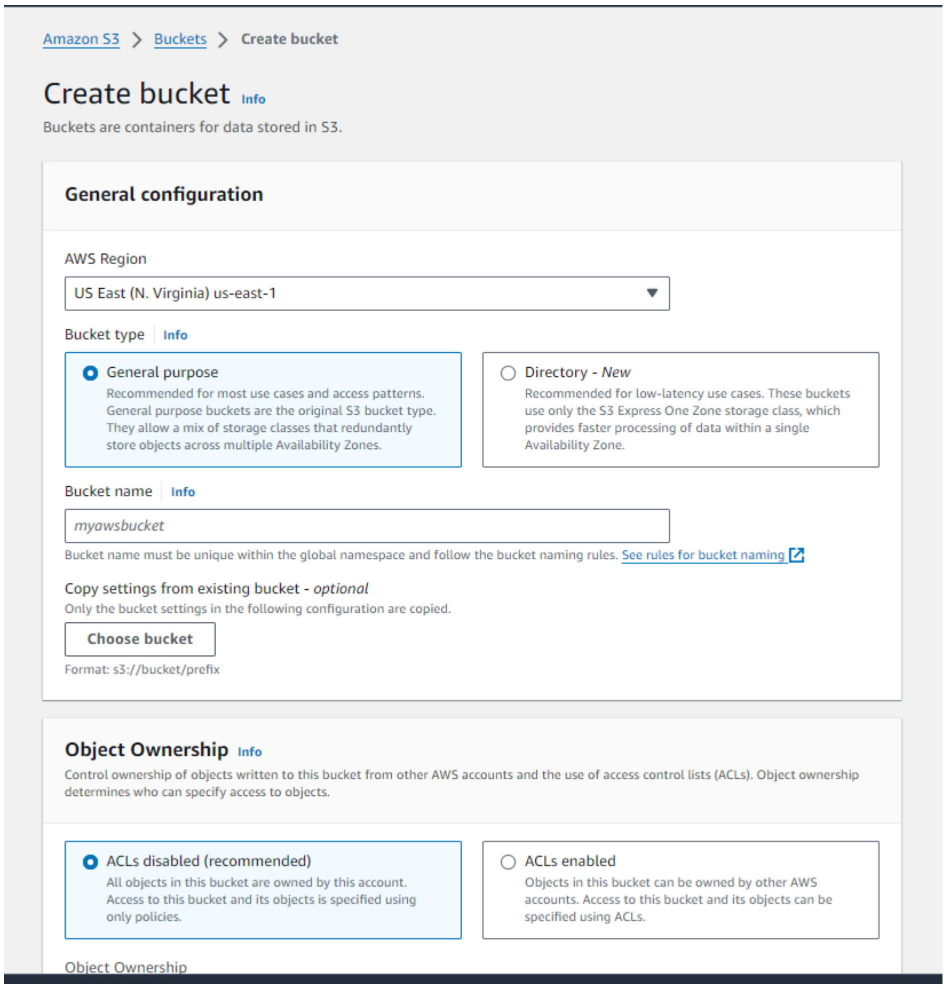
Work with RXNORM file,

1. Scrap the latest RXNORM file from NLM webpage
2. Download the latest RXNORM file with api_key
3. Create a log file for the downloaded file
4. Add header into each rff from RXNORM.xlsx
5. Add CODE_SET & VERSION_MONTH column with default values RxNorm and version month from downloaded filename
6. Convert dates into YYYY-MM-DD
7. Save files as txt delimited by comma(,)c
8. Validate row_count between original and converted files

Steps Involved:

1) A bucket is created to store the zip file containing the (.RFF) files, Excel file and the files created after transformations and headers are applied.



Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region: US East (N. Virginia) us-east-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 - *New* 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

myawsbucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

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

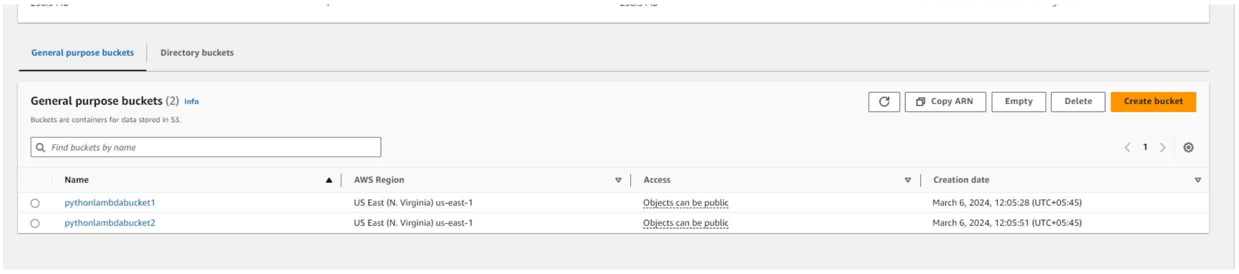
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

[✓ 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

Buckets created successfully



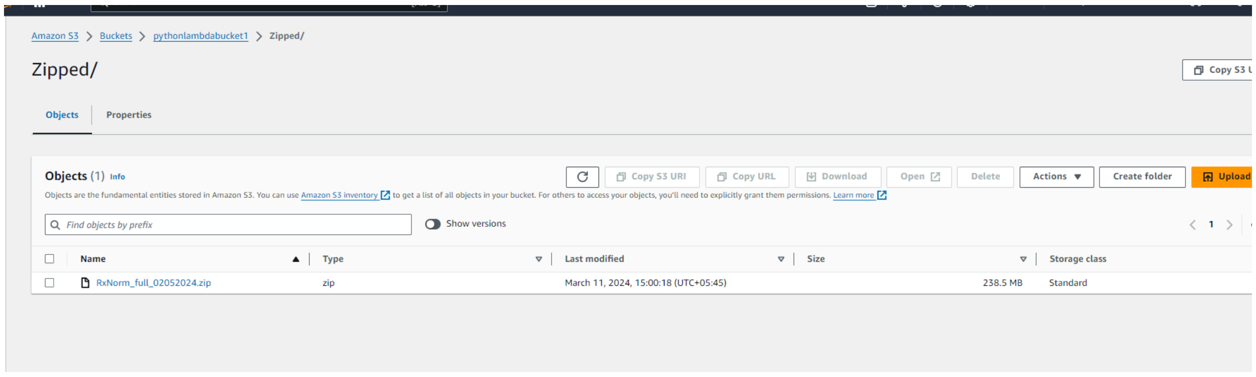
General purpose buckets | Directory buckets

General purpose buckets (2) Info

Buckets are containers for data stored in S3.

Name	AWS Region	Access	Creation date
pythonlambdabucket1	US East (N. Virginia) us-east-1	Objects can be public	March 6, 2024, 12:05:28 (UTC+05:45)
pythonlambdabucket2	US East (N. Virginia) us-east-1	Objects can be public	March 6, 2024, 12:05:51 (UTC+05:45)

2) **Folders for zipped and excel sheets are created and respective file is uploaded in it.**



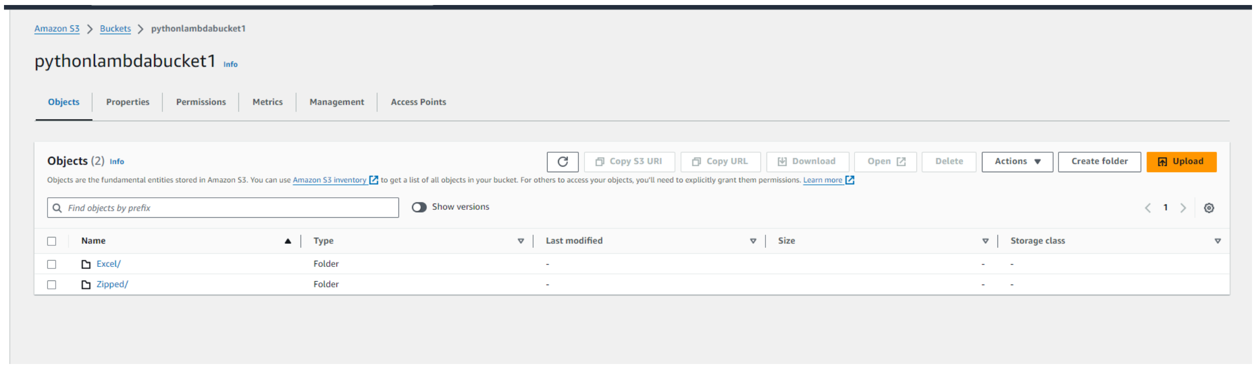
Amazon S3 > Buckets > pythonlambdabucket1 > Zipped/

Zipped/

Objects (1) Info

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
RxNorm_full_02052024.zip	zip	March 11, 2024, 15:00:18 (UTC+05:45)	238.5 MB	Standard



Amazon S3 > Buckets > pythonlambdabucket1

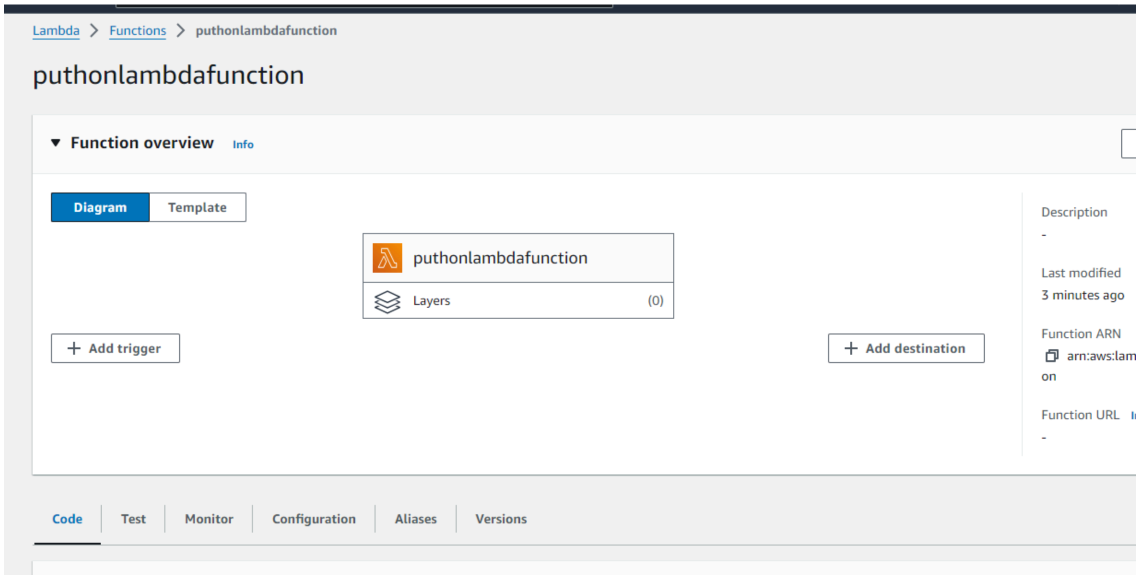
pythonlambdabucket1 Info

Objects (2) Info

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
Excel/	Folder	-	-	-
Zipped/	Folder	-	-	-

3) Creation of AWS Lambda Function


Lambda > Functions > puthonlambdafunction

puthonlambdafunction

Function overview Info

Description
-

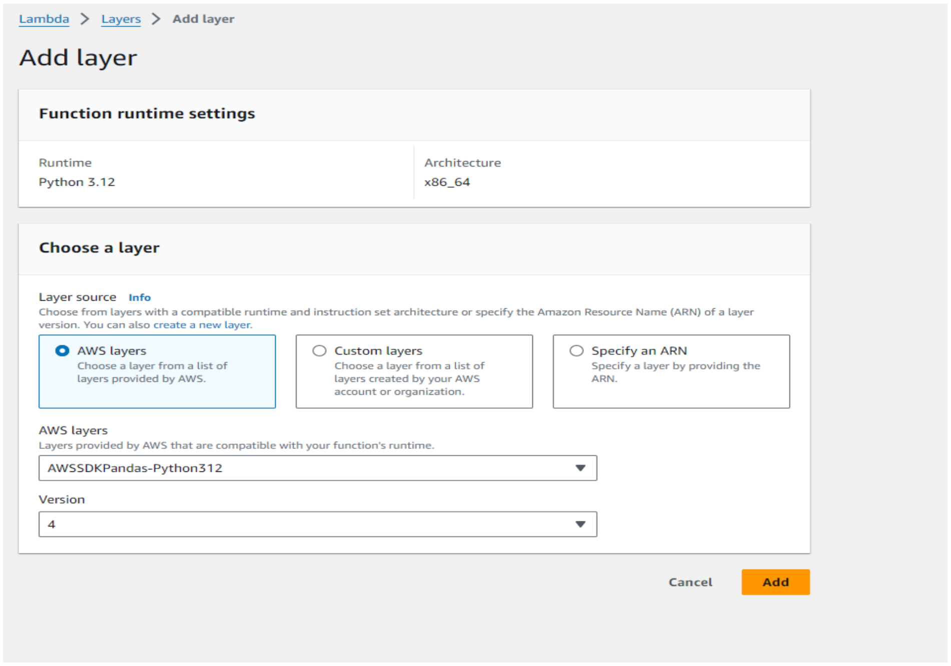
Last modified
3 minutes ago

Function ARN
arn:aws:lam
on

Function URL
-

4) Adding Lambda Layers

Pandas Layer is added to the Lambda for processing the files per task requirements


Lambda > Layers > Add layer

Add layer

Function runtime settings

Runtime
Python 3.12

Architecture
x86_64

Choose a layer

Layer source Info
Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also create a new layer.

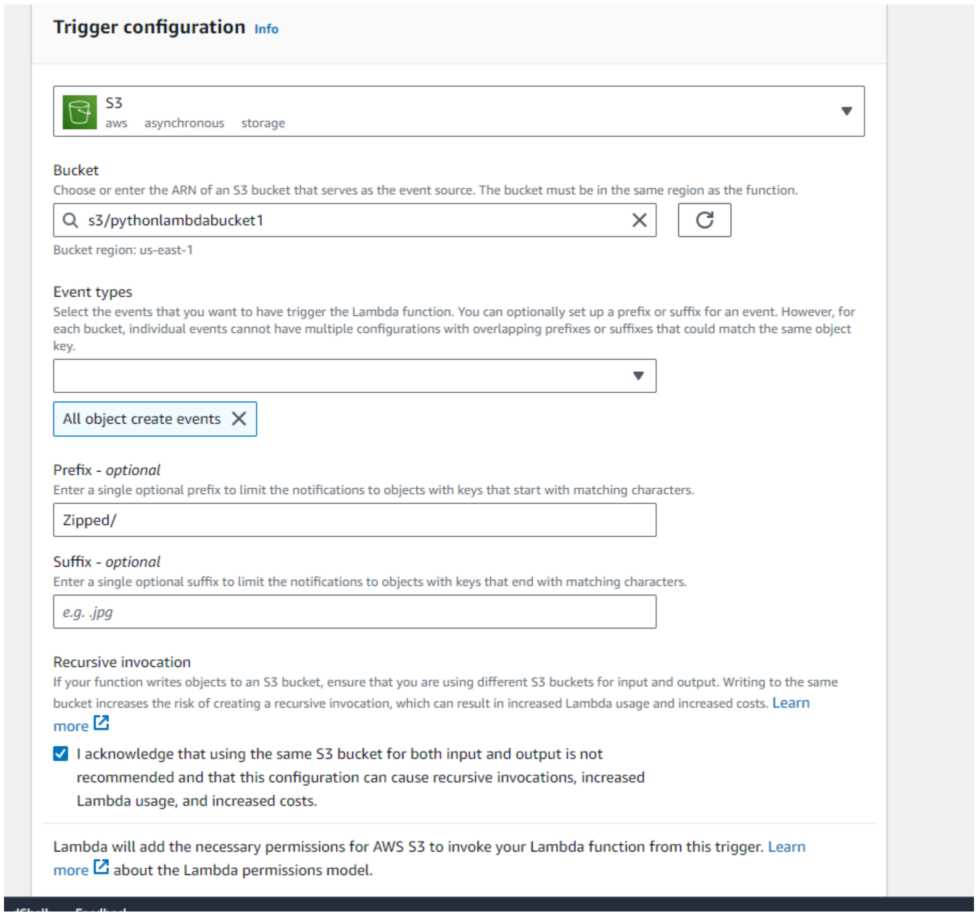
[✓ AWS layers] Choose a layer from a list of layers provided by AWS.
Custom layers Choose a layer from a list of layers created by your AWS account or organization.
Specify an ARN Specify a layer by providing the ARN.

AWS layers
Layers provided by AWS that are compatible with your function's runtime.
AWSSDKPandas-Python312

Version
4

5) Adding Trigger

Choose Event types – all
S3 bucket made previously is added as a trigger to process files.



Trigger configuration Info

S3
aws asynchronous storage

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/pythonlambdabucket1
Bucket region: us-east-1

Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events

Prefix - *optional*
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.
Zipped/

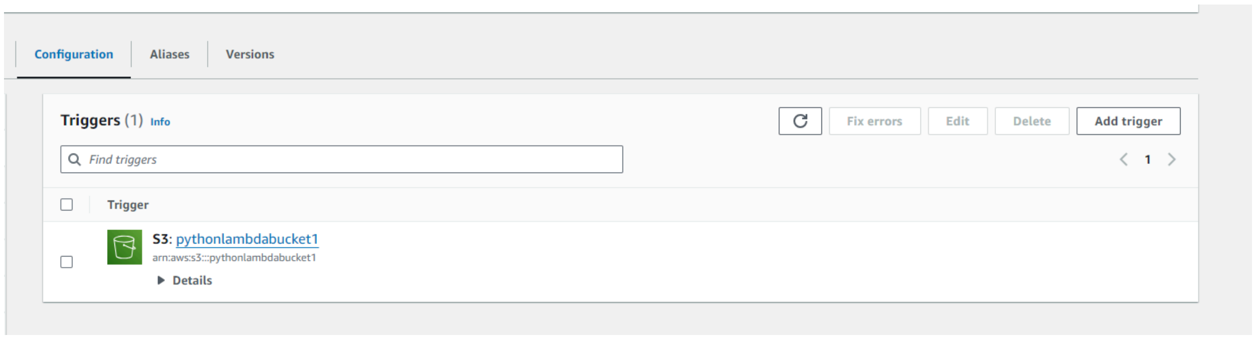
Suffix - *optional*
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.
e.g. .jpg

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. Learn more

[✓] I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Triggered added successfully.

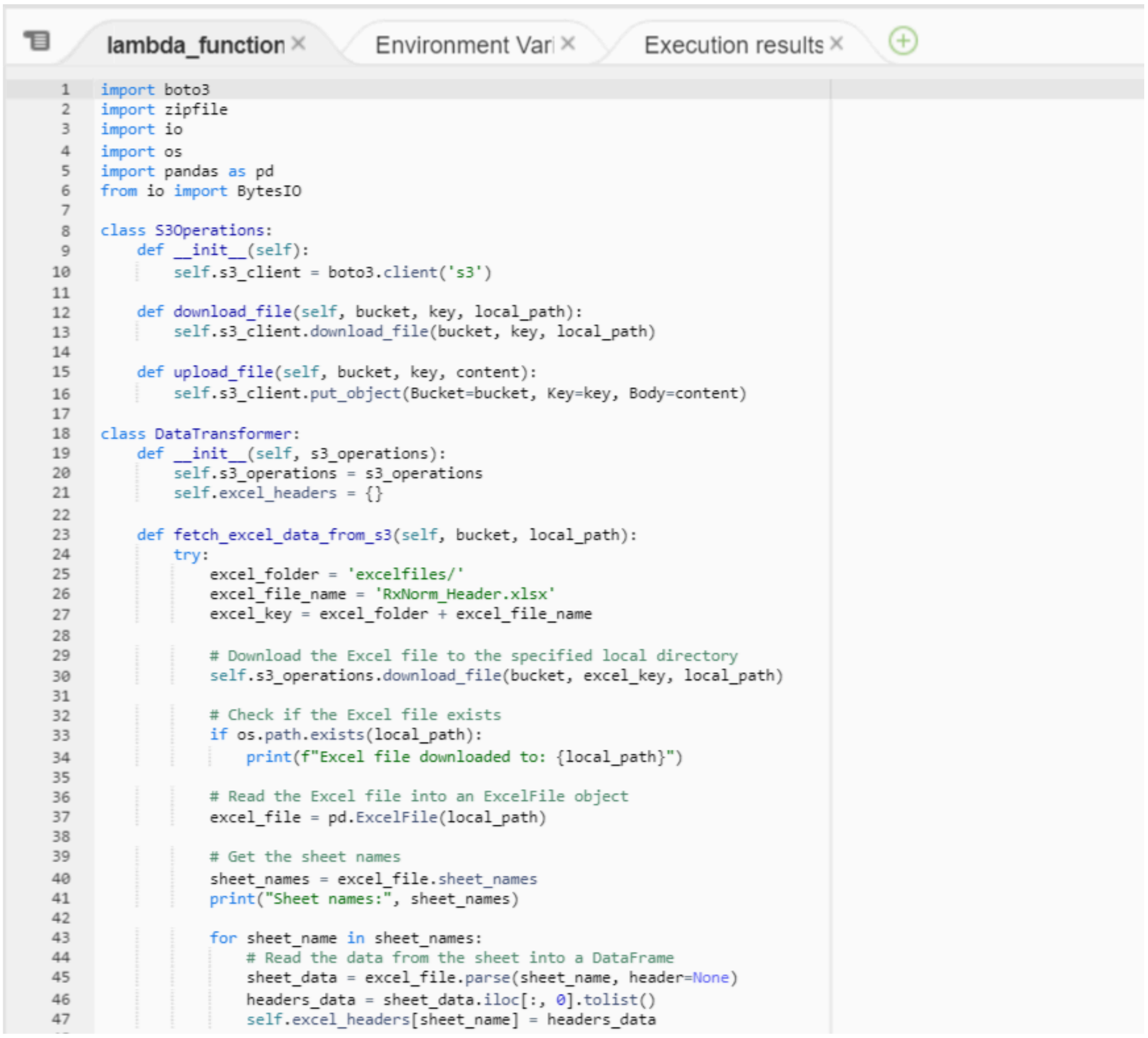


Triggers (1) Info

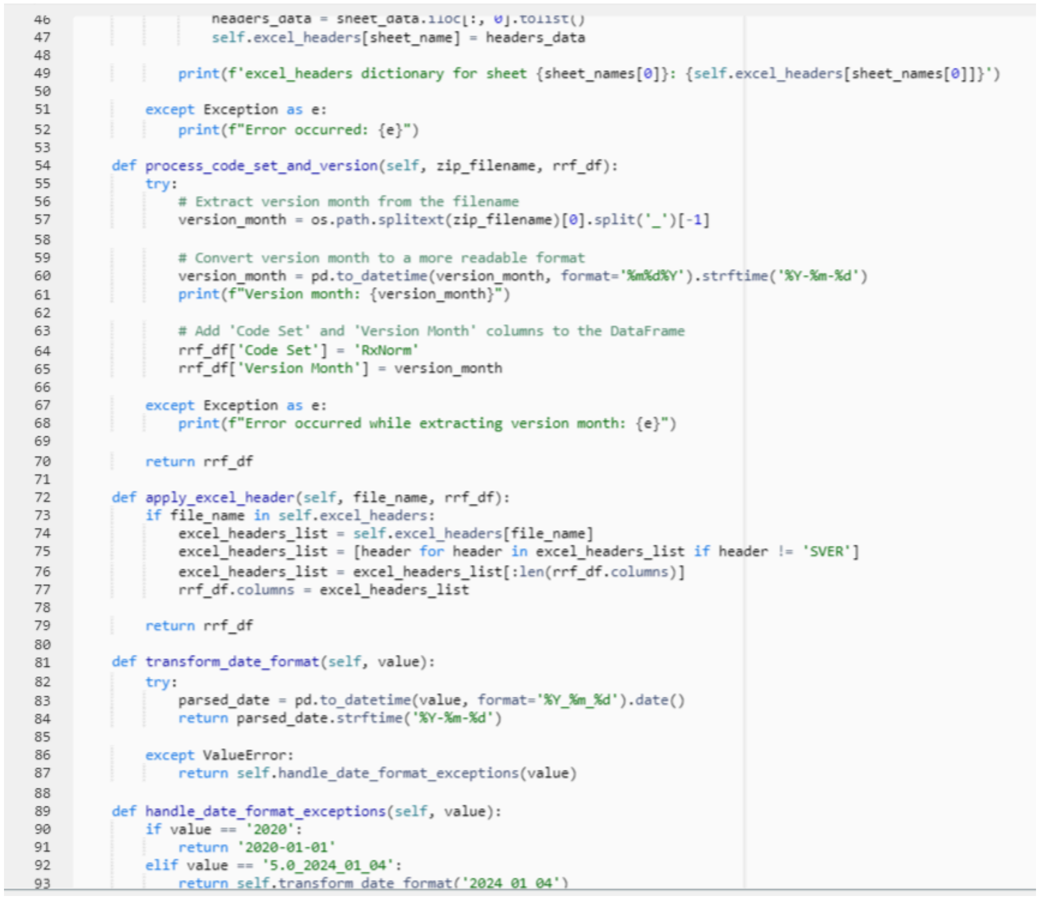
Find triggers

Trigger
S3: pythonlambdabucket1
arn:aws:s3:::pythonlambdabucket1
Details

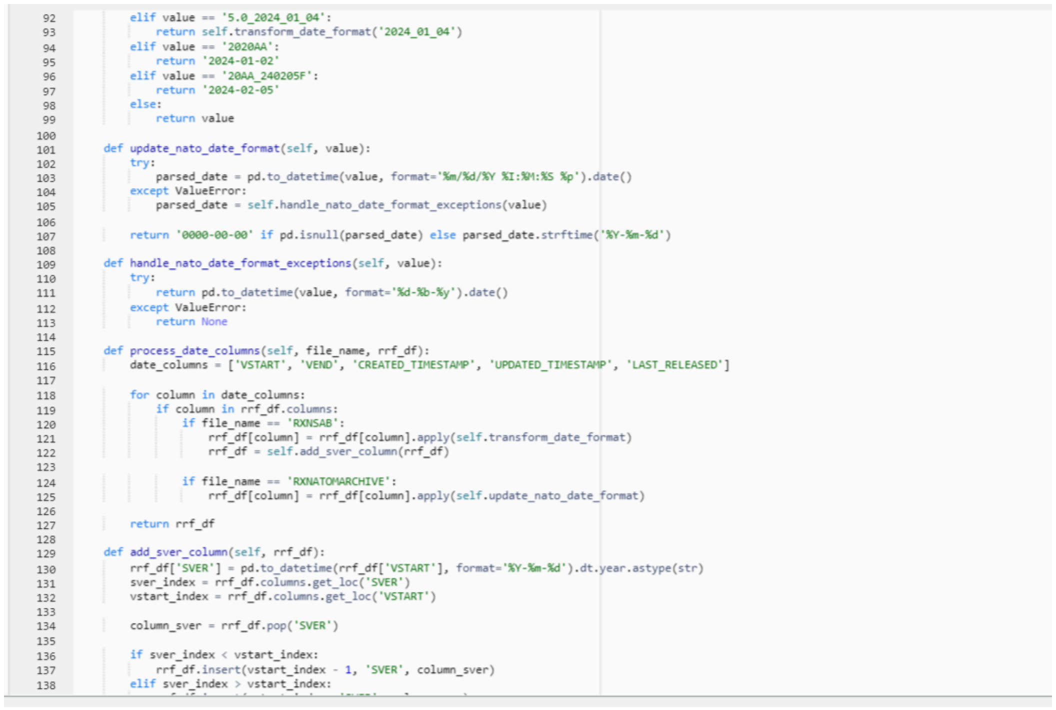
6) The complete code is written:



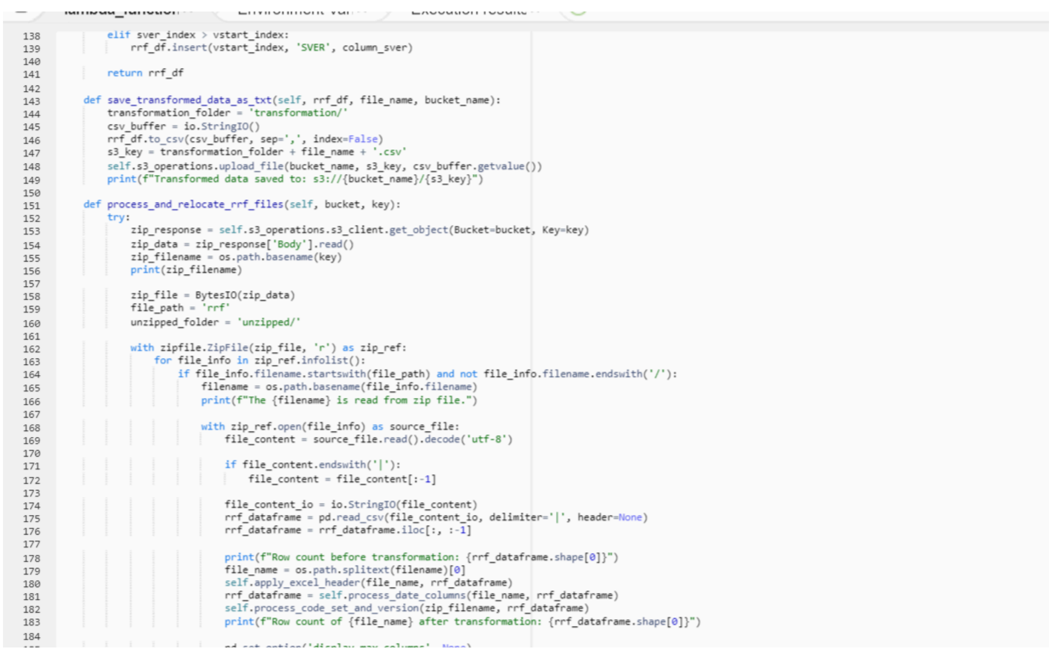
```python
import boto3
import zipfile
import io
import os
import pandas as pd
from io import BytesIO

class S3Operations:
    def __init__(self):
        self.s3_client = boto3.client('s3')

    def download_file(self, bucket, key, local_path):
        self.s3_client.download_file(bucket, key, local_path)

    def upload_file(self, bucket, key, content):
        self.s3_client.put_object(Bucket=bucket, Key=key, Body=content)

class DataTransformer:
    def __init__(self, s3_operations):
        self.s3_operations = s3_operations
        self.excel_headers = {}

    def fetch_excel_data_from_s3(self, bucket, local_path):
        try:
            excel_folder = 'excelfiles/'
            excel_file_name = 'RxNorm_Header.xlsx'
            excel_key = excel_folder + excel_file_name

            # Download the Excel file to the specified local directory
            self.s3_operations.download_file(bucket, excel_key, local_path)

            # Check if the Excel file exists
            if os.path.exists(local_path):
                print(f"Excel file downloaded to: {local_path}")

            # Read the Excel file into an ExcelFile object
            excel_file = pd.ExcelFile(local_path)

            # Get the sheet names
            sheet_names = excel_file.sheet_names
            print("Sheet names:", sheet_names)

            for sheet_name in sheet_names:
                # Read the data from the sheet into a DataFrame
                sheet_data = excel_file.parse(sheet_name, header=None)
                headers_data = sheet_data.iloc[:, 0].tolist()
                self.excel_headers[sheet_name] = headers_data
```



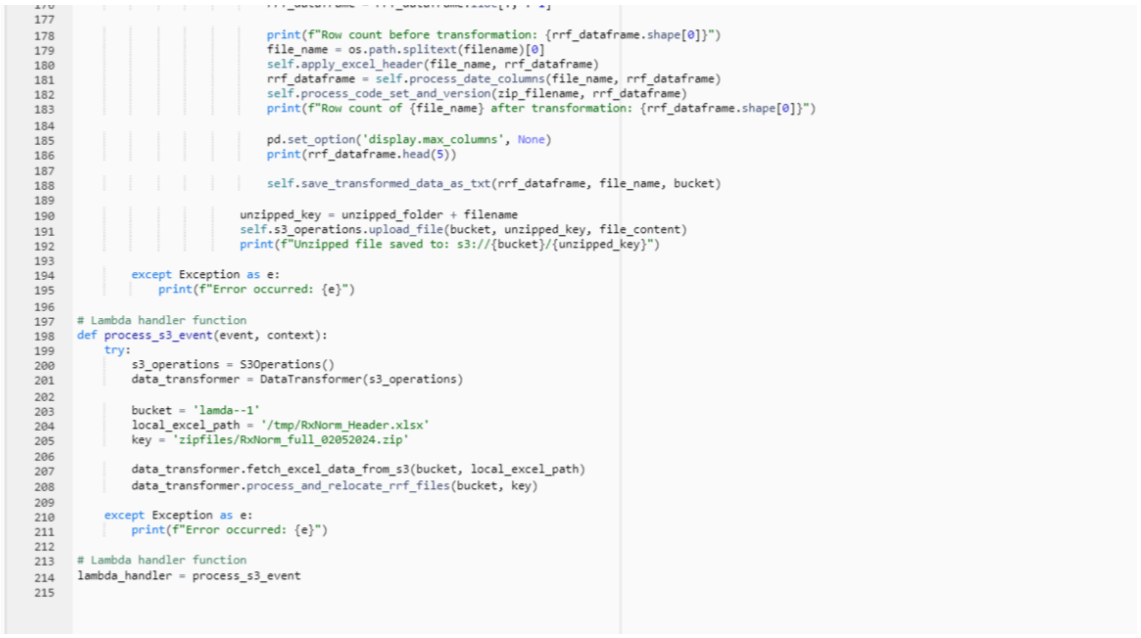
```python
                headers_data = sheet_data.iloc[:, 0].tolist()
                self.excel_headers[sheet_name] = headers_data

            print(f'excel_headers dictionary for sheet {sheet_names[0]}: {self.excel_headers[sheet_names[0]]}')

        except Exception as e:
            print(f"Error occurred: {e}")

    def process_code_set_and_version(self, zip_filename, rrf_df):
        try:
            # Extract version month from the filename
            version_month = os.path.splitext(zip_filename)[0].split('_')[-1]

            # Convert version month to a more readable format
            version_month = pd.to_datetime(version_month, format='%m%d%Y').strftime('%Y-%m-%d')
            print(f"Version month: {version_month}")

            # Add 'Code Set' and 'Version Month' columns to the DataFrame
            rrf_df['Code Set'] = 'RxNorm'
            rrf_df['Version Month'] = version_month

        except Exception as e:
            print(f"Error occurred while extracting version month: {e}")

        return rrf_df

    def apply_excel_header(self, file_name, rrf_df):
        if file_name in self.excel_headers:
            excel_headers_list = self.excel_headers[file_name]
            excel_headers_list = [header for header in excel_headers_list if header != 'SVER']
            excel_headers_list = excel_headers_list[:len(rrf_df.columns)]
            rrf_df.columns = excel_headers_list

        return rrf_df

    def transform_date_format(self, value):
        try:
            parsed_date = pd.to_datetime(value, format='%Y_%m_%d').date()
            return parsed_date.strftime('%Y-%m-%d')

        except ValueError:
            return self.handle_date_format_exceptions(value)

    def handle_date_format_exceptions(self, value):
        if value == '2020':
            return '2020-01-01'
        elif value == '5.0_2024_01_04':
            return self.transform_date_format('2024_01_04')
```



```python
        elif value == '5.0_2024_01_04':
            return self.transform_date_format('2024_01_04')
        elif value == '2020AA':
            return '2024-01-02'
        elif value == '20AA_240205F':
            return '2024-02-05'
        else:
            return value

    def update_nato_date_format(self, value):
        try:
            parsed_date = pd.to_datetime(value, format='%m/%d/%Y %I:%M:%S %p').date()
        except ValueError:
            parsed_date = self.handle_nato_date_format_exceptions(value)

        return '0000-00-00' if pd.isnull(parsed_date) else parsed_date.strftime('%Y-%m-%d')

    def handle_nato_date_format_exceptions(self, value):
        try:
            return pd.to_datetime(value, format='%d-%b-%y').date()
        except ValueError:
            return None

    def process_date_columns(self, file_name, rrf_df):
        date_columns = ['VSTART', 'VEND', 'CREATED_TIMESTAMP', 'UPDATED_TIMESTAMP', 'LAST_RELEASED']

        for column in date_columns:
            if column in rrf_df.columns:
                if file_name == 'RXNSAB':
                    rrf_df[column] = rrf_df[column].apply(self.transform_date_format)
                    rrf_df = self.add_sver_column(rrf_df)

                if file_name == 'RXNATOMARCHIVE':
                    rrf_df[column] = rrf_df[column].apply(self.update_nato_date_format)

        return rrf_df

    def add_sver_column(self, rrf_df):
        rrf_df['SVER'] = pd.to_datetime(rrf_df['VSTART'], format='%Y-%m-%d').dt.year.astype(str)
        sver_index = rrf_df.columns.get_loc('SVER')
        vstart_index = rrf_df.columns.get_loc('VSTART')

        column_sver = rrf_df.pop('SVER')

        if sver_index < vstart_index:
            rrf_df.insert(vstart_index - 1, 'SVER', column_sver)
        elif sver_index > vstart_index:
```



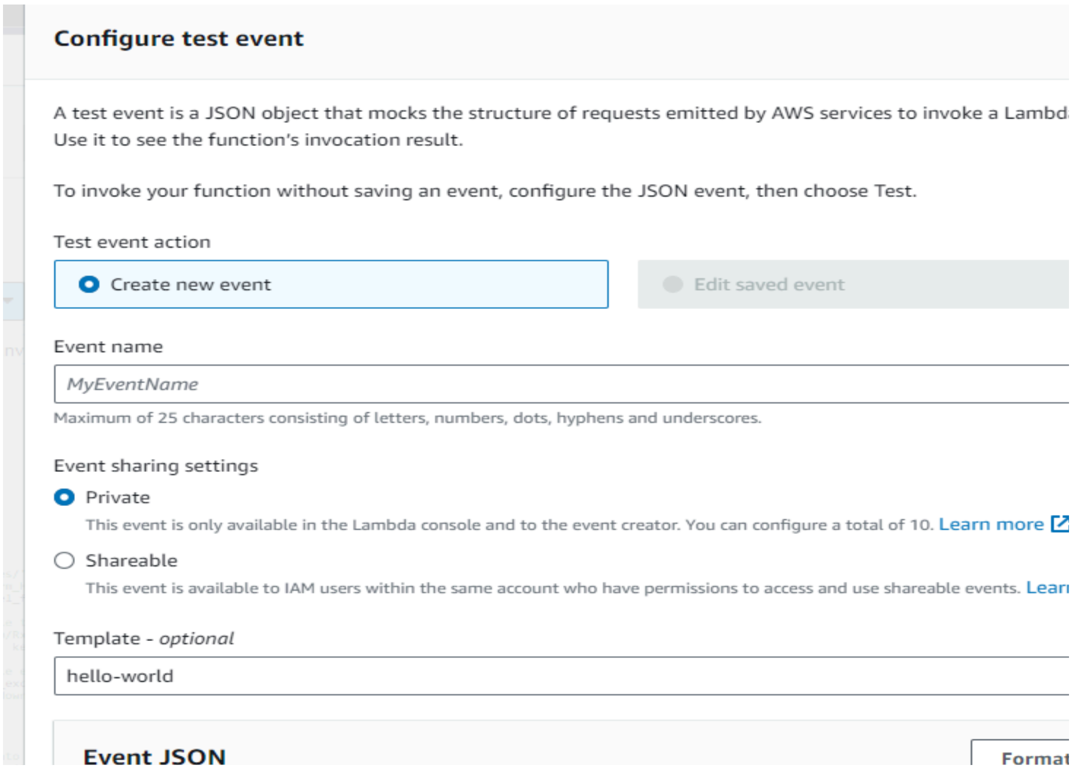
```python
        elif sver_index > vstart_index:
            rrf_df.insert(vstart_index, 'SVER', column_sver)

        return rrf_df

    def save_transformed_data_as_txt(self, rrf_df, file_name, bucket_name):
        transformation_folder = 'transformation/'
        csv_buffer = io.StringIO()
        rrf_df.to_csv(csv_buffer, sep=',', index=False)
        s3_key = transformation_folder + file_name + '.csv'
        self.s3_operations.upload_file(bucket_name, s3_key, csv_buffer.getvalue())
        print(f"Transformed data saved to: s3://{bucket_name}/{s3_key}")

    def process_and_relocate_rrf_files(self, bucket, key):
        try:
            zip_response = self.s3_operations.s3_client.get_object(Bucket=bucket, Key=key)
            zip_data = zip_response['Body'].read()
            zip_filename = os.path.basename(key)
            print(zip_filename)

            zip_file = BytesIO(zip_data)
            file_path = 'rrf'
            unzipped_folder = 'unzipped/'

            with zipfile.ZipFile(zip_file, 'r') as zip_ref:
                for file_info in zip_ref.infolist():
                    if file_info.filename.startswith(file_path) and not file_info.filename.endswith('/'):
                        filename = os.path.basename(file_info.filename)
                        print(f"The {filename} is read from zip file.")

                        with zip_ref.open(file_info) as source_file:
                            file_content = source_file.read().decode('utf-8')

                            if file_content.endswith('|'):
                                file_content = file_content[:-1]

                            file_content_io = io.StringIO(file_content)
                            rrf_dataframe = pd.read_csv(file_content_io, delimiter='|', header=None)
                            rrf_dataframe = rrf_dataframe.iloc[:, :-1]

                            print(f"Row count before transformation: {rrf_dataframe.shape[0]}")
                            file_name = os.path.splitext(filename)[0]
                            self.apply_excel_header(file_name, rrf_dataframe)
                            rrf_dataframe = self.process_date_columns(file_name, rrf_dataframe)
                            self.process_code_set_and_version(zip_filename, rrf_dataframe)
                            print(f"Row count of {file_name} after transformation: {rrf_dataframe.shape[0]}")
```



```python
                            print(f"Row count before transformation: {rrf_dataframe.shape[0]}")
                            file_name = os.path.splitext(filename)[0]
                            self.apply_excel_header(file_name, rrf_dataframe)
                            rrf_dataframe = self.process_date_columns(file_name, rrf_dataframe)
                            self.process_code_set_and_version(zip_filename, rrf_dataframe)
                            print(f"Row count of {file_name} after transformation: {rrf_dataframe.shape[0]}")
```

```python
                            pd.set_option('display.max_columns', None)
                            print(rrf_dataframe.head(5))

                            self.save_transformed_data_as_txt(rrf_dataframe, file_name, bucket)

                        unzipped_key = unzipped_folder + filename
                        self.s3_operations.upload_file(bucket, unzipped_key, file_content)
                        print(f"Unzipped file saved to: s3://{bucket}/{unzipped_key}")

        except Exception as e:
            print(f"Error occurred: {e}")

# Lambda handler function
def process_s3_event(event, context):
    try:
        s3_operations = S3Operations()
        data_transformer = DataTransformer(s3_operations)

        bucket = 'lamda--1'
        local_excel_path = '/tmp/RxNorm_Header.xlsx'
        key = 'zipfiles/RxNorm_full_02052024.zip'

        data_transformer.fetch_excel_data_from_s3(bucket, local_excel_path)
        data_transformer.process_and_relocate_rrf_files(bucket, key)

    except Exception as e:
        print(f"Error occurred: {e}")

# Lambda handler function
lambda_handler = process_s3_event
```

7) **Configuring Test Event**
A simple test event is made to check for correct functioning of the code.



Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambd
Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action
● Create new event ○ Edit saved event

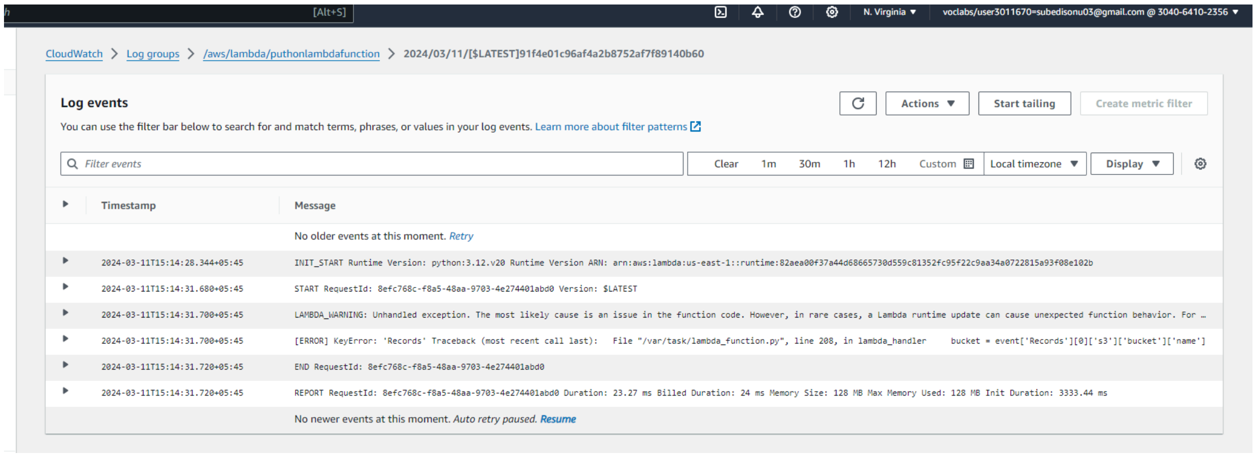
Event name
MyEventName
Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings
● Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more
○ Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. Lear

Template - *optional*
hello-world

Event JSON

8) **Checking log events.. The issues persist and are solved accordingly.**



CloudWatch > Log groups > /aws/lambda/puthonlambdafunction > 2024/03/11/[$LATEST]91f4e01c96af4a2b8752af7f89140b60

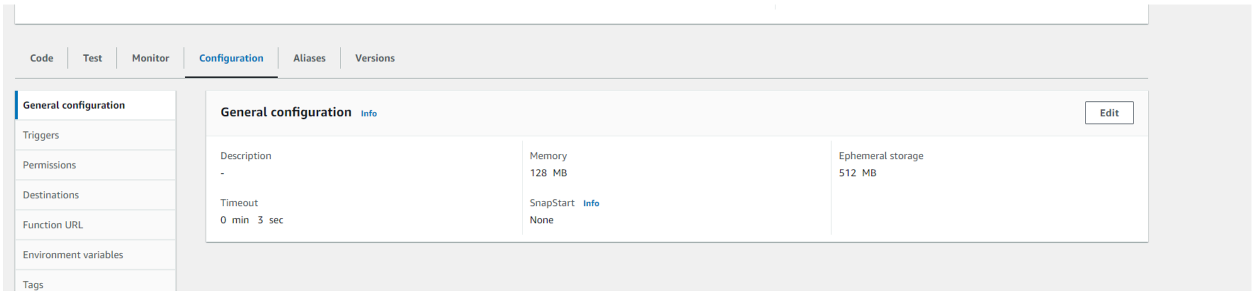
Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Filter events

```
Timestamp                     Message
                              No older events at this moment. Retry
2024-03-11T15:14:28.344+05:45 INIT_START Runtime Version: python:3.12.v20 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:82aea00f37a44d68665730d559c81352fc95f22c9aa34a0722815a93f08e102b
2024-03-11T15:14:31.680+05:45 START RequestId: 8efc768c-f8a5-48aa-9703-4e274401abd0 Version: $LATEST
2024-03-11T15:14:31.700+05:45 LAMBDA_WARNING: Unhandled exception. The most likely cause is an issue in the function code. However, in rare cases, a Lambda runtime update can cause unexpected function behavior. For …
2024-03-11T15:14:31.700+05:45 [ERROR] KeyError: 'Records' Traceback (most recent call last):   File "/var/task/lambda_function.py", line 208, in lambda_handler     bucket = event['Records'][0]['s3']['bucket']['name']
2024-03-11T15:14:31.720+05:45 END RequestId: 8efc768c-f8a5-48aa-9703-4e274401abd0
2024-03-11T15:14:31.720+05:45 REPORT RequestId: 8efc768c-f8a5-48aa-9703-4e274401abd0 Duration: 23.27 ms Billed Duration: 24 ms Memory Size: 128 MB Max Memory Used: 128 MB Init Duration: 3333.44 ms
                              No newer events at this moment. Auto retry paused. Resume
```

9) Go to the General Configuration of lambda function and change its basic settings.


Code | Test | Monitor | Configuration | Aliases | Versions

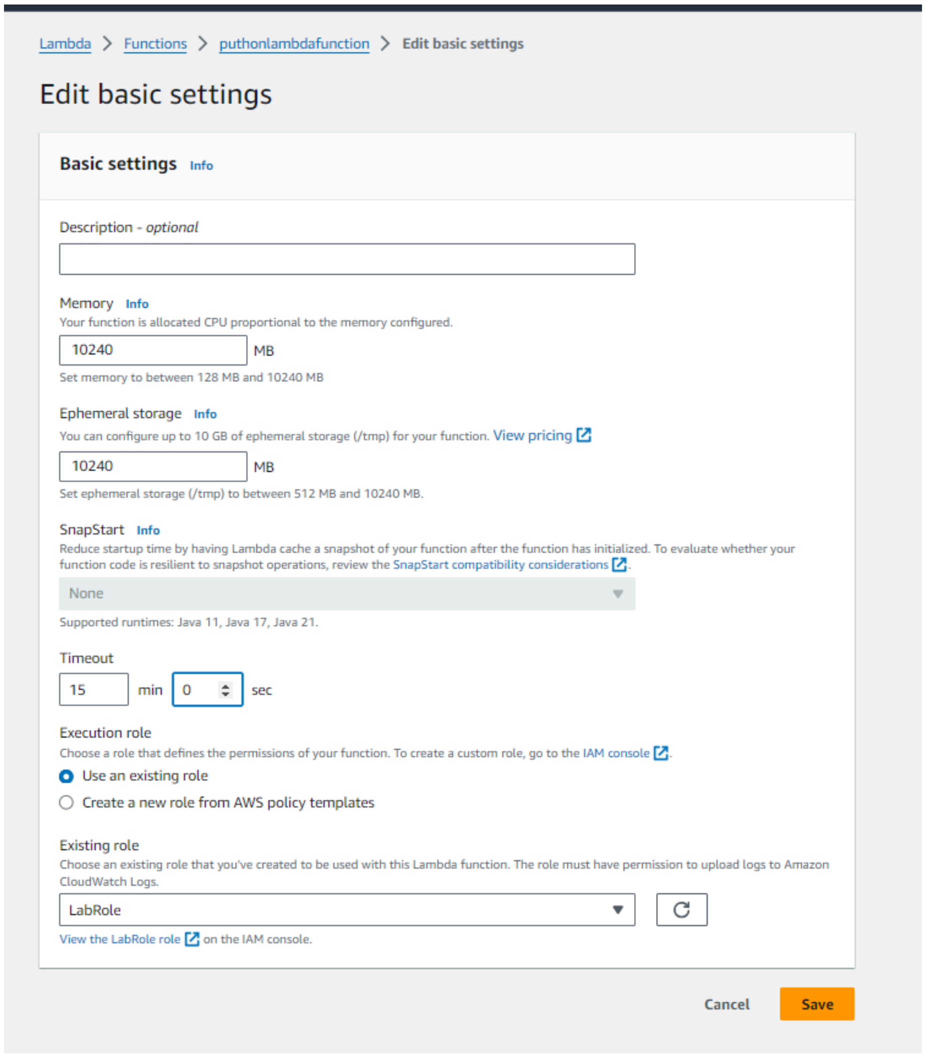
General configuration
Triggers
Permissions
Destinations
Function URL
Environment variables
Tags

General configuration Info

Edit

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart Info	
0 min 3 sec	None	

Changing Basic Settings. Edit Memory configuration andTimeout as required.


Lambda > Functions > puthonlambdafunction > Edit basic settings

Edit basic settings

Basic settings Info

Description - *optional*

Memory Info
Your function is allocated CPU proportional to the memory configured.
10240 MB
Set memory to between 128 MB and 10240 MB

Ephemeral storage Info
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing
10240 MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.
None
Supported runtimes: Java 11, Java 17, Java 21.

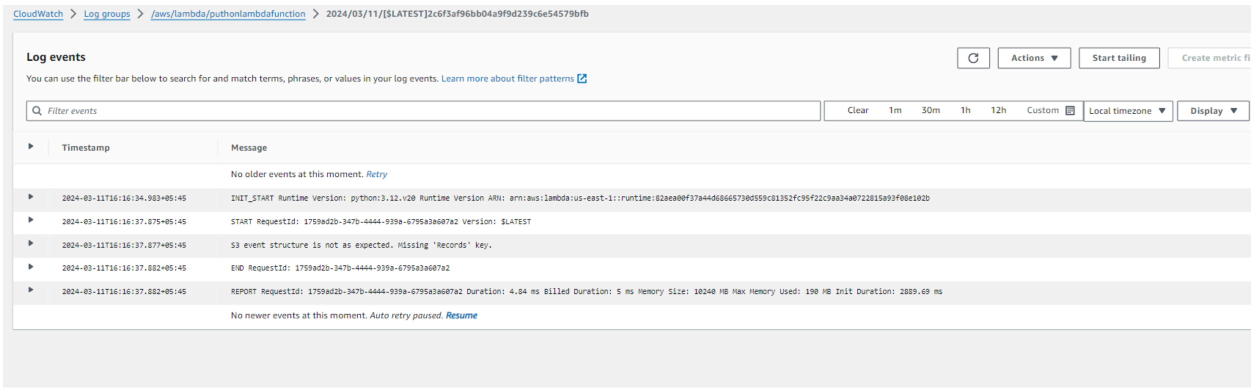
Timeout
15 min 0 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
● Use an existing role
○ Create a new role from AWS policy templates

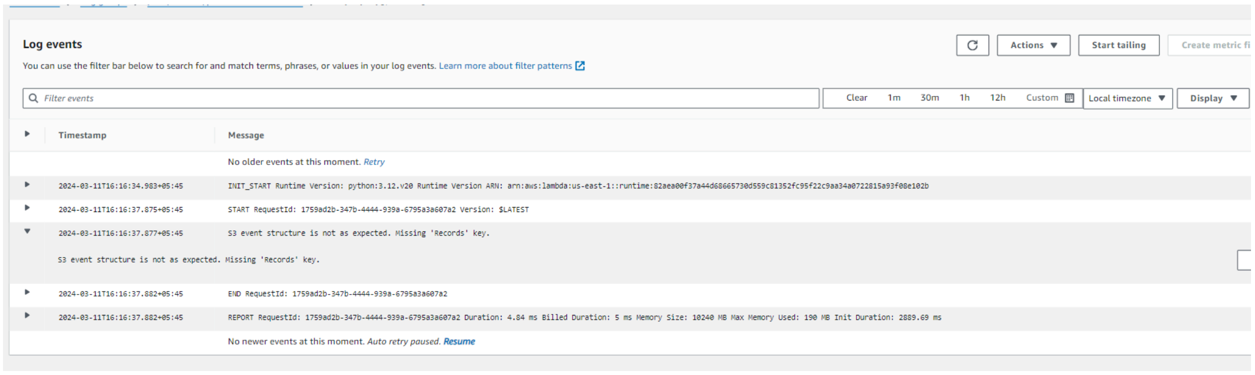
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LabRole
View the LabRole role on the IAM console.

Cancel Save

Some of the errors seen in logs of CloudWatch:

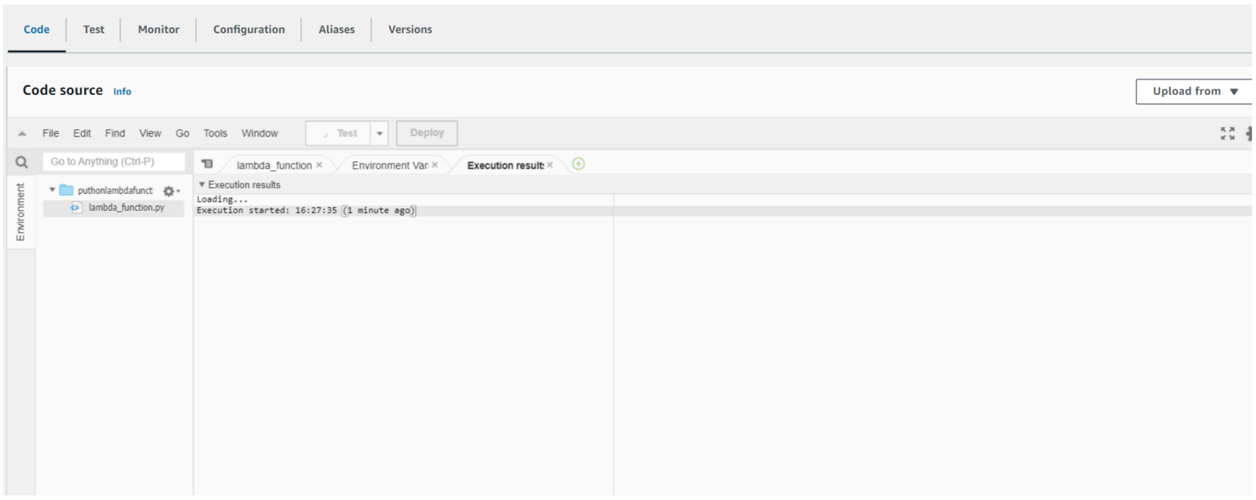

CloudWatch > Log groups > /aws/lambda/puthonlambdafunction > 2024/03/11/[$LATEST]2c6f3af96bb04a9f9d239c6e54579bfb
Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Actions
Start tailing
Create metric fi
Filter events
Clear
1m
30m
1h
12h
Custom
Local timezone
Display
Timestamp
Message

```
No older events at this moment. Retry
2024-03-11T16:16:34.983+05:45   INIT_START Runtime Version: python:3.12.v20 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:82aea00f37a44d68665730d559c81352fc95f22c9aa34a0722815a93f08e102b
2024-03-11T16:16:37.875+05:45   START RequestId: 1759ad2b-347b-4444-939a-6795a3a607a2 Version: $LATEST
2024-03-11T16:16:37.877+05:45   S3 event structure is not as expected. Missing 'Records' key.
2024-03-11T16:16:37.882+05:45   END RequestId: 1759ad2b-347b-4444-939a-6795a3a607a2
2024-03-11T16:16:37.882+05:45   REPORT RequestId: 1759ad2b-347b-4444-939a-6795a3a607a2 Duration: 4.84 ms Billed Duration: 5 ms Memory Size: 10240 MB Max Memory Used: 190 MB Init Duration: 2889.69 ms
No newer events at this moment. Auto retry paused. Resume
```


Log events
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Actions
Start tailing
Create metric fi
Filter events
Clear
1m
30m
1h
12h
Custom
Local timezone
Display
Timestamp
Message

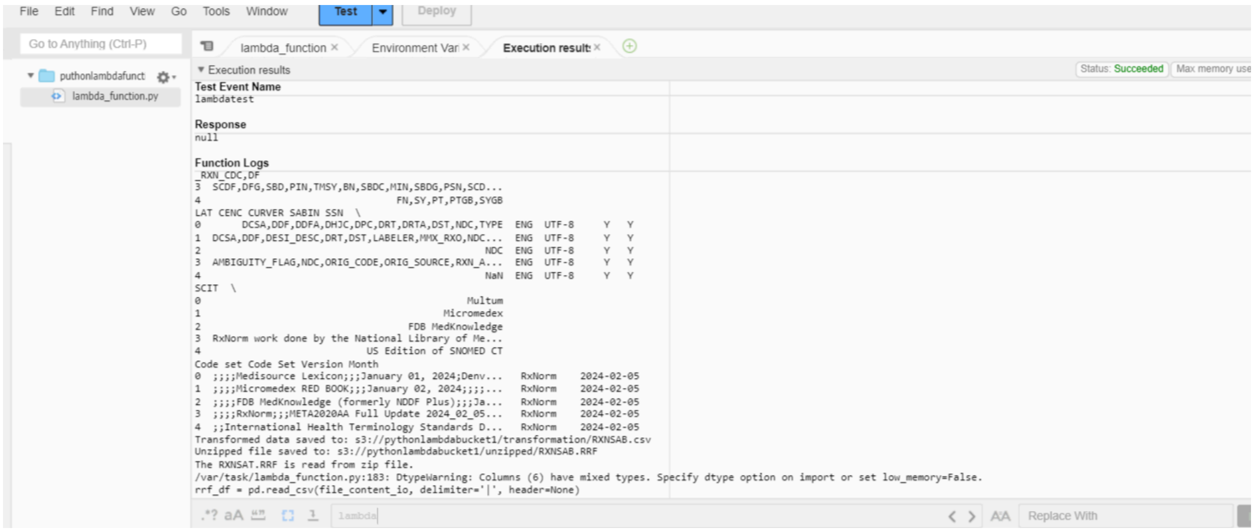
```
No older events at this moment. Retry
2024-03-11T16:16:34.983+05:45   INIT_START Runtime Version: python:3.12.v20 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:82aea00f37a44d68665730d559c81352fc95f22c9aa34a0722815a93f08e102b
2024-03-11T16:16:37.875+05:45   START RequestId: 1759ad2b-347b-4444-939a-6795a3a607a2 Version: $LATEST
2024-03-11T16:16:37.877+05:45   S3 event structure is not as expected. Missing 'Records' key.
S3 event structure is not as expected. Missing 'Records' key.
2024-03-11T16:16:37.882+05:45   END RequestId: 1759ad2b-347b-4444-939a-6795a3a607a2
2024-03-11T16:16:37.882+05:45   REPORT RequestId: 1759ad2b-347b-4444-939a-6795a3a607a2 Duration: 4.84 ms Billed Duration: 5 ms Memory Size: 10240 MB Max Memory Used: 190 MB Init Duration: 2889.69 ms
No newer events at this moment. Auto retry paused. Resume
```

10) Resolving errors and testing the code.



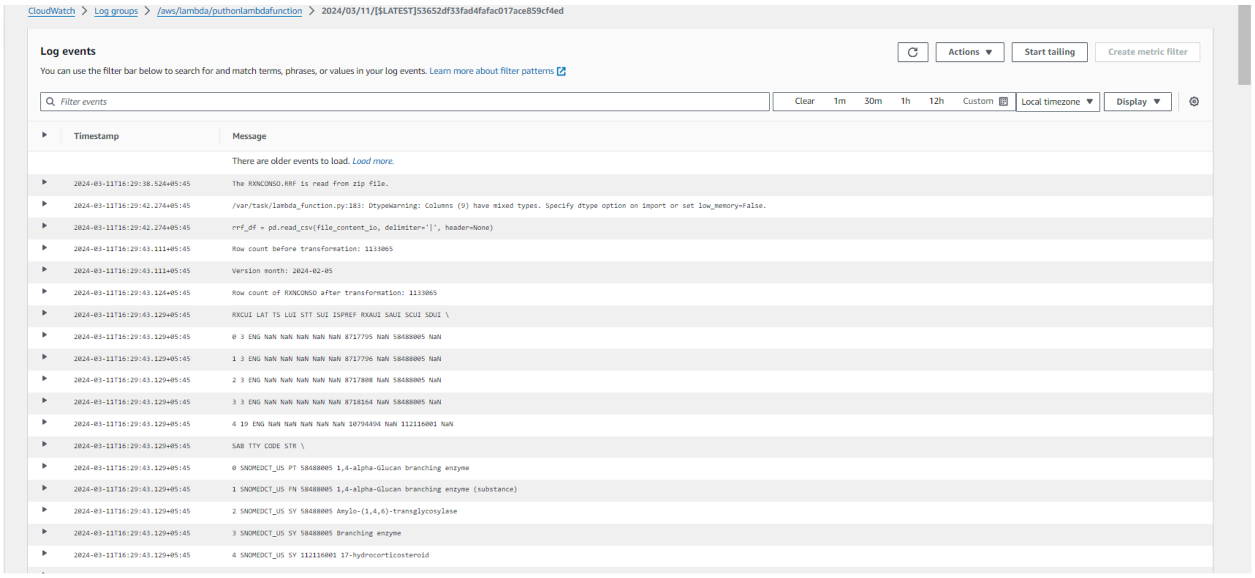
```
Loading...
Execution started: 16:27:35 (1 minute ago)
```

11) Test done successfully.



```
Test Event Name
lambdatest

Response
null

Function Logs
_RXN_CDC,DF
3  SCDF,DFG,SBD,PIN,TMSY,BN,SBDC,MIN,SBDG,PSN,SCD...
4                             FN,SY,PT,PTGB,SYGB
LAT CENC CURVER SABIN SSN  \
0       DCSA,DDF,DDFA,DHJC,DPC,DRT,DRTA,DST,NDC,TYPE  ENG  UTF-8     Y   Y
1  DCSA,DDF,DESI_DESC,DRT,DST,LABELER,MMX_RXO,NDC...  ENG  UTF-8     Y   Y
2                                               NDC  ENG  UTF-8     Y   Y
3  AMBIGUITY_FLAG,NDC,ORIG_CODE,ORIG_SOURCE,RXN_A...  ENG  UTF-8     Y   Y
4                                               NaN  ENG  UTF-8     Y   Y
SCIT  \
0                                           Multum
1                                       Micromedex
2                                   FDB MedKnowledge
3  RxNorm work done by the National Library of Me...
4                             US Edition of SNOMED CT
Code set Code Set Version Month
0  ;;;;Medisource Lexicon;;;January 01, 2024;Denv...   RxNorm    2024-02-05
1  ;;;;Micromedex RED BOOK;;;January 02, 2024;;;;...   RxNorm    2024-02-05
2  ;;;;FDB MedKnowledge (formerly NDDF Plus);;;Ja...   RxNorm    2024-02-05
3  ;;;;RxNorm;;;META2020AA Full Update 2024_02_05...   RxNorm    2024-02-05
4  ;;International Health Terminology Standards D...   RxNorm    2024-02-05
Transformed data saved to: s3://pythonlambdabucket1/transformation/RXNSAB.csv
Unzipped file saved to: s3://pythonlambdabucket1/unzipped/RXNSAB.RRF
The RXNSAT.RRF is read from zip file.
/var/task/lambda_function.py:183: DtypeWarning: Columns (6) have mixed types. Specify dtype option on import or set low_memory=False.
rrf_df = pd.read_csv(file_content_io, delimiter='|', header=None)
```

12) Logs Events in Cloudwatch



CloudWatch > Log groups > /aws/lambda/puthonlambdafunction > 2024/03/11/[$LATEST]53652df33fad4fafac017ace859cf4ed

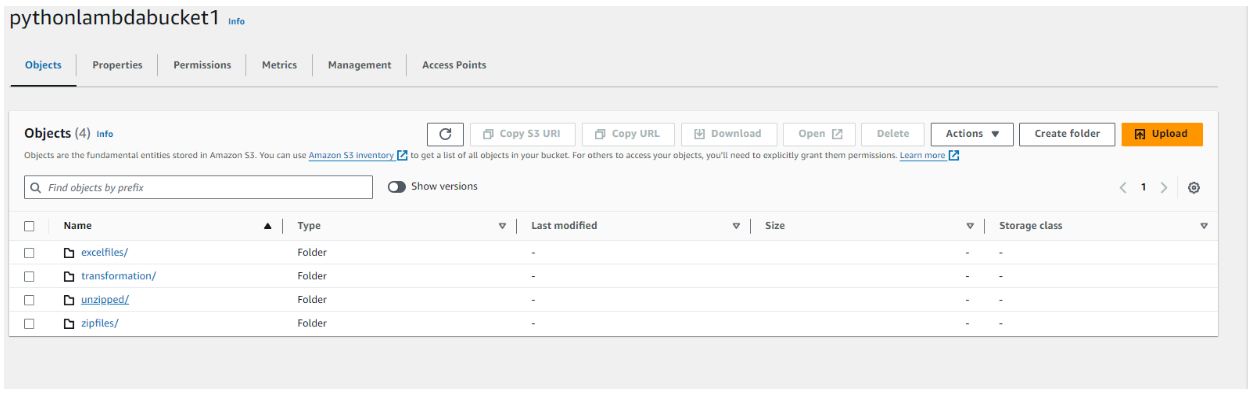
Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

There are older events to load. Load more.

```
Timestamp                      Message
2024-03-11T16:29:38.524+05:45  The RXNCONSO.RRF is read from zip file.
2024-03-11T16:29:42.274+05:45  /var/task/lambda_function.py:183: DtypeWarning: Columns (9) have mixed types. Specify dtype option on import or set low_memory=False.
2024-03-11T16:29:42.274+05:45  rrf_df = pd.read_csv(file_content_io, delimiter='|', header=None)
2024-03-11T16:29:43.111+05:45  Row count before transformation: 1133065
2024-03-11T16:29:43.111+05:45  Version month: 2024-02-05
2024-03-11T16:29:43.124+05:45  Row count of RXNCONSO after transformation: 1133065
2024-03-11T16:29:43.129+05:45  RXCUI LAT TS LUI STT SUI ISPREF RXAUI SAUI SCUI SDUI \
2024-03-11T16:29:43.129+05:45  0 3 ENG NaN NaN NaN NaN NaN 8717795 NaN 58488005 NaN
2024-03-11T16:29:43.129+05:45  1 3 ENG NaN NaN NaN NaN NaN 8717796 NaN 58488005 NaN
2024-03-11T16:29:43.129+05:45  2 3 ENG NaN NaN NaN NaN NaN 8717808 NaN 58488005 NaN
2024-03-11T16:29:43.129+05:45  3 3 ENG NaN NaN NaN NaN NaN 8718164 NaN 58488005 NaN
2024-03-11T16:29:43.129+05:45  4 19 ENG NaN NaN NaN NaN NaN 10794494 NaN 112116001 NaN
2024-03-11T16:29:43.129+05:45  SAB TTY CODE STR \
2024-03-11T16:29:43.129+05:45  0 SNOMEDCT_US PT 58488005 1,4-alpha-Glucan branching enzyme
2024-03-11T16:29:43.129+05:45  1 SNOMEDCT_US FN 58488005 1,4-alpha-Glucan branching enzyme (substance)
2024-03-11T16:29:43.129+05:45  2 SNOMEDCT_US SY 58488005 Amylo-(1,4,6)-transglycosylase
2024-03-11T16:29:43.129+05:45  3 SNOMEDCT_US SY 58488005 Branching enzyme
2024-03-11T16:29:43.129+05:45  4 SNOMEDCT_US SY 112116001 17-hydrocorticosteroid
```

13) The two files transformed and unzipped created successfully.



pythonlambdabucket1 Info

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

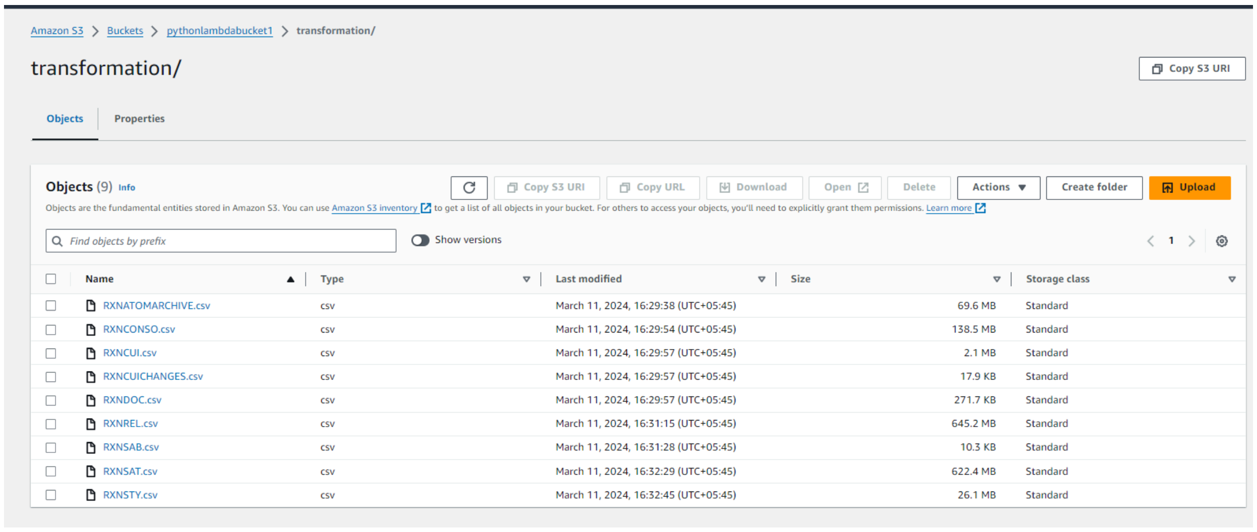
Objects (4) Info

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
excelfiles/	Folder	-	-	-
transformation/	Folder	-	-	-
unzipped/	Folder	-	-	-
zipfiles/	Folder	-	-	-

14) Transformation

After execution, headers are added, delimiter is changed to comma(,), and date format is changed.



Amazon S3 > Buckets > pythonlambdabucket1 > transformation/

transformation/

Objects | Properties

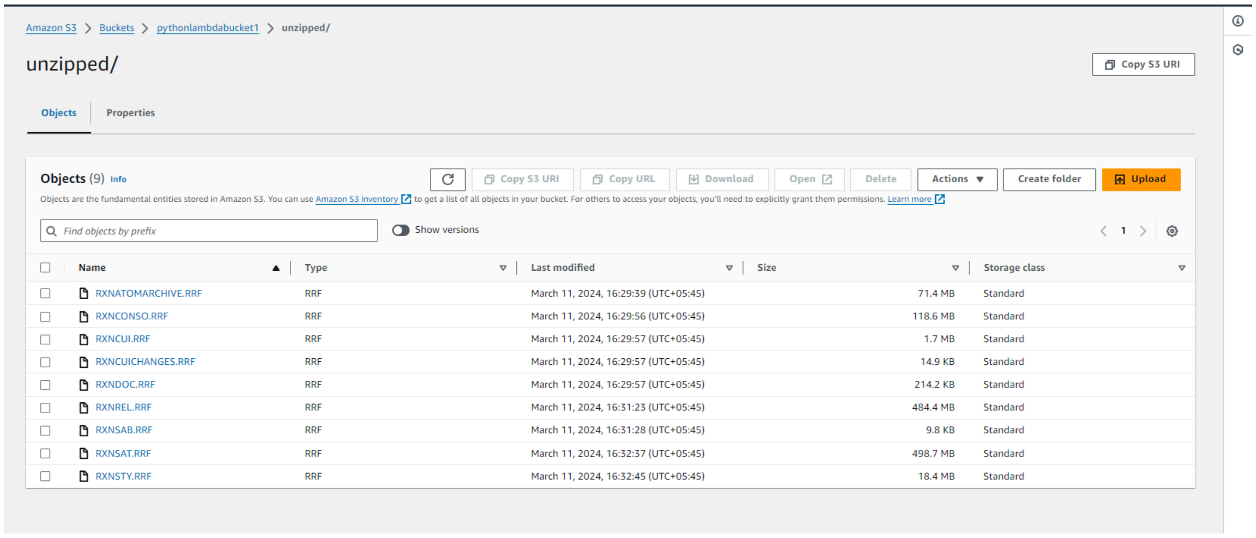
Objects (9) Info

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
RXNATOMARCHIVE.csv	csv	March 11, 2024, 16:29:38 (UTC+05:45)	69.6 MB	Standard
RXNCONSO.csv	csv	March 11, 2024, 16:29:54 (UTC+05:45)	138.5 MB	Standard
RXNCUI.csv	csv	March 11, 2024, 16:29:57 (UTC+05:45)	2.1 MB	Standard
RXNCUICHANGES.csv	csv	March 11, 2024, 16:29:57 (UTC+05:45)	17.9 KB	Standard
RXNDOC.csv	csv	March 11, 2024, 16:29:57 (UTC+05:45)	271.7 KB	Standard
RXNREL.csv	csv	March 11, 2024, 16:31:15 (UTC+05:45)	645.2 MB	Standard
RXNSAB.csv	csv	March 11, 2024, 16:31:28 (UTC+05:45)	10.3 KB	Standard
RXNSAT.csv	csv	March 11, 2024, 16:32:29 (UTC+05:45)	622.4 MB	Standard
RXNSTY.csv	csv	March 11, 2024, 16:32:45 (UTC+05:45)	26.1 MB	Standard

15) Unzipped

The zipped file is unzipped successfully.



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
RXNATOMARCHIVE.RRF	RRF	March 11, 2024, 16:29:39 (UTC+05:45)	71.4 MB	Standard
RXNCONSO.RRF	RRF	March 11, 2024, 16:29:56 (UTC+05:45)	118.6 MB	Standard
RXNCUI.RRF	RRF	March 11, 2024, 16:29:57 (UTC+05:45)	1.7 MB	Standard
RXNCUICHANGES.RRF	RRF	March 11, 2024, 16:29:57 (UTC+05:45)	14.9 KB	Standard
RXNDOC.RRF	RRF	March 11, 2024, 16:29:57 (UTC+05:45)	214.2 KB	Standard
RXNREL.RRF	RRF	March 11, 2024, 16:31:23 (UTC+05:45)	484.4 MB	Standard
RXNSAB.RRF	RRF	March 11, 2024, 16:31:28 (UTC+05:45)	9.8 KB	Standard
RXNSAT.RRF	RRF	March 11, 2024, 16:32:37 (UTC+05:45)	498.7 MB	Standard
RXNSTY.RRF	RRF	March 11, 2024, 16:32:45 (UTC+05:45)	18.4 MB	Standard