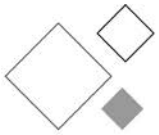

第5章

数据存取

数据存储是数据分析中非常重要的一环。在第 4 章爬取的数据仅仅是显示在程序的输出结果中，这样很不利于后期的数据分析和可视化。为此在爬取数据后，还需要对数据进行存储。常用的数据存储有文件和数据库两种方式，文件存储的格式有 TXT、CSV、XLSX 等，数据库存储有关系型数据库和非关系型数据库两类。

5.1 JSON

在当今 Web 3.0 时代，以 JavaScript 与 XML 为代表的结合 JavaScript、CSS、HTML 等网页开发技术，仍然占据主流地位。

5.1.1 JSON 概述

1. JSON 的概念

JSON 的全称为 JavaScript Object Notation，即 JavaScript 对象标记。JSON 通过对象和数组的组合表示数据，是一种轻量级的数据交换格式。

JSON 示例代码如下。

```
{
    "sites": [
        { "name":"重庆美食" , "url":"www.link1.com" },
        { "name":"重庆美景" , "url":"www. link2.com" },
        { "name":"重庆好玩" , "url":"www. link3.com" }
    ]
}
```

XML 示例代码如下。

```
<sites>
  <site>
    <name>重庆美食</name><url> www.link1.com </url>
  </site>
  <site>
    <name>重庆美景</name><url> www.link2.com </url>
  </site>
```

```
  <site>
    <name>重庆好玩</name><url>www.link3.com</url>
  </site>
</sites>
```

通过以上两个示例的比较可以发现，JSON 是比 XML 更简单的一种数据交换格式，它采用独立于编程语言的文本格式来存储和表示数据。JSON 的语法规则如下。

(1) 使用键值对(key:value)表示对象属性和值。JSON 的值有多种类型，如数字(整数或浮点数)、字符串(在双引号中)、逻辑值(true 或 false)、数组(在方括号中)、对象(在花括号中)和空(null)。

(2) 使用逗号“(,)”分隔多条数据。

(3) 使用花括号“{}”包含对象。

(4) 使用方括号“[]”表示数组。

2. JSON 与 XML 比较

JSON 和 XML 都是文本格式语言，均普遍用于数据交换和网络传输，它们的区别有以下几个方面。

(1) 可扩展性。

二者都有很好的扩展性，但 JSON 与 JavaScript 语言的结合更紧密，在 JavaScript 语言编写的网页中使用 JSON 更为合适。

(2) 可读性。

二者的可读性都很好，JSON 的特点是简洁的语法，XML 则是规范的标签形式。

(3) 编码难度。

XML 诞生时间相对 JSON 而言要早一些，因此处理 XML 语言的编码工具更丰富。但是在相同结果下，XML 文档需要的字符量更多。

(4) 解码难度。

JSON 和 XML 都是可扩展性的结构，如果不知道文档结构，则解析文档会非常不方便。因此，两者都需要知道文档结构后才能进行解析。对于一般网页结构，可以通过 F12 键在开发者模式中进行直接观察，并由此作出判断。

(5) 有效数据率。

由于省去了大量的标签，因此 JSON 的有效数据率比 XML 高很多。

5.1.2 用 JSON 库存取 JSON 文件

JSON 库是 Python 中用于编码、解码 JSON 格式的标准库模块，主要用于将 Python 对象编码为 JSON 格式输出或存储，以及将 JSON 格式对象解码为 Python 对象。

JSON 库提供了 4 种函数：json.dump()、json.load()、json.dumps()、json.loads()。json.loads()可以将字符串转换为 Python 的数据结构，变成列表或字典。json.dumps()则相反，将 Python 类型的列表或字典转换为 JSON 字符串。json.load()可以传入一个 JSON 格式的文件流，并将其解码为 Python 对象。json.dump()函数用于将 Python 类型的数据以 JSON 格式存储到文件中。

1. 存取 JSON 数据

在存取 JSON 数据时，用 json.dumps()和 json.loads()来实现 JSON 字符串和 Python 数据类型的互转。例如：

```
import json
data = '''
[{ "美食": "火锅",
    "价格": "100",
    "城市": "重庆"
},
{     "美食": "小面",
    "价格": "10",
    "城市": "重庆"
}]
'''
print(data)                                          #输出 data
print(type(data))                                    #输出 data 的类型
py_data = json.loads(data)                           #将 JSON 类型的数据转换为 Python 类型的数据
print(py_data)                                       #输出 py_data 数据
print(type(py_data))                                 #输出 py_data 的类型
js_str = json.dumps(py_data,ensure_ascii=False)      #将 Python 类型的数据转换为 JSON 字符串
print(js_str)                                        #输出转换后的 JSON 数据
print(type(js_str))                                  #输出转换后的 JSON 类型
```

输出结果如下。

```
[{ "美食": "火锅",
    "价格": "100",
    "城市": "重庆"
},
{     "美食": "小面",
    "价格": "10",
    "城市": "重庆"
}]
<class 'str'>
[{'美食': '火锅', '价格': '100', '城市': '重庆'}, {'美食': '小面', '价格': '10', '城市': '重庆'}]
<class 'list'>
[{"美食": "火锅", "价格": "100", "城市": "重庆"}, {"美食": "小面", "价格": "10", "城市":
"重庆"}]
<class 'str'>
```

在以上代码中，使用 json.dumps 将 Python 类型的数据转换为 JSON 字符串时，添加了参数 ensure_ascii＝False。如果不添加 ensure_ascii 参数，默认输出为 ASCII 编码，而本例中的中文字符是无法正常输出的，因此将参数设置为 False 才能正常输出中文字符。

2. 存取 JSON 文件

在存取 JSON 文件时，用 json.dump()和 json.load()来实现 JSON 文件和 Python 数据类型的互转。例如：

```
import json
data = '''
```

```
[{ "美食": "火锅",
    "价格": "100",
    "城市": "重庆"
},
{    "美食": "小面",
    "价格": "10",
    "城市": "重庆"
}]
'''
with open("jd_json.json",'w',encoding = 'utf - 8') as f:
    json.dump(data,f,ensure_ascii = False)
with open("jd_json.json",'r',encoding = 'utf - 8') as f:
    data_p = json.load(f)
    print(data_p)
```

程序执行结束后，会在当前目录生成一个 jd_json.json 文件并输出文件内容，输出结果如下。

```
[{ "美食": "火锅",
    "价格": "100",
    "城市": "重庆"
},
{    "美食": "小面",
    "价格": "10",
    "城市": "重庆"
}]
```

注意：文件操作也存在编码问题，encoding＝'utf-8'可以避免中文不被正常显示。

5.1.3 用 Pandas 库存取 JSON 文件

Pandas 是 Python 中常用的第三方库，它在数据分析领域中占据重要的地位。除此之外，Pandas 也可以很方便地处理 JSON 文件。

安装 Pandas 库的方法有命令行方法和菜单方法两种，在 Windows 命令行中输入 pip install pandas 即可成功安装 Pandas 库。要使用 Pandas 库，需要在程序前输入 import pandas as pd，这里的 pd 是对 Pandas 库的简写，方便编写时调用。

接下来介绍 Pandas 库对 JSON 文件的常用存取方法。

1. 写入文件

写入文件使用 to_json()语句，在转换格式时，如果出现中文乱码，则添加参数 force_ascii＝False 即可解决编码问题。例如：

```
#导入 Pandas 库
import pandas as pd
#定义数据
data =[{"美食": "火锅","价格": "100","城市": "重庆"}, {"美食": "小面","价格": "10","城
市": "重庆"}]
#转为 Pandas 数据框
```

```
import pandas as pd
frame = pd.DataFrame(data)
print(frame)
frame.to_json('frame.json',force_ascii = False)
```

程序执行后，会在当前目录生成一个 frame.json 文件，用记事本打开，内容如下。

```
{"美食":{"0":"火锅","1":"小面"},"价格":{"0":"100","1":"10"},"城市":{"0":"重庆","1":"重庆"}}
```

2. 读取文件

读取文件使用 to_json()函数，将待读取文件名传入参数即可。例如：

```
import pandas as pd
frame = pd.read_json('frame.json')
print(frame)
```

输出结果如下。

```
   美食  价格  城市
0  火锅   100  重庆
1  小面    10  重庆
```

5.2 CSV 存取

CSV(Comma-Separated Values，逗号分隔值)，即字符分隔值。CSV 文件以纯文本(字符序列)的形式存储数据。CSV 文件由多条记录组成，记录之间以换行符分隔，每条记录由字段组成，字段之间的分隔符可以是逗号或制表符等其他字符。CSV 文件的扩展名是.csv，可以使用记事本或 Excel 打开。

5.2.1 用 CSV 库存取 CSV 文件

Python 内置了 CSV 库，用于对 CSV 文件进行读/写，因此不需要另外安装便可使用。要使用 CSV 库，需要在程序前输入 import csv。

CSV 库包含了处理 CSV 文件的多种方法，这里主要介绍读文件和写文件的常用对象和方法，如表 5-1 所示。

表 5-1 CSV 库的常用对象和方法

对象/方法名	描 述	类 型
csv.writer(csvfile, dialect = 'excel', **fmtparams)	返回将数据写入 CSV 文件的写入器对象	对象
csv.reader(csvfile, dialect = 'excel', **fmtparams)	返回一个遍历 CSV 文件各行的读取器对象	

续表

对象/方法名	描　　述	类　　型
writerows(Iterable)	一次性写入多行数据	方法
writerow(Iterable)	写入单行数据	
csv. DictWriter (f, fieldnames, restval = "", extrasaction="raise", dialect="excel", * args, ** kwds)	以字典的形式写入数据	
csv. DictReader(f, fieldnames = None, restkey = None, restval=None, dialect="excel", * args, ** kwds)	以字典的形式返回读取的数据	
writeheader()	写入标题行	

1. 写入数据

下面通过代码介绍用 CSV 库写入数据文件的基本操作。

(1) 列表写入。

```
#导入 CSV 库
import csv
#定义写文件函数
def writer():
    #创建列表,保存标题内容
    biaoti = ["景点编号", "景点名称", "门票"]
    #创建列表,保存数据
    data = [
        ["001", "动物园", 25],
        ["002", "濯水古镇", 50],
        ["003", "科技馆", 0]
    ]
    #以写方式打开文件.注意添加 newline = "",否则会在两行数据之间都插入一行空白
    with open("jingdian.csv", mode = "w", encoding = "utf - 8 - sig", newline = "") as f:
        #基于打开的文件,创建 csv.writer 实例
        w = csv.writer(f)
        #写入标题,writerow() 一次只能写入一行
        w.writerow(biaoti)
        #写入数据,writerows() 一次写入多行
        w.writerows(data)
writer()
```

以上代码首先定义了 biaoti 的列表,描述每列的标题;同时创建了 data 列表,定义了 3 行 3 列的数据,并使用“with…open…”语句创建了 jingdian. csv 文件。需要注意的是,f=open('names. csv', 'w')也可以创建一个 names. csv 文件,但是它必须要结合 f. close()一起使用,否则后面对文件的操作会出现问题。“with…open…”语句通常不会出错,因此选用这种方式创建文件比较可靠。参数 mode="w"表示写入文件;encoding="utf-8-sig"表示一种编码方式,这种编码方式可以解读中文字符,未指定编码方式将会导致中文字符不显示;newline=""可以避免在两行数据之间插入一行空白。然后基于打开的文件,用 csv. writer(f)将 f 文件定义为一个写文件实例。接着用 writerows()写入一行的标题内容,用 writerows 写入了数据。最后调用 writer()函数,执行函数的所有内容,在当前 Python 文件的所在目

录生成了一个 jingdian.csv 文件。

通过 CSV 库写入 CSV 文件，用 Excel 打开该文件，如图 5-1 所示。

	A	B	C
1	景点编号	景点名称	门票
2	1	动物园	25
3	2	濯水古镇	50
4	3	科技馆	0
5			

图 5-1　通过 CSV 库写入 CSV 文件

（2）字典写入。

```
#导入 CSV 库
import csv
def writer():
    #创建列表,保存标题内容
    biaoti = ["景点编号", "景点名称", "门票"]
    data = [
        {"景点编号":"001", "景点名称":"动物园","门票":25},
        {"景点编号":"002", "景点名称":"濯水古镇", "门票":50},
        {"景点编号":"003", "景点名称":"科技馆", "门票":0}
    ]
    #以写方式打开文件.注意添加 newline = "",否则会在两行数据之间都插入一行空白
    with open("jingdian.csv", mode = "w", encoding = "utf - 8 - sig", newline = "") as f:
        #基于打开的文件,创建 csv.DictWriter 实例
        w = csv.DictWriter(f,biaoti)
        #写入列标题
        w.writeheader()
        #写入数据,writerows() 一次写入多行
        w.writerows(data)
writer()
```

以上代码定义了 biaoti 作为列名称列表，将 data 定义为数据的字典列表，再用 DictWriter()方法将 biaoti 指定为 data 中列标题，将二者进行关联后用 writeheader()写入标题。此种编写方法相对列表而言稍显烦琐，但优点在于便于初学者理解。

2. 读取数据

```
#导入 CSV
import csv
#以指定编码的只读方式打开 jingdian.csv
with open("jingdian.csv", "r", encoding = "utf - 8 - sig") as f:
    jingdian = csv.reader(f)
    #逐行输出 CSV 文件内容
    for line in jingdian:
        print(line)
```

读取 CSV 文件的方法比较简单，用 with open 语句以只读的方式打开 jingdian.csv 文件，再用 csv.reader(f)将 CSV 文件转为读取器对象，通过 for 循环逐行输出 CSV 文件的所有内容。

5.2.2 用 Pandas 库存取 CSV 文件

Pandas 库也可以很方便地处理 CSV 文件，它的实现代码要比 CSV 库少很多。接下来介绍 Pandas 库对 CSV 文件的存取功能。

1. 写入数据

```
#Pandas 写入 CSV 文件
#导入 Pandas 库并简写为 pd
import pandas as pd
#定义 3 个列表 jdbh、jdmc、jdmp,用于存放数据
jdbh = ["001", "002", "003"]
jdmc = ["动物园", "濯水古镇", "科技馆"]
jdmp = [25, 50, 0]
#定义字典,为每一列数据加上字段名
jingdian = {"景点编号": jdbh, "景点名称": jdmc, "门票": jdmp}
#将 jingdian 这个字典定义为一个数据框
jd = pd.DataFrame(jingdian)
#保存数据到 jingdian_pandas.csv 文件中
jd.to_csv('jingdian_pandas.csv',encoding = "utf - 8 - sig", index = False)
```

在以上代码中，特别要注意保存文件时需要指定编码为 utf-8-sig，否则无法显示中文内容。最终生成一个 jingdian_pandas.csv 文件，打开该文件，如图 5-2 所示。

	A	B	C	D
1		景点编号	景点名称	门票
2	0	1	动物园	25
3	1	2	濯水古镇	50
4	2	3	科技馆	0
5				

图 5-2 通过 Pandas 库写入 CSV 文件

在使用.to_csv()写入数据时，会自动生成索引和列名称，因此需要将参数设置为 index＝False 或 headers＝False，以避免写入索引和列名称。

2. 读取数据

```
import pandas as pd
#打开文件
with open('jingdian_pandas.csv', 'r',encoding = "utf - 8 - sig") as f:
    #读取文件
    data = pd.read_csv(f, encoding = "utf - 8 - sig")
print(data)
```

输出结果如下。

```
   Unnamed: 0    景点编号    景点名称    门票
0           0       1       动物园       25
1           1       2       濯水古镇     50
2           2       3       科技馆        0
```

用 Pandas 库写入文件，其特点在于增加了索引和列名称。通常可以使用 names 选项指定表头，并将存有列名的数组赋给它。以下代码为读取结果添加了列名称。

```
data_tou = pd.read_csv("jingdian_pandas.csv",names=['1列','2列','3列'])
print(data_tou)
```

输出结果如下。

```
        1列       2列     3列
NaN  景点编号  景点名称  门票
0.0     001      动物园     25
1.0     002     濯水古镇    50
2.0     003      科技馆      0
```

若要将上述输出结果的索引“0.0,1.0,2.0”修改为所指定的索引,则需要通过参数index_col实现。

3. 读取部分数据

读取部分数据的方法如下。

(1) head(*n*)方法：用于读取前 *n* 行数据。如果未填写参数 *n*,则默认为前5行。此外,.read_csv()中添加参数 nrows=n 也可以读取前 *n* 行数据。

(2) tail(*n*)方法：用于读取尾部的 *n* 行。如果未填写参数 *n*,则默认返回5行,空行各字段的值返回 NaN。

(3) info()方法：返回表格的一些基本信息。

5.2.3　应用案例

实现效果：爬取“重庆—北京”的12306旅行车次信息,将其存储为CSV格式。

1. 直接存储文件

直接存储文件的实现代码如下。

```
# ============== 爬取部分 ==============================
import time
from selenium import webdriver
from lxml import etree
base_url = r'https://kyfw.12306.cn/otn/leftTicket/init?linktypeid=dc&fs=%E9%87%8D%
E5%BA%86,CQW&ts=%E5%8C%97%E4%BA%AC,BJP&date=2023-03-24&flag=N,N,Y'
print("正在爬取数据……请等待……")
options = webdriver.FirefoxOptions()
#设置浏览器为 headless 无界面模式
options.add_argument("--headless")
options.add_argument("--disable-gpu")
#打开浏览器处理,注意浏览器无显示
browser = webdriver.Firefox(options=options)
browser.get(base_url)
time.sleep(4)
res = browser.page_source
html = etree.HTML(res)
#车次
result1 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[1]/
div/a/text()')
#始发站
```

```
result2 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[2]/
strong[1]/text()')
#到达站
result3 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[2]/
strong[2]/text()')
#出发时间
result4 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[3]/
strong[1]/text()')
#到达时间
result5 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[3]/
strong[2]/text()')
#历时
result6 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[4]/
strong/text()')
#输出车次信息
print('----共计{0}个车次信息,分别是:----'.format(len(result1)))
for x in range(0,len(result1)):
    print('车次:',result1[x],'始发站:',result2[x],'到达站:',result3[x],'出发时间:',result4
[x],'到达时间:',result5[x],'历时:',result6[x])
print('----爬取的车次信息,显示完成----')
#等待3s,关闭浏览器
time.sleep(3)
browser.close()
#  ========== CSV存储部分 ===============
for x in range(len(result1)):
    with open('火车信息.csv','a') as f:
        f.write(result1[x] + ',')
        f.write(result2[x] + ',')
        f.write(result3[x] + ',')
        f.write(result4[x] + ',')
        f.write(result5[x] + ',')
        f.write(result6[x] + '\n')
```

运行代码,生成一个“火车信息.csv”文件,用记事本打开文件,如图5-3所示。

图5-3 直接存储文件

2. 用CSV库存储文件

用CSV库存储文件的实现代码如下。

```
l = [result1,result2,result3,result4,result5,result6]
l = list(map(list, zip( * l)))
print(l)
biaoti = ['车次','始发站','终到站','始发时间','终到时间','用时']
with open("火车信息_csv.csv", mode = "w", encoding = "utf - 8 - sig", newline = "") as f:
    w = csv.writer(f)
    # 写入标题,writerow() 一次只能写入一行
    w.writerow(biaoti)
    # 写入数据,writerows() 一次写入多行
    w.writerows(l)
```

3. 用 Pandas 库存储文件

用 Pandas 库存储文件的实现代码如下。

```
# ============== 爬取部分 ============================
import time
from selenium import webdriver
from lxml import etree
import pandas as pd
base_url = r'https://kyfw.12306.cn/otn/leftTicket/init?linktypeid = dc&fs = %E9%87%8D%
E5%BA%86,CQW&ts = %E5%8C%97%E4%BA%AC,BJP&date = 2023 - 03 - 24&flag = N,N,Y'
print("正在爬取数据 …… 请等待 …… ")
options = webdriver.FirefoxOptions()
#设置浏览器为 headless 无界面模式
options.add_argument(" -- headless")
options.add_argument(" -- disable - gpu")
#打开浏览器处理,注意浏览器无显示
browser = webdriver.Firefox(options = options)
browser.get(base_url)
time.sleep(4)
res = browser.page_source
html = etree.HTML(res)
#车次
result1 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[1]/
div/a/text()')
#始发站
result2 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[2]/
strong[1]/text()')
#到达站
result3 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[2]/
strong[2]/text()')
#出发时间
result4 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[3]/
strong[1]/text()')
#到达时间
result5 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[3]/
strong[2]/text()')
#历时
result6 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[4]/
strong/text()')
#输出车次信息
print('---- 共计{0}个车次信息,分别是: ---- '.format(len(result1)))
```

```
for x in range(0,len(result1)):
    print('车次:',result1[x],'始发站:',result2[x],'到达站:',result3[x],'出发时间:',result4
[x],'到达时间:',result5[x],'历时:',result6[x])
print('----爬取的车次信息,显示完成----')
#等待 3s,关闭浏览器
time.sleep(3)
browser.close()
#  ========= CSV 存储部分 ================
df = pd.DataFrame({'车次':result1,'始发站':result2,'到达站':result3,'出发时间':result4,'到达
时间':result5,'历时':result6})
df.to_csv('火车查询信息'+time.strftime("-%m%d-%H%M%S", time.localtime())+'.csv',
index=False,encoding='utf-8-sig')
print('----爬取的车次信息,存储到 CSV 中----')
```

运行代码,自动生成一个“火车查询信息-0316-231407.csv”文件,用 Excel 打开文件,如图 5-4 所示。

	A	B	C	D	E	F
1	车次	始发站	到达站	出发时间	到达时间	历时
2	G52	重庆北	北京西	7:32	14:26	6:54
3	G352	重庆北	北京西	8:15	17:03	8:48
4	G388	重庆西	北京西	8:38	20:03	11:25
5	T10	重庆西	北京西	9:56	11:12	25:16:00
6	G332	重庆北	北京西	10:48	19:06	8:18
7	Z96	重庆西	北京西	11:25	10:51	23:26
8	G372	重庆西	北京西	11:57	21:34	9:37
9	G372	重庆北	北京西	12:24	21:34	9:10
10	G54	重庆北	北京西	14:33	21:44	7:11
11	Z50	重庆北	北京西	14:37	10:05	19:28
12	Z4	重庆北	北京西	15:24	10:11	18:47
13	K508	重庆西	北京西	21:00	21:34	24:34:00
14						

图 5-4　用 Pandas 库存储文件

5.3　XLSX 存取

目前,Python 中的读/写 XLSX 文件的第三方库有很多,如 xlrd、xlwt、xlutils、xlwings、Openpyxl、xlsxwriter、Pandas 等。每个库都各有特点,本节选取常用的 xlrd、xlsxwriter、Openpyxl 和 Pandas 库对 XLSX 文件进行存取。

5.3.1　用 xlrd 库存取 XLSX 文件

1. 安装与导入

安装 xlrd 库时,用命令行方式输入 pip install xlrd==1.2.0,导入模块使用 import xlrd。需要注意的是,只有 1.2.0 及以上版本才支持 XLSX 文件格式,否则文件读取失败。

2. 使用方法

(1) 定义一个 XLSX 文件对象。

```
data = xlrd.open_workbook("E:\火车信息.xlsx")
```

注意：如果路径或文件名有中文，则需要在前面加一个"r"。

(2) 获取工作表。

- 按索引顺序获取：table = data.sheets()[0]。
- 按工作表名称获取：table = data.sheet_by_name(sheet_name)。
- 查询工作表名称：names = data.sheet_names()。

(3) 行的操作。

- 获取有效行数：nrows = table.nrows。
- 返回由该行中所有的单元格对象组成的列表：table.row(rowx)。
- 返回由该列中所有的单元格对象组成的列表：table.row_slice(rowx)。
- 返回由该行中所有单元格的数据类型组成的列表：table.row_types(rowx，start_colx=0，end_colx=None)。
- 返回由该行中所有单元格的数据组成的列表：table.row_values(rowx，start_colx=0，end_colx=None)。
- 返回该列的有效单元格长度：table.row_len(rowx)。

(4) 列的操作。

- 获取列表的有效列数：ncols = table.ncols。
- 返回由该列中所有的单元格对象组成的列表：table.col(colx，start_rowx=0，end_rowx=None)。
- 返回由该列中所有的单元格对象组成的列表：table.col_slice(colx，start_rowx=0，end_rowx=None)。
- 返回由该列中所有单元格的数据类型组成的列表：table.col_types(colx，start_rowx=0，end_rowx=None)。
- 返回由该列中所有单元格的数据组成的列表：table.col_values(colx，start_rowx=0，end_rowx=None)。

(5) 单元格的操作。

- 返回单元格对象：table.cell(rowx,colx)。
- 返回单元格中的数据类型：table.cell_type(rowx,colx)。
- 返回单元格中的数据：table.cell_value(rowx,colx)。

3. 基础操作案例

对“E:\火车信息.xlsx”文件进行操作如下。

```
#导入 xlrd 模块
import xlrd
#定义一个 XLSX 文件对象
data = xlrd.open_workbook(r"E:\火车信息.xlsx")#文件名和路径,如果路径或文件名有中
#文,则需要在前面加一个"r"
#查询工作表名称
names = data.sheet_names()
#获取第 1 个工作表
table = data.sheet_by_index(0)
#获取表格行数
nrows = table.nrows
```

```
print("表格一共有", nrows, "行")
#获取表格第 2 行内容
row = table.row(1)
print("表格第 2 行内容有:", row)
#获取表格列数
nclos = table.ncols
print("表格一共有", nclos, "列")
#获取表格第一列内容
col = table.col(0)
print("表格第 1 列内容有:", col)
#获取单个表格值
value = table.cell_value(1, 2)
print("第 2 行第 3 列值为", value)
```

输出结果如下。

```
表格一共有 13 行
表格第 2 行内容有: [text:'G52', text:'重庆北', text:'北京西', xldate:0.3138888888888889,
xldate:0.6013888888888889, xldate:0.28750000000000003]
表格一共有 6 列
表格第 1 列内容有: [text:'车次', text:'G52', text:'G352', text:'G388', text:'T10', text:'G332',
text:'Z96', text:'G372', text:'G372', text:'G54', text:'Z50', text:'Z4', text:'K508']
第 2 行第 3 列值为 北京西
```

5.3.2 用 xlsxwriter 库写入 XLSX 文件

xlsxwriter 库支持 XLSX 文件的写入，支持 VBA，可以实现写入图表、设置数据验证、下拉列表、条件格式等操作。在写入 XLSX 文件较大时使用内存优化模式，是常用的一种写入 Excel 文件方式。

1. 安装与导入

安装 xlsxwriter 库时，用命令行方式输入 pip install xlsxwriter，导入模块需要输入 import xlsxwriter。

2. 使用方法

(1) 创建工作簿文件。

```
workbook = xlsxwriter.Workbook('旅游数据.xlsx')
```

可以创建一个名为“旅游数据.xlsx”的文件，定义工作簿变量为 workbook。

(2) 创建工作表。

```
worksheet1 = workbook.add_worksheet()
```

使用 workbook.add_worksheet()可以为 workbook 工作簿对象创建工作表，定义工作表变量为 worksheet1。

(3) 写入数据。

① 第 1 种方法：按单元格添加。

```
worksheet1.write(A1, '景点编号')
```

用.write()为 A1 单元格录入数据“景点编号”。

② 第 2 种方法：按行号、列表添加。

```
worksheet.write(0,0,'景点编号')
```

用 write()为第 1 行第 1 列录入数据“景点编号”。

在 Excel 中，录入的数据类型有文本、数字、日期等，因此，在 xlsxwriter 库中定义了不同类型数据的写入方法。例如，write_string()写入字符串，write_number()写入数字，write_blank()写入空值，write_formula()写入公式与函数，write_datetime()写入日期，write_boolean()写入逻辑值，write_url()写入链接。

(4) 调整格式。

① 调整行高：set_row(row，num)。

例如，worksheet1.set_row(1，20)可以将第 2 行的行高设置为 20。

② 调整列宽：set_column(col1，col2，num)，其中 col1 和 col2 分别指起始列和结束列。

例如，worksheet.set_column('B:C'，20) 可以将 B 列和 C 列列宽都设定为 20。

③ 合并单元格。

例如，worksheet1.merge_range(0，0，0，4，'景点信息表')表示从第 0 行第 0 列开始，合并 1 行 5 列单元格，并在合并后的单元格中输入“景点信息表”的文字。

④ 设置单元格格式。

先在 workbook 中定义样式，然后在写入数据中加上样式即可。例如：

```
import xlsxwriter
workbook = xlsxwriter.Workbook('景点信息.xlsx')
worksheet1 = workbook.add_worksheet()
format = workbook.add_format({
        'bold': True,                  #字体加粗
        'align': 'center',             #水平居中
        'valign': 'vcenter',           #垂直居中
        'border': 1,                   #单元格边框宽度
        'bg_color': '#ffFF00',         #单元格背景颜色
        'num_format': '0.00',          #格式化数据格式为小数点后两位
        'font_size': 18,               #字体大小
        'font_color': 'red'            #字体颜色
})
worksheet1.merge_range(0, 0, 0, 4, '景点信息表', format)
workbook.close()
```

运行代码，生成一个“景点信息.xlsx”文件，打开文件内容，如图 5-5 所示。

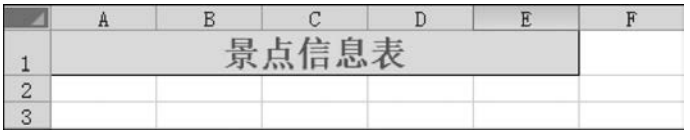

图 5-5　用 xlsxwriter 库生成 XLSX 文件并设置格式效果图

3. 应用案例

用 xlsxwriter 库写入文件的实现代码如下。

```
#导入模块
import xlsxwriter
#创建工作簿
workbook = xlsxwriter.Workbook('景点信息.xlsx')
#创建工作表
worksheet1 = workbook.add_worksheet()
#定义标题格式
format = workbook.add_format({
        'bold': True,                  #字体加粗
        'align': 'center',             #水平居中
        'valign': 'vcenter',           #垂直居中
        'border': 1,                   #单元格边框宽度
        'bg_color': '#ffFF00',         #单元格背景颜色
        'num_format': '0.00',          #格式化数据格式为小数点后两位
        'font_size': 18,               #字体大小
        'font_color': 'red'            #字体颜色
})
#合并1行5列,录入标题内容
worksheet1.merge_range(0, 0, 0, 4, '景点信息表', format)
#录入标题
worksheet1.write(1, 0, '景点编号')
worksheet1.write(1, 1, '景点名称')
worksheet1.write(1, 2, '项目')
worksheet1.write(1, 3, '票价')
worksheet1.write(1, 4, '地址')
#定义数据
data = (
    ['1',"动物园",'门票','25','重庆'],
    ['2', "科技馆", '门票', '25', '重庆'],
    ['3',"金佛山",'套票','100','重庆'],
)
#写入数据
for i in range(2,len(data) + 2):
    for j in range(5):
        worksheet1.write_string(i,j,data[i - 2][j])
#关闭保存
workbook.close()
```

运行代码,在当前目录生成一个“景点信息.xlsx”文件,打开文件,效果如图5-6所示。

	A	B	C	D	E
1	景点信息表				
2	景点编号	景点名称	项目	票价	地址
3	1	动物园	门票	25	重庆
4	2	科技馆	门票	25	重庆
5	3	金佛山	套票	100	重庆
6					
7					

图5-6 用xlsxwriter库写入文件效果图

5.3.3 用Openpyxl库读/写、修改XLSX文件

Openpyxl是一个用于读取和编写XLSX、XLSM、XLTX和XLTM文件的库。它支持

XLSX 文件的读/写和修改，几乎可以实现 Excel 的所有功能，而且接口清晰，文档丰富。

1. 安装和导入

安装 Openpyxl 库时，用命令行方式输入 pip install openpyxl，导入模块需要输入 import openpyxl。

2. 使用方法

(1) 操作工作簿。

- 打开并读取工作簿：wb＝openpyxl. load_workbook(filepath)，新建文件时默认有一个名为 Sheet 的工作表。
- 创建工作簿：wb＝openpyxl. Workbook()。
- 保存工作簿：wb. save(filename)。

(2) 操作工作表。

- 获取第一个工作表对象：ws＝wb. active。
- 获取指定名称的工作表对象：wb[sheet_name]。
- 获取所有工作表名称：wb. sheetnames。
- 获取所有工作表对象：wb. worksheets，wb. worksheets[0]表示第一个工作表。
- 创建工作表并返回工作表对象：wb. create_sheet(sheet_name，index＝"end")，默认添加到末尾。
- 复制工作表并返回工作表对象：wb. copy_worksheet(sheet)。
- 删除工作表：wb. remove(sheet)。

(3) 操作单元格。

- 设置工作表名称：ws. title。
- 设置工作表最大行号：ws. max_row。
- 设置工作表最大列表数字：ws. max_column。
- 表格末尾追加数据：ws. append(list)。
- 合并单元格：ws. merge_cells('A2：F2')。
- 取消合并单元格：ws. unmerge_cells('A2：F2')。

(4) 单元格读取。

- 根据坐标读取单元格：ws['A1']。
- 根据行列读取单元格：ws. cell(row，column，value＝None)。
- 获取第一行的所有单元格对象：ws[1]。
- 获取第 B 列的所有单元格 ws["B"]。
- 获取第 A～B 列的所有单元格：ws["A：B"]。
- 获取第 1～2 行的所有单元格对象：ws[1：2]。
- 获取单元格区域：ws["A1：D4"]。
- 以列表形式返回所有单元格数据：ws. values。
- 设置单元格的值：cell. value。
- 设置数字列标：cell. column。
- 设置字母列标：cell. column_letter。
- 设置行号：cell. row。

- 设置单元格的坐标：cell. coordinate。
- 设置单元格的数据类型：cell. data_type。
- 设置单元格格式：cell. number_format，默认为常规型。
- 设置单元格 Font 对象：cell. font。
- 设置单元格边框：cell. border。
- 设置单元格水平/垂直对齐方式：cell. alignment。
- 设置单元格填充颜色：cell. fill。

(5) 行/列操作。

- 获取行对象：ws. row_dimensions[行号]。
- 获取列对象：ws. column_dimensions[字母列表]。
- 返回列表：get_column_letter(index)。
- 返回数字列表：column_index_from_string(string)。
- 设置行高：row. height。
- 设置列宽：column. width。

3. 基础操作案例

输入以下代码，实现写入数据到 XLSX 文件，并读取文件信息，输出文件内容。

```
import openpyxl
# ============ 写入部分 ==============
#创建工作簿文件
jd = openpyxl.Workbook()
#激活当前工作表
js = jd.active
#创建一个 sheet1
s1 = jd.create_sheet("景点表",0)
#创建一个 sheet2
s2 = jd.create_sheet("游客表",1)
#设置 sheet1 标签背景色
s1.sheet_properties.tabColor = "00ff00"
#修改工作表名称
s1.title = "景点信息表"
#合并 1 行 5 列,录入标题内容
s1.merge_cells('A1:E1')
s1['A1'] = "景点信息表"
#写入小标题
title = ['景点编号','景点名称','项目','票价','地址']
for col in range(len(title)):
    c = col + 1
    s1.cell(row = 2,column = c).value = title[col]
jd.save("景点信息_openpy.xlsx")
#输入数据
data = [
    ('1',"动物园",'门票','25','重庆'),
    ('2', "科技馆", '门票', '25', '重庆'),
    ('3',"金佛山",'套票','100','重庆'),
]
rows = len(data)
```

```
l = len(data[0])
for i in range(rows):
    for j in range(l):
        s1.cell(row = i + 2, column = j + 1).value = data[i][j]
jd.save("景点信息_openpy.xlsx")
# ============ 读取部分 ==============
import openpyxl
#打开工作簿文件
jd = openpyxl.load_workbook("景点信息_openpy.xlsx")
js = jd.active
#获取所有表格(worksheet)的名字
sheets = jd.sheetnames
print(sheets)
#获取工作表
s1 = jd["景点信息表"]
#获取表格所有行和列,两者都是可迭代的
rows = s1.rows
columns = s1.columns
#迭代所有的行
for i in rows:
    line = [col.value for col in i]
    print(line)
#通过坐标读取值
print(s1['A1'].value)     #A 表示列,1 表示行
#查看第 3 行第 2 列的数据
print(s1.cell(row = 3, column = 2).value)
```

运行程序,在当前目录生成一个“景点信息_openpy.xlsx”文件,输出结果如下。

```
['景点信息表', '游客表', 'Sheet']
['景点信息表', Null, Null, Null, Null]
['1', '动物园', '门票', '25', '重庆']
['2', '科技馆', '门票', '25', '重庆']
['3', '金佛山', '套票', '100', '重庆']
景点信息表
科技馆
```

5.3.4　用 Pandas 库读/写 XLSX 文件

Pandas 库的功能十分强大,在数据处理分析领域应用广泛,且同样具有对 XLSX 文件的读/写操作。

1. 安装与导入

安装 Pandas 库时,用命令行方式输入 pip install pandas,导入模块需要输入以下代码。

```
import pandas as pd
```

2. 使用方法

(1) 写入数据到 XLSX 文件。

写入数据到 XLSX 文件的方法是 to_excel('文件名')。如果将单个对象写入 Excel 文件,则必须指定目标文件名;如果将数据写入多张工作表,则需要创建一个 ExcelWriter 对

象，并通过 sheet_name 参数依次指定工作表的名称。to_ecxel 包含的参数如下。

```
df.to_excel(excel_writer, sheet_name = 'Sheet1', na_rep = '', float_format = Null, columns = Null, header = True, index = True, index_label = Null, startrow = 0, startcol = 0, engine = Null, merge_cells = True, encoding = Null, inf_rep = 'inf', verbose = True, freeze_panes = Null)
```

其中，excel_writer 表示文件路径或 ExcelWriter 对象；sheet_name 表示指定某个工作表；columns 表示需要写入的列；header 表示列名；index 表示要写入的索引；index_label 表示引用索引列的列标签；startrow 表示初始写入的行号，默认值为 0；startcol 表示初始写入的列序号，默认值为 0。engine 是一个可选参数，用于指定要使用的引擎，可以是 openpyxl 或 xlsxwriter。

① 写入数据到单个工作表。

```
import pandas as pd
data = [
    ('1',"动物园",'门票','25','重庆'),
    ('2', "科技馆", '门票', '25', '重庆'),
    ('3',"金佛山",'套票','100','重庆'),
]
df = pd.DataFrame(data)
df.to_excel('景点_单表.xlsx', sheet_name = 'Sheet1', index = False)
```

执行以上代码，可以生成一个“景点_单表.xlsx”文件，文件内容如图 5-7 所示。

	A	B	C	D	E
1	0	1	2	3	4
2	1	动物园	门票	25	重庆
3	2	科技馆	门票	25	重庆
4	3	金佛山	套票	100	重庆

图 5-7　用 Pandas 库写入单个工作表

如果要去掉列名，则需要为 to_excel()添加参数 header＝False，即可去掉第 1 行上自动生成的列名。

② 写入文件到多个工作表。

```
import pandas as pd
data1 = [
    ('1',"动物园",'门票','25','重庆'),
    ('2', "科技馆", '门票', '25', '重庆'),
    ('3',"金佛山",'套票','100','重庆')
]
data2 = [
    ('1',"张三",'186****0011'),
    ('2', "李四", '186****0012'),
    ('3',"王五",'186****0013')
]
df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)
with pd.ExcelWriter('旅游统计表.xlsx') as writer:
    df1.to_excel(writer, sheet_name = '景点信息', index = False,header = False)
    df2.to_excel(writer, sheet_name = '游客信息', index = False,header = False)
```

执行以上代码，将会在当前目录创建一个“旅游统计表.xlsx”文件，其中包含了两个工作表“景点信息”和“游客信息”，且包含相应的数据。

如果原有文件存在且文件中有数据，则使用上述方法写入工作表后，文件中原有的数据会被覆盖。为了避免发生这种情况，需要使用在原文件中新增工作表的方法。

③ 在原有文件中新增工作表。

```
import pandas as pd
data = [
    ('1',"动物园",'门票','25','重庆'),
    ('2', "科技馆", '门票', '25', '重庆'),
    ('3',"金佛山",'套票','100','重庆'),
]
df = pd.DataFrame(data)
with pd.ExcelWriter('汇总表.xlsx', mode = 'a',engine = 'openpyxl') as writer:
    df.to_excel(writer, sheet_name = 'sheet1', index = False)
```

执行以上代码，将会在“汇总表.xlsx”文件中新增一个工作表“sheet1”，内容为 data 中的数据。

(2) 读取 XLSX 文件。

读取 XLSX 文件最常用的方法是 read_excel('文件名')。如果指定某个工作表，则可以在参数中添加数字，第 1 个工作表从 0 开始编号，如 read_excel('文件名',0)表示获取第 1 个工作表对象；也可以用工作表名称进行定位，如 read_excel('文件名','sheet1')表示获取 sheet1 工作表对象。执行以下代码，将会输出“汇总表.xlsx”文件中的第 1 个工作表的信息。

```
import pandas as pd
df = pd.read_excel('汇总表.xlsx',0)
print(df)
```

除此之外，read_excel()还有其他参数。例如，io 表示文件路径；header 表示规定哪几列为列名；names 表示重新定义列名的值；index_col 表示索引列，可以用数字表示使用哪一列作为索引，也可以是列表；usecols 表示读取的列的范围，默认为所有列；converters 表示列的数据类型；nrows 表示需要读取的行数；skiprows 表示跳过的行数。这些参数可以根据需要进行选择。

(3) 读取工作表内的数据。

- 获取工作表的尺寸：df.shape。
- 获取工作表前 n 行数据：df.head(n)。
- 获取工作表第 m～n 行的数据：df[m:n]。
- 获取工作表第 m 行和第 n 行(不连续的两行)的数据：df[[m,n]]。
- 获取工作表第 m 列的数据：df.iloc[:,m]。
- 获取工作表第 m 行第 n 列的值：df.iloc[m,n]。
- 获取工作表第 m 列和第 n 列的数据：df.iloc[:,[m,n]]。
- 获取工作表第 m～n 行且在第 a～b 列的数据区域：df.iloc[[m:n],[a:b]]。

以上数字的编号均从 0 开始。一般情况下，整数索引切片是前闭后开，标签索引切片是

前闭后闭。df.loc[]只能使用标签索引，不能使用整数索引，通过标签索引切片进行筛选时，前闭后闭。df.iloc[]只能使用整数索引，不能使用标签索引，通过整数索引切片进行筛选时，前闭后开。df.ix[]既可以使用标签索引，也可以使用整数索引。

3. 基础操作案例

现有“火车信息.xlsx”和“景点信息.xlsx”两个文件，需要将两个文件合并到一个“汇总表.xlsx”文件中，并分别输出汇总表中的数据。

```
import pandas as pd
#读取两个文件中的数据
jdb = pd.read_excel('景点表.xlsx')
cpb = pd.read_excel('车票表.xlsx')
#同时写入两个文件到汇总表中
with pd.ExcelWriter('汇总表.xlsx', mode = 'w',engine = 'openpyxl') as writer:
    jdb.to_excel(writer, sheet_name = '景点表', index = False)
    cpb.to_excel(writer, sheet_name = '车票表', index = False)
#读取汇总表的两个工作表
jdb_out = pd.read_excel('汇总表.xlsx',sheet_name = '景点表')
cpb_out = pd.read_excel('汇总表.xlsx',sheet_name = '车票表')
#输出工作表的尺寸
print(jdb_out.shape)
#输出工作表前 10 行数据
print(jdb_out.head(10))
#输出工作表第 2～10 行的数据
print(jdb_out[3:10])
#输出工作表第 2 列的数据
print(jdb_out.iloc[:,1])
#输出工作表第 2 行第 2 列的值
print(jdb_out.iloc[1,1])
#整张表输出
print(jdb_out)
```

5.3.5 应用案例

1. 12306 车次信息爬取与存储

本案例实现步骤如下。

- 第 1 步：爬取数据。
- 第 2 步：存储数据。
- 第 3 步：读取数据。

其中，存储数据分别用 Pandas 库、Openpyxl 库和 xlsxwriter 库来写入，读取数据分别用 xlrd 库、Pandas 库和 Openpyxl 库来写入，以便于比较各个库之间的区别及特点。

（1）爬取数据。

爬取数据的实现代码如下。

```
import time
from selenium import webdriver
from lxml import etree
base_url = r'https://kyfw.12306.cn/otn/leftTicket/init?linktypeid = dc&fs = %E9%87%8D%
E5%BA%86,CQW&ts = %E5%8C%97%E4%BA%AC,BJP&date = 2023 - 03 - 24&flag = N,N,Y'
```

```
print("正在爬取数据 …… 请等待 …… ")
options = webdriver.FirefoxOptions()
#设置浏览器为 headless 无界面模式
options.add_argument("--headless")
options.add_argument("--disable-gpu")
#打开浏览器处理,注意浏览器无显示
browser = webdriver.Firefox(options = options)
browser.get(base_url)
time.sleep(4)
res = browser.page_source
html = etree.HTML(res)
#车次
result1 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[1]/
div/a/text()')
#始发站
result2 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[2]/
strong[1]/text()')
#到达站
result3 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[2]/
strong[2]/text()')
#出发时间
result4 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[3]/
strong[1]/text()')
#到达时间
result5 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[3]/
strong[2]/text()')
#历时
result6 = html.xpath('/html/body/div[2]/div[8]/div[8]/table/tbody/tr/td[1]/div/div[4]/
strong/text()')
#输出车次信息
print('----共计{0}个车次信息,分别是:----'.format(len(result1)))
for x in range(0,len(result1)):
    print('车次:',result1[x],'始发站:',result2[x],'到达站:',result3[x],'出发时间:',result4
[x],'到达时间:',result5[x],'历时:',result6[x])
print('----爬取的车次信息,显示完成----')
#等待 3s,关闭浏览器
time.sleep(3)
browser.close()
```

(2) 存储数据。

① 用 Pandas 库实现存储。

```
import pandas as pd
df = pd.DataFrame({'车次':result1,'始发站':result2,'终点站':result3,'出发时间':result4,'到达
时间':result5,'历时':result6})
df.to_excel('火车查询信息'+time.strftime("-%m%d-%H%M%S", time.localtime())+'.xlsx
',index = False)
print('----爬取的车次信息,存储到 XLSX 中----')
```

② 用 xlsxwriter 库实现存储。

```
import xlsxwriter
workbook = xlsxwriter.Workbook('车票 writer.xlsx')
worksheet1 = workbook.add_worksheet()
```

```
#录入标题
worksheet1.write(0, 0, '车次')
worksheet1.write(0, 1, '始发站')
worksheet1.write(0, 2, '终点站')
worksheet1.write(0, 3, '始发时间')
worksheet1.write(0, 4, '到达时间')
worksheet1.write(0, 5, '用时')
#定义数据
l1,l2,l3,l4,l5,l6 = [],[],[],[],[],[]
for i in range(len(result1)):
    l1.append(result1[i])
    l2.append(result2[i])
    l3.append(result3[i])
    l4.append(result4[i])
    l5.append(result5[i])
    l6.append(result6[i])
l = [l1,l2,l3,l4,l5,l6]
l = list(map(list, zip(*l)))
#写入数据
for i in range(1,len(l)+1):
    for j in range(6):
        worksheet1.write_string(i,j,l[i-2][j])
#关闭保存
workbook.close()
```

③ 用 Openpyxl 库实现存储。

```
import openpyxl
#创建工作簿文件
cc = openpyxl.Workbook()
#激活当前工作表
js = cc.active
#创建一个 sheet1
s1 = cc.create_sheet("景点表",0)
#写入小标题
title = ['车次','始发站','到达站','始发时间','到达时间','用时']
for col in range(len(title)):
    c = col + 1
    s1.cell(row = 1,column = c).value = title[col]
#输入数据
l1,l2,l3,l4,l5,l6 = [],[],[],[],[],[]
for i in range(len(result1)):
    l1.append(result1[i])
    l2.append(result2[i])
    l3.append(result3[i])
    l4.append(result4[i])
    l5.append(result5[i])
    l6.append(result6[i])
data = [l1,l2,l3,l4,l5,l6]
data = list(map(list, zip(*data)))
rows = len(data)
l = len(data[0])
for i in range(rows):
```

```
    for j in range(l):
        s1.cell(row = i + 2, column = j + 1).value = data[i][j]
cc.save("火车信息_openpy.xlsx")
```

(3) 读取数据。

① 用 xlrd 库读取 XLSX 文件。

```
#导入 xlrd 模块
import xlrd
#定义一个 XLSX 文件对象
data = xlrd.open_workbook(r"车票 writer.xlsx ")
#查询工作表名称
names = data.sheet_names()
#获取第 1 个工作表
table = data.sheet_by_index(0)
#获取表格行数
nrows = table.nrows
print("表格一共有", nrows, "行")
#获取表格列数
nclos = table.ncols
print("表格一共有", nclos, "列")
#遍历表数据
for i in range(nrows):
    print(table.row_values(i))
```

② 用 Openpyxl 库读取 XLSX 文件。

```
import openpyxl
#打开工作簿文件
jd = openpyxl.load_workbook("车票 writer.xlsx")
js = jd.active
# 获取所有表格(worksheet)的名字
sheets = jd.sheetnames
print(sheets)
#获取工作表
s1 = jd['Sheet1']
#获取表格所有行和列,两者都是可迭代的
rows = s1.rows
columns = s1.columns
#迭代所有的行
for i in rows:
    line = [col.value for col in i]
    print(line)
```

③ 用 Pandas 库读取 XLSX 文件。

```
import pandas as pd
hc = pd.read_excel('火车信息_openpy.xlsx',sheet_name = '景点表')
print(hc)
```

对 12306 车次信息进行爬取与存储,最终生成的 XLSX 文件如图 5-8 所示。

	A	B	C	D	E	F
1	车次	始发站	终点站	始发时间	到达时间	用时
2	K508	重庆西	北京西	21:00	21:34	24:34
3	G52	重庆北	北京西	07:32	14:26	06:54
4	G352	重庆北	北京西	08:15	17:03	08:48
5	G388	重庆西	北京西	08:38	20:03	11:25
6	T10	重庆西	北京西	09:56	11:12	25:16
7	G332	重庆北	北京西	10:48	19:06	08:18
8	Z96	重庆西	北京西	11:25	10:51	23:26
9	G372	重庆西	北京西	11:57	21:34	09:37
10	G372	重庆北	北京西	12:24	21:34	09:10
11	G54	重庆北	北京西	14:33	21:44	07:11
12	Z50	重庆北	北京西	14:37	10:05	19:28
13	Z4	重庆北	北京西	15:24	10:11	18:47
14						

图 5-8　车次信息 XLSX 文件

2. 携程网景点信息爬取与存储

本案例实现步骤为 3 步：爬取数据、存储数据和读取数据。

(1) 爬取数据。

爬取数据的实现代码如下。

```
import requests
import time
from bs4 import BeautifulSoup
# ==== 景点信息爬取部分 =============
if __name__ == '__main__':
    #通过观察网页,生成景点信息前 10 页的网址
    headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36'}
    url_list = []
    for x in range(1,11):
        url = 'https://you.ctrip.com/sight/chongqing158/s0-p{0}.html#sightname'.format(x)
        url_list.append(url)
    #开始爬取景点信息
    i = 1
    list1 = []
    for base_url in    url_list:
        print('======== 开始爬取第{0}页信息,共计 10 个景点 ======== '.format(i))
        #添加 2s 延时
        time.sleep(2)
        res = requests.get(base_url, headers = headers)
        res.encoding = res.apparent_encoding
        soup = BeautifulSoup(res.text, 'lxml')
        res = soup.find_all(class_ = 'list_mod2')
        for x in res:
            #景点名称
            res1 = x.find_all('a')
            #景点地址
            res2 = x.find_all(class_ = 'ellipsis')
            #景点热度
            res3 = x.find_all(class_ = 'hot_score_number')
```

```
            #景点评分
            res4 = x.find_all(class_ = 'score')
            #景点点评数
            res5 = x.find_all(class_ = 'recomment')
            #显示信息
            print('-' * 30)
            print('景点名称:',res1[1].string.strip())
            print('景点地址:',res2[0].string.strip())
            print('景点热度:',res3[0].string.strip())
            print('景点评分:',res4[0].find_all('strong')[0].string.strip())
            print('景点点评数:',res5[0].string.strip())
            #将数据存入字典后,加入列表中
            dist1 = {}
            dist1['景点名称'] = res1[1].string.strip()
            dist1['景点地址'] = res2[0].string.strip()
            dist1['景点热度'] = res3[0].string.strip()
            dist1['景点评分'] = res4[0].find_all('strong')[0].string.strip()
            dist1['景点点评数'] = res5[0].string.strip().strip('条点评()')
            time.sleep(0.4)
            list1.append(dist1)
            time.sleep(0.4)
        print('======== 第{0}页信息,爬取完成 ======== '.format(i))
        i = i + 1
    print('======== 景点信息前 10 页信息,爬取完成 ======== '.format(i))
```

(2) 存储数据。

① 用 Pandas 库实现存储。

```
import pandas as pd
#将数据保存到 Execl 表格中
name = '景点信息' + '.xlsx'
df = pd.DataFrame(list1)
df.to_excel(name)
print('----- 爬取的景点信息,存储到 Excel 中 ---- ')
```

执行以上代码,会生成一个“景点信息.xlsx”文件。

② 用 xlsxwriter 库实现存储。

用 xlsxwriter 库也可以实现同样的存储效果,代码如下。

```
import xlsxwriter
workbook = xlsxwriter.Workbook('景点 writer.xlsx')
worksheet1 = workbook.add_worksheet()
#录入标题
worksheet1.write(0, 0, '景点名称')
worksheet1.write(0, 1, '景点地址')
worksheet1.write(0, 2, '景点热度')
worksheet1.write(0, 3, '景点评分')
worksheet1.write(0, 4, '景点点评数')
#定义数据
```

```
l1, l2, l3, l4, l5, l6 = [], [], [], [], [], []
for i in range(len(list1)):
    l1.append(list1[i].get("景点名称"))
    l2.append(list1[i].get("景点地址"))
    l3.append(list1[i].get("景点热度"))
    l4.append(list1[i].get("景点评分"))
    l5.append(list1[i].get("景点点评数"))
l = [l1,l2,l3,l4,l5]
l = list(map(list, zip(*l)))
#写入数据
for i in range(1,len(l)+1):
    for j in range(5):
        worksheet1.write(i,j,l[i-1][j])
#关闭保存
workbook.close()
```

③ 用 Openpyxl 库实现存储。

用 Openpyxl 库也可以实现同样的存储效果,代码如下。

```
import openpyxl
# ============写入部分==============
#创建工作簿文件
cc = openpyxl.Workbook()
#激活当前工作表
js = cc.active
#创建一个 sheet1
s1 = cc.create_sheet("景点表",0)
#写入小标题
title = ['景点名称','景点地址','景点热度','景点评分','景点点评数']
for col in range(len(title)):
    c = col + 1
    s1.cell(row = 1,column = c).value = title[col]
#输入数据
l1, l2, l3, l4, l5, l6 = [], [], [], [], [], []
for i in range(len(list1)):
    l1.append(list1[i].get("景点名称"))
    l2.append(list1[i].get("景点地址"))
    l3.append(list1[i].get("景点热度"))
    l4.append(list1[i].get("景点评分"))
    l5.append(list1[i].get("景点点评数"))
l = [l1,l2,l3,l4,l5]
l = list(map(list, zip(*l)))
#写入数据
rows = len(l)
le = len(l[0])
for i in range(rows):
    for j in range(le):
        s1.cell(row = i +2, column = j + 1).value = l[i][j]
cc.save("景点_openpy.xlsx")
```

（3）读取数据。

① 用 xlrd 库读取 XLSX 文件。

```
#导入 xlrd 模块
import xlrd
#定义一个 XLSX 文件对象
data = xlrd.open_workbook(r"景点_openpy.xlsx ")
#查询工作表名称
names = data.sheet_names()
#获取第 1 个工作表
table = data.sheet_by_index(0)
#获取表格行数
nrows = table.nrows
print("表格一共有", nrows, "行")
#获取表格列数
nclos = table.ncols
print("表格一共有", nclos, "列")
#遍历表数据
for i in range(nrows):
    print(table.row_values(i))
```

② 用 Openpyxl 库读取 XLSX 文件。

```
import openpyxl
#打开工作簿文件
jd = openpyxl.load_workbook("景点_openpy.xlsx")
js = jd.active
#获取所有表格(worksheet)的名字
sheets = jd.sheetnames
print(sheets)
#获取工作表
s1 = jd['景点表']
#获取表格所有行和列,两者都是可迭代的
rows = s1.rows
columns = s1.columns
#迭代所有的行
for i in rows:
    line = [col.value for col in i]
    print(line)
```

③ 用 Pandas 库读取 XLSX 文件。

```
import pandas as pd
jd = pd.read_excel('景点信息.xlsx',index_col = False)
print(jd)
```

对携程网景点信息进行爬取与存储，最终生成的 XLSX 文件如图 5-9 所示。

	A	B	C	D	E
1	景点名称	景点地址	景点热度	景点评分	景点点评数
2	武隆天生三	重庆市武隆	8.3	4.7	8661
3	洪崖洞民俗	重庆市渝中	8.7	4.6	6446
4	长江索道	重庆市渝中	8.5	4.2	9262
5	重庆欢乐谷	重庆市渝北	7.8	4.5	5253
6	磁器口古镇	重庆市沙坪	8.5	4.4	10702
7	武隆喀斯特	重庆市武隆	8.3	4.6	3264
8	重庆动物园	重庆市九力	7.2	4.7	2433
9	解放碑步行	重庆市渝中	8.3	4.6	4022
10	重庆两江游	重庆市渝中	7.7	4.3	7826
11	重庆1949	重庆市沙坪	7.1	4.8	267
12	仙女山国家	重庆市武隆	7.7	4.5	3102
13	重庆云端之	重庆市渝中	7.7	4.3	440
14	乐和乐都动	重庆市永川	6.7	4.8	5890
15	大足石刻	重庆市大足	7.4	4.6	4293
16	渣滓洞	重庆市沙坪	7.3	4.5	820
17	重庆融创茂	重庆市沙坪	5.8	4.8	1676
18	丰都鬼城名	重庆市丰都	7.2	4.3	1987
19	白帝城.瞿	奉节县夔门	7.2	4.5	1356
20	龙水峡地缝	武隆县仙女	6.9	4.6	1433
21	朝天门广场	重庆市渝中	7.2	4.5	1037
22	梦幻奥陶纪	重庆綦江区	7.0	4.2	3026
23	湖广会馆	重庆市渝中	7.0	4.5	902
24	万州大瀑布	重庆市万州	7.0	4.8	2570
25	巫山小三峡	重庆市巫山	7.0	4.4	1100
26	南山风景区	重庆市南岸	7.0	4.6	265

图 5-9　景点信息 XLSX 文件

5.4　数据库存取

在进行旅游大数据分析时，根据数据的结构，可以选择关系数据库和非关系数据库来存储数据。关系数据库能够实现复杂的数据查询，且有事务支持，可以实现数据存储的高安全性，目前 MySQL 在数据分析领域应用较为广泛。非关系数据库则主要存储非结构化的数据，如文本、图片、音频和视频等数据，常用的有 MongoDB、Redis 等。在大数据时代数据类型多且数据增长快的情况下，非关系数据库发展迅速，但是关系数据库凭借其高可靠性、高效率的数据管理优势，依然是主流数据库。因此，下面主要介绍关系数据库的使用方法。

5.4.1　数据模型

数据库使用数据模型对现实世界进行抽象化，数据模型是对数据和数据之间联系的描述。现有的数据库系统均是基于某种数据模型的。常见的数据模型有 3 种：层次模型、网状模型和关系模型。

1. 层次模型

层次模型使用树状结构表示实体及实体间的联系，满足以下两个条件的数据模型称为层次模型。

（1）有且仅有一个结点无父结点，该结点为根结点。

（2）其他结点有且仅有一个父结点。

2. 网状模型

网状模型使用网状结构表示实体及实体间的联系，满足以下两个条件之一的数据模型称为网状模型。

(1) 允许一个以上的结点无父结点。

(2) 允许结点可以有多于一个的父结点。

3. 关系模型

关系模型使用一组二维表表示实体及实体间的关系，它将世界看作是由实体和联系构成的。联系就是实体之间的关系，可以分为 3 种：一对一、一对多、多对多。

关系模型的特点如下。

(1) 表中的每列都是不可再细分的基本数据项。

(2) 每列的名称不同，数据类型相同或兼容。

(3) 行的顺序无关紧要。

(4) 列的顺序无关紧要。

(5) 关系中不能存在完全相同的两行。

5.4.2　关系数据库的基本概念与运算

在使用关系数据库之前，需要先了解它的基本概念和关系之间的运算。

1. 基本概念

(1) 关系：一个关系对应一张二维表，每个关系有一个关系名。

(2) 记录：表中的一行为一条记录，记录也称为元组。

(3) 属性：表中的一列为一个属性，属性也称为字段。每一个属性都有一个名称，即属性名。

(4) 关键字：表中的某个属性集，它可以唯一确定一条记录。

(5) 主键：一个表中可能有多个关键字，但在实际的应用中只能选择一个，被选用的关键字称为主键。

(6) 值域：属性的取值范围。

2. 关系运算

对关系数据库进行查询时，若要找到用户关心的数据，就需要进行一定的关系运算。关系运算有两种：一种是传统的集合运算(如并、差、交、广义笛卡儿积等)；另一种是专门的关系运算(如选择、投影、连接等)。

专门的关系运算的概念如下。

(1) 选择：在关系中选择满足指定条件的元组。

(2) 投影：在关系中选择某些属性(列)。

(3) 连接：从两个关系的笛卡儿积中选取属性间满足一定条件的元组。

5.4.3　关系数据库设计

在创建关系数据库之前，不论基于什么平台，都需要先设计关系数据库，其设计步骤为以下 4 步。

1. 建立 E-R(实体-关系)模型

将现实世界抽象化为信息世界是设计数据库的第一步。将现实世界中客观存在的事物及其所具有的特征抽象为信息世界的实体和属性,进而绘制出 E-R 图。绘图时涉及的实体、属性、实体标识符、联系及联系类型的概念如下。

(1) 实体:客观存在并可以相互区分的事物称为实体,用矩形表示。

(2) 属性:实体所具有的某一特性,用椭圆表示,并用连线与实体相连接。

(3) 实体标识符:能唯一标识实体的属性或属性组合。

(4) 联系:实体与实体之间的联系。联系的类型有一对一、一对多、多对多。用菱形框表示,并用连线与有关实体相连接。

例如,在设计旅游数据库时,将现实世界的旅游景点和游客抽象为景点和游客两个实体。景点实体具有景点编号、景点名称、景点位置、项目编号、项目名称、项目标价、项目折扣、旅游荐点(周边游、亲子游、团建游)等属性,游客实体具有身份证号、姓名、年龄、籍贯、手机号、家庭人数等属性。每位游客可以选多个景点,每个景点可以有多名游客旅游,游客与景点之间的联系是多对多,因此用游客旅游表作为游客和景点之间的联系名,并设置游客编号、游客姓名、游客年龄、景点编号、项目名称、旅游时间、消费金额、同行人数、交通方式、旅游形式(报团、自驾、周边游)等属性。

根据以上分析,绘制出旅游数据库 E-R 图,如图 5-10 所示。

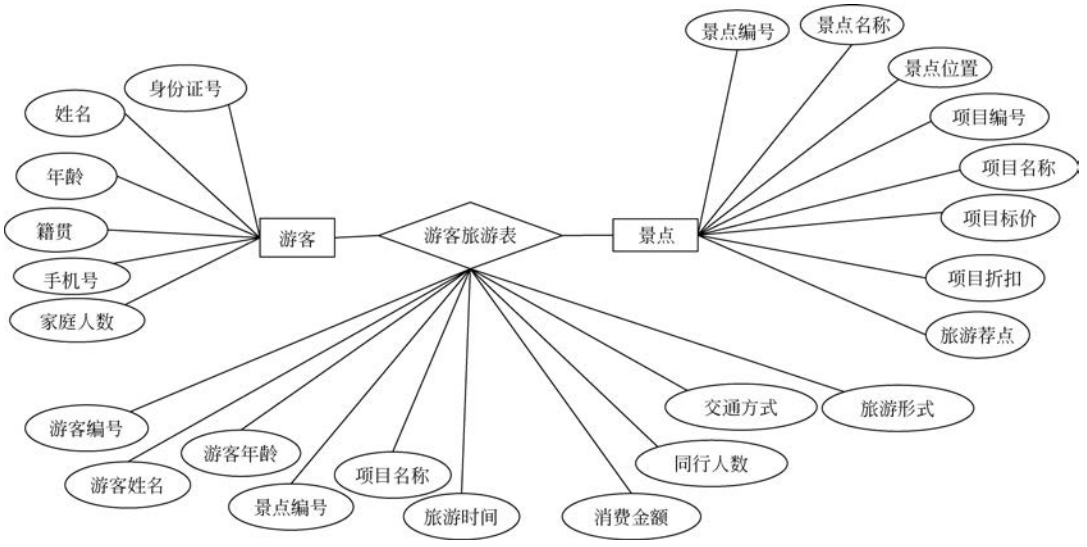

图 5-10 旅游数据库 E-R 图

2. 将 E-R 模型转换为关系模型

将第 1 步绘制的 E-R 图转换为二维表的形式,即可得到关系模型。将上述案例中的 E-R 图转换为关系模型后,得到以下 3 个表。

表 5-2 游客表(原)

身份证号	姓名	年龄/岁	籍贯	手机号	家庭人数/人
50011…3121	张三	20	重庆	18688888888	3
…	…	…	…	…	…

表 5-3　景点表(原)

景点编号	景点名称	景点位置	项目编号	项目名称	项目标价	项目折扣	旅游荐点
CQ001	动物园	重庆主城	DWY001	门票	25	20	亲子游
CQ001	黔江濯水古镇	重庆黔江	ZS001	游船	75	50	周边游
…	…	…	…	…	…	…	…

表 5-4　游客旅游表(原)

游客编号	游客姓名	游客年龄/岁	景点编号	项目名称	旅游时间	消费金额/元	同行人数/人	交通方式	旅游形式
yk0001	张三	30	CQ001	门票	2023/5/1	100	4	汽车	自驾
yk0001	张三	30	CQ001	游船	2023/5/2	300	4	火车+大巴	跟团
…	…	…	…	…	…	…	…	…	…

3. 对关系模型进行规范化

对第 2 步所得的 3 个二维表进行规范。规范化的目的是消除存储异常,减少数据冗余,保证数据完整性和存储效率。对于不同的规范化程序,可用“范式”来衡量,记作 NF。一般规范化可实现第三范式。

(1) 第一范式。

定义:一个关系的每个属性都是不可再分的基本数据项。

经分析,表 5-1、表 5-2 和表 5-3 三个表都符合第一范式。

为了理解第二范式和第三范式,需要了解函数依赖、部分函数依赖和函数传递依赖的概念。

① 函数依赖:完全依赖或部分依赖于主关键字。

例如,景点表中的景点名称、景点地址、旅游项目等都函数依赖于主关键字景点编号。

② 部分函数依赖:表中某属性只函数依赖于主关键字中的部分属性。

例如,游客旅游表中的旅游时间依赖于主关键字的游客编号,也依赖于主关键字中的景点编号,它完全函数依赖主关键字(游客编号、景点编号);而景点名称只函数依赖于主关键字(游客编号、景点编号)中的景点编号,它与游客编号无关,是部分函数依赖。

③ 函数传递依赖:属性之间存在传递的函数依赖关系。

例如,景点表中的景点编号决定项目编号,项目编号决定项目名称。如果项目名称通过项目编号的传递而依赖于主关键字景点编号,则称项目名称和景点编号之间存在函数传递依赖关系。

(2) 第二范式。

定义:首先是第一范式,并且关系中的每个非主属性完全函数依赖(而不是部分依赖)于主关键字。

将非第二范式转换为第二范式:提取部分函数依赖关系中的主属性(决定方)和非主属性从关系中提取,使其单独构成一个关系;将关系中余下的其他属性加主关键字,从而构成关系。例如,游客旅游表中的景点名称部分函数依赖于主关键字(游客编号、景点编号)中的景点编号,需要对景点名称和景点编号进行分离。

(3) 第三范式。

定义:首先是第二范式,并且关系中的任何一个非主属性都不函数传递依赖于任何主

关键字。

消除函数传递依赖关系的方法如下。

例如，对景点表中的项目编号、项目名称、项目标价和折扣价进行分离，使其单独组成一个关系，并将剩余的景点编号、景点名称、项目编号构成一个关系表。

经过以上 3 个范式的规范，将上文中的 3 个表最终规范为 5 个表，如表 5-5～表 5-9 所示。

表 5-5　游客表

身份证号	姓名	年龄/岁	籍贯	手机号	家庭人数
50011…3121	张三	20	重庆	18688888888	3
…	…	…	…	…	…

表 5-6　景点表

景点编号	景点名称	景点位置	旅游荐点
CQ001	动物园	重庆主城	亲子游
CQ002	黔江濯水古镇	重庆黔江	周边游
…	…	…	…

表 5-7　项目表

项目编号	项目名称	项目标价/元	折扣价/元
DWY001	门票	25	20
ZS001	游船	75	50
…	…	…	…

表 5-8　景点项目表

景 点 编 号	项 目 编 号
CQ001	DWY001
CQ001	DWY001
…	…

表 5-9　游客旅游表

游客编号	景点编号	项目编号	旅游时间	消费金额/元	同行人数/人	交通方式	旅游形式
yk0001	CQ001	DWY001	2023/5/1	100	4	汽车	自驾
yk0001	CQ002	ZS001	2023/5/2	300	4	火车＋大巴	跟团
…	…	…	…	…	…	…	…

4. 数据完整性

规范后的多个二维表组成了数据库的主要内容，在此基础上，还需要保证数据完整性，才能创建数据库。数据完整性的规则体现在以下几方面。

(1) 列(域)完整性：表中的每一列数据都必须满足所定义的数据类型，并且其值在有效范围之内。

(2) 表完整性：表中必须有一个主关键字(不能为 NULL)。

(3) 参照完整性：每两个关联的表中的数据必须是一致的、协调的，主关键字与外关键

字也必须是一致的、协调的。

(4) 在从表中进行插入操作时,要保证外关键字的值一定在主表中存在。

(5) 在主表中修改主关键字值时,要在从表中同步修改,或者禁止修改主表。

(6) 在从表中修改外关键字值时,要保证修改的值在主表中存在。

(7) 在删除主表记录时,要注意从表中是否引用主关键字。若存在引用,则禁止删除或同步删除从表记录。

5.4.4 SQL 语句

结构化查询语言 SQL 是操作关系数据库的工业标准语言。在 SQL 中,有以下 4 类语言。

(1) 数据查询语言(Data Query Language,DQL):SELECT。

(2) 数据操纵语言(Data Manipulation Language,DML):INSERT、UPDATE、DELETE。

(3) 数据定义语言(Data Definition Language,DDL):CREAT、ALTER、DROP。

(4) 数据控制语言(Data Control Language,DCL):GRANT、REVOKE。

这些语句是非常重要的,特别是在用 Visual Basic、Power Builder 等工具开发数据库应用程序时,这些语句是操作数据库的重要途径。

1. 常用运算符及函数

(1) 常用运算符。

运算符是表示实现某种运算的符号,一般分为 4 类:算术运算符、关系运算符、逻辑运算符和字符串运算符。表 5-10 列出了常用运算符,其中,Like 通常与"?""*""#"等通配符结合使用,主要用于模糊查询。在这 3 种通配符中,"?"表示任何单一字符;"*"表示零个或多个字符;"#"表示任何一个数字(0~9)。

表 5-10 常用运算符

类 型	运 算 符
算术运算符	+ * / ^(乘方) \(整除) MOD(取余数)
关系运算符	< <= <> > >= Between Like
逻辑运算符	NOT AND OR
字符串运算符	&

(2) 常用函数。

在 SQL 语句中可以使用大量的函数。表 5-11 列出了 SQL 语句中的常用函数。

表 5-11 常用内部函数和聚合函数

函数类型	函 数 名	说 明
内部函数	Date()	返回系统日期
	Year()	返回年份
	AVG()	计算某一列的平均值
	COUNT(*)	统计记录的个数
	COUNT(列名)	统计某一列值的个数
	SUM(列名)	计算某一列的总和

续表

函数类型	函 数 名	说 明
聚合函数	MAX(列名)	计算某一列的最大值
	MIN(列名)	计算某一列的最小值
	FIRST(列名)/LAST(列名)	分组查询时,选择同一组中第一条(或最后一条)记录在指定列上的值,将其作为查询结果中现在应记录在该列上的值

2. 创建数据库

创建一个数据库的 SQL 语句为 CREATE DATABASE db_name。

例如,要创建一个旅游系统(tourism system)的数据库,编写 SQL 语句如下。

```
CREATE DATABASE tousys
```

其他常用操作如下。

(1) 查看数据库:show db_name。

(2) 选择指定数据库:use db_name。

(3) 删除数据库:drop database db_name。

3. 创建数据表

创建数据表的 SQL 语句如下。

```
CREATE TABLE 表名
(列名称 1 数据类型,
列名称 2 数据类型,
列名称 3 数据类型,
…)
```

常见的数据类型如下。

(1) int(size):整型。在括号内规定数字的最大位数。

(2) decimal(size,d):容纳带有小数的数字。“size”规定数字的最大位数,“d”规定小数点右侧的最大位数。

(3) char(size):容纳固定长度的字符串(可容纳字母、数字和特殊字符)。在括号中规定字符串的长度。

(4) varchar(size):容纳可变长度的字符串(可容纳字母、数字和特殊字符)。在括号中规定字符串的最大长度。

(5) date(yyyymmdd):容纳日期。

例如,创建一个用户信息表,编写 SQL 语句如下。

```
CREATE TABLE Persons
(Id_P int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255))
```

又如,创建一个景点信息表,编写 SQL 语句如下。

```
CREATE TABLE jdxxb
(jdbh int,jdmc varchar(255),jdrd varchar(255),Address varchar(255),jddps varchar(255),
PRIMARY KEY (jdbh ))
```

该景点信息表的表结构如表 5-12 所示。

表 5-12 jdxxb 表结构

列名称(jdbh)	数据类型(int)
jdmc	varchar(255)
jdrd	varchar(255)
address	varchar(255)
jddps	varchar(255)

在创建 jdxxb 表时,最后一行的 PRIMARY KEY (jdbh)表示设置 jdbh 为表的主键。此外,设置主键也可以直接在定义列名时写入,如 jdbh int PRIMARY KEY。

删除数据表的 SQL 语句为 DROP TABLE 表名,清空数据表的 SQL 语句为 DELETE FROM 表名。

4. 插入语句(INSERT)

插入数据的 SQL 语句如下。

```
INSERT INTO 表名[(字段 1,字段 2,…,字段 n)]
VALUES(常量 1,常量 2,…,常量 n)
```

例如,在 jdxxb 表中插入数据。

```
INSERT INTO jdxxb VALUES (
(1001,"动物园",8.9,"重庆",8900),
(1002,"科技馆",9.3,"重庆",6000)
)
```

5. 查询语句(SELECT)

数据查询是数据库的核心操作,SELECT 语句如下。

```
SELECT [ALL|DISTINCT]目标列 FROM 表(或查询)
[WHERE 条件表达式]
[GROUP BY 列名 1[ HAVING 过滤表达式]]
[ORDER BY 列名 2[ ASC|DESC]]
```

在 SELECT 语句中,选择目标列部分是最基本的、不可缺少的,属于基本语句;其余部分是可以省略的,称为子句。

整个语句的功能是,根据 WHERE 子句中的表达式,从 FROM 子句指定的表或查询中找出满足条件的记录,再按 SELECT 子句中的目标列显示数据。SELECT 语句是数据查询语句,不会更改数据库中的数据。

对 SELECT 语句的进一步分析如下。

(1) 查询多行或多列。

在水平方向上查询满足条件的行的记录,称为选择;在垂直方向上查询满足条件的列

的记录，称为投影。

基本语句：SELECT[ALL|DISTINCT]目标列 FROM 表 WHERE 条件。

功能：从 FROM 子句指定的表或查询中找出满足条件的记录，再按照目标列或行显示数据。

SELECT 语句的一个简单用法如下。

```
SELECT 列名 1,…,列名 n FROM 表 WHERE 条件
```

如果要修改目标里的显示名称，则可以在列名后添加“AS 别名”；如果要查询所有列的内容，则用“*”表示。DISTINCT 表示去除查询结果中的重复值。除此之外，目标列中的列名也可以是一个使用聚合函数的表达式。如果没有 GROUP BY 子句，则对整个表进行统计，整个表只产生一条记录；否则，进行分组统计，一组产生一条记录。

例如，从景点信息表(jdxxb)中查询所有重庆的景点信息。

```
SELECT * FROM jdxxb whereaddress = "重庆"
```

若要查询所有的景点名称列(jdmc)，则 SELECT 语句为

```
SELECTjdmc FROM jdxxb
```

(2) 排序查询结果。

ORDER BY 子句用于指定查询结果的排列顺序。ASC 表示升序，DESC 表示降序，默认是升序。ORDER BY 可以指定多个列作为排序关键字。

例如，从景点信息表(jdxxb)中查询所有重庆的景点信息，并按景点编号从大到小降序排序。

```
SELECT * FROM jdxxb
WHERE address = "重庆"
ORDER BY jdbh DESC
```

(3) 分组查询。

GROUP BY 子句用于对查询结果进行分组，即将在某一列上值相同的记录分在一组，一组产生一条记录。

例如，在景点信息表(jdxxb)中查询所有地区的景点信息。

```
SELECT * FROM jdxxb GROUP BY address
```

GROUP BY 后可以有多个列名，分组时把在这些列上值相同的记录分在一组。

例如，在景点信息表(jdxxb)中查询所有地区的景点信息，并计算每个地区的平均点评数。

```
SELECT address, AVG(jddps) FROM jdxxb GROUP BY address
```

若要对分组后的结果进行筛选，则可以使用 HAVING 子句。

例如，在景点信息表(jdxxb)中查询所有地区的景点信息，计算每个地区的平均点评数

并筛选出平均点评数大于 6000 的结果。

```
SELECT address, AVG(jddps) FROM jdxxb GROUP BY address HAVING AVG(jddps)> 6000
```

HAVING 子句与 WHERE 子句都是用来设置筛选条件的，二者的区别在于，HAVING 子句是在分组统计之后进行过滤，而 WHERE 子句是在分组统计之前进行选择记录。

(4) 多表查询。

在查询关系数据库时，有时需要的数据分布在几个表或视图中，此时需要按照某个条件将这些表或视图连接起来，形成一个临时的表，然后再对该临时表进行简单查询。

例如，有两个表 jdxxb(jdbh，jdmc，jdrd，address，jddps)和 jddpb(jdbh，user，jdpf)，分别存储了景点信息表(景点编号、景点名称、景点热度、景点地址、景点点评数)和景点点评表(景点编号、用户、景点评分)，查询景点编号为 10 的景点评分。

```
SELECT jdbh, jdmc,address,AVE(jdpf)
FROM jdxxb,jddpb
WHERE jdxxb.jdbh = jddpb.jdbh
GROUP BY jdbh, jdmc,address, jdpf
```

6. 删除语句(DELETE)

在 SQL 中，DELETE 语句用于数据删除，其语法格式如下。

```
DELETE FROM 表[ WHERE 条件]
```

DELETE 语句用于从表中删除满足条件的记录。如果 WHERE 子句为默认，则删除表中所有的记录，此时表没有被删除，仅仅删除了表中的数据。

例如，删除表 jdxxb 中所有地址为“重庆”的记录。

```
DELETE * from jdxb where address = "重庆"
```

7. 修改语句(UPDATE)

在 SQL 中，UPDATE 语句用于数据修改，其语法格式如下。

```
UPDATE 表 SET 字段 1 = 表达式 1,…,字段 n = 表达式 n  [ WHERE 条件]
```

UPDATE 语句用于修改指定表中满足条件的记录，并按表达式的值修改相应的值。如果 WHERE 子句为默认，则修改表中的所有记录。

例如，将表 jdxxb 中地址为“重庆”的数据改为“西南”。

```
UPDATE jdxxb SET address = "西南" WHERE address = "重庆"
```

需要注意的是，UPDATE 语句一次只能对一个表进行修改，这就有可能破坏数据库中数据的一致性。因此，更新数据时需要考虑相关联的表格因素。

8. 其他语句

在某些特殊情况里需要使用其他 SQL 语句。例如，修改表的结构，使用 ALTER TABLE 语句；授权用户，使用 GRANT 语句；回收授权，使用 REVOKE 语句。

5.4.5 在 Python 中操作 MySQL

1. 安装 MySQL 数据库

在 Python 中调用 MySQL 数据库，首先需要安装 MySQL 数据库。

登录 https://www.mysql.com/cn/downloads/，下载 MySQL Community，下载后的文件名为 mysql-installer-community-8.0.32.0.msi。对数据库进行运行安装，具体步骤如下。

(1) 选择 Custom 安装方式，单击 Next。

(2) 在 Select Products 选项中，选择 MySQL Server8.0.32-64、Connector/ODBC 8.0.32-X64 和 Connector/NET 8.0.32 -X86，分别添加到右侧安装列，如图 5-12 所示。修改 Advanced Option，将安装目录和数据目录分别改为 D 盘和 E 盘，如图 5-13 所示。

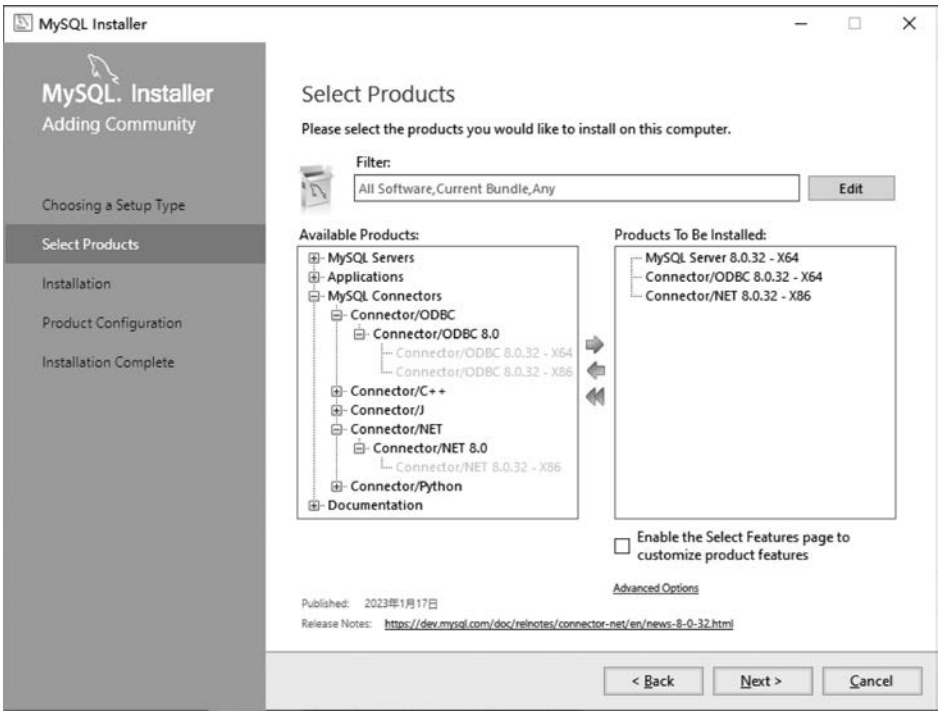

图 5-12 选择安装文件

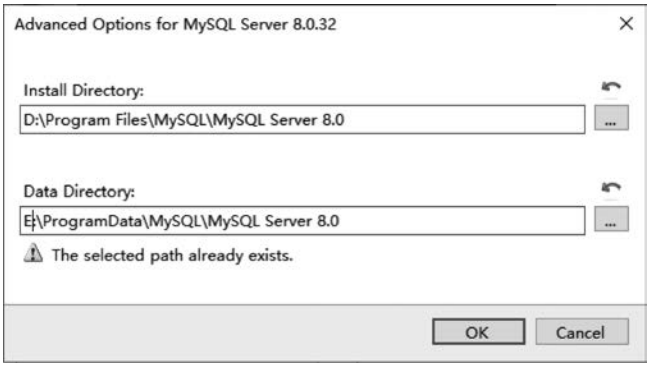

图 5-13 修改安装目录和数据目录

（3）在 Installation 中，执行上述选择的 3 个产品安装即可。

（4）在 Products Configuration 中对软件进行配置。在 Type and Networking 界面，对配置类型和网络连接进行设置，如图 5-14 所示。在 Named Pipe 界面，选择 Full access to all users(NOT RECOMMENDED)。在 Authentication Methord 界面，选择第 1 个选项。在 Accounts and Roles 界面，可以设置默认用户名是 root 的管理员账户，也可以添加自定义账户，具体操作如图 5-15 和图 5-16 所示。

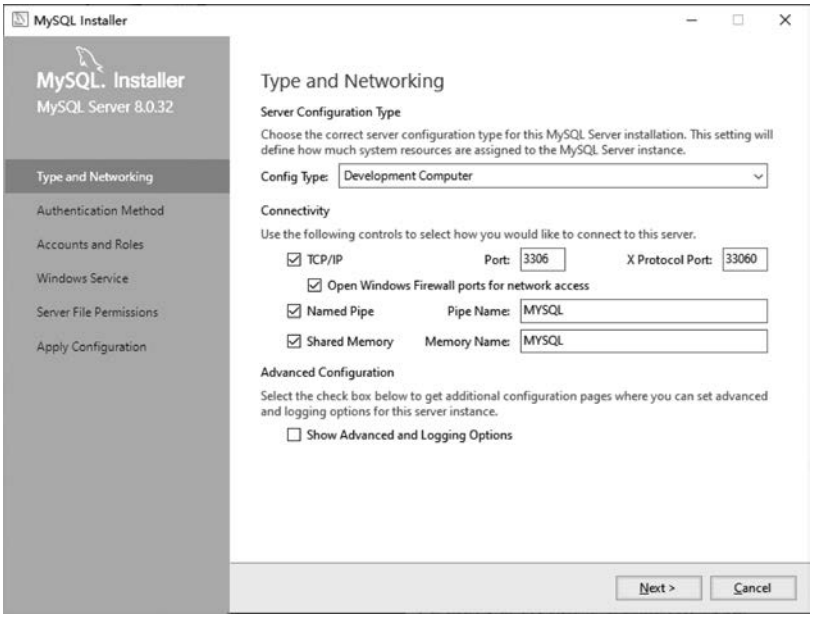

图 5-14　设置配置类型和网络连接

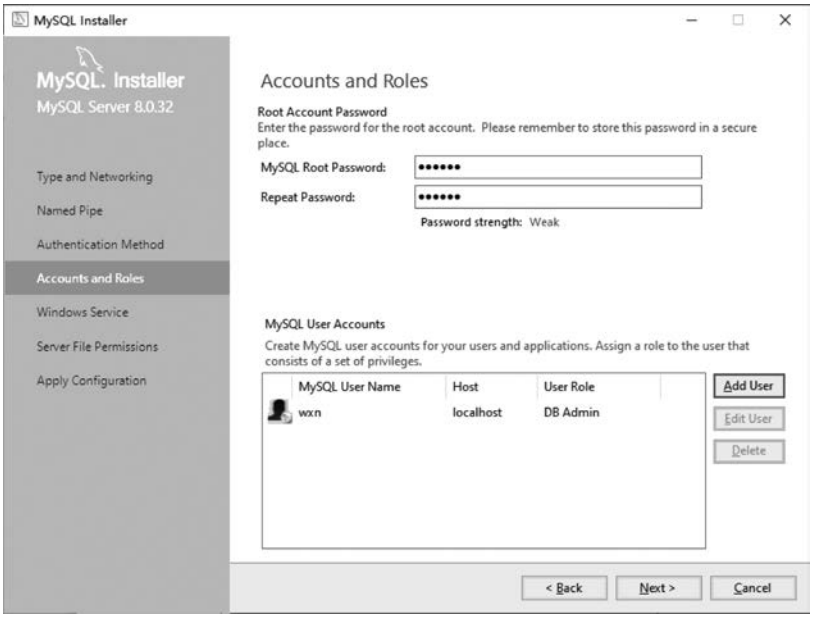

图 5-15　创建管理员账户

图 5-16　添加自定义账户

接下来跟随软件提示进行操作，直到出现 Installation Complete 界面，此时安装完成。

安装后可以进入开始菜单，查找 MySQL 8.0 Command Line Client 并运行，在命令框中输入设置好的管理员账户密码，如果看到如图 5-17 所示的效果，则提示安装成功。

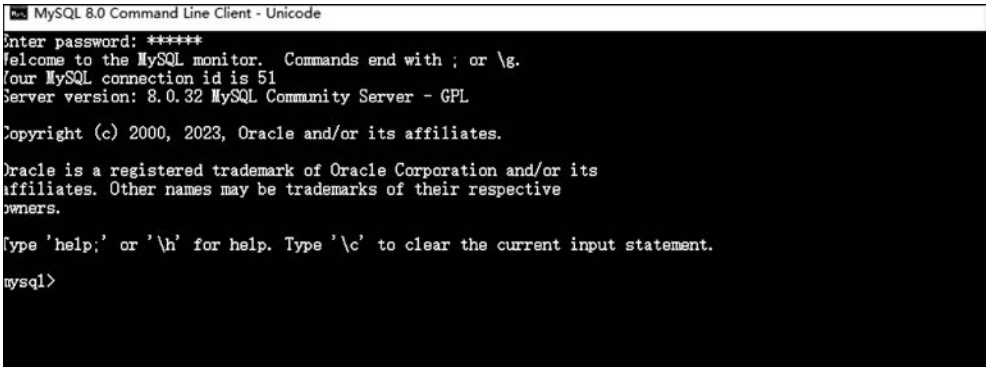

图 5-17　测试连接

如果数据库未安装成功，则需要找到 MySQL 的安装路径，并在计算机的系统高级设置中配置环境变量，如图 5-18 所示。

2. 用 pymysql 库操作 MySQL 数据库

pymysql 库是在 Python 中操作数据库的一个第三方库。因此，除了掌握 SQL 语句外，还要掌握 pymysql 库的操作语法，才能操作 MySQL 数据库。安装 pymysql 库后，输入以下代码导入库。

```
import pymysql
```

在 Python 中执行 SQL 语句的步骤如下。

(1) 创建数据库的连接对象。

(2) 创建游标对象获取当前游标。

(3) 使用游标对象中的 execute()或 fetchall()等方法，执行某个 SQL 语句或获取数据。

(4) 关闭连接，释放资源。

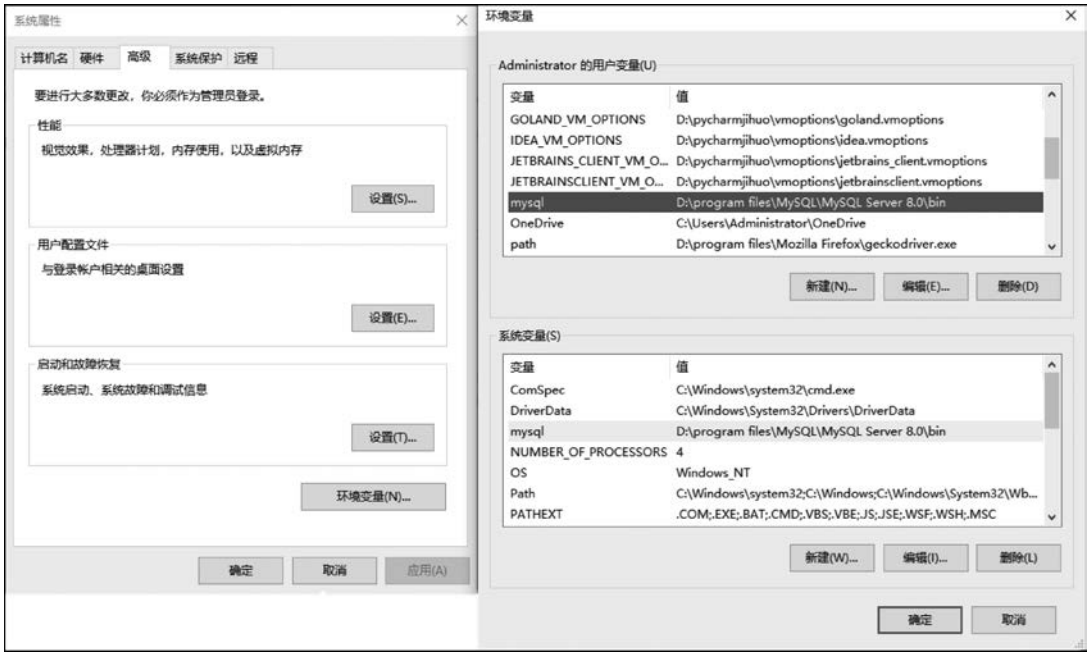

图 5-18　配置环境变量

对上述步骤的进一步分析如下。

(1) 创建连接对象。

在创建数据库的连接对象时，可以输入以下代码，以此获取本机数据库的连接对象。如果服务器不在本机，则需要将 host 设置为服务器的 IP 地址，代码如下。

```
mydb = pymysql.connect(host = "127.0.0.1",user = "wxn",password = "123456")
```

执行以下代码，测试与 MySQL 数据库是否连接成功。

```
import pymysql
try:
    mydb = pymysql.connect(
        host = "127.0.0.1",user = "wxn",password = "123456")
    print("wxndb 连接成功")
except Exception as err:
    print("wxndb 连接失败")
```

(2) 获取游标。

从字面上理解，游标是指流动的标志。使用游标功能，可以存储 SQL 语句的执行结果，并提供一个游标接口给用户。在需要获取数据时，可以直接从游标中获取。

创建游标对象 mycursor，并获取当前数据库对象的游标，代码如下。

```
cursor = mydb.cursor()
```

(3) 使用游标方法执行 SQL 语句或获取数据。

游标的常用方法有 execute(SQL 语句)、fetchone()、fetchall()等。

① 用 execute()可以执行 SQL 语句,执行后要用 commit()进行提交。

② 用 fetchone()可以获取一条数据,用 fetchall()可以获取所有数据。这两个函数主要是用于将获取到的数据赋值给变量,便于之后的调用。

(4) 关闭连接。

在数据库操作结束后,断开数据库并释放资源,代码如下。

```
cursor.close()
```

3. 基础操作案例

创建旅游系统数据库(data base),其中包含 5 个表,分别是游客表(身份证号、姓名、年龄、籍贯、手机号、家庭人数)、景点表(景点编号、景点名称、景点位置、旅游荐点)、项目表(项目编号、项目名称、项目标价、折扣价)、景点项目表(景点编号、项目编号)、游客旅游表(游客编号、景点编号、项目编号、旅游时间、消费金额、同行人数、交通方式、旅游形式)。创建成功后录入数据,查看数据库中的内容,并删除数据库。

该案例的实现步骤如下。

(1) 导入 pymysql 库并获取账户连接。

```
import pymysql
#准备工作:获取账户连接
myconn = pymysql.connect(host = "127.0.0.1",user = "wxn",password = "123456")
```

(2) 创建数据库。

```
#定义 SQL 语句
sql1 = '''
create database if not exists toursys'''
#获取游标
mycursor = myconn.cursor()
#执行 SQL 语句
mycursor.execute(sql1)
```

(3) 获取数据库的连接对象。

```
myconn = pymysql.connect(host = "127.0.0.1",user = "wxn",password = "123456",database =
"toursys")
```

(4) 创建 5 个数据表。

```
#第 1 个:游客表
sql1 = '''create table if not exists ykb(ykbh varchar(18) primary key,sfzh varchar(18),xm
varchar(10),nl varchar(3),jg varchar(10),sj char(11),jtrs varchar(3))'''
mycursor = myconn.cursor()
mycursor.execute(sql1)
#第 2 个:景点表
sql2 = '''create table if not exists jdb(jdbh varchar(18) primary key,jdmc varchar(18),jdwz
varchar(18),lyjd varchar(10))'''
mycursor = myconn.cursor()
mycursor.execute(sql2)
```

```
#第 3 个:项目表
sql3 = '''create table if not exists xmb(xmbh varchar(18) primary key,xmmc varchar(18),xmbj
varchar(10),zkj varchar(10))'''
mycursor = myconn.cursor()
mycursor.execute(sql3)
#第 4 个:景点项目表
sql4 = '''create table if not exists jdxmb(jdbh varchar(18),xmbh varchar(18),primary key(jdbh,
xmbh))'''
mycursor = myconn.cursor()
mycursor.execute(sql4)
#第 5 个:游客旅游表
sql5 = '''create table if not exists yklyb(ykbh varchar(18),jdbh varchar(18),xmbh varchar(18),
lysj date,xfje varchar(10),yxrs varchar(3),jtfs varchar(10),lyxs char(11))'''
mycursor = myconn.cursor()
mycursor.execute(sql5)
```

(5) 插入数据。

```
#第 1 个:游客表
sqlin1 = '''insert into ykb values( %s, %s, %s, %s, %s, %s, %s)'''
# try:
mycursor. execute ( sqlin1, [ " 1001 "," 500114233533332312 "," 张 三 "," 20 "," 重 庆 ",
"18688888888","5"])
mycursor. execute ( sqlin1, [ " 1002 "," 500114233533332322 "," 李 四 "," 25 "," 成 都 ",
"18688888888","10"])
mycursor. execute ( sqlin1, [ " 1003 "," 500114233533332332 "," 王 五 "," 40 "," 西 藏 ",
"18688888888","3"])
myconn.commit()
# except:
    print('插入数据失败')
#第 2 个:景点表
sqlin2 = '''insert into jdb values( %s, %s, %s, %s)'''
try:
    mycursor.execute(sqlin2,["CQ001","动物园","重庆主城","亲子游"])
    mycursor.execute(sqlin2,["CQ002","黔江濯水古镇","重庆黔江","周边游"])
    mycursor.execute(sqlin2,["CQ003","大足石刻","重庆大足","周边游"])
    myconn.commit()
except:
    print('插入数据失败')
#第 3 个:项目表
sqlin3 = '''insert into xmb values( %s, %s, %s, %s)'''
try:
    mycursor.execute(sqlin3,["DWY001","门票",25,25])
    mycursor.execute(sqlin3,["ZS001","游船",75,50])
    myconn.commit()
except:
    print('插入数据失败')
#第 4 个:景点项目表
sqlin4 = '''insert into jdxmb values( %s, %s)'''
```

```
try:
    mycursor.execute(sqlin4,["CQ001","DWY001"])
    mycursor.execute(sqlin4,["CQ001","DWY002"])
    myconn.commit()
except:
    print('插入数据失败')
#第5个:游客旅游表
sqlin5 = '''insert into yklyb values( %s, %s, %s, %s, %s, %s, %s, %s)'''
mycursor.execute(sqlin5,["yk0001","CQ001","DWY001","2023.5.1","100",4,"汽车","自驾"])
mycursor.execute(sqlin5,["yk0001","CQ002","DWY001","2023.5.2","300",3,"火车+大巴","跟团"])
myconn.commit()
```

(6) 查询数据。

```
myconn = pymysql.connect(host = "127.0.0.1", user = "wxn", password = "123456", database = 'toursys')
mycursor = myconn.cursor()
#全表查询
sqlsel1 = '''select * from ykb;'''
mycursor.execute(sqlsel1)
myconn.commit()
#获取一条数据
result1 = mycursor.fetchone()
print(result1)
#获取所有数据
data = mycursor.fetchall()
print(data)
#将数据转换为DataFrame
from pandas import DataFrame
df = DataFrame(data)
print(df)
```

(7) 删除数据库。

```
sql4 = "drop database toursys;"
mycursor.execute(sql4)
myconn.commit()
```

(8) 关闭连接。

```
myconn.close()
```

在PyCharm中的输出结果如下,查询的是游客表的所有信息。

```
     0             1              2   3   4          5         6
0  1001  500114233533332312  张三  20  重庆  18688888888   5
1  1002  500114233533332322  李四  25  成都  18688888888  10
2  1003  500114233533332332  王五  40  西藏  18688888888   3
```

在 MySQL 程序的执行过程中，输入“show databases”可以查看所有的数据库，输入“use toursys”可以选择 toursys 数据库，输入“select * from ykb”可以查看游客表的信息。数据库写入结果如图 5-19 所示。

```
mysql> use toursys;
Database changed
mysql> select * from ykb;
+------+--------------------+------+------+------+-------------+------+
| ykbh | sfzh               | xm   | nl   | jg   | sj          | jtrs |
+------+--------------------+------+------+------+-------------+------+
| 1001 | 500114233533332312 | 张三 | 20   | 重庆 | 18688888888 | 5    |
| 1002 | 500114233533332322 | 李四 | 25   | 成都 | 18688888888 | 10   |
| 1003 | 500114233533332332 | 王五 | 40   | 西藏 | 18688888888 | 3    |
+------+--------------------+------+------+------+-------------+------+
3 rows in set (0.00 sec)

mysql> select * from xmb;
+--------+--------+------+------+
| xmbh   | xmmc   | xmbj | zkj  |
+--------+--------+------+------+
| DWY001 | 门票   | 25   | 25   |
| ZS001  | 游船   | 75   | 50   |
+--------+--------+------+------+
2 rows in set (0.00 sec)
```

图 5-19　数据库写入结果

5.4.6　应用案例

案例实现效果：爬取携程网上前 5 页的重庆景点信息（景点名称、景点地址、景点热度、点评分数、景点点评数），以及每个景点的点评信息（用户名、点评内容、点评分数、点评时间、IP 属地），将这些信息插入 MySQL 数据库并对数据库进行查询。

由于爬取数据较多，因此需要先将爬取到的数据存储为 CSV 文件。为了便于初学者理解，又将 CSV 文件转换为 XLSX 文件，用 Excel 软件进行查看。然后创建数据库和数据表，读取 XLSX 文件中的数据并将其存入数据库，以此实现对数据库中的表信息进行查看。

1. 爬取数据

（1）爬取景点信息。

爬取景点信息的代码如下。

```
import requests
import time
from selenium import webdriver
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By
from bs4 import BeautifulSoup
if __name__ == '__main__':
    #通过观察网页,生成景点信息前 5 页的网址
    headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36'}
    url_list = []
    for x in range(1,6):
        url = 'https://you.ctrip.com/sight/chongqing158/s0-p{0}.html#sightname'.format(x)
        url_list.append(url)
    #开始爬取景点信息
    i = 1
```

```
    #将信息组合成字典,再存入列表,为存储做准备
    list1 = []
    for base_url in    url_list:
        print('======== 开始爬取第{0}页信息,共计 10 个景点 ======== '.format(i))
        #添加 2s 延时
        time.sleep(2)
        res = requests.get(base_url, headers = headers)
        res.encoding = res.apparent_encoding
        soup = BeautifulSoup(res.text, 'lxml')
        res = soup.find_all(class_ = 'list_mod2')
        for x in res:
            #景点名称
            res1 = x.find_all('a')
            #景点地址
            res2 = x.find_all(class_ = 'ellipsis')
            #景点热度
            res3 = x.find_all(class_ = 'hot_score_number')
            #点评分数
            res4 = x.find_all(class_ = 'score')
            #景点点评数
            res5 = x.find_all(class_ = 'recomment')
            #景点链接
            res6 = x.find_all(class_ = 'recomment')
            #显示信息
            print('-' * 30)
            print('景点链接:', res1[0]['href'])
            print('景点名称:',res1[1].string.strip())
            print('景点地址:',res2[0].string.strip())
            print('景点热度:',res3[0].string.strip())
            print('点评分数:',res4[0].find_all('strong')[0].string.strip().strip('()条点评'))
            print('景点点评数:',res5[0].string.strip())
            #将数据存入字典后,加入列表中
            dist1 = {}
            dist1['景点链接'] = res1[0]['href']
            dist1['景点名称'] = res1[1].string.strip()
            dist1['景点地址'] = res2[0].string.strip()
            dist1['景点热度'] = res3[0].string.strip()
            dist1['点评分数'] = res4[0].find_all('strong')[0].string.strip()
            dist1['景点点评数'] = res5[0].string.strip().strip('()条点评')
            time.sleep(0.4)
            list1.append(dist1)
            time.sleep(0.4)
        print('======== 第{0}页信息,爬取完成 ======== '.format(i))
        i = i + 1
    print('======== 景点信息前 5 页信息,爬取完成 ======== '.format(i))
```

(2) 爬取景点评论。

爬取景点评论的代码如下。

```
list2 = []
for x in list1:
    base_url = x['景点链接']
```

```
        jd_name = x['景点名称']
        print(" ======== 开始爬取({0})景点的评论 ===== ".format(jd_name))
        options = webdriver.FirefoxOptions()
        #设置浏览器为 headless 无界面模式
        options.add_argument(" -- headless")
        options.add_argument(" -- disable - gpu")
        #打开浏览器处理,注意浏览器无显示
        browser = webdriver.Firefox(options = options)
        browser.get(base_url)
        print("正在获取数据 …… 请稍等 …… .")
        time.sleep(2)
        #循环获取 5 页评价
        for x in range(1):
            print("第{0}页数据加载中,请稍等 …… ".format(x + 1))
            time.sleep(5)
            print(' --------- 正在获取第{0}页数据 --------- '.format(x + 1))
            url1 = browser.current_url
            res = browser.page_source
            soup = BeautifulSoup(res, 'lxml')
            res = soup.find_all(class_ = 'commentItem')
            print(len(res))
            for x in res:
                #用户名
                result1 = x.find_all(class_ = 'userName')
                #评分
                result2 = x.find_all(class_ = 'averageScore')
                #评语
                result3 = x.find_all(class_ = 'commentDetail')
                #点评时间
                l2 = x.find_all(class_ = 'commentTime')
                # IP 属地
                l3 = x.find_all(class_ = 'ipContent')
                print('用户名:', result1[0].string, '点评时间:',l2[0].text[0:10], l3[0].text,
'评分:',result2[0].text[0:3] , '评语:',result3[0].string)
                dist2 = {}
                dist2['景点名称'] = jd_name
                dist2['用户名'] = result1[0].string
                dist2['点评时间'] = l2[0].text[0:10]
                dist2['IP 属地'] = l3[0].text[5:]
                dist2['评分'] = result2[0].text[0:3]
                dist2['评语'] = result3[0].string
                list2.append(dist2)
            #换页操作
            #获取底部下一页
            canzhao = browser.find_elements(By.CLASS_NAME, 'seotitle1')
            nextpage = browser.find_elements(By.CLASS_NAME,'ant - pagination - item - comment')
            time.sleep(2)
            #移动到元素 element 对象的"顶端"与当前窗口的"底部"对齐
            browser.execute_script("arguments[0].scrollIntoView(false);", canzhao[0])
            time.sleep(2)
            #鼠标移至下一页
            ActionChains(browser).move_to_element(nextpage[1]).perform()
            time.sleep(2)
```

```
        #鼠标单击下一页
        nextpage[1].click()
        time.sleep(4)
    #数据爬取完成,关闭浏览器
    print('--------- 获取数据完成 --------- ')
    time.sleep(4)
    browser.close()
```

2. 存储数据

(1) 将数据存储为 CSV 文件。

将数据存储为 CSV 文件的代码如下。

```
import pandas as pd
#将数据保存到 CSV 文件中
df1 = pd.DataFrame(list1)
df1.to_csv('景点数据表.csv)
df2 = pd.DataFrame(list2)
df2.to_csv('景点点评表.csv)
print('---- 爬取的景点信息,存储到文件中 ---- ')
```

(2) 将 CSV 文件转换为 Excel 文件。

将 CSV 文件转换为 Excel 文件的代码如下。

```
import pandas as pd
jd = pd.read_csv('景点数据表.csv',index_col = False, encoding = 'utf - 8')
dp = pd.read_csv('景点点评表.csv',index_col = False, encoding = 'utf - 8')
#将 CSV 文件转换为 Excel 文件,并将数据保存在创建的表格 data 中
jd.index = jd.index + 1
dp.index = dp.index + 1
jd.to_excel('景点数据表.xlsx',index_label = '景点编号',columns = ['景点链接','景点名称','景点
地址','景点热度','点评分数','景点点评数'])
dp.to_excel('景点点评表.xlsx',index_label = '评价编号',columns = ["景点名称","用户名","评价
时间","IP 属地","评分","评语"])
```

3. 存入数据库

(1) 从 XLSX 文件中读取数据。

从 XLSX 文件中读取数据的代码如下。

```
import pandas as pd
list1 = pd.read_excel('景点数据表.xlsx')
list2 = pd.read_excel("景点点评表.xlsx")
```

(2) 获取连接。

获取连接的代码如下。

```
#获取到 database 的连接对象
print('=' * 40)
print('将爬取到的数据插入 MySQL 数据库')
print('=' * 40)
```

```
import pymysql
myconn = pymysql.connect(host = "127.0.0.1",user = "wxn",password = "123456")
```

(3) 创建数据库。

创建数据库的代码如下。

```
#创建数据库
sql_crate = '''create database if not exists toursys'''
mycursor = myconn.cursor()
mycursor.execute(sql_crate)

#获取到 database 的连接对象
myconn = pymysql.connect(host = "127.0.0.1", user = "wxn", password = "123456", database =
'toursys')
```

(4) 创建数据表。

创建数据表的代码如下。

```
#创建数据表(景点信息表,景点点评表)
sql1 = '''create table     if not exists jdxxb(id int primary key auto_increment,jdname varchar
(100), jdlj varchar(100), jdaddress varchar(100), jdhot varchar(20), jdscore varchar(20),
jdnumber varchar(20))'''
mycursor = myconn.cursor()
mycursor.execute(sql1)
sql2 = '''create table     if not exists jddpb(id int primary key auto_increment,jdname varchar
(100),pname varchar(100), content varchar(160), score varchar(20), dptime varchar(20), ip
varchar(100))'''
mycursor = myconn.cursor()
mycursor.execute(sql2)
```

(5) 插入数据。

插入数据的代码如下。

```
#将数据存入 MySQL 数据库
print('='*40)
print('查看存入 MySQL 数据库的数据')
print('='*40)
#添加数据
sql3 = '''insert into jdxxb(id,jdname,jdlj,jdaddress,jdhot,jdscore,jdnumber) values(Null,%s,
%s,%s,%s,%s,%s)'''
for x in range(len(list1['景点名称'])):
    try:
        mycursor.execute(sql3,[list1['景点名称'][x],list1['景点链接'][x],list1['景点地址']
[x],str(list1['景点热度'][x]),str(list1['点评分数'][x]),str(list1['景点点评数'][x])])
        print(list1['景点名称'][x],'数据插入成功')
        myconn.commit()
    except:
        print(list1['景点名称'][x],'==景点信息==插入数据失败')
sql4 = '''insert into jddpb(id,jdname,pname,content,score,dptime,ip) values(Null,%s,%s,
%s,%s,%s,%s)'''
```

```
for y in range(len(list2['景点名称'])):
    try:
        mycursor.execute(sql4,[list2['景点名称'][y],list2['用户名'][y],list2['评语'][y],
list2['评分'][y],list2['评价时间'][y],list2['IP属地'][y]])
        print(list2['景点名称'][y],'数据插入成功')
        myconn.commit()
    except:
        print(list2['景点名称'][y],'--景点评价信息--插入数据失败')
```

4. 查询数据库

(1) 查询数据库中的景点信息。

查询数据库中的景点信息的代码如下。

```
sql5 = '''select * from jdxxb;'''
mycursor.execute(sql5)
myconn.commit()
#获取所有数据
data = mycursor.fetchall()
print(data)
```

景点信息查询结果如图 5-20 所示,所有的景点信息输出成功。

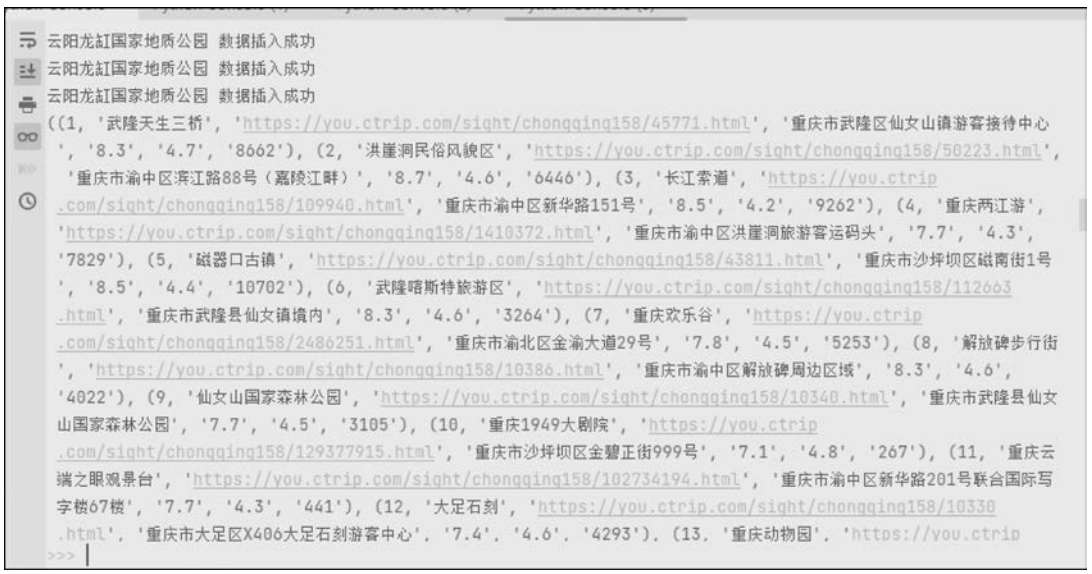

图 5-20 景点信息查询结果

(2) 查询数据库中的评价信息。

查询数据库中的评价信息的代码如下。

```
sql6 = '''select * from jddpb;'''
mycursor.execute(sql6)
myconn.commit()
#获取所有数据
data1 = mycursor.fetchall()
```

```
print(data1)
#关闭连接
myconn.close()
```

点评信息查询结果如图 5-21 所示。

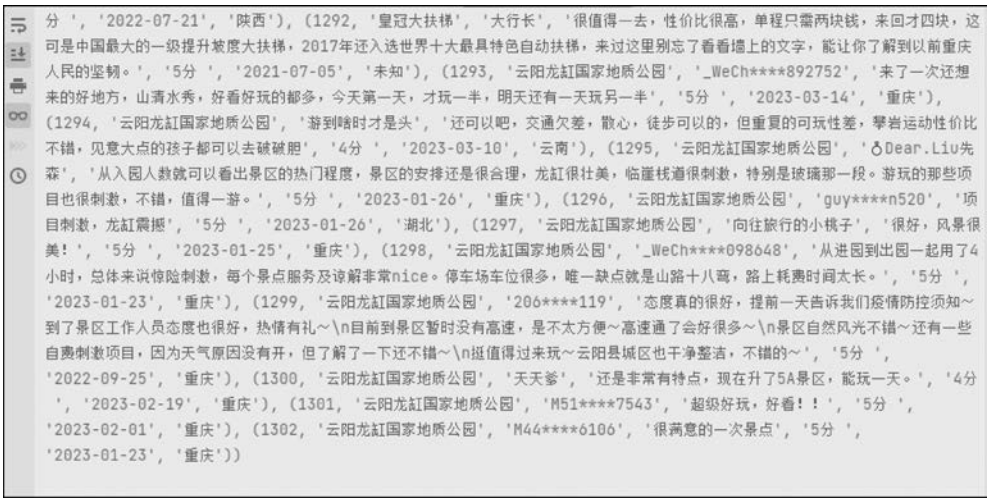

图 5-21　点评信息查询结果