**赛题01**

**背景描述**

大数据时代背景下，电商经营模式发生很大改变。在传统运营模式中，缺乏数据积累，人们在做出一些决策行为过程中，更多是凭借个人经验和直觉，发展路径比较自我封闭。而大数据时代，为人们提供一种全新的思路，通过大量的数据分析得出的结果将更加现实和准确。商家可以对客户的消费行为信息数据进行收集和整理，比如消费者购买产品的花费、选择产品的渠道、偏好产品的类型、产品回购周期、购买产品的目的、消费者家庭背景、工作和生活环境、个人消费观和价值观等。通过数据追踪，知道顾客从哪儿来，是看了某网站投放的广告还是通过朋友推荐链接，是新访客还是老用户，喜欢浏览什么产品，购物车有无商品，是否清空，还有每一笔交易记录，精准锁定一定年龄、收入、对产品有兴趣的顾客，对顾客进行分组、标签化，通过不同标签组合运用，获得不同目标群体，以此开展精准推送。

因数据驱动的零售新时代已经到来，没有大数据，我们无法为消费者提供这些体验，为完成电商的大数据分析工作，你所在的小组将应用大数据技术，以Scala作为整个项目的基础开发语言，基于大数据平台综合利用Hive、Spark、Flink、Vue.js等技术，对数据进行处理、分析及可视化呈现，你们作为该小组的技术人员，请按照下面任务完成本次工作。

## 任务A：大数据平台搭建（容器环境）（15分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**宿主机可通过Asbru工具或SSH客户端进行SSH访问；  相关软件安装包在宿主机的/opt目录下，请选择对应的安装包进行安装，用不到的可忽略；  所有任务中应用命令必须采用绝对路径；  进入Master节点的方式为  docker exec -it master /bin/bash  进入Slave1节点的方式为  docker exec -it slave1 /bin/bash  进入Slave2节点的方式为  docker exec -it slave2 /bin/bash  三个容器节点的root密码均为123456 |

### 子任务一：Hadoop 完全分布式安装配置

本任务需要使用root用户完成相关配置，安装Hadoop需要配置前置环境。命令中要求使用绝对路径，具体要求如下:

1. 从宿主机/opt目录下将文件hadoop-3.1.3.tar.gz、jdk-8u212-linux-x64.tar.gz复制到容器Master中的/opt/software路径中（若路径不存在，则需新建），将Master节点JDK安装包解压到/opt/module路径中(若路径不存在，则需新建)，将JDK解压命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
2. 修改容器中/etc/profile文件，设置JDK环境变量并使其生效，配置完毕后在Master节点分别执行“java -version”和“javac”命令，将命令行执行结果分别截图并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
3. 请完成host相关配置，将三个节点分别命名为master、slave1、slave2，并做免密登录，用scp命令并使用绝对路径从Master复制JDK解压后的安装文件到slave1、slave2节点（若路径不存在，则需新建），并配置slave1、slave2相关环境变量，将全部scp复制JDK的命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
4. 在Master将Hadoop解压到/opt/module(若路径不存在，则需新建)目录下，并将解压包分发至slave1、slave2中，其中master、slave1、slave2节点均作为datanode，配置好相关环境，初始化Hadoop环境namenode，将初始化命令及初始化结果截图（截取初始化结果日志最后20行即可）粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
5. 启动Hadoop集群（包括hdfs和yarn），使用jps命令查看Master节点与slave1节点的Java进程，将jps命令与结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下。

### 子任务二：Spark on Yarn安装配置

本任务需要使用root用户完成相关配置，已安装Hadoop及需要配置前置环境，具体要求如下：

1. 从宿主机/opt目录下将文件spark-3.1.1-bin-hadoop3.2.tgz复制到容器Master中的/opt/software（若路径不存在，则需新建）中，将Spark包解压到/opt/module路径中(若路径不存在，则需新建)，将完整解压命令复制粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
2. 修改容器中/etc/profile文件，设置Spark环境变量并使环境变量生效，在/opt目录下运行命令spark-submit --version，将命令与结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
3. 完成on yarn相关配置，使用spark on yarn 的模式提交$SPARK\_HOME/examples/jars/spark-examples\_2.12-3.1.1.jar 运行的主类为org.apache.spark.examples.SparkPi，将运行结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下（截取Pi结果的前后各5行）。

（运行命令为：spark-submit --master yarn --class org.apache.spark.examples.SparkPi $SPARK\_HOME/examples/jars/spark-examples\_2.12-3.1.1.jar）

### 子任务三：HBase分布式安装配置

本任务需要使用root用户完成相关配置，安装HBase需要配置Hadoop和ZooKeeper等前置环境。命令中要求使用绝对路径，具体要求如下:

1. 从宿主机/opt目录下将文件apache-zookeeper-3.5.7-bin.tar.gz、hbase-2.2.3-bin.tar.gz复制到容器Master中的/opt/software路径中（若路径不存在，则需新建），将ZooKeeper、HBase安装包解压到/opt/module目录下，将HBase的解压命令复制并粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；
2. 完成ZooKeeper相关部署，用scp命令并使用绝对路径从容器master复制HBase解压后的包分发至slave1、slave2中，并修改相关配置，配置好环境变量，在容器Master节点中运行命令hbase version，将全部复制命令复制并将hbase version命令的结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下;
3. 启动HBase后在三个节点分别使用jps命令查看，并将结果分别截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下；正常启动后在hbase shell中查看命名空间，将查看命名空间的结果截图粘贴至客户端桌面【Release\任务A提交结果.docx】中对应的任务序号下。

## 任务B：离线数据处理（25分）

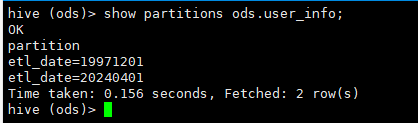
**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**各节点可通过Asbru工具或SSH客户端进行SSH访问；  主节点MySQL数据库用户名/密码：root/123456（已配置远程连接）；  Hive的配置文件位于/opt/apache-hive-2.3.4-bin/conf/  Spark任务在Yarn上用Client运行，方便观察日志。 |

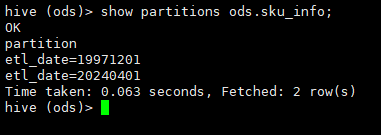
### 子任务一：数据抽取

编写Scala代码，使用Spark将MySQL的shtd\_store库中表user\_info、sku\_info、base\_province、base\_region、order\_info、order\_detail的数据增量抽取到Hive的ods库中对应表user\_info、sku\_info、base\_province、base\_region、order\_info、order\_detail中。(若ods库中部分表没有数据，正常抽取即可)

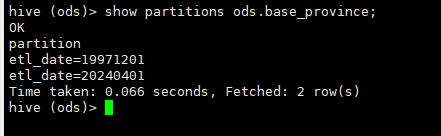
1. 抽取shtd\_store库中user\_info的增量数据进入Hive的ods库中表user\_info。根据ods.user\_info表中operate\_time或create\_time作为增量字段(即MySQL中每条数据取这两个时间中较大的那个时间作为增量字段去和ods里的这两个字段中较大的时间进行比较)，只将新增的数据抽入，字段名称、类型不变，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.user\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



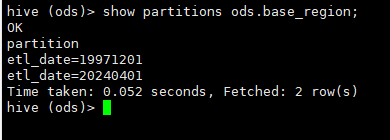
1. 抽取shtd\_store库中sku\_info的增量数据进入Hive的ods库中表sku\_info。根据ods.sku\_info表中create\_time作为增量字段，只将新增的数据抽入，字段名称、类型不变，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.sku\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



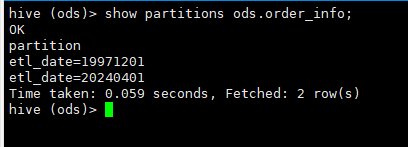
1. 抽取shtd\_store库中base\_province的增量数据进入Hive的ods库中表base\_province。根据ods.base\_province表中id作为增量字段，只将新增的数据抽入，字段名称、类型不变并添加字段create\_time取当前时间，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.base\_province命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



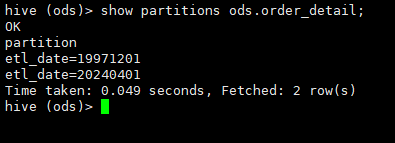
1. 抽取shtd\_store库中base\_region的增量数据进入Hive的ods库中表base\_region。根据ods.base\_region表中id作为增量字段，只将新增的数据抽入，字段名称、类型不变并添加字段create\_time取当前时间，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.base\_region命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



1. 抽取shtd\_store库中order\_info的增量数据进入Hive的ods库中表order\_info，根据ods.order\_info表中operate\_time或create\_time作为增量字段(即MySQL中每条数据取这两个时间中较大的那个时间作为增量字段去和ods里的这两个字段中较大的时间进行比较)，只将新增的数据抽入，字段名称、类型不变，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.order\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



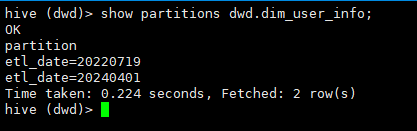
1. 抽取shtd\_store库中order\_detail的增量数据进入Hive的ods库中表order\_detail，根据ods.order\_detail表中create\_time作为增量字段，只将新增的数据抽入，字段名称、类型不变，同时添加静态分区，分区字段为etl\_date，类型为String，且值为当前比赛日的前一天日期（分区字段格式为yyyyMMdd）。使用hive cli执行show partitions ods.order\_detail命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下。



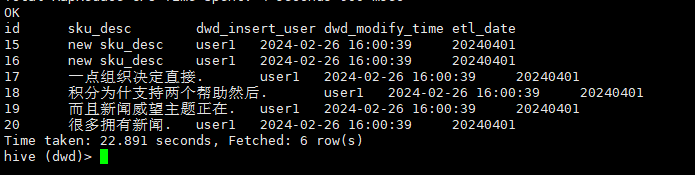
### 子任务二：数据清洗

编写Scala代码，使用Spark将ods库中相应表数据全量抽取到Hive的dwd库中对应表中。表中有涉及到timestamp类型的，均要求按照yyyy-MM-dd HH:mm:ss，不记录毫秒数，若原数据中只有年月日，则在时分秒的位置添加00:00:00，添加之后使其符合yyyy-MM-dd HH:mm:ss。(若dwd库中部分表没有数据，正常抽取即可)

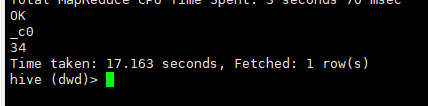
1. 抽取ods库中user\_info表中昨天的分区（子任务一生成的分区）数据，并结合dim\_user\_info最新分区现有的数据，根据id合并数据到dwd库中dim\_user\_info的分区表（合并是指对dwd层数据进行插入或修改，需修改的数据以id为合并字段，根据operate\_time排序取最新的一条），分区字段为etl\_date且值与ods库的相对应表该值相等，同时若operate\_time为空，则用create\_time填充，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”。若该条记录第一次进入数仓dwd层则dwd\_insert\_time、dwd\_modify\_time均存当前操作时间，并进行数据类型转换。若该数据在进入dwd层时发生了合并修改，则dwd\_insert\_time时间不变，dwd\_modify\_time存当前操作时间，其余列存最新的值。使用hive cli执行show partitions dwd.dim\_user\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



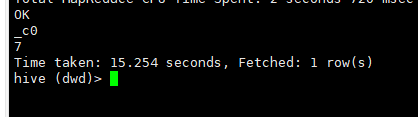
1. 抽取ods库sku\_info表中昨天的分区（子任务一生成的分区）数据，并结合dim\_sku\_info最新分区现有的数据，根据id合并数据到dwd库中dim\_sku\_info的分区表（合并是指对dwd层数据进行插入或修改，需 修改的数据以id为合并字段，根据create\_time排序取最新的一条），分区字段为etl\_date且值与ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”。若该条数据第一次进入数仓dwd层则dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。若该数据在进入dwd层时发生了合并修改，则dwd\_insert\_time时间不变，dwd\_modify\_time存当前操作时间，其余列存最新的值。使用hive cli查询表dim\_sku\_info的字段id、sku\_desc、dwd\_insert\_user、dwd\_modify\_time、etl\_date，条件为最新分区的数据，id大于等于15且小于等于20，并且按照id升序排序，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



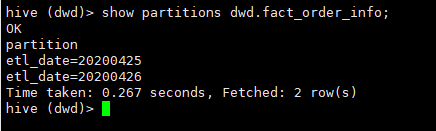
1. 抽取ods库base\_province表中昨天的分区（子任务一生成的分区）数据，并结合dim\_province最新分区现有的数据，根据id合并数据到dwd库中dim\_province的分区表（合并是指对dwd层数据进行插入或修改，需修改的数据以id为合并字段，根据create\_time排序取最新的一条），分区字段为etl\_date且值与ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”。若该条数据第一次进入数仓dwd层则dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。若该数据在进入dwd层时发生了合并修改，则dwd\_insert\_time时间不变，dwd\_modify\_time存当前操作时间，其余列存最新的值。使用hive cli在表dwd.dim\_province最新分区中，查询该分区中数据的条数，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



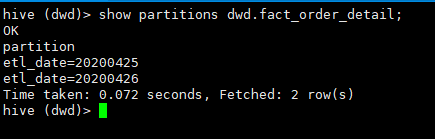
1. 抽取ods库base\_region表中昨天的分区（子任务一生成的分区）数据，并结合dim\_region最新分区现有的数据，根据id合并数据到dwd库中dim\_region的分区表（合并是指对dwd层数据进行插入或修改，需修改的数据以id为合并字段，根据create\_time排序取最新的一条），分区字段为etl\_date且值与ods库的相对应表该值相等，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列,其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”。若该条数据第一次进入数仓dwd层则dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。若该数据在进入dwd层时发生了合并修改，则dwd\_insert\_time时间不变，dwd\_modify\_time存当前操作时间，其余列存最新的值。使用hive cli在表dwd.dim\_region最新分区中，查询该分区中数据的条数，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



1. 将ods库中order\_info表昨天的分区（子任务一生成的分区）数据抽取到dwd库中fact\_order\_info的动态分区表，分区字段为etl\_date，类型为String，取create\_time值并将格式转换为yyyyMMdd，同时若operate\_time为空，则用create\_time填充，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列，其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”，dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。使用hive cli执行show partitions dwd.fact\_order\_info命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



1. 将ods库中order\_detail表昨天的分区（子任务一中生成的分区）数据抽取到dwd库中fact\_order\_detail的动态分区表，分区字段为etl\_date，类型为String，取create\_time值并将格式转换为yyyyMMdd，并添加dwd\_insert\_user、dwd\_insert\_time、dwd\_modify\_user、dwd\_modify\_time四列，其中dwd\_insert\_user、dwd\_modify\_user均填写“user1”，dwd\_insert\_time、dwd\_modify\_time均填写当前操作时间，并进行数据类型转换。使用hive cli执行show partitions dwd.fact\_order\_detail命令，将结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下。

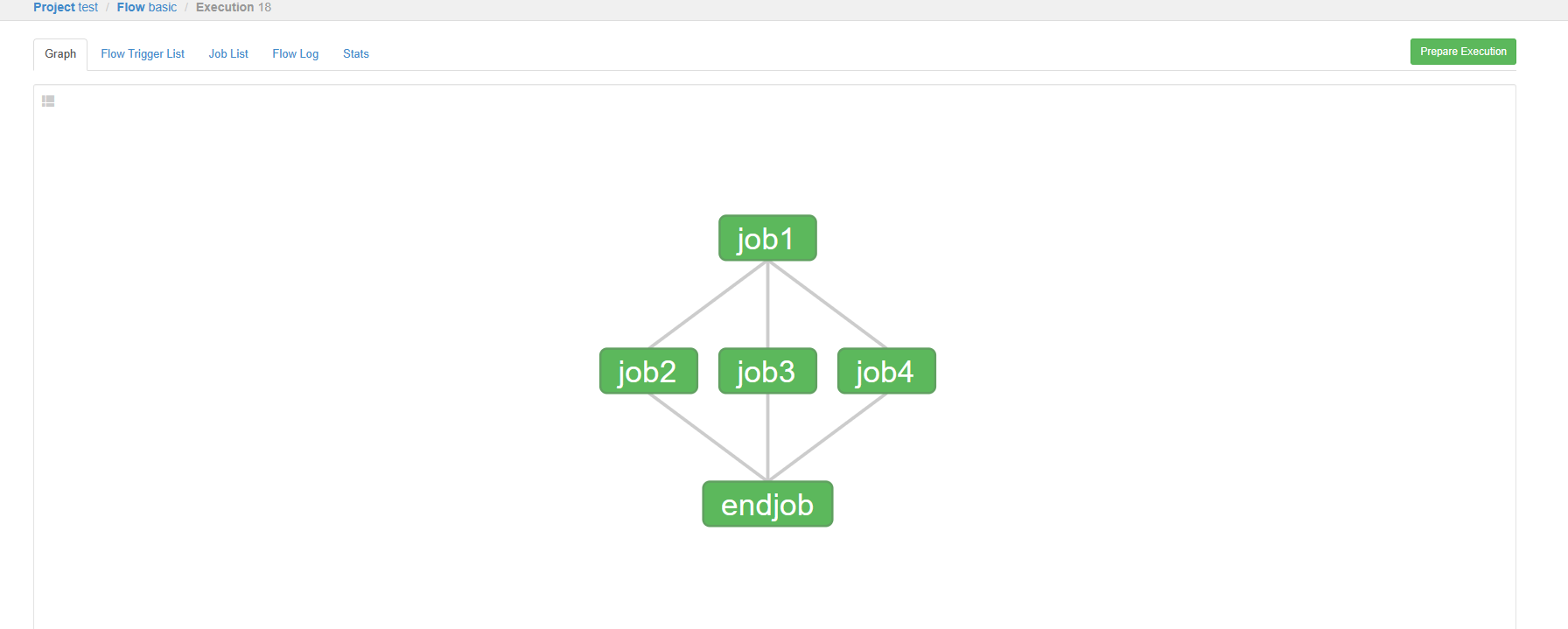


### 子任务三：指标计算

编写Scala代码，使用Spark计算相关指标。

**注：在指标计算中，不考虑订单信息表中order\_status字段的值，将所有订单视为有效订单。计算订单金额或订单总金额时只使用final\_total\_amount字段。需注意dwd所有的维表取最新的分区。**

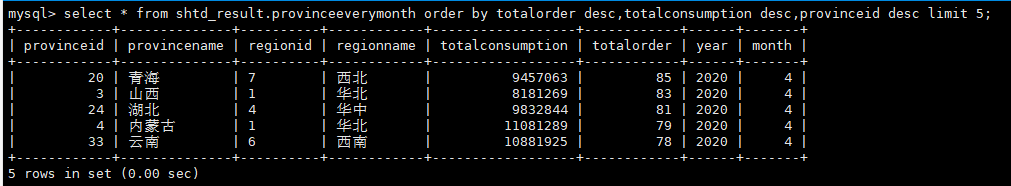
1. 本任务基于以下2、3、4小题完成（不包括5，6题），使用Azkaban完成第2、3、4题任务代码的调度。工作流要求，使用shell输出“开始”作为工作流的第一个job（job1），2、3、4题任务为串行任务且它们依赖job1的完成（命名为job2、job3、job4），job2、job3、job4完成之后使用shell输出“结束”作为工作流的最后一个job（endjob），endjob依赖job2、job3、job4，并将最终任务调度完成后的工作流截图，将截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下；



1. 根据dwd层表统计每个省份、每个地区、每个月下单的数量和下单的总金额，存入MySQL数据库shtd\_result的provinceeverymonth表中（表结构如下），然后在Linux的MySQL命令行中根据订单总数、订单总金额、省份表主键均为降序排序，查询出前5条，将SQL语句复制粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下;

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| provinceid | int | 省份表主键 |  |
| provincename | text | 省份名称 |  |
| regionid | int | 地区表主键 |  |
| regionname | text | 地区名称 |  |
| totalconsumption | double | 订单总金额 | 当月订单总金额 |
| totalorder | int | 订单总数 | 当月订单总数 |
| year | int | 年 | 订单产生的年 |
| month | int | 月 | 订单产生的月 |

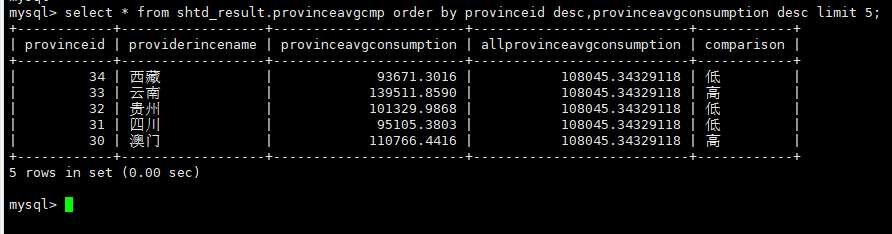
select \* from shtd\_result.provinceeverymonth order by totalorder desc,totalconsumption desc,provinceid desc limit 5;



1. 请根据dwd层表计算出2020年4月每个省份的平均订单金额和所有省份平均订单金额相比较结果（“高/低/相同”）,存入MySQL数据库shtd\_result的provinceavgcmp表（表结构如下）中，然后在Linux的MySQL命令行中根据省份表主键、该省平均订单金额均为降序排序，查询出前5条，将SQL语句复制粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下;

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| provinceid | int | 省份表主键 |  |
| provincename | text | 省份名称 |  |
| provinceavgconsumption | double | 该省平均订单金额 |  |
| allprovinceavgconsumption | double | 所有省平均订单金额 |  |
| comparison | text | 比较结果 | 该省平均订单金额和所有省平均订单金额比较结果，值为：高/低/相同 |

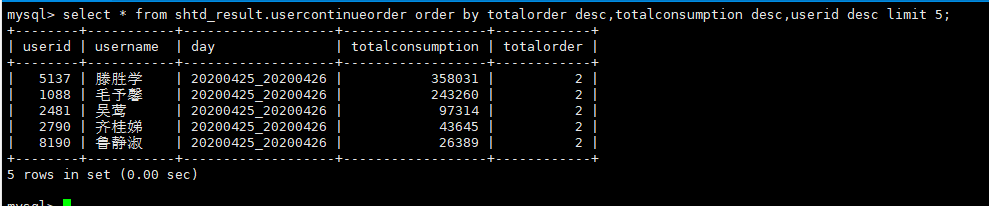
select \* from shtd\_result.provinceavgcmp order by provinceid desc,provinceavgconsumption desc limit 5;



1. 根据dwd层表统计在两天内连续下单并且下单金额保持增长的用户，存入MySQL数据库shtd\_result的usercontinueorder表(表结构如下)中，然后在Linux的MySQL命令行中根据订单总数、订单总金额、客户主键均为降序排序，查询出前5条，将SQL语句复制粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务B提交结果.docx】中对应的任务序号下。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| userid | int | 客户主键 |  |
| username | text | 客户名称 |  |
| day | text | 日 | 记录下单日的时间，格式为  yyyyMMdd\_yyyyMMdd  例如： 20220101\_20220102 |
| totalconsumption | double | 订单总金额 | 连续两天的订单总金额 |
| totalorder | int | 订单总数 | 连续两天的订单总数 |

select \* from shtd\_result.usercontinueorder order by totalorder desc,totalconsumption desc,userid desc limit 5;



## 任务C：数据挖掘（10分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**各节点可通过Asbru工具或SSH客户端进行SSH访问；  主节点MySQL数据库用户名/密码：root/123456（已配置远程连接）；  Hive的配置文件位于/opt/apache-hive-2.3.4-bin/conf/  Spark任务在Yarn上用Client运行，方便观察日志。  该任务均使用Scala编写，利用Spark相关库完成。 |

### 子任务一：特征工程

剔除订单信息表与订单详细信息表中用户id与商品id不存在现有的维表中的记录，同时建议多利用缓存并充分考虑并行度来优化代码，达到更快的计算效果。

1. 根据Hive的dwd库中相关表或MySQL中shtd\_store中相关表（order\_detail、sku\_info），计算出与用户id为6708的用户所购买相同商品种类最多的前10位用户（只考虑他俩购买过多少种相同的商品，不考虑相同的商品买了多少次），将10位用户id进行输出，若与多个用户购买的商品种类相同，则输出结果按照用户id升序排序，输出格式如下，将结果截图粘贴至客户端桌面【Release\任务C提交结果.docx】中对应的任务序号下；

结果格式如下：

-------------------相同种类前10的id结果展示为：--------------------

1,2,901,4,5,21,32,91,14,52

1. 根据Hive的dwd库中相关表或MySQL中shtd\_store中相关商品表（sku\_info），获取id、spu\_id、price、weight、tm\_id、category3\_id 这六个字段并进行数据预处理，对price、weight进行规范化(StandardScaler)处理，对spu\_id、tm\_id、category3\_id进行one-hot编码处理（若该商品属于该品牌则置为1，否则置为0）,并按照id进行升序排序，在集群中输出第一条数据前10列（无需展示字段名），将结果截图粘贴至客户端桌面【Release\任务C提交结果.docx】中对应的任务序号下。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| id | double | 主键 |  |
| price | double | 价格 |  |
| weight | double | 重量 |  |
| spu\_id#1 | double | spu\_id 1 | 若属于该spu\_id，则内容为1否则为0 |
| spu\_id#2 | double | spu\_id 2 | 若属于该spu\_id，则内容为1否则为0 |
| ..... | double |  |  |
| tm\_id#1 | double | 品牌1 | 若属于该品牌，则内容为1否则为0 |
| tm\_id#2 | double | 品牌2 | 若属于该品牌，则内容为1否则为0 |
| …… | double |  |  |
| category3\_id#1 | double | 分类级别3 1 | 若属于该分类级别3，则内容为1否则为0 |
| category3\_id#2 | double | 分类级别3 2 | 若属于该分类级别3，则内容为1否则为0 |
| …… |  |  |  |

结果格式如下：

--------------------第一条数据前10列结果展示为：---------------------

1.0,0.892346,1.72568,0.0,0.0,0.0,0.0,1.0,0.0,0.0

### 子任务二：推荐系统

1. 根据子任务一的结果，计算出与用户id为6708的用户所购买相同商品种类最多的前10位用户id（只考虑他俩购买过多少种相同的商品，不考虑相同的商品买了多少次），并根据Hive的dwd库中相关表或MySQL数据库shtd\_store中相关表，获取到这10位用户已购买过的商品，并剔除用户6708已购买的商品，通过计算这10位用户已购买的商品（剔除用户6708已购买的商品）与用户6708已购买的商品数据集中商品的余弦相似度累加再求均值，输出均值前5商品id作为推荐使用，将执行结果截图粘贴至客户端桌面【Release\任务C提交结果.docx】中对应的任务序号下。

结果格式如下：

------------------------推荐Top5结果如下------------------------

相似度top1(商品id：1，平均相似度：0.983456)

相似度top2(商品id：71，平均相似度：0.782672)

相似度top3(商品id：22，平均相似度：0.7635246)

相似度top4(商品id：351，平均相似度：0.7335748)

相似度top5(商品id：14，平均相似度：0.522356)

## 任务D：数据采集与实时计算（20分）

**环境说明：**

|  |
| --- |
| **服务端登录地址详见各任务服务端说明。**  **补充说明：**各节点可通过Asbru工具或SSH客户端进行SSH访问；  Flink任务在Yarn上用per job模式（即Job分离模式，不采用Session模式），方便Yarn回收资源。 |

### 子任务一：实时数据采集

1. 在主节点使用Flume采集实时数据生成器10050端口的socket数据（实时数据生成器脚本为主节点/data\_log目录下的dj\_data\_gen脚本，该脚本为主节点本地部署且使用socket传输），将数据存入到Kafka的Topic中（Topic名称为order，分区数为4），使用Kafka自带的消费者消费order（Topic）中的数据，将前2条数据的结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下；

**注：需先启动已配置好的Flume再启动脚本，否则脚本将无法成功启动，启动方式为进入/data\_log目录执行./dj\_data\_gen(如果没有权限，请执行授权命令chmod 777 /data\_log/dj\_data\_gen)**

1. 采用多路复用模式，Flume接收数据注入kafka 的同时，将数据备份到HDFS目录/user/test/flumebackup下，将查看备份目录下的第一个文件的前2条数据的命令与结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下。

### 子任务二：使用Flink处理Kafka中的数据

编写Scala代码，使用Flink消费Kafka中Topic为order的数据并进行相应的数据统计计算（订单信息对应表结构order\_info,订单详细信息对应表结构order\_detail（来源类型和来源编号这两个字段不考虑，所以在实时数据中不会出现），同时计算中使用order\_info或order\_detail表中create\_time或operate\_time取两者中值较大者作为EventTime，若operate\_time为空值或无此列，则使用create\_time填充，允许数据延迟5s，订单状态分别为1001:创建订单、1002:支付订单、1003:取消订单、1004:完成订单、1005:申请退回、1006:退回完成。另外对于数据结果展示时，不要采用例如：1.9786518E7的科学计数法）。

1. 使用Flink消费Kafka中的数据，统计商城实时订单实收金额（需要考虑订单状态，若有取消订单、申请退回、退回完成则不计入订单实收金额，其他状态的则累加），将key设置成totalprice存入Redis中。使用redis cli以get key方式获取totalprice值，将结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下，需两次截图，第一次截图和第二次截图间隔1分钟以上，第一次截图放前面，第二次截图放后面；
2. 在任务1进行的同时，使用侧边流，监控若发现order\_status字段为退回完成, 将key设置成totalrefundordercount存入Redis中，value存放用户退款消费额。使用redis cli以get key方式获取totalrefundordercount值，将结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下，需两次截图，第一次截图和第二次截图间隔1分钟以上，第一次截图放前面，第二次截图放后面；
3. 在任务1进行的同时，使用侧边流，监控若发现order\_status字段为取消订单,将数据存入MySQL数据库shtd\_result的order\_info表中，然后在Linux的MySQL命令行中根据id降序排序，查询列id、consignee、consignee\_tel、final\_total\_amount、feight\_fee，查询出前5条，将SQL语句复制粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下，将执行结果截图粘贴至客户端桌面【Release\任务D提交结果.docx】中对应的任务序号下。

## 任务E：数据可视化（15分）

**环境说明：**

|  |
| --- |
| **数据接口地址及接口描述详见各任务服务端说明。** |

### 子任务一：用柱状图展示消费额最高的省份

编写Vue工程代码，根据接口，用柱状图展示2020年消费额最高的5个省份，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

### 子任务二：用柱状图展示消费额最低的省份

编写Vue工程代码，根据接口，用柱状图展示2020年消费额最低的5个省份，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

### 子任务三：用折线图展示每年上架商品数量变化

编写Vue工程代码，根据接口，用折线图展示每年上架商品数量的变化情况，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

### 子任务四：用条形图展示平均消费额最高的省份

编写Vue工程代码，根据接口，用条形图展示2020年平均消费额（四舍五入保留两位小数）最高的5个省份，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

### 子任务五：用折柱混合图展示省份平均消费额和地区平均消费额

编写Vue工程代码，根据接口，用折柱混合图展示2020年各省份平均消费额（四舍五入保留两位小数）和地区平均消费额（四舍五入保留两位小数）的对比情况，柱状图展示平均消费额最高的5个省份，折线图展示这5个省所在的地区的平均消费额变化，同时将用于图表展示的数据结构在浏览器的console中进行打印输出，将图表可视化结果和浏览器console打印结果分别截图并粘贴至客户端桌面【Release\任务E提交结果.docx】中对应的任务序号下。

## 任务F：综合分析（10分）

### 子任务一：Kafka中的数据如何保证不丢失？

本次竞赛使用到了Kafka，将内容编写至客户端桌面【Release\任务F提交结果.docx】中对应的任务序号下。

### 子任务二：请描述HBase的rowkey设计原则。

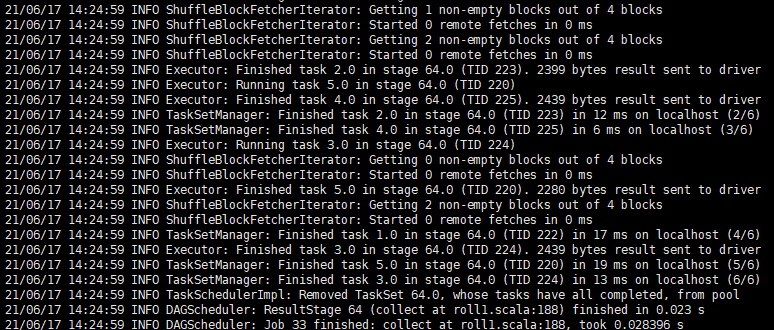
请简要概述HBase的rowkey的重要性并说明在设计rowkey时应遵循哪些原则，将内容编写至客户端桌面【Release\任务F提交结果.docx】中对应的任务序号下。

### 子任务三：Spark的数据本地性有哪几种，分别表示什么？

本次竞赛中使用到了Spark，其中有些JOB运行较慢，有些较快，一部分原因与数据位置和计算位置相关，其数据本地性可以在SparkUI中查看到。请问Spark的数据本地性有哪几种（英文）？分别表示的含义是什么（中文）？将内容编写至客户端桌面【Release\任务F提交结果.docx】中对应的任务序号下。

## 附录：补充说明

### 命令行截图样例（保证清晰）



### 表结构说明

#### 订单信息表（order\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| consignee | 收货人 |  |
| consignee\_tel | 收件人电话 |  |
| final\_total\_amount | 总金额 |  |
| order\_status | 订单状态 |  |
| user\_id | 用户id | 对应用户表id |
| delivery\_address | 送货地址 |  |
| order\_comment | 订单备注 |  |
| out\_trade\_no | 订单交易编号（第三方支付用） |  |
| trade\_body | 订单描述（第三方支付用） |  |
| create\_time | 创建时间 |  |
| operate\_time | 操作时间 |  |
| expire\_time | 失效时间 |  |
| tracking\_no | 物流单编号 |  |
| parent\_order\_id | 父订单编号 |  |
| img\_url | 图片路径 |  |
| province\_id | 省份id | 对应省份表id |
| benefit\_reduce\_amount | 优惠金额 |  |
| original\_total\_amount | 原价金额 |  |
| feight\_fee | 运费 |  |

#### 订单详细信息表（order\_detail）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| order\_id | 订单编号 | 对应订单信息表id |
| sku\_id | 商品id | 对应商品表id |
| sku\_name | 商品名称 |  |
| img\_url | 图片路径 |  |
| order\_price | 购买价格 | 下单时的商品价格 |
| sku\_num | 购买数量 |  |
| create\_time | 创建时间 |  |
| source\_type | 来源类型 | 无实际意义 |
| source\_id | 来源编号 | 无实际意义 |

#### 1.3 商品表（sku\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| spu\_id | spuId |  |
| price | 价格 |  |
| sku\_name | 商品名称 |  |
| sku\_desc | 商品描述 |  |
| weight | 重量 |  |
| tm\_id | 品牌id |  |
| category3\_id | 三级分类id |  |
| sku\_default\_img | 默认显示图片 |  |
| create\_time | 创建时间 |  |

#### 1.4 用户表（user\_info）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| login\_name | 用户名 |  |
| nick\_name | 用户昵称 |  |
| passwd | 用户密码 |  |
| name | 姓名 |  |
| phone\_num | 手机号 |  |
| email | 邮箱 |  |
| head\_img | 头像 |  |
| user\_level | 用户级别 |  |
| birthday | 生日 |  |
| gender | 性别M男,F女 |  |
| create\_time | 创建时间 |  |
| operate\_time | 操作时间 |  |

#### 1.5 省份表（base\_province）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| name | 省份名称 |  |
| region\_id | 地区id | 对应地区表id |
| area\_code | 地区编码 |  |
| iso\_code | 国际编码 |  |

#### 1.6 地区表（base\_region）

|  |  |  |
| --- | --- | --- |
| 标签 | 中文含义 | 备注 |
| id | 主键 |  |
| region\_name | 地区名称 |  |