

Exploratory Data Analysis dengan Python

Topik

- Pengenalan Library dalam Python
 - Library NumPy
 - Library Pandas
 - Library SciPy
 - Library Matplotlib
 - Memanggil library di Python
- Exploratory Data Analysis dengan Pandas – Bagian 1
 - Membaca file dari Excel atau CSV sebagai data frame
 - Inspeksi struktur data frame
 - Melihat struktur kolom dan baris dari data frame
 - Melihat preview data dari data frame
 - Statistik Deskriptif dari Data Frame — Part 1
 - Statistik Deskriptif dari Data Frame — Part 2
 - Statistik Deskriptif dari Data Frame — Part 3
- Exploratory Data Analysis dengan Pandas - Bagian 2
 - Mengenal dan Membuat Distribusi Data dengan Histogram
 - Standar Deviasi dan Varians pada Pandas
 - Menemukan Outliers Menggunakan Pandas
 - Rename Kolom Data Frame
 - .groupby menggunakan Pandas
 - Sorting Menggunakan Pandas

Exploratory Data Analysis dengan Python

- *Exploratory Data Analysis* (EDA) adalah tahapan penting sebelum menganalisis suatu data karena EDA digunakan untuk mempelajari karakteristik dari data tersebut.
- Karakteristik data yang dipelajari berupa memeriksa kesalahan-kesalahan data yang mungkin terjadi pada berbagai tahap, mulai dari pencatatan data di lapangan sampai pada *entry* data pada komputer.
- Selain itu, EDA dapat digunakan untuk eksplorasi data sehingga kita bisa menentukan model analisis yang tepat.

Pengenalan Library dalam Python

Pengenalan Library Python

- Dalam menganalisis data, kita tidak asing lagi dengan bahasa pemrograman — Python.
- Dalam topik ini akan dibahas kegunaan masing — masing library dasar pada Python beserta contoh kasus penggunaan library tersebut dalam menyelesaikan real — case di dunia kerja.

Macam-macam Library pada Python

- Library pada Python merupakan kumpulan code yang bersifat open-source yang dapat dipanggil ke dalam Python dan digunakan untuk membantu komputasi.
- Library dasar pada Python yang digunakan untuk analisis data:
 - Library NumPy
 - Library Pandas
 - Library SciPy
 - Library Matplotlib
- Tiap library memiliki fungsional yang berbeda — beda.
- Selanjutnya akan dibahas apa saja kegunaan dari masing-masing *library* beserta contoh kasus penggunaannya dalam menyelesaikan berbagai *real case* di dunia kerja.

Library NumPy

- Numpy berasal dari kata '*Numerical Python*'
- Sesuai namanya NumPy berfungsi sebagai library untuk melakukan proses komputasi numerik terutama dalam bentuk *array* multidimensional (1-Dimensi ataupun 2-Dimensi).
- *Array* merupakan kumpulan dari variabel yang memiliki tipe data yang sama.
- NumPy menyimpan data dalam bentuk *arrays*.
- Bentuk 1D NumPy *array* dapat diilustrasikan sebagai berikut:

--	--	--	--	--

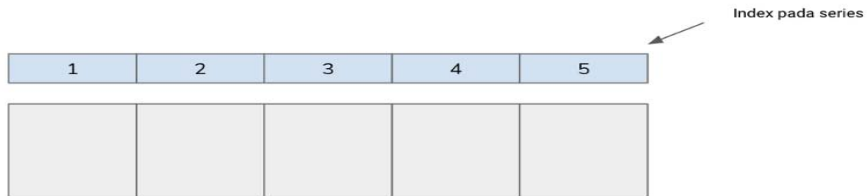
- Bentuk 2D NumPy *array* dapat diilustrasikan sebagai berikut:

Library Pandas

- Pandas merupakan library yang memudahkan dalam melakukan manipulasi, *cleansing* maupun analisis struktur data.
- Dengan menggunakan Pandas, dapat memanfaatkan lima fitur utama dalam pemrosesan dan analisis data, yaitu:
 - *Load data*
 - *Prepare data*
 - *Manipulate data*
 - *Modelling data*
 - *Analysis data*
- Pandas menggunakan konsep array dari NumPy namun memberikan **index** kepada array tersebut, sehingga disebut ***series*** ataupun ***data frame***.
- Sehingga bisa dikatakan Pandas menyimpan data dalam *dictionary-based* NumPy arrays.
- 1-Dimensi labelled array dinamakan sebagai ***Series***.
- Sedangkan 2-Dimensi dinamakan sebagai ***Data Frame***.

Library Pandas

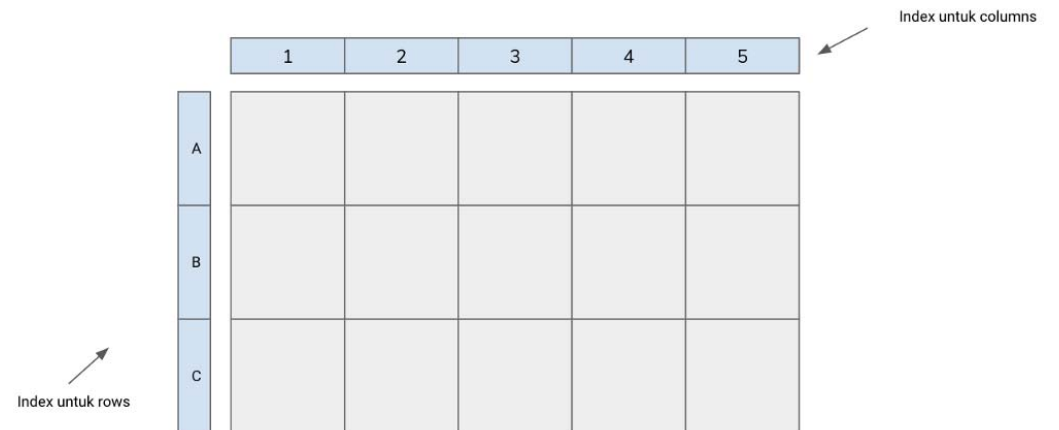
Bentuk 1D Pandas (Series)



1	2	3	4	5

Index pada series

Bentuk 2D Pandas (Data Frame)



	1	2	3	4	5
A					
B					
C					

Index untuk columns

Index untuk rows

Library SciPy

- Scipy dibangun untuk bekerja dengan array NumPy dan menyediakan banyak komputasi numerik yang ramah pengguna dan efisien seperti rutinitas untuk **integrasi**, **diferensiasi** dan **optimasi numerik**.
- Baik NumPy maupun SciPy berjalan pada semua operating system, cepat dan mudah untuk diinstall dan gratis.
- NumPy dan SciPy mudah digunakan, dan kehandalannya membuat beberapa data *scientist* dan *researcher* terkemuka dunia memilih menggunakannya.

Library Matplotlib

- Matplotlib merupakan library dari Python yang umum digunakan untuk visualisasi data.
- Matplotlib memiliki kapabilitas untuk membuat visualisasi data 2-dimensional.
- Contoh visualisasi yang dapat dibuat dengan menggunakan matplotlib diantaranya adalah:
 1. Line chart
 2. Bar chart
 3. Pie chart
 4. Box plot chart
 5. Violin chart
 6. Errorbar chart
 7. Scatter chart
- Jenis-jenis *chart* lainnya juga dapat dibuat melalui library ini.

Tes 1

- Karyawan A mendapatkan tugas dari karyawan B untuk membuat summary dari hasil dataset penjualan di e-commerce ABC, dimana order dataset tersebut disimpan di CSV file. Namun karyawan A memiliki kesulitan dalam melakukan proses loading dataset tersebut ke dalam Python. Library dari Python manakah yang dapat direkomendasikan kepada karyawan A?

Tes 2

- Setelah melakukan proses manipulasi data, untuk menyajikan data tersebut, karyawan A butuh membuat beberapa chart terkait distribusi data. Library manakah yang paling cocok digunakan oleh karyawan A?

Memanggil library di Python

- Sebelum dapat digunakan, library tersebut harus terlebih dahulu dipanggil ke dalam lingkungan Python.
- Command untuk memanggil library di Python menggunakan syntax (menggunakan huruf kecil):

```
import library_name as alias
```

- **Alias** berfungsi sebagai pengganti nama library, sehingga menghemat komputasi saat function dari library tersebut dipanggil.
- Contoh: mengimport library numpy & pandas:

```
import numpy as np  
import pandas as pd
```

- Secara otomatis library numpy dan pandas akan diimport.

Exploratory Data Analysis dengan Pandas - Bagian 1

Membaca file dari Excel atau CSV sebagai data frame

- Salah satu fungsi Pandas yaitu melakukan load data dari CSV atau Excel file.
- Syntax yang digunakan untuk melakukan operasi tersebut, yaitu:

```
# Membaca file CSV  
[nama_variabel] = pd.read_csv("nama_file.csv")  
  
# Membaca file Excel  
[nama_variabel] = pd.read_excel("nama_file.xlsx")
```

- Nama variabel (**[nama_variabel]**) dari contoh diatas menunjukkan nama variabel dari dataframe untuk menampung data dari datasets tersebut!

Tugas Praktek 1

- Cobalah untuk mengimport dataset marketplace ABC dari **order.csv** dan disimpan ke dalam dataframe bernama **order_df**.
- Dari informasi tersebut dapat kita ketahui nama variabel dari dataframe tersebut adalah **order_df**.
- Dan nama file nya adalah **order.csv**.
- Lengkapi syntax berikut ini:

```
import --- as ---  
--- = pd.---(---)
```

Inspeksi struktur data frame

- Setelah melakukan proses loading dataframe ke dalam Python, hal selanjutnya sebelum memulai analisis adalah mengerti struktur dataset tersebut.
- Sehingga langkah selanjutnya dari pre-analisis biasanya dilakukan untuk:
 1. melihat struktur kolom dan baris data frame (`.shape`)
 2. melihat preview data dari dataframe tersebut (`.head()` dan `.tail()`)
 3. membuat summary data sederhana dari dataset (`.describe()`)

Inspeksi struktur data frame:

1. Melihat struktur kolom dan baris dari data frame

- Hal pertama dalam mengerti struktur dari dataframe adalah informasi mengenai berapa size dari dataframe yang akan digunakan termasuk berapa jumlah kolom dan jumlah baris data frame tersebut.
- Dalam kasus ini, kita dapat menggunakan fungsi `.shape` pada suatu dataframe.
- Syntax:

```
print([nama_dataframe].shape)
```

Tugas Praktek 2

- Pada **order_df** dataframe tuliskan syntax Python untuk melihat struktur dari **order_df** dengan menggunakan fungsi **shape**!

- Lengkapi syntax berikut ini:

```
import --- as ---  
order_df = pd.---(---)  
print(order_df.---)
```

- Output:

```
(49999, 12)
```

- Ketika diprint, tampilan akan menjelaskan bahwa dataframe **order_df** memiliki 12 kolom & 49999 baris.

Inspeksi struktur data frame:

2. Melihat preview data dari data frame

- Selanjutnya, untuk mendapatkan gambaran dari konten dataframe tersebut. Kita dapat menggunakan function `.head()` dan `.tail()`, dengan syntax:

```
# Menampilkan konten teratas dari [nama_dataframe]
# untuk sejumlah bilangan bulat [jumlah_data]
print([nama_dataframe].head([jumlah_data]))

# Menampilkan konten terbawah dari [nama_dataframe]
# untuk sejumlah bilangan bulat [jumlah_data]
print([nama_dataframe].tail([jumlah_data]))
```

- Jika `[jumlah_data]` pada function `.head()` dan `.tail()` dikosongkan maka secara default akan di ditampilkan sebanyak 5 (lima) baris saja. Sehingga bisa ditulis sebagai berikut:

```
# DEFAULT: Menampilkan 5 (baris) konten teratas
# dari [nama_dataframe]
print([nama_dataframe].head())

# DEFAULT: Menampilkan 5 (baris) konten terbawah
# dari [nama_dataframe]
print([nama_dataframe].tail())
```

Tugas Praktek 3

- Cobalah untuk check bagaimana contoh data dari dataframe tersebut dengan fungsi **head** dengan **limit 10 baris!**
- Berikut syntaxnya:

```
import --- as --  
order_df = pd.---(---)  
print(order_df.---
```

Outputnya adalah tampilan 10 baris teratas yang memiliki 12 kolom mulai dari kolom `order_id`, ... (read: dan seterusnya) hingga kolom `product_weight_gram`.

	order_id	...	product_weight_gram
0	2e7a8482f6fb09756ca50c10d7bfc047	...	1800.0
1	2e7a8482f6fb09756ca50c10d7bfc047	...	1400.0
2	e5fa5a7210941f7d56d0208e4e071d35	...	700.0
3	3b697a20d9e427646d92567910af6d57	...	300.0
4	71303d7e93b399f5bcd537d124c0bcfa	...	500.0
5	be5bc2f0da14d8071e2d45451ad119d9	...	400.0
6	0a0837a5eee9e7a9ce2b1fa831944d27	...	3100.0
7	1ff217aa612f6cd7c4255c9bfe931c8b	...	200.0
8	22613579f7d11cc59c4347526fc3c79e	...	600.0
9	356b492aba2d1a7da886e54e0b6212b7	...	610.0

[10 rows x 12 columns]

Inspeksi struktur data frame:

3. Statistik Deskriptif dari Data Frame — Bagian 1

- Statistik deskriptif atau *summary* dalam Python — Pandas, dapat diperoleh dengan menggunakan fungsi `.describe()`, yaitu:

```
print([nama_dataframe].describe())
```

- Function describe dapat memberikan informasi mengenai nilai:
 - Rataan
 - Standar deviasi
 - IQR (interquartile range)

Inspeksi struktur data frame:

3. Statistik Deskriptif dari Data Frame — Bagian 1

- Ketentuan umum:
 - Secara umum function `describe()` akan secara otomatis mengabaikan kolom category dan hanya memberikan summary statistik untuk kolom berjenis numerik.
 - Kita perlu menambahkan argument bernama `include = "all"` untuk mendapatkan summary statistik atau statistik deskriptif dari kolom **numerik** dan karakter (**category**).
- yaitu:

```
print([nama_dataframe].describe(include="all"))
```

Inspeksi struktur data frame:

3. Statistik Deskriptif dari Data Frame — Bagian 1

Contoh penggunaan `describe()` di Pandas!

- Terdapat dataframe Pandas dengan nama `nilai_skor_df` dengan informasi seperti gambar dibawah:

	Age	Name	Score
1.	20	Lindsay	82
2.	18	Robert	75
3.	23	Ricko	78
4.	21	Nicky	64
5.	25	Daniella	98

Menghasilkan:

	Age	Score
count	5.000000	5.000000
mean	21.400000	79.400000
std	2.701851	12.361229
min	18.000000	64.000000
25%	20.000000	75.000000
50%	21.000000	78.000000
75%	23.000000	82.000000
max	25.000000	98.000000

- dengan menggunakan fungsi describe pada `nilai_skor_df`

```
print([nama_dataframe].describe())
```

Inspeksi struktur data frame:

3. Statistik Deskriptif dari Data Frame — Bagian 2

- Jika ingin mendapatkan summary statistik dari kolom yang tidak bernilai angka, maka dapat menambahkan command `include=["object"]` pada syntax `describe()`.

```
print(nilai_skor_df.describe(include=["object"]))
```

- Hasil `include=["object"]`:

	Name
count	5
unique	5
top	Daniella
freq	1

- Function `describe()` dengan `include="all"` akan memberikan summary statistic dari semua kolom. Contoh penggunaannya:

```
print(nilai_skor_df.describe(include=["all"]))
```

Hasil `include="all"`:

	Age	Name	Score
count	5.000000	5	5.000000
Unique	NaN	5	NaN
Top	NaN	Daniella	NaN
Freq	NaN	1	NaN
mean	21.400000	NaN	79.400000
std	2.701851	NaN	12.361229
min	18.000000	NaN	64.000000
25%	20.000000	NaN	75.000000
50%	21.000000	NaN	78.000000
75%	23.000000	NaN	82.000000
max	25.000000	NaN	98.000000

Inspeksi struktur data frame:

3. Statistik Deskriptif dari Data Frame — Bagian 3

- Selanjutnya, untuk mencari rata-rata dari suatu data dari dataframe. Gunakan syntax **mean**, **median**, dan **mode** dari Pandas.

```
print([nama_dataframe].loc[:, "nama_kolom"].mean())  
print([nama_dataframe].loc[:, "nama_kolom"].median())  
print([nama_dataframe].loc[:, "nama_kolom"].mode())
```

- Contoh penggunaan:

```
print(nilai_skor_df.loc[:, "Age"].mean())  
print(nilai_skor_df.loc[:, "Score"].median())
```

- Memberikan hasil:

```
21.4    # Mean  
78      # Median
```

Tugas Praktek 4

1. Mengevaluasi performa cabang A dengan tambahan quick summary dari segi kuantitas, harga, freight value, dan weight yang dibeli Konsumen. Manfaatkan dataframe **order_df**.
 2. Mencari median dari total pembelian konsumen per transaksi kolom price
- **Solusi nomor 1:** Untuk bisa memperoleh data *quick summary* tersebut, gunakan fungsi `describe()` terhadap `order_df` untuk melihat statistik deskriptif dari kolom `quantity`, `price`, `freight_value`, dan `product_weight_gram` yang masing-masing mewakili kuantitas, harga, *freight value*, serta *weight* dari produk yang dibeli konsumen.

```
import --- as ---
order_df = pd.---("/path/order.csv")
# Quick summary dari segi kuantitas, harga, freight value, dan weight
print(order_df.---)
```

- Ketika di run akan menampilkan:

	quantity	price	freight_value	product_weight_gram
count	49999.000000	4.999900e+04	49999.000000	49980.000000
mean	1.197484	2.607784e+06	104521.390428	2201.830892
std	0.722262	1.388312e+06	55179.844962	3929.896875
min	1.000000	2.000000e+05	9000.000000	50.000000
25%	1.000000	1.410500e+06	57000.000000	300.000000
50%	1.000000	2.610000e+06	104000.000000	800.000000
75%	1.000000	3.810000e+06	152000.000000	1850.000000
max	21.000000	5.000000e+06	200000.000000	40425.000000

Tugas Praktek 4

- Hanya dengan menggunakan fungsi `describe()` tanpa parameter apapun, kita sudah langsung mendapatkan *summary* dari kolom data yang kita inginkan.
- Mengapa? Karena kolom-kolom tersebut bertipe numerik sehingga kita hanya perlu menampilkannya dengan `describe()` secara *default* saja.

Tugas Praktek 4

- No. 2: Mencari median dari total pembelian konsumen per transaksi kolom “price”.
- Solusi:
 - Pada bagian median kita harus mengetahui nama kolom yang ingin kita analisis, sesuai permintaan yaitu kolom total pembelian konsumen per transaksi, jika kita membuka dataframe dengan excel kita dapat melihat bahwa kolom tersebut bernama “price”.
 - Kemudian mengapa kita menggunakan syntax `describe()` pada kasus ini dan bukannya `.describe(include = “all”)`, dikarenakan sesuai permintaan quick summary yang diinginkan hanya numerik saja.

Tugas Praktek 4

- Solusi No. 2: Menentukan median dari total pembelian Konsumen (kolom *price*) dengan menggunakan fungsi `median()`. Dataframe yang akan kita gunakan dalam praktek ini adalah `order_df`.

```
# Median dari total pembelian konsumen per transaksi kolom price
print(order_df.---[:, ---].---)
```

- Output:

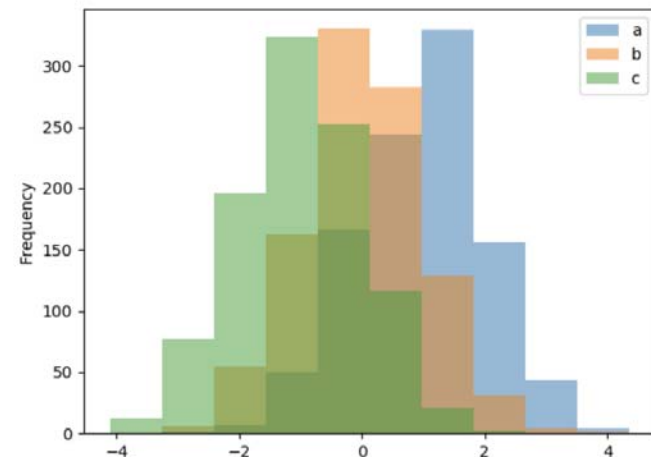
	quantity	price	freight_value	product_weight_gram
count	49999.000000	4.999900e+04	49999.000000	49980.000000
mean	1.197484	2.607784e+06	104521.390428	2201.830892
std	0.722262	1.388312e+06	55179.844962	3929.896875
min	1.000000	2.000000e+05	9000.000000	50.000000
25%	1.000000	1.410500e+06	57000.000000	300.000000
50%	1.000000	2.610000e+06	104000.000000	800.000000
75%	1.000000	3.810000e+06	152000.000000	1850.000000
max	21.000000	5.000000e+06	200000.000000	40425.000000
		2610000.0		

- Pada output diatas bahwa angka 2610000 merupakan nilai median dari total pembelian konsumen per transaksi pada kolom *price*.

Exploratory Data Analysis dengan Pandas - Bagian 2

Mengenai dan Membuat Distribusi Data dengan Histogram

- Histogram merupakan salah satu cara untuk mengidentifikasi sebaran distribusi dari data.
- Histogram adalah grafik yang berisi ringkasan dari sebaran (dispersi atau variasi) suatu data.
- Pada histogram, tidak ada jarak antar bar dari grafik.
- Hal ini dikarenakan bahwa titik data kelas bisa muncul dimana saja di daerah cakupan grafik.
- Sedangkan ketinggian bar sesuai dengan frekuensi atau frekuensi relatif jumlah data di kelas.
- Semakin tinggi bar, semakin tinggi frekuensi data.
- Semakin rendah bar, semakin rendah frekuensi data.



Mengenal dan Membuat Distribusi Data dengan Histogram

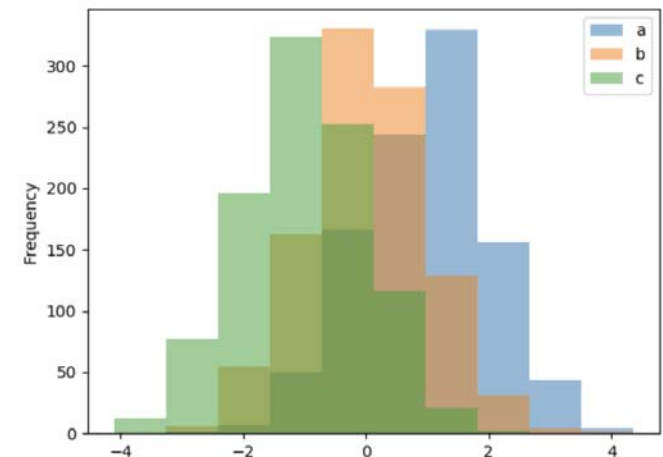
- Syntax umum:

```
nama_dataframe[["nama_kolom"]].plot.hist(bins=jumlah_bins)
```

- Beberapa atribut penting dalam histogram pandas:

- *bins* = jumlah_bins dalam histogram yang akan digunakan. Jika tidak didefinisikan jumlah_bins, maka function akan secara default menentukan jumlah_bins sebanyak 10.
- *by* = Kolom di DataFrame untuk di group by. (valuenya berupa nama column di dataframe tersebut).
- *alpha* = Menentukan opacity dari plot di histogram. (value berupa range 0.0–1.0, dimana semakin kecil akan semakin kecil opacity nya)
- *figsize* = digunakan untuk menentukan ukuran dari plot histogram. Contoh: figsize=(10,12)

Contoh Histogram



Tugas Praktek 5

- Membuat suatu distribusi harga dari pembelian produk di cabang A.
- Dataframe yang akan kita gunakan sama seperti data sebelumnya, yaitu **order_df**.
- Supaya lebih mudah dipahami, kita akan membuatnya dalam bentuk histogram dari kolom **price**.
- Solusi:
 - Pertama-tama, import *library* yang dibutuhkan, yaitu *Pandas* dan *Matplotlib*.
 - Selanjutnya, lakukan *loading* terhadap *dataset* **order.csv** kemudian membuat histogram dengan fungsi **hist()** untuk membuat distribusi dari kolom **price**.
 - Supaya histogram ini dapat ditampilkan, jangan lupa untuk menambahkan perintah **plt.show()**,
 - Buat histogram pada kolom **price** dengan **bins=10**.

- Lengkapi:

```
import [redacted] as [redacted]
import [redacted].[redacted] as [redacted]

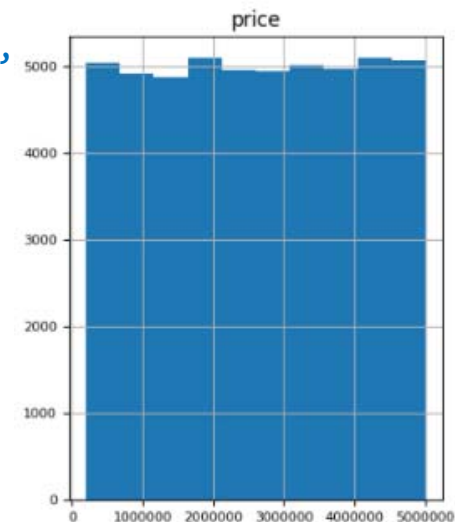
order_df = pd.[redacted]("order.csv")

# Membuat plot histogram menggunakan kolom price

order_df[[[redacted]]].[redacted]([redacted], [redacted],
xlabelsize=8, ylabelsize=8)

plt.[redacted]() # Untuk menampilkan histogram yang sudah dibuat
```

- Terlihat setelah di run akan menampilkan histogram pada kolom **price**.
- terlihat bahwa banyaknya harga dari kolom **price** untuk tiap-tiap datanya adalah hampir sama.



Standar Deviasi dan Varians pada Pandas

- Varians dan standar deviasi juga merupakan suatu ukuran dispersi atau variasi.
- Standar deviasi merupakan ukuran dispersi yang paling banyak dipakai.
- Hal ini mungkin karena standar deviasi mempunyai satuan ukuran yang sama dengan satuan ukuran data asalnya.
- Sedangkan varians memiliki satuan kuadrat dari data asalnya (misalnya cm²).
- Syntax dari standar deviasi dan varians pada Pandas:

```
print([nama_dataframe].loc[:, "nama_kolom"].std())  
print([nama_dataframe].loc[:, "nama_kolom"].var())
```

- Contoh penggunaan pada dataframe **nilai_skor_df**:

```
print(nilai_skor_df.loc[:, "Age"].std())  
print(nilai_skor_df.loc[:, "Score"].var())
```

- Hasil:

```
2.701851217  
152.8
```

Tugas Praktek 6

- Menampilkan data persebaran **product_weight_gram** pada data penjualan cabang A yang kita peroleh dari **order.csv**.
- Tampilkan *standar deviasi* dan *varians* dari kolom **product_weight_gram** pada dataframe **order_df** tadi untuk menganalisis lebar persebaran distribusi tersebut.

• Kode:

```
import pandas as pd
order_df = pd.read_csv("/path/order.csv")

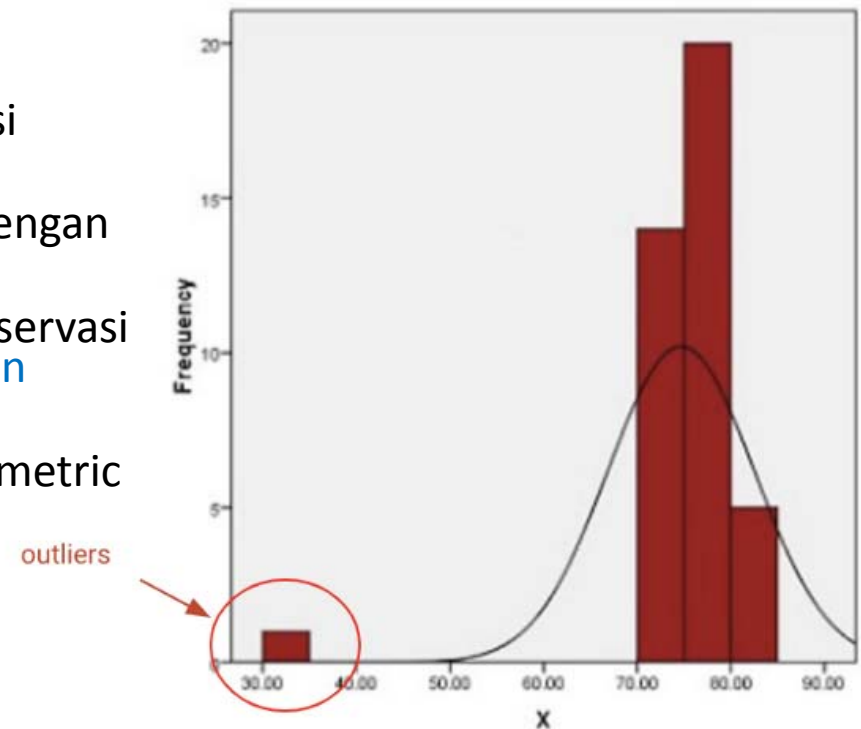
# Standar variasi kolom product_weight_gram
print(order_df['product_weight_gram'].std())
# Varians kolom product_weight_gram
print(order_df['product_weight_gram'].var())
```

- Setelah di run akan menghasilkan standar variasi & varians nya, yaitu:

```
3929.896875372737
15444089.451064402
```

Menemukan Outliers Menggunakan Pandas

- Sebelum menuju ke step by step dalam menemukan **outliers**, kita harus memahami definisi dari **outliers**.
- **Outliers** merupakan data observasi yang muncul dengan nilai-nilai ekstrim.
- Yang dimaksud dengan nilai-nilai ekstrim dalam observasi adalah **nilai yang jauh atau beda sama sekali dengan sebagian besar nilai lain dalam kelompoknya**.
- Pada umumnya, outliers dapat ditentukan dengan metric IQR (*interquartile range*).
- Rumus dasar dari **IQR: $Q3 - Q1$** .
- Data suatu observasi dapat dikatakan **outliers jika memenuhi kedua syarat** dibawah ini:
 - **$data < Q1 - 1.5 * IQR$**
 - **$data > Q3 + 1.5 * IQR$**



```
Q1 = nama_dataframe.quantile(0.25)
Q3 = nama_dataframe.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

Menemukan Outliers Menggunakan Pandas

- Contoh case: mengidentifikasi IQR dari dataframe `nilai_skor_df`

```
Q1 = nilai_skor_df[["Score"]].quantile(0.25)
Q3 = nilai_skor_df[["Score"]].quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

- Maka query tersebut akan menghasilkan:

```
Age      3.0
Score    7.0
dtype: float64
```

Hasil

	Age	Name	Score
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	True
4	False	False	False

- Setelah mendapatkan skor IQR, maka Outliers dapat ditentukan.
- Kode di bawah ini akan memberikan output dengan beberapa nilai True atau False. Titik data di mana terdapat **False** yang berarti nilai-nilai ini valid sedangkan **True** menunjukkan adanya Outliers.

```
print((nilai_skor_df < (Q1 - 1.5*IQR)) | (nilai_skor_df > (Q3 + 1.5*IQR)))
```

Tugas Praktek 7

- Setelah menentukan standar deviasi & varians pada kolom **product_weight_gram**, selanjutnya dicoba menentukan batas IQR untuk bisa menentukan outliers bagi kolom **product_weight_gram**. Manfaatkan dataframe order_df.

```
import --- as ---  
order_df = pd.---("/path/order.csv")
```

- Dilanjutkan dengan menghitung Q1, Q3, dan IQR:

```
# Hitung quartile 1  
Q1 = order_df[[  
# Hitung quartile 3  
Q3 = order_df[[  
# Hitung inter quartile range dan cetak ke console  
IQR =   
print(  

```

- Setelah di run maka yang ditampilkan adalah **IQR** yang merupakan hasil dari **Q3-Q1**:

```
product_weight_gram 1550.0  
dtype: float64
```

Rename Kolom Data Frame

- Mengganti nama kolom dataframe menggunakan library Pandas.
- Mengganti nama kolom pada Pandas dapat dilakukan dengan 2 cara:
 1. Menggunakan nama kolom.
 2. Menggunakan indeks kolom.

1. Rename menggunakan nama kolom. Syntax:

```
nama_dataframe.rename(columns={"column_name_before": "column_name_after"}, inplace=True)
```

- Contoh penggunaan: `nilai_skor_df.rename(columns={"Age": "Umur"}, inplace=True)`

2. Rename menggunakan indeks kolom. Syntax:

```
nama_dataframe.columns.values[no_of_column] = "column_name_after"
```

- Contoh penggunaan: `nilai_skor_df.columns.values[0] = "Umur"`

Tugas Praktek 8

- Cobalah untuk mengubah kolom `freight_value` menjadi `shipping_cost` dalam data frame `order_df`, dengan menggunakan fungsi `rename()`.
- Syntax:

```
import --- as ---  
order_df = --.---("/path/order.csv")
```

```
# Ganti nama kolom freight_value menjadi shipping_cost  
order_df.rename(columns={'freight_value': 'shipping_cost'})  
print(order_df)
```

- Ketika dirun, maka kolom `freight_value` berganti nama menjadi kolom `shipping_cost`. Berikut tampilannya:

	seller	id shipping_cost \
0	1554a68530182680ad5c8b042c3ab563	28000
1	1554a68530182680ad5c8b042c3ab563	45000
2	a425f92c199eb576938df686728acd20	174000
3	522620dcb18a6b31cd7bdf73665113a9	154000
4	25e6ffe976bd75618accfe16cefcdbd0d	147000
...	...	...
49994	7ddcbb64b5bc1ef36ca8c151f6ec77df	172000
49995	4d6d651bd7684af3ffabd5f08d12e5a	130000
49996	955fee9216a65b617aa5c0531780ce60	14000
49997	955fee9216a65b617aa5c0531780ce60	108000
49998	1900267e848cee8a8fa32d80c1a5f5a8	189000

.groupby menggunakan Pandas

- Kegunaan **.groupby** adalah mencari summary dari data frame dengan menggunakan **aggregate** dari kolom tertentu.

- **Contoh penggunaan:**

Diberikan dataframe bernama **df** seperti pada gambar dibawah!

Penggunaan groupby

	Name	Exam	Subject	Score
0	Alisa	Semester 1	Mathematics	62
1	Bobby	Semester 1	Mathematics	47
2	Cathrine	Semester 1	Mathematics	55
3	Alisa	Semester 1	Science	74
4	Bobby	Semester 1	Science	31
5	Cathrine	Semester 1	Science	77
6	Alisa	Semester 2	Mathematics	85
7	Bobby	Semester 2	Mathematics	63
8	Cathrine	Semester 2	Mathematics	42
9	Alisa	Semester 2	Science	67
10	Bobby	Semester 2	Science	89
11	Cathrine	Semester 2	Science	81

Syntax penggunaan groupby:

```
df["Score"].groupby([df["Name"]]).mean()
```

Hasil:

```
Name
Alisa      72.00
Bobby      57.50
Cathrine    63.75
Name: Score, dtype: float64
```

Penjelasan:

Kode menggunakan kolom 'Name' sebagai *aggregate* dan kemudian menghitung mean dari kolom 'Score' pada tiap-tiap aggregate tersebut.

Contoh lainnya:

```
df["Score"].groupby([df["Name"], df["Exam"]]).sum()
```

Hasil:

```
Name      Exam
Alisa      Semester 1    136
           Semester 2    152
Bobby      Semester 1     78
           Semester 2    152
Cathrine    Semester 1    132
           Semester 2    123
Name: Score, dtype: int64
```

Penjelasan:

Kode menggunakan kolom 'Name' dan 'Exam' sebagai *aggregate* dan kemudian menghitung mean dari kolom 'Score' pada tiap-tiap aggregate tersebut.

Tugas Praktek 9

- Setelah memahami penggunaan fungsi `groupby()`. Carilah **rata rata** dari kolom '`price`' dengan mengelompokkannya per `payment_type` dari dataset `order_df`!

- Syntax:

```
import --- as --
order_df = ---.---("/path/order.csv")
```

```
# Hitung rata rata dari price per payment_type
rata_rata = order_df[        ].        (order_df[        ]).        ()
print(        )
```

- Hasil:

```
payment_type
credit card      2.600706e+06
debit card      2.611974e+06
e-wallet        2.598562e+06
virtual account 2.619786e+06
Name: price, dtype: float64
```
- Terlihat rata-rata transaksi dari setiap tipe pembayaran.
- Terdapat 4 jenis tipe pembayaran dalam data tersebut, yaitu menggunakan kartu kredit (**credit_card**), kartu debit (**debit_card**), e-wallet (**e-wallet**), dan akun virtual (**virtual account**).
- Rata-rata nilai kolom price untuk masing-masing tipe pembayaran tersebut memiliki jumlah yang hampir sama.

Sorting Menggunakan Pandas

- **Sorting** adalah sebuah metode mengurutkan data berdasarkan syarat kolom tertentu, dan biasanya digunakan untuk melihat nilai maksimum dan minimum dari dataset.
- Library Pandas menyediakan fungsi sorting sebagai fundamental dari exploratory data analysis.
- Syntax untuk operasi sorting pada Pandas:

```
nama_dataframe.sort_values(by="nama_kolom")
```

- Contoh: Sorting terhadap dataset `nilai_skor_df` berdasarkan kolom 'age'!

```
nilai_skor_df.sort_values(by="Age")
```

- Menghasilkan sorting ascending:

	Age	Name	Score
1	18	Robert	75
0	20	Lindsay	82
3	21	Nicky	64
2	23	Ricko	78
4	25	Daniella	98

Sorting Menggunakan Pandas

- Fungsi *sorting* secara default mengurutkan secara ascending (dimulai dari nilai terkecil), untuk dapat mengurutkan secara descending (nilai terbesar lebih dahulu), dapat menggunakan properti tambahan:

```
nama_dataframe.sort_values(by="nama_kolom", ascending=False)
```

- Contoh:

```
nilai_skor_df.sort_values(by="Age", ascending=False)
```

- Sorting terhadap dataset nilai_skor_df berdasarkan age dimulai dari umur tertua!

```
nilai_skor_df.sort_values(by="Age", ascending=False)
```

- Menghasilkan:

	Age	Name	Score
4	25	Daniella	98
2	23	Ricko	78
3	21	Nicky	64
0	20	Lindsay	82
1	18	Robert	75

Sorting Menggunakan Pandas

- Fungsi sorting di Pandas juga dapat dilakukan menggunakan **lebih dari satu kolom sebagai syarat**.
- Contoh nya pada skenario dibawah, akan mencoba mengaplikasikan fungsi Sorting menggunakan kolom 'Age' dan 'Score' sekaligus:

```
# Syntax  
nama_dataframe.sort_values(by=["nama_kolom_1", "nama_kolom_2"], ascending=[False, True])  
  
# Contoh penggunaan pada dataframe nilai_skor_df  
nilai_skor_df.sort_values(by=["Age", "Score"], ascending=[True, False])
```

Tugas Praktek 10

- Cari berapa harga maksimum pembelian customer pada kolom 'price' di dataset `order_df`.

- Syntax:

```
import pandas as pd
order_df = pd.read_csv("/path/order.csv")

# Hitung harga maksimum pembelian customer
sort_harga = order_df.sort_values("price", ascending=False)
print(sort_harga["price"].iloc[0])
```

- Setelah dijalankan, maka hasilnya:

	product_id	price \
37085	35afc973633aaeb6b877ff57b2793310	5000000
41958	7c1bd920dbdf22470b68bde975dd3ccf	5000000
3976	777d2e438a1b645f3aec9bd57e92672c	5000000
21072	f8cfb63e323be2e1c4172f255d61843d	5000000
47074	2ea92fab7565c4fe9f91a5e4e1756258	5000000
33786	f93213a23c50edc16c27b96333f734dc	200000
42166	1166bc797ddf5fb009c376d133f61204	200000
31745	eb38a7604070a2b8465101ed53cba72b	200000
42452	db5efde3ad0cc579b130d71c4b2db522	200000
11939	06c6e01186af8b98ee1fc9e01f9471e9	200000

- Terlihat setelah di run, pembelian maksimum customer adalah 5000000.

Tugas Praktek 11

Analisis data penjualan dari dataset order.csv dengan detail laporan sebagai berikut. Lakukan dengan cara:

1. Mengitung median `price` yang dibayar customer dari masing-masing metode pembayaran.
2. Tentukan metode pembayaran yang memiliki basket size (rataan median price) terbesar.
3. Ubah `freight_value` menjadi `shipping_cost` dan cari `shipping_cost` termahal dari data penjualan tersebut menggunakan sort.
4. Untuk `product_category_name`, berapa rata-rata weight produk tersebut dan standar deviasi mana yang terkecil dari weight tersebut,
5. Buat histogram quantity penjualan dari dataset tersebut untuk melihat persebaran quantity penjualan tersebut dengan bins = 5 dan figsize= (4,5)

Solusi 1: Median price yang dibayar customer dari masing-masing metode pembayaran.
Solusi 2: Metode pembayaran yang memiliki basket size (rata-rata median price) terbesar.

- Soal pertama: caranya adalah menggunakan kolom *payment_type* sebagai *aggregate* dan kemudian menghitung median dari kolom *price* pada tiap-tiap *aggregate* tersebut. Dengan kata lain, kita akan menghitung *median* harga (pada kolom '*price*') dengan mengelompokkannya menurut masing-masing metode pembayarannya (pada kolom '*payment_type*')
- Dengan contoh diatas kita dapat menerapkan fungsi *.groupby* pada kasus ini.

- Hasil:

```
import --- as ---
import ---.pyplot as ---
order_df = ---.---("/path/order.csv")
# Median price yang dibayar customer dari masing-masing metode pembayaran
median_price = order_df[---].---(order_df[---]).---()
print(---)
```

```
payment_type
credit card    2587500
debit card     2621500
e-wallet       2589000
virtual account 2643000
Name: price, dtype: int64
```

- Terlihat median dari price berdasarkan tipe pembayaran nya masing-masing.
- Soal kedua: setelah dijalankan terlihat bahwa tipe pembayaran **virtual account** yang memiliki rata-rata median price yang paling besar, yaitu sebesar 2643000.

- Selanjutnya, ganti nama kolom `freight_value` menjadi `shipping_cost`, serta cari nilai `shipping_cost` termahal dengan menggunakan fungsi `sort_values()`.

```
import --- as ---
import ---.pyplot as ---
order_df = pd.read_csv("/path/order.csv")
# Ubah freight_value menjadi shipping_cost dan cari shipping_cost
# termahal dari data penjualan tersebut menggunakan sort
order_df.---(columns={---: ---}, --- = ---)
sort_value = order_df.---(--- = ---, ascending=0)
print(---)
```

- Setelah di run, maka kolom tersebut berubah menjadi:

	seller_id	shipping_cost \
23829	cc419e0650a3c5ba77189a1882b7556a	200000
12011	2e0dba2da448400b1c11d7b4b22f32a4	200000
10984	d12c926d74ceff0a90a21184466ce161	200000
29722	259f7b5e6e482c230e5bfaa670b6bb8f	200000
15259	efcd8d2104f1a05d028af7bad20d974b	200000
...	...	...
2891	82e0a475a88cc9595229d8029273f045	9000
36815	d12c926d74ceff0a90a21184466ce161	9000
8175	6560211a19b47992c3666cc44a7e94c0	9000
4034	004c9cd9d87a3c30c522c48c4fc07416	9000
32994	cb3dd9ce66268c7a3ca7241ac70ab58c	9000

Solusi 3. Ubah `freight_value` menjadi `shipping_cost` dan cari `shipping_cost` termahal dari data penjualan tersebut menggunakan `sort`.

- Kemudian kita tentukan shipping cost yang termahal. Berdasarkan data tersebut shipping cost atau ongkos kirim yang paling mahal sebesar 200000.

Solusi 4: Untuk `product_category_name`, berapa rata-rata weight produk tersebut dan standar deviasi mana yang terkecil dari weight tersebut.

- Tugas No. 4, mencari berat rata-rata `mean()` dari setiap kategori produk serta mencari standar deviasi `std()` dari berat setiap kategori produk. Lalu mengembalikan kedua nilai tersebut masing-masing pada variabel `mean_value` dan `std_value`.

- Hasil:

```
product_category_name
automotive      2129.559039
beauty          2120.088396
electronics     2144.751191
fashion         2246.516901
gadget          2229.632942
sports          2251.879909
toys            2203.863643
utilities       2291.029868
Name: product_weight_gram, dtype: float64
```

```
product_category_name
automotive      3799.884166
beauty          3837.474929
electronics     3839.271768
fashion         3858.885213
gadget          3955.792741
sports          4032.850044
toys            3956.814749
utilities       4150.509909
Name: product_weight_gram, dtype: float64
```

```
# Untuk product_category_name, berapa rata-rata weight produk tersebut
# dan standar deviasi mana yang terkecil dari weight tersebut,
mean_value = order_df[ ].mean(order_df[ ]).mean()
print( )
std_value = order_df[ ].std(order_df[ ]).std()
print( )
```

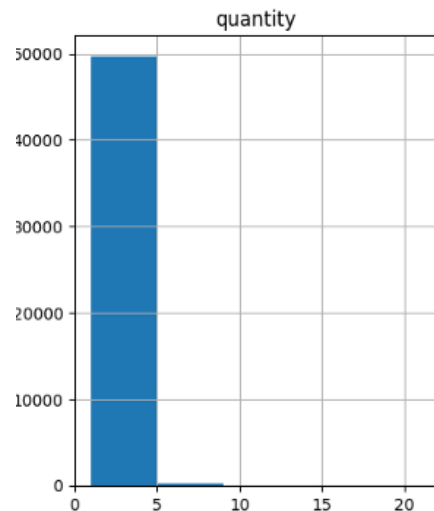
- Terlihat dari data tersebut, produk **"beauty"** memiliki berat rata-rata paling kecil, yaitu sebesar **2120.088396 gram** sedangkan produk **"automotive"** memiliki standar deviasi paling kecil, yaitu sebesar **3799.884166 gram**.⁵³

Solusi 5: Buat histogram quantity penjualan dari dataset tersebut untuk melihat persebaran **quantity** penjualan tersebut dengan bins = 5 dan figsize= (4,5).

- Syntax:

```
import pandas as pd
import matplotlib.pyplot as plt
# Buat histogram quantity penjualan dari dataset tersebut untuk melihat persebaran quantity
# penjualan tersebut dengan bins = 5 dan figsize= (4,5)
order_df[['quantity']].hist(bins=5, figsize=(4,5))
plt.show()
```

- Hasil:



Referensi

- Exploratory Data Analysis with Python for Beginner by DQLab.
- <https://academy.dqlab.id/main/package/practice/163?pf=0>