ASSIGNMENT 3: PART A

TASK A: Build, deploy, and run a Containerized Application using GCP.

LINK TO GITLAB: https://git.cs.dal.ca/rtyagi/csci5410_b00872269_ruhirajnish_tyagi/-/tree/main/assignment-3

Language used: Python 3.8.9
Library used for AWS SDK: boto3 [1].

Step 1: Buckets creation and uploading files to each programmatically .

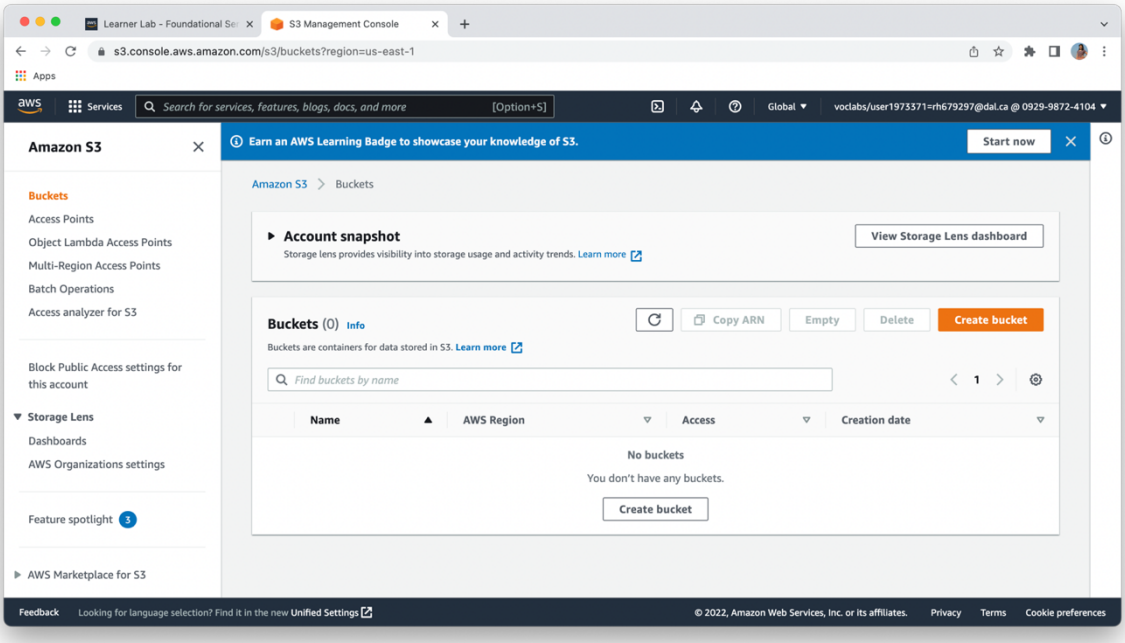

Figure 1. *Screenshot of Amazon S3 console BEFORE creating buckets.*

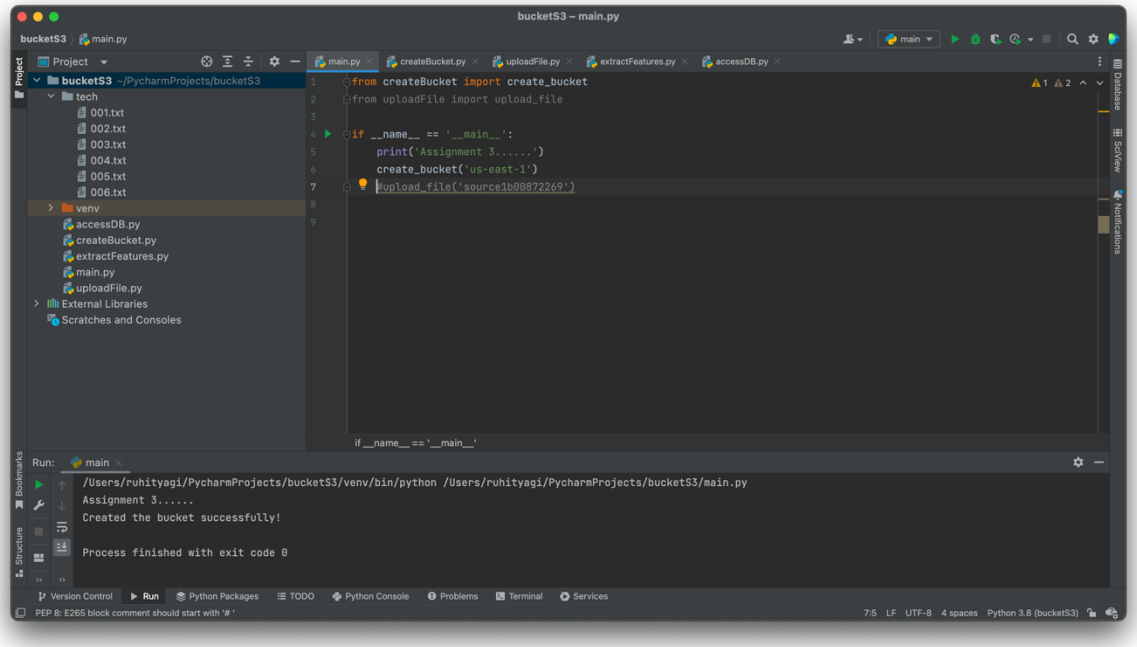

Figure 2. *Screenshot of main.py for creating bucket.*

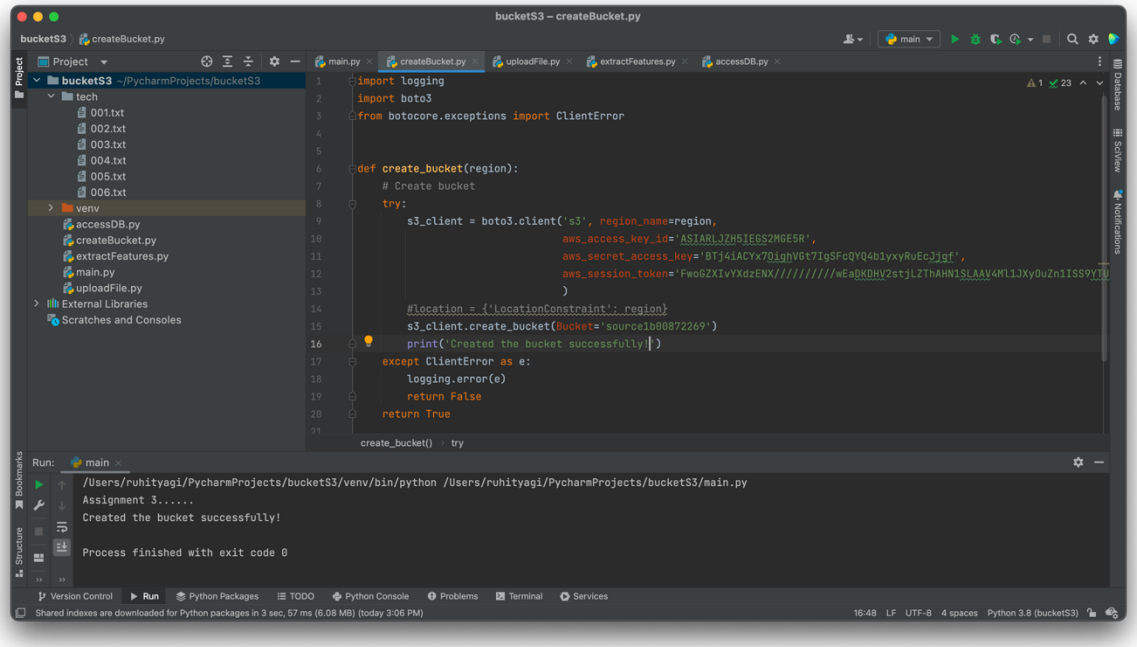

Figure 3. *Screenshot of successful creation of bucket* ***source1b00872269*** *.*

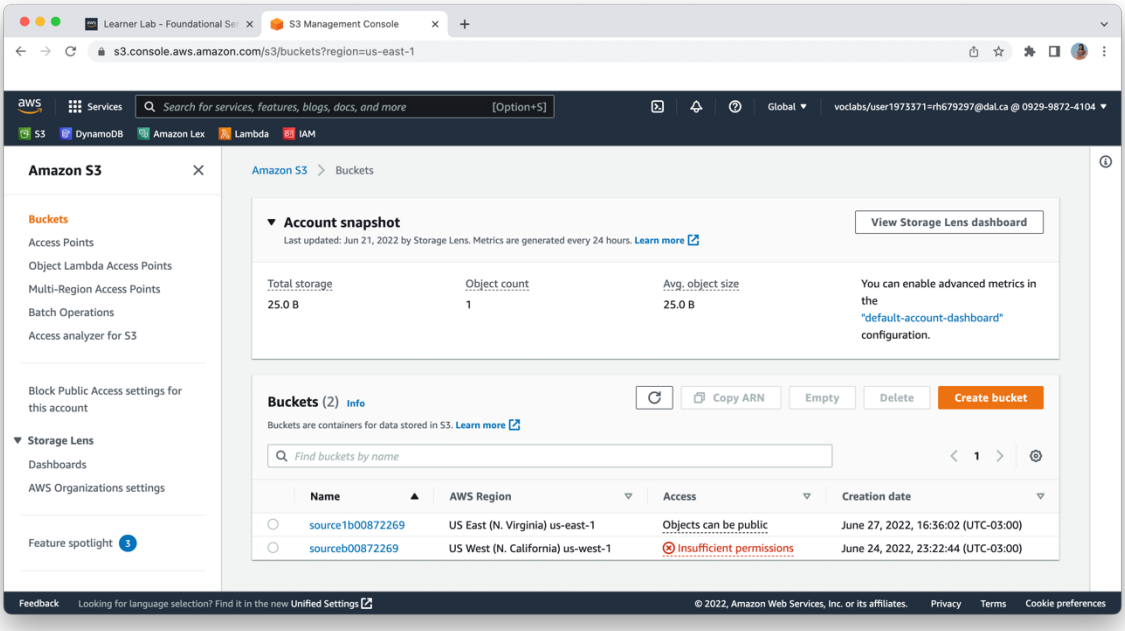

Figure 4. *Screenshot of successful creation of bucket* ***source1b00872269*** *in Amazon S3 console .*

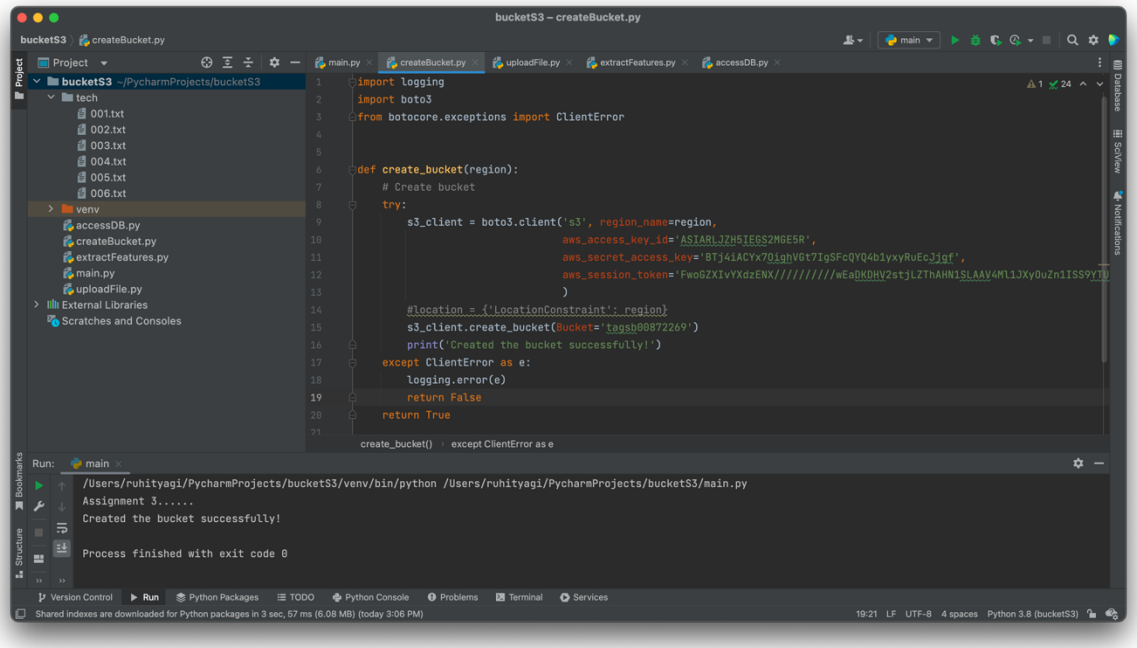

Figure 5. *Screenshot of successful creation of bucket* ***tagsb00872269*** *.*

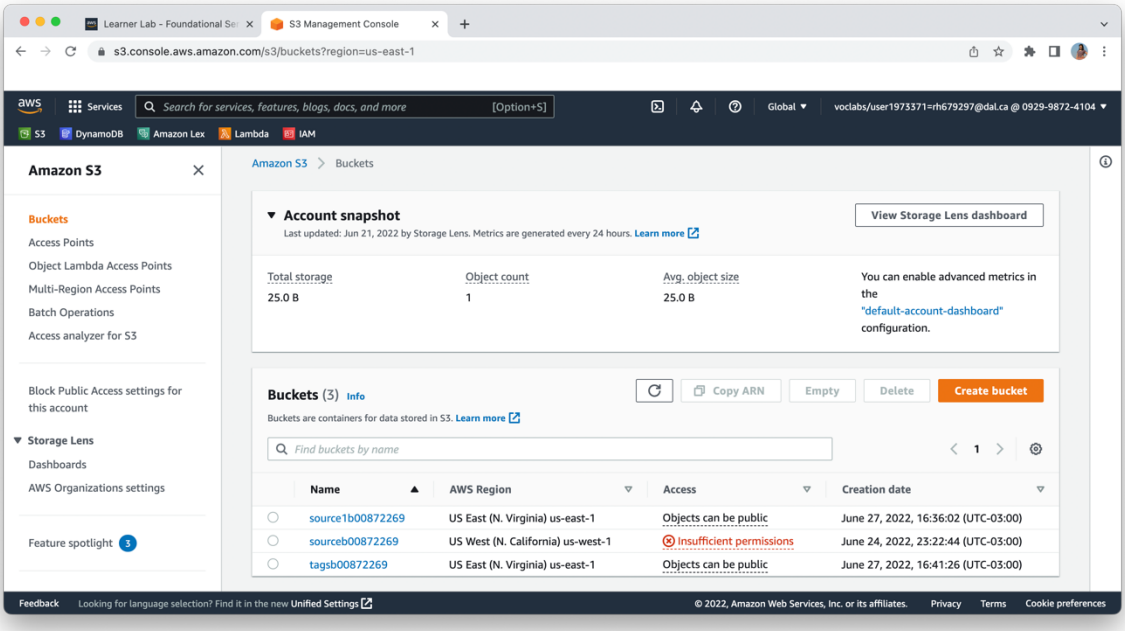

Figure 6. *Screenshot of successful creation of bucket* ***source1b00872269*** *in Amazon S3 console .*

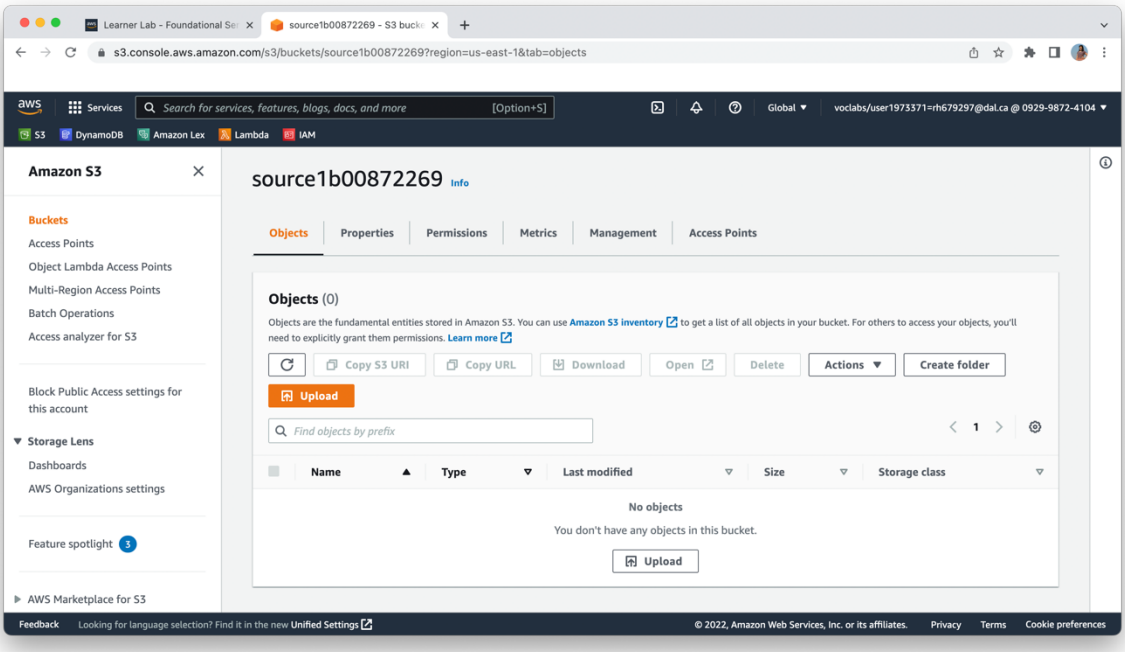

Figure 7. *Screenshot of empty bucket* ***source1b00872269*** *in Amazon S3 console .*

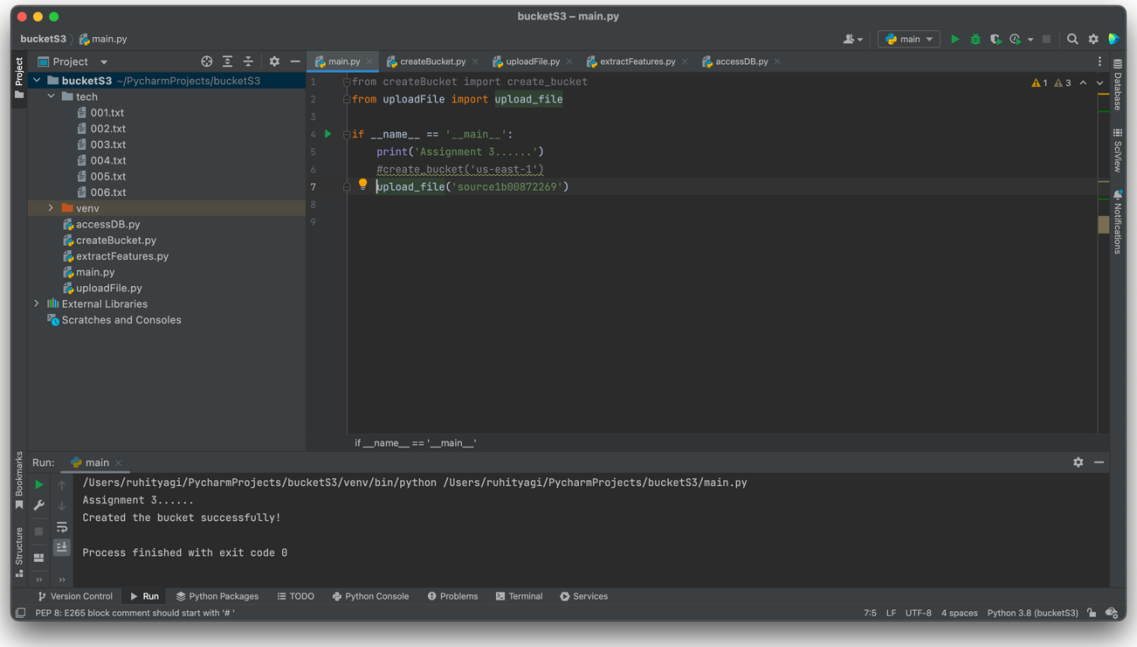

Figure 8. *Screenshot of main.py for uploading files to bucket.*

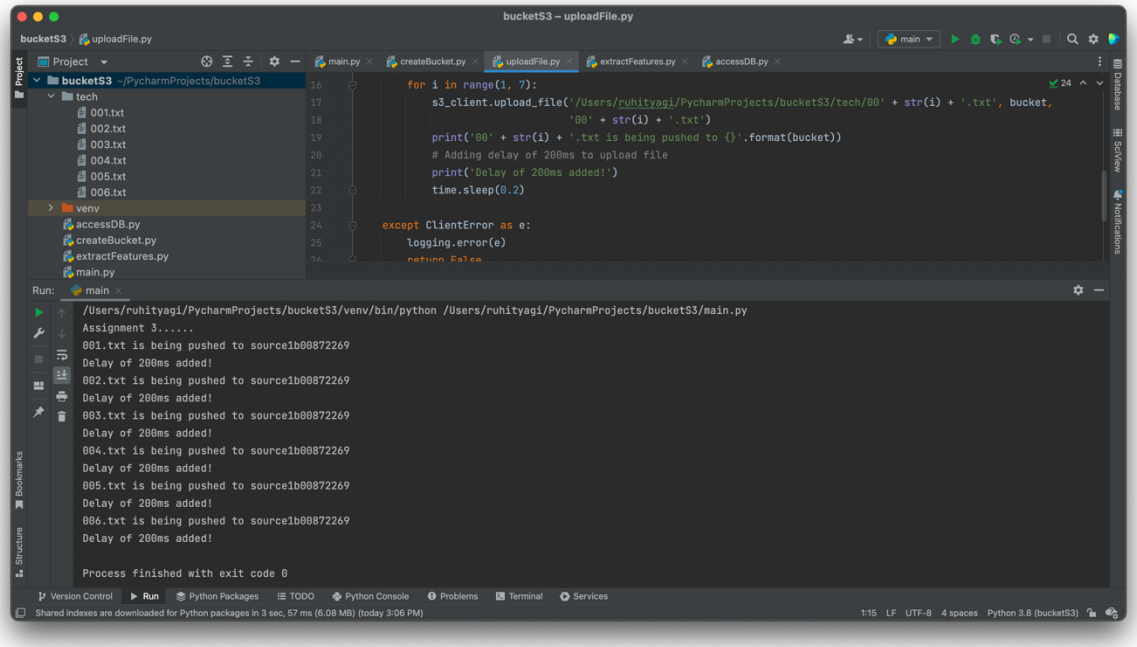

Figure 9. *Screenshot of uploading file to bucket* ***source1b00872269*** *programmatically.*

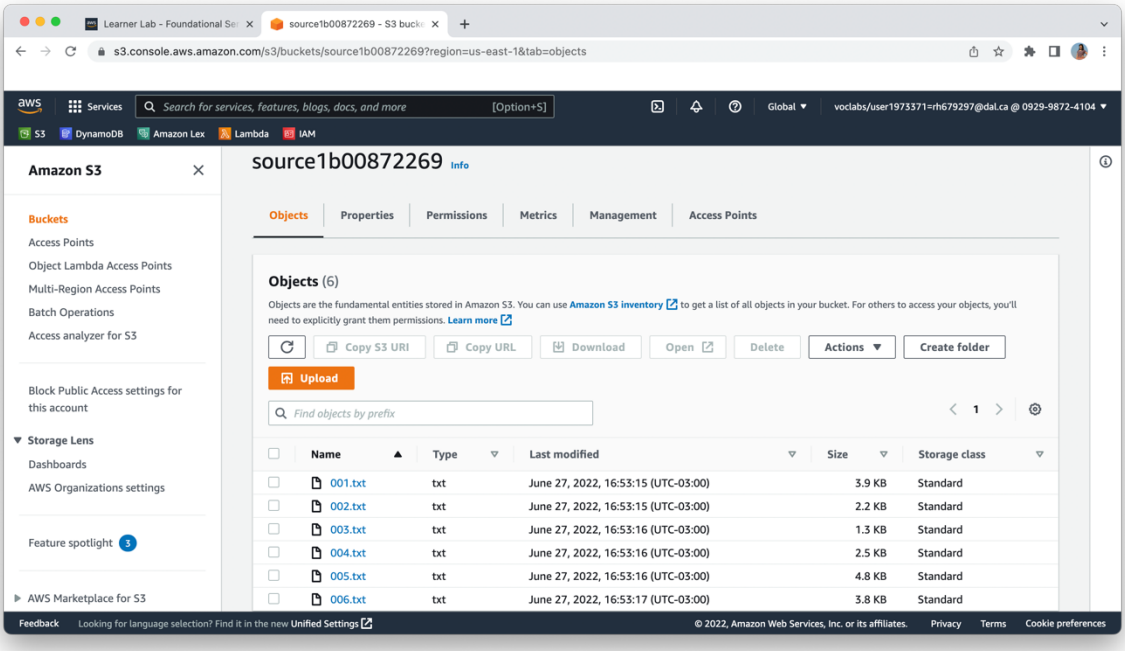

Figure 10. *Screenshot of SUCCESSFUL upload of .txt's to bucket* ***source1b00872269****.*

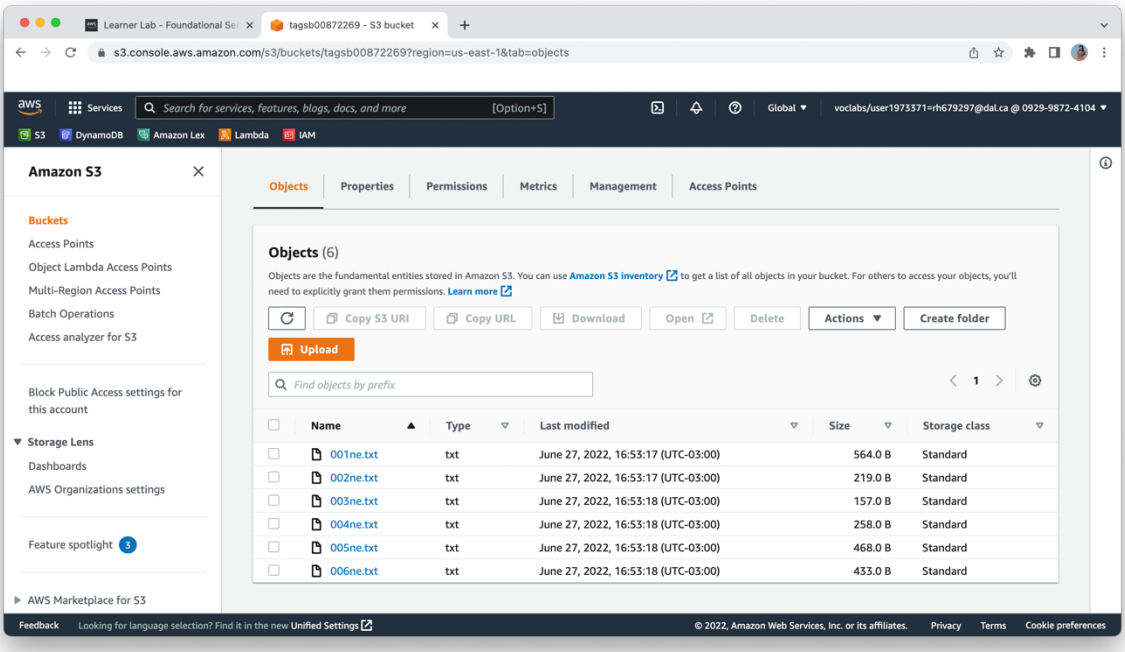

Figure 11. *Screenshot of SUCCESSFUL upload of ***ne.txt's to bucket* ***tagsb00872269****.*

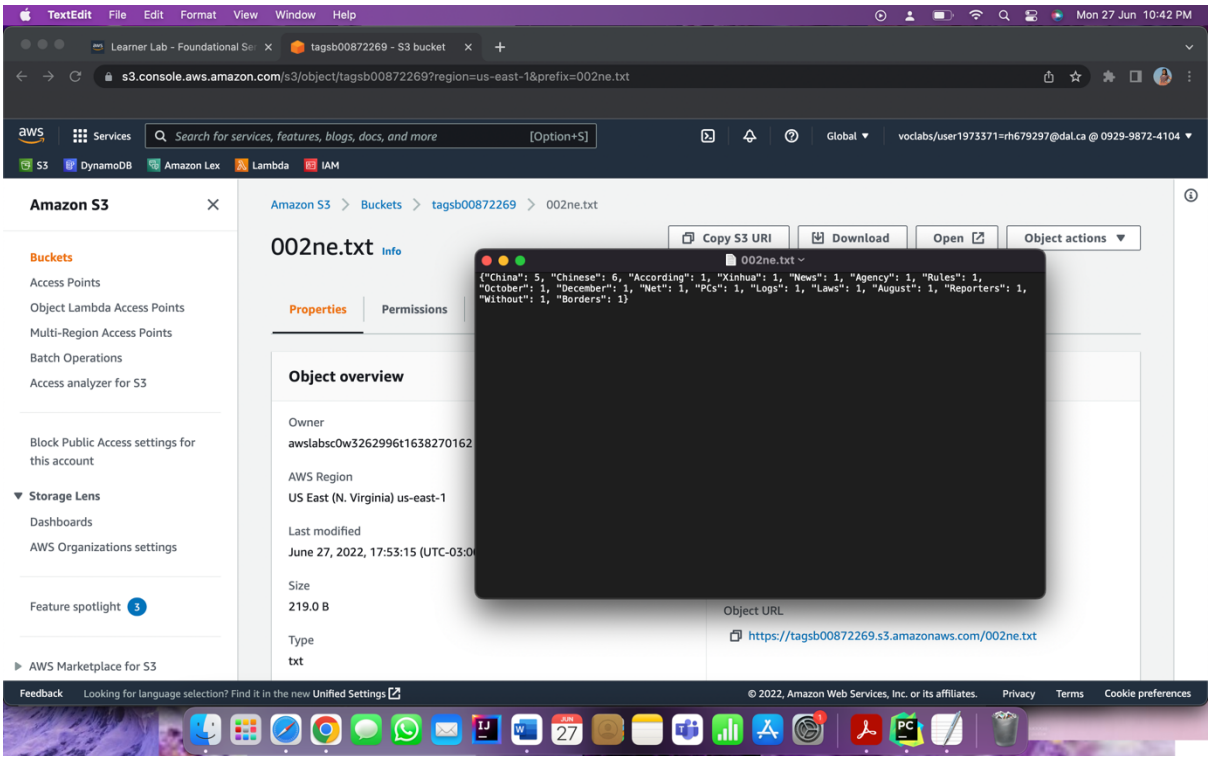

Figure 12. *Screenshot of contents of 002ne.txt's.*

Step 2: Rendering Lambda functions and updating DynamoDB table.

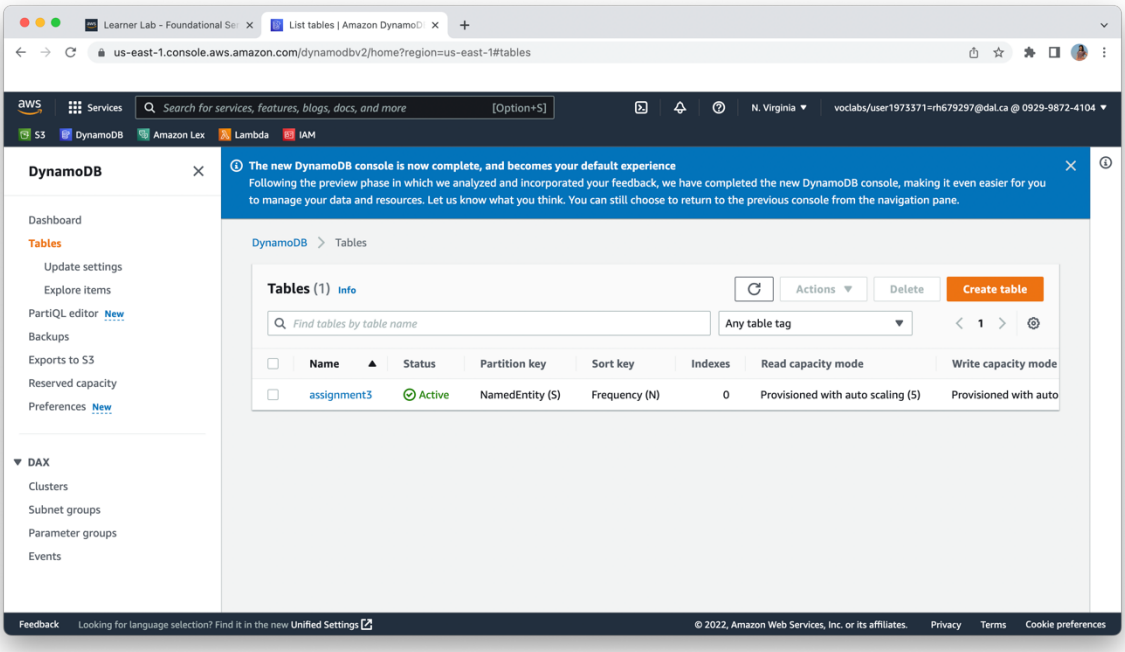

Figure 13. *Screenshot of table assignment3 in DynamoDB console.*

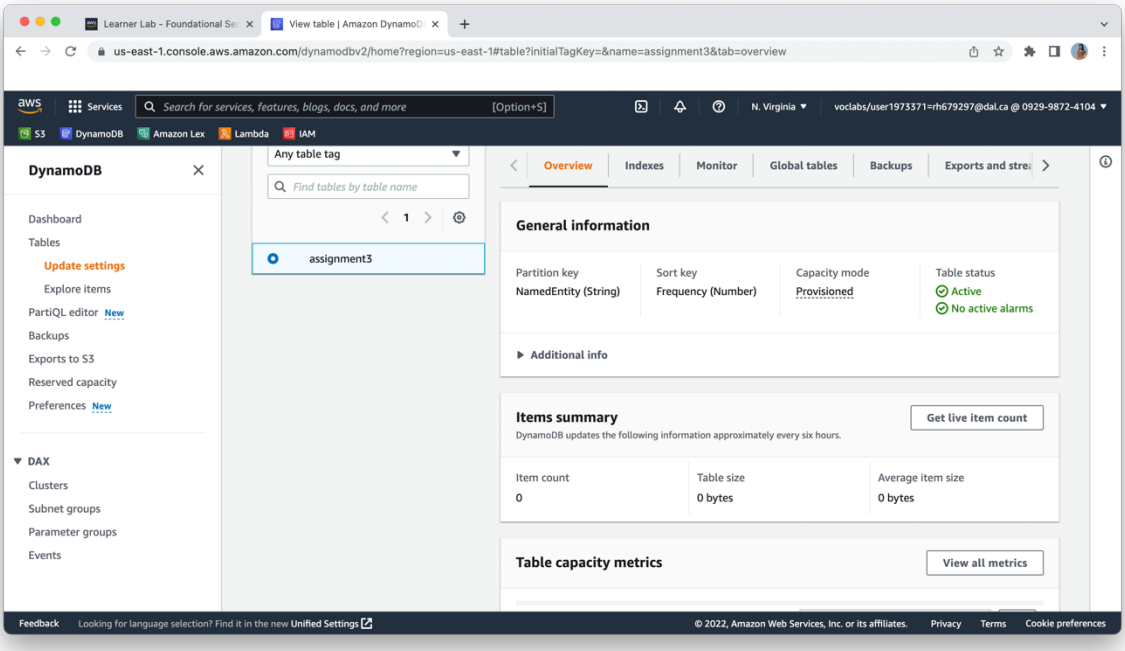

Figure 14. *Screenshot of EMPTY table assignment3 in DynamoDB console.*

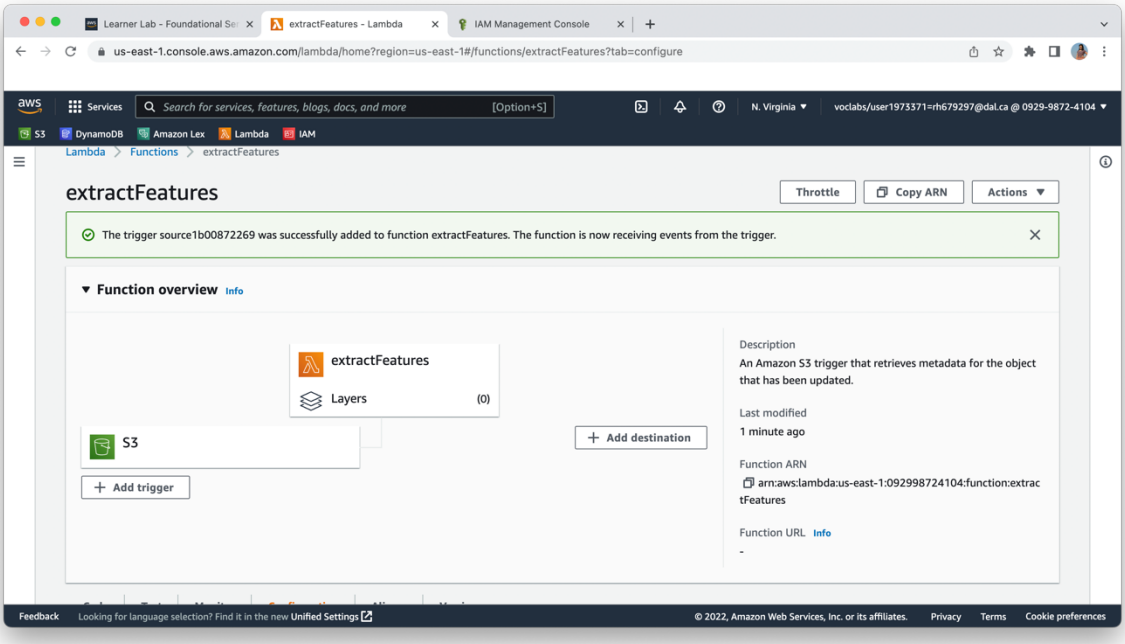

Figure 15. *Screenshot of SUCCESSFUL creation of* ***extractFeatures*** *lambda function with trigger set to S3.*

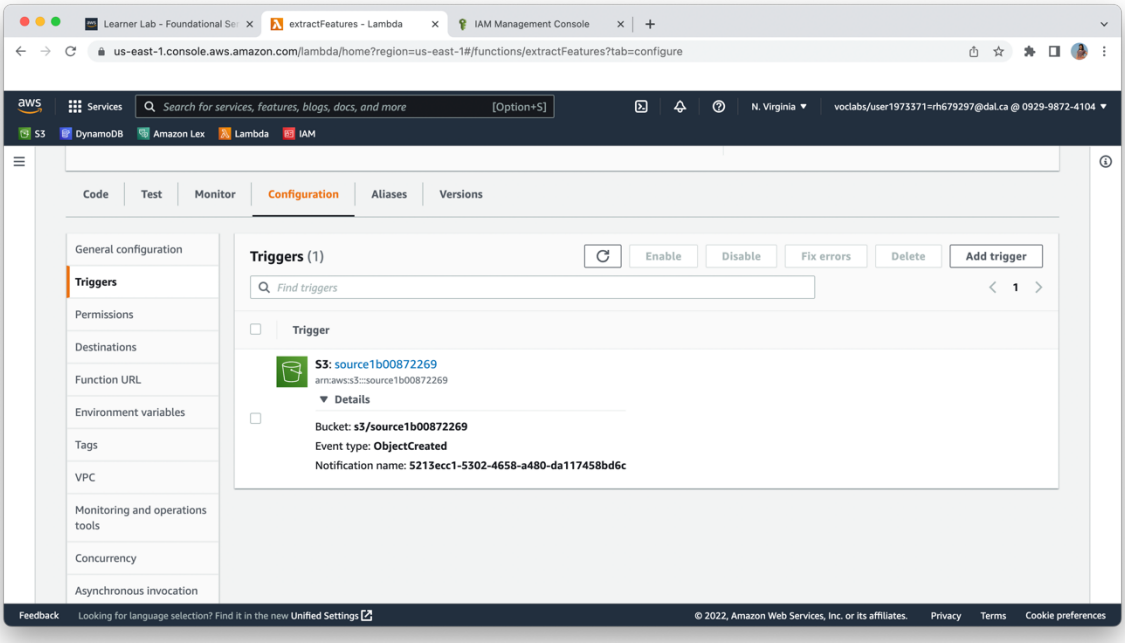

Figure 16. *Screenshot of SUCCESSFUL creation of* ***extractFeatures*** *lambda function with S3 resource set to* ***source1b00872269****.*

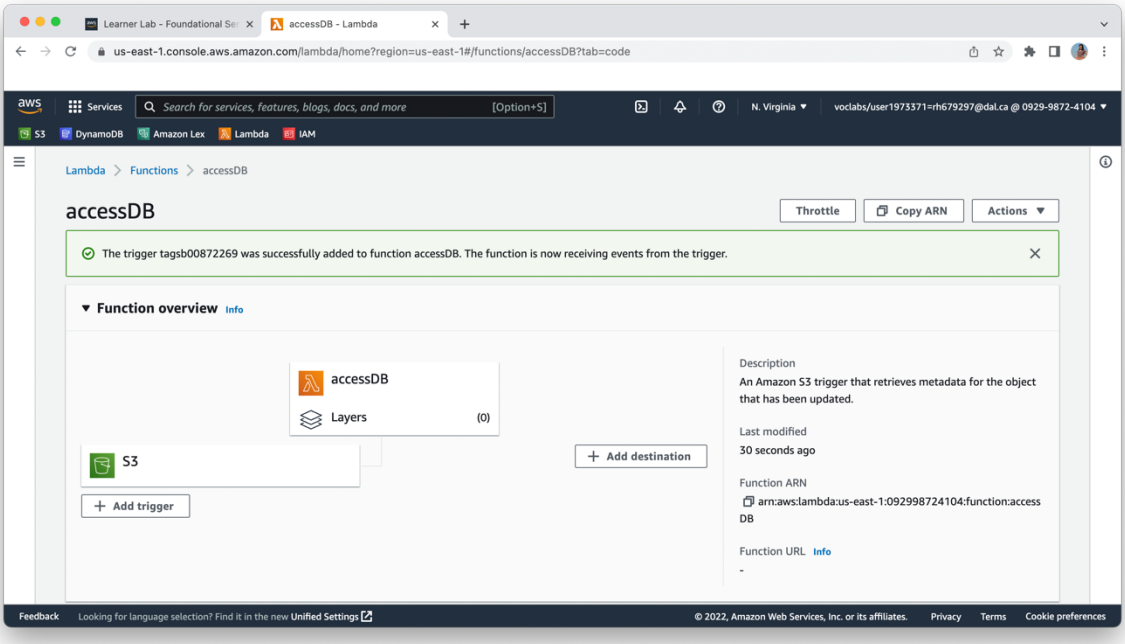

Figure 17. *Screenshot of SUCCESSFUL creation of* ***accessDB*** *lambda function with trigger set to S3.*

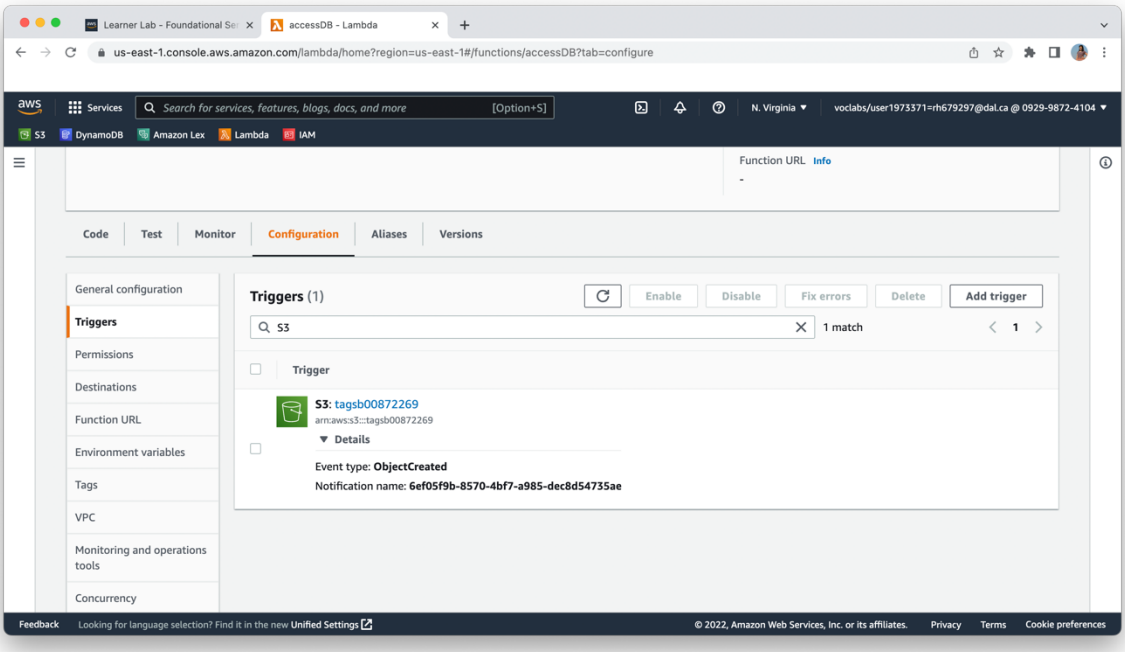

Figure 18. *Screenshot of SUCCESSFUL creation of* ***accessDB*** *lambda function with S3 resource set to* ***tagsb00872269****.*

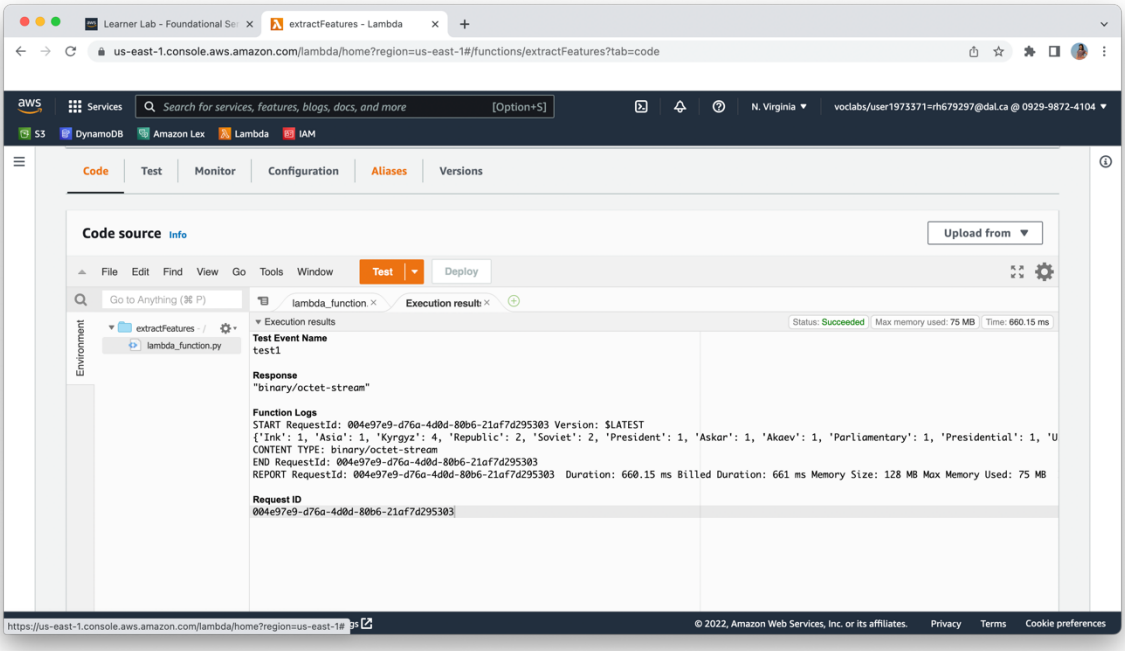

Figure 19. *Screenshot of SUCCESSFUL test run of* ***extractFeatures*** *lambda function.*

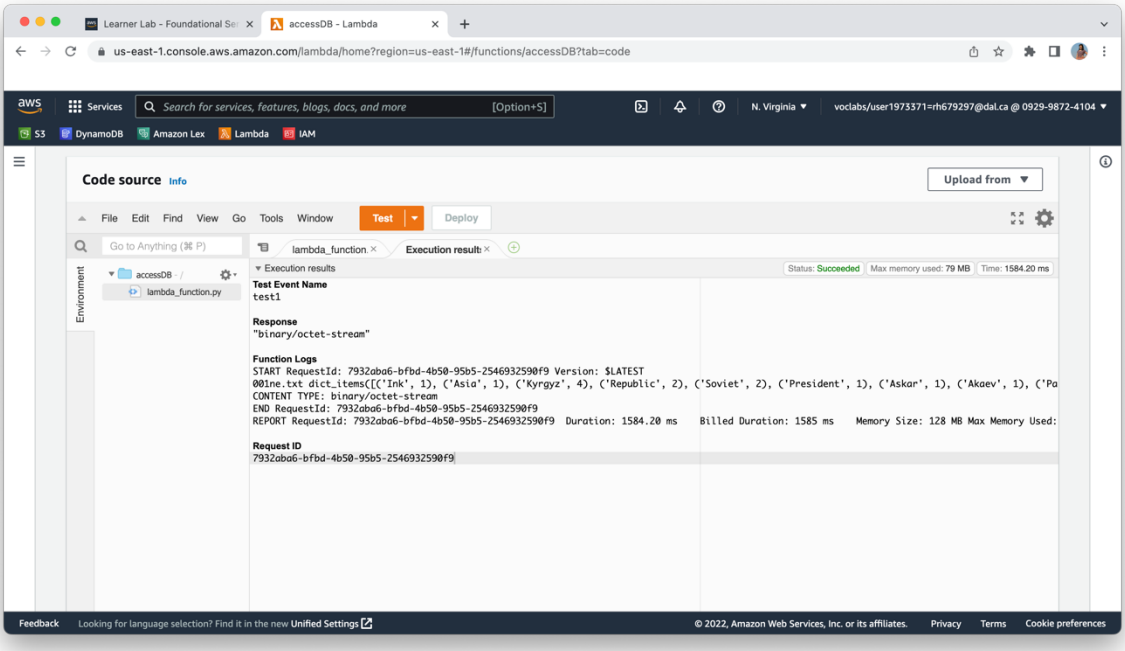

Figure 20. *Screenshot of SUCCESSFUL test run of* ***accessDB*** *lambda function.*

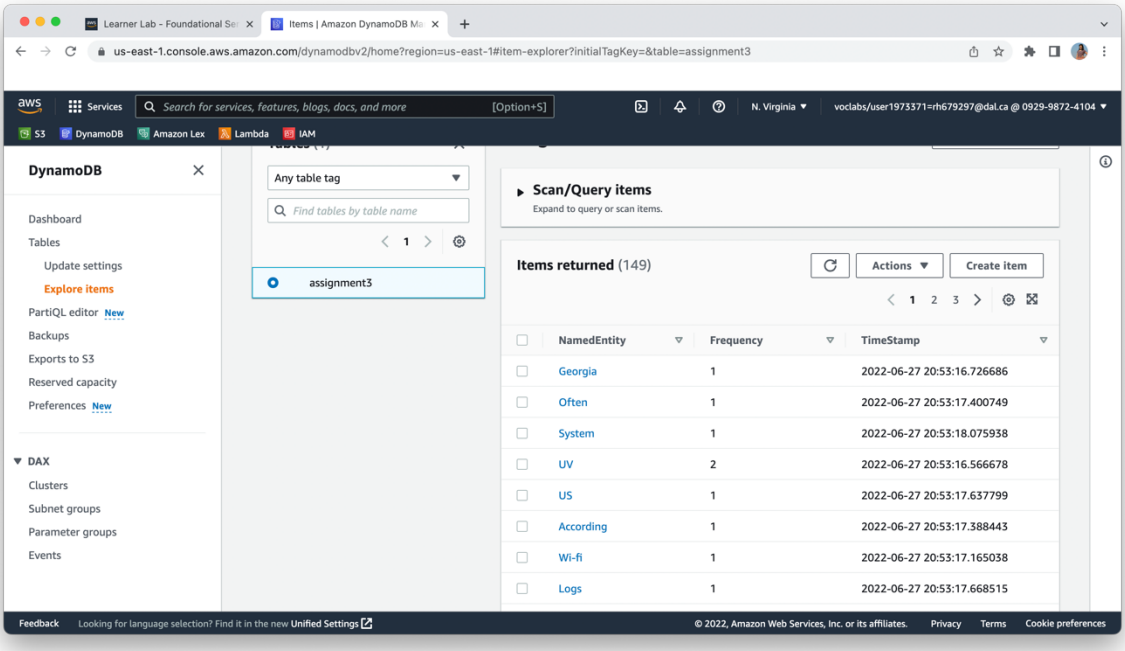

Figure 21. *Screenshot of SUCCESSFUL update to DynamoDB table-1 .*

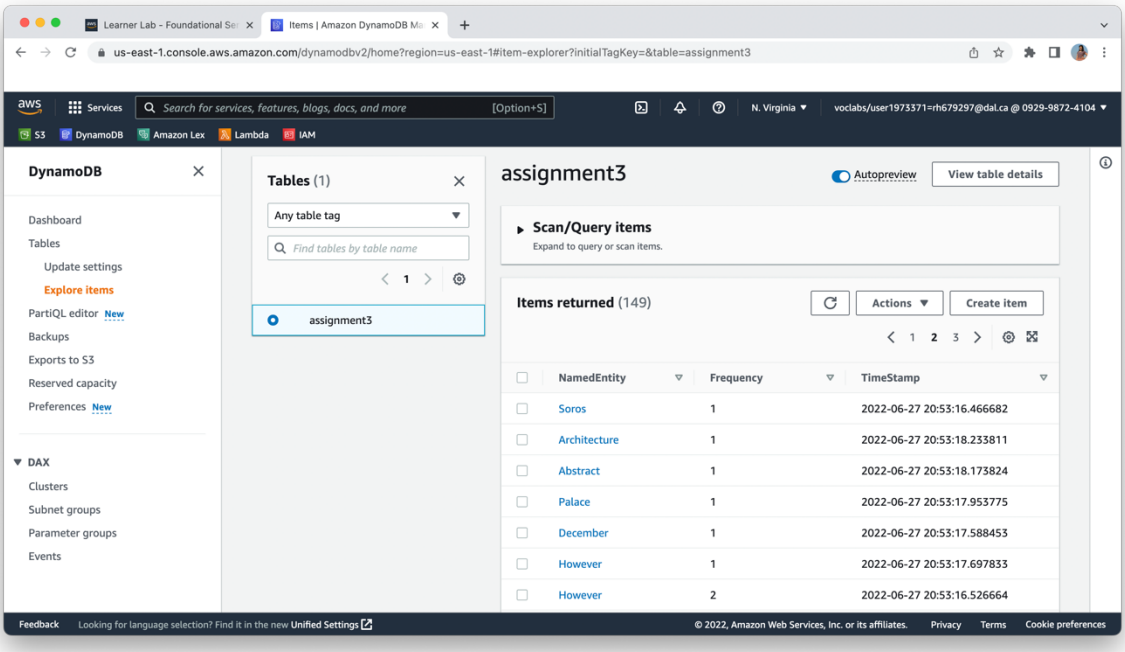

Figure 22. *Screenshot of SUCCESSFUL update to DynamoDB table-2.*

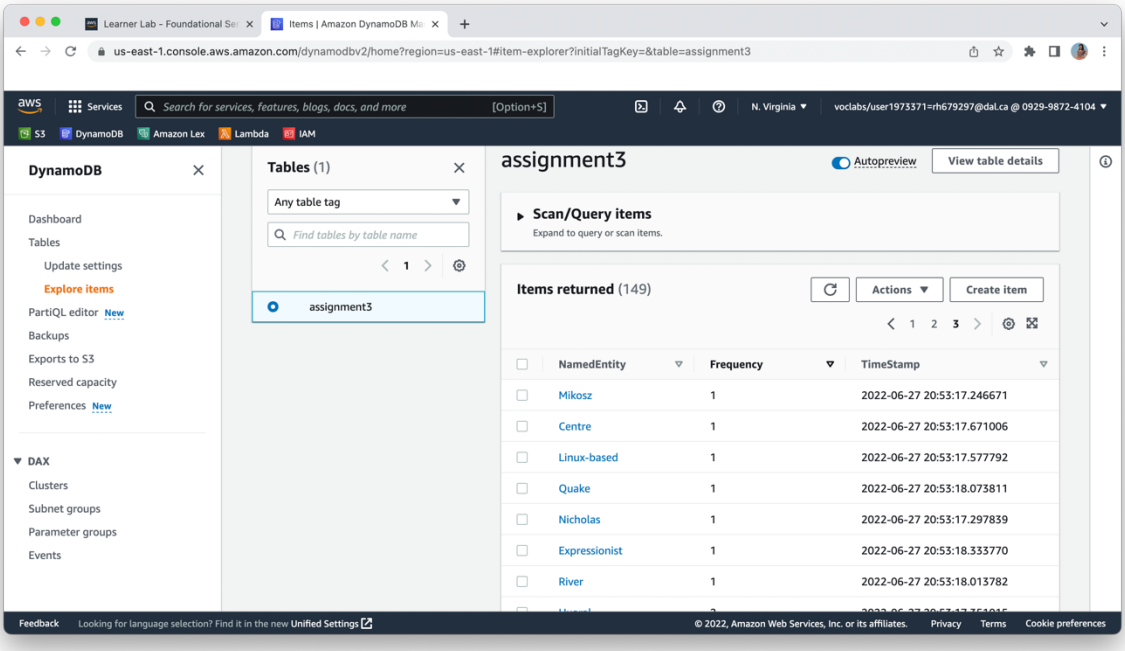

Figure 23. *Screenshot of SUCCESSFUL update to DynamoDB table-3 .*

Step 3: extractFeatures.py lambda function script [2][3][4]:

```
import json
import urllib.parse
import boto3
import re

print('Loading function')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    # print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key =
urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        var1 = response["Body"].read().decode('utf-8')
        dictionary = {}
        stopwords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours',
'ourselves', 'you', "you're", "you've", "you'll",
                     "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's",
                     'her', 'hers', 'herself', 'it', "it's", 'its',
'itself', 'they', 'them', 'their', 'theirs',
                     'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', 'am',
                     'is', 'are', 'was', 'were', 'be', 'been', 'being',
'have', 'has', 'had', 'having', 'do', 'does',
                     'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if',
'or', 'because', 'as', 'until', 'while',
                     'of', 'at', 'by', 'for', 'with', 'about', 'against',
'between', 'into', 'through', 'during',
                     'before', 'after', 'above', 'below', 'to', 'from',
'up', 'down', 'in', 'out', 'on', 'off', 'over',
                     'under', 'again', 'further', 'then', 'once', 'here',
'there', 'when', 'where', 'why', 'how', 'all',
                     'any', 'both', 'each', 'few', 'more', 'most', 'other',
'some', 'such', 'no', 'nor', 'not', 'only',
                     'own', 'same', 'so', 'than', 'too', 'very', 's', 't',
'can', 'will', 'just', 'don', "don't",
                     'should', "should've", 'now', 'd', 'll', 'm', 'o',
're', 've', 'y', 'ain', 'aren', "aren't",
                     'couldn', "couldn't", 'didn', "didn't", 'doesn',
"doesn't", 'hadn', "hadn't", 'hasn', "hasn't",
                     'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn', "mustn't", 'needn',
                     "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
'wasn', "wasn't", 'weren', "weren't", 'won',
                     "won't", 'wouldn', "wouldn't"]

        for word in var1.split():
            if word.lower() not in stopwords and word[0].isupper():
                w = re.sub("[()\".,']", "", word)

                if w in dictionary:
```

```
                    dictionary[w] += 1
                else:
                    dictionary[w] = 1
        print(dictionary)

        actual_file_name = key.split(".")[0]
        file_name = f"/tmp/{actual_file_name}ne.txt"
        file = open(file_name, 'w+')
        file.write(json.dumps(dictionary))
        file.close()

        s3.upload_file(file_name, 'tagsb00872269',
f'{actual_file_name}ne.txt')
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print(
            'Error getting object {} from bucket {}. Make sure they exist
and your bucket is in the same region as this function.'.format(
                key, bucket))
        raise e
```

Step 4: accessDB.py lambda function script [2][3][4][5]:

```
import json
import urllib.parse
import boto3
from datetime import datetime

print('Loading function')

s3 = boto3.client('s3')
dynamodb = boto3.client('dynamodb')


def lambda_handler(event, context):
    # print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key =
urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        var1 = response["Body"].read().decode('utf-8')
        json_data = json.loads(var1)
        key1 = key.split(":")[0]
        print(key1, json_data.items())
        for k, v in json_data.items():
            dynamodb.put_item(TableName='assignment3', Item={'NamedEntity':
{'S': k}, 'Frequency': {'N': str(v)},
                                                             'TimeStamp':
{'S': str(datetime.now())}})
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print(
            'Error getting object {} from bucket {}. Make sure they exist
and your bucket is in the same region as this function.'.format(
                key, bucket))
        raise e
```

REFERENCES:

[1] "Amazon S3 buckets — Boto3 Docs 1.24.17 documentation", *Boto3.amazonaws.com*, 2022. [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-example-creating-buckets.html. [Accessed: 24- Jun- 2022]

[2] "S3 — Boto3 Docs 1.24.16 documentation", Boto3.amazonaws.com, 2022. [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html. [Accessed: 25- Jun- 2022]

[3] "Tutorial: Using an Amazon S3 trigger to invoke a Lambda function", docs.aws.amazon.com, 2022. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html. [Accessed: 25-Jun- 2022]

[4] "NLTK :: Natural Language Toolkit", Nltk.org, 2022. [Online]. Available: https://www.nltk.org/. [Accessed: 25- Jun- 2022]

[5] "DynamoDB — Boto3 Docs 1.24.16 documentation", Boto3.amazonaws.com, 2022.[Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html. [Accessed: 25- Jun- 2022]