Name: Pradeep Medagiri

Homework 10: Lambda

You are building an information system for a startup financial company in Northern Virginia. It needs a database to store companies' financial information, such as their stock price, industry they are in, and some other information (some companies have more information than others). This database is intended for the customers so that they can **query** information quickly using company's **ticker**. You decide to use the NoSQL and fast AWS **DynamoDB** for the database.

As part of their work process, a staff will research the financial information of a company, create a json file to record the data, and upload the json file in S3. One company per json file.

The company wants to implement Robotic Process Automation (RPA). In other words, they want to automate repetitive activities with software. So instead of populating the database manually, they want the system to automatically read financial data every time a json file is uploaded to the S3 bucket, and to populate the data into the database accordingly.

You need to create a lambda function (similar to the demo) to automatically read uploaded json file and populate the dynamodb database.

You have data of 4 companies and you need to create a json file for each of them for testing purposes (all numbers are integers to simplify things).

Ticker: AAPL
Stock_price: 389
Industry: Computer
Name: Apple
Volume: 20279276

Ticker: IBM
Stock_price: 129
Name: International Business Machines

Ticker: HPE
Stock_price: 10
Industry: Computer
Name: Hewlett Packard Enterprise

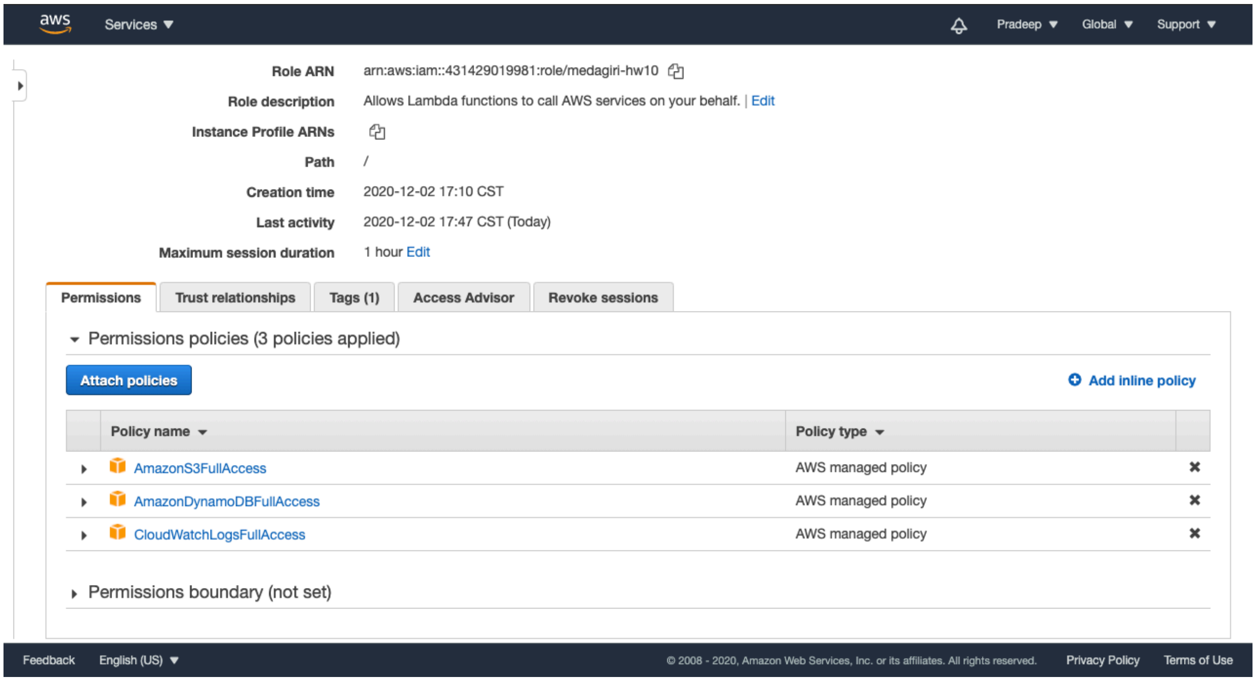
Ticker: TSLA
Stock_price: 1592
Industry: Automobile
Name: Tesla
Volume: 14161080
State: California
City: San Francisco

Your tasks:

1. Create a dynamodb table – name it yourlastname-hw10-dynamodb.
2. Create an S3 bucket with private access to store the json files.
3. Create a json file for each of the companies listed above. Name the each of the json file using the ticker. For example, the json file for the apple is AAPL.json
4. Create a lambda function that will read json file uploaded to the S3 and automatically populate the dynamodb table.
5. Upload all four json files. All four companies must be in the dynamodb database.

What to submit:

1. The screenshot of the IAM Role you created that shows all its policies. (16%)



Role ARN arn:aws:iam::431429019981:role/medagiri-hw10

Role description Allows Lambda functions to call AWS services on your behalf. | Edit

Instance Profile ARNs

Path /

Creation time 2020-12-02 17:10 CST

Last activity 2020-12-02 17:47 CST (Today)

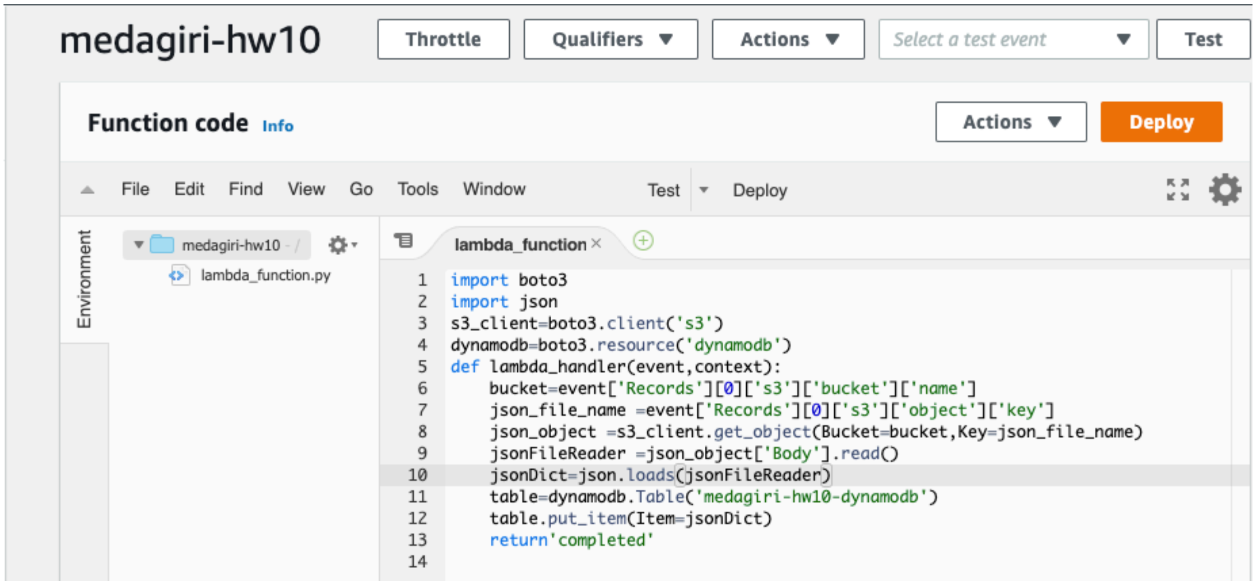
Maximum session duration 1 hour Edit

Permissions policies (3 policies applied)

Policy name	Policy type	
AmazonS3FullAccess	AWS managed policy	✖
AmazonDynamoDBFullAccess	AWS managed policy	✖
CloudWatchLogsFullAccess	AWS managed policy	✖

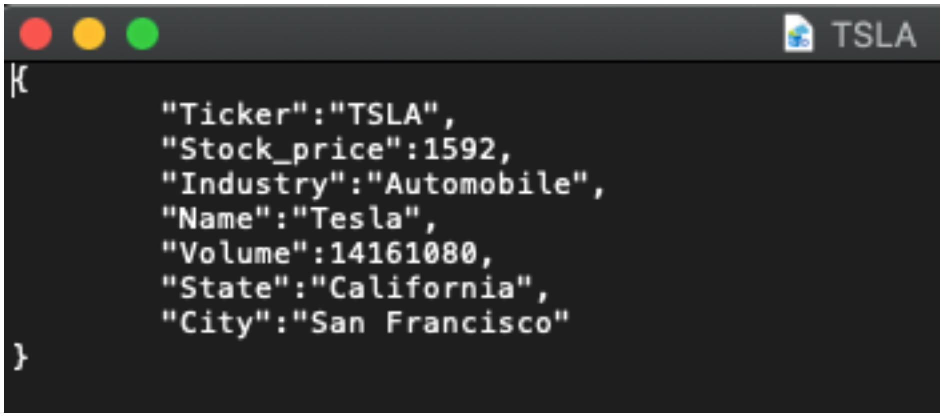
Permissions boundary (not set)

2. The screenshot of the final working lambda function. (20%)



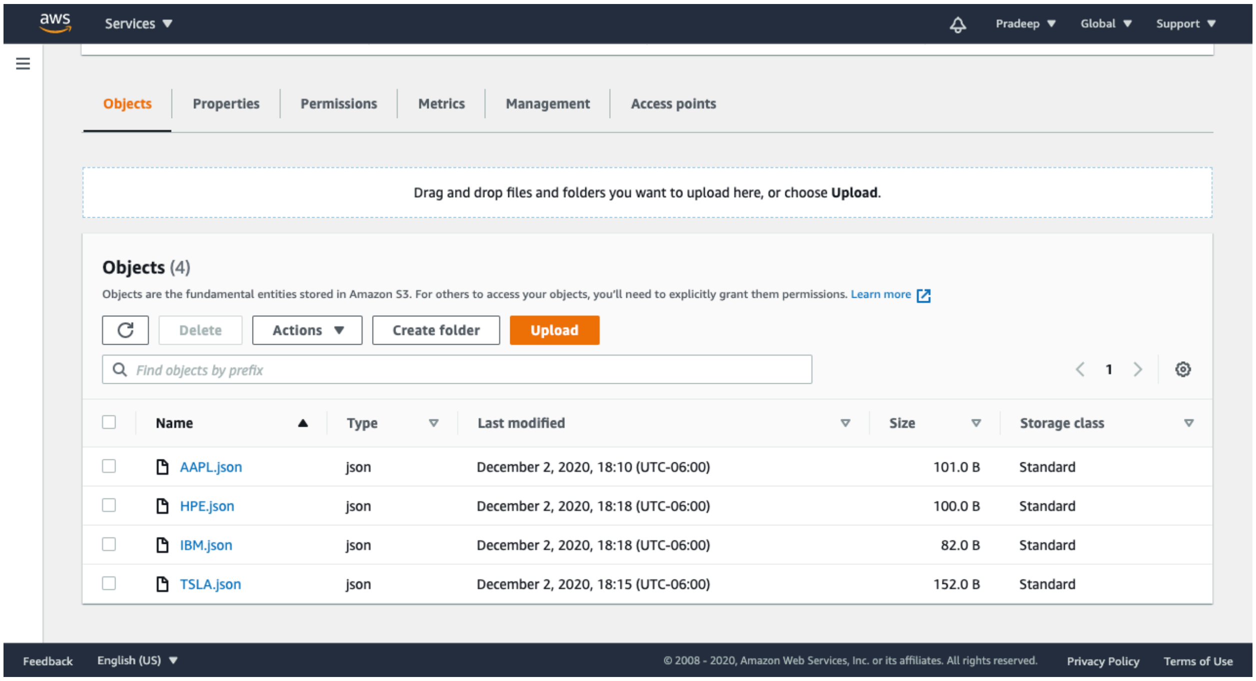
```python
import boto3
import json
s3_client=boto3.client('s3')
dynamodb=boto3.resource('dynamodb')
def lambda_handler(event,context):
    bucket=event['Records'][0]['s3']['bucket']['name']
    json_file_name =event['Records'][0]['s3']['object']['key']
    json_object =s3_client.get_object(Bucket=bucket,Key=json_file_name)
    jsonFileReader =json_object['Body'].read()
    jsonDict=json.loads(jsonFileReader)
    table=dynamodb.Table('medagiri-hw10-dynamodb')
    table.put_item(Item=jsonDict)
    return'completed'

```

3. The screenshot of Tesla json file. (16%)



```
{
        "Ticker":"TSLA",
        "Stock_price":1592,
        "Industry":"Automobile",
        "Name":"Tesla",
        "Volume":14161080,
        "State":"California",
        "City":"San Francisco"
}
```

4. The screenshot of the S3 bucket with all the json files uploaded. (16%)



Drag and drop files and folders you want to upload here, or choose **Upload**.

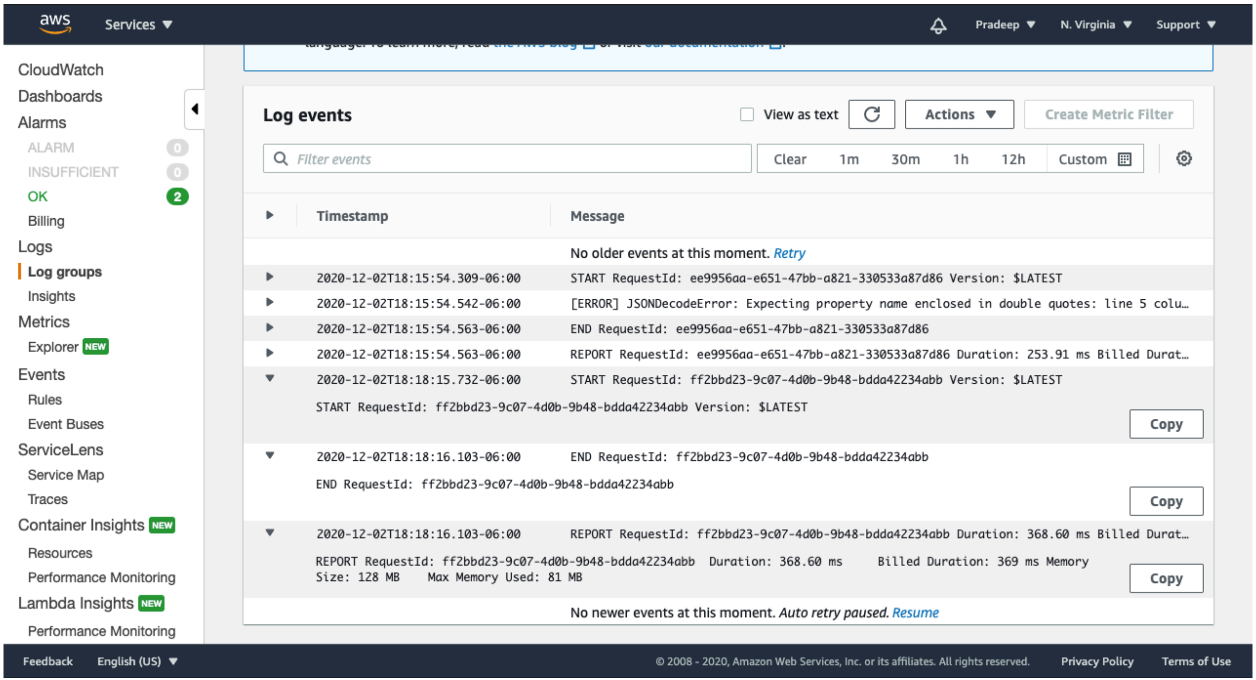
Objects (4)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

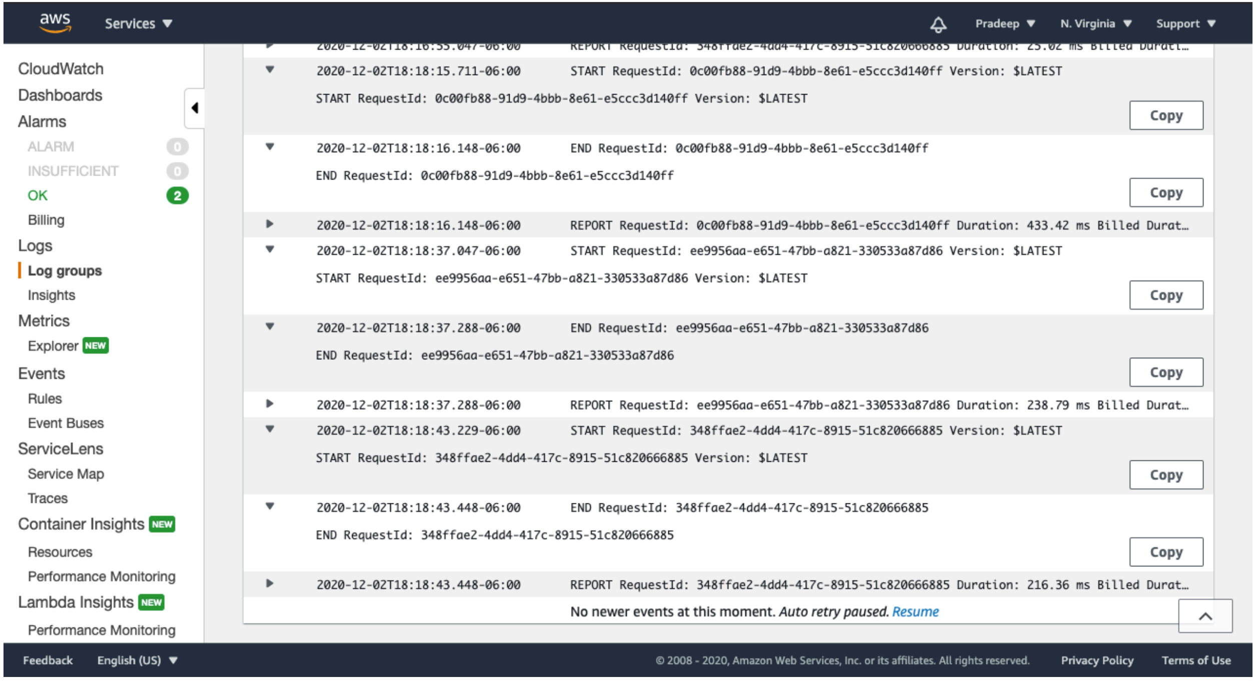
□	Name ▲	Type	Last modified	Size	Storage class
□	AAPL.json	json	December 2, 2020, 18:10 (UTC-06:00)	101.0 B	Standard
□	HPE.json	json	December 2, 2020, 18:18 (UTC-06:00)	100.0 B	Standard
□	IBM.json	json	December 2, 2020, 18:18 (UTC-06:00)	82.0 B	Standard
□	TSLA.json	json	December 2, 2020, 18:15 (UTC-06:00)	152.0 B	Standard

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

5. The screenshot CloudWatch log groups for this lambda function that shows no errors (after all 4 json files have been uploaded). (16%)

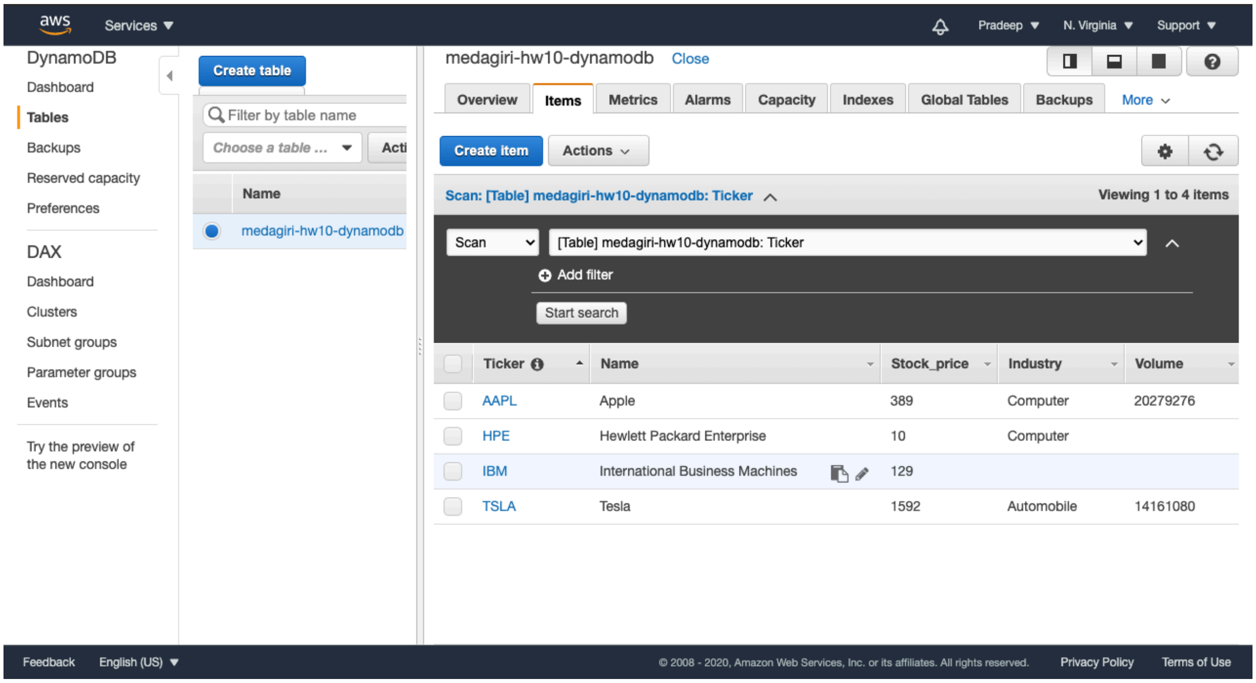


```
No older events at this moment. Retry
2020-12-02T18:15:54.309-06:00   START RequestId: ee9956aa-e651-47bb-a821-330533a87d86 Version: $LATEST
2020-12-02T18:15:54.542-06:00   [ERROR] JSONDecodeError: Expecting property name enclosed in double quotes: line 5 colu…
2020-12-02T18:15:54.563-06:00   END RequestId: ee9956aa-e651-47bb-a821-330533a87d86
2020-12-02T18:15:54.563-06:00   REPORT RequestId: ee9956aa-e651-47bb-a821-330533a87d86 Duration: 253.91 ms Billed Durat…
2020-12-02T18:18:15.732-06:00   START RequestId: ff2bbd23-9c07-4d0b-9b48-bdda42234abb Version: $LATEST
START RequestId: ff2bbd23-9c07-4d0b-9b48-bdda42234abb Version: $LATEST
2020-12-02T18:18:16.103-06:00   END RequestId: ff2bbd23-9c07-4d0b-9b48-bdda42234abb
END RequestId: ff2bbd23-9c07-4d0b-9b48-bdda42234abb
2020-12-02T18:18:16.103-06:00   REPORT RequestId: ff2bbd23-9c07-4d0b-9b48-bdda42234abb Duration: 368.60 ms Billed Durat…
REPORT RequestId: ff2bbd23-9c07-4d0b-9b48-bdda42234abb  Duration: 368.60 ms     Billed Duration: 369 ms Memory
Size: 128 MB    Max Memory Used: 81 MB
No newer events at this moment. Auto retry paused. Resume
```



```
2020-12-02T18:16:55.047-06:00   REPORT RequestId: 348ffae2-4dd4-417c-8915-51c820666885 Duration: 25.02 ms Billed Durat…
2020-12-02T18:18:15.711-06:00   START RequestId: 0c00fb88-91d9-4bbb-8e61-e5ccc3d140ff Version: $LATEST
START RequestId: 0c00fb88-91d9-4bbb-8e61-e5ccc3d140ff Version: $LATEST
2020-12-02T18:18:16.148-06:00   END RequestId: 0c00fb88-91d9-4bbb-8e61-e5ccc3d140ff
END RequestId: 0c00fb88-91d9-4bbb-8e61-e5ccc3d140ff
2020-12-02T18:18:16.148-06:00   REPORT RequestId: 0c00fb88-91d9-4bbb-8e61-e5ccc3d140ff Duration: 433.42 ms Billed Durat…
2020-12-02T18:18:37.047-06:00   START RequestId: ee9956aa-e651-47bb-a821-330533a87d86 Version: $LATEST
START RequestId: ee9956aa-e651-47bb-a821-330533a87d86 Version: $LATEST
2020-12-02T18:18:37.288-06:00   END RequestId: ee9956aa-e651-47bb-a821-330533a87d86
END RequestId: ee9956aa-e651-47bb-a821-330533a87d86
2020-12-02T18:18:37.288-06:00   REPORT RequestId: ee9956aa-e651-47bb-a821-330533a87d86 Duration: 238.79 ms Billed Durat…
2020-12-02T18:18:43.229-06:00   START RequestId: 348ffae2-4dd4-417c-8915-51c820666885 Version: $LATEST
START RequestId: 348ffae2-4dd4-417c-8915-51c820666885 Version: $LATEST
2020-12-02T18:18:43.448-06:00   END RequestId: 348ffae2-4dd4-417c-8915-51c820666885
END RequestId: 348ffae2-4dd4-417c-8915-51c820666885
2020-12-02T18:18:43.448-06:00   REPORT RequestId: 348ffae2-4dd4-417c-8915-51c820666885 Duration: 216.36 ms Billed Durat…
No newer events at this moment. Auto retry paused. Resume
```

6. The screenshot of the items in the **dynamodb table (after all 4 json files have been uploaded)**. (16%)



Scan: [Table] medagiri-hw10-dynamodb: Ticker

Viewing 1 to 4 items

Scan [Table] medagiri-hw10-dynamodb: Ticker

Ticker	Name	Stock_price	Industry	Volume
AAPL	Apple	389	Computer	20279276
HPE	Hewlett Packard Enterprise	10	Computer	
IBM	International Business Machines	129		
TSLA	Tesla	1592	Automobile	14161080