需求：

对数据进行ETL

需求要点：

数据源位置（E），数据转换脚本（T），数据存入位置（L），脚本拼接生成。

需求实现：

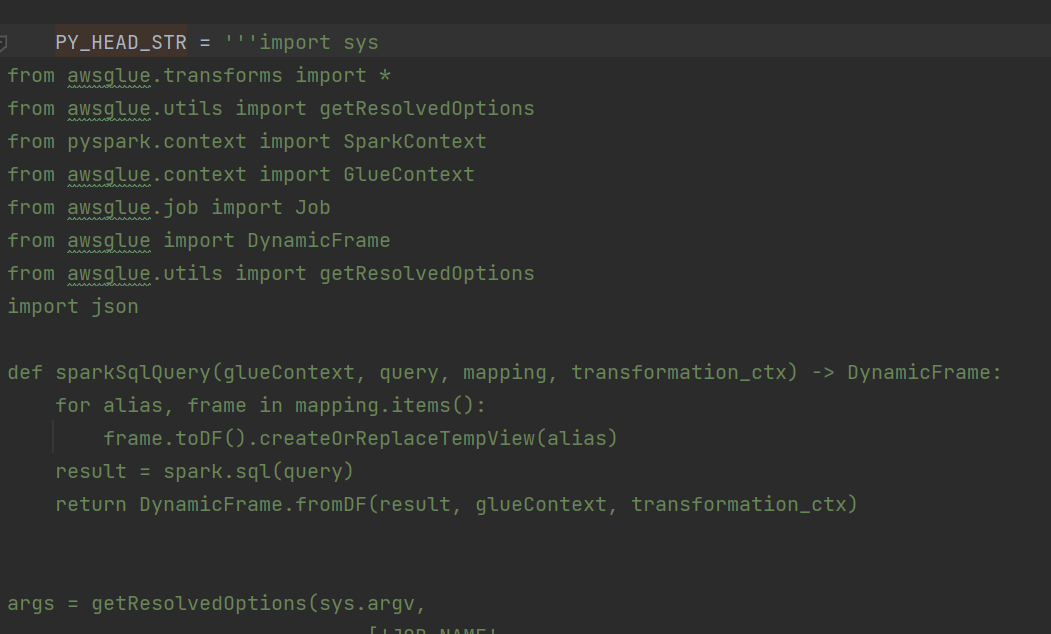
**Glue\_scrpit\_generator.py:**

完成整个ELT过程

**公用部分拼接**：

PY\_HEAD\_STR：内部封装的是公用的参数

py\_head\_str = constants.Constants.PY\_HEAD\_STR



PY\_TAIL\_STR：

py\_tail\_str = constants.Constants.PY\_TAIL\_STR

源文件中：

PY\_TAIL\_STR = 'job.commit()'

**Source：**

数据源位置，每个表都会在表名循环里分别生成自己的source\_ctx和source\_node\_part。source\_ctx会被添加到source\_node\_part，source\_node\_part先被添加到列表py\_source\_str里再进过一个字符串处理将最后加入了一个换行符。

Source\_ctx: table1\_node1948892233232 表名+随机数

source\_node\_part结构如下：

# Script generated for node table1

table1\_node1948892233232 = glueContext.create\_dynamic\_frame.from\_catalog(

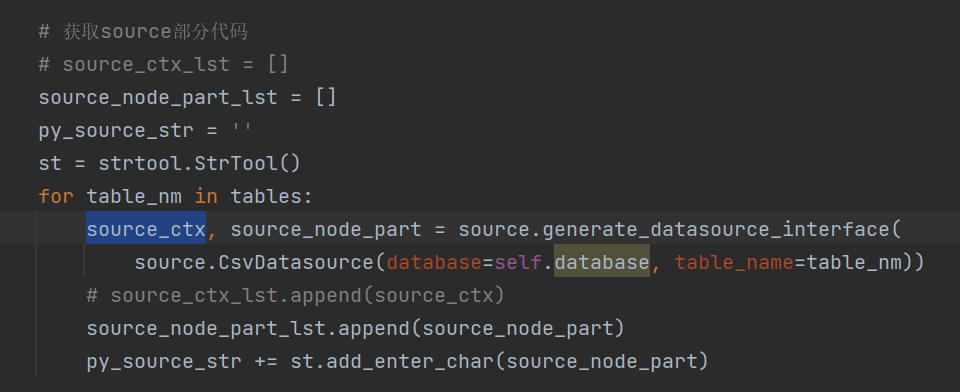
database=mysql,

table\_name="table1",

transformation\_ctx="table1\_node1948892233232",

)

Source在Glue\_scrpit\_generator代码如下：



**target：**

target和source类似，会返回一个bucket name和一个配置文件，在target源文件里被命名为re1和re2, 在glue generator里为re1和

Re1：S3bucket\_node1395973994766 py\_target\_str

Re2：

S3bucket\_node1395973994766 = glueContext.write\_dynamic\_frame.from\_options(

frame=repartition\_frame,

connection\_type="s3",

format="csv",

connection\_options = {

"path": target\_path,

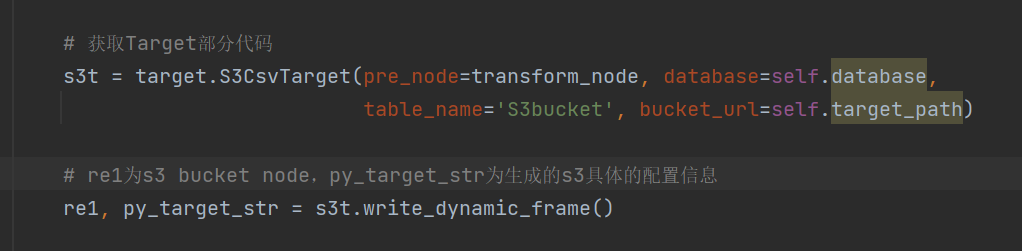
"partitionKeys": [],

},

transformation\_ctx="S3bucket\_node1395973994766",

)

target在Glue\_scrpit\_generator代码如下：



**Transform：**

transform是对目标地址文件的sql进行读取将其封装在spark SQL内。与前面target和source类似，transform会返回SQLTransform\_node和生成的spark SQL脚本，生成的脚本结构如下：

# Script generated for node SQL Query

SqlQuery0 = """

select \* from test;

"""

SQLTransform\_node1092649869238 = sparkSqlQuery(

glueContext,

query=SqlQuery0,

mapping={

"employees":employees\_node1948554591402,

"departments":departments\_node1406226729501,

},

transformation\_ctx="SQLTransform\_node1092649869238",

)

最后将所有生成的脚本进行拼接成一个完整的PySpark任务，并将其放在指定目录下

py\_str = st.concate\_strings\_with\_enter\_char(py\_head\_str, py\_source\_str, py\_transform\_str, py\_target\_str,  
 py\_tail\_str)

# 输出py文件到对应目录  
ft = filetool.FileTool(self.out\_py\_path)  
ft.write\_file(py\_str)  
return self.out\_py\_path