PYTHON: PANDAS AND MANIPULATION DATA

Anita Mila Oktafani

DQLab Live Class Data Analyst with SQL & Python in Google Platform

TABLE OF CONTENT

PANDAS

PANDAS

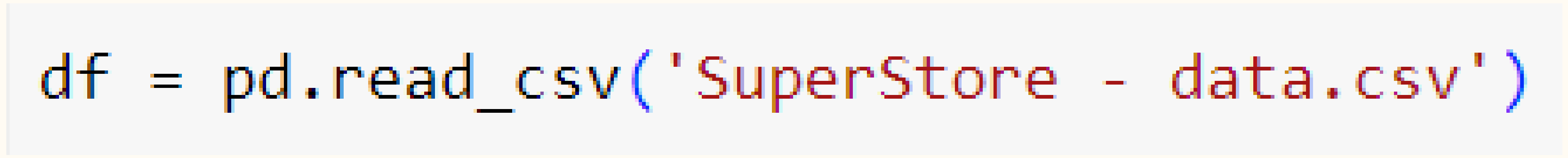
- Pandas is a python library used for data manipulation and analysis
- Pandas has a certain data structure, namely DataFrame, which makes the data analysis process easier
- To be able to use pandas, import pandas into the program using:

```
import pandas as pd
```

READ DATA WITH PANDAS

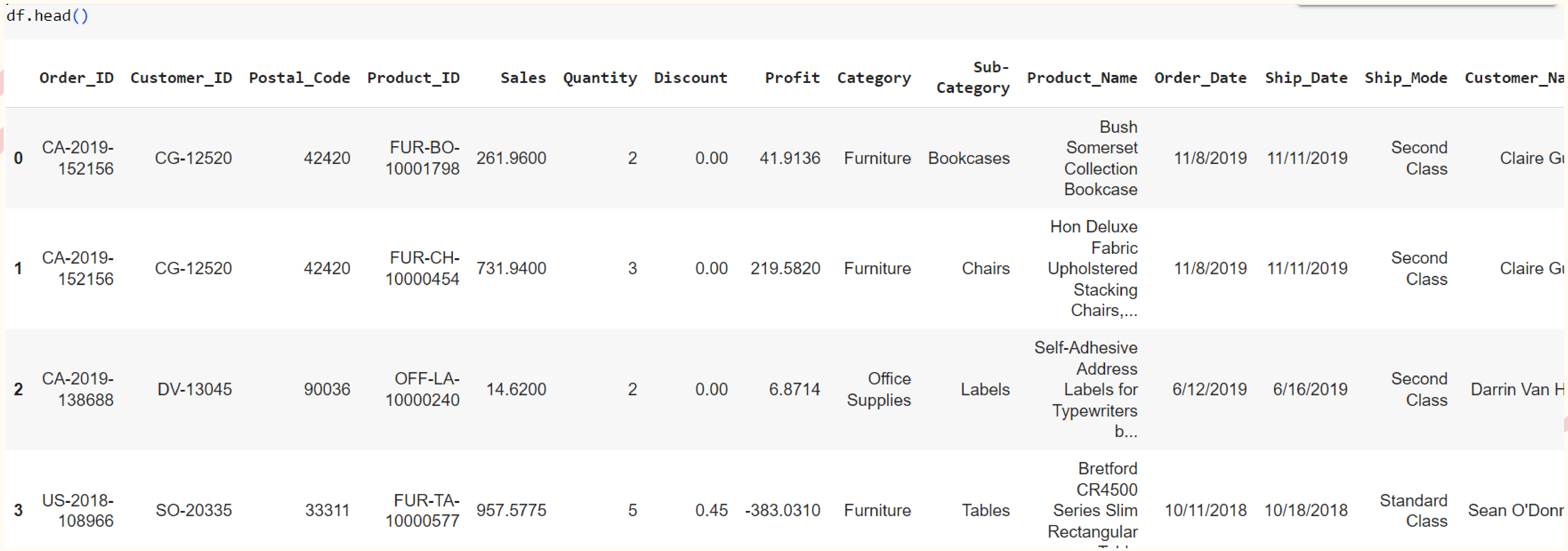
READ FILE WITH PANDAS

- Use `pd.read_csv ( <file_name_csv> )` to read the data from file csv into dataframe, for example:



```
df = pd.read_csv('SuperStore - data.csv')
```

- To show top 5 rows from df use `df.head()`

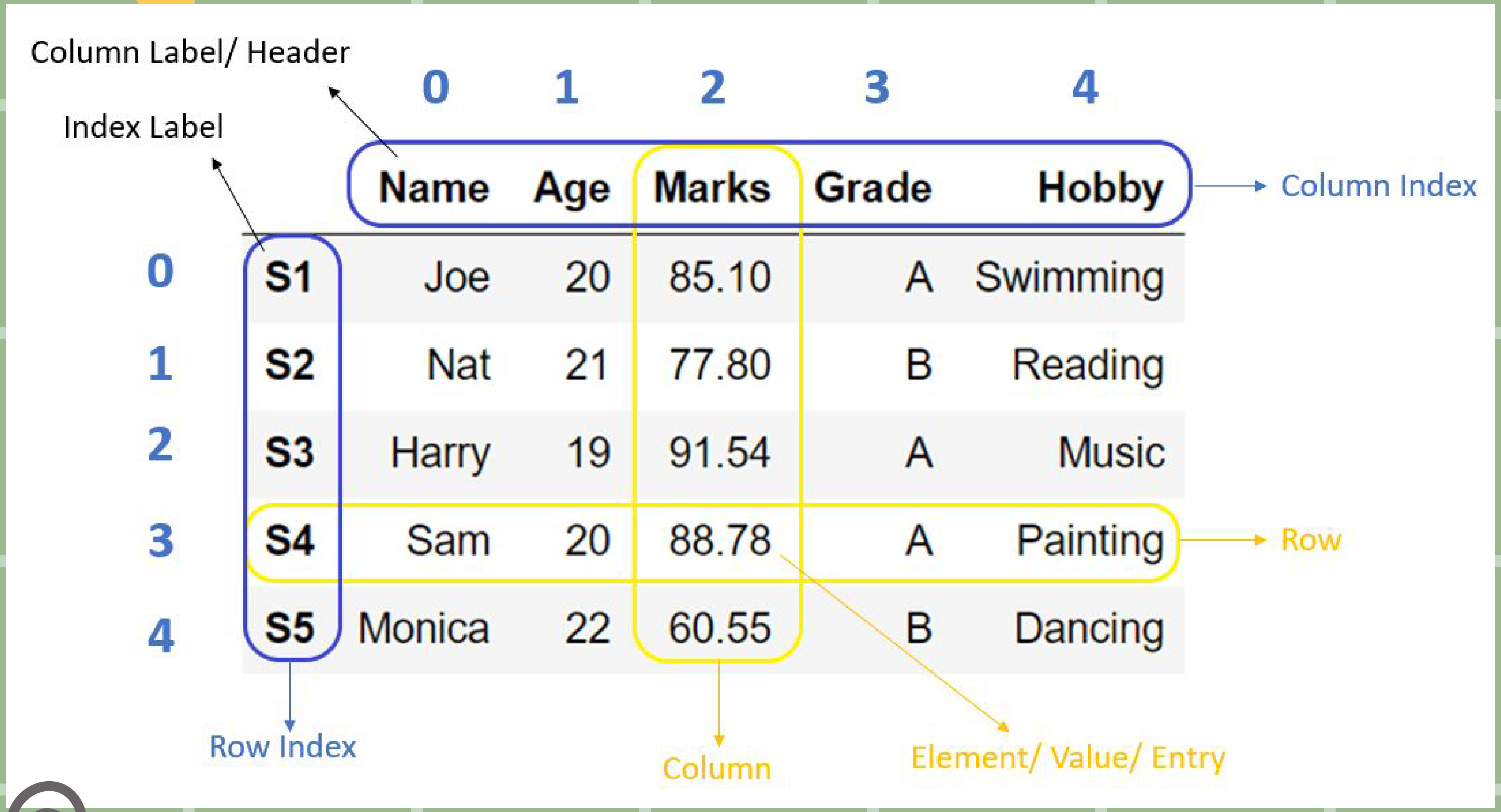


```
df.head()
```

	Order_ID	Customer_ID	Postal_Code	Product_ID	Sales	Quantity	Discount	Profit	Category	Sub-Category	Product_Name	Order_Date	Ship_Date	Ship_Mode	Customer_Na
0	CA-2019-152156	CG-12520	42420	FUR-BO-10001798	261.9600	2	0.00	41.9136	Furniture	Bookcases	Bush Somerset Collection Bookcase	11/8/2019	11/11/2019	Second Class	Claire Gu
1	CA-2019-152156	CG-12520	42420	FUR-CH-10000454	731.9400	3	0.00	219.5820	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	11/8/2019	11/11/2019	Second Class	Claire Gu
2	CA-2019-138688	DV-13045	90036	OFF-LA-10000240	14.6200	2	0.00	6.8714	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	6/12/2019	6/16/2019	Second Class	Darrin Van H
3	US-2018-108966	SO-20335	33311	FUR-TA-10000577	957.5775	5	0.45	-383.0310	Furniture	Tables	Bretford CR4500 Series Slim Rectangular	10/11/2018	10/18/2018	Standard Class	Sean O'Donn

CONSEPT OF DATAFRAME AND SERIES

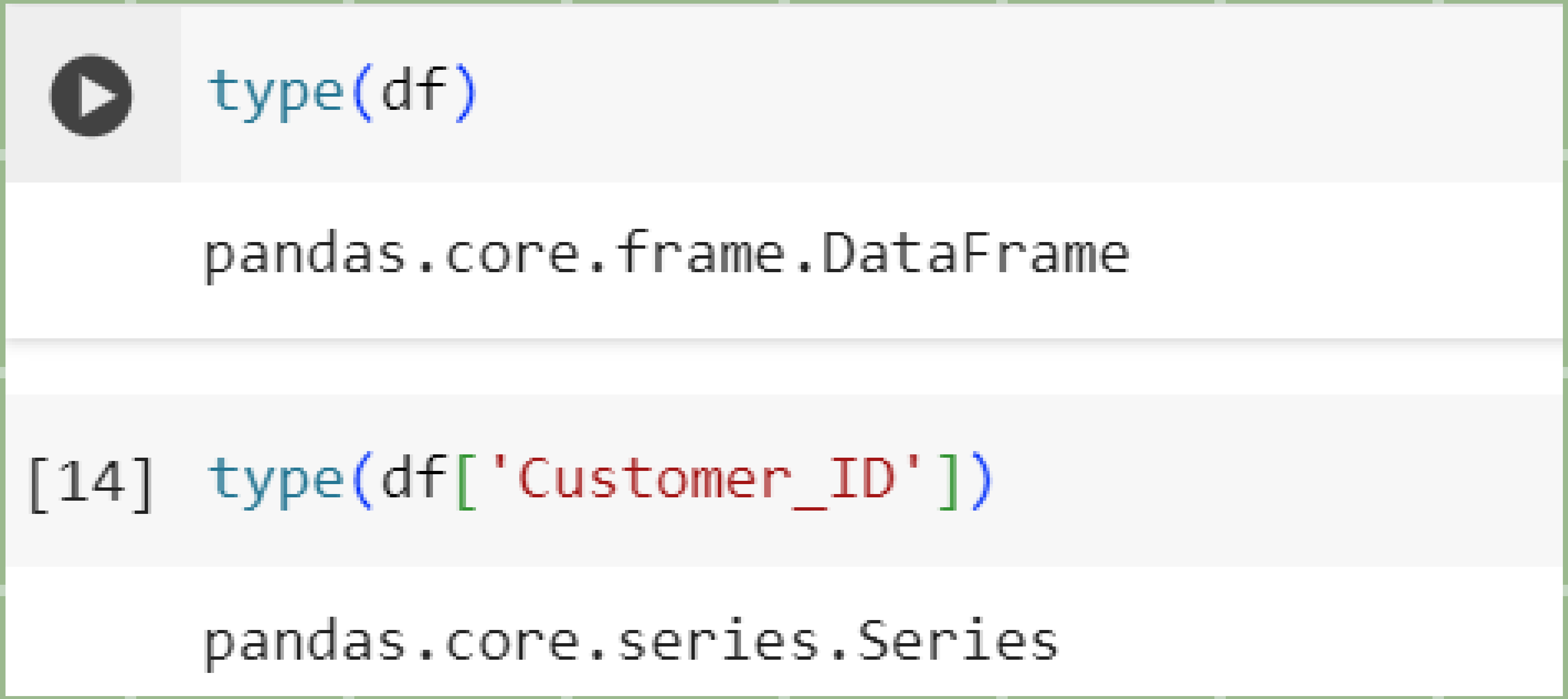
DATAFRAME AND SERIES



	Name	Age	Marks	Grade	Hobby
S1	Joe	20	85.10	A	Swimming
S2	Nat	21	77.80	B	Reading
S3	Harry	19	91.54	A	Music
S4	Sam	20	88.78	A	Painting
S5	Monica	22	60.55	B	Dancing

DATAFRAME is a table, while each column is a SERIES

- DATAFRAME is a 2-dimensional data structure (table) consisting of rows and columns. DataFrame consists of various series of the same length
- Each column of a dataframe may have a different data type, but the data in the same column has the same data type
- SERIES is homogeneous 1-dimensional data



```
type(df)
```

```
pandas.core.frame.DataFrame
```

```
[14] type(df['Customer_ID'])
```

```
pandas.core.series.Series
```

SELECTION

- To select one particular column from a Dataframe use

df [< column_name >]

```
df['Category']
```

- If more than one column is selected, use it

df [< list_columns_name >]

```
df[['Category','Customer_ID','City']]
```

or

```
nama_kolom = ['Category','Customer_ID','City']
df[nama_kolom]
```

SELECTION WITH LOC AND ILOC

- To select or retrieve part of a dataframe down to rows and columns certain things can be done in two ways: loc and iloc

- In loc and iloc, the row in question is a range of index

01

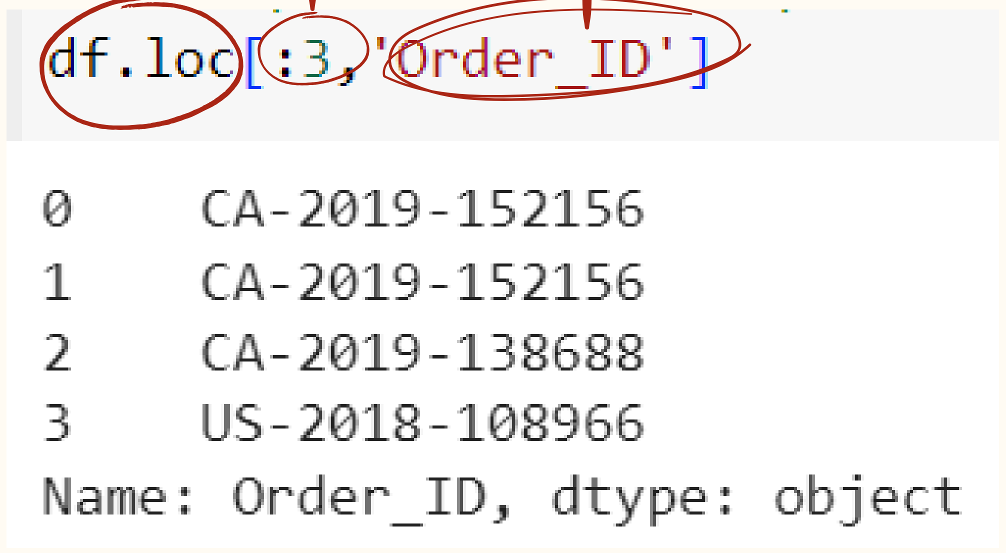
In loc, to call a column just use the column name or a list containing column names (if there is more than one column)

02

In iloc, columns are called using the index of the column df.columns functions to display a list of columns

LOC VS ILOC

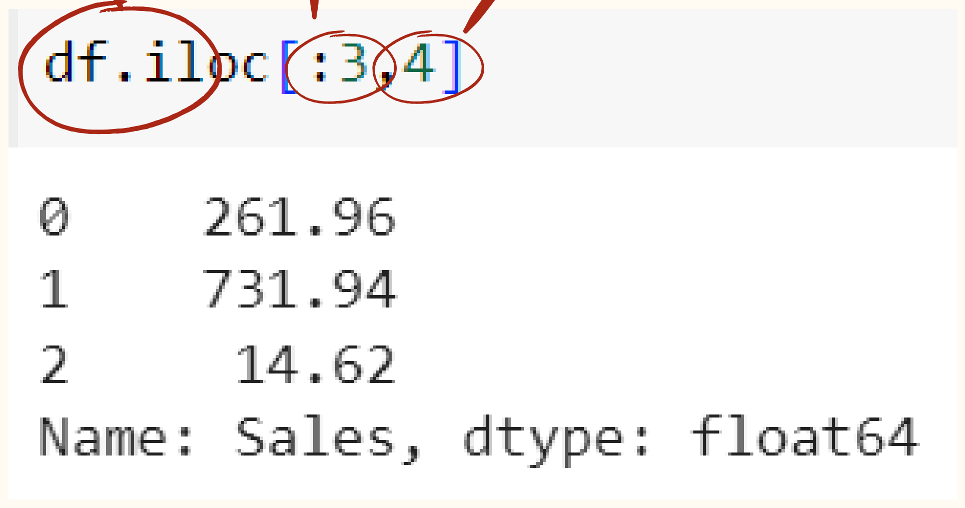
df.loc [< row > : < column_name >]



```
df.loc[:3,'Order_ID']
```

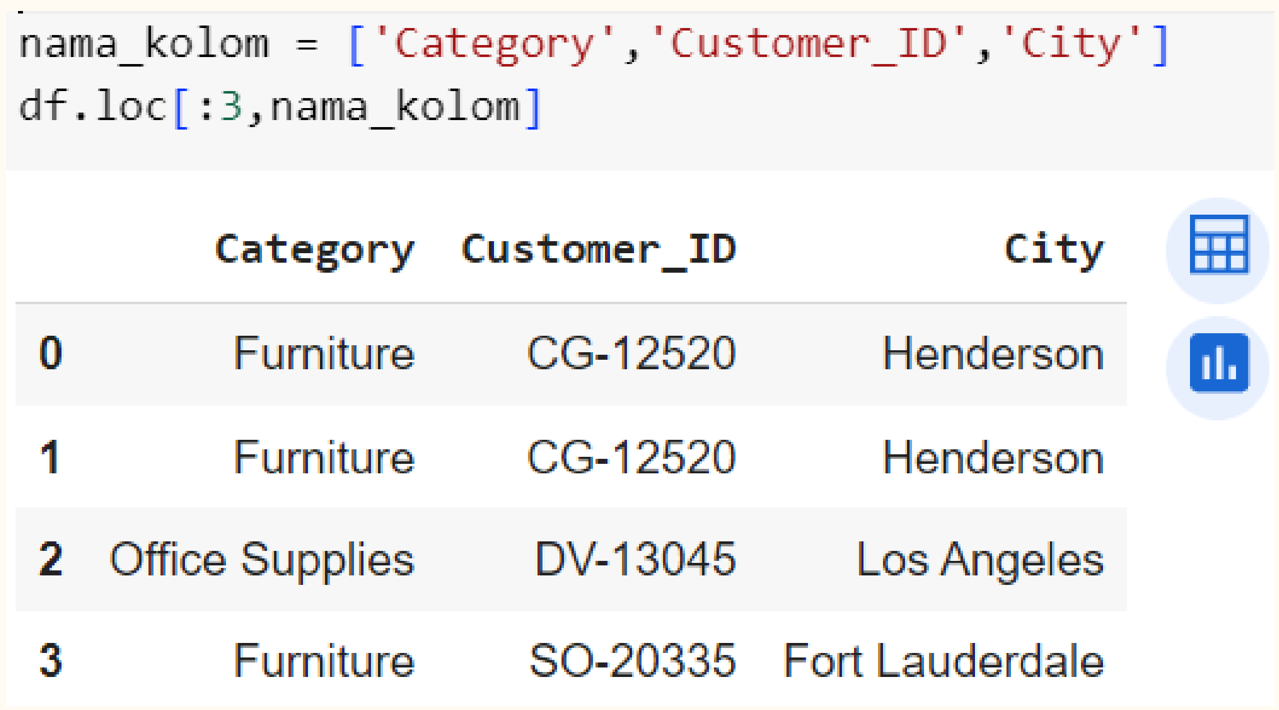
```
0    CA-2019-152156
1    CA-2019-152156
2    CA-2019-138688
3    US-2018-108966
Name: Order_ID, dtype: object
```

df.iloc [< row > : < column_index >]



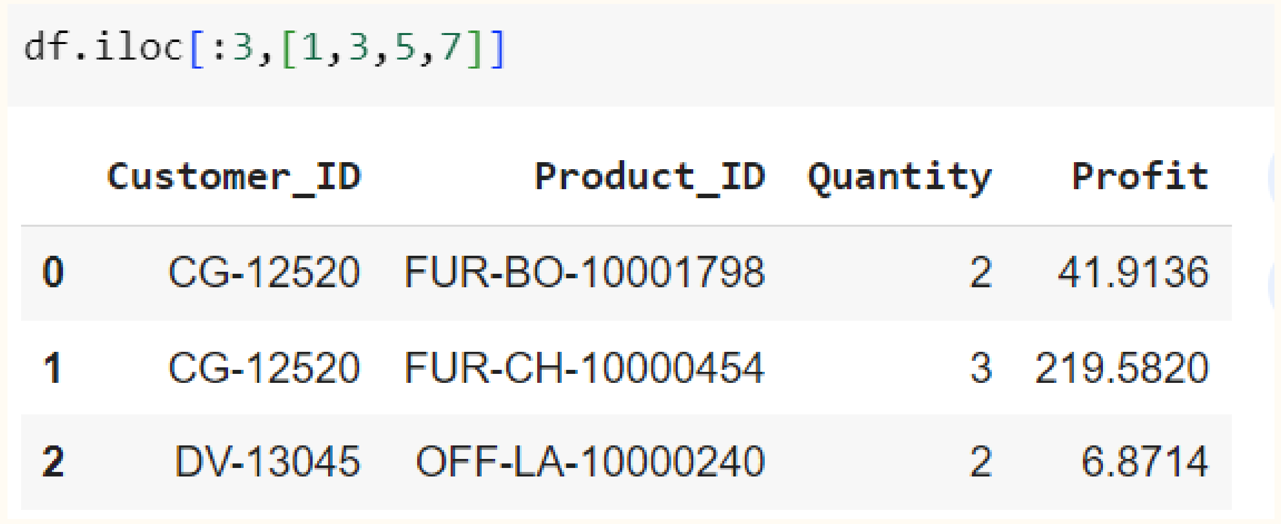
```
df.iloc[:3,4]
```

```
0    261.96
1    731.94
2     14.62
Name: Sales, dtype: float64
```



```
nama_kolom = ['Category','Customer_ID','City']
df.loc[:3,nama_kolom]
```

	Category	Customer_ID	City
0	Furniture	CG-12520	Henderson
1	Furniture	CG-12520	Henderson
2	Office Supplies	DV-13045	Los Angeles
3	Furniture	SO-20335	Fort Lauderdale



```
df.iloc[:3,[1,3,5,7]]
```

	Customer_ID	Product_ID	Quantity	Profit
0	CG-12520	FUR-BO-10001798	2	41.9136
1	CG-12520	FUR-CH-10000454	3	219.5820
2	DV-13045	OFF-LA-10000240	2	6.8714

FILTERING DATA

FILTERING

FILTERING is used to select rows that meet certain conditions

df [< condition >]

If there is <u>more than one condition</u>, separate the conditions with parentheses and connect them with bitwise operators like

- & for AND

```
df[(df['Sales'] > 100) & (df['Category'] == 'Furniture')].head()
```

- | for OR

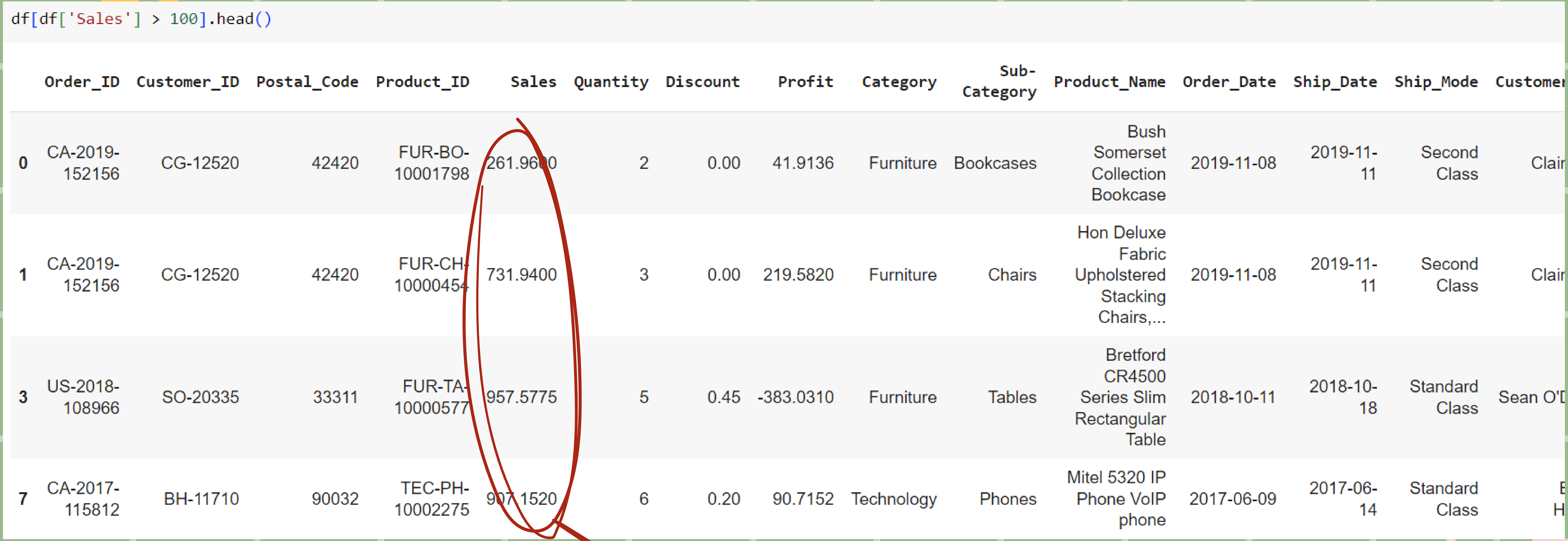
```
df[(df['City']=='Henderson') | (df['City']=='Los Angeles')].head()
```

- – for NOT

```
df[~(df['Category'] == 'Furniture')].head()
```

ONE CONDITION

```python
df[df['Sales'] > 100].head()
```



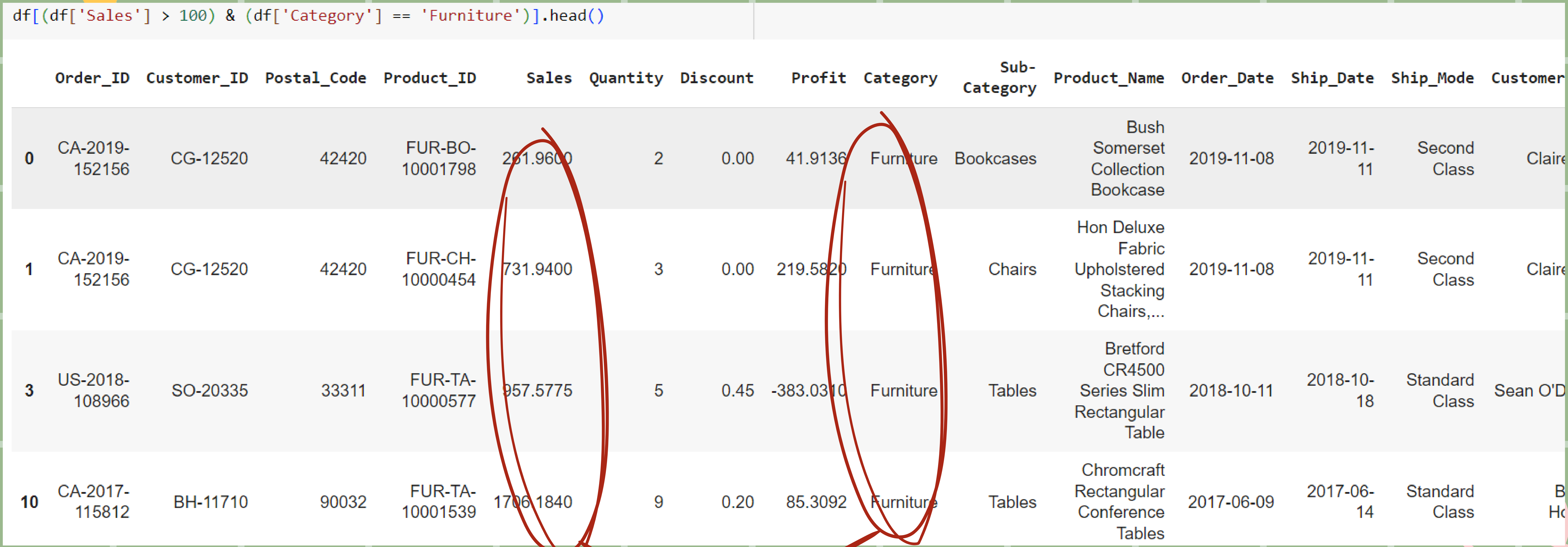
```python
df[df['Sales'] > 100].head()
```

	Order_ID	Customer_ID	Postal_Code	Product_ID	Sales	Quantity	Discount	Profit	Category	Sub-Category	Product_Name	Order_Date	Ship_Date	Ship_Mode	Custome
0	CA-2019-152156	CG-12520	42420	FUR-BO-10001798	261.9600	2	0.00	41.9136	Furniture	Bookcases	Bush Somerset Collection Bookcase	2019-11-08	2019-11-11	Second Class	Clai
1	CA-2019-152156	CG-12520	42420	FUR-CH-10000454	731.9400	3	0.00	219.5820	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	2019-11-08	2019-11-11	Second Class	Clai
3	US-2018-108966	SO-20335	33311	FUR-TA-10000577	957.5775	5	0.45	-383.0310	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	2018-10-11	2018-10-18	Standard Class	Sean O'D
7	CA-2017-115812	BH-11710	90032	TEC-PH-10002275	907.1520	6	0.20	90.7152	Technology	Phones	Mitel 5320 IP Phone VoIP phone	2017-06-09	2017-06-14	Standard Class	B H

'Sales' show the value > 100

TWO CONDITION

```
df[(df['Sales'] > 100) & (df['Category'] == 'Furniture')].head()
```



```
df[(df['Sales'] > 100) & (df['Category'] == 'Furniture')].head()
```

	Order_ID	Customer_ID	Postal_Code	Product_ID	Sales	Quantity	Discount	Profit	Category	Sub-Category	Product_Name	Order_Date	Ship_Date	Ship_Mode	Customer
0	CA-2019-152156	CG-12520	42420	FUR-BO-10001798	261.9600	2	0.00	41.9136	Furniture	Bookcases	Bush Somerset Collection Bookcase	2019-11-08	2019-11-11	Second Class	Claire
1	CA-2019-152156	CG-12520	42420	FUR-CH-10000454	731.9400	3	0.00	219.5820	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	2019-11-08	2019-11-11	Second Class	Claire
3	US-2018-108966	SO-20335	33311	FUR-TA-10000577	957.5775	5	0.45	-383.0310	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	2018-10-11	2018-10-18	Standard Class	Sean O'D
10	CA-2017-115812	BH-11710	90032	FUR-TA-10001539	1706.1840	9	0.20	85.3092	Furniture	Tables	Chromcraft Rectangular Conference Tables	2017-06-09	2017-06-14	Standard Class	B Ho

‘Sales’ show the value > 100 and ‘Category’ is Furniture

SORTING DATA

SORTING

Sorting or ordering data based on certain columns.

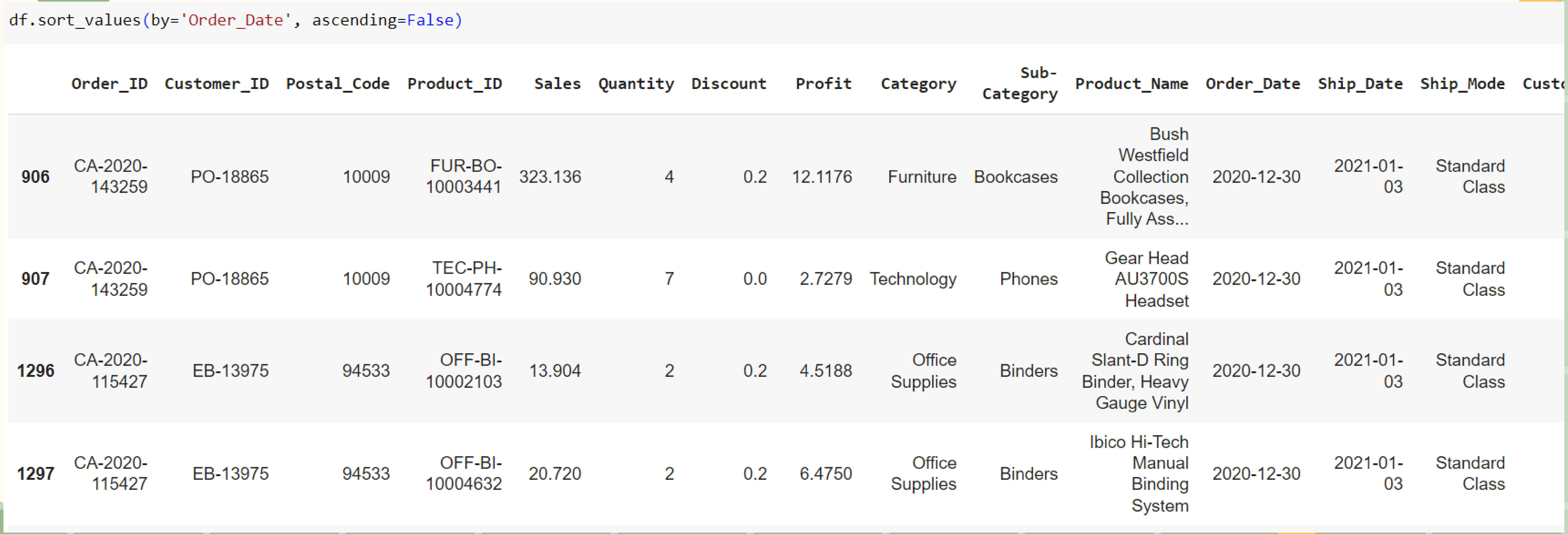
Ascending = True (default) means that data is sorted in ascending order, otherwise data is sorted in descending order.

If you want to sort more than 1 column and each column has a different sorting method, then the ascending parameter is filled with a list of boolean values for each column.

SORTING

df.sort_values (by = < name or column list > , ascending = True/False)

```
df.sort_values(by='Order_Date', ascending=False)
```



```
df.sort_values(by='Order_Date', ascending=False)
```

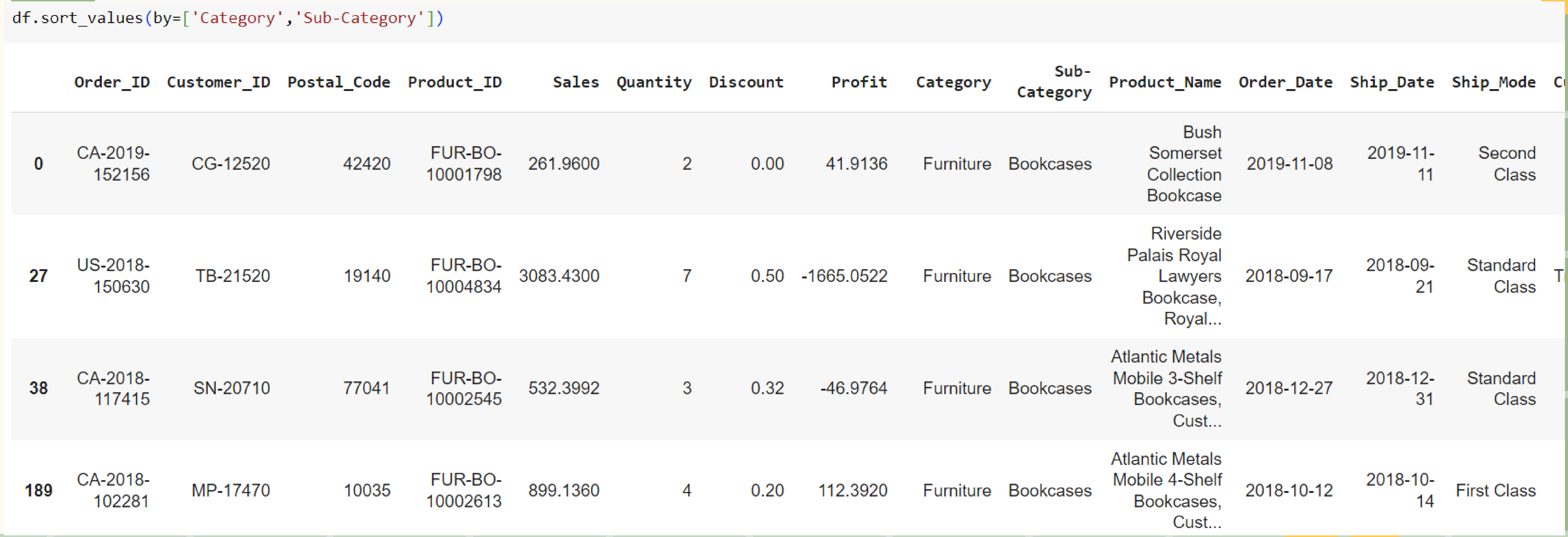
	Order_ID	Customer_ID	Postal_Code	Product_ID	Sales	Quantity	Discount	Profit	Category	Sub-Category	Product_Name	Order_Date	Ship_Date	Ship_Mode	Custo
906	CA-2020-143259	PO-18865	10009	FUR-BO-10003441	323.136	4	0.2	12.1176	Furniture	Bookcases	Bush Westfield Collection Bookcases, Fully Ass...	2020-12-30	2021-01-03	Standard Class	
907	CA-2020-143259	PO-18865	10009	TEC-PH-10004774	90.930	7	0.0	2.7279	Technology	Phones	Gear Head AU3700S Headset	2020-12-30	2021-01-03	Standard Class	
1296	CA-2020-115427	EB-13975	94533	OFF-BI-10002103	13.904	2	0.2	4.5188	Office Supplies	Binders	Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl	2020-12-30	2021-01-03	Standard Class	
1297	CA-2020-115427	EB-13975	94533	OFF-BI-10004632	20.720	2	0.2	6.4750	Office Supplies	Binders	Ibico Hi-Tech Manual Binding System	2020-12-30	2021-01-03	Standard Class	

SORTING 2 COLUMNS

df.sort_values (by = ['column 1', 'column 2'] , ascending = [True/False])

if it is not written it will automatically be ascending (True)

```
df.sort_values(by=['Category','Sub-Category'])
```

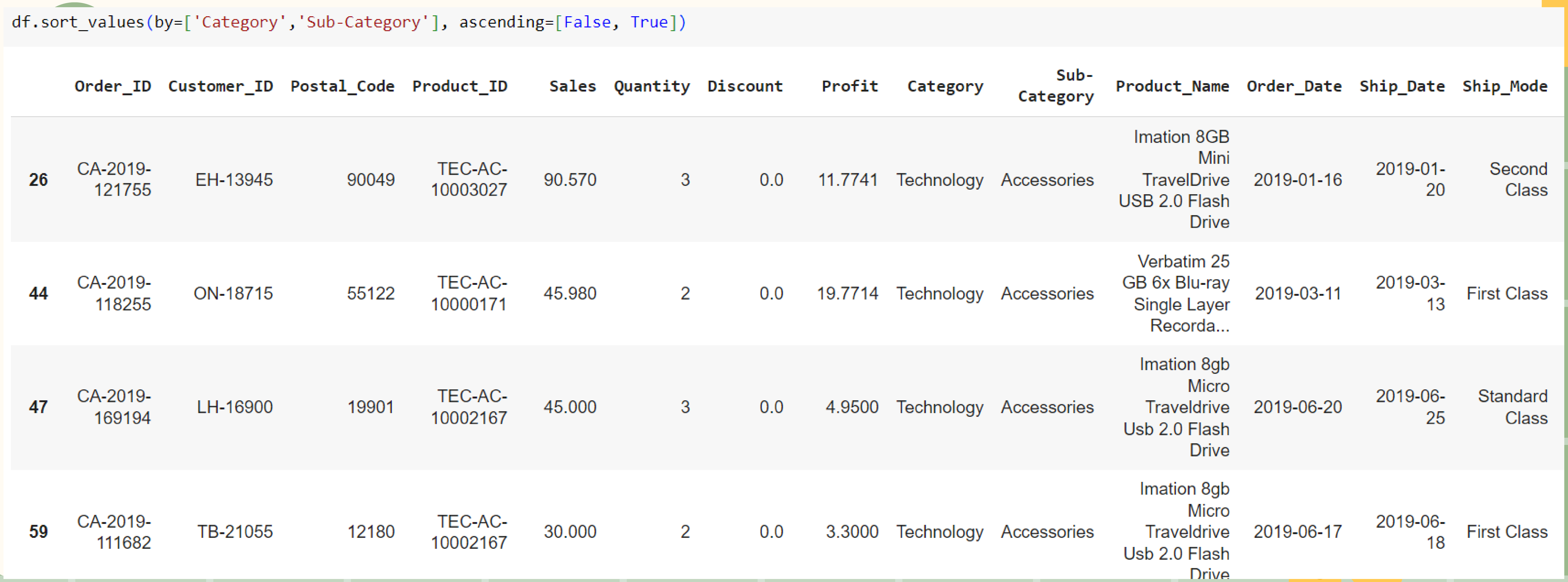


```
df.sort_values(by=['Category','Sub-Category'])
```

	Order_ID	Customer_ID	Postal_Code	Product_ID	Sales	Quantity	Discount	Profit	Category	Sub-Category	Product_Name	Order_Date	Ship_Date	Ship_Mode	C
0	CA-2019-152156	CG-12520	42420	FUR-BO-10001798	261.9600	2	0.00	41.9136	Furniture	Bookcases	Bush Somerset Collection Bookcase	2019-11-08	2019-11-11	Second Class	
27	US-2018-150630	TB-21520	19140	FUR-BO-10004834	3083.4300	7	0.50	-1665.0522	Furniture	Bookcases	Riverside Palais Royal Lawyers Bookcase, Royal...	2018-09-17	2018-09-21	Standard Class	T
38	CA-2018-117415	SN-20710	77041	FUR-BO-10002545	532.3992	3	0.32	-46.9764	Furniture	Bookcases	Atlantic Metals Mobile 3-Shelf Bookcases, Cust...	2018-12-27	2018-12-31	Standard Class	
189	CA-2018-102281	MP-17470	10035	FUR-BO-10002613	899.1360	4	0.20	112.3920	Furniture	Bookcases	Atlantic Metals Mobile 4-Shelf Bookcases, Cust...	2018-10-12	2018-10-14	First Class	

SORTING WITH 2 COLUMNS WITH DIFFERENT DIRECTION

```python
df.sort_values(by=['Category','Sub-Category'], ascending=[False, True])
```



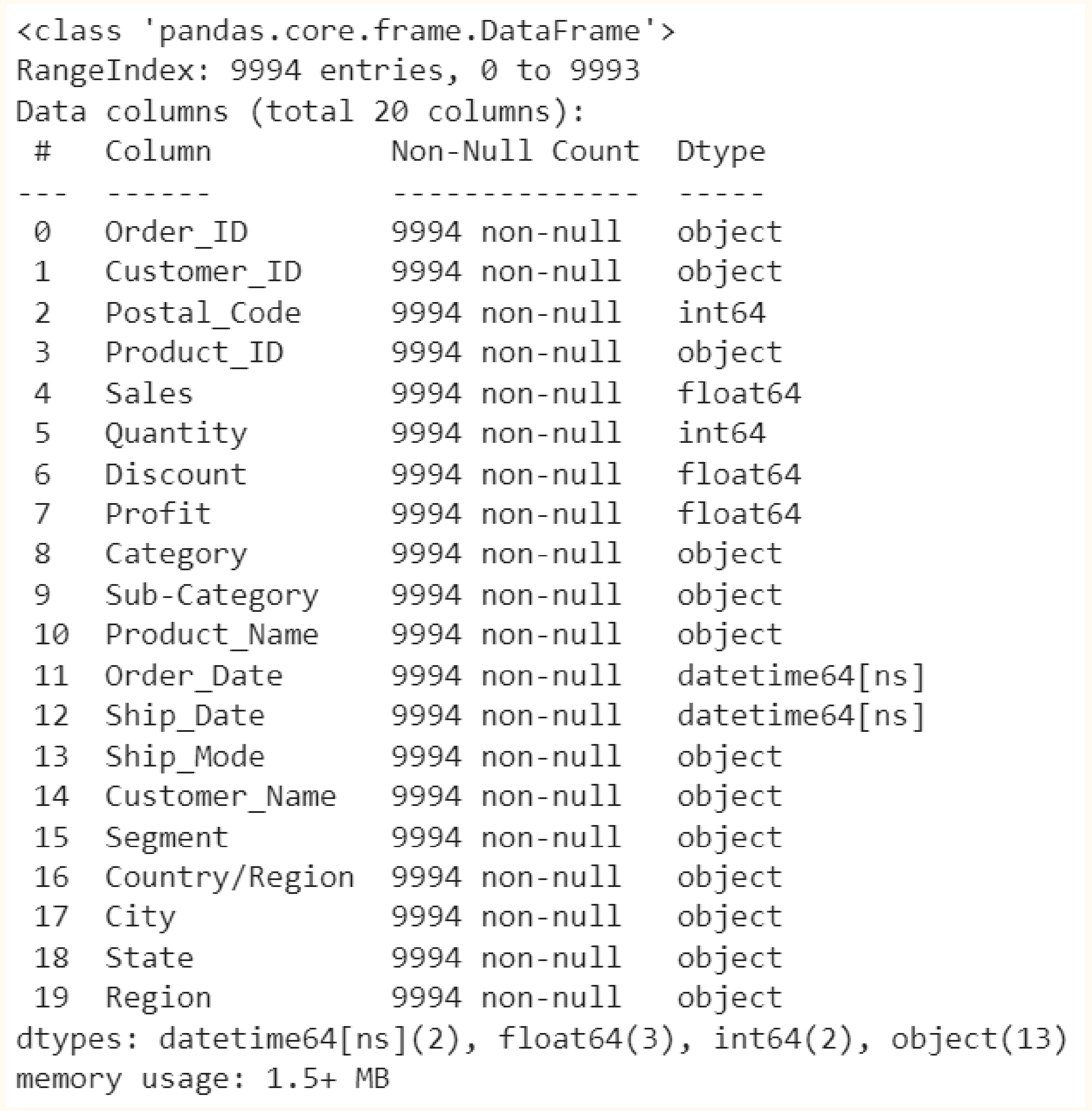
```python
df.sort_values(by=['Category','Sub-Category'], ascending=[False, True])
```

	Order_ID	Customer_ID	Postal_Code	Product_ID	Sales	Quantity	Discount	Profit	Category	Sub-Category	Product_Name	Order_Date	Ship_Date	Ship_Mode
26	CA-2019-121755	EH-13945	90049	TEC-AC-10003027	90.570	3	0.0	11.7741	Technology	Accessories	Imation 8GB Mini TravelDrive USB 2.0 Flash Drive	2019-01-16	2019-01-20	Second Class
44	CA-2019-118255	ON-18715	55122	TEC-AC-10000171	45.980	2	0.0	19.7714	Technology	Accessories	Verbatim 25 GB 6x Blu-ray Single Layer Recorda...	2019-03-11	2019-03-13	First Class
47	CA-2019-169194	LH-16900	19901	TEC-AC-10002167	45.000	3	0.0	4.9500	Technology	Accessories	Imation 8gb Micro Traveldrive Usb 2.0 Flash Drive	2019-06-20	2019-06-25	Standard Class
59	CA-2019-111682	TB-21055	12180	TEC-AC-10002167	30.000	2	0.0	3.3000	Technology	Accessories	Imation 8gb Micro Traveldrive Usb 2.0 Flash Drive	2019-06-17	2019-06-18	First Class

SUMMARIZING DATA

SUMMARIZE: DATASET INFO

df.info() contains some basic information from the dataset including column names and their data types `df.info()`

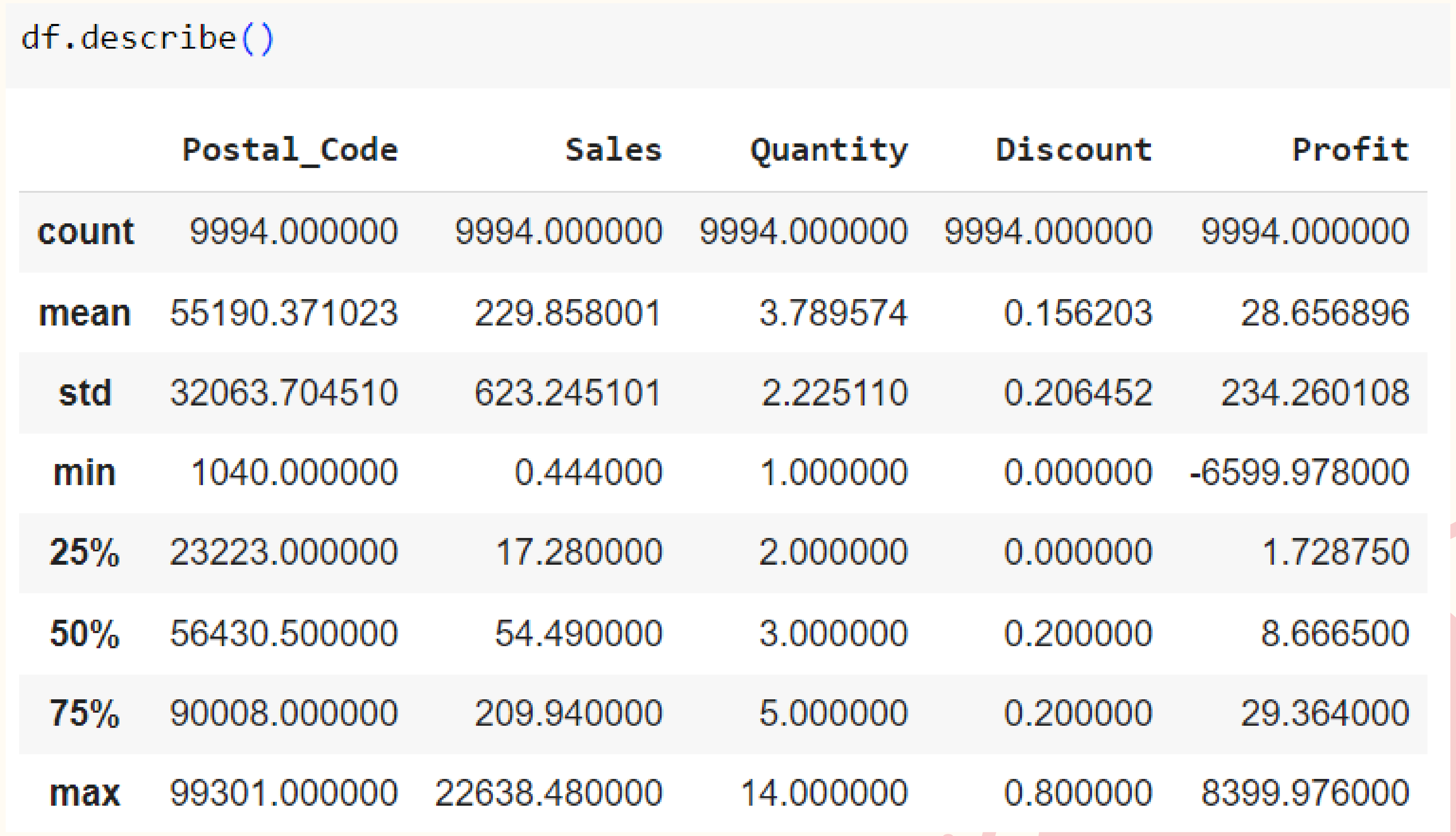


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Order_ID        9994 non-null   object
 1   Customer_ID     9994 non-null   object
 2   Postal_Code     9994 non-null   int64
 3   Product_ID      9994 non-null   object
 4   Sales           9994 non-null   float64
 5   Quantity        9994 non-null   int64
 6   Discount        9994 non-null   float64
 7   Profit          9994 non-null   float64
 8   Category        9994 non-null   object
 9   Sub-Category    9994 non-null   object
 10  Product_Name    9994 non-null   object
 11  Order_Date      9994 non-null   datetime64[ns]
 12  Ship_Date       9994 non-null   datetime64[ns]
 13  Ship_Mode       9994 non-null   object
 14  Customer_Name   9994 non-null   object
 15  Segment         9994 non-null   object
 16  Country/Region  9994 non-null   object
 17  City            9994 non-null   object
 18  State           9994 non-null   object
 19  Region          9994 non-null   object
dtypes: datetime64[ns](2), float64(3), int64(2), object(13)
memory usage: 1.5+ MB
```

SUMMARIZE: DESCRIPTIVE STATISTICS (NUMERIC)

To display descriptive statistics from data such as displaying count, mean, std deviation, min, max, 25%, 50%, 75% quartiles, you can use `df.describe()`

However, by default df.describe() will display descriptive statistics for numeric type columns

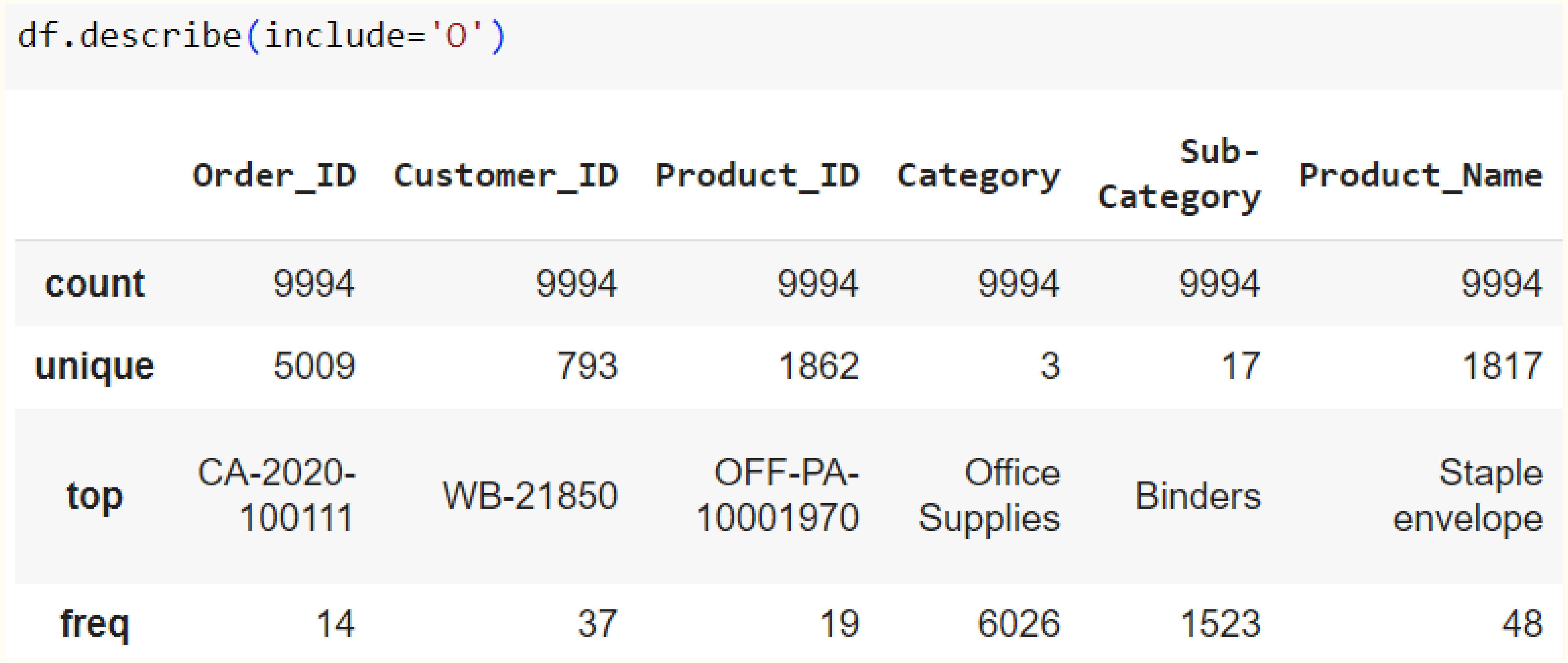


```
df.describe()
```

	Postal_Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	55190.371023	229.858001	3.789574	0.156203	28.656896
std	32063.704510	623.245101	2.225110	0.206452	234.260108
min	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	90008.000000	209.940000	5.000000	0.200000	29.364000
max	99301.000000	22638.480000	14.000000	0.800000	8399.976000

SUMMARIZE: DESCRIPTIVE STATISTICS (OBJECT)

To display descriptive statistics from non numeric columns use `df.describe( include = ‘o‘ )`



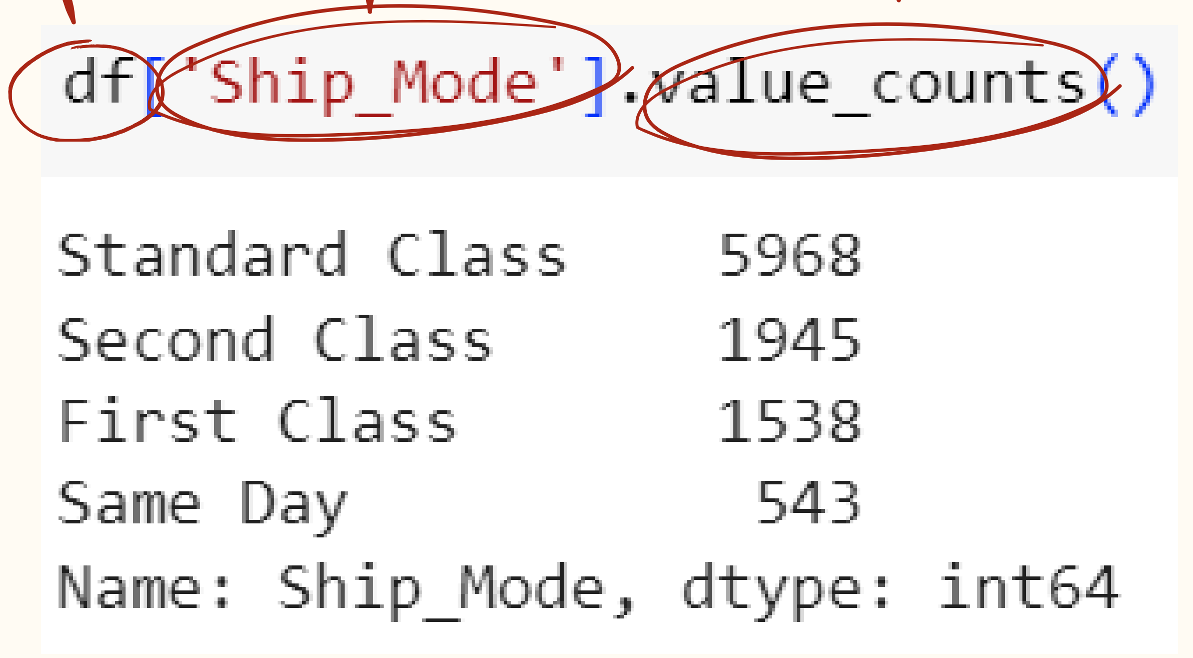
```
df.describe(include='O')
```

	Order_ID	Customer_ID	Product_ID	Category	Sub-Category	Product_Name
count	9994	9994	9994	9994	9994	9994
unique	5009	793	1862	3	17	1817
top	CA-2020-100111	WB-21850	OFF-PA-10001970	Office Supplies	Binders	Staple envelope
freq	14	37	19	6026	1523	48

SUMMARIZE: VALUE COUNTS

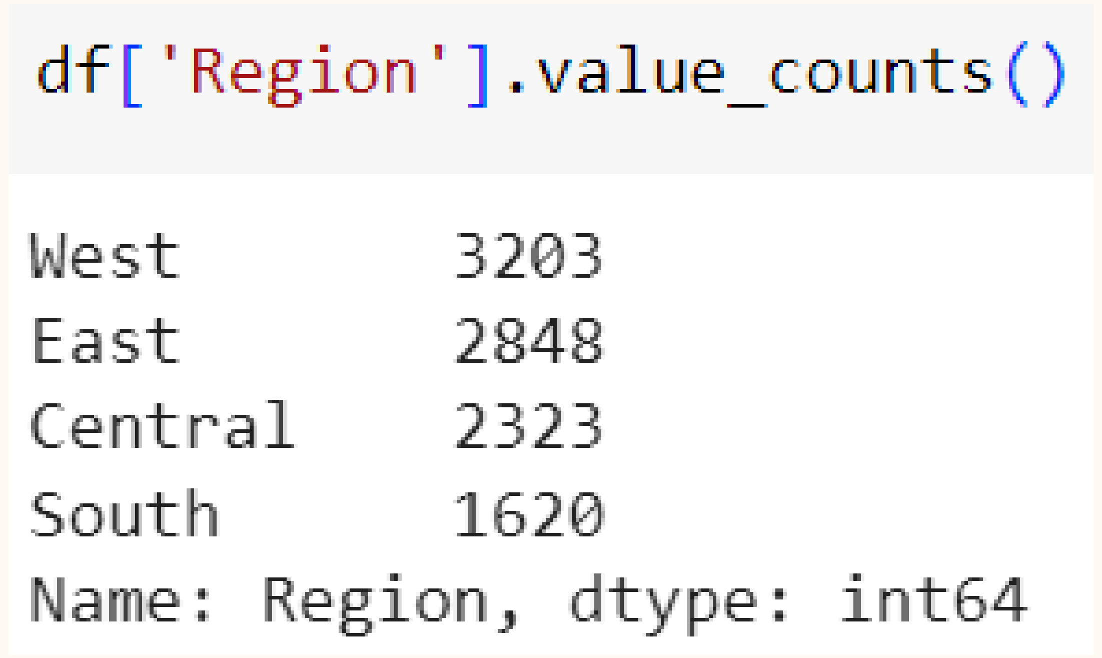
Value counts are used mainly in non-numeric columns to find out the number of values per item

```
df [ < column_name > ] . value_counts()
```



```
df['Ship_Mode'].value_counts()

Standard Class     5968
Second Class       1945
First Class        1538
Same Day            543
Name: Ship_Mode, dtype: int64
```



```
df['Region'].value_counts()

West       3203
East       2848
Central    2323
South      1620
Name: Region, dtype: int64
```

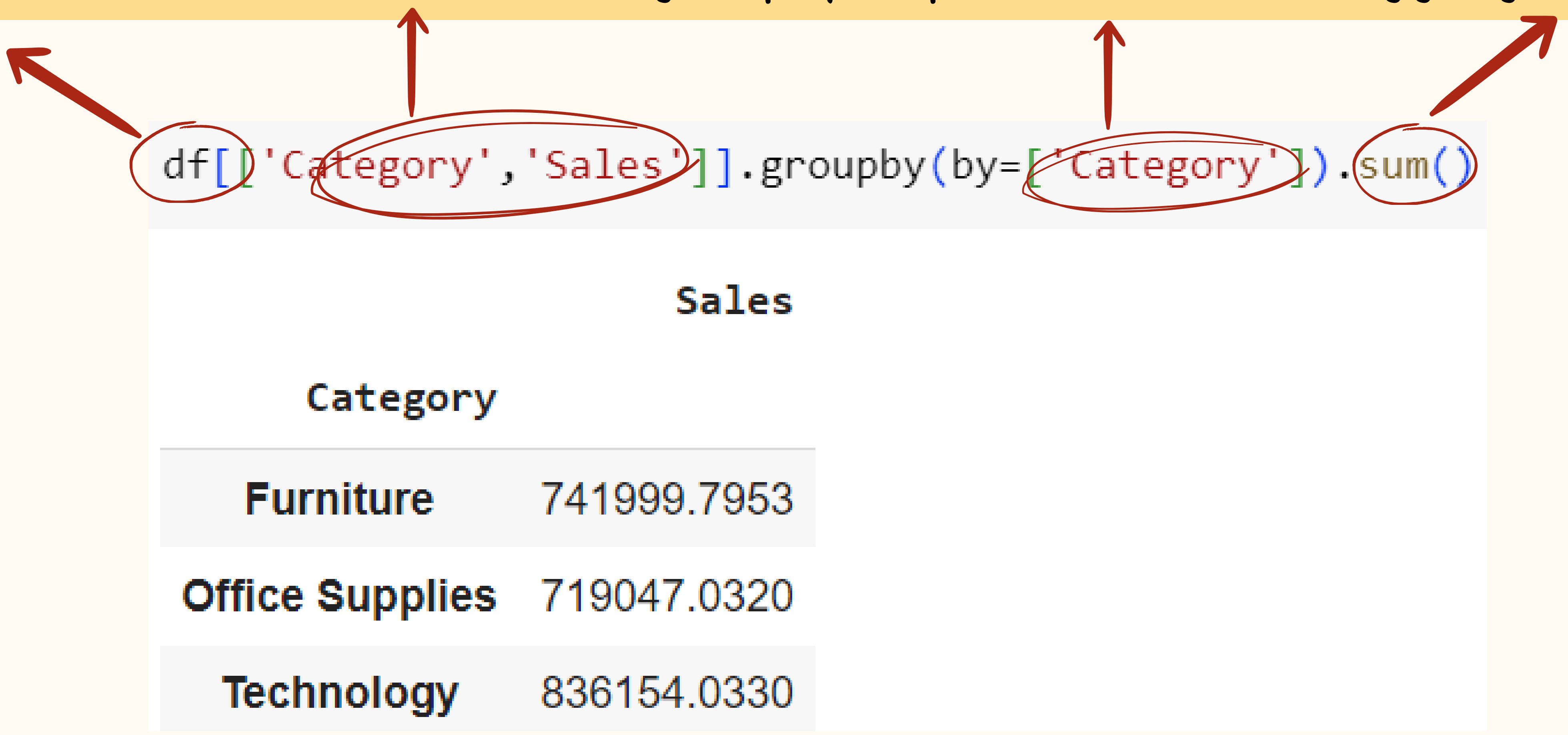
SUMMARIZE: GROUP BY

Group by is used to perform aggregation calculations per group in certain columns

Some aggregate functions include: sum(), min(), max(), mean(), etc

The column name in the groupby parameter is the column name that will be the group name

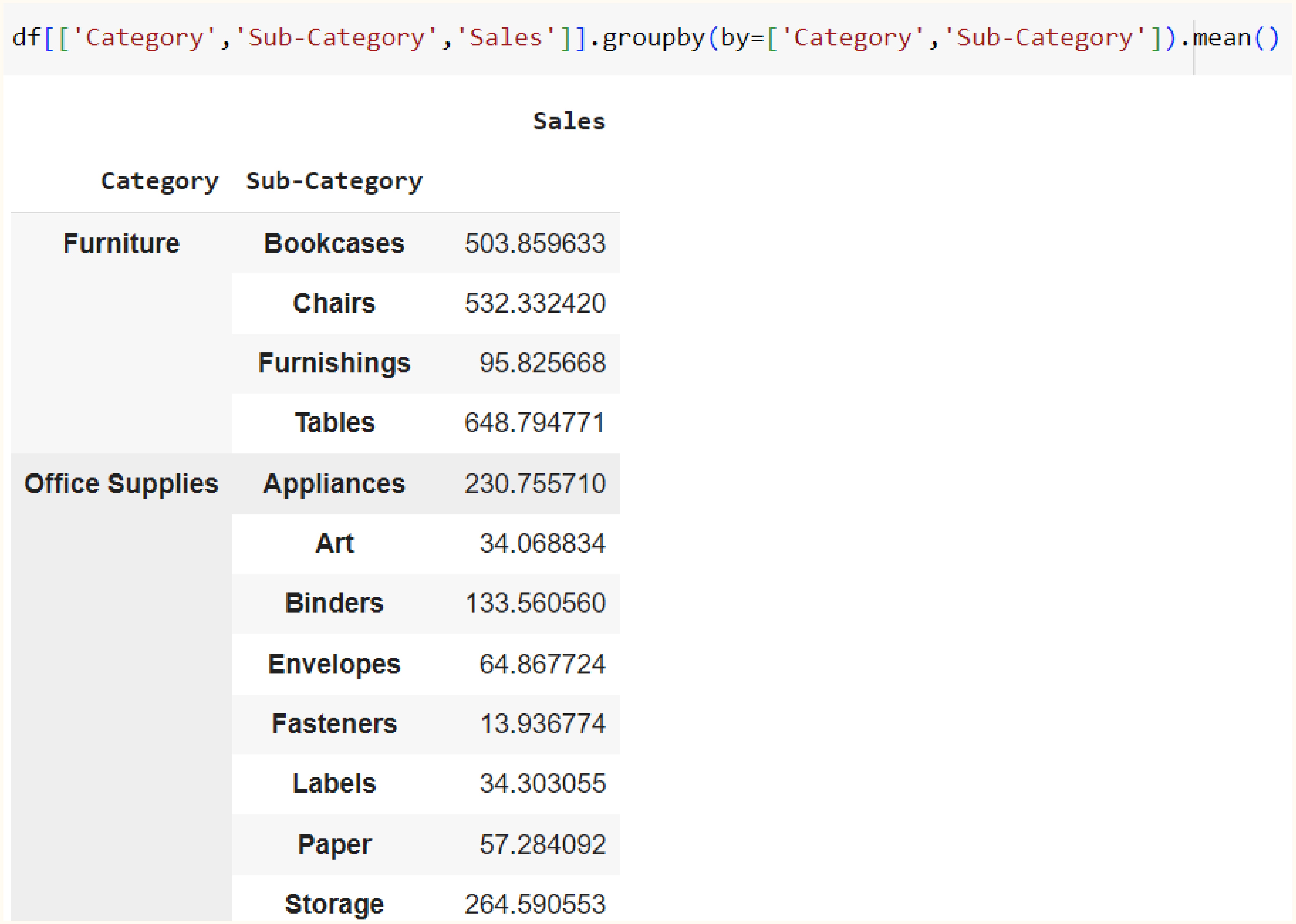
```
df [ < column or list column > ] . groupby ( [ parameter ] ) . < aggregate_function > ()
```



```
df[['Category','Sales']].groupby(by=['Category']).sum()
```

```
                     Sales
      Category
     Furniture  741999.7953
Office Supplies 719047.0320
    Technology  836154.0330
```

SUMMARIZE: MULTICOLUMN



```python
df[['Category','Sub-Category','Sales']].groupby(by=['Category','Sub-Category']).mean()
```

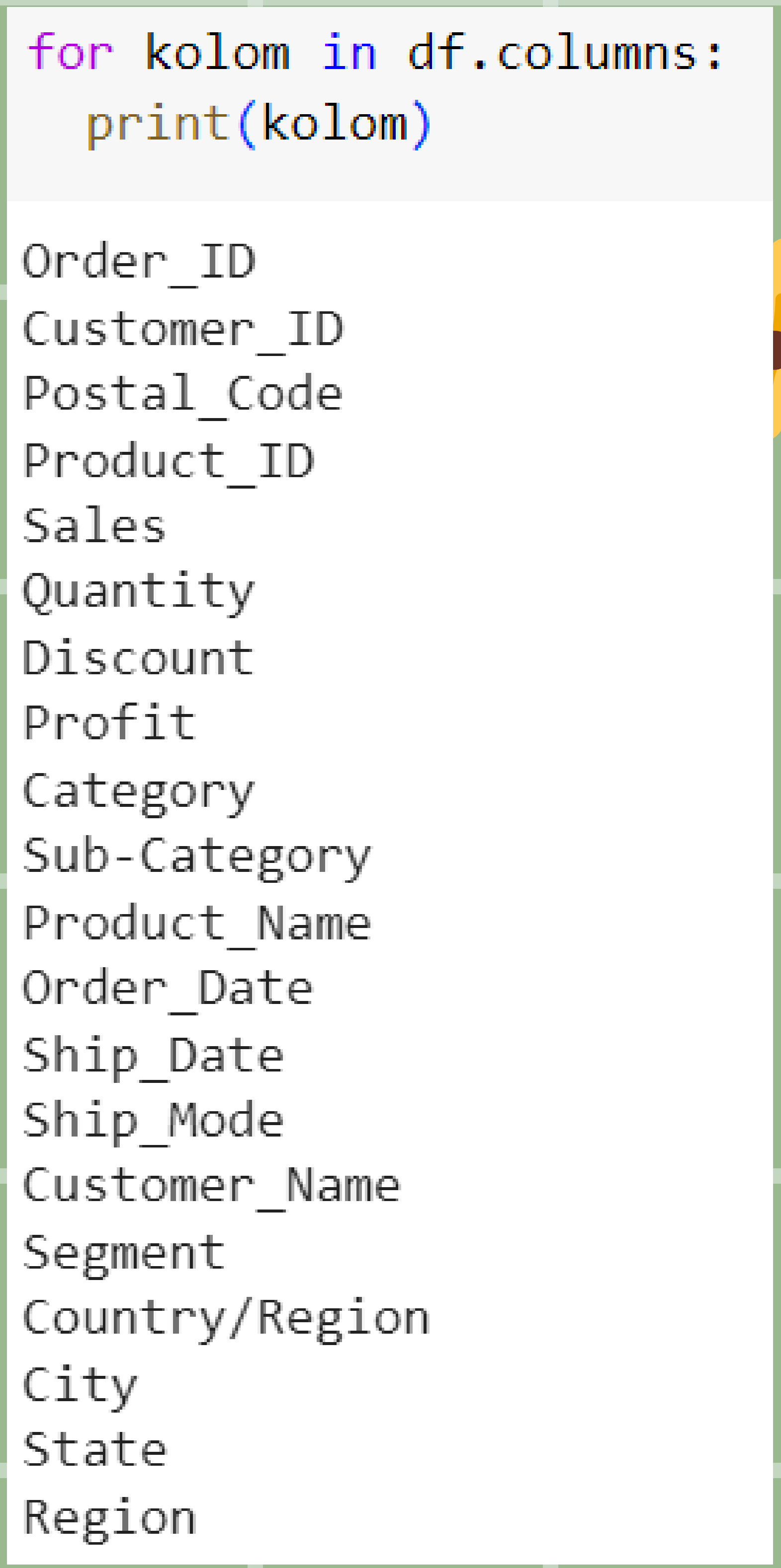
```
                                      Sales
Category         Sub-Category
Furniture        Bookcases       503.859633
                 Chairs          532.332420
                 Furnishings      95.825668
                 Tables          648.794771
Office Supplies  Appliances      230.755710
                 Art              34.068834
                 Binders         133.560560
                 Envelopes        64.867724
                 Fasteners        13.936774
                 Labels           34.303055
                 Paper            57.284092
                 Storage         264.590553
```

ITERATION DATA

ITERATION

Iteration using dataframes can be done in three ways: iterating with columns, indexes, and row by row

To iterate using columns use df.columns



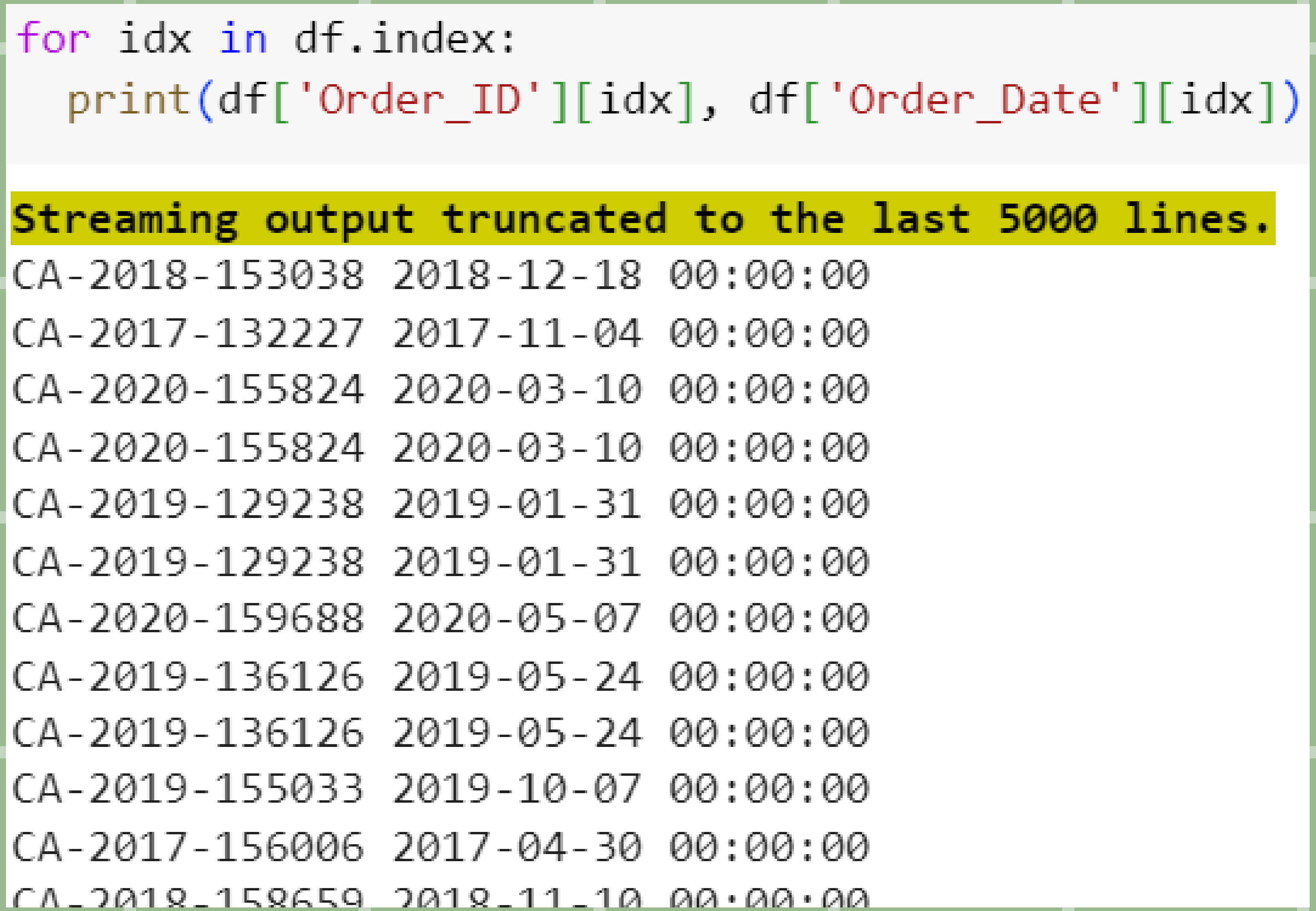
```
for kolom in df.columns:
  print(kolom)
```

```
Order_ID
Customer_ID
Postal_Code
Product_ID
Sales
Quantity
Discount
Profit
Category
Sub-Category
Product_Name
Order_Date
Ship_Date
Ship_Mode
Customer_Name
Segment
Country/Region
City
State
Region
```

ITERATION

To iterate using index use

df.index

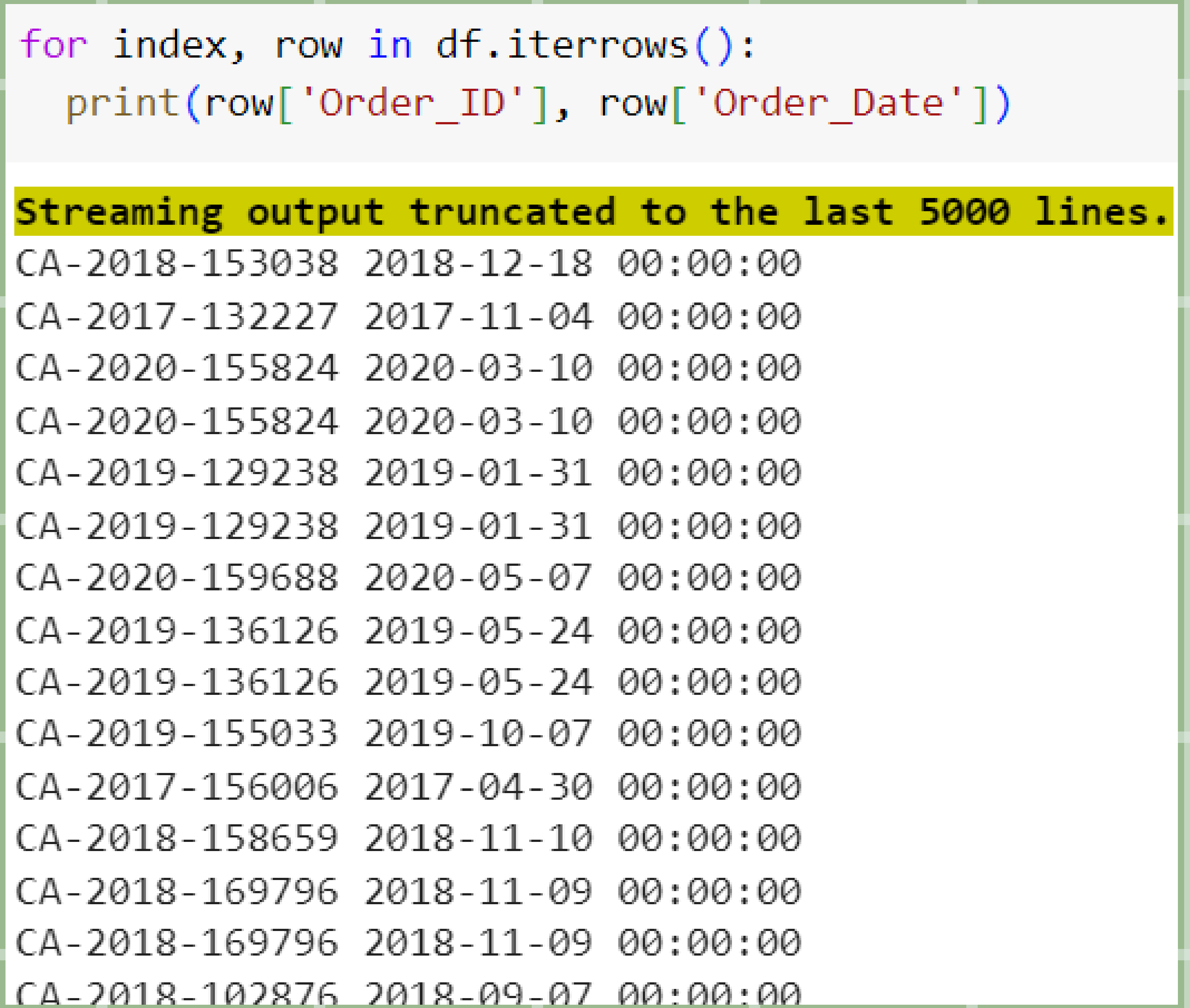


```
for idx in df.index:
  print(df['Order_ID'][idx], df['Order_Date'][idx])
```

```
Streaming output truncated to the last 5000 lines.
CA-2018-153038 2018-12-18 00:00:00
CA-2017-132227 2017-11-04 00:00:00
CA-2020-155824 2020-03-10 00:00:00
CA-2020-155824 2020-03-10 00:00:00
CA-2019-129238 2019-01-31 00:00:00
CA-2019-129238 2019-01-31 00:00:00
CA-2020-159688 2020-05-07 00:00:00
CA-2019-136126 2019-05-24 00:00:00
CA-2019-136126 2019-05-24 00:00:00
CA-2019-155033 2019-10-07 00:00:00
CA-2017-156006 2017-04-30 00:00:00
CA-2018-158659 2018-11-10 00:00:00
```

ITERATION

To iterate using row use **df.iterrows**



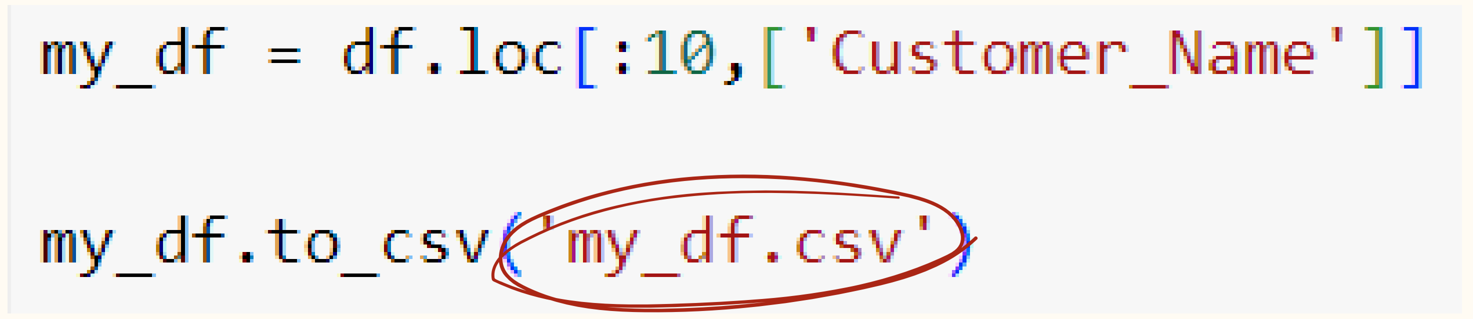
```
for index, row in df.iterrows():
  print(row['Order_ID'], row['Order_Date'])
```

```
Streaming output truncated to the last 5000 lines.
CA-2018-153038 2018-12-18 00:00:00
CA-2017-132227 2017-11-04 00:00:00
CA-2020-155824 2020-03-10 00:00:00
CA-2020-155824 2020-03-10 00:00:00
CA-2019-129238 2019-01-31 00:00:00
CA-2019-129238 2019-01-31 00:00:00
CA-2020-159688 2020-05-07 00:00:00
CA-2019-136126 2019-05-24 00:00:00
CA-2019-136126 2019-05-24 00:00:00
CA-2019-155033 2019-10-07 00:00:00
CA-2017-156006 2017-04-30 00:00:00
CA-2018-158659 2018-11-10 00:00:00
CA-2018-169796 2018-11-09 00:00:00
CA-2018-169796 2018-11-09 00:00:00
CA-2018-102876 2018-09-07 00:00:00
```

EXPORT DATA

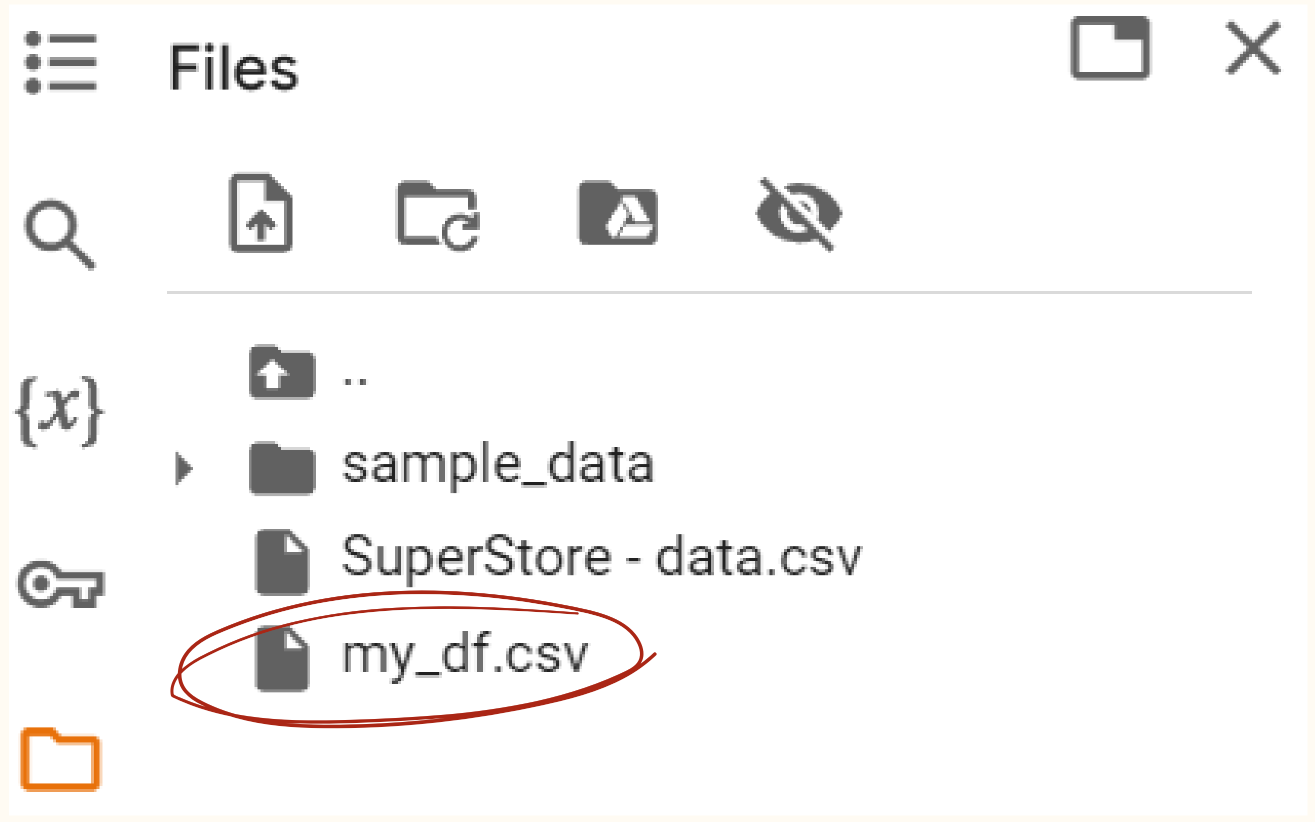
EXPORT DATA

For example, we create a dataframe containing consumer names in the first 10 rows and then save it into my_df. Then to convert my_df into a csv file use



```
my_df = df.loc[:10,['Customer_Name']]

my_df.to_csv('my_df.csv')
```



TERIMAKASIH

LinkedIn:

https://www.linkedin.com/in/anitamilaoktafani/