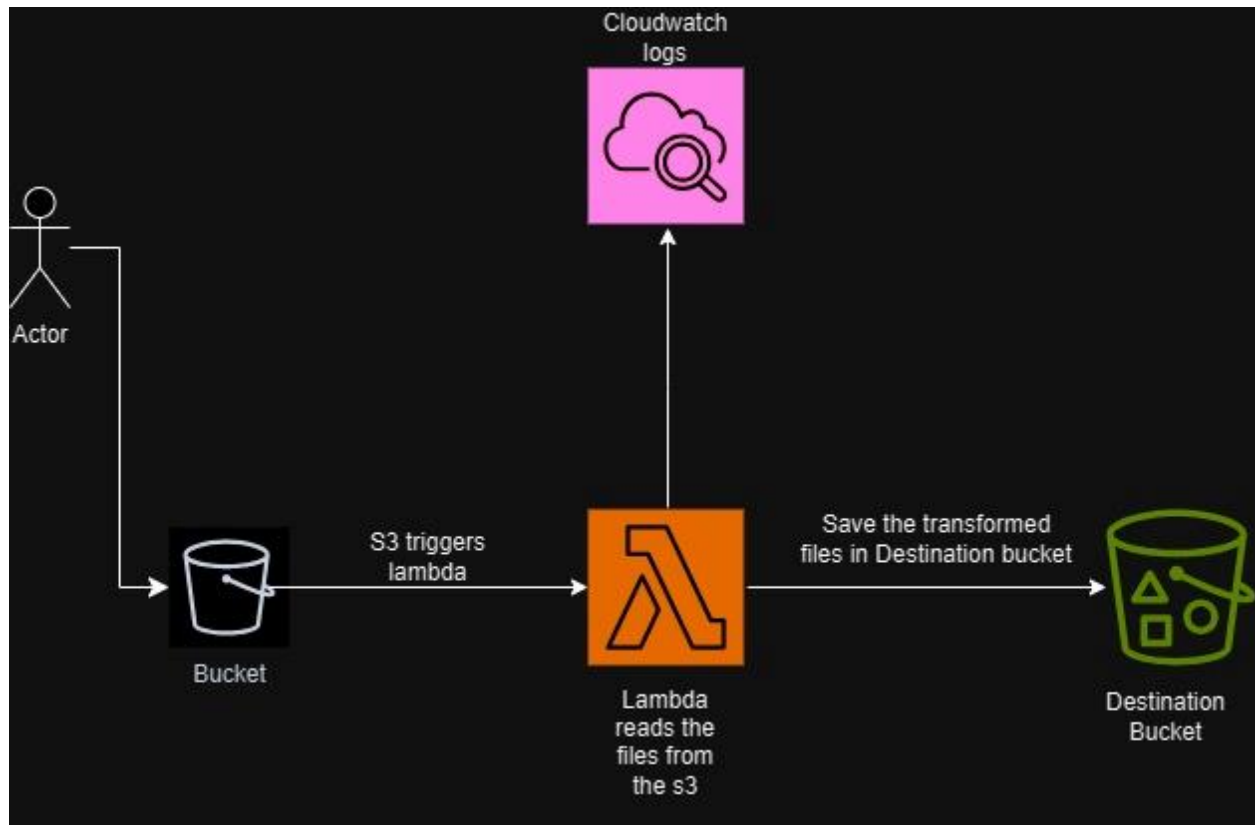


Python and Lambda

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

Amazon S3 > Buckets

Account snapshot View Storage Lens dashboard

Last updated: Feb 28, 2024 by Storage Lens. Metrics are generated every 24 hours. Metrics don't include directory buckets. [Learn more](#)

Total storage	Object count	Average object size	You can enable advanced metrics in the "default-account-dashboard" configuration.
7.6 KB	10	779.6 B	

General purpose buckets | Directory buckets

General purpose buckets (1) [Info](#)

Buckets are containers for data stored in S3.

< 1 > ⚙

Name	AWS Region	Access	Creation date
<input type="radio"/> bucketrrf	US East (N. Virginia) us-east-1	Objects can be public	March 11, 2024, 08:30:07 (UTC+05:45)

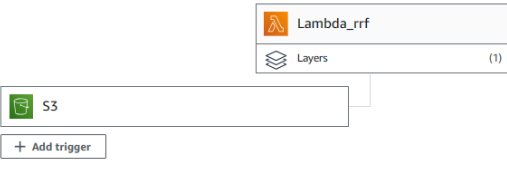
A lambda function is created and s3 bucket which is created before is added

Lambda > Functions > Lambda_rrf

Lambda_rrf Throttle Copy ARN Actions

Function overview [Info](#) Export to Application Composer Download

Diagram Template



Description

-

Last modified
9 minutes ago

Function ARN
[arn:aws:lambda:us-east-1:829478592201:function:Lambda_rrf](#)

Function URL [Info](#)

-

Also, a layer is added in lambda

Layers [Info](#) Edit Add a layer

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	AWSSDKPandas-Python312	4	python3.12	x86_64	arn:aws:lambda:us-east-1:336392948345:layer:AWSSDKPandas-Python312:4

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

Amazon S3 > Buckets > bucketrrf

bucketrrf [Info](#)

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

Objects (2) [Info](#) Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Show versions < 1 > ⚙

Name	Type	Last modified	Size	Storage class
<input type="checkbox"/> excelFile/	Folder	-	-	-
<input type="checkbox"/> zipFile/	Folder	-	-	-

Header file is added

Summary

Destination

s3://bucketrrf/excel File/

Succeeded

1 file, 27.9 KB (100.00%)

Failed

0 files, 0 B (0%)

Files and folders

Configuration

Files and folders (1 Total, 27.9 KB)

Find by name

< 1 >

Name	Folder	Type	Size	Status	Error
ReNorm_He...	-	application/...	27.9 KB	Succeeded	-

After the lambda is triggered

Amazon S3 > Buckets > bucketrrf

bucketrrf

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (4)

Find objects by prefix

Show versions

< 1 >

Name	Type	Last modified	Size	Storage class
excelFile/	Folder	-	-	-
transformation/	Folder	-	-	-
unzipped/	Folder	-	-	-
zipFile/	Folder	-	-	-

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

Amazon S3 > Buckets > bucketrrf > unzipped/

unzipped/

Objects

Properties

Objects (9)

Find objects by prefix

Show versions

< 1 >

Name	Type	Last modified	Size	Storage class
R0NAT0MARCHIVE.RRF	RRF	March 11, 2024, 11:50:51 (UTC+05:45)	71.4 MB	Standard
R0NCONSO.RRF	RRF	March 11, 2024, 11:51:08 (UTC+05:45)	118.6 MB	Standard
R0NCULR.RRF	RRF	March 11, 2024, 11:51:10 (UTC+05:45)	1.7 MB	Standard
R0NCLICHANGES.RRF	RRF	March 11, 2024, 11:51:09 (UTC+05:45)	14.9 KB	Standard
R0NDOC.RRF	RRF	March 11, 2024, 11:51:10 (UTC+05:45)	214.2 KB	Standard
R0NREL.RRF	RRF	March 11, 2024, 11:52:35 (UTC+05:45)	484.4 MB	Standard
R0NSAB.RRF	RRF	March 11, 2024, 11:52:41 (UTC+05:45)	9.8 KB	Standard
R0NSAT.RRF	RRF	March 11, 2024, 11:48:26 (UTC+05:45)	498.7 MB	Standard
R0NSTY.RRF	RRF	March 11, 2024, 11:48:34 (UTC+05:45)	18.4 MB	Standard

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

Amazon S3 > Buckets > buckettrf > transformation/

transformation/ Copy S3 URI

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](#)

☐ Show versions < 1 > @

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	RXNATOMARCHIVE.txt	txt	March 11, 2024, 11:50:51 (UTC+05:45)	69.5 MB	Standard
<input type="checkbox"/>	RXNCONSO.txt	txt	March 11, 2024, 11:51:07 (UTC+05:45)	138.1 MB	Standard
<input type="checkbox"/>	RXNCULT.txt	txt	March 11, 2024, 11:51:10 (UTC+05:45)	2.1 MB	Standard
<input type="checkbox"/>	RXNCUICHANGES.txt	txt	March 11, 2024, 11:51:09 (UTC+05:45)	17.8 KB	Standard
<input type="checkbox"/>	RXNDOC.txt	txt	March 11, 2024, 11:51:10 (UTC+05:45)	271.6 KB	Standard
<input type="checkbox"/>	RXNREL.txt	txt	March 11, 2024, 11:52:27 (UTC+05:45)	645.2 MB	Standard
<input type="checkbox"/>	RXNSAB.txt	txt	March 11, 2024, 11:52:41 (UTC+05:45)	10.3 KB	Standard
<input type="checkbox"/>	RXNSAT.txt	txt	March 11, 2024, 11:53:42 (UTC+05:45)	621.9 MB	Standard
<input type="checkbox"/>	RXNSTY.txt	txt	March 11, 2024, 11:53:58 (UTC+05:45)	26.0 MB	Standard

Following are the codes for the given task

```
1 import boto3
2 import zipfile
3 import io
4 import os
5 import pandas as pd
6 import json
7 import openpyxl
8 from io import BytesIO
9
10
11 s3 = boto3.client('s3')
12 excel_headers = {}
13
14 def read_excel_from_s3(bucket):
15
16     try:
17         folder_path = 'excelFile/'
18         excel_file_name = 'RxNorm_Header.xlsx'
19         key = folder_path + excel_file_name
20
21         # Download the Excel file to the /tmp directory
22         local_excel_file = '/tmp/RxNorm_Header.xlsx'
23         s3.download_file(bucket, key, local_excel_file)
24
25         # Check if the Excel file exists
26         if os.path.exists(local_excel_file):
27             print(f"Excel file downloaded to: {local_excel_file}")
28
29         # Read the Excel file into an ExcelFile object
30         excel_file = pd.ExcelFile(local_excel_file)
31
32         # Get the sheet names
33         sheet_names = excel_file.sheet_names
34         print("Sheet names:", sheet_names)
35         for sheets in sheet_names:
36
37             # Read the data from the sheet into a DataFrame
38             sheets_data = excel_file.parse(sheets, header=None)
39             headers_data = sheets_data.iloc[:, 0].tolist()
40             excel_headers[sheets] = headers_data
41
42         print(f'excel_headers dictionary for sheet {sheet_names[0]}: {excel_headers[sheet_names[0]]}')
43
44     except Exception as e:
45         print(f"Error occurred: {e}")
46
```

```

48 def code_set_and_version_month(zip_filename, rrf_df):
49
50     try:
51         # Extract version month from the filename
52         version_month = os.path.splitext(zip_filename)[0].split('_')[-1]
53
54         # Convert version month to a more readable format
55         version_month = pd.to_datetime(version_month, format='%m%d%Y').strftime('%Y-%m-%d')
56
57         print(f"Version month: {version_month}")
58
59         # Add 'Code Set' and 'Version Month' columns to the DataFrame
60         rrf_df['Code Set'] = 'RxNorm'
61         rrf_df['Version Month'] = version_month
62
63     except Exception as e:
64         print(f"Error occurred while extracting version month: {e}")
65     return rrf_df
66
67 def apply_header_to_rrf(file_name, rrf_df):
68
69     # Check if the corresponding Excel sheet exists
70     if file_name in excel_headers:
71
72         # Get the headers from the Excel sheet
73         excel_headers_list = excel_headers[file_name]
74         excel_headers_list = [header for header in excel_headers_list if header != 'SVER']
75
76         # Take names from the excel header list up to the length of the split DataFrame
77         excel_headers_list = excel_headers_list[:len(rrf_df.columns)]
78
79         # Set the correct header for the DataFrame
80         rrf_df.columns = excel_headers_list
81
82     return rrf_df

```

```

84 def convert_date_format(value):
85     try:
86         # Try to parse the value into datetime format
87         parsed_date = pd.to_datetime(value, format='%Y_%m_%d').date()
88
89         # Extract only the date part
90         return parsed_date.strftime('%Y-%m-%d')
91
92     except ValueError:
93         if value == '2020':
94             return '2020-01-01'
95         elif value == '5.0_2024_01_04':
96
97             # Remove the float value and parse the remaining string
98             return convert_date_format('2024_01_04')
99         elif value == '2020AA':
100             return '2024-01-02'
101
102         elif value == '20AA_240205F':
103             return '2024-02-05'
104         else:
105             return value
106
107 def update_nato_date(value):
108
109     try:
110         # Attempt to parse the value using the first date format
111         parsed_date = pd.to_datetime(value, format='%m/%d/%Y %I:%M:%S %p').date()
112
113     except ValueError:
114         try:
115             # If the first format fails, attempt to parse using the second date format
116             parsed_date = pd.to_datetime(value, format='%d-%b-%y').date()
117
118         except ValueError:
119             # If both formats fail, return None or handle the error appropriately
120             return None # Or handle the error appropriately
121
122         # Check if the parsed_date is NaT
123     if pd.isnull(parsed_date):
124         return '0000-00-00' # Replace NaT with '0000-00-00'
125
126     else:
127         # Extract only the date part and return it in the desired format
128         return parsed_date.strftime('%Y-%m-%d')

```

```

130 def process_date_columns(file_name, rrf_df):
131     date_columns = ['VSTART', 'VEND', 'CREATED_TIMESTAMP', 'UPDATED_TIMESTAMP', 'LAST_RELEASED']
132     for column in date_columns:
133         if column in rrf_df.columns:
134             if file_name == 'RXNSAB':
135
136                 # Apply the conversion function to each value in the column
137                 rrf_df[column] = rrf_df[column].apply(convert_date_format)
138
139                 # Extract year from the VSTART column after date conversion and save it directly as a string
140                 rrf_df['SVER'] = pd.to_datetime(rrf_df['VSTART'], format='%Y-%m-%d').dt.year.astype(str)
141                 # Reorder the columns to place 'SVER' before 'VSTART'
142                 # Reorder columns
143                 sver_index = rrf_df.columns.get_loc('SVER')
144                 vstart_index = rrf_df.columns.get_loc('VSTART')
145                 sf_index = rrf_df.columns.get_loc('SF')
146
147                 # Remove 'SVER' from its original position
148                 column_sver = rrf_df.pop('SVER')
149
150                 # Insert 'SVER' after 'SF', before 'VSTART'
151                 if sver_index < vstart_index:
152                     rrf_df.insert(vstart_index - 1, 'SVER', column_sver)
153
154                 elif sver_index > vstart_index:
155                     rrf_df.insert(vstart_index, 'SVER', column_sver)
156
157             if file_name == 'RXNATOMARCHIVE':
158                 rrf_df[column] = rrf_df[column].apply(update_nato_date)
159
160     return rrf_df
161
162 def save_as_txt_file(rrf_df, file_name, bucket_name):
163
164     # Construct the filename for the output text file
165     transformation_folder = 'transformation/'
166
167     # Convert DataFrame to CSV format in memory
168     csv_buffer = io.StringIO()
169
170     rrf_df.to_csv(csv_buffer, sep='|', index=False)
171
172     # Upload the CSV buffer to S3
173     s3_key = transformation_folder + file_name + '.txt'
174     s3.put_object(Bucket=bucket_name, Key=s3_key, Body=csv_buffer.getvalue())
175     print(f"Transformed data saved to: s3://{bucket_name}/{s3_key}")
176
177

```

```

178 def read_and_relocate_rrf_files(s3, bucket, key):
179     try:
180         zip_response = s3.get_object(Bucket=bucket, Key=key)
181         zip_data = zip_response['Body'].read()
182         zip_filename = os.path.basename(key)
183         print(zip_filename)
184
185         # Wrap the zip data in a BytesIO object
186         zip_file = BytesIO(zip_data)
187         file_path = 'rrf'
188         unzipped_folder = 'unzipped/'
189
190         with zipfile.ZipFile(zip_file, 'r') as zip_ref:
191             for file_info in zip_ref.infolist():
192                 if file_info.filename.startswith(file_path) and not file_info.filename.endswith('/'):
193                     filename = os.path.basename(file_info.filename)
194                     print(f"The {filename} is read from zip file.")
195                     with zip_ref.open(file_info) as source_file:
196                         file_content = source_file.read().decode('utf-8')
197                         if file_content.endswith('|'):
198                             file_content = file_content[:-1]
199                         file_content_io = io.StringIO(file_content)
200                         rrf_df = pd.read_csv(file_content_io, delimiter='|', header=None)
201                         rrf_df = rrf_df.iloc[:, :-1]
202                         print(f"Row count before transformation: {rrf_df.shape[0]}")
203                         file_name = os.path.splitext(filename)[0]
204                         apply_header_to_rrf(file_name, rrf_df)
205                         rrf_df = process_date_columns(file_name, rrf_df)
206                         code_set_and_version_month(zip_filename, rrf_df)
207                         print(f"Row count of {file_name} after transformation: {rrf_df.shape[0]}")
208                         pd.set_option('display.max_columns', None)
209                         print(rrf_df.head(5))
210                         # Save the transformed DataFrame to a text file
211                         save_as_txt_file(rrf_df, file_name, bucket)

```

```
214         # Upload the unzipped file to the 'unzipped' folder
215         unzipped_key = unzipped_folder + filename
216         s3.put_object(Bucket=bucket, Key=unzipped_key, Body=file_content)
217         print(f"Unzipped file saved to: s3://{bucket}/{unzipped_key}")
218
219     except Exception as e:
220
221         print(f"Error occurred: {e}")
222
223 def lambda_handler(event, context):
224     bucket = event['Records'][0]['s3']['bucket']['name']
225     key = event['Records'][0]['s3']['object']['key']
226
227     # This is the function that relocate the rrf files from zip file
228     read_excel_from_s3(bucket)
229     read_and_relocate_rrf_files(s3,bucket,key)
230
```

The cloud log watch is as

▶	2024-03-11T11:48:31.128+05:45	INIT_START Runtime Version: python:3.11.v20 Runtime Version ARN: arn:aws:lambda:us-east-1:runtime:82ae0bf37d4d6d8865738d559c81352fc95f22c9a34a0/22815a93f0be382b
▶	2024-03-11T11:48:34.381+05:45	START RequestId: 968b4fd4-fda-4df1-9535-0f5dc8baddc4 Version: \$LATEST
▶	2024-03-11T11:48:34.496+05:45	Excel file downloaded to: /tmp/RxNormHeader.xlsx
▶	2024-03-11T11:48:34.511+05:45	Sheet names: ['RXNCONGO', 'RXNSAT', 'RXNDOC', 'RXNREL', 'RXNLAB', 'RXNSTV', 'RXNATOMARCHIVE', 'RXNCUI', 'RXNCUICHANGES']
▶	2024-03-11T11:48:34.533+05:45	excel_headers dictionary for sheet RXNCONGO: ['RXNCUI', '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 Mesa 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 A10792816 Eicosapentaenoic Acid 2020-04-27
▶	2024-03-11T11:50:46.235+05:45	4 2265 A10334511 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_1040_200406F 44 RXNORM IN 44 RxNorm
▶	2024-03-11T11:50:46.235+05:45	1 NaN RXNORM_1040_200406F 61 RXNORM IN 61 RxNorm
▶	2024-03-11T11:50:46.235+05:45	2 NaN RXNORM_1040_200406F 74 RXNORM IN 74 RxNorm
▶	2024-03-11T11:50:46.235+05:45	3 NaN RXNORM_1040_200406F 90 RXNORM IN 90 RxNorm
▶	2024-03-11T11:50:46.235+05:45	4 NaN RXNORM_1040_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/RXNATOWARCHIVE.txt
▶	2024-03-11T11:50:51.452+05:45	Unzipped file saved to: s3://bucketrrf/unzipped/RXNATOWARCHIVE.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: 1113065
▶	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: 1113065
▶	2024-03-11T11:50:55.970+05:45	RXCUI LAT TS LUT STT SUZ ISPREF RXAUJ 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 10794404 NaN 112116001 NaN
▶	2024-03-11T11:50:55.970+05:45	SAB TTY CODE STR \
▶	2024-03-11T11:50:55.970+05:45	0 SMOEDCT_US PT 58488005 1,4-alpha-Glucan branching enzyme
▶	2024-03-11T11:50:55.970+05:45	1 SMOEDCT_US FN 58488005 1,4-alpha-Glucan branching enzyme (substance)
▶	2024-03-11T11:50:55.970+05:45	2 SMOEDCT_US SY 58488005 Anylo-(1,4,6)-transglycosylase
▶	2024-03-11T11:50:55.970+05:45	3 SMOEDCT_US SY 58488005 Branching enzyme
▶	2024-03-11T11:50:55.970+05:45	4 SMOEDCT_US SY 112116001 17-hydrocorticosteroid
▶	2024-03-11T11:50:55.970+05:45	SRL SUPPRESS CWF Code Set Version Month

▶	2024-03-11T11:52:40.485+05:45	2 ;;;FDB MedKnowledge (formerly NDOF Plus);;;3a... RxNorm 2024-02-05
▶	2024-03-11T11:52:40.485+05:45	3 ;;;RxNorm;;METAD200AA 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 SUZ RXAUJ STYPE CODE ATUI SATUI ATN \
▶	2024-03-11T11:52:57.280+05:45	0 3R NaN NaN 829 AUJ 3R NaN NaN RXN_BN_CARDINALITY
▶	2024-03-11T11:52:57.280+05:45	1 44 NaN NaN 955 AUJ 001411 NaN NaN DPC
▶	2024-03-11T11:52:57.280+05:45	2 44 NaN NaN 2798745 AUJ NR701485Q9 NaN NaN SPL_SET_ID
▶	2024-03-11T11:52:57.280+05:45	3 44 NaN NaN 2982613 AUJ NR701485Q9 NaN NaN SPL_SET_ID
▶	2024-03-11T11:52:57.280+05:45	4 44 NaN NaN 2982613 AUJ NR701485Q9 NaN NaN SPL_SET_ID
▶	2024-03-11T11:52:57.280+05:45	SAB ATV SUPPRESS CWF Code Set \
▶	2024-03-11T11:52:57.280+05:45	0 RXNORM single N 4896.0 RxNorm
▶	2024-03-11T11:52:57.280+05:45	1 MHSL N N NaN RxNorm
▶	2024-03-11T11:52:57.280+05:45	2 MTHSPL 2a2a526f-6360-7920-4261-6c626f612a2a N 4896.0 RxNorm
▶	2024-03-11T11:52:57.280+05:45	3 MTHSPL 2a8eebc3-ea75-4c57-bfe1-2c5dc2c37806 N 4896.0 RxNorm
▶	2024-03-11T11:52:57.280+05:45	4 MTHSPL 49f0bc3-60f0-4706-0d11-bf68e4e7811b N 4896.0 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.

