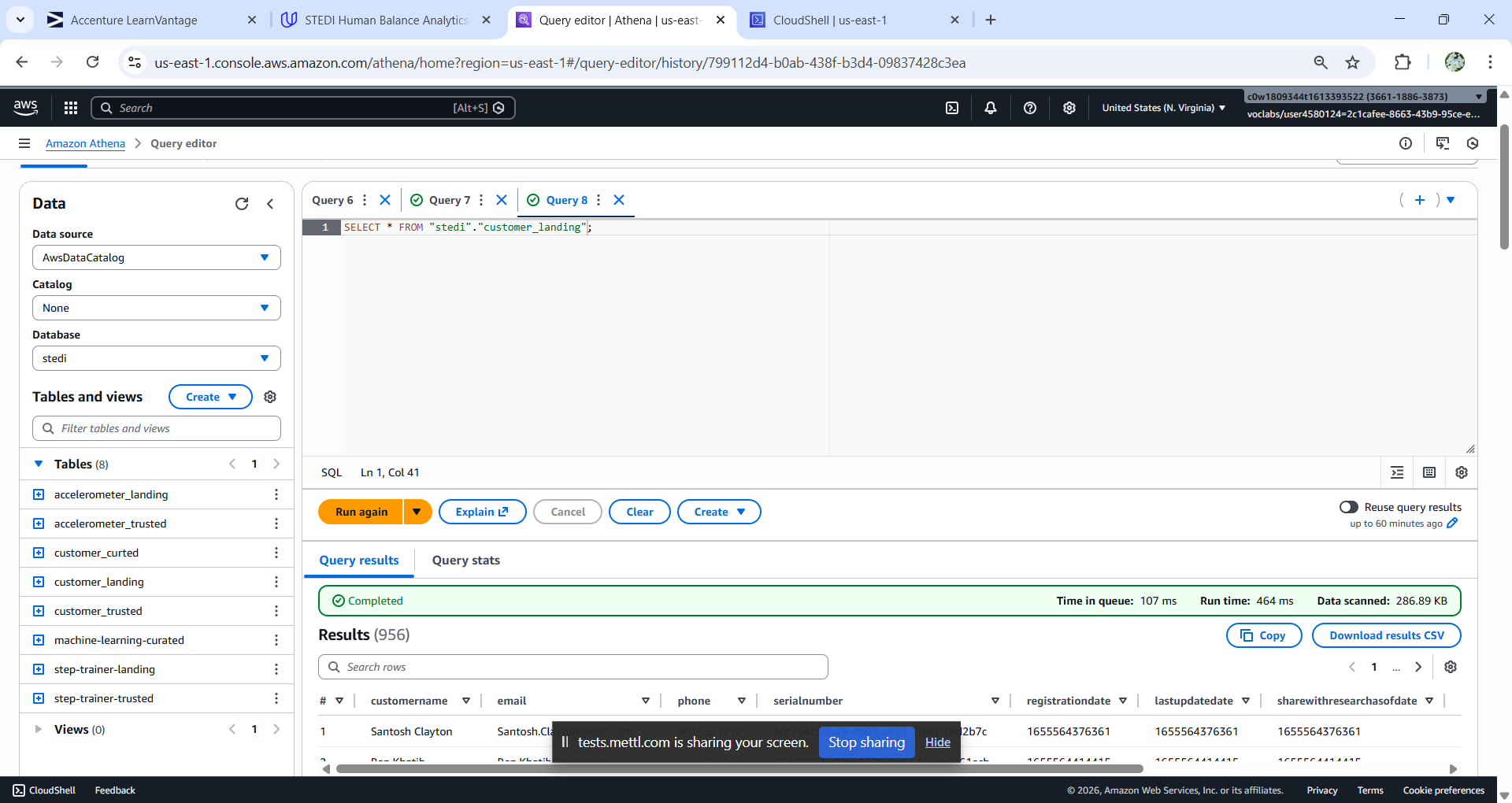
**STEDI HUMAN BALANCE ANALYTICS PROJECT REPORT**

Bucket name:teja-bucket-01

Tables in the S3 bucket, Database : stedi

Total 7 tables:

Customer Landing Table:



SQL DDL Script:

CREATE EXTERNAL TABLE `customer\_landing`(

`customername` string COMMENT 'from deserializer',

`email` string COMMENT 'from deserializer',

`phone` string COMMENT 'from deserializer',

`serialnumber` string COMMENT 'from deserializer',

`registrationdate` bigint COMMENT 'from deserializer',

`lastupdatedate` bigint COMMENT 'from deserializer',

`sharewithresearchasofdate` bigint COMMENT 'from deserializer',

`sharewithpublicasofdate` bigint COMMENT 'from deserializer',

`sharewithfriendsasofdate` bigint COMMENT 'from deserializer',

`birthday` string COMMENT 'from deserializer')

ROW FORMAT SERDE

'org.openx.data.jsonserde.JsonSerDe'

WITH SERDEPROPERTIES (

'case.insensitive'='TRUE',

'dots.in.keys'='FALSE',

'ignore.malformed.json'='FALSE',

'mapping'='TRUE')

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

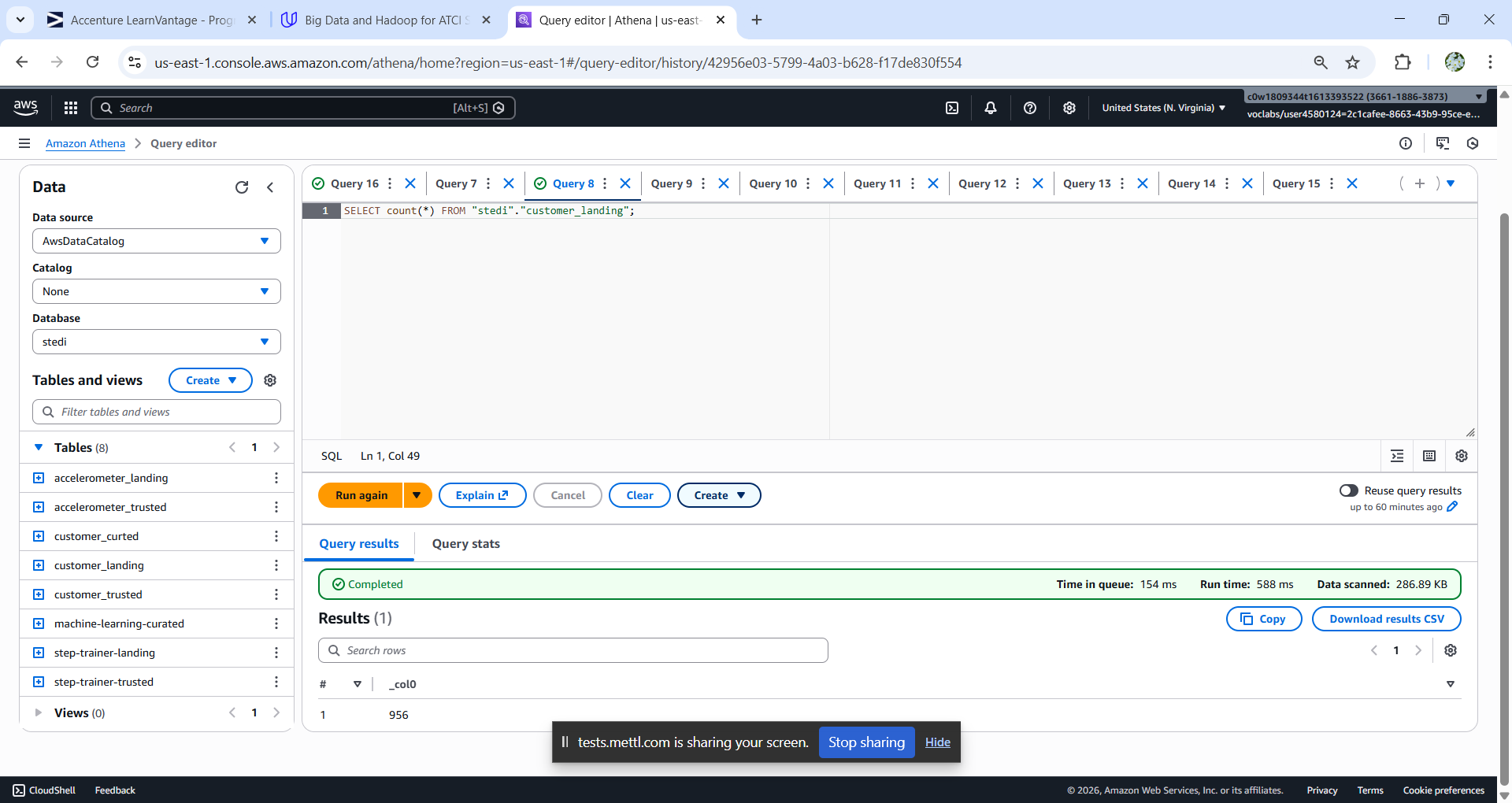
's3://teja-bucket-01/customer/landing'

TBLPROPERTIES (

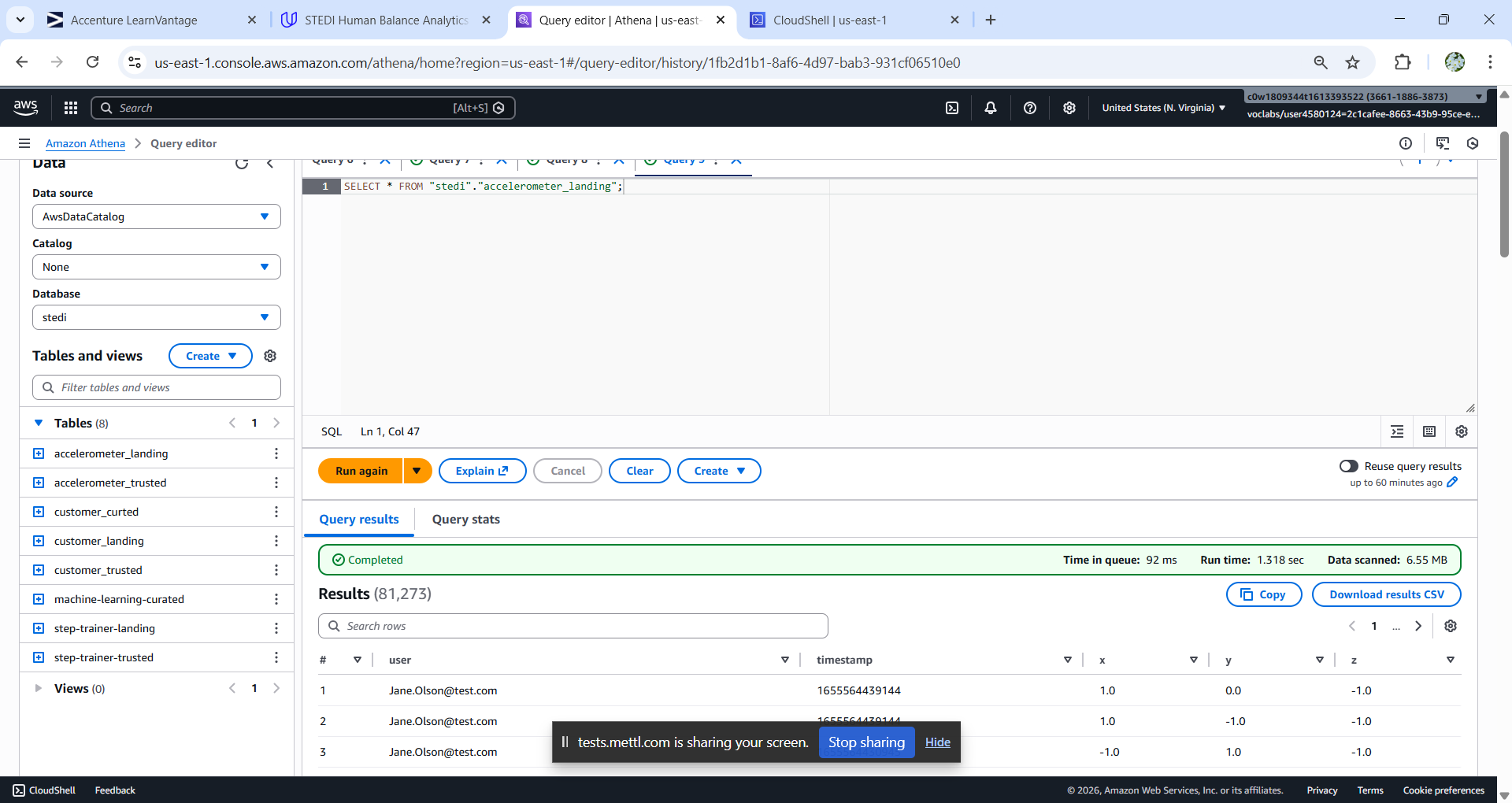
'classification'='json',

'transient\_lastDdlTime'='1768627255')

Count of rows in customer landing table:



Accelerometer Landing Table :



SQL DDL Script:

CREATE EXTERNAL TABLE `accelerometer\_landing`(

`user` string COMMENT 'from deserializer',

`timestamp` bigint COMMENT 'from deserializer',

`x` float COMMENT 'from deserializer',

`y` float COMMENT 'from deserializer',

`z` float COMMENT 'from deserializer')

ROW FORMAT SERDE

'org.openx.data.jsonserde.JsonSerDe'

WITH SERDEPROPERTIES (

'case.insensitive'='TRUE',

'dots.in.keys'='FALSE',

'ignore.malformed.json'='FALSE',

'mapping'='TRUE')

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

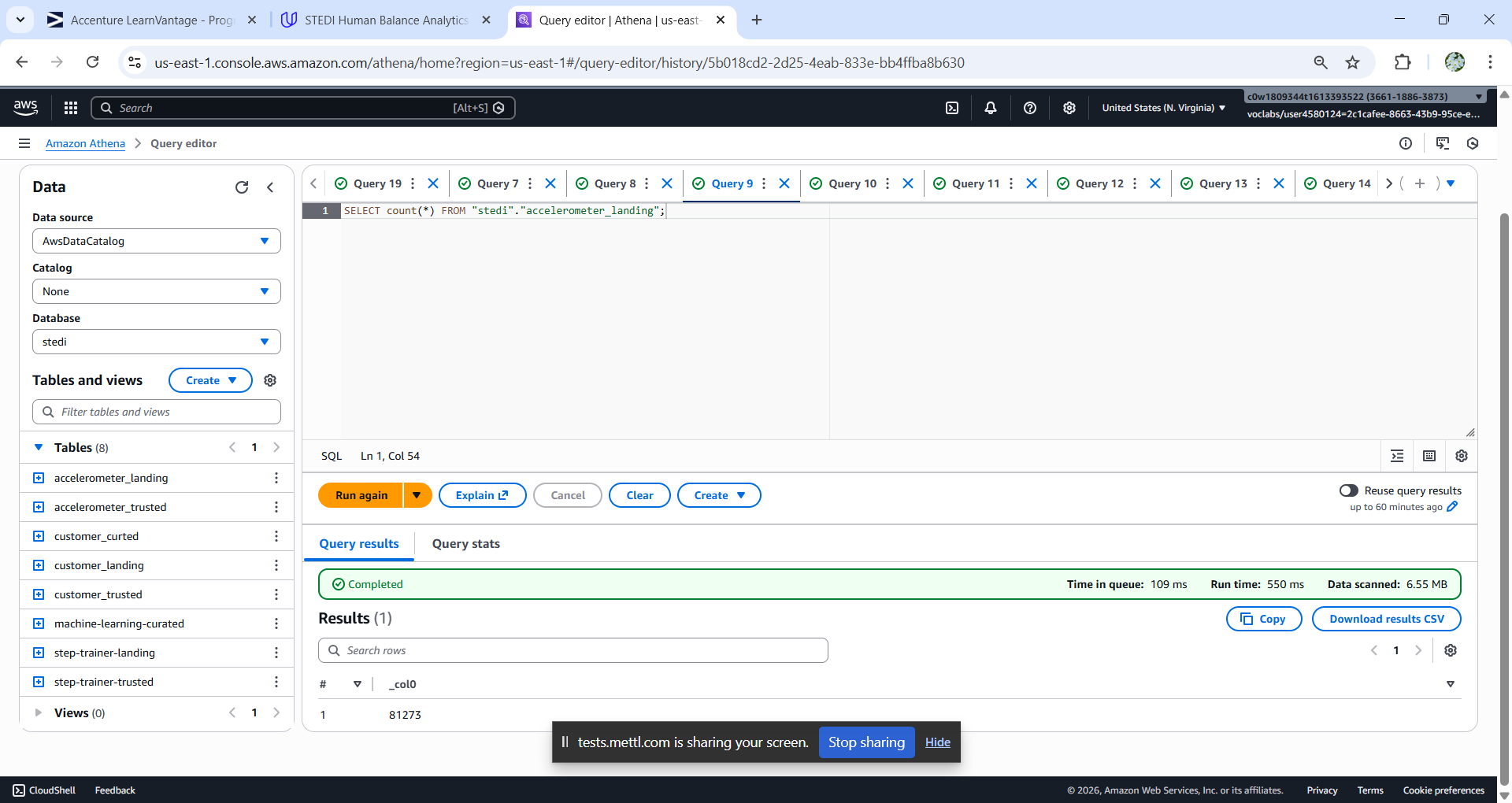
's3://teja-bucket-01/accelerometer/landing'

TBLPROPERTIES (

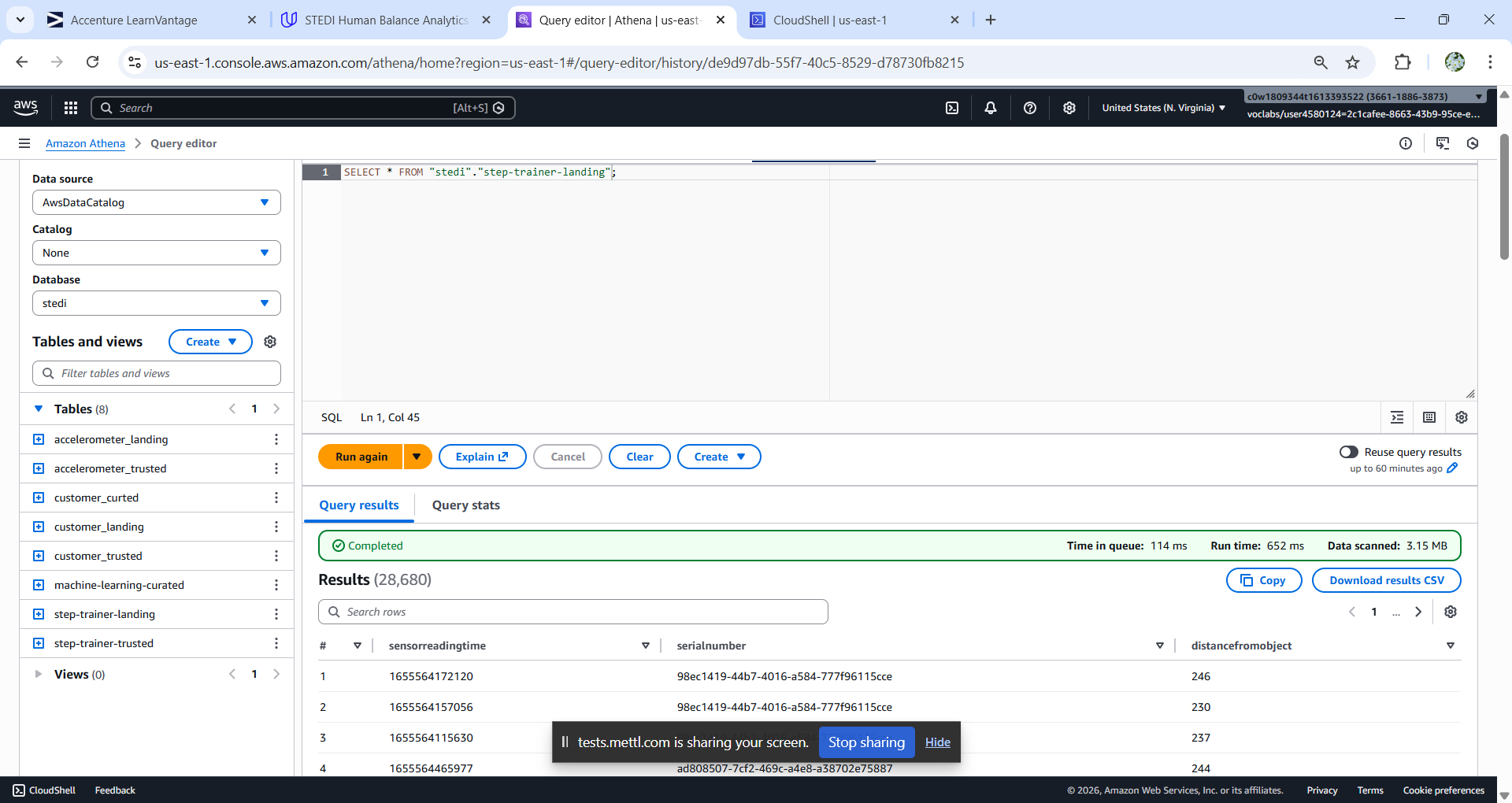
'classification'='json',

'transient\_lastDdlTime'='1768629270')

Count of rows in accelerometer landing table:



Step Trainer Landing Table:



SQL DDL Script:

CREATE EXTERNAL TABLE `step-trainer-landing`(

`sensorreadingtime` bigint COMMENT 'from deserializer',

`serialnumber` string COMMENT 'from deserializer',

`distancefromobject` bigint COMMENT 'from deserializer')

ROW FORMAT SERDE

'org.openx.data.jsonserde.JsonSerDe'

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

's3://teja-bucket-01/step\_trainer/landing/'

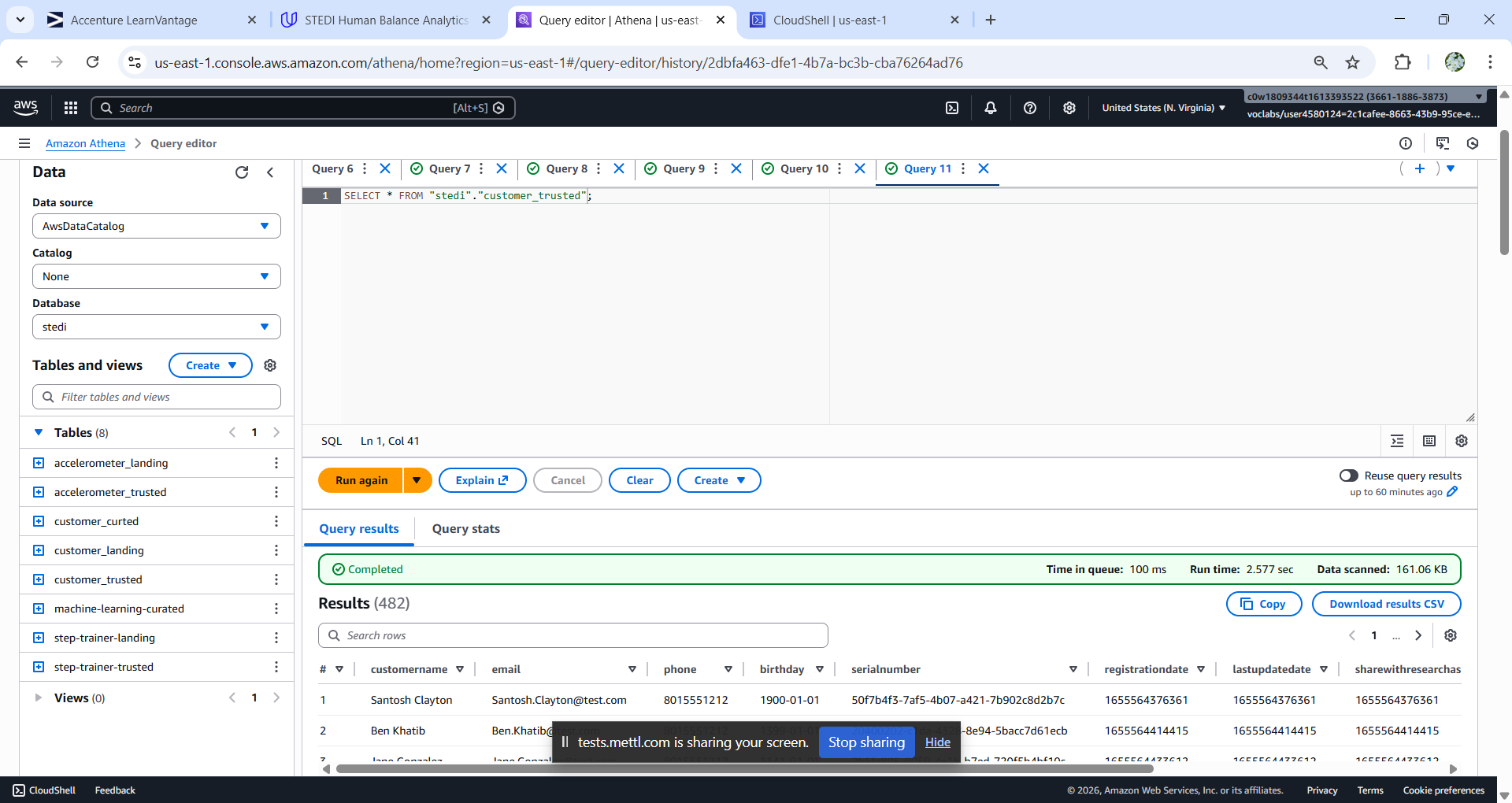
TBLPROPERTIES (

'classification'='json')

Count of row in step trainer landing table:



Customer Trusted Table:



SQL DDL Script:

CREATE EXTERNAL TABLE `customer\_trusted`(

`customername` string COMMENT 'from deserializer',

`email` string COMMENT 'from deserializer',

`phone` string COMMENT 'from deserializer',

`birthday` string COMMENT 'from deserializer',

`serialnumber` string COMMENT 'from deserializer',

`registrationdate` bigint COMMENT 'from deserializer',

`lastupdatedate` bigint COMMENT 'from deserializer',

`sharewithresearchasofdate` bigint COMMENT 'from deserializer',

`sharewithpublicasofdate` bigint COMMENT 'from deserializer',

`sharewithfriendsasofdate` bigint COMMENT 'from deserializer')

ROW FORMAT SERDE

'org.openx.data.jsonserde.JsonSerDe'

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION 's3://teja-bucket-01/customer/trusted/'

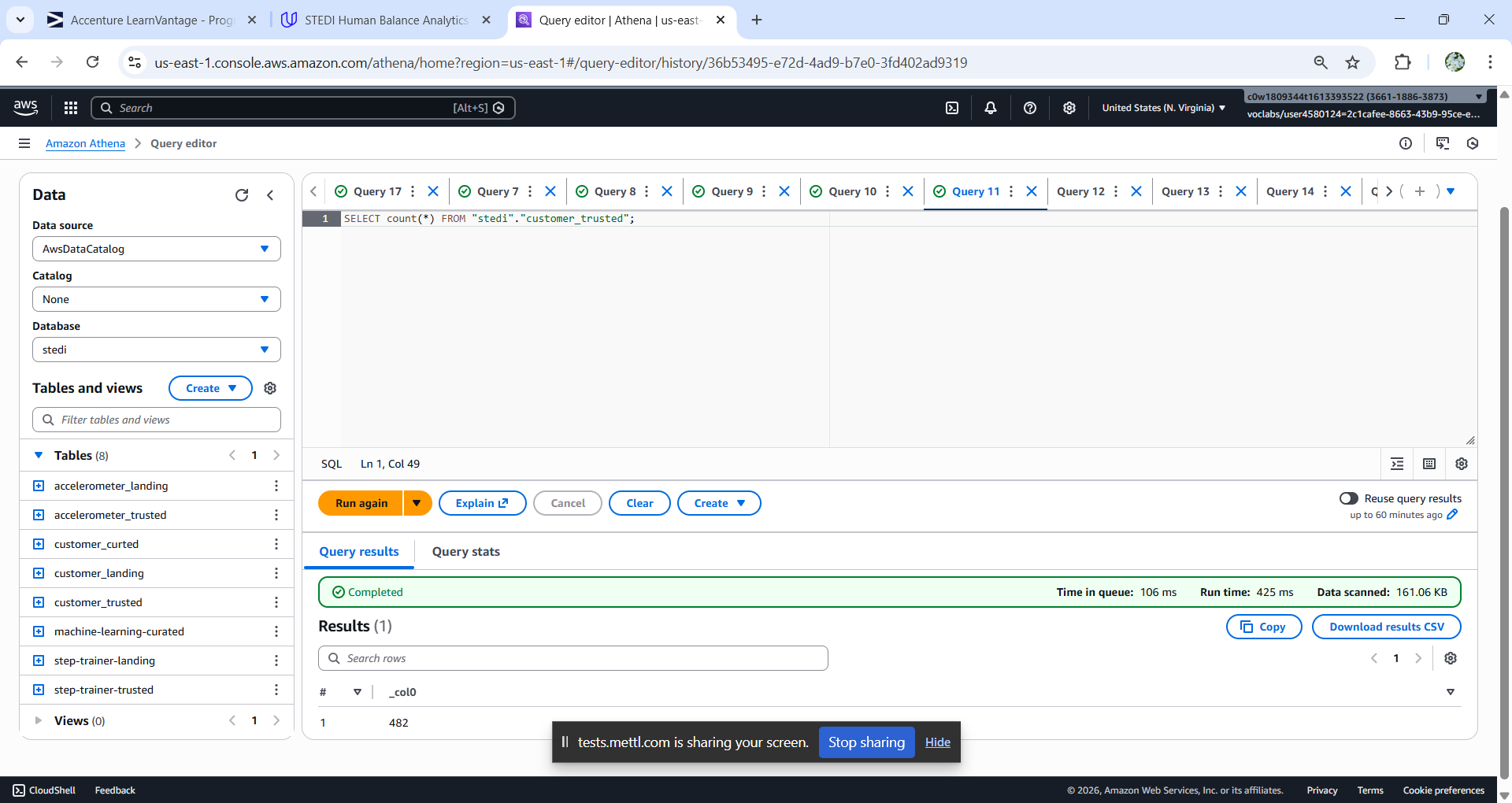
TBLPROPERTIES (

'CreatedByJob'='customer\_landing\_to\_trusted',

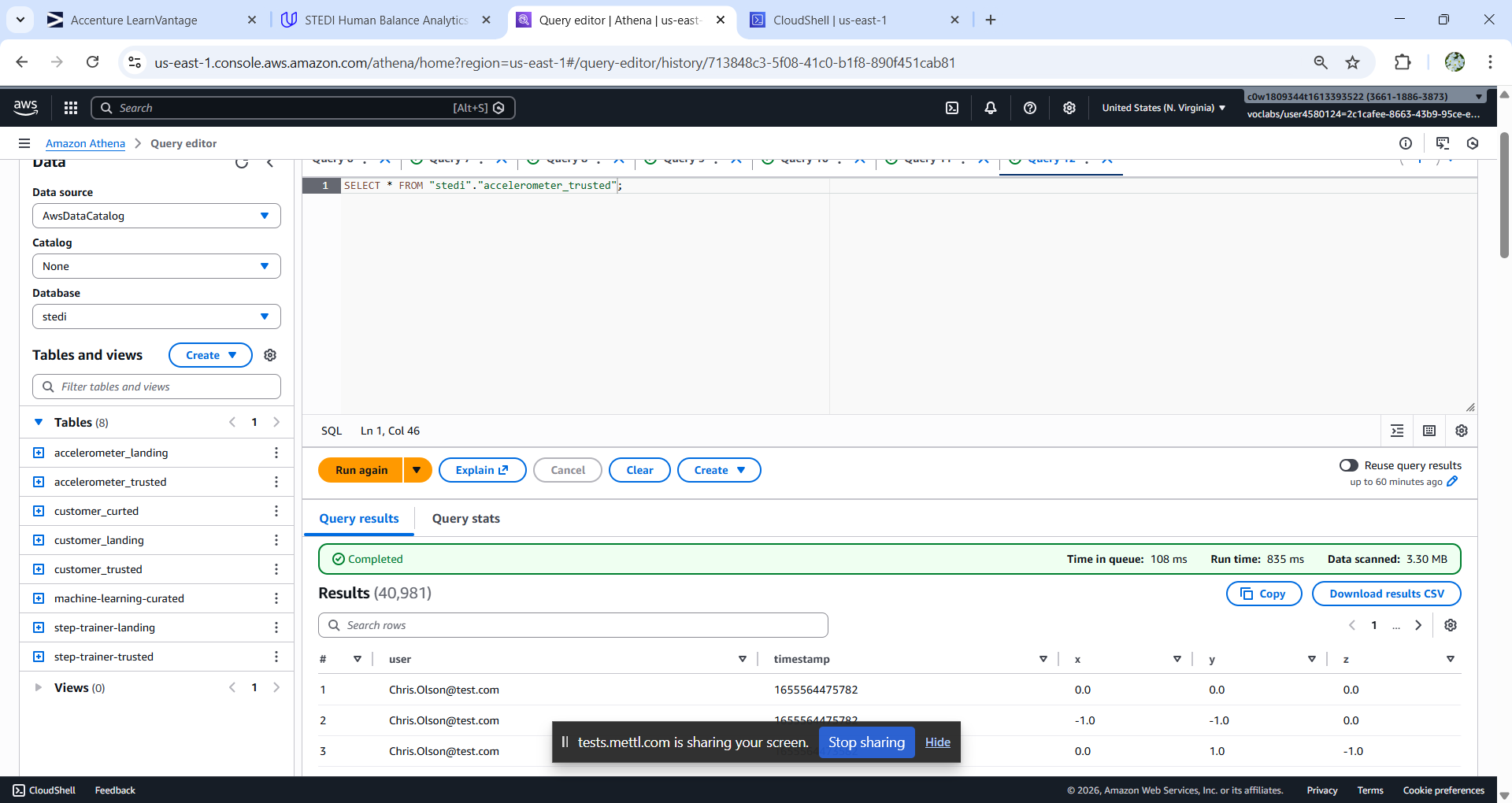
'CreatedByJobRun'='jr\_09e029be9d58b080c8fbb64870ca44d83dcc366b453df2de897c10be74d43548',

'classification'='json')

Count of rows in customer trusted table:



Accelerometer Trusted Table:



SQL DDL Script:

CREATE EXTERNAL TABLE `accelerometer\_trusted`(

`user` string COMMENT 'from deserializer',

`timestamp` bigint COMMENT 'from deserializer',

`x` double COMMENT 'from deserializer',

`y` double COMMENT 'from deserializer',

`z` double COMMENT 'from deserializer')

ROW FORMAT SERDE

'org.openx.data.jsonserde.JsonSerDe'

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

's3://teja-bucket-01/accelerometer/trusted/'

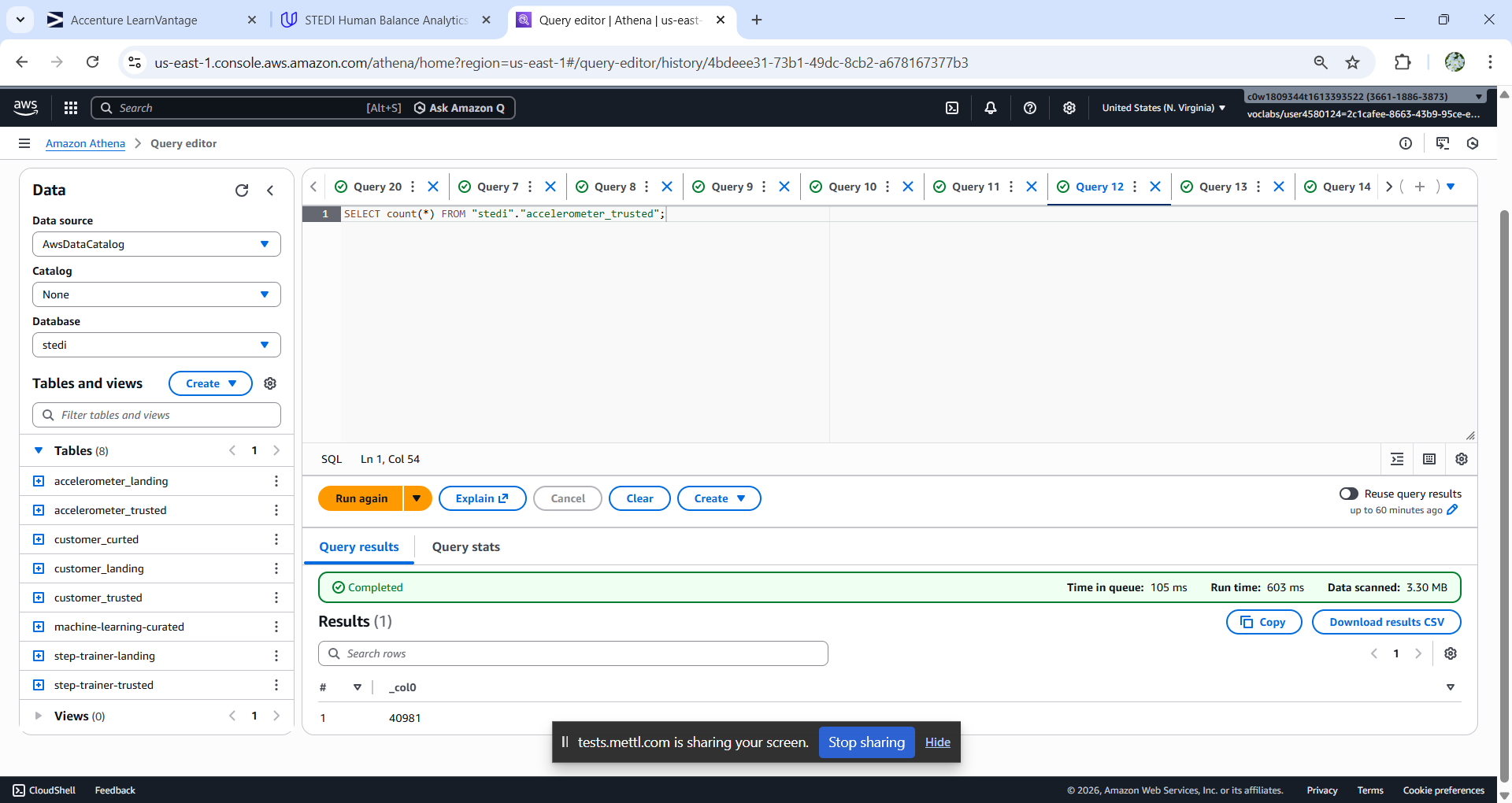
TBLPROPERTIES (

'CreatedByJob'='accelerometer\_landing\_to\_trusted',

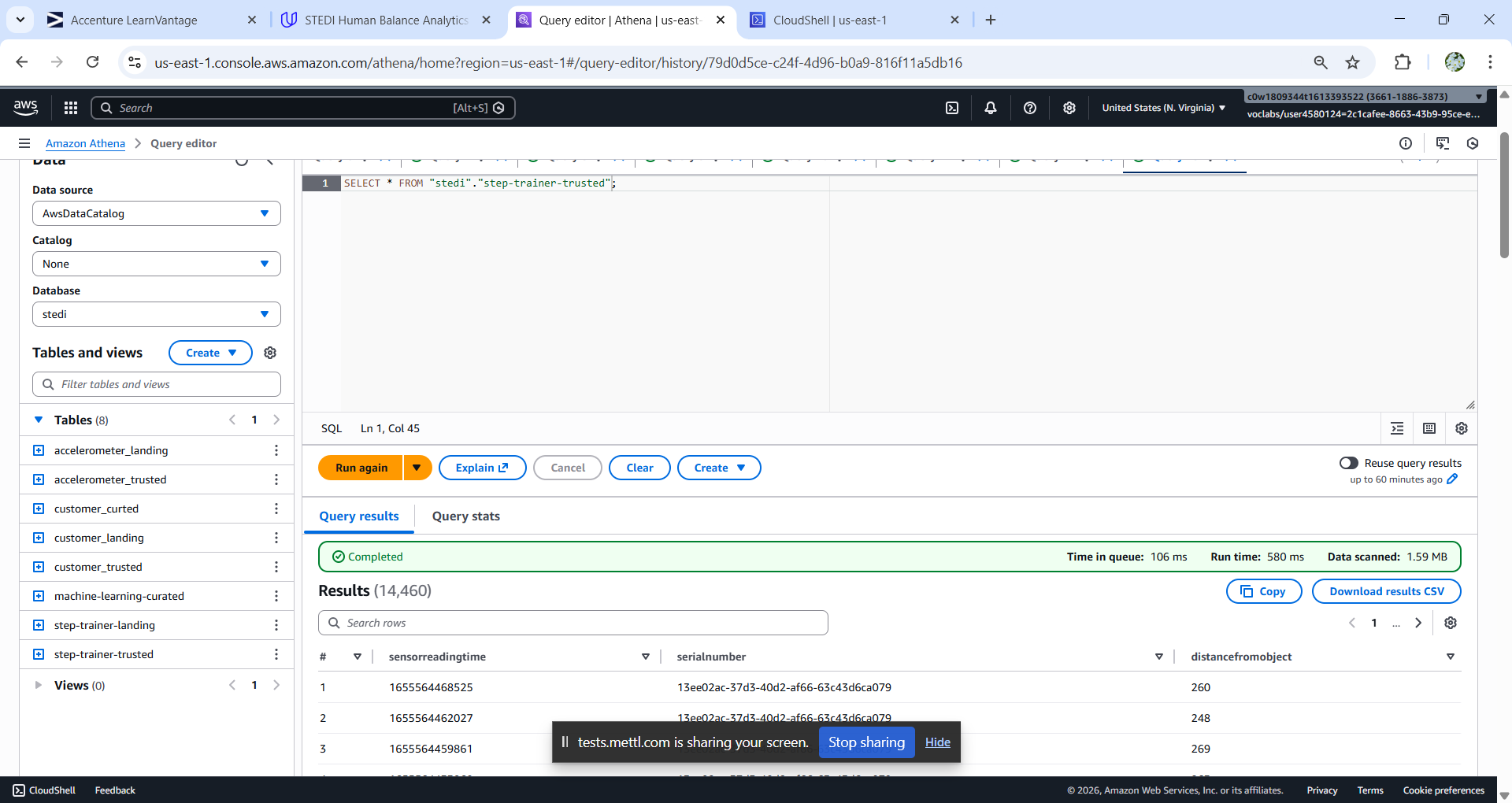
'CreatedByJobRun'='jr\_8802190122dca2949d08fecfc8efb310a3984892b1163d6dddfb6652297f8ec2',

'classification'='json')

Count of rows in Accelerometer trusted table:



Step Trainer Trusted Table:



SQL DDL Script:

CREATE EXTERNAL TABLE `step-trainer-trusted`(

`sensorreadingtime` bigint COMMENT 'from deserializer',

`serialnumber` string COMMENT 'from deserializer',

`distancefromobject` int COMMENT 'from deserializer')

ROW FORMAT SERDE

'org.openx.data.jsonserde.JsonSerDe'

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

's3://teja-bucket-01/step\_trainer/trusted/'

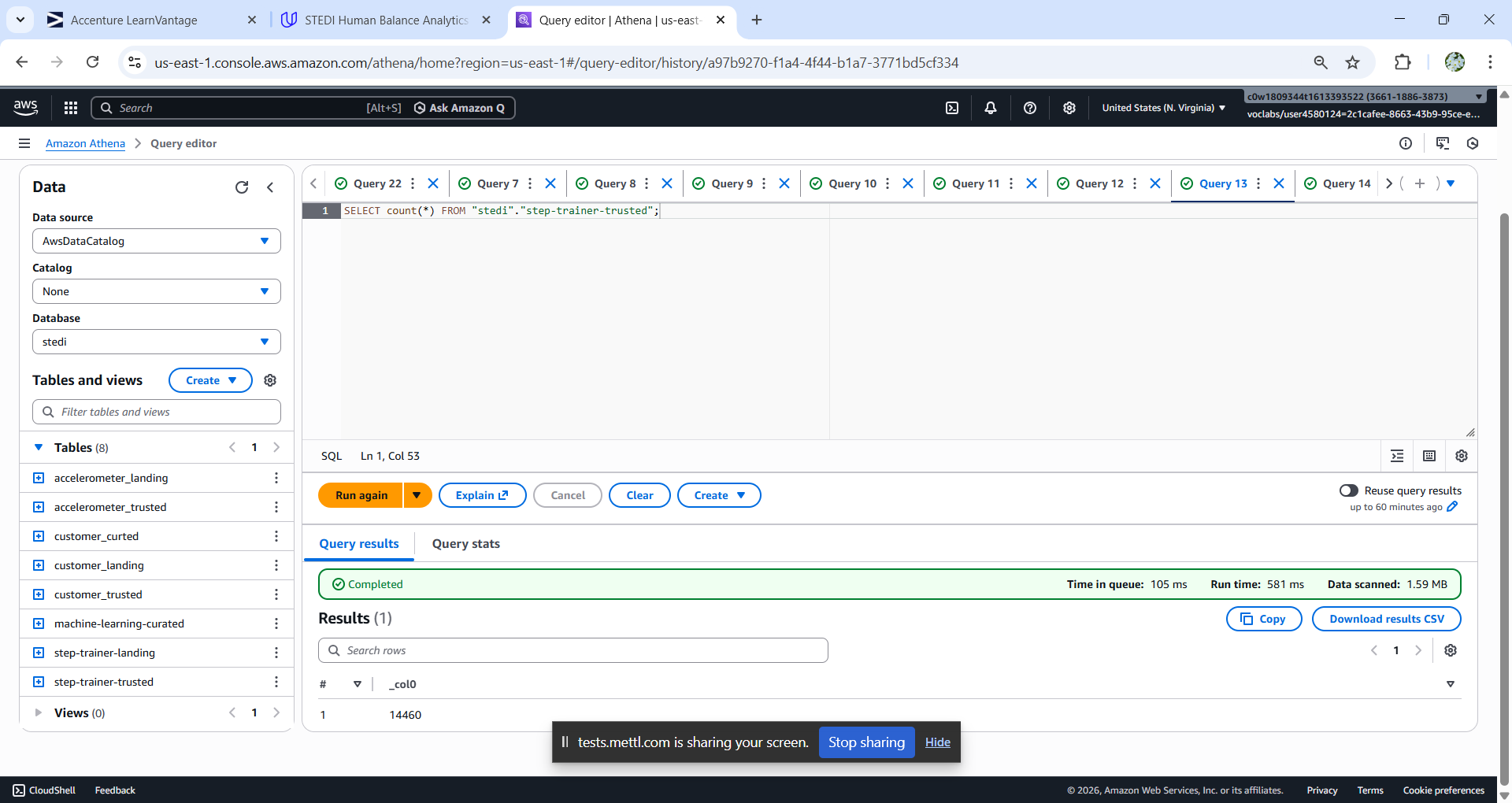
TBLPROPERTIES (

'CreatedByJob'='step-trainer-landing-to-trusted',

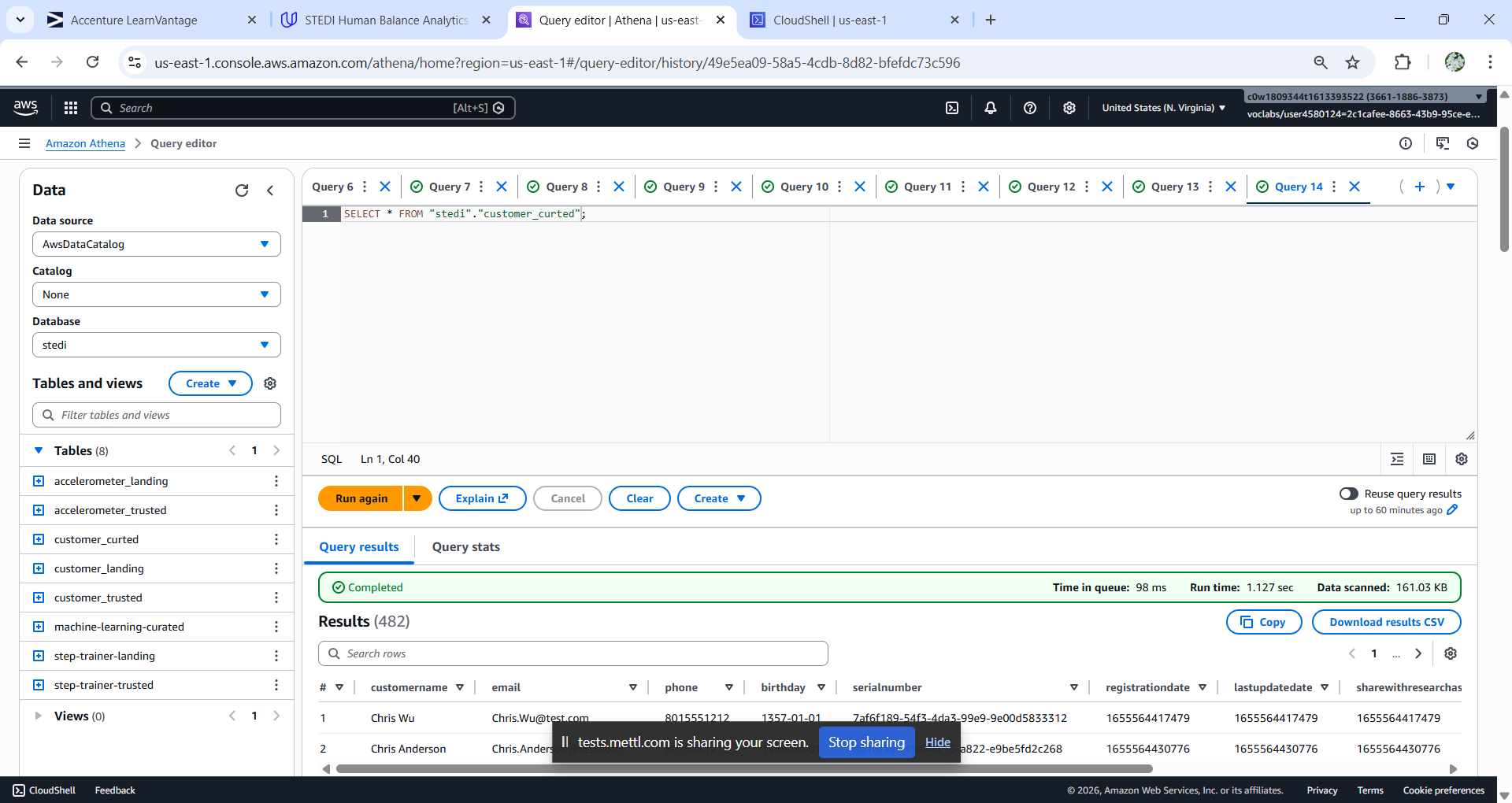
'CreatedByJobRun'='jr\_cadb50649acc8db5666cddc57af511f74f5d47310fcaa3e5dac7ae155bf3676a',

'classification'='json')

Count of rows in step trainer trusted table:



Customer Curated Table:



SQL DDL Script:

CREATE EXTERNAL TABLE `customer\_curted`(

`customername` string COMMENT 'from deserializer',

`email` string COMMENT 'from deserializer',

`phone` string COMMENT 'from deserializer',

`birthday` string COMMENT 'from deserializer',

`serialnumber` string COMMENT 'from deserializer',

`registrationdate` bigint COMMENT 'from deserializer',

`lastupdatedate` bigint COMMENT 'from deserializer',

`sharewithresearchasofdate` bigint COMMENT 'from deserializer',

`sharewithpublicasofdate` bigint COMMENT 'from deserializer',

`sharewithfriendsasofdate` bigint COMMENT 'from deserializer')

ROW FORMAT SERDE

'org.openx.data.jsonserde.JsonSerDe'

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

's3://teja-bucket-01/customer/curted/'

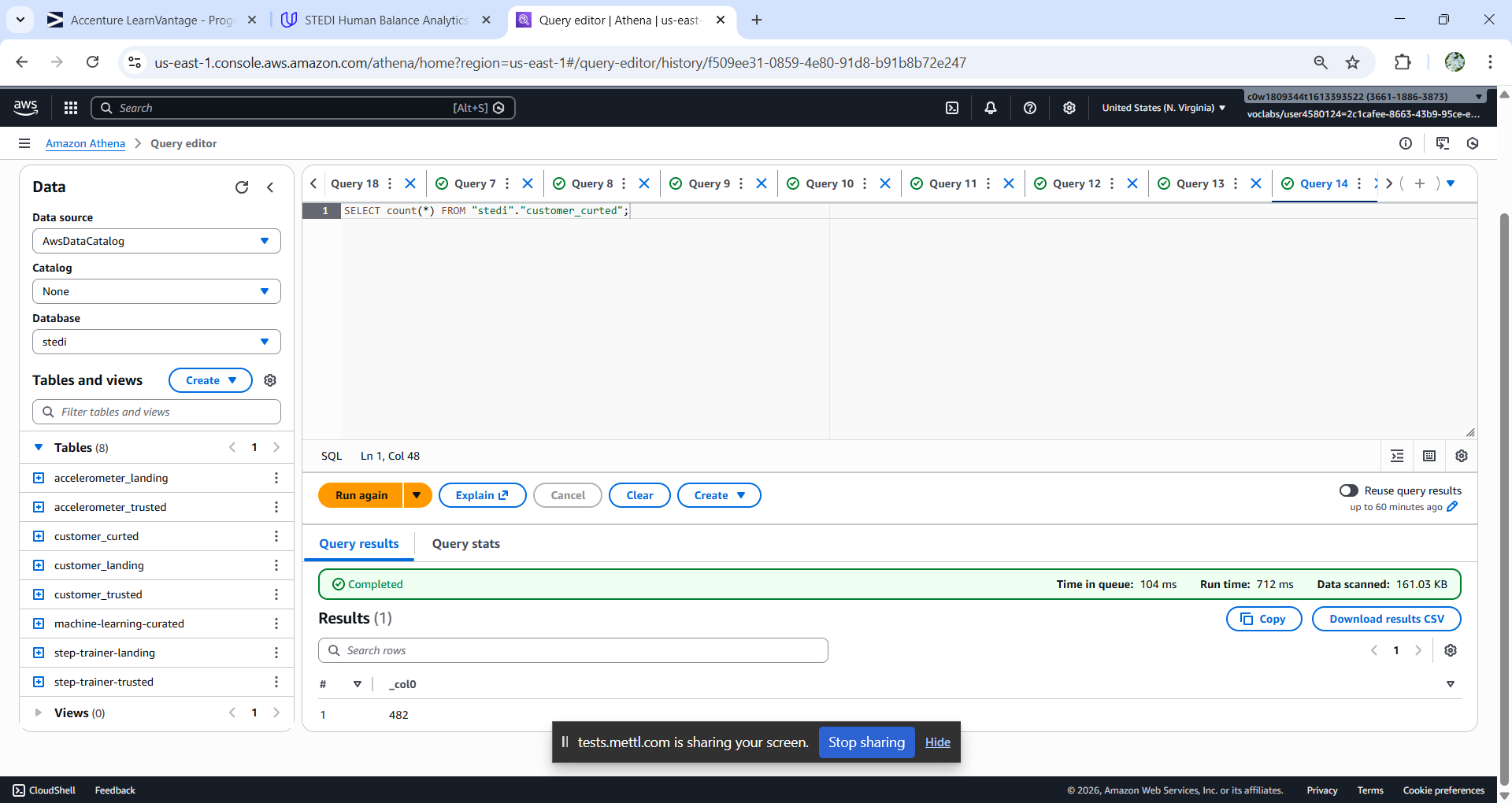
TBLPROPERTIES (

'CreatedByJob'='customer\_trusted\_to\_curted',

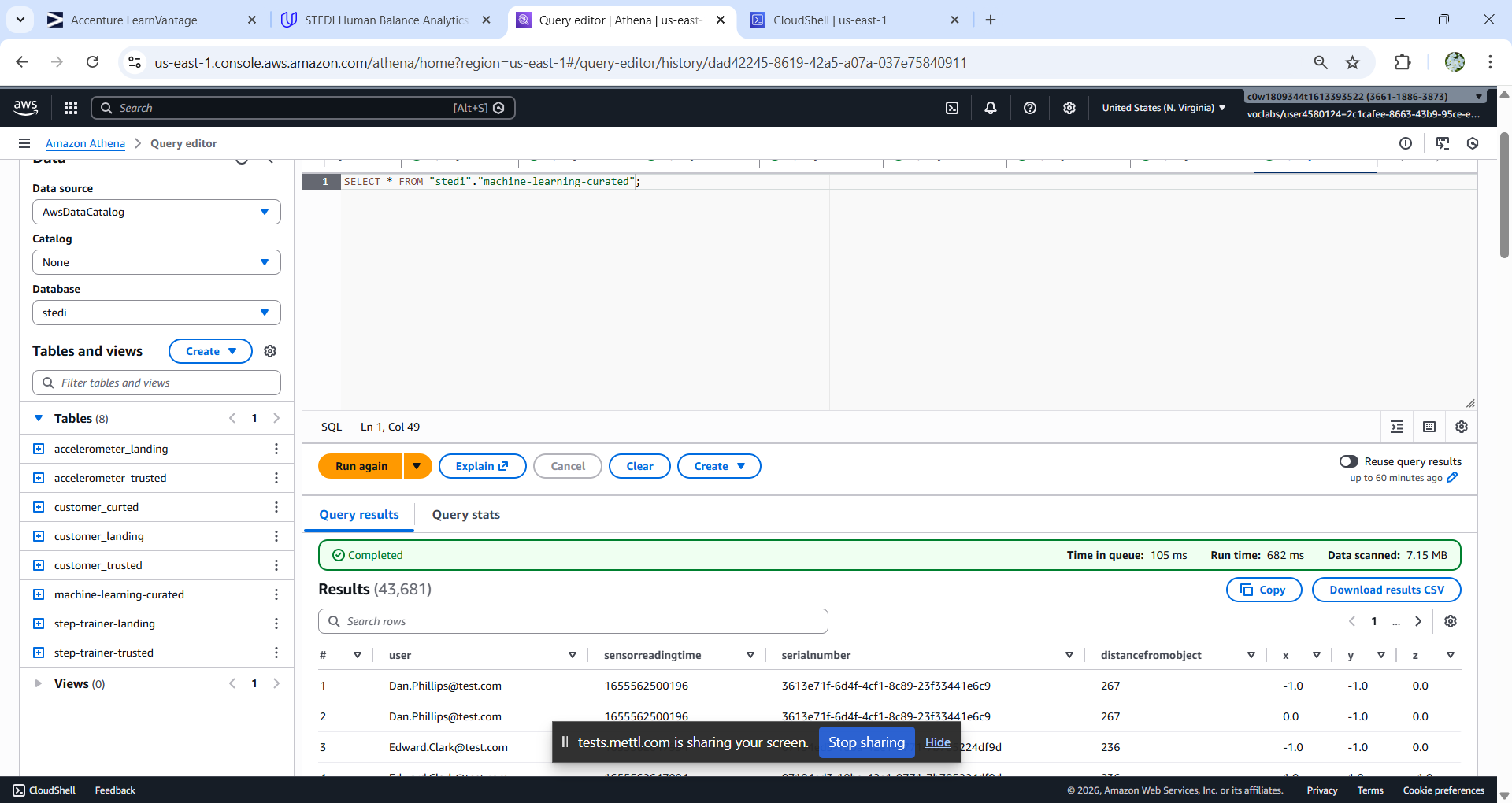
'CreatedByJobRun'='jr\_5b9f9666a2d3b31a84b10b7eaab3ee5889b8bc98ee3b7d698f3a3c12aec1ed99',

'classification'='json')

Count of rows in customer curated:



Machine Learning Table:



SQL DDL Script:

CREATE EXTERNAL TABLE `machine-learning-curated`(

`user` string COMMENT 'from deserializer',

`sensorreadingtime` bigint COMMENT 'from deserializer',

`serialnumber` string COMMENT 'from deserializer',

`distancefromobject` int COMMENT 'from deserializer',

`x` double COMMENT 'from deserializer',

`y` double COMMENT 'from deserializer',

`z` double COMMENT 'from deserializer')

ROW FORMAT SERDE

'org.openx.data.jsonserde.JsonSerDe'

STORED AS INPUTFORMAT

'org.apache.hadoop.mapred.TextInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION

's3://teja-bucket-01/machine-learning-curated/'

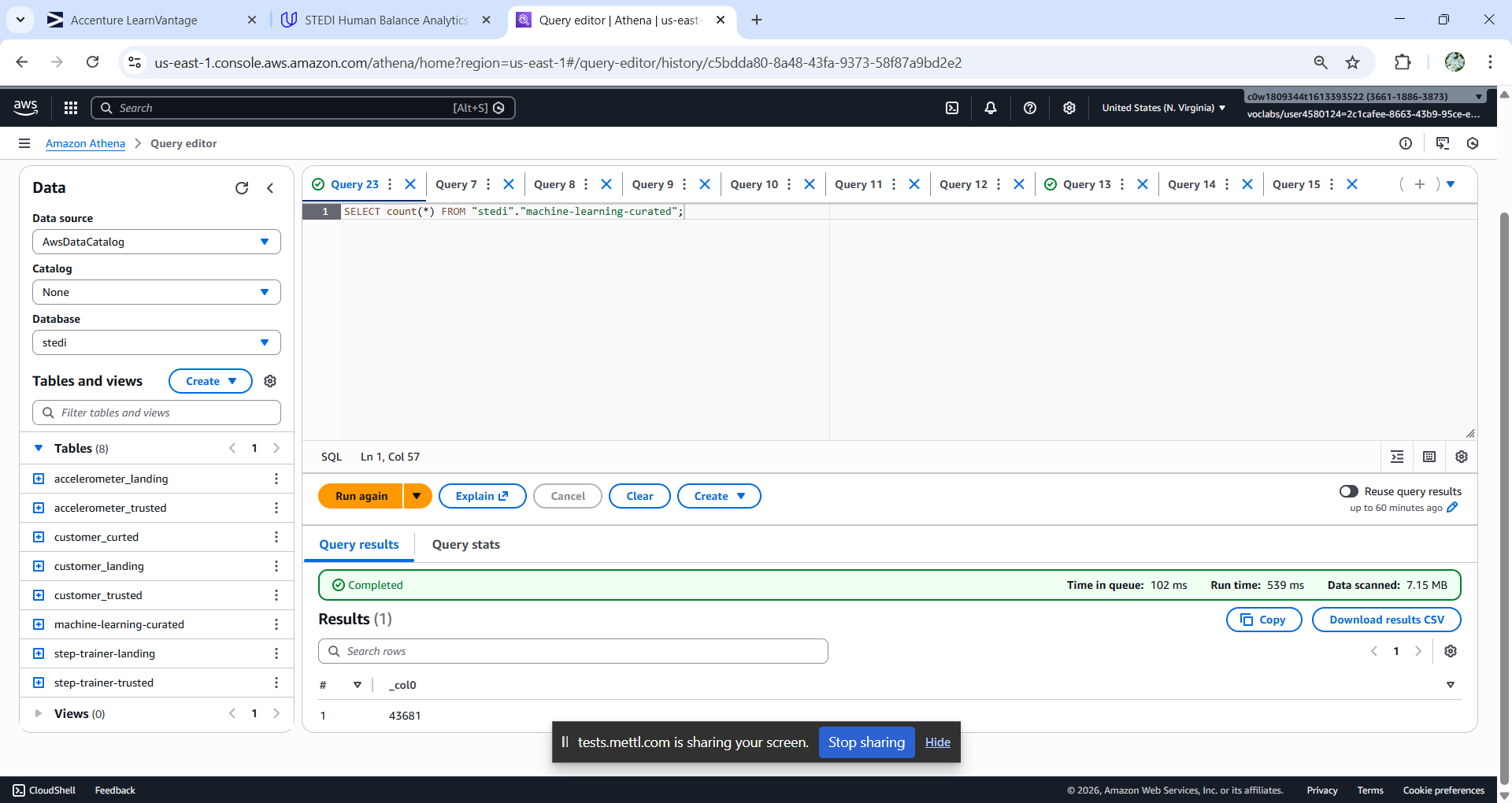
TBLPROPERTIES (

'CreatedByJob'='Machine-learning-curated',

'CreatedByJobRun'='jr\_f4016419cbcd5fda6283a96e32d3352a3e6d2c941e2fbd3d77ebc26a656a6b22',

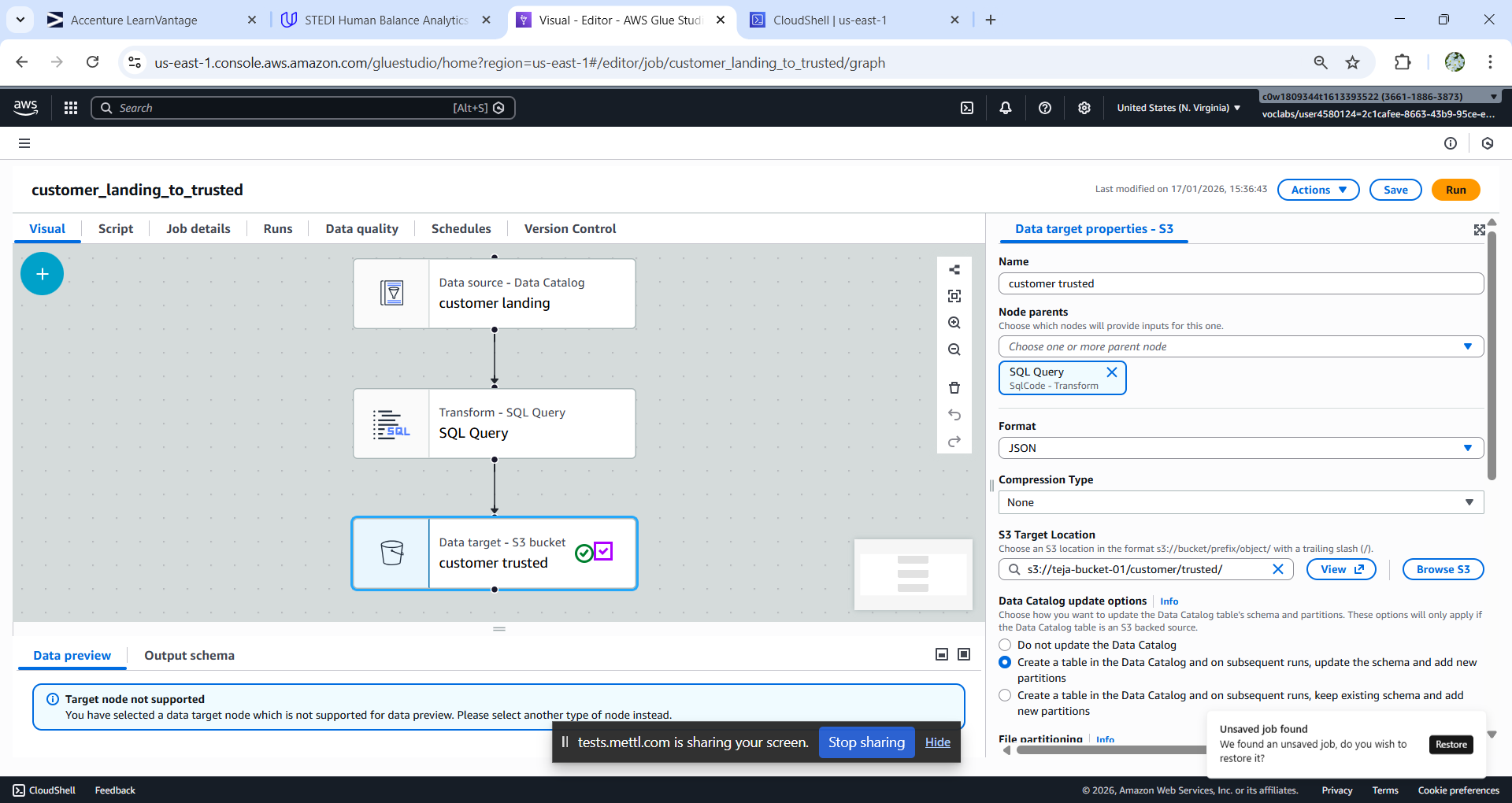
'classification'='json')

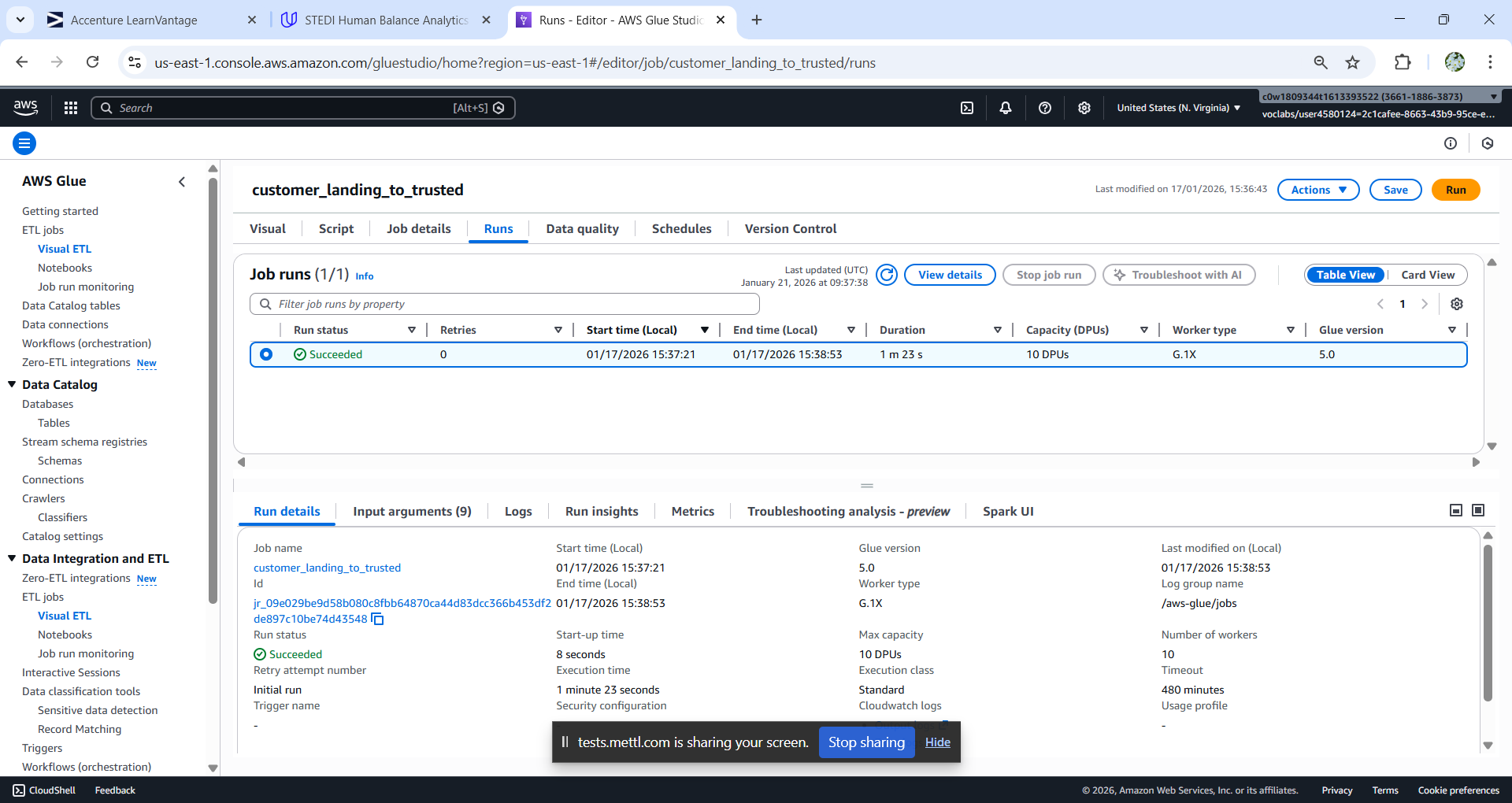
Count of rows in machine learning curated table:



**Glue jobs:** Total 5 glue jobs.

Customer landing to Trusted:





Python 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 import DynamicFrame

def sparkSqlQuery(glueContext, query, mapping, transformation\_ctx) -> DynamicFrame:

for alias, frame in mapping.items():

frame.toDF().createOrReplaceTempView(alias)

result = spark.sql(query)

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 customer landing

customerlanding\_node1768643608842 = glueContext.create\_dynamic\_frame.from\_catalog(database="stedi", table\_name="customer\_landing", transformation\_ctx="customerlanding\_node1768643608842")

# Script generated for node SQL Query

SqlQuery0 = '''

select \* from myDataSource where

shareWithResearchAsOfDate is not null;

'''

SQLQuery\_node1768643659655 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"myDataSource":customerlanding\_node1768643608842}, transformation\_ctx = "SQLQuery\_node1768643659655")

# Script generated for node customer trusted

EvaluateDataQuality().process\_rows(frame=SQLQuery\_node1768643659655, ruleset=DEFAULT\_DATA\_QUALITY\_RULESET, publishing\_options={"dataQualityEvaluationContext": "EvaluateDataQuality\_node1768643451766", "enableDataQualityResultsPublishing": True}, additional\_options={"dataQualityResultsPublishing.strategy": "BEST\_EFFORT", "observations.scope": "ALL"})

customertrusted\_node1768643693440 = glueContext.getSink(path="s3://teja-bucket-01/customer/trusted/", connection\_type="s3", updateBehavior="UPDATE\_IN\_DATABASE", partitionKeys=[], enableUpdateCatalog=True, transformation\_ctx="customertrusted\_node1768643693440")

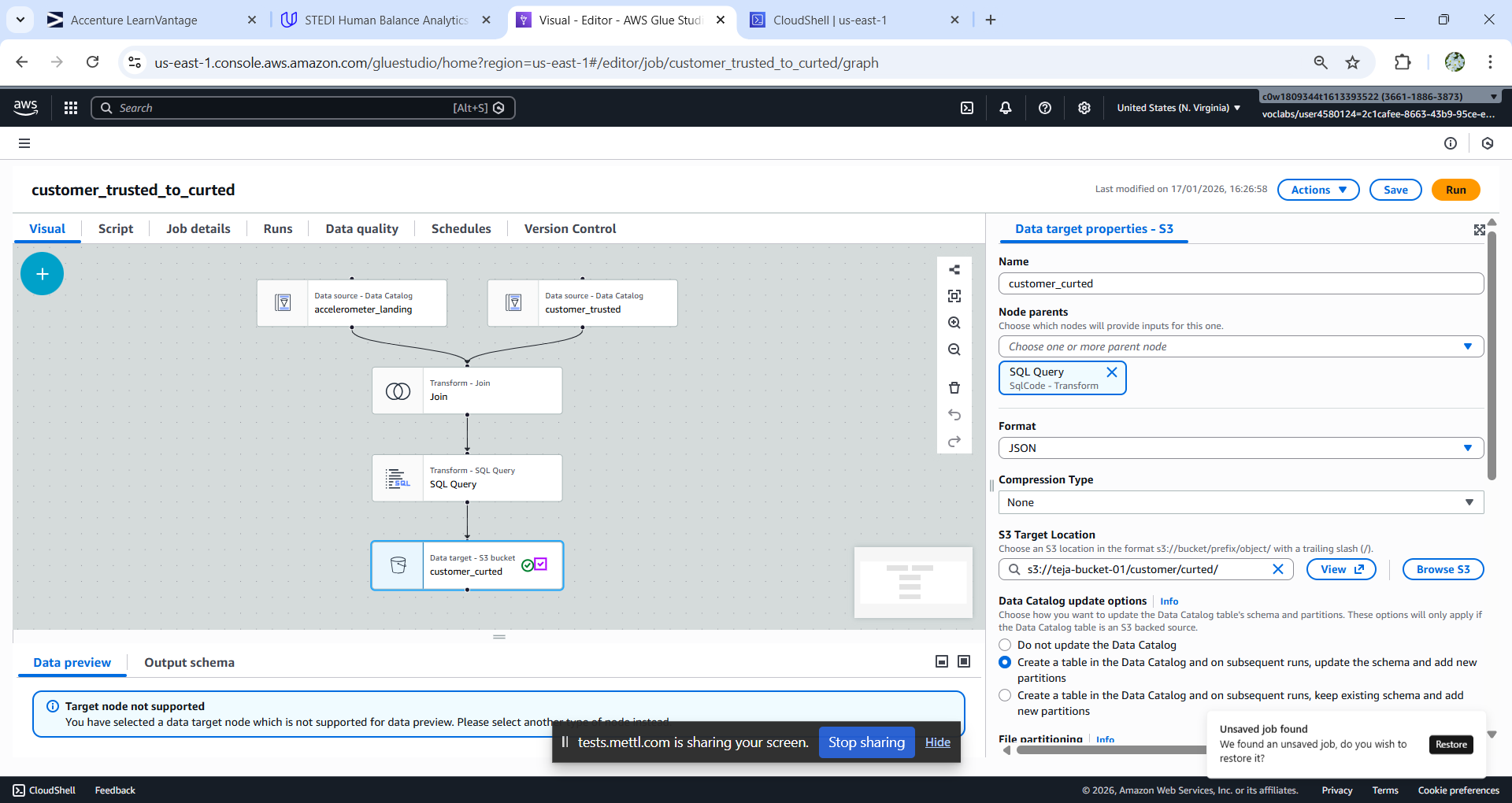
customertrusted\_node1768643693440.setCatalogInfo(catalogDatabase="stedi",catalogTableName="customer\_trusted")

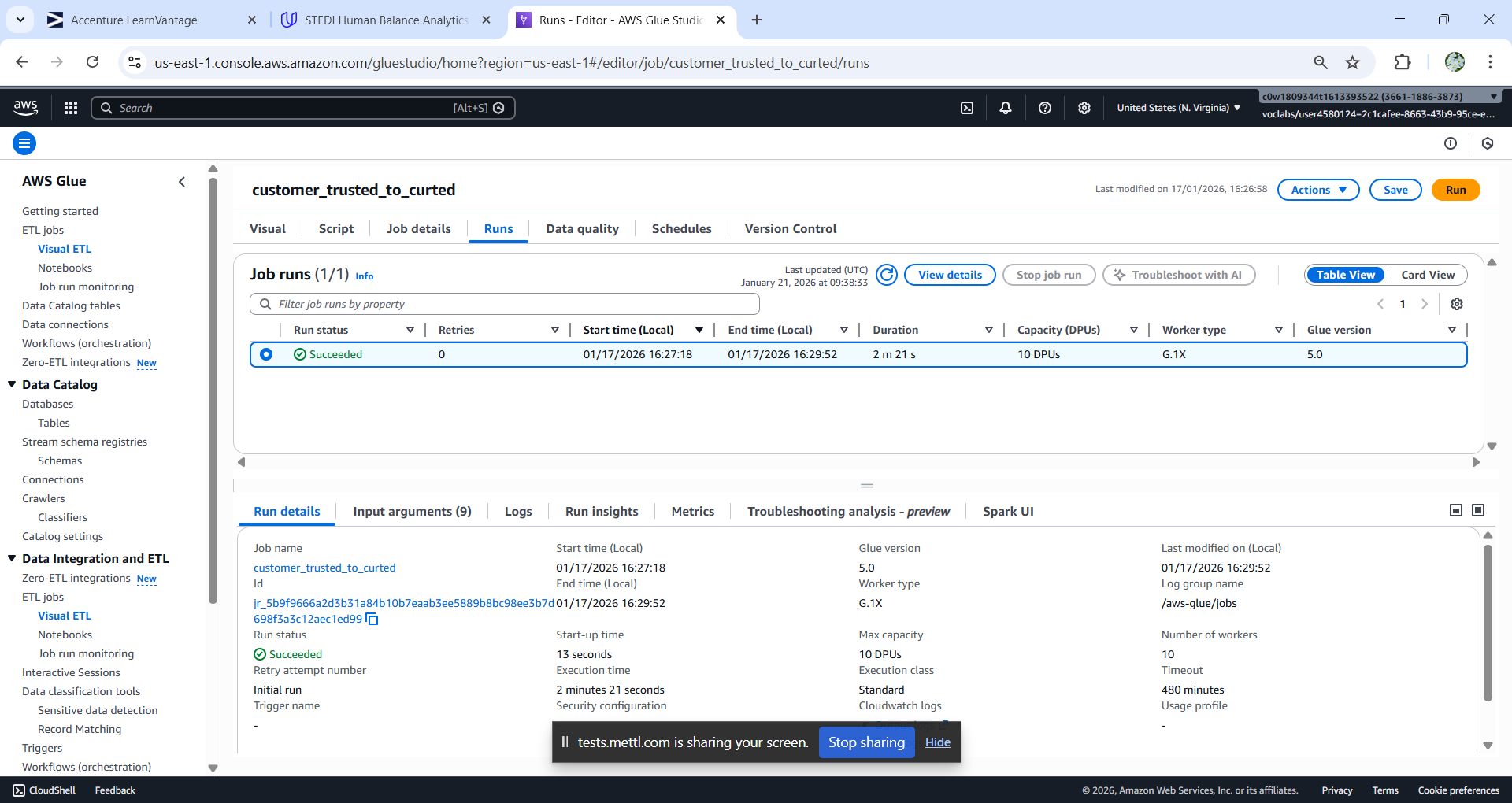
customertrusted\_node1768643693440.setFormat("json")

customertrusted\_node1768643693440.writeFrame(SQLQuery\_node1768643659655)

job.commit()

Customer trusted to curated:





Python 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 import DynamicFrame

def sparkSqlQuery(glueContext, query, mapping, transformation\_ctx) -> DynamicFrame:

for alias, frame in mapping.items():

frame.toDF().createOrReplaceTempView(alias)

result = spark.sql(query)

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 accelerometer\_landing

accelerometer\_landing\_node1768646651256 = glueContext.create\_dynamic\_frame.from\_catalog(database="stedi", table\_name="accelerometer\_landing", transformation\_ctx="accelerometer\_landing\_node1768646651256")

# Script generated for node customer\_trusted

customer\_trusted\_node1768646605490 = glueContext.create\_dynamic\_frame.from\_catalog(database="stedi", table\_name="customer\_trusted", transformation\_ctx="customer\_trusted\_node1768646605490")

# Script generated for node Join

Join\_node1768646703742 = Join.apply(frame1=accelerometer\_landing\_node1768646651256, frame2=customer\_trusted\_node1768646605490, keys1=["user"], keys2=["email"], transformation\_ctx="Join\_node1768646703742")

# Script generated for node SQL Query

SqlQuery0 = '''

select distinct customerName,email,phone,birthDay,serialNumber,registrationDate,lastUpdateDate,shareWithResearchAsOfDate,shareWithPublicAsOfDate,shareWithFriendsAsOfDate from myDataSource;

'''

SQLQuery\_node1768646753871 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"myDataSource":Join\_node1768646703742}, transformation\_ctx = "SQLQuery\_node1768646753871")

# Script generated for node customer\_curted

EvaluateDataQuality().process\_rows(frame=SQLQuery\_node1768646753871, ruleset=DEFAULT\_DATA\_QUALITY\_RULESET, publishing\_options={"dataQualityEvaluationContext": "EvaluateDataQuality\_node1768646591062", "enableDataQualityResultsPublishing": True}, additional\_options={"dataQualityResultsPublishing.strategy": "BEST\_EFFORT", "observations.scope": "ALL"})

customer\_curted\_node1768647254241 = glueContext.getSink(path="s3://teja-bucket-01/customer/curted/", connection\_type="s3", updateBehavior="UPDATE\_IN\_DATABASE", partitionKeys=[], enableUpdateCatalog=True, transformation\_ctx="customer\_curted\_node1768647254241")

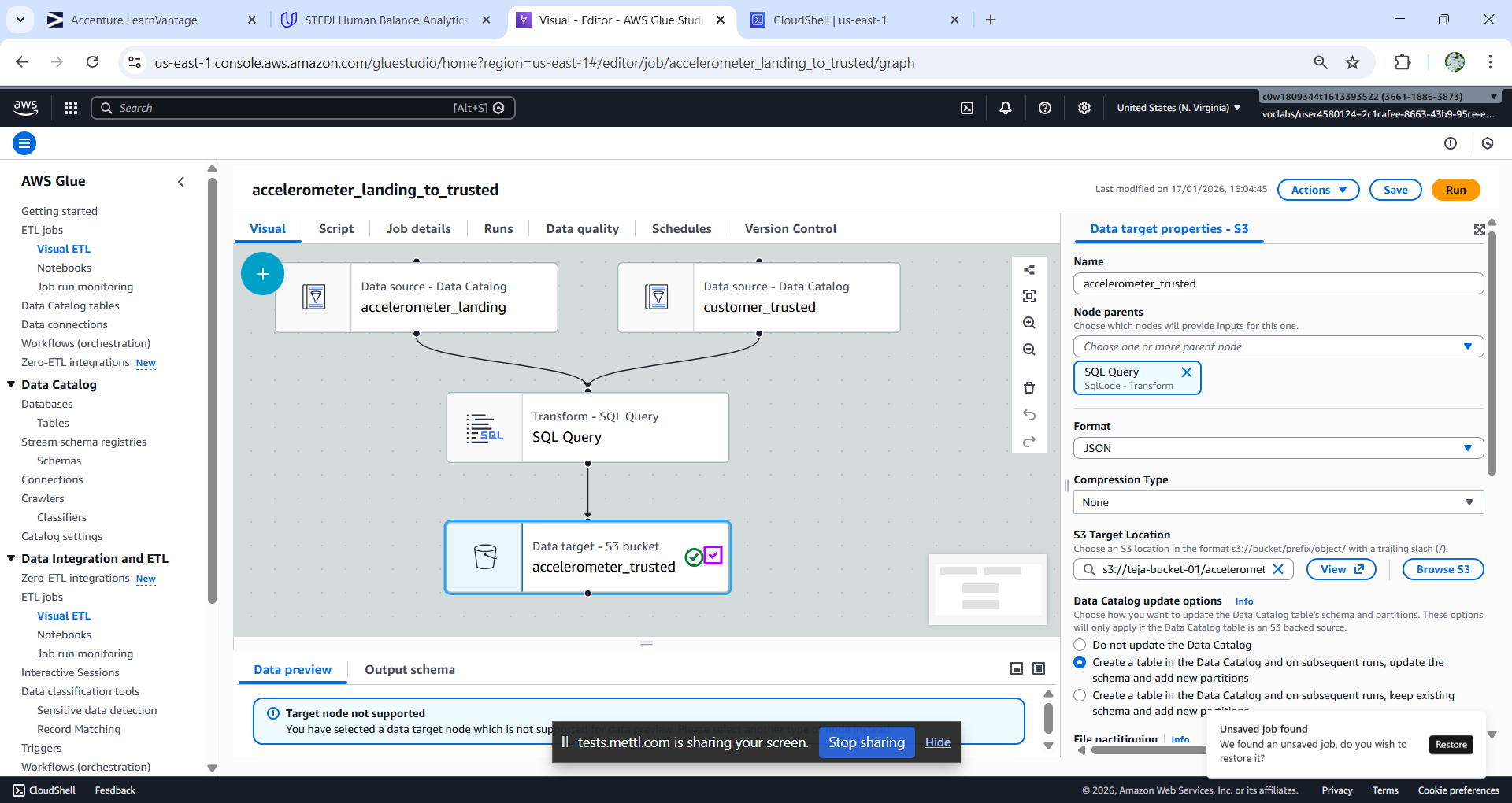
customer\_curted\_node1768647254241.setCatalogInfo(catalogDatabase="stedi",catalogTableName="customer\_curted")

customer\_curted\_node1768647254241.setFormat("json")

customer\_curted\_node1768647254241.writeFrame(SQLQuery\_node1768646753871)

job.commit()

Accelerometer landing to trusted:





Python 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 import DynamicFrame

def sparkSqlQuery(glueContext, query, mapping, transformation\_ctx) -> DynamicFrame:

for alias, frame in mapping.items():

frame.toDF().createOrReplaceTempView(alias)

result = spark.sql(query)

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 customer\_trusted

customer\_trusted\_node1768645493306 = glueContext.create\_dynamic\_frame.from\_catalog(database="stedi", table\_name="customer\_trusted", transformation\_ctx="customer\_trusted\_node1768645493306")

# Script generated for node accelerometer\_landing

accelerometer\_landing\_node1768645337937 = glueContext.create\_dynamic\_frame.from\_catalog(database="stedi", table\_name="accelerometer\_landing", transformation\_ctx="accelerometer\_landing\_node1768645337937")

# Script generated for node SQL Query

SqlQuery0 = '''

select al.\* from al join ct on al.user=ct.email;

'''

SQLQuery\_node1768645375097 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"al":accelerometer\_landing\_node1768645337937, "ct":customer\_trusted\_node1768645493306}, transformation\_ctx = "SQLQuery\_node1768645375097")

# Script generated for node accelerometer\_trusted

EvaluateDataQuality().process\_rows(frame=SQLQuery\_node1768645375097, ruleset=DEFAULT\_DATA\_QUALITY\_RULESET, publishing\_options={"dataQualityEvaluationContext": "EvaluateDataQuality\_node1768645277152", "enableDataQualityResultsPublishing": True}, additional\_options={"dataQualityResultsPublishing.strategy": "BEST\_EFFORT", "observations.scope": "ALL"})

accelerometer\_trusted\_node1768645773698 = glueContext.getSink(path="s3://teja-bucket-01/accelerometer/trusted/", connection\_type="s3", updateBehavior="UPDATE\_IN\_DATABASE", partitionKeys=[], enableUpdateCatalog=True, transformation\_ctx="accelerometer\_trusted\_node1768645773698")

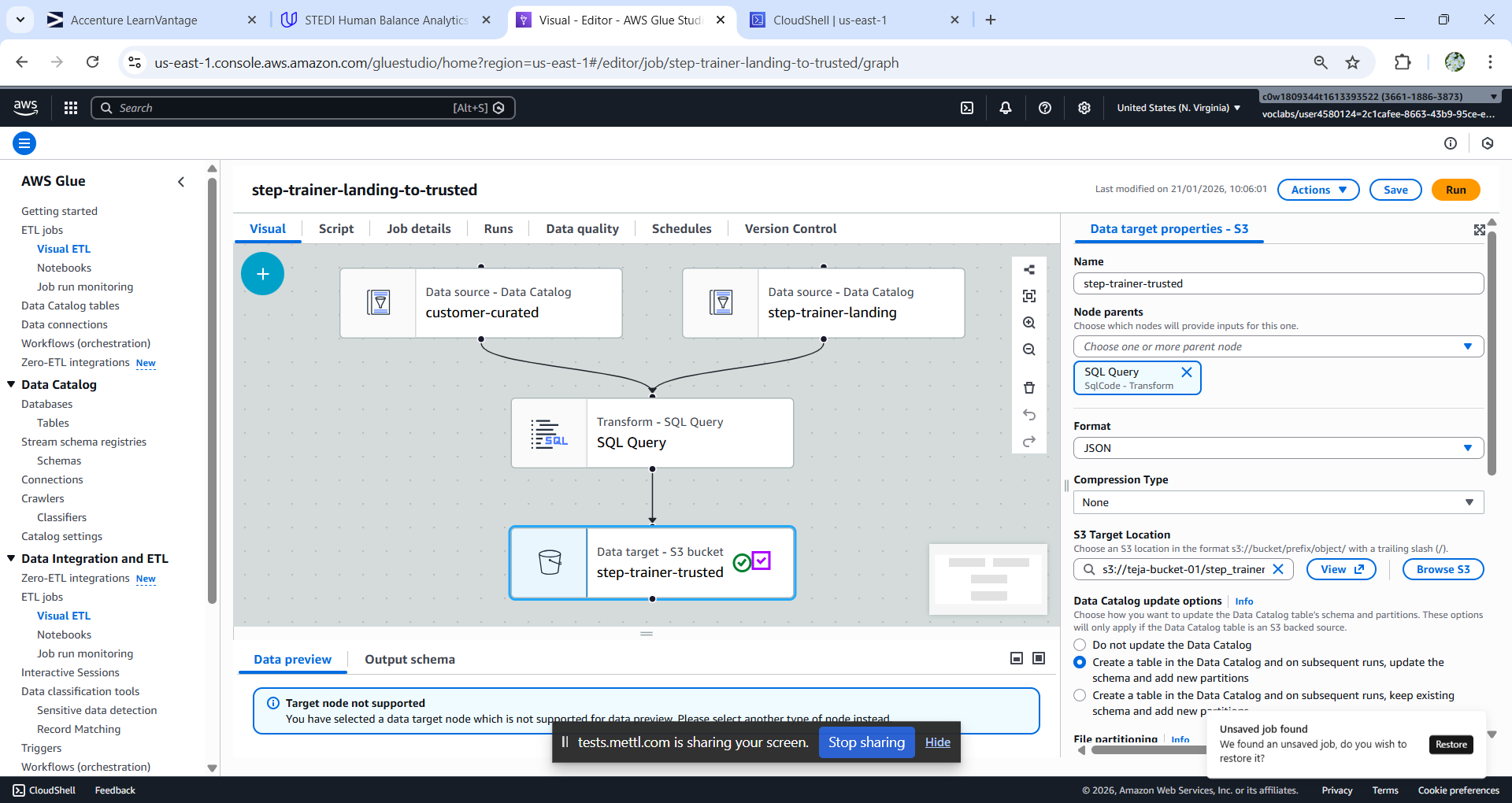
accelerometer\_trusted\_node1768645773698.setCatalogInfo(catalogDatabase="stedi",catalogTableName="accelerometer\_trusted")

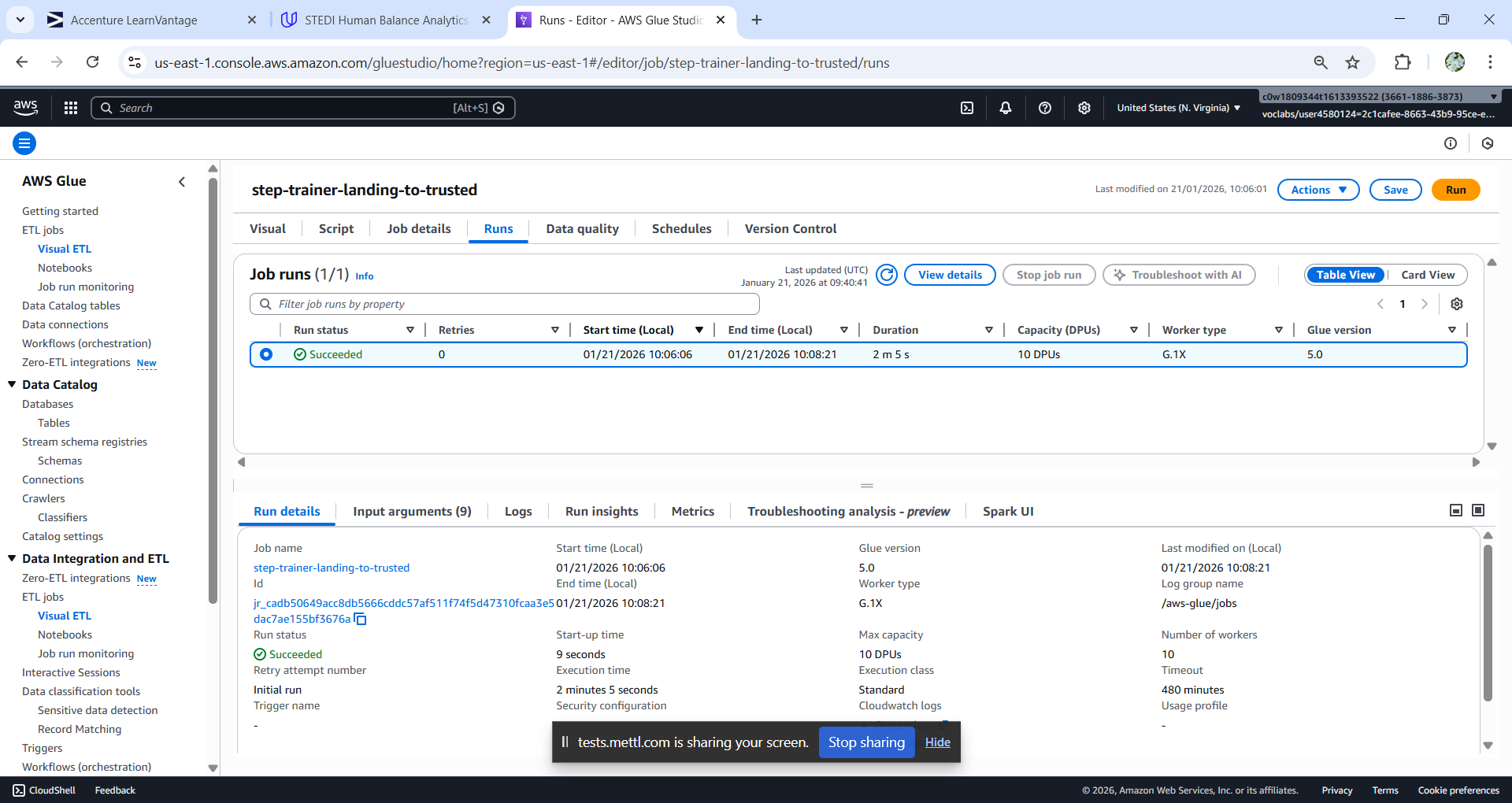
accelerometer\_trusted\_node1768645773698.setFormat("json")

accelerometer\_trusted\_node1768645773698.writeFrame(SQLQuery\_node1768645375097)

job.commit()

Step trainer landing to trusted:





Python 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 import DynamicFrame

def sparkSqlQuery(glueContext, query, mapping, transformation\_ctx) -> DynamicFrame:

for alias, frame in mapping.items():

frame.toDF().createOrReplaceTempView(alias)

result = spark.sql(query)

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 customer-curated

customercurated\_node1768969877844 = glueContext.create\_dynamic\_frame.from\_catalog(database="stedi", table\_name="customer\_curted", transformation\_ctx="customercurated\_node1768969877844")

# Script generated for node step-trainer-landing

steptrainerlanding\_node1768969817882 = glueContext.create\_dynamic\_frame.from\_catalog(database="stedi", table\_name="step-trainer-landing", transformation\_ctx="steptrainerlanding\_node1768969817882")

# Script generated for node SQL Query

SqlQuery0 = '''

select stl.\* from stl join cc on cc.serialNumber=stl.serialNumber;

'''

SQLQuery\_node1768969911136 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"cc":customercurated\_node1768969877844, "stl":steptrainerlanding\_node1768969817882}, transformation\_ctx = "SQLQuery\_node1768969911136")

# Script generated for node step-trainer-trusted

EvaluateDataQuality().process\_rows(frame=SQLQuery\_node1768969911136, ruleset=DEFAULT\_DATA\_QUALITY\_RULESET, publishing\_options={"dataQualityEvaluationContext": "EvaluateDataQuality\_node1768969807164", "enableDataQualityResultsPublishing": True}, additional\_options={"dataQualityResultsPublishing.strategy": "BEST\_EFFORT", "observations.scope": "ALL"})

steptrainertrusted\_node1768970001798 = glueContext.getSink(path="s3://teja-bucket-01/step\_trainer/trusted/", connection\_type="s3", updateBehavior="UPDATE\_IN\_DATABASE", partitionKeys=[], enableUpdateCatalog=True, transformation\_ctx="steptrainertrusted\_node1768970001798")

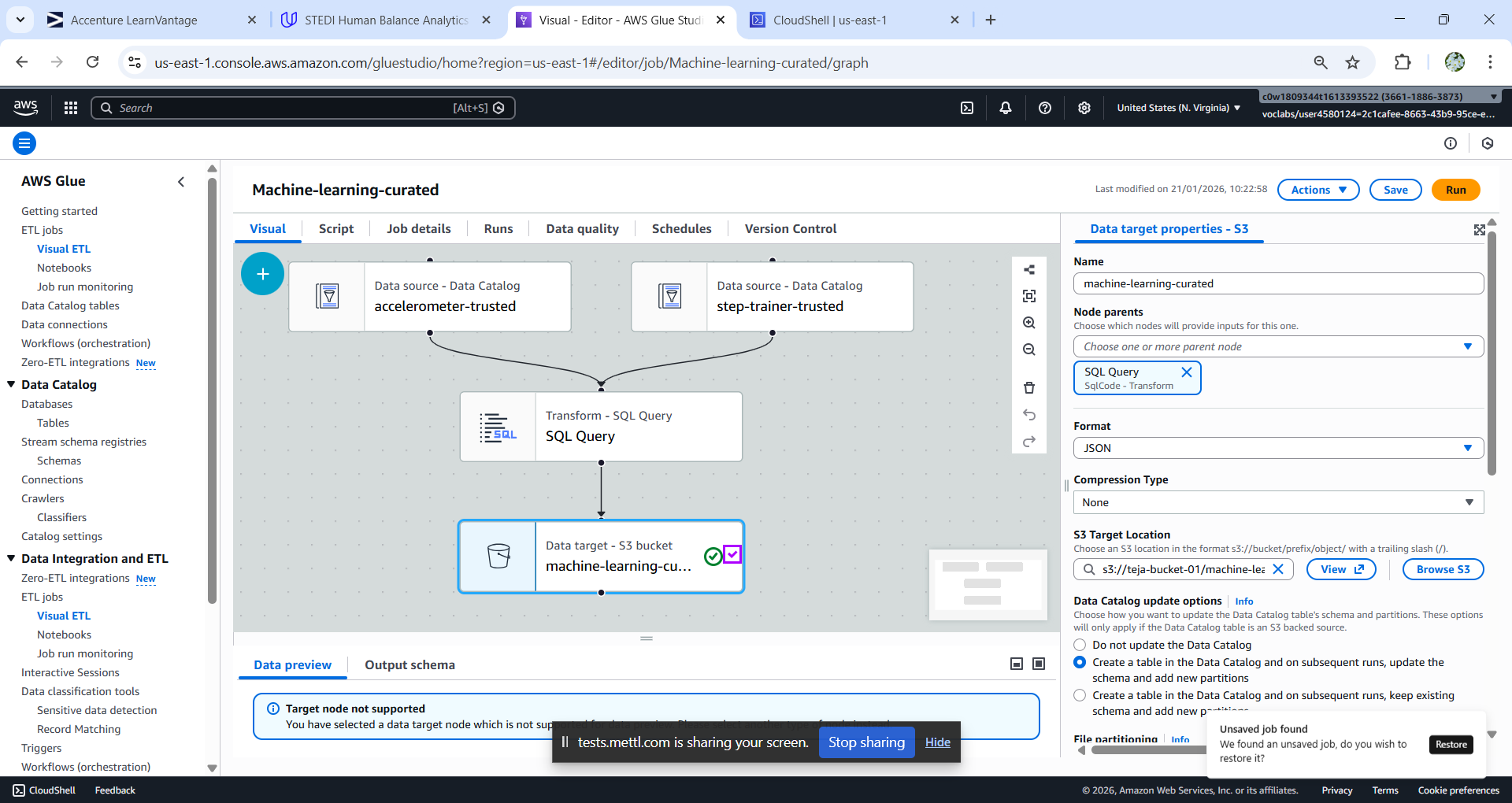
steptrainertrusted\_node1768970001798.setCatalogInfo(catalogDatabase="stedi",catalogTableName="step-trainer-trusted")

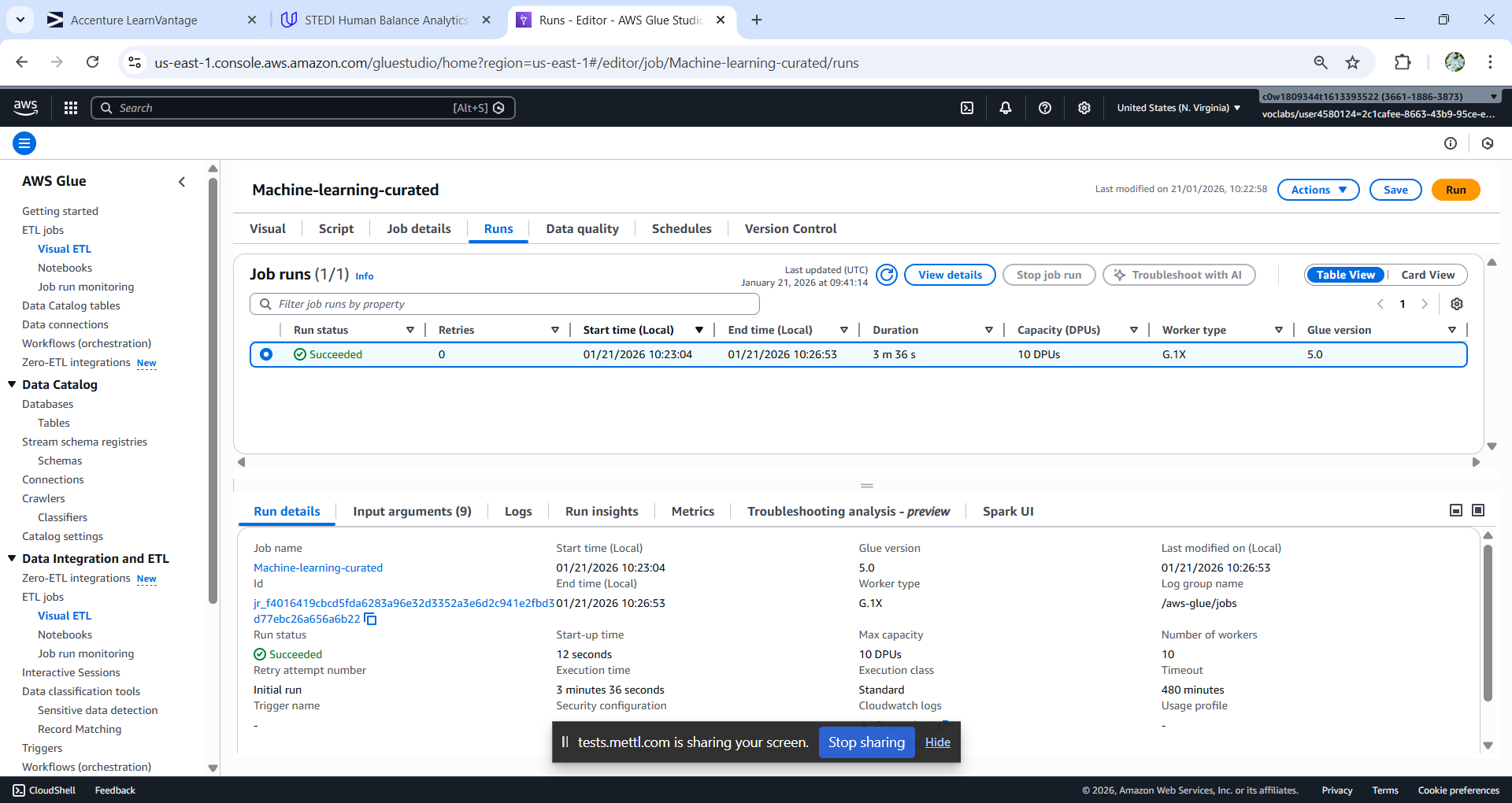
steptrainertrusted\_node1768970001798.setFormat("json")

steptrainertrusted\_node1768970001798.writeFrame(SQLQuery\_node1768969911136)

job.commit()

Machine Learning Curated:





Python 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 import DynamicFrame

def sparkSqlQuery(glueContext, query, mapping, transformation\_ctx) -> DynamicFrame:

for alias, frame in mapping.items():

frame.toDF().createOrReplaceTempView(alias)

result = spark.sql(query)

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 step-trainer-trusted

steptrainertrusted\_node1768970666842 = glueContext.create\_dynamic\_frame.from\_catalog(database="stedi", table\_name="step-trainer-trusted", transformation\_ctx="steptrainertrusted\_node1768970666842")

# Script generated for node accelerometer-trusted

accelerometertrusted\_node1768970604873 = glueContext.create\_dynamic\_frame.from\_catalog(database="stedi", table\_name="accelerometer\_trusted", transformation\_ctx="accelerometertrusted\_node1768970604873")

# Script generated for node SQL Query

SqlQuery0 = '''

select at.user,stt.\*,at.x,at.y,at.z from stt join at on at.timestamp=stt.sensorReadingTime;

'''

SQLQuery\_node1768970874226 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"at":accelerometertrusted\_node1768970604873, "stt":steptrainertrusted\_node1768970666842}, transformation\_ctx = "SQLQuery\_node1768970874226")

# Script generated for node machine-learning-curated

EvaluateDataQuality().process\_rows(frame=SQLQuery\_node1768970874226, ruleset=DEFAULT\_DATA\_QUALITY\_RULESET, publishing\_options={"dataQualityEvaluationContext": "EvaluateDataQuality\_node1768970843028", "enableDataQualityResultsPublishing": True}, additional\_options={"dataQualityResultsPublishing.strategy": "BEST\_EFFORT", "observations.scope": "ALL"})

machinelearningcurated\_node1768971022384 = glueContext.getSink(path="s3://teja-bucket-01/machine-learning-curated/", connection\_type="s3", updateBehavior="UPDATE\_IN\_DATABASE", partitionKeys=[], enableUpdateCatalog=True, transformation\_ctx="machinelearningcurated\_node1768971022384")

machinelearningcurated\_node1768971022384.setCatalogInfo(catalogDatabase="stedi",catalogTableName="machine-learning-curated")

machinelearningcurated\_node1768971022384.setFormat("json")

machinelearningcurated\_node1768971022384.writeFrame(SQLQuery\_node1768970874226)

job.commit()