Cycle 08 AWS Homework

1. Practice Creating Indexes with Boto3

Local Secondary Index (LSI)

- An LSI is created at the same time as the base table.
- It uses the **same Partition Key** as the table but allows a **different Sort Key**.
- Purpose: Useful for querying the same data with an alternate sort order.
- Example case: Table has `UserID` as Partition Key, `OrderID` as Sort Key. An LSI might use `OrderDate` as the alternate Sort Key, enabling queries like *"show all orders for UserID sorted by OrderDate"*.

Global Secondary Index (GSI)

- A GSI can be created at any time, even after the base table is created.
- It allows both a **different Partition Key and Sort Key** from the base table.
- Purpose: Helps create completely new query patterns.
- Example case: Table keyed by `UserID`, but a GSI with `Email` as Partition Key allows queries like *"fetch user data by email address"*.

Steps:

1. Define a base table with keys.
2. Add LSIs at creation time by specifying alternate Sort Keys.
3. Add GSIs either during or after creation to support new query patterns.
4. Share the syntax with the group for peer learning.

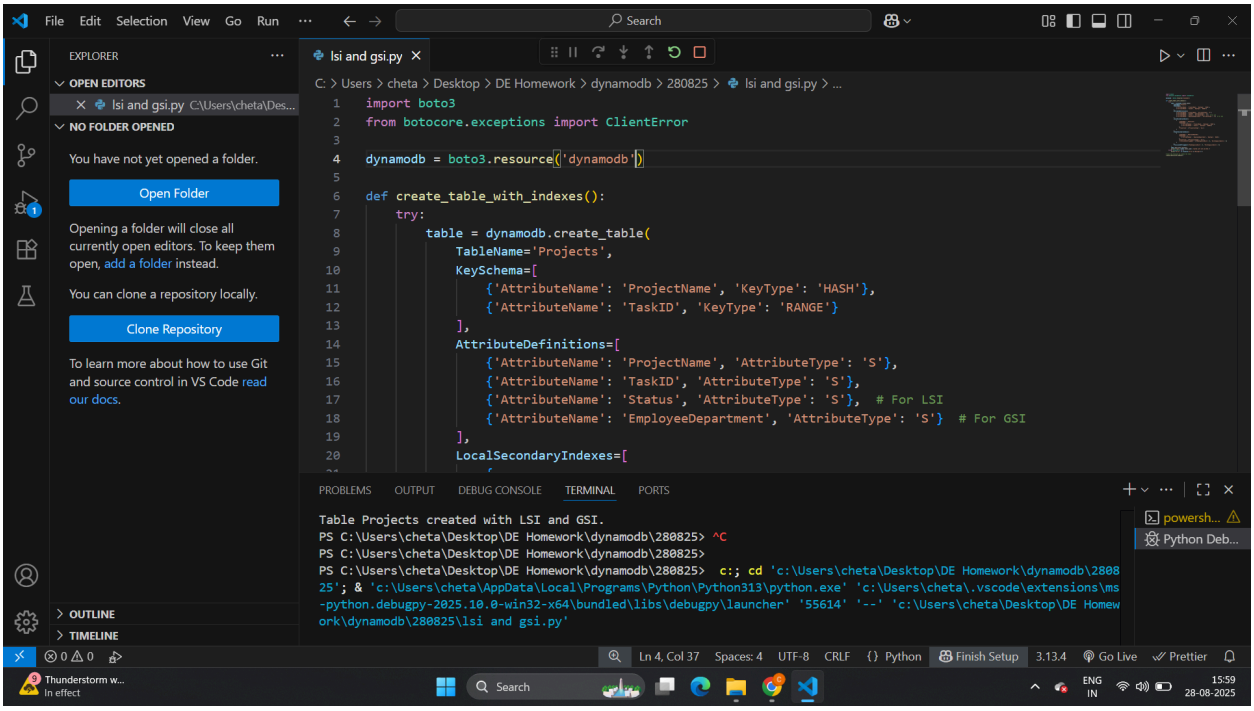
CODE:

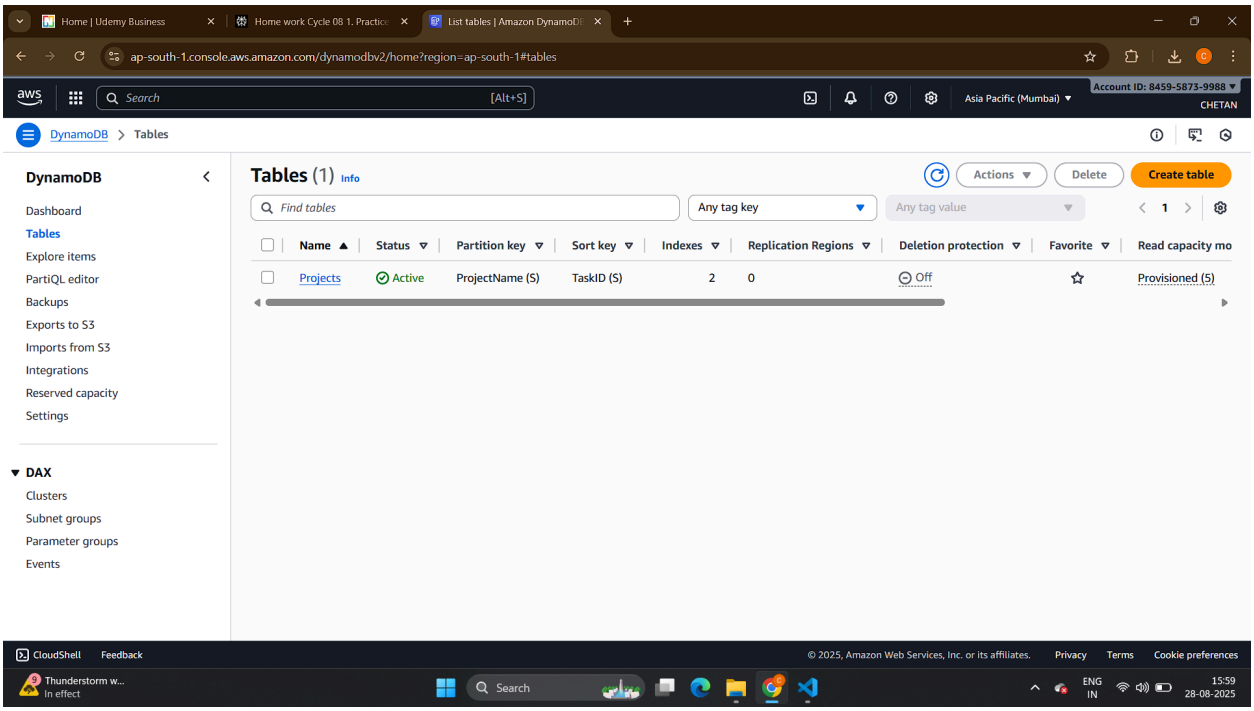
```
import boto3
from botocore.exceptions import ClientError
```

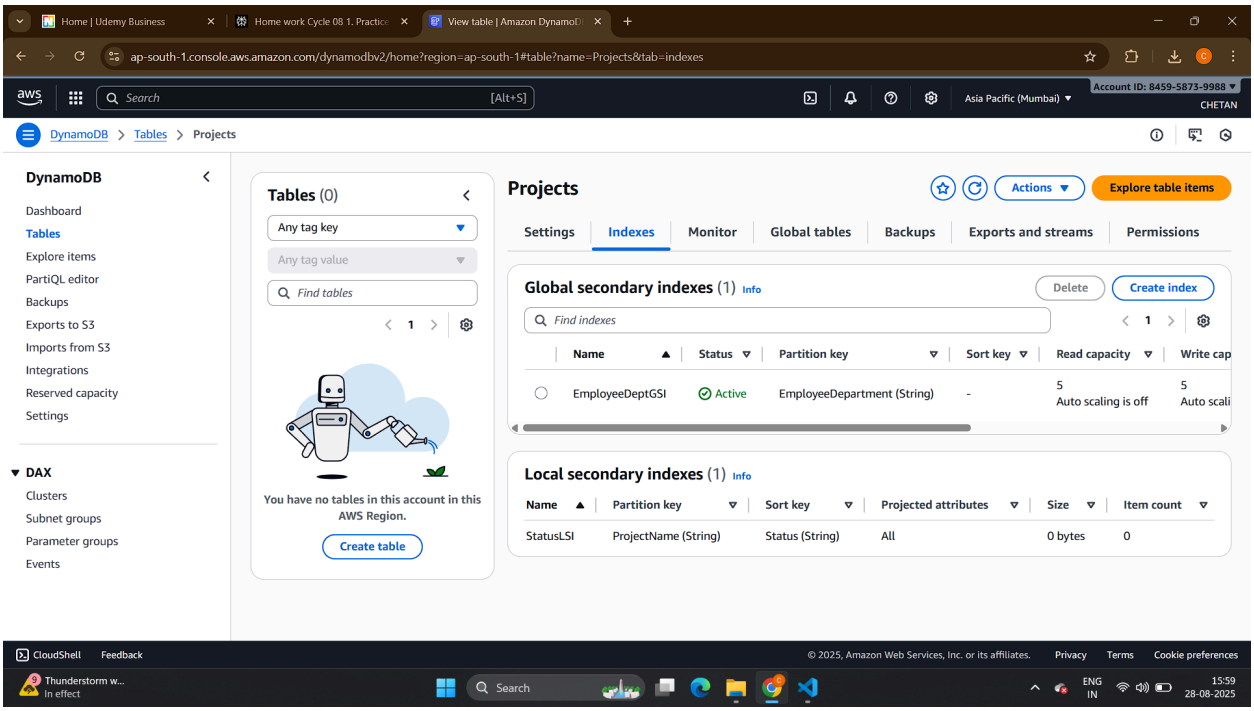
```
dynamodb = boto3.resource('dynamodb')

def create_table_with_indexes():
    try:
        table = dynamodb.create_table(
            TableName='Projects',
            KeySchema=[
                {'AttributeName': 'ProjectName', 'KeyType': 'HASH'},
                {'AttributeName': 'TaskID', 'KeyType': 'RANGE'}
            ],
            AttributeDefinitions=[
                {'AttributeName': 'ProjectName', 'AttributeType': 'S'},
                {'AttributeName': 'TaskID', 'AttributeType': 'S'},
                {'AttributeName': 'Status', 'AttributeType': 'S'},  # For LSI
                {'AttributeName': 'EmployeeDepartment', 'AttributeType': 'S'}  # For
GSI
            ],
            LocalSecondaryIndexes=[
                {
                    'IndexName': 'StatusLSI',
                    'KeySchema': [
                        {'AttributeName': 'ProjectName', 'KeyType': 'HASH'},
                        {'AttributeName': 'Status', 'KeyType': 'RANGE'}
                    ],
                    'Projection': {'ProjectionType': 'ALL'}
                }
            ],
            GlobalSecondaryIndexes=[
                {
                    'IndexName': 'EmployeeDeptGSI',
                    'KeySchema': [
                        {'AttributeName': 'EmployeeDepartment', 'KeyType': 'HASH'}
                    ],
                    'Projection': {'ProjectionType': 'ALL'},
                    'ProvisionedThroughput': {'ReadCapacityUnits': 5, 'WriteCapacity
```

```
Units': 5}
            }
        ],
        ProvisionedThroughput={'ReadCapacityUnits': 5, 'WriteCapacityUnits':
5}
    )
    table.wait_until_exists()
    print(f"Table {table.table_name} created with LSI and GSI.")
  except ClientError as e:
    print(f"Error: {e.response['Error']['Message']}")

# Call the function to create the table
create_table_with_indexes()
```

SCREENSHOTS:







2. Explore Billing Implications of Indexes

- DynamoDB billing is based on:

1. **Read Capacity Units (RCU)** – for reads.
2. **Write Capacity Units (WCU)** – for writes.
3. **Storage costs** – for table + indexes.

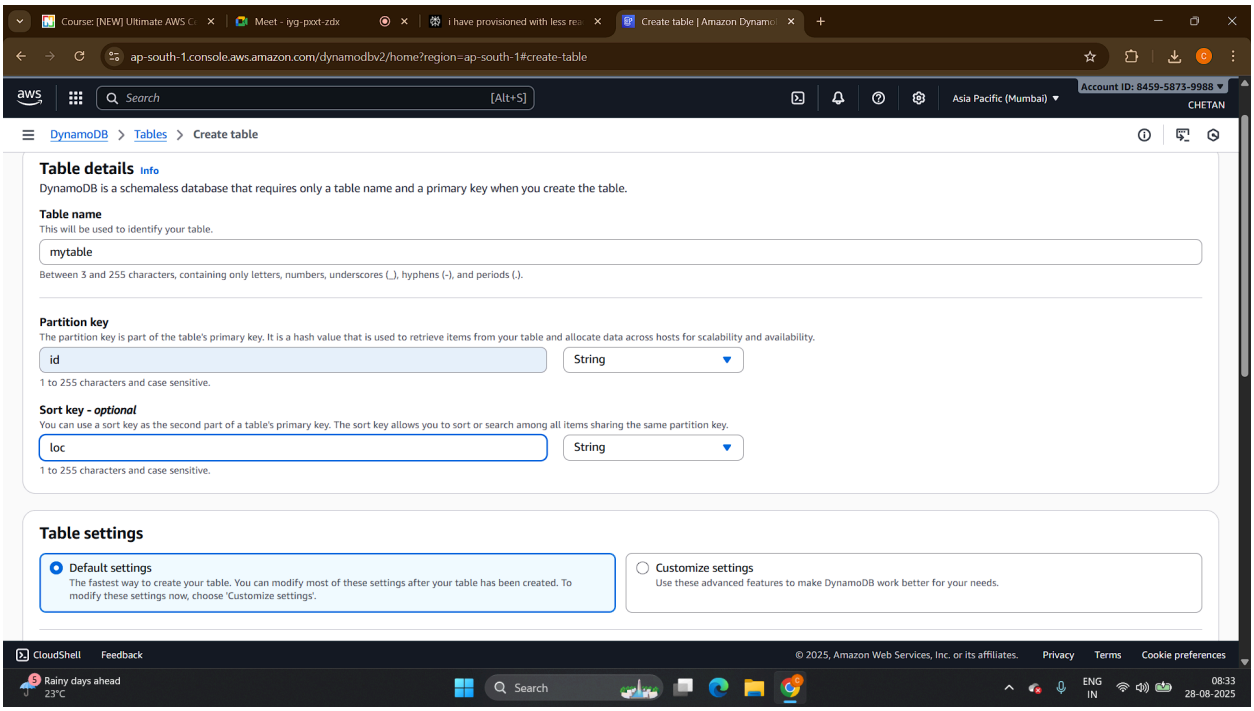
- **Without indexes:**
  - A `Scan` operation reads through the entire dataset, consuming RCUs proportional to table size.
  - This is inefficient and costly, especially as table size grows.
- **With LSIs/GSIs:**
  - Queries are targeted to indexed attributes.
  - Instead of scanning 1000 items, you can directly fetch matching items.
  - Saves RCUs by avoiding full table scans.
- **Trade-off:**
  - Indexes themselves take up storage.
  - Write operations cost more because every index must also be updated when the base table updates.
- **Conclusion:**
  - Indexes save cost at query time.
  - Extra cost appears in storage and write throughput.
  - For workloads with frequent reads and predictable query patterns, indexes lower overall cost.

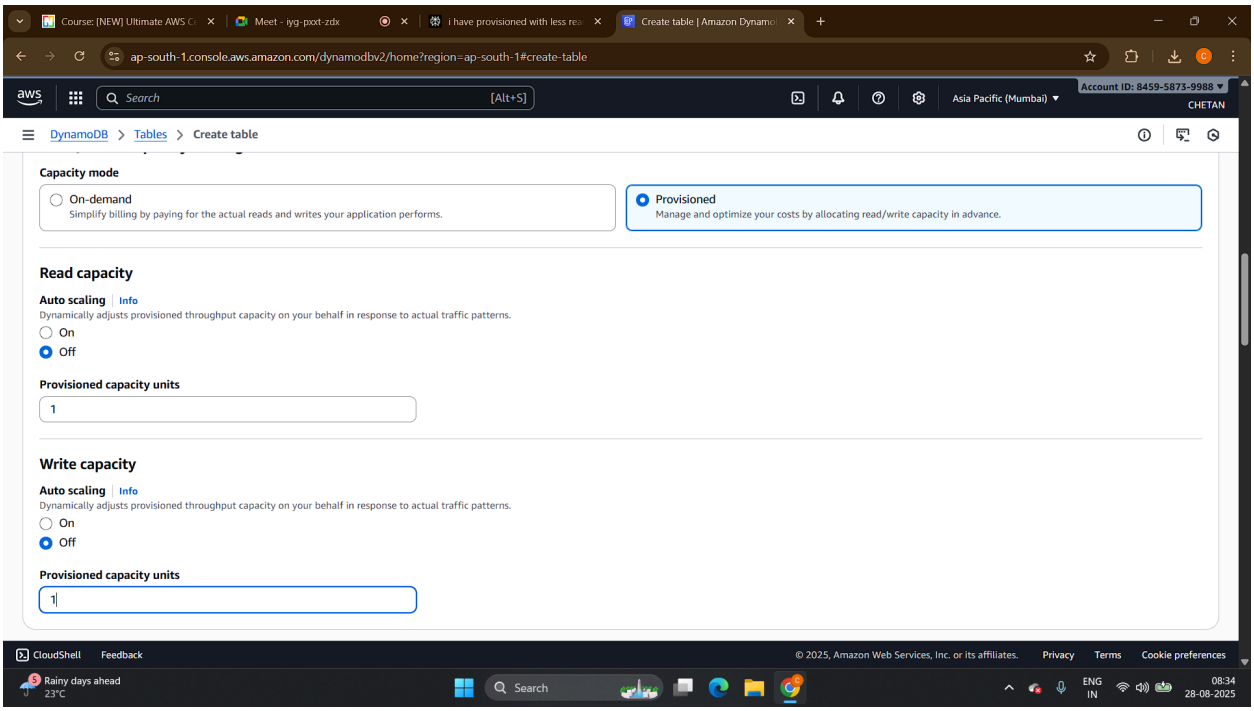
---

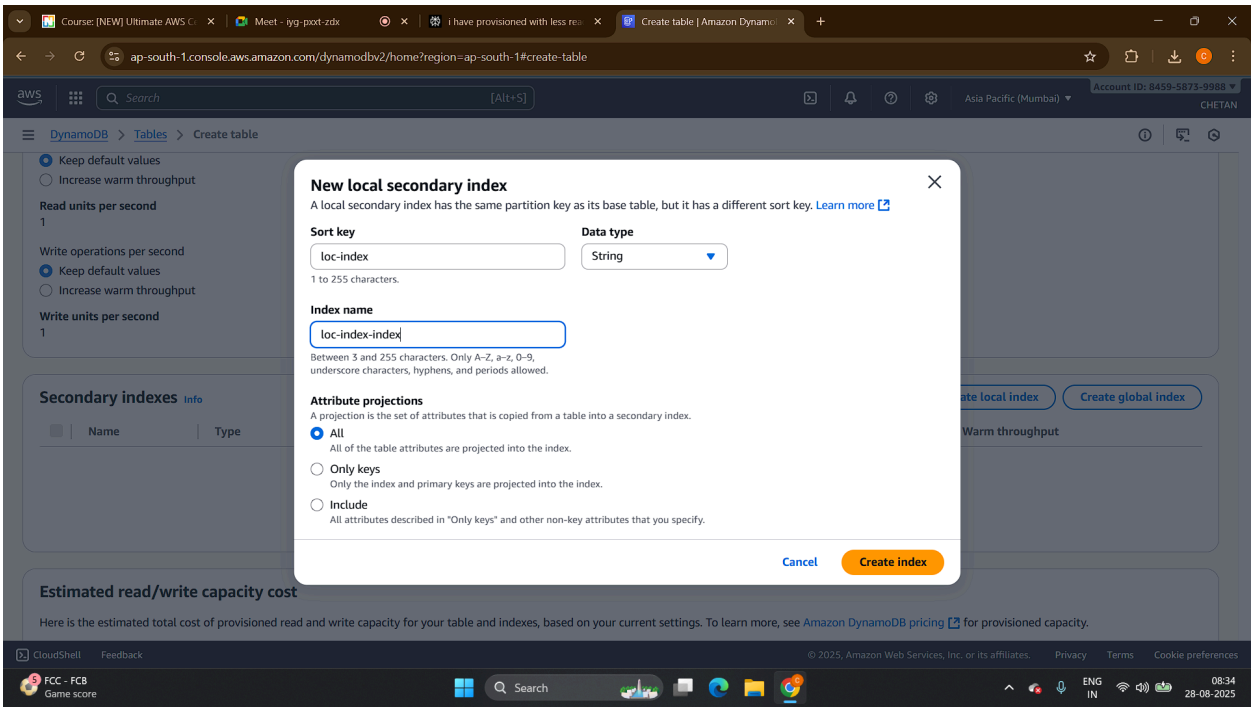
3. Syntax Sharing Task

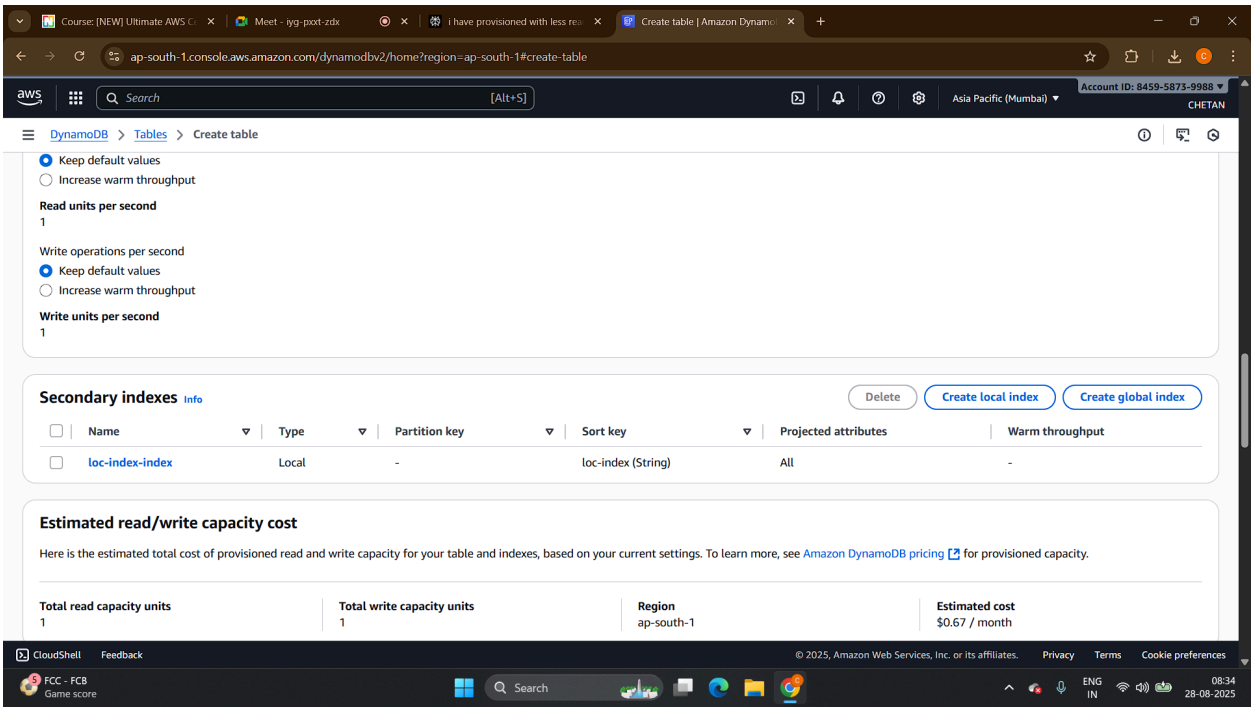
1. Write the boto3 syntax to create a table with an LSI.
2. Write the boto3 syntax to add a GSI.
3. Share the snippets with the group.

- Purpose: Collaborative learning. Everyone verifies correctness and understands variations in syntax.


Course: [NEW] Ultimate AWS C
Meet - iyg-pxxt-zdx
i have provisioned with less rea
Create table | Amazon Dynamo
ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#create-table
Search
[Alt+S]
Asia Pacific (Mumbai)
Account ID: 8459-5873-9988
CHETAN
DynamoDB
Tables
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.
mytable
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.
id
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.
loc
String
1 to 255 characters and case sensitive.
Table settings
Default settings
The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.
Customize settings
Use these advanced features to make DynamoDB work better for your needs.
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Rainy days ahead
23°C
Search
ENG
IN
08:33
28-08-2025


Course: [NEW] Ultimate AWS C
Meet - iyg-pxxt-zdx
i have provisioned with less rea
Create table | Amazon Dynamo
ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#create-table
Search
[Alt+S]
Asia Pacific (Mumbai)
Account ID: 8459-5873-9988
CHETAN
DynamoDB
Tables
Create table
Capacity mode
On-demand
Simplify billing by paying for the actual reads and writes your application performs.
Provisioned
Manage and optimize your costs by allocating read/write capacity in advance.
Read capacity
Auto scaling Info
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.
On
Off
Provisioned capacity units
1
Write capacity
Auto scaling Info
Dynamically adjusts provisioned throughput capacity on your behalf in response to actual traffic patterns.
On
Off
Provisioned capacity units
1
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Rainy days ahead
23°C
Search
ENG
IN
08:34
28-08-2025





4. CRUD & Batch Operations Script (Concept Explanation)

Steps:

1. **Create a table** with a composite key:
   - Partition Key: `ProjectName`.
   - Sort Key: `TaskID`.
   - This allows multiple tasks to be grouped under a single project.
2. **Batch insert items**:
   - Use batch_writer to add at least 10 items at once.
   - Efficient compared to inserting items one by one.
3. **CRUD functions:**
   - **Get Item:** Retrieve a specific task using its project and task ID.
   - **Update Item:** Modify an attribute (e.g., change task status from “TODO” to “IN_PROGRESS”).
   - **Delete Item:** Remove a task.
4. **Stretch Goal – Scan with filter:**
   - Use a non-key attribute (like `status`).
   - Example: return all tasks where status = “IN_PROGRESS”.
   - This demonstrates filtering beyond the primary key.

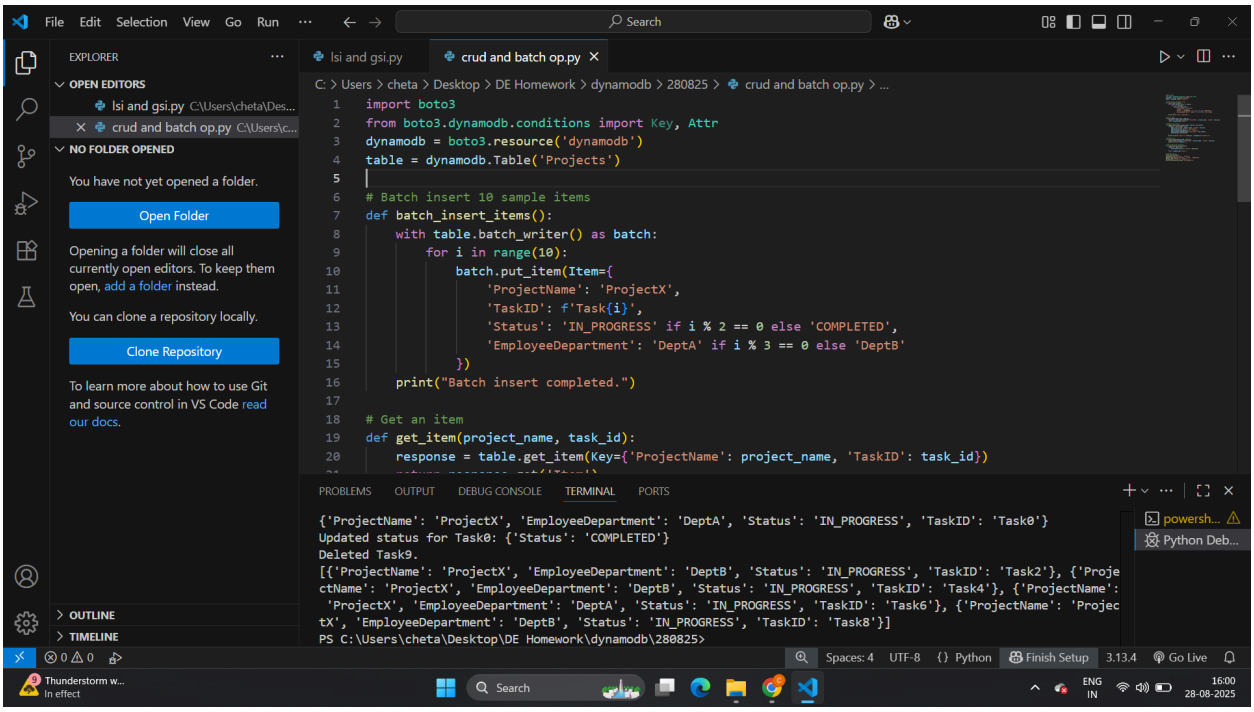
CODE:

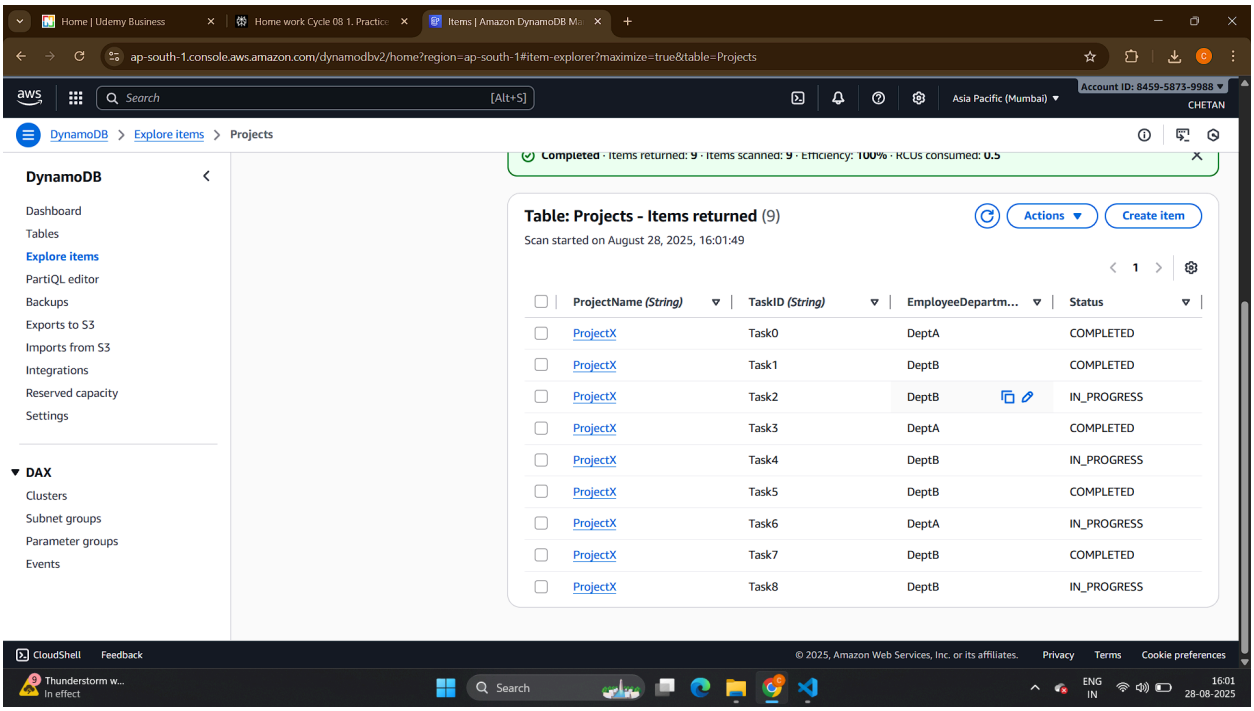
```
import boto3
from boto3.dynamodb.conditions import Key, Attr
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Projects')

# Batch insert 10 sample items
def batch_insert_items():
    with table.batch_writer() as batch:
        for i in range(10):
```

```
            batch.put_item(Item={
                'ProjectName': 'ProjectX',
                'TaskID': f'Task{i}',
                'Status': 'IN_PROGRESS' if i % 2 == 0 else 'COMPLETED',
                'EmployeeDepartment': 'DeptA' if i % 3 == 0 else 'DeptB'
            })
    print("Batch insert completed.")

# Get an item
def get_item(project_name, task_id):
    response = table.get_item(Key={'ProjectName': project_name, 'TaskID': tas
k_id})
    return response.get('Item')

# Update item status
def update_item_status(project_name, task_id, new_status):
    response = table.update_item(
        Key={'ProjectName': project_name, 'TaskID': task_id},
        UpdateExpression='SET #s = :new_status',
        ExpressionAttributeNames={'#s': 'Status'},
        ExpressionAttributeValues={':new_status': new_status},
        ReturnValues='UPDATED_NEW'
    )
    print(f"Updated status for {task_id}: {response['Attributes']}")

# Delete item
def delete_item(project_name, task_id):
    table.delete_item(Key={'ProjectName': project_name, 'TaskID': task_id})
    print(f"Deleted {task_id}.")

# Scan with filter function
def find_tasks_by_status(status):
    response = table.scan(
        FilterExpression=Attr('Status').eq(status)
    )
    return response.get('Items')
```

```
# Example calls:
batch_insert_items()
print(get_item('ProjectX', 'Task0'))
update_item_status('ProjectX', 'Task0', 'COMPLETED')
delete_item('ProjectX', 'Task9')
print(find_tasks_by_status('IN_PROGRESS'))
```

SCREENSHOTS:





5. Index Performance & Cost Analysis

Steps:

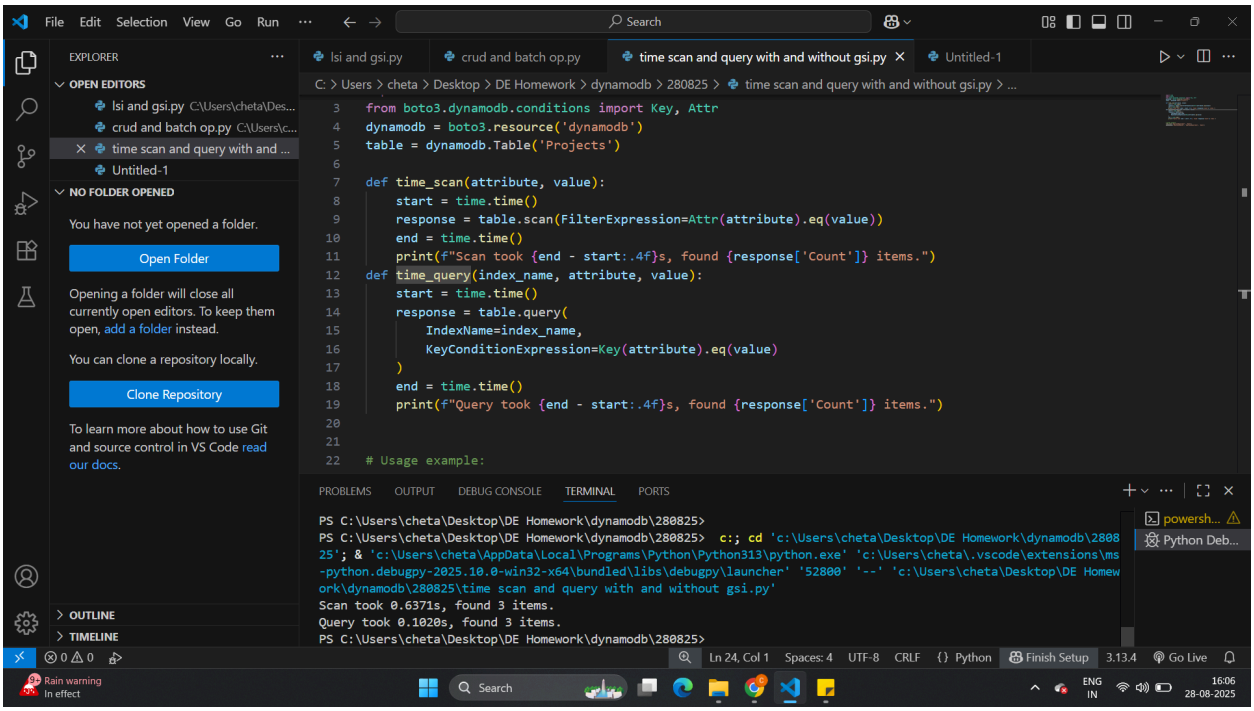
1. **Baseline Table (No GSI):**
    - Load 1000+ items into a table.
    - Run a Scan to search for `EmployeeDepartment`.
    - Note execution time and RCUs consumed.
2. **Add GSI:**
    - Create a GSI with `EmployeeDepartment` as the Partition Key.
    - Run the same query using `Query` on the GSI.
    - Note execution time and RCUs consumed.
3. **Compare Results:**
    - Scan will be slower and consume more RCUs.
    - Query on GSI will be faster and cheaper.

4. **Analysis Report:**
    - Document how GSIs improve performance.
    - Explain how, at large scale, the savings in read capacity outweigh the storage cost of maintaining the index.

CODE:

```
import time
import boto3
from boto3.dynamodb.conditions import Key, Attr
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Projects')

def time_scan(attribute, value):
    start = time.time()
    response = table.scan(FilterExpression=Attr(attribute).eq(value))
    end = time.time()
    print(f"Scan took {end - start:.4f}s, found {response['Count']} items.")
def time_query(index_name, attribute, value):
    start = time.time()
    response = table.query(
        IndexName=index_name,
        KeyConditionExpression=Key(attribute).eq(value)
    )
    end = time.time()
    print(f"Query took {end - start:.4f}s, found {response['Count']} items.")


# Usage example:
time_scan('EmployeeDepartment', 'DeptA')
time_query('EmployeeDeptGSI', 'EmployeeDepartment', 'DeptA')
```

SCREENSHOTS:



6. Auto-Scaling and Throttling Simulation

Steps:

1. **Provisioned Mode with 1 WCU:**
    - Create a table with minimal capacity (1 WCU).
    - Try to write 100 items quickly.
    - Expect throttling errors (`ProvisionedThroughputExceededException`).
2. **Enable Auto-Scaling:**
    - Configure scaling with target utilization = 70%.
    - Minimum = 1 WCU, Maximum = 10 WCU.
3. **Run the test again:**
    - Initially, throttling may occur.
    - After scaling adjusts, writes succeed without errors.
4. **Document findings:**

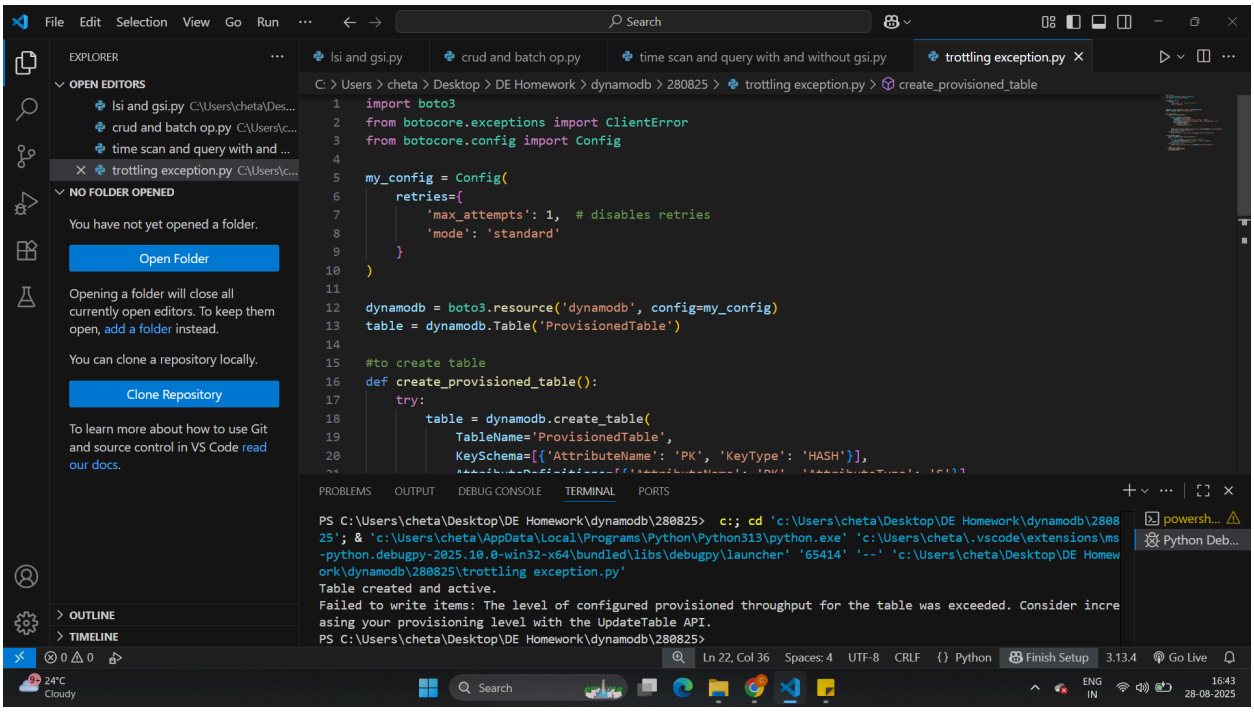
- Scenario A: Without auto-scaling → throttling and failures.
- Scenario B: With auto-scaling → table adapts capacity automatically.

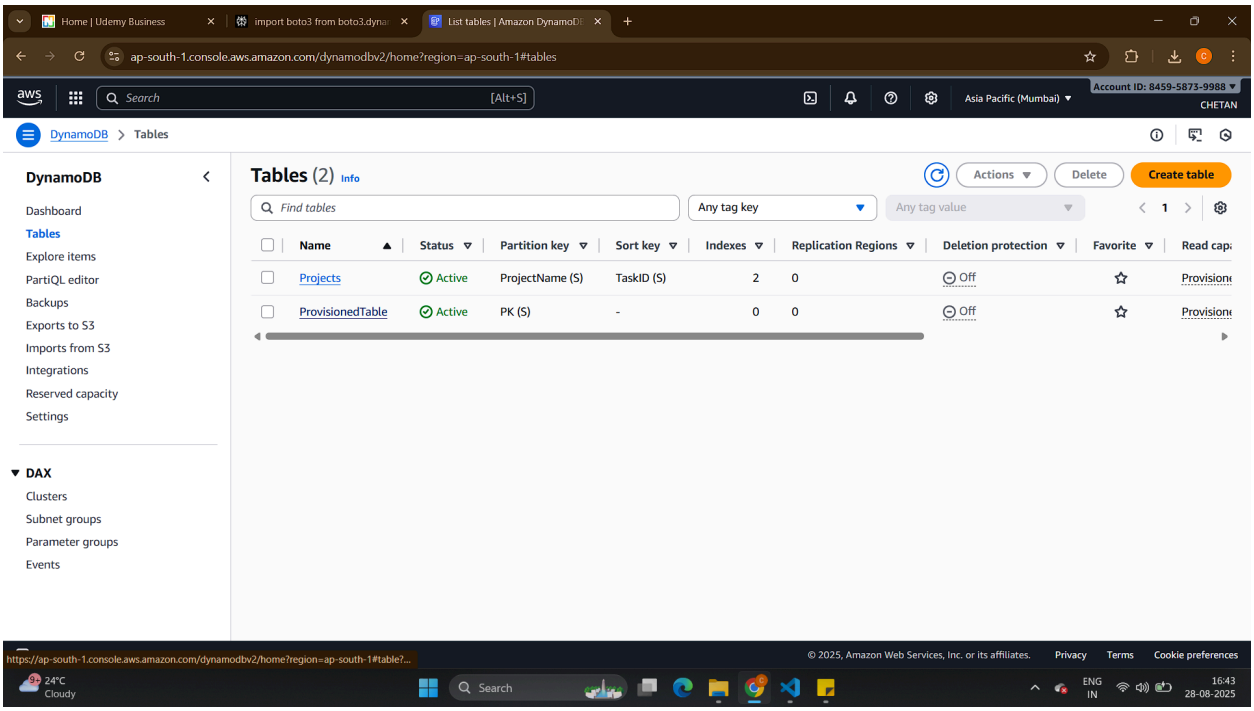
TROTTLING ERROR CODE:

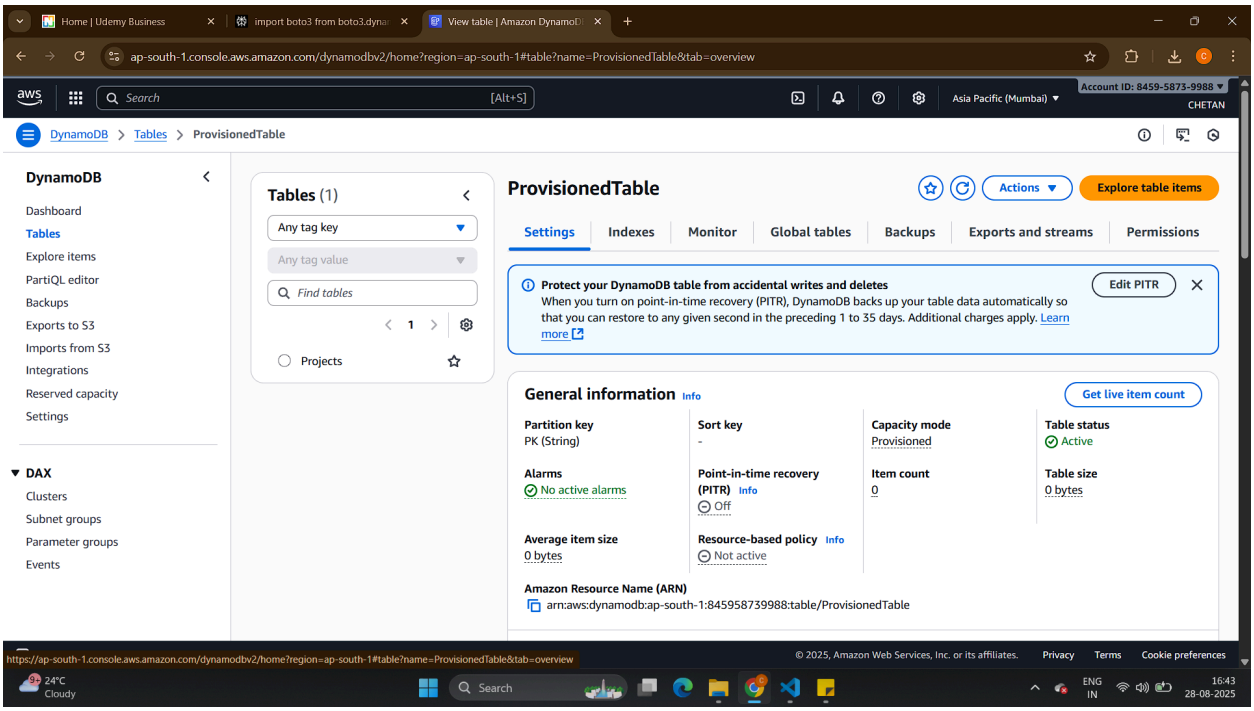
```
import boto3
from botocore.exceptions import ClientError
from botocore.config import Config

my_config = Config(
    retries={
        'max_attempts': 1,  # disables retries
        'mode': 'standard'
    }
)

dynamodb = boto3.resource('dynamodb', config=my_config)
table = dynamodb.Table('ProvisionedTable')

#to create table
def create_provisioned_table():
    try:
        table = dynamodb.create_table(
            TableName='ProvisionedTable',
            KeySchema=[{'AttributeName': 'PK', 'KeyType': 'HASH'}],
            AttributeDefinitions=[{'AttributeName': 'PK', 'AttributeType': 'S'}],
            ProvisionedThroughput={
                'ReadCapacityUnits': 1,
                'WriteCapacityUnits': 1
            }
        )
        table.meta.client.get_waiter('table_exists').wait(TableName='Provisioned
Table')
        print("Table created and active.")
    except ClientError as e:
```

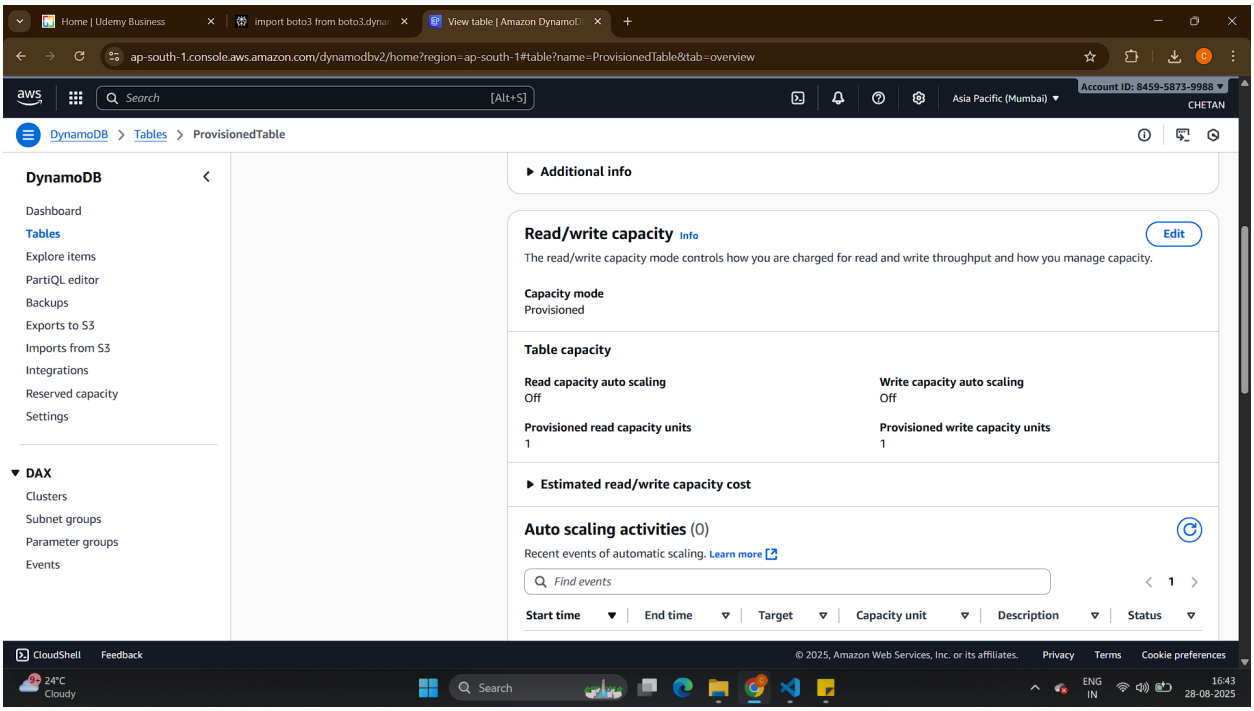
```
        print(f"Failed to create table: {e.response['Error']['Message']}")

def put_sample_items():
    table = dynamodb.Table('ProvisionedTable')
    try:
        for i in range(100):  # fewer writes for simplicity
            table.put_item(Item={'PK': f'Item{i}', 'Data': 'sample'})
        print("Items inserted successfully.")
    except ClientError as e:
        print(f"Failed to write items: {e.response['Error']['Message']}")

if __name__ == '__main__':
    create_provisioned_table()
    put_sample_items()
```

SCREENSHOTS FOR TROTTLING ERROR:




Home | Udemy Business
import boto3 from boto3.dyna
List tables | Amazon DynamoDB
ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#tables
Search [Alt+S]
Asia Pacific (Mumbai)
Account ID: 8459-5873-9988
CHETAN
DynamoDB > Tables
DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
Tables (2) Info
Actions
Delete
Create table
Find tables
Any tag key
Any tag value
Name
Status
Partition key
Sort key
Indexes
Replication Regions
Deletion protection
Favorite
Read capa
Projects
Active
ProjectName (S)
TaskID (S)
2
0
Off
ProvisionedTable
Active
PK (S)
-
0
0
Off
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
24°C
Cloudy
Search
ENG
IN
16:43
28-08-2025


Home | Udemy Business
import boto3 from boto3.dyna
View table | Amazon DynamoDB
ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#table?name=ProvisionedTable&tab=overview
Search [Alt+S]
Asia Pacific (Mumbai)
Account ID: 8459-5873-9988
CHETAN
DynamoDB > Tables > ProvisionedTable
DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
Tables (1)
Any tag key
Any tag value
Find tables
1
Projects
ProvisionedTable
Actions
Explore table items
Settings
Indexes
Monitor
Global tables
Backups
Exports and streams
Permissions
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 1 to 35 days. Additional charges apply. Learn more
Edit PITR
General information Info
Get live item count
Partition key
PK (String)
Sort key
-
Capacity mode
Provisioned
Table status
Active
Alarms
No active alarms
Point-in-time recovery (PITR) Info
Off
Item count
0
Table size
0 bytes
Average item size
0 bytes
Resource-based policy Info
Not active
Amazon Resource Name (ARN)
arn:aws:dynamodb:ap-south-1:845958739988:table/ProvisionedTable
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
24°C
Cloudy
Search
ENG
IN
16:43
28-08-2025



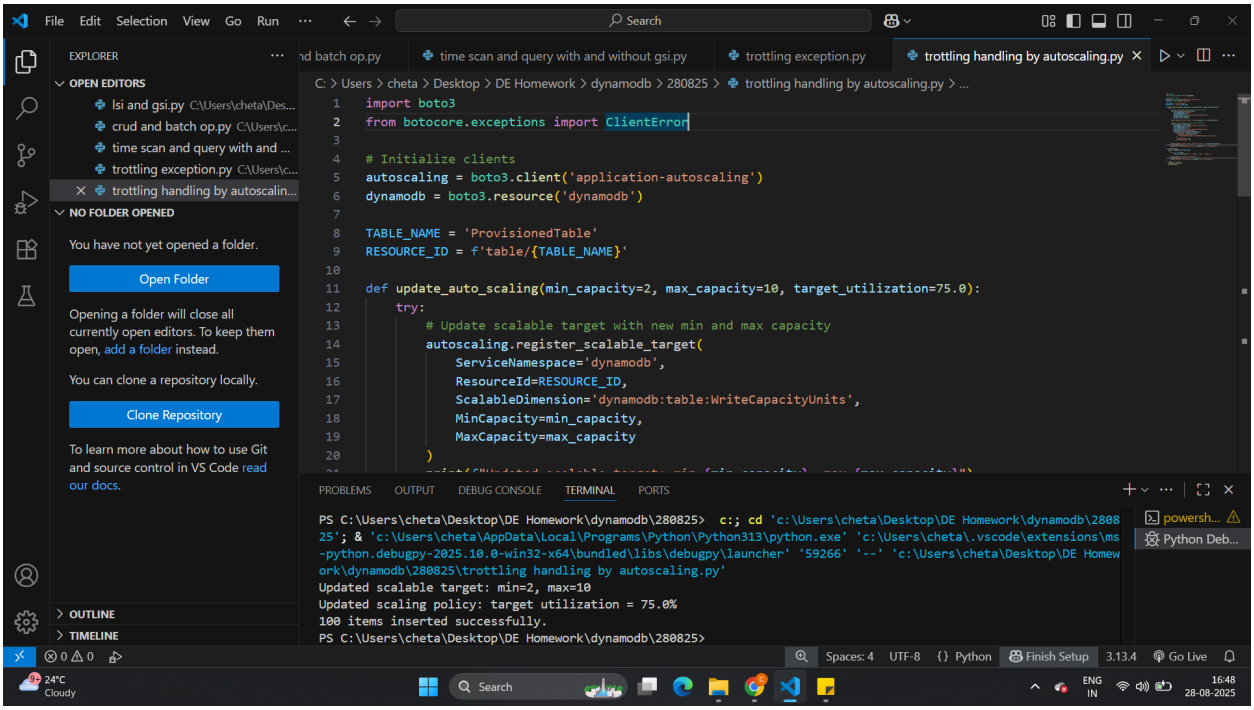
CODE FOR EDITING AUTOSCALING :

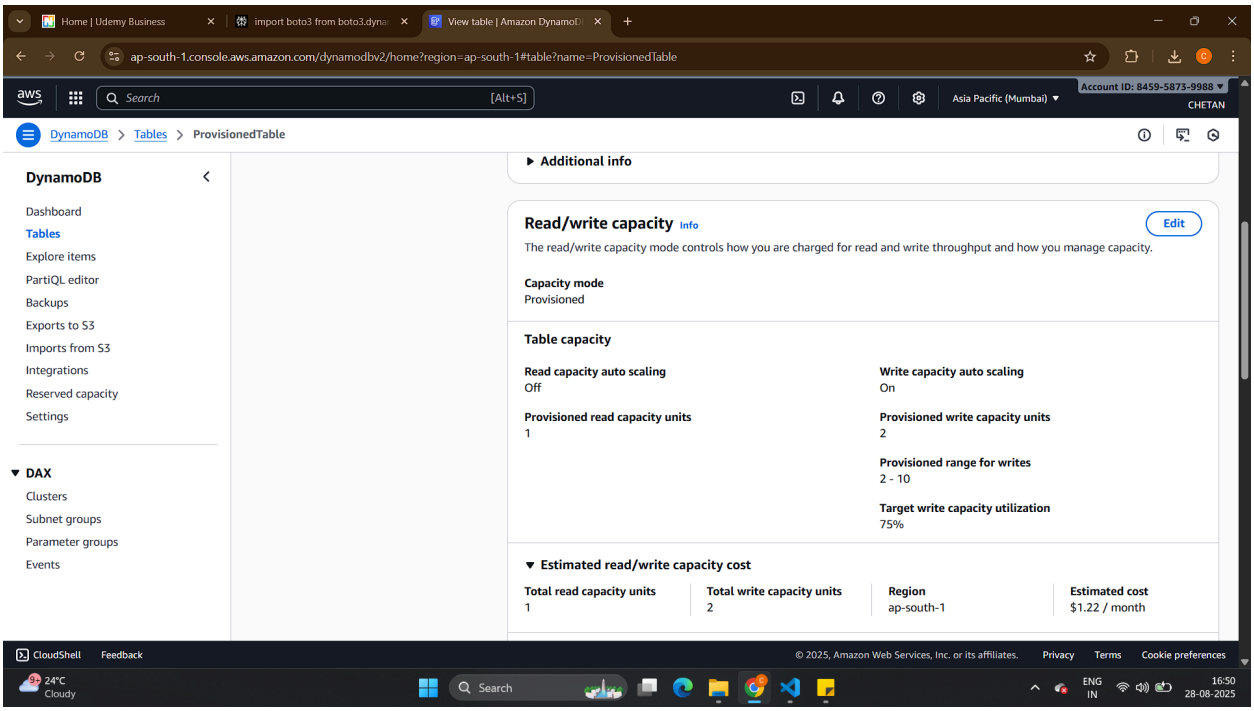
```
import boto3
from botocore.exceptions import ClientError

# Initialize clients
autoscaling = boto3.client('application-autoscaling')
dynamodb = boto3.resource('dynamodb')

TABLE_NAME = 'ProvisionedTable'
RESOURCE_ID = f'table/{TABLE_NAME}'

def update_auto_scaling(min_capacity=2, max_capacity=10, target_utilization=
75.0):
    try:
        # Update scalable target with new min and max capacity
        autoscaling.register_scalable_target(
            ServiceNamespace='dynamodb',
            ResourceId=RESOURCE_ID,
            ScalableDimension='dynamodb:table:WriteCapacityUnits',
```

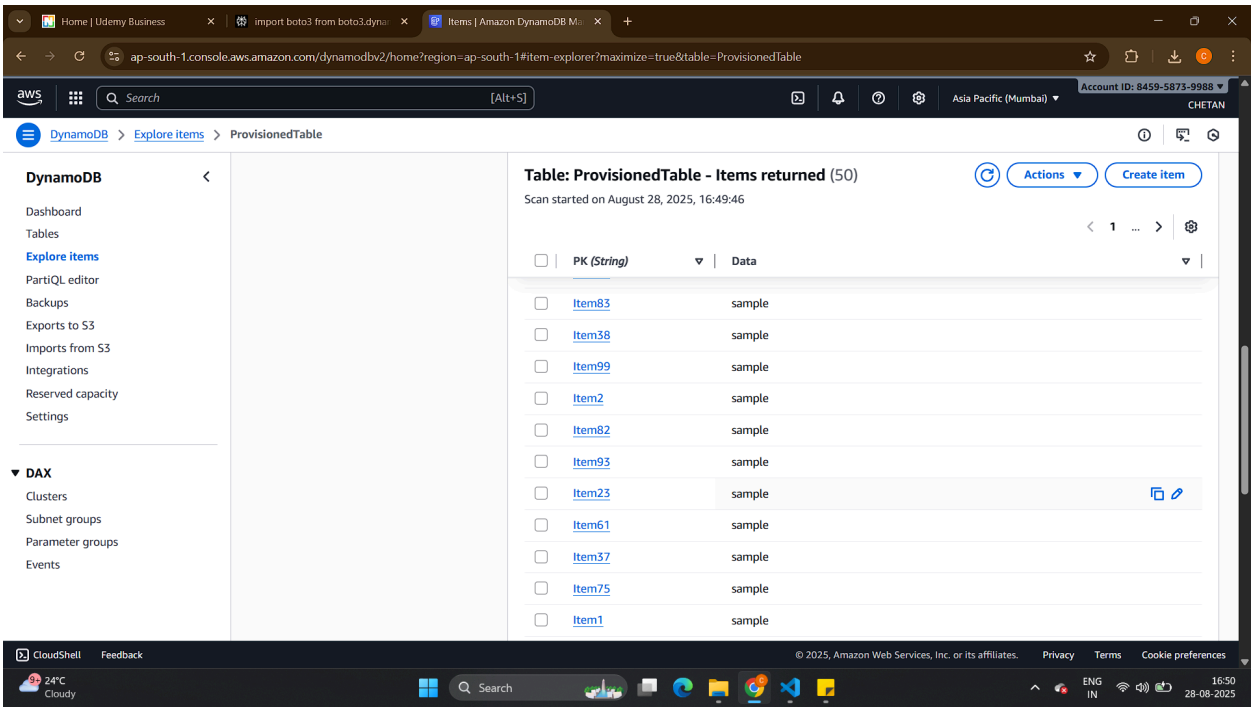
```
            MinCapacity=min_capacity,
            MaxCapacity=max_capacity
        )
        print(f"Updated scalable target: min={min_capacity}, max={max_capacit
y}")

        # Update scaling policy with new target utilization
        autoscaling.put_scaling_policy(
            PolicyName=f'{TABLE_NAME}-write-auto-scaling-policy',
            ServiceNamespace='dynamodb',
            ResourceId=RESOURCE_ID,
            ScalableDimension='dynamodb:table:WriteCapacityUnits',
            PolicyType='TargetTrackingScaling',
            TargetTrackingScalingPolicyConfiguration={
                'TargetValue': target_utilization,
                'PredefinedMetricSpecification': {
                    'PredefinedMetricType': 'DynamoDBWriteCapacityUtilization'
                },
                'ScaleInCooldown': 60,
                'ScaleOutCooldown': 60
            }
        )
        print(f"Updated scaling policy: target utilization = {target_utilization}%")
    except ClientError as e:
        print(f"Error updating auto scaling: {e.response['Error']['Message']}")

def put_100_items():
    table = dynamodb.Table(TABLE_NAME)
    try:
        for i in range(100):
            table.put_item(Item={'PK': f'Item{i}', 'Data': 'sample'})

        print("100 items inserted successfully.")
    except ClientError as e:
        print(f"Failed to write items: {e.response['Error']['Message']}")
```

```
if __name__ == '__main__':
    update_auto_scaling()
    put_100_items()
```

SCREENSHOTS FOR AUTOSCALING AND ADDING 100 ELEMENTS:



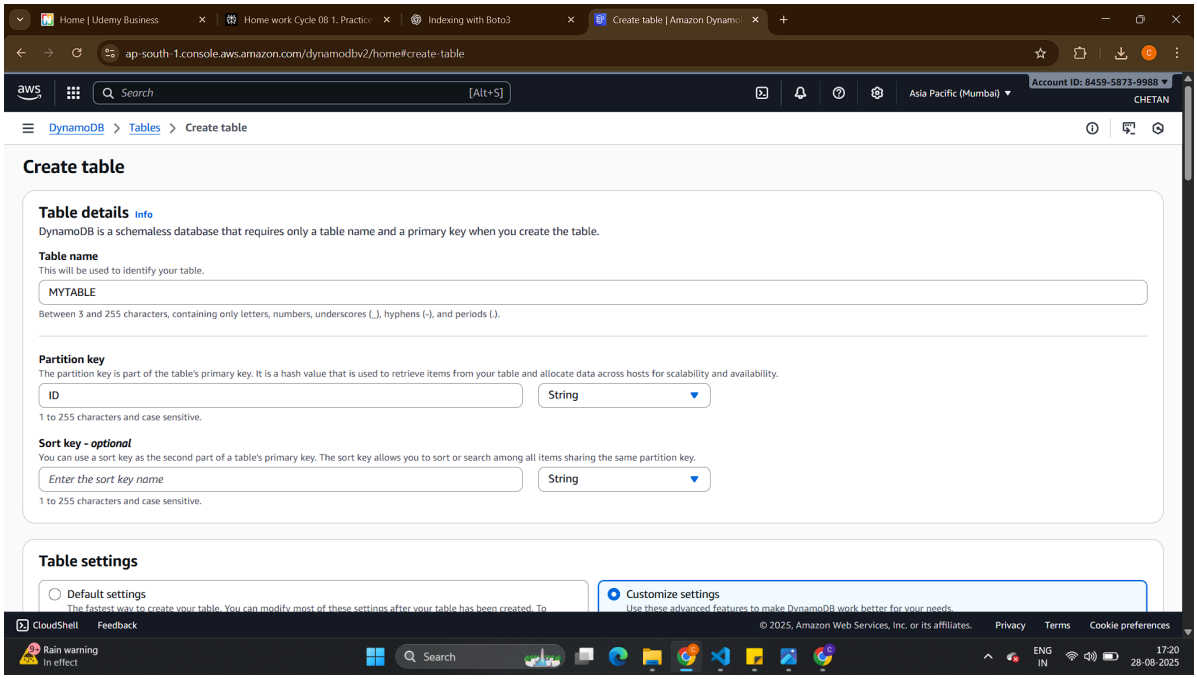


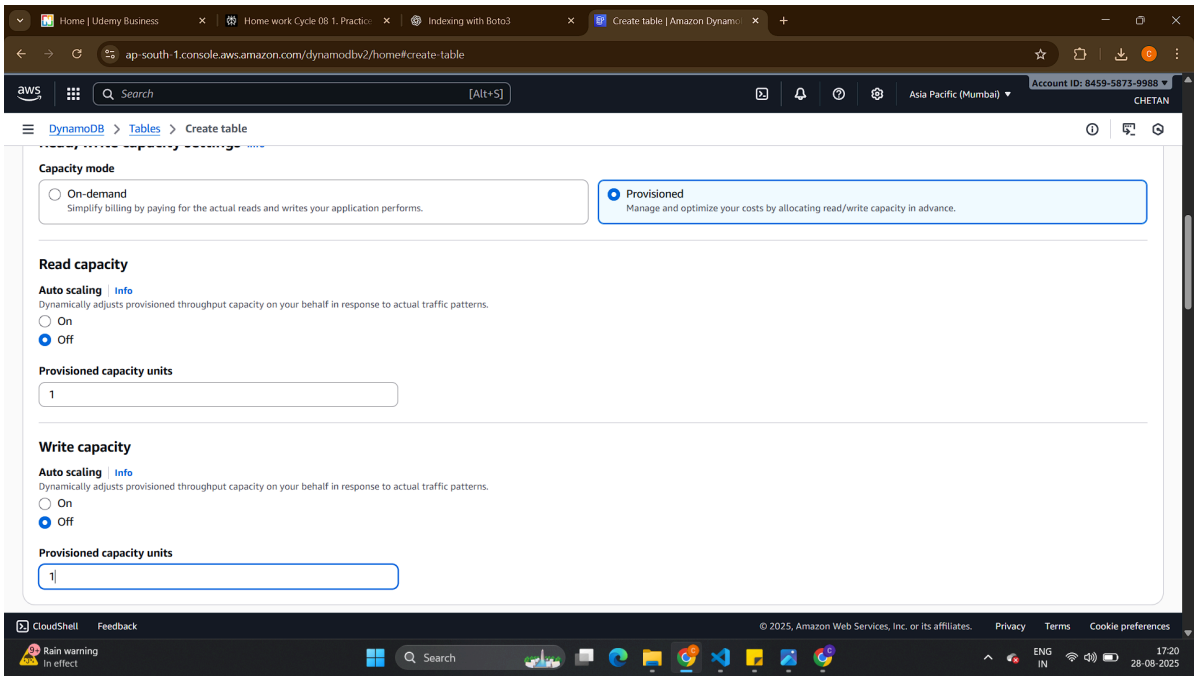


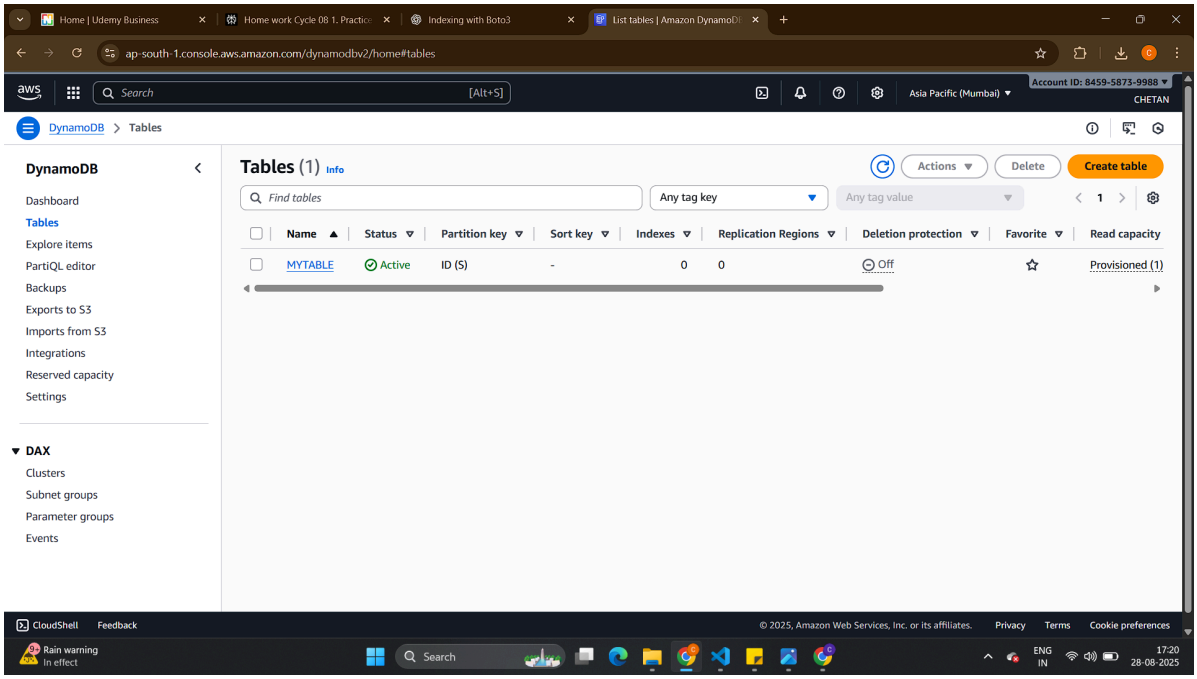
7. Implementing Global Tables and Backups

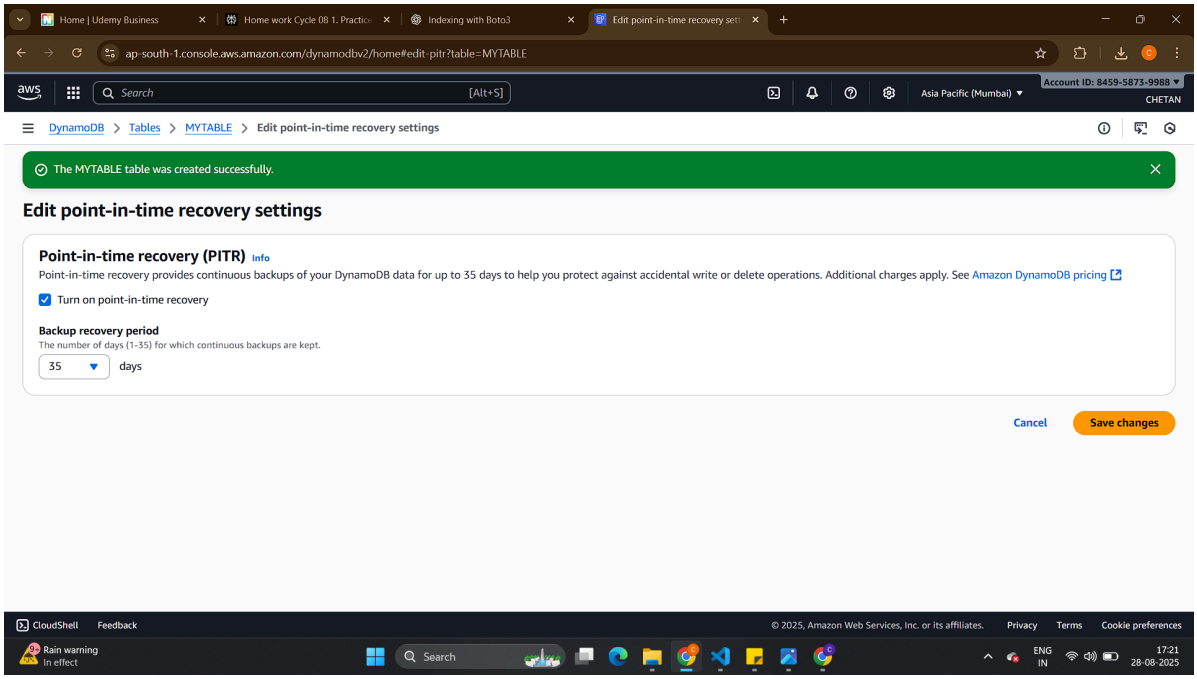
Steps:

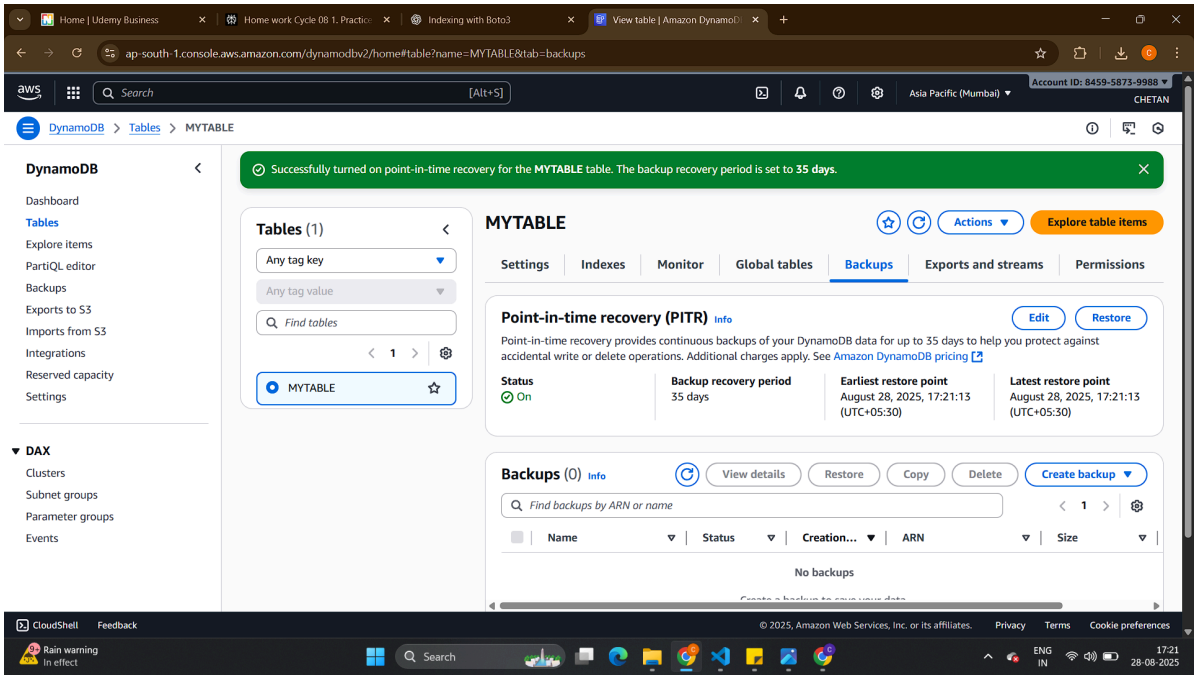
1. **Enable PITR (Point-in-Time Recovery):**
    - Allows restoring table to any second within the last 35 days.
    - Provides continuous backup.





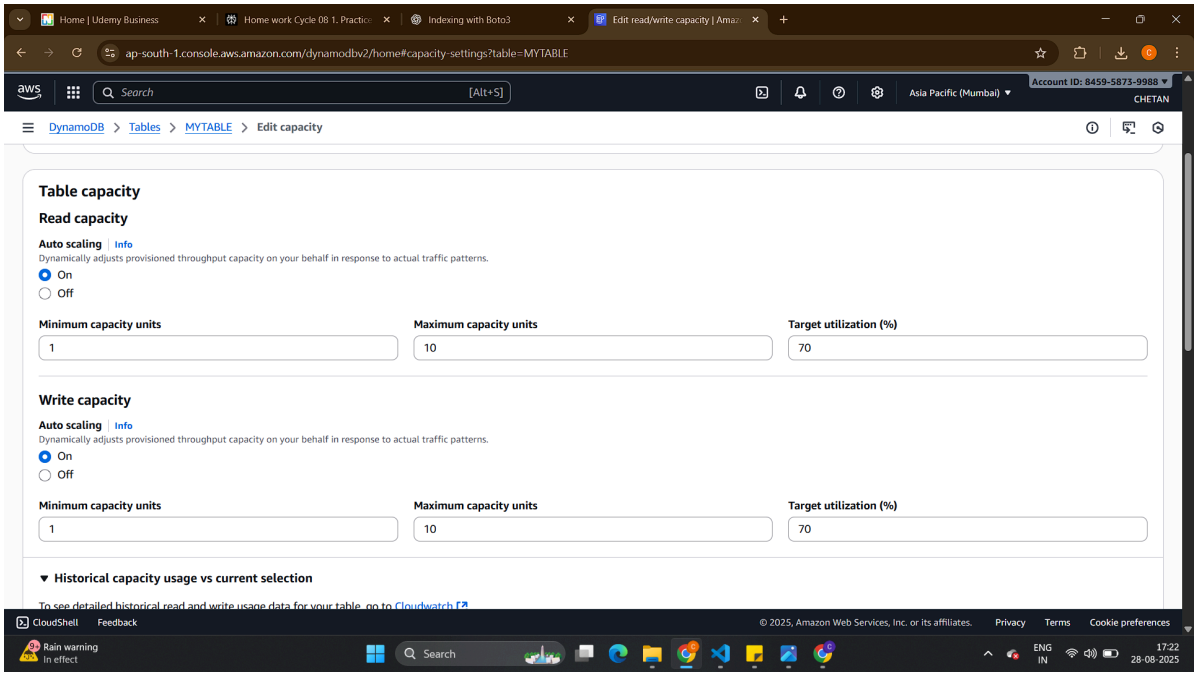

Home | Udemy Business
Home work Cycle 08 1. Practice
Indexing with Boto3
List tables | Amazon DynamoDB
ap-south-1.console.aws.amazon.com/dynamodbv2/home#tables
Search
[Alt+S]
Asia Pacific (Mumbai)
Account ID: 8459-5873-9988
CHETAN
DynamoDB > Tables
DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
Tables (1) Info
Actions
Delete
Create table
Find tables
Any tag key
Any tag value
Name
Status
Partition key
Sort key
Indexes
Replication Regions
Deletion protection
Favorite
Read capacity
MYTABLE
Active
ID (S)
-
0
0
Off
Provisioned (1)
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Rain warning
In effect
Search
ENG
IN
17:20
28-08-2025

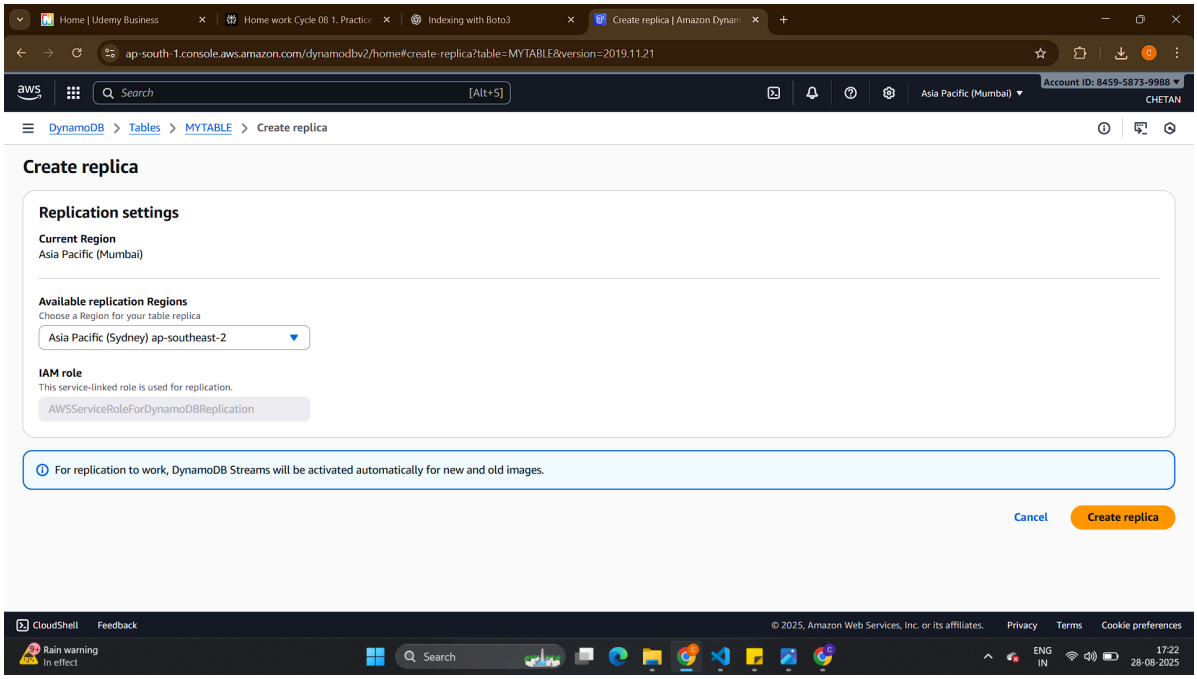

Home | Udemy Business
Home work Cycle 08 1. Practice
Indexing with Boto3
Edit point-in-time recovery sett
ap-south-1.console.aws.amazon.com/dynamodbv2/home#edit-pitr?table=MYTABLE
Search
[Alt+S]
Asia Pacific (Mumbai)
Account ID: 8459-5873-9988
CHETAN
DynamoDB > Tables > MYTABLE > Edit point-in-time recovery settings
The MYTABLE table was created successfully.
Edit point-in-time recovery settings
Point-in-time recovery (PITR) Info
Point-in-time recovery provides continuous backups of your DynamoDB data for up to 35 days to help you protect against accidental write or delete operations. Additional charges apply. See Amazon DynamoDB pricing
Turn on point-in-time recovery
Backup recovery period
The number of days (1-35) for which continuous backups are kept.
35
days
Cancel
Save changes
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Rain warning
In effect
Search
ENG
IN
17:21
28-08-2025

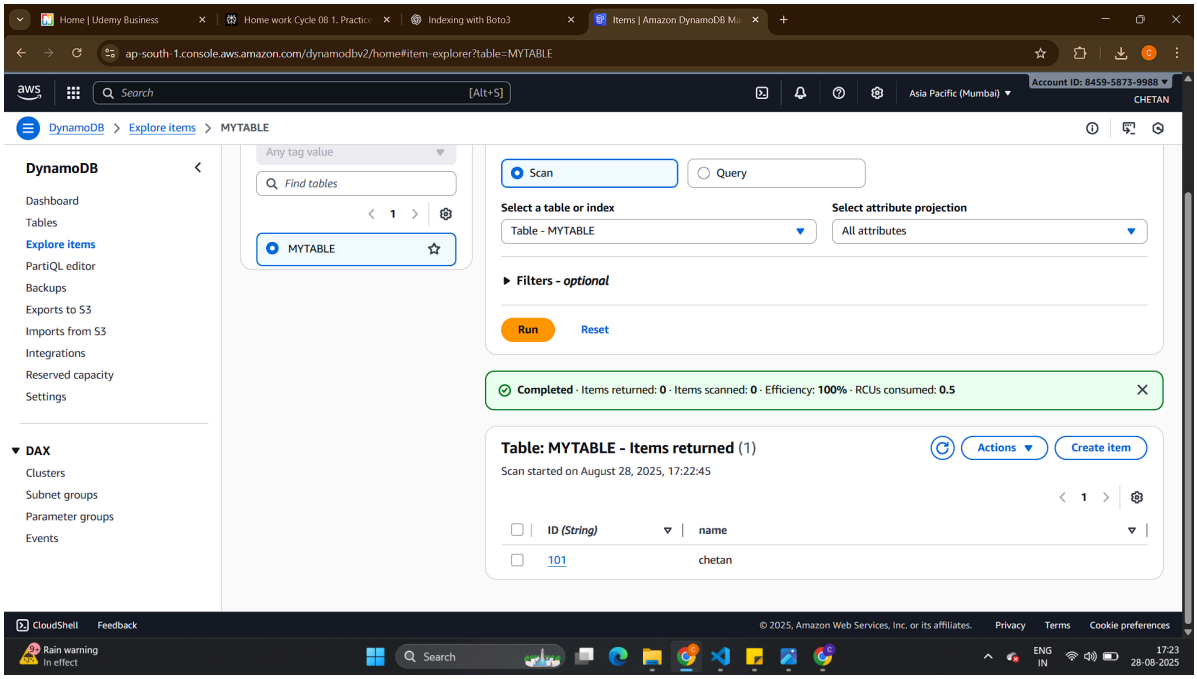


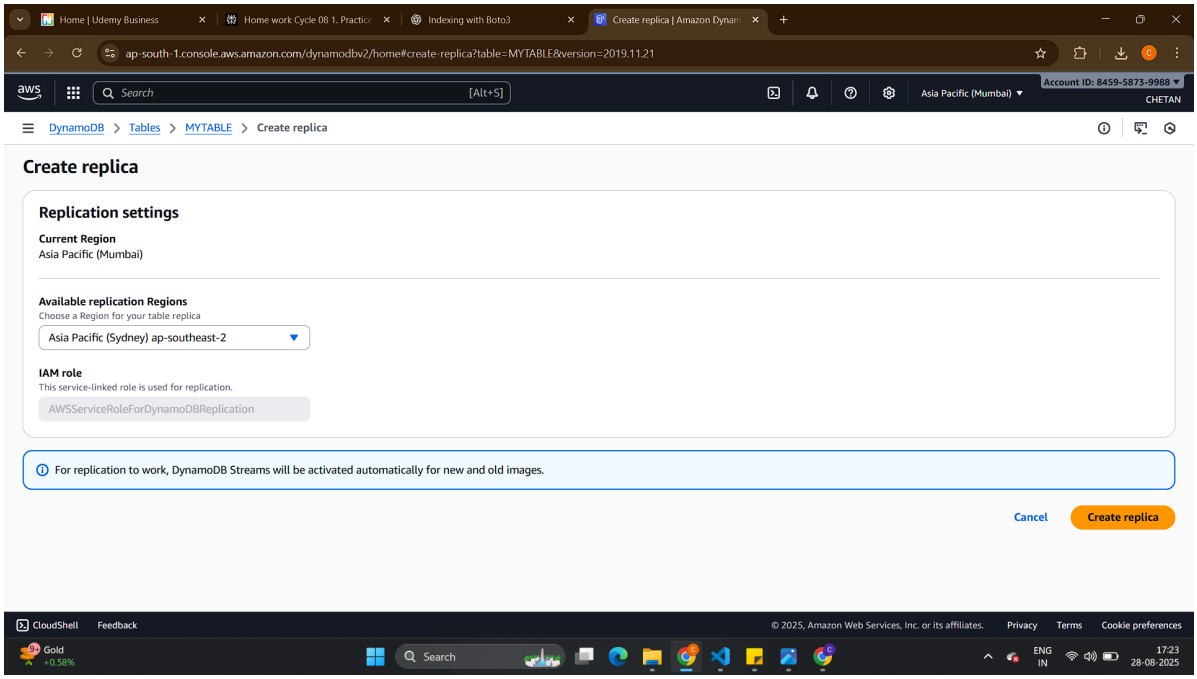
2. **Create a Global Table:**

   - Add a replica table in another AWS region.
   - Example: Primary in `us-east-1`, replica in `us-west-2`.



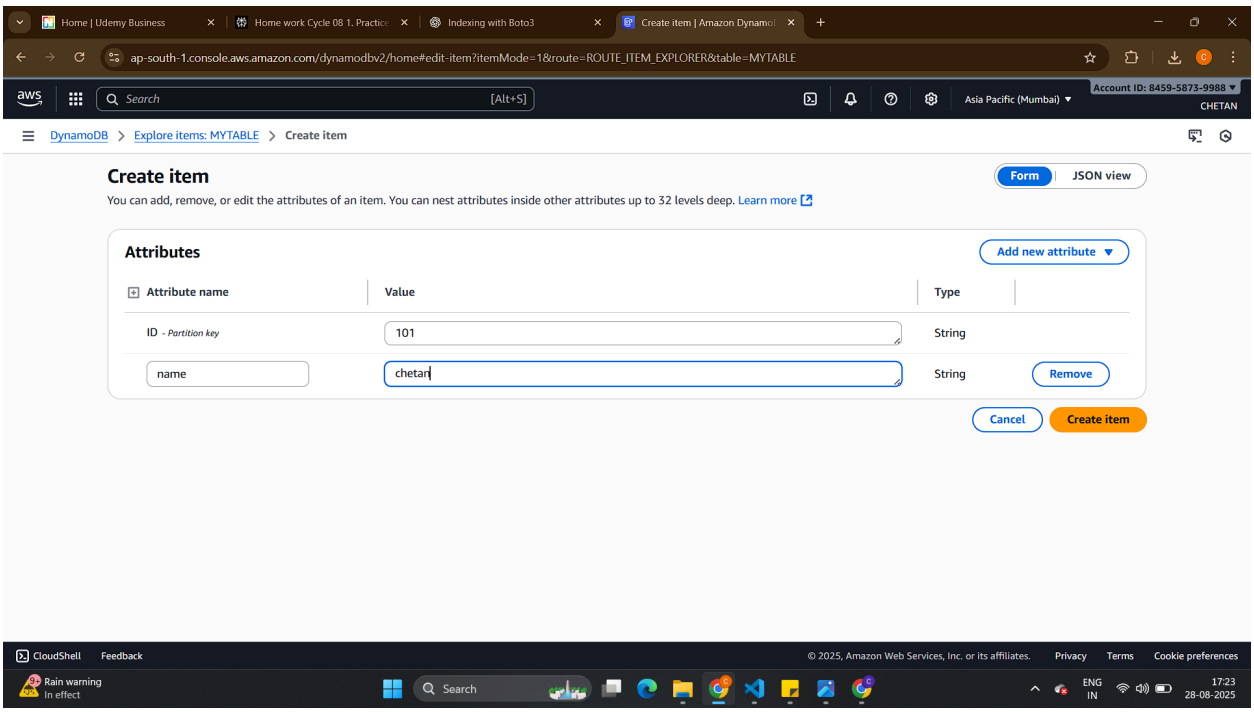

Home | Udemy Business
Home work Cycle 08 1. Practice
Indexing with Boto3
Create replica | Amazon Dynam
ap-south-1.console.aws.amazon.com/dynamodbv2/home#create-replica?table=MYTABLE&version=2019.11.21
Search [Alt+S]
Asia Pacific (Mumbai)
Account ID: 8459-5873-9988
CHETAN
DynamoDB > Tables > MYTABLE > Create replica
Create replica
Replication settings
Current Region
Asia Pacific (Mumbai)
Available replication Regions
Choose a Region for your table replica
Asia Pacific (Sydney) ap-southeast-2
IAM role
This service-linked role is used for replication.
AWSServiceRoleForDynamoDBReplication
For replication to work, DynamoDB Streams will be activated automatically for new and old images.
Cancel
Create replica
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Rain warning
In effect
Search
ENG
IN
17:22
28-08-2025

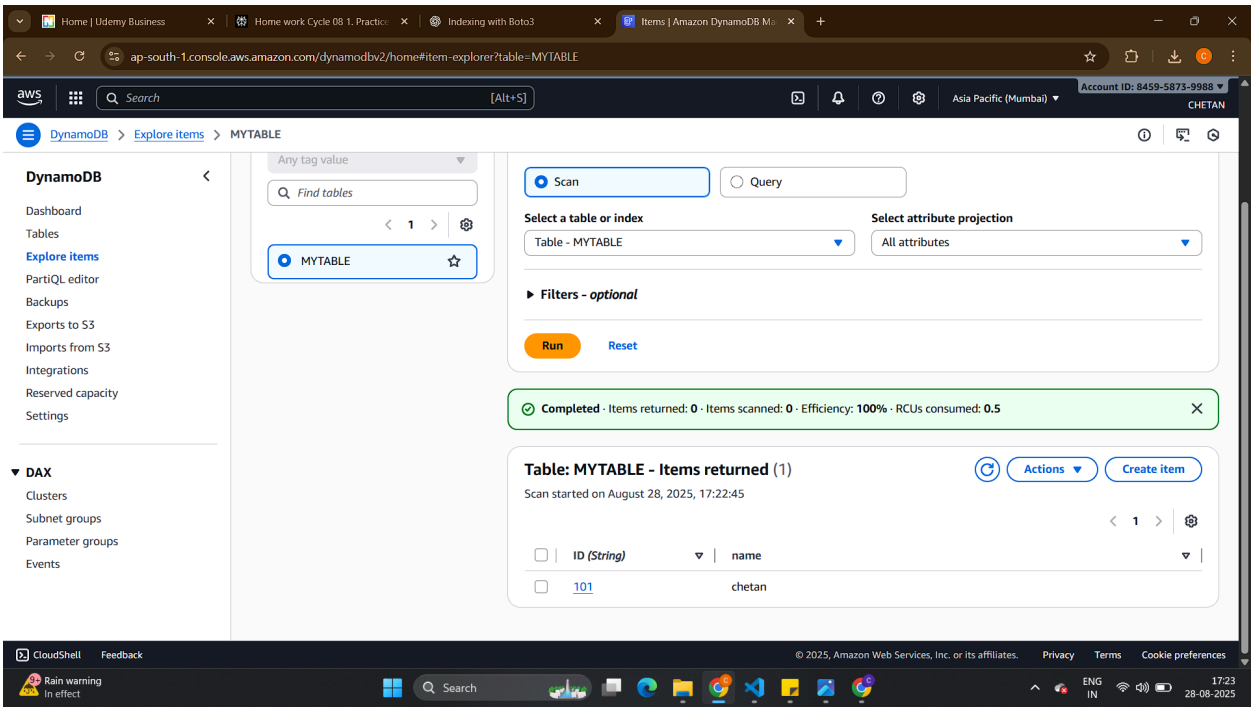

Home | Udemy Business
Home work Cycle 08 1. Practice
Indexing with Boto3
Items | Amazon DynamoDB Ma
ap-south-1.console.aws.amazon.com/dynamodbv2/home#item-explorer?table=MYTABLE
Search [Alt+S]
Asia Pacific (Mumbai)
Account ID: 8459-5873-9988
CHETAN
DynamoDB > Explore items > MYTABLE
DynamoDB
Dashboard
Tables
Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Integrations
Reserved capacity
Settings
DAX
Clusters
Subnet groups
Parameter groups
Events
Any tag value
Find tables
MYTABLE
Scan
Query
Select a table or index
Table - MYTABLE
Select attribute projection
All attributes
Filters - optional
Run
Reset
Completed · Items returned: 0 · Items scanned: 0 · Efficiency: 100% · RCUs consumed: 0.5
Table: MYTABLE - Items returned (1)
Scan started on August 28, 2025, 17:22:45
Actions
Create item
ID (String)
name
101
chetan
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Rain warning
In effect
Search
ENG
IN
17:23
28-08-2025

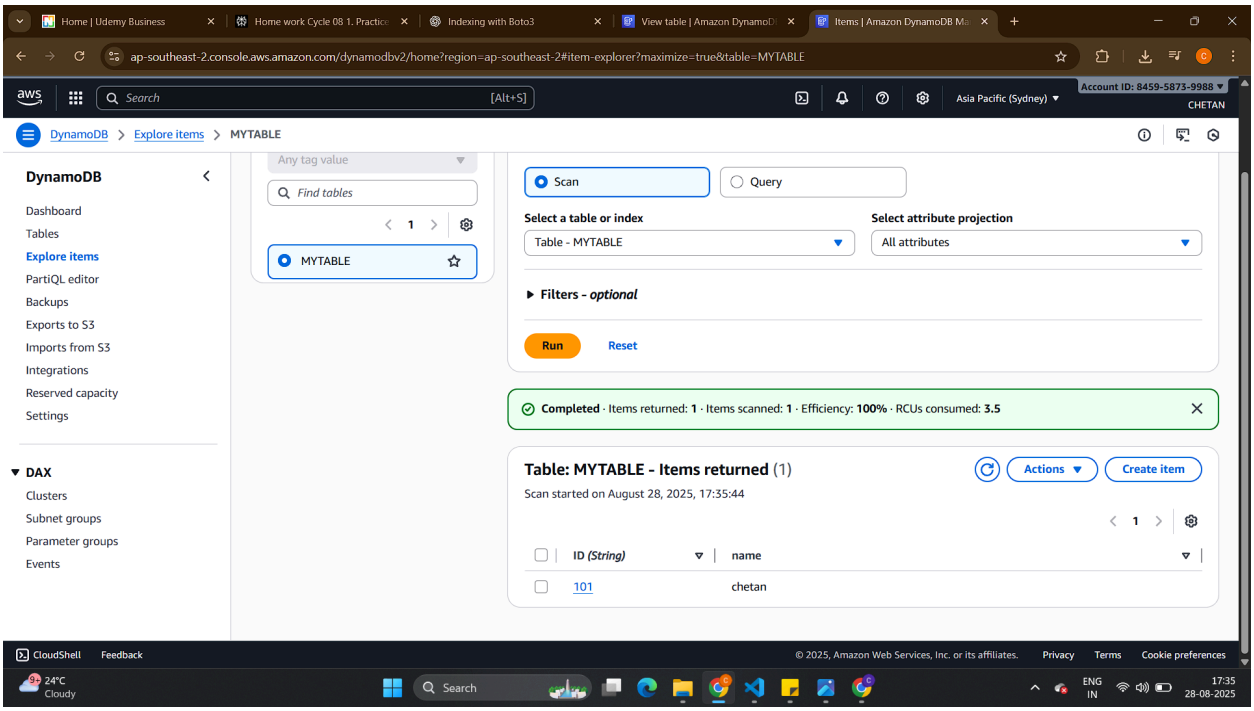


3. **Replication Test:**

   - Insert an item into the main table.
   - Verify it automatically appears in the replica table.

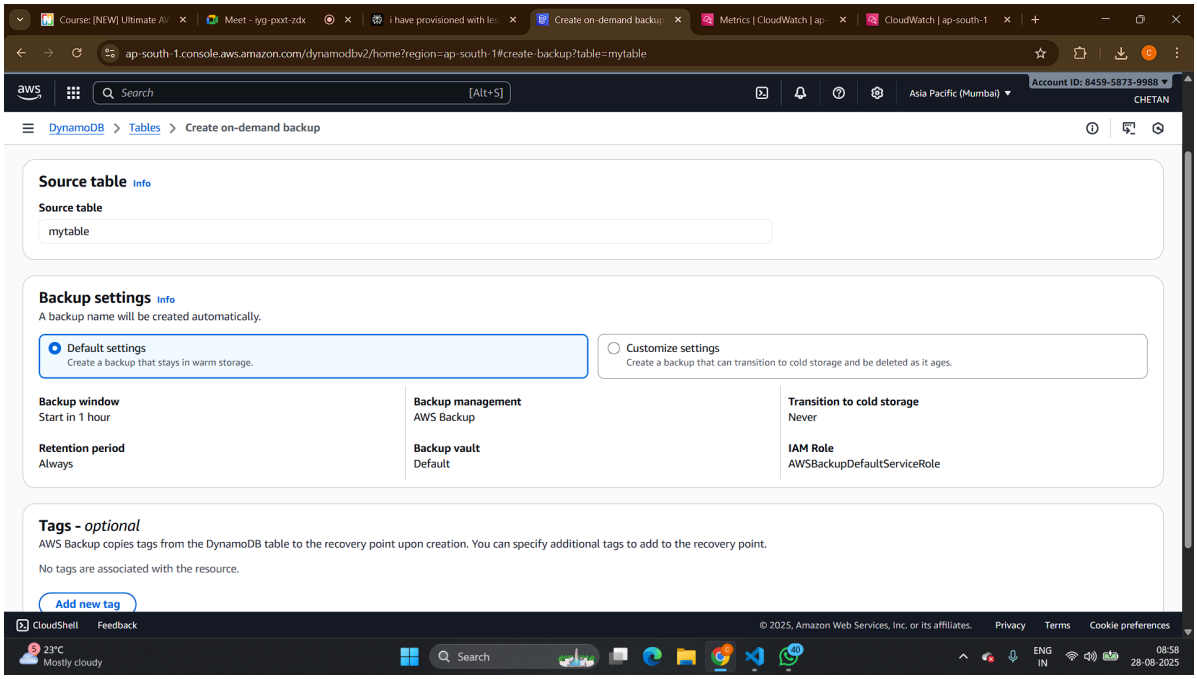


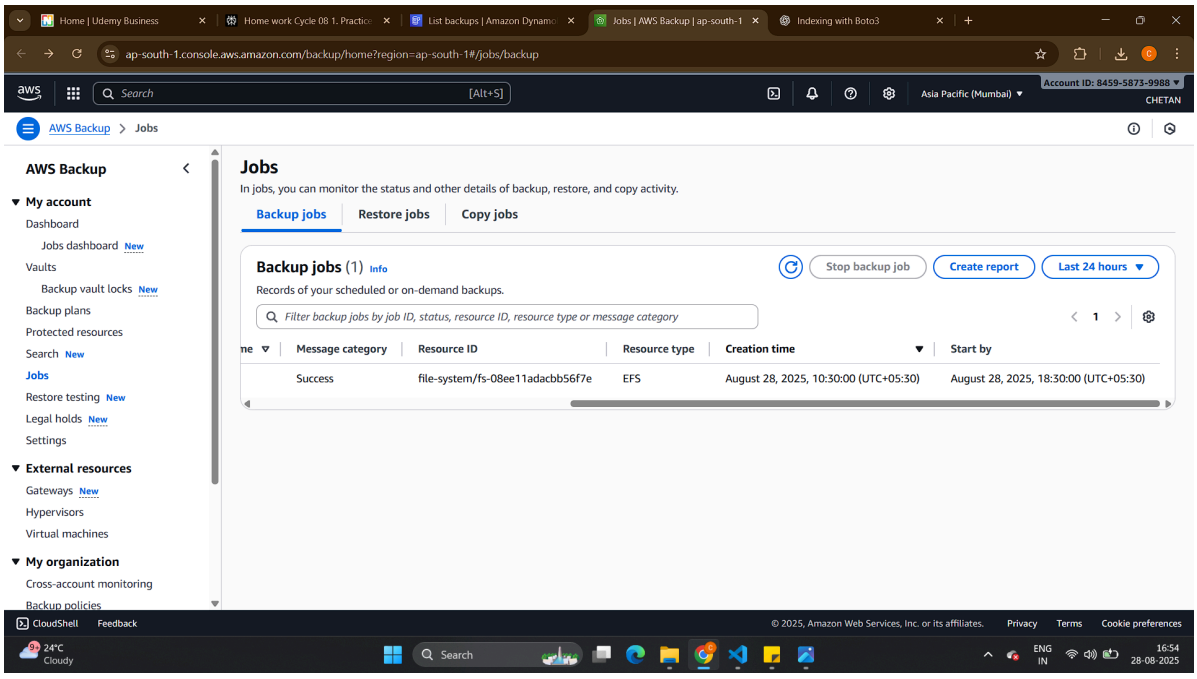




1. **On-Demand Backup:**
    - Create a manual snapshot of the table.
    - Note the backup ARN for documentation.

SCREENSHOTS:





2. **Documentation:**
   - List the steps, regions, ARNs of global tables, and backups created.
   - Mention replication delay observed.

Final Note

Each of these tasks builds practical DynamoDB skills:

- Index design and billing trade-offs.
- CRUD and batch operations.
- Index performance testing.
- Capacity planning and auto-scaling.
- Cross-region replication and backup strategies.