任务3.1 从csv文件中读取餐饮订单数据

目录

3.1 从csv文件中读取餐饮订单数据

【任务描述】

数据读取是进行数据预处理、分析与建模的前提。对于不同的数据源pandas提供了不同的函数进行读取。Pandas提供了10多种不同数据源的读取函数和对应的数据源写入函数。我们这里掌握两种常见的数据源数据的读取/写入函数。分别是文本文件(包括文本文件和CSV文件)、Excel文件。这里以餐饮信息数据为例，方便学习。

【知识准备】

- ***1.了解pandas库***

- pandas是Python的核心数据分析支持库，提供了快速、灵活、明确的数据结构，旨在简单、直观地处理关系型、标记型数据。同时由于pandas建造在Numpy之上，所以使得pandas在以Numpy为中心的应用中得以容易的使用，而pandas库在与其它第三方科学计算支持库结合时也能够完美的进行集成。

- 在Python中，pandas库的功能十分强大，它可提供高性能的矩阵运算。也可用于数据挖掘和数据分析，同时也提供数据清洗功能；支持类似SQL的数据增、删、查、改，并且带有丰富的数据处理函数；支持时间序列分析功能；支持灵活处理缺失数据等。

- pandas有两个常用的数据结构 。使用这两个数据结构已经能够处理大多数的统计分析问题。

- （1）Series（一维数据）是一种类似于一维数组的对象，是由一组数据（各种Numpy数据类型）以及一组与之相关的数据标签（即索引）组成，而仅由一组数据也可产生简单的Series对象。

- （2）DataFrame是pandas中的一个表格型的数据结构，包含有一组有序的列，每列可以是不同的值类型（数值、字符串、布尔型等），DataFrame既有行索引也有列索引，可以被看做是由Series组成的字典。

【知识准备】

- ***2.读/写文本文件***

- 文本文件是一种由若干行字符构成的计算机文件，它是一种典型的顺序文件。

- CSV是一种用分隔符分隔的文件格式，因为其分隔符不一定是逗号，所以又被称为字符分隔文件。

- 文本文件以纯文本形式存储表格数据（数字和文本），它是一种通用、相对简单的文件格式，最广泛的应用是在程序之间转移表格数据，而这些程序本身是在不兼容的格式上进行操作的。

- 由于大量程序都支持CSV或其变体，所以CSV或其变体可以作为大多数程序的输入和输出格式。

【任务实施】

- ***1文本文件读取***

- CSV文件根据其定义也是一种文本文件。在数据读取过程中可以使用文本文件的读取函数对CSV文件进行读取。同时，如果文本文件是字符分隔文件，那么可以使用读取CSV文件的函数进行读取。

- pandas提供了read_table()函数读取文本文件，提供了read_csv()函数读取CSV文件。两个函数的基本使用格式（部分参数）如下：

```
pandas.read_table(filepath, sep=<no_default>,
 header='infer', encoding='gbk',names=<no_default>,
 index_col=None, dtype=None,  engine=None,
 nrows=None, skiprows=None)
pandas.read_csv(filepath, sep=<no_default>,
 header='infer', encoding='utf-
 8',names=<no_default>, index_col=None, dtype=None,
 engine=None, nrows=None, skiprows=None)
```

【任务实施】

- read_table()函数和read_csv()函数的许多数参数相同，它们的常用参数及其说明如表3-1所示。

表3-1 read_table()函数和read_csv()函数的常用参数说明

参数名称	用途说明
filepath	接收str。表示文件路径。无默认值。
sep	接收str。表示分隔符。read_csv函数默认为“,”，read_table函数默认为制表符“Tab”
names	接收array。表示列名。无默认值
index_col	接收int、sequence或False。表示索引列的位置，取值为sequence则代表多重索引。默认为None。
dtype	接收字典形式的列名或type name。表示写入的数据类型（列名为key，数据格式为values）。默认为None
nrows	接收int。要读取的文件行数。默认为None
encoding	接收文件的编码格式。常用有utf-8,gbk。

【任务实施】

- 以某餐饮公司的信息表为例子，其主要的数据特征为 ：detail_id，order_id，dishes_id，logicprn_name，dishes_name，counts，amounts，place_order_time等，部分信息如图3-1所示。

detail_id	order_id	dishes_id	logicprn_n	parent_clas	dishes_nar	itemis_add	counts	amounts	cost	place_order_time
2956	417	610062	NA	NA	蒜蓉生蚝	0	1	49	NA	2016/8/1 11:05
2958	417	609957	NA	NA	蒙古烤羊服	0	1	48	NA	2016/8/1 11:07
2961	417	609950	NA	NA	大蒜苋菜	0	1	30	NA	2016/8/1 11:07
2966	417	610038	NA	NA	芝麻烤紫薯	0	1	25	NA	2016/8/1 11:11
2968	417	610003	NA	NA	蒜香包	0	1	13	NA	2016/8/1 11:11
1899	301	610019	NA	NA	白斩鸡	0	1	88	NA	2016/8/1 11:15
1902	301	609991	NA	NA	香烤牛排	0	1	55	NA	2016/8/1 11:19
1906	301	609983	NA	NA	干锅田鸡	0	1	88	NA	2016/8/1 11:22
1907	301	609981	NA	NA	桂圆枸杞含	0	1	48	NA	2016/8/1 11:22
1908	301	610030	NA	NA	番茄有机花	0	1	32	NA	2016/8/1 11:23
1910	301	610011	NA	NA	白饭/大碗	0	1	10	NA	2016/8/1 11:31
2916	413	609966	NA	NA	芝士烩波士	0	1	175	NA	2016/8/1 12:42
2919	413	609965	NA	NA	葱姜炒蟹	0	1	109	NA	2016/8/1 12:43
2921	413	609936	NA	NA	皮蛋瘦肉粥	0	1	16	NA	2016/8/1 12:43
2923	413	609978	NA	NA	爆炒鳝碌	0	1	55	NA	2016/8/1 12:44
2925	413	609983	NA	NA	干锅田鸡	0	1	88	NA	2016/8/1 12:44
2927	413	610050	NA	NA	番茄甘蓝	0	1	33	NA	2016/8/1 12:45
2926	413	609984	NA	NA	重庆特色泡	0	1	69	NA	2016/8/1 12:45
2928	413	610013	NA	NA	番茄炖秋葵	0	1	35	NA	2016/8/1 12:46
2930	413	610032	NA	NA	长城窖酿角	0	1	35	NA	2016/8/1 12:48
2932	413	609973	NA	NA	紫薯面包卷	0	1	20	NA	2016/8/1 12:49

图3-1 某餐饮店订单的信息

【任务实施】

根据餐饮店订单信息表，分别使用read_table和read_csv函数对其读取数据，

代码如下所示：

```
In[1]:import pandas as pd
# 使用read_table函数读取餐饮订单信息表
order = pd.read_table('D:\Anaconda\order1.csv', sep=',', encoding='gbk')
print('使用read_table函数读取餐饮订单信息表的长度为：', len(order))
# 使用read_csv函数读取餐饮订单信息表
order1 = pd.read_csv('D:\Anaconda\order1.csv', encoding='gbk')
print('使用read_csv函数读取餐饮订单信息表的长度为：', len(order1))
Out[1]: 使用read_table函数读取餐饮订单信息表的长度为：  2779
使用read_csv函数读取餐饮订单信息表的长度为：  2779
```

【任务实施】

read_table()函数和read_csv()函数应注意。

- sep参数是指定文本的分隔符，如果分隔符指定错误，那么在读取数据的时候，每一行数据将连成一片。
- header参数用于指定列名，如果header参数值是None，那么将会添加一个默认的列名。
- encoding代表文件的编码格式，常用的编码格式有UTF-8、UTF-16、GBK、GB 2312、GB 18030等。如果编码指定错误，那么数据将无法读取，IPython解释器会报解析错误。数据将无法读取。改变read_table函数和read_csb中的参数读取餐饮订单信息表具体代码**如下所示：**

【任务实施】

```
In[2]:order2 = pd.read_table('D:\Anaconda\order1.csv', sep=';',
 encoding='gbk')
print('使用read_table函数读取餐饮订单信息表的长度为：', order2)

# 使用read_csv函数读取餐饮订单信息表
order3 = pd.read_csv('D:\Anaconda\order1.csv',
 sep=',',header=None,encoding='gbk')
print('使用read_csv函数读取餐饮订单信息表的长度为：', order3)
```

【任务实施】

```
Out[2]: 使用read_table函数读取餐饮订单信息表的长度为:
 detail_id,order_id,dishes_id,logicprn_name,parent_class_name,dishes_name,
 itemis_add,counts,amounts,cost,place_order_time,discount_amt,discount_rea
 son,kick_back,add_inprice,add_info,bar_code,picture_file,emp_id
0     2956,417,610062,NA,NA,蒜蓉生蚝,0,1,49,NA,2016/8/1 ...
1     2958,417,609957,NA,NA,蒙古烤羊腿,0,1,48,NA,2016/8/1...

2     2966,417,610038,NA,NA,芝麻烤紫菜,0,1,25,NA,2016/8/1...
3     2968,417,610003,NA,NA,蒜香包,0,1,13,NA,2016/8/1 1...
...                 ...                              ...
2774 6750,774,610011,NA,NA,白饭/大碗,0,1,10,NA,2016/8/1...
2775 6742,774,609996,NA,NA,牛尾汤,0,1,40,NA,2016/8/10 ...
2776 6756,774,609949,NA,NA,意文柠檬汁 ,0,1,13,NA,2016/8/...

...                 ...                              ...
[2779  rows x 1 columns]
```

【任务实施】

使用read_csv函数读取餐饮订单信息表的长度为：

```
0     detail_id  order_id  dishes_id  logicprn_name  parent_class_name
1          2956       417     610062            NaN                NaN
2          2958       417     609957            NaN                NaN
3          2961       417     609950            NaN                NaN
4          2966       417     610038            NaN                NaN
...         ...       ...        ...            ...                ...
2775       6750       774     610011            NaN                NaN
2776       6742       774     609996            NaN                NaN
2777       6756       774     609949            NaN                NaN
2778       6763       774     610014            NaN                NaN
2779       6764       774     610017            NaN                NaN
[2780 rows x 19 columns]
```

【任务实施】

2.文本文件的写入

文本文件存储的方法和文件读取的方法类似，对于结构化数据，可以通过pandas库中的to_csv()方法 实现以CSV文件格式存储。to_csv()方法同样具有许多参数，如果有多个< expression > ，那么表达式之间用逗号隔开，基本使用格式如下：

```
DataFrame.to_csv(path=None, sep=',', na_rep='', columns=None, header=True, index=True,
 index_label=None, mode='w', encoding=None)
```

【任务实施】

to_csv()方法的常用参数说明如表3-2所示。

表3-2 to_csv()方法的常用参数说明

参数名称	用途说明
Path	接收str。表示文件路径。默认为None
sep	接收str。表示分隔符。默认为“,”
na_rep	接收str。表示缺失值。默认为“”
header	接收bool或列表形式的str。表示是否将列名写出。
index	接收bool。表示是否将行名（索引）写出。默认为True
mode	接收特定str。表示数据写入模式。
encoding	接收特定str。表示存储文件的编码格式。默认为None

【任务实施】

使用to_csv（）方法的将餐饮订单信息表写入CSV文件，**代码如下所示：**

```
In[3]:import os
print('餐饮订单信息表写入文本文件前目录内文件列表为：\n',
 os.listdir('D:\Anaconda\csv'))
# 将order以csv格式存储
order.to_csv('D:\Anaconda\csv\orderInfo.csv', sep=';', index=False)
print('餐饮订单信息表写入文本文件后目录内文件列表为：\n',
 os.listdir('D:\Anaconda\csv'))
Out[3]: 餐饮订单信息表写入文本文件前目录内文件列表为：
 ['arr.txt', 'oedre1.xlsx', 'order.csv', 'savez_arr.npz', 'save_arr.npy']
餐饮订单信息表写入文本文件后目录内文件列表为：
 ['arr.txt', 'oedre1.xlsx', 'order.csv', 'orderInfo.csv',
 'savez_arr.npz', 'save_arr.npy']
```

【任务实施】

3.读取/存储Excel文件

Excel是微软公司的办公软件Microsoft Office的组件之一，它可以对数据进行处理、统计分析等操作，广泛地应用于管理、金融等各个领域。

读取Excel文件

pandas库提供了read_excel()函数读取“xls”和“xlsx”两种Excel文件，基本使用格式如下：

```
pandas.read_excel(path, sheet_name=0, header=0, names=None,
 index_col=None, dtype=None, skiprows=None)
```

read_excel()函数的常用参数及其说明如表3-3所示。

【任务实施】

表3-3 read_excel函数的常用参数说明

参数名称	用途说明
path	接收str。表示文件路径。无默认值
sheet_name	接收str、int、list或None。表示Excel表内数据的分表位置。默认为0
header	接收int或列表形式的int。表示将某行数据作为列名。如果传递整数列表，那么行位置将合并为MultiIndex。如果没有表头，那么使用None。
names	接收array。表示要使用的列名列表。默认为None
index_col	接收int或列表形式的int。表示将列索引用作DataFrame 的行索引。默认为None
dtype	接收dict。表示写入的数据类型（列名为key，数据格式为values）。默认为None
skiprows	接收list、int或callable。表示读取数据开头跳过的行数。默认为None

【任务实施】

使用read_excel()函数读取餐饮订单信息表中的数据，代码如下所示。

```
In[4]order = pd.read_excel('D:\Anaconda/order1.xlsx')
print('餐饮订单信息表长度为：', len(order))
Out[4]: 餐饮订单信息表长度为： 2779
```

【任务实施】

写入Excel文件

将数据写入到excel文件，方法和前面写入CSV文件类似，可以使用to_excel()方法，其基本格式如下：

```
DataFrame.to_excel(path, sheet_name='Sheet1', na_rep='', columns=None, header=True,
 index=True, index_label=None, encoding=None)
```

To_excel()方法的常用参数及其说明如表3-4所示。表3-4 To_excel()方法的常用参数及其说明

参数名称	用途说明
Path	接收str。表示文件路径。无默认值
sheet_name	接收str。表示Excel文件中工作簿的名称。默认为Sheet1
na_rep	接收str。表示缺失值。默认为“”
columns	接收列表形式的str或sequence。表示写出的列名。
header	接收bool或列表形式的str。表示是否将列名写出。
index	接收bool。表示是否将行名（索引）写出。默认为True

【任务实施】

To_excel（）方法把餐饮订单信息表写入Excel文件中，代码如下所示：

```
In[5]print('餐饮订单信息表写入Excel文件前，目录内文件列表为：\n',
 os.listdir('D:\Anaconda\csv'))
order.to_excel('D:\Anaconda\csv/order.xlsx')
print('餐饮订单信息表写入Excel文件后，目录内文件列表为：\n',
 os.listdir('D:\Anaconda\csv')) Out[5]: 餐饮订单信息表写入Excel文件前，
 目录内文件列表为：
 ['arr.txt', 'oedre1.xlsx', 'order.csv', 'orderInfo.csv',
 'savez_arr.npz', 'save_arr.npy']
餐饮订单信息表写入Excel文件后，目录内文件列表为：
 ['arr.txt', 'oedre1.xlsx', 'order.csv', 'order.xlsx',
'orderInfo.csv', 'savez_arr.npz', 'save_arr.npy']
```

任务3.2 创建餐饮订单数据的DataFrame

目录

3.2 创建餐饮订单数据的DataFrame

【任务描述】

DataFrame是pandas常见的操作对象。在使用数据读取函数后，数据将以DataFrame数据结构存储在内存中。但此时并不能直接开始统计分析。需要使用DataFrame的属性与方法对数据进行了解才能进行统计分析。

【知识准备】

- *查看DataFrame的常用属性*
  - DataFrame的基础属性如下。
    - Values:可以获取元素;
    - Index:可以获取索引;
    - Column:可以获取列名;
    - Dtypes:可以获取类型 。

【知识准备】

- 分别查看餐饮订单信息表的4个基本属性的代码如下所示：

```
In[6]:import pandas as pd
order = pd.read_csv('D:\Anaconda\order1.csv', encoding='gbk')
print('餐饮订单信息表的索引为：', order.index)
print('餐饮订单信息表的元素为：\n', order.values)
print('餐饮订单信息表的列名为：\n', order.columns)
print('餐饮订单信息表的数据类型为：\n', order.dtypes)
Out[6]：餐饮订单信息表的索引为：  RangeIndex(start=0,
 stop=2779,step=1)
```

【知识准备】

餐饮订单信息表的所有值为：

```
[[2956 417 610062 … n ‘n ’caipu/104001.’ pg’ 1442]
 [2958 417 609957 ... n ‘n ’caipu/202003.’ pg’ 1442]
 [2961 417 609950 ... n ‘n ’caipu/303001.’ pg’ 1442]
 ...
 [6756 774 609949 ... n ‘n ’caipu/404005.’ pg’ 1138]
 [6763 774 610014 ... n ‘n ’caipu/302003.’ pg’ 1138]
 [6764 774 610017 ... n ‘n ’caipu/302006.’ pg’ 1138]]
```

餐饮订单信息表的列名为：

```
Index(['detail_id', 'order_id', 'dishes_id', 'logicprn_name',
       'parent_class_name', 'dishes_name', 'itemis_add',
    'counts', 'amounts',
    'cost', 'place_order_time', 'discount_amt',
'discount_reason',
       'kick_back', 'add_inprice', 'add_info', 'bar_code',
'picture_file',
       'emp_id'],
      dtype='object')
```

餐饮订单信息表的数据类型为：

```
detail_id              int64
order_id               int64
dishes_id              int64
logicprn_name        float64
parent_class_name    float64
dishes_name           object
itemis_add             int64
counts                 int64
amounts                int64
cost                 float64
…   …     …     …   …
picture_file
 object
emp_id
 int64
dtype: object
```

【知识准备】

- 除了上述4个基本属性，还可以通过size、ndim和shape属性获取DataFrame的元素个数、维度数和数据形状（行列数目），代码如下所示：

```
In[7]:# 查看DataFrame的元素个数、维度数、形状
print('餐饮订单信息表的元素个数为：', order.size)
print('餐饮订单信息表的维度数为：', order.ndim)
print('餐饮订单信息表的数据形状 为：', order.shape)
Out[7]: 餐饮订单信息表的元素个数为：  52801
餐饮订单信息表的维度数为：  2
餐饮订单信息表的形状为：  (2779, 19)
```

【任务实施】

- DataFrame的单列数据为一个Series。根据DataFrame的定义可知，DataFrame是一个带有标签的二维数组，每个标签相当于每一列的列名。只要以字典访问某一个key的值的方式使用对应的列名，即可实现单列数据的访问，除了使用字典访问内部数据的获取方式外，还能以访问属性的方式访问数据，代码如下所示：

```
In[8]:# 使用字典访问的方式取出order中的某一列
order_id = order['order_id']
print('餐饮订单信息表中的order_id的形状为：', order_id.shape)
Out[8]: 餐饮订单信息表中的order_id的形状为：  (2779,)
```

【任务实施】

- 以上两种方法均可以获得DataFrame中的某一列数据，但是使用访问属性的方法访问数据并不建议使用。由于在多数时候数据的列名为英文，以属性方式访问某一列的形式和DataFrame属性访问， 其方法和使用的格式相同，难免存在部分列名和pandas提供的方法相同，这时候将会引起程序混乱，也会使得代码晦涩难懂。

- 当访问DataFrame中某一列的某几行时，单独一列的DataFrame可以视为一个Series（另一种pandas提供的类，可以看作是只有一列的DataFrame），而访问一个Series 的方法和访问一个一维的ndarray的方法相同，代码如下所示：

```
In[9]:detail_id5 = order['detail_id'][:5]
print('餐饮订单信息表中的detail_id前5个元素为: \n', detail_id5)
Out[9]: 餐饮订单信息表中的detail_id前5个元素为:
0    2956
1    2958
2    2961
3    2966
4    2968
Name: detail_id, dtype: int64
```

【任务实施】

- 访问DataFrame多列数据时可以将多个列索引名称放入一个列表，同时，访问DataFrame多列数据中的多行数据和访问单列数据的多行数据的方法基本相同，代码如下所示：

```
In[10]:order_id_detail_id = order[['order_id', 'detail_id']][:5]
print('餐饮订单信息表中的order_id和detail_id前5个元素为: \n', order_id_detail_id)
Out[10]: 餐饮订单信息表中的order_id和detail_id前5个元素为:
    order_id  detail_id
0        417        2956
1        417        2958
2        417        2961
3        417        2966
4        417        2968
```

【任务实施】

- 如果只是需要访问DataFrame某几行数据，那么实现方式和上述的访问多列多行的方式相似，选择所有列，使用":"代替即可，代码如下所示：

```
In[11]:order5 = order[:][1:6]
print('餐饮订单信息表的1～6行元素为: \n', order5)
Out[11]: 餐饮订单信息表的1～6行元素为:
```

	detail_id	order_id	dishes_id	logicprn_name	parent_class_name	dishes_name	itemis_add	counts	amounts	cost	place_order_time
1	2958	417	609957	NaN	NaN	蒙古烤羊腿	0	1	48	NaN	2016/8/1 11:07
2	2961	417	609950	NaN	NaN	大蒜苋菜	0	1	30	NaN	2016/8/1 11:07
3	2966	417	610038	NaN	NaN	芝麻烤紫菜	0	1	25	NaN	2016/8/1 11:11
4	2968	417	610003	NaN	NaN	蒜香包	0	1	13	NaN	2016/8/1 11:11
5	1899	301	610019	NaN	NaN	白斩鸡	0	1	88	NaN	2016/8/1 11:15

【任务实施】

- 除了使用上述方法能够得到多行数据以外，通过DataFrame提供的方法head()和tail()也可以得到多行数据，但是用这两种方法得到的数据都是从开始或末尾获取的连续数据，代码如下所示：

```
In[12]:print('餐饮订单信息表中前5行数据 为:\n', order.head(5) )
print('餐饮订单信息表中后5行元素为: \n', order.tail(5) )
Out[12]:
餐饮订单信息表中前5行数据为:
    detail_id  order_id  dishes_id  logicprn_name  parent_class_name  \
0        2956       417     610062            NaN                NaN
1        2958       417     609957            NaN                NaN
2        2961       417     609950            NaN                NaN
3        2966       417     610038            NaN                NaN
4        2968       417     610003            NaN                NaN
   discount_amt  discount_reason  kick_back  add_inprice  add_info  bar_code
0           NaN              NaN        NaN            0       NaN       NaN
1           NaN              NaN        NaN            0       NaN       NaN
2           NaN              NaN        NaN            0       NaN       NaN
3           NaN              NaN        NaN            0       NaN       NaN
4           NaN              NaN        NaN            0       NaN       NaN
```

【任务实施】

餐饮订单信息表中后5行元素为：

	detail_id	order_id	dishes_id	logicprn_name	parent_class_name
2774	6750	774	610011	NaN	NaN
2775	6742	774	609996	NaN	NaN
2776	6756	774	609949	NaN	NaN
2777	6763	774	610014	NaN	NaN
2778	6764	774	610017	NaN	NaN

	discount_amt	discount_reason	kick_back	add_inprice	add_info	\
2774	NaN	NaN	NaN	0	NaN	
2775	NaN	NaN	NaN	0	NaN	
2776	NaN	NaN	NaN	0	NaN	
2777	NaN	NaN	NaN	0	NaN	
2778	NaN	NaN	NaN	0	NaN	

- 这里head()方法和tail()方法使用的都是默认参数，所以访问的是前、后5行。在方法后的“()”中输入访问行数，即可实现目标行数的查看。

任务3.3 利用行列索引查看餐饮订单数据的子集

目录

3.3 利用行列索引查看餐饮订单数据的子集

【任务描述】

DataFrame的数据查看与访问基本方法虽然能够基本满足数据查看要求，但是终究还是不够灵活。pandas提供了loc()和iloc()两种以行列索引的方法来实现数据访问。

【知识准备】

- loc()方法是针对DataFrame索引名称的切片方法，如果传入的不是索引名称，那么切片操作将无法执行。利用loc()方法，能够实现所有单层索引切片操作。loc()方法基本使用格式如下：

```
DataFrame.loc[行索引名称或条件, 列索引名称]
```

- loc()方法的使用如下代码所示。

```
In[13]:order_id1 = order.loc[:, 'order_id']
print('使用loc()方法提取order_id列的size为: ', order_id1.size)
Out[13]: 使用loc()方法提取order_id列的size为:  2779
```

【知识准备】

- iloc()方法和local（）方法的区别在于，iloc()方法接收的必须是行索引和列索引的位置。 iloc()方法基本使用格式如下：

```
DataFrame.iloc[行索引位置, 列索引位置]
```

- iloc()方法的使用如下代码所示。

```
In[14]:order_id2 = order.iloc[:, 3]
print('使用iloc()方法提取第3列的size为: ',
order_id2.size)
Out[14]: 使用iloc()方法提取第3列的size为:  2779
```

【任务实施】

- 使用loc()方法和iloc()方法可以对DataFrame进行多种操作。
- （1）单列切片，具体代码如前述所示。
- （2）多列切片，其原理是将多列的列名或位置作为一个列表或数据传入，代码如下所示：

```
In[15]:order_id_detail_id1 = order.loc[:, ['order_id', 'detail_id']]
print('使用loc()方法提取order_id和detail_id列的size为：',
 order_id_detail_id1.size)
order_id_detail_id2 = order.iloc[:, [1, 3]]
print('使用iloc()方法提取第1和第3列的size为：', order_id_detail_id2.size)
Out[15]: 使用loc()方法提取order_id和detail_id列的size为： 5558
使用iloc()方法提取第1和第3列的size为： 5558
```

【任务实施】

- （3）取出DataFrame中的任意数据，代码如下所示：

```
In[16]:print('列名为order_id和detail_id的行名为3的数据为: \n',
      order.loc[3, ['order_id', 'detail_id']])
print('列名为order_id和detail_id行名为2,3,4,5,6的数据为: \n',
      order.loc[2: 6, ['order_id', 'detail_id']])
print('列位置为1和3，行位置为3的数据为: \n', order.iloc[3, [1,
 3]])
print('列位置为1和3，行位置为2,3,4,5,6的数据为: \n',
      order.iloc[2: 7, [1, 3]])
Out[16]: 列名为order_id和detail_id的行名为3的数据为:
order_id      417
detail_id    2966
Name: 3, dtype: object
```

【任务实施】

- （3）取出DataFrame中的任意数据，代码如下所示：

```
列名为order_id和detail_id行名为2,3,4,5,6的数据为:
    order_id  detail_id
2        417        2961
3        417        2966
4        417        2968
5        301        1899
6        301        1902
列位置为1和3，行位置为3的数据为:
 order_id          417
logicprn_name     NaN
```

```
logicprn_name     NaN
Name: 3, dtype: object
列位置为1和3，行位置为2,3,4,5,6的
 数据为:
    order_id  logicprn_name
2        417            NaN
3        417            NaN
4        417            NaN
5        301            NaN
6        301            NaN
```

【任务实施】

- 从此段代码中看出在使用loc()方法的时候，如果内部传入的行索引名称为一个区间，那么前后均为闭区间。而使用iloc()方法时，如果内部传入的行索引位置或列索引位置为区间，那么为前闭后开区间。

- loc()方法的内部还可以传入表达式，结果会返回满足表达式的所有值，代码如下所示：

```
# 传入表达式
In[17]print('order中detail_id为"2956"的order_id为：\n',
      order.loc[order['detail_id']=='2956',['order_id','detail_id' ]])
Out[17]：order中detail_id为"2956"的order_id为：
 Empty DataFrame
Columns: [order_id, detail_id]
Index: []
In[18]:print('order中detail_id为"2956"的第1、4列数据为：\n',
     order.iloc[order['detail_id'] == '2956', [1, 4]])
Out[18]: NotImplementedError : iLocation based boolean indexing on an integer type is
not available
```

【任务实施】

- iloc()方法不能接收表达式，原因在于，iloc()方法可以接收的数据类型并不包括Series。根据Series的构成，应取出该Series的values，代码如下所示：

```
In[19]:print('order中detail_id为"2956"的第1、4列数据为: \n',
      order.iloc[(order['detail_id'] == '2956').values, [1, 4]])
Out[19]: order中detail_id为"2956"的第1、4列数据为:
 Empty DataFrame
Columns: [order_id, parent_class_name]
Index: []
```

- 总体来说，loc()方法更加灵活多变，代码的可读性更高；iloc()方法的代码简洁，但可读性不高。在数据分析工作中具体使用哪一种方法，应根据情况而定，大多数时候建议使用loc()方法。

任务3.4 生成餐饮订单数据的销售额

目录

3.4 生成餐饮订单数据的销售额

【任务描述】

描述性统计是用于概括、表述事物整体状况，以及事物间关联、类属关系的统计方法。通过几个统计值可简捷地表示一组数据的集中趋势和离散程度等。Pandas对两种不同的数据特征使用了不同的方法。

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/426044033151010135