Introduction to Amazon Aurora

Overview

This lab introduces you to Amazon Aurora and provides you with a basic understanding of how to use Aurora. You will follow the steps to create an Aurora instance and then connect to it.

Topics covered

After completing this lab, you will be able to:

- Create an Aurora instance
- Connect to a pre-created Amazon Elastic Compute Cloud (Amazon EC2) instance
- Configure the Amazon EC2 instance to connect to Aurora
- Query the Aurora instance

Duration

This lab requires approximately *40 minutes* to complete.

Icon key

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

- **Command:** A command that you must run.
- **Expected output:** A sample output that you can use to verify the output of a command or edited file.
- **Note:** A hint, tip, or important guidance.
- **CAUTION:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
- **Copy edit:** A time when copying a command, script, or other text to a text editor (to edit specific variables within it) might be easier than editing directly in the command line or terminal.
- **Task complete:** A conclusion or summary point in the lab.

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that you need to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that this lab describes.

Prerequisites

To successfully complete this lab, you should have some experience using the Linux operating system and have a basic understanding of structured query language (SQL).

Accessing the AWS Management Console

1. At the upper-right corner of these instructions, choose ▸ **Start Lab**.

   Note: If you get an **Access Denied** error, close the error box, and choose ▸ **Start Lab** again.

2. The lab status can be interpreted as follows:
   - A red circle next to AWS ● at the upper-left corner of this page indicates the lab has not been started.
   - A yellow circle next to AWS ● at the upper-left corner of this page indicates the lab is starting.
   - A green circle next to AWS ● at the upper-left corner of this page indicates the lab is ready.

2. The lab status can be interpreted as follows:
    - A red circle next to **AWS** at the upper-left corner of this page indicates the lab has not been started.
    - A yellow circle next to **AWS** at the upper-left corner of this page indicates the lab is starting.
    - A green circle next to **AWS** at the upper-left corner of this page indicates the lab is ready.

Wait for the lab to be ready before proceeding.

3. At the top of these instructions, choose the green circle next to **AWS**

    This option opens the AWS Management Console in a new browser tab. The system automatically sign you in.

    Note: If a new browser tab does not open, a banner or icon at the top of your browser indicates that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop-ups**.

4. Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you should be able to see both browser tabs at the same time so that you can follow the lab steps.

    Caution: Do not change the lab Region unless specifically instructed to do so.

    It takes a few minutes to provision the resources necessary to complete this lab.

Introducing the technologies

Amazon Aurora

Aurora is a fully managed, MySQL-compatible, relational database engine that combines the performance and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. It delivers up to five times the performance of MySQL without requiring changes to most of your existing applications that use MySQL databases.

Amazon Elastic Compute Cloud (Amazon EC2)

Amazon EC2 is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2 reduces the time required to provision new server instances to minutes, giving you the ability to quickly scale capacity, both up and down, as your computing requirements change.

Amazon Relational Database Service (Amazon RDS)

Amazon RDS makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing you up to focus on your applications and business. Amazon RDS provides you with six database engines to choose from, including Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL, and MariaDB.

Task 1: Create an Aurora instance

In this task, you create an Aurora database (DB) instance.

5. At the top of the AWS Management Console, in the search bar, search for and choose `RDS`.
6. In the left navigation menu, choose **Databases**.
7. Choose **Create database** and then configure the following options:
    - For **Choose a database creation method**, choose **Standard create**.
    - For **Engine type**, choose **Aurora (MySQL Compatible)**.
    - For **Engine version**, choose the version specified as the **default for major version 8.0**.
    - For **Templates**, choose **Dev/Test**.
8. In the **Settings** section, configure the following options:
    - For **DB cluster identifier**, enter `aurora`.
    - For **Master username**, enter `admin`.
    - For **Master password**, enter `admin123`.
    - For **Confirm password**, enter `admin123`.
9. In the **Instance configuration** section for the **DB instance class** section, choose **Burstable classes (includes t classes)**, and choose **db.t3.medium** from the dropdown list.
10. In the **Availability & durability** section for **Multi-AZ deployment**, choose **Don't create an Aurora Replica**.

    Note: Amazon RDS Multi-AZ deployments provide enhanced availability and durability for DB instances, making them a natural fit for production database workloads. When you provision a Multi-AZ DB instance, Amazon RDS automatically creates a primary DB instance and synchronously replicates the data to a standby instance in a different Availability Zone.

10. In the **Availability & durability** section for **Multi-AZ deployment**, choose **Don't create an Aurora Replica**.

    Note: Amazon RDS Multi-AZ deployments provide enhanced availability and durability for DB instances, making them a natural fit for production database workloads. When you provision a Multi-AZ DB instance, Amazon RDS automatically creates a primary DB instance and synchronously replicates the data to a standby instance in a different Availability Zone.

    Since this is a lab environment, you do not need to perform a multi-AZ deployment.

11. In the **Connectivity** section, configure the following options and leave any not mentioned with their default value:

```
* For **Virtual private cloud (VPC)**, choose **LabVPC**.
* For **Subnet group**, choose **dbsubnetgroup**.
* For **Public access**, select **No**.
* For **VPC security group**, select **Choose existing**.
* For **Existing VPC security groups**, remove the **default** security group.
* From the **Existing VPC security groups** dropdown list, choose
**DBSecurityGroup**.
```

    Note: Subnets are segments of a virtual private cloud (VPC) IP address range that you designate to group your resources based on security and operational needs. A DB subnet group is a collection of subnets (typically private) that you create in a VPC and that you then designate for your DB instances. With a DB subnet group, you can specify a particular VPC when creating DB instances using the command line interface (CLI) or application programming interface (API); if you use the console, you can select the VPC and subnets that you want to use.

    The *aurora* subnet group was created for you when you launched the lab using AWS CloudFormation.

    Consider: You can use the Amazon Virtual Private Cloud (Amazon VPC) service to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

12. In the **Monitoring** section, clear the check box for **Enable Enhanced monitoring**.
13. Expand ▸ **Additional configuration** section. For **Initial database name**, enter `world`
14. In the **Encryption** section, clear the check box for **Enable encryption**.

```
<i class="fas fa-sticky-note" style="color:#ff6633" aria-hidden="true"></i> **Note:**
You can encrypt your Amazon RDS instances and snapshots at rest by enabling the
encryption option for your RDS DB instance. Data that is encrypted at rest includes
the underlying storage for a DB instance, its automated backups, read replicas, and
snapshots.
```

15. In the **Maintenance** section, clear the check box for **Enable auto minor version upgrade**.
16. Scroll to the bottom of the screen, and then choose **Create database**.

```
<i class="fas fa-sticky-note" style="color:#ff6633" aria-hidden="true"></i> **Note:**
```

    - Your Aurora DB instance is in the process of launching and can take up to 5 minutes to launch. However, you can continue to the next task.
    - If you encounter the *Suggested add-ons for aurora* pop-up window, you can ignore it and choose **Close**.

    Once the database has completed creating, you should see a similar notification message:

    Successfully created database aurora.

Task complete: You have successfully created an Aurora instance

---

Task 2: Connect to an Amazon EC2 Linux instance

In this task, you log into to your Amazon EC2 Linux instance. This instance was launched for you when you started your lab using CloudFormation.

17. At the top of the AWS Management Console, in the search bar, search for and choose `EC2`.
18. In the left navigation menu, choose **Instances**.
19. Next to the instance labelled **Command Host**, select the check box, and then choose **Connect**.

```
<i class="fas fa-sticky-note" style="color:#ff6633" aria-hidden="true"></i> **Note:**
If you do not see the **Command Host** instance, the lab is possibly still being
provisioned, or you may be using another Region.
```

20. For **Connect to instance**, choose **Session Manager**.

20. For **Connect to instance**, choose **Session Manager**.
21. Choose **Connect** to open a terminal window.

```
<i class="fas fa-sticky-note" style="color:#ff6633" aria-hidden="true"></i> **Note:**
If the **Connect** button is not available, wait for a few minutes and try again.
```

Task complete: You have successfully connected to the Amazon EC2 instance named Command Host.

Task 3: Configure the Amazon EC2 Linux instance to connect to Aurora

In this task, you use the yum package manager to install the MariaDB client and then configure the Amazon EC2 Linux instance to connect to the Aurora database.

22. ⌨ **Command:** To install the MariaDB client, run the following command. The MariaDB client is what you use in later steps to connect to the Aurora instance that you just created.

```
sudo yum install mariadb -y
```

Expected output: Output has been truncated.

```
```sql
******************************
**** This is OUTPUT ONLY. ****
******************************

Install  1 Package

Total download size: 8.8 M
Installed size: 49 M
Downloading packages:
mariadb-5.5.68-1.amzn2.0.1.x86_64.rpm                    |  8.8 MB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
    Installing : 1:mariadb-5.5.68-1.amzn2.0.1.86_64       1/1
    Verifying  : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64      1/1

Installed:
mariadb.x86_64 1:5.5.68-1.amzn2.0.1

Complete!
```
```

23. Using a different browser tab, go back to the AWS Management Console and in the search bar, search for and choose `RDS`.
24. In the left navigation menu, choose **Databases**.
25. Wait for **aurora-instance-1** to display **Available**.
26. Choose **aurora**.
27. Choose the **Connectivity & security** tab, and in the **Endpoints** section, copy the **Endpoint name** for the **Writer instance** to your text editor.

    The endpoint should look similar to the following: *aurora.cluster-cabcdefghijklm.us-west-2.rds.amazonaws.com*.

    Note: An endpoint is represented as an Aurora specific URL that contains a host address and a port. The following types of endpoints are available from an Aurora DB cluster.

    - *Cluster endpoint*:
      A cluster endpoint for an Aurora DB cluster connects to the current primary DB instance for that DB cluster. This endpoint is the only one that can perform write operations such as DDL statements. Because of this, the cluster endpoint is the one that you connect to when you first set up a cluster or when your cluster contains only a single DB instance.

```
Each Aurora DB cluster has one cluster endpoint and one primary DB instance.

You use the cluster endpoint for all write operations on the DB cluster, including
inserts, updates, deletes, and DDL changes. You can also use the cluster endpoint
for read operations, such as queries.
```

EN-US

27. Choose the **Connectivity & security** tab, and in the **Endpoints** section, copy the **Endpoint name** for the **Writer instance** to your text editor.

    The endpoint should look similar to the following: *aurora.cluster-cabcdefghijklm.us-west-2.rds.amazonaws.com*.

    Note: An endpoint is represented as an Aurora specific URL that contains a host address and a port. The following types of endpoints are available from an Aurora DB cluster.

    - *Cluster endpoint:*
      A cluster endpoint for an Aurora DB cluster connects to the current primary DB instance for that DB cluster. This endpoint is the only one that can perform write operations such as DDL statements. Because of this, the cluster endpoint is the one that you connect to when you first set up a cluster or when your cluster contains only a single DB instance.

      ```
      Each Aurora DB cluster has one cluster endpoint and one primary DB instance.

      You use the cluster endpoint for all write operations on the DB cluster, including inserts, updates, deletes, and DDL changes. You can also use the cluster endpoint for read operations, such as queries.

      The cluster endpoint provides failover support for read/write connections to the DB cluster. If the current primary DB instance of a DB cluster fails, Aurora automatically fails over to a new primary DB instance. During a failover, the DB cluster continues to serve connection requests to the cluster endpoint from the new primary DB instance, with minimal interruption of service.

      The following example illustrates a cluster endpoint for an Aurora MySQL DB cluster.

      *mydbcluster.cluster-123456789012.us-west-2.rds.amazonaws.com:3306*
      ```

    - *Reader endpoint:*
      A reader endpoint for an Aurora DB cluster connects to one of the available Aurora replicas for that DB cluster. Each Aurora DB cluster has one reader endpoint. If there is more than one Aurora replica, the reader endpoint directs each connection request to one of the Aurora replicas.

      The reader endpoint provides load-balancing support for read-only connections to the DB cluster. Use the reader endpoint for read operations, such as queries. You can't use the reader endpoint for write operations.

      The DB cluster distributes connection requests to the reader endpoint among the available Aurora replicas. If the DB cluster contains only a primary DB instance, the reader endpoint serves connection requests from the primary DB instance. If one or more Aurora replicas are created for that DB cluster, subsequent connections to the reader endpoint are load balanced among the replicas.

      The following example represents a reader endpoint for an Aurora MySQL DB cluster.

      mydbcluster.cluster-ro-123456789012.us-west-2.rds.amazonaws.com:3306

    Next, log into the database.

28. **Copy edit:** In the following command, replace *<endpoint_goes_here>* with the endpoint that you copied to your text editor.

    ```
    ```bash
    mysql -u admin --password='admin123' -h <endpoint_goes_here>
    ```
    ```

    Your command should look similar to the following:

    > mysql -u admin --password='admin123' -h mydbcluster.cluster-123456789012.us-west-2.rds.amazonaws.com

    The MySQL Command-Line Client is a SQL shell which enables interaction with database engines. More information is available here.

Switch	Description
-u or --user	The MySQL username used to connect to a database instance.
-p or --password	The MySQL password used to connect to a database instance.
-h or --host	The host address of the database engine.

    Once the command is updated, copy the command to your clipboard.

29. **Command:** Return to the **Session Manager** browser tab that was used to connect to the **Command Host**. To connect to the Aurora instance, run the command you had copied in the previous step.

29. ⌨ **Command:** Return to the **Session Manager** browser tab that was used to connect to the **Command Host**. To connect to the Aurora instance, run the command you had copied in the previous step.

Expected output:

```plain
******************************
**** This is OUTPUT ONLY. ****
******************************
Welcome to the MariaDB monitor.  Commands end with ;or \g.
Your MySQL connection id is 173
Server version: 8.0.28 Source distribution
Copyright (c) 2000, 2018, Oracle, MariaDB CorporationAb and others.
Type 'help;' or '\h' for help. Type '\c' to clear thecurrent input statement.
MySQL [(none)]>
```

👍 **Task complete:** You have successfully configured the Amazon EC2 Linux instance to connect to Aurora.

Task 4: Create a table and insert and query records

In this task, you learn how to create a table in a database, load data, and run a query.

30. ⌨ **Command:** To list the available databases, run the following command.

```sql
SHOW DATABASES;
```

Expected output:

```text
******************************
**** This is OUTPUT ONLY. ****
******************************
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| world              |
+--------------------+
5 rows in set (0.02 sec)
MySQL [(none)]>
```

31. To switch to the **world** database that you created in Task 1 when you provisioned the Aurora instance, run the following command.

```sql
USE world;
```

Expected output:

```text
******************************
**** This is OUTPUT ONLY. ****
******************************
Database changed
MySQL [world]>
```

32. ⌨ **Command:** To create a new table in the **world** database, run the following command.

```sql
CREATE TABLE `country` (
`Code` CHAR(3) NOT NULL DEFAULT '',
`Name` CHAR(52) NOT NULL DEFAULT '',
`Continent` enum('Asia','Europe','North
America','Africa','Oceania','Antarctica','South America') NOT NULL DEFAULT 'Asia',
`Region` CHAR(26) NOT NULL DEFAULT '',
`SurfaceArea` FLOAT(10,2) NOT NULL DEFAULT '0.00',
`IndepYear` SMALLINT(6) DEFAULT NULL,
`Population` INT(11) NOT NULL DEFAULT '0',
`LifeExpectancy` FLOAT(3,1) DEFAULT NULL,
`GNP` FLOAT(10,2) DEFAULT NULL,
`GNPOld` FLOAT(10,2) DEFAULT NULL,
`LocalName` CHAR(45) NOT NULL DEFAULT '',
`GovernmentForm` CHAR(45) NOT NULL DEFAULT '',
`Capital` INT(11) DEFAULT NULL,
`Code2` CHAR(2) NOT NULL DEFAULT '',
PRIMARY KEY (`Code`)
);
```

✅ **Expected output:**

```plain
******************************
**** This is OUTPUT ONLY. ****
******************************

Query OK, 0 rows affected, 7 warnings (0.02 sec)

MySQL [world]>
```

33. ⌨ **Command:** To insert new records into the **country** table that you just created, run the following commands.

```sql
INSERT INTO `country` VALUES ('GAB','Gabon','Africa','Central
Africa',267668.00,1960,1226000,50.1,5493.00,5279.00,'Le Gabon','Republic',902,'GA');

INSERT INTO `country` VALUES ('IRL','Ireland','Europe','British
Islands',70273.00,1921,3775100,76.8,75921.00,73132.00,'Ireland/
Éire','Republic',1447,'IE');

INSERT INTO `country` VALUES ('THA','Thailand','Asia','Southeast
Asia',513115.00,1350,61399000,68.6,116416.00,153907.00,'Prathet Thai','Constitutional
Monarchy',3320,'TH');

INSERT INTO `country` VALUES ('CRI','Costa Rica','North America','Central
America',51100.00,1821,4023000,75.8,10226.00,9757.00,'Costa
Rica','Republic',584,'CR');

INSERT INTO `country` VALUES ('AUS','Australia','Oceania','Australia and New
Zealand',7741220.00,1901,18886000,79.8,351182.00,392911.00,'Australia','Constitutiona
l Monarchy, Federation',135,'AU');
```

✅ **Expected output:**

```plain
******************************
**** This is OUTPUT ONLY. ****
******************************
Query OK, 1 row affected (0.00 sec)
MySQL [world]>
```

34. ⌨ **Command:** To query the table, run the following **SELECT** statement.

```sql
SELECT * FROM country WHERE GNP > 35000 and Population > 10000000;
```

Expected output:

```plain
*******************************
**** This is OUTPUT ONLY. ****
*******************************
```

```
------+----------+----------+-------------------------+------------+----------+----------+--------------+---------- +----------+-------------+---
-------------------------------+--------+------+
  | Code | Name     | Continent | Region                 | SurfaceArea | IndepYear | Population | LifeExpectancy |
GNP      | GNPOld   | LocalName   | GovernmentForm                |Capital | Code2 |
  +------+----------+----------+-------------------------+------------+----------+----------+--------------+----------+----------+-------------
+-------------------------------+--------+------+
  | AUS  | Australia | Oceania   | Australia and New Zealand |  7741220.00 |     1901 |   18886000 |         79.8 |
351182.00 | 392911.00 | Australia    | Constitutional Monarchy,Federation |     135 | AU    |
  | THA  | Thailand  | Asia      | Southeast Asia          |   513115.00 |     1350 |   61399000 |         68.6 |
116416.00 | 153907.00 | Prathet Thai | ConstitutionalMonarchy          |    3320 | TH    |
  +------+----------+----------+-------------------------+------------+----------+----------+--------------+----------+----------+-------------
+-------------------------------+--------+------+
  2 rows in set (0.00 sec)
  MySQL [world]>
  ```
```

The query should return two records for Australia and Thailand.

👍 **Task complete:** You have successfully created a table named *country*, inserted data into the table, and queried records returning two results.

---

# Conclusion

You have now successfully:

- Created an Aurora instance.
- Connected to a pre-created Amazon EC2 instance.
- Configured the Amazon EC2 instance to connect to Aurora.
- Queried the Aurora instance.

# Lab complete 🎓

35. Choose ■ **End Lab** at the top of this page, and then select **Yes** to confirm that you want to end the lab.
36. An **Ended AWS Lab Successfully** message is briefly displayed indicating that the lab has ended.
```

Introduction to Amazon DynamoDB

Lab overview

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed database and supports both document and key-value data models. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, Internet of Things (IoT), and many other applications.

In this lab, you will create a table in DynamoDB to store information about a music library. You will query the music library and then delete the DynamoDB table.

Topics covered

In this lab, you will:

- Create an Amazon DynamoDB table
- Enter data into an Amazon DynamoDB table
- Query an Amazon DynamoDB table
- Delete an Amazon DynamoDB table

Duration

This lab requires approximately **35 minutes** to complete.

Accessing the AWS Management Console

1. At the upper-right corner of these instructions, choose ▸ **Start Lab**

   Troubleshooting tip: If you get an **Access Denied** error, close the error box, and choose ▸ **Start Lab** again.

2. The lab status can be interpreted as follows:
   - A red circle next to AWS ● at the upper-left corner of this page indicates the lab has not been started.
   - A yellow circle next to AWS ● at the upper-left corner of this page indicates the lab is starting.
   - A green circle next to AWS ● at the upper-left corner of this page indicates the lab is ready.

   Please wait for the lab to be ready before proceeding.

3. At the top of these instructions, choose the green circle next to AWS ●

   This option will open the AWS Management Console in a new browser tab. The system will automatically sign you in.

   Tip: If a new browser tab does not open, a banner or icon at the top of your browser will indicate that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop-ups**.

4. Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you should be able to see both browser tabs at the same time so that you can follow the lab steps.

   ⚠ **Do not change the lab Region unless specifically instructed to do so**.

Task 1: Create a new table

In this task, you create a new table named **Music** in DynamoDB. Each table requires a partition key (or a primary key) that is used to partition data across DynamoDB servers. A table can also have a sort key. The combination of a partition key and sort key uniquely identifies each item in a DynamoDB table.

5. In the AWS Management Console, choose the **Services** menu. Under **Database**, choose **DynamoDB**.
6. Choose **Create table**.
7. For the **Table name**, enter `Music`
8. For the **Partition key**, enter `Artist` and leave **String** selected in the dropdown list.
9. For **Sort key - *optional***, enter `Song` and leave **String** selected.

   Your table will use the default settings for indexes and provisioned capacity.

10. Scroll down, and choose **Create table**.

    The table will be created in less than 1 minute. Wait for the **Music** table to be **Active** before moving on to the next task.

Task 2: Add data

In this task, you will add data to the **Music** table. A *table* is a collection of data on a particular topic.

Each table contains multiple *items*. An item is a group of attributes that is uniquely identifiable among all of the other items. Items in DynamoDB are similar in many ways to rows in other database systems. In DynamoDB, there is no limit to the number of items you can store in a table.

Each item consists of one or more *attributes*. An attribute is a fundamental data element, something that does not need to be broken down any further. For example, an item in a **Music** table contains attributes such as song and artist. Attributes in DynamoDB are similar columns in other database systems, but each item (row) can have different attributes (columns).

When you write an item to a DynamoDB table, only the partition key and sort key (if used) are required. Other than these fields, the table does not require a schema. This means that you can add attributes to one item that may be different than the attributes for other items.

11. Choose the **Music** table.
12. Choose **Actions**, and then choose **Create item**.
13. For the **Artist** value, enter `Pink Floyd`
14. For the **Song** value, enter `Money`

    These are the only required attributes, but you now add additional attributes.
15. To create an additional attribute, choose **Add new attribute**.
16. In the dropdown list, select **String**.

    A new attribute row will be added.
17. For the new attribute, enter the following:
    - **FIELD**: `Album`
    - **VALUE**: `The Dark Side of the Moon`
18. To add another new attribute, choose **Add new attribute**.
19. In the dropdown list, choose **Number**.

    A new number attribute will be added.
20. For the new attribute, enter the following:
    - **FIELD**: `Year`
    - **VALUE**: `1973`
21. Choose **Create item**.

    The item has now been added to the **Music** table.
22. Similarly, to create a second item, use the following attributes:

Attribute Name	Attribute Type	Attribute Value
Artist	String	John Lennon
Song	String	Imagine
Album	String	Imagine
Year	Number	1971
Genre	String	Soft rock

    This item has an additional attribute called **Genre**. This is an example of each item being capable of having different attributes without having to pre-define a table schema.
23. To create a third item, use the following attributes:

Attribute Name	Attribute Type	Attribute Value
Artist	String	Psy
Song	String	Gangnam Style
Album	String	Psy 6 (Six Rules), Part 1
Year	Number	2011
LengthSeconds	Number	219

    Once again, this item has a new **LengthSeconds** attribute identifying the length of the song. This demonstrates the flexibility of a NoSQL database.

    There are also faster ways to load data into DynamoDB, such as using AWS Command Line Interface,

Once again, this item has a new **LengthSeconds** attribute identifying the length of the song. This demonstrates the flexibility of a NoSQL database.

There are also faster ways to load data into DynamoDB, such as using AWS Command Line Interface, programmatically loading data, or using one of the free tools available on the internet.

Task 3: Modify an existing item

You now notice that there is an error in your data. In this task, you will modify an existing item.

24. In the DynamoDB dashboard, under **Tables**, choose **Explore Items**.
25. Choose the ⦿ **Music** button.
26. Choose **Psy**.
27. Change the **Year** from **2011** to **2012**.
28. Choose **Save changes**.

    The item is now updated.

Task 4: Query the table

There are two ways to query a DynamoDB table: *query* and *scan*.

A query operation finds items based on the primary key and optionally the sort key. It is fully indexed, so it runs very fast.

29. Expand **Scan/Query items**, and choose **Query**.

    Fields for the Artist (Partition key) and Song (Sort key) are now displayed.
30. Enter the following details:
    - **Artist (Partition key):** `Psy`
    - **Song (Sort key):** `Gangnam Style`
31. Choose **Run**.

    The song quickly appears in the list. You might need to scroll down to see this result.

    A query is the most efficient way to retrieve data from a DynamoDB table.

    Alternatively, you can scan for an item. This option involves looking through every item in a table, so it is less efficient and can take significant time for larger tables.
32. Scroll up on the page, and choose **Scan**.
33. Expand **Filters**, and enter the following values:
    - For **Attribute name**, enter `Year`
    - For **Type**, choose **Number**.
    - For **Value**, enter `1971`
34. Choose **Run**

    Only the song released in 1971 is displayed.

Task 5: Delete the table

In this task, you will delete the **Music** table, which will also delete all the data in the table.

35. In the DynamoDB dashboard, under **Tables**, choose **Update settings**.
36. Choose the **Music** table if it is not already selected.
37. Choose **Actions**, and then choose **Delete table**.
38. On the confirmation panel, enter `delete` and choose **Delete table**.

    The table will be deleted.

Conclusion

👍 Congratulations! You now have successfully:

- Created an Amazon DynamoDB table
- Entered data into an Amazon DynamoDB table
- Queried an Amazon DynamoDB table
- Deleted an Amazon DynamoDB table

Task 4: Query the table

There are two ways to query a DynamoDB table: *query* and *scan*.

A query operation finds items based on the primary key and optionally the sort key. It is fully indexed, so it runs very fast.

29. Expand **Scan/Query items**, and choose **Query**.

    Fields for the Artist (Partition key) and Song (Sort key) are now displayed.

30. Enter the following details:

    - **Artist (Partition key):** `Psy`
    - **Song (Sort key):** `Gangnam Style`

31. Choose **Run**.

    The song quickly appears in the list. You might need to scroll down to see this result.

    A query is the most efficient way to retrieve data from a DynamoDB table.

    Alternatively, you can scan for an item. This option involves looking through every item in a table, so it is less efficient and can take significant time for larger tables.

32. Scroll up on the page, and choose **Scan**.

33. Expand **Filters**, and enter the following values:

    - For **Attribute name**, enter `Year`
    - For **Type**, choose **Number**.
    - For **Value**, enter `1971`

34. Choose **Run**

    Only the song released in 1971 is displayed.

Task 5: Delete the table

In this task, you will delete the **Music** table, which will also delete all the data in the table.

35. In the DynamoDB dashboard, under **Tables**, choose **Update settings**.
36. Choose the **Music** table if it is not already selected.
37. Choose **Actions**, and then choose **Delete table**.
38. On the confirmation panel, enter `delete` and choose **Delete table**.

    The table will be deleted.

Conclusion

Congratulations! You now have successfully:

- Created an Amazon DynamoDB table
- Entered data into an Amazon DynamoDB table
- Queried an Amazon DynamoDB table
- Deleted an Amazon DynamoDB table

For more information about DynamoDB, see the DynamoDB documentation.

Lab complete

39. Choose ■ **End Lab** at the top of this page, and then select **Yes** to confirm that you want to end the lab.
40. An **Ended AWS Lab Successfully** message is briefly displayed indicating that the lab has ended.

For more information about AWS Training and Certification, see AWS Training and Certification.

Your feedback is welcome and appreciated.
If you would like to share any suggestions or corrections, please provide the details in our AWS Training and Certification Contact Form.