zhuanlan.zhihu.com

10套练习，教你如何用Pandas做数据分析【6-10】

29-37 minutes

Pandas是入门Python做数据分析所必须要掌握的一个库，本文精选了十套练习题，帮助读者上手Python代码，完成数据集探索。

本文内容由和鲸社区翻译整理自Github，建议读者完成科赛网 从零上手Python关键代码 和 Pandas基础命令速查表 教程学习的之后，再对本教程代码进行调试学习。

【小提示：本文所使用的数据集下载地址：DATA | TRAIN 练习数据集】

↓↓↓练习【6-10】↓↓↓

练习6-统计

探索风速数据

相应数据集：wind.data

步骤1 导入必要的库

```
# 运行以下代码
import pandas as pd
import datetime
```

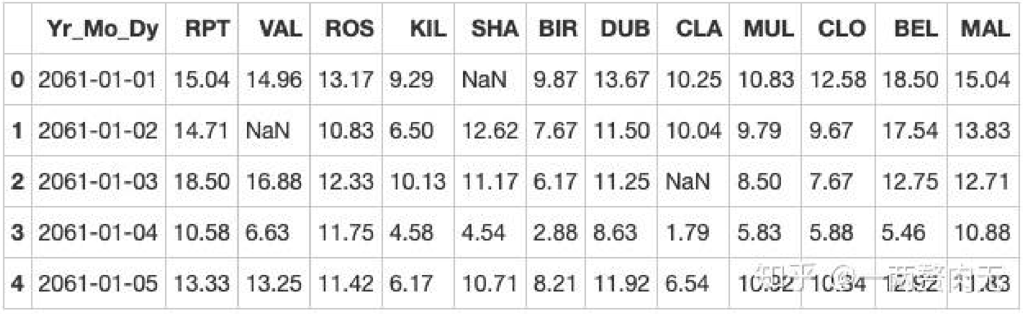
步骤2 从以下地址导入数据

```
import pandas as pd

# 运行以下代码

path6 = "../input/pandas_exercise/exercise_data/wind.data"  # wind.data
```

步骤3 将数据作存储并且设置前三列为合适的索引

```
import datetime

# 运行以下代码

data = pd.read_table(path6, sep = "\s+", parse_dates = [[0,1,2]])

data.head()
```

out[293]:

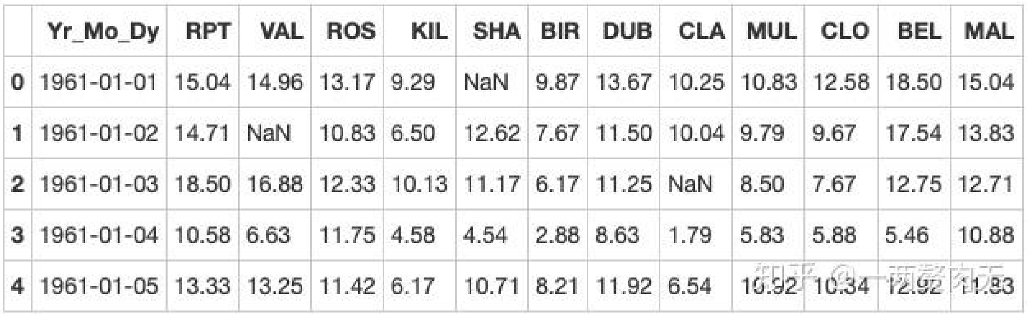


	Yr_Mo_Dy	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
0	2061-01-01	15.04	14.96	13.17	9.29	NaN	9.87	13.67	10.25	10.83	12.58	18.50	15.04
1	2061-01-02	14.71	NaN	10.83	6.50	12.62	7.67	11.50	10.04	9.79	9.67	17.54	13.83
2	2061-01-03	18.50	16.88	12.33	10.13	11.17	6.17	11.25	NaN	8.50	7.67	12.75	12.71
3	2061-01-04	10.58	6.63	11.75	4.58	4.54	2.88	8.63	1.79	5.83	5.88	5.46	10.88
4	2061-01-05	13.33	13.25	11.42	6.17	10.71	8.21	11.92	6.54	10.92	10.34	12.92	11.83

步骤4 2061年？我们真的有这一年的数据？创建一个函数并用它去修复这个bug

```
# 运行以下代码

def fix_century(x):

    year = x.year - 100 if x.year > 1989 else x.year

    return datetime.date(year, x.month, x.day)


# apply the function fix_century on the column and replace the values to the right ones

data['Yr_Mo_Dy'] = data['Yr_Mo_Dy'].apply(fix_century)


# data.info()

data.head()
```

out[294]:

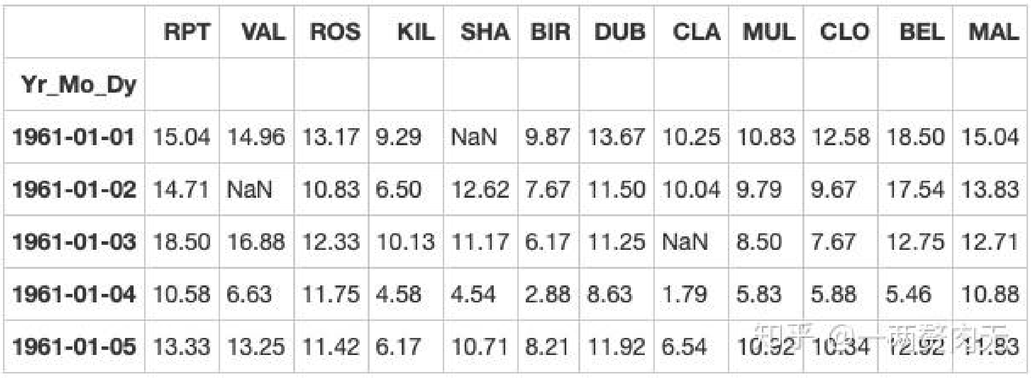


	Yr_Mo_Dy	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
0	1961-01-01	15.04	14.96	13.17	9.29	NaN	9.87	13.67	10.25	10.83	12.58	18.50	15.04
1	1961-01-02	14.71	NaN	10.83	6.50	12.62	7.67	11.50	10.04	9.79	9.67	17.54	13.83
2	1961-01-03	18.50	16.88	12.33	10.13	11.17	6.17	11.25	NaN	8.50	7.67	12.75	12.71
3	1961-01-04	10.58	6.63	11.75	4.58	4.54	2.88	8.63	1.79	5.83	5.88	5.46	10.88
4	1961-01-05	13.33	13.25	11.42	6.17	10.71	8.21	11.92	6.54	10.92	10.34	12.92	11.83

步骤5 将日期设为索引，注意数据类型，应该是`datetime64[ns]`

```
# 运行以下代码
# transform Yr_Mo_Dy it to date type datetime64
data["Yr_Mo_Dy"] = pd.to_datetime(data["Yr_Mo_Dy"])

# set 'Yr_Mo_Dy' as the index
data = data.set_index('Yr_Mo_Dy')

data.head()
# data.info()
```

out[295]:

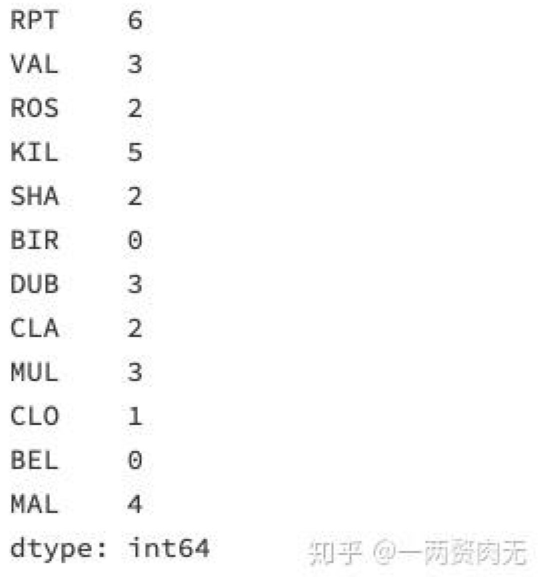


	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL
Yr_Mo_Dy												
1961-01-01	15.04	14.96	13.17	9.29	NaN	9.87	13.67	10.25	10.83	12.58	18.50	15.04
1961-01-02	14.71	NaN	10.83	6.50	12.62	7.67	11.50	10.04	9.79	9.67	17.54	13.83
1961-01-03	18.50	16.88	12.33	10.13	11.17	6.17	11.25	NaN	8.50	7.67	12.75	12.71
1961-01-04	10.58	6.63	11.75	4.58	4.54	2.88	8.63	1.79	5.83	5.88	5.46	10.88
1961-01-05	13.33	13.25	11.42	6.17	10.71	8.21	11.92	6.54	10.92	10.34	12.92	11.83

步骤6 对应每一个location，一共有多少数据值缺失

```
# 运行以下代码
data.isnull().sum()
```

out[296]:

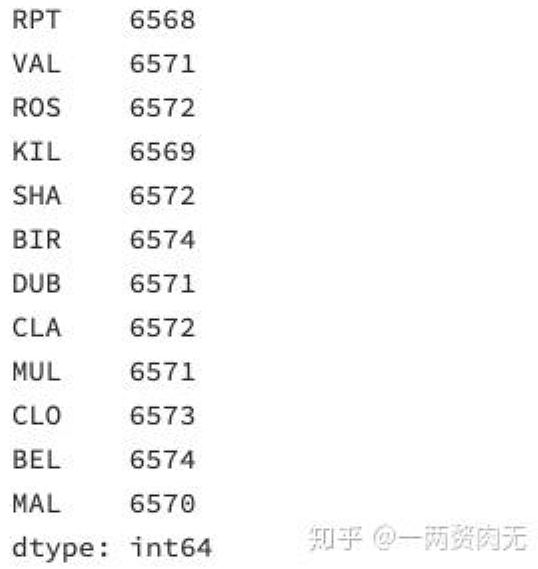


```
RPT    6
VAL    3
ROS    2
KIL    5
SHA    2
BIR    0
DUB    3
CLA    2
MUL    3
CLO    1
BEL    0
MAL    4
dtype: int64
```

步骤7 对应每一个location，一共有多少完整的数据值

```
# 运行以下代码
data.shape[0] - data.isnull().sum()
```

out[297]:



```
RPT    6568
VAL    6571
ROS    6572
KIL    6569
SHA    6572
BIR    6574
DUB    6571
CLA    6572
MUL    6571
CLO    6573
BEL    6574
MAL    6570
dtype: int64
```

步骤8 对于全体数据，计算风速的平均值

```
# 运行以下代码
data.mean().mean()
```

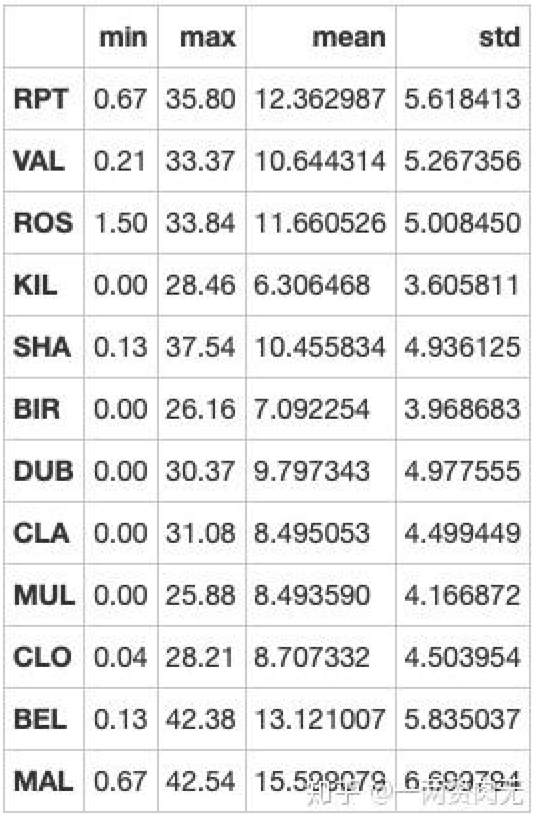
out[298]:

10.227982360836924

步骤9 创建一个名为`loc_stats`的数据框去计算并存储每个location的风速最小值，最大值，平均值和标准差

```
# 运行以下代码
loc_stats = pd.DataFrame()

loc_stats['min'] = data.min() # min
loc_stats['max'] = data.max() # max
loc_stats['mean'] = data.mean() # mean
loc_stats['std'] = data.std() # standard deviations

loc_stats
```

out[299]:



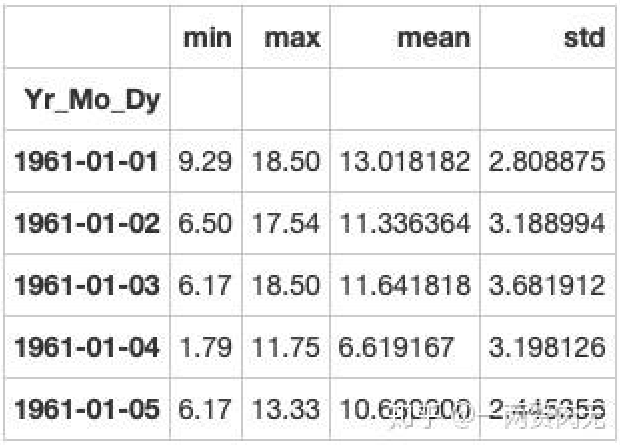
	min	max	mean	std
RPT	0.67	35.80	12.362987	5.618413
VAL	0.21	33.37	10.644314	5.267356
ROS	1.50	33.84	11.660526	5.008450
KIL	0.00	28.46	6.306468	3.605811
SHA	0.13	37.54	10.455834	4.936125
BIR	0.00	26.16	7.092254	3.968683
DUB	0.00	30.37	9.797343	4.977555
CLA	0.00	31.08	8.495053	4.499449
MUL	0.00	25.88	8.493590	4.166872
CLO	0.04	28.21	8.707332	4.503954
BEL	0.13	42.38	13.121007	5.835037
MAL	0.67	42.54	15.599079	6.699794

步骤10 创建一个名为`day_stats`的数据框去计算并存储所有location的风速最小值，最大值，平均值和标准差

```
# 运行以下代码
# create the dataframe
day_stats = pd.DataFrame()
```

```
# this time we determine axis equals to one so it gets each row.
day_stats['min'] = data.min(axis = 1) # min
day_stats['max'] = data.max(axis = 1) # max
day_stats['mean'] = data.mean(axis = 1) # mean
day_stats['std'] = data.std(axis = 1) # standard deviations

day_stats.head()
```

out[300]:



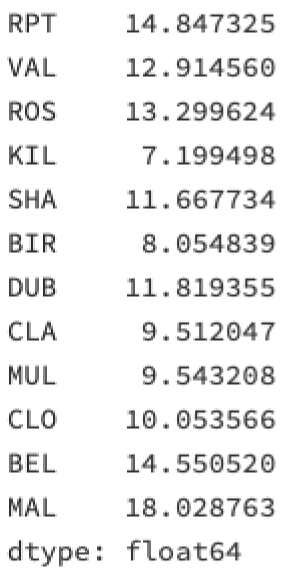
	min	max	mean	std
Yr_Mo_Dy				
1961-01-01	9.29	18.50	13.018182	2.808875
1961-01-02	6.50	17.54	11.336364	3.188994
1961-01-03	6.17	18.50	11.641818	3.681912
1961-01-04	1.79	11.75	6.619167	3.198126
1961-01-05	6.17	13.33	10.630000	2.445356

步骤11 对于每一个location，计算一月份的平均风速

(注意，1961年的1月和1962年的1月应该区别对待)

```
# 运行以下代码
# creates a new column 'date' and gets the values from the index
data['date'] = data.index

# creates a column for each value from date
data['month'] = data['date'].apply(lambda date: date.month)
data['year'] = data['date'].apply(lambda date: date.year)
data['day'] = data['date'].apply(lambda date: date.day)

# gets all value from the month 1 and assign to janyary_winds
january_winds = data.query('month == 1')

# gets the mean from january_winds, using .loc to not print the mean of month,
year and day
january_winds.loc[:,'RPT':"MAL"].mean()
```

out[301]:

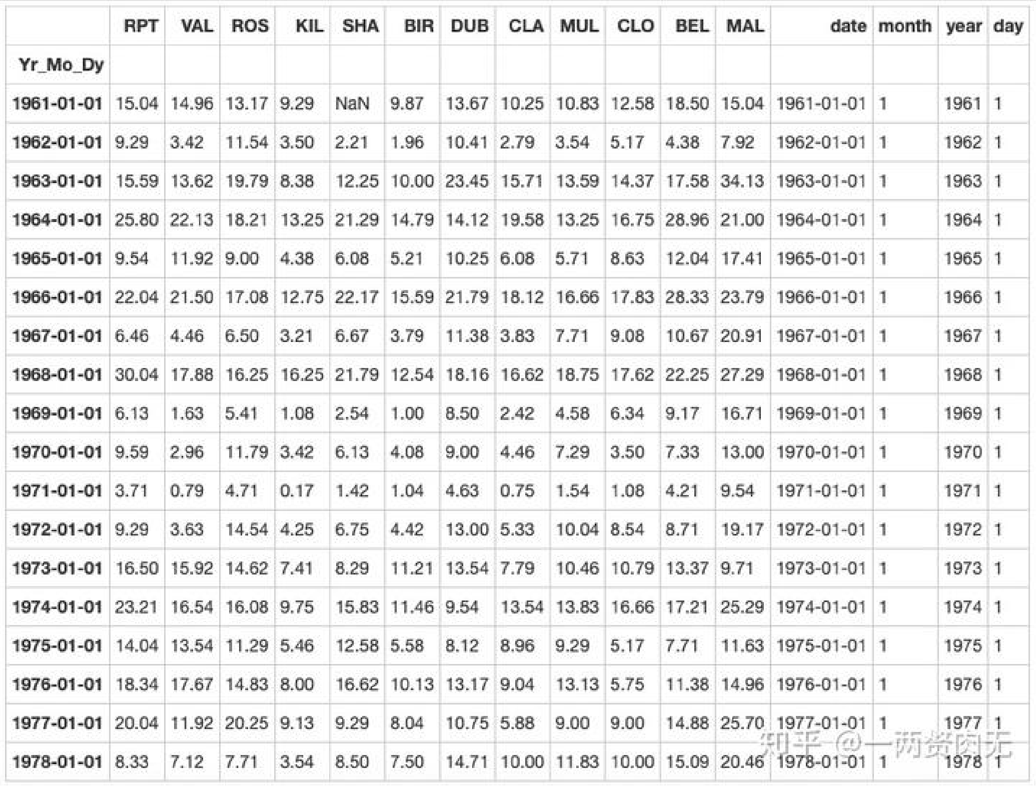


```
RPT    14.847325
VAL    12.914560
ROS    13.299624
KIL     7.199498
SHA    11.667734
BIR     8.054839
DUB    11.819355
CLA     9.512047
MUL     9.543208
CLO    10.053566
BEL    14.550520
MAL    18.028763
dtype: float64
```

步骤12 对于数据记录按照年为频率取样

```
# 运行以下代码
data.query('month == 1 and day == 1')
```

out[302]:

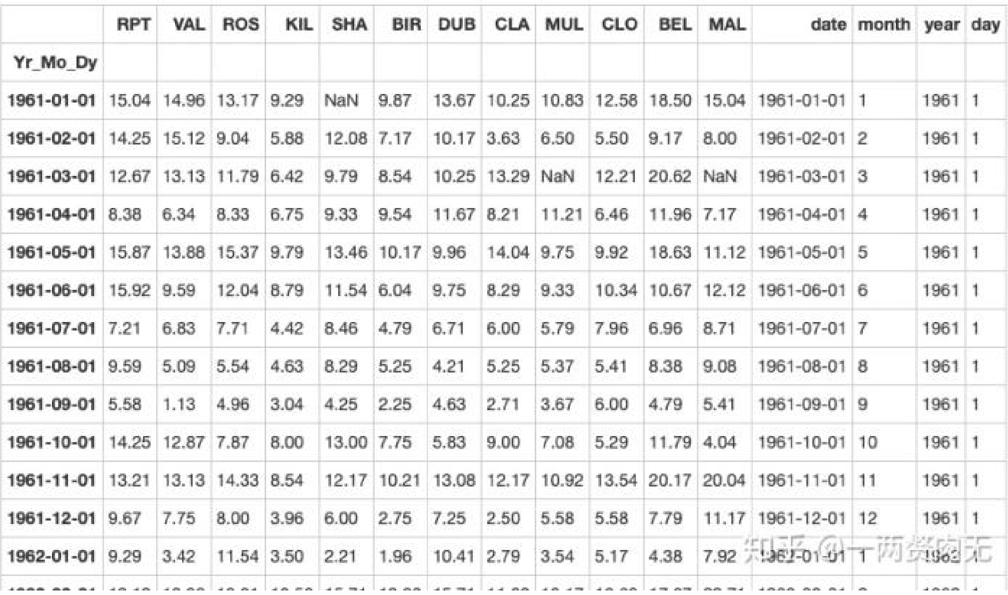


	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL	date	month	year	day
Yr_Mo_Dy																
1961-01-01	15.04	14.96	13.17	9.29	NaN	9.87	13.67	10.25	10.83	12.58	18.50	15.04	1961-01-01	1	1961	1
1962-01-01	9.29	3.42	11.54	3.50	2.21	1.96	10.41	2.79	3.54	5.17	4.38	7.92	1962-01-01	1	1962	1
1963-01-01	15.59	13.62	19.79	8.38	12.25	10.00	23.45	15.71	13.59	14.37	17.58	34.13	1963-01-01	1	1963	1
1964-01-01	25.80	22.13	18.21	13.25	21.29	14.79	14.12	19.58	13.25	16.75	28.96	21.00	1964-01-01	1	1964	1
1965-01-01	9.54	11.92	9.00	4.38	6.08	5.21	10.25	6.08	5.71	8.63	12.04	17.41	1965-01-01	1	1965	1
1966-01-01	22.04	21.50	17.08	12.75	22.17	15.59	21.79	18.12	16.66	17.83	28.33	23.79	1966-01-01	1	1966	1
1967-01-01	6.46	4.46	6.50	3.21	6.67	3.79	11.38	3.83	7.71	9.08	10.67	20.91	1967-01-01	1	1967	1
1968-01-01	30.04	17.88	16.25	16.25	21.79	12.54	18.16	16.62	18.75	17.62	22.25	27.29	1968-01-01	1	1968	1
1969-01-01	6.13	1.63	5.41	1.08	2.54	1.00	8.50	2.42	4.58	6.34	9.17	16.71	1969-01-01	1	1969	1
1970-01-01	9.59	2.96	11.79	3.42	6.13	4.08	9.00	4.46	7.29	3.50	7.33	13.00	1970-01-01	1	1970	1
1971-01-01	3.71	0.79	4.71	0.17	1.42	1.04	4.63	0.75	1.54	1.08	4.21	9.54	1971-01-01	1	1971	1
1972-01-01	9.29	3.63	14.54	4.25	6.75	4.42	13.00	5.33	10.04	8.54	8.71	19.17	1972-01-01	1	1972	1
1973-01-01	16.50	15.92	14.62	7.41	8.29	11.21	13.54	7.79	10.46	10.79	13.37	9.71	1973-01-01	1	1973	1
1974-01-01	23.21	16.54	16.08	9.75	15.83	11.46	9.54	13.54	13.83	16.66	17.21	25.29	1974-01-01	1	1974	1
1975-01-01	14.04	13.54	11.29	5.46	12.58	5.58	8.12	8.96	9.29	5.17	7.71	11.63	1975-01-01	1	1975	1
1976-01-01	18.34	17.67	14.83	8.00	16.62	10.13	13.17	9.04	13.13	5.75	11.38	14.96	1976-01-01	1	1976	1
1977-01-01	20.04	11.92	20.25	9.13	9.29	8.04	10.75	5.88	9.00	9.00	14.88	25.70	1977-01-01	1	1977	1
1978-01-01	8.33	7.12	7.71	3.54	8.50	7.50	14.71	10.00	11.83	10.00	15.09	20.46	1978-01-01	1	1978	1

步骤13 对于数据记录按照月为频率取样

```
# 运行以下代码
data.query('day == 1')
```

out[303]:



	RPT	VAL	ROS	KIL	SHA	BIR	DUB	CLA	MUL	CLO	BEL	MAL	date	month	year	day
Yr_Mo_Dy																
1961-01-01	15.04	14.96	13.17	9.29	NaN	9.87	13.67	10.25	10.83	12.58	18.50	15.04	1961-01-01	1	1961	1
1961-02-01	14.25	15.12	9.04	5.88	12.08	7.17	10.17	3.63	6.50	5.50	9.17	8.00	1961-02-01	2	1961	1
1961-03-01	12.67	13.13	11.79	6.42	9.79	8.54	10.25	13.29	NaN	12.21	20.62	NaN	1961-03-01	3	1961	1
1961-04-01	8.38	6.34	8.33	6.75	9.33	9.54	11.67	8.21	11.21	6.46	11.96	7.17	1961-04-01	4	1961	1
1961-05-01	15.87	13.88	15.37	9.79	13.46	10.17	9.96	14.04	9.75	9.92	18.63	11.12	1961-05-01	5	1961	1
1961-06-01	15.92	9.59	12.04	8.79	11.54	6.04	9.75	8.29	9.33	10.34	10.67	12.12	1961-06-01	6	1961	1
1961-07-01	7.21	6.83	7.71	4.42	8.46	4.79	6.71	6.00	5.79	7.96	6.96	8.71	1961-07-01	7	1961	1
1961-08-01	9.59	5.09	5.54	4.63	8.29	5.25	4.21	5.25	5.37	5.41	8.38	9.08	1961-08-01	8	1961	1
1961-09-01	5.58	1.13	4.96	3.04	4.25	2.25	4.63	2.71	3.67	6.00	4.79	5.41	1961-09-01	9	1961	1
1961-10-01	14.25	12.87	7.87	8.00	13.00	7.75	5.83	9.00	7.08	5.29	11.79	4.04	1961-10-01	10	1961	1
1961-11-01	13.21	13.13	14.33	8.54	12.17	10.21	13.08	12.17	10.92	13.54	20.17	20.04	1961-11-01	11	1961	1
1961-12-01	9.67	7.75	8.00	3.96	6.00	2.75	7.25	2.50	5.58	5.58	7.79	11.17	1961-12-01	12	1961	1
1962-01-01	9.29	3.42	11.54	3.50	2.21	1.96	10.41	2.79	3.54	5.17	4.38	7.92	1962-01-01	1	1962	1

练习7-可视化

探索泰坦尼克灾难数据

相应数据集：train.csv

步骤1 导入必要的库

```
# 运行以下代码
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

%matplotlib inline
```

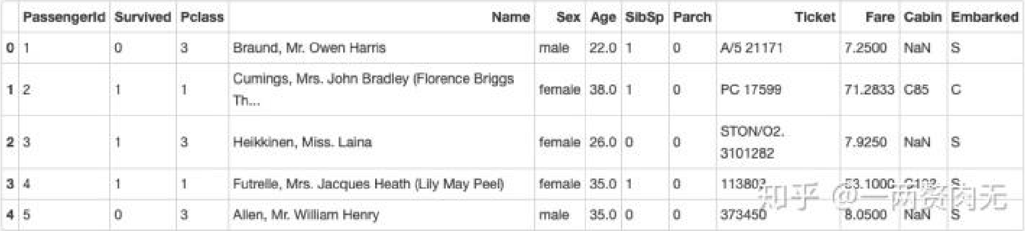
步骤2 从以下地址导入数据

```
# 运行以下代码
path7 = '../input/pandas_exercise/exercise_data/train.csv'  # train.csv
```

步骤3 将数据框命名为titanic

```
# 运行以下代码
titanic = pd.read_csv(path7)
titanic.head()
```

out[306]:

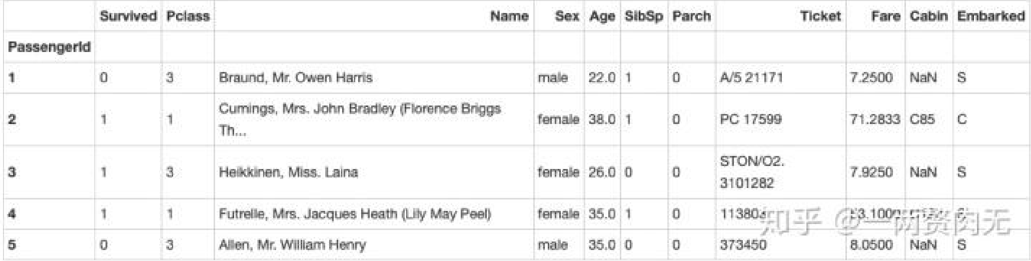


	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

步骤4 将PassengerId设置为索引

```
# 运行以下代码
titanic.set_index('PassengerId').head()
```

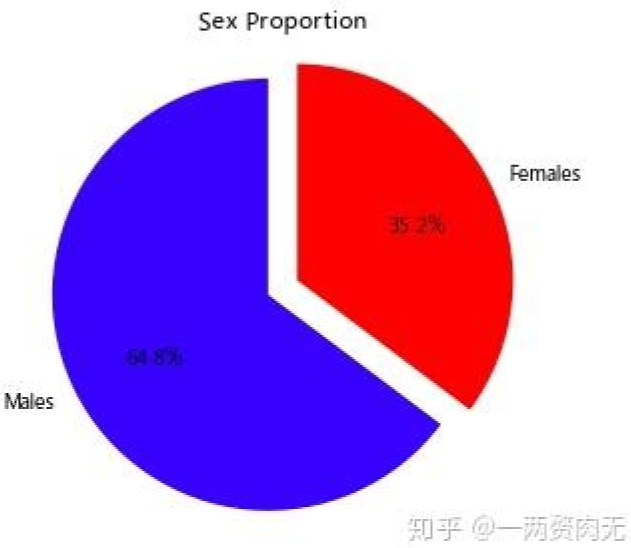
out[307]:



	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId											
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

步骤5 绘制一个展示男女乘客比例的扇形图

```
# 运行以下代码
# sum the instances of males and females
males = (titanic['Sex'] == 'male').sum()
females = (titanic['Sex'] == 'female').sum()

# put them into a list called proportions
proportions = [males, females]

# Create a pie chart
plt.pie(
    # using proportions
    proportions,

    # with the labels being officer names
    labels = ['Males', 'Females'],
        # with no shadows
        shadow = False,

        # with colors
        colors = ['blue','red'],

        # with one slide exploded out
        explode = (0.15 , 0),

        # with the start angle at 90%
        startangle = 90,

        # with the percent listed as a fraction
        autopct = '%1.1f%%'
        )

# View the plot drop above
plt.axis('equal')

# Set labels
plt.title("Sex Proportion")

# View the plot
plt.tight_layout()
plt.show()
```

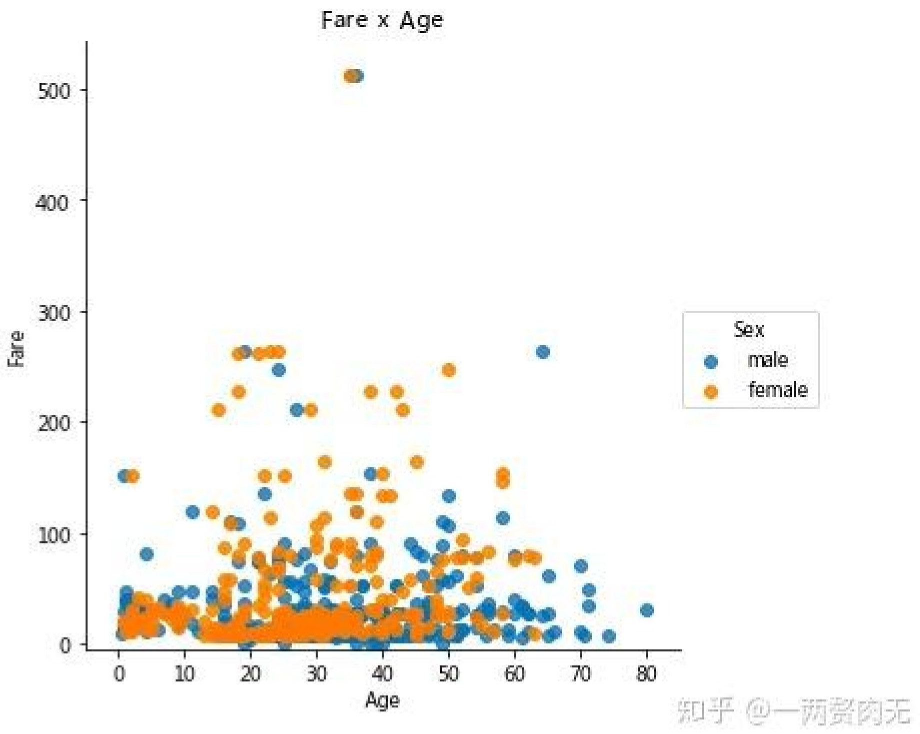


步骤6 绘制一个展示船票Fare, 与乘客年龄和性别的散点图

```
# 运行以下代码
# creates the plot using
lm = sns.lmplot(x = 'Age', y = 'Fare', data = titanic, hue = 'Sex',
fit_reg=False)

# set title
lm.set(title = 'Fare x Age')

# get the axes object and tweak it
axes = lm.axes
axes[0,0].set_ylim(-5,)
axes[0,0].set_xlim(-5,85)
```

out[309]:

(-5, 85)



步骤7 有多少人生还?

```
# 运行以下代码
titanic.Survived.sum()
```

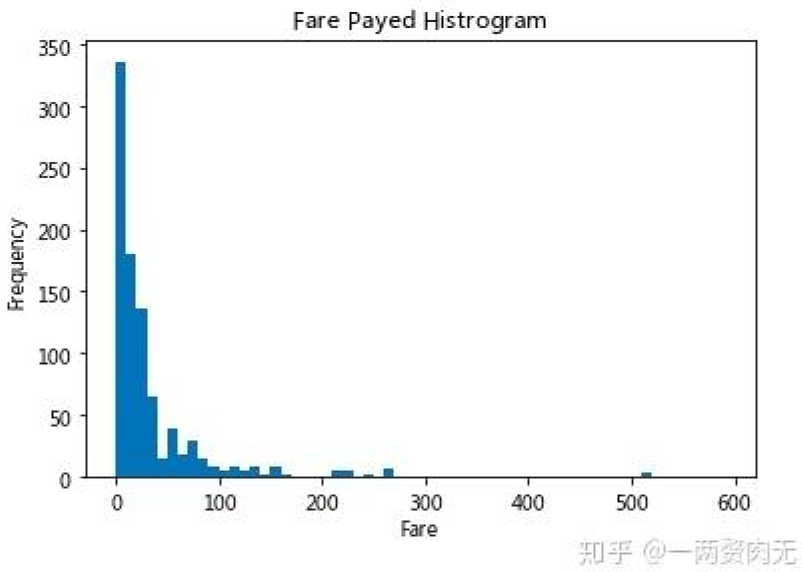
out[310]:

```
342
```

步骤8 绘制一个展示船票价格的直方图

```
# 运行以下代码
# sort the values from the top to the least value and slice the first 5 items
df = titanic.Fare.sort_values(ascending = False)
df

# create bins interval using numpy
binsVal = np.arange(0,600,10)
binsVal

# create the plot
plt.hist(df, bins = binsVal)
```

```
# Set the title and labels

plt.xlabel('Fare')

plt.ylabel('Frequency')

plt.title('Fare Payed Histrogram')


# show the plot

plt.show()
```



练习8-创建数据框

探索Pokemon数据

相应数据集：练习中手动内置的数据

步骤1 导入必要的库

```
# 运行以下代码
import pandas as pd
```

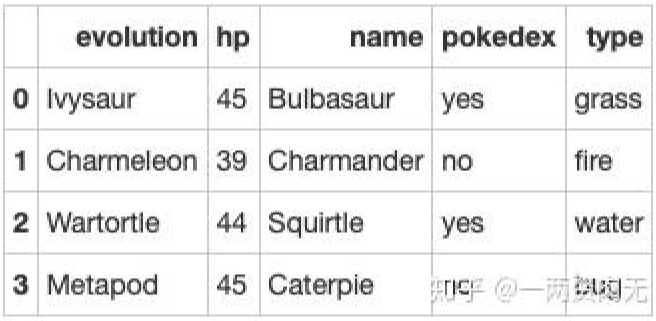
步骤2 创建一个数据字典

```
# 运行以下代码
raw_data = {"name": ['Bulbasaur', 'Charmander','Squirtle','Caterpie'],
            "evolution": ['Ivysaur','Charmeleon','Wartortle','Metapod'],
            "type": ['grass', 'fire', 'water', 'bug'],
            "hp": [45, 39, 44, 45],
            "pokedex": ['yes', 'no','yes','no']
            }
```

步骤3 将数据字典存为一个名叫pokemon的数据框中

```
# 运行以下代码
pokemon = pd.DataFrame(raw_data)
pokemon.head()
```

out[314]:

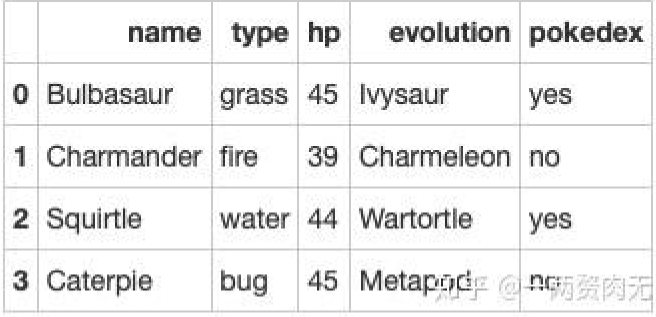


	evolution	hp	name	pokedex	type
0	Ivysaur	45	Bulbasaur	yes	grass
1	Charmeleon	39	Charmander	no	fire
2	Wartortle	44	Squirtle	yes	water
3	Metapod	45	Caterpie	no	bug

步骤4 数据框的列排序是字母顺序，请重新修改为name，type，hp，evolution，pokedex这个顺序

```
# 运行以下代码
pokemon = pokemon[['name', 'type', 'hp', 'evolution','pokedex']]
pokemon
```

out[315]:

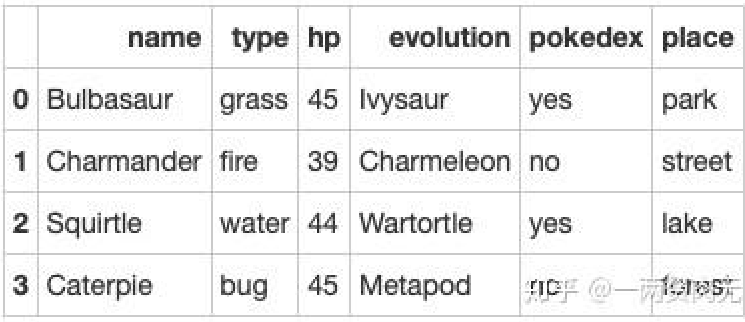


	name	type	hp	evolution	pokedex
0	Bulbasaur	grass	45	Ivysaur	yes
1	Charmander	fire	39	Charmeleon	no
2	Squirtle	water	44	Wartortle	yes
3	Caterpie	bug	45	Metapod	no

步骤5 添加一个列`place`

```
# 运行以下代码
pokemon['place'] = ['park','street','lake','forest']
pokemon
```

out[316]:



	name	type	hp	evolution	pokedex	place
0	Bulbasaur	grass	45	Ivysaur	yes	park
1	Charmander	fire	39	Charmeleon	no	street
2	Squirtle	water	44	Wartortle	yes	lake
3	Caterpie	bug	45	Metapod	no	forest

步骤6 查看每个列的数据类型

out[317]:

```
name object
type object
hp int64
evolution object
pokedex object
place object
dtype: object
```

练习9-时间序列

探索Apple公司股价数据

相应数据集：Apple_stock.csv

步骤1 导入必要的库

```
# 运行以下代码
import pandas as pd
import numpy as np

# visualization
import matplotlib.pyplot as plt

%matplotlib inline
```

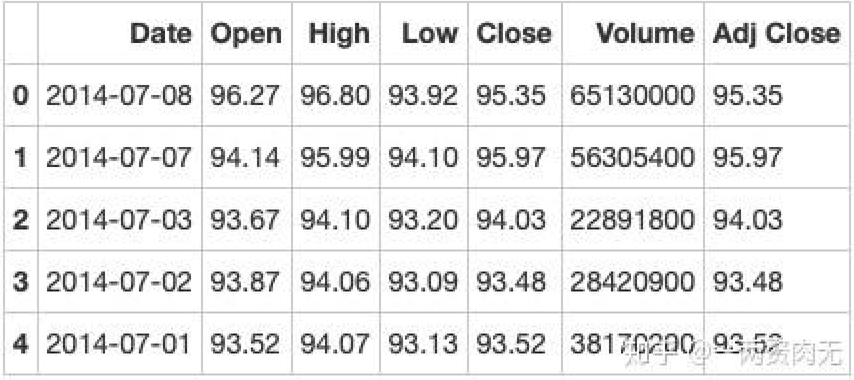
步骤2 数据集地址

```
# 运行以下代码
path9 = '../input/pandas_exercise/exercise_data/Apple_stock.csv'   #
Apple_stock.csv
```

步骤3 读取数据并存为一个名叫apple的数据框

```
# 运行以下代码
apple = pd.read_csv(path9)
apple.head()
```

out[320]:



	Date	Open	High	Low	Close	Volume	Adj Close
0	2014-07-08	96.27	96.80	93.92	95.35	65130000	95.35
1	2014-07-07	94.14	95.99	94.10	95.97	56305400	95.97
2	2014-07-03	93.67	94.10	93.20	94.03	22891800	94.03
3	2014-07-02	93.87	94.06	93.09	93.48	28420900	93.48
4	2014-07-01	93.52	94.07	93.13	93.52	38170200	93.52

步骤4 查看每一列的数据类型

out[321]:

```
Date object
Open float64
High float64
Low float64
Close float64
Volume int64
Adj Close float64
dtype: object
```

步骤5 将Date这个列转换为datetime类型

```
# 运行以下代码
apple.Date = pd.to_datetime(apple.Date)
apple['Date'].head()
```

out[322]:

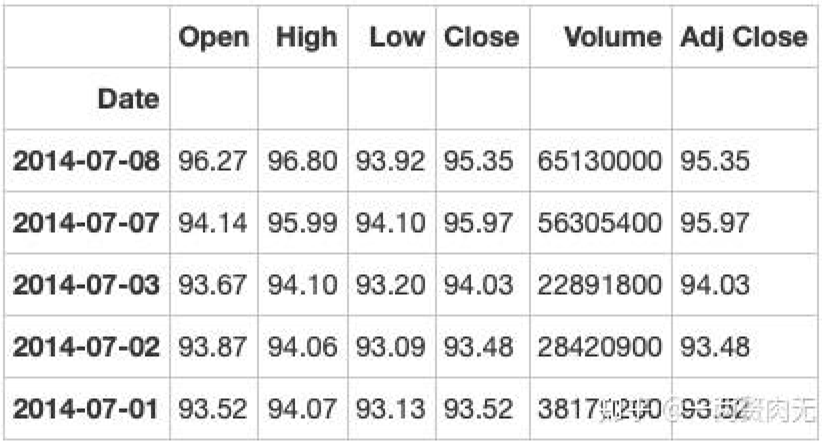
```
0 2014-07-08
1 2014-07-07
2 2014-07-03
3 2014-07-02
4 2014-07-01
Name: Date, dtype: datetime64[ns]
```

步骤6 将Date设置为索引

```
# 运行以下代码
apple = apple.set_index('Date')
```

```
apple.head()
```

out[323]:



	Open	High	Low	Close	Volume	Adj Close
Date						
2014-07-08	96.27	96.80	93.92	95.35	65130000	95.35
2014-07-07	94.14	95.99	94.10	95.97	56305400	95.97
2014-07-03	93.67	94.10	93.20	94.03	22891800	94.03
2014-07-02	93.87	94.06	93.09	93.48	28420900	93.48
2014-07-01	93.52	94.07	93.13	93.52	38170200	93.52

步骤7 有重复的日期吗?

```
# 运行以下代码
apple.index.is_unique
```

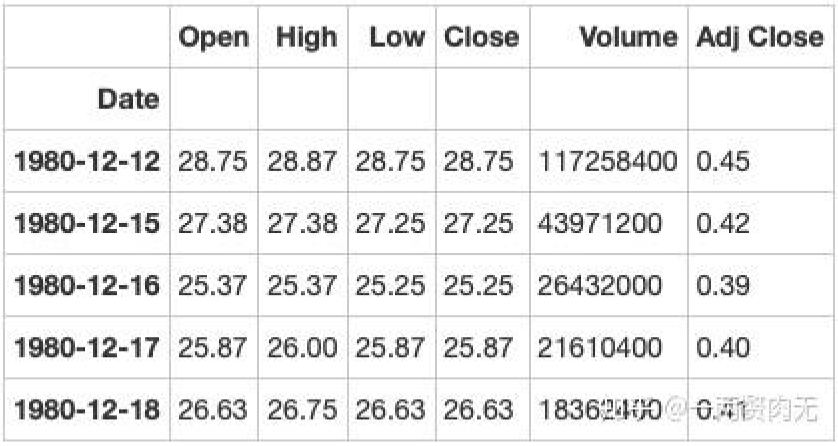
out[324]:

True

步骤8 将index设置为升序

```
# 运行以下代码
apple.sort_index(ascending = True).head()
```

out[325]:

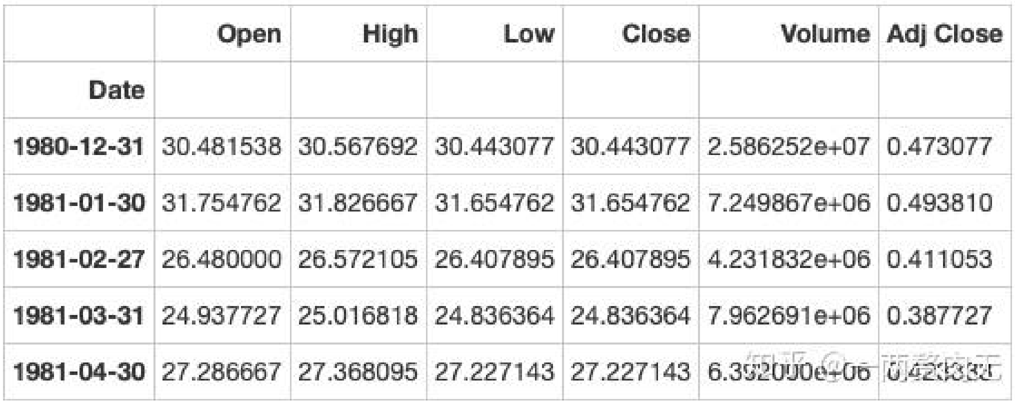


	Open	High	Low	Close	Volume	Adj Close
Date						
1980-12-12	28.75	28.87	28.75	28.75	117258400	0.45
1980-12-15	27.38	27.38	27.25	27.25	43971200	0.42
1980-12-16	25.37	25.37	25.25	25.25	26432000	0.39
1980-12-17	25.87	26.00	25.87	25.87	21610400	0.40
1980-12-18	26.63	26.75	26.63	26.63	18362400	0.41

步骤9 找到每个月的最后一个交易日(business day)

```
# 运行以下代码
apple_month = apple.resample('BM')
apple_month.head()
```

out[326]:



	Open	High	Low	Close	Volume	Adj Close
Date						
1980-12-31	30.481538	30.567692	30.443077	30.443077	2.586252e+07	0.473077
1981-01-30	31.754762	31.826667	31.654762	31.654762	7.249867e+06	0.493810
1981-02-27	26.480000	26.572105	26.407895	26.407895	4.231832e+06	0.411053
1981-03-31	24.937727	25.016818	24.836364	24.836364	7.962691e+06	0.387727
1981-04-30	27.286667	27.368095	27.227143	27.227143	6.392000e+06	0.423333

步骤10 数据集中最早的日期和最晚的日期相差多少天?

```
# 运行以下代码
(apple.index.max() - apple.index.min()).days
```

out[327]:

12261

步骤11 在数据中一共有多少个月?

```
# 运行以下代码
apple_months = apple.resample('BM').mean()
len(apple_months.index)
```

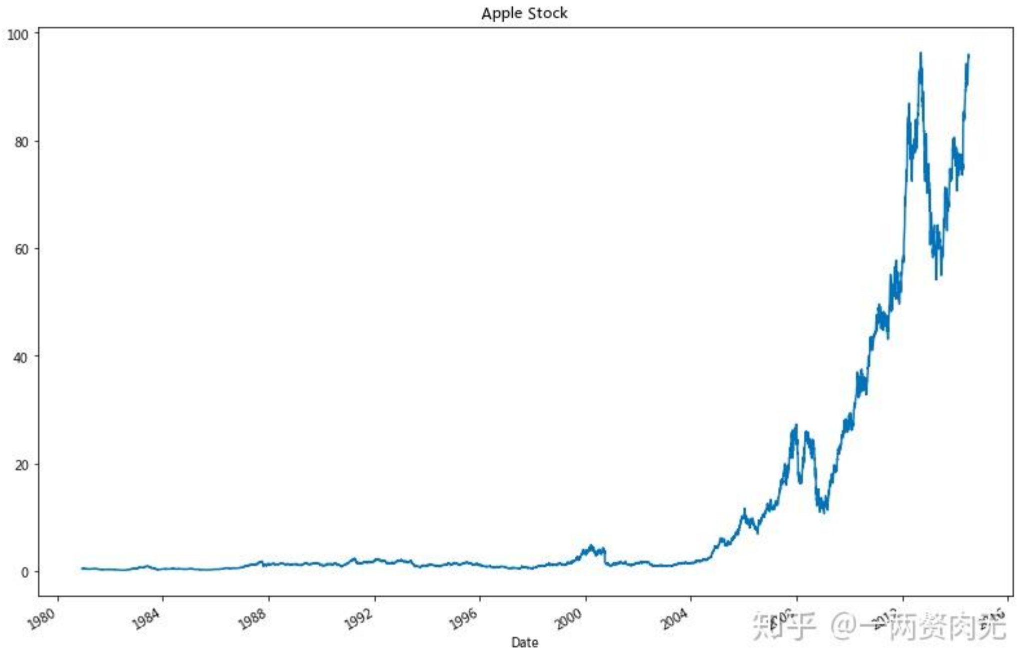
out[328]:

404

步骤12 按照时间顺序可视化Adj Close值

```
# 运行以下代码
# makes the plot and assign it to a variable
appl_open = apple['Adj Close'].plot(title = "Apple Stock")


# changes the size of the graph
```

```
fig = appl_open.get_figure()

fig.set_size_inches(13.5, 9)
```



练习10-删除数据

探索Iris纸鸢花数据

相应数据集：iris.csv

步骤1 导入必要的库

```
# 运行以下代码

import pandas as pd
```

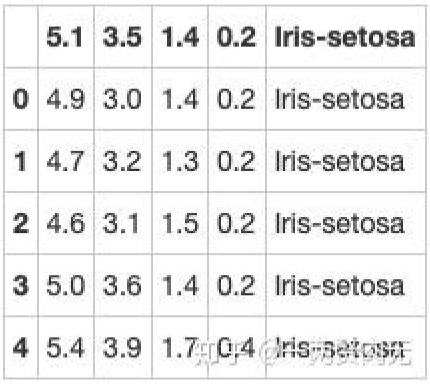
步骤2 数据集地址

```
# 运行以下代码

path10 ='../input/pandas_exercise/exercise_data/iris.csv'   # iris.csv
```

步骤3 将数据集存成变量`iris`

```
# 运行以下代码

iris = pd.read_csv(path10)

iris.head()
```

out[332]:

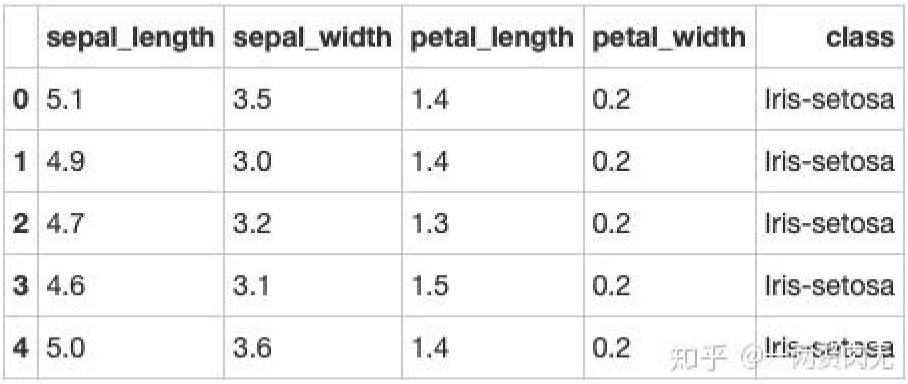


	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

步骤4 创建数据框的列名称

```
iris = pd.read_csv(path10,names = ['sepal_length','sepal_width', 'petal_length',
'petal_width', 'class'])
iris.head()
```

out[333]:



	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

步骤5 数据框中有缺失值吗?

```
# 运行以下代码
pd.isnull(iris).sum()
```

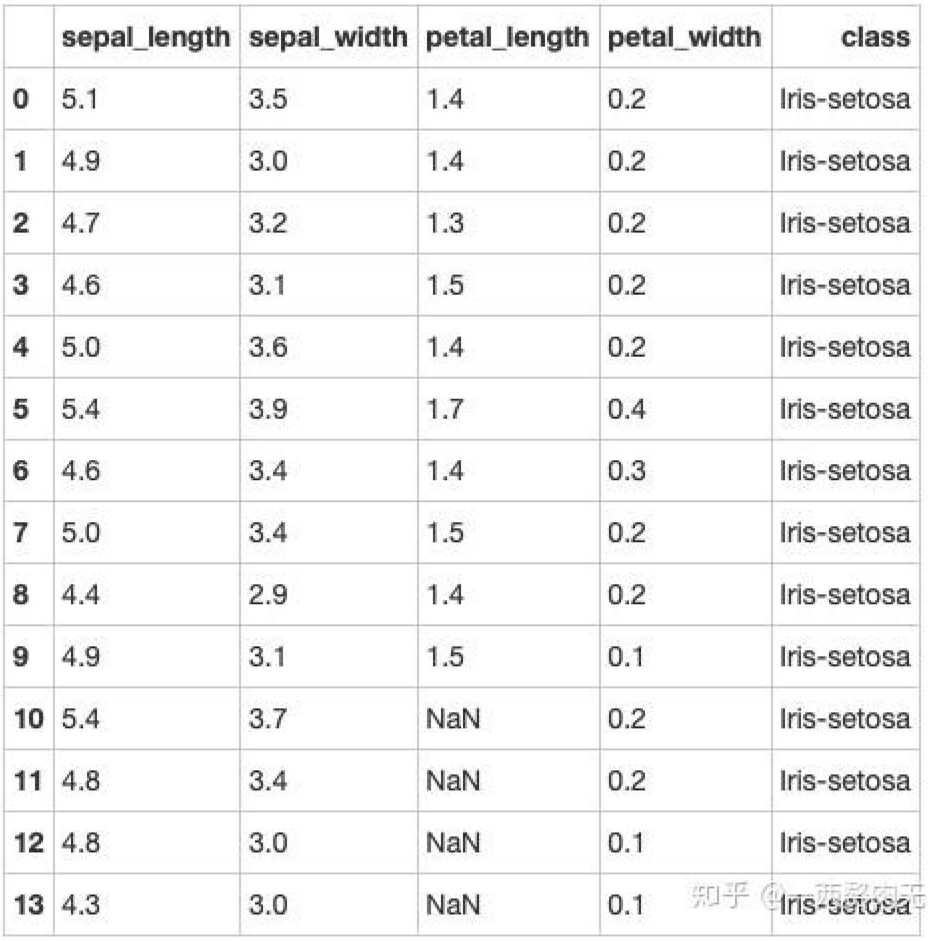
out[334]:

```
sepal_length 0
sepal_width 0
petal_length 0
petal_width 0
class 0
dtype: int64
```

步骤6 将列`petal_length`的第10到19行设置为缺失值

```
# 运行以下代码
iris.iloc[10:20,2:3] = np.nan
iris.head(20)
```

out[335]:

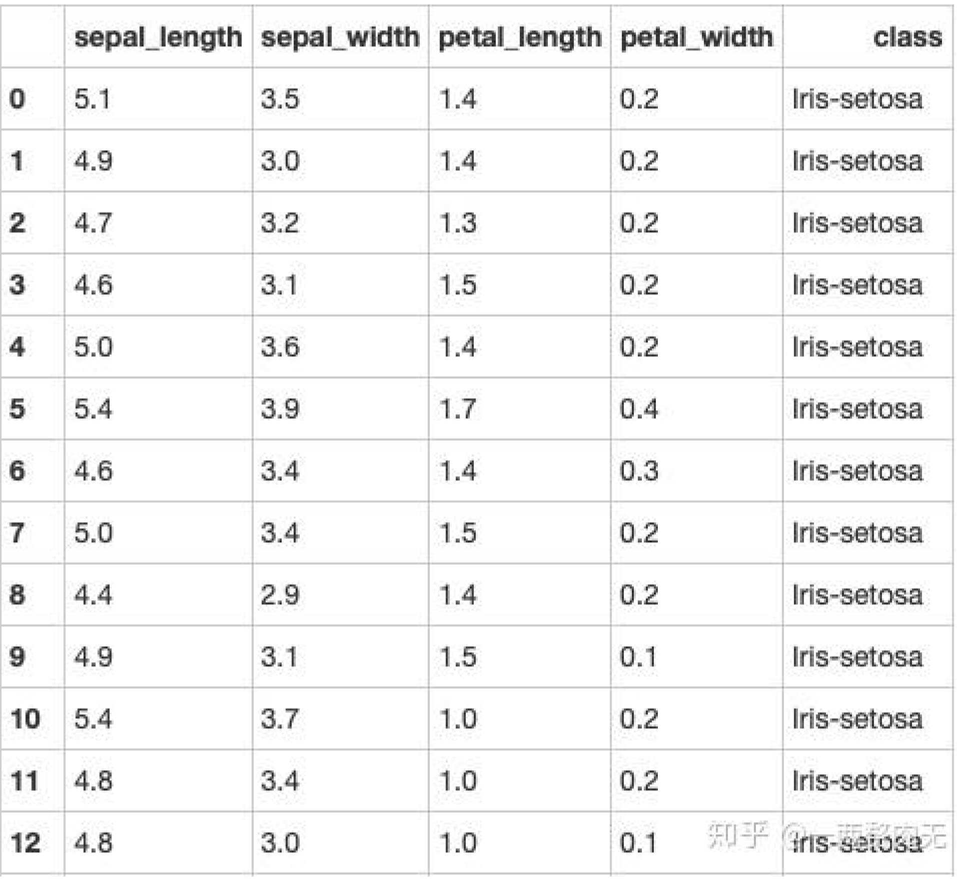


	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	NaN	0.2	Iris-setosa
11	4.8	3.4	NaN	0.2	Iris-setosa
12	4.8	3.0	NaN	0.1	Iris-setosa
13	4.3	3.0	NaN	0.1	Iris-setosa

步骤7 将缺失值全部替换为1.0

```
# 运行以下代码
iris.petal_length.fillna(1, inplace = True)
iris
```

out[336]:

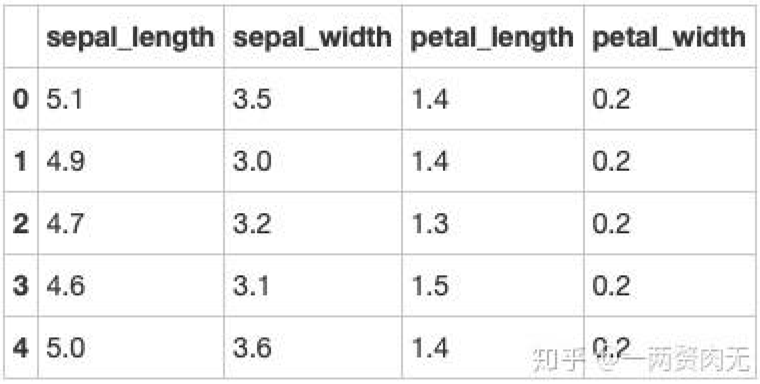


	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.0	0.2	Iris-setosa
11	4.8	3.4	1.0	0.2	Iris-setosa
12	4.8	3.0	1.0	0.1	Iris-setosa

步骤8 删除列`class`

```
# 运行以下代码
del iris['class']
iris.head()
```

out[337]:

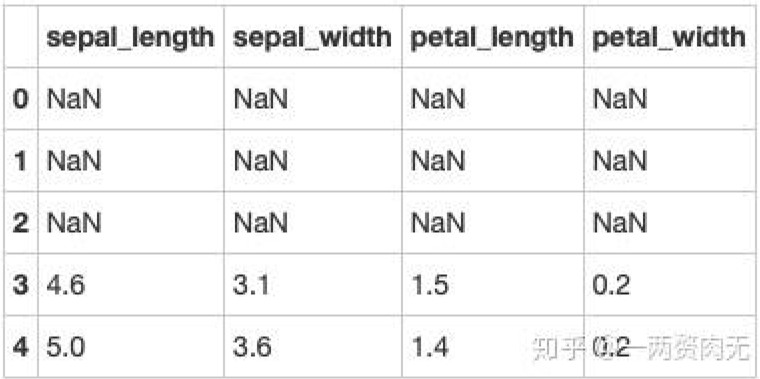


	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

步骤9 将数据框前三行设置为缺失值

```
# 运行以下代码
iris.iloc[0:3 ,:] = np.nan
iris.head()
```

out[338]:

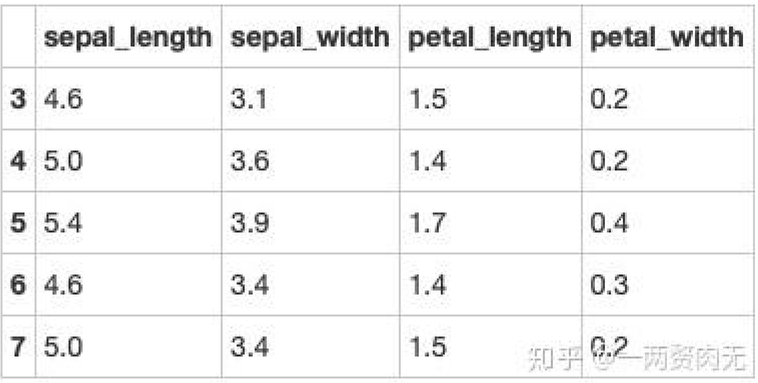


	sepal_length	sepal_width	petal_length	petal_width
0	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

步骤10 删除有缺失值的行

```
# 运行以下代码
iris = iris.dropna(how='any')
iris.head()
```

out[339]:

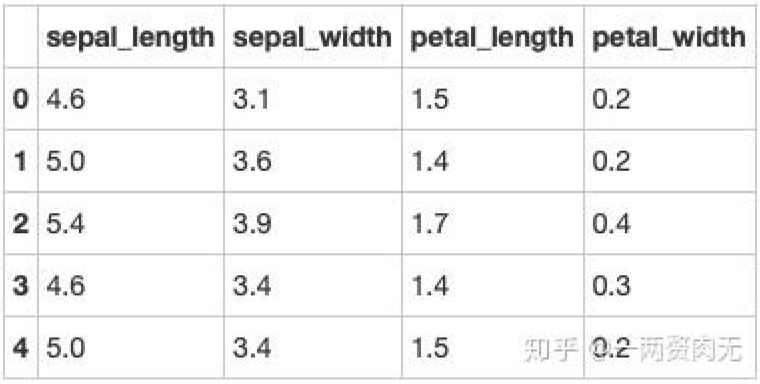


	sepal_length	sepal_width	petal_length	petal_width
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2

步骤11 重新设置索引

```
# 运行以下代码
iris = iris.reset_index(drop = True)
iris.head()
```

out[340]:



	sepal_length	sepal_width	petal_length	petal_width
0	4.6	3.1	1.5	0.2
1	5.0	3.6	1.4	0.2
2	5.4	3.9	1.7	0.4
3	4.6	3.4	1.4	0.3
4	5.0	3.4	1.5	0.2

转载本文请联系和鲸社区取得授权。