zenius × Kampus Merdeka INDONESIA JAYA

Python for Data Analysis: DataFrame Basics and Data Cleansing

Put the Date Here

Accelerated Machine Learning Program

Program Studi Independen Bersertifikat
Zenius Bersama Kampus Merdeka

1. What is Pandas
2. Creating a Dataframe
3. Basic Operations
4. Columns Reordering
5. Data Filtering
6. Data Cleansing (Handle Missing Values & Outliers)

Pandas

https://github.com/pandas-dev/pandas

An open-sourced Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive.

Doing data analysis in Python.

What is Pandas?

Introducing to Pandas library

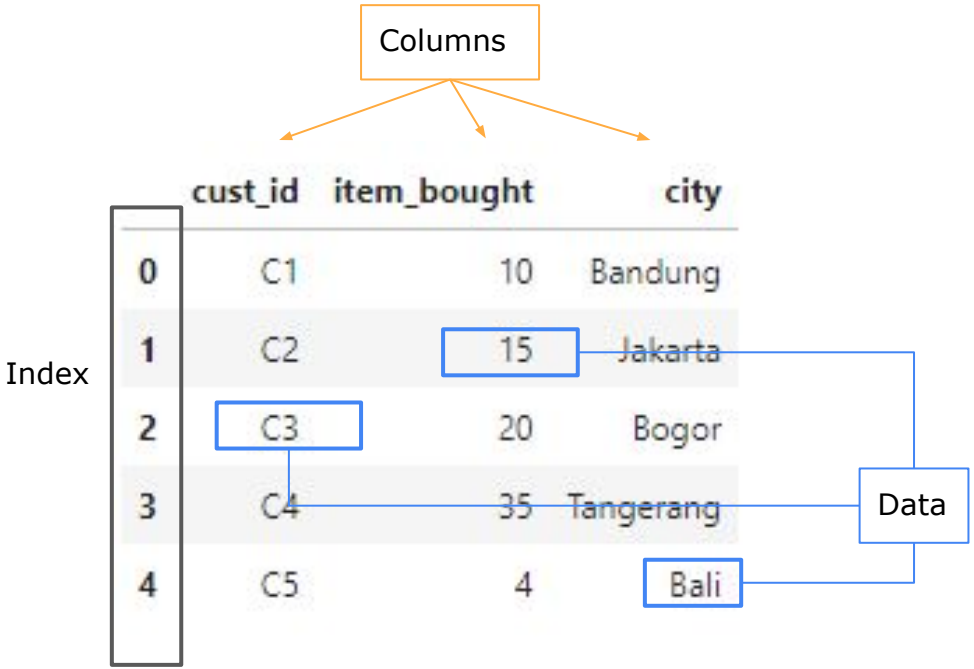
How to Install & Import?

```
pip install pandas
```

```
conda install pandas
```

```
import pandas as pd
```

Intro to Pandas DataFrame



Intro to Pandas Series

```
0    C1
1    C2
2    C3
3    C4
4    C5
Name: cust_id, dtype: object
```

```
0    10
1    15
2    20
3    35
4     4
Name: item_bought, dtype: int64
```

```
0       Bandung
1       Jakarta
2         Bogor
3     Tangerang
4          Bali
Name: city, dtype: object
```

Creating DataFrame

Create DataFrame from scratch,
save and load DataFrame

Creating DataFrame

There are several ways to create a Pandas dataframe:

1. Creating Dataframe from **List of List**

```
df = pd.DataFrame([["C1", 10, "Bandung"],
                   ["C2", 15, "Jakarta"],
                   ["C3", 20, "Bogor"],
                   ["C4", 35, "Tangerang"],
                   ["C5", 4, "Bali"]],
                   columns = ["cust_id", "item_bought", "city"])
df
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	15	Jakarta
2	C3	20	Bogor
3	C4	35	Tangerang
4	C5	4	Bali

Creating DataFrame

There are several ways to create a Pandas dataframe:

2. Creating Dataframe from **Numpy Array**

```
df = pd.DataFrame(np.array([["C1", 10, "Bandung"],
                            ["C2", 15, "Jakarta"],
                            ["C3", 20, "Bogor"],
                            ["C4", 35, "Tangerang"],
                            ["C5", 4, "Bali"]]),
                            columns = ["cust_id", "item_bought", "city"])
df
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	15	Jakarta
2	C3	20	Bogor
3	C4	35	Tangerang
4	C5	4	Bali

Creating DataFrame

There are several ways to create a Pandas dataframe:

3. Creating Dataframe from **Dictionary of List**

```
df = pd.DataFrame({
    "cust_id": ["C1", "C2", "C3", "C4", "C5"],
    "item_bought": [10,15,20,35,4],
    "city": ["Bandung", "Jakarta", "Bogor", "Tangerang", "Bali"]
    })
df
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	15	Jakarta
2	C3	20	Bogor
3	C4	35	Tangerang
4	C5	4	Bali

Intro to CSV

- CSV: Comma-Separated Values
- A **delimited** text file that used **comma** to **separate** values from different columns
- What makes CSV more popular than excel?
  - Faster
  - Smaller in size
  - Simple schema
  - More human-readable than JSON or XML
  - Very easy to handle (understood by almost every piece of software, even in Excel)

Saving a DataFrame

Pandas can save to both CSV and Excel format.

```
df.to_csv('figures.csv')
```

```
df.to_excel('figures.xlsx')
```

Load a DataFrame

Pandas can load both CSV and Excel files.

```
# Load CSV
df = pd.read_csv("path/to/file.csv")
```

```
# Load Excel
df = pd.read_excel("path/to/file.xlsx")
```

Basic Operations

Basic Operations using Pandas

Extracting Basic DataFrame Info

- Summary info
- Statistics of numerical columns
- Columns name
- Dataframe length
- Values of a particular column
- Values of several columns
- First N rows of dataframe
- Last N rows of dataframe
- Random N rows of dataframe

Extracting Basic DataFrame Info

- Summary info

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   cust_id      5 non-null      object
 1   item_bought  5 non-null      int64 
 2   city         5 non-null      object
dtypes: int64(1), object(2)
memory usage: 248.0+ bytes
```

Extracting Basic DataFrame Info

- Statistics of numerical columns

```
: df.describe()
```

:

	item_bought
count	5.000000
mean	16.800000
std	11.777096
min	4.000000
25%	10.000000
50%	15.000000
75%	20.000000
max	35.000000

Extracting Basic DataFrame Info

- Columns name

```
df.columns
```

```
Index(['cust_id', 'item_bought', 'city'], dtype='object')
```

Extracting Basic DataFrame Info

- Dataframe length

```
: df.shape

: (5, 3)

: df.shape[0]

: 5

: df.shape[1]

: 3
```

Extracting Basic DataFrame Info

- Values of a particular column

```
df["cust_id"]
```

```
0    C1
1    C2
2    C3
3    C4
4    C5
Name: cust_id, dtype: object
```

Extracting Basic DataFrame Info

- Values of several columns

```
df[["cust_id","city"]]
```

	cust_id	city
0	C1	Bandung
1	C2	Jakarta
2	C3	Bogor
3	C4	Tangerang
4	C5	Bali

Extracting Basic DataFrame Info

- First N rows of dataframe

```
df.head()
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	15	Jakarta
2	C3	20	Bogor
3	C4	35	Tangerang
4	C5	4	Bali

```
df.head(2)
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	15	Jakarta

Extracting Basic DataFrame Info

- Last N rows of dataframe

```
df.tail()
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	15	Jakarta
2	C3	20	Bogor
3	C4	35	Tangerang
4	C5	4	Bali

```
df.tail(2)
```

	cust_id	item_bought	city
3	C4	35	Tangerang
4	C5	4	Bali

Extracting Basic DataFrame Info

- Random N rows of dataframe

```
df.sample()
```

	cust_id	item_bought	city
0	C1	10	Bandung

```
df.sample(3)
```

	cust_id	item_bought	city
0	C1	10	Bandung
2	C3	20	Bogor
3	C4	35	Tangerang

Columns Reordering

Reordering Columns in A DataFrame

```
df[["item_bought", "city", "cust_id"]]
```

	item_bought	city	cust_id
0	10	Bandung	C1
1	15	Jakarta	C2
2	20	Bogor	C3
3	35	Tangerang	C4
4	4	Bali	C5

Reordering Columns in A DataFrame

```
df = df.reindex(columns=["item_bought", "city", "cust_id"])
df
```

	item_bought	city	cust_id
0	10	Bandung	C1
1	15	Jakarta	C2
2	20	Bogor	C3
3	35	Tangerang	C4
4	4	Bali	C5

Data Filtering

Data Filtering

- Get the value of specific row and column (via its name)
- Get the value of specific row and column (via its index num)
- Get the specific ranges of rows and columns (via its name)
- Get the specific ranges of rows and columns (via its index num)
- Filter dataframe with a criteria
- Filter dataframe with several criterias (AND)
- Filter dataframe with several criterias (OR)

Data Filtering

- Get the value of specific row and column (via its name)

```
df.loc[0,"item_bought"]
```

```
10
```

```
df.loc[3,"city"]
```

```
'Tangerang'
```

Data Filtering

- Get the value of specific row and column (via its index num)

```
df.iloc[0,0]
```

```
10
```

```
df.iloc[3,1]
```

```
'Tangerang'
```

Data Filtering

- Get the specific ranges of rows and columns (via its name)

```
df.loc[0:2,["item_bought","city"]]
```

	item_bought	city
0	10	Bandung
1	15	Jakarta
2	20	Bogor

Data Filtering

- Get the specific ranges of rows and columns (via its index num)

```
df.iloc[0:3,0:2]
```

	item_bought	city
0	10	Bandung
1	15	Jakarta
2	20	Bogor

Data Filtering

- Filter dataframe with a criteria

```
df[df["item_bought"]>15]
```

	item_bought	city	cust_id
2	20	Bogor	C3
3	35	Tangerang	C4

Data Filtering

- Filter dataframe with several criterias (AND)

```
df[(df["item_bought"]>15) & (df["city"]!= "Bogor")]
```

	item_bought	city	cust_id
3	35	Tangerang	C4

Data Filtering

- Filter dataframe with several criterias (OR)

```
df[(df["item_bought"]>15) | (df["city"]!= "Bogor")]
```

	item_bought	city	cust_id
0	10	Bandung	C1
1	15	Jakarta	C2
2	20	Bogor	C3
3	35	Tangerang	C4
4	4	Bali	C5

Data Cleansing (Handle Missing Values & Outliers)

Data Cleansing

- Check Missing Values in DataFrame
- Handle Missing Values
- Check Outliers in DataFrame
- Handle Outliers
- Check Duplicated Data
- Handle Duplicates

Data Cleansing

- Check Missing Values in DataFrame

```
df = pd.DataFrame({
    "cust_id": ["C1", "C2", "C3", "C4", "C5", "C6"],
    "item_bought": [10,15,20,35,4, np.nan],
    "city": ["Bandung", "Jakarta", "Bogor", "Tangerang", "Bali", np.nan]
    })
df
```

	cust_id	item_bought	city
0	C1	10.0	Bandung
1	C2	15.0	Jakarta
2	C3	20.0	Bogor
3	C4	35.0	Tangerang
4	C5	4.0	Bali
5	C6	NaN	NaN

```
df.isna().sum()
```

```
cust_id        0
item_bought    1
city           1
dtype: int64
```

Data Cleansing

- Handle Missing Values: Remove Rows

```
#Remove rows
```

```
df.dropna()
```

	cust_id	item_bought	city
0	C1	10.0	Bandung
1	C2	15.0	Jakarta
2	C3	20.0	Bogor
3	C4	35.0	Tangerang
4	C5	4.0	Bali

Data Cleansing

- Handle Missing Values: Fill with 0

```
df["item_bought"] = df["item_bought"].fillna(0)
df
```

	cust_id	item_bought	city
0	C1	10.0	Bandung
1	C2	15.0	Jakarta
2	C3	20.0	Bogor
3	C4	35.0	Tangerang
4	C5	4.0	Bali
5	C6	0.0	NaN

Data Cleansing

- Handle Missing Values: Fill with mean/median

```
df["item_bought"] = df["item_bought"].fillna(df["item_bought"].mean())
df
```

	cust_id	item_bought	city
0	C1	10.0	Bandung
1	C2	15.0	Jakarta
2	C3	20.0	Bogor
3	C4	35.0	Tangerang
4	C5	4.0	Bandung
5	C6	16.8	NaN

```
df["item_bought"] = df["item_bought"].fillna(df["item_bought"].median())
df
```

	cust_id	item_bought	city
0	C1	10.0	Bandung
1	C2	15.0	Jakarta
2	C3	20.0	Bogor
3	C4	35.0	Tangerang
4	C5	4.0	Bandung
5	C6	15.0	NaN

Data Cleansing

- Handle Missing Values: Fill with mode

```
df["city"] = df["city"].fillna(df["city"].mode()[0])
df
```

	cust_id	item_bought	city
0	C1	10.0	Bandung
1	C2	15.0	Jakarta
2	C3	20.0	Bogor
3	C4	35.0	Tangerang
4	C5	4.0	Bandung
5	C6	15.0	Bandung

Data Cleansing

- Check Outliers in DataFrame

```
#Check outliers
```

```
df = pd.DataFrame({
    "cust_id": ["C1", "C2", "C3", "C4", "C5", "C6", "C7", "C8", "C9", "C10", "C11"],
    "item_bought": [10,7,5,4,10000, 10, 5, 5, 7, 4, 5],
    "city": ["Bandung", "Jakarta", "Bogor", "Tangerang", "Bandung", "Bogor", "Bogor", "Bogor", "Bandung", "Jakarta", "Bogor"]
    })
df
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	7	Jakarta
2	C3	5	Bogor
3	C4	4	Tangerang
4	C5	10000	Bandung
5	C6	10	Bogor
6	C7	5	Bogor
7	C8	5	Bogor
8	C9	7	Bandung
9	C10	4	Jakarta
10	C11	5	Bogor

Data Cleansing

- Check Outliers in DataFrame: Z-Score

```
#Z-Score
```

```
import scipy.stats as stats
```

```
df[(np.abs(stats.zscore(df["item_bought"])) >= 3)]
```

	cust_id	item_bought	city
4	C5	10000	Bandung

Data Cleansing

- Check Outliers in DataFrame: IQR

```
#IQR
```

```
q1 = df["item_bought"].quantile(0.25)
q3 = df["item_bought"].quantile(0.75)

iqr = q3-q1 #Interquartile range
fence_low  = q1-1.5*iqr
fence_high = q3+1.5*iqr

df.loc[(df["item_bought"] < fence_low) | (df["item_bought"] > fence_high)]
```

	cust_id	item_bought	city
4	C5	10000	Bandung

Data Cleansing

- Handle Outliers: Remove Rows

```
#Z-Score
df_clean = df[(np.abs(stats.zscore(df["item_bought"])) < 3)]

#IQR
df_clean = df.loc[(df["item_bought"] >= fence_low) & (df["item_bought"] <= fence_high)]

df_clean
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	7	Jakarta
2	C3	5	Bogor
3	C4	4	Tangerang
5	C6	10	Bogor
6	C7	5	Bogor
7	C8	5	Bogor
8	C9	7	Bandung
9	C10	4	Jakarta
10	C11	5	Bogor

Data Cleansing

- Check Duplicated Data

```
#Check Duplicates
```

```
df = pd.DataFrame({
    "cust_id": ["C1", "C2", "C3", "C4", "C5", "C6", "C7", "C8", "C9", "C10", "C7"],
    "item_bought": [10,7,5,4,10000, 10, 5, 5, 7, 4, 5],
    "city": ["Bandung", "Jakarta", "Bogor", "Tangerang", "Bandung", "Bogor", "Bogor", "Bogor", "Bandung", "Jakarta", "Bogor"]
    })
df
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	7	Jakarta
2	C3	5	Bogor
3	C4	4	Tangerang
4	C5	10000	Bandung
5	C6	10	Bogor
6	C7	5	Bogor
7	C8	5	Bogor
8	C9	7	Bandung
9	C10	4	Jakarta
10	C7	5	Bogor

```
df.duplicated().sum()
```

```
1
```

Data Cleansing

- Handle Duplicates

```
#Handle Duplicates
```

```
df.drop_duplicates()
```

	cust_id	item_bought	city
0	C1	10	Bandung
1	C2	7	Jakarta
2	C3	5	Bogor
3	C4	4	Tangerang
4	C5	10000	Bandung
5	C6	10	Bogor
6	C7	5	Bogor
7	C8	5	Bogor
8	C9	7	Bandung
9	C10	4	Jakarta

Terima kasih!

Ada pertanyaan?