

2024 年河南省职业院校 技能大赛高职组

“大数据分析与应用” 赛项

任 务 书 3

参赛队编号：_____

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 Java、Scala、JavaScript 作为整个项目的基础开发语言，基于大数据平台综合利用 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。

模块 A：数据采集（15 分）

环境说明：

服务端登录地址详见各模块服务端说明。

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 的配置文件位于主节点/opt/module/hive-3.1.2/conf/
Spark 任务在 Yarn 上用 Client 运行，方便观察日志；
建议使用 gson 解析 json 数据。

任务一：离线数据采集

编写 Scala 工程代码，将 MySQL 的 ds_db01 库中表 order_master、order_detail、coupon_info、coupon_use、product_browse、product_info、customer_inf、customer_login_log、order_cart、customer_level_inf、customer_addr 的数据增量抽取到 Hive 的 ods 库中对应表 order_master、order_detail、coupon_info、coupon_use、product_browse、product_info、customer_inf、customer_login_log、order_cart、customer_level_inf、customer_addr 中(ods 库中部分表没有数据，正常抽取即可)。

1、抽取 ds_db01 库中 order_master 的增量数据进入 Hive 的 ods 库中表 order_master。根据 ods.order_master 表中 modified_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_master 命令，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下；

2、抽取 ds_db01 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 modified_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，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下；

3、抽取 ds_db01 库中 coupon_info 的增量数据进入 Hive 的 ods 库中表 coupon_info，根据 ods.coupon_info 表中 modified_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.coupon_info 命令，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下；

4、抽取 ds_db01 库中 coupon_use 的增量数据进入 Hive 的 ods 库中表 coupon_use，增量字段取 ods.coupon_use 表中 get_time、used_time、pay_time 中的最大者，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 查询最新分区数据总条数，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下；

5、抽取 ds_db01 库中 product_browse 的增量数据进入 Hive 的 ods 库中表 product_browse，根据 ods.product_browse 表中 modified_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.product_browse 命令，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下。

- 6、抽取 ds_db01 库中 product_info 的增量数据进入 Hive 的 ods 库中表 product_info，根据 ods.product_info 表中 modified_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.product_info 命令，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下；
- 7、抽取 ds_db01 库中 customer_inf 的增量数据进入 Hive 的 ods 库中表 customer_inf，根据 ods.customer_inf 表中 modified_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.customer_inf 命令，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下；
- 8、抽取 ds_db01 库中 customer_login_log 的增量数据进入 Hive 的 ods 库中表 customer_login_log，根据 ods.customer_login_log 表中 login_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.customer_login_log 命令，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下；
- 9、抽取 ds_db01 库中 order_cart 的增量数据进入 Hive 的 ods 库中表 order_cart，根据 ods.order_cart 表中 modified_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_cart 命令，

将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下；

10、抽取 ds_db01 库中 customer_addr 的增量数据进入 Hive 的 ods 库中表 customer_addr，根据 ods.customer_addr 表中 modified_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.customer_addr 命令，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下；

11、抽取 ds_db01 库中 customer_level_inf 的增量数据进入 Hive 的 ods 库中表 customer_level_inf，根据 ods.customer_level_inf 表中 modified_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.customer_level_inf 命令，将执行结果截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下。

任务二：实时数据采集

1、在主节点使用 Flume 采集实时数据生成器 25001 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 gen_ds_data_to_socket 脚本，该脚本为 Master 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 ods_mall_log，分区数为 2，ZK 关于 Kafka 的信息在其/kafka 节点），使用 Kafka 自带的消费者消费 ods_mall_log (Topic) 中的数据，查看 Topic 中的前 1 条数据的结果，将查看命令与结果完整的截图粘贴至客户端桌面【Release\模块 A 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./gen_ds_data_to_socket（如果没有权限，请执行授权命令 chmod 777 /data_log/gen_ds_data_to_socket）

2、实时脚本启动后，在主节点进入到 maxwell-1.29.0 的解压后目录下（在 /opt/module 下），配置相关文件并启动，读取主节点 MySQL 数据的 binlog 日志（MySQL 的 binlog 相关配置已完毕，只需要关注 ds_realtime_db 数据库的表）到 Kafka 的 Topic 中（Topic 名称为 ods_mall_data，分区数为 2，ZK 关于 Kafka 的信息在其/kafka 节点）。使用 Kafka 自带的消费者消费 ods_mall_data（Topic）中的数据，查看 Topic 中的前 1 条数据的结果，将查看命令与结果完整的截图粘贴至客户端桌面【Release\模块 A 提交结果.docx】中对应的任务序号下。

模块 B：实时数据处理（25 分）

环境说明：

服务端登录地址详见各模块服务端说明。

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)；
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；
建议使用 gson 解析 json 数据。

任务一：实时数据清洗

编写 Java 代码，使用 Flink 消费 Kafka 中 Topic 为 ods_mall_log 和 ods_mall_data 的数据并进行相应的数据统计计算(使用 Processing Time)。

- 1、使用 Flink 消费 Kafka 中 topic 为 ods_mall_data 的数据，根据数据中不同的表将数据分别分发至 kafka 的 DWD 层的 fact_order_master、fact_order_detail 的 Topic 中（只获取 data 的内容，具体的内容格式请自查，其分区数均为 2），其他的表则无需处理。使用 Kafka 自带的消费者消费 fact_order_master (Topic) 的前 1 条数据，将结果截图粘贴至客户端桌面【Release\模块 B 提交结果.docx】中对应的任务序号下；

fact_order_master 表结构，存储位置：Kafka，存储格式：json

字段	类型	中文含义（和 MySQL 中相同）	备注
order_id	int		
order_sn	string		
customer_id	int		
shipping_user	string		

province	string		
city	string		
address	string		
order_source	int		
payment_method	int		
order_money	double		
district_money	double		
shipping_money	double		
payment_money	double		
shipping_comp_name	string		
shipping_sn	string		
create_time	timestamp		
shipping_time	timestamp		
pay_time	timestamp		
receive_time	timestamp		
order_status	string		
order_point	int		
invoice_title	string		
modified_time	timestamp		

fact_order_detail 表结构，存储位置：Kafka，存储格式：json

字段	类型	中文含义（和 MySQL 中相同）	备注
order_detail_id	int		
order_sn	string		
product_id	int		
product_name	string		
product_cnt	int		
product_price	double		

average_cost	double		
weight	double		
fee_money	double		
w_id	int		
create_time	timestamp		
modified_time	timestamp		

2、使用 Flink 消费 Kafka 中 topic 为 ods_mall_log 的数据，根据数据中不同的表前缀区分，过滤出 product_browse 的数据，将数据分别分发至 kafka 的 DWD 层 log_product_browse 的 Topic 中，其分区数为 2，其他的表则无需处理。使用 Kafka 自带的消费者消费 log_product_browse (Topic) 的前 1 条数据，将结果截图粘贴至客户端桌面【Release\模块 B 提交结果.docx】中对应的任务序号下。

log_product_browse 表结构，存储位置：Kafka，存储格式：json

字段	类型	中文含义 (和 MySQL 中相同)	备注
log_id	long	自增长 id	可以使用随机数(0-9)+MMddHHmmssSSS 代替
product_id	string		
customer_id	int		
gen_order	int		
order_sn	string		
modified_time	timestamp		

3、在任务 1、2 进行的同时，需要将 order_master、order_detail、product_browse 备份至 HBase 中（若 Int 类型长度不够，可使用 BigInt 或

Long 类型代替），使用 HBase Shell 查看 ods:order_master 表的任意 2 条数据，查看字段为 row_key 与 shipping_user、查看 ods:order_detail 表的任意 2 条数据，查看字段为 row_key 与 product_name、查看 ods:product_browse 表的任意 2 条数据，查看字段为 row_key 与 order_sn。将结果分别截图粘贴至客户端桌面【Release\模块 B 提交结果.docx】中对应的任务序号下（截图中不能有乱码）。

三个 HBase 中的数据结构为：

ods:order_master 数据结构如下：

字段	类型	中文含义 (和 MySQL 中相同)	备注
rowkey	string	rowkey	可以使用随机数 (0-9)+yyyyMMddHHmmssSSS (date 的格式) 代替
Info		列族名	
order_id	int		
order_sn	string		
customer_id	int		
shipping_user	string		
province	string		
city	string		
address	string		
order_source	int		
payment_method	int		
order_money	double		
district_money	double		
shipping_money	double		
payment_money	double		

shipping_comp_name	string		
shipping_sn	string		
create_time	string		
shipping_time	string		
pay_time	string		
receive_time	string		
order_status	string		
order_point	int		
invoice_title	string		
modified_time	string		

ods:order_detail 数据结构如下:

字段	类型	中文含义 (和 MySQL 中 相同)	备注
rowkey	string	rowkey	可以使用随机数 (0-9) +yyyyMMddHHmmssSSS (date 的格式) 代替
Info		列族名	
order_detail_id	int		
order_sn	string		
product_id	int		
product_name	string		
product_cnt	int		
product_price	double		
average_cost	double		
weight	double		

fee_money	double		
w_id	int		
create_time	string		
modified_time	string		

ods:product_browse 数据结构如下:

字段	类型	中文含义 (和 MySQL 中 相同)	备注
rowkey	string	rowkey	该字段使用 logid 进行拆分, 将 log_id 拆分为随机数和 MMddHHmmssSSS 两块,在其 中插入 yyyy (date 的格式) 最终格式为: 随机数 (0-9) +yyyy +MMddHHmmssSSS
Info		列族名	
log_id	int		该字段缺失,使用随机数 (0-9) + MMddHHmmssSSS (date 的格式)
order_sn	string		
product_id	int		
customer_id	string		
gen_order	int		
modified_time	double		

模块 C：离线数据处理（30 分）

环境说明：

服务端登录地址详见各模块服务端说明。

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 的配置文件位于主节点/opt/module/hive-3.1.2/conf/
Spark 任务在 Yarn 上用 Client 运行，方便观察日志；
ClickHouse 的 jdbc 连接端口 8123,用户名/密码：default/123456,
命令行客户端（tcp）端口 9001；
建议使用 gson 解析 json 数据。

任务一：离线数据清洗

编写 Scala 工程代码，将 ods 库中表 order_master、order_detail、coupon_info、coupon_use、product_browse、product_info、customer_inf、customer_login_log、order_cart、customer_level_inf、customer_addr 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。

- 1、抽取 ods 库中表 customer_inf 最新分区数据，并结合 dim_customer_inf 最新分区现有的数据，根据 customer_id 合并数据到 dwd 库中 dim_customer_inf 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 customer_id 为合并字段，根据 modified_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

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 查询 modified_time 为 2022 年 10 月 01 日当天的数据，查询字段为 customer_id、customer_email、modified_time、dwd_insert_time、dwd_modify_time，并按照 customer_id 进行升序排序，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

- 2、抽取 ods 库中表 coupon_info 最新分区数据，并结合 dim_coupon_info 最新分区现有的数据，根据 coupon_id 合并数据到 dwd 库中 dim_coupon_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 coupon_id 为合并字段，根据 modified_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_coupon_info 命令，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

- 3、抽取 ods 库中表 product_info 最新分区的数据，并结合 dim_product_info 最新分区现有的数据，根据 product_core 合并数据到 dwd 库中 dim_product_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 product_core 为合并字段，根据 modified_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

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_product_info 命令，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

- 4、抽取 ods 库中表 order_master 最新分区的数据，并结合 HBase 中 order_master_offline 表中的数据合并抽取到 dwd 库中 fact_order_master 的分区表，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”，dwd_insert_time、dwd_modify_time 均填写当前操作时间（年月日必须是今天，时分秒只需在比赛时间范围内即可），抽取 HBase 中的数据时，只抽取 2022 年 10 月 01 日的数据（以 rowkey 为准），并进行数据类型转换。使用 hive cli 查询 modified_time 为 2022 年 10 月 01 日当天的数据，查询字段为 order_id、order_sn、shipping_user、create_time、shipping_time，并按照 order_id 进行升序排序，将结果截图复制粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

ods: order_master_offline 数据结构如下：

字段	类型	中文含义	备注
rowkey	string	rowkey	随机数（0-9） +yyyyMMddHHmmssSSS（date 的格式）
Info		列族名	
order_id	int		
order_sn	string		
customer_id	int		
shipping_user	string		

province	string		
city	string		
address	string		
order_source	int		
payment_method	int		
order_money	double		
district_money	double		
shipping_money	double		
payment_money	double		
shipping_comp_name	string		
shipping_sn	string		
create_time	string		
shipping_time	string		
pay_time	string		
receive_time	string		
order_status	string		
order_point	int		
invoice_title	string		
modified_time	string		

5、抽取 ods 库中表 order_detail 表最新分区的数据，并结合 HBase 中 order_detail_offline 表中的数据合并抽取到 dwd 库中 fact_order_detail 的分区表，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”，dwd_insert_time、dwd_modify_time 均填写当前操作时间（年月日必须是今天，时分秒只需在比赛时间范围内即可），抽取 HBase 中的数据时，只抽取 2022 年 10 月 01 日的数据（以 rowkey 为准），并进行数据类型转换。使用 hive cli 查询 modified_time 为 2022 年 10 月 01 日当天的数据，查询字段

为 order_detail_id、order_sn、product_name、create_time，并按照 order_detail_id 进行升序排序，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

ods:order_detail_offline 数据结构如下：

字段	类型	中文含义	备注
rowkey	string	rowkey	随机数（0-9） +yyyyMMddHHmmssSSS（date 的格式）
Info		列族名	
order_detail_id	int		
order_sn	string		
product_id	int		
product_name	string		
product_cnt	int		
product_price	double		
average_cost	double		
weight	double		
fee_money	double		
w_id	int		
create_time	string		
modified_time	string		

6、抽取 ods 库中表 coupon_use 最新分区的数据到 dwd 库中 fact_coupon_use 的分区表，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”，dwd_insert_time、dwd_modify_time 均填写当前操作时间（年月日必须是今天，时分秒只需在比赛时间范围内即可），并进行数据类型转换。使用 hive cli 执行 show partitions dwd.fact_coupon_use 命令，将结果截图粘贴至

客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

- 7、抽取 ods 库中表 customer_login_log 最新分区的数据到 dwd 库中 log_customer_login 的分区表，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”，dwd_insert_time、dwd_modify_time 均填写当前操作时间（年月日必须是今天，时分秒只需在比赛时间范围内即可），并进行数据类型转换。使用 hive cli 执行 show partitions dwd.log_customer_login 命令，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；
- 8、抽取 ods 库中表 order_cart 最新分区的数据到 dwd 库中 fact_order_cart 的分区表，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”，dwd_insert_time、dwd_modify_time 均填写当前操作时间（年月日必须是今天，时分秒只需在比赛时间范围内即可），并进行数据类型转换。使用 hive cli 执行 show partitions dwd.fact_order_cart 命令，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；
- 9、抽取 ods 库中表 product_browse 最新分区的数据，并结合 HBase 中 product_browse_offline 表中的数据合并抽取到 dwd 库中 log_product_browse 的分区表，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”，dwd_insert_time、dwd_modify_time 均填写当前操作时间（年月日必须是今天，时分秒只需在比赛时间范围内即可），抽取 HBase 中的数据时，只抽取 2022 年 10 月 01 日的数据（以 rowkey

为准），并进行数据类型转换。使用 hive cli 查询 modified_time 为 2022 年 10 月 01 日当天的数据，查询字段为 log_id、product_id、order_sn、modified_time，并按照 log_id 进行升序排序，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

ods:product_browse_offline 数据结构如下：

字段	类型	中文含义	备注
rowkey	string	rowkey	随机数（0-9） +MMddHHmmssSSS
Info		列族名	
log_id	int		
product_id	int		
customer_id	string		
gen_order	int		
order_sn	string		
modified_time	double		

10、抽取 ods 库中表 customer_level_inf 最新分区的数据到 dwd 库中 dim_customer_level_inf 的分区表，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”，dwd_insert_time、dwd_modify_time 均填写当前操作时间（年月日必须是今天，时分秒只需在比赛时间范围内即可），并进行数据类型转换。使用 hive cli 执行 show partitions dwd.dim_customer_level_inf 命令，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

11、抽取 ods 库中表 customer_addr 最新分区的数据到 dwd 库中 dim_customer_addr 的分区表，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”，dwd_insert_time、dwd_modify_time 均填写当前操作时间（年月日必须是今天，时分秒只需在比赛时间范围内即可），并进行数据类型转换。使用 hive cli 执行 show partitions dwd.dim_customer_addr 命令，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

12、 将 dwd 库中 dim_customer_inf 、 dim_customer_addr 、 dim_customer_level_inf 表的数据关联到 dws 库中 customer_addr_level_aggr 的分区表,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），并添加 dws_insert_user、dws_insert_time、dws_modify_user、dws_modify_time 四列，其中 dws_insert_user、dws_modify_user 均填写“user1”，dws_insert_time、dws_modify_time 均填写当前操作时间（年月日必须是今天，时分秒只需在比赛时间范围内即可），并进行数据类型转换。使用 hive cli 统计最新分区中得数据总量，将结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下。

任务二：离线指标计算

1、 编写 Scala 工程代码，根据 dwd 的订单表 dwd.fact_order_master，求各省份下单时间为 2022 年的支付转化率，并将计算结果按照下述表结构写入 clickhouse 的 ds_result 库的 payment_cvr 表。在 Linux 的 clickhouse 命令行中根据 ranking 字段查询出转化率前三的省份，将 SQL 语句与执行结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下；

注:支付转化率 = 完成支付的订单数 / 已下单数。

payment_cvr 表结构：

字段	类型	中文含义	备注
province	string	省份名	
creat_order	int	已下单数	

payment	int	已支付的订单数	
payCVR	float64	支付转化率	四舍五入保留三位小数
ranking	int	转化率排名	

2、编写 Scala 工程代码，根据 dwd 的登录日志表 `dwd.log_customer_login`，求 `login_time` 字段值为 2022-08-10 的最近连续三周登录的用户数，并将计算结果按照下述表结构写入 clickhouse 的 `ds_result` 库的 `continuous_3week` 表。然后在 Linux 的 clickhouse 命令行中根据 `active_total` 降序查询，将 SQL 语句与执行结果截图粘贴至客户端桌面【Release\模块 C 提交结果.docx】中对应的任务序号下。

`continuous_3week` 表结构：

字段	类型	中文含义	备注
end_date	string	数据统计日期	2022-08-10
active_total	int	活跃用户数	
date_range	string	统计周期	格式：统计开始时间_结束时间

`date_range`: 例：假设统计 2022 年 9 月 8 日的连续三周登录用户数，则该字段值应该为 2022-08-22_2022-09-11。

模块 D：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模块服务端说明。

任务一：用柱状图展示各省份消费额的中位数

编写 Vue 工程代码，根据接口，用柱状图展示 2020 年部分省份所有订单消费额的中位数（前 10 省份，降序排列，若有小数则四舍五入保留两位），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\模块 D 提交结果.docx】中对应的任务序号下。

任务二：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\模块 D 提交结果.docx】中对应的任务序号下。

模块 E：综合分析（10 分）

任务一：Kafka 的消费组如何实现消息的负载均衡？

请简述 Kafka 的消费组如何实现消息的负载均衡。将内容编写至客户端桌面【Release\模块 E 提交结果.docx】中对应的任务序号下。

任务二：Flink CDC 如何确保数据一致性？

请简述 Flink CDC 如何确保数据一致性，将内容编写至客户端桌面【Release\模块 E 提交结果.docx】中对应的任务序号下。

任务三：Hive 中如何优化查询性能？

请简述 Hive 中如何优化查询性能，将内容编写至客户端桌面【Release\模块 E 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）

```
21/06/17 14:24:59 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 4 blocks
21/06/17 14:24:59 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
21/06/17 14:24:59 INFO ShuffleBlockFetcherIterator: Getting 2 non-empty blocks out of 4 blocks
21/06/17 14:24:59 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
21/06/17 14:24:59 INFO Executor: Finished task 2.0 in stage 64.0 (TID 223). 2399 bytes result sent to driver
21/06/17 14:24:59 INFO Executor: Running task 5.0 in stage 64.0 (TID 220)
21/06/17 14:24:59 INFO Executor: Finished task 4.0 in stage 64.0 (TID 225). 2439 bytes result sent to driver
21/06/17 14:24:59 INFO TaskSetManager: Finished task 2.0 in stage 64.0 (TID 223) in 12 ms on localhost (2/6)
21/06/17 14:24:59 INFO TaskSetManager: Finished task 4.0 in stage 64.0 (TID 225) in 6 ms on localhost (3/6)
21/06/17 14:24:59 INFO Executor: Running task 3.0 in stage 64.0 (TID 224)
21/06/17 14:24:59 INFO ShuffleBlockFetcherIterator: Getting 0 non-empty blocks out of 4 blocks
21/06/17 14:24:59 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
21/06/17 14:24:59 INFO Executor: Finished task 5.0 in stage 64.0 (TID 220). 2280 bytes result sent to driver
21/06/17 14:24:59 INFO ShuffleBlockFetcherIterator: Getting 2 non-empty blocks out of 4 blocks
21/06/17 14:24:59 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
21/06/17 14:24:59 INFO TaskSetManager: Finished task 1.0 in stage 64.0 (TID 222) in 17 ms on localhost (4/6)
21/06/17 14:24:59 INFO Executor: Finished task 3.0 in stage 64.0 (TID 224). 2439 bytes result sent to driver
21/06/17 14:24:59 INFO TaskSetManager: Finished task 5.0 in stage 64.0 (TID 220) in 19 ms on localhost (5/6)
21/06/17 14:24:59 INFO TaskSetManager: Finished task 3.0 in stage 64.0 (TID 224) in 13 ms on localhost (6/6)
21/06/17 14:24:59 INFO TaskSchedulerImpl: Removed TaskSet 64.0, whose tasks have all completed, from pool
21/06/17 14:24:59 INFO DAGScheduler: ResultStage 64 (collect at rol11.scala:188) finished in 0.023 s
21/06/17 14:24:59 INFO DAGScheduler: Job 33 finished: collect at rol11.scala:188, took 0.028396 s
```

表结构说明

MySQL 数据库中已自带注释，自行连接使用工具查看。

若 IDEA 运行代码时候出现

scalac: No 'scala-library*.jar' in Scala compiler classpath in Scala SDK Maven:
org.scala-lang:scala-library:2.12.0

则需要先在 File->Project Structure->Project Settings->Libraries->添加 scala
的包（2.12 大版本一致即可）。