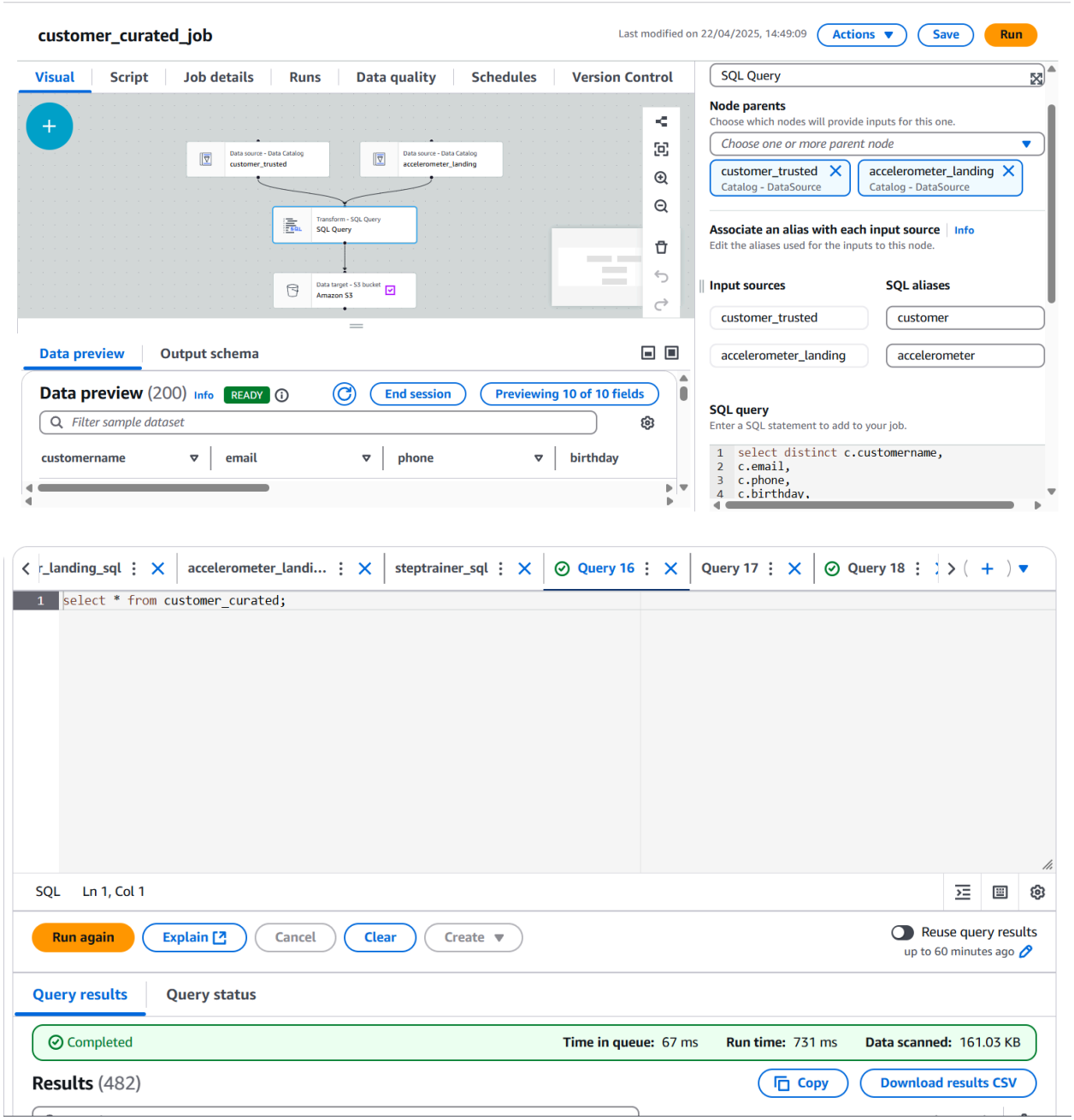
The Data Science team would like you to write a Glue job that does the following:

- Sanitize the Customer data (Trusted Zone) and create a Glue Table (Curated Zone) that only includes customers who have accelerometer data *and* have agreed to share their data for research called **customers_curated**.

Finally, you need to create two Glue Studio jobs that do the following tasks:

- Read the Step Trainer IoT data stream (S3) and populate a Trusted Zone Glue Table called **step_trainer_trusted** that contains the Step Trainer Records data for customers who have accelerometer data and have agreed to share their data for research (customers_curated).
- Create an aggregated table that has each of the Step Trainer Readings, and the associated accelerometer reading data for the same timestamp, but only for customers who have agreed to share their data, and make a glue table called **machine_learning_curated.**

Customer_curated


customer_curated_job
Last modified on 22/04/2025, 14:49:09
Actions
Save
Run
Visual
Script
Job details
Runs
Data quality
Schedules
Version Control
Data source - Data Catalog
customer_trusted
Data source - Data Catalog
accelerometer_landing
Transform - SQL Query
SQL Query
Data target - S3 bucket
Amazon S3
Data preview
Output schema
Data preview (200) Info
READY
End session
Previewing 10 of 10 fields
Filter sample dataset
customername
email
phone
birthday
SQL Query
Node parents
Choose which nodes will provide inputs for this one.
Choose one or more parent node
customer_trusted
Catalog - DataSource
accelerometer_landing
Catalog - DataSource
Associate an alias with each input source
Info
Edit the aliases used for the inputs to this node.
Input sources
SQL aliases
customer_trusted
customer
accelerometer_landing
accelerometer
SQL query
Enter a SQL statement to add to your job.
1 select distinct c.customername,
2 c.email,
3 c.phone,
4 c.birthday,

r_landing_sql
accelerometer_landi...
steptrainer_sql
Query 16
Query 17
Query 18
1 select * from customer_curated;
SQL
Ln 1, Col 1
Run again
Explain
Cancel
Clear
Create
Reuse query results
up to 60 minutes ago
Query results
Query status
Completed
Time in queue: 67 ms
Run time: 731 ms
Data scanned: 161.03 KB
Results (482)
Copy
Download results CSV

Query results | Query status

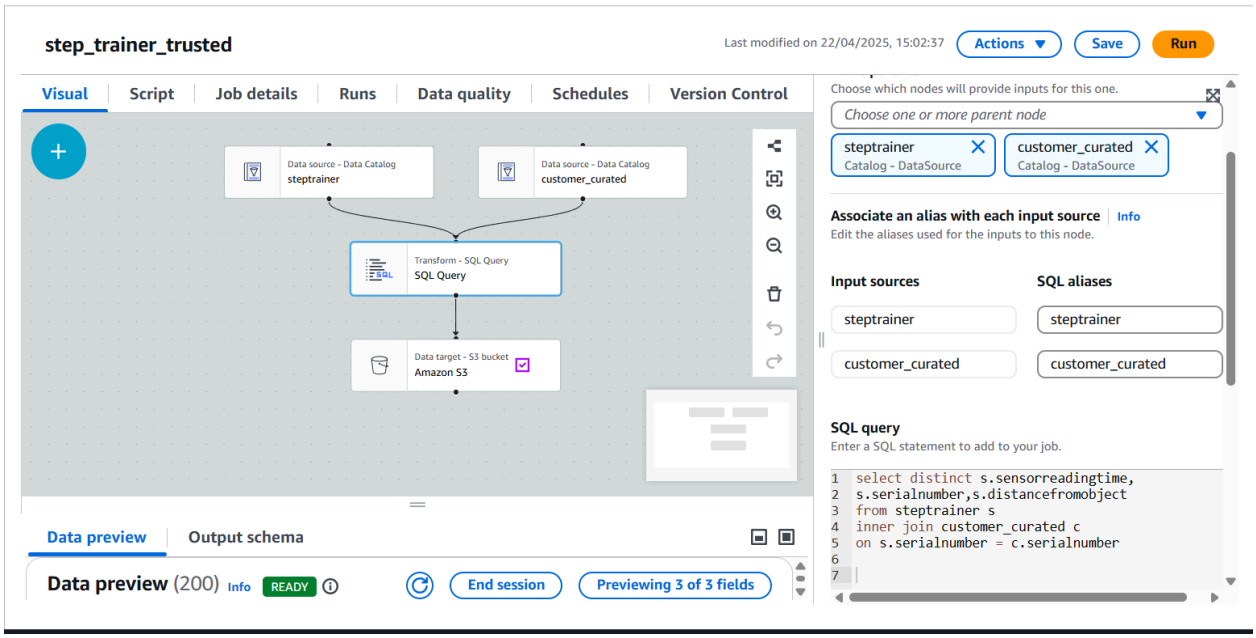
Completed — Time in queue: 76 ms — Run time: 1.203 sec — Data scanned: 161.03 KB

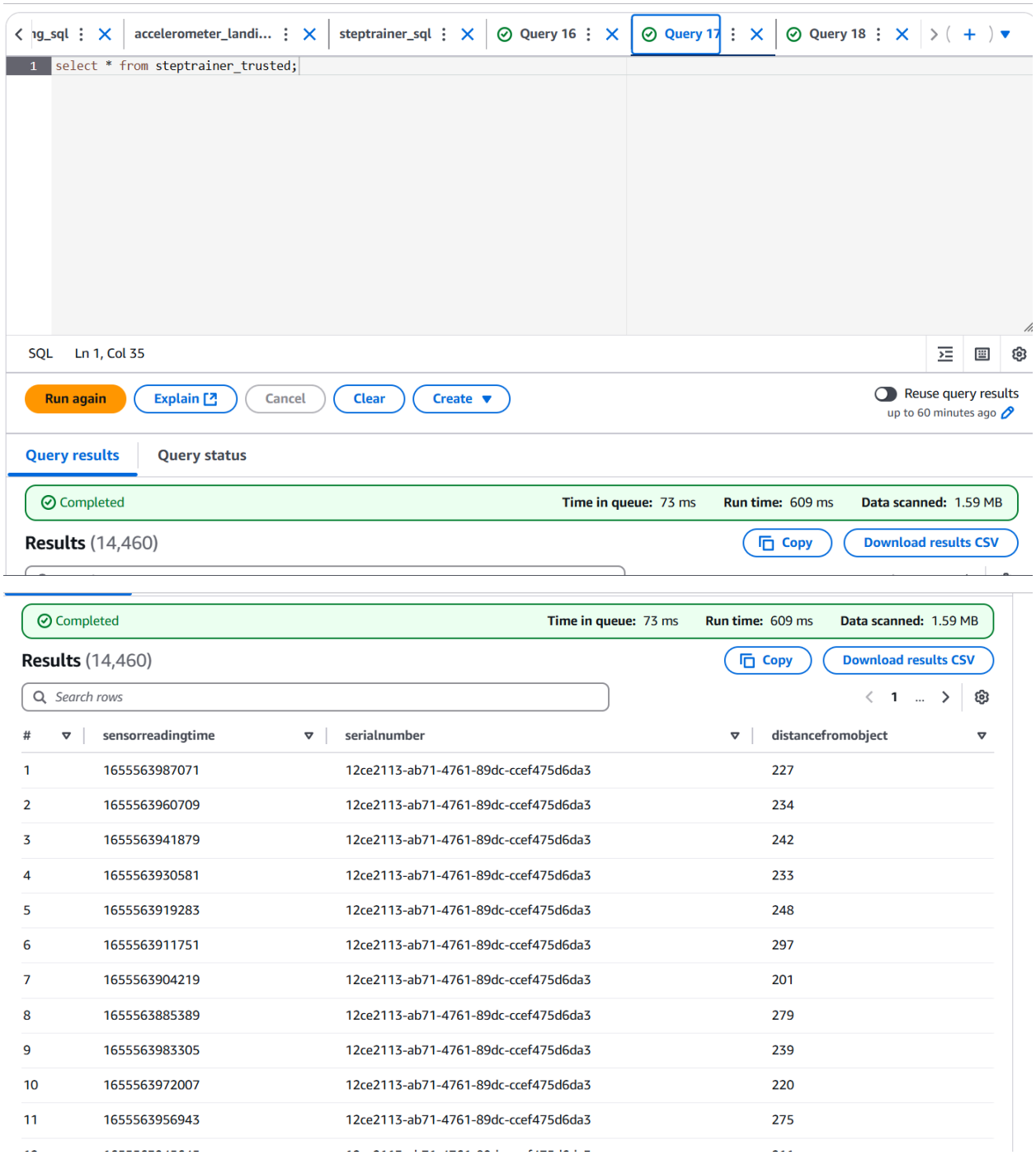
Results (482) — Copy — Download results CSV

Search rows — 1 ...

#	customername	email	phone	birthday	serialnumber	registrationdate	lastupdatec
1	Angie Abram	Angie.Abram@test.com	8015551212	1539-01-01	0d30f1a9-7492-4cc1-95df-d9f5a159ed92	1655564403947	165556440
2	Bobby Davis	Bobby.Davis@test.com	8015551212	1138-01-01	db0fbb25-f82c-428c-823a-35274be4f259	1655564433827	165556443
3	Christina Mitra	Christina.Mitra@test.com	8015551212	1160-01-01	7414a106-ee97-4824-ab25-70bedeae1136	1655564432243	165556443
4	Craig Jefferson	Craig.Jefferson@test.com	8015551212	1666-01-01	f8e09680-6e0c-46b4-b7a7-eacd8372b079	1655564394456	165556439
5	Edward Abram	Edward.Abram@test.com	8015551212	1435-01-01	31cc254a-8982-4ace-a538-a556fdc2fec6	1655564411759	165556441
6	Eric Spencer	Eric.Spencer@test.com	8015551212	1002-01-01	9d2f683d-7588-41e9-b5af-704a191b2cec	1655564443889	165556444
7	John Wu	John.Wu@test.com	8015551212	1938-01-01	70cdc373-146e-4de9-80b3-941129bc7e42	1655564373109	165556437
8	Sarah Khatib	Sarah.Khatib@test.com	8015551212	1097-01-01	be0294fc-67e9-48dc-aa8a-3015fdcfefcb	1655564436771	165556443
9	Suresh Jefferson	Suresh.Jefferson@test.com	8015551212	1219-01-01	8dc7645e-9700-4304-bba0-c5818b438806	1655564138779	165556413
10	Travis Abram	Travis.Abram@test.com	8015551212	1374-01-01	e2d30dec-120a-4f37-b571-7c27f49cbde8	1655562436181	165556249
11	Travis Clayton	Travis.Clayton@test.com	8015551212	988-01-01	c87d2224-b0f1-4392-9182-f9f7dcfdf45c	1655564444965	165556444
12	Trevor Ahmed	Trevor.Ahmed@test.com	8015551212	1654-01-01	13ee02ac-37d3-40d2-af66-63c43d6ca079	1655564395318	165556439

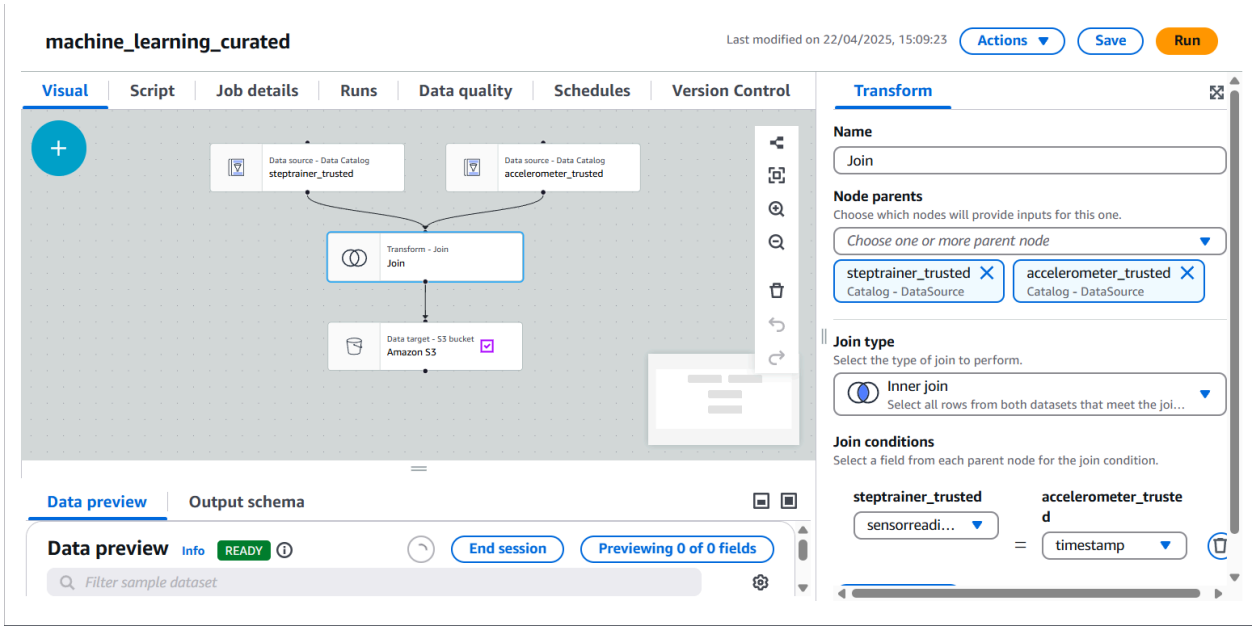
Steptrainer trusted:

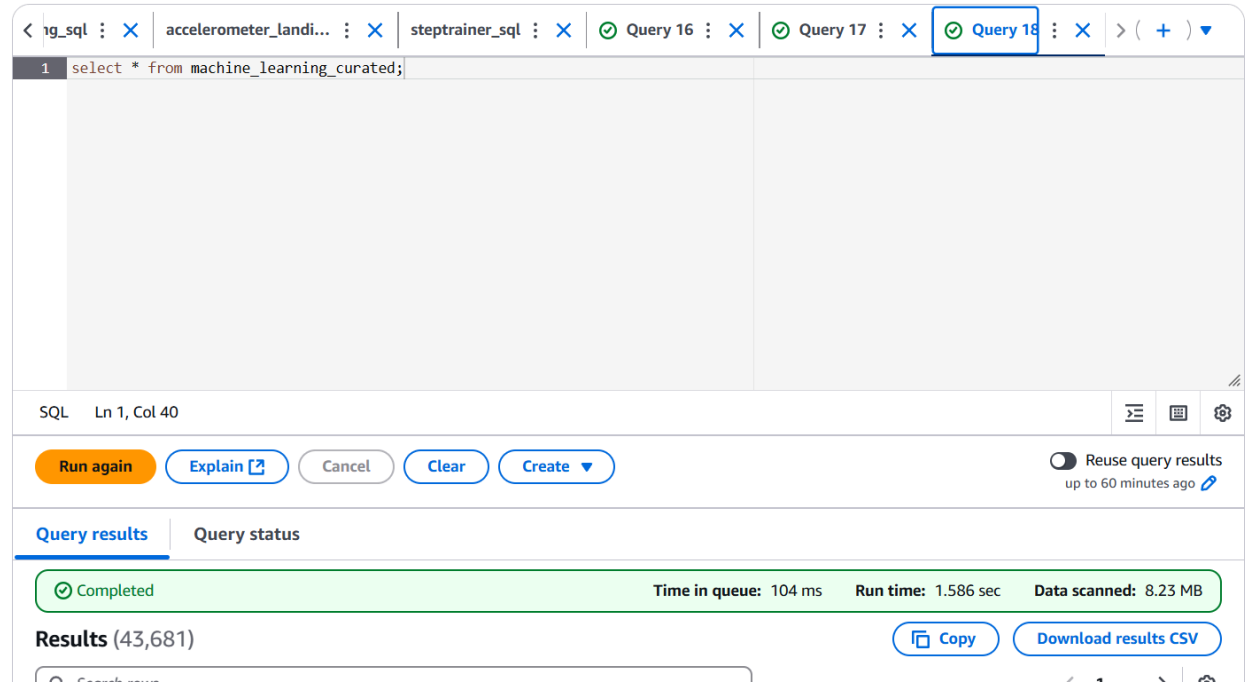


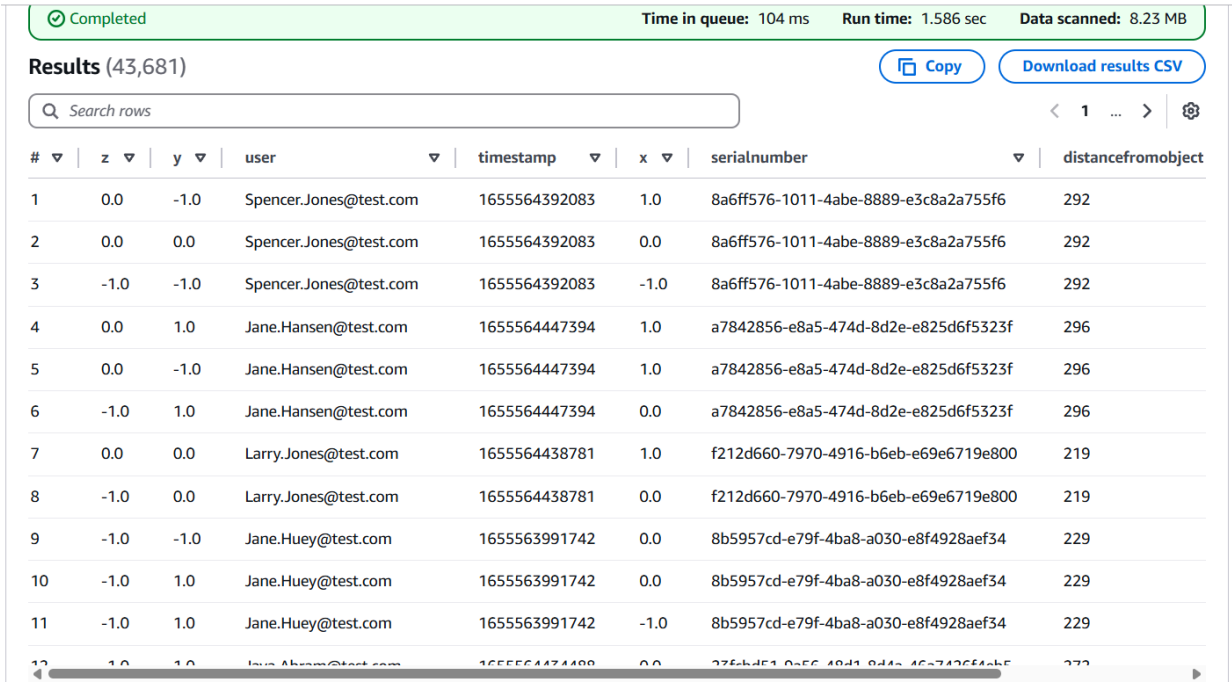


#	sensorreadingtime	serialnumber	distancefromobject
1	1655563987071	12ce2113-ab71-4761-89dc-ccef475d6da3	227
2	1655563960709	12ce2113-ab71-4761-89dc-ccef475d6da3	234
3	1655563941879	12ce2113-ab71-4761-89dc-ccef475d6da3	242
4	1655563930581	12ce2113-ab71-4761-89dc-ccef475d6da3	233
5	1655563919283	12ce2113-ab71-4761-89dc-ccef475d6da3	248
6	1655563911751	12ce2113-ab71-4761-89dc-ccef475d6da3	297
7	1655563904219	12ce2113-ab71-4761-89dc-ccef475d6da3	201
8	1655563885389	12ce2113-ab71-4761-89dc-ccef475d6da3	279
9	1655563983305	12ce2113-ab71-4761-89dc-ccef475d6da3	239
10	1655563972007	12ce2113-ab71-4761-89dc-ccef475d6da3	220
11	1655563956943	12ce2113-ab71-4761-89dc-ccef475d6da3	275

Machinelearningcurated:


machine_learning_curated
Last modified on 22/04/2025, 15:09:23
Actions
Save
Run
Visual
Script
Job details
Runs
Data quality
Schedules
Version Control
Data source - Data Catalog
steptrainer_trusted
Data source - Data Catalog
accelerometer_trusted
Transform - Join
Join
Data target - S3 bucket
Amazon S3
Transform
Name
Join
Node parents
Choose which nodes will provide inputs for this one.
Choose one or more parent node
steptrainer_trusted
Catalog - DataSource
accelerometer_trusted
Catalog - DataSource
Join type
Select the type of join to perform.
Inner join
Select all rows from both datasets that meet the joi...
Join conditions
Select a field from each parent node for the join condition.
steptrainer_trusted
sensorreadi...
=
accelerometer_trusted
timestamp
Data preview
Output schema
Data preview
Info
READY
End session
Previewing 0 of 0 fields
Filter sample dataset


ıg_sql
accelerometer_landi...
steptrainer_sql
Query 16
Query 17
Query 18
1
select * from machine_learning_curated;
SQL
Ln 1, Col 40
Run again
Explain
Cancel
Clear
Create
Reuse query results
up to 60 minutes ago
Query results
Query status
Completed
Time in queue: 104 ms
Run time: 1.586 sec
Data scanned: 8.23 MB
Results (43,681)
Copy
Download results CSV



Completed | Time in queue: 104 ms | Run time: 1.586 sec | Data scanned: 8.23 MB

Results (43,681) — Copy | Download results CSV

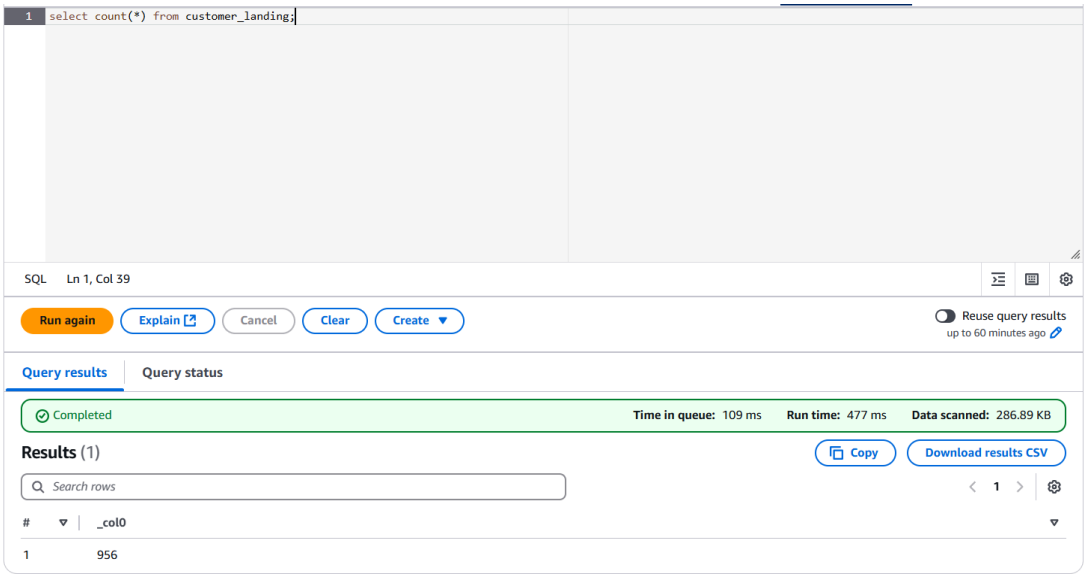
Search rows — 1 ...

#	z	y	user	timestamp	x	serialnumber	distancefromobject
1	0.0	-1.0	Spencer.Jones@test.com	1655564392083	1.0	8a6ff576-1011-4abe-8889-e3c8a2a755f6	292
2	0.0	0.0	Spencer.Jones@test.com	1655564392083	0.0	8a6ff576-1011-4abe-8889-e3c8a2a755f6	292
3	-1.0	-1.0	Spencer.Jones@test.com	1655564392083	-1.0	8a6ff576-1011-4abe-8889-e3c8a2a755f6	292
4	0.0	1.0	Jane.Hansen@test.com	1655564447394	1.0	a7842856-e8a5-474d-8d2e-e825d6f5323f	296
5	0.0	-1.0	Jane.Hansen@test.com	1655564447394	1.0	a7842856-e8a5-474d-8d2e-e825d6f5323f	296
6	-1.0	1.0	Jane.Hansen@test.com	1655564447394	0.0	a7842856-e8a5-474d-8d2e-e825d6f5323f	296
7	0.0	0.0	Larry.Jones@test.com	1655564438781	1.0	f212d660-7970-4916-b6eb-e69e6719e800	219
8	-1.0	0.0	Larry.Jones@test.com	1655564438781	0.0	f212d660-7970-4916-b6eb-e69e6719e800	219
9	-1.0	-1.0	Jane.Huey@test.com	1655563991742	0.0	8b5957cd-e79f-4ba8-a030-e8f4928aef34	229
10	-1.0	1.0	Jane.Huey@test.com	1655563991742	0.0	8b5957cd-e79f-4ba8-a030-e8f4928aef34	229
11	-1.0	1.0	Jane.Huey@test.com	1655563991742	-1.0	8b5957cd-e79f-4ba8-a030-e8f4928aef34	229

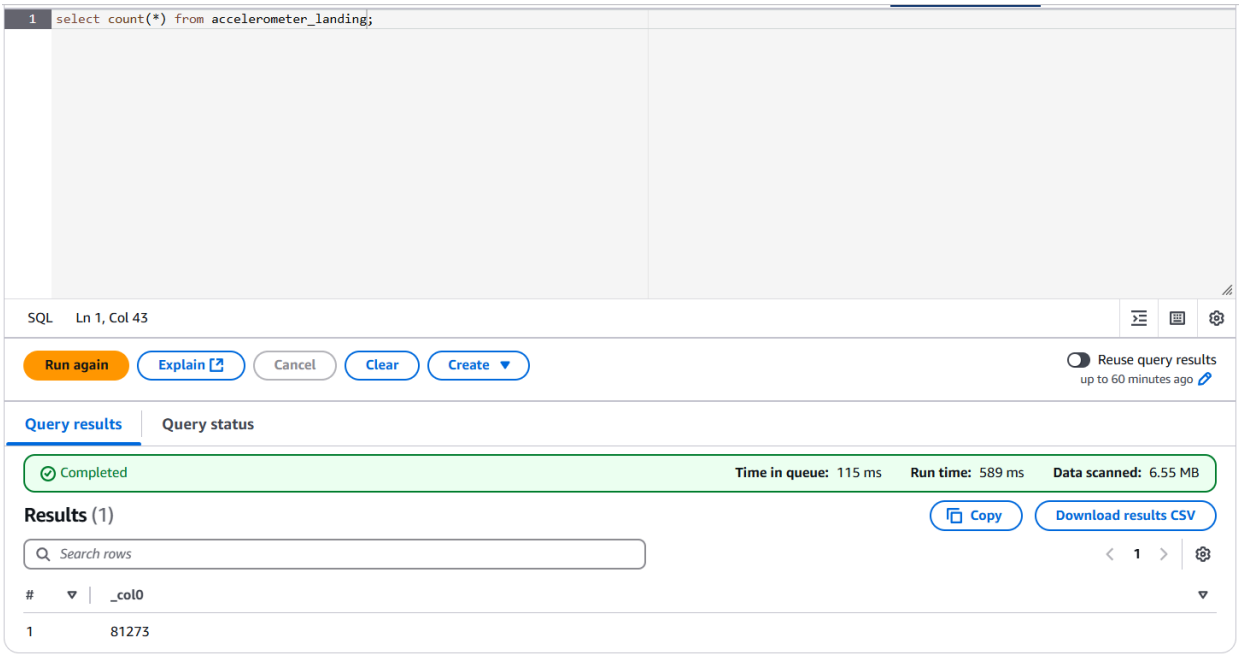
Counts :

- Landing
  - Customer: 956
  - Accelerometer: 81273
  - Step Trainer: 28680

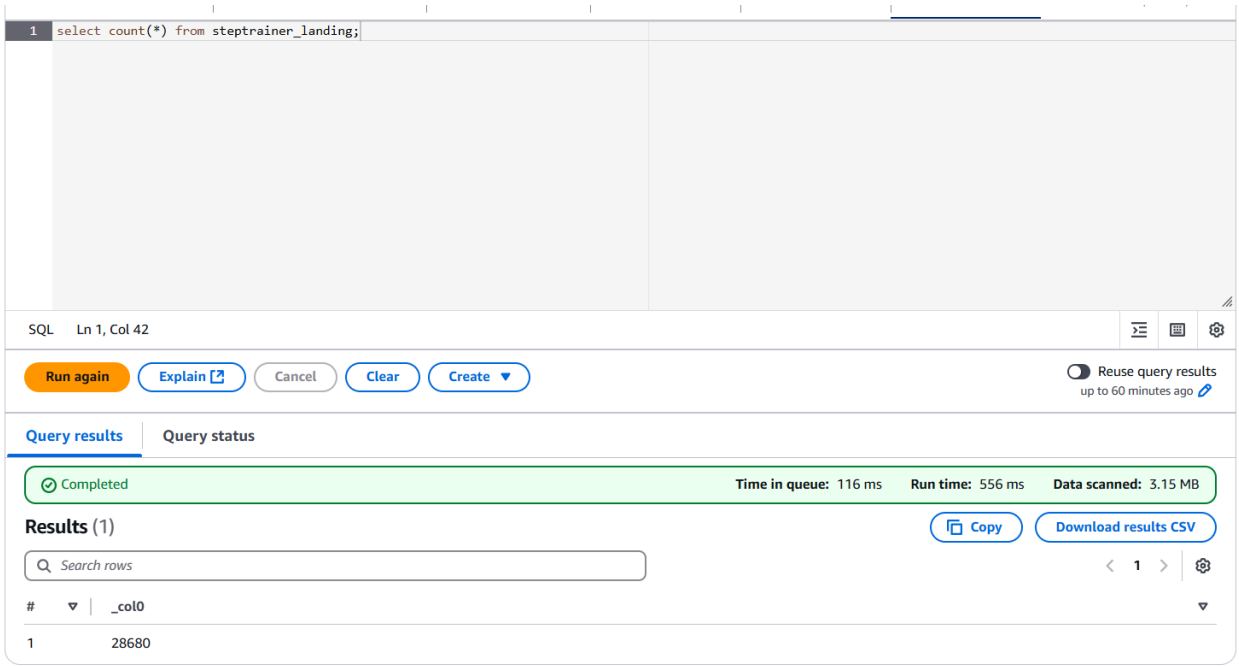
Customer_landing



Accelerometer_landing

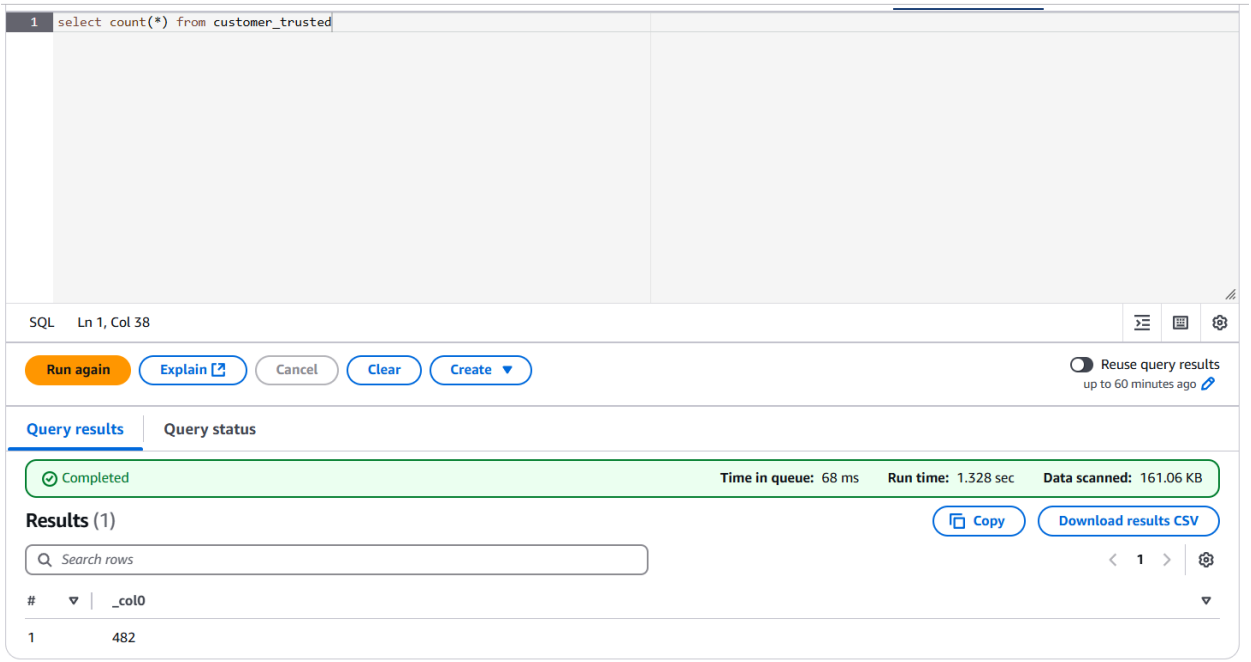


Steptrainer_landing

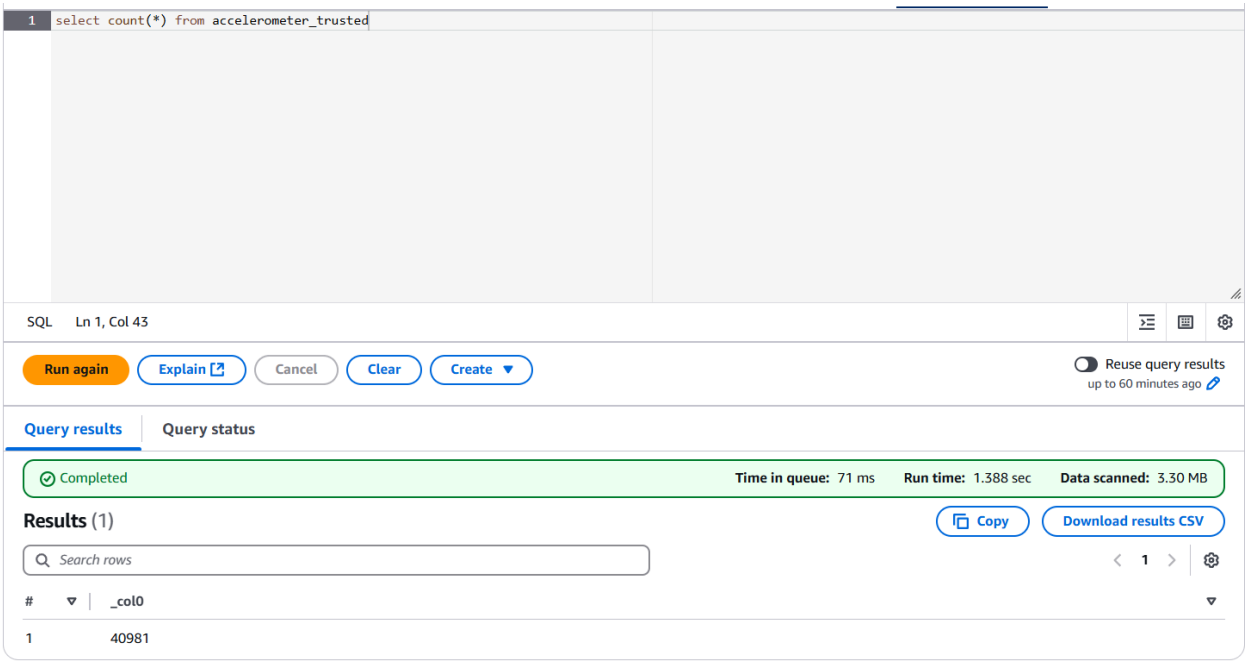


- Trusted
    - Customer: 482
    - Accelerometer: 40981
    - Step Trainer: 14460

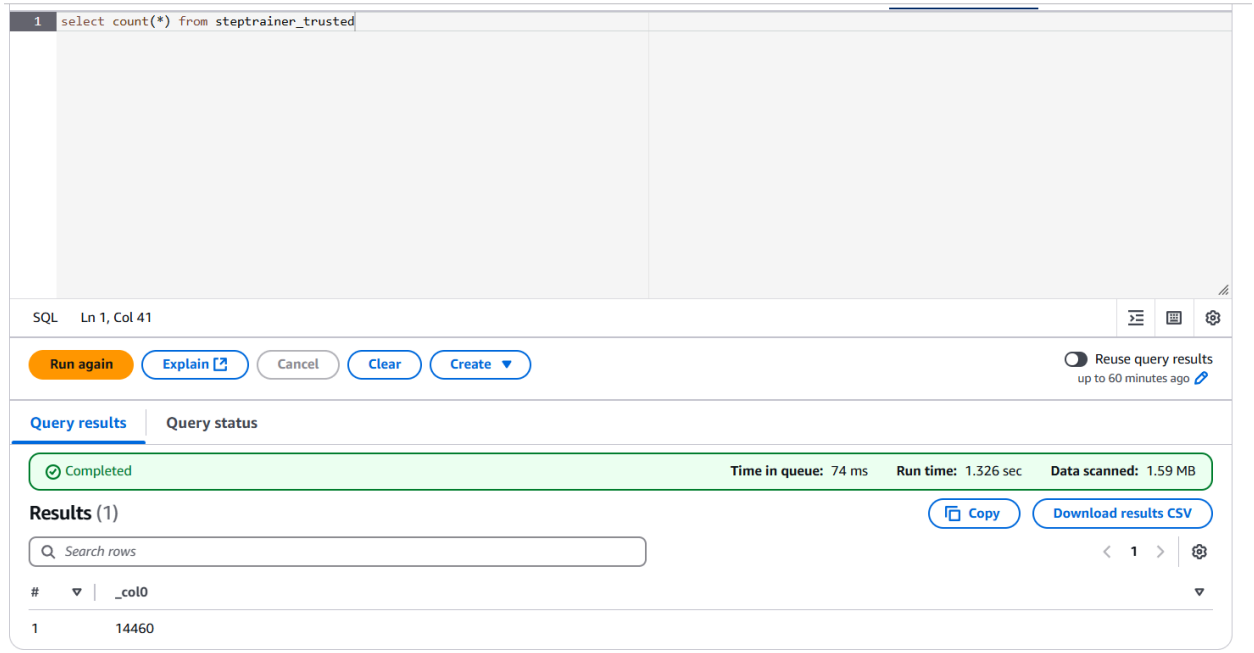
customer_trusted



Accelerometer_trusted

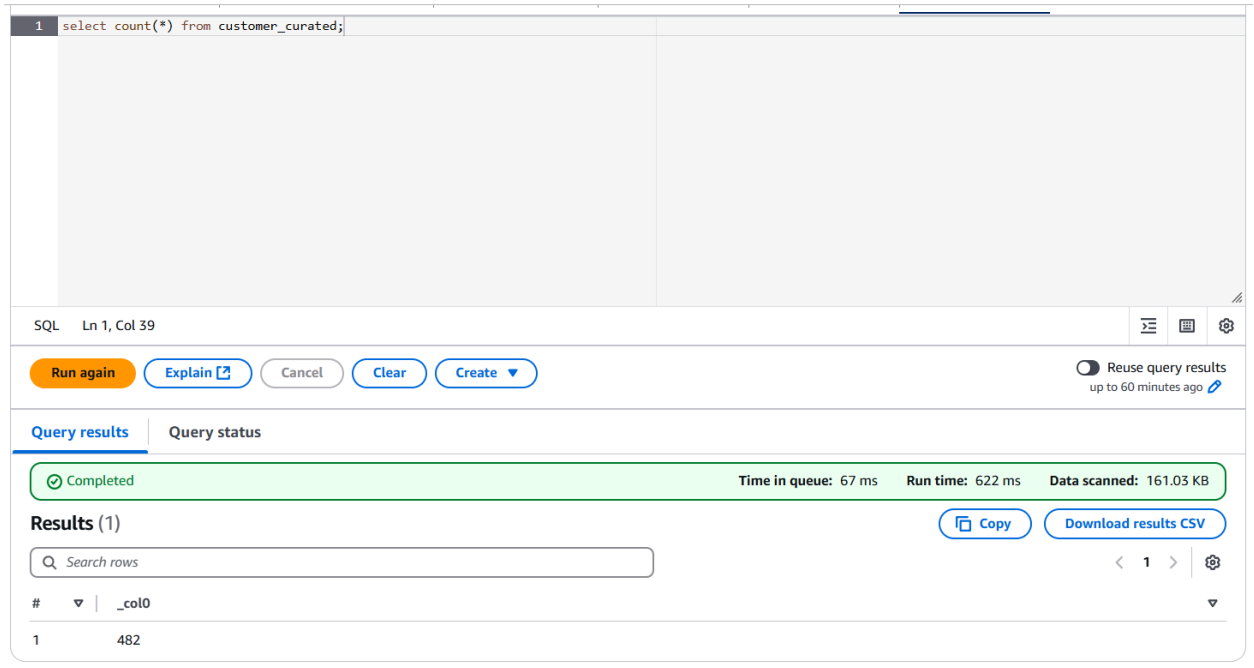


Steptrainer_trusted

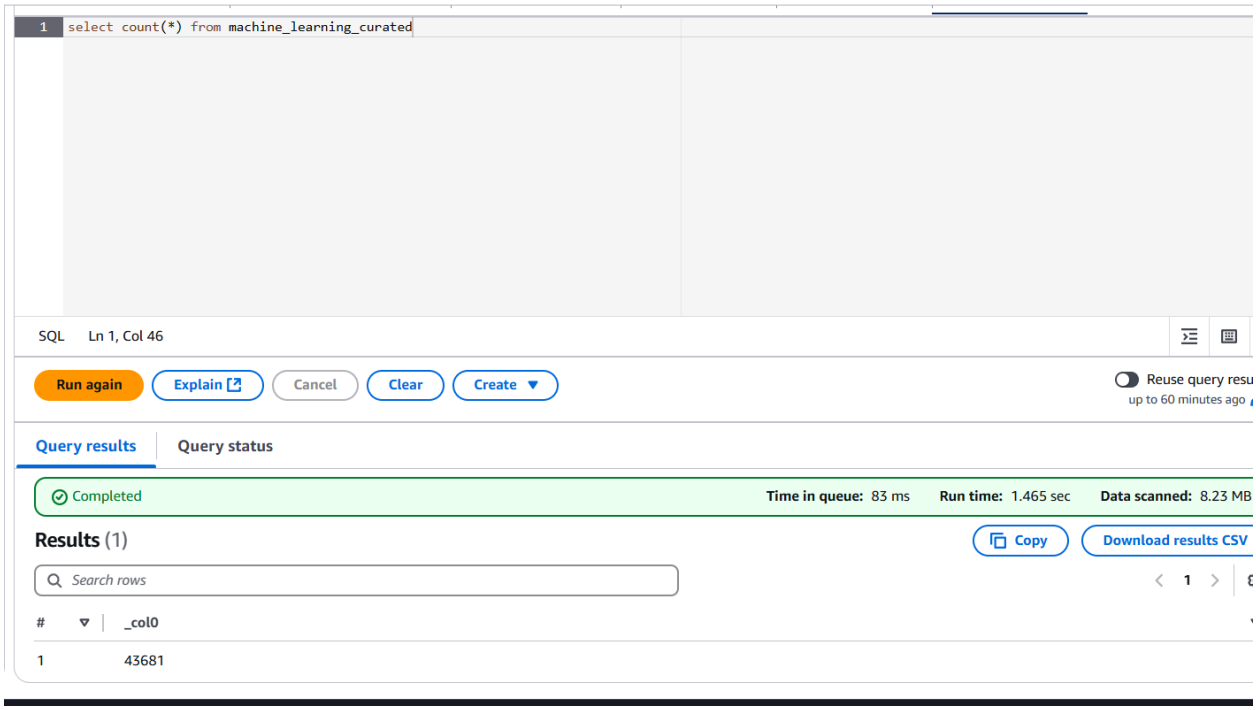


- Curated
    - Customer: 482
    - Machine Learning: 43681

Customer_curated:



Machinelearing_curated



Not clear whether to paste script here or not : it is mention to upload in github

So pasting here :

Customer_landing_to_trusted:

```python
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
    ]
```

Accelerometer_landing_to_trusted :

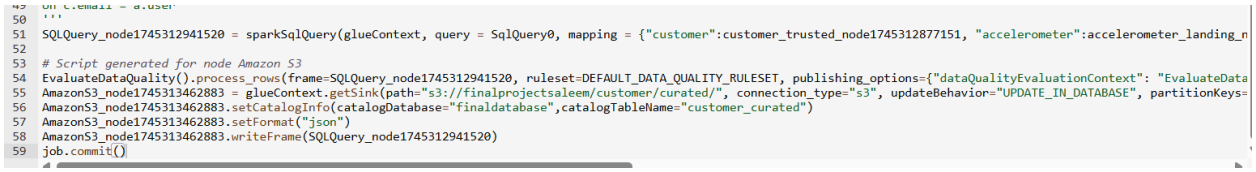
```python
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsgluedq.transforms import EvaluateDataQuality

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 CustomertrustedData
CustomertrustedData_node1745301464483 = glueContext.create_dynamic_frame.from_catalog(database="finaldatabase", table_name="customer_trusted", transformation_ctx="CustomertrustedData_node1745301464483")

# Script generated for node AccelerometerData
```

```python
# Script generated for node AccelerometerData
AccelerometerData_node1745301165475 = glueContext.create_dynamic_frame.from_catalog(database="finaldatabase", table_name="accelerometer_landing", transformation_ctx="AccelerometerData_node1745301165475")

# Script generated for node Join
Join_node1745301213533 = Join.apply(frame1=AccelerometerData_node1745301165475, frame2=CustomertrustedData_node1745301464483, keys1=["user"], keys2=["email"], transformation_ctx="Join_node1745301213533")

# Script generated for node Select Fields
SelectFields_node1745301261258 = SelectFields.apply(frame=Join_node1745301213533, paths=["z", "user", "y", "x", "timestamp"], transformation_ctx="SelectFields_node1745301261258")

# Script generated for node Amazon S3
EvaluateDataQuality().process_rows(frame=SelectFields_node1745301261258, ruleset=DEFAULT_DATA_QUALITY_RULESET, publishing_options={"dataQualityEvaluationContext": "EvaluateDataQuality_node1745300140421", "ena
AmazonS3_node1745301294820 = glueContext.getSink(path="s3://finalprojectsaleem/accelerometer/trusted/", connection_type="s3", updateBehavior="UPDATE_IN_DATABASE", partitionKeys=[], enableUpdateCatalog=True, t
AmazonS3_node1745301294820.setCatalogInfo(catalogDatabase="finaldatabase",catalogTableName="accelerometer_trusted")
AmazonS3_node1745301294820.setFormat("json")
AmazonS3_node1745301294820.writeFrame(SelectFields_node1745301261258)
job.commit()
```

Customer_curated :

```
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_node1745312916301 = glueContext.create_dynamic_frame.from_catalog(database="finaldatabase", table_name="accelerometer_landing", transformation_ctx="ac

# Script generated for node customer_trusted
customer_trusted_node1745312877151 = glueContext.create_dynamic_frame.from_catalog(database="finaldatabase", table_name="customer_trusted", transformation_ctx="customer_tru

# Script generated for node SQL Query
SqlQuery0 = '''
select distinct c.customername,
c.email,
c.phone,
c.birthday,
c.serialnumber,
c.registrationdate,
c.lastupdatedate,
c.sharewithresearchasofdate,
c.sharewithpublicasofdate,
c.sharewithfriendsasofdate
from customer c
join accelerometer a
on c.email = a.user
'''
SQLQuery_node1745312941520 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"customer":customer_trusted_node1745312877151, "accelerometer":accelerometer_landing_n
```



```
'''
SQLQuery_node1745312941520 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"customer":customer_trusted_node1745312877151, "accelerometer":accelerometer_landing_n

# Script generated for node Amazon S3
EvaluateDataQuality().process_rows(frame=SQLQuery_node1745312941520, ruleset=DEFAULT_DATA_QUALITY_RULESET, publishing_options={"dataQualityEvaluationContext": "EvaluateData
AmazonS3_node1745313462883 = glueContext.getSink(path="s3://finalprojectsaleem/customer/curated/", connection_type="s3", updateBehavior="UPDATE_IN_DATABASE", partitionKeys=
AmazonS3_node1745313462883.setCatalogInfo(catalogDatabase="finaldatabase",catalogTableName="customer_curated")
AmazonS3_node1745313462883.setFormat("json")
AmazonS3_node1745313462883.writeFrame(SQLQuery_node1745312941520)
job.commit()
```

Steptrainer_trusted:

```python
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
    ]
```

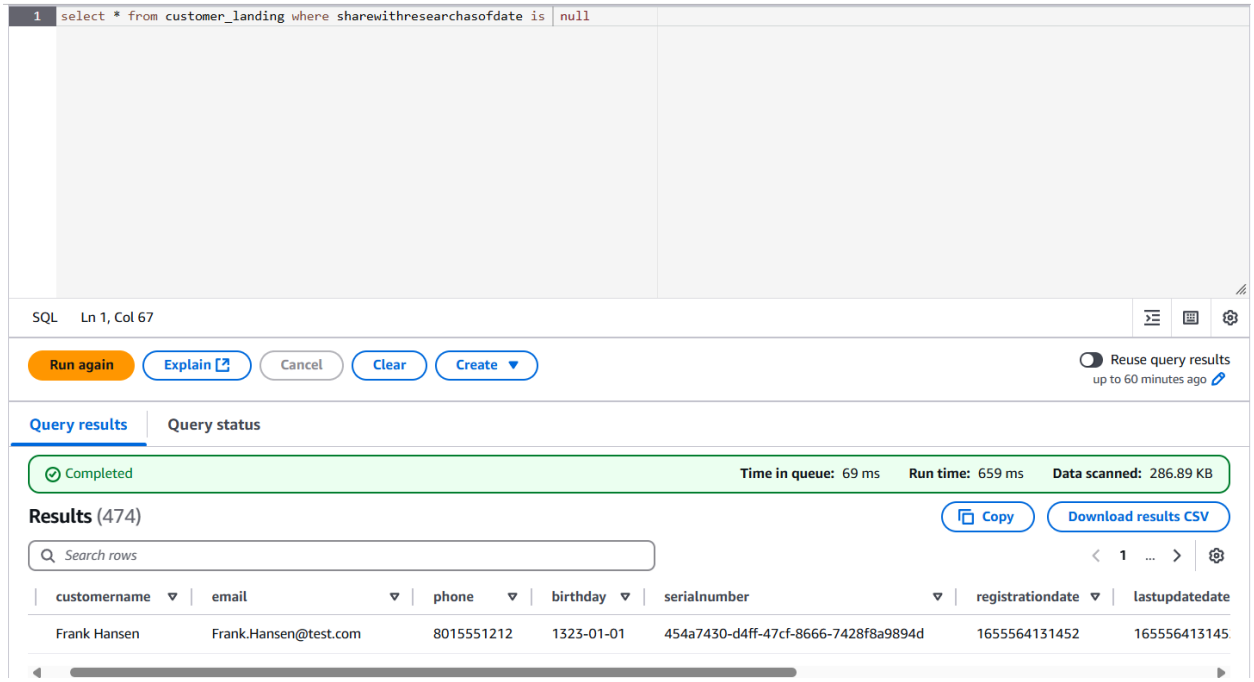
```python

# Script generated for node customer_curated
customer_curated_node1745313758346 = glueContext.create_dynamic_frame.from_catalog(database="finaldatabase", table_name="customer_curated", transformation_ctx="customer_curated_node1745313758346")

# Script generated for node steptrainer
steptrainer_node1745313734418 = glueContext.create_dynamic_frame.from_catalog(database="finaldatabase", table_name="steptrainer_landing", transformation_ctx="steptrainer_node1745313734418")

# Script generated for node SQL Query
SqlQuery0 = '''
select distinct s.sensorreadingtime,
s.serialnumber,s.distancefromobject
from steptrainer s
inner join customer_curated c
on s.serialnumber = c.serialnumber


'''
SQLQuery_node1745314070251 = sparkSqlQuery(glueContext, query = SqlQuery0, mapping = {"steptrainer":steptrainer_node1745313734418, "customer_curated":customer_curated_node1745313758346}, transformation_ctx =

# Script generated for node Amazon S3
EvaluateDataQuality().process_rows(frame=SQLQuery_node1745314070251, ruleset=DEFAULT_DATA_QUALITY_RULESET, publishing_options={"dataQualityEvaluationContext": "EvaluateDataQuality_node1745313648135", "enableD
AmazonS3_node1745314302198 = glueContext.getSink(path="s3://finalprojectsaleem/step_trainer/trusted/", connection_type="s3", updateBehavior="UPDATE_IN_DATABASE", partitionKeys=[], enableUpdateCatalog=True, tr
AmazonS3_node1745314302198.setCatalogInfo(catalogDatabase="finaldatabase",catalogTableName="steptrainer_trusted")
AmazonS3_node1745314302198.setFormat("json")
AmazonS3_node1745314302198.writeFrame(SQLQuery_node1745314070251)
job.commit()
```

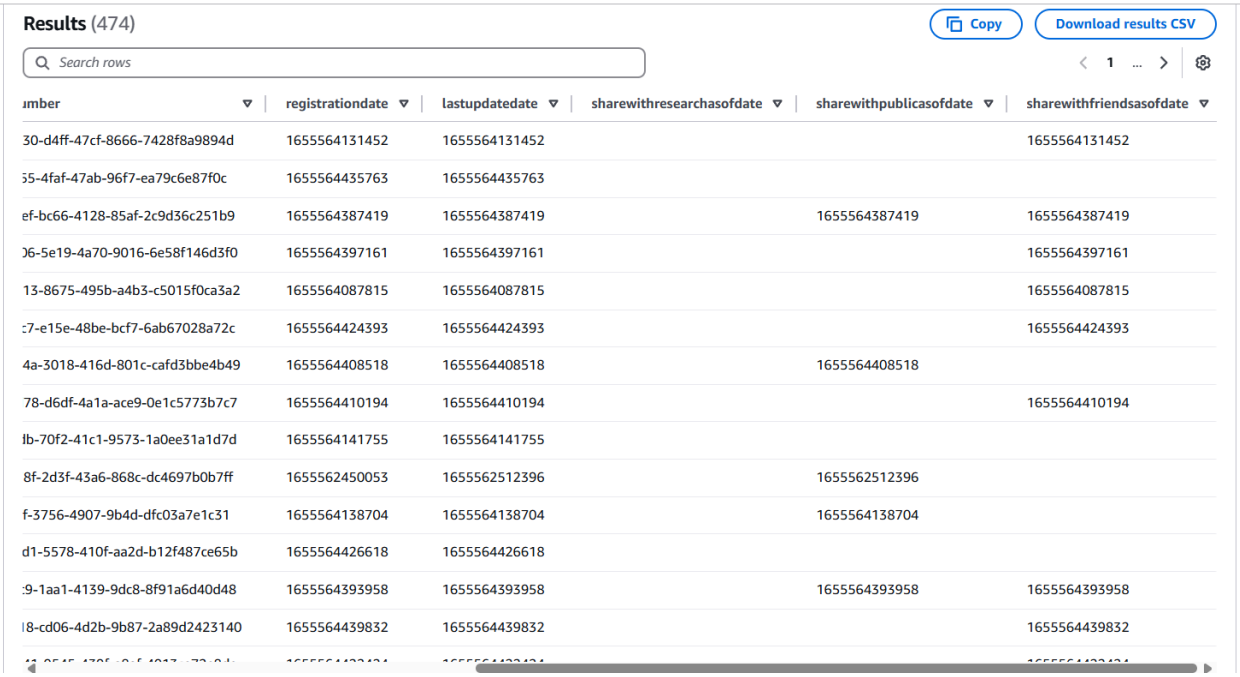
Machinelearing_curated:

```python
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

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 steptrainer_trusted
steptrainer_trusted_node1745314537862 = glueContext.create_dynamic_frame.from_catalog(database="finaldatabase", table_name="steptrainer_trusted", transformation_ctx="steptrainer_trusted_node1745314537862")

# Script generated for node accelerometer_trusted
```

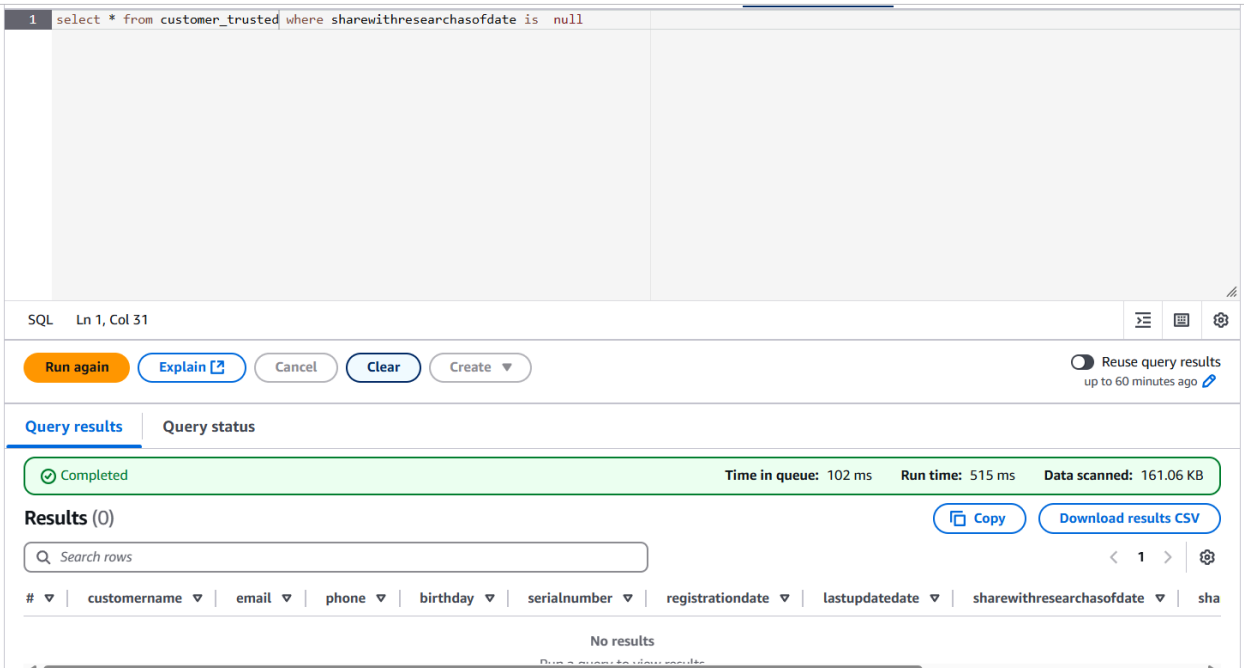
```
"""

# Script generated for node steptrainer_trusted
steptrainer_trusted_node1745314537862 = glueContext.create_dynamic_frame.from_catalog(database="finaldatabase", table_name="steptrainer_trusted", transformation_ctx="steptrainer_trusted_node1745314537862")

# Script generated for node accelerometer_trusted
accelerometer_trusted_node1745314562737 = glueContext.create_dynamic_frame.from_catalog(database="finaldatabase", table_name="accelerometer_trusted", transformation_ctx="accelerometer_trusted_node174531456273

# Script generated for node Join
Join_node1745314589281 = Join.apply(frame1=steptrainer_trusted_node1745314537862, frame2=accelerometer_trusted_node1745314562737, keys1=["sensorreadingtime"], keys2=["timestamp"], transformation_ctx="Join_nod

# Script generated for node Amazon S3
EvaluateDataQuality().process_rows(frame=Join_node1745314589281, ruleset=DEFAULT_DATA_QUALITY_RULESET, publishing_options={"dataQualityEvaluationContext": "EvaluateDataQuality_node1745314496747", "enableDataQ
AmazonS3_node1745314690522 = glueContext.getSink(path="s3://finalprojectsaleem/machine_learning_curated/", connection_type="s3", updateBehavior="UPDATE_IN_DATABASE", partitionKeys=[], enableUpdateCatalog=True
AmazonS3_node1745314690522.setCatalogInfo(catalogDatabase="finaldatabase",catalogTableName="machine_learning_curated")
AmazonS3_node1745314690522.setFormat("json")
AmazonS3_node1745314690522.writeFrame(Join_node1745314589281)
job.commit()
```

- The customer_landing data contains multiple rows with a blank shareWithResearchAsOfDate.





- The resulting customer_trusted data has no rows where shareWithResearchAsOfDate is blank.



I have included everything only the order is mismatched , but have provided proper heading and references , since some parts were completed in morning sessions and some work was dependent on the other one to complete.