第4章 Python财务数据采集

当我们准备进行财务数据分析时，首先就要获取相应的财务数据。

1. 本地文件读取

1.1 使用 `pandas` 读取 Excel 表格数据

`pandas` 是处理数据的强大工具，可轻松读取 Excel 表格。

读取 Excel 文件：

```python
import pandas as pd
# pd.set_option('display.max_rows', 10) # 最大显示10行
# pd.set_option('display.max_columns', 10) # 最大显示10列

# 读取 Excel 文件
file_path = "数据.xlsx"
df = pd.read_excel(file_path)
df
```

读取大型 Excel

```python
df2 = pd.read_excel('销售发票列表.xlsx')
df2
```

1.2 使用 `pandas` 读取 CSV 表格数据

CSV（Comma Separated Values）是一种轻量级的表格数据格式，非常适合数据交换和存储。

读取 CSV 文件：

读取小型 CSV

```python
# 将df2转换为CSV
df2.to_csv('销售发票列表.csv', index=False)

# 读取CSV文件
df3 = pd.read_csv('销售发票列表.csv')
df3
```

读取大型 CSV

```python
# 读取金融交易数据集
transactions_df = pd.read_csv('金融交易数据集/transactions_data.csv')
transactions_df

# 按merchant_city分组，并计算每个城市的交易次数
result = transactions_df[["merchant_city","zip"]].groupby("merchant_city").sum()
result
```

🚀 Excel vs CSV

特性	Excel	CSV
文件格式	二进制	纯文本
文件大小	较大	较小
兼容性	需要专门软件支持	任何文本编辑器均可
读取速度	较慢	较快
数据结构	可有多个工作表	仅一张表

可视化和公式支持	内建图表、公式等	无公式，无图表
数据交换	复杂格式，兼容性较差	通用，交换方便
用途	财务报表、格式复杂的报表	数据交换、数据分析

1.3 使用 pandas 读取 Parquet 表格数据

Parquet 是一种列式存储格式，常用于大数据处理场景。相比于 CSV，Parquet 格式具有以下优势：

- **压缩率更高**：减少磁盘空间占用。
- **读取速度更快**：按列读取数据，更适合分析型任务。
- **数据类型更丰富**：支持嵌套数据结构。

在 Python 中，`pandas` 提供了方便的方法来读取和写入 Parquet 文件：

```python
# 将数据保存到 Parquet 文件中 优化数据存储格式
# 使用 Parquet 存储数据，读写性能更高，文件更小
# 一般使用 engine='pyarrow' 和 compression='snappy'，因为 PyArrow 性能更好，而 Snappy 压缩在速度和文件大小之间取得了很好的平衡。
transactions_df.to_parquet("金融交易数据集/transactions_data.parquet", engine="pyarrow",compression="snappy")

# 读取 Parquet 文件
df_parquet = pd.read_parquet("金融交易数据集/transactions_data.parquet", engine='pyarrow')
df_parquet

# 按merchant_city分组，并计算每个城市的交易次数
df_parquet[["merchant_city","zip"]].groupby("merchant_city").sum()
```

🚀 Parquet vs CSV

特性	Parquet	CSV
存储方式	列式存储	行式存储
文件大小	较小（压缩后）	较大
读取速度	快速（按列读取）	较慢
数据类型支持	丰富，支持嵌套结构	仅支持基础数据类型

兼容性	主要用于大数据生态	兼容性强，可直接打开查看

✅ **结论：**

- 如果数据量较小且需与其他软件交换数据，用 CSV。
- 如果处理大数据、做数据分析或模型训练，推荐用 Parquet。

1.3 使用 pandas 读取 MySQL 数据库

🎯 **安装必要库**

- **SQLAlchemy**：Python 的 ORM 框架，用于处理数据库连接、查询、映射等操作。
- **PyMySQL**：一个用于连接 MySQL 的驱动。

可以使用 `create_engine()` 函数来建立与 MySQL 数据库的连接：

```python
from sqlalchemy import create_engine

# MySQL 数据库的连接字符串
engine = create_engine("mysql+pymysql://用户名:密码@主机:端口/数据库名")

# 示例
engine = create_engine("mysql+pymysql://root:123456@localhost:3306/financial_data")
```

连接字符串格式：

```bash
dialect+driver://username:password@host:port/database
```

- **dialect**：数据库类型，比如 `mysql`
- **driver**：驱动程序，比如 `pymysql`
- **username**：数据库用户名
- **password**：数据库密码
- **host**：数据库服务器地址，`localhost` 表示本地数据库
- **port**：MySQL 默认端口是 `3306`
- **database**：要连接的数据库名

完整代码：

```python
'''
pip install sqlalchemy -i https://pypi.mirrors.ustc.edu.cn/simple/
pip install pymysql -i https://pypi.mirrors.ustc.edu.cn/simple/
'''

import pandas as pd
from sqlalchemy import create_engine

# 初始化数据库连接，使用pymysql模块

# MySQL的用户名，密码，主机，端口，数据库：financial_data
engine = create_engine("mysql+pymysql://root:123456@localhost:3306/financial_data")

'''
select * from `资产负债表` WHERE ts_code = '600519.sh' AND end_date = 20201231 AND update_flag = 1
select * from `资产负债表` WHERE ts_code = '600519.sh' AND end_date in (20181231, 20191231, 20201231) AND update_flag = 1
'''

sql = """

    select * from `资产负债表`
    WHERE ts_code = '600519.sh'
    AND end_date = 20201231
    AND update_flag = 1

"""

# read_sql_query的两个参数: sql语句,  数据库连接
df = pd.read_sql_query(sql, engine)

# 输出查询结果
df
```

1.4 使用 `pdfplumber` 读取 PDF 数据

PDF 文件常用于不可编辑的文本和表格展示，`pdfplumber` 是提取 PDF 内容的利器。

读取基本信息：

```python
import pdfplumber
import pandas as pd

file_path = "贵州茅台2020年年度报告.pdf"

# 读取PDF
f = pdfplumber.open(file_path)

# 读取PDF文档信息
f.metadata

# 读取PDF页面信息
f.pages

# 关闭PDF
# f.close()

# 总页数
len(f.pages)


# 读取指定页
page = f.pages[7]

# 查看页码
print('页码: ', page.page_number)

# 查看页宽
print('页宽: ', page.width)

# 查看页高
print('页高: ', page.height)

# .objects / .chars / .lines / .rects / .curves / .images
page.rects
```

提取文本和表格:

```python
# 读取文本
text = page.extract_text()  # keep_blank_chars=True时，将所有空白字符保留为文字字符
print(text)

# 读取第一个表格
table = page.extract_table()
table

table_df = pd.DataFrame(table[1:], columns=table[0])
table_df

# 读取所有表格
tables = page.extract_tables()
tables

table_df = pd.DataFrame(tables[0][1:], columns=tables[0][0])
table_df
```

✅ **常见用途**：

- 提取合同文本、报告数据
- 读取财务报表中的表格内容

⚠️ **注意事项**：

- 如果 PDF 是图片格式（扫描件），需要结合OCR库进行文字识别。

2. 财经大数据接口

Welcome to AKShare's Online Documentation! — AKShare 1.16.32 文档

当我们对上市公司财务数据进行分析时，就需要获取大量的历史数据，采用传统的数据采集方法很难满足分析需求，所以当我们学习了Python后，就可以直接通过第三方数据接口Akshare采集财务数据，其特点是数据覆盖范围广，接口调用简单,响应快速。

根据其官网说明，Akshare是一个基于Python的财经数据接口库，也是一套从数据采集、数据清洗到数据落地的工具，这些数据包括股票、期货、期权、基金、外汇、债券、指数、加密货币等金融产品的基本面数据、实时和历史行情数据、衍生数据。Akshare主要用于财经研究，它能够解决财经研究过程中的数据获取问题，其获取的是相对权威的财经数据网站公布的原始数据，通过利用原始数据进行各数据源之间的交叉验证，进行再加工，从而得出科学的结论。

Akshare的原理就是在用户本地运行Python代码，实时从网络采集数据到本地，便利与数据分析。由于网络数据采集需要维护的接口众多，且经常由于目标网站变换网页格式需要维护及更新相关接口，所以用户在使用Akshare的过程中需要定期更新至最新版本，同时也需要关注官方文档的更新，因为最新的使用方式和接口变更都会第一时间更新到文档中。

Akshare提供最佳的文档支持，每个数据接口均提供详细的说明和示例，只需要复制粘贴就可以下载数据。这些丰富的数据接口,便于用户简便快速的获取各类金融数据、财务数据。

我们这里使用Akshare提供的stock_financial_report_sina接口获取上市公司财务报表数据，单次可获取指定报表所有年份的历史数据，其数据来源于新浪财经官网。

语法格式：

```python
stock_financial_report_sina(stock, symbol)
```

参数说明：

```
stock：类型为str，stock="sh600600"；带市场标识（如上证sh、深证sz等）的股票代码
symbol：类型为str，symbol="现金流量表"；choice of {"资产负债表", "利润表", "现金流量表"}
```

【例】通过Akshare接口依次获取上市公司贵州茅台（sh600519）的三大财务报表历史数据。

第1步，获取贵州茅台的资产负债表历史数据。

示例代码：

```python
import akshare as ak # 导入akshare模块
pd.set_option('max_rows', 10) # 最大显示10行
pd.set_option('max_columns', 10) # 最大显示10列

# 获取资产负债表数据
df = ak.stock_financial_report_sina(stock="sh600519", symbol="资产负债表")
df
```

第2步，获取贵州茅台的利润表历史数据。

示例代码：

```python
# 获取利润表数据
df = ak.stock_financial_report_sina(stock="sh600519", symbol="利润表")
df
```

第3步，获取贵州茅台的现金流量表历史数据。

示例代码：

```python
# 获取现金流量表数据
df = ak.stock_financial_report_sina(stock="sh600519", symbol="现金流量表")
df
```