第 10 章 Spark SQL 实战—空气质量指数 PM2.5 分析

★本章导读★

本章通过对生活中常见的业务场景——空气质量指数 PM2.5 的分析，将上一章节所涉及的 DataFrame 相关处理方法进行练习。通过实际业务的开发更深层次理解 DataFrame 算子的适用场景和方法，能够加深对 DataFrame 结构化数据的理解。同时通过实战练习，能够初步培养面对业务需求，提炼处理问题思路的方法。

★知识要点★

通过本章内容的学习，读者将掌握以下知识：

- DataFrame 结构化数据的特点
- DataFrame 增删改查方法
- 能熟练适用 DataFrame 进行数据分析处理

10.1 Spark DataFrame 实战介绍

空气质量分析是日常生活中常常涉及的问题，空气质量的好坏对我们生活有着重大影响。细颗粒物又称细粒、细颗粒、PM2.5，对人体健康的危害大，因为直径越小，进入呼吸道的部位越深。本章节通过空气质量分析的实战，将上一章节关于 DataFrame 常用算子的应用进行巩固，加深对 DataFrame 数据分析处理方法的理解。

10.1.1 空气质量指数分析需求介绍

什么是 PM2.5？

PM2.5 是指大气中直径小于或等于 2.5 微米的颗粒物，也称为可入肺颗粒物。被吸入人体后会直接进入支气管，干扰肺部的气体交换，引发哮喘、支气管炎等方面的疾病。虽然 PM2.5 只是地球大气成分中含量很少的组成成分，但它对空气质量和能见度等有重要的影响。与较粗的大气颗粒物相比，PM2.5 粒径小，面积大，活性强，易附带有毒有害物质（例如，重金属、微生物等），且在大气中的停留时间长、输送距离远，因而对人体健康和大气环境质量的影响更大。PM10 是直径小于等于 10 微米的可吸入颗粒物，能够进入上呼吸道，但部分可通过痰液等排出体外，另外也会被鼻腔内部的绒毛阻挡，对人体健康危害相对较小。

那么，空气质量指数（AQI）与 PM2.5 有什么关系呢？

空气质量指数（AQI）的取值方法为：在 PM2.5，PM10 等六种主要污染物中，根据其浓度分别计算对应 AQI 值，然后取数值最大的那个作为最终报告的 AQI 值；PM2.5 指空气中直径≤2.5 微米的颗粒物，其数值一般来源于实测浓度值并计算得出。两者的取值途径完全不同，所以我们为了更严谨的呈现

出空气质量情况，选择将空气质量指数(AQI)和 PM2.5 分榜列出。

空气质量指数（AQI）的标准如表 10-1 所示。

表 10-1 空气质量指数（AQI）标准

空气质量指数（AQI）	PM2.5 健康建议
健康（0-50）	空气质量令人满意，基本无空气污染，各类人群可多参加户外活动，多呼吸一下清新的空气。
中等（51-100）	除少数对某些污染物特别容易过敏的人群外，其他人群可以正常进行室外活动。
对敏感人群不健康（101-150）	儿童、老年人及心脏病、呼吸系统疾病患者应尽量减少体力消耗大的户外活动。
不健康（151-200）	儿童、老年人及心脏病、呼吸系统疾病患者应尽量减少外出，停留在室内，一般人群应适量减少户外运动。
非常不健康（201-300）	儿童、老年人及心脏病、肺病患者应停留在室内，停止户外运动，一般人群尽量减少户外运动。
危险（301-500）	空气状况极差，儿童、老年人和病人应停留在室内，避免体力消耗，除有特殊需要的人群外，一般人群尽量不要停留在室外。

通过合理的方法检测和分析城市空气质量情况，对人们日常户外活动、工作提供有力依据，使人们拥有一个更加健康的生活环境。

10.1.2 需求分析及思路

本章节对北京空气质量指数 PM2.5 进行分析统计

需求思路如下：

1. 读取日志数据，查看数据集原始内容及数据结构；
2. 预处理原始数据集，并为原始数据集建立健康等级设置列；
3. 按照健康等级对数据进行不同维度的统计分析。
4. 绘制饼图，更直观的对北京空气质量情况进行分析

通过本章节实战，能帮助读者更好的掌握 DataFrame 算子的使用、统计分析的各个维度以及了解 Pyspark DataFrame 与 Pandas DataFrame 的转换方法。

10.2 空气质量指数 PM2.5 分析实战

10.2.1 数据集介绍

本章节实战使用的数据为北京 2017 年 1 月份至 6 月份的小时级别空气质量指数表，是数据分析处理、机器学习预测中常用的一份公开数据集。数据文件格式为 csv，数据文件为 Beijing2017_PM25.csv，数据内容如下图 10-1 所示。

Site	Parameter	Date (LST)	Year	Month	Day	Hour	Value	Unit	Duration	QC Name
Beijing	PM2.5	1/1/2017 0:00	2017	1	1	0	505	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 1:00	2017	1	1	1	485	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 2:00	2017	1	1	2	466	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 3:00	2017	1	1	3	435	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 4:00	2017	1	1	4	405	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 5:00	2017	1	1	5	402	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 6:00	2017	1	1	6	407	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 7:00	2017	1	1	7	435	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 8:00	2017	1	1	8	472	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 9:00	2017	1	1	9	465	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 10:00	2017	1	1	10	473	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 11:00	2017	1	1	11	456	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 12:00	2017	1	1	12	474	μg/m?1 Hr	Valid	Valid
Beijing	PM2.5	1/1/2017 13:00	2017	1	1	13	510	μg/m?1 Hr	Valid	Valid

图 10-1 北京 PM2.5 空气质量指数数据集

由上图可知，数据集包含信息有：Year、Month、Day、Hour、Value（AQI，空气质量指数）、QC Name（数据状态，invalid 或者 missing）等字段数据。

10.2.2 读取数据集

接下来，我们通过代码实战的方式进行读取并分析处理数据集。

步骤 1：启动 Pyspark Jupyter

虚拟机 Master 节点终端输入命令启动 Local 模式下的 Pyspark Notebook，后续章节均采用 Local 模式的 Pyspark Jupyter 进行实战，不再赘述，启动方式请参考第 6 章节内容。

在启动后，可以通过终端上打印的 Jupyter 地址在浏览器访问打开 Jupyter 页面。

步骤 2：新建 Notebook

通过右上角【New】下拉菜单选择【Python3】，并确定，完成 Notebook 的新建工作，如图 10-2 所示。

图 10-2 新建 Notebook 文件

步骤 3：重命名 Notebook

单击【Untitled】，输入“Chapter10”并确认，为 Notebook 重命名，如图 10-3 所示。

图 10-3 重命名 Notebook 文件

步骤 4：获取 Notebook 路径

在 Notebook 代码行中输入如下代码，导入本节需要使用到的库，并且获取 Notebook 所在路径并打印，如图 10-4 所示。

```
In [1]: # coding: utf-8
        from pyspark.sql.types import *
        import pyspark.sql.functions as F
```

导入相关依赖库

```
In [2]: pwd = !pwd
        data_path = "file://" + list(pwd)[0] + '/data/'
        print(data_path)

        file:///root/pyspark-book/data/
```

图 10-4 初始化代码

温馨提示：

```
from pyspark.sql.types import *    : Pyspark 字段类型库
import pyspark.sql.functions as F  : Pyspark 内部函数库
```

步骤 5：导入数据集

我们使用 sc.textFile 读取 Beijing2017_PM25.csv 数据文件，如图 10-5 所示。

```
In [5]: df = spark.read.format("csv") \
            .option("header", "true") \
            .option("inferSchema", "true") \
            .load(data_path + "Beijing_PM25.csv") \
            .select("Year", "Month", "Day", "Hour", "Value", "QC Name")
```

图 10-5 读取 PM2.5 数据集

读取 csv 文件代码命令实现详解如表 10-2 所示。

表 10-2 读取 csv 文件代码参数

代码	代码含义
spark.read.format("csv")	读取 csv 格式文件，
option(…)	读取数据/文件参数选项
"header", "true"	获取表头（即列名）
"inferSchema", "true"	指定字段类型
load(…)	读取文件路径
select(…)	筛选指定字段

步骤 6：查看数据内容

我们通过 show()算子查看读取的数据集前 10 行数据内容，如图 10-6 所示。

```
In [6]: df.show(10)
```

```
+----+-----+---+----+-----+--------+
|Year|Month|Day|Hour|Value|QC Name|
+----+-----+---+----+-----+--------+
|2017|    1|  1|   0|  505|  Valid|
|2017|    1|  1|   1|  485|  Valid|
|2017|    1|  1|   2|  466|  Valid|
|2017|    1|  1|   3|  435|  Valid|
|2017|    1|  1|   4|  405|  Valid|
|2017|    1|  1|   5|  402|  Valid|
|2017|    1|  1|   6|  407|  Valid|
|2017|    1|  1|   7|  435|  Valid|
|2017|    1|  1|   8|  472|  Valid|
|2017|    1|  1|   9|  465|  Valid|
+----+-----+---+----+-----+--------+
only showing top 10 rows
```

图 10-6 查看数据集前 10 行

步骤 7：查看数据集字段类型

我们通过 printSchema()算子查看读取的数据集各个字段类型，如图 10-7 所示。

```
In [7]: df.printSchema()
```

```
root
 |-- Year: integer (nullable = true)
 |-- Month: integer (nullable = true)
 |-- Day: integer (nullable = true)
 |-- Hour: integer (nullable = true)
 |-- Value: integer (nullable = true)
 |-- QC Name: string (nullable = true)
```

图 10-7 查看数据集字段类型

从以上结果可以看到，除“QC Name”字段为 string 类型外，其余均为 integer 类型，若读取代码不指定参数 option("inferSchema", "true")，则读取后字段各类型均为 string 类型，读者可自行尝试。

10.2.3 数据预处理

在上一小节中，我们通过代码读取本地数据集，本小节将对原始数据集进行预处理，例如创建健康等级函数及相应 UDF 函数等。

步骤 1：创建健康级别函数

在 Notebook 代码行中创建 PM2.5 对应健康级别设置函数，如图 10-8 所示。

```
In [3]: def get_health_level(value):
            """
            PM2.5对应健康级别设置
            :param value: PM2.5
            :return: level
            """
            if 0 <= value <= 50:
                return "Very Good"
            elif 50 < value <= 100:
                return "Good"
            elif 100 < value <= 150:
                return "Unhealthy for Sensi"
            elif value <= 200:
                return "Unhealthy"
            elif 200 < value <= 300:
                return "Very Unhealthy"
            elif 300 < value <= 500:
                return "Hazardous"
            elif value > 500:
                return "Extreme danger"
            else:
                return None
```

图 10-8 PM2.5 对应健康级别函数

步骤 2：创建 UDF 函数

在步骤 1 中我们创建了健康级别设置的函数，接下来需要创建对应的 UDF 函数，从而我们可以应用到 DataFrame 数据中，创建 UDF 代码，如图 10-9 所示。

```
In [9]: health_level_udf = F.udf(get_health_level, StringType())
```

图 10-9 创建 health_level_udf 函数

代码使用的了 Spark 字典的函数 udf(…)来创建 UDF 函数，其中第一个参数为自定义函数，第二个参数为 UDF 函数返回值类型。

步骤 3：创建 health_level 列

通过步骤 1 和步骤 2 已经完成了 UDF 函数的创建工作，接下来需要将 UDF 应用到 DataFrame 数据中，为其创建 health_level 列，代码实现如图 10-10 所示。

```
In [10]: df = df.withColumn("healthy_level", health_level_udf("Value"))
In [11]: df.show(10)
```

1. 增加 health_level 列

```
+----+-----+---+----+-----+--------+--------------+
|Year|Month|Day|Hour|Value|QC Name| healthy_level|
+----+-----+---+----+-----+--------+--------------+
|2017|    1|  1|   0|  505|  Valid|Extreme danger|
|2017|    1|  1|   1|  485|  Valid|     Hazardous|
|2017|    1|  1|   2|  466|  Valid|     Hazardous|
|2017|    1|  1|   3|  435|  Valid|     Hazardous|
|2017|    1|  1|   4|  405|  Valid|     Hazardous|
|2017|    1|  1|   5|  402|  Valid|     Hazardous|
|2017|    1|  1|   6|  407|  Valid|     Hazardous|
|2017|    1|  1|   7|  435|  Valid|     Hazardous|
|2017|    1|  1|   8|  472|  Valid|     Hazardous|
|2017|    1|  1|   9|  465|  Valid|     Hazardous|
+----+-----+---+----+-----+--------+--------------+
only showing top 10 rows
```

2. 打印新数据

图 10-10 添加 healthy_level 列

由上图可知，我们使用 withColumn 方法为原始数据集添加了名为“healthy_level”的一列。

10.2.4 空气质量统计与分析

在上一小节中，我们对原始数据集进行了读取和预处理，为原始数据建立了健康级别等级列。在本小节中，将通过结合 DataFrame 函数方法与数学统计方法实现对空气质量的计算分析。

步骤 1：根据健康等级分组统计

使用 groupBy 算子根据健康级别情况进行分组统计，代码实现如图 10-11 所示。

```
In [13]: group_df = df.groupBy("healthy_level").count()

In [14]: group_df.show(10)
+--------------------+-----+
|      healthy_level|count|
+--------------------+-----+
|           Unhealthy|  198|
|      Extreme danger|   27|
|                Good| 1021|
|           Very Good| 2438|
|Unhealthy for Sensi|  374|
|           Hazardous|  107|
|      Very Unhealthy|  179|
+--------------------+-----+
```

图 10-11 健康等级分组统计

由上图统计可知，对于本数据集，空气质量等级为 Very Good 的小时数有 2438，空气质量等级为 Good 的小时数有 1024；而空气质量等级为 Unhealthy 的小时数有 198，空气质量等级为 Very Unhealthy 的小时数有 179。由以上统计数据可以侧面反映。现如今空气质量良好的占有非常大的比重。

步骤 2：占比统计

在步骤 1 中我们通过数据展示可以直观的看到空气质量等级较好的小时数占有很大比例，那么具体各个等级空气质量占比值有多少天呢？我们可以通过进一步的处理进行分析，首先我们使用 count(0 方法统计数据总量；然后，通过 winColumn 方法新增两列数据，分别是累积天数 days 列和各等级天数所占比例 percentage 列，具体实现代码如图 10-12 所示。

```
In [16]: sums = df.count()
         print(sums)
```

1. 统计数据总数

```
4344

In [17]: group_df.printSchema()
root
 |— healthy_level: string (nullable = true)
 |— count: long (nullable = false)

In [18]: result_df = group_df\
             .withColumn("days", (F.col("count") / 24).cast(IntegerType())) \
             .withColumn("percentage", F.round(F.col("count") / sums, 3))
```

2. 新增列，计算各等级累积天数

3. 新增列，统计各等级占比

图 10-12 空气质量等级占比

> 温馨提示：
>
> 在 In[18] 代码中第 1 行和第二行末尾使用的了"\"，这是 Python 的代码规则，即一行代码过长需要换行，则要通过"\"进行连接。

步骤 3：占比结果展示

批注 [雷1]: 图片内容步骤需在正文中描述出来

步骤 2 中统计结果如图 10-13 所示。

```
In [19]: result_df.show(10)
```

```
+--------------------+-----+----+----------+
|      healthy_level|count|days|percentage|
+--------------------+-----+----+----------+
|           Unhealthy|  198|   8|     0.046|
|      Extreme danger|   27|   1|     0.006|
|                Good| 1021|  42|     0.235|
|           Very Good| 2438| 101|     0.561|
|Unhealthy for Sensi|  374|  15|     0.086|
|           Hazardous|  107|   4|     0.025|
|      Very Unhealthy|  179|   7|     0.041|
+--------------------+-----+----+----------+
```

图 10-13 占比统计结果展示

通过打印结果我们可以更直观的看到各个空气质量等级累计天数情况以及所占比例。

步骤 4：月度 AQI 均值统计

对每个月 AQI 均值情况进行统计，代码实现如图 10-14 所示。

```
In [27]: month_avg_df = df.groupBy("month").agg({"Value": "mean"})
         month_avg_df.show()
```

```
+-----+------------------+
|month|        avg(Value)|
+-----+------------------+
|    1|108.73118279569893|
|    6|32.918055555555554|
|    3| 60.81720430107527|
|    5| 57.62096774193548|
|    4| 45.46944444444444|
|    2| 72.95982142857143|
+-----+------------------+
```

图 10-14 月度 AQI 均值统计

步骤 5：月度健康等级状况统计

在前面章节讲到，groupBy 算子可以传入多列，实现按多列的分组统计，我们可以对月度健康等级状况进行统计，代码实现如图 10-15 所示。

```
In [28]: month_df = df.groupBy("month", "healthy_level").count()
         month_df.show()
```

```
+-----+--------------------+-----+
|month|      healthy_level|count|
+-----+--------------------+-----+
|    5|           Unhealthy|    4|
|    1|      Very Unhealthy|   94|
|    5|                Good|  212|
|    3|      Very Unhealthy|   16|
|    2|                Good|  110|
|    2|           Unhealthy|   45|
|    6|           Very Good|  525|
|    1|           Very Good|  319|
|    4|Unhealthy for Sensi|   71|
|    6|Unhealthy for Sensi|    7|
|    4|                Good|  225|
|    6|           Unhealthy|    8|
|    4|           Unhealthy|   29|
|    4|           Very Good|  391|
|    6|                Good|  180|
|    2|      Very Unhealthy|   63|
|    1|           Hazardous|   71|
|    1|           Unhealthy|   54|
|    1|                Good|  109|
|    3|                Good|  185|
+-----+--------------------+-----+
only showing top 20 rows
```

图 10-15 月度健康等级状况统计

步骤 6：结果数据存储

最后一步，将对我们统计分析的数据结果进行本地存储，我们同样将结果存储为 csv 格式，这里传入了两个参数，path 为 csv 数据文件存储路径和 mode 为数据文件存储的方式，这里采用了“overwrite”模式，即覆盖的方式，存储代码如图 10-16 所示。

```
In [20]: result_df.repartition(1).write.csv(path=data_path+'result_PM2.5', mode="overwrite")

In [22]: %cat ./data/result_PM2.5/part-00000-86dde251-a54c-4eda-aa35-81ae26418afc-c000.csv
```

```
Unhealthy,198,8,0.046
Extreme danger,27,1,0.006
Good,1021,42,0.235
Very Good,2438,101,0.561
Unhealthy for Sensi,374,15,0.086
Hazardous,107,4,0.025
Very Unhealthy,179,7,0.041
```

图 10-16 结果存储 csv 文件

10.3 Pyspark 与 Pandas 交互

在上一小节中，我们通过代码实现了关于空气质量指数的相关统计，并对结果进行了展示和分析。在本节中将对 Pyspark DataFrame 与 Pandas DataFrame 的交互进行实战。

10.3.1 Pyspark 与 Pandas 的转换

Pyspark DataFrame 可以通过代码转换成 Pandas DataFrame，这样可以方便使用 Python 自带的一些库进行数据统计分析，例如画图。转换实现的代码如图 10-17 所示。

```
In [29]: pd_df = result_df.toPandas()

In [30]: pd_df.head()
```

Out[30]:

	healthy_level	count	days	percentage
0	Unhealthy	198	8	0.046
1	Extreme danger	27	1	0.006
2	Good	1021	42	0.235
3	Very Good	2438	101	0.561
4	Unhealthy for Sensi	374	15	0.086

图 10-17 Pyspark 与 Pandas 的转换

10.3.2 Pandas 数据绘图

使用 Pandas 可以进行图形绘制，更方便、直观的帮助我们去理解数据处理的结果，Python 使用 matplotlib 可以实现图形绘制，本小节将对上一小节的结果绘制饼图，展示各个空气质量健康等级所占百分比，具体关于使用 Python matplotlib 库绘图的方法本书不再展开赘述，实现代码如图 10-18 所示。

```
In [65]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt

In [66]: plt.title("Beijing AQI PM2.5")
         labels = pd_df['healthy_level']
         x = pd_df["percentage"]
         plt.pie(x, labels=labels,autopct='%1.1f%%')
         plt.axis("equal")
         plt.show()
```

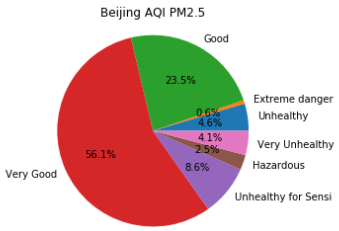

图 10-18 空气质量等级百分比饼图

★回顾思考★

01　PySpark 读取 csv 文件通过什么 API 指定读取参数？

答：

```
.option(key , value)
```

其中，key 指要设置的参数名，value 指要设置的参数名对应的参数值。

02　Pspark 自带函数库和数据类型库是什么？

答：

```
pyspark.sql.types　：Pyspark 字段类型库
pyspark.sql.functions ：Pyspark 内部函数库
```

★练习★

一、选择题

1. Spark DataFrame 增加一列使用的函数是（ ）
   A. with
   B. withColumnRename
   C. withCols
   D. withColumn
2. Spark 读取 csv 文件使用 csv 列名的参数代码（ ）
   A. .option("header", true)
   B. .option("header", "true")
   C. .option("header", 1)
   D. .option("header", True)
3. Pyspark DataFrame 使用哪个算子可以打印数据结构（ ）
   A. printSchema()
   B. print()
   C. getSchema()
   D. show()

二、判断题

1. Pyspark DataFrame 和 Pandas DataFrame 可以相互转换。（√）
2. Pyspark DataFrame 分组统计 groupBy 算子可以按多列分组。（√）

三、实战练习

任务 1：手动实现空气质量指数 PM2.5 分析的项目实战。

柱状图提示：

```
plt.title("Beijing AQI PM2.5")
labels = pd_df['healthy_level']
x = pd_df["percentage"]
```

```
plt. bar (x, labels=labels,autopct='%1.1f%%')
plt.axis("equal")
plt.show()
```

本章小结

本章详细介绍了日常生活中常见的空气质量指数统计分析的场景，更加体现了大数据分析处理的重要性和广泛应用。我们通过对空气质量指数的分析统计对 Pyspark DataFrame 的使用方法有了更进一步的巩固和掌握，同时简单介绍了 Pyspark DataFrame 和 Pandas DataFrame 之间的转换和处理，使读者能够更灵活的进行大数据分析与处理。