

# Experiment 06 - Cloud Computing

December 28, 2022

## Aim

To create table in DynamoDB and perform CRUD operations through console and AWS SDK along with exploring various ways to query data in DynamoDB

## Theory

- **DynamoDB** : Amazon DynamoDB is a fully managed proprietary NoSQL database service that supports key-value and document data structures and is offered by Amazon.com as part of the Amazon Web Services portfolio. DynamoDB exposes a similar data model to and derives its name from Dynamo, but has a different underlying implementation.
- **Queries in Dynamo** : The Query operation in Amazon DynamoDB finds items based on primary key values. You must provide the name of the partition key attribute and a single value for that attribute. Query returns all items with that partition key value. Optionally, you can provide a sort key attribute and use a comparison operator to refine the search results.
- **Backups in Dynamo** : Point-in-time recovery (PITR) provides continuous backups of your DynamoDB table data. When enabled, DynamoDB maintains incremental backups of your table for the last 35 days until you explicitly turn it off.

## Results

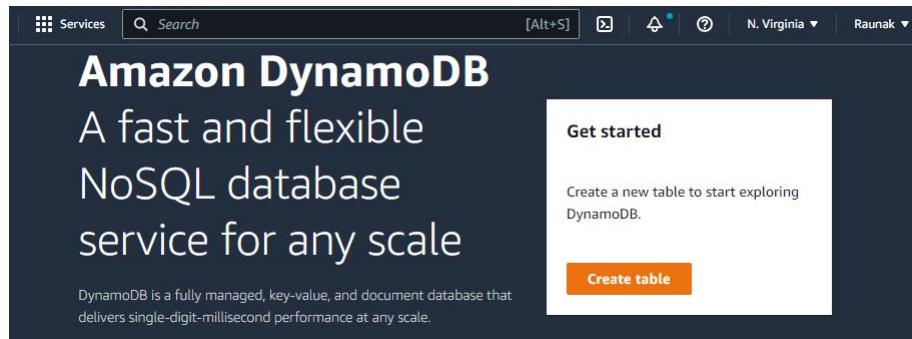


Figure 1: Start by creating a table by searching for DynamoDB Service from the search bar

**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 for scalability and availability.

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.

Figure 2: Name the table and name a primary key

**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.

Figure 3: Create the table using the default settings

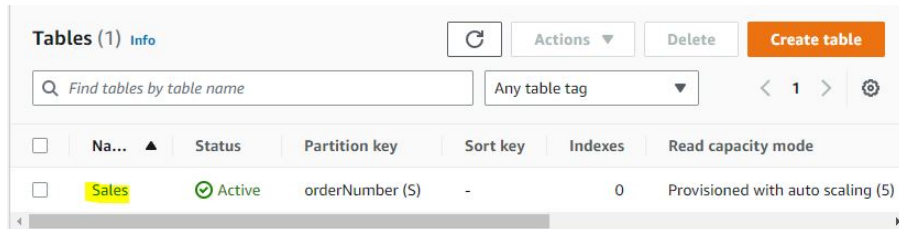


Figure 4: Successful creation of the table in Dynamo

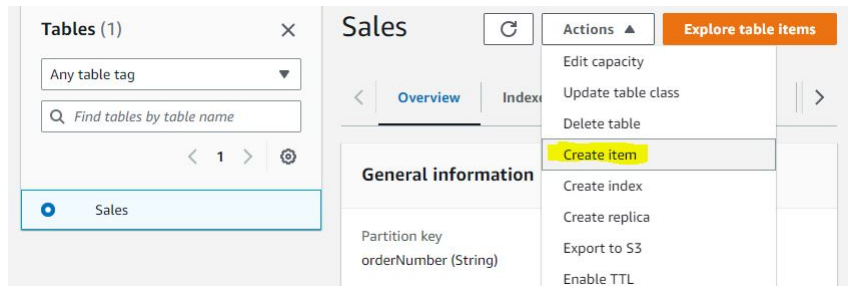


Figure 5: After selecting the table, one can create the item

The screenshot displays the 'Attributes' form for creating a new item. It features a table with three columns: 'Attribute name', 'Value', and 'Type'. The first row shows 'orderNumber - Partition key' with a value of '\$500' and a type of 'String'. Subsequent rows show 'subscriptionNumber' (True, Binary), 'quantity' (2, Number), 'emailId' (asdf@socalledsite.com, String), and 'product' (Bag, String). Each row has a 'Remove' button to its right. At the top right, there's a button to 'Add new attribute'. At the bottom right, there are 'Cancel' and 'Create item' buttons.

Attribute name	Value	Type
orderNumber - Partition key	\$500	String
subscriptionNumber	True	Binary
quantity	2	Number
emailId	asdf@socalledsite.com	String
product	Bag	String

Figure 6: Name the attributes with values and note the respective data types



Figure 7: JSON View can be selected too

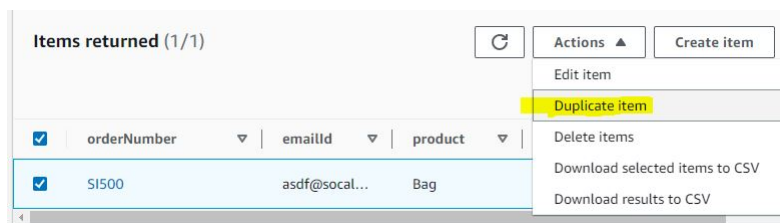


Figure 8: Duplication of the items can be done and specific changes can be made in values

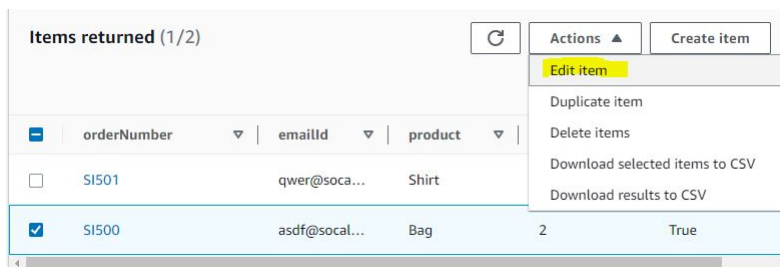


Figure 9: Successful replicate value created

Attributes

Add new attribute ▼

Attribute name	Value	Type	
orderNumber - Partition key	\$1500	String	
emailId	asdf@socalledsite.com	String	Remove
product	Bag	String	Remove
quantity	3	Number	Remove
subscriptionNumber	True	Binary	Remove

Cancel

Save changes

Figure 10: Values can be edited too

Details

Name

dynamoTesting

Limit of 60 characters, alphanumeric, and unique per user.

Description - optional

Limit 200 characters.

Environment type [Info](#)

Determines what the Cloud9 IDE will run on.

☒ New EC2 instance  
 Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.

☐ Existing compute  
 You have an existing instance or server that you'd like to use.

Figure 11: Managing the dynamoDB table using cloud9 environment

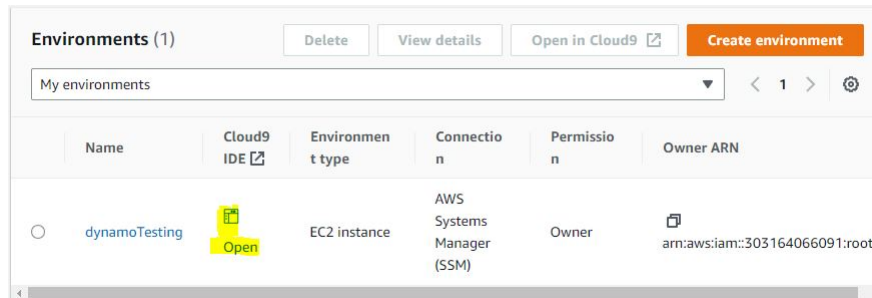


Figure 12: Open option can be selected that opens environment in browser tab

```
python2 - "ip-172-31-25-1" x Immediate x (+)
ec2-user:~/environment $ aws dynamodb list-tables
{
  "TableNames": [
    "Sales"
  ]
}
ec2-user:~/environment $
```

Figure 13: The terminal based console can be used and checked for existing tables using `aws dynamodb list-tables` command

```
dynamodbWrite.py x (+)
1 import boto3
2
3 dynamodb = boto3.resource('dynamodb')
4 table = dynamodb.Table('Sales')
5
6 with table.batch_writer() as batch:
7     batch.put_item(Item={"OrderNo": "SI5003", "Email": "john@handson.cloud",
8       "Product": "Dress", "Quantity": "1", "SubscriptionPremium": "true" })
9     batch.put_item(Item={"OrderNo": "SI5004", "Email": "jahdsw@handson.cloud",
10      "Product": "Shirt", "Quantity": "2", "SubscriptionPremium": "false" })
11     batch.put_item(Item={"OrderNo": "SI5005", "Email": "aennniri@gmail.com",
12      "Product": "Earphones", "Quantity": "3", "SubscriptionPremium": "true" })
13
14 print(batch)
```

Figure 14: Creation of the new file and following code inserts values in table using boto3

```

ec2-user:~/environment $ sudo python3 -m pip install boto3
WARNING: Running pip install with root privileges is generally not a good idea. Try 'python3 -m pip install --user' instead.
Collecting boto3
  Downloading boto3-1.26.50-py3-none-any.whl (132 kB)
    | 132 kB 12.9 MB/s
Collecting botocore<1.30.0,>=1.29.50
  Downloading botocore-1.29.50-py3-none-any.whl (10.3 MB)
    | 10.3 MB 25 kB/s
Collecting s3transfer<0.7.0,>=0.6.0
  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)
    | 79 kB 10.1 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.7/site-packages (from boto3) (1.0.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.7/site-packages (from botocore<1.30.0,>=1.29.50->boto3) (2.8.2)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/local/lib/python3.7/site-packages (from botocore<1.30.0,>=1.29.50->boto3) (1.26.13)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.30.0,>=1.29.50->boto3)
Installing collected packages: botocore, s3transfer, boto3
  Attempting uninstall: botocore
    Found existing installation: botocore 1.29.46
    Uninstalling botocore-1.29.46:
      Successfully uninstalled botocore-1.29.46
Successfully installed boto3-1.26.50 botocore-1.29.50 s3transfer-0.6.0
ec2-user:~/environment $ python dynamodbWrite.py

```

Figure 15: Installation of boto SDK and file execution can be seen in the highlighted commands

Items returned (5)

Actions

Create item

< 1 >

<input type="checkbox"/>	orderNumber	emailId	product	quantity	subscriptionNumb
<input type="checkbox"/>	SI5003	john@hand...	Dress	1	True
<input type="checkbox"/>	SI5005	aennniri@g...	Earphones	3	True
<input type="checkbox"/>	SI5004	jahdsw@ha...	Shirt	2	False
<input type="checkbox"/>	SI500	asdf@socal...	Bag	3	True
<input type="checkbox"/>	SI501	qwer@soca...	Shirt	1	True

Figure 16: Successful addition of the records

▼ Scan or query items

☒ Scan
 ☐ Query

Select a table or index  
 Table - Sales ▼

Select attribute projection  
 Specific attributes ▼

Specific attributes to project  
 product

Add attribute

orderNumber ✕ emailId ✕

▼ Filters

Attribute name	Type	Condition	Value	
product	String ▼	Equal to ▼	Shirt	Remove

Add filter

Run

Reset

Figure 17: Scanning for items can be done by specifying the attributes

Items returned (2)

↺

Actions ▼

Create item

<

1

>

⚙

✕

	orderNumber ▼	emailId ▼
<input type="checkbox"/>	SI5004	jahdsw@handson.cloud
<input type="checkbox"/>	SI501	qwer@socalledsite.com

Figure 18: Retrieved items after execution of the scan



▼ Scan or query items

☐ Scan
 ☒ Query

Select a table or index: Table - Sales
 Select attribute projection: All attributes

orderNumber (Partition key): SI500

► Filters

Figure 19: Query can be used and retrieving the item can be done using primary key

Items returned (1)

<input type="checkbox"/>	orderNumber	emailId	product	quantity	subscriptionNur
<input type="checkbox"/>	SI500	asdf@socal...	Bag	3	True

Figure 20: Retrieved items after execution of the query

```

dynamodbWrite.py x
import boto3
from boto3.dynamodb.conditions import Key, Attr

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Sales')

table.update_item(
    Key={'orderNumber': 'SI5005'},
    UpdateExpression="set quantity = :r",
    ExpressionAttributeValues={
        ':r': '5',
    },
    ReturnValues="UPDATED_NEW"
)

table.delete_item(Key = {'orderNumber': 'SI5003'})

response = table.scan()
response['Items']
print(response)

response = table.scan(FilterExpression=Attr('product').eq('Shirt'))
print("The query returned the following items:")
for item in response['Items']:
    print(item)
  
```

Figure 21: Editing the existing item using the SDK from Environment

```

ec2-user:~/environment $ python dynamoDBWrite.py
{'Items': [{'product': 'Earphones', 'quantity': '5', 'emailId': 'aennniri@gmail.com', 'orderNumber': 'SI5005', 'subscriptionNumber': 'True'}, {'product': 'Shirt', 'quantity': '2', 'emailId': 'jahdsw@handson.cloud', 'orderNumber': 'SI5004', 'subscriptionNumber': 'False'}, {'product': 'Bag', 'quantity': '3', 'emailId': 'asdf@socalledsite.com', 'orderNumber': 'SI500', 'subscriptionNumber': 'Binary(b'\x0b\x0e')'}, {'product': 'Shirt', 'quantity': '1', 'emailId': 'qwer@socalledsite.com', 'orderNumber': 'SI501', 'subscriptionNumber': 'Binary(b'\x0b\x0e')'}], 'Count': 4, 'ScannedCount': 4, 'ResponseMetadata': {'RequestId': 'QR0HQSMGF09BLDMCSJ0ISEVH17VW4KQNS0SAEMVJF66Q9ASUAAJG', 'HTTPStatusCode': 200, 'HTTPHeaders': {'server': 'Server', 'date': 'Tue, 17 Jan 2023 15:22:13 GMT', 'content-type': 'application/x-amz-json-1.0', 'content-length': '635', 'connection': 'keep-alive', 'x-amzn-requestid': 'QR0HQSMGF09BLDMCSJ0ISEVH17VW4KQNS0SAEMVJF66Q9ASUAAJG', 'x-amz-crc32': '3548423936'}, 'RetryAttempts': 0}}
The query returned the following items:
{'product': 'Shirt', 'quantity': '2', 'emailId': 'jahdsw@handson.cloud', 'orderNumber': 'SI5004', 'subscriptionNumber': 'False'}
{'product': 'Shirt', 'quantity': '1', 'emailId': 'qwer@socalledsite.com', 'orderNumber': 'SI501', 'subscriptionNumber': 'Binary(b'\x0b\x0e')'}

```

Figure 22: Successful execution of the script

Items returned (4)

	orderNumber	emailId	product	quantity	subscriptionNumber
<input type="checkbox"/>	SI5005	aennniri@g...	Earphones	5	True
<input type="checkbox"/>	SI5004	jahdsw@ha...	Shirt	2	False
<input type="checkbox"/>	SI500	asdf@socal...	Bag	3	True
<input type="checkbox"/>	SI501	qwer@soca...	Shirt	1	True

Figure 23: Changes of the tuple can also be seen on the respective dashboard

Delete "dynamoTesting"?

Delete **dynamoTesting** environment permanently? This action cannot be undone.

*Proceeding with this action will permanently delete this environment, including all settings, associated user data, and uncommitted code.*

To avoid accidental deletions we ask you to provide additional written consent.

Type **Delete** to agree.

Delete

Cancel
Delete

Figure 24: Deletion of the cloud9 environment

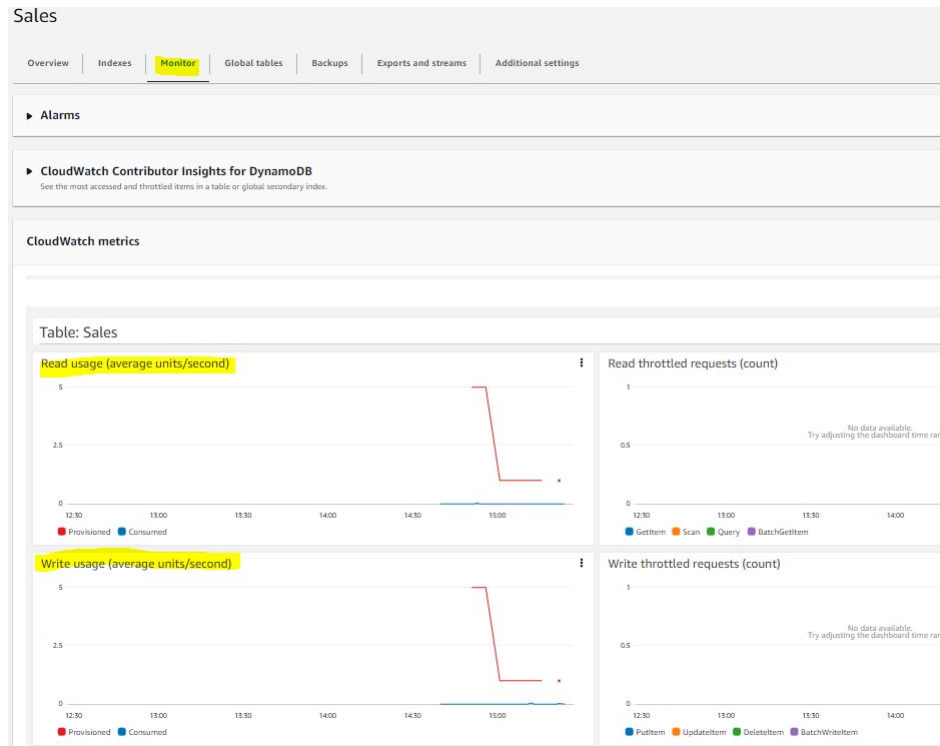


Figure 25: Monitoring of the table can be done which provides analytics

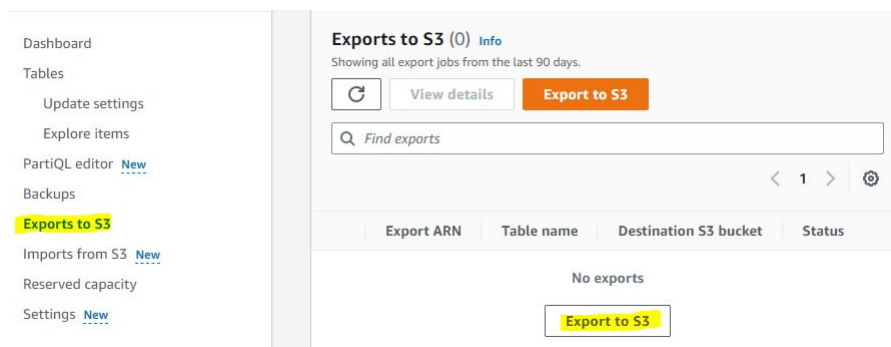


Figure 26: The table can be exported to S3 Bucket

**Export details** [Info](#)

Source table

Q Sales X

Sales

Sales

Q Enter bucket name View Browse S3

Format: s3://bucket/prefix

Figure 27: Exporting the table by first naming it

**Export details** [Info](#)

Source table

Q Sales X

Note that you can't export archived tables to Amazon S3.

Destination S3 bucket

Q s3://generic-buckets X View Browse S3

Format: s3://bucket/prefix

S3 bucket owner

☒ This AWS account (303164066091)

☐ A different AWS account

► Additional settings

Cancel Export

Figure 28: Create a S3 bucket prior and browse it through the panel of export

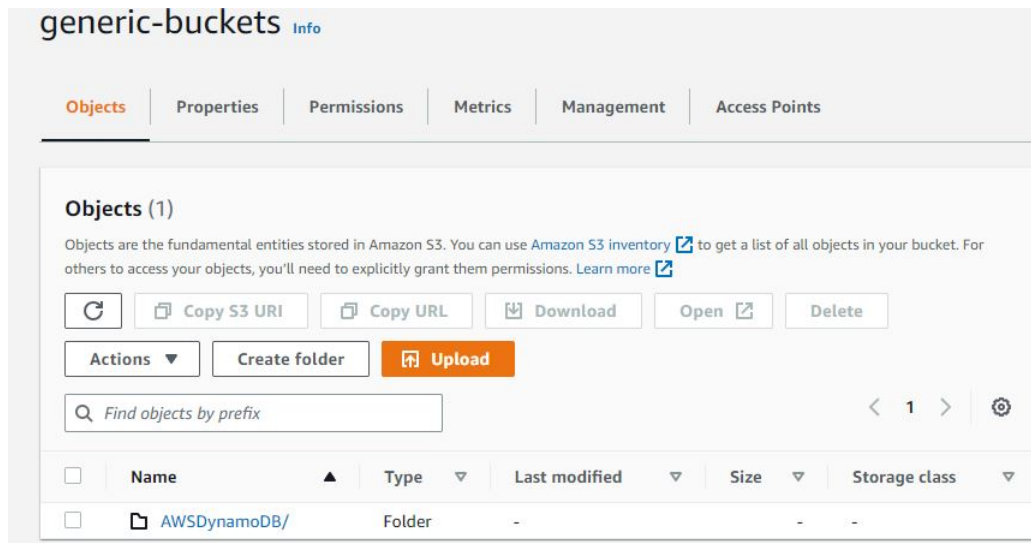


Figure 29: Successful table export in the existing bucket

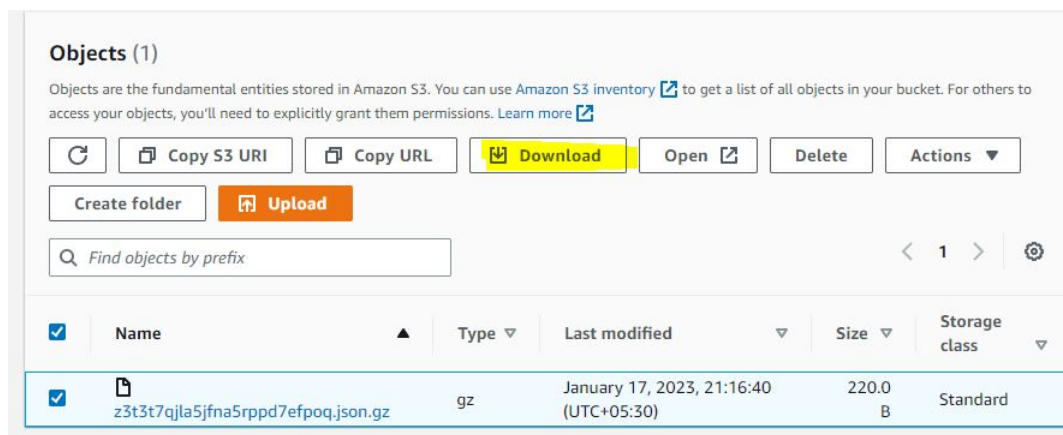


Figure 30: The data can be downloaded by browsing to the object in the exported directory

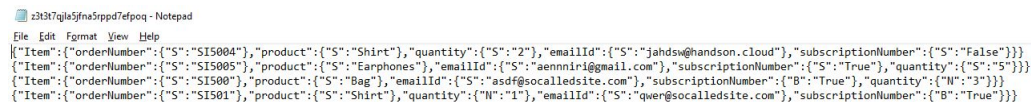


Figure 31: After unzipping the downloaded file the JSON file can be viewed

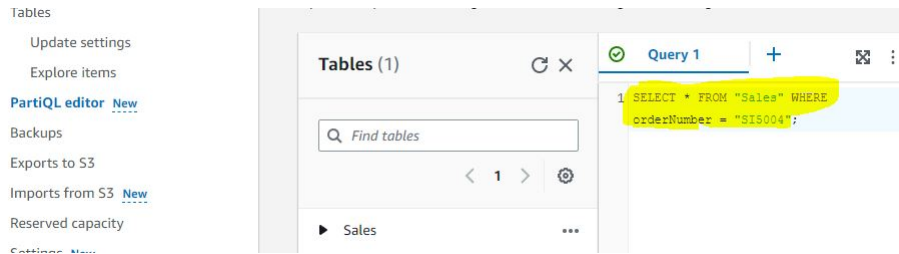


Figure 32: PartiQL can be used to execute queries respectively

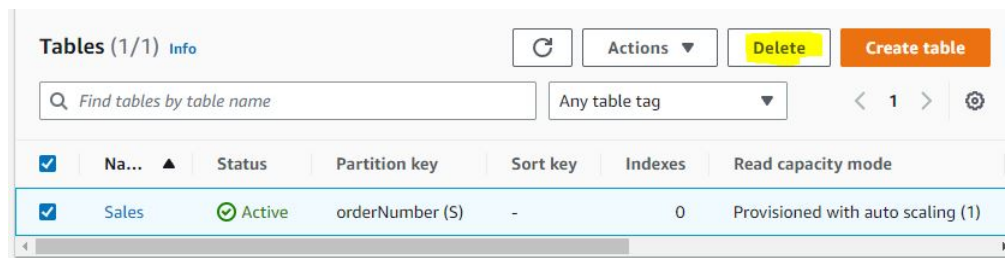


Figure 33: Deletion of the table can be done respectively

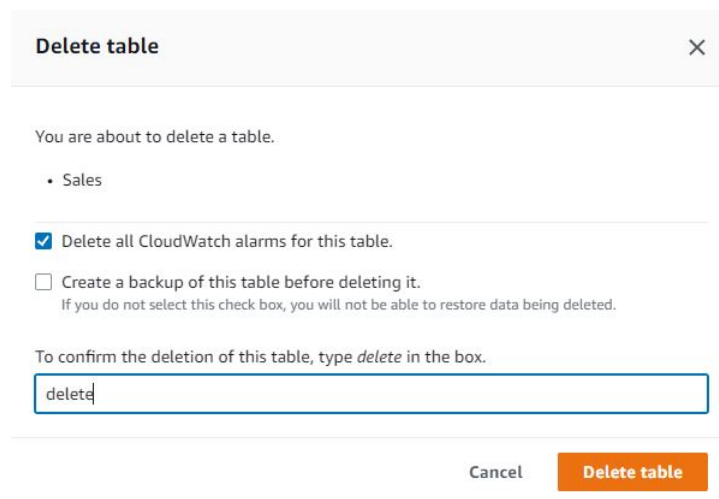


Figure 34: Deletion of the table by confirming the latter step

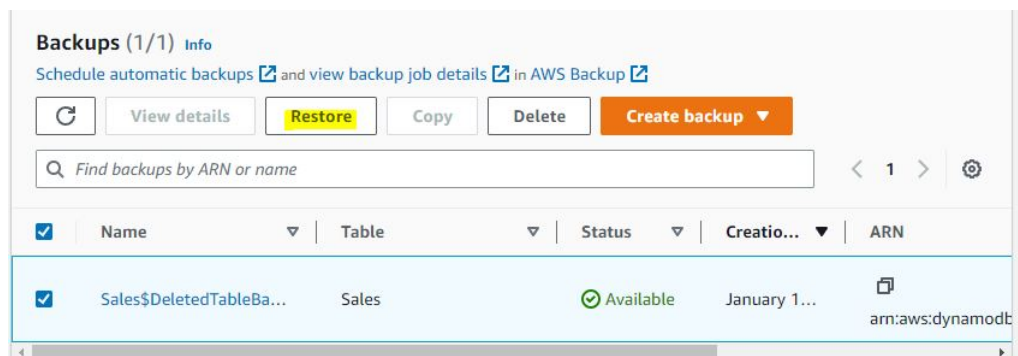


Figure 35: In the section of backups found on left panel the retrieval of the deleted data can be done. Since the option of PITR is enabled the backup cannot be deleted. It will be automatically deleted in the span of 35 days

## Conclusion

This experiment is successful demonstration of many ways of operating with the DynamoDB. The service can be used to create the NoSQL based tables. The deletion process can be done along with restoration using the backups seamlessly and using the PITR that deletes the backup in the span of 35 days. The demonstration of Cloud9 environment is also done which effectively can be used to connect the table. The boto SDK is also demonstrated for manipulation of the table by leveraging Python programming language. The querying can be done using PartiQL in form of typical SQL as well as exporting to S3 Bucket can also be done. The execution of the all the process has been successfully shown in the results section of this experiment respectively.