

# ABC final file (by Sumrit)

This file contains the file that followed following procedures

---

- ☐ Have a S3 /RDS etc. as data source and files ABC\_out,ref1,ref2,ref3 uploaded to that source
- ☐ Create a Data base in AWS glue
- ☐ ADD crawlers to crawl files and add tables
- ☐ Use a jupyter notebook for writing codes (optional)
- ☐ Create a job using that code and test in so final file got in result folder of S3target

## @@ Steps for code

### #Glue initiating

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 sc = SparkContext.getOrCreate()
9 glueContext = GlueContext(sc)
10 spark = glueContext.spark_session
11 job = Job(glueContext)
```

### #Creating data source after crawling tables

```
1 abc_out = glueContext.create_dynamic_frame.from_catalog(
2     database = "bh_glue_test",
3     table_name = "abc_out_csv"
4 )
5 ref1 = glueContext.create_dynamic_frame.from_catalog(
6     database = "bh_glue_test",
7     table_name = "ref1_csv"
8 )
9 ref2 = glueContext.create_dynamic_frame.from_catalog(
10    database = "bh_glue_test",
11    table_name = "ref2_csv"
12 )
13 ref3 = glueContext.create_dynamic_frame.from_catalog(
14    database = "bh_glue_test",
15    table_name = "ref3_csv"
16 )
```

### OPTIONAL #can check the schema of tables

```
1 #abc_out.printSchema()
2 #ref1.printSchema()
```

```
3 #ref2.printSchema()
4 #ref3.printSchema()
```

#### #Dynamic Frame to Spark Data Frame

```
1 sparkDf_abc_out = abc_out.toDF()
2 sparkDf_ref1 = ref1.toDF()
3 sparkDf_ref2 = ref2.toDF()
4 sparkDf_ref3 = ref3.toDF()
```

#### OPTIONAL

```
1 #sparkDf_Customer_abc_out.show()
2 #sparkDf_ref1.show()
3 #sparkDf_ref2.show()
4 #sparkDf_ref3.show()
```

#### #import concat from functions and lit one,

#### #for concat to work,nested bracket structure needed

```
1 from pyspark.sql.functions import concat
2 from pyspark.sql.functions import lit
3 sparkDf_abc_out = sparkDf_abc_out.withColumn("com",concat("subnum",concat(lit('-'),'visitid',concat(lit('-'),'vis
4 #need to drop column name "statusid_dec" cause it duplicate in another table and partially filled in abc_out tabl
5 sparkDf_abc_out = sparkDf_abc_out.drop("statusid_dec")
6 sparkDf_ref1 = sparkDf_ref1.withColumn("com1",concat("subnum",concat(lit('-'),'visitid',concat(lit('-'),'visitsec
7 sparkDf_ref2 = sparkDf_ref2.withColumn("com1",concat("subnum",concat(lit('-'),'visitid',concat(lit('-'),'visitsec
8 sparkDf_ref3 = sparkDf_ref3.withColumn("com1",concat("subnum",concat(lit('-'),'visitid',concat(lit('-'),'visitsec
9
```

#### OPTIONAL

```
1 #sparkDf_abc_out.show()
2 # sparkDf_ref1.show()
3 # sparkDf_ref1.show()
4 # sparkDf_ref1.show()
5 # sparkDf_ref1.show()
```

#### #Select columns from spark data frame

```
1 up_ref1 = sparkDf_ref1.select("eamnum","com1")
2 up_ref2 = sparkDf_ref2.select("aenum","com1")
3 up_ref3 = sparkDf_ref3.select("statusid_dec","com1")
```

#### OPTIONAL

```
1 #up_ref1.show()
2 #up_ref2.show()
3 #up_ref3.show()
```

```

4
5 #if unable to see we can shorten the tabke for a change
6 #abd_small = sparkDf_abc_out.select("visitseq","com")
7 #combo = abd_small.join(up_ref1,abd_small.com == sparkDf_ref1.com1,"inner").show(1)

```

#combining abc file and ref1

```

1 combo = sparkDf_abc_out.join(up_ref1,sparkDf_abc_out.com == sparkDf_ref1.com1,"inner")

```

#combining combo and ref2

```

1 combo1 = combo.join(up_ref2,combo.com == up_ref2.com1,"inner")

```

#combining combo1 and ref3

```

1 combo2 = combo1.join(up_ref3,combo1.com == up_ref3.com1,"inner")

```

#Now dropping initial primary key so column doesn't duplicates and then concatenating for a new primary key

```

1 df_final = combo2.drop("com1")
2 df_final = df_final.withColumn("Primary",concat("com",concat(lit('-'),"aenum")))
3 #df_final.printSchema()

```

#now only selected columns are needed

```

1 Result = df_final.select("Primary","reldeva_dec","reldevb_dec","reldevc_dec","reldeva_dec","reldevb_dec","reldevc_dec")

```

#Import Dyanmic DataFrame class

#Convert from Spark Data Frame to Glue Dynamic Frame

```

1 from awsglue.dynamicframe import DynamicFrame
2
3 Result_conv = DynamicFrame.fromDF(Result, glueContext, "convert")
4

```

#transform for a single file and providing path

```

1 retransform = Result_conv.repartition(1)
2
3 glueContext.write_dynamic_frame.from_options(
4     frame=retransform,
5     connection_type="s3",
6     connection_options={"path":"s3://aasumrit-bucket/result_main/"},
7     format="csv",
8     format_options={
9         "separator": ",",
10     },
11     transformation_ctx="datasink01"
12 )

```

#Using custom code to rename file

(this codes copy the output file then change its name and then delete the initial file)

```
1 import boto3
2 client = boto3.client('s3')
3 BUCKET_NAME= "aasumrit-bucket"
4 PREFIX="result_main/"
5 response = client.list_objects(
6     Bucket=BUCKET_NAME,
7     Prefix=PREFIX,
8 )
9 # print(response)
10 name = response["Contents"][0]["Key"]
11 # print(name)
12 client.copy_object(Bucket="aasumrit-bucket", CopySource=BUCKET_NAME+'/'+name, Key=PREFIX+"RESULT_FILE")
13 client.delete_object(Bucket=BUCKET_NAME, Key=name)
14 job.commit()
```

##### COMPLETE CODE #####

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 sc = SparkContext.getOrCreate()
9 glueContext = GlueContext(sc)
10 spark = glueContext.spark_session
11 job = Job(glueContext)
12
13 abc_out = glueContext.create_dynamic_frame.from_catalog(
14     database = "bh_glue_test",
15     table_name = "abc_out_csv"
16 )
17 ref1 = glueContext.create_dynamic_frame.from_catalog(
18     database = "bh_glue_test",
19     table_name = "ref1_csv"
20 )
21 ref2 = glueContext.create_dynamic_frame.from_catalog(
22     database = "bh_glue_test",
23     table_name = "ref2_csv"
24 )
25 ref3 = glueContext.create_dynamic_frame.from_catalog(
26     database = "bh_glue_test",
27     table_name = "ref3_csv"
28 )
29
30 sparkDf_abc_out = abc_out.toDF()
31 sparkDf_ref1 = ref1.toDF()
32 sparkDf_ref2 = ref2.toDF()
33 sparkDf_ref3 = ref3.toDF()
34
35 from pyspark.sql.functions import concat
```

```

36 from pyspark.sql.functions import lit
37
38 sparkDf_abc_out = sparkDf_abc_out.withColumn("com",concat("subnum",concat(lit('-'),'visitid",concat(lit('-'),'vi
39 #need to drop column name "statusid_dec" cause it duplicate in another table and partially filled in abc_out tab
40 sparkDf_abc_out = sparkDf_abc_out.drop("statusid_dec")
41 sparkDf_ref1 = sparkDf_ref1.withColumn("com1",concat("subnum",concat(lit('-'),'visitid",concat(lit('-'),'visitse
42 sparkDf_ref2 = sparkDf_ref2.withColumn("com1",concat("subnum",concat(lit('-'),'visitid",concat(lit('-'),'visitse
43 sparkDf_ref3 = sparkDf_ref3.withColumn("com1",concat("subnum",concat(lit('-'),'visitid",concat(lit('-'),'visitse
44
45 up_ref1 = sparkDf_ref1.select("eamnum","com1")
46 up_ref2 = sparkDf_ref2.select("aenum","com1")
47 up_ref3 = sparkDf_ref3.select("statusid_dec","com1")
48
49 combo = sparkDf_abc_out.join(up_ref1,sparkDf_abc_out.com == sparkDf_ref1.com1,"inner")
50 combo1 = combo.join(up_ref2,combo.com == up_ref2.com1,"inner")
51 combo2 = combo1.join(up_ref3,combo1.com == up_ref3.com1,"inner")
52
53 df_final = combo2.drop("com1")
54 df_final = df_final.withColumn("Primary",concat("com",concat(lit('-'),'aenum")))
55
56 Result = df_final.select("Primary","reldeva_dec","reldevb_dec","reldevc_dec","reldeva_dec","reldevb_dec","reldev
57
58 from awsglue.dynamicframe import DynamicFrame
59 Result_conv = DynamicFrame.fromDF(Result, glueContext, "convert")
60
61 retransform = Result_conv.repartition(1)
62
63 glueContext.write_dynamic_frame.from_options(
64     frame=retransform,
65     connection_type="s3",
66     connection_options={"path":"s3://aasumrit-bucket/result_main/"},
67     format="csv",
68     format_options={
69         "separator": ",",
70     },
71     transformation_ctx="datasink01"
72 )
73
74 import boto3
75 client = boto3.client('s3')
76 BUCKET_NAME= "aasumrit-bucket"
77 PREFIX="result_main/"
78 response = client.list_objects(
79     Bucket=BUCKET_NAME,
80     Prefix=PREFIX,
81 )
82 # print(response)
83 name = response["Contents"][0]["Key"]
84 # print(name)
85 client.copy_object(Bucket="aasumrit-bucket", CopySource=BUCKET_NAME+'/'+name, Key=PREFIX+"RESULT_FILE")
86 client.delete_object(Bucket=BUCKET_NAME, Key=name)
87 job.commit()

```