

合肥工业大学

专业课程

(计算机与信息学院)

大数据处理技术实验报告

专 业 班 级	计算机科学与技术 16-3 班
学 生 姓 名 及 学 号	任恒 2016212063
课 程 教 学 班 号	001 班
任 课 教 师	吴共庆
实 验 指 导 教 师	吴共庆
实 验 地 点	科教楼 D502

2018 ~ 2019 学 年 第 二 学 期

目录

综合设计-大数据处理技术 2018-2019 第二学期.....	3
1. Hadoop 分布式环境搭建与 Hive 安装	4
(1) 安装 Hadoop	4
(2) 安装 Hive	8
2. Hive 数据清洗.....	8
1) 创建 raw_test 数据库	9
2) 创建 shop1, shop2 表	9
3) 创建 shop1_1 表.....	11
4) 创建 shop1_2 表.....	11
5) 创建 shop2_1 表.....	12
3. 数据分析.....	13
1) 统计评论量最多的 5 部电影	14
2) 统计 2013 年以后电影最多的 5 个地区	15
3) 电影总体热度系数	17
4. 数据可视化.....	18
1) 折线图.....	19
2) 雷达图.....	21
5. 总结	24
5. 总结	24

综合设计-大数据处理技术 2018-2019 第二学期

根据麦肯锡全球研究所的定义,大数据是一种规模大到在获取、存储、管理、分析方面大大超出了传统数据库软件工具能力范围的数据集合,具有海量的数据规模、快速的数据流转、多样的数据类型和价值密度低四大特征。而当前人们所说的人工智能,是指研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术以及应用系统的一门新的技术科学,是由人工制造出来的系统所表现出来的智能。但由于传统人工智能受制于计算能力,并没能完成大规模的并行计算和并行处理,人工智能系统的能力较差。

随着移动互联网的爆发,数据量呈现出指数级的增长,大数据的积累为人工智能提供了基础支撑。同时受益于计算机技术在数据采集、存储、计算等环节的突破,人工智能已从简单的算法+数据库发展演化到了机器学习+深度理解的状态。大数据+人工智能帮助了各行各业的企业从原本毫无价值的海量数据中挖掘出用户的需求,使数据能够从量变到质变,真正产生价值。

本综合设计将分为大数据平台搭建、数据预处理、数据分析、数据可视化以及综合题等五部分内容,请按照要求完成。

1. Hadoop 分布式环境搭建与 Hive 安装. （30 分）

请先按照如下规划表对三台虚拟机进行配置

主机名	IP	备注
master	192.168.10.2	主节点
slave1	192.168.10.3	从节点
slave2	192.168.10.4	从节点

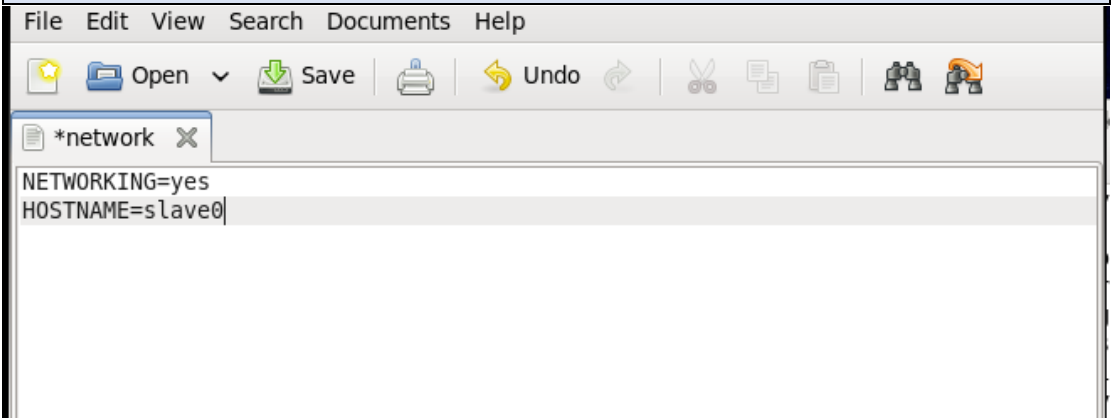
安装包均放在/opt/hadoop-package/文件夹下，要求如下：

(1) 安装 Hadoop （20 分）

1) 配置 hdfs-site.xml，内容为文件的副本数 1。并将配置页面截图保存到实验报告任务 1-1-1 中。（4 分）

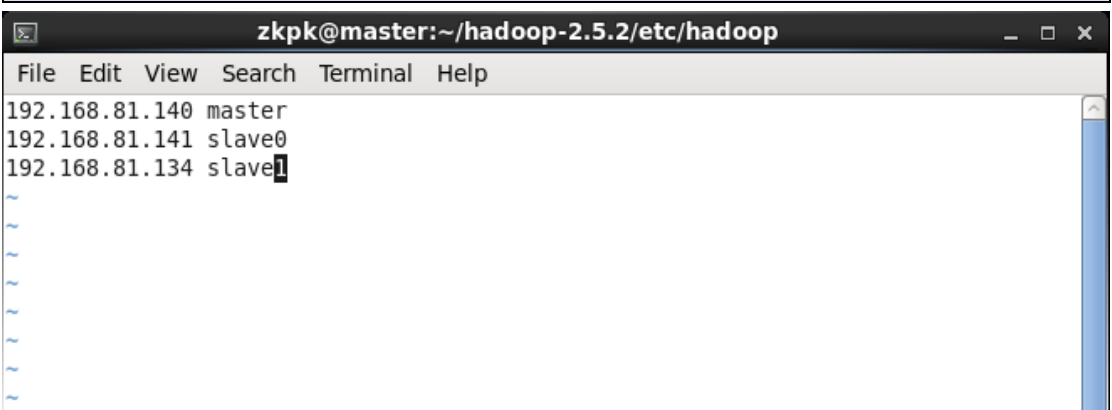
更改主机名：将 hostname 改为 master、salve0、slave1.

```
>>gedit /etc/sysconfig/network
```



在主节点中配置 hosts 文件：

```
>>sudo vim /etc/hosts
```



将主节点上 hosts 文件复制到两个 slave 节点上:

```
[zkpk@master hadoop]$ scp /etc/hosts root@192.168.81.134:/etc
root@192.168.81.134's password:
hosts                                100%  66    0.1KB/s   00:00
[zkpk@master hadoop]$ scp /etc/hosts root@192.168.81.141:/etc
The authenticity of host '192.168.81.141 (192.168.81.141)' can't be established.
RSA key fingerprint is 5b:4b:df:e4:26:47:c0:31:54:ec:22:3b:6c:7b:d6:bd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.81.141' (RSA) to the list of known hosts.
root@192.168.81.141's password:
Permission denied, please try again.
root@192.168.81.141's password:
hosts                                100%  66    0.1KB/s   00:00
[zkpk@master hadoop]$
```

三个节点刷新 hostname: (如 master)

```
>>hostname master
```

配置密钥:

每个节点下分别执行:

```
>>ssh-keygen rsa
```

```
[root@slave1 Desktop]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
e6:64:fd:c9:48:07:d7:1c:8d:59:23:34:4b:ba:dd:97 root@slave
The key's randomart image is:
+--[ RSA 2048 ]-----+
|          . = . * |
|         00 * . 0 |
|        .... 0   |
|       . 00 . .  |
|      S 0... E.  |
|     = . = . .   |
|    . . +        |
|                  |
+-----+
[root@slave1 Desktop]#
```

将从节点密钥拷贝到主节点 (以 slave1 为例):

```
>>scp ~/.ssh/id_rsa.pub root@master:~/.ssh/id_rsa_slave1.pub
```

主节点上用 scp 命令将 authorized_keys 文件拷贝到子节点上的 "/root/.ssh"目录下

```
>>scp authorized_keys root@slave0:/root/.ssh/
```

```
>>scp authorized_keys root@slave1:/root/.ssh/
```

hdfs-site.xml 文件配置:

```
>>gedit ~/Hadoop-2.5.2/etc/Hadoop/hdfs-site.xml
```

Slave 文件配置:

```
>>gedit ~/hadoop-2.5.2/etc/hadoop/slave
```

输入:

```
slave0
```

```
slave1
```

拷贝主节点上的"/etc/profile"文件到子节点:

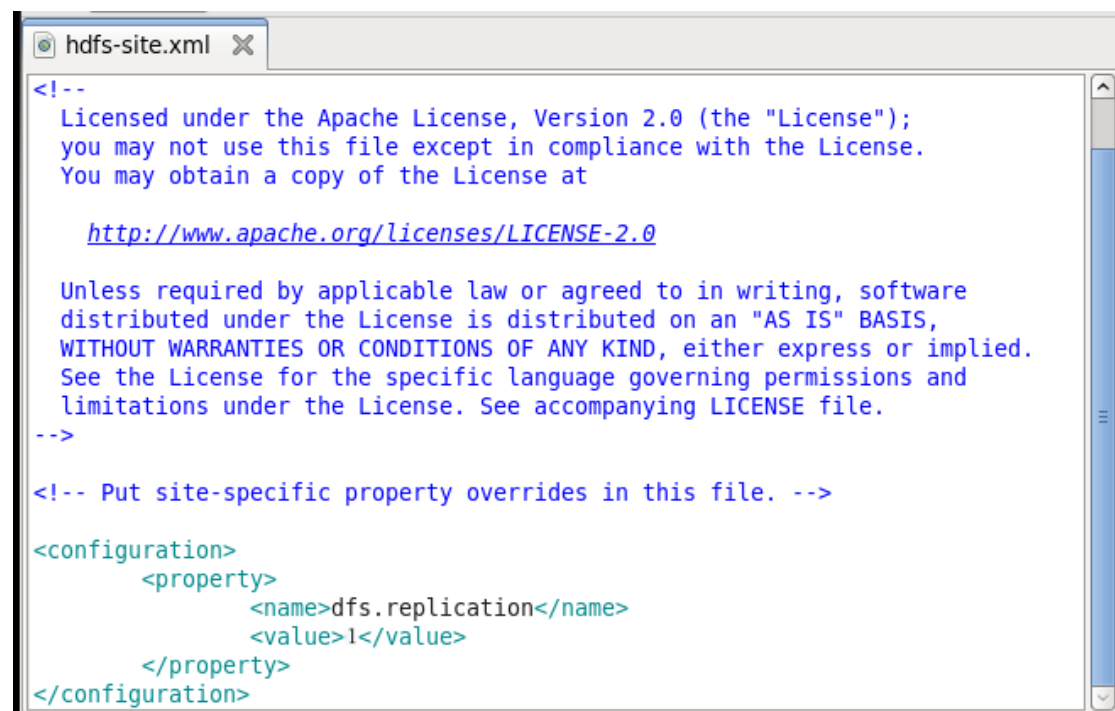
```
>>scp /etc/profile root@slave0:/etc/
```

```
>>scp /etc/profile root@slave1:/etc/
```

并分别在子节点上使 "/etc/profile"文件生效:

```
>>source /etc/profile
```

其他文件配置参考教程（此处主要是将两个节点扩展到三个节点）



```
hdfs-site.xml X
<!--
Licensed 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

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

2) 配置 mapred-site.xml, 指定 mapreduce 的环境为 yarn。并将配置页面截图保存到实验报告任务 1-1-2 中。(4 分)

```
>>gedit ~/Hadoop-2.5.2/etc/hadoop/mapred-site.xml
```

```
mapred-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed 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

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

3) 设置 `mapreduce.map.java.opts` 的值为 `-Xmx768m`, 设置 `mapreduce.reduce.memory.mb` 的值为 `2048MB`, 设置 `mapreduce.reduce.java.opts` 的值为 `-Xmx1536m`. 并将配置页面截图并保存到实验报告任务 1-1-3 中。(6 分)

```
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.map.memory.mb</name>
    <value>2048</value>
  </property>
  <property>
    <name>mapreduce.map.java.opts</name>
    <value>-Xmx768m</value>
  </property>
  <property>
    <name>mapreduce.reduce.java.opts</name>
    <value>-Xmx1536m</value>
  </property>
</configuration>
```

启动集群。在三个节点上分别执行 `jps` 命令查看 Hadoop 进程并将查看到的进程结果截图保存到实验报告任务 1-1-4 中。(6 分)

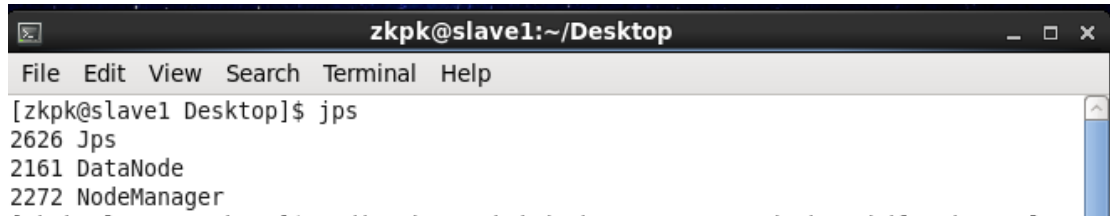
主节点查看启动的情况:

```
[zkpk@master hadoop]$ jps
2795 SecondaryNameNode
3387 Jps
2935 ResourceManager
2605 NameNode
```

Slave0 查看启动的情况:

```
zkpk@slave0:~/Desktop
File Edit View Search Terminal Help
[zkpk@slave0 Desktop]$ jps
2524 NodeManager
2672 Jps
2419 DataNode
[zkpk@slave0 Desktop]$
```

Slave1 查看启动的情况:



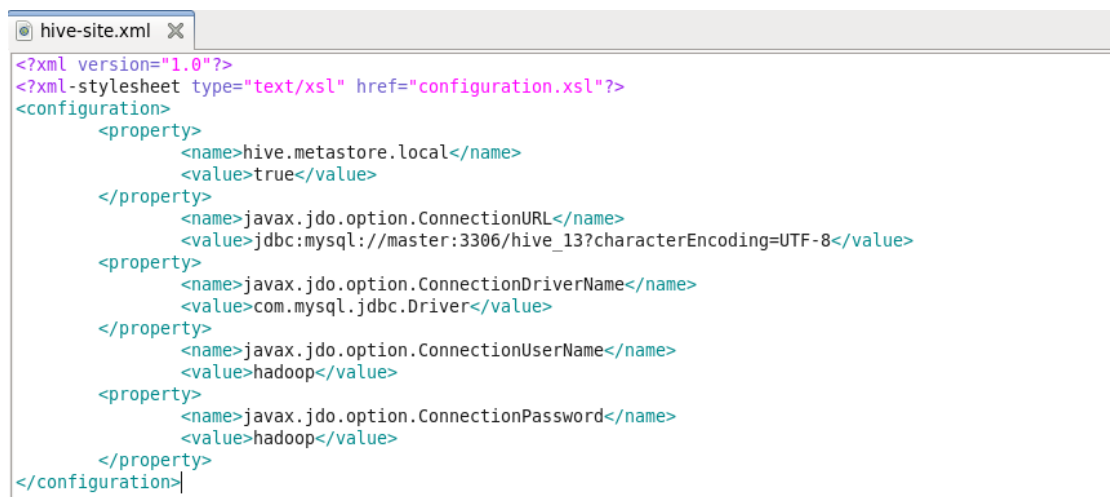
```
zkpk@slave1:~/Desktop
File Edit View Search Terminal Help
[zkpk@slave1 Desktop]$ jps
2626 Jps
2161 DataNode
2272 NodeManager
```

(2) 安装 Hive (mysql 已经安装好) (10 分)

拷贝 hive-env.sh.template 并命名为 hive-env.sh, 配置 hive-env.sh, 内容为 Hadoop 安装目录。并将配置页面截图保存到实验报告任务 1-2-1 中。(2 分) 配置 hive-env.sh:

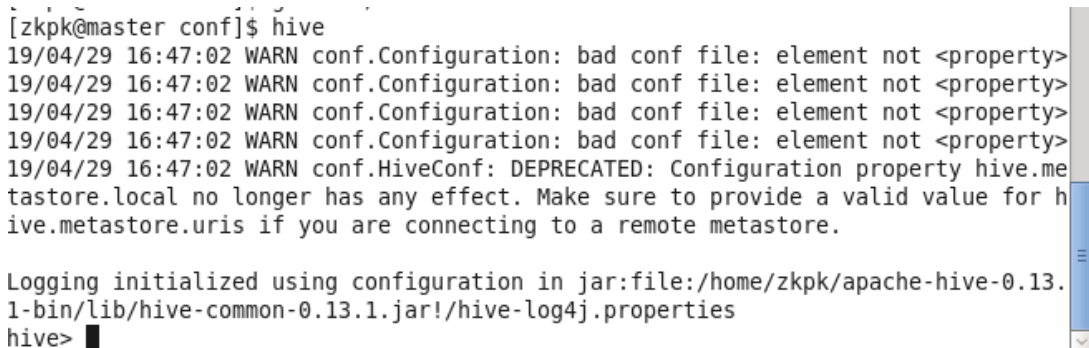
```
# Set HADOOP_HOME to point to a specific hadoop install directory
HADOOP_HOME=~/hadoop-2.5.2/
```

2) 新建 hive.site.xml (可参考如下截图中的 hive-site.xml 模板, 注意: 星号内容需自行补充) 添加 mysql 连接信息。并将配置页面截图保存到实验报告任务 1-2-2 中。(2 分)



```
hive-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>hive.metastore.local</name>
    <value>true</value>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://master:3306/hive_13?characterEncoding=UTF-8</value>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>hadoop</value>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>hadoop</value>
  </property>
</configuration>
```

3) 启动 hive, 并将查看到的结果截图保存到实验报告任务 1-2-3 中。(6 分)



```
[zkpk@master conf]$ hive
19/04/29 16:47:02 WARN conf.Configuration: bad conf file: element not <property>
19/04/29 16:47:02 WARN conf.Configuration: bad conf file: element not <property>
19/04/29 16:47:02 WARN conf.Configuration: bad conf file: element not <property>
19/04/29 16:47:02 WARN conf.Configuration: bad conf file: element not <property>
19/04/29 16:47:02 WARN conf.HiveConf: DEPRECATED: Configuration property hive.metastore.local no longer has any effect. Make sure to provide a valid value for hive.metastore.uris if you are connecting to a remote metastore.

Logging initialized using configuration in jar:file:/home/zkpk/apache-hive-0.13.1-bin/lib/hive-common-0.13.1.jar!/hive-log4j.properties
hive>
```

2. Hive 数据清洗 (30 分)

(在 Hive 交互环境下完成操作)

“安徽省大数据学院”跟学校进行校企合作，提供了一些商品交易的数据源给学校学生进行数据清洗，想请你帮他们完成数据的清洗。

“数据源”目录下有 shop_1.txt 和 shop_2.txt 数据源，里面是一些商品的交易记录，数据格式如下：

其中，数据源属性从左往右以此是 id(商品记录 id)，goods_name(商品名称)，biz_price(商品交易金额)，count(商品交易数量)，biz_date(商品交易日期)

ID	goods_name	biz_price	count	biz_date
1:	钢笔:	105.00:	7:	2018-7-20 10:45:09
2:	钢笔:	15.00:	1:	2018-7-1 20:59:17
3:	圆珠笔:	12.00:	6:	2018-7-29 07:51:50
4:	钢笔:	225.00:	15:	2018-8-30 06:05:20
5:	足球:	960.00:	12:	2018-9-29 07:59:21
6:	橡皮:	1.50:	3:	2018-8-15 15:39:11
7:	钢笔:	45.00:	3:	2018-9-9 08:33:11
8:	足球:	0.00:	0:	2018-8-28 04:20:11
9:	钢笔:	105.00:	7:	2018-7-9 08:06:16
10:	笔记本:	96.00:	12:	2018-7-19 06:44:46
11:	篮球:	550.00:	11:	2018-9-1 13:11:38
12:	羽毛球拍:	-20.00:	-1:	2018-9-19 14:44:40
13:	水彩套餐:	360.00:	9:	2018-9-12 00:30:39
14:	橡皮:	-0.50:	-1:	2018-7-7 09:58:25
15:	水彩套餐:	-80.00:	-2:	2018-7-25 12:18:11
16:	象棋:	220.00:	11:	2018-9-9 22:45:35
17:	铅笔:	5.00:	5:	2018-8-28 17:22:47
18:	水彩套餐:	-40.00:	-1:	2018-9-5 13:57:27
19:	象棋:	300.00:	15:	2018-9-29 03:36:25
20:	圆珠笔:	30.00:	15:	2018-8-16 02:11:45
21:	篮球:	150.00:	3:	2018-9-13 21:33:18
22:	橡皮:	9.00:	18:	2018-9-1 04:01:20
23:	笔记本:	88.00:	11:	2018-9-15 03:07:11
24:	象棋:	120.00:	6:	2018-9-19 22:33:23
25:	橡皮:	1.50:	3:	2018-7-22 20:03:13
26:	钢笔:	225.00:	15:	2018-8-4 21:02:47
27:	圆珠笔:	20.00:	10:	2018-9-16 07:04:33
28:	羽毛球拍:	380.00:	19:	2018-9-10 22:10:01
29:	钢笔:	225.00:	15:	2018-7-24 23:38:44
30:	钢笔:	165.00:	11:	2018-9-8 14:33:57
31:	篮球:	0.00:	0:	2018-8-18 12:26:38
32:	篮球:	100.00:	2:	2018-8-26 11:03:48

要求如下：

1) 创建 raw_test 数据库，进入 raw_test 数据库。并将该截图保存到实验报告任务 2-1-1 中。(5 分)

创建 raw_test 数据库：

```
hive> create database raw_test;
OK
Time taken: 0.069 seconds
hive>
```

2) 基于 raw_test 数据库，创建 shop1, shop2 表，将数据源 shop_1.txt 和 shop_2.txt 分别导入对应的表中。分别查询两个表中的前十条数据并将这两个截图保存到实验报告任务 2-1-2 中。(10 分)

创建 shop1 表:

```
<<create table shop1(id string,goods_name string,biz_prize string,count string,biz_data string);
```

```
hive> create table shop1(id string,goods_name string,biz_prize string,count string,biz_data string);
OK
Time taken: 0.25 seconds
hive>
```

创建 shop2 表:

```
<<create table shop2(id string,goods_name string,biz_prize string,count string,biz_data string);
```

```
hive> create table shop2(id string,goods_name string,biz_prize string,count string,biz_data string);
OK
Time taken: 0.039 seconds
hive>
```

```
hive> show tables;
OK
shop1
shop2
Time taken: 0.16 seconds, Fetched: 2 row(s)
hive>
```

上传数据到 shop1 和 shop2 表中:

```
hive> load data local inpath '/home/zkpk/shop_1.txt' into table shop1;
Copying data from file:/home/zkpk/shop_1.txt
Copying file: file:/home/zkpk/shop_1.txt
Loading data to table default.shop1
Table default.shop1 stats: [numFiles=1, numRows=0, totalSize=41458, rawDataSize=0]
OK
Time taken: 0.747 seconds
hive> load data local inpath '/home/zkpk/shop_2.txt' into table shop2;
Copying data from file:/home/zkpk/shop_2.txt
Copying file: file:/home/zkpk/shop_2.txt
Loading data to table default.shop2
Table default.shop2 stats: [numFiles=1, numRows=0, totalSize=38283, rawDataSize=0]
OK
Time taken: 0.261 seconds
hive>
```

Shop1 中前十条数据:

```
hive> select * from shop1 limit 10;
OK
1      钢笔      105.00  7      2018-7-20 10:45:09
2      钢笔      15.00   1      2018-7-1 20:59:17
3      圆珠笔     12.00   6      2018-7-29 07:51:50
4      钢笔      225.00  15     2018-8-30 06:05:20
5      足球      960.00  12     2018-9-29 07:59:21
6      橡皮      1.50    3      2018-8-15 15:39:11
7      钢笔      45.00   3      2018-9-9 08:33:11
8      足球      0.00    0      2018-8-28 04:20:11
9      钢笔      105.00  7      2018-7-9 08:06:16
10     笔记本     96.00   12     2018-7-19 06:44:46
Time taken: 0.067 seconds, Fetched: 10 row(s)
```

Shop2 中前十条数据:

```
hive> select * from shop2 limit 10;
OK
1      羽毛球拍      80      4      2018-7-24 12:57:56
2      足球      640      8      2018-9-13 19:41:11
3      篮球      0      0      2018-8-21 20:17:46
4      笔记本      48      6      2018-8-18 11:19:09
5      铅笔      0      0      2018-8-14 10:43:23
6      水彩套餐      400      10      2018-7-18 08:38:08
7      羽毛球拍      100      5      2018-8-22 06:19:08
8      篮球      600      12      2018-9-22 01:58:19
9      橡皮      6      11      2018-7-12 07:52:42
10     羽毛球拍      60      3      2018-7-3 02:22:01
Time taken: 0.042 seconds, Fetched: 10 row(s)
hive>
```

3) 创建 shop1_1 表, 过滤掉 shop1 表中交易金额为负数或者交易数量为负数的记录, 并将结果保存到 shop1_1 表中。查询 shop1_1 表中的记录总数并将结果截图保存到实验报告任务 2-1-3 中。(5 分)

```
<<create table shop1_1(id string,goods_name string,biz_prize string,count
string,biz_data string) row format delimited fields terminated by '\t' stored as
textfile;
```

```
hive> insert into shop1_1 select * from shop1 where shop1.biz_prize>=0 and shop1.count>=0;
FAILED: ParseException line 1:12 missing TABLE at 'shop1_1' near '<EOF>'
hive> insert into table shop1_1 select * from shop1 where shop1.biz_prize>=0 and shop1.count>=0;
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1556556784806_0001, Tracking URL = http://master:18088/proxy/application_1556556784806_0001/
Kill Command = /home/hfut/hadoop-2.5.2/bin/hadoop job -kill job_1556556784806_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2019-04-30 01:19:02,945 Stage-1 map = 0%, reduce = 0%
2019-04-30 01:19:11,426 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.44 sec
MapReduce Total cumulative CPU time: 1 seconds 440 msec
Ended Job = job_1556556784806_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://master:9000/tmp/hive-hfut/hive_2019-04-30_01-18-48_925_508292962046404567-1/-ext-10000
Loading data to table raw test.shop1_1
Table raw test.shop1_1 stats: [numFiles=1, numRows=839, totalSize=33771, rawDataSize=32932]
MapReduce Jobs Launched:
Job 0: Map: 1 Cumulative CPU: 1.44 sec HDFS Read: 41679 HDFS Write: 33847 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 440 msec
OK
Time taken: 23.984 seconds
```

查询 shop1_1 表中的记录总数

```
<<select count(*) from shop1_1;
```

```
hive> select count(*) from shop1_1;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1556556784806_0002, Tracking URL = http://master:18088/proxy/application_1556556784806_0002/
Kill Command = /home/hfut/hadoop-2.5.2/bin/hadoop job -kill job_1556556784806_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2019-04-30 01:21:52,105 Stage-1 map = 0%, reduce = 0%
2019-04-30 01:22:00,418 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.04 sec
2019-04-30 01:22:08,729 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 1.04 sec
MapReduce Total cumulative CPU time: 1 seconds 40 msec
Ended Job = job_1556556784806_0002
MapReduce Jobs Launched:
Job 0: Map: 1 Reduce: 1 Cumulative CPU: 2.31 sec HDFS Read: 33992 HDFS Write: 4 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 310 msec
OK
839
Time taken: 27.107 seconds, Fetched: 1 row(s)
```

4) 创建 shop1_2 表, 过滤掉 shop1 表中存在字段为空的记录, 并将结果保存到

shop1_2 表中。查询 shop1_2 表中的记录总数并将结果截图保存到实验报告任务 2-1-4 中。（5 分）

```
<<create table shop1_2(id string,goods_name string,biz_prize string,count
string,biz_data string) row format delimited fields terminated by '\t' stored as
textfile;
```

```
hive> create table shop1_2(id string,goods_name string,biz_prize string,count string,biz_data string) row format delimited fields terminated by '\t' stored
as textfile;
OK
Time taken: 0.062 seconds
```

```
<<insert into table shop1_2 select * from shop1 where id!=null and
goods_name!=null and biz_prize!=null and count!=null and biz_data!=null;
```

```
hive> insert into table shop1_2 select * from shop1 where id!=null and goods_name!=null and biz_prize!=null and count!=null and biz_data!=null;
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1556556784806_0003, Tracking URL = http://master:18088/proxy/application_1556556784806_0003/
Kill Command = /home/hfut/hadoop-2.5.2/bin/hadoop job -kill job_1556556784806_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2019-04-30 01:25:48,396 Stage-1 map = 0%, reduce = 0%
2019-04-30 01:25:55,638 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
MapReduce Total cumulative CPU time: 1 seconds 320 msec
Ended Job = job_1556556784806_0003
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://master:9000/tmp/hive-hfut/hive_2019-04-30_01-25-39_292_6020379375125689106-1/-ext-10000
Loading data to table raw_test.shop1_2
Table raw_test.shop1_2 stats: [numFiles=1, numRows=0, totalSize=0, rawDataSize=0]
MapReduce Jobs Launched:
Job 0: Map: 1 Cumulative CPU: 1.32 sec HDFS Read: 41679 HDFS Write: 42 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 320 msec
OK
Time taken: 17.632 seconds
```

查询 shop1_2 表中的记录总数

```
<<select count(*) from shop1_2;
```

```
hive> select count(*) from shop1_2;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1556556784806_0006, Tracking URL = http://master:18088/proxy/application_1556556784806_0006/
Kill Command = /home/hfut/hadoop-2.5.2/bin/hadoop job -kill job_1556556784806_0006
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2019-04-30 01:39:47,308 Stage-1 map = 0%, reduce = 0%
2019-04-30 01:40:04,472 Stage-1 map = 50%, reduce = 0%, Cumulative CPU 1.91 sec
2019-04-30 01:40:05,510 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.21 sec
2019-04-30 01:40:17,900 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 3.58 sec
MapReduce Total cumulative CPU time: 3 seconds 580 msec
Ended Job = job_1556556784806_0006
MapReduce Jobs Launched:
Job 0: Map: 2 Reduce: 1 Cumulative CPU: 3.58 sec HDFS Read: 40907 HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 580 msec
OK
1000
Time taken: 40.941 seconds, Fetched: 1 row(s)
```

5) 创建 shop2_1 表，过滤掉 shop2 表中重复的记录（除 id 字段外，其他字段都相同的记录被认为重复记录；如果存在 2 条及 2 条以上的重复记录，则只保留 1 条记录），并将结果保存到 shop2_1 表中。查询 shop2_1 表中的记录总数并将结果截图保存到实验报告任务 2-1-5 中。（5 分）

```
<<create table shop2_1(id string,goods_name string,biz_prize string,count
```

string,biz_data string) row format delimited fields terminated by '\t' stored as textfile;

```
hive> create table shop2_1(id string,goods_name string,biz_prize string,count string,biz_data string) row format delimited fields terminated by '\t' stored as textfile;
OK
Time taken: 0.035 seconds
hive>
```

```
>> insert into table shop2_1

>>select shop2_s.id , shop2_f.goods_name , shop2_f.biz_prize,
>>shop2_f.count, shop2_f.biz_data  from
>>(select distinct goods_name, biz_prize,count, biz_data from shop2)
>>shop2_f join shop2 shop2_s on shop2_f.goods_name=shop2_s.goods_name
>>and shop2_f.biz_prize=shop2_s.biz_prize and shop2_f.count=shop2_s.count and
>>shop2_f.biz_data=shop2_s.biz_data;
```

```
hive> insert into table shop2_1 select shop2_s.id,shop2_f.goods_name,shop2_f.biz_prize,shop2_f.count,shop2_f.biz_data from (select distinct goods_name,biz_prize,count,biz_data from shop2)
> shop2_f join shop2 shop2_s on shop2_f.goods_name=shop2_s.goods_name
> and shop2_f.biz_prize=shop2_s.biz_prize and shop2_f.count=shop2_s.count and shop2_f.biz_data=shop2_s.biz_data
> ;
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1556556784806_0010, Tracking URL = http://master:18088/proxy/application_1556556784806_0010/
Kill Command = /home/hfut/hadoop-2.5.2/bin/hadoop job -kill job_1556556784806_0010
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2019-04-30 02:22:26,399 Stage-3 map = 0%, reduce = 0%
2019-04-30 02:22:33,634 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.11 sec
2019-04-30 02:22:41,907 Stage-3 map = 100%, reduce = 100%, Cumulative CPU 2.5 sec
MapReduce Total cumulative CPU time: 2 seconds 500 msec
Ended Job = job_1556556784806_0010
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hfut/hadoop-2.5.2/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hfut/hbase-1.1.2/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
19/04/30 02:22:46 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
19/04/30 02:22:46 WARN conf.Configuration: file:/tmp/hfut/hive 2019-04-30 02-22-18_191_5697143392928225902-1/-local-10008/jobconf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval; Ignoring.
19/04/30 02:22:46 WARN conf.Configuration: file:/tmp/hfut/hive 2019-04-30 02-22-18_191_5697143392928225902-1/-local-10008/jobconf.xml:an attempt to override
```

查看 shop2_1 总数:

```
>>select count(*) from shop2_1;
```

```
hive> select count(*) from shop2_1;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1556556784806_0012, Tracking URL = http://master:18088/proxy/application_1556556784806_0012/
Kill Command = /home/hfut/hadoop-2.5.2/bin/hadoop job -kill job_1556556784806_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2019-04-30 02:26:40,648 Stage-1 map = 0%, reduce = 0%
```

3. 数据分析(20 分)

“安徽省大数据学院”推出校园公众号以来受到广泛师生的关注，成为师生校园生活中必不可少的一部分；学校影视社为丰富师生课余生活开阔眼界和见识，为广大师生推送相关主题的优质影片信息。在请教了相关的老师以后，现在需要从影片的数据源进行影片的分析。在老师的推荐下影视社的成员找到你了，想请

你帮他们完成影片数据的分析。

“数据源”目录下有 top250_f1.txt 数据源，存放着部分影片信息。具体数据格式如下：

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
num	title	director	role	init_year	area	genre	rating_num	comment_num	comment	url				
1	肖申克的救赎	弗兰克·德拉邦特 Frank Darabont	弗兰克·德拉邦特 Frank Darabont	1994	美国	['犯罪', '剧情']	9.6	1151286	希望让人自由。	https://movie.douban.com/subject/1290962/				
2	霸王别姬	陈凯歌 Kaige Chen	张国荣 Leslie Cheung	1993	中国	['剧情', '爱情', '同性']	9.6	840607	风华绝代。	https://movie.douban.com/subject/1291546/				
3	这个杀手不太冷	吕克·贝松 Luc Besson	让·雷诺 Jean Reno	1994	法国	['剧情', '动作', '犯罪']	9.4	1063161	哈里森和萝莉不得不说的故事	https://movie.douban.com/subject/1295644/				
4	阿甘正传	Robert Zemeckis	Tom Hanks	1994	美国	['剧情', '爱情']	9.4	908943	一部美国近现代史。	https://movie.douban.com/subject/1292720/				
5	美丽人生	罗伯托·贝尼尼 Roberto Benigni	罗伯托·贝尼尼 Roberto Benigni	1997	意大利	['剧情', '喜剧', '爱情', '战争']	9.5	530038	最美的谎言。	https://movie.douban.com/subject/1292063/				
6	泰坦尼克号	詹姆斯·卡梅隆 James Cameron	詹姆斯·卡梅隆 James Cameron	1997	美国	['剧情', '爱情', '灾难']	9.3	844481	失去的才是永恒的。	https://movie.douban.com/subject/1292722/				
7	千与千寻	宫崎骏 Hayao Miyazaki	柊瑠美 Rumi Hiragi	2001	日本	['剧情', '动画', '奇幻']	9.3	842438	最好的宫崎骏，最好的久石让。	https://movie.douban.com/subject/1291561/				
8	辛德勒的名单	史蒂文·斯皮尔伯格 Steven Spielberg	连姆·尼森 Liam Neeson	1993	美国	['剧情', '历史', '战争']	9.4	478695	拯救一个人，就是拯救整个世界。	https://movie.douban.com/subject/1295124/				
9	盗梦空间	克里斯托弗·诺兰 Christopher Nolan	莱昂纳多·迪卡普里奥 Leonardo DiCaprio	2010	美国	['剧情', '科幻', '悬疑', '惊悚']	9.3	930867	诺兰给了我们一场无法忘却的梦。	https://movie.douban.com/subject/13541415/				
10	机器人总动员	安德鲁·斯坦顿 Andrew Stanton	本·巴特 Ben Burtt	2008	美国	['爱情', '科幻', '动画', '冒险']	9.3	615170	小瓦力，大人生。	https://movie.douban.com/subject/2131459/				
11	忠犬八公的故事	莱塞·霍尔斯道姆 Lasse Hallström	理查德·基尔 Richard Gere	2009	美国	['剧情']	9.3	595846	永远都不能忘记你最爱的人。	https://movie.douban.com/subject/3011091/				
12	三傻大闹宝莱坞	拉库马·希拉尼 Rakesh Joshi	阿米尔·汗 Amir Khan	2009	印度	['剧情', '喜剧', '爱情', '歌舞']	9.2	834758	英俊幽默豆，高情商高智商。	https://movie.douban.com/subject/3793023/				
13	海上钢琴师	朱塞佩·托纳多雷 Giuseppe Tornatore	比尔·罗斯 Bill Ross	1998	意大利	['剧情', '音乐']	9.2	701473	每个人都有一首自己谱写的歌。	https://movie.douban.com/subject/12920001/				
14	放牛班的春天	克里斯托夫·巴拉德 Christophe Barratier	朱诺·甘布诺 Juno Gamblin	2004	法国	['剧情', '音乐']	9.2	569769	天籁一般的童声，是最接近上帝的神迹。	https://movie.douban.com/subject/1291549/				
15	大话西游之大圣娶亲	刘镇伟 Jeffrey Lau	周星驰 Stephen Chow	1995	中国	['喜剧', '爱情', '奇幻', '冒险']	9.2	626297	一生所爱。	https://movie.douban.com/subject/1292213/				
16	楚门的世界	彼得·威尔 Peter Weir	金·凯瑞 Jim Carrey	1998	美国	['剧情', '科幻']	9.2	603163	如果再也不能见到你，祝你平安。	https://movie.douban.com/subject/1292064/				
17	教父	弗朗西斯·福特·科波拉 Francis Ford Coppola	阿尔·帕西诺 Al Pacino	1972	美国	['剧情', '犯罪']	9.2	422311	千万不要让你恨的人，这样会https://movie.douban.com/subject/1291941/					
18	龙猫	宫崎骏 Hayao Miyazaki	日高法子 Noriko Hidai	1988	日本	['动画', '奇幻', '冒险']	9.1	517992	人心中都有个龙猫，童年就是一场梦。	https://movie.douban.com/subject/1291560/				
19	星际穿越	克里斯托弗·诺兰 Christopher Nolan	马修·麦康纳 Matthew McConaughey	2014	美国	['剧情', '科幻', '冒险']	9.2	623731	爱是一种力量，让我们超越时空。	https://movie.douban.com/subject/1889243/				
20	烧炉	黄东赫 Dong-hyuk Hwang	孔侑 Yoo Gong	2011	韩国	['剧情']	9.2	348808	我们一路奋战不是为了改变世界，https://movie.douban.com/subject/5912992/					

其中，数据源属性为:num(影片序号)，title(电影名)，director(导演)，role(主演)，init_year(上映年份)，area(上映地区)，genre(电影类别)，rating_num(评分)，comment_num(评论数量)，comment(评论)，url(链接)

要求如下：

使用 Spark 平台完成以下题目，可使用 spark-sql、spark-shell 或者 java/python 调用 spark 库实现，截图时需保留 spark-sql、spark-shell 界面或者 java/python 代码以证实使用了 spark。

1) 统计评论量最多的 5 部电影，并将结果截图保存到实验报告任务 3-1-1 中。(5 分)

```
>>>val textfile = sc.textFile(file:///home/hfut/top_f1.txt)

//读入文件成为 RDD

scala> val textfile=sc.textFile("file:///home/hfut/top_f1.txt")
19/05/08 22:18:43 INFO storage.MemoryStore: ensureFreeSpace(33514) called with curMem=128886, maxMem=280248975
19/05/08 22:18:43 INFO storage.MemoryStore: Block broadcast 46 stored as values in memory (estimated size 32.7 KB, free 267.1 MB)
19/05/08 22:18:43 INFO storage.MemoryStore: ensureFreeSpace(5069) called with curMem=162489, maxMem=280248975
19/05/08 22:18:43 INFO storage.MemoryStore: Block broadcast 46 piece0 stored as bytes in memory (estimated size 5.0 KB, free 267.1 MB)
19/05/08 22:18:43 INFO storage.BlockManagerInfo: Added broadcast 46 piece0 in memory on localhost:49863 (size: 5.0 KB, free: 267.2 MB)
19/05/08 22:18:43 INFO storage.BlockManagerMaster: Updated info of block broadcast 46 piece0
19/05/08 22:18:43 INFO spark.SparkContext: Created broadcast 46 from textFile at <console>:12
textfile: org.apache.spark.rdd.RDD[String] = file:///home/hfut/top_f1.txt MappedRDD[53] at textFile at <console>:12

>>>textfile.map(line=>(line.split("\t")(8),line.split("\t")(1))).sortByKey(false).take(5)

).mkString("\t")

//生成（电影名，评论数）的 key，value 对按照 value 值倒序排序，取最大的 5
//部输出
```


从上图可以看到结果为：

1. 《步履不停》 99994 条
 2. 《千钧一发》 97525 条
 3. 《地球上的星星》 96742 条
 4. 《惊魂记》 95862 条
 5. 《勇士》 94905 条
- 2) 统计 2013 年以后电影最多的 5 个地区，并将结果截图保存到实验报告任务 3-1-2 中。（5 分）

```
>> val textfile = sc.textFile(file:///home/hfut/top_f1.txt)

//读入文件成为 RDD

//file 指示从本地读取，类似的有 HDFS 表示从 HDFS 读取文件

>> textfile.filter(line=>line.split("\t")(4).toInt>2013).map(line=>(line.split("\t")(5),1)).reduceByKey((a,b)=>a+b).sortBy(_._2,false).take(5).mkString("\t")

//使用 filter 筛除 2013 年即以前的电影，使用 map 获得（地区，1）的 key，

//value 对然后 reduceByKey 对相同 key 的 key，value 相加，按照 value 大小倒

//序排列，取最大的五个输出
```

```
hfut@master:~/app/spark-1.2.0/sbin

scala> textfile.filter(line=>line.split("\t")(4).toInt>2013).map(line=>(line.split("\t")(5),1)).reduceByKey((a,b)=>a+b).sortBy(_._2,false).collect
19/05/09 08:39:46 INFO spark.SparkContext: Starting job: collect at <console>:15
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Registering RDD 11 (map at <console>:15)
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Registering RDD 13 (sortBy at <console>:15)
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Got job 2 (collect at <console>:15) with 1 output partitions (allowLocal=false)
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Final stage: Stage 5(collect at <console>:15)
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Parents of final stage: List(Stage 4)
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Missing parents: List(Stage 4)
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Submitting Stage 3 (MappedRDD[11] at map at <console>:15), which has no missing parents
19/05/09 08:39:46 INFO spark.MemoryStore: ensureFreeSpace(3576) called with curMem=48281, maxMem=280248975
19/05/09 08:39:46 INFO storage.MemoryStore: Block broadcast 4 stored as values in memory (estimated size 3.5 KB, free 267.2 MB)
19/05/09 08:39:46 INFO storage.MemoryStore: ensureFreeSpace(2540) called with curMem=51857, maxMem=280248975
19/05/09 08:39:46 INFO storage.MemoryStore: Block broadcast 4 piece0 stored as bytes in memory (estimated size 2.5 KB, free 267.2 MB)
19/05/09 08:39:46 INFO storage.BlockManagerInfo: Added broadcast 4 piece0 in memory on localhost:53609 (size: 2.5 KB, free: 267.3 MB)
19/05/09 08:39:46 INFO storage.BlockManagerMaster: Updated info of block broadcast 4 piece0
19/05/09 08:39:46 INFO spark.SparkContext: Created broadcast 4 from broadcast at DAGScheduler.scala:838
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from Stage 3 (MappedRDD[11] at map at <console>:15)
19/05/09 08:39:46 INFO scheduler.TaskSchedulerImpl: Adding task set 3.0 with 1 tasks
19/05/09 08:39:46 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 3.0 (TID 3, localhost, PROCESS_LOCAL, 1283 bytes)
19/05/09 08:39:46 INFO executor.Executor: Running task 0.0 in stage 3.0 (TID 3)
19/05/09 08:39:46 INFO rdd.HadoopRDD: Input split: file:/home/hfut/top250 fl.txt:0+52719
19/05/09 08:39:46 INFO executor.Executor: Finished task 0.0 in stage 3.0 (TID 3). 1895 bytes result sent to driver
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Stage 3 (map at <console>:15) finished in 0.058 s
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Looking for newly runnable stages
19/05/09 08:39:46 INFO scheduler.DAGScheduler: running: Set()
19/05/09 08:39:46 INFO scheduler.DAGScheduler: waiting: Set(Stage 5, Stage 4)
19/05/09 08:39:46 INFO scheduler.DAGScheduler: failed: Set()
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Missing parents for Stage 5: List(Stage 4)
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Missing parents for Stage 4: List()
19/05/09 08:39:46 INFO scheduler.DAGScheduler: Submitting Stage 4 (MappedRDD[13] at sortBy at <console>:15), which is now runnable
19/05/09 08:39:46 INFO storage.MemoryStore: ensureFreeSpace(3072) called with curMem=54397, maxMem=280248975
19/05/09 08:39:46 INFO storage.MemoryStore: Block broadcast 5 stored as values in memory (estimated size 3.0 KB, free 267.2 MB)
19/05/09 08:39:46 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 3.0 (TID 3) in 58 ms on localhost (1/1)
19/05/09 08:39:46 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
19/05/09 08:39:46 INFO storage.MemoryStore: ensureFreeSpace(2193) called with curMem=57469, maxMem=280248975
19/05/09 08:39:46 INFO storage.MemoryStore: Block broadcast 5 piece0 stored as bytes in memory (estimated size 2.1 KB, free 267.2 MB)
```

```
hfut@master:~/app/spark-1.2.0/sbin

19/05/09 08:39:46 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from Stage 4 (MappedRDD[13] at sortBy at <console>:15)
19/05/09 08:39:46 INFO scheduler.TaskSchedulerImpl: Adding task set 4.0 with 1 tasks
19/05/09 08:39:46 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 4.0 (TID 4, localhost, PROCESS_LOCAL, 1045 bytes)
19/05/09 08:39:46 INFO executor.Executor: Running task 0.0 in stage 4.0 (TID 4)
19/05/09 08:39:46 INFO storage.ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
19/05/09 08:39:46 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 1 ms
19/05/09 08:39:47 INFO executor.Executor: Finished task 0.0 in stage 4.0 (TID 4). 1000 bytes result sent to driver
19/05/09 08:39:47 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 4.0 (TID 4) in 123 ms on localhost (1/1)
19/05/09 08:39:47 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 4.0, whose tasks have all completed, from pool
19/05/09 08:39:47 INFO scheduler.DAGScheduler: Stage 4 (sortBy at <console>:15) finished in 0.118 s
19/05/09 08:39:47 INFO scheduler.DAGScheduler: Looking for newly runnable stages
19/05/09 08:39:47 INFO scheduler.DAGScheduler: running: Set()
19/05/09 08:39:47 INFO scheduler.DAGScheduler: waiting: Set(Stage 5)
19/05/09 08:39:47 INFO scheduler.DAGScheduler: failed: Set()
19/05/09 08:39:47 INFO scheduler.DAGScheduler: Missing parents for Stage 5: List()
19/05/09 08:39:47 INFO scheduler.DAGScheduler: Submitting Stage 5 (MappedRDD[15] at sortBy at <console>:15), which is now runnable
19/05/09 08:39:47 INFO storage.MemoryStore: ensureFreeSpace(2776) called with curMem=59662, maxMem=280248975
19/05/09 08:39:47 INFO storage.MemoryStore: Block broadcast 6 stored as values in memory (estimated size 2.7 KB, free 267.2 MB)
19/05/09 08:39:47 INFO storage.MemoryStore: ensureFreeSpace(2026) called with curMem=62438, maxMem=280248975
19/05/09 08:39:47 INFO storage.MemoryStore: Block broadcast 6 piece0 stored as bytes in memory (estimated size 2026.0 B, free: 267.3 MB)
19/05/09 08:39:47 INFO storage.BlockManagerInfo: Added broadcast 6 piece0 in memory on localhost:53609 (size: 2026.0 B, free: 267.3 MB)
19/05/09 08:39:47 INFO storage.BlockManagerMaster: Updated info of block broadcast 6 piece0
19/05/09 08:39:47 INFO spark.SparkContext: Created broadcast 6 from broadcast at DAGScheduler.scala:838
19/05/09 08:39:47 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from Stage 5 (MappedRDD[15] at sortBy at <console>:15)
19/05/09 08:39:47 INFO scheduler.TaskSchedulerImpl: Adding task set 5.0 with 1 tasks
19/05/09 08:39:47 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 5.0 (TID 5, localhost, PROCESS_LOCAL, 1056 bytes)
19/05/09 08:39:47 INFO executor.Executor: Running task 0.0 in stage 5.0 (TID 5)
19/05/09 08:39:47 INFO storage.ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
19/05/09 08:39:47 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
19/05/09 08:39:47 INFO executor.Executor: Finished task 0.0 in stage 5.0 (TID 5). 1181 bytes result sent to driver
19/05/09 08:39:47 INFO scheduler.DAGScheduler: Stage 5 (collect at <console>:15) finished in 0.015 s
19/05/09 08:39:47 INFO scheduler.DAGScheduler: Job 2 finished: collect at <console>:15, took 0.246130 s
res0: Array[(String, Int)] = Array((美国,10), (日本,4), (印度,2), (中国,2), (意大利,2), (爱尔兰,1), (阿根廷,1), (西班牙,1))

scala> 19/05/09 08:39:47 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 5.0 (TID 5) in 15 ms on localhost (1/1)
19/05/09 08:39:47 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 5.0, whose tasks have all completed, from pool
```

```
19/05/09 08:43:12 INFO storage.BlockManagerMaster: Updated info of block broadcast 9 piece0
19/05/09 08:43:12 INFO spark.SparkContext: Created broadcast 9 from broadcast at DAGScheduler.scala:838
19/05/09 08:43:12 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from Stage 8 (MappedRDD[21] at sortBy at <console>:15)
19/05/09 08:43:12 INFO scheduler.TaskSchedulerImpl: Adding task set 8.0 with 1 tasks
19/05/09 08:43:12 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 8.0 (TID 8, localhost, PROCESS_LOCAL, 1056 bytes)
19/05/09 08:43:12 INFO executor.Executor: Running task 0.0 in stage 8.0 (TID 8)
19/05/09 08:43:12 INFO storage.ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
19/05/09 08:43:12 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
19/05/09 08:43:12 INFO executor.Executor: Finished task 0.0 in stage 8.0 (TID 8). 1067 bytes result sent to driver
19/05/09 08:43:12 INFO scheduler.DAGScheduler: Stage 8 (take at <console>:15) finished in 0.008 s
19/05/09 08:43:12 INFO scheduler.DAGScheduler: Job 3 finished: take at <console>:15, took 0.186133 s
res11: String = (美国,10) (日本,4) (印度,2) (中国,2) (意大利,2)
```

最终结果：

1. 美国 10 部
2. 日本 4 部
3. 印度 2 部
4. 中国 2 部

5. 意大利 2 部

3) 电影排行榜一般是按分数来排列的, 但如果相同分又怎样区分谁更好看, 或者说某一部电影只有少部分人看但评分都是高分, 那它不就可以登上排行榜前列。所以现在创建一个总体热度系数 (影片评论数量乘以评分), 再用这个系数对影片作一个排序, 然后找出电影类型中同时包含 “剧情” 和 “爱情” 的, 系数最大的前十部, 并将结果截图保存到实验报告任务 3-1-3 中。(10 分)

使用的命令如下:

```
>> val textfile = sc.textFile(file:///home/hfut/top_f1.txt)

//读入文件

>> val allNum = textfile.filter(line=>line.contains("剧情"))

.filter(line=>line.contains("爱情"))

.map(line=>(line.split("\t")(1),(line.split("\t")(7).toDouble)

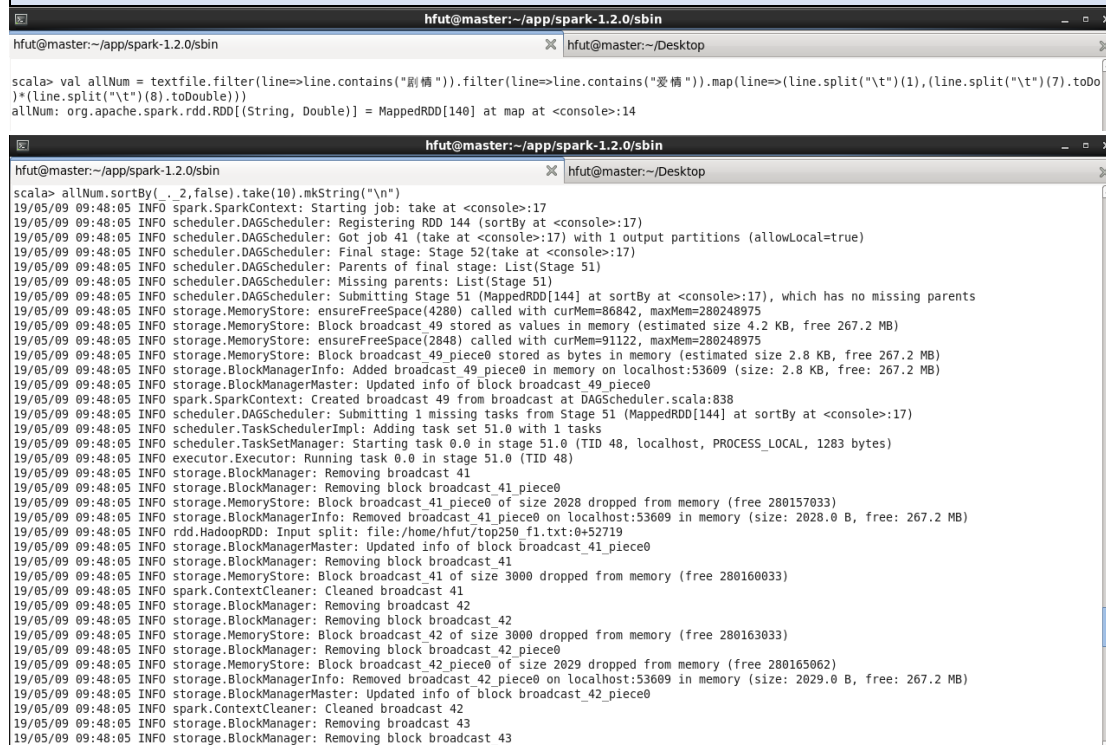
*(line.split("\t")(8).toDouble)))

//每一行一次为筛除不包含 “爱情”, “剧情” 的行, 使用 “ ” 切分获得电影名、

//以及评论数和评论进行相乘

>> allNum.sortBy(_._2,false).take(10).mkString("\n")

//使用 value 值进行倒序排序取最大的 10 个
```



The first screenshot shows the Spark shell interface with the command `scala> val allNum = textfile.filter(line=>line.contains("剧情")).filter(line=>line.contains("爱情")).map(line=>(line.split("\t")(1),(line.split("\t")(7).toDouble)*(line.split("\t")(8).toDouble)))` and the output `allNum: org.apache.spark.rdd.RDD[(String, Double)] = MappedRDD[140] at map at <console>:14`.

The second screenshot shows the continuation of the command `allNum.sortBy(_._2,false).take(10).mkString("\n")` and its output, which is a list of 10 movie titles and their corresponding coefficients, sorted in descending order. The output is as follows:

```
scala> val allNum = textfile.filter(line=>line.contains("剧情")).filter(line=>line.contains("爱情")).map(line=>(line.split("\t")(1),(line.split("\t")(7).toDouble)*(line.split("\t")(8).toDouble)))
allNum: org.apache.spark.rdd.RDD[(String, Double)] = MappedRDD[140] at map at <console>:14

scala> allNum.sortBy(_._2,false).take(10).mkString("\n")
19/05/09 09:48:05 INFO spark.SparkContext: Starting job: take at <console>:17
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Registering RDD 144 (sortBy at <console>:17)
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Got job 41 (take at <console>:17) with 1 output partitions (allowLocal=true)
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Final stage: Stage 52 (take at <console>:17)
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Parents of final stage: List(Stage 51)
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Missing parents: List(Stage 51)
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Submitting Stage 51 (MappedRDD[144] at sortBy at <console>:17), which has no missing parents
19/05/09 09:48:05 INFO storage.MemoryStore: ensureFreeSpace(4280) called with curMem=86842, maxMem=280248975
19/05/09 09:48:05 INFO storage.MemoryStore: Block broadcast 49 stored as values in memory (estimated size 4.2 KB, free 267.2 MB)
19/05/09 09:48:05 INFO storage.MemoryStore: ensureFreeSpace(2848) called with curMem=91122, maxMem=280248975
19/05/09 09:48:05 INFO storage.MemoryStore: Block broadcast 49 piece0 stored as bytes in memory (estimated size 2.8 KB, free 267.2 MB)
19/05/09 09:48:05 INFO storage.BlockManagerInfo: Added broadcast 49 piece0 in memory on localhost:53609 (size: 2.8 KB, free: 267.2 MB)
19/05/09 09:48:05 INFO storage.BlockManagerMaster: Updated info of block broadcast 49 piece0
19/05/09 09:48:05 INFO spark.SparkContext: Created broadcast 49 from broadcast at DAGScheduler.scala:838
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from Stage 51 (MappedRDD[144] at sortBy at <console>:17)
19/05/09 09:48:05 INFO scheduler.TaskSchedulerImpl: Adding task set 51.0 with 1 tasks
19/05/09 09:48:05 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 51.0 (TID 48, localhost, PROCESS_LOCAL, 1283 bytes)
19/05/09 09:48:05 INFO executor.Executor: Running task 0.0 in stage 51.0 (TID 48)
19/05/09 09:48:05 INFO storage.BlockManager: Removing broadcast 41
19/05/09 09:48:05 INFO storage.BlockManager: Removing block broadcast 41 piece0
19/05/09 09:48:05 INFO storage.MemoryStore: Block broadcast 41 piece0 of size 2028 dropped from memory (free 280157033)
19/05/09 09:48:05 INFO storage.BlockManagerInfo: Removed broadcast 41 piece0 on localhost:53609 in memory (size: 2028.0 B, free: 267.2 MB)
19/05/09 09:48:05 INFO rdd.HadoopRDD: Input split: file:/home/hfut/top250_f1.txt:8+52719
19/05/09 09:48:05 INFO storage.BlockManagerMaster: Updated info of block broadcast 41 piece0
19/05/09 09:48:05 INFO storage.MemoryStore: Block broadcast 41 of size 3000 dropped from memory (free 280160033)
19/05/09 09:48:05 INFO spark.ContextCleaner: Cleaned broadcast 41
19/05/09 09:48:05 INFO storage.BlockManager: Removing broadcast 42
19/05/09 09:48:05 INFO storage.BlockManager: Removing block broadcast 42
19/05/09 09:48:05 INFO storage.MemoryStore: Block broadcast 42 of size 3000 dropped from memory (free 280163033)
19/05/09 09:48:05 INFO storage.BlockManager: Removing block broadcast 42 piece0
19/05/09 09:48:05 INFO storage.MemoryStore: Block broadcast 42 piece0 of size 2029 dropped from memory (free 280165062)
19/05/09 09:48:05 INFO storage.BlockManagerInfo: Removed broadcast 42 piece0 on localhost:53609 in memory (size: 2029.0 B, free: 267.2 MB)
19/05/09 09:48:05 INFO storage.BlockManagerMaster: Updated info of block broadcast 42 piece0
19/05/09 09:48:05 INFO spark.ContextCleaner: Cleaned broadcast 42
19/05/09 09:48:05 INFO storage.BlockManager: Removing broadcast 43
19/05/09 09:48:05 INFO storage.BlockManager: Removing block broadcast 43
```

```
hfut@master:~/app/spark-1.2.0/sbin
hfut@master:~/app/spark-1.2.0/sbin
19/05/09 09:48:05 INFO scheduler.DAGScheduler: running: Set()
19/05/09 09:48:05 INFO scheduler.DAGScheduler: waiting: Set(Stage 52)
19/05/09 09:48:05 INFO scheduler.DAGScheduler: failed: Set()
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Missing parents for Stage 52: List()
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Submitting Stage 52 (MappedRDD[146] at sortBy at <console>:17), which is now runnable
19/05/09 09:48:05 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 51.0 (TID 48) in 38 ms on localhost (1/1)
19/05/09 09:48:05 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 51.0, whose tasks have all completed, from pool
19/05/09 09:48:05 INFO storage.MemoryStore: ensureFreeSpace(2848) called with curMem=51753, maxMem=280248975
19/05/09 09:48:05 INFO storage.MemoryStore: Block broadcast_50 stored as values in memory (estimated size 2.0 KB, free 267.2 MB)
19/05/09 09:48:05 INFO storage.MemoryStore: ensureFreeSpace(2874) called with curMem=54601, maxMem=280248975
19/05/09 09:48:05 INFO storage.MemoryStore: Block broadcast_50 piece0 stored as bytes in memory (estimated size 2.0 KB, free 267.2 MB)
19/05/09 09:48:05 INFO storage.BlockManagerInfo: Added broadcast_50 piece0 in memory on localhost:53609 (size: 2.0 KB, free: 267.3 MB)
19/05/09 09:48:05 INFO storage.BlockManagerMaster: Updated info of block broadcast_50 piece0
19/05/09 09:48:05 INFO spark.SparkContext: Created broadcast 50 from broadcast at DAGScheduler.scala:838
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from Stage 52 (MappedRDD[146] at sortBy at <console>:17)
19/05/09 09:48:05 INFO scheduler.TaskSchedulerImpl: Adding task set 52.0 with 1 tasks
19/05/09 09:48:05 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 52.0 (TID 49, localhost, PROCESS_LOCAL, 1056 bytes)
19/05/09 09:48:05 INFO executor.Executor: Running task 0.0 in stage 52.0 (TID 49)
19/05/09 09:48:05 INFO storage.ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 1 blocks
19/05/09 09:48:05 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
19/05/09 09:48:05 INFO executor.Executor: Finished task 0.0 in stage 52.0 (TID 49). 1325 bytes result sent to driver
19/05/09 09:48:05 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 52.0 (TID 49) in 58 ms on localhost (1/1)
19/05/09 09:48:05 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 52.0, whose tasks have all completed, from pool
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Stage 52 (take at <console>:17) finished in 0.056 s
19/05/09 09:48:05 INFO scheduler.DAGScheduler: Job 41 finished: take at <console>:17, took 0.160518 s
res91: String =
(阿甘正传,8543124.200000001)
(霸王别姬,8069827.199999999)
(泰坦尼克号,7853673.300000001)
(三傻大闹宝莱坞,7679773.6)
(怦然心动,6447915.0)
(美丽人生,5035361.0)
(剪刀手爱德华,4888791.0)
(你的名字.,4735626.0)
(初恋这件小事,4279978.0)
(罗马假日,4115021.8000000003)
```

总体热度系数前十结果：

电影名称	总体热度系数
阿甘正传	8543124.200000001
霸王别姬	8069827.199999999
泰坦尼克号	7853673.300000001
三傻大闹宝莱坞	7679773.6
怦然心动	6447915.0
美丽人生	5035361.0
剪刀手爱德华	4888791.0
你的名字	4735626.0
初恋这件小事	4279978.0
罗马假日	4115021.8000000003

4. 数据可视化(20 分)

“安徽省大数据学院”跟学校进行校企合作，提供了一些气象的数据源给学校学生进行数据的可视化，想请你帮他们完成数据的可视化。

“数据源”目录下有 weather.txt 数据源，存放部分气象信息，如下图所示：

日期	天气状况	最高气温℃	最低气温℃	风力	风向
2016/1/1	晴	22	12	无持续	风向 <3级
2016/1/2	多云	22	16	无持续	风向 <3级
2016/1/3	阴	25	18	无持续	风向 <3级
2016/1/4	小雨	24	18	无持续	风向 <3级
2016/1/5	中雨	22	16	无持续	风向 <3级
2016/1/6	多云	23	12	无持续	风向 <3级
2016/1/7	多云	20	14	无持续	风向 <3级
2016/1/8	多云	21	14	无持续	风向 <3级
2016/1/9	多云	20	14	无持续	风向 <3级
2016/1/10	中到大雨		18	14	无持续风向 <3级
2016/1/11	小雨	17	11	无持续	风向 <3级
2016/1/12	阴	17	11	无持续	风向 <3级
2016/1/13	多云	18	10	无持续	风向 <3级
2016/1/14	多云	18	12	无持续	风向 <3级
2016/1/15	小到中雨		14	12	无持续风向 <3级
2016/1/16	小雨	15	12	无持续	风向 <3级
2016/1/17	小雨	17	10	无持续	风向 <3级
2016/1/18	晴	20	10	无持续	风向 <3级
2016/1/19	多云	17	12	无持续	风向 <3级
2016/1/20	小雨	15	12	无持续	风向 <3级
2016/1/21	小到中雨		14	9	无持续风向 <3级
2016/1/22	小到中雨		12	4	北风 3-4级
2016/1/23	阴	6	2	东北风	4-5级
2016/1/24	小雨	6	0	东北风	3-4级
2016/1/25	多云	10	2	无持续	风向 <3级
2016/1/26	小雨	12	6	无持续	风向 <3级
2016/1/27	中到大雨		10	9	无持续风向 <3级
2016/1/28	大到暴雨		14	13	无持续风向 <3级
2016/1/29	中雨	18	12	无持续	风向 <3级
2016/1/30	多云	19	11	无持续	风向 <3级
2016/1/31	小雨	16	8	无持续	风向 <3级
2016/2/1	中雨	11	7	无持续	风向 <3级
2016/2/2	小雨	10	6	无持续	风向 <3级

要求如下：

1) 通过折线图展示 2018 年 9 月份天每天最高温度变化图和最低温度变化图 。
 并将 2018 年 9 月份每天最高温度和最低温度查询结果和折线图展示结果截图保存到实验报告任务 4-1-1 中。(10 分)

使用的 sql 语句如下：依次为创建数据库，指定编码；

```
Mysql>>CREATE DATABASE `wea`
->CHARACTER SET 'utf8'
->COLLATE 'utf8_general_ci';
```

```
Database changed
mysql> CREATE DATABASE `wea`
-> CHARACTER SET 'utf8'
-> COLLATE 'utf8_general_ci';
Query OK, 1 row affected (0.00 sec)

mysql> █
```

指定当前使用的数据库：

```
mysql>>use wea;
```

创建 weather 表：

```
CREATE TABLE `weather` (
```

```
`data` date DEFAULT NULL,  
`cloud` varchar(11) DEFAULT NULL,  
`high` int(3) DEFAULT NULL,  
`low` int(3) DEFAULT NULL,  
`wind` varchar(20) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

从本地加载数据到 weather 表中：

```
Mysql>>load data local infile "/home/hfut/weather.txt" into table weather fields  
terminated by 't';
```

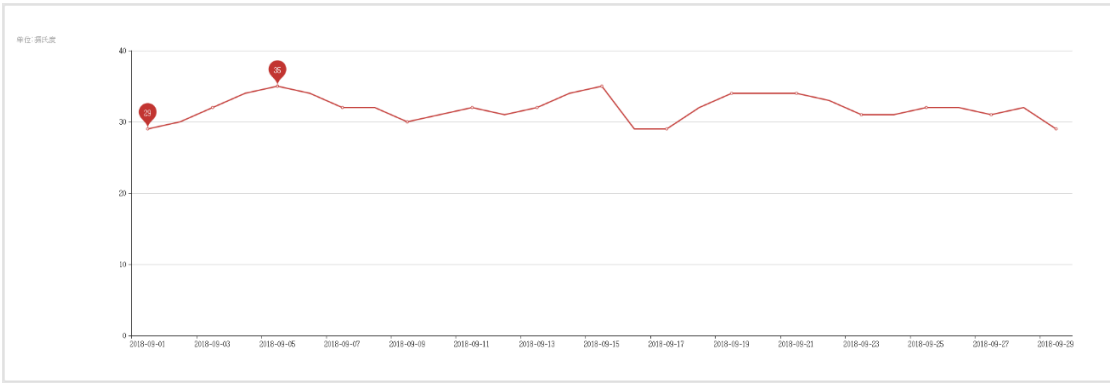
使用查询语句查询表的结构（篇幅有限仅展示部分）：

-19	多云	34	25	无持续风向1-2级
-20	多云	34	25	无持续风向1-2级
-21	雷阵雨	34	25	无持续风向1-2级
-22	多云	33	26	无持续风向1-2级
-23	雷阵雨	31	25	无持续风向1-2级
-24	雷阵雨	31	25	无持续风向1-2级
-25	雷阵雨	32	25	无持续风向1-2级
-26	雷阵雨	32	25	无持续风向1-2级
-27	多云	31	24	无持续风向1-2级
	晴	32	24	北风3-4级
	晴	29	22	北风3-4级

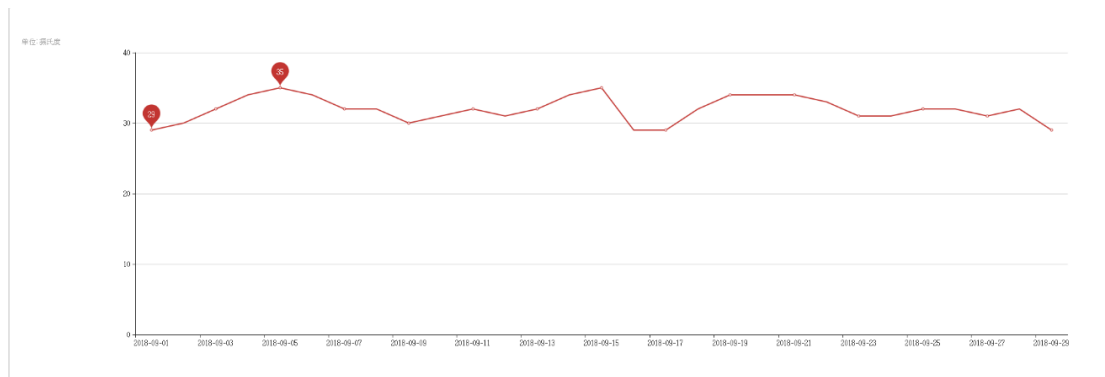
1003 rows in set (0.01 sec)

```
String sql = "select * from weather where data between '2018-9-1' and '2018-9-30'  
order by data asc";
```

最高气温折线图：（图中的这两个红点分别表示最高气温中的最高气温和最低气温，可以用来标志数据，判断数据是否正确）。



最低气温折线图：（图中的这两个红点分别表示最高气温中的最低气温和最低气温，可以用来标志数据，判断数据是否正确）。



使用 mysql 查询在 2018 年 9 月 1 日到 2018 年 10 月 1 日之间的最高和最低气温情况验证上方的折线图。

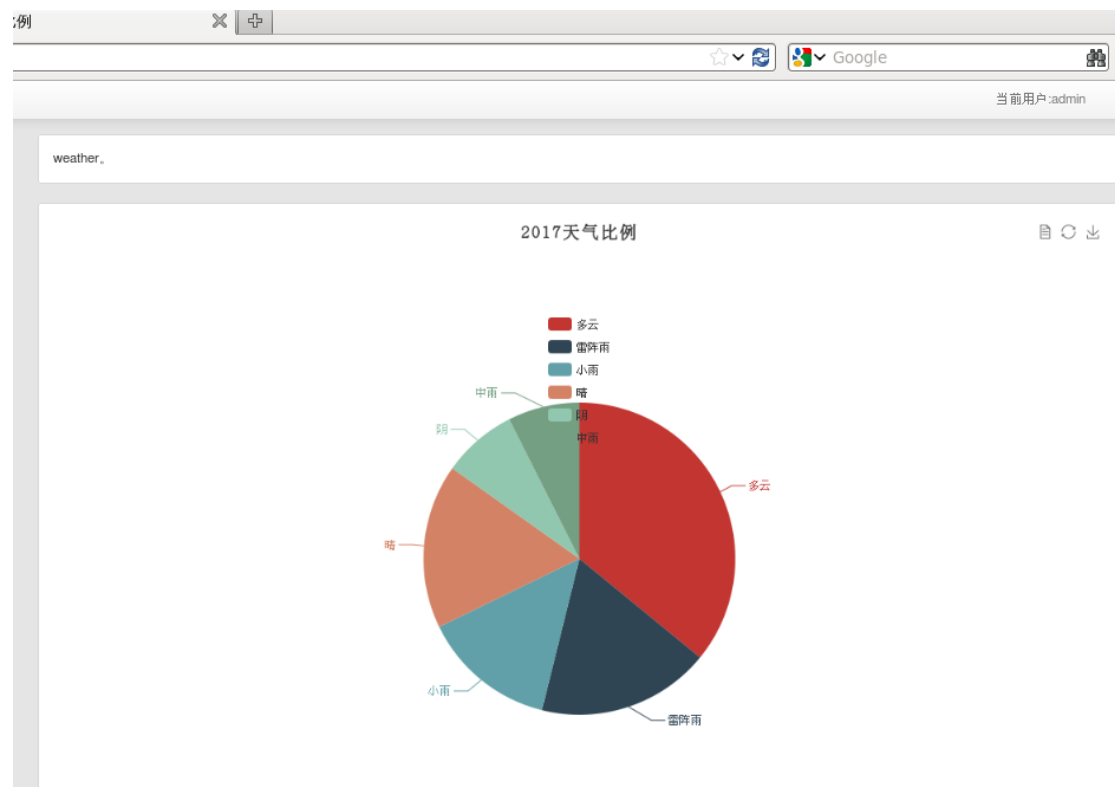
```
>> select data,high,low from weather where data between '2018-9-1' and '2018-9-30';
```

```
mysql> select data,high,low from weather where data between '2018-9-1' and '2018-10-1';
+-----+-----+-----+
| data | high | low |
+-----+-----+-----+
| 2018-09-01 | 29 | 24 |
| 2018-09-02 | 30 | 24 |
| 2018-09-03 | 32 | 26 |
| 2018-09-04 | 34 | 26 |
| 2018-09-05 | 35 | 26 |
| 2018-09-06 | 34 | 25 |
| 2018-09-07 | 32 | 24 |
| 2018-09-08 | 32 | 23 |
| 2018-09-09 | 30 | 24 |
| 2018-09-10 | 31 | 24 |
| 2018-09-11 | 32 | 24 |
| 2018-09-12 | 31 | 25 |
| 2018-09-13 | 32 | 25 |
| 2018-09-14 | 34 | 26 |
| 2018-09-15 | 35 | 25 |
| 2018-09-16 | 29 | 24 |
| 2018-09-17 | 29 | 25 |
| 2018-09-18 | 32 | 26 |
| 2018-09-19 | 34 | 25 |
| 2018-09-20 | 34 | 25 |
| 2018-09-21 | 34 | 25 |
| 2018-09-22 | 33 | 26 |
| 2018-09-23 | 31 | 25 |
| 2018-09-24 | 31 | 25 |
| 2018-09-25 | 32 | 25 |
| 2018-09-26 | 32 | 25 |
| 2018-09-27 | 31 | 24 |
| 2018-09-28 | 32 | 24 |
| 2018-09-29 | 29 | 22 |
+-----+-----+-----+
29 rows in set (0.00 sec)
```

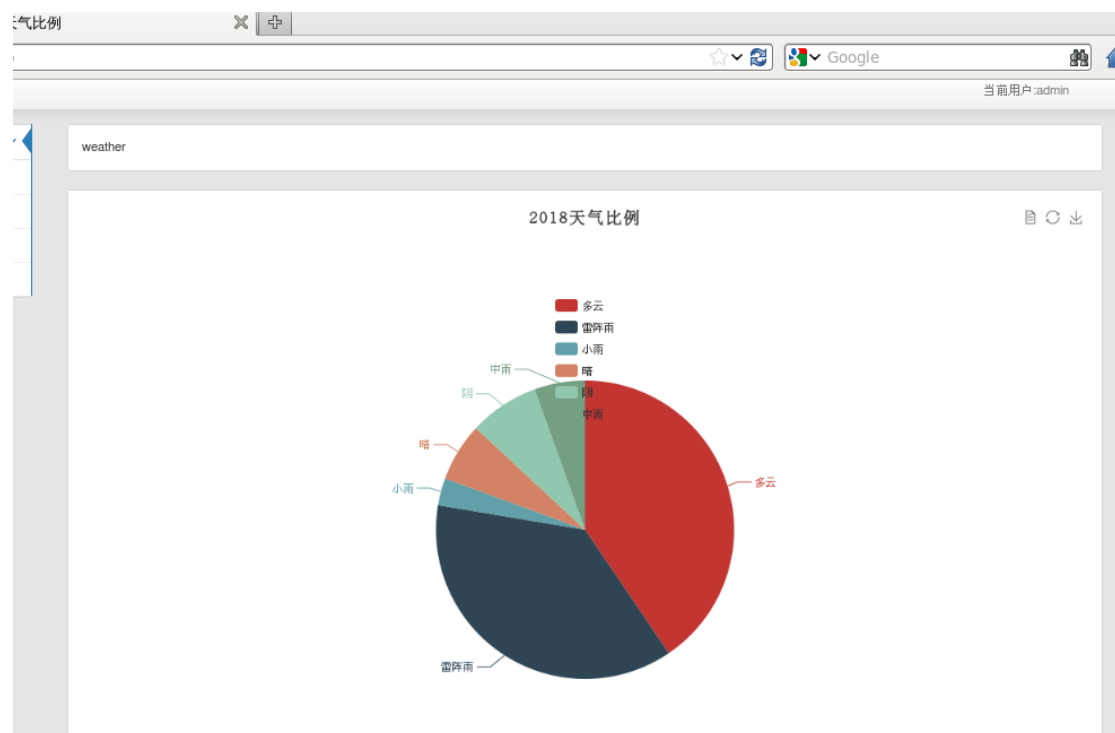
通过 sql 语句查询的结果可以看到折线图的结果与数据相符合。

2) 通过雷达图展示 2017 年和 2018 年天气概况信息(雷达图中所统计的 6 种天气概况分别为:多云,雷阵雨,小雨,晴,阴,中雨;这 6 种天气概况最大极限值为:200)。并将 2017 年和 2018 年天气概况信息查询结果和雷达图展示结果截图保存到实验报告任务 4-1-2 中。(10 分)

2017 年各种天气所占的比例如下图所示:(阴天的标志由于显示的原因在图中和饼图重叠)



2018 年各种天气情况所占的比例如下图所示：



使用 sql 语句查询 2017 年 1 月 1 日到 2018 年 1 月 1 日之间以及 2018 年 1 月 1 日到 2019 年 1 月 1 日之间的各种天气的天数情况与饼图数据对比：

```
//2017-2018 年的各种天气查询
select count(*)  from weather where data between '2017-1-1' and '2018-1-1' and
```

```
cloud='小雨';
select count(*) from weather where data between '2017-1-1' and '2018-1-1' and
cloud='中雨';
select count(*) from weather where data between '2017-1-1' and '2018-1-1' and
cloud='雷震雨';
select count(*) from weather where data between '2017-1-1' and '2018-1-1' and
cloud='晴';
select count(*) from weather where data between '2017-1-1' and '2018-1-1' and
cloud='多云';
select count(*) from weather where data between '2017-1-1' and '2018-1-1' and
cloud='阴';

//2018-2019 年的各种天气查询
select count(*) from weather where data between '2018-1-1' and '2019-1-1' and
cloud='小雨';
select count(*) from weather where data between '2018-1-1' and '2019-1-1' and
cloud='中雨';
select count(*) from weather where data between '2018-1-1' and '2019-1-1' and
cloud='雷阵雨';
select count(*) from weather where data between '2018-1-1' and '2019-1-1' and
cloud='晴';
select count(*) from weather where data between '2018-1-1' and '2019-1-1' and
cloud='阴';
select count(*) from weather where data between '2018-1-1' and '2019-1-1' and
cloud='多云';
```

各个年段的各种天气查询结果如下：

```

mysql> select count(*) from weather where data between '2017-1-1' and '2018-1-1' and cloud='小雨';
+-----+
| count(*) |
+-----+
|      45 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from weather where data between '2017-1-1' and '2018-1-1' and cloud='中雨';
+-----+
| count(*) |
+-----+
|      24 |
+-----+
1 row in set (0.01 sec)

mysql> select count(*) from weather where data between '2017-1-1' and '2018-1-1' and cloud='雷暴雨';
+-----+
| count(*) |
+-----+
|        0 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from weather where data between '2017-1-1' and '2018-1-1' and cloud='晴';
+-----+
| count(*) |
+-----+
|      55 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from weather where data between '2017-1-1' and '2018-1-1' and cloud='多云';
+-----+
| count(*) |
+-----+
|     116 |
+-----+

mysql> select count(*) from weather where data between '2017-1-1' and '2018-1-1' and cloud='阴';
+-----+
| count(*) |
+-----+
|      25 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from weather where data between '2018-1-1' and '2019-1-1' and cloud='小雨';
+-----+
| count(*) |
+-----+
|        7 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from weather where data between '2018-1-1' and '2019-1-1' and cloud='中雨';
+-----+
| count(*) |
+-----+
|       13 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from weather where data between '2018-1-1' and '2019-1-1' and cloud='雷阵雨';
+-----+
| count(*) |
+-----+
|       88 |
+-----+
1 row in set (0.01 sec)

mysql> select count(*) from weather where data between '2018-1-1' and '2019-1-1' and cloud='晴';
+-----+
| count(*) |
+-----+
|       15 |
+-----+

1 row in set (0.00 sec)

mysql> select count(*) from weather where data between '2018-1-1' and '2019-1-1' and cloud='阴';
+-----+
| count(*) |
+-----+
|       18 |
+-----+
1 row in set (0.00 sec)

mysql> select count(*) from weather where data between '2018-1-1' and '2019-1-1' and cloud='多云';
+-----+
| count(*) |
+-----+
|       96 |
+-----+
1 row in set (0.00 sec)

```

天气比例 - Mozilla Firefox

通过上述数据，发现饼图的每一个部分和数据库的数据相一致，符合题目要求。

5. 总结

在本次课程综合设计中我收获颇多，包括从单 slave 扩展到 2 个 slave 的 hadoop、hive 上去，再有使用了更多、更复杂的 hive 的 sql 语句，基本上简单的操作都有所了解，还有就是又一次的使用了可视化工具，这是再课程实验的基础上的自己尝试，虽然过程中也出现了不少的 bug，但是当第一个图出现的时候

还是很开心的，最后还就是自己装了一遍 spark，虽然过程有些坎坷，但是装完之后感觉也就没什么了，然后又用了不是很长的时间学习了下 scala 以及 spark-shell 下的 spark 编程，跑出来的第一条有意义的语句也同时意味着我对 spark 有了更深的认识与了解，虽然说这个了解还是比较基础的，但是一定会更加深入的去了解。自学 spark 可能在一定程度上也意味着本门课程的成功，因为怎么说也是自学了一门新技术。总的来说，这门课收获颇多，感受颇深，受益匪浅！