In [ ]:
```
import pandas as pd
```

In data analytics, slicing and filtering allow us to extract specific portions of a dataset that meet certain criteria. Think of slicing as cutting your dataset by position or label, and filtering as selecting rows or columns based on conditions.

In pandas, the two primary accessors are:

.iloc — integer position–based slicing

.loc — label–based slicing

step 1: read data from a file. observe the extension of the file and select the appropriate function to read it. Here we wanted to read data from sales.csv file using pandas function "read_csv" and assigned the data to a variable sales_data

In [ ]:
```
sales_data = pd.read_csv('sales.csv')
```

Use functions like head, tail, shape, and columns to get an initial understanding of data. It also verify that the data you want to use is correctly assigned to the defined variable.

In [ ]:
```
#displaying first 10 columns
sales_data.head(10)
```

Out[ ]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Countr
0	1	CA-2016-152156	2016/11/08	2016/11/11	Second Class	CG-12520	Claire Gute	Consumer	Unite Stat
1	2	CA-2016-152156	2016/11/08	2016/11/11	Second Class	CG-12520	Claire Gute	Consumer	Unite Stat
2	3	CA-2016-138688	2016/06/12	2016/06/16	Second Class	DV-13045	Darrin Van Huff	Corporate	Unite Stat
3	4	US-2015-108966	2015/10/11	2015/10/18	Standard Class	SO-20335	Sean O'Donnell	Consumer	Unite Stat
4	5	US-2015-108966	2015/10/11	2015/10/18	Standard Class	SO-20335	Sean O'Donnell	Consumer	Unite Stat
5	6	CA-2014-115812	2014/06/09	2014/06/14	Standard Class	BH-11710	Brosina Hoffman	Consumer	Unite Stat
6	7	CA-2014-115812	2014/06/09	2014/06/14	Standard Class	BH-11710	Brosina Hoffman	Consumer	Unite Stat
7	8	CA-2014-115812	2014/06/09	2014/06/14	Standard Class	BH-11710	Brosina Hoffman	Consumer	Unite Stat
8	9	CA-2014-115812	2014/06/09	2014/06/14	Standard Class	BH-11710	Brosina Hoffman	Consumer	Unite Stat
9	10	CA-2014-115812	2014/06/09	2014/06/14	Standard Class	BH-11710	Brosina Hoffman	Consumer	Unite Stat

10 rows × 21 columns

In [ ]:
```
#displaying last 10 columns
sales_data.tail(10)
```

Out[ ]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment
9984	9985	CA-2015-100251	2015/05/17	2015/05/23	Standard Class	DV-13465	Dianna Vittorini	Consumer
9985	9986	CA-2015-100251	2015/05/17	2015/05/23	Standard Class	DV-13465	Dianna Vittorini	Consumer
9986	9987	CA-2016-125794	2016/09/29	2016/10/03	Standard Class	ML-17410	Maris LaWare	Consumer
9987	9988	CA-2017-163629	2017/11/17	2017/11/21	Standard Class	RA-19885	Ruben Ausman	Corporate
9988	9989	CA-2017-163629	2017/11/17	2017/11/21	Standard Class	RA-19885	Ruben Ausman	Corporate
9989	9990	CA-2014-110422	2014/01/21	2014/01/23	Second Class	TB-21400	Tom Boeckenhauer	Consumer
9990	9991	CA-2017-121258	2017/02/26	2017/03/03	Standard Class	DB-13060	Dave Brooks	Consumer
9991	9992	CA-2017-121258	2017/02/26	2017/03/03	Standard Class	DB-13060	Dave Brooks	Consumer
9992	9993	CA-2017-121258	2017/02/26	2017/03/03	Standard Class	DB-13060	Dave Brooks	Consumer
9993	9994	CA-2017-119914	2017/05/04	2017/05/09	Second Class	CC-12220	Chris Cortes	Consumer

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment

10 rows × 21 columns

In [ ]:
```
#shape function give us the information about number of columns and rows in the dat
sales_data.shape
```

Out[ ]:
```
(9994, 21)
```

In [ ]:
```
#further, we can check column names using column function
sales_data.columns
```

Out[ ]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

Now we have an initial understanding of our dataset. we know how many rows and columns it has and what are the names of columns which we wanted to use for data analysis.

In [ ]:
```
# Select a single column from the DataFrame
# When we use a single column name inside square brackets,
# pandas returns a "Series" — a one-dimensional array-like structure.
sales = sales_data["Sales"]

# Display the first five values to verify the selection
print(sales.head())
```

```
0    261.9600
1    731.9400
2     14.6200
3    957.5775
4     22.3680
Name: Sales, dtype: float64
```

In [ ]:
```
# Select multiple columns from the DataFrame
# When we pass a list of column names inside double square brackets,
# pandas returns a new "DataFrame" with just those columns.
subset = sales_data[["Sales", "Profit", "Category"]]

# Show the first few rows to confirm we have 3 columns now
print(subset.head())
```

```
     Sales    Profit         Category
0  261.9600   41.9136        Furniture
1  731.9400  219.5820        Furniture
2   14.6200    6.8714  Office Supplies
3  957.5775 -383.0310        Furniture
4   22.3680    2.5164  Office Supplies
```

Note the distinction between a Series and a DataFrame early. type(df["col"]) and type(df[["col"]]) give pandas.Series and pandas.DataFrame respectively.

In [ ]:

Position-based Slicing — .iloc

Concept: .iloc (short for integer location) is used to access rows and columns by their numeric positions rather than labels.
It behaves like standard Python list slicing:
-- Indexing starts at 0
-- The end index is exclusive (:5 means indices 0–4)
-- You can use negative indices to count from the end

This is useful when:
-- You want to quickly preview or subset data by row number
-- The DataFrame does not have meaningful labels or an index

In [ ]:
```
# Display the first five rows and all columns
# .iloc[:5, :] means:
#   - ":5" → take rows from index 0 up to (but not including) 5
#   - ":"  → take all columns
first_five = sales_data.iloc[:5, :]
print(first_five)
```

```
   Row ID        Order ID  Order Date   Ship Date       Ship Mode Customer ID  \
0       1  CA-2016-152156  2016/11/08  2016/11/11    Second Class    CG-12520
1       2  CA-2016-152156  2016/11/08  2016/11/11    Second Class    CG-12520
2       3  CA-2016-138688  2016/06/12  2016/06/16    Second Class    DV-13045
3       4  US-2015-108966  2015/10/11  2015/10/18  Standard Class    SO-20335
4       5  US-2015-108966  2015/10/11  2015/10/18  Standard Class    SO-20335

     Customer Name    Segment        Country             City  ...  \
0      Claire Gute   Consumer  United States        Henderson  ...
1      Claire Gute   Consumer  United States        Henderson  ...
2  Darrin Van Huff  Corporate  United States      Los Angeles  ...
3   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...
4   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...

  Postal Code  Region       Product ID         Category Sub-Category  \
0       42420   South  FUR-BO-10001798        Furniture    Bookcases
1       42420   South  FUR-CH-10000454        Furniture       Chairs
2       90036    West  OFF-LA-10000240  Office Supplies       Labels
3       33311   South  FUR-TA-10000577        Furniture       Tables
4       33311   South  OFF-ST-10000760  Office Supplies      Storage

                                        Product Name     Sales  Quantity  \
0                 Bush Somerset Collection Bookcase  261.9600         2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...  731.9400         3
2  Self-Adhesive Address Labels for Typewriters b...   14.6200         2
3      Bretford CR4500 Series Slim Rectangular Table  957.5775         5
4                    Eldon Fold 'N Roll Cart System   22.3680         2

   Discount    Profit
0      0.00   41.9136
1      0.00  219.5820
2      0.00    6.8714
3      0.45 -383.0310
4      0.20    2.5164

[5 rows x 21 columns]
```

In [ ]:
```python
# Select specific rows and specific columns by integer positions
# Syntax: df.iloc[row_slice, column_slice]
# Below: rows 10–14 and columns 0–3
subset = sales_data.iloc[10:15, 0:4]
print(subset)
```

```
    Row ID        Order ID  Order Date   Ship Date
10      11  CA-2014-115812  2014/06/09  2014/06/14
11      12  CA-2014-115812  2014/06/09  2014/06/14
12      13  CA-2017-114412  2017/04/15  2017/04/20
13      14  CA-2016-161389  2016/12/05  2016/12/10
14      15  US-2015-118983  2015/11/22  2015/11/26
```

In [ ]:
```python
# Select non-contiguous rows and columns by using lists of indices
# Here we pick rows 0, 50, and 100 — and columns 1 and 3
# This is handy for sampling arbitrary positions.
sample_subset = sales_data.iloc[[0, 50, 100], [1, 3]]
print(sample_subset)
```

```
          Order ID   Ship Date
0    CA-2016-152156  2016/11/11
50   CA-2015-115742  2015/04/22
100  CA-2016-158568  2016/09/02
```

In [ ]:
```
# Using negative indices with .iloc
# "-5:" means "start from the fifth row from the bottom until the end"
last_five_rows = sales_data.iloc[-5:, :]
print(last_five_rows)
```

```
      Row ID        Order ID  Order Date   Ship Date       Ship Mode  \
9989    9990  CA-2014-110422  2014/01/21  2014/01/23    Second Class
9990    9991  CA-2017-121258  2017/02/26  2017/03/03  Standard Class
9991    9992  CA-2017-121258  2017/02/26  2017/03/03  Standard Class
9992    9993  CA-2017-121258  2017/02/26  2017/03/03  Standard Class
9993    9994  CA-2017-119914  2017/05/04  2017/05/09    Second Class

     Customer ID     Customer Name   Segment        Country         City  ...  \
9989    TB-21400  Tom Boeckenhauer  Consumer  United States        Miami  ...
9990    DB-13060       Dave Brooks  Consumer  United States   Costa Mesa  ...
9991    DB-13060       Dave Brooks  Consumer  United States   Costa Mesa  ...
9992    DB-13060       Dave Brooks  Consumer  United States   Costa Mesa  ...
9993    CC-12220      Chris Cortes  Consumer  United States  Westminster  ...

     Postal Code  Region       Product ID         Category Sub-Category  \
9989       33180   South  FUR-FU-10001889        Furniture  Furnishings
9990       92627    West  FUR-FU-10000747        Furniture  Furnishings
9991       92627    West  TEC-PH-10003645       Technology       Phones
9992       92627    West  OFF-PA-10004041  Office Supplies        Paper
9993       92683    West  OFF-AP-10002684  Office Supplies   Appliances

                                        Product Name    Sales  Quantity  \
9989                           Ultra Door Pull Handle   25.248         3
9990  Tenex B1-RE Series Chair Mats for Low Pile Car...   91.960         2
9991                            Aastra 57i VoIP phone  258.576         2
9992  It's Hot Message Books with Stickers, 2 3/4" x 5"   29.600         4
9993  Acco 7-Outlet Masterpiece Power Center, Wihtou...  243.160         2

      Discount   Profit
9989       0.2   4.1028
9990       0.0  15.6332
9991       0.2  19.3932
9992       0.0  13.3200
9993       0.0  72.9480

[5 rows x 21 columns]
```

In [ ]:
```
# Selecting a single cell by position
# Syntax: df.iloc[row_index, column_index]
# Example: value at row 0, column 0
value = sales_data.iloc[0, 0]
print("Value at first cell:", value)
```

```
Value at first cell: 1
```

In [ ]:
```
# You can also use slicing to skip rows
# For instance, take every 1000th row from the dataset
```

```
# Syntax: df.iloc[start:end:step, :]
every_thousandth = sales_data.iloc[::1000, :]
print(every_thousandth)
```

```
      Row ID        Order ID  Order Date   Ship Date       Ship Mode  \
0          1  CA-2016-152156  2016/11/08  2016/11/11    Second Class
1000    1001  CA-2016-155488  2016/11/13  2016/11/17  Standard Class
2000    2001  CA-2017-166128  2017/04/11  2017/04/18  Standard Class
3000    3001  CA-2014-138317  2014/06/21  2014/06/25  Standard Class
4000    4001  CA-2014-116834  2014/10/11  2014/10/16  Standard Class
5000    5001  CA-2017-159688  2017/05/07  2017/05/12  Standard Class
6000    6001  US-2014-112991  2014/12/10  2014/12/14  Standard Class
7000    7001  CA-2017-123687  2017/05/26  2017/05/29     First Class
8000    8001  US-2015-151407  2015/11/08  2015/11/12  Standard Class
9000    9001  CA-2014-133389  2014/06/22  2014/06/22        Same Day

     Customer ID    Customer Name      Segment        Country          City  \
0       CG-12520      Claire Gute     Consumer  United States     Henderson
1000    FM-14290     Frank Merwin  Home Office  United States     Vancouver
2000    LW-17215       Luke Weiss     Consumer  United States      Pasadena
3000    NW-18400   Natalie Webber     Consumer  United States  Philadelphia
4000    Dp-13240      Dean percer  Home Office  United States       Seattle
5000    AB-10060  Adam Bellavance  Home Office  United States   Los Angeles
6000    SH-19975    Sally Hughsby    Corporate  United States      Caldwell
7000    KC-16675  Kimberly Carter    Corporate  United States    Louisville
8000    RD-19585         Rob Dowd     Consumer  United States       Dubuque
9000    TB-21280  Toby Braunhardt     Consumer  United States       Phoenix

      ... Postal Code   Region       Product ID         Category Sub-Category  \
0     ...       42420    South  FUR-BO-10001798        Furniture    Bookcases
1000  ...       98661     West  OFF-AR-10002956  Office Supplies          Art
2000  ...       91104     West  TEC-AC-10001767       Technology  Accessories
3000  ...       19120     East  OFF-AP-10003860  Office Supplies   Appliances
4000  ...       98115     West  FUR-FU-10001196        Furniture  Furnishings
5000  ...       90004     West  TEC-AC-10000736       Technology  Accessories
6000  ...       83605     West  OFF-PA-10002222  Office Supplies        Paper
7000  ...       40214    South  OFF-SU-10004498  Office Supplies     Supplies
8000  ...       52001  Central  TEC-PH-10003885       Technology       Phones
9000  ...       85023     West  OFF-BI-10001553  Office Supplies      Binders

                                          Product Name    Sales  Quantity  \
0                    Bush Somerset Collection Bookcase  261.960         2
1000    Boston 16801 Nautilus Battery Pencil Sharpener   44.020         2
2000  SanDisk Ultra 64 GB MicroSDHC Class 10 Memory ...  199.950         5
3000  Fellowes Advanced 8 Outlet Surge Suppressor wi...   44.416         2
4000                         DAX Cubicle Frames - 8x10   63.470        11
5000                  Logitech G600 MMO Gaming Mouse     79.990         1
6000          Xerox Color Copier Paper, 11" x 17", Ream   91.360         4
7000                 Martin-Yale Premier Letter Opener   25.760         2
8000                                   Cisco SPA508G  263.960         4
9000  SpineVue Locking Slant-D Ring Binders by Cardinal    8.226         3

      Discount   Profit
0          0.0  41.9136
1000       0.0  11.4452
2000       0.0  21.9945
3000       0.2   3.8864
4000       0.0  19.0410
5000       0.0  28.7964
6000       0.0  42.0256
```

```
7000       0.0   0.7728
8000       0.0  76.5484
9000       0.7  -6.0324

[10 rows x 21 columns]
```

.iloc does not care about column names or index labels — only their numeric positions.

Exercise:

1. Display the first 8 rows and columns 2–5 using .iloc.
2. Retrieve the last 10 rows using negative indices.
3. Print the Sales value (by position) in the 100th row.

Label-based Slicing — .loc

Concept: .loc (short for location by label) is used to select rows and columns by their names (labels) instead of numeric positions. It works directly with index labels and column names, not integer positions like .iloc.

It's particularly powerful when:

-- The DataFrame has a meaningful index (like Order IDs or dates).

-- You want to perform inclusive slicing (both start and end labels are included).

In [ ]:
```
#Select *all rows* but *only certain columns* by name.

# ":" means "all rows"; the list is the selected columns.

only_cols = sales_data.loc[:, ["Order Date", "Category", "Sales", "Profit"]]
print(only_cols.head())
```

```
  Order Date         Category     Sales    Profit
0  2016/11/08        Furniture  261.9600   41.9136
1  2016/11/08        Furniture  731.9400  219.5820
2  2016/06/12  Office Supplies   14.6200    6.8714
3  2015/10/11        Furniture  957.5775 -383.0310
4  2015/10/11  Office Supplies   22.3680    2.5164
```

In [ ]:
```
#Convert 'Order Date' column to datetime format
sales_data["Order Date"] = pd.to_datetime(sales_data["Order Date"])
```

In [ ]:
```
sales_date = sales_data.set_index("Order Date")
```

In [ ]:
```
sales_date = sales_date.sort_index()
```

In [ ]:
```
subset = sales_date.loc["2017-01-01":"2017-03-31", ["Sales", "Profit"]]
```

In [ ]:
```
subset
```

Out[ ]:

Order Date	Sales	Profit
2017-01-01	3.6000	1.7280
2017-01-01	48.8960	8.5568
2017-01-01	474.4300	199.2606
2017-01-01	310.7440	-26.6352
2017-01-01	12.7360	2.2288
...	...	...
2017-03-31	84.9500	22.0870
2017-03-31	29.7800	8.0406
2017-03-31	75.0400	36.0192
2017-03-31	205.3328	-36.2352
2017-03-31	677.5800	176.1708

500 rows × 2 columns

Exercise:

1. Slice the first quarter of 2017. Retrieve all orders between January 1, 2017 and March 31, 2017, showing only "Sales", "Profit", and "Category" columns.
2. Find all summer orders in 2018. Slice the data between June 1, 2018 and August 31, 2018. Show "Region", "Sales", and "Quantity".

Boolean Filtering (Masks)

Concept: Boolean filtering means selecting rows where one or more conditions are True. You create a Boolean mask (a Series of True/False values) and use it to filter the DataFrame.

It's called a mask because it "masks out" the rows that don't meet your conditions.

In [ ]:
```
# Every comparison (>, <, ==, etc.) returns True or False
sales_data["Profit"] > 100
```

Out[ ]:
```
0       False
1        True
2       False
3       False
4       False
        ...
9989    False
9990    False
9991    False
9992    False
9993    False
Name: Profit, Length: 9994, dtype: bool
```

In [ ]:
```python
# Example 1: Filter rows where Profit > 100
high_profit = sales[sales_data["Profit"] > 100]
print(high_profit.head())
```

```
1      731.940
13     407.976
24    1044.630
35    1097.544
54    1029.950
Name: Sales, dtype: float64
```

In [ ]:
```python
# Example 2: Combine multiple conditions using logical operators
# &  → AND   |  → OR   ~  → NOT
# Always use parentheses around each condition
filtered = sales_data[(sales_data["Profit"] > 100) & (sales_data["Sales"] > 500)]
print(filtered.head())
```

```
    Row ID        Order ID Order Date   Ship Date       Ship Mode Customer ID  \
1        2  CA-2016-152156 2016-11-08  2016/11/11    Second Class    CG-12520
24      25  CA-2015-106320 2015-09-25  2015/09/30  Standard Class    EB-13870
35      36  CA-2016-117590 2016-12-08  2016/12/10     First Class    GH-14485
54      55  CA-2016-105816 2016-12-11  2016/12/17  Standard Class    JM-15265
67      68  CA-2014-106376 2014-12-05  2014/12/10  Standard Class    BS-11590

     Customer Name    Segment        Country           City  ... Postal Code  \
1      Claire Gute   Consumer  United States      Henderson  ...       42420
24     Emily Burns   Consumer  United States          Orem  ...       84057
35       Gene Hale  Corporate  United States     Richardson  ...       75080
54  Janet Molinari  Corporate  United States  New York City  ...       10024
67   Brendan Sweed  Corporate  United States        Gilbert  ...       85234

     Region       Product ID         Category Sub-Category  \
1     South  FUR-CH-10000454        Furniture       Chairs
24     West  FUR-TA-10000577        Furniture       Tables
35  Central  TEC-PH-10004977       Technology       Phones
54     East  TEC-PH-10002447       Technology       Phones
67     West  OFF-AR-10002671  Office Supplies          Art

                                        Product Name     Sales  Quantity  \
1   Hon Deluxe Fabric Upholstered Stacking Chairs,...   731.940         3
24     Bretford CR4500 Series Slim Rectangular Table  1044.630         3
35                                     GE 30524EE4  1097.544         7
54                 AT&T CL83451 4-Handset Telephone  1029.950         5
67  Hunt BOSTON Model 1606 High-Volume Electric Pe...  1113.024         8

    Discount    Profit
1        0.0  219.5820
24       0.0  240.2649
35       0.2  123.4737
54       0.0  298.6855
67       0.2  111.3024

[5 rows x 21 columns]
```

In [ ]:
```
# Example 3: Using .loc for more explicit filtering
# Syntax: df.loc[row_condition, columns_to_display]
result = sales_data.loc[(sales_data["Category"] == "Furniture") & (sales_data["Prof
                ["Category", "Sales", "Profit"]]
print(result.head())
```

```
     Category     Sales    Profit
1   Furniture   731.940  219.5820
10  Furniture  1706.184   85.3092
24  Furniture  1044.630  240.2649
117 Furniture   787.530  165.3813
149 Furniture  1951.840  585.5520
```

In [ ]:
```
# Example 4: Using .isin() for membership filtering
# Keep rows where Category is Furniture or Technology
subset = sales_data[sales_data["Category"].isin(["Furniture", "Technology"])]
print(subset.head())
```

```
   Row ID        Order ID Order Date   Ship Date       Ship Mode Customer ID  \
0       1  CA-2016-152156 2016-11-08  2016/11/11    Second Class    CG-12520
1       2  CA-2016-152156 2016-11-08  2016/11/11    Second Class    CG-12520
3       4  US-2015-108966 2015-10-11  2015/10/18  Standard Class    SO-20335
5       6  CA-2014-115812 2014-06-09  2014/06/14  Standard Class    BH-11710
7       8  CA-2014-115812 2014-06-09  2014/06/14  Standard Class    BH-11710

     Customer Name   Segment        Country             City  ... Postal Code  \
0      Claire Gute  Consumer  United States        Henderson  ...       42420
1      Claire Gute  Consumer  United States        Henderson  ...       42420
3   Sean O'Donnell  Consumer  United States  Fort Lauderdale  ...       33311
5  Brosina Hoffman  Consumer  United States      Los Angeles  ...       90032
7  Brosina Hoffman  Consumer  United States      Los Angeles  ...       90032

   Region       Product ID    Category Sub-Category  \
0   South  FUR-BO-10001798   Furniture    Bookcases
1   South  FUR-CH-10000454   Furniture       Chairs
3   South  FUR-TA-10000577   Furniture       Tables
5    West  FUR-FU-10001487   Furniture  Furnishings
7    West  TEC-PH-10002275  Technology       Phones

                                        Product Name     Sales  Quantity  \
0                 Bush Somerset Collection Bookcase  261.9600         2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...  731.9400         3
3      Bretford CR4500 Series Slim Rectangular Table  957.5775         5
5  Eldon Expressions Wood and Plastic Desk Access...   48.8600         7
7                    Mitel 5320 IP Phone VoIP phone  907.1520         6

   Discount    Profit
0      0.00   41.9136
1      0.00  219.5820
3      0.45 -383.0310
5      0.00   14.1694
7      0.20   90.7152

[5 rows x 21 columns]
```

In [ ]:
```python
# Example 5: Using .between() for range filtering
# Select rows where Sales are between 100 and 300 (inclusive)
range_subset = sales_data[sales_data["Sales"].between(100, 300)]
print(range_subset.head())
```

```
    Row ID        Order ID Order Date   Ship Date       Ship Mode Customer ID  \
0        1  CA-2016-152156 2016-11-08  2016/11/11    Second Class    CG-12520
9       10  CA-2014-115812 2014-06-09  2014/06/14  Standard Class    BH-11710
19      20  CA-2014-143336 2014-08-27  2014/09/01    Second Class    ZD-21925
29      30  US-2015-150630 2015-09-17  2015/09/21  Standard Class    TB-21520
36      37  CA-2016-117590 2016-12-08  2016/12/10     First Class    GH-14485

         Customer Name    Segment        Country           City  ...  \
0          Claire Gute   Consumer  United States      Henderson  ...
9      Brosina Hoffman   Consumer  United States    Los Angeles  ...
19  Zuschuss Donatelli   Consumer  United States  San Francisco  ...
29     Tracy Blumstein   Consumer  United States   Philadelphia  ...
36           Gene Hale  Corporate  United States     Richardson  ...

   Postal Code   Region       Product ID         Category Sub-Category  \
0        42420    South  FUR-BO-10001798        Furniture    Bookcases
9        90032     West  OFF-AP-10002892  Office Supplies   Appliances
19       94109     West  TEC-PH-10001949       Technology       Phones
29       19140     East  FUR-FU-10004848        Furniture  Furnishings
36       75080  Central  FUR-FU-10003664        Furniture  Furnishings

                                        Product Name   Sales  Quantity  \
0                    Bush Somerset Collection Bookcase  261.96         2
9                     Belkin F5C206VTEL 6 Outlet Surge  114.90         5
19                            Cisco SPA 501G IP Phone  213.48         3
29  Howard Miller 13-3/4" Diameter Brushed Chrome ...  124.20         3
36  Electrix Architect's Clamp-On Swing Arm Lamp, ...  190.92         5

    Discount    Profit
0        0.0   41.9136
9        0.0   34.4700
19       0.2   16.0110
29       0.2   15.5250
36       0.6 -147.9630

[5 rows x 21 columns]
```

Conditional Filtering with .where()

Concept: .where() is not used to select rows like df[mask]; instead, it's used to keep values that meet a condition and replace the rest with NaN.

Think of it as a soft filter — it marks what doesn't fit the condition instead of removing it.

In [ ]:
```
# Example 1: Keep Sales greater than 500, replace others with NaN
filtered_sales = sales_data["Sales"].where(sales_data["Sales"] > 500)
print(filtered_sales.head(10))
```

```
0         NaN
1    731.9400
2         NaN
3    957.5775
4         NaN
5         NaN
6         NaN
7    907.1520
8         NaN
9         NaN
Name: Sales, dtype: float64
```

In [ ]:
```
# Example 2: Apply .where() on the whole DataFrame
# Keep only positive Profit values; all others become NaN
positive_profit = sales_data.where(sales_data["Profit"] > 0)
print(positive_profit.head())
```

```
   Row ID        Order ID Order Date   Ship Date       Ship Mode Customer ID  \
0     1.0  CA-2016-152156 2016-11-08  2016/11/11    Second Class    CG-12520
1     2.0  CA-2016-152156 2016-11-08  2016/11/11    Second Class    CG-12520
2     3.0  CA-2016-138688 2016-06-12  2016/06/16    Second Class    DV-13045
3     NaN             NaN        NaT         NaN             NaN         NaN
4     5.0  US-2015-108966 2015-10-11  2015/10/18  Standard Class    SO-20335

     Customer Name    Segment        Country             City  ...  \
0      Claire Gute   Consumer  United States        Henderson  ...
1      Claire Gute   Consumer  United States        Henderson  ...
2  Darrin Van Huff  Corporate  United States      Los Angeles  ...
3              NaN        NaN            NaN              NaN  ...
4   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...

  Postal Code  Region       Product ID         Category Sub-Category  \
0     42420.0   South  FUR-BO-10001798        Furniture    Bookcases
1     42420.0   South  FUR-CH-10000454        Furniture       Chairs
2     90036.0    West  OFF-LA-10000240  Office Supplies       Labels
3         NaN     NaN              NaN              NaN          NaN
4     33311.0   South  OFF-ST-10000760  Office Supplies      Storage

                                        Product Name    Sales  Quantity  \
0                 Bush Somerset Collection Bookcase  261.960       2.0
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...  731.940       3.0
2  Self-Adhesive Address Labels for Typewriters b...   14.620       2.0
3                                                NaN      NaN       NaN
4                   Eldon Fold 'N Roll Cart System   22.368       2.0

   Discount    Profit
0       0.0   41.9136
1       0.0  219.5820
2       0.0    6.8714
3       NaN       NaN
4       0.2    2.5164

[5 rows x 21 columns]
```

In [ ]:
```
# Example 3: Use the `other` parameter to replace False values with something custo
# Here we mark low sales as 'Low'
```

```
sales_flag = sales_data["Sales"].where(sales_data["Sales"] > 500, other="Low")
print(sales_flag.head(10))
```

```
0         Low
1      731.94
2         Low
3    957.5775
4         Low
5         Low
6         Low
7     907.152
8         Low
9         Low
Name: Sales, dtype: object
```

In [ ]:
```
# Example 4: Combine with multiple conditions
# Keep Profit > 100 AND Sales > 500, else mark NaN
filtered = sales_data.where((sales_data["Profit"] > 100) & (sales_data["Sales"] > 5
print(filtered.head())
```

```
   Row ID        Order ID Order Date   Ship Date     Ship Mode Customer ID  \
0     NaN             NaN        NaT         NaN           NaN         NaN
1     2.0  CA-2016-152156 2016-11-08  2016/11/11  Second Class    CG-12520
2     NaN             NaN        NaT         NaN           NaN         NaN
3     NaN             NaN        NaT         NaN           NaN         NaN
4     NaN             NaN        NaT         NaN           NaN         NaN

  Customer Name   Segment        Country       City  ... Postal Code  Region  \
0           NaN       NaN            NaN        NaN  ...         NaN     NaN
1   Claire Gute  Consumer  United States  Henderson  ...     42420.0   South
2           NaN       NaN            NaN        NaN  ...         NaN     NaN
3           NaN       NaN            NaN        NaN  ...         NaN     NaN
4           NaN       NaN            NaN        NaN  ...         NaN     NaN

       Product ID   Category Sub-Category  \
0             NaN        NaN          NaN
1  FUR-CH-10000454  Furniture       Chairs
2             NaN        NaN          NaN
3             NaN        NaN          NaN
4             NaN        NaN          NaN

                                        Product Name   Sales  Quantity  \
0                                                NaN     NaN       NaN
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...  731.94       3.0
2                                                NaN     NaN       NaN
3                                                NaN     NaN       NaN
4                                                NaN     NaN       NaN

   Discount   Profit
0       NaN      NaN
1       0.0  219.582
2       NaN      NaN
3       NaN      NaN
4       NaN      NaN

[5 rows x 21 columns]
```

Finding and Filtering Missing Values

Concept: In pandas, missing or empty entries are represented as NaN (Not a Number). You can find, count, or filter them using two simple and powerful functions:

isna() or isnull() → checks where data is missing

notna() or notnull() → checks where data is present

In [ ]:
```
# Example 1: Detect missing values in the entire DataFrame
# Returns True where a value is missing (NaN)
missing_mask = sales_data.isna()
print(missing_mask.head())
```

```
   Row ID  Order ID  Order Date  Ship Date  Ship Mode  Customer ID  \
0   False     False       False      False      False        False
1   False     False       False      False      False        False
2   False     False       False      False      False        False
3   False     False       False      False      False        False
4   False     False       False      False      False        False

   Customer Name  Segment  Country   City  ...  Postal Code  Region  \
0          False    False    False  False  ...        False   False
1          False    False    False  False  ...        False   False
2          False    False    False  False  ...        False   False
3          False    False    False  False  ...        False   False
4          False    False    False  False  ...        False   False

   Product ID  Category  Sub-Category  Product Name  Sales  Quantity  \
0       False     False         False         False  False     False
1       False     False         False         False  False     False
2       False     False         False         False  False     False
3       False     False         False         False  False     False
4       False     False         False         False  False     False

   Discount  Profit
0     False   False
1     False   False
2     False   False
3     False   False
4     False   False

[5 rows x 21 columns]
```

In [ ]:
```
# Example 2: Count total missing values per column
# sum() adds up True values (True = 1, False = 0)
missing_counts = sales_data.isna().sum()
print(missing_counts)
```

```
Row ID          0
Order ID        0
Order Date      0
Ship Date       0
Ship Mode       0
Customer ID     0
Customer Name   0
Segment         0
Country         0
City            0
State           0
Postal Code     0
Region          0
Product ID      0
Category        0
Sub-Category    0
Product Name    0
Sales           0
Quantity        0
Discount        0
Profit          0
dtype: int64
```

In [ ]:
```python
# Example 3: Filter rows where a specific column has missing values
# Shows only rows where Discount is missing
missing_discount = sales_data[sales_data["Discount"].isna()]
print(missing_discount.head())
```

```
Empty DataFrame
Columns: [Row ID, Order ID, Order Date, Ship Date, Ship Mode, Customer ID, Customer
Name, Segment, Country, City, State, Postal Code, Region, Product ID, Category, Sub-
Category, Product Name, Sales, Quantity, Discount, Profit]
Index: []

[0 rows x 21 columns]
```

In [ ]:
```python
# Example 4: Keep only rows with valid (non-missing) Profit values
valid_profit = sales_data[sales_data["Profit"].notna()]
print(valid_profit.head())
```

```
   Row ID        Order ID Order Date   Ship Date       Ship Mode Customer ID  \
0       1  CA-2016-152156 2016-11-08  2016/11/11    Second Class    CG-12520
1       2  CA-2016-152156 2016-11-08  2016/11/11    Second Class    CG-12520
2       3  CA-2016-138688 2016-06-12  2016/06/16    Second Class    DV-13045
3       4  US-2015-108966 2015-10-11  2015/10/18  Standard Class    SO-20335
4       5  US-2015-108966 2015-10-11  2015/10/18  Standard Class    SO-20335

     Customer Name    Segment        Country             City  ...  \
0      Claire Gute   Consumer  United States        Henderson  ...
1      Claire Gute   Consumer  United States        Henderson  ...
2  Darrin Van Huff  Corporate  United States      Los Angeles  ...
3   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...
4   Sean O'Donnell   Consumer  United States  Fort Lauderdale  ...

  Postal Code  Region       Product ID         Category Sub-Category  \
0       42420   South  FUR-BO-10001798        Furniture    Bookcases
1       42420   South  FUR-CH-10000454        Furniture       Chairs
2       90036    West  OFF-LA-10000240  Office Supplies       Labels
3       33311   South  FUR-TA-10000577        Furniture       Tables
4       33311   South  OFF-ST-10000760  Office Supplies      Storage

                                        Product Name     Sales  Quantity  \
0                 Bush Somerset Collection Bookcase  261.9600         2
1  Hon Deluxe Fabric Upholstered Stacking Chairs,...  731.9400         3
2  Self-Adhesive Address Labels for Typewriters b...   14.6200         2
3      Bretford CR4500 Series Slim Rectangular Table  957.5775         5
4                    Eldon Fold 'N Roll Cart System   22.3680         2

   Discount    Profit
0      0.00   41.9136
1      0.00  219.5820
2      0.00    6.8714
3      0.45 -383.0310
4      0.20    2.5164

[5 rows x 21 columns]
```

In [ ]:
```
# Example 5: Drop rows that contain any missing values
# Be careful — this removes data permanently unless you make a copy.
clean_df = sales_data.dropna()
print(clean_df.shape)
```

```
(9994, 21)
```

Filtering Practice Exercises (Using Superstore Dataset) 🔹 Task 1 — Simple Conditional Filtering

Display all rows where Profit is greater than 200.

Find all rows where Sales are less than 100.

Show only Category, Sales, and Profit for rows where Profit < 0.

Task 2 — Combined Conditions

Filter orders where Profit > 100 and Sales > 500.

Select orders where Category is "Furniture" or "Technology".

Find all rows where Region is not "West" and Discount < 0.2`.

Task 3 — Range-based Filtering

Display rows where Sales are between 200 and 800 (inclusive).

Select rows where Discount is between 0.1 and 0.3.

Find all rows where Profit is between -50 and 50.

(Hint: use .between(start, end) for each.)

Task 4 — Membership Filtering

Show all orders where Ship Mode is either "First Class" or "Same Day".

Filter rows where Segment belongs to ["Consumer", "Home Office"].

Display rows where State is one of ["California", "Texas", "New York"].

(Hint: use .isin(list))

- Task 5 — Missing Value Filtering

Find all rows where Discount is missing.

Keep only rows where Profit is not missing.

Count total missing values in each column using .isna().sum().

Task 6 — Conditional Replacement using .where()

Keep only rows where Profit > 100, replace others with NaN.

Replace low profits (≤ 0) with "Loss" using the other parameter.

Keep rows where Sales > 500, replace others with 0.

- Task 7 — Combined Filtering Challenge

Use .loc and conditions together:

Display all orders from the Technology category where:

Profit > 200

Sales between 300 and 800

Columns shown: ["Category", "Sales", "Profit", "Region"]