😊 Conditional filtering

Conditional filtering in Pandas is the process of selecting rows in a DataFrame or Series based on whether they meet certain conditions or criteria. It allows you to filter and analyze only the data that matches specific requirements, which is especially useful when working with large datasets

Use Cases:

1. **Customer Segmentation**
   Businesses can filter customer data to isolate specific segments, such as customers who made purchases over a certain amount, who live in specific regions, or who visited during a particular timeframe.
2. **Sales and Revenue Analysis**
   Analysts can filter sales records to examine transactions over a specific value, during peak seasons, or from particular product categories.
3. **Employee Data Management**
   Human resource departments can extract employees based on job titles, performance ratings, years of service, or departments for targeted planning.
4. **Quality Control in Manufacturing**
   Manufacturing data can be filtered to identify defective items, production batches with specific error codes, or units that failed quality tests.
5. **Educational Performance Tracking**
   Schools or universities can filter student data to find those above or below a certain grade threshold, enrolled in specific courses, or who meet scholarship criteria.
6. **Healthcare Data Analysis**
   Medical researchers can filter patient records based on age groups, diagnosis codes, treatment types, or outcomes for more focused studies.
7. **Financial Risk Assessment**
   Banks and financial institutions use filtering to assess loan applications based on credit scores, income levels, or existing liabilities.
8. **Environmental Data Monitoring**
   Filtering helps isolate data from specific sensors, time periods, or conditions such as temperature above a certain level or locations experiencing high pollution.

Benefits:

- **Focused Analysis:** It allows analysts to zoom in on relevant data, eliminating noise and irrelevant records.
- **Improved Decision Making:** By examining targeted data, organizations can make informed decisions faster and more accurately.
- **Efficiency:** Reduces the amount of data being processed, leading to faster computations and cleaner reports.

- **Clarity:** Simplifies complex datasets, making patterns and anomalies easier to detect.
- **Flexibility:** Conditions can be combined and customized to handle diverse analytical needs across industries.
- **Automation-Friendly:** Enables easy integration into automated scripts for regular reporting or monitoring.

😄 To begin with the Lab

1. We start by importing NumPy and Pandas. Then, using Pandas, we read a CSV file called tips.
2. Then, we displayed the first few commands of the CSV file using the df.head() command.

```
[2]: import numpy as np
     import pandas as pd

[4]: df = pd.read_csv('tips.csv')

[6]: df.head()
```

[6]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

3. This code creates a filter for rows in the DataFrame where the value in the total_bill column is greater than 30.
4. First, it generates a Boolean series (True or False for each row) by checking which rows meet the condition.
5. Then, it uses this Boolean series to extract and display only those rows from the DataFrame where the total_bill is more than 30.

```
[8]: # df['total_bill'] > 30

[10]: bool_series = df['total_bill'] > 30

[12]: df[bool_series]
```

180	34.65	3.68	Male	Yes	Sun	Dinner	4	8.66	James Hebert DDS	676168737648	Sun7544
182	45.35	3.50	Male	Yes	Sun	Dinner	3	15.12	Jose Parsons	4112207559459910	Sun2337
184	40.55	3.00	Male	Yes	Sun	Dinner	2	20.27	Stephen Cox	3547798222044029	Sun5140
187	30.46	2.00	Male	Yes	Sun	Dinner	5	6.09	David Barrett	4792882899700988	Sun9987
197	43.11	5.00	Female	Yes	Thur	Lunch	4	10.78	Brooke Soto	5544902205760175	Thur9313
207	38.73	3.00	Male	Yes	Sat	Dinner	4	9.68	Ricky Ramirez	347817964484033	Sat4505
210	30.06	2.00	Male	Yes	Sat	Dinner	3	10.02	Shawn Mendoza	30184049218122	Sat8361
212	48.33	9.00	Male	No	Sat	Dinner	4	12.08	Alex Williamson	676218815212	Sat4590
219	30.14	3.09	Female	Yes	Sat	Dinner	4	7.54	Shelby House	502097403252	Sat8863
237	32.83	1.17	Male	Yes	Sat	Dinner	2	16.42	Thomas Brown	4284722681265508	Sat2929

6. These lines of code filter the DataFrame based on specific conditions.
7. The first line returns all rows where the value in the total_bill column is greater than 30.

8. The second line returns all rows where the value in the sex column is 'Male'.
9. This allows for targeted analysis based on numerical or categorical criteria.

```
[14]: df[df['total_bill']>30]
```

[14]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
11	35.26	5.00	Female	No	Sun	Dinner	4	8.82	Diane Macias	4577817359320969	Sun6686
23	39.42	7.58	Male	No	Sat	Dinner	4	9.86	Lance Peterson	3542584061609808	Sat239
39	31.27	5.00	Male	No	Sat	Dinner	3	10.42	Mr. Brandon Berry	6011525851069856	Sat6373
44	30.40	5.60	Male	No	Sun	Dinner	4	7.60	Todd Cooper	503846761263	Sun2274
47	32.40	6.00	Male	No	Sun	Dinner	4	8.10	James Barnes	3552002592874186	Sun9677
52	34.81	5.20	Female	No	Sun	Dinner	4	8.70	Emily Daniel	4291280793094374	Sun6165
56	38.01	3.00	Male	Yes	Sat	Dinner	4	9.50	James Christensen DDS	349793629453226	Sat8903
59	48.27	6.73	Male	No	Sat	Dinner	4	12.07	Brian Ortiz	6596453823950595	Sat8139
83	32.68	5.00	Male	Yes	Thur	Lunch	2	16.34	Daniel Murphy	5356177501009133	Thur8801

```
[16]: df[df['sex'] == 'Male']
```

3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
5	25.29	4.71	Male	No	Sun	Dinner	4	6.32	Erik Smith	213140353657882	Sun9679
6	8.77	2.00	Male	No	Sun	Dinner	2	4.38	Kristopher Johnson	2223727524230344	Sun5985
...	...	...	...	...	...	...	...	...	...	...	...
236	12.60	1.00	Male	Yes	Sat	Dinner	2	6.30	Matthew Myers	3543676378973965	Sat5032
237	32.83	1.17	Male	Yes	Sat	Dinner	2	16.42	Thomas Brown	4284722681265508	Sat2929
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842	Sat2657
241	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196	Sat3880
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950	Sat17

157 rows × 11 columns

10. These lines of code demonstrate how to apply **multiple conditions** to filter rows in a DataFrame.
11. The first line selects rows where the total bill is greater than 30 and the customer is male.
12. The second and third lines return rows where the total bill is over 30 and the customer is **not** male (using two equivalent methods).
13. The last line retrieves rows for transactions that occurred on the weekend—specifically, Saturday or Sunday.
14. These logical operations help narrow down data for specific scenarios or analyses.

[18]:
```
df[(df['total_bill'] > 30) & (df['sex']=='Male')]
```

[18]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
23	39.42	7.58	Male	No	Sat	Dinner	4	9.86	Lance Peterson	3542584061609808	Sat239
39	31.27	5.00	Male	No	Sat	Dinner	3	10.42	Mr. Brandon Berry	6011525851069856	Sat6373
44	30.40	5.60	Male	No	Sun	Dinner	4	7.60	Todd Cooper	503846761263	Sun2274
47	32.40	6.00	Male	No	Sun	Dinner	4	8.10	James Barnes	3552002592874186	Sun9677
56	38.01	3.00	Male	Yes	Sat	Dinner	4	9.50	James Christensen DDS	349793629453226	Sat8903
59	48.27	6.73	Male	No	Sat	Dinner	4	12.07	Brian Ortiz	6596453823950595	Sat8139
83	32.68	5.00	Male	Yes	Thur	Lunch	2	16.34	Daniel Murphy	5356177501009133	Thur8801
95	40.17	4.73	Male	Yes	Fri	Dinner	4	10.04	Aaron Bentley	180026611638690	Fri9628
112	38.07	4.00	Male	No	Sun	Dinner	3	12.69	Jeff Lopez	3572865915176463	Sun591

[20]:
```
df[(df['total_bill'] > 30) & ~(df['sex']=='Male')]
```

[20]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
11	35.26	5.00	Female	No	Sun	Dinner	4	8.82	Diane Macias	4577817359320969	Sun6686
52	34.81	5.20	Female	No	Sun	Dinner	4	8.70	Emily Daniel	4291280793094374	Sun6165
85	34.83	5.17	Female	No	Thur	Lunch	4	8.71	Shawna Cook	6011787464177340	Thur7972
102	44.30	2.50	Female	Yes	Sat	Dinner	3	14.77	Heather Cohen	379771118886604	Sat6240
197	43.11	5.00	Female	Yes	Thur	Lunch	4	10.78	Brooke Soto	5544902205760175	Thur9313
219	30.14	3.09	Female	Yes	Sat	Dinner	4	7.54	Shelby House	502097403252	Sat8863
238	35.83	4.67	Female	No	Sat	Dinner	3	11.94	Kimberly Crane	676184013727	Sat9777

[22]:
```
df[(df['total_bill'] > 30) & (df['sex']!='Male')]
```

[22]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
11	35.26	5.00	Female	No	Sun	Dinner	4	8.82	Diane Macias	4577817359320969	Sun6686
52	34.81	5.20	Female	No	Sun	Dinner	4	8.70	Emily Daniel	4291280793094374	Sun6165
85	34.83	5.17	Female	No	Thur	Lunch	4	8.71	Shawna Cook	6011787464177340	Thur7972
102	44.30	2.50	Female	Yes	Sat	Dinner	3	14.77	Heather Cohen	379771118886604	Sat6240
197	43.11	5.00	Female	Yes	Thur	Lunch	4	10.78	Brooke Soto	5544902205760175	Thur9313
219	30.14	3.09	Female	Yes	Sat	Dinner	4	7.54	Shelby House	502097403252	Sat8863
238	35.83	4.67	Female	No	Sat	Dinner	3	11.94	Kimberly Crane	676184013727	Sat9777

[24]:
```
# The Weekend
df[(df['day'] =='Sun') | (df['day']=='Sat')]
```

[24]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251
...	...	...	...	...	...	...	...	...	...	...	...
238	35.83	4.67	Female	No	Sat	Dinner	3	11.94	Kimberly Crane	676184013727	Sat9777
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842	Sat2657
240	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404	Sat1766

15. These lines filter the DataFrame to use the **.isin()** operator to show only the rows where the day column has values that are either 'Sat' or 'Sun'.
16. The first line creates a Boolean Series indicating whether each row's day value is in the list ['Sat', 'Sun'].
17. The second line uses that condition to return only the rows corresponding to weekend days.
18. This approach is helpful when filtering based on multiple possible values in a column.

```
[26]: options = ['Sat','Sun']
      df['day'].isin(options)
```

```
[26]: 0       True
      1       True
      2       True
      3       True
      4       True
             ...
      239     True
      240     True
      241     True
      242     True
      243    False
      Name: day, Length: 244, dtype: bool
```

```
[28]: df[df['day'].isin(['Sat','Sun'])]
```

[28]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251
...	...	...	...	...	...	...	...	...	...	...	...
238	35.83	4.67	Female	No	Sat	Dinner	3	11.94	Kimberly Crane	676184013727	Sat9777
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842	Sat2657
240	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404	Sat1766