**Script: Sales CSV To Parquest**

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

args = getResolvedOptions(sys.argv, ["my-sales-job-csv-to-parquet"])

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

job = Job(glueContext)

job.init(args["my-sales-job-csv-to-parquet"], args)

# Script generated for node S3 bucket

S3bucket\_node1 = glueContext.create\_dynamic\_frame.from\_options(

format\_options={"quoteChar": '"', "withHeader": True, "separator": ","},

connection\_type="s3",

format="csv",

connection\_options={

"paths": ["s3://my-sales-bucket/Project\_Dataset\_1.csv"],

"recurse": True,

},

transformation\_ctx="S3bucket\_node1",

)

# Script generated for node S3 bucket

S3bucket\_node2 = glueContext.write\_dynamic\_frame.from\_options(

frame=S3bucket\_node1,

connection\_type="s3",

format="glueparquet",

connection\_options={"path": "s3://mysalesdestinationbucket", "partitionKeys": []},

format\_options={"compression": "snappy"},

transformation\_ctx="S3bucket\_node2",

)

job.commit()

**Script:- Parquet\_to\_Catalogue**

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

## @params: [Parquet\_to\_Catalogue]

args = getResolvedOptions(sys.argv, ['Parquet\_to\_Catalogue'])

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

job = Job(glueContext)

job.init(args['Parquet\_to\_Catalogue'], args)

# Creating a dynamic frame from the catalog

datasource0 = glueContext.create\_dynamic\_frame.from\_catalog(database="my-sales-database", table\_name="project\_dataset\_1\_csv")

from pyspark.sql import functions as F

def MyTransformations(input\_data\_frame):

# Resolve choice for handling corrupt records

input\_data\_frame = input\_data\_frame.resolveChoice(specs=[('\_corrupt\_record', 'cast:long')])

# Transformation: Calculate total sales for each product

total\_sales\_df = input\_data\_frame.groupBy("StockCode", "Description") \

.agg(F.sum("Quantity").alias("total\_sales"))

# Transformation: Identify the best-selling item

best\_selling\_item\_df = total\_sales\_df.orderBy(F.desc("total\_sales")).limit(1)

# Transformation: Identify countries where customers bought the most-sold item

most\_sold\_item\_country\_df = input\_data\_frame.filter(

(F.col("StockCode") == best\_selling\_item\_df.select("StockCode").first()["StockCode"])

).groupBy("Country").agg(F.sum("Quantity").alias("total\_quantity"))

most\_sold\_item\_country\_df = most\_sold\_item\_country\_df.orderBy(F.desc("total\_quantity"))

return best\_selling\_item\_df, most\_sold\_item\_country\_df

# Use the transformation function

transformed\_data, most\_sold\_item\_country = MyTransformations(datasource0)

## Transformed data to Parquet format

glueContext.write\_dynamic\_frame.from\_catalog(frame=transformed\_data,

database="parquetdataresultdb",

table\_name="project\_dataset\_1\_csv",

transformation\_ctx="transformed\_data")

job.commit()

5. Query the data to identify the best-selling item and countries where customers have bought the most-sold item using Athena

**Athena Query:**

-- Calculate total sales per product

WITH product\_sales AS (

SELECT

StockCode,

Description,

SUM(Quantity) AS total\_quantity,

SUM(Quantity \* UnitPrice) AS total\_sales

FROM

project\_dataset\_1\_csv

GROUP BY

StockCode, Description

)

-- Identify the best-selling item

SELECT

StockCode AS best\_selling\_stock\_code,

Description AS best\_selling\_description,

total\_quantity AS total\_quantity\_sold,

total\_sales AS total\_sales\_amount

FROM

product\_sales

ORDER BY

total\_quantity DESC

LIMIT 1;

-- Identify the countries where customers have bought the most-sold item

WITH most\_sold\_item AS (

SELECT

StockCode,

Description,

Country,

SUM(Quantity) AS total\_quantity\_sold

FROM

project\_dataset\_1\_csv

WHERE

StockCode = (SELECT best\_selling\_stock\_code FROM product\_sales ORDER BY total\_quantity DESC LIMIT 1)

GROUP BY

StockCode, Description, Country

)

SELECT

StockCode AS most\_sold\_stock\_code,

Description AS most\_sold\_description,

Country,

total\_quantity\_sold

FROM

most\_sold\_item

ORDER BY

total\_quantity\_sold DESC;