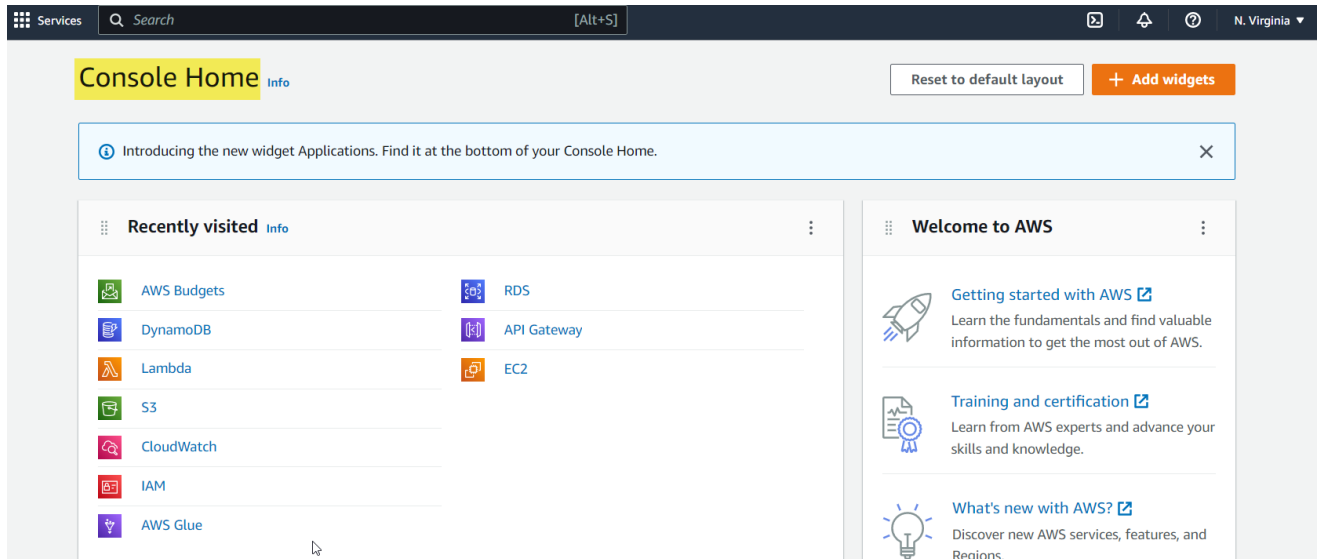


# READING DATA FROM AMAZON S3 USING LAMBDA IN DYNAMODB

## Step 1:

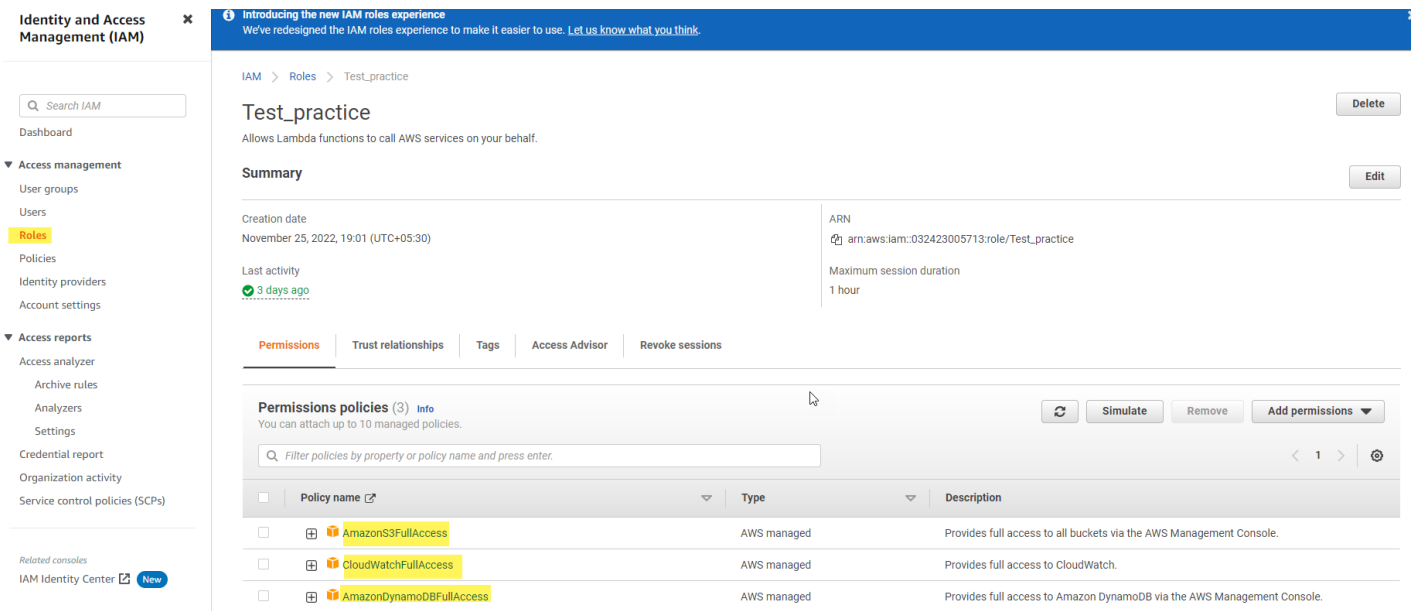
Go to your Amazon Management Console. Once you login with your root user account, we can search any AWS services by using the search bar at the top.



## Step 2:

First, we must create an IAM role. For that first we will go inside IAM and then give access and permission to 3 items in for our case:

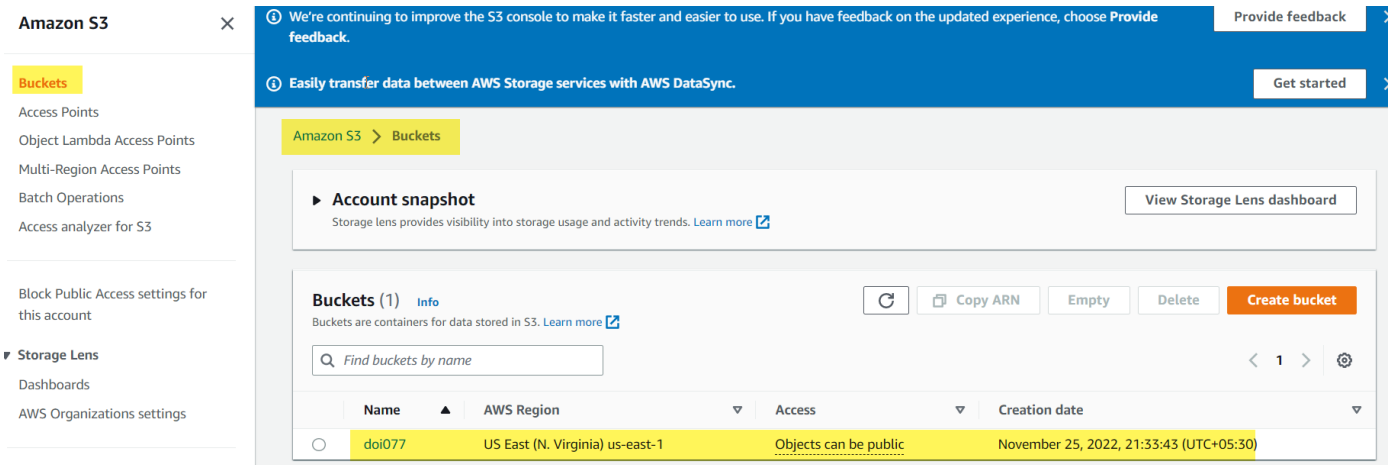
- S3 full access
- DynamoDB full access
- Lambda full access



So, Test\_practice is the role that we have created. And inside that I gave full access to the services that would need to complete this task.

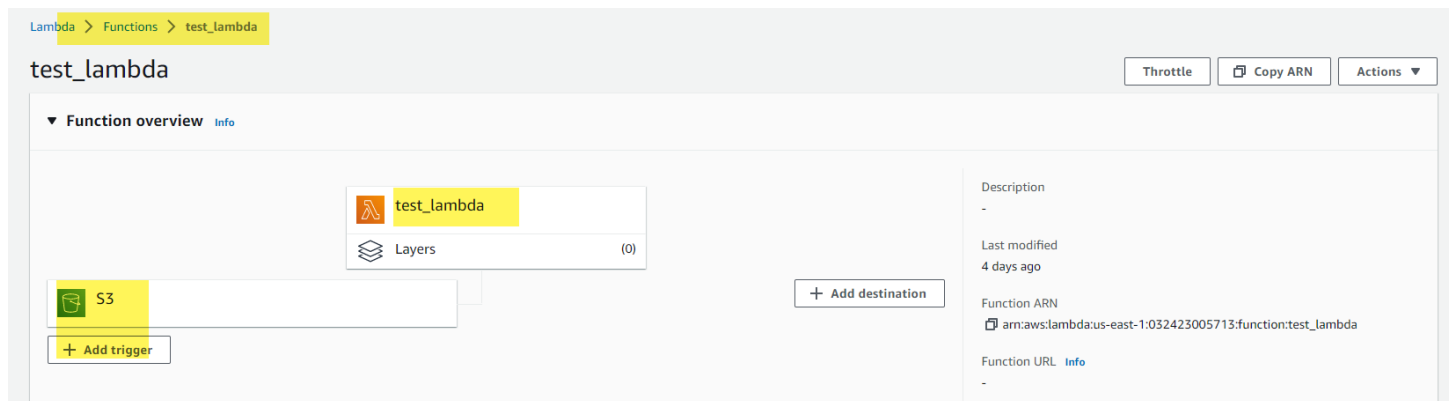
### Step 3:

Once we are done with the creation of the IAM roles and setting up the permission, we now go to Amazon s3 to create a bucket and allow relevant access while creating the bucket. Below shown is the s3 bucket (doi077) that we have created for doing this task.



### Step 4:

After our S3 bucket is ready, we open the AWS LAMBDA service. Here, we will create a lambda function and triggers. First, we will create a trigger by naming it and most importantly we will link the trigger with S3 bucket.



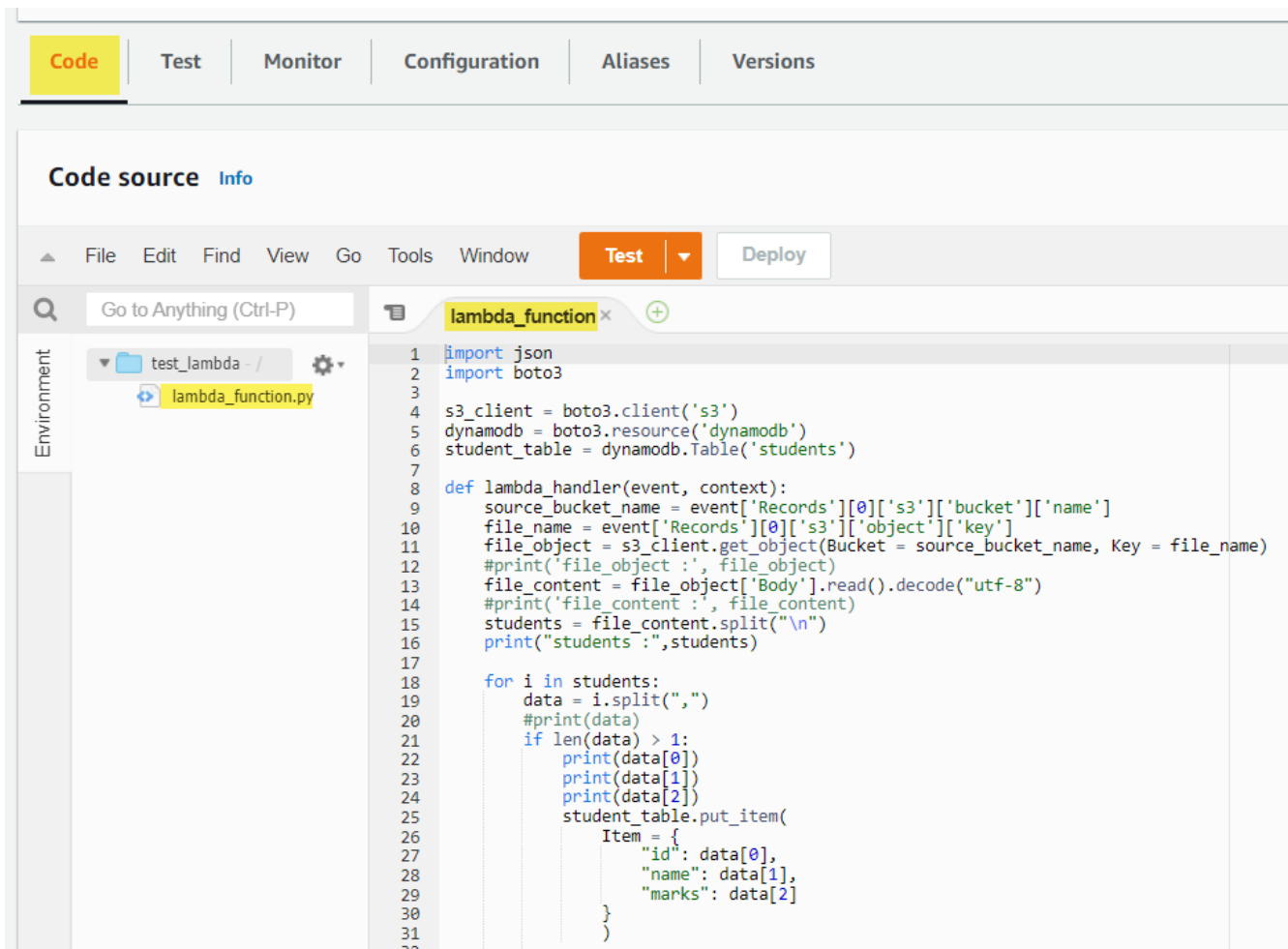
Here in the picture, we have created the lambda function (test\_lambda) in our case and we have also created a trigger, named the trigger, and added the trigger with S3 bucket.

### Step 5:

Now we go back to the lambda function. In the lambda function (test\_lambda), there is a tab named code. Once we go into that, we can write Python code to connect S3 and read the data from S3 using the boto3 package. Below shown is the code to connect S3 and lambda.

```
S3 client = boto3.client('S3')
```

Below picture shows the lambda function that we created to connect with S3 bucket. Once we write the python code in lambda function, we need to deploy the code to trigger the lambda function which will run the function to fetch the data from S3 bucket.

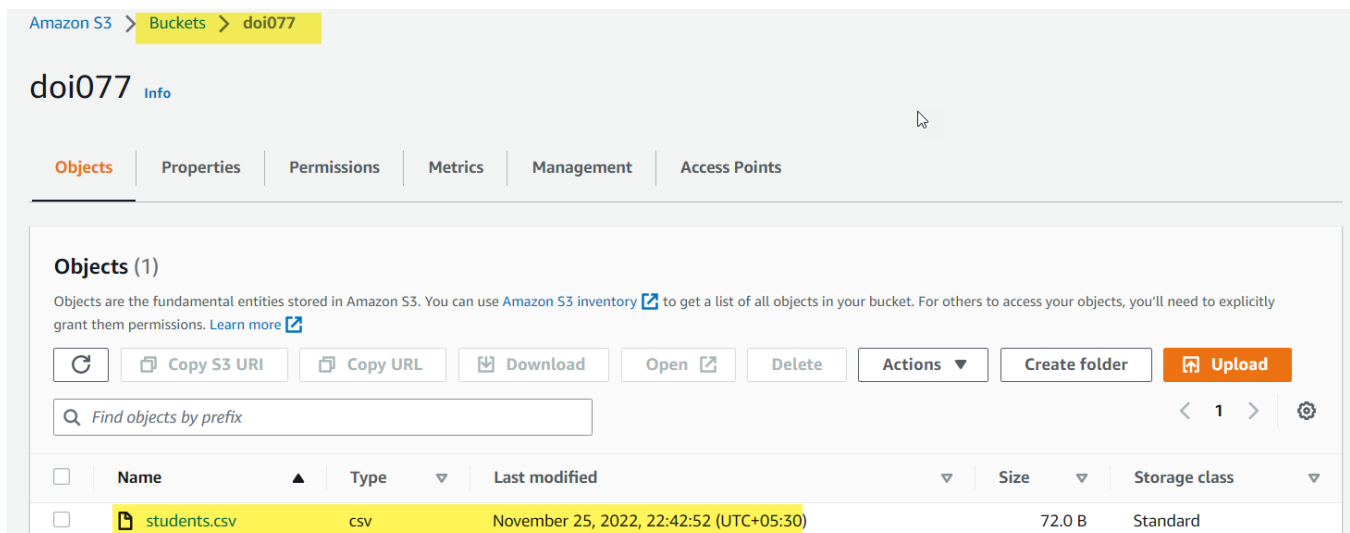


The screenshot shows the AWS Lambda console's 'Code source' tab for a function named 'lambda\_function'. The code is written in Python and uses the boto3 library to interact with S3 and DynamoDB. The function 'lambda\_handler' takes an event and context as input. It extracts the S3 bucket name and object key from the event, retrieves the object from S3, decodes its content, splits it by newlines, and then iterates through the resulting lines. Each line is split by commas, and if it contains more than one element, it is printed and also stored in a DynamoDB table named 'students' with fields for 'id', 'name', and 'marks'.

```
1 import json
2 import boto3
3
4 s3_client = boto3.client('s3')
5 dynamodb = boto3.resource('dynamodb')
6 student_table = dynamodb.Table('students')
7
8 def lambda_handler(event, context):
9     source_bucket_name = event['Records'][0]['s3']['bucket']['name']
10    file_name = event['Records'][0]['s3']['object']['key']
11    file_object = s3_client.get_object(Bucket = source_bucket_name, Key = file_name)
12    #print('file_object :', file_object)
13    file_content = file_object['Body'].read().decode("utf-8")
14    #print('file_content :', file_content)
15    students = file_content.split("\n")
16    print("students :",students)
17
18    for i in students:
19        data = i.split(",")
20        #print(data)
21        if len(data) > 1:
22            print(data[0])
23            print(data[1])
24            print(data[2])
25            student_table.put_item(
26                Item = {
27                    "id": data[0],
28                    "name": data[1],
29                    "marks": data[2]
30                }
31            )
32
```

### Step 6:

After we deploy the code, we go back to the S3 bucket (doi077) and we upload a csv file inside the S3 bucket. We are using students.csv file for our task.



## Step 7:

Soon after we upload the file in the S3 bucket, a trigger has been sent to the lambda function to read the data from students.csv in S3. To read the data logs we use **Amazon CloudWatch** service. We can see if the data is being read correctly, if not it will show some error. We need to correct those error in our Python Lambda function.

To read the log, go to the log groups as highlighted and click on the log streams as shown:

The screenshot shows the Amazon CloudWatch console. On the left is a navigation menu with options like Alarms, Logs, Metrics, and Events. The 'Logs' section is selected, and 'Log groups New' is highlighted. The main area displays the details for the log group `/aws/lambda/test_lambda`. The 'Log group details' section shows the ARN, creation time (4 days ago), retention (Never expire), and stored bytes (10.35 KB). Below this, there are tabs for 'Log streams', 'Metric filters', 'Subscription filters', 'Contributor Insights', 'Tags', and 'Data protection - new'. The 'Log streams' tab is active, showing a list of log streams.

Log stream	Last event time
2022/11/25/[\$LATEST]c03919fb12104629a8a7e16250a40763	2022-11-25 22:42:52 (UTC+05:30)
2022/11/25/[\$LATEST]9988b5a310d246ab8e4124eb15c61d17	2022-11-25 22:22:37 (UTC+05:30)
2022/11/25/[\$LATEST]a1b2120907334911b883e5505adc8989	2022-11-25 22:04:46 (UTC+05:30)
2022/11/25/[\$LATEST]fa6123e9ffde4921b35f3f3271ac9c98	2022-11-25 22:02:21 (UTC+05:30)
2022/11/25/[\$LATEST]454e73a18a2847b987ea1d576192cc5f	2022-11-25 21:57:29 (UTC+05:30)
2022/11/25/[\$LATEST]d44c0d8efcdf44179008ad1f9e1b755a	2022-11-25 21:55:47 (UTC+05:30)
2022/11/25/[\$LATEST]977c4c6e072049d88ed12722643c5f69	2022-11-25 21:45:54 (UTC+05:30)
2022/11/25/[\$LATEST]0dd36e1d779244a2992ce1180238ebd2	2022-11-25 21:43:52 (UTC+05:30)

We can select the log stream to view the output of the lambda function as shown below:

This screenshot shows the 'Log streams' tab in the CloudWatch console. It displays a list of log streams with their names and last event times. The list is sorted by last event time in descending order. The log stream names follow the pattern `2022/11/25/[$LATEST]<hash>`.

Log stream	Last event time
2022/11/25/[\$LATEST]c03919fb12104629a8a7e16250a40763	2022-11-25 22:42:52 (UTC+05:30)
2022/11/25/[\$LATEST]9988b5a310d246ab8e4124eb15c61d17	2022-11-25 22:22:37 (UTC+05:30)
2022/11/25/[\$LATEST]a1b2120907334911b883e5505adc8989	2022-11-25 22:04:46 (UTC+05:30)
2022/11/25/[\$LATEST]fa6123e9ffde4921b35f3f3271ac9c98	2022-11-25 22:02:21 (UTC+05:30)
2022/11/25/[\$LATEST]454e73a18a2847b987ea1d576192cc5f	2022-11-25 21:57:29 (UTC+05:30)
2022/11/25/[\$LATEST]d44c0d8efcdf44179008ad1f9e1b755a	2022-11-25 21:55:47 (UTC+05:30)
2022/11/25/[\$LATEST]977c4c6e072049d88ed12722643c5f69	2022-11-25 21:45:54 (UTC+05:30)
2022/11/25/[\$LATEST]0dd36e1d779244a2992ce1180238ebd2	2022-11-25 21:43:52 (UTC+05:30)

**NOTE:** Now, if suppose in the CloudWatch we find some errors related to our Python lambda function, we go back again to the lambda function code and debug the code. Once we debug the python code, this time we need not again run the complete process starting from uploading a file to S3 till CloudWatch to view the output if its correct. We can create a test environment by simply going in the TEST tab in Lambda and then we configure the test. After configuration, we just deploy our updated code and then click test. We will get the result output in the test environment itself.

### Step 8:

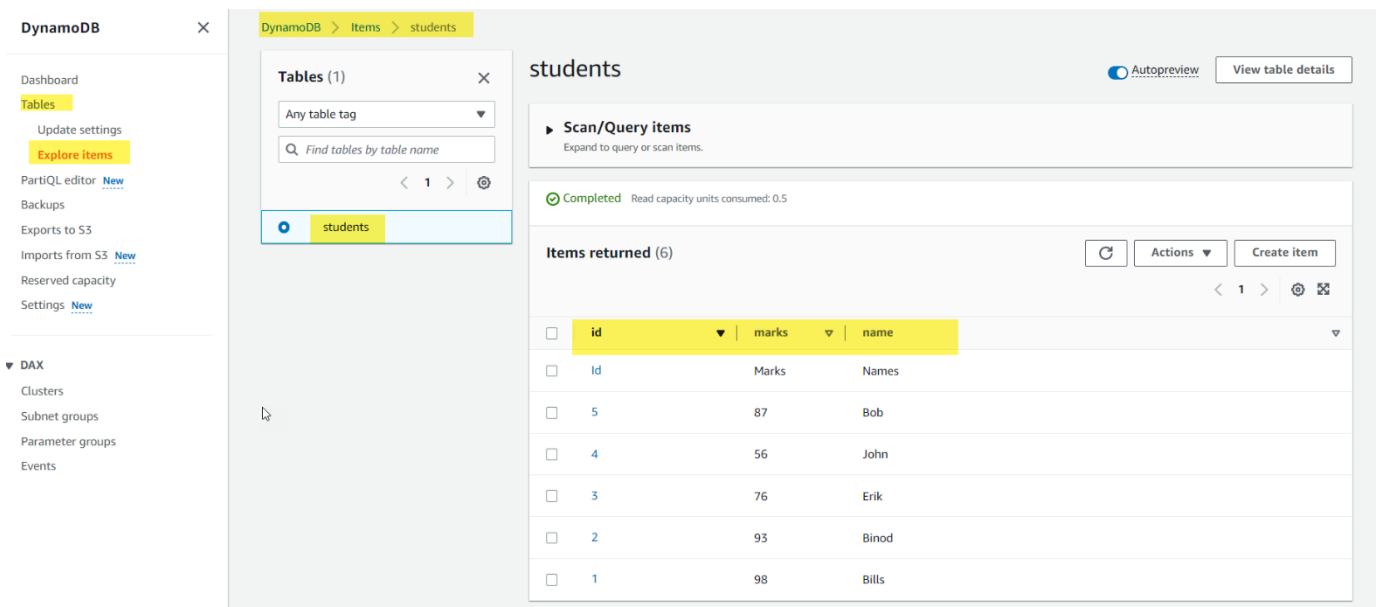
Now, after the output has been received in CloudWatch, we go to Amazon DynamoDB. In DynamoDB we first create a table with the same name (students in our task).

After creating table in DynamoDB, we go back to the lambda function and connect the Lambda with DynamoDB using the follow code:

```
dynamodb = boto3.resource('dynamodb')
```

Once we configure the DynamoDB with lambda, now if we again go to the S3 bucket and upload any file, we can read the output directly in the DynamoDB table.

Below shown is the final output of our task to read the data from S3 bucket from DynamoDB.



The screenshot displays the Amazon DynamoDB console interface. On the left, a navigation sidebar lists various services including Dashboard, Tables, Update settings, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Reserved capacity, and Settings. The main content area is titled 'students' and shows a 'Scan/Query items' section with a 'Completed' status and 'Read capacity units consumed: 0.5'. Below this, a table lists 'Items returned (6)' with columns for 'id', 'marks', and 'name'. The items are as follows:

id	marks	name
5	87	Bob
4	56	John
3	76	Erik
2	93	Binod
1	98	Bills