PRADYOTH SINGENAHALLI PRABHU

02071847

[psingenahalliprabhu@umassd.edu](mailto:psingenahalliprabhu@umassd.edu)

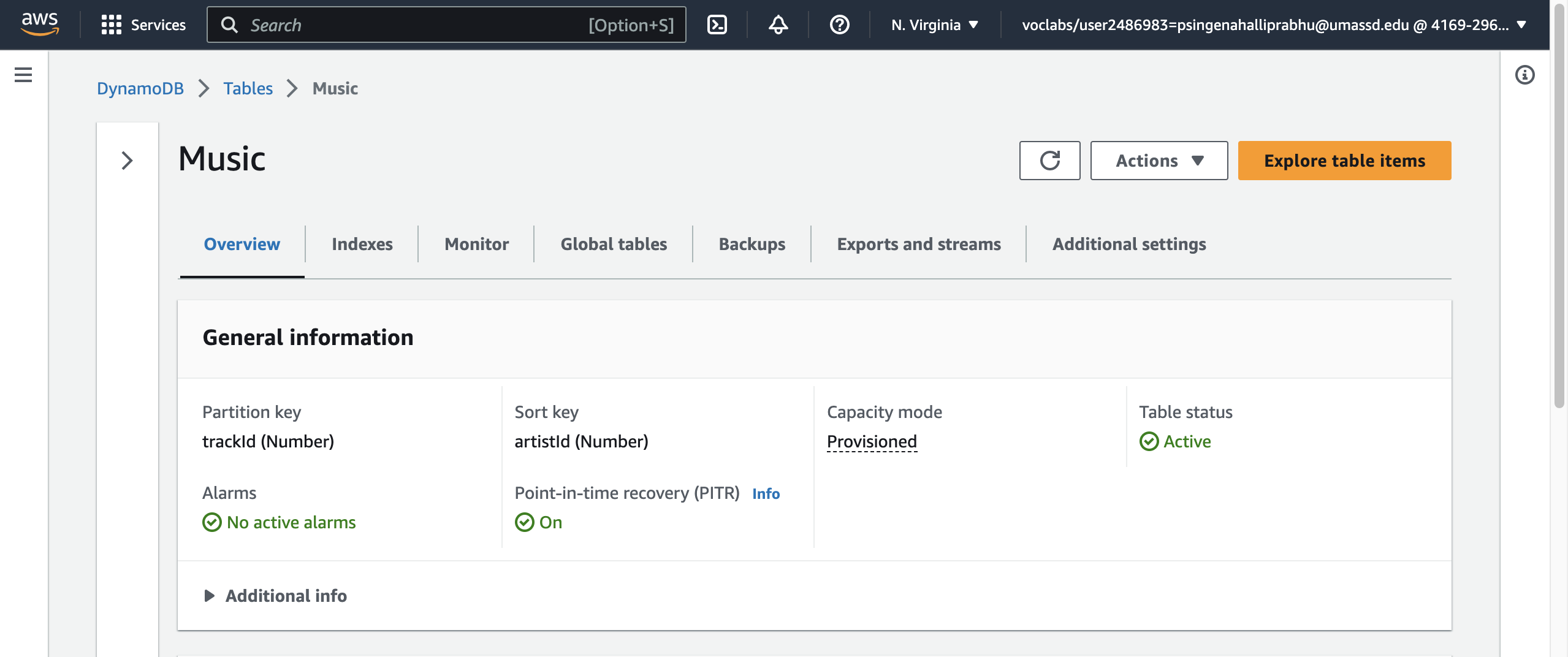
CIS 552: Database Design – Homework 5

# Introduction:

I successfully created an AWS Academy for Students account and set up a DynamoDB database in the AWS cloud. To store my music-related data, I created a table named "Music" with a well-designed schema.

# Table Creation:

The "Music" table has a partition key named "trackId," which serves as the primary key for the table. Additionally, I added a Sort Key called "artistId" to the table, which allows for efficient querying of data related to specific artists.



# Populating table:

I leveraged the publicly available iTunes Search API provided by Apple Computer to retrieve relevant data for populating my database. This API provides a comprehensive set of endpoints for searching and retrieving music-related data, including track names, artist names, album names, and other metadata.

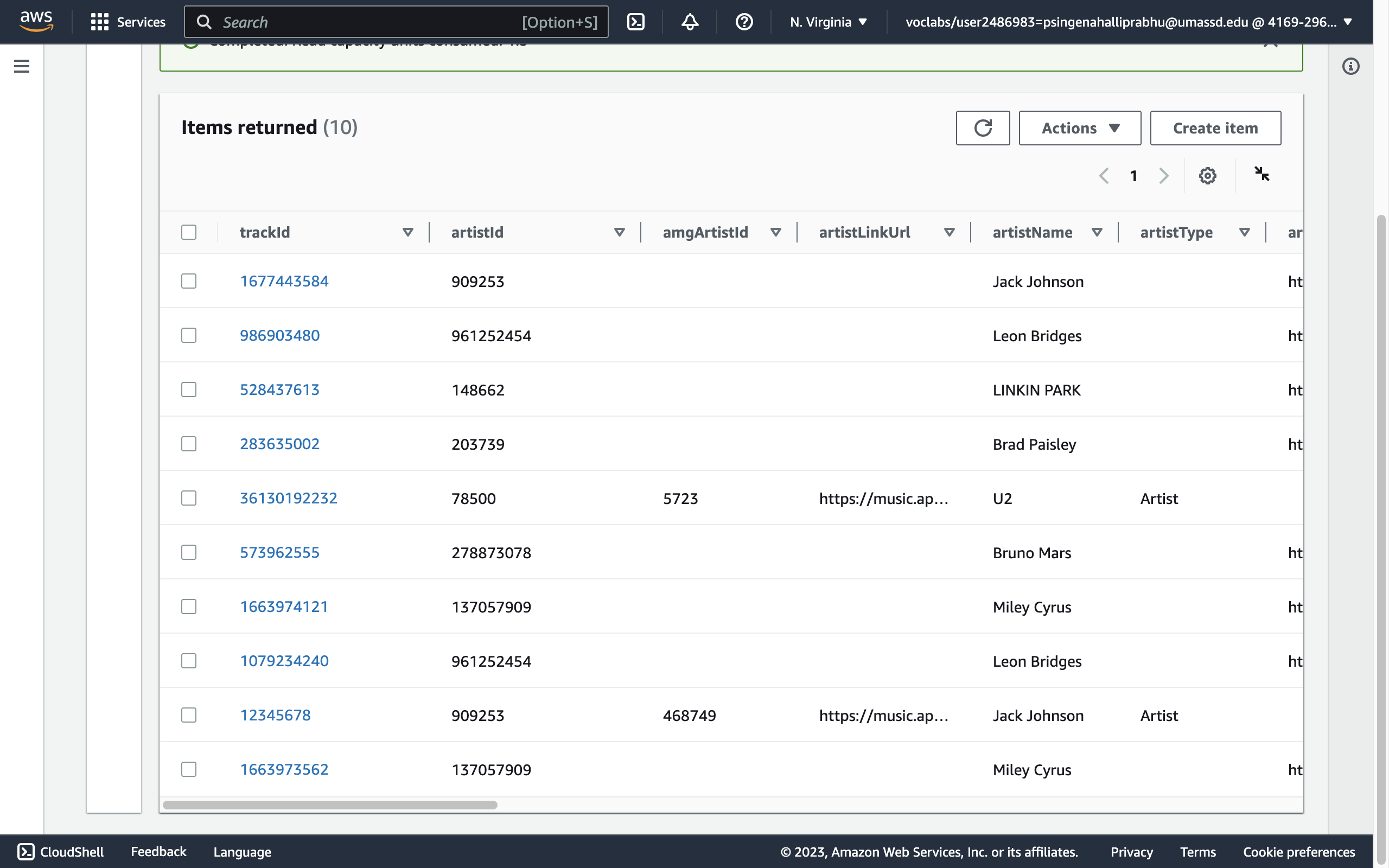
By utilizing this API, I was able to efficiently gather large amounts of data from the iTunes Store, and easily integrate it into my database. This approach allowed me to ensure the accuracy and consistency of the data, while also minimizing the time and effort required to manually enter the information.

I made sure to create items with varying fields to take advantage of the inherent flexibility and irregularity that non-SQL data and databases provide.

Graphical user interface, text, application

Description automatically generated

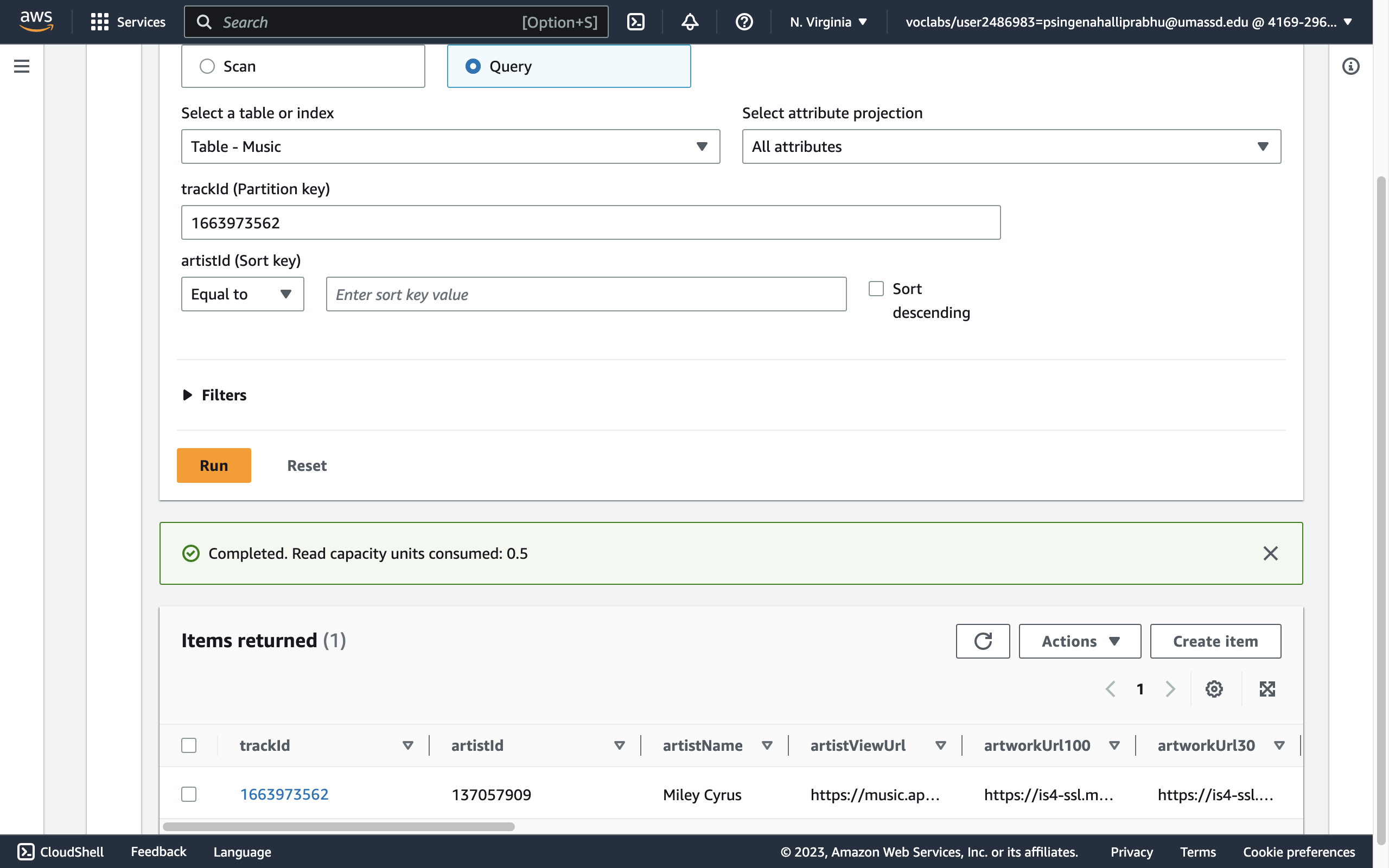
We can see from the below image, that the date is saved in the table.



# Query Data:

To retrieve specific tracks from my DynamoDB database, I utilized the trackId field as the partition key for my table. By performing queries using this key, I was able to retrieve precise data from the database with minimal latency.

The following image illustrates the results of a typical query using trackId as the partition key:



# Creating a Secondary Index:

As part of my DynamoDB implementation, I recognized the need for a secondary index that would enable me to efficiently query data based on artist names. To address this requirement, I created a secondary index on the "artistName" field, which allows me to quickly retrieve all tracks associated with a particular artist.

Graphical user interface, text, application

Description automatically generated

# Querying using Secondary Index

With the addition of a secondary index on the "artistName" field in my DynamoDB database, I am now able to easily query data based on specific artist names.

Graphical user interface, text, application, email

Description automatically generated

# Conclusion

In conclusion, my assignment has successfully utilized DynamoDB, a NoSQL database provided by AWS, to store and retrieve music-related data. Through the use of efficient querying techniques and secondary indexes, I have been able to effectively manage a large and diverse dataset, and easily retrieve information based on various criteria such as track ID and artist name. The use of the publicly available iTunes Search API has also proved to be an effective means of populating the database with accurate and comprehensive data. Overall, this assignment has provided me with valuable experience working with cloud-based databases and exploring the unique capabilities and challenges of NoSQL data management.