REFERENSI CEPAT PANDAS

DataFrame, seleksi, agregasi, penggabungan, dan lainnya

DataFrame

Membuat DataFrame

```
import pandas as pd
df = pd.DataFrame({
    "name": ["Alice", "Bob", "Carol"],
    "age": [25, 30, 35],
    "score": [88, 92, 79]
})
```

Inspeksi

`df.head(n)`	n baris pertama (default 5)
`df.tail(n)`	n baris terakhir
`df.shape`	Tuple (baris, kolom)
`df.dtypes`	Tipe data setiap kolom
`df.info()`	Tipe kolom, jumlah non-null
`df.describe()`	Statistik untuk kolom numerik
`df.columns`	Nama kolom sebagai Index
`df.index`	Label baris

Membaca Data

Reader Umum

```
df = pd.read_csv("data.csv")
df = pd.read_excel("data.xlsx")
df = pd.read_json("data.json")
df = pd.read_sql(query, connection)
```

Menulis Data

```
df.to_csv("out.csv", index=False)
df.to_excel("out.xlsx", index=False)
df.to_json("out.json", orient="records")
```

Opsi Baca

`sep=";"`	Delimiter kustom
`header=None`	Tidak ada baris header di file
`usecols=[0,2]`	Hanya baca kolom tertentu
`nrows=100`	Baca 100 baris pertama
`na_values=["N/A"]`	Perlakukan sebagai NaN

Seleksi

Kolom

```
df["name"]              # kolom tunggal (Series)
df[["name", "age"]]     # beberapa kolom (DataFrame)
df.name                 # akses atribut (nama sederhana)
```

Baris dengan loc / iloc

```
df.loc[0]             # baris berdasarkan label
df.loc[0:2, "name"]   # baris 0-2, kolom "name"
df.iloc[0]            # baris berdasarkan posisi
df.iloc[0:2, 0:2]     # 2 baris dan 2 kolom pertama
```

loc vs iloc

`df.loc[row, col]`	Pilih berdasarkan **label** (akhir inklusif)
`df.iloc[row, col]`	Pilih berdasarkan **posisi** (akhir eksklusif)
`df.at[row, col]`	Akses scalar cepat berdasarkan label
`df.iat[row, col]`	Akses scalar cepat berdasarkan posisi

Filter

Filter Boolean

```
df[df["age"] > 25]
df[df["name"].str.contains("li")]
df[(df["age"] > 25) & (df["score"] > 80)]
df[df["name"].isin(["Alice", "Bob"])]
```

Penanganan Data Hilang

```
df.isna().sum()          # jumlah NaN per kolom
df.dropna()              # hapus baris dengan NaN
df.fillna(0)             # isi NaN dengan 0
df["col"].fillna(df["col"].mean())
```

Pengurutan

```
df.sort_values("age")                # ascending
df.sort_values("age", ascending=False)
df.sort_values(["age", "score"])     # multi kolom
```

Agregasi

Agregasi Umum

`df["col"].sum()`	Jumlah kolom
`df["col"].mean()`	Rata-rata
`df["col"].median()`	Median
`df["col"].std()`	Standar deviasi
`df["col"].min() / .max()`	Min / maks
`df["col"].count()`	Jumlah non-null
`df["col"].nunique()`	Jumlah nilai unik
`df["col"].value_counts()`	Frekuensi setiap nilai

Beberapa Agregasi

```
df.agg({"age": "mean", "score": ["min", "max"]})
df.describe()  # statistik ringkasan untuk semua numerik
```

GroupBy

Pengelompokan Dasar

```
df.groupby("dept")["salary"].mean()
df.groupby("dept").agg(
    avg_sal=("salary", "mean"),
    count=("salary", "count")
)
```

Beberapa Grup

```
df.groupby(["dept", "year"])["sales"].sum()
df.groupby("dept").size()  # baris per grup
```

Transform & Apply

```
df["z_score"] = df.groupby("dept")["salary"] \
    .transform(lambda x: (x - x.mean()) / x.std())
df.groupby("dept").apply(lambda g: g.nlargest(3, "salary"))
```

Penggabungan

Merge (Join Gaya SQL)

```
pd.merge(df1, df2, on="id")          # inner
pd.merge(df1, df2, on="id", how="left")
pd.merge(df1, df2, left_on="uid",
         right_on="user_id")
```

Tipe Join

`how="inner"`	Hanya baris yang cocok (default)
`how="left"`	Semua baris kiri, NaN jika tidak cocok
`how="right"`	Semua baris kanan
`how="outer"`	Semua baris dari kedua sisi

Concatenation

```
pd.concat([df1, df2])            # susun baris
pd.concat([df1, df2], axis=1)    # berdampingan
pd.concat([df1, df2], ignore_index=True)
```

Pivot Table

Pivot Table

```
df.pivot_table(
    values="sales", index="region",
    columns="quarter", aggfunc="sum"
)
```

Reshaping

```
df.melt(id_vars=["name"],
        value_vars=["q1", "q2"],
        var_name="quarter", value_name="sales")
```

Cross Tabulation

```
pd.crosstab(df["dept"], df["gender"])
pd.crosstab(df["dept"], df["gender"],
            normalize="index")  # persentase per baris
```

Time Series

Dasar DateTime

```
df["date"] = pd.to_datetime(df["date"])
df["year"] = df["date"].dt.year
df["month"] = df["date"].dt.month
df["weekday"] = df["date"].dt.day_name()
```

Rentang Tanggal & Resampling

```
pd.date_range("2025-01-01", periods=12, freq="ME")
df.set_index("date").resample("ME")["sales"].sum()
```

Atribut Accessor

`.dt.year / .dt.month / .dt.day`	Ekstrak komponen tanggal
`.dt.hour / .dt.minute`	Ekstrak komponen waktu
`.dt.day_name()`	Nama hari (Monday, dll.)
`.dt.days_in_month`	Jumlah hari dalam bulan tersebut

Pola Umum

Ganti Nama Kolom

```
df.rename(columns={"old": "new"})
df.columns = ["a", "b", "c"]  # ganti semua
```

Tambah / Modifikasi Kolom

```
df["total"] = df["q1"] + df["q2"]
df["grade"] = df["score"].apply(
    lambda x: "A" if x >= 90 else "B"
)
```

Hapus Kolom / Baris

```
df.drop(columns=["temp"])
df.drop_duplicates(subset=["name"])
df.reset_index(drop=True)
```

Operasi String

```
df["name"].str.lower()
df["name"].str.contains("ali", case=False)
df["name"].str.split(" ").str[0]  # nama depan
```