Project Overview

As you have learned in this course Spark and AWS Glue allow you to process data from multiple sources, categorize the data, and curate it to be queried in the future for multiple purposes. In this project you will directly use the skills you have used, including some of the code you have already written.
You will go beyond that to write additional AWS Glue jobs to create curated step trainer data that can be used for machine learning.
In this project, you'll act as a data engineer for the STEDI team to build a data lake house solution for sensor data that trains a machine learning model.

Project Details

The STEDI Team has been hard at work developing a hardware **STEDI Step Trainer** that:

- trains the user to do a STEDI balance exercise;
- has sensors on the device that collect data to train a machine-learning algorithm to detect steps;
- has a companion mobile app that collects customer data and interacts with the device sensors.

STEDI has heard from millions of early adopters who are willing to purchase the STEDI Step Trainers and use them.
Several customers have already received their Step Trainers, installed the mobile application, and begun using them together to test their balance. The Step Trainer is just a motion sensor that records the distance of the object detected. The app uses a mobile phone accelerometer to detect motion in the X, Y, and Z directions.
The STEDI team wants to use the motion sensor data to train a machine learning model to detect steps accurately in real-time. *Privacy will be a primary consideration in deciding what data can be used.*
Some of the early adopters have agreed to share their data for research purposes. **Only these customers' Step Trainer and accelerometer data should be used in the training data for the machine learning model**.
As a data engineer on the STEDI Step Trainer team, you'll need to extract the data produced by the STEDI Step Trainer sensors *and* the mobile app, and curate them into a data lake house solution on AWS so that Data Scientists can train the learning model.

Environments

AWS Environment

You'll use the data from the STEDI Step Trainer and mobile app to develop a lakehouse solution in the cloud that curates the data for the machine learning model using:

- Python and Spark
- AWS Glue
- AWS Athena
- AWS S3

On the next page, you'll find instructions for accessing a temporary AWS account you can use to complete the project. This account has a budget of $25 you should be aware of when creating resources on the AWS platform. Pay special attention to what you are creating and running and the cost of these services.

Github Environment

You'll also need a github repository to store your SQL scripts and Python code in. You'll submit the code in this github repo for the project submission.

Workflow Environment Configuration

You'll be creating Python scripts using AWS Glue and Glue Studio. These web-based tools and services contain multiple options for editors to write or generate Python code that uses PySpark. Remember to save any code you develop or run in these editors on AWS to a local Github Repository.
You can use any Python editor locally to work with and save code as well, but be aware that to actually test or run Glue Jobs, you'll need to submit them to your AWS Glue environment.

Project Data

STEDI has **three JSON data sources** to use from the Step Trainer. Check out the JSON data in the following folders in the Github repo:

- customer
- step_trainer
- accelerometer

Download the data from **nd027-Data-Engineering-Data-Lakes-AWS-Exercises**

You should have:

- 956 rows in the customer_landing table,
- 81273 rows in the accelerometer_landing table, and
- 28680 rows in the step_trainer_landing table.

1. Customer Records

This is the data from fulfillment and the STEDI website.

Data Download URL

AWS S3 Bucket URI - s3://cd0030bucket/customers/

contains the following fields:

- serialnumber
- sharewithpublicasofdate
- birthday
- registrationdate
- sharewithresearchasofdate
- customername
- email
- lastupdatedate
- phone
- sharewithfriendsasofdate

2. Step Trainer Records

This is the data from the motion sensor.

Data Download URL

AWS S3 Bucket URI - s3://cd0030bucket/step_trainer/

contains the following fields:

- sensorReadingTime
- serialNumber
- distanceFromObject

3. Accelerometer Records

This is the data from the mobile app.

Data Download URL

AWS S3 Bucket URI - s3://cd0030bucket/accelerometer/

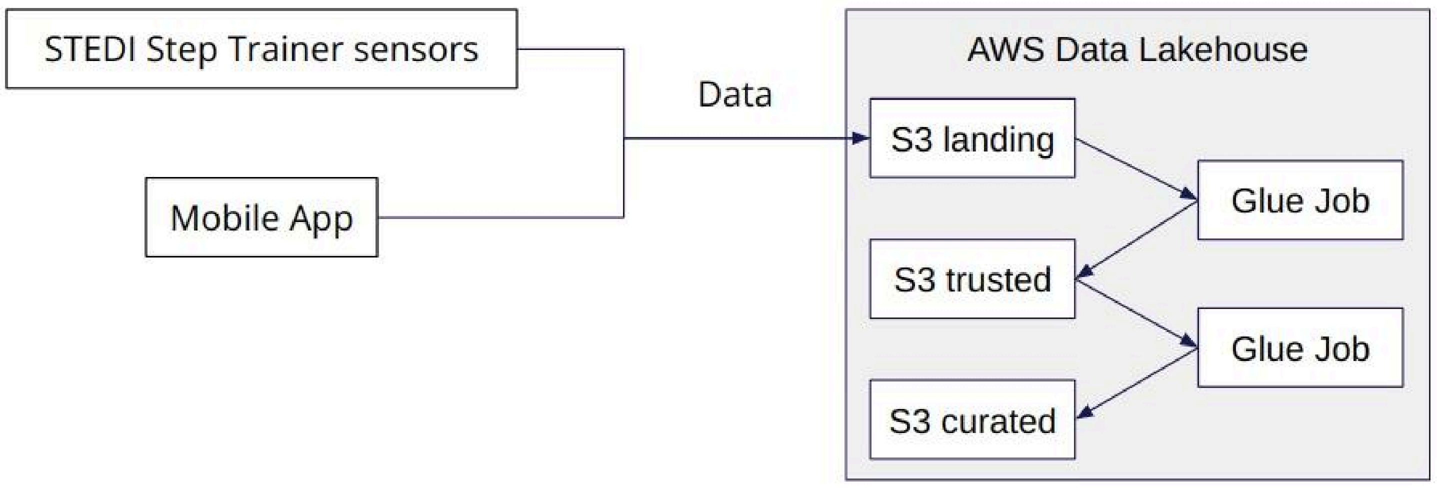
contains the following fields:

- timeStamp
- user
- x
- y
- z

Review the project **README** in the Github repo to understand the project structure.

Project Instructions

Using AWS Glue, AWS S3, Python, and Spark, create or generate Python scripts to build a lakehouse solution in AWS that satisfies these requirements from the STEDI data scientists.
Refer to the flowchart below to better understand the workflow.



Requirements

To simulate the data coming from the various sources, you will need to create your own S3 directories for customer_landing, step_trainer_landing, and accelerometer_landing zones, and copy the data there as a starting point.

- You have decided you want to get a feel for the data you are dealing with in a semi-structured format, so you decide to create **three Glue tables** for the three landing zones. Share your customer_landing.sql, accelerometer_landing.sql, and step_trainer_landing.sql scripts in git.
- Query those tables using Athena, and take a screenshot of each one showing the resulting data. Name the screenshots customer_landing(.png,.jpeg, etc.), accelerometer_landing(.png,.jpeg, etc.), step_trainer_landing (.png, .jpeg, etc.).

The Data Science team has done some preliminary data analysis and determined that the **Accelerometer Records** each match one of the **Customer Records**. They would like you to create 2 AWS Glue Jobs that do the following:

- Sanitize the Customer data from the Website (Landing Zone) and only store the Customer Records who agreed to share their data for research purposes (Trusted Zone) - creating a Glue Table called **customer_trusted**.
- Sanitize the Accelerometer data from the Mobile App (Landing Zone) - and only store Accelerometer Readings from customers who agreed to share their data for research purposes (Trusted Zone) - creating a Glue Table called **accelerometer_trusted**.
- You need to verify your Glue job is successful and only contains Customer Records from people who agreed to share their data. Query your Glue customer_trusted table with Athena and take a screenshot of the data. Name the screenshot customer_trusted(.png,.jpeg, etc.).

Data Scientists have discovered a data quality issue with the Customer Data. The serial number should be a unique identifier for the STEDI Step Trainer they purchased. However, there was a defect in the fulfillment website, and it used the same 30 serial numbers over and over again for millions of customers! Most customers have not received their Step Trainers yet, but those who have, are submitting Step Trainer data over the IoT network (Landing Zone). *The data from the Step Trainer Records has the correct serial numbers.*
The problem is that because of this serial number bug in the fulfillment data (Landing Zone), we don't know which customer the Step Trainer Records data belongs to.

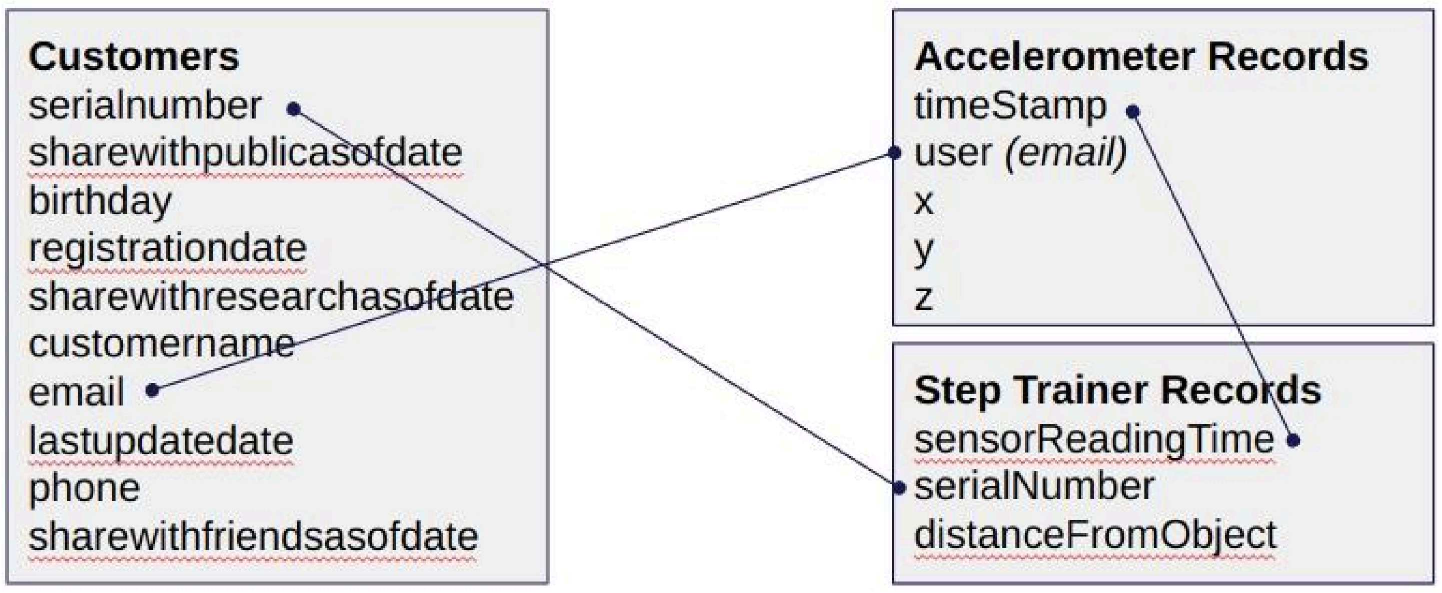
The Data Science team would like you to write a Glue job that does the following:

- 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.**

Refer to the relationship diagram below to understand the desired state.



Check your work!

After each stage of your project, check if the row count in the produced table is correct. You should have the following number of rows in each table:

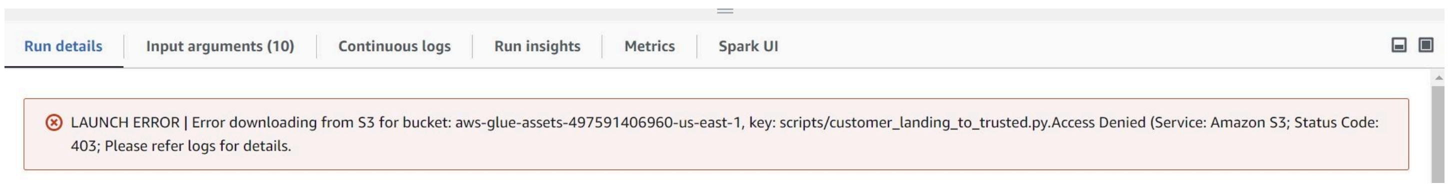
- Landing
  - Customer: 956
  - Accelerometer: 81273
  - Step Trainer: 28680
- Trusted
  - Customer: 482
  - Accelerometer: 40981
  - Step Trainer: 14460
- Curated
  - Customer: 482
  - Machine Learning: 43681

Hint: Use Transform - SQL Query nodes whenever you can. Other node types may give you unexpected results.
For example, rather than a Join node, you may use a SQL node that has two parents, then join them through a SQL query.

Troubleshooting

I get an Access Denied error when running a Glue Job

If you get the following error:



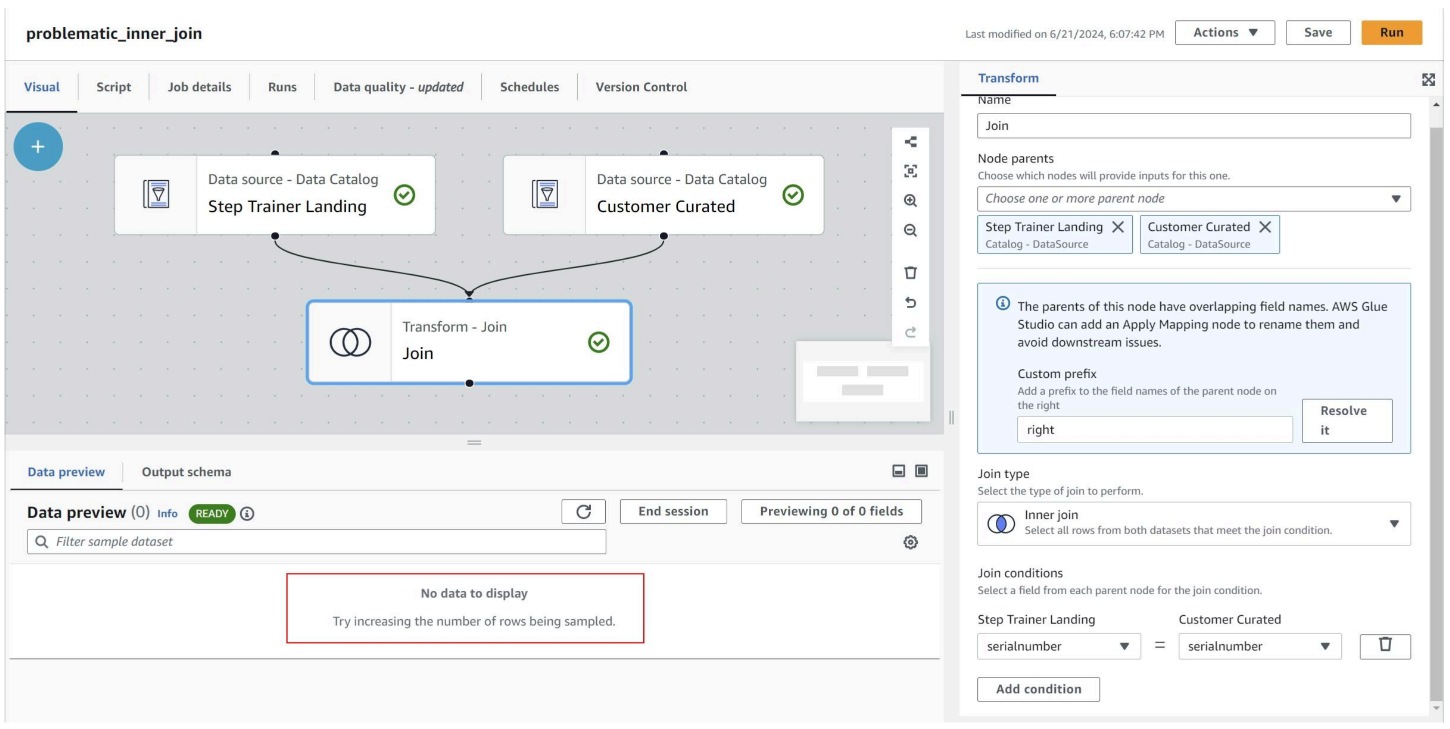
```
LAUNCH ERROR | Error downloading from S3 for bucket: aws-glue-assets-497591406960-us-east-1, key: scripts/customer_landing_to_trusted.py.Access Denied (Service: Amazon S3; Status Code: 403; Please refer logs for details.
```

Access Denied error when running a Glue Job

That means Glue can’t access your S3 bucket. Make sure the IAM role's **Permissions** and **Trust Relationships** are properly set.

step_trainer_landing JOIN customer_curated to create step_trainer_trusted, gives 0 rows

This problem happens when you perform an inner join between step_trainer_landing and customer_curated **in a Glue job**.



Data preview (0) Info READY

No data to display

Try increasing the number of rows being sampled.

Name: Join

Node parents: Step Trainer Landing (Catalog - DataSource), Customer Curated (Catalog - DataSource)

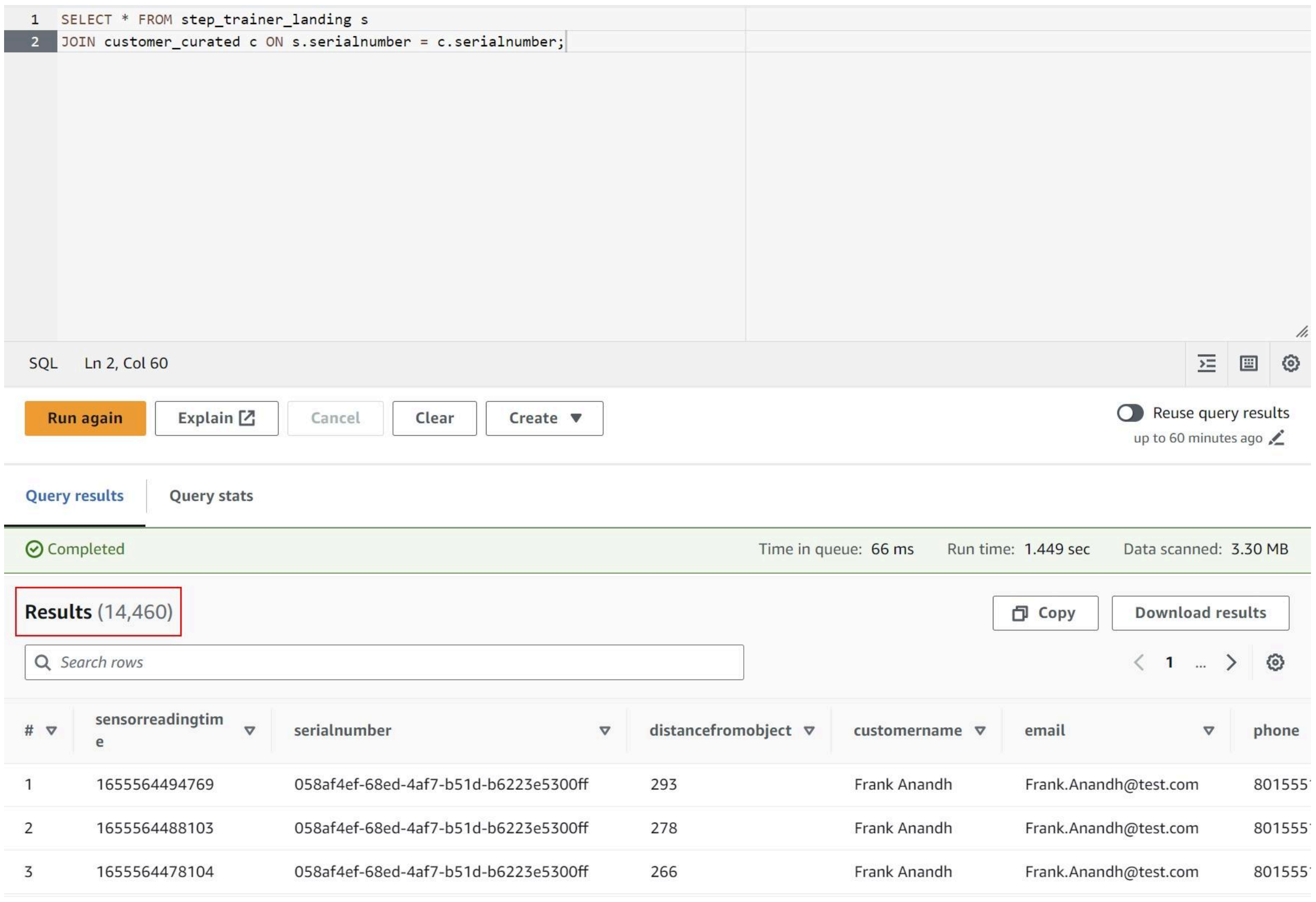
The parents of this node have overlapping field names. AWS Glue Studio can add an Apply Mapping node to rename them and avoid downstream issues.

Custom prefix: right

Join type: Inner join — Select all rows from both datasets that meet the join condition.

Join conditions: Step Trainer Landing serialnumber = Customer Curated serialnumber

This issue is strange because it only occurs in Glue Jobs. In Amazon Athena, the inner join works perfectly fine:



```sql
SELECT * FROM step_trainer_landing s
JOIN customer_curated c ON s.serialnumber = c.serialnumber;
```

Completed — Time in queue: 66 ms — Run time: 1.449 sec — Data scanned: 3.30 MB

Results (14,460)

#	sensorreadingtime	serialnumber	distancefromobject	customername	email	phone
1	1655564494769	058af4ef-68ed-4af7-b51d-b6223e5300ff	293	Frank Anandh	Frank.Anandh@test.com	801555
2	1655564488103	058af4ef-68ed-4af7-b51d-b6223e5300ff	278	Frank Anandh	Frank.Anandh@test.com	801555
3	1655564478104	058af4ef-68ed-4af7-b51d-b6223e5300ff	266	Frank Anandh	Frank.Anandh@test.com	801555

The solution here is to avoid using inner join, since it is not appropriate in this case anyway, due to non-unique values. Instead, you should **Select all rows in step_trainer_landing table that also exist IN the customer_curated table.**

Project requirements

Landing Zone

Criteria	Submission Requirements
Use Glue Studio to ingest data from an S3 bucket	**customer_landing_to_trusted.py**, **accelerometer_landing_to_trusted.py**, and **step_trainer_trusted.py** Glue jobs have a node that connects to S3 bucket for customer, accelerometer, and step trainer landing zones.
Manually create a Glue Table using Glue Console from JSON data	SQL DDL scripts **customer_landing.sql**, **accelerometer_landing.sql**, and **step_trainer_landing.sql** include all of the JSON fields in the data input files and are appropriately typed (not everything is a string).
Use Athena to query the Landing Zone.	Include screenshots showing various queries run on Athena, along with their results: • Count of customer_landing: 956 rows • The customer_landing data contains multiple rows with a blank shareWithResearchAsOfDate. • Count of accelerometer_landing: 81273 rows • Count of step_trainer_landing: 28680 rows

Trusted Zone

Criteria	Submission Requirements
Configure Glue Studio to dynamically update a Glue Table schema from JSON data	Glue Job Python code shows that the option to dynamically infer and update schema is enabled. To do this, set the *Create a table in the Data Catalog and, on subsequent runs, update the schema and add new partitions* option to True.
Use Athena to query Trusted Glue Tables	Include screenshots showing various queries run on Athena, along with their results: • Count of customer_trusted: 482 rows • The resulting customer_trusted data has no rows where shareWithResearchAsOfDate is blank. • Count of accelerometer_trusted: 40981 rows • Count of step_trainer_trusted: 14460 rows However, **if you are following the stand-out suggestions**, your row counts should be as follows: • Count of customer_trusted: 482 rows • The resulting customer_trusted data has no rows where shareWithResearchAsOfDate is blank. • Count of accelerometer_trusted: 32025 rows • Count of step_trainer_trusted: 14460 rows
Filter protected PII with Spark in Glue Jobs	**customer_landing_to_trusted.py** has a node that drops rows that do not have data in the sharedWithResearchAsOfDate column. Hints: • **Transform - SQL Query** node often gives more consistent outputs than other node types. • Glue Jobs do not replace any file. Delete your S3 files and Athena table whenever you update your visual ETLs.
Join Privacy tables with Glue Jobs	**accelerometer_landing_to_trusted.py** has a node that inner joins the customer_trusted data with the accelerometer_landing data by emails. The produced table should have only columns from the accelerometer table.

Curated Zone

Criteria	Submission Requirements
Write a Glue Job to join trusted data	**customer_trusted_to_curated.py** has a node that inner joins the `customer_trusted` data with the `accelerometer_trusted` data by emails. The produced table should have only columns from the customer table.
Write a Glue Job to create curated data	**step_trainer_trusted.py** has a node that inner joins the step_trainer_landing data with the customer_curated data by serial numbers **machine_learning_curated.py** has a node that inner joins the step_trainer_trusted data with the accelerometer_trusted data by sensor reading time and timestamps Hints: • **Data Source - S3 bucket** node sometimes extracted incomplete data. Use the **Data Source - Data Catalog** node when that's the case. • Use the Data Preview feature with at least 500 rows to ensure the number of customer-curated rows is correct. Click "Start data preview session", then click the gear next to the "Filter" text box to update the number of rows • As before, the **Transform - SQL Query** node often gives more consistent outputs than any other node type. Tip - replace the JOIN node with it. • The `step_trainer_trusted` may take about 8 minutes to run.
Use Athena to query Curated Glue Tables	Include screenshots showing various queries run on Athena, along with their results: • Count of customer_curated: 482 rows • Count of machine_learning_curated: 43681 rows However, **if you are following the stand-out suggestions**, your row counts should be as follows: • Count of customer_curated: 464 rows • Count of machine_learning_curated: 34437 rows Hint: If you get unexpected results, consider using the **Transform - SQL Query** node rather than Glue-provided nodes.

Suggestions to Make Your Project Stand Out

Consider these additions to your project to make it stand out!

- When creating the Glue Job to join data from the accelerometer readings and the customer table, filter out any readings that were prior to the research consent date. This will ensure consent was in place at the time that data was gathered. This helps in the case that in the future the customer revokes consent. We can be sure that the data we used for research was used when consent was in place for that particular data.
- Anonymize the final curated table so that it is not subject to GDPR or other privacy regulations, in case a customer requests deletion of PII, we will not be in violation by retaining PII data --remove email, and any other personally identifying information up front.