**ETL job**

##Created By Mudit For loading MongoDB collections Data into S3.

**Importing the necessary packages:**

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

from pymongo import MongoClient

from pyspark.sql.functions import udf

from pyspark.sql.types import \*

from pyspark.sql.functions import lit

from pyspark.sql.functions import col

import time

from datetime import date

import boto3

from pymongo.mongo\_client import MongoClient

from pymongo.server\_api import ServerApi

sc = SparkContext.getOrCreate()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

job = Job(glueContext)

**Taking external parameters from Glue:**

args = getResolvedOptions(sys.argv,['db','bucket','coupons\_db','mongo\_uri'])

db\_name =args['db']

bucket\_name=args['bucket']

coupons\_db =args['coupons\_db']

mongo\_uri =args['mongo\_uri']

**Function: get\_all\_collections**

def get\_all\_collections(db):

#Get names of all the collection in MongoDB Database

uri = mongo\_uri

client = MongoClient(uri, server\_api=ServerApi('1'))

mydatabase = client[db]

collections = mydatabase.list\_collection\_names()

return collections

This function gets names of all the collections in a database in MongoDB. It establishes a connection with MongoDB cluster with provided URI and gets the list of all the collection in the database that we provide as a parameter to the function. And returns a list of the collection names.

**Function: collection\_lists**

def collection\_lists(collections):

#Create list of all the static and Dynamic Collections

tors = []

dors = []

vectortable4\_1 =[]

vectortable4\_2 =[]

vectortable6=[]

vectortable7=[]

vectortable8=[]

vectortable9=[]

static\_coll =[]

for i in collections :

if '\_tors' in i :

tors.append(i)

if '\_dors' in i :

dors.append(i)

if '\_vectortable4\_1' in i :

vectortable4\_1.append(i)

if '\_vectortable4\_2' in i :

vectortable4\_2.append(i)

if '\_vectortable6' in i :

vectortable6.append(i)

if '\_vectortable7' in i :

vectortable7.append(i)

if '\_vectortable8' in i :

vectortable8.append(i)

if '\_vectortable9' in i :

vectortable9.append(i)

static\_coll = list (set(collections) -set(tors) -set(dors) - set(vectortable4\_1) -set(vectortable6) -set(vectortable7)-set(vectortable8)-set(vectortable9))

return tors,dors,vectortable4\_1,vectortable4\_2,static\_coll , vectortable6,vectortable7,vectortable8,vectortable9

This function stores the dynamic collections in the list with the collection's name according to the presence of substring in the collection's name. And it stores all the static collections in a list called static\_coll. And then returns all the lists.

**Function: none\_type\_to\_str**

def none\_type\_to\_str(df):

# get dataframe schema

my\_schema = list(df.schema)

null\_cols = []

# iterate over schema list to filter for NullType columns

for st in my\_schema:

if str(st.dataType) == 'NullType':

null\_cols.append(st)

# cast null type columns to string (or whatever you'd like)

for ncol in null\_cols:

mycolname = str(ncol.name)

df = df \

.withColumn(mycolname, df[mycolname].cast('string'))

return df

The above function takes a DataFrame as argument and converts columns which are of NullType in the DataFrame to String. And returns the same DataFrame back.

# import the time module

**Function: mongo\_to\_DF**

def mongo\_to\_DF(collection\_name,DB\_Name,mongo\_uri):

#Returns A MongoDB collection in a DF

try:

df = spark.read.format("com.mongodb.spark.sql.DefaultSource")\

.option("uri", mongo\_uri)\

.option("spark.mongodb.input.database",DB\_Name)\

.option("spark.mongodb.input.collection",collection\_name)\

.load

#df = none\_type\_to\_str(df)

return df

except Exception as e:

return (str(e) + ' in collection -->' + collection\_name )

This function tries to load the MongoDB collection into a DataFrame. If successful, it returns the DataFrame. Otherwise, it returns an error message along with the name of the collection where the error occurred.

**Function: write\_to\_s3\_storage**

def write\_to\_s3\_storage(df,bucket\_name,folder\_name,file\_name,coll\_name,dt):

#write to S3 Storage

s3 = boto3.client('s3')

s3\_url = 's3://' + bucket\_name + '/' +folder\_name + '/' + coll\_name + '/' +dt + '/'

print(s3\_url)

s3.put\_object(Bucket=bucket\_name , Key=(folder\_name + '/'))

df.write.json(s3\_url + file\_name)

The function takes dataframe, bucket name, folder name, file name, collection name and date as input parameters. Create a URI by concatenating bucket name, folder name, collection name and date. Using s3.put\_object it creates necessary folder structure. And then writes the dataframe into the location in the S3 bucket.

**Function: delete\_from\_S3**

def delete\_from\_S3(bucket,folder\_location):

folder = folder\_location +'/'

s3 = boto3.resource('s3')

try:

bucket = s3.Bucket(bucket)

bucket.objects.filter(Prefix=folder).delete()

return True

except Exception as e:

print(f"Failed to delete s3 folder : {e}")

return False

This function is used to delete all the objects in the specified folder in the bucket. If deletion is successful it returns true else, it returns false.

**Function: save\_collections\_to\_S3**

def save\_collections\_to\_S3(coll\_list,folder\_name,db,bucket\_name,mongo\_uri):

#save Mongo collection to S3 . Parameters --> list of collection, Folder Name

import time

from datetime import date

dt = date.today()

# get the current time in seconds since the epoch

seconds = time.time()

try:

for i in coll\_list:

try:

df = spark.read.format("com.mongodb.spark.sql.DefaultSource")\

.option("uri", mongo\_uri)\

.option("spark.mongodb.input.database",db)\

.option("spark.mongodb.input.collection",i).load()

#df = df.fillna("")

bucket\_name = bucket\_name

#folder\_name = db + '/' + folder\_name

coll\_name =i

file\_name = i + '\_' + str(int( time.time())) +'.json'

print(file\_name)

#df.show()

if df.count() > 0 :

write\_to\_s3\_storage(df,bucket\_name,db +'/'+ folder\_name,file\_name,coll\_name,str(dt))

else:

print("collection has 0 records: " + coll\_name)

except Exception as e:

print(str(e) + ' in collection' + i)

continue

except Exception as e:

print(str(e) + ' in collection' + i)

This function performs the task of saving the MongoDB collections into S3 bucket.

It takes 3 input parameters i.e. the list of collections, folder name and the database name. The second and the third parameters are used to construct the path to save the collection in S3.

Inside the function, we iterate through the collection list, convert each collection into a dataframe, and if the dataframe contains records, we write the dataframe into S3.

**Function: main**

def main():

db = db\_name

testingdb = get\_all\_collections(db)

coupons = get\_all\_collections("coupons")

tors,dors,vectortable4\_1,vectortable4\_2,static\_coll , vectortable6,vectortable7,vectortable8,vectortable9 = collection\_lists(testingdb)

static\_coll

coll\_folders =['tors','dors','vectortable4\_1','vectortable4\_2','static\_coll' , 'vectortable6','vectortable7','vectortable8','vectortable9']

coll\_list=[tors,dors,vectortable4\_1,vectortable4\_2,static\_coll , vectortable6,vectortable7,vectortable8,vectortable9]

i =0

bucket = bucket\_name

folder\_location = db

delete\_from\_S3(bucket,folder\_location)

while i<len(coll\_folders):

save\_collections\_to\_S3(coll\_list[i],coll\_folders[i],folder\_location,bucket,mongo\_uri)

i =i+1

db = coupons\_db

coupons = get\_all\_collections(db)

bucket = bucket\_name

folder\_location = db

delete\_from\_S3(bucket,folder\_location)

save\_collections\_to\_S3(coupons,db,folder\_location,bucket,mongo\_uri)

if \_\_name\_\_ == "\_\_main\_\_":

main()

job.commit()

This is where we use the functions defined above and make the Glue job happen.

From the “fake-data-v2" we separate the static and the dynamic collections. All the dynamic collections are taken in the list with the name according to the substring in the name of the collections and all the static collections are stored in the list called static\_coll. SO that we have all our static collections in one folder.