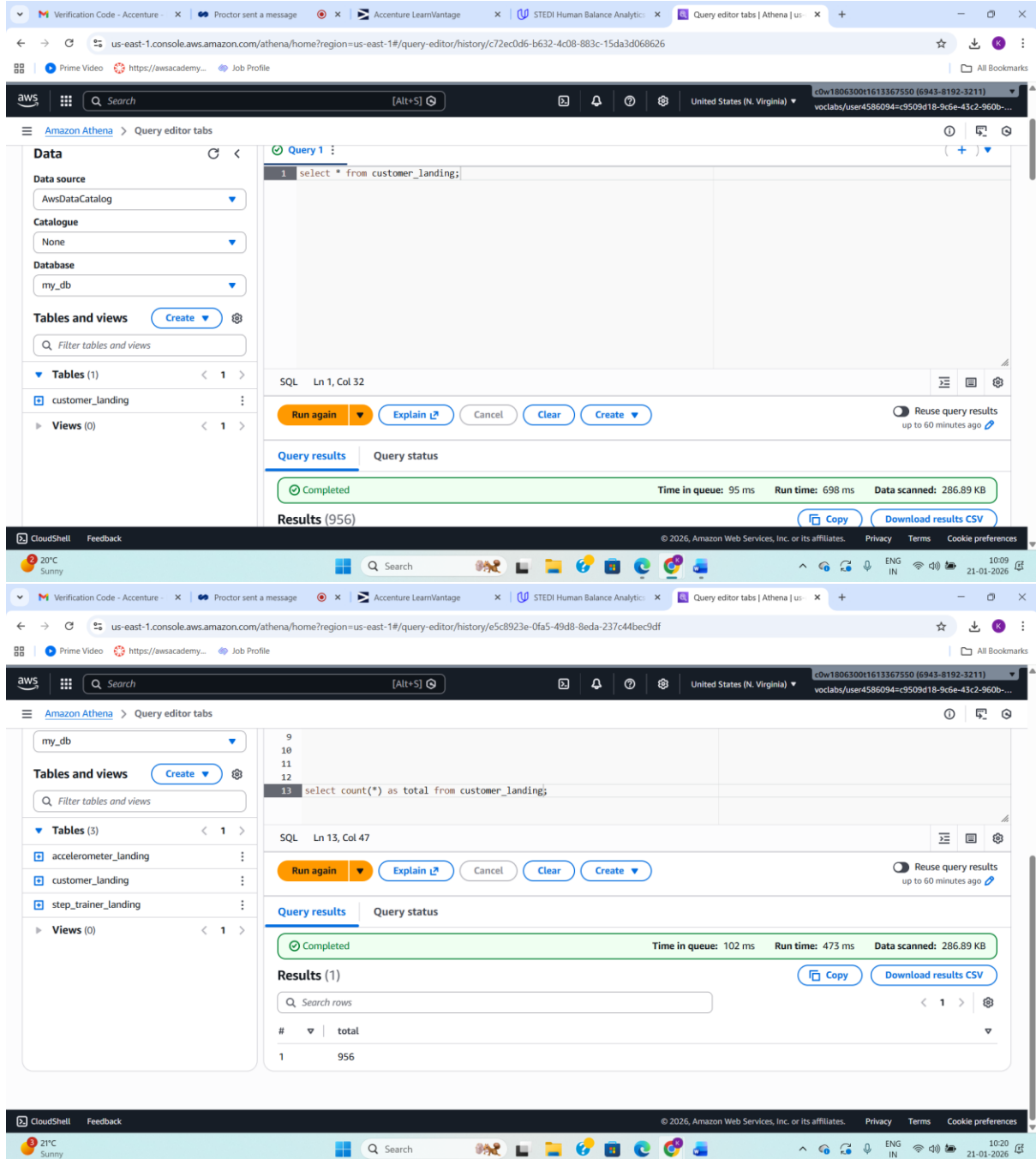


1.customer_landing



us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/73403ba4-d587-44c8-a0d6-6fb821223f36

Amazon Athena > Query editor tabs

Database: my_db

Tables and views [Create](#)

Filter tables and views

Tables (3)

- accelerometer_landing
- customer_landing
- step_trainer_landing

Views (0)

```
13 select count(sharewithResearchAsOfDate) as total from customer_landing;
```

SQL Ln 13, Col 71

[Run again](#) [Explain](#) [Cancel](#) [Clear](#) [Create](#)

☐ Reuse query results up to 60 minutes ago

[Query results](#) [Query status](#)

[Completed](#) Time in queue: 103 ms Run time: 460 ms Data scanned: 286.89 KB

[Results \(1\)](#) [Copy](#) [Download results CSV](#)

Search rows

| # | total |
|---|-------|
| 1 | 482 |

CloudShell Feedback

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

21°C Sunny 10:24 21-01-2026

2.accelerometer_landing

The screenshot displays the Amazon Athena Query Editor interface. The left sidebar shows the 'Data source' as 'AwsDataCatalog', 'Catalogue' as 'None', and 'Database' as 'my_db'. Under 'Tables and views', two tables are listed: 'accelerometer_landing' and 'customer_landing'. The main query editor contains the SQL statement: `1 select * from accelerometer_landing;`. The query status is 'Completed' with a 'Time in queue' of 109 ms, 'Run time' of 1.06 sec, and 'Data scanned' of 6.55 MB. The results section shows 81,273 rows.

Below the first screenshot, a second screenshot shows the same interface with a different query: `13 select count(*) as total from accelerometer_landing;`. The query status is 'Completed' with a 'Time in queue' of 112 ms, 'Run time' of 500 ms, and 'Data scanned' of 6.55 MB. The results section shows 1 row with the total count of 81273.

| # | total |
|---|-------|
| 1 | 81273 |

3.step_trainer_landing

The top screenshot shows the Amazon Athena Query Editor interface. The left sidebar displays the 'Data source' as 'AwsDataCatalog', 'Catalogue' as 'None', and 'Database' as 'my_db'. Under 'Tables and views', there are three tables: 'accelerometer_landing', 'customer_landing', and 'step_trainer_landing'. The main query editor shows the SQL query: `1 select * from step_trainer_landing;`. The query status is 'Completed' with a 'Time in queue' of 112 ms, 'Run time' of 688 ms, and 'Data scanned' of 3.15 MB. The results show 28,680 rows.

The bottom screenshot shows the same interface with a different query: `13 select count(*) as total from step_trainer_landing;`. The query status is 'Completed' with a 'Time in queue' of 84 ms, 'Run time' of 655 ms, and 'Data scanned' of 3.15 MB. The results show 1 row with a total of 28,680.

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

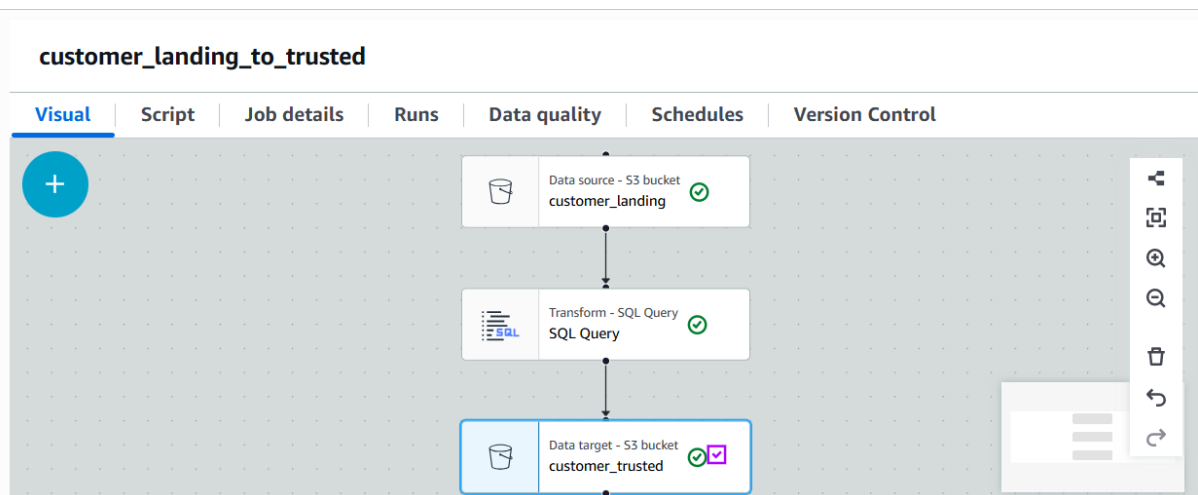
's3://stedi-uda-project/landing/step_trainer_landing/'

TBLPROPERTIES (

'classification'='json')

GLUE STUDIO OUTPUTS

1.customer_landing_to_trusted



Customer_trusted

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/48c6e2ca-be87-4a7a-96e5-5c3042c560a7

Amazon Athena > Query editor tabs

Database: my_db

Tables and views: [Create](#) [Filter tables and views](#)

Tables (4): accelerometer_landing, customer_landing, customer_trusted, step_trainer_landing

Views (0)

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

SQL Ln 11, Col 1

[Run](#) [Explain](#) [Cancel](#) [Clear](#) [Create](#)

[Reuse query results](#) up to 60 minutes ago

Query results | Query status

Completed Time in queue: 118 ms Run time: 594 ms Data scanned: 161.06 KB

Results (482) [Copy](#) [Download results CSV](#)

Search rows

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

CloudShell Feedback

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/93cf87f6-c341-4348-83ad-b7953dab673c

Amazon Athena > Query editor tabs

Database: my_db

Tables and views: [Create](#) [Filter tables and views](#)

Tables (4): accelerometer_landing, customer_landing, customer_trusted, step_trainer_landing

Views (0)

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

SQL Ln 10, Col 25

[Run again](#) [Explain](#) [Cancel](#) [Clear](#) [Create](#)

[Reuse query results](#) up to 60 minutes ago

Query results | Query status

Completed Time in queue: 123 ms Run time: 707 ms Data scanned: 161.06 KB

Results (1) [Copy](#) [Download results CSV](#)

Search rows

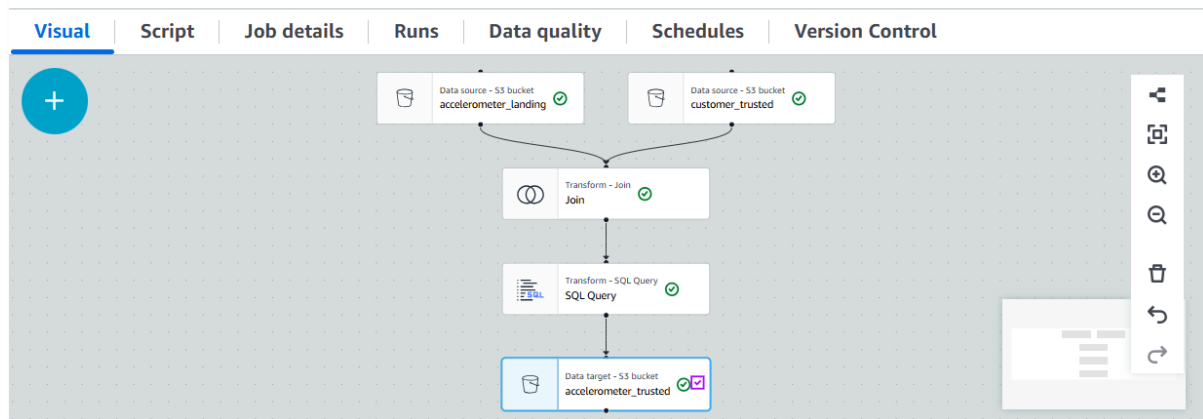
| # | total |
|---|-------|
| 1 | 482 |

CloudShell Feedback

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

2.accelerometer_landing_to_trusted

accelerometer_landing_to_trusted



us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/af4e2819-62bf-417d-9957-d6bd48f864f4

Amazon Athena > Query editor tabs

Database: my_db

Tables and views: Create

Tables (5): accelerometer_landing, accelerometer_trusted, customer_landing, customer_trusted, step_trainer_landing

Views (0)

SQL: `select * from accelerometer_trusted;`

Run again Explain Cancel Clear Create

Query results: Completed Time in queue: 108 ms Run time: 902 ms Data scanned: 3.30 MB

Results (40,981)

| # | x | y | z | user | timestamp |
|---|-----|-----|------|-------------------------|---------------|
| 1 | 1.0 | 0.0 | -1.0 | Neeraj.Sanchez@test.com | 1655564422163 |
| 2 | 1.0 | 0.0 | -1.0 | Neeraj.Sanchez@test.com | 1655564411363 |

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/20a0b8e8-8e7f-4e1c-a3a2-4ddd5cdc541

Amazon Athena > Query editor tabs

Database: my_db

Tables and views: Create

Tables (5): accelerometer_landing, accelerometer_trusted, customer_landing, customer_trusted, step_trainer_landing

Views (0)

SQL: `select count(*) as total from accelerometer_trusted;`

Run again Explain Cancel Clear Create

Query results: Completed Time in queue: 107 ms Run time: 927 ms Data scanned: 3.30 MB

Results (1)

| # | total |
|---|-------|
| 1 | 40981 |

3.customer curated

customer_curated

Visual | Script | Job details | Runs | Data quality | Schedules | Version Control

The workflow diagram illustrates the process of creating the 'customer_curated' table. It begins with two data sources: 'Data source - S3 bucket customer_trusted' and 'Data source - S3 bucket accelerometer_trusted'. Both sources feed into a 'Transform - SQL Query' step. This step then feeds into another 'Transform - SQL Query' step, which finally feeds into the 'Data target - S3 bucket customer_curated'.

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/1333704c-cbca-46ef-a97c-dfe0fcb65cc9

Amazon Athena > Query editor tabs

Database: my_db

Tables and views: Create

Filter tables and views

Tables (6): accelerometer_landing, accelerometer_trusted, customer_curated, customer_landing, customer_trusted, step_trainer_landing

Views (0)

SQL: select * from customer_curated;

Run | Explain | Cancel | Clear | Create

Query results | Query status

Completed | Time in queue: 109 ms | Run time: 716 ms | Data scanned: 161.03 KB

Results (482) | Copy | Download results CSV

| # | serialnumber | sharewithpublicasofdate | birthday | registrationdate | sharewithresearchasofdate |
|---|--------------------------------------|-------------------------|------------|------------------|---------------------------|
| 1 | 20e0e292-48de-41a2-bfc1-821972e1334b | 1655563976992 | 1175-01-01 | 1655563976992 | 1655563976992 |
| 2 | dd192d35-6441-4a70-9c34-5fdc22a9b396 | 1655564441612 | 1032-01-01 | 1655564441612 | 1655564441612 |

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy | Terms | Cookie preferences

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/a4fb8f61-963a-4b0c-b8a6-dc0aef9abb28

Amazon Athena > Query editor tabs

Database: my_db

Tables and views: Create Filter tables and views

Tables (6): accelerometer_landing, accelerometer_trusted, customer_curated, customer_landing, customer_trusted, step_trainer_landing

Views (0)

SQL: select count(*) from customer_curated;

Run again Explain Cancel Clear Create

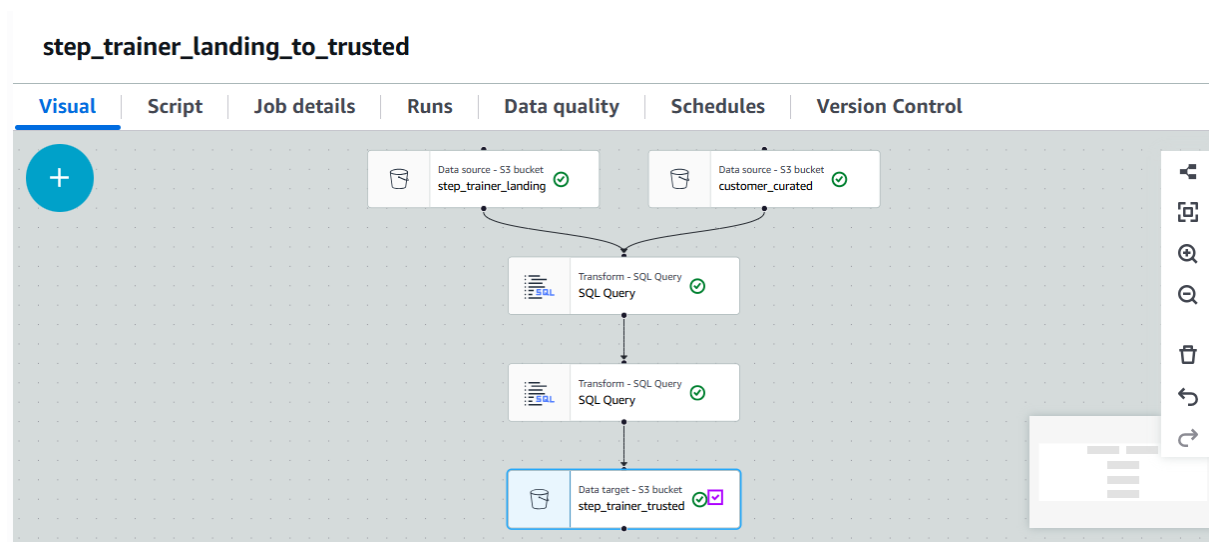
Query results: Completed Time in queue: 459 ms Run time: 644 ms Data scanned: 161.03 KB

Results (1): Search rows

| # | _col0 |
|---|-------|
| 1 | 482 |

CloudShell Feedback

4.step_trainer_trusted



us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/70232a90-45cd-4776-a659-a3aac9d7a922

Amazon Athena > Query editor tabs

Database: my_db

Tables and views: [Create](#)

Filter tables and views

Tables (7): accelerometer_landing, accelerometer_trusted, customer_curated, customer_landing, customer_trusted, step_trainer_landing, step_trainer_trusted

Views (0)

SQL: `select * from step_trainer_trusted;`

Run again Explain Cancel Clear Create

Query results Query status

Completed Time in queue: 109 ms Run time: 687 ms Data scanned: 1.59 MB

Results (14,460) [Copy](#) [Download results CSV](#)

Search rows

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

CloudShell Feedback

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/26922cef-15ce-40f7-b6c7-36f89f039893

Amazon Athena > Query editor tabs

Database: my_db

Tables and views: [Create](#)

Filter tables and views

Tables (7): accelerometer_landing, accelerometer_trusted, customer_curated, customer_landing, customer_trusted, step_trainer_landing, step_trainer_trusted

Views (0)

SQL: `select count(*) as total from step_trainer_trusted;`

Run again Explain Cancel Clear Create

Query results Query status

Completed Time in queue: 120 ms Run time: 500 ms Data scanned: 1.59 MB

Results (1) [Copy](#) [Download results CSV](#)

Search rows

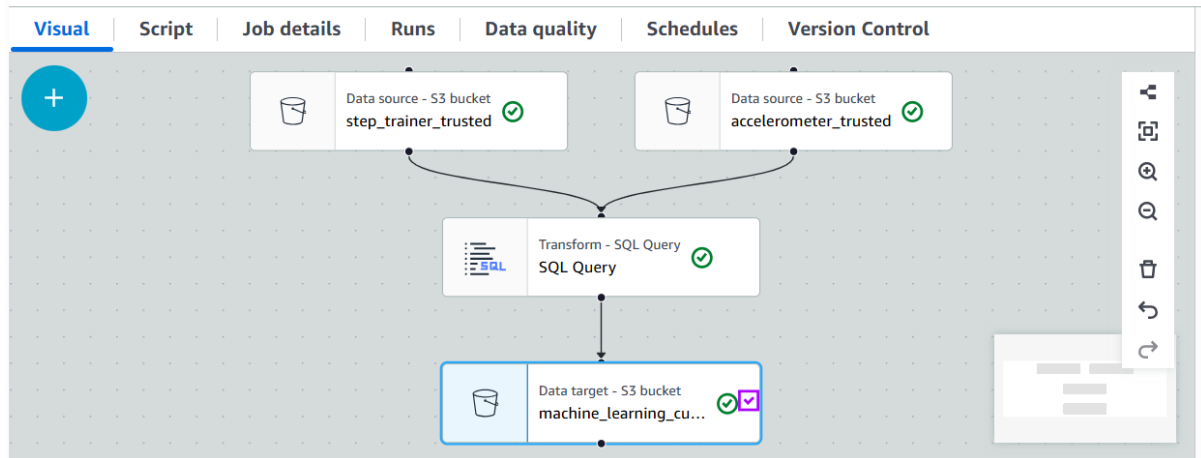
| # | total |
|---|-------|
| 1 | 14460 |

CloudShell Feedback

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

5.machine_learning_curated

machine_learning_curated



The first screenshot shows the Amazon Athena console with a SQL query editor. The query is: `select * from machine_learning_curated;`. The query status is 'Completed' with a time in queue of 113 ms, a run time of 1.117 sec, and data scanned of 8.23 MB. The results are displayed in a table with 43,681 rows. The table has columns: #, x, y, z, user, timestamp, sensorreadingtime, and serialnumber. The results show two rows of data.

| # | x | y | z | user | timestamp | sensorreadingtime | serialnumber |
|---|-----|-----|------|------------------------|---------------|-------------------|-------------------------------------|
| 1 | 1.0 | 1.0 | 0.0 | Danny.Lincoln@test.com | 1655561962765 | 1655561962765 | 7808965b-6444-4261-9e03-47ee0244e1f |
| 2 | 1.0 | 0.0 | -1.0 | Danny.Lincoln@test.com | 1655561962765 | 1655561962765 | 7808965b-6444-4261-9e03-47ee0244e1f |

The second screenshot shows the Amazon Athena console with a different SQL query. The query is: `select count(*) as total from machine_learning_curated;`. The query status is 'Completed' with a time in queue of 145 ms, a run time of 466 ms, and data scanned of 8.23 MB. The results are displayed in a table with 1 row. The table has columns: # and total. The results show one row with a total of 43681.

| # | total |
|---|-------|
| 1 | 43681 |

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/"], "recurse": 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",catalogTableNam
e="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/"], "recurse": 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",catalogTable
Name="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/"], "recurse": 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",catalogTableNam
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",catalogTableN  
ame="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/"], "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", catalogTableN  
ame="step_trainer_trusted")
```

```
step_trainer_trusted_node1768973796390.setFormat("json")
```

```
step_trainer_trusted_node1768973796390.writeFrame(SQLQuery_node1768973792561)
```

```
job.commit()
```