**环境说明：**

|  |
| --- |
| 集群启动命令为在Master上执行allstart.sh；  各主机可通过SSH客户端进行SSH访问（需通过22映射的宿主机端口号访问）；  Master节点MySQL数据库用户名/密码：root/123456（已配置远程连接，需通过3306映射的宿主机端口号连接）；  Hive的配置文件位于/opt/module/hive-3.1.2/conf/  Spark任务在Yarn上用Client运行，方便观察日志。 |

## 离线数据处理

### 任务一：数据抽取

编写Scala代码，使用Spark将MySQL的shtd\_industry库中表EnvironmentData，ChangeRecord，BaseMachine，MachineData,ProduceRecord全量抽取到Hive的ods库（需自建）中对应表environmentdata，changerecord，basemachine， machinedata， producerecord中。

1. 抽取MySQL的shtd\_industry库中EnvironmentData表的全量数据进入Hive的ods库中表environmentdata，字段排序、类型不变，同时添加静态分区，分区字段类型为String，且值为当前日期的前一天日期（分区字段格式为yyyyMMdd）。并在hive cli执行show partitions ods.environmentdata命令，将结果截图粘贴至对应报告中；
2. 抽取MySQL的shtd\_industry库中ChangeRecord表的全量数据进入Hive的ods库中表changerecord，字段排序、类型不变，同时添加静态分区，分区字段类型为String，且值为当前日期的前一天日期（分区字段格式为yyyyMMdd）。并在hive cli执行show partitions ods.changerecord命令，将结果截图粘贴至对应报告中；
3. 抽取MySQL的shtd\_industry库中BaseMachine表的全量数据进入Hive的ods库中表basemachine，字段排序、类型不变，同时添加静态分区，分区字段类型为String，且值为当前日期的前一天日期（分区字段格式为yyyyMMdd）。并在hive cli执行show partitions ods.basemachine命令，将结果截图粘贴至对应报告中；
4. 抽取MySQL的shtd\_industry库中ProduceRecord表的全量数据进入Hive的ods库中表producerecord，字段排序、类型不变，同时添加静态分区，分区字段类型为String，且值为当前日期的前一天日期（分区字段格式为yyyyMMdd）。并在hive cli执行show partitions ods.producerecord命令，将结果截图粘贴至对应报告中；
5. 抽取MySQL的shtd\_industry库中MachineData表的全量数据进入Hive的ods库中表machinedata，字段排序、类型不变，同时添加静态分区，分区字段类型为String，且值为当前日期的前一天日期（分区字段格式为yyyyMMdd）。并在hive cli执行show partitions ods.machinedata命令，将结果截图粘贴至对应报告中。

### 任务二：数据清洗

编写Hive SQL代码，将ods库中相应表数据全量抽取到Hive的dwd库（需自建）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environmentdata的全量数据进入Hive的dwd库中表fact\_environment\_data，分区字段为etldate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，并进行数据类型转换。并在hive cli按照envoid降序排序，查询前5条数据，将结果内容截图粘贴至对应的报告中；
2. 抽取ods库中changerecord的全量数据进入Hive的dwd库中表fact\_change\_record，抽取数据之前需要对数据根据changeid和changemachineid进行联合去重处理，分区字段为etldate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，并进行数据类型转换。并在hive cli按照change\_machine\_id降序排序，查询前1条数据，将结果内容截图粘贴至对应报告中；
3. 抽取ods库中basemachine的全量数据进入Hive的dwd库中表dim\_machine,抽取数据之前需要对数据根据basemachineid进行去重处理。分区字段为etldate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，并进行数据类型转换。在hive cli中按照base\_machine\_id顺序排序，查询dim\_machine前2条数据，将结果内容截图粘贴至对应报告中；
4. 抽取ods库中producerecord的全量数据进入Hive的dwd库中表fact\_produce\_record,分区字段为etldate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，并进行数据类型转换。在hive cli中按照produce\_machine\_id顺序排序，查询fact\_produce\_record前2条数据，将结果内容截图粘贴至对应报告中；
5. 抽取ods库中machinedata的全量数据进入Hive的dwd库中表fact\_machine\_data。分区字段为etldate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，并进行数据类型转换。并在hive cli中按照machine\_id降序排序，查询前1条数据，将结果内容截图粘贴至对应报告中。

### 任务三：指标计算

1. 编写Scala代码，使用Spark根据dwd层fact\_change\_record表统计每个月（change\_start\_time的月份）、每个设备、每种状态的时长，若某状态当前未结束（即change\_end\_time值为空）则该状态不参与计算。计算结果存入MySQL数据库shtd\_industry的machine\_state\_time表（表结构如下）中，然后在Linux的MySQL命令行中根据设备id、状态持续时长均为逆序排序，查询出前10条，将SQL语句与执行结果截图粘贴至对应报告中;

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_id | int | 设备id |  |
| change\_record\_state | varchar | 状态 |  |
| duration\_time | varchar | 持续时长（秒） | 当月该状态的时长和 |
| year | int | 年 | 状态产生的年 |
| month | int | 月 | 状态产生的月 |

1. 编写Scala代码，使用Spark根据dwd层fact\_change\_record表关联dim\_machine表统计每个车间中所有设备运行时长（即设备状态为“运行”）的中位数在哪个设备（为偶数时，两条数据原样保留输出），若某个设备运行状态当前未结束（即change\_end\_time值为空）则该状态不参与计算。计算结果存入clickhouse数据库shtd\_industry的表machine\_running\_median（表结构如下）中，然后在Linux的clickhouse命令行中根据所属车间、设备id均为倒序排序，查询出前10条数据，将SQL语句与执行结果截图粘贴至对应报告中;

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_id | int | 设备id |  |
| machine\_factory | int | 所属车间 |  |
| total\_running\_time | int | 运行总时长 | 结果以秒为单位 |

1. 编写scala代码，使用Spark根据dwd层dwd.fact\_produce\_record表，基于全量历史增加设备生产一个产品的平均耗时字段（produce\_per\_avgtime），produce\_code\_end\_time值为1900-01-01 00:00:00的数据为脏数据，需要剔除，并以produce\_record\_id和ProduceMachineID为联合主键进行去重（注：fact\_produce\_record表中，一条数据代表加工一个产品，produce\_code\_start\_time字段为开始加工时间，produce\_code\_end\_time字段为完成加工时间）。将得到的数据提取下表所需字段然后写入dws层表dws.machine\_produce\_per\_avgtime表中，然后在Linux的命令行中根据设备id倒序排序查询前3条数据，将SQL语句与执行结果截图粘贴至对应报告中；

dws.machine\_produce\_per\_avgtime：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| produce\_record\_id | int | 生产记录id | 每生产一件产品产生一条数据 |
| produce\_machine\_id | int | 设备id |  |
| producetime | int | 该产品耗时 |  |
| produce\_per\_avgtime | int | 设备生产一个产品平均耗时 | 单位：秒 |

1. 编写Scala代码，使用Spark根据dwd层dwd.fact\_produce\_record表，基于全量历史数据计算各设备生产一个产品的平均耗时，produce\_code\_end\_time值为1900-01-01 00:00:00的数据为脏数据，需要剔除，并以produce\_record\_id和produce\_machine\_id为联合主键进行去重（注：fact\_produce\_record表中，一条数据代表加工一个产品，produce\_code\_start\_time字段为开始加工时间，produce\_code\_end\_time字段为完成加工时间）。将设备每个产品的耗时与该设备平均耗时作比较，保留耗时高于平均值的产品数据。将得到的数据写入clickhouse表machine\_produce\_per\_avgtime表中，然后在Linux的clickhouse命令行中根据设备id倒序排序查询前3条数据，将SQL语句与执行结果截图粘贴至对应报告中；

machine\_produce\_per\_avgtime：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| produce\_record\_id | int | 生产记录id | 每生产一件产品产生一条数据 |
| produce\_machine\_id | int | 设备id |  |
| producetime | int | 该产品耗时 |  |
| produce\_per\_avgtime | int | 设备生产一个产品平均耗时 | 单位：秒 |

1. 编写scala代码，使用Spark根据dws层的dws.machine\_produce\_per\_avgtime表,获取各设备生产耗时最长的两个产品的用时。将计算结果存入clickhouse数据库shtd\_industry的表machine\_produce\_timetop2中，然后在Linux的clickhouse命令行中根据设备id倒序排序，查询出前2条，将SQL语句与执行结果截图粘贴至对应报告中。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_id | int | 设备id |  |
| first\_time | int | 耗时最长 |  |
| second\_time | int | 耗时次长 |  |

1. 编写Hive SQL代码，根据dwd层dwd.fact\_environment\_data表，统计检测设备（BaseID）每月的平均湿度（Humidity），然后将每个设备的每月平均湿度与厂内所有检测设备每月检测结果的平均湿度做比较（结果值为：高/低/相同）存入MySQL数据库shtd\_industry的表machine\_humidityAVG\_compare（表结构如下）中，然后在Linux的MySQL命令行中根据检测设备ID倒序排序，查询出前5条，将SQL语句与执行结果截图粘贴至对应报告中。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| base\_id | int | 检测设备ID |  |
| machine\_avg | varchar | 单设备检测平均值 |  |
| factory\_avg | varchar | 厂内所有设备平均值 |  |
| comparison | varchar | 比较结果 | 高/低/相同 |
| env\_date\_year | varchar | 检测年份 | 如：2021 |
| env\_date\_month | varchar | 检测月份 | 如：12 |

1. 编写Scala代码，使用Spark根据dwd层fact\_change\_record表和dim\_machine表统计，计算每个车间设备的月平均运行时长与所有设备的月平均运行时长对比结果（即设备状态为“运行”，结果值为：高/低/相同），月份取值使用状态开始时间的月份，若某设备的运行状态当前未结束（即change\_end\_time值为空）则该状态不参与计算。计算结果存入clickhouse数据库shtd\_store的表machine\_running\_compare（表结构如下）中，然后在Linux的clickhouse命令行中根据车间号倒序排序，查询出前2条，将SQL语句与执行结果截图粘贴至对应报告中;

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| start\_month | String | 月份 | 如：2021-12 |
| machine\_factory | int | 车间号 |  |
| comparison | String | 比较结果 | 高/低/相同 |
| factory\_avg | String | 车间平均时长 |  |
| company\_avg | String | 所有设备平均时长 |  |

1. 编写Scala代码，使用Spark根据dwd层fact\_change\_record表展示每一个设备最近第二次的状态（倒数第二次），时间字段选用change\_start\_time，如果设备仅有一种状态，返回该状态（一个设备不会同时拥有两种状态），存入clickhouse数据库shtd\_store的表recent\_state（表结构如下）中，然后在Linux的clickhouse命令行中根据设备id进行倒序排序，查询出前5条，将SQL语句与执行结果截图粘贴至对应报告中。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_id | int | 设备id |  |
| record\_state | String | 状态信息 | 最近第二次的状态 |
| change\_start\_time | String | 状态开始时间 |  |
| change\_end\_time | String | 状态结束时间 |  |

1. 编写Scala代码，使用Spark根据dwd层dwd.fact\_environment\_data表，统计检测设备（baseid）每月的PM10的检测平均浓度，然后将每个设备的每月平均浓度与厂内所有检测设备每月检测结果的平均浓度做比较（结果值为：高/低/相同）。计算结果存入MySQL数据库shtd\_industry的表machine\_runningAVG\_compare（表结构如下）中，然后在Linux的MySQL命令行中根据检测设备ID倒序排序，查询出前5条，将SQL语句与执行结果截图粘贴至对应报告中。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| base\_id | int | 检测设备ID |  |
| machine\_avg | varchar | 单设备检测平均值 |  |
| factory\_avg | varchar | 厂内所有设备平均值 |  |
| comparison | varchar | 比较结果 | 高/低/相同 |
| env\_date\_year | varchar | 检测年份 | 如：2021 |
| env\_date\_month | varchar | 检测月份 | 如：12 |

1. 编写scala代码，使用Spark根据dwd层dwd.fact\_machine\_data表统计出每日每台设备，状态为“运行”的时长（若运行无结束时间，则需根据时间判断这个设备的运行状态的下一个状态是哪条数据，将下一条数据的状态开始的时间置为这个设备运行状态的结束时间。如果某个设备数据的运行状态不存在下一个状态，则该设备数据的运行状态不参与计算，即该设备的这条数据的运行状态时长按0计算），将结果数据写入dws层表machine\_data\_total\_time，然后在Linux的hive cli命令行中根据machine\_id倒序和machine\_record\_date正序排序查询前5条数据，将SQL语句与执行结果截图粘贴至对应报告中；

dws.machine\_data\_total\_time：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_id | int | 设备id |  |
| machine\_record\_date | string | 状态日期 | 如：2021-10-01 |
| total\_time | int | 一天运行总时长 | 秒 |

1. 编写scala代码，使用Spark根据dws层表machine\_data\_total\_time，计算每日运行时长前三的设备（若存在运行时长相同的数据时应全部输出，例如有两条并列第二，则第三名次不变，总共输出四条数据）。将计算结果写入clickhouse的machine\_data\_total\_time\_top3表（表结构如下）中。然后在Linux的clickhouse命令行中根据查询所有数据，将SQL语句与执行结果截图粘贴至对应报告中；

machine\_data\_total\_time\_top3：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| day | String | 日期 | 如：2021-10-01 |
| first\_id | int | 第一 | 一天运行总时长第一 |
| second\_id | int | 第二 | 一天运行总时长第二 |
| tertiary\_id | int | 第三 | 一天运行总时长第二 |
| first\_time | int | 第一的时长 | 秒 |
| second\_time | int | 第二的时长 | 秒 |
| tertiary\_time | int | 第三的时长 | 秒 |

1. 根据dwd层的fact\_change\_record表，计算状态开始时间为2021年10月12日凌晨0点0分0秒且状态为“待机”的数据，计算当天每小时新增的“待机”状态时长与当天“待机”状态总时长，将结果存入MySQL数据库shtd\_industry的accumulate\_standby表中（表结构如下），然后在Linux的MySQL的命令行根据start\_hour降序查询accumulate\_standby表前3条数据，将SQL语句与执行结果截图粘贴至对应报告中；

accumulate\_standby表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| start\_date | varchar | 状态开始日期 | yyyy-MM-dd |
| start\_hour | varchar | 状态开始时间 | 存当前时间的小时整数，如：1点存1、  13点存13 |
| hour\_add\_standby | int | 该小时新增“待机”状态时长 |  |
| day\_agg\_standby | int | 截止到这个小时，当天“待机”状态总时长 |  |

1. 根据dwd层的fact\_change\_record表，计算状态开始时间为2021年10月13日凌晨0点0分0秒且状态为“运行”的数据，以3小时为一个计算窗口，做滑动窗口计算，滑动步长为1小时，窗口不满3小时不触发计算（即从凌晨3点开始触发计算，第一条数据的state\_start\_time为2021-10-13\_02，以有数据的3个小时为一个时间窗口），将计算结果存入MySQL数据库shtd\_industry的slide\_window\_runnning表中，然后在Linux的MySQL命令行中按照state\_start\_time字段所带的小时信息进行升序排序查询slide\_window\_runnning表前3条数据，将核心业务代码中的开窗相关代码与MySQL查询结果分别截图粘贴至对应报告中。

slide\_window\_runnning表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| state\_start\_time | varchar | 运行状态开始时间 | yyyy-MM-dd\_HH |
| window\_sum | int | 3小时运行总时长 | 单位：秒 |
| window\_avg | int | 每小时平均运行时长 | 基于所在窗口 |

## 数据挖掘

### 任务一：特征工程

1. 根据dwd库中fact\_machine\_data表（或MySQL的shtd\_industry库中MachineData表），根据以下要求转换：获取最大分区（MySQL不用考虑）的数据后，首先解析列machine\_record\_data（MySQL中为MachineRecordData）的数据（数据格式为xml，采用dom4j解析，会给出解析demo），并获取每条数据的主轴转速，主轴倍率，主轴负载，进给倍率，进给速度，PMC程序号，循环时间，运行时间，有效轴数，总加工个数，已使用内存，未使用内存，可用程序量，注册程序量等相关的值（若该条数据没有相关值，则按下表设置默认值），同时转换machine\_record\_state字段的值，若值为报警，则填写1，否则填写0，以下为表结构，将数据保存在dwd.fact\_machine\_learning\_data，在hive cli中按照machine\_record\_id顺序排序，查询dwd.fact\_machine\_learning\_data前1条数据，将结果内容复制粘贴至对应报告中；

dwd.fact\_machine\_learning\_data表结构：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_record\_id | int | 自增长id |  |
| machine\_id | double | 机器id |  |
| machine\_record\_state | double | 机器状态 | 默认0.0 |
| machine\_record\_mainshaft\_speed | double | 主轴转速 | 默认0.0 |
| machine\_record\_mainshaft\_multiplerate | double | 主轴倍率 | 默认0.0 |
| machine\_record\_mainshaft\_load | double | 主轴负载 | 默认0.0 |
| machine\_record\_feed\_speed | double | 进给倍率 | 默认0.0 |
| machine\_record\_feed\_multiplerate | double | 进给速度 | 默认0.0 |
| machine\_record\_pmc\_code | double | PMC程序号 | 默认0.0 |
| machine\_record\_circle\_time | double | 循环时间 | 默认0.0 |
| machine\_record\_run\_time | double | 运行时间 | 默认0.0 |
| machine\_record\_effective\_shaft | double | 有效轴数 | 默认0.0 |
| machine\_record\_amount\_process | double | 总加工个数 | 默认0.0 |
| machine\_record\_use\_memory | double | 已使用内存 | 默认0.0 |
| machine\_record\_free\_memory | double | 未使用内存 | 默认0.0 |
| machine\_record\_amount\_use\_code | double | 可用程序量 | 默认0.0 |
| machine\_record\_amount\_free\_code | double | 注册程序量 | 默认0.0 |
| machine\_record\_date | timestamp | 记录日期 |  |
| dwd\_insert\_user | string |  |  |
| dwd\_insert\_time | timestamp |  |  |
| dwd\_modify\_user | string |  |  |
| dwd\_modify\_time | timestamp |  |  |

### 任务二：报警预测

1. 根据任务一的结果，建立随机森林（随机森林相关参数可自定义，不做限制），使用任务一的结果训练随机森林模型，然后再将hive中dwd.fact\_machine\_learning\_data\_test（该表字段含义与dwd.fact\_machine\_learning\_data表相同，machine\_record\_state列值为空，表结构自行查看）转成向量，预测其是否报警将结果输出到MySQL库shtd\_industry中的ml\_result表中。在Linux的MySQL命令行中查询出machine\_record\_id为1、8、20、28和36的5条数据，将SQL语句与执行结果截图粘贴至对应报告中。

ml\_result表结构：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 中文含义 | 备注 |
| machine\_record\_id | int | 主键 |  |
| machine\_record\_state | double | 设备状态 | 报警为1，其他状态则为0 |

## 数据采集与实时计算

### 任务一：实时数据采集

1. 在Master节点使用Flume采集/data\_log目录下实时日志文件中的数据（实时日志文件生成方式为在/data\_log目录下执行./make\_data\_file\_v1命令即可生成日志文件，如果没有权限，请执行授权命令chmod 777 /data\_log/make\_data\_file\_v1），将数据存入到Kafka的Topic中（Topic名称分别为ChangeRecord、ProduceRecord和EnvironmentData，分区数为4），将Flume采集ChangeRecord主题的配置截图粘贴至对应报告中；
2. 编写新的Flume配置文件，将数据备份到HDFS目录/user/test/flumebackup下，要求所有主题的数据使用同一个Flume配置文件完成，将Flume的配置截图粘贴至对应报告中。

### 任务二：使用Flink处理Kafka中的数据

编写Scala工程代码，使用Flink消费Kafka中的数据并进行相应的数据统计计算。

1. 使用Flink消费Kafka中ProduceRecord主题的数据，统计在已经检验的产品中，各设备每五分钟生产产品总数，将结果存入Redis中，key值为“totalproduce”，value值为“设备id，最近五分钟生产总数”。使用redis cli以HGETALL key方式获取totalproduce值，将结果截图粘贴至对应报告中，需两次截图，第一次截图和第二次截图间隔五分钟以上，第一次截图放前面，第二次放后面；

注：ProduceRecord主题，生产一个产品产生一条数据；

change\_handle\_state字段为1代表已经检验，0代表未检验；

时间语义使用Processing Time。

1. 使用Flink消费Kafka中ChangeRecord主题的数据，当某设备30秒状态连续为“预警”，输出预警信息。当前预警信息输出后，最近30秒不再重复预警（即如果连续1分钟状态都为“预警”只输出两次预警信息）。将结果存入Redis中，key值为“warning30sMachine”，value值为“设备id，预警信息”。使用redis cli以HGETALL key方式获取warning30sMachine值，将结果截图粘贴至对应报告中，需两次截图，第一次截图和第二次截图间隔一分钟以上，第一次截图放前面，第二次放后面；

注：时间使用change\_start\_time字段，忽略数据中的change\_end\_time不参与任何计算。忽略数据迟到问题。

Redis的value示例：115,2022-01-01 09:53:10:设备115 连续30秒为预警状态请尽快处理！

(2022-01-01 09:53:10 为change\_start\_time字段值，中文内容及格式必须为示例所示内容。)

1. 使用Flink消费Kafka中ChangeRecord主题的数据，统计每三分钟各设备状态为“预警”且未处理的数据总数。将结果存入MySQL的shtd\_industry.threemin\_warning\_state\_agg表（追加写入），表结构如下，同时备份到Hbase一份，表结构同MySQL表的。请在将任务启动命令截图，启动且数据进入后按照设备id倒序排序查询threemin\_warning\_state\_agg表进行截图，第一次截图后等待三分钟再次查询并截图,将结果截图粘贴至对应报告中。

threemin\_warning\_state\_agg表：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 中文含义 |
| change\_machine\_id | int | 设备id |
| totalwarning | int | 未被处理预警的数据总数 |
| window\_end\_time | varchar | 窗口结束时间（yyyy-MM-dd HH:mm:ss） |

注：时间语义使用Processing Time。

1. 使用Flink消费Kafka中ChangeRecord主题的数据，实时统计每个设备从其他状态转变为“运行”状态的总次数。将结果存入MySQL的shtd\_industry.change\_state\_other\_to\_run\_agg表，表结构如下，同时备份到Hbase一份，表结构同MySQL表的。请在将任务启动命令截图，启动一分钟后根据change\_machine\_id降序查询change\_state\_other\_to\_run\_agg表并截图，启动两分钟后根据change\_machine\_id降序查询change\_state\_other\_to\_run\_agg表并再次截图；

注：时间语义使用Processing Time。

change\_state\_other\_to\_run\_agg表：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 中文含义 |
| change\_machine\_id | int | 设备id |
| last\_machine\_state | varchar | 上一状态。即触发本次统计的最近一次非运行状态 |
| total\_change\_torun | int | 从其他状态转为运行的总次数 |
| in\_time | varchar | flink计算完成时间（yyyy-MM-dd HH:mm:ss） |

1. 使用Flink消费Kafka中ChangeRecord主题的数据，每隔一分钟输出最近三分钟的预警次数最多的设备。将结果存入Redis中，key值为“warning\_last3min\_everymin\_out”，value值为“窗口结束时间，设备id”（窗口结束时间格式：yyyy-MM-dd HH:mm:ss）。使用redis cli以HGETALL key方式获取warning\_last3min\_everymin\_out值，将结果截图粘贴至对应报告中，需两次截图，第一次截图和第二次截图间隔一分钟以上，第一次截图放前面，第二次放后面；

注：时间语义使用Processing Time。

1. 使用Flink消费Kafka中EnvironmentData主题的数据,监控各环境检测设备数据，当温度（Temperature字段）持续3分钟高于38度时记录为预警数据。将结果存入Redis中，key值为“env\_temperature\_monitor”，value值为“设备id-预警信息生成时间，预警信息”（预警信息生成时间格式：yyyy-MM-dd HH:mm:ss）。使用redis cli以HGETALL key方式获取env\_temperature\_monitor值，将结果截图粘贴至对应报告中，需要两次截图间隔三分钟以上，第一次截图放前面，第二次放后面。

注：时间语义使用Processing Time。

value示例：114-2022-01-01 14:12:19，设备114连续三分钟温度高于38度请及时处理！

中文内容及格式必须为示例所示内容。

同一设备3分钟只预警一次。