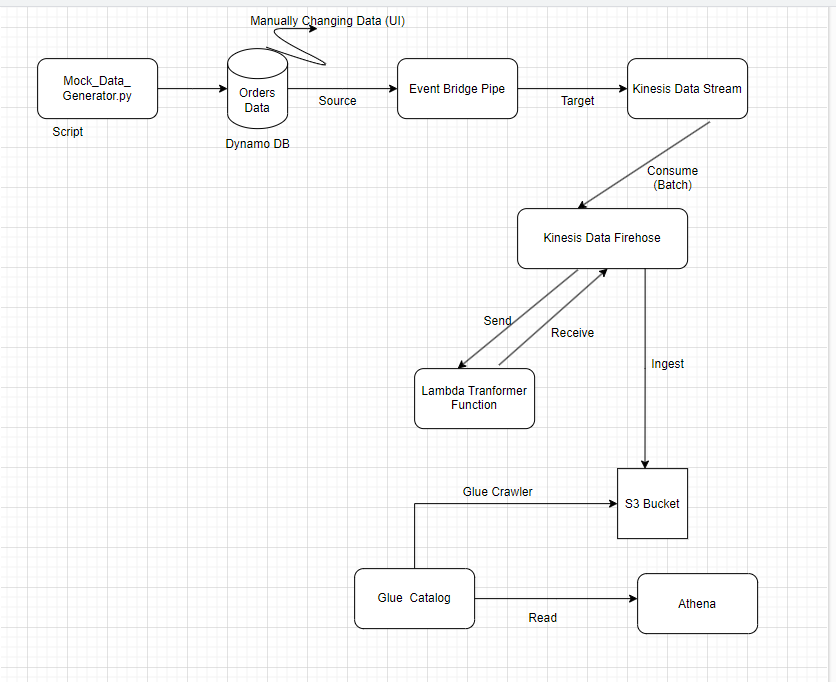
**CDC Data Pipeline for Sales Data Analysis**

**PROJECT :**

Designed and implemented a Change Data Capture (CDC) pipeline for sales data analysis. The pipeline ingests random order data generated by a mock script into DynamoDB via EventBridge. Kinesis streams capture CDC events from DynamoDB, and a Kinesis Firehose batch-processes and transforms the data using Lambda. The transformed data is then stored in S3, and Glue Crawler collects metadata. Finally, Athena is connected to query the data.  
  
**Architecture of Project**:  
  


🡪First will write python mock script to generate Data i.e Order data every time we run this code it will generate random order data, next ingesting this data to dynamodb through function ( insert\_into\_ dynaomdb ) .This function used to make API calls such as get ,put



**Mock generator python script for generating Random Orders**

import boto3

import random

import time

from decimal import Decimal

# Initialize the DynamoDB resource

# dynamodb = boto3.resource('dynamodb')

session = boto3.Session(profile\_name='default', region\_name='us-east-1')

dynamodb = session.resource('dynamodb')

table = dynamodb.Table('orders\_raw\_table')

def generate\_order\_data():

    """Generate random order data."""

    orderid = str(random.randint(1, 10000))  # Random order ID between 1 and 10000

    product\_name = random.choice(['Laptop', 'Phone', 'Tablet', 'Headphones', 'Charger'])

    quantity = random.randint(1, 5)

    price = Decimal(str(round(random.uniform(10.0, 500.0), 2)))

    return {

        'orderid': orderid,

        'product\_name': product\_name,

        'quantity': quantity,

        'price': price

    }

def insert\_into\_dynamodb(data):

    """Insert data into DynamoDB."""

    try:

        table.put\_item(Item=data)

        print(f"Inserted data: {data}")

    except Exception as e:

        print(f"Error inserting data: {str(e)}")

if \_\_name\_\_ == '\_\_main\_\_':

    try:

        while True:

            data = generate\_order\_data()

            insert\_into\_dynamodb(data)

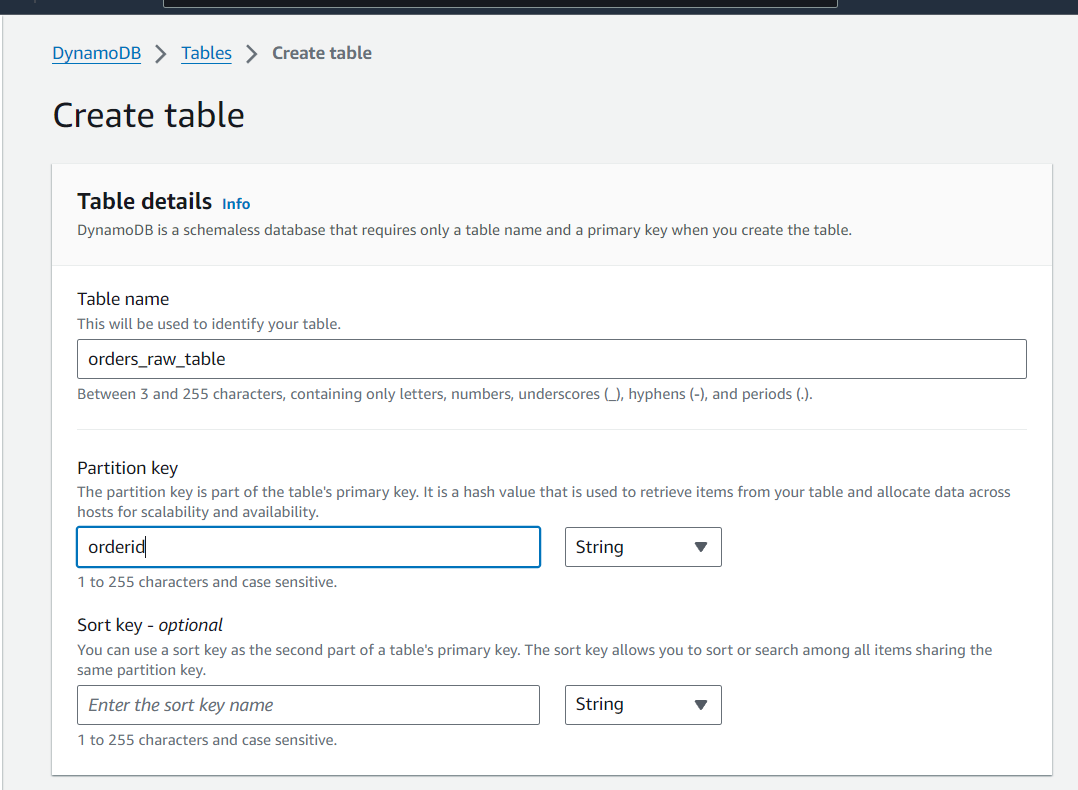
            time.sleep(1)  # Sleep for 10 seconds

    except KeyboardInterrupt:

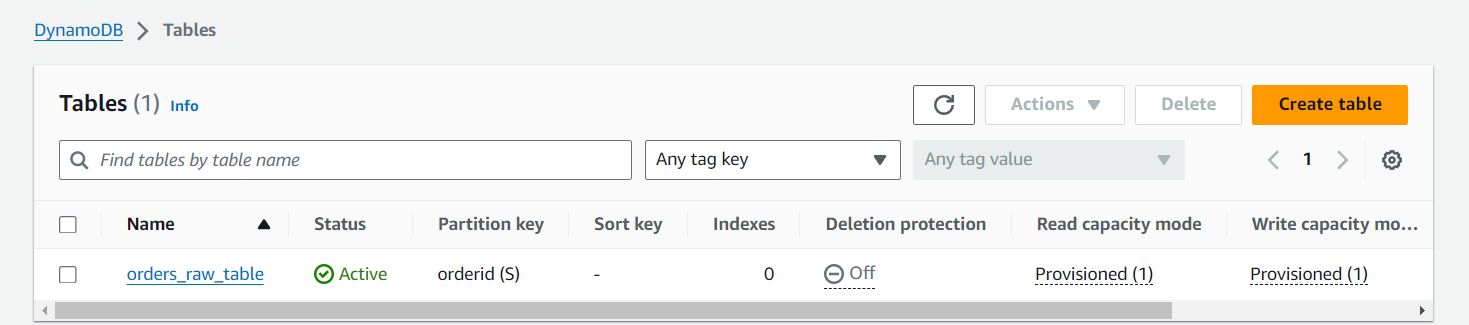
        print("\nScript stopped by manual intervention!")

**Next**

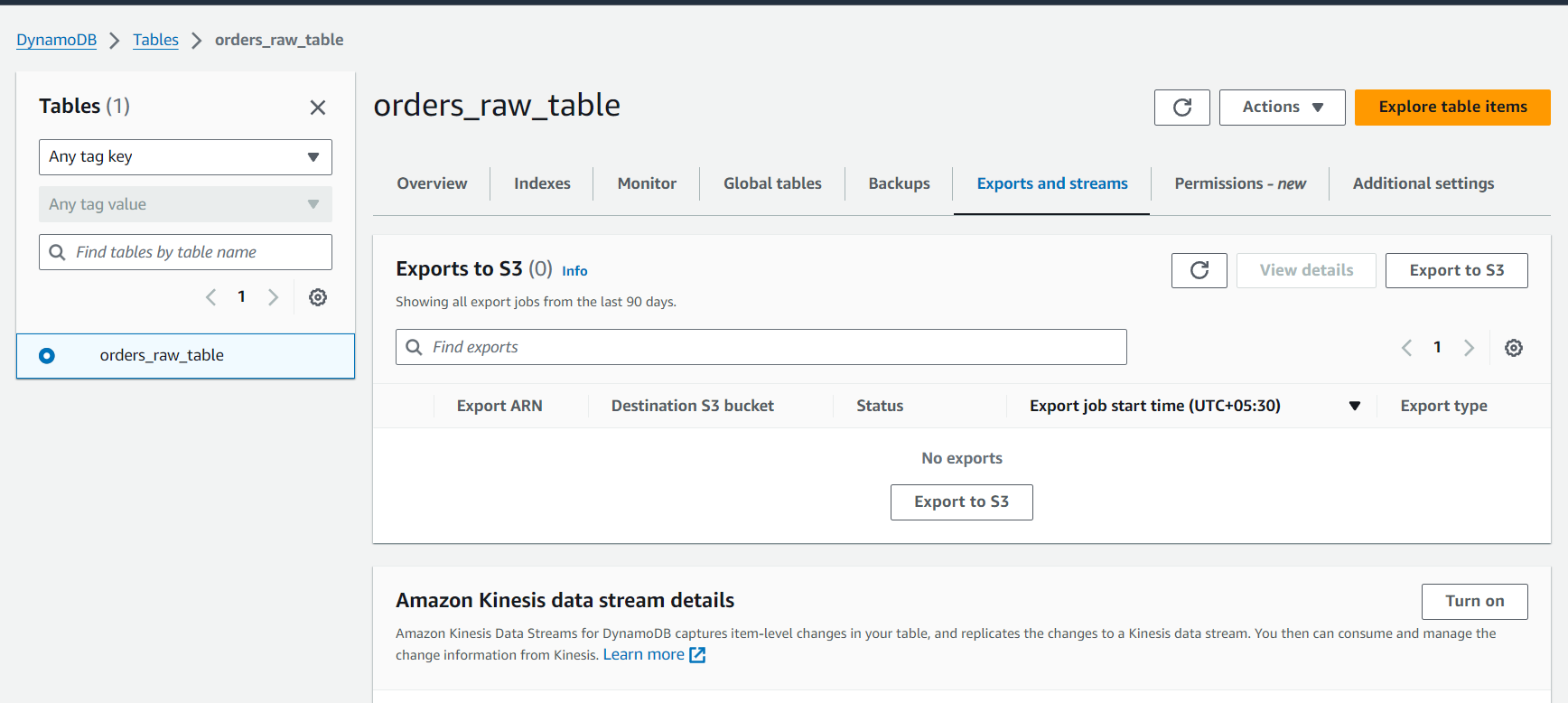
First we will create Dynamo DB table



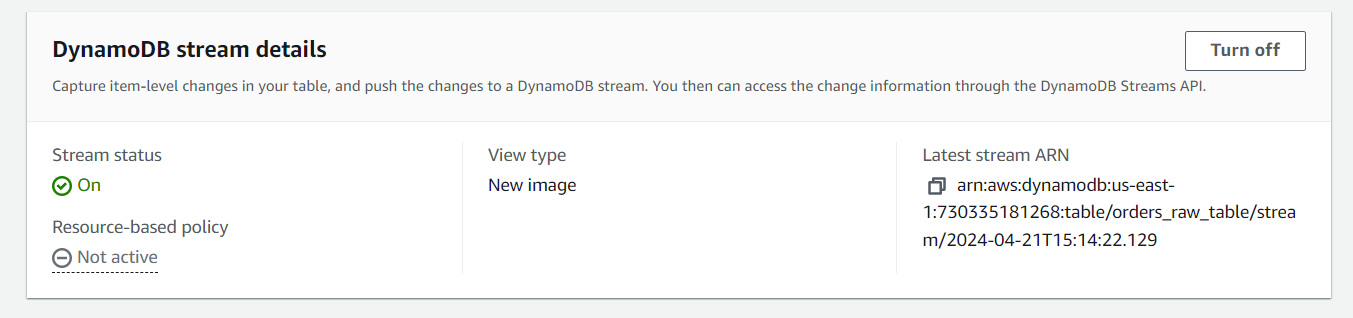
We created



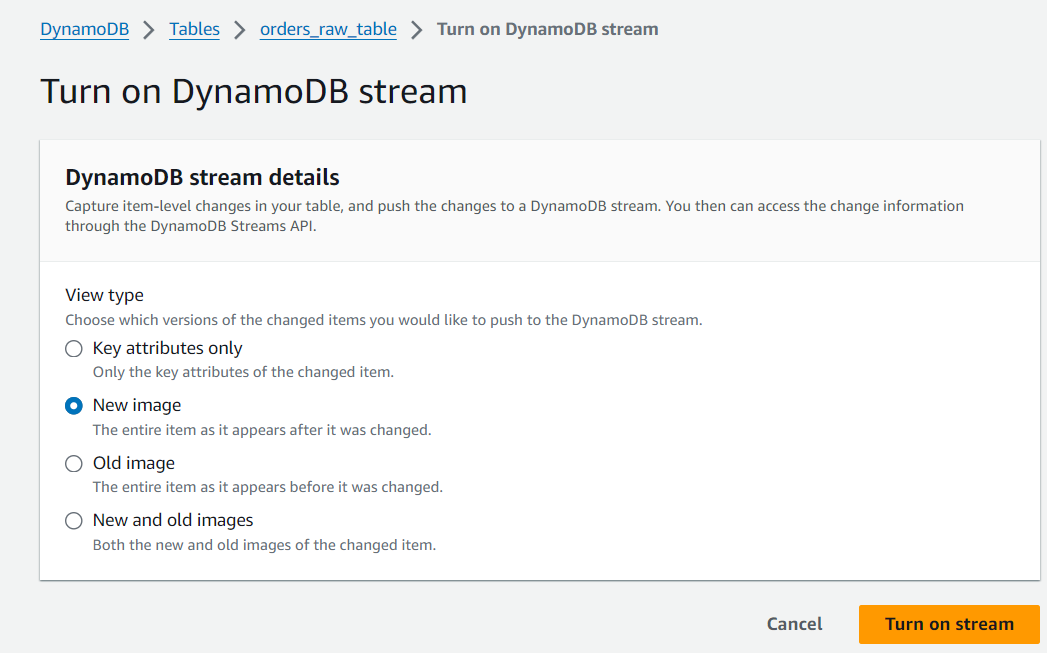
Next open the table go below and turn on DynamoDB stream details



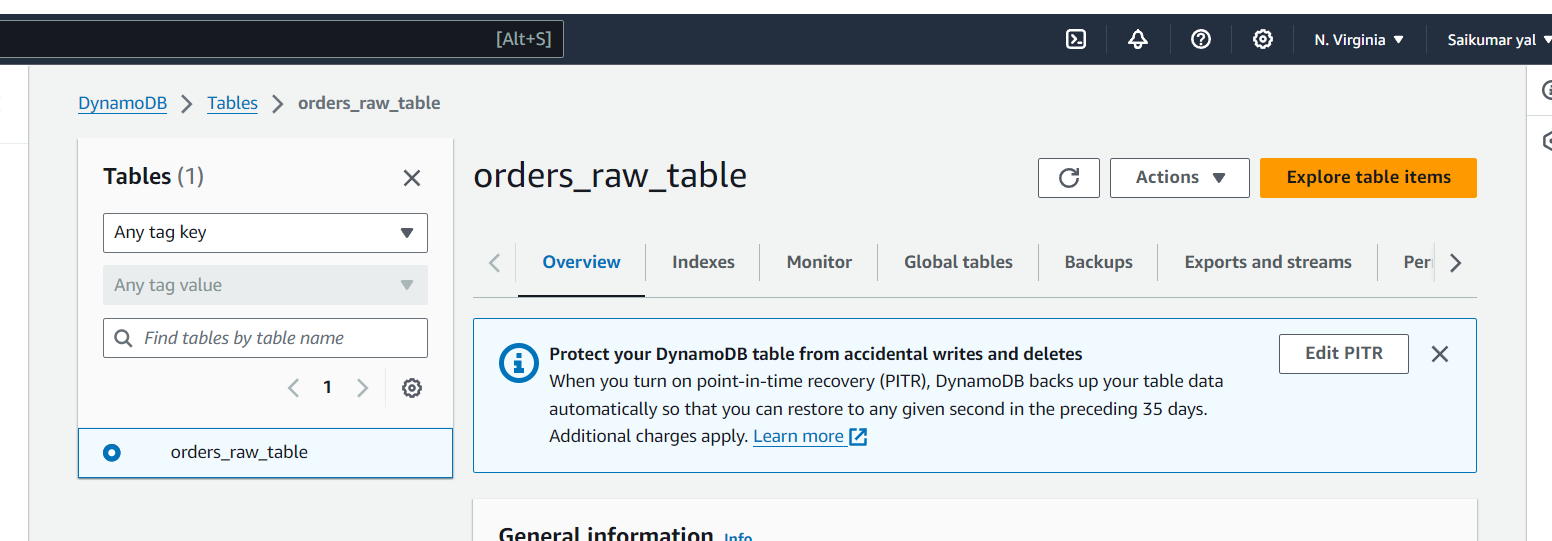
Go below



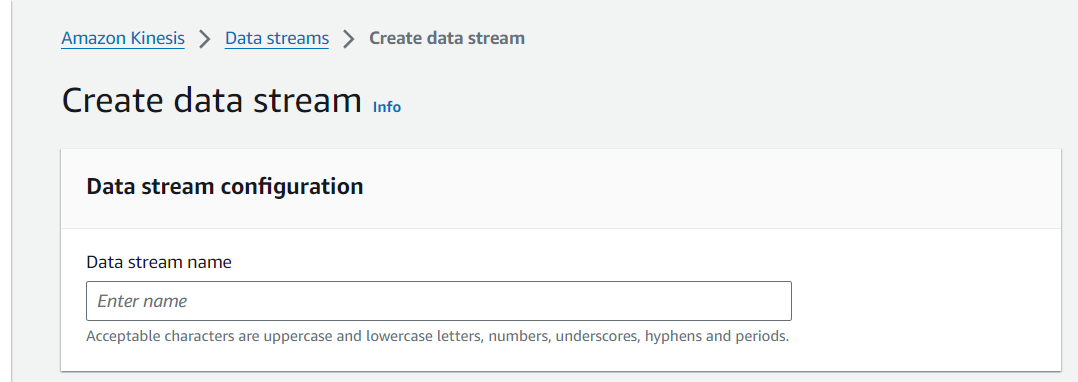
Here select new image in the streaming and Turn on stream.

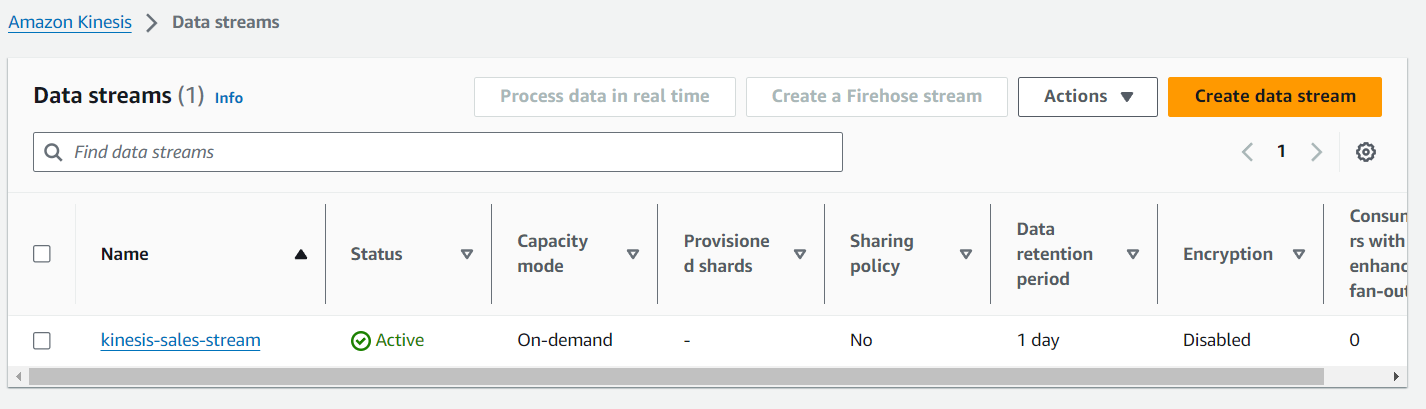


Next click on Explore Table items go below and just see



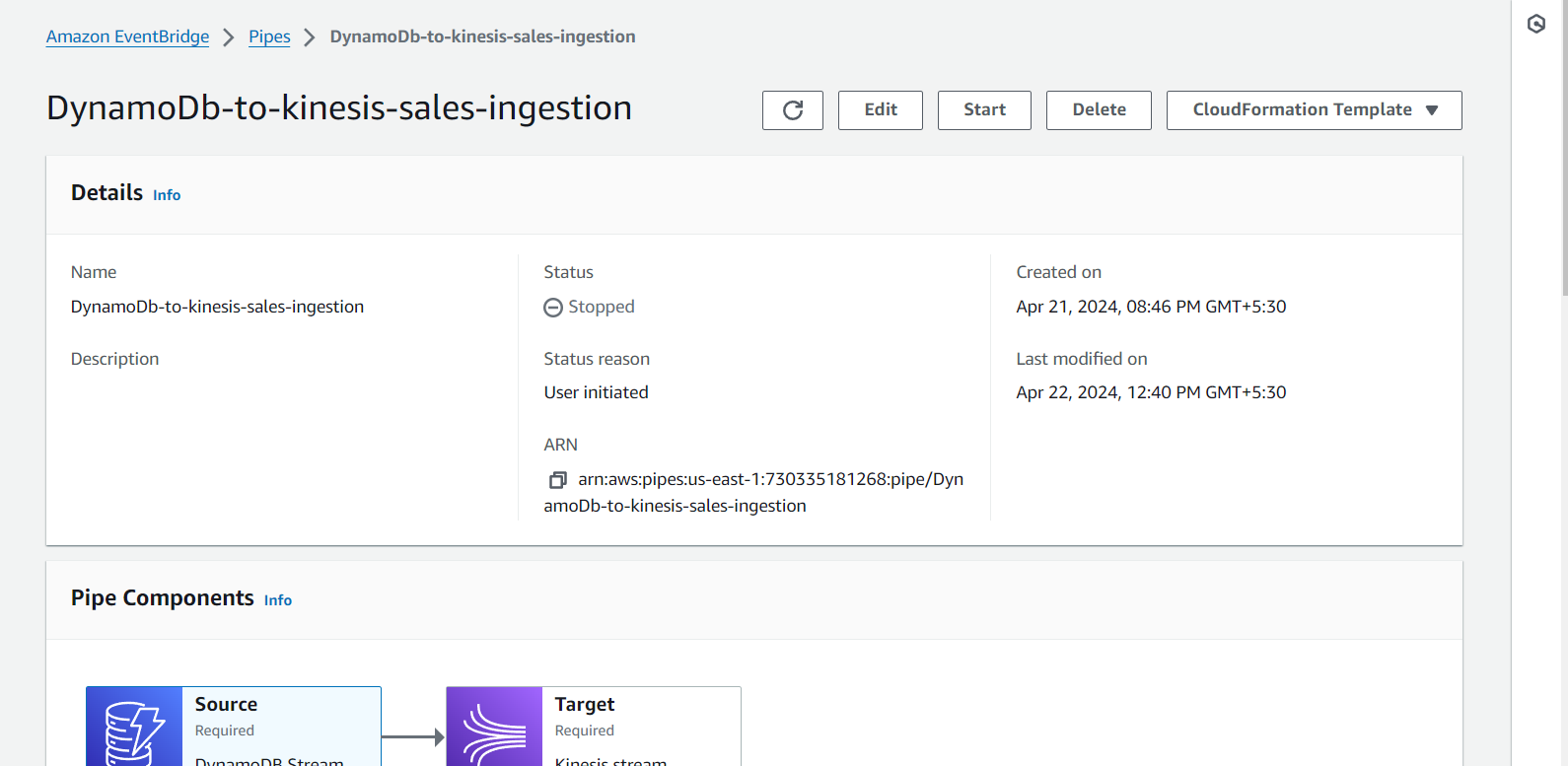
Next Open Kinesis Data streams and Click 🡪create Data stream

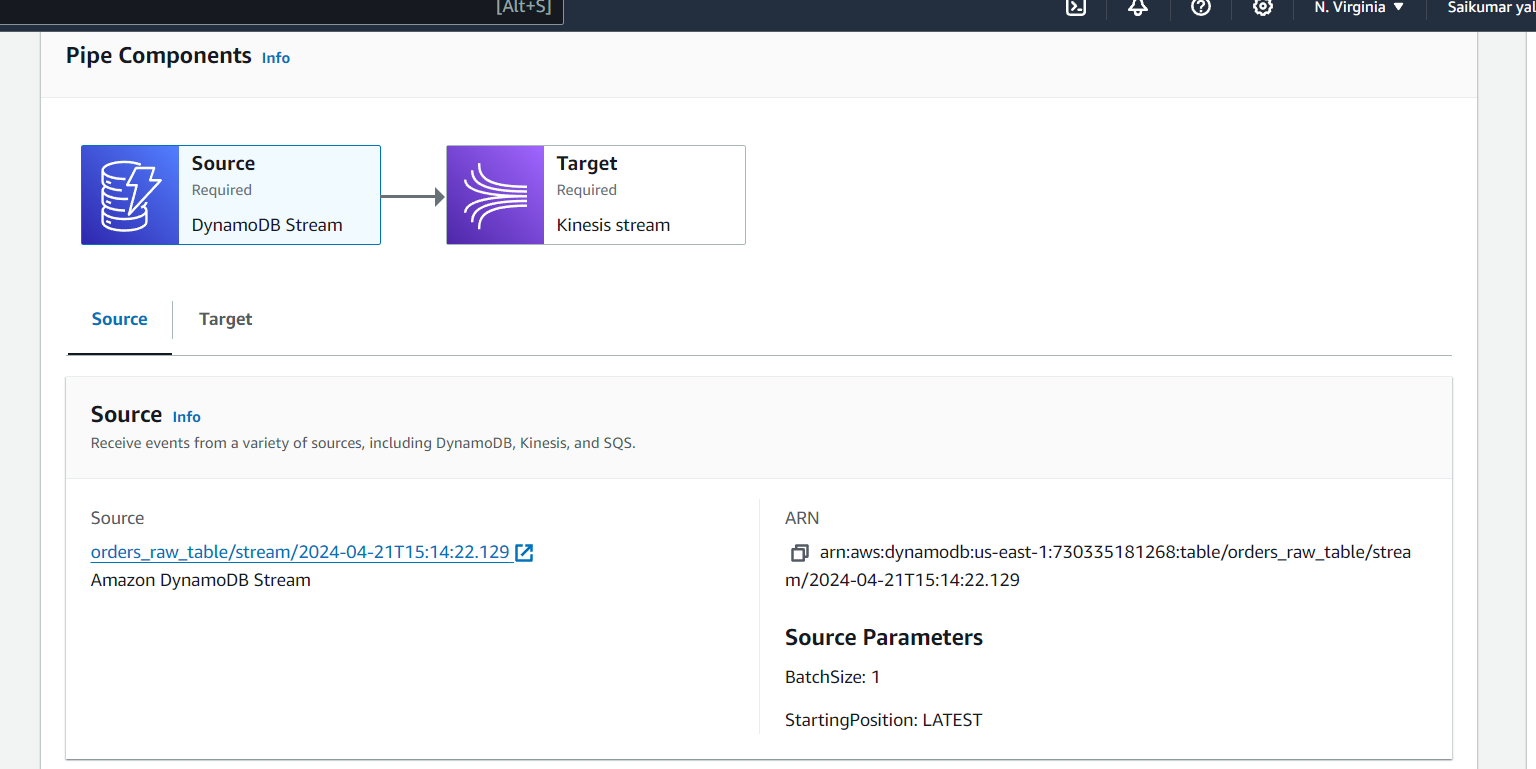




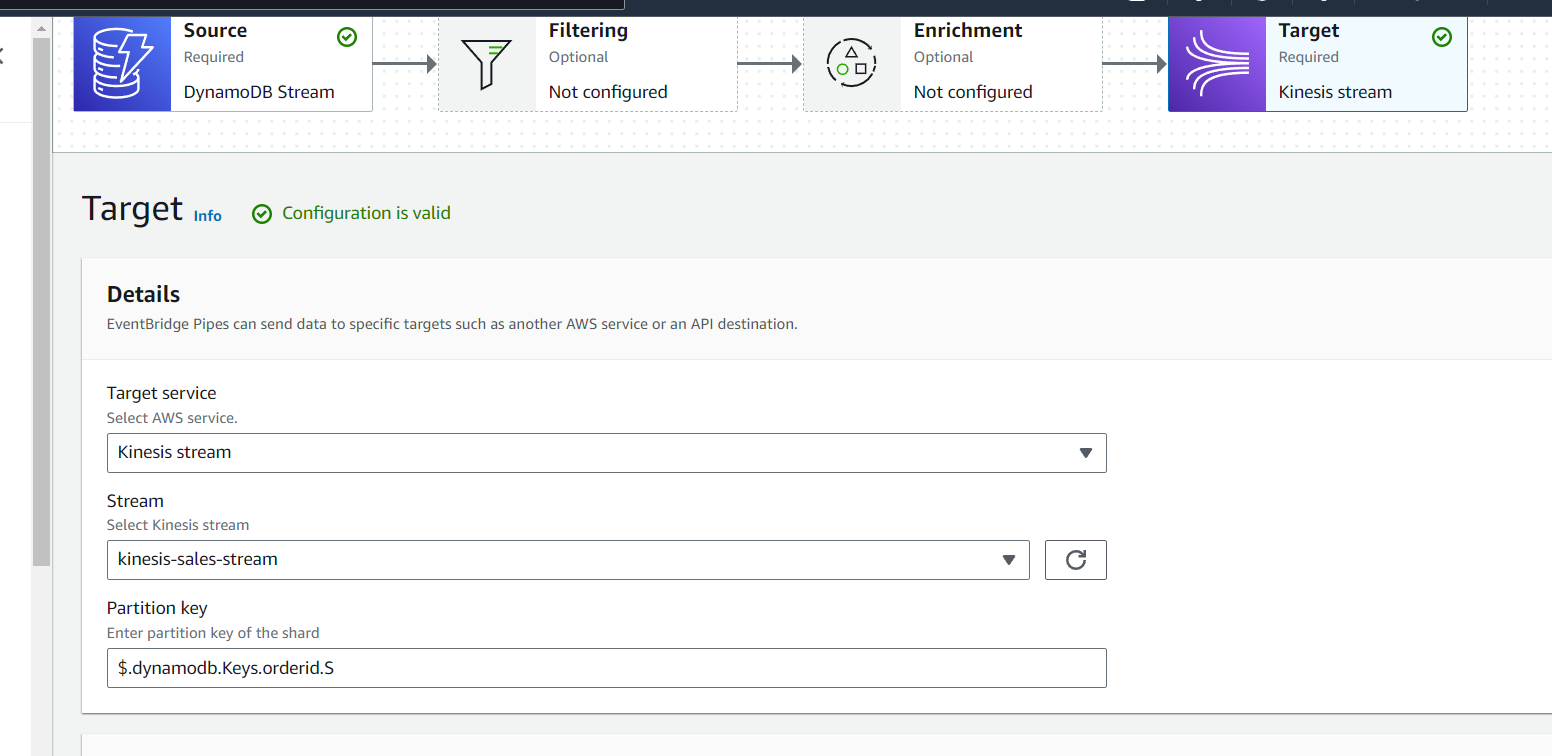
Next open Event bridge pipe 🡪 create pipe

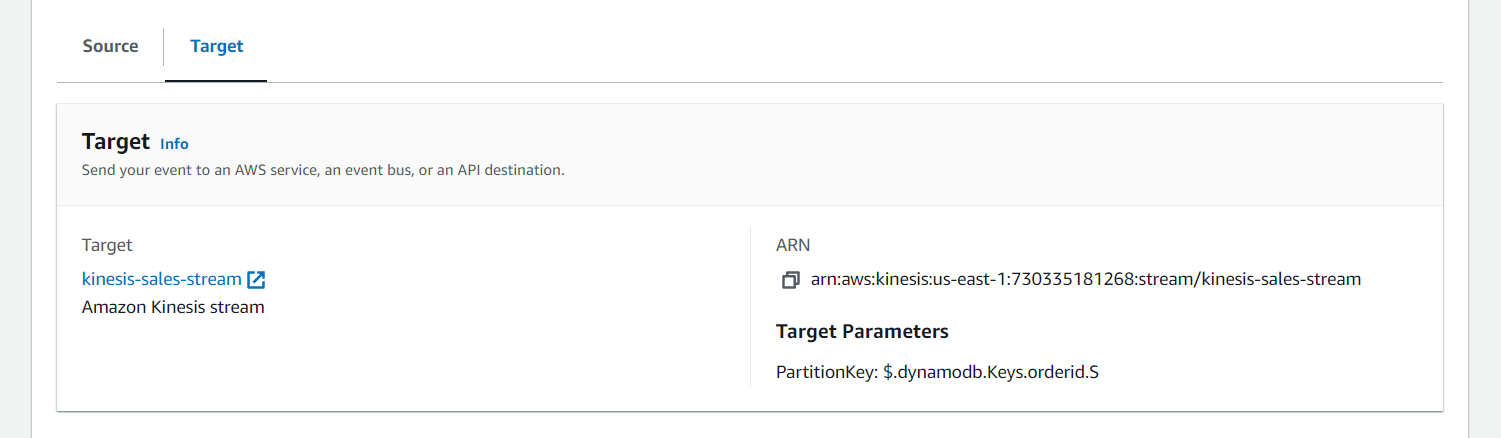
We added Dynamo DB as source and while creating add Dynamo DB Table we created before and other information can be see below



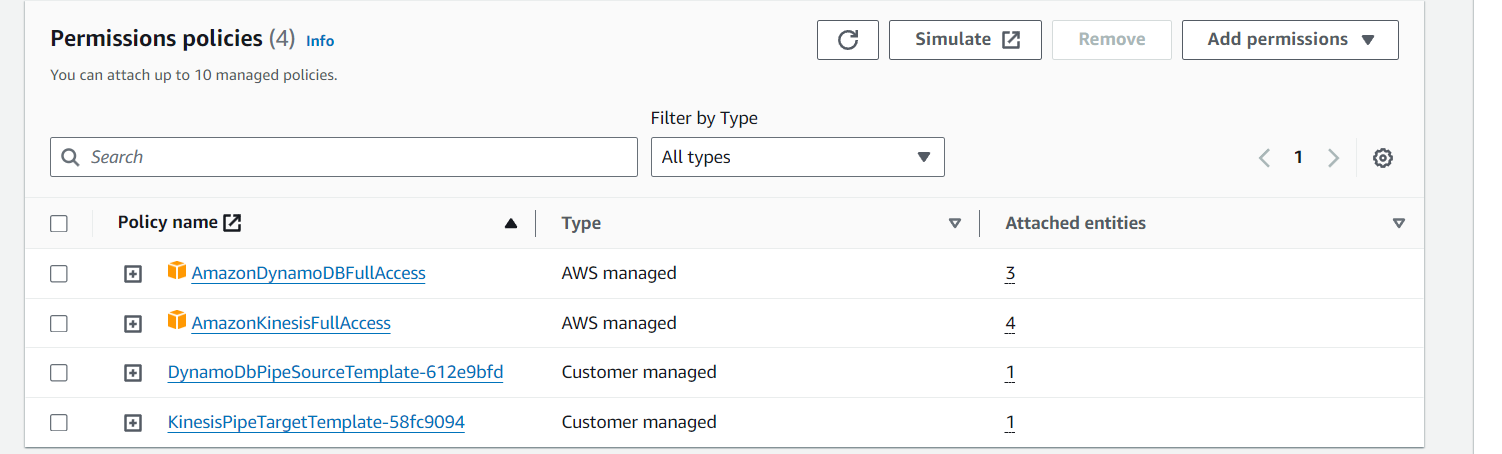


And for Target we added Kinesis data stream we created Before and added partition key 🡪$.dynamodb.Keys.orderid.S

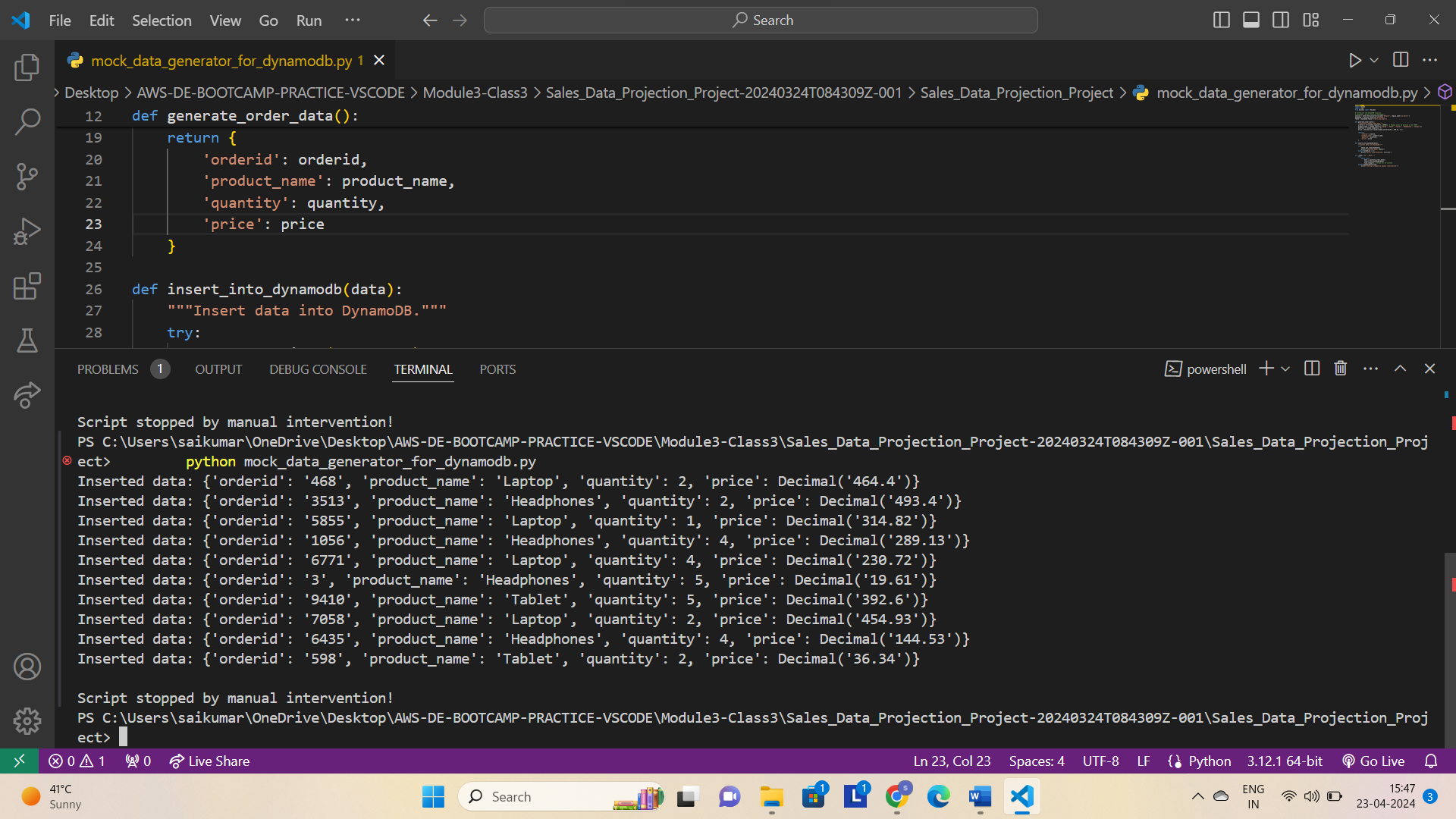




Next create and for IAM role added Permissions Below Dynamo , Kinesis Full access

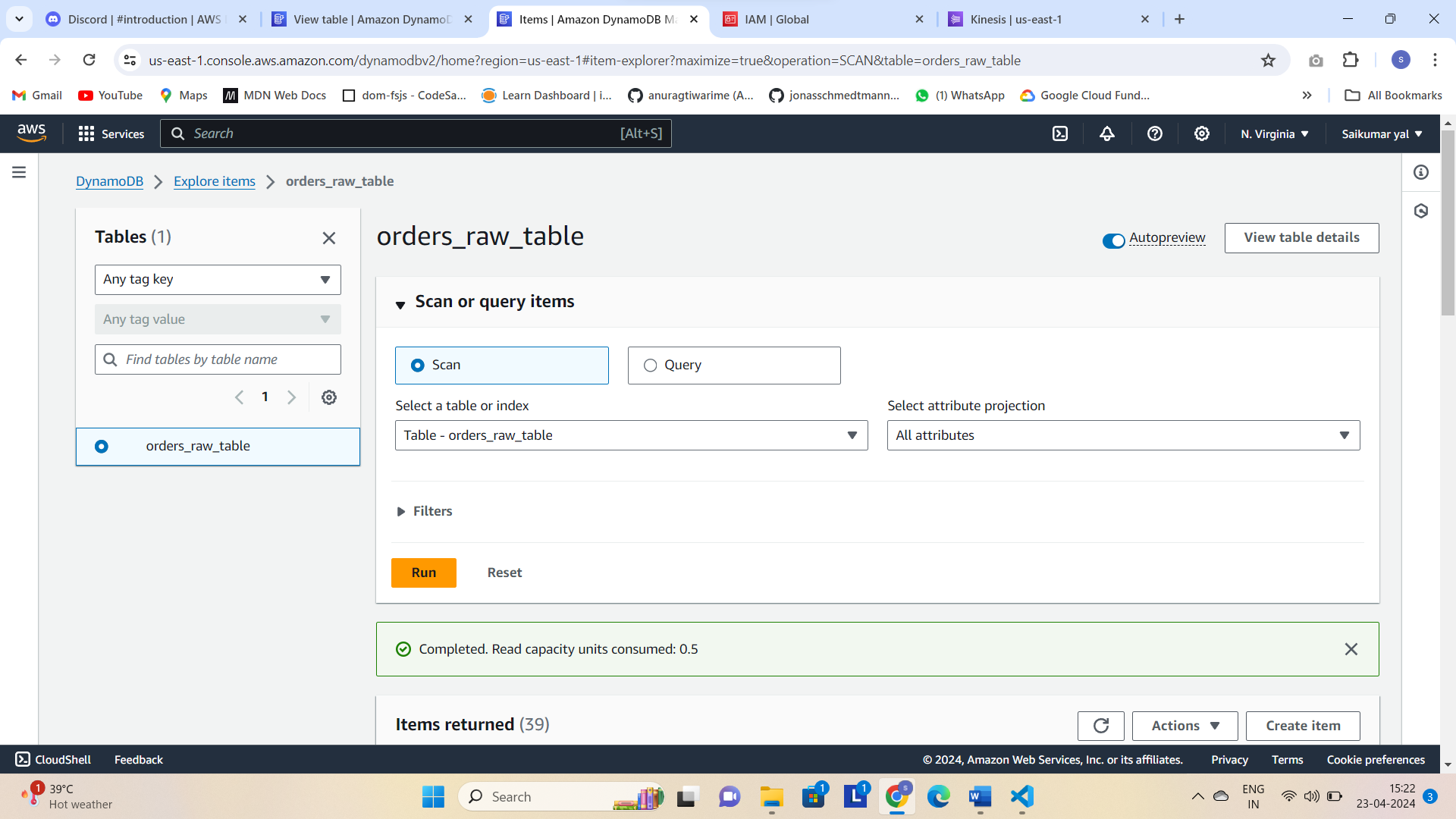


Next we will run Python Script we created which generates random orders I.e 

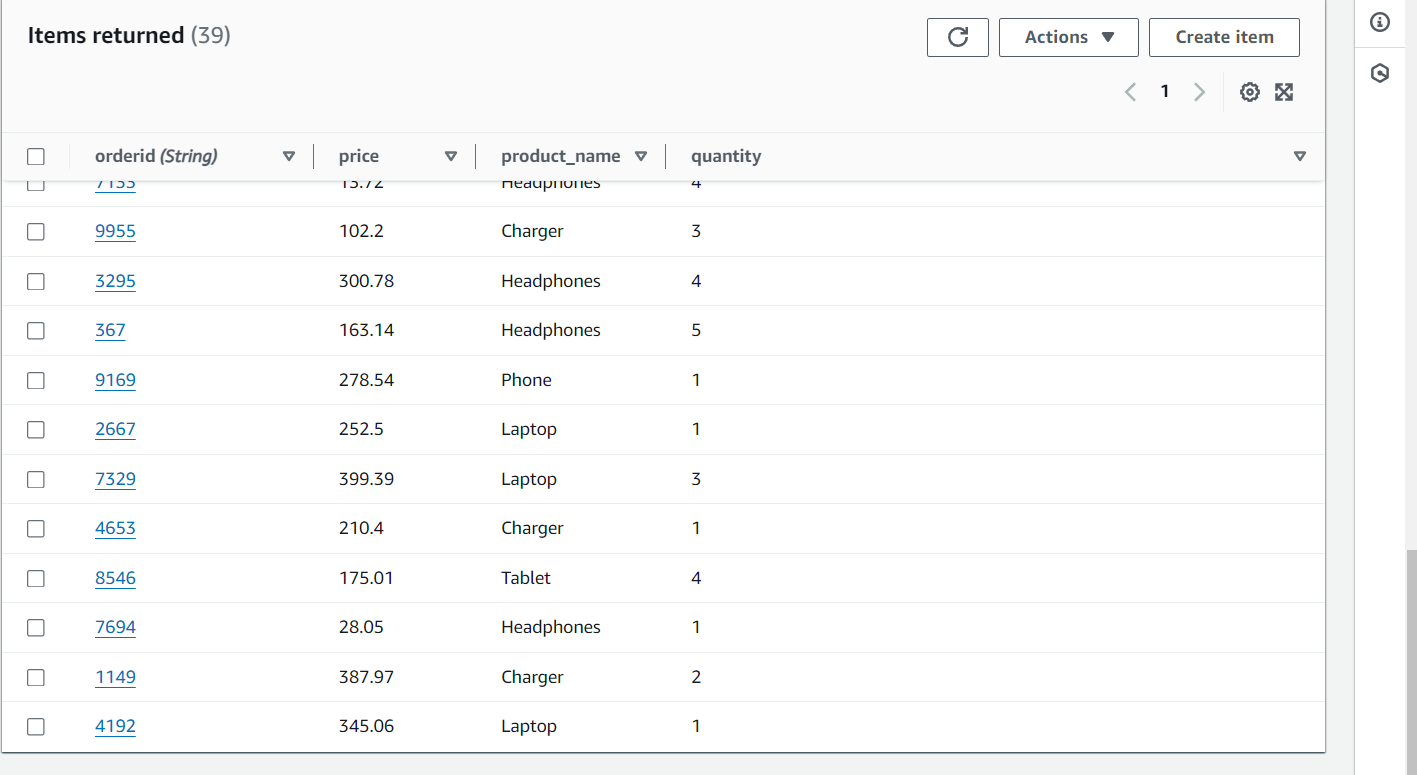


We run for some time that generated 10 records and stopped by manual intervention i.e ctrl+c

Next go to DynamoDB 🡪 Explore table items and Run and see the records are received by it



We can see items returned and here some older items are also there because I run this script before also

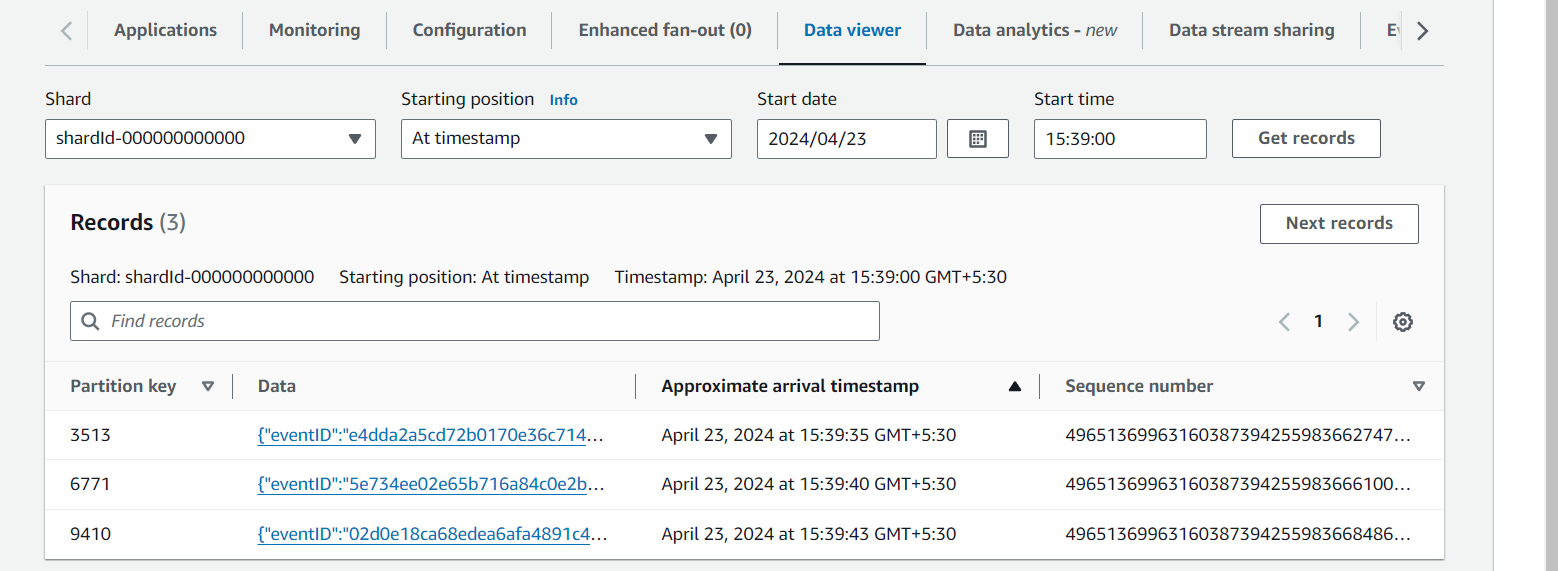


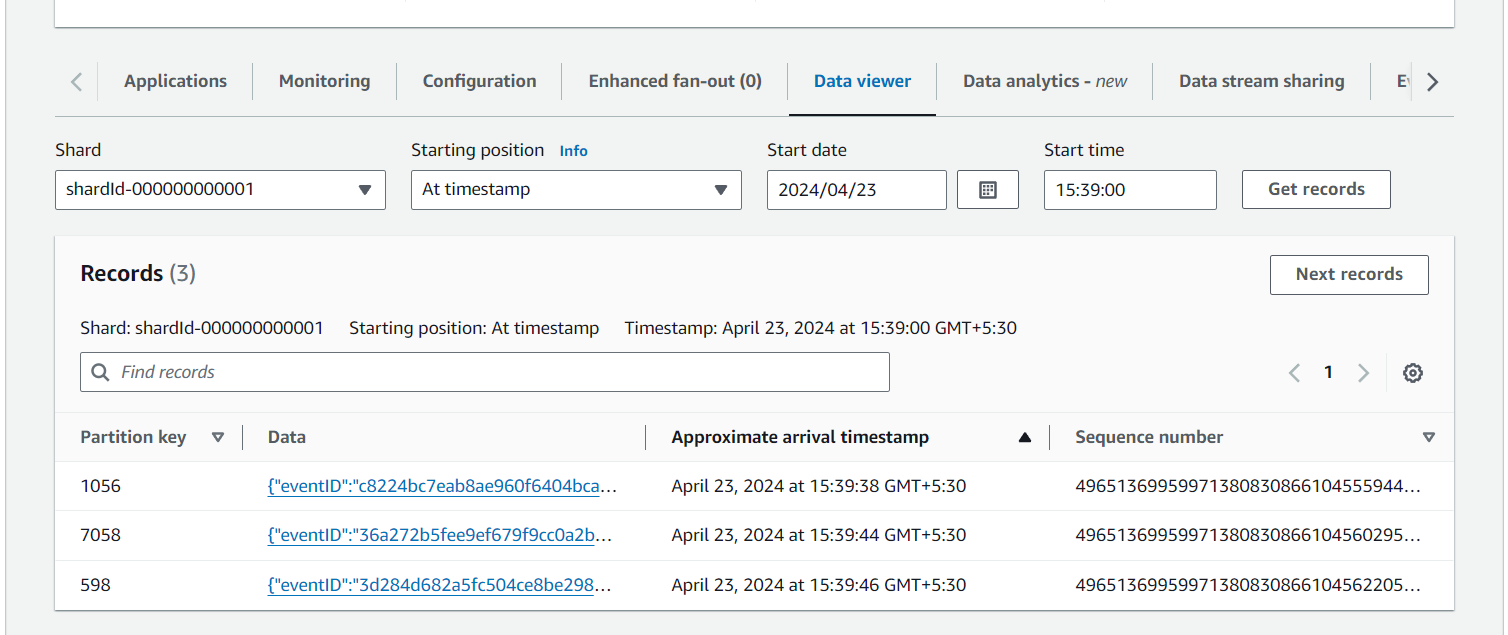
Next go to kinesis stream 🡪 Data Viewer and check as we create this as Target from Event bridge so this stream should receive those items/records

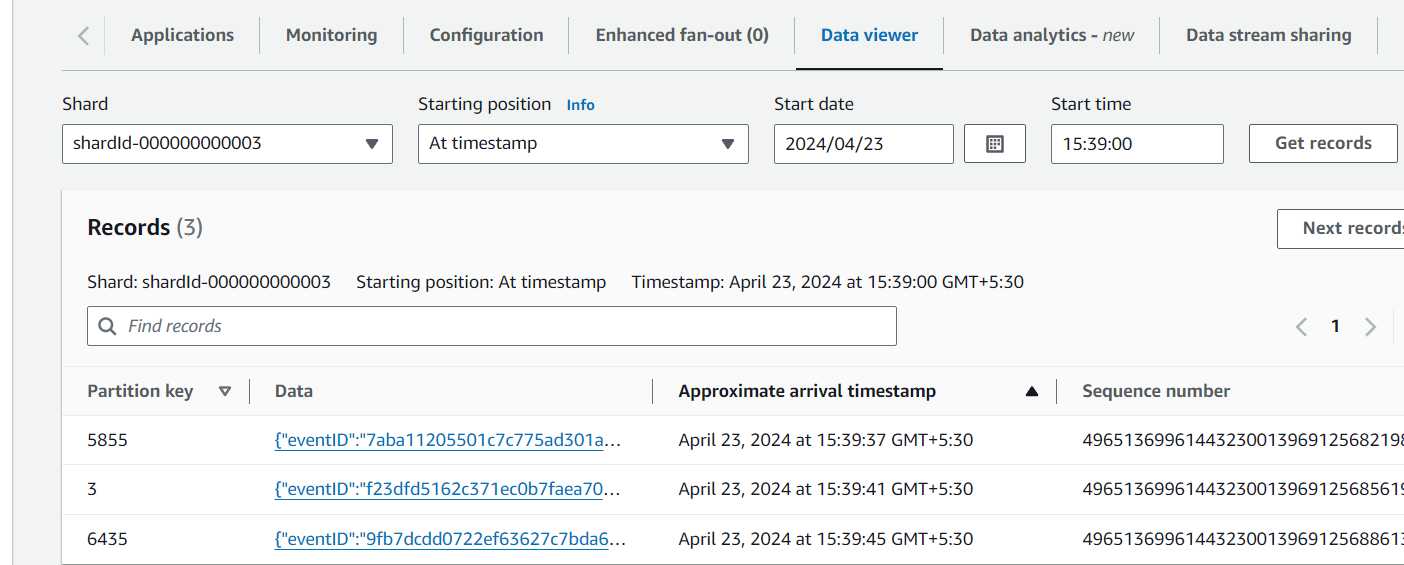
In shards select timestamp 🡪 start date and time when script run and click on 🡪Get Records

Select shards there will be records by partition key which is ordered and id is same for records from Dynamo DB and in Kinesis Stream ex: 3513 ..etc..

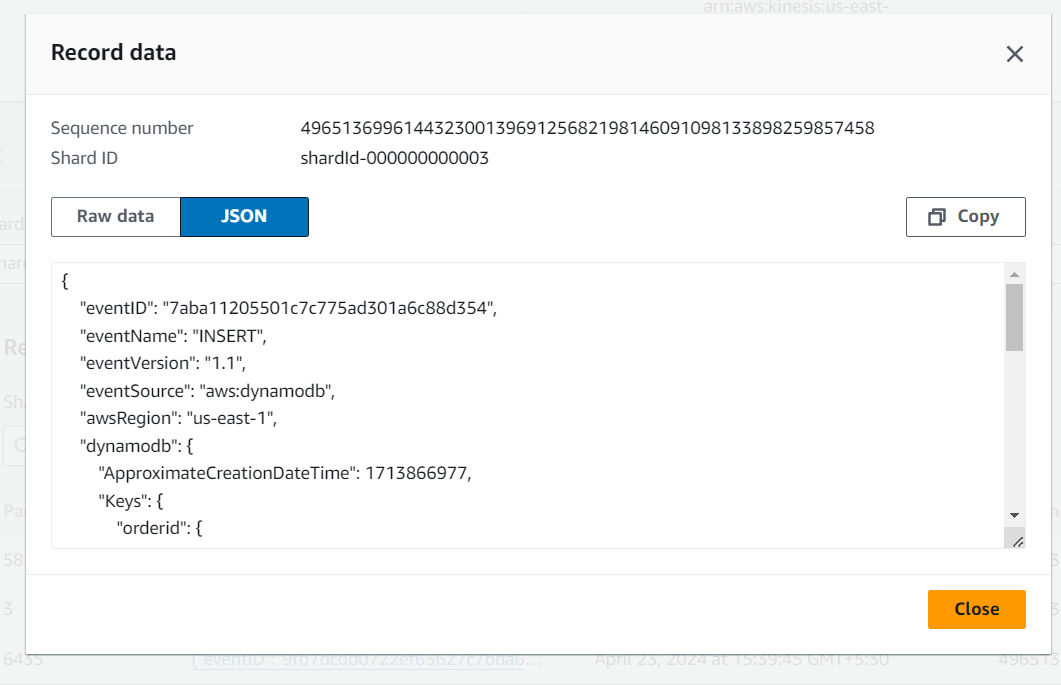
Select all shards to see all records

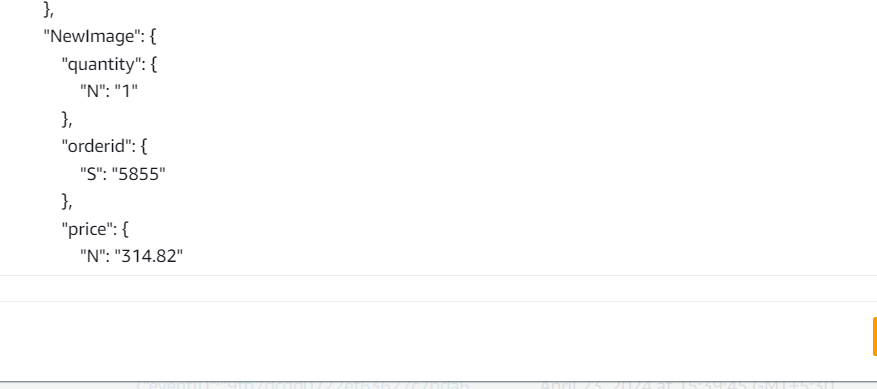






Just click on them and we can see Event happened i.e Insert and order id below product name and etc..





{

"eventID": "7aba11205501c7c775ad301a6c88d354",

"eventName": "INSERT",

"eventVersion": "1.1",

"eventSource": "aws:dynamodb",

"awsRegion": "us-east-1",

"dynamodb": {

"ApproximateCreationDateTime": 1713866977,

"Keys": {

"orderid": {

"S": "5855"

}

},

"NewImage": {

"quantity": {

"N": "1"

},

"orderid": {

"S": "5855"

},

"price": {

"N": "314.82"

},

"product\_name": {

"S": "Laptop"

}

},

"SequenceNumber": "8498100000000039955777332",

"SizeBytes": 59,

"StreamViewType": "NEW\_IMAGE"

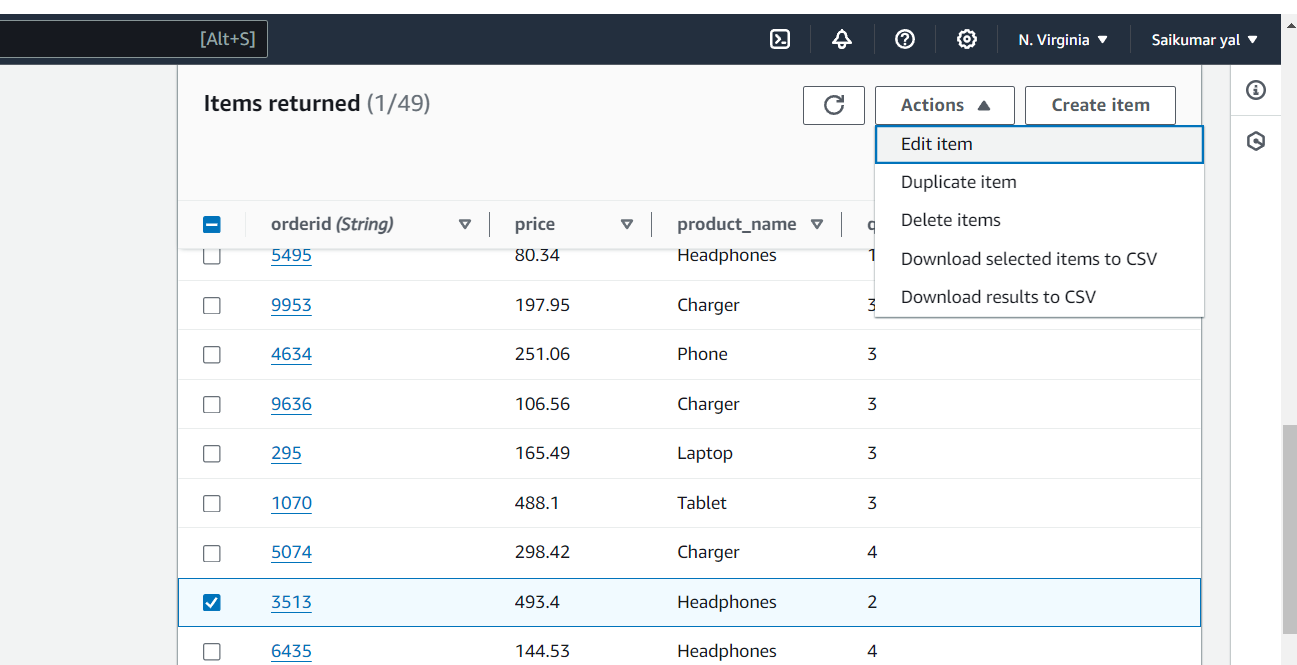
},

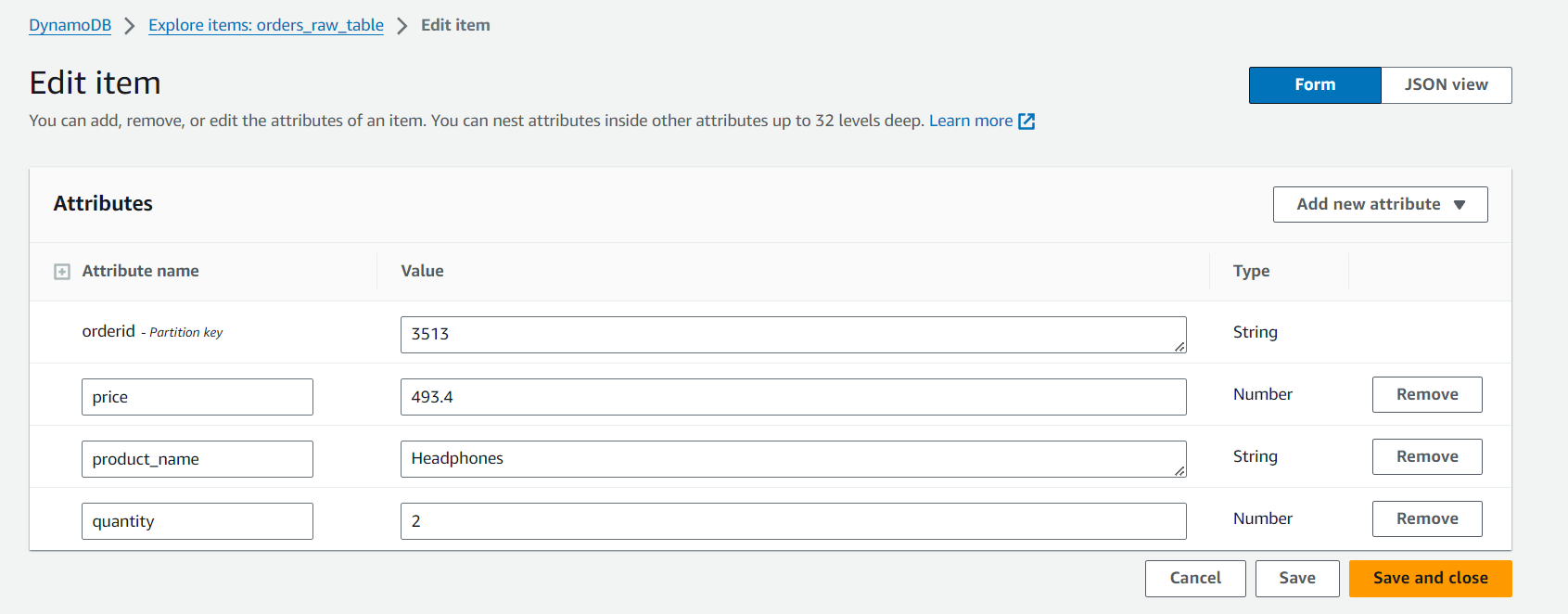
"eventSourceARN": "arn:aws:dynamodb:us-east-1:730335181268:table/orders\_raw\_table/stream/2024-04-23T10:00:21.201"

}

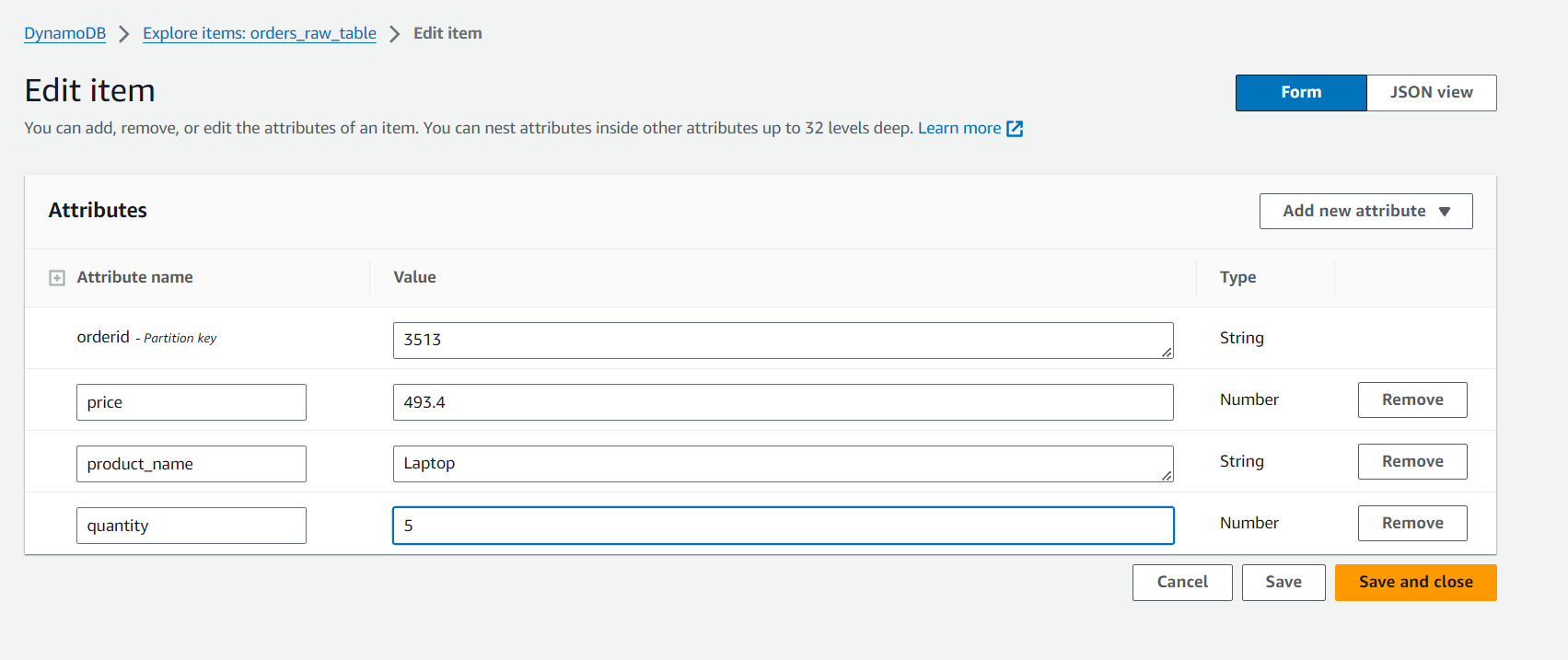
Now we will change quantity for 1 record in Dynamo DB and we will see is it updated in Kinesis

This is Update Operation

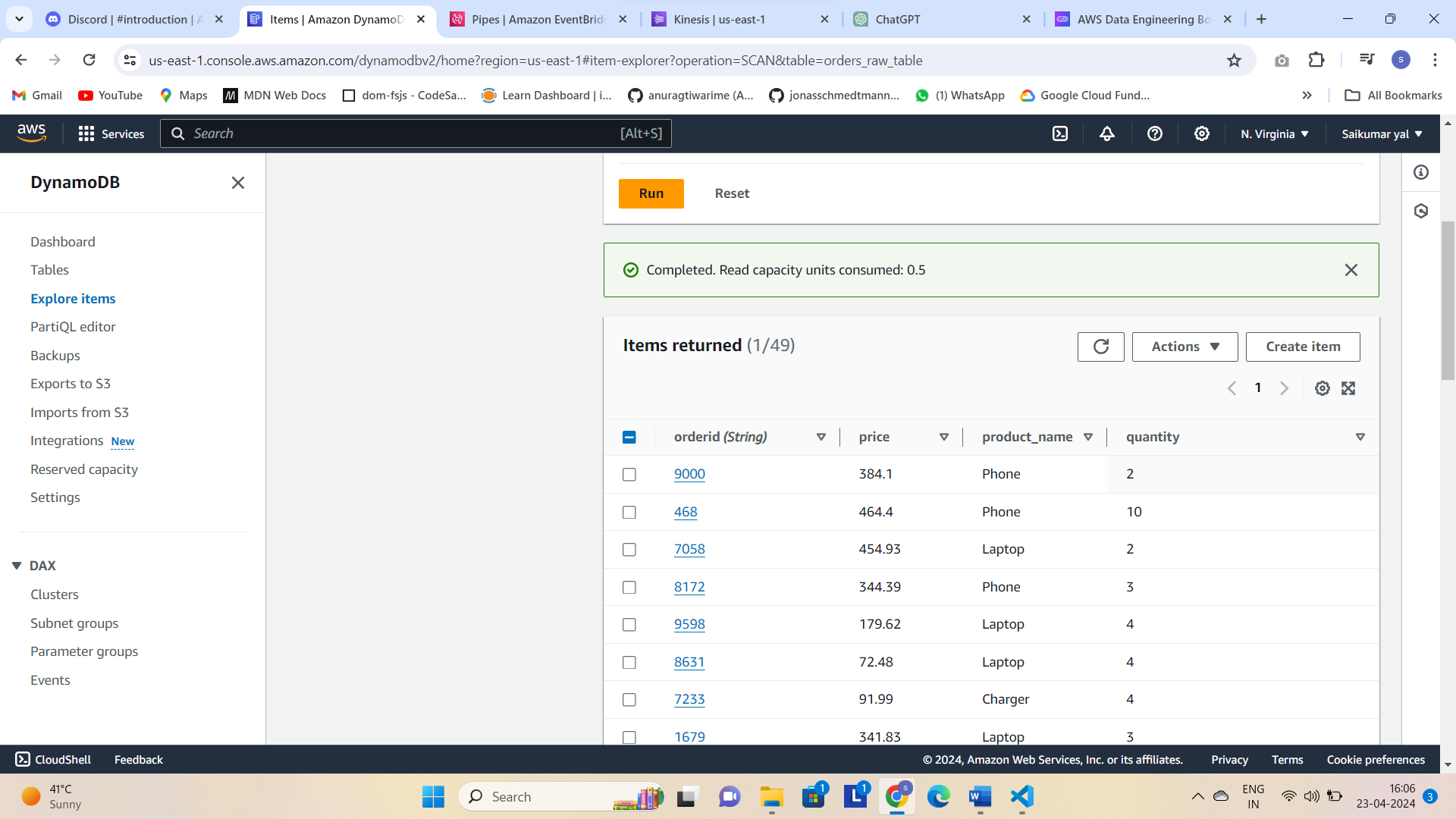




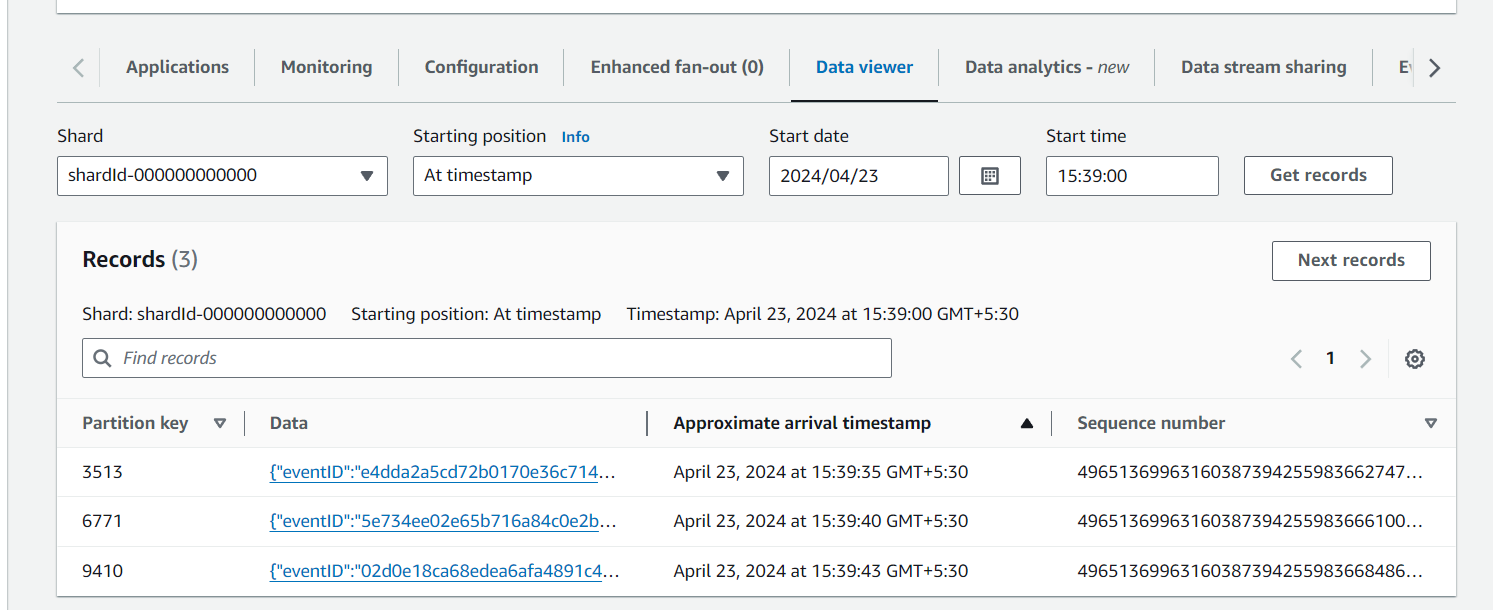
Now we Are Editing quantity to 5 and Product name



Now we made some changes for (Partition key) order id 3513 and we can see that the changes will not shown in DynamoDB the items will not be increased But The Event Bridge Captures the changes CDC Events and sends to Kinesis we can see there



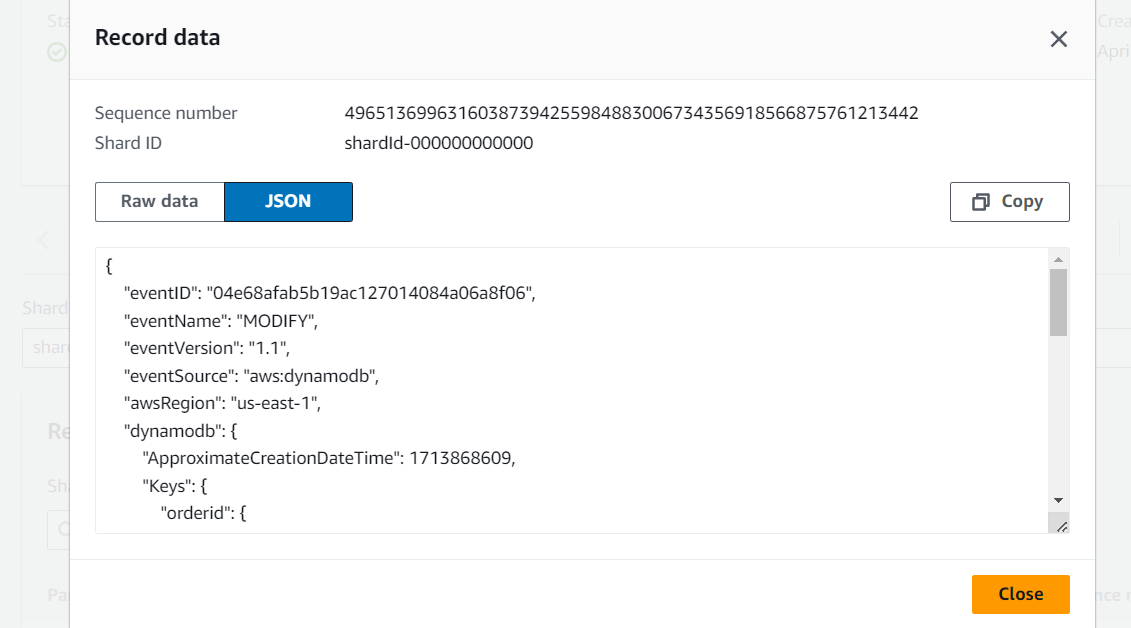
Next in kinesis see in shards see Timestamp this is befor inserted one for id 3513

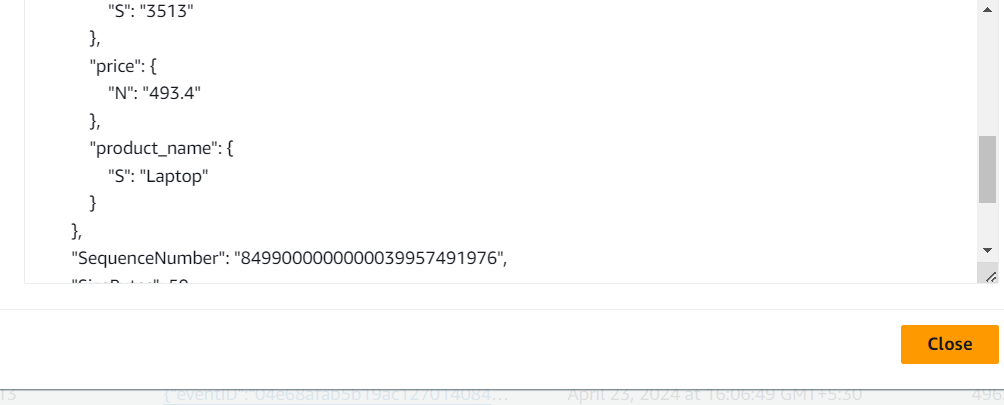
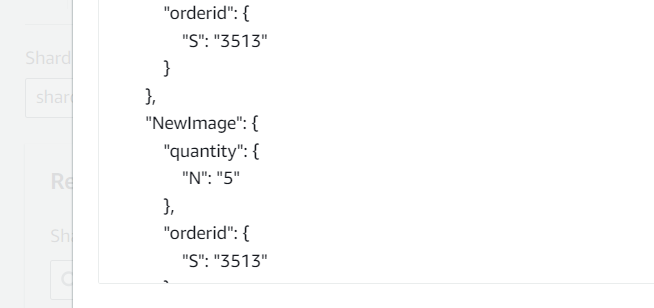


After we modified in Dynamo DB we can see changes this order id will be same i.e partition key but event will be changed u can see below



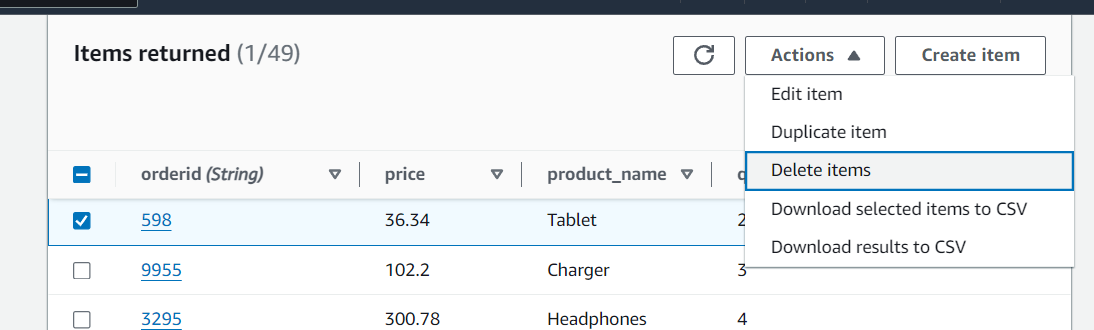
We can see it’s event is MODIFY and also check changes made such as quantity changed from 2 to 5 and Product\_name modified from Headphones to Laptop.

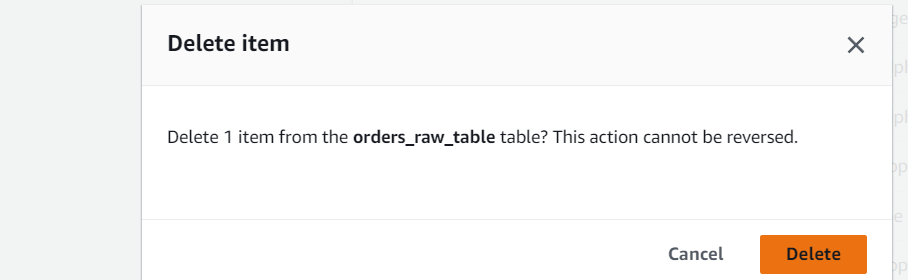
****

****

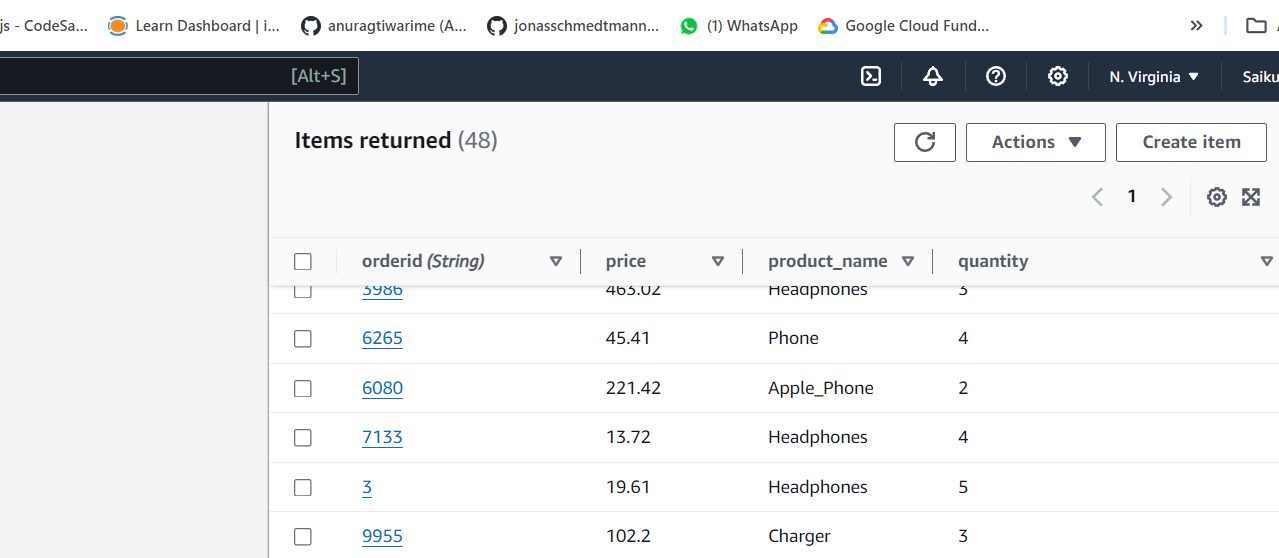
**🡪Next**

**Now** we will delete 1 record in DynamoDB and we will see in kinesis it is captured that deleted Event or thing

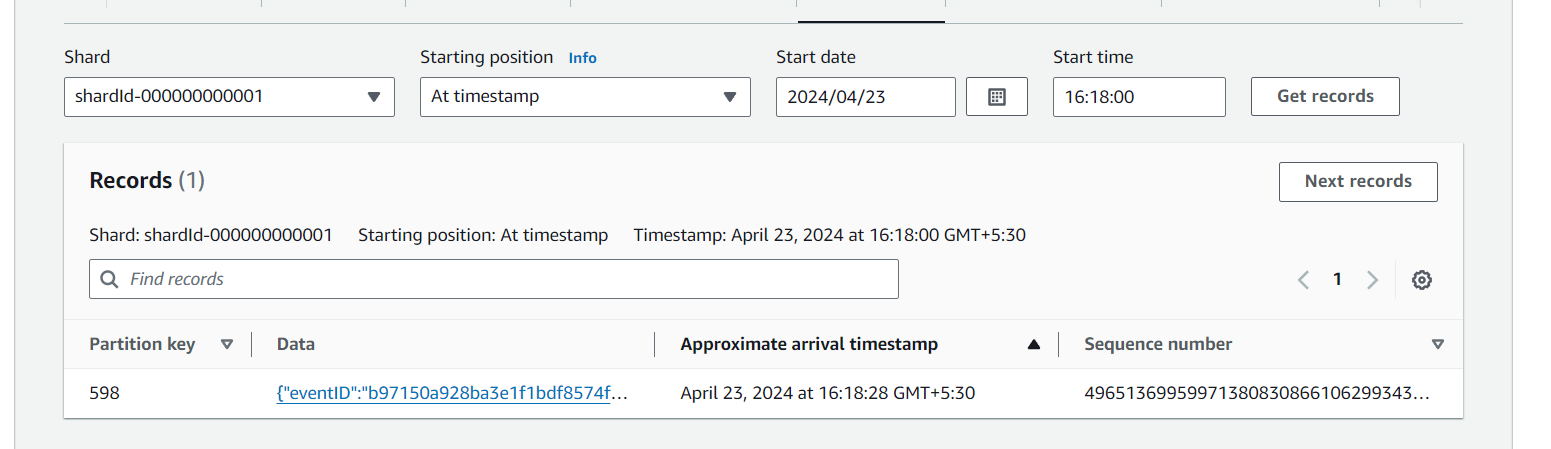


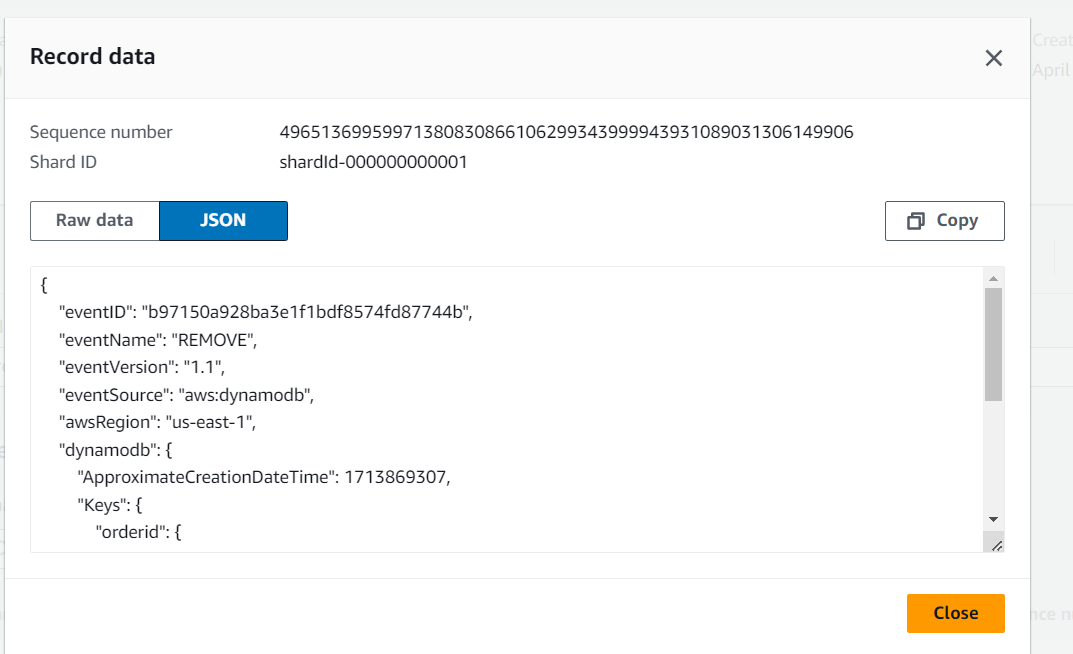


Deleted and item is also deleted in Dynamo DB list



Go to kinesis and check removed orderid i.e 598 we can see Event name as Remove in SO Kinesis Captured that too



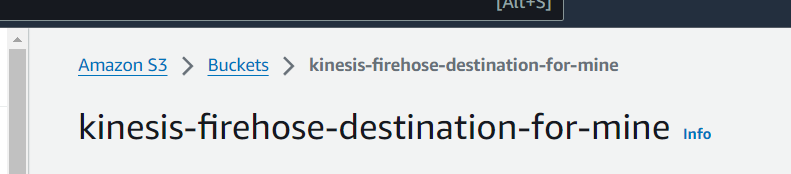


**🡪So Kinesis Data streams Capture CDC(Change Data Capture) changed Events such as Insert, Update, delete etc..**

**Next Open Kinesis Firehose Before that we will create S3 bucket and lambda function**

**S3 Bucket**

We created

****

**Next will Create Transformer Lambda function**

This function will do some transformations from Firehose and sends back to it only

The transformation function processes DynamoDB stream records by decoding base64-encoded data, extracting relevant fields (such as order ID, product name, quantity, price, and event type), and converting them into standard JSON format. Exception handling ensures robust processing even in case of invalid data or missing keys.

****

**The code written for transformation in lambda was**

import base64

import json

from datetime import datetime

def lambda\_handler(event, context):

    output\_records = []

    for record in event['records']:

        # Try to decode and transform the record

        try:

            # Decode the input data from base64

            payload = base64.b64decode(record['data'])

            payload\_json = json.loads(payload)

            # Access the 'eventName' and 'ApproximateCreationDateTime' from the payload

            event\_name = payload\_json['eventName']

            approx\_creation\_datetime = payload\_json['dynamodb']['ApproximateCreationDateTime']

            # Convert the Unix timestamp to a human-readable date format

            creation\_datetime = datetime.utcfromtimestamp(approx\_creation\_datetime).isoformat() + 'Z'

            # Access the data in the 'dynamodb' key

            dynamodb\_data = payload\_json['dynamodb']

            new\_image = dynamodb\_data['NewImage']

            print(new\_image)

            # Extract required fields from NewImage

            transformed\_data = {

                'orderid': new\_image['orderid']['S'],

                'product\_name': new\_image['product\_name']['S'],

                'quantity': int(new\_image['quantity']['N']),

                'price': float(new\_image['price']['N']),

                'cdc\_event\_type': event\_name,  # Include the event name

                'creation\_datetime': creation\_datetime  # Include the formatted creation datetime

            }

            # Convert the transformed data to a JSON string and then encode it as base64

            transformed\_data\_str = json.dumps(transformed\_data) + '\n'

            transformed\_data\_encoded = base64.b64encode(transformed\_data\_str.encode('utf-8')).decode('utf-8')

            # Append the transformed record to the output using 'eventID' as 'recordId'

            output\_records.append({

                'recordId': record['recordId'],

                'result': 'Ok',

                'data': transformed\_data\_encoded

            })

        except Exception as e:

            # If there's any error with processing the record, mark it as ProcessingFailed but still return the recordId

            output\_records.append({

                'recordId': record['recordId'],

                'result': 'ProcessingFailed',

                'data': record['data']  # simply pass the original data back

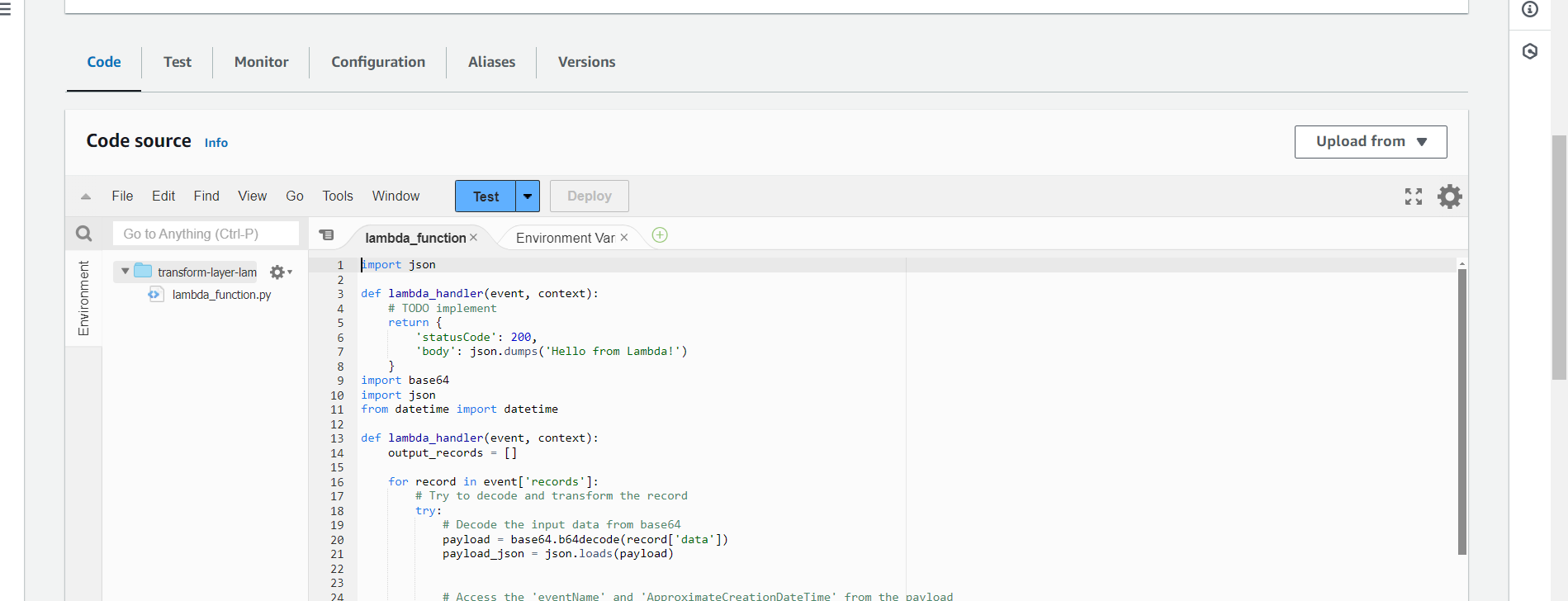
            })

    return {

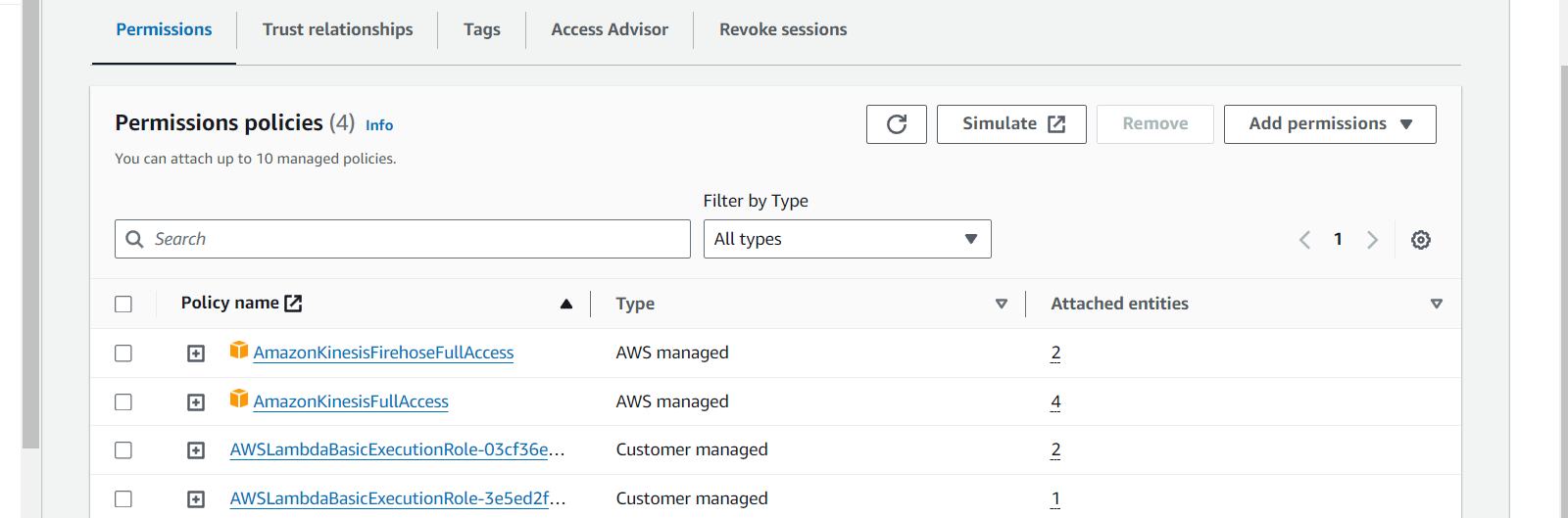
        'records': output\_records

    }

**And pasted this code in lambda function**

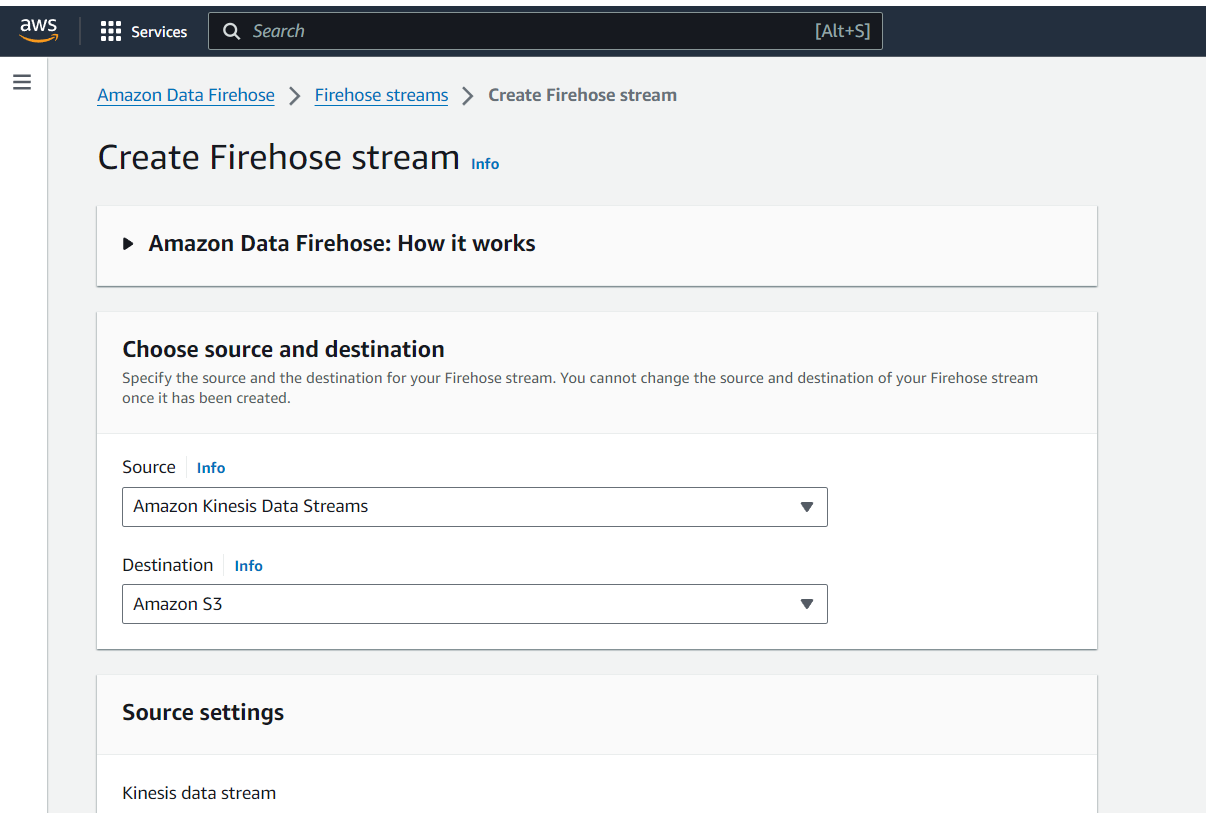


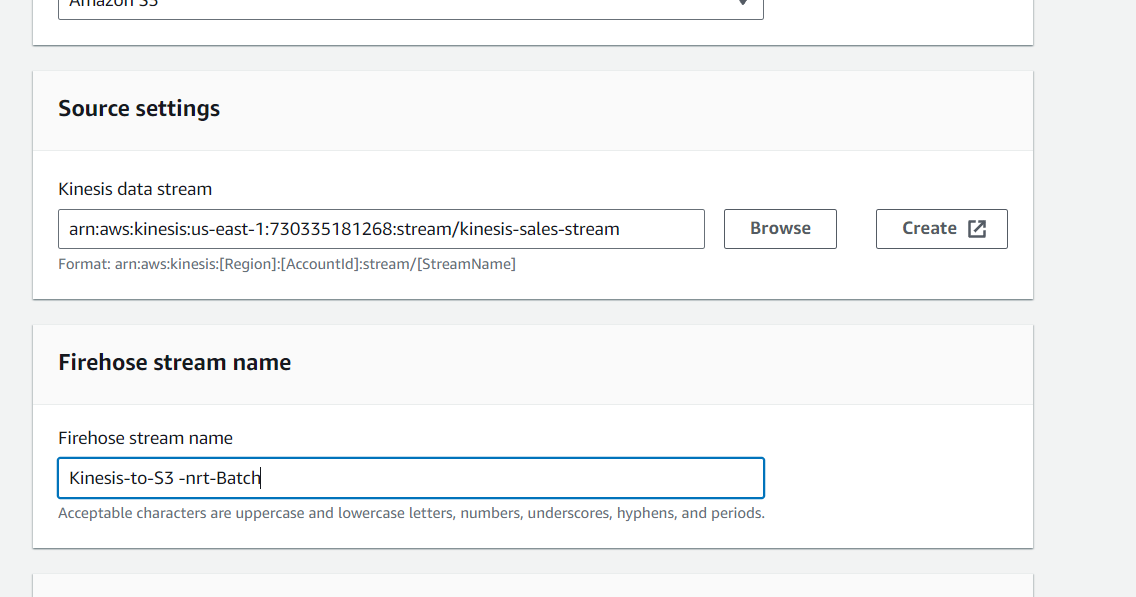
And change configurations to 15 min and add permissions below



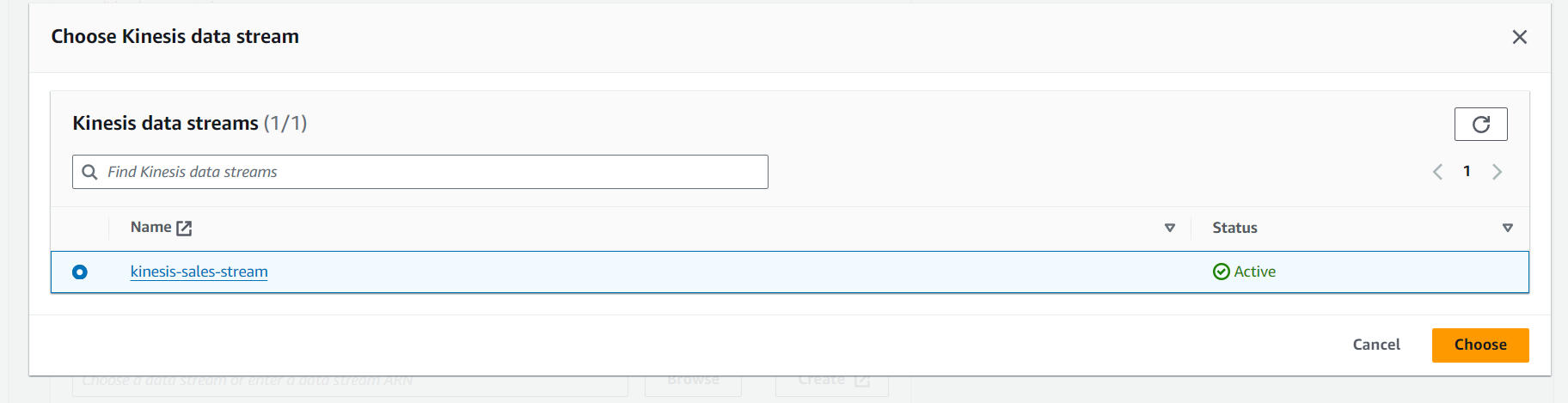
**Next Create Firehose 🡪**

Next Kinesis Firehose Stream And create it add source as Kinesis data stream and destinations as S3

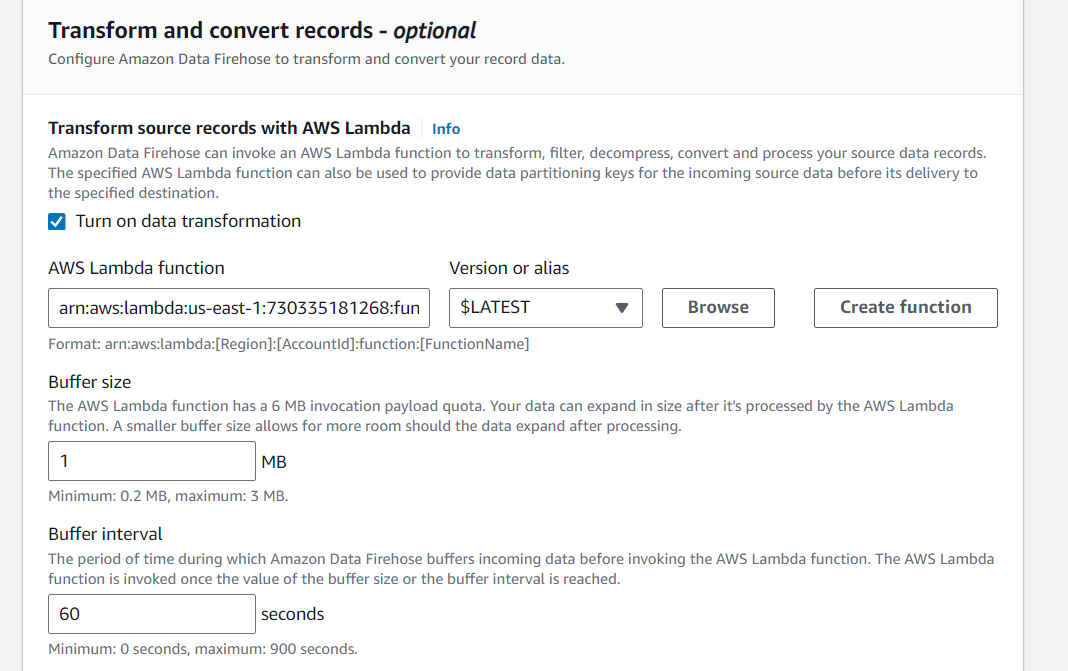
****

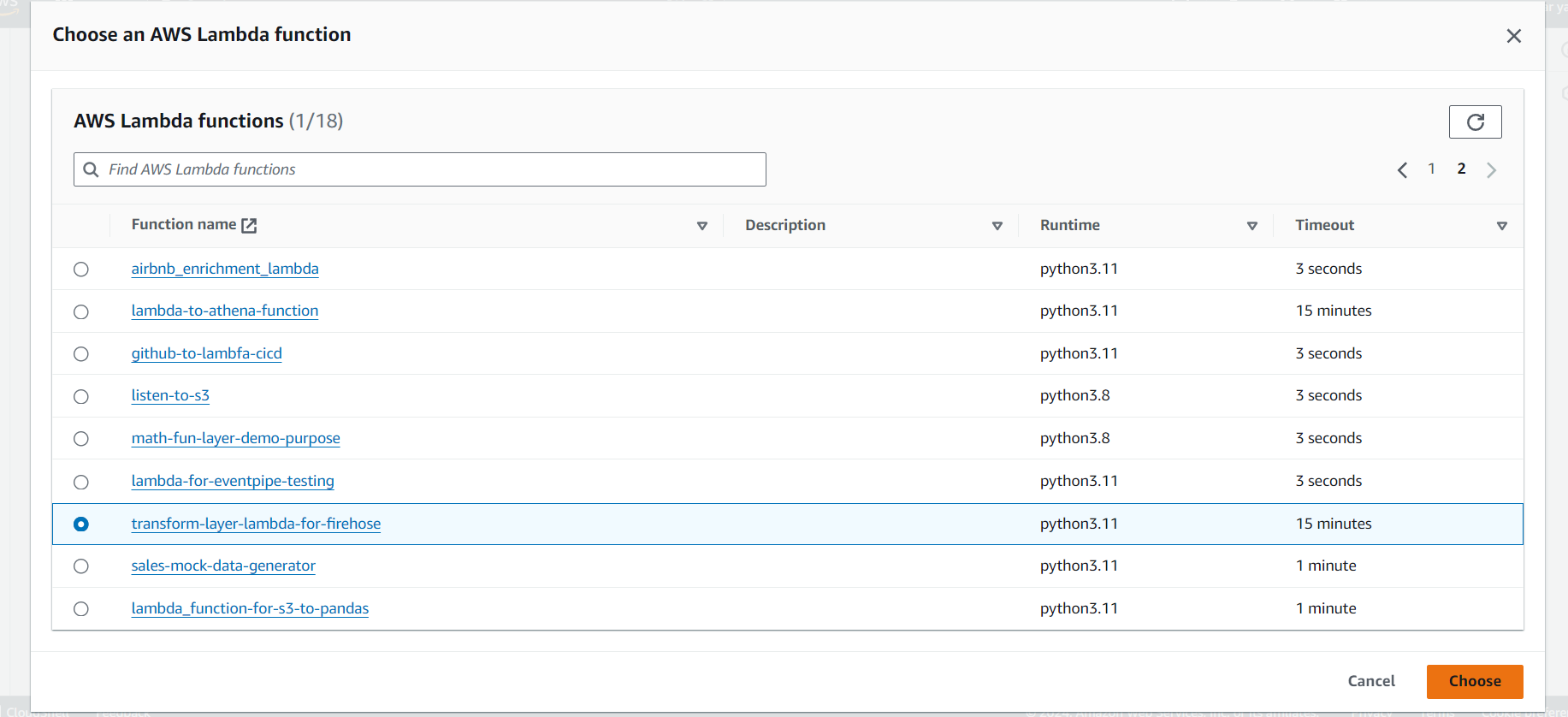
****

Browse and add kinesis stream

****

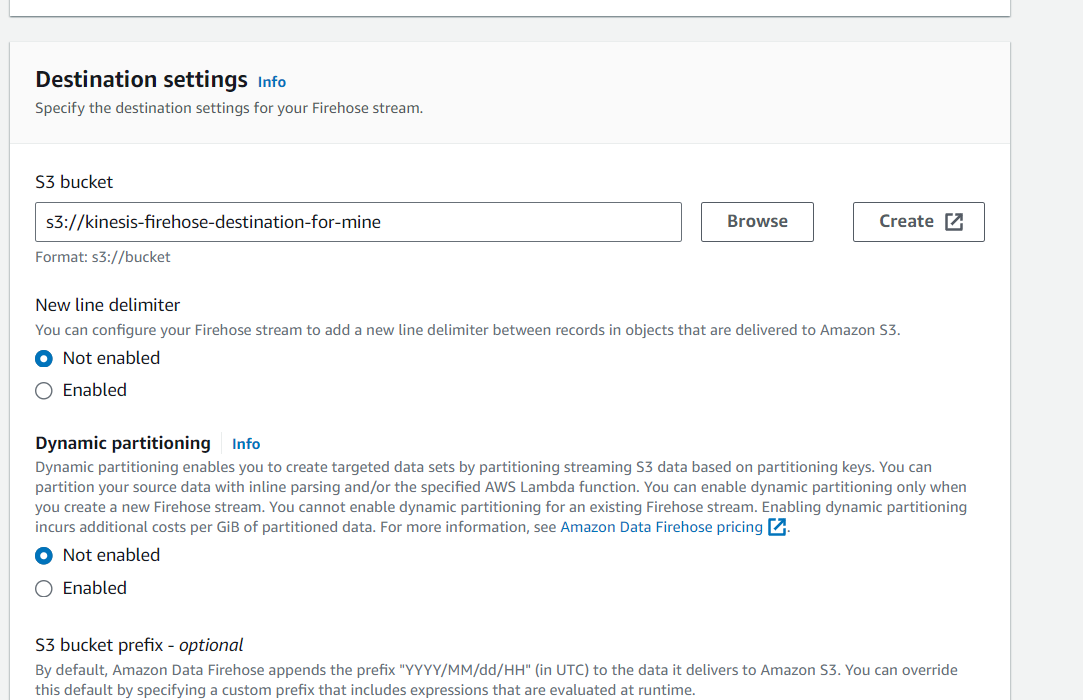
Choose lambda function we created before and Turn on Data transformation

****

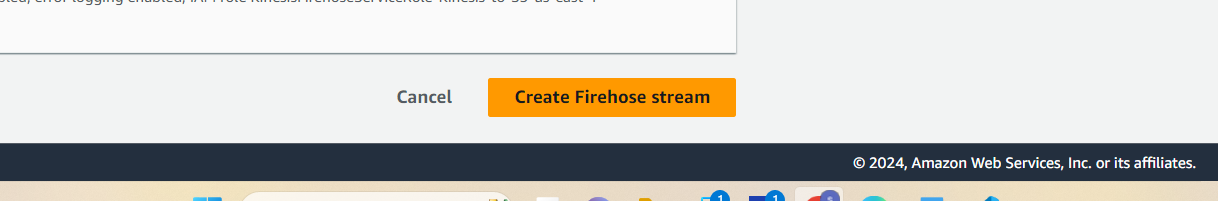
****

**Next**

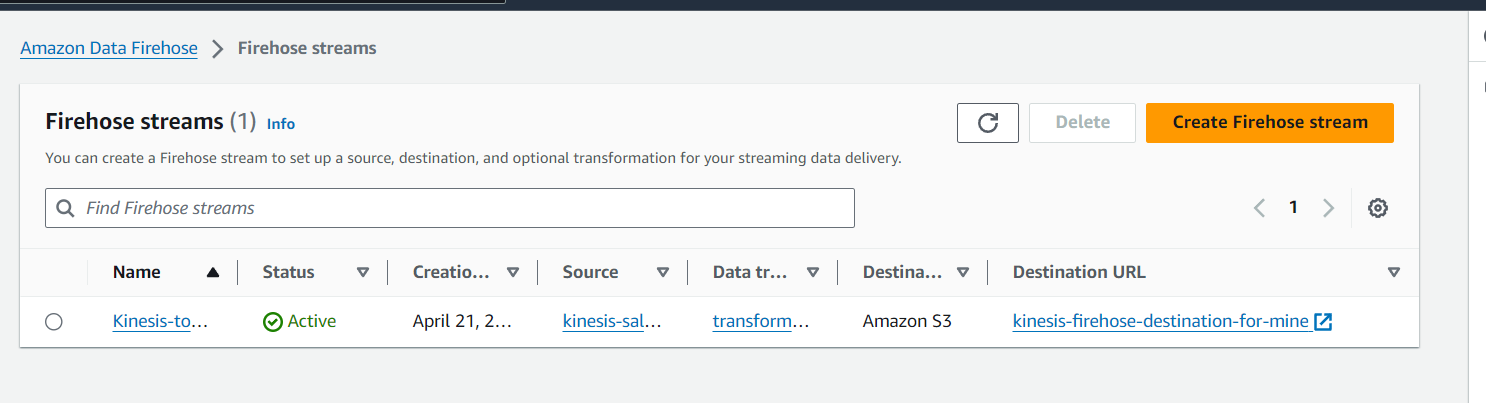
Choose the S3 bucket we created



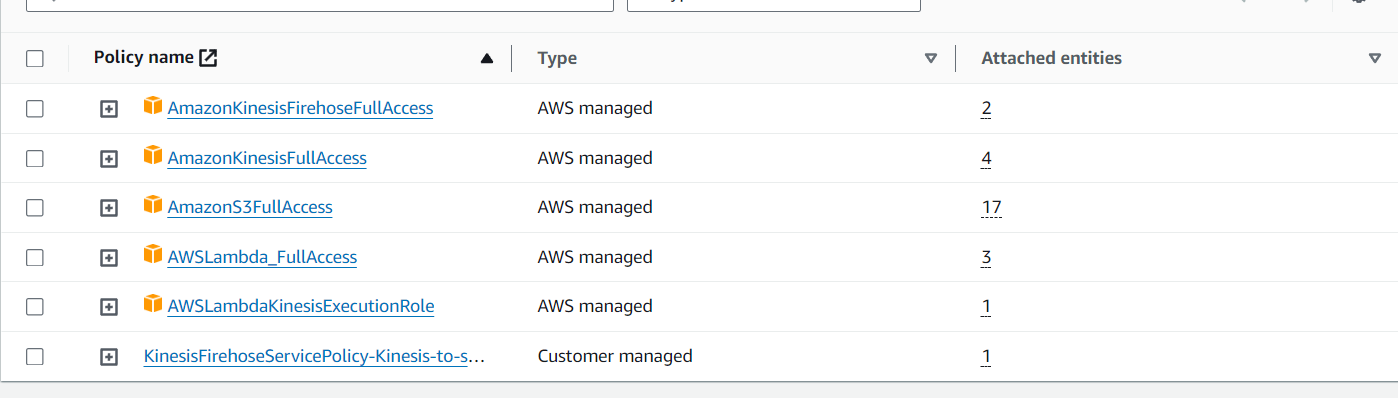
**Finally create it**

****

Firehose stream is created

****

**Add Permissions IAM Role to this Firehose**

****

**Now**

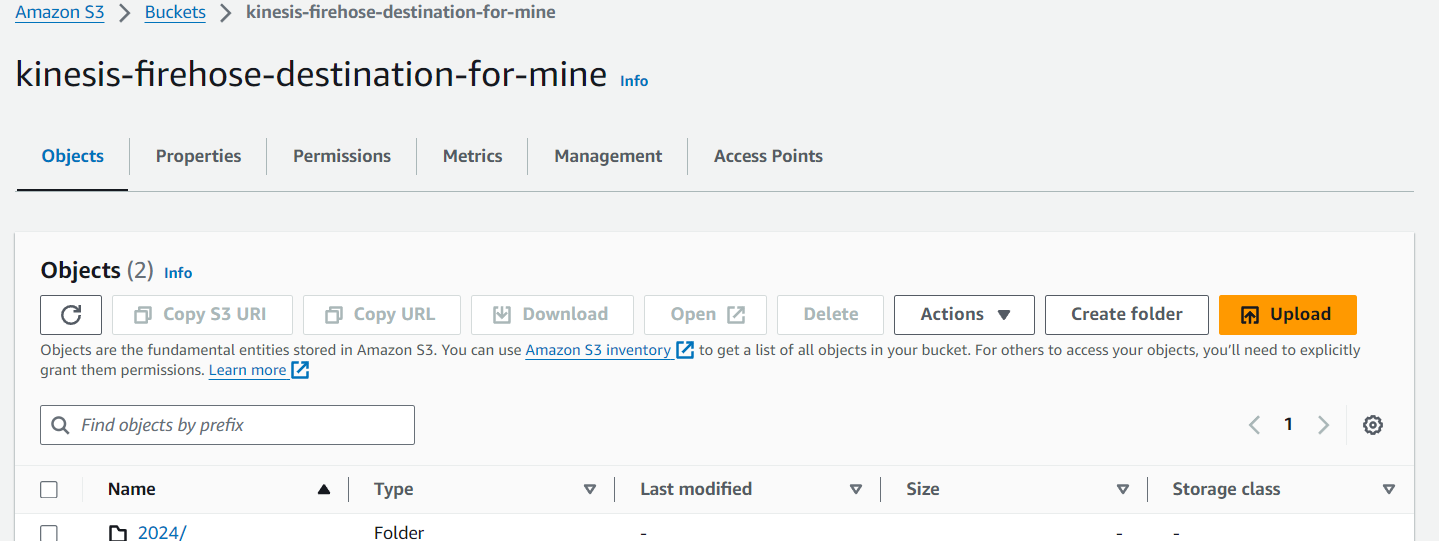
🡪Again run mock\_data\_generator\_for\_dynamodb.py script for 50 to 60 records and



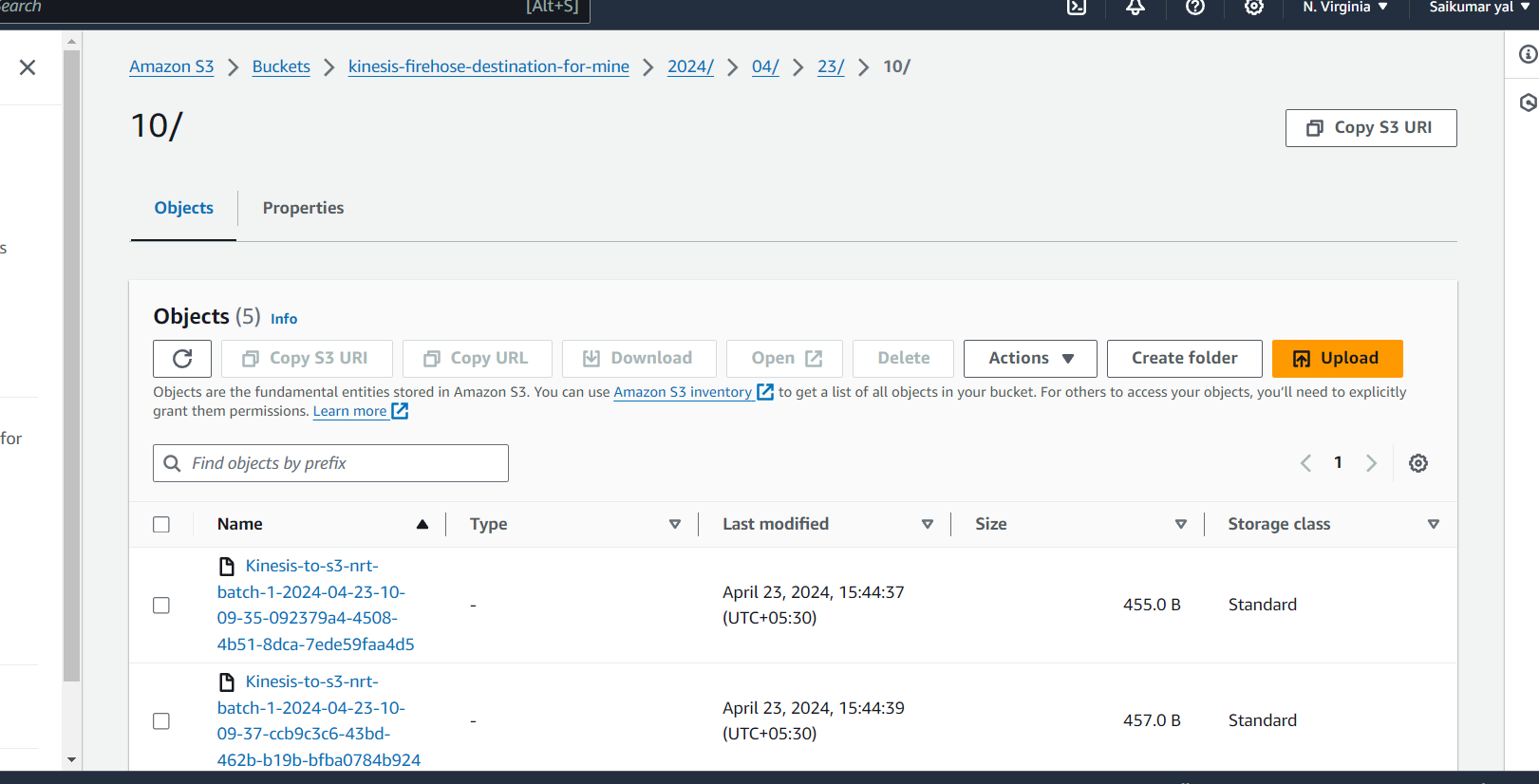
Now the thing will happen is This script will generate orders and that are pushed to DynamoDB and through eventBridge they are reached to Kinesis Data streams and from there Firehose Consume them as Batch and do transformation in Lambda and sends to s3 Bucket

So see those files are received in S3

🡪Next See that files is received in S3 Bucket

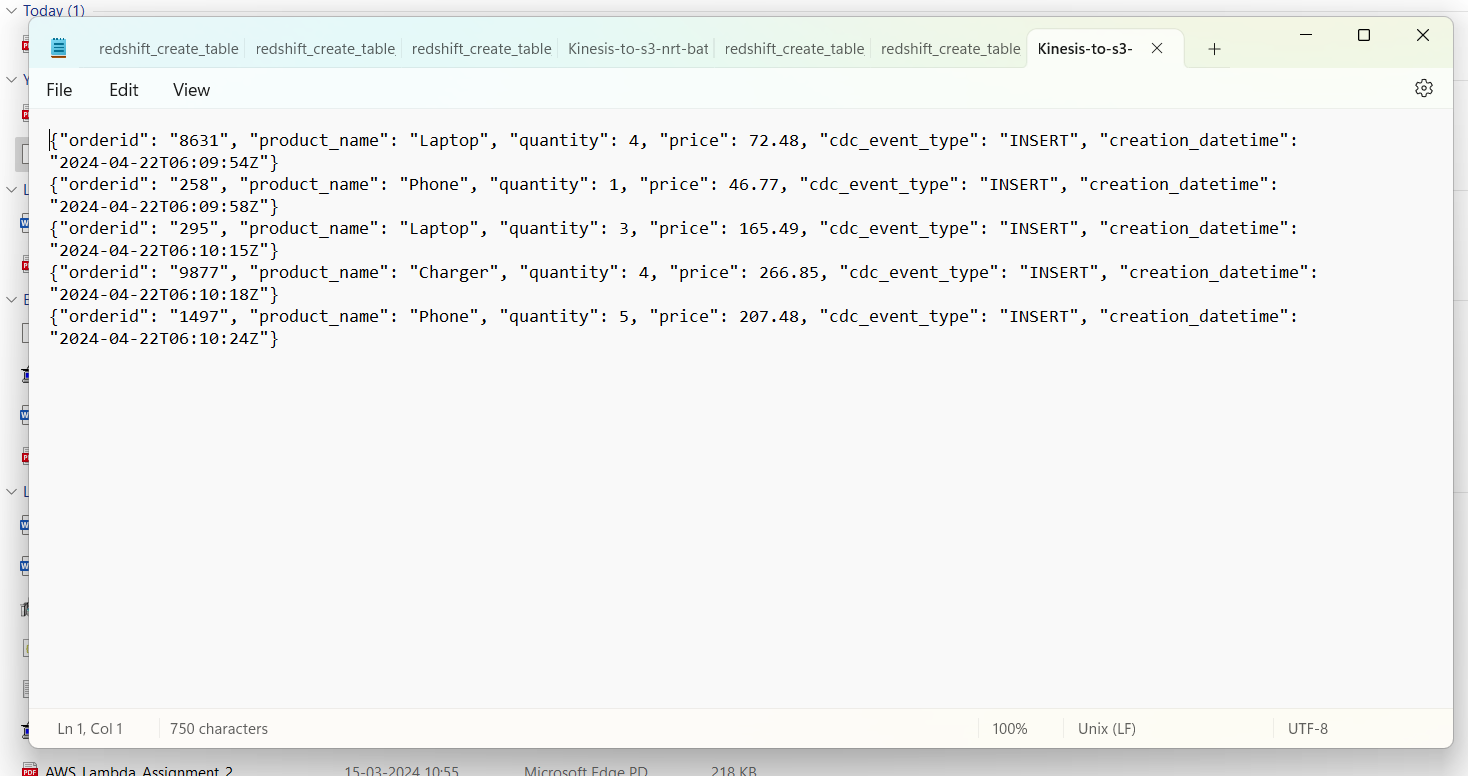


We can see some files are received in s3 Bucket

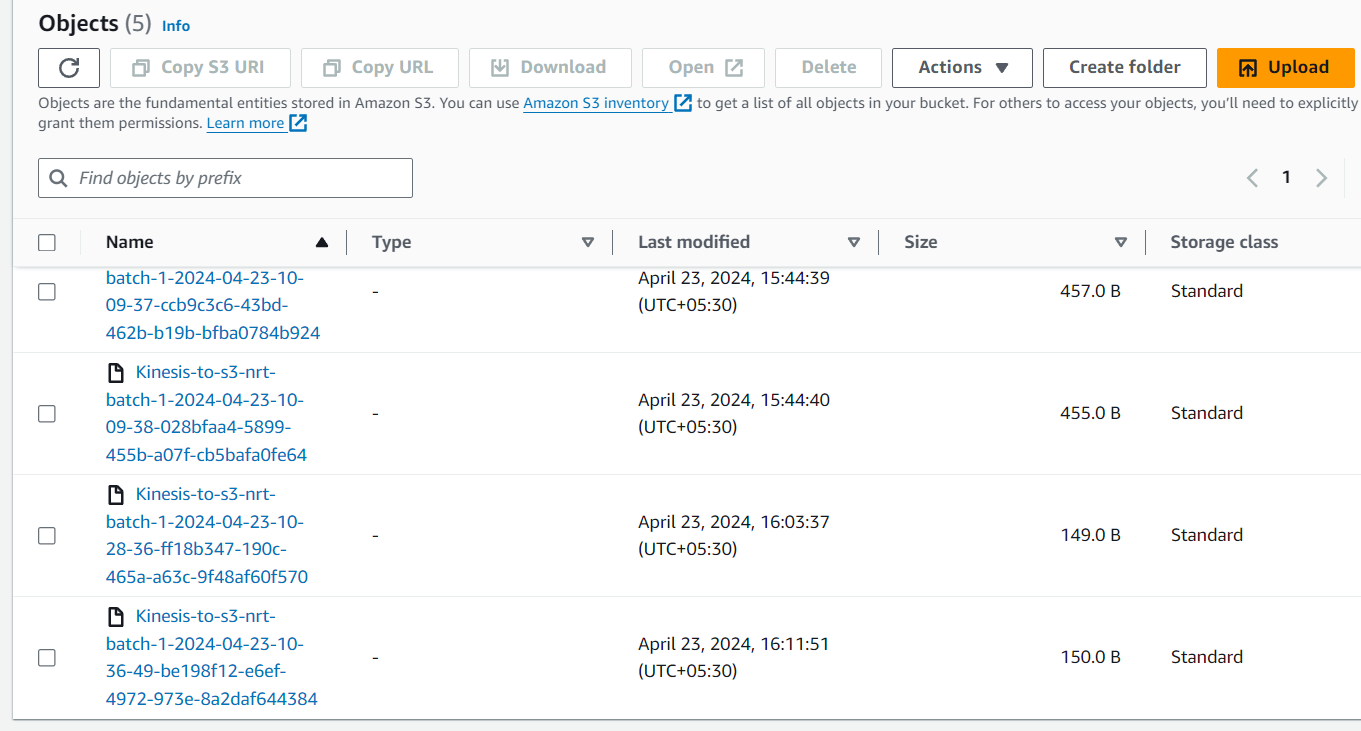


Download and Open them we can see records came as batch to S3 files are Received like Batches

This is only 1 file opened

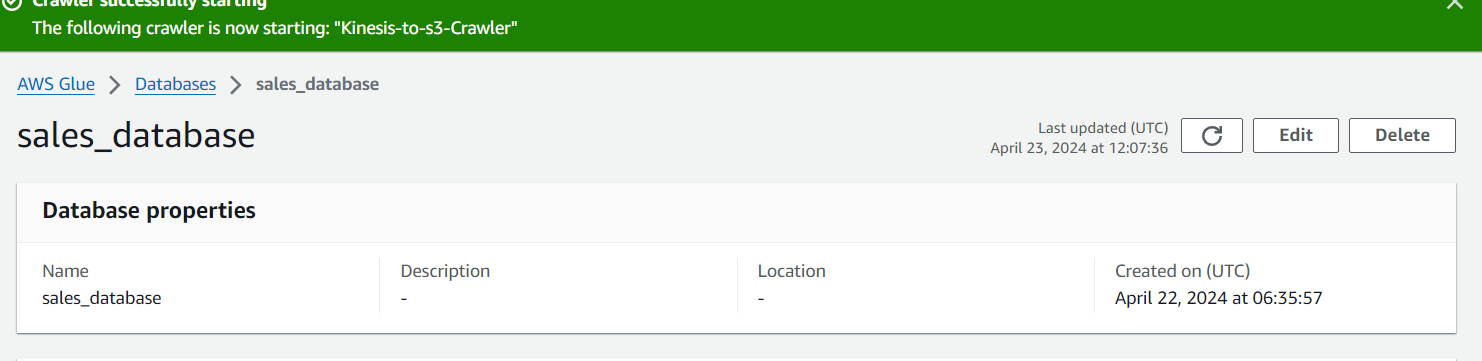


We have other files also

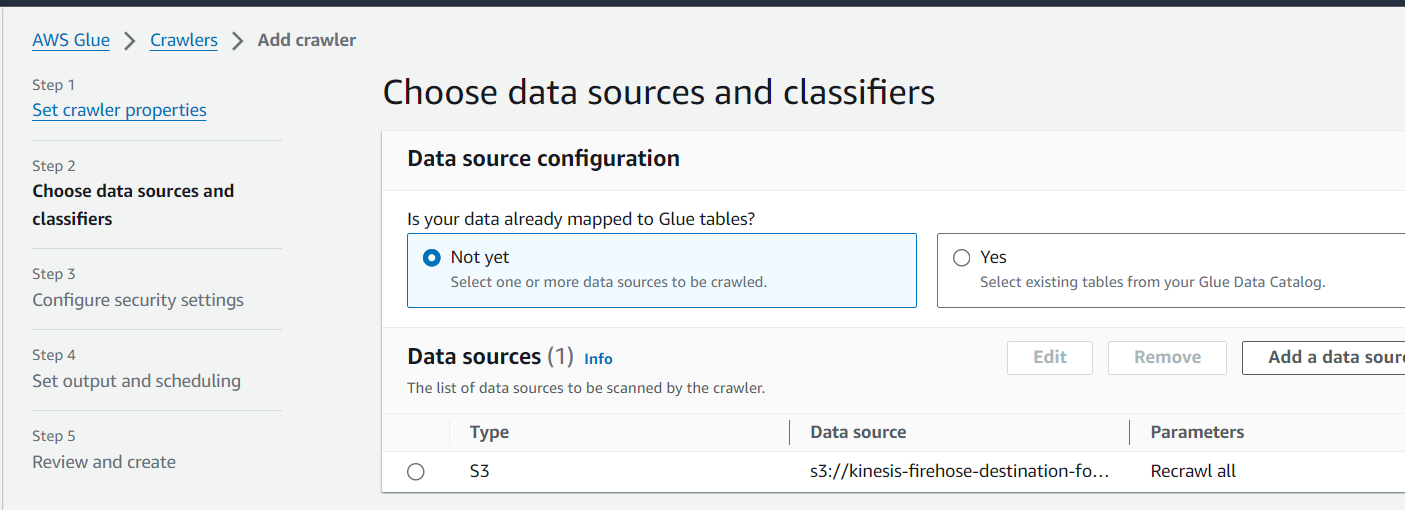


Next Glue Part

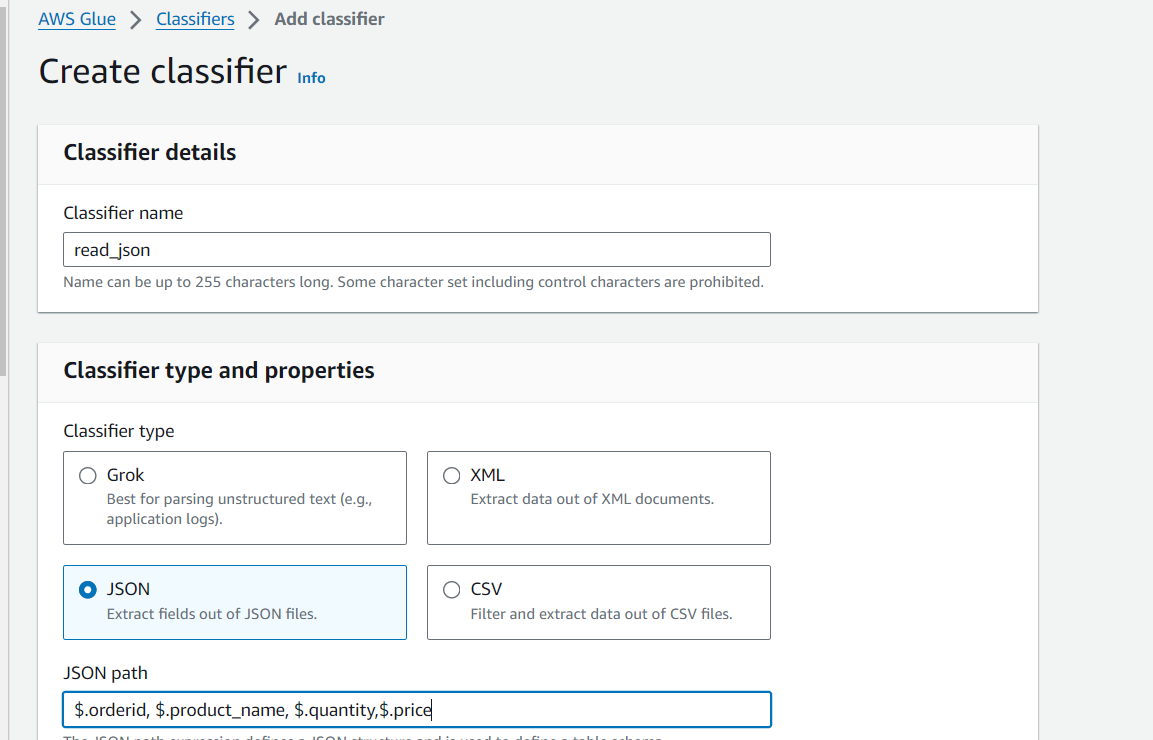
Create Database in Glue



Create Crawler and add S3 as source and choose Bucket in that

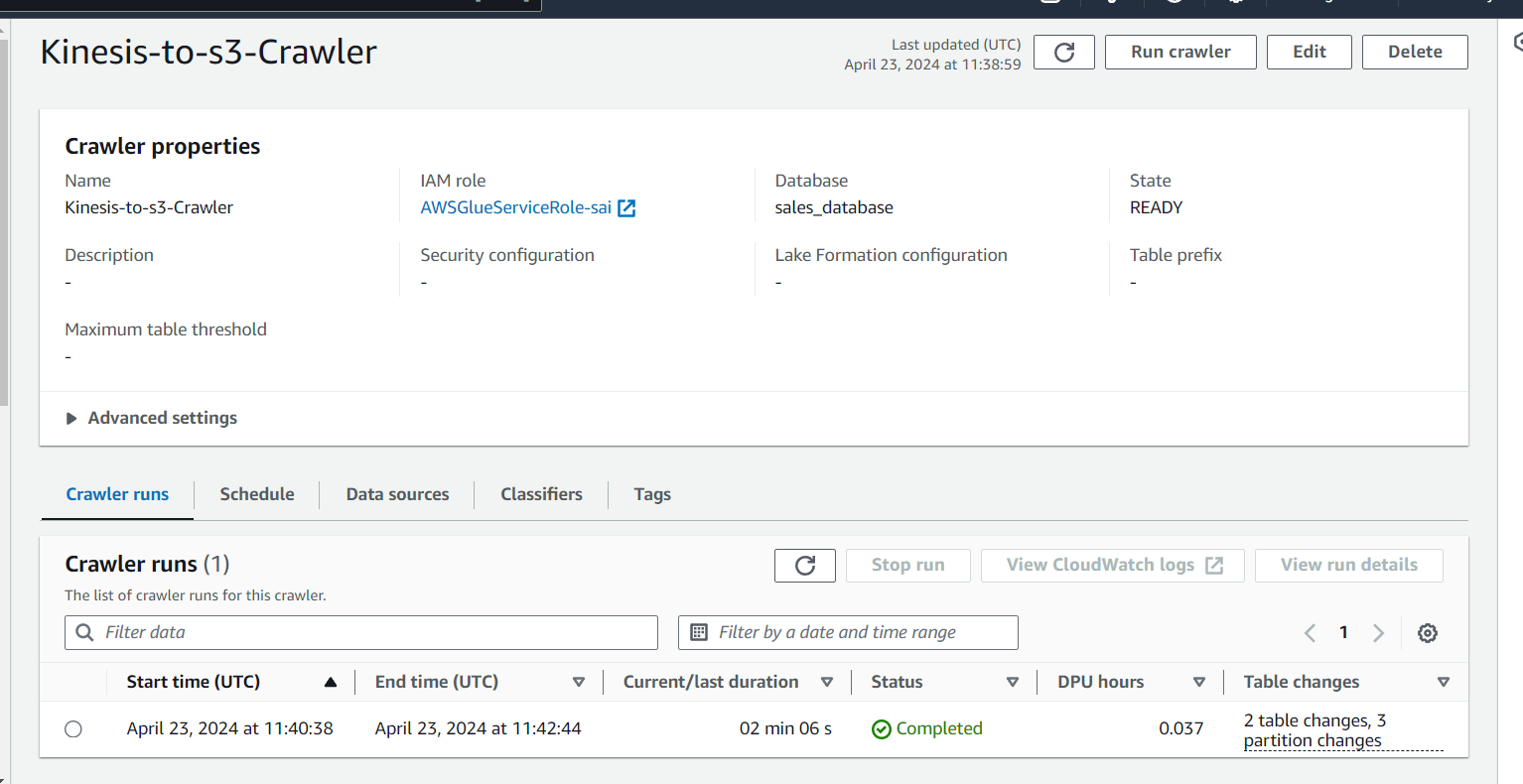


Next add Custom Classifiers and Add database and create

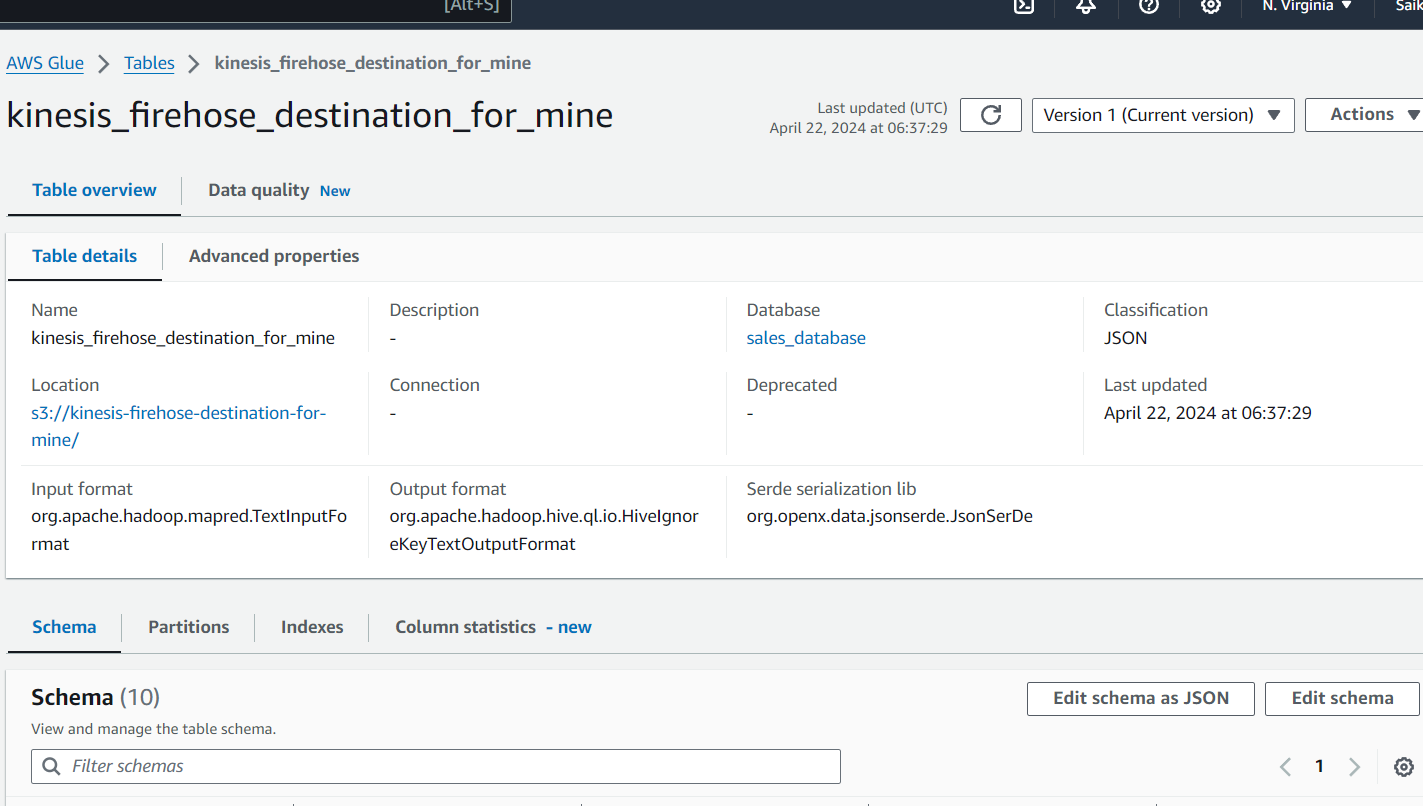


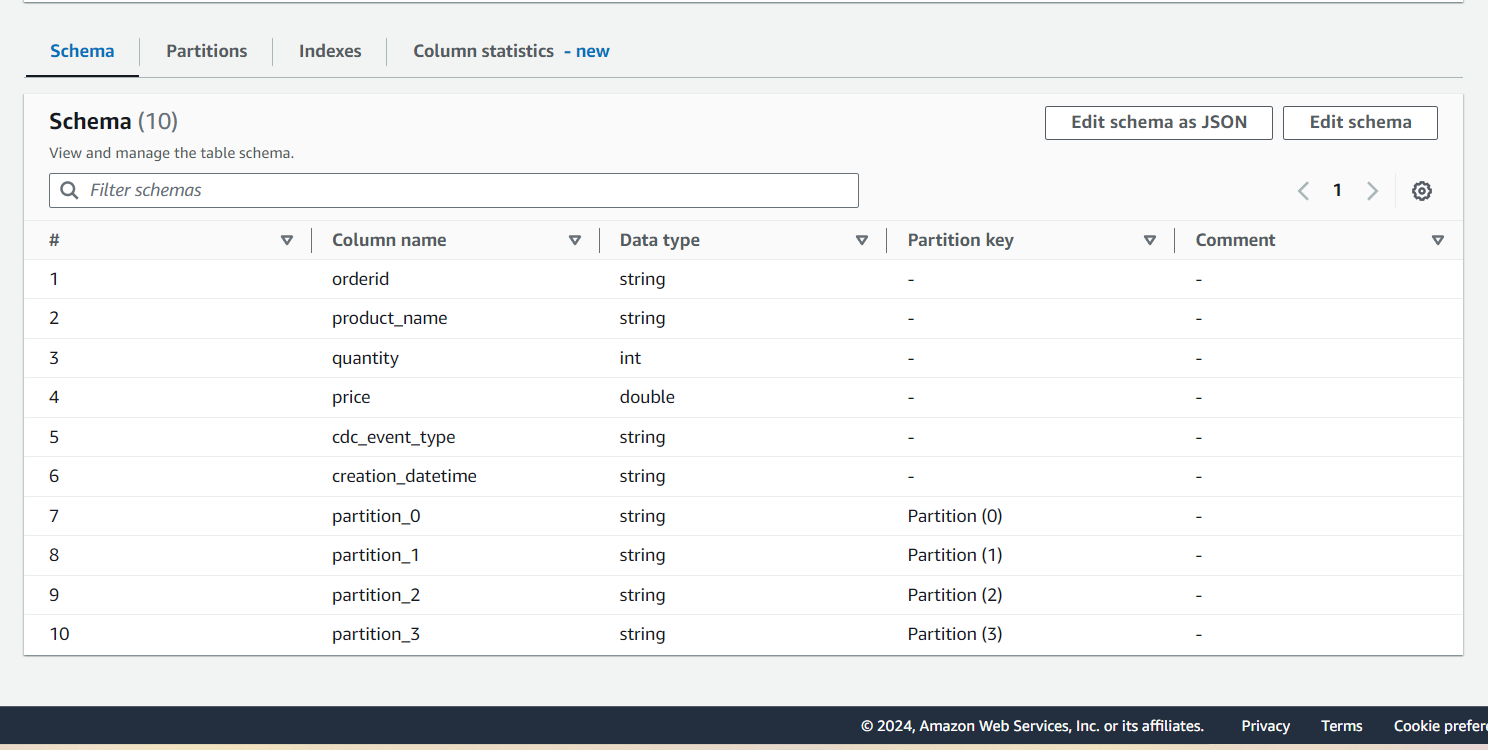
**Crawler created**

create and Run it so it will collect metadata from s3 bucket files and store metadata in Tables

****

**See table and partitions created in Glue**

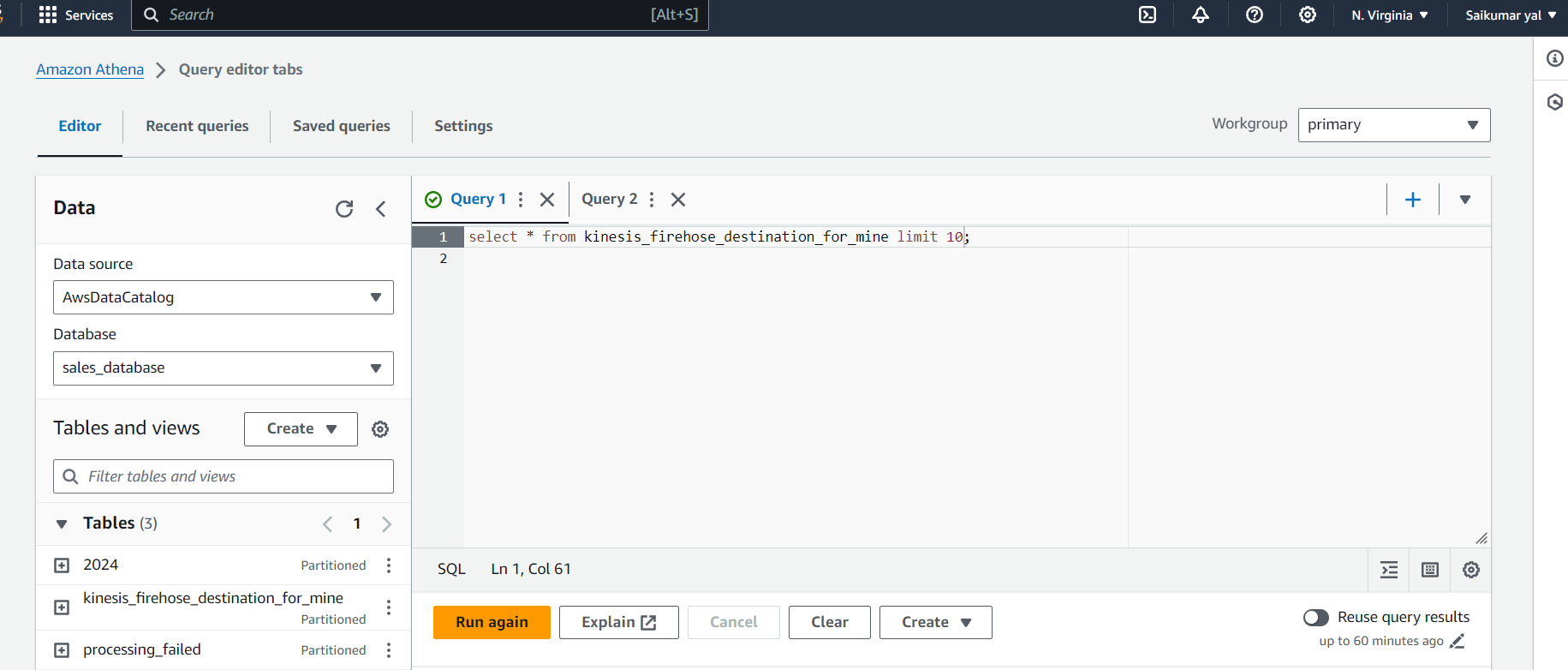
****

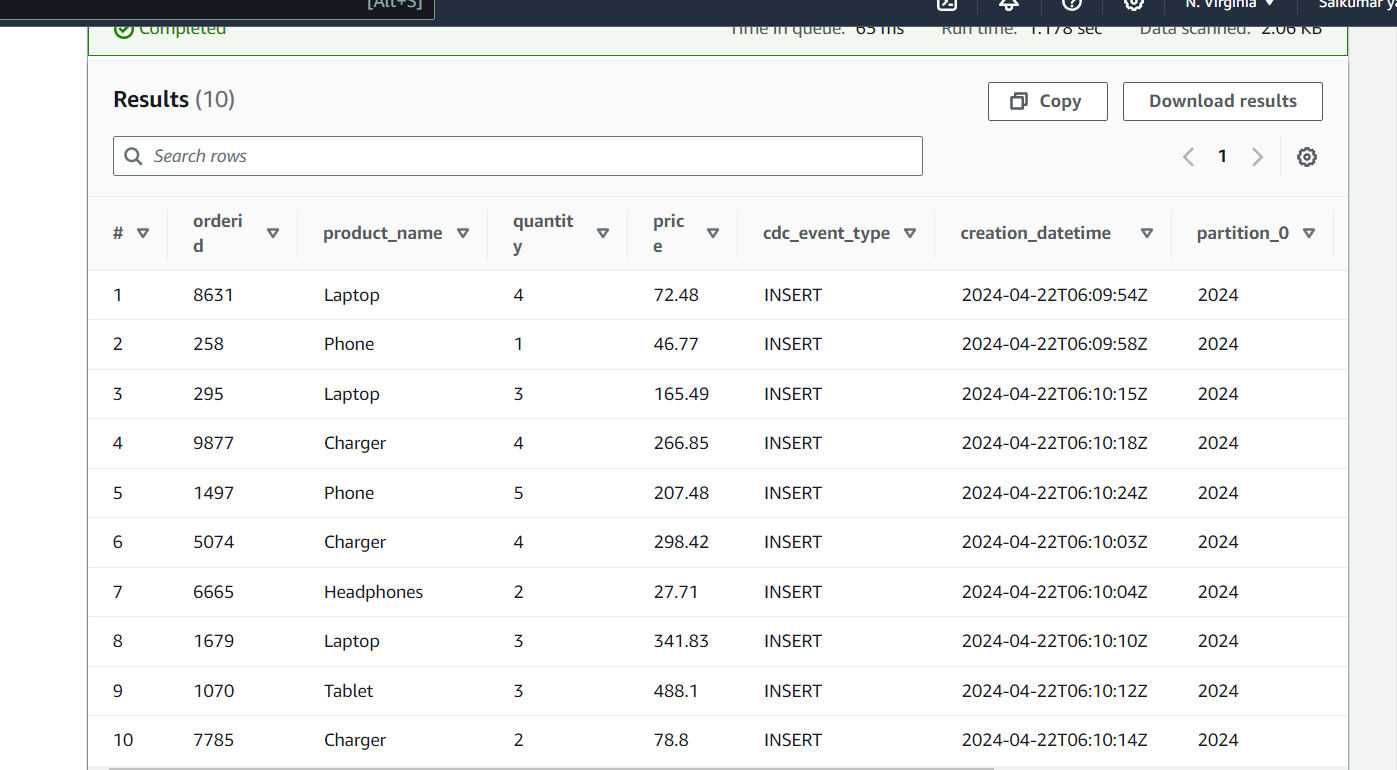
****

**🡪Next Open Athena**

Select glue database of this project and do select query from table name

select \* from kinesis\_firehose\_destination\_for\_mine limit 10;

****



And hence we can query from Athena and check .