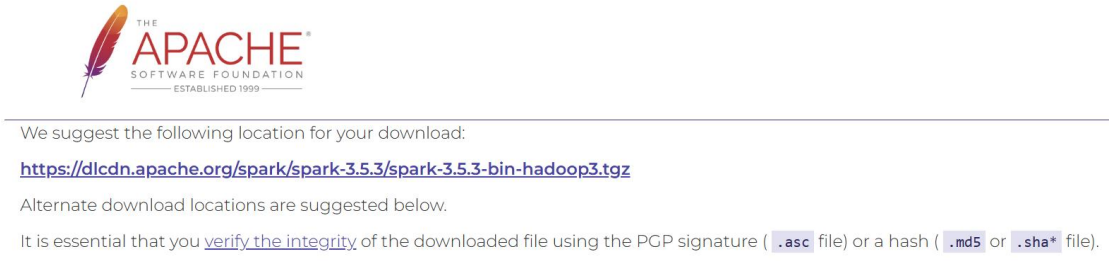
实验 4 实验报告

任务 0：下载并配置 spark 环境

1.下载与解压

下载 spark-3.5.3-bin-hadoop3.tgz 并移到虚拟机中。



使用命令“**tar -xvzf spark-3.5.3-bin-hadoop3.tgz**”解压。

```
njucs@njucs-VirtualBox:~/Desktop$ ls
apache-hive-3.1.3-bin.tar.gz  spark-3.5.3-bin-hadoop3.tgz  work
Purchase_Redemption_Data      test2                        work5
njucs@njucs-VirtualBox:~/Desktop$ tar -xvzf spark-3.5.3-bin-hadoop3.tgz
spark-3.5.3-bin-hadoop3/
spark-3.5.3-bin-hadoop3/data/
spark-3.5.3-bin-hadoop3/data/graphx/
spark-3.5.3-bin-hadoop3/data/graphx/users.txt
spark-3.5.3-bin-hadoop3/data/graphx/followers.txt
spark-3.5.3-bin-hadoop3/data/mllib/
spark-3.5.3-bin-hadoop3/data/mllib/sample_linear_regression_data.txt
spark-3.5.3-bin-hadoop3/data/mllib/sample_fpgrowth.txt
spark-3.5.3-bin-hadoop3/data/mllib/sample_libsvm_data.txt
spark-3.5.3-bin-hadoop3/data/mllib/gmm_data.txt
spark-3.5.3-bin-hadoop3/data/mllib/kmeans_data.txt
spark-3.5.3-bin-hadoop3/data/mllib/streaming_kmeans_data_test.txt
spark-3.5.3-bin-hadoop3/data/mllib/sample_lda_data.txt
```

使用命令”**sudo mv spark-3.5.3-bin-hadoop3 /home/njucs**” 将文件移位。

```
njucs@njucs-VirtualBox:~/Desktop$ sudo mv spark-3.5.3-bin-hadoop3 /home/njucs
[sudo] password for njucs:
njucs@njucs-VirtualBox:~/Desktop$ cd ..
njucs@njucs-VirtualBox:~$ ls
derby.log  Downloads     hive-3.1.3  Pictures                switchyard
Desktop    hadoop         input       Public                  Templates
Documents  hadoop-3.4.0   Music       spark-3.5.3-bin-hadoop3  Videos
njucs@njucs-VirtualBox:~$ mv spark-3.5.3-bin-hadoop3 spark-3.5.3
njucs@njucs-VirtualBox:~$ ls
derby.log  Downloads     hive-3.1.3  Pictures     switchyard
Desktop    hadoop         input       Public       Templates
Documents  hadoop-3.4.0   Music       spark-3.5.3  Videos
```

使用命令”**mv spark-3.5.3-bin-hadoop3 spark-3.5.3**” 将文件移位。

2. 配置文件

创建本地文件目录：“**mkdir /home/njucs/spark-3.5.3/store/logDirectory**”
“**mkdir /home/njucs/spark-3.5.3/store/eventLogdir**”

```
njucs@njucs-VirtualBox:~/spark-3.5.3$ mkdir /home/njucs/spark-3.5.3/store
njucs@njucs-VirtualBox:~/spark-3.5.3$ ls
bin   data      jars        LICENSE   NOTICE  R           RELEASE  store
conf  examples  kubernetes  licenses  python  README.md   sbin     yarn
njucs@njucs-VirtualBox:~/spark-3.5.3$ cd store/
njucs@njucs-VirtualBox:~/spark-3.5.3/store$ mkdir /home/njucs/spark-3.5.3/store/
logDirectory
njucs@njucs-VirtualBox:~/spark-3.5.3/store$ mkdir /home/njucs/spark-3.5.3/store/
eventLogdir
njucs@njucs-VirtualBox:~/spark-3.5.3/store$ ls
eventLogdir  logDirectory
njucs@njucs-VirtualBox:~/spark-3.5.3/store$
```

使用“**cp spark-defaults.conf.template spark-defaults.conf**”
“**cp spark-env.sh.template spark-env.sh**”复制副本

```
njucs@njucs-VirtualBox:~/spark-3.5.3/conf$ ls
fairscheduler.xml.template      spark-defaults.conf.template
log4j2.properties.template      spark-env.sh
metrics.properties.template     spark-env.sh.template
spark-defaults.conf             workers.template
```

设置 Spark 环境变量，为本地文件目录

```
export SPARK_HISTORY_OPTS="-Dspark.history.ui.port=18080 -Dspark.history.fs.logD
irectory= </home/njucs/spark-3.5.3/store/logDirectory>"
#!/usr/bin/env bash

#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
```

配置 spark-defaults.conf，将 spark.eventLog.enabled 设置为 true；
spark.eventLog.dir 设置为本地文件目录。

```
# Default system properties included when running spark-submit.
# This is useful for setting default environmental settings.

# Example:
# spark.master                     spark://master:7077
 spark.eventLog.enabled           true
 spark.eventLog.dir               /home/njucs/spark-3.5.3/store/eventLogdir
# spark.serializer                 org.apache.spark.serializer.KryoSerializer
# spark.driver.memory              5g
# spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="on
e two three"
```

3.启动 spark

使用“./start-all.sh”启动节点，jps 后显示节点正常运行

```
njucs@njucs-VirtualBox:~/spark-3.5.3/sbin$ ./start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /home/njucs/spark-3.5
.3/logs/spark-njucs-org.apache.spark.deploy.master.Master-1-njucs-VirtualBox.out
localhost: starting org.apache.spark.deploy.worker.Worker, logging to /home/njuc
s/spark-3.5.3/logs/spark-njucs-org.apache.spark.deploy.worker.Worker-1-njucs-Vir
tualBox.out
njucs@njucs-VirtualBox:~/spark-3.5.3/sbin$ jps
4150 Jps
3918 Master
4062 Worker
```

启动 shell，成功启动，说明配置成功（但是接下来会在 vscode 中编程，貌似不需要 shell）

```
njucs@njucs-VirtualBox:~/spark-3.5.3/bin$ spark-shell
24/12/13 21:10:40 WARN Utils: Your hostname, njucs-VirtualBox resolves to a loop
back address: 127.0.1.1; using 192.168.208.146 instead (on interface ens33)
24/12/13 21:10:40 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
24/12/13 21:10:51 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Spark context Web UI available at http://192.168.208.146:4040
Spark context available as 'sc' (master = local[*], app id = local-1734095453863
).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.3
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.19)
Type in expressions to have them evaluated.
```

任务 1：Spark RDD 编程

1. 查询特定日期的资金流入和流出情况

使用 user_balance_table ，计算出所有用户在每一天的总资金流入和总资金流出量。

格式：<日期> <资金流入量><资金流出量>

思路：

遍历每一行数据，按日期进行计算资金流入和流出，即将 report_date 相同的行中的 total_purchase_amt, total_redeem_amt 各自相加，然后以(日期, (资金流入，资金流出))的形式，储存在 RDD 中，最终输出。

核心代码：

处理数据：

定义 parse_line 函数，按照逗号区分不同元素的数据，然后判定第一个字符串为“user_id”的行为表头行，将其跳过，然后查询本次实验所需要的列,最终以（日期，资金流入，资金流出)的元组形式输出数据

```
def parse_line(line):
    # 按照逗号分割字段
    fields = line.split(",")
    # 跳过表头行
    if fields[0] == "user_id":
        return None

    report_date = fields[1]  # 日期
    total_purchase_amt = int(fields[4])  # 购买总金额
    total_redeem_amt = int(fields[8])  # 赎回总金额

    return (report_date, total_purchase_amt, total_redeem_amt)
```

计算每天的总资金流入和总资金流出量：

data 即本次的原始数据文件 user_balance_table.csv

map(parse_line)：map 是 spark 的转换操作，将数据集中的每一行数据输入到 parse_line 函数中。

filter(lambda x: x is not None)：filter 操作会去除 parse_line 返回值为 None 的行，保留有效的数据，从而增加程序的鲁棒性。

parsed_data.map(lambda x: (x[0], (x[1], x[2]))）：map 将每一行的数据转换为(日期, (资金流入，资金流出)) 形式的元组，可以理解为将日期设置为键，将(资金流入，资金流出)设置为值，方便接下来根据日期来计算总值。

reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1]))：reduceByKey 是 Spark 中的一个聚合操作，它根据键（日期）对值进行聚合，将每个日期所对应的（资金流入，资金流出）元组进行累加求和。

lambda a, b: (a[0] + b[0], a[1] + b[1]) ：a[0]和 b[0]是日期相同的两行数据中的流入金额，a[1]和 b[1]是流出金额，此处表示对于每个日期，累加所有的资金流入和资金流出。

```
# 解析每一行数据
parsed_data = data.map(parse_line).filter(lambda x: x is not None)
# 按日期进行计算资金流入和流出
aggregated_data = parsed_data.map(lambda x: (x[0], (x[1], x[2]))).\
    reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1]))
# 显示结果
result = aggregated_data.collect()
for record in result:
    print(f"{record[0]} {record[1][0]} {record[1][1]}")
```

运行程序:

使用“./bin/spark-submit --master local[*] /home/njucs/Desktop/test4/work1/step1.py”提交 step1 的代码，其中 master local[*]是本地运行

```
njucs@njucs-VirtualBox:~/spark-3.5.3$ ./bin/spark-submit --master local[*] /home
/njucs/Desktop/test4/work1/step1.py
24/12/16 14:43:43 WARN Utils: Your hostname, njucs-VirtualBox resolves to a loop
back address: 127.0.1.1; using 192.168.208.146 instead (on interface ens33)
24/12/16 14:43:43 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
24/12/16 14:43:45 INFO SparkContext: Running Spark version 3.5.3
24/12/16 14:43:45 INFO SparkContext: OS info Linux, 5.0.0-23-generic, amd64
24/12/16 14:43:45 INFO SparkContext: Java version 11.0.19
24/12/16 14:43:45 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
24/12/16 14:43:45 INFO ResourceUtils: ==========================================
====================
24/12/16 14:43:45 INFO ResourceUtils: No custom resources configured for spark.d
river.
24/12/16 14:43:45 INFO ResourceUtils: ==========================================
====================
24/12/16 14:43:45 INFO SparkContext: Submitted application: Calculate Flow
24/12/16 14:43:45 INFO ResourceProfile: Default ResourceProfile created, executo
r resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory ->
name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount
: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
```

展示部分输出，其余输出保存在/home/njucs/Desktop/test4/work1/output1 中

```
24/12/16 14:44:03 INFO DAGScheduler: Job 0 finished: collect at /home/njucs/Desk
top/test4/work1/step1.py:28, took 13.315661 s
20140823 141412027 199377531
20140827 302194801 468164147
20140812 258493673 309754858
20140731 191728916 277194379
20140716 394890140 234775948
20140412 177642053 123295320
20140325 314345006 312710515
20140810 259534870 189909225
20140729 228093046 303480103
20140702 384555819 328950951
20140405 202336542 163199682
20140411 237829882 277077434
20140504 303087562 413222034
20140510 287240171 147248328
20140519 259077930 293791406
20140320 365011495 336076380
20140407 196936223 176966561
20140428 324937272 327724735
```

遇到的问题：

1. 一开始输出格式为('20140514', (305474522, 316225442)) ：

因为这是在 print()中元组的默认格式，如果不要输出标点符号，将数据使用空格分隔，要加入“map(lambda x: f"{x[0]} {x[1][0]} {x[1][1]}")”来规范其输出格式

2. 保存输出文件后发现输出被分到了四个文件里储存:

这是因为 Spark 是分布式计算框架，saveAsTextFile()默认会将结果保存为多个部分文件，每个分区的数据会保存到不同的文件中，要使用 .coalesce(1) 来将所有的结果合并到一个分区，从而输出到一个文件中

2. 活跃用户分析

使用 user_balance_table ，定义活跃用户为在指定月份内有至少 5 天记录的用户，统计 2014 年 8月的活跃用户总数。

格式：<活跃用户总数>

思路：

遍历每一行数据，先筛选出所有日期为 2014 年 8月的数据，然后统计每个用户在 2014 年 8 月的不同日期数量，筛选出至少有 5 天记录的活跃用户，再获取活跃用户的总数。

核心代码：

处理数据：

本次只用获取用户 id 和日期这两列数据，但是要根据日期对数据进行筛选，仅返回 report_date 为“201408”开头的数据，其余返回 none。

此处就体现了上个任务中 filter(lambda x: x is not None)的重要性。

```
def parse_line(line):
    fields = line.split(",")

    if fields[0] == "user_id":
        return None

    user_id = fields[0]  # 用户ID
    report_date = fields[1]  # 日期

    # 如果日期是2014年8月，返回元组 (user_id, report_date)
    if report_date.startswith("201408"):
        return (user_id, report_date)
    else:
        return None
```

计算活跃用户总量：

distinct()：用于移除重复的（用户 id,日期），确保每个用户在每个日期只被计算一次。

groupByKey()：按用户 ID 对数据进行分组，生成一个新的 RDD，键是用户 id，值是一个用户活动过的所有日期的列表。

mapValues(len)：对每个用户的日期列表应用 len 函数，len 函数用于返回对象中元素的数量，此处的元素数量就是每个用户在 2014 年 8 月的活动日期数量，最终输出格式为(用户 ID，活动日期数)。

filter(lambda x: x[1] >= 5)：筛选活跃天数 x[1]>=5 的用户。

count()：计算 active_users 中的元素数量，即活跃用户的总数。

```
# 过滤2014年8月的数据
parsed_data = data.map(parse_line).filter(lambda x: x is not None)
# 统计每个用户在2014年8月的不同日期数量
user_activity = parsed_data.map(lambda x: (x[0], x[1])).distinct().\
    groupByKey().mapValues(len)
# 筛选出活跃用户
active_users = user_activity.filter(lambda x: x[1] >= 5)
# 获取活跃用户的总数
active_user_count = active_users.count()
print(f"{active_user_count}")
```

运行程序:

使用“./bin/spark-submit --master local[*] /home/njucs/Desktop/test4/work1/step2.py”提交 step2 的代码，其中 master local[*]是本地运行

```
njucs@njucs-VirtualBox:~/spark-3.5.3$ ./bin/spark-submit --master local[*] /home
/njucs/Desktop/test4/work1/step2.py
24/12/16 14:55:36 WARN Utils: Your hostname, njucs-VirtualBox resolves to a loop
back address: 127.0.1.1; using 192.168.208.146 instead (on interface ens33)
24/12/16 14:55:36 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
24/12/16 14:55:38 INFO SparkContext: Running Spark version 3.5.3
24/12/16 14:55:38 INFO SparkContext: OS info Linux, 5.0.0-23-generic, amd64
24/12/16 14:55:38 INFO SparkContext: Java version 11.0.19
24/12/16 14:55:38 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
24/12/16 14:55:39 INFO ResourceUtils: ==========================================
==================
24/12/16 14:55:39 INFO ResourceUtils: No custom resources configured for spark.d
river.
24/12/16 14:55:39 INFO ResourceUtils: ==========================================
==================
24/12/16 14:55:39 INFO SparkContext: Submitted application: Active Users in Augu
st 2014
24/12/16 14:55:39 INFO ResourceProfile: Default ResourceProfile created, executo
r resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory ->
name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount
```

输出保存在/home/njucs/Desktop/test4/work1/output2 中

```
24/12/16 14:56:03 INFO DAGScheduler: Job 0 finished: count at /home/njucs/Deskto
p/test4/work1/step2.py:31, took 19.757833 s
12767
```

遇到的问题：本次任务暂未遇到问题

任务 2：Spark SQL 编程

1. 按城市统计 2014 年 3月1日的平均余额:

计算每个城市在 2014 年 3月1日的用户平均余额 tBalance ，按平均余额降序排列。

格式：<城市 ID> <平均余额>

思路:

查询 **user_balance_table** 和 **user_profile_table，**将两表通过 **user_id** 联立起来，即将相同 **user_id** 行合并。先筛选 report_date 为 20140301 的数据，然后计算每个城市的用户账户余额的平均值，最后排序后输出。

核心代码：

处理数据：将 tBalance 转换为 int 型，方便接下来求均值，然后再注册临时视图。

```
# 转换字段类型
user_balance_table = user_balance_table.\
    withColumn("tBalance", col("tBalance").cast("int"))

# 注册临时视图
user_balance_table.createOrReplaceTempView("user_balance")
user_profile_table.createOrReplaceTempView("user_profile")
```

使用 Spark SQL 执行查询：

```
SELECT up.city,                          //选择 up 表中 的 city 列作为输出之一
AVG(ub.tBalance) AS avg_balance          //计算 ub 表中每个城市的用户账户余额的平均值
FROM user_balance ub                     //指定数据源为 user_balance_table，取别名 ub
JOIN user_profile up ON ub.user_id = up.user_id //加入另一个表 up，连接两表
WHERE ub.report_date = '20140301' //筛选 report_date 为 20140301 的数据
GROUP BY up.city                         //根据 up.city 对数据进行分组
ORDER BY avg_balance DESC                //对结果按照 avg_balance 进行从大到小排序
```

```
# 使用Spark SQL执行查询：按城市统计2014年3月1日的平均余额
query = """
SELECT up.city, AVG(ub.tBalance) AS avg_balance
FROM user_balance ub
JOIN user_profile up ON ub.user_id = up.user_id
WHERE ub.report_date = '20140301'
GROUP BY up.city
ORDER BY avg_balance DESC
"""
```

运行程序：

使用“./bin/spark-submit --master local[*] /home/njucs/Desktop/test4/work2/step1.py”提交 step1 的代码，其中 master local[*]是本地运行

```
njucs@njucs-VirtualBox:~/spark-3.5.3$ ./bin/spark-submit --master local[*] /home
/njucs/Desktop/test4/work2/step1.py
24/12/17 09:09:29 WARN Utils: Your hostname, njucs-VirtualBox resolves to a loop
back address: 127.0.1.1; using 192.168.208.146 instead (on interface ens33)
24/12/17 09:09:29 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
24/12/17 09:09:31 INFO SparkContext: Running Spark version 3.5.3
24/12/17 09:09:31 INFO SparkContext: OS info Linux, 5.0.0-23-generic, amd64
24/12/17 09:09:31 INFO SparkContext: Java version 11.0.19
24/12/17 09:09:31 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
24/12/17 09:09:31 INFO ResourceUtils: ==========================================
====================
24/12/17 09:09:31 INFO ResourceUtils: No custom resources configured for spark.d
river.
24/12/17 09:09:31 INFO ResourceUtils: ==========================================
====================
24/12/17 09:09:31 INFO SparkContext: Submitted application: City_Avg_Balance
24/12/17 09:09:31 INFO ResourceProfile: Default ResourceProfile created, executo
r resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory ->
name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount
: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
```

输出保存在/home/njucs/Desktop/test4/work2/output1 中

```
24/12/17 09:09:54 INFO CodeGenerator: Code generated in 41.013921 ms
24/12/17 09:09:54 INFO CodeGenerator: Code generated in 46.49714 ms
+-------+------------------+
|   city|       avg_balance|
+-------+------------------+
|6281949| 2795923.837298216|
|6301949|2650775.0664451825|
|6081949|2643912.7566638007|
|6481949|2087617.2136986302|
|6411949|1929838.5617977527|
|6412149| 1896363.471625767|
|6581949|1526555.5551020408|
+-------+------------------+

24/12/17 09:09:54 INFO FileSourceStrategy: Pushed Filters: IsNotNull(report_date
),EqualTo(report_date,20140301),IsNotNull(user_id)
24/12/17 09:09:54 INFO FileSourceStrategy: Post-Scan Filters: isnotnull(report_d
```

遇到的问题：本次任务暂未遇到问题

2. 统计每个城市总流量前 3 高的用户:

统计每个城市中每个用户在 2014 年 8 月的总流量（定义为 total_purchase_amt + total_redeem_amt），并输出每个城市总流量排名前三的用户 ID 及其总流量。

格式：<城市 ID> <用户 ID> <总流量>

思路：查询 **user_balance_table** 和 **user_profile_table**，将两表通过 **user_id** 联立起来，即将相同 **user_id** 行合并。先筛选 report_date 在 **20140801** 到 **20140831** 之间的数据，然后计算每个用户总流量 total_flow，然后再按城市分组，给每个城市中的用户的 total_flow 排序，最后输出每个城市排名前三的用户。

核心代码：

```
WITH UserTotalFlow AS (
//定义临时子查询,UserTotalFlow 为临时表，储存每个用户的总流量
 SELECT up.city,ub.user_id,
 SUM(ub.total_purchase_amt + ub.total_redeem_amt) AS total_flow
//选择 up 中的 city,ub 中的 user_id 并计算每个用户的总流量 total_flow 为查询对象
 FROM user_balance ub //指定数据源为 user_balance，取别名 ub
 JOIN user_profile up //加入另一个表 up
 ON ub.user_id = up.user_id//通过相同的用户 id 连接两表
WHERE ub.report_date BETWEEN '20140801' AND '20140831' //设置日期条件
  GROUP BY up.city, ub.user_id), //按城市和 id 分组
  RankedUsers AS (//RankedUsers 为临时表，代表每个城市中的用户的排名
SELECT city,user_id,total_flow,
ROW_NUMBER() OVER (PARTITION BY city ORDER BY total_flow DESC) AS rank
  //选择 city，user_id，total_flow，并给每个用户在其城市中的 total_flow 降序排列
  生成一个排名为 rank 为查询对象
FROM UserTotalFlow)//指定数据源为 UserTotalFlow ， 结束 RankedUsers 子查询
  SELECT city, user_id, total_flow//选择 city，user_id，total_flow 为查询对象
  FROM Rankedusers        //指定数据源为 ranked_users
```

WHERE rank <= 3 //设置条件，只返回排名在前 3 的用户

ORDER BY city, rank; //先按城市分组，再在每个城市内按排名排序

```
# 使用Spark SQL进行查询:统计每个城市前3名总流量用户
query = """
WITH UserTotalFlow AS (
    SELECT up.city,ub.user_id,
    SUM(ub.total_purchase_amt + ub.total_redeem_amt) AS total_flow
    FROM user_balance ub JOIN user_profile up ON ub.user_id = up.user_id
    WHERE ub.report_date BETWEEN '20140801' AND '20140831'
    GROUP BY up.city, ub.user_id),
RankedUsers AS (
    SELECT city,user_id,total_flow,
    ROW_NUMBER() OVER (PARTITION BY city ORDER BY total_flow DESC) AS rank
    FROM UserTotalFlow
)
SELECT city, user_id, total_flow
FROM RankedUsers
WHERE rank <= 3
ORDER BY city, rank;
"""
```

运行程序:

使用“./bin/spark-submit --master local[*] /home/njucs/Desktop/test4/work2/step2.py”提交 step2 的代码，其中 master local[*]是本地运行

```
njucs@njucs-VirtualBox:~/spark-3.5.3$ ./bin/spark-submit --master local[*] /home
/njucs/Desktop/test4/work2/step2.py
24/12/17 21:30:34 WARN Utils: Your hostname, njucs-VirtualBox resolves to a loop
back address: 127.0.1.1; using 192.168.208.147 instead (on interface ens33)
24/12/17 21:30:34 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
24/12/17 21:30:35 INFO SparkContext: Running Spark version 3.5.3
24/12/17 21:30:35 INFO SparkContext: OS info Linux, 5.0.0-23-generic, amd64
24/12/17 21:30:35 INFO SparkContext: Java version 11.0.19
24/12/17 21:30:36 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
24/12/17 21:30:36 INFO ResourceUtils: ==========================================
====================
24/12/17 21:30:36 INFO ResourceUtils: No custom resources configured for spark.d
river.
24/12/17 21:30:36 INFO ResourceUtils: ==========================================
====================
24/12/17 21:30:36 INFO SparkContext: Submitted application: TopUsersByCity
24/12/17 21:30:36 INFO ResourceProfile: Default ResourceProfile created, executo
r resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory ->
name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount
```

输出保存在/home/njucs/Desktop/test4/work2/output1 中

```
+-------+-------+----------+
|   city|user_id|total_flow|
+-------+-------+----------+
|6081949|  27235| 108475680|
|6081949|  27746|  76065458|
|6081949|  18945|  55304049|
|6281949|  15118| 149311909|
|6281949|  11397| 124293438|
|6281949|  25814| 104428054|
|6301949|   2429| 109171121|
|6301949|  26825|  95374030|
|6301949|  10932|  74016744|
|6411949|    662|  75162566|
|6411949|  21030|  49933641|
|6411949|  16769|  49383506|
|6412149|  22585| 200516731|
|6412149|  14472| 138262790|
|6412149|  25147|  70594902|
|6481949|  12026|  51161825|
|6481949|    670|  49626204|
|6481949|  14877|  34488733|
|6581949|   9494|  38854436|
|6581949|  26876|  23449539|
+-------+-------+----------+
```

遇到的问题：此问题和实验三中的最后一个小问相同，但是批改后实验三中该问题出现错误，输出如下：

很明显连续出现了三个 22585, 这是错的, 因为每个用户可能在 8 月份有多次记录，需要将每个用户在 8 月份每天的记录再求和，才是总的流量，原来错误的代码是计算 **(ub.total_purchase_amt + ub.total_redeem_amt)** 应该改成 **SUM(ub.total_purchase_amt + ub.total_redeem_amt) AS total_flow**

```
|   city|user_id|total_flow|
+-------+-------+----------+
|6081949|  21932|  35400077|
|6081949|  27746|  28494771|
|6081949|  27235|  22435196|
|6281949|  15118| 147626250|
|6281949|  25814|  55040670|
|6281949|  22871|  49574861|
|6301949|   2429|  54636517|
|6301949|  10932|  51731105|
|6301949|   5330|  41828956|
|6411949|  21030|  48883465|
|6411949|  25025|  30539038|
|6411949|  25525|  26636950|
|6412149|  22585|  69556206|
|6412149|  22585|  47453506|
|6412149|  22585|  39838840|
|6481949|    670|  49497479|
|6481949|  21021|  31286393|
|6481949|  12026|  25079319|
|6581949|  21761|  14885517|
|6581949|   7107|   8743312|
+-------+-------+----------+
```

任务 3：Spark ML 编程

使用 Spark MLlib 提供的机器学习模型，预测 2014 年 9 月每天的申购与赎回总额。

格式：<日期> <申购总额><赎回总额>

思路：使用线性回归模型进行预测，预测的特征为年份，月份，日期

核心代码：

处理数据：只使用 user_balance_table 表中内容，需要用到的是日期和 total_purchase_amt 以及 total_redeem_amt, 首先是要将表中的相同日期的 total_purchase_amt 和 total_redeem_amt 的值加起来，求得每天的申购、赎回总额。然后处理 report_date，从中提取出 year,month 和 day 三个特征值，于是得到了训练集 daily_data。

```
# 加载数据
user_balance_table = spark.read.option("header", "true").csv("/home/njucs/Desktop/test4/Purchase_Redemption_Data/user_balance_table.csv")

user_balance_table = user_balance_table.withColumn("total_purchase_amt", F.col("total_purchase_amt").cast("double"))
user_balance_table = user_balance_table.withColumn("total_redeem_amt", F.col("total_redeem_amt").cast("double"))

# 计算用户的每日申购和赎回金额
daily_data = user_balance_table.groupBy("report_date").agg(
    F.sum("total_purchase_amt").alias("purchase"),
    F.sum("total_redeem_amt").alias("redeem")
)

# 从 report_date 字符串中提取年、月、日作为特征
daily_data = daily_data.withColumn("year", F.substring("report_date", 1, 4).cast("int"))
daily_data = daily_data.withColumn("month", F.substring("report_date", 5, 2).cast("int"))
daily_data = daily_data.withColumn("day", F.substring("report_date", 7, 2).cast("int"))

# 使用 year, month, day 作为特征列
assembler = VectorAssembler(inputCols=["year", "month", "day"], outputCol="features", handleInvalid="skip")
daily_data = assembler.transform(daily_data)
# 显示数据，查看结果
daily_data.show()
```

训练模型：

然后使用线性回归模型，以 year, month, day 作为特征列，训练模型。

handleInvalid="skip"：这表示如果有 null 值，则跳过这些数据行，从而增加鲁棒性。

assembler.transform(daily_data)：调用了 VectorAssembler 的 transform 方法，新增一列 features，将 year、month、day 三列的数据合并成一个向量。

LinearRegression：就是线性回归算法模型。

featuresCol="features"：指定输入特征列，即用于模型训练的输入特征向量所在的列名。

labelCol="purchase"：指定标签列，即模型要预测的目标变量所在的列名

```
# 使用 year, month, day 作为特征列
assembler = VectorAssembler(inputCols=["year", "month", "day"], outputCol="features", handleInvalid="skip")
daily_data = assembler.transform(daily_data)
# 显示数据，查看结果
daily_data.show()

# 线性回归模型预测申购金额（purchase）
lr_purchase = LinearRegression(featuresCol="features", labelCol="purchase")
lr_model_purchase = lr_purchase.fit(daily_data)

# 线性回归模型预测赎回金额（redeem）
lr_redeem = LinearRegression(featuresCol="features", labelCol="redeem")
lr_model_redeem = lr_redeem.fit(daily_data)
```

预测结果：

从 2014 年 9 月 1 日到 30 日中，分别提取年月日，构造 30 个向量，然后输入训练好的模型，最终得到预测的申购、赎回总额。

```
dates_sept_2014 = spark.createDataFrame(
    [(f"201409{day:02d}",) for day in range(1, 31)], ["report_date"]
)

# 从 report_date 字符串中提取年、月、日作为特征
dates_sept_2014 = dates_sept_2014.withColumn("year", F.substring("report_date", 1, 4).cast("int"))
dates_sept_2014 = dates_sept_2014.withColumn("month", F.substring("report_date", 5, 2).cast("int"))
dates_sept_2014 = dates_sept_2014.withColumn("day", F.substring("report_date", 7, 2).cast("int"))
# 通过 VectorAssembler 构建特征
dates_sept_2014 = assembler.transform(dates_sept_2014)
dates_sept_2014.show()
# 使用线性回归模型进行预测
predictions_purchase = lr_model_purchase.transform(dates_sept_2014)
predictions_redeem = lr_model_redeem.transform(dates_sept_2014)
```

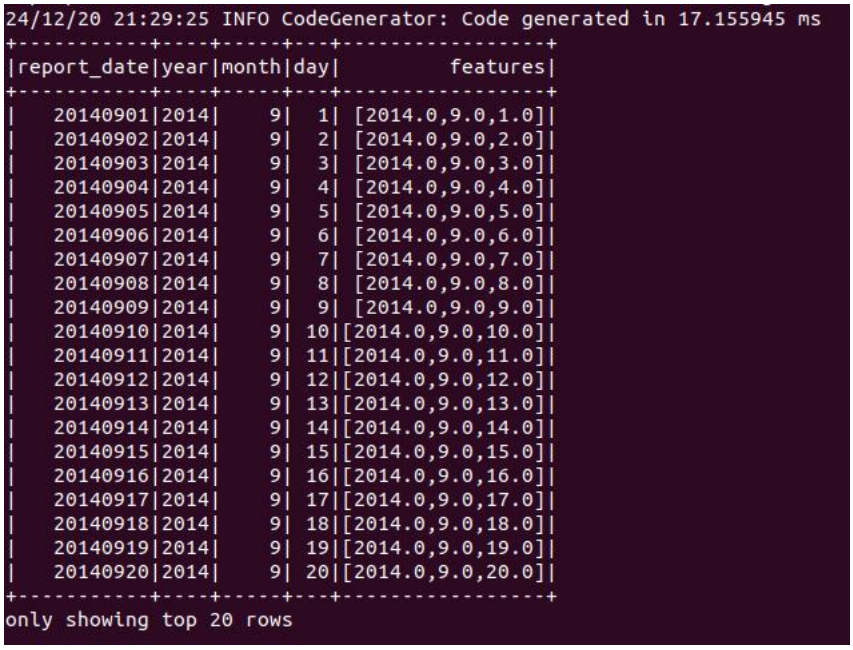
运行程序：

使用“./bin/spark-submit --master local[*] /home/njucs/Desktop/test4/work3/ML.py”提交 ML.py 的代码，其中 master local[*]是本地运行

运行过程中输出了训练集的部分内容：

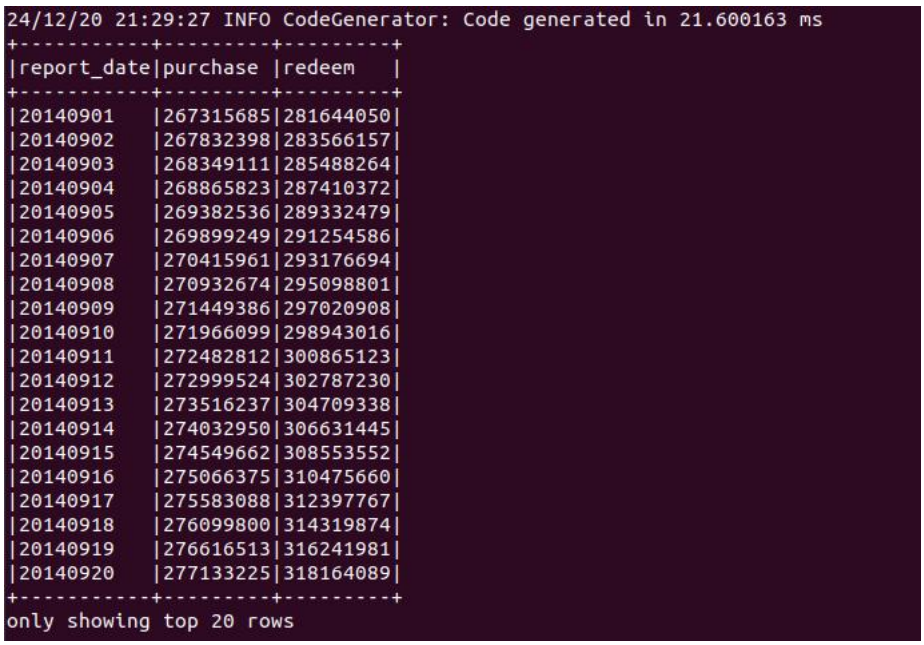
```
24/12/20 21:28:50 INFO CodeGenerator: Code generated in 19.14368 ms
+-----------+------------+------------+----+-----+---+------------------+
|report_date|    purchase|      redeem|year|month|day|          features|
+-----------+------------+------------+----+-----+---+------------------+
|   20140413|2.08172985E8|1.78934722E8|2014|    4| 13| [2014.0,4.0,13.0]|
|   20130919| 2.4778048E7| 1.1418512E7|2013|    9| 19| [2013.0,9.0,19.0]|
|   20140711|2.08671021E8|2.40050748E8|2014|    7| 11| [2014.0,7.0,11.0]|
|   20140410| 3.8656746E8|2.86914864E8|2014|    4| 10| [2014.0,4.0,10.0]|
|   20140512|3.25108597E8|2.93952908E8|2014|    5| 12| [2014.0,5.0,12.0]|
|   20140530|2.26547701E8|3.12802179E8|2014|    5| 30| [2014.0,5.0,30.0]|
|   20140303|5.05305862E8| 5.1301736E8|2014|    3|  3|  [2014.0,3.0,3.0]|
|   20140202| 5.7912761E7|  2.439572E7|2014|    2|  2|  [2014.0,2.0,2.0]|
|   20140817|1.49978271E8|1.39564084E8|2014|    8| 17| [2014.0,8.0,17.0]|
|   20140310|4.97338076E8|3.08040624E8|2014|    3| 10| [2014.0,3.0,10.0]|
|   20130809| 3.3425186E7| 3.0131015E7|2013|    8|  9|  [2013.0,8.0,9.0]|
|   20130722| 4.0448896E7| 1.9144267E7|2013|    7| 22| [2013.0,7.0,22.0]|
|   20131015| 8.5704304E7| 3.5099198E7|2013|   10| 15|[2013.0,10.0,15.0]|
|   20130817| 1.7670519E7|   4674983.0|2013|    8| 17| [2013.0,8.0,17.0]|
|   20140505|3.70924149E8|3.09330781E8|2014|    5|  5|  [2014.0,5.0,5.0]|
|   20140709|2.78005555E8|2.69642881E8|2014|    7|  9|  [2014.0,7.0,9.0]|
|   20140114|3.56907128E8|1.59778389E8|2014|    1| 14| [2014.0,1.0,14.0]|
|   20131029|1.02050144E8| 4.3052468E7|2013|   10| 29|[2013.0,10.0,29.0]|
|   20130709| 2.6798941E7|   3473059.0|2013|    7|  9|  [2013.0,7.0,9.0]|
|   20140130|2.27916211E8|1.05288105E8|2014|    1| 30| [2014.0,1.0,30.0]|
+-----------+------------+------------+----+-----+---+------------------+
only showing top 20 rows
```

输入集合的部分内容：



```
24/12/20 21:29:25 INFO CodeGenerator: Code generated in 17.155945 ms
+-----------+----+-----+---+-----------------+
|report_date|year|month|day|         features|
+-----------+----+-----+---+-----------------+
|   20140901|2014|    9|  1| [2014.0,9.0,1.0]|
|   20140902|2014|    9|  2| [2014.0,9.0,2.0]|
|   20140903|2014|    9|  3| [2014.0,9.0,3.0]|
|   20140904|2014|    9|  4| [2014.0,9.0,4.0]|
|   20140905|2014|    9|  5| [2014.0,9.0,5.0]|
|   20140906|2014|    9|  6| [2014.0,9.0,6.0]|
|   20140907|2014|    9|  7| [2014.0,9.0,7.0]|
|   20140908|2014|    9|  8| [2014.0,9.0,8.0]|
|   20140909|2014|    9|  9| [2014.0,9.0,9.0]|
|   20140910|2014|    9| 10|[2014.0,9.0,10.0]|
|   20140911|2014|    9| 11|[2014.0,9.0,11.0]|
|   20140912|2014|    9| 12|[2014.0,9.0,12.0]|
|   20140913|2014|    9| 13|[2014.0,9.0,13.0]|
|   20140914|2014|    9| 14|[2014.0,9.0,14.0]|
|   20140915|2014|    9| 15|[2014.0,9.0,15.0]|
|   20140916|2014|    9| 16|[2014.0,9.0,16.0]|
|   20140917|2014|    9| 17|[2014.0,9.0,17.0]|
|   20140918|2014|    9| 18|[2014.0,9.0,18.0]|
|   20140919|2014|    9| 19|[2014.0,9.0,19.0]|
|   20140920|2014|    9| 20|[2014.0,9.0,20.0]|
+-----------+----+-----+---+-----------------+
only showing top 20 rows
```

最终预测值的部分内容，完整输出保存在/home/njucs/Desktop/test4/work3/output 中：



```
24/12/20 21:29:27 INFO CodeGenerator: Code generated in 21.600163 ms
+-----------+---------+---------+
|report_date|purchase |redeem   |
+-----------+---------+---------+
|20140901   |267315685|281644050|
|20140902   |267832398|283566157|
|20140903   |268349111|285488264|
|20140904   |268865823|287410372|
|20140905   |269382536|289332479|
|20140906   |269899249|291254586|
|20140907   |270415961|293176694|
|20140908   |270932674|295098801|
|20140909   |271449386|297020908|
|20140910   |271966099|298943016|
|20140911   |272482812|300865123|
|20140912   |272999524|302787230|
|20140913   |273516237|304709338|
|20140914   |274032950|306631445|
|20140915   |274549662|308553552|
|20140916   |275066375|310475660|
|20140917   |275583088|312397767|
|20140918   |276099800|314319874|
|20140919   |276616513|316241981|
|20140920   |277133225|318164089|
+-----------+---------+---------+
only showing top 20 rows
```

在平台上提交后的分数：

日期: 2024-12-20 21:33:32

分数: 76.5435

日期: 2024-12-20 21:32:44

分数:

ERROR Bad input file

日期: 2024-12-20 19:34:09

分数: 0.0000

遇到的问题：平台上要求输出的预测值以分为单位，但是本来提供的数据中，金额就是以分为单位，不需要再特地乘 100，不然预测值的数量级出错，得分就是 0 分。