

## 实验五： Spark SQL 编程实践

### 实验基本信息：

时间： ： 2024 年 9 月 30 日

实验类型： ☐验证性 ☐设计性 ☒综合性

班级： 计算机科学与技术专业（中外合作）4 班 班级代码：

理论老师： 邱开金 实验指导： 邱开金

### 实验报告提交说明：

本次实验需要撰写实验报告，实验报告填写时以完成实验任务为目的，先简要回答完成实验任务需要的步骤或需要执行的命令代码，再配以结果截图予以说明，图片数量不宜过多，能说明问题即可。最后配上总结，并提交到教师指定位置。

### 实验目的：

1. 了解 SparkSession 创建 DataFrame。
2. 了解从 RDD 转化为 DataFrame 的方法。
3. 熟练使用 DataFrame 提供的算子完成具体任务。
4. 熟练使用 spark.sql 执行 SQL 语句，完成具体任务。

### 实验任务：

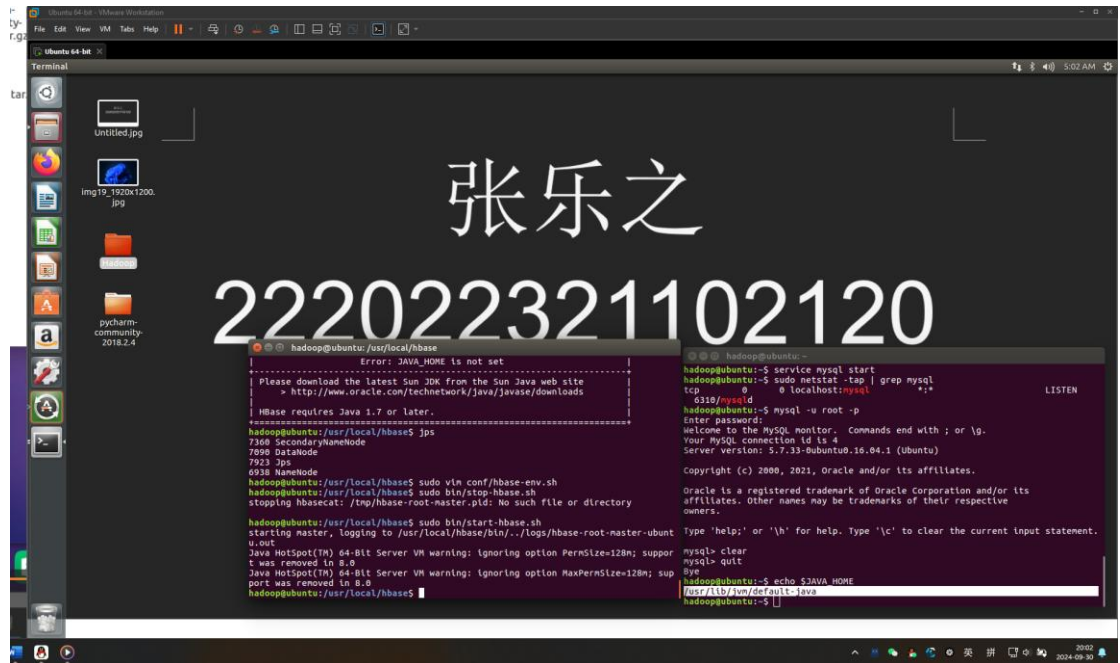
1. 安装 mysql 和 HBase. 把安装的截图放上来。

MySQL:



```
hadoop@ubuntu:~$ sudo apt-get update
[sudo] password for hadoop:
Hit:1 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease
Reading package lists... Done
hadoop@ubuntu:~$ sudo apt-get install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libatoi libhtml-template-perl mysql-client-5.7 mysql-client-core-5.7
  mysql-common mysql-server-5.7 mysql-server-core-5.7
Suggested packages:
  libipc-sharedcache-perl mailx tinyca
The following NEW packages will be installed:
  libatoi libhtml-template-perl mysql-client-5.7 mysql-client-core-5.7
  mysql-common mysql-server mysql-server-5.7 mysql-server-core-5.7
0 upgraded, 8 newly installed, 0 to remove and 47 not upgraded.
Need to get 17.4 MB of archives.
After this operation, 156 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 mysql-comm
```





2. 读取文件 people.json, 转换为 DataFrame, 完成以下实验任务

1. 文件位于: /usr/local/spark/examples/src/main/resources/people.json

1) 统计文件中记录总数。

2) 查询年龄小于 20 岁的记录。

3) 根据当前年份及年龄计算出出生日期。

```
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder.appName("People Analysis").getOrCreate()
```

```
df = spark.read.json("file:///usr/local/spark/examples/src/main/resources/people.json")
```

```
df.createOrReplaceTempView("people")
```

```
total_count = df.count()
```

```
print("Total record count: {}".format(total_count))
```

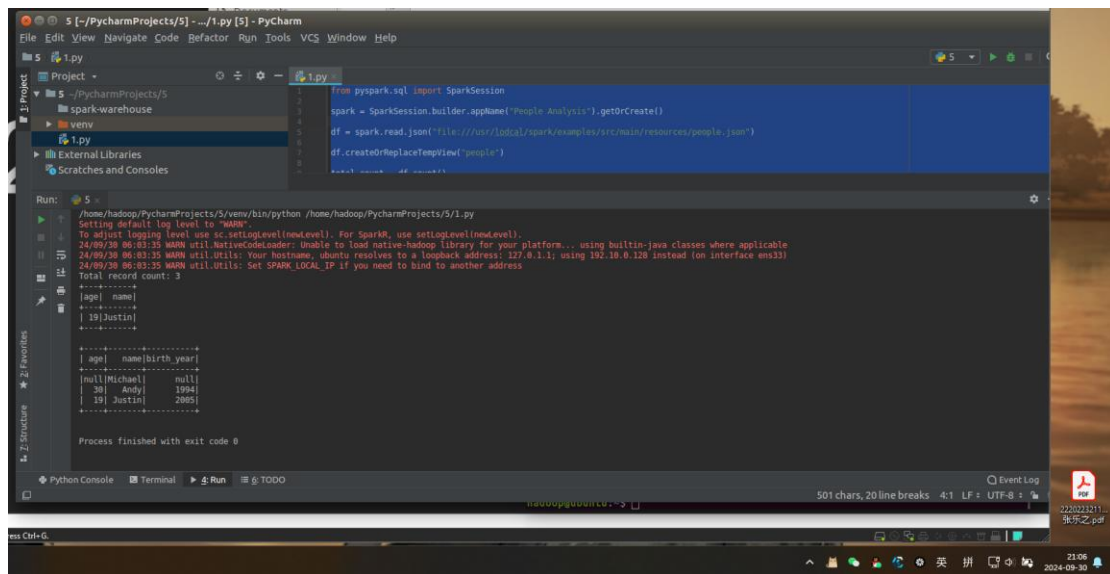
```
under_20 = df.filter(df.age < 20)
```

```
under_20.show()
```

```
from pyspark.sql.functions import expr
```

```
df_with_birthdate = df.withColumn("birth_year", expr("2024 - age"))
```

```
df_with_birthdate.show()
```

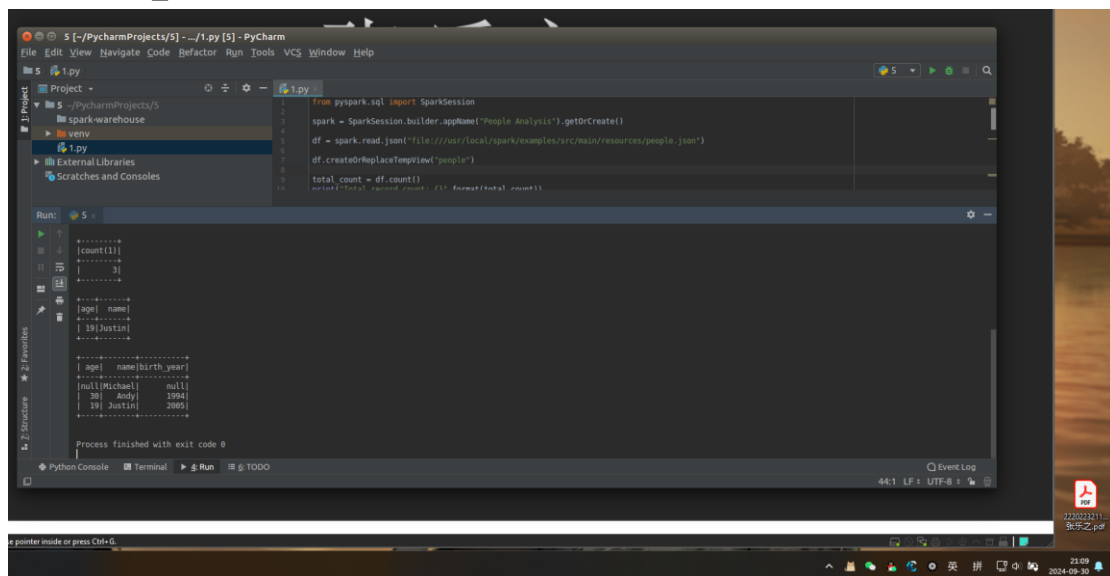


4) 利用 SQL 语句重新完成 1) -3) 的任务。

```
total_count_sql = spark.sql("SELECT COUNT(*) FROM people")
total_count_sql.show()
```

```
under_20_sql = spark.sql("SELECT * FROM people WHERE age < 20")
under_20_sql.show()
```

```
birthdate_sql = spark.sql("SELECT *, (2024 - age) AS birth_year FROM people")
birthdate_sql.show()
```



2. 读取 iris.txt 并转换成 DataFrame, 完成计算任务

1) iris.txt 与实验四内容一致。读入 iris.txt 到 RDD 中, 显示文件行数。

```
from pyspark.sql import SparkSession
```

```
from pyspark.sql import Row
```

```
spark = SparkSession.builder.appName("Iris Analysis").getOrCreate()
```

```
rdd = spark.sparkContext.textFile("file:///home/hadoop/Desktop/Hadoop/iris.txt")
```

```
line_count = rdd.count()
```

```
print("Total number of lines: {}".format(line_count))
```

2) 利用任意一个种方法将 RDD 转换为 DataFrame，其中涉及的模式为 SepalLength, SepalWidth, PetalLength, PetalWidth, TrainingClass。

```
columns = ["SepalLength", "SepalWidth", "PetalLength", "PetalWidth", "TrainingClass"]
```

```
row_rdd = rdd.map(lambda line: line.split(",")).map(lambda p: Row(SepalLength=float(p[0]),
```

```
SepalWidth=float(p[1]),
```

```
PetalLength=float(p[2]),
```

```
PetalWidth=float(p[3]),
```

```
TrainingClass=p[4]))
```

```
iris_df = spark.createDataFrame(row_rdd, schema=columns)
```

```
iris_df.show()
```

3) 转换得到 DataFrame 后，利用 xx.createOrReplaceTempView("iris\_view") 代码创建 iris\_view。

```
iris_df.createOrReplaceTempView("iris_view")
```

4) 利用 spark.sql 执行 SQL 语句完成如下任务：

- a. 统计总行数；
- b. 统计每种花的数量；
- c. 统计 PetalWidth 小于 2 的记录有哪些。
- d. 计算 Iris-virginica 花的平均 PetalLength 值，并输出。

```
total_count_sql = spark.sql("SELECT COUNT(*) FROM iris_view")
```

```
total_count_sql.show()
```

```
flower_count_sql = spark.sql("SELECT TrainingClass, COUNT(*) AS count FROM iris_view GROUP BY TrainingClass")
```

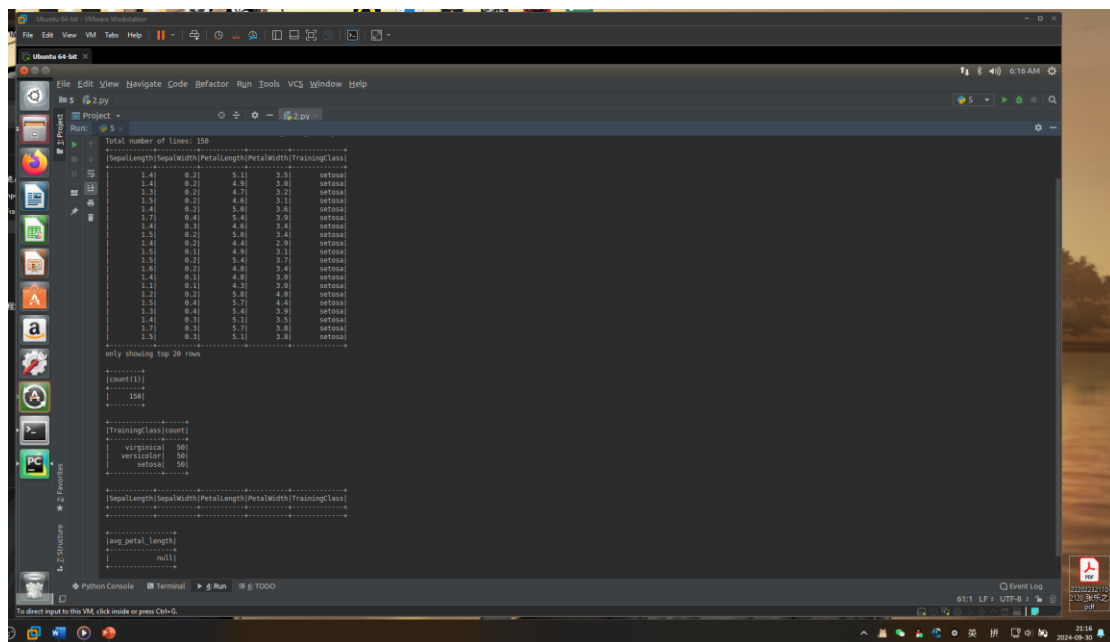
```
flower_count_sql.show()
```

```
petalwidth_filter_sql = spark.sql("SELECT * FROM iris_view WHERE PetalWidth < 2")
```

```
petalwidth_filter_sql.show()
```

```
avg_petal_length_sql = spark.sql("SELECT AVG(PetalLength) AS avg_petal_length FROM iris_view WHERE TrainingClass = 'Iris-virginica'")
```

```
avg_petal_length_sql.show()
```



```
Total number of Lines: 150
[SepalLength|SepalWidth|PetalLength|PetalWidth|TrainingClass]
1.4| 0.2| 5.1| 1.5| setosa
1.4| 0.2| 4.9| 1.6| setosa
1.3| 0.2| 4.7| 1.2| setosa
1.5| 0.2| 4.6| 1.1| setosa
1.4| 0.2| 5.0| 1.6| setosa
1.7| 0.4| 5.4| 1.9| setosa
1.4| 0.3| 4.8| 1.4| setosa
1.5| 0.2| 5.0| 1.4| setosa
1.4| 0.2| 4.4| 1.9| setosa
1.5| 0.2| 4.9| 1.3| setosa
1.5| 0.2| 5.4| 1.7| setosa
1.4| 0.2| 4.8| 1.4| setosa
1.4| 0.1| 4.9| 1.6| setosa
1.1| 0.1| 4.3| 1.0| setosa
1.2| 0.2| 5.0| 1.6| setosa
1.5| 0.4| 5.7| 1.4| setosa
1.3| 0.4| 5.4| 1.9| setosa
1.4| 0.3| 5.1| 1.5| setosa
1.7| 0.3| 5.7| 1.6| setosa
1.5| 0.3| 5.1| 1.8| setosa

only showing top 20 rows
count()
150

TrainingClass|count()
virginica| 50
versicolor| 50
setosa| 50

[SepalLength|SepalWidth|PetalLength|PetalWidth|TrainingClass]
avg_petal_length
null
```

## 实验总结:

在本次实验中，我主要学习了 **SparkSQL** 技术来处理并分析数据。基于 **Mysql** 和 **hbase** 平台，并使用了 **RDD** 和 **DataFrame** 数据结构，我们编写了 **python** 程序来处理问题。首先将文本文件读取为 **RDD**，(**Spark** 中用于并行处理数据的底层结构)。然后通过 **map** 操作将 **RDD** 转换为 **Row** 格式，再使用 **createDataFrame** 将其转为 **DataFrame**，这样我们可以像操作数据库表一样对数据进行结构化查询。接着通过 **createOrReplaceTempView** 创建 **SQL** 视图，并利用 **Spark SQL** 进行总行数统计、分组统计、条件过滤等操作。**Spark** 的分布式计算模型和内存优化使其非常高效，适合大规模数据处理。