**Stock Market Ticker Stream Cleanup**

**Script**

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 awsgluedq.transforms import EvaluateDataQuality

from awsglue.dynamicframe import DynamicFrame

import re

from pyspark.sql import functions as SqlFuncs

def sparkAggregate(glueContext, parentFrame, groups, aggs, transformation\_ctx) -> DynamicFrame:

aggsFuncs = []

for column, func in aggs:

aggsFuncs.append(getattr(SqlFuncs, func)(column))

result = parentFrame.toDF().groupBy(\*groups).agg(\*aggsFuncs) if len(groups) > 0 else parentFrame.toDF().agg(\*aggsFuncs)

return DynamicFrame.fromDF(result, glueContext, transformation\_ctx)

args = getResolvedOptions(sys.argv, ['JOB\_NAME'])

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark\_session

job = Job(glueContext)

job.init(args['JOB\_NAME'], args)

# Default ruleset used by all target nodes with data quality enabled

DEFAULT\_DATA\_QUALITY\_RULESET = """

Rules = [

ColumnCount > 0

]

"""

# Script generated for node Amazon S3

AmazonS3\_node1749657292056 = glueContext.create\_dynamic\_frame.from\_catalog(database="project\_db", table\_name="rawinput", transformation\_ctx="AmazonS3\_node1749657292056")

# Script generated for node Filter

Filter\_node1749657489325 = Filter.apply(frame=AmazonS3\_node1749657292056, f=lambda row: (row["price"] > 0), transformation\_ctx="Filter\_node1749657489325")

# Script generated for node Aggregate

Aggregate\_node1749657678000 = sparkAggregate(glueContext, parentFrame = Filter\_node1749657489325, groups = [], aggs = [["price", "avg"]], transformation\_ctx = "Aggregate\_node1749657678000")

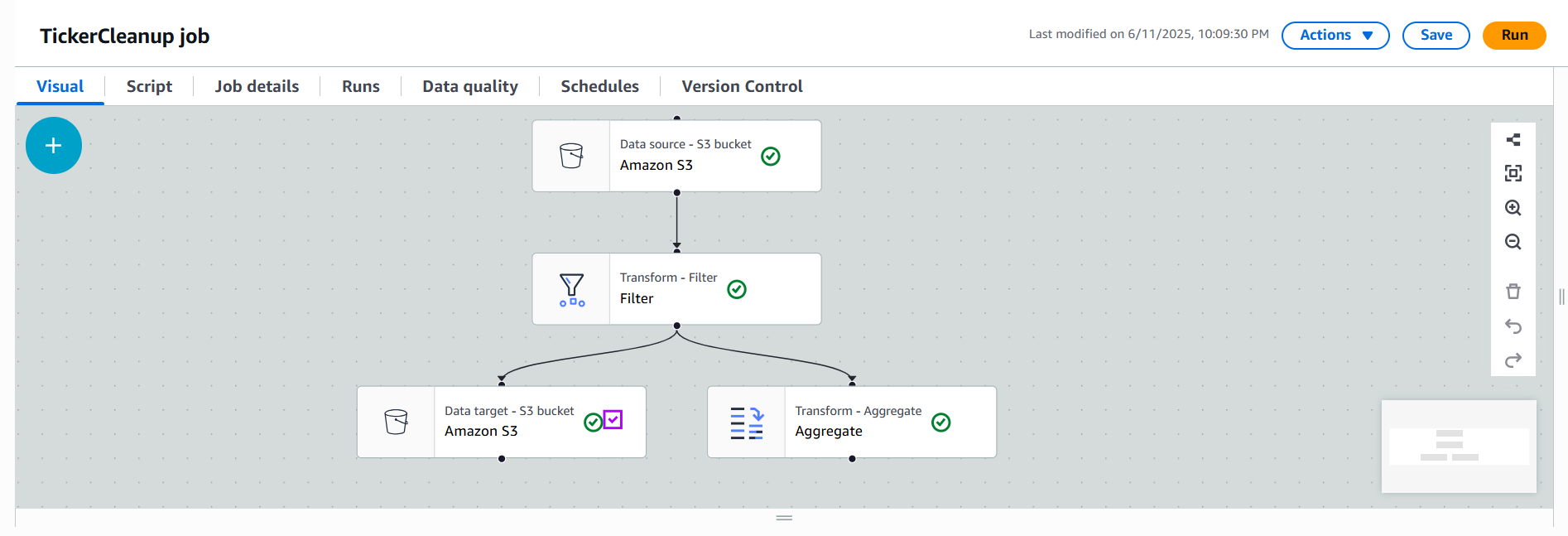
# Script generated for node Amazon S3

EvaluateDataQuality().process\_rows(frame=Filter\_node1749657489325, ruleset=DEFAULT\_DATA\_QUALITY\_RULESET, publishing\_options={"dataQualityEvaluationContext": "EvaluateDataQuality\_node1749657285797", "enableDataQualityResultsPublishing": True}, additional\_options={"dataQualityResultsPublishing.strategy": "BEST\_EFFORT", "observations.scope": "ALL"})

AmazonS3\_node1749657788503 = glueContext.write\_dynamic\_frame.from\_options(frame=Filter\_node1749657489325, connection\_type="s3", format="glueparquet", connection\_options={"path": "s3://glueprojects3/Output/", "partitionKeys": []}, format\_options={"compression": "snappy"}, transformation\_ctx="AmazonS3\_node1749657788503")

job.commit()

**Visual Structure**

****