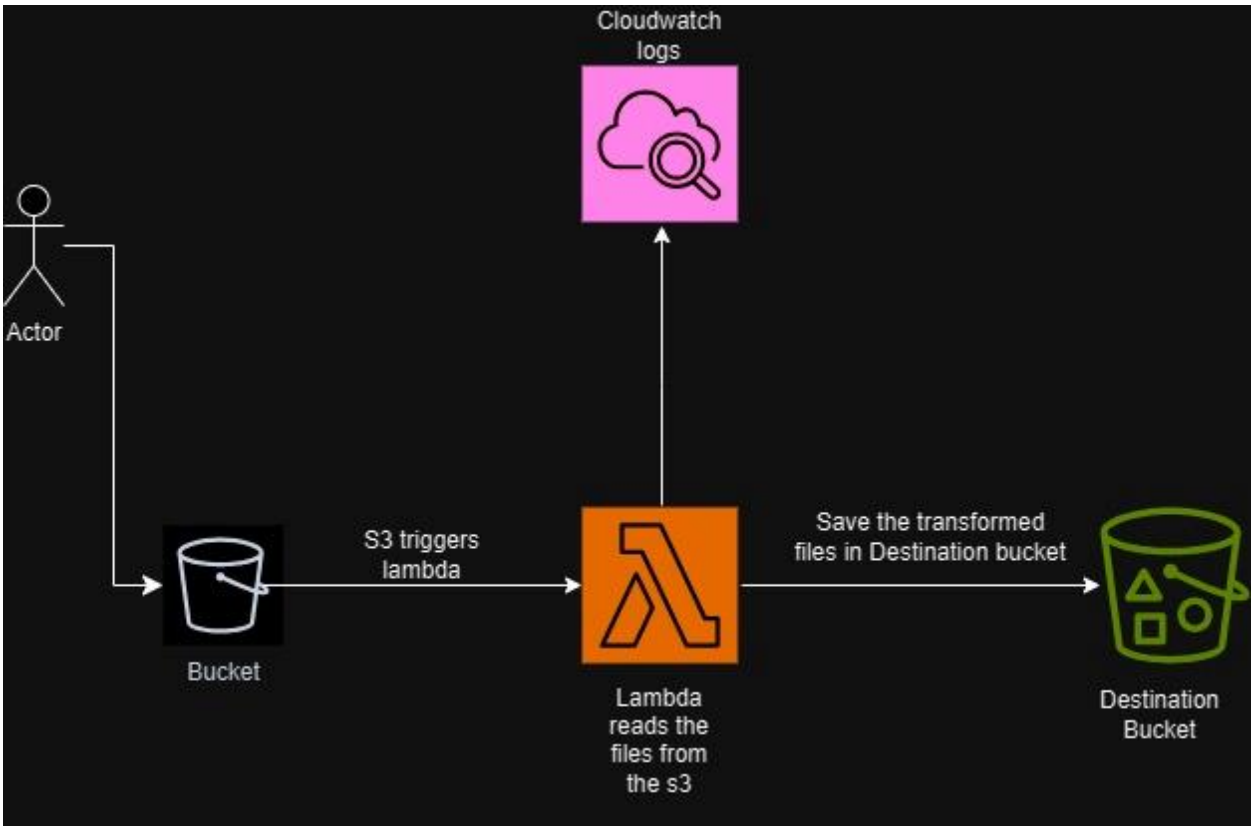
Python and Lambda

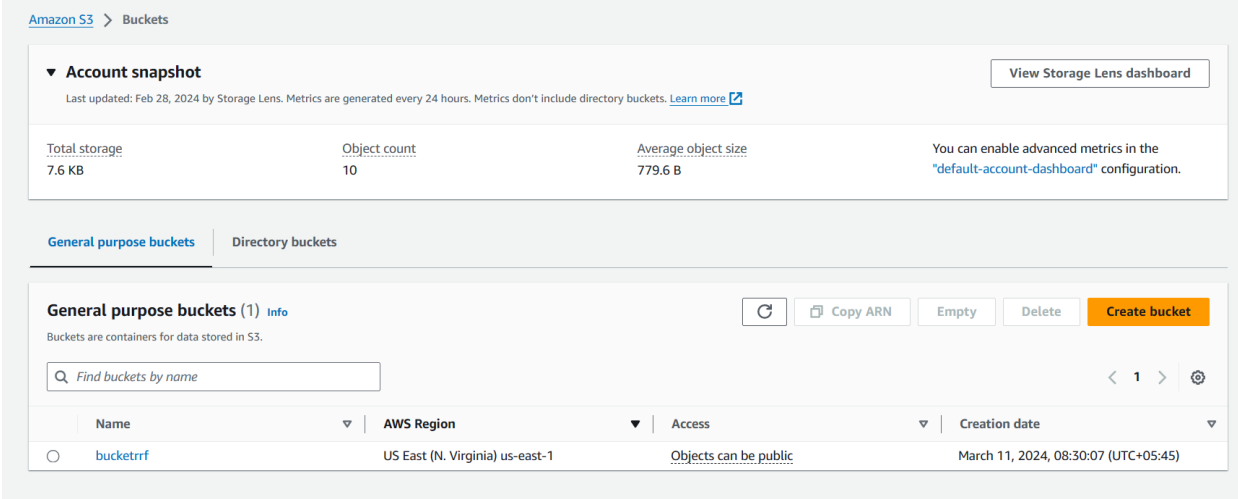
Work with `RXNORM` file,

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. While uploading the file to S3, create CloudWatch log file
5. Add header into each `rff` from RXNORM.xlsx
6. Add `CODE_SET` & `VERSION_MONTH` column with default values `RxNorm` and version month from downloaded filename
7. Convert dates into `YYYY-MM-DD`
8. Save files as `txt` delimited by comma (,)
9. Validate `row_count` between original and converted files

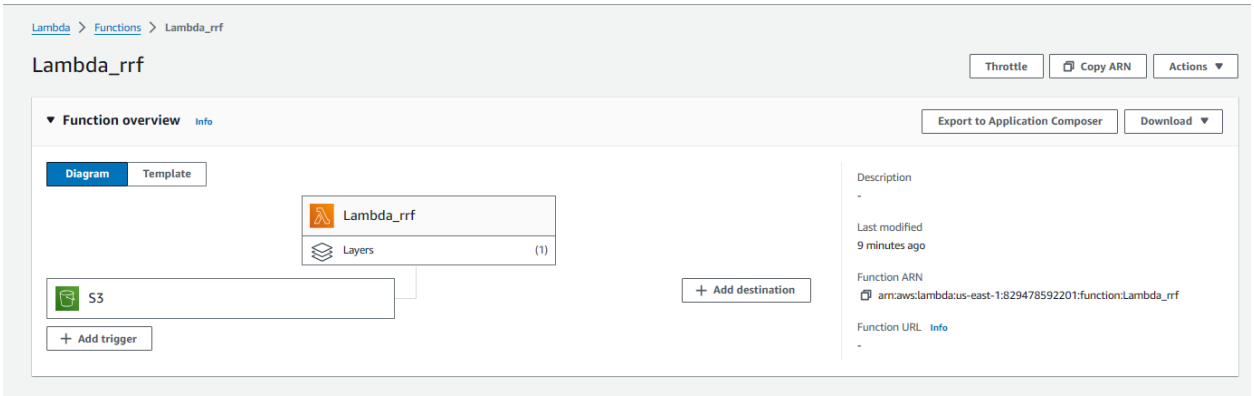
The architecture diagram for the given task is



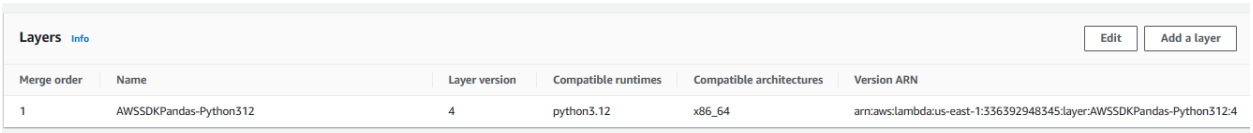
S3 bucket is created as



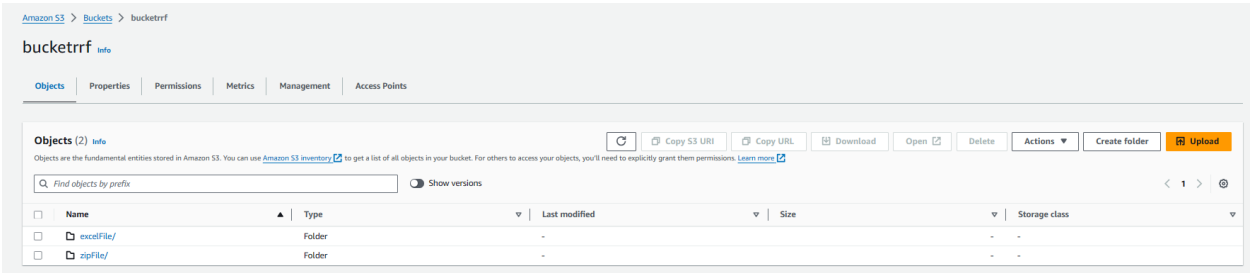
A lambda function is created and s3 bucket which is created before is added



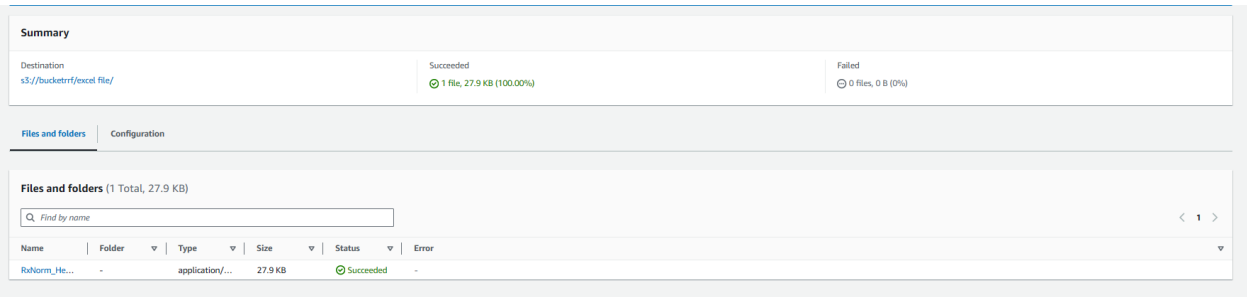
Also, a layer is added in lambda



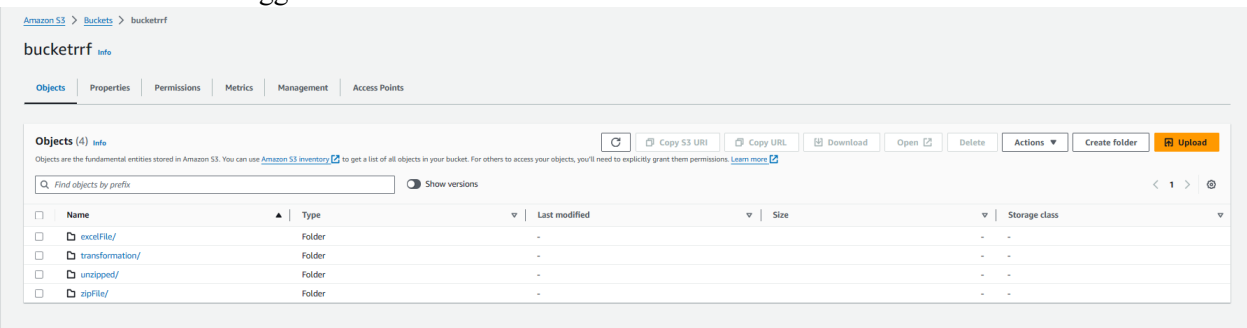
Inside s3 two folder are created for zip file and header file



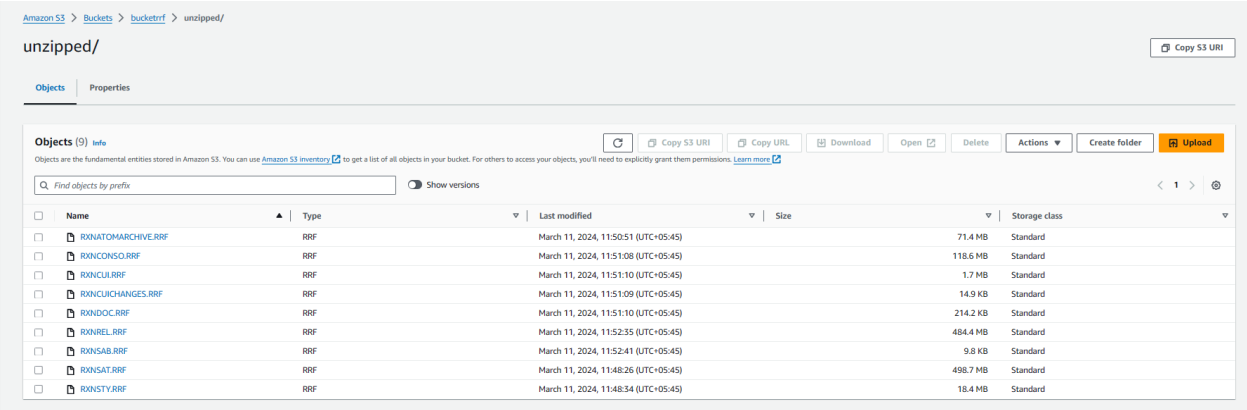
Header file is added



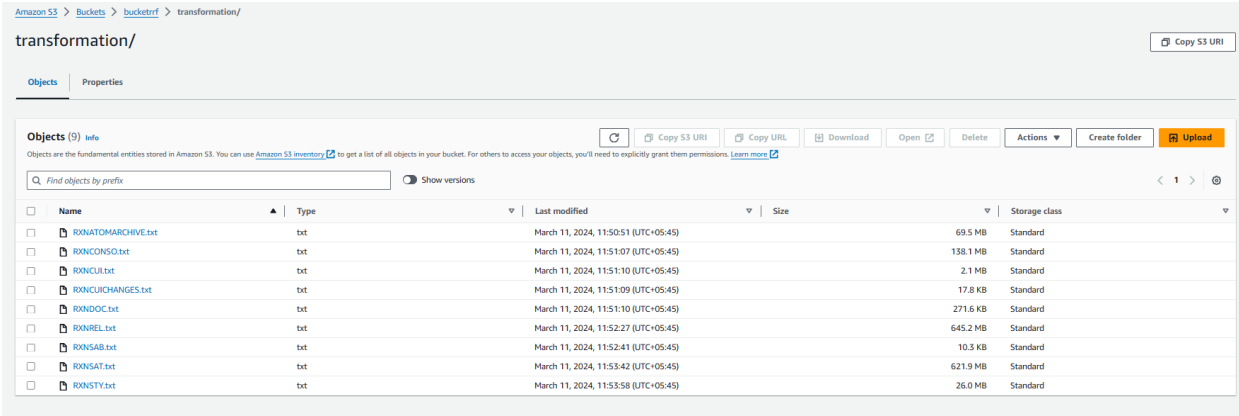
After the lambda is triggered



Inside the unzipped folder all the files which are unzipped are stored as



Inside the transformation folder, the files after transformation are stored as



Following are the codes for the given task

```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 ={}

def read_excel_from_s3(bucket):

    try:
        folder_path = 'excelFile/'
        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}")
```

```python
def code_set_and_version_month(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_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
```

```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

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 format
            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')
```

```python
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

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}")
```

```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()
        zip_filename = os.path.basename(key)
        print(zip_filename)

        # 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]}")
                        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_filename, rrf_df)
                        print(f"Row count of {file_name} after transformation: {rrf_df.shape[0]}")
                        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)
```

```python
                # 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}")

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)

```

The cloud log watch is as

```
2024-03-11T11:48:31.126+05:45    INIT_START Runtime Version: python:3.12.v20 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:82aea00f37a44d6866573 0d559c81352fc95f22c9aa34a0722815a93f08e102b
2024-03-11T11:48:34.381+05:45    START RequestId: 960b4fdc-fe4a-4df1-9535-0f5dc8ba6dc4 Version: $LATEST
2024-03-11T11:48:34.496+05:45    Excel file downloaded to: /tmp/RxNorm_Header.xlsx
2024-03-11T11:48:34.511+05:45    Sheet names: ['RXNCONSO', 'RXNSAT', 'RXNDOC', 'RXNREL', 'RXNSAB', 'RXNSTY', 'RXNATOMARCHIVE', 'RXNCUI', 'RXNCUICHANGES']
2024-03-11T11:48:34.531+05:45    excel_headers dictionary for sheet RXNCONSO: ['RXCUI', 'LAT', 'TS', 'LUI', 'STT', 'SUI', 'ISPREF', 'RXAUI', 'SAUI', 'SCUI', 'SDUI', 'SAB', 'TTY', 'CODE', 'STR', 'SRL', 'SUPPRESS', 'CVF', 'Code set', 'Version Month']
2024-03-11T11:48:37.037+05:45    RxNorm_full_02052024.zip
2024-03-11T11:48:37.038+05:45    The RXNATOMARCHIVE.RRF is read from zip file.
2024-03-11T11:48:39.622+05:45    Row count before transformation: 371368
2024-03-11T11:50:46.223+05:45    Version month: 2024-02-05
2024-03-11T11:50:46.228+05:45    Row count of RXNATOMARCHIVE after transformation: 371368
2024-03-11T11:50:46.235+05:45    RXAUI AUI STR ARCHIVE_TIMESTAMP \
2024-03-11T11:50:46.235+05:45    0 947 A10335796 Mesna 2020-04-27
2024-03-11T11:50:46.235+05:45    1 1424 A10334758 beta-Alanine 2020-04-27
2024-03-11T11:50:46.235+05:45    2 1684 A10334529 4-Aminobenzoic Acid 2020-04-27
2024-03-11T11:50:46.235+05:45    3 2192 A16791816 Eicosapentaenoic Acid 2020-04-27
2024-03-11T11:50:46.235+05:45    4 2265 A10334531 5-Hydroxytryptophan 2020-04-27
2024-03-11T11:50:46.235+05:45    CREATED_TIMESTAMP UPDATED_TIMESTAMP CODE IS_BRAND LAT LAST_RELEASED \
2024-03-11T11:50:46.235+05:45    0 2005-03-10 2020-04-27 44 NaN ENG 2020-04-06
2024-03-11T11:50:46.235+05:45    1 2005-03-10 2020-04-27 61 NaN ENG 2020-04-06
2024-03-11T11:50:46.235+05:45    2 2005-03-10 2020-04-27 74 NaN ENG 2020-04-06
2024-03-11T11:50:46.235+05:45    3 2005-03-10 2020-04-27 90 NaN ENG 2020-04-06
2024-03-11T11:50:46.235+05:45    4 2005-03-10 2020-11-06 94 NaN ENG 2020-04-06
2024-03-11T11:50:46.235+05:45    SAUI VSAB RXCUI SAB TTY MERGED_TO_RXCUI Code Set \
2024-03-11T11:50:46.235+05:45    0 NaN RXNORM_19AB_200406F 44 RXNORM IN 44 RxNorm
2024-03-11T11:50:46.235+05:45    1 NaN RXNORM_19AB_200406F 61 RXNORM IN 61 RxNorm
2024-03-11T11:50:46.235+05:45    2 NaN RXNORM_19AB_200406F 74 RXNORM IN 74 RxNorm
2024-03-11T11:50:46.235+05:45    3 NaN RXNORM_19AB_200406F 90 RXNORM PIN 90 RxNorm
2024-03-11T11:50:46.235+05:45    4 NaN RXNORM_19AB_200406F 94 RXNORM IN 94 RxNorm
```

```
2024-03-11T11:50:46.235+05:45	Version Month
2024-03-11T11:50:46.235+05:45	0 2024-02-05
2024-03-11T11:50:46.235+05:45	1 2024-02-05
2024-03-11T11:50:46.235+05:45	2 2024-02-05
2024-03-11T11:50:46.235+05:45	3 2024-02-05
2024-03-11T11:50:46.235+05:45	4 2024-02-05
2024-03-11T11:50:50.688+05:45	Transformed data saved to: s3://bucketrrf/transformation/RXNATOMARCHIVE.txt
2024-03-11T11:50:51.452+05:45	Unzipped file saved to: s3://bucketrrf/unzipped/RXNATOMARCHIVE.RRF
2024-03-11T11:50:51.452+05:45	The RXNCONSO.RRF is read from zip file.
2024-03-11T11:50:55.155+05:45	/var/task/lambda_function.py:200: DtypeWarning: Columns (9) have mixed types. Specify dtype option on import or set low_memory=False.
2024-03-11T11:50:55.155+05:45	rrf_df = pd.read_csv(file_content_io, delimiter='|', header=None)
2024-03-11T11:50:55.945+05:45	Row count before transformation: 1133065
2024-03-11T11:50:55.946+05:45	Version month: 2024-02-05
2024-03-11T11:50:55.965+05:45	Row count of RXNCONSO after transformation: 1133065
2024-03-11T11:50:55.970+05:45	RXCUI LAT TS LUI STT SUI ISPREF RXAUI SAUI SCUI SDUI \
2024-03-11T11:50:55.970+05:45	0 3 ENG NaN NaN NaN NaN NaN 8717795 NaN 58488005 NaN
2024-03-11T11:50:55.970+05:45	1 3 ENG NaN NaN NaN NaN NaN 8717796 NaN 58488005 NaN
2024-03-11T11:50:55.970+05:45	2 3 ENG NaN NaN NaN NaN NaN 8717808 NaN 58488005 NaN
2024-03-11T11:50:55.970+05:45	3 3 ENG NaN NaN NaN NaN NaN 8718164 NaN 58488005 NaN
2024-03-11T11:50:55.970+05:45	4 19 ENG NaN NaN NaN NaN NaN 10794494 NaN 112116001 NaN
2024-03-11T11:50:55.970+05:45	SAB TTY CODE STR \
2024-03-11T11:50:55.970+05:45	0 SNOMEDCT_US PT 58488005 1,4-alpha-Glucan branching enzyme
2024-03-11T11:50:55.970+05:45	1 SNOMEDCT_US FN 58488005 1,4-alpha-Glucan branching enzyme (substance)
2024-03-11T11:50:55.970+05:45	2 SNOMEDCT_US SY 58488005 Amylo-(1,4,6)-transglycosylase
2024-03-11T11:50:55.970+05:45	3 SNOMEDCT_US SY 58488005 Branching enzyme
2024-03-11T11:50:55.970+05:45	4 SNOMEDCT_US SY 112116001 17-hydrocorticosteroid
2024-03-11T11:50:55.970+05:45	SRL SUPPRESS CVF Code Set Version Month
```

```
2024-03-11T11:52:40.485+05:45	2 ;;;;FDB MedKnowledge (formerly NDDF Plus);;;Ja... RxNorm 2024-02-05
2024-03-11T11:52:40.485+05:45	3 ;;;;RxNorm;;;META20200AA Full Update 2024_02_05... RxNorm 2024-02-05
2024-03-11T11:52:40.485+05:45	4 ;;International Health Terminology Standards D... RxNorm 2024-02-05
2024-03-11T11:52:40.565+05:45	Transformed data saved to: s3://bucketrrf/transformation/RXNSAB.txt
2024-03-11T11:52:40.627+05:45	Unzipped file saved to: s3://bucketrrf/unzipped/RXNSAB.RRF
2024-03-11T11:52:40.627+05:45	The RXNSAT.RRF is read from zip file.
2024-03-11T11:52:53.758+05:45	/var/task/lambda_function.py:200: DtypeWarning: Columns (6) have mixed types. Specify dtype option on import or set low_memory=False.
2024-03-11T11:52:53.758+05:45	rrf_df = pd.read_csv(file_content_io, delimiter='|', header=None)
2024-03-11T11:52:57.186+05:45	Row count before transformation: 7222404
2024-03-11T11:52:57.186+05:45	Version month: 2024-02-05
2024-03-11T11:52:57.276+05:45	Row count of RXNSAT after transformation: 7222404
2024-03-11T11:52:57.280+05:45	RXCUI LUI SUI RXAUI STYPE CODE ATUI SATUI ATN \
2024-03-11T11:52:57.280+05:45	0 38 NaN NaN 829 AUI 38 NaN NaN RXN_BN_CARDINALITY
2024-03-11T11:52:57.280+05:45	1 44 NaN NaN 955 AUI d01411 NaN NaN DPC
2024-03-11T11:52:57.280+05:45	2 44 NaN NaN 2798745 AUI NR7O1405Q9 NaN NaN SPL_SET_ID
2024-03-11T11:52:57.280+05:45	3 44 NaN NaN 2982613 AUI NR7O1405Q9 NaN NaN SPL_SET_ID
2024-03-11T11:52:57.280+05:45	4 44 NaN NaN 2982613 AUI NR7O1405Q9 NaN NaN SPL_SET_ID
2024-03-11T11:52:57.280+05:45	SAB ATV SUPPRESS CVF Code Set \
2024-03-11T11:52:57.280+05:45	0 RXNORM single N 4096.0 RxNorm
2024-03-11T11:52:57.280+05:45	1 MMSL N N NaN RxNorm
2024-03-11T11:52:57.280+05:45	2 MTHSPL 2a2a526f-636b-7920-4261-6c626f612a2a N 4096.0 RxNorm
2024-03-11T11:52:57.280+05:45	3 MTHSPL 2e8eebc3-ea75-4c57-bfe1-2c5dc2c37806 N 4096.0 RxNorm
2024-03-11T11:52:57.280+05:45	4 MTHSPL 49fdfbc3-69f8-4706-9d11-bf60e4e7811b N 4096.0 RxNorm
2024-03-11T11:52:57.280+05:45	Version Month
2024-03-11T11:52:57.280+05:45	0 2024-02-05
2024-03-11T11:52:57.280+05:45	1 2024-02-05
2024-03-11T11:52:57.280+05:45	2 2024-02-05
2024-03-11T11:52:57.280+05:45	3 2024-02-05
2024-03-11T11:52:57.280+05:45	4 2024-02-05
```

Back to top

All the log are uploaded in github.