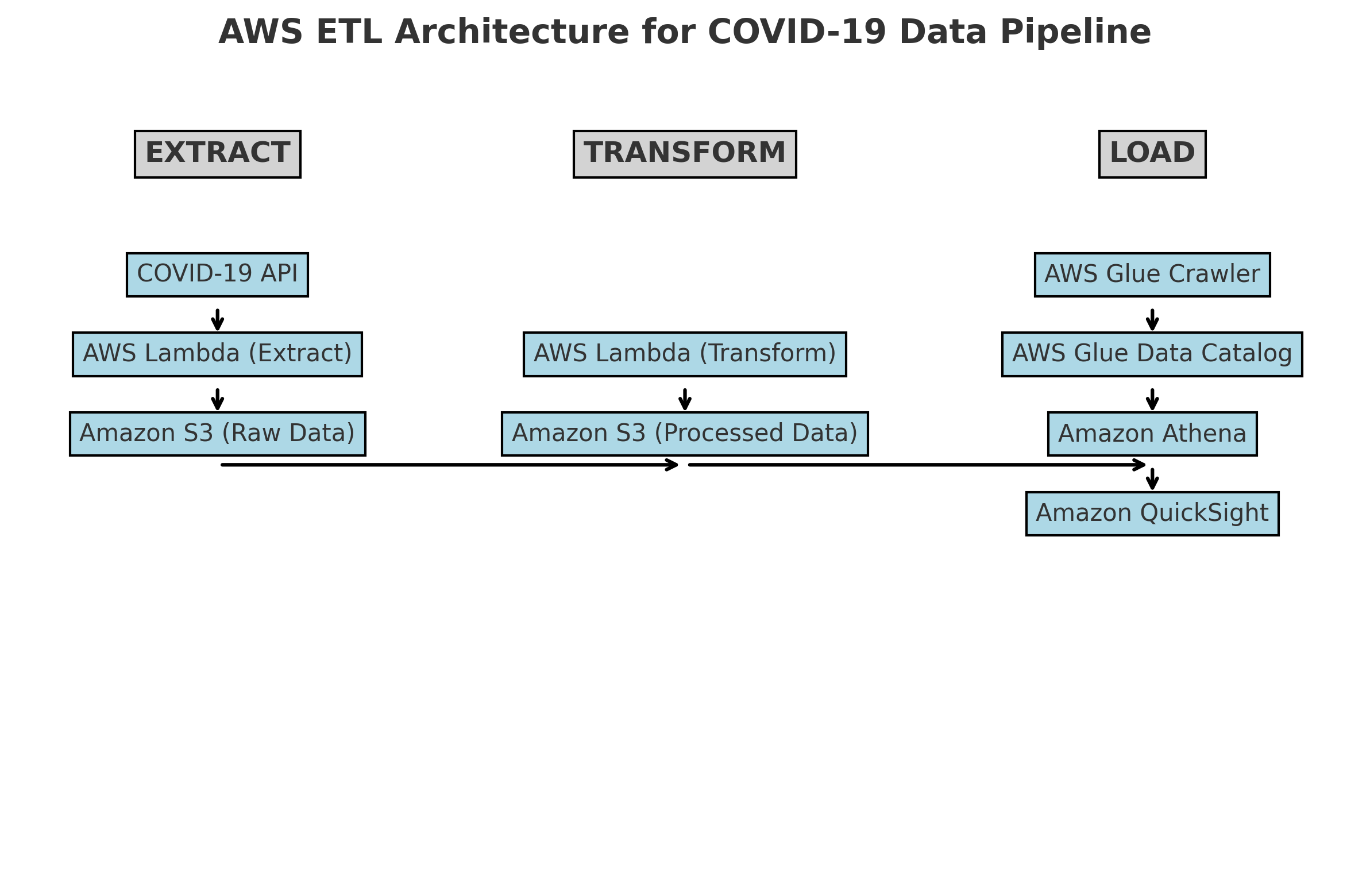
AWS ETL Pipeline for COVID-19 Data: End-to-End Project

* Goal:

Build an automated ETL pipeline using AWS services to extract COVID-19 data, transform it, and analyze it using AWS Athena & QuickSight.



**Step 1: Extract Data from COVID-19 API**

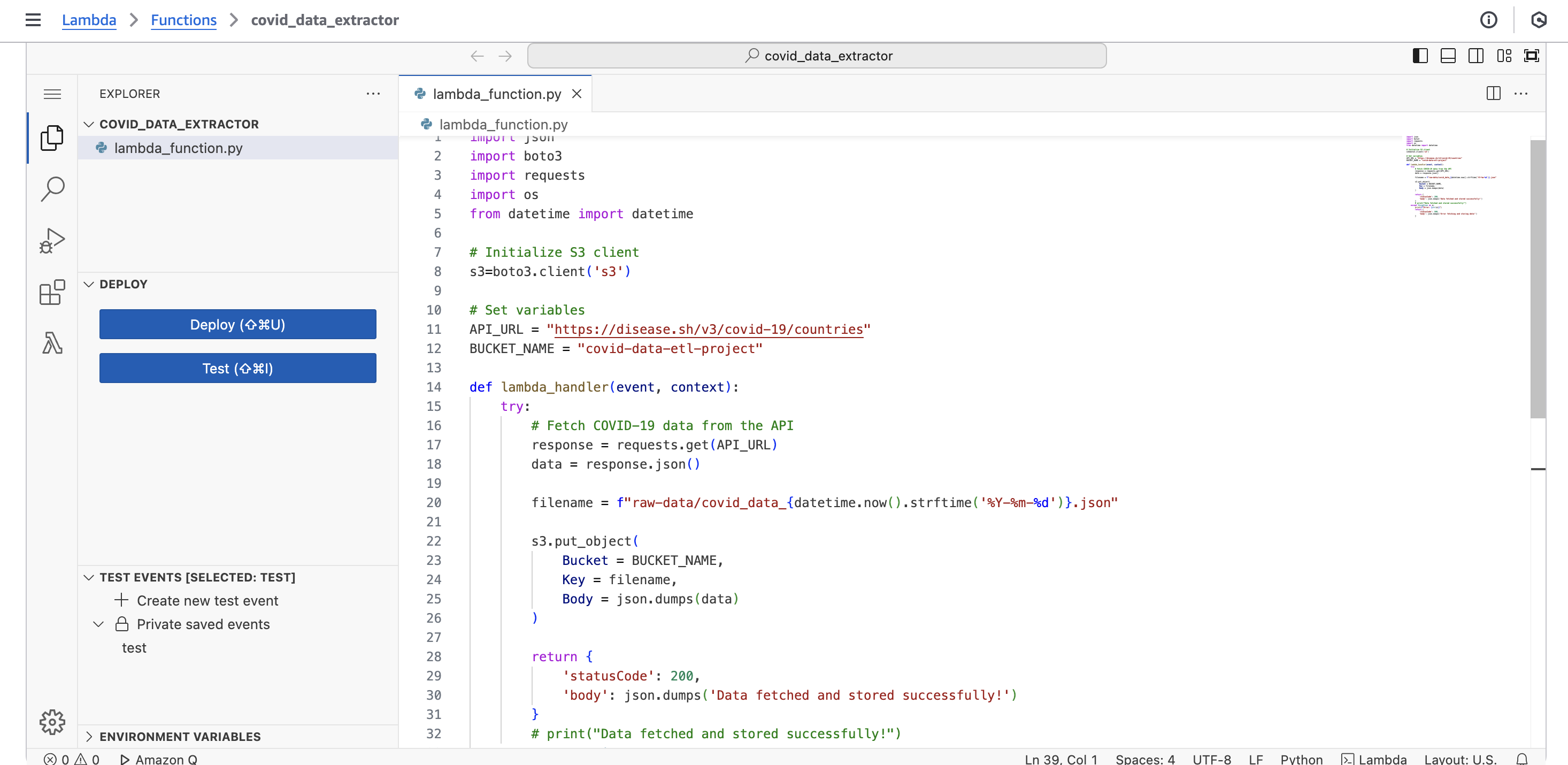
**Tools Used:**

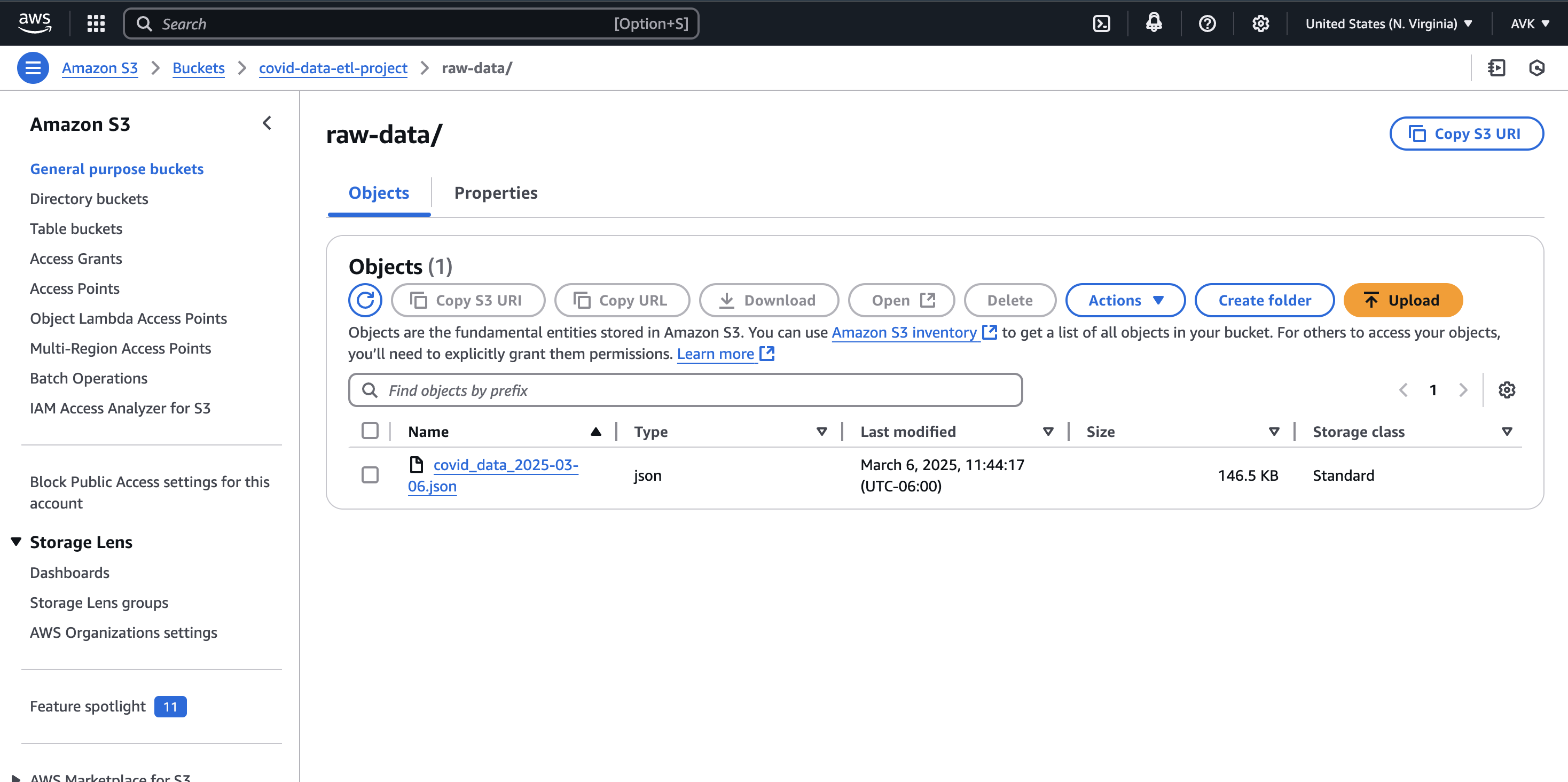
* AWS Lambda (Extract Function)
* Amazon S3 (Raw Data Storage)
* AWS CloudWatch (Scheduled Automation - Initially Disabled)

**WorkFlow:**

* **Created an AWS Lambda function (covid\_data\_extractor)**
* Used Python to **fetch COVID-19 data** from an external API.
* Converted it into **JSON format**.
* Stored the JSON data in **Amazon S3 (raw-data/ folder)**.

**Screenshots:**





**Step 2: Transform JSON Data to CSV**

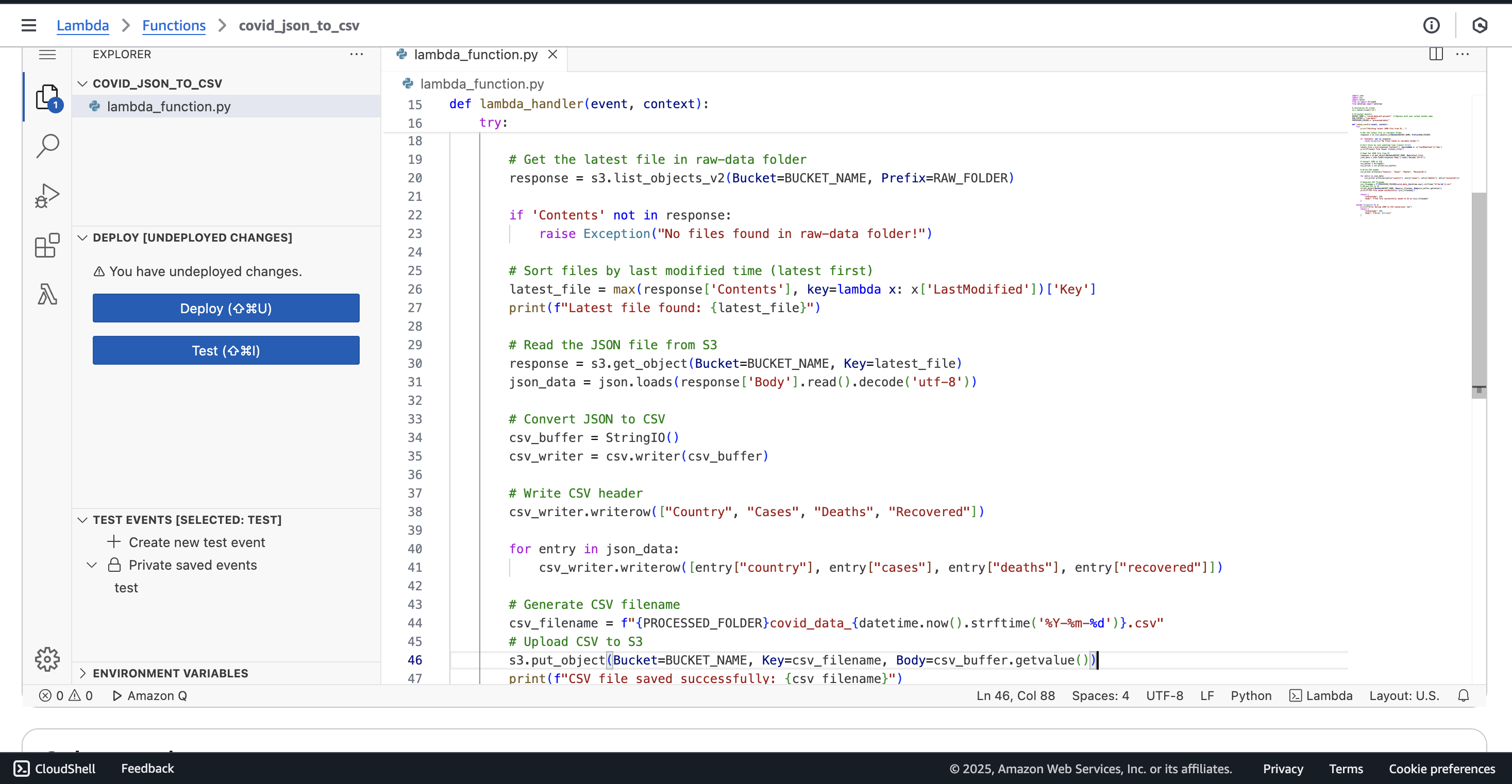
**Tools Used:**

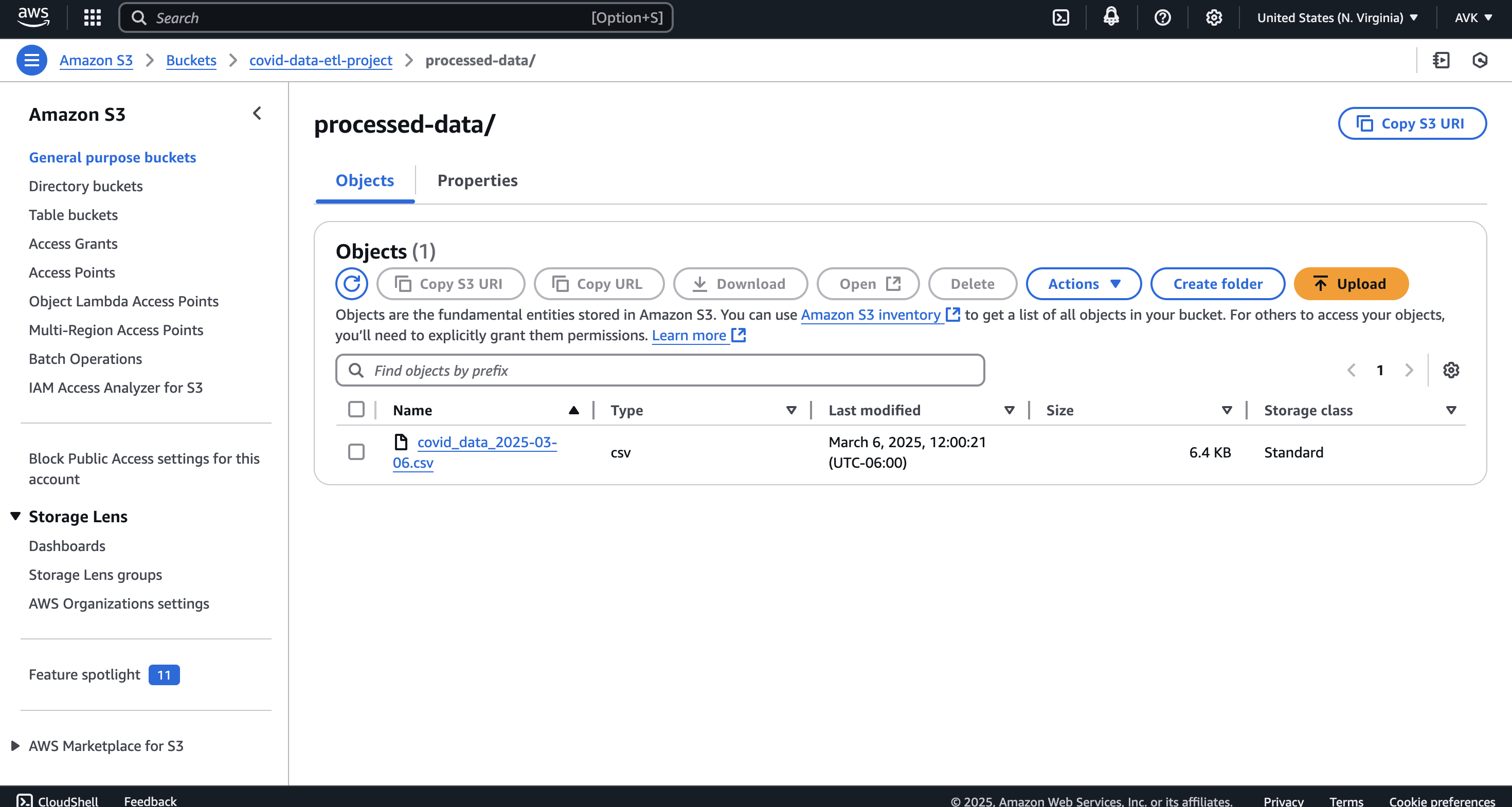
* AWS Lambda (Transform Function)
* Amazon S3 (Processed Data Storage)
* AWS CloudWatch (Scheduled Automation - Initially Disabled)

**WorkFlow:**

* Created another AWS Lambda function (covid\_json\_to\_csv)
* Read the latest JSON file from S3 (raw-data/).
* Converted JSON data to CSV format.
* Stored the processed CSV file in Amazon S3 (processed-data/ folder).

**Screenshots:**





**Step 3: Load Data into AWS Athena for Querying**

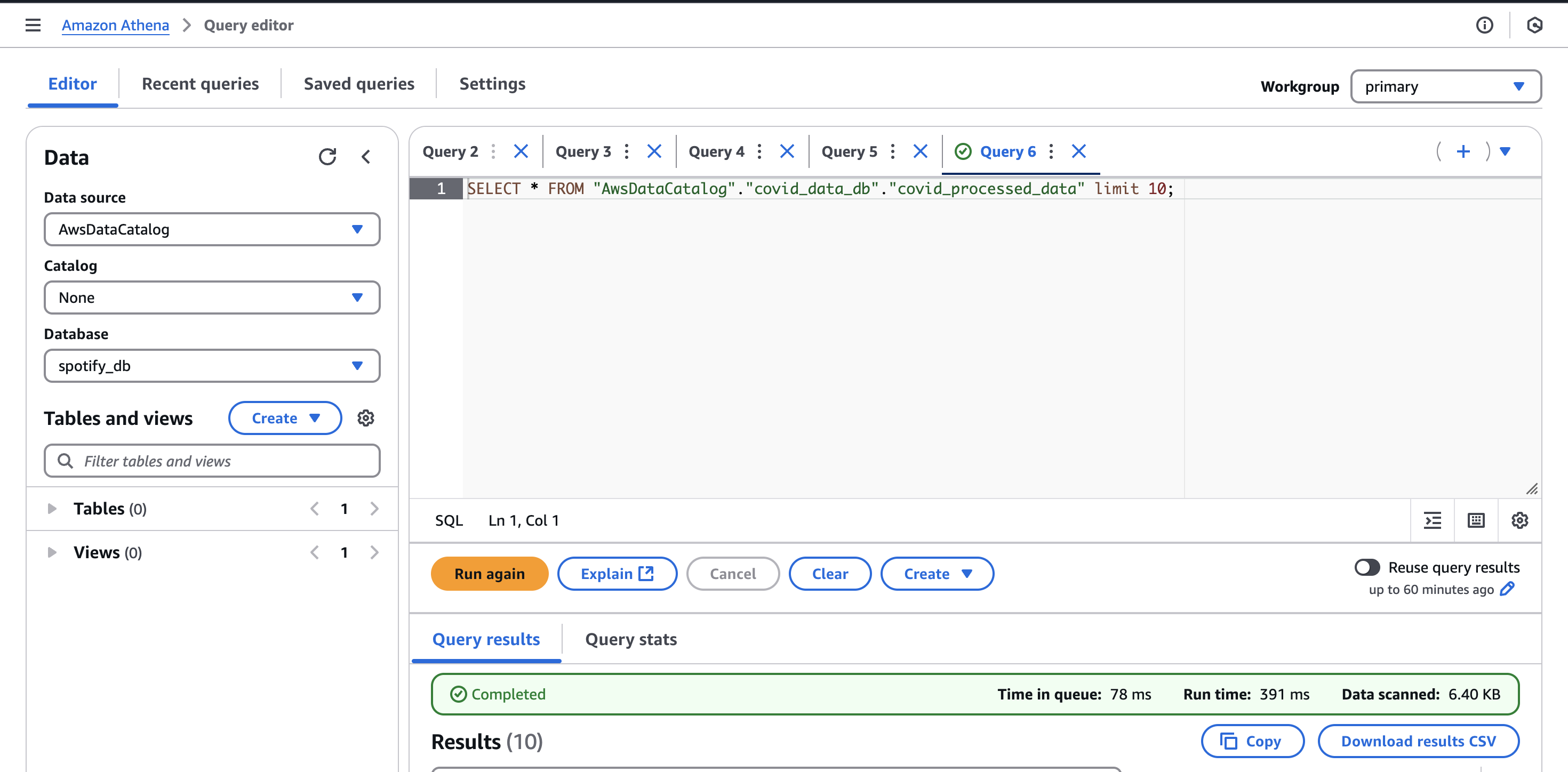
**Tools Used:**

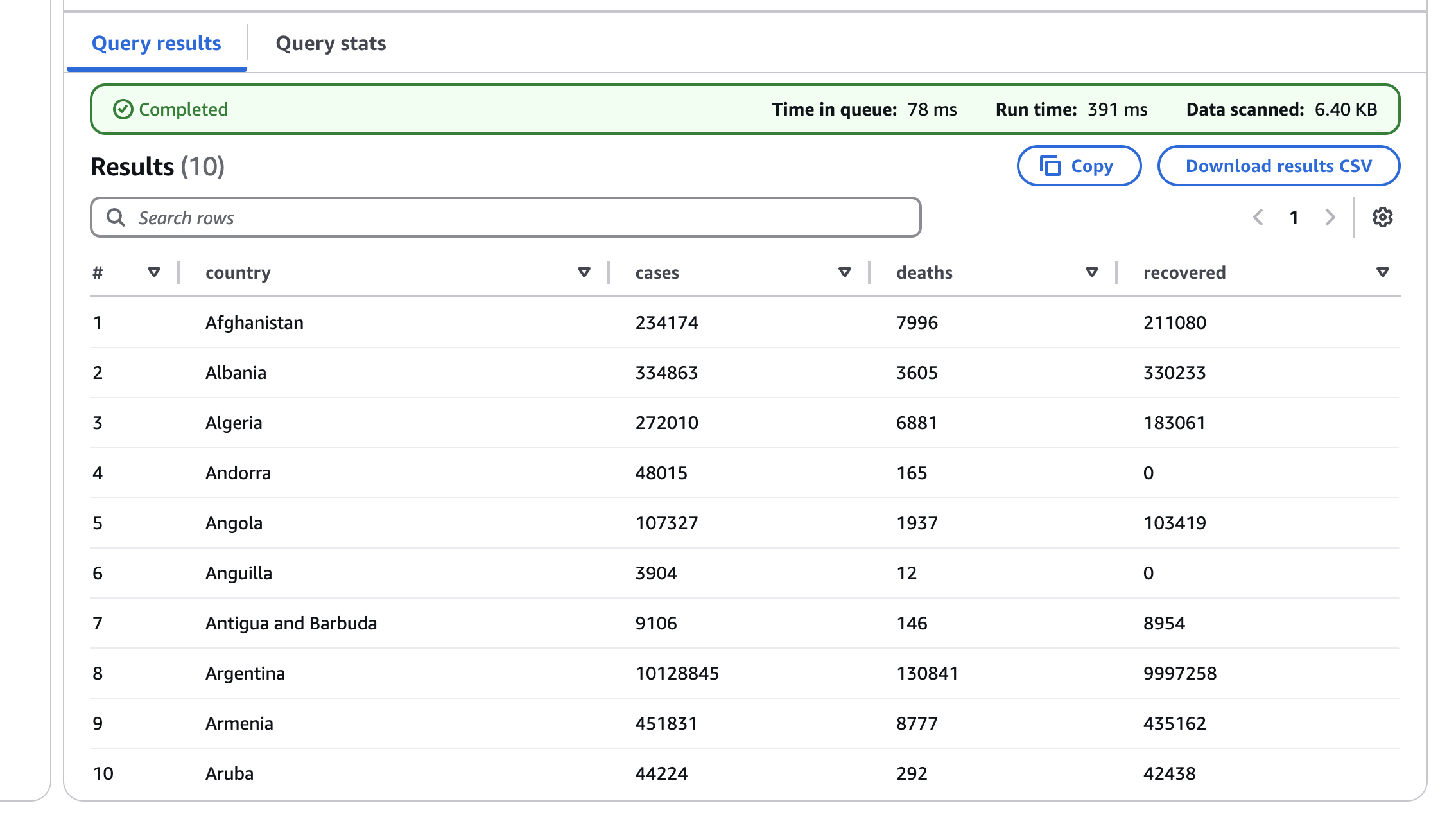
* AWS Glue Crawler (Schema Discovery)
* AWS Glue Data Catalog (Metadata Storage)
* AWS Athena (SQL Querying on S3 Data)

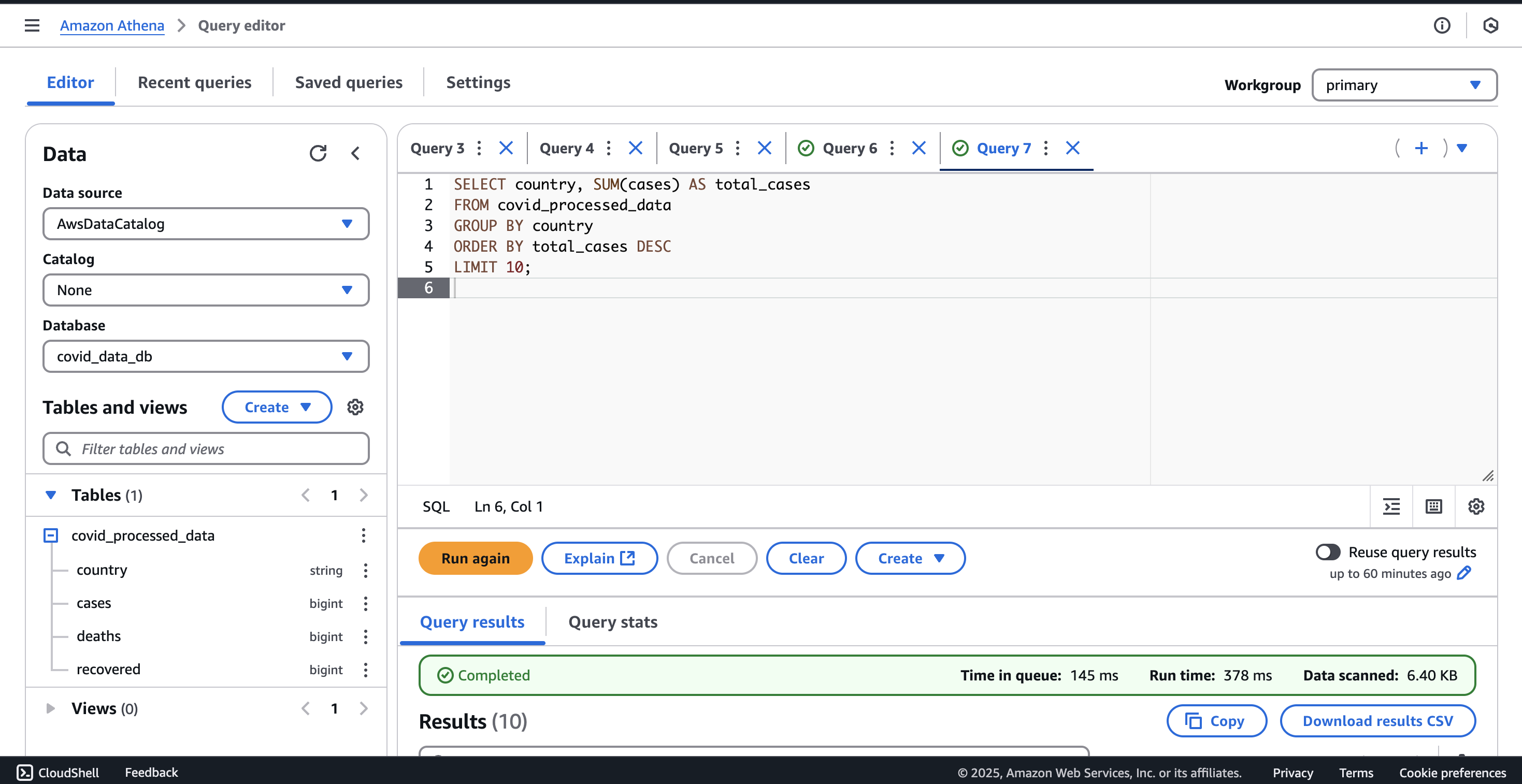
WorkFlow:

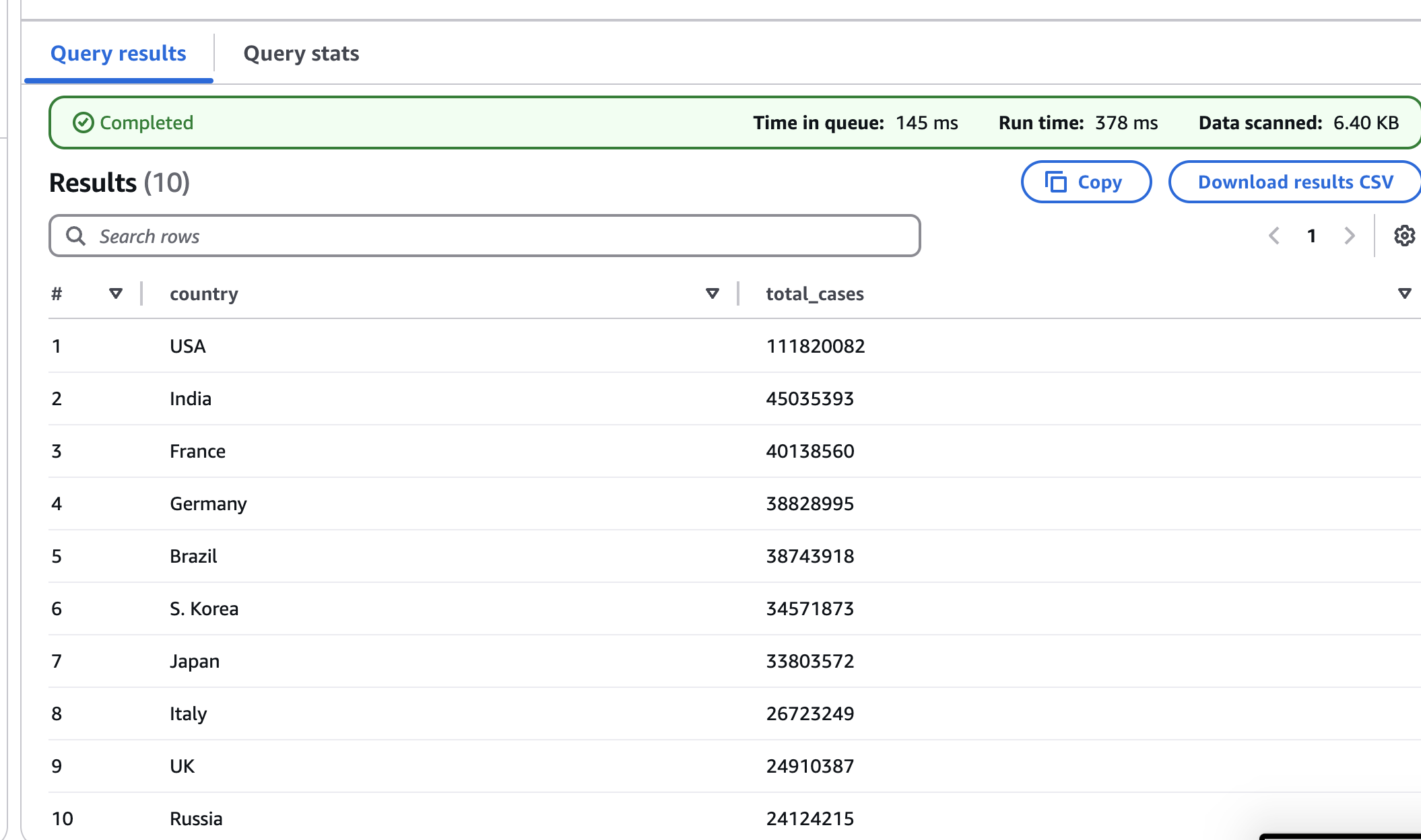
* Created an AWS Glue Crawler
* Scanned processed-data/ folder in S3 to automatically detect schema.
* Created a table covid\_data in AWS Glue Data Catalog.
* Queried the Data in AWS Athena

Screenshots:









**Step 4: Visualize Data in AWS QuickSight**

**Tools Used:**

Amazon QuickSight (Business Intelligence Tool)

**Workflow:**

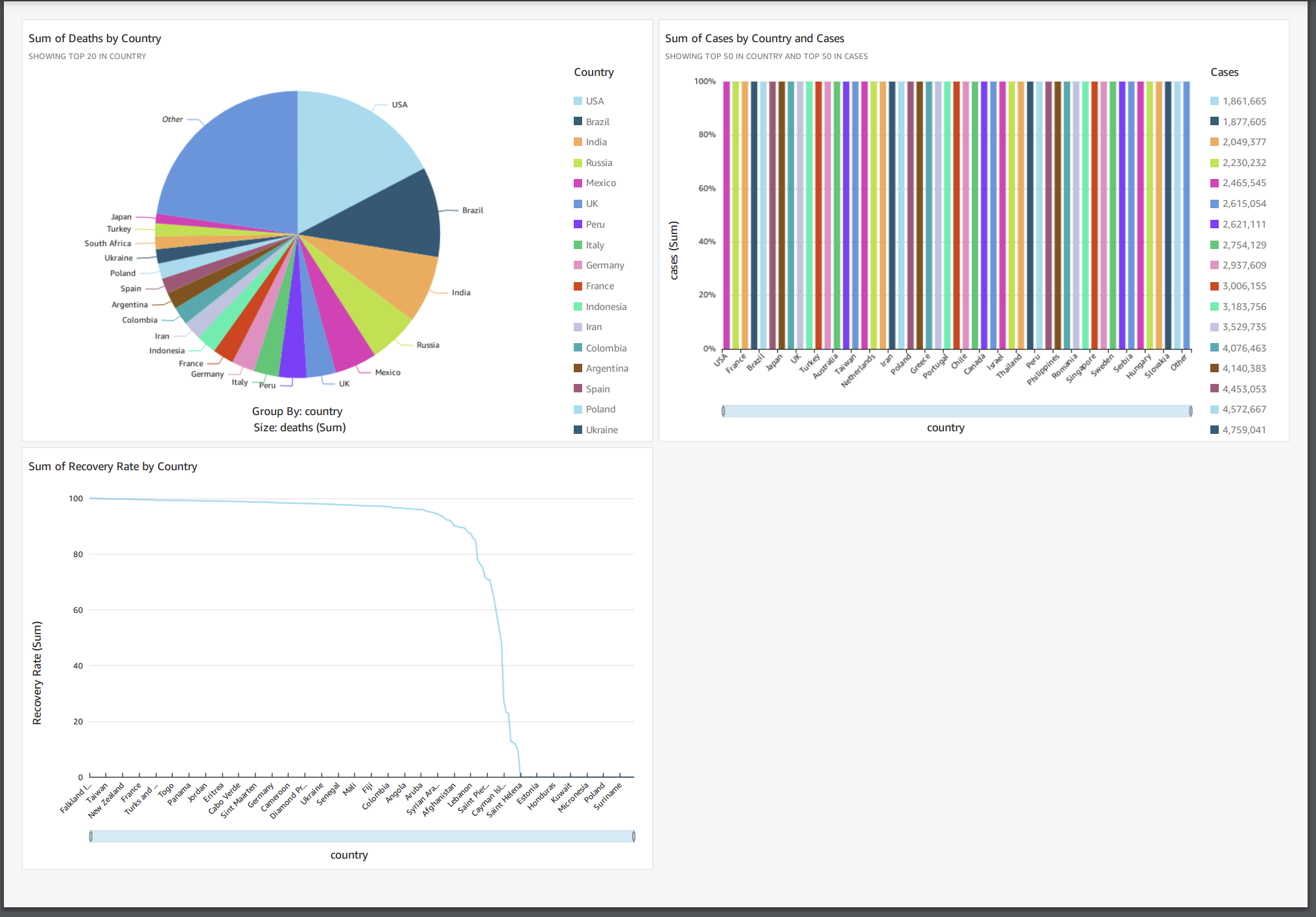
* Connected AWS QuickSight to Athena
* Selected Athena table (covid\_data) as the data source.
* Enabled Direct Query Mode for live updates.
* Created Visualizations & Dashboards

* Bar Chart: COVID-19 cases by country
* Pie Chart: Death rate per country
* Line Chart: Recovery rate trends

**Exported Reports**

* Successfully exported the dashboard to PDF for reporting.

**Screenshots:**



**Appendix:**  
  
**Lambda Function for Data Extraction:**  
  
import json

import boto3

import requests

import os

from datetime import datetime

# Initialize S3 client

s3=boto3.client('s3')

# Set variables

API\_URL = "<https://disease.sh/v3/covid-19/countries>"

BUCKET\_NAME = "covid-data-etl-project"

def lambda\_handler(event, context):

try:

# Fetch COVID-19 data from the API

response = requests.get(API\_URL)

data = response.json()

filename = f"raw-data/covid\_data\_{datetime.now().strftime('%Y-%m-%d')}.json"

s3.put\_object(

Bucket = BUCKET\_NAME,

Key = filename,

Body = json.dumps(data)

)

return {

'statusCode': 200,

'body': json.dumps('Data fetched and stored successfully!')

}

# print("Data fetched and stored successfully!")

except Exception as e:

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

return {

'statusCode': 500,

'body': json.dumps('Error fetching and storing data!')

}

**Lambda Function for Transformation JSON TO CSV:**

import json

import csv

import boto3

from io import StringIO

from datetime import datetime

# Initialize S3 client

s3 = boto3.client('s3')

# S3 bucket details

BUCKET\_NAME = "covid-data-etl-project" # Replace with your actual bucket name

RAW\_FOLDER = "raw-data/"

PROCESSED\_FOLDER = "processed-data/"

def lambda\_handler(event, context):

try:

print("Fetching latest JSON file from S3...")

# Get the latest file in raw-data folder

response = s3.list\_objects\_v2(Bucket=BUCKET\_NAME, Prefix=RAW\_FOLDER)

if 'Contents' not in response:

raise Exception("No files found in raw-data folder!")

# Sort files by last modified time (latest first)

latest\_file = max(response['Contents'], key=lambda x: x['LastModified'])['Key']

print(f"Latest file found: {latest\_file}")

# Read the JSON file from S3

response = s3.get\_object(Bucket=BUCKET\_NAME, Key=latest\_file)

json\_data = json.loads(response['Body'].read().decode('utf-8'))

# Convert JSON to CSV

csv\_buffer = StringIO()

csv\_writer = csv.writer(csv\_buffer)

# Write CSV header

csv\_writer.writerow(["Country", "Cases", "Deaths", "Recovered"])

for entry in json\_data:

csv\_writer.writerow([entry["country"], entry["cases"], entry["deaths"], entry["recovered"]])

# Generate CSV filename

csv\_filename = f"{PROCESSED\_FOLDER}covid\_data\_{datetime.now().strftime('%Y-%m-%d')}.csv"

# Upload CSV to S3

s3.put\_object(Bucket=BUCKET\_NAME, Key=csv\_filename, Body=csv\_buffer.getvalue())

print(f"CSV file saved successfully: {csv\_filename}")

return {

"statusCode": 200,

"body": f"CSV file successfully saved to S3 as {csv\_filename}"

}

except Exception as e:

print(f"Error during JSON to CSV conversion: {e}")

return {

"statusCode": 500,

"body": f"Error: {str(e)}"

}