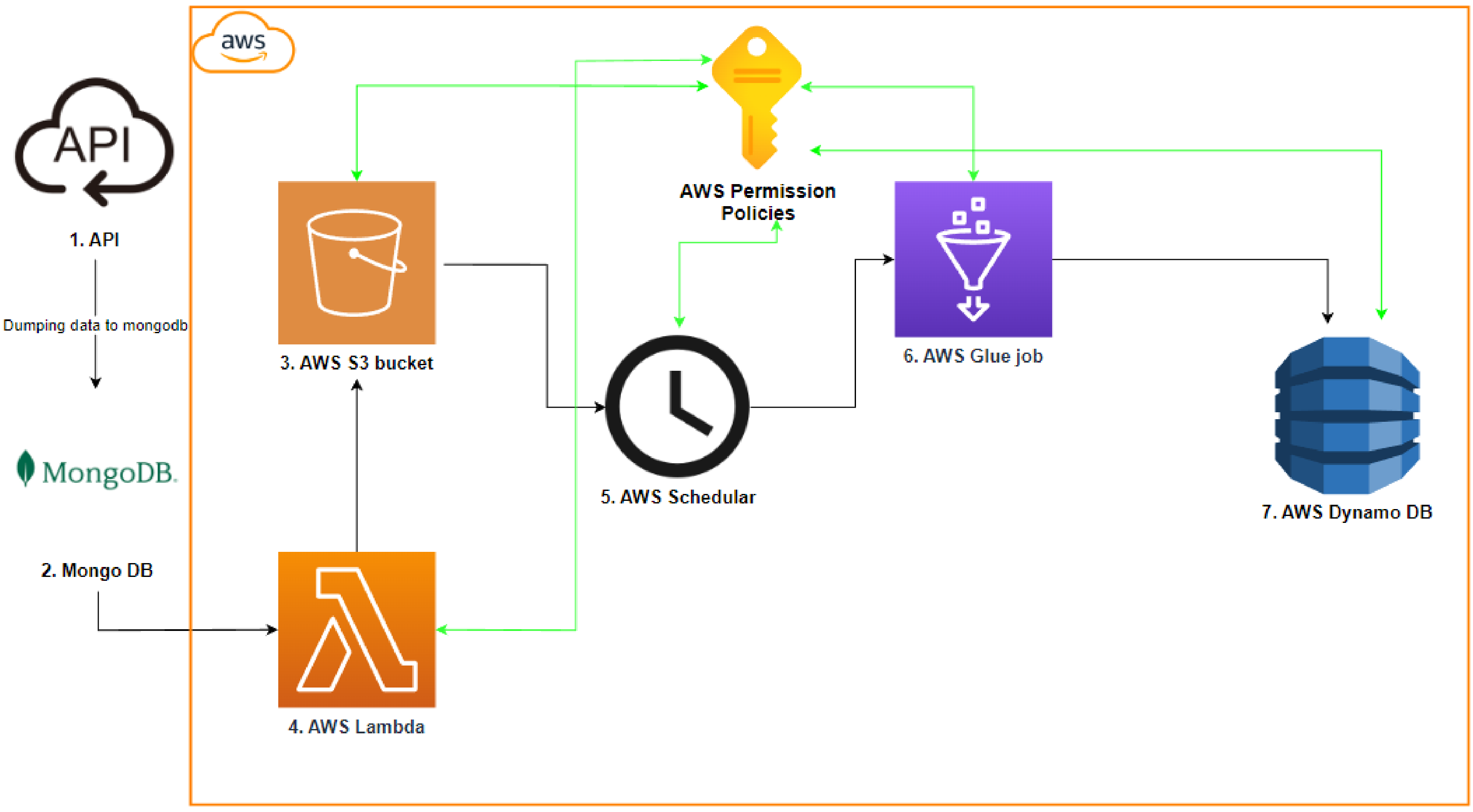
ETL PIPELINE IN AWS CLOUD

Introduction

The Aim of the project is to perform ETL Pipeline in AWS using different services of AWS.

Architecture



Explanation:

First, we will be extract data from the API (https://www.consumerfinance.gov/data-research/consumer-complaints/search/api/v1/) From some specific interval date and dumping data in MongoDB.

Before dumping data to s3 we will be attaching policies roles for lambda,s3, event bridge scheduler, glue, dynamo DB

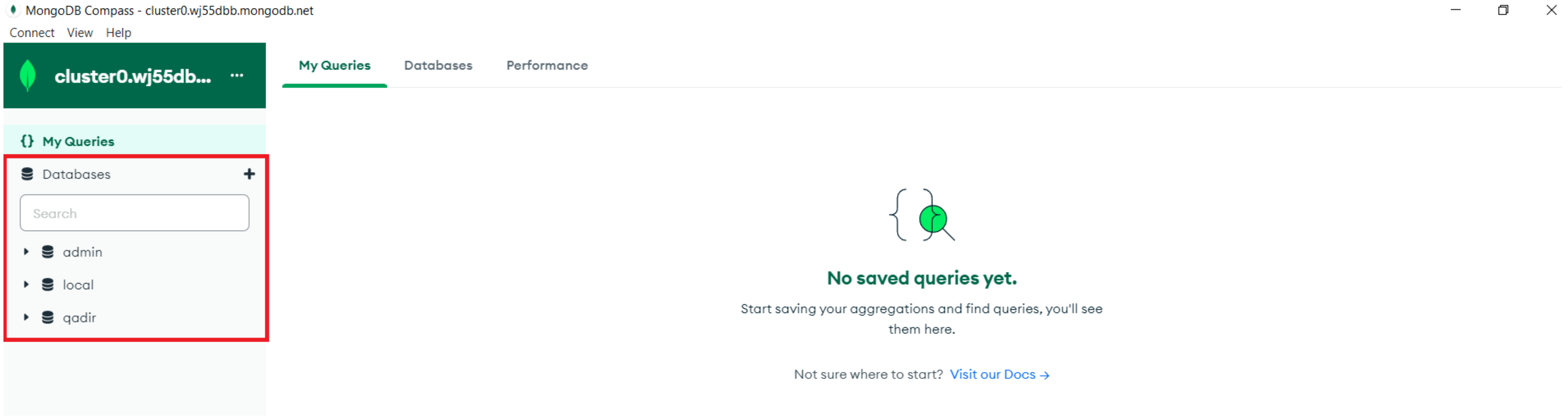
Next from MongoDB dumping data to s3, with help of AWS lambda using python request while dumping data into s3 configured AWS event bridge to dump data for every 2 min with the help of scheduler.

Now, we need to create Dynamo DB without creating any kind of schema, with the help of glue job writing script dynamically we will be dumping data from s3 to dynamo db.

Finally, we create workflow in glue, will we be dumping data in to Dynamo DB.

Implementation:

Mongo db. without dumping any records into it from api

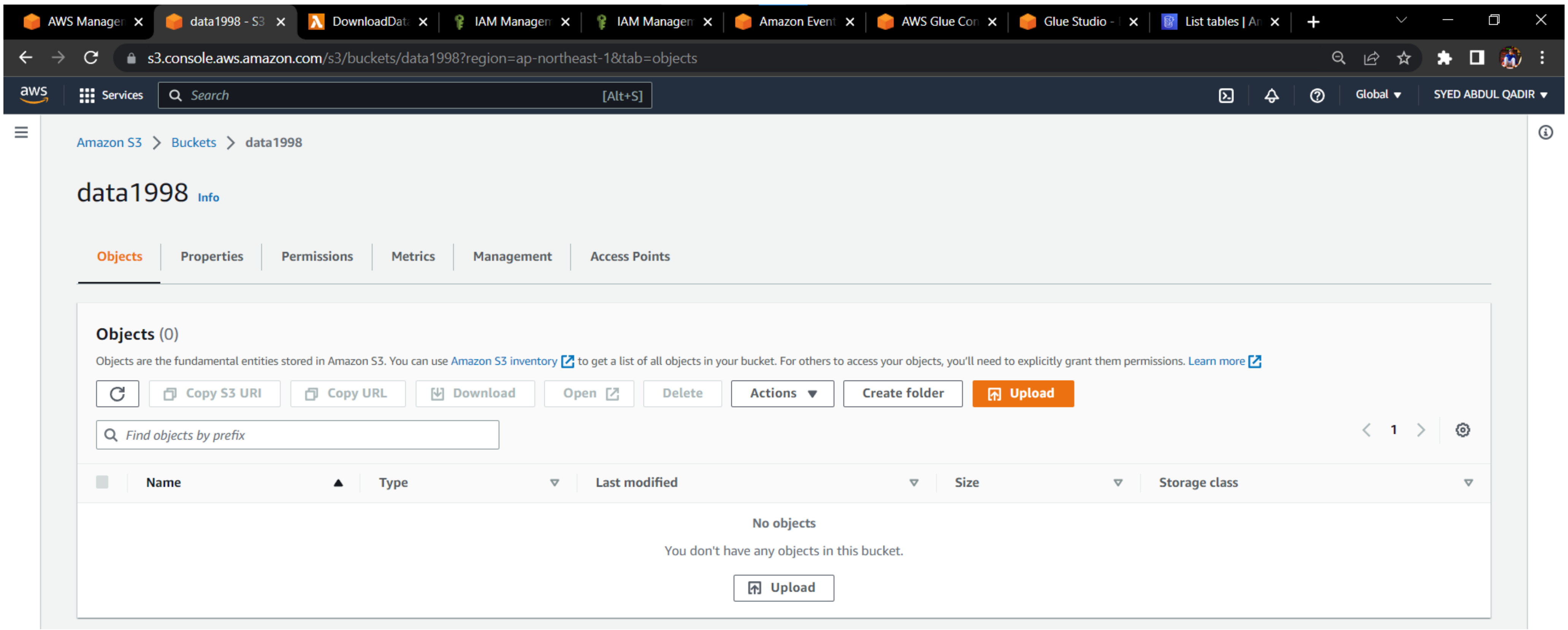


No saved queries yet.

Start saving your aggregations and find queries, you'll see them here.

Not sure where to start? Visit our Docs →

Aws s3 bucket with name **data1998** without dumping any records.



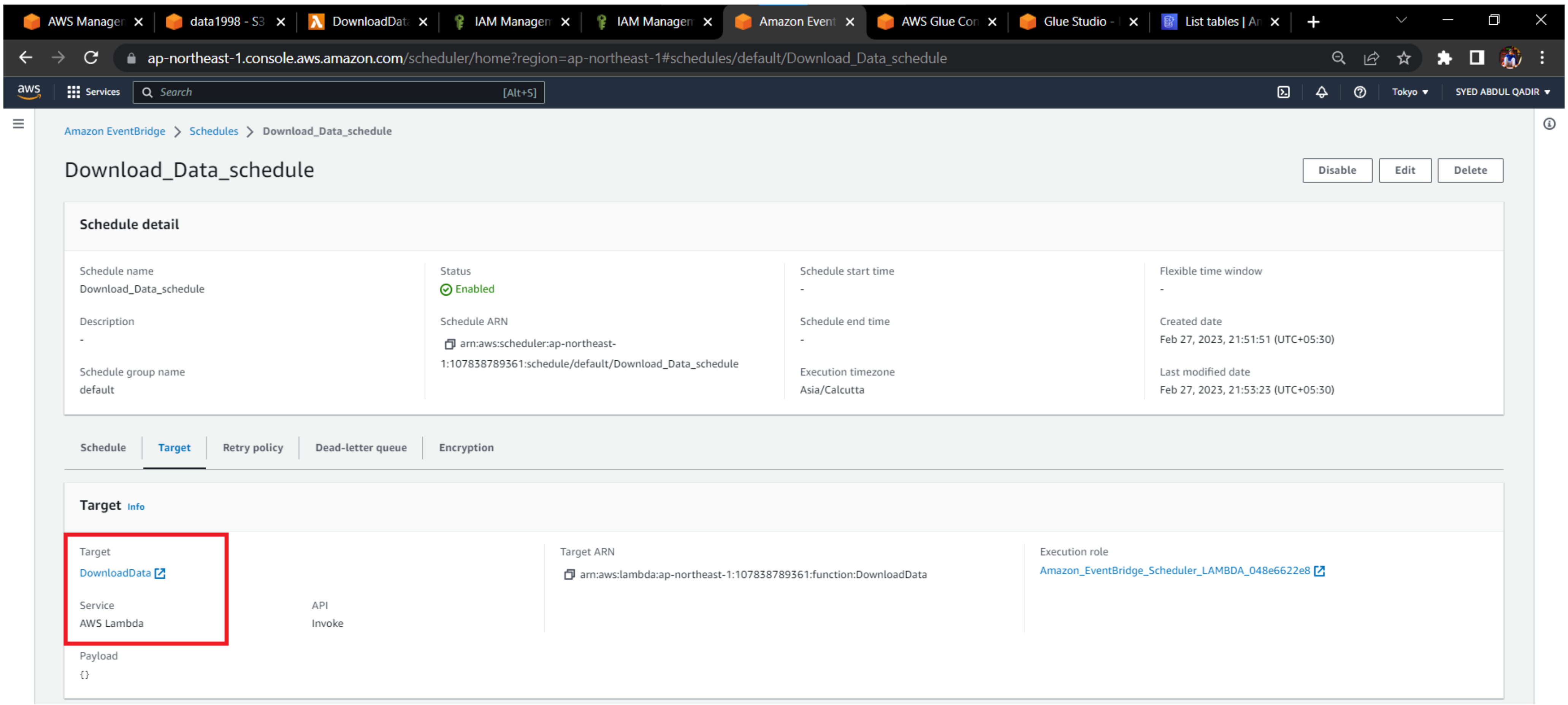
Objects (0)

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

No objects

You don't have any objects in this bucket.

Configuring amazon event bridge schedules, Name it as **Download_Data_schedule** and will be scheduling it for every 2 min and select targeted lambda with name **DowloadData**



Download_Data_schedule

Schedule detail

Schedule name
Download_Data_schedule

Description
-

Schedule group name
default

Status
Enabled

Schedule ARN
arn:aws:scheduler:ap-northeast-1:107838789361:schedule/default/Download_Data_schedule

Schedule start time
-

Schedule end time
-

Execution timezone
Asia/Calcutta

Flexible time window
-

Created date
Feb 27, 2023, 21:51:51 (UTC+05:30)

Last modified date
Feb 27, 2023, 21:53:23 (UTC+05:30)

Target Info

Target
DownloadData

Service
AWS Lambda

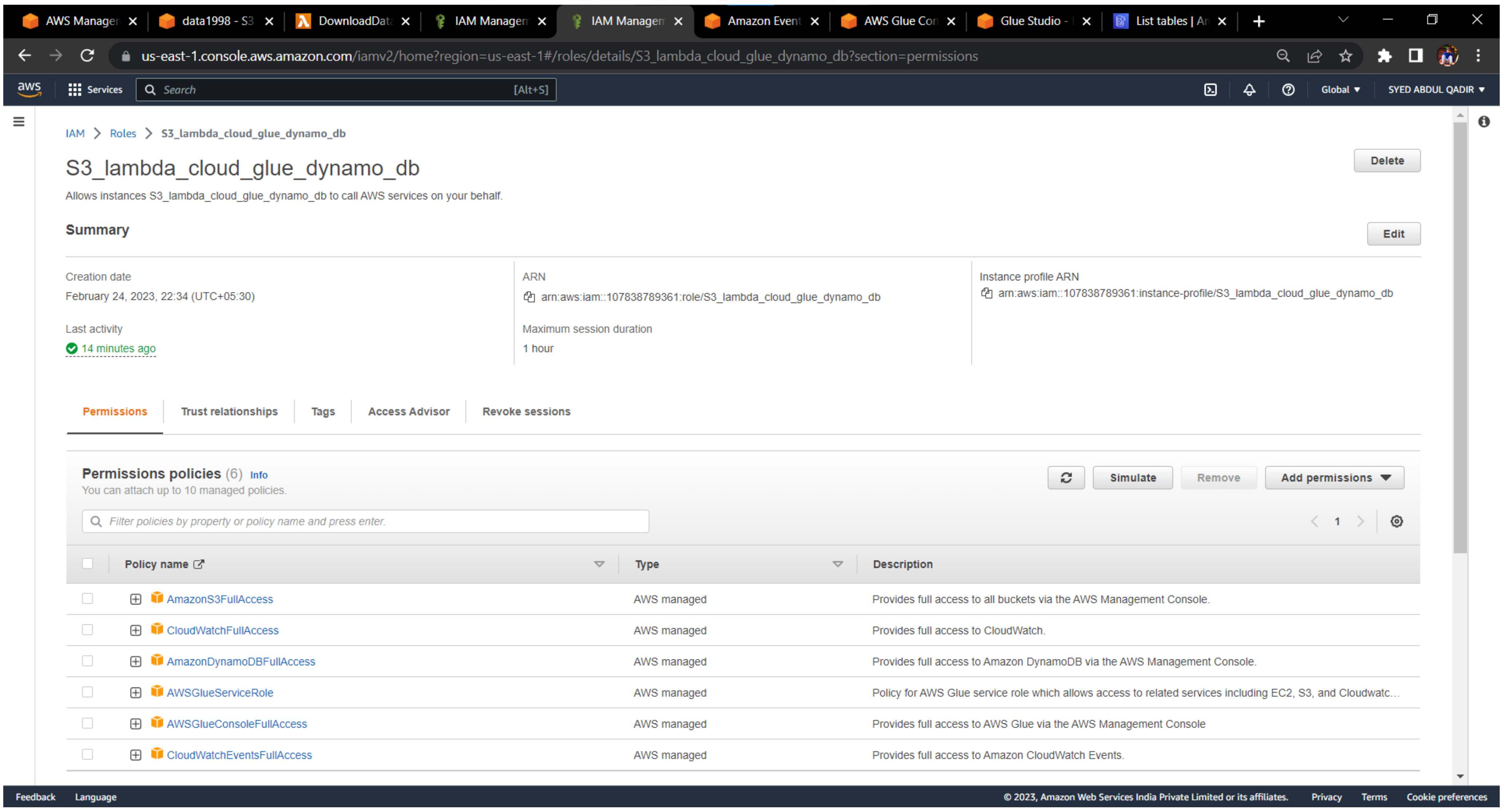
API
Invoke

Payload
{}

Target ARN
arn:aws:lambda:ap-northeast-1:107838789361:function:DownloadData

Execution role
Amazon_EventBridge_Scheduler_LAMBDA_048e6622e8

Attaching Permission policies with name **S3_lambda_cloud_glue_dynamo_db**



us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/roles/details/S3_lambda_cloud_glue_dynamo_db?section=permissions

IAM > Roles > S3_lambda_cloud_glue_dynamo_db

S3_lambda_cloud_glue_dynamo_db

Allows instances S3_lambda_cloud_glue_dynamo_db to call AWS services on your behalf.

Summary

Creation date
February 24, 2023, 22:34 (UTC+05:30)

Last activity
14 minutes ago

ARN
arn:aws:iam::107838789361:role/S3_lambda_cloud_glue_dynamo_db

Maximum session duration
1 hour

Instance profile ARN
arn:aws:iam::107838789361:instance-profile/S3_lambda_cloud_glue_dynamo_db

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

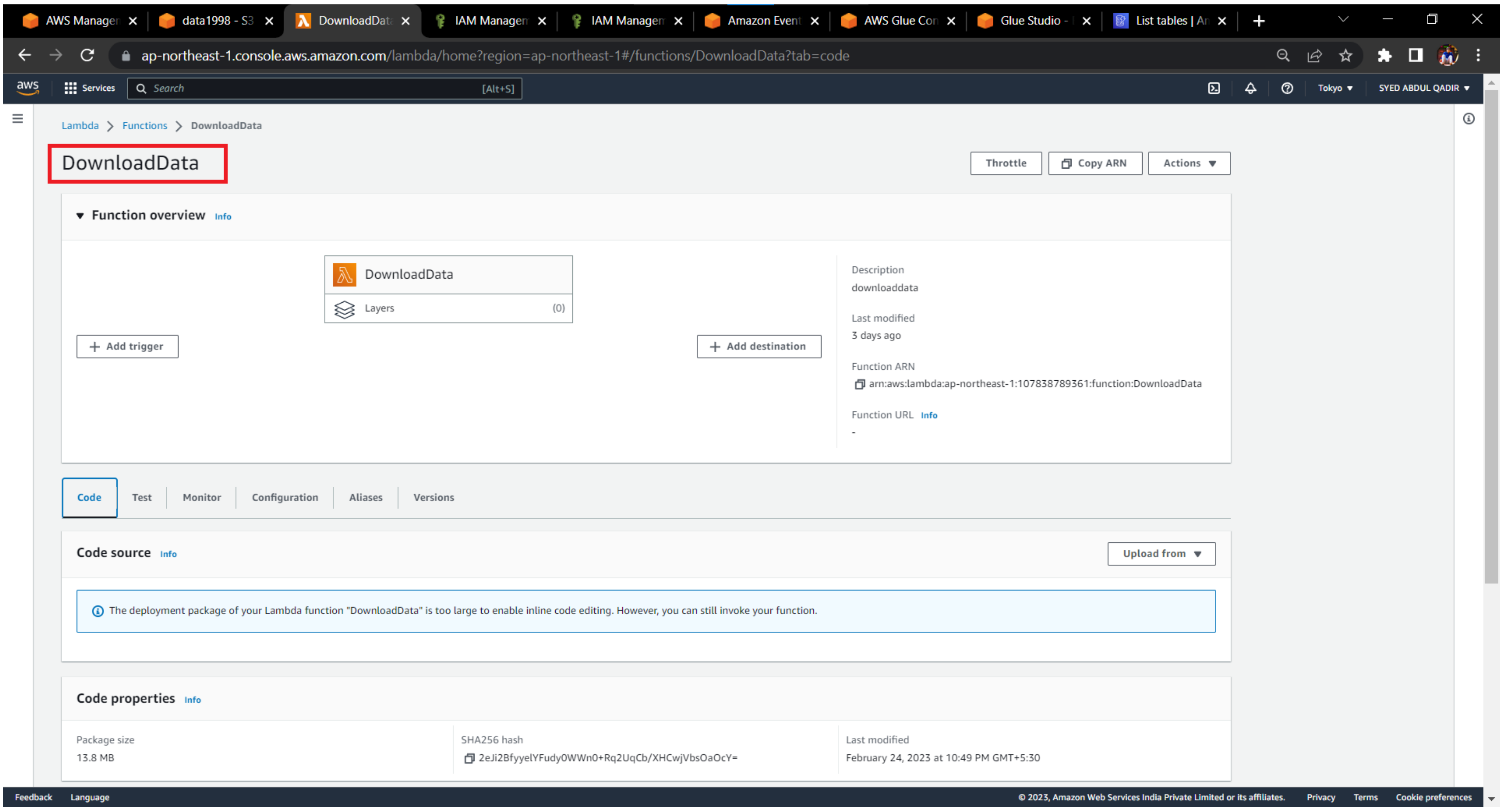
Permissions policies (6) Info
You can attach up to 10 managed policies.

Simulate | Remove | Add permissions

Filter policies by property or policy name and press enter

Policy name	Type	Description
AmazonS3FullAccess	AWS managed	Provides full access to all buckets via the AWS Management Console.
CloudWatchFullAccess	AWS managed	Provides full access to CloudWatch.
AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon DynamoDB via the AWS Management Console.
AWSGlueServiceRole	AWS managed	Policy for AWS Glue service role which allows access to related services including EC2, S3, and Cloudwatc...
AWSGlueConsoleFullAccess	AWS managed	Provides full access to AWS Glue via the AWS Management Console
CloudWatchEventsFullAccess	AWS managed	Provides full access to Amazon CloudWatch Events.

Configuring AWS Lambda with name **DownloadData ,** where we will be dumping api source data to mongo dB and to s3 and we will be filling configuration for environment variables as below.



ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/functions/DownloadData?tab=code

Lambda > Functions > DownloadData

DownloadData

Function overview Info

DownloadData
Layers (0)

Description
downloaddata

Last modified
3 days ago

Function ARN
arn:aws:lambda:ap-northeast-1:107838789361:function:DownloadData

Function URL Info
-

Code | Test | Monitor | Configuration | Aliases | Versions

Code source Info

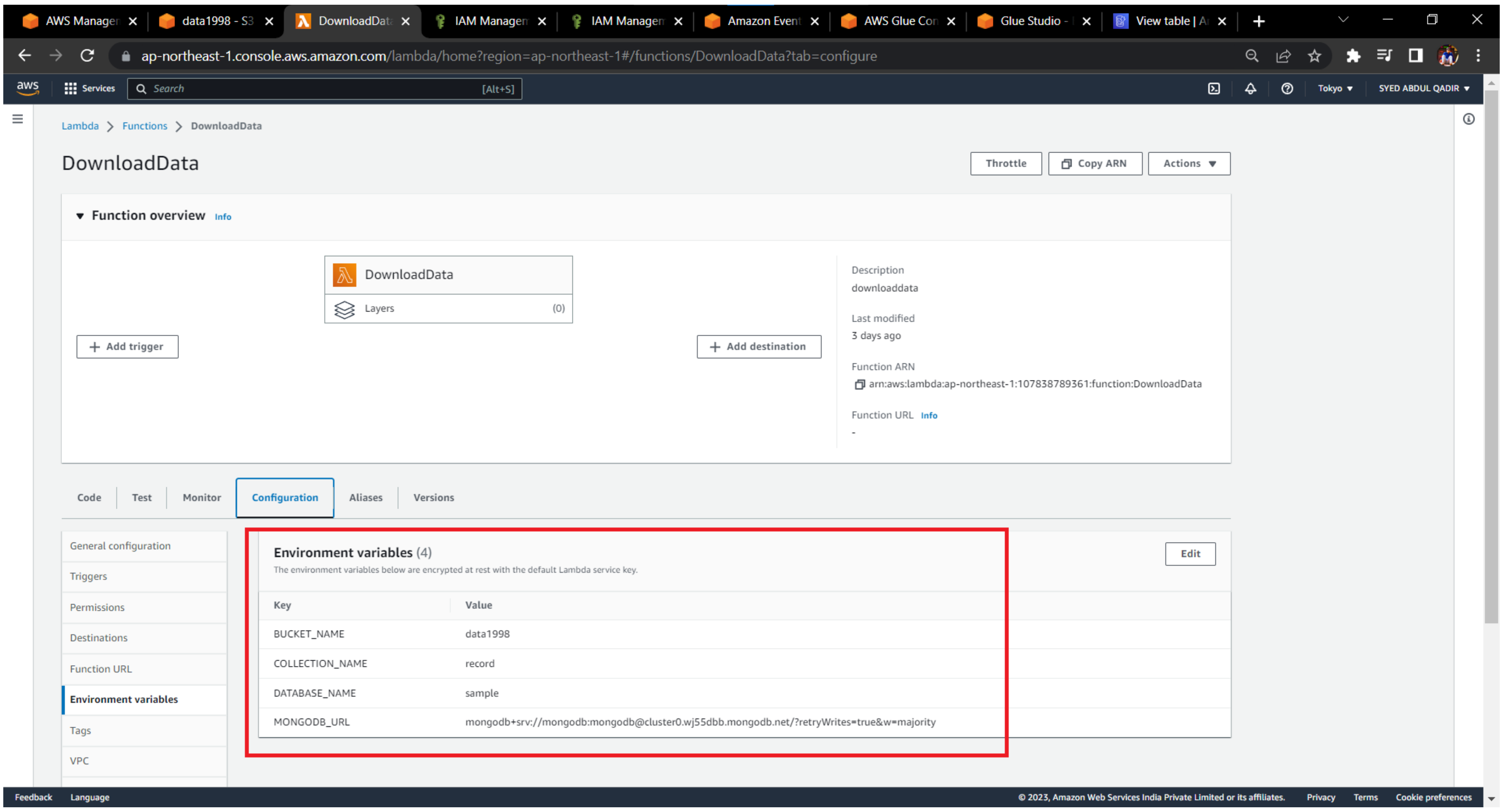
The deployment package of your Lambda function "DownloadData" is too large to enable inline code editing. However, you can still invoke your function.

Code properties Info

Package size
13.8 MB

SHA256 hash
2eJi2BfyyelYFudy0WWn0+Rq2UqCb/XHCwjVbsOaOcY=

Last modified
February 24, 2023 at 10:49 PM GMT+5:30



DownloadData

Description
downloaddata

Last modified
3 days ago

Function ARN
arn:aws:lambda:ap-northeast-1:107838789361:function:DownloadData

Function URL
-

Environment variables (4)

The environment variables below are encrypted at rest with the default Lambda service key.

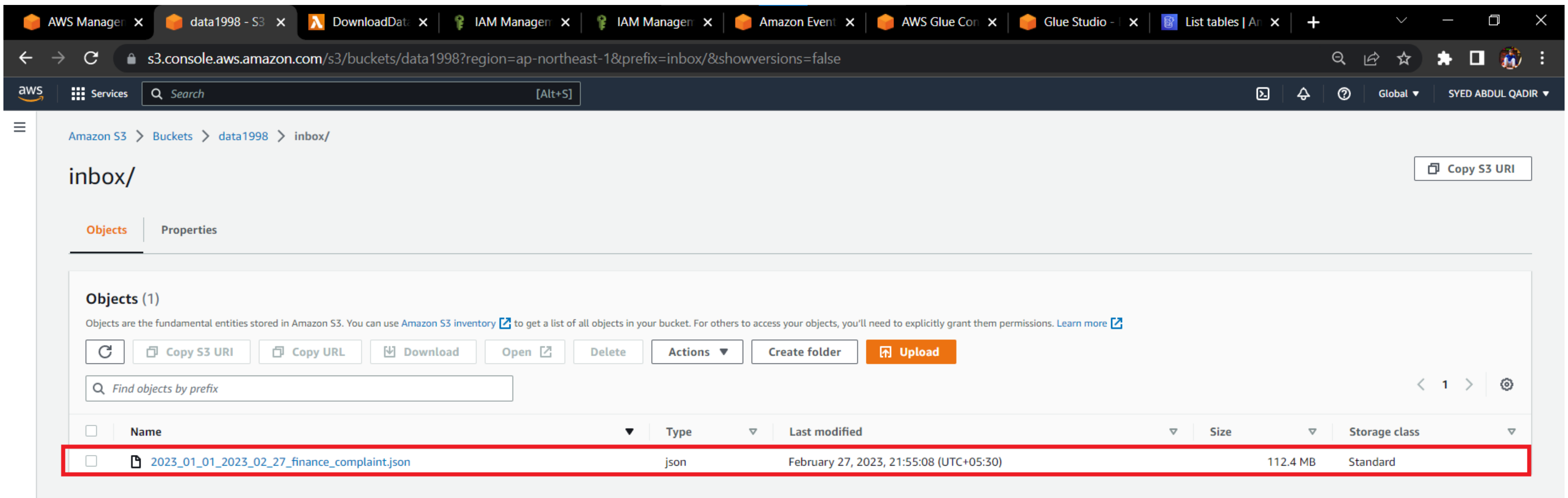
Key	Value
BUCKET_NAME	data1998
COLLECTION_NAME	record
DATABASE_NAME	sample
MONGODB_URL	mongodb+srv://mongodb:mongodb@cluster0.wj55dbb.mongodb.net/?retryWrites=true&w=majority

Lambda function code:

```python
import json
import pymongo
import certifi
import logging
import os
import boto3
import datetime
import os
import requests

      ca = certifi.where()
      import os
      DATABASE_NAME = os.getenv("DATABASE_NAME")
      COLLECTION_NAME = os.getenv("COLLECTION_NAME")
      MONGODB_URL = os.getenv("MONGODB_URL")
      BUCKET_NAME=os.getenv("BUCKET_NAME")

      DATA_SOURCE_URL = f"https://www.consumerfinance.gov/data-research/consumer-complaints/search/api/v1/" \
                        f"?date_received_max=<todate>&date_received_min=<fromdate>" \
                        f"&field=all&format=json"
      client = pymongo.MongoClient(MONGODB_URL, tlsCAFile=ca)

      def get_from_date_to_date():
          from_date = "2023-01-01"
          from_date = datetime.datetime.strptime(from_date, "%Y-%m-%d")

          if COLLECTION_NAME in client[DATABASE_NAME].list_collection_names():

              res = client[DATABASE_NAME][COLLECTION_NAME].find_one(sort=[("to_date", pymongo.DESCENDING)])
              if res is not None:
                  from_date = res["to_date"]

          to_date = datetime.datetime.now() #current date

          response = {
              "form_date": from_date.strftime("%Y-%m-%d"),
              "to_date": to_date.strftime("%Y-%m-%d"),
              "from_date_obj": from_date,
              "to_date_obj": to_date
          }
          logging.info(f"From date and to date {response}")
          return response

      def save_from_date_to_date(data, status=True):
          data.update({"status": status})
          logging.info(f"saving from data and to date {data}")
          client[DATABASE_NAME][COLLECTION_NAME].insert_one(data)

      def lambda_handler(event, context):
          print(event,context)
```

```
        from_date, to_date, from_date_obj, to_date_obj = get_from_date_to_date().values()
        if to_date==from_date:
            return {
                'statusCode': 200,
                'body': json.dumps('Pipeline has already downloaded all data upto yesterday')
            }
        url = DATA_SOURCE_URL.replace("<todate>", to_date).replace("<fromdate>", from_date)
        data = requests.get(url, params={'User-agent': f'your bot '})

        finance_complaint_data = list(map(lambda x: x["_source"],
                                        filter(lambda x: "_source" in x.keys(),
                                                json.loads(data.content)))
                                    )
        s3 = boto3.resource('s3')
        s3object = s3.Object(BUCKET_NAME, f"inbox/{from_date.replace('-','_')}_{to_date.replace('-','_')}_finance_complaint.json")
        s3object.put(
            Body=(bytes(json.dumps(finance_complaint_data).encode('UTF-8')))
        )

        save_from_date_to_date({"from_date": from_date_obj, "to_date": to_date_obj})
        return {
            'statusCode': 200,
            'body': json.dumps('Hello from Lambda!')
 }
```

Able to dump data to mongo Db and as well as s3 Bucket.



s3.console.aws.amazon.com/s3/buckets/data1998?region=ap-northeast-1&prefix=inbox/&showversions=false

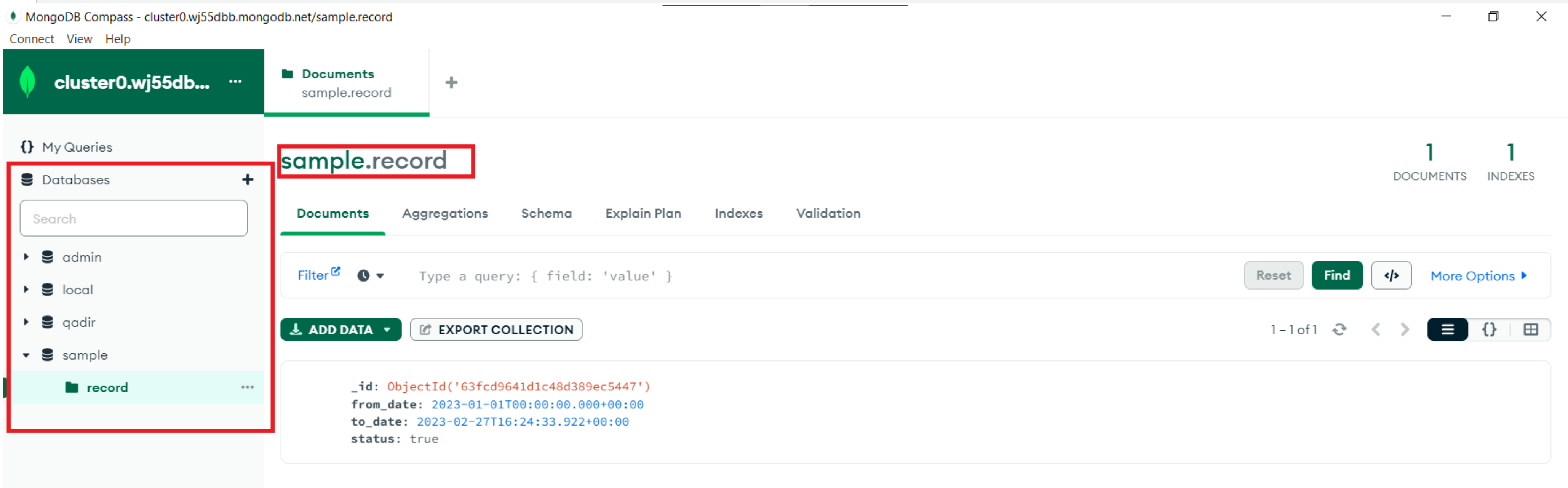
Amazon S3 > Buckets > data1998 > inbox/

inbox/

Objects (1)

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

Name	Type	Last modified	Size	Storage class
2023_01_01_2023_02_27_finance_complaint.json	json	February 27, 2023, 21:55:08 (UTC+05:30)	112.4 MB	Standard



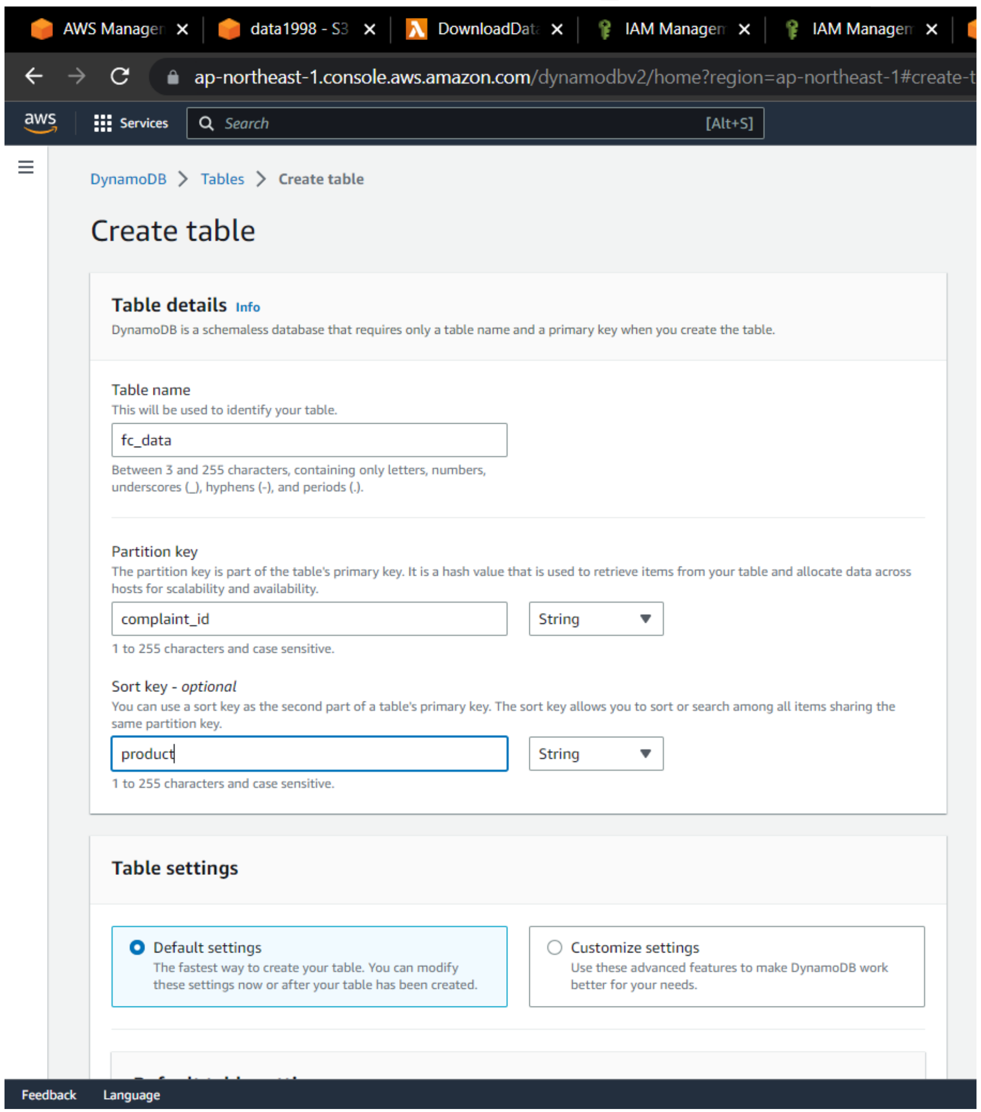
MongoDB Compass - cluster0.wj55dbb.mongodb.net/sample.record

sample.record

1 DOCUMENTS 1 INDEXES

```
_id: ObjectId('63fcd9641d1c48d389ec5447')
from_date: 2023-01-01T00:00:00.000+00:00
to_date: 2023-02-27T16:24:33.922+00:00
status: true
```

Creating DynamoDB table with name **fc_data** with Partition key **complaint_id** and sort key as **product**



DynamoDB > Tables > Create table

Create table

Table details Info

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.

fc_data

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.

complaint_id | 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.

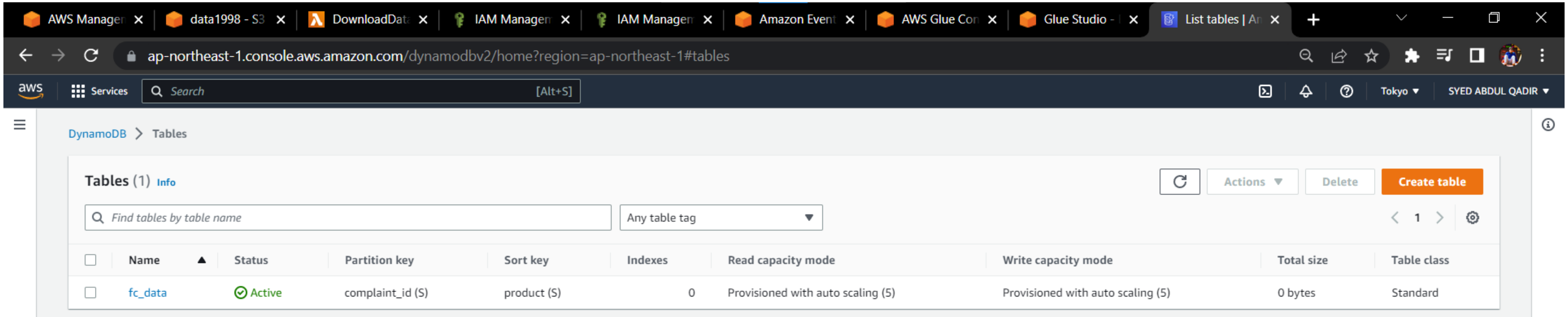
product | String

1 to 255 characters and case sensitive.

Table settings

● Default settings
The fastest way to create your table. You can modify these settings now or after your table has been created.

○ Customize settings
Use these advanced features to make DynamoDB work better for your needs.



DynamoDB > Tables

Tables (1) Info

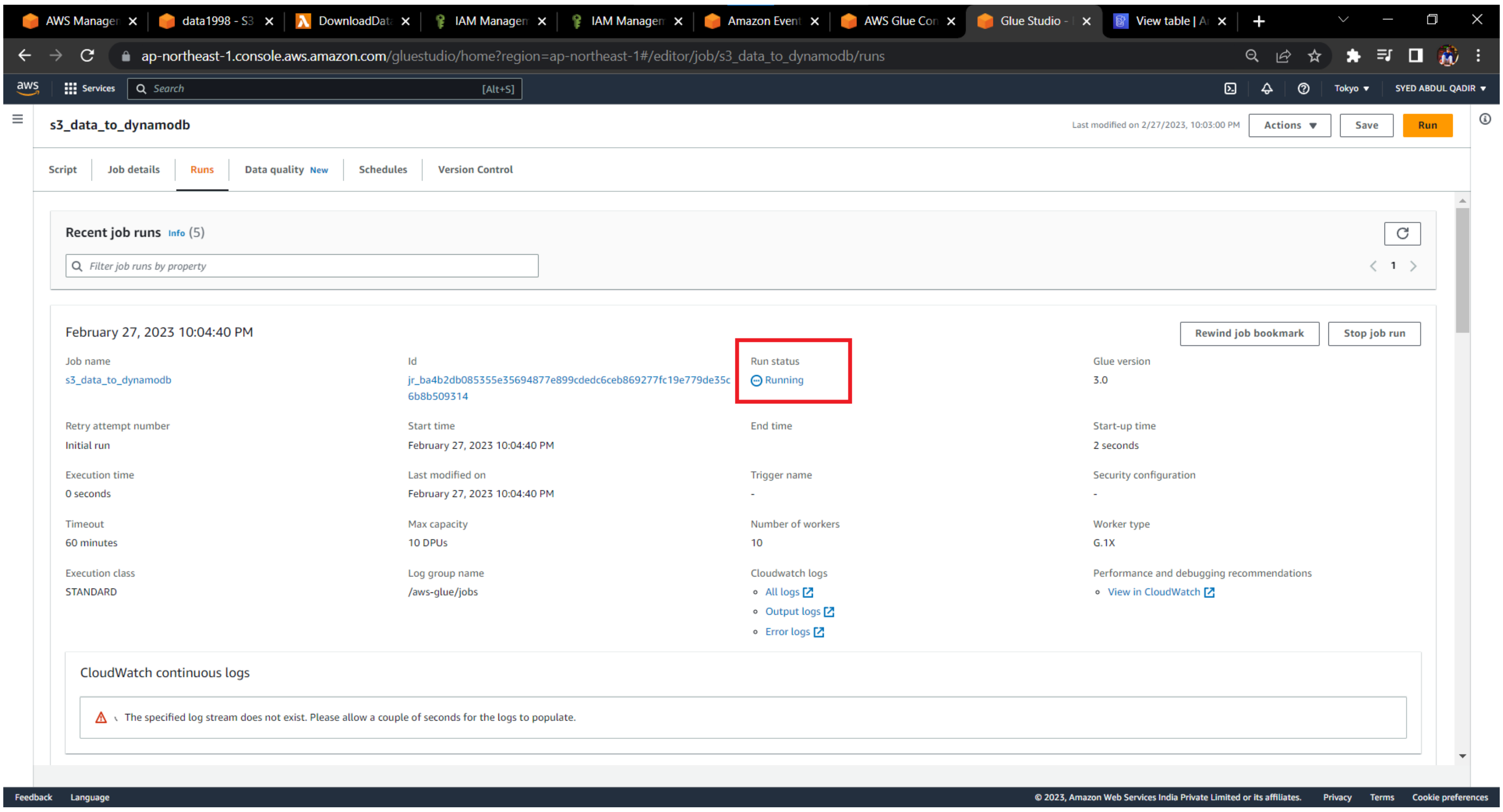
Find tables by table name | Any table tag

☐	Name ▲	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode	Total size	Table class
☐	fc_data	✓ Active	complaint_id (S)	product (S)	0	Provisioned with auto scaling (5)	Provisioned with auto scaling (5)	0 bytes	Standard

Glue script job as **s3_data_to_dynamodb** code:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from pyspark.sql import functions as func
from awsglue.context import GlueContext
from awsglue.dynamicframe import DynamicFrame
from pyspark.sql.types import LongType
from awsglue.job import Job
      import os
      ## @params: [JOB_NAME]
      args = getResolvedOptions(sys.argv, ['JOB_NAME'])

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

      #declaring constant variables
      BUCKET_NAME="data327030"
      DYNAMODB_TABLE_NAME="fc_data"
      INPUT_FILE_PATH=f"s3://{BUCKET_NAME}/inbox/*json"

      #getting logger object to log the progress
      logger  = glueContext.get_logger()
      logger.info(f"Started reading json file from {INPUT_FILE_PATH}")
      df_sparkdf=spark.read.json(INPUT_FILE_PATH)
      logger.info(f"Type casting columns of spark dataframe to Long type")
      df_sparkdf = df_sparkdf.withColumn("complaint_id",func.col("complaint_id").cast(LongType()))

      logger.info(f"Columns in dataframe : {len(df_sparkdf.columns)}--> {df_sparkdf.columns}")
      logger.info(f"Number of rows found in file: {df_sparkdf.count()} ")

      dyf = glueContext.create_dynamic_frame.from_options(
          connection_type="dynamodb",
          connection_options={"dynamodb.input.tableName": DYNAMODB_TABLE_NAME,
              "dynamodb.throughput.read.percent": "1.0",
              "dynamodb.splits": "100"
          }
      )
      dyf_sparkdf=dyf.toDF()
      new_sparkdf=None
      if dyf_sparkdf.count()!=0:
          logger.info(f"Columns in dynamodb dataframe : {len(dyf_sparkdf.columns)}--> {dyf_sparkdf.columns}")
          logger.info(f"Number of rows found in file: {dyf_sparkdf.count()} ")
          logger.info(f"Renaming exiting complaint id column of dynamodb ")
          existing_complaint_spark_df = dyf_sparkdf.select("complaint_id").withColumnRenamed("complaint_id","existing_complaint_id")
          logger.info(f"Applying left join on new dataframe from s3 and dynamo db ")
          joined_sparkdf = df_sparkdf.join(existing_complaint_spark_df,df_sparkdf.complaint_id==existing_complaint_spark_df.existing_complaint_id,"left")
          logger.info(f"Number of row after left join : {joined_sparkdf.count()}")
          new_sparkdf = joined_sparkdf.filter("existing_complaint_id is null")
          new_sparkdf.drop("existing_complaint_id")
          new_sparkdf=new_sparkdf.coalesce(10)
      else:
          new_sparkdf=df_sparkdf.coalesce(10)

      logger.info(f"Converting spark dataframe to DynamicFrame")
      newDynamicFrame= DynamicFrame.fromDF(new_sparkdf, glueContext, "new_sparkdf")
      logger.info(f"Started writing new records into dynamo db dataframe.")
      logger.info(f"Number of records will be written to dynamodb: {new_sparkdf.count()}")
      glueContext.write_dynamic_frame_from_options(
          frame=newDynamicFrame,
          connection_type="dynamodb",
          connection_options={"dynamodb.output.tableName": DYNAMODB_TABLE_NAME,
              "dynamodb.throughput.write.percent": "1.0"
          }
      )

      logger.info(f"Data has been dumped into dynamodb ")
      logger.info(f"Archiving file from inbox source: s3://{BUCKET_NAME}/inbox  to archive: s3://{BUCKET_NAME}/archive ")
      os.system(f"aws s3 sync s3://{BUCKET_NAME}/inbox s3://{BUCKET_NAME}/archive")

      logger.info(f"File is successfully archived.")
      os.system(f"aws s3 rm s3://{BUCKET_NAME}/inbox/ --recursive")

job.commit()
```

Running glue job below



s3_data_to_dynamodb

Last modified on 2/27/2023, 10:03:00 PM

Recent job runs (5)

February 27, 2023 10:04:40 PM

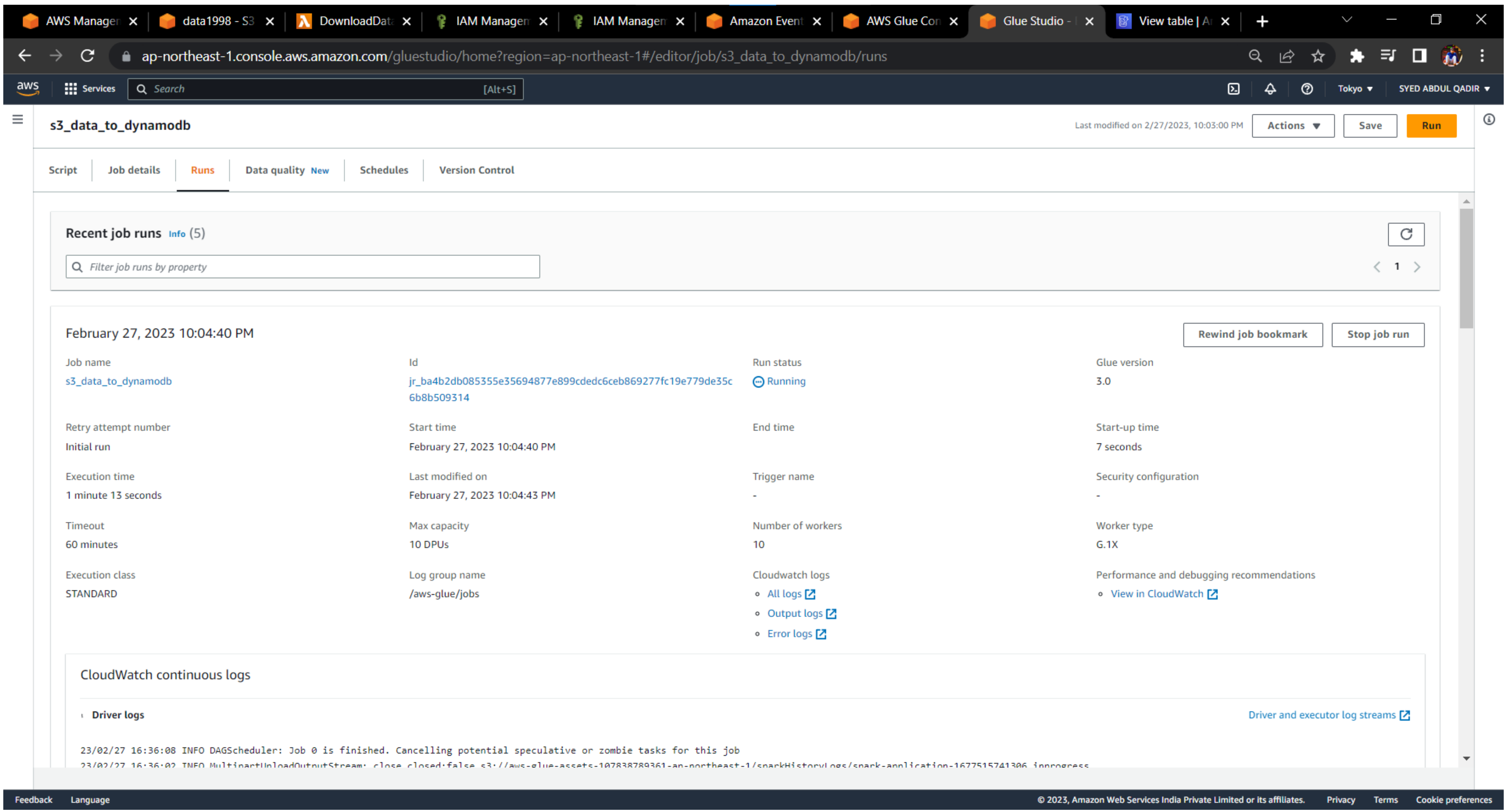
Job name: s3_data_to_dynamodb
Id: jr_ba4b2db085355e35694877e899cdedc6ceb869277fc19e779de35c6b8b509314
Run status: Running
Glue version: 3.0
Retry attempt number: Initial run
Start time: February 27, 2023 10:04:40 PM
End time:
Start-up time: 2 seconds
Execution time: 0 seconds
Last modified on: February 27, 2023 10:04:40 PM
Trigger name: -
Security configuration: -
Timeout: 60 minutes
Max capacity: 10 DPUs
Number of workers: 10
Worker type: G.1X
Execution class: STANDARD
Log group name: /aws-glue/jobs
Cloudwatch logs
- All logs
- Output logs
- Error logs

Performance and debugging recommendations
- View in CloudWatch

CloudWatch continuous logs

⚠ The specified log stream does not exist. Please allow a couple of seconds for the logs to populate.

© 2023, Amazon Web Services India Private Limited or its affiliates.



s3_data_to_dynamodb

Recent job runs (5)

February 27, 2023 10:04:40 PM

Job name: s3_data_to_dynamodb

Id: jr_ba4b2db085355e35694877e899cdedc6ceb869277fc19e779de35c6b8b509314

Run status: Running

Glue version: 3.0

Retry attempt number: Initial run

Start time: February 27, 2023 10:04:40 PM

End time:

Start-up time: 7 seconds

Execution time: 1 minute 13 seconds

Last modified on: February 27, 2023 10:04:43 PM

Trigger name: -

Security configuration: -

Timeout: 60 minutes

Max capacity: 10 DPUs

Number of workers: 10

Worker type: G.1X

Execution class: STANDARD

Log group name: /aws-glue/jobs

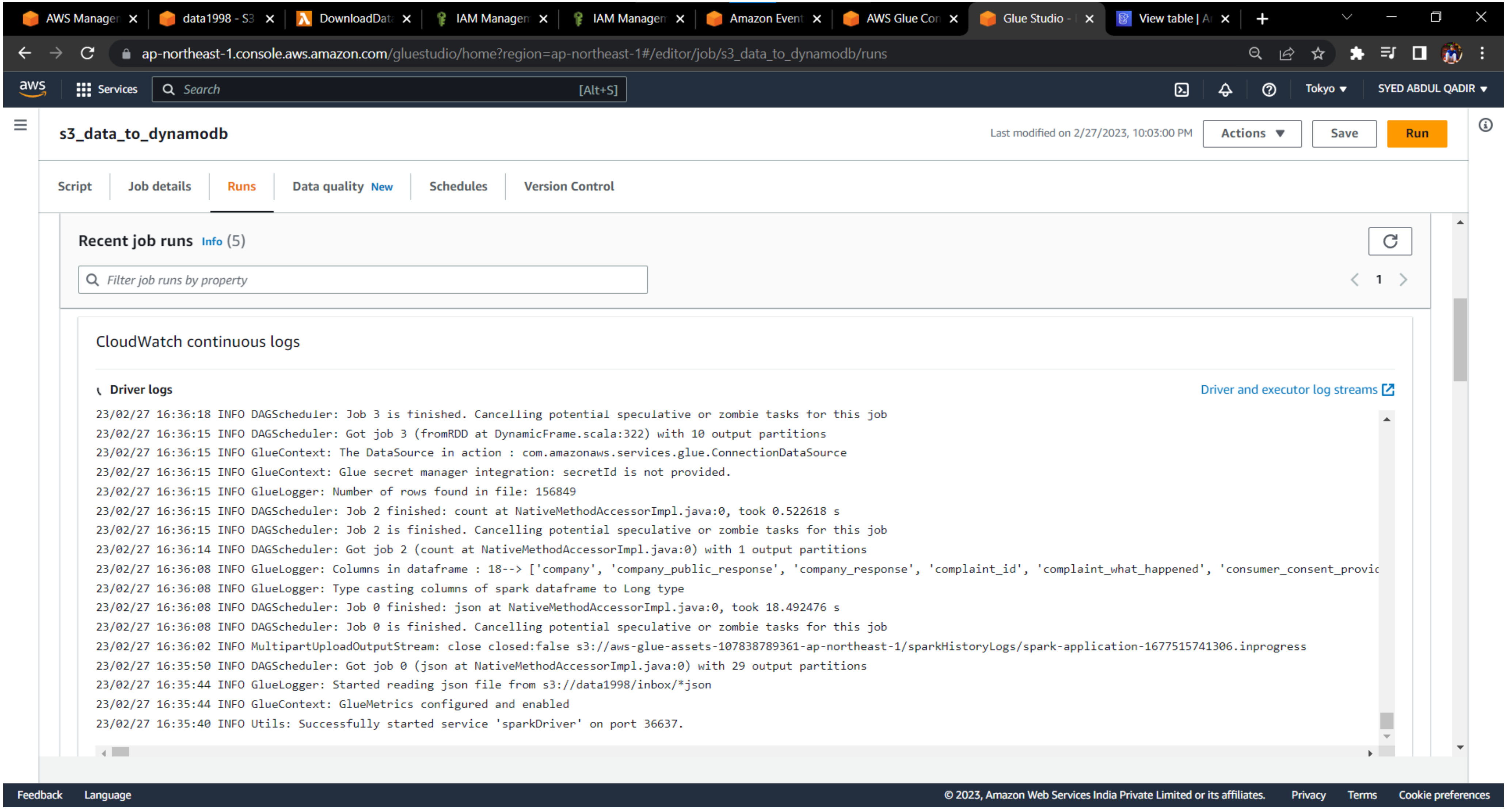
Cloudwatch logs:
- All logs
- Output logs
- Error logs

Performance and debugging recommendations:
- View in CloudWatch

CloudWatch continuous logs

Driver logs — Driver and executor log streams

```
23/02/27 16:36:08 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
23/02/27 16:36:02 INFO MultipartUploadOutputStream: close closed:false s3://aws-glue-assets-107838789361-ap-northeast-1/sparkHistoryLogs/spark-application-1677515741306.inprogress
```



s3_data_to_dynamodb

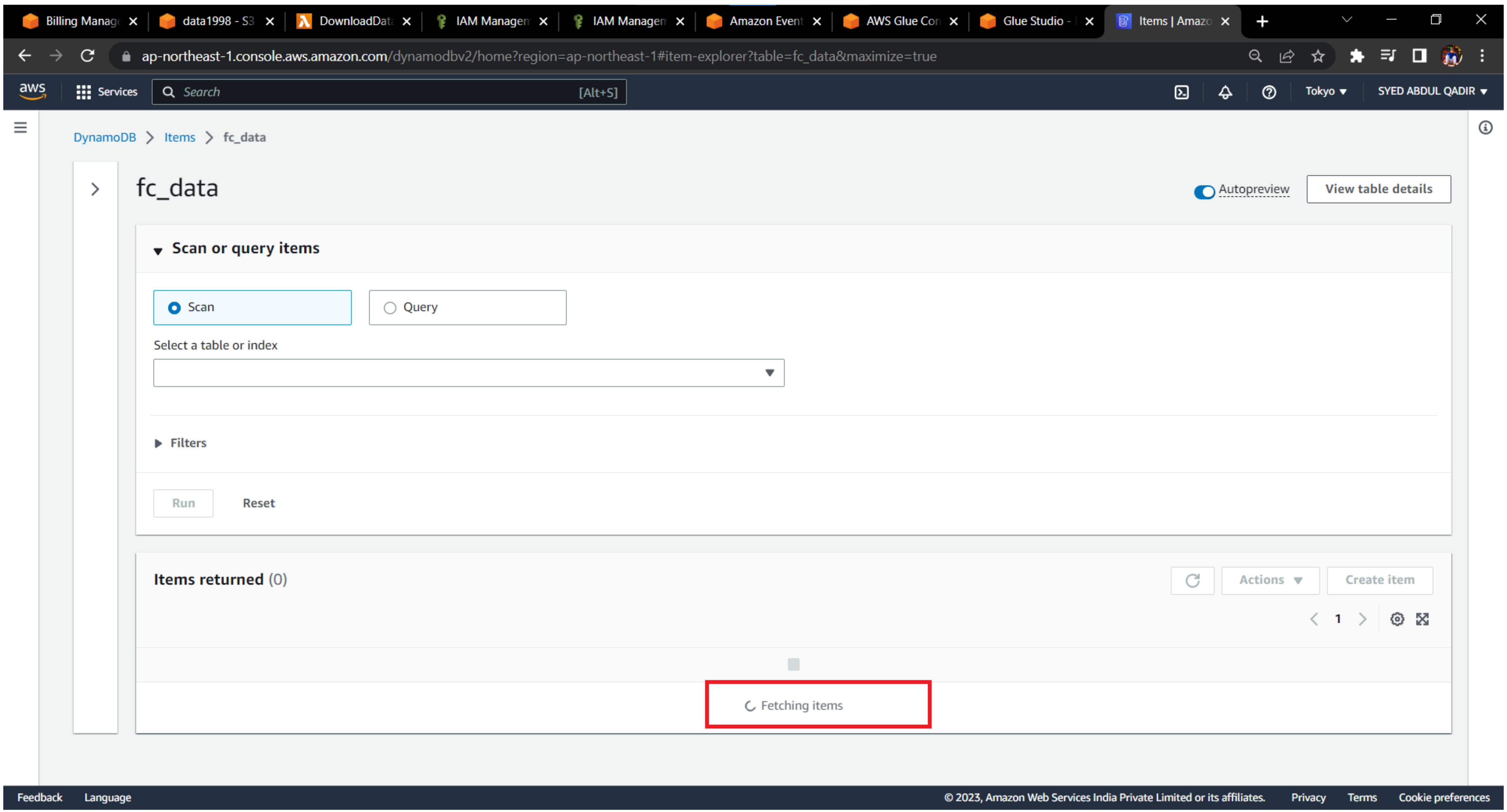
Recent job runs (5)

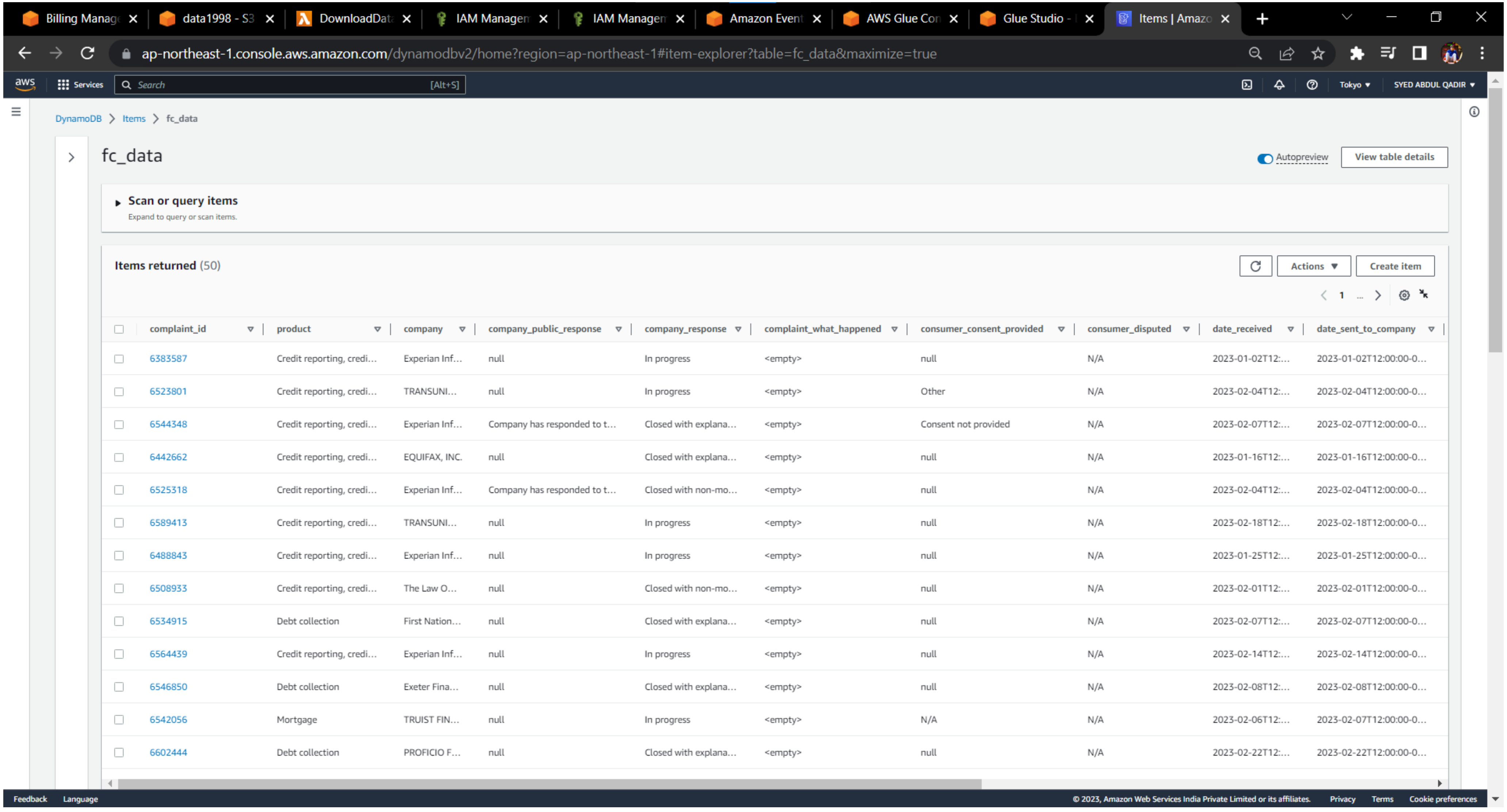
CloudWatch continuous logs

Driver logs — Driver and executor log streams

```
23/02/27 16:36:18 INFO DAGScheduler: Job 3 is finished. Cancelling potential speculative or zombie tasks for this job
23/02/27 16:36:15 INFO DAGScheduler: Got job 3 (fromRDD at DynamicFrame.scala:322) with 10 output partitions
23/02/27 16:36:15 INFO GlueContext: The DataSource in action : com.amazonaws.services.glue.ConnectionDataSource
23/02/27 16:36:15 INFO GlueContext: Glue secret manager integration: secretId is not provided.
23/02/27 16:36:15 INFO GlueLogger: Number of rows found in file: 156849
23/02/27 16:36:15 INFO DAGScheduler: Job 2 finished: count at NativeMethodAccessorImpl.java:0, took 0.522618 s
23/02/27 16:36:15 INFO DAGScheduler: Job 2 is finished. Cancelling potential speculative or zombie tasks for this job
23/02/27 16:36:14 INFO DAGScheduler: Got job 2 (count at NativeMethodAccessorImpl.java:0) with 1 output partitions
23/02/27 16:36:08 INFO GlueLogger: Columns in dataframe : 18--> ['company', 'company_public_response', 'company_response', 'complaint_id', 'complaint_what_happened', 'consumer_consent_provic
23/02/27 16:36:08 INFO GlueLogger: Type casting columns of spark dataframe to Long type
23/02/27 16:36:08 INFO DAGScheduler: Job 0 finished: json at NativeMethodAccessorImpl.java:0, took 18.492476 s
23/02/27 16:36:08 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
23/02/27 16:36:02 INFO MultipartUploadOutputStream: close closed:false s3://aws-glue-assets-107838789361-ap-northeast-1/sparkHistoryLogs/spark-application-1677515741306.inprogress
23/02/27 16:35:50 INFO DAGScheduler: Got job 0 (json at NativeMethodAccessorImpl.java:0) with 29 output partitions
23/02/27 16:35:44 INFO GlueLogger: Started reading json file from s3://data1998/inbox/*json
23/02/27 16:35:44 INFO GlueContext: GlueMetrics configured and enabled
23/02/27 16:35:40 INFO Utils: Successfully started service 'sparkDriver' on port 36637.
```

Finally able to dump data form s3 to Dynamo DB.





Items returned (50)

complaint_id	product	company	company_public_response	company_response	complaint_what_happened	consumer_consent_provided	consumer_disputed	date_received	date_sent_to_company
6383587	Credit reporting, credi...	Experian Inf...	null	In progress	<empty>	null	N/A	2023-01-02T12:...	2023-01-02T12:00:00-0...
6523801	Credit reporting, credi...	TRANSUNI...	null	In progress	<empty>	Other	N/A	2023-02-04T12:...	2023-02-04T12:00:00-0...
6544348	Credit reporting, credi...	Experian Inf...	Company has responded to t...	Closed with explana...	<empty>	Consent not provided	N/A	2023-02-07T12:...	2023-02-07T12:00:00-0...
6442662	Credit reporting, credi...	EQUIFAX, INC.	null	Closed with explana...	<empty>	null	N/A	2023-01-16T12:...	2023-01-16T12:00:00-0...
6525318	Credit reporting, credi...	Experian Inf...	Company has responded to t...	Closed with non-mo...	<empty>	null	N/A	2023-02-04T12:...	2023-02-04T12:00:00-0...
6589413	Credit reporting, credi...	TRANSUNI...	null	In progress	<empty>	null	N/A	2023-02-18T12:...	2023-02-18T12:00:00-0...
6488843	Credit reporting, credi...	Experian Inf...	null	In progress	<empty>	null	N/A	2023-01-25T12:...	2023-01-25T12:00:00-0...
6508933	Credit reporting, credi...	The Law O...	null	Closed with non-mo...	<empty>	null	N/A	2023-02-01T12:...	2023-02-01T12:00:00-0...
6534915	Debt collection	First Nation...	null	Closed with explana...	<empty>	null	N/A	2023-02-07T12:...	2023-02-07T12:00:00-0...
6564439	Credit reporting, credi...	Experian Inf...	null	In progress	<empty>	null	N/A	2023-02-14T12:...	2023-02-14T12:00:00-0...
6546850	Debt collection	Exeter Fina...	null	Closed with explana...	<empty>	null	N/A	2023-02-08T12:...	2023-02-08T12:00:00-0...
6542056	Mortgage	TRUIST FIN...	null	In progress	<empty>	N/A	N/A	2023-02-06T12:...	2023-02-07T12:00:00-0...
6602444	Debt collection	PROFICIO F...	null	Closed with explana...	<empty>	null	N/A	2023-02-22T12:...	2023-02-22T12:00:00-0...