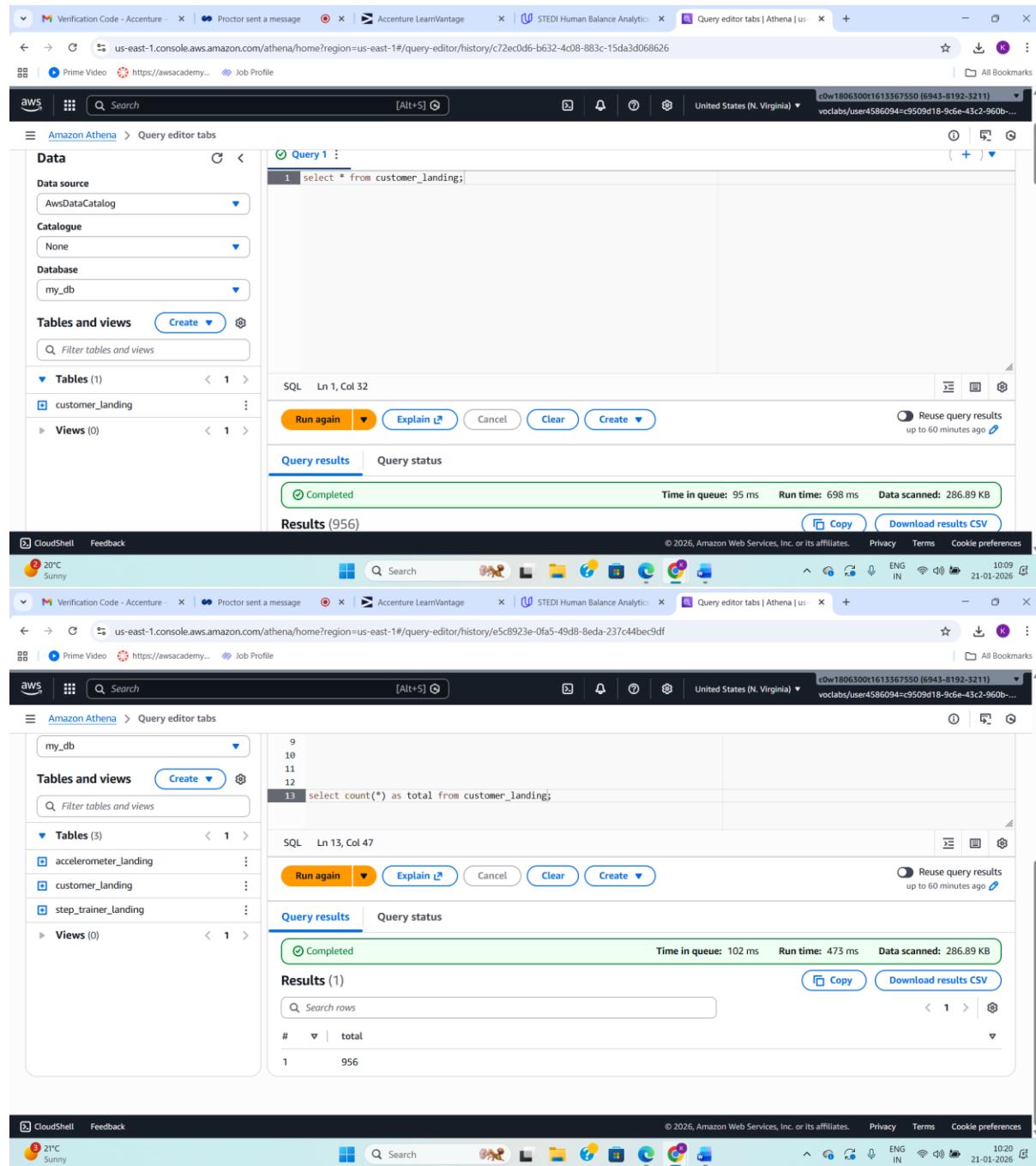


# STEDI HUMAN BALANCE ANALYTICS PROJECT

## ATHENA OUTPUTS

### 1.customer\_landing



The screenshot shows three stacked screenshots of the AWS Athena Query Editor interface. The top screenshot displays a single query: `select * from customer_landing;`. The middle screenshot shows a query for the count of rows in the `customer_landing` table. The bottom screenshot shows the results of the count query, which is 956.

**Query 1:**

```
1 select * from customer_landing;
```

**Results (956)**

#	total
1	956

Screenshot of the Amazon Athena Query Editor interface.

The interface shows a sidebar with the database "my\_db" selected. Under "Tables and views", there are three tables: "accelerometer\_landing", "customer\_landing", and "step\_trainer\_landing".

The main area displays the following SQL query:

```
8  
9  
10  
11  
12  
13 select count(shareWithResearchAsOfDate) as total from customer_landing;
```

The results section shows the output of the query:

#	total
1	482

Below the results, the status bar indicates:

- Time in queue: 103 ms
- Run time: 460 ms
- Data scanned: 286.89 KB

At the bottom, there are links for CloudShell, Feedback, and various AWS services like Lambda, S3, and CloudWatch.

## 2.accelerometer\_landing

The screenshot shows the AWS Athena Query Editor interface. On the left, the sidebar displays the Data source (AwsDataCatalog), Catalogue (None), Database (my\_db), and Tables and views (accelerometer\_landing, customer\_landing). The main area shows the SQL query:

```
1 select * from accelerometer_landing;
```

The status bar indicates the query is completed with the following metrics:

Time in queue: 109 ms Run time: 1.06 sec Data scanned: 6.55 MB

Results (81,273)

Below the results, there is a CloudShell interface showing the output of the query:

```
8
9
10
11
12
13 select count(*) as total from accelerometer_landing;
```

The CloudShell status bar shows the following information:

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences  
CloudShell Feedback 20°C Sunny Search ENG IN 1014 21-01-2026

The second screenshot shows the same interface after running the query again:

```
8
9
10
11
12
13 select count(*) as total from accelerometer_landing;
```

The status bar indicates the query is completed with the following metrics:

Time in queue: 112 ms Run time: 500 ms Data scanned: 6.55 MB

Results (1)

CloudShell status bar: © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences  
CloudShell Feedback 21°C Sunny Search ENG IN 1021 21-01-2026

### 3.step\_trainer\_landing

The image consists of three vertically stacked screenshots of the Amazon Athena Query Editor. Each screenshot shows a left sidebar with 'Data source' (AwsDataCatalog), 'Catalogue' (None), and 'Database' (my\_db). The 'Tables and views' section lists three tables: accelerometer\_landing, customer\_landing, and step\_trainer\_landing. The main area contains a SQL editor with the following queries:

Screenshot 1 (Top):

```
1 select * from step_trainer_landing;
```

Screenshot 2 (Middle):

```
9  
10  
11  
12  
13 select count(*) as total from step_trainer_landing;
```

Screenshot 3 (Bottom):

```
SQL Ln 13, Col 1
```

Run again Explain Cancel Clear Create Reuse query results up to 60 minutes ago

Query results | Query status

Completed Time in queue: 112 ms Run time: 688 ms Data scanned: 3.15 MB

Results (28,680) Copy Download results CSV

CloudShell Feedback 21°C Sunny Search Privacy Terms Cookie preferences 10:18 21-01-2026

Verification Code - Accenture Proctor sent a message Accenture LearnVantage STEDI Human Balance Analytics Query editor tabs | Athena | us... + All Bookmarks Prime Video https://awsacademy... Job Profile AWS Search [Alt+S] United States (N. Virginia) c0w1806300t1613367550 (6943-8192-3211) vclabs/user4586094=c9509d18-9c6e-43c2-960b...

CloudShell Feedback 21°C Sunny Search Privacy Terms Cookie preferences 10:18 21-01-2026

Verification Code - Accenture Proctor sent a message Accenture LearnVantage STEDI Human Balance Analytics Query editor tabs | Athena | us... + All Bookmarks Prime Video https://awsacademy... Job Profile AWS Search [Alt+S] United States (N. Virginia) c0w1806300t1613367550 (6943-8192-3211) vclabs/user4586094=c9509d18-9c6e-43c2-960b...

CloudShell Feedback 21°C Sunny Search Privacy Terms Cookie preferences 10:20 21-01-2026

Verification Code - Accenture Proctor sent a message Accenture LearnVantage STEDI Human Balance Analytics Query editor tabs | Athena | us... + All Bookmarks Prime Video https://awsacademy... Job Profile AWS Search [Alt+S] United States (N. Virginia) c0w1806300t1613367550 (6943-8192-3211) vclabs/user4586094=c9509d18-9c6e-43c2-960b...

## SQL DDL SCRIPTS

### 1.customer\_landing.sql

```
CREATE EXTERNAL TABLE `customer_landing`(  
`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',
`sharewithpublicasofdate` bigint COMMENT 'from deserializer',
`registrationdate` bigint COMMENT 'from deserializer',
`sharewithresearchasofdate` bigint COMMENT 'from deserializer',
`lastupdatedate` 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://stedi-uda-project/landing/customer_landing/'
```

TBLPROPERTIES (

```
'classification'='json')
```

## **2.accelerometer\_landing.sql**

```
CREATE EXTERNAL TABLE `accelerometer_landing`(
`user` string COMMENT 'from deserializer',
`timestamp` string 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'
```

STORED AS INPUTFORMAT

```
'org.apache.hadoop.mapred.TextInputFormat'
```

OUTPUTFORMAT

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

LOCATION

```
's3://stedi-uda-project/landing/accelerometer_landing/'
```

TBLPROPERTIES (

'classification'='json')

### **3.step\_trainer\_landing.sql**

CREATE EXTERNAL TABLE `step\_trainer\_landing`(`

`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

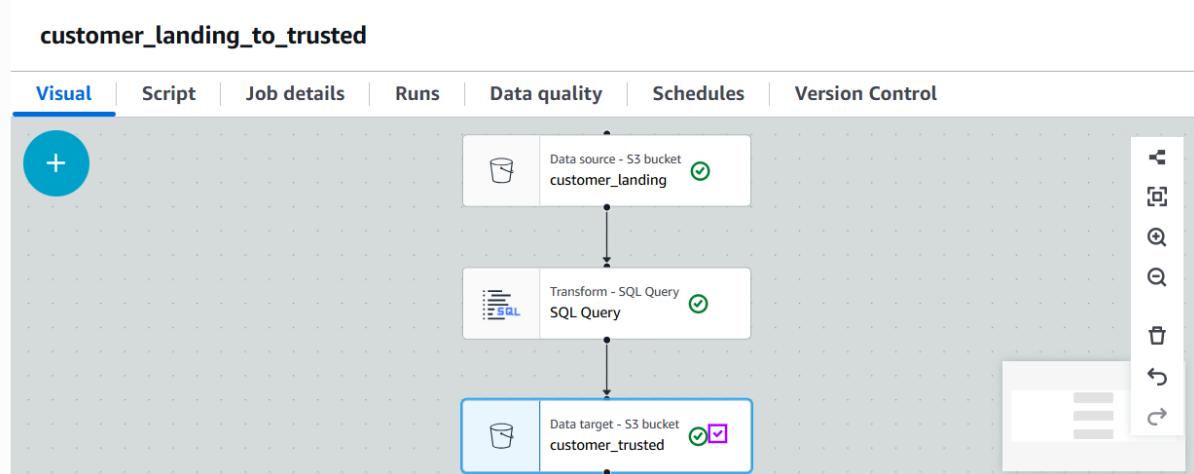
's3://stedi-uda-project/landing/step\_trainer\_landing/'

TBLPROPERTIES (

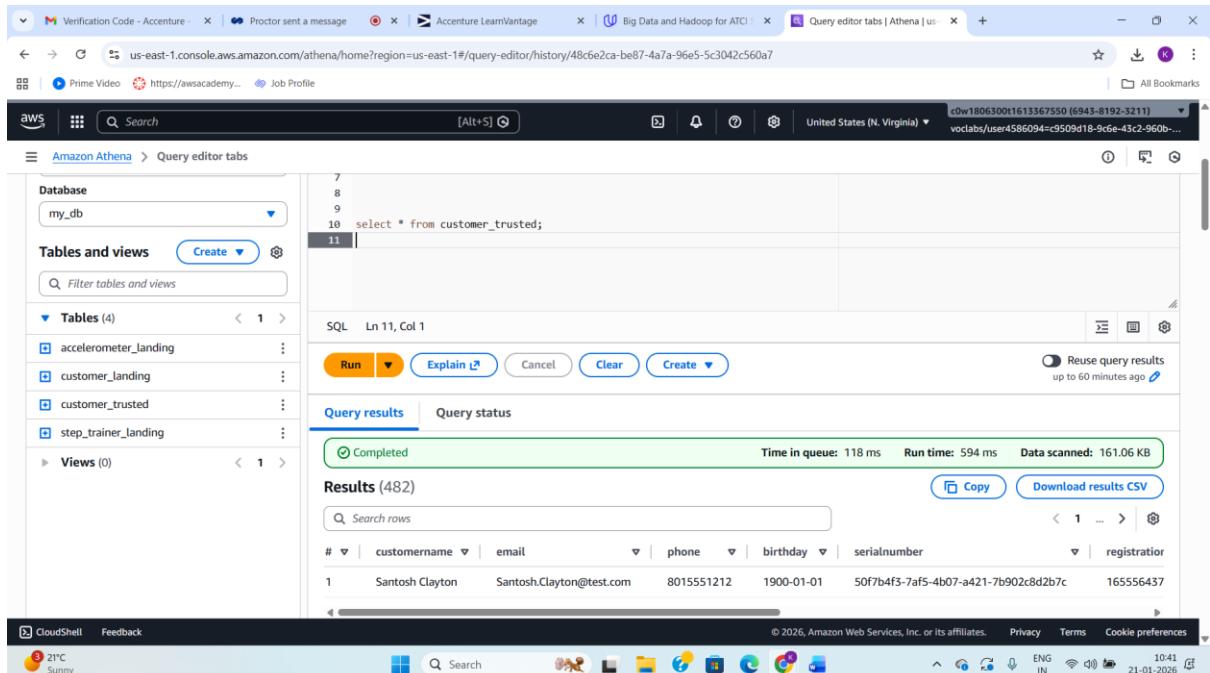
'classification'='json')

### **GLUE STUDIO OUTPUTS**

#### **1.customer\_landing\_to\_trusted**



## Customer\_trusted



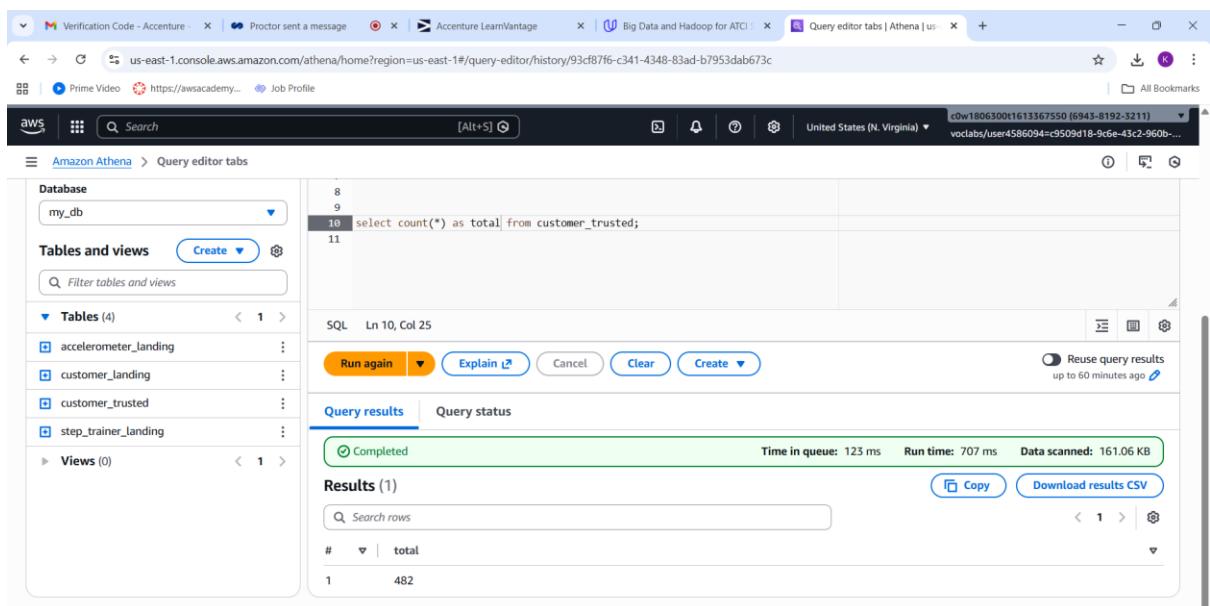
The screenshot shows the AWS Athena Query Editor interface. The database is set to 'my\_db'. In the 'Tables and views' section, the 'customer\_trusted' table is selected. The query editor contains the following SQL:

```
7  
8  
9  
10 select * from customer_trusted;  
11
```

The results tab shows 482 rows of data:

#	customername	email	phone	birthday	serialnumber	registration
1	Santosh Clayton	Santosh.Clayton@test.com	8015551212	1900-01-01	50f7b4f3-7af5-4b07-a421-7b902c8d2b7c	165556437

CloudShell and Feedback buttons are visible at the bottom left, and a system tray with weather information (21°C, Sunny) is at the bottom right.

The screenshot shows the AWS Athena Query Editor interface. The database is set to 'my\_db'. In the 'Tables and views' section, the 'customer\_trusted' table is selected. The query editor contains the following SQL:

```
8  
9  
10 select count(*) as total from customer_trusted;  
11
```

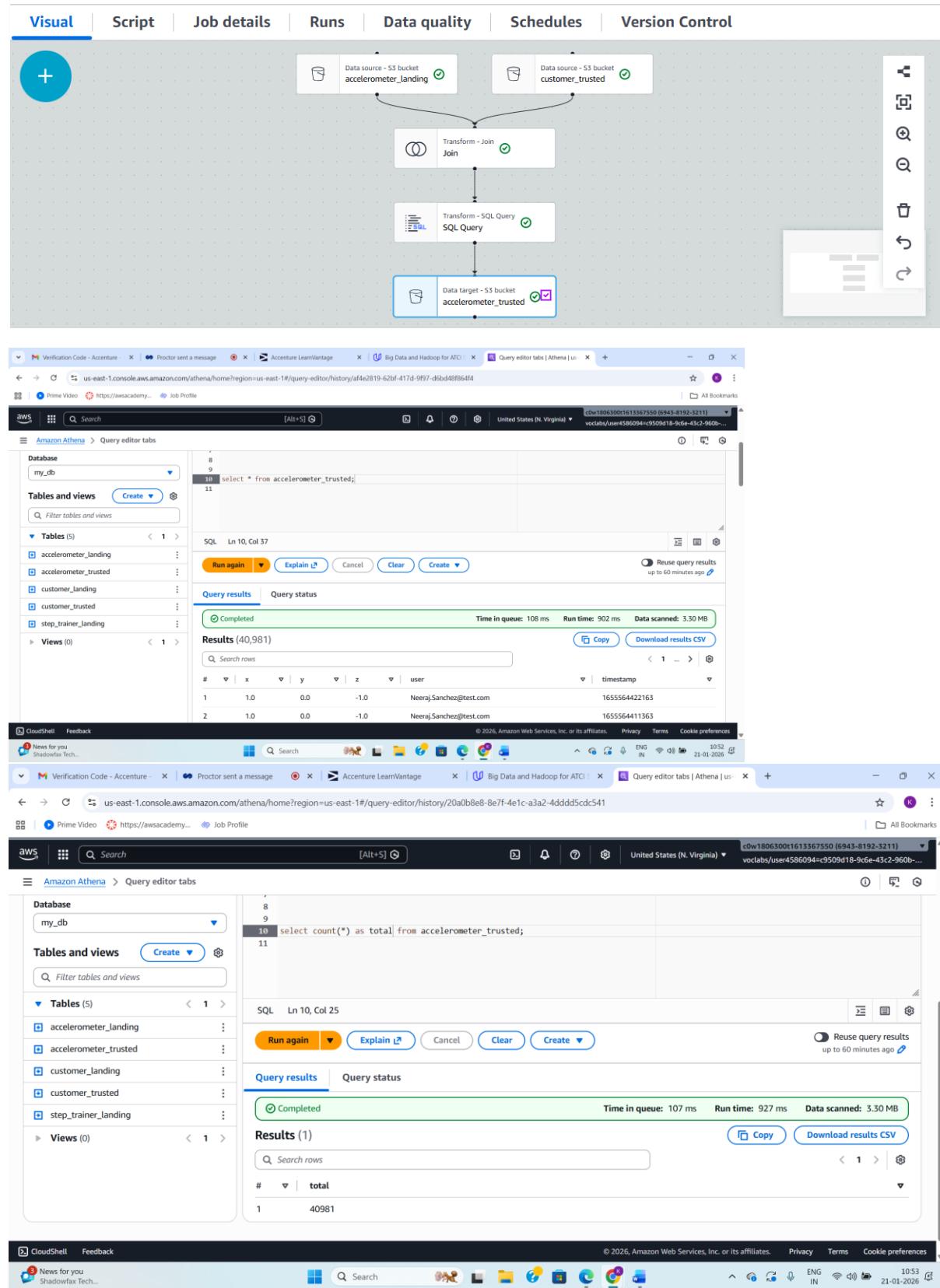
The results tab shows 1 row of data:

#	total
1	482

CloudShell and Feedback buttons are visible at the bottom left, and a system tray with weather information (21°C, Sunny) is at the bottom right.

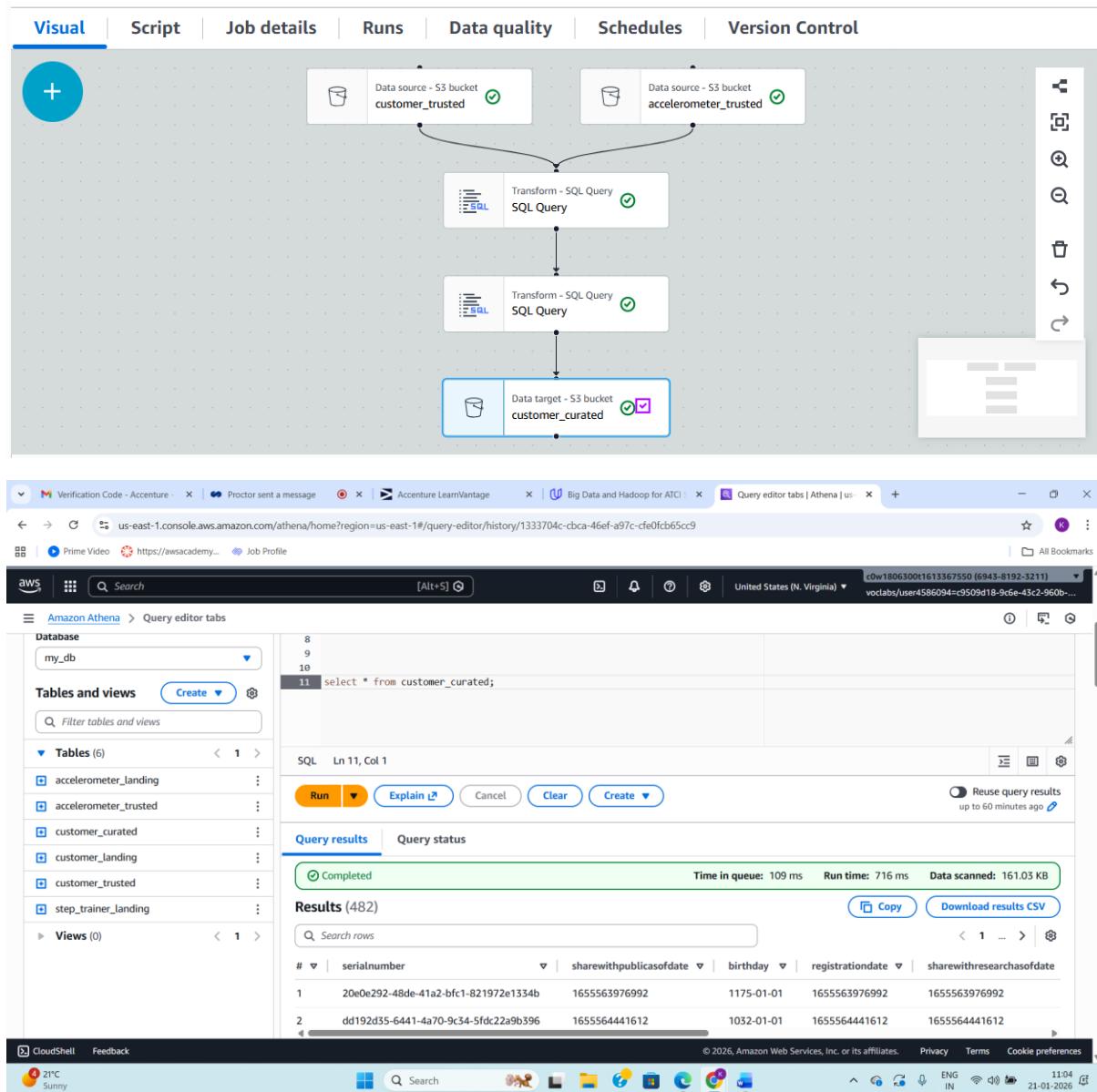
## 2.accelerometer\_landing\_to\_trusted

### accelerometer\_landing\_to\_trusted



### 3.customer curated

#### customer\_curated



The screenshot shows the Amazon Athena Query Editor interface. On the left, the 'Tables and views' section lists tables such as 'accelerometer\_landing', 'customer\_curated', and 'step\_trainer\_landing'. The main area displays a query log with the following content:

```

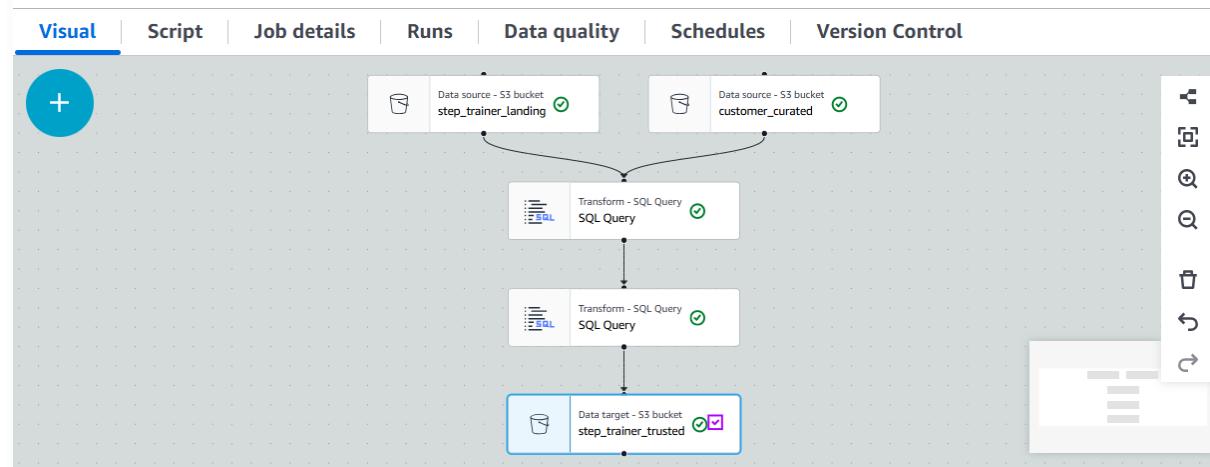
8
9
10
11 select count(*) from customer_curated;

```

Below the log, it says 'SQL Ln 11, Col 16'. There are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. The 'Query results' tab is selected, showing a green status bar with 'Completed' and performance metrics: 'Time in queue: 459 ms', 'Run time: 644 ms', and 'Data scanned: 161.03 KB'. The results table shows one row with '\_col0' and value '482'. At the bottom right, there are 'Copy' and 'Download results CSV' buttons.

#### 4.step\_trainer\_trusted

##### step\_trainer\_landing\_to\_trusted



The screenshot shows the AWS Lambda interface with the following details:

- Database:** my\_db
- Tables and views:** Step\_trainer\_trusted
- Query:**

```
8  
9  
10  
11 select * from step_trainer_trusted;
```

- SQL:** SQL Ln 11, Col 35
- Buttons:** Run again, Explain, Cancel, Clear, Create, Reuse query results (up to 60 minutes ago)
- Query results:** Completed, Time in queue: 109 ms, Run time: 687 ms, Data scanned: 1.59 MB
- Results:** 14,460 rows

#	sensorreadingtime	serialnumber	distancefromobject
1	1655564474771	058af4ef-68ed-4af7-b51d-b6223e5300ff	237
2	1655564192614	bc49e3fb-d660-4058-87f0-59434133146e	215

- CloudShell:** Feedback, Weather: 21°C Sunny
- Footer:** © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences, ENG IN, 21-01-2026, 11:17

The screenshot shows the AWS Lambda interface with the following details:

- Database:** my\_db
- Tables and views:** Step\_trainer\_trusted
- Query:**

```
8  
9  
10  
11 select count(*) as total from step_trainer_trusted;
```

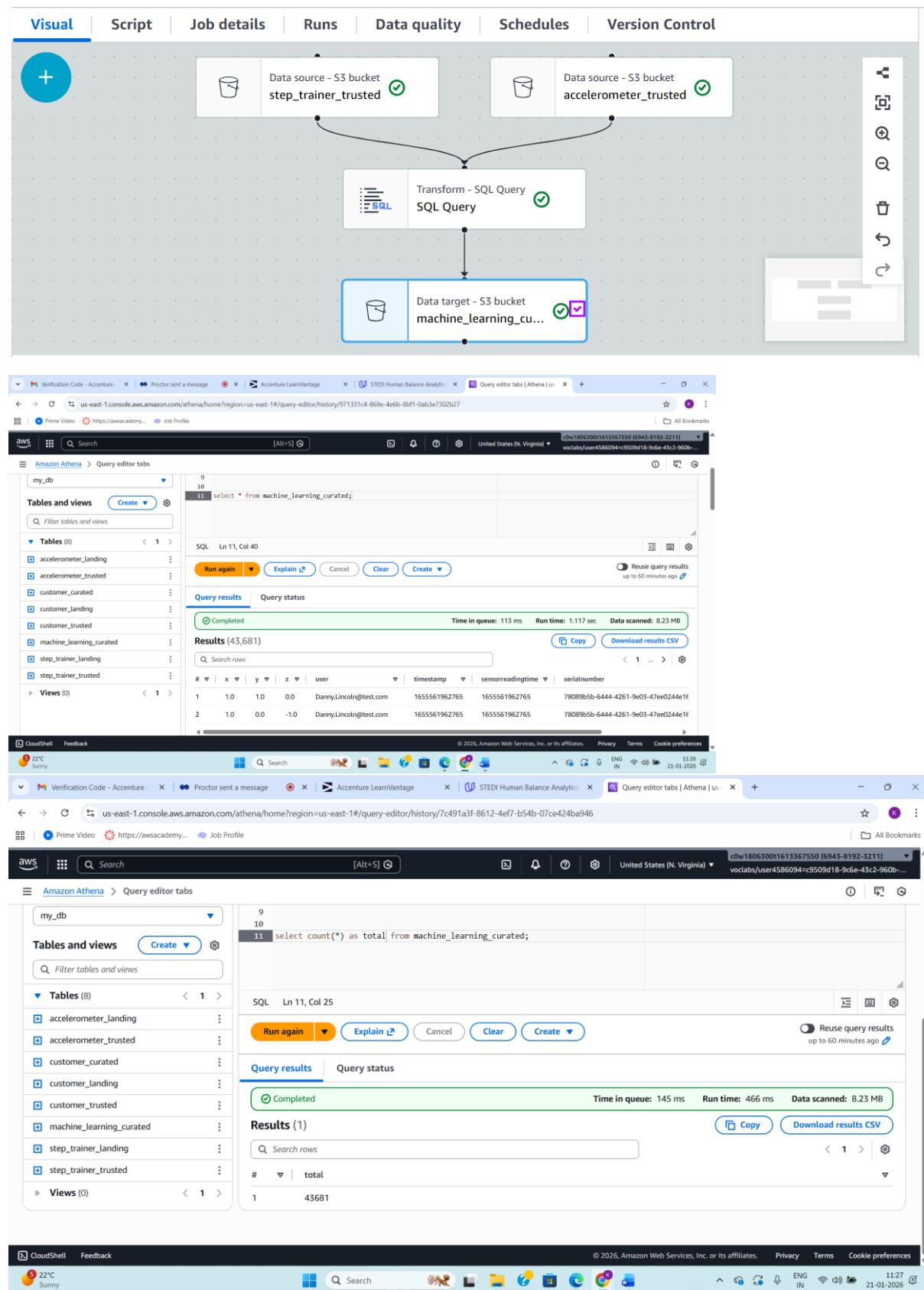
- SQL:** SQL Ln 11, Col 25
- Buttons:** Run again, Explain, Cancel, Clear, Create, Reuse query results (up to 60 minutes ago)
- Query results:** Completed, Time in queue: 120 ms, Run time: 500 ms, Data scanned: 1.59 MB
- Results:** 1 row

#	total
1	14460

- CloudShell:** Feedback, Weather: 21°C Sunny
- Footer:** © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences, ENG IN, 21-01-2026, 11:17

## 5.machine\_learning\_curated

### machine\_learning\_curated



## PYTHON SCRIPTS

### 1.customer\_landing\_to\_trusted.py

```
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
```

```

customer_landing_node1768971819631 =
glueContext.create_dynamic_frame.from_options(format_options={"multiLine": "false"},
connection_type="s3", format="json", connection_options={"paths": ["s3://stedi-uda-
project/landing/customer_landing/"]}, "reurse": True},
transformation_ctx="customer_landing_node1768971819631")

# Script generated for node SQL Query
SqlQuery0 = ""
select * from myDataSource
where shareWithResearchAsOfDate IS NOT NULL;
"""

SQLQuery_node1768971823092 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping =
{"myDataSource":customer_landing_node1768971819631}, transformation_ctx =
"SQLQuery_node1768971823092")

# Script generated for node customer_trusted
EvaluateDataQuality().process_rows(frame=SQLQuery_node1768971823092,
ruleset=DEFAULT_DATA_QUALITY_RULESET, publishing_options={"dataQualityEvaluationContext":
"EvaluateDataQuality_node1768971763016", "enableDataQualityResultsPublishing": True},
additional_options={"dataQualityResultsPublishing.strategy": "BEST EFFORT", "observations.scope":
"ALL"})

customer_trusted_node1768971825664 = glueContext.getSink(path="s3://stedi-uda-
project/landing/customer_landing/trusted/", connection_type="s3",
updateBehavior="UPDATE_IN_DATABASE", partitionKeys=[], enableUpdateCatalog=True,
transformation_ctx="customer_trusted_node1768971825664")

customer_trusted_node1768971825664.setCatalogInfo(catalogDatabase="my_db",catalogTableName=
"customer_trusted")

customer_trusted_node1768971825664.setFormat("json")

customer_trusted_node1768971825664.writeFrame(SQLQuery_node1768971823092)

job.commit()

2.accelerometer_landing_to_trusted.py

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_node1768972441910 =
glueContext.create_dynamic_frame.from_options(format_options={"multiLine": "false"}, connection_type="s3", format="json", connection_options={"paths": ["s3://stedi-uda-project/landing/accelerometer_landing/"]}, "reurse": True}, transformation_ctx="accelerometer_landing_node1768972441910")

# Script generated for node customer_trusted
customer_trusted_node1768972439414 =
glueContext.create_dynamic_frame.from_options(format_options={"multiLine": "false"},
```

```
connection_type="s3", format="json", connection_options={"paths": ["s3://stedi-uda-project/landing/customer_landing/trusted/"], "recurse": True}, transformation_ctx="customer_trusted_node1768972439414")
```

```
# Script generated for node Join
```

```
Join_node1768972445017 = Join.apply(frame1=accelerometer_landing_node1768972441910, frame2=customer_trusted_node1768972439414, keys1=["user"], keys2=["email"], transformation_ctx="Join_node1768972445017")
```

```
# Script generated for node SQL Query
```

```
SqlQuery0 = ""  
select DISTINCT x,y,z,user,timestamp from myDataSource;
```

```
""
```

```
SQLQuery_node1768972446638 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"myDataSource":Join_node1768972445017}, transformation_ctx = "SQLQuery_node1768972446638")
```

```
# Script generated for node accelerometer_trusted
```

```
EvaluateDataQuality().process_rows(frame=SQLQuery_node1768972446638, ruleset=DEFAULT_DATA_QUALITY_RULESET, publishing_options={"dataQualityEvaluationContext": "EvaluateDataQuality_node1768972399765", "enableDataQualityResultsPublishing": True}, additional_options={"dataQualityResultsPublishing.strategy": "BEST EFFORT", "observations.scope": "ALL"})
```

```
accelerometer_trusted_node1768972450560 = glueContext.getSink(path="s3://stedi-uda-project/landing/accelerometer_landing/trusted/", connection_type="s3", updateBehavior="UPDATE_IN_DATABASE", partitionKeys=[], enableUpdateCatalog=True, transformation_ctx="accelerometer_trusted_node1768972450560")
```

```
accelerometer_trusted_node1768972450560.setCatalogInfo(catalogDatabase="my_db",catalogTableName="accelerometer_trusted")
```

```
accelerometer_trusted_node1768972450560.setFormat("json")
```

```
accelerometer_trusted_node1768972450560.writeFrame(SQLQuery_node1768972446638)
```

```
job.commit()
```

### **3.customer\_curated.py**

```
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_trusted
accelerometer_trusted_node1768973088768 =
    glueContext.create_dynamic_frame.from_options(format_options={"multiLine": "false"}, connection_type="s3", format="json", connection_options={"paths": ["s3://stedi-uda-project/landing/accelerometer_landing/trusted/"]}, "recurse": True}, transformation_ctx="accelerometer_trusted_node1768973088768")

```

```
# Script generated for node customer_trusted

customer_trusted_node1768973090716 =
glueContext.create_dynamic_frame.from_options(format_options={"multiLine": "false"}, connection_type="s3", format="json", connection_options={"paths": ["s3://stedi-uda-project/landing/customer_landing/trusted/"]}, "reurse": True}, transformation_ctx="customer_trusted_node1768973090716")

# Script generated for node SQL Query

SqlQuery0 = ""

select * from a join c on a.user=c.email;

"""

SQLQuery_node1768973093934 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"c":customer_trusted_node1768973090716, "a":accelerometer_trusted_node1768973088768}, transformation_ctx = "SQLQuery_node1768973093934")

# Script generated for node SQL Query

SqlQuery1 = ""

select distinct
    serialnumber,
    sharewithpublicasofdate,
    birthday,
    registrationdate,
    sharewithresearchasofdate,
    customername,
    email,
    lastupdatedate,
    phone,
    sharewithfriendsasofdate
from myDataSource;

"""


```

```

SQLQuery_node1768973095370 = sparkSqlQuery(glueContext, query = SqlQuery1, mapping =
>{"myDataSource":SQLQuery_node1768973093934}, transformation_ctx =
"SQLQuery_node1768973095370")

# Script generated for node customer_curated

EvaluateDataQuality().process_rows(frame=SQLQuery_node1768973095370,
ruleset=DEFAULT_DATA_QUALITY_RULESET, publishing_options={"dataQualityEvaluationContext":
"EvaluateDataQuality_node1768973060146", "enableDataQualityResultsPublishing": True},
additional_options={"dataQualityResultsPublishing.strategy": "BEST EFFORT", "observations.scope":
"ALL"})

customer_curated_node1768973099281 = glueContext.getSink(path="s3://stedi-uda-
project/landing/customer_landing/curated/", connection_type="s3",
updateBehavior="UPDATE_IN_DATABASE", partitionKeys=[], enableUpdateCatalog=True,
transformation_ctx="customer_curated_node1768973099281")

customer_curated_node1768973099281.setCatalogInfo(catalogDatabase="my_db",catalogTableName
e="customer_curated")

customer_curated_node1768973099281.setFormat("json")

customer_curated_node1768973099281.writeFrame(SQLQuery_node1768973095370)

job.commit()

```

#### **4.step\_trainer\_landing\_to\_trusted.py**

```

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

customer_curated_node1768973786330 =
glueContext.create_dynamic_frame.from_options(format_options={"multiLine": "false"}, connection_type="s3", format="json", connection_options={"paths": ["s3://stedi-uda-project/landing/customer_landing/curated/"]}, "recurse": True}, transformation_ctx="customer_curated_node1768973786330")

# Script generated for node step_trainer_landing

step_trainer_landing_node1768973787902 =
glueContext.create_dynamic_frame.from_options(format_options={"multiLine": "false"}, connection_type="s3", format="json", connection_options={"paths": ["s3://stedi-uda-project/landing/step_trainer_landing/"]}, "recurse": True}, transformation_ctx="step_trainer_landing_node1768973787902")

# Script generated for node SQL Query

SqlQuery0 = ""

select s.sensorReadingTime,
s.serialNumber,
s.distanceFromObject from s join c
on s.serialnumber=c.serialnumber;

```

```

"""

SQLQuery_node1768973791218 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping =
{"s":step_trainer_landing_node1768973787902, "c":customer_curated_node1768973786330},
transformation_ctx = "SQLQuery_node1768973791218")

# Script generated for node SQL Query

SqlQuery1 = ""

select distinct sensorReadingTime,
serialNumber,
distanceFromObject from myDataSource;

"""

SQLQuery_node1768973792561 = sparkSqlQuery(glueContext, query = SqlQuery1, mapping =
{"myDataSource":SQLQuery_node1768973791218}, transformation_ctx =
"SQLQuery_node1768973792561")

# Script generated for node step_trainer_trusted

EvaluateDataQuality().process_rows(frame=SQLQuery_node1768973792561,
ruleset=DEFAULT_DATA_QUALITY_RULESET, publishing_options={"dataQualityEvaluationContext":
"EvaluateDataQuality_node1768973743058", "enableDataQualityResultsPublishing": True},
additional_options={"dataQualityResultsPublishing.strategy": "BEST EFFORT", "observations.scope":
"ALL"})

step_trainer_trusted_node1768973796390 = glueContext.getSink(path="s3://stedi-uda-
project/landing/step_trainer_landing/trusted/", connection_type="s3",
updateBehavior="UPDATE_IN_DATABASE", partitionKeys=[], enableUpdateCatalog=True,
transformation_ctx="step_trainer_trusted_node1768973796390")

step_trainer_trusted_node1768973796390.setCatalogInfo(catalogDatabase="my_db",catalogTableName=
"step_trainer_trusted")

step_trainer_trusted_node1768973796390.setFormat("json")

step_trainer_trusted_node1768973796390.writeFrame(SQLQuery_node1768973792561)

job.commit()

5.machine_learning_curated.py

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
customer_curated_node1768973786330 =
    glueContext.create_dynamic_frame.from_options(format_options={"multiLine": "false"}, connection_type="s3", format="json", connection_options={"paths": ["s3://stedi-uda-project/landing/customer_landing/curated/"]}, "recurse": True}, transformation_ctx="customer_curated_node1768973786330")

```

```

# Script generated for node step_trainer_landing

step_trainer_landing_node1768973787902 =
glueContext.create_dynamic_frame.from_options(format_options={"multiLine": "false"},
connection_type="s3", format="json", connection_options={"paths": ["s3://stedi-uda-
project/landing/step_trainer_landing/"]}, "reurse": True},
transformation_ctx="step_trainer_landing_node1768973787902")



# Script generated for node SQL Query

SqlQuery0 = """
select s.sensorReadingTime,
s.serialNumber,
s.distanceFromObject from s join c
on s.serialnumber=c.serialnumber;

"""

SQLQuery_node1768973791218 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping =
{"s":step_trainer_landing_node1768973787902, "c":customer_curated_node1768973786330},
transformation_ctx = "SQLQuery_node1768973791218")



# Script generated for node SQL Query

SqlQuery1 = """
select distinct sensorReadingTime,
serialNumber,
distanceFromObject from myDataSource;

"""

SQLQuery_node1768973792561 = sparkSqlQuery(glueContext, query = SqlQuery1, mapping =
{"myDataSource":SQLQuery_node1768973791218}, transformation_ctx =
"SQLQuery_node1768973792561")



# Script generated for node step_trainer_trusted

EvaluateDataQuality().process_rows(frame=SQLQuery_node1768973792561,
ruleset=DEFAULT_DATA_QUALITY_RULESET, publishing_options={"dataQualityEvaluationContext":
```

```
"EvaluateDataQuality_node1768973743058", "enableDataQualityResultsPublishing": True},  
additional_options={"dataQualityResultsPublishing.strategy": "BEST EFFORT", "observations.scope":  
"ALL"})  
  
step_trainer_trusted_node1768973796390 = glueContext.getSink(path="s3://stedi-uda-  
project/landing/step_trainer_landing/trusted/", connection_type="s3",  
updateBehavior="UPDATE_IN_DATABASE", partitionKeys=[], enableUpdateCatalog=True,  
transformation_ctx="step_trainer_trusted_node1768973796390")  
  
step_trainer_trusted_node1768973796390.setCatalogInfo(catalogDatabase="my_db",catalogTableName=  
"step_trainer_trusted")  
  
step_trainer_trusted_node1768973796390.setFormat("json")  
  
step_trainer_trusted_node1768973796390.writeFrame(SQLQuery_node1768973792561)  
  
job.commit()
```