Day 32 - 12th Sept 2025

With ref to Daywise Docs .. named Day 31 - 7th Sept 2025 DynamoDB

Task 01:

Create a table using Java code and check if the table is created.

For example:

In your java code if employees03 table is created..

package org.example;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;

import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;

import software.amazon.awssdk.regions.Region;

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import software.amazon.awssdk.services.dynamodb.model.\*;

import java.net.URI;

//create table in DynamoDB

public class Demo01 {

public static void main(String[] args) throws Exception {

System.*out*.println("hello create table in DynamoDB");

AwsBasicCredentials awsCreds = AwsBasicCredentials.*create*("fakeAccesskey","fakeSecretKey");

DynamoDbClient client = DynamoDbClient.*builder*()

.endpointOverride(URI.*create*("http://localhost:8001"))

.region(Region.*AP\_SOUTH\_1*)

.credentialsProvider(StaticCredentialsProvider.*create*(awsCreds))

.build();

String tableName = "Employees01";

try {

CreateTableRequest request = CreateTableRequest.*builder*()

.tableName(tableName)

.keySchema(KeySchemaElement.*builder*()

.attributeName("ID")

.keyType(KeyType.*HASH*)

.build())

.attributeDefinitions(AttributeDefinition.*builder*()

.attributeName("ID")

.attributeType(ScalarAttributeType.*N*)

.build())

.provisionedThroughput(ProvisionedThroughput.*builder*()

.readCapacityUnits(5L)

.writeCapacityUnits(5L)

.build())

.build();

client.createTable(request);

System.*out*.println(tableName + " table is created. ");

}catch (ResourceInUseException ex) {

System.*out*.println(" plz choose different tablename as it already exists");

}

client.close();

}

}

package org.example;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;

import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;

import software.amazon.awssdk.regions.Region;

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import software.amazon.awssdk.services.dynamodb.model.AttributeValue;

import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.fasterxml.jackson.databind.JsonNode;

import java.io.InputStream;

import java.net.URI;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

// loading data to the table created

public class Demo02 {

public static void main(String[] args) throws Exception {

System.*out*.println("loading data to table...");

// Fake credentials for local DynamoDB

AwsBasicCredentials awsCreds = AwsBasicCredentials.*create*("fakeAccesskey", "fakeSecretKey");

// Connect to DynamoDB Local running on port 8001

DynamoDbClient client = DynamoDbClient.*builder*()

.endpointOverride(URI.*create*("http://localhost:8001"))

.region(Region.*AP\_SOUTH\_1*)

.credentialsProvider(StaticCredentialsProvider.*create*(awsCreds))

.build();

String tableName = "Employees01";

// Load JSON from resources

ObjectMapper mapper = new ObjectMapper();

InputStream jsonStream = Demo02.class.getClassLoader().getResourceAsStream("Employee.json");

if (jsonStream == null) {

System.*err*.println("❌ Could not find Employee.json in resources!");

System.*exit*(1);

}

JsonNode rootNode = mapper.readTree(jsonStream);

Iterator<JsonNode> iterator = rootNode.elements();

// Insert records into DynamoDB

while (iterator.hasNext()) {

JsonNode node = iterator.next();

Map<String, AttributeValue> item = new HashMap<>();

item.put("ID", AttributeValue.*builder*().n(node.get("ID").asText()).build());

item.put("Name", AttributeValue.*builder*().s(node.get("Name").asText()).build());

item.put("Address", AttributeValue.*builder*().s(node.get("Address").asText()).build());

PutItemRequest putRequest = PutItemRequest.*builder*()

.tableName(tableName)

.item(item)

.build();

client.putItem(putRequest);

System.*out*.printf("✔ Inserted: ID=%d | Name=%s | Address=%s%n",

node.get("ID").asInt(),

node.get("Name").asText(),

node.get("Address").asText());

}

client.close();

System.*out*.println("✅ All records inserted successfully.");

}

}

package org.example;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;

import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;

import software.amazon.awssdk.regions.Region;

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import software.amazon.awssdk.services.dynamodb.model.AttributeValue;

import software.amazon.awssdk.services.dynamodb.model.DynamoDbResponse;

import software.amazon.awssdk.services.dynamodb.model.ScanRequest;

import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

import java.net.URI;

import java.util.Map;

public class Demo03 {

public static void main(String[] args) {

System.*out*.println("Scanning data from table");

AwsBasicCredentials awsCreds = AwsBasicCredentials.*create*("fakeAccesskey","fakeSecretKey");

DynamoDbClient client = DynamoDbClient.*builder*()

.endpointOverride(URI.*create*("http://localhost:8001"))

.region(Region.*AP\_SOUTH\_1*)

.credentialsProvider(StaticCredentialsProvider.*create*(awsCreds))

.build();

String tableName = "Employees01";

ScanRequest req = ScanRequest.*builder*().tableName(tableName).build();

ScanResponse resp = client.scan(req);

for(Map<String, AttributeValue> dbitem : resp.items()) {

System.*out*.println(dbitem);

}

client.close();

}

}

[

{

"ID": 1001,

"Name": "Prasunamba",

"Address": "India"

},

{

"ID": 1002,

"Name": "Meher",

"Address": "Australia"

},

{

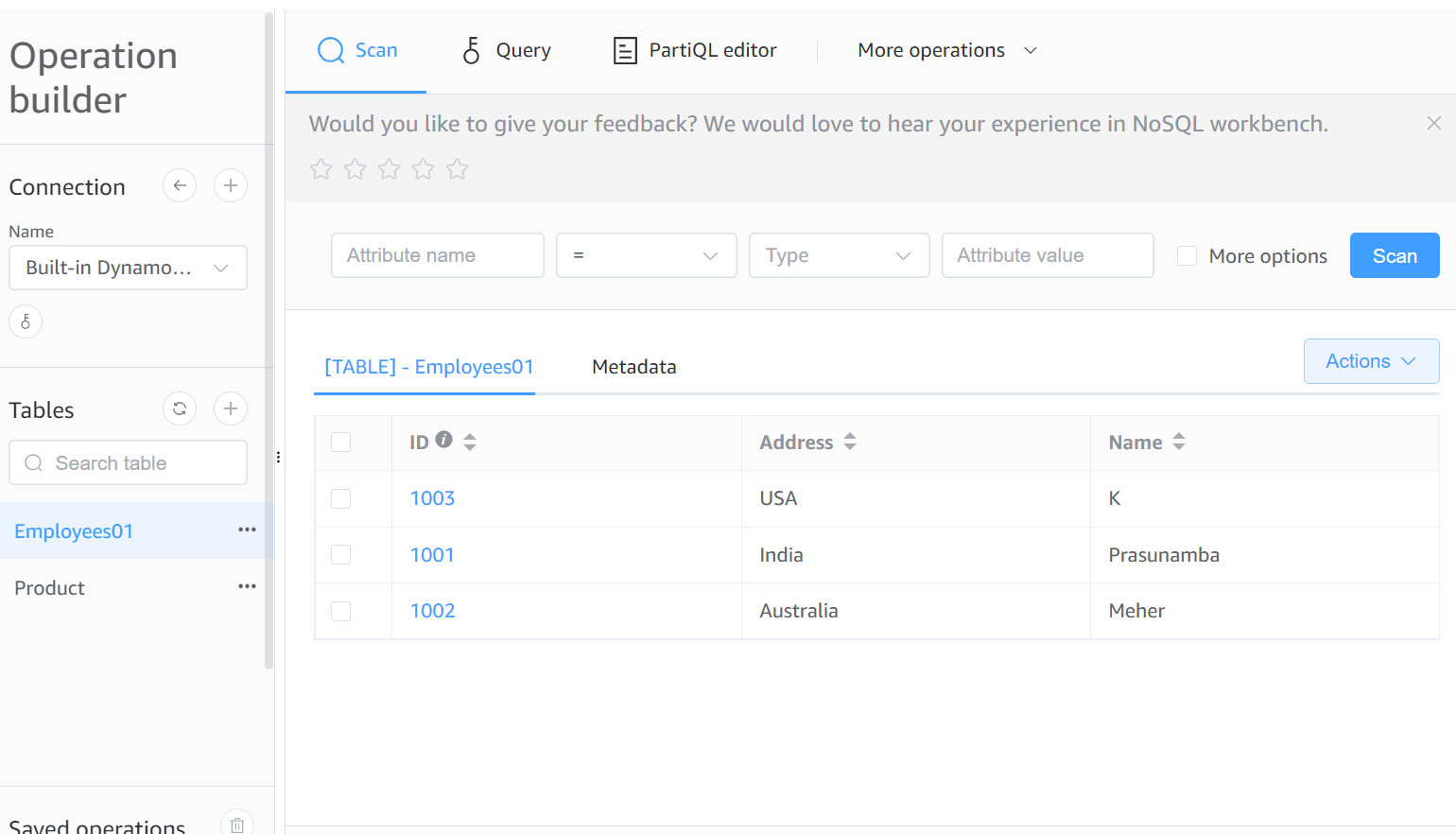
"ID": 1003,

"Name": "K",

"Address": "USA"

}

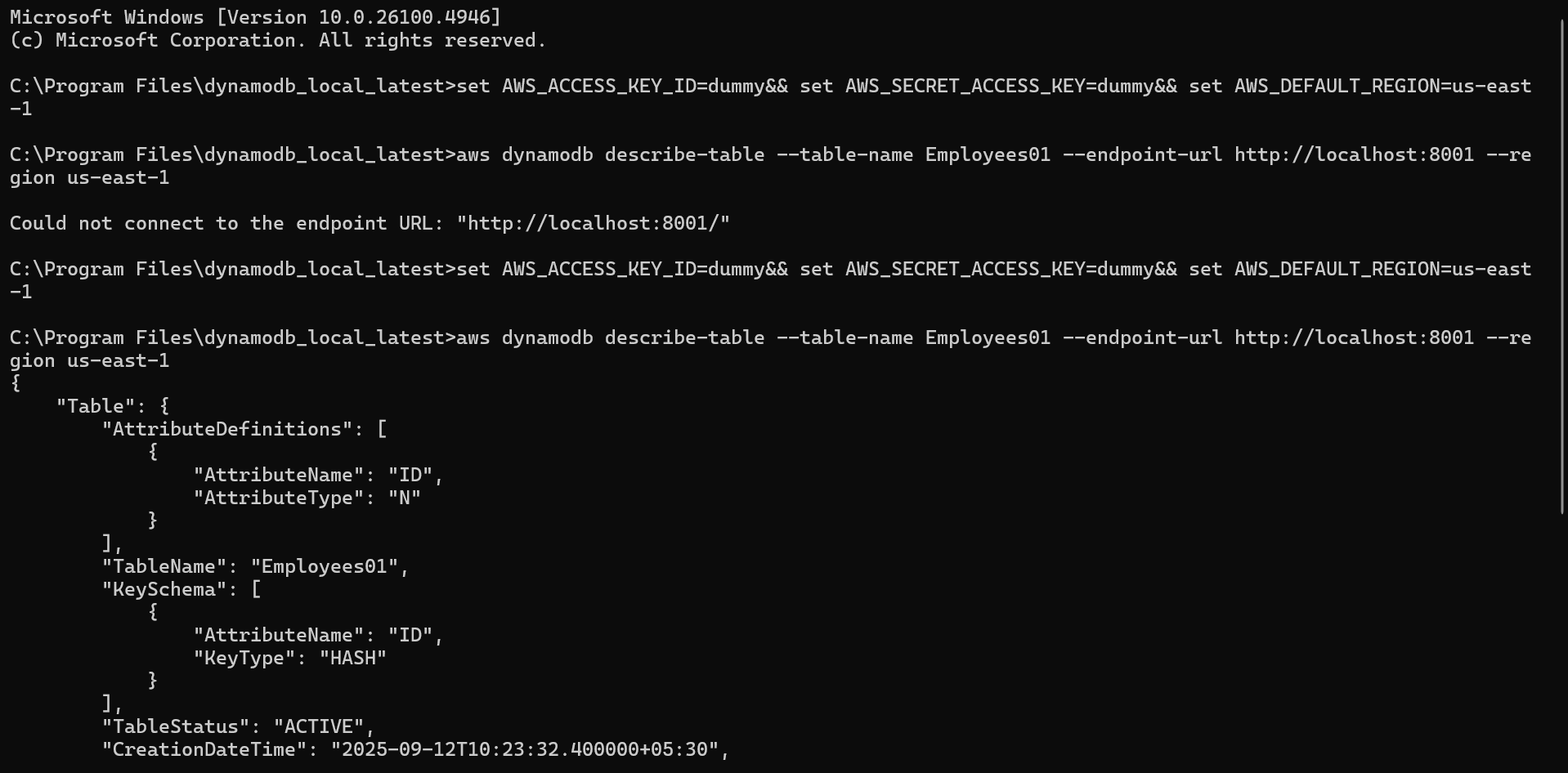
]



java -Djava.library.path=./DynamoDBLocal\_lib -jar DynamoDBLocal.jar -sharedDb

set AWS\_ACCESS\_KEY\_ID=dummy&& set AWS\_SECRET\_ACCESS\_KEY=dummy&& set AWS\_DEFAULT\_REGION=us-east-1

aws dynamodb describe-table --table-name Employees01 --endpoint-url http://localhost:8001 --region us-east-1



Task 02:

Using the same above java code

Change the post no and table name

To see the table reflecting in your cli prompt:

package org.example;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;

import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;

import software.amazon.awssdk.regions.Region;

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import software.amazon.awssdk.services.dynamodb.model.\*;

import java.net.URI;

//create table in DynamoDB

public class Demo04 {

public static void main(String[] args) throws Exception {

System.*out*.println("hello create table in DynamoDB");

AwsBasicCredentials awsCreds = AwsBasicCredentials.*create*("fakeAccesskey","fakeSecretKey");

DynamoDbClient client = DynamoDbClient.*builder*()

.endpointOverride(URI.*create*("http://localhost:8001"))

.region(Region.*AP\_SOUTH\_1*)

.credentialsProvider(StaticCredentialsProvider.*create*(awsCreds))

.build();

String tableName = "Employees02";

try {

CreateTableRequest request = CreateTableRequest.*builder*()

.tableName(tableName)

.keySchema(KeySchemaElement.*builder*()

.attributeName("ID")

.keyType(KeyType.*HASH*)

.build())

.attributeDefinitions(AttributeDefinition.*builder*()

.attributeName("ID")

.attributeType(ScalarAttributeType.*N*)

.build())

.provisionedThroughput(ProvisionedThroughput.*builder*()

.readCapacityUnits(5L)

.writeCapacityUnits(5L)

.build())

.build();

client.createTable(request);

System.*out*.println(tableName + " table is created. ");

}catch (ResourceInUseException ex) {

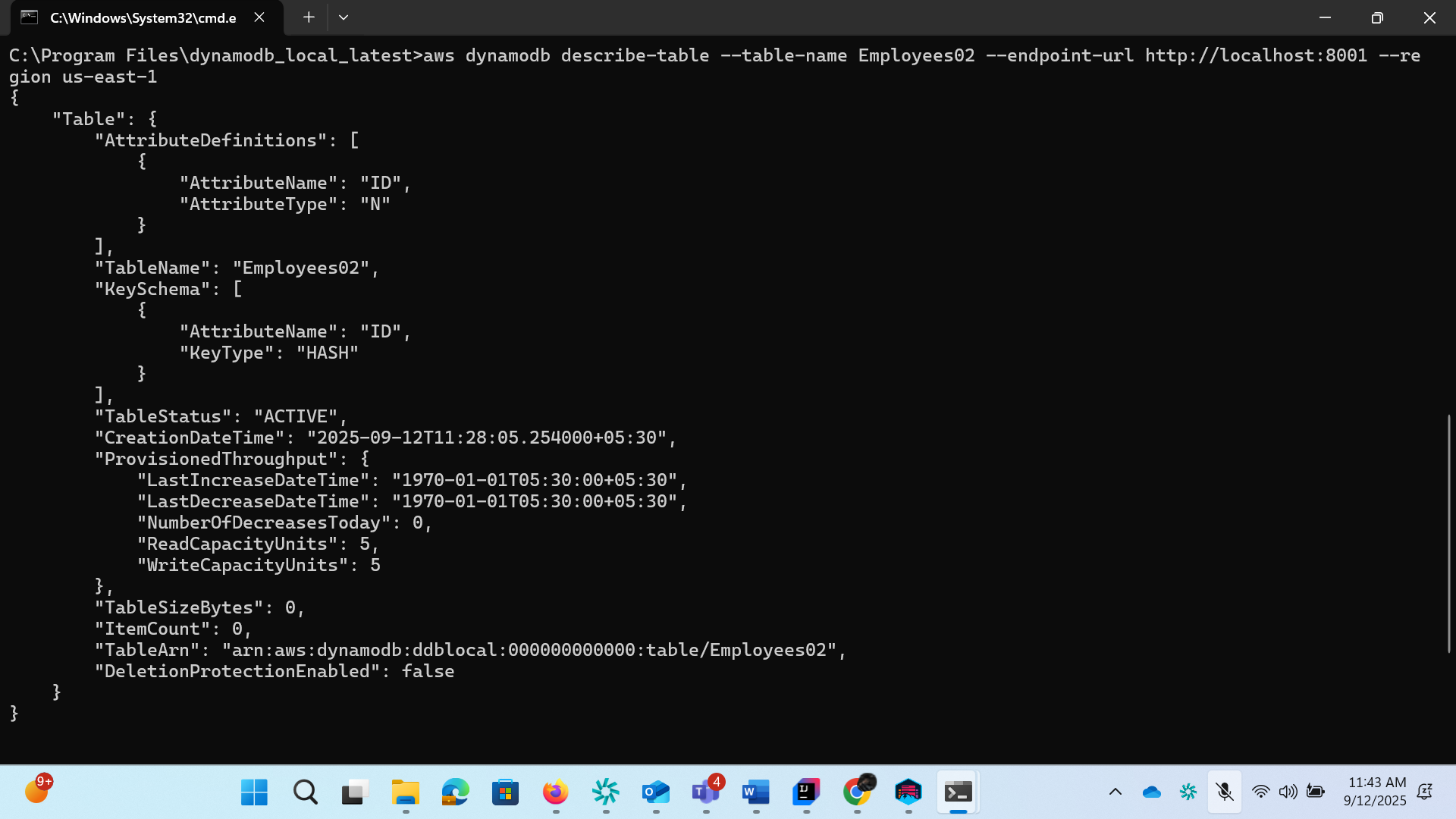
System.*out*.println(" plz choose different tablename as it already exists");

}

client.close();

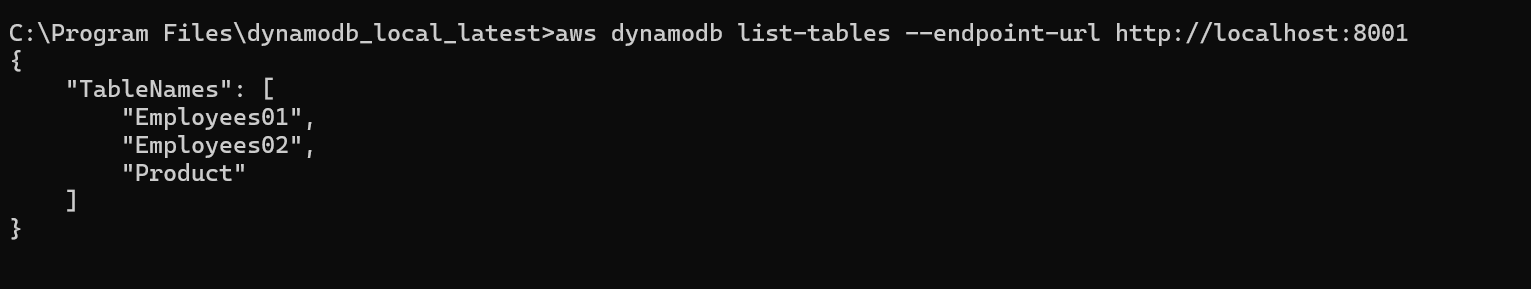
}

}



**List of tables in host**

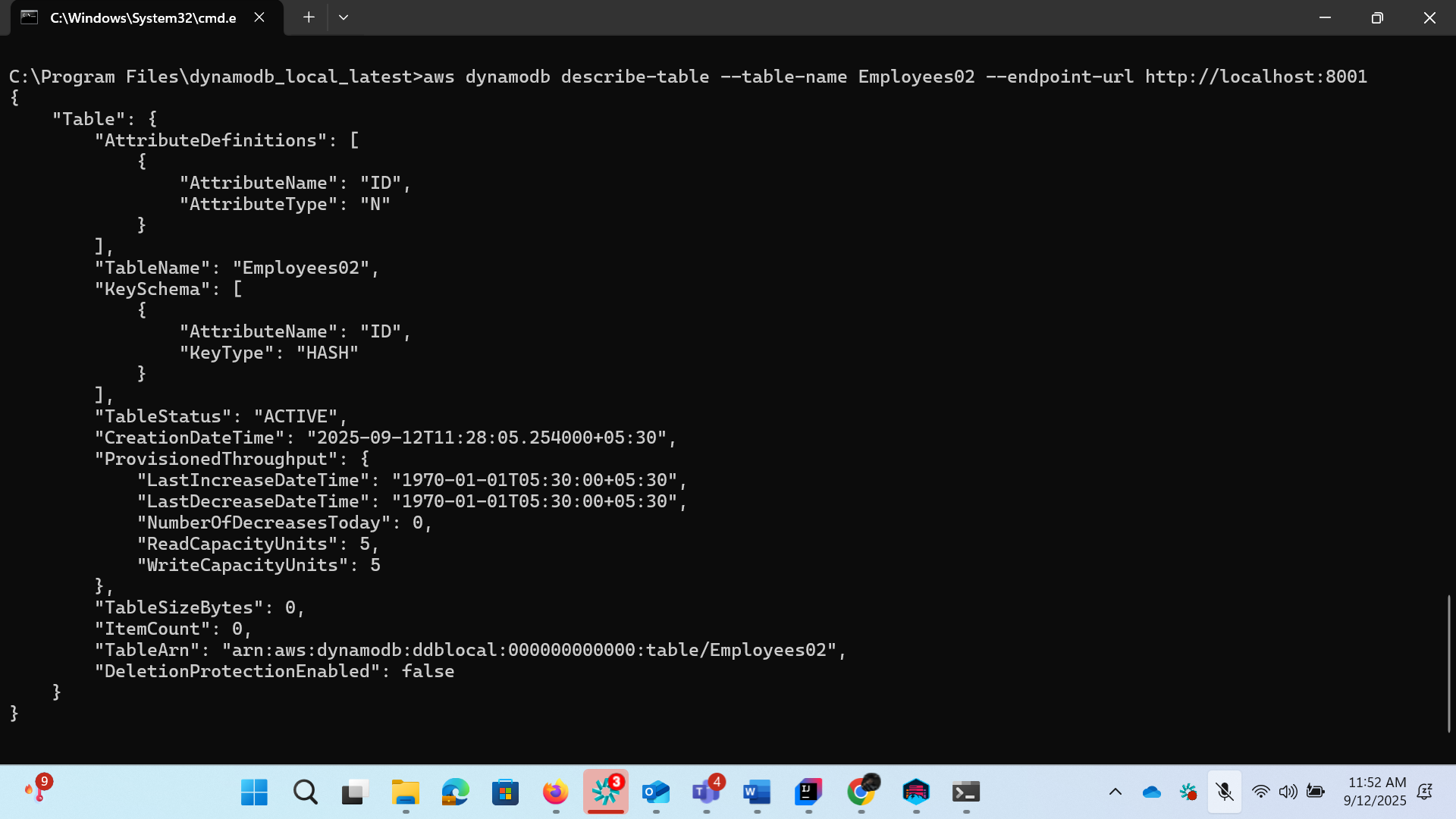
aws dynamodb list-tables --endpoint-url http://localhost:8001



Task 03:

For the same above tables see the description of the tables..

aws dynamodb describe-table --table-name Employees02 --endpoint-url http://localhost:8001



Task 04:

Loading / inserting data to the table

Plz make sure your json file is these in the resources folder

package org.example;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;

import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;

import software.amazon.awssdk.regions.Region;

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import software.amazon.awssdk.services.dynamodb.model.AttributeValue;

import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;

import java.io.InputStream;

import java.net.URI;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

// loading data to our DynamoDB table

public class LoadingData02 {

public static void main(String[] args) throws Exception {

// using theb Aws credentials

AwsBasicCredentials awsCreds = AwsBasicCredentials.*create*("fakeaccess", "fakeaccess");

// create a DynamoDb client

DynamoDbClient client = DynamoDbClient.*builder*()

.endpointOverride(URI.*create*("http://localhost:8001"))

.region(Region.*AP\_SOUTH\_1*)

.credentialsProvider(StaticCredentialsProvider.*create*(awsCreds))

.build();

System.*out*.println("connection created successfully");

String tableName = "Employees02";

//object Mapper

ObjectMapper mapper = new ObjectMapper();

//giving input stream of data

InputStream stream = LoadingData02.class.getClassLoader()

.getResourceAsStream(("Employee.json"));

System.*out*.println("the json file in the input stream");

JsonNode node = mapper.readTree(stream);

Iterator<JsonNode> iterator = node.elements();

while (iterator.hasNext()) {

JsonNode Jsonnode2 = iterator.next();

Map<String, AttributeValue> item = new HashMap<>();

item.put("ID", AttributeValue.*builder*().n(Jsonnode2.get("ID").asText()).build());

item.put("Name", AttributeValue.*builder*().s(Jsonnode2.get("Name").asText()).build());

item.put("Address", AttributeValue.*builder*().s(Jsonnode2.get("Address").asText()).build());

PutItemRequest request = PutItemRequest.*builder*()

.tableName(tableName)

.item(item)

.build();

client.putItem(request);

System.*out*.println(" loading the data to the table");

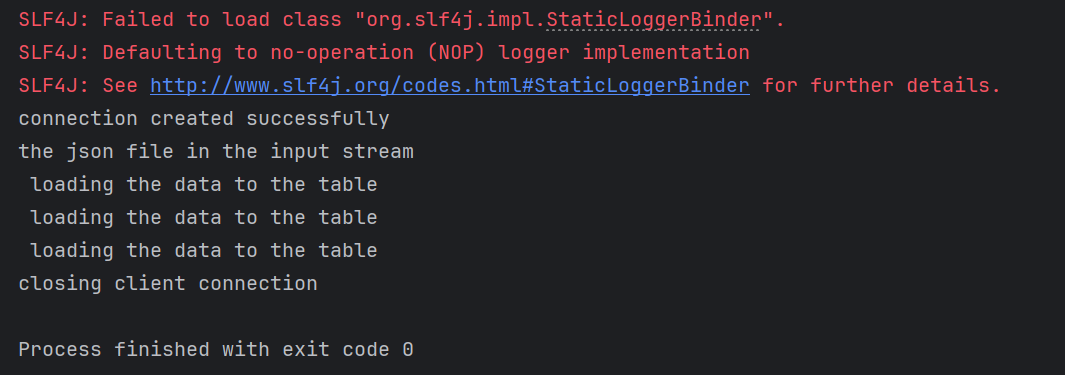
}

client.close();

System.*out*.println("closing client connection");

}

}

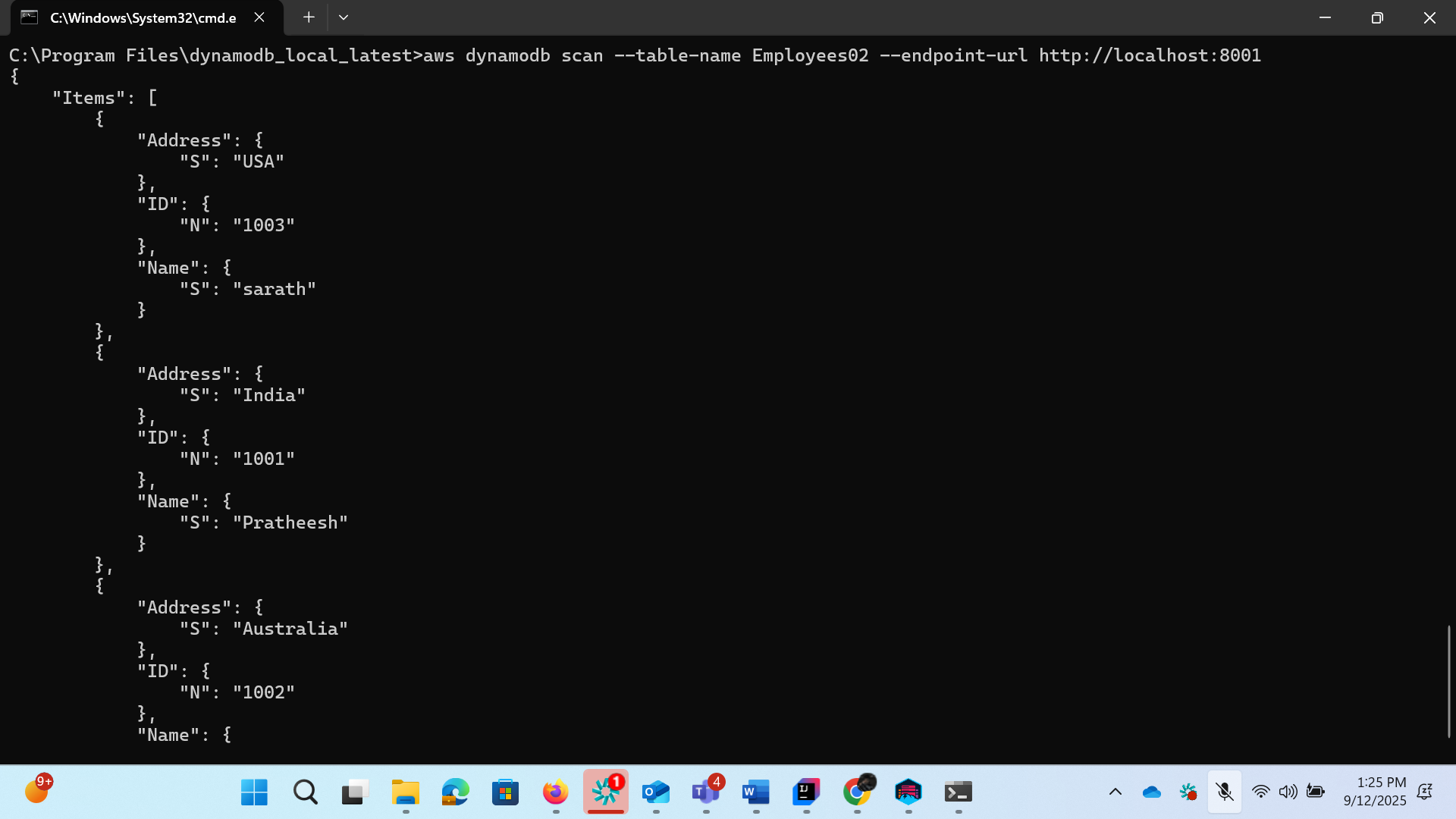


To check if the table is created… in the server

Your server should be open on the same port no

In another cmd promt type the below command to scan the items:

aws dynamodb scan --table-name Employees04 --endpoint-url http://localhost:8000



Task 05:

Reading / Scanning the data from the server.. Using java code..

package org.example;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;

import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;

import software.amazon.awssdk.regions.Region;

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import software.amazon.awssdk.services.dynamodb.model.AttributeValue;

import software.amazon.awssdk.services.dynamodb.model.ScanRequest;

import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

import java.net.URI;

import java.util.Map;

public class ScanData {

public static void main(String[] args) {

// Create DynamoDB client

DynamoDbClient client = DynamoDbClient.*builder*()

.endpointOverride(URI.*create*("http://localhost:8001")) // DynamoDB Local endpoint

.region(Region.*AP\_SOUTH\_1*)

.credentialsProvider(

StaticCredentialsProvider.*create*(

AwsBasicCredentials.*create*("fakeAccessKey", "fakeSecretKey")

)

)

.build();

String tableName = "Employees02"; // replace with your table name

try {

// Create a ScanRequest

ScanRequest scanRequest = ScanRequest.*builder*()

.tableName(tableName)

.build();

// Execute scan

ScanResponse response = client.scan(scanRequest);

System.*out*.println("Scanning data from table: " + tableName);

for (Map<String, AttributeValue> item : response.items()) {

// Print each item

System.*out*.println("---------------------------");

item.forEach((key, value) -> {

if (value.s() != null) {

System.*out*.println(key + ": " + value.s());

} else if (value.n() != null) {

System.*out*.println(key + ": " + value.n());

} else {

System.*out*.println(key + ": " + value.toString());

}

});

}

} catch (Exception e) {

System.*err*.println("Error scanning table: " + e.getMessage());

e.printStackTrace();

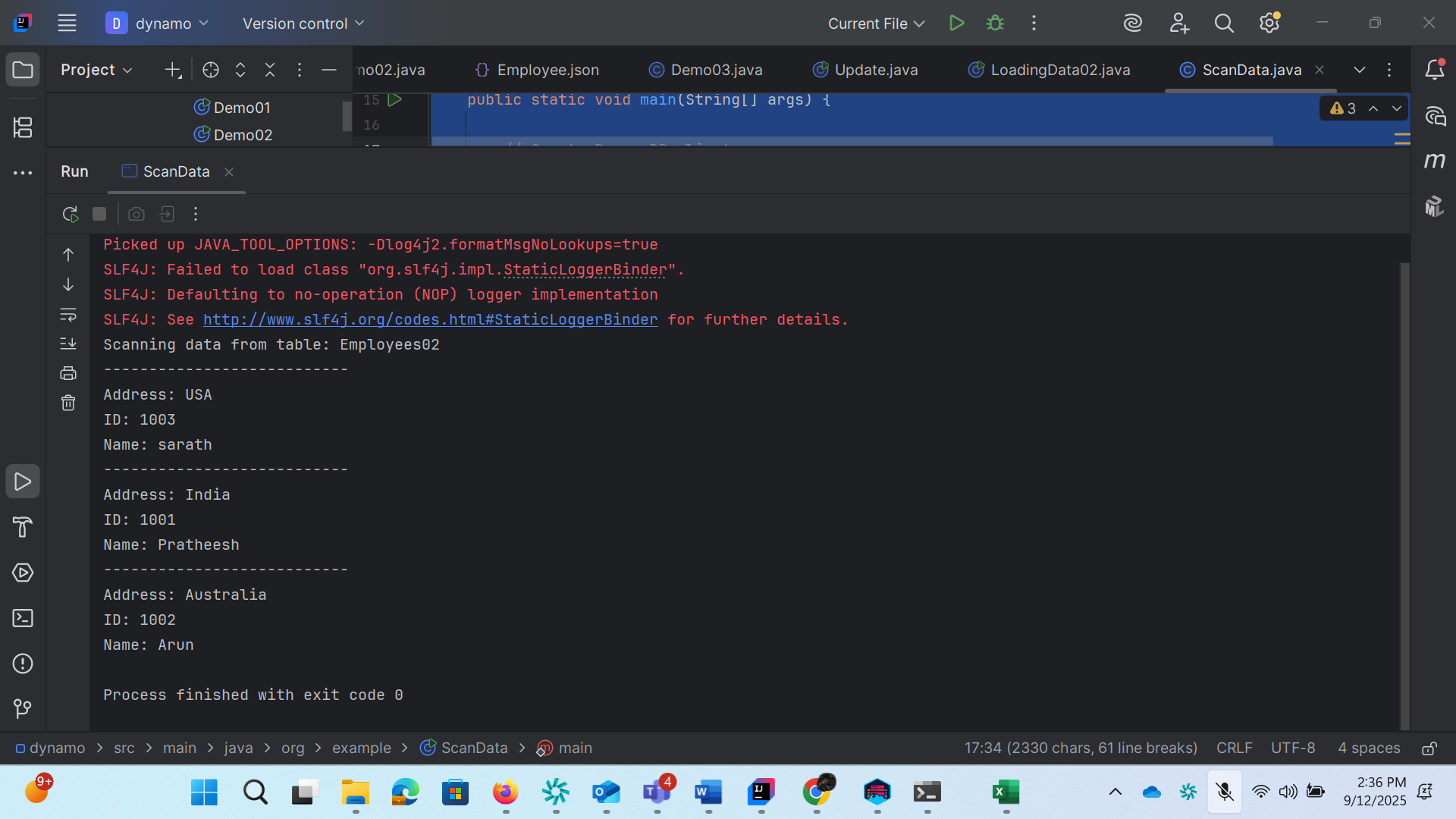
} finally {

client.close();

}

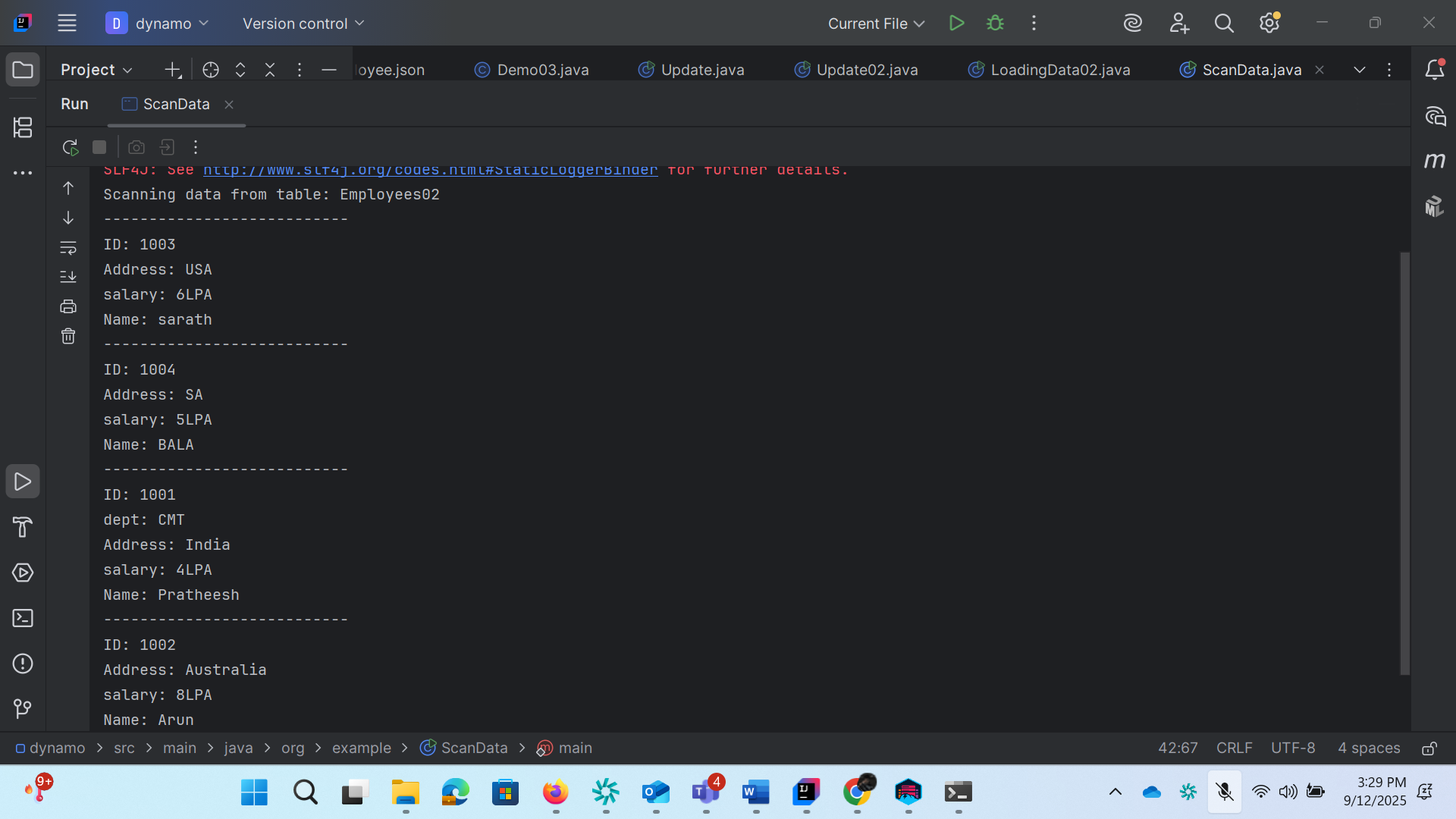
}

}



Task 06:

Insert an item in the nosql workbench.. And check if the item reflects in your java output…

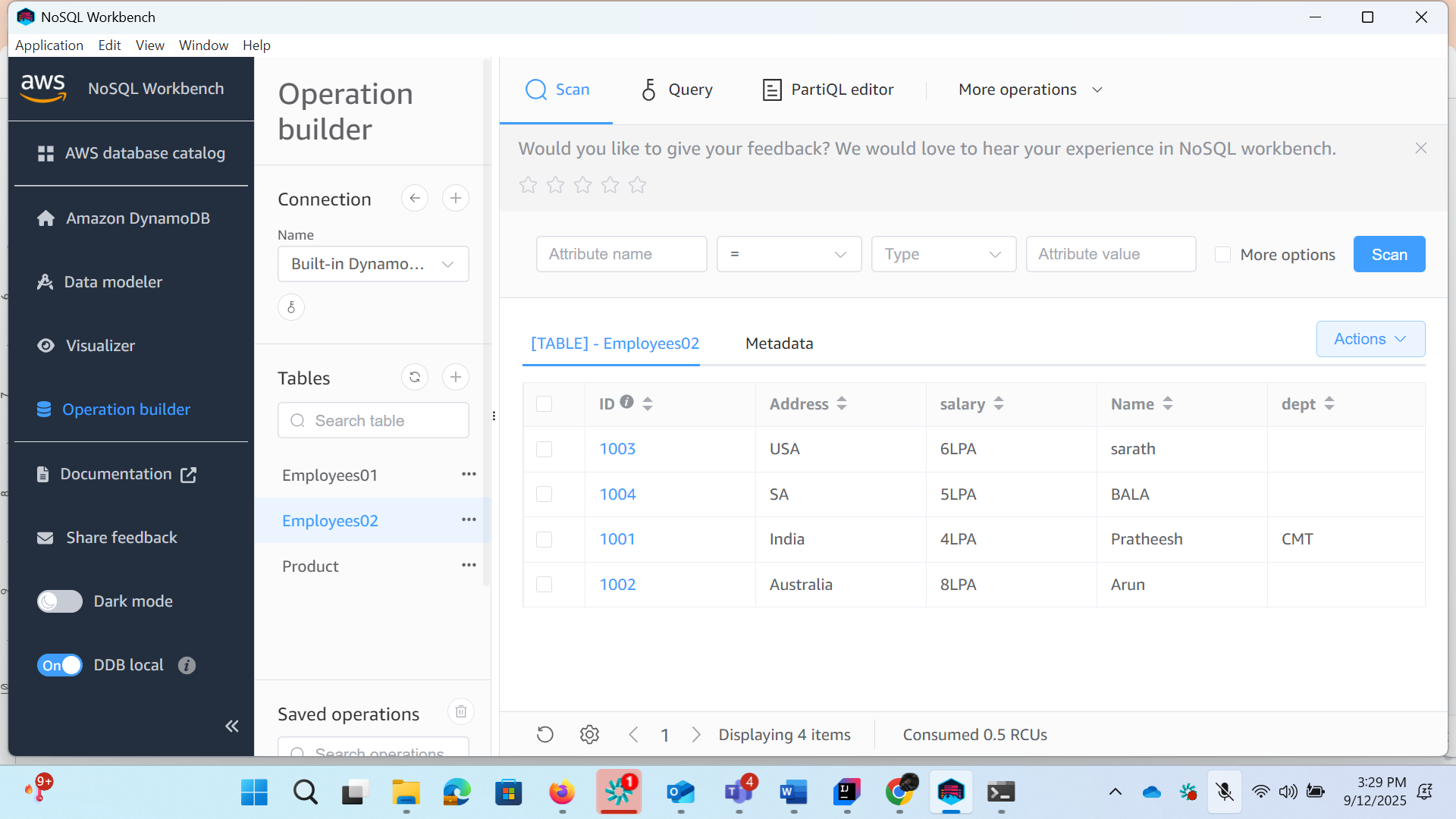


Task 07:

How do you lower the cost of DynomaDB..explain ways to do so…

Task 08:

Update item details using java code.. And check if it reflects in the server..



Task 09:

Delete a particular item from the table …

package org.example;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;

import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;

import software.amazon.awssdk.regions.Region;

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import software.amazon.awssdk.services.dynamodb.model.AttributeValue;

import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;

import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;

import java.net.URI;

import java.util.HashMap;

import java.util.Map;

public class DeleteItemExample {

public static void main(String[] args) {

// 1. Create DynamoDB client

DynamoDbClient client = DynamoDbClient.*builder*()

.endpointOverride(URI.*create*("http://localhost:8001")) // DynamoDB local

.region(Region.*AP\_SOUTH\_1*)

.credentialsProvider(StaticCredentialsProvider.*create*(

AwsBasicCredentials.*create*("fakeAccessKey", "fakeSecretKey")

))

.build();

String tableName = "Employees02";

// 2. Define primary key of the item to delete

Map<String, AttributeValue> key = new HashMap<>();

key.put("ID", AttributeValue.*builder*().n("1004").build());

// 3. Build DeleteItemRequest

DeleteItemRequest deleteRequest = DeleteItemRequest.*builder*()

.tableName(tableName)

.key(key)

.build();

// 4. Perform delete

client.deleteItem(deleteRequest);

System.*out*.println("Item deleted successfully!");

// 5. Try to fetch the item to verify

GetItemRequest getRequest = GetItemRequest.*builder*()

.tableName(tableName)

.key(key)

.build();

Map<String, AttributeValue> deletedItem = client.getItem(getRequest).item();

if (deletedItem == null || deletedItem.isEmpty()) {

System.*out*.println("Verified: Item no longer exists in table.");

} else {

System.*out*.println("Delete failed, item still exists: " + deletedItem);

}

client.close();

}

}

