1. Data Migration and Transformation Tool for Amazon RDS Data Warehouses

Overview

Goal:

Extraction of the data from a zip file that is available at a URL and uploading it into Amazon S3 and Amazon RDS.

Technologies used:

Python, Requests library, Zipfile library, boto3 library, psycopg2 library pandas, Amazon S3, Amazon RDS

Steps:

1. Using Python libraries like [requests, urlib, wget, curl] to download the zip file from the URL.
2. Zipfile library in Python to extract the data from the zip file. This will create a file or directory containing the extracted data.
3. AWS SDK for Python (Boto3) to create an S3 bucket.
4. Using put_object function of Boto3 to upload the extracted data to the S3 bucket.
5. Boto3 is used to create an RDS instance.
6. Using psycopg2 library in Python to connect to the RDS instance.
7. Using pandas library in Python to read the extracted data into a DataFrame and this data is pushed to RDS instance.

The codes used and screenshots of the task done is given below.

Codes used:

Using boto3 a connection was established with the local system and amazon s3 services. After that a session and client were created in order to push the object from URL to s3 bucket

```
import boto3
```
✓ 0.3s

```
s3 = boto3.resource(
    service_name='s3',
    region_name='ap-south-1',
    aws_access_key_id='AKIAQMHO4RIROXREEIXX',
    aws_secret_access_key='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'
)
```
✓ 0.3s

```
for bucket in s3.buckets.all():
    print(bucket.name)
```
✓ 0.7s

```
migbuck1
project1hari
```

```
session = boto3.Session(
aws_access_key_id='AKIAQMHO4RIROXREEIXX',
aws_secret_access_key='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'
)
```
✓ 0.1s

```
s3 = session.client(service_name='s3',region_name='ap-south-1')
```
✓ 0.5s

Using the requests library, the zip file was downloaded from the URL.

Using put_object() the zipfile was uploaded to s3 and then using the zipfile module the zipfile was extracted into another folder inside s3.

```
for i in s3.list_buckets().get('Buckets'):
    print(i['Name'])
```

✓ 0.4s

```
migbuck1
project1hari
```

```
import requests

try:
    url = 'https://www.sec.gov/Archives/edgar/daily-index/bulkdata/submissions.zip'

    headers = {'user-agent':
        'mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, Like Gecko) Chrome/109.0.0.0 Safari/537.36 Edg/109.0.1518.55'}

    response = requests.get(url,headers = headers,stream=True)

    content = response.content

    print('successfully got the zipfile from url')

except Exception as e:
    print(e)
```

✓ 1.7s

```
successfully got the zipfile from url
```

```
try:
    bucket_name = 'migbuck1'
    file_path = 'try1/file.zip'

    s3.put_object(Bucket=bucket_name, Key=file_path, Body=content)
    print("successfully upload the zipfile into s3")
except Exception as e:
    print(e)
```

✓ 0.4s

```
successfully upload the zipfile into s3
```

```
import io, zipfile


try:
    bucket_name = 'migbuck1'
    file_path = 'try1/file.zip'

    obj = s3.get_object(Bucket=bucket_name, Key=file_path)

    zip_file = io.BytesIO(obj['Body'].read())

    with zipfile.ZipFile(zip_file) as z:
        z.extractall()

    for file in z.namelist():
        s3.upload_file(Filename=file, Bucket=bucket_name, Key='unzipped/' + file)

    print('Successfully extracted and uploaded in the unzipped folder')

except Exception as e:
    print(e)
```

Issues faced while doing the project:

The zip file downloaded from the given URL was not accessible and it had badzip file error. More over the URL was giving a 403 forbidden error. Hence the zip file was downloaded manually to the local system and only 50 files were uploaded into the s3 from the local system to check the code and complete the project.

```
import io, zipfile


try:
    bucket_name = 'migbuck1'
    file_path = 'try1/file.zip'

    obj = s3.get_object(Bucket=bucket_name, Key=file_path)

    zip_file = io.BytesIO(obj['Body'].read())

    with zipfile.ZipFile(zip_file) as z:
        z.extractall()

    for file in z.namelist():
        s3.upload_file(Filename=file, Bucket=bucket_name, Key='unzipped/' + file)

    print('Successfully extracted and uploaded in the unzipped folder')

except Exception as e:
    print(e)
```

✓ 0.8s

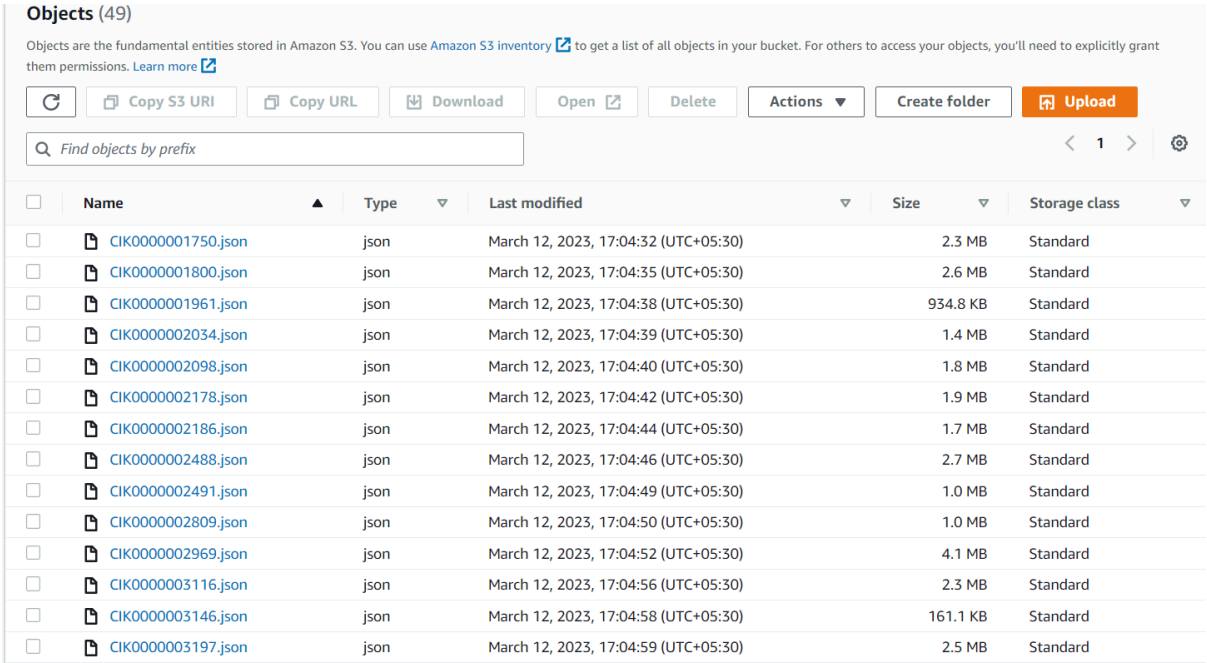
```
File is not a zip file
```

From the local system using Boto3, fifty files were uploaded to S3 using the code below

In [27]:
```
import os
import boto3
```

In [35]:
```
s3 = boto3.client(
    service_name='s3',
    region_name='ap-south-1',
    aws_access_key_id='AKIAQMHO4RIROXREEIXX',
    aws_secret_access_key='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'
)

bucket = "migbuck1"
prefix = "try1/"

filepath = r"C:\Users\Malavika Vipin\Desktop\Hari\project_1"
file_names = os.listdir(filepath)

for i, filename in enumerate(file_names):

    file_key = filepath +"\\" + filename
    if i == 49:
        break
    object_name = prefix  + os.path.basename(filename)

    s3.upload_file(file_key, bucket, object_name)
```

Result for the above code



Objects (49)

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

Copy S3 URI | Copy URL | Download | Open | Delete | Actions | Create folder | Upload

Find objects by prefix

1

Name	Type	Last modified	Size	Storage class
CIK0000001750.json	json	March 12, 2023, 17:04:32 (UTC+05:30)	2.3 MB	Standard
CIK0000001800.json	json	March 12, 2023, 17:04:35 (UTC+05:30)	2.6 MB	Standard
CIK0000001961.json	json	March 12, 2023, 17:04:38 (UTC+05:30)	934.8 KB	Standard
CIK0000002034.json	json	March 12, 2023, 17:04:39 (UTC+05:30)	1.4 MB	Standard
CIK0000002098.json	json	March 12, 2023, 17:04:40 (UTC+05:30)	1.8 MB	Standard
CIK0000002178.json	json	March 12, 2023, 17:04:42 (UTC+05:30)	1.9 MB	Standard
CIK0000002186.json	json	March 12, 2023, 17:04:44 (UTC+05:30)	1.7 MB	Standard
CIK0000002488.json	json	March 12, 2023, 17:04:46 (UTC+05:30)	2.7 MB	Standard
CIK0000002491.json	json	March 12, 2023, 17:04:49 (UTC+05:30)	1.0 MB	Standard
CIK0000002809.json	json	March 12, 2023, 17:04:50 (UTC+05:30)	1.0 MB	Standard
CIK0000002969.json	json	March 12, 2023, 17:04:52 (UTC+05:30)	4.1 MB	Standard
CIK0000003116.json	json	March 12, 2023, 17:04:56 (UTC+05:30)	2.3 MB	Standard
CIK0000003146.json	json	March 12, 2023, 17:04:58 (UTC+05:30)	161.1 KB	Standard
CIK0000003197.json	json	March 12, 2023, 17:04:59 (UTC+05:30)	2.5 MB	Standard

Pushing the data from S3 to RDS instance using the following code

Here using the below code, we are establishing a connection between RDS and S3 using psycopg2 and boto3 libraries.

```
import psycopg2
import boto3
import json
import pandas as pd

# establishing coonection with the RDS instance
conn = psycopg2.connect(
    host="database-1.co6hkqdzo3zl.us-east-1.rds.amazonaws.com",
    database="",
    user="postgres",
    password="admin1234"
)

# creating a client using S3 bucket details
s3 = boto3.client('s3',
                  aws_access_key_id='AKIAQMHO4RIROXREEIXX',
                  aws_secret_access_key='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')

# creating a resource using s3 bucket details
s3_resource = boto3.resource('s3',
                  aws_access_key_id='AKIAQMHO4RIROXREEIXX',
                  aws_secret_access_key='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')

#specifying the targeted bucket
my_bucket = s3_resource.Bucket("migbuck1")
```

After that we are creating a table of our desire inside RDS for accommodating the data.

```
# create a cursor object to execute SQL queries
cur = conn.cursor()

# creating a new table with the desired name
cur.execute("""
    CREATE TABLE project1 (
        type varchar(30),
        cik int,
        entityName varchar(30),
        facts text
    )
""")
conn.commit()
```

From S3 we are accessing the data in order to push it inside the RDS

```
# accessing the data inside S3 bucket and looping it one by one for pushing it into RDS
for i, obj_summary in enumerate(my_bucket.objects.filter(Prefix="try1/")):
    if ".json" in obj_summary.key:
        s3_object = s3.get_object(Bucket='migbuck1', Key=obj_summary.key)
        s3_data = s3_object['Body'].read().decode('utf-8')
        data = json.loads(s3_data)
        #print(obj_summary.key)

        # create a cursor object to execute SQL queries
        cur = conn.cursor()

        # converting the data into a dataframe using pandas
        df = pd.DataFrame(data)

        # reseting index for better readability
        df = df.reset_index()
```

Here we are pushing data inside the table created in the RDS using for loop

```
# pushing data into the table created inside RDS
for row in data:
    cur.execute(f"INSERT INTO project1 (type, cik, entityname, facts) VALUES (%s, %s, %s, %s)", (row['column1'], row['column2'], row['column3']))

    for ind in range(df.shape[0]):
      df.iloc[ind]
      cur.execute(f"INSERT INTO project1 (type, cik, entityName, facts) VALUES (%s, %s, %s, %s)",(df.iloc[ind]['index'],int(df.iloc[ind]['cik']),df.iloc[ind]['entityName'
      conn.commit()
```

Using select query, we are able to access the data stored inside the RDS

```
# fetching the data from the RDS using select queries
cur = conn.cursor()
cur.execute("SELECT * from project1")
results = cur.fetchall()
print(results)
conn.commit()

# close the cursor and connection objects
cur.close()
conn.close()
```

Results:

Converting the data into DataFrames

```
In [11]: df
```

Out[11]:

	index	cik	entityName	facts
0	dei	1750	AAR CORP	{'EntityCommonStockSharesOutstanding': {'label...
1	us-gaap	1750	AAR CORP	{'AccountsPayableCurrent': {'label': 'Accounts...

Looping the data into the table created inside RDS

```
In [ ]: f"INSERT INTO project1 (type, cik, entityname, facts) VALUES (%s, %s, %s, %s)", (row['column1'], row['column2'], row['column3'])
```

```
In [21]: for ind in range(df.shape[0]):
             df.iloc[ind]
             print(f"INSERT INTO project1 (type, cik, entityname, facts) VALUES (%s, %s, %s)",(df.iloc[ind]['index'],df.iloc[ind]['cik'],c
             break
```

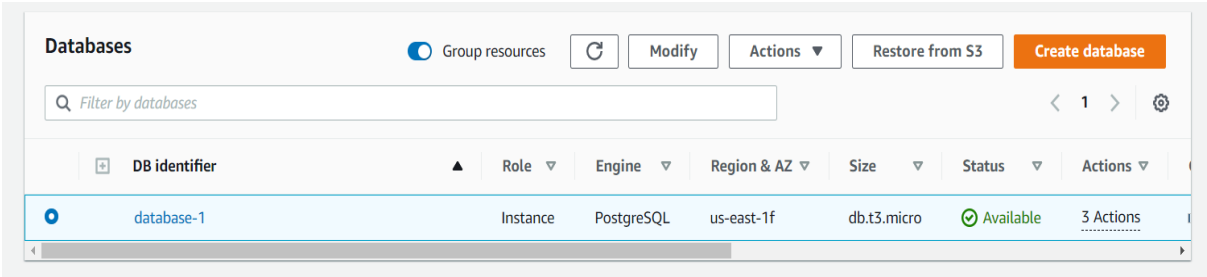
```
INSERT INTO project1 (type, cik, entityname, facts) VALUES (%s, %s, %s) ('dei', 1750, 'AAR CORP')
```

Fetching the data from the RDS using select query

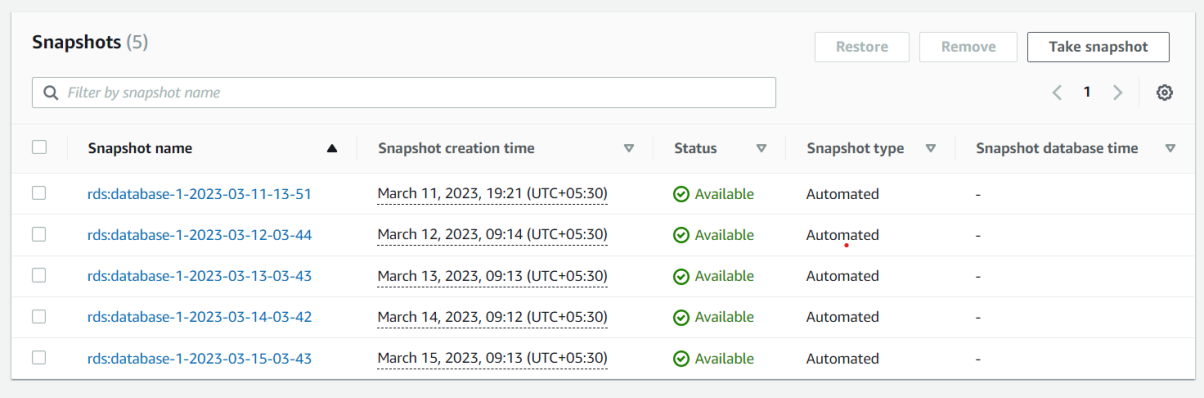
```
# fetching the data from the RDS using select queries
cur = conn.cursor()
cur.execute("SELECT type,cik,entityname from project1")
results = cur.fetchall()
print(results)
conn.commit()

# close the cursor and connection objects
cur.close()
conn.close()
```

```
[('dei', 1750, 'AAR CORP'), ('us-gaap', 1750, 'AAR CORP'), ('dei', 1750, 'AAR CORP'), ('us-gaap', 1750, 'AAR CORP'), ('dei', 18
00, 'ABBOTT LABORATORIES'), ('us-gaap', 1800, 'ABBOTT LABORATORIES'), ('dei', 1961, 'WORLDS INC.'), ('invest', 1961, 'WORLDS IN
C.'), ('us-gaap', 1961, 'WORLDS INC.'), ('dei', 2034, 'ACETO CORP'), ('us-gaap', 2034, 'ACETO CORP')]
```

RDS database that was created



Snapshots of the RDS database created



Snapshots (5)

Restore | Remove | Take snapshot

Filter by snapshot name

1

Snapshot name	Snapshot creation time	Status	Snapshot type	Snapshot database time
rds:database-1-2023-03-11-13-51	March 11, 2023, 19:21 (UTC+05:30)	Available	Automated	-
rds:database-1-2023-03-12-03-44	March 12, 2023, 09:14 (UTC+05:30)	Available	Automated	-
rds:database-1-2023-03-13-03-43	March 13, 2023, 09:13 (UTC+05:30)	Available	Automated	-
rds:database-1-2023-03-14-03-42	March 14, 2023, 09:12 (UTC+05:30)	Available	Automated	-
rds:database-1-2023-03-15-03-43	March 15, 2023, 09:13 (UTC+05:30)	Available	Automated	-