🐼 Pandas-part-04

a. Filtering Data 🔍

a.1 `.loc` – Label-based 🏷️

- **Sintaks dasar:** `dataOrang.loc[kondisi_boolean, kolom_list]`
- **Penjelasan:** Seleksi baris berdasarkan kondisi boolean, sekaligus memilih kolom dengan nama label.
- **Contoh Penggunaan:**

```
# Ambil semua kolom untuk baris Berat Badan > 80 kg
dataOrang.loc[dataOrang['berat_badan'] > 80]

# Ambil kolom nama & Berat Badan untuk baris Berat Badan > 80 kg
dataOrang.loc[
    dataOrang['berat_badan'] > 80,
    ['nama', 'berat_badan']
]
```

- **Kapan Gunakan `.loc`?**
  - Saat butuh pilih kolom spesifik setelah filter.
  - DataFrame pakai index non-angka (misal tanggal, ID).

---

a.2 Boolean Indexing tanpa `.loc` 📐

- **Sintaks dasar:** `df[kondisi_boolean][kolom_list]`
- **Penjelasan:** Terapkan mask boolean pada DataFrame, kemudian pilih kolom lewat indexing biasa.
- **Contoh Penggunaan:**

```
# nama, Berat Badan, tempat_lahir untuk Berat Badan > 80 kg
dataOrang[
    dataOrang['berat_badan'] > 80
][['nama', 'berat_badan', 'tempat_lahir']]

# Berat Badan > 80 dan tempat_lahir == "Salatiga"
dataOrang[
    (dataOrang['berat_badan'] > 80) &
```

```
    (dataOrang['tempat_lahir'] == "Salatiga")
]
```

- **Kapan Gunakan?**
  - Filter sederhana tanpa banyak slicing berdasarkan label.
  - Kode ringkas untuk satu operasi filter + kolom.

---

a.3 String Filtering dengan `.str` – Contains & Startswith ✂️

- **Sintaks dasar:**
  - `df[df['kolom'].str.contains(pola, case=Bool, regex=Bool)]`
  - `df[df['kolom'].str.startswith(awalan)]`
- **Penjelasan:** Cari baris di mana teks dalam kolom memenuhi pola (regex) atau literal, serta mendukung case sensitivity.
- **Contoh Penggunaan:**

```
# Mengandung "Gita" (case-sensitive)
dataOrang[dataOrang['nama'].str.contains("Gita")]

# Mengandung "Gita" tanpa peduli huruf besar/kecil
dataOrang[dataOrang['nama'].str.contains("Gita", case=False)]

# Gita atau Agus (regex)
dataOrang[dataOrang['nama'].str.contains("Gita|agus", case=False)]

# nama diawali dengan "Gita"
dataOrang[dataOrang['nama'].str.startswith("Gita")]
```

- **Kapan Gunakan?**
  - Mencari substring dalam kolom teks.
  - Filter berdasarkan pola atau awalan karakter.

---

a.4 Membership dengan `.isin` ✅

- **Sintaks dasar:** `df[df['kolom'].isin(list_of_values)]`
- **Penjelasan:** Filter baris berdasarkan apakah nilai kolom termasuk salah satu di daftar tertentu.
- **Contoh Penggunaan:**

```
# Lahir di Salatiga
dataOrang[dataOrang['tempat_lahir'].isin(["Salatiga"])]

# Lahir di Salatiga & nama mengandung "Gita"
dataOrang[
    dataOrang['tempat_lahir'].isin(["Salatiga"]) &
    dataOrang['nama'].str.contains("Gita")
]
```

- **Kapan Gunakan?**
  - Seleksi berdasarkan beberapa kategori atau grup nilai.
  - Filter multi-value secara efisien.

---

a.5 SQL-like Filtering dengan `.query` 

- **Sintaks dasar:** `df.query('expression')`
- **Penjelasan:** Menulis kondisi filter seperti SQL; kolom dipanggil langsung tanpa tanda kurung.
- **Contoh Penggunaan:**

```
# nama persis "Gita Maulani"
dataOrang.query('nama == "Gita Hadi"')

# nama "Gita Maulani" dan lahir di Salatiga
dataOrang.query(
    'nama == "Gita Maulani" and `tempat_lahir` == "Salatiga"'
)
```

- **Kapan Gunakan?**
  - Sintaks lebih bersih untuk kondisi kompleks.
  - Ketika ingin menulis logika filter seperti di SQL.