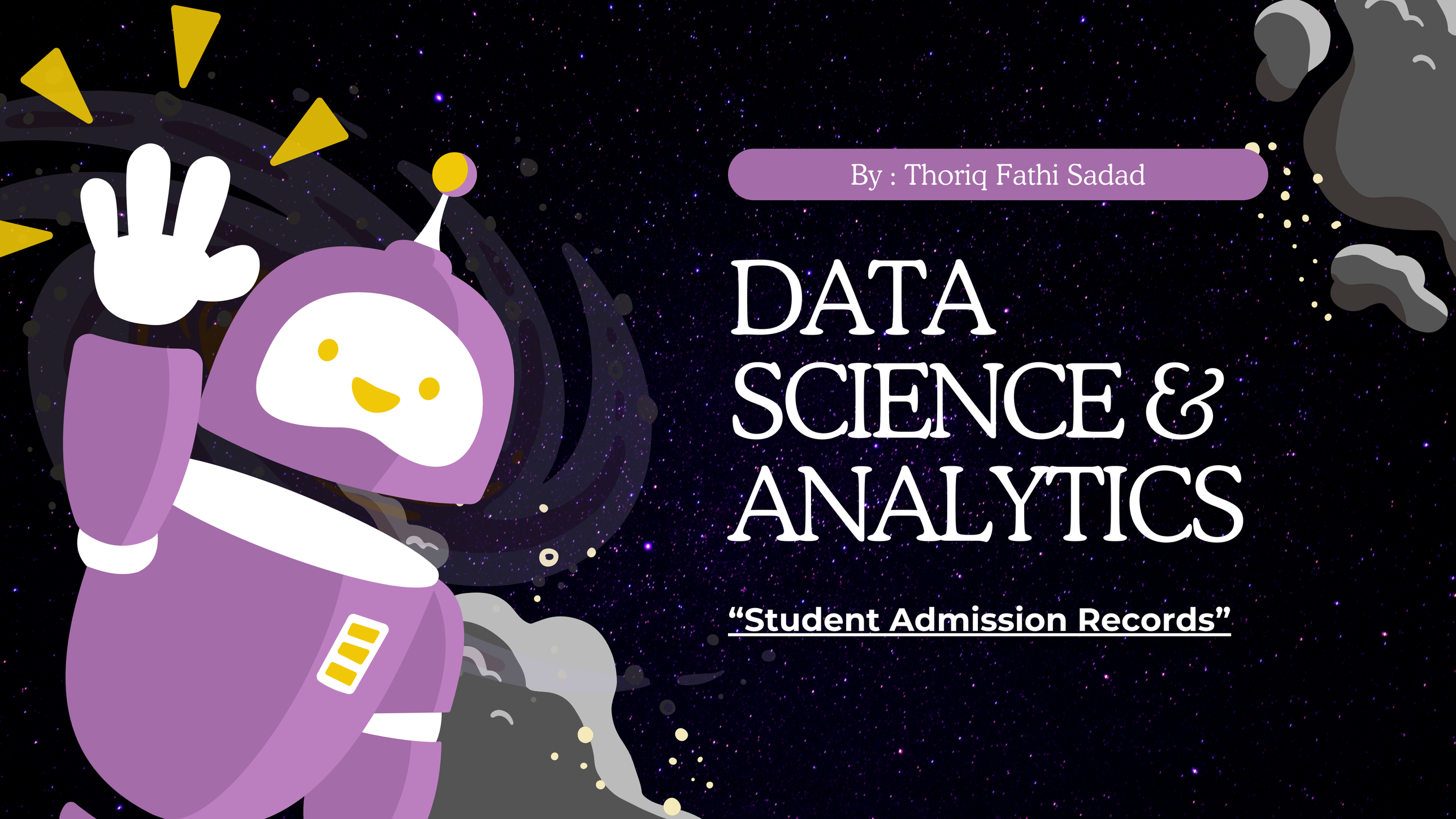
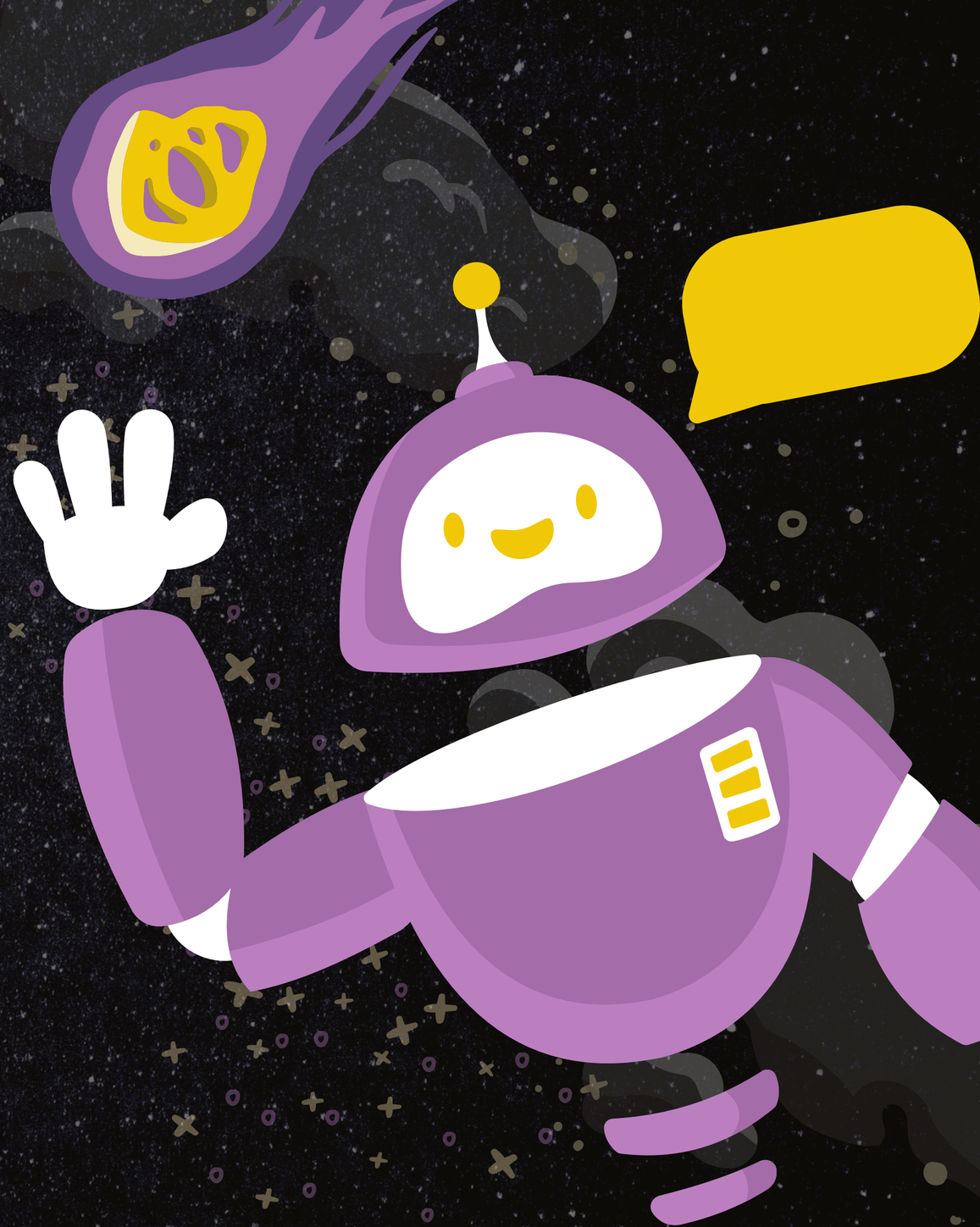

By : Thoriq Fathi Sadad
DATA SCIENCE & ANALYTICS
“Student Admission Records”

Introduction

Hello, my name is Thoriq Fathi Sadad. I am a student at SMK Telkom, and I have a strong interest in data science. I enjoy exploring how data can be transformed into valuable insights, and I’m excited to learn more about topics like data analysis, machine learning, and programming. I believe that data science plays a powerful role in solving real-world problems, and I look forward to growing my skills in this field.



Project objectives

Data Observation

We aim to understand the structure, distribution, and types of data involved in the project.

Handling Duplicate Data

Our goal is to identify and remove redundant entries to ensure data accuracy.

Handling Missing Values

We address incomplete data by applying appropriate strategies such as imputation or removal.

Data

```
[1] import pandas as pd
```

```
df = pd.read_csv('/content/drive/MyDrive/projek dibimbing/student_admission_record_dirty.csv')
df.head()
```

	Name	Age	Gender	Admission Test Score	High School Percentage	City	Admission Status
0	Shehroz	24.0	Female	50.0	68.90	Quetta	Rejected
1	Waqar	21.0	Female	99.0	60.73	Karachi	NaN
2	Bushra	17.0	Male	89.0	NaN	Islamabad	Accepted
3	Aliya	17.0	Male	55.0	85.29	Karachi	Rejected
4	Bilal	20.0	Male	65.0	61.13	Lahore	NaN

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
[3] df.head()
```

	Name	Age	Gender	Admission Test Score	High School Percentage	City	Admission Status
0	Shehroz	24.0	Female	50.0	68.90	Quetta	Rejected
1	Waqar	21.0	Female	99.0	60.73	Karachi	NaN
2	Bushra	17.0	Male	89.0	NaN	Islamabad	Accepted
3	Aliya	17.0	Male	55.0	85.29	Karachi	Rejected
4	Bilal	20.0	Male	65.0	61.13	Lahore	NaN

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
[4] df.tail()
```

	Name	Age	Gender	Admission Test Score	High School Percentage	City	Admission Status
152	Ali	19.0	Female	85.0	78.09	Quetta	Accepted
153	Bilal	17.0	Female	81.0	84.40	Islamabad	Rejected
154	Fatima	21.0	Female	98.0	50.86	Multan	Accepted
155	Shoaib	-1.0	Male	91.0	80.12	Quetta	Accepted
156	Maaz	17.0	Male	88.0	86.85	Lahore	Accepted

import pandas as pd: Imports the Pandas library, which is commonly used for data analysis and manipulation in Python

pd.read_csv(...): Loads the CSV file located in your Google Drive into a DataFrame called df

df.head(): used to display the 5 data from the bottom

About the data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157 entries, 0 to 156
Data columns (total 7 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Name                     147 non-null    object 
 1   Age                      147 non-null    float64
 2   Gender                   147 non-null    object 
 3   Admission Test Score     146 non-null    float64
 4   High School Percentage   146 non-null    float64
 5   City                     147 non-null    object 
 6   Admission Status         147 non-null    object 
dtypes: float64(3), object(4)
memory usage: 8.7+ KB
```

- Rows (entries): 157
- Columns: 7
  - Name (object)
  - Age (float64)
  - Gender (object)
  - Admission Test Score (float64)
  - High School Percentage (float64)
  - City (object)
  - Admission Status (object)

❗ Missing Values:

From .info():

- Admission Test Score: 146 non-null → 11 missing
- High School Percentage: 146 non-null → 11 missing
- Name, Age, Gender, City, Admission Status: 147 non-null → 10 missing total across them

Statistical summary

```
df.describe()
```

	Age	Admission Test Score	High School Percentage
count	147.000000	146.000000	146.000000
mean	19.680272	77.657534	75.684726
std	4.540512	16.855343	17.368014
min	-1.000000	-5.000000	-10.000000
25%	18.000000	68.250000	65.052500
50%	20.000000	79.000000	77.545000
75%	22.000000	89.000000	88.312500
max	24.000000	150.000000	110.500000

A statistical summary gives you a quick overview of the numerical data in your dataset. It helps you understand the central tendency, spread, and shape of the data distribution.

Duplicate Analysis & Handle Duplicates

```python
# Check for full row duplicates
duplicate_rows = df[df.duplicated(keep=False)]  # Show all instances of duplicates
duplicate_count = df.duplicated().sum()

# Drop duplicates (for later)
df_no_duplicates = df.drop_duplicates()

duplicate_count, duplicate_rows
```

```
(np.int64(6),
        Name   Age  Gender  Admission Test Score  High School Percentage  \
3     Aliya  17.0    Male                  55.0                   85.29
9    Kamran  18.0    Male                  53.0                   98.98
13    Ahmed  21.0    Male                  62.0                   79.03
22   Kamran  18.0    Male                  53.0                   98.98
36   Ayesha  24.0    Male                  94.0                   98.43
52    Ahmed  21.0    Male                  62.0                   79.03
61    Hamza  22.0    Male                  99.0                   86.58
69     Tuba  17.0  Female                  75.0                   78.43
88    Aliya  17.0    Male                  55.0                   85.29
110   Hamza  22.0    Male                  99.0                   86.58
120  Ayesha  24.0    Male                  94.0                   98.43
122    Tuba  17.0  Female                  75.0                   78.43

         City Admission Status
3     Karachi         Rejected
9      Multan         Rejected
13    Karachi         Accepted
```

Duplicate Analysis Results

- Total duplicate rows: 6
- These are full row duplicates (all column values are the same).
- Here are some examples:
  - "Aliya", 17, Male, 55.0, 85.29, Karachi, Rejected appears more than once.
  - "Ahmed", 21, Male, 62.0, 79.03, Karachi, Accepted appears more than once.

```python
df_cleaned = df.drop_duplicates()
# Remove duplicates
```

To clean the data, you can remove exact duplicates like this....

This will retain only the first occurrence of each duplicate.

Handle missing values

```
#Detecting Missing Data
df.isnull().sum()
```

This will show how many values are missing in each column.

```
#Filling Missing Values
df_filled = df.fillna({
    "Name": "Unknown",
    "Age": 0,
    "Gender": "Unknown",
    "Admission Test Score": 0,
    "High School Percentage": 0,
    "City": "Unknown",
    "Admission Status": "Unknown"
})

#b. With the mean/median (for numerical data)
df["Age"].fillna(df["Age"].mean(), inplace=True)
df["Admission Test Score"].fillna(df["Admission Test Score"].median(), inplace=True)

#c. With the mode (for categorical data)
df["Gender"].fillna(df["Gender"].mode()[0], inplace=True)
df["City"].fillna(df["City"].mode()[0], inplace=True)
df["Admission Status"].fillna(df["Admission Status"].mode()[0], inplace=True)
```

- This method fills all missing values with a predefined constant.
- It's useful for quick cleaning or when we need placeholder values.
- However, it can be problematic:
- Filling numeric columns like "Age" or "Test Score" with 0 may distort the data.
- Zero could be interpreted as a valid score rather than missing data.

- We use mean for "Age" because age values tend to follow a normal distribution.
- We use median for "Admission Test Score" to reduce the effect of outliers or extreme values.
- This preserves the central tendency of the data without removing any rows.

- The mode is the most frequently occurring value in a column.
- It is ideal for categorical data such as "Gender", "City", and "Admission Status".
- Filling with the mode ensures consistency and does not introduce unknown or rare values.

```
#Forward Fill / Backward Fill
df.fillna(method='ffill', inplace=True)  # forward fill
df.fillna(method='bfill', inplace=True)  # backward fill
```

Purpose: Filling Missing Values with Adjacent Data

- When working with sequential or time-based data, one common strategy for handling missing values is to propagate values from nearby cells. This is where forward fill and backward fill come into play.

Forward Fill (method='ffill')

What it does:
Fills missing values by copying the last known (non-missing) value downward into the next missing cell.

Backward Fill (method='bfill')

What it does:
Fills missing values by copying the next known value upward into the previous missing cell.

Thank You!