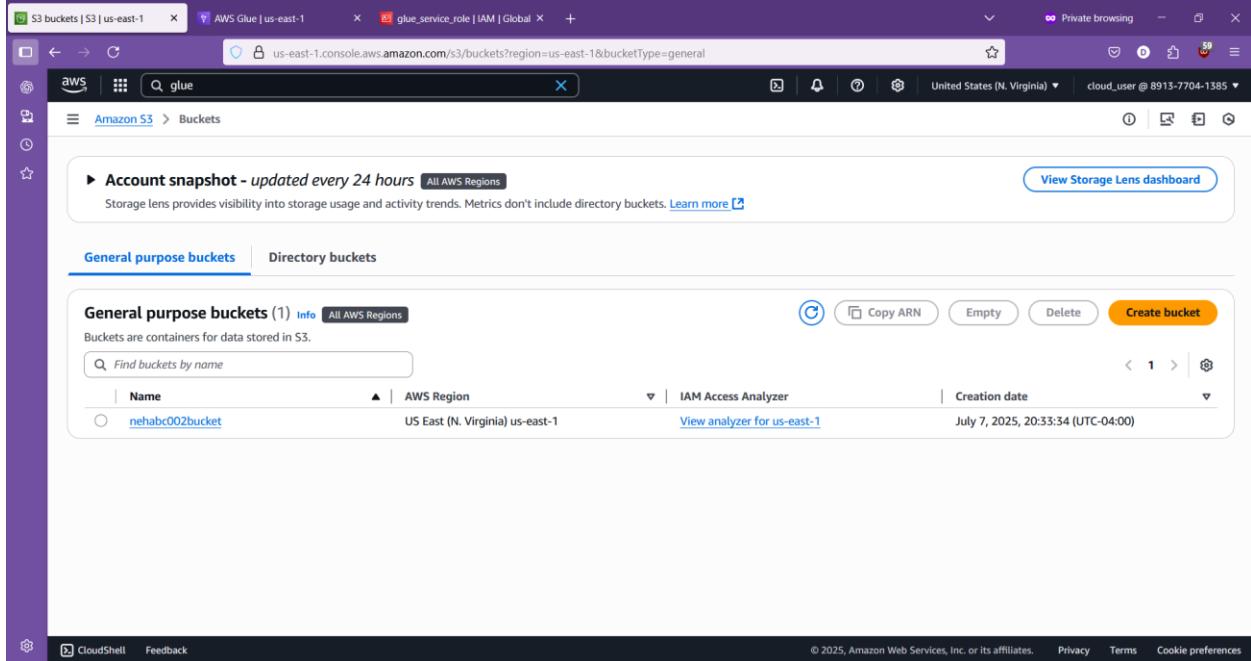


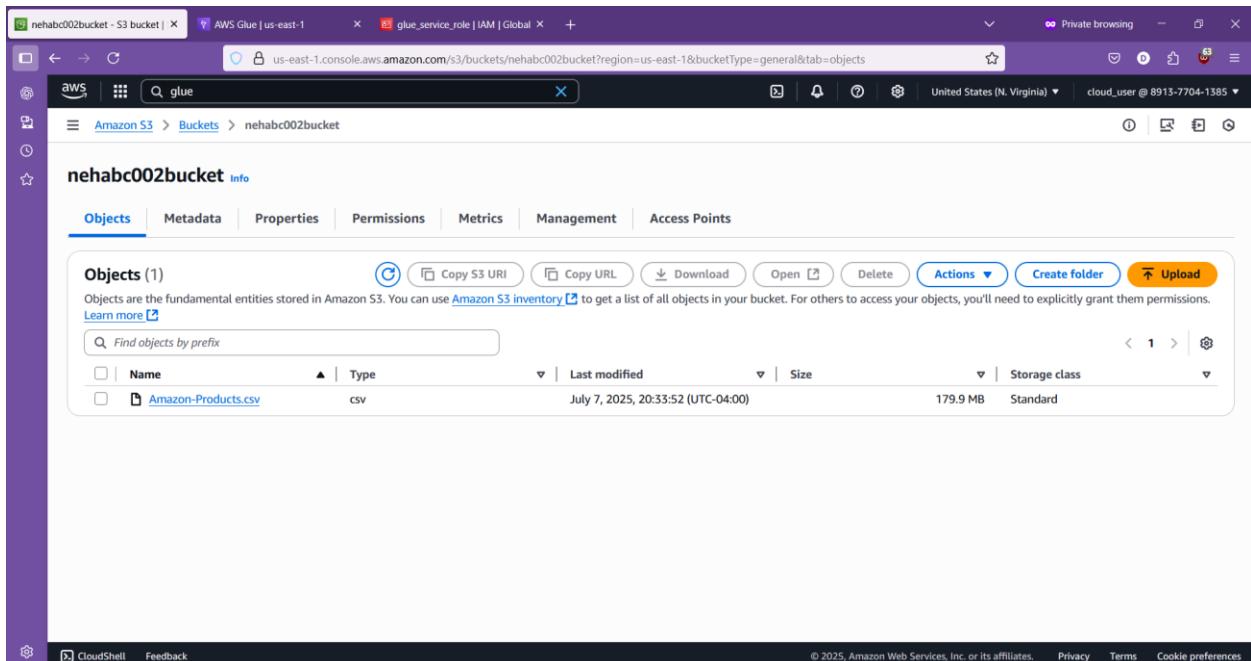
# Advanced Product Analytics Platform Using Databricks & AWS

## 1. Bronze Layer (Raw Ingestion) - S3 Database



The screenshot shows the AWS S3 console with the "General purpose buckets" tab selected. It displays a single bucket named "nehabc002bucket" created on July 7, 2025, at 20:33:34 UTC. The bucket is located in the US East (N. Virginia) region.

Name	AWS Region	IAM Access Analyzer	Creation date
nehabc002bucket	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>	July 7, 2025, 20:33:34 (UTC-04:00)

The screenshot shows the AWS S3 console with the "Objects" tab selected for the "nehabc002bucket". It lists a single object named "Amazon-Products.csv" which is a CSV file. The file was last modified on July 7, 2025, at 20:33:52 UTC and has a size of 179.9 MB.

Name	Type	Last modified	Size	Storage class
Amazon-Products.csv	csv	July 7, 2025, 20:33:52 (UTC-04:00)	179.9 MB	Standard

## Step 1. Create Glue catalog for schema detection.

### 1. Create a Glue Database

The screenshot shows the AWS Glue Databases page. On the left, there's a sidebar with links like Notebooks, Job run monitoring, Data Catalog tables, Data connections, Workflows (orchestration), Zero-ETL integrations, Data Catalog, Databases, Tables, Stream schema registries, Schemas, Connections, Crawlers, Classifiers, Catalog settings, Data Integration and ETL, and Legacy pages. The main area is titled 'Databases (1)' and contains a table with one entry: 'amazon\_bronze\_db'. The table has columns for Name, Description, Location URI, and Created on (UTC). The 'Created on (UTC)' column shows 'July 8, 2025 at 00:44:12'. At the top right, there are 'Edit', 'Delete', and 'Add database' buttons. The status bar at the bottom right indicates 'Last updated (UTC) July 8, 2025 at 00:44:10'.

### 2. Create a Glue Crawler

The screenshot shows the 'Add crawler' page in the AWS Glue console. The sidebar on the left is identical to the previous screenshot. The main area is titled 'Choose data sources and classifiers'. It shows a step-by-step process: Step 1 (Set crawler properties) is completed (indicated by a solid dot). Step 2 (Choose data sources and classifiers) is currently selected (indicated by a blue outline). Step 3 (Configure security settings), Step 4 (Set output and scheduling), and Step 5 (Review and create) are shown as unselected options. Under 'Data source configuration', there are two radio button options: 'Not yet' (selected) and 'Yes'. The 'Not yet' option has a note: 'Select one or more data sources to be crawled.' Under 'Data sources (1)', there is a table with one entry: Type 'S3', Data source 's3://nehabc002bucket/bronze/Am...', and Parameters 'Recrawl all'. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons. The status bar at the bottom right indicates 'Last updated (UTC) July 8, 2025 at 00:44:10'.

Screenshot of the AWS Glue 'Add crawler' wizard Step 3: Configure security settings.

The sidebar shows the navigation path: AWS Glue > Crawlers > Add crawler.

The main panel title is "Configure security settings".

**IAM role** info: Existing IAM role is set to "glue\_service\_role". Options include "Create new IAM role" and "View".

**Lake Formation configuration - optional**: A checkbox "Use Lake Formation credentials for crawling S3 data source" is checked. A note states: "Checking this box will allow the crawler to use Lake Formation credentials for crawling the data source. If the data source is registered in another account, you must provide the registered account ID. Otherwise, the crawler will crawl only those data sources associated to the account. Only applicable to S3, Glue Catalog, Iceberg, and Hudi data sources."

**Security configuration - optional**: A note states: "Enable at-rest encryption with a security configuration." A "Next" button is visible at the bottom right.

Screenshot of the AWS Glue 'Add crawler' wizard Step 4: Set output and scheduling.

The sidebar shows the navigation path: AWS Glue > Crawlers > Add crawler.

The main panel title is "Set output and scheduling".

**Output configuration** info: Target database is set to "amazon\_bronze\_db". Options include "Clear selection" and "Add database".

**Table name prefix - optional**: A text input field with placeholder "Type a prefix added to table names".

**Maximum table threshold - optional**: A text input field with placeholder "Type a number greater than 0".

**Advanced options**: A section with a "Next" button at the bottom right.

**Crawler schedule**: A note: "You can define a time-based schedule for your crawlers and jobs in AWS Glue. The definition of these schedules uses the Unix-like cron syntax. Learn more." Frequency is set to "On demand".

Screenshot of the AWS Glue console showing the 'Add crawler' wizard. The sidebar shows 'Data Catalog' selected. The main steps are:

- Step 2: Choose data sources and classifiers
- Step 3: Configure security settings
- Step 4: Set output and scheduling
- Step 5: Review and create (highlighted)

**Step 1: Set crawler properties**

**Set crawler properties**

Name	Description	Tags
amazon_products_crawler	-	-

**Step 2: Choose data sources and classifiers**

**Data sources (1) Info**  
The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://nehabc002bucket/bronze/Amazon...	Recrawl all

**Step 3: Configure security settings**

**Configure security settings**

IAM role	Security configuration	Lake Formation configuration
glue_service_role	-	-

**Step 4: Set output and scheduling**

**Set output and scheduling**

Database	Table prefix - optional	Maximum table threshold - optional	Schedule
amazon_bronze_db	-	-	On demand

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS Glue console showing the 'Add crawler' wizard. The sidebar shows 'Data Catalog' selected. The main steps are:

- Step 4: Set output and scheduling
- Step 5: Review and create (highlighted)

**Step 1: Set crawler properties**

**Set crawler properties**

Name	Description	Tags
amazon_products_crawler	-	-

**Step 2: Choose data sources and classifiers**

**Data sources (1) Info**  
The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://nehabc002bucket/bronze/Amazon...	Recrawl all

**Step 3: Configure security settings**

**Configure security settings**

IAM role	Security configuration	Lake Formation configuration
glue_service_role	-	-

**Step 4: Set output and scheduling**

**Set output and scheduling**

Database	Table prefix - optional	Maximum table threshold - optional	Schedule
amazon_bronze_db	-	-	On demand

**Create crawler**

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS Glue Console showing the crawler configuration for "amazon\_products\_crawler".

The crawler properties are as follows:

- Name: amazon\_products\_crawler
- IAM role: glue\_service\_role
- Description: -
- Database: amazon\_bronze\_db
- State: READY

The Crawler runs section shows 0 runs. The last update was on July 8, 2025 at 00:46:25.

Actions available: Run crawler, Edit, Delete.

Screenshot of the AWS Glue Console showing the crawler configuration for "amazon\_products\_crawler".

The crawler properties are as follows:

- Name: amazon\_products\_crawler
- IAM role: glue\_service\_role
- Description: -
- Database: amazon\_bronze\_db
- State: READY

The Crawler runs section shows 1 run. The last update was on July 8, 2025 at 00:46:25. The run status is Running.

Actions available: Run crawler, Edit, Delete.

Crawler properties

Name	IAM role	Database	State
amazon_products_crawler	glue_service_role	amazon_bronze_db	READY

Description: -

Maximum table threshold: -

Advanced settings: ▶

Crawler runs | Schedule | Data sources | Classifiers | Tags

Crawler runs (1)

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
July 8, 2025 at 00:46:31	July 8, 2025 at 00:47:49	01 min 17 s	Completed	-	-

### 3. Table:

Table overview | Data quality - new

Table details

Name	Classification	Deprecated
amazon_products_csv	CSV	-

Location: s3://nehabc002bucket/bronze/Amazon-Products.csv

Description: -

Last updated: July 8, 2025 at 01:12:50

Advanced properties: ▶

Schema | Partitions | Indexes | Column statistics - new

Schema (10)

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8	Column 9	Column 10
Product ID	Product Name	Category	Price	Stock Level	Supplier ID	Supplier Name	Manufacturing Date	Expiry Date	Rating

The screenshot shows the AWS Glue Data Catalog interface. On the left, there's a sidebar with various navigation options like 'Data Catalog', 'Databases', 'Data connections', and 'Workflows'. The main area displays the 'amazon\_products\_csv' table under the 'Tables' section. A detailed schema view is open, showing 10 columns with their respective data types: row\_id (bigint), name (string), main\_category (string), sub\_category (string), image (string), link (string), ratings (string), no\_of\_ratings (string), discount\_price (string), and actual\_price (string). Each column has a 'Partition key' status indicated by a minus sign.

## Silver Layer (Data Cleaning & Transformation)

### Objectives:

1. Clean and convert price fields (remove ₹, commas, cast to float)
2. Convert ratings and number of ratings to numeric
3. Remove duplicates and handle missing values
4. Trim and standardize category fields

Step 1. Connect the dataset in S3 bucket with Databricks

The screenshot shows the Databricks workspace with a Python notebook named 'silver\_layer\_job'. The notebook contains the following code:

```

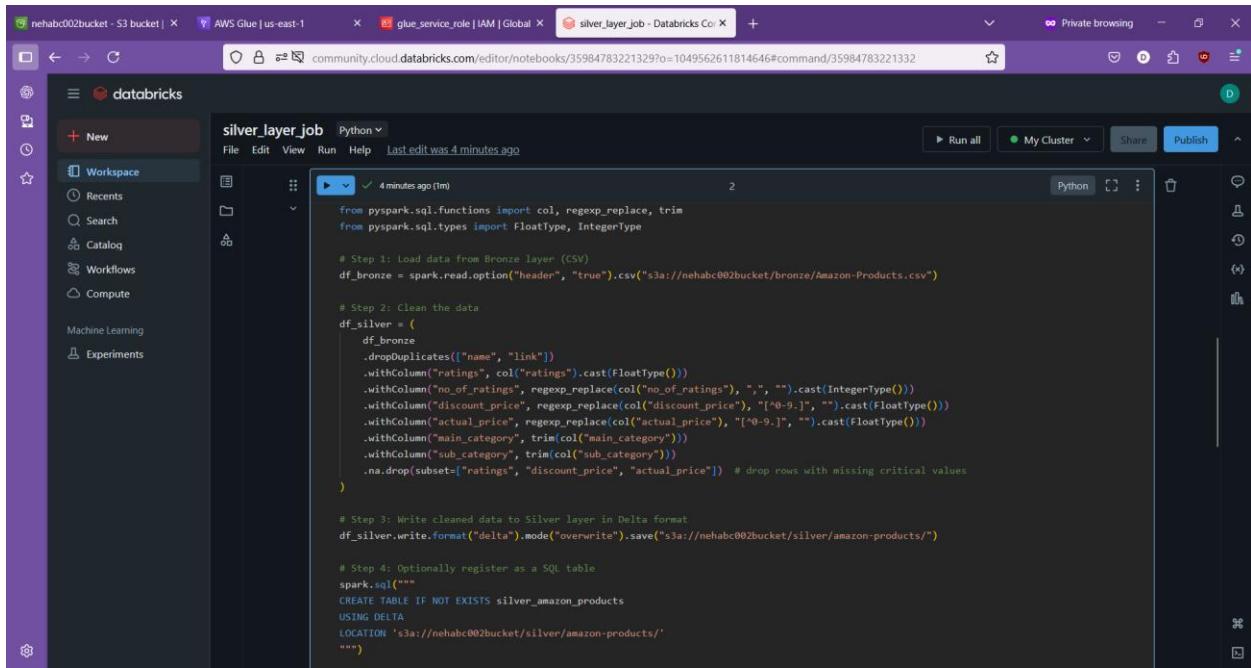
# Now read the CSV
df = spark.read.option("header", "true").csv("s3a://nehabc002bucket/bronze/Amazon-Products.csv")
df.display()

```

The output of the code shows a DataFrame with the following data:

row_id	name	main_category
1	Lloyd 1.5 Ton 3 Star Inverter Split AC (5 In 1 Convertible, Copper, Anti-Viral + Pm 2.5 Filter, 2023 Model, White, Gis183...	appliances
2	LG 1.5 Ton 5 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling, HD Filter with Anti-Virus Protection...	appliances
3	LG 1 Ton 4 Star AI Dual Inverter Split Ac (Copper, Super Convertible 6-In-1 Cooling, Hd Filter With Anti Virus Protection...	appliances
4	LG 1.5 Ton 3 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling, HD Filter with Anti-Virus Protection...	appliances
5	Carrier 1.5 Ton 3 Star Inverter Split AC (Copper,ESTER Oil, 4-in-1 Flexcool Inverter, 2022 Model,R32,White)	appliances
6	Voltas 1.4 Ton 3 Star Inverter Split AC(Copper, Adjustable Cooling, Anti-dust Filter, 2023 Model, 173V Vectra Platina, White)	appliances
7	Lloyd 1.0 Ton 3 Star Inverter Split AC (5 In 1 Convertible, Copper, Anti-Viral + Pm 2.5 Filter, 2023 Model, White With Chr...	appliances
8	Lloyd 1.5 Ton 5 Star Inverter Split AC (5 In 1 Convertible, Copper, Anti-Viral + Pm 2.5 Filter, 2023 Model, White With Chr...	appliances
9	Carrier 1 Ton 3 Star AI Flexcool Inverter Split AC (Copper, Convertible 4-in-1 Cooling,Dual Filtration with HD & PM 2.5 F...	appliances

## Step 2. Silver Layer Script in Databricks (PySpark)



The screenshot shows the Databricks notebook interface with the title "silver\_layer.job". The code editor contains a Python script for processing Amazon product data. The script includes steps for loading data from the Bronze layer, cleaning it, and writing it to the Silver layer in Delta format. It also includes optional SQL registration.

```
from pyspark.sql.functions import col, regexp_replace, trim
from pyspark.sql.types import FloatType, IntegerType

# Step 1: Load data from Bronze layer (CSV)
df_bronze = spark.read.option("header", "true").csv("s3a://nehabc002bucket/bronze/Amazon-Products.csv")

# Step 2: Clean the data
df_silver = (
    df_bronze
    .dropDuplicates(["name", "link"])
    .withColumn("ratings", col("ratings").cast(FloatType()))
    .withColumn("no_of_ratings", regexp_replace(col("no_of_ratings"), ",", "").cast(IntegerType()))
    .withColumn("discount_price", regexp_replace(col("discount_price"), "[^0-9.]", "").cast(FloatType()))
    .withColumn("actual_price", regexp_replace(col("actual_price"), "[^0-9.]", "").cast(FloatType()))
    .withColumn("main_category", trim(col("main_category")))
    .withColumn("sub_category", trim(col("sub_category")))
    .na.drop(subset=["ratings", "discount_price", "actual_price"]) # drop rows with missing critical values
)

# Step 3: Write cleaned data to Silver layer in Delta format
df_silver.write.format("delta").mode("overwrite").save("s3a://nehabc002bucket/silver/amazon-products/")

# Step 4: Optionally register as a SQL table
spark.sql("""
CREATE TABLE IF NOT EXISTS silver_amazon_products
USING DELTA
LOCATION 's3a://nehabc002bucket/silver/amazon-products/'
""")
```

Code:

```
from pyspark.sql.functions import col, regexp_replace, trim
from pyspark.sql.types import FloatType, IntegerType
```

```
# Step 1: Load data from Bronze layer (CSV)
```

```
df_bronze = spark.read.option("header", "true").csv("s3a://nehabc002bucket/bronze/Amazon-Products.csv")
```

```
# Step 2: Clean the data
```

```
df_silver = (
    df_bronze
    .dropDuplicates(["name", "link"])
    .withColumn("ratings", col("ratings").cast(FloatType()))
    .withColumn("no_of_ratings", regexp_replace(col("no_of_ratings"), ",", "").cast(IntegerType()))
    .withColumn("discount_price", regexp_replace(col("discount_price"), "[^0-9.]", "").cast(FloatType()))
    .withColumn("actual_price", regexp_replace(col("actual_price"), "[^0-9.]", "").cast(FloatType()))
    .withColumn("main_category", trim(col("main_category")))
    .withColumn("sub_category", trim(col("sub_category")))
    .na.drop(subset=["ratings", "discount_price", "actual_price"]) # drop rows with missing critical values
)
```

```

.withColumn("sub_category", trim(col("sub_category")))

.na.drop(subset=["ratings", "discount_price", "actual_price"]) # drop rows with missing critical values
)

```

# Step 3: Write cleaned data to Silver layer in Delta format

```
df_silver.write.format("delta").mode("overwrite").save("s3a://nehabc002bucket/silver/amazon-products/")
```

# Step 4: Optionally register as a SQL table

```
spark.sql("""
```

```
CREATE TABLE IF NOT EXISTS silver_amazon_products
```

```
USING DELTA
```

```
LOCATION 's3a://nehabc002bucket/silver/amazon-products/'
```

```
""")
```

# Display sample

```
df_silver.display()
```

Output:

row_id	name	main_category
1	"PH" POSHAKHUB Women Georgette Hand Embroidery on Neck and Front Yoke Anarkali Kurta"	women's clothing
2	"Pets Empire Retractable Dog Leash Walking Jogging Training Leash with Polyester Tape with Hand Grip and One Button Brake..."	pet supplies
3	"TGC" Light Maroon Colour Dotted Matt Metal Bangle / Mehroon Color Chudi Set for Women and Girls (Pack of 24 Bangles)"	accessories
4	"TGC" Red Colour Dotted Matt Metal Bangle / Lal Color Chudi Set for Women and Girls (Pack of 24 Bangles)"	accessories
5	"10 Pcs Round Design Cases Plastic Clear White 4.8" Diameter DVD CD Box Holder"	tv, audio & cameras
6	"4X Luggage Strap, 2M 78" Adjustable Travel Suitcase Baggage Packing Belt Security Rainbow Straps"	accessories
7	"5 Micron PS-05 10" In PP Spun Filter Candle Set for All Type RO Water Purifier 10 inch for Domestic RO Purifier"	appliances
8	"AIS spares" Butterfly" mixer grinder" 2 Unit Motor and Jar couplers (4 Units   White)"	appliances
9	"ALFASIVA® Soft White Diffuser Cloth Cover for 7"/18cm Studio Standard Strobe Reflector"	tv, audio & cameras
10	"AMPEREUS 5" Cowl Cover for Kitchen Chimney Exhaust Duct Pipe"	appliances
11	"AMPEREUS 9" inch nova/i-Nova/Infinity/Crystal Plus Filter Cartridge with Activated Carbon Block Compatible for Aquaudard ..."	appliances
12	"AMPEREUS 9" inch nova/i-Nova/Infinity/Crystal Plus Filter Cartridge with Activated Carbon Block Compatible for Aquaudard ..."	appliances
13	"AMPEREUS All House Filter Set 20 Inch Jumbo Pre-Filter Housing + Jumbo Spun Filter 20" x 4.5" Dia 1.5 Inch Inlet & Outlet ..."	appliances
14	"AMPEREUS RO Double Wrench Spanner-2 in 1 Model-Suited for 10" Pre Filter Housing and Membrane Housing (White)"	appliances
15	"ACQUAFLUID RO Pre-Filter + Sediment Filter + 4" inch UF Membrane Filter Fiber 0.001 Micron + 4" Inch Mineral Cartrid..."	appliances

Screenshot of a Databricks workspace showing a Python notebook titled "silver\_layer\_job". The notebook contains code for reading data from AWS Glue and performing basic operations. A table view shows 5,815 rows of data with columns: 1.2 ratings, 1.2 no\_of\_ratings, 1.2 discount\_price, and 1.2 actual\_price. The data includes various product IDs and their metrics.

	1.2 ratings	1.2 no_of_ratings	1.2 discount_price	1.2 actual_price
1	4.09999904632568	26	989.0999755859375	1666
2	4	354	799	999
3	4.40000095367432	3	159	499
4	4.40000095367432	6	159	499
5	3.400000953674316	17	549	899
6	4.3000009190734863	39	898	1999
7	3.70000047683716	10	147	199
8	3.900000953674316	49	339	485
9	3.20000047683716	8	198	599
10	3.79999952316284	102	220	399
11	3.5	48	499	999
12	3.5	48	499	999
13	3.70000047683716	21	1599	1999
14	3.5999999046325684	887	179	299
15	4	46	649	1299

Screenshot of the AWS S3 console showing the contents of the "silver" folder within the "nehabc002bucket". The folder contains a single item named "amazon-products/".

The screenshot shows the AWS S3 console with the path `Amazon S3 > Buckets > nehabc002bucket > silver/ > amazon-products/`. The 'Objects' tab is selected, displaying 10 objects. Each object is a parquet file with a name starting with 'part-' followed by a 9-digit ID and ending with '-part-0.parquet'. The last modified date for all files is July 7, 2025, at 21:54:42 UTC-04:00. The file sizes range from 4.8 MB to 5.1 MB, and all are stored in the Standard storage class.

Name	Type	Last modified	Size	Storage class
part-00000-5fbfb486-12f8-4cd6-9363-9e0514433b33-.0000.snappy.parquet	parquet	July 7, 2025, 21:54:42 (UTC-04:00)	4.8 MB	Standard
part-00001-43807760-7735-4a0b-85d7-16fb8207ed6c-.0000.snappy.parquet	parquet	July 7, 2025, 21:54:42 (UTC-04:00)	4.8 MB	Standard
part-00002-96150ba0-0b4a-45c1-aace-07150036e000.snappy.parquet	parquet	July 7, 2025, 21:54:42 (UTC-04:00)	4.9 MB	Standard
part-00003-3bc1b609b46b35-4631-59fc-1eb2351d1838-.0000.snappy.parquet	parquet	July 7, 2025, 21:54:42 (UTC-04:00)	5.0 MB	Standard
part-00004-77e1841-734f-4150-850f-57501329047a-.0000.snappy.parquet	parquet	July 7, 2025, 21:54:42 (UTC-04:00)	4.9 MB	Standard
part-00005-4c790ba-4fb8-46b6-83d8-3323735957ed-.0000.snappy.parquet	parquet	July 7, 2025, 21:54:42 (UTC-04:00)	5.1 MB	Standard
part-00006-13daef0d-31e6-4541-8000-144dd4533333-.0000.snappy.parquet	parquet	July 7, 2025, 21:54:42 (UTC-04:00)	4.9 MB	Standard
part-00007-7246f190-079e-4f7e-96fa-532d099423de-.0000.snappy.parquet	parquet	July 7, 2025, 21:54:42 (UTC-04:00)	5.0 MB	Standard
part-00008-82d5-5353592989ef72-.0000.snappy.parquet	parquet	July 7, 2025, 21:54:45 (UTC-04:00)	1.1 MB	Standard

## Gold Layer: Business Aggregations with PySpark

### Objective:

Metric	Description
Top Categories	Highest-rated product categories
Avg Ratings per Category	Mean rating by main_category
Total Reviews per Category	Sum of no_of_ratings grouped by category
Average Price per Category	Avg of discount_price per sub_category
High-Demand Products	Products with most reviews

### Gold Layer Script in Databricks

The screenshot shows a Databricks workspace with a notebook titled "silver\_layer\_job". The notebook is set to Python and contains the following code:

```
# Gold Layer Script

from pyspark.sql.functions import col, avg, sum, count, desc

# Load Silver layer Delta table
silver_df = spark.read.format("delta").load("s3a://nehabc002bucket/silver/amazon-products/")

# 1. Average rating per main_category
avg_rating_df = (
    silver_df
    .groupBy("main_category")
    .agg(avg("ratings").alias("avg_rating"))
)

# 2. Total reviews per main_category
total_reviews_df = (
    silver_df
    .groupBy("main_category")
    .agg(sum("no_of_ratings").alias("total_reviews"))
)

# 3. Average discount price per sub_category
avg_price_df = (
    silver_df
    .groupBy("sub_category")
    .agg(avg("discount_price").alias("avg_discount_price"))
)
```

Code:

```
# Gold Layer Script
```

```
from pyspark.sql.functions import col, avg, sum, count, desc
```

```
# Load Silver layer Delta table
```

```
silver_df = spark.read.format("delta").load("s3a://nehabc002bucket/silver/amazon-products/")
```

```
# 1. Average rating per main_category
```

```
avg_rating_df = (
    silver_df
    .groupBy("main_category")
    .agg(avg("ratings").alias("avg_rating"))
)
```

```
# 2. Total reviews per main_category
```

```
total_reviews_df = (
    silver_df
)
```

```

    .groupBy("main_category")
    .agg(sum("no_of_ratings").alias("total_reviews"))
)

# 3. Average discount price per sub_category
avg_price_df = (
    silver_df
    .groupBy("sub_category")
    .agg(avg("discount_price").alias("avg_discount_price"))
)

```

# 4. High-demand products (top 10 by no\_of\_ratings)

```

top_products_df = (
    silver_df
    .select("name", "main_category", "sub_category", "ratings", "no_of_ratings")
    .orderBy(desc("no_of_ratings"))
    .limit(10)
)

```

# Save each result to separate folder inside the Gold layer

```

avg_rating_df.write.format("delta").mode("overwrite").save("s3a://nehabc002bucket/gold/avg_rating_by_category/")
total_reviews_df.write.format("delta").mode("overwrite").save("s3a://nehabc002bucket/gold/total_reviews_by_category/")
avg_price_df.write.format("delta").mode("overwrite").save("s3a://nehabc002bucket/gold/avg_price_by_subcategory/")
top_products_df.write.format("delta").mode("overwrite").save("s3a://nehabc002bucket/gold/top_demand_products/")

```

# Optional: Register as SQL Tables

```

spark.sql("CREATE TABLE IF NOT EXISTS gold_avg_rating_by_category USING DELTA
LOCATION 's3a://nehabc002bucket/gold/avg_rating_by_category/'")

```

```

spark.sql("CREATE TABLE IF NOT EXISTS gold_total_reviews_by_category USING DELTA
LOCATION 's3a://nehabc002bucket/gold/total_reviews_by_category/'")

spark.sql("CREATE TABLE IF NOT EXISTS gold_avg_price_by_subcategory USING DELTA
LOCATION 's3a://nehabc002bucket/gold/avg_price_by_subcategory/'")

spark.sql("CREATE TABLE IF NOT EXISTS gold_top_demand_products USING DELTA LOCATION
's3a://nehabc002bucket/gold/top_demand_products/'")

```

### Output Folders in S3:

The screenshot shows the AWS S3 console interface. The URL in the address bar is [us-east-1.console.aws.amazon.com/s3/buckets/nehabc002bucket?region=us-east-1&bucketType=general&prefix=gold/&showversions=false](https://us-east-1.console.aws.amazon.com/s3/buckets/nehabc002bucket?region=us-east-1&bucketType=general&prefix=gold/&showversions=false). The page displays the contents of the 'gold' folder within the 'nehabc002bucket'. There are four objects listed:

Name	Type	Last modified	Size	Storage class
avg_price_by_subcategory/	Folder	-	-	-
avg_rating_by_category/	Folder	-	-	-
top_demand_products/	Folder	-	-	-
total_reviews_by_category/	Folder	-	-	-

Screenshot of the AWS S3 console showing the contents of the 'gold/' folder in the 'nehabc002bucket' bucket. The objects listed are '\_delta\_log/' (Folder) and 'part-00000-17e31d80-e567-405da26f-1fb1ec60055e-c000.snappy.parquet' (parquet).

Name	Type	Last modified	Size	Storage class
_delta_log/	Folder	-	-	-
part-00000-17e31d80-e567-405da26f-1fb1ec60055e-c000.snappy.parquet	parquet	July 7, 2025, 21:48:10 (UTC-04:00)	3.5 KB	Standard

Screenshot of the AWS S3 console showing the contents of the 'avg\_rating\_by\_category/' folder in the 'nehabc002bucket' bucket. The objects listed are '\_delta\_log/' (Folder) and 'part-00000-14599f17-bebd-4fb7-865b-7ebfeda6adb9-c000.snappy.parquet' (parquet).

Name	Type	Last modified	Size	Storage class
_delta_log/	Folder	-	-	-
part-00000-14599f17-bebd-4fb7-865b-7ebfeda6adb9-c000.snappy.parquet	parquet	July 7, 2025, 21:47:52 (UTC-04:00)	1.4 KB	Standard

**Objects (2)**

Name	Type	Last modified	Size	Storage class
_delta_log/	Folder	-	-	-
part-00000-63d04cdd-9b3e-4d78-bc63-ddb885f4964f-c000.snappy.parquet	parquet	July 7, 2025, 21:48:20 (UTC-04:00)	2.4 KB	Standard

**Objects (2)**

Name	Type	Last modified	Size	Storage class
_delta_log/	Folder	-	-	-
part-00000-b95120ab-42f1-4f18-87a7-4979f352073a-c000.snappy.parquet	parquet	July 7, 2025, 21:48:02 (UTC-04:00)	1.3 KB	Standard

## Step 2. Create Glue Crawlers for Gold Tables

Glue Database

The screenshot shows the AWS Glue Databases console. On the left, there's a navigation sidebar with sections like Getting started, ETL jobs, Data Catalog tables, Data Integration and ETL, and Legacy pages. The main area is titled "Databases (2)" and contains a table with two entries:

Name	Description	Location URI	Created on (UTC)
amazon_bronze_db	-	-	July 8, 2025 at 00:44:12
gold_layer_db	-	-	July 8, 2025 at 01:54:20

## Crawler for Gold Data

The screenshot shows the "Add crawler" wizard, Step 2: Choose data source. The sidebar has sections like Getting started, ETL jobs, Data Catalog tables, Data Integration and ETL, and Legacy pages. The main area shows a step-by-step process:

- Step 1: Set crawler properties (disabled)
- Step 2: Choose data source** (highlighted)
- Step 3: Configure security (disabled)
- Step 4: Set output and schedule (disabled)
- Step 5: Review and create (disabled)

In the "Choose data source" section, "S3" is selected as the data source. Below it, there's a "Network connection - optional" section with a note about network connections being shared. The "Location of S3 data" section shows "In this account" selected. The "S3 path" field contains "s3://nehabc002bucket/gold/". The "Subsequent crawler runs" section has "Crawl all sub-folders" selected. At the bottom are "Cancel" and "Add an S3 data source" buttons.

Screenshot of the AWS Glue 'Add crawler' wizard Step 2: Choose data sources and classifiers.

**Choose data sources and classifiers**

**Data source configuration**  
Is your data already mapped to Glue tables?  
 Not yet Select one or more data sources to be crawled.  
 Yes Select existing tables from your Glue Data Catalog.

**Data sources (1) Info**  
The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3	s3://nehabc002bucket/gold/	Recrawl all

**Custom classifiers - optional**  
A classifier checks whether a given file is in a format the crawler can handle. If it is, the classifier creates a schema in the form of a StructType object that matches that data format.

Cancel Previous Next

Screenshot of the AWS Glue 'Add crawler' wizard Step 3: Configure security settings.

**Configure security settings**

**IAM role** [Info](#)  
Existing IAM role  
 glue\_service\_role [View](#) [Create new IAM role](#) [Update chosen IAM role](#)  
 Only IAM roles created by the AWS Glue console and have the prefix "AWSGlueServiceRole" can be updated.

**Lake Formation configuration - optional**  
Allow the crawler to use Lake Formation credentials for crawling the data source. [Learn more](#)  
 Use Lake Formation credentials for crawling S3 data source  
 Checking this box will allow the crawler to use Lake Formation credentials for crawling the data source. If the data source is registered in another account, you must provide the registered account ID. Otherwise, the crawler will crawl only those data sources associated to the account. Only applicable to S3, Glue Catalog, Iceberg, and Hudi data sources.

**Security configuration - optional**  
Enable at-rest encryption with a security configuration.

Cancel Previous Next

Screenshot of the AWS Glue 'Add crawler' wizard Step 4: Set output and scheduling.

The sidebar shows the AWS Glue navigation menu with 'Crawlers' selected. The main area displays the 'Set output and scheduling' step, which includes:

- Output configuration**: Target database is set to 'amazon gold db'. Buttons for 'Clear selection' and 'Add database' are available.
- Table name prefix - optional**: A placeholder 'Type a prefix added to table names' is present.
- Maximum table threshold - optional**: A placeholder 'Type a number greater than 0' is present.
- Advanced options**: A link to expand additional settings.
- Crawler schedule**: A section for defining time-based schedules using Unix-like cron syntax, with a note about generating tables based on schema.
- Frequency**: Set to 'On demand'.

At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

Screenshot of the AWS Glue 'Add crawler' wizard Step 4: Set output and scheduling.

The sidebar shows the AWS Glue navigation menu with 'Crawlers' selected. The main area displays the 'Set output and scheduling' step, which includes:

- Output configuration**: Target database is set to 'gold\_layer\_db'. Buttons for 'Clear selection' and 'Add database' are available.
- Table name prefix - optional**: A placeholder 'Type a prefix added to table names' is present.
- Maximum table threshold - optional**: A placeholder 'Type a number greater than 0' is present.
- Advanced options**: A link to expand additional settings.
- Crawler schedule**: A section for defining time-based schedules using Unix-like cron syntax, with a note about generating tables based on schema.
- Frequency**: Set to 'On demand'.

At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

Screenshot of the AWS Glue Console showing the 'Add crawler' wizard.

**Step 1: Set crawler properties**

- Name: gold\_layer\_crawler
- Description: -
- Tags: -

**Step 2: Choose data sources and classifiers**

- Data sources (1) Info
- Type: S3
- Data source: s3://nehabc002bucket/gold/
- Parameters: Recrawl all

**Step 3: Configure security settings**

- IAM role: glue\_service\_role
- Security configuration: -
- Lake Formation configuration: -

**Step 4: Set output and scheduling**

- Database: gold\_layer\_db
- Table prefix - optional: -
- Maximum table threshold - optional: -
- Schedule: On demand

**Buttons:** Cancel, Previous, Create crawler

Screenshot of the AWS Glue Console showing the 'gold\_layer\_crawler' details page.

**Crawler properties**

- Name: gold\_layer\_crawler
- IAM role: glue\_service\_role
- Description: -
- Database: gold\_layer\_db
- State: READY
- Table prefix: -

**Crawler runs (0)**

The list of crawler runs for this crawler.

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
-	-	-	-	-	-

**Buttons:** Stop run, View CloudWatch logs, View run details, Run crawler

Crawlers - AWS Glue Console

Databases - AWS Glue Console

glue\_service\_role | IAM | Global

silver\_layer\_job - Databricks Co

Private browsing

us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#/v2/data-catalog/crawlers/view/gold\_layer\_crawler

marketplace Deployments

AWS Glue > Crawlers > gold\_layer\_crawler

**gold\_layer\_crawler**

**Crawler successfully starting**  
The following crawler is now starting: "gold\_layer\_crawler"

Last updated (UTC): July 8, 2025 at 01:55:15

**Crawler properties**

Name: gold_layer_crawler	IAM role: glue_service_role	Database: gold_layer_db	State: READY
Description: -	Security configuration: -	Lake Formation configuration: -	Table prefix: -
Maximum table threshold: -			
<b>Advanced settings</b>			

**Crawler runs (1)**  
The list of crawler runs for this crawler.

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
July 8, 2025 at 01:55:22	-	04 s	Running	-	-

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Crawlers - AWS Glue Console

Databases - AWS Glue Console

glue\_service\_role | IAM | Global

silver\_layer\_job - Databricks Co

Private browsing

us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#/v2/data-catalog/crawlers/view/gold\_layer\_crawler

marketplace Deployments

AWS Glue > Crawlers > gold\_layer\_crawler

**gold\_layer\_crawler**

**Crawler successfully starting**  
The following crawler is now starting: "gold\_layer\_crawler"

Last updated (UTC): July 8, 2025 at 01:55:15

**Crawler properties**

Name: gold_layer_crawler	IAM role: glue_service_role	Database: gold_layer_db	State: READY
Description: -	Security configuration: -	Lake Formation configuration: -	Table prefix: -
Maximum table threshold: -			
<b>Advanced settings</b>			

**Crawler runs (1)**  
The list of crawler runs for this crawler.

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
July 8, 2025 at 01:55:22	July 8, 2025 at 01:56:49	01 min 27 s	Completed	-	-

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Screenshot of the AWS Glue Data Catalog console showing the 'gold\_layer\_db' database details.

**Database properties:**

Name	Description	Location	Created on (UTC)
gold_layer_db			July 8, 2025 at 01:54:20

**Tables (4):**

Name	Database	Location	Classification	Deprecated	View data	Data quality	Column statistics
part_00000_14599f17_bebd	gold_layer_db	s3://nehabc002bucket/go	Parquet	-	Table data	View data quality	View statistics
part_00000_17e31db0_e567	gold_layer_db	s3://nehabc002bucket/go	Parquet	-	Table data	View data quality	View statistics
part_00000_63d04cd9_9b3	gold_layer_db	s3://nehabc002bucket/go	Parquet	-	Table data	View data quality	View statistics
part_00000_b95120ab_42f1	gold_layer_db	s3://nehabc002bucket/go	Parquet	-	Table data	View data quality	View statistics

Screenshot of the AWS Glue Data Catalog console showing the 'part\_00000\_14599f17\_bebd\_4fb7\_865b\_7ebfed6adb9\_c000\_snappy\_parquet' table details.

**Table overview:** Data quality - new

**Table details:**

Name	Classification	Deprecated
part_00000_14599f17_bebd_4fb7_865b_7ebfed6adb9_c000_snappy_parquet	Parquet	-

**Schema (2):**

#	Column name	Data type	Partition key	Comment
1	main_category	string	-	-
2	avg_rating	double	-	-

Parquet Viewer

Back

▲ Limited view mode - please purchase more credit to unlock all features.

SQL Query    Chart    Schema    Export

Enter your SQL query here  
Example: SELECT \* FROM data LIMIT 4

**EXECUTE**    CLEAR QUERY

← PREVIOUS    → NEXT

Showing row #1 to #20 of 20 total

main_category - str	avg_rating - f64
women's clothing	3.754666
men's shoes	3.59909
toys & baby products	4.01303
home, kitchen, pets	3.408333
appliances	3.843582
beauty & health	3.985489
tv, audio & cameras	3.815087

Home    Settings    About

nehabc002bucket - S3 bucket | AWS Glue | us-east-1 | Databases - AWS Glue Console | glue\_service\_role | IAM | Global | silver\_layer\_job - Databricks Co | Private browsing

us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#v2/data-catalog/tables/view/part\_00000\_17e31d80\_e567\_405d\_a26f\_1f81ec60055e\_c000\_snappy\_parquet

Last updated (UTC) July 8, 2025 at 01:58:48    Version 0 (Current version)    Actions

AWS Glue    Getting started    ETL jobs    Visual ETL    Notebooks    Job run monitoring    Data Catalog tables    Data connections    Workflows (orchestration)    Zero-ETL Integrations    Data Catalog    Databases    Tables    Stream schema registries    Schemas    Connections    Crawlers    Classifiers    Catalog settings    Data Integration and ETL    Legacy pages

Name: part\_00000\_17e31d80\_e567\_405d\_a26f\_1f81ec60055e\_c000\_snappy\_parquet    Classification: Parquet    Location: s3://nehabc002bucket/gold/avg\_price\_by\_subcategory/part\_00000-17e31d80-e567-405d-a26f-1f81ec60055e-c000.snappy.parquet    Deprecated: No statistics

Table details

Description: -    Last updated: July 8, 2025 at 01:56:45    Connection: -

Table overview    Data quality - new

Advanced properties

Schema (2)    Edit schema as JSON    Edit schema

View and manage the table schema.

Filter schemas

#	Column name	Data type	Partition key	Comment
1	sub_category	string	-	-
2	avg_discount_price	double	-	-

CloudShell    Feedback    © 2025, Amazon Web Services, Inc. or its affiliates.    Privacy    Terms    Cookie preferences

Parquet Viewer

Back

⚠ Limited view mode - please purchase more credit to unlock all features.

SQL Query    Chart    Schema    Export

Enter your SQL query here  
Example: SELECT \* FROM data LIMIT 4

**EXECUTE**    **CLEAR QUERY**

← PREVIOUS    → NEXT

Showing row #1 to #100 of 112 total

sub_category - str	avg_discount_price - f64
Refrigerators	28144.244889
Industrial & Scientific Supplies	791.559875
Strollers & Prams	5451.750653
Heating & Cooling Appliances	4547.45854
Home Improvement	885.996685
Air Conditioners	40818.433022
Casual Shoes	1751.589749

Home    Settings    About

nehabc002bucket - S3 bucket | AWS Glue [us-east-1] | Databases - AWS Glue Console | glue\_service\_role | IAM | Global | silver\_layer\_job - Databricks Co | Private browsing

marketplace Deployments

AWS Glue > Tables > part\_00000\_63d04cdd\_9b3e\_4d78\_bc63\_ddb885f4964f\_c000\_snappy\_parquet

**Table details**

Name: part\_00000\_63d04cdd\_9b3e\_4d78\_bc63\_ddb885f4964f\_c000\_snappy\_parquet

Classification: Parquet

Location: s3://nehabc002bucket/gold/hop\_demand\_products/part-00000-63d04cdd-9b3e-4d78-bc63-ddb885f4964fc000.snappy.parquet

Connection:

Last updated: July 8, 2025 at 01:56:45

**Advanced properties**

**Schema (5)**

View and manage the table schema.

#	Column name	Data type	Partition key	Comment
1	name	string	-	-
2	main_category	string	-	-
3	sub_category	string	-	-
4	ratings	float	-	-
5	no_of_ratings	int	-	-

Edit schema as JSON    Edit schema

CloudShell    Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Parquet Viewer

Back

▲ Limited view mode - please purchase more credit to unlock all features.

SQL Query    Chart    Schema    Export

Enter your SQL query here  
Example: SELECT \* FROM data LIMIT 4

► EXECUTE    ⌂CLEAR QUERY

◀ PREVIOUS    ▶ NEXT

Showing row #1 to #10 of 10 total

name - str	main_category - str	sub_category - str	ratings - f32	no_of_ratings - i32
SanDisk 512GB Ultra MicroSDXC UHS-I Memory Card - 100MB/s, C10, U1, Full HD, A1, Micro SD Card - SDSQUR-512G-GN6MN	tv, audio & cameras	Camera Accessories	4.3	589547
SanDisk Ultra 64GB UHS-I Class 10 Micro SD Memory Card (SDSQUAR-064G-GN3MN)	tv, audio & cameras	Cameras	4.3	589547
SanDisk 400GB Class 10 MicroSD Card (SDSQUAR-400G-GN6MA) with Adapter	tv, audio & cameras	Cameras	4.3	589547
SanDisk UHS-I A1 98Mbps 32GB Ultra MicroSD Memory Card	tv, audio & cameras	Camera Accessories	4.3	589547
SanDisk 128GB Class 10 microSDXC Memory Card with Adapter (SDSQUAR-128G-GN6MA)	tv, audio & cameras	Cameras	4.3	589547
SanDisk 16GB Ultra MicroSDHC Memory Card (SDSQUAR-016G-GN6MN)	tv, audio & cameras	Cameras	4.3	589547
SanDisk 16GB Ultra MicroSDHC Memory Card (SDSQUAR-016G-GN6MN)	tv, audio & cameras	Camera Accessories	4.3	589547

Home    Settings    About

nehabc002bucket | AWS Glue | us-east-1 | Databases - AWS Glue Console | glue\_service\_role | IAM | Global | silver\_layer\_job - Databricks Co | Private browsing

us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#v2/data-catalog/tables/view/part\_00000\_b95120ab\_42f1\_4f18\_87a7\_4979f352073a\_c000\_snappy\_parquet

AWS Glue > Tables > part\_00000\_b95120ab\_42f1\_4f18\_87a7\_4979f352073a\_c000\_snappy\_parquet

Last updated (UTC) July 8, 2025 at 01:59:05   Version 0 (Current version) Actions

Table overview Data quality - new

Table details

Name	part_00000_b95120ab_42f1_4f18_87a7_4979f352073a_c000_snappy_parquet
Classification	Parquet
Location	s3://nehabc002bucket/gold/total_reviews_by_category/part-00000-b95120ab-42f1-4f18-87a7-4979f352073a-c000.snappy.parquet
Connection	-
Deprecation	-
Column statistics	No statistics

Advanced properties

Schema   Partitions   Indexes   Column statistics - new

Schema (2)

View and manage the table schema.

#	Column name	Data type	Partition key	Comment
1	main_category	string	-	-
2	total_reviews	bigint	-	-

Edit schema as JSON   Edit schema

CloudShell   Feedback   © 2025, Amazon Web Services, Inc. or its affiliates.   Privacy   Terms   Cookie preferences

The screenshot shows a Parquet Viewer interface with a yellow banner at the top stating "Limited view mode - please purchase more credit to unlock all features." Below the banner is a search bar with placeholder text "Enter your SQL query here" and an example query "Example: SELECT \* FROM data LIMIT 4". There are buttons for "EXECUTE" and "CLEAR QUERY". Below the search bar are navigation buttons for "PREVIOUS" and "NEXT". A message "Showing row #1 to #20 of 20 total" is displayed. The main content area shows a table with two columns: "main\_category - str" and "total\_reviews - i64". The data includes:

main_category - str	total_reviews - i64
women's clothing	13215786
men's shoes	4777835
toys & baby products	12431732
home, kitchen, pets	47
appliances	16543920
beauty & health	9241585
tv, audio & cameras	173350397

At the bottom are links for "Home", "Settings", and "About".

## ML Layer

### 1: Predict Ratings using Regression

#### Goal:

Predict a product's ratings based on features like discount\_price, actual\_price, no\_of\_ratings, main\_category / sub\_category.

The screenshot shows a Databricks notebook titled "silver\_layer.job" in Python. The code imports necessary libraries and reads a Delta table from S3. It then performs data cleaning on numeric fields. The output shows a sample of the data frame with columns "name", "rating", and "prediction".

```

from pyspark.sql.functions import col, regexp_replace
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml import Pipeline

# Correct way to read Delta
df = spark.read.format("delta").load("s3a://nehabc002bucket/silver/amazon-products/")
# Clean numeric fields
df = (
    df.withColumn("discount_price", regexp_replace(col("discount_price"), "[^0-9.]", "").cast("float"))
    .withColumn("actual_price", regexp_replace(col("actual_price"), "[^0-9.]", "").cast("float"))
    .withColumn("no_of_ratings", regexp_replace(col("no_of_ratings"), "[^0-9]", "").cast("int")))
)

```

name	rating	prediction
"Handicraft-Palace" Women's Caftan Free Size Tunic Kimono Bikini Cover Up Dressing Gown Cotton Blue Abstract Printed Linge...	4.0	[3.7975743290114883]
"PH" POSHAKHUB Women Cotton Printed Anarkali Kurti with Legging	2.9	[3.799682437912147]
"TGC" Haldi Colour Velvet Metal Bangle / Yellow Chudi Set for Women and Girls (Pack of 24 Bangles)	5.0	[3.789172707092521]
"TGC" Kae Colour Dotted Matt Metal Bangle / Dark Green Color Chudi Set for Women and Girls (Pack of 24 Bangles)	3.8	[3.7894922059580964]
"TGC" Mango Yellow Colour Dotted Matt Metal Bangle / Yellow Color Chudi Set for Women and Girls (Pack of 24 Bangles)	4.6	[3.7896958211124594]
"(CD) 18" INCH (450mm) EXHAUST FAN 1440RPM HEAVY DUTY (1.5 YEARS WARRANTY) 250VAC (SINGLE PHASE)	2.9	[3.8316404650322213]
"10 Inches Big LED Ring Light for Camera, Phone tiktok YouTube Video Shooting and Makeup, 10" inch Ring Light with 7 Feet L...	2.9	[3.7958162646456115]

#### Code:

```

from pyspark.sql.functions import col, regexp_replace
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml import Pipeline

# Correct way to read Delta
df = spark.read.format("delta").load("s3a://nehabc002bucket/silver/amazon-products/")

# Clean numeric fields
df = (
    df.withColumn("discount_price", regexp_replace(col("discount_price"), "[^0-9.]", "")).cast("float"))
    .withColumn("actual_price", regexp_replace(col("actual_price"), "[^0-9.]", "")).cast("float"))
    .withColumn("no_of_ratings", regexp_replace(col("no_of_ratings"), ",", "")).cast("int"))
    .withColumn("ratings", col("ratings").cast("float"))
)

# Drop nulls
df = df.dropna(subset=["ratings", "discount_price", "actual_price", "no_of_ratings", "main_category"])

# Encode categorical features
category_indexer = StringIndexer(inputCol="main_category", outputCol="main_category_index")

# Assemble features into a vector
assembler = VectorAssembler(
    inputCols=["discount_price", "actual_price", "no_of_ratings", "main_category_index"],
    outputCol="features"
)

# Create model
lr = LinearRegression(featuresCol="features", labelCol="ratings")

```

```
# Build and run pipeline
pipeline = Pipeline(stages=[category_indexer, assembler, lr])
model = pipeline.fit(df)
```

```
# Make predictions
predictions = model.transform(df)
predictions.select("name", "ratings", "prediction").show(10, truncate=False)
```

```
silver_layer.job Python v
File Edit View Run Help Last edit was 1 minute ago
Run all My Cluster Share Publish
Python
Output Terminal Debug console

6 # ✅ Correct way to read Delta
7 df = spark.read.format("delta").load("s3a://nehabc002bucket/silver/amazon-products/")
8
9 # Clean numeric fields
10 df = (
| name | ratings | prediction |
+-----+-----+
| "Handicraft-Palace" Women's Caftan Free Size Tunic Kimono Bikini Cover Up Dressing Gown Cotton Blue Abstract Printed Linge... | [3.7975743299114883] | |
| "PH" POSHAKHUB Women Cotton Printed Anarkali Kurti with Legging | [2.9] | [3.799682437912147] |
| "TG" Haldi Colour Velvet Metal Bangle / Yellow Chudi Set For Women and Girls (Pack of 24 Bangles) | [5.0] | [3.789172707092521] |
| "TG" Kee Colour Dotted Matt Metal Bangle / Dark Green Color Chudi Set For Women and Girls (Pack of 24 Bangles) | [3.8] | [3.7894922059580964] |
| "TG" Mango Yellow Colour Dotted Matt Metal Bangle / Yellow Color Chudi Set For Women and Girls (Pack of 24 Bangles) | [4.6] | [3.789695821124594] |
| "(CD) 18" INCH (45mm) EXHAUST FAN 1440RPM HEAVY DUTY (1.5 YEARS WARRANTY) 250VAC (SINGLE PHASE) | [2.9] | [3.8316404650322213] |
| "18 Inches Big LED Ring Light for Camera, Phone tiktok YouTube Video Shooting and Makeup, 10" inch Ring Light with 7 Feet L..." | [2.9] | [3.7958162646456115] |
| "A2D 7" 18W 6000K 6 Led Spotlight LED Light Bar ATV LED Fog Lamp Light Driving Working Assembly Set of 2 White For Hyundai ..." | [3.1] | [3.8292999219368933] |
| "AIOIOW 230 V AC Solenoid Valve Normal Close 1/2" 1/2" Hose Pipe Quick Connection RO Water Reverse Osmosis System Suitable..." | [4.0] | [3.8037197572933447] |
| "ADZOY Premium Combo of 10" Selfie Light (3 Light Modes) with 6.5ft Tripod & Collar Mic for Your Personal and Professional ..." | [3.5] | [3.7997893413277159] |
only showing top 10 rows
```

## 2. Isolation Forest in Databricks (via scikit-learn)

### Goal:

Use IsolationForest to detect anomalous products based on discount\_price, actual\_price, ratings, no\_of\_ratings

Silver Layer Job - Python

Last edit was 1 minute ago

10,000+ rows | Truncated data | 2.18s runtime

Step 1: Convert Spark DataFrame to Pandas

```
# Convert selected columns to pandas
selected_df = predictions.select("name", "discount_price", "actual_price", "ratings", "no_of_ratings")
pandas_df = selected_df.toPandas()
```

(1) Spark Jobs

selected\_df: pyspark.sql.dataframe.DataFrame = [name: string, discount\_price: float ... 3 more fields]

Step 2: Fit Isolation Forest

```
from sklearn.ensemble import IsolationForest
```

# Define feature columns

```
features = ["discount_price", "actual_price", "ratings", "no_of_ratings"]
```

# Initialize and train model

Silver Layer Job - Python

Last edit was 2 minutes ago

Run all

My Cluster

Step 2: Fit Isolation Forest

```
from sklearn.ensemble import IsolationForest
```

# Define feature columns

```
features = ["discount_price", "actual_price", "ratings", "no_of_ratings"]
```

# Initialize and train model

```
iso_forest = IsolationForest(contamination=0.02, random_state=42)
pandas_df["anomaly"] = iso_forest.fit_predict(pandas_df[features])
```

# Convert -1 (anomaly) and 1 (normal) into labels

```
pandas_df["anomaly_label"] = pandas_df["anomaly"].apply(lambda x: "Anomaly" if x == -1 else "Normal")
```

/databricks/python/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but IsolationForest was fitted with feature names
warnings.warn(

```

    anomalies_sdf = spark.createDataFrame(pandas_df)
    display(
        anomalies_sdf.select("name", "discount_price", "actual_price", "ratings", "no_of_ratings", "anomaly_label")
        .orderBy("anomaly_label", ascending=False)
    )

```

A <sub>0</sub> name	1.2 discount_price	1.2 actual_price	1.2 ratings	s <sub>0</sub> no_of_ratings	A <sub>0</sub> anomaly_label
1 ELEGANTE UV Protected Transparent Silver Aviator Sunglasses for Men-Pack of 1	198	999	3.29999952316284	373	Normal
2 JN HANDICRAFT Traditional Gold Pearl Stone Chain MangTikka Stylish Hair Jewellery Set for Women Girls(Design-6)	159	499	3.70000047683716	29	Normal
3 Deepsum Potli Bags For Return Gifts For Women Embroidered Potli Purse for Return Gifts, Mehendi, Potli Pouches for Baby Sh...	569.0499877929688	7999	4.19999809265137	24	Normal
4 RANI SAHIBA Women's Chiffon Plain Daily Wear Casual Dupatta	349	999	3.79999952316284	188	Normal
5 FURO Men's R1042 1465 Sneaker	2016	2095	4.300000190734863	2	Normal
6 JNKC 360 Degree Wireless Panoramic Bulb Light 360° IP Camera with Night Vision, 2-Way Audio and Micro 128GB SD Card Suppor...	1795	3999	4	16	Normal
7 Levi's Men's Slim Fit Shirt	1079	3099	2.5999999046325684	12	Normal
8 Brado Jewellery American Diamond One Gram Gold Plated Jewellery Combo of 2 Mangalsutra Pendant Tanmaniya Nallapusal...	218	999	4	71	Normal

10,000+ rows | Truncated data | 33.08s runtime

Convert Back to Spark for Visualization:

```

    anomalies_sdf = spark.createDataFrame(pandas_df)
    display(
        anomalies_sdf.select("name", "discount_price", "actual_price", "ratings", "no_of_ratings", "anomaly_label")
        .orderBy("anomaly_label", ascending=False)
    )

```

A <sub>0</sub> name	1.2 discount_price	1.2 actual_price	1.2 ratings	s <sub>0</sub> no_of_ratings	A <sub>0</sub> anomaly_label
1 ELEGANTE UV Protected Transparent Silver Aviator Sunglasses for Men-Pack of 1	198	999	3.29999952316284	373	Normal
2 JN HANDICRAFT Traditional Gold Pearl Stone Chain MangTikka Stylish Hair Jewellery Set for Women Girls(Design-6)	159	499	3.70000047683716	29	Normal
3 Deepsum Potli Bags For Return Gifts For Women Embroidered Potli Purse for Return Gifts, Mehendi, Potli Pouches for Baby Sh...	569.0499877929688	7999	4.19999809265137	24	Normal
4 RANI SAHIBA Women's Chiffon Plain Daily Wear Casual Dupatta	349	999	3.79999952316284	188	Normal
5 FURO Men's R1042 1465 Sneaker	2016	2095	4.300000190734863	2	Normal
6 JNKC 360° IP Camera with Night Vision, 2-Way Audio and Micro 128GB SD Card Suppor...	1795	3999	4	16	Normal
7 Levi's Men's Slim Fit Shirt	1079	3099	2.5999999046325684	12	Normal
8 Brado Jewellery American Diamond One Gram Gold Plated Jewellery Combo of 2 Mangalsutra Pendant Tanmaniya Nallapusal...	218	999	4	71	Normal

10,000+ rows | Truncated data | 33.08s runtime

### 3. K-Means Clustering in Databricks (PySpark MLlib)

Step 1: Prepare the Data

Silver Layer Job - Python

Step 1: Load & Prepare the Data

```
1 minute ago (16)
from pyspark.sql.functions import col, regexp_replace
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans
from pyspark.ml import Pipeline

# Load from Silver layer
df = spark.read.format("delta").load("s3a://nehabc002bucket/silver/amazon-products/")

# Clean and cast columns
df = (
    df.withColumn("discount_price", regexp_replace("discount_price", "[^0-9.]", "") .cast("float"))
    .withColumn("actual_price", regexp_replace("actual_price", "[^0-9.]", "") .cast("float"))
    .withColumn("ratings", col("ratings") .cast("float"))
    .withColumn("no_of_ratings", regexp_replace("no_of_ratings", ",", "") .cast("int"))
    .dropna(subset=["discount_price", "actual_price", "ratings", "no_of_ratings"])
)
```

Silver Layer Job - Python

Step 2: Assemble Features & Fit KMeans Model

```
1 minute ago (16)
df: pyspark.sql.dataframe.DataFrame = [row_id: string, name: string ... 8 more fields]
```

```
# Step 2: Assemble numeric features
feature_cols = ["discount_price", "actual_price", "ratings", "no_of_ratings"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")

# Step 3: Define and fit the KMeans model
kmeans = KMeans(featuresCol="features", predictionCol="cluster", k=4, seed=42)

# Step 4: Build and run the pipeline
pipeline = Pipeline(stages=[assembler, kmeans])
model = pipeline.fit(df)

# Step 5: Transform the data to get cluster predictions
clustered_df = model.transform(df)
```

(19) Spark Jobs

```
clustered_df: pyspark.sql.dataframe.DataFrame
```

Step 3: View Clusters

```

# Visualize clusters
display(
    clustered_df.select("name", "main_category", "discount_price", "ratings", "no_of_ratings", "cluster")
)

```

	name	main_category	discount_price	ratings	no_of_ratings	cluster
1	"Handicraft-Palace" Women's Caftan Free Size Tunic Kimono Bikini Cover Up Dressing Gown Cotton Blue Abstract Printed Linge..."	women's clothing				
2	"PH" POSHAKHUB Women Cotton Printed Anarkali Kurti With Legging"	women's clothing	1169.0999			
3	"TGC" Haldi Colour Velvet Metal Bangle / Yellow Chudi Set for Women and Girls (Pack of 24 Bangles)"	accessories				
4	"TGC" Kae Color Dotted Matt Metal Bangle / Dark Green Color Chudi Set for Women and Girls (Pack of 24 Bangles)"	accessories				
5	"TGC" Mango Yellow Colour Dotted Matt Metal Bangle / Yellow Color Chudi Set for Women and Girls (Pack of 24 Bangles)"	accessories				
6	"CD) 18" INCH (450mm) EXHAUST FAN 1440RPM HEAVY DUTY (5 YEARS WARRANTY) 250VAC (SINGLE PHASE)"	appliances				
7	"10 Inches Big LED Ring Light for Camera, Phone tiktok YouTube Video Shooting and Makeup, 10" inch Ring Light with 7 Feet L..."	tv, audio & cameras				
8	"A2B 7" 18W 6000K 6 Led Spotlight LED Light Bar ATV LED Fog Lamp Light Driving Working Assembly Set of 2 White For Hyundai..."	car & motorbike				
9	"AOOB 230 V AC Solenoid Valve Normal Close 1/2" 1/2"" Hose Pipe Quick Connection RO Water Reverse Osmosis System Suitable..."	appliances				

	name	main_category	discount_price	ratings	no_of_ratings	cluster
1	"Handicraft-Palace" Women's Caftan Free Size Tunic Kimono Bikini Cover Up Dressing Gown Cotton Blue Abstract Printed Linge..."	women's clothing	749	4	3	0
2	"PH" POSHAKHUB Women Cotton Printed Anarkali Kurti With Legging"	women's clothing	1169.099975589375	2.900000953674316	20	0
3	"TGC" Haldi Colour Velvet Metal Bangle / Yellow Chudi Set for Women and Girls (Pack of 24 Bangles)"	accessories	159	5	1	0
4	"TGC" Kae Color Dotted Matt Metal Bangle / Dark Green Color Chudi Set for Women and Girls (Pack of 24 Bangles)"	accessories	159	3.79999952316284	21	0
5	"TGC" Mango Yellow Colour Dotted Matt Metal Bangle / Yellow Color Chudi Set for Women and Girls (Pack of 24 Bangles)"	accessories	159	4.59999904632568	7	0
6	"CD) 18" INCH (450mm) EXHAUST FAN 1440RPM HEAVY DUTY (5 YEARS WARRANTY) 250VAC (SINGLE PHASE)"	appliances	4650	2.900000953674316	8	0
7	"10 Inches Big LED Ring Light for Camera, Phone tiktok YouTube Video Shooting and Makeup, 10" inch Ring Light with 7 Feet L..."	tv, audio & cameras	709	2.900000953674316	11	0
8	"A2B 7" 18W 6000K 6 Led Spotlight LED Light Bar ATV LED Fog Lamp Light Driving Working Assembly Set of 2 White For Hyundai..."	car & motorbike	488	3.099999946325684	50	0
9	"AOOB 230 V AC Solenoid Valve Normal Close 1/2" 1/2"" Hose Pipe Quick Connection RO Water Reverse Osmosis System Suitable..."	appliances	390	4	109	0
10	"Bass Maxie Plus V2" (Steel)"	tv, audio & cameras	999	3.5	16	0
11	"Bass Maxie Plus V2" (Steel)"	appliances	849	2.79999952316284	3	0
12	"Standard Studio Strobe Reflector"	tv, audio & cameras	332	3.400000953674316	25	0
13	"Standard Studio Strobe Reflector(Pack of 4)"	tv, audio & cameras	298	3.20000047683716	8	0
14	"Tenn Manifold Coupling Union Conversion Kit 1/4" -"	appliances	111	2.700000953674316	4	0
15	"Tenn Manifold Coupling Union Conversion Kit 1/4" -"	appliances	111	2.700000953674316	4	0

Databricks Code:

<https://community.cloud.databricks.com/editor/notebooks/35984783221329?o=1049562611814646#notebook/35984783221330/command/35984783221330>

## 4. Batch Inference in Databricks

Goal:

Predict Ratings for Products (Batch Inference)

```
# STEP 0: IMPORT LIBRARIES
import pandas as pd
import boto3
import joblib
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split

# STEP 1: LOAD SILVER DATASET FROM S3
df = spark.read.format("delta").load("s3a://nehabc002bucket/silver/amazon-products/")

# STEP 2: CLEAN AND PREPARE DATA
df = df[["name", "discount_price", "actual_price", "no_of_ratings", "ratings"]].dropna()

from pyspark.sql.functions import regexp_replace, col

# Clean and cast columns
df = df.withColumn("discount_price", regexp_replace("discount_price", "[\$,]", "").cast("float"))
df = df.withColumn("actual_price", regexp_replace("actual_price", "[\$,]", "").cast("float"))
df = df.withColumn("no_of_ratings", regexp_replace("no_of_ratings", "[\$,]", "").cast("int"))
```

Code:

```
# STEP 0: IMPORT LIBRARIES
import pandas as pd
import boto3
import joblib
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split

# STEP 1: LOAD SILVER DATASET FROM S3
df = spark.read.format("delta").load("s3a://nehabc002bucket/silver/amazon-products/")

# STEP 2: CLEAN AND PREPARE DATA
df = df[["name", "discount_price", "actual_price", "no_of_ratings", "ratings"]].dropna()
```

```
from pyspark.sql.functions import regexp_replace, col

# Clean and cast columns

df = df.withColumn("discount_price", regexp_replace("discount_price", "[₹,]", "")).cast("float"))
df = df.withColumn("actual_price", regexp_replace("actual_price", "[₹,]", "")).cast("float"))
df = df.withColumn("no_of_ratings", regexp_replace("no_of_ratings", ",","", "").cast("int"))
df = df.withColumn("ratings", col("ratings").cast("float"))

# Convert to Pandas

df_pd = df.select("name", "discount_price", "actual_price", "no_of_ratings",
"ratings").dropna().toPandas()

# Prepare features and labels from Pandas

X = df_pd[["discount_price", "actual_price", "no_of_ratings"]]
y = df_pd["ratings"]

# Train/test split and model training

from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

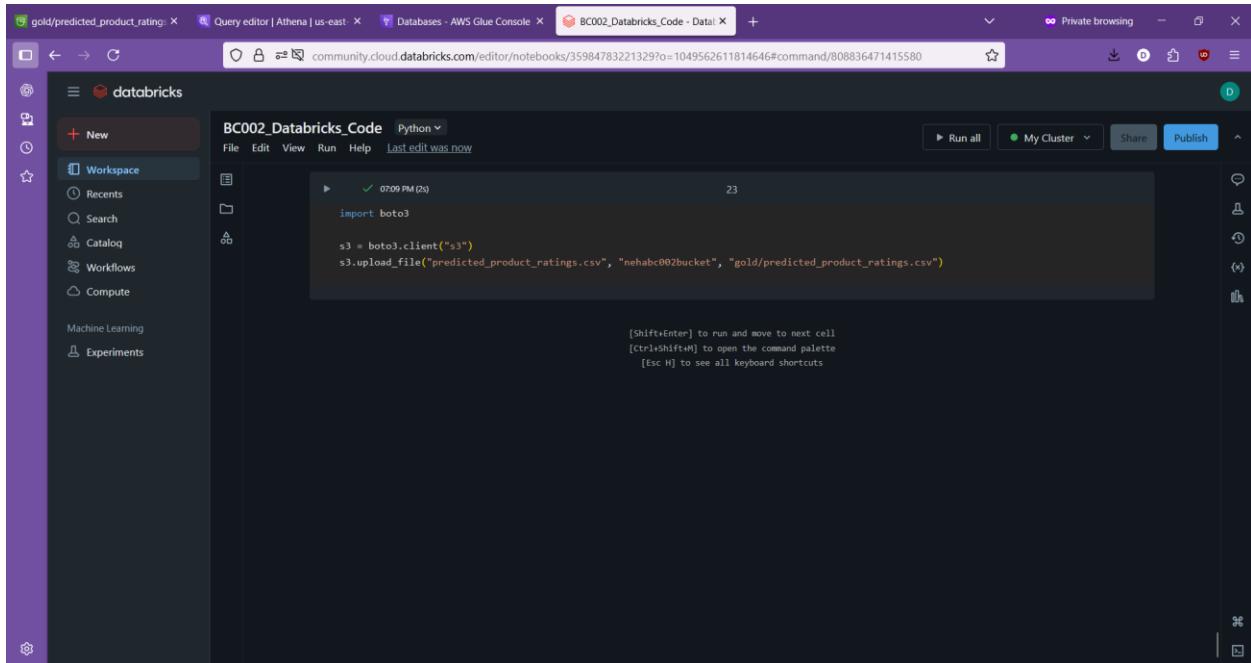
model = XGBRegressor()
model.fit(X_train, y_train)

# Predict

df_pd["predicted_rating"] = model.predict(X)
```

```
# Save result
```

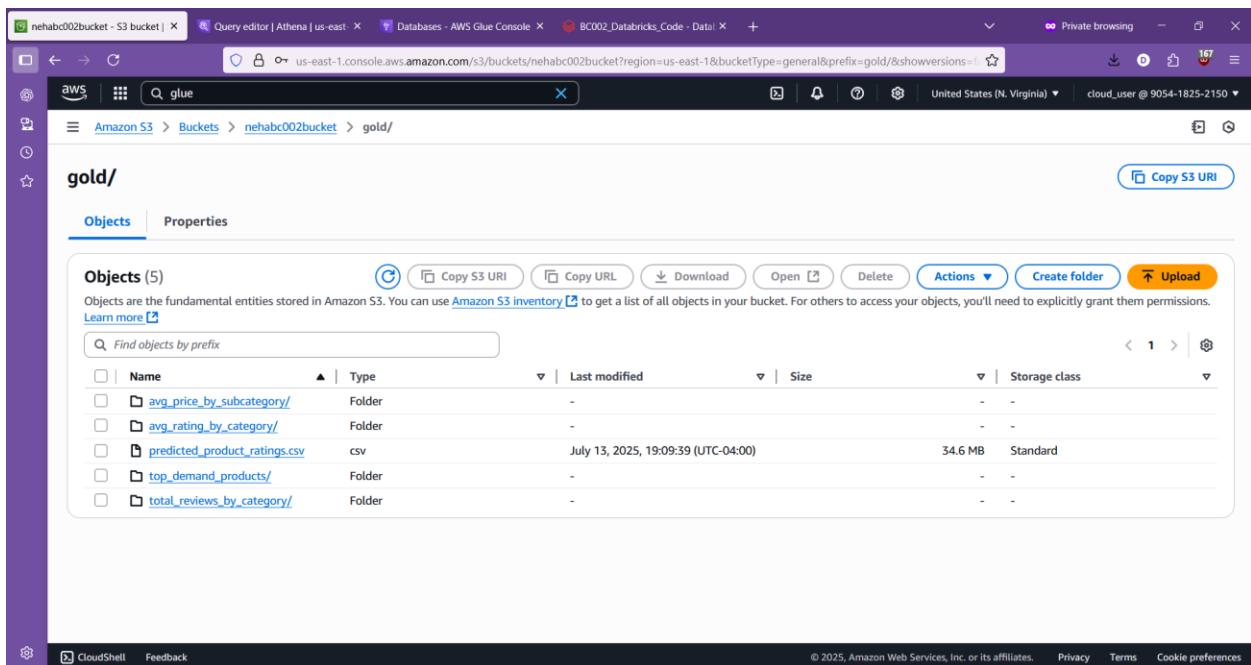
```
df_pd[["name", "discount_price", "actual_price", "no_of_ratings",
"predicted_rating"]].to_csv("predicted_product_ratings.csv", index=False)
```



The screenshot shows a Databricks notebook interface. The left sidebar has a 'New' button and sections for Workspace, Recents, Search, Catalog, Workflows, Compute, Machine Learning, and Experiments. The main area is titled 'BC002\_Databricks\_Code' and contains a single code cell:

```
import boto3
s3 = boto3.client("s3")
s3.upload_file("predicted_product_ratings.csv", "nehabc002bucket", "gold/predicted_product_ratings.csv")
```

Below the code cell, there are keyboard shortcut instructions: [Shift+Enter] to run and move to next cell, [Ctrl+Shift+M] to open the command palette, and [Esc H] to see all keyboard shortcuts.



The screenshot shows the AWS S3 console. The URL is [us-east-1.console.aws.amazon.com/s3/buckets/nehabc002bucket?region=us-east-1&bucketType=general&prefix=gold&showversions=false](https://us-east-1.console.aws.amazon.com/s3/buckets/nehabc002bucket?region=us-east-1&bucketType=general&prefix=gold&showversions=false). The page displays the 'gold' folder under 'Objects'. There are five objects listed:

Name	Type	Last modified	Size	Storage class
avg_price_by_subcategory/	Folder	-	-	-
avg_rating_by_category/	Folder	-	-	-
predicted_product_ratings.csv	csv	July 13, 2025, 19:09:39 (UTC-04:00)	34.6 MB	Standard
top_demand_products/	Folder	-	-	-
total_reviews_by_category/	Folder	-	-	-

At the bottom, there are links for CloudShell, Feedback, and cookie preferences.

Screenshot of the AWS S3 console showing the object details for 'predicted\_product\_ratings.csv'.

**Object overview**

- Owner:** lab-aws+LabServices-prod-12463
- AWS Region:** US East (N. Virginia) us-east-1
- Last modified:** July 13, 2025, 19:09:39 (UTC-04:00)
- Size:** 34.6 MB
- Type:** CSV
- Key:** gold/predicted\_product\_ratings.csv

**Properties** | Permissions | Versions

**S3 URI:** s3://nehabc002bucket/gold/predicted\_product\_ratings.csv

**Amazon Resource Name (ARN):** arn:aws:s3:::nehabc002bucket/gold/predicted\_product\_ratings.csv

**Entity tag (Etag):** 98fe7ee8ae457c5017582bafaf2a1b54-5

**Object URL:** https://nehabc002bucket.s3.us-east-1.amazonaws.com/gold/predicted\_product\_ratings.csv

**Object management overview**

## Quicksight

### Sum of Predicted\_rating by Name

SHOWING TOP 2500 IN NAME

