Shaikh Ateeb Ahmed
08-09-2025

Serverless Data Pipeline

1) Create the Orders DynamoDB table

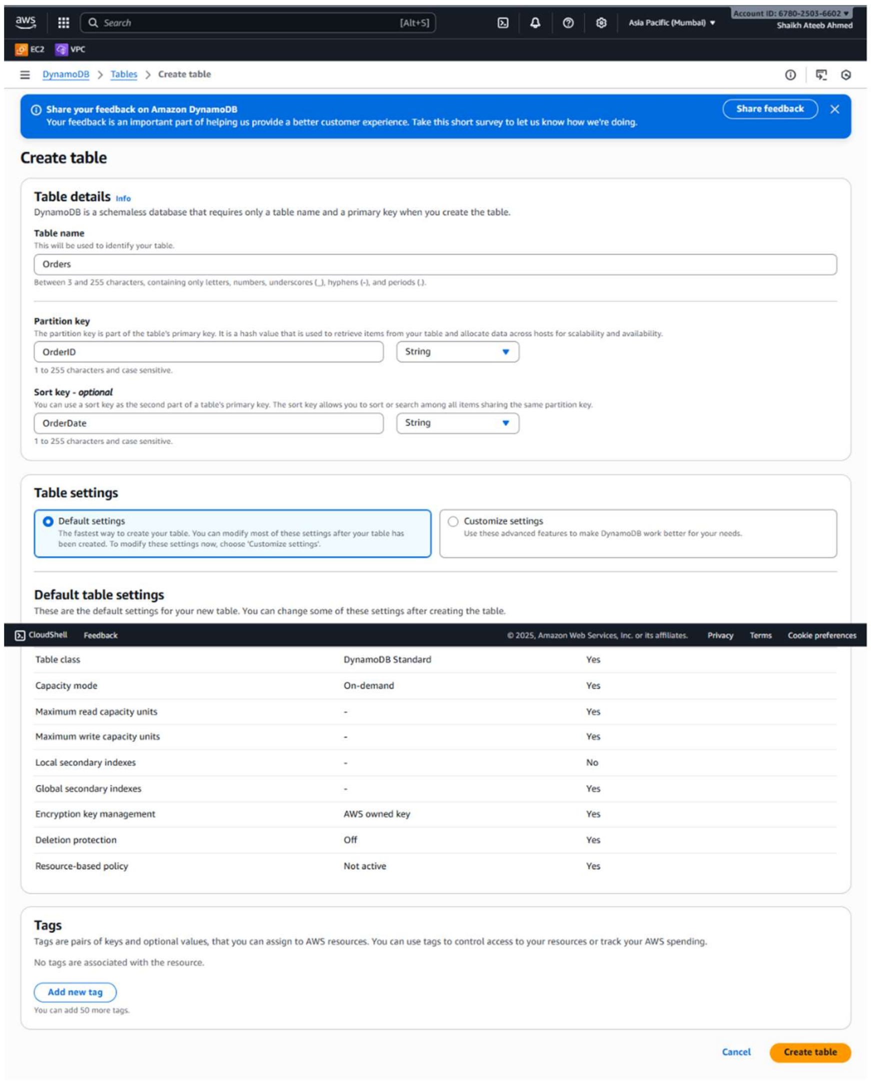
1. Sign in to the AWS Management Console
2. Open **Services → DynamoDB**.
3. Click **Create table**.
4. Under **Table name**, enter Orders.
5. Under **Primary key**:

Partition key: OrderID → select **String**.

Click **Add sort key** and enter **Sort key**: OrderDate → **String**.

6. Under **Table settings** → **Capacity mode**, select **On-demand (Pay per request)**.

Shaikh Ateeb Ahmed
08-09-2025



DynamoDB > Tables > Create table

Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

Create table

Table details
DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.
Orders
Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.
OrderID — String
1 to 255 characters and case sensitive.

Sort key - *optional*
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.
OrderDate — String
1 to 255 characters and case sensitive.

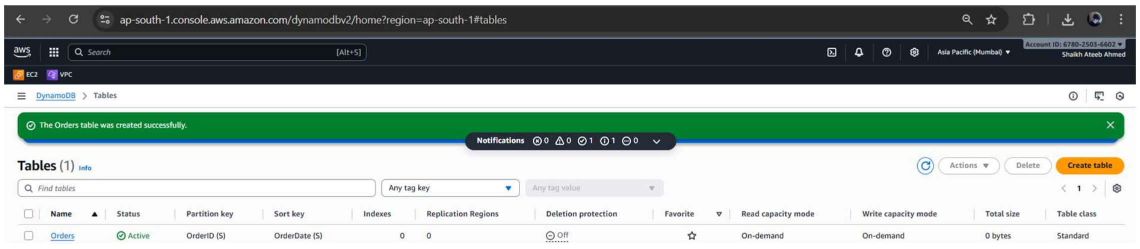
Table settings
[✓ Default settings] The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.
Customize settings: Use these advanced features to make DynamoDB work better for your needs.

Default table settings
These are the default settings for your new table. You can change some of these settings after creating the table.

Table class	DynamoDB Standard	Yes
Capacity mode	On-demand	Yes
Maximum read capacity units	-	Yes
Maximum write capacity units	-	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	AWS owned key	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

Tags
Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.
No tags are associated with the resource.
You can add 50 more tags.

7. Click **Create table**.



ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#tables

DynamoDB > Tables

The Orders table was created successfully.

Notifications ⊗0 ⚠0 ✓1 ⓘ1 ⊖0

Tables (1)

	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode	Write capacity mode	Total size	Table class
☐	Orders	✓ Active	OrderID (S)	OrderDate (S)	0	0	⊖ Off	☆	On-demand	On-demand	0 bytes	Standard

Step 2: Insert Sample Data

1) Quick note before you start

Your table has keys: OrderID (PK, String) and OrderDate (SK, String). **Every item must include both**.

Using ISO-8601 datetimes (e.g. 2025-09-01T09:10:00Z) for OrderDate so lexicographic sorting works for ranges.

2) Sample dataset (15 orders)

```
{
  "OrderID": { "S": "O1001" },
  "OrderDate": { "S": "2025-09-01T09:10:00Z" },
  "Customer": { "S": "Alice" },
  "Amount": { "N": "250" },
  "Status": { "S": "Shipped" }
}
{
  "OrderID": { "S": "O1002" },
  "OrderDate": { "S": "2025-09-01T10:24:00Z" },
  "Customer": { "S": "Bob" },
  "Amount": { "N": "400" },
  "Status": { "S": "Pending" }
}
{
  "OrderID": { "S": "O1003" },
  "OrderDate": { "S": "2025-09-02T11:30:00Z" },
  "Customer": { "S": "Charlie" },
  "Amount": { "N": "125" },
  "Status": { "S": "Delivered" }
}
{
  "OrderID": { "S": "O1004" },
  "OrderDate": { "S": "2025-09-03T14:05:00Z" },
  "Customer": { "S": "Diana" },
  "Amount": { "N": "78.5" },

  "Status": { "S": "Processing" }

}

{

  "OrderID": { "S": "O1005" },

  "OrderDate": { "S": "2025-09-03T16:50:00Z" },

  "Customer": { "S": "Eve" },

  "Amount": { "N": "560" },

  "Status": { "S": "Shipped" }

}

{

  "OrderID": { "S": "O1006" },

  "OrderDate": { "S": "2025-09-04T08:20:00Z" },

  "Customer": { "S": "Frank" },

  "Amount": { "N": "30" },

  "Status": { "S": "Cancelled" }

}

{

  "OrderID": { "S": "O1007" },

  "OrderDate": { "S": "2025-09-04T09:15:00Z" },

  "Customer": { "S": "Grace" },

  "Amount": { "N": "210" },

  "Status": { "S": "Pending" }

}

{

  "OrderID": { "S": "O1008" },

  "OrderDate": { "S": "2025-09-05T12:00:00Z" },

  "Customer": { "S": "Heidi" },
  "Amount": { "N": "999.99" },
  "Status": { "S": "Shipped" }
}
{
  "OrderID": { "S": "O1009" },
  "OrderDate": { "S": "2025-09-05T13:45:00Z" },
  "Customer": { "S": "Ivan" },
  "Amount": { "N": "150" },
  "Status": { "S": "Delivered" }
}
{
  "OrderID": { "S": "O1010" },
  "OrderDate": { "S": "2025-09-06T15:00:00Z" },
  "Customer": { "S": "Judy" },
  "Amount": { "N": "49.99" },
  "Status": { "S": "Processing" }
}
{
  "OrderID": { "S": "O1011" },
  "OrderDate": { "S": "2025-09-06T16:30:00Z" },
  "Customer": { "S": "Ken" },
  "Amount": { "N": "320" },
  "Status": { "S": "Shipped" }
}
{
  "OrderID": { "S": "O1012" },
  "OrderDate": { "S": "2025-09-07T10:00:00Z" },
  "Customer": { "S": "Leo" },
  "Amount": { "N": "215.5" },

  "Status": { "S": "Returned" }

}

{

  "OrderID": { "S": "O1013" },

  "OrderDate": { "S": "2025-09-07T11:11:00Z" },

  "Customer": { "S": "Mallory" },

  "Amount": { "N": "700" },

  "Status": { "S": "Pending" }

}

{

  "OrderID": { "S": "O1014" },

  "OrderDate": { "S": "2025-09-08T18:00:00Z" },

  "Customer": { "S": "Niaj" },

  "Amount": { "N": "1200" },

  "Status": { "S": "Delivered" }

}

{

  "OrderID": { "S": "O1015" },

  "OrderDate": { "S": "2025-09-09T09:00:00Z" },

  "Customer": { "S": "Olivia" },

  "Amount": { "N": "15.75" },

  "Status": { "S": "Processing" }

}
```

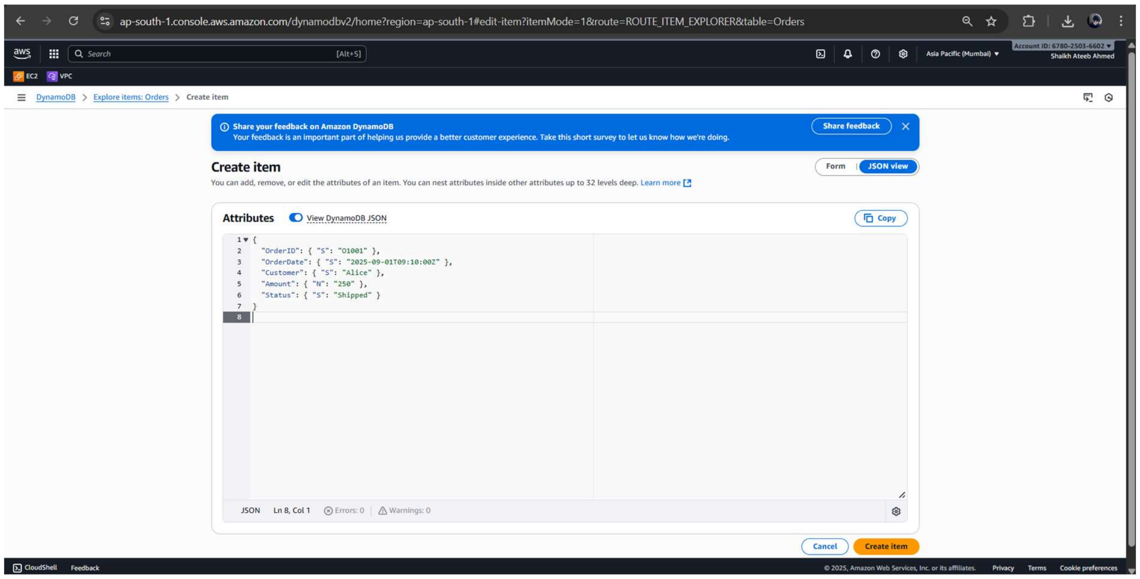
3) Manual insert via AWS Console

1. Console → Services → **DynamoDB** → **Tables** → click Orders.
2. Click **Explore items** (or **Items**), then **Create item**.
3. Switch to **JSON** view (easier for full items).

4. Paste one item from the dataset (document format, not typed AttributeValue).

   Example:

   ```
   {
     "OrderID": { "S": "O1001" },
     "OrderDate": { "S": "2025-09-01T09:10:00Z" },
     "Customer": { "S": "Alice" },
     "Amount": { "N": "250" },
     "Status": { "S": "Shipped" }
   }
   ```



ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#edit-item?itemMode=1&route=ROUTE_ITEM_EXPLORER&table=Orders

DynamoDB > Explore items: Orders > Create item

Share your feedback on Amazon DynamoDB
Your feedback is an important part of helping us provide a better customer experience. Take this short survey to let us know how we're doing.

Create item
You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more

Attributes — View DynamoDB JSON

```
{
  "OrderID": { "S": "O1001" },
  "OrderDate": { "S": "2025-09-01T09:10:00Z" },
  "Customer": { "S": "Alice" },
  "Amount": { "N": "250" },
  "Status": { "S": "Shipped" }
}
```

JSON Ln 8, Col 1 Errors: 0 Warnings: 0

5. Click **Create**. Repeat for other items.

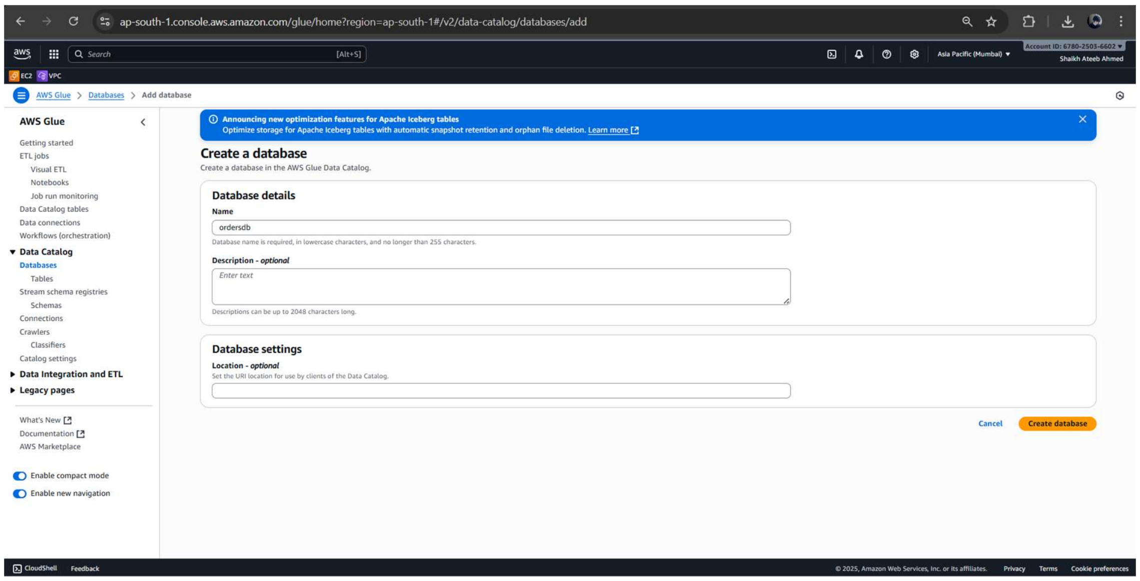
Console is manual but handy for quick checks.

Step 3: Create an AWS Glue Crawler for the Orders DynamoDB table

1) Create Glue Database (OrdersDB)

1. Open **AWS Console → AWS Glue → Data Catalog → Databases**.
2. Click **Add database**.
3. Name: ordersdb. (Optionally add description and location).

Shaikh Ateeb Ahmed
08-09-2025


ap-south-1.console.aws.amazon.com/glue/home?region=ap-south-1#/v2/data-catalog/databases/add

AWS Glue > Databases > Add database

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

Create a database
Create a database in the AWS Glue Data Catalog.

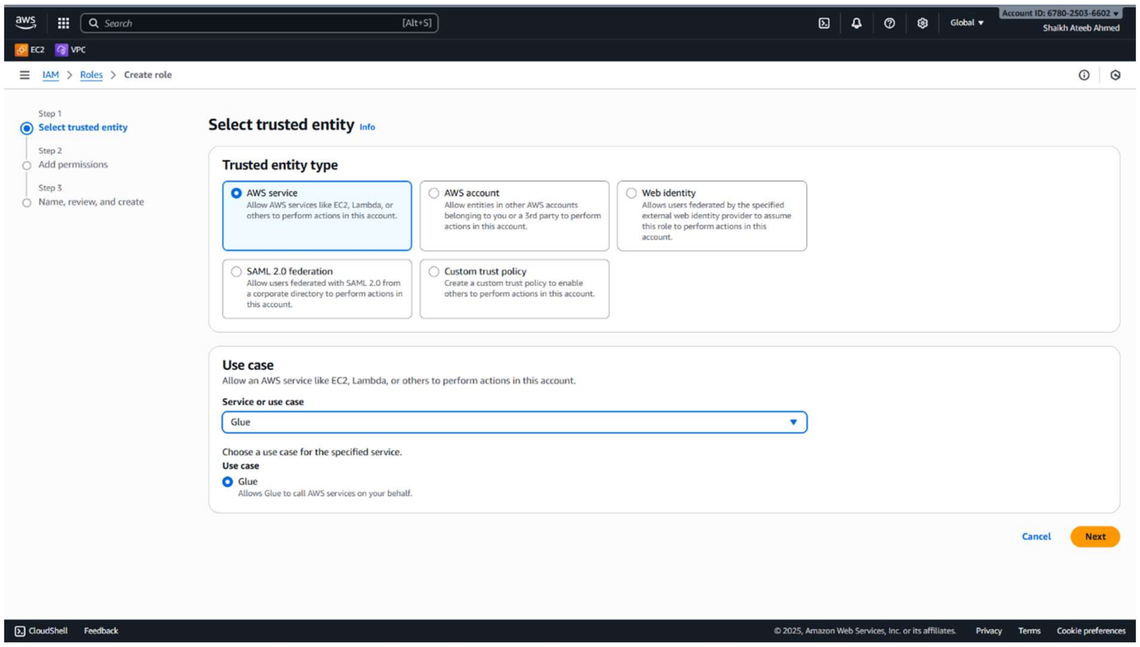
Database details
Name: ordersdb
Database name is required, in lowercase characters, and no longer than 255 characters.
Description - *optional*
Descriptions can be up to 2048 characters long.

Database settings
Location - *optional*
Set the URI location for use by clients of the Data Catalog.

4. Click **Create**.

2) Create IAM Role for Glue

1. Open **IAM → Roles → Create role**.
2. Select **Glue** as the trusted service (choose **Glue** so the trust relationship glue.amazonaws.com is set).


IAM > Roles > Create role

Select trusted entity

Trusted entity type
- [✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

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.

3. Attach policies:

AmazonDynamoDBReadOnlyAccess (or a custom restricted DynamoDB policy — see example below).

AmazonS3ReadOnlyAccess *or* a custom S3 policy granting access to a specific S3 temp bucket if you use S3.

AWSGlueServiceRole or any AWS-managed Glue service policy if shown.

AmazonS3FullAccess

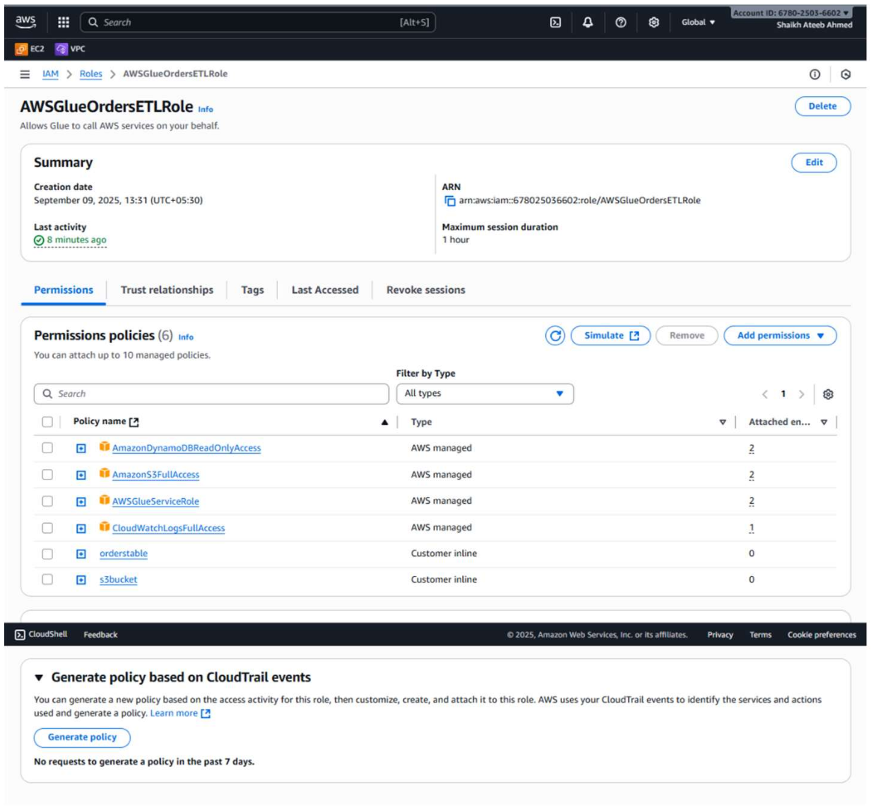
DynamoDB read: DescribeTable, Scan, GetItem, Query for the Orders table ARN.

S3 access to your bucket: GetObject, PutObject, ListBucket for arn:aws:s3:::my-orders-analytics01.

4. Give the role a name such as AWSGlueOrdersRole and finish.

Minimal custom trust policy (if using CLI):

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{"Service":"glue.amazonaws.com"},
      "Action":"sts:AssumeRole"
    }
  ]
}
```



AWSGlueOrdersETLRole

Allows Glue to call AWS services on your behalf.

Summary

Creation date: September 09, 2025, 13:31 (UTC+05:30)

ARN: arn:aws:iam::678025036602:role/AWSGlueOrdersETLRole

Last activity: 8 minutes ago

Maximum session duration: 1 hour

Permissions policies (6)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached en...
AmazonDynamoDBReadOnlyAccess	AWS managed	2
AmazonS3FullAccess	AWS managed	2
AWSGlueServiceRole	AWS managed	2
CloudWatchLogsFullAccess	AWS managed	1
orderstable	Customer inline	0
s3bucket	Customer inline	0

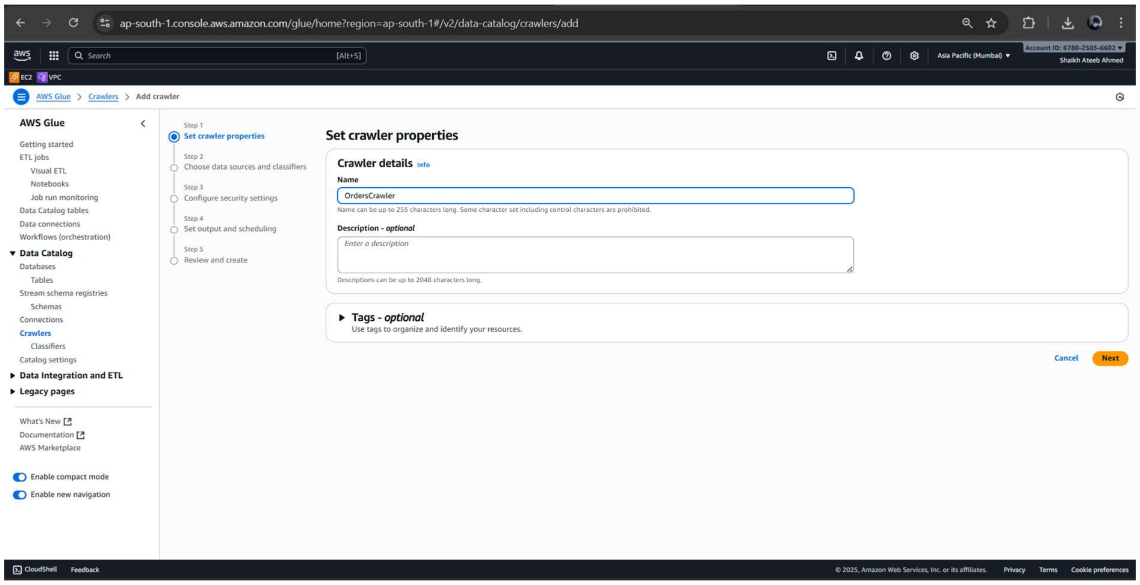
Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

No requests to generate a policy in the past 7 days.

3) Create the Crawler

1. Open **AWS Console → AWS Glue → Crawlers**.
2. Click **Add crawler** (or **Create crawler**).
3. **Name**: OrdersCrawler → Next.



ap-south-1.console.aws.amazon.com/glue/home?region=ap-south-1#/v2/data-catalog/crawlers/add

Set crawler properties

Crawler details

Name: OrdersCrawler

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.

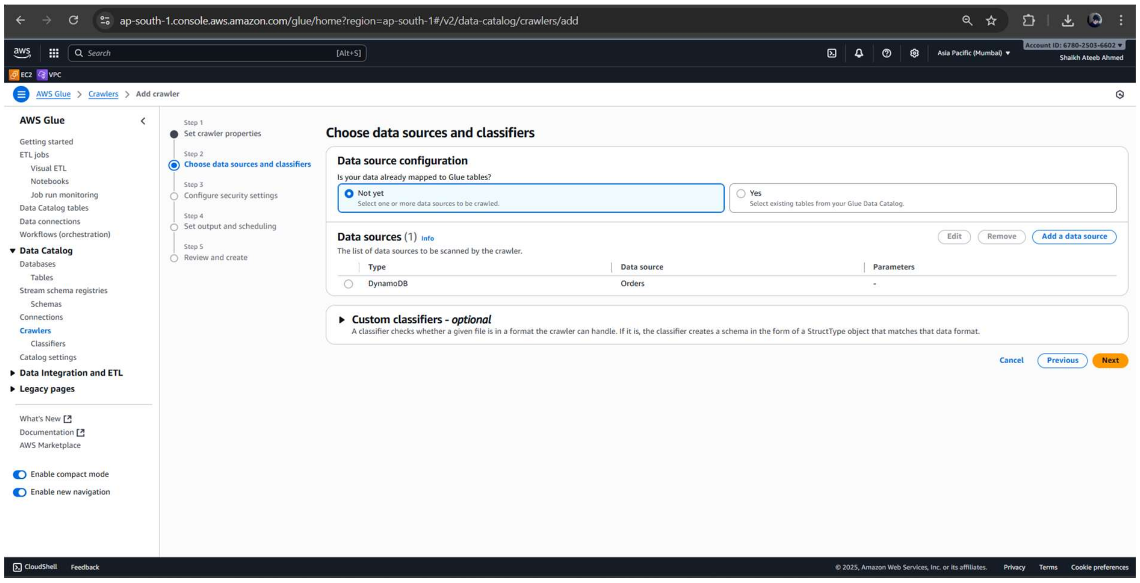
Tags - *optional*: Use tags to organize and identify your resources.

AWS Glue > Crawlers > Add crawler
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
Cancel
Next

4. **Data source**: Choose **Add a data store** → Select **DynamoDB**.

Select the table Orders from the list (region must match).

Click **Add** → Next.


ap-south-1.console.aws.amazon.com/glue/home?region=ap-south-1#/v2/data-catalog/crawlers/add
AWS Glue > Crawlers > Add crawler
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
Cancel
Previous
Next

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

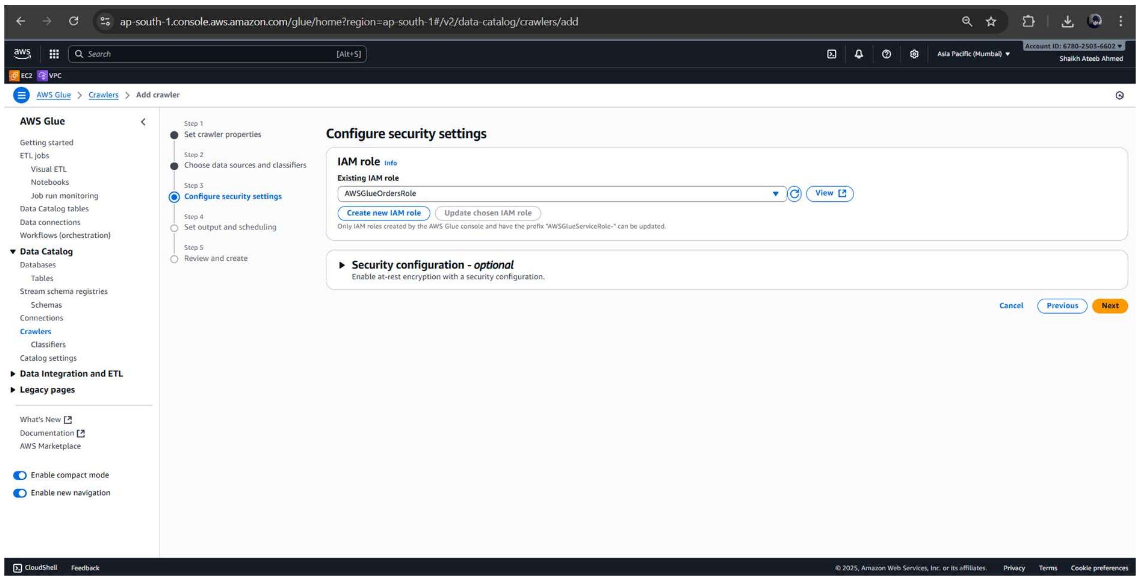
The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
DynamoDB	Orders	-

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.

5. **Choose IAM Role**: Select the role you created (AWSGlueOrdersRole).


ap-south-1.console.aws.amazon.com/glue/home?region=ap-south-1#/v2/data-catalog/crawlers/add
AWS Glue > Crawlers > Add crawler
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
View
Create new IAM role
Update chosen IAM role
Cancel
Previous
Next

Configure security settings

IAM role Info

Existing IAM role: AWSGlueOrdersRole

Only IAM roles created by the AWS Glue console and have the prefix "AWSGlueServiceRole-" can be updated.

Security configuration - *optional*

Enable at-rest encryption with a security configuration.

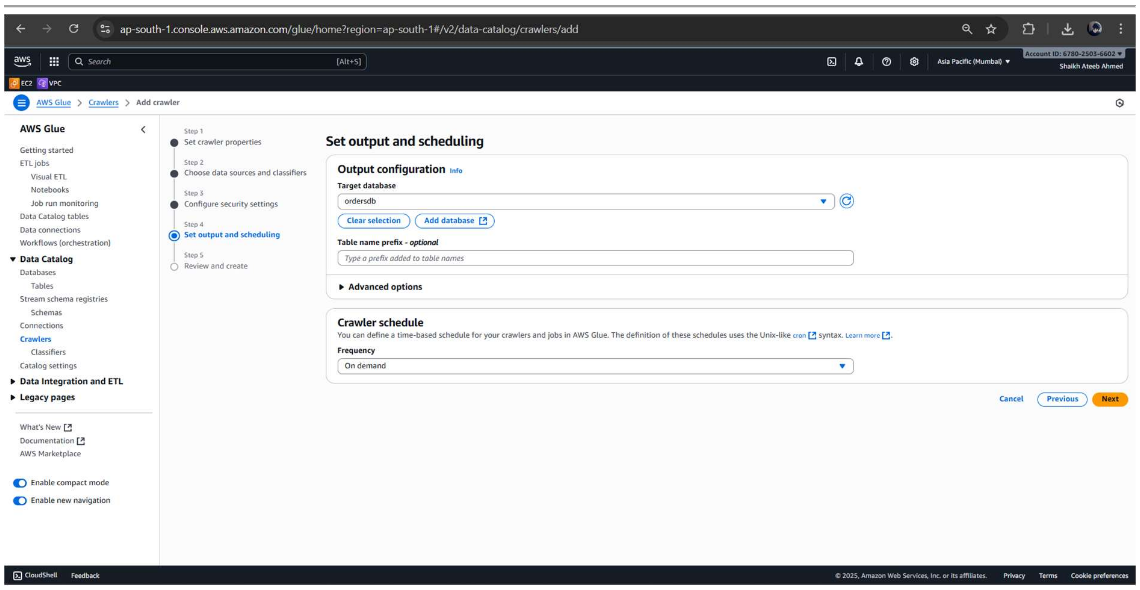
6. **Crawler output**:

   Choose **Output to a Data Catalog database**, Database: **OrdersDB**.

7. **Configure crawler options**:

   Frequency: **Run on demand**.

Shaikh Ateeb Ahmed
08-09-2025

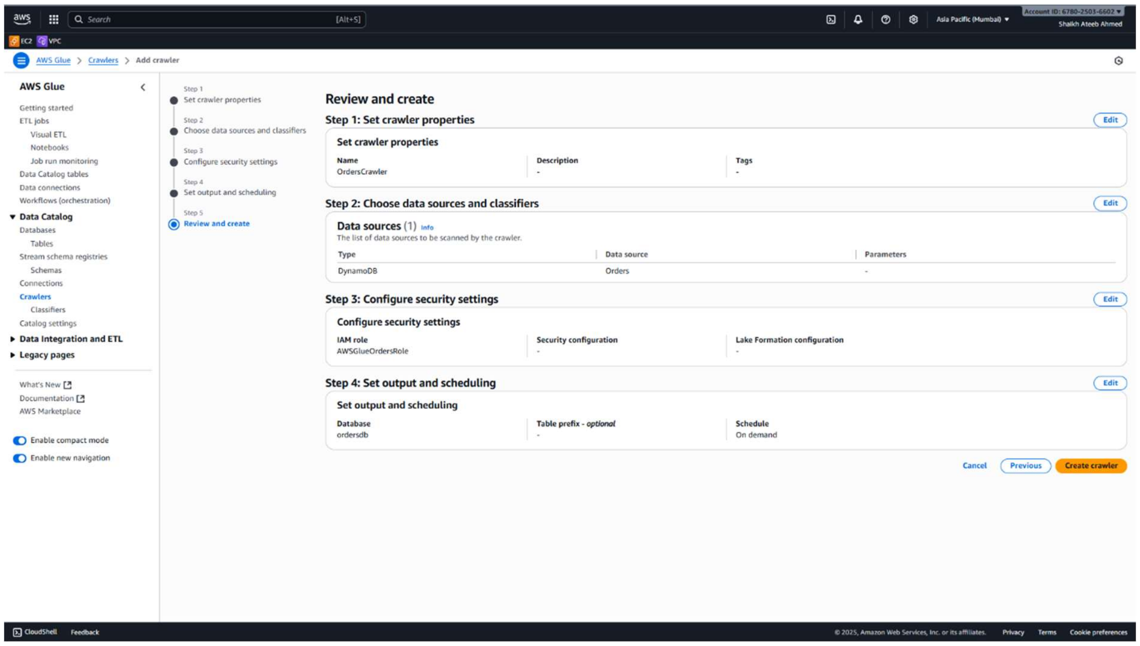


Set output and scheduling

Output configuration
Target database: ordersdb
Table name prefix - optional: Type a prefix added to table names

Crawler schedule
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: On demand

8. Review and **Finish**.



Review and create

Step 1: Set crawler properties

Name	Description	Tags
OrdersCrawler	-	-

Step 2: Choose data sources and classifiers

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

Type	Data source	Parameters
DynamoDB	Orders	-

Step 3: Configure security settings

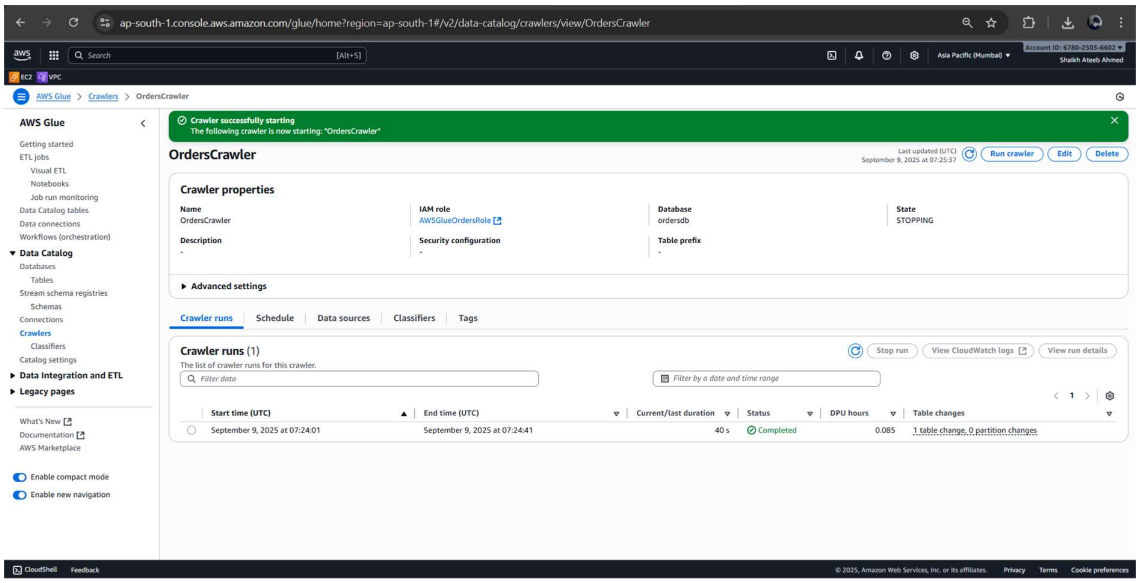
IAM role	Security configuration	Lake Formation configuration
AWSGlueOrdersRole	-	-

Step 4: Set output and scheduling

Database	Table prefix - optional	Schedule
ordersdb	-	On demand

9. After creation, select the crawler and click **Run crawler**.

Shaikh Ateeb Ahmed
08-09-2025



Crawler successfully starting
The following crawler is now starting: "OrdersCrawler"

OrdersCrawler

Last updated (UTC)
September 9, 2025 at 07:25:37

Crawler properties

Name	IAM role	Database	State
OrdersCrawler	AWSGlueOrdersRole	ordersdb	STOPPING
Description	**Security configuration**	**Table prefix**	
-	-	-	

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

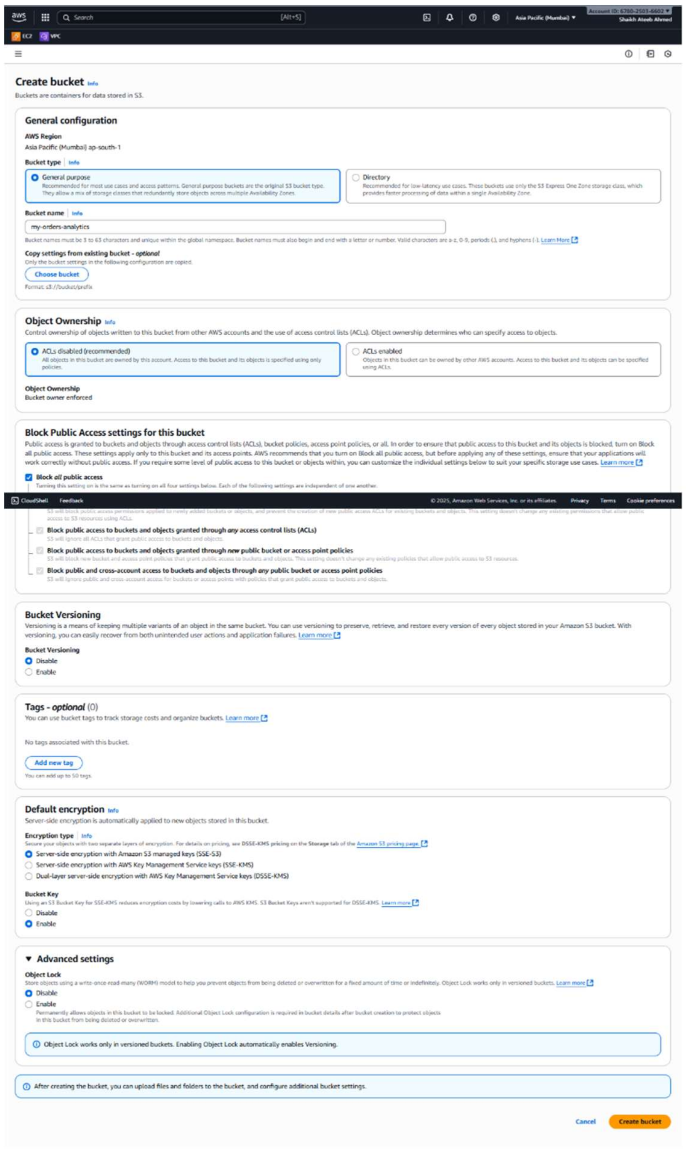
Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
September 9, 2025 at 07:24:01	September 9, 2025 at 07:24:41	40 s	Completed	0.085	1 table change, 0 partition changes

Step 4 — Create the AWS Glue ETL Job

1) Create the S3 target bucket

S3 → Create bucket → name my-orders-analytics (or your preferred name) → choose Region → Create.

Shaikh Ateeb Ahmed
08-09-2025



Account ID: 6780-2503-6602
Asia Pacific (Mumbai)
Shaikh Ateeb Ahmed

Create bucket Info
Buckets are containers for data stored in S3.

General configuration

AWS Region
Asia Pacific (Mumbai) ap-south-1

Bucket type Info
[✓ General purpose] Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory – Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info
my-orders-analytics
Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). Learn More

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix

Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.
[✓ ACLs disabled (recommended)] All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
ACLs enabled – Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

Block Public Access settings for this bucket
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

- [✓] **Block *all* public access**
  Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

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

  S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
  - [✓] **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
    S3 will ignore all ACLs that grant public access to buckets and objects.
  - [✓] **Block public access to buckets and objects granted through *new* public bucket or access point policies**
    S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
  - [✓] **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
    S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Bucket Versioning
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
[✓ Disable] Enable

Tags - *optional* (0)
You can use bucket tags to track storage costs and organize buckets. Learn more

No tags associated with this bucket.
Add new tag
You can add up to 50 tags.

Default encryption Info
Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type Info
Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing** on the **Storage** tab of the Amazon S3 pricing page.
[✓ Server-side encryption with Amazon S3 managed keys (SSE-S3)]
Server-side encryption with AWS Key Management Service keys (SSE-KMS)
Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. Learn more
Disable [✓ Enable]

▼ Advanced settings

Object Lock
Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock works only in versioned buckets. Learn more
[✓ Disable]
Enable
Permanently allows objects in this bucket to be locked. Additional Object Lock configuration is required in bucket details after bucket creation to protect objects in this bucket from being deleted or overwritten.

Object Lock works only in versioned buckets. Enabling Object Lock automatically enables Versioning.

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

2) Create / verify the IAM role for Glue jobs

Create role AWSGlueOrdersETLRole and attach these managed policies:

AWSGlueServiceRole (or similar Glue-managed role)

AmazonDynamoDBReadOnlyAccess

AmazonS3FullAccess (or tightly-scoped S3 policy to your bucket)

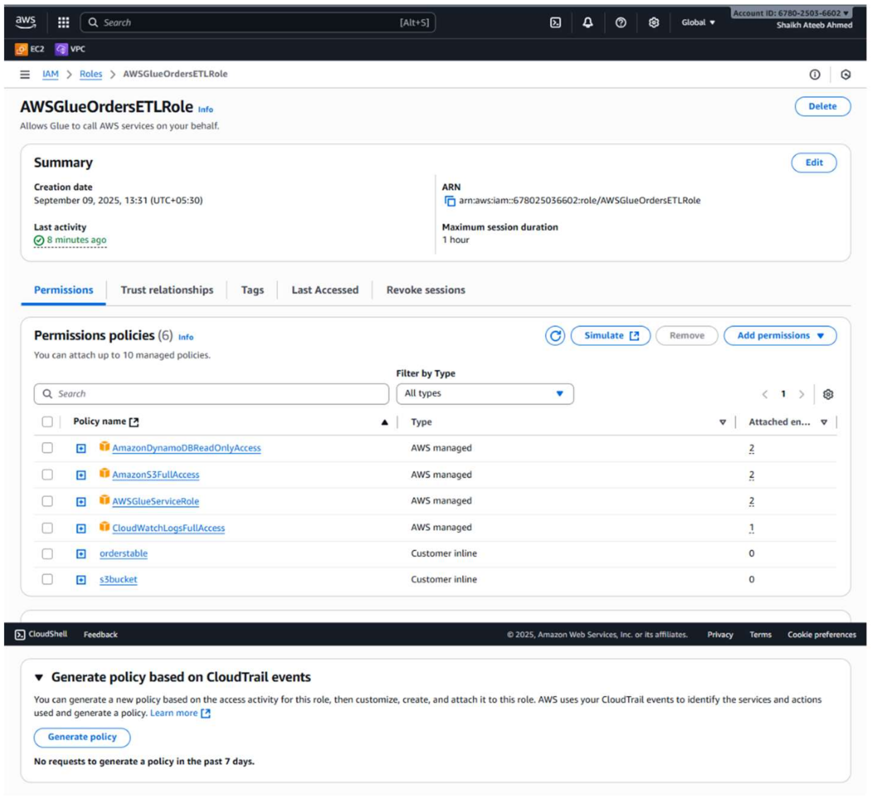
DynamoDB read: DescribeTable, Scan, GetItem, Query for the Orders table ARN.

S3 access to your bucket: GetObject, PutObject, ListBucket for arn:aws:s3:::my-orders-analytics01.

Shaikh Ateeb Ahmed
08-09-2025

Minimal trust policy (Glue service principal):

```
{
 "Version":"2012-10-17",
 "Statement":[
  {
   "Effect":"Allow",
   "Principal":{"Service":"glue.amazonaws.com"},
   "Action":"sts:AssumeRole"
  }
 ]
}
```


Account ID: 6780-2503-6602
Shaikh Ateeb Ahmed

IAM > Roles > AWSGlueOrdersETLRole

AWSGlueOrdersETLRole

Allows Glue to call AWS services on your behalf.

Summary

Creation date
September 09, 2025, 13:31 (UTC+05:30)

Last activity
8 minutes ago

ARN
arn:aws:iam::678025036602:role/AWSGlueOrdersETLRole

Maximum session duration
1 hour

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

Permissions policies (6)

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached en...
AmazonDynamoDBReadOnlyAccess	AWS managed	2
AmazonS3FullAccess	AWS managed	2
AWSGlueServiceRole	AWS managed	2
CloudWatchLogsFullAccess	AWS managed	1
orderstable	Customer inline	0
s3bucket	Customer inline	0

© 2025, Amazon Web Services, Inc. or its affiliates.

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

No requests to generate a policy in the past 7 days.

3) Create the Glue Job

1. **Open AWS Console → AWS Glue → Jobs → Add job → Visual.**
2. **Name: AWS Glue Data Catalog**
3. **Database: ordersdb**

4. **Table: orders**
5. **IAM role: select AWSGlueOrdersETLRole (created above).**

Create Glue job:

AWS Console → AWS Glue → Jobs → Add job.

Name: OrdersToParquet (or whatever you prefer).

IAM role: AWSGlueOrdersETLRole.

Type: Spark, Glue version: 3.0 or 4.0, Python 3.

This job runs: A new script authored by you OR point to an S3 script (see below).

TempDir: s3://my-orders-analytics01/temp/ (set as --TempDir in job properties).

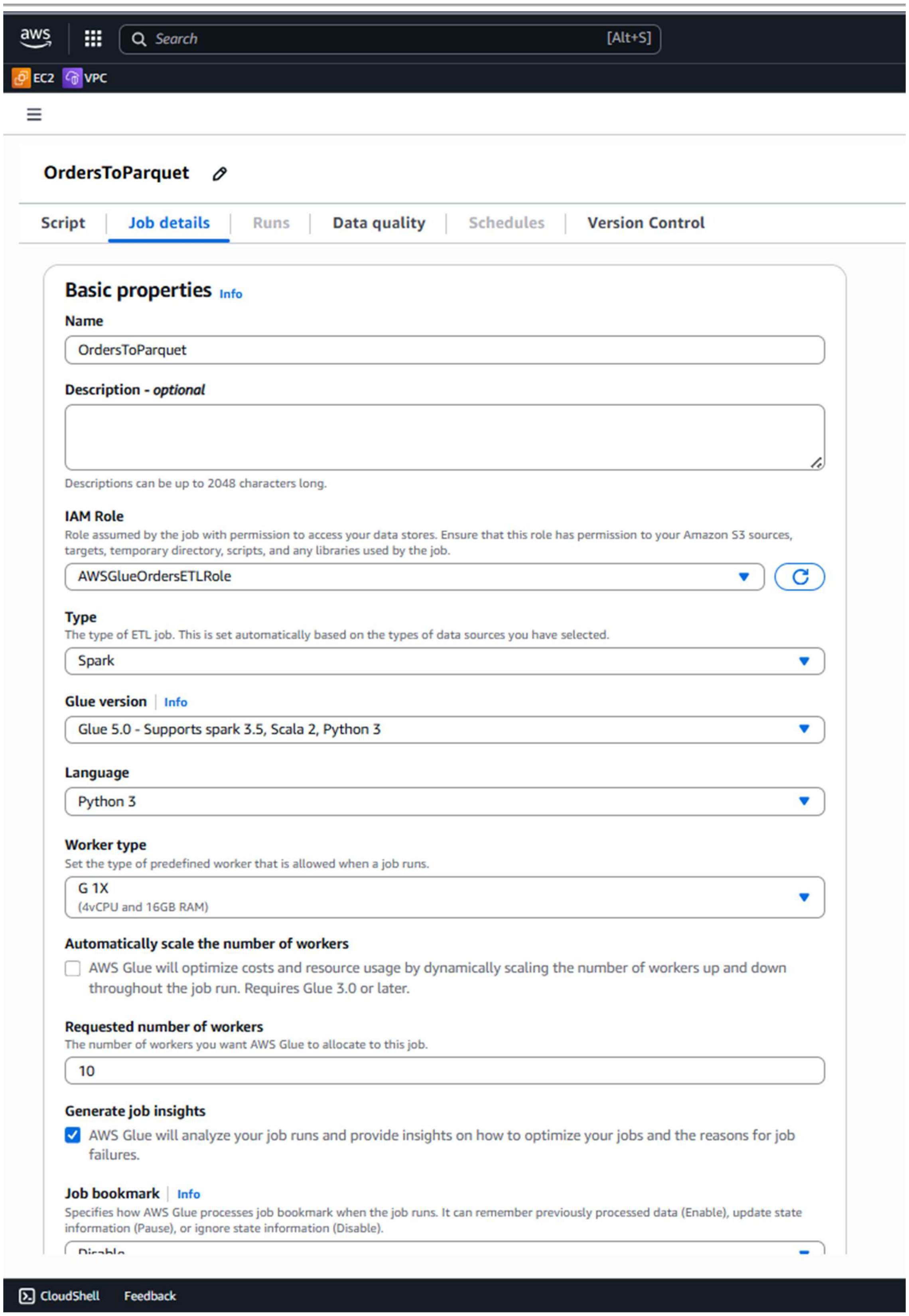
Worker type / number: start small (Standard, 2 workers).

Paste the script (below) into the job script editor (or upload to S3 and supply path).

Run the job (Console → select job → Run) or via CLI:

aws glue start-job-run --job-name OrdersToParquet

Shaikh Ateeb Ahmed
08-09-2025



OrdersToParquet

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

Basic properties Info

Name
OrdersToParquet

Description - *optional*

Descriptions can be up to 2048 characters long.

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

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

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

Language
Python 3

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

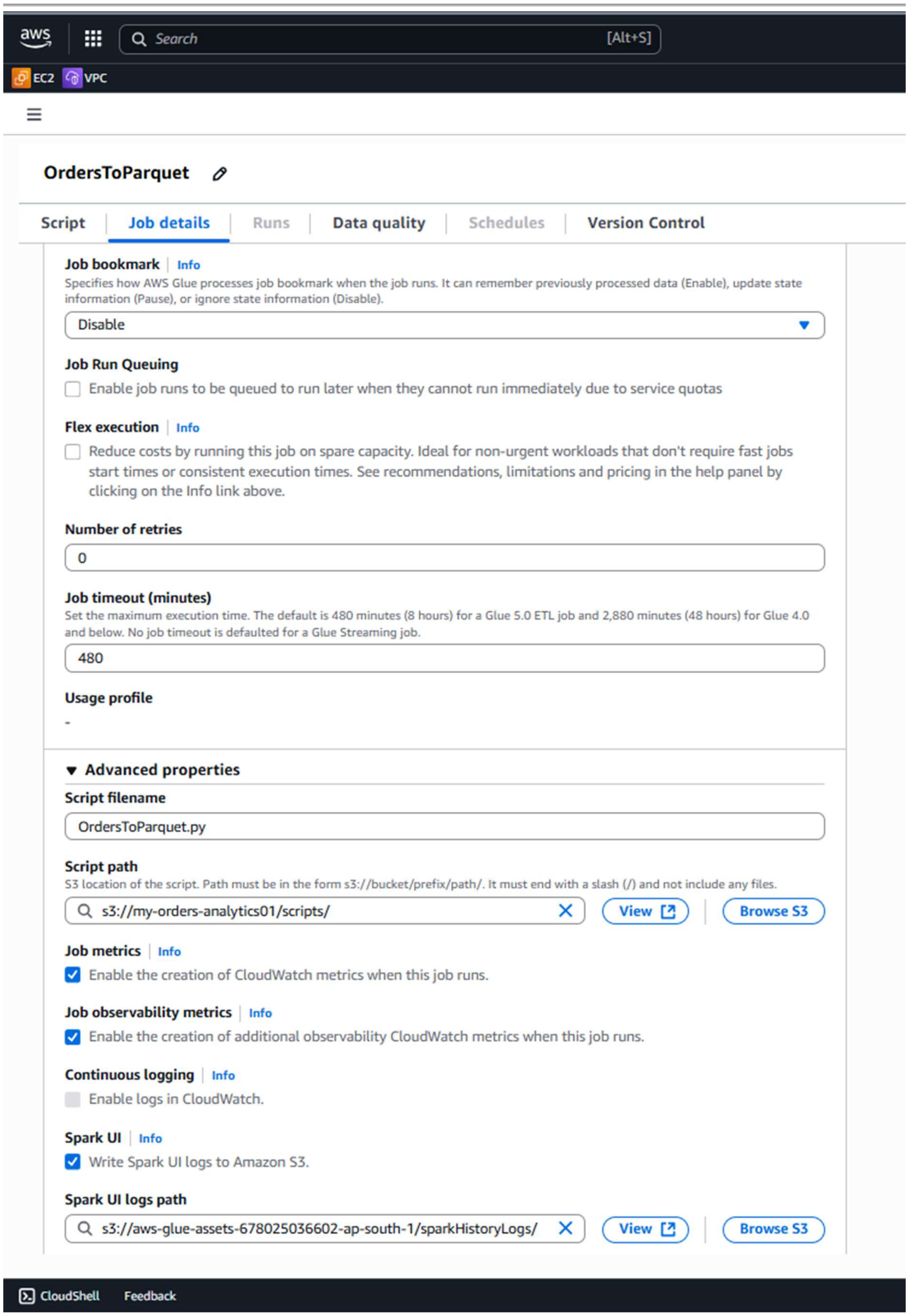
Automatically scale the number of workers
☐ AWS Glue will optimize costs and resource usage by dynamically scaling the number of workers up and down throughout the job run. Requires Glue 3.0 or later.

Requested number of workers
The number of workers you want AWS Glue to allocate to this job.
10

Generate job insights
☑ AWS Glue will analyze your job runs and provide insights on how to optimize your jobs and the reasons for job failures.

Job bookmark | Info
Specifies how AWS Glue processes job bookmark when the job runs. It can remember previously processed data (Enable), update state information (Pause), or ignore state information (Disable).

Shaikh Ateeb Ahmed
08-09-2025


aws
Search [Alt+S]
EC2
VPC

OrdersToParquet

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

Job bookmark | Info
Specifies how AWS Glue processes job bookmark when the job runs. It can remember previously processed data (Enable), update state information (Pause), or ignore state information (Disable).
Disable

Job Run Queuing
☐ Enable job runs to be queued to run later when they cannot run immediately due to service quotas

Flex execution | Info
☐ Reduce costs by running this job on spare capacity. Ideal for non-urgent workloads that don't require fast jobs start times or consistent execution times. See recommendations, limitations and pricing in the help panel by clicking on the Info link above.

Number of retries
0

Job timeout (minutes)
Set the maximum execution time. The default is 480 minutes (8 hours) for a Glue 5.0 ETL job and 2,880 minutes (48 hours) for Glue 4.0 and below. No job timeout is defaulted for a Glue Streaming job.
480

Usage profile
-

▼ **Advanced properties**

Script filename
OrdersToParquet.py

Script path
S3 location of the script. Path must be in the form s3://bucket/prefix/path/. It must end with a slash (/) and not include any files.
s3://my-orders-analytics01/scripts/

Job metrics | Info
✓ Enable the creation of CloudWatch metrics when this job runs.

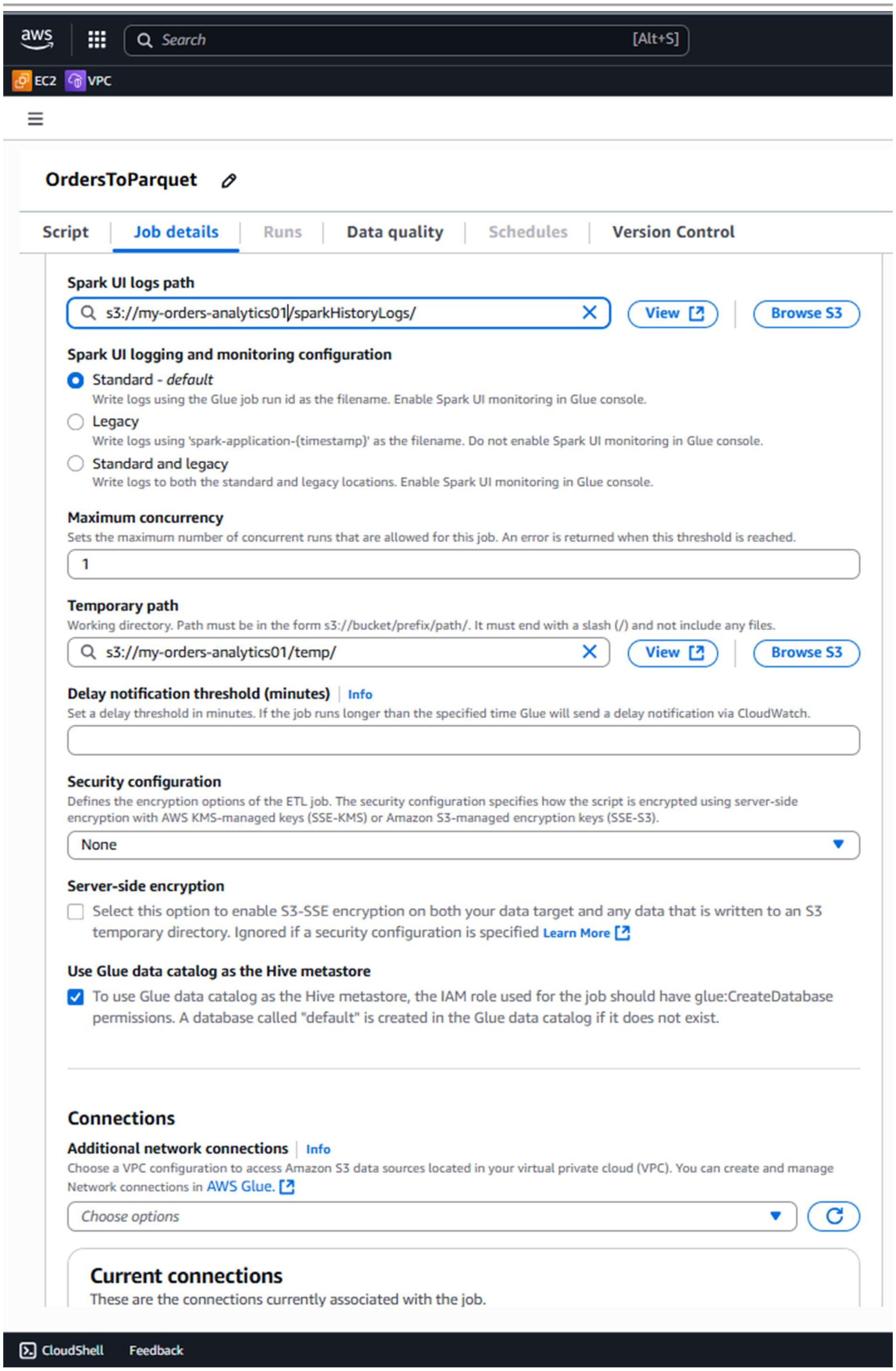
Job observability metrics | Info
✓ Enable the creation of additional observability CloudWatch metrics when this job runs.

Continuous logging | Info
☐ Enable logs in CloudWatch.

Spark UI | Info
✓ Write Spark UI logs to Amazon S3.

Spark UI logs path
s3://aws-glue-assets-678025036602-ap-south-1/sparkHistoryLogs/

Shaikh Ateeb Ahmed
08-09-2025



OrdersToParquet

Spark UI logs path

s3://my-orders-analytics01/sparkHistoryLogs/

Spark UI logging and monitoring configuration

● Standard - *default*
Write logs using the Glue job run id as the filename. Enable Spark UI monitoring in Glue console.

○ Legacy
Write logs using 'spark-application-{timestamp}' as the filename. Do not enable Spark UI monitoring in Glue console.

○ Standard and legacy
Write logs to both the standard and legacy locations. Enable Spark UI monitoring in Glue console.

Maximum concurrency
Sets the maximum number of concurrent runs that are allowed for this job. An error is returned when this threshold is reached.

1

Temporary path
Working directory. Path must be in the form s3://bucket/prefix/path/. It must end with a slash (/) and not include any files.

s3://my-orders-analytics01/temp/

Delay notification threshold (minutes) Info
Set a delay threshold in minutes. If the job runs longer than the specified time Glue will send a delay notification via CloudWatch.

Security configuration
Defines the encryption options of the ETL job. The security configuration specifies how the script is encrypted using server-side encryption with AWS KMS-managed keys (SSE-KMS) or Amazon S3-managed encryption keys (SSE-S3).

None

Server-side encryption
☐ Select this option to enable S3-SSE encryption on both your data target and any data that is written to an S3 temporary directory. Ignored if a security configuration is specified Learn More

Use Glue data catalog as the Hive metastore
☑ To use Glue data catalog as the Hive metastore, the IAM role used for the job should have glue:CreateDatabase permissions. A database called "default" is created in the Glue data catalog if it does not exist.

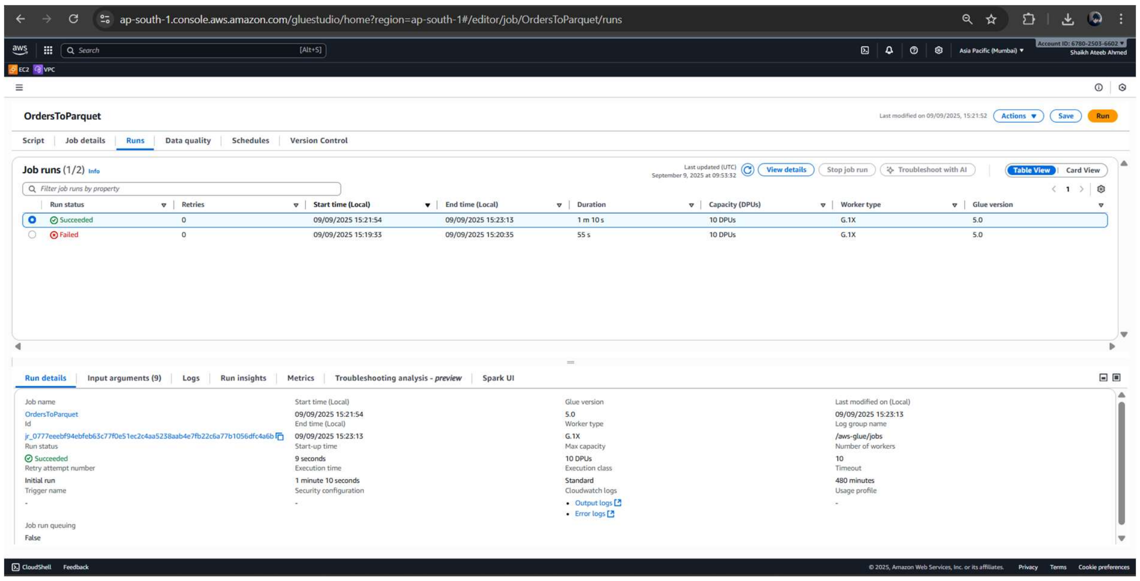
Connections

Additional network connections Info
Choose a VPC configuration to access Amazon S3 data sources located in your virtual private cloud (VPC). You can create and manage Network connections in AWS Glue.

Choose options

Current connections
These are the connections currently associated with the job.

Shaikh Ateeb Ahmed
08-09-2025


ap-south-1.console.aws.amazon.com/gluestudio/home?region=ap-south-1#/editor/job/OrdersToParquet/runs
Search [Alt+S]
Asia Pacific (Mumbai)
Account ID: 6780-2503-6602
Shaikh Ateeb Ahmed
EC2 VPC
OrdersToParquet
Last modified on 09/09/2025, 15:21:52
Actions
Save
Run
Script | Job details | Runs | Data quality | Schedules | Version Control
Job runs (1/2) Info
Last updated (UTC) September 9, 2025 at 09:53:32
View details
Stop job run
Troubleshoot with AI
Table View | Card View
Filter job runs by property
Run details | Input arguments (9) | Logs | Run insights | Metrics | Troubleshooting analysis - preview | Spark UI
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Succeeded	0	09/09/2025 15:21:54	09/09/2025 15:23:13	1 m 10 s	10 DPUs	G.1X	5.0
Failed	0	09/09/2025 15:19:33	09/09/2025 15:20:35	55 s	10 DPUs	G.1X	5.0

Job name: OrdersToParquet	Start time (Local): 09/09/2025 15:21:54	Glue version: 5.0	Last modified on (Local): 09/09/2025 15:23:13
Id: jr_0777eeebf94ebfeb63c77f0e51ec2c4aa5238aab4e7fb22c6a77b1056dfc4a6b	End time (Local): 09/09/2025 15:23:13	Worker type: G.1X	Log group name: /aws-glue/jobs
Run status: Succeeded	Start-up time: 9 seconds	Max capacity: 10 DPUs	Number of workers: 10
Retry attempt number: Initial run	Execution time: 1 minute 10 seconds	Execution class: Standard	Timeout: 480 minutes
Trigger name: -	Security configuration: -	Cloudwatch logs: Output logs, Error logs	Usage profile: -
Job run queuing: False			

Copy-paste-ready Glue PySpark script

import sys

import re

from awsglue.transforms import *

from awsglue.utils import getResolvedOptions

from pyspark.context import SparkContext

from awsglue.context import GlueContext

from awsglue.job import Job

from awsglue.dynamicframe import DynamicFrame

import pyspark.sql.functions as F

# --- JOB ARGS ---

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

sc = SparkContext()

glueContext = GlueContext(sc)

spark = glueContext.spark_session

job = Job(glueContext)

```
job.init(args['JOB_NAME'], args)


# --- CONFIG ---

DATABASE = "ordersdb"

TABLE_NAME = "orders"

OUTPUT_PATH = "s3://my-orders-analytics01/shipped/"


# --- READ from Glue Data Catalog (DynamoDB table) ---

datasource_dyf = glueContext.create_dynamic_frame.from_catalog(

    database=DATABASE,

    table_name=TABLE_NAME,

    transformation_ctx="datasource_dyf"

)


# Convert to Spark DataFrame

df = datasource_dyf.toDF()


# --- FIX Amount column ---

# DynamoDB "choice" type: struct with {double, long}

df = df.withColumn(

    "Amount",

    F.when(F.col("Amount.double").isNotNull(), F.col("Amount.double"))

     .when(F.col("Amount.long").isNotNull(), F.col("Amount.long").cast("double"))

     .otherwise(F.lit(None))

)


# --- FILTER shipped orders (case-insensitive) ---

filtered_df = df.filter(F.lower(F.col("Status")) == "shipped")
```

# Convert back to DynamicFrame

filtered_dyf = DynamicFrame.fromDF(filtered_df, glueContext, "filtered_dyf")

# --- WRITE to S3 ---

glueContext.write_dynamic_frame.from_options(

frame=filtered_dyf,

connection_type="s3",

connection_options={"path": OUTPUT_PATH, "partitionKeys": []},

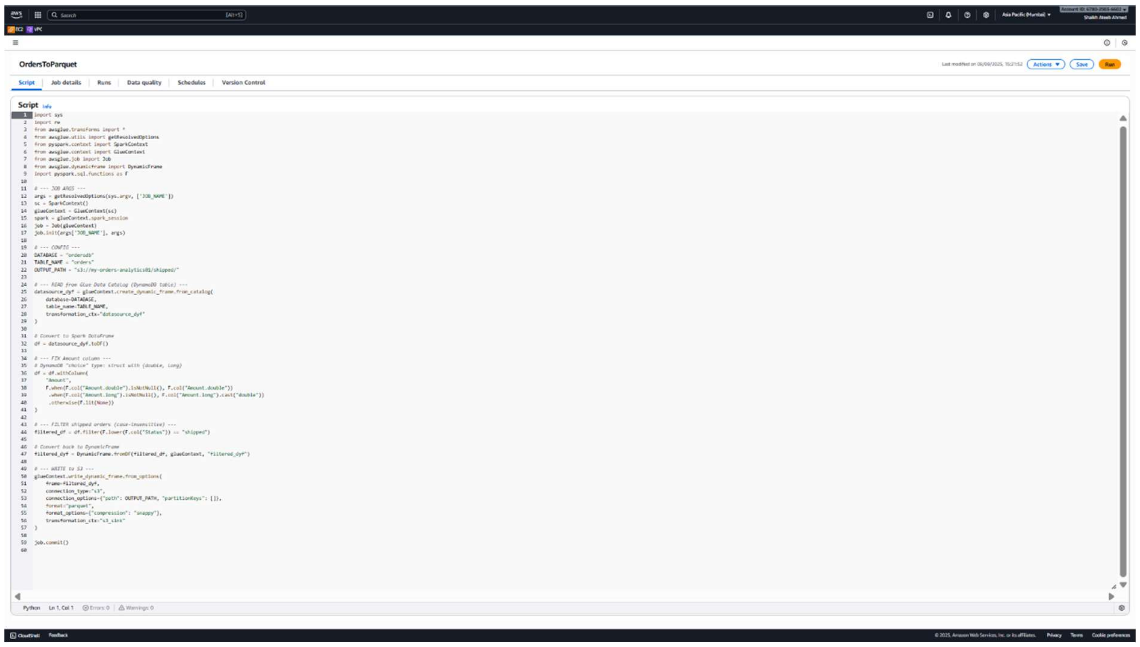
format="parquet",

format_options={"compression": "snappy"},

transformation_ctx="s3_sink"

)

job.commit()



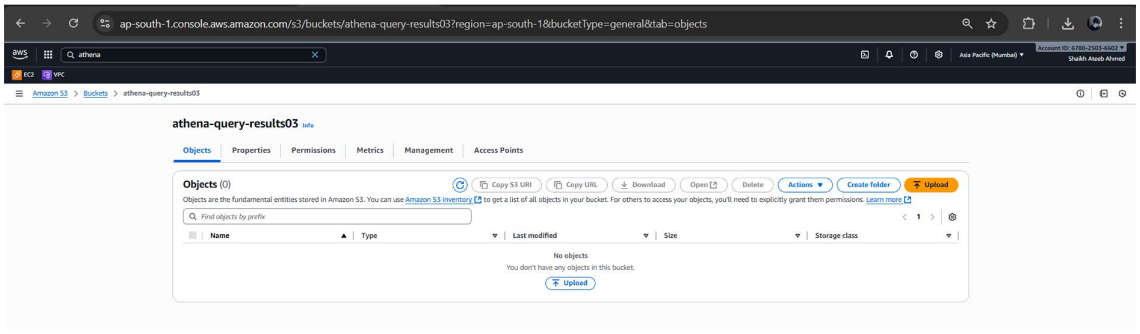
```python
import sys
import re
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.dynamicframe import DynamicFrame
import pyspark.sql.functions as F

# --- JOB ARGS ---
args = getResolvedOptions(sys.argv, ['JOB_NAME'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# --- CONFIG ---
DATABASE = "ordersdb"
TABLE_NAME = "orders"
OUTPUT_PATH = "s3://my-orders-analytics01/shipped/"

# --- READ from Glue Data Catalog (DynamoDB table) ---
datasource_dyf = glueContext.create_dynamic_frame.from_catalog(
    database=DATABASE,
    table_name=TABLE_NAME,
    transformation_ctx="datasource_dyf"
)

# Convert to Spark DataFrame
df = datasource_dyf.toDF()

# --- FIX Amount column ---
# DynamoDB "choice" type: struct with (double, long)
df = df.withColumn(
    "Amount",
    F.when(F.col("Amount.double").isNotNull(), F.col("Amount.double"))
     .when(F.col("Amount.long").isNotNull(), F.col("Amount.long").cast("double"))
     .otherwise(F.lit(None))
)

# --- FILTER shipped orders (case-insensitive) ---
filtered_df = df.filter(F.lower(F.col("Status")) == "shipped")

# Convert back to DynamicFrame
filtered_dyf = DynamicFrame.fromDF(filtered_df, glueContext, "filtered_dyf")

# --- WRITE to S3 ---
glueContext.write_dynamic_frame.from_options(
    frame=filtered_dyf,
    connection_type="s3",
    connection_options={"path": OUTPUT_PATH, "partitionKeys": []},
    format="parquet",
    format_options={"compression": "snappy"},
    transformation_ctx="s3_sink"
)

job.commit()
```

Shaikh Ateeb Ahmed
08-09-2025

Step 5: Query with Athena

Create an S3 bucket for Athena query results

1. **Open S3 → Create bucket.**
2. **Name it e.g. athena-query-results03 (bucket names must be globally unique).**
3. **Choose same Region as your Glue/Athena region (ap-south-1).**
4. **Create the bucket.**


ap-south-1.console.aws.amazon.com/s3/buckets/athena-query-results03?region=ap-south-1&bucketType=general&tab=objects

Amazon S3 > Buckets > athena-query-results03

athena-query-results03 Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (0)

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

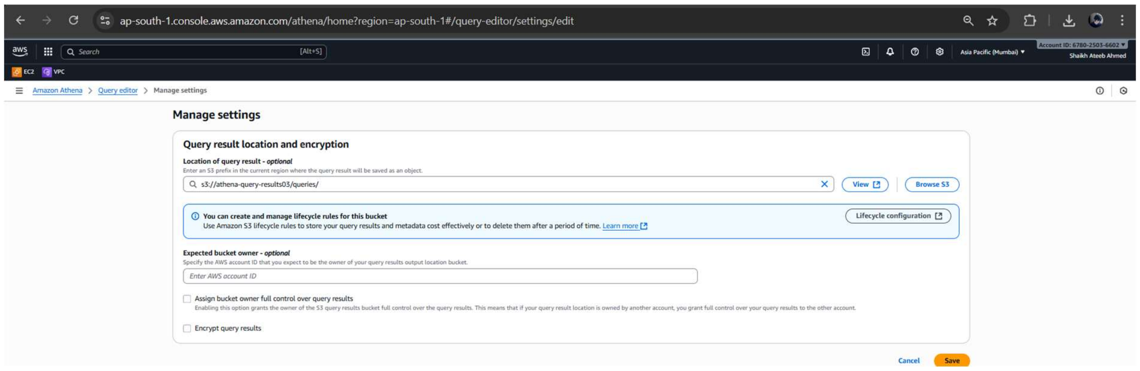
Name	Type	Last modified	Size	Storage class

No objects
You don't have any objects in this bucket.

Configure Athena to use that S3 location

1. **Open AWS Console → Athena (ensure region = same region as Glue & S3).**
2. **Click the edit Settings.**
3. **Under Query result location, paste:**

   s3://athena-query-results03/queries/
4. **Save.**


ap-south-1.console.aws.amazon.com/athena/home?region=ap-south-1#/query-editor/settings/edit

Amazon Athena > Query editor > Manage settings

Manage settings

Query result location and encryption

Location of query result - *optional*
Enter an S3 prefix in the current region where the query result will be saved as an object.
s3://athena-query-results03/queries/

You can create and manage lifecycle rules for this bucket
Use Amazon S3 lifecycle rules to store your query results and metadata cost effectively or to delete them after a period of time. Learn more

Expected bucket owner - *optional*
Specify the AWS account ID that you expect to be the owner of your query results output location bucket.
Enter AWS account ID

☐ Assign bucket owner full control over query results
Enabling this option grants the owner of the S3 query results bucket full control over the query results. This means that if your query result location is owned by another account, you grant full control over your query results to the other account.

☐ Encrypt query results

Example queries (use your database/table name)

If your table name in Glue/Athena is orders in ordersdb, qualify it as shipped

Shaikh Ateeb Ahmed
08-09-2025

Top customers by total spend (Shipped)

SELECT Customer, SUM(Amount) AS TotalSpent

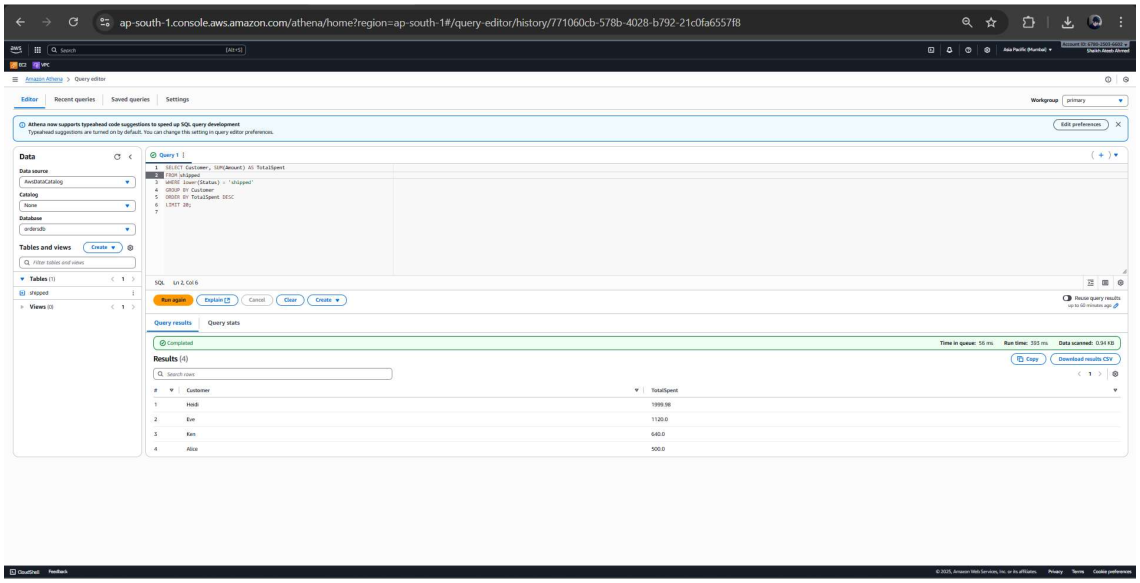
FROM orders_shipped

WHERE lower(Status) = 'shipped'

GROUP BY Customer

ORDER BY TotalSpent DESC

LIMIT 20;



Athena now supports typeahead code suggestions to speed up SQL query development
Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

Data source: AwsDataCatalog
Catalog: None
Database: ordersdb
Tables (1): shipped
Views (0)

```
SELECT Customer, SUM(Amount) AS TotalSpent
FROM shipped
WHERE lower(Status) = 'shipped'
GROUP BY Customer
ORDER BY TotalSpent DESC
LIMIT 20;
```

SQL Ln 2, Col 6

Completed — Time in queue: 56 ms Run time: 393 ms Data scanned: 0.94 KB

Results (4)

#	Customer	TotalSpent
1	Heidi	1999.98
2	Eve	1120.0
3	Ken	640.0
4	Alice	500.0
