SCALA - AWS

1 JOB NAME : anif\_scala\_1

import com.amazonaws.services.glue.GlueContext

import com.amazonaws.services.glue.MappingSpec

import com.amazonaws.services.glue.errors.CallSite

import com.amazonaws.services.glue.util.GlueArgParser

import com.amazonaws.services.glue.util.Job

import com.amazonaws.services.glue.util.JsonOptions

import org.apache.spark.SparkContext

import scala.collection.JavaConverters.\_

import com.amazonaws.services.glue.DynamicFrame

import org.apache.spark.sql.SQLContext

import com.amazonaws.services.glue.errors.CallSite

object SparkSqlQuery {

def execute(glueContext: GlueContext, sqlContext: SQLContext, query: String, mapping: Map[String, DynamicFrame]) : DynamicFrame = {

for ((alias, frame) <- mapping) {

frame.toDF().createOrReplaceTempView(alias)

}

val resultDataFrame = sqlContext.sql(query)

return DynamicFrame(resultDataFrame, glueContext)

}

}

object GlueApp {

def main(sysArgs: Array[String]) {

val spark: SparkContext = new SparkContext()

val glueContext: GlueContext = new GlueContext(spark)

val sql: SQLContext = new SQLContext(spark)

// @params: [JOB\_NAME]

val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB\_NAME").toArray)

Job.init(args("JOB\_NAME"), glueContext, args.asJava)

// Script generated for node S3 bucket

val S3bucket\_node1 = glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"", "withHeader": true, "separator": ","}"""), connectionType="s3", format="csv", options=JsonOptions("""{"paths": ["s3://training-labtarget1/TrainingLab/Abble/INPUT/filtered\_covid\_data\_new.csv"], "recurse": true}"""), transformationContext="S3bucket\_node1").getDynamicFrame()

// Script generated for node SQL

val SqlQuery0: String = """select \* from myDataSource

|""".stripMargin

val SQL\_node1646365344251 = SparkSqlQuery.execute(glueContext = glueContext, sqlContext = sql, query = SqlQuery0, mapping = Map("myDataSource" -> S3bucket\_node1)

)

// Script generated for node Amazon S3

val repartition = SQL\_node1646365344251.repartition(1)

val AmazonS3\_node1646365350757 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Abble/OUTPUT/scala/", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646365350757", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

Output



2 JOB NAME : anif\_scala\_2\_crawlers

import com.amazonaws.services.glue.GlueContext

import com.amazonaws.services.glue.MappingSpec

import com.amazonaws.services.glue.errors.CallSite

import com.amazonaws.services.glue.util.GlueArgParser

import com.amazonaws.services.glue.util.Job

import com.amazonaws.services.glue.util.JsonOptions

import org.apache.spark.SparkContext

import scala.collection.JavaConverters.\_

import com.amazonaws.services.glue.DynamicFrame

import org.apache.spark.sql.SQLContext

import com.amazonaws.services.glue.errors.CallSite

object SparkSqlQuery {

def execute(glueContext: GlueContext, sqlContext: SQLContext, query: String, mapping: Map[String, DynamicFrame]) : DynamicFrame = {

for ((alias, frame) <- mapping) {

frame.toDF().createOrReplaceTempView(alias)

}

val resultDataFrame = sqlContext.sql(query)

return DynamicFrame(resultDataFrame, glueContext)

}

}

object GlueApp {

def main(sysArgs: Array[String]) {

val spark: SparkContext = new SparkContext()

val glueContext: GlueContext = new GlueContext(spark)

val sql: SQLContext = new SQLContext(spark)

// @params: [JOB\_NAME]

val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB\_NAME").toArray)

Job.init(args("JOB\_NAME"), glueContext, args.asJava)

// Script generated for node S3 bucket

val S3bucket\_node1 = glueContext.getCatalogSource(database="anif\_covid\_db", tableName="filtered\_covid\_data\_new\_csv", transformationContext="S3bucket\_node1").getDynamicFrame()

// Script generated for node SQL

val SqlQuery0: String = """select total\_cases,new\_cases,total\_deaths,(total\_cases-new\_cases) as recovered from myDataSource

|""".stripMargin

val SQL\_node1646635693652 = SparkSqlQuery.execute(glueContext = glueContext, sqlContext = sql, query = SqlQuery0, mapping = Map("myDataSource" -> S3bucket\_node1)

)

// Script generated for node Amazon S3

val repartition=SQL\_node1646635693652.repartition(1)

val AmazonS3\_node1646635822978 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Abble/OUTPUT/scala/", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646635822978", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

Output



3 JOB NAME : anif\_scala\_3

import com.amazonaws.services.glue.GlueContext

import com.amazonaws.services.glue.MappingSpec

import com.amazonaws.services.glue.errors.CallSite

import com.amazonaws.services.glue.util.GlueArgParser

import com.amazonaws.services.glue.util.Job

import com.amazonaws.services.glue.util.JsonOptions

import org.apache.spark.SparkContext

import scala.collection.JavaConverters.\_

import com.amazonaws.services.glue.DynamicFrame

import org.apache.spark.sql.SQLContext

import com.amazonaws.services.glue.errors.CallSite

object SparkSqlQuery {

def execute(glueContext: GlueContext, sqlContext: SQLContext, query: String, mapping: Map[String, DynamicFrame]) : DynamicFrame = {

for ((alias, frame) <- mapping) {

frame.toDF().createOrReplaceTempView(alias)

}

val resultDataFrame = sqlContext.sql(query)

return DynamicFrame(resultDataFrame, glueContext)

}

}

object GlueApp {

def main(sysArgs: Array[String]) {

val spark: SparkContext = new SparkContext()

val glueContext: GlueContext = new GlueContext(spark)

val sql: SQLContext = new SQLContext(spark)

// @params: [JOB\_NAME]

val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB\_NAME").toArray)

Job.init(args("JOB\_NAME"), glueContext, args.asJava)

// Script generated for node S3 bucket

val S3bucket\_node1 = glueContext.getCatalogSource(database="anif\_covid\_db", tableName="filtered\_covid\_data\_new\_csv", transformationContext="S3bucket\_node1").getDynamicFrame()

// Script generated for node SQL

val SqlQuery0: String = """select total\_cases,new\_cases,total\_deaths,(total\_cases-new\_cases) as recovered from myDataSource

|""".stripMargin

val SQL\_node1646635693652 = SparkSqlQuery.execute(glueContext = glueContext, sqlContext = sql, query = SqlQuery0, mapping = Map("myDataSource" -> S3bucket\_node1)

)

// Script generated for node Amazon S3

val repartition=SQL\_node1646635693652.repartition(1)

val AmazonS3\_node1646635822978 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Abble/OUTPUT/scala/", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646635822978", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

Output



4 JOB NAME : anif\_scala\_4

import com.amazonaws.services.glue.GlueContext

import com.amazonaws.services.glue.MappingSpec

import com.amazonaws.services.glue.errors.CallSite

import com.amazonaws.services.glue.util.GlueArgParser

import com.amazonaws.services.glue.util.Job

import com.amazonaws.services.glue.util.JsonOptions

import org.apache.spark.SparkContext

import scala.collection.JavaConverters.\_

import com.amazonaws.services.glue.DynamicFrame

import org.apache.spark.sql.SQLContext

import com.amazonaws.services.glue.errors.CallSite

object SparkSqlQuery {

def execute(glueContext: GlueContext, sqlContext: SQLContext, query: String, mapping: Map[String, DynamicFrame]) : DynamicFrame = {

for ((alias, frame) <- mapping) {

frame.toDF().createOrReplaceTempView(alias)

}

val resultDataFrame = sqlContext.sql(query)

return DynamicFrame(resultDataFrame, glueContext)

}

}

object GlueApp {

def main(sysArgs: Array[String]) {

val spark: SparkContext = new SparkContext()

val glueContext: GlueContext = new GlueContext(spark)

val sql: SQLContext = new SQLContext(spark)

// @params: [JOB\_NAME]

val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB\_NAME").toArray)

Job.init(args("JOB\_NAME"), glueContext, args.asJava)

// Script generated for node S3 bucket

val S3bucket\_node1 = glueContext.getCatalogSource(database="anif\_covid\_db", tableName="filtered\_covid\_data\_new\_csv", transformationContext="S3bucket\_node1").getDynamicFrame()

// Script generated for node SQL

val SqlQuery0: String = """select sum(new\_cases) from myDataSource where date between '2/17/2020' and '3/21/2020'

|""".stripMargin

val SQL\_node1646647990662 = SparkSqlQuery.execute(glueContext = glueContext, sqlContext = sql, query = SqlQuery0, mapping = Map("myDataSource" -> S3bucket\_node1)

)

// Script generated for node Amazon S3

val repartition=SQL\_node1646647990662.repartition(1)

val AmazonS3\_node1646648056654 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Abble/OUTPUT/scala/", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646648056654", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

Output



5 JOB NAME : anif\_scala\_5

import com.amazonaws.services.glue.GlueContext

import com.amazonaws.services.glue.MappingSpec

import com.amazonaws.services.glue.errors.CallSite

import com.amazonaws.services.glue.util.GlueArgParser

import com.amazonaws.services.glue.util.Job

import com.amazonaws.services.glue.util.JsonOptions

import org.apache.spark.SparkContext

import scala.collection.JavaConverters.\_

import com.amazonaws.services.glue.DynamicFrame

import org.apache.spark.sql.SQLContext

import com.amazonaws.services.glue.errors.CallSite

object SparkSqlQuery {

def execute(glueContext: GlueContext, sqlContext: SQLContext, query: String, mapping: Map[String, DynamicFrame]) : DynamicFrame = {

for ((alias, frame) <- mapping) {

frame.toDF().createOrReplaceTempView(alias)

}

val resultDataFrame = sqlContext.sql(query)

return DynamicFrame(resultDataFrame, glueContext)

}

}

object GlueApp {

def main(sysArgs: Array[String]) {

val spark: SparkContext = new SparkContext()

val glueContext: GlueContext = new GlueContext(spark)

val sql: SQLContext = new SQLContext(spark)

// @params: [JOB\_NAME]

val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB\_NAME").toArray)

Job.init(args("JOB\_NAME"), glueContext, args.asJava)

// Script generated for node S3 bucket

val S3bucket\_node1 = glueContext.getCatalogSource(database="anif\_covid\_db", tableName="filtered\_covid\_data\_new\_csv", transformationContext="S3bucket\_node1").getDynamicFrame()

// Script generated for node SQL

val SqlQuery0: String = """select first(date) from myDataSource where continent ='Asia'

|

|""".stripMargin

val SQL\_node1646648637645 = SparkSqlQuery.execute(glueContext = glueContext, sqlContext = sql, query = SqlQuery0, mapping = Map("myDataSource" -> S3bucket\_node1)

)

// Script generated for node Amazon S3

val repartition=SQL\_node1646648637645.repartition(1)

val AmazonS3\_node1646648681418 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Abble/OUTPUT/scala/", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646648681418", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

Output

