Data Wrangling

🔹 Reminder

Yesterday we cleaned the Titanic dataset (handled missing values, duplicates, datatypes).
👉 Today we'll **transform** it so it becomes usable for insights and analysis.

🔹 Notes on Data Wrangling

- **Definition** → Data wrangling is the process of **transforming raw data into a structured, usable format**.
- **Why important?**
    - Raw datasets are messy.
    - Analysis/ML requires tidy data (rows = observations, columns = features).
    - Enables pattern discovery and insights.

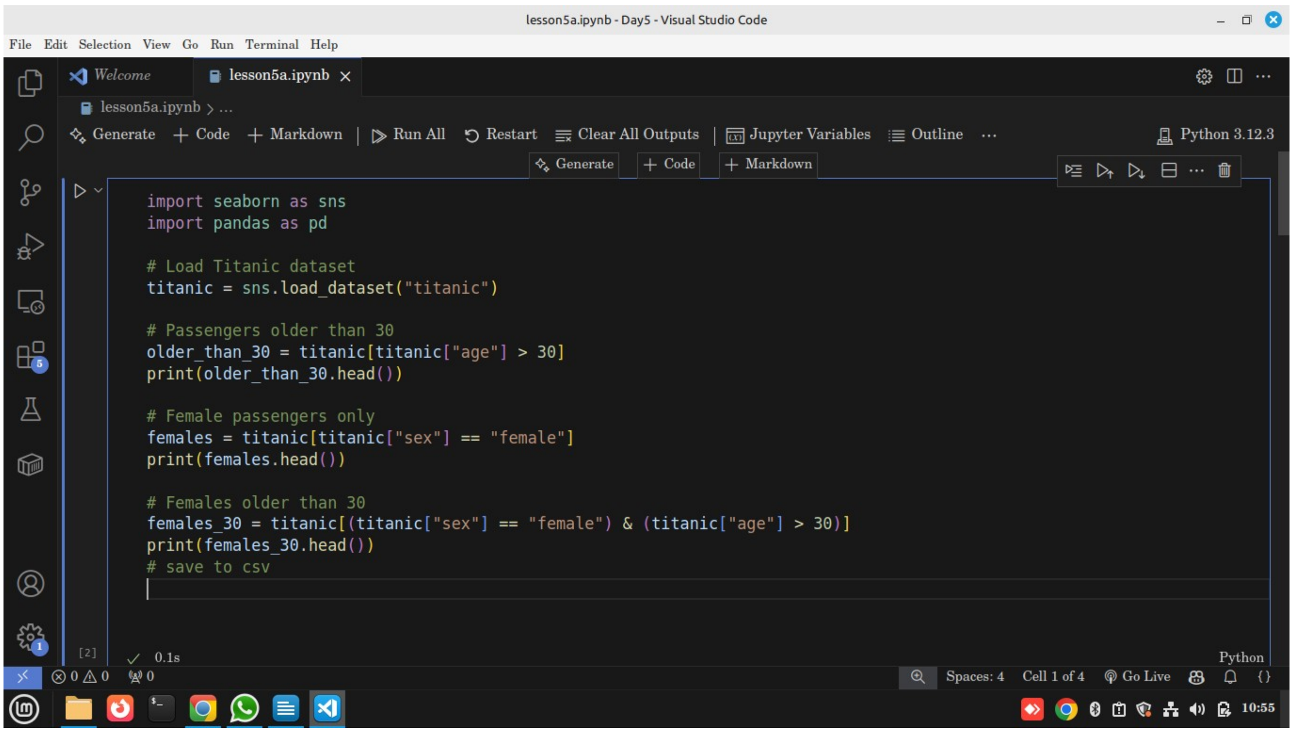
Common Wrangling Operations:

1. Filtering → Selecting specific rows.
2. Grouping → Summarizing by categories.
3. Merging → Combining datasets.
4. Pivoting → Reshaping data.

📌 **Use in Data Science**: Wrangling prepares datasets for visualization, dashboards, and machine learning models.

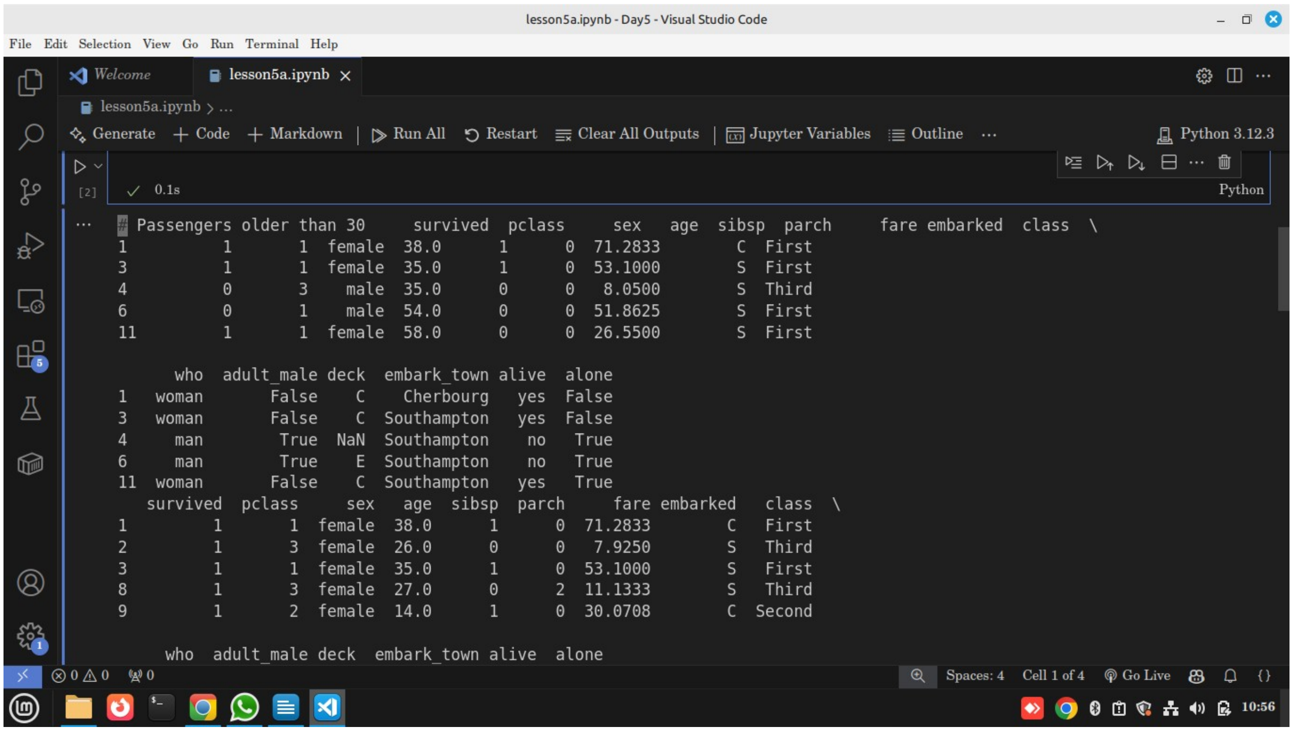
◆ Filtering

Practical



```python
import seaborn as sns
import pandas as pd

# Load Titanic dataset
titanic = sns.load_dataset("titanic")

# Passengers older than 30
older_than_30 = titanic[titanic["age"] > 30]
print(older_than_30.head())

# Female passengers only
females = titanic[titanic["sex"] == "female"]
print(females.head())

# Females older than 30
females_30 = titanic[(titanic["sex"] == "female") & (titanic["age"] > 30)]
print(females_30.head())
# save to csv
```

Sample output



```
# Passengers older than 30     survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
1           1       1  female  38.0      1      0  71.2833        C  First
3           1       1  female  35.0      1      0  53.1000        S  First
4           0       3    male  35.0      0      0   8.0500        S  Third
6           0       1    male  54.0      0      0  51.8625        S  First
11          1       1  female  58.0      0      0  26.5500        S  First

      who  adult_male deck  embark_town alive  alone
1   woman       False    C    Cherbourg   yes  False
3   woman       False    C  Southampton   yes  False
4     man        True  NaN  Southampton    no   True
6     man        True    E  Southampton    no   True
11  woman       False    C  Southampton   yes   True
   survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
1         1       1  female  38.0      1      0  71.2833        C   First
2         1       3  female  26.0      0      0   7.9250        S   Third
3         1       1  female  35.0      1      0  53.1000        S   First
8         1       3  female  27.0      0      2  11.1333        S   Third
9         1       2  female  14.0      1      0  30.0708        C  Second

     who  adult_male deck  embark_town alive  alone
```

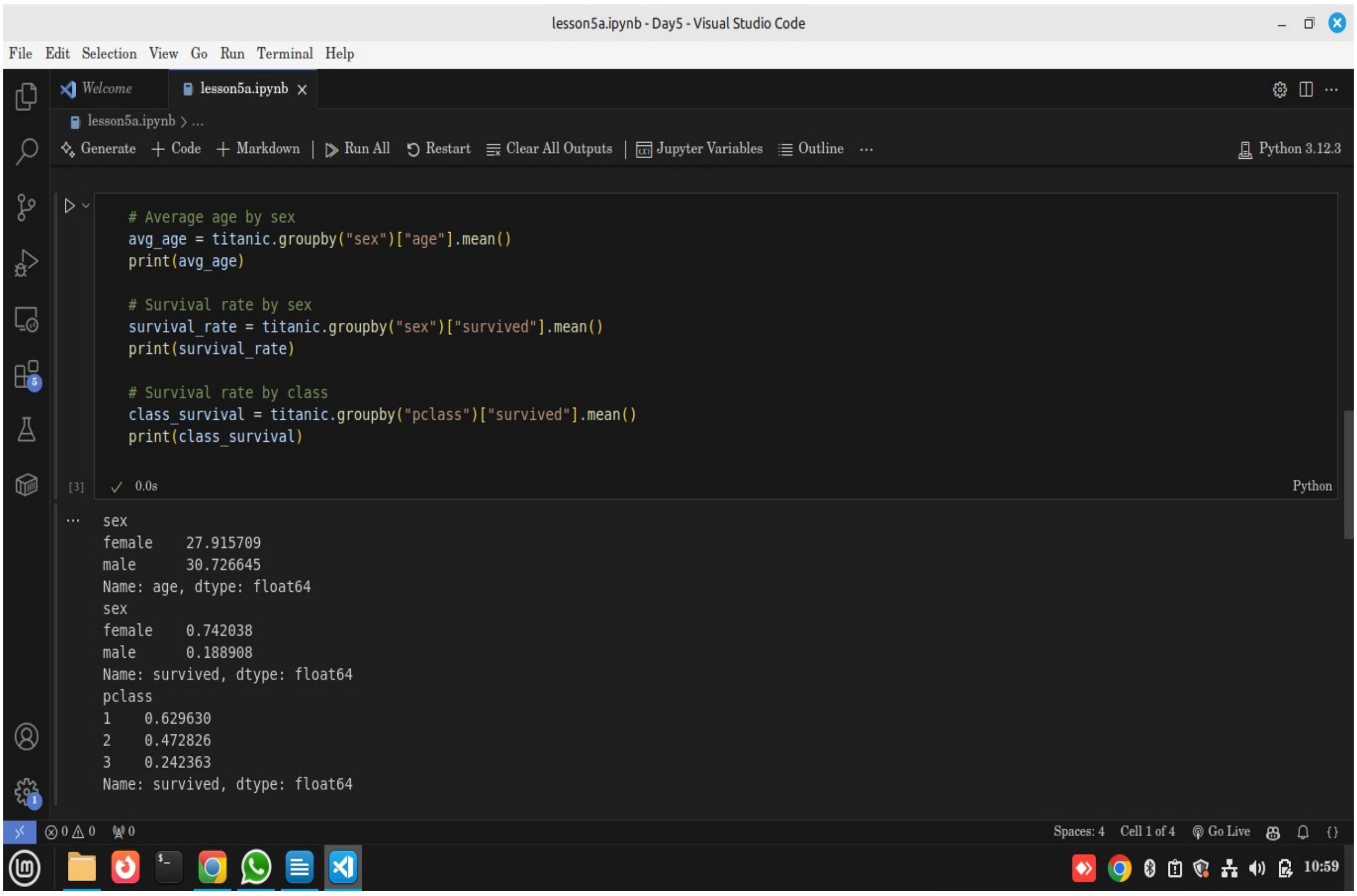
Explanation

- `df[condition]` → selects rows matching condition.
- `&` = AND, `|` = OR.
- Used for subsetting datasets.

Use in Data Science → Filtering helps analysts zoom in on specific groups (e.g., “Find customers over 40 in Nairobi with purchases > $500”).

🔹 GroupBy & Aggregations

Practical


lesson5a.ipynb - Day5 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Welcome
lesson5a.ipynb
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
Python 3.12.3

```python
# Average age by sex
avg_age = titanic.groupby("sex")["age"].mean()
print(avg_age)

# Survival rate by sex
survival_rate = titanic.groupby("sex")["survived"].mean()
print(survival_rate)

# Survival rate by class
class_survival = titanic.groupby("pclass")["survived"].mean()
print(class_survival)
```

```
sex
female    27.915709
male      30.726645
Name: age, dtype: float64
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
pclass
1    0.629630
2    0.472826
3    0.242363
Name: survived, dtype: float64
```

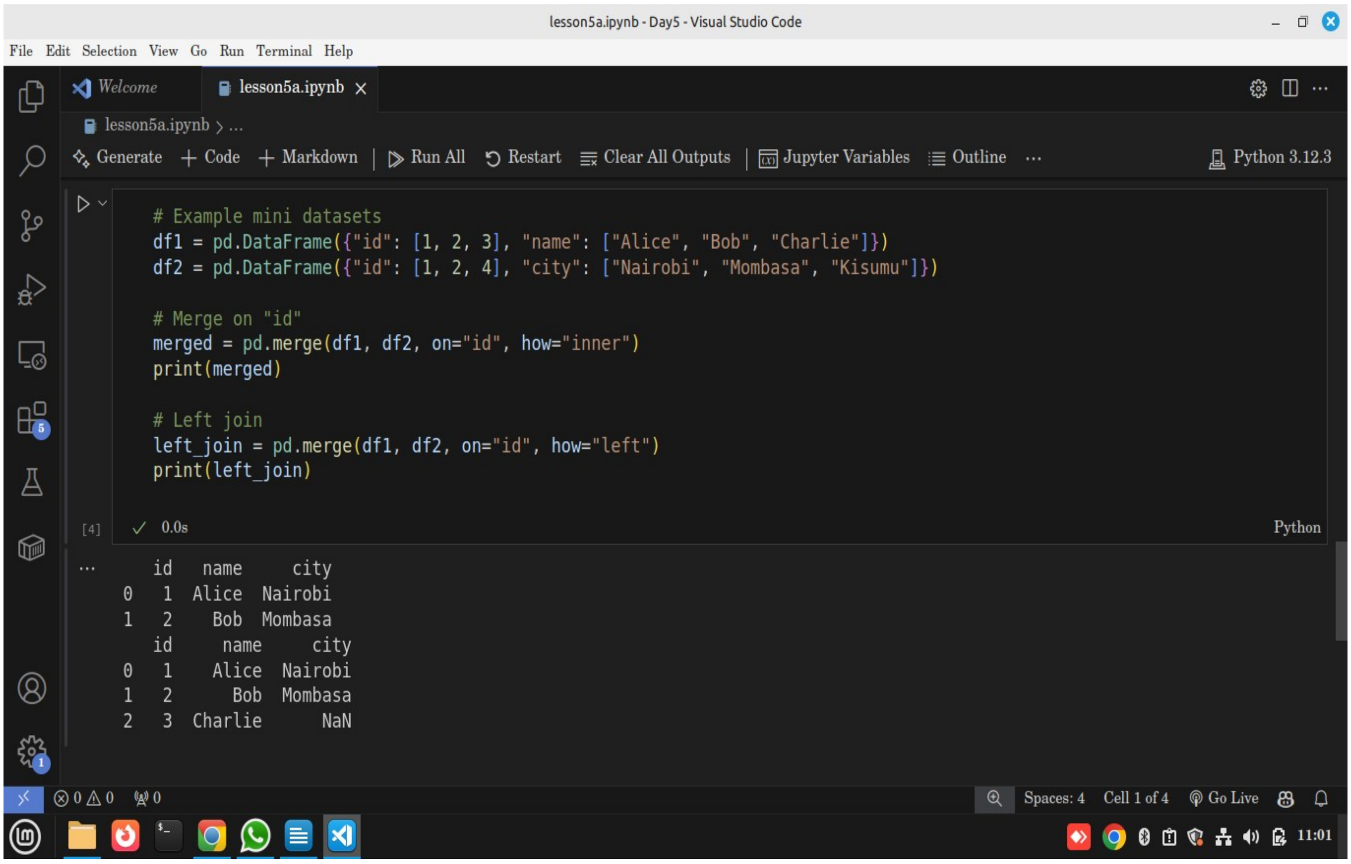
Explanation

- `groupby()` → splits data into groups.
- `.mean(), .sum(), .count()` → applies aggregation.

Use in Data Science → GroupBy helps compare segments (e.g., “average salary by department”, “churn rate by subscription plan”).

🔹 Merge & Join

Practical


lesson5a.ipynb - Day5 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Welcome
lesson5a.ipynb
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
Python 3.12.3
[4] ✓ 0.0s
Python
Spaces: 4 Cell 1 of 4 Go Live
11:01

```python
# Example mini datasets
df1 = pd.DataFrame({"id": [1, 2, 3], "name": ["Alice", "Bob", "Charlie"]})
df2 = pd.DataFrame({"id": [1, 2, 4], "city": ["Nairobi", "Mombasa", "Kisumu"]})

# Merge on "id"
merged = pd.merge(df1, df2, on="id", how="inner")
print(merged)

# Left join
left_join = pd.merge(df1, df2, on="id", how="left")
print(left_join)
```

```
   id   name     city
0   1  Alice  Nairobi
1   2    Bob  Mombasa
   id     name     city
0   1    Alice  Nairobi
1   2      Bob  Mombasa
2   3  Charlie      NaN
```

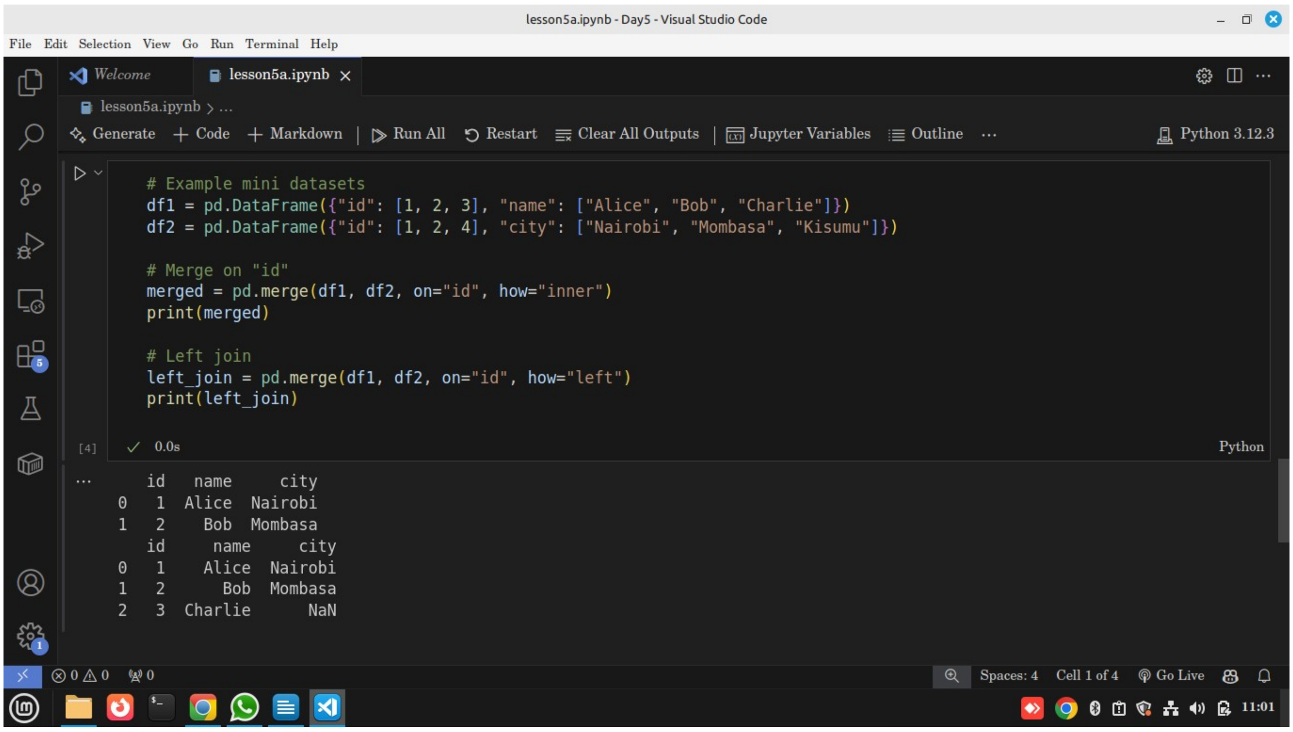
Explanation

- `pd.merge()` combines datasets by keys (like SQL joins).
- Types → `inner`, `left`, `right`, `outer`.

Use in Data Science → Merging combines info from multiple sources (e.g., transactions + customer demographics).

🔹 Pivot Tables

Practical



```python
# Example mini datasets
df1 = pd.DataFrame({"id": [1, 2, 3], "name": ["Alice", "Bob", "Charlie"]})
df2 = pd.DataFrame({"id": [1, 2, 4], "city": ["Nairobi", "Mombasa", "Kisumu"]})

# Merge on "id"
merged = pd.merge(df1, df2, on="id", how="inner")
print(merged)

# Left join
left_join = pd.merge(df1, df2, on="id", how="left")
print(left_join)
```

```
   id   name     city
0   1  Alice  Nairobi
1   2    Bob  Mombasa
   id     name     city
0   1    Alice  Nairobi
1   2      Bob  Mombasa
2   3  Charlie      NaN
```

Explanation

- Pivot tables reorganize data to show summaries across **two dimensions**.
- Example → Survival rate by sex *and* passenger class.

Use in Data Science → Pivot tables simplify comparisons (like Excel pivot tables for quick insights).

🔹 Mini Challenge

👉 On Titanic dataset:

1. Filter passengers under 18.
2. Group by class and calculate survival rate.
3. Pivot survival rates by sex and class.

🔹 Reflection

- Wrangling transforms raw data into **analysis-ready datasets**.
- Without wrangling, you can’t do visualization or ML.
- Which operation felt most powerful today? (Filtering, GroupBy, Merge, Pivot).