# 斗鱼直播数据获取与展示

目录

[斗鱼直播数据获取与展示 1](#_Toc2268794)

[框架说明 2](#_Toc2268795)

[主要工程目录介绍 3](#_Toc2268796)

[各部分内容简述 4](#_Toc2268797)

[主控制程序 4](#_Toc2268798)

[创建oracle表 5](#_Toc2268799)

[执行爬虫程序 5](#_Toc2268800)

[数据插入oracle 6](#_Toc2268801)

[数据处理 6](#_Toc2268802)

[数据展示 7](#_Toc2268803)

[记录日志 8](#_Toc2268804)

[附件 9](#_Toc2268805)

[附件1：主控制程序代码 9](#_Toc2268806)

[附件2：创建oracle表 10](#_Toc2268807)

[附件3：爬虫程序 15](#_Toc2268808)

[附件4：插入数据到oracle 17](#_Toc2268809)

[附件5：Oracle数据处理脚本 18](#_Toc2268810)

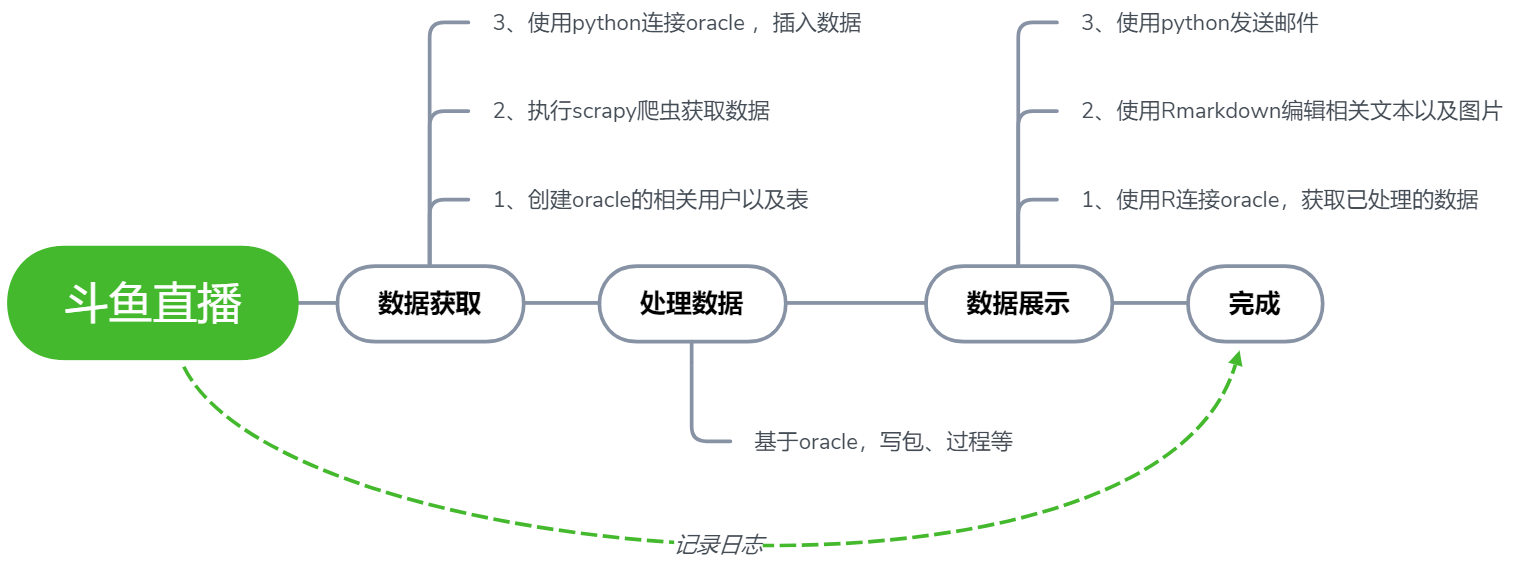
[附件6：数据展示 23](#_Toc2268811)

[附件7: 日志文件 26](#_Toc2268812)

## 框架说明

整个project分为数据获取、数据处理、数据展示三个部分，程序启动后的执行流程为：

每个部分主要的执行内容具体如下：



### 主要工程目录介绍

**./douyu**

**│**

**│ myEmail.py --用于发送email**

**│ mylogging.py --用户记录执行日志**

**│ startproject.py --主控制程序，用于执行各部分的模块**

**│**

**├─readme --说明文档**

**│ DOUYU\_README.docx --工程介绍文件**

**│**

**├─douyu --斗鱼scrapy爬虫程序**

**│ │ pipelines.py --用于连接并将数据插入oracle**

**│ │**

**│ ├─spiders**

**│ │ │ douyu\_spider.py --爬虫文件**

**│**

**├─html --用于生成以及保存html文件**

**│ douyu\_analysis.html --使用Rmarkdown**

**│ douyu\_analysis.Rmd --Rmarkdown的模板文件**

**│**

**├─log --日志文件**

**│ ├─debug --DEBUG级别的日志，定时清理**

**│ │ debug**

**│ │**

**│ └─error --error级别的日志，不清理**

**│ error**

**│**

**├─sql**

**│ COMMON.pck --package，存放通用的程序，如添加表分区**

**│ create\_table.bat --用于sqlplus执行create\_table.sql**

**│ create\_table.sql --用于oracle创建用户，创建所需的相关表**

**│ DOUYU\_ANALYSIS.pck --package，用于数据清洗，处理等**

**│ exec\_proc.bat --用于sqlplus执行exec\_proc.sql**

**│ exec\_proc.sql --用于创建执行oracle的相关处理过程**

**│**

**...**

## 各部分内容简述

### 主控制程序

目录：./douyu/startproject.py

用于启动执行各个程序，具体代码详见：附件1

1. 记录日志
2. 创建用户以及创建表
3. 执行爬虫，并将数据插入oracle
4. 对oracle中的数据进行处理
5. 数据可视化:R
6. 发送邮件



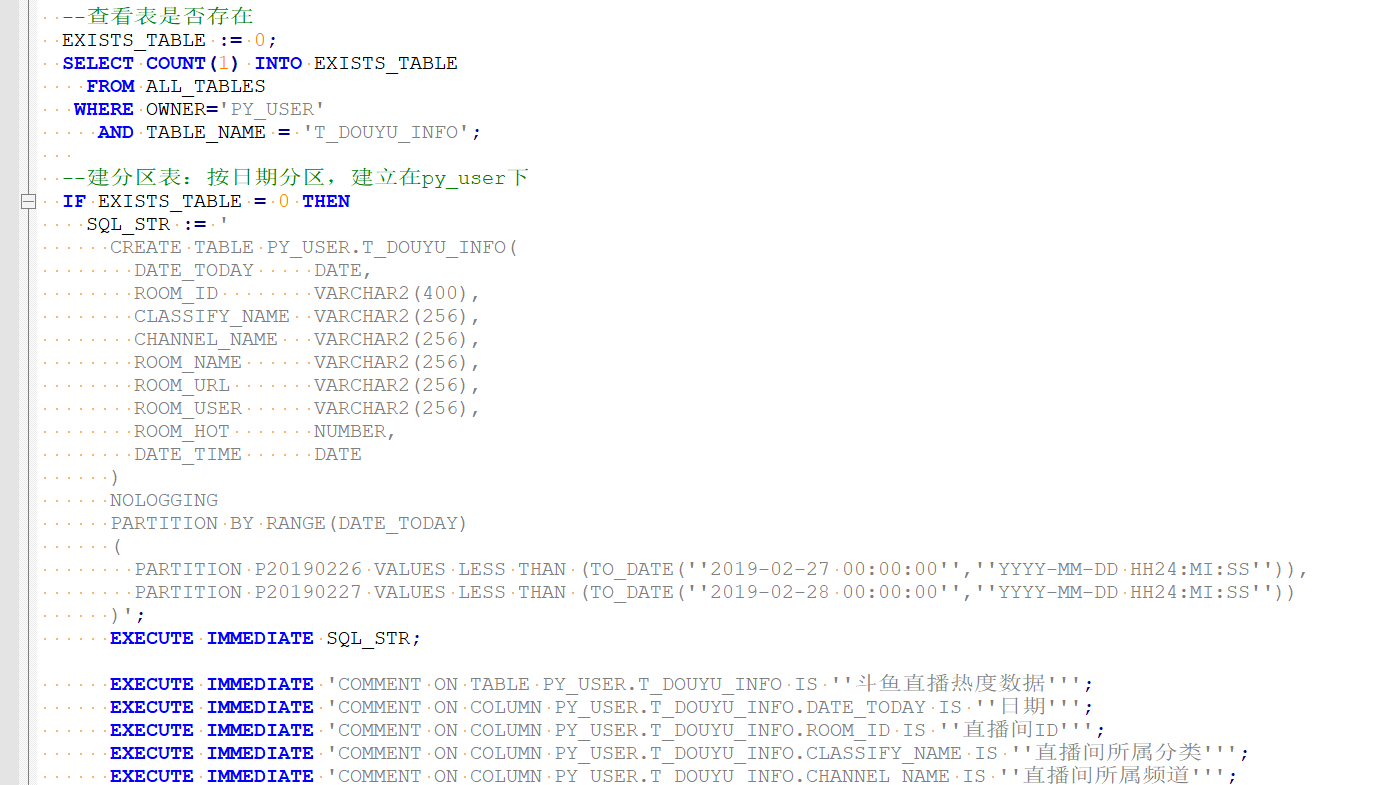
### 创建oracle表

目录：./douyu/sql/create\_table.\*

说明：点击或使用python执行批处理文件，批处理文件调用sqlplus，判断用户和表是否存在，不存在则创建。

其中数据表T\_DOUYU\_INFO根据实际需求，创建为分区表，按日期分区，粒度为1天

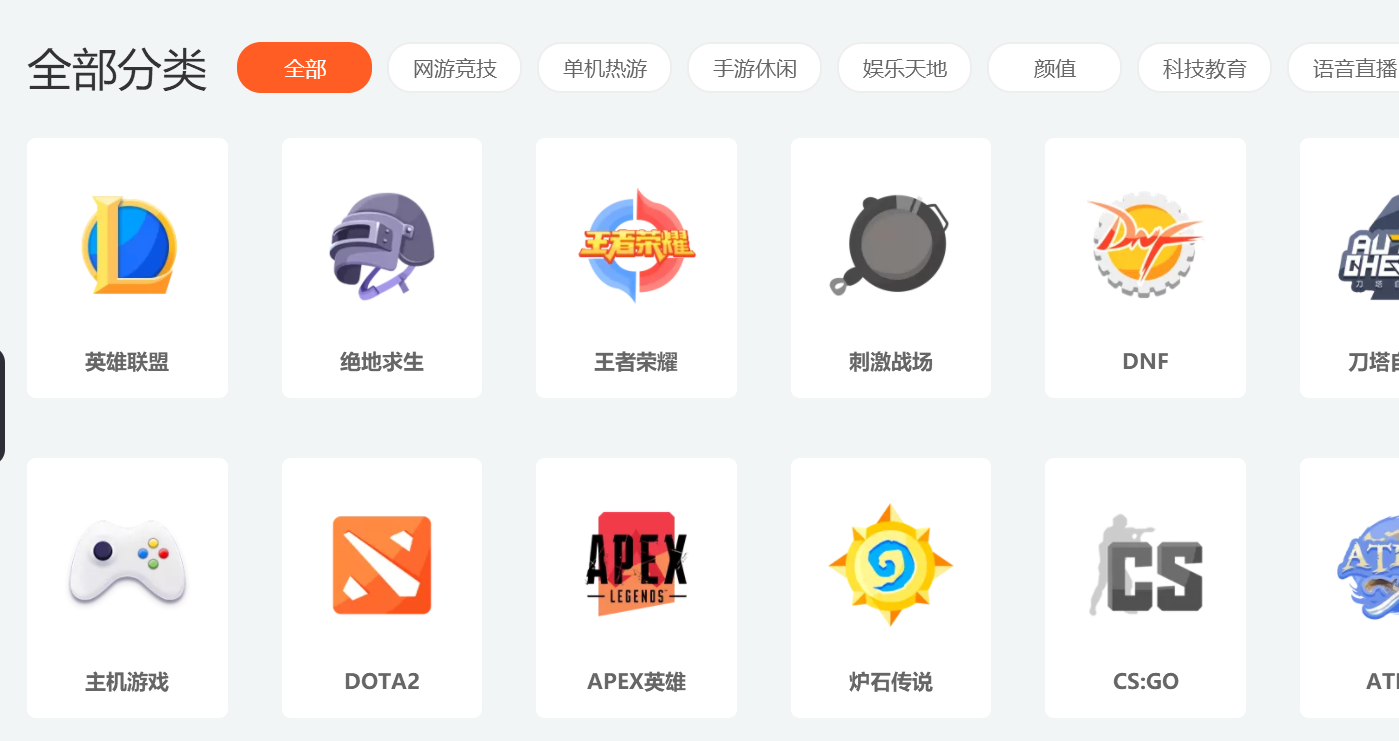
具体代码详见：附件2



### 执行爬虫程序

目录：./douyu/douyu/spider/douyu\_spider.py

说明：获取斗鱼所有分类下的直播间的信息，以及直播间相应的热度。详细代码见：附件3



### 数据插入oracle

目录：./douyu/douyu/pipelines.py

说明：直接用insert into进行插入数据，详见附件4

获取的数据如下：



### 数据处理

目录：./douyu/sql/\*.pck

说明：共建立2个包，COMMON以及DOUYU\_ANALYSIS，代码详见附件5

其中COMMON放置一些通用的脚本，目前含有过程ADD\_PARTITIONS，用于添加分区表的表分区。

DOUYU\_ANALYSIS用于直播数据的处理，目前含有4个过程：

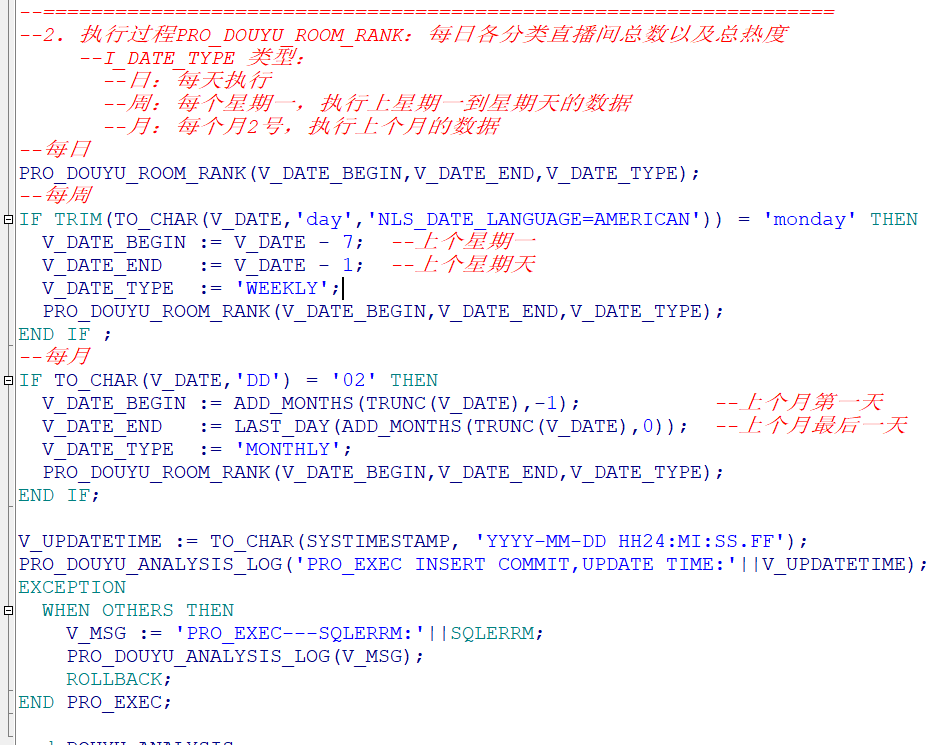
PRO\_DOUYU\_ANALYSIS\_LOG --记录DOUYU\_ANALYSIS的执行日志

PRO\_DOUYU\_CLASSIFY --获取每日各分类直播间总数以及总热度

PRO\_DOUYU\_ROOM\_RANK --获取每日各分类下主播的排行

PRO\_EXEC --调用执行以上过程，其中PRO\_DOUYU\_CLASSIFY需每日执行，

PRO\_DOUYU\_ROOM\_RANK每日、每周(周一)、每月(2号)都需执行



### 数据展示

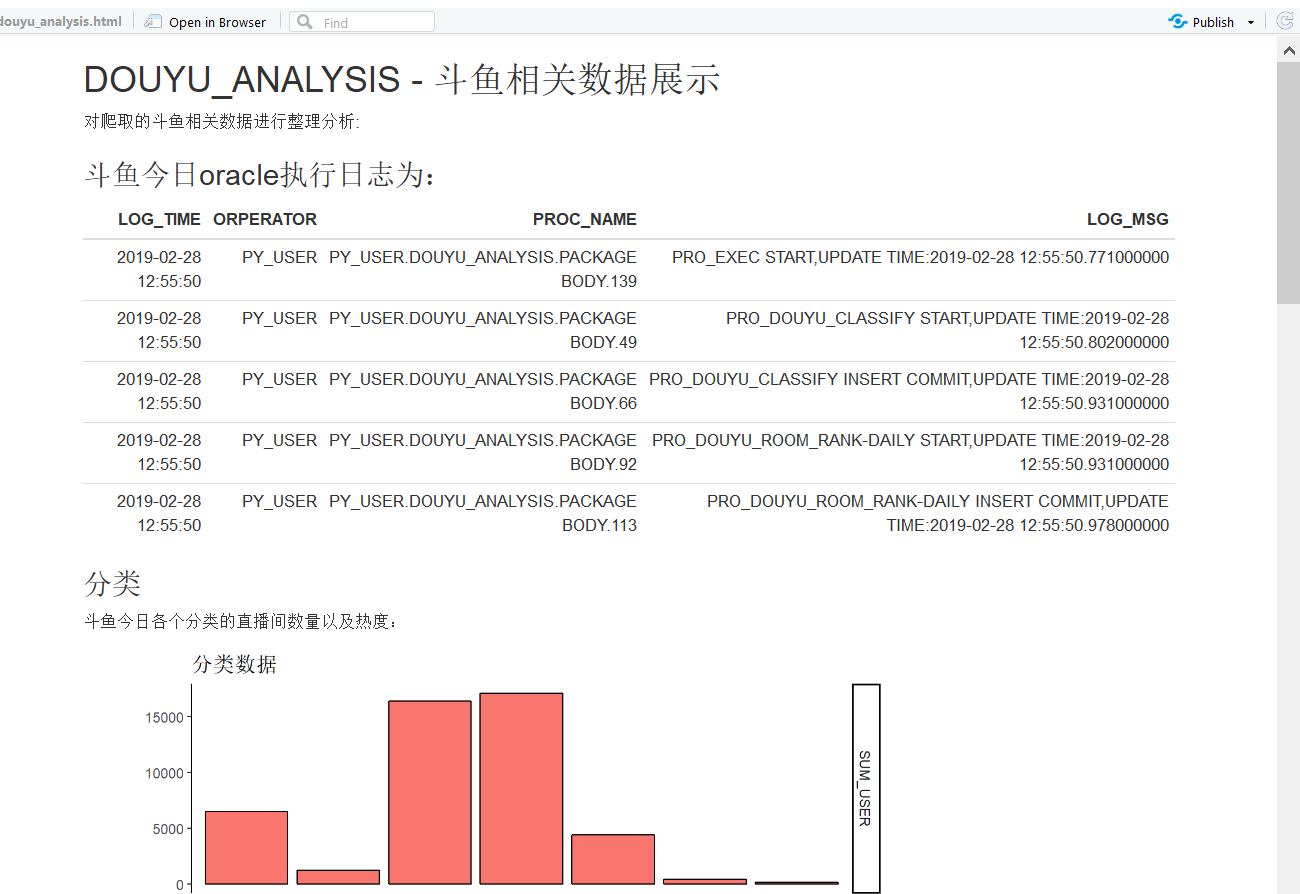
目录：./douyu/html/ \*

说明：基于Rmarkdown，可对数据进行可视化或输出结论，形成html之后，将其内容发送邮件。脚本详见附件6。

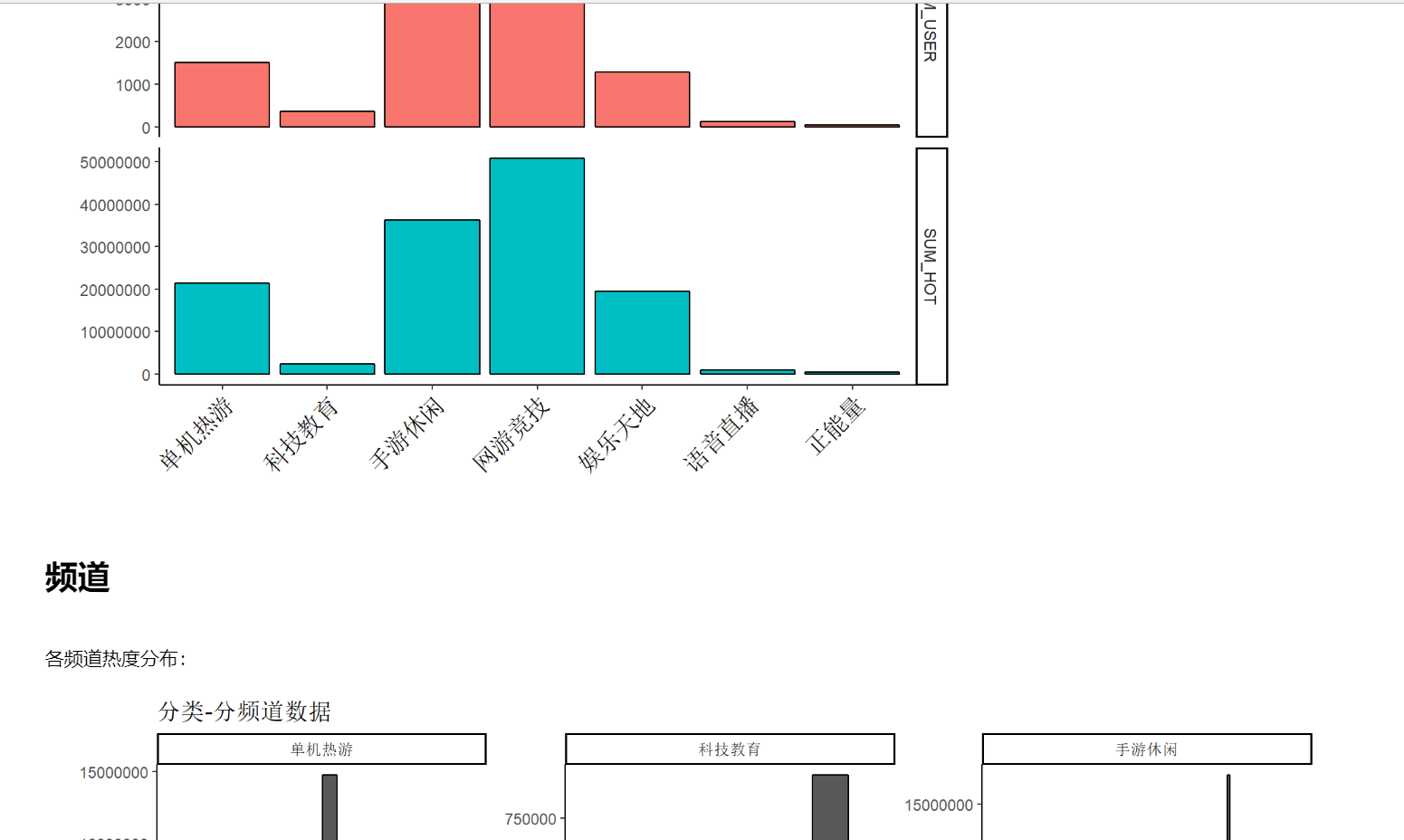
**Rmarkdown进行对应图表以及文字内容的编辑：**



**生成html:**

**发送邮件：**

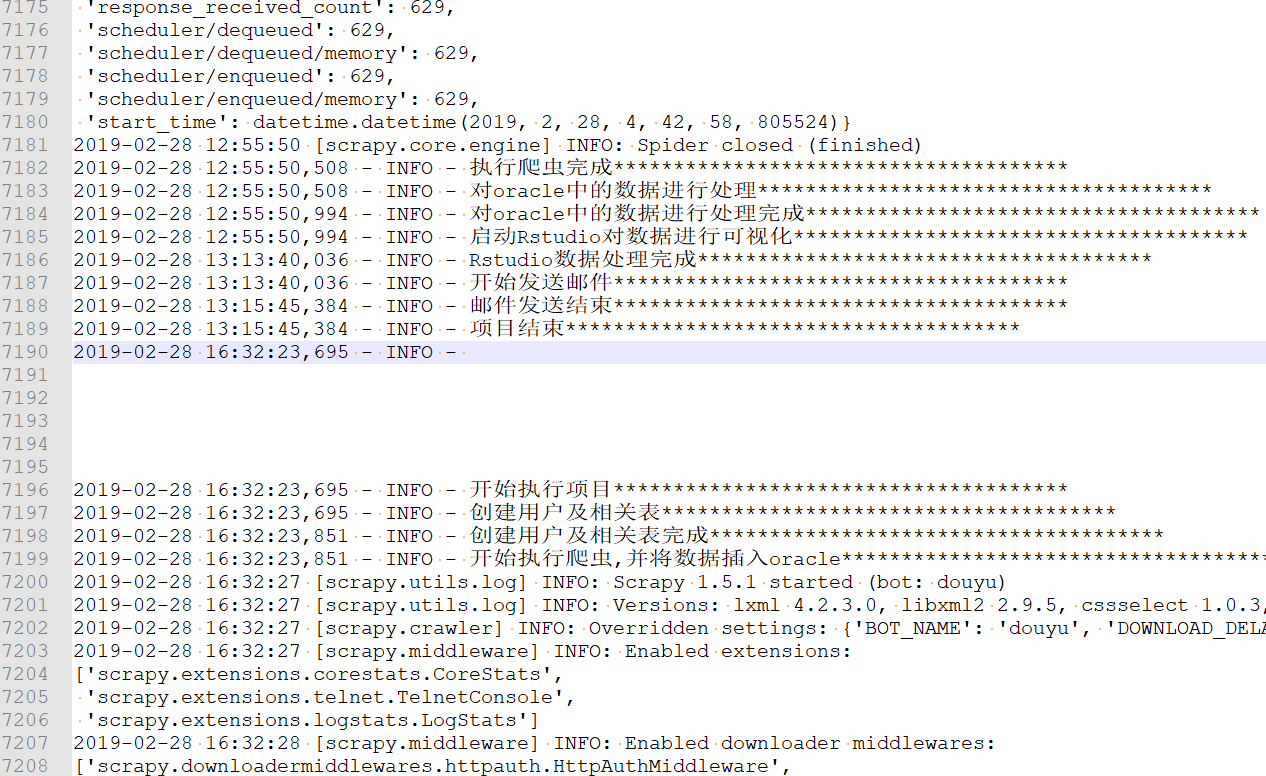
### 记录日志

目录：./douyu/mylogging.py

说明：分别记录debug和error这2种等级的日志，存放在/douyu/log文件夹中

其中，debug级别的日志，目前设置为记录总共记录365次日志，如果超过则删除最早的记录，

error级别的日志，不进行删除，具体设置详见附件



## 附件

### 附件1：主控制程序代码

|  |
| --- |
| # !/usr/bin/env python3  # \_\*\_ coding=utf-8 \_\*\_  # from scrapy.cmdline import execute  import subprocess  import os  from mylogging import MyLog  from myEmail import myEmail  # =======================================================================  # 记录日志(分别记录debug级别以及error级别的日志)  myLog = MyLog()  myLogger = myLog.get\_logger()  myLogger.info('\n'\*5) # 每次记录日之前插入5行，与之前的日志进行分割  myLogger.info("开始执行项目\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  # =======================================================================  # 获取当前路径  my\_path = os.path.abspath('.')  # 1. 创建用户以及创建表  myLogger.info("创建用户及相关表\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  file\_path = my\_path + '\\sql\\create\_table.bat'  command = file\_path + " " + my\_path + '\\sql'  subprocess.run(command, shell=True)  myLogger.info("创建用户及相关表完成\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  # =======================================================================  # 2. 执行爬虫，并将数据插入oracle  myLogger.info("开始执行爬虫,并将数据插入oracle\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  subprocess.run(r"scrapy crawl douyu\_spider", shell=True)  # execute("scrapy crawl douyu\_spider".split())  myLogger.info("执行爬虫完成\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  # =======================================================================  # 3. 对oracle中的数据进行处理  myLogger.info("对oracle中的数据进行处理\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  file\_path = my\_path + '\\sql\\exec\_proc.bat'  command = file\_path + " " + my\_path + '\\sql'  subprocess.run(command, shell=True)  myLogger.info("对oracle中的数据进行处理完成\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  # =======================================================================  # 4. 数据可视化:R  myLogger.info("启动Rstudio对数据进行可视化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  # 删除历史html文件  html\_path = my\_path + "\\html\\douyu\_analysis.html"  if os.path.exists(html\_path):  os.remove(html\_path)  # 启动Rstudio  rstudio\_path = r"D:\R\RStudio\bin\rstudio.exe"  rmd\_path = my\_path + "\\html\\\douyu\_analysis.Rmd"  command = rstudio\_path + " " + rmd\_path  subprocess.run(command, shell=True)  myLogger.info("Rstudio数据处理完成\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  # =======================================================================  # 5. 发送邮件  myLogger.info("开始发送邮件\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  # 读取html  html = open(html\_path, encoding='utf-8').read()  # 发送邮件  myEmail = myEmail()  myEmail.send\_email(html)  myLogger.info("邮件发送结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  # =======================================================================  # 6. 发送邮件  myLogger.info("项目结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*") |

### 附件2：创建oracle表

批处理文件：

|  |
| --- |
| @echo off  ::set path=D:\oracle\app\jr\product\11.2.0\dbhome\_1\bin;D:\instantclient\_11\_2;  ::set TNS\_ADMIN=D:\instantclient\_11\_2\NETWORK\ADMIN  ::set ORACLE\_HOME=D:\instantclient\_11\_2  set NLS\_LANG=SIMPLIFIED CHINESE\_CHINA.ZHS16GBK  ::set NLS\_LANG=american\_america.AL32UTF8  ::设置日期(yyyymmdd)  set y=%date:~0,4%  set m=%date:~5,2%  set d=%date:~8,2%  set rq=%y%%m%%d%  ::获取路径,如果没有传参,则取当前路径  set my\_path=%1  if "%my\_path%"=="" (  set my\_path=%cd%  )  ::执行sql  sqlplus system/\*\*\*\*\*\*@ORCL @%my\_path%\create\_table.sql %rq% |

Sql建表文件：

|  |
| --- |
| --屏幕输出  SET SERVEROUTPUT ON  --输出语句 for test  --SET ECHO ON  --获取日期  DEFINE P\_DATE=&1  DECLARE  --sql字符串  SQL\_STR VARCHAR2(2000);  --用户是否存在，默认为否(0)  EXISTS\_USER NUMBER := 0;  --表是否存在，默认为否(0)  EXISTS\_TABLE NUMBER := 0;  BEGIN  --===========================================================  --===========================================================  --查看用户是否存在  EXISTS\_USER := 0;  SELECT COUNT(1) INTO EXISTS\_USER  FROM ALL\_USERS  WHERE USERNAME = 'PY\_USER';    --用户不存在则创建并授权  IF EXISTS\_USER = 0 THEN  --创建用户  EXECUTE IMMEDIATE 'create user py\_user identified by ';  --授予用户创建SESSION的权限，即登陆权限，允许用户登录数据库  EXECUTE IMMEDIATE 'grant create session to py\_user';  --授予用户使用表空间的权限  EXECUTE IMMEDIATE 'grant unlimited tablespace to py\_user';  --授权:增删查改  EXECUTE IMMEDIATE 'grant create any table to py\_user';  EXECUTE IMMEDIATE 'grant drop any table to py\_user';  EXECUTE IMMEDIATE 'grant insert any table to py\_user';  EXECUTE IMMEDIATE 'grant update any table to py\_user';  --授权，包、函数、存储过程  EXECUTE IMMEDIATE 'grant create any procedure to py\_user';  EXECUTE IMMEDIATE 'grant execute any procedure to py\_user';  END IF;    --=================================================================  --=================================================================  --查看表是否存在  EXISTS\_TABLE := 0;  SELECT COUNT(1) INTO EXISTS\_TABLE  FROM ALL\_TABLES  WHERE OWNER='PY\_USER'  AND TABLE\_NAME = 'T\_DOUYU\_INFO';    --建分区表：按日期分区，建立在py\_user下  IF EXISTS\_TABLE = 0 THEN  SQL\_STR := '  CREATE TABLE PY\_USER.T\_DOUYU\_INFO(  DATE\_TODAY DATE,  ROOM\_ID VARCHAR2(400),  CLASSIFY\_NAME VARCHAR2(256),  CHANNEL\_NAME VARCHAR2(256),  ROOM\_NAME VARCHAR2(256),  ROOM\_URL VARCHAR2(256),  ROOM\_USER VARCHAR2(256),  ROOM\_HOT NUMBER,  DATE\_TIME DATE  )  NOLOGGING  PARTITION BY RANGE(DATE\_TODAY)  (  PARTITION P20190226 VALUES LESS THAN (TO\_DATE(''2019-02-27 00:00:00'',''YYYY-MM-DD HH24:MI:SS'')),  PARTITION P20190227 VALUES LESS THAN (TO\_DATE(''2019-02-28 00:00:00'',''YYYY-MM-DD HH24:MI:SS''))  )';  EXECUTE IMMEDIATE SQL\_STR;  EXECUTE IMMEDIATE 'COMMENT ON TABLE PY\_USER.T\_DOUYU\_INFO IS ''斗鱼直播热度数据''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_INFO.DATE\_TODAY IS ''日期''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_INFO.ROOM\_ID IS ''直播间ID''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_INFO.CLASSIFY\_NAME IS ''直播间所属分类''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_INFO.CHANNEL\_NAME IS ''直播间所属频道''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_INFO.ROOM\_NAME IS ''直播间名称''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_INFO.ROOM\_URL IS ''直播间url''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_INFO.ROOM\_USER IS ''直播间主播名称''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_INFO.ROOM\_HOT IS ''直播间热度''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_INFO.DATE\_TIME IS ''插入时间''';  END IF;    --创建表当前分区  PY\_USER.COMMON.ADD\_PARTITIONS('T\_DOUYU\_INFO',TO\_DATE('&P\_DATE','YYYYMMDD'));    --==========================================================================================  --==========================================================================================  --创建package-DOUYU\_ANALYSIS的日志表  EXISTS\_TABLE := 0;  SELECT COUNT(1) INTO EXISTS\_TABLE  FROM ALL\_TABLES  WHERE OWNER='PY\_USER'  AND TABLE\_NAME = 'T\_DOUYU\_ANALYSIS\_LOG';    IF EXISTS\_TABLE = 0 THEN  SQL\_STR := '  CREATE TABLE PY\_USER.T\_DOUYU\_ANALYSIS\_LOG(  LOG\_TIME TIMESTAMP,  ORPERATOR VARCHAR2(64),  PROC\_NAME VARCHAR2(64),  LOG\_MSG VARCHAR2(1000)  )';  EXECUTE IMMEDIATE SQL\_STR;    EXECUTE IMMEDIATE 'COMMENT ON TABLE PY\_USER.T\_DOUYU\_ANALYSIS\_LOG IS ''斗鱼相关数据DOUYU\_ANALYSIS日志表''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ANALYSIS\_LOG.LOG\_TIME IS ''日志记录时间''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ANALYSIS\_LOG.ORPERATOR IS ''执行者''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ANALYSIS\_LOG.PROC\_NAME IS ''存储过程名''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ANALYSIS\_LOG.LOG\_MSG IS ''日志信息''';  END IF;  --=========================================================  --=========================================================  --创建热度分类表  EXISTS\_TABLE := 0;  SELECT COUNT(1) INTO EXISTS\_TABLE  FROM ALL\_TABLES  WHERE OWNER='PY\_USER'  AND TABLE\_NAME = 'T\_DOUYU\_CLASSIFY';    IF EXISTS\_TABLE = 0 THEN  SQL\_STR := '  CREATE TABLE PY\_USER.T\_DOUYU\_CLASSIFY(  DATE\_TODAY DATE,  CLASSIFY\_NAME VARCHAR2(256),  CHANNEL\_NAME VARCHAR2(256),  SUM\_USER NUMBER,  SUM\_HOT NUMBER,  UP\_DATE DATE  )';  EXECUTE IMMEDIATE SQL\_STR;    EXECUTE IMMEDIATE 'COMMENT ON TABLE PY\_USER.T\_DOUYU\_CLASSIFY IS ''各分类直播间总数以及总热度''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_CLASSIFY.DATE\_TODAY IS ''日期''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_CLASSIFY.CLASSIFY\_NAME IS ''分类名称''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_CLASSIFY.CHANNEL\_NAME IS ''频道名称''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_CLASSIFY.SUM\_USER IS ''总主播数''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_CLASSIFY.SUM\_HOT IS ''总认读''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_CLASSIFY.UP\_DATE IS ''数据插入时间''';  END IF;    --=====================================================  --=====================================================  --各频道下主播排行表  EXISTS\_TABLE := 0;  SELECT COUNT(1) INTO EXISTS\_TABLE  FROM ALL\_TABLES  WHERE OWNER='PY\_USER'  AND TABLE\_NAME = 'T\_DOUYU\_ROOM\_RANK';    IF EXISTS\_TABLE = 0 THEN  SQL\_STR := '  CREATE TABLE PY\_USER.T\_DOUYU\_ROOM\_RANK(  DATE\_START DATE,  DATE\_END DATE,  DATE\_TYPE VARCHAR2(16),  CLASSIFY\_NAME VARCHAR2(256),  CLASS\_RANK NUMBER,  ROOM\_USER VARCHAR2(256),  SUM\_HOT NUMBER,  UP\_DATE DATE  )';  EXECUTE IMMEDIATE SQL\_STR;    EXECUTE IMMEDIATE 'COMMENT ON TABLE PY\_USER.T\_DOUYU\_ROOM\_RANK IS ''各频道下主播排行(前10)''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ROOM\_RANK.DATE\_START IS ''开始日期(闭区间)''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ROOM\_RANK.DATE\_END IS ''结束日期(闭区间)''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ROOM\_RANK.DATE\_TYPE IS ''日期类型(日/周/月)''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ROOM\_RANK.CLASSIFY\_NAME IS ''频道名称''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ROOM\_RANK.CLASS\_RANK IS ''直播间在该分类的热度排行''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ROOM\_RANK.ROOM\_USER IS ''主播名称''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ROOM\_RANK.SUM\_HOT IS ''总热度''';  EXECUTE IMMEDIATE 'COMMENT ON COLUMN PY\_USER.T\_DOUYU\_ROOM\_RANK.UP\_DATE IS ''数据插入时间''';  END IF;    END;  /  EXIT |

### 附件3：爬虫程序

|  |
| --- |
| # -\*- coding: utf-8 -\*-  import scrapy  import re  import json  from douyu.items import DouyuItem  """  描述：获取斗鱼各频道下的各个直播间的相关信息，包括  参数：无  修改：jr 2018-02-27  """  class DouyuSpiderSpider(scrapy.Spider):  name = 'douyu\_spider'  allowed\_domains = ['www.douyu.com']  # start\_urls = ['http://www.douyu.com/']  def start\_requests(self):  # 斗鱼直播分类（分类名称：url）  classify\_dict = {  '网游竞技': 'https://www.douyu.com/directory?shortName=PCgame',  '单机热游': 'https://www.douyu.com/directory?shortName=djry',  '手游休闲': 'https://www.douyu.com/directory?shortName=syxx',  '娱乐天地': 'https://www.douyu.com/directory?shortName=yl',  '科技教育': 'https://www.douyu.com/directory?shortName=kjjy',  '语音直播': 'https://www.douyu.com/directory?shortName=voice',  '正能量': 'https://www.douyu.com/directory?shortName=znl'  }  for key,value in classify\_dict.items():  yield scrapy.Request(url=value, meta={'classify\_name':key}, callback=self.parse)  def parse(self, response):  # 频道名称  channel\_list = response.xpath("//ul[@class='layout-Classify-list']/li[@class='layout-Classify-item']/a")  for each in channel\_list:  # 频道名称  channel\_name = each.xpath(".//strong/text()").extract\_first()  # 频道链接  channel\_url = "https://www.douyu.com%s" % each.xpath("./@href").extract\_first()  meta = {  # 分类名称  'classify\_name': response.meta['classify\_name'],  # 频道名称  'channel\_name': channel\_name  }  yield scrapy.Request(url=channel\_url, meta=meta, callback=self.get\_tag\_path)  # 获取直播间列表  def get\_tag\_path(self, response):  # 获取监本内容  data = response.xpath("//script[contains(text(),'tabTagPath')]/text()").extract\_first()  # 获取列表数据页数  page\_count = re.search('"pageCount":(.+?),"', data, flags=re.I).group(1)  # 获取列表数据请求接口地址  tag\_path = re.search('"tabTagPath":"(.+?)"', data, flags=re.I).group(1)  for page in range(1, int(page\_count)+1):  # 拼接请求接口链接，接口形式类似于https://www.douyu.com/gapi/rkc/directory/2\_1/5  tag\_url = "https://www.douyu.com%s" % tag\_path.replace("/c\_tag", "").replace("list", str(page))  meta = {  # 分类名称  'classify\_name': response.meta['classify\_name'],  # 频道名称  'channel\_name': response.meta['channel\_name']  }  yield scrapy.Request(url=tag\_url, meta=response.meta, callback=self.get\_room\_list)  def get\_room\_list(self, response):  item = DouyuItem()  # 获取当前分类的js数据  json\_data = json.loads(response.text)  # 获取直播间列表数据  room\_list = json\_data.get("data").get('rl')  for room\_info in room\_list:  # 直播间ID  room\_id = room\_info.get("rid")  # 直播间url  room\_url = "https://www.douyu.com%s" % room\_info.get("url")  # 直播间名称  room\_name = room\_info.get("rn")  # 主播名称  room\_user = room\_info.get("nn")  # 直播间热度  room\_hot = room\_info.get("ol")  # 分类名称  item['classify\_name'] = response.meta['classify\_name']  # 频道名称  item['channel\_name'] = response.meta['channel\_name']  # 直播间ID  item['room\_id'] = room\_id  # 直播间url  item['room\_url'] = room\_url  # 直播间名称  item['room\_name'] = room\_name  # 直播间主播  item['room\_user'] = room\_user  # 直播间热度  item['room\_hot'] = room\_hot  yield item |

### 附件4：插入数据到oracle

|  |
| --- |
| **class** OraclePipeline(object):  **def** open\_spider(self, spider):  *# 连接数据库* self.connect = cx\_Oracle.connect(**"py\_user/\*\*\*\*\*\*\*\*@127.0.0.1/ORCL"**)  *# 建立游标* self.cursor = self.connect.cursor()   **def** process\_item(self, item, spider):  *# 插入数据* sql = **"""  INSERT INTO T\_DOUYU\_INFO(DATE\_TODAY,ROOM\_ID,CLASSIFY\_NAME,  CHANNEL\_NAME,ROOM\_NAME,ROOM\_URL,ROOM\_USER,ROOM\_HOT,DATE\_TIME)  VALUES(TO\_DATE('{DATE\_TODAY}','YYYY-MM-DD'),'{ROOM\_ID}','{CLASSIFY\_NAME}',  '{CHANNEL\_NAME}','{ROOM\_NAME}','{ROOM\_URL}','{ROOM\_USER}','{ROOM\_HOT}',SYSDATE)  """**.format( DATE\_TODAY = datetime.date.today().strftime(**"%Y-%m-%d"**),  ROOM\_ID = item[**'room\_id'**],  CLASSIFY\_NAME = item[**'classify\_name'**],  CHANNEL\_NAME = item[**'channel\_name'**],  ROOM\_NAME = item[**'room\_name'**],  ROOM\_URL = item[**'room\_url'**],  ROOM\_USER = item[**'room\_user'**],  ROOM\_HOT = item[**'room\_hot'**])  self.cursor.execute(sql)  *# 提交* self.connect.commit()   **return** item   **def** close\_spider(self):  *# 关闭连接* self.cursor.close()  self.connect.close() |

### 附件5：Oracle数据处理脚本

**Package-COMMON:**

|  |
| --- |
| create or replace package body COMMON is  *---========================================================================*  *--添加分区： 添加表分区(日期类型),分区名称形如：Pyyyymmdd，如果该分区存在，则不添加。*  *-- 参数： P\_TABLE 表名（不需要带用户名）*  *-- P\_DATE 需添加的表分区日期*  *-- 日期： 2018-02-27*  *-- 修改： JR*  *-- 查询表： USER\_TAB\_PARTITIONS*  *-- 修改表： -*  *--===========================================================================*  PROCEDURE ADD\_PARTITIONS(P\_TABLE VARCHAR2,P\_DATE DATE) IS  V\_PARTITIONNAME VARCHAR2(120);  V\_ISEXIT NUMBER DEFAULT 0;  V\_SQL VARCHAR2(3000);  V\_PARTITION\_DATE VARCHAR2(20);  BEGIN  V\_PARTITIONNAME:='P'||TO\_CHAR(P\_DATE,'yyyymmdd');  V\_PARTITION\_DATE:=TO\_CHAR(P\_DATE+1,'yyyy-mm-dd');    *--检查表是否存在*  SELECT COUNT(1) INTO V\_ISEXIT FROM USER\_TAB\_PARTITIONS P  WHERE P.TABLE\_NAME=P\_TABLE  AND P.PARTITION\_NAME=V\_PARTITIONNAME;    IF V\_ISEXIT = 0 THEN  V\_SQL := 'alter table ' || P\_TABLE || ' add partition ' || V\_PARTITIONNAME ||  ' VALUES LESS THAN ( to\_date(''' || V\_PARTITION\_DATE || ''',''yyyy-mm-dd'') ) ';  EXECUTE IMMEDIATE V\_SQL;  *--DBMS\_OUTPUT.PUT\_LINE(V\_SQL);*  END IF;  END ADD\_PARTITIONS;  end COMMON; |

**Package-DOUYU\_ANALYSIS:**

|  |
| --- |
| create or replace package body DOUYU\_ANALYSIS is  *--更新日期*  V\_UPDATETIME VARCHAR2(64);  *--错误信息*  V\_MSG VARCHAR2(1000);  *---====================================================================*  *-- 说明：记录存储过程名、调用者信息、类型、以及相应的信息*  *-- 日期：2018-02-27*  *-- 修改：JR*  *--查询表：-*  *--修改表：T\_DOUYU\_ANALYSIS\_LOG*  *--=====================================================================*  PROCEDURE PRO\_DOUYU\_ANALYSIS\_LOG(I\_MSG VARCHAR2) IS  PRAGMA AUTONOMOUS\_TRANSACTION;  V\_PROC\_NAME VARCHAR2(128);  V\_OPERATOR VARCHAR2(64);  V\_OWNER VARCHAR2(64);  V\_NAME VARCHAR2(64);  V\_LINENO NUMBER;  V\_TYPE VARCHAR2(64);  BEGIN  OWA\_UTIL.WHO\_CALLED\_ME(V\_OWNER,V\_NAME,V\_LINENO,V\_TYPE);  V\_PROC\_NAME := V\_OWNER||'.'||V\_NAME||'.'||V\_TYPE||'.'||V\_LINENO;  V\_OPERATOR := SYS\_CONTEXT('USERENV','CURRENT\_USER') ;  INSERT INTO T\_DOUYU\_ANALYSIS\_LOG(LOG\_TIME,ORPERATOR,PROC\_NAME,LOG\_MSG)  VALUES(SYSTIMESTAMP,V\_OPERATOR,V\_PROC\_NAME,I\_MSG);  COMMIT;  END PRO\_DOUYU\_ANALYSIS\_LOG;  *---==================================================================*  *-- 过程：每日各分类直播间总数以及总热度*  *-- 参数：I\_DATE 数据日期*  *-- 日期：2018-02-27*  *-- 修改：JR*  *--查询表：T\_DOUYU\_INFO*  *--修改表：T\_DOUYU\_CLASSIFY*  *--====================================================================*  PROCEDURE PRO\_DOUYU\_CLASSIFY(I\_DATE DATE)  IS  V\_DATE DATE := I\_DATE; *--数据日期*  BEGIN  V\_UPDATETIME := TO\_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS.FF');  PRO\_DOUYU\_ANALYSIS\_LOG('PRO\_DOUYU\_CLASSIFY START,UPDATE TIME:'||V\_UPDATETIME);  INSERT INTO T\_DOUYU\_CLASSIFY NOLOGGING  SELECT TRUNC(V\_DATE) AS DATE\_TODAY,  NVL(T.CLASSIFY\_NAME,'-') AS CLASSIFY\_NAME,  NVL(T.CHANNEL\_NAME,'-') AS CHANNEL\_NAME,  COUNT(1) AS SUM\_USER,  SUM(NVL(T.ROOM\_HOT,0)) AS SUM\_HOT,  SYSDATE AS UP\_DATE  FROM T\_DOUYU\_INFO T  WHERE T.DATE\_TODAY = TRUNC(V\_DATE)  AND T.CHANNEL\_NAME NOT IN ('心悦大咖秀')  GROUP BY GROUPING SETS((),(T.CLASSIFY\_NAME),(T.CLASSIFY\_NAME,T.CHANNEL\_NAME))  ;  COMMIT;  V\_UPDATETIME := TO\_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS.FF');  PRO\_DOUYU\_ANALYSIS\_LOG('PRO\_DOUYU\_CLASSIFY INSERT COMMIT,UPDATE TIME:'||V\_UPDATETIME);  EXCEPTION  WHEN OTHERS THEN  V\_MSG := 'PRO\_DOUYU\_CLASSIFY---SQLERRM:'||SQLERRM;  PRO\_DOUYU\_ANALYSIS\_LOG(V\_MSG);  ROLLBACK;  END PRO\_DOUYU\_CLASSIFY;  *---====================================================================*  *-- 过程：每日各分类下主播的排行*  *-- 参数：I\_DATE\_START 数据开始日期*  *-- I\_DATE\_END 数据结束日期*  *-- 日期：2018-02-27*  *-- 修改：JR*  *--查询表：T\_DOUYU\_INFO*  *--修改表：T\_DOUYU\_ROOM\_RANK*  *--====================================================================*  PROCEDURE PRO\_DOUYU\_ROOM\_RANK(I\_DATE\_START DATE,I\_DATE\_END DATE,I\_DATE\_TYPE VARCHAR2)  IS  V\_DATE\_START DATE := I\_DATE\_START; *--数据开始日期*  V\_DATE\_END DATE := I\_DATE\_END; *--数据结束日期*  V\_DATE\_TYPE VARCHAR2(16) := I\_DATE\_TYPE; *--日期类型*  BEGIN  V\_UPDATETIME := TO\_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS.FF');  PRO\_DOUYU\_ANALYSIS\_LOG('PRO\_DOUYU\_ROOM\_RANK-'||V\_DATE\_TYPE||' START,UPDATE TIME:'||V\_UPDATETIME);  INSERT INTO T\_DOUYU\_ROOM\_RANK NOLOGGING  SELECT TRUNC(V\_DATE\_START),TRUNC(V\_DATE\_END),V\_DATE\_TYPE,  CLASSIFY\_NAME,CLASS\_RANK,ROOM\_USER,SUM\_HOT,SYSDATE AS UP\_DATE  FROM (  SELECT T.CLASSIFY\_NAME,  RANK()OVER(PARTITION BY T.CLASSIFY\_NAME  ORDER BY SUM(T.ROOM\_HOT) DESC) AS CLASS\_RANK,  T.ROOM\_USER,  SUM(T.ROOM\_HOT) AS SUM\_HOT  FROM T\_DOUYU\_INFO T  WHERE T.DATE\_TODAY >= TRUNC(V\_DATE\_START)  AND T.DATE\_TODAY <= TRUNC(V\_DATE\_END)  AND NVL(T.ROOM\_HOT,0) <> 0  GROUP BY T.CLASSIFY\_NAME,T.ROOM\_ID,T.ROOM\_USER  )TB  WHERE TB.CLASS\_RANK <= 10  ;  COMMIT;  V\_UPDATETIME := TO\_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS.FF');  PRO\_DOUYU\_ANALYSIS\_LOG('PRO\_DOUYU\_ROOM\_RANK-'||V\_DATE\_TYPE||' INSERT COMMIT,UPDATE TIME:'||V\_UPDATETIME);  EXCEPTION  WHEN OTHERS THEN  V\_MSG := 'PRO\_DOUYU\_ROOM\_RANK-'||V\_DATE\_TYPE||'---SQLERRM:'||SQLERRM;  PRO\_DOUYU\_ANALYSIS\_LOG(V\_MSG);  ROLLBACK;  END PRO\_DOUYU\_ROOM\_RANK;  *---========================================================================*  *-- 过程：执行过程*  *-- 参数：I\_DATE 数据日期*  *-- 日期：2018-02-27*  *-- 修改：JR*  *--查询表：-*  *--修改表：-*  *--=========================================================================*  PROCEDURE PRO\_EXEC(I\_DATE DATE)  IS  V\_DATE DATE := I\_DATE; *--数据日期*  V\_DATE\_BEGIN DATE := I\_DATE; *--数据开始日期*  V\_DATE\_END DATE := I\_DATE; *--数据结束日期*  V\_DATE\_TYPE VARCHAR2(16) := 'DAILY'; *--日期类型*  BEGIN  V\_UPDATETIME := TO\_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS.FF');  PRO\_DOUYU\_ANALYSIS\_LOG('PRO\_EXEC START,UPDATE TIME:'||V\_UPDATETIME);  *--========================================================*  *--1. 执行过程PRO\_DOUYU\_CLASSIFY：每日各分类直播间总数以及总热度*  PRO\_DOUYU\_CLASSIFY(V\_DATE);  *--========================================================*  *--2. 执行过程PRO\_DOUYU\_ROOM\_RANK：每日各分类直播间总数以及总热度*  *--I\_DATE\_TYPE 类型：*  *--日：每天执行*  *--周：每个星期一，执行上星期一到星期天的数据*  *--月：每个月2号，执行上个月的数据*  *--每日*  PRO\_DOUYU\_ROOM\_RANK(V\_DATE\_BEGIN,V\_DATE\_END,V\_DATE\_TYPE);  *--每周*  IF TRIM(TO\_CHAR(V\_DATE,'day','NLS\_DATE\_LANGUAGE=AMERICAN')) = 'monday' THEN  V\_DATE\_BEGIN := V\_DATE - 7; *--上个星期一*  V\_DATE\_END := V\_DATE - 1; *--上个星期天*  V\_DATE\_TYPE := 'WEEKLY';  PRO\_DOUYU\_ROOM\_RANK(V\_DATE\_BEGIN,V\_DATE\_END,V\_DATE\_TYPE);  END IF ;  *--每月*  IF TO\_CHAR(V\_DATE,'DD') = '02' THEN  V\_DATE\_BEGIN := ADD\_MONTHS(TRUNC(V\_DATE),-1); *--上个月第一天*  V\_DATE\_END := LAST\_DAY(ADD\_MONTHS(TRUNC(V\_DATE),0)); *--上个月最后一天*  V\_DATE\_TYPE := 'MONTHLY';  PRO\_DOUYU\_ROOM\_RANK(V\_DATE\_BEGIN,V\_DATE\_END,V\_DATE\_TYPE);  END IF;  V\_UPDATETIME := TO\_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS.FF');  PRO\_DOUYU\_ANALYSIS\_LOG('PRO\_EXEC INSERT COMMIT,UPDATE TIME:'||V\_UPDATETIME);  EXCEPTION  WHEN OTHERS THEN  V\_MSG := 'PRO\_EXEC---SQLERRM:'||SQLERRM;  PRO\_DOUYU\_ANALYSIS\_LOG(V\_MSG);  ROLLBACK;  END PRO\_EXEC;  end DOUYU\_ANALYSIS; |

### 附件6：数据展示

Rmarkdown模板：

|  |
| --- |
| ```{r setup, include=FALSE}  knitr::opts\_chunk$set(echo=FALSE, message=FALSE)  ```  ```{r echo=FALSE,warning=FALSE}  library(DBI)  library(ROracle)  library(reshape2)  library(ggplot2)  library(ggthemes)  library(scales)  library(formattable)  library(reshape2)  library(ggthemes)  library(ggplot2)  library(wordcloud2)  # ======================================================  # 连接oracle  drv <-dbDriver("Oracle")  connect.tns <-"(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)  (PORT=1521))(CONNECT\_DATA=(SERVER=DEDICATED)(SERVICE\_NAME=orcl)))"  conn <-dbConnect(drv,username="py\_user",password="\*\*\*\*\*\*",dbname = connect.tns)  ```  ## DOUYU\_ANALYSIS - 斗鱼相关数据展示  对爬取的斗鱼相关数据进行整理分析:    ### 斗鱼今日oracle执行日志为：    ```{r echo=FALSE, warning=FALSE}  # ======================================================  # 获取日志数据  sql<-"  SELECT TO\_CHAR(T.LOG\_TIME,'YYYY-MM-DD HH24:MI:SS') AS LOG\_TIME,  T.ORPERATOR,  T.PROC\_NAME,  T.LOG\_MSG  FROM T\_DOUYU\_ANALYSIS\_LOG T  WHERE TRUNC(T.LOG\_TIME) = TRUNC(SYSDATE)  --AND T.LOG\_MSG LIKE '%SQLERRM%'  AND ROWNUM<=5  ORDER BY T.LOG\_TIME ASC  "  #fetch data  sql\_send<-dbSendQuery(conn,sql)  data <- fetch(sql\_send,-1)  # 显示日志表格  formattable(data)  ```  ### 分类  斗鱼今日各个分类的直播间数量以及热度：  ```{r echo=FALSE, warning=FALSE}  # ======================================================  # 分类条形图  sql="  SELECT T.CLASSIFY\_NAME,T.SUM\_USER,T.SUM\_HOT  FROM T\_DOUYU\_CLASSIFY T  WHERE TRUNC(T.DATE\_TODAY) = TRUNC(SYSDATE)  AND T.CLASSIFY\_NAME <> '-'  AND T.CHANNEL\_NAME = '-'  "  #fetch data  sql\_send<-dbSendQuery(conn,sql)  data <- fetch(sql\_send,-1)  # 数据变形：变为长数据  data\_melt <- melt(data, id.vars = "CLASSIFY\_NAME", measure.vars = c("SUM\_USER","SUM\_HOT") , variable.name = "condition", value.name = "values")  # 做条形图  options(scipen=200)  p<- ggplot(data\_melt, aes(x=CLASSIFY\_NAME, y = values , fill=condition )) + xlab("") + ylab("")+  geom\_histogram( stat='identity',colour = 'black',position=position\_dodge())  # p <- p+theme(axis.text.y=element\_text(colour="black",size=13))+scale\_fill\_manual(values=c("grey","black"))  # p <- p+theme(panel.background=element\_blank(),panel.grid.minor=element\_blank(),  axis.line=element\_line(size=0.5),legend.title=element\_blank())  p <- p+theme\_classic()  p<-p+theme(legend.title=element\_blank())+ scale\_fill\_discrete(guide = FALSE) ##把所有图例的标题去掉  p <- p+theme(axis.text.x=element\_text(angle=45,colour="black",size=13,hjust=1))  p <- p+facet\_grid(condition~.,scales = "free")  p+ggtitle("分类数据")  ```  ### 频道  各频道热度分布：  ```{r echo=FALSE, fig.height=10, fig.width=10, warning=FALSE}  # ======================================================  # 各频道热度条形图  sql="  SELECT T.CLASSIFY\_NAME,T.CHANNEL\_NAME,T.SUM\_HOT  FROM T\_DOUYU\_CLASSIFY T  WHERE TRUNC(T.DATE\_TODAY) = TRUNC(SYSDATE)  AND T.CHANNEL\_NAME <> '-'  "  #fetch data  sql\_send<-dbSendQuery(conn,sql)  data <- fetch(sql\_send,-1)  # 做条形图  options(scipen=200)  p<- ggplot(data, aes(x=CHANNEL\_NAME, y = SUM\_HOT)) + xlab("") + ylab("")+  geom\_histogram( stat='identity',colour = 'black',position=position\_dodge())  # p <- p+theme(axis.text.y=element\_text(colour="black",size=13))+scale\_fill\_manual(values=c("grey","black"))  # p <- p+theme(panel.background=element\_blank(),panel.grid.minor=element\_blank(),  axis.line=element\_line(size=0.5),legend.title=element\_blank())  p <- p+theme\_classic()  p<-p+theme(legend.title=element\_blank())+ scale\_fill\_discrete(guide = FALSE) ##把所有图例的标题去掉  p <- p+theme(axis.text.x=element\_text(angle=45,colour="black",size=13,hjust=1))  p <- p+facet\_wrap(~CLASSIFY\_NAME,scales = "free")  p+ggtitle("分类-分频道数据")  ```  ### 主播  各类型主播热度排行:    ```{r echo=FALSE,warning=FALSE}  # ======================================================  # 各类型主播热度排行  sql="  SELECT T.CLASSIFY\_NAME,T.CLASS\_RANK,T.ROOM\_USER,T.SUM\_HOT  FROM T\_DOUYU\_ROOM\_RANK T  WHERE TRUNC(T.DATE\_START) = TRUNC(SYSDATE)  AND T.DATE\_TYPE = 'DAILY'  AND T.CLASS\_RANK <= 3  "  #fetch data  sql\_send<-dbSendQuery(conn,sql)  data <- fetch(sql\_send,-1)  # 显示排行榜  formattable(data)  ```  各主播的热度对比：  ```{r echo=FALSE,warning=FALSE}  # ======================================================  # 云词：各主播的热度对比  data\_hot\_frq = data[c('ROOM\_USER','SUM\_HOT')]  wordcloud2(data\_hot\_frq, size = 2, rotateRatio=1)  # ======================================================  #关闭连接  dbDisconnect(conn)  ``` |

### 附件7: 日志文件

|  |
| --- |
| class MyLog:  # 设置日志文件的路径，默认为当前路径：  # ./log/debug/debug 最新的日志文件为debug，历史的日志文件为debug%Y-%m-%d.log  # ./log/error/error  def \_\_init\_\_(self, debug\_file\_name='debug', error\_file\_name='error',  log\_path=None):  self.log\_path = log\_path  # 默认路径为当前路径  self.debug\_file\_path = './log/debug'  self.error\_file\_path = './log/error'  # 创建文件目录  self.create\_path()  # 设置debug级别以上的日志的文件路径：debug\_file\_path  self.debug\_file = self.debug\_file\_path + '/' + debug\_file\_name  # 设置error级别以上的日志的文件路径：error\_file\_name  self.error\_file = self.error\_file\_path + '/' + error\_file\_name  def create\_path(self):  # 创建日志文件路径  # 如果路径为空(默认值),则使用当前路径,否则就使用输入的路径  if self.log\_path:  self.debug\_file\_path = str(self.log\_path) + '/log/debug'  self.error\_file\_path = str(self.log\_path) + '/log/error'  # 判断日志路径是否存在，如果不存在则创建对应的目录  if not os.path.exists(self.debug\_file\_path):  os.makedirs(self.debug\_file\_path)  if not os.path.exists(self.error\_file\_path):  os.makedirs(self.error\_file\_path)  def get\_logger(self):  # 获取logger实例，如果参数为空，则返回root logger  logger = logging.getLogger('MyLogger')  # 设置logger等级 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  logger.setLevel(logging.DEBUG)  # 设置handler debug  # when-日志以天为单位,过了0点就进行分割  # interval-when个单位的日志记录写在同一日志文件，设为为同一天的日志写在同一日志文件  # backupCount-保留365个日志文件  debug\_log\_handler=logging.handlers.TimedRotatingFileHandler(  filename=self.debug\_file, when='MIDNIGHT', interval=1, backupCount=365)  # 设置日志文件名后缀  debug\_log\_handler.suffix = "%Y-%m-%d.log"  # debug\_log\_handler.extMatch = re.compile(r"^\d{4}-\d{2}-\d{2}")  # 设置日期格式  debug\_log\_handler.setFormatter(logging.Formatter("%(asctime)s - %(levelname)s - %(message)s"))  # 添加handler  logger.addHandler(debug\_log\_handler)  # 设置handler error  error\_log\_handler = logging.FileHandler(self.error\_file)  error\_log\_handler.setLevel(logging.ERROR)  error\_log\_handler.setFormatter(  logging.Formatter("%(asctime)s - %(levelname)s - %(filename)s[:%(lineno)d] - %(message)s"))  # 添加handler  logger.addHandler(error\_log\_handler)  return logger |