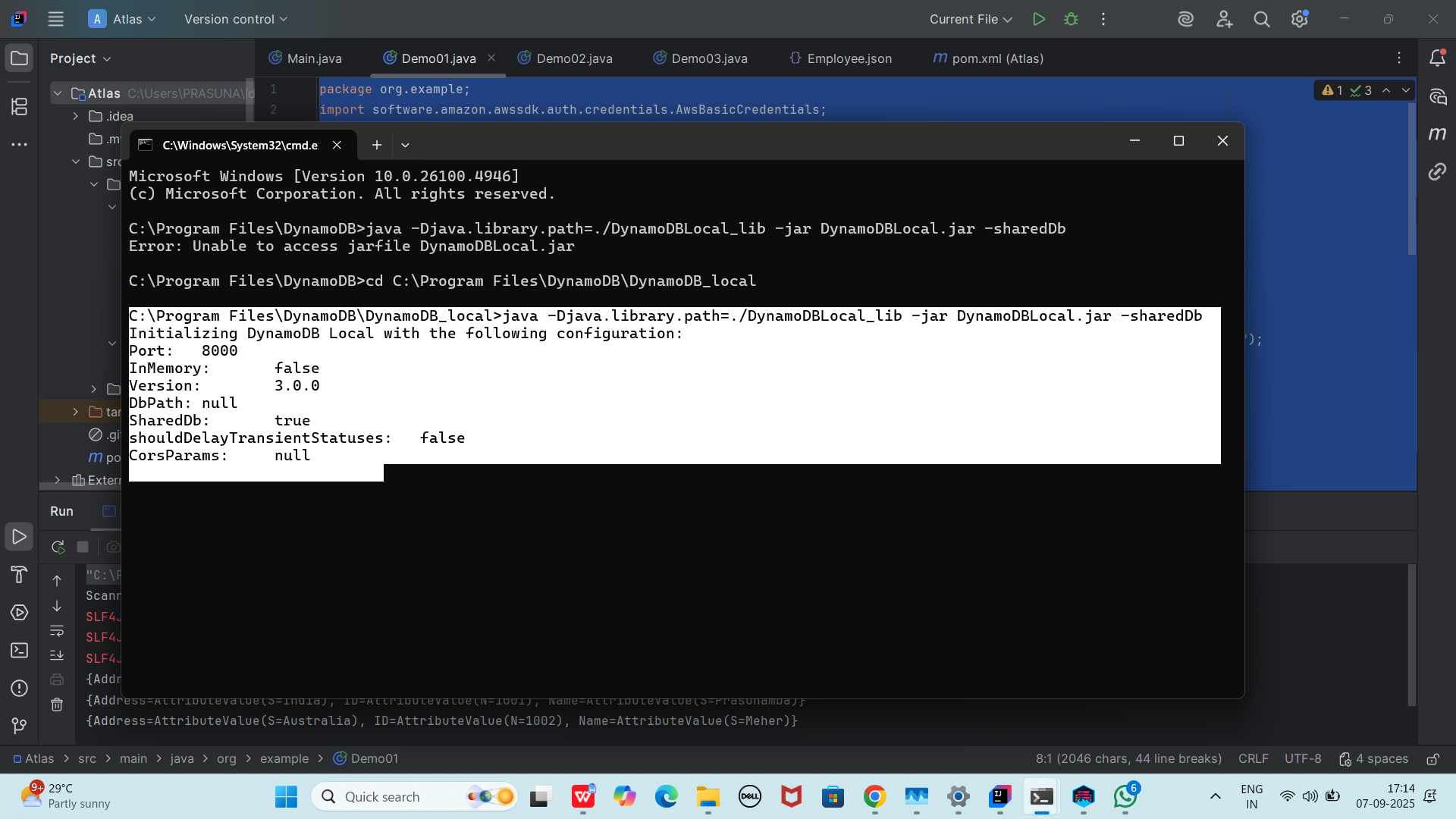
Day 31 - 07th sept 2025 - DynamoDB

My Server is at C:\Program Files\DynamoDB\DynamoDB\_local>

So open cmd at that location and run below command

Run the Server

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



Pom.xml

<dependencies>

<!-- AWS SDK v2 for DynamoDB -->

<!-- https://mvnrepository.com/artifact/software.amazon.awssdk/dynamodb -->

<dependency>

<groupId>software.amazon.awssdk</groupId>

<artifactId>dynamodb</artifactId>

<version>2.33.4</version>

</dependency>

<!-- Jackson (for JSON) -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.17.2</version>

</dependency>

</dependencies>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>1.7.36</version> *<!-- Use the latest stable version -->*

</dependency>

Create a 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.\*;

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();

}

}

Output:

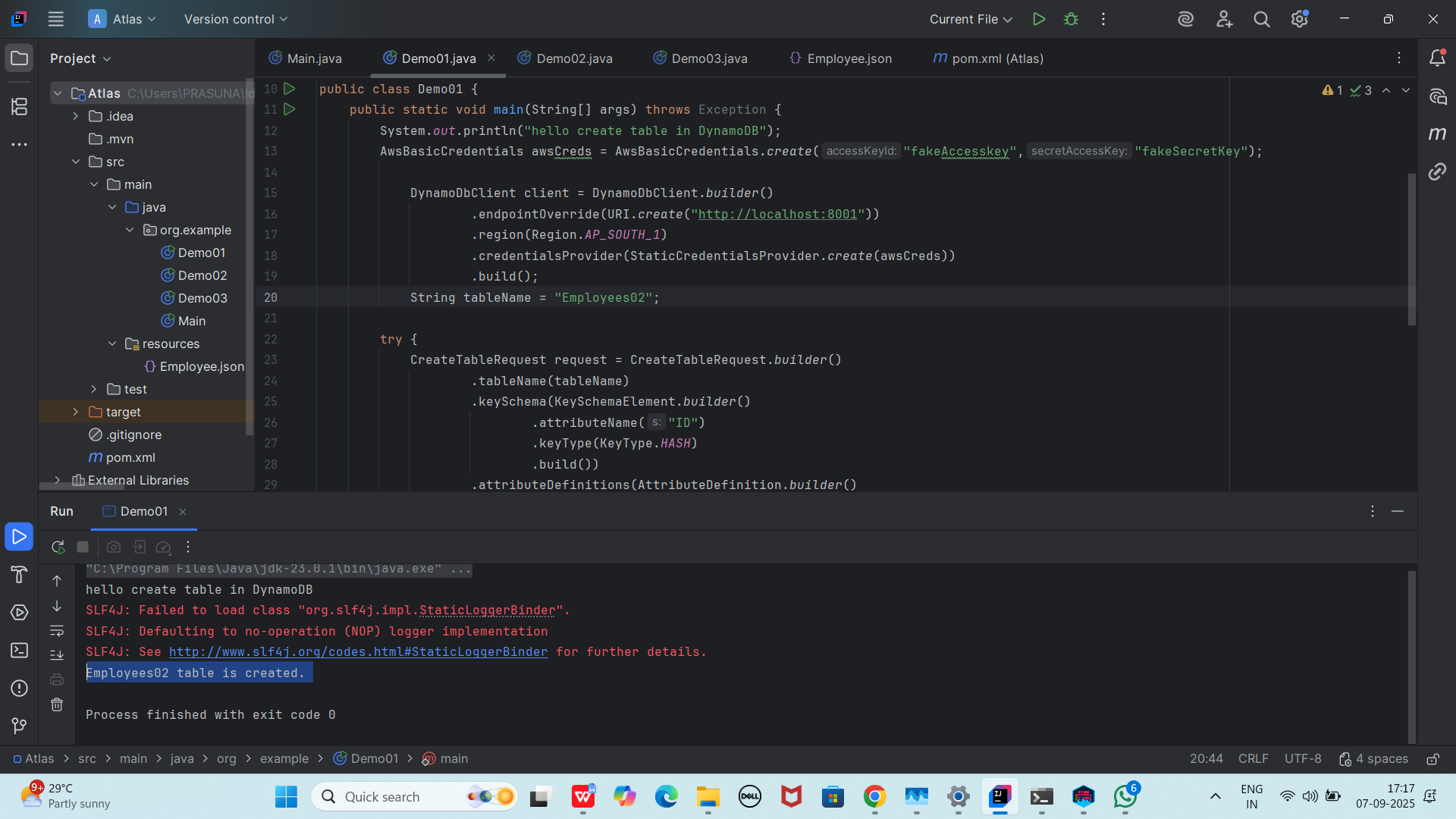
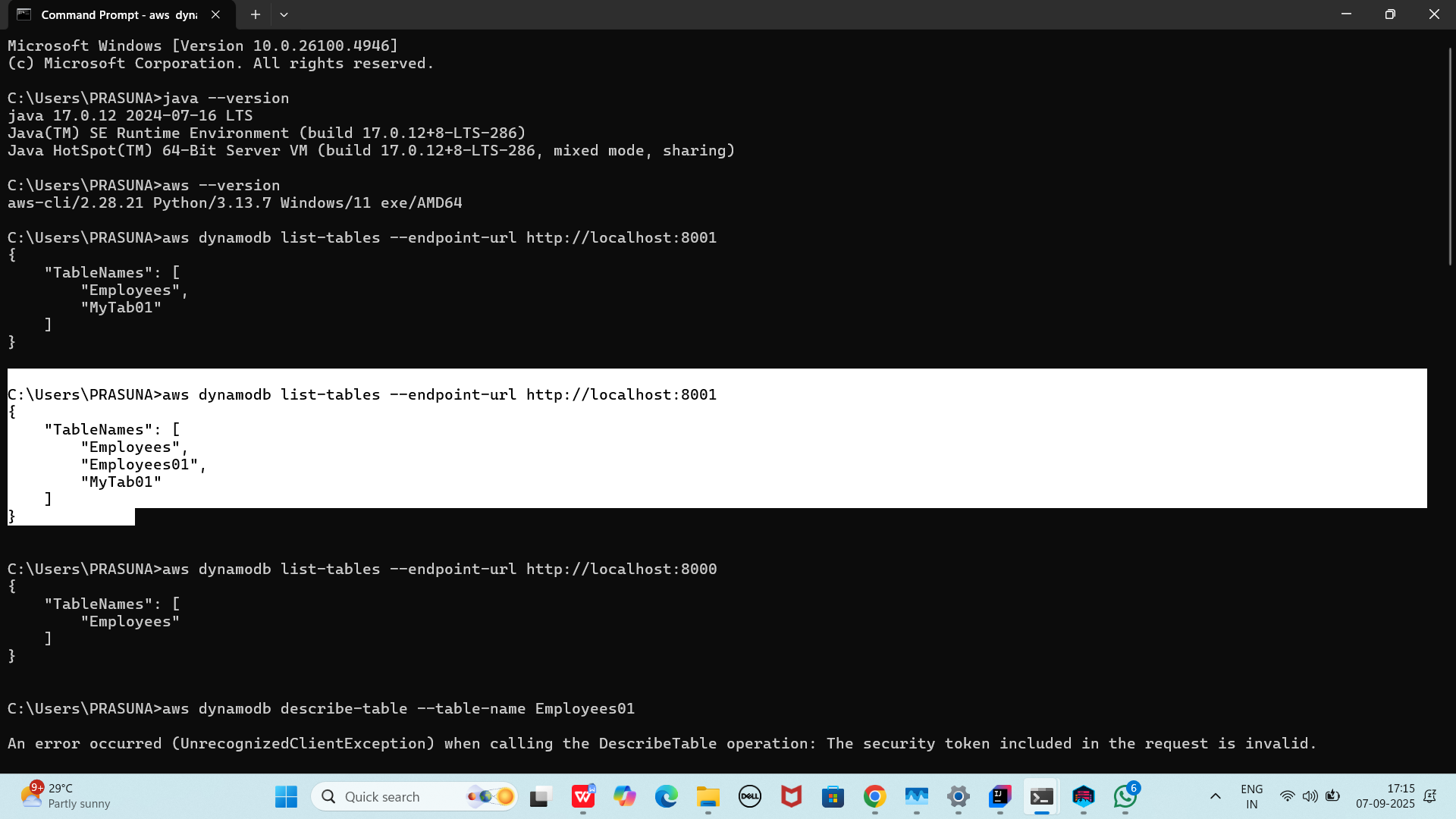
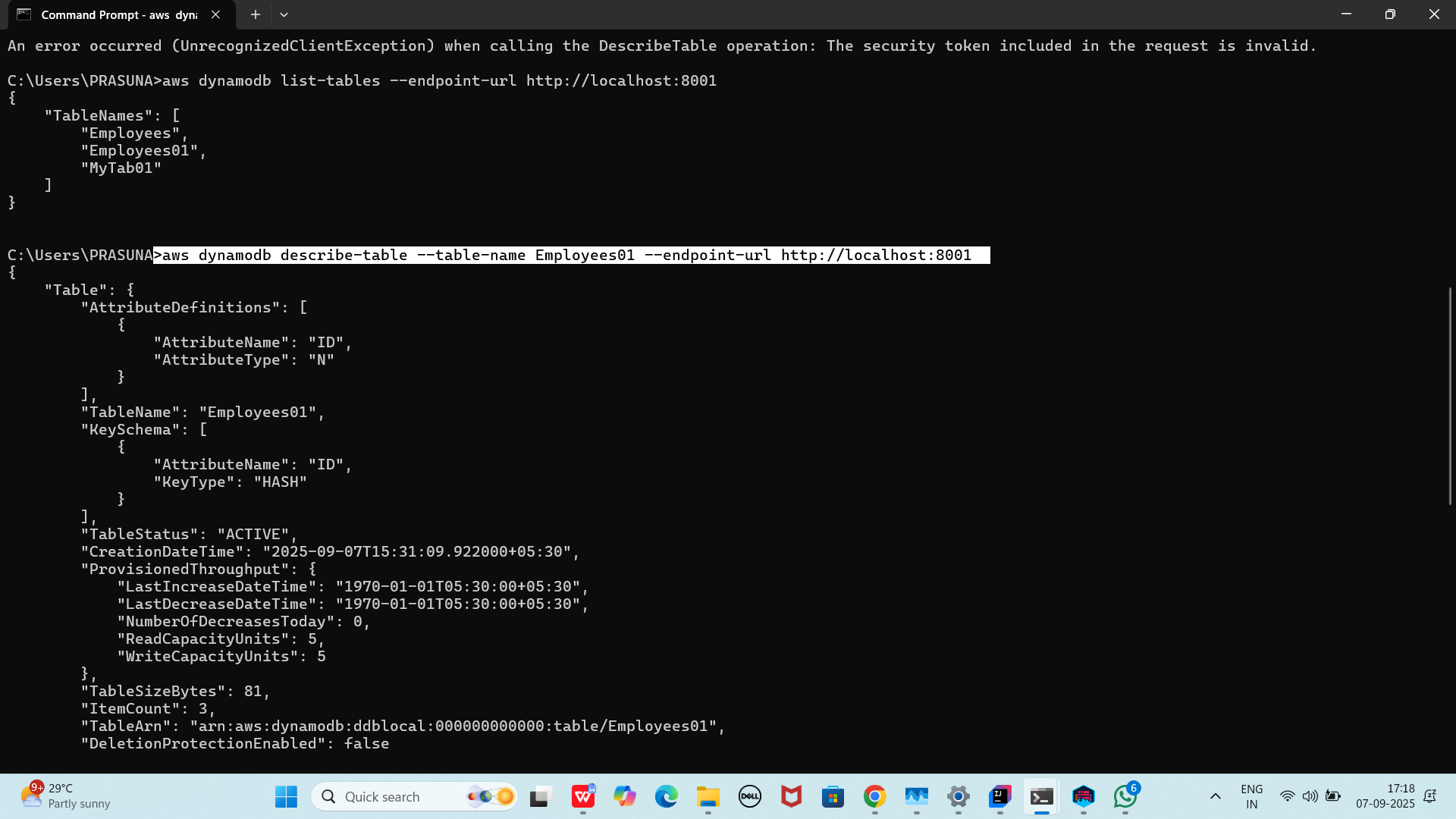


Table created Employees01



To see the description of the table 👍

>aws dynamodb describe-table --table-name Employees01 --endpoint-url http://localhost:8001



Loading data to the table // inserting data

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 ");

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";

ObjectMapper mapper = new ObjectMapper();

InputStream jsonStream = Demo02.class.getClassLoader()

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

if(jsonStream == null) {

System.*err*.println("plz provide json file");

System.*exit*(1);

}

JsonNode rootNode = mapper.readTree(jsonStream);

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

//now inserting data to your table at port no 800\*

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*.println("inserted data "+ node.get("ID").asInt()+ " \*\*\*\*\* " + node.get("Name").asText()+" $$$ "+node.get("Address").asText());

}

client.close();

}

}

This is my json file Employee.json

[

{

"ID": 1001,

"Name": "Prasunamba",

"Address": "India"

},

{

"ID": 1002,

"Name": "Meher",

"Address": "Australia"

},

{

"ID": 1003,

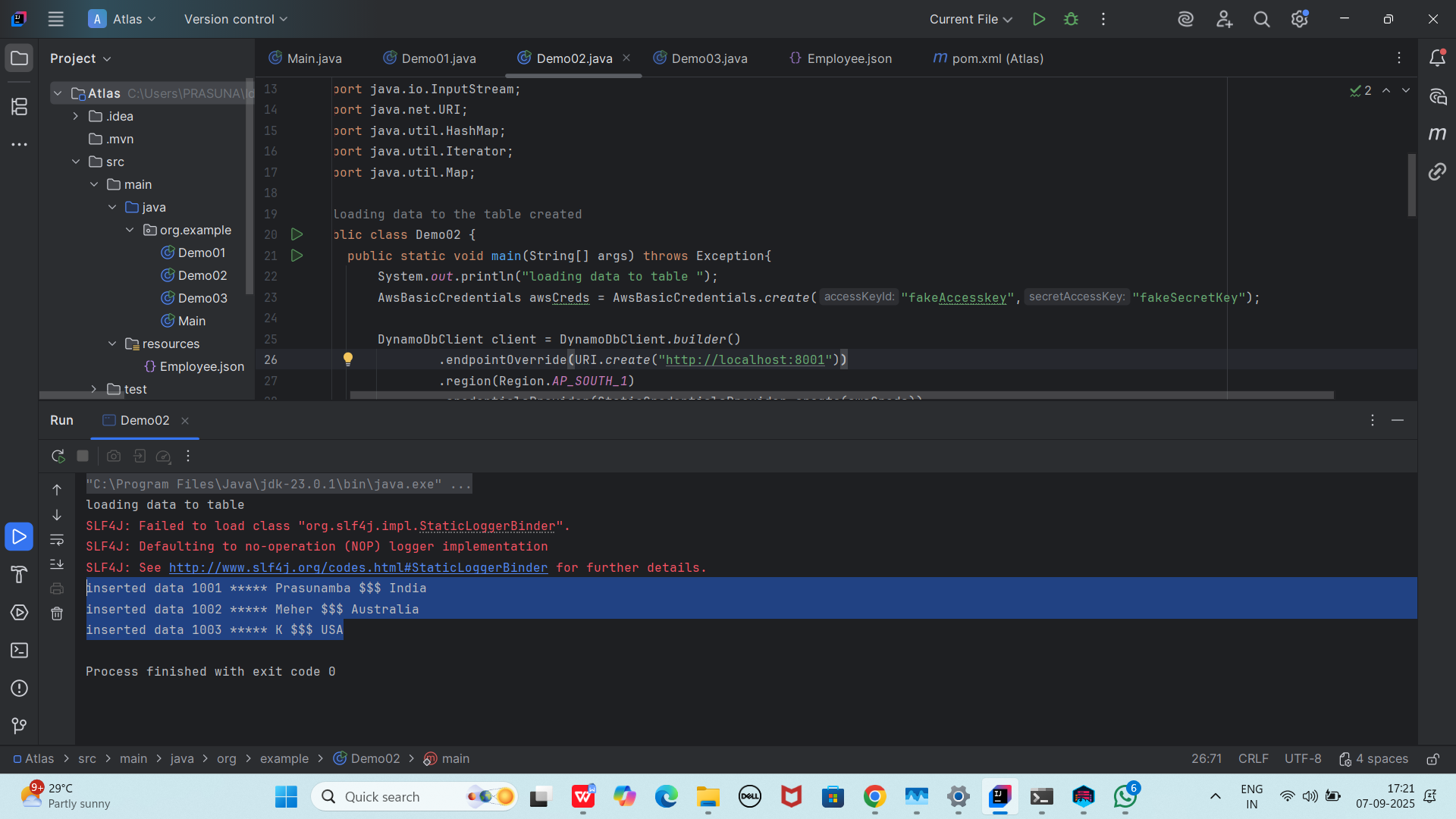
"Name": "K",

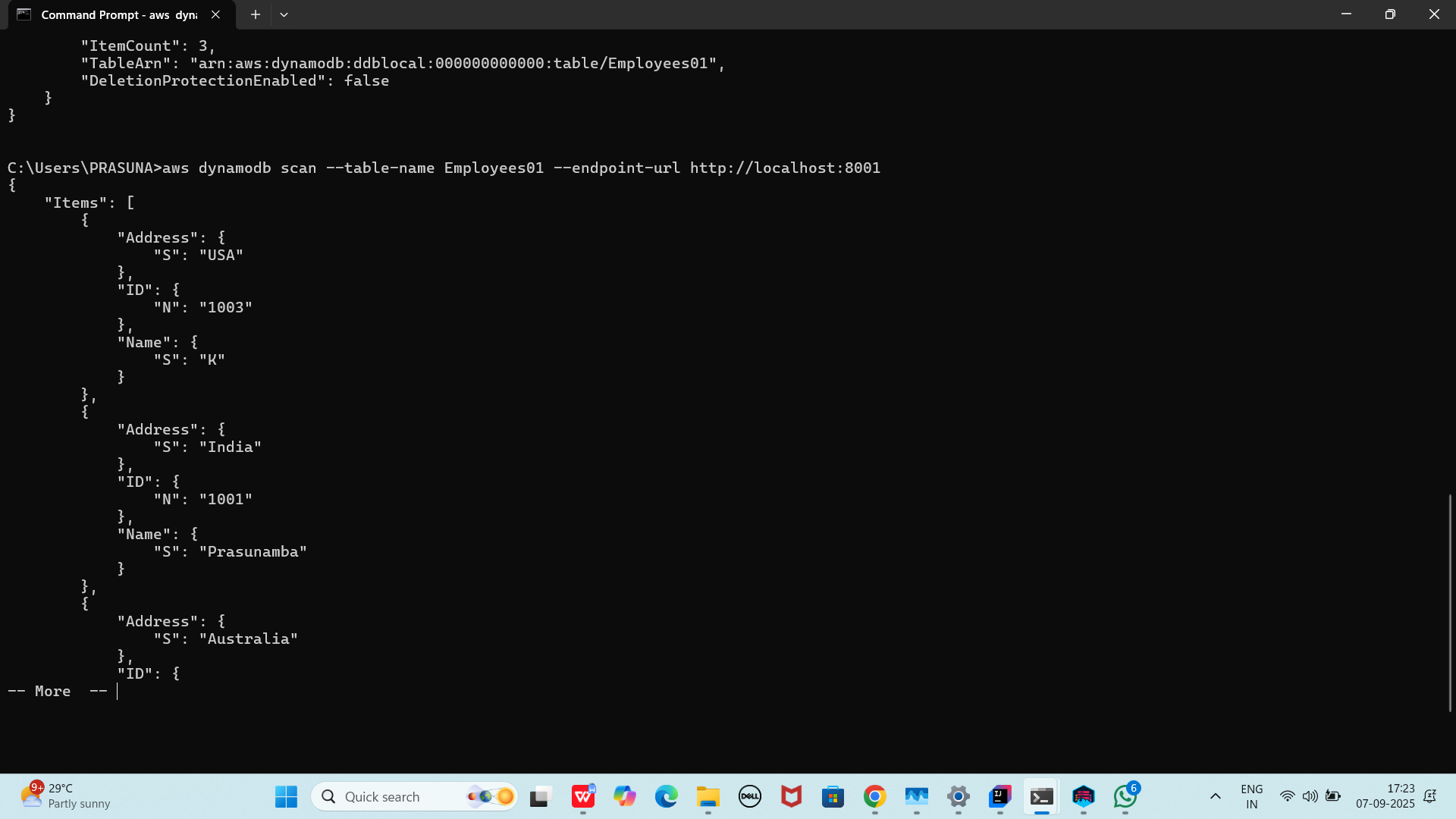
"Address": "USA"

}

]

Output:





Scanning data from the table // getting data // loading data

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 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);

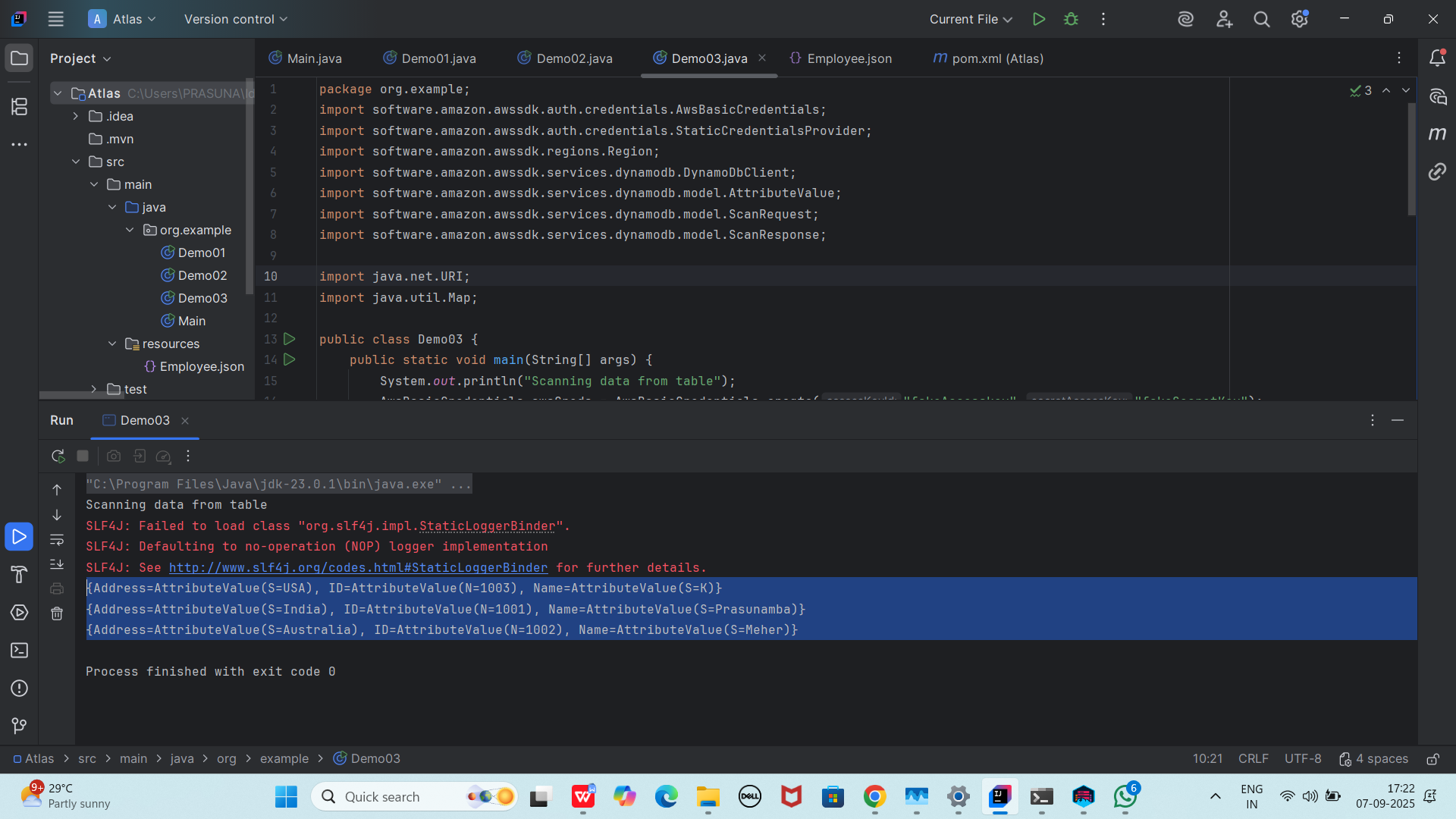
}

client.close();

}

}

Output:



Home Task:

Plz work on

Task 01:

Delete a record operation

Task 02:

Delete a table

Task 03:

Update a record in a table

—----------------------------------------------------------------------------------------------------------------------------

Delete operation

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.GetItemResponse;

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

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

import java.net.URI;

import java.util.HashMap;

import java.util.Map;

public class DeleteItem {

public static void main(String[] args) {

System.*out*.println("hello Delete item by id from 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";

int delId = 1002;

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

item.put("ID", AttributeValue.*builder*().n(String.*valueOf*(delId)).build());

GetItemResponse getResponse =client.getItem(builder -> builder.tableName(tableName).key(item));

client.deleteItem(builder -> builder.tableName(tableName).key(item));

System.*out*.println("items after deletion");

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

ScanResponse resp = client.scan(req);

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

System.*out*.println(dbitem);

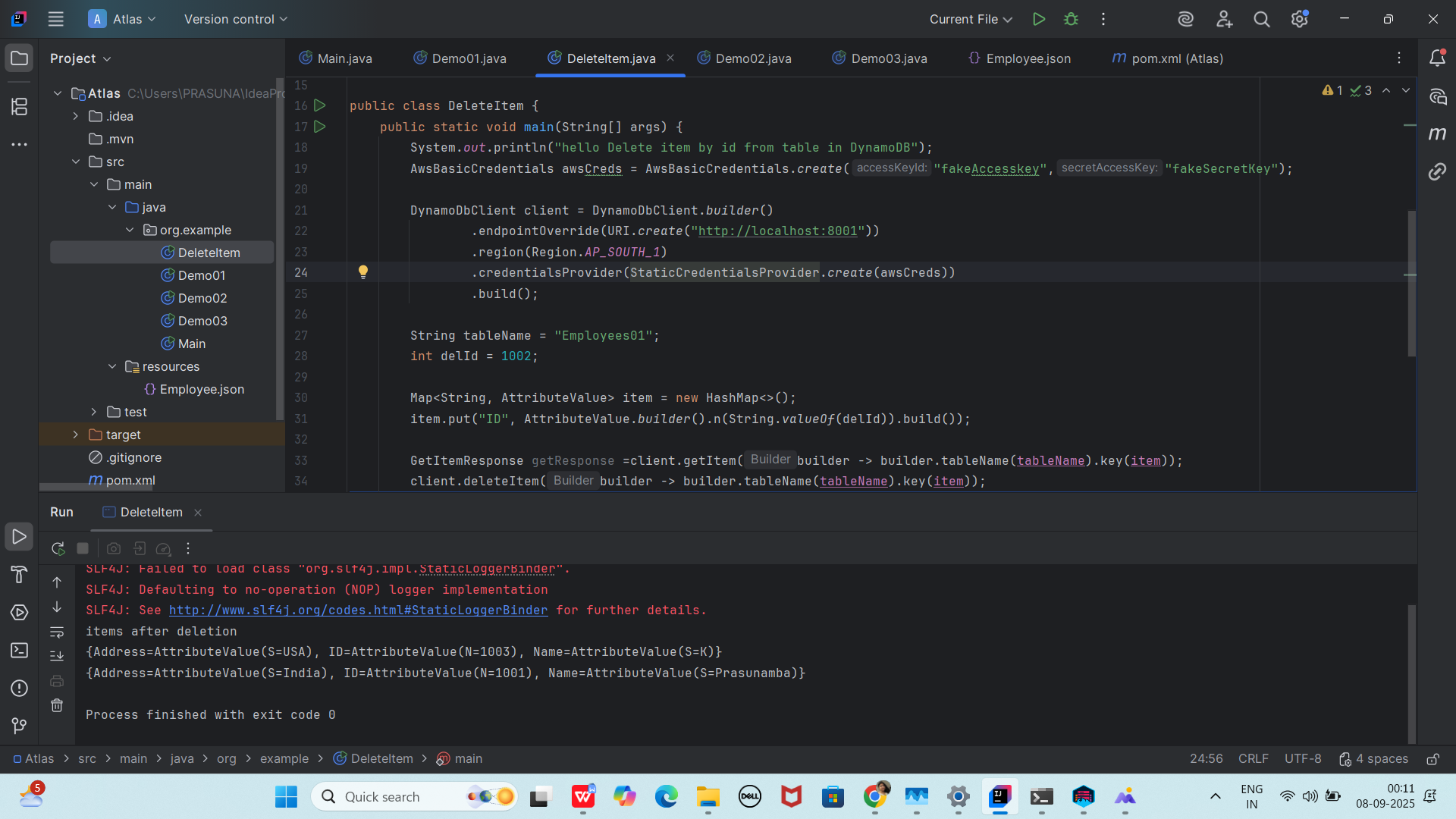
}

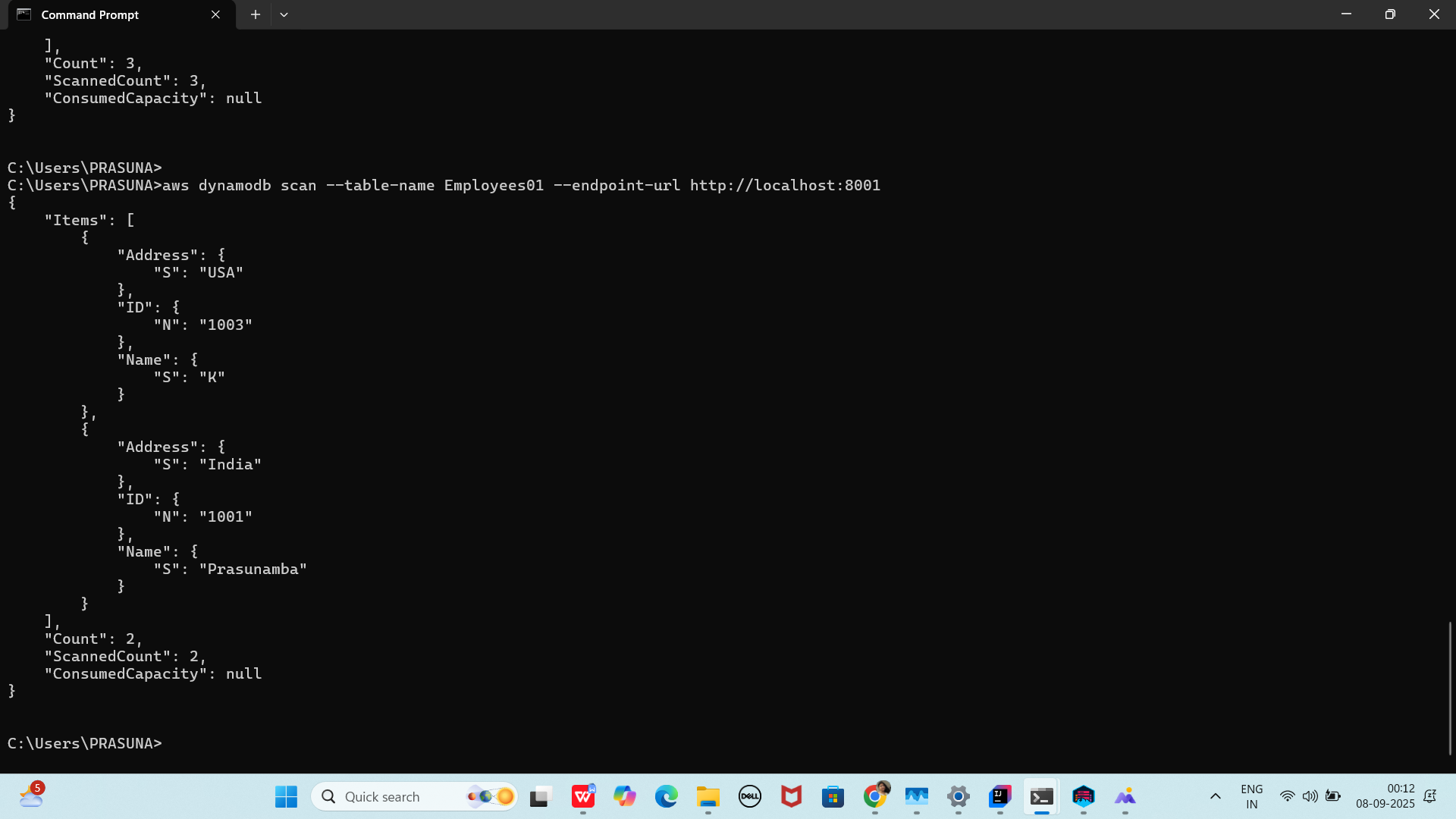
client.close();

}

}

Output:

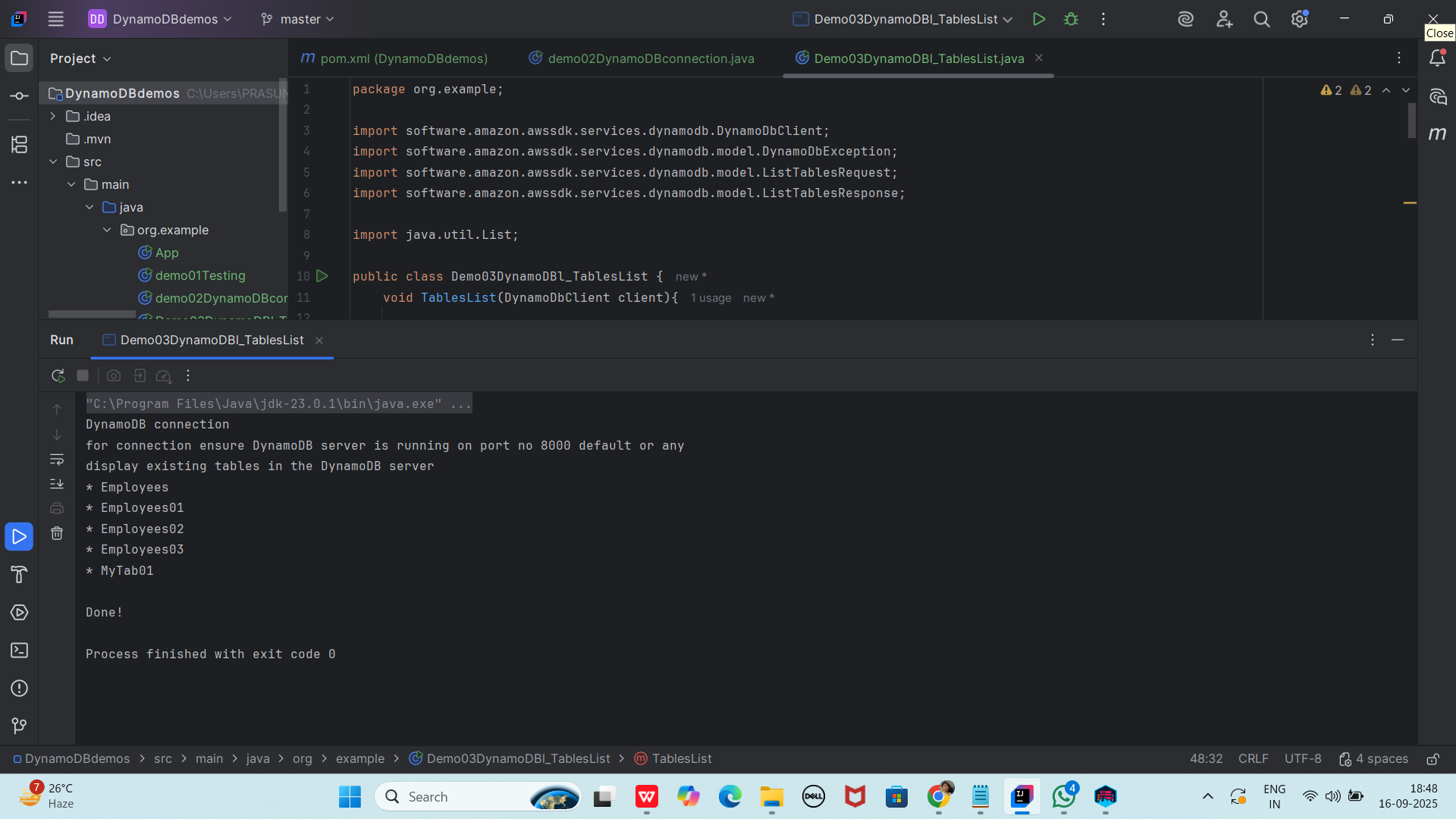




16th Sept 2025: 18.47

Add ons:

Displaying list of all tables in server:



Code:

package org.example;

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

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

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

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

import java.util.List;

public class Demo03DynamoDBl\_TablesList {

void tablesList(DynamoDbClient client){

System.*out*.println("display existing tables in the DynamoDB server");

boolean tables = true;

String lastName = null;

while (tables) {

try {

ListTablesResponse res = null;

if (lastName == null) {

ListTablesRequest req = ListTablesRequest.*builder*().build();

res = client.listTables(req);

} else {

ListTablesRequest req = ListTablesRequest.*builder*()

.exclusiveStartTableName(lastName).build();

res = client.listTables(req);

}

List<String> namesOfTables = res.tableNames();

if (namesOfTables.size() > 0) {

for (String currentName : namesOfTables) {

System.*out*.format("\* %s\n", currentName);

}

} else {

System.*out*.println("No tables found!");

System.*exit*(0);

}

lastName = res.lastEvaluatedTableName();

if (lastName == null) {

tables = false;

}

} catch (DynamoDbException ex) {

System.*err*.println(ex.getMessage());

System.*exit*(1);

}

}

System.*out*.println("\nDone!");

}

public static void main(String[] args) {

demo02DynamoDBconnection obj = new demo02DynamoDBconnection();

DynamoDbClient client = obj.dynamoDBConnection();

Demo03DynamoDBl\_TablesList obj2 = new Demo03DynamoDBl\_TablesList();

obj2.tablesList(client);

}

}

========================================================================================

Info BOX: