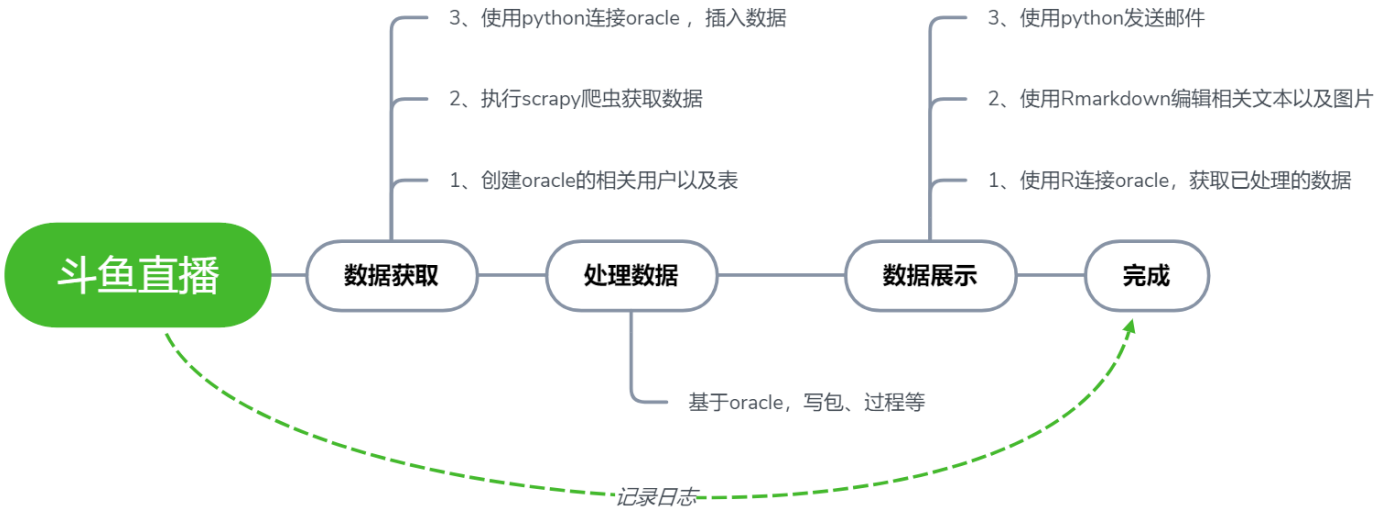
斗鱼直播数据获取与展示

目录

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

```
# ===================================================================
# 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**************************************")
```

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

```
--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';

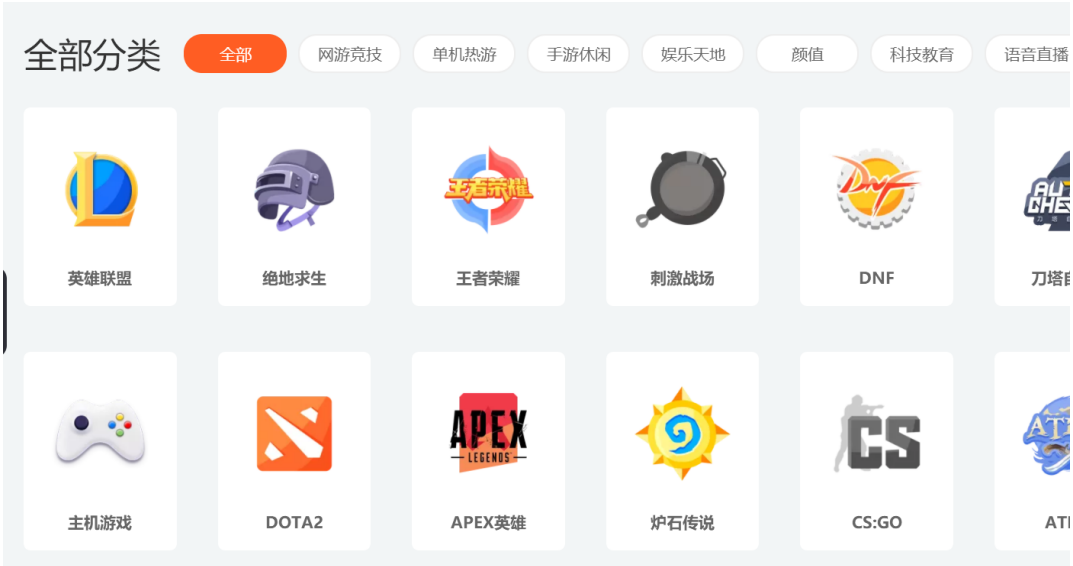
--建分区表：按日期分区，建立在py_user下
IF EXISTS_TABLE = 0 THEN
  SQL_STR := '
    CREATE TABLE PY_USER.T_DOUYU_INFO(
      DATE_TODAY    DATE,
      ROOM_ID       VARCHAR2(400),
      CLASSIFY_NAME VARCHAR2(256),
      CHANNEL_NAME  VARCHAR2(256),
      ROOM_NAME     VARCHAR2(256),
      ROOM_URL      VARCHAR2(256),
      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''';
```

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：

	DATE_TOD	ROOM_ID	CLASSIFY	CHANNEL_	ROOM_NAME	ROOM_USER	ROOM_	DATE_TIME
1	2019/2/28	21760	网游竞技	炉石传说	老年人棋牌室	僧僧僧	2380	2019/2/28 12:43:35
2	2019/2/28	6494612	网游竞技	炉石传说	你负责教，我负责打，	1007892721	2378	2019/2/28 12:43:35
3	2019/2/28	3904940	网游竞技	炉石传说	老佛爷的直播间	屁鼓	2369	2019/2/28 12:43:35
4	2019/2/28	6439565	网游竞技	炉石传说	待业君：红尘三杯酒，	待业君	2321	2019/2/28 12:43:35
5	2019/2/28	5093705	网游竞技	炉石传说	jjc神之仰望	dame0o0ws	2084	2019/2/28 12:43:35
6	2019/2/28	4735612	网游竞技	炉石传说	只玩骑士的菜鸡主播	是小M啊	1942	2019/2/28 12:43:35
7	2019/2/28	2976799	网游竞技	炉石传说	【狂野】下会棋	戰鬥嘿咻6	1884	2019/2/28 12:43:35
8	2019/2/28	1475496	网游竞技	炉石传说	不叠四甲的战士不是好	GD小老头	1273	2019/2/28 12:43:35
9	2019/2/28	6470182	网游竞技	炉石传说	我为炉石献青春	文艺复兴之路上[	1100	2019/2/28 12:43:35
10	2019/2/28	1947711	网游竞技	炉石传说	我是谁，我在哪儿，我	水晶心的眼泪	1086	2019/2/28 12:43:35
11	2019/2/28	6451430	网游竞技	炉石传说	哇呜哇呜哇哇	撸管叔叔爱萝莉	1006	2019/2/28 12:43:35
12	2019/2/28	6114048	网游竞技	炉石传说	随便玩一下。 611404	Mystill	932	2019/2/28 12:43:35
13	2019/2/28	2605084	网游竞技	炉石传说	月末2-传说。。。	zp是我呀	891	2019/2/28 12:43:35

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
```

```
--=================================================================
--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;
```

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：

```
### 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("SU
 variable.name = "condition", value.name = "values")

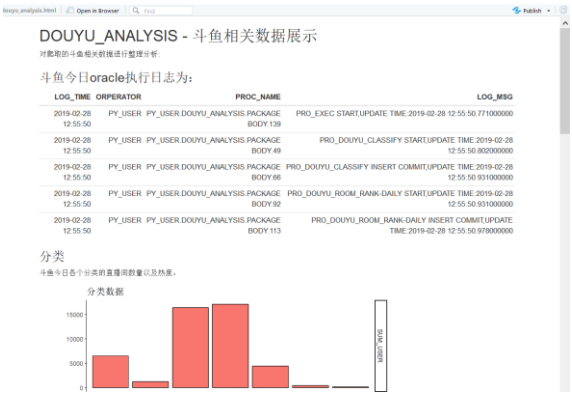
 # 做条形图
 options(scipen=200)
 p<- ggplot(data_melt, aes(x=CLASSIFY_NAME, y = values , fill=condition ))
     geom_histogram( stat='identity',colour = 'black',position=position_do
 # p <- p+theme(axis.text.y=element_text(colour="black",size=13))+scale_fi
 "))
 # p <- p+theme(panel.background=element_blank(),panel.grid.minor=element_
 ize=0.5),legend.title=element_blank())
 p <- p+theme_classic()
 p<-p+theme(legend.title=element_blank())+ scale_fill_discrete(guide = FAL
 p <- p+theme(axis.text.x=element_text(angle=45,colour="black",size=13,hju
 p <- p+facet_grid(condition~.,scales = "free")
 p+ggtitle("分类数据")
 ```

频道

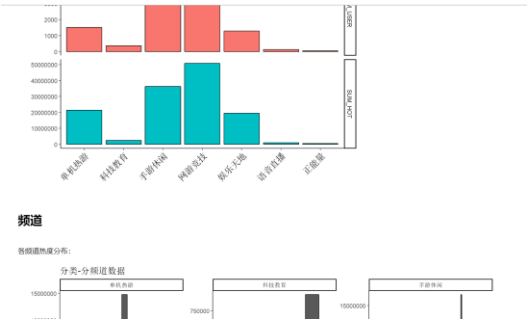
 各频道热度分布：

```{r echo=FALSE, fig.height=10, fig.width=10, warning=FALSE}
```

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

```
'response_received_count': 629,
'scheduler/dequeued': 629,
'scheduler/dequeued/memory': 629,
'scheduler/enqueued': 629,
'scheduler/enqueued/memory': 629,
'start_time': datetime.datetime(2019, 2, 28, 4, 42, 58, 805524)}
2019-02-28 12:55:50 [scrapy.core.engine] INFO: Spider closed (finished)
2019-02-28 12:55:50,508 - - INFO - - 执行爬虫完成**************************************
2019-02-28 12:55:50,508 - - INFO - - 对oracle中的数据进行处理**************************************
2019-02-28 12:55:50,994 - - INFO - - 对oracle中的数据进行处理完成**************************************
2019-02-28 12:55:50,994 - - INFO - - 启动Rstudio对数据进行可视化**************************************
2019-02-28 13:13:40,036 - - INFO - - Rstudio数据处理完成**************************************
2019-02-28 13:13:40,036 - - INFO - - 开始发送邮件**************************************
2019-02-28 13:15:45,384 - - INFO - - 邮件发送结束**************************************
2019-02-28 13:15:45,384 - - INFO - - 项目结束**************************************
2019-02-28 16:32:23,695 - - INFO - -



2019-02-28 16:32:23,695 - - INFO - - 开始执行项目**************************************
2019-02-28 16:32:23,695 - - INFO - - 创建用户及相关表**************************************
2019-02-28 16:32:23,851 - - INFO - - 创建用户及相关表完成**************************************
2019-02-28 16:32:23,851 - - INFO - - 开始执行爬虫,并将数据插入oracle**************************************
2019-02-28 16:32:27 [scrapy.utils.log] INFO: Scrapy 1.5.1 started (bot: douyu)
2019-02-28 16:32:27 [scrapy.utils.log] INFO: Versions: lxml 4.2.3.0, libxml2 2.9.5, cssselect 1.0.3,
2019-02-28 16:32:27 [scrapy.crawler] INFO: Overridden settings: {'BOT_NAME': 'douyu', 'DOWNLOAD_DELA
2019-02-28 16:32:27 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
'scrapy.extensions.telnet.TelnetConsole',
'scrapy.extensions.logstats.LogStats']
2019-02-28 16:32:28 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
```

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
```