数据质量管理

# 第1章 数据质量管理概述

## 1.1 数据质量管理定义

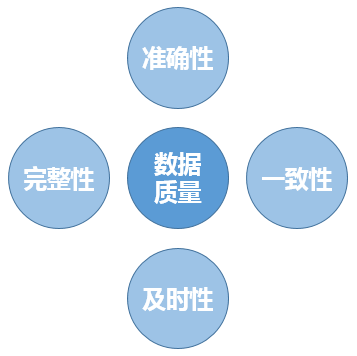
数据质量管理**DQM**（Data Quality Management），是指对[数据](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE/5947370" \t "_blank)从计划、获取、存储、共享、维护、应用、消亡生命周期的每个阶段里可能引发的各类数据质量问题，进行**识别、度量、监控、预警**等一系列管理活动，并通过改善和提高组织的管理水平使得[数据质量](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E8%B4%A8%E9%87%8F/5134536" \t "_blank)获得进一步提高。

数据质量管理是循环管理过程，其终极目标是**通过可靠的数据提升数据在使用中的价值，并最终为企业赢得经济效益**。

## 1.2 数据质量评价指标

数据质量管理的最终目标是改善，任何改善都是建立在评价的基础上。通常[数据质量](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E8%B4%A8%E9%87%8F/5134536" \t "_blank)的评价标准包括以下内容。

|  |  |  |
| --- | --- | --- |
| **评价标准** | **描述** | **监控项** |
| **唯一性** | 指**主键**保持唯一 | 字段**唯一性**检查 |
| **完整性** | 主要包括**记录缺失**和**字段值缺失**等方面 | 字段**枚举值**检查 |
| 字段**记录数**检查 |
| 字段**空值**检查 |
| **准确性** | 数据生成的正确性，数据在整个链路流转的正确性 | 波动阈值检查 |
| **合法性** | 主要包括**格式、类型、域值**的合法性 | 字段日期格式检查 |
| 字段长度检查 |
| 字段值域检查 |
| **时效性** | 主要包括数据处理的时效性 | 批处理是否按时完成 |





# 第2章 数据质量管理实操

## 2.1 需求分析

我们的数仓项目主要监控以下数据的指标：

* **ODS**层：**数据量**，**每日环比**和**每周同比**变化不能超过一定范围
* **DIM**层：**不能出现id空值**，**重复值**；
* **DWD**层：**不能出现id空值，重复值；**

在每层中任意挑选一张表作为示例。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **表** | **检查项目** | **依据** | **异常值下限** | | **异常值上限** |
| **ods\_order\_info\_inc** | 同比增长 | 数据总量 | -10% | 10% | |
| 环比增长 | 数据总量 | -10% | 50% | |
| **值域检查** | final\_amount | 0 | 100 | |
| **dwd\_trade\_order\_detail\_inc** | 空值检查 | id | 0 | 10 | |
| 重复值检查 | id | 0 | 5 | |
| **dim\_sku\_full** | 空值检查 | id | 0 | 10 | |
| 重复值检查 | id | 0 | 5 | |

## 2.2 功能模块



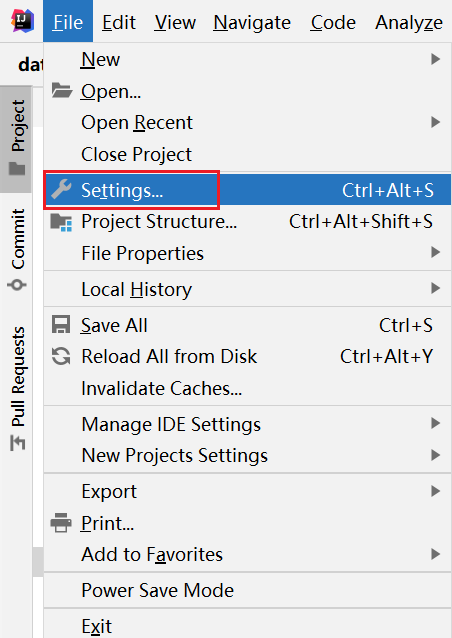
## 2.3 开发环境准备

### 2.3.1 Python开发环境准备

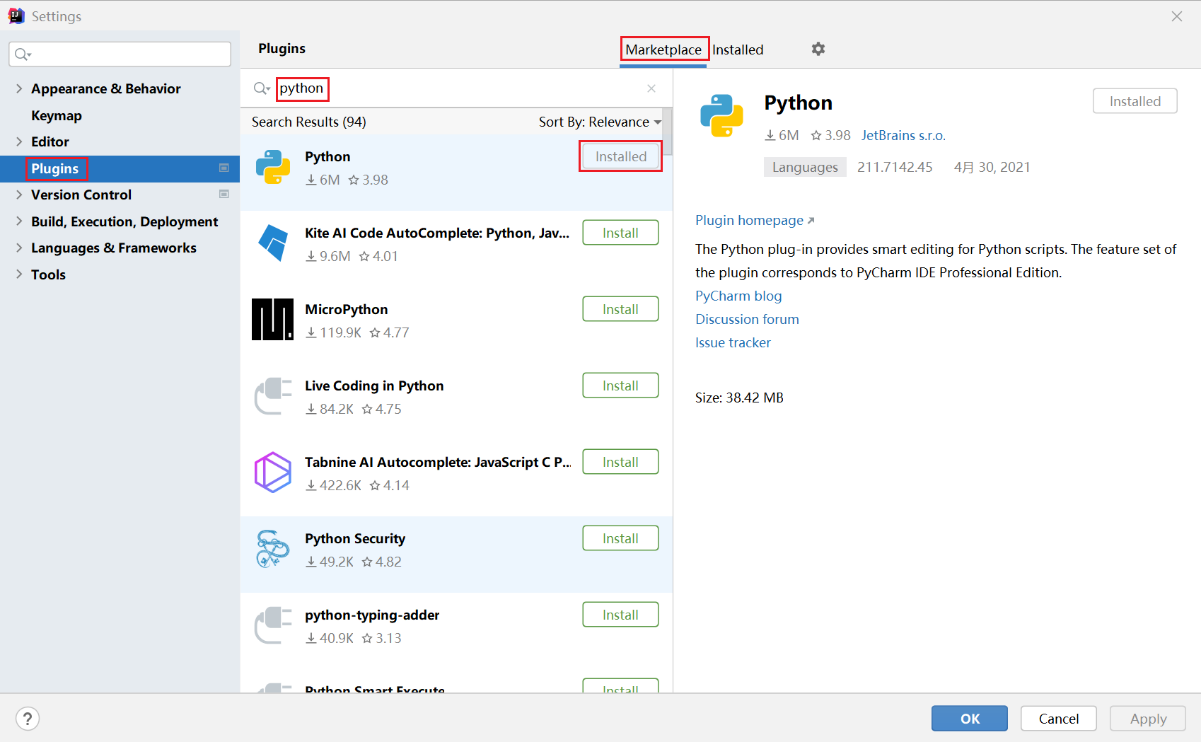
本文使**用Python和Shell脚本**实现数据质量监控的各项功能，故需先搭建相应的开发环境，Python开发可选择IDEA（需安装Python插件），或PyCharm等工具，本文使用IDEA作为开发工具。

**1. 安装Python插件**

（1）在IDEA中点击“File”，在下拉选择中点击“Settings…”

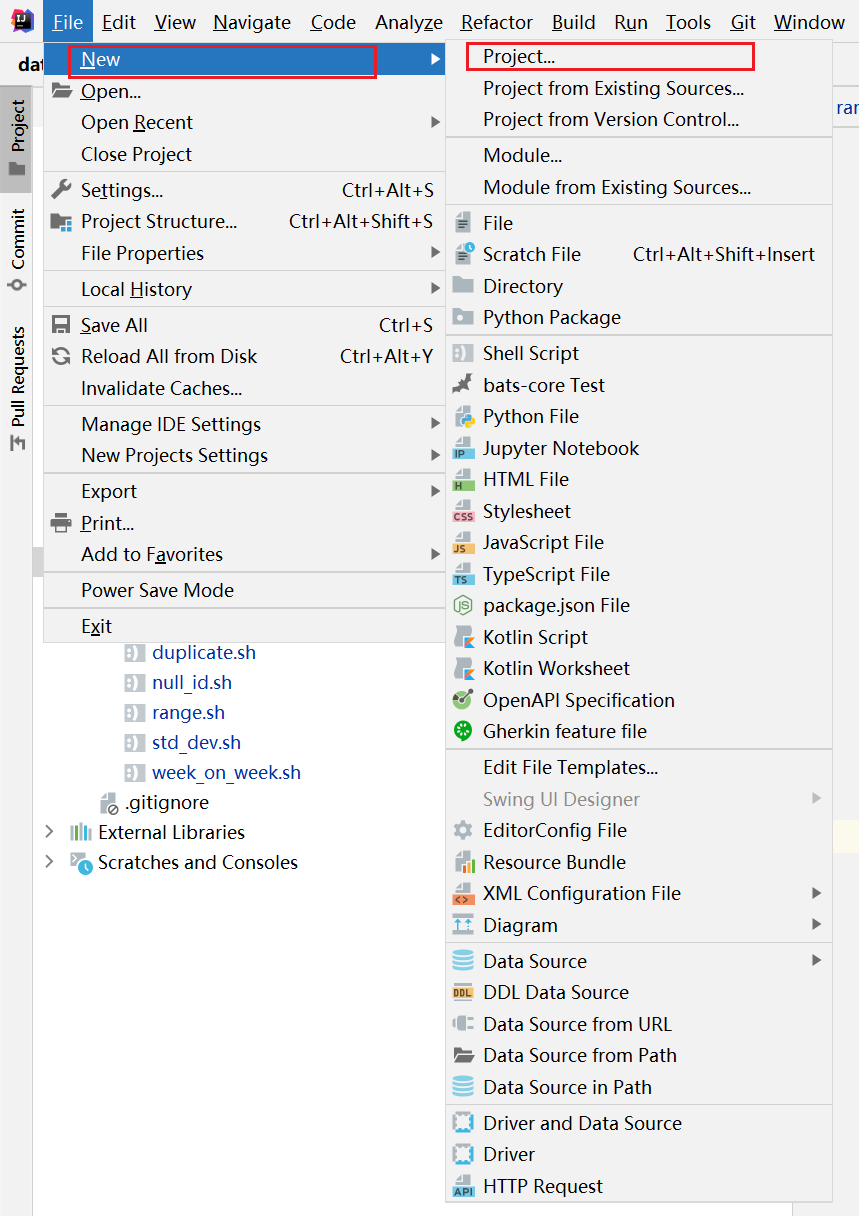


（2）点击“Plugins”，点击右上角的“Marketplace”，然后在搜索框中输入“**python**”，在搜索结果列表中找到Python插件，点击“Install”，安装插件。

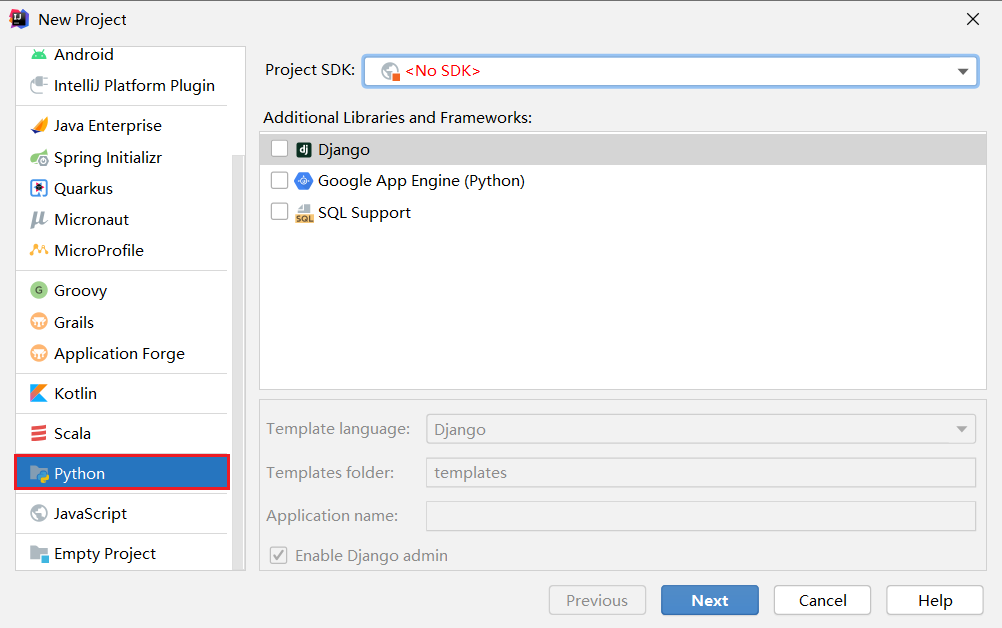


**2. 新建一个Python项目**

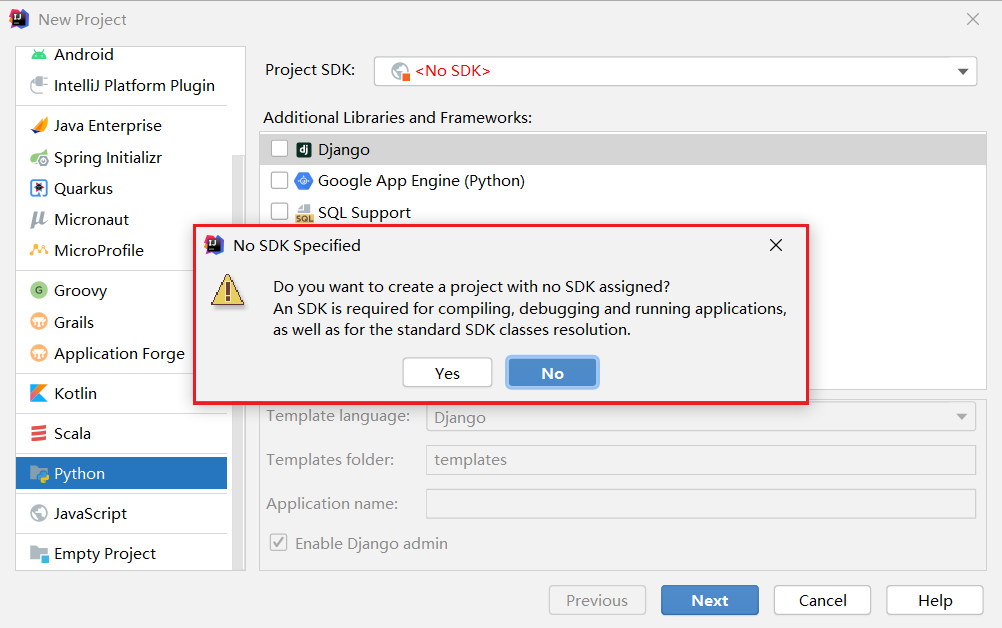
（1）点击Idea中的“File”，在下列列表中点击“New”，在右侧弹出的列表中点击“Project…”



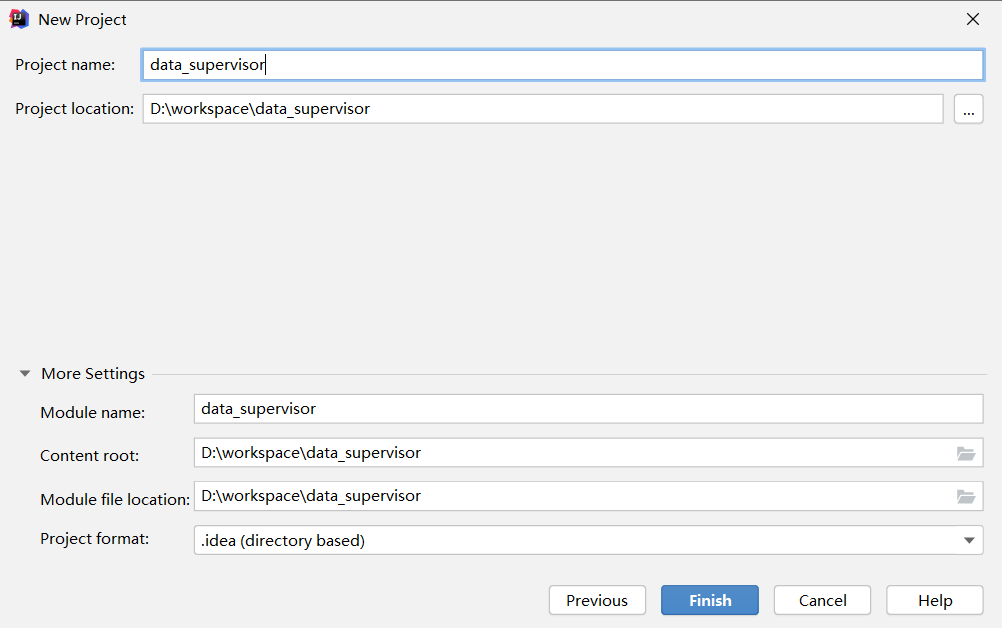
（2）在新建的工程中，点击“Python”，然后点击Next



（3）首次创建Python项目，会提示无Python SDK，此处选择Yes，后续再添加SDK。



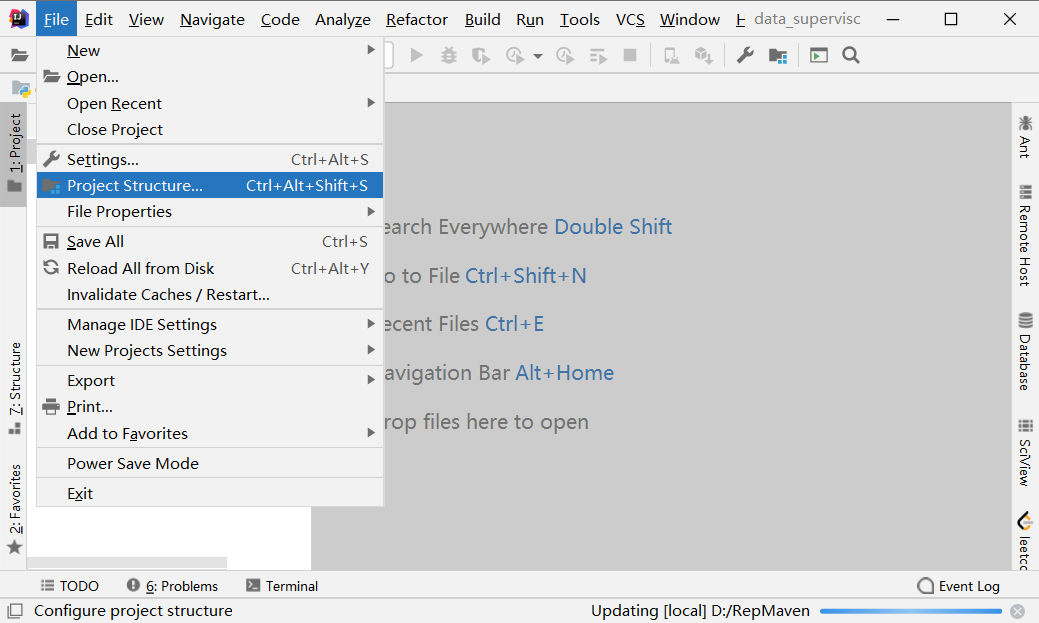
（4）填写项目名称和项目路径等基本信息，点击Finish



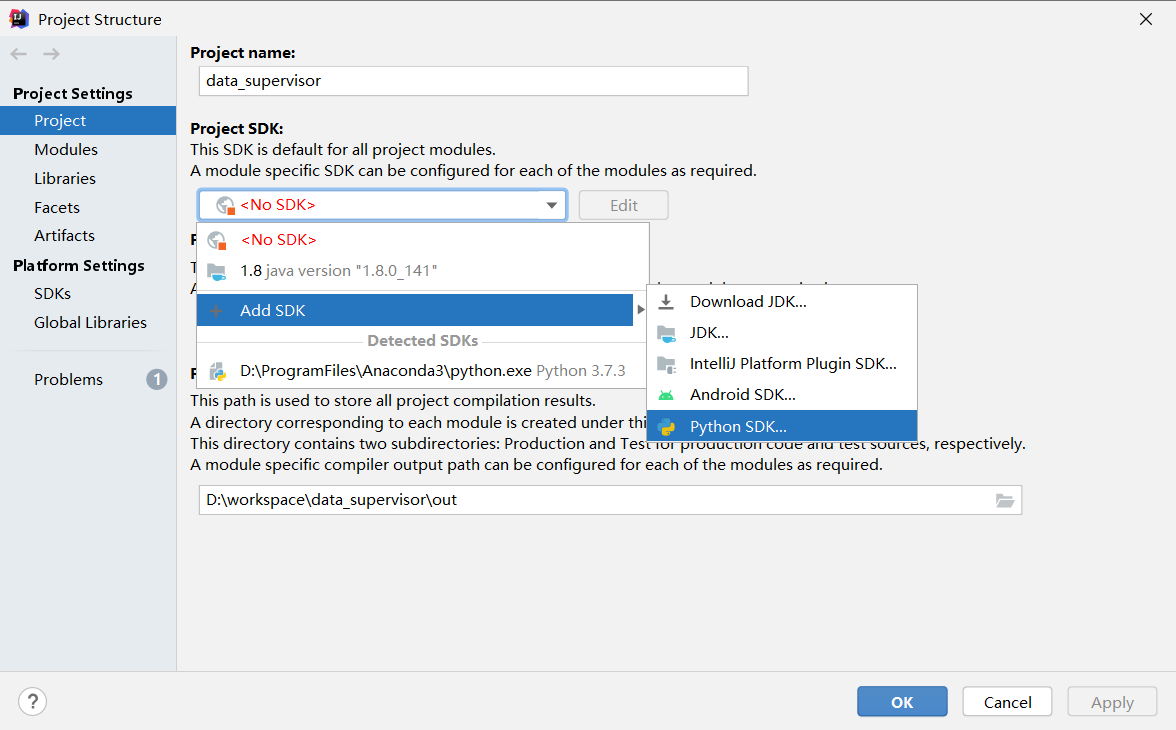
（5）添加Python SDK

为了保证测试和运行的Python环境一致，我们配置项目采用**远程集群的Python环境执行本地代码**，以下为具体配置步骤。

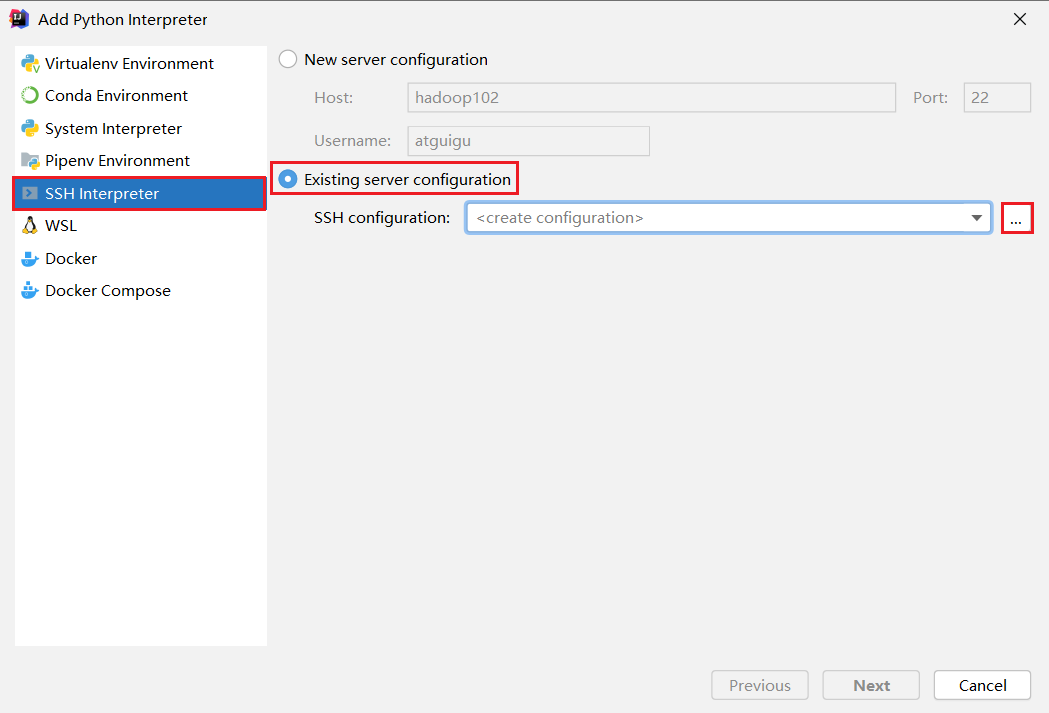
第一步：点击“File”→“Project Structure”



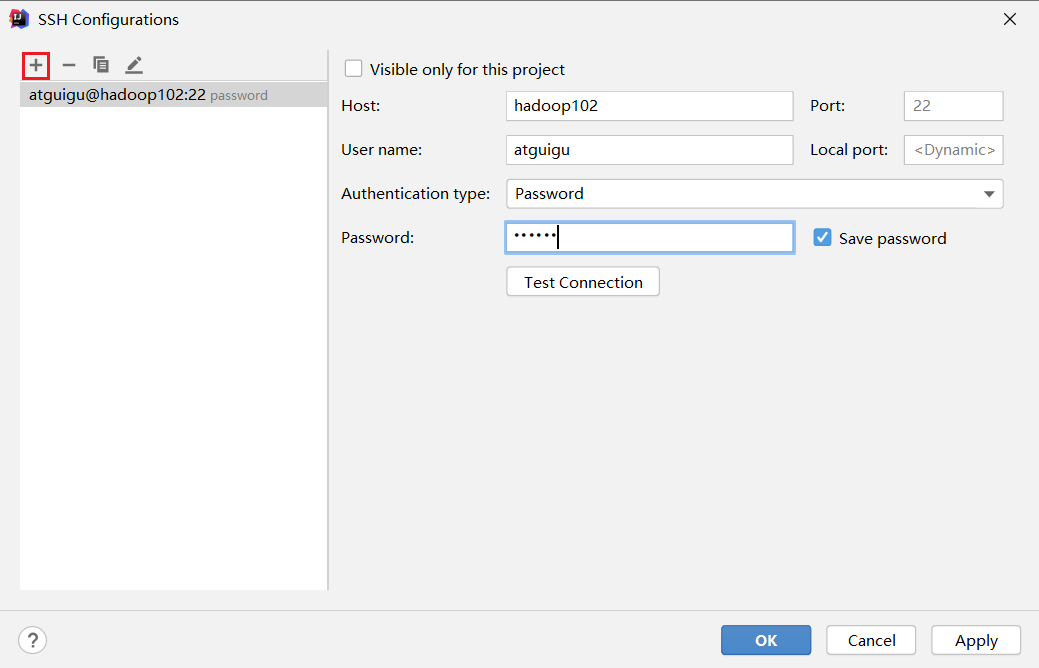
第二步：按照下图操作，增加Python SDK。



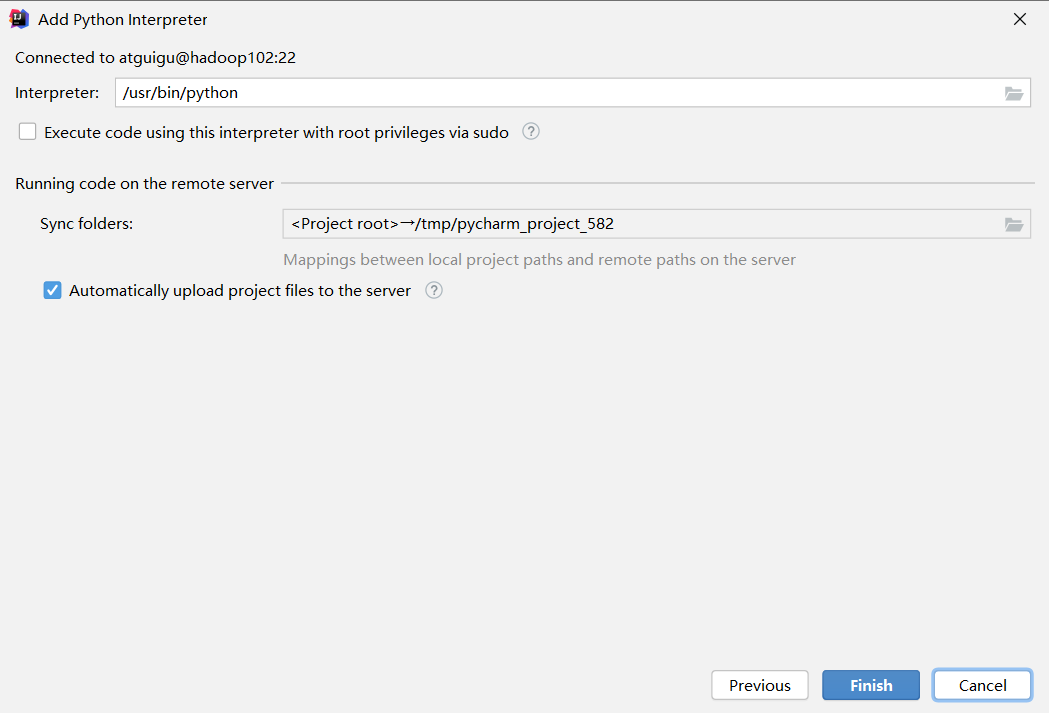
第三步：点击“SSH Interpreter”，选择“Existing server configuration”，点击“…”

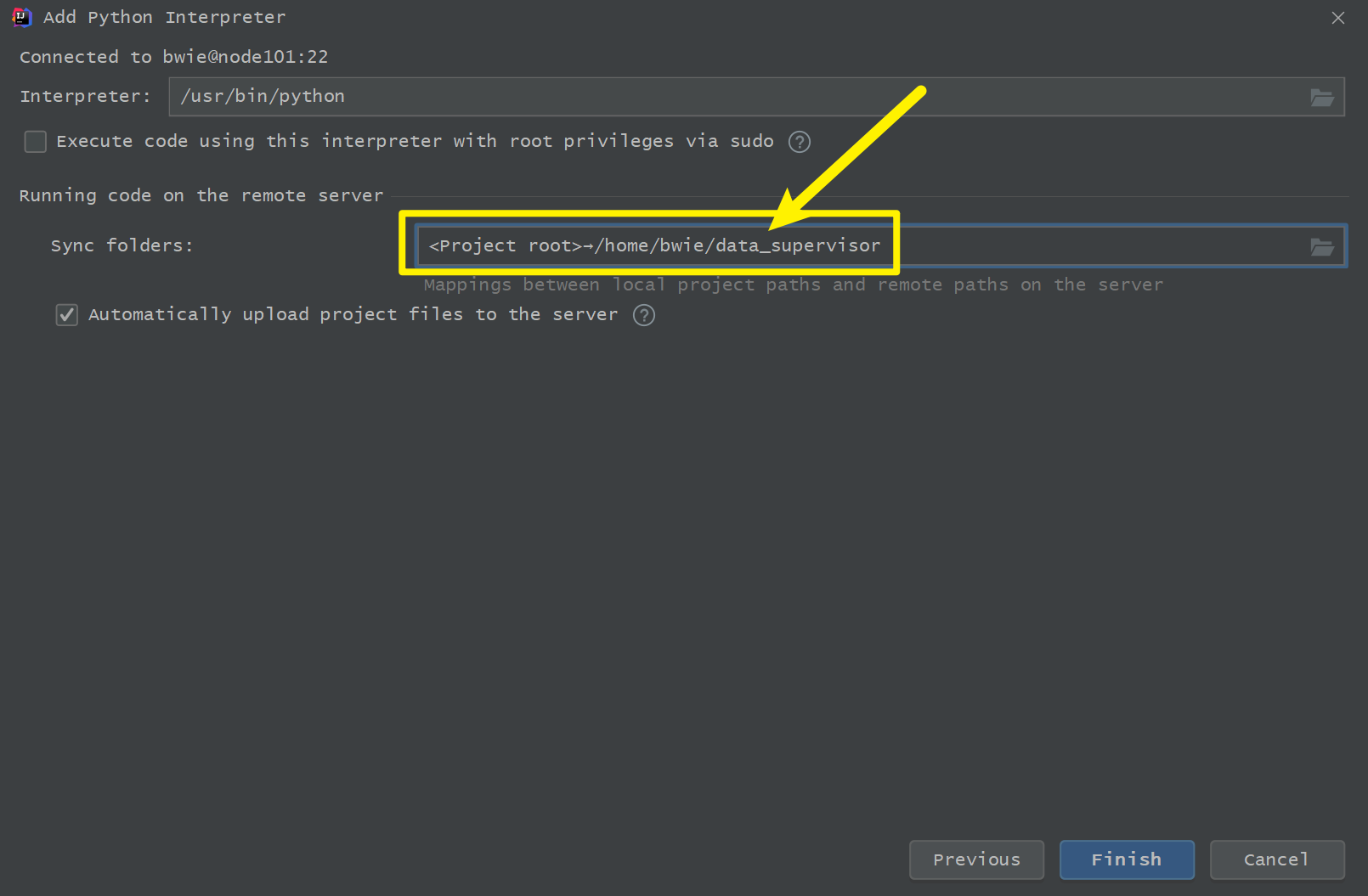


第四步：点击“+”，填入ssh连接信息，点击Next

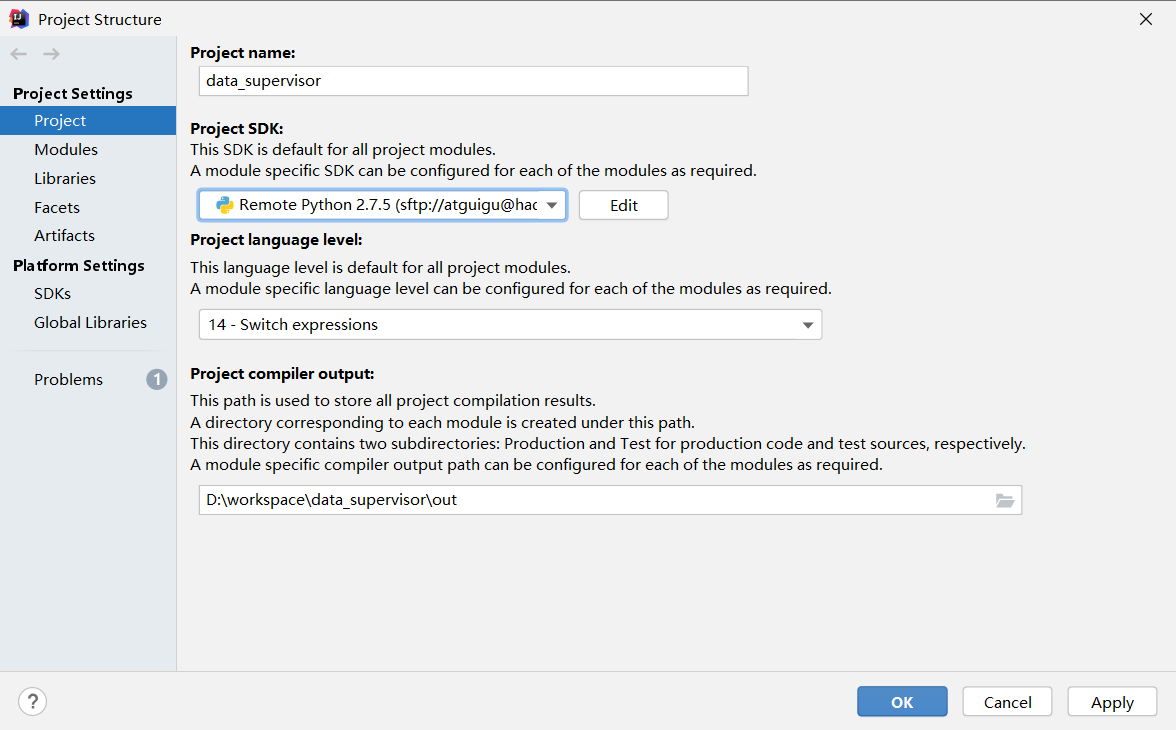


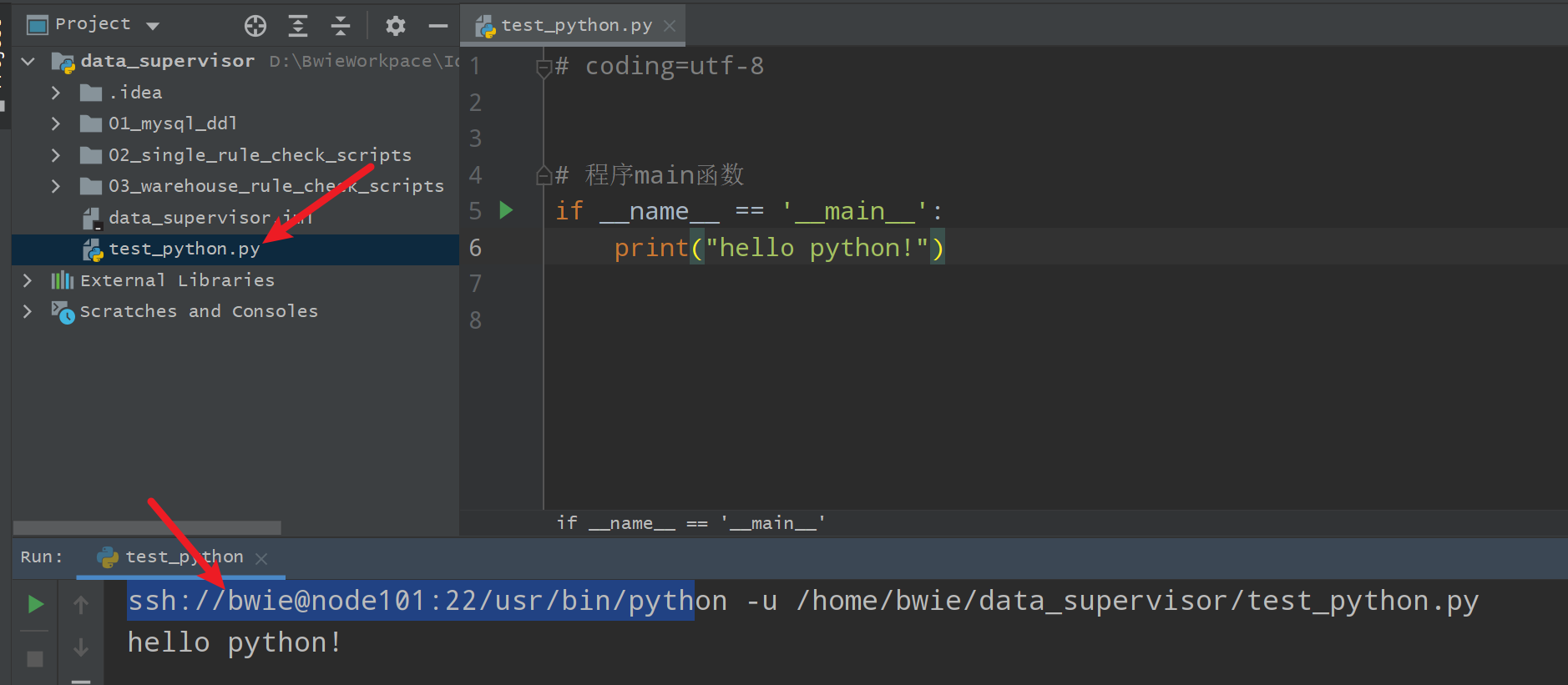
第五步：点击Finish





第六步：点击OK





### 2.3.2 初始化MySQL环境

MySQL主要用于**存储数据质量监控的结果值**，这里需要**提前建库建表**。详细建表语句如下：

（1）创建data\_supervisor库

drop database if exists data\_supervisor;

create database data\_supervisor;

（2）创建空值指标表，null\_id

CREATE TABLE data\_supervisor.`null\_id`

(

`dt` date NOT NULL COMMENT '日期',

`tbl` varchar(50) NOT NULL COMMENT '表名',

`col` varchar(50) NOT NULL COMMENT '列名',

`value` int DEFAULT NULL COMMENT '空ID个数',

`value\_min` int DEFAULT NULL COMMENT '下限',

`value\_max` int DEFAULT NULL COMMENT '上限',

`notification\_level` int DEFAULT NULL COMMENT '警告级别',

PRIMARY KEY (`dt`, `tbl`, `col`)

) ENGINE = InnoDB

DEFAULT CHARSET = utf8

comment '空值指标表';

（3）创建重复值指标表，duplicate

CREATE TABLE data\_supervisor.`duplicate`

(

`dt` date NOT NULL COMMENT '日期',

`tbl` varchar(50) NOT NULL COMMENT '表名',

`col` varchar(50) NOT NULL COMMENT '列名',

`value` int DEFAULT NULL COMMENT '重复值个数',

`value\_min` int DEFAULT NULL COMMENT '下限',

`value\_max` int DEFAULT NULL COMMENT '上限',

`notification\_level` int DEFAULT NULL COMMENT '警告级别',

PRIMARY KEY (`dt`, `tbl`, `col`)

) ENGINE = InnoDB

DEFAULT CHARSET = utf8

comment '重复值指标表';

（4）创建值域指标表，rng

CREATE TABLE data\_supervisor.`rng`

(

`dt` date NOT NULL COMMENT '日期',

`tbl` varchar(50) NOT NULL COMMENT '表名',

`col` varchar(50) NOT NULL COMMENT '列名',

`value` int DEFAULT NULL COMMENT '超出预定值域个数',

`range\_min` int DEFAULT NULL COMMENT '值域下限',

`range\_max` int DEFAULT NULL COMMENT '值域上限',

`value\_min` int DEFAULT NULL COMMENT '下限',

`value\_max` int DEFAULT NULL COMMENT '上限',

`notification\_level` int DEFAULT NULL COMMENT '警告级别',

PRIMARY KEY (`dt`, `tbl`, `col`)

) ENGINE = InnoDB

DEFAULT CHARSET = utf8

comment '值域指标表';

（5）创建环比增长指标表，day\_on\_day

CREATE TABLE data\_supervisor.`day\_on\_day`

(

`dt` date NOT NULL COMMENT '日期',

`tbl` varchar(50) NOT NULL COMMENT '表名',

`value` double DEFAULT NULL COMMENT '环比增长百分比',

`value\_min` double DEFAULT NULL COMMENT '增长上限',

`value\_max` double DEFAULT NULL COMMENT '增长上限',

`notification\_level` int DEFAULT NULL COMMENT '警告级别',

PRIMARY KEY (`dt`, `tbl`)

) ENGINE = InnoDB

DEFAULT CHARSET = utf8

comment '环比增长指标表';

（6）创建同比增长指标表，week\_on\_week

CREATE TABLE data\_supervisor.`week\_on\_week`

(

`dt` date NOT NULL COMMENT '日期',

`tbl` varchar(50) NOT NULL COMMENT '表名',

`value` double DEFAULT NULL COMMENT '同比增长百分比',

`value\_min` double DEFAULT NULL COMMENT '增长上限',

`value\_max` double DEFAULT NULL COMMENT '增长上限',

`notification\_level` int DEFAULT NULL COMMENT '警告级别',

PRIMARY KEY (`dt`, `tbl`)

) ENGINE = InnoDB

DEFAULT CHARSET = utf8

comment '同比增长指标表';

## 2.4 规则检测模块

### 2.4.1 单一规则检测脚本编写

检测规则脚本分为五类：分别是**空id（主键空值）检查脚本、重复id（主键重复）检查脚本、值域（字段值范围）检查脚本、数据量环比（天与天环比）检查脚本和数据量同比（周与周同比）检查脚本**。

下面分别给大家介绍一下五类检测脚本的具体编写。

#### 空id检查脚本

在Idea中创建一个文件null\_id.sh，在文件中编写如下内容：

实现的主要功能是：计算空值个数，并将结果和自己定义的阈值上下限，插入到MySQL表中。

#!/usr/bin/env bash

# -\*- coding: utf-8 -\*-

# 检查id空值

# 解析参数

while getopts "t:d:c:s:x:l:" arg; do

case $arg in

# 要处理的表名

t)

TABLE=$OPTARG

;;

# 日期

d)

DT=$OPTARG

;;

# 要计算空值的列名

c)

COL=$OPTARG

;;

# 空值指标下限

s)

MIN=$OPTARG

;;

# 空值指标上限

x)

MAX=$OPTARG

;;

# 告警级别

l)

LEVEL=$OPTARG

;;

?)

echo "unkonw argument"

exit 1

;;

esac

done

#如果dt和level没有设置，那么默认值dt是昨天 告警级别是0

[ "$DT" ] || DT=$(date -d '-1 day' +%F)

[ "$LEVEL" ] || LEVEL=0

# 数仓DB名称

HIVE\_DB=gmall

# 查询引擎

HIVE\_ENGINE=hive

# MySQL相关配置

mysql\_user="root"

mysql\_passwd="000000"

mysql\_host="hadoop102"

mysql\_DB="data\_supervisor"

mysql\_tbl="null\_id"

# 认证为hive用户，如在非安全(Hadoop未启用Kerberos认证)环境中，则无需认证

kinit -kt /etc/security/keytab/hive.keytab hive

# 空值个数

RESULT=$($HIVE\_ENGINE -e "set hive.cli.print.header=false;select count(1) from $HIVE\_DB.$TABLE where dt='$DT' and $COL is null;")

#结果插入MySQL

mysql -h"$mysql\_host" -u"$mysql\_user" -p"$mysql\_passwd" \

-e"INSERT INTO $mysql\_DB.$mysql\_tbl VALUES('$DT', '$TABLE', '$COL', $RESULT, $MIN, $MAX, $LEVEL)

ON DUPLICATE KEY UPDATE \`value\`=$RESULT, value\_min=$MIN, value\_max=$MAX, notification\_level=$LEVEL;"

#### 重复id检查脚本

在Idea中创建一个文件duplicate.sh，在文件中编写如下内容：

实现的主要功能是：计算重复值个数，并将结果和自己定义的阈值上下限，插入到MySQL表中。

#!/usr/bin/env bash

# -\*- coding: utf-8 -\*-

# 监控某张表一列的重复值

# 参数解析

while getopts "t:d:c:s:x:l:" arg; do

case $arg in

# 要处理的表名

t)

TABLE=$OPTARG

;;

# 日期

d)

DT=$OPTARG

;;

# 要计算重复值的列名

c)

COL=$OPTARG

;;

# 重复值指标下限

s)

MIN=$OPTARG

;;

# 重复值指标上限

x)

MAX=$OPTARG

;;

# 告警级别

l)

LEVEL=$OPTARG

;;

?)

echo "unkonw argument"

exit 1

;;

esac

done

#如果dt和level没有设置，那么默认值dt是昨天 告警级别是0

[ "$DT" ] || DT=$(date -d '-1 day' +%F)

[ "$LEVEL" ] || LEVEL=0

# 数仓DB名称

HIVE\_DB=gmall

# 查询引擎

HIVE\_ENGINE=hive

# MySQL相关配置

mysql\_user="root"

mysql\_passwd="000000"

mysql\_host="hadoop102"

mysql\_DB="data\_supervisor"

mysql\_tbl="duplicate"

# 认证为hive用户，如在非安全(Hadoop未启用Kerberos认证)环境中，则无需认证

kinit -kt /etc/security/keytab/hive.keytab hive

# 重复值个数

RESULT=$($HIVE\_ENGINE -e "set hive.cli.print.header=false;select count(1) from (select $COL from $HIVE\_DB.$TABLE where dt='$DT' group by $COL having count($COL)>1) t1;")

# 将结果插入MySQL

mysql -h"$mysql\_host" -u"$mysql\_user" -p"$mysql\_passwd" \

-e"INSERT INTO $mysql\_DB.$mysql\_tbl VALUES('$DT', '$TABLE', '$COL', $RESULT, $MIN, $MAX, $LEVEL)

ON DUPLICATE KEY UPDATE \`value\`=$RESULT, value\_min=$MIN, value\_max=$MAX, notification\_level=$LEVEL;"

#### 值域检查脚本

在Idea中创建一个文件range.sh，在文件中编写如下内容：

实现的主要功能是：计算超出规定值域的值的个数，并将结果和自己定义的阈值上下限，插入到MySQL表中。

#!/usr/bin/env bash

# -\*- coding: utf-8 -\*-

# 计算某一列异常值个数

while getopts "t:d:l:c:s:x:a:b:" arg; do

case $arg in

# 要处理的表名

t)

TABLE=$OPTARG

;;

# 日替

d)

DT=$OPTARG

;;

# 要处理的列

c)

COL=$OPTARG

;;

# 不在规定值域的值的个数下限

s)

MIN=$OPTARG

;;

# 不在规定值域的值的个数上限

x)

MAX=$OPTARG

;;

# 告警级别

l)

LEVEL=$OPTARG

;;

# 规定值域为a-b

a)

RANGE\_MIN=$OPTARG

;;

b)

RANGE\_MAX=$OPTARG

;;

?)

echo "unkonw argument"

exit 1

;;

esac

done

#如果dt和level没有设置，那么默认值dt是昨天 告警级别是0

[ "$DT" ] || DT=$(date -d '-1 day' +%F)

[ "$LEVEL" ] || LEVEL=0

# 数仓DB名称

HIVE\_DB=gmall

# 查询引擎

HIVE\_ENGINE=hive

# MySQL相关配置

mysql\_user="root"

mysql\_passwd="000000"

mysql\_host="hadoop102"

mysql\_DB="data\_supervisor"

mysql\_tbl="rng"

# 认证为hive用户，如在非安全(Hadoop未启用Kerberos认证)环境中，则无需认证

kinit -kt /etc/security/keytab/hive.keytab hive

# 查询不在规定值域的值的个数

RESULT=$($HIVE\_ENGINE -e "set hive.cli.print.header=false;select count(1) from $HIVE\_DB.$TABLE where dt='$DT' and $COL not between $RANGE\_MIN and $RANGE\_MAX;")

# 将结果写入MySQL

mysql -h"$mysql\_host" -u"$mysql\_user" -p"$mysql\_passwd" \

-e"INSERT INTO $mysql\_DB.$mysql\_tbl VALUES('$DT', '$TABLE', '$COL', $RESULT, $RANGE\_MIN, $RANGE\_MAX, $MIN, $MAX, $LEVEL)

ON DUPLICATE KEY UPDATE \`value\`=$RESULT, range\_min=$RANGE\_MIN, range\_max=$RANGE\_MAX, value\_min=$MIN, value\_max=$MAX, notification\_level=$LEVEL;"

#### 4. 数据量环比检查脚本

在Idea中创建一个文件day\_on\_day.sh，在文件中编写如下内容：

实现的主要功能是：计算数据量环比增长值，并将结果和自己定义的阈值上下限，插入到MySQL表中。

#!/usr/bin/env bash

# -\*- coding: utf-8 -\*-

# 计算一张表单日数据量环比增长值

# 参数解析

while getopts "t:d:s:x:l:" arg; do

case $arg in

# 要处理的表名

t)

TABLE=$OPTARG

;;

# 日期

d)

DT=$OPTARG

;;

# 环比增长指标下限

s)

MIN=$OPTARG

;;

# 环比增长指标上限

x)

MAX=$OPTARG

;;

# 告警级别

l)

LEVEL=$OPTARG

;;

?)

echo "unkonw argument"

exit 1

;;

esac

done

#如果dt和level没有设置，那么默认值dt是昨天 告警级别是0

[ "$DT" ] || DT=$(date -d '-1 day' +%F)

[ "$LEVEL" ] || LEVEL=0

# 数仓DB名称

HIVE\_DB=gmall

# 查询引擎

HIVE\_ENGINE=hive

# MySQL相关配置

mysql\_user="root"

mysql\_passwd="000000"

mysql\_host="hadoop102"

mysql\_DB="data\_supervisor"

mysql\_tbl="day\_on\_day"

# 认证为hive用户，如在非安全(Hadoop未启用Kerberos认证)环境中，则无需认证

kinit -kt /etc/security/keytab/hive.keytab hive

# 昨日数据量

YESTERDAY=$($HIVE\_ENGINE -e "set hive.cli.print.header=false; select count(1) from $HIVE\_DB.$TABLE where dt=date\_add('$DT',-1);")

# 今日数据量

TODAY=$($HIVE\_ENGINE -e "set hive.cli.print.header=false;select count(1) from $HIVE\_DB.$TABLE where dt='$DT';")

# 计算环比增长值

if [ "$YESTERDAY" -ne 0 ]; then

RESULT=$(awk "BEGIN{print ($TODAY-$YESTERDAY)/$YESTERDAY\*100}")

else

RESULT=10000

fi

# 将结果写入MySQL表格

mysql -h"$mysql\_host" -u"$mysql\_user" -p"$mysql\_passwd" \

-e"INSERT INTO $mysql\_DB.$mysql\_tbl VALUES('$DT', '$TABLE', $RESULT, $MIN, $MAX, $LEVEL)

ON DUPLICATE KEY UPDATE \`value\`=$RESULT, value\_min=$MIN, value\_max=$MAX, notification\_level=$LEVEL;"

#### 5. 数据量同比检查脚本

在Idea中创建一个文件week\_on\_week.sh，在文件中编写如下内容：

实现的主要功能是：计算数据量同比增长值，并将结果和自己定义的阈值上下限，插入到MySQL表中。

#!/usr/bin/env bash

# -\*- coding: utf-8 -\*-

# 计算一张表一周数据量同比增长值

# 参数解析

while getopts "t:d:s:x:l:" arg; do

case $arg in

# 要处理的表名

t)

TABLE=$OPTARG

;;

# 日期

d)

DT=$OPTARG

;;

# 同比增长指标下限

s)

MIN=$OPTARG

;;

# 同比增长指标上限

x)

MAX=$OPTARG

;;

# 告警级别

l)

LEVEL=$OPTARG

;;

?)

echo "unkonw argument"

exit 1

;;

esac

done

#如果dt和level没有设置，那么默认值dt是昨天 告警级别是0

[ "$DT" ] || DT=$(date -d '-1 day' +%F)

[ "$LEVEL" ] || LEVEL=0

# 数仓DB名称

HIVE\_DB=gmall

# 查询引擎

HIVE\_ENGINE=hive

# MySQL相关配置

mysql\_user="root"

mysql\_passwd="000000"

mysql\_host="hadoop102"

mysql\_DB="data\_supervisor"

mysql\_tbl="week\_on\_week"

# 认证为hive用户，如在非安全(Hadoop未启用Kerberos认证)环境中，则无需认证

kinit -kt /etc/security/keytab/hive.keytab hive

# 上周数据量

LASTWEEK=$($HIVE\_ENGINE -e "set hive.cli.print.header=false;select count(1) from $HIVE\_DB.$TABLE where dt=date\_add('$DT',-7);")

# 本周数据量

THISWEEK=$($HIVE\_ENGINE -e "set hive.cli.print.header=false;select count(1) from $HIVE\_DB.$TABLE where dt='$DT';")

# 计算增长

if [ $LASTWEEK -ne 0 ]; then

RESULT=$(awk "BEGIN{print ($THISWEEK-$LASTWEEK)/$LASTWEEK\*100}")

else

RESULT=10000

fi

# 将结果写入MySQL

mysql -h"$mysql\_host" -u"$mysql\_user" -p"$mysql\_passwd" \

-e"INSERT INTO $mysql\_DB.$mysql\_tbl VALUES('$DT', '$TABLE', $RESULT, $MIN, $MAX, $LEVEL)

ON DUPLICATE KEY UPDATE \`value\`=$RESULT, value\_min=$MIN, value\_max=$MAX, notification\_level=$LEVEL;"

### 2.4.2 数仓各层检测脚本编写

将上一节编写的单一规则检测脚本按照数仓分层进行集成，分别**编写ODS层检测脚本，DWD层检测脚本和DIM层检测脚本**。

每层详细集成步骤如下

#### 1. ODS层

ODS层需要检查的指标如下表所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表** | **检查项目** | **依据** | **异常值下限** | **异常值上限** |
| **ods\_order\_info** | 同比增长 | 数据总量 | -10% | 10% |
| 环比增长 | 数据总量 | -10% | 50% |
| 值域检查 | final\_amount | 0 | 100 |

在Idea中创建一个文件check\_ods.sh，在文件中编写如下内容：

#!/usr/bin/env bash

DT=$1

[ "$DT" ] || DT=$(date -d '-1 day' +%F)

#检查表 ods\_order\_info 数据量日环比增长

#参数： -t 表名

# -d 日期

# -s 环比增长下限

# -x 环比增长上限

# -l 告警级别

bash day\_on\_day.sh -t ods\_order\_info -d "$DT" -s -10 -x 10 -l 1

#检查表 ods\_order\_info 数据量周同比增长

#参数： -t 表名

# -d 日期

# -s 同比增长下限

# -x 同比增长上限

# -l 告警级别

bash week\_on\_week.sh -t ods\_order\_info -d "$DT" -s -10 -x 50 -l 1

#检查表 ods\_order\_info 订单异常值

#参数： -t 表名

# -d 日期

# -s 指标下限

# -x 指标上限

# -l 告警级别

# -a 值域下限

# -b 值域上限

bash range.sh -t ods\_order\_info -d "$DT" -c final\_amount -a 0 -b 100000 -s 0 -x 100 -l 1

#### 2. DWD层

DWD层需要检查的项目下标所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表** | **检查项目** | **依据** | **异常值下限** | **异常值上限** |
| **dwd\_order\_info** | 空值检查 | id | 0 | 10 |
| 重复值检查 | id | 0 | 5 |

在Idea中创建一个文件check\_dwd.sh，在文件中编写如下内容：

#!/usr/bin/env bash

DT=$1

[ "$DT" ] || DT=$(date -d '-1 day' +%F)

# 检查表 dwd\_order\_info 重复ID

#参数： -t 表名

# -d 日期

# -c 检查重复值的列

# -s 异常指标下限

# -x 异常指标上限

# -l 告警级别

bash duplicate.sh -t dwd\_order\_info -d "$DT" -c id -s 0 -x 5 -l 0

#检查表 dwd\_order\_info 的空ID

#参数： -t 表名

# -d 日期

# -c 检查空值的列

# -s 异常指标下限

# -x 异常指标上限

# -l 告警级别

bash null\_id.sh -t dwd\_order\_info -d "$DT" -c id -s 0 -x 10 -l 0

#### 3. DIM层

DIM层需要检查的项目如下表所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表** | **检查项目** | **依据** | **异常值下限** | **异常值上限** |
| **dim\_user\_info** | 空值检查 | id | 0 | 10 |
| 重复值检查 | id | 0 | 5 |

在Idea中创建一个文件check\_dim.sh，在文件中编写如下内容：

#!/usr/bin/env bash

DT=$1

[ "$DT" ] || DT=$(date -d '-1 day' +%F)

#检查表 dim\_user\_info 的重复ID

#参数： -t 表名

# -d 日期

# -c 检查重复值的列

# -s 异常指标下限

# -x 异常指标上限

# -l 告警级别

bash duplicate.sh -t dim\_user\_info -d "$DT" -c id -s 0 -x 5 -l 0

#检查表 dim\_user\_info 的空ID

#参数： -t 表名

# -d 日期

# -c 检查空值的列

# -s 异常指标下限

# -x 异常指标上限

# -l 告警级别

bash null\_id.sh -t dim\_user\_info -d "$DT" -c id -s 0 -x 10 -l 0

## 2.5 告警集成模块（可选）

该模块主要用于**检查MySQL中的检测结果的异常**，**若有异常出现就发送警告**。警告方式可选择邮件或者集成第三方告警平台睿象云。

（1）环境准备

在MySQL官网下载mysql-connector-python-2.1.7-1.el7.x86\_64.rpm，下载地址如下：

<https://repo.mysql.com/yum/mysql-connectors-community/el/7/x86_64/mysql-connector-python-2.1.7-1.el7.x86_64.rpm>



将该rpm包上传至每台服务器，并安装：

[atguigu@hadoop102 ~]$ **sudo rpm -i mysql-connector-python-2.1.7-1.el7.x86\_64.rpm**

[atguigu@hadoop102 ~]$ **sudo rpm -qa|grep mysql-connector-python**

（2）新建python脚本用于查询数据监控结果表格并发送告警邮件，该脚本主要由三个函数组成：

* read\_table用于读取指标有问题的数据
* one\_alert函数用于向睿象云发送告警
* mail\_alert函数用于发送邮件告警

在Idea中创建一个文件check\_notification.py，在文件中编写如下内容：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import mysql.connector

import sys

import smtplib

from email.mime.text import MIMEText

from email.header import Header

import datetime

import urllib

import urllib2

import random

def get\_yesterday():

"""

:return: 前一天的日期

"""

today = datetime.date.today()

one\_day = datetime.timedelta(days=1)

yesterday = today - one\_day

return str(yesterday)

def read\_table(table, dt):

"""

:param table:读取的表名

:param dt:读取的数据日期

:return:表中的异常数据(统计结果超出规定上下限的数据)

"""

# mysql必要参数设置，需根据实际情况作出修改

mysql\_user = "root"

mysql\_password = "000000"

mysql\_host = "hadoop102"

mysql\_schema = "data\_supervisor"

# 获取Mysql数据库连接

connect = mysql.connector.connect(user=mysql\_user, password=mysql\_password, host=mysql\_host, database=mysql\_schema)

cursor = connect.cursor()

# 查询表头

# ['dt', 'tbl', 'col', 'value', 'value\_min', 'value\_max', 'notification\_level']

query = "desc " + table

cursor.execute(query)

head = map(lambda x: str(x[0]), cursor.fetchall())

# 查询异常数据(统计结果超出规定上下限的数据)

# [(datetime.date(2021, 7, 16), u'dim\_user\_info', u'id', 7, 0, 5, 1),

# (datetime.date(2021, 7, 16), u'dwd\_order\_id', u'id', 10, 0, 5, 1)]

query = ("select \* from " + table + " where dt='" + dt + "' and `value` not between value\_min and value\_max")

cursor.execute(query)

cursor\_fetchall = cursor.fetchall()

# 将指标和表头映射成为dict数组

#[{'notification\_level': 1, 'value\_min': 0, 'value': 7, 'col': u'id', 'tbl': u'dim\_user\_info', 'dt': datetime.date(2021, 7, 16), 'value\_max': 5},

# {'notification\_level': 1, 'value\_min': 0, 'value': 10, 'col': u'id', 'tbl': u'dwd\_order\_id', 'dt': datetime.date(2021, 7, 16), 'value\_max': 5}]

fetchall = map(lambda x: dict(x), map(lambda x: zip(head, x), cursor\_fetchall))

return fetchall

def one\_alert(line):

"""

集成第三方告警平台睿象云，使用其提供的通知媒介发送告警信息

:param line: 一个等待通知的异常记录，{'notification\_level': 1, 'value\_min': 0, 'value': 7, 'col': u'id', 'tbl': u'dim\_user\_info', 'dt': datetime.date(2021, 7, 16), 'value\_max': 5}

"""

# 集成睿象云需要使用的rest接口，和APP KEY，须在睿象云平台获取

one\_alert\_key = "c2030c9a-7896-426f-bd64-59a8889ac8e3"

one\_alert\_host = "http://api.aiops.com/alert/api/event"

# 根据睿象云的rest api要求，传入必要的参数

data = {

"app": one\_alert\_key,

"eventType": "trigger",

"eventId": str(random.randint(10000, 99999)),

"alarmName": "".join(["表格", str(line["tbl"]), "数据异常."]),

"alarmContent": "".join(["指标", str(line["norm"]), "值为", str(line["value"]),

", 应为", str(line["value\_min"]), "-", str(line["value\_max"]),

", 参考信息：" + str(line["col"]) if line.get("col") else ""]),

"priority": line["notification\_level"] + 1

}

# 使用urllib和urllib2向睿象云的rest结构发送请求，从而触发睿象云的通知策略

body = urllib.urlencode(data)

request = urllib2.Request(one\_alert\_host, body)

urlopen = urllib2.urlopen(request).read().decode('utf-8')

print urlopen

def mail\_alert(line):

"""

使用电子邮件的方式发送告警信息

:param line: 一个等待通知的异常记录，{'notification\_level': 1, 'value\_min': 0, 'value': 7, 'col': u'id', 'tbl': u'dim\_user\_info', 'dt': datetime.date(2021, 7, 16), 'value\_max': 5}

"""

# smtp协议发送邮件的必要设置

mail\_host = "smtp.126.com"

mail\_user = "skiinder@126.com"

mail\_pass = "KADEMQZWCPFWZETF"

# 告警内容

message = ["".join(["表格", str(line["tbl"]), "数据异常."]),

"".join(["指标", str(line["norm"]), "值为", str(line["value"]),

", 应为", str(line["value\_min"]), "-", str(line["value\_max"]),

", 参考信息：" + str(line["col"]) if line.get("col") else ""])]

# 告警邮件，发件人

sender = mail\_user

# 告警邮件，收件人

receivers = [mail\_user]

# 将邮件内容转为html格式

mail\_content = MIMEText("".join(["<html>", "<br>".join(message), "</html>"]), "html", "utf-8")

mail\_content["from"] = sender

mail\_content["to"] = receivers[0]

mail\_content["Subject"] = Header(message[0], "utf-8")

# 使用smtplib发送邮件

try:

smtp = smtplib.SMTP\_SSL()

smtp.connect(mail\_host, 465)

smtp.login(mail\_user, mail\_pass)

content\_as\_string = mail\_content.as\_string()

smtp.sendmail(sender, receivers, content\_as\_string)

except smtplib.SMTPException as e:

print e

def main(argv):

"""

:param argv: 系统参数，共三个，第一个为python脚本本身，第二个为告警方式，第三个为日期

"""

# 如果没有传入日期参数，将日期定为昨天

if len(argv) >= 3:

dt = argv[2]

else:

dt = get\_yesterday()

notification\_level = 0

# 通过参数设置告警方式，默认是睿象云

alert = None

if len(argv) >= 2:

alert = {

"mail": mail\_alert,

"one": one\_alert

}[argv[1]]

if not alert:

alert = one\_alert

# 遍历所有表，查询所有错误内容，如果大于设定警告等级，就发送警告

for table in ["day\_on\_day", "duplicate", "null\_id", "rng", "week\_on\_week"]:

for line in read\_table(table, dt):

if line["notification\_level"] >= notification\_level:

line["norm"] = table

alert(line)

if \_\_name\_\_ == "\_\_main\_\_":

# 两个命令行参数

# 第一个为警告类型：one或者mail

# 第二个为日期，留空取昨天

main(sys.argv)

## 2.6 调度模块（可选）

该模块的主要功能为**调度数据质量监控流程。**数据质量监控工作流也采用Azkaban进行调度。数据质量监控工作流必定依赖数据仓库工作流，此处为了解耦，**利用Azkaban API主动监视数据仓库工作流的执行状态**，进而触发数据质量监控工作流。

以下是所有脚本内容：

### 2.6.1 Azkaban REST API 封装脚本

该脚本主要是对Azkaban API的封装，主要有三个方法：

* login函数可以登录Azkanban并返回session\_id
* get\_exec\_id函数可以获取正在执行的工作流程的Execution ID
* wait\_node可以等待指定Flow中某一结点执行完毕并判断其是否执行成功

在Idea中创建一个文件azclient.py，在文件中编写如下内容：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import time

import urllib

import urllib2

import json

# Azkaban API 接口地址

az\_url = "http://hadoop102:8081/"

# Azkaban用户名

az\_username = "atguigu"

# Azkaban密码

az\_password = "atguigu"

# 工程名称

project = "gmall"

# flow名称

flow = "gmall"

def post(url, data):

"""

发送post请求到指定网址

:param url: 指定网址

:param data: 请求参数

:return: 请求结果

"""

body = urllib.urlencode(data)

request = urllib2.Request(url, body)

urlopen = urllib2.urlopen(request).read().decode('utf-8')

return json.loads(urlopen)

def get(url, data):

"""

发送get请求到指定网址

:param url: 指定网址

:param data: 请求参数

:return: 请求结果

"""

body = urllib.urlencode(data)

urlopen = urllib2.urlopen(url + body).read().decode('utf-8')

return json.loads(urlopen)

def login():

"""

使用`Authenticate`API进行azkaban身份认证，获取session ID

:return: 返回session\_id

"""

data = {

"action": "login",

"username": az\_username,

"password": az\_password

}

auth = post(az\_url, data)

return str(auth.get(u"session.id"))

def get\_exec\_id(session\_id):

"""

使用`Fetch Running Executions of a Flow`API获取正在执行的Flow的ExecId

:param session\_id: 和azkaban通讯的session\_id

:param project: 项目名称

:param flow: 工作流名称

:return: 执行ID

"""

data = {

"session.id": session\_id,

"ajax": "getRunning",

"project": project,

"flow": flow

}

execs = get(az\_url + "executor?", data).get(u"execIds")

if execs:

return str(execs[0])

else:

return None

def wait\_node(session\_id, exec\_id, node\_id):

"""

循环使用`Fetch a Flow Execution`API获取指定Flow中的某个节点(job)的执行状态，直到其执行完成

:param session\_id: 和azkaban通讯的session\_id

:param exec\_id: 执行ID

:param node\_id: 指定节点(job)

:return: 该节点是否成功执行完毕

"""

data = {

"session.id": session\_id,

"ajax": "fetchexecflow",

"execid": exec\_id

}

status = None

# 若指定Flow中的指定Node(job)的执行状态是未完成的状态，就一直循环

while status not in ["SUCCEEDED", "FAILED", "CANCELLED", "SKIPPED", "KILLED"]:

# 获取指定Flow的当前的执行信息

flow\_exec = get(az\_url + "executor?", data)

# 从该Flow的执行信息中获取nodes字段的值，并遍历寻找特定的节点(job)信息，进而获取该节点(job)的状态

for node in flow\_exec.get(u"nodes"):

if unicode(node\_id) == node.get(u"id"):

status = str(node.get(u"status"))

print " ".join([node\_id, status])

# 等待1s，进入下一轮循环判断

time.sleep(1)

return status == "SUCCEEDED"

### 2.6.2 ODS层调度脚本

该脚本用于检查ODS层数据质量。

在Idea中创建一个文件check\_ods.py，在文件中编写如下内容：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import sys

import os

from azclient import login,wait\_node,get\_exec\_id

from check\_notification import get\_yesterday

def check\_ods(dt, session\_id, exec\_id):

"""

检查ODS层数据质量

:param dt: 日期

:param session\_id: 和azkaban通讯的session\_id

:param exec\_id: 指定的执行ID

:return: None

"""

if wait\_node(session\_id, exec\_id, "hdfs\_to\_ods\_db") and wait\_node(session\_id, exec\_id, "hdfs\_to\_ods\_log"):

os.system("bash check\_ods.sh " + dt)

if \_\_name\_\_ == '\_\_main\_\_':

argv = sys.argv

# 获取session\_id

session\_id = login()

# 获取执行ID。只有在原Flow正在执行时才能获取

exec\_id = get\_exec\_id(session\_id)

# 获取日期，如果不存在取昨天

if len(argv) >= 2:

dt = argv[1]

else:

dt = get\_yesterday()

# 检查各层数据质量

if exec\_id:

check\_ods(dt, session\_id, exec\_id)

### 2.6.3 DWD层调度脚本

该脚本用于检查DWD层数据质量。

在Idea中创建一个文件check\_dwd.py，在文件中编写如下内容：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import sys

import os

from azclient import login, wait\_node, get\_exec\_id

from check\_notification import get\_yesterday

def check\_dwd(dt, session\_id, exec\_id):

"""

检查DWD层数据质量

:param dt: 日期

:param session\_id: 和azkaban通讯的session\_id

:param exec\_id: 指定的执行ID

:return: None

"""

if wait\_node(session\_id, exec\_id, "ods\_to\_dwd\_db") and wait\_node(session\_id, exec\_id, "ods\_to\_dwd\_log"):

os.system("bash check\_dwd.sh " + dt)

if \_\_name\_\_ == '\_\_main\_\_':

argv = sys.argv

# 获取session\_id

session\_id = login()

# 获取执行ID。只有在原Flow正在执行时才能获取

exec\_id = get\_exec\_id(session\_id)

# 获取日期，如果不存在取昨天

if len(argv) >= 2:

dt = argv[1]

else:

dt = get\_yesterday()

# 检查各层数据质量

if exec\_id:

check\_dwd(dt, session\_id, exec\_id)

### 2.6.4 DIM层调度脚本

该脚本用于检查DIM层数据质量。

在Idea中创建一个文件check\_dim.py，在文件中编写如下内容：

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import sys

import os

from azclient import login, wait\_node, get\_exec\_id

from check\_notification import get\_yesterday

def check\_dim(dt, session\_id, exec\_id):

"""

检查DIM层数据质量

:param dt: 日期

:param session\_id: 和azkaban通讯的session\_id

:param exec\_id: 指定的执行ID

:return: None

"""

if wait\_node(session\_id, exec\_id, "ods\_to\_dim\_db"):

os.system("bash check\_dim.sh " + dt)

if \_\_name\_\_ == '\_\_main\_\_':

argv = sys.argv

# 获取session\_id

session\_id = login()

# 获取执行ID。只有在原Flow正在执行时才能获取

exec\_id = get\_exec\_id(session\_id)

# 获取日期，如果不存在取昨天

if len(argv) >= 2:

dt = argv[1]

else:

dt = get\_yesterday()

# 检查各层数据质量

if exec\_id:

check\_dim(dt, session\_id, exec\_id)

### 2.6.5 Azkaban工作流配置文件

（1）在Idea中创建一个文件azkaban.project，在文件中编写如下内容：

azkaban-flow-version: 2.0

（2）在Idea中创建一个文件data\_supervisor.flow，在文件中编写如下内容：

nodes:

- name: check\_ods

type: command

config:

command: python check\_ods.py ${dt}

- name: check\_dwd

type: command

config:

command: python check\_dwd.py ${dt}

- name: check\_dim

type: command

config:

command: python check\_dim.py ${dt}

- name: check\_notification

type: command

dependsOn:

- check\_ods

- check\_dwd

- check\_dim

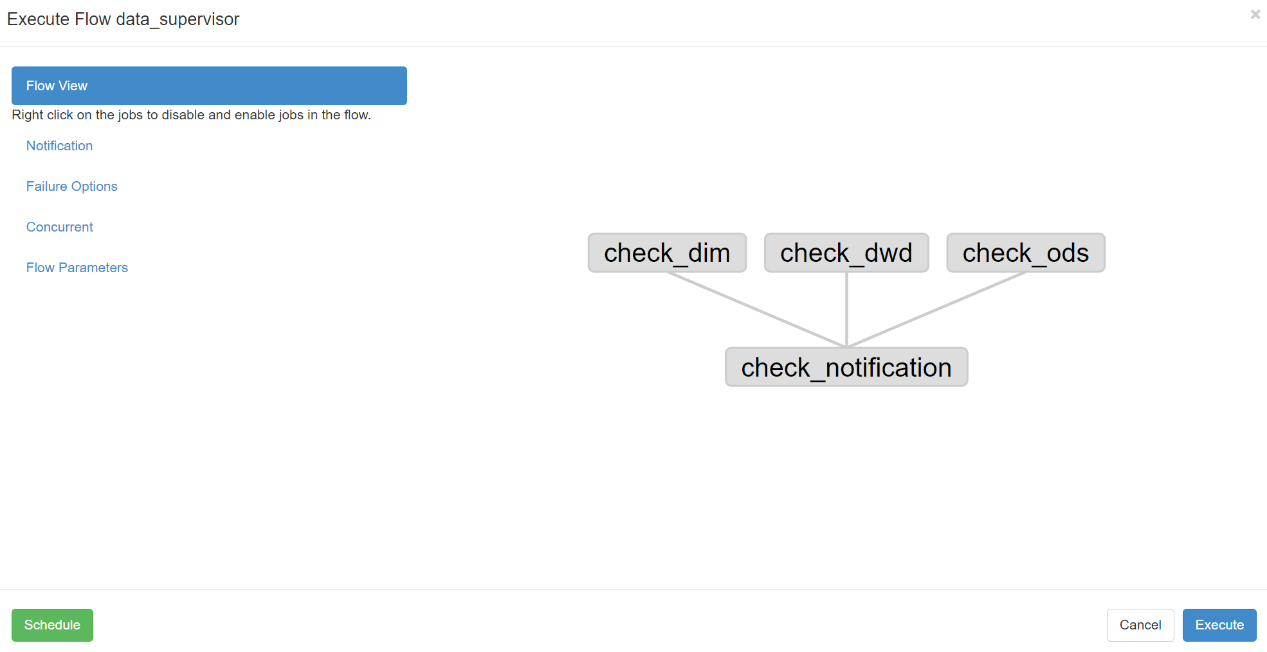
config:

command: python check\_notification.py ${alert} ${dt}

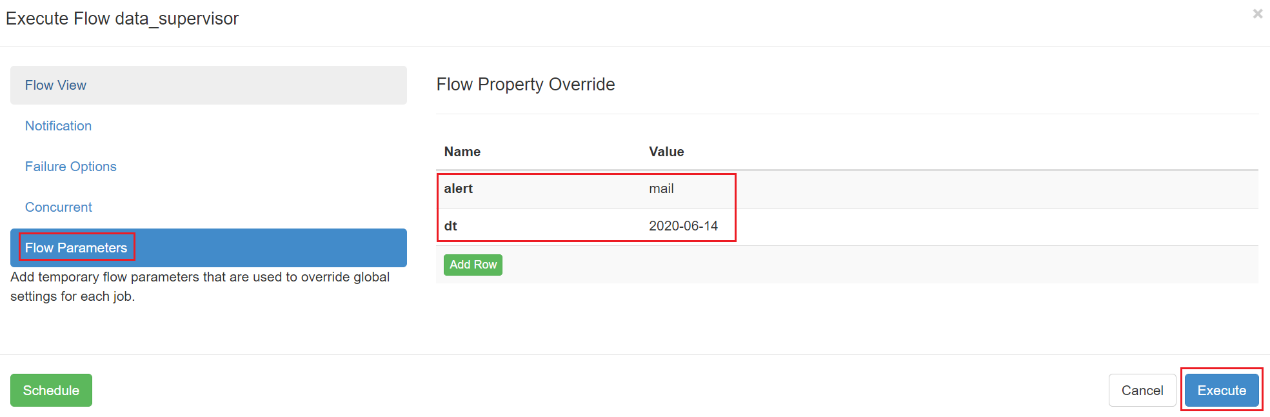
（3）将所有文件打包成data\_supervisor.zip文件



（4）在Azkaban框架中新建项目并上传该文件，可看到如下图所示工作流。



（5）先启动数仓工作流，在执行过程中，启动质量监控工作流，并传入如下参数



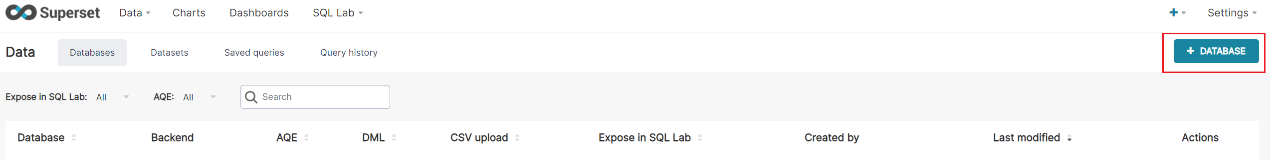
等待任务执行完毕，观察邮箱是否有告警邮件

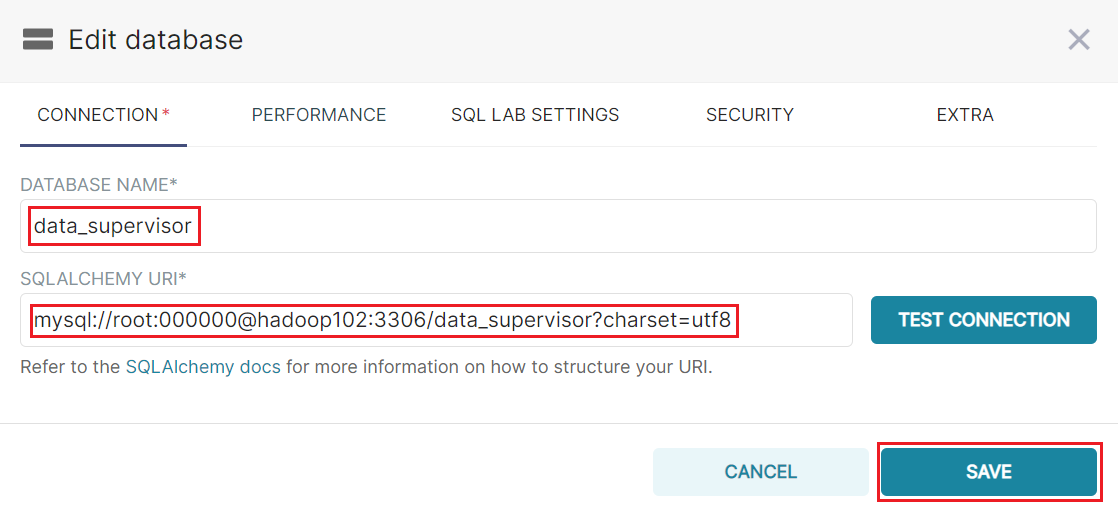
## 2.7 可视化模块

该模块的主要作用是对数据质量监控结果进行可视化展示。

检测结果可以采用Superset进行可视化展示。具体配置步骤如下：

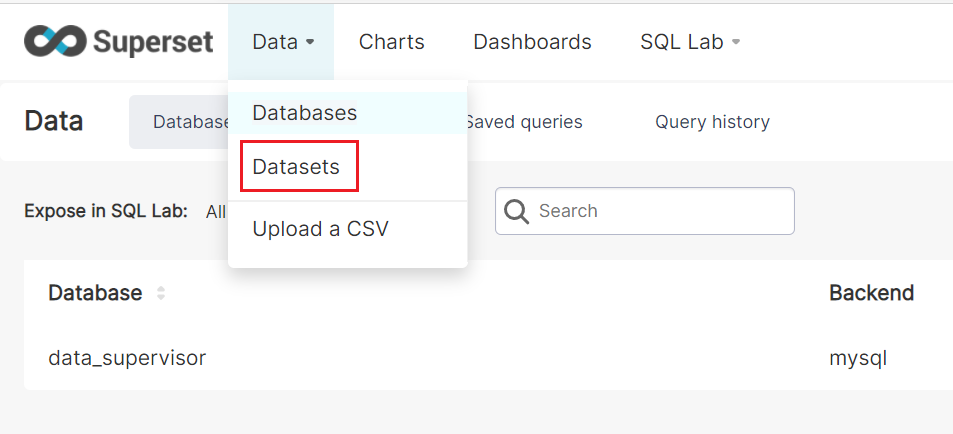
（1）在Superset中新建数据库连接

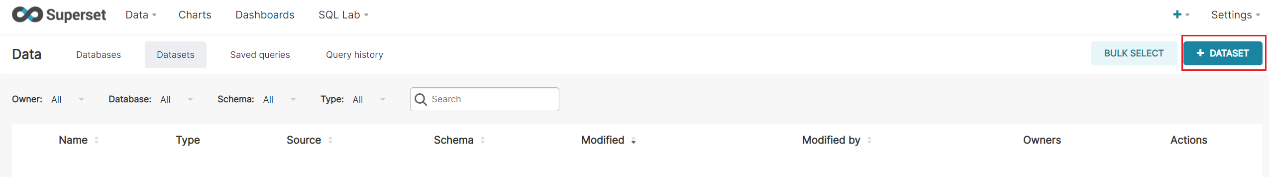


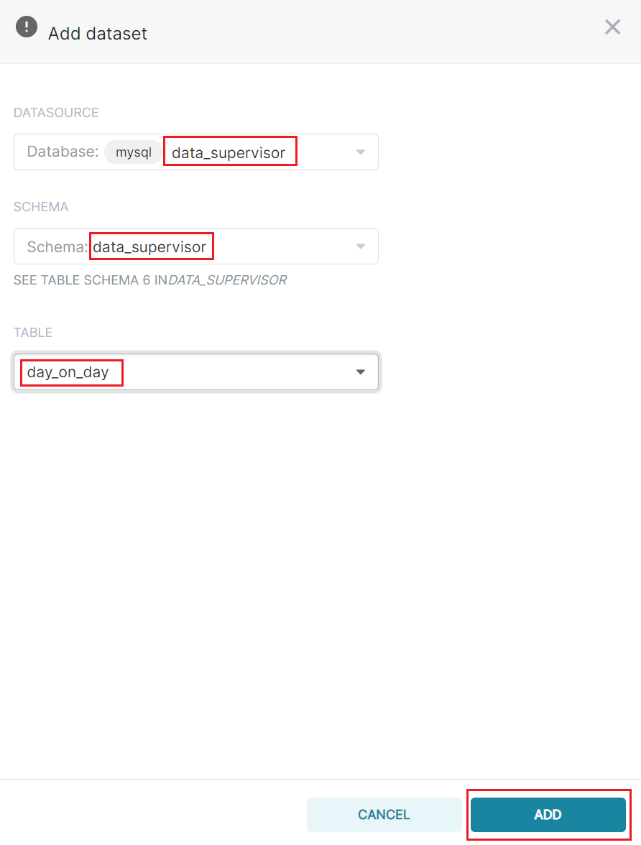


注：**mysql://root:000000@hadoop102:3306/data\_supervisor?charset=utf8**

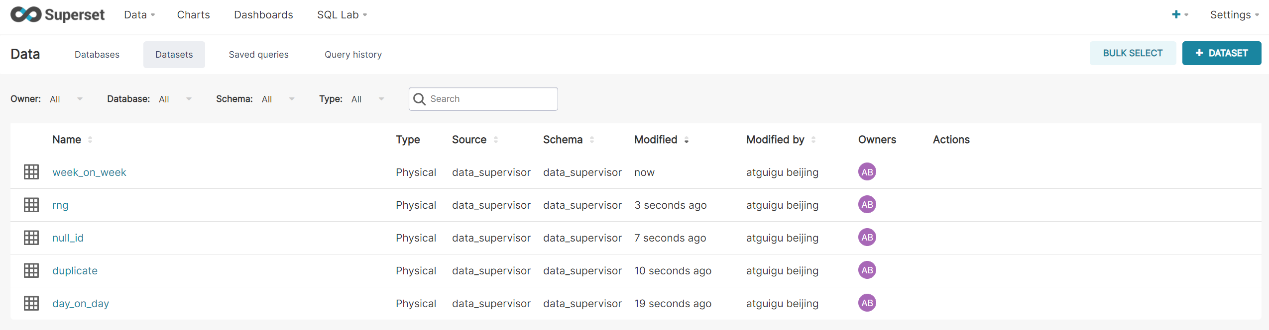
（2）点击“Datasets”，然后点击“+Dataset”，将所有数据表格都导入为dataset



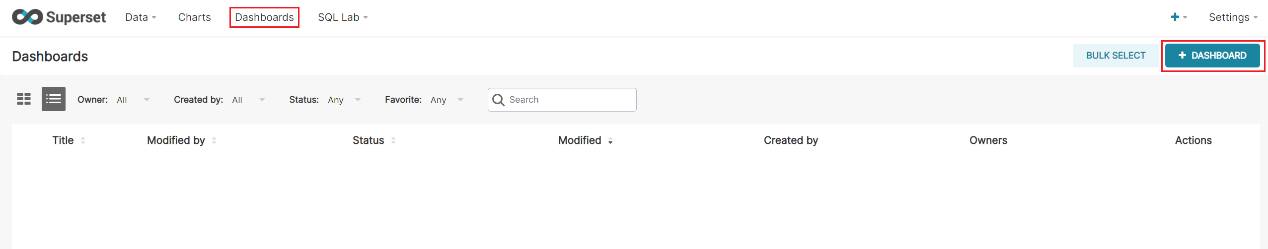


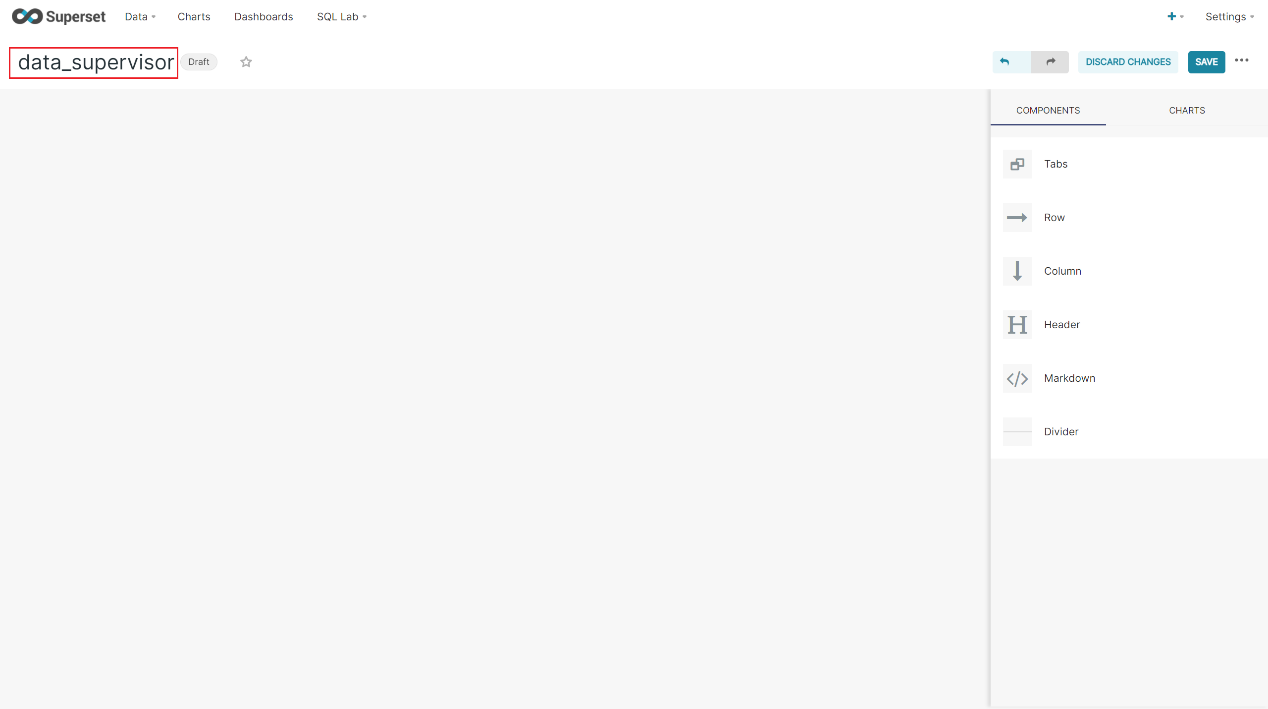


五张表格全部添加后，如下图所示。

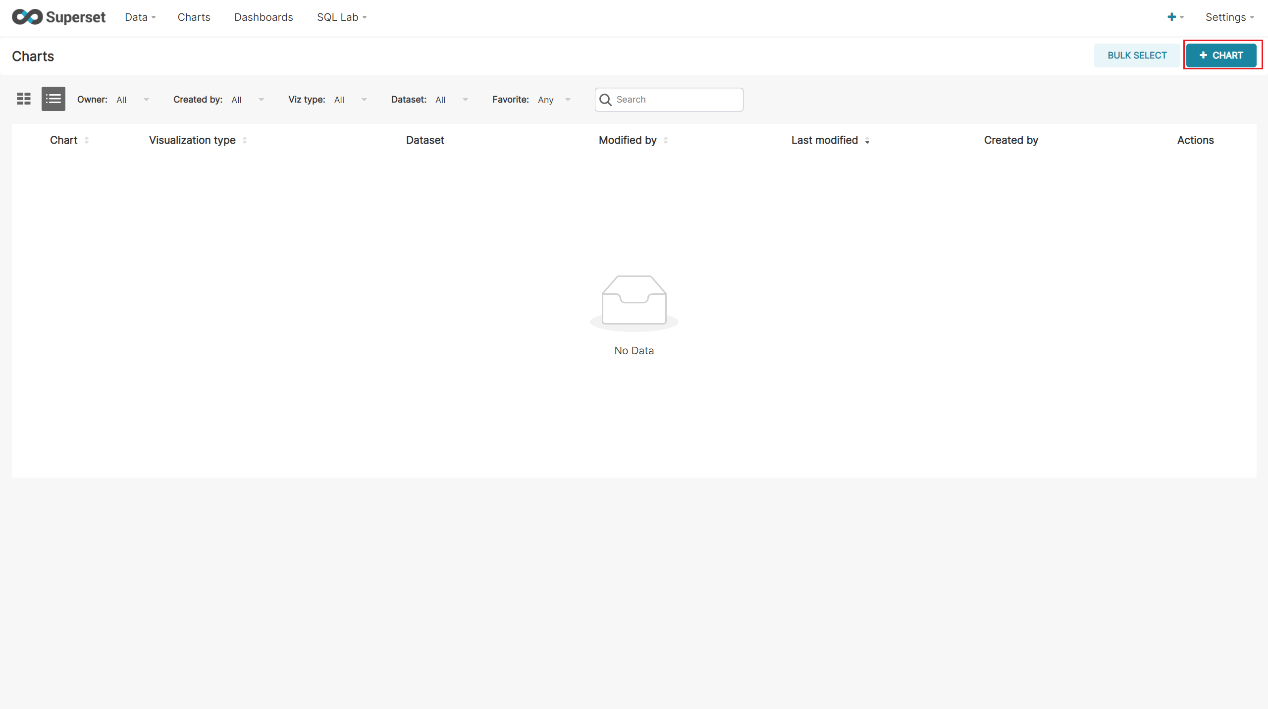


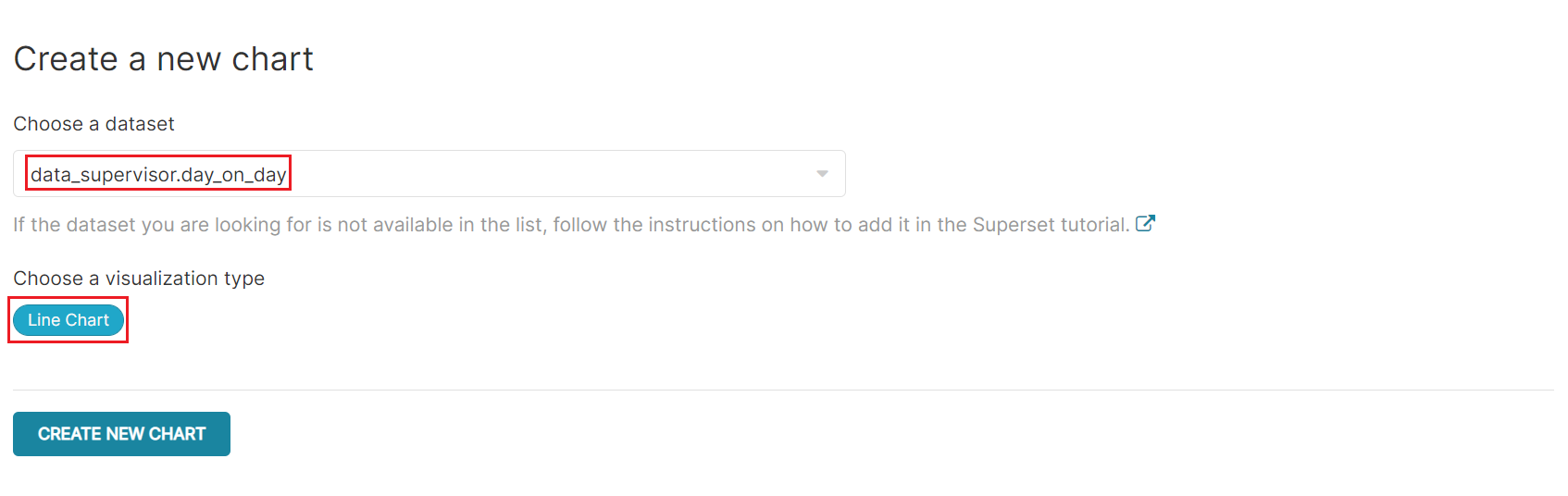
（3）新建一个dashboard，并命名，如下图所示。



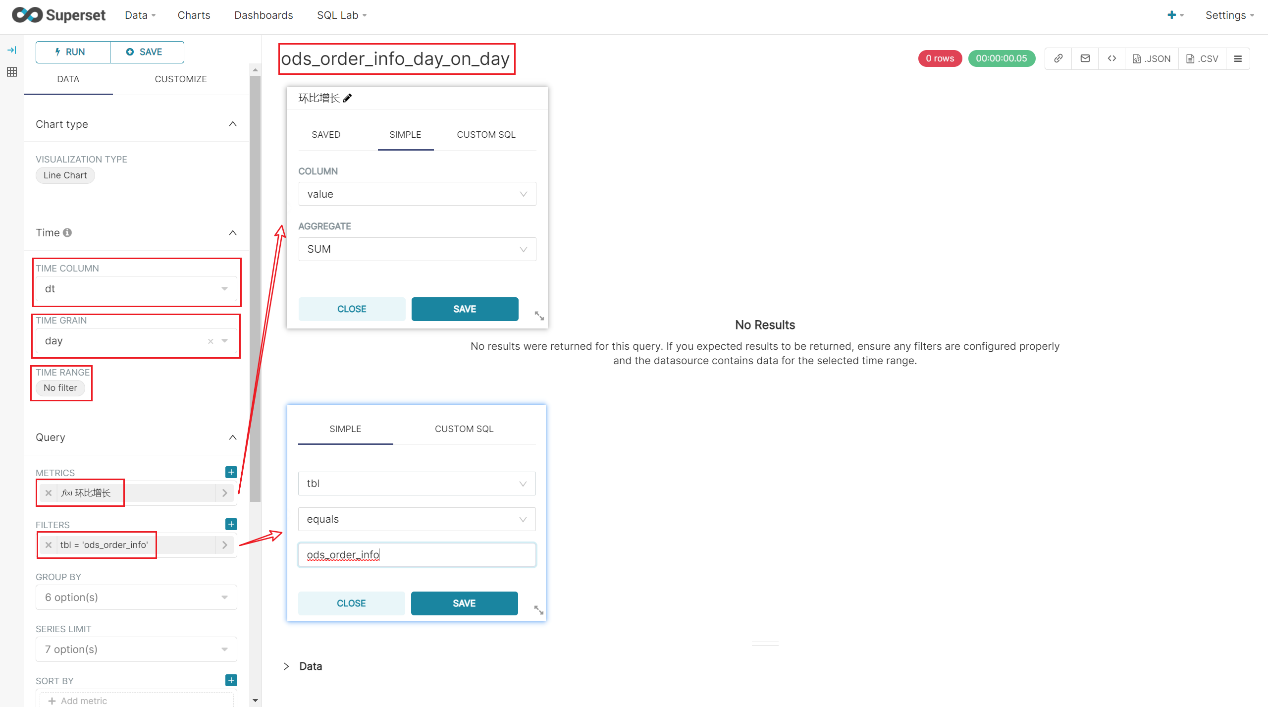


（4）新建一张图表并保存到dashboard，在chart页面中选择新建chart，如下图所示。

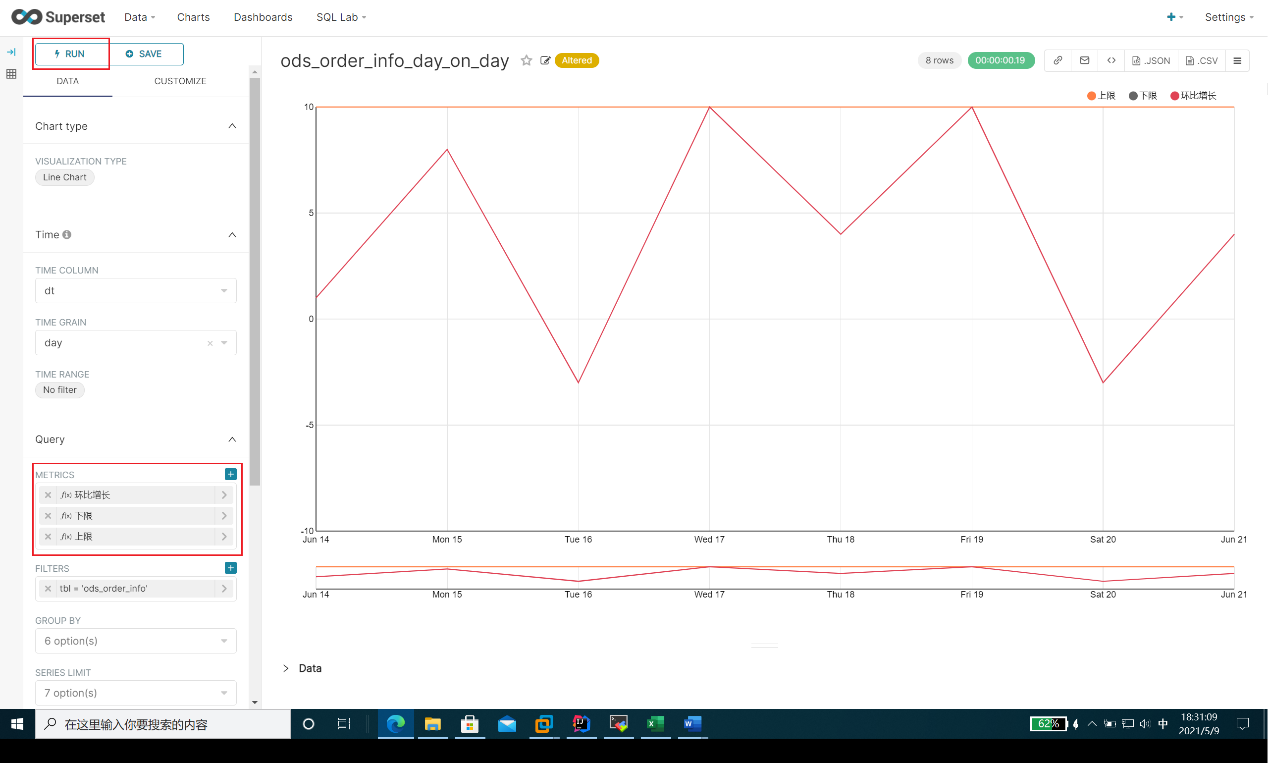




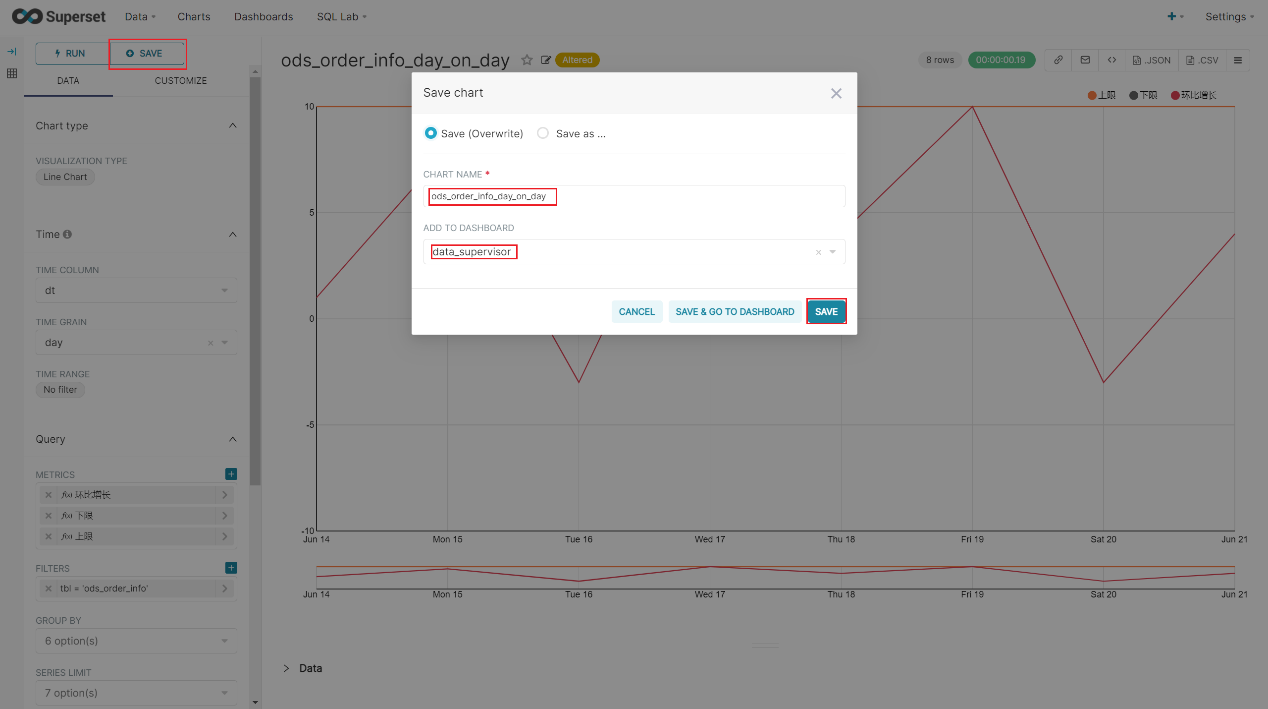
配置chart内容，如下图所示。



在Metrics中添加value，value\_min, value\_max三列，然后点击“run”，就完成了chart的配置流程，如下图所示。



点击“save”，保存chart到dashboard，如下图所示。



（5）为所有监控的指标创建图表，并保存到dashboard，如下图所示。

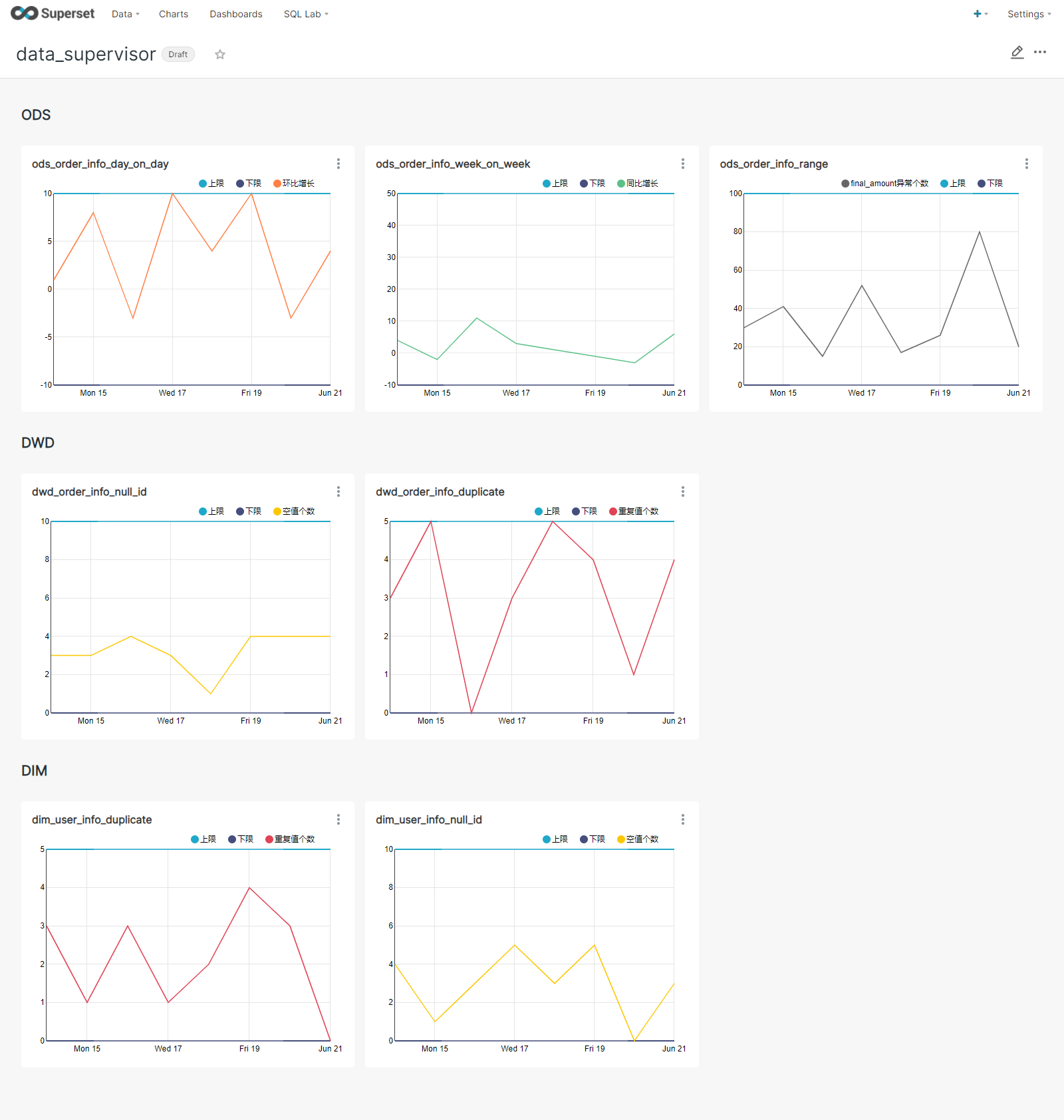


图14-31 保存chart到dashboard