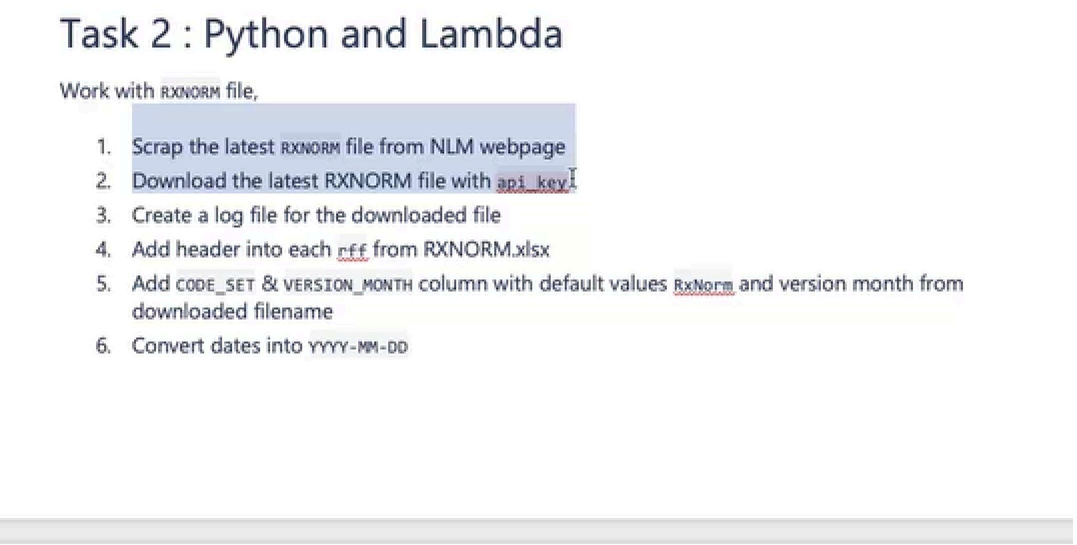
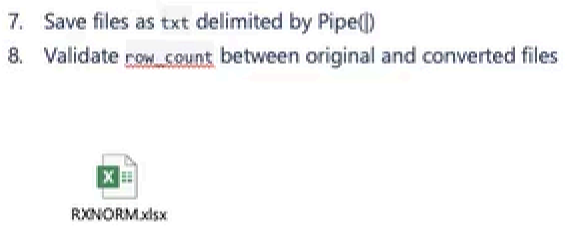


Task 2 : Python and Lambda

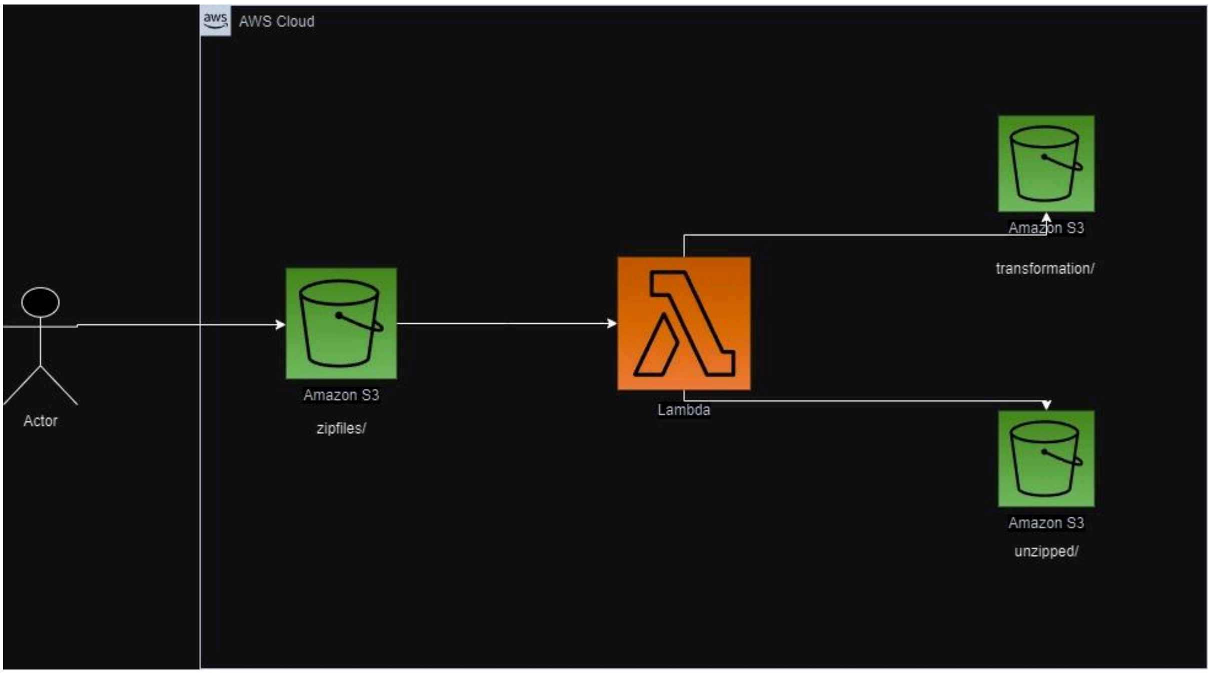
Work with RXNORM file,

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

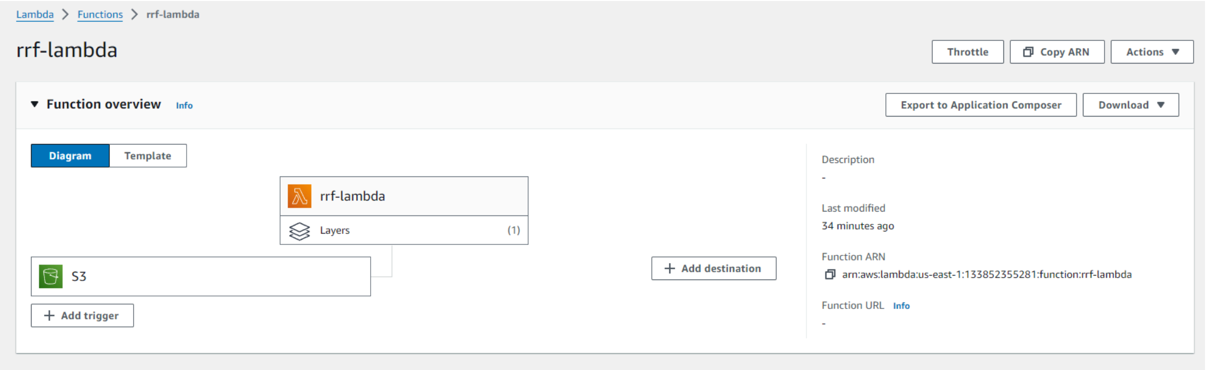


7. Save files as txt delimited by Pipe(|)
8. Validate row_count between original and converted files

The Cloud architecture of the given project is designed as follow:



Step 1: Proper setup of aws was done: lambda was created and s3 bucket was setup with four folders- zipfiles/ (to input the zipdata), excelfiles/(to input the excel file), unzipped/(to hold the unzipped files) and finally transformation/(to hold the final transformed data in .txt format)

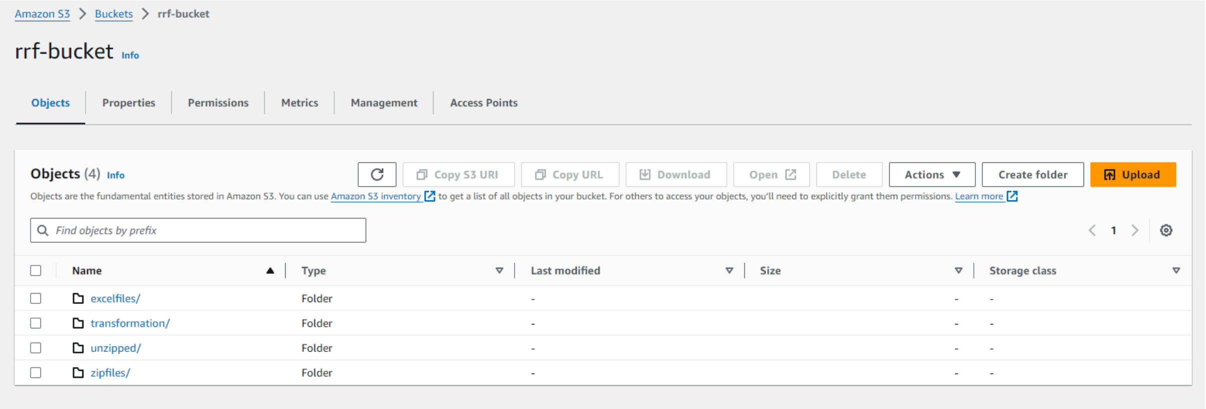


Description
-

Last modified
34 minutes ago

Function ARN
arn:aws:lambda:us-east-1:133852355281:function:rrf-lambda

Function URL Info
-



Objects (4) Info

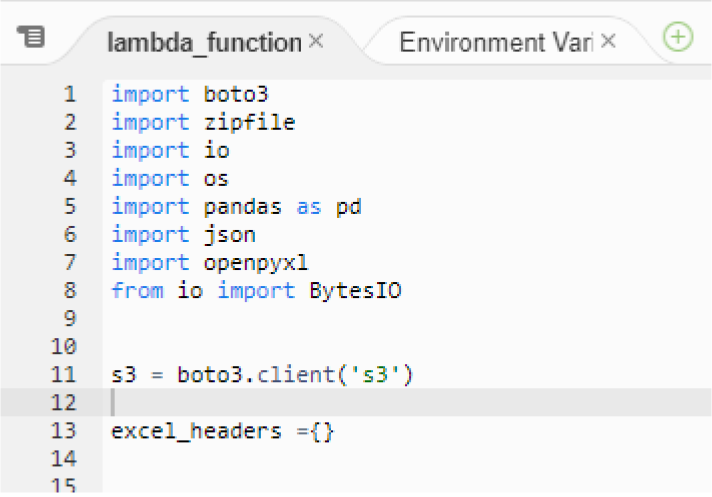
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	excelfiles/	Folder	-	-	-
□	transformation/	Folder	-	-	-
□	unzipped/	Folder	-	-	-
□	zipfiles/	Folder	-	-	-

Here, unzipped and transformation folder are created after lambda function is triggered

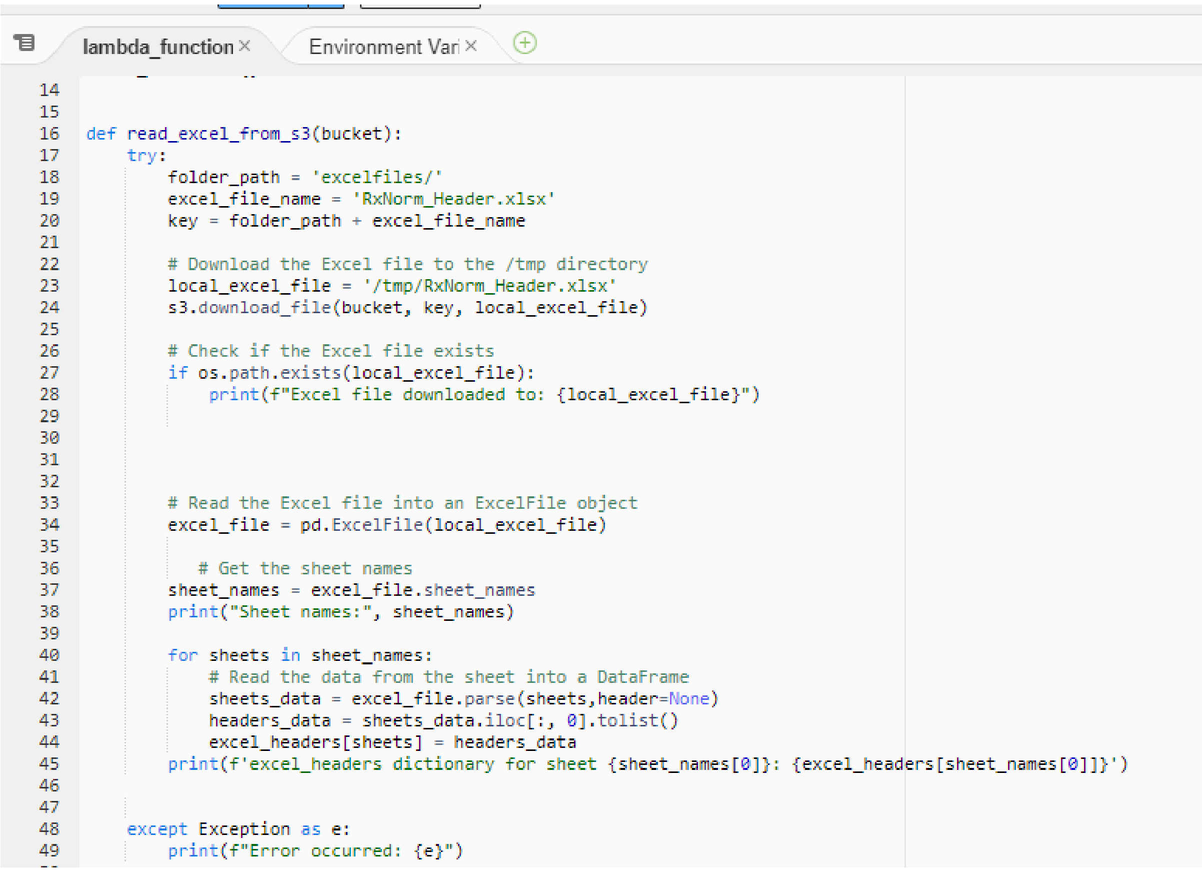
Step 2: The code structure

A. The imports and excel_headers dictionary was initiated to hold the headers to be assigned to the rrf files



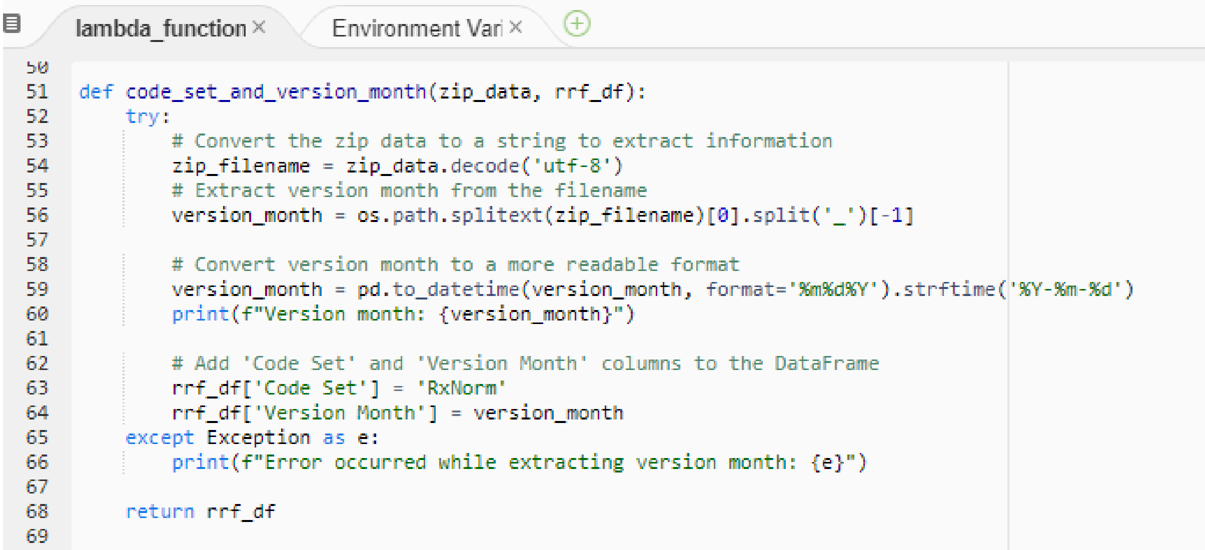
```python
import boto3
import zipfile
import io
import os
import pandas as pd
import json
import openpyxl
from io import BytesIO


s3 = boto3.client('s3')

excel_headers ={}

```

B. Function to read the excel_file and update the excel_header



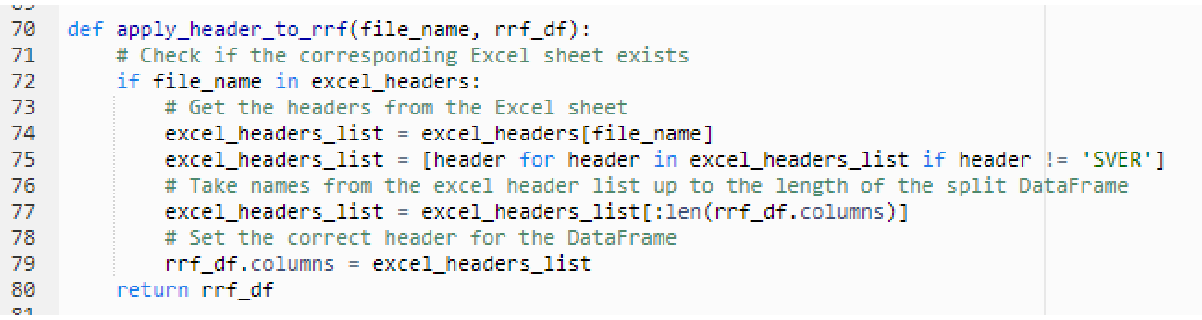
```
def read_excel_from_s3(bucket):
    try:
        folder_path = 'excelfiles/'
        excel_file_name = 'RxNorm_Header.xlsx'
        key = folder_path + excel_file_name

        # Download the Excel file to the /tmp directory
        local_excel_file = '/tmp/RxNorm_Header.xlsx'
        s3.download_file(bucket, key, local_excel_file)

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




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

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

        for sheets in sheet_names:
            # Read the data from the sheet into a DataFrame
            sheets_data = excel_file.parse(sheets,header=None)
            headers_data = sheets_data.iloc[:, 0].tolist()
            excel_headers[sheets] = headers_data
        print(f'excel_headers dictionary for sheet {sheet_names[0]}: {excel_headers[sheet_names[0]]}')


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

C. Function to add Code Set and Version Month: Code set was set to RXNORM and version month was extracted from the zip file name which is RxNorm_full_02052024.zip
Hence Version Month is: 2024-05-02



```
def code_set_and_version_month(zip_data, rrf_df):
    try:
        # Convert the zip data to a string to extract information
        zip_filename = zip_data.decode('utf-8')
        # 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
```

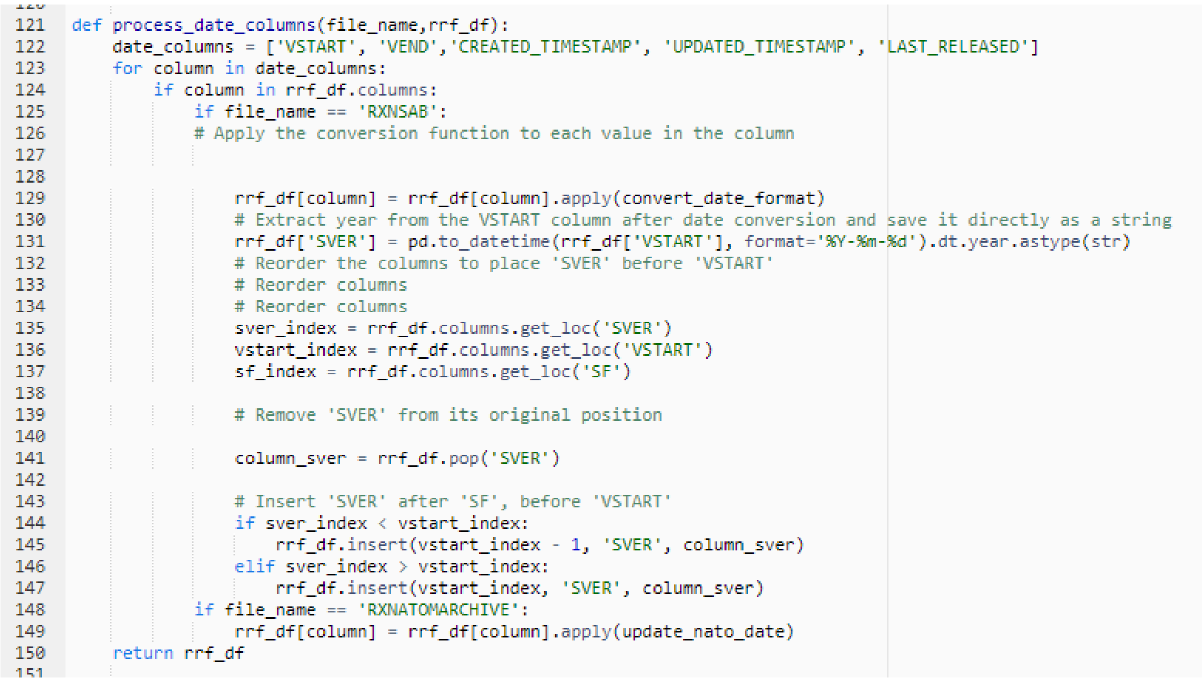
D. Function to apply header to rrf from excel_header



```
def apply_header_to_rrf(file_name, rrf_df):
    # Check if the corresponding Excel sheet exists
    if file_name in excel_headers:
        # Get the headers from the Excel sheet
        excel_headers_list = excel_headers[file_name]
        excel_headers_list = [header for header in excel_headers_list if header != 'SVER']
        # Take names from the excel header list up to the length of the split DataFrame
        excel_headers_list = excel_headers_list[:len(rrf_df.columns)]
        # Set the correct header for the DataFrame
        rrf_df.columns = excel_headers_list
    return rrf_df
```

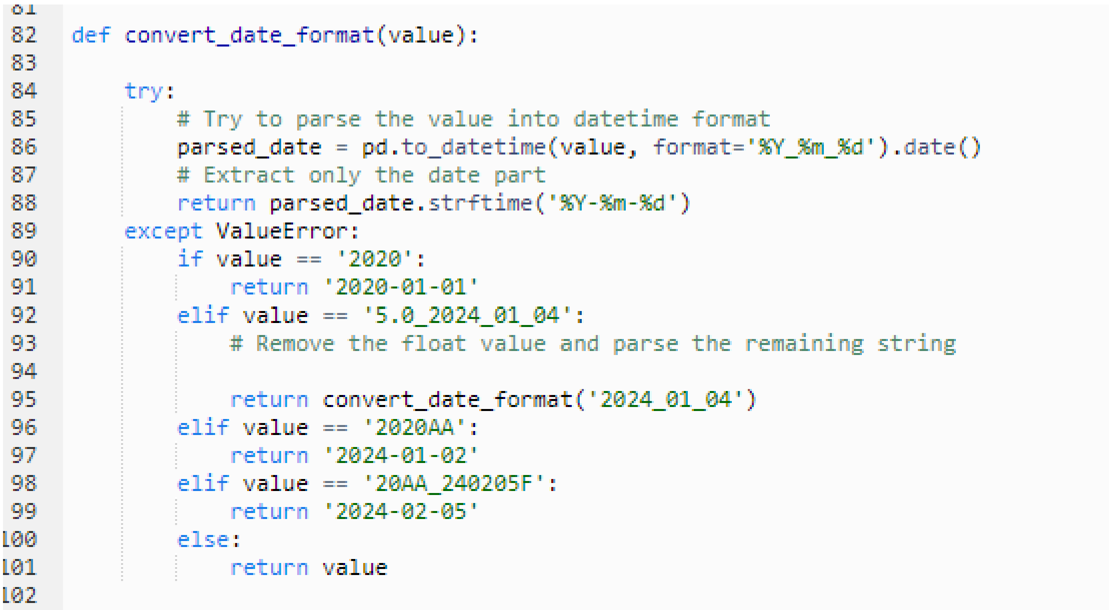
E. Function to convert date to indicated format

a. Here, in the context of RXNSAB, the column SVER derived its value from extracting year part from the VSTART column as done below



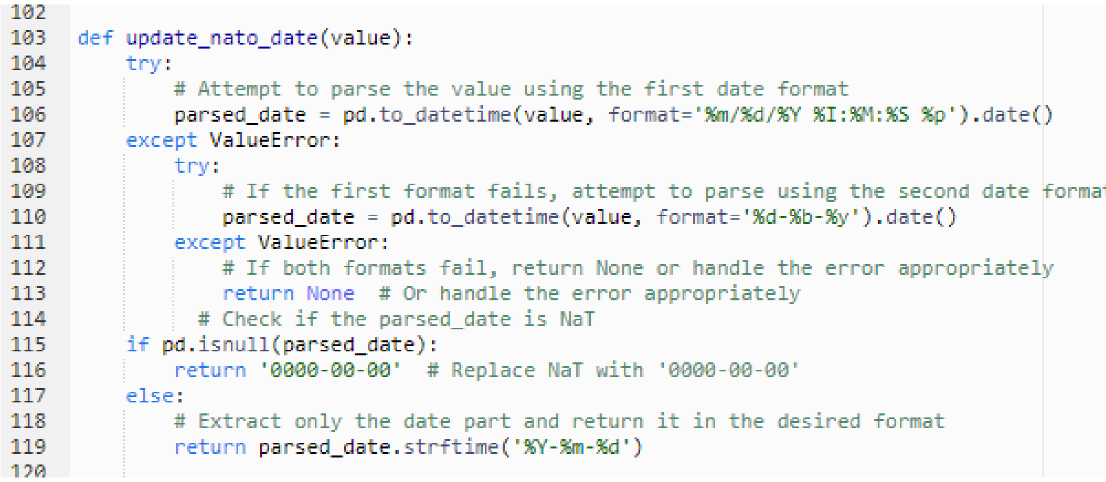
```
def process_date_columns(file_name,rrf_df):
    date_columns = ['VSTART', 'VEND','CREATED_TIMESTAMP', 'UPDATED_TIMESTAMP', 'LAST_RELEASED']
    for column in date_columns:
        if column in rrf_df.columns:
            if file_name == 'RXNSAB':
            # Apply the conversion function to each value in the column


                rrf_df[column] = rrf_df[column].apply(convert_date_format)
                # Extract year from the VSTART column after date conversion and save it directly as a string
                rrf_df['SVER'] = pd.to_datetime(rrf_df['VSTART'], format='%Y-%m-%d').dt.year.astype(str)
                # Reorder the columns to place 'SVER' before 'VSTART'
                # Reorder columns
                # Reorder columns
                sver_index = rrf_df.columns.get_loc('SVER')
                vstart_index = rrf_df.columns.get_loc('VSTART')
                sf_index = rrf_df.columns.get_loc('SF')

                # Remove 'SVER' from its original position

                column_sver = rrf_df.pop('SVER')

                # Insert 'SVER' after 'SF', before 'VSTART'
                if sver_index < vstart_index:
                    rrf_df.insert(vstart_index - 1, 'SVER', column_sver)
                elif sver_index > vstart_index:
                    rrf_df.insert(vstart_index, 'SVER', column_sver)
            if file_name == 'RXNATOMARCHIVE':
                rrf_df[column] = rrf_df[column].apply(update_nato_date)
    return rrf_df
```

b. Now, the anomalies in the VSTART column of RXNSAB is processed as below:



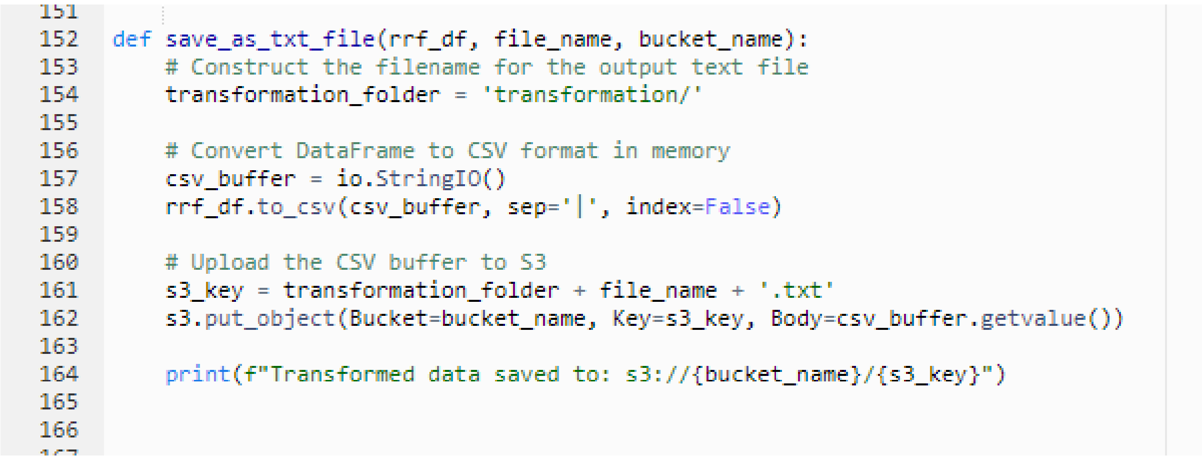
```python
def convert_date_format(value):

    try:
        # Try to parse the value into datetime format
        parsed_date = pd.to_datetime(value, format='%Y_%m_%d').date()
        # Extract only the date part
        return parsed_date.strftime('%Y-%m-%d')
    except ValueError:
        if value == '2020':
            return '2020-01-01'
        elif value == '5.0_2024_01_04':
            # Remove the float value and parse the remaining string

            return convert_date_format('2024_01_04')
        elif value == '2020AA':
            return '2024-01-02'
        elif value == '20AA_240205F':
            return '2024-02-05'
        else:
            return value
```

c. Now, different date formats of RXNATOMARCHIVE is handled as below:



```python
def update_nato_date(value):
    try:
        # Attempt to parse the value using the first date format
        parsed_date = pd.to_datetime(value, format='%m/%d/%Y %I:%M:%S %p').date()
    except ValueError:
        try:
            # If the first format fails, attempt to parse using the second date forma
            parsed_date = pd.to_datetime(value, format='%d-%b-%y').date()
        except ValueError:
            # If both formats fail, return None or handle the error appropriately
            return None  # Or handle the error appropriately
          # Check if the parsed_date is NaT
    if pd.isnull(parsed_date):
        return '0000-00-00'  # Replace NaT with '0000-00-00'
    else:
        # Extract only the date part and return it in the desired format
        return parsed_date.strftime('%Y-%m-%d')
```

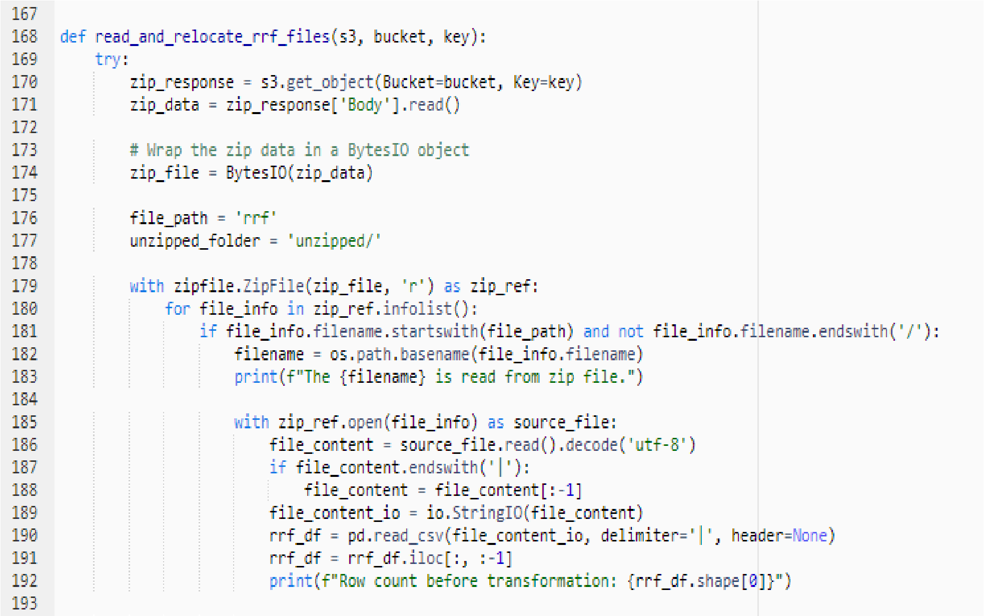
F. Now the function to save the final transformed data as .txt delimited by pipe



```python

def save_as_txt_file(rrf_df, file_name, bucket_name):
    # Construct the filename for the output text file
    transformation_folder = 'transformation/'

    # Convert DataFrame to CSV format in memory
    csv_buffer = io.StringIO()
    rrf_df.to_csv(csv_buffer, sep='|', index=False)

    # Upload the CSV buffer to S3
    s3_key = transformation_folder + file_name + '.txt'
    s3.put_object(Bucket=bucket_name, Key=s3_key, Body=csv_buffer.getvalue())

    print(f"Transformed data saved to: s3://{bucket_name}/{s3_key}")


```

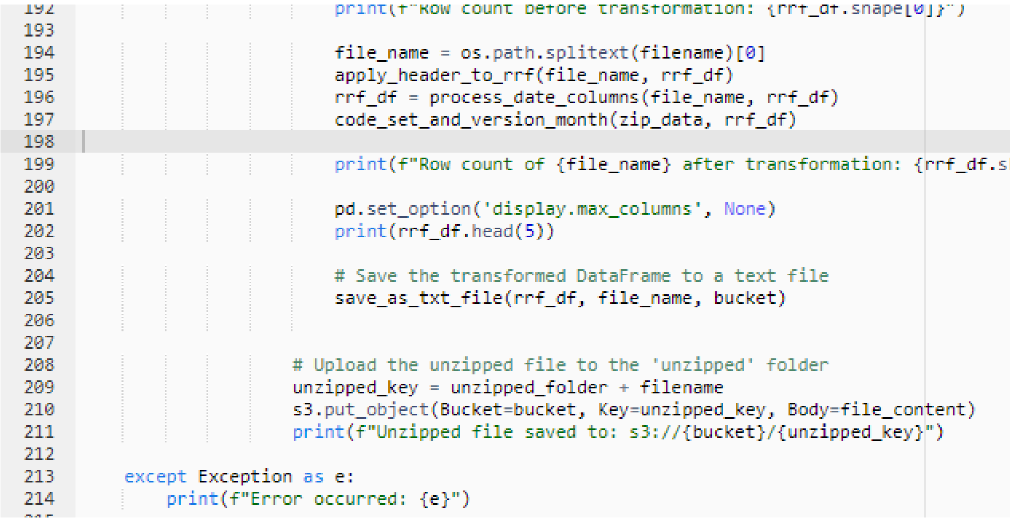
G. The main function where the previous functions are called:

a. Here zip files are read from the proper location and the last pipe is ignored by splitting



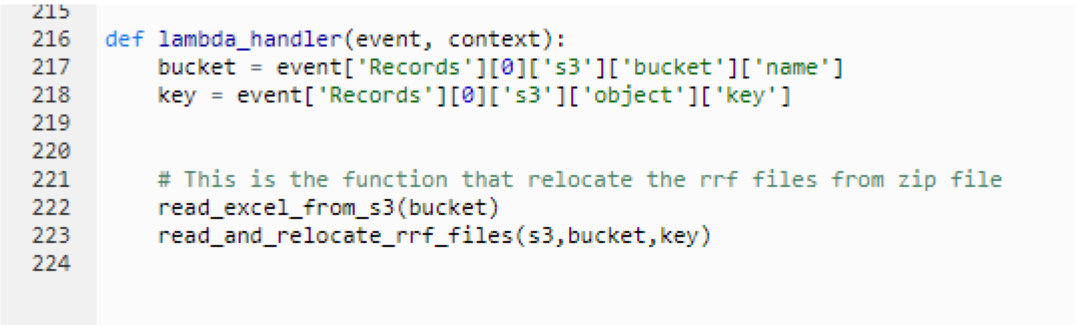
```python

def read_and_relocate_rrf_files(s3, bucket, key):
    try:
        zip_response = s3.get_object(Bucket=bucket, Key=key)
        zip_data = zip_response['Body'].read()

        # Wrap the zip data in a BytesIO object
        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_df = pd.read_csv(file_content_io, delimiter='|', header=None)
                        rrf_df = rrf_df.iloc[:, :-1]
                        print(f"Row count before transformation: {rrf_df.shape[0]}")

```

b. Previous functions are called to transform according to the requirements and unzipped files are placed in proper folder



```python
                        print(f"Row count before transformation: {rrf_df.shape[0]}")

                        file_name = os.path.splitext(filename)[0]
                        apply_header_to_rrf(file_name, rrf_df)
                        rrf_df = process_date_columns(file_name, rrf_df)
                        code_set_and_version_month(zip_data, rrf_df)

                        print(f"Row count of {file_name} after transformation: {rrf_df.s

                        pd.set_option('display.max_columns', None)
                        print(rrf_df.head(5))

                        # Save the transformed DataFrame to a text file
                        save_as_txt_file(rrf_df, file_name, bucket)


                    # Upload the unzipped file to the 'unzipped' folder
                    unzipped_key = unzipped_folder + filename
                    s3.put_object(Bucket=bucket, Key=unzipped_key, Body=file_content)
                    print(f"Unzipped file saved to: s3://{bucket}/{unzipped_key}")

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

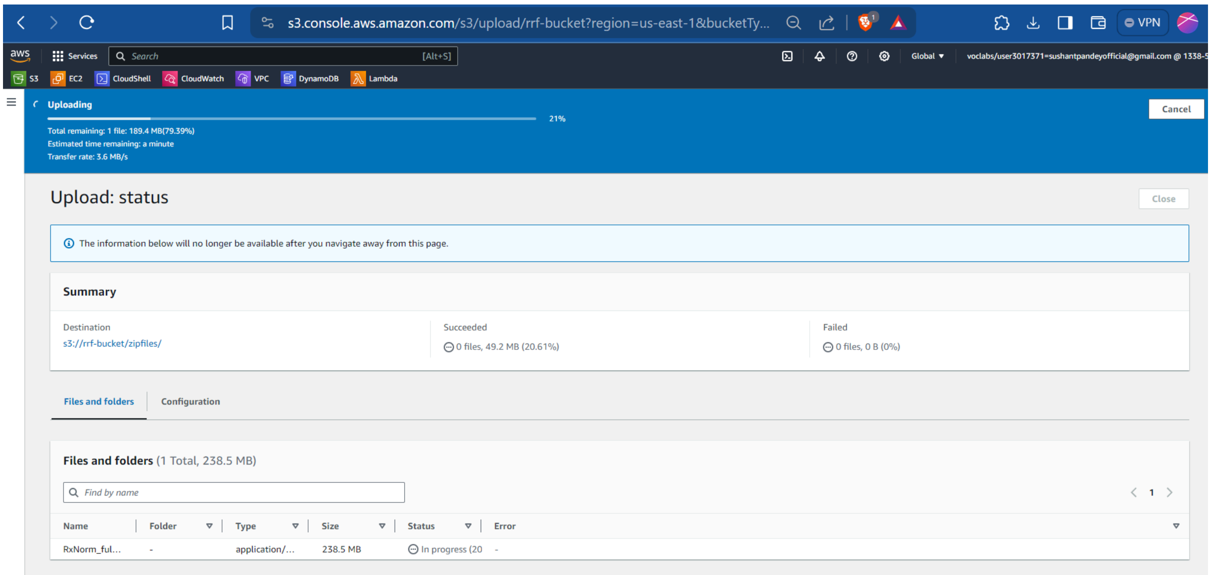
H. The main lambda function



```
def lambda_handler(event, context):
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']


    # This is the function that relocate the rrf files from zip file
    read_excel_from_s3(bucket)
    read_and_relocate_rrf_files(s3,bucket,key)
```

Step 3: Results and analysis

A. Zip file is uploaded to zipfiles/ in order to trigger the lambda and excel file is loaded to excelfiles/



s3.console.aws.amazon.com/s3/upload/rrf-bucket?region=us-east-1&bucketTy...
VPN
aws
Services
Search
[Alt+S]
Global ▾
voclabs/user3017371=sushantpandeyofficial@gmail.com @ 1338-5
S3
EC2
CloudShell
CloudWatch
VPC
DynamoDB
Lambda
Cancel
Close
Files and folders
Configuration
Find by name
1

Uploading 21%

Total remaining: 1 file: 189.4 MB(79.39%)
Estimated time remaining: a minute
Transfer rate: 3.6 MB/s

Upload: status

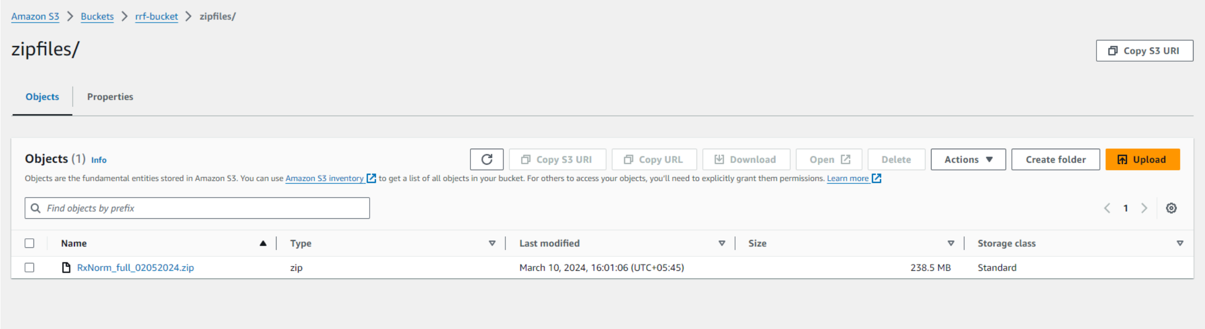
The information below will no longer be available after you navigate away from this page.

Summary

Destination	Succeeded	Failed
s3://rrf-bucket/zipfiles/	0 files, 49.2 MB (20.61%)	0 files, 0 B (0%)

Files and folders (1 Total, 238.5 MB)

Name	Folder	Type	Size	Status	Error
RxNorm_ful...	-	application/...	238.5 MB	In progress (20	-

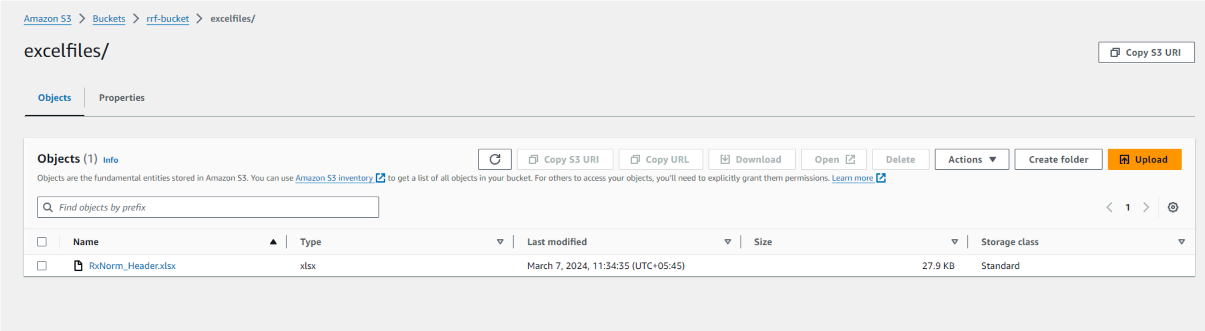


zipfiles/

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 10, 2024, 16:01:06 (UTC+05:45)	238.5 MB	Standard



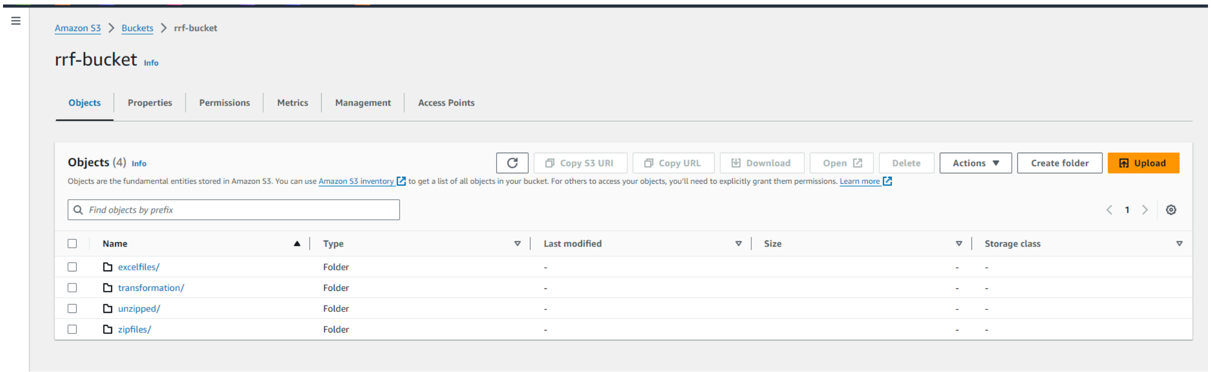
excelfiles/

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_Header.xlsx	xlsx	March 7, 2024, 11:34:35 (UTC+05:45)	27.9 KB	Standard

B. unzipped/ and transformation/ folders were created after trigger

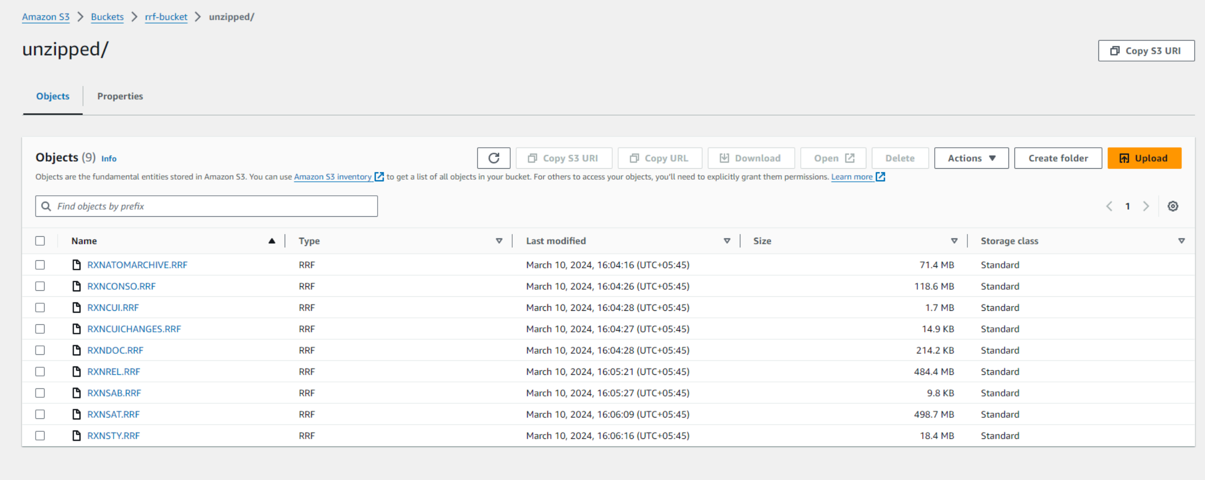


rrf-bucket Info

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	-	-	-

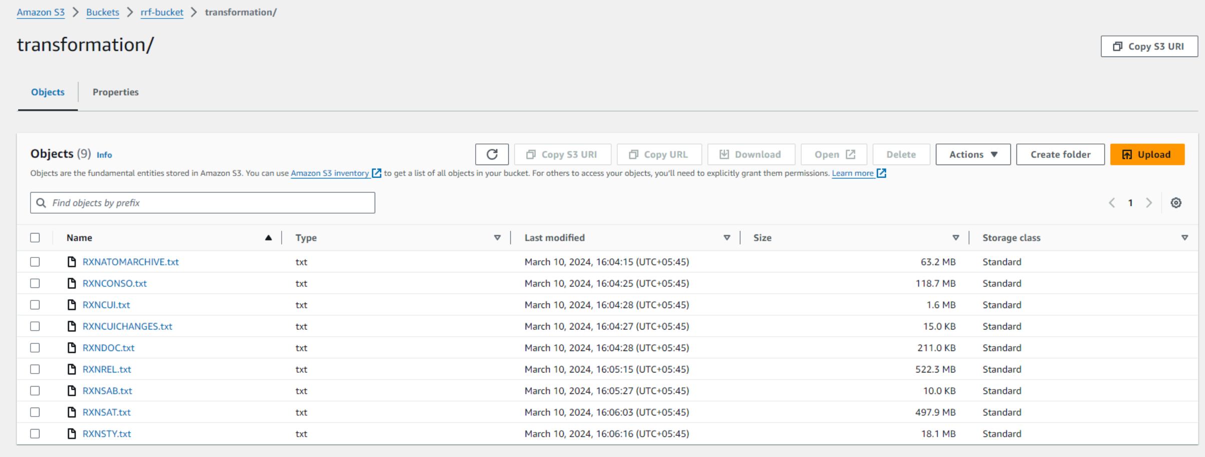


unzipped/

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.RRF	RRF	March 10, 2024, 16:04:16 (UTC+05:45)	71.4 MB	Standard
□	RXNCONSO.RRF	RRF	March 10, 2024, 16:04:26 (UTC+05:45)	118.6 MB	Standard
□	RXNCUI.RRF	RRF	March 10, 2024, 16:04:28 (UTC+05:45)	1.7 MB	Standard
□	RXNCUICHANGES.RRF	RRF	March 10, 2024, 16:04:27 (UTC+05:45)	14.9 KB	Standard
□	RXNDOC.RRF	RRF	March 10, 2024, 16:04:28 (UTC+05:45)	214.2 KB	Standard
□	RXNREL.RRF	RRF	March 10, 2024, 16:05:21 (UTC+05:45)	484.4 MB	Standard
□	RXNSAB.RRF	RRF	March 10, 2024, 16:05:27 (UTC+05:45)	9.8 KB	Standard
□	RXNSAT.RRF	RRF	March 10, 2024, 16:06:09 (UTC+05:45)	498.7 MB	Standard
□	RXNSTY.RRF	RRF	March 10, 2024, 16:06:16 (UTC+05:45)	18.4 MB	Standard



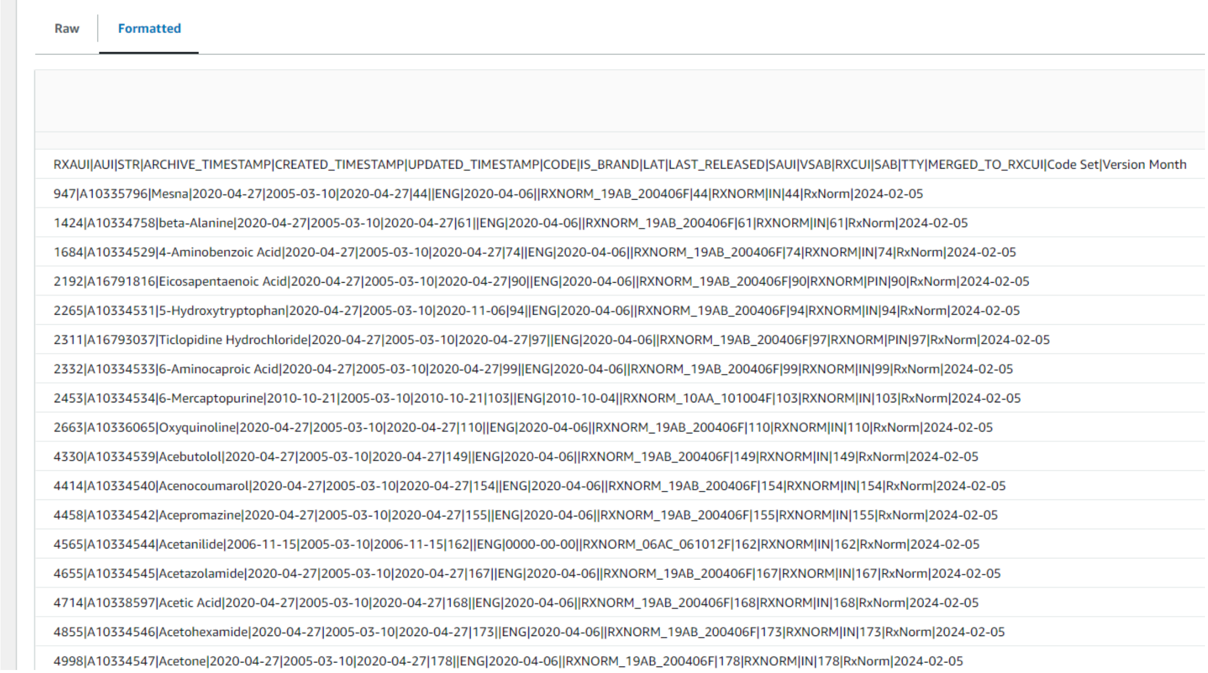
transformation/

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

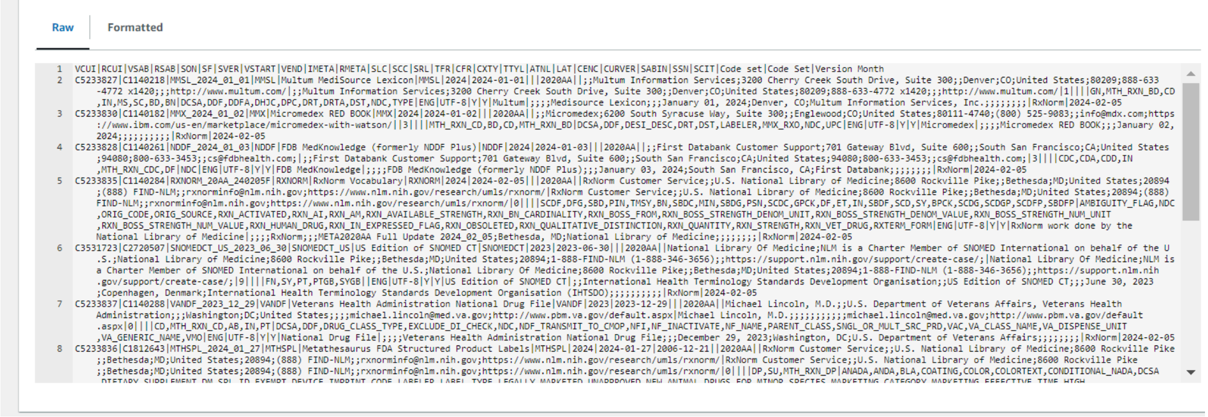
□	Name	Type	Last modified	Size	Storage class
□	RXNATOMARCHIVE.txt	txt	March 10, 2024, 16:04:15 (UTC+05:45)	63.2 MB	Standard
□	RXNCONSO.txt	txt	March 10, 2024, 16:04:25 (UTC+05:45)	118.7 MB	Standard
□	RXNCUI.txt	txt	March 10, 2024, 16:04:28 (UTC+05:45)	1.6 MB	Standard
□	RXNCUICHANGES.txt	txt	March 10, 2024, 16:04:27 (UTC+05:45)	15.0 KB	Standard
□	RXNDOC.txt	txt	March 10, 2024, 16:04:28 (UTC+05:45)	211.0 KB	Standard
□	RXNREL.txt	txt	March 10, 2024, 16:05:15 (UTC+05:45)	522.3 MB	Standard
□	RXNSAB.txt	txt	March 10, 2024, 16:05:27 (UTC+05:45)	10.0 KB	Standard
□	RXNSAT.txt	txt	March 10, 2024, 16:06:03 (UTC+05:45)	497.9 MB	Standard
□	RXNSTY.txt	txt	March 10, 2024, 16:06:16 (UTC+05:45)	18.1 MB	Standard

C. Here is the RXNATOMARCHIVE.rrf files after transformation of dates in the txt format



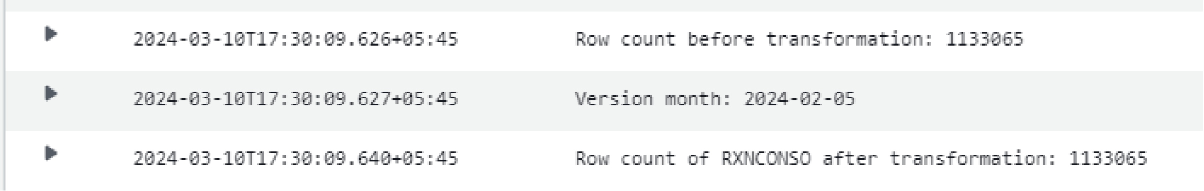
```
RXAUI|AUI|STR|ARCHIVE_TIMESTAMP|CREATED_TIMESTAMP|UPDATED_TIMESTAMP|CODE|IS_BRAND|LAT|LAST_RELEASED|SAUI|VSAB|RXCUI|SAB|TTY|MERGED_TO_RXCUI|Code Set|Version Month
947|A10335796|Mesna|2020-04-27|2005-03-10|2020-04-27|44||ENG|2020-04-06||RXNORM_19AB_200406F|44|RXNORM|IN|44|RxNorm|2024-02-05
1424|A10334758|beta-Alanine|2020-04-27|2005-03-10|2020-04-27|61||ENG|2020-04-06||RXNORM_19AB_200406F|61|RXNORM|IN|61|RxNorm|2024-02-05
1684|A10334529|4-Aminobenzoic Acid|2020-04-27|2005-03-10|2020-04-27|74||ENG|2020-04-06||RXNORM_19AB_200406F|74|RXNORM|IN|74|RxNorm|2024-02-05
2192|A16791816|Eicosapentaenoic Acid|2020-04-27|2005-03-10|2020-04-27|90||ENG|2020-04-06||RXNORM_19AB_200406F|90|RXNORM|PIN|90|RxNorm|2024-02-05
2265|A10334531|5-Hydroxytryptophan|2020-04-27|2005-03-10|2020-11-06|94||ENG|2020-04-06||RXNORM_19AB_200406F|94|RXNORM|IN|94|RxNorm|2024-02-05
2311|A16793037|Ticlopidine Hydrochloride|2020-04-27|2005-03-10|2020-04-27|97||ENG|2020-04-06||RXNORM_19AB_200406F|97|RXNORM|PIN|97|RxNorm|2024-02-05
2332|A10334533|6-Aminocaproic Acid|2020-04-27|2005-03-10|2020-04-27|99||ENG|2020-04-06||RXNORM_19AB_200406F|99|RXNORM|IN|99|RxNorm|2024-02-05
2453|A10334534|6-Mercaptopurine|2010-10-21|2005-03-10|2010-10-21|103||ENG|2010-10-04||RXNORM_10AA_101004F|103|RXNORM|IN|103|RxNorm|2024-02-05
2663|A10336065|Oxyquinoline|2020-04-27|2005-03-10|2020-04-27|110||ENG|2020-04-06||RXNORM_19AB_200406F|110|RXNORM|IN|110|RxNorm|2024-02-05
4330|A10334539|Acebutolol|2020-04-27|2005-03-10|2020-04-27|149||ENG|2020-04-06||RXNORM_19AB_200406F|149|RXNORM|IN|149|RxNorm|2024-02-05
4414|A10334540|Acenocoumarol|2020-04-27|2005-03-10|2020-04-27|154||ENG|2020-04-06||RXNORM_19AB_200406F|154|RXNORM|IN|154|RxNorm|2024-02-05
4458|A10334542|Acepromazine|2020-04-27|2005-03-10|2020-04-27|155||ENG|2020-04-06||RXNORM_19AB_200406F|155|RXNORM|IN|155|RxNorm|2024-02-05
4565|A10334544|Acetanilide|2006-11-15|2005-03-10|2006-11-15|162||ENG|0000-00-00||RXNORM_06AC_061012F|162|RXNORM|IN|162|RxNorm|2024-02-05
4655|A10334545|Acetazolamide|2020-04-27|2005-03-10|2020-04-27|167||ENG|2020-04-06||RXNORM_19AB_200406F|167|RXNORM|IN|167|RxNorm|2024-02-05
4714|A10338597|Acetic Acid|2020-04-27|2005-03-10|2020-04-27|168||ENG|2020-04-06||RXNORM_19AB_200406F|168|RXNORM|IN|168|RxNorm|2024-02-05
4855|A10334546|Acetohexamide|2020-04-27|2005-03-10|2020-04-27|173||ENG|2020-04-06||RXNORM_19AB_200406F|173|RXNORM|IN|173|RxNorm|2024-02-05
4998|A10334547|Acetone|2020-04-27|2005-03-10|2020-04-27|178||ENG|2020-04-06||RXNORM_19AB_200406F|178|RXNORM|IN|178|RxNorm|2024-02-05
```

D. Here is the RXNSAB.rrf files transformed after adding SVER column and changing the date format

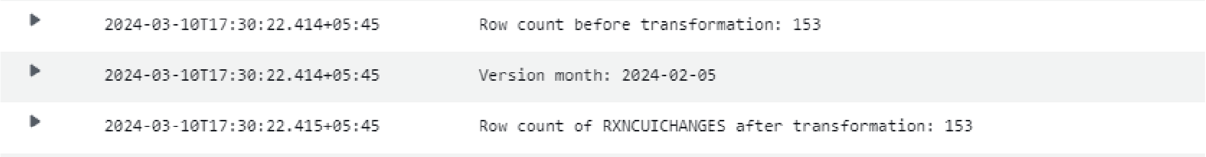


```
VCUI|RCUI|VSAB|RSAB|SON|SF|SVER|VSTART|VEND|IMETA|RMETA|SLC|SCC|SRL|TFR|CFR|CXTY|TTYL|ATNL|LAT|CENC|CURVER|SABIN|SSN|SCIT|Code set|Code Set|Version Month
C5233827|C1140218|MMSL_2024_01_01|MMSL|Multum MediSource Lexicon|MMSL|2024|2024-01-01|||2020AA||;;Multum Information Services;3200 Cherry Creek South Drive, Suite 300;;Denver;CO;United States;80209;888-633
    -4772 x1420;;;http://www.multum.com/|;;Multum Information Services;3200 Cherry Creek South Drive, Suite 300;;Denver;CO;United States;80209;888-633-4772 x1420;;;http://www.multum.com/|1||||GN,MTH_RXN_BD,CD
    ,IN,MS,SC,BD,BN|DCSA,DDF,DDFA,DHJC,DPC,DRT,DRTA,DST,NDC,TYPE|ENG|UTF-8|Y|Y|Multum|;;;;Medisource Lexicon;;;January 01, 2024;Denver, CO;Multum Information Services, Inc.;;;;;;;;|RxNorm|2024-02-05
C5233830|C1140182|MMX_2024_01_02|MMX|Micromedex RED BOOK|MMX|2024|2024-01-02|||2020AA||;;Micromedex;6200 South Syracuse Way, Suite 300;;Englewood;CO;United States;80111-4740;(800) 525-9083;;info@mdx.com;https
    ://www.ibm.com/us-en/marketplace/micromedex-with-watson/||3||||MTH_RXN_CD,BD,CD,MTH_RXN_BD|DCSA,DDF,DESI_DESC,DRT,DST,LABELER,MMX_RXO,NDC,UPC|ENG|UTF-8|Y|Y|Micromedex|;;;;Micromedex RED BOOK;;;January 02,
    2024;;;;;;;;;|RxNorm|2024-02-05
C5233828|C1140261|NDDF_2024_01_03|NDDF|FDB MedKnowledge (formerly NDDF Plus)|NDDF|2024|2024-01-03|||2020AA||;;First Databank Customer Support;701 Gateway Blvd, Suite 600;;South San Francisco;CA;United States
    ;94080;800-633-3453;;cs@fdbhealth.com;|;;First Databank Customer Support;701 Gateway Blvd, Suite 600;;South San Francisco;CA;United States;94080;800-633-3453;;cs@fdbhealth.com;|3||||CDC,CDA,CDD,IN
    ,MTH_RXN_CDC,DF|NDC|ENG|UTF-8|Y|Y|FDB MedKnowledge|;;;;FDB MedKnowledge (formerly NDDF Plus);;;January 03, 2024;South San Francisco, CA;First Databank;;;;;;;;|RxNorm|2024-02-05
C5233835|C1140284|RXNORM_20AA_240205F|RXNORM|RxNorm Vocabulary|RXNORM|2024|2024-02-05|||2020AA||RxNorm Customer Service;;U.S. National Library of Medicine;8600 Rockville Pike;;Bethesda;MD;United States;20894
    ;(888) FIND-NLM;;rxnorminfo@nlm.nih.gov;https://www.nlm.nih.gov/research/umls/rxnorm/|RxNorm Customer Service;;U.S. National Library of Medicine;8600 Rockville Pike;;Bethesda;MD;United States;20894;(888)
    FIND-NLM;;rxnorminfo@nlm.nih.gov;https://www.nlm.nih.gov/research/umls/rxnorm/|0||||SCDF,DFG,SBD,PIN,TMSY,BN,SBDC,MIN,SBDG,PSN,SCDC,GPCK,DF,ET,IN,SBDF,SCD,SY,BPCK,SCDG,SCDGP,SCDFP,SBDFP|AMBIGUITY_FLAG,NDC
    ,ORIG_CODE,ORIG_SOURCE,RXN_ACTIVATED,RXN_AI,RXN_AM,RXN_AVAILABLE_STRENGTH,RXN_BN_CARDINALITY,RXN_BOSS_FROM,RXN_BOSS_STRENGTH_DENOM_UNIT,RXN_BOSS_STRENGTH_DENOM_VALUE,RXN_BOSS_STRENGTH_NUM_UNIT
    ,RXN_BOSS_STRENGTH_NUM_VALUE,RXN_HUMAN_DRUG,RXN_IN_EXPRESSED_FLAG,RXN_OBSOLETED,RXN_QUALITATIVE_DISTINCTION,RXN_QUANTITY,RXN_STRENGTH,RXN_VET_DRUG,RXTERM_FORM|ENG|UTF-8|Y|Y|RxNorm work done by the
    National Library of Medicine|;;;;RxNorm;;META2020AA Full Update 2024_02_05;Bethesda, MD;National Library of Medicine;;;;;;;;|RxNorm|2024-02-05
C3531723|C2720507|SNOMEDCT_US_2023_06_30|SNOMEDCT_US|US Edition of SNOMED CT|SNOMEDCT|2023|2023-06-30|||2020AA||National Library Of Medicine;NLM is a Charter Member of SNOMED International on behalf of the U
    .S.;National Library of Medicine;8600 Rockville Pike;;Bethesda;MD;United States;20894;1-888-FIND-NLM (1-888-346-3656);;https://support.nlm.nih.gov/support/create-case/;|National Library Of Medicine;NLM is
    a Charter Member of SNOMED International on behalf of the U.S.;National Library Of Medicine;8600 Rockville Pike;;Bethesda;MD;United States;20894;1-888-FIND-NLM (1-888-346-3656);;https://support.nlm.nih
    .gov/support/create-case/;|9||||FN,SY,PT,PTGB,SYGB||ENG|UTF-8|Y|Y|US Edition of SNOMED CT|;;International Health Terminology Standards Development Organisation;;US Edition of SNOMED CT;;;June 30, 2023
    ;Copenhagen, Denmark;International Health Terminology Standards Development Organisation (IHTSDO);;;;;;;;;|RxNorm|2024-02-05
C5233837|C1140288|VANDF_2023_12_29|VANDF|Veterans Health Administration National Drug File|VANDF|2023|2023-12-29|||2020AA||Michael Lincoln, M.D.;;U.S. Department of Veterans Affairs, Veterans Health
    Administration;;;Washington;DC;United States;;;;michael.lincoln@med.va.gov;http://www.pbm.va.gov/default.aspx|Michael Lincoln, M.D.;;;;;;;;;;;michael.lincoln@med.va.gov;http://www.pbm.va.gov/default
    .aspx|0||||CD,MTH_RXN_CD,AB,IN,PT|DCSA,DDF,DRUG_CLASS_TYPE,EXCLUDE_DI_CHECK,NDC,NDF_TRANSMIT_TO_CMOP,NFI,NF_INACTIVATE,NF_NAME,PARENT_CLASS,SNGL_OR_MULT_SRC_PRD,VAC,VA_CLASS_NAME,VA_DISPENSE_UNIT
    ,VA_GENERIC_NAME,VMO|ENG|UTF-8|Y|Y|National Drug File|;;;;Veterans Health Administration National Drug File;;;December 29, 2023;Washington, DC;U.S. Department of Veterans Affairs;;;;;;;;|RxNorm|2024-02-05
C5233836|C1812643|MTHSPL_2024_01_27|MTHSPL|Metathesaurus FDA Structured Product Labels|MTHSPL|2024|2024-01-27|2006-12-21||2020AA||RxNorm Customer Service;;U.S. National Library of Medicine;8600 Rockville Pike
    ;;Bethesda;MD;United States;20894;(888) FIND-NLM;;rxnorminfo@nlm.nih.gov;https://www.nlm.nih.gov/research/umls/rxnorm/|RxNorm Customer Service;;U.S. National Library of Medicine;8600 Rockville Pike
    ;;Bethesda;MD;United States;20894;(888) FIND-NLM;;rxnorminfo@nlm.nih.gov;https://www.nlm.nih.gov/research/umls/rxnorm/|0||||DP,SU,MTH_RXN_DP|ANADA,ANDA,BLA,COATING,COLOR,COLORTEXT,CONDITIONAL_NADA,DCSA
    ,DIETARY_SUPPLEMENT,DM_SPL_ID,EXEMPT_DEVICE,IMPRINT_CODE,LABELER,LABEL_TYPE,LEGALLY_MARKETED_UNAPPROVED_NEW_ANIMAL_DRUGS_FOR_MINOR_SPECIES,MARKETING_CATEGORY,MARKETING_EFFECTIVE_TIME_HIGH
```

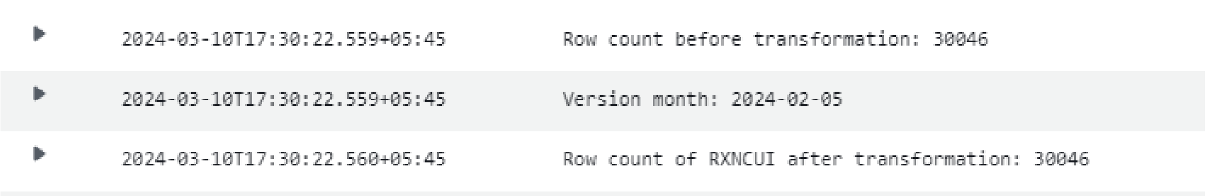
E. Finally validating the row counts as before and after:



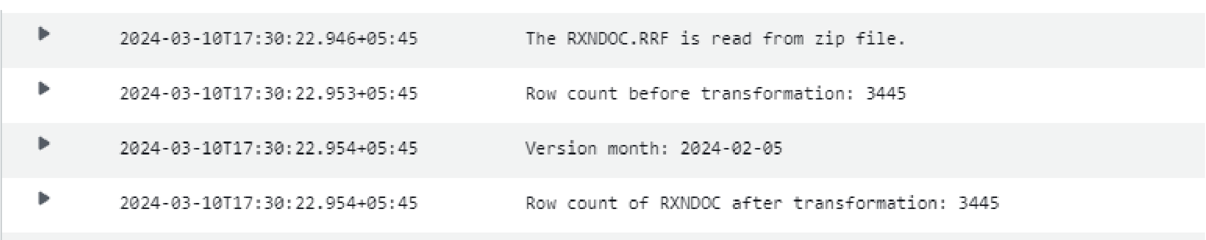
```
▶   2024-03-10T17:30:09.626+05:45      Row count before transformation: 1133065
▶   2024-03-10T17:30:09.627+05:45      Version month: 2024-02-05
▶   2024-03-10T17:30:09.640+05:45      Row count of RXNCONSO after transformation: 1133065
```



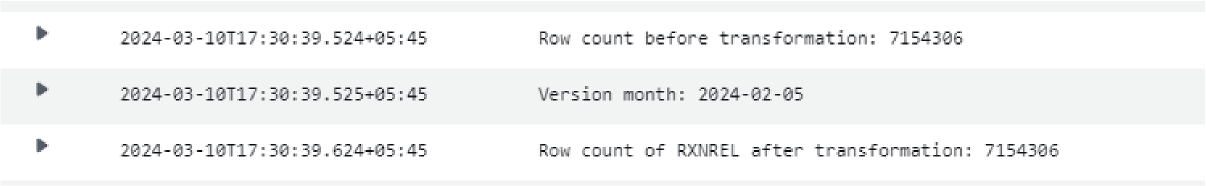
```
▶   2024-03-10T17:30:22.414+05:45      Row count before transformation: 153
▶   2024-03-10T17:30:22.414+05:45      Version month: 2024-02-05
▶   2024-03-10T17:30:22.415+05:45      Row count of RXNCUICHANGES after transformation: 153
```



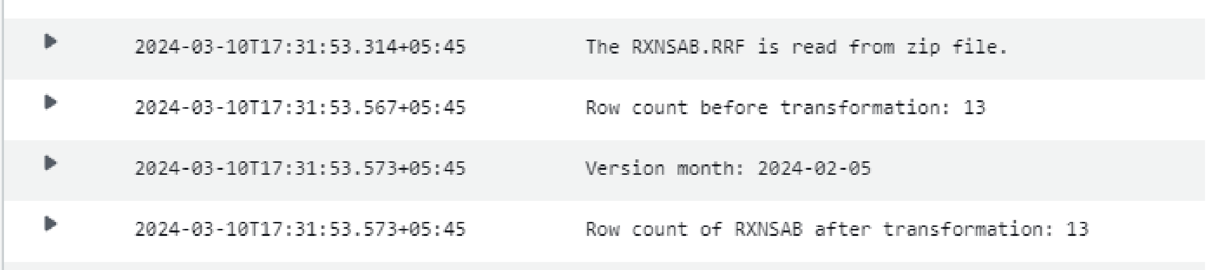
```
▶   2024-03-10T17:30:22.559+05:45      Row count before transformation: 30046
▶   2024-03-10T17:30:22.559+05:45      Version month: 2024-02-05
▶   2024-03-10T17:30:22.560+05:45      Row count of RXNCUI after transformation: 30046
```



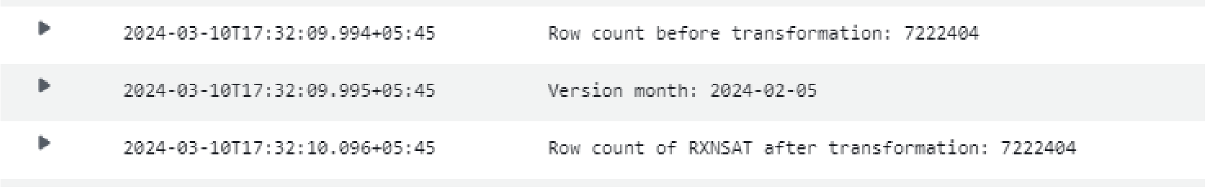
```
▶   2024-03-10T17:30:22.946+05:45      The RXNDOC.RRF is read from zip file.
▶   2024-03-10T17:30:22.953+05:45      Row count before transformation: 3445
▶   2024-03-10T17:30:22.954+05:45      Version month: 2024-02-05
▶   2024-03-10T17:30:22.954+05:45      Row count of RXNDOC after transformation: 3445
```



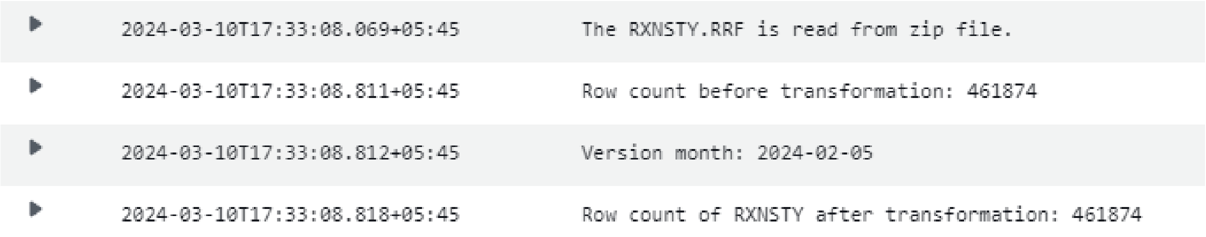
```
▶   2024-03-10T17:30:39.524+05:45      Row count before transformation: 7154306
▶   2024-03-10T17:30:39.525+05:45      Version month: 2024-02-05
▶   2024-03-10T17:30:39.624+05:45      Row count of RXNREL after transformation: 7154306
```



```
▶   2024-03-10T17:31:53.314+05:45      The RXNSAB.RRF is read from zip file.
▶   2024-03-10T17:31:53.567+05:45      Row count before transformation: 13
▶   2024-03-10T17:31:53.573+05:45      Version month: 2024-02-05
▶   2024-03-10T17:31:53.573+05:45      Row count of RXNSAB after transformation: 13
```



```
▶    2024-03-10T17:32:09.994+05:45        Row count before transformation: 7222404
▶    2024-03-10T17:32:09.995+05:45        Version month: 2024-02-05
▶    2024-03-10T17:32:10.096+05:45        Row count of RXNSAT after transformation: 7222404
```



```
▶    2024-03-10T17:33:08.069+05:45        The RXNSTY.RRF is read from zip file.
▶    2024-03-10T17:33:08.811+05:45        Row count before transformation: 461874
▶    2024-03-10T17:33:08.812+05:45        Version month: 2024-02-05
▶    2024-03-10T17:33:08.818+05:45        Row count of RXNSTY after transformation: 461874
```