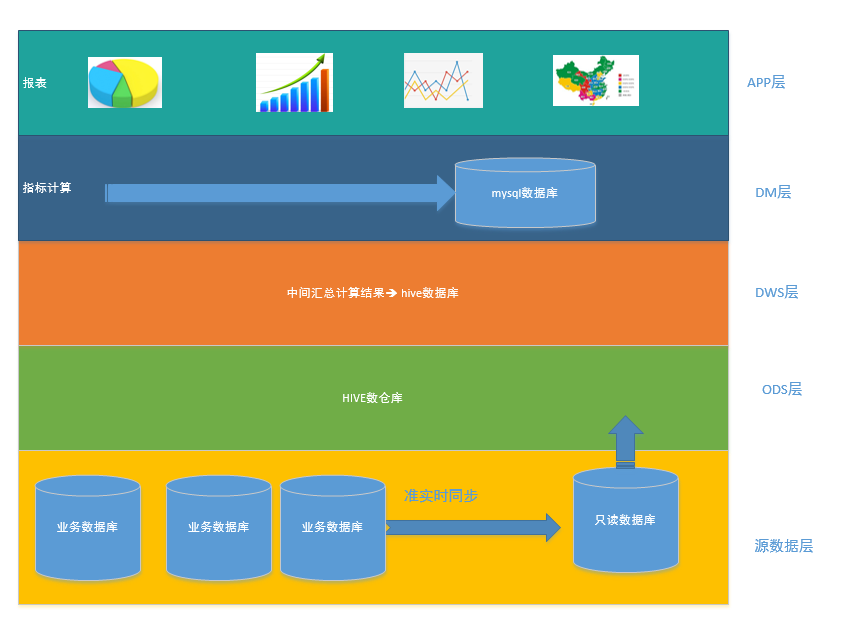
MT高效运营支撑平台

# 项目描述

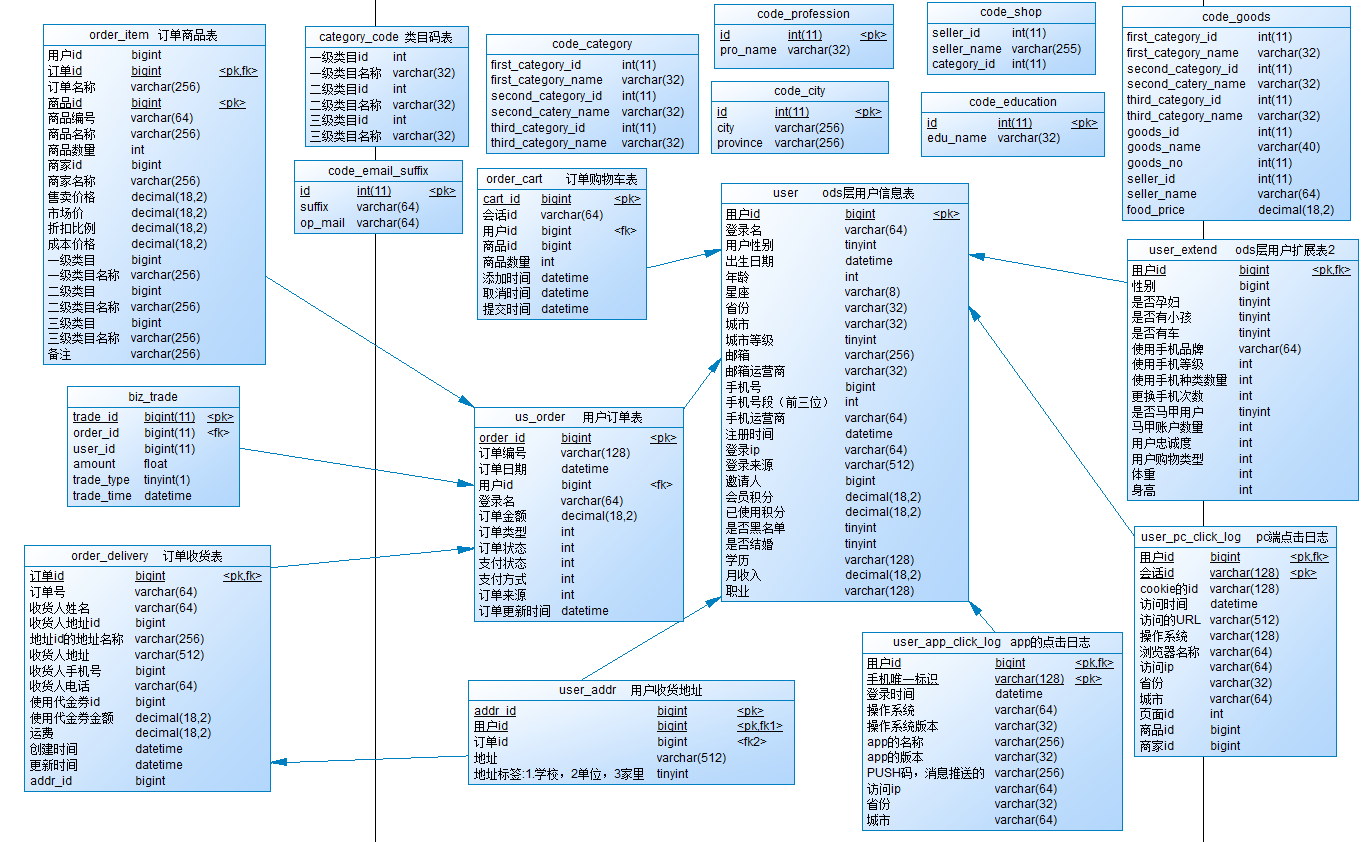
本平台是外卖行业类的自营业务的运营分析平台的一部分,通过对业务数据的分析,获得用户主题和订单的相关分析情况,生成用户的访问模型和用户的订单行为模型,对访问时段和下单区域分布进行分析和报表展示.

# 业务系统说明

用户使用浏览器(PC端)和app对外卖平台进行访问,记录用户的注册信息,通过调研平台和数据挖掘记录用户的扩展信息,业务系统对用户的每次点击和下单信息记录到数据库中,运营平台通过sqoop将数据抽取到运营分析平台,运营分析平台进行指标的计算,将结果存入到mysql中,将结果展示到报表.具体架构如下



业务表数据模型



# 需求分析

## 用户访问模型

1. 生成用户基本宽表user\_basic

将用户的基本信息表和用户的扩展表进行合并形成用户宽表.

包括信息如下:

用户id,登录名 ,用户性别,出生日期,年龄,星座,省份,城市 ,城市等级,邮箱 ,邮箱运营商 ,手机号 ,手机号段（前三位）,手机运营商,注册时间,登录ip,登录来源,邀请人,会员积分,已使用积分,会员等级名称,是否黑名单,是否结婚,学历,月收入 ,职业,是否孕妇,是否有小孩,是否有车,使用手机品牌,使用手机等级,使用手机种类数量,更换手机数量 ,是否马甲用户,马甲账户数量,用户忠诚度,用户购物类型,体重,身高。

1. 根据用户的访问日志信息,统计以下指标,结果存入用户访问模型表user\_visit

指标如下:

PC端指标

* + - * 最近一次访问时间
      * 最近一次访问使用的session
      * 最近一次使用的coookie
      * 最近一次的pc端的pv量
      * 最近一次访问使用的浏览器
      * 最近一次访问使用的操作系统
      * 第一次pc端访问的日期
      * 第一次pc端访问的session
      * 第一次pc端访问的cookie
      * 第一次访问的pv
      * 第一次访问使用的浏览器
      * 第一次访问的os
      * PC连续7天访问次数(跑任务的日期的前7天,以下相同不再赘述)
      * 连续15天访问次数
      * 连续30天访问次数
      * 连续60天访问的次数
      * 连续90天访问的次数
      * 近30天pc端访问的次数
      * 近30天pc端的pv
      * 近30天pc端每天的平均pv
      * 近30天的0到5点的pv数量
      * 近30天的6到7点的pv数量
      * 近30天的8到9的pv数量
      * 近30天的10到11的pv数量
      * 近30天的12到13的pv数量
      * 近30天的14到16点的pv数量
      * 近30天的17到19点的pv数量
      * 近30天的18到19点的pv数量
      * 近30天的20到21点的pv数量
      * 近30天的22到23点的pv数量
      * 近30天访问使用的不同ip数量
      * 近30天最常用的ip
      * 近30天使用的cookie的数量
      * 近30使用最常用的cookie\_id
      * 近30pc最常用浏览器
      * 近30天使用最常用系统

APP端指标

* + - * 最近一次app访问的日期
      * 最近一次访问app的名称
      * 最近一次app访问的操作系统
      * 第一次app访问日期
      * 第一app访问名称
      * 第一次app访问os
      * app第一次访问ip
      * app 近7天访问pv数
      * app 近15天访问pv数
      * app 近30天的访问pv数
      * app近60天的访问pv数
      * app近90天的访问pv数
      * app近30天0到5点的访问pv数
      * app近30天的6到7点的访问pv数
      * app近30天8到9的访问pv数
      * app近30天10到11访问pv数
      * app近30天12到13点的访问pv数
      * app近30天14到15点的访问pv数
      * app近30天16到17点的访问pv数
      * app近30天18到19点的访问pv数
      * app近30天20到21点的访问pv数
      * app近30天22到23点的访问pv数

综合指标

* 最近一次访问的ip
* 最近一次访问的城市
* 最近一次访问的省份
* 第一次访问的ip
* 第一次访问的城市
* 第一次访问的省份

## 用户订单模型

订单指标

* 第一次下单时间,
* 最近一次下单时间,
* 首单距今时间,
* 尾单距今时间,
* 近30天订单数量（不含退拒）,
* 近30天订单金额（不含退拒）,
* 近60天订单数量（不含退拒）,
* 近60天订单金额（不含退拒）,
* 近90天订单数量（不含退拒）,
* 近90天订单金额（不含退拒）,
* 近30天订单数量（含退拒）,
* 近30天订单金额（含退拒）,
* 近60天购买次数（含退拒）,
* 近60天购买金额（含退拒）,
* 近90天购买次数（含退拒）,
* 近90天购买金额（含退拒）,
* 近90天的客单价（含退拒）,
* 最大消费金额 ,
* 最小消费金额,
* 累计消费次数（不含退拒）,
* 累计消费金额（不含退拒）,
* 客单价（含退拒）,
* 最常用收货地址,
* 最常用支付方式
* 退货商品数量,
* 退货商品金额,
* 拒收商品数量,
* 拒收商品金额,
* 最近一次退货时间

下单分布：

* 学校下单总数,
* 公司下单总数,
* 家里下单总数,
* 凌晨下单总数,
* 上午下单总数,
* 中午下单总数，
* 下午下单总数,
* 晚上下单总数,
* 指标口径:凌晨:0-05 上午:06-12 中午:13-15 下午16-20 晚上：21-24

## 访问情况时段统计图

将指定的日期的每个时间区间的访问统计情况进行图形展示,横坐标是时间点,纵坐标是访问次数

## 当天访问用户top20

以柱状图的形式显示当天访问用户的top20,横坐标为用户登录名,纵坐标为pv量

## 订单情况时段统计图

将指定的日期的每个时间区间的订单统计情况进行图形展示,横坐标是时间点,纵坐标是下单数量

## 当天区域下单量的分布图

以地图的方式显示当天下单金额的各省份的分布图.

# 技术选型

数据导出:sqoop

数据计算:Spark SQL

数据仓库:Hive

任务调度:Azkaban

# 千峰运营平台模型设计

## ODS层 源数据层

映射关系

|  |  |  |
| --- | --- | --- |
| 源表(mysql) | 目标表(hive) | 映射关系 |
| user | qfbap\_ods.ods\_user |  |
| user\_extend | qfbap\_ods.ods\_user\_extend |  |
| user\_pc\_click\_log | qfbap\_ods.ods\_user\_pc\_click |  |
| user\_app\_click\_log | qfbap\_ods.ods\_user\_app\_click\_log |  |
| user\_addr | qfbap\_ods.ods\_user\_addr |  |
| user\_order | qfbap\_ods.ods\_us\_order |  |
| order\_item | qfbap\_ods.ods\_order\_item |  |
| order\_delivery | qfbap\_ods.ods\_order\_delivery |  |
| cart | qfbap\_ods.ods\_cart |  |
| biz\_trade | qfbap\_ods.ods\_biz\_trade |  |
| code\_category | qfbap\_ods.ods\_code\_category |  |

## DWD层

|  |  |  |
| --- | --- | --- |
| 源表(hive) | 目标表(hive) | 映射关系 |
| qfbap\_ods.ods\_user | qfbap\_dwd.dwd\_qfbap\_user |  |
| qfbap\_ods.ods\_user\_extend | qfbap\_dwd.dwd\_user\_extend |  |
| qfbap\_ods.ods\_user\_pc\_click\_log | qfbap\_dwd.dwd\_user\_pc\_pv | 创建表给你们，需求就是取停留时长,如果最大的访问时间跟最小的是一样的,那么就按3秒算,如果是不一样的,就是最大的时间-最小的时间获得的秒数 |
| qfbap\_ods.ods\_user\_app\_click\_log | qfbap\_dwd.dwd\_user\_app\_pv | 增加log\_hour,对hour(log\_time)🡪 log\_hour |
| qfbap\_ods.ods\_user\_addr | qfbap\_dwd.dwd\_user\_addr |  |
| qfbap\_ods.ods\_user\_order | qfbap\_dwd.dwd\_user\_order |  |
| qfbap\_ods.ods\_order\_item | qfbap\_dwd.dwd\_order\_item |  |
| qfbap\_ods.ods\_order\_delivery | qfbap\_dwd.dwd\_order\_delivery |  |
| qfbap\_ods.ods\_order\_cart | qfbap\_dwd.dwd\_order\_cart |  |
| qfbap\_ods.ods\_biz\_trade | qfbap\_dwd.dwd\_biz\_trade |  |
| qfbap\_ods.ods\_code\_category | qfbap\_dwd.dwd\_code\_category |  |

## DWS层

|  |  |  |
| --- | --- | --- |
| 源表(hive) | 目标表(hive) | 映射关系 |
| qfbap\_dwd.dwd\_user\_pc\_pv | qfbap\_dws.dws\_user\_visit\_month1 |  |
|  |  |  |

## DM层

|  |  |  |
| --- | --- | --- |
| 源表(hive) | 目标表(mysql) | 映射关系 |
| qfbap\_dwd.dwd\_user  qfbap\_dwd.dwd\_user\_extend | dm\_user\_basic |  |
| dws\_user\_basic  qfbap\_dwd.dwd\_user\_app\_pv  qfbap\_dwd.dwd\_user\_pc\_pv  qfbap\_dws.dws\_user\_visit\_month1 | qfbap\_dm.dm\_user\_visit | 经过计算,详见下方脚本:计算用户行为 |

# 环境搭建

开发环境

版本控制

流程和任务及bug追踪

业务数据的初始化

# SQL思路分析

给大家介绍两种常见的SQL思路,最常用的max,min这种很简单就不说了,说一下常见的分析思路.

1. 当看到有求第一次最后一次,最大最小或者topn的详细数据,增加排名序号

例如我们需求中的用户行为求最近一次访问时间,最近一次访问地址,第一次访问时间,第一次访问地址,我们可以对数据进行两次排名

1. 根据用户分组按照时间进行升序和降序排名

select

row\_number() over(distribute by user\_id sort by visit\_time asc) as rn\_asc,

row\_number() over(distribute by user\_id sort by visit\_time desc) as rn\_desc,

visit\_time,

visit\_ip

from user\_app\_click\_log

1. 从上面的结果中取rn\_asc=1的数据就是第一次的访问记录行
2. 从上面的结果中取rn\_desc=1的数据就是最后一次的访问记录行
3. 使用case语句进行判断,进行行转列

case when rn\_asc=1 then visit\_time end as first\_time

case when rn\_desc=1 then visit\_time end as latest\_time

1. 取数据

select

(case when rn\_asc=1 then visit\_time end) as first\_time,

(case when rn\_desc=1 then visit\_time end) as latest\_time,

(case when rn\_asc=1 then visit\_ip end) as first\_ip,

(case when rn\_desc=1 then visit\_ip end) as latest\_ip

from (

select

row\_number() over(distribute by user\_id sort by visit\_time) as rn\_asc,

row\_number() over(distribute by user\_id sort by visit\_time) as rn\_desc,

visit\_time,

visit\_ip

from user\_app\_click\_log ) t

where rn\_asc=1 and rn\_desc=1

f) topn,有了排名topn就很容易了.

1. 计算N天的访问次数

给每条记录打上一个标签,这也是数仓中常用的存储设计方式,例如有销售记录,按照区域进行统计,数仓中会转换为一张宽表

源数据

2018-12-29 1000000 北京

2018-12-28 2000000 上海

数仓中的存储格式很可能是这样的

日期 北京 上海 广州

2018-12-29 1000000 0 0

2018-12-28 0 2000000 0

那么我们在统计数据的时候,北京的金额就sum(北京),上海的就sum(上海),广州的就sum(广州)即可

计算N天的访问次数也可以采用类似的思想,如果访问日期在这个范围内,则给一个标识值,比如当前日期为2018-12-29

select

case when visit\_time>=date\_sub(‘2018-12-29’,7) then 1 end as dt\_7,

case when visit\_time>=date\_sub(‘2018-12-29’,14) then 1 end as dt\_15,

case when visit\_time>=date\_sub(‘2018-12-29’,29) then 1 end as dt\_30

from user\_app\_click\_log

有了flag值以后,我们就可以进行汇总了

select

sum(dt\_7) day7,

sum(dt\_15) day15,

sum(dt\_30) day30

from (

select

case when visit\_time>=date\_sub(‘2018-12-29’,7) then 1 end as dt\_7,

case when visit\_time>=date\_sub(‘2018-12-29’,14) then 1 end as dt\_15,

case when visit\_time>=date\_sub(‘2018-12-29’,29) then 1 end as dt\_30

from user\_app\_click\_log) t

group by user\_id

# 任务调度

调度的任务主要是抽取数据和进行指标计算,调度工具使用azkaban

## 数据初始化

数据初始化分为两部分mysql部分和hive部分

### mysql

1. 执行sql脚本,创建数据库和表,脚本为\qfbap\数据初始化\qfbap\_data\_init.sql
2. 执行脚本后,会自动开始生成数据
3. 数据生成一定量后,可以将生成数据的任务停止

使用sqlyog等数据库客户端软件,新建一个查询窗口,执行以下语句,修改事件

源事件为:

|  |
| --- |
| DELIMITER $$  ALTER EVENT `even\_init` ON SCHEDULE EVERY 1 SECOND STARTS '2018-12-17 00:22:00' ON COMPLETION PRESERVE ENABLE DO BEGIN  CALL usp\_create\_user\_withorder(10);  END$$ |

**修改为:**

|  |
| --- |
| DELIMITER $$  ALTER EVENT `even\_init` ON SCHEDULE EVERY 1 SECOND STARTS '2018-12-17 00:22:00' ON COMPLETION PRESERVE ENABLE DO BEGIN  -- CALL usp\_create\_user\_withorder(10);  END$$ |

**或者修改为:**

|  |
| --- |
| DELIMITER $$  ALTER EVENT `even\_init` ON SCHEDULE EVERY 1 SECOND STARTS '2018-12-17 00:22:00' ON COMPLETION PRESERVE disable DO BEGIN  CALL usp\_create\_user\_withorder(10);  END$$  DELIMITER ; |

### hive

1. 执行三个sql文件,来创建库和表:

hive\_ods\_create\_table.sql,

hive\_dwd\_create\_table.sql,

hive\_dws\_create\_table.sql

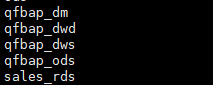
使用hive -f 来执行上面的三个文件即可.

hive -f hive\_ods\_create\_table.sql

hive -f hive\_dwd\_create\_table.sql

hive -f hive\_dws\_create\_table.sql

执行完后,show databases查看数据库是否创建成功



1. 执行两个shell文件,来创建sqoop任务

注意修改文件,更改成自己的密码

load\_data\_full\_sqoop.sh

load\_data\_incr\_sqoop.sh

这两个文件中的所有的用户名和密码都更改为自己的

并且更改sqoop的配置sqoop-site.xml文件

将这段配置去掉注释,执行一次sqoop的任务,记录客户端的密码

|  |
| --- |
| <property>      <name>sqoop.metastore.client.record.password</name>      <value>true</value>      <description>If true, allow saved passwords in the metastore.      </description>  </property> |

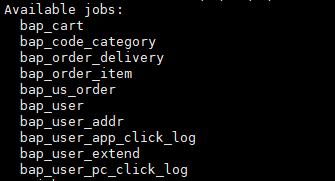
chmod +x \*.sh

./load\_data\_full\_sqoop.sh

./load\_data\_incr\_sqoop.sh

执行完后,查看命令

sqoop job –list 看一下sqoop任务是否创建成功了



注:说明\*\*\*\_full\_\*\*\*是全量,\*\*\*\_incr\_\*\*\*是增量

## ods层

从mysql数据库使用sqoop将数据抽取到hive的ods层

有两个脚本:

load\_data\_full.sh

load\_data\_incr.sh

## dwd层

从hive的ods层将数据抽取到hive的dwd层

有两个脚本

load\_data\_dwd.sh

load\_data\_dwd.sql

## dws层

从hive的dwd层抽取数据到hive的dwd层

有两个脚本

load\_data\_dws.sh

load\_data\_dws.sql

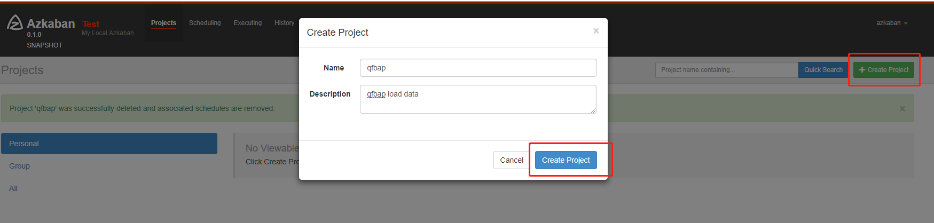
## dm层

从hive的dws层计算数据到hive的dm层和mysql的dm层

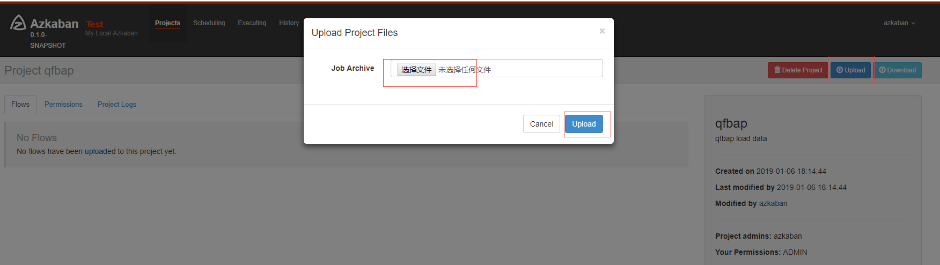
spark任务

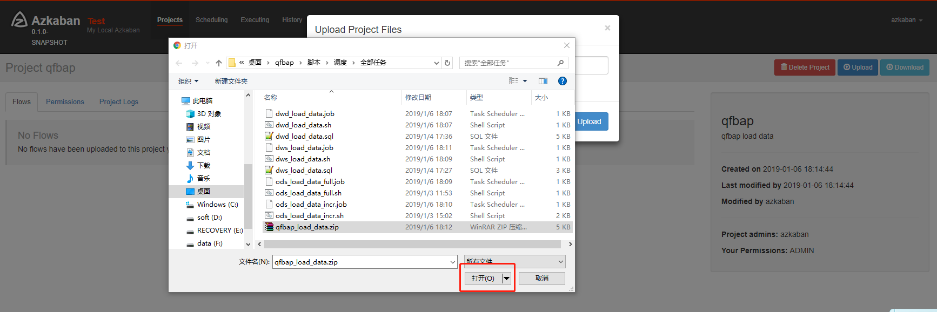
## 调度任务

1. 创建一个任务

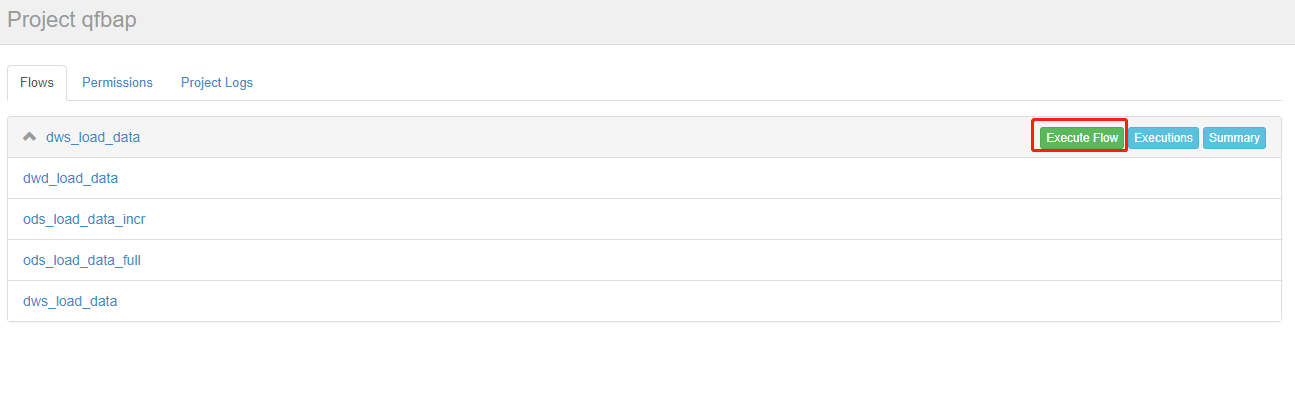


1. 上传任务





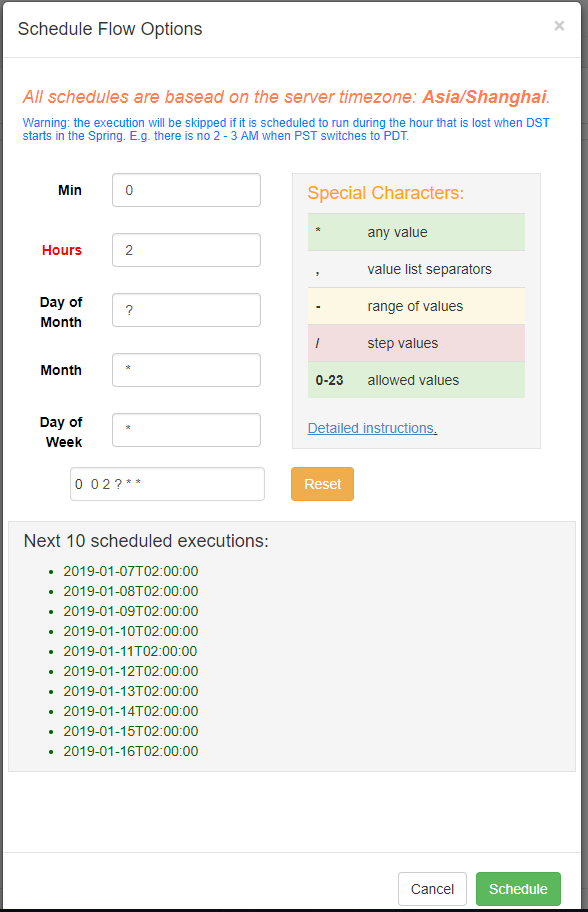
1. 上传后可以看到有ods层的,dwd层的和dws层的



1. 点上面红框的按钮,执行,可以看到任务的流程



1. 点schedule按钮,设置调度计划,设置每天凌晨2点执行一次



调度任务就执行完成了

# 数据展示