

# Lab: DynamoDB

## Lab overview and objectives

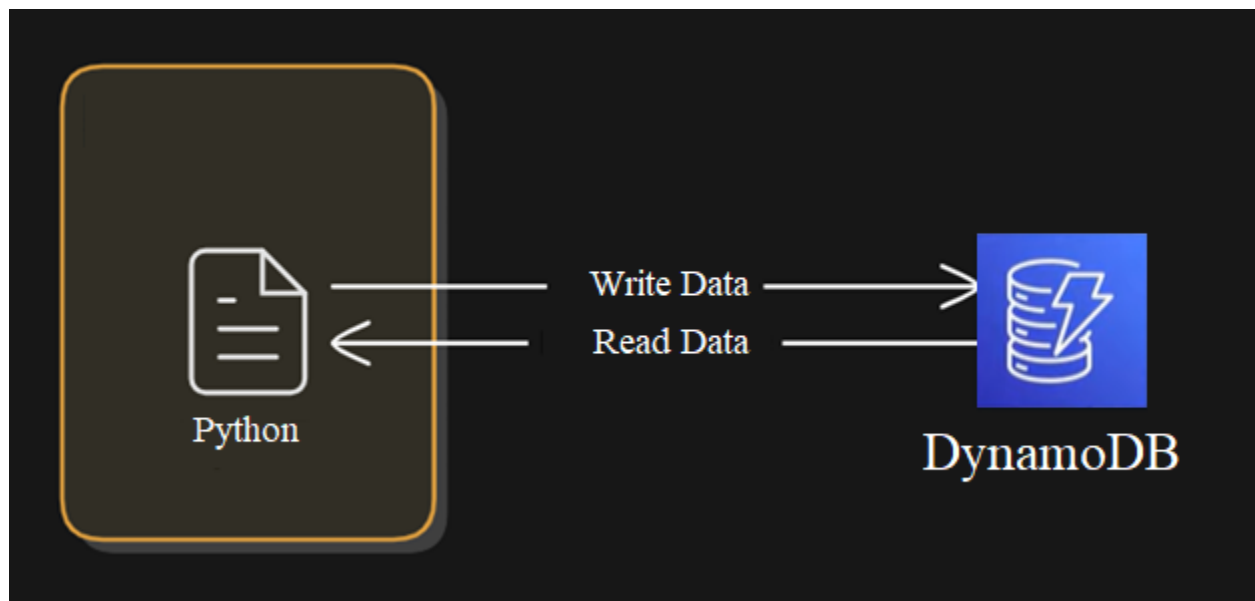
---

In this lab, you will use Amazon DynamoDB to store and query data.

You will learn to use both **AWS Management Console** and **AWS SDK for Python (Boto3)** to work with DynamoDB.

After completing this lab, you should be able to:

- Create a new DynamoDB table, add data to the table and query the data from the **AWS Management Console**.
- Add data to a **DynamoDB** table and query data from **DynamoDB** table using **AWS SDK for Python (Boto3)**.



# Scenario

You will create a DynamoDB table that will store calculator calculations.

## Accessing the AWS Management Console

For this lab, we will use the **AWS Academy learner lab** and the AWS Management Console.

## Task 1: Create a DynamoDB Table

1. In the AWS Console search box to the right of **Services**, search for and choose **DynamoDB** to open the **DynamoDB** console.
2. Choose **Create table**.
3. In the **Create table** screen, configure these settings:
  - Table name: `calculations-table`
  - Partition key name: `timestamp`.

☰ [DynamoDB](#) > [Tables](#) > Create table

### Create table

**Table details** [Info](#)  
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.

`calculations-table`

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.

`timestamp` String

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.

4. Choose **Create table**.
5. The table will now be created, this operation might take a few minutes.

✓ Creating the calculations-table table. It will be available for use shortly.

6. Once the table is ready to use, the following message will appear:

🔔 The calculations-table table was created successfully. ✕

**Tables (1)** [Info](#)

🔍 Find tables  Any tag key  Any tag value  1 match < 1 > ⚙️

<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode	Write capacity mode
<input type="checkbox"/>	<a href="#">calculations-table</a>	Active	timestamp (S)	-	0	0	Off	☆	On-demand	On-demand

# Task 2: Create and Explore table items using AWS Console

1. Select the table `calculations-table`, the `calculations-table` screen will open.

The screenshot shows the AWS DynamoDB console for the table 'calculations-table'. At the top, there's a header with the table name, a refresh icon, an 'Actions' dropdown, and an 'Explore table items' button. Below this is a navigation bar with tabs: Overview (selected), Indexes, Monitor, Global tables, Backups, Exports and streams, Permissions, and Additional settings. A blue banner at the top of the Overview tab contains a warning about protecting the table from accidental writes and deletes, with an 'Edit PITR' button. Below the banner is a 'General information' section with a grid of table properties: Partition key (timestamp (String)), Sort key (-), Capacity mode (On-demand), Table status (Active), Alarms (No active alarms), Point-in-time recovery (PITR) (Off), and Resource-based policy (Not active). An 'Additional info' link is at the bottom.

**calculations-table** ☆

Actions ▾ Explore table items

Overview | Indexes | Monitor | Global tables | Backups | Exports and streams | Permissions | Additional settings

ⓘ 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

<b>Partition key</b> timestamp (String)	<b>Sort key</b> -	<b>Capacity mode</b> On-demand	<b>Table status</b> ✓ Active
<b>Alarms</b> ✓ No active alarms	<b>Point-in-time recovery (PITR)</b> Info ⊖ Off	<b>Resource-based policy</b> Info ⊖ Not active	

► Additional info

2. Choose the **Actions** drop down and select **Create item**. The **Create item** screen will open. Add the following information and press **Create item**:

The screenshot shows the 'Create item' screen in the AWS DynamoDB console. At the top, there's a header with 'Create item', a 'Form' tab (selected), and a 'JSON view' tab. Below the header is a sub-header 'Attributes' with an 'Add new attribute' button. The main area is a table with columns: Attribute name, Value, Type, and an empty column. The table contains five rows of attributes: 'timestamp - Partition key' (String, value: 1732986883), 'num1' (Number, value: 4), 'operation' (String, value: +), 'num2' (Number, value: 3), and 'result' (Number, value: 7). Each row has a 'Remove' button. At the bottom right, there are 'Cancel' and 'Create item' buttons.

**Create item** Form JSON view

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

**Attributes** Add new attribute ▾

Attribute name	Value	Type	
timestamp - Partition key	1732986883	String	
num1	4	Number	Remove
operation	+	String	Remove
num2	3	Number	Remove
result	7	Number	Remove

Cancel Create item

**Note 1:** To get the current timestamp, you can use your preferred programming language or visit a site that provides it, such as: <https://www.utctime.net/utc-timestamp>

**Note 2:** To add a new value, press **Add new attribute** and select the desired attribute type.

3. The item should be created, and the following screen will appear:

The item has been saved successfully.

Tables (1)

×

Any tag key

▼

Any tag value

▼

Find tables

1 match

<

1

>

⚙

calculations-table

☆

calculations-table

▼ Scan or query items

Scan

Query

Select a table or index

Table - calculations-table

▼

Filters

Run

Reset

Completed. Read capacity units consumed: 2

Items returned (1)

timestamp (String)

▼

num1

▼

num2

▼

operation

▼

result

1732986883

4

3

+

7

# Task 3: Create and Explore table items using AWS SDK for Python (Boto3)

---

In this task, you will interact with DynamoDB using Boto3. Python knowledge is required.

1. *Create a Python script that interacts with an AWS DynamoDB table. The script should be able to write an item to the table and read all items from the table.*

## *1.1 Setup AWS Credentials:*

- *Use the boto3 library to create a session with your AWS credentials.*
- *As we use a temp session, you will need to use: `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN`*
- *Ensure you have the necessary permissions to access DynamoDB.*
- *We will use region 'us-east-1'*

## *2.1 Create a DynamoDB Client:*

- *Initialize a DynamoDB client using the session created.*

## *3.1 Write an Item to the Table:*

- *Write a function `write_item_to_table` that adds an item to the DynamoDB table named `calculations-table`. The item should include the following attributes:*
  - *timestamp: Current timestamp in ISO format.*
  - *num1: A number (e.g., 5).*
  - *num2: A number (e.g., 3).*
  - *operation: A string representing the operation (e.g., '\*').*
  - *result: The result of the operation (e.g., 15 for multiplication).*

## *4.1 Read Items from the Table:*

- *Write a function `read_items_from_table` that reads all items from the `calculations-table` and prints them (Optional: Print also in a formatted JSON structure).*

## *5.1 Execute the Functions:*

- *Call the `write_item_to_table` function to add an item to the table.*
- *Call the `read_items_from_table` function to read and print all items from the table.*

- Alternatively, copy the following code, and paste it in your IDE:

**Note:** Replace **YOUR\_AWS\_ACCESS\_KEY\_ID**, **YOUR\_AWS\_SECRET\_ACCESS\_KEY**, and **YOUR\_AWS\_SESSION\_TOKEN** with your actual AWS credentials, similarly, replace the region-name with the region-name you are using.

```
import boto3
import json
from datetime import datetime

# Create a session using your AWS credentials
session = boto3.Session(
    aws_access_key_id='YOUR_AWS_ACCESS_KEY_ID',
    aws_secret_access_key='YOUR_AWS_SECRET_ACCESS_KEY',
    aws_session_token='YOUR_AWS_SESSION_TOKEN',
    region_name='us-east-1'
)
# Create a DynamoDB client
dynamodb = session.client('dynamodb')

# Function to write items from the calc-history table
def write_item_to_table():
    # Add data to the calc-history table
    response = dynamodb.put_item(
        TableName='calculations-table',
        Item={
            'timestamp': {'S': datetime.now().isoformat()},
            'num1': {'N': '5'},
            'num2': {'N': '3'},
            'operation': {'S': '*'},
            'result': {'N': str(5*3)}
        }
    )
    print("Data added to calc-history table:", response)

# Function to read items from the calc-history table
def read_items_from_table():
    response = dynamodb.scan(TableName='calculations-table')
    items = response.get('Items', [])
    for item in items:
        print(item)
        print(json.dumps(item, indent=4))

# Call the function to write item
write_item_to_table()
# Call the function to read items
read_items_from_table()
```

3. Install boto3 module.

Verify if boto3 is installed, in your terminal run:

```
pip show boto3
```

if you're getting a warning: WARNING: Package(s) not found: boto3

You should install boto3 by running the following command in your terminal:

```
pip install boto3
```

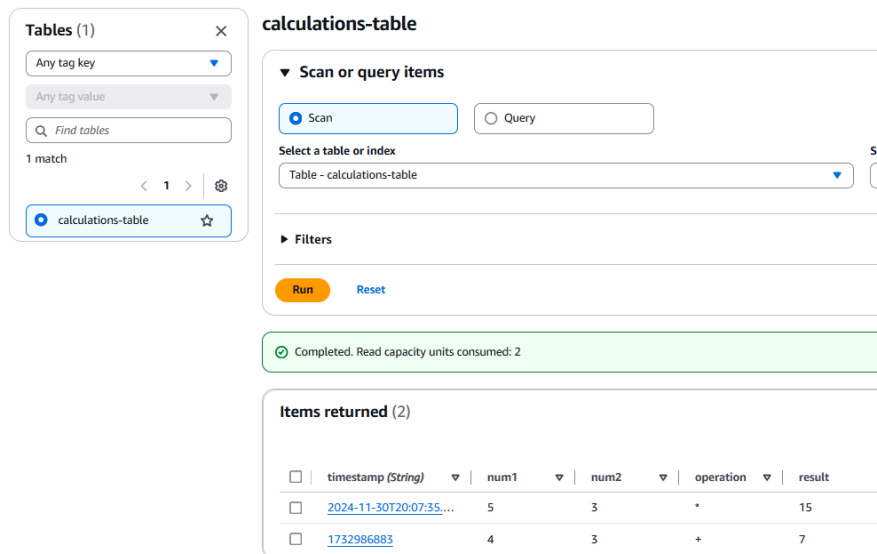
4. Run the code. You should get the following successful reply:

```
Data added to calc-history table: {'ResponseMetadata': {'RequestId':
'FG9EAU0P5QLKDDSIHQKPLLCCFVVV4KQNSO5AEMVJF66Q9ASUAAJG', 'HTTPStatusCode': 200,
'HTTPHeaders': {'server': 'Server', 'date': 'Sat, 30 Nov 2024 18:07:38 GMT', 'content-type': 'application/x-amz-json-1.0',
'content-length': '2', 'connection': 'keep-alive', 'x-amzn-requestid':
'FG9EAU0P5QLKDDSIHQKPLLCCFVVV4KQNSO5AEMVJF66Q9ASUAAJG', 'x-amz-crc32': '2745614147'},
'RetryAttempts': 0}}
```

```
{'result': {'N': '15'}, 'operation': {'S': '*'}, 'num2': {'N': '3'}, 'num1': {'N': '5'}, 'timestamp': {'S': '2024-11-
30T20:07:35.431889'}}
{
  "result": {
    "N": "15"
  },
  "operation": {
    "S": "*"
  },
  "num2": {
    "N": "3"
  },
  "num1": {
    "N": "5"
  },
  "timestamp": {
    "S": "2024-11-30T20:07:35.431889"
  }
}
{'result': {'N': '7'}, 'operation': {'S': '+'}, 'num2': {'N': '3'}, 'num1': {'N': '4'}, 'timestamp': {'S': '1732986883'}}
{
  "result": {
    "N": "7"
  },
  "operation": {
    "S": "+"
  },
  "num2": {
    "N": "3"
  },
  "num1": {
    "N": "4"
  },
  "timestamp": {
    "S": "1732986883"
  }
}
```

## Task 4: Explore the new item created using AWS SDK (Boto3) from the AWS Console

1. In the AWS Console search box to the right of **Services**, search for and choose **DynamoDB** to open the **DynamoDB** console.
2. In the left menu, select **Tables** menu and select the `calculations-table`.
3. Press the **Explore table items** option. You should now see both the item you added manually from the AWS console and the item added from your Python script.



## Task 5: Explore the new item created using AWS Lambda using SDK (Boto3)

In this task, you will interact with DynamoDB using an **AWS Lambda** function. Both Python knowledge and **AWS Lambda** function are required to finish this task.

1. *Create an AWS Lambda function, that will support HTTP GET Method, and it will read a **DynamoDB** table. It will get the table name from a parameter called `table_name` and will return all table content.*

Alternatively, use the following code, in your lambda function:

```
import json
import boto3
```



```
def lambda_handler(event, context):
    # Get the table name from queryStringParameters in the event
    table_name = event.get('queryStringParameters', {}).get('table_name')

    if not table_name:
        return {
            "statusCode": 400,
            "body": json.dumps({"error": "table_name is required"})
        }

    # Create a DynamoDB client
    dynamodb = boto3.client('dynamodb')

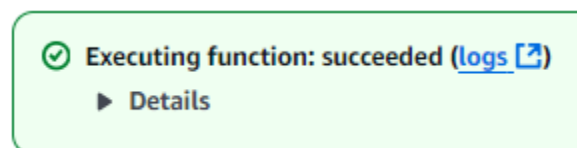
    try:
        # Scan the table
        response = dynamodb.scan(Table=table_name)
        items = response.get('Items', [])

        return {
            "statusCode": 200,
            "body": json.dumps({"items": items})
        }
    except Exception as e:
        return {
            "statusCode": 500,
            "body": json.dumps({"error": str(e)})
        }
```

- Click **Test** tab, and update the following JSON in the **Event JSON** panel:

```
{
  "queryStringParameters": {
    "table_name": "calculations-table"
  }
}
```

- Click **Test**. The function will execute and should finish successfully:



- Click the **Details** drop down and inspect the execution. You should see the response, summary, and log output.

## Activity complete

Congratulations! You have successfully completed the activity.