資料分析

資料IO

```
# 資料的匯入
import pandas as pd

df = pd.read_csv('data/employees.csv')
print(type(df))
print(df.tail())
print(df.salary.mean())
# 資料的匯出
df = df.tail()
df.to_csv('data/new.csv')
```

```
<class 'pandas.core.frame.DataFrame'>
   employee_id first_name last_name    email  phone_number     job_id  \
102       202      Pat      Fay     PFAY  603.123.6666     MK_REP
103       203    Susan    Mavris   SMAVRIS  515.123.7777     HR_REP
104       204   Hermann     Baer    HBAER  515.123.8888     PR_REP
105       205   Shelley   Higgins  SHIGGINS  515.123.8080     AC_MGR
106       206   William    Gietz    WGIETZ  515.123.8181  AC_ACCOUNT

    salary  commission_pct  manager_id  department_id
102  6000.0         NaN     201.0        20.0
103  6500.0         NaN     101.0        40.0
104 10000.0          NaN     101.0        70.0
105 12000.0          NaN     101.0       110.0
106  8300.0         NaN     205.0       110.0
6461.682242990654
```

JSON

```
# json
df = pd.read_json('data/data1.json')
```

```
print(type(df))

import json

with open('data/test.json') as f:
    data = json.load(f)
# print(data['users'])
print(type(data))
df = pd.DataFrame(data['users'])
print(type(df))
df
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'dict'>
<class 'pandas.core.frame.DataFrame'>
```

	id	name	age	email	is_active	join_date
0	1	張三	28	zhangsan@example.com	True	2022-03-15
1	2	李四	35	lisi@example.com	False	2021-11-02
2	3	王五	24	wangwu@example.com	True	2023-01-20

缺失值的處理

```
# nan:not a number
import pandas as pd
import numpy as np

s = pd.Series([12, 25, np.nan, None, pd.NA])
df = pd.DataFrame([1, pd.NA, 2](1,%20pd.NA,%202), columns=['第1列', '第2列', '第3列'])
print(s)
# 檢視是否是缺失值
print(s.isna())
```

```
print(s.isnull())
print(df.isna())
print(df.isnull())
print(df.isna().sum(axis=1))
print(s.isna().sum()) #檢視缺失值的個數

# 剔除缺失值
print(s.dropna())
print('-' * 30)
print(df)
print(df.dropna()) #剔除一整條的記錄
print(df.dropna(how='all')) #如果所有的值都是缺失值，刪除這一行
print(df.dropna(thresh=1)) #如果至少有n個值不是缺失值，就保留
print(df.dropna(axis=1)) #剔除一整列的記錄
print(df.dropna(subset=['第1列'])) #如果某列有缺失值，則刪除這一行

# 填充缺失值
df = pd.read_csv('data/weather_withna.csv')
df.tail()
df.isna().sum(axis=0)
df.head()
print(df.fillna({'temp_max': 20, 'wind': 2.5}).tail()) #使用字典來填充
print(df.fillna(df['temp_max', 'wind']('temp_max',%20'wind').mean()).tail()) #使用統計值來填充
print(df.ffill().tail()) #用前面的相鄰值填充
print(df.bfill().tail()) #用後面的相鄰值填充
```

```
0    12
1    25
2   NaN
3  None
4  <NA>
dtype: object
0   False
1   False
2    True
3    True
4    True
```

```
dtype: bool
0   False
1   False
2    True
3    True
4    True
dtype: bool
   第1列   第2列   第3列
0  False   True  False
1  False  False  False
2   True  False  False
   第1列   第2列   第3列
0  False   True  False
1  False  False  False
2   True  False  False
0   1
1   0
2   1
dtype: int64
3
0   12
1   25
dtype: object
------------------------------
  第1列  第2列 第3列
0  1.0 <NA>   2
1  2.0    3   5
2  NaN    4   6
  第1列 第2列 第3列
1  2.0  3   5
  第1列  第2列 第3列
0  1.0 <NA>   2
1  2.0    3   5
2  NaN    4   6
  第1列  第2列 第3列
0  1.0 <NA>   2
1  2.0    3   5
2  NaN    4   6
```

```
  第3列
0   2
1   5
2   6
  第1列  第2列  第3列
0  1.0 <NA>   2
1  2.0    3   5
        date  precipitation  temp_max  temp_min  wind weather
1456  2015-12-27         NaN      20.0       NaN  2.5     NaN
1457  2015-12-28         NaN      20.0       NaN  2.5     NaN
1458  2015-12-29         NaN      20.0       NaN  2.5     NaN
1459  2015-12-30         NaN      20.0       NaN  2.5     NaN
1460  2015-12-31        20.6      12.2       5.0  3.8    rain
        date  precipitation   temp_max  temp_min     wind weather
1456  2015-12-27         NaN  15.851468       NaN  3.242055     NaN
1457  2015-12-28         NaN  15.851468       NaN  3.242055     NaN
1458  2015-12-29         NaN  15.851468       NaN  3.242055     NaN
1459  2015-12-30         NaN  15.851468       NaN  3.242055     NaN
1460  2015-12-31        20.6  12.200000       5.0  3.800000    rain
        date  precipitation  temp_max  temp_min  wind weather
1456  2015-12-27         0.0      11.1       4.4  4.8     sun
1457  2015-12-28         0.0      11.1       4.4  4.8     sun
1458  2015-12-29         0.0      11.1       4.4  4.8     sun
1459  2015-12-30         0.0      11.1       4.4  4.8     sun
1460  2015-12-31        20.6      12.2       5.0  3.8    rain
        date  precipitation  temp_max  temp_min  wind weather
1456  2015-12-27        20.6      12.2       5.0  3.8    rain
1457  2015-12-28        20.6      12.2       5.0  3.8    rain
1458  2015-12-29        20.6      12.2       5.0  3.8    rain
1459  2015-12-30        20.6      12.2       5.0  3.8    rain
1460  2015-12-31        20.6      12.2       5.0  3.8    rain
```

時間資料的處理

```
import pandas as pd
```

```
d = pd.Timestamp('2015-02-28 10:22')
d1 = pd.Timestamp('2015-02-28 13:22')
print(d)
print(type(d))
print("年：", d.year)
print("月：", d.month)
print("日：", d.day)
print(d.hour, d.minute, d.second)
print("季度：", d.quarter)
print("是否是月底：", d.is_month_end)
# 方法
print("星期幾：", d.day_name())
print("轉換為天：", d.to_period("D"))
print("轉換為季度：", d1.to_period("Q"))
print("轉換為年度：", d1.to_period("Y"))
print("轉換為月度：", d1.to_period("M"))
print("轉換為周維度：", d1.to_period("W"))
```

```
2015-02-28 10:22:00
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
年： 2015
月： 2
日： 28
10 22 0
季度： 1
是否是月底： True
星期幾： Saturday
轉換為天： 2015-02-28
轉換為季度： 2015Q1
轉換為年度： 2015
轉換為月度： 2015-02
轉換為周維度： 2015-02-23/2015-03-01
```

```
# 字串轉換為日期型別
a = pd.to_datetime('20150228')
print(a)
```

```
print(type(a))
print(a.day_name())

# dataFrame 日期轉換
df = pd.DataFrame({
   'sales': [100, 200, 300],
   'date': ['20250601', '20250602', '20250603']
})
df['datetime'] = pd.to_datetime(df['date'])
df
print(df.info())
print(type(df['datetime']))
df['week'] = df['datetime'].dt.day_name()
df['datetime'].dt.year
```

```
2015-02-28 00:00:00
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
Saturday
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   sales     3 non-null      int64
 1   date      3 non-null      object
 2   datetime  3 non-null      datetime64[ns]
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 204.0+ bytes
None
<class 'pandas.core.series.Series'>

0    2025
1    2025
2    2025
Name: datetime, dtype: int32
```

```
# csv 日期轉換
df = pd.read_csv('data/weather.csv', parse_dates=['date'])
df.info()
df['date'].dt.day_name()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   date           1461 non-null   datetime64[ns]
 1   precipitation  1461 non-null   float64       
 2   temp_max       1461 non-null   float64       
 3   temp_min       1461 non-null   float64       
 4   wind           1461 non-null   float64       
 5   weather        1461 non-null   object        
dtypes: datetime64[ns](1), float64(4), object(1)
memory usage: 68.6+ KB

0        Sunday
1        Monday
2       Tuesday
3     Wednesday
4      Thursday
        ...    
1456     Sunday
1457     Monday
1458    Tuesday
1459  Wednesday
1460   Thursday
Name: date, Length: 1461, dtype: object
```

```
# 日期資料作為索引
# df.set_index('date' , inplace=True)#設定原來的df的索引
print(df.loc["2013-01":"2013-02"])
```

```
Empty DataFrame
Columns: [date, precipitation, temp_max, temp_min, wind, weather]
Index: []
```

```
# 時間間隔
d1 = pd.Timestamp('2013-01-15')
d2 = pd.Timestamp('2023-02-23')
d3 = d2 - d1
print(type(d3))
print(d3)
```

```
<class 'pandas._libs.tslibs.timedeltas.Timedelta'>
3691 days 00:00:00
```

```
df = pd.read_csv('data/weather.csv', parse_dates=['date'])
df.info()
df['delta'] = df['date'] - df['date'][0]
df.set_index('delta', inplace=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   date          1461 non-null   datetime64[ns]
 1   precipitation  1461 non-null   float64       
 2   temp_max      1461 non-null   float64       
 3   temp_min      1461 non-null   float64       
 4   wind          1461 non-null   float64       
 5   weather       1461 non-null   object        
dtypes: datetime64[ns](1), float64(4), object(1)
memory usage: 68.6+ KB
```

```
df
print(df.loc['10 days':'20 days'])
```

```
          date  precipitation  temp_max  temp_min  wind weather
delta
10 days 2012-01-11            0.0       6.1      -1.1   5.1     sun
11 days 2012-01-12            0.0       6.1      -1.7   1.9     sun
12 days 2012-01-13            0.0       5.0      -2.8   1.3     sun
13 days 2012-01-14            4.1       4.4       0.6   5.3    snow
14 days 2012-01-15            5.3       1.1      -3.3   3.2    snow
15 days 2012-01-16            2.5       1.7      -2.8   5.0    snow
16 days 2012-01-17            8.1       3.3       0.0   5.6    snow
17 days 2012-01-18           19.8       0.0      -2.8   5.0    snow
18 days 2012-01-19           15.2      -1.1      -2.8   1.6    snow
19 days 2012-01-20           13.5       7.2      -1.1   2.3    snow
20 days 2012-01-21            3.0       8.3       3.3   8.2    rain
```

```
days = pd.date_range("2025-07-03", "2026-02-09", freq="W")
days = pd.date_range("2025-07-03", periods=10, freq="QE")
print(days)
```

```
DatetimeIndex(['2025-09-30', '2025-12-31', '2026-03-31', '2026-06-30',
               '2026-09-30', '2026-12-31', '2027-03-31', '2027-06-30',
               '2027-09-30', '2027-12-31'],
              dtype='datetime64[ns]', freq='QE-DEC')
```

```
df = pd.read_csv('data/weather.csv', parse_dates=['date'])
# 重新取樣
df.set_index('date', inplace=True)
```

```
df["temp_max", "temp_min"]
("temp_max",%20"temp_min").resample("MS").mean()
```

	temp_max	temp_min
date		
2012-01-01	7.054839	1.541935
2012-02-01	9.275862	3.203448
2012-03-01	9.554839	2.838710

	temp_max	temp_min
date		
2012-04-01	14.873333	5.993333
2012-05-01	17.661290	8.190323
2012-06-01	18.693333	10.480000
2012-07-01	22.906452	12.932258
2012-08-01	25.858065	14.009677
2012-09-01	22.880000	11.243333
2012-10-01	15.829032	8.380645
2012-11-01	11.326667	5.226667
2012-12-01	7.235484	3.293548
2013-01-01	6.106452	0.796774
2013-02-01	9.467857	4.325000
2013-03-01	12.709677	4.977419
2013-04-01	14.243333	6.696667
2013-05-01	19.625806	9.922581
2013-06-01	23.253333	13.163333
2013-07-01	26.093548	13.932258
2013-08-01	26.119355	15.480645
2013-09-01	21.360000	13.590000
2013-10-01	14.229032	7.638710
2013-11-01	12.053333	5.590000
2013-12-01	7.022581	1.570968
2014-01-01	9.600000	4.096774
2014-02-01	8.200000	2.635714
2014-03-01	12.906452	5.425806
2014-04-01	15.460000	6.730000
2014-05-01	19.870968	10.216129
2014-06-01	21.590000	11.756667
2014-07-01	26.900000	14.425806
2014-08-01	26.383871	14.893548
2014-09-01	23.163333	13.233333

	temp_max	temp_min
date		
2014-10-01	17.961290	10.883871
2014-11-01	11.030000	4.510000
2014-12-01	10.138710	4.609677
2015-01-01	10.154839	4.351613
2015-02-01	12.517857	6.085714
2015-03-01	14.377419	6.193548
2015-04-01	15.503333	6.030000
2015-05-01	20.025806	10.129032
2015-06-01	26.063333	13.576667
2015-07-01	28.093548	15.500000
2015-08-01	26.087097	14.693548
2015-09-01	20.293333	11.366667
2015-10-01	17.538710	10.500000
2015-11-01	9.683333	3.480000
2015-12-01	8.380645	3.825806

```
df["temp_max", "temp_min"]("temp_max",%20"temp_min").resample("YE").mean()
```

	temp_max	temp_min
date		
2012-12-31	15.276776	7.289617
2013-12-31	16.058904	8.153973
2014-12-31	16.995890	8.662466
2015-12-31	17.427945	8.835616

```
import pandas as pd

data = {
   "name": ['alice', 'alice', 'bob', 'alice', 'jack', 'bob'],
   "age": [26, 25, 30, 25, 35, 30],
   'city': ['NY', 'NY', 'LA', 'NY', 'SF', 'LA']
```

```
}
df = pd.DataFrame(data)
```

```
df.duplicated() #一整條記錄都是一樣的，標記為重複，返回True
df.drop_duplicates(subset=['name']) #根據指定列去重
df.drop_duplicates(subset=['name'], keep='last') #保留最後一次出現的行
```

	name	age	city
3	alice	25	NY
4	jack	35	SF
5	bob	30	LA

資料型別的轉換

```
df = pd.read_csv('data/sleep.csv')
df.dtypes
```

```
person_id                int64
gender                  object
age                      int64
occupation              object
sleep_duration         float64
sleep_quality          float64
physical_activity_level    int64
stress_level             int64
bmi_category            object
blood_pressure          object
heart_rate               int64
daily_steps              int64
sleep_disorder          object
dtype: object
```

```
df['age'] = df['age'].astype('int16')
```

```
df['gender'] = df['gender'].astype('category')
```

```
df.gender
```

```
0      Male
1    Female
2      Male
3      Male
4      Male
      ...
395    Female
396    Female
397    Female
398    Female
399     Male
Name: gender, Length: 400, dtype: category
Categories (2, object): ['Female', 'Male']
```

```
df['is_male'] = df['gender'].map({'Female': True, 'Male': False})
```

```
df.is_male
```

```
0    False
1     True
2    False
3    False
4    False
      ...
395    True
396    True
397    True
398    True
399   False
```

```
Name: is_male, Length: 400, dtype: category
Categories (2, bool): [True, False]
```

資料變形

```
import pandas as pd

data = {
   'ID': [1, 2],
   'name': ['alice', 'bob'],
   'Math': [90, 85],
   'English': [88, 92],
   'Science': [95, 89]
}
df = pd.DataFrame(data)
print(df)
df.T  #行列轉置
# 寬錶轉換成長表
df2 = pd.melt(df, id_vars=['ID', 'name'], var_name='科目', value_name='分數')
df2.sort_values('name')
print(df2)
# 長錶轉寬表
pd.pivot(df2, index=['ID', 'name'], columns='科目', values='分數')
```

```
  ID   name  Math  English  Science
0  1  alice    90       88       95
1  2    bob    85       92       89
  ID   name      科目  分數
0  1  alice    Math  90
1  2    bob    Math  85
2  1  alice  English  88
3  2    bob  English  92
4  1  alice  Science  95
5  2    bob  Science  89
```

	科目	English	Math	Science
ID	name			
1	alice	88	90	95
2	bob	92	85	89

```
data = {
    'ID': [1, 2],
    'name': ['alice smith', 'bob smith'],
    'Math': [90, 85],
    'English': [88, 92],
    'Science': [95, 89]
}
df = pd.DataFrame(data)
# 分列
df['first', 'last']('first',%20'last') = df['name'].str.split(" ", expand=True)
df = pd.read_csv('data/sleep.csv')
df = df['person_id', 'blood_pressure']('person_id',%20'blood_pressure')
df['high', 'low']('high',%20'low') = df['blood_pressure'].str.split('/', expand=True)
df['high'] = df['high'].astype('int64')
df['low'] = df['low'].astype('int64')
df.info()
df.high.mean()
df.low.mean()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   person_id      400 non-null    int64
 1   blood_pressure  400 non-null    object
 2   high           400 non-null    int64
 3   low            400 non-null    int64
dtypes: int64(3), object(1)
memory usage: 12.6+ KB
```

```
np.float64(73.04)
```

```
# 資料分箱 pd.cut(x,bins,labels)
import pandas as pd
df = pd.read_csv('data/employees.csv')
df.head(10)
```

	employee_id	first_name	last_name	email	phone_number
0	100	Steven	King	SKING	515.123.4567
1	101	N_ann	Kochhar	NKOCHHAR	515.123.4568
2	102	Lex	De Haan	LDEHAAN	515.123.4569
3	103	Alexander	Hunold	AHUNOLD	590.423.4567
4	104	Bruce	Ernst	BERNST	590.423.4568
5	105	David	Austin	DAUSTIN	590.423.4569
6	106	Valli	Pataballa	VPATABAL	590.423.4560
7	107	Diana	Lorentz	DLORENTZ	590.423.5567
8	108	Nancy	Greenberg	NGREENBE	515.124.4569
9	109	Daniel	Faviet	DFAVIET	515.124.4169

```
df1 = df.head(10)['employee_id', 'salary']('employee_id',%20'salary')
df1
```

	employee_id	salary
0	100	24000.0
1	101	17000.0
2	102	17000.0
3	103	9000.0
4	104	6000.0
5	105	4800.0
6	106	4800.0
7	107	4200.0
8	108	12000.0

	employee_id	salary
9	109	9000.0

```
pd.cut(df1['salary'], bins=3)  #bins=n，分成n段區間，起始值、結束值是所有資料的最小值、最大值
#4180~14100~24000
pd.cut(df1['salary'], bins=3).value_counts()
pd.cut(df1['salary'], bins=[0, 10000, 20000, 30000])  #bins=list，分成n段區間
pd.cut(df1['salary'], bins=[0, 10000, 20000, 30000]).value_counts()
df1['收入範圍'] = pd.cut(df1['salary'], bins=[0, 10000, 20000, 30000], labels=['低', '中', '高'])  #bins=list，分成n段區間
pd.qcut(df1['salary'], 3).value_counts()
```

```
salary
(12000.0, 24000.0]    4
(4199.999, 6000.0]    3
(6000.0, 12000.0]     3
Name: count, dtype: int64
```

```
# 睡眠資料
df = pd.read_csv('data/sleep.csv')
df1 = df.head(10)['person_id', 'sleep_quality']('person_id',%20'sleep_quality')
df1
df['睡眠質量'] = pd.cut(df['sleep_quality'], bins=3, labels=
['差', '中', '優'])
df['睡眠質量'].value_counts()
df.head(10)
df['gender'] = df['gender'].astype('category')
df['gender'].value_counts()
# 字串-->類別-->統計
# 數值-->分箱-->統計
print(df['gender'].dtype)
print(df['睡眠質量'].dtype)
```

```
category
category
```

```
# df.rename()  df.set_index()  df.reset_index()
df = pd.DataFrame({
   'name': ['jack', 'alice', 'tom', 'bob'],
   'age': [20, 30, 40, 50],
   'gender': ['female', 'male', 'female', 'male']
})
df.set_index("name", inplace=True)
df.reset_index(inplace=True)
df.rename(columns={"age": "年齡"}, index={0: 4})
```

	name	年齡	gender
4	jack	20	female
1	alice	30	male
2	tom	40	female
3	bob	50	male

	姓名	年齡	性別
1	jack	20	female
2	alice	30	male
3	tom	40	female
4	bob	50	male

df = pd.read_csv('data/employees.csv')
df = df.dropna(subset=['department_id'])
df['department_id'] = df['department_id'].astype('int64')

計算不同部門的平均薪資

df.groupby('department_id').groups #檢視分組
df.groupby('department_id').get_group(20) #檢視具體的某個分組資料
df2 = df.groupby('department_id')'salary'.mean()
df2['salary'] = df2['salary'].round(2)
df2 = df2.reset_index()
df2.sort_values('salary', ascending=False)

```
<div>
<style scoped>
    .dataframe tbody tr th:only-of-type {
        vertical-align: middle;
    }

    .dataframe tbody tr th {
        vertical-align: top;
    }

    .dataframe thead th {
        text-align: right;
    }
</style>
<table border="1" class="dataframe">
  <thead>
    <tr style="text-align: right;">
      <th></th>
      <th>department_id</th>
      <th>salary</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>8</th>
      <td>90</td>
      <td>19333.33</td>
    </tr>
    <tr>
      <th>10</th>
      <td>110</td>
      <td>10150.00</td>
    </tr>
    <tr>
      <th>6</th>
      <td>70</td>
      <td>10000.00</td>
```

```
</tr>
<tr>
 <th>1</th>
 <td>20</td>
 <td>9500.00</td>
</tr>
<tr>
 <th>7</th>
 <td>80</td>
 <td>8955.88</td>
</tr>
<tr>
 <th>9</th>
 <td>100</td>
 <td>8600.00</td>
</tr>
<tr>
 <th>3</th>
 <td>40</td>
 <td>6500.00</td>
</tr>
<tr>
 <th>5</th>
 <td>60</td>
 <td>5760.00</td>
</tr>
<tr>
 <th>0</th>
 <td>10</td>
 <td>4400.00</td>
</tr>
<tr>
 <th>2</th>
 <td>30</td>
 <td>4150.00</td>
</tr>
<tr>
 <th>4</th>
```

```
    <td>50</td>
    <td>3475.56</td>
   </tr>
  </tbody>
</table>
</div>
```

```python
# 計算不同部門不同崗位的人的平均薪資
df2 = df.groupby(['department_id', 'job_id']('salary').mean()
df2 = df2.reset_index()
df2['salary'] = df2['salary'].round(1)
df2.sort_values('salary', ascending=False)
```

	department_id	job_id	salary
13	90	AD_PRES	24000.0
14	90	AD_VP	17000.0
1	20	MK_MAN	13000.0
11	80	SA_MAN	12200.0
16	100	FI_MGR	12000.0
18	110	AC_MGR	12000.0
4	30	PU_MAN	11000.0
10	70	PR_REP	10000.0
12	80	SA_REP	8396.6
17	110	AC_ACCOUNT	8300.0
15	100	FI_ACCOUNT	7920.0
8	50	ST_MAN	7280.0
5	40	HR_REP	6500.0
2	20	MK_REP	6000.0
9	60	IT_PROG	5760.0
0	10	AD_ASST	4400.0
6	50	SH_CLERK	3215.0

	department_id	job_id	salary
7	50	ST_CLERK	2785.0
3	30	PU_CLERK	2780.0

```
# 企鵝資料分析
# 1. 匯入必要的庫
import pandas as pd
import numpy as np

# 2. 匯入資料 喙
df = pd.read_csv('data/penguins.csv')
df.head(5)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   species           344 non-null    object 
 1   island            344 non-null    object 
 2   bill_length_mm    342 non-null    float64
 3   bill_depth_mm     342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g       342 non-null    float64
 6   sex               333 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

```
# 3. 資料清洗
# 缺失值的檢查
print(df.isna().sum())
df.dropna(inplace=True)
```

```
species           0
island            0
bill_length_mm    2
bill_depth_mm     2
flipper_length_mm 2
body_mass_g       2
sex              11
dtype: int64
```

```
# 4. 資料特徵的構造
df['sex'] = df['sex'].astype('category')
df['bill_ratio'] = df['bill_length_mm'] / df['bill_depth_mm']
df.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_
0	Adelie	Torgersen	39.1	18.7	181.0
1	Adelie	Torgersen	39.5	17.4	186.0
2	Adelie	Torgersen	40.3	18.0	195.0
4	Adelie	Torgersen	36.7	19.3	193.0
5	Adelie	Torgersen	39.3	20.6	190.0

```
# 5. 資料分析
# 資料分箱-把體重分為三個等級
labels = ['低', '中', '高']
df['mass_level'] = pd.cut(df['body_mass_g'], bins=3, labels=labels)
print(df['mass_level'].value_counts())
# 按島嶼、性別分組分析
df.groupby(['sex', 'island']).agg({
   'body_mass_g': ['mean', 'count'],
})
```

```
mass_level
低   150
中   128
高    55
```

```
Name: count, dtype: int64


/tmp/ipykernel_696402/1536504977.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  df.groupby(['sex', 'island']).agg({
```

		body_mass_g	
		mean	count
sex	island		
Female	Biscoe	4319.375000	80
	Dream	3446.311475	61
	Torgersen	3395.833333	24
Male	Biscoe	5104.518072	83
	Dream	3987.096774	62
	Torgersen	4034.782609	23

```
# 睡眠質量分析
# 1.匯入庫
import pandas as pd
import numpy as np

# 2.匯入資料
df = pd.read_csv('data/sleep.csv')
df.head()
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   person_id           400 non-null    int64
```

```
 1  gender                  400 non-null    object
 2  age                     400 non-null    int64
 3  occupation              400 non-null    object
 4  sleep_duration          400 non-null    float64
 5  sleep_quality           400 non-null    float64
 6  physical_activity_level 400 non-null    int64
 7  stress_level            400 non-null    int64
 8  bmi_category            400 non-null    object
 9  blood_pressure          400 non-null    object
 10 heart_rate              400 non-null    int64
 11 daily_steps             400 non-null    int64
 12 sleep_disorder          110 non-null    object
dtypes: float64(2), int64(6), object(5)
memory usage: 40.8+ KB
```

	person_id	age	sleep_duration	sleep_quality	physical_act
count	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	39.950000	8.041250	6.125750	64.985000
std	115.614301	14.038883	2.390787	1.975733	32.297874
min	1.000000	18.000000	4.100000	1.000000	10.000000
25%	100.750000	29.000000	5.900000	4.700000	35.000000
50%	200.500000	40.000000	8.200000	6.100000	65.500000
75%	300.250000	49.000000	10.125000	7.425000	94.000000
max	400.000000	90.000000	12.000000	10.000000	120.000000

```
# 3.資料清洗
df.isna().sum()
df.drop(columns='sleep_disorder', inplace=True)
```

```
# 4. 資料特徵的構造
df['gender'] = df['gender'].astype('category')
df['occupation'] = df['occupation'].astype('category')
df['bmi_category'] = df['bmi_category'].astype('category')
df['high', 'low']('high',%20'low') = df['blood_pressure'].str.split('/', expand=True)

# 睡眠質量的分箱
```

```
labels = ['差', '中', '優']
df['quality_level'] = pd.cut(df['sleep_quality'], bins=3, labels=labels)
age_labels = ['青少年', '中年', '老年']
df['age_level'] = pd.cut(df['age'], bins=3, labels=age_labels)
df.head()
```

	person_id	gender	age	occupation	sleep_duration	sleep_qualit
0	1	Male	29	Manual Labor	7.4	7.0
1	2	Female	43	Retired	4.2	4.9
2	3	Male	44	Retired	6.1	6.0
3	4	Male	29	Office Worker	8.3	10.0
4	5	Male	67	Retired	9.1	9.5

```
# 5.資料的統計、分析
print(df['bmi_category'].value_counts())
```

```
bmi_category
Overweight    109
Underweight   102
Obese          98
Normal         91
Name: count, dtype: int64
```

```
# 根據不同的bmi分組，睡眠質量
df.groupby(['age_level', 'bmi_category']).agg({
   'sleep_duration': 'mean',
   'sleep_quality': 'mean',
   'stress_level': 'mean'
})
```

```
/tmp/ipykernel_696402/3959039482.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt
```

```
the future default and silence this warning.
  df.groupby(['age_level', 'bmi_category']).agg({
```

		sleep_duration	sleep_quality	stress_level
age_level	bmi_category			
青少年	Normal	8.100000	6.332000	4.860000
	Obese	8.250000	6.253448	5.534483
	Overweight	8.214286	6.171429	5.317460
	Underweight	7.603279	5.883607	5.426230
中年	Normal	7.422222	6.650000	4.944444
	Obese	7.805556	6.216667	5.888889
	Overweight	8.246154	5.956410	5.974359
	Underweight	8.497500	5.907500	5.750000
老年	Normal	7.420000	4.240000	4.200000
	Obese	7.900000	5.025000	8.000000
	Overweight	8.971429	6.285714	6.714286
	Underweight	10.500000	6.200000	6.000000