📅 DAY 11 — Pandas Advanced

Goal: **Group, merge, and transform data**

1️⃣ `groupby()` (MOST IMPORTANT)

Why `groupby`?

Used to:

- Aggregate data
- Analyze categories
- Create features for ML

Simple Example

```
import pandasas pd

data = {
"department": ["IT","IT","HR","HR","Sales"],
"salary": [60000,65000,50000,52000,70000]
}

df = pd.DataFrame(data)
```

```
df.groupby("department")["salary"].mean()

# Output:
department
HR51000
IT62500
Sales70000
```

Multiple Aggregations

```
df.groupby("department")["salary"].agg(["mean","max","min"])
```

2️⃣ `merge()` (Like SQL JOIN)

Why merge?

Real data is split across files.

Example

```
employees = pd.DataFrame({
"emp_id": [1,2,3],
"name": ["Alice","Bob","Charlie"]
})

salaries = pd.DataFrame({
"emp_id": [1,2,3],
"salary": [60000,65000,70000]
})
```

```
merged = pd.merge(employees, salaries, on="emp_id")
print(merged)
```

Types of joins

```
pd.merge(a, b, how="inner")
pd.merge(a, b, how="left")
pd.merge(a, b, how="right")
pd.merge(a, b, how="outer")
```

3️⃣ `apply()` (Row-wise Logic)

Why apply?

When built-in functions are not enough.

Example

```
df = pd.DataFrame({
"score": [85,90,72]
})

df["grade"] = df["score"].apply(
lambda x:"Pass"if x >=80else"Fail"
)
```

4️⃣ `map()` vs `apply()` (Simple)

Method	Used for
`map()`	Single column mapping
`apply()`	Row or column logic

Example:

```
df["score"].map(lambda x: x *1.1)
```

5️⃣ `pivot_table()` (Summary Tables)

```
sales = pd.DataFrame({
"region": ["East","East","West","West"],
"product": ["A","B","A","B"],
"revenue": [100,150,200,250]
})
```

```
pd.pivot_table(
    sales,
    values="revenue",
    index="region",
    columns="product",
    aggfunc="sum"
)
```

6️⃣ ML Feature Engineering Example

```
df["salary_scaled"] = (
    df["salary"] - df["salary"].mean()
) / df["salary"].std()
```

This uses:

✔️ aggregation

✔️ column creation

✔️ ML preparation
