Pandas 3
Data Preprocessing and Calculation

Data analysis process

Data Understanding	•Descriptive statistics •Types of data (numerical/categorical)
Data Preprocessing	•Basic operations •Subset selection and data consolidation •Missing data handling
Calculation (Modeling)	•Basic calculation •Data aggregation
Data Visualization	•Univariate chart •Bivariate chart •Multivariate chart

Outline

- Missing data handling
  - Pandas missing values
  - Detect missing values
  - Drop missing values
  - Fill in missing values

- Calculation
  - Basic operations
  - Data aggregation

Missing data

- In applications, missing data may either be data that does not exist or that exists but not observed. Missing data also referred to as **NA** (Non Available).
- None is a Python object that is often used for missing data.

```
x = None
type(x)
```

```
NoneType
```

- Pandas uses NaN (Not a Number) to indicate missing data.

```
mySeries = pd.Series([1,2, None, 4])
mySeries
```

```
0    1.0
1    2.0
2    NaN
3    4.0
dtype: float64
```

```
type(mySeries[2])
```

```
numpy.float64
```

Missing data

- For various reasons, many real-world data sets may contain missing values.

```
fashion_df = pd.read_csv("../dataset/fashion.csv")
fashion_df.head(15)
```

	Date	Amanda_Christensen	Calvin_Klein	Eton	J_Lindeberg	Lacoste	Levi_s	Oscar_Jacobson	Ray_Ban	Tiger_of_Sweden
0	2014-07-01	5744.000000	29976.0000	78835.127273	89833.846154	65226.40000	63884.8	18971.813333	NaN	287420.566400
1	2014-08-01	7372.800000	33969.0000	98835.054545	153530.892308	43368.68000	57153.6	48796.800000	NaN	322481.827200
2	2014-09-01	8881.000000	28602.0000	70640.000000	146138.461538	26553.20000	47048.0	37864.266667	NaN	263211.054400
3	2014-10-01	10693.215000	23257.0000	70230.181818	151481.846154	37045.60000	33032.0	23762.000000	NaN	295135.536000
4	2014-11-01	17121.800000	29817.0000	96073.745455	180756.000000	35666.80000	25476.0	41173.600000	NaN	328531.016000
5	2014-12-01	34321.600000	43738.5000	143256.363636	144416.615385	33796.40000	25960.8	47829.466667	NaN	519894.808000
6	2015-01-01	7085.200000	28526.5000	120765.818182	182552.492308	31064.80000	55235.2	30837.733333	NaN	300741.560000
7	2015-02-01	7233.600000	27148.4658	82245.672727	134998.584492	29403.62392	30020.8	21506.095840	19107.20000	270403.424000
8	2015-03-01	7635.205680	48290.5000	81373.939491	189940.558031	62399.68420	302.4	66656.124907	67636.40000	260036.051003
9	2015-04-01	8710.552920	30567.9132	80840.245018	136470.398400	73999.01736	NaN	52975.765947	57623.21312	200100.461966
10	2015-05-01	17661.463395	39250.6262	116184.852655	179055.8					
11	2015-06-01	12468.388200	56983.3720	87637.331418	122439.3					
12	2015-07-01	6179.535600	42632.2670	135494.885091	153376.0					
13	2015-08-01	13285.157680	40665.5152	154798.000145	138074.2					
14	2015-09-01	12214.492200	53042.9611	90523.556436	233688.1					

Open csv file in Notepad++

```
Date,Amanda_Christensen,Calvin_Klein,Eton,J_Lindeberg,Lacoste,Levi_s,Oscar_Jacobson,Ray_Ban,Tiger_of_Sweden
2014-07-01,5744,29976,78835.1272727272,89833.8461538462,65226.4,63884.8,18971.8133333333,,287420.566400001
2014-08-01,7372.8,33969,98835.0545454545,153530.892307692,43368.6799999999,57153.6,48796.8,,322481.827200001
2014-09-01,8881,28602,70639.9999999999,146138.461538462,26553.1999999999,47048,37864.2666666667,,263211.0544
2014-10-01,10693.215,23257,70230.1818181817,151481.846153846,37045.5999999999,33032,23762,,295135.536000001
2014-11-01,17121.8,29817,96073.7454545454,180756.000000001,35666.8,25476,41173.6,,328531.016000001
2014-12-01,34321.5999999999,43738.5,143256.363636364,144416.615384615,33796.4,25960.8,47829.4666666667,,519894.808000004
2015-01-01,7085.2,28526.5,120765.818181818,182552.492307693,31064.7999999999,55235.2,30837.7333333333,,300741.560000001
2015-02-01,7233.6,27148.4658,82245.6727272726,134998.584492308,29403.62392,30020.8,21506.09584,19107.2,270403.424
```

https://pandas.pydata.org/pandas-docs/dev/user_guide/missing_data.html

Non-null count

- Use `info()` to see the number of non-missing values in each columns

```
fashion_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Date                51 non-null     object
 1   Amanda_Christensen  51 non-null     float64
 2   Calvin_Klein        51 non-null     float64
 3   Eton                51 non-null     float64
 4   J_Lindeberg         51 non-null     float64
 5   Lacoste             51 non-null     float64
 6   Levi_s              22 non-null     float64
 7   Oscar_Jacobson      51 non-null     float64
 8   Ray_Ban             44 non-null     float64
 9   Tiger_of_Sweden     51 non-null     float64
dtypes: float64(9), object(1)
memory usage: 4.1+ KB
```

Handling missing data

- A list of methods for missing data.

Method	Description
`isna`	Return Boolean values indicating which values are missing.
`notna`	Negation of isna.
`dropna`	Filter out missing data.
`fillna`	Fill in missing data with some value or using an interpolation method such as ‘ffill’ or ‘bfill’

isnull

- Use `isna()` to detect if the value is missing.

	winery	country	price	province
0	Domaine Vincent Girardin	France	94.0	Burgundy
1	None	None	NaN	None
2	Au Pied du Mont Chauve	France	43.0	Burgundy
3	Chateau Raymond-Lafon	France	120.0	Bordeaux
4	Chateau Balac	France	45.0	Bordeaux
5	Etienne de Tauriac	France	25.0	Bordeaux
6	Borgogno	Italy	80.0	Piedmont
7	Sottimano	Italy	58.0	Piedmont
8	Punset	None	43.0	None
9	La Fiorita	Italy	60.0	Tuscany
10	Piccini	Italy	NaN	Tuscany
11	Pietranera	Italy	75.0	Tuscany

```
wine_df1.isna()
```

	winery	country	price	province
0	False	False	False	False
1	True	True	True	True
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
5	False	False	False	False
6	False	False	False	False
7	False	False	False	False
8	False	True	False	True
9	False	False	False	False
10	False	False	True	False
11	False	False	False	False

True = 1
False = 0

```
wine_df1.isna().sum()
```

```
winery      1
country     2
price       2
province    2
dtype: int64
```

The number of missing values for each column.

By default, the method `sum()` adds across columns (axis =0).

```
wine_df1.isna().sum(axis = 1)
```

```
0     0
1     4
2     0
3     0
4     0
5     0
6     0
7     0
8     2
9     0
10    1
11    0
dtype: int64
```

The number of missing values for each row.

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.isnull.html

isnull

- Use `isna()` to filter data.

	winery	country	price	province
0	Domaine Vincent Girardin	France	94.0	Burgundy
1	None	None	NaN	None
2	Au Pied du Mont Chauve	France	43.0	Burgundy
3	Chateau Raymond-Lafon	France	120.0	Bordeaux
4	Chateau Balac	France	45.0	Bordeaux
5	Etienne de Tauriac	France	25.0	Bordeaux
6	Borgogno	Italy	80.0	Piedmont
7	Sottimano	Italy	58.0	Piedmont
8	Punset	None	43.0	None
9	La Fiorita	Italy	60.0	Tuscany
10	Piccini	Italy	NaN	Tuscany
11	Pietranera	Italy	75.0	Tuscany

```
wine_df1.price.isna()
```

```
0     False
1      True
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10     True
11    False
Name: price, dtype: bool
```

Boolean array

```
wine_df1[wine_df1.price.isna()]
```

	winery	country	price	province
1	None	None	NaN	None
10	Piccini	Italy	NaN	Tuscany

dropna – drop rows

- Use `dropna(how="any")` to remove the rows containing any NaNs.

	winery	country	price	province
0	Domaine Vincent Girardin	France	94.0	Burgundy
1	None	None	NaN	None ✘
2	Au Pied du Mont Chauve	France	43.0	Burgundy
3	Chateau Raymond-Lafon	France	120.0	Bordeaux
4	Chateau Balac	France	45.0	Bordeaux
5	Etienne de Tauriac	France	25.0	Bordeaux
6	Borgogno	Italy	80.0	Piedmont
7	Sottimano	Italy	58.0	Piedmont
8	Punset	None	43.0	None ✘
9	La Fiorita	Italy	60.0	Tuscany
10	Piccini	Italy	NaN	Tuscany ✘
11	Pietranera	Italy	75.0	Tuscany

12 rows

```
wine_df1.dropna()
```

By default, how="any"

	winery	country	price	province
0	Domaine Vincent Girardin	France	94.0	Burgundy
2	Au Pied du Mont Chauve	France	43.0	Burgundy
3	Chateau Raymond-Lafon	France	120.0	Bordeaux
4	Chateau Balac	France	45.0	Bordeaux
5	Etienne de Tauriac	France	25.0	Bordeaux
6	Borgogno	Italy	80.0	Piedmont
7	Sottimano	Italy	58.0	Piedmont
9	La Fiorita	Italy	60.0	Tuscany
11	Pietranera	Italy	75.0	Tuscany

9 rows

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html

dropna – drop rows

- Use `dropna(how="all")` to remove the rows that are all NaN.

	winery	country	price	province
0	Domaine Vincent Girardin	France	94.0	Burgundy
1	None	None	NaN	None
2	Au Pied du Mont Chauve	France	43.0	Burgundy
3	Chateau Raymond-Lafon	France	120.0	Bordeaux
4	Chateau Balac	France	45.0	Bordeaux
5	Etienne de Tauriac	France	25.0	Bordeaux
6	Borgogno	Italy	80.0	Piedmont
7	Sottimano	Italy	58.0	Piedmont
8	Punset	None	43.0	None
9	La Fiorita	Italy	60.0	Tuscany
10	Piccini	Italy	NaN	Tuscany
11	Pietranera	Italy	75.0	Tuscany

12 rows

```
wine_df1.dropna(how = "all")
```

	winery	country	price	province
0	Domaine Vincent Girardin	France	94.0	Burgundy
2	Au Pied du Mont Chauve	France	43.0	Burgundy
3	Chateau Raymond-Lafon	France	120.0	Bordeaux
4	Chateau Balac	France	45.0	Bordeaux
5	Etienne de Tauriac	France	25.0	Bordeaux
6	Borgogno	Italy	80.0	Piedmont
7	Sottimano	Italy	58.0	Piedmont
8	Punset	None	43.0	None
9	La Fiorita	Italy	60.0	Tuscany
10	Piccini	Italy	NaN	Tuscany
11	Pietranera	Italy	75.0	Tuscany

11 rows

dropna – drop rows

- Define in which columns to look for missing values.

	winery	country	price	province
0	Domaine Vincent Girardin	France	94.0	Burgundy
1	None	None	NaN	None ✘
2	Au Pied du Mont Chauve	France	43.0	Burgundy
3	Chateau Raymond-Lafon	France	120.0	Bordeaux
4	Chateau Balac	France	45.0	Bordeaux
5	Etienne de Tauriac	France	25.0	Bordeaux
6	Borgogno	Italy	80.0	Piedmont
7	Sottimano	Italy	58.0	Piedmont
8	Punset	None	43.0	None ✘
9	La Fiorita	Italy	60.0	Tuscany
10	Piccini	Italy	NaN	Tuscany
11	Pietranera	Italy	75.0	Tuscany

12 rows

```
wine_df1.dropna(subset = ['country', 'province'], how = "all")
```

	winery	country	price	province
0	Domaine Vincent Girardin	France	94.0	Burgundy
2	Au Pied du Mont Chauve	France	43.0	Burgundy
3	Chateau Raymond-Lafon	France	120.0	Bordeaux
4	Chateau Balac	France	45.0	Bordeaux
5	Etienne de Tauriac	France	25.0	Bordeaux
6	Borgogno	Italy	80.0	Piedmont
7	Sottimano	Italy	58.0	Piedmont
9	La Fiorita	Italy	60.0	Tuscany
10	Piccini	Italy	NaN	Tuscany
11	Pietranera	Italy	75.0	Tuscany

10rows

dropna – drop columns

- To drop the columns in the same way, pass `axis=1`.

5 columns

	winery	country	price	province	description
0	Domaine Vincent Girardin	France	94	Burgundy	None
1	Domaine de Bellene	France	60	Burgundy	None
2	Au Pied du Mont Chauve	France	43	Burgundy	None
3	Chateau Raymond-Lafon	None	120	Bordeaux	None
4	Chateau Balac	France	45	Bordeaux	None
5	Etienne de Tauriac	France	25	Bordeaux	None
6	Borgogno	Italy	80	Piedmont	None
7	Sottimano	Italy	58	Piedmont	None
8	Punset	Italy	43	Piedmont	None
9	La Fiorita	Italy	60	Tuscany	None
10	Piccini	Italy	60	Tuscany	None
11	Pietranera	Italy	75	Tuscany	None

✘ ✘

```
wine_df2.dropna(axis = 1)
```

	winery	price	province
0	Domaine Vincent Girardin	94	Burgundy
1	Domaine de Bellene	60	Burgundy
2	Au Pied du Mont Chauve	43	Burgundy
3	Chateau Raymond-Lafon	120	Bordeaux
4	Chateau Balac	45	Bordeaux
5	Etienne de Tauriac	25	Bordeaux
6	Borgogno	80	Piedmont
7	Sottimano	58	Piedmont
8	Punset	43	Piedmont
9	La Fiorita	60	Tuscany
10	Piccini	60	Tuscany
11	Pietranera	75	Tuscany

3 columns

fillna – constant value

- Replace missing values with a constant value.

```
sales_df
```

	Month	Sales
0	Jan	266.0
1	Feb	145.9
2	Mar	183.1
3	Apr	NaN
4	May	180.3
5	Jun	168.5
6	Jul	231.8
7	Aug	NaN
8	Sep	192.8
9	Oct	122.9
10	Nov	203.4
11	Dec	211.7

```
sales_df.fillna(0)
```

	Month	Sales
0	Jan	266.0
1	Feb	145.9
2	Mar	183.1
3	Apr	0.0
4	May	180.3
5	Jun	168.5
6	Jul	231.8
7	Aug	0.0
8	Sep	192.8
9	Oct	122.9
10	Nov	203.4
11	Dec	211.7

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html

fillna – forward

- Replace missing values with the last valid observation.

```
sales_df
```

	Month	Sales
0	Jan	266.0
1	Feb	145.9
2	Mar	183.1
3	Apr	NaN
4	May	180.3
5	Jun	168.5
6	Jul	231.8
7	Aug	NaN
8	Sep	192.8
9	Oct	122.9
10	Nov	203.4
11	Dec	211.7

```
sales_df.fillna(method = "ffill")
```

	Month	Sales
0	Jan	266.0
1	Feb	145.9
2	Mar	183.1
3	Apr	183.1
4	May	180.3
5	Jun	168.5
6	Jul	231.8
7	Aug	231.8
8	Sep	192.8
9	Oct	122.9
10	Nov	203.4
11	Dec	211.7

fillna – backward

- Replace missing values with the next valid observation.

```
sales_df
```

	Month	Sales
0	Jan	266.0
1	Feb	145.9
2	Mar	183.1
3	Apr	NaN
4	May	180.3
5	Jun	168.5
6	Jul	231.8
7	Aug	NaN
8	Sep	192.8
9	Oct	122.9
10	Nov	203.4
11	Dec	211.7

```
sales_df.fillna(method = "bfill")
```

	Month	Sales
0	Jan	266.0
1	Feb	145.9
2	Mar	183.1
3	Apr	180.3
4	May	180.3
5	Jun	168.5
6	Jul	231.8
7	Aug	192.8
8	Sep	192.8
9	Oct	122.9
10	Nov	203.4
11	Dec	211.7

Exercise

(A.1) Import the dataset `melbourne.csv`. Calculate the number of rows and columns.

(A.2) Delete the row containing any NaN. Calculate the number of rows and columns.

(A.3) Use the dataframe in (A.1). Calculate the number of missing value in each columns.

(A.4) What is the average number of parking space (column `Car`)?

(A.5) Fill the missing values in the `Car` column with 0 and apply this change directly to the data frame. What is the average number of parking space?

Calculation

Calculation – Series

- Calculate the statistics of a series.

```
s1 = pd.Series([1,2,3,4,5])
print(s1.mean())
print(s1.sum())
print(s1.max())
print(s1.count())
```

```
3.0
15
5
5
```

```
s2 = pd.Series([1,2,3,4,None])
print(s2.mean())
print(s2.sum())
print(s2.max())
print(s2.count())
```

```
2.5
10.0
4.0
4
```

(1+2+3+4)/4 = 2.5

Calculation – DataFrame

- Calculate the sum of all the values over the column axis

	grocery	transportation	dining out	entertainment
Jan	3050	1050	1200	1250
Feb	2800	900	1950	1050
Mar	2750	1150	1350	2500
Apr	2300	1850	3250	3150
May	3150	1250	1050	2000
Jun	2900	950	2800	1050

```
df_expense["total_expense"] = df_expense.sum(axis = 1)
df_expense
```

	grocery	transportation	dining out	entertainment	total_expense
Jan	3050	1050	1200	1250	6550
Feb	2800	900	1950	1050	6700
Mar	2750	1150	1350	2500	7750
Apr	2300	1850	3250	3150	10550
May	3150	1250	1050	2000	7450
Jun	2900	950	2800	1050	7700

Calculation – DataFrame

- Calculate the sum of the values of two columns

```
df_expense['necessary_expense']  = df_expense.grocery + df_expense.transportation
df_expense
```

	grocery	transportation	dining out	entertainment	total_expense	necessary_expense
Jan	3050	1050	1200	1250	6550	4100
Feb	2800	900	1950	1050	6700	3700
Mar	2750	1150	1350	2500	7750	3900
Apr	2300	1850	3250	3150	10550	4150
May	3150	1250	1050	2000	7450	4400
Jun	2900	950	2800	1050	7700	3850

Calculation – DataFrame

- Calculate the percentage

```
df_expense['necessary_expense(%)'] =  round((df_expense.necessary_expense/ df_expense.total_expense)*100,2)
df_expense
```

	grocery	transportation	dining out	entertainment	total_expense	necessary_expense	necessary_expense(%)
Jan	3050	1050	1200	1250	6550	4100	62.60
Feb	2800	900	1950	1050	6700	3700	55.22
Mar	2750	1150	1350	2500	7750	3900	50.32
Apr	2300	1850	3250	3150	10550	4150	39.34
May	3150	1250	1050	2000	7450	4400	59.06
Jun	2900	950	2800	1050	7700	3850	50.00

$$\frac{4100}{6550} \times 100 = 62.60$$

Calculation – convert numerical data to categorical data

- Use `cut()` to segment and sort continuous data into bins.

points	grade
75-100	A
65-74.99	B
55-64.99	C
45-54.99	D
35-44.99	E
0-34.99	F

bin edges for the segmentation

```
score_df["grade"] = pd.cut(score_df.score, bins = [0,34,44,54,64,74,100],
                           labels = ["F","E","D","C","B","A"],
                           include_lowest= True)
score_df
```

	score
0	75
1	60
2	40
3	100
4	85
5	55
6	65
7	20
8	70
9	0

	score	grade
0	75	A
1	60	C
2	40	E
3	100	A
4	85	A
5	55	C
6	65	B
7	20	F
8	70	B
9	0	F

By default, bins include the rightmost edge (`right = True`)

https://pandas.pydata.org/docs/reference/api/pandas.cut.html

Exercise

	Q1	Q2	Q3	Q4
A	124	132	150	128
B	148	131	142	138
C	126	125	157	128
D	102	150	133	126
E	116	119	152	159

(B.1) Given the synthetic dataset above, each column represents quarterly sales, and the index is the product name. Caculate the annual sales of each product.

(B.2) Calculate the average quarterly sales for each product.

(B.3) For each product, calculate the proportion of sales in the first half of the year to annual sales. Round the number to the two decimal place.

(B.4) For each product, calculate the percentage increase in sales in the second quarter compared to the first quarter. Round the number to the two decimal place.

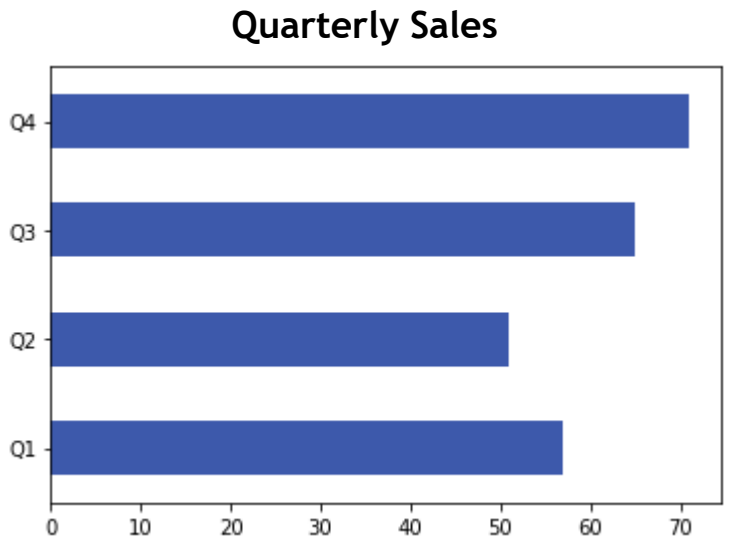
Data aggregation

- In some cases, you may need to compute group statistics for reporting or visualization purposes.

Quarter	Month	Sales
Q1	1	20
Q1	2	16
Q1	3	21
Q2	4	15
Q2	5	14
Q2	6	22
Q3	7	27
Q3	8	15
Q3	9	23
Q4	10	25
Q4	11	22
Q4	12	24

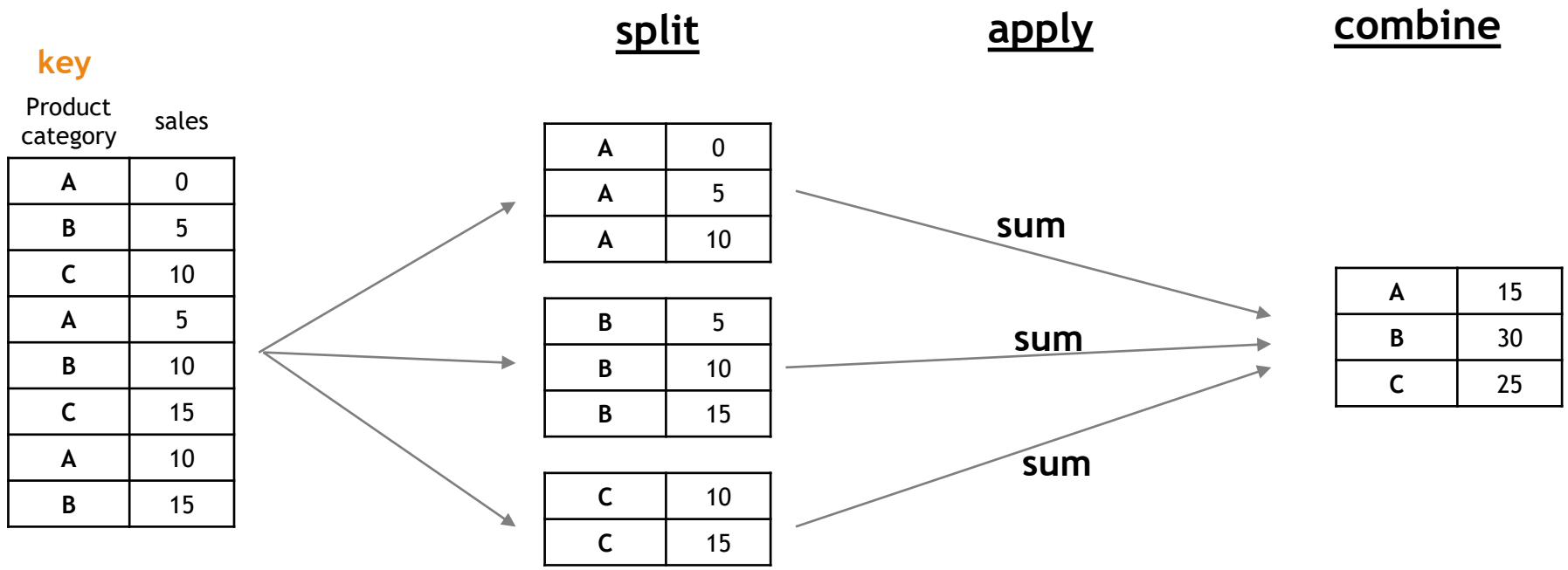
20+16+21 = 57

Quarter	Sales
Q1	57
Q2	51
Q3	65
Q4	71



Group by: split-apply-combine

- By “group by” we are referring to a process involving one or more of the following steps:
  - Splitting the data into groups based on key(s).
  - Applying a function to each group independently.
  - Combining the results into a data structure.



GroupBy object

- Use `groupby()` to group the data according to the keys and apply a function to the groups.
- The method `groupby()` returns a `GroupBy` object.

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

```
sales_gb_product = sales_df.groupby("Product")
type(sales_gb_product)
```

```
pandas.core.groupby.generic.DataFrameGroupBy
```

```
sales_gb_product.groups
```

```
{'A': [0, 1, 2, 3, 4, 5], 'B': [6, 7, 8, 9, 10, 11]}
```

https://pandas.pydata.org/docs/reference/groupby.html

GroupBy object - statistics

- The **GroupBy object** has all of the information needed to then apply some operation to each of the groups.

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

```
sales_gb_product.size()
```

```
Product
A    6
B    6
dtype: int64
```

```
sales_gb_product.mean()
```

Product	Sales
A	71.0
B	92.5

Calculate the average of a numeric column

GroupBy object - statistics

- Some methods can be applied to numerical columns and categorical columns.

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

```
sales_gb_product.max()
```

Product	Quarter	Month	Sales
A	Q2	May	97
B	Q2	May	113

For categorical columns, max() sorts the values alphabetically and return the last one.

```
sales_gb_product.Sales.max()
```

```
Product
A     97
B    113
Name: Sales, dtype: int64
```

Select the numerical column and calculate the statistics.

GroupBy object - statistics

- Use `agg()` to aggregate more operations.

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

```
sales_gb_product.Sales.agg(['mean','min','max'])
```

Product	mean	min	max
A	71.0	50	97
B	92.5	78	113

GroupBy object – two keys

- Grouping the data according to a list of keys.

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

```
sales_gb_product_Q = sales_df.groupby(['Product','Quarter'])
```

```
sales_gb_product_Q.groups
```

```
{('A', 'Q1'): [0, 1, 2], ('A', 'Q2'): [3, 4, 5], ('B', 'Q1'): [6, 7, 8], ('B', 'Q2'): [9, 10, 11]}
```

2 products × 2 quarters = 4 groups

GroupBy – two keys

- Statistics

	Product	Quarter	Month	Sales
0	A	Q1	Jan	67
1	A	Q1	Feb	57
2	A	Q1	Mar	87
3	A	Q2	Apr	50
4	A	Q2	May	97
5	A	Q2	Jun	68
6	B	Q1	Jan	78
7	B	Q1	Feb	102
8	B	Q1	Mar	113
9	B	Q2	Apr	98
10	B	Q2	May	80
11	B	Q2	Jun	84

```
sales_gb_product_Q.Sales.sum()
```

```
Product  Quarter
A        Q1         211
         Q2         215
B        Q1         293
         Q2         262
Name: Sales, dtype: int64
```

Total sales of each product per quarter

```
round(sales_gb_product_Q.Sales.mean(),2)
```

```
Product  Quarter
A        Q1         70.33
         Q2         71.67
B        Q1         97.67
         Q2         87.33
Name: Sales, dtype: float64
```

Average sales of each product per quarter

Convert Series to DataFrame

- The Groupby object can calculate group statistics and return a DataFrame or a Series.
- Use `to_frame()` to convert a Series to a Dataframe.

```
Product  Quarter
A        Q1         211
         Q2         215
B        Q1         293
         Q2         262
Name: Sales, dtype: int64
```

```
type(sales_gb_product_Q.Sales.sum())
```

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

```
quarterly_sales = sales_gb_product_Q.Sales.sum().to_frame(name = "Q_sales")
quarterly_sales
```

Pass a column name

		Q_sales
Product	Quarter	
A	Q1	211
	Q2	215
B	Q1	293
	Q2	262

Convert Series to DataFrame

- If you need group keys as columns
  - `df[col_name] = df.index`
  - `Reset_index()`

```
quarterly_sales.index
```

```
MultiIndex([('A', 'Q1'),
            ('A', 'Q2'),
            ('B', 'Q1'),
            ('B', 'Q2')],
           names=['Product', 'Quarter'])
```

		Q_sales
Product	Quarter	
A	Q1	211
	Q2	215
B	Q1	293
	Q2	262

Multiple index

```
quarterly_sales.reset_index(inplace = True)
quarterly_sales
```

	Product	Quarter	Q_sales
0	A	Q1	211
1	A	Q2	215
2	B	Q1	293
3	B	Q2	262

BI

- Multiple index https://pandas.pydata.org/pandas-docs/stable/user_guide/advanced.html

Exercise

(C.1) Given the following data, calculate the average price of wine in each province.

```
wine_df.head(5)
```

	winery	country	price	province
0	Domaine Vincent Girardin	France	94	Burgundy
1	Domaine de Bellene	France	60	Burgundy
2	Au Pied du Mont Chauve	France	43	Burgundy
3	Chateau Raymond-Lafon	France	120	Bordeaux
4	Chateau Balac	France	45	Bordeaux

(C.2) Import the dataset `fashion.csv`. Remove the columns with missing values.

(C.3) Extract the years from the column Date and put them in a new column named `Year`. Display the first 10 rows.

(C.4) Grouping data based on column `Year`.

(C.5) Calculate the total annual sales of each brand.