

## **Software Engineer for Cloud Project 1**

### **Instructions**

Every screenshot requested in this workbook is compulsory and carries 5 marks. Updated Python Script carries 20 marks. Lambda Function Code In Python carries 25 marks.

Your AWS account ID must be clearly visible in every screenshot using the AWS console; missing id or using someone else's id is not permitted. Such cases will be considered as plagiarism and severe penalty will be imposed.

All screenshots must be in the order mentioned under "Expected Screenshots" for every step

DO NOT WAIT UNTIL THE LAST MINUTE. The program office will not extend the project submission deadline under any circumstances.

The file should be renamed in the format BATCH\_FIRSTNAME\_LASTNAME\_PROJECT1.  
For example: ACSEOCT20\_VIJAY\_DWIVEDI\_PROJECT1.docx

### **Resource Clean Up**

Cloud is always pay per use model and all resources/services that we consume are chargeable. Cleaning up when you've completed your lab or project is always necessary. This is true whether you're doing a lab or implementing a project at your workplace.

After completing the lab, make sure to delete each resource created in reverse chronological order.

### **Submission Files -**

You need to submit the following -

1. Project workbook with added screenshots.
2. Updated Python script (StockPriceIngestion.py)
3. Lambda Function Code In Python

### **Problem Statement :**

We want to build a system that streams stock pricing information for various stocks and then notifies the stakeholders about the fetched information of the stocks.

We'll use the Yahoo Finance APIs to query the running price of stocks and general information like 52-week high/low values.

**Yahoo Finance API** - It provides functions to download historical market data from Yahoo! finance. While this functionality in production would generally run on a paid api that provides real-time stock price data, we'll mimic it by using historical data for an older time period and streaming it over kinesis.

**Link** - <https://pypi.org/project/yfinance/>

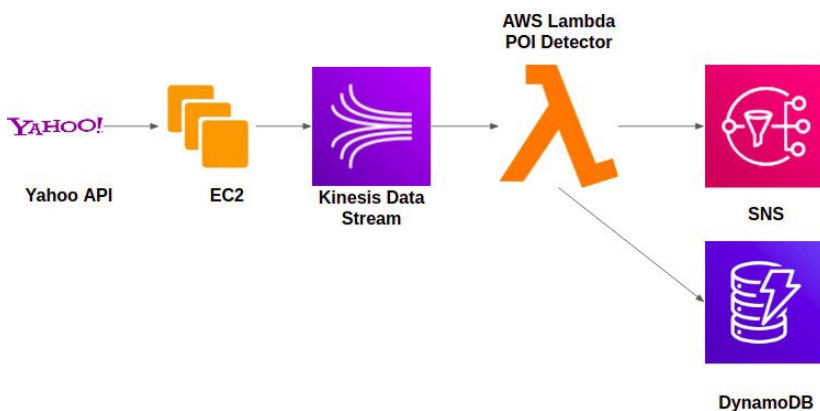
Please go through this link to understand how to access different stocks data and information.

You need to pull the data for the following 10 stocks -

MSFT, MVIS, GOOG, SPOT, INO, OCGN, ABML, RLLCF, JNJ, PSFE

You can check the details of these stocks here - <https://finance.yahoo.com/lookup/>

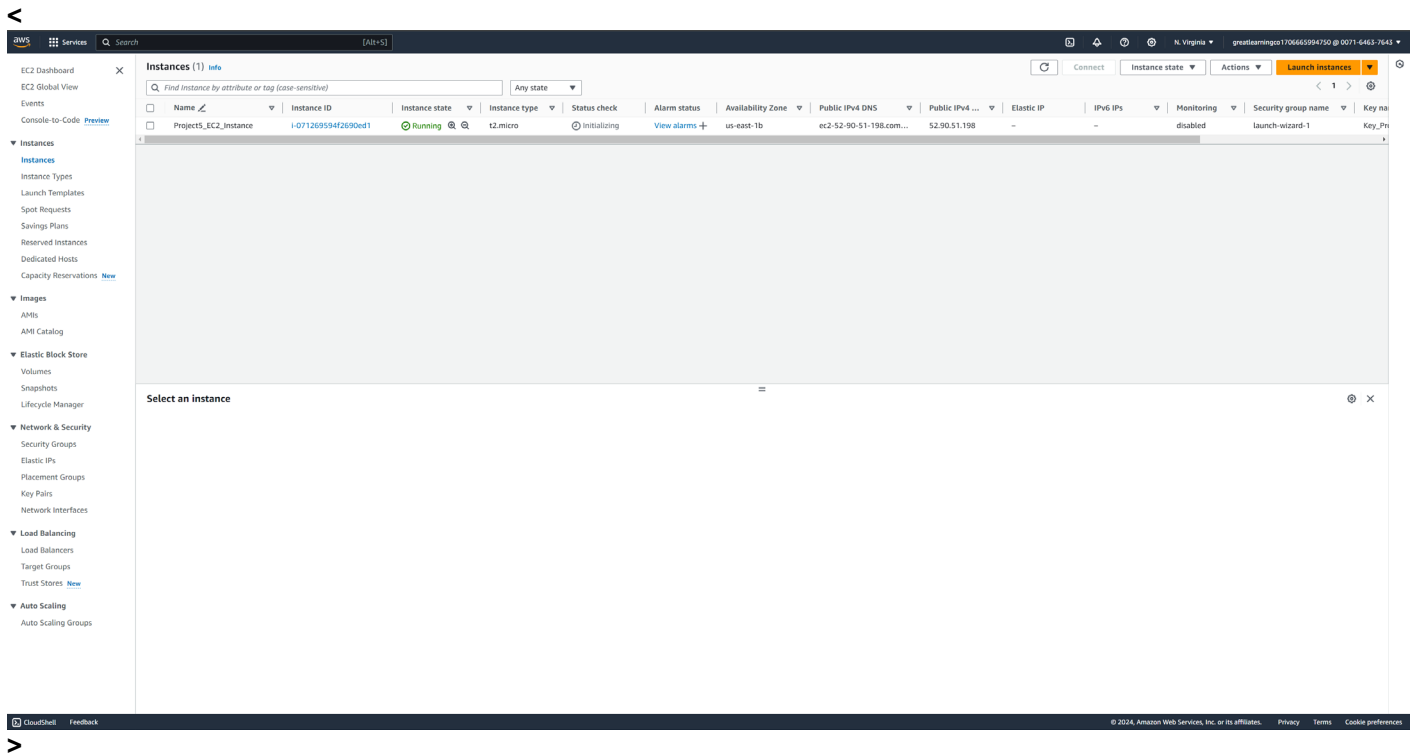
### Architecture diagram



Architecture Implementation	
1	Create EC2 Instance
2	Run Python Script to Pull data from Yahoo Stock API
3	Use boto3 and kinesis client to Push data from EC2 to Kinesis Stream
4	Configure SNS to publish the notification through mail, Configure DynamoDB to store the alert data.
5	Write a Lambda function to detect the POIs and to Push the Notification to SNS
6	Use the same Lambda Function to push data to DynamoDB

### Step 1: Create EC2 Instance

Step number	a
Step name	Create EC2 Instance
Instructions	<ol style="list-style-type: none"> <li>1) Navigate to EC2 Services from AWS management console page</li> <li>2) Choose `instances` available in the left pane.</li> <li>3) Click on `launch instances`</li> <li>4) Choose the free tier for `Amazon Linux`</li> <li>5) Choose instance type as t2.micro</li> <li>6) Configure instance details keep all the option as be default</li> <li>7) Keep default storage option and provide optional tag details</li> <li>8) Create a new security group for the instance.(Configure SSH option) Make it more secure by choosing source as MyIP.</li> <li>9) Click on the launch button, It will prompt you to create a new key pair. Provide the name and download the key for login purposes.</li> </ol>
Expected screenshots	1) Created EC2 instance running on the instances page.

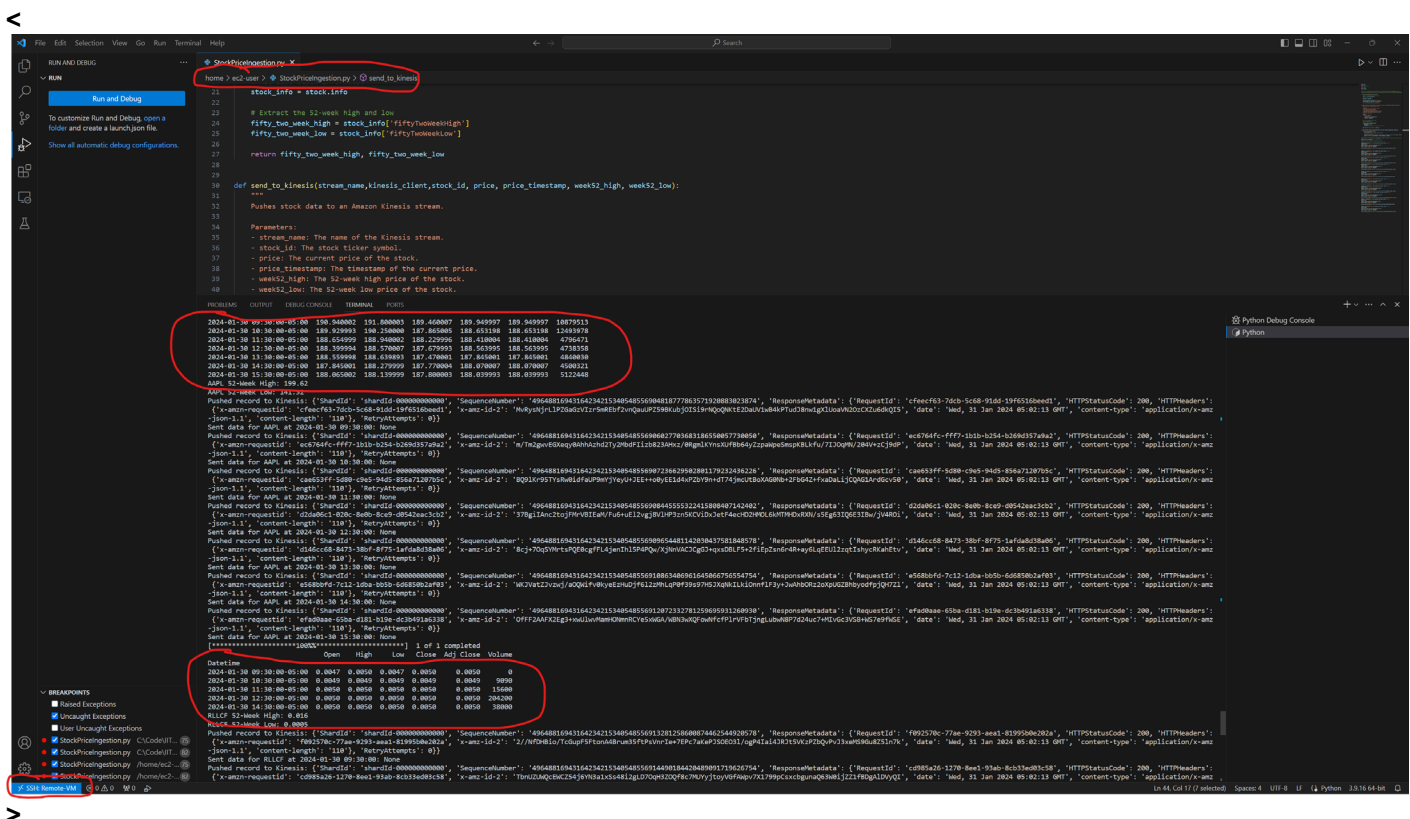




## Step 2 : Run Python Script to Pull data from Public API

Step number	a
Step name	
Instructions	<ol style="list-style-type: none"> <li>1) Your goal is to get stock information for the ten stocks specified in the doc. Please take the postMarketPrice.</li> <li>2) Further, you should call the static info api for the stocks to get their current 52WeekHigh and 52WeekLow values.</li> <li>3) Update the given python code (StockPriceIngestion.py) accordingly.</li> <li>4) Print the data on the console and run the script.</li> </ol>
Expected screenshots	<ol style="list-style-type: none"> <li>1) EC2 terminal after successful execution of the edited script. It doesn't have to show all of the data.</li> </ol>

### SCREEN OF VISUAL CODE REMOTE SSH PROGRAMMING/DEBUGGING INSIDE EC2 . LEFT BOTTOM SHOWS SSH VISUAL CODE CONNECTION



This file is meant for personal use by muratgguzel77@yahoo.com only.  
Sharing or publishing the contents in part or full is liable for legal action.

## Step 3 : Use boto3 and kinesis client to Push data from EC2 to Kinesis Stream

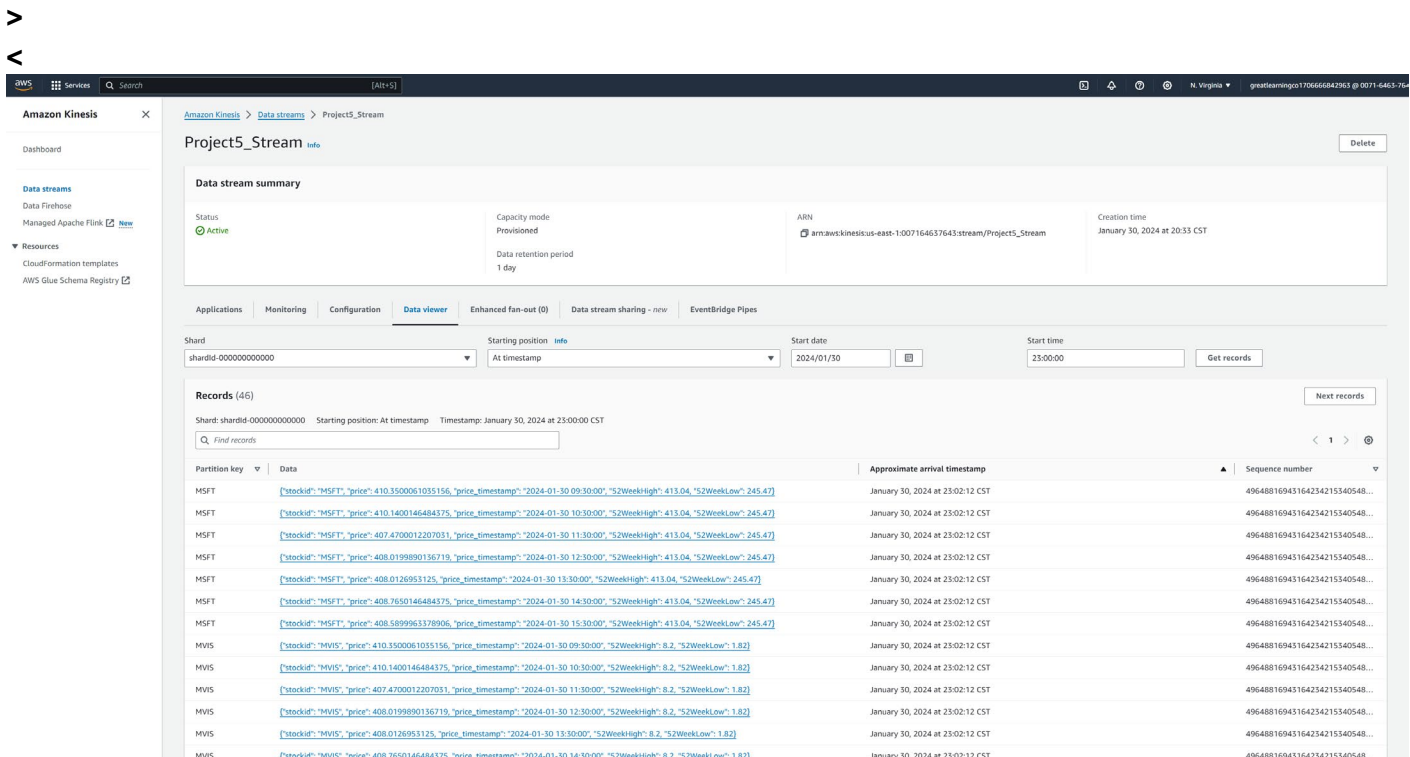
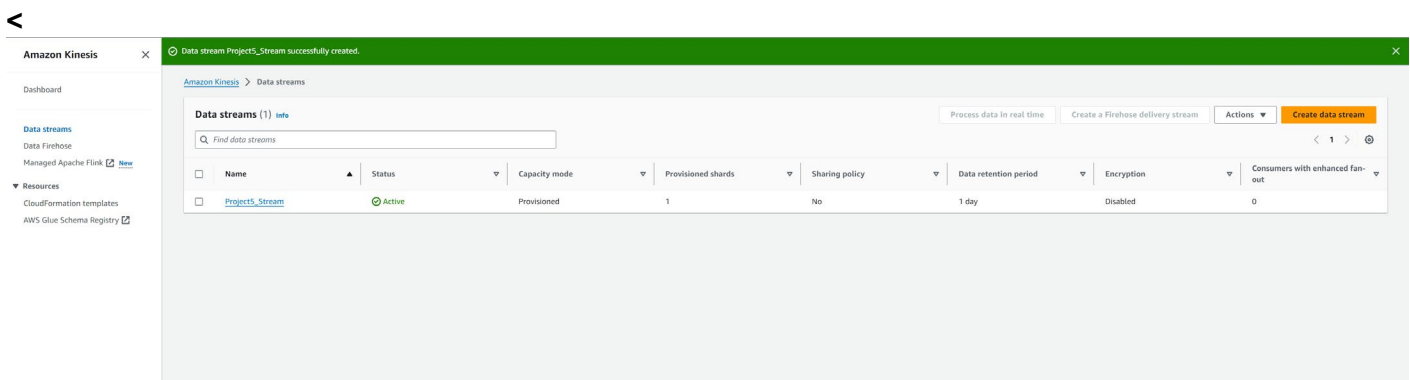
Step number	a
Step name	Data in kinesis stream

## Instructions

- 1) Go to the Kinesis stream services.
- 2) Configure the data stream to accept the data pushed by the script.
- 3) Create and attach appropriate policy and the IAM role to push the data to Kinesis stream from EC2. (helpful link - [https://docs.amazonaws.cn/en\\_us/streams/latest/dev/tutorial-stock-data-kplkcl-iam.html](https://docs.amazonaws.cn/en_us/streams/latest/dev/tutorial-stock-data-kplkcl-iam.html))
- 4) Edit the name of the created input kinesis stream inside the python script as required.
- 5) You should craft individual data records with information about the stockid, postMarketPrice, 52WeekHigh and 52WeekLow values and push them individually on the Kinesis stream.

## Expected screenshots

- 1) Screenshot of created kinesis data stream
- 2) Screenshot of the minimum 10 data points pushed on kinesis data stream (Screenshot of the EC2 terminal)

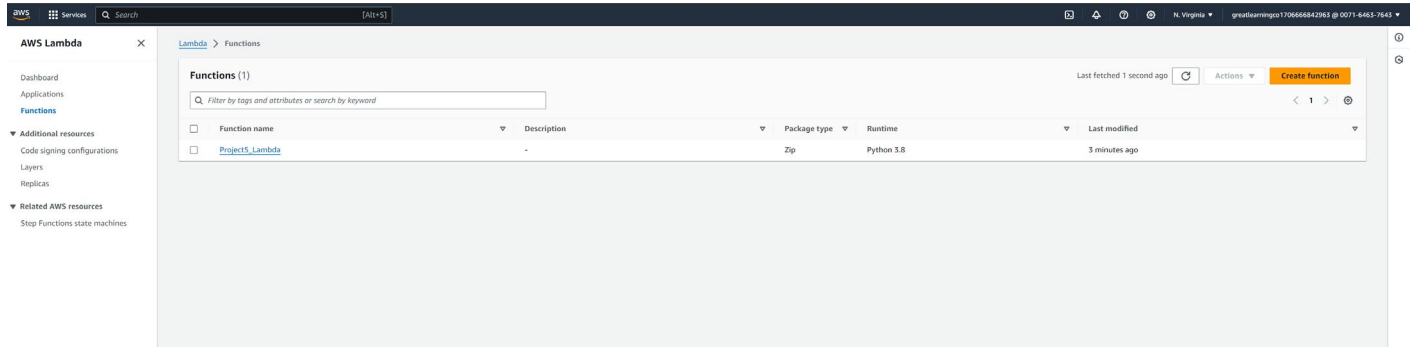


>

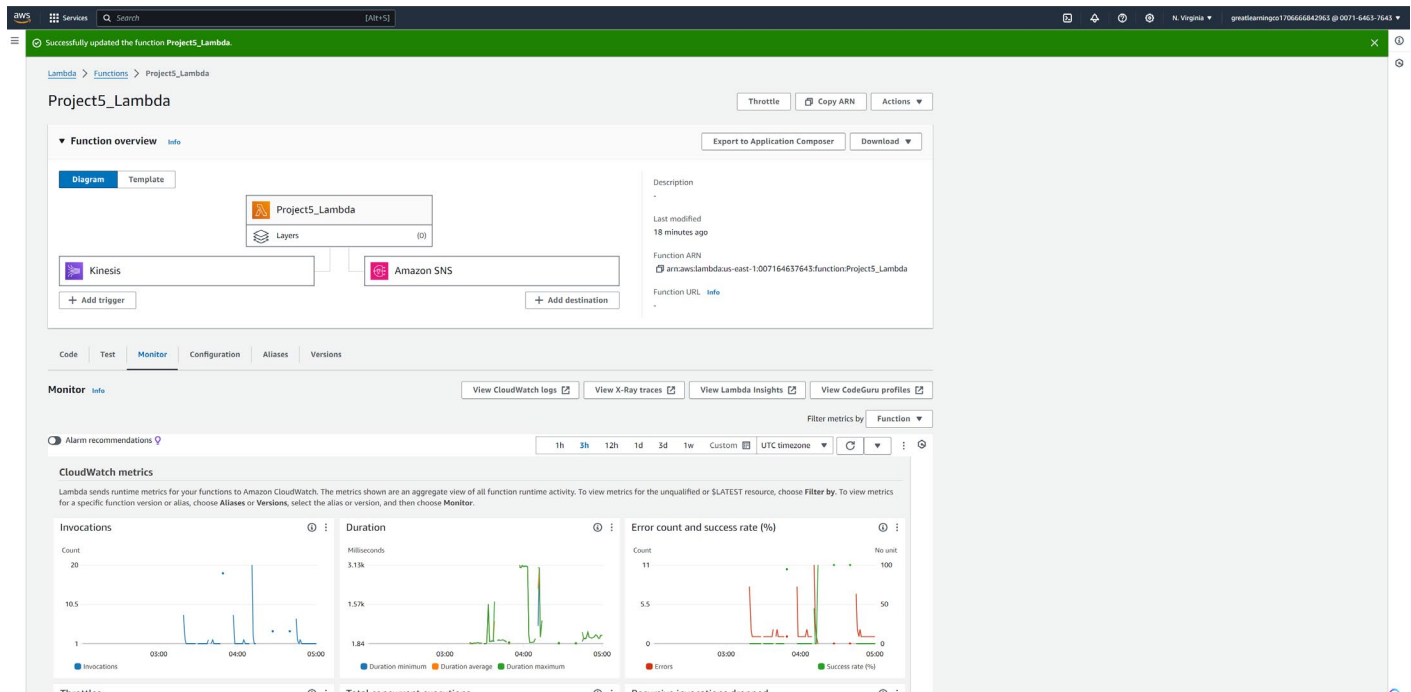
Step 4 : Write a Lambda function to get stock information and to push the Notification to SNS and add to DynamoDB

Step number	a
Step name	Write the lambda function
Instructions	<ol style="list-style-type: none"><li>1) Choose lambda services and create a lambda handler. Set it up to act as a consumer to your Kinesis data stream (<a href="https://docs.aws.amazon.com/lambda/latest/dg/with-kinesis.html">https://docs.aws.amazon.com/lambda/latest/dg/with-kinesis.html</a>)</li><li>2) For each stock, get the information and push it to SNS and dynamodb.</li></ol>
Expected screenshots	<ol style="list-style-type: none"><li>1) Created Lambda handler</li></ol>



The screenshot shows the AWS Lambda console 'Functions' page. On the left, there's a sidebar with navigation links: Dashboard, Applications, Functions (selected), Additional resources, Code signing configurations, Layers, Replicas, and Related AWS resources. The main area displays a table of functions. A search bar at the top allows filtering by tags and attributes. The table has columns for Function name, Description, Package type, Runtime, and Last modified. One function, 'Project5\_Lambda', is listed with a description of '-', package type of 'Zip', runtime of 'Python 3.8', and last modified '3 minutes ago'. A 'Create function' button is visible in the top right.

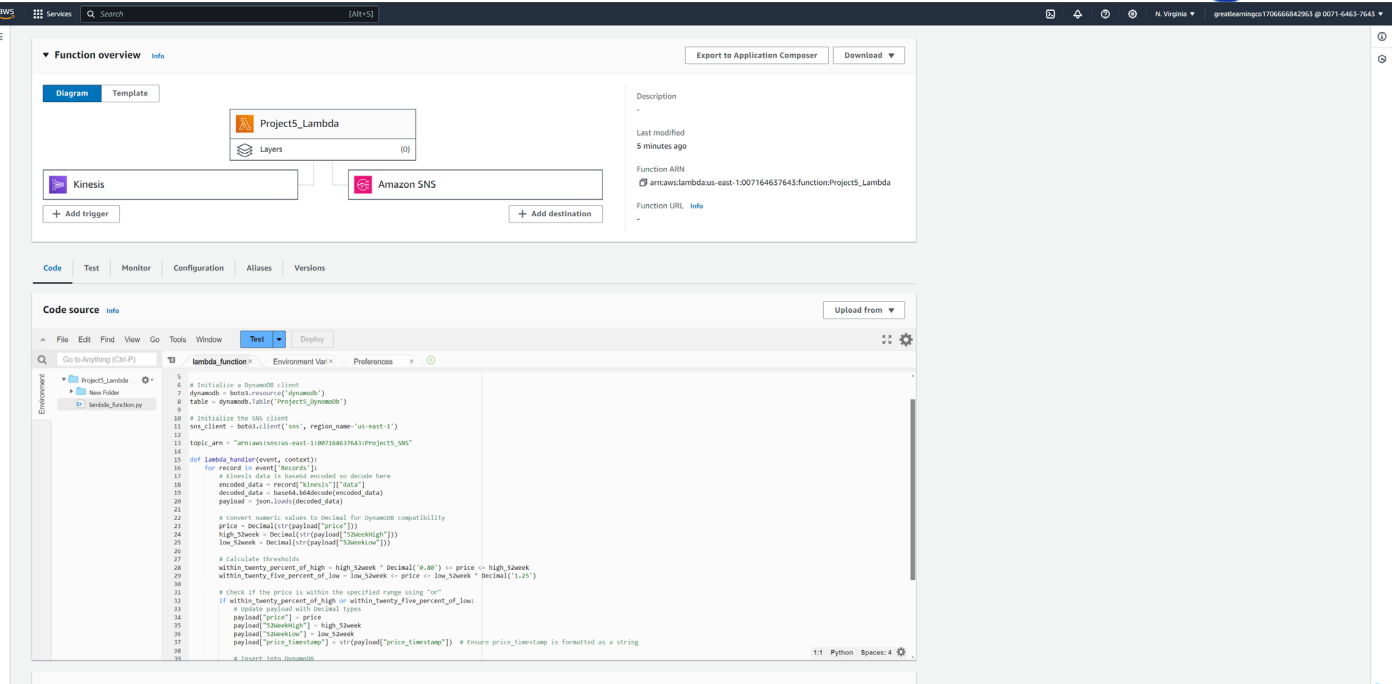


This screenshot shows the detailed view of the 'Project5\_Lambda' function in the AWS Lambda console. A green banner at the top indicates 'Successfully updated the function Project5\_Lambda'. The page is divided into several sections:
 

- Function overview:** Includes a diagram showing the function connected to Kinesis and Amazon SNS triggers. It also lists layers (0), description, last modified time (18 minutes ago), function ARN, and function URL.
- Monitor:** This section contains three line graphs:
  - Invocations:** Shows the count of function invocations over time, with a peak around 04:00.
  - Duration:** Shows the duration of function execution in milliseconds, with a peak around 04:00.
  - Error count and success rate (%):** Shows the number of errors (red line) and the success rate percentage (green line) over time.

 The bottom of the page shows tabs for 'Throttles', 'Test recent invocations', and 'Recent invocations failed'.





**Function overview**

Project\_Lambda

Kinesis

Amazon SNS

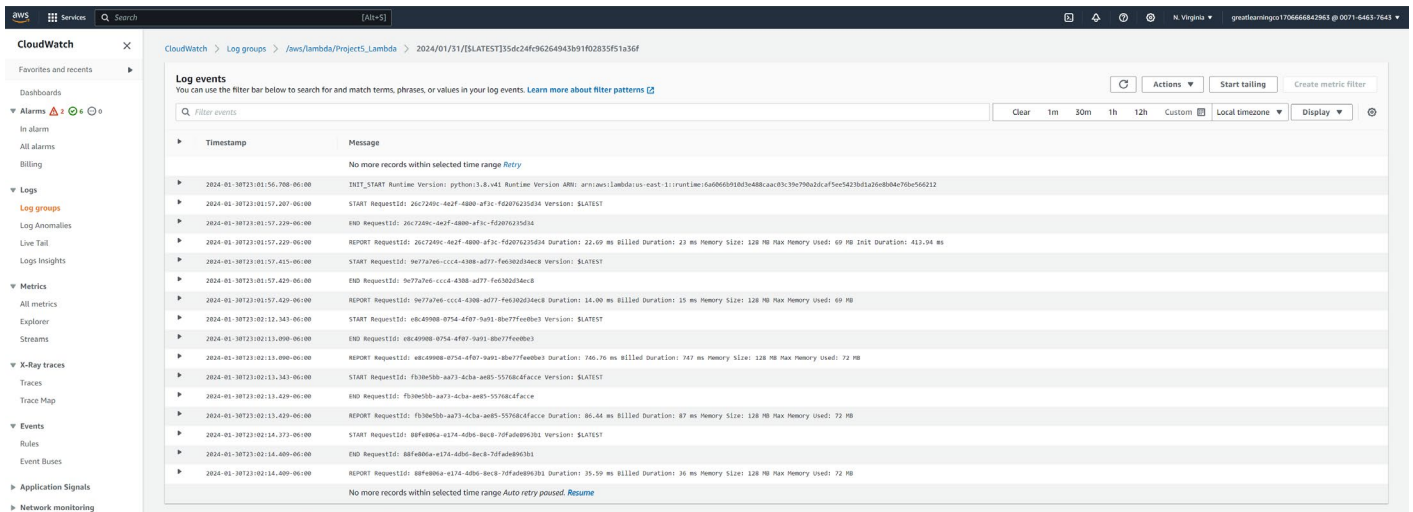
Code source

```

1  # Initialize a DynamoDB client
2  dynamodb = boto3.resource('dynamodb')
3  table = dynamodb.Table('Projects_Dynamodb')
4
5  # Initialize the SNS client
6  sns_client = boto3.client('sns', region_name='us-east-1')
7
8  # SNS topic ARN
9  topic_arn = "arn:aws:sns:us-east-1:1007164637643:Project5_SNS"
10
11 def lambda_handler(event, context):
12     for record in event['Records']:
13         # Kinesis data is base64 encoded so decode here
14         encoded_data = record['kinesis']['data']
15         decoded_data = base64.b64decode(encoded_data)
16         payload = json.loads(decoded_data)
17
18         # Convert numeric values to Decimal for dynamodb compatibility
19         price = Decimal(str(payload['price']))
20         low_stock = Decimal(str(payload['low_stock']))
21         high_stock = Decimal(str(payload['high_stock']))
22
23         # Calculate thresholds
24         within_twenty_percent_of_high = high_stock * Decimal('0.80') <= price <= high_stock
25         within_twenty_five_percent_of_low = low_stock <= price <= low_stock * Decimal('1.25')
26
27         # Check if the price is within the specified range using "or"
28         if within_twenty_percent_of_high or within_twenty_five_percent_of_low:
29             # Update payload with Decimal types
30             payload['price'] = price
31             payload['low_stock'] = low_stock
32             payload['high_stock'] = high_stock
33             payload['price_timestamp'] = str(payload['price_timestamp']) # Ensure price_timestamp is formatted as a string
34
35             # Insert into dynamodb

```

## ADDED ALSO CLOUD WATCH LOGS BELOW



**CloudWatch**

Log groups

/aws/lambda/Project\_Lambda

2024/01/31/[SLATEST]5dc24fc9c264943b91f02835f51a36f

**Log events**

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

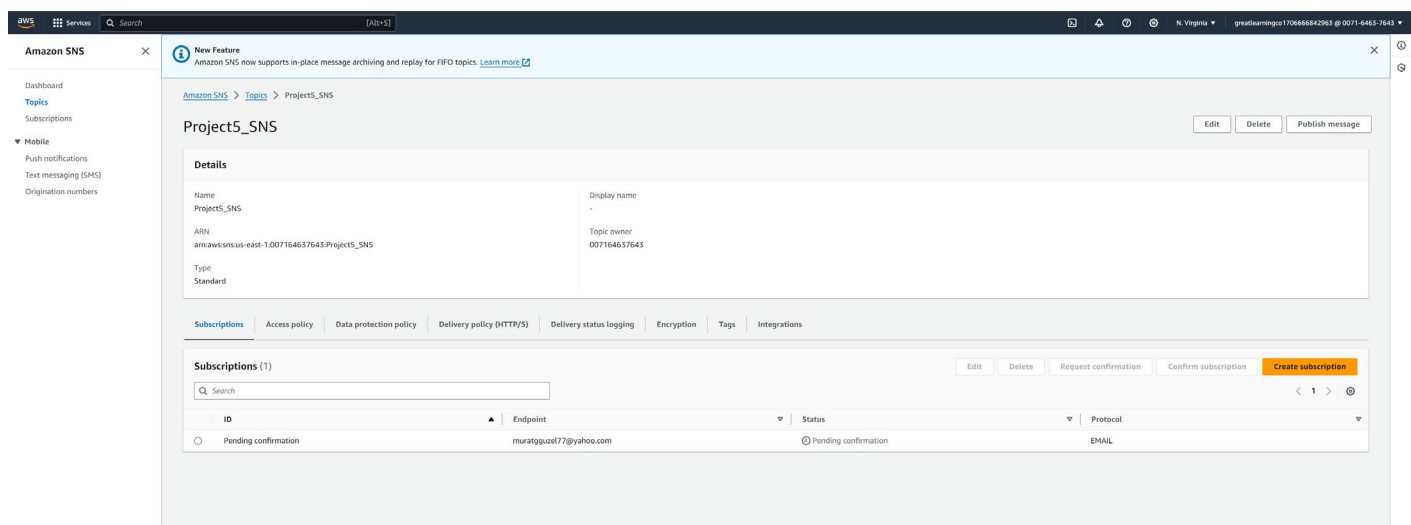
Filter events

Clear 1m 30m 1h 12h Custom Local timezone Display

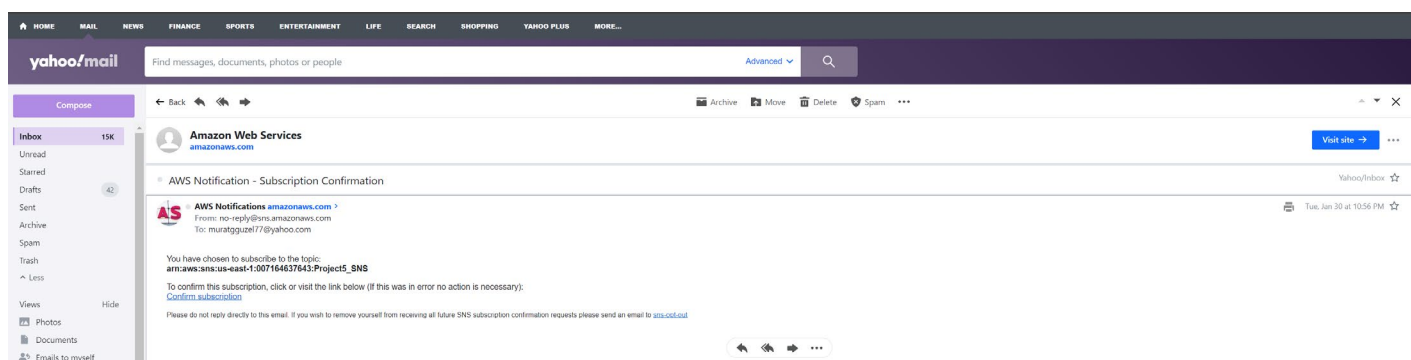
Timestamp	Message
No more records within selected time range <a href="#">Retry</a>	
2024-01-30T23:01:56.708-06:00	START RequestId: 9e77a7e6-cc44-4308-ad77-f6b302d34bc8 Runtime Version: python3.8.10 Runtime Version ARN: arn:aws:lambda:us-east-1:runtime:runtime-640000505b34888-aa03c39e79062dc4f5ee5423b1a26e6084c70be566212
2024-01-30T23:01:57.207-06:00	START RequestId: 26c7249c-4e2f-4800-af3c-fd2079252d36 Version: SLATEST
2024-01-30T23:01:57.229-06:00	END RequestId: 26c7249c-4e2f-4800-af3c-fd2079252d36
2024-01-30T23:01:57.229-06:00	REPORT RequestId: 26c7249c-4e2f-4800-af3c-fd2079252d36 Duration: 22.69 ms Billed Duration: 23 ms Memory Size: 128 MB Max Memory Used: 69 MB Init Duration: 413.34 ms
2024-01-30T23:01:57.415-06:00	START RequestId: 9e77a7e6-cc44-4308-ad77-f6b302d34bc8 Runtime Version: SLATEST
2024-01-30T23:01:57.429-06:00	END RequestId: 9e77a7e6-cc44-4308-ad77-f6b302d34bc8
2024-01-30T23:01:57.429-06:00	REPORT RequestId: 9e77a7e6-cc44-4308-ad77-f6b302d34bc8 Duration: 14.00 ms Billed Duration: 15 ms Memory Size: 128 MB Max Memory Used: 69 MB
2024-01-30T23:02:12.343-06:00	START RequestId: eb449008-0754-4f07-9a91-8be77feeb6c3 Runtime Version: SLATEST
2024-01-30T23:02:13.006-06:00	END RequestId: eb449008-0754-4f07-9a91-8be77feeb6c3
2024-01-30T23:02:13.006-06:00	REPORT RequestId: eb449008-0754-4f07-9a91-8be77feeb6c3 Duration: 746.76 ms Billed Duration: 747 ms Memory Size: 128 MB Max Memory Used: 72 MB
2024-01-30T23:02:13.343-06:00	START RequestId: f33b55b0-a073-4c8a-a085-5578bdcf4fce Runtime Version: SLATEST
2024-01-30T23:02:13.429-06:00	END RequestId: f33b55b0-a073-4c8a-a085-5578bdcf4fce
2024-01-30T23:02:13.429-06:00	REPORT RequestId: f33b55b0-a073-4c8a-a085-5578bdcf4fce Duration: 86.44 ms Billed Duration: 87 ms Memory Size: 128 MB Max Memory Used: 72 MB
2024-01-30T23:02:14.373-06:00	START RequestId: 88f4e08a-e174-4d08-bec8-70fad608301a Runtime Version: SLATEST
2024-01-30T23:02:14.409-06:00	END RequestId: 88f4e08a-e174-4d08-bec8-70fad608301a
2024-01-30T23:02:14.409-06:00	REPORT RequestId: 88f4e08a-e174-4d08-bec8-70fad608301a Duration: 35.59 ms Billed Duration: 36 ms Memory Size: 128 MB Max Memory Used: 72 MB
No more records within selected time range <a href="#">Auto retry paused</a> <a href="#">Resume</a>	

Step number	b
Step name	SNS topic and subscription creation
Instructions	<ol style="list-style-type: none"> <li>1) Goto AWS services look for SNS service</li> <li>2) Create a standard topic</li> <li>3) Create a subscription and subscribe to the topic using your email</li> </ol>
Expected screenshots	<ol style="list-style-type: none"> <li>1) SNS ARN for the created topic</li> <li>2) Confirmation email for the created subscription</li> <li>3) Point of interest data received through, once lambda handler is deployed</li> </ol>

<Insert Screenshot for b(1) here >



<Insert Screenshot for b(2) here>





## Simple Notification Service

### Subscription confirmed!

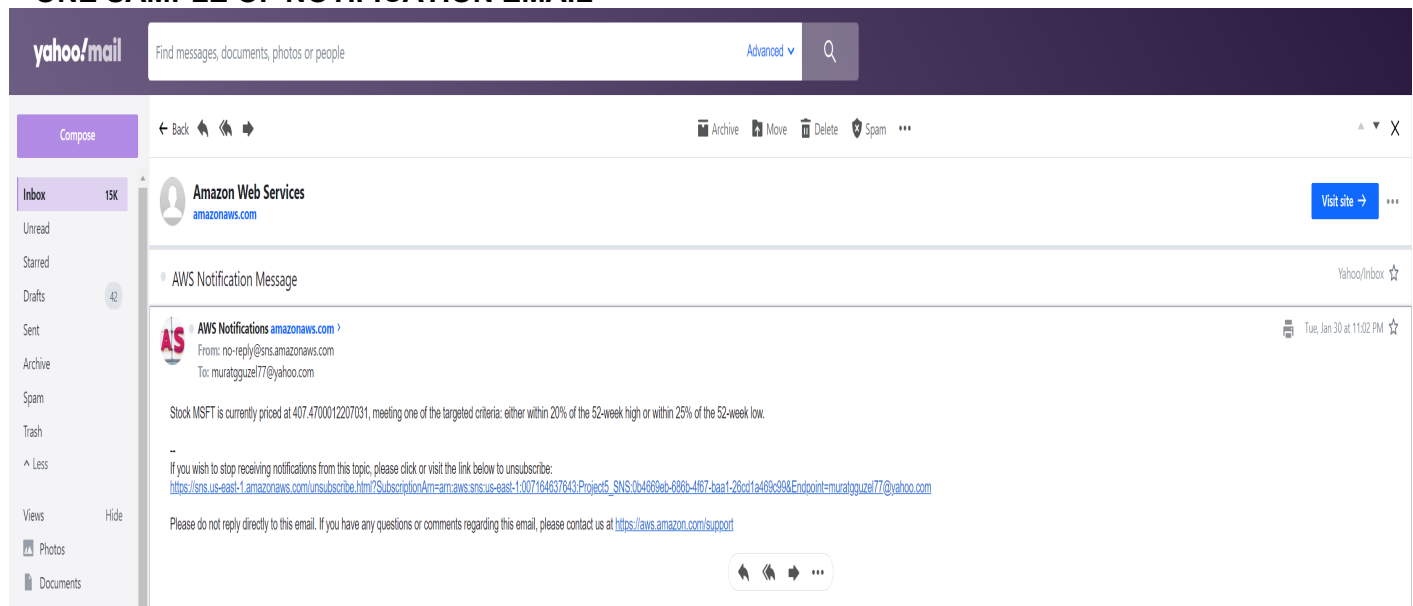
You have successfully subscribed.

Your subscription's id is:

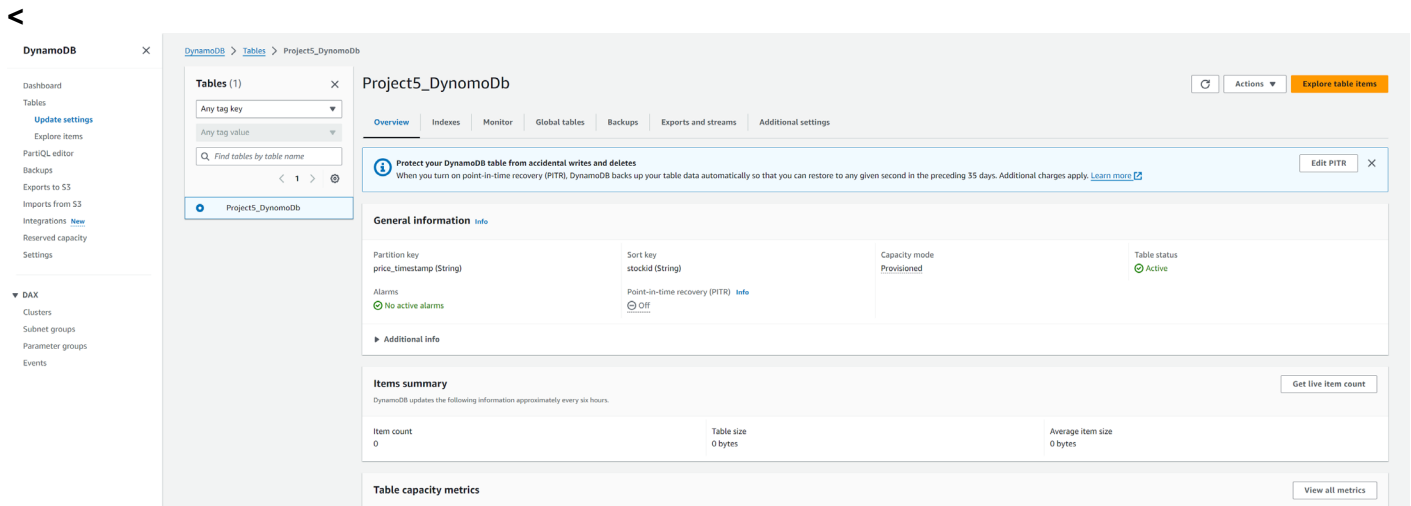
arn:aws:sns:us-east-1:007164637643:Project5\_SNS:0b4669eb-686b-4f67-baa1-26cd1a469c99

If it was not your intention to subscribe, [click here to unsubscribe](#).

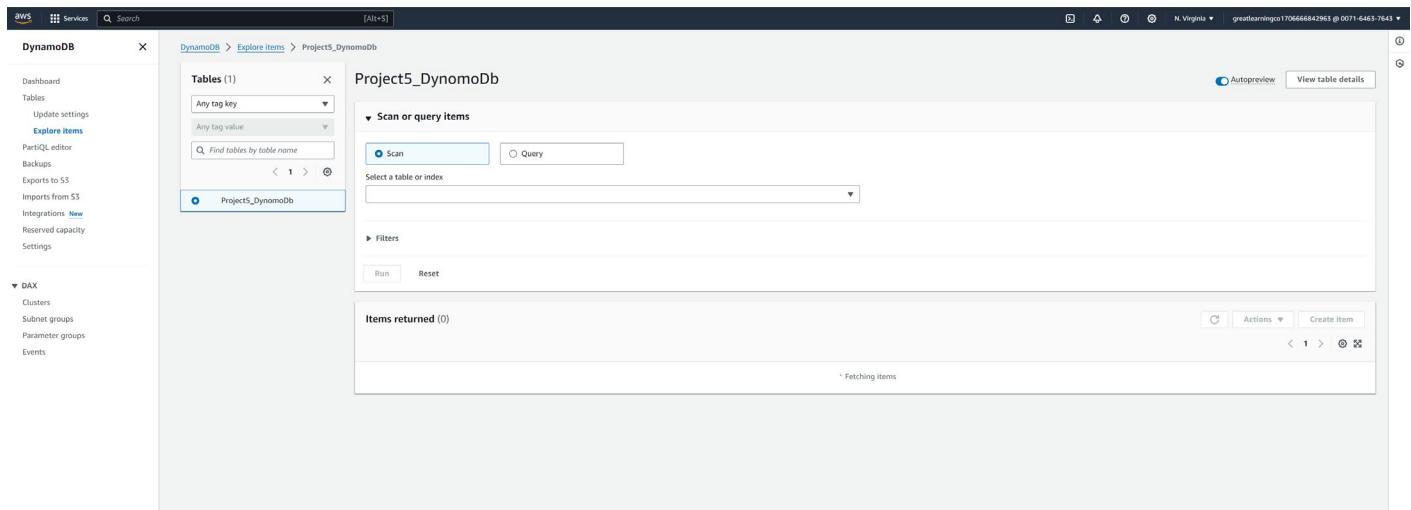
## < ONE SAMPLE OF NOTIFICATION EMAIL



Step number	c
Step name	Dynamodb table creation
Instructions	<ol style="list-style-type: none"> <li>1) Goto AWS services and search for dynamodb</li> <li>2) Create a table by providing the partition key and sort key</li> </ol>
Expected screenshots	<ol style="list-style-type: none"> <li>1) Empty Table</li> <li>2) Table with stored data</li> </ol>



The screenshot shows the AWS DynamoDB console for a table named 'Project5\_DynamoDb'. The table is currently empty, with 0 items and 0 bytes of data. The partition key is 'price\_timestamp (String)' and the sort key is 'stockid (String)'. The table status is 'Active'. The console also shows a warning about protecting the table from accidental writes and deletes, and a section for 'Items summary'.



The screenshot shows the AWS DynamoDB console for a table named 'Project5\_DynamoDb'. The table is currently empty, with 0 items and 0 bytes of data. The partition key is 'price\_timestamp (String)' and the sort key is 'stockid (String)'. The table status is 'Active'. The console also shows a warning about protecting the table from accidental writes and deletes, and a section for 'Items summary'.

## < DATA INSERTION BASED ON CRITERIAS

DynamoDB

Dashboard

Tables

Update settings

Explore items

PartiQL editor

Backups

Exports to S3

Imports from S3

Integrations

Reserved capacity

Settings

DAX

Clusters

Subnet groups

Parameter groups

Events

Selected items have been deleted successfully.

DynamoDB

Explore items

Project5\_DynamoDb

Project5\_DynamoDb

Autopreview

View table details

Tables (1)

Any tag key

Any tag value

Find tables by table name

Project5\_DynamoDb

▼ Scan or query items

Scan

Query

Select a table or index

Table - Project5\_DynamoDb

Select attribute projection

All attributes

▼ Filters

Attribute name

Type

Condition

Value

Remove

Add filter

Run

Reset

Completed. Read capacity units consumed: 0.5

Items returned (6)

Actions

Create Item

	price_timestamp (String)	stockid (String)	52WeekH...	52Week...	price
<input type="checkbox"/>	<a href="#">2024-01-30 15:30:00</a>	MSFT	413.04	245.47	408.5899963378906
<input type="checkbox"/>	<a href="#">2024-01-30 13:30:00</a>	MSFT	413.04	245.47	408.0126953125
<input type="checkbox"/>	<a href="#">2024-01-30 12:30:00</a>	MSFT	413.04	245.47	408.0199890136719
<input type="checkbox"/>	<a href="#">2024-01-30 10:30:00</a>	MSFT	413.04	245.47	410.1400146484375
<input type="checkbox"/>	<a href="#">2024-01-30 11:30:00</a>	MSFT	413.04	245.47	407.4700012207031
<input type="checkbox"/>	<a href="#">2024-01-30 14:30:00</a>	MSFT	413.04	245.47	408.7650146484375