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

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
```

#Creating data source after crawling tables

```
abc_out = glueContext.create_dynamic_frame.from_catalog(
    database = "bh_glue_test",
    table_name = "abc_out_csv"
)
ref1 = glueContext.create_dynamic_frame.from_catalog(
    database = "bh_glue_test",
    table_name = "ref1_csv"
)
ref2 = glueContext.create_dynamic_frame.from_catalog(
    database = "bh_glue_test",
    table_name = "ref2_csv"
)
ref3 = glueContext.create_dynamic_frame.from_catalog(
    database = "bh_glue_test",
    table_name = "ref3_csv"
)
```

OPTIONAL #can check the schema of tables

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

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

#Dynamic Frame to Spark Data Frame

```
sparkDf_abc_out = abc_out.toDF()
sparkDf_ref1 = ref1.toDF()
sparkDf_ref2 = ref2.toDF()
sparkDf_ref3 = ref3.toDF()
```

OPTIONAL

```
#sparkDf_Customer_abc_out.show()
#sparkDf_ref1.show()
#sparkDf_ref2.show()
#sparkDf_ref3.show()
```

#import concat from functions and lit one,

#for concate to work,nested bracket structure needed

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

```

OPTIONAL

```
#sparkDf_abc_out.show()
# sparkDf_ref1.show()
# sparkDf_ref1.show()
# sparkDf_ref1.show()
# sparkDf_ref1.show()
```

#Select columns from spark data frame

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

OPTIONAL

```
#up_ref1.show()
#up_ref2.show()
#up_ref3.show()
```

```

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

#combining abc file and ref1

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

#combining combo and ref2

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

#combining combo1 and ref3

```
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

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

#now only selected columns are needed

```
Result = df_final.select("Primary","reldeva_dec","reldevb_dec","reldevc_dec","reldeva_dec","reldevb_dec","reldevc
```

#Import Dyanmic DataFrame class

#Convert from Spark Data Frame to Glue Dynamic Frame

```
from awsglue.dynamicframe import DynamicFrame

Result_conv = DynamicFrame.fromDF(Result, glueContext, "convert")

```

#transform for a single file and providing path

```
retransform = Result_conv.repartition(1)

glueContext.write_dynamic_frame.from_options(
frame =retransform,
connection_type = "s3",
connection_options = {"path":"s3://aasumrit-bucket/result_main/"},
format = "csv",
format_options={
"separator": ","
},
transformation_ctx = "datasink01"
)
```

#Using custom code to rename file

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

```
import boto3
client = boto3.client('s3')
BUCKET_NAME= "aasumrit-bucket"
PREFIX="result_main/"
response = client.list_objects(
    Bucket=BUCKET_NAME,
    Prefix=PREFIX,
)
# print(response)
name = response["Contents"][0]["Key"]
# print(name)
client.copy_object(Bucket="aasumrit-bucket", CopySource=BUCKET_NAME+'/'+name, Key=PREFIX+"RESULT_FILE")
client.delete_object(Bucket=BUCKET_NAME, Key=name)
job.commit()
```

COMPLETE CODE

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

abc_out = glueContext.create_dynamic_frame.from_catalog(
    database = "bh_glue_test",
    table_name = "abc_out_csv"
)
ref1 = glueContext.create_dynamic_frame.from_catalog(
    database = "bh_glue_test",
    table_name = "ref1_csv"
)
ref2 = glueContext.create_dynamic_frame.from_catalog(
    database = "bh_glue_test",
    table_name = "ref2_csv"
)
ref3 = glueContext.create_dynamic_frame.from_catalog(
    database = "bh_glue_test",
    table_name = "ref3_csv"
)

sparkDf_abc_out = abc_out.toDF()
sparkDf_ref1 = ref1.toDF()
sparkDf_ref2 = ref2.toDF()
sparkDf_ref3 = ref3.toDF()

from  pyspark.sql.functions import concat
```

```
from  pyspark.sql.functions import lit

sparkDf_abc_out = sparkDf_abc_out.withColumn("com",concat("subnum",concat(lit('-'),"visitid",concat(lit('-'),"vi
#need to drop column name "statusid_dec" cause it duplicate in another table and partially filled in abc_out tab
sparkDf_abc_out = sparkDf_abc_out.drop("statusid_dec")
sparkDf_ref1 = sparkDf_ref1.withColumn("com1",concat("subnum",concat(lit('-'),"visitid",concat(lit('-'),"visitse
sparkDf_ref2 = sparkDf_ref2.withColumn("com1",concat("subnum",concat(lit('-'),"visitid",concat(lit('-'),"visitse
sparkDf_ref3 = sparkDf_ref3.withColumn("com1",concat("subnum",concat(lit('-'),"visitid",concat(lit('-'),"visitse

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

combo = sparkDf_abc_out.join(up_ref1,sparkDf_abc_out.com ==  sparkDf_ref1.com1,"inner")
combo1 = combo.join(up_ref2,combo.com ==  up_ref2.com1,"inner")
combo2 = combo1.join(up_ref3,combo1.com ==  up_ref3.com1,"inner")

df_final = combo2.drop("com1")
df_final = df_final.withColumn("Primary",concat("com",concat(lit('-'),"aenum")))

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

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

retransform = Result_conv.repartition(1)

glueContext.write_dynamic_frame.from_options(
frame =retransform,
connection_type = "s3",
connection_options = {"path":"s3://aasumrit-bucket/result_main/"},
format = "csv",
format_options={
"separator": ","
},
transformation_ctx = "datasink01"
)

import boto3
client = boto3.client('s3')
BUCKET_NAME= "aasumrit-bucket"
PREFIX="result_main/"
response = client.list_objects(
    Bucket=BUCKET_NAME,
    Prefix=PREFIX,
)
# print(response)
name = response["Contents"][0]["Key"]
# print(name)
client.copy_object(Bucket="aasumrit-bucket", CopySource=BUCKET_NAME+'/'+name, Key=PREFIX+"RESULT_FILE")
client.delete_object(Bucket=BUCKET_NAME, Key=name)
job.commit()
```