pandas 快速参考

DataFrame、数据选择、聚合、合并等核心操作

DataFrame

创建 DataFrame

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

数据概览

df.head(n)	前 n 行（默认 5 行）
df.tail(n)	后 n 行
df.shape	（行数, 列数）元组
df.dtypes	每列的数据类型
df.info()	列类型与非空计数
df.describe()	数值列统计摘要
df.columns	列名 Index
df.index	行标签

读取数据

常用读取函数

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

写出数据

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

读取选项

sep=";"	自定义分隔符
header=None	文件无表头行
usecols=[0,2]	只读指定列
nrows=100	只读前 100 行
na_values=["N/A"]	将该值视为 NaN

数据选取

列选取

```
df["name"]              # 单列（Series）
df[["name", "age"]]     # 多列（DataFrame）
df.name                 # 属性访问（简单列名）
```

用 loc / iloc 选行

```
df.loc[0]              # 按标签选行
df.loc[0:2, "name"]    # 0-2 行，name 列
df.iloc[0]             # 按位置选行
df.iloc[0:2, 0:2]     # 前 2 行，前 2 列
```

loc 与 iloc 对比

df.loc[row, col]	按**标签**选取（含右端点）
df.iloc[row, col]	按**位置**选取（不含右端点）
df.at[row, col]	按标签快速访问单个标量
df.iat[row, col]	按位置快速访问单个标量

数据过滤

布尔过滤

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

处理缺失值

```
df.isna().sum()          # 各列 NaN 计数
df.dropna()              # 删除含 NaN 的行
df.fillna(0)             # 用 0 填充 NaN
df["col"].fillna(df["col"].mean())
```

排序

```
df.sort_values("age")                # 升序
df.sort_values("age", ascending=False)
df.sort_values(["age", "score"])      # 多列排序
```

聚合

常用聚合函数

df["col"].sum()	列求和
df["col"].mean()	均值
df["col"].median()	中位数
df["col"].std()	标准差
df["col"].min() / .max()	最小值 / 最大值
df["col"].count()	非空值计数
df["col"].nunique()	唯一值数量
df["col"].value_counts()	各值出现频次

多指标聚合

```
df.agg({"age": "mean", "score": ["min", "max"]})
df.describe()  # 所有数值列的统计摘要
```

分组

基础分组

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

多键分组

```
df.groupby(["dept", "year"])["sales"].sum()
df.groupby("dept").size()  # 每组行数
```

transform 与 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"))
```

合并

merge（SQL 风格 Join）

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

连接类型

how="inner"	只保留两表匹配行（默认）
how="left"	保留左表全部行，右表无匹配则 NaN
how="right"	保留右表全部行
how="outer"	保留两表全部行

拼接

```
pd.concat([df1, df2])             # 按行堆叠
pd.concat([df1, df2], axis=1)     # 按列并排
pd.concat([df1, df2], ignore_index=True)
```

透视表

pivot_table

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

重塑数据

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

交叉表

```
pd.crosstab(df["dept"], df["gender"])
pd.crosstab(df["dept"], df["gender"],
            normalize="index")  # 行百分比
```

时间序列

日期时间基础

```
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()
```

日期范围与重采样

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

dt 访问器属性

.dt.year / .dt.month / .dt.day	提取日期组件
.dt.hour / .dt.minute	提取时间组件
.dt.day_name()	星期名称（如 Monday）
.dt.days_in_month	该月天数

常用模式

重命名列

```
df.rename(columns={"old": "new"})
df.columns = ["a", "b", "c"]  # 替换所有列名
```

新增/修改列

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

删除列/行

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

pandas 快速参考

字符串操作

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