

# AWS Lambda with DynamoDB:

DynamoDB: Amazon DynamoDB is a serverless, NoSQL, fully managed database that provides fast and predictable performance with seamless scalability. It is a part of the Amazon Web Services (AWS) cloud platform and allows you to store and retrieve data at any scale.

## Create DynamoDB

[] select 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.  
  
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.  
   
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.  
   
1 to 255 characters and case sensitive.

[] in **Table name** we can mention whatever.

[] in **Partition Key** what we mentioned in code, in item which we mentioned first word that name we should mention. (Example:

```
item = {  
    'roll-number': '101', # Partition Key  
    'name': 'John Doe', # Other attributes  
    'age': 21,  
    'course': 'Computer Science'
```

```
} )
```

[] select create table.

## Create lambda Function.

[] Create an AWS Lambda function with Python 3.10.

[] Once lambda is created. Add Below Code and Deploy.

[] On the Configuration tab, choose General configuration, and then choose Edit. Set Timeout to 10 seconds and then choose Save.

[] add permission to lambda role, i.e. CloudWatch full access and dynamoDB (go to IAM, select role, choose which we create lambda add the permissions).

### CODE: -

```
import json
import boto3
import logging
from botocore.exceptions import ClientError

# Set up logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

# Initialize DynamoDB client with the correct region
dynamodb = boto3.resource('dynamodb', region_name='us-west-2') # Replace with
your region
table_name = 'student-123' # Replace with your actual table name
table = dynamodb.Table(table_name)

# Get the region using the session (for logging purposes)
region_name = boto3.Session().region_name

def lambda_handler(event, context):
    # Log the region and table name being used
    logger.info(f"Attempting to access DynamoDB table: {table_name} in region
{region_name}")
```

```

# Sample data to insert into DynamoDB
item = {
    'roll-number': '101', # Partition Key
    'name': 'John Doe', # Other attributes
    'age': 21,
    'course': 'Computer Science'
}

try:
    # Insert data into DynamoDB
    table.put_item(Item=item)
    logger.info(f"Item successfully inserted into {table_name}")

    # Return a successful response
    return {
        'statusCode': 200,
        'body': json.dumps('Data inserted successfully')
    }

except ClientError as e:
    # Handle any errors that occur during the insertion
    error_message = e.response['Error']['Message']
    logger.error(f"Error inserting data: {error_message}")

    return {
        'statusCode': 500,
        'body': json.dumps(f'Error inserting data: {error_message}')
    }

```

[] After test the code we will get tables in DynamoDB.

[] in DynamoDB go to table and select which table we created.

[] select explore table item.

Items returned (1)

Actions

Create item

<1>

<input type="checkbox"/>	roll-number (String)	▼	age	▼	course	▼	name	▼
<input type="checkbox"/>	<a href="#">101</a>		21		Computer S...		John Doe	

Edit item

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](#)

FormJSON view

Attributes

Add new attribute

Attribute name	Value	Type	
roll-number - Partition key	101	String	
age	21	Number	Remove
course	Computer Science	String	Remove
name	John Doe	String	Remove

CancelSaveSave and close