

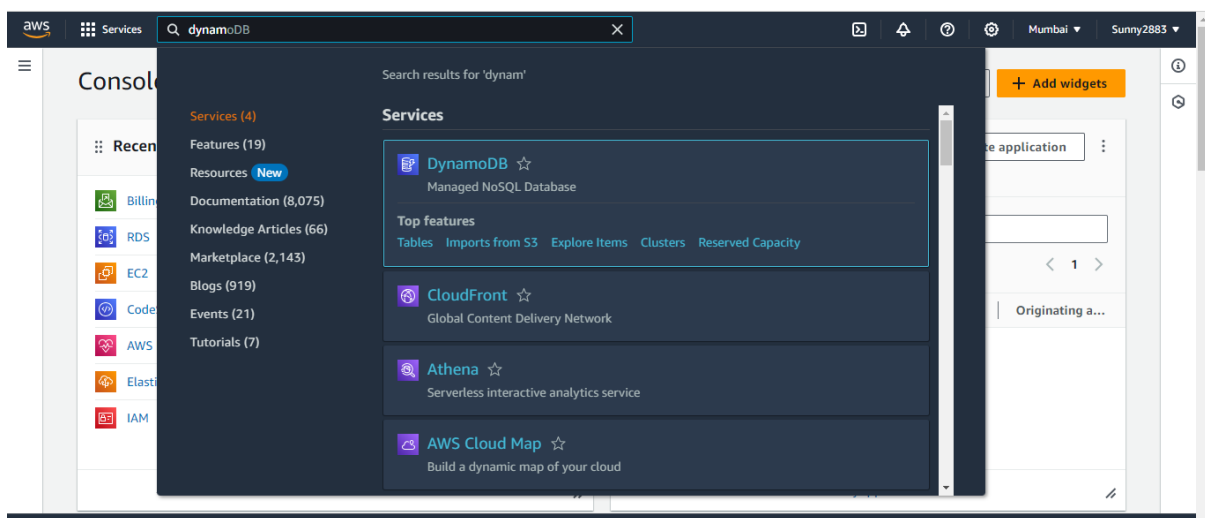
Assignment DB

Task:2

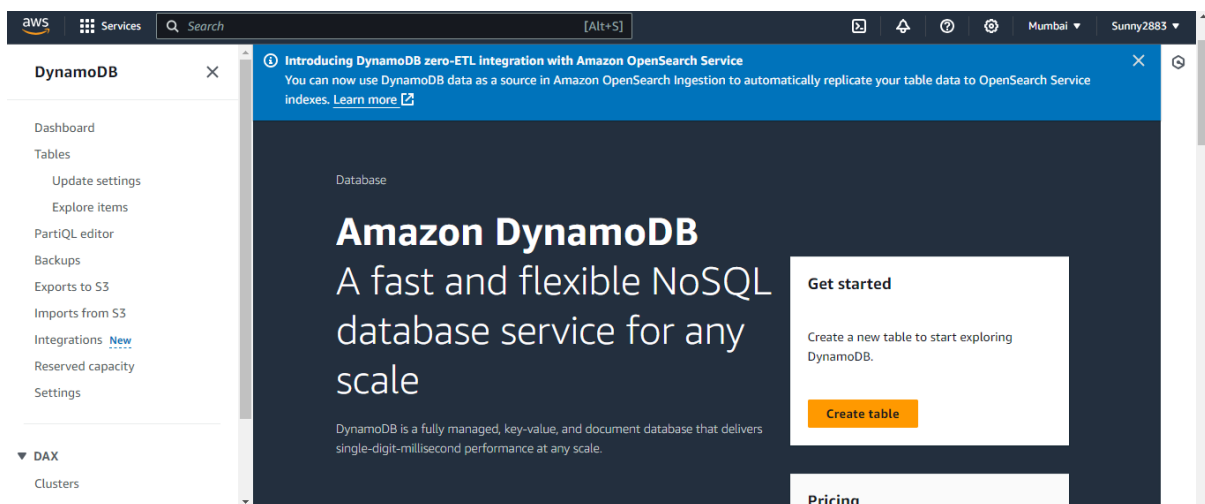
Create a DynamoDB Table:

Create a new DynamoDB table with a primary key of your choice.

Step1: In the AWS Management Console, navigate to the DynamoDB service.



Step2: Click on the "Create table" button.



Step3: Provide a name for your table in the "Table name" field and enter a name for your primary key in the "Partition key" field. This will be your hash key. Optionally, you can also add a sort key by enabling the "Add sort key" option and entering a name for the sort key.

aws

Services

Search

[Alt+S]

Mumbai

Sunny2883

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Students

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

Roll_No

Number

1 to 255 characters and case sensitive.

Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name

String

1 to 255 characters and case sensitive.

Table settings

CloudShell

Feedback

© 2024, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

Step4: Configure Other Settings I choose default.

Table settings

☒ Default settings

The fastest way to create your table. You can modify these settings now or after your table has been created.

☐ Customize settings

Use these advanced features to make DynamoDB work better for your needs.

Step5: Configure default setting.

aws

Services

Search

[Alt+S]

Mumbai

Sunny2883

Default table settings

These are the default settings for your new table. You can change some of these settings after creating the table.

Setting	Value	Editable after creation
Table class	DynamoDB Standard	Yes
Capacity mode	Provisioned	Yes
Provisioned read capacity	5 RCU	Yes
Provisioned write capacity	5 WCU	Yes
Auto scaling	On	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	Owned by Amazon DynamoDB	Yes
Deletion protection	Off	Yes

CloudShell

Feedback

© 2024, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

Step6: Review the configuration setting and click on create table.

Auto scaling: On, Yes

Local secondary indexes: -, No

Global secondary indexes: -, Yes

Encryption key management: Owned by Amazon DynamoDB, Yes

Deletion protection: Off, Yes

Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

[Cancel](#) [Create table](#)

Step7: Check Table Status:

DynamoDB

Share your feedback on Amazon DynamoDB

Introducing DynamoDB zero-ETL integration with Amazon OpenSearch Service

Creating the Students table. It will be available for use shortly.

Tables (1)

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode
Students	Creating	Roll_No (N)	-	0	Off	Provisioned (5)	Provisioned (5)

DynamoDB

Introducing DynamoDB zero-ETL integration with Amazon OpenSearch Service

Students

Overview | Indexes | Monitor | Global tables | Backups | Exp

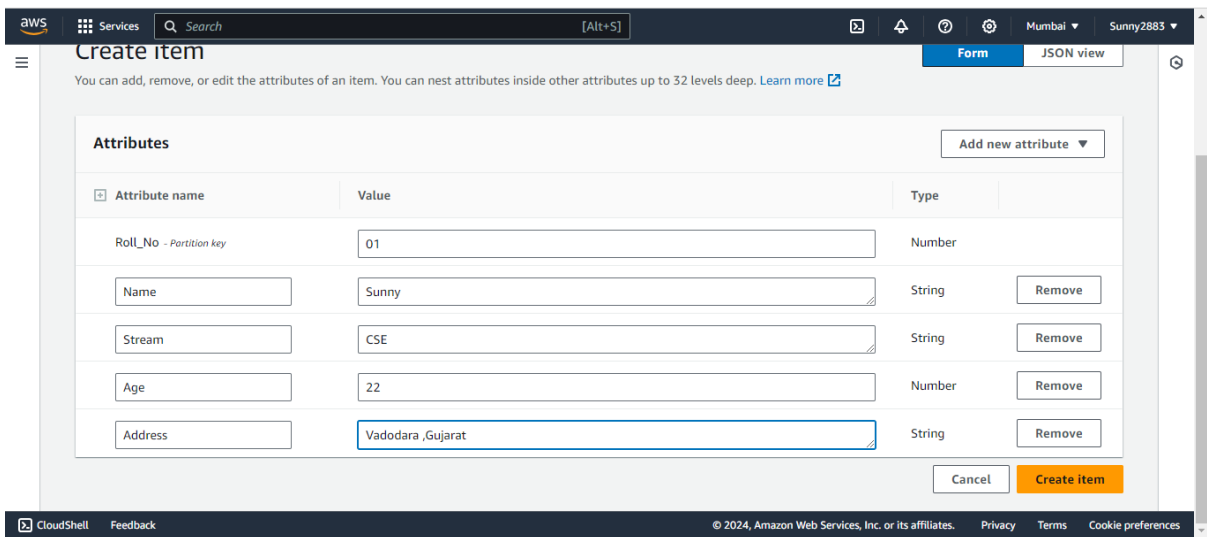
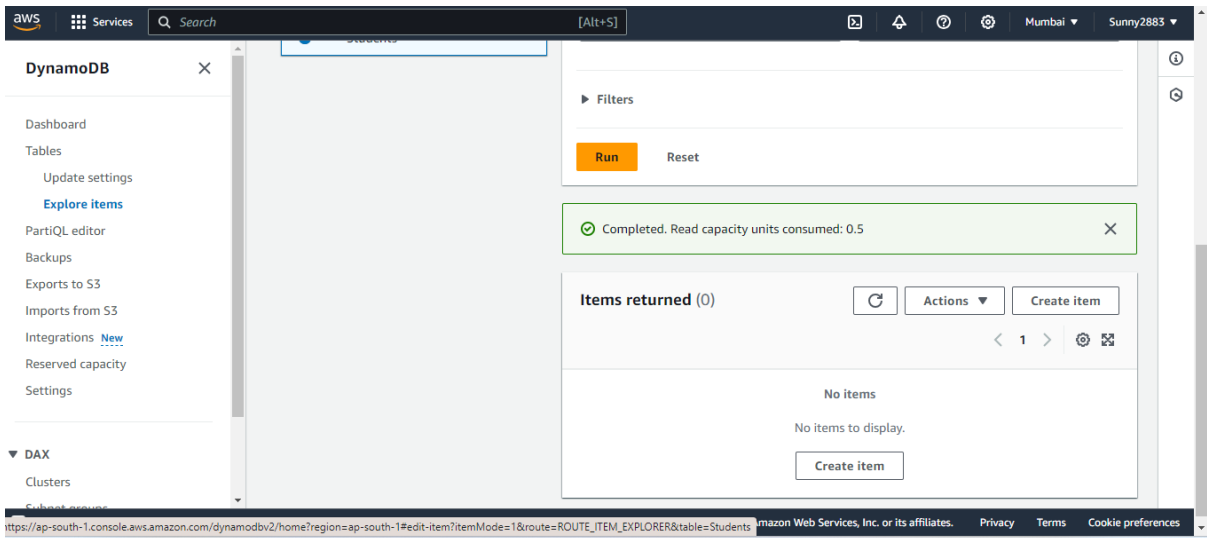
Protect your DynamoDB table from accidental writes and deletes

When you turn on point-in-time recovery (PITR), DynamoDB backs up your table data automatically so that you can restore to any given second in the preceding 35 days. Additional charges apply. [Learn more](#)

[Edit PITR](#)

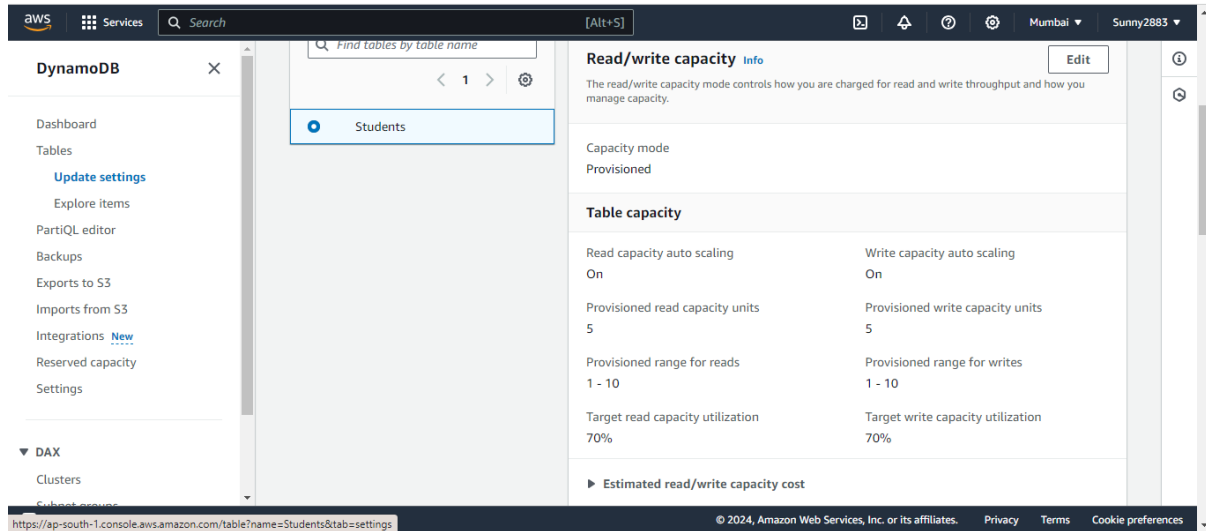
General information | Info

You can use the console to insert, retrieve, update, and delete items in the table by navigating to the "Items" tab of your table.



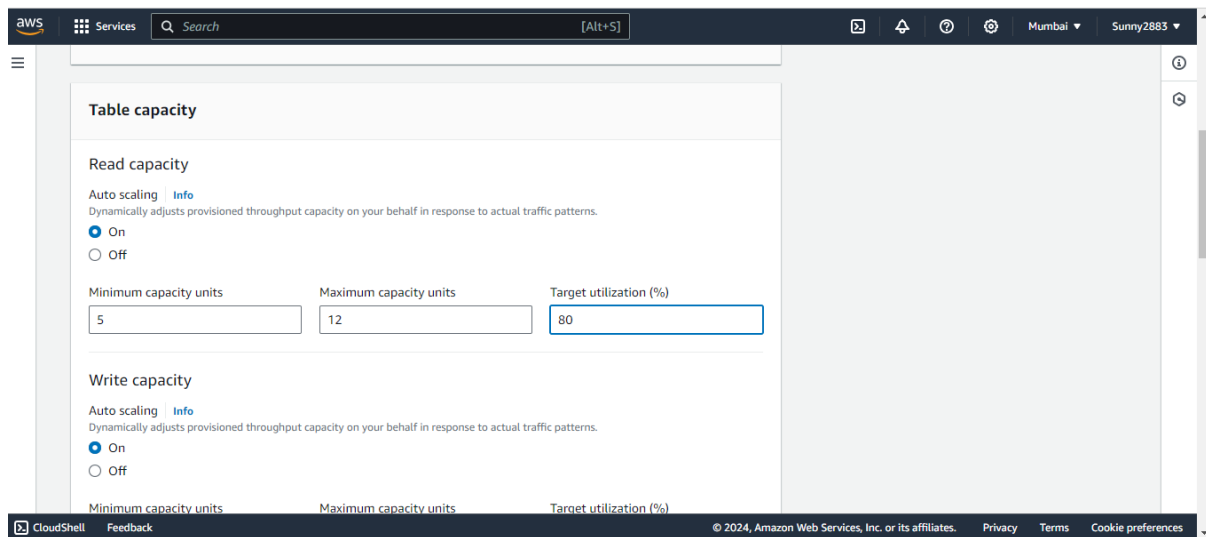
2. Define the provisioned throughput for the table.

Step1: select the table for which you want to define provisioned throughput and in the selected table's details, go to the "Overview" tab in the selected table's details, go to the "Overview" tab.



Step2: Under "Provisioned Throughput," you will see options to set read capacity and write capacity.

Enter the desired values for "Read capacity units" and "Write capacity units." These values represent the number of reads and writes per second that your table can handle.



Write capacity

Auto scaling [Info](#)
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.

☒ On
☐ Off

Minimum capacity units: 4
Maximum capacity units: 11
Target utilization (%): 70

▼ Historical capacity usage vs current selection

To see detailed historical read and write usage data for your table, go to [Cloudwatch](#)

Read usage vs current unit selection
The number of read capacity units consumed over the last month. [Learn more](#)

Filter displayed data

Write usage vs current unit selection
The number of write capacity units consumed over the last month. [Learn more](#)

Filter displayed data

Step3: After configuring the provisioned throughput settings, click on the "Save" button.

Maximum capacity units
Consumed read capacity units

Maximum capacity units
Consumed write capacity units

Estimated read/write capacity cost

Here is the estimated total cost of provisioned read and write capacity for your table and indexes, based on your current settings. To learn more, see [Amazon DynamoDB pricing](#) for provisioned capacity.

Total read capacity units	Total write capacity units	Region	Estimated cost
1	1	ap-south-1	\$0.67 / month

By turning on auto scaling, you authorize DynamoDB to scale capacity by using the DynamoDB auto scaling service-linked role: AWSServiceRoleForApplicationAutoScaling_DynamoDBTable.

Cancel **Save changes**

Step4: Review your configuration settings on the confirmation screen.

DynamoDB

Dashboard
Tables
[Update settings](#)
Explore items
 PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations [New](#)
Reserved capacity
Settings

▼ DAX
Clusters

Table capacity

Read capacity auto scaling On	Write capacity auto scaling On
Provisioned read capacity units 5	Provisioned write capacity units 1
Provisioned range for reads 5 - 12	Provisioned range for writes 4 - 11
Target read capacity utilization 80%	Target write capacity utilization 70%

► Estimated read/write capacity cost

Auto scaling activities (3) [Refresh](#)

Recent events of automatic scaling. [Learn more](#)

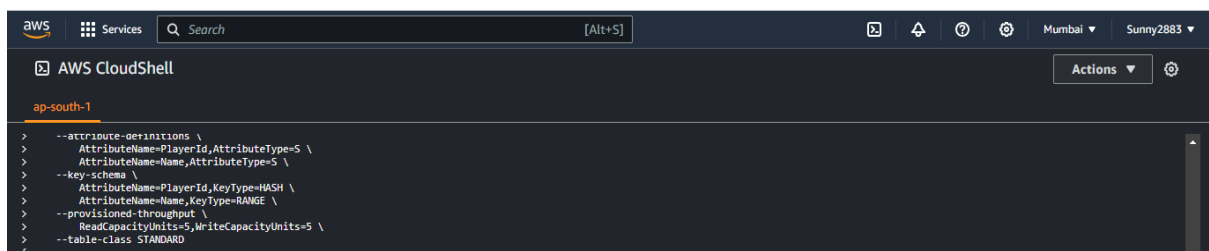
Find events

3. Run CRUD operation in the in-DB table using Queries.

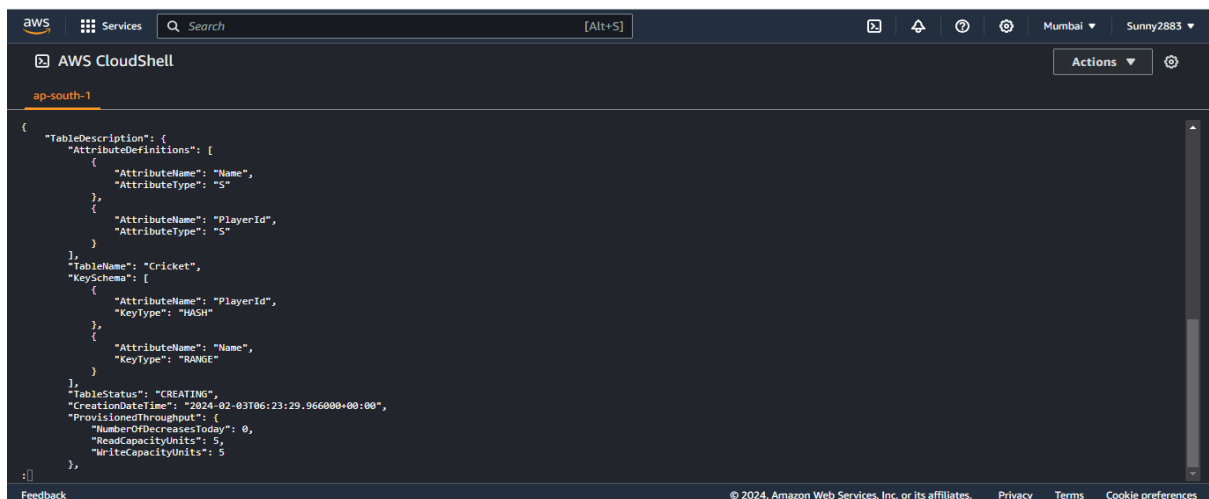
1. Create DynamoDB Table:

Run the command to create a DynamoDB table

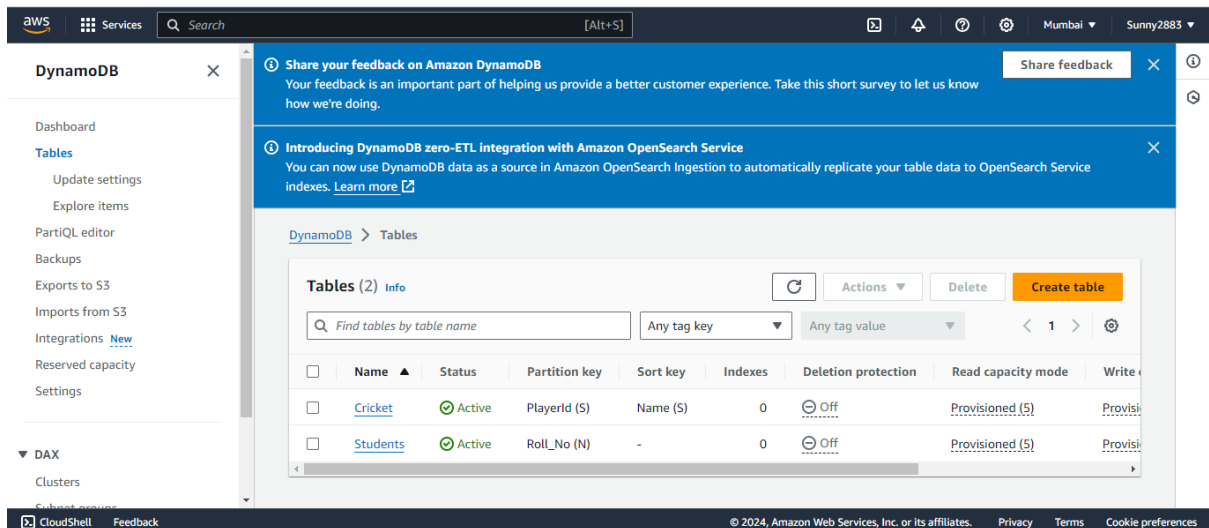
```
aws dynamodb create-table \  
  --table-name Cricket \  
  --attribute-definitions \  
    AttributeName=PlayerId,AttributeType=S \  
    AttributeName=Name,AttributeType=S \  
  --key-schema \  
    AttributeName=PlayerId,KeyType=HASH \  
    AttributeName=Name,KeyType=RANGE \  
  --provisioned-throughput \  
    ReadCapacityUnits=5,WriteCapacityUnits=5 \  
  --table-class STANDARD
```



```
aws dynamodb create-table \  
  --table-name Cricket \  
  --attribute-definitions \  
    AttributeName=PlayerId,AttributeType=S \  
    AttributeName=Name,AttributeType=S \  
  --key-schema \  
    AttributeName=PlayerId,KeyType=HASH \  
    AttributeName=Name,KeyType=RANGE \  
  --provisioned-throughput \  
    ReadCapacityUnits=5,WriteCapacityUnits=5 \  
  --table-class STANDARD
```



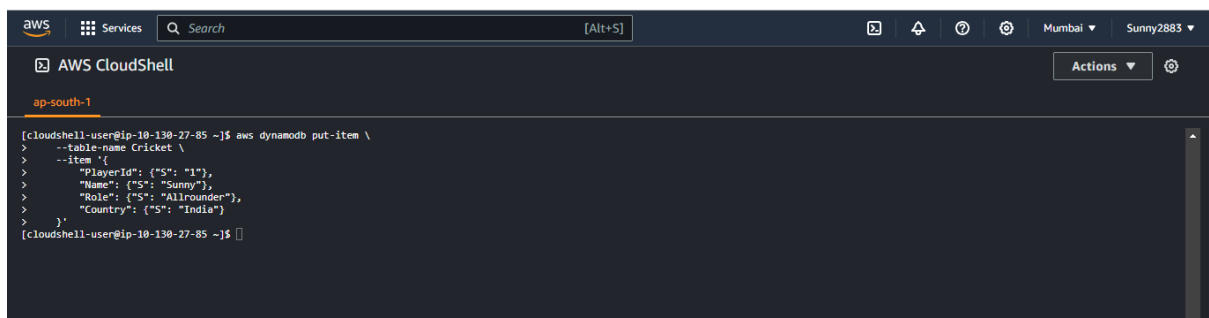
```
{  
  "TableDescription": {  
    "AttributeDefinitions": [  
      {  
        "AttributeName": "Name",  
        "AttributeType": "S"  
      },  
      {  
        "AttributeName": "PlayerId",  
        "AttributeType": "S"  
      }  
    ],  
    "TableName": "Cricket",  
    "KeySchema": [  
      {  
        "AttributeName": "PlayerId",  
        "KeyType": "HASH"  
      },  
      {  
        "AttributeName": "Name",  
        "KeyType": "RANGE"  
      }  
    ],  
    "TableStatus": "CREATING",  
    "CreationDateTime": "2024-02-03T06:23:29.966000+00:00",  
    "ProvisionedThroughput": {  
      "NumberOfDecreasesToday": 0,  
      "ReadCapacityUnits": 5,  
      "WriteCapacityUnits": 5  
    }  
  },  
  "TableArn": "arn:aws:dynamodb:us-east-1:123456789012:table/Cricket"  
}
```



2. Insert Item:

Run the command to insert an item into the table:

```
aws dynamodb put-item \
  --table-name Cricket \
  --item '{
    "PlayerId": {"S": "1"},
    "Name": {"S": "Sunny"},
    "Role": {"S": "Allrounder"},
    "Country": {"S": "India"}
  }'
```




```
aws dynamodb put-item \  
  --table-name Cricket \  
  --item '{  
    "PlayerId": {"S": "2"},  
    "Name": {"S": "John"},  
    "Role": {"S": "Batsman"},  
    "Country": {"S": "England"}  
  }'
```

```
> --table-name Cricket \  
> --item '{  
>   "PlayerId": {"S": "2"},  
>   "Name": {"S": "John"},  
>   "Role": {"S": "Batsman"},  
>   "Country": {"S": "England"}  
> }'
```

The screenshot shows the AWS Management Console interface for a DynamoDB table named 'Cricket'. The left sidebar contains navigation options like Dashboard, Tables, and Settings. The main panel shows the 'Cricket' table with a 'Scan or query items' button. Below this, a notification states 'Completed. Read capacity units consumed: 0.5'. The 'Items returned (2)' section displays a table with two items:

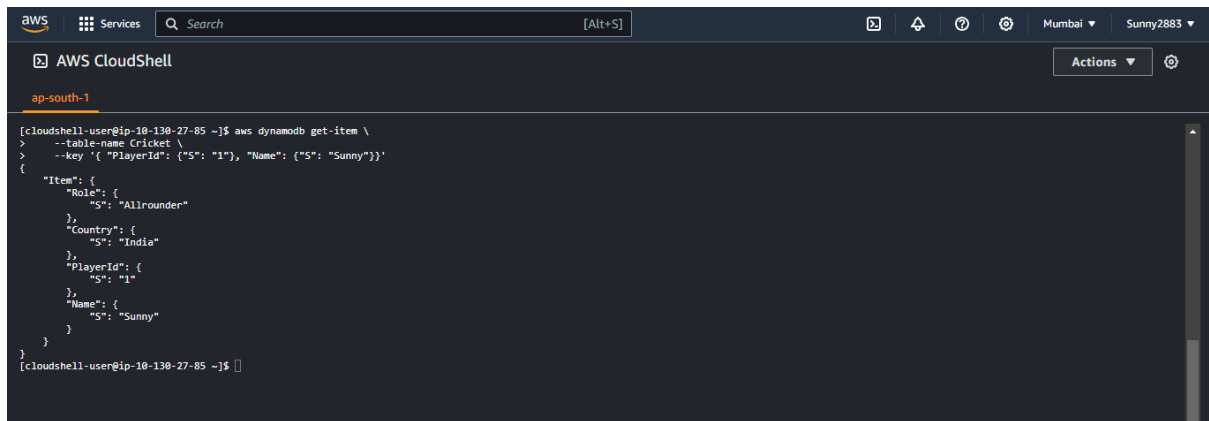
	Playerid (String)	Name (String)	Country	Role
<input type="checkbox"/>	2	John	England	Batsman
<input type="checkbox"/>	1	Sunny	India	Allround

3. Retrieve Item (Read):

Run the command to retrieve an item based on its primary key:

```
aws dynamodb get-item \  
  --table-name Cricket \  
  --key '{ "PlayerId": {"S": "1"}, "Name": {"S": "Sunny"}}'
```

```
aws CloudShell  
ap-south-1  
[cloudshell-user@ip-10-130-27-85 ~]$ aws dynamodb get-item \  
> --table-name Cricket \  
> --key '{ "PlayerId": {"S": "1"}, "Name": {"S": "Sunny"}}'  
{
```



```
aws
Services Search [Alt+S] Mumbai Sunny2883
AWS CloudShell
ap-south-1
[cloudshell-user@ip-10-130-27-85 ~]$ aws dynamodb get-item \
> --table-name Cricket \
> --key '{ "PlayerId": {"S": "1"}, "Name": {"S": "Sunny"}}'
{
  "Item": {
    "Role": {
      "S": "Allrounder"
    },
    "Country": {
      "S": "India"
    },
    "PlayerId": {
      "S": "1"
    },
    "Name": {
      "S": "Sunny"
    }
  }
}
[cloudshell-user@ip-10-130-27-85 ~]$
```

4. Update Item:

Run the command to update an item:

aws dynamodb update-item \

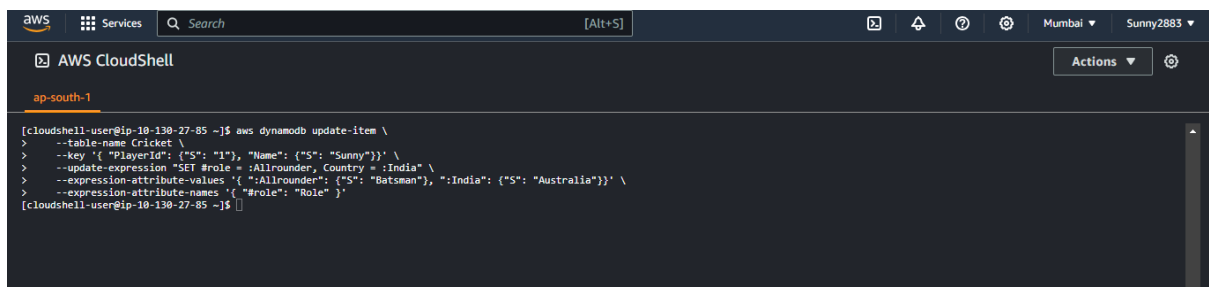
--table-name Cricket \

--key '{ "PlayerId": {"S": "1"}, "Name": {"S": "Sunny"}}' \

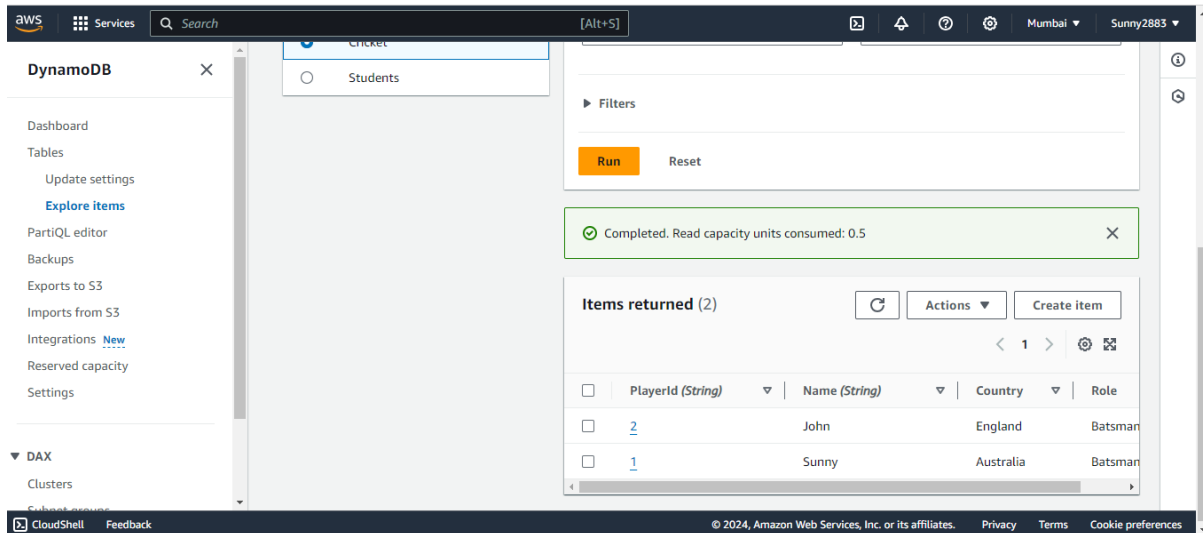
--update-expression "SET #role = :Allrounder, Country = :India" \

--expression-attribute-values '{ ":Allrounder": {"S": "Batsman"}, ":India": {"S": "Australia"}}' \

--expression-attribute-names '{ "#role": "Role" }'



```
aws
Services Search [Alt+S] Mumbai Sunny2883
AWS CloudShell
ap-south-1
[cloudshell-user@ip-10-130-27-85 ~]$ aws dynamodb update-item \
> --table-name Cricket \
> --key '{ "PlayerId": {"S": "1"}, "Name": {"S": "Sunny"}}' \
> --update-expression "SET #role = :Allrounder, Country = :India" \
> --expression-attribute-values '{ ":Allrounder": {"S": "Batsman"}, ":India": {"S": "Australia"}}' \
> --expression-attribute-names '{ "#role": "Role" }'
[cloudshell-user@ip-10-130-27-85 ~]$
```



5. Delete Item:

Run the command to delete an item:

aws dynamodb delete-item \

--table-name Cricket \

--key '{ "PlayerId": {"S": "2"}, "Name": {"S": "John"} }'

