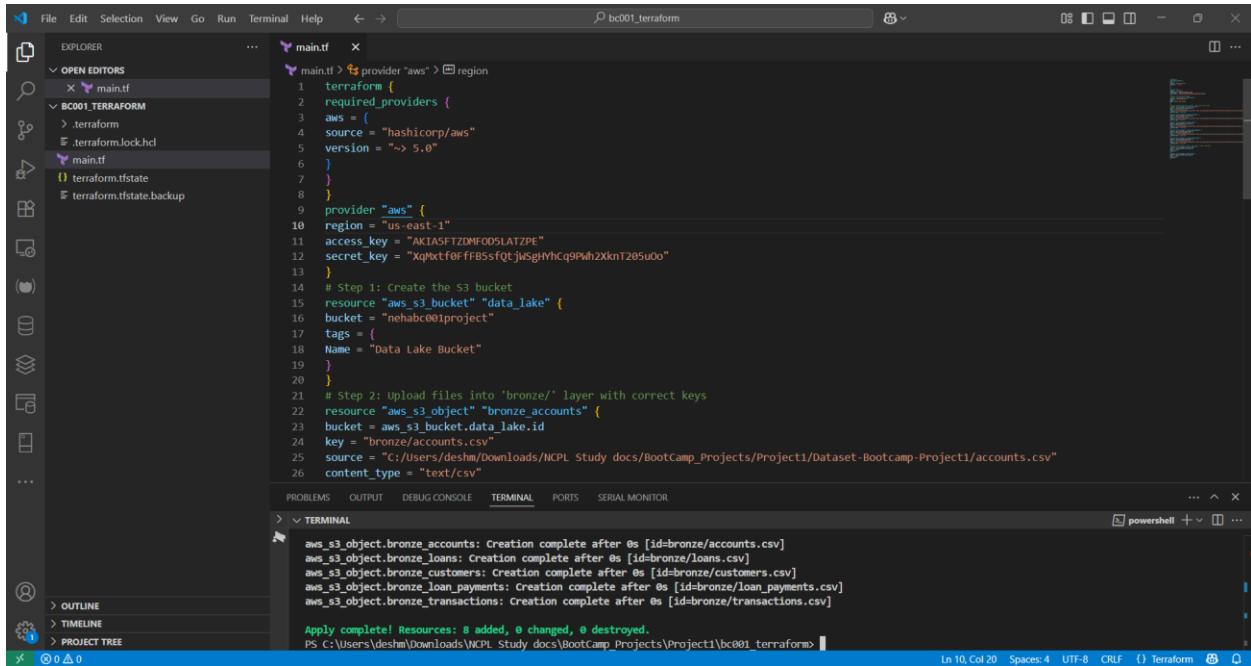


Managing Infrastructure

Created Resources using Terraform



```

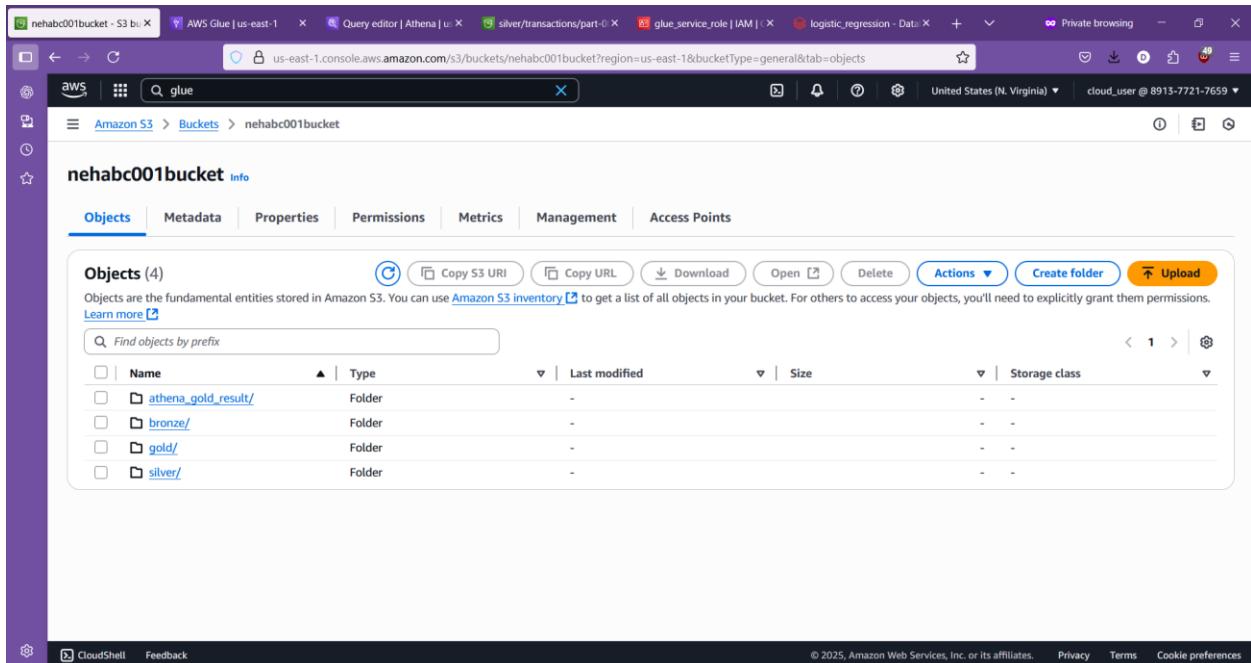
provider "aws" {
  region = "us-east-1"
  access_key = "AKIA5FTZDMFOOSLATZPE"
  secret_key = "Xqkxtf0ffB5fqTjwSgfhCq9WhzXknT205uOo"
}

resource "aws_s3_bucket" "data_lake" {
  bucket = "nehabc001project"
  tags = {
    Name = "Data Lake Bucket"
  }
}

resource "aws_s3_object" "bronze_accounts" {
  bucket = aws_s3_bucket.data_lake.id
  key = "bronze/accounts.csv"
  source = "C:/Users/deshm/Downloads/NCPL Study docs/BootCamp_Projects/Project1/Dataset-Bootcamp-Project1/accounts.csv"
  content_type = "text/csv"
}

```

S3 Buckets



Name	Type	Last modified	Size	Storage class
athena_gold_result/	Folder	-	-	-
bronze/	Folder	-	-	-
gold/	Folder	-	-	-
silver/	Folder	-	-	-

IAM Roles and Policies

Identity and Access Management (IAM)

glue_service_role info

Allows Glue to call AWS services on your behalf.

Summary

Creation date: July 06, 2025, 14:45 (UTC-04:00)

Last activity: 4 hours ago

ARN: arn:aws:iam::891377217659:role/glue_service_role

Maximum session duration: 1 hour

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (4)

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonAthenaFullAccess	AWS managed	1
AmazonS3FullAccess	AWS managed	1
AWSGlueServiceRole	AWS managed	1
CloudWatchEventsFullAccess	AWS managed	1

Glue Databases

Databases (2)

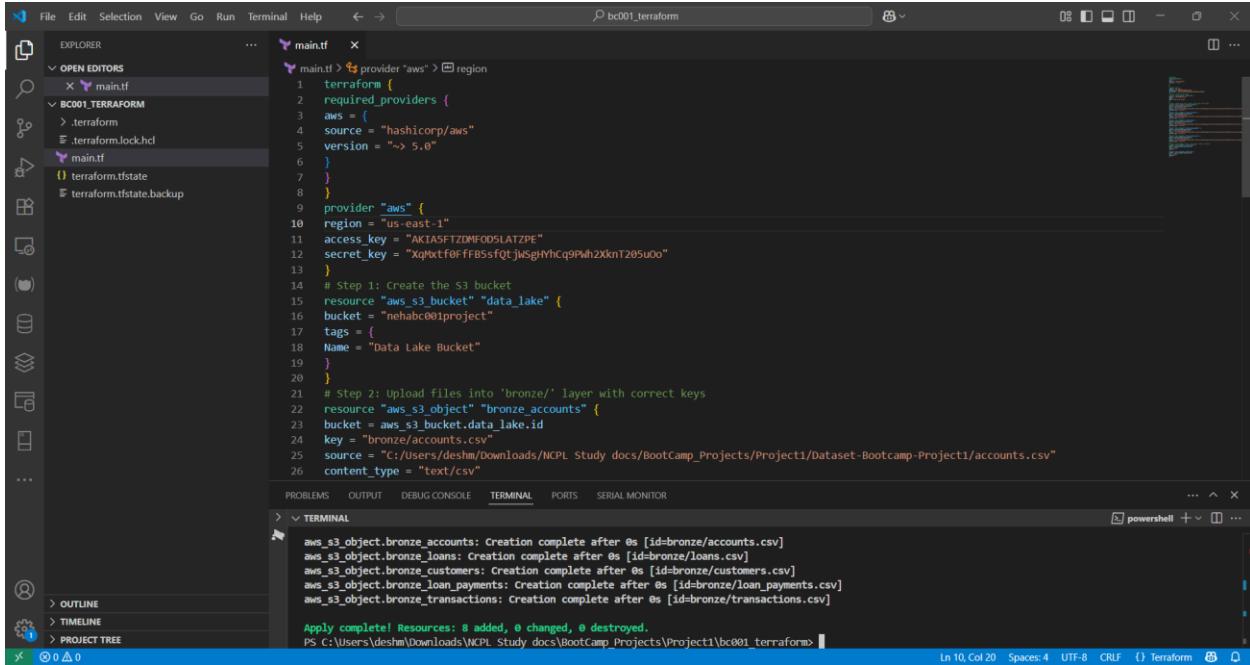
A database is a set of associated table definitions, organized into a logical group.

Name	Description	Location URI	Created on (UTC)
gold_layer_db	-	-	July 6, 2025 at 19:00:27
silver_layer_db	-	-	July 6, 2025 at 18:43:20

Data Ingestion - Bronze Layer (Raw Data Collection)

Step 1. Create S3 Bucket using Terraform

Step 1.1. Create your Terraform configuration file.



```

provider "aws" {
  region = "us-east-1"
  access_key = "AKIA5FTZDMFO05LATZPE"
  secret_key = "xqhfctffrff85f0tjw5gfhvhpwh2knt205u00"
}

resource "aws_s3_bucket" "data_lake" {
  bucket = "nehabc001project"
  tags = {
    Name = "Data Lake Bucket"
  }
}

resource "aws_s3_object" "bronze_accounts" {
  bucket = aws_s3_bucket.data_lake.id
  key = "bronze/accounts.csv"
  source = "c:/Users/deshm/Downloads/NCPL Study docs/BootCamp_Projects/Project1/Dataset-Bootcamp-Project1/accounts.csv"
  content_type = "text/csv"
}

```

Code:



```
terraform {
```

```
  required_providers {
```

```
    aws = {
```

```
      source = "hashicorp/aws"
```

```
      version = "~> 5.0"
```

```
}
```

```
}
```

```
}
```

```
  provider "aws" {
```

```
    region = "us-east-1"
```

```

access_key = "AKIA5FTZDMFOD5LATZPE"
secret_key = "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"
}

# Step 1: Create the S3 bucket

resource "aws_s3_bucket" "data_lake" {
  bucket = "nehabc001project"
  tags = {
    Name = "Data Lake Bucket"
  }
}

# Step 2: Upload files into 'bronze/' layer with correct keys

resource "aws_s3_object" "bronze_accounts" {
  bucket = aws_s3_bucket.data_lake.id
  key = "bronze/accounts.csv"
  source = "C:/Users/deshm/Downloads/NCPL Study docs/BootCamp_Projects/Project1/Dataset-Bootcamp-Project1/accounts.csv"
  content_type = "text/csv"
}

resource "aws_s3_object" "bronze_customers" {
  bucket = aws_s3_bucket.data_lake.id
  key = "bronze/customers.csv"
  source = "C:/Users/deshm/Downloads/NCPL Study docs/BootCamp_Projects/Project1/Dataset-Bootcamp-Project1/customers.csv"
  content_type= "text/csv"
}

resource "aws_s3_object" "bronze_loans" {
  bucket = aws_s3_bucket.data_lake.id
  key = "bronze/loans.csv"
  source = "C:/Users/deshm/Downloads/NCPL Study docs/BootCamp_Projects/Project1/Dataset-Bootcamp-Project1/loans.csv"
  content_type = "text/csv"
}

```

```

}

resource "aws_s3_object" "bronze_loan_payments" {
  bucket = aws_s3_bucket.data_lake.id
  key    = "bronze/loan_payments.csv"
  source = "C:/Users/deshm/Downloads/NCPL Study docs/BootCamp_Projects/Project1/Dataset-Bootcamp-Project1/loan_payments.csv"
  content_type = "text/csv"
}

resource "aws_s3_object" "bronze_transactions" {
  bucket = aws_s3_bucket.data_lake.id
  key    = "bronze/transactions.csv"
  source = "C:/Users/deshm/Downloads/NCPL Study docs/BootCamp_Projects/Project1/Dataset-Bootcamp-Project1/transactions.csv"
  content_type = "text/csv"
}

# Step 3: Create empty 'silver/' and 'gold/' folders (optional)

resource "aws_s3_object" "silver_layer" {
  bucket = aws_s3_bucket.data_lake.id
  key    = "silver/"
  content = ""
}

resource "aws_s3_object" "gold_layer" {
  bucket = aws_s3_bucket.data_lake.id
  key    = "gold/"
  content = ""
}

```

Step 1.2. To run a code and create s3 bucket run this command on terminal: terraform init

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows the project structure with files: main.tf, .terraform, .terraform.lock.hcl, main.tfstate, and terraform.tfstate.backup.
- TERMINAL:** Shows the command-line interface running on PowerShell. The output of the `terraform init` command is displayed, indicating successful initialization of the backend and provider plugins.

```

PS C:\Users\deshm\Downloads\WCP1_Study_docs\BootCamp_Projects\Project1\bc001_terraform> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.100.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\deshm\Downloads\WCP1_Study_docs\BootCamp_Projects\Project1\bc001_terraform> terraform validate
Terraform has been successfully initialized!

```

Step 1.3. Run command: terraform validate

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows the project structure with files: main.tf, .terraform, .terraform.lock.hcl, main.tfstate, and terraform.tfstate.backup.
- TERMINAL:** Shows the command-line interface running on PowerShell. The output of the `terraform validate` command is displayed, showing success with the message "Success! The configuration is valid."

```

PS C:\Users\deshm\Downloads\WCP1_Study_docs\BootCamp_Projects\Project1\bc001_terraform> terraform validate
Success! The configuration is valid.

PS C:\Users\deshm\Downloads\WCP1_Study_docs\BootCamp_Projects\Project1\bc001_terraform> terraform plan

```

Step 1.4. Run command: terraform plan

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like `main.tf`, `.terraform`, `.terraform.lock.hcl`, `main.tfstate`, and `terraform.tfstate.backup`.
- Code Editor:** Displays the `main.tf` file content:


```
1 terraform {
2   required_providers {
3     aws = {
4       source  = "hashicorp/aws"
5       version = ">= 5.0"
6     }
7   }
8 }
```
- Terminal:** Shows the command `terraform plan` being run in a PowerShell window. The output indicates that an AWS S3 bucket named "Data Lake Bucket" will be created. The terminal also shows the following message: "Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols: + create".

Step 1.5. Run command: `terraform apply`

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like `main.tf`, `.terraform`, `.terraform.lock.hcl`, `main.tfstate`, and `terraform.tfstate.backup`.
- Code Editor:** Displays the `main.tf` file content, identical to the previous screenshot.
- Terminal:** Shows the command `terraform apply` being run in a PowerShell window. The output includes a note: "Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now." It also shows the same execution plan as the previous step, indicating the creation of the "Data Lake Bucket" S3 bucket.

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure with files: main.tf, .terraform, .terraform.lock.hcl, main.tfstate, and terraform.tfstate.backup.
- TERMINAL:** Shows the execution of Terraform commands and the resulting output. The output indicates the creation of various AWS S3 objects and layers, such as gold_layer, silver_layer, and bronze_customers, along with their associated accounts and loan payments.
- CODE:** The main.tf file is open, showing the Terraform configuration code.

```

resource "aws_s3_object" "gold_layer"
  terraform {
    required_providers {
      aws = {
        source = "hashicorp/aws"
        version = ">= 5.0"
      }
    }
  }
  provider "aws" {}

```

Step 2. Goto AWS console and check s3 bucket.

The screenshot shows the AWS S3 console with the following details:

- Left Sidebar:** Shows the navigation menu for Amazon S3, including General purpose buckets, Directory buckets, Table buckets, Access Grants, Access Points for general purpose buckets, Access Points for directory buckets, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and IAM Access Analyzer for S3.
- Right Content Area:** Displays the **General purpose buckets** section. It shows one bucket named **nehab001project**. The bucket details include:
 - Name:** nehab001project
 - AWS Region:** US East (N. Virginia) us-east-1
 - IAM Access Analyzer:** View analyzer for us-east-1
 - Creation date:** June 25, 2025, 20:54:56 (UTC-04:00)

Subfolders in bucket:

The screenshot shows the AWS S3 console interface. On the left, the navigation sidebar for 'Amazon S3' is visible, featuring sections for General purpose buckets, Storage Lens, and IAM Access Analyzer for S3. The main content area displays the 'nehabc001project' bucket. At the top, there are tabs for Objects, Metadata, Properties, Permissions, Metrics, Management, and Access Points. The 'Objects' tab is selected, showing a list of 3 objects: 'bronze/' (Folder), 'gold/' (Folder), and 'silver/' (Folder). Below the list is a search bar labeled 'Find objects by prefix'.

Uploaded csv files in the folder bronze:

This screenshot shows the contents of the 'bronze/' folder within the 'nehabc001project' bucket. The navigation sidebar on the left remains the same. The main content area shows the 'bronze/' folder with 5 objects. The 'Properties' tab is selected. The objects listed are: 'accounts.csv' (csv file, 9.3 KB, Standard storage), 'customers.csv' (csv file, 13.0 KB, Standard storage), 'loan_payments.csv' (csv file, 23.6 KB, Standard storage), 'loans.csv' (csv file, 5.0 KB, Standard storage), and 'transactions.csv' (csv file, 62.8 KB, Standard storage). Each object has a preview icon, a 'Copy S3 URI' button, and a 'Actions' dropdown menu.

Data Processing - Silver Layer (Cleaned, Trusted Data)

The Silver Layer, also called the Clean Layer, contains data that has been refined from the raw Bronze layer. This includes:

- Filtering out bad data
- Normalizing formats
- Enforcing schema
- Resolving duplicates
- Enriching with business context

Step 1: Create a database in AWS glue.

The screenshot shows the AWS Glue Console interface. On the left, there's a navigation sidebar with sections like 'AWS Glue', 'Data Catalog', 'Data Integration and ETL', and 'Legacy pages'. The main content area is titled 'Databases (1)' and contains a table with one entry:

Name	Description	Location URI	Created on (UTC)
silver_layer_db			June 28, 2025 at 23:12:12

Step 2: Create IAM role for glue.

Select trusted entity

Trusted entity type

- AWS service
- AWS account
- Web identity
- SAML 2.0 federation
- Custom trust policy

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case
Glue

Choose a use case for the specified service.
Use case
 Glue

Allows Glue to call AWS services on your behalf.

Add permissions

Permissions policies (3/1057)

Choose one or more policies to attach to your new role.

Filter by Type

Policy name	Type	Description
<input type="checkbox"/> AdministratorAccess	AWS managed - job function	Provides full access to AWS services an...
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	Grants account administrative permis...
<input type="checkbox"/> AdministratorAccess-AWSElasticBeanstalk	AWS managed	Grants account administrative permis...
<input type="checkbox"/> AIOpsAssistantPolicy	AWS managed	Provides ReadOnly permissions requir...
<input type="checkbox"/> AIOpsConsoleAdminPolicy	AWS managed	Grants full access to Amazon AI Opera...
<input type="checkbox"/> AIOpsOperatorAccess	AWS managed	Grants access to the Amazon AI Opera...
<input type="checkbox"/> AIOpsReadOnlyAccess	AWS managed	Grants ReadOnly permissions to the A...
<input type="checkbox"/> AlexaForBusinessDeviceSetup	AWS managed	Provide device setup access to AlexaFo...
<input type="checkbox"/> AlexaForBusinessFullAccess	AWS managed	Grants full access to AlexaForBusiness ...

Screenshot of the AWS IAM 'Create role' wizard - Step 1: Name, review, and create.

Role details

Role name: glue_service_role

Description: Allows Glue to call AWS services on your behalf.

Step 1: Select trusted entities

Trust policy:

```

1  {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Principal": {
7         "Service": "glue.amazonaws.com"
8       }
9     }
10  ]
11 }
```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AmazonS3FullAccess	AWS managed	Permissions policy
AWSGlueServiceRole	AWS managed	Permissions policy
CloudWatchEventsFullAccess	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional

No tags associated with the resource.

Create role

Identity and Access Management (IAM)

glue_service_role Info

Allows Glue to call AWS services on your behalf.

Summary

Creation date: June 25, 2025, 21:11 (UTC-04:00)

Last activity: -

ARN: arn:aws:iam::905418334556:role/glue_service_role

Maximum session duration: 1 hour

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (3) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	1
AWSGlueServiceRole	AWS managed	1
CloudWatchEventsFullAccess	AWS managed	1

Step 3: Create silver layer job to perform transformations

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

What's New

Documentation

BC001_Silver_Transform

Last modified on 2025-06-28, 7:26:05 p.m.

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

```

Script Info
1 import sys
2 from pyspark.context import SparkContext
3 from pyspark.sql import SparkSession
4 from pyspark.sql.functions import col, trim, to_date, to_timestamp
5 from pyspark.sql.types import IntegerType, DoubleType
6 from awsglue.context import glueContext
7 from awsglue.utils import getResolvedOptions
8 from awsglue.job import Job
9
10 # Job arguments
11 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
12
13 # Glue context and Spark session
14 sc = SparkContext()
15 glueContext = GlueContext(sc)
16 spark = glueContext.spark_session
17 job = Job(glueContext)
18 job.init(args['JOB_NAME'], args)
19

```

Python | Ln 44, Col 53 | Errors: 0 | Warnings: 0

Code:

```

import sys

from pyspark.context import SparkContext

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, trim, to_date, to_timestamp

```

```

from pyspark.sql.types import IntegerType, DoubleType
from awsglue.context import GlueContext
from awsglue.utils import getResolvedOptions
from awsglue.job import Job

# Job arguments
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

# Glue context and Spark session
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# -----
# Read Raw Bronze Data
# -----


bronze_path = "s3://nehaawsbc001/bronze/"


# Read each CSV file
customers_df = spark.read.option("header", "true").csv(f'{bronze_path}/customers.csv')
accounts_df = spark.read.option("header", "true").csv(f'{bronze_path}/accounts.csv')
transactions_df = spark.read.option("header", "true").csv(f'{bronze_path}/transactions.csv')
loans_df = spark.read.option("header", "true").csv(f'{bronze_path}/loans.csv')
loan_payments_df = spark.read.option("header", "true").csv(f'{bronze_path}/loan_payments.csv')

# -----
# Cleaning and Transformation

```

```
# -----
```

```
# Clean customers
```

```
customers_clean = customers_df.dropDuplicates(["customer_id"]).na.drop(subset=["customer_id"])

customers_clean = customers_clean.withColumn("customer_id", col("customer_id").cast("string")) \
    .withColumn("email", trim(col("email"))) \
    .withColumn("date_joined", to_date("date_joined", "yyyy-MM-dd"))
```

```
# Clean accounts
```

```
accounts_clean = accounts_df.dropDuplicates(["account_id"]) \
    .withColumn("account_id", col("account_id").cast("string")) \
    .withColumn("customer_id", col("customer_id").cast("string")) \
    .withColumn("balance", col("balance").cast(DoubleType()))
```

```
# Clean transactions
```

```
transactions_clean = transactions_df.dropDuplicates(["transaction_id"]) \
    .withColumn("transaction_id", col("transaction_id").cast("string")) \
    .withColumn("account_id", col("account_id").cast("string")) \
    .withColumn("amount", col("amount").cast(DoubleType()))
```

```
# Clean loans
```

```
loans_clean = loans_df.dropDuplicates(["loan_id"]) \
    .withColumn("loan_id", col("loan_id").cast("string")) \
    .withColumn("customer_id", col("customer_id").cast("string")) \
    .withColumn("loan_amount", col("loan_amount").cast(DoubleType()))
```

```
# Clean loan payments
```

```
loan_payments_clean = loan_payments_df.dropDuplicates(["payment_id"]) \
    .withColumn("payment_id", col("payment_id").cast("string")) \
    .withColumn("loan_id", col("loan_id").cast("string")) \
```

```

.withColumn("payment_amount", col("payment_amount").cast(DoubleType())))

# -----
# Write to Silver Layer
# -----


silver_path = "s3://nehaawsbc001/silver/"

customers_clean.write.mode("overwrite").parquet(f'{silver_path}/customers/')
accounts_clean.write.mode("overwrite").parquet(f'{silver_path}/accounts/')
transactions_clean.write.mode("overwrite").parquet(f'{silver_path}/transactions/')
loans_clean.write.mode("overwrite").parquet(f'{silver_path}/loans/')
loan_payments_clean.write.mode("overwrite").parquet(f'{silver_path}/loan_payments/')

job.commit()

```

The screenshot shows the AWS Glue Job Editor interface. On the left, there's a sidebar with navigation links like 'AWS Glue', 'ETL jobs', 'Data Catalog', 'Data Integration and ETL', and 'Legacy pages'. The main area is titled 'BC001_Silver_Transform' and shows the 'Job details' tab selected. It includes fields for 'Name' (BC001_Silver_Transform), 'Description - optional' (empty), 'IAM Role' (glue_role), 'Type' (Spark), and 'Glue version' (Glue 5.0 - Supports spark 3 & Scala 2, Python 3). The top of the page has tabs for 'Script', 'Job details', 'Runs', 'Data quality', 'Schedules', and 'Version Control'. There are also 'Actions' (Save, Run), 'Last modified on 2025-06-28, 7:26:05 p.m.', and browser-specific icons.

BC001_Silver_Transform

Last modified on 2025-06-28, 7:26:05 p.m.

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPU)	Worker type	Glue version
Succeeded	0	06/28/2025 19:26:19	06/28/2025 19:28:06	1 m 35 s	10 DPU	G.1X	5.0
Failed	0	06/28/2025 19:17:28	06/28/2025 19:18:53	1 m 17 s	10 DPU	G.1X	5.0

Run details

Job name	Start time (Local)	Glue version	Last modified on (Local)
BC001_Silver_Transform	06/28/2025 19:26:19	5.0	06/28/2025 19:28:06
Id	End time (Local)	Worker type	Log group name
jr_320b5dbb3a1ff0b2457b008b3b17c77f	06/28/2025 19:28:06	G.1X	/aws-glue/jobs
d93c4c4f58e0c78eda6f46f4ff781c8			
Run status	Start-up time	Max capacity	Number of workers
Succeeded	11 seconds	10 DPU	10
Retry attempt number	Execution time	Execution class	Timeout

BC001_Silver_Transform

Last modified on 2025-06-28, 7:26:05 p.m.

Run details	Input arguments (9)	Logs	Run insights	Metrics	Troubleshooting analysis - preview	Spark UI
Job name	Start time (Local)	Glue version	Last modified on (Local)			
BC001_Silver_Transform	06/28/2025 19:26:19	5.0	06/28/2025 19:28:06			
Id	End time (Local)	Worker type	Log group name			
jr_320b5dbb3a1ff0b2457b008b3b17c77f	06/28/2025 19:28:06	G.1X	/aws-glue/jobs			
d93c4c4f58e0c78eda6f46f4ff781c8						
Run status	Start-up time	Max capacity	Number of workers			
Succeeded	11 seconds	10 DPU	10			
Retry attempt number	Execution time	Execution class	Timeout			
Initial run	1 minute 35 seconds	Standard	480 minutes			
Trigger name	Security configuration	Cloudwatch logs	Usage profile			
-	-	• Output logs	-			
-	-	• Error logs	-			
Job run queuing						
False						

Step 4: Create a crawler and give database location.

Set crawler properties

Crawler details Info

Name Name can be up to 255 characters long. Some character set including control characters are prohibited.

Description - optional Descriptions can be up to 2048 characters long.

Tags - optional Use tags to organize and identify your resources.

Cancel **Next**

Edit data source

Data source Choose the source of data to be crawled.

Network connection - optional Optionally include a Network connection to use with this S3 target. Note that each crawler is limited to one Network connection so any other S3 targets will also use the same connection (or none, if left blank). **Add new connection**

S3 path Browse for or enter an existing S3 path. All folders and files contained in the S3 path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

Subsequent crawler runs This field is a global field that affects all S3 data sources.

- Crawl all sub-folders** Crawl all folders again with every subsequent crawl.
- Crawl new sub-folders only** Only Amazon S3 folders that were added since the last crawl will be crawled. If the schemas are compatible, new partitions will be added to existing tables.
- Crawl based on events** Rely on Amazon S3 events to control what folders to crawl.
- Sample only a subset of files**
- Exclude files matching pattern**

Parameters Recrawl all

Cancel **Previous** **Next**

The screenshot shows the AWS Glue 'Add crawler' wizard at Step 2: Choose data sources and classifiers. The sidebar navigation includes 'AWS Glue', 'Data Catalog', 'Data Integration and ETL', and 'Legacy pages'. The main panel displays the 'Choose data sources and classifiers' step, which is currently selected. It shows a table for 'Data sources (1)' with one entry: 's3://nehabc001project/' under 'Type' and 'Recrawl all' under 'Parameters'. A note indicates that the crawler will scan the S3 bucket for new or modified files. There is also a section for 'Custom classifiers - optional'.

The screenshot shows the AWS Glue 'Add crawler' wizard at Step 3: Configure security settings. The sidebar navigation includes 'AWS Glue', 'Data Catalog', 'Data Integration and ETL', and 'Legacy pages'. The main panel displays the 'Configure security settings' step, which is currently selected. It shows an 'IAM role' dropdown set to 'glue_service_role' with a 'Create new IAM role' button. A note states that only IAM roles created by the AWS Glue console can be used. Below this is a 'Lake Formation configuration - optional' section with a checkbox for 'Use Lake Formation credentials for crawling S3 data source'. A note explains that this allows the crawler to use Lake Formation credentials for crawling data from another account. There is also a section for 'Security configuration - optional'.

The screenshot shows the AWS Glue 'Add crawler' wizard at Step 4: Set output and scheduling. The left sidebar shows the navigation menu with 'Crawlers' selected. The main panel has a title 'Set output and scheduling'. It includes sections for 'Output configuration' (target database set to 'financial_db'), 'Table name prefix - optional' (empty), 'Maximum table threshold - optional' (set to 'Type a number greater than 0'), and 'Advanced options' (frequency set to 'On demand'). A note states: '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.' The bottom right corner shows copyright information: '© 2025, Amazon Web Services, Inc. or its affiliates.'

The screenshot shows the AWS Glue 'Add crawler' wizard at Step 5: Review and create. The left sidebar shows the navigation menu with 'Crawlers' selected. The main panel displays five steps: Step 1 (Set crawler properties), Step 2 (Choose data sources and classifiers), Step 3 (Configure security settings), Step 4 (Set output and scheduling), and Step 5 (Review and create). The 'Review and create' step is currently active. It contains four sections: 'Step 1: Set crawler properties' (with an 'Edit' button), 'Step 2: Choose data sources and classifiers' (with a table showing one S3 data source: 's3://nehabc001project/' under 'Data source' and 'Recrawl all' under 'Parameters'), 'Step 3: Configure security settings' (with an 'Edit' button), and 'Step 4: Set output and scheduling' (with an 'Edit' button). The bottom right corner shows copyright information: '© 2025, Amazon Web Services, Inc. or its affiliates.'

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Name	State	Last run	Last run time...	Log	Table changes f...
silver_crawler_ne...	Ready	Succeeded	June 28, 2025 at...	View log	5 created

silver_crawler_neha

Crawler properties

Name silver_crawler_neha	IAM role glue_role	Database silver_layer_db	State READY
Description -	Security configuration -	Lake Formation configuration -	Table prefix -
Maximum table threshold -			

Advanced settings

Crawler runs (1)

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
June 28, 2025 at 23:31:57	June 28, 2025 at 23:33:17	01 min 19 s	Completed	0.040	5 table changes, 0 partition changes

Step 5: Check tables under database

silver_layer_db

Database properties

Name	Description	Location	Created on (UTC)
silver_layer_db	-	-	June 28, 2025 at 23:12:12

Tables (5)

Name	Database	Location	Classification	Deprecated	View data	Data quality	Column stats...
accounts	silver_layer_db	s3://nehaawsc001	Parquet	-	Table data	View data quality	View statistics
customers	silver_layer_db	s3://nehaawsc001	Parquet	-	Table data	View data quality	View statistics
loan_payments	silver_layer_db	s3://nehaawsc001	Parquet	-	Table data	View data quality	View statistics
loans	silver_layer_db	s3://nehaawsc001	Parquet	-	Table data	View data quality	View statistics
transactions	silver_layer_db	s3://nehaawsc001	Parquet	-	Table data	View data quality	View statistics

silver_layer_db

Schema (5)

#	Column name	Data type	Partition key	Comment
1	account_id	string	-	-
2	customer_id	string	-	-
3	account_type	string	-	-
4	balance	double	-	-
5	open_date	string	-	-

The screenshot shows the AWS Glue Data Catalog interface. On the left, a sidebar navigation includes 'AWS Glue' (selected), 'Getting started', 'ETL jobs', 'Data Catalog tables', 'Data connections', 'Workflows (orchestration)', 'Zero-ETL integrations', 'Data Catalog' (expanded), 'Databases' (selected), 'Tables', 'Stream schema registries', 'Schemas', 'Connections', 'Crawlers', 'Classifiers', 'Catalog settings', 'Data Integration and ETL', and 'Legacy pages'. Below this are 'What's New' and 'Documentation'. At the bottom are 'CloudShell' and 'Feedback' buttons.

The main content area displays the 'customers' table under 'silver_layer_db'. It shows the table's description, last updated (June 28, 2025 at 23:33:16), and connection information (S3://nehaawbsc001/silver/customers). A 'No statistics' message is present. Below this is an 'Advanced properties' section and a 'Schema' tab. The 'Schema' tab lists five columns: customer_id (string), customer_name (string), email (string), address (string), and date_joined (date). There are 'Edit schema as JSON' and 'Edit schema' buttons at the top of the schema table.

This screenshot is identical to the one above, showing the 'loan_payments' table in the 'silver_layer_db' database. The table structure is identical, with five columns: payment_id (string), loan_id (string), customer_id (string), payment_amount (double), and payment_date (string).

The screenshot shows the AWS Glue Data Catalog interface. On the left, a sidebar navigation includes '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', and 'Legacy pages'. A 'What's New' link is also present. The main content area displays the 'loans' table details. The 'Description' section shows a note about last update on June 28, 2025. The 'Connection' section is empty. The 'Advanced properties' section is collapsed. The 'Schema' tab is selected, showing a table with 6 columns: loan_id (string), customer_id (string), loan_amount (double), status (string), start_date (string), and term_months (string). Each column has a 'Partition key' and 'Comment' field set to '-'.

This screenshot shows the AWS Glue Data Catalog interface for the 'transactions' table. The sidebar navigation is identical to the previous screenshot. The main content area displays the 'transactions' table details. The 'Description' section shows a note about last update on June 28, 2025. The 'Connection' section is empty. The 'Advanced properties' section is collapsed. The 'Schema' tab is selected, showing a table with 6 columns: transaction_id (string), customer_id (string), account_id (string), amount (double), transaction_type (string), and timestamp (string). Each column has a 'Partition key' and 'Comment' field set to '-'.

The screenshot shows the AWS S3 console with the path `Buckets > nehaawbsc001 > silver/`. The left sidebar has sections for General purpose buckets, Storage Lens, and IAM Access Analyzer for S3. The main area displays 5 objects under the heading **Objects (5)**. The objects are:

Name	Type	Last modified	Size	Storage class
<code>accounts/</code>	Folder	-	-	-
<code>customers/</code>	Folder	-	-	-
<code>loan_payments/</code>	Folder	-	-	-
<code>loans/</code>	Folder	-	-	-
<code>transactions/</code>	Folder	-	-	-

Cleaned parquet files:



part-00000-356c62ea part-00000-f76c845b part-00000-8eb28f07 part-00000-da79e97c part-00000-0989f3a5
 -65ba-446c-8e22-cb2-bb48-4ae6-aa76-a59-1ef6-4caa-b5f7-2877-dbbf-415f-aaa9-447c-edef-46f7-abc7-38a6

Data Warehouse & SCD - Gold Layer (Business-Ready Analytics)

The Gold Layer transforms curated data from the Silver Layer into analytics-ready datasets that support business use cases such as: Loan performance summaries per customer. These datasets are stored in optimized formats (Parquet/Delta) for use in Athena.

Step 1: Create Glue Database - To catalog the output of Gold-layer Glue jobs for query access in Athena and visualization tools.

The screenshot shows the AWS Glue Console interface. On the left, there's a navigation sidebar with options like AWS Glue, Data Catalog, Data Integration and ETL, and Legacy pages. The main content area is titled 'Databases (2)' and shows a table with the following data:

Name	Description	Location URI	Created on (UTC)
gold_layer_db	-	-	June 28, 2025 at 23:57:11
silver_layer_db	-	-	June 28, 2025 at 23:12:12

Step 2: Gold Layer Glue Job –

1. Join loans with customers on customer_id.
2. Aggregate per customer: total_loans: count of loans, avg_loan_amount: average loan value

The screenshot shows two consecutive pages from the AWS Glue Studio interface.

Page 1: Script Editor

- Job Name:** BC001_Glue_Gold_LoanSummary
- Script Tab:** The script code is displayed in a code editor window. It imports necessary libraries like sys, SparkContext, SparkSession, and various functions from pyspark.sql and awsglue.context. It also defines a glueContext, sets args, and initializes a Job object.
- Python:** Python Ln 47, Col 36 | Errors: 0 | Warnings: 0

Page 2: Job Details

- Job Name:** BC001_Glue_Gold_LoanSummary
- Basic Properties:**
 - Name:** BC001_Glue_Gold_LoanSummary
 - Description - optional:** (Empty text area)
 - IAM Role:** glue_role
 - Type:** Spark
 - Glue version:** Glue 5.0 - Current

The screenshot shows the AWS Glue Studio interface. At the top, there are two tabs: "nehaawscb001 - S3 bucket" and "Job details - Editor - AWS Glue". The main content area displays a green success message: "Successfully updated job BC001_Glue_Gold_LoanSummary. To run the job choose the Run Job button." Below this, the job name "BC001_Glue_Gold_LoanSummary" is shown, along with a timestamp "Last modified on 2025-06-28, 7:59:33 p.m." and three buttons: "Actions", "Save", and "Run". A navigation bar at the bottom includes "Script", "Job details" (which is selected), "Runs", "Data quality", "Schedules", and "Version Control". The "Basic properties" section contains fields for "Name" (BC001_Glue_Gold_LoanSummary), "Description" (empty), "IAM Role" (glue_role), and "Type" (ETL job). The "Description" field has a note: "Descriptions can be up to 2048 characters long." The "IAM Role" field has a note: "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." The "Type" field has a note: "The type of ETL job. This is set automatically based on the types of data sources you have selected."

This screenshot is identical to the one above, showing the same job details page. The main difference is the message at the top: "Successfully started job BC001_Glue_Gold_LoanSummary. Navigate to Run details for more details." The rest of the interface, including the navigation bar, basic properties, and notes, remains the same.

BC001_Glue_Gold_LoanSummary

Last modified on 2025-06-28, 7:59:33 p.m. Actions Save Run

Script Job details Runs Data quality Schedules Version Control

Job runs (1/1) Info

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Running	0	06/28/2025 19:59:40	-	0 s	10 DPUs	G.1X	5.0

Run details

Job name	Start time (Local)	Glue version	Last modified on (Local)
BC001_Glue_Gold_LoanSummary	06/28/2025 19:59:40	5.0	06/28/2025 19:59:44
Id	End time (Local)	Worker type	Log group name
jr_07d1b271c4031a946917a4679c998916d5c4ad5c - dff3c5eb2291b0cb8e6c195	-	G.1X	/aws-glue/jobs
Run status	Start-up time	Max capacity	Number of workers
Running	0	10 DPUs	10

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

BC001_Glue_Gold_LoanSummary

Last modified on 2025-06-28, 8:05:59 p.m. Actions Save Run

Script Job details Runs Data quality Schedules Version Control

Script Info

```

1 import sys
2 from pyspark.context import SparkContext
3 from pyspark.sql import SparkSession
4 from pyspark.sql.functions import col, count, avg
5 from awsglue.context import GlueContext
6 from awsglue.utils import getResolvedOptions
7 from awsglue.job import Job
8
9 # Job args
10 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
11
12 # Glue context setup
13 sc = SparkContext()
14 glueContext = GlueContext(sc)
15 spark = glueContext.spark_session
16 job = Job(glueContext)
17 job.init(args['JOB_NAME'], args)
18
19 #

```

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

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

Code:

```

import sys

from pyspark.context import SparkContext

from pyspark.sql import SparkSession

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

from awsglue.context import GlueContext

```

```
from awsglue.utils import getResolvedOptions
from awsglue.job import Job

# Job args
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

# Glue context setup
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# -----
# Load Silver Layer Data
# -----


silver_path = "s3://nehaawsbc001/silver/"

customers = spark.read.parquet(f'{silver_path}/customers/')
loans = spark.read.parquet(f'{silver_path}/loans/')

# -----
# Join and Aggregation
# -----


# Join on customer_id
joined_df = loans.join(customers, on="customer_id", how="inner")

# Use customer_name directly
```

```
summary_df = joined_df.groupBy("customer_id", "customer_name") \
```

```
.agg(
```

```
    count("loan_id").alias("total_loans"),
```

```
    avg("loan_amount").alias("avg_loan_amount")
```

```
)
```

```
# -----
```

```
# Write to Gold Layer
```

```
# -----
```

```
gold_path = "s3://nehaawbsc001/gold/customer_loan_summary/"
```

```
summary_df.write.mode("overwrite").parquet(gold_path)
```

```
job.commit()
```

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Succeeded	0	06/28/2025 20:06:00	06/28/2025 20:07:43	1 m 13 s	10 DPUs	G.1X	5.0
Failed	0	06/28/2025 19:59:40	06/28/2025 20:01:17	1 m 29 s	10 DPUs	G.1X	5.0

Run details

Job name	Start time (Local)	Glue version	Last modified on (Local)
BC001_Glue_Gold_LoanSummary	06/28/2025 20:06:00	5.0	06/28/2025 20:07:43
Id	End time (Local)	Worker type	Log group name
jr_39b1298a632458e883765ac13c7588ea44fa826896	06/28/2025 20:07:43	G.1X	/aws-glue/jobs
Run status	Start-up time	Max capacity	Number of workers
Succeeded	29 seconds	10 DPUs	10
Retry attempt number	Execution time	Execution class	Timeout

Step 3: Crawler for Gold Layer

Set crawler properties

Crawler details Info

Name gold_customer_summary_crawler

Name can be up to 255 characters long. Some character set including control characters are prohibited.

Description - optional

Enter a description

Descriptions can be up to 2048 characters long.

Tags - optional

Use tags to organize and identify your resources.

Step 1 Set crawler properties **Step 2** Choose data sources and classifiers **Step 3** Configure security settings **Step 4** Set output and scheduling **Step 5** Review and create

AWS Glue

- Getting started
- ETL jobs
 - Visual ETL
 - Notebooks
 - Job run monitoring
- Data Catalog tables
- Data connections
- Workflows (orchestration)
- Zero-ETL integrations New

Crawlers

- Classifiers
- Catalog settings

Data Integration and ETL

Legacy pages

What's New [?] Documentation [?]

CloudShell Feedback

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

Add data source

Data source Choose the source of data to be crawled.

S3

Network connection - optional

Optionally include a Network connection to use with this S3 target. Note that each crawler is limited to one Network connection so any other S3 targets will also use the same connection (or none, if left blank).

Clear selection **Add new connection**

Location of S3 data

In this account

S3 path

Browse for or enter an existing S3 path.

bc001/gold/customer_loan_summary/ **View** **Browse S3**

All folders and files contained in the S3 path are crawled. For example, type s3://MyBucket/ MyFolder/ to crawl all objects in MyFolder within MyBucket.

Subsequent crawler runs

This field is a global field that affects all S3 data sources.

Crawl all sub-folders

Crawl all folders again with every subsequent crawl.

Crawl new sub-folders only

Only Amazon S3 folders that were added since the last crawl will be crawled. If the schemas are compatible, new partitions will be added to existing tables.

Crawl based on events

Rely on Amazon S3 events to control what folders to crawl.

Cancel **Add an S3 data source**

Step 1 Set crawler properties **Step 2** Choose data sources and classifiers **Step 3** Configure security settings **Step 4** Set output and scheduling **Step 5** Review and create

AWS Glue

- Getting started
- ETL jobs
 - Visual ETL
 - Notebooks
 - Job run monitoring
- Data Catalog tables
- Data connections
- Workflows (orchestration)
- Zero-ETL integrations New

Crawlers

- Classifiers
- Catalog settings

Data Integration and ETL

Legacy pages

What's New [?] Documentation [?]

CloudShell Feedback

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

The screenshot shows the AWS Glue 'Add crawler' wizard at Step 2: Choose data sources and classifiers. The sidebar navigation includes 'AWS Glue', 'Data Catalog', 'Data Integration and ETL', and 'Legacy pages'. The main panel displays the 'Choose data sources and classifiers' step, which is currently selected. It shows a 'Data source configuration' section where the user has chosen 'Not yet' (Select one or more data sources to be crawled). A table lists a single data source: 'S3' with type 's3://nehaawsc001/gold/customer...' and parameters 'Recrawl all'. Below this is a 'Custom classifiers - optional' section. Navigation buttons 'Cancel', 'Previous', and 'Next' are visible.

The screenshot shows the AWS Glue 'Add crawler' wizard at Step 3: Configure security settings. The sidebar navigation is identical to the previous screenshot. The main panel displays the 'Configure security settings' step, which is currently selected. It shows an 'IAM role' section with 'glue_role' selected, and options to 'Create new IAM role' or 'Update chosen IAM role'. A note states that only IAM roles created by the AWS Glue console can be updated. Below this is a 'Lake Formation configuration - optional' section, which is currently collapsed. A 'Security configuration - optional' section is also present. Navigation buttons 'Cancel', 'Previous', and 'Next' are visible.

The screenshot shows the AWS Glue 'Add crawler' wizard at Step 4: Set output and scheduling. The sidebar on the left lists various AWS Glue services. The main panel has a title 'Set output and scheduling'. It contains sections for 'Output configuration' (target database set to 'gold_layer_db'), 'Table name prefix - optional' (empty), 'Maximum table threshold - optional' (set to 0), and 'Advanced options' (frequency set to 'On demand'). A note states: '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.' The bottom right of the panel shows the copyright notice '© 2025, Amazon Web Services, Inc. or its affiliates.'

This screenshot continues the AWS Glue 'Add crawler' wizard. After Step 4, it moves to Step 5: Review and create. The sidebar remains the same. The main panel now shows 'Step 4: Set output and scheduling' (with the same configurations) and 'Step 5: Review and create'. The bottom right shows 'Cancel', 'Previous', and a prominent orange 'Create crawler' button.

One crawler successfully created
The following crawler is now created: "gold_customer_summary_crawler"

gold_customer_summary_crawler

Crawler properties

Name gold_customer_summary_crawler	IAM role glue_role	Database gold_layer_db	State READY
Description	Security configuration	Lake Formation configuration	Table prefix
Maximum table threshold			

Advanced settings

Crawler runs | Schedule | Data sources | Classifiers | Tags

Crawler runs (0)
The list of crawler runs for this crawler.

Filter data | Start time (UTC) | End time (UTC) | Current/last duration | Status | DPU hours | Table changes

Last updated (UTC) June 29, 2025 at 00:11:57 | Run crawler | Edit | Delete

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
June 29, 2025 at 00:12:04	-	47 s	Running	-	-

Filter data | Start time (UTC) | End time (UTC) | Current/last duration | Status | DPU hours | Table changes

Last updated (UTC) June 29, 2025 at 00:11:57 | Run crawler | Edit | Delete

Screenshot of the AWS Glue Console showing the configuration of the 'gold_customer_summary_crawler'.

Crawler properties:

- Name: gold_customer_summary_crawler
- IAM role: glue_role
- Database: gold_layer_db
- Description: -
- Security configuration: -
- Lake Formation configuration: -
- Table prefix: -
- State: READY
- Maximum table threshold: -

Advanced settings:

Crawler runs: (1) Last updated (UTC): June 29, 2025 at 00:11:57

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
June 29, 2025 at 00:12:04	June 29, 2025 at 00:13:39	01 min 34 s	Completed	-	-

Screenshot of the AWS Glue Console showing the configuration of the database 'gold_layer_db'.

Announcing new optimization features for Apache Iceberg tables: Optimize storage for Apache Iceberg tables with automatic snapshot retention and orphan file deletion. Learn more.

Database properties:

- Name: gold_layer_db
- Description: -
- Location: -
- Created on (UTC): June 28, 2025 at 23:57:11

Tables (1): Last updated (UTC): June 29, 2025 at 00:13:51

Name	Database	Location	Classificat...	Deprecated	View data	Data quality	Column stati...
customer_loan_sur	gold_layer_db	s3://nehaawsc00	Parquet	-	Table data	View data quality	View statistics

AWS Glue > Tables > customer_loan_summary

Database: gold_layer_db

Description: -

Last updated: June 29, 2025 at 00:13:58

Location: s3://nehaawbc001/gold/customer_loan_summary/

Connection: -

Column statistics: No statistics

Advanced properties:

Schema | Partitions | Indexes | Column statistics - new

Schema (4)

#	Column name	Data type	Partition key	Comment
1	customer_id	string	-	-
2	customer_name	string	-	-
3	total_loans	bigint	-	-
4	avg_loan_amount	double	-	-

Step 4: Athena Table Verification

Query result encryption

Query result location: s3://nehaawbc001/athena_gold_result/

Encrypt query results: -

Expected bucket owner: -

Assign bucket owner full control over query results: Turned off

The screenshot shows the AWS Athena Query Editor interface. On the left, the 'Data' sidebar is open, displaying the 'Data source' set to 'AwsDataCatalog', 'Catalog' set to 'None', and 'Database' set to 'gold_layer_db'. Under 'Tables and views', there is one table named 'customer_loan_summary' and zero views. The main area shows 'Query 1' with the SQL query: `SELECT * FROM customer_loan_summary LIMIT 10;`. Below the query, it says 'Completed' with a green checkmark. At the bottom, performance metrics are listed: 'Time in queue: 108 ms', 'Run time: 563 ms', and 'Data scanned: 2.48 KB'. There are also buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'.

- Query: `SELECT * FROM gold_layer_db.customer_loan_summary LIMIT 10;`

The screenshot shows the results of the query execution. The results are titled 'Results (10)' and contain 10 rows of data. The columns are: #, customer_id, customer_name, total_loans, and avg_loan_amount. The data is as follows:

#	customer_id	customer_name	total_loans	avg_loan_amount
1	C55B20CFB	Justin Holt	1	1317.04
2	C151B855E	Benjamin Moore	1	5765.19
3	C190248A3	Jorge Dean	2	19445.010000000002
4	C75B35276	Rachel Martinez	2	33056.990000000005
5	C7F668206	Karen Williams	1	48932.02
6	CA49664B1	Beverly Landry	3	28223.726666666666
7	C0F93DD54	Timothy Smith	1	1064.87
8	C8FD55A6A	Brittney Yates	1	2056.34
9	C9E16D019	Katherine Todd	1	49690.39
10	CA978B71C	Eric Robertson	1	16021.47

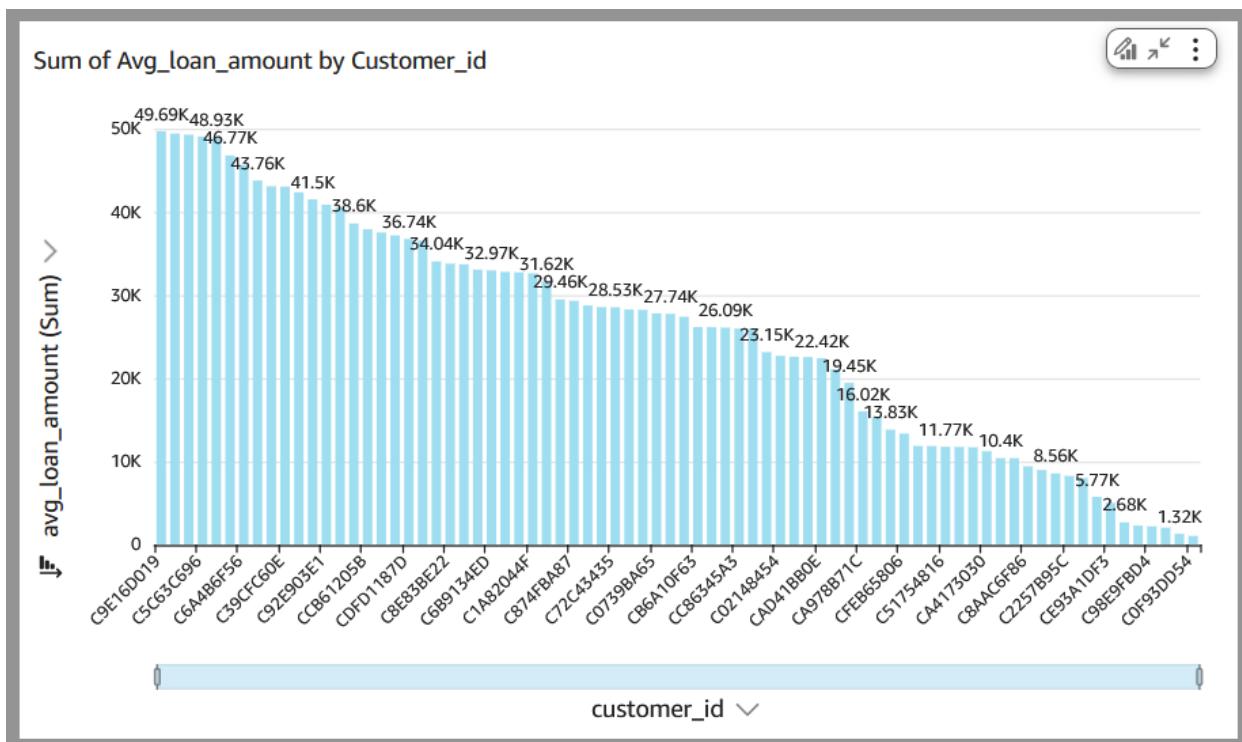
Analytics & Business Intelligence - Logistic Regression – Predict Loan Default Risk Per Customer

Objective

Train a logistic regression model using PySpark to predict the probability that a customer will default on a loan, based on their historical loan and payment behaviour. This output helps analysts identify high-risk customers for credit monitoring, risk control, or pre-emptive communication.

customer_id	label	default_probability	prediction
C72EB206A	0	0.3856081230660465	0
C9E160019	1	0.2536106597400225	0
C39E86242	0	0.44794535961658	0
C0890C1F1	0	0.16031074104795384	0
C0F93D054	0	0.2325876299162022	0
C75835276	1	0.957323073011247	1
C1E9FB404	1	0.2905051436248176	0
C72C43435	1	0.3340064338097747	0
CB180F247	1	0.22821657560190123	0
CD17EA9F2	0	0.19250668151235828	0
C900E58DF	1	0.453131568260640506	0
CDFD1187D	0	0.2207491750033458	0
C92E903E1	1	0.8614353065740894	1
C3E304FB0	1	0.7031916868321123	1
C0739BA65	0	0.5168341675505133	1

Dashboard in Quicksight Arena with Gold layer output



Databricks code:

<https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1049562611814646/3051986115664030/5718004312911206/latest.html>