FBDP 实验四

hive 安装

1. 下载mysql

`brew install mysql`

```
(base) chenyuanshan@chenyuanshandeMacBook-Air /usr/local mysql.server start
Starting MySQL
.. SUCCESS!
```

打开mysql shell：`mysql -u root -p`

```
(base) chenyuanshan@chenyuanshandeMacBook-Air /usr/local mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.21 Homebrew

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

可以用`exit`退出shell界面

至此完成mysql安装

2. 安装hive

`brew install hive`

创建hive-site.xml，复制网上内容进入（配置信息）

将hive-default.xml.template重命名为hive-default.xml

又根据别的教程，将hive-env.sh.template复制得到hive-env.sh，其中修改了hadoop_home

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

<configuration>
        <property>
                <name>javax.jdo.option.ConnectionURL</name>
                <value>jdbc:mysql://localhost:3306/metastore?
createDatabaseIfNotExist=true</value><!--MySQL地址metastore是元数据库的名称，需要在
mysql中创建相同名字的数据库-->
        </property>
        <property>
                <name>javax.jdo.option.ConnectionDriverName</name>
```

```
                <value>com.mysql.jdbc.Driver</value><!--MySQL驱动-->
        </property>
        <property>
                <name>javax.jdo.option.ConnectionUserName</name>
                <value>root</value><!--MySQL用户名-->
        </property>
        <property>
                <name>javax.jdo.option.ConnectionPassword</name>
                <value>123456</value><!--MySQL密码-->
        </property>
</configuration>
```

问题：初始化时出现了`java.lang.ClassNotFoundException: com.mysql.jdbc.Driver`的报错，据说是缺少了一个jar包，于是去官网下载了mysql-connector-java与本地mysql版本相对应的jar包，放入HIVE_HOME/lib下。初始化成功后，创建的原数据库就有了以下表格，是hive运行必须的相关内容。

```
| PARTITION_PARAMS              |
| PARTITIONS                    |
| REPL_TXN_MAP                  |
| ROLE_MAP                      |
| ROLES                         |
| RUNTIME_STATS                 |
| SCHEMA_VERSION                |
| SD_PARAMS                     |
| SDS                           |
| SEQUENCE_TABLE                |
| SERDE_PARAMS                  |
| SERDES                        |
| SKEWED_COL_NAMES              |
| SKEWED_COL_VALUE_LOC_MAP      |
| SKEWED_STRING_LIST            |
| SKEWED_STRING_LIST_VALUES     |
| SKEWED_VALUES                 |
| SORT_COLS                     |
| TAB_COL_STATS                 |
| TABLE_PARAMS                  |
| TBL_COL_PRIVS                 |
| TBL_PRIVS                     |
| TBLS                          |
| TXN_COMPONENTS                |
| TXN_TO_WRITE_ID               |
| TXNS                          |
| TYPE_FIELDS                   |
| TYPES                         |
| VERSION                       |
| WM_MAPPING                    |
| WM_POOL                       |
| WM_POOL_TO_TRIGGER            |
| WM_RESOURCEPLAN               |
| WM_TRIGGER                    |
| WRITE_SET                     |
+-------------------------------+
74 rows in set (0.00 sec)

mysql>
```

下面就打开Hadoop，再启动hive，（有错误警告信息，但是可以先忽略，不影响使用）

```
default
Time taken: 1.006 seconds, Fetched: 1 row(s)
hive> exit
    > ;
(base) chenyuanshan@chenyuanshandeMacBook-Air /usr/local/Cellar/hive/3.1.2_1/libexec/conf hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/Cellar/hive/3.1.2_1/libexec/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.cla
ss]
SLF4J: Found binding in [jar:file:/usr/local/Cellar/hadoop/3.3.0/libexec/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/Stati
cLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 4637855d-017e-43a4-b1a5-bfc01a3c8e62

Logging initialized using configuration in jar:file:/usr/local/Cellar/hive/3.1.2_1/libexec/lib/hive-common-3.1.2.jar!/hive-log4j2.properties As
ync: true
2020-12-02 20:21:24,292 INFO DataNucleus.Persistence: Property datanucleus.cache.level2 unknown - will be ignored
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically regi
stered via the SPI and manual loading of the driver class is generally unnecessary.
2020-12-02 20:21:26,163 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:26,165 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:26,167 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:26,167 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:26,168 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:26,168 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:28,243 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:28,244 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:28,244 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:28,244 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:28,245 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
2020-12-02 20:21:28,245 WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
Hive Session ID = b37def39-6c76-4522-aba1-300bc2320181
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, te
z) or using Hive 1.X releases.
hive> show databases;
OK
default
Time taken: 1.337 seconds, Fetched: 1 row(s)
hive>
```

数据库创建、查询、删除均成功。

```
hive> create database if not exists test;
OK
Time taken: 0.44 seconds
hive> show databases;
OK
default
test
Time taken: 0.061 seconds, Fetched: 2 row(s)
hive> show databases like 't.*';
OK
test
Time taken: 0.066 seconds, Fetched: 1 row(s)
```

安装sqoop

sqoop可以实现hive与mysql之间的联通，使用`brew install sqoop`安装，配置好sqoop-env.sh文件，利用`sqoop list-databases --connect jdbc:mysql://127.0.0.1:3306/ --username root -P`尝试与mysql联通，报错

`java.lang.NoClassDefFoundError: org/apache/commons/lang/StringUtils`

解决：下载驱动`commons-lang-2.6.tar`驱动

再次连接，报错`Unable to load authentication plugin 'caching_sha2_password'.`

解决：下载的`mysql-connector`驱动版本过

低，重新下载了高版本

```
(base) chenyuanshan@chenyuanshandeMacBook-Air /usr/local/Cellar/sqoop/1.4.7/libexec/conf/mysql-connector-java-8.0.15 sqoop list-databases
--connect jdbc:mysql://127.0.0.1:3306/ --username root -P
Warning: /usr/local/Cellar/sqoop/1.4.7/libexec/bin/../../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
错误：找不到或无法加载主类 org.apache.hadoop.hbase.util.GetJavaProperty
2020-12-08 15:32:07,267 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
Enter password:
2020-12-08 15:32:11,120 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically regi
stered via the SPI and manual loading of the driver class is generally unnecessary.
mysql
information_schema
performance_schema
sys
hive
```

检查环境版本

1. Java版本

```
(base) chenyuanshan@chenyuanshandeMacBook-Air ~ master java -version
java version "1.8.0_221"
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)
```

符合Java8/11的要求

2. Hadoop版本

```
(base) chenyuanshan@chenyuanshandeMacBook-Air ~ master hadoop version
Hadoop 3.3.0
Source code repository https://gitbox.apache.org/repos/asf/hadoop.git -r aa96f1871bfd858f9bac59cf2a81ec470da649af
Compiled by brahma on 2020-07-06T18:44Z
Compiled with protoc 3.7.1
From source with checksum 5dc29b802d6ccd77b262ef9d04d19c4
This command was run using /usr/local/Cellar/hadoop/3.3.0/libexec/share/hadoop/common/hadoop-common-3.3.0.jar
```

3. Python 版本

```
(base) chenyuanshan@chenyuanshandeMacBook-Air ~ master python --version
Python 2.7.16
```

配置Scala

使用homebrew下载，但是发现homebrew自动下载最新版本，而spark需要scala2.12，于是查询homebrew下载如何选择版本

首先 `brew search scala` 查看可选版本，再选择目标进行下载，如下所示

```
(base) ✘ chenyuanshan@chenyuanshandeMacBook-Air ~ master brew search scala
==> Formulae
scala              scala@2.11         scala@2.12         scalaenv           scalapack          scalariform        scalastyle
==> Casks
scala-ide
(base) chenyuanshan@chenyuanshandeMacBook-Air ~ master brew install scala@2.12
==> Downloading https://downloads.lightbend.com/scala/2.12.12/scala-2.12.12.tgz
######################################################################## 100.0%
==> Caveats
To use with IntelliJ, set the Scala home to:
  /usr/local/opt/scala@2.12/idea

scala@2.12 is keg-only, which means it was not symlinked into /usr/local,
because this is an alternate version of another formula.

If you need to have scala@2.12 first in your PATH run:
  echo 'export PATH="/usr/local/opt/scala@2.12/bin:$PATH"' >> ~/.zshrc

==> Summary
🍺  /usr/local/Cellar/scala@2.12/2.12.12: 44 files, 21.2MB, built in 6 seconds
```

安装好后，如下检验是否成功：

```
(base) chenyuanshan@chenyuanshandeMacBook-Air ~ master scala -version
Scala code runner version 2.12.12 -- Copyright 2002-2020, LAMP/EPFL and Lightbend, Inc.
(base) chenyuanshan@chenyuanshandeMacBook-Air ~ master scala
Welcome to Scala 2.12.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_221).
Type in expressions for evaluation. Or try :help.

scala> :exit()
No such command ':exit()'.  Type :help for help.

scala> :quit
```

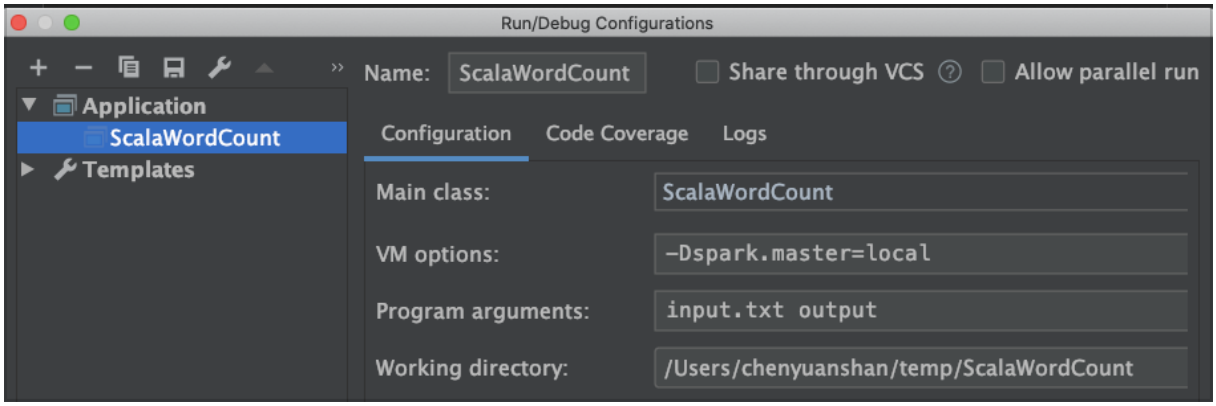
spark standalone配置

下载最新spark版本，安装后启动spark-shell，如下所示

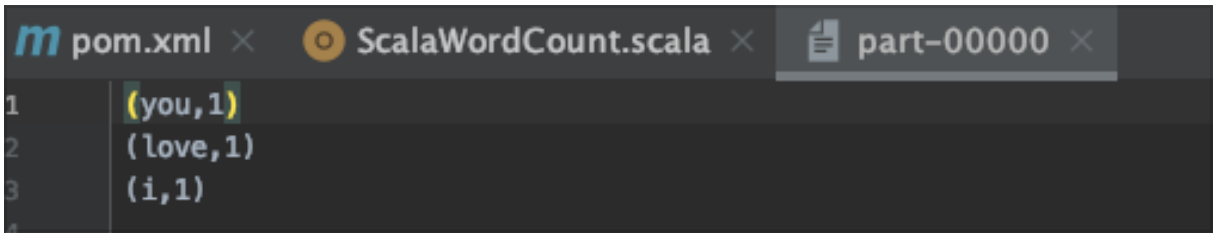
```
(base) chenyuanshan@chenyuanshandeMacBook-Air  /usr/local  spark-shell
20/11/30 19:18:42 WARN Utils: Your hostname, chenyuanshandeMacBook-Air.local resolves to a loopback address: 127.0.0.1; using 192.168.31.35 ins
tead (on interface en0)
20/11/30 19:18:42 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
20/11/30 19:18:43 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://192.168.31.35:4040
Spark context available as 'sc' (master = local[*], app id = local-1606735136535).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.0.1
      /_/

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_221)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

IDE配置（Intellij + Maven + Scala）

1. 配置jdk，下载Scala插件
2. 配置全局Scala sdk
3. 创建maven工程，在工程右键Add Framework Surport，选择Scala。在src下创建新文件夹WordCount，并Mark as sources directory，新建scala class，标记为object
4. 粘贴wordcount文件以及pom文件
5. Build project时报错`Multiple 'scala-library*.jar'`，原因是我导入sdk的时候，直接选择Scala的路径为/usr/local/Cellar/scala，但是在ide中应该选择/usr/local/Cellar/scala/idea/bin
6. 运行时报错no such method，查询发现是Scala版本不论本地为多少，都应该选择Scala11
7. 配置运行设置，如下



8. 最后得到运行结果，可以查看wordcount输出



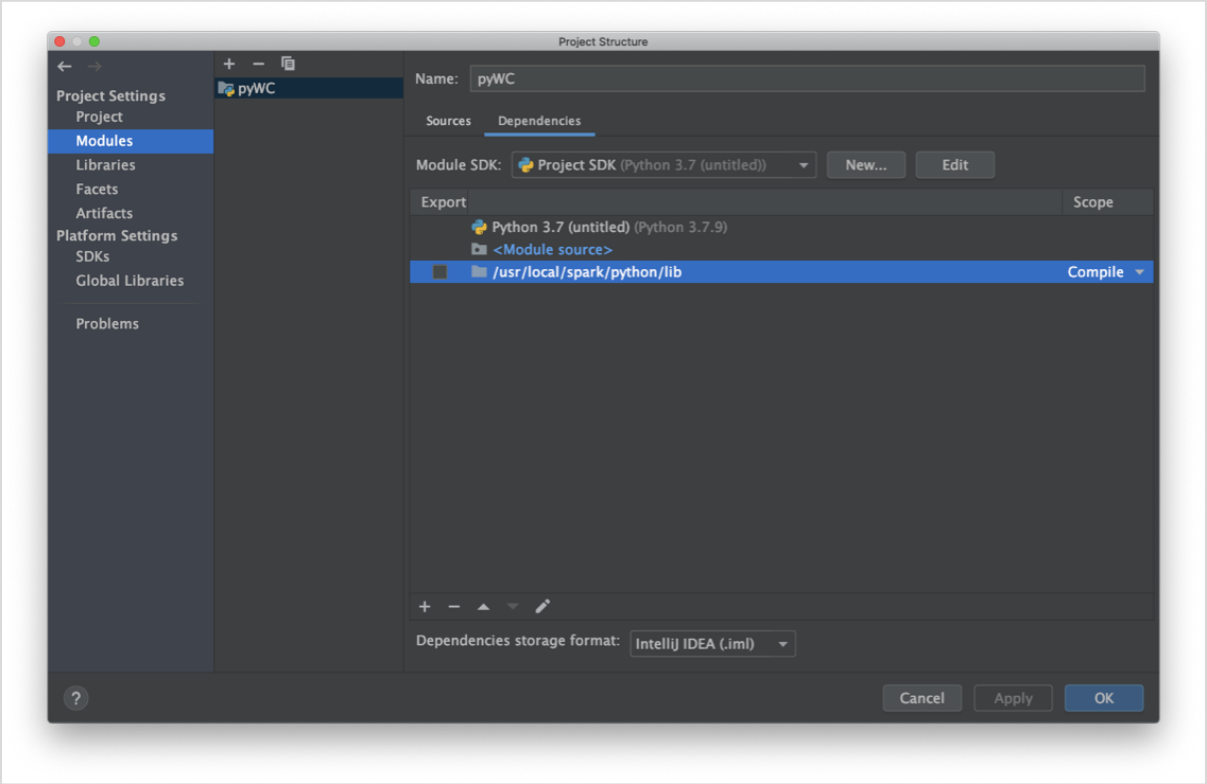
ide配置（intellij + spark + python）

1. 首先下载python3.7，因为Mac自带python2.7，会有语法区别，同时pip等一系列服务不再支持python2。官网下载后安装。
2. 在intellij中配置new project structure，用python3 sdk建立。但是在后续运行中，无法import pyspark

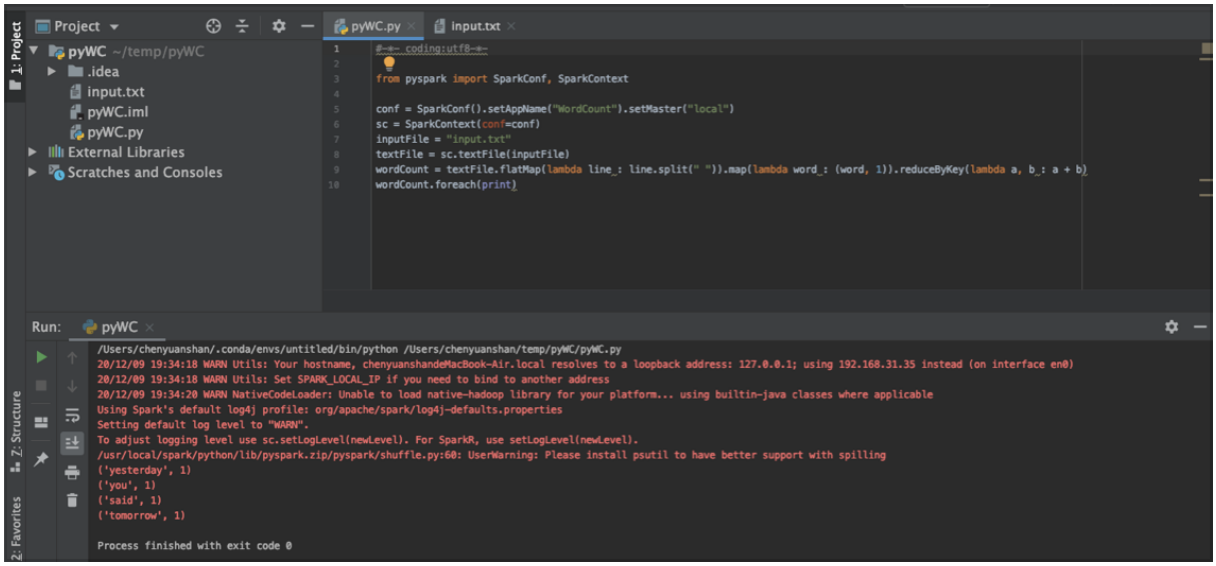
解决：在dependencies中添加pyspark package

3. 运行报错，`py4j.protocol.Py4JError: org.apache.spark.api.python.PythonUtils.getEncryptionEnabled does not exist in the JVM`发现问题是ide与spark运行出现了问题

   解决：run->Edit Configurations->Templates->Python，在右侧找到Environment->Environment Variables，配置PYTHONHOME为`/usr/local/spark/python`，SPARKHOME为`/usr/local/spark`，在modules引入jar directory，路径为`/usr/local/spark/python/lib`



4. 运行demo，成功



至此，环境安装配置完毕

实验分析

双十一热门商品和最受年轻人欢迎商家

- spark程序
  1. 最热门商品top100

jupyter popular_Item_py.txt a minute ago

File Edit View Language

```
191499: 2494
353560: 2250
1059899: 1917
713695: 1754
655904: 1674
67897: 1572
221663: 1547
1039919: 1511
454937: 1387
81360: 1361
514725: 1356
783997: 1351
823766: 1343
107407: 1319
889095: 1272
936203: 1270
770668: 1257
698879: 1235
349999: 1218
671759: 1167
186456: 1162
```

  2. 最热门商家top100

jupyter popular_merchant_py.txt 几秒前

File Edit View Language

```
4044: 12698
3828: 11800
3760: 9255
1892: 8586
1: 8475
1102: 8384
4173: 8039
422: 7200
1393: 6709
4976: 6584
1535: 6566
3491: 6297
2385: 5938
4644: 5704
3734: 5644
420: 5344
798: 5161
4760: 5145
184: 5016
4918: 4854
4218: 4789
```

- mapreduce程序
  1. 最热门商品top100

pom.xml | PopularItemMR.java | part-r-00000 | log_info_1111.csv

```
1	191499 times: 2494
2	353560 times: 2250
3	1059899 times: 1917
4	713695 times: 1754
5	655904 times: 1674
6	67897 times: 1572
7	221663 times: 1547
8	1039919 times: 1511
9	454937 times: 1387
10	81360 times: 1361
11	514725 times: 1356
12	783997 times: 1351
13	823766 times: 1343
14	107407 times: 1319
15	889095 times: 1272
16	936203 times: 1270
17	770668 times: 1257
18	698879 times: 1235
19	349999 times: 1218
20	671759 times: 1167
21	186456 times: 1162
22	315345 times: 1067
23	729259 times: 1021
24	946001 times: 1015
25	181387 times: 1002
26	926069 times: 1002
27	28895 times: 983
28	89953 times: 975
```

2. 最受欢迎商家top100

pom.xml | PopularMerchantMR.java | part-r-00000

```
1	4044 times: 12698
2	3828 times: 11800
3	3760 times: 9255
4	1892 times: 8586
5	1 times: 8475
6	1102 times: 8384
7	4173 times: 8039
8	422 times: 7200
9	1393 times: 6709
10	4976 times: 6584
11	1535 times: 6566
12	3491 times: 6297
13	2385 times: 5938
14	4644 times: 5704
15	3734 times: 5644
16	420 times: 5344
17	798 times: 5161
18	4760 times: 5145
19	184 times: 5016
20	4918 times: 4854
21	4218 times: 4789
22	361 times: 4684
23	2403 times: 4564
```

双十一购物的性别比例和年龄分布

- spark程序

```
In [13]: result4 = result3.map(lambda x : (x[1],x[0])).map(lambda x : (x[0],1))
         print(result4.take(5))

         [(5.0, 1), (4.0, 1), (5.0, 1), (5.0, 1), (5.0, 1)]

In [16]: result5 = result4.reduceByKey(lambda a,b:a+b)
         output = result5.collect()

In [19]: for (age_range, count) in output:
             print("%s: %i" % (age_range, count))

         4.0: 79991
         0.0: 92914
         8.0: 1266
         None: 2217
         5.0: 40777
         1.0: 24
         6.0: 35464
         2.0: 52871
         3.0: 111654
         7.0: 6992
```

```
In [21]: rdd2 = df1.select("user_id","gender").rdd
         rs1 = rdd2.map(lambda x: (x['user_id'],x['gender']))
         rs2 = rs1.groupByKey()
         rs3 = rs2.map(lambda x : (x[0],list(x[1]))).map(lambda x: (x[0],x[1][0]))
         rs4 = rs3.map(lambda x : (x[1],x[0])).map(lambda x : (x[0],1))
         rs5 = rs4.reduceByKey(lambda a,b:a+b)
         op = rs5.collect()
```

```
In [22]: for (gender, count) in op:
             print("%s: %i" % (gender, count))
         with open('output/gender_distribution.txt','w') as f:
             for (gender, count) in output:
                 f.write("%s: %i \n" % (gender, count))
```

```
0.0: 285638
None: 6436
1.0: 121670
2.0: 10426
```

- hive查询

  首先删除表头。启动hdfs，将数据上传。

```
(base) chenyuanshan@chenyuanshandeMacBook-Air ~/temp/proj4/data  master  sed 1d user_log_format1.csv >> user_log.csv
(base) chenyuanshan@chenyuanshandeMacBook-Air ~/temp/proj4/data  master  ls
test_format1.csv      train_format1.csv     user_info_format1.csv user_log.csv          user_log_format1.csv
(base) chenyuanshan@chenyuanshandeMacBook-Air ~/temp/proj4/data  master  head -5 user_log.csv
328862,323294,833,2882,2661,0829,0
328862,844400,1271,2882,2661,0829,0
328862,575153,1271,2882,2661,0829,0
328862,996875,1271,2882,2661,0829,0
328862,1086186,1271,1253,1049,0829,0
```

Shell 操作记录

```
cd /usr/local/Cellar/hadoop/3.3.0
sbin/start-dfs.sh
mysql.server start
hdfs dfs -mkdir -p /taobao/data
hdfs dfs -put user_log.csv /taobao/data
hive
hive> create database taobao;
hive> use taobao;
hive> CREATE EXTERNAL TABLE taobao.user_log(user_id INT,item_id INT,cat_id
INT,seller_id INT,brand_id INT,time_stamp INT,action_type INT) COMMENT 'Now
create taobao.user_log!' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED
AS TEXTFILE LOCATION '/taobao/data';
hive> select * from user_log limit 5;
```

```
OK
Time taken: 0.219 seconds
hive> select * from user_log limit 5;
OK
328862  323294  833     2882    2661    829     0
328862  844400  1271    2882    2661    829     0
328862  575153  1271    2882    2661    829     0
328862  996875  1271    2882    2661    829     0
328862  1086186 1271    1253    1049    829     0
Time taken: 3.242 seconds, Fetched: 5 row(s)
```

接着上传user_info

```
hive> CREATE EXTERNAL TABLE taobao.user_info(user_id INT,age_range INT,gender INT) COMMENT 'Now create taobao.user_info!' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/taobao/dataInfo';
```

```
OK
Time taken: 0.245 seconds
hive> select * from user_info limit 5;
OK
376517  6       1
234512  5       0
344532  5       0
186135  5       0
30230   5       0
Time taken: 0.224 seconds, Fetched: 5 row(s)
```

下面开始操作，报错memory溢出，决定减少文件体积，并采用view

```
hive> CREATE VIEW log_info AS (SELECT t1.user_id, t2.age_range, t2.gender FROM ((SELECT DISTINCT user_id FROM user_log WHERE (time_stamp = 1111 AND action_type = 2))t1 LEFT JOIN (SELECT user_id, age_range, gender FROM user_info)t2 ON t1.user_id = t2.user_id));
```

但是发现，正常select操作可以成功，涉及聚合函数则会报错，估计是运行内存不足，而保存为view导致每次都会再次运行原代码，造成资源使用拥挤。`FAILED: Execution Error, return code 2 from org.apache.hadoop.hive.ql.exec.mr.MapRedTask`决定改为table，再次执行无报错。

```
hive> SELECT gender,COUNT(user_id) FROM log_info GROUP BY gender;
Query ID = chenyuanshan_20201212170046_790c92dd-26eb-41e0-956b-6083e032177f
Total jobs = 1
Launching Job 1 out of 1
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>
Job running in-process (local Hadoop)
2020-12-12 17:00:48,106 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1662340505_0004
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 3838785174 HDFS Write: 9126996 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
NULL    6436
0       285638
1       121670
2       10426
Time taken: 1.95 seconds, Fetched: 4 row(s)
```

```
hive> SELECT COUNT(user_id) FROM log_info GROUP BY gender;

hive> SELECT COUNT(user_id) FROM log_info GROUP BY age_range;

hive> INSERT OVERWRITE DIRECTORY '/taobao/dataLogInfo' SELECT * FROM log_info;
```

```
hive> SELECT age_range,COUNT(DISTINCT user_id) from log_info GROUP BY age_range;
Query ID = chenyuanshan_20201212170346_cdd2cdb4-06d4-4ca9-bba3-c38be84a8982
Total jobs = 1
Launching Job 1 out of 1
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>
Job running in-process (local Hadoop)
2020-12-12 17:03:49,097 Stage-1 map = 0%,  reduce = 0%
2020-12-12 17:03:52,255 Stage-1 map = 100%,  reduce = 0%
2020-12-12 17:03:53,261 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1116004394_0005
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 3847912010 HDFS Write: 9126996 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
NULL    2217
0       92914
1       24
2       52871
3       111654
4       79991
5       40777
6       35464
7       6992
8       1266
Time taken: 6.338 seconds, Fetched: 10 row(s)
```

由于数据量大，要找到最节省资源和高效的方法

实验部分

数据处理

- 首先数据处理：train与test都要先与user_info merge，得到带有用户特征的数据集。
  - 1. 对于用户来说，更大可能购买的应该用户对网购的习惯程度（用所产生log次数来作为代理变量），用户对这家店的行为（点击，加购，购买以及收藏）
  - 2. 对于商户来说，商店的吸引人程度，用被所有用户购买/点击比例作为代理变量
- 什么指标最合适，还需要多次选择验证才能得出结论。
- 首先在制作需要使用的数据集时，需要把log信息的用户购买行为等统计出来，再统计商户行为等，产生大量数据，导致运行缓慢且无法将结果输出。
  1. 尝试解决1: 每次处理之后采取`createOrReplaceTempView()`的方法

     性能提升不明显，与hive中一致，view仅仅保存了运行代码而非结果，实际运行速度仍然很慢
  2. 尝试解决2: 每次处理之后才需`registerTempTable()`效果仍不明显
  3. 尝试解决3: 在sql语句中，直接采用`CREATE TABLE XXX AS (...)`速度提升明显，有效
- 在结果输出时，要设置repartition(1)，否则多线程输出会导致多个结果文件

机器学习模型部分

- 在处理好特征以及变量后，下一步是设计模型。首先尝试的是**支持向量机SVM分类器**
  - 模拟结果输出不满意：尽管使用训练集进行测试的准确率较高，但是模型的参数auc为0.5，该模型的使用意义不大，且分布极不均匀。

```
svmAccuracy = float(svmTotalCorrect)/train.count()
print(svmAccuracy)
```

```
0.9388378618743867
```

```
scoreAndLabels = train.map(lambda x:(float(model.predict(x.features)),x.label))
metrics = BinaryClassificationMetrics(scoreAndLabels)
print('PR:{:.4f}, AUC:{:.4f}'.format(metrics.areaUnderPR, metrics.areaUnderROC))
```

```
PR:0.2229, AUC:0.5001
```

- 尝试使用原数据特征，使用**随机森林模型**

```
acc2 = MulticlassClassificationEvaluator(labelCol='label', metricName='accuracy').evaluate(testRslt)
```

```
auc = BinaryClassificationEvaluator(labelCol='label').evaluate(testRslt)
print('acc[{}], auc[{}]'.format(acc2,auc))
```

```
acc[0.9396709323583181], auc[0.619954063732903]
```

得到auc效果较好，但召回率不高，说明模型在分辨可回购用户时作用不明显。第一次提交得分0.5 。