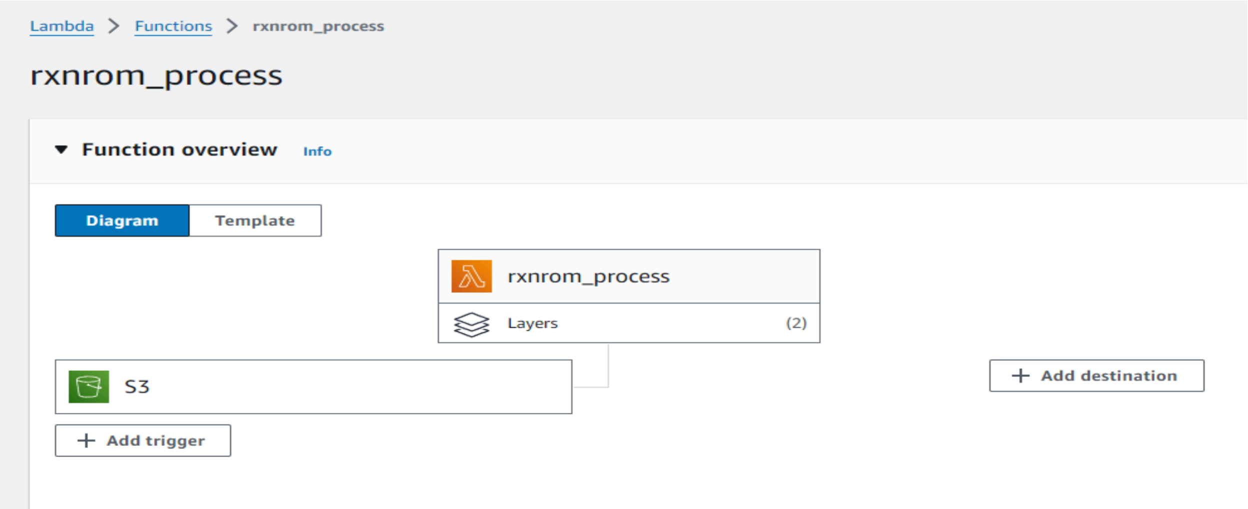
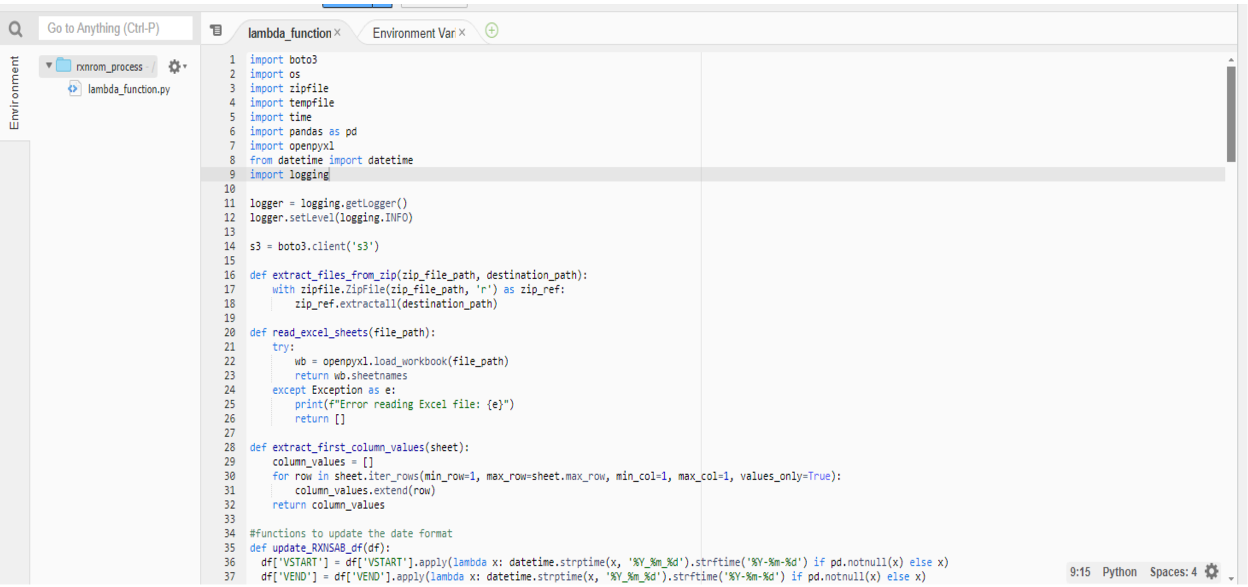
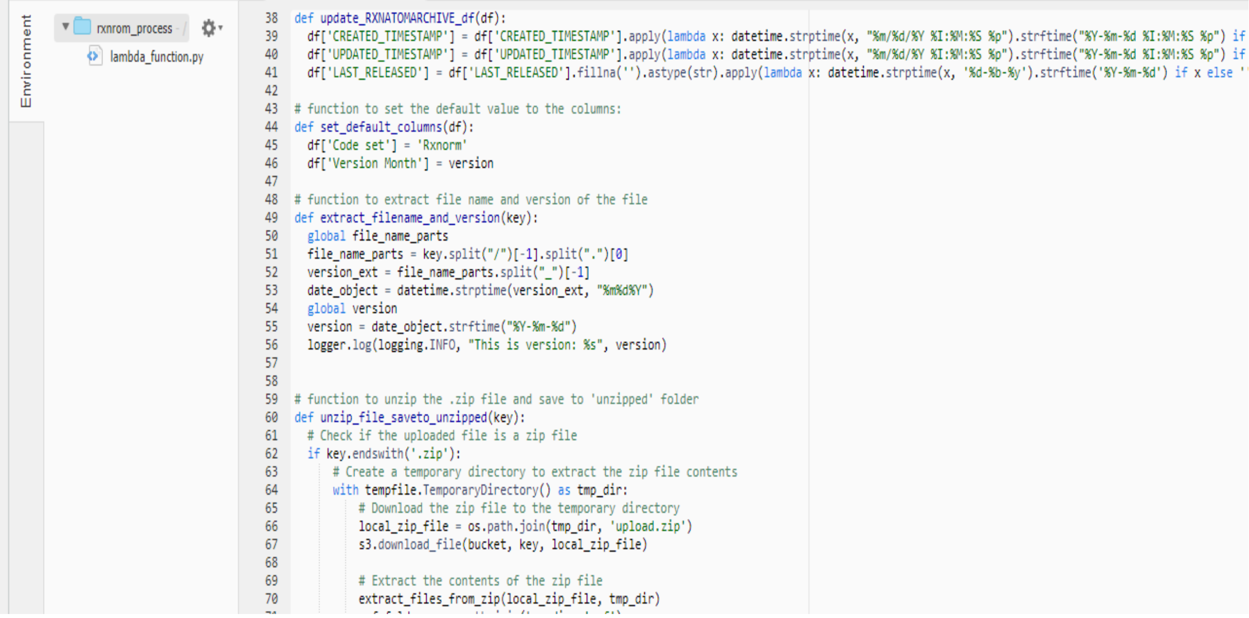
Lambda Structure:

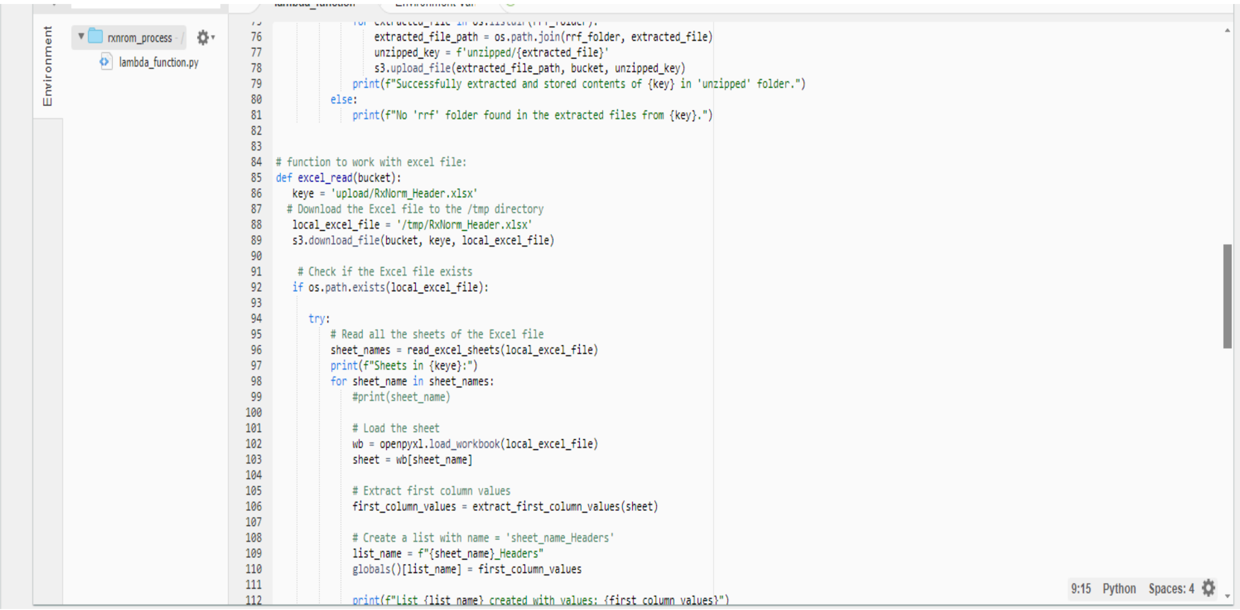




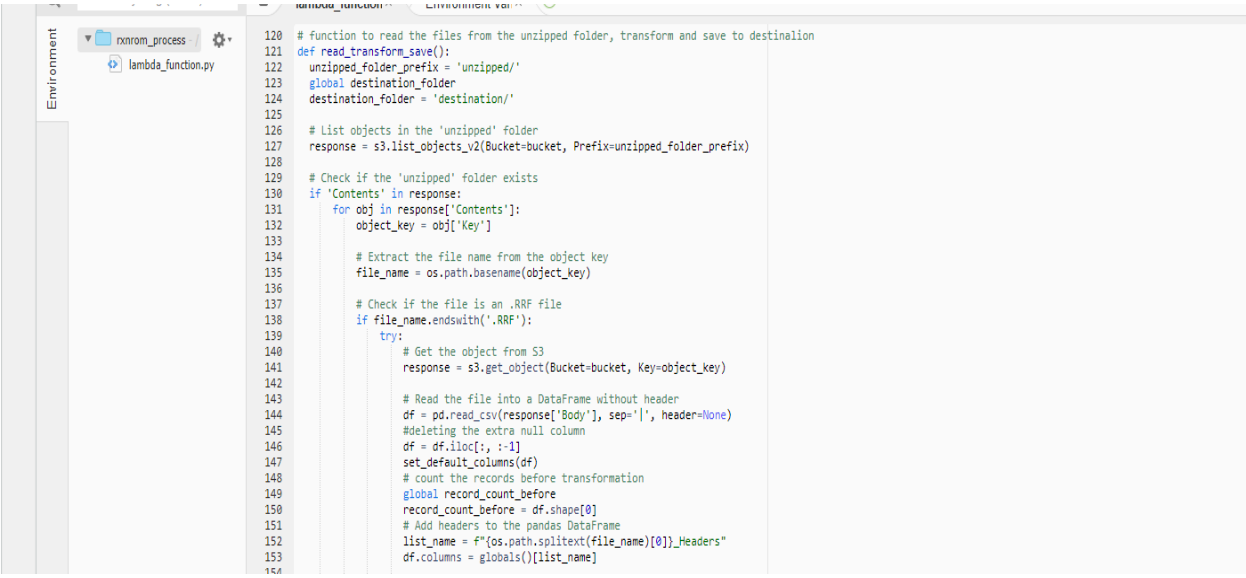
```python
import boto3
import os
import zipfile
import tempfile
import time
import pandas as pd
import openpyxl
from datetime import datetime
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

s3 = boto3.client('s3')

def extract_files_from_zip(zip_file_path, destination_path):
    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        zip_ref.extractall(destination_path)

def read_excel_sheets(file_path):
    try:
        wb = openpyxl.load_workbook(file_path)
        return wb.sheetnames
    except Exception as e:
        print(f"Error reading Excel file: {e}")
        return []

def extract_first_column_values(sheet):
    column_values = []
    for row in sheet.iter_rows(min_row=1, max_row=sheet.max_row, min_col=1, max_col=1, values_only=True):
        column_values.extend(row)
    return column_values

#functions to update the date format
def update_RXNSAB_df(df):
  df['VSTART'] = df['VSTART'].apply(lambda x: datetime.strptime(x, '%Y_%m_%d').strftime('%Y-%m-%d') if pd.notnull(x) else x)
  df['VEND'] = df['VEND'].apply(lambda x: datetime.strptime(x, '%Y_%m_%d').strftime('%Y-%m-%d') if pd.notnull(x) else x)
```



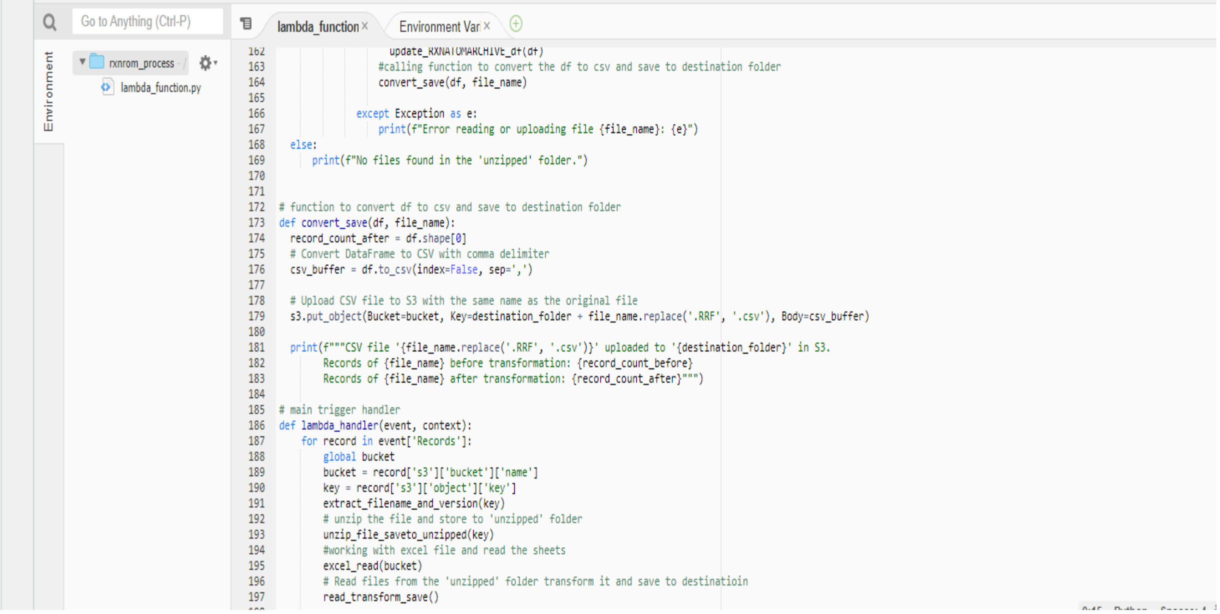
```
def update_RXNATOMARCHIVE_df(df):
  df['CREATED_TIMESTAMP'] = df['CREATED_TIMESTAMP'].apply(lambda x: datetime.strptime(x, "%m/%d/%Y %I:%M:%S %p").strftime("%Y-%m-%d %I:%M:%S %p") if
  df['UPDATED_TIMESTAMP'] = df['UPDATED_TIMESTAMP'].apply(lambda x: datetime.strptime(x, "%m/%d/%Y %I:%M:%S %p").strftime("%Y-%m-%d %I:%M:%S %p") if
  df['LAST_RELEASED'] = df['LAST_RELEASED'].fillna('').astype(str).apply(lambda x: datetime.strptime(x, '%d-%b-%y').strftime('%Y-%m-%d') if x else '

# function to set the default value to the columns:
def set_default_columns(df):
  df['Code set'] = 'Rxnorm'
  df['Version Month'] = version

# function to extract file name and version of the file
def extract_filename_and_version(key):
  global file_name_parts
  file_name_parts = key.split("/")[-1].split(".")[0]
  version_ext = file_name_parts.split("_")[-1]
  date_object = datetime.strptime(version_ext, "%m%d%Y")
  global version
  version = date_object.strftime("%Y-%m-%d")
  logger.log(logging.INFO, "This is version: %s", version)


# function to unzip the .zip file and save to 'unzipped' folder
def unzip_file_saveto_unzipped(key):
  # Check if the uploaded file is a zip file
  if key.endswith('.zip'):
    # Create a temporary directory to extract the zip file contents
    with tempfile.TemporaryDirectory() as tmp_dir:
      # Download the zip file to the temporary directory
      local_zip_file = os.path.join(tmp_dir, 'upload.zip')
      s3.download_file(bucket, key, local_zip_file)

      # Extract the contents of the zip file
      extract_files_from_zip(local_zip_file, tmp_dir)
```



```
        for extracted_file in os.listdir(rrf_folder):
          extracted_file_path = os.path.join(rrf_folder, extracted_file)
          unzipped_key = f'unzipped/{extracted_file}'
          s3.upload_file(extracted_file_path, bucket, unzipped_key)
        print(f"Successfully extracted and stored contents of {key} in 'unzipped' folder.")
      else:
        print(f"No 'rrf' folder found in the extracted files from {key}.")


# function to work with excel file:
def excel_read(bucket):
  keye = 'upload/RxNorm_Header.xlsx'
 # Download the Excel file to the /tmp directory
  local_excel_file = '/tmp/RxNorm_Header.xlsx'
  s3.download_file(bucket, keye, local_excel_file)

   # Check if the Excel file exists
  if os.path.exists(local_excel_file):

    try:
      # Read all the sheets of the Excel file
      sheet_names = read_excel_sheets(local_excel_file)
      print(f"Sheets in {keye}:")
      for sheet_name in sheet_names:
        #print(sheet_name)

        # Load the sheet
        wb = openpyxl.load_workbook(local_excel_file)
        sheet = wb[sheet_name]

        # Extract first column values
        first_column_values = extract_first_column_values(sheet)

        # Create a list with name = 'sheet_name_Headers'
        list_name = f"{sheet_name}_Headers"
        globals()[list_name] = first_column_values

        print(f"List {list_name} created with values: {first_column_values}")
```

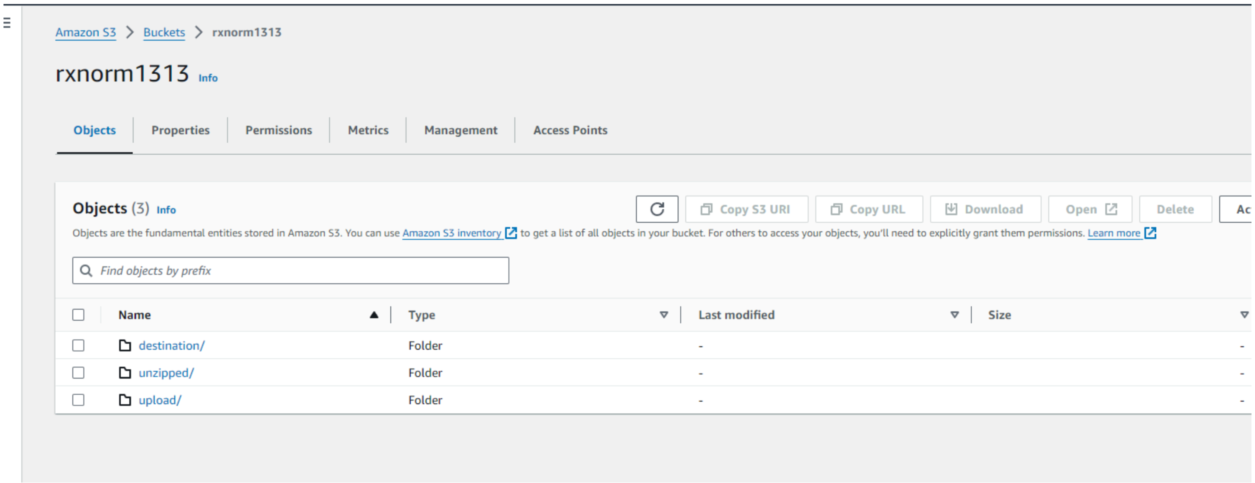


```
# function to read the files from the unzipped folder, transform and save to destinalion
def read_transform_save():
  unzipped_folder_prefix = 'unzipped/'
  global destination_folder
  destination_folder = 'destination/'

  # List objects in the 'unzipped' folder
  response = s3.list_objects_v2(Bucket=bucket, Prefix=unzipped_folder_prefix)

  # Check if the 'unzipped' folder exists
  if 'Contents' in response:
      for obj in response['Contents']:
          object_key = obj['Key']

          # Extract the file name from the object key
          file_name = os.path.basename(object_key)

          # Check if the file is an .RRF file
          if file_name.endswith('.RRF'):
              try:
                  # Get the object from S3
                  response = s3.get_object(Bucket=bucket, Key=object_key)

                  # Read the file into a DataFrame without header
                  df = pd.read_csv(response['Body'], sep='|', header=None)
                  #deleting the extra null column
                  df = df.iloc[:, :-1]
                  set_default_columns(df)
                  # count the records before transformation
                  global record_count_before
                  record_count_before = df.shape[0]
                  # Add headers to the pandas DataFrame
                  list_name = f"{os.path.splitext(file_name)[0]}_Headers"
                  df.columns = globals()[list_name]
```



```
                    update_RXNATOMARCHIVE_df(df)
                  #calling function to convert the df to csv and save to destination folder
                  convert_save(df, file_name)

              except Exception as e:
                  print(f"Error reading or uploading file {file_name}: {e}")
  else:
      print(f"No files found in the 'unzipped' folder.")


# function to convert df to csv and save to destination folder
def convert_save(df, file_name):
  record_count_after = df.shape[0]
  # Convert DataFrame to CSV with comma delimiter
  csv_buffer = df.to_csv(index=False, sep=',')

  # Upload CSV file to S3 with the same name as the original file
  s3.put_object(Bucket=bucket, Key=destination_folder + file_name.replace('.RRF', '.csv'), Body=csv_buffer)

  print(f"""CSV file '{file_name.replace('.RRF', '.csv')}' uploaded to '{destination_folder}' in S3.
        Records of {file_name} before transformation: {record_count_before}
        Records of {file_name} after transformation: {record_count_after}""")

# main trigger handler
def lambda_handler(event, context):
    for record in event['Records']:
        global bucket
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']
        extract_filename_and_version(key)
        # unzip the file and store to 'unzipped' folder
        unzip_file_saveto_unzipped(key)
        #working with excel file and read the sheets
        excel_read(bucket)
        # Read files from the 'unzipped' folder transform it and save to destinatioin
        read_transform_save()
```

S3 folder structure:



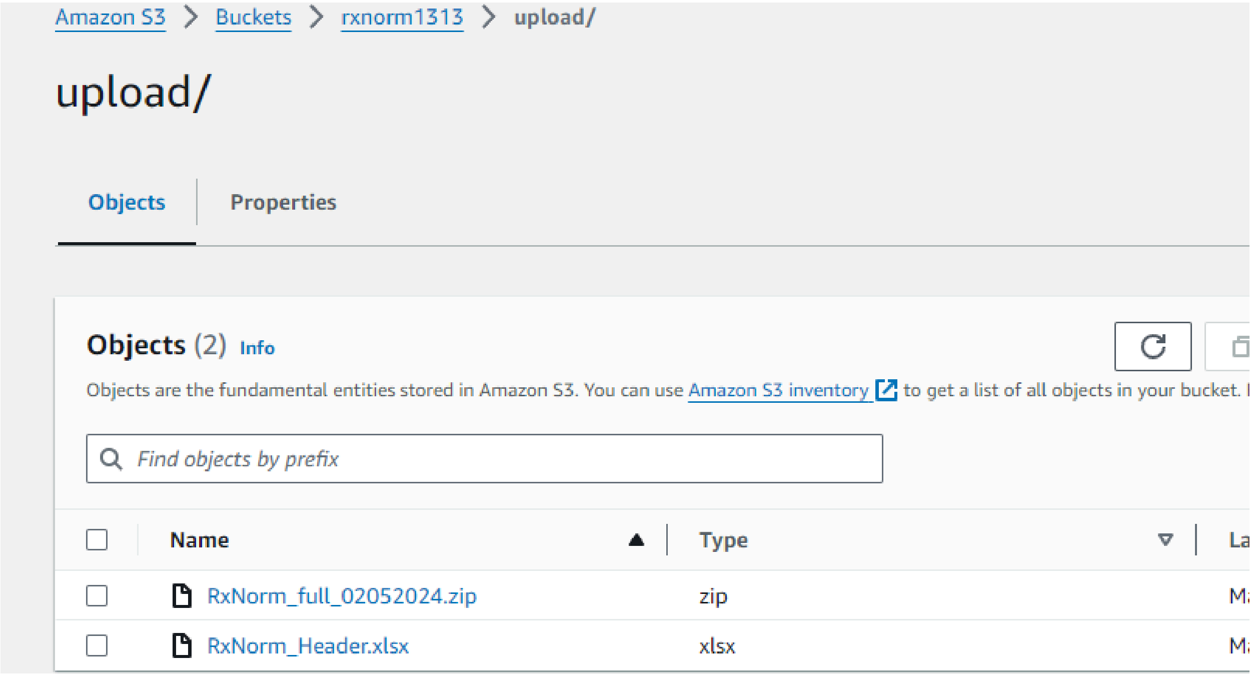
rxnorm1313 Info

Objects (3) 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
destination/	Folder	-	-
unzipped/	Folder	-	-
upload/	Folder	-	-

Upload Folder:



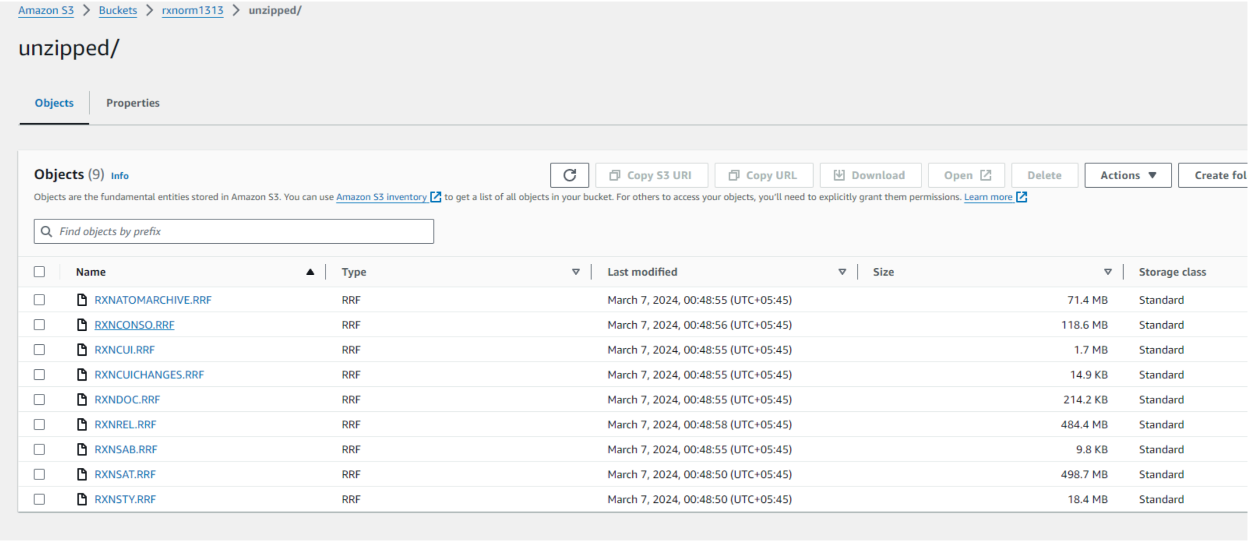
upload/

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. I

Name	Type	La
RxNorm_full_02052024.zip	zip	M
RxNorm_Header.xlsx	xlsx	M

Unzipped folder:



Amazon S3 > Buckets > rxnorm1313 > unzipped/

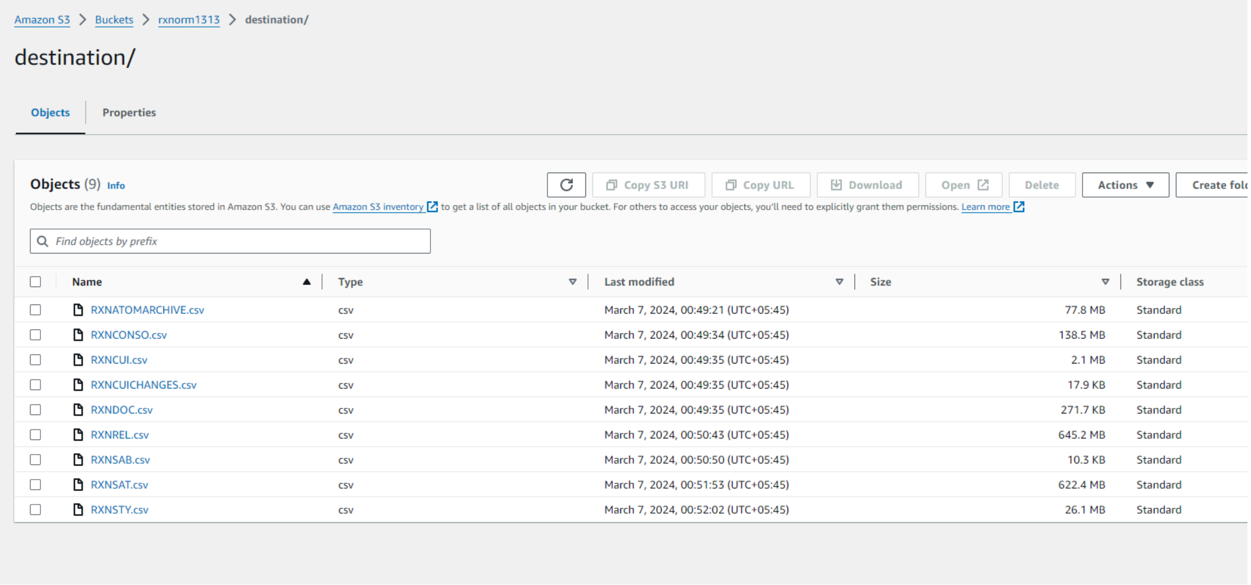
unzipped/

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 7, 2024, 00:48:55 (UTC+05:45)	71.4 MB	Standard
□	RXNCONSO.RRF	RRF	March 7, 2024, 00:48:56 (UTC+05:45)	118.6 MB	Standard
□	RXNCUI.RRF	RRF	March 7, 2024, 00:48:55 (UTC+05:45)	1.7 MB	Standard
□	RXNCUICHANGES.RRF	RRF	March 7, 2024, 00:48:55 (UTC+05:45)	14.9 KB	Standard
□	RXNDOC.RRF	RRF	March 7, 2024, 00:48:55 (UTC+05:45)	214.2 KB	Standard
□	RXNREL.RRF	RRF	March 7, 2024, 00:48:58 (UTC+05:45)	484.4 MB	Standard
□	RXNSAB.RRF	RRF	March 7, 2024, 00:48:55 (UTC+05:45)	9.8 KB	Standard
□	RXNSAT.RRF	RRF	March 7, 2024, 00:48:50 (UTC+05:45)	498.7 MB	Standard
□	RXNSTY.RRF	RRF	March 7, 2024, 00:48:50 (UTC+05:45)	18.4 MB	Standard

Destination folder:



Amazon S3 > Buckets > rxnorm1313 > destination/

destination/

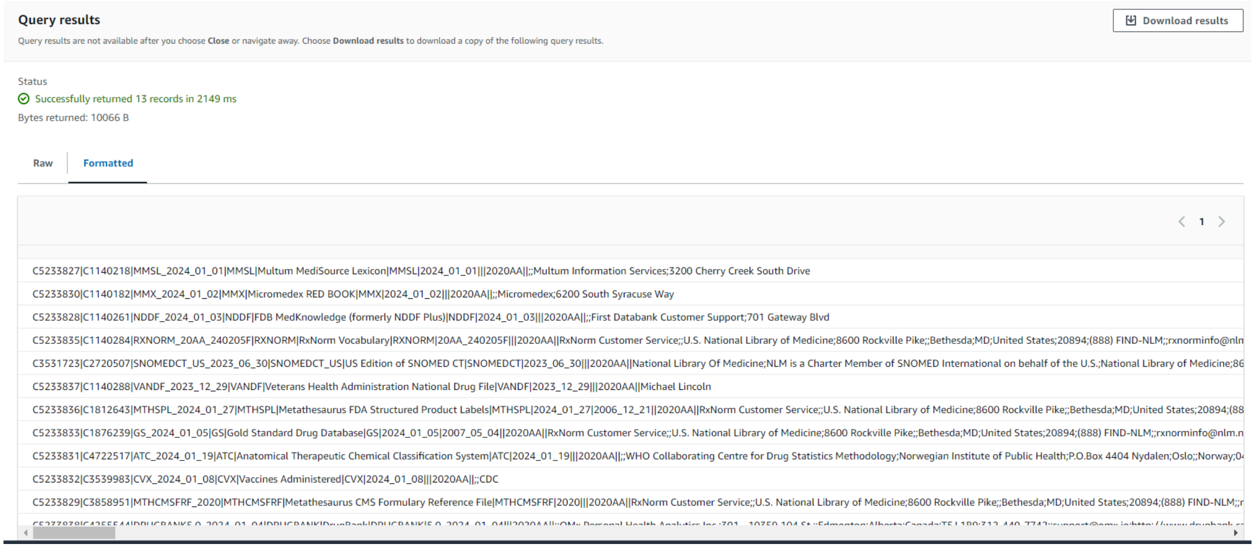
Objects (9) Info

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 7, 2024, 00:49:21 (UTC+05:45)	77.8 MB	Standard
□	RXNCONSO.csv	csv	March 7, 2024, 00:49:34 (UTC+05:45)	138.5 MB	Standard
□	RXNCUI.csv	csv	March 7, 2024, 00:49:35 (UTC+05:45)	2.1 MB	Standard
□	RXNCUICHANGES.csv	csv	March 7, 2024, 00:49:35 (UTC+05:45)	17.9 KB	Standard
□	RXNDOC.csv	csv	March 7, 2024, 00:49:35 (UTC+05:45)	271.7 KB	Standard
□	RXNREL.csv	csv	March 7, 2024, 00:50:43 (UTC+05:45)	645.2 MB	Standard
□	RXNSAB.csv	csv	March 7, 2024, 00:50:50 (UTC+05:45)	10.3 KB	Standard
□	RXNSAT.csv	csv	March 7, 2024, 00:51:53 (UTC+05:45)	622.4 MB	Standard
□	RXNSTY.csv	csv	March 7, 2024, 00:52:02 (UTC+05:45)	26.1 MB	Standard

Visualizing the original file and Transformed file
Example for RXNSAB file,

RXNSAB.RRF:



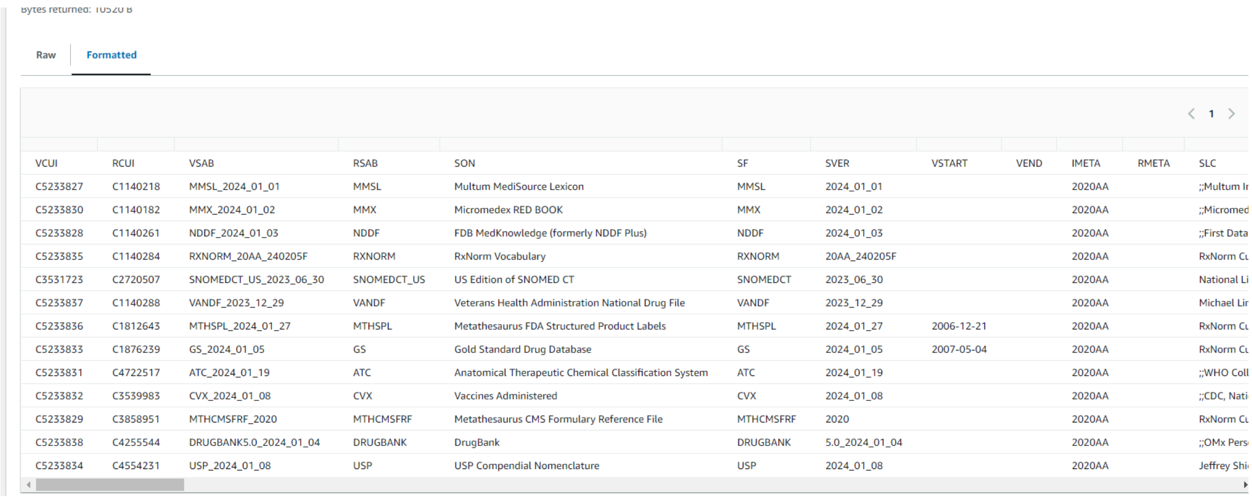
Query results

Query results are not available after you choose **Close** or navigate away. Choose **Download results** to download a copy of the following query results.

Status
Successfully returned 13 records in 2149 ms
Bytes returned: 10066 B

```
C5233827|C1140218|MMSL_2024_01_01|MMSL|Multum MediSource Lexicon|MMSL|2024_01_01|||2020AA||;;Multum Information Services;3200 Cherry Creek South Drive
C5233830|C1140182|MMX_2024_01_02|MMX|Micromedex RED BOOK|MMX|2024_01_02|||2020AA||;;Micromedex;6200 South Syracuse Way
C5233828|C1140261|NDDF_2024_01_03|NDDF|FDB MedKnowledge (formerly NDDF Plus)|NDDF|2024_01_03|||2020AA||;;First Databank Customer Support;701 Gateway Blvd
C5233835|C1140284|RXNORM_20AA_240205F|RXNORM|RxNorm Vocabulary|RXNORM|20AA_240205F|||2020AA||RxNorm Customer Service;;U.S. National Library of Medicine;8600 Rockville Pike;;Bethesda;MD;United States;20894;(888) FIND-NLM;;rxnorminfo@nlm
C3531723|C2720507|SNOMEDCT_US_2023_06_30|SNOMEDCT_US|US Edition of SNOMED CT|SNOMEDCT|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;86
C5233837|C1140288|VANDF_2023_12_29|VANDF|Veterans Health Administration National Drug File|VANDF|2023_12_29|||2020AA||Michael Lincoln
C5233836|C1812643|MTHSPL_2024_01_27|MTHSPL|Metathesaurus FDA Structured Product Labels|MTHSPL|2024_01_27|2006_12_21||2020AA||RxNorm Customer Service;;U.S. National Library of Medicine;8600 Rockville Pike;;Bethesda;MD;United States;20894;(88
C5233833|C1876239|GS_2024_01_05|GS|Gold Standard Drug Database|GS|2024_01_05|2007_05_04||2020AA||RxNorm Customer Service;;U.S. National Library of Medicine;8600 Rockville Pike;;Bethesda;MD;United States;20894;(888) FIND-NLM;;rxnorminfo@nlm.n
C5233831|C4722517|ATC_2024_01_19|ATC|Anatomical Therapeutic Chemical Classification System|ATC|2024_01_19|||2020AA||;;WHO Collaborating Centre for Drug Statistics Methodology;Norwegian Institute of Public Health;P.O.Box 4404 Nydalen;Oslo;;Norway;04
C5233832|C3539983|CVX_2024_01_08|CVX|Vaccines Administered|CVX|2024_01_08|||2020AA||;;CDC
C5233829|C3858951|MTHCMSFRF_2020|MTHCMSFRF|Metathesaurus CMS Formulary Reference File|MTHCMSFRF|2020|||2020AA||RxNorm Customer Service;;U.S. National Library of Medicine;8600 Rockville Pike;;Bethesda;MD;United States;20894;(888) FIND-NLM;;r
C5233838|C4255544|DRUGBANK5.0_2024_01_04|DRUGBANK|DrugBank|DRUGBANK|5.0_2024_01_04|||2020AA||;;OMx Personal Health Analytics Inc.;301 - 10359 104 St.;;Edmonton;Alberta;Canada;T5J 1B9;312-440-7742;;support@omx.io;http://www.drugbank.ca
```

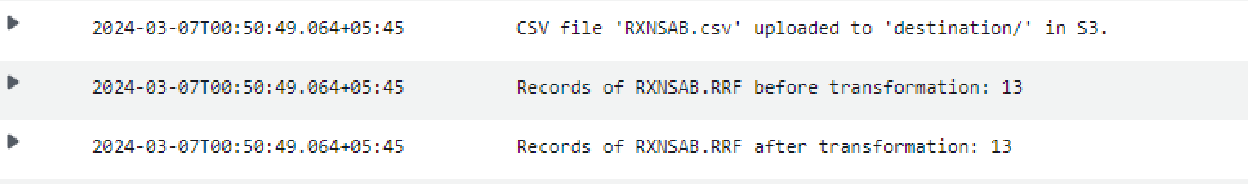
RXNSAB.csv:



Bytes returned: 10520 B

VCUI	RCUI	VSAB	RSAB	SON	SF	SVER	VSTART	VEND	IMETA	RMETA	SLC
C5233827	C1140218	MMSL_2024_01_01	MMSL	Multum MediSource Lexicon	MMSL	2024_01_01			2020AA		;;Multum Ir
C5233830	C1140182	MMX_2024_01_02	MMX	Micromedex RED BOOK	MMX	2024_01_02			2020AA		;;Micromed
C5233828	C1140261	NDDF_2024_01_03	NDDF	FDB MedKnowledge (formerly NDDF Plus)	NDDF	2024_01_03			2020AA		;;First Data
C5233835	C1140284	RXNORM_20AA_240205F	RXNORM	RxNorm Vocabulary	RXNORM	20AA_240205F			2020AA		RxNorm Cu
C3531723	C2720507	SNOMEDCT_US_2023_06_30	SNOMEDCT_US	US Edition of SNOMED CT	SNOMEDCT	2023_06_30			2020AA		National Li
C5233837	C1140288	VANDF_2023_12_29	VANDF	Veterans Health Administration National Drug File	VANDF	2023_12_29			2020AA		Michael Lir
C5233836	C1812643	MTHSPL_2024_01_27	MTHSPL	Metathesaurus FDA Structured Product Labels	MTHSPL	2024_01_27	2006-12-21		2020AA		RxNorm Cu
C5233833	C1876239	GS_2024_01_05	GS	Gold Standard Drug Database	GS	2024_01_05	2007-05-04		2020AA		RxNorm Cu
C5233831	C4722517	ATC_2024_01_19	ATC	Anatomical Therapeutic Chemical Classification System	ATC	2024_01_19			2020AA		;;WHO Coll
C5233832	C3539983	CVX_2024_01_08	CVX	Vaccines Administered	CVX	2024_01_08			2020AA		;;CDC, Nati
C5233829	C3858951	MTHCMSFRF_2020	MTHCMSFRF	Metathesaurus CMS Formulary Reference File	MTHCMSFRF	2020			2020AA		RxNorm Cu
C5233838	C4255544	DRUGBANK5.0_2024_01_04	DRUGBANK	DrugBank	DRUGBANK	5.0_2024_01_04			2020AA		;;OMx Pers
C5233834	C4554231	USP_2024_01_08	USP	USP Compendial Nomenclature	USP	2024_01_08			2020AA		Jeffrey Shi

Validating the record for RXNSAB file:



```
▶   2024-03-07T00:50:49.064+05:45      CSV file 'RXNSAB.csv' uploaded to 'destination/' in S3.
▶   2024-03-07T00:50:49.064+05:45      Records of RXNSAB.RRF before transformation: 13
▶   2024-03-07T00:50:49.064+05:45      Records of RXNSAB.RRF after transformation: 13
```

Same goes for all the 9 files which can be verified with cloudwatch logs.