

# Cycle 08 AWS Homework

## PartiQL Editor in DynamoDB

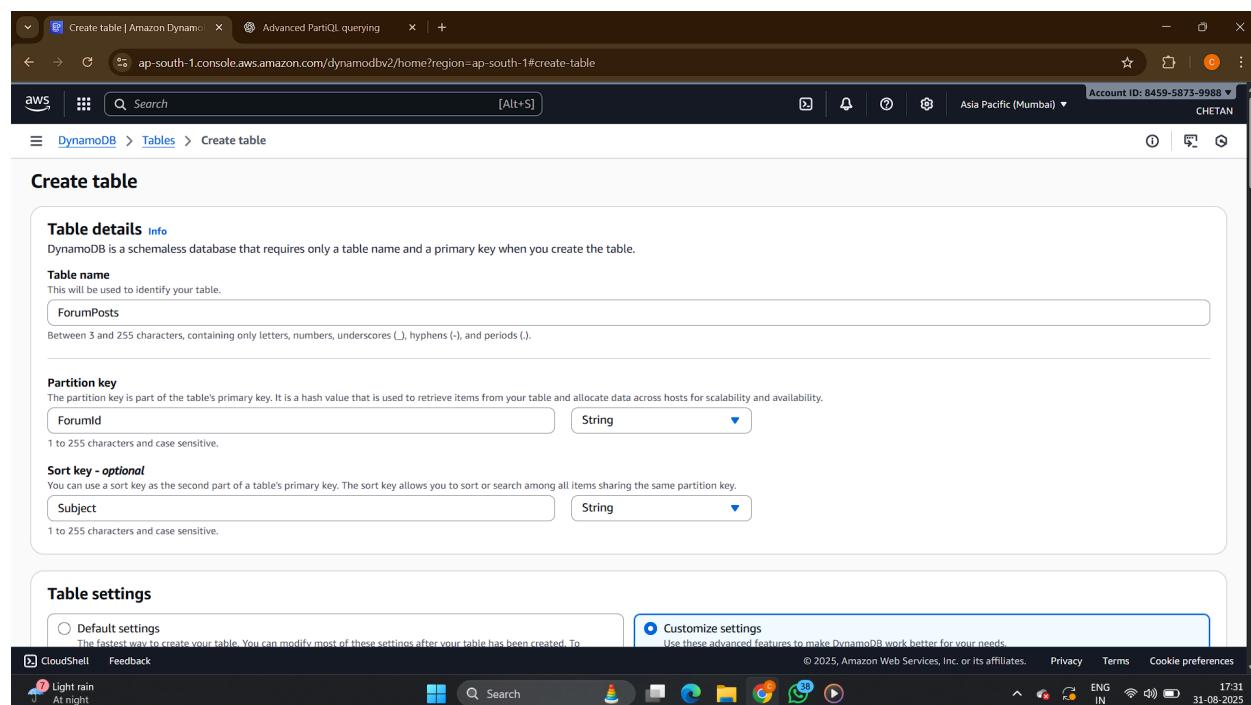
### Task 1: Basic PartiQL Operations

**Objective:** Practice CRUD (Create, Read, Update, Delete) operations using PartiQL on DynamoDB.

#### Step 1: Create a new table

- **Table Name:** Example: `ForumPosts`
- **Partition Key:** `ForumName` (String)
- **Sort Key:** `Subject` (String)

You can create this table either via the **DynamoDB Console > Tables > Create Table**, or use the AWS CLI or SDK. PartiQL itself doesn't create tables.



The screenshot shows the 'Capacity calculator' section of the 'Create table' wizard. It includes fields for Average item size (KB), Item read/second, Item write/second, Read consistency (Eventually consistent), Write consistency (Standard), Read capacity units (1), Write capacity units (1), Region (ap-south-1), and Estimated cost (\$0.67 / month).

**Read/write capacity settings**

**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.

The screenshot shows the 'Tables (1)' section of the 'List tables' page. A message indicates 'Creating the ForumPosts table. It will be available for use shortly.' The table list shows one entry: ForumPosts (Active, Partition key: ForumId, Sort key: Subject, Read capacity: Provisioned (1)).

**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

## Step 2: INSERT 5–7 new items

Using the PartiQL Editor:

```
INSERT INTO "ForumPosts" VALUE {'ForumName': 'DynamoDB Deep Dive', 'Subject': 'How to optimize reads', 'Author': 'Alice', 'Replies': 2};  
INSERT INTO "ForumPosts" VALUE {'ForumName': 'DynamoDB Deep Dive', 'Subject': 'How to use PartiQL', 'Author': 'Bob', 'Replies': 5};  
INSERT INTO "ForumPosts" VALUE {'ForumName': 'DynamoDB Tips', 'Subject': 'Secondary Indexes Explained', 'Author': 'Charlie', 'Replies': 1};  
INSERT INTO "ForumPosts" VALUE {'ForumName': 'DynamoDB Deep Dive', 'Subject': 'Table Design Best Practices', 'Author': 'Alice'};  
INSERT INTO "ForumPosts" VALUE {'ForumName': 'DynamoDB Basics', 'Subject': 'Getting Started Guide', 'Author': 'Eve', 'Replies': 0};
```

### Step 3: SELECT all items (SCAN)

```
SELECT * FROM "ForumPosts";
```

This performs a **full scan** and reads all items from the table.

### Step 4: SELECT where ForumName = 'DynamoDB Deep Dive' (Query)

```
SELECT * FROM "ForumPosts" WHERE ForumName = 'DynamoDB Deep Dive';
```

This is a **query operation** and is more efficient than a full scan because it uses the partition key.

### Step 5: UPDATE an item

```
UPDATE "ForumPosts"  
SET Replies = 5  
WHERE ForumName = 'DynamoDB Deep Dive' AND Subject = 'Table Design B'
```

est Practices';

This modifies an existing item by adding or changing the value of an attribute.

## Step 6: DELETE one item

DELETE FROM "ForumPosts"

WHERE ForumName = 'DynamoDB Basics' AND Subject = 'Getting Started Guide';

This removes the item with the specified primary key.

The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The 'Table Details' step is active. The 'Table name' field contains 'mytable'. The 'Partition key' section shows 'id' as the primary key of type String. The 'Sort key - optional' section shows 'email' as a secondary key of type String. Under 'Table settings', the 'Default settings' radio button is selected, with a note that these can be modified later. The bottom of the screen shows the Windows taskbar with various icons and system status.

The screenshot shows the AWS DynamoDB console. On the left, a sidebar menu includes 'Dashboard', 'Tables' (selected), 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Under 'Tables', there's a 'DAX' section with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area displays a table titled 'Tables (1) Info'. A message at the top says 'Creating the mytable table. It will be available for use shortly.' The table has columns: Name, Status, Partition key, Sort key, Indexes, Replication Regions, Deletion protection, Favorite, and Read capacity n. One row is shown for 'mytable', which is 'Creating', has 'id (\$)' as the partition key, 'email (\$)' as the sort key, and '0' for both indexes and replication regions. The 'Deletion protection' status is 'Off'. The 'Actions' dropdown shows 'Actions' and 'Delete', and there's a 'Create table' button.

The screenshot shows the AWS PartiQL editor interface. The URL in the browser bar is 'ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#edit-item?itemMode=1&route=ROUTE\_ITEM\_EXPLORER&ttable=mytable'. The main content area shows a success message: 'The item has been saved successfully.' Below it, a 'Create item' form is displayed. The 'Form' tab is selected. The 'Attributes' section contains two entries: 'id - Partition key' with value '102' and type 'String', and 'email - Sort key' with value 'dfsjkf@gmail.com' and type 'String'. At the bottom right are 'Cancel' and 'Create item' buttons. The bottom of the screen shows a standard Windows taskbar with icons for CloudShell, Feedback, Search, and other applications.

The screenshot shows the AWS DynamoDB Items page for the 'mytable' table. The table contains two items:

	id (String)	email (String)
1	102	dfsjkf@gmail.com
2	101	cgetan@gmail.com

The screenshot shows the AWS PartiQL editor interface. A query is being run against the 'mytable' table:

```
1 SELECT * FROM "mytable"
```

The screenshot shows the AWS DynamoDB PartiQL editor interface. On the left, a sidebar menu includes options like Dashboard, Tables, Explore items, PartiQL editor (which is selected), Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below this is a section for DAX with Clusters, Subnet groups, Parameter groups, and Events. The main content area displays a query execution status: "Completed" (green checkmark), "Started on 8/29/2025, 8:30:17 AM", and "Elapsed time 104ms". A "Run" button and a "Clear" button are at the top. Below this, a "Table view" tab is selected, showing a table with two rows of results:

id	email
102	dfsjkf@gmail.com
101	cgetan@gmail.com

A "Download results to CSV" button is located to the right of the table. At the bottom of the main area, there's a search bar with placeholder text "Find items" and navigation controls (back, forward, refresh). The bottom of the screen shows the Windows taskbar with icons for CloudShell, Feedback, Search, File Explorer, Task View, Task Manager, and Google Chrome.

The screenshot shows the AWS DynamoDB Explore items interface. The sidebar menu is identical to the previous screenshot. The main content area displays a "Sort key: email" configuration with dropdowns for "Equal to" and "Enter sort key value", and a checkbox for "Sort descending". Below this is a "Filters - optional" section with a "Run" and "Reset" button. A green message box indicates "Completed - Items returned: 3 - Items scanned: 3 - Efficiency: 100% - RCU consumed: 2".

Below this, a table titled "Table: mytable - Items returned (3)" shows the following data:

	id (String)	email (String)	name
103	rajesh@gmail.com	kumar	
102	dfsjkf@gmail.com	rajesh	
104	karthick1808@hotmail...		

At the bottom, there are "Actions" and "Create item" buttons. The bottom of the screen shows the Windows taskbar with icons for CloudShell, Feedback, Search, File Explorer, Task View, Task Manager, and Google Chrome.

The screenshot shows the AWS PartiQL editor interface. On the left, a sidebar titled 'DynamoDB' lists various options: Dashboard, Tables, Explore items, PartiQL editor (which is selected and highlighted in blue), Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below this, under 'DAX', are Clusters, Subnet groups, Parameter groups, and Events. The main workspace is titled 'PartiQL editor' and contains a message: 'Operations performed using the PartiQL editor might incur charges. Learn more'. It shows a table named 'mytable' listed under 'Tables (1)'. On the right, there are five tabs labeled 'Query 1' through 'Query 5', each with a green circular icon. The first tab's query is: '1 SELECT \* FROM "mytable"'. Below the queries are two buttons: 'Run' (orange) and 'Clear'. At the bottom, there are two tabs: 'Table view' (selected) and 'JSON view'. A status bar at the bottom indicates 'Completed'.

This screenshot shows the same AWS PartiQL editor interface after a query has been run. The 'Completed' status is displayed at the top. Below it, the message 'Started on 8/29/2025, 8:33:16 AM' and 'Elapsed time 35ms' are shown. The 'Table view' tab is selected, displaying the results of the query: '1 SELECT \* FROM "mytable"'. The results table has columns 'id' and 'email'. Two items are returned: item 102 with email 'dfsjkf@gmail.com' and item 101 with email 'cgetan@gmail.com'. There is a 'Download results to CSV' button on the right. The rest of the interface and sidebar are identical to the first screenshot.

The screenshot shows the AWS DynamoDB PartiQL editor interface. On the left, a sidebar menu includes 'Dashboard', 'Tables', 'Explore items', 'PartiQL editor' (which is selected), 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Below this is a 'DAX' section with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area is titled 'PartiQL editor' and contains a 'Tables (1)' section with a dropdown menu and a search bar. To the right, there are five tabs labeled 'Query 1', 'Query 2' (which is active), 'Query 3', 'Query 4', and 'Query 5'. The 'Query 2' tab contains the following SQL-like query:

```
1 SELECT * FROM "mytable" WHERE "id" = '102'
```

Below the tabs are 'Run' and 'Clear' buttons. Underneath the tabs, there are 'Table view' and 'JSON view' buttons, with 'Table view' currently selected. At the bottom of the editor, a status bar indicates 'Completed'.

This screenshot shows the same AWS DynamoDB PartiQL editor interface after a query has been run. The 'Completed' status is displayed at the bottom of the editor. The 'Elapsed time' is shown as 2423ms. The 'Items returned (1)' section displays the following table:

id	email
102	dfsjkf@gmail.com

A 'Download results to CSV' button is located to the right of the table. The rest of the interface is identical to the first screenshot, including the sidebar and the browser header.

The screenshot shows the AWS DynamoDB PartiQL editor interface. On the left sidebar, under the 'DynamoDB' section, 'PartiQL editor' is selected. The main area displays a 'Tables (1)' list with 'mytable'. Below it, five tabs are open: 'Query 1', 'Query 2', 'Query 3' (which is active), 'Query 4', and 'Query 5'. The 'Query 3' tab contains the following PartiQL code:

```
1 UPDATE "mytable"
2 SET "name" = 'rajesh'
3 WHERE "id" = '102' AND "email" = 'dfsjkf@gmail.com'
```

Below the code are 'Run' and 'Clear' buttons. The 'Table view' tab is selected. At the bottom, a green status bar indicates: 'Completed', 'Started on 8/29/2025, 8:40:05 AM', and 'Elapsed time 37ms'. A message box states: 'The command has been executed successfully.' The bottom navigation bar shows the AWS logo, search bar, and various browser icons.

This screenshot is identical to the one above, showing the same interface and PartiQL code. However, the message box at the bottom now states: 'No results' and 'The query did not return any results.' This indicates that the update operation did not return any items.

The screenshot shows the AWS DynamoDB PartiQL editor interface. On the left, the navigation menu is visible with options like Dashboard, Tables, Explore items, PartiQL editor (which is selected), Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below this is a section for DAX with Clusters, Subnet groups, Parameter groups, and Events. The main workspace is titled "PartiQL editor" and contains a "Tables (1)" section with a table named "mytable". To the right, there are five tabs labeled "Query 1" through "Query 5", with "Query 4" currently active. The query content is:

```
1 DELETE FROM "mytable" WHERE "id" = '101' and "email"='cgetan@gmail.com'
```

Below the queries are two buttons: "Run" (orange) and "Clear". Underneath the queries, there are two tabs: "Table view" (selected) and "JSON view". At the bottom of the editor, a status bar indicates "Completed". The browser's address bar shows the URL: ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#partiql-editor. The browser status bar at the bottom shows the date as 29-08-2025.

This screenshot is nearly identical to the one above, showing the AWS DynamoDB PartiQL editor. The main difference is in the "Completed" status bar at the bottom, which provides more detailed information about the executed command:

- Started on 8/29/2025, 8:53:07 AM
- Elapsed time 36ms
- The command has been executed successfully.

The rest of the interface, including the sidebar menu, tables, and query editor, remains the same.

The screenshot shows the AWS DynamoDB PartiQL editor interface. On the left, the navigation menu includes 'Dashboard', 'Tables', 'Explore items', 'PartiQL editor' (which is selected), 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Under 'PartiQL editor', there are sections for 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area displays a table named 'mytable' with one item. The PartiQL query window contains the following code:

```
1 insert into "mytable" value {"id": "104", "email": "karthick1808@hotmail.com"}
```

Below the query window are 'Run' and 'Clear' buttons, and tabs for 'Table view' (selected) and 'JSON view'. A status bar at the bottom indicates the command was completed successfully on 8/29/2025 at 8:54:18 AM with an elapsed time of 29ms.

This screenshot is identical to the one above, showing the successful execution of the insert query. The status bar now includes a green message: 'The command has been executed successfully.'

## Task 2: Advanced PartiQL Querying

### 1. Use **begins\_with** on Sort Key

```

SELECT * FROM "ForumPosts"
WHERE ForumName = 'DynamoDB Deep Dive' AND begins_with(Subject, 'Ho
w to');

```

This returns all items where the **Sort Key starts with "How to"**, within the same partition key (`ForumName`).

The screenshot shows the AWS DynamoDB console interface. On the left, there's a navigation sidebar with options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below that is a section for DAX with Clusters, Subnet groups, Parameter groups, and Events.

The main area is titled "Explore items > ForumPosts". It has a search bar at the top with "Scan" selected. Under "Select a table or index", "Table - ForumPosts" is chosen. Under "Select attribute projection", "All attributes" is selected. There are "Run" and "Reset" buttons below the filters. A green status bar at the bottom indicates "Completed - Items returned: 1 - Items scanned: 1 - Efficiency: 100% - RCU consumed: 0.5".

Below this, a table titled "Table: ForumPosts - Items returned (1)" is displayed. It shows one item with the following details:

	ForumId (String)	Subject (String)	Replies
<input type="checkbox"/>	general	how to play	10

At the bottom of the page, there are links for CloudShell, Feedback, and a sports headline about UEFA Champion... The footer includes copyright information, privacy terms, and a date of 31-08-2025.

The screenshot shows the Amazon DynamoDB PartiQL editor interface. On the left, a sidebar lists navigation options: Dashboard, Tables, Explore items, PartiQL editor (which is selected), Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below this is a section for DAX: Clusters, Subnet groups, Parameter groups, and Events. The main area is titled "PartiQL editor" and contains a "Tables (1)" section with a "Find tables" input field and a "ForumPosts" table selection. To the right is a "Query 1" panel with the following SQL-like code:

```
1 SELECT * FROM "ForumPosts"
2 WHERE "ForumId" = 'general'
3 AND begins_with("Subject", 'how to')
```

Below the code are "Run" and "Clear" buttons, and tabs for "Table view" (selected) and "JSON view". At the bottom of the editor, there's a "Completed" status message. The browser's address bar shows the URL: ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#partiql-editor. The status bar at the bottom indicates "CloudShell Feedback", "Rain warning In effect", and system information like "17:35 ENG IN 31-08-2025".

This screenshot shows the same Amazon DynamoDB PartiQL editor interface after a query has been run. The "Completed" status message now includes the start time ("Started on 8/31/2025, 5:35:45 PM") and elapsed time ("Elapsed time 46ms"). Below this, the "Items returned (1)" section displays the result of the previous query:

Replies	ForumId	Subject
10	general	how to play

There is also a "Download results to CSV" button. The rest of the interface and browser context remain the same as the first screenshot.

## 2. Filter on a non-key attribute (e.g., Replies > 3)

```
SELECT * FROM "ForumPosts" WHERE Replies > 3;
```

- This is a **scan with a filter**, not a query.
- Performance will be lower and it will consume more **read capacity units (RCUs)**.
- Only filters **after** scanning the full table.

The screenshot shows the AWS Lambda console with multiple tabs open: 'Items | Amazon DynamoDB' (active), 'PartiQL editor | Amazon DynamoDB', and 'Advanced PartiQL querying'. The URL in the address bar is `ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#item-explorer?table=ForumPosts`. The sidebar on the left shows 'DynamoDB' with 'Explore items' selected, and 'DAX' with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area displays a table titled 'Table: ForumPosts - Items returned (5)'. The table has columns: ForumId (String), Subject (String), and Replies. The data is as follows:

ForumId (String)	Subject (String)	Replies
general	where is	15
general	how to	10
general	how to play	10
general	where can i	21
privileged	what is the	56

A message at the top right says: 'Completed - Items returned: 3 - Items scanned: 3 - Efficiency: 100% - RCU consumed: 0.5'. The bottom of the screen shows the AWS navigation bar with CloudShell, Feedback, and various icons.

The screenshot shows the AWS DynamoDB console with the 'PartiQL editor' selected. On the left, the navigation menu includes 'Dashboard', 'Tables', 'Explore items', 'PartiQL editor' (which is highlighted in blue), 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Below this is a section for 'DAX' with options like 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area is titled 'PartiQL editor' and contains a 'Tables (1)' section with a search bar and a list of tables: 'ForumPosts'. To the right, there are two tabs: 'Query 1' and 'Query 2'. 'Query 1' is active and contains the following SQL-like code:

```

1 SELECT * FROM "ForumPosts"
2 WHERE "ForumId" = 'general'
3 AND "Replies" > 3
4

```

Below the code are 'Run' and 'Clear' buttons. Underneath the queries, there are 'Table view' and 'JSON view' tabs, with 'Table view' currently selected. A green 'Completed' status indicator is present. At the bottom of the editor, there's a toolbar with icons for CloudShell, Feedback, and other AWS services.

This screenshot is identical to the one above, but it shows the results of the query execution. The 'Completed' status is now green with a checkmark. Below it, the message 'Started on 8/31/2025, 6:13:09 PM' and 'Elapsed time 1180ms' are displayed. The 'Items returned (2)' section shows the following data:

Replies	ForumId	Subject
10	general	how to
15	general	where is

A 'Download results to CSV' button is located to the right of the table. The rest of the interface is the same as the first screenshot, including the sidebar and the bottom toolbar.

### 3. BatchExecuteStatement (Insert multiple items in one call)

- PartiQL supports batch execution via **BatchExecuteStatement API** in SDKs or CLI.

- Example (CLI or SDK required):

```
{
  "Statements": [
    {
      "Statement": "INSERT INTO ForumPosts VALUE {'ForumName': 'DynamoDB Advanced', 'Subject': 'Consistency Models', 'Author': 'Daniel'}"
    },
    {
      "Statement": "INSERT INTO ForumPosts VALUE {'ForumName': 'DynamoDB Advanced', 'Subject': 'Auto Scaling', 'Author': 'Emily'}"
    }
  ]
}
```

This reduces network calls and is more efficient than separate inserts.

```
PS C:\Users\amogh> aws dynamodb batch-execute-statement --statements file://batch.json
{
  "Responses": [
    {
      "TableName": "ForumPosts"
    },
    {
      "TableName": "ForumPosts"
    },
    {
      "TableName": "ForumPosts"
    }
  ]
}
```

```
PS C:\Users\amogh> aws dynamodb batch-execute-statement --statements file://batch.json
{
  "Responses": [
    {
      "TableName": "ForumPosts"
    },
    {
      "TableName": "ForumPosts"
    },
    {
      "TableName": "ForumPosts"
    }
  ]
}
```

## DynamoDB Streams

## Task 1: Simple Stream Processing with Lambda

**Objective:** Set up a Lambda function to listen to table changes and log them.

### Steps:

#### 1. Enable DynamoDB Streams:

- Go to your table > **Exports and streams** > Enable stream
- Choose **New and old images**

#### 2. Create a Lambda function (Python or Node.js)

Example (Python):

```
import json

def lambda_handler(event, context):
    for record in event['Records']:
        event_name = record['eventName']
        keys = record['dynamodb']['Keys']
        new_image = record['dynamodb'].get('NewImage', {})

        print(f"Event: {event_name}")
        print(f"Primary Key: {keys}")
        print(f"New Image: {json.dumps(new_image)}")
```

1. **Add trigger:** In the Lambda console, add your DynamoDB stream as a trigger.
2. **Test:** Go back to your table and perform INSERT, UPDATE, DELETE.
3. **Check CloudWatch Logs:** Confirm that the function logged the stream events.

**DynamoDB stream details**

**Stream status**: On

**Resource-based policy**: Not active

**Latest stream ARN**: arn:aws:dynamodb:ap-south-1:845958739988:table/ForumPosts/stream/2025-08-31T12:47:16.110

**Trigger (0)**

No triggers

**Create a trigger**

**AWS Lambda function details**

Use triggers to invoke an AWS Lambda function every time an item is changed, and then your DynamoDB stream is updated.

**Lambda function**

Choose an AWS Lambda function you want to trigger every time an item is changed.

Choose function

**Batch size**

The number of records Lambda reads at once before invoking the function.

1

Between 1 and 10,000.

Turn on trigger

Specify whether the trigger should be activated when you create it. You can turn it off later from the Lambda console.

**Create function** [Info](#)

Choose one of the following options to create your function.

- Author from scratch Start with a simple Hello World example.
- Use a blueprint Build a Lambda application from sample code and configuration presets for common use cases.
- Container image Select a container image to deploy for your function.

**Basic information** [Info](#)

**Blueprint name**

Process updates made to a DDB table

An Amazon DynamoDB trigger that logs the updates made to a table.

**Runtime**

python3.12

**Function name**

Enter a name that describes the purpose of your function.

myfunction

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

**Architecture**

x86\_64

**Execution role**

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences CAD/INR +0.70% ENG IN 18:22 31-08-2025

**Execution role**

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- Create a new role with basic Lambda permissions
- Use an existing role
- Create a new role from AWS policy templates

Role creation might take a few minutes. Do not delete the role or edit the trust or permissions policies in this role.

**Role name**

Enter a name for your new role.

DynamoDBLambdaRole

Use only letters, numbers, hyphens, or underscores with no spaces.

**Policy templates - optional** [Info](#)

Choose one or more policy templates.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences BAN - NED Game score ENG IN 18:23 31-08-2025

The screenshot shows the AWS Lambda 'Create function' page. A 'DynamoDB trigger' is being configured. The 'DynamoDB table' dropdown is set to 'arn:aws:dynamodb:ap-south-1:845958739988:table/ForumPosts'. The 'Batch size' field is set to 1. The 'Enable metrics' checkbox is unchecked. The 'Info' tab is selected. On the right, a 'Tutorials' sidebar is open, showing a 'Create a simple web app' tutorial.

The screenshot shows the AWS Lambda 'myfunction' function details page. A yellow warning box states: 'Your Lambda function "myfunction" was successfully created, but an error occurred when creating the trigger: Cannot access stream arn:aws:dynamodb:ap-south-1:845958739988:table/ForumPosts/stream/2025-08-31T12:47:16.110. Please ensure the role can perform the GetRecords, GetShardIterator, DescribeStream, and ListStreams Actions on your stream in IAM.' The 'Function overview' section shows the function name 'myfunction', a diagram, and a 'Description' field stating it's an Amazon DynamoDB trigger that logs updates made to a table. The 'Code' tab is selected at the bottom.

Screenshot of the AWS IAM Roles page:

**Search Bar:** Search IAM

**Left Sidebar:**

- Identity and Access Management (IAM)
- Dashboard
- Access management**
  - User groups
  - Users
  - Roles**
    - Policies
    - Identity providers
    - Account settings
    - Root access management
- Access reports**
  - Access Analyzer
  - Resource analysis
  - Unused access
  - Analyzer settings
  - Credential report

**Top Right:** Account ID: 8459-5873-9988, CHETAN

**Table Headers:** Role name, Trusted entities, Last activity

**Table Data:**

Role name	Trusted entities	Last activity
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application	3 days ago
AWSServiceRoleForDynamoDBReplication	AWS Service: replication.dynamodb	3 days ago
DynamoDBLambdaRole	AWS Service: lambda	-

**Bottom Section:**

- Roles Anywhere**: Info, Manage
- Access AWS from your non AWS workloads**: Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.
- X.509 Standard**: Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities.
- Temporary credentials**: Use temporary credentials with ease and benefit from the enhanced security they provide.

**Page Bottom:** https://us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/roles/details/Dyn... © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 27°C Mostly cloudy 18:26 31-08-2025

Screenshot of the AWS IAM Add permissions page for the DynamoDBLambdaRole:

**Search Bar:** Search

**Left Sidebar:** IAM > Roles > DynamoDBLambdaRole > Add permissions

**Section:** Attach policy to DynamoDBLambdaRole

**Current permissions policies:** (1) AdministratorAccess

**Other permissions policies:** (1/1078) Filter by Type: All types

**Table Headers:** Policy name, Type, Description

**Table Data:**

Policy name	Type	Description
AdministratorAccess	AWS managed - job function	Provides full access to AWS services an...
AdministratorAccess-Amplify	AWS managed	Grants account administrative permissi...
AdministratorAccess-AWSElasticBeanstalk	AWS managed	Grants account administrative permissi...
AIOpsConsoleAdminPolicy	AWS managed	Grants full access to Amazon AI Opera...
AmazonAPIGatewayAdministrator	AWS managed	Provides full access to create/edit/delete...
AmazonNimbleStudio-StudioAdmin	AWS managed	This policy grants access to Amazon Ni...
AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy	AWS managed	Service role policy used by the AWS Se...
AmazonSageMakerHyperPodObservabilityAdminAccess	AWS managed	This policy provides administrative pri...

**Page Bottom:** CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 27°C Mostly cloudy 18:27 31-08-2025

**myfunction**

**Function overview**

Description: An Amazon DynamoDB trigger that logs the updates made to a table.

Last modified: 3 minutes ago

Function ARN: arn:aws:lambda:ap-south-1:845958739988:function:myfunction

Function URL: -

**Code** | Test | Monitor | Configuration | Aliases | Versions

**Code source**

Open in Visual Studio Code | Upload from |

CloudShell | Feedback | 27°C Mostly cloudy | © 2025, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences | ENG IN | 18:27 | 31-08-2025

**Create a trigger**

**AWS Lambda function details**

Use triggers to invoke an AWS Lambda function every time an item is changed, and then your DynamoDB stream is updated.

**Lambda function**

Choose an AWS Lambda function you want to trigger every time an item is changed.

Search bar: myfunction | Create new |

**Batch size**

The number of records Lambda reads at once before invoking the function.

1 | Between 1 and 10,000.

Turn on trigger

Specify whether the trigger should be activated when you create it. You can turn it off later from the Lambda console.

**Create trigger**

CloudShell | Feedback | 27°C Mostly cloudy | © 2025, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences | ENG IN | 18:27 | 31-08-2025

The screenshot shows the AWS DynamoDB console. On the left, there's a navigation sidebar with 'Dashboard', 'Tables' (selected), 'Explore items', 'PartiQL editor', 'Backups', 'Exports to S3', 'Imports from S3', 'Integrations', 'Reserved capacity', and 'Settings'. Below that is a section for 'DAX' with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main content area shows a table named 'ForumPosts'. It has a 'Resource-based policy' section with a note that it's 'Not active'. A 'Trigger (1)' section shows a single trigger named 'myfunction' that is 'Enabled' and has 'No records processed'. Below this is an 'Amazon Kinesis data stream details' section with a 'Turn on' button. The bottom of the screen shows the AWS navigation bar with 'CloudShell', 'Feedback', 'Search', and various status icons.

The screenshot shows the AWS Lambda console. The top navigation bar includes 'View table | Amazon DynamoDB', 'myfunction | Functions | Lambda', 'DynamoDBLambdaRole | IAM', 'Advanced PartiQL querying', 'begins with in sql', and 'Account ID: 8459-5873-9988'. The main content area is titled 'Lambda > Functions > myfunction'. It features a 'CloudWatch metrics' section with six charts: 'Invocations', 'Duration', 'Error count and success rate', 'Throttles', 'Total concurrent executions', and 'Recursive invocations'. Each chart shows data over a 3-hour period. To the right of the charts is a 'Tutorials' sidebar with a 'Create a simple web app' section containing a list of steps: 'Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage', 'Invoke your function through its function URL', and a 'Learn more' link. The bottom of the screen shows the AWS navigation bar with 'CloudShell', 'Feedback', 'Search', and various status icons.

**myfunction**

**Function overview**

Description: An Amazon DynamoDB trigger that logs the updates made to a table.

Last modified: 5 minutes ago

Function ARN: arn:aws:lambda:ap-south-1:845958739988:myfunction

Function URL: -

Code | Test | **Monitor** | Configuration | Aliases | Versions

Filter metrics by Function | View CloudWatch logs

**Create a simple web app**

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Learn more | Start tutorial

**CloudWatch**

**/aws/lambda/myfunction**

**Log group details**

Log class: Info

ARN: arn:aws:logs:ap-south-1:845958739988:log-group:/aws/lambda/myfunction\*

Creation time: -

Retention: Never expire

Stored bytes: -

Account: -

Metric filters: Not supported

Subscription filters: Not supported

Contributor Insights rules: Not supported

KMS key ID: -

Anomaly detection: Not supported

Data protection: Not supported

Sensitive data count: Not supported

Custom field indexes: Not supported

Transformer: Not supported

**Log streams** | Tags | Anomaly detection | Metric filters | Subscription filters | Contributor Insights | Data protection | Field in >

The screenshot shows the AWS PartiQL editor interface. On the left, there's a sidebar with 'DynamoDB' selected, showing options like Dashboard, Tables, Explore items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, Reserved capacity, and Settings. Below that is a section for 'DAX' with Clusters, Subnet groups, Parameter groups, and Events. The main area has a title 'Tables (1)' and a search bar 'Find tables'. It lists a single table 'ForumPosts'. To the right, there are three tabs for queries: 'Query 1', 'Query 2', and 'Query 3'. The 'Query 3' tab is active, containing the following code:

```

1 INSERT INTO "ForumPosts" VALUE {'ForumId': 'general', 'Subject': 'how to create', 'Replies': 16}
2

```

Below the code are 'Run' and 'Clear' buttons. Underneath the queries, it says 'Table view' and 'JSON view', with 'Table view' being selected. It also shows a status message: 'Completed', 'Started on 8/31/2025, 6:41:51 PM', and 'Elapsed time 30ms'. At the bottom, there's a footer with CloudShell, Feedback, 26°C Mostly cloudy, and standard browser navigation controls.

The screenshot shows the AWS CloudWatch Log groups interface. On the left, there's a sidebar with 'CloudWatch' selected, showing Favorites and recents, Dashboards, AI Operations (with New), Alarms (0), Logs (with Log groups, Log Anomalies, Live Tail, Logs Insights, Contributor Insights), Metrics, Application Signals (with APM), and Network Monitoring. The main area shows a log group for 'arn:aws:logs:ap-south-1:845958739988:log-group:/aws/lambda/myfunction'. It displays details like Log class (Info, Standard), ARN (arn:aws:logs:ap-south-1:845958739988:log-group:/aws/lambda/myfunction:\*), Creation time (Now), Retention (Never expire), and Stored bytes (-). On the right, there are sections for Metric filters (0), Subscription filters (0), Contributor Insights rules (-), KMS key ID (-), Anomaly detection (Configure), Data protection (Sensitive data count, Custom field indexes, Transformer), and Field indexes. Below this, the 'Log streams (1)' section shows a single log stream: '2025/08/31/[\$LATEST]231553e1eff746e5beb2e53a6dca9cc6' with a timestamp of '2025-08-31 13:11:47 (UTC)'. The interface includes tabs for Tags, Anomaly detection, Metric filters, Subscription filters, Contributor Insights, Data protection, Field indexes, and Transfer. At the bottom, there's a footer with CloudShell, Feedback, 26°C Mostly cloudy, and standard browser navigation controls.

The screenshot shows the AWS CloudWatch Log Events interface. The left sidebar navigation includes 'CloudWatch' (selected), 'Favorites and recents', 'Dashboards', 'AI Operations', 'Alarms', 'Logs' (selected), 'Log groups' (selected), 'Log Anomalies', 'Live Tail', 'Logs Insights', 'Contributor Insights', 'Metrics', 'Application Signals (APM)', and 'Network Monitoring'. The main content area displays 'Log events' with a search bar and filter options (Clear, 1m, 30m, 1h, 12h, Custom, UTC timezone). A table lists log entries with columns for 'Timestamp' and 'Message'. The first entry is 'INIT\_START Runtime Version: python:3.12.v85 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:86ca4474369fee64939d2cf1cc596cb..'. Subsequent entries show function loading, start requests, and DynamoDB records. The bottom status bar shows 'CloudShell Feedback', weather '26°C Mostly cloudy', and system info '© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 18:42 31-08-2025'.

This screenshot is identical to the one above, showing the AWS CloudWatch Log Events interface for the same Lambda function. The left sidebar and main content area are identical, displaying log events and their timestamps. The bottom status bar also remains the same, showing 'CloudShell Feedback', weather '26°C Mostly cloudy', and system info '© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 18:42 31-08-2025'.

The screenshot shows the AWS CloudWatch Log events interface. The left sidebar navigation includes 'CloudWatch' (selected), 'Favorites and recents', 'Dashboards', 'AI Operations', 'Alarms', 'Logs' (selected), 'Log groups' (selected), 'Metrics', 'Application Signals (APM)', and 'Network Monitoring'. The main content area is titled 'Log events' and displays a table of log entries. The columns are 'Timestamp' and 'Message'. The table contains approximately 15 log entries from August 31, 2025, at 11:47:294Z. The 'Message' column shows JSON-like log data, including fields like 'Subject', 'S', 'SequenceNumber', 'SizeBytes', 'StreamViewType', and 'REPORT RequestId'. A message at the bottom states 'No newer events at this moment. Auto retry paused. Resume'. The top right of the interface has buttons for 'Actions', 'Start tailing', and 'Create metric filter'. The bottom right shows the AWS footer with links for 'CloudShell', 'Feedback', 'Privacy', 'Terms', 'Cookie preferences', and a date/time stamp '31-08-2025'.

## Task 2: Advanced Stream Use-Case

### 1. Create an SNS topic and subscribe email

- Go to **SNS Console** > Topics > Create topic
- Type: Standard
- Add a subscription to your email (confirm via email)

### 2. Modify Lambda to publish to SNS

Update Lambda function:

```
import boto3
import json
sns = boto3.client('sns')

SNS_TOPIC_ARN = 'arn:aws:sns:your-region:your-account-id:YourTopicName'
```

```

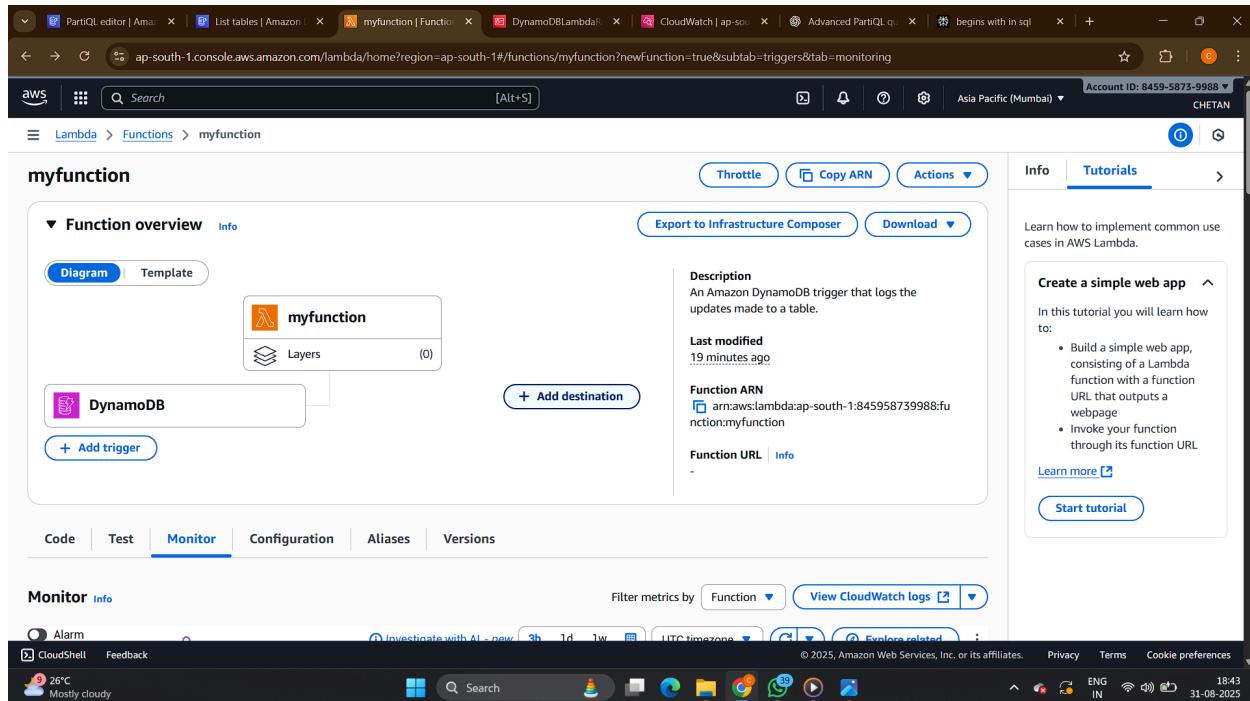
def lambda_handler(event, context):
    for record in event['Records']:
        if record['eventName'] == 'INSERT':
            new_item = record['dynamodb'].get('NewItem', {})
            subject = new_item.get('Subject', {}).get('S', '')
            forum = new_item.get('ForumName', {}).get('S', '')
            message = f"New post: {subject} in forum: {forum}"

            sns.publish(
                TopicArn=SNS_TOPIC_ARN,
                Message=message,
                Subject='New DynamoDB Post'
            )

```

### 3. Test

- Insert a new item into the table via PartiQL or Console.
- You should receive an email notification within seconds.



**Source**  
Choose the invocation type that Lambda sends records for.  
 Asynchronous invocation  
 Event source mapping invocation

**Condition**  
Choose whether to send invocation records for event processing failures or for successful invocations.  
 On failure  
 On success

**Destination type**  
Choose the destination type that Lambda sends invocation records to.  
 SNS topic

**Destination**  
Choose the ARN of the destination, or enter the ARN manually.  
 Use: "arn:aws:sns:ap-south-1:84595873998:dynamodbalert.fifo"  
 dynamodbalert.fifo

**PERMISSIONS**  
If your execution role doesn't have the required permissions for the selected destination, then Lambda will attempt to add the permissions to the role.

**Add required permissions**  
For more information about required permissions, see the [Destinations documentation](#).

**CloudShell Feedback** 26°C Mostly cloudy © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 18:51 31-08-2025

**Details**  
**Type** Standard  
 Topic type cannot be modified after topic is created

FIFO (first-in, first-out)  

- Strictly-preserved message ordering
- Exactly-once message delivery
- Subscription protocols: SQS

Standard  

- Best-effort message ordering
- At-least once message delivery
- Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

**Name**  
 dynamodbalert  
 Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (\_).

**Display name - optional**  
 alert  
 Maximum 100 characters.

**Encryption - optional**  
 Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

**CloudShell Feedback** 26°C Mostly cloudy © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences ENG IN 18:54 31-08-2025

The screenshot shows the AWS SNS Topics page. On the left, there's a navigation sidebar with 'Amazon SNS' at the top, followed by 'Dashboard', 'Topics' (which is selected and highlighted in blue), 'Subscriptions', and a 'Mobile' section containing 'Push notifications' and 'Text messaging (SMS)'. The main content area is titled 'Topics (1)' and contains a table with one row. The table has columns for 'Name', 'Type', and 'ARN'. The single entry is 'dynamodbalert', which is a 'Standard' type topic with the ARN: arn:aws:sns:ap-south-1:845958739988:dynamodbal... .

The screenshot shows the AWS Lambda Functions page. The URL in the browser bar is ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/myfunction?subtab=triggers&tab=monitoring. The left sidebar shows 'Lambda' and 'Functions' with 'myfunction' selected. The main area displays the 'Function overview' tab. It shows a diagram where 'myfunction' (Lambda function) triggers 'DynamoDB' and 'SNS'. Below the diagram, there are buttons for '+ Add trigger' and '+ Add destination'. To the right, there are sections for 'Description' (An Amazon DynamoDB trigger that logs the updates made to a table), 'Last modified' (32 minutes ago), 'Function ARN' (arn:aws:lambda:ap-south-1:845958739988:function:myfunction), and 'Function URL' (Info). A 'Tutorials' sidebar on the right provides instructions for creating a simple web app using Lambda.

Create item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. [Learn more](#)

Attribute name	Value	Type
ForumName - Partition key	SNS	String
Subject - Sort key	To send notifications	String
priority	High	String

[Cancel](#) [Create item](#)

High Priority Item Added

New item inserted with HIGH priority.

```
{
  "Priority": {
    "S": "High"
  },
  "ForumName": {
    "S": "SNS"
  },
  "Subject": {
    "S": "To send notifications"
  }
}
```

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:  
<https://sns.ap-south-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:ap-south-1:172446570120:New-ForumPosts-Topic:1946#03-fd7-4692-bb63-f1bea624bb00&Endpoi>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

Enable desktop notifications for Gmail. [OK](#) [No, thanks](#)

## 4. Bonus: Only notify if Priority = 'High'

Add condition in Lambda:

```
priority = new_item.get('Priority', {}).get('S', '')
if priority == 'High':
```

```
# publish to SNS
```

This ensures only important posts trigger notifications.

---

## **End of Day (EOD) Status Report Template**

### **Activities Completed:**

- Created DynamoDB table with composite key
- Inserted and queried items using PartiQL
- Used PartiQL for filtering and batch operations
- Enabled DynamoDB Streams and integrated with Lambda
- Processed stream records and sent SNS notifications

### **Learnings & Takeaways:**

- PartiQL makes DynamoDB querying more SQL-like
- Queries using keys are faster and cheaper than scans
- Streams allow real-time processing of changes
- Lambda + SNS enables serverless notifications
- BatchExecuteStatement can improve write efficiency

### **Challenges Faced:**

- Filter operations on non-key attributes are less efficient
- Stream event structure can be verbose and requires parsing
- SNS email delivery requires email confirmation before it works