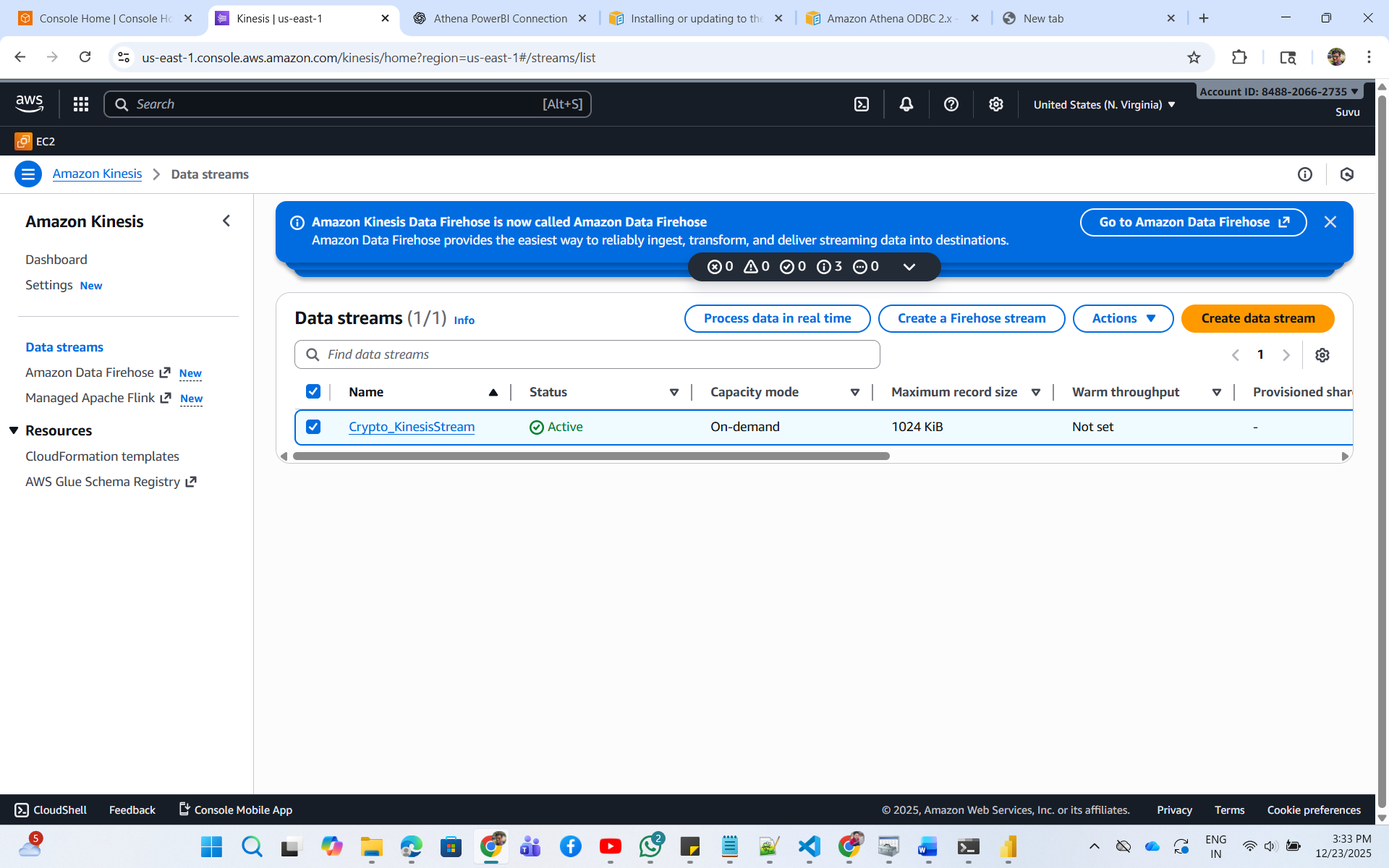
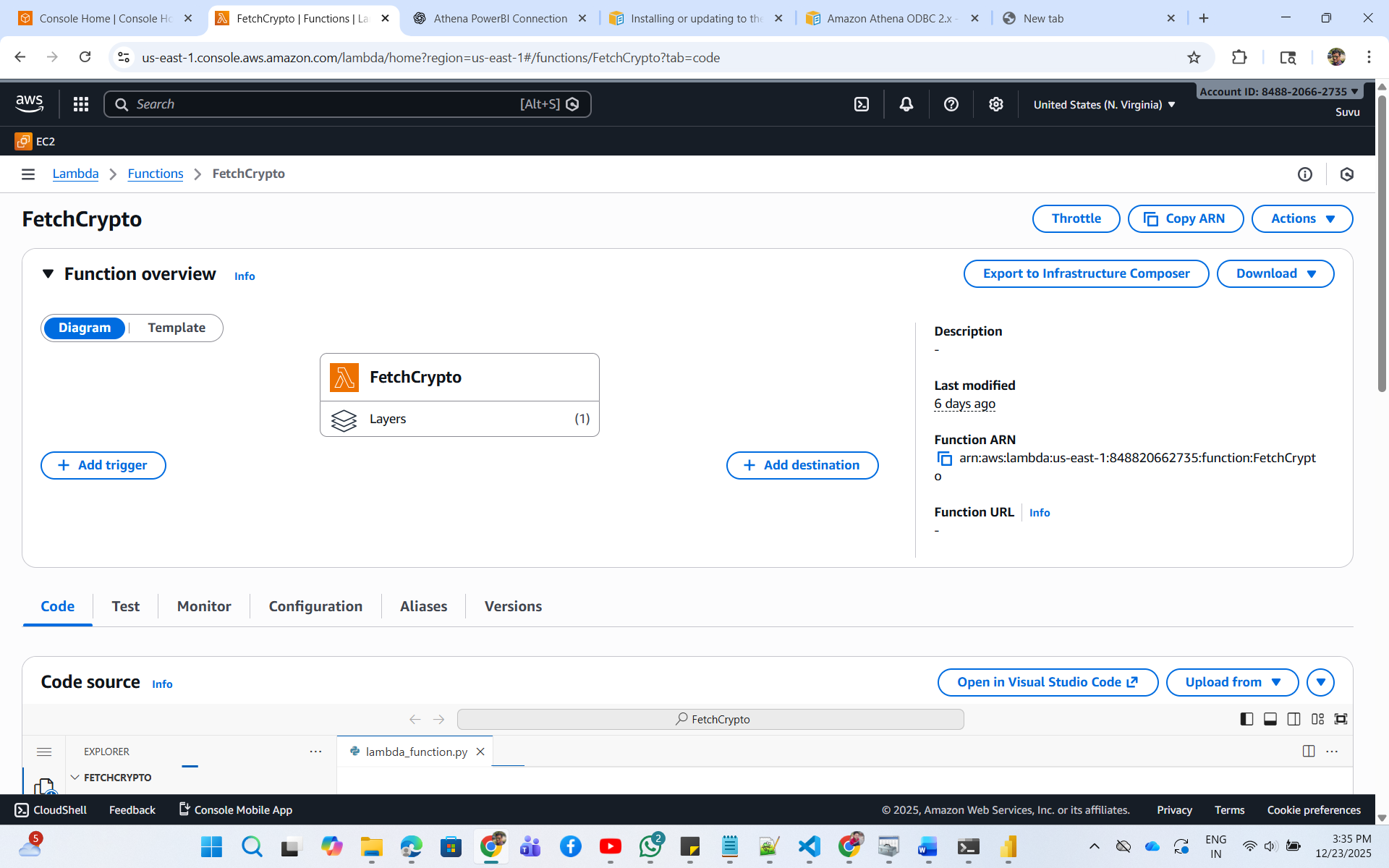
CrytoCurrency Project-

1. Kinesis-



1. Lambda Functions-

LoadCrypto:



import json

import requests

import os

import boto3

def lambda\_handler(event, context):

    # aws kinesis

    kinesis = boto3.client('kinesis', region\_name='us-east-1')

    crypto\_map = {

            'bitcoin': 'BTC',

            'ethereum': 'ETH',

            'solana': 'SOL',

            'cardano': 'ADA'

        }

    data = []

    try:

        for crypto\_full,crypto\_symbol in crypto\_map.items():

            url = "https://api.coingecko.com/api/v3/simple/price?vs\_currencies=usd&ids="+crypto\_full+"&names="+crypto\_full.capitalize()+"&symbols="+crypto\_symbol

            headers = {"x-cg-demo-api-key": os.environ['coingecko\_api\_key']}

            response = requests.get(url, params={'vs\_currencies': 'usd', 'include\_market\_cap': 'true', 'include\_24hr\_vol': 'true',

            'include\_24hr\_change': 'true'}, headers=headers)

            # print(response.text)

            kinesis.put\_record(

                StreamName=os.environ['KINESIS\_DATASTREAM'],

                Data=json.dumps(response.json()),

                PartitionKey=crypto\_full

            )

            print("Data sent to Kinesis")

            data.append({

                'crypto': crypto\_full,

                'price': response.json()[crypto\_full]['usd'],

                'market\_cap': response.json()[crypto\_full]['usd\_market\_cap'],

                '24hr\_vol': response.json()[crypto\_full]['usd\_24h\_vol'],

                '24hr\_change': response.json()[crypto\_full]['usd\_24h\_change']

            })

        print(data)

    except Exception as e:

        print(e)

        raise e

        exit(1)

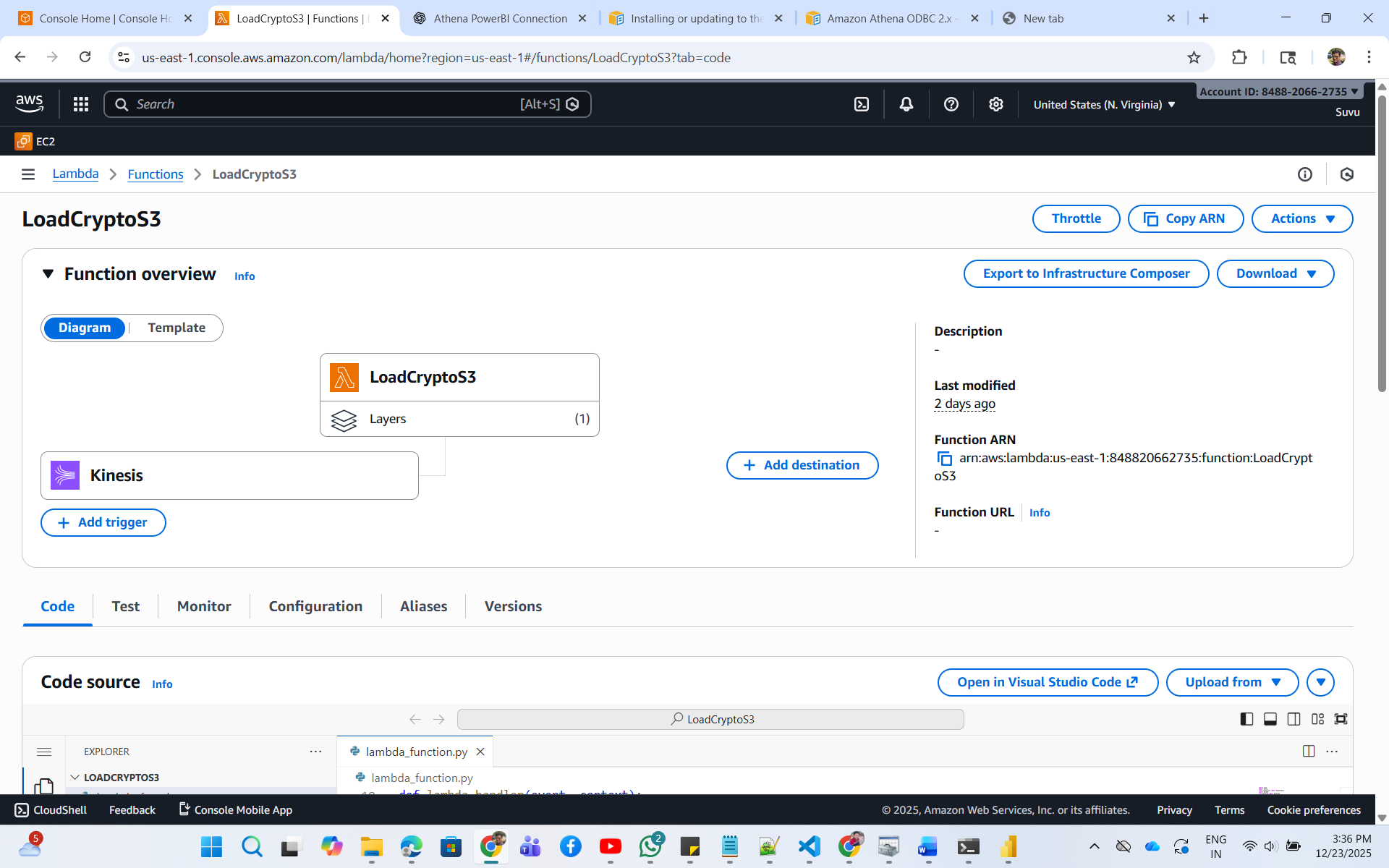
    return {

        'statusCode': 200,

        'body': json.dumps('Executed Successfully!!')

    }

Lambda2- LoadCryptoS3



import json

import boto3

from collections import defaultdict

import base64

from datetime import datetime

import logging

logger = logging.getLogger()

logger.setLevel(logging.INFO)

s3 = boto3.client('s3')

# CONFIG

BUCKET = "suvus3bucket"

WINDOW = 10  # moving average window

price\_buffer = defaultdict(list)  # in-memory buffer

def lambda\_handler(event, context):

    """

    Lambda triggered by Kinesis.

    Each record from Kinesis should contain JSON like:

    {

        "bitcoin": {

            "usd": 87037,

            "usd\_market\_cap": 1738332206735.3381,

            "usd\_24h\_vol": 45102475159.332664,

            "usd\_24h\_change": 1.5662022163293967

        }

    }

    """

    logger.info("EVENT = %s", json.dumps(event))

    if 'Records' not in event:

        logger.error("Not a Kinesis event: %s", event)

        return {

            'statusCode': 400,

            'body': 'Not a Kinesis event'

        }

    for record in event['Records']:

        # Kinesis data is base64 encoded

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

        data = json.loads(payload)

        logger.info("Decoded payload: %s", data)

        # Extract crypto info

        crypto\_id = list(data.keys())[0]      # 'bitcoin'

        crypto\_data = data[crypto\_id]

        ts = datetime.utcnow()  # current processing time

        price\_usd = crypto\_data['usd']

        market\_cap\_usd = crypto\_data['usd\_market\_cap']

        hr24\_volume = crypto\_data['usd\_24h\_vol']

        hr24\_change = crypto\_data['usd\_24h\_change']

        # --- Compute simple moving average ---

        price\_buffer[crypto\_id].append(price\_usd)

        if len(price\_buffer[crypto\_id]) > WINDOW:

            price\_buffer[crypto\_id].pop(0)

        ma\_10 = sum(price\_buffer[crypto\_id]) / len(price\_buffer[crypto\_id])

        # --- Compute simple signal ---

        signal = "BUY" if price\_usd > ma\_10 else "SELL"

        # --- Prepare record for S3 ---

        enriched = {

            "timestamp": ts.isoformat() + "Z",

            "symbol": crypto\_id,

            "price": price\_usd,

            "ma\_10": round(ma\_10, 2),

            "signal": signal,

            "market\_cap\_usd": market\_cap\_usd,

            "hr24\_volume": hr24\_volume,

            "hr24\_change": hr24\_change

        }

        # --- Generate S3 key (partitioned by symbol/year/month/day) ---

        key = (

            f"symbol={crypto\_id}/"

            f"year={ts.year}/month={ts.month:02d}/day={ts.day:02d}/"

            f"data-{int(ts.timestamp())}.json"

        )

        # --- Write to S3 ---

        s3.put\_object(

            Bucket=BUCKET,

            Key=key,

            Body=json.dumps(enriched)

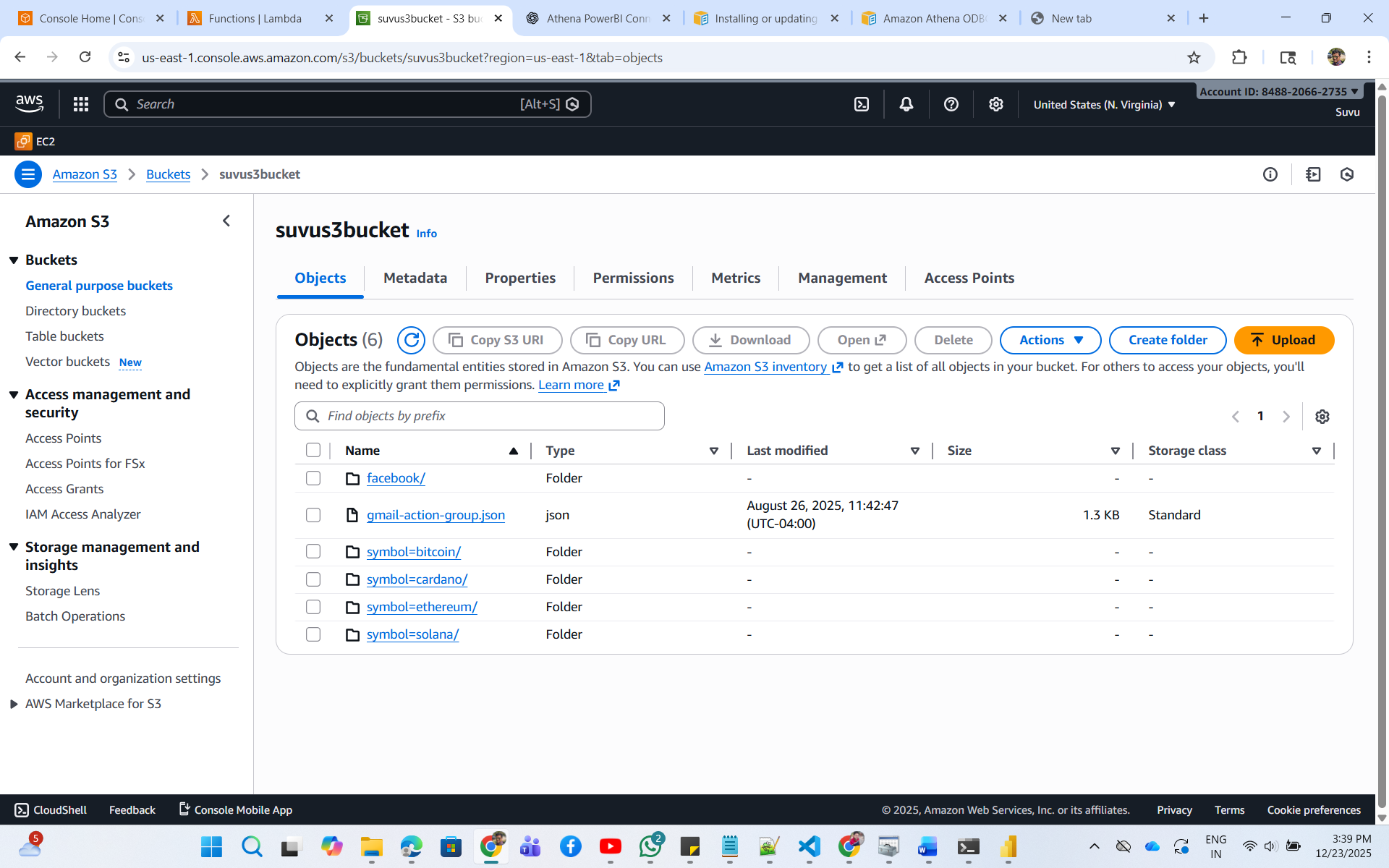
        )

        logger.info("Written record to S3: %s", key)

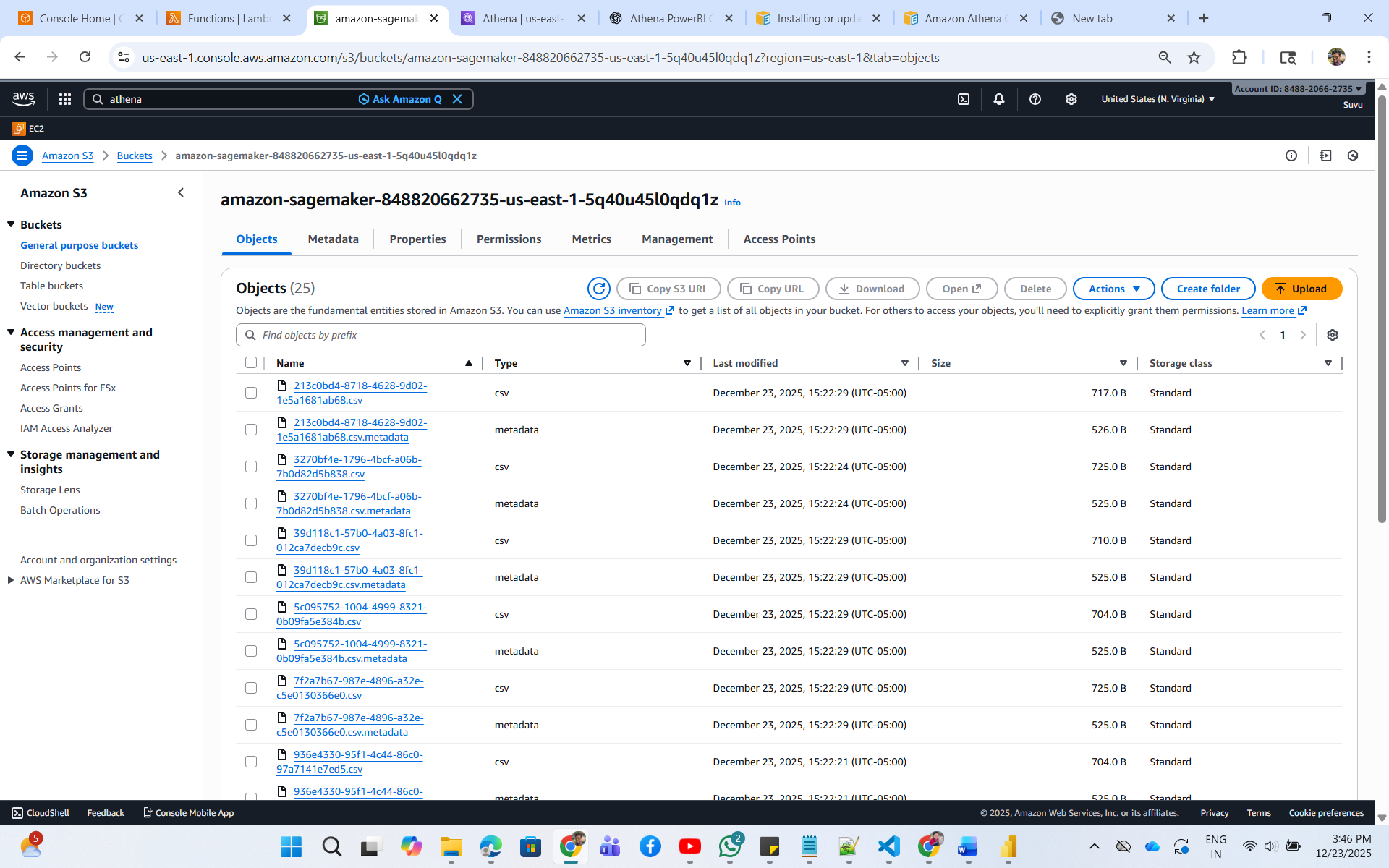
    return {"status": "success", "records\_processed": len(event['Records'])}

1. S3 bucket-

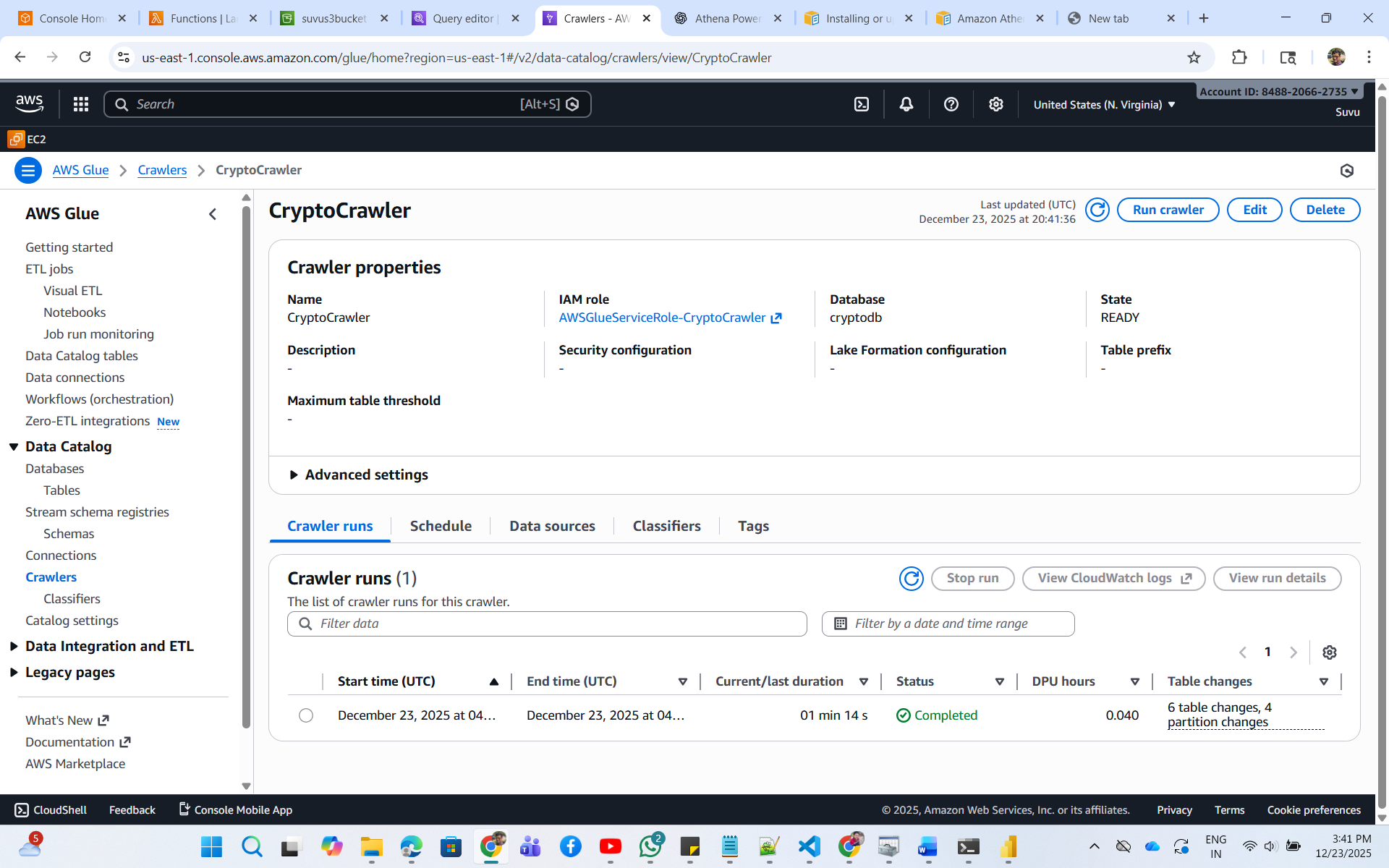
Lambda loads to S3 bucket:



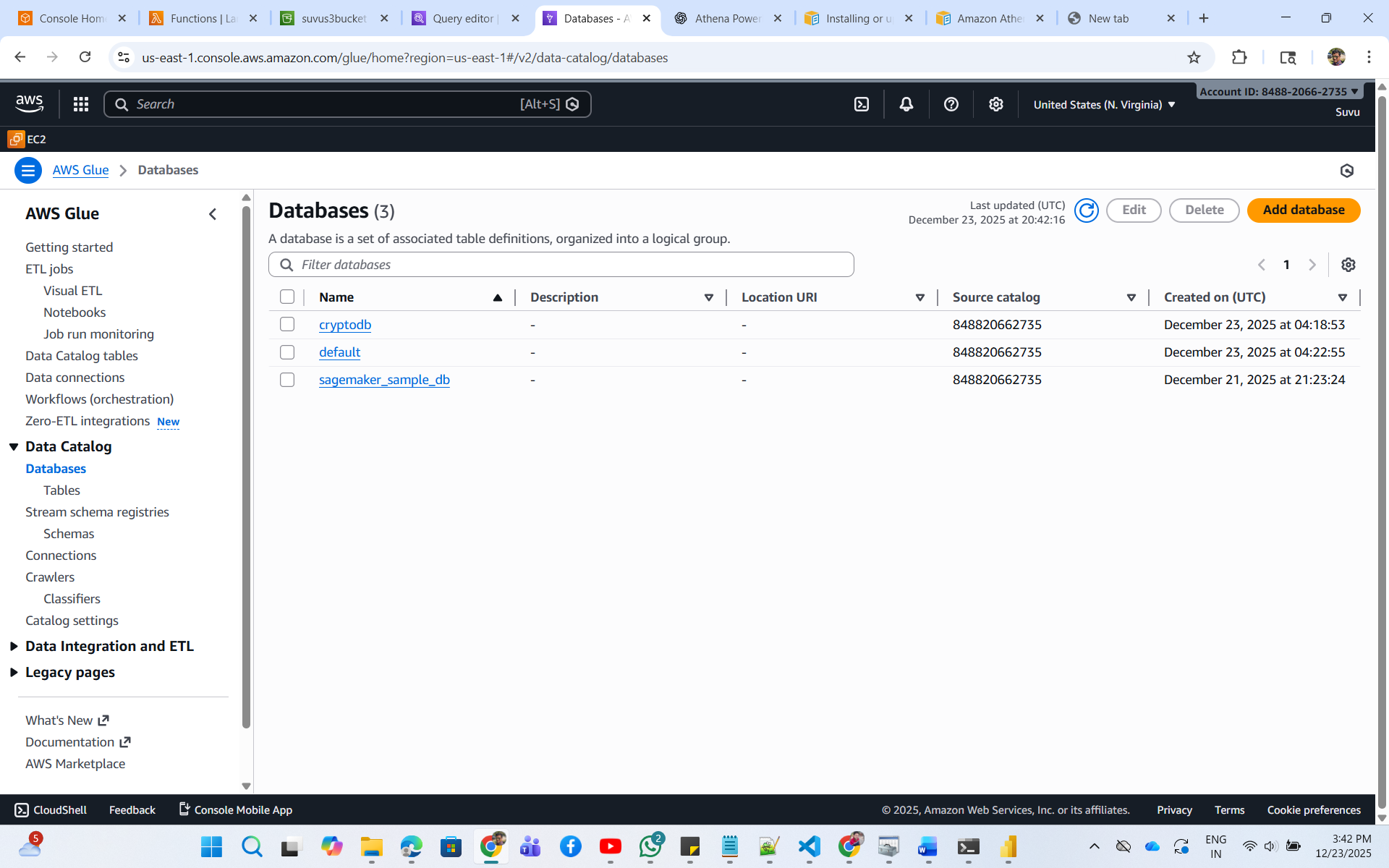
Athena Query results stores in S3-

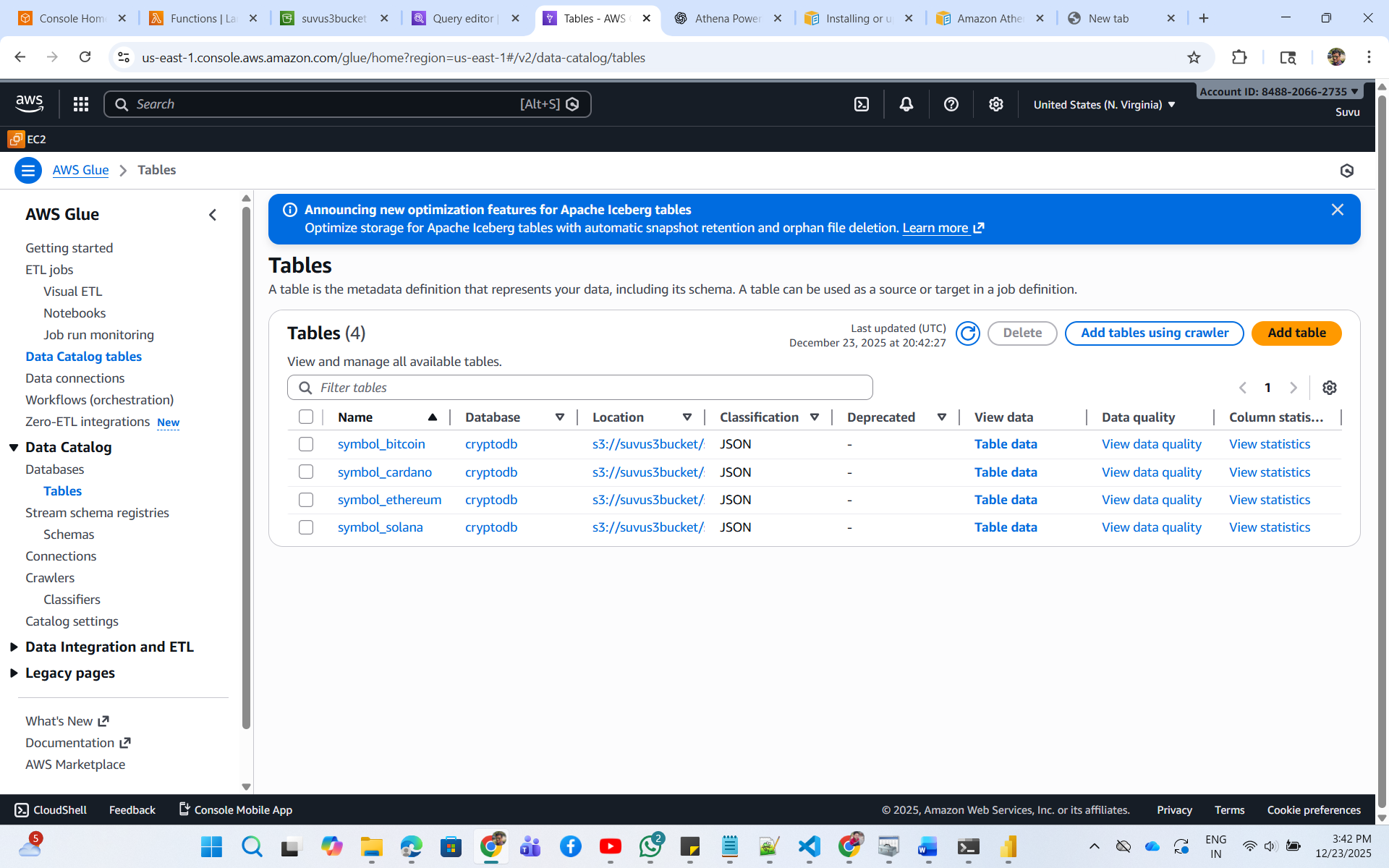


1. Glue Job- (Crawler Job)

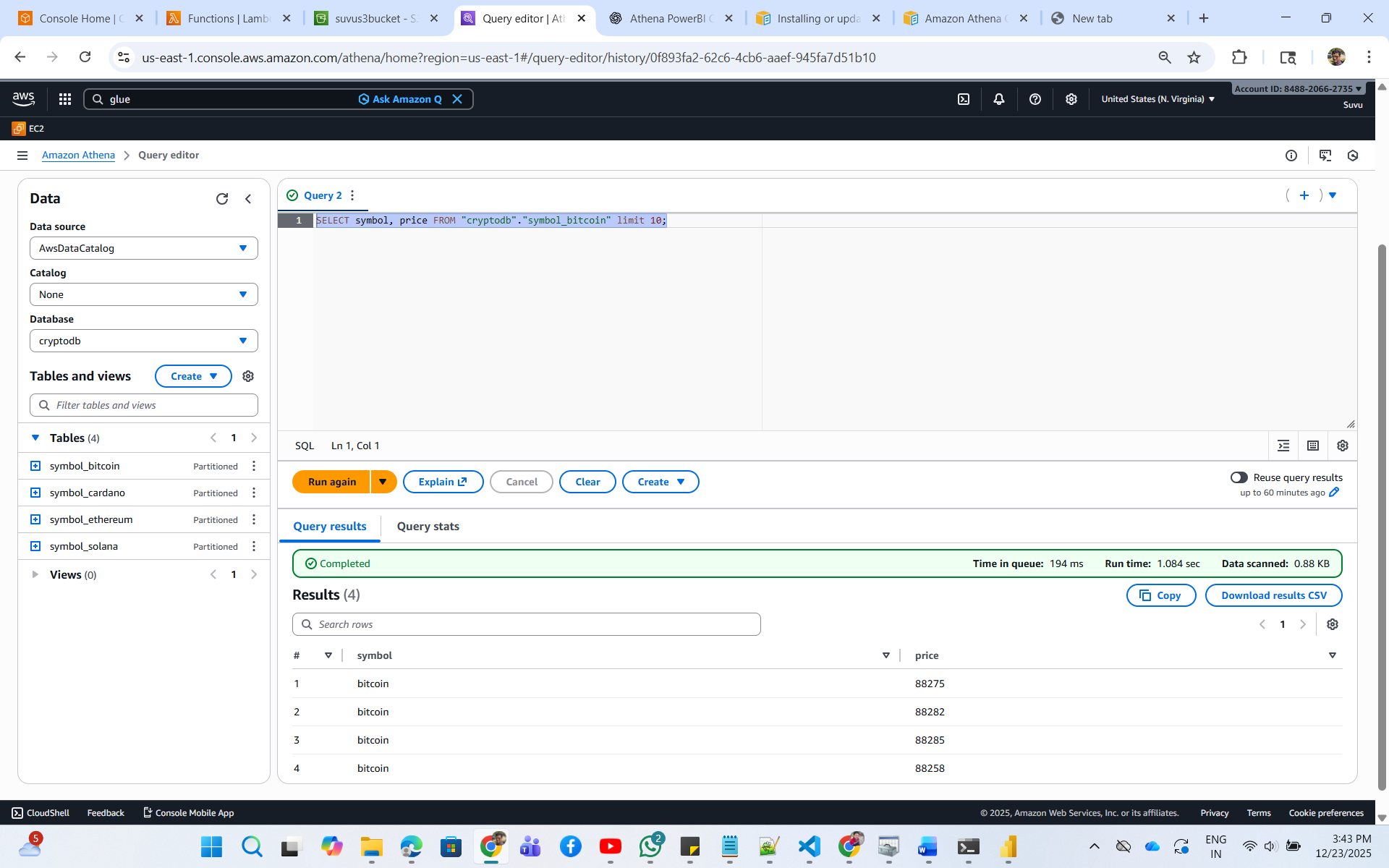


DB and Tables-

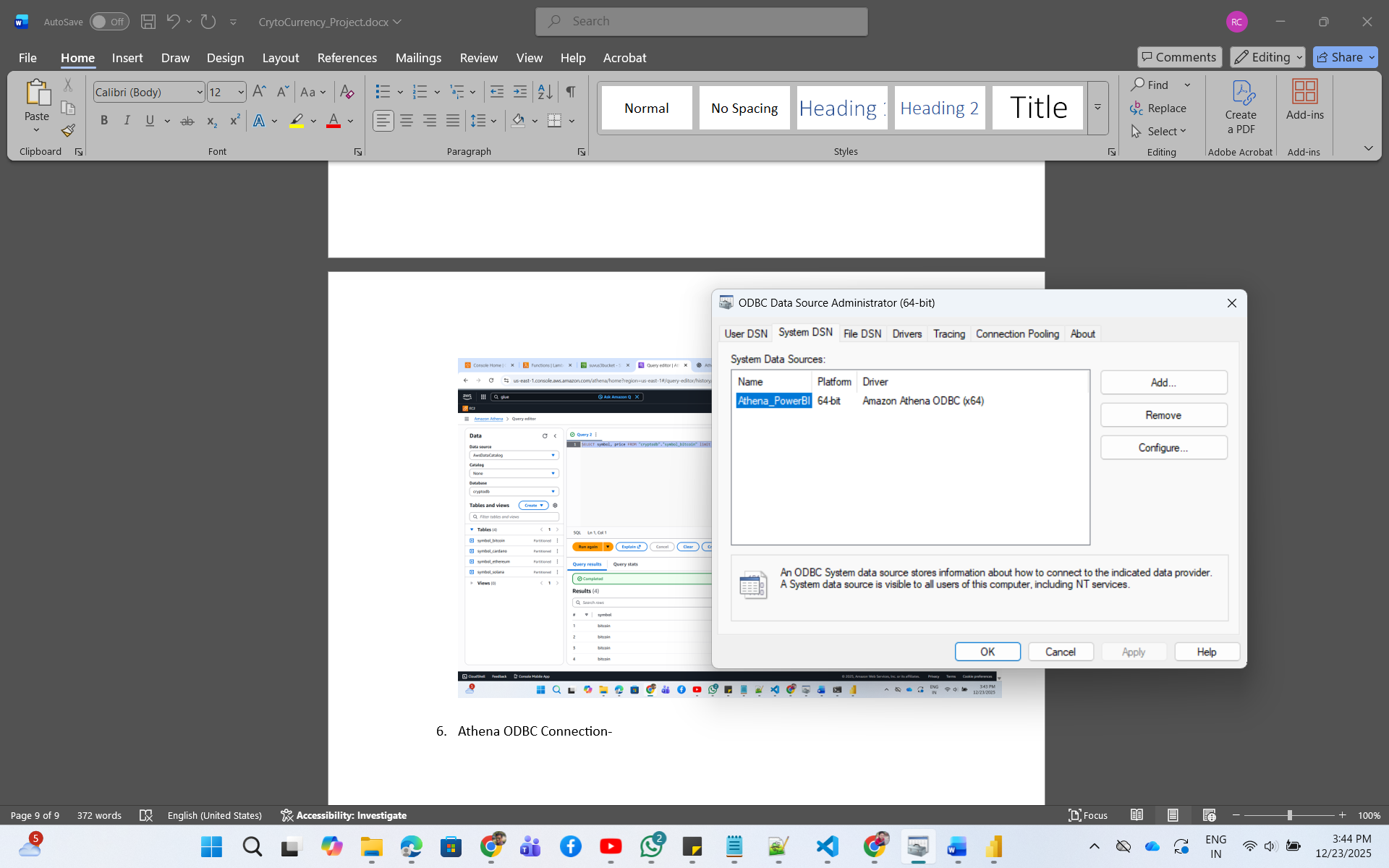


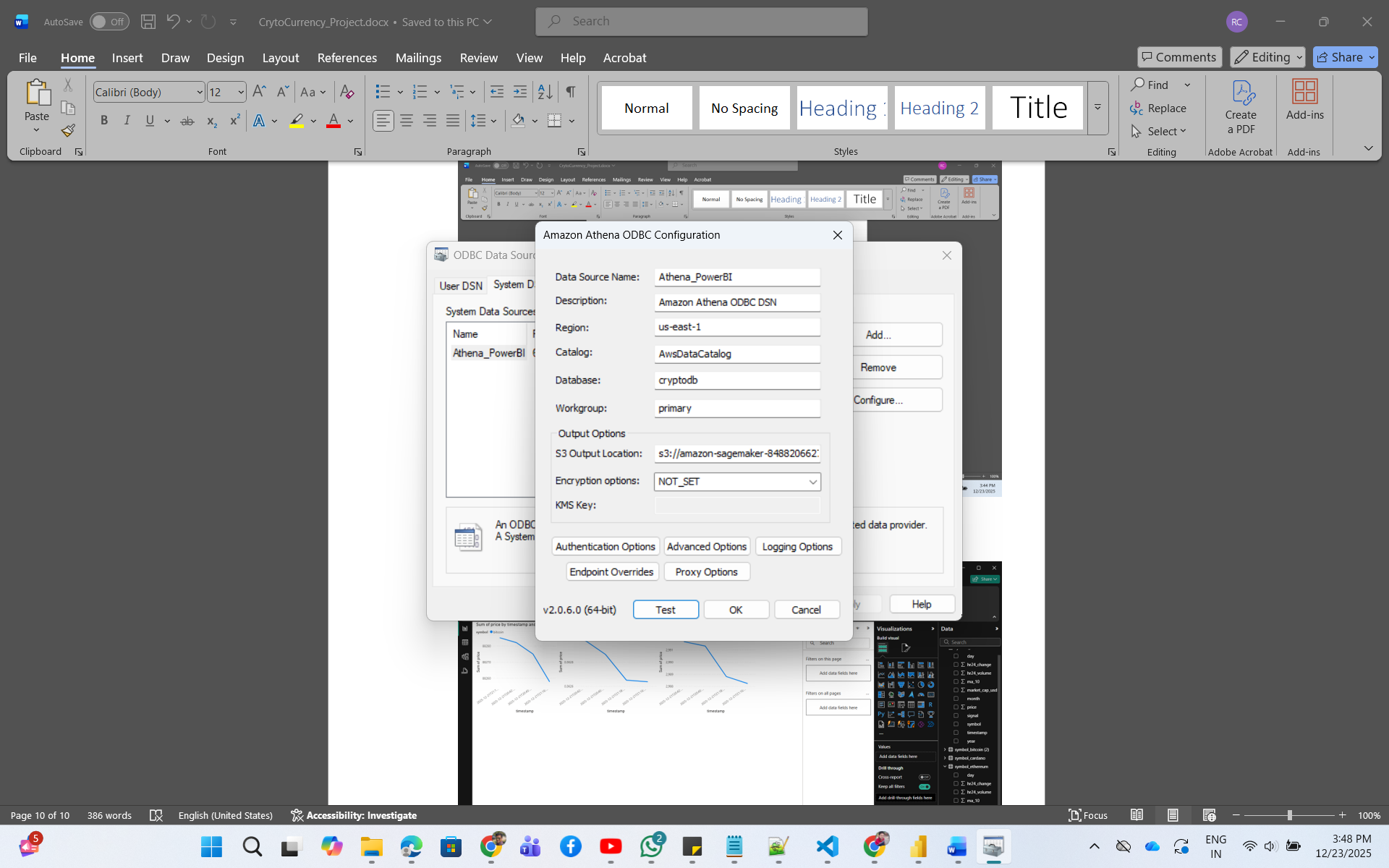


1. Athena-



1. Athena ODBC Connection-





1. PowerBI report-

