**AWS Scala Practice**

**Job Name: scala\_test \_1**

**Script:**

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 AWS Glue Data Catalog

val AWSGlueDataCatalog\_node1646633936301 = glueContext.getCatalogSource(database="tanuja\_db", tableName="covid\_filtered\_data\_csv", transformationContext="AWSGlueDataCatalog\_node1646633936301").getDynamicFrame()

// Script generated for node SQL

val SqlQuery0: String = """select location,sum(total\_cases) from myDataSource group by location

|""".stripMargin

val SQL\_node1646633973725 = SparkSqlQuery.execute(glueContext = glueContext, sqlContext = sql, query = SqlQuery0, mapping = Map("myDataSource " -> AWSGlueDataCatalog\_node1646633936301)

)

// Script generated for node Amazon S3

val repartition=SQL\_node1646633973725.repartition(1)

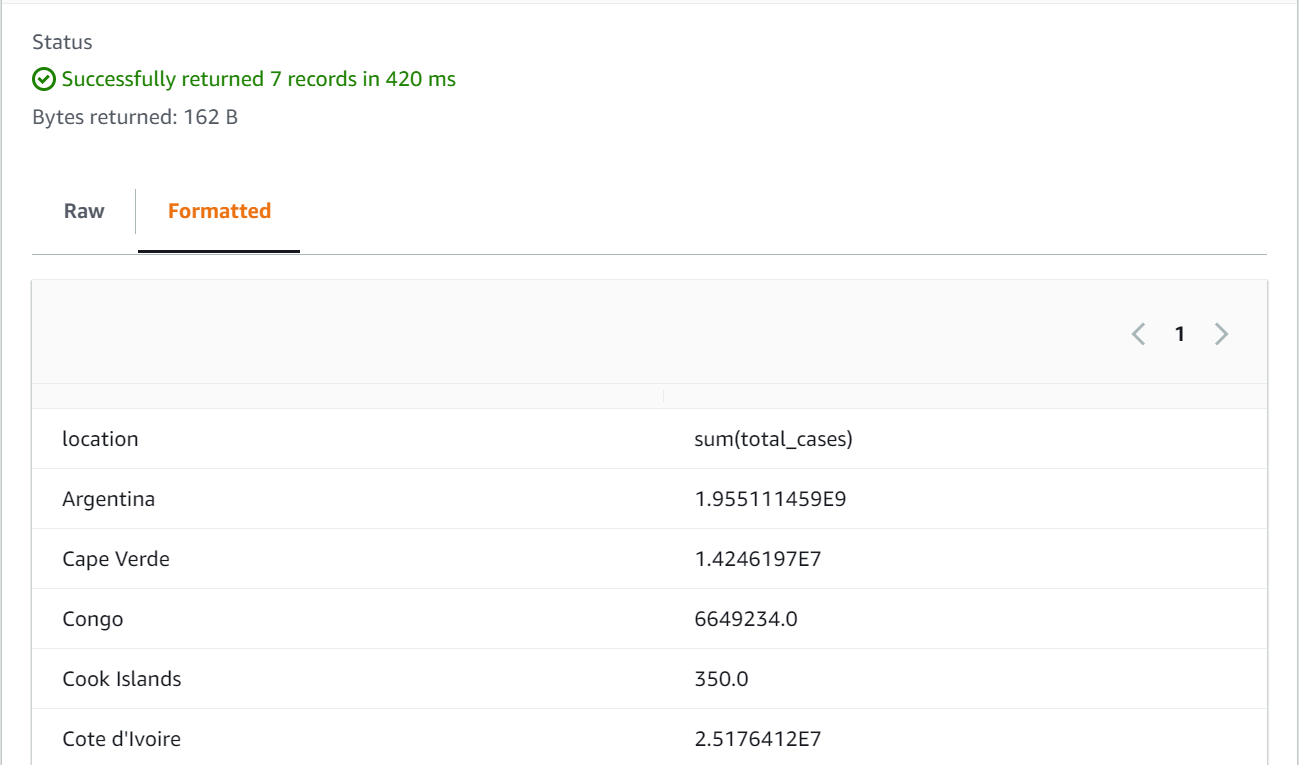
val AmazonS3\_node1646634397536 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Saitanuja /OUTPUT/Scala/scala\_test \_1/", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646634397536", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

**Output:**



**Job Name: scala\_test\_2**

**Script:**

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/Saitanuja /OUTPUT/ Usecase\_Covid/Covid\_Filtered\_Data.csv"], "recurse": true}"""), transformationContext="S3bucket\_node1").getDynamicFrame()

// Script generated for node SQL

val SqlQuery0: String = """select location, sum(total\_cases) as active\_cases ,sum(icu\_patients) as critical\_cases from myDataSource group by location

|

|""".stripMargin

val SQL\_node1646636888083 = SparkSqlQuery.execute(glueContext = glueContext, sqlContext = sql, query = SqlQuery0, mapping = Map("myDataSource" -> S3bucket\_node1)

)

// Script generated for node Amazon S3

val repartition=SQL\_node1646636888083.repartition(1)

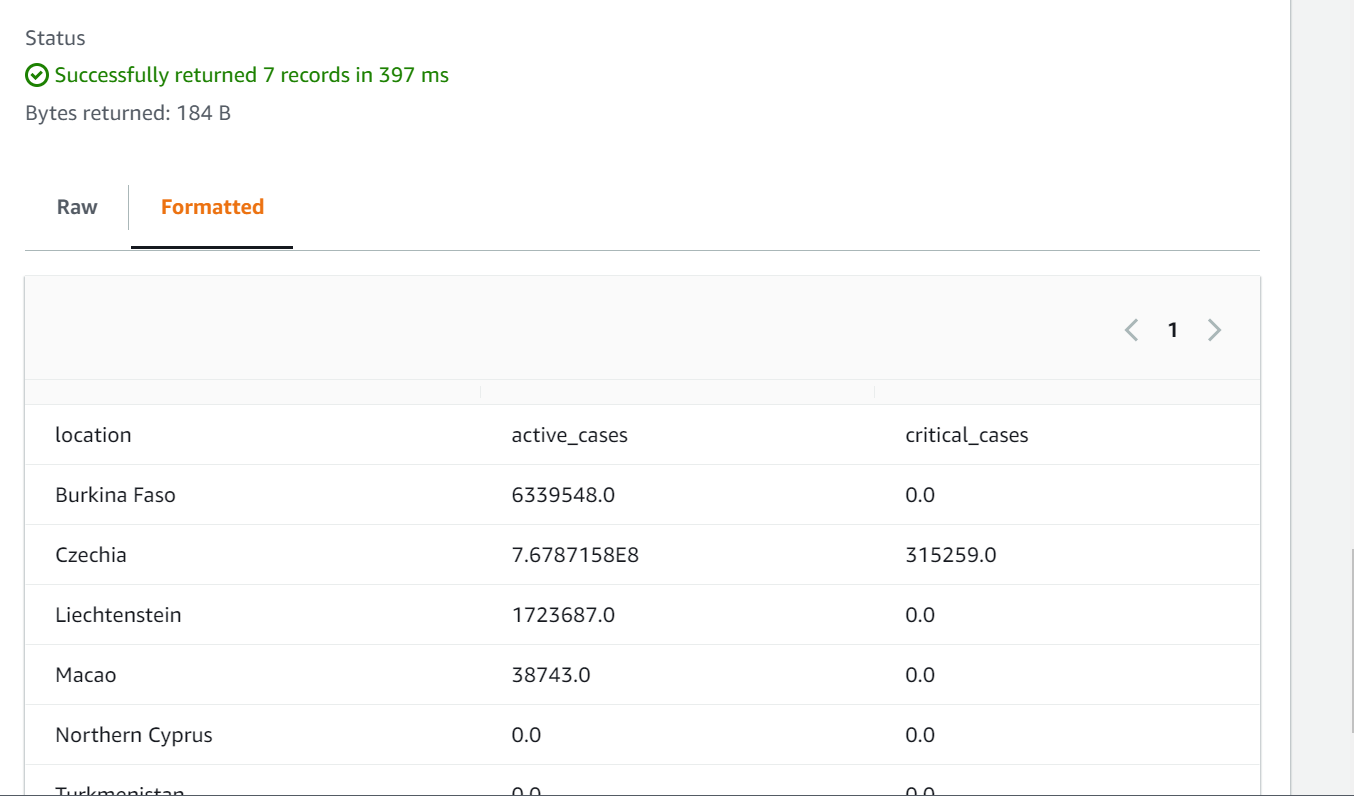
val AmazonS3\_node1646636966045 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Saitanuja/OUTPUT/Scala/scala\_test\_2/", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646636966045", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

**Output:**



**Job Name: scala\_test \_3**

**Script:**

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.\_

object GlueApp {

def main(sysArgs: Array[String]) {

val spark: SparkContext = new SparkContext()

val glueContext: GlueContext = new GlueContext(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 Amazon Redshift

val AmazonRedshift\_node1646638206955 = glueContext.getCatalogSource(database="dev", redshiftTmpDir=args("TempDir"), tableName="Covid\_Usecase10", transformationContext="AmazonRedshift\_node1646638206955").getDynamicFrame()

// Script generated for node Select Fields

val SelectFields\_node1646638247337 = AmazonRedshift\_node1646638206955.selectFields(paths=Seq("total\_cases", "total\_deaths", "recoveries"), transformationContext="SelectFields\_node1646638247337")

// Script generated for node Amazon S3

val repartition=SelectFields\_node1646638247337.repartition(1)

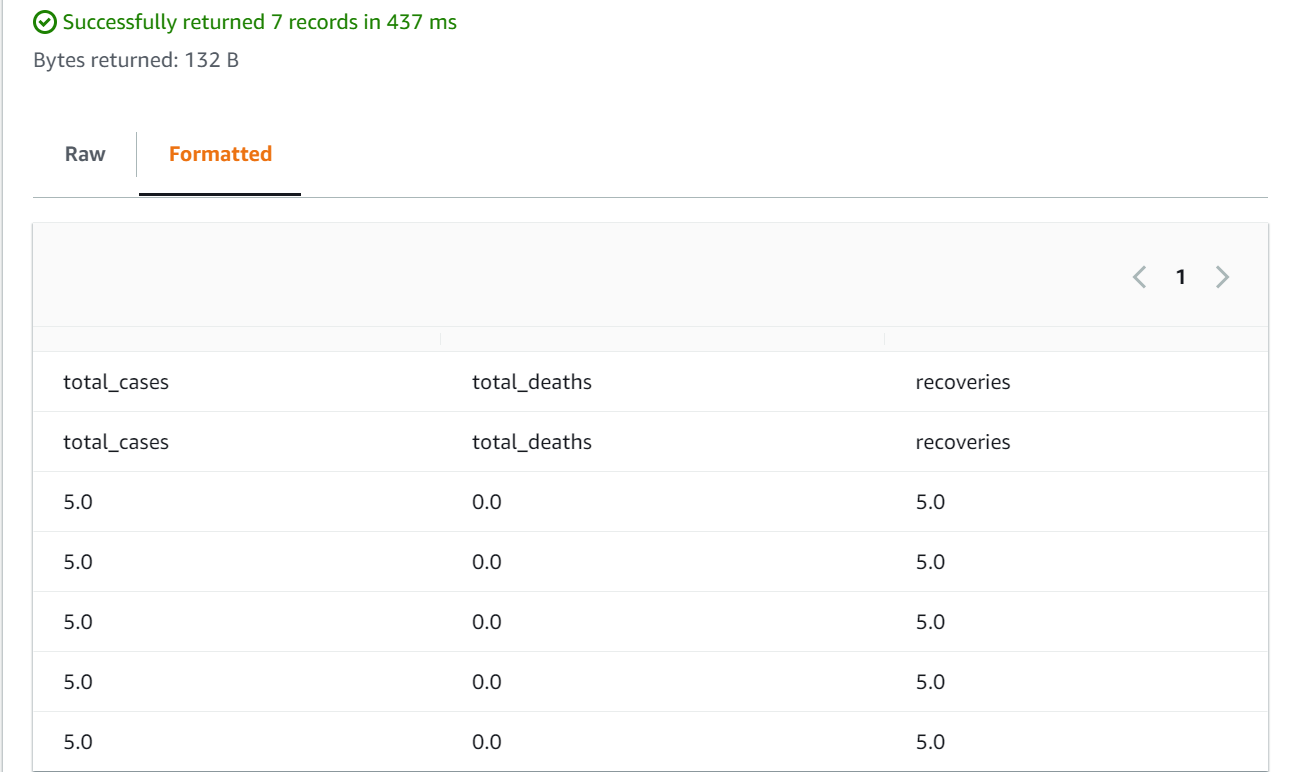
val AmazonS3\_node1646638270814 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Saitanuja /OUTPUT/Scala/scala\_test\_3/", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646638270814", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

**Output:**



**Job Name: scala\_test \_4**

**Script:**

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 Amazon S3

val AmazonS3\_node1646642980603 = glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"", "withHeader": true, "separator": ","}"""), connectionType="s3", format="csv", options=JsonOptions("""{"paths": ["s3://training-labtarget1/TrainingLab/Saitanuja /OUTPUT/Usecase\_Covid/Covid\_Filtered\_Data.csv"], "recurse": true}"""), transformationContext="AmazonS3\_node1646642980603").getDynamicFrame()

// Script generated for node SQL

val SqlQuery0: String = """select date, new\_cases from myDataSource order by new\_cases desc

|""".stripMargin

val SQL\_node1646643036791 = SparkSqlQuery.execute(glueContext = glueContext, sqlContext = sql, query = SqlQuery0, mapping = Map("myDataSource " -> AmazonS3\_node1646642980603)

)

// Script generated for node Amazon S3

val repartition=SQL\_node1646643036791.repartition(1)

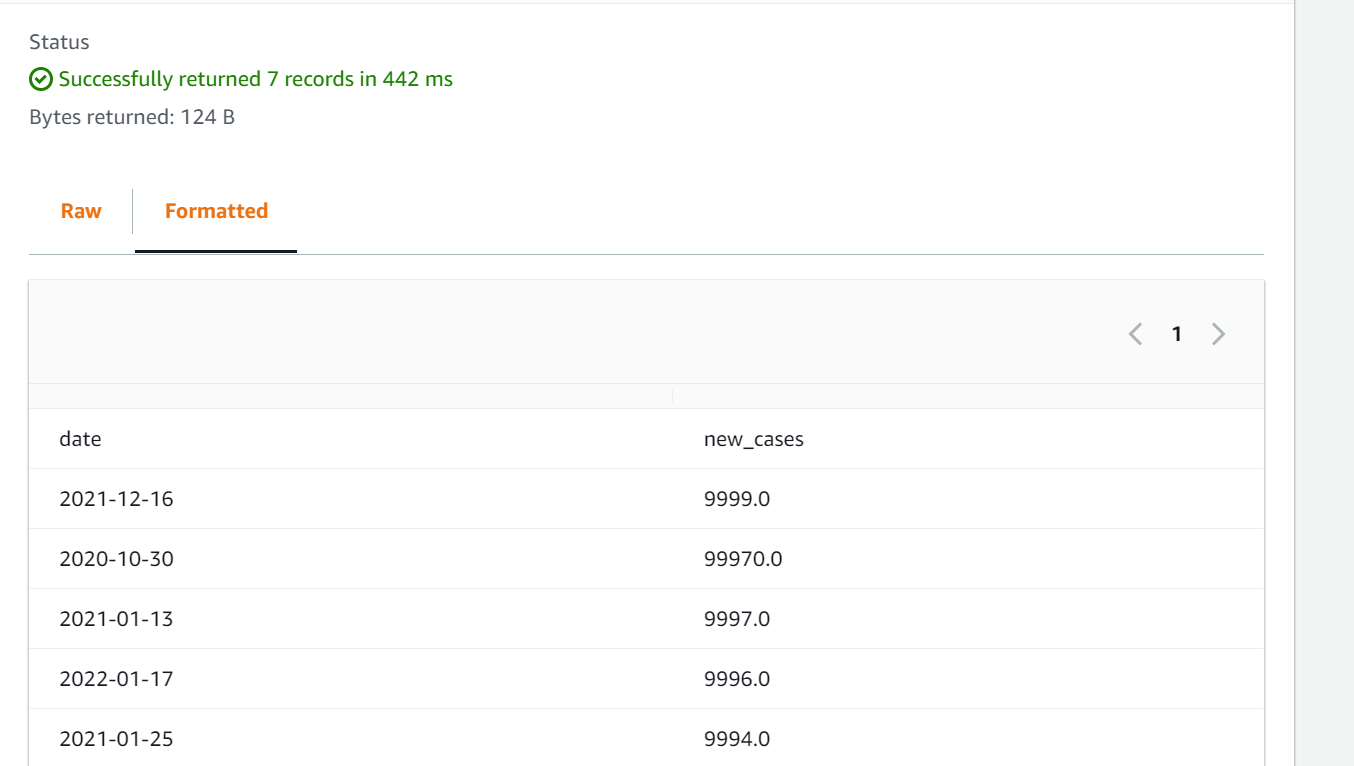
val AmazonS3\_node1646643101992 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Saitanuja /OUTPUT/Scala/scala\_test \_4", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646643101992", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

**Output:**



**Job Name: scala\_test \_5**

**Script:**

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/Saitanuja /OUTPUT/Usecase\_Covid/Covid\_Filtered\_Data.csv"], "recurse": true}"""), transformationContext="S3bucket\_node1").getDynamicFrame()

// Script generated for node SQL

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

val SQL\_node1646645020148 = SparkSqlQuery.execute(glueContext = glueContext, sqlContext = sql, query = SqlQuery0, mapping = Map("myDataSource " -> S3bucket\_node1)

)

// Script generated for node Amazon S3

val repartition=SQL\_node1646645020148.repartition(1)

val AmazonS3\_node1646645050653 = glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path": "s3://training-labtarget1/TrainingLab/Saitanuja /OUTPUT/Scala/scala\_test \_5/", "partitionKeys": []}"""), transformationContext="AmazonS3\_node1646645050653", format="csv").writeDynamicFrame(repartition)

Job.commit()

}

}

**Output:**

