

尚硅谷大数据项目之尚品汇（数据质量管理）

(作者：尚硅谷研究院)

版本：4.0

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（Data Quality Management），是指对数据从计划、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
精确度	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	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_order_info	空值检查	id	0	10
	重复值检查	id	0	5
dim_user_info	空值检查	id	0	10
	重复值检查	id	0	5

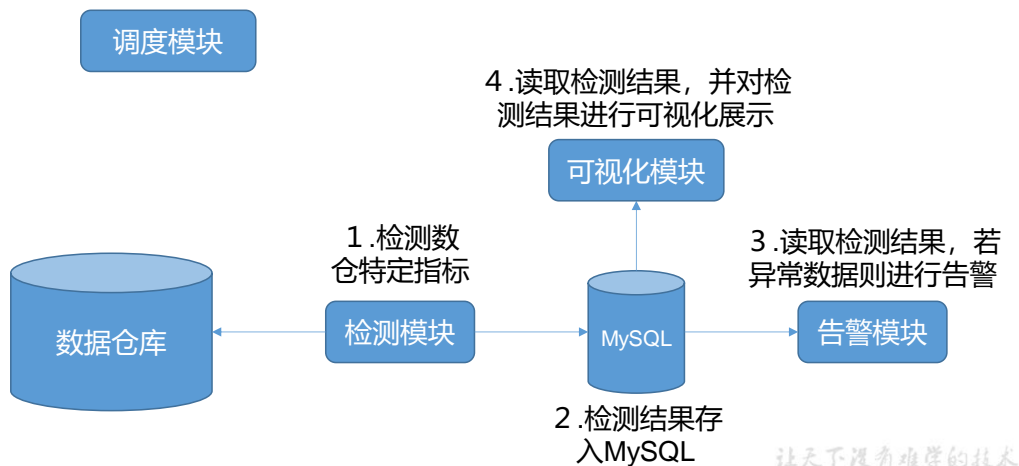
2.2 功能模块



功能模块



5.调度整个检测流程



2.3 开发环境准备

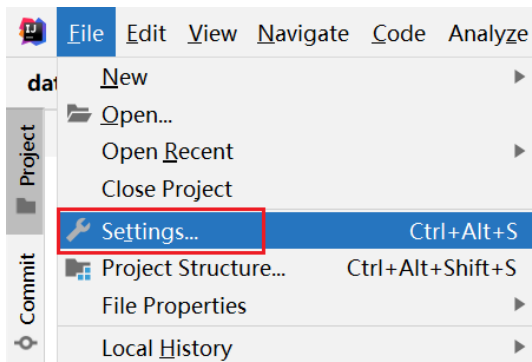
2.3.1 Python 开发环境准备

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

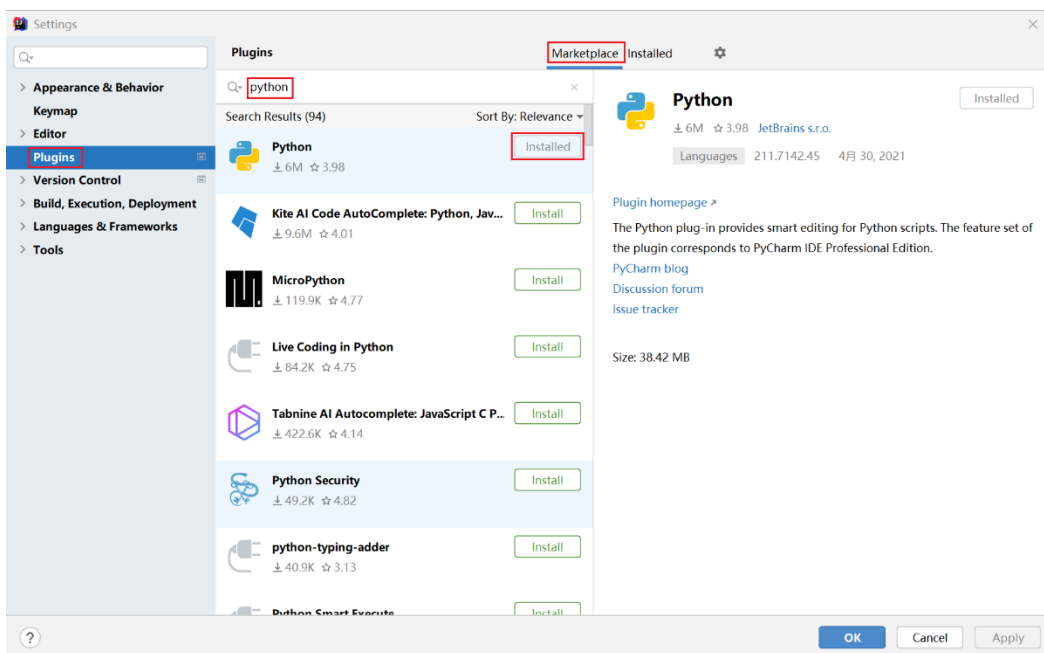
境，Python 开发可选择 IDEA（需安装 Python 插件），或 PyCharm 等工具，本文使用 IDEA 作为开发工具。

1. 安装 Python 插件

(1) 在 IDEA 中点击“File”，在下拉选择中点击“Settings...”

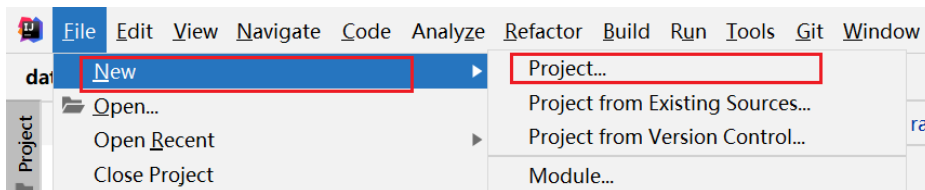


(2) 点击“Plugins”，点击右上角的“Marketplace”，然后在搜索框中输入“python”，在搜索结果列表中找到 Python 插件，点击“Install”，安装插件。

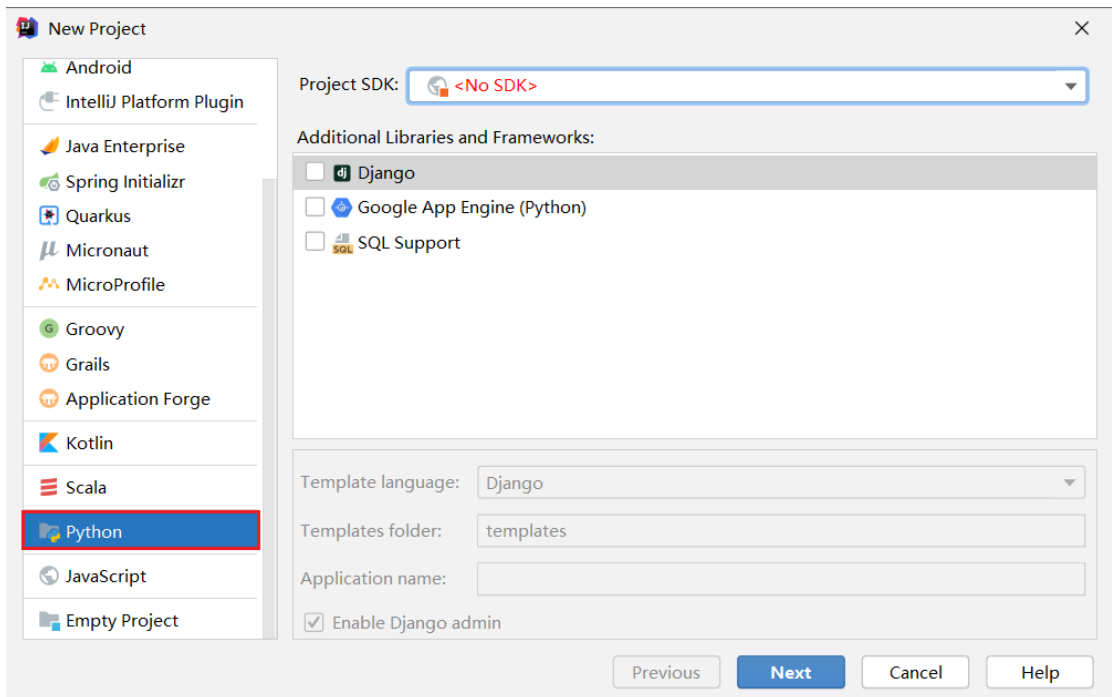


2. 新建一个 Python 项目

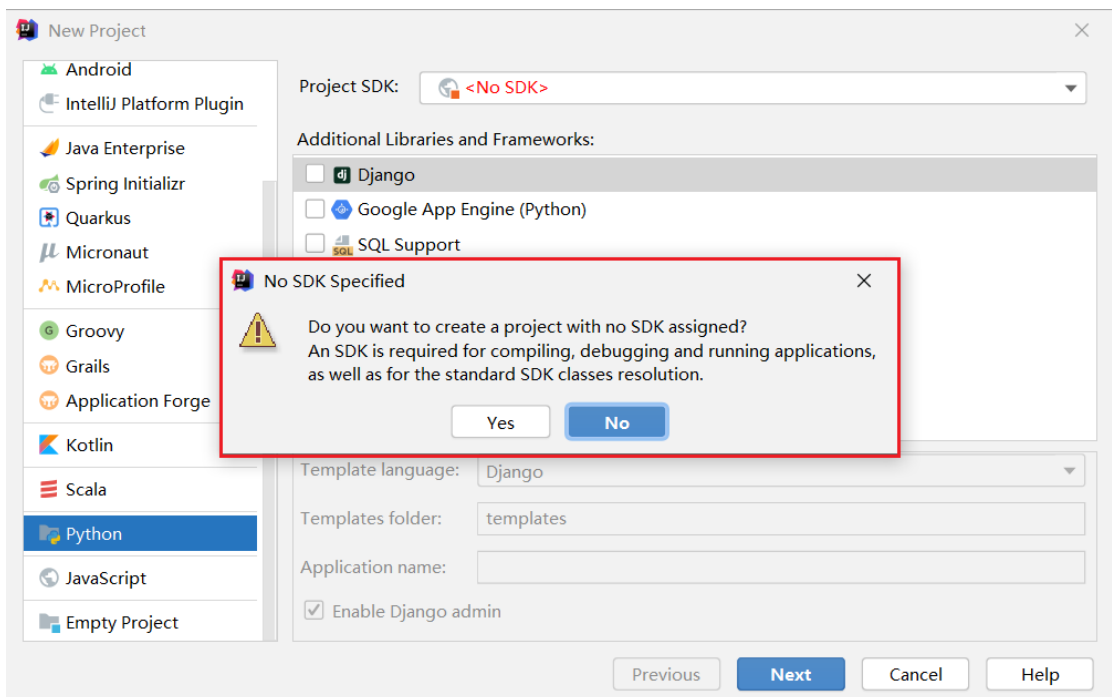
(1) 点击 Idea 中的“File”，在下列列表中点击“New”，在右侧弹出的列表中点击“Project...”



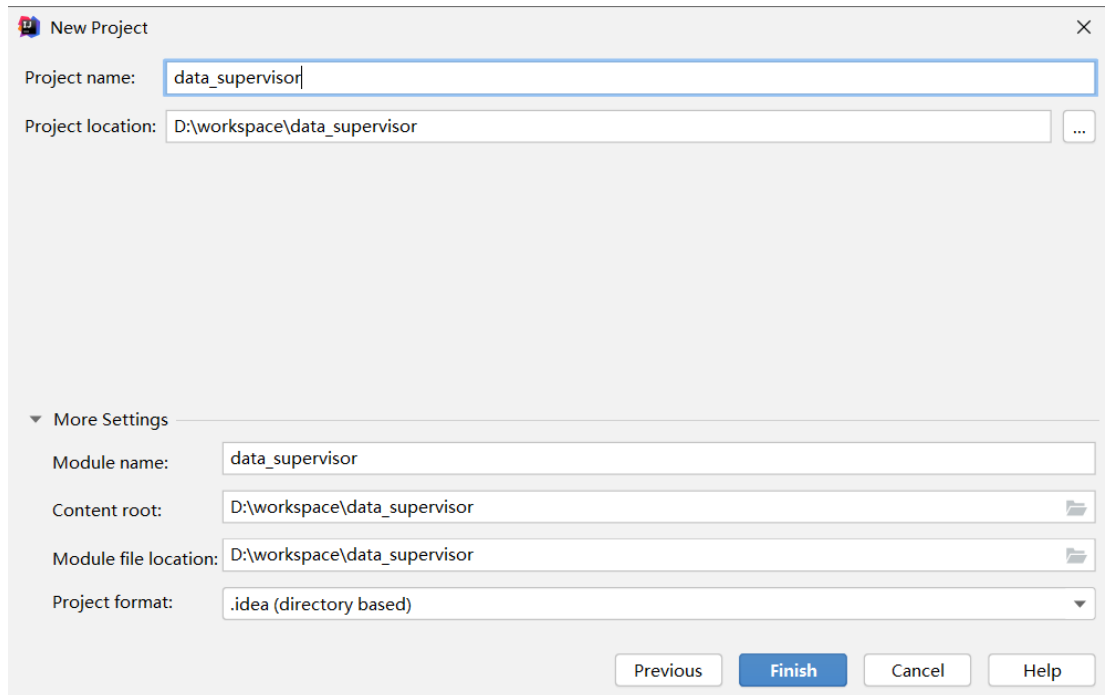
(2) 在新建的工程中，点击“Python”，然后点击 Next



(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。



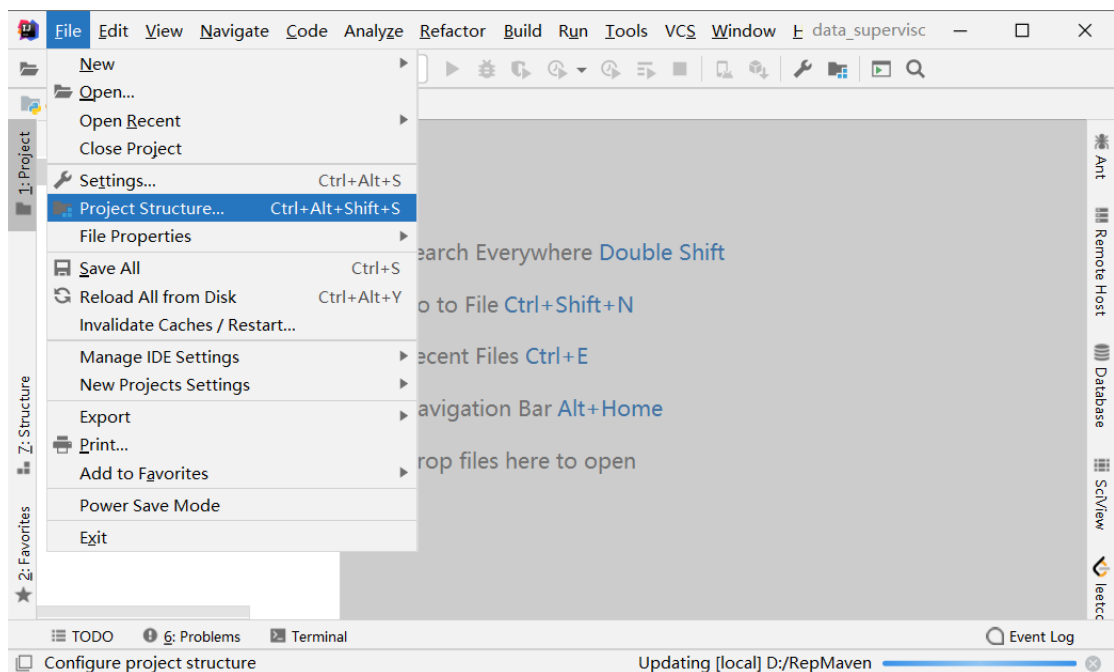
(4) 填写项目名称和项目路径等基本信息，点击 Finish



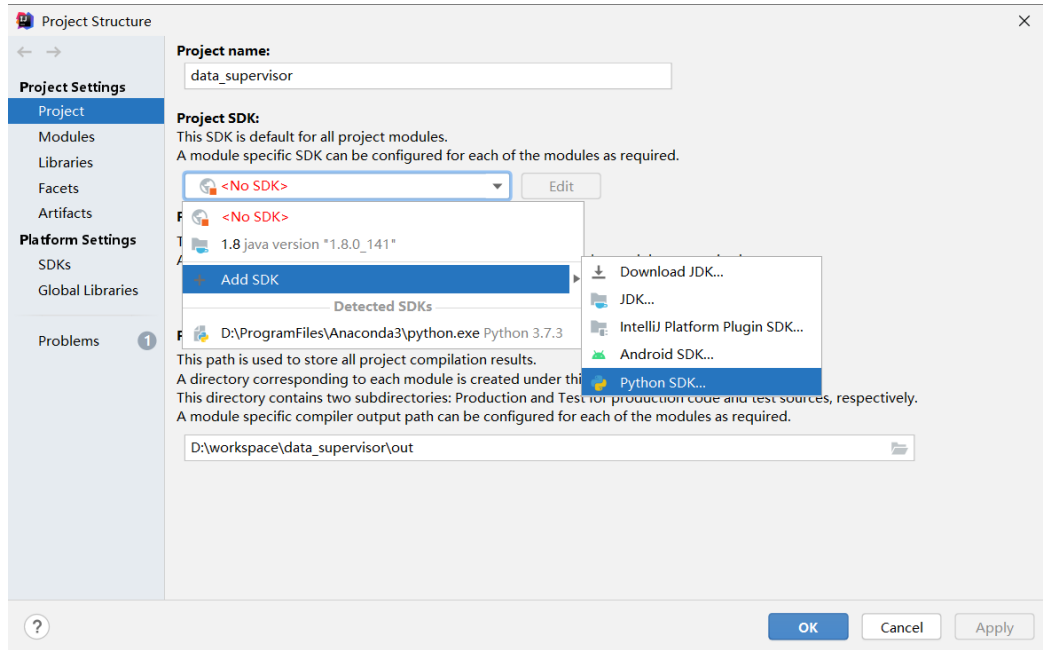
（5）添加 Python SDK

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。

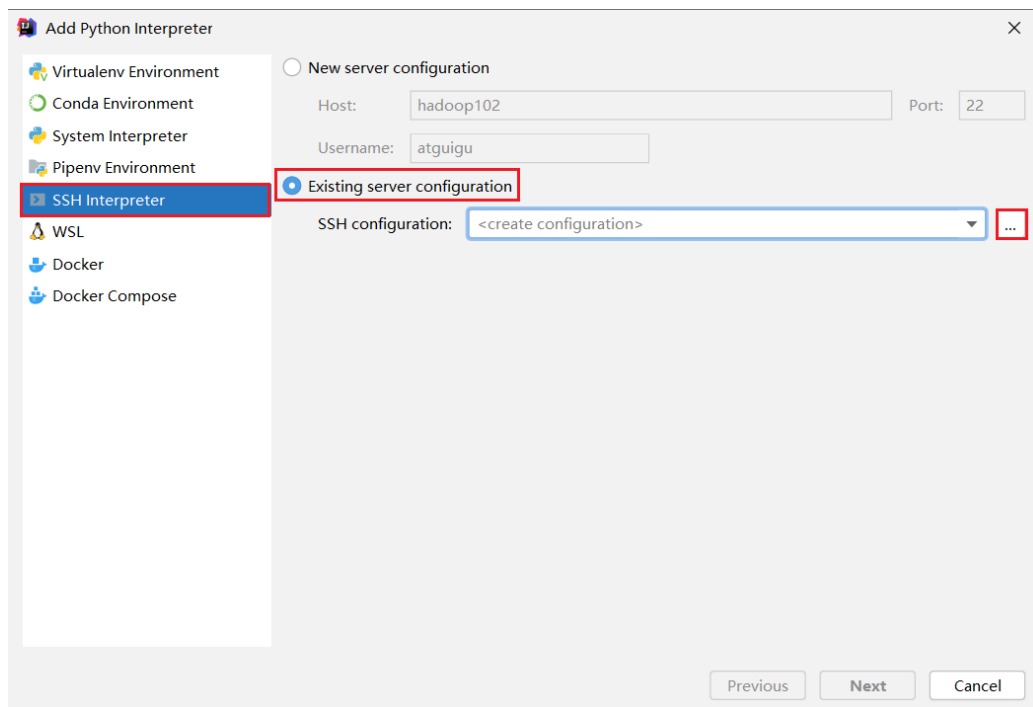
第一步：点击“File”→“Project Structure”



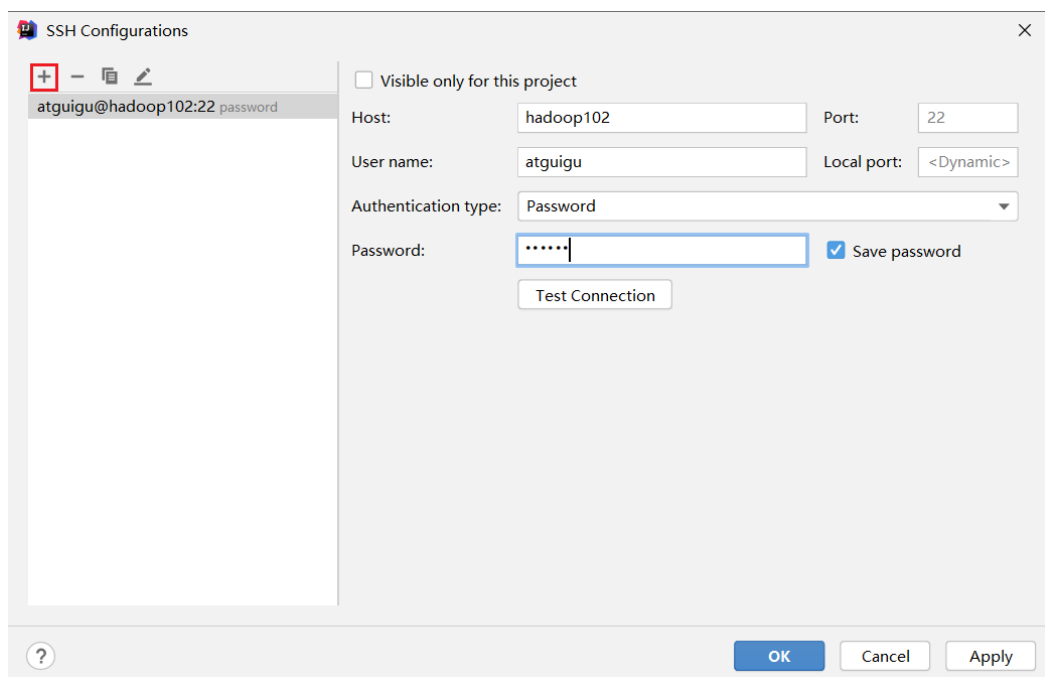
第二步：按照下图操作，增加 Python SDK。



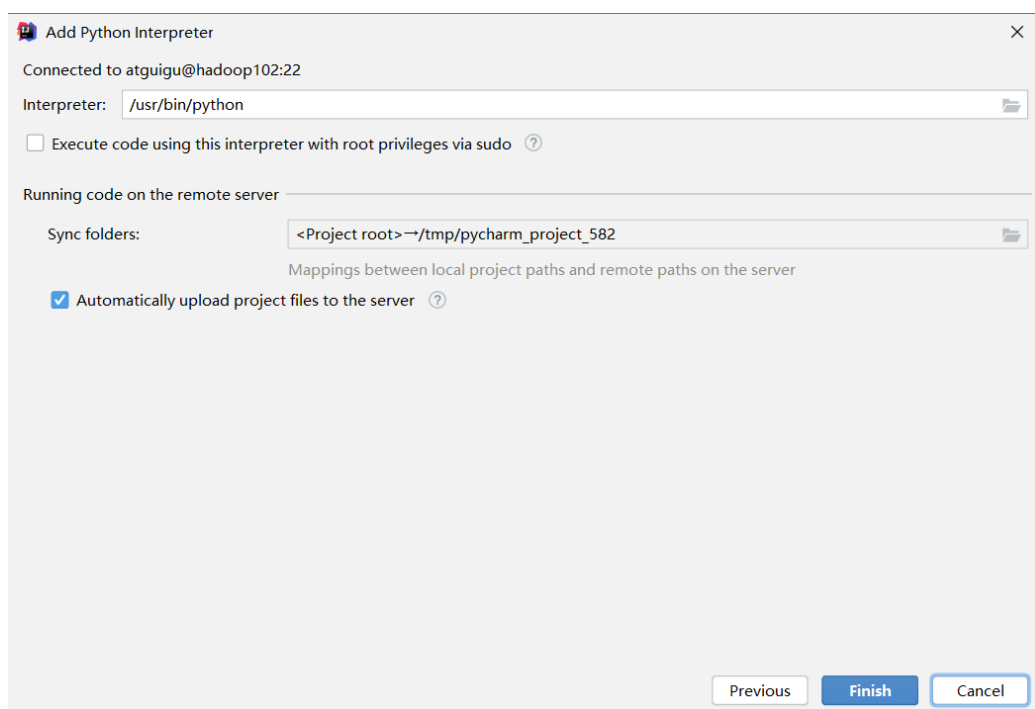
第三步：点击“SSH Interpreter”，选择“Existing server configuration”，点击“...”



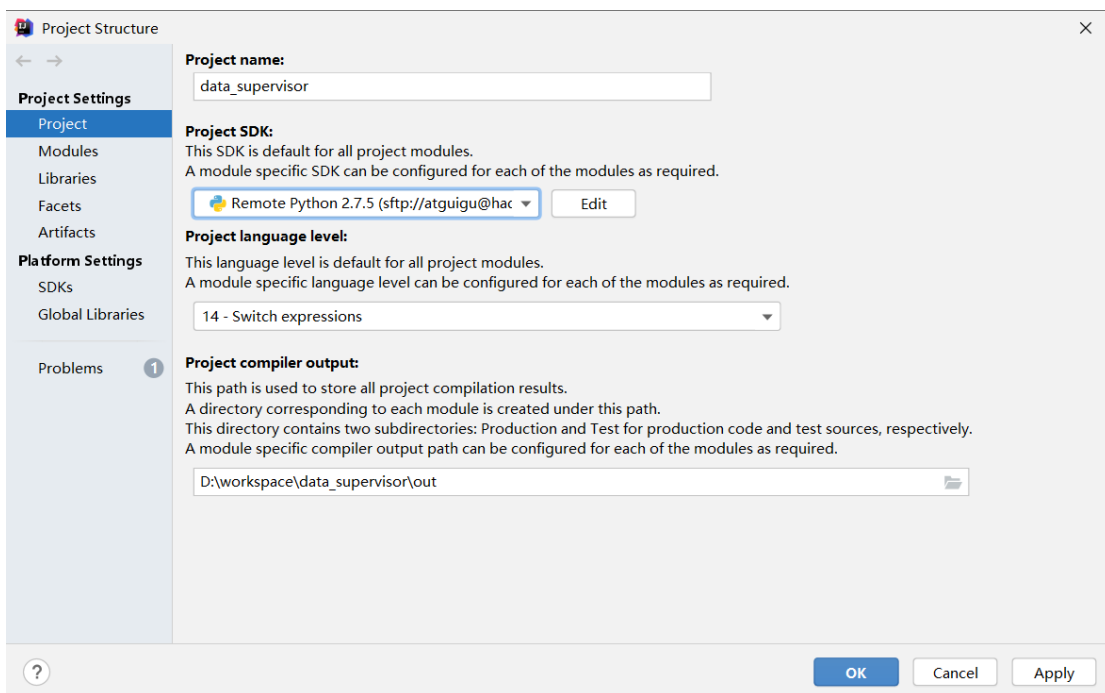
第四步：点击“+”，填入 ssh 连接信息，点击 Next



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',
```

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载，可百度访问：尚硅谷官网


```
`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 检查脚本、重复 id 检查脚本、值域检查脚本、数据量环比检查脚本和数据量同比检查脚本。

下面分别给大家介绍一下五类检测脚本的具体编写。

1. 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;
```

2.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
            ;;
    esac
done
```

```
# 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;"
```

3. 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
            ;;
    esac
done
```

```
?)
    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
```

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载, 可百度访问: [尚硅谷官网](#)

```
# -*- 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
```

更多 [Java](#) -[大数据](#) -[前端](#) -[python](#) 人工智能资料下载,可百度访问: [尚硅谷官网](#)

```
#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

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
```

(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 = "KADEM0ZWC0FWZET0"
```

```
# 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

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：

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 Azkaban 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,
```

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载, 可百度访问: [尚硅谷官网](#)


```

        "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.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)
```

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)
```

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)
```

5.Azkaban workflow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}

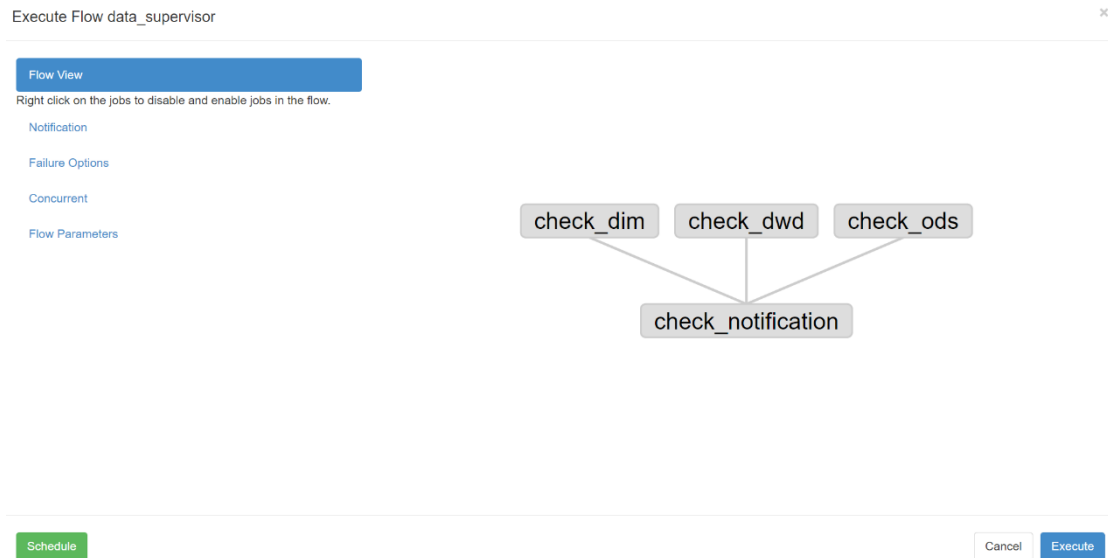
- name: check_dim
  type: command
  config:
    command: python check_dim.py ${dt}

- name: check_notification
  type: command
  dependsOn:
    - check_ods
    - check_dwd
    - check_dim
  config:
    command: python check_notification.py ${alert} ${dt}
```

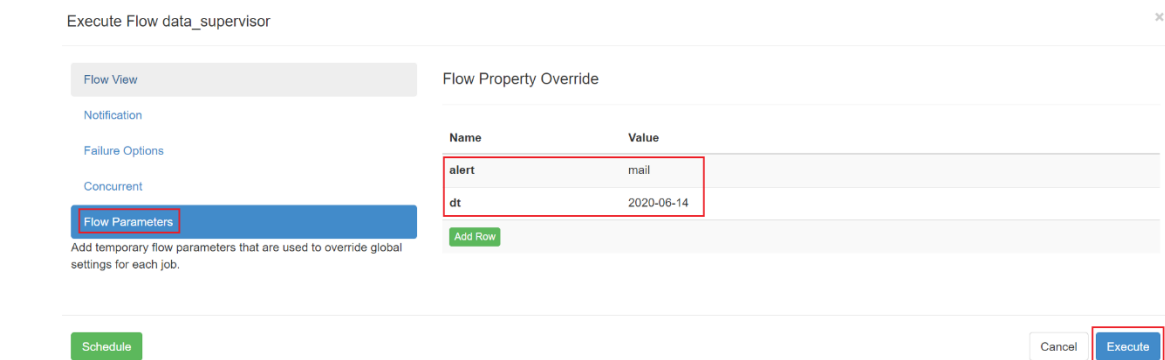
(3) 将所有文件打包成 `data_supervisor.zip` 文件

名称	修改日期	类型	大小
 azkaban.project	2021/5/8 13:25	PROJECT 文件	1 KB
 check_all.py	2021/5/8 16:40	PY 文件	5 KB
 check_dim.py	2021/5/8 14:35	PY 文件	1 KB
 check_dim.sh	2021/5/8 16:46	Shell Script	1 KB
 check_dwd.py	2021/5/8 14:33	PY 文件	1 KB
 check_dwd.sh	2021/5/8 16:46	Shell Script	1 KB
 check_notification.py	2021/5/8 17:00	PY 文件	4 KB
 check_ods.py	2021/5/8 14:10	PY 文件	1 KB
 check_ods.sh	2021/5/8 16:46	Shell Script	2 KB
 create.sql	2021/5/8 13:28	SQL 文件	4 KB
 data_supervisor.flow	2021/5/8 14:35	FLOW 文件	1 KB
 day_on_day.sh	2021/5/8 13:28	Shell Script	2 KB
 duplicate.sh	2021/5/8 13:29	Shell Script	2 KB
 null_id.sh	2021/5/8 13:29	Shell Script	1 KB
 range.sh	2021/5/8 16:48	Shell Script	2 KB
 std_dev.sh	2021/5/8 13:29	Shell Script	1 KB
 week_on_week.sh	2021/5/8 13:29	Shell Script	2 KB

(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



Execute Flow data_supervisor

Flow View

Notification

Failure Options

Concurrent

Flow Parameters

Add temporary flow parameters that are used to override global settings for each job.

Flow Property Override

Name	Value
alert	mail
dt	2020-06-14

Add Row

Schedule

Cancel Execute

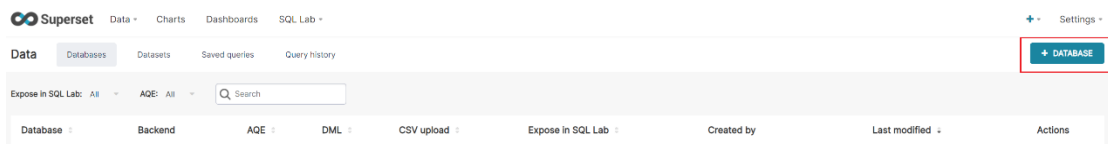
等待任务执行完毕，观察邮箱是否有告警邮件

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



Edit database

CONNECTION *

PERFORMANCE

SQL LAB SETTINGS

SECURITY

EXTRA

DATABASE NAME*

data_supervisor

SQLALCHEMY URI*

mysql://root:000000@hadoop102:3306/data_supervisor?charset=utf8

TEST CONNECTION

Refer to the [SQLAlchemy docs](#) for more information on how to structure your URI.

CANCEL

SAVE

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

Superset

Data
Charts
Dashboards
SQL Lab

Data

Database
Datasets
Saved queries
Query history

Expose in SQL Lab: All

Database
Backend

data_supervisor
mysql

Superset

Data
Charts
Dashboards
SQL Lab

Data

Database
Datasets
Saved queries
Query history

Owner: All
Database: All
Schema: All
Type: All
Search

Name
Type
Source
Schema
Modified
Modified by
Owners
Actions

BULK SELECT

+ DATASET

Add dataset

DATASOURCE

Database: mysql data_supervisor

SCHEMA

Schema: data_supervisor

SEE TABLE SCHEMA 6 IN DATA_SUPERVISOR

TABLE

day_on_day

CANCEL
ADD

五张表格全部添加后，如下图所示。

Superset Data Charts Dashboards SQL Lab

Settings

Data Databases Datasets Saved queries Query history BULK SELECT + DATASET

Owner: All Database: All Schema: All Type: All Search

Name	Type	Source	Schema	Modified	Modified by	Owners	Actions
week_on_week	Physical	data_supervisor	data_supervisor	now	atguigu beijing	AB	
rng	Physical	data_supervisor	data_supervisor	3 seconds ago	atguigu beijing	AB	
nuil_id	Physical	data_supervisor	data_supervisor	7 seconds ago	atguigu beijing	AB	
duplicate	Physical	data_supervisor	data_supervisor	10 seconds ago	atguigu beijing	AB	
day_on_day	Physical	data_supervisor	data_supervisor	18 seconds ago	atguigu beijing	AB	

(3) 新建一个 dashboard，并命名，如下图所示。

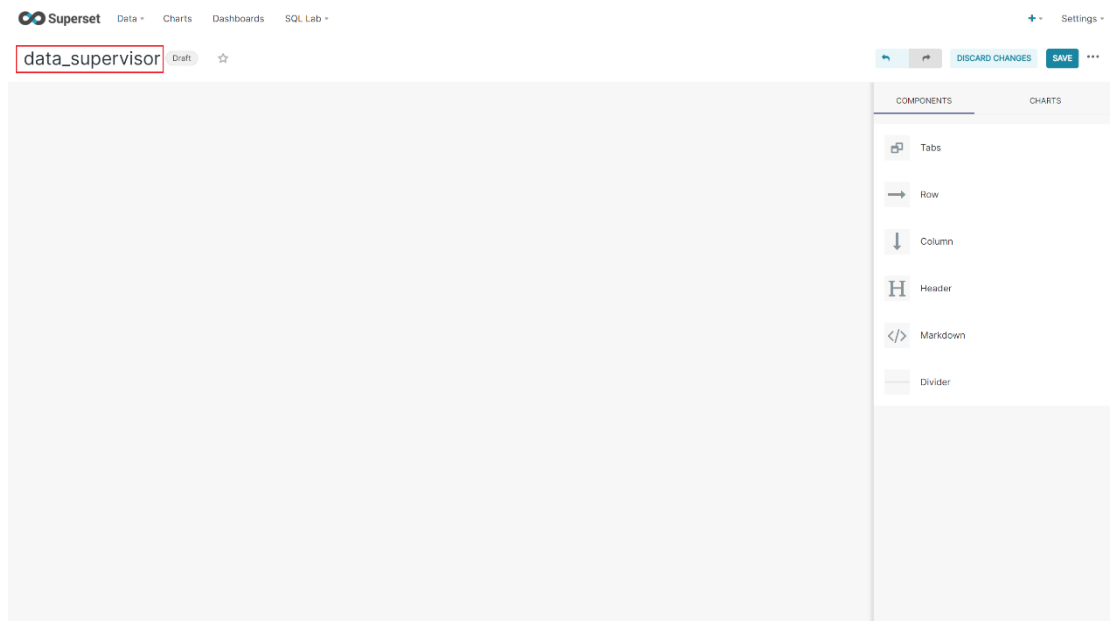
Superset Data Charts Dashboards SQL Lab

Settings

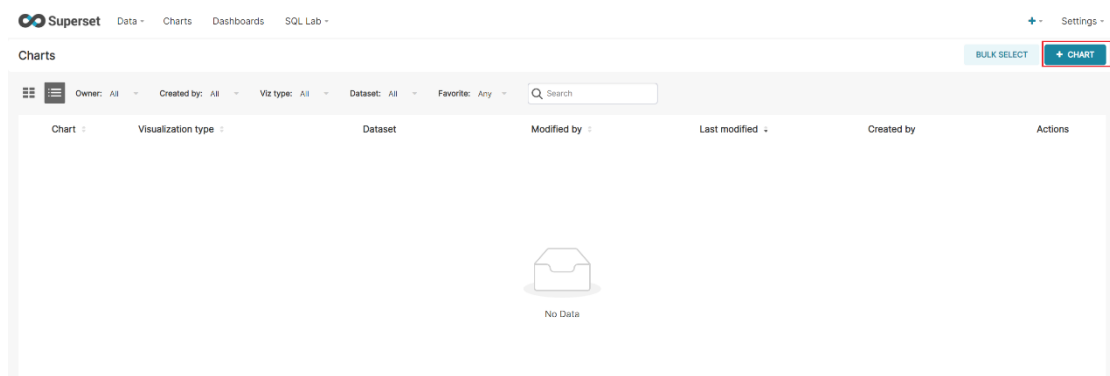
Dashboards BULK SELECT + DASHBOARD

Owner: All Created by: All Status: Any Favorite: Any Search

Title	Modified by	Status	Modified	Created by	Owners	Actions
-------	-------------	--------	----------	------------	--------	---------



(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。



Create a new chart

Choose a dataset

data_supervisor.day_on_day

If the dataset you are looking for is not available in the list, follow the instructions on how to add it in the Superset tutorial. [🔗](#)

Choose a visualization type

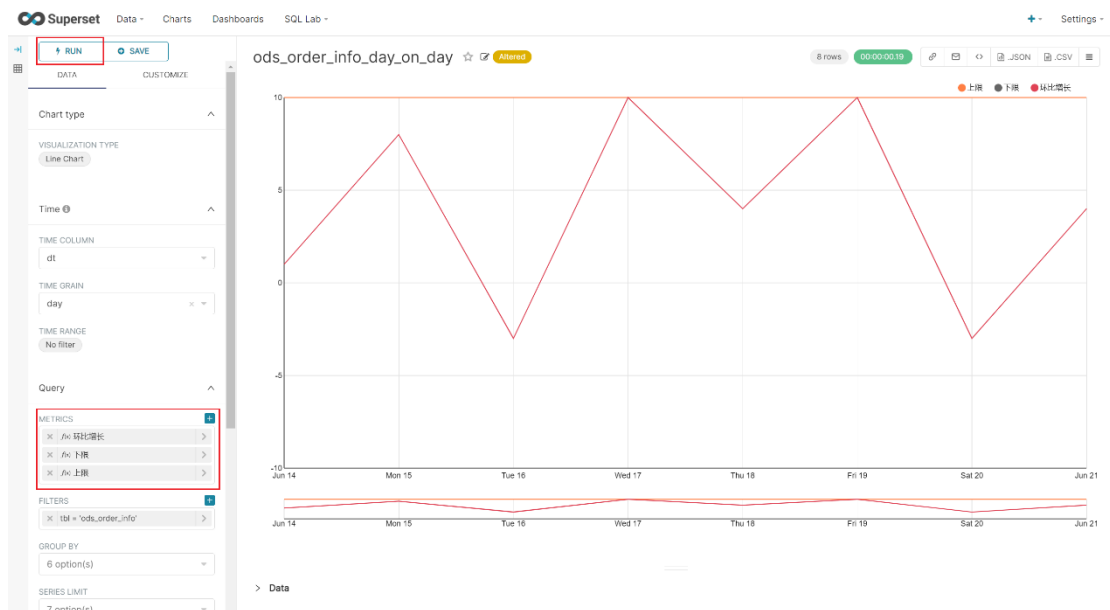
Line Chart

CREATE NEW CHART

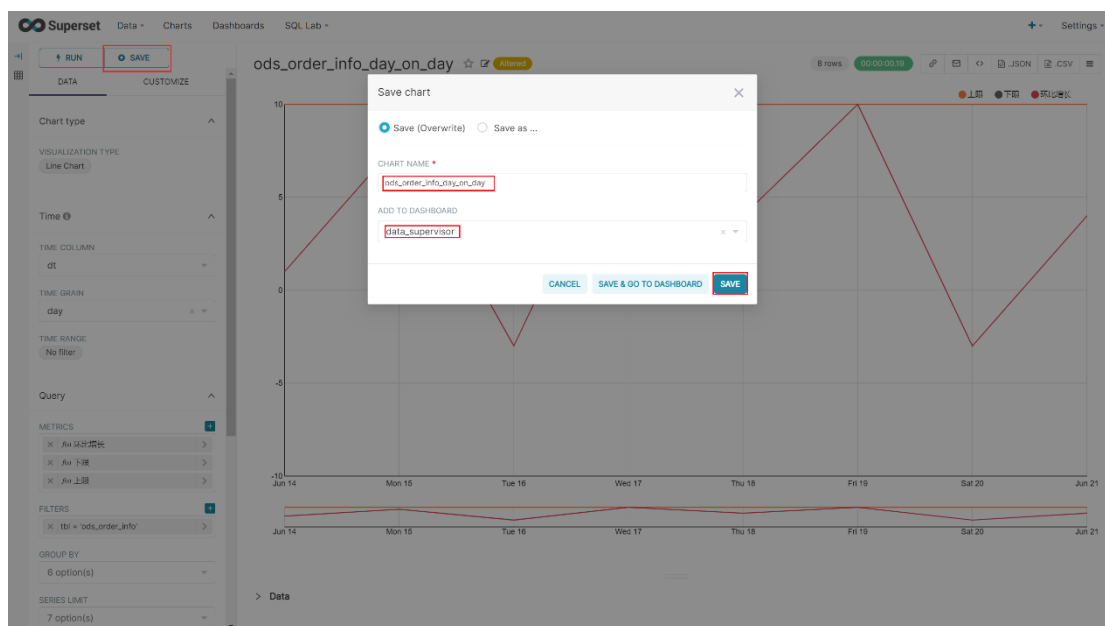
配置 chart 内容，如下图所示。

The image shows the Superset interface for configuring a chart. The chart is titled "ods_order_info_day_on_day". The visualization type is "Line Chart". The time column is "dt", time grain is "day", and time range is "No filter". The metrics section shows "环比增长" (环比增长) selected. The filters section shows "tbl = 'ods_order_info'" selected. The chart is currently showing "No Results".

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。

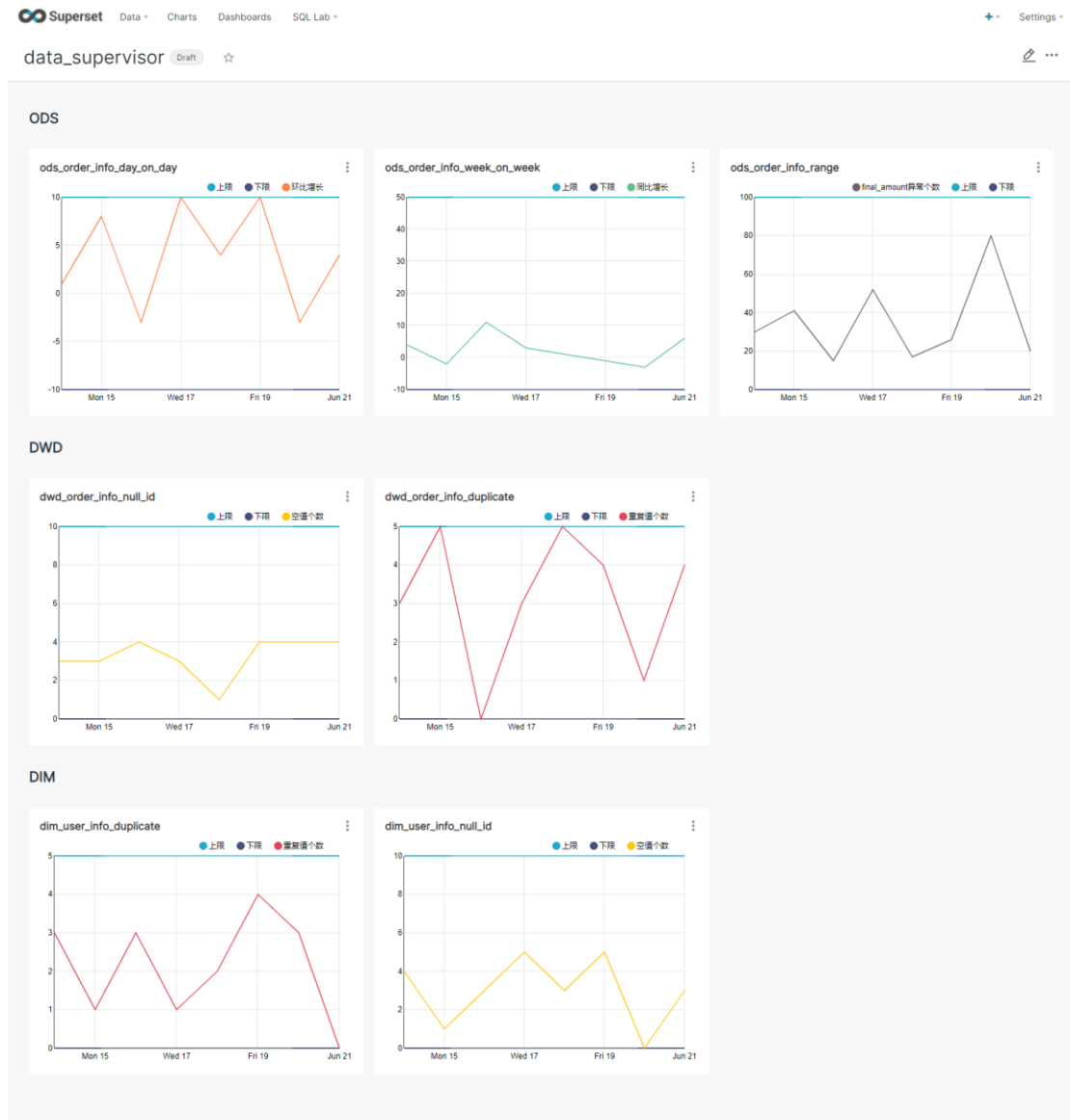


图 14-31 保存 chart 到 dashboard