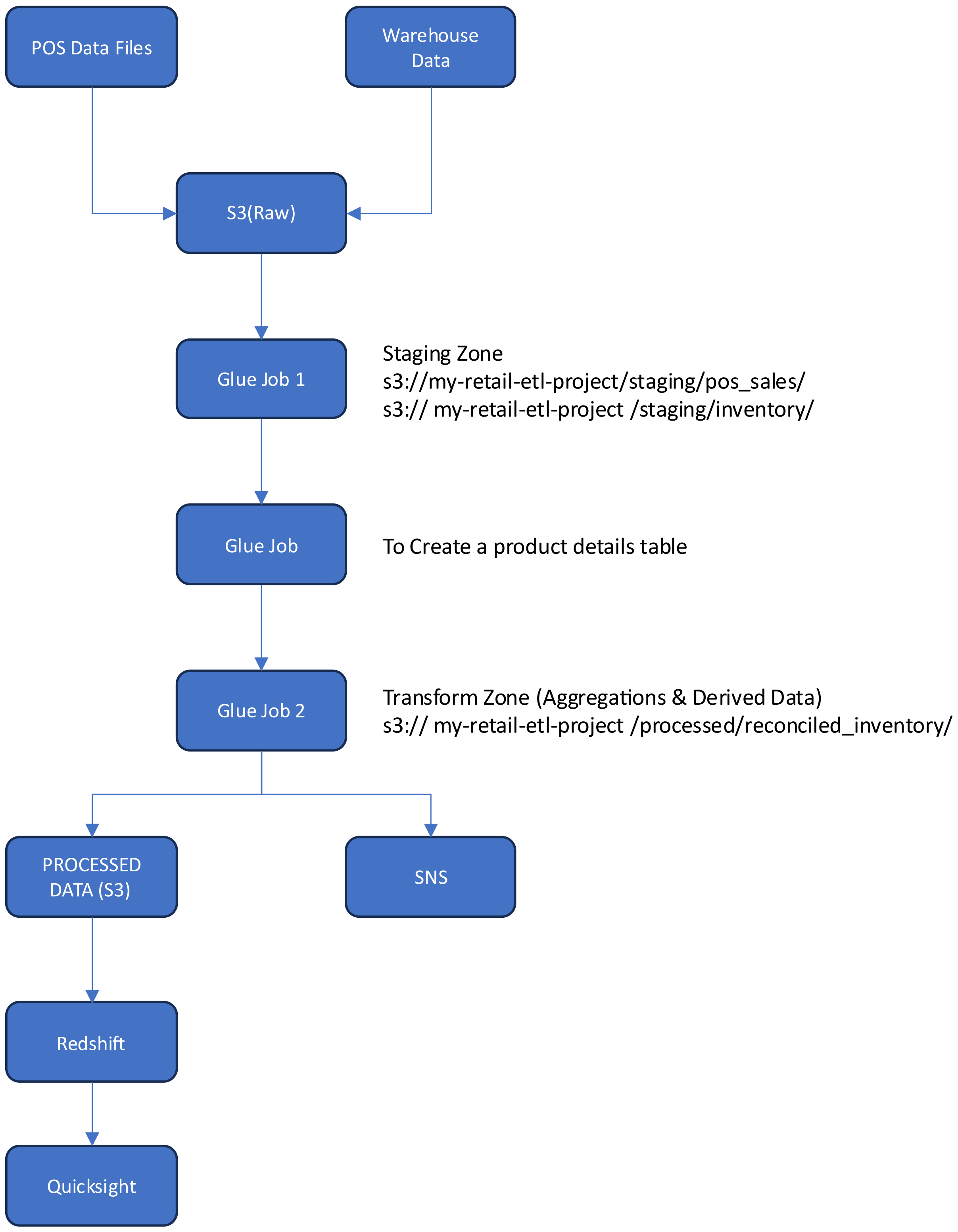
Retail Inventory Reconciliation (Batch ETL)

Summary:

- Purpose: This document outlines the architecture and data flow for the automated sales data pipeline. The primary goal of this pipeline is to ingest Point-of-Sale (POS) and supplementary warehouse data, process it, and load it into a centralized data warehouse to enable business intelligence (BI) and reporting.
- Business Goal: To provide the sales and marketing teams with daily updated dashboards in Amazon QuickSight for analyzing sales performance, product trends, and store efficiency.
- Technologies Used: AWS S3, AWS Glue, AWS SNS, Amazon Redshift, Amazon QuickSight, Airflow.

Architecture Diagram:

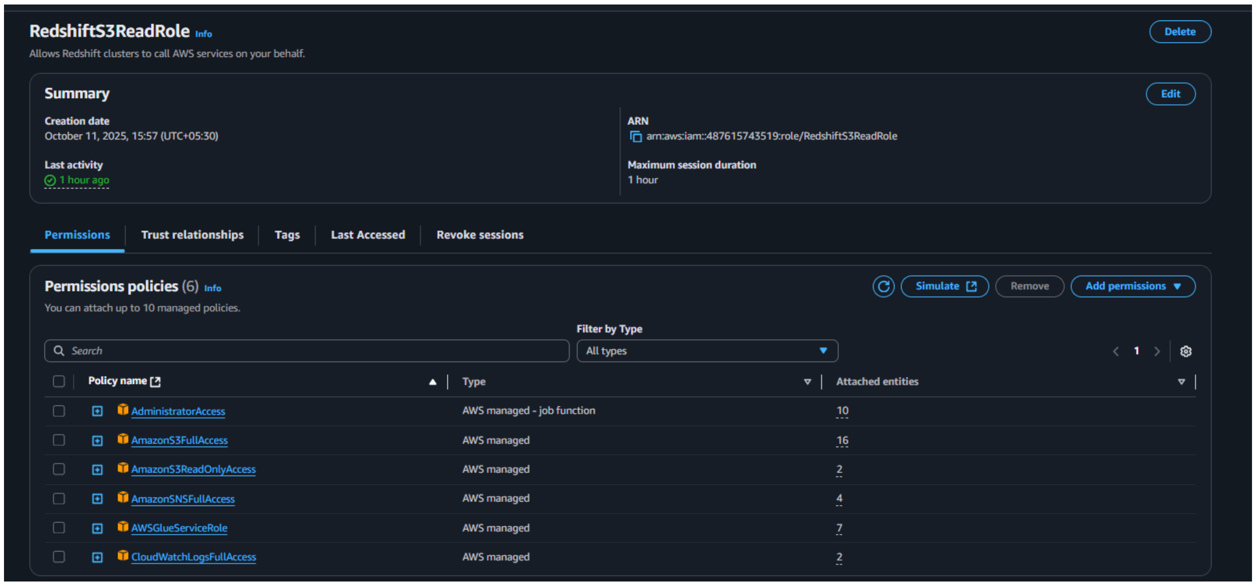

POS Data Files
Warehouse Data
S3(Raw)
Glue Job 1
Staging Zone
s3://my-retail-etl-project/staging/pos_sales/
s3:// my-retail-etl-project /staging/inventory/
Glue Job
To Create a product details table
Glue Job 2
Transform Zone (Aggregations & Derived Data)
s3:// my-retail-etl-project /processed/reconciled_inventory/
PROCESSED DATA (S3)
SNS
Redshift
Quicksight

Detailed Implementation Steps & Configuration:

Step 1: Prerequisite: IAM Role Configuration

Before creating the pipeline, we need IAM roles with the correct permissions to allow AWS services to interact with each other.

IAM Role:



RedshiftS3ReadRole

Allows Redshift clusters to call AWS services on your behalf.

Summary

Creation date: October 11, 2025, 15:57 (UTC+05:30)

ARN: arn:aws:iam::487615743519:role/RedshiftS3ReadRole

Last activity: 1 hour ago

Maximum session duration: 1 hour

Permissions policies (6)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AdministratorAccess	AWS managed - job function	10
AmazonS3FullAccess	AWS managed	16
AmazonS3ReadOnlyAccess	AWS managed	2
AmazonSNSFullAccess	AWS managed	4
AWSGlueServiceRole	AWS managed	7
CloudWatchLogsFullAccess	AWS managed	2

Step 2: Amazon S3 Bucket Setup

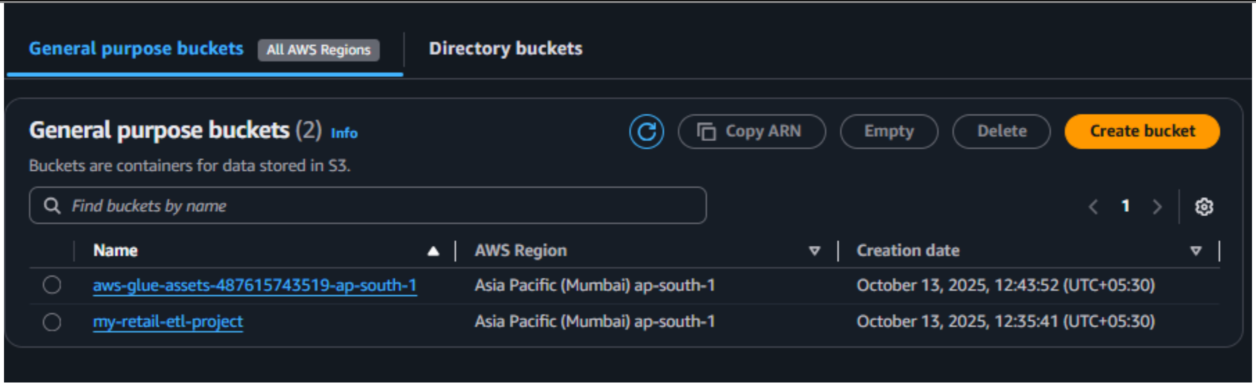
First, we'll create the main container for your project files. Remember, S3 bucket names must be **globally unique**.

1. **Navigate to S3:** In the AWS Management Console, search for and select "S3".
2. **Start Bucket Creation:** Click the **Create bucket** button.
3. **Configure Bucket Name and Region:**
   - **Bucket name:** Choose a unique name. A good practice is [project-name]-[purpose]-[date].
     - Example: retail-etl-project-datalake-20251013
   - **AWS Region:** Select the region closest to you, for example, **Asia Pacific (Mumbai) ap-south-1**.
4. **Object Ownership:** Leave this as the recommended default, **ACLs disabled**.

5. **Block Public Access (Security):**
    - Keep the box for **Block all public access** checked. This is the most secure setting and is correct for a data pipeline where access is programmatic, not public.
6. **Bucket Versioning (Data Protection):**
    - Select **Enable**. This is a best practice that keeps a history of all your file versions, protecting you from accidental overwrites or deletions.
7. **Create the Bucket:**
    - Scroll to the bottom and click **Create bucket**.

You now have a secure, versioned S3 bucket ready for your data.

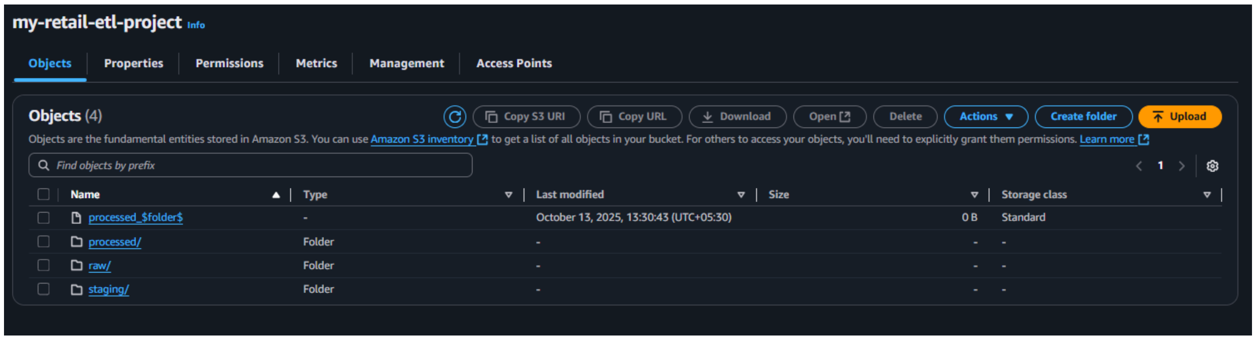
```
raw/
├── pos_sales/
│   └── date=2025-10-13/
│       └── pos_sales_2025-10-13.csv       <- Sales for Oct 13th
└── warehouse_inventory/
    ├── date=2025-10-12/
    │   └── warehouse_inventory_2025-10-12.csv   <- Opening Stock for Oct 13th
    └── date=2025-10-13/
        └── warehouse_inventory_2025-10-13.csv   <- Closing Stock for Oct 13th
```



General purpose buckets (2) Info

Buckets are containers for data stored in S3.

Name	AWS Region	Creation date
aws-glue-assets-487615743519-ap-south-1	Asia Pacific (Mumbai) ap-south-1	October 13, 2025, 12:43:52 (UTC+05:30)
my-retail-etl-project	Asia Pacific (Mumbai) ap-south-1	October 13, 2025, 12:35:41 (UTC+05:30)

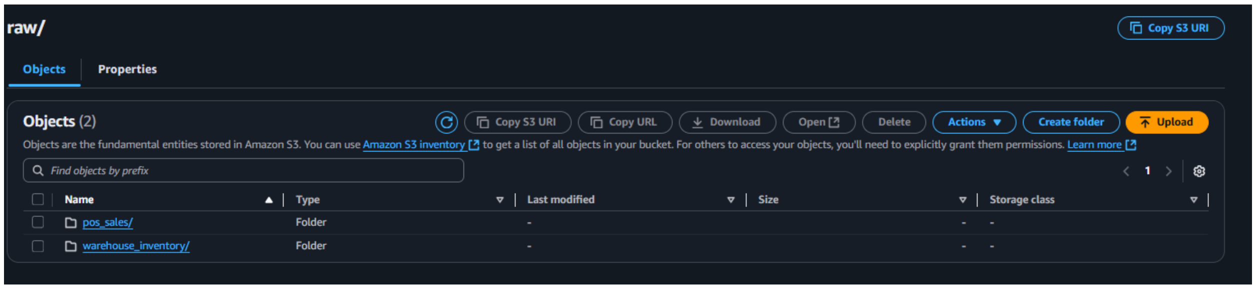


my-retail-etl-project Info

Objects (4)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
processed_$folder$	-	October 13, 2025, 13:30:43 (UTC+05:30)	0 B	Standard
processed/	Folder	-	-	-
raw/	Folder	-	-	-
staging/	Folder	-	-	-

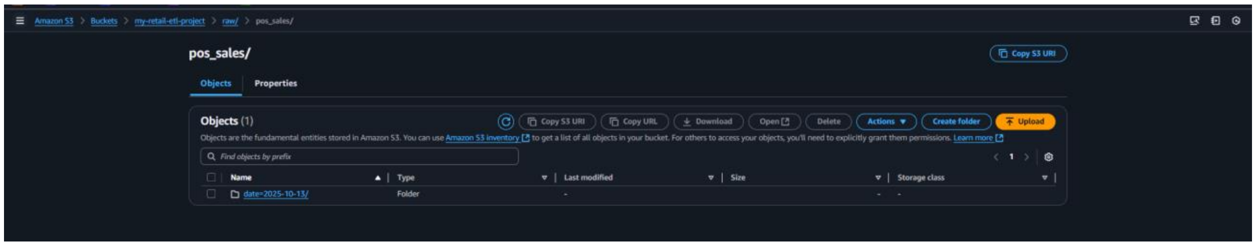


raw/

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
pos_sales/	Folder	-	-	-
warehouse_inventory/	Folder	-	-	-



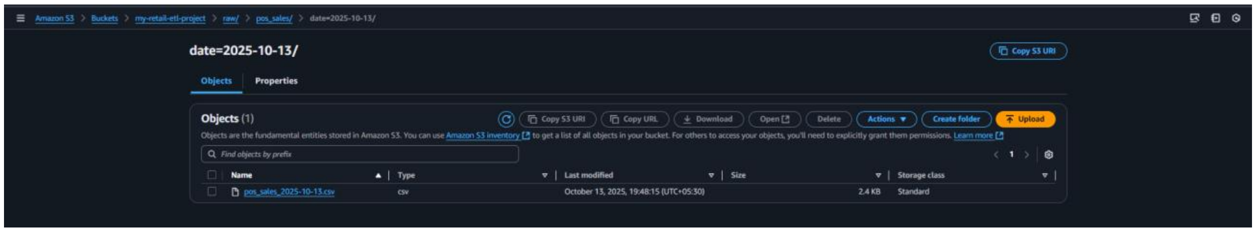
Amazon S3 > Buckets > my-retail-etl-project > raw/ > pos_sales/

pos_sales/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
date=2025-10-13/	Folder	-	-	-



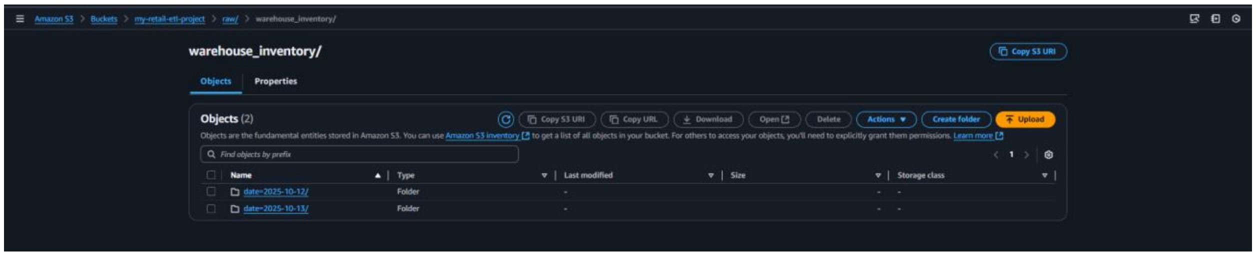
Amazon S3 > Buckets > my-retail-etl-project > raw/ > pos_sales/ > date=2025-10-13/

date=2025-10-13/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
pos_sales_2025-10-13.csv	csv	October 13, 2025, 19:48:15 (UTC+05:30)	2.4 KB	Standard



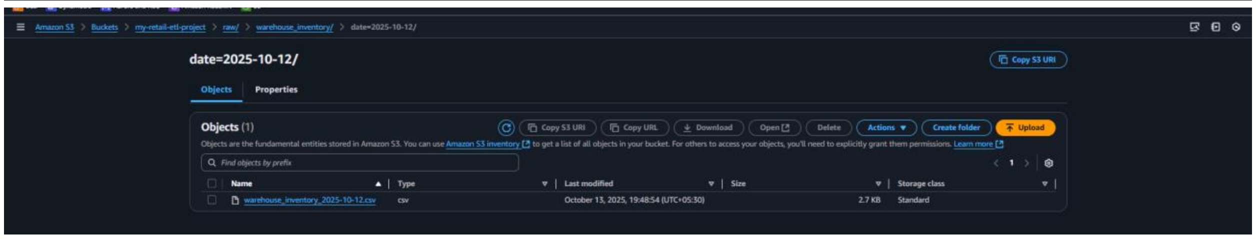
Amazon S3 > Buckets > my-retail-etl-project > raw/ > warehouse_inventory/

warehouse_inventory/

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
date=2025-10-12/	Folder	-	-	-
date=2025-10-13/	Folder	-	-	-



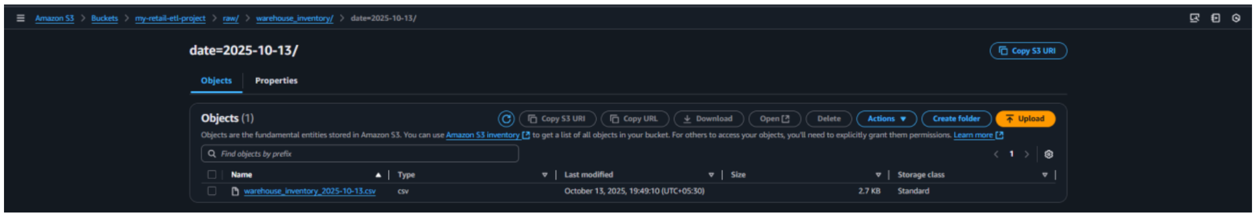
Amazon S3 > Buckets > my-retail-etl-project > raw/ > warehouse_inventory/ > date=2025-10-12/

date=2025-10-12/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
warehouse_inventory_2025-10-12.csv	csv	October 13, 2025, 19:48:54 (UTC+05:30)	2.7 KB	Standard



Amazon S3 > Buckets > my-retail-etl-project > raw/ > warehouse_inventory/ > date=2025-10-13/

date=2025-10-13/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
warehouse_inventory_2025-10-13.csv	csv	October 13, 2025, 19:49:10 (UTC+05:30)	2.7 KB	Standard

pos_sales_2025-10-13:

```
transaction_id,store_id,sku,product_name,sale_date,quantity,price
T051,S001,SKU001,Wireless Mouse,2025-10-13,3,599
T052,S002,SKU008,Headphones,2025-10-13,5,1299
T053,S003,SKU016,Monitor 27Inch,2025-10-13,1,10999
T054,S004,SKU022,Microphone,2025-10-13,2,2999
T055,S001,SKU030,VR Headset,2025-10-13,1,9999
T056,S002,SKU045,Headphones,2025-10-13,4,1299
T057,S003,SKU012,Pen Drive 32GB,2025-10-13,8,499
T058,S004,SKU004,HDMI Cable,2025-10-13,12,299
T059,S001,SKU009,Mouse Pad,2025-10-13,10,199
T060,S002,SKU019,Portable SSD,2025-10-13,2,7999
T061,S003,SKU028,Stylus Pen,2025-10-13,5,599
T062,S004,SKU036,Wifi Adapter,2025-10-13,3,799
T063,S001,SKU050,Router,2025-10-13,1,2399
T064,S002,SKU041,HDMI Cable,2025-10-13,7,299
T065,S003,SKU033,Screen Cleaner Kit,2025-10-13,4,299
T066,S004,SKU025,Power Bank,2025-10-13,3,1299
T067,S001,SKU017,Smart Watch,2025-10-13,1,4999
T068,S002,SKU002,Keyboard,2025-10-13,4,899
T069,S003,SKU011,External HDD 1TB,2025-10-13,2,4599
T070,S004,SKU020,Wireless Charger,2025-10-13,6,999
T071,S001,SKU031,Smartphone Stand,2025-10-13,5,499
T072,S002,SKU048,External HDD 1TB,2025-10-13,1,4599
T073,S003,SKU039,Keyboard,2025-10-13,3,899
T074,S004,SKU049,Pen Drive 32GB,2025-10-13,10,499
T075,S001,SKU005,USB Hub,2025-10-13,2,499
T076,S002,SKU015,Laptop Bag,2025-10-13,2,1499
T077,S003,SKU024,Phone Case,2025-10-13,15,299
T078,S004,SKU032,Wireless Earbuds,2025-10-13,3,3499
T079,S001,SKU042,USB Hub,2025-10-13,4,499
T080,S002,SKU007,WebCam,2025-10-13,1,1999
T081,S003,SKU014,Printer Ink,2025-10-13,2,699
T082,S004,SKU021,HD Webcam,2025-10-13,2,2499
T083,S001,SKU035,Laptop Cooling Pad,2025-10-13,1,1499
T084,S002,SKU044,Webcam,2025-10-13,3,1999
T085,S003,SKU003,Monitor 24Inch,2025-10-13,1,7999
T086,S004,SKU013,Router,2025-10-13,1,2399
T087,S001,SKU023,Speaker Bluetooth,2025-10-13,2,1799
T088,S002,SKU034,Phone Tripod,2025-10-13,2,799
T089,S003,SKU043,Laptop Stand,2025-10-13,4,999
T090,S004,SKU047,Wireless Keyboard,2025-10-13,2,1299
T091,S001,SKU010,Wireless Keyboard,2025-10-13,3,1299
T092,S002,SKU026,Extension Board,2025-10-13,5,899
T093,S003,SKU037,Graphic Tablet,2025-10-13,1,8999
T094,S004,SKU046,Mouse Pad,2025-10-13,8,199
T095,S001,SKU006,Laptop Stand,2025-10-13,3,999
T096,S002,SKU018,Tablet 10Inch,2025-10-13,1,14999
T097,S003,SKU027,Monitor Stand,2025-10-13,1,1199
T098,S004,SKU029,HDMI Splitter,2025-10-13,2,699
T099,S001,SKU038,Wireless Mouse,2025-10-13,3,599
T100,S002,SKU040,Monitor 24Inch,2025-10-13,1,7999
```

warehouse_inventory_2025-10-12:

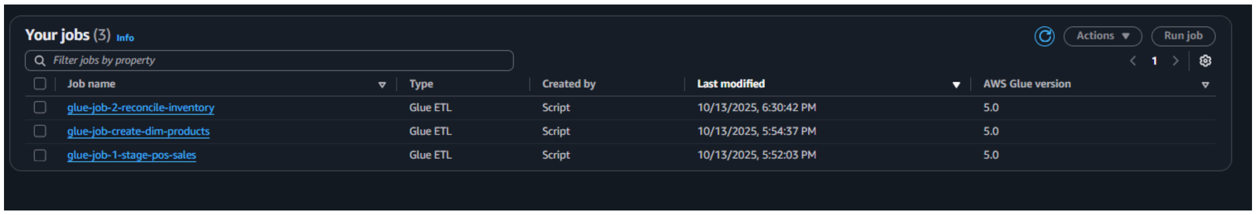
```
transaction_id,store_id,sku,product_name,sale_date,quantity,price
T051,S001,SKU001,Wireless Mouse,2025-10-13,3,599
T052,S002,SKU008,Headphones,2025-10-13,5,1299
T053,S003,SKU016,Monitor 27Inch,2025-10-13,1,10999
T054,S004,SKU022,Microphone,2025-10-13,2,2999
T055,S001,SKU030,VR Headset,2025-10-13,1,9999
T056,S002,SKU045,Headphones,2025-10-13,4,1299
T057,S003,SKU012,Pen Drive 32GB,2025-10-13,8,499
T058,S004,SKU004,HDMI Cable,2025-10-13,12,299
T059,S001,SKU009,Mouse Pad,2025-10-13,10,199
T060,S002,SKU019,Portable SSD,2025-10-13,2,7999
T061,S003,SKU028,Stylus Pen,2025-10-13,5,599
T062,S004,SKU036,Wifi Adapter,2025-10-13,3,799
T063,S001,SKU050,Router,2025-10-13,1,2399
T064,S002,SKU041,HDMI Cable,2025-10-13,7,299
T065,S003,SKU033,Screen Cleaner Kit,2025-10-13,4,299
T066,S004,SKU025,Power Bank,2025-10-13,3,1299
T067,S001,SKU017,Smart Watch,2025-10-13,1,4999
T068,S002,SKU002,Keyboard,2025-10-13,4,899
T069,S003,SKU011,External HDD 1TB,2025-10-13,2,4599
T070,S004,SKU020,Wireless Charger,2025-10-13,6,999
T071,S001,SKU031,Smartphone Stand,2025-10-13,5,499
T072,S002,SKU048,External HDD 1TB,2025-10-13,1,4599
T073,S003,SKU039,Keyboard,2025-10-13,3,899
T074,S004,SKU049,Pen Drive 32GB,2025-10-13,10,499
T075,S001,SKU005,USB Hub,2025-10-13,2,499
T076,S002,SKU015,Laptop Bag,2025-10-13,2,1499
T077,S003,SKU024,Phone Case,2025-10-13,15,299
T078,S004,SKU032,Wireless Earbuds,2025-10-13,3,3499
T079,S001,SKU042,USB Hub,2025-10-13,4,499
T080,S002,SKU007,WebCam,2025-10-13,1,1999
T081,S003,SKU014,Printer Ink,2025-10-13,2,699
T082,S004,SKU021,HD Webcam,2025-10-13,2,2499
T083,S001,SKU035,Laptop Cooling Pad,2025-10-13,1,1499
T084,S002,SKU044,Webcam,2025-10-13,3,1999
T085,S003,SKU003,Monitor 24Inch,2025-10-13,1,7999
T086,S004,SKU013,Router,2025-10-13,1,2399
T087,S001,SKU023,Speaker Bluetooth,2025-10-13,2,1799
T088,S002,SKU034,Phone Tripod,2025-10-13,2,799
T089,S003,SKU043,Laptop Stand,2025-10-13,4,999
T090,S004,SKU047,Wireless Keyboard,2025-10-13,2,1299
T091,S001,SKU010,Wireless Keyboard,2025-10-13,3,1299
T092,S002,SKU026,Extension Board,2025-10-13,5,899
T093,S003,SKU037,Graphic Tablet,2025-10-13,1,8999
T094,S004,SKU046,Mouse Pad,2025-10-13,8,199
T095,S001,SKU006,Laptop Stand,2025-10-13,3,999
T096,S002,SKU018,Tablet 10Inch,2025-10-13,1,14999
T097,S003,SKU027,Monitor Stand,2025-10-13,1,1199
T098,S004,SKU029,HDMI Splitter,2025-10-13,2,699
T099,S001,SKU038,Wireless Mouse,2025-10-13,3,599
T100,S002,SKU040,Monitor 24Inch,2025-10-13,1,7999
```

warehouse_inventory_2025-10-13:

```
record_id,sku,product_name,category,warehouse_location,stock_on_hand,received_date
R051,SKU001,Wireless Mouse,Accessories,WH1,111,2025-10-13
R052,SKU002,Keyboard,Accessories,WH1,73,2025-10-13
R053,SKU003,Monitor 24Inch,Displays,WH2,42,2025-10-13
R054,SKU004,HDMI Cable,Cables,WH3,278,2025-10-13
R055,SKU005,USB Hub,Accessories,WH2,134,2025-10-13
R056,SKU006,Laptop Stand,Accessories,WH1,101,2025-10-13
R057,SKU007,WebCam,Electronics,WH4,47,2025-10-13
R058,SKU008,Headphones,Audio,WH2,185,2025-10-13
R059,SKU009,Mouse Pad,Accessories,WH3,158,2025-10-13
R060,SKU010,Wireless Keyboard,Accessories,WH1,133,2025-10-13
R061,SKU011,External HDD 1TB,Storage,WH2,65,2025-10-13
R062,SKU012,Pen Drive 32GB,Storage,WH3,232,2025-10-13
R063,SKU013,Router,Networking,WH4,57,2025-10-13
R064,SKU014,Printer Ink,Printers,WH1,33,2025-10-13
R065,SKU015,Laptop Bag,Accessories,WH2,94,2025-10-13
R066,SKU016,Monitor 27Inch,Displays,WH3,27,2025-10-13
R067,SKU017,Smart Watch,Wearables,WH4,22,2025-10-13
R068,SKU018,Tablet 10Inch,Tablets,WH1,18,2025-10-13
R069,SKU019,Portable SSD,Storage,WH2,35,2025-10-13
R070,SKU020,Wireless Charger,Accessories,WH3,66,2025-10-13
R071,SKU021,HD Webcam,Electronics,WH1,40,2025-10-13
R072,SKU022,Microphone,Audio,WH4,31,2025-10-13
R073,SKU023,Speaker Bluetooth,Audio,WH2,104,2025-10-13
R074,SKU024,Phone Case,Accessories,WH3,127,2025-10-13
R075,SKU025,Power Bank,Electronics,WH1,71,2025-10-13
R076,SKU026,Extension Board,Accessories,WH2,60,2025-10-13
R077,SKU027,Monitor Stand,Displays,WH4,22,2025-10-13
R078,SKU028,Stylus Pen,Accessories,WH1,82,2025-10-13
R079,SKU029,HDMI Splitter,Cables,WH3,37,2025-10-13
R080,SKU030,VR Headset,Electronics,WH2,13,2025-10-13
R081,SKU031,Smartphone Stand,Accessories,WH1,118,2025-10-13
R082,SKU032,Wireless Earbuds,Audio,WH2,52,2025-10-13
R083,SKU033,Screen Cleaner Kit,Accessories,WH3,190,2025-10-13
R084,SKU034,Phone Tripod,Accessories,WH4,65,2025-10-13
R085,SKU035,Laptop Cooling Pad,Accessories,WH1,57,2025-10-13
R086,SKU036,Wifi Adapter,Networking,WH2,39,2025-10-13
R087,SKU037,Graphic Tablet,Electronics,WH3,18,2025-10-13
R088,SKU038,Wireless Mouse,Accessories,WH4,93,2025-10-13
R089,SKU039,Keyboard,Accessories,WH1,77,2025-10-13
R090,SKU040,Monitor 24Inch,Displays,WH2,48,2025-10-13
R091,SKU041,HDMI Cable,Cables,WH3,294,2025-10-13
R092,SKU042,USB Hub,Accessories,WH4,71,2025-10-13
R093,SKU043,Laptop Stand,Accessories,WH1,89,2025-10-13
R094,SKU044,Webcam,Electronics,WH2,50,2025-10-13
R095,SKU045,Headphones,Audio,WH3,162,2025-10-13
R096,SKU046,Mouse Pad,Accessories,WH4,147,2025-10-13
R097,SKU047,Wireless Keyboard,Accessories,WH1,69,2025-10-13
R098,SKU048,External HDD 1TB,Storage,WH2,57,2025-10-13
R099,SKU049,Pen Drive 32GB,Storage,WH3,212,2025-10-13
R100,SKU050,Router,Networking,WH4,52,2025-10-13
```

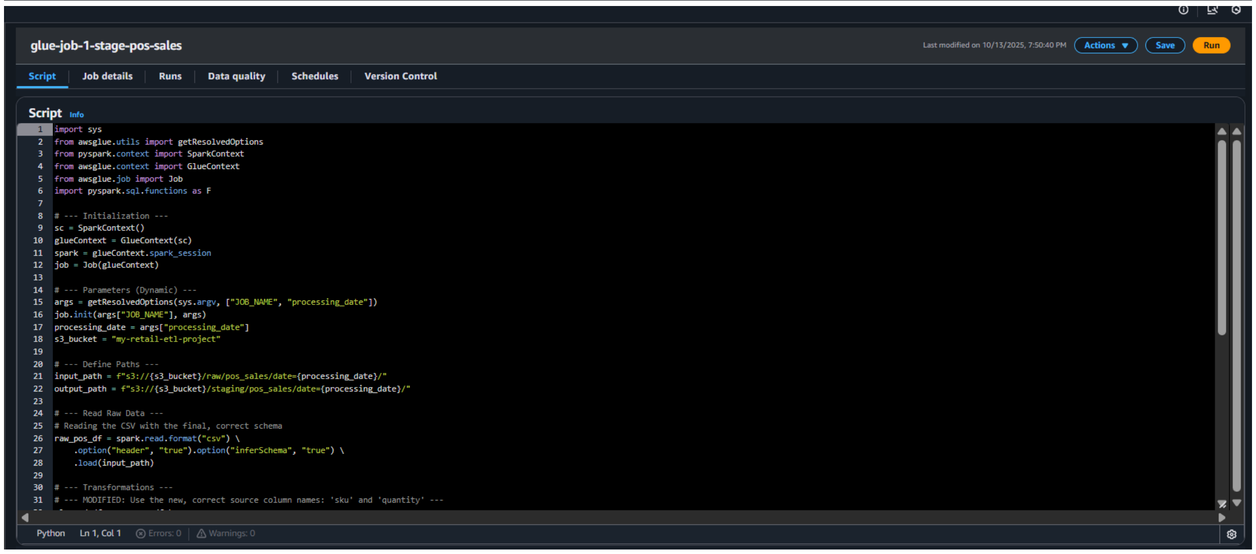
Step 3: Glue Job 1

```python
import sys
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
import pyspark.sql.functions as F

# --- Initialization ---
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

# --- Parameters (Dynamic) ---
args = getResolvedOptions(sys.argv, ["JOB_NAME", "processing_date"])
job.init(args["JOB_NAME"], args)
processing_date = args["processing_date"]
s3_bucket = "my-retail-etl-project"

# --- Define Paths ---
input_path = f"s3://{s3_bucket}/raw/pos_sales/date={processing_date}/"
output_path = f"s3://{s3_bucket}/staging/pos_sales/date={processing_date}/"

# --- Read Raw Data ---
# Reading the CSV with the final, correct schema
raw_pos_df = spark.read.format("csv") \
    .option("header", "true").option("inferSchema", "true") \
    .load(input_path)

# --- Transformations ---
# --- MODIFIED: Use the new, correct source column names: 'sku' and 'quantity' ---
cleaned_df = raw_pos_df \
    .withColumn("sku", F.upper(F.trim(F.col("sku")))) \
    .withColumn("quantity", F.col("quantity").cast("int"))

# --- MODIFIED: Group by 'sku'. We are no longer renaming this column. ---
# We also drop product_name from the grouping as it is descriptive data.
aggregated_sales_df = cleaned_df.groupBy("sku") \
    .agg(F.sum("quantity").alias("total_quantity_sold")) \
    .withColumn("date", F.lit(processing_date))

# --- Final Selection ---
# The 'sku' column name already matches our Redshift table, so no rename is needed.
final_df = aggregated_sales_df.select(
    F.col("date").cast("date").alias("date_key"),
    F.col("sku"),
    F.col("total_quantity_sold")
)

# --- Write to Staging Zone ---
final_df.write \
    .mode("overwrite") \
    .parquet(output_path)

job.commit()
```



Your jobs (3) Info

Job name	Type	Created by	Last modified	AWS Glue version
glue-job-2-reconcile-inventory	Glue ETL	Script	10/13/2025, 6:30:42 PM	5.0
glue-job-create-dim-products	Glue ETL	Script	10/13/2025, 5:54:37 PM	5.0
glue-job-1-stage-pos-sales	Glue ETL	Script	10/13/2025, 5:52:03 PM	5.0



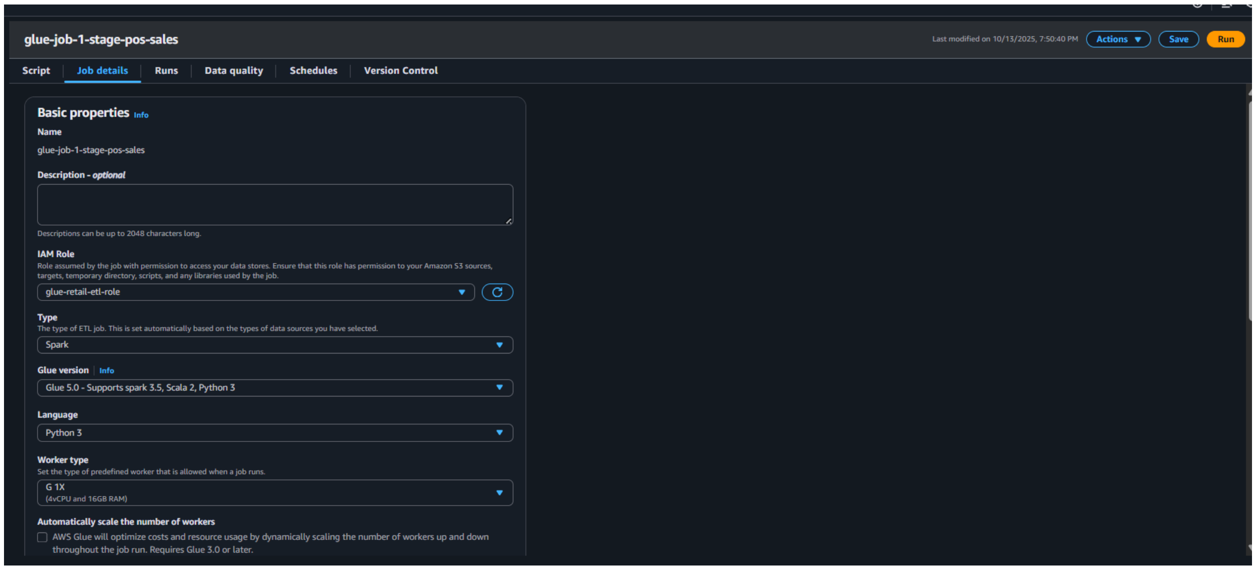
glue-job-1-stage-pos-sales

Last modified on 10/13/2025, 7:50:40 PM

Script

```python
import sys
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
import pyspark.sql.functions as F

# --- Initialization ---
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

# --- Parameters (Dynamic) ---
args = getResolvedOptions(sys.argv, ["JOB_NAME", "processing_date"])
job.init(args["JOB_NAME"], args)
processing_date = args["processing_date"]
s3_bucket = "my-retail-etl-project"

# --- Define Paths ---
input_path = f"s3://{s3_bucket}/raw/pos_sales/date={processing_date}/"
output_path = f"s3://{s3_bucket}/staging/pos_sales/date={processing_date}/"

# --- Read Raw Data ---
# Reading the CSV with the final, correct schema
raw_pos_df = spark.read.format("csv") \
    .option("header", "true").option("inferSchema", "true") \
    .load(input_path)

# --- Transformations ---
# --- MODIFIED: Use the new, correct source column names: 'sku' and 'quantity' ---
```

Python Ln 1, Col 1 Errors: 0 Warnings: 0



glue-job-1-stage-pos-sales

Last modified on 10/13/2025, 7:50:40 PM

Basic properties Info

Name
glue-job-1-stage-pos-sales

Description - *optional*

Descriptions can be up to 2048 characters long.

IAM Role
Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.
glue-retail-etl-role

Type
The type of ETL job. This is set automatically based on the types of data sources you have selected.
Spark

Glue version Info
Glue 5.0 - Supports spark 3.5, Scala 2, Python 3

Language
Python 3

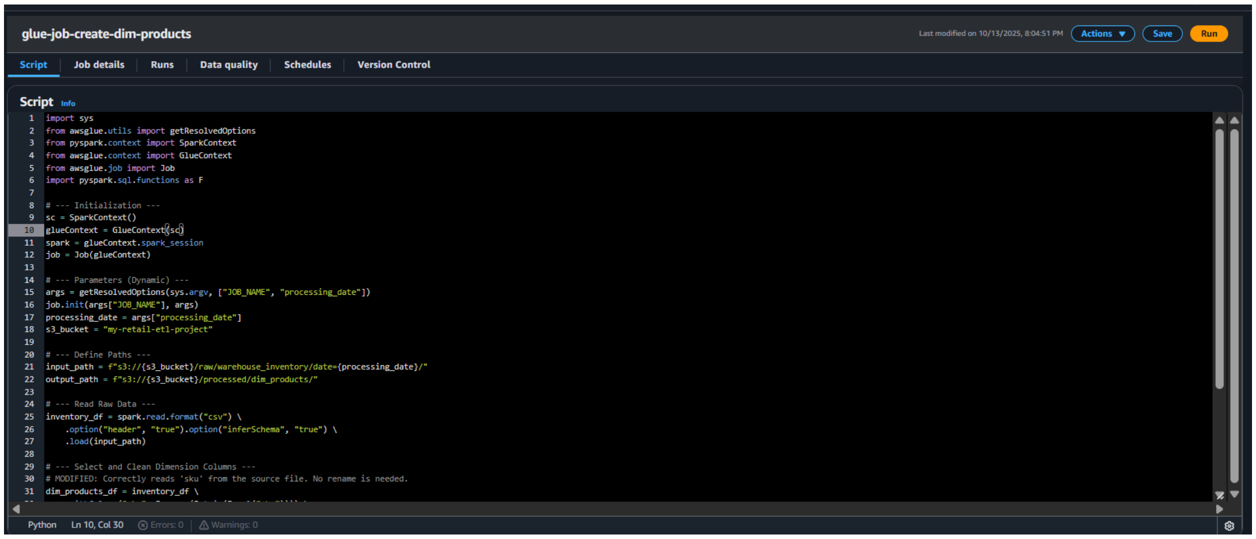
Worker type
Set the type of predefined worker that is allowed when a job runs.
G 1X
(4vCPU and 16GB RAM)

Automatically scale the number of workers
☐ AWS Glue will optimize costs and resource usage by dynamically scaling the number of workers up and down throughout the job run. Requires Glue 3.0 or later.

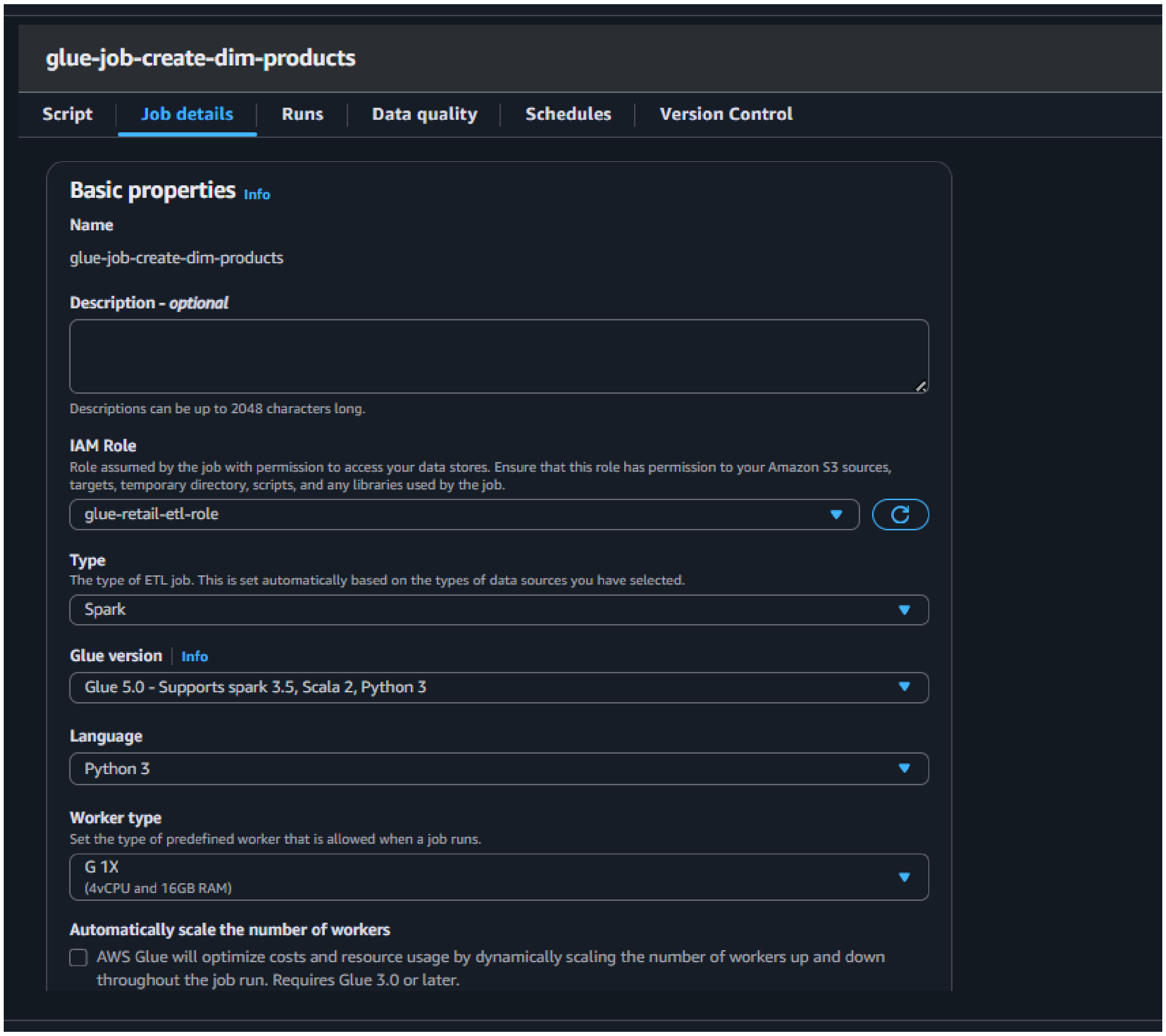
Step 4: Glue Job For dim-products table

glue-job-create-dim-products

```python
import sys
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
import pyspark.sql.functions as F

# --- Initialization ---
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

# --- Parameters (Dynamic) ---
args = getResolvedOptions(sys.argv, ["JOB_NAME", "processing_date"])
job.init(args["JOB_NAME"], args)
processing_date = args["processing_date"]
s3_bucket = "my-retail-etl-project"

# --- Define Paths ---
input_path = f"s3://{s3_bucket}/raw/warehouse_inventory/date={processing_date}/"
output_path = f"s3://{s3_bucket}/processed/dim_products/"

# --- Read Raw Data ---
inventory_df = spark.read.format("csv") \
    .option("header", "true").option("inferSchema", "true") \
    .load(input_path)

# --- Select and Clean Dimension Columns ---
# MODIFIED: Correctly reads 'sku' from the source file. No rename is needed.
dim_products_df = inventory_df \
    .withColumn("sku", F.upper(F.trim(F.col("sku")))) \
    .withColumn("product_name", F.trim(F.col("product_name"))) \
    .withColumn("category", F.trim(F.col("category"))) \
    .select("sku", "product_name", "category") \
    .dropDuplicates(["sku"])

# --- Write to Processed Zone ---
dim_products_df.coalesce(1).write \
    .mode("overwrite") \
    .parquet(output_path)

job.commit()
```



```python
import sys
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
import pyspark.sql.functions as F

# --- Initialization ---
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

# --- Parameters (Dynamic) ---
args = getResolvedOptions(sys.argv, ["JOB_NAME", "processing_date"])
job.init(args["JOB_NAME"], args)
processing_date = args["processing_date"]
s3_bucket = "my-retail-etl-project"

# --- Define Paths ---
input_path = f"s3://{s3_bucket}/raw/warehouse_inventory/date={processing_date}/"
output_path = f"s3://{s3_bucket}/processed/dim_products/"

# --- Read Raw Data ---
inventory_df = spark.read.format("csv") \
    .option("header", "true").option("inferSchema", "true") \
    .load(input_path)

# --- Select and Clean Dimension Columns ---
# MODIFIED: Correctly reads 'sku' from the source file. No rename is needed.
dim_products_df = inventory_df \
```



Basic properties

Name
glue-job-create-dim-products

Description - *optional*

Descriptions can be up to 2048 characters long.

IAM Role
Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.
glue-retail-etl-role

Type
The type of ETL job. This is set automatically based on the types of data sources you have selected.
Spark

Glue version
Glue 5.0 - Supports spark 3.5, Scala 2, Python 3

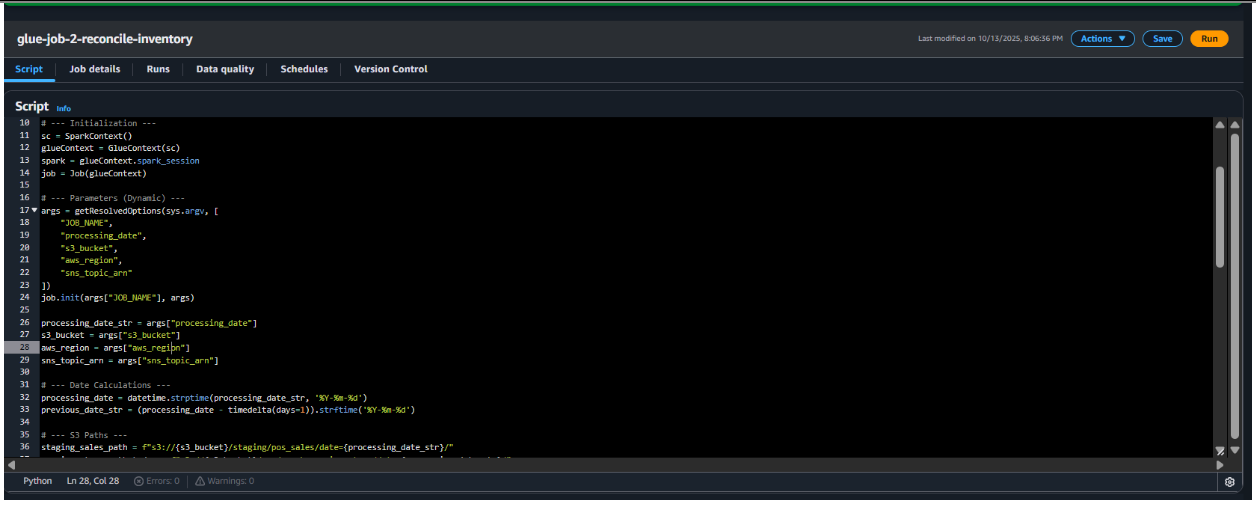
Language
Python 3

Worker type
Set the type of predefined worker that is allowed when a job runs.
G 1X
(4vCPU and 16GB RAM)

Automatically scale the number of workers
☐ AWS Glue will optimize costs and resource usage by dynamically scaling the number of workers up and down throughout the job run. Requires Glue 3.0 or later.

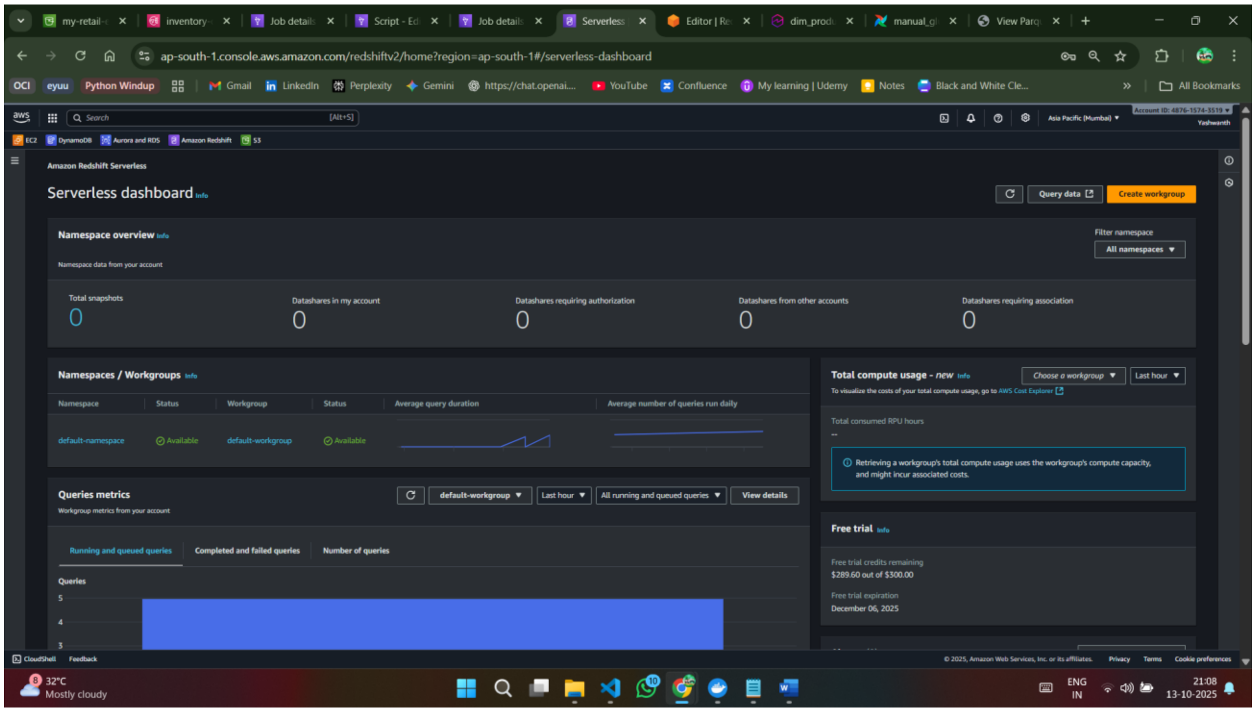
Step 5: Glue Job 2

```python
import sys
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
import pyspark.sql.functions as F
from datetime import datetime, timedelta
import boto3

# --- Initialization ---
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

# --- Parameters (Dynamic) ---
args = getResolvedOptions(sys.argv, [
    "JOB_NAME",
    "processing_date",
    "s3_bucket",
    "aws_region",
    "sns_topic_arn"
])
job.init(args["JOB_NAME"], args)

processing_date_str = args["processing_date"]
s3_bucket = args["s3_bucket"]
aws_region = args["aws_region"]
sns_topic_arn = args["sns_topic_arn"]

# --- Date Calculations ---
processing_date = datetime.strptime(processing_date_str, '%Y-%m-%d')
previous_date_str = (processing_date - timedelta(days=1)).strftime('%Y-%m-%d')

# --- S3 Paths ---
staging_sales_path = f"s3://{s3_bucket}/staging/pos_sales/date={processing_date_str}/"
raw_inventory_path_today = f"s3://{s3_bucket}/raw/warehouse_inventory/date={processing_date_str}/"
raw_inventory_path_yesterday = f"s3://{s3_bucket}/raw/warehouse_inventory/date={previous_date_str}/"
dim_products_path = f"s3://{s3_bucket}/processed/dim_products/"
processed_output_path =
f"s3://{s3_bucket}/processed/reconciled_inventory/date={processing_date_str}/"

# --- 1. Load DataFrames ---
daily_sales_df = spark.read.parquet(staging_sales_path)
opening_stock_df = spark.read.format("csv").option("header", "true").option("inferSchema",
"true").load(raw_inventory_path_yesterday).select(F.upper(F.trim(F.col("sku"))).alias("sku"),F.col("
stock_on_hand").alias("opening_stock"))
actual_closing_stock_df = spark.read.format("csv").option("header", "true").option("inferSchema",
"true").load(raw_inventory_path_today).select(F.upper(F.trim(F.col("sku"))).alias("sku"),F.col("stoc
k_on_hand").alias("actual_closing_stock"))
dim_products_df = spark.read.parquet(dim_products_path)

# --- 2. Join and Reconcile ---
inventory_df = opening_stock_df.join(actual_closing_stock_df, "sku", "full_outer")
reconciliation_df = inventory_df.join(daily_sales_df, "sku", "left")
reconciliation_with_names_df = reconciliation_df.join(dim_products_df.select("sku", "product_name"),
"sku", "left")

# --- 3. Calculate and Finalize ---
final_df = reconciliation_with_names_df.fillna(0, subset=['opening_stock', 'actual_closing_stock',
'total_quantity_sold']).withColumn("expected_closing_stock", (F.col("opening_stock") -
F.col("total_quantity_sold"))).withColumn("discrepancy", (F.col("actual_closing_stock") -
F.col("expected_closing_stock"))).withColumn("date", F.lit(processing_date_str))

# --- 4. Select and Rename Final Columns ---
final_df_selected =
final_df.select(F.col("date").cast("date").alias("date_key"),F.col("sku"),F.col("product_name"),F.co
l("opening_stock"),F.col("total_quantity_sold").alias("quantity_sold"),F.col("expected_closing_stock
"),F.col("actual_closing_stock"),F.col("discrepancy").alias("discrepancy_amount"))

# --- 5. Write to Processed Zone ---
final_df_selected.write.mode("overwrite").parquet(processed_output_path)

# --- 6. Check for Discrepancies and Send Notification ---
discrepancy_df = final_df_selected.filter(F.col("discrepancy_amount") != 0)
if discrepancy_df.count() > 0:
    print("Discrepancies found. Preparing to send SNS notification.")
    sns_client = boto3.client('sns', region_name=aws_region)
    discrepancy_examples = discrepancy_df.limit(5).collect()
    message = f"Inventory reconciliation for date {processing_date_str} found discrepancies.\n\n"
    message += f"Total items with discrepancies: {discrepancy_df.count()}\n\n"
    message += "Example Discrepancies:\n"
    for row in discrepancy_examples:
        product_name = row['product_name'] if row['product_name'] else "N/A"
        message += f"- SKU: {row['sku']}, Product: {product_name}, Discrepancy:
{row['discrepancy_amount']}\n"
    message += f"\nFull report available at: {processed_output_path}"
    subject = f"Alert: Inventory Discrepancy Found for {processing_date_str}"
    response = sns_client.publish(TopicArn=sns_topic_arn, Message=message, Subject=subject)
    print(f"SNS notification sent! Message ID: {response['MessageId']}")
else:
    print("No discrepancies found. No notification sent.")

job.commit()
```



```python
# --- Initialization ---
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

# --- Parameters (Dynamic) ---
args = getResolvedOptions(sys.argv, [
    "JOB_NAME",
    "processing_date",
    "s3_bucket",
    "aws_region",
    "sns_topic_arn"
])
job.init(args["JOB_NAME"], args)

processing_date_str = args["processing_date"]
s3_bucket = args["s3_bucket"]
aws_region = args["aws_region"]
sns_topic_arn = args["sns_topic_arn"]

# --- Date Calculations ---
processing_date = datetime.strptime(processing_date_str, '%Y-%m-%d')
previous_date_str = (processing_date - timedelta(days=1)).strftime('%Y-%m-%d')

# --- S3 Paths ---
staging_sales_path = f"s3://{s3_bucket}/staging/pos_sales/date={processing_date_str}/"
```

Step 6: Create a Redshift Serverless


ap-south-1.console.aws.amazon.com/redshiftv2/home?region=ap-south-1#/serverless-dashboard

Amazon Redshift Serverless

Serverless dashboard Info

Namespace overview Info

Namespace data from your account

Filter namespace: All namespaces

Total snapshots	Datashares in my account	Datashares requiring authorization	Datashares from other accounts	Datashares requiring association
0	0	0	0	0

Namespaces / Workgroups Info

Namespace	Status	Workgroup	Status	Average query duration	Average number of queries run daily
default-namespace	Available	default-workgroup	Available		

Total compute usage - *new* Info

To visualize the costs of your total compute usage, go to AWS Cost Explorer

Total consumed RPU hours

--

Retrieving a workgroup's total compute usage uses the workgroup's compute capacity, and might incur associated costs.

Queries metrics

Workgroup metrics from your account

default-workgroup · Last hour · All running and queued queries · View details

Running and queued queries · Completed and failed queries · Number of queries

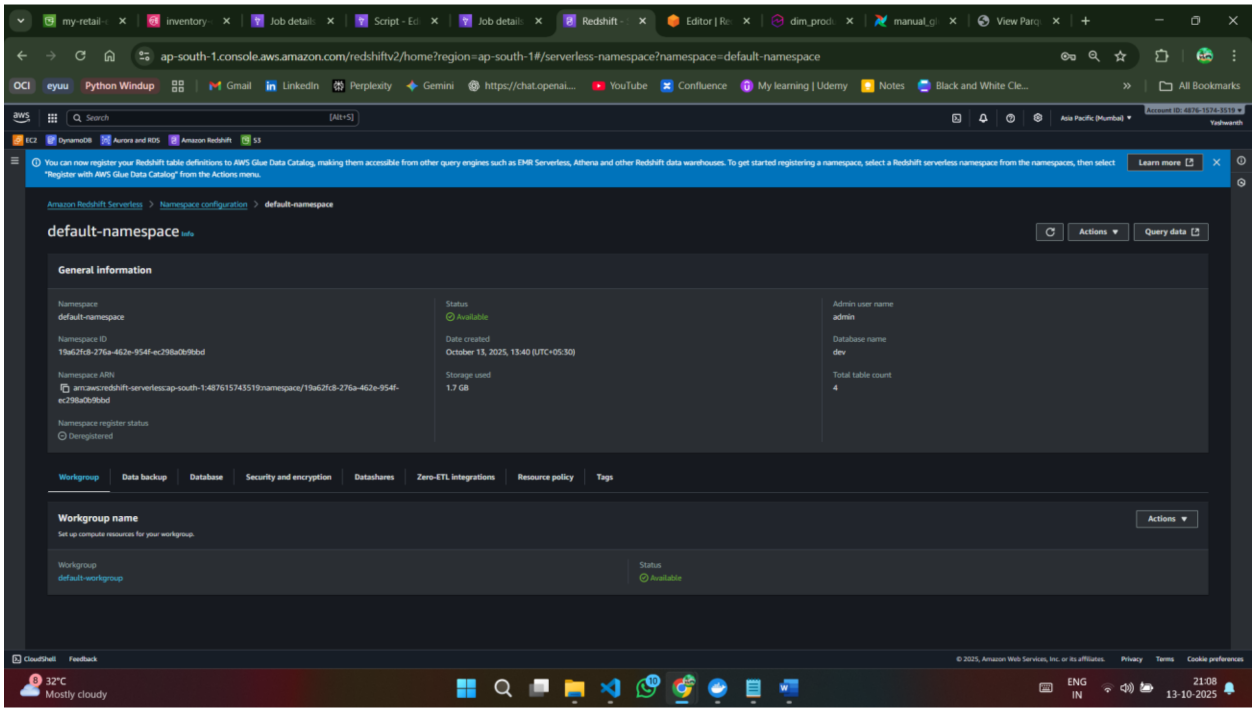
Free trial Info

Free trial credits remaining

$289.60 out of $300.00

Free trial expiration

December 06, 2025


ap-south-1.console.aws.amazon.com/redshiftv2/home?region=ap-south-1#/serverless-namespace?namespace=default-namespace

You can now register your Redshift table definitions to AWS Glue Data Catalog, making them accessible from other query engines such as EMR Serverless, Athena and other Redshift data warehouses. To get started registering a namespace, select a Redshift serverless namespace from the namespaces, then select "Register with AWS Glue Data Catalog" from the Actions menu.

Amazon Redshift Serverless > Namespace configuration > default-namespace

default-namespace Info

General information

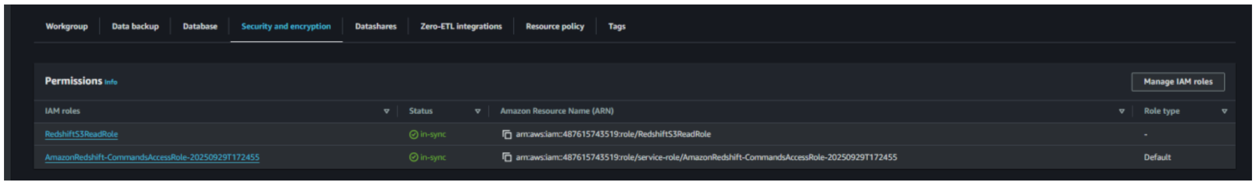
Namespace: default-namespace	Status: Available	Admin user name: admin
Namespace ID: 19a62fc8-276a-462e-954f-ec298a0b9bbd	Date created: October 13, 2025, 13:40 (UTC+05:30)	Database name: dev
Namespace ARN: arn:aws:redshift-serverless:ap-south-1:487615743519:namespace/19a62fc8-276a-462e-954f-ec298a0b9bbd	Storage used: 1.7 GB	Total table count: 4
Namespace register status: Deregistered		

Workgroup · Data backup · Database · Security and encryption · Datashares · Zero-ETL integrations · Resource policy · Tags

Workgroup name

Set up compute resources for your workgroup.

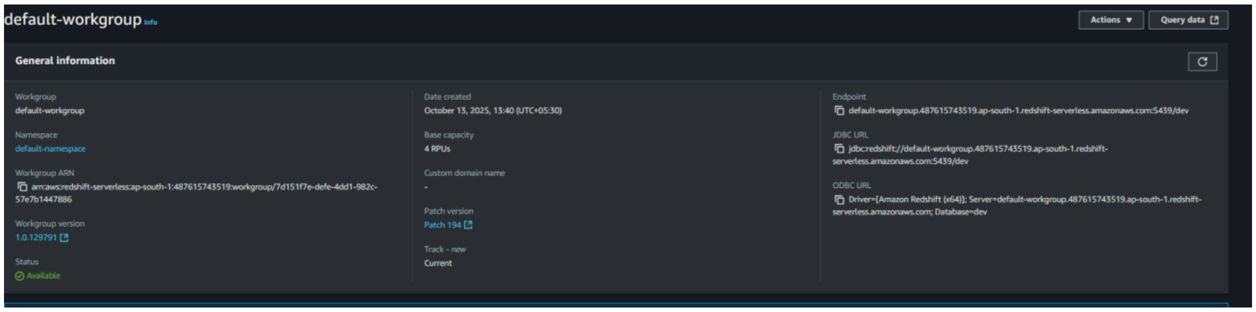
Workgroup	Status
default-workgroup	Available



Workgroup | Data backup | Database | Security and encryption | Datashares | Zero-ETL integrations | Resource policy | Tags

Permissions Info

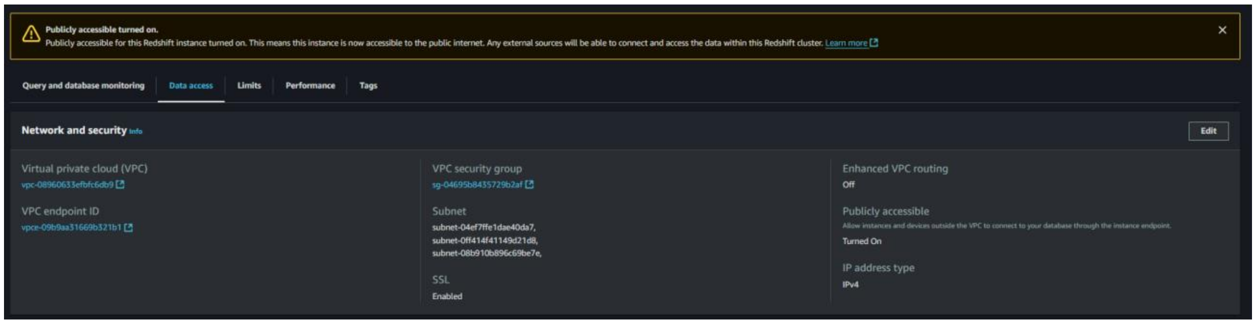
IAM roles	Status	Amazon Resource Name (ARN)	Role type
RedshiftS3ReadRole	in-sync	arn:aws:iam::487615743519:role/RedshiftS3ReadRole	-
AmazonRedshift-CommandsAccessRole-20250929T172455	in-sync	arn:aws:iam::487615743519:role/service-role/AmazonRedshift-CommandsAccessRole-20250929T172455	Default



default-workgroup Info

General information

Workgroup default-workgroup	Date created October 13, 2025, 13:40 (UTC+05:30)	Endpoint default-workgroup.487615743519.ap-south-1.redshift-serverless.amazonaws.com:5439/dev
Namespace default-namespace	Base capacity 4 RPUs	JDBC URL jdbc:redshift://default-workgroup.487615743519.ap-south-1.redshift-serverless.amazonaws.com:5439/dev
Workgroup ARN arn:aws:redshift-serverless:ap-south-1:487615743519:workgroup/7d151f7e-defe-4dd1-982c-57e7b1447886	Custom domain name -	ODBC URL Driver={Amazon Redshift (x64)}; Server=default-workgroup.487615743519.ap-south-1.redshift-serverless.amazonaws.com; Database=dev
Workgroup version 1.0.129791	Patch version Patch 194	
Status Available	Track - new Current	



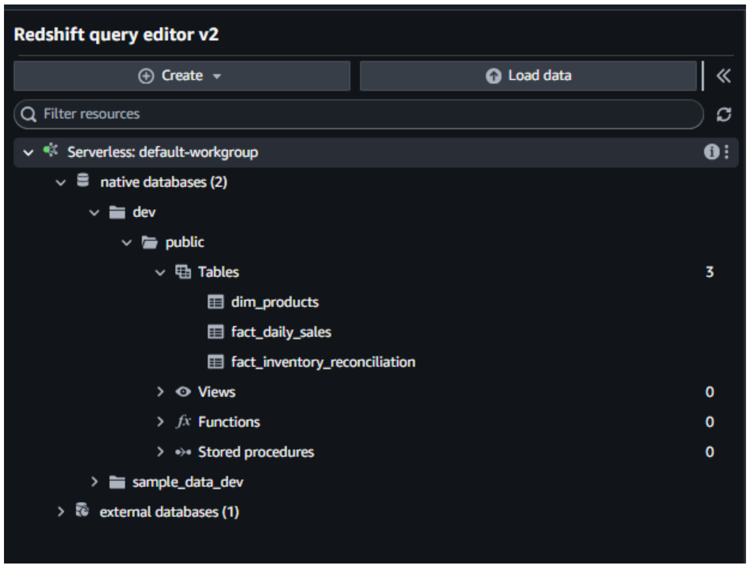
Publicly accessible turned on.
Publicly accessible for this Redshift instance turned on. This means this instance is now accessible to the public internet. Any external sources will be able to connect and access the data within this Redshift cluster. Learn more

Query and database monitoring | Data access | Limits | Performance | Tags

Network and security Info

Virtual private cloud (VPC) vpc-08960633efbfc6db9	VPC security group sg-04695b8435729b2af	Enhanced VPC routing Off
VPC endpoint ID vpce-09b9aa31669b321b1	Subnet subnet-04ef7ffe1dae40da7, subnet-0ff414f41149d21d8, subnet-08b910b896c69be7e,	Publicly accessible Allow instances and devices outside the VPC to connect to your database through the instance endpoint. Turned On
	SSL Enabled	IP address type IPv4

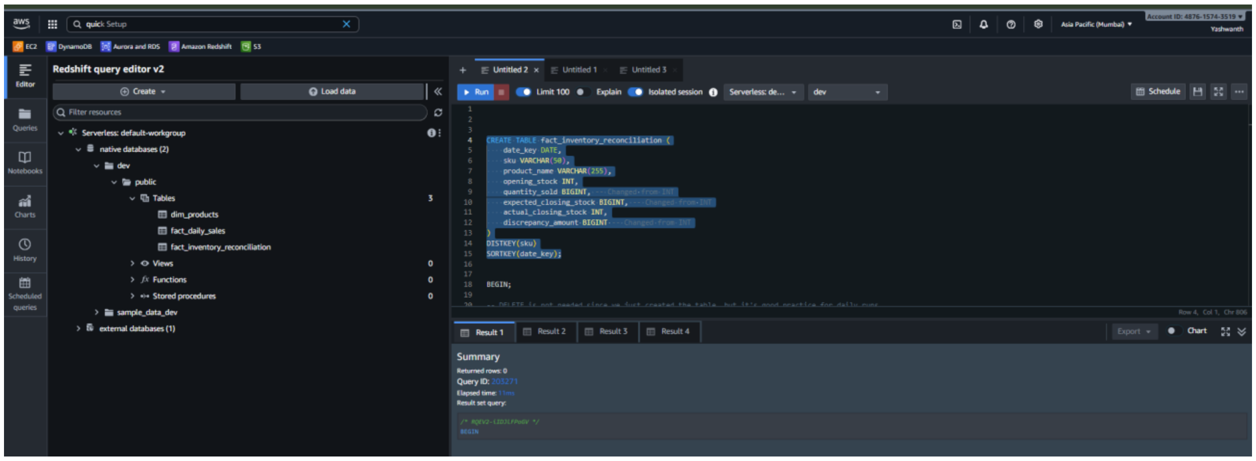
Step 7: Create Tables in Redshift



Redshift query editor v2

```
Serverless: default-workgroup
  native databases (2)
    dev
      public
        Tables                     3
          dim_products
          fact_daily_sales
          fact_inventory_reconciliation
        Views                      0
        Functions                  0
        Stored procedures          0
    sample_data_dev
  external databases (1)
```

```sql
CREATE TABLE fact_inventory_reconciliation (
    date_key DATE,
    sku VARCHAR(50),
    product_name VARCHAR(255),
    opening_stock INT,
    quantity_sold BIGINT, -- Changed from INT
    expected_closing_stock BIGINT, -- Changed from INT
    actual_closing_stock INT,
    discrepancy_amount BIGINT -- Changed from INT
)
DISTKEY(sku)
SORTKEY(date_key);

CREATE TABLE IF NOT EXISTS dim_products (
    sku VARCHAR(50) NOT NULL,
    product_name VARCHAR(255),
    category VARCHAR(100)
)
DISTSTYLE ALL      -- Use DISTSTYLE ALL because it's a small table that joins to large fact tables.
SORTKEY(sku);

CREATE TABLE IF NOT EXISTS fact_daily_sales (
    date_key DATE,
    sku VARCHAR(50),
    product_name VARCHAR(255),
    total_quantity_sold BIGINT -- Using BIGINT to match the Spark sum() output
)
DISTKEY(sku)
SORTKEY(date_key);
```

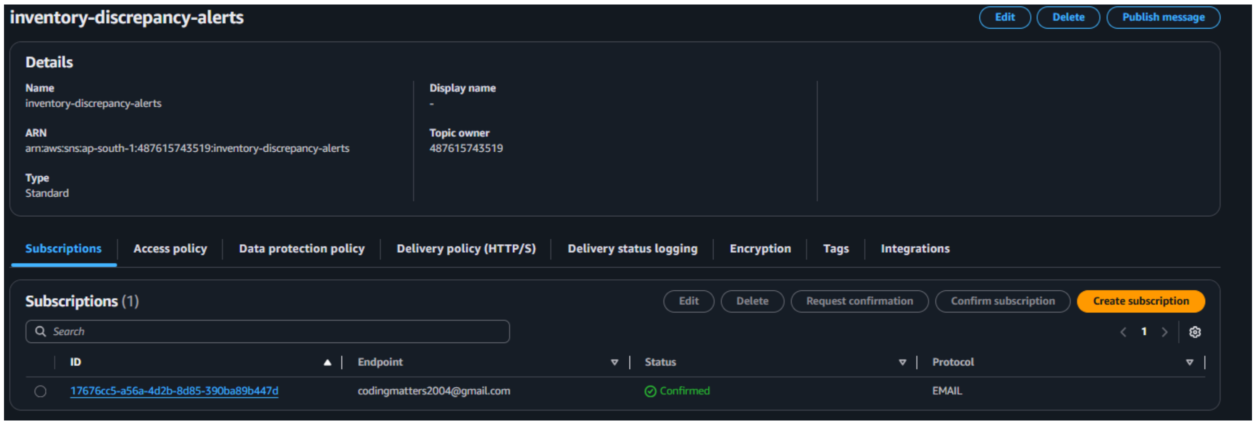


```sql



CREATE TABLE fact_inventory_reconciliation (
    date_key DATE,
    sku VARCHAR(50),
    product_name VARCHAR(255),
    opening_stock INT,
    quantity_sold BIGINT, -- Changed from INT
    expected_closing_stock BIGINT, -- Changed from INT
    actual_closing_stock INT,
    discrepancy_amount BIGINT -- Changed from INT
)
DISTKEY(sku)
SORTKEY(date_key);


BEGIN;

-- DELETE is not needed since we just created the table, but it's good practice for daily runs
```

Summary
Returned rows: 0
Query ID: 203271
Elapsed time: 11ms
Result set query:

```sql
/* RQEV2-LIDJLFPoGV */
BEGIN
```

Step 8: Create Sns and Subscribe to email

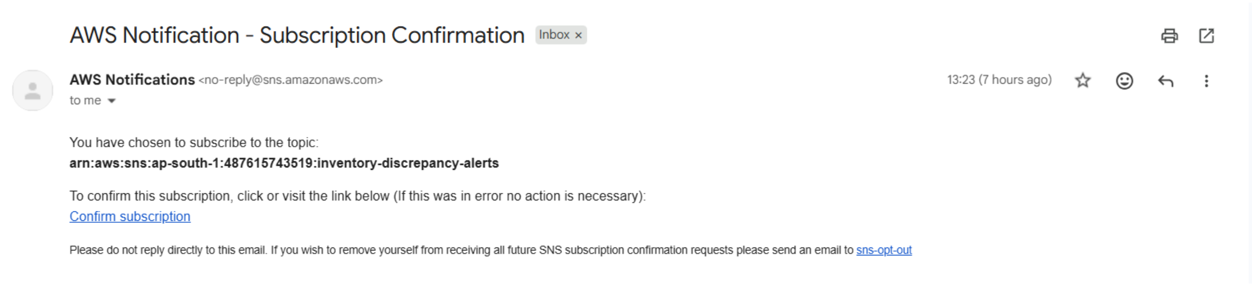

inventory-discrepancy-alerts

Details

Name inventory-discrepancy-alerts	**Display name** -
ARN arn:aws:sns:ap-south-1:487615743519:inventory-discrepancy-alerts	**Topic owner** 487615743519
Type Standard	

Subscriptions (1)

ID	Endpoint	Status	Protocol
17676cc5-a56a-4d2b-8d85-390ba89b447d	codingmatters2004@gmail.com	Confirmed	EMAIL


AWS Notification - Subscription Confirmation

AWS Notifications <no-reply@sns.amazonaws.com>
to me

13:23 (7 hours ago)

You have chosen to subscribe to the topic:
arn:aws:sns:ap-south-1:487615743519:inventory-discrepancy-alerts

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out

Project Resource Summary

Here is a breakdown of the AWS services utilized in this data pipeline and the rationale for their use.

---

1. IAM (Identity and Access Management)

- **Resource:** An **IAM Role**.
- **Purpose:** Created to grant the necessary permissions for AWS services to communicate with each other securely. For example, this role allows the AWS Glue jobs to read data from the S3 bucket and write the processed data to the Amazon Redshift warehouse without needing to manage static credentials.

---

2. S3 (Simple Storage Service)

- **Resource:** An **S3 Bucket**.
- **Purpose:** Serves as the central **data lake** for this project. It is used to store raw source data (like daily sales files) and potentially intermediate, processed data before it's loaded into the data warehouse. Its scalability and cost-effectiveness make it ideal for storing large volumes of data.

---

3. AWS Glue

- **Resource:** Three distinct **AWS Glue ETL jobs**.
- **Purpose:** Used as the serverless Extract, Transform, and Load (ETL) service to process the raw data from S3.
  - **Glue Job (dim_products):** This job is responsible for processing the product-related data to create and populate the dim_products dimension table. It cleans, transforms, and standardizes product information.
  - **Glue Job 1 (fact_daily_sales):** This job processes the daily transactional sales data, aggregating and structuring it to be loaded into the fact_daily_sales table.
  - **Glue Job 2 (reconciliation):** This job performs a reconciliation process, likely comparing inventory or sales data from different sources to ensure accuracy and consistency. The output is loaded into the fact_inventory_reconciliation table.

---

4. Amazon Redshift Serverless

- **Resource:** A **Redshift Serverless** data warehouse.
- **Purpose:** Acts as the project's central **data warehouse**. It stores the final, structured, and transformed data in a columnar format, which is highly optimized for fast analytical queries and business intelligence (BI) reporting. The serverless option was chosen to automatically scale compute resources based on workload, simplifying management and optimizing costs.

---

5. Redshift Tables

- **Resource:** Three tables within the Redshift warehouse.
- **Purpose:** To store the cleaned and processed data in a structured, queryable format.
  - **dim_products:** A **dimension table** that stores descriptive attributes about the products (e.g., name, category, brand).
  - **fact_daily_sales:** A **fact table** that stores quantitative sales metrics (e.g., units sold, revenue) on a daily basis.
  - **fact_inventory_reconciliation:** A **fact table** containing the results of the data reconciliation process, highlighting discrepancies or confirming consistency.

---

6. SNS (Simple Notification Service)

- **Resource:** An **SNS Topic** with an email subscription.
- **Purpose:** Implemented as an alerting and monitoring mechanism. It sends out automated email notifications to subscribers upon the success or failure of the Glue jobs or other pipeline events. This ensures that stakeholders are immediately aware of the pipeline's status.

Apache Airflow?

Airflow is used for **workflow orchestration**. While we have created individual jobs (like your Glue jobs), a real-world project requires you to run them in a specific order, on a schedule, and handle any failures gracefully.

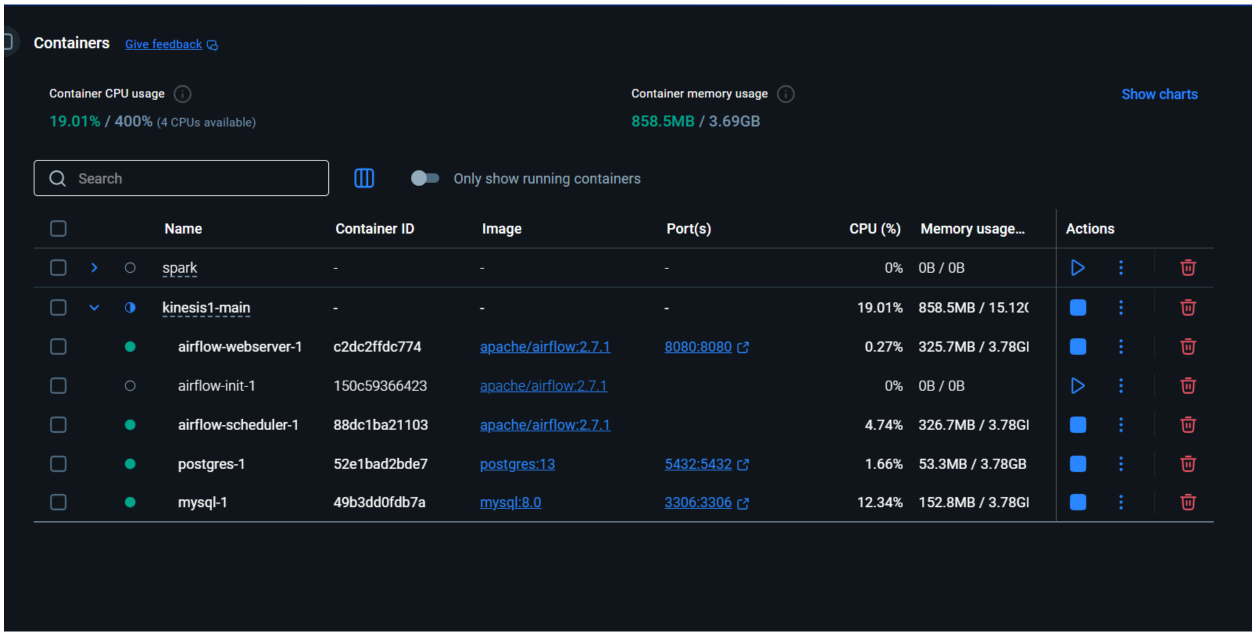
Airflow allows you to:

- **Schedule & Automate:** Automatically run your entire data pipeline at a specific time (e.g., daily at 2 AM) without manual intervention.
- **Manage Dependencies:** Define the exact order of operations. For example, you can ensure your fact_daily_sales and dim_products jobs complete successfully **before** the reconciliation job begins.
- **Monitor & Alert:** Provides a powerful user interface to visualize your pipeline's progress, check logs for each step, and send alerts when tasks fail.
- **Retry & Recover:** Automatically retries a failed task a set number of times, which can resolve temporary issues without waking you up in the middle of the night.
- **Scalability:** Manages complex workflows with many steps and dependencies, which is difficult to do with simple scripts or cron jobs.

Essentially, Airflow acts as the "brain" or "conductor" of your data pipeline, ensuring every part runs correctly, in the right order, and at the right time.

Set Up an Airflow Environment:

Docker Airflow: Build locally, deploy anywhere, and orchestrate with confidence.



Containers Give feedback

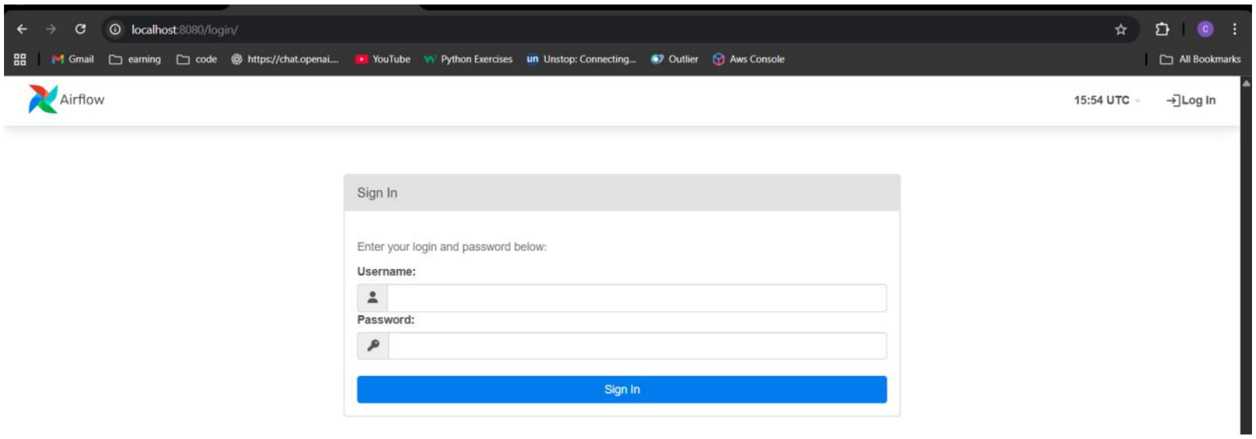
Container CPU usage 19.01% / 400% (4 CPUs available)

Container memory usage 858.5MB / 3.69GB

Show charts

Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Actions
spark	-	-	-	0%	0B / 0B	
kinesis1-main	-	-	-	19.01%	858.5MB / 15.12(	
airflow-webserver-1	c2dc2ffdc774	apache/airflow:2.7.1	8080:8080	0.27%	325.7MB / 3.78Gl	
airflow-init-1	150c59366423	apache/airflow:2.7.1		0%	0B / 0B	
airflow-scheduler-1	88dc1ba21103	apache/airflow:2.7.1		4.74%	326.7MB / 3.78Gl	
postgres-1	52e1bad2bde7	postgres:13	5432:5432	1.66%	53.3MB / 3.78GB	
mysql-1	49b3dd0fdb7a	mysql:8.0	3306:3306	12.34%	152.8MB / 3.78Gl	



localhost:8080/login/

Airflow

15:54 UTC

Log In

Sign In

Enter your login and password below:

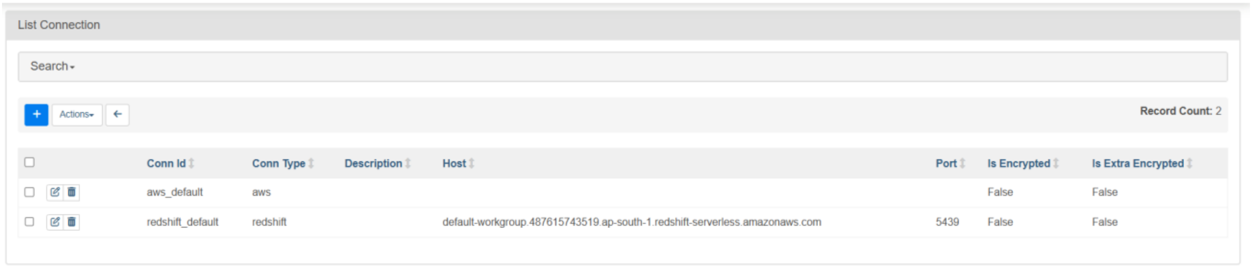
Username:

Password:

Sign In

Define Connections:

- In the Airflow UI, you configure connections to your external systems. You would create an **AWS Connection** that gives Airflow the permissions (via an IAM AccessKeys) to interact with your S3, Glue, and Redshift services.

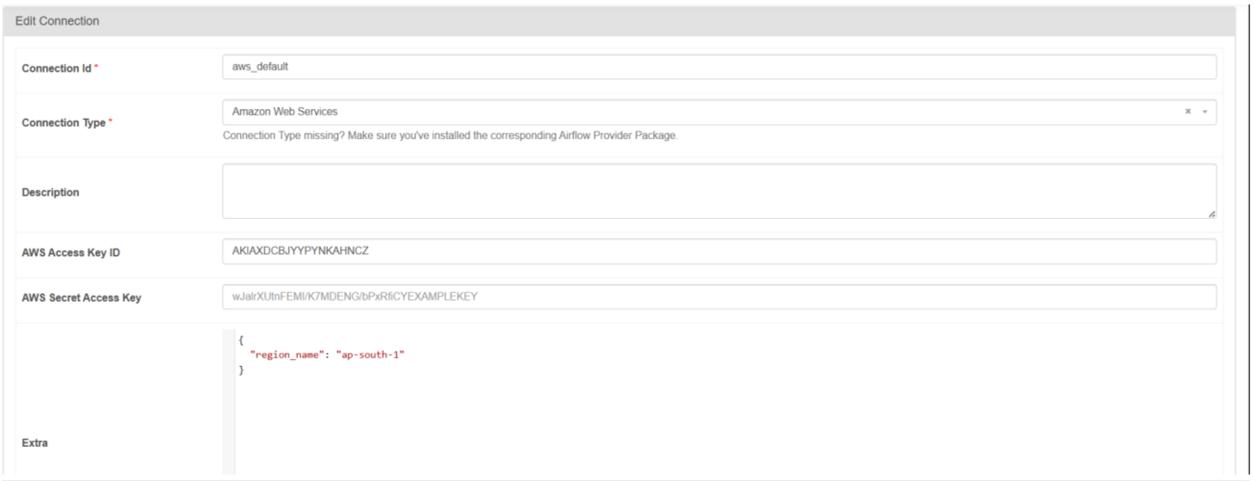

List Connection

Search

Actions

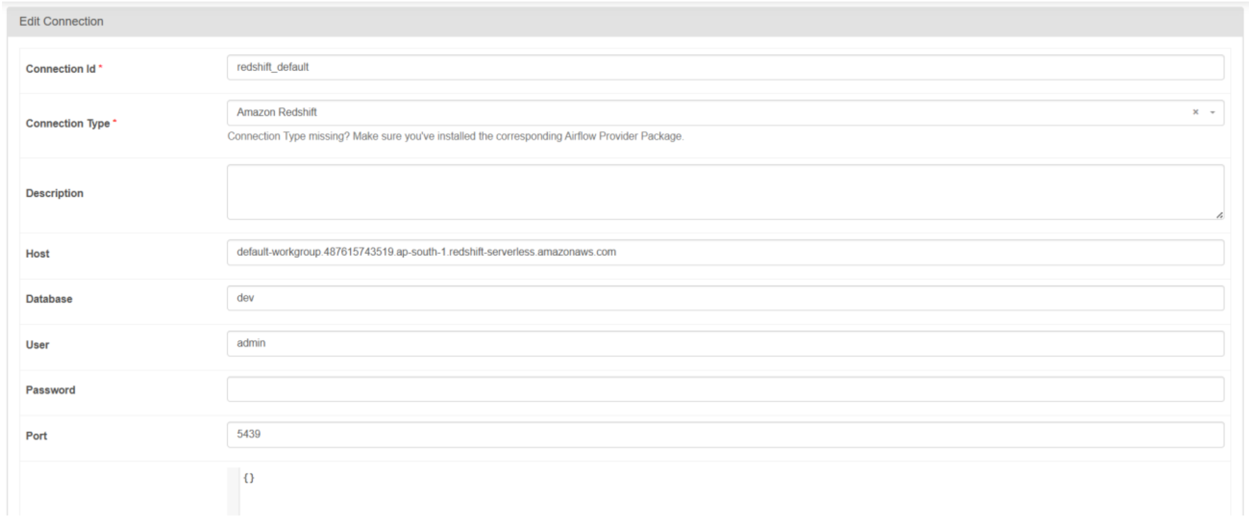
Record Count: 2

	Conn Id	Conn Type	Description	Host	Port	Is Encrypted	Is Extra Encrypted
	aws_default	aws				False	False
	redshift_default	redshift		default-workgroup.487615743519.ap-south-1.redshift-serverless.amazonaws.com	5439	False	False


Edit Connection

Connection Id *	aws_default
Connection Type *	Amazon Web Services Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.
Description	
AWS Access Key ID	AKIAXDCBJYYPYNKAHNCZ
AWS Secret Access Key	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Extra	

```
{
  "region_name": "ap-south-1"
}
```

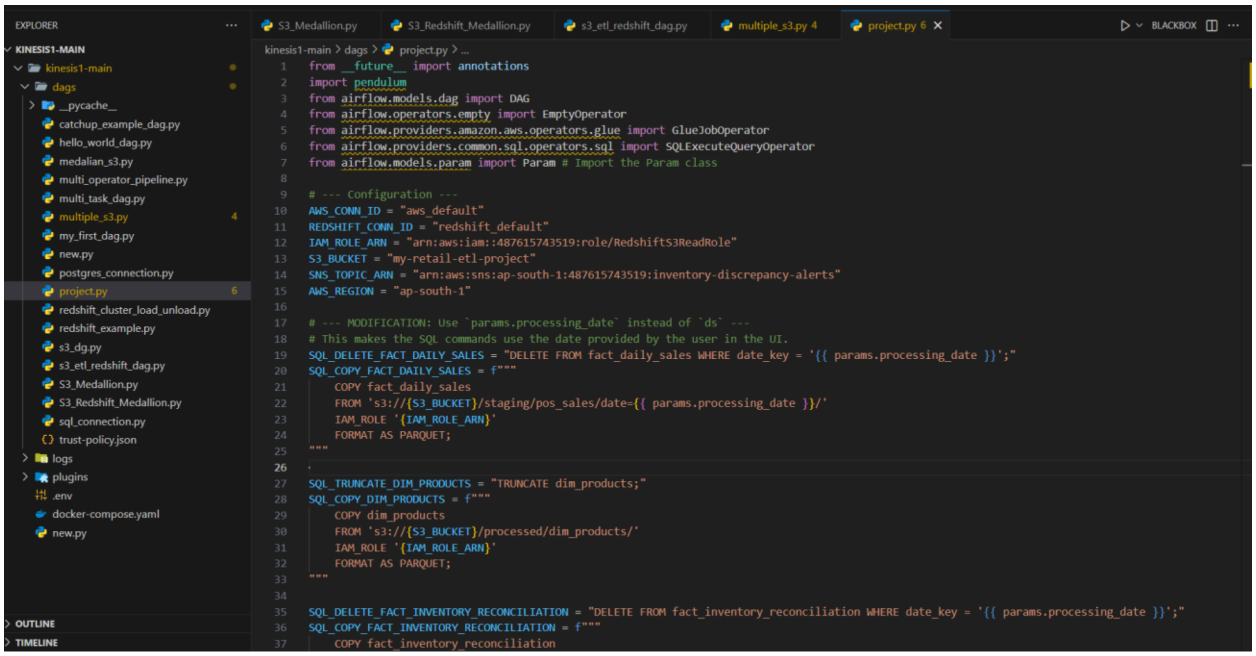

Edit Connection

Connection Id *	redshift_default
Connection Type *	Amazon Redshift Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.
Description	
Host	default-workgroup.487615743519.ap-south-1.redshift-serverless.amazonaws.com
Database	dev
User	admin
Password	
Port	5439

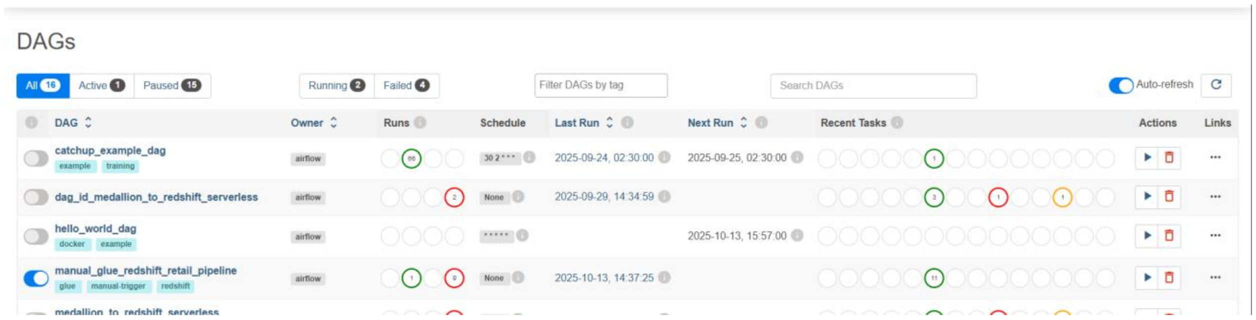
```
{}
```

Airflow Dag:

```
from __future__ import annotations
import pendulum
from airflow.models.dag import DAG
from airflow.operators.empty import EmptyOperator
from airflow.providers.amazon.aws.operators.glue import GlueJobOperator
from airflow.providers.common.sql.operators.sql import SQLExecuteQueryOperator
from airflow.models.param import Param # Import the Param class

# --- Configuration ---
AWS_CONN_ID = "aws_default"
REDSHIFT_CONN_ID = "redshift_default"
IAM_ROLE_ARN = "arn:aws:iam::487615743519:role/RedshiftS3ReadRole"
S3_BUCKET = "my-retail-etl-project"
SNS_TOPIC_ARN = "arn:aws:sns:ap-south-1:487615743519:inventory-discrepancy-alerts"
AWS_REGION = "ap-south-1"

# --- MODIFICATION: Use `params.processing_date` instead of `ds` ---
# This makes the SQL commands use the date provided by the user in the UI.
SQL_DELETE_FACT_DAILY_SALES = "DELETE FROM fact_daily_sales WHERE date_key = '{{
params.processing_date }}';"
SQL_COPY_FACT_DAILY_SALES = f"""
    COPY fact_daily_sales
    FROM 's3://{S3_BUCKET}/staging/pos_sales/date={{ params.processing_date }}/'
    IAM_ROLE '{IAM_ROLE_ARN}'
    FORMAT AS PARQUET;
"""

SQL_TRUNCATE_DIM_PRODUCTS = "TRUNCATE dim_products;"
SQL_COPY_DIM_PRODUCTS = f"""
    COPY dim_products
    FROM 's3://{S3_BUCKET}/processed/dim_products/'
    IAM_ROLE '{IAM_ROLE_ARN}'
    FORMAT AS PARQUET;
"""

SQL_DELETE_FACT_INVENTORY_RECONCILIATION = "DELETE FROM fact_inventory_reconciliation WHERE date_key
= '{{ params.processing_date }}';"
SQL_COPY_FACT_INVENTORY_RECONCILIATION = f"""
    COPY fact_inventory_reconciliation
    FROM 's3://{S3_BUCKET}/processed/reconciled_inventory/date={{ params.processing_date }}/'
    IAM_ROLE '{IAM_ROLE_ARN}'
    FORMAT AS PARQUET;
"""

with DAG(
    dag_id="manual_glue_redshift_retail_pipeline",
    start_date=pendulum.datetime(2025, 1, 1, tz="UTC"),
    schedule=None,
    catchup=False,
    doc_md="""
    ## Manual Retail Data Pipeline
    Orchestrates AWS Glue jobs and Redshift loads.
    Click the play button and enter a `processing_date` to run.
    """,
    # --- MODIFICATION: Define a parameter for the user to enter ---
    params={
        "processing_date": Param(
            "2025-10-12", # This is the default value that will appear in the box
            type="string",
            title="Processing Date",
            description="The date for which to run the pipeline (format: YYYY-MM-DD).",
        )
    },
    tags=["glue", "redshift", "manual-trigger"],
) as dag:

    start = EmptyOperator(task_id="start")

    # --- MODIFICATION: Use `params.processing_date` in script_args ---
    glue_job_stage_pos_sales = GlueJobOperator(task_id="glue_job_stage_pos_sales", job_name="glue-
job-1-stage-pos-sales", aws_conn_id=AWS_CONN_ID, script_args={"--processing_date": "{{
params.processing_date }}"})
    glue_job_create_dim_products = GlueJobOperator(task_id="glue_job_create_dim_products",
job_name="glue-job-create-dim-products", aws_conn_id=AWS_CONN_ID, script_args={"--processing_date":
"{{ params.processing_date }}"})
    glue_job_reconcile_inventory = GlueJobOperator(
        task_id="glue_job_reconcile_inventory",
        job_name="glue-job-2-reconcile-inventory",
        aws_conn_id=AWS_CONN_ID,
        script_args={
            "--processing_date": "{{ params.processing_date }}",
            "--s3_bucket": S3_BUCKET,
            "--aws_region": AWS_REGION,
            "--sns_topic_arn": SNS_TOPIC_ARN
        }
    )

    # Redshift tasks now use the single-statement SQL variables
    delete_fact_daily_sales = SQLExecuteQueryOperator(task_id="delete_fact_daily_sales",
conn_id=REDSHIFT_CONN_ID, sql=SQL_DELETE_FACT_DAILY_SALES)
    load_fact_daily_sales = SQLExecuteQueryOperator(task_id="load_fact_daily_sales_to_redshift",
conn_id=REDSHIFT_CONN_ID, sql=SQL_COPY_FACT_DAILY_SALES)

    truncate_dim_products = SQLExecuteQueryOperator(task_id="truncate_dim_products",
conn_id=REDSHIFT_CONN_ID, sql=SQL_TRUNCATE_DIM_PRODUCTS)
    load_dim_products = SQLExecuteQueryOperator(task_id="load_dim_products_to_redshift",
conn_id=REDSHIFT_CONN_ID, sql=SQL_COPY_DIM_PRODUCTS)

    delete_fact_inventory_reconciliation =
SQLExecuteQueryOperator(task_id="delete_fact_inventory_reconciliation", conn_id=REDSHIFT_CONN_ID,
sql=SQL_DELETE_FACT_INVENTORY_RECONCILIATION)
    load_fact_inventory_reconciliation =
SQLExecuteQueryOperator(task_id="load_fact_inventory_reconciliation_to_redshift",
conn_id=REDSHIFT_CONN_ID, sql=SQL_COPY_FACT_INVENTORY_RECONCILIATION)

    end = EmptyOperator(task_id="end")

    # --- Dependencies (Unchanged) ---
    start >> [glue_job_stage_pos_sales, glue_job_create_dim_products]
    glue_job_stage_pos_sales >> delete_fact_daily_sales >> load_fact_daily_sales
    glue_job_create_dim_products >> truncate_dim_products >> load_dim_products
    [glue_job_stage_pos_sales, glue_job_create_dim_products] >> glue_job_reconcile_inventory
    glue_job_reconcile_inventory >> delete_fact_inventory_reconciliation >>
load_fact_inventory_reconciliation
    [load_fact_daily_sales, load_dim_products, load_fact_inventory_reconciliation] >> end
```

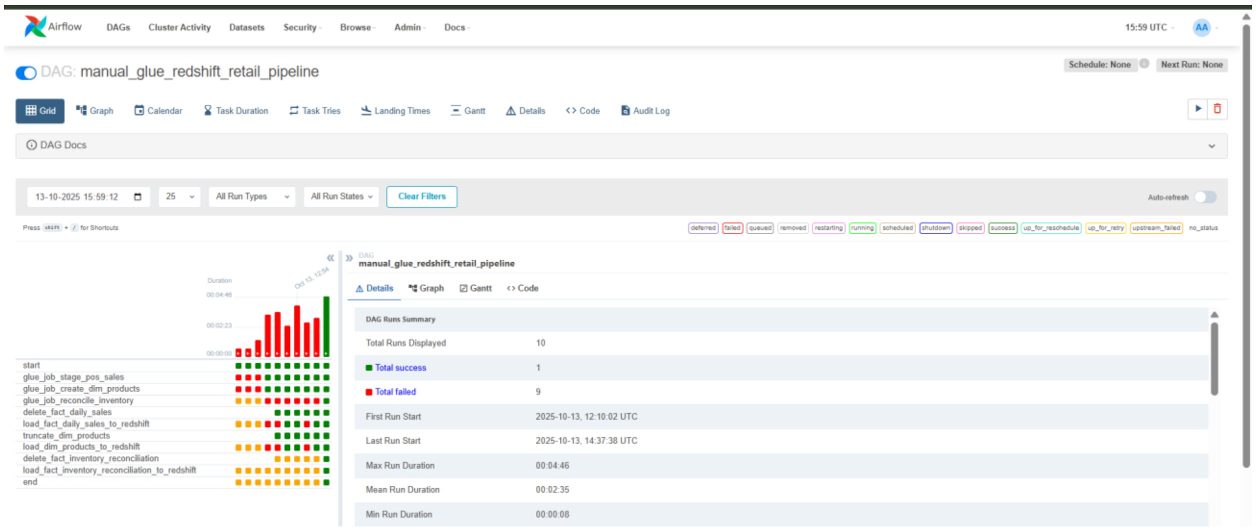


```
from __future__ import annotations
import pendulum
from airflow.models.dag import DAG
from airflow.operators.empty import EmptyOperator
from airflow.providers.amazon.aws.operators.glue import GlueJobOperator
from airflow.providers.common.sql.operators.sql import SQLExecuteQueryOperator
from airflow.models.param import Param # Import the Param class

# --- Configuration ---
AWS_CONN_ID = "aws_default"
REDSHIFT_CONN_ID = "redshift_default"
IAM_ROLE_ARN = "arn:aws:iam::487615743519:role/RedshiftS3ReadRole"
S3_BUCKET = "my-retail-etl-project"
SNS_TOPIC_ARN = "arn:aws:sns:ap-south-1:487615743519:inventory-discrepancy-alerts"
AWS_REGION = "ap-south-1"

# --- MODIFICATION: Use `params.processing_date` instead of `ds` ---
# This makes the SQL commands use the date provided by the user in the UI.
SQL_DELETE_FACT_DAILY_SALES = "DELETE FROM fact_daily_sales WHERE date_key = '{{ params.processing_date }}';"
SQL_COPY_FACT_DAILY_SALES = f"""
    COPY fact_daily_sales
    FROM 's3://{S3_BUCKET}/staging/pos_sales/date={{ params.processing_date }}/'
    IAM_ROLE '{IAM_ROLE_ARN}'
    FORMAT AS PARQUET;
"""

SQL_TRUNCATE_DIM_PRODUCTS = "TRUNCATE dim_products;"
SQL_COPY_DIM_PRODUCTS = f"""
    COPY dim_products
    FROM 's3://{S3_BUCKET}/processed/dim_products/'
    IAM_ROLE '{IAM_ROLE_ARN}'
    FORMAT AS PARQUET;
"""

SQL_DELETE_FACT_INVENTORY_RECONCILIATION = "DELETE FROM fact_inventory_reconciliation WHERE date_key = '{{ params.processing_date }}';"
SQL_COPY_FACT_INVENTORY_RECONCILIATION = f"""
    COPY fact_inventory_reconciliation
```



DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
catchup_example_dag (example, training)	airflow	66	30 2 * * *	2025-09-24, 02:30:00	2025-09-25, 02:30:00	1		...
dag_id_medallion_to_redshift_serverless	airflow	2	None	2025-09-29, 14:34:59		3, 1, 1		...
hello_world_dag (docker, example)	airflow		* * * * *		2025-10-13, 15:57:00			...
manual_glue_redshift_retail_pipeline (glue, manual-trigger, redshift)	airflow	1, 9	None	2025-10-13, 14:37:25		11		...
medallion_to_redshift_serverless								

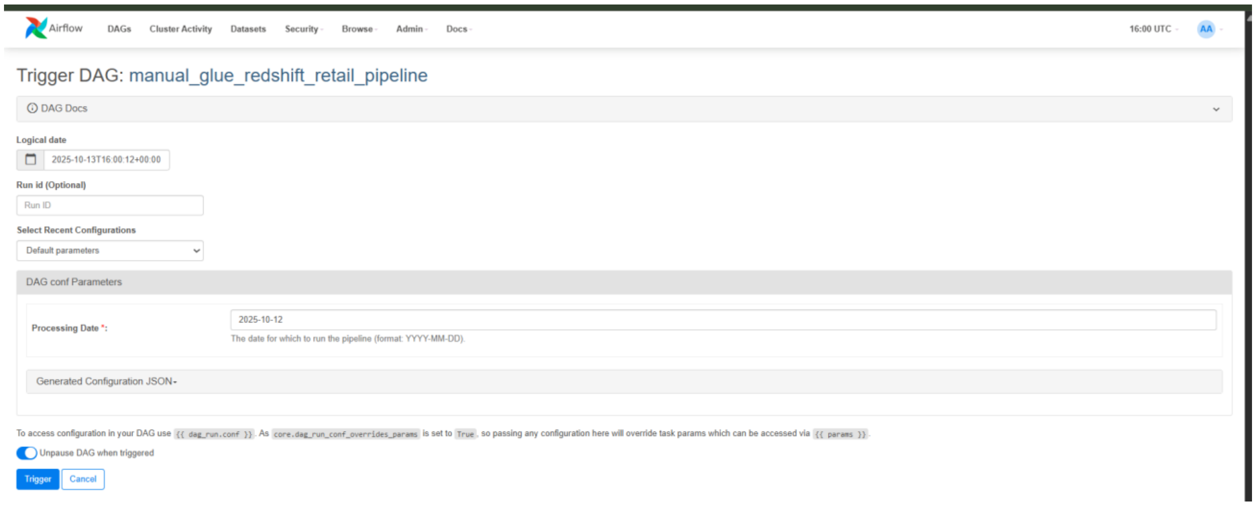


DAG: manual_glue_redshift_retail_pipeline

DAG Runs Summary	
Total Runs Displayed	10
🟩 Total success	1
🟥 Total failed	9
First Run Start	2025-10-13, 12:10:02 UTC
Last Run Start	2025-10-13, 14:37:38 UTC
Max Run Duration	00:04:46
Mean Run Duration	00:02:35
Min Run Duration	00:00:08

Run Dag:

Choose the Date.



Trigger DAG: manual_glue_redshift_retail_pipeline

DAG Docs

Logical date: 2025-10-13T16:00:12+00:00

Run id (Optional): Run ID

Select Recent Configurations: Default parameters

DAG conf Parameters

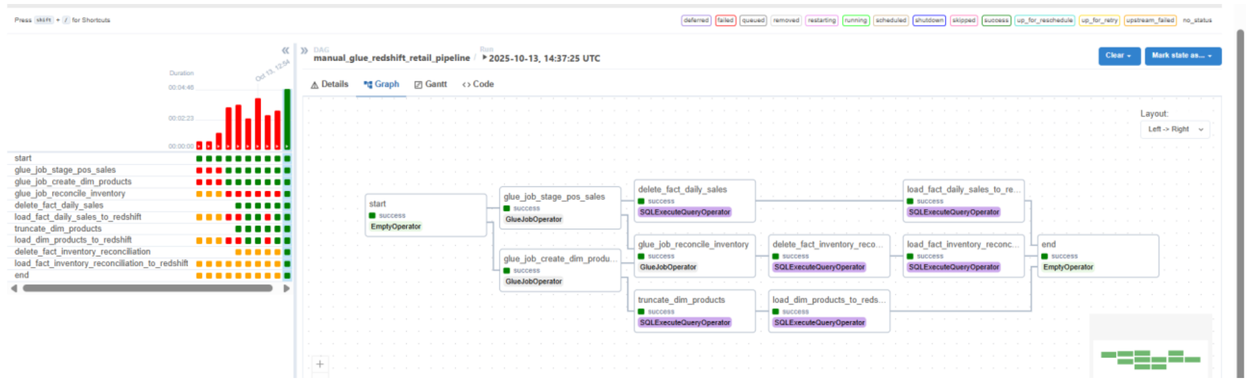
Processing Date *: 2025-10-12

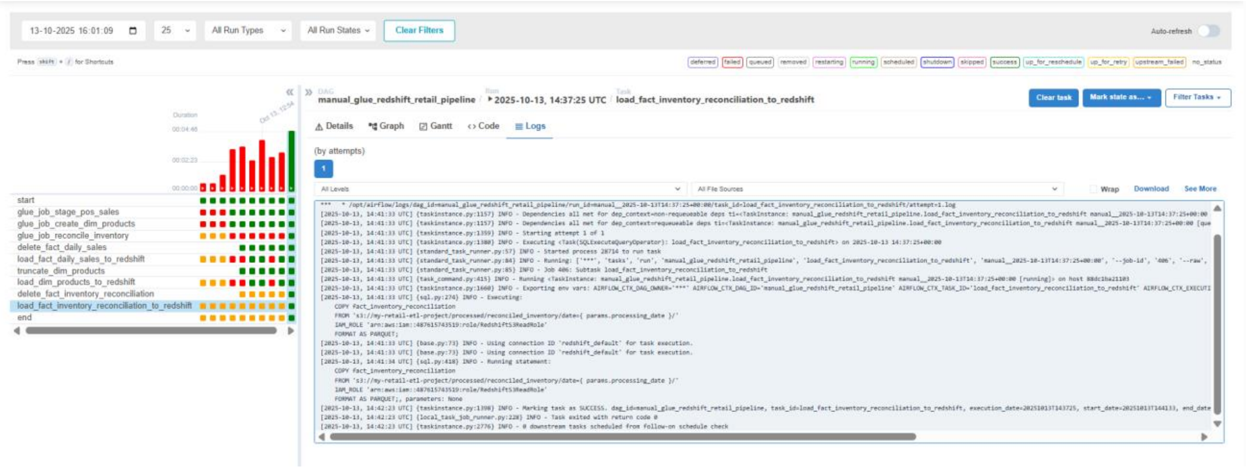
The date for which to run the pipeline (format: YYYY-MM-DD).

Generated Configuration JSON

To access configuration in your DAG use `{{ dag_run.conf }}`. As `core.dag_run_conf_overrides_params` is set to `True`, so passing any configuration here will override task params which can be accessed via `{{ params }}`.

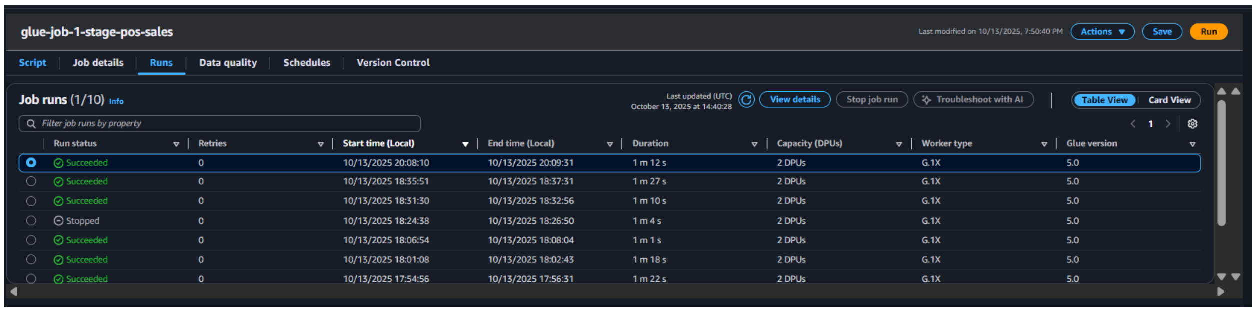
Unpause DAG when triggered





```
***   * /opt/airflow/logs/dag_id=manual_glue_redshift_retail_pipeline/run_id=manual__2025-10-13T14:37:25+00:00/task_id=load_fact_inventory_reconciliation_to_redshift/attempt=1.log
[2025-10-13, 14:41:33 UTC] {taskinstance.py:1157} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: manual_glue_redshift_retail_pipeline.load_fact_inventory_reconciliation_to_redshift manual__2025-10-13T14:37:25+00:00
[2025-10-13, 14:41:33 UTC] {taskinstance.py:1157} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: manual_glue_redshift_retail_pipeline.load_fact_inventory_reconciliation_to_redshift manual__2025-10-13T14:37:25+00:00 [que
[2025-10-13, 14:41:33 UTC] {taskinstance.py:1359} INFO - Starting attempt 1 of 1
[2025-10-13, 14:41:33 UTC] {taskinstance.py:1380} INFO - Executing <Task(SQLExecuteQueryOperator): load_fact_inventory_reconciliation_to_redshift> on 2025-10-13 14:37:25+00:00
[2025-10-13, 14:41:33 UTC] {standard_task_runner.py:57} INFO - Started process 28714 to run task
[2025-10-13, 14:41:33 UTC] {standard_task_runner.py:84} INFO - Running: ['***', 'tasks', 'run', 'manual_glue_redshift_retail_pipeline', 'load_fact_inventory_reconciliation_to_redshift', 'manual__2025-10-13T14:37:25+00:00', '--job-id', '406', '--raw',
[2025-10-13, 14:41:33 UTC] {standard_task_runner.py:85} INFO - Job 406: Subtask load_fact_inventory_reconciliation_to_redshift
[2025-10-13, 14:41:33 UTC] {task_command.py:415} INFO - Running <TaskInstance: manual_glue_redshift_retail_pipeline.load_fact_inventory_reconciliation_to_redshift manual__2025-10-13T14:37:25+00:00 [running]> on host 88dc1ba21103
[2025-10-13, 14:41:33 UTC] {taskinstance.py:1660} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='***' AIRFLOW_CTX_DAG_ID='manual_glue_redshift_retail_pipeline' AIRFLOW_CTX_TASK_ID='load_fact_inventory_reconciliation_to_redshift' AIRFLOW_CTX_EXECUTI
[2025-10-13, 14:41:33 UTC] {sql.py:274} INFO - Executing:
    COPY fact_inventory_reconciliation
    FROM 's3://my-retail-etl-project/processed/reconciled_inventory/date={ params.processing_date }/'
    IAM_ROLE 'arn:aws:iam::487615743519:role/RedshiftS3ReadRole'
    FORMAT AS PARQUET;
[2025-10-13, 14:41:33 UTC] {base.py:73} INFO - Using connection ID 'redshift_default' for task execution.
[2025-10-13, 14:41:33 UTC] {base.py:73} INFO - Using connection ID 'redshift_default' for task execution.
[2025-10-13, 14:41:34 UTC] {sql.py:418} INFO - Running statement:
    COPY fact_inventory_reconciliation
    FROM 's3://my-retail-etl-project/processed/reconciled_inventory/date={ params.processing_date }/'
    IAM_ROLE 'arn:aws:iam::487615743519:role/RedshiftS3ReadRole'
    FORMAT AS PARQUET;, parameters: None
[2025-10-13, 14:42:23 UTC] {taskinstance.py:1398} INFO - Marking task as SUCCESS. dag_id=manual_glue_redshift_retail_pipeline, task_id=load_fact_inventory_reconciliation_to_redshift, execution_date=20251013T143725, start_date=20251013T144133, end_date
[2025-10-13, 14:42:23 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2025-10-13, 14:42:23 UTC] {taskinstance.py:2776} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

Verify in Glue, S3 and Redshift:



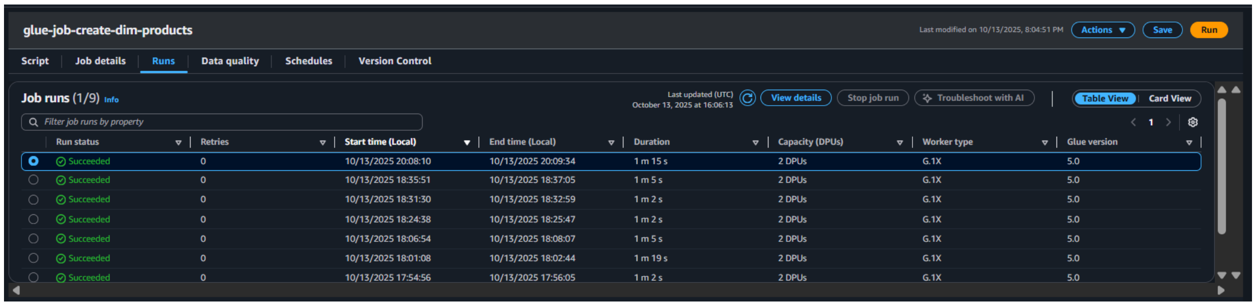
glue-job-1-stage-pos-sales

Last modified on 10/13/2025, 7:50:40 PM

Job runs (1/10)

Last updated (UTC) October 13, 2025 at 14:40:28

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Succeeded	0	10/13/2025 20:08:10	10/13/2025 20:09:31	1 m 12 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 18:35:51	10/13/2025 18:37:31	1 m 27 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 18:31:30	10/13/2025 18:32:56	1 m 10 s	2 DPUs	G.1X	5.0
Stopped	0	10/13/2025 18:24:38	10/13/2025 18:26:50	1 m 4 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 18:06:54	10/13/2025 18:08:04	1 m 1 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 18:01:08	10/13/2025 18:02:43	1 m 18 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 17:54:56	10/13/2025 17:56:31	1 m 22 s	2 DPUs	G.1X	5.0



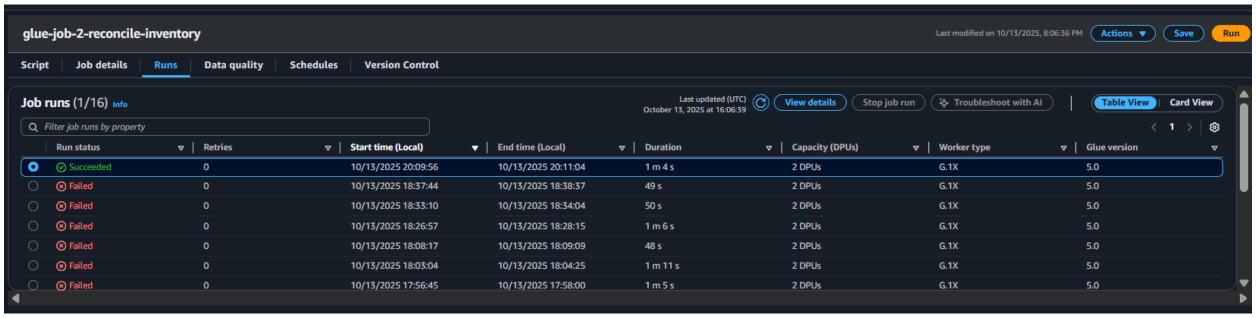
glue-job-create-dim-products

Last modified on 10/13/2025, 8:04:51 PM

Job runs (1/9)

Last updated (UTC) October 13, 2025 at 16:06:13

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Succeeded	0	10/13/2025 20:08:10	10/13/2025 20:09:34	1 m 15 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 18:35:51	10/13/2025 18:37:05	1 m 5 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 18:31:30	10/13/2025 18:32:59	1 m 2 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 18:24:38	10/13/2025 18:25:47	1 m 2 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 18:06:54	10/13/2025 18:08:07	1 m 5 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 18:01:08	10/13/2025 18:02:44	1 m 19 s	2 DPUs	G.1X	5.0
Succeeded	0	10/13/2025 17:54:56	10/13/2025 17:56:05	1 m 2 s	2 DPUs	G.1X	5.0



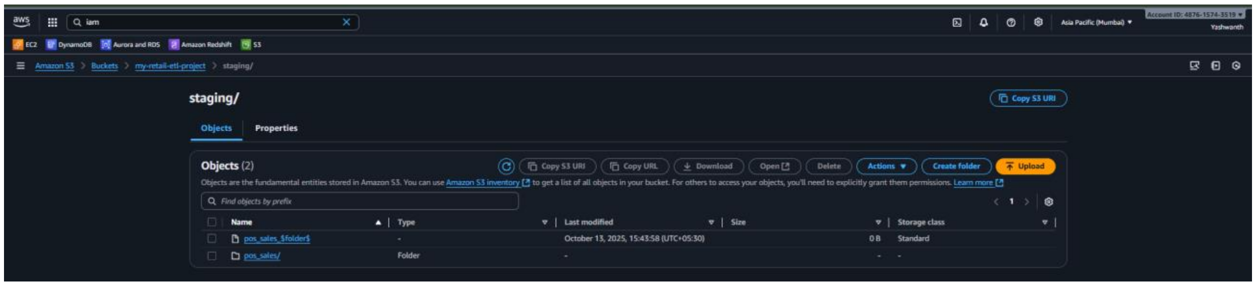
glue-job-2-reconcile-inventory

Last modified on 10/13/2025, 8:06:36 PM

Job runs (1/16)

Last updated (UTC) October 13, 2025 at 16:06:39

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Succeeded	0	10/13/2025 20:09:56	10/13/2025 20:11:04	1 m 4 s	2 DPUs	G.1X	5.0
Failed	0	10/13/2025 18:37:44	10/13/2025 18:38:37	49 s	2 DPUs	G.1X	5.0
Failed	0	10/13/2025 18:33:10	10/13/2025 18:34:04	50 s	2 DPUs	G.1X	5.0
Failed	0	10/13/2025 18:26:57	10/13/2025 18:28:15	1 m 6 s	2 DPUs	G.1X	5.0
Failed	0	10/13/2025 18:08:17	10/13/2025 18:09:09	48 s	2 DPUs	G.1X	5.0
Failed	0	10/13/2025 18:03:04	10/13/2025 18:04:25	1 m 11 s	2 DPUs	G.1X	5.0
Failed	0	10/13/2025 17:56:45	10/13/2025 17:58:00	1 m 5 s	2 DPUs	G.1X	5.0



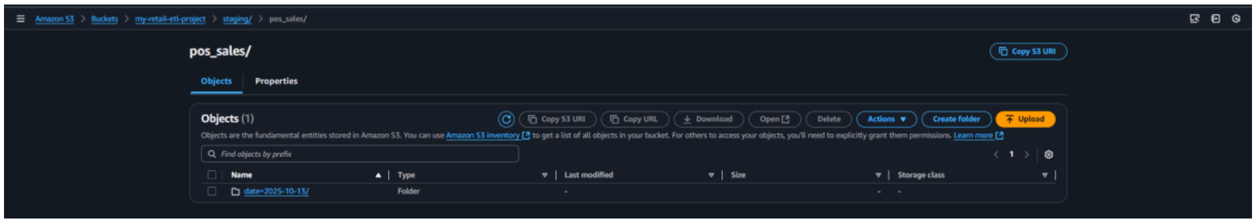
Amazon S3 > Buckets > my-retail-etl-project > staging/

staging/

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
pos_sales_$folder$	-	October 13, 2025, 15:43:58 (UTC+05:30)	0 B	Standard
pos_sales/	Folder	-	-	-



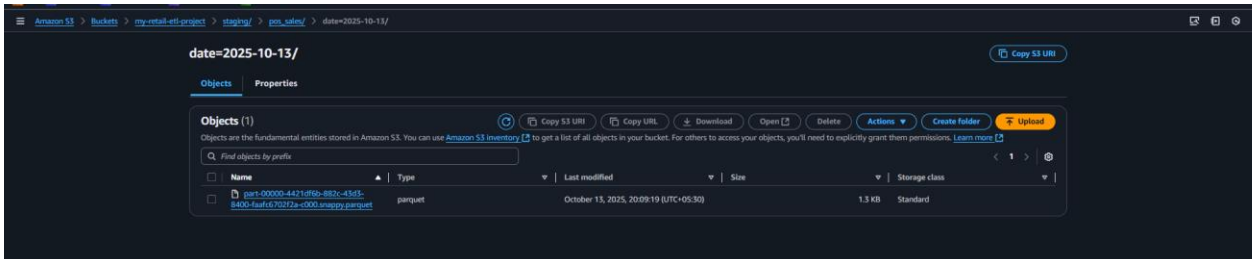
Amazon S3 > Buckets > my-retail-etl-project > staging/ > pos_sales/

pos_sales/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
date=2025-10-13/	Folder	-	-	-



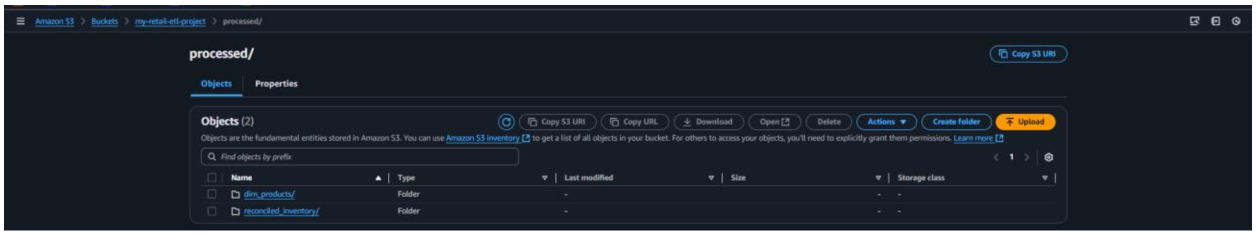
Amazon S3 > Buckets > my-retail-etl-project > staging/ > pos_sales/ > date=2025-10-13/

date=2025-10-13/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
part-00000-4421df6b-882c-43d3-8400-faafc6702f2a-c000.snappy.parquet	parquet	October 13, 2025, 20:09:19 (UTC+05:30)	1.3 KB	Standard



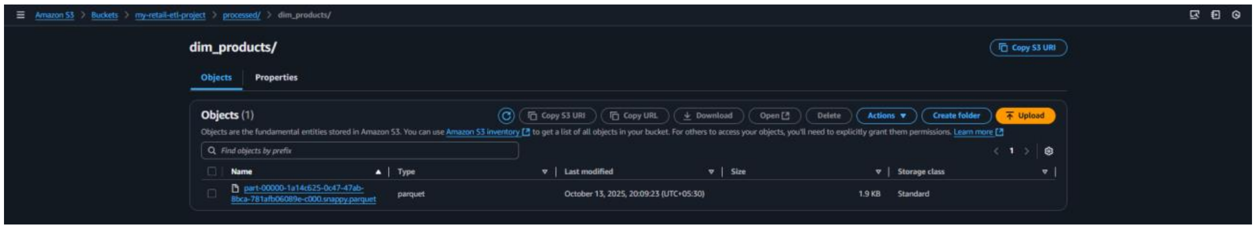
Amazon S3 > Buckets > my-retail-etl-project > processed/

processed/

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
dim_products/	Folder	-	-	-
reconciled_inventory/	Folder	-	-	-



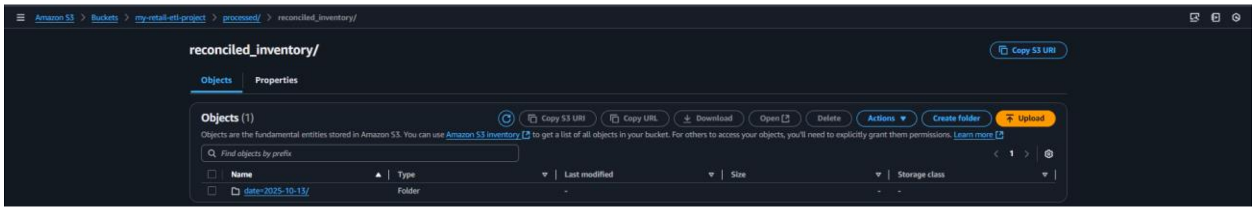
Amazon S3 > Buckets > my-retail-etl-project > processed/ > dim_products/

dim_products/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
part-00000-1a14c625-0c47-47ab-8bca-781afb06089e-c000.snappy.parquet	parquet	October 13, 2025, 20:09:23 (UTC+05:30)	1.9 KB	Standard



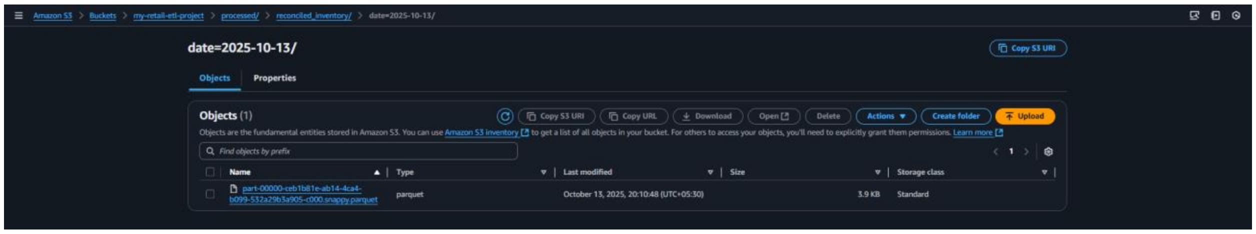
Amazon S3 > Buckets > my-retail-etl-project > processed/ > reconciled_inventory/

reconciled_inventory/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
date=2025-10-13/	Folder	-	-	-



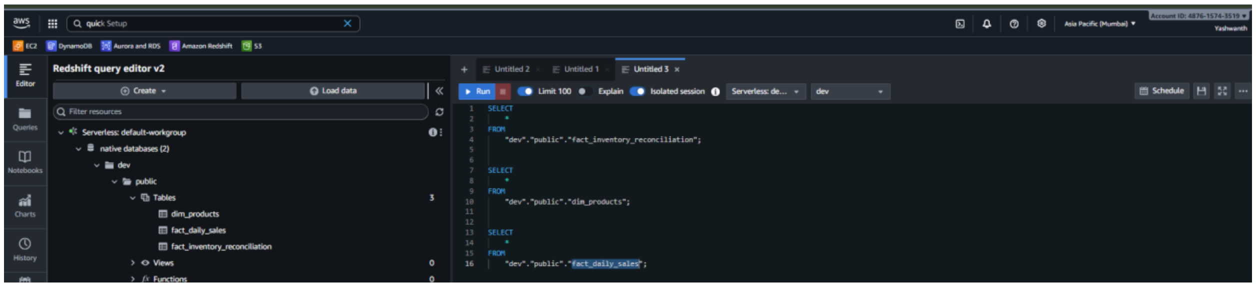
Amazon S3 > Buckets > my-retail-etl-project > processed/ > reconciled_inventory/ > date=2025-10-13/

date=2025-10-13/

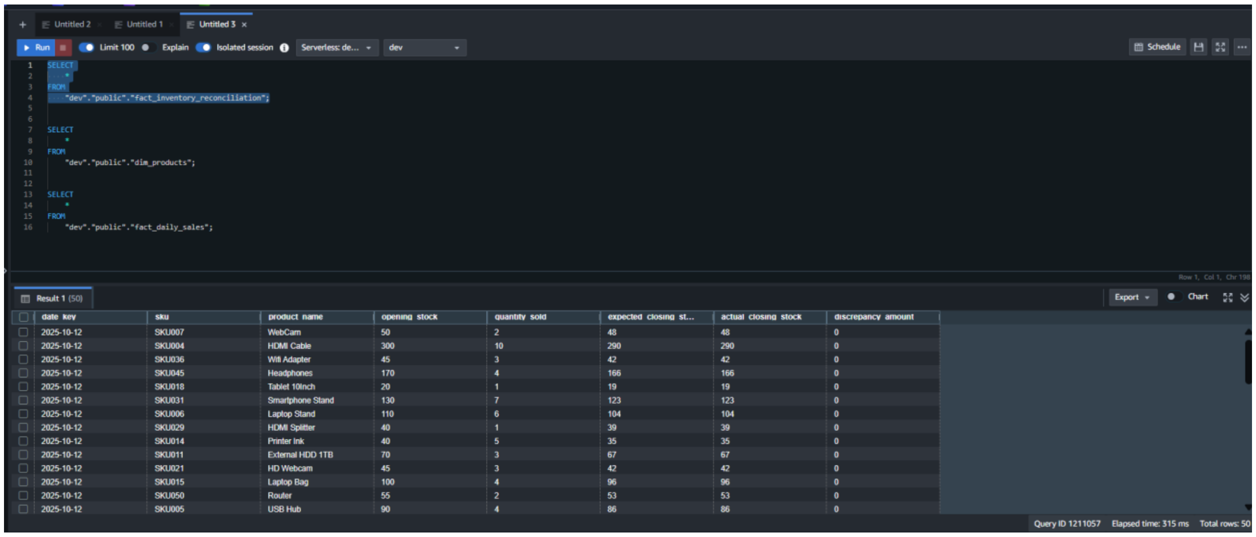
Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
part-00000-ceb1b81e-ab14-4ca4-b099-532a29b3a905-c000.snappy.parquet	parquet	October 13, 2025, 20:10:48 (UTC+05:30)	3.9 KB	Standard



```
SELECT
    *
FROM
    "dev"."public"."fact_inventory_reconciliation";


SELECT
    *
FROM
    "dev"."public"."dim_products";


SELECT
    *
FROM
    "dev"."public"."fact_daily_sales";
```

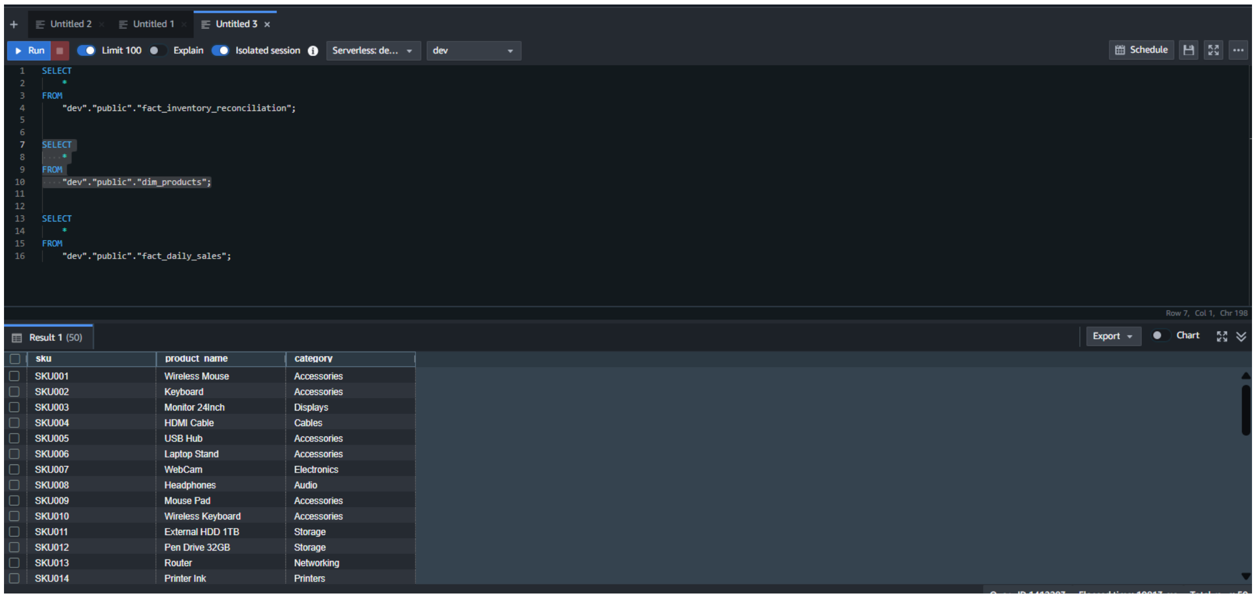


```
SELECT
    *
FROM
    "dev"."public"."fact_inventory_reconciliation";


SELECT
    *
FROM
    "dev"."public"."dim_products";


SELECT
    *
FROM
    "dev"."public"."fact_daily_sales";
```

Result 1 (50)

date key	sku	product name	opening stock	quantity sold	expected closing st...	actual closing stock	discrepancy amount
2025-10-12	SKU007	WebCam	50	2	48	48	0
2025-10-12	SKU004	HDMI Cable	300	10	290	290	0
2025-10-12	SKU036	Wifi Adapter	45	3	42	42	0
2025-10-12	SKU045	Headphones	170	4	166	166	0
2025-10-12	SKU018	Tablet 10inch	20	1	19	19	0
2025-10-12	SKU031	Smartphone Stand	130	7	123	123	0
2025-10-12	SKU006	Laptop Stand	110	6	104	104	0
2025-10-12	SKU029	HDMI Splitter	40	1	39	39	0
2025-10-12	SKU014	Printer Ink	40	5	35	35	0
2025-10-12	SKU011	External HDD 1TB	70	3	67	67	0
2025-10-12	SKU021	HD Webcam	45	3	42	42	0
2025-10-12	SKU015	Laptop Bag	100	4	96	96	0
2025-10-12	SKU050	Router	55	2	53	53	0
2025-10-12	SKU005	USB Hub	90	4	86	86	0

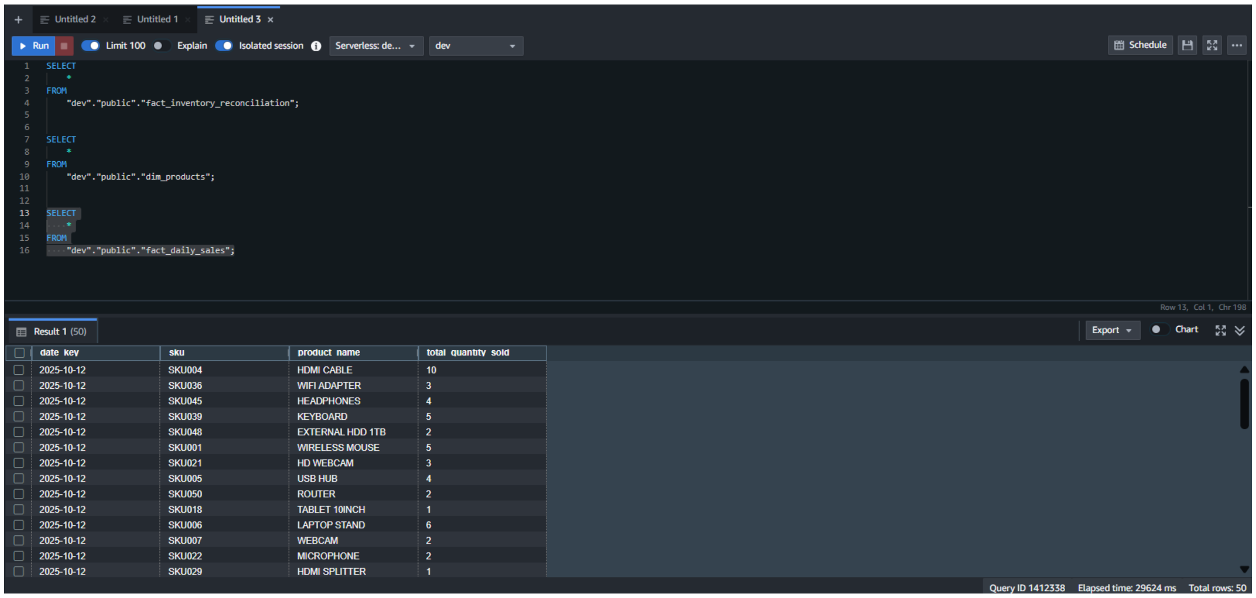
Query ID 1211057   Elapsed time: 315 ms   Total rows: 50



```
SELECT
    *
FROM
    "dev"."public"."fact_inventory_reconciliation";


SELECT
    *
FROM
    "dev"."public"."dim_products";


SELECT
    *
FROM
    "dev"."public"."fact_daily_sales";
```

Result 1 (50)

sku	product name	category
SKU001	Wireless Mouse	Accessories
SKU002	Keyboard	Accessories
SKU003	Monitor 24Inch	Displays
SKU004	HDMI Cable	Cables
SKU005	USB Hub	Accessories
SKU006	Laptop Stand	Accessories
SKU007	WebCam	Electronics
SKU008	Headphones	Audio
SKU009	Mouse Pad	Accessories
SKU010	Wireless Keyboard	Accessories
SKU011	External HDD 1TB	Storage
SKU012	Pen Drive 32GB	Storage
SKU013	Router	Networking
SKU014	Printer Ink	Printers



```sql
SELECT
    *
FROM
    "dev"."public"."fact_inventory_reconciliation";


SELECT
    *
FROM
    "dev"."public"."dim_products";


SELECT
    *
FROM
    "dev"."public"."fact_daily_sales";
```

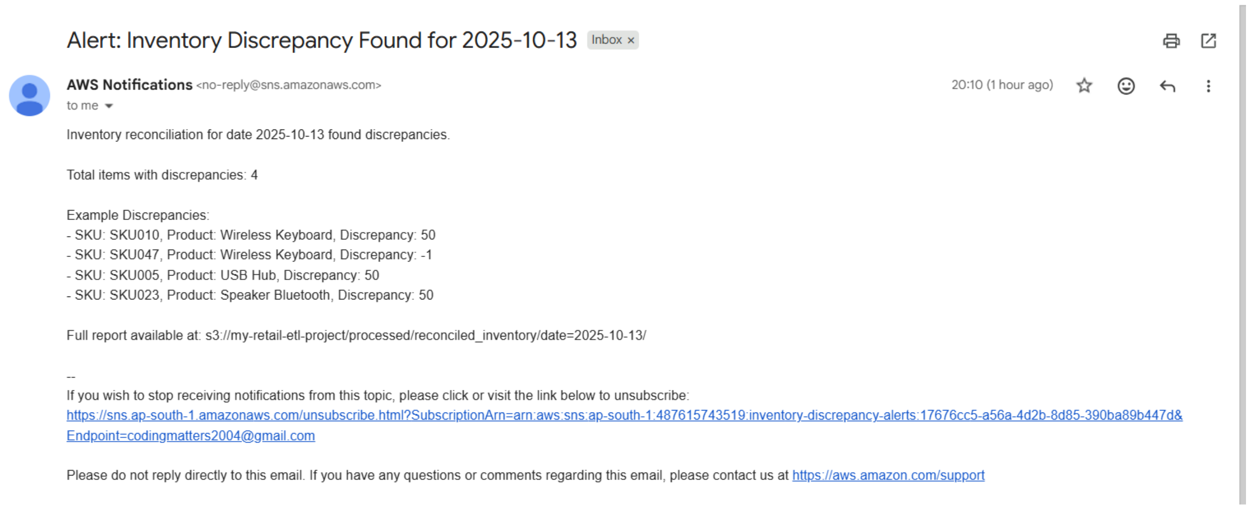
Row 13, Col 1, Chr 198

Result 1 (50)

date key	sku	product name	total quantity sold
2025-10-12	SKU004	HDMI CABLE	10
2025-10-12	SKU036	WIFI ADAPTER	3
2025-10-12	SKU045	HEADPHONES	4
2025-10-12	SKU039	KEYBOARD	5
2025-10-12	SKU048	EXTERNAL HDD 1TB	2
2025-10-12	SKU001	WIRELESS MOUSE	5
2025-10-12	SKU021	HD WEBCAM	3
2025-10-12	SKU005	USB HUB	4
2025-10-12	SKU050	ROUTER	2
2025-10-12	SKU018	TABLET 10INCH	1
2025-10-12	SKU006	LAPTOP STAND	6
2025-10-12	SKU007	WEBCAM	2
2025-10-12	SKU022	MICROPHONE	2
2025-10-12	SKU029	HDMI SPLITTER	1

Query ID 1412338 Elapsed time: 29624 ms Total rows: 50

Check Email:



Alert: Inventory Discrepancy Found for 2025-10-13

AWS Notifications <no-reply@sns.amazonaws.com>
to me

20:10 (1 hour ago)

Inventory reconciliation for date 2025-10-13 found discrepancies.

Total items with discrepancies: 4

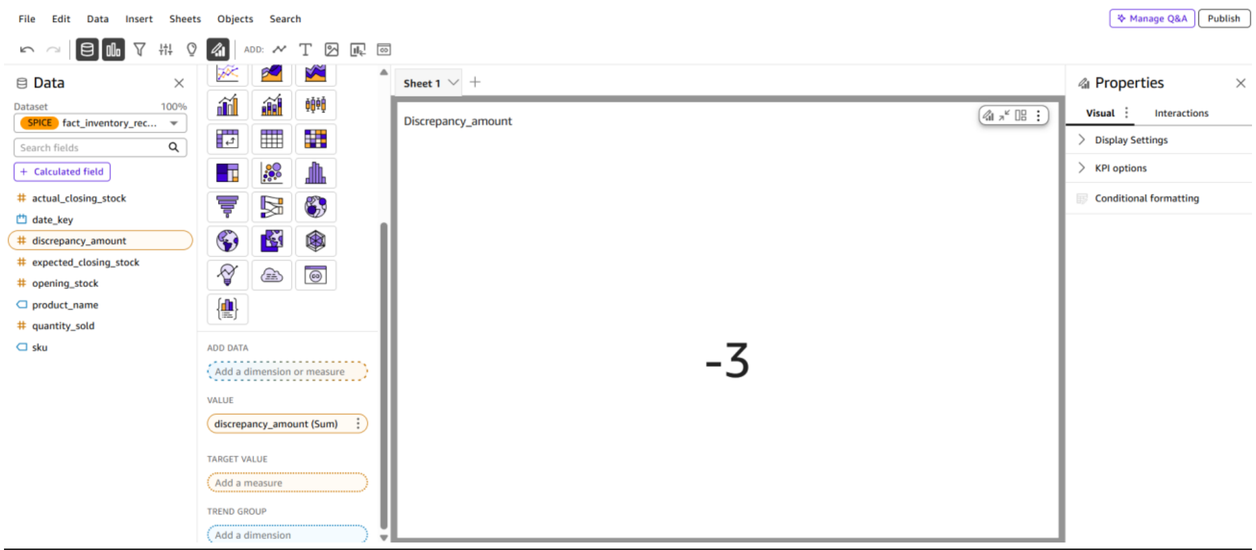
Example Discrepancies:
- SKU: SKU010, Product: Wireless Keyboard, Discrepancy: 50
- SKU: SKU047, Product: Wireless Keyboard, Discrepancy: -1
- SKU: SKU005, Product: USB Hub, Discrepancy: 50
- SKU: SKU023, Product: Speaker Bluetooth, Discrepancy: 50

Full report available at: s3://my-retail-etl-project/processed/reconciled_inventory/date=2025-10-13/

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.ap-south-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:ap-south-1:487615743519:inventory-discrepancy-alerts:17676cc5-a56a-4d2b-8d85-390ba89b447d&Endpoint=codingmatters2004@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

QuickSight:



Data

Dataset 100%

SPICE fact_inventory_rec...

Search fields

+ Calculated field

- actual_closing_stock
- date_key
- discrepancy_amount
- expected_closing_stock
- opening_stock
- product_name
- quantity_sold
- sku

ADD DATA: Add a dimension or measure

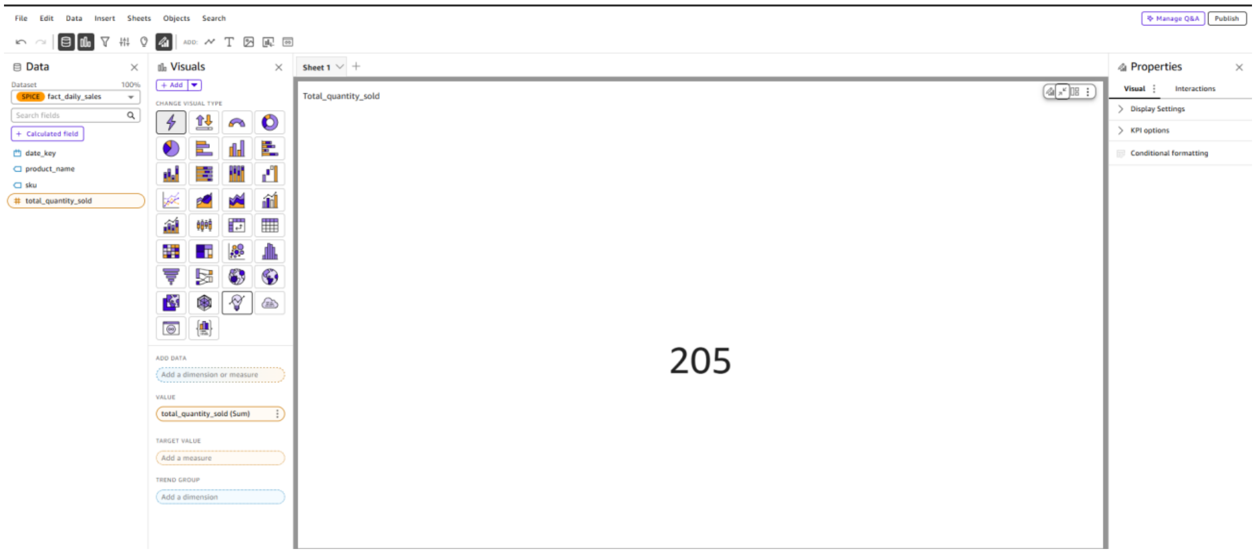
VALUE: discrepancy_amount (Sum)

TARGET VALUE: Add a measure

TREND GROUP: Add a dimension

Discrepancy_amount

-3



Data

Dataset 100%

SPICE fact_daily_sales

Search fields

+ Calculated field

- date_key
- product_name
- sku
- total_quantity_sold

Visuals

ADD DATA: Add a dimension or measure

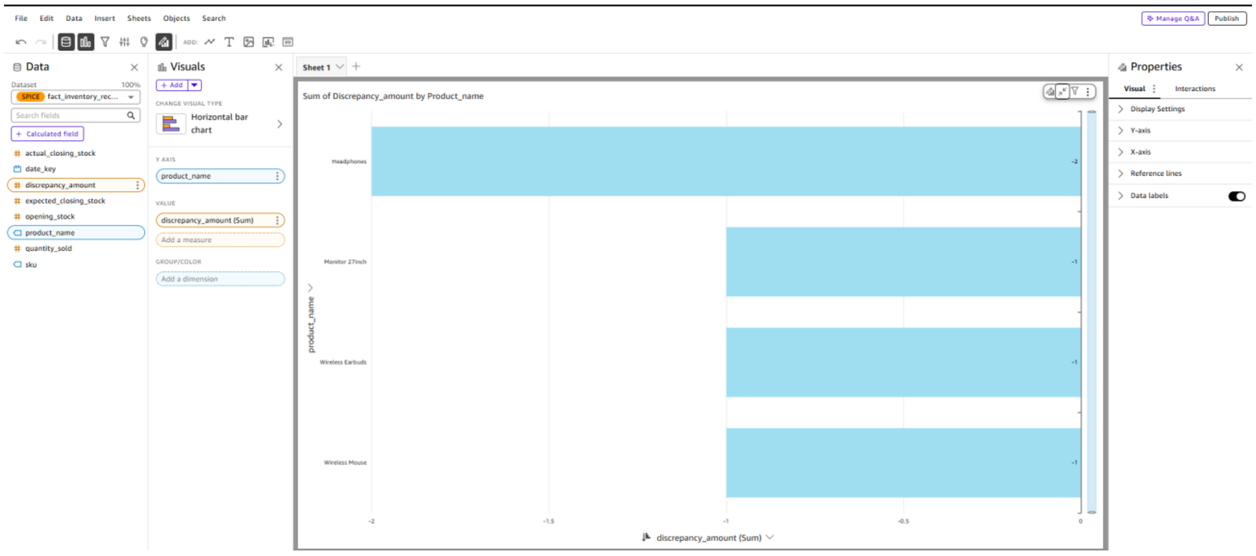
VALUE: total_quantity_sold (Sum)

TARGET VALUE: Add a measure

TREND GROUP: Add a dimension

Total_quantity_sold

205



Data

Dataset 100%

SPICE fact_inventory_rec...

Search fields

+ Calculated field

- actual_closing_stock
- date_key
- discrepancy_amount
- expected_closing_stock
- opening_stock
- product_name
- quantity_sold
- sku

Visuals

CHANGE VISUAL TYPE: Horizontal bar chart

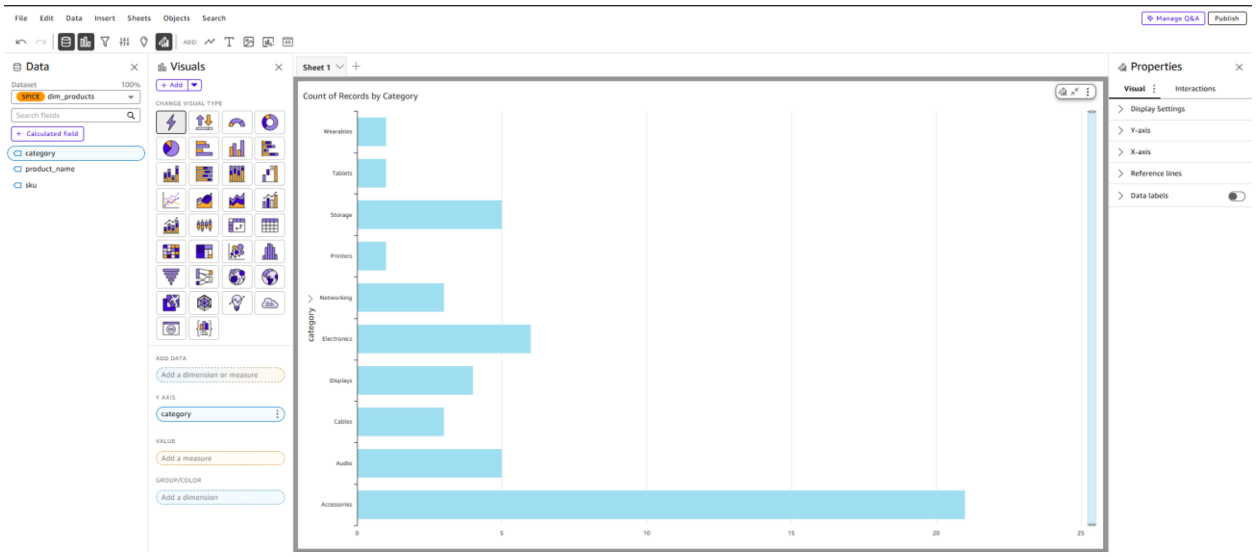
Y AXIS: product_name

VALUE: discrepancy_amount (Sum)

Add a measure

GROUP/COLOR: Add a dimension

Sum of Discrepancy_amount by Product_name



Dataset 100%
SPICE dim_products

category
product_name
sku

ADD DATA
Y AXIS: category
VALUE
GROUP/COLOR

Count of Records by Category