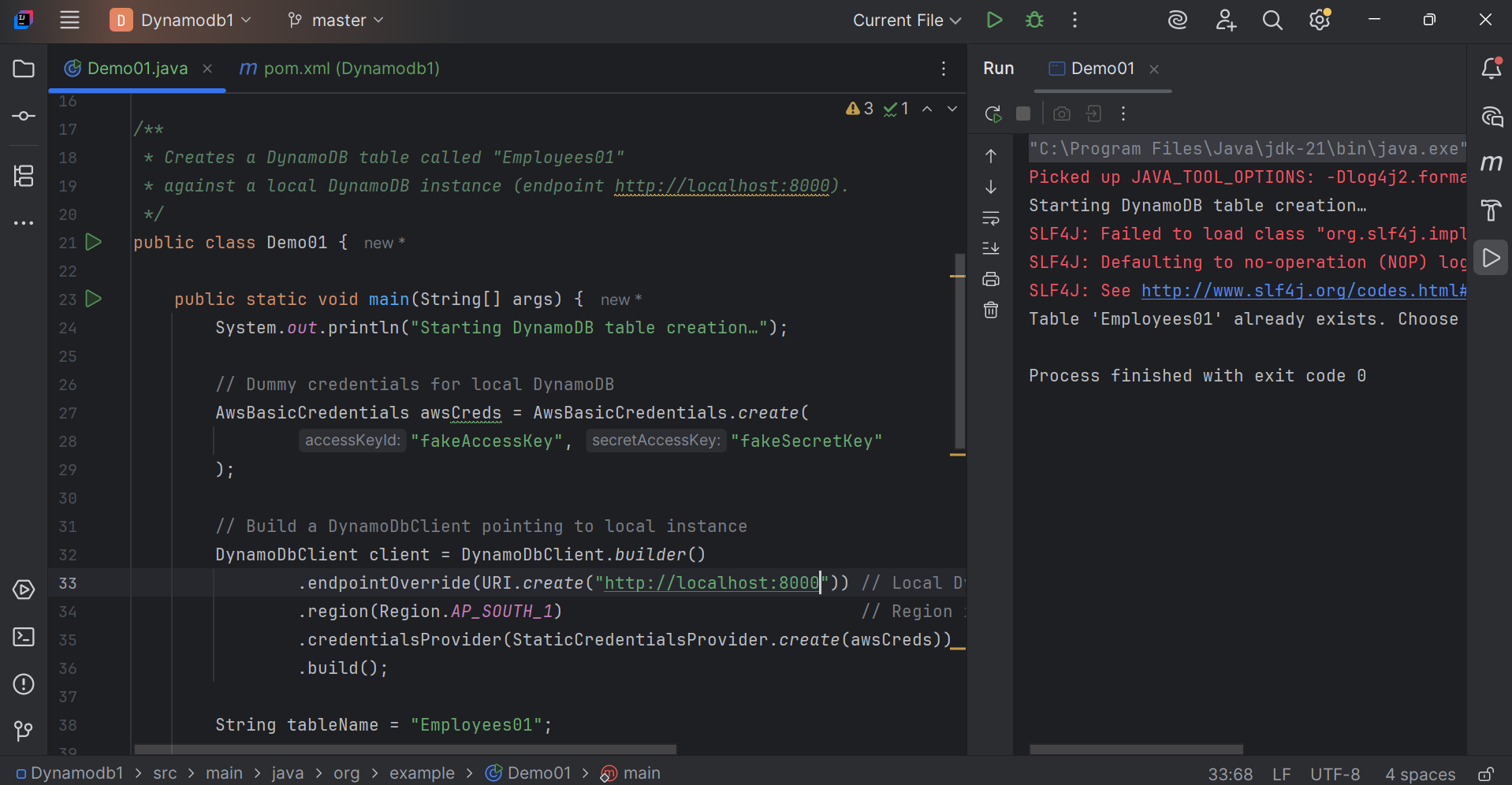
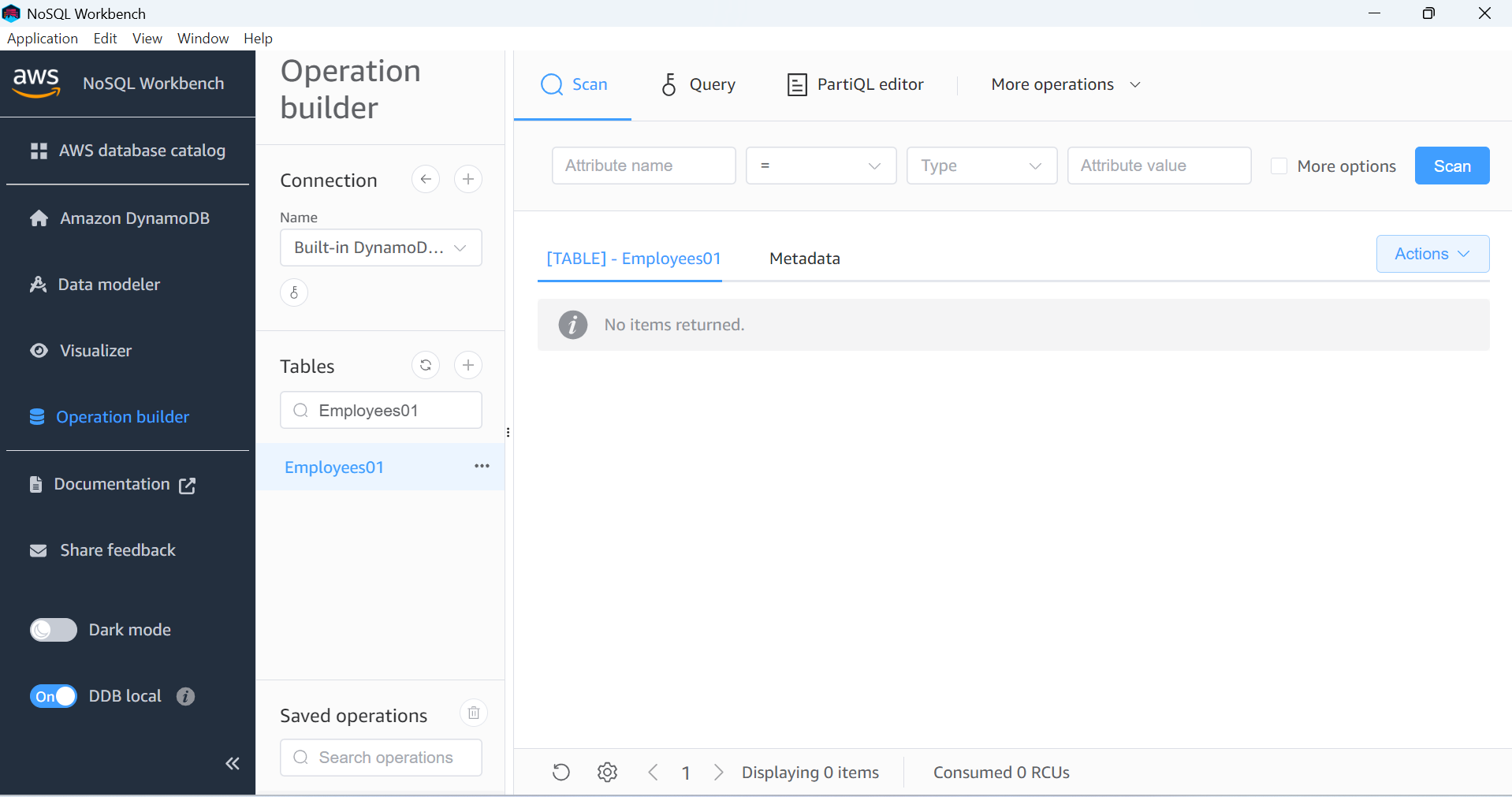
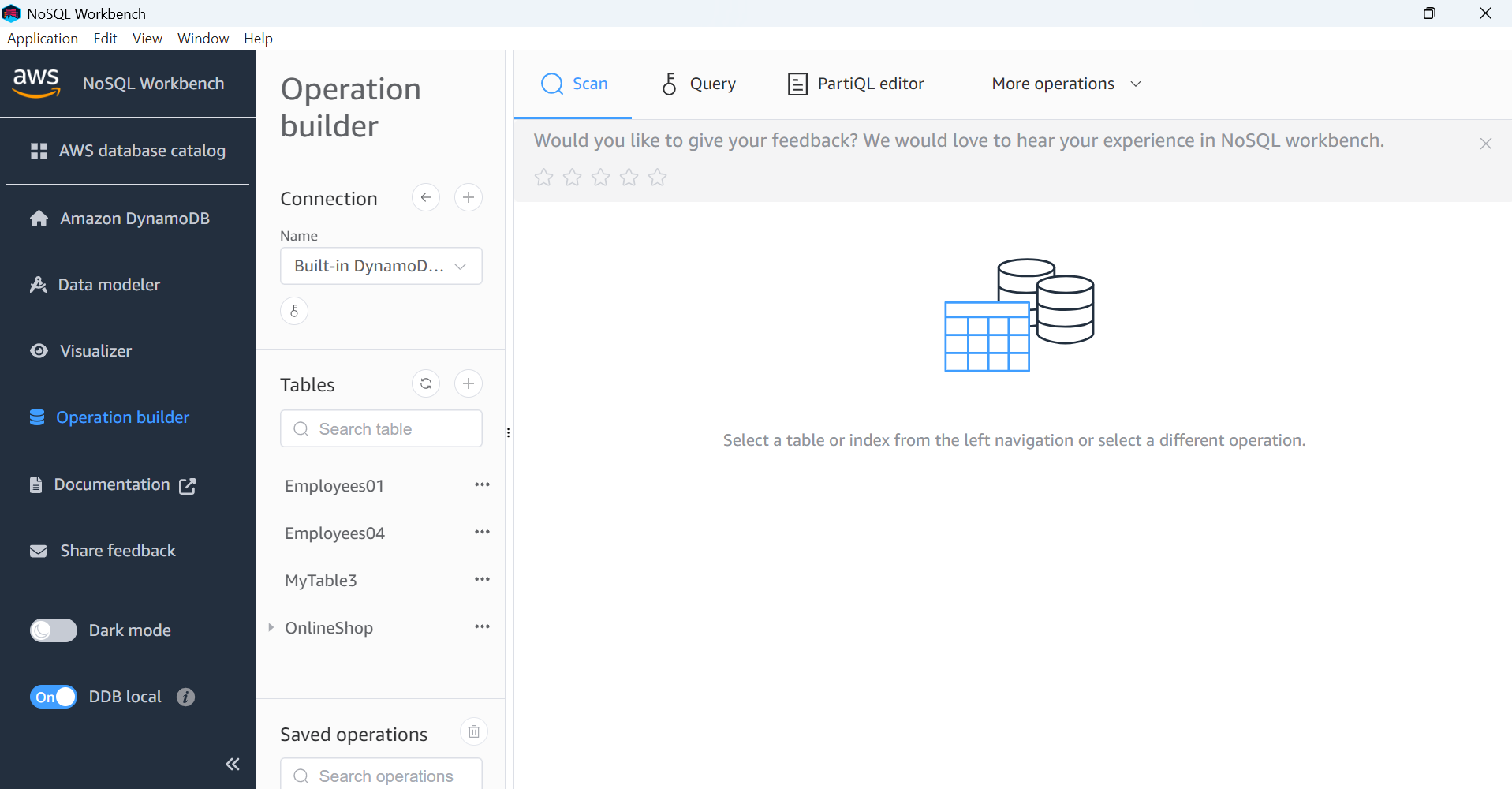
**Task 01:**

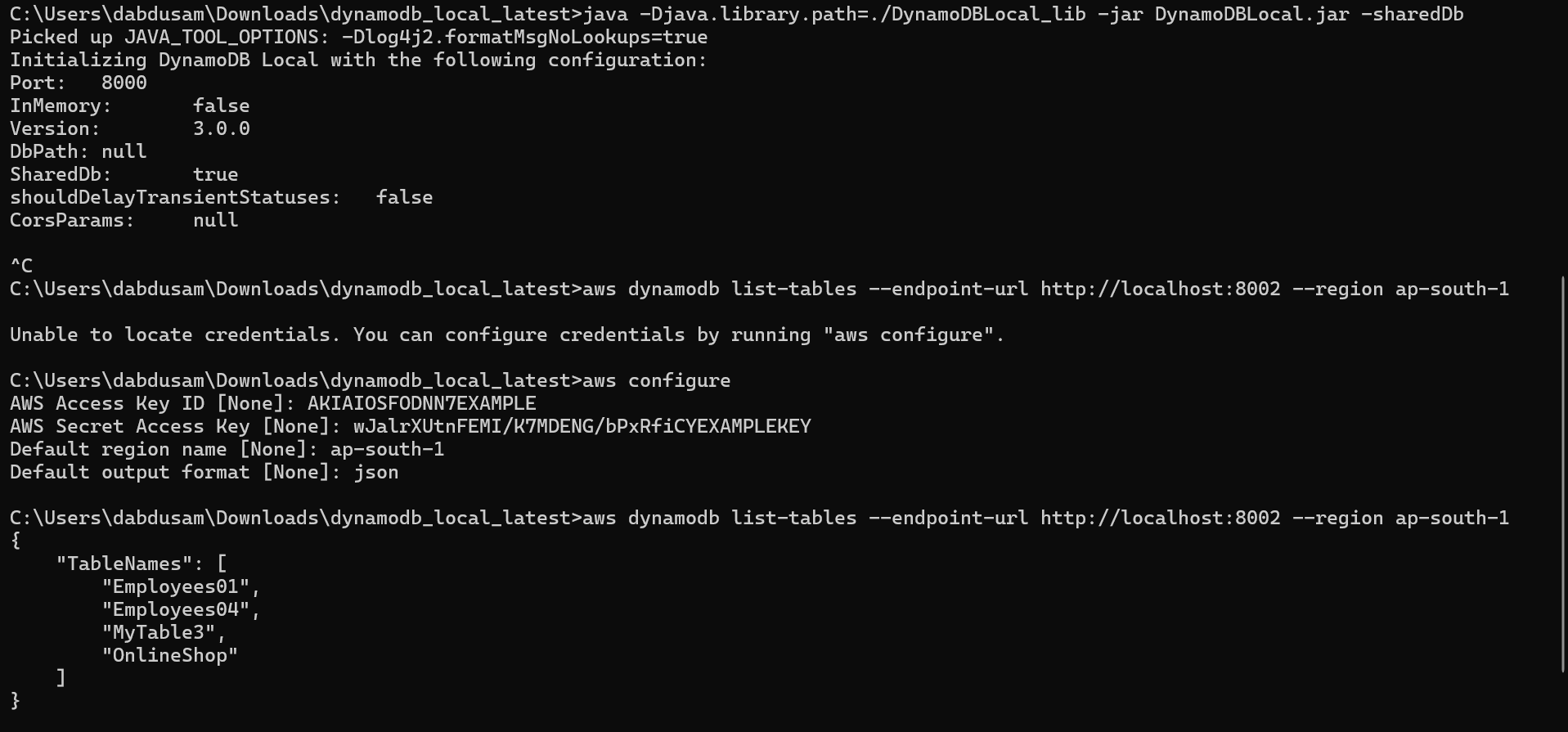




**Task -02:**

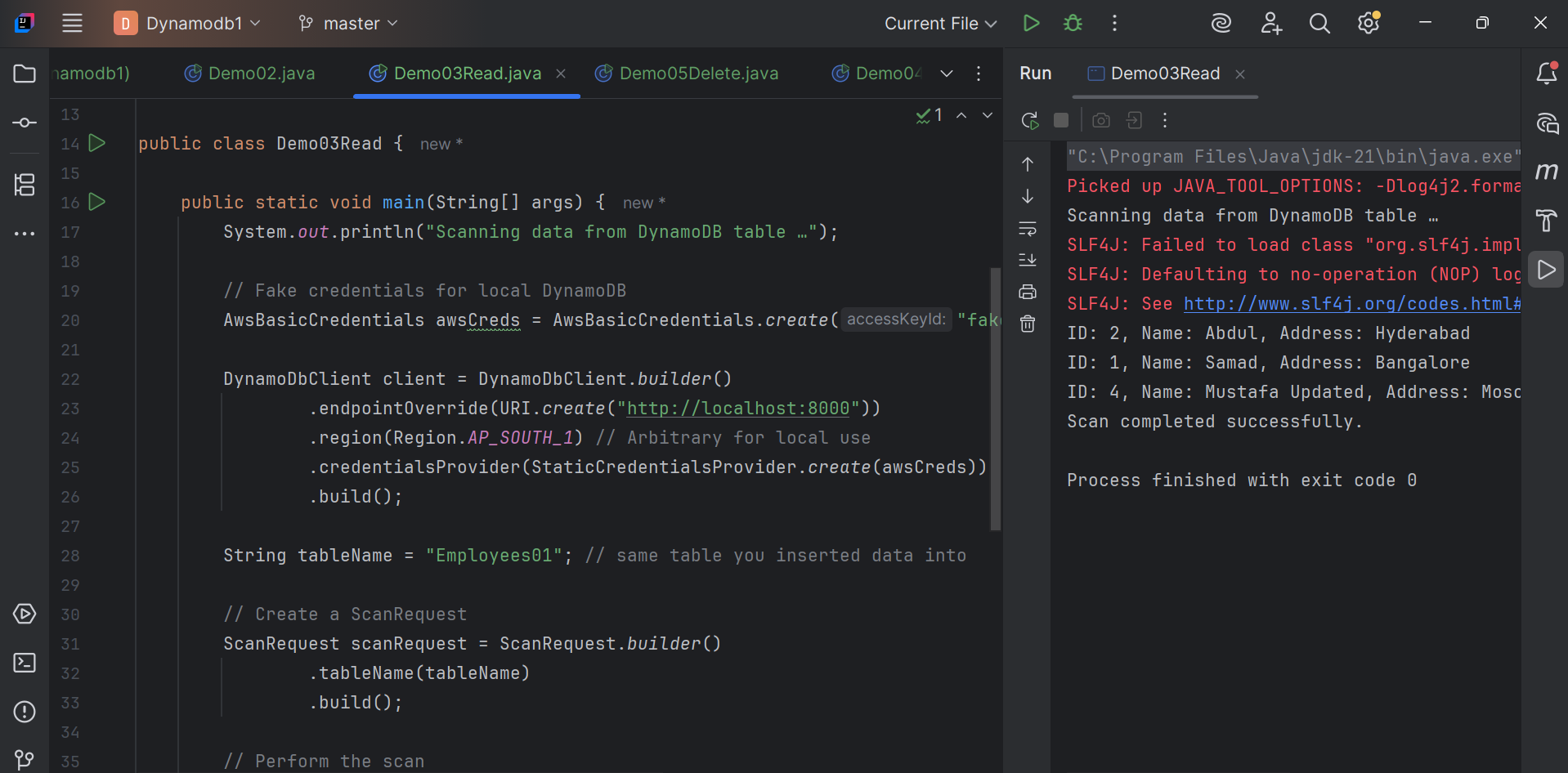


Task -03:



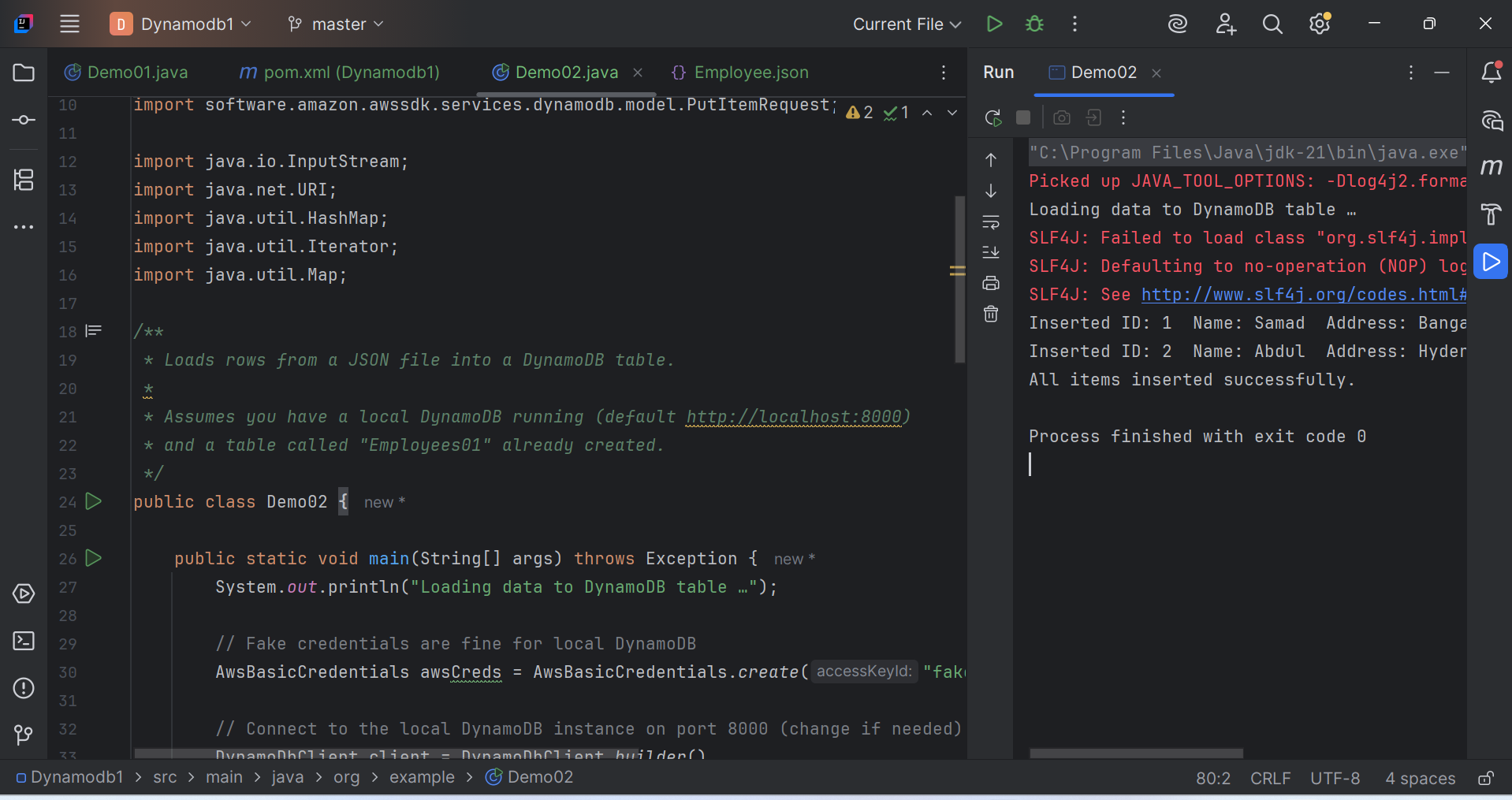
**Task -04**

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



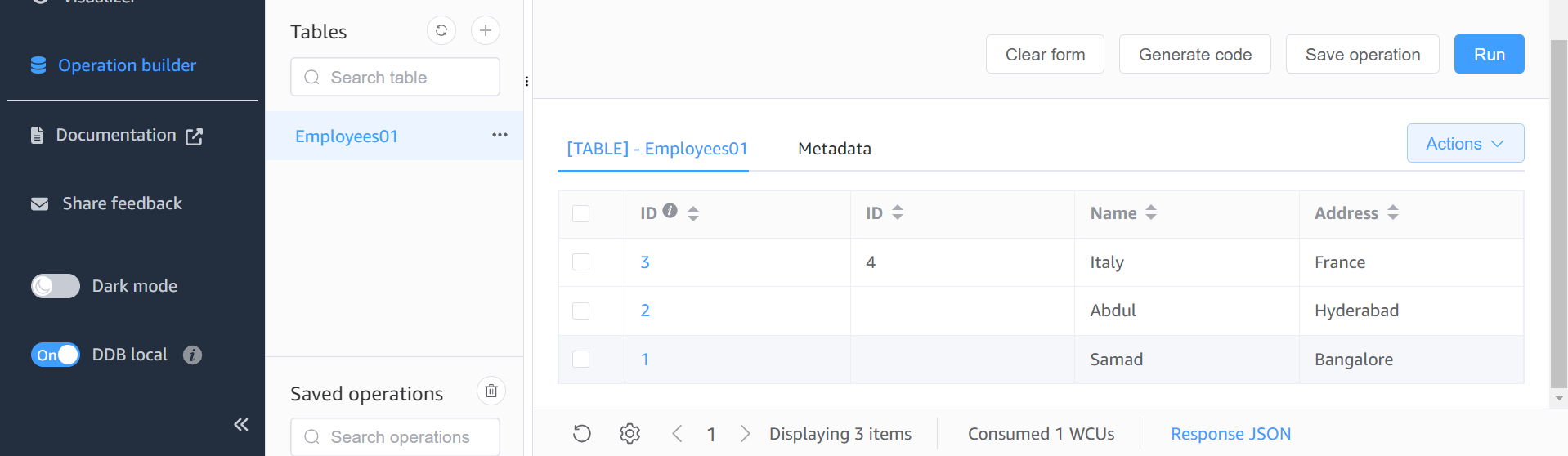
**Task -05**

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



**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…**

DynamoDB costs are mainly based on how much data you store, how many reads/writes you do, and optional features like streams or backups.

1. Choose the Right Capacity Mode

DynamoDB has two modes:

1. On-Demand Mode  
   * You pay for each read/write request.
   * Good if traffic is unpredictable or spiky.
   * Can get expensive if traffic is constant and high.
2. Provisioned Mode  
   * You specify how many reads/writes per second you need.
   * Cheaper if traffic is predictable.
   * You can enable Auto Scaling to adjust capacity automatically.

Tip: Use Provisioned + Auto Scaling for predictable workloads.

2. Reduce Read and Write Costs

* Use efficient reads/writes:  
  + Prefer batch operations (BatchWriteItem, BatchGetItem) instead of many single requests.
  + Avoid reading more data than needed; select only the attributes you need using ProjectionExpression.
* Use eventually consistent reads instead of strongly consistent reads if you don’t need instant updates.  
  + Eventually consistent reads are half the cost.

3. Optimize Data Storage

* Store only what you need: avoid storing large unused attributes.
* Use smaller data types: shorter strings, numbers instead of strings if possible.
* Remove old/unneeded items: use Time-to-Live (TTL) to automatically delete old data.

4. Reduce Index Costs

* Secondary indexes (GSI/LSI) increase cost because DynamoDB replicates data.
* Only create indexes you actually need.
* Avoid too many attributes in indexes.

5. Manage Streams, Backups, and Global Tables

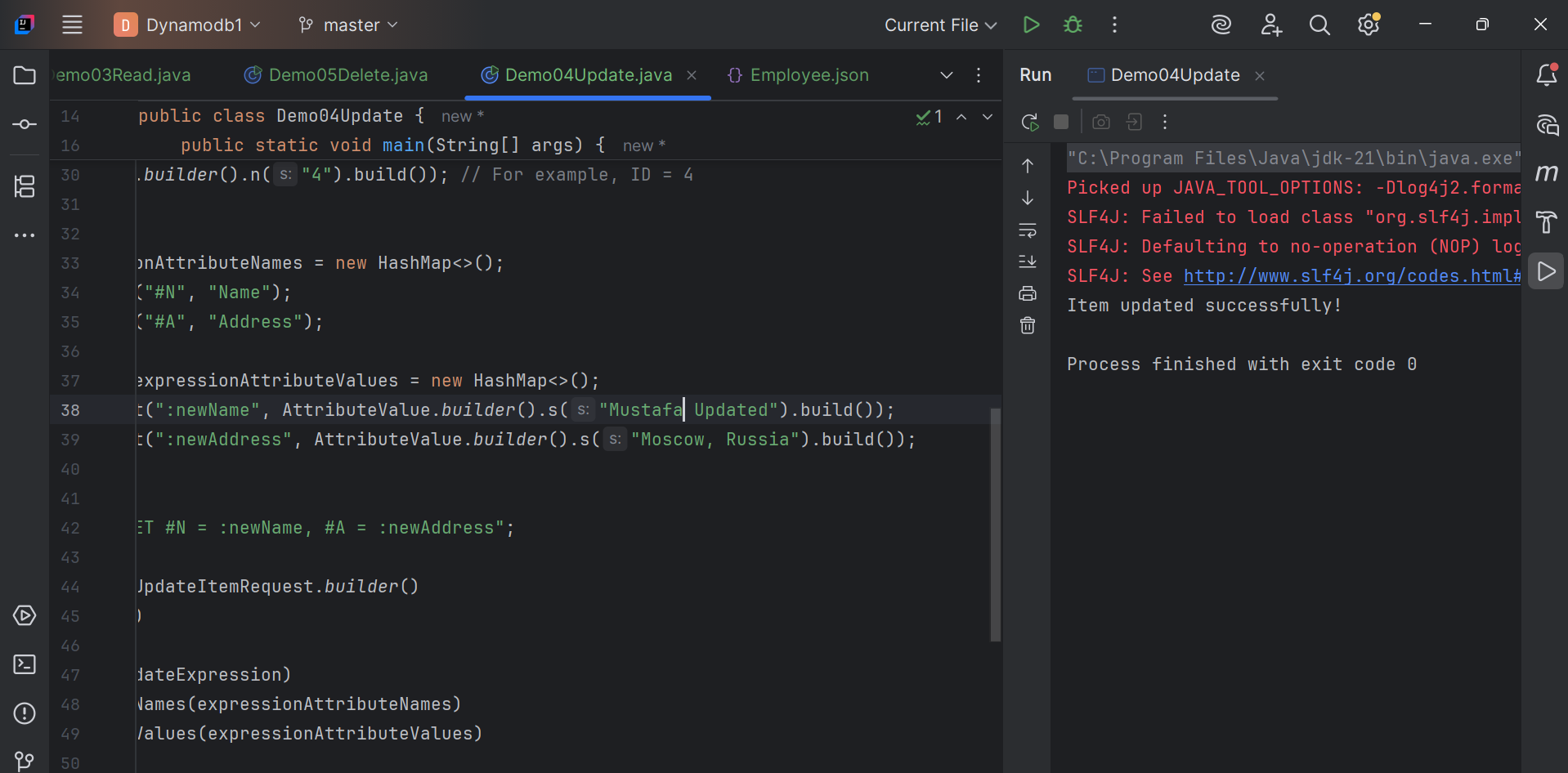
* DynamoDB Streams: only enable if needed.
* On-demand backups: create sparingly; scheduled backups can cost extra.
* Global tables: cost more due to replication across regions. Use only if you need multi-region writes.

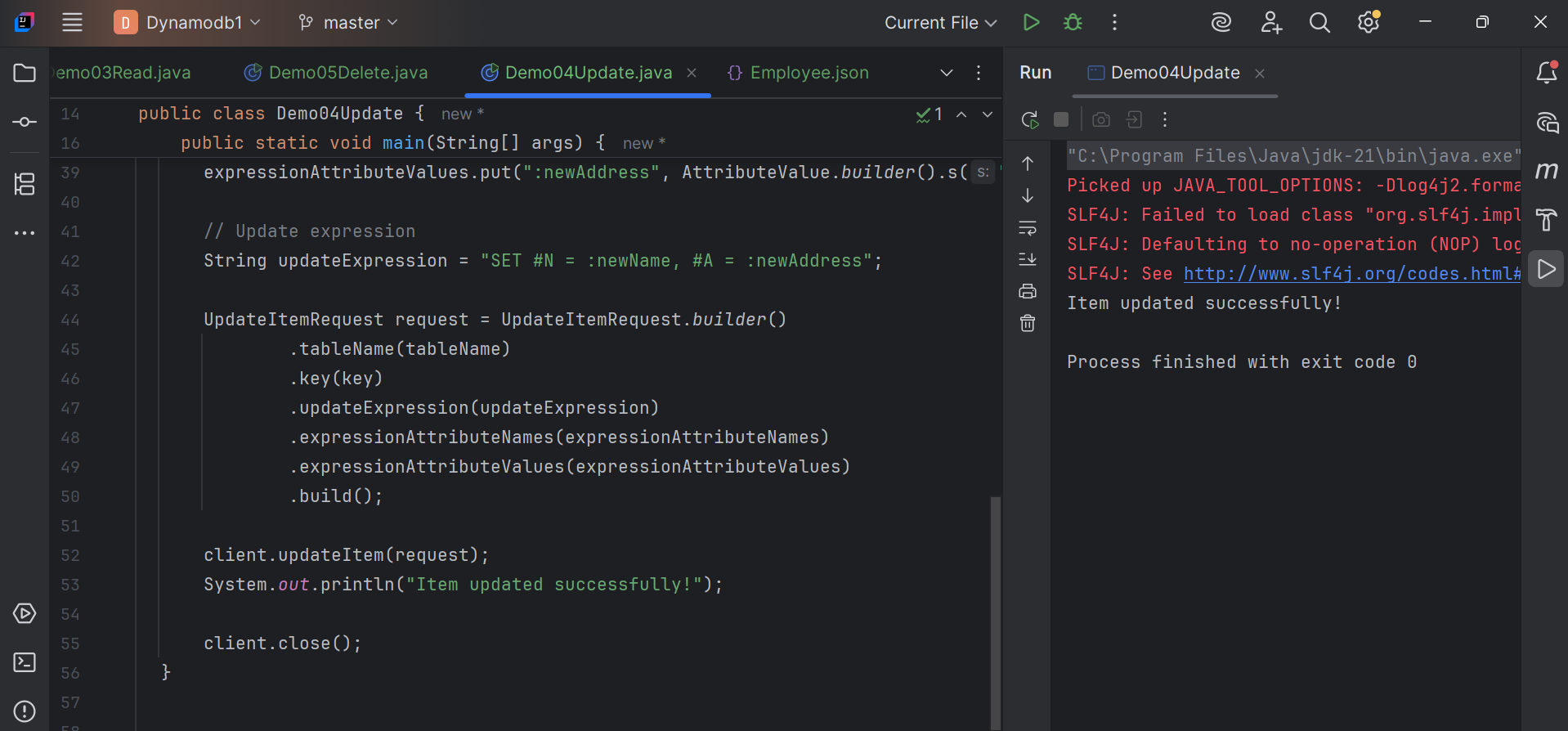
6. Monitor Usage

* Use AWS CloudWatch metrics:  
  + Check ConsumedReadCapacityUnits and ConsumedWriteCapacityUnits.
  + Adjust provisioned capacity or auto-scaling limits based on real usage.

**Task 08:**

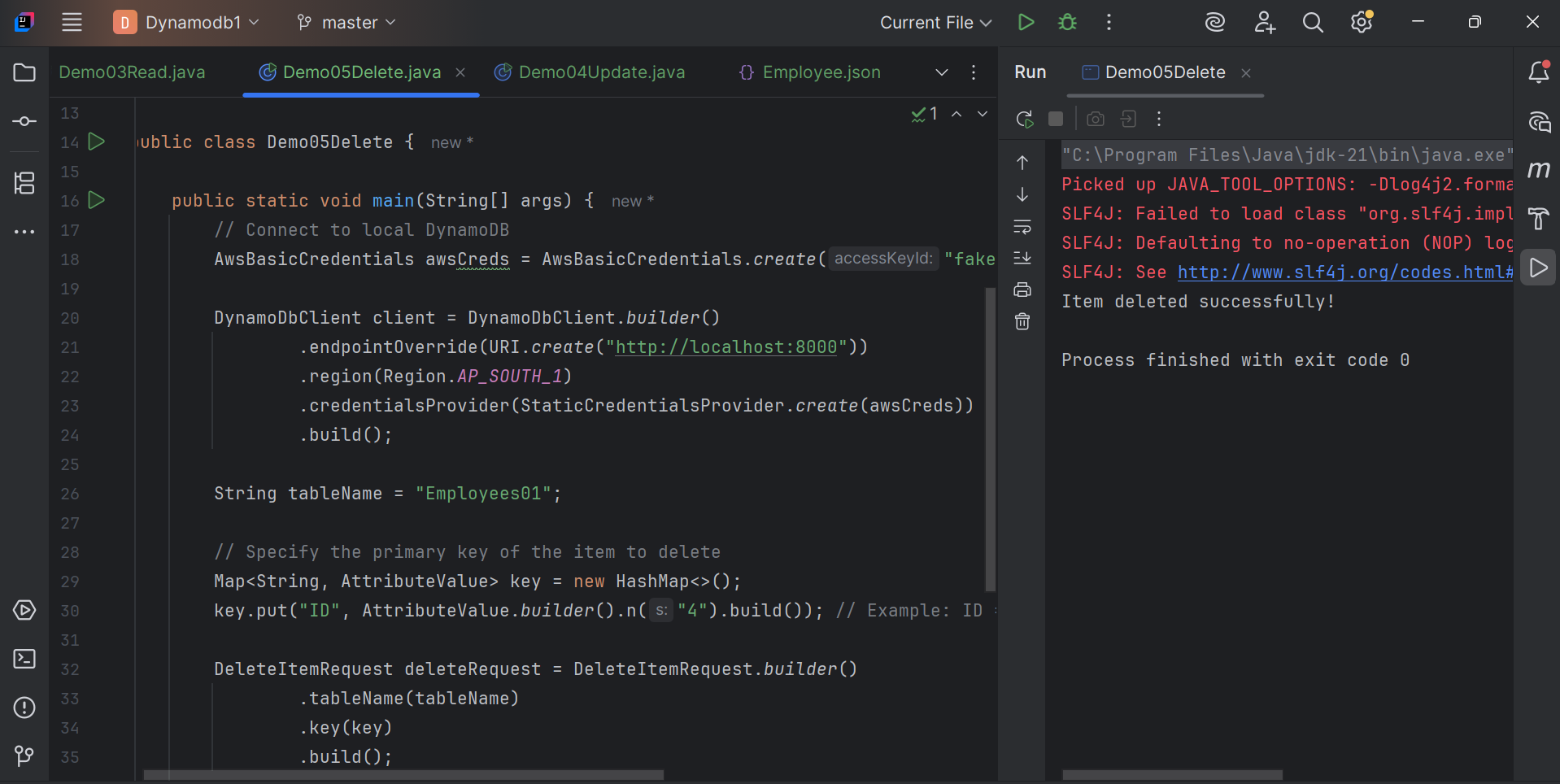
**Update item details using java code.. And check if it reflects in the server..**

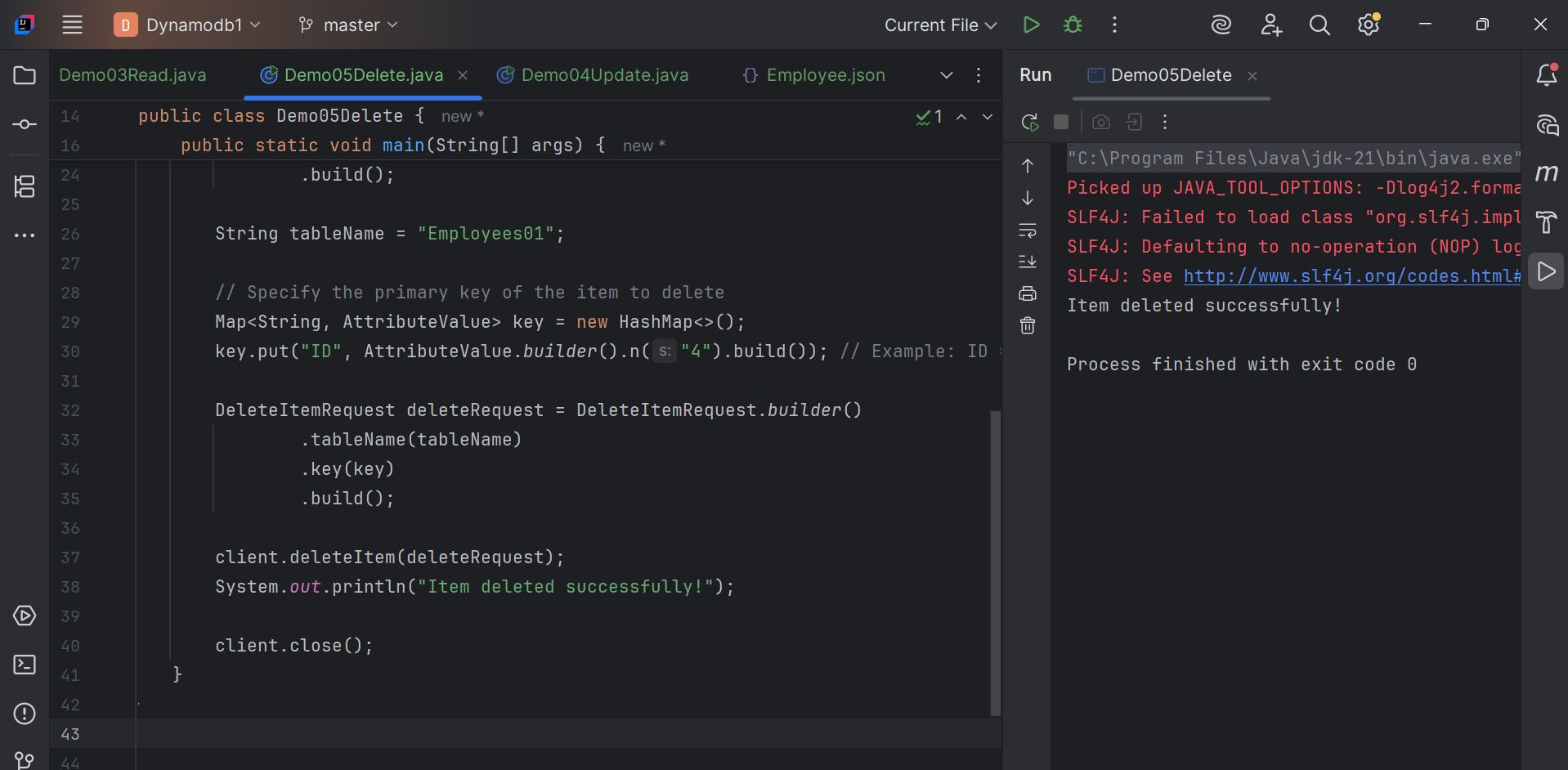




**Task 09:**

**Delete a particular item from the table …**

****

****

**Home task:**

**Task 01:**

**What are the features of DynamoDB?**

### Features of DynamoDB

Amazon DynamoDB is a fully managed NoSQL database that provides key-value and document storage, automatic scaling, and high availability across multiple AZs (Availability Zones).

1. It offers low-latency performance (single-digit milliseconds) and supports on-demand or provisioned capacity modes.

2. Built-in encryption, backups, TTL (time-to-live), streams, and global tables are included.

**Task 02:**

**What are the advantages and disadvantages of Dynamodb**

**Advantages:** Serverless (no admin), auto scaling, predictable low latency, flexible schema, seamless integration with other AWS services.

**Disadvantages:** Query patterns must be planned up-front, limited item size (400 KB), costs can grow with large reads/writes or many GSIs, and no multi-item transactions beyond what DynamoDB transactions allow.

**Task 03:**

**Where do we use dynamoDB(the uses cases of DynamoDb)**

Used for real-time applications like gaming leaderboards, IoT device data, e-commerce carts, mobile apps, session storage, and event logging.

Ideal when you need massive scale with low latency and flexible schemas.

**Task 04:**

**What is DynamoDBMapper?**

A high-level Java data-model API provided by the AWS SDK.

It lets you map Java classes to DynamoDB tables using annotations and perform CRUD operations with simple method calls instead of manual requests.

**Task 05:**

**What are projections in DynamoDB?**

Projections define which attributes are copied into a secondary index.

Types include: KEYS\_ONLY (just keys), INCLUDE (keys + chosen attributes), and ALL (all attributes).

They reduce storage and read costs by keeping only the needed data in the index.

**Task 06:**

**How can you say DynamoDB prevents Dataloss?**

All data is synchronously replicated across multiple Availability Zones in an AWS region.

You can also enable Point-in-Time Recovery (PITR) and on-demand backups.

These features protect against hardware failures and accidental deletes.

**Task 07:**

**What is in-place atomic update means ..? do you think your dynamoDb supports it?**

It means updating a single attribute directly on the server in one step, e.g., incrementing a counter without reading the whole item.

Yes the DynamoDB supports it through UpdateItem with atomic counters or conditional expressions.

**Task 08:**

**What are Streams in DynamoDb?**

DynamoDB Streams capture real-time change logs (insert, update, delete) of items.

We can process these events with AWS Lambda or other consumers for triggers, replication, or analytics.

**Task 09:**

**What are DynamoDB Pricing tiers?**

You pay for read/write capacity (provisioned or on-demand), data storage, backups/PITR, streams, and global table replication.

There is also a free tier (25 GB storage + limited reads/writes per month).

**Task 10:**

**Do you have any max limit for Item Size in DynamoDb? If so what is it?**

The maximum size of a single DynamoDB item (including attribute names and values) is 400 KB.

**Task 11:**

**At a max of how many GSI’s Global Secondary Indexes can you create in a table?**

A single DynamoDB table can have up to 20 Global Secondary Indexes (GSIs).

**Task 12:**

**What is DynamoDb Accelerator?**

DAX is a fully managed, in-memory cache for DynamoDB.  
 It provides microsecond response times for read-heavy workloads without changing your application logic.

**Task 13:**

**What are DynamoDB Global tables?**

Global Tables automatically replicate data across multiple AWS regions, enabling multi-region, multi-master writes and low-latency reads worldwide.

**Task 14:**

**What are indexes and Secondary indexes in DynamoDB?**

An index lets you query data efficiently beyond the primary key.

Secondary indexes (Global or Local) let you create alternative query keys with their own partition/sort keys and projections.

**Task 15:**

**What are Hot Keys and Hot Partitions?**

A hot key or hot partition happens when one partition key receives disproportionate traffic, causing throttling and higher latency.

Good partition key design and traffic distribution prevent this.

**Task 16:**

**What are Table level operations and Item level operations in DynamoDB?**

Table-level operations affect the whole table: create, delete, describe, update table settings (capacity, indexes).

Item-level operations work on specific records: PutItem, GetItem, UpdateItem, DeleteItem, Query, and Scan.