UDACITY

Logout

Return to Classroom

Data Modeling with Cassandra

REVIEW HISTORY

Meets Specifications

Hi There!

Amazing work in this submission! I was truly happy to see the results of all your learnings come so clearly into your second project. The queries appropriately created the **tables** and able to fetch the data from database by using the `SELECT` statements. Also, you perfectly show the understanding of the `DDL` and `DML` queries by creating the tables and migrating the data into them.

> **Don't forget to rate and share your views about the project review. Your feedback will help me improve my project review quality for your fellow students in the future.**

Extra Materials

Here are a few resources you might find useful for more insight and further learning.

- Why Data Warehouse Projects Fail
- How to build a data architecture to drive innovation—today and tomorrow

Keep up the good work and congratulations on passing the project. 🎉

All the best for your next project

ETL Pipeline Processing

✓ Student creates `event_data_new.csv` file.

Great Work!

The `event_data_new.csv` file was successfully created with `6821` rows. ✅

You Know!

ETL, which stands for **extract**, **transform**, and **load**, is a data integration process that combines data from multiple data sources into a single, consistent data store that is loaded into a **data warehouse** or other target systems. For further learning, you can visit ETL (Extract, Transform, Load).

✓ Student uses the appropriate datatype within the `CREATE` statement.

Well Done!

In the `CREATE` statements, all appropriate data types were correctly chosen. ✅

You Know!

The **Cassandra Query Language (CQL)** is a simple alternative to **Structured Query Language (SQL)**. It is a declarative language developed to provide communication with its database. Similar to `SQL`, `CQL` also stores data in tables and organizes data into rows and columns. For further learning, you can visit CQL Data Types. In this post In, you will learn some of the **different data types** of the **Apache Cassandra database**.

Data Modeling

✓ Student creates the correct Apache Cassandra tables for each of the three queries. The `CREATE TABLE` statement should include the appropriate table.

Good Work!

You had perfectly written the `CREATE TABLE` statements based on the question asked. ✅

✓ Student demonstrates good understanding of data modeling by generating correct SELECT statements to generate the result being asked for in the question.

The SELECT statement should NOT use `ALLOW FILTERING` to generate the results.

Well Done!

I'm able to get the required results based on the `SELECT` statements you had written. ✅

Must Read

Using SELECT * in an SQL Query Is a Bad Practice. In this article, you will learn some practical reasons why using `SELECT *` in an **SQL query is not a good idea**.

✓ **Student should use table names that reflect the query and the result it will generate. Table names should include alphanumeric characters and underscores, and table names must start with a letter.**

Well done!

You had followed the correct naming convention for each table query that provides a good general sense of query will generate. ✅

Naming Convention

A **naming convention** is a set of **unwritten rules** you should use if you want to **increase the readability** of the whole data model. For further learning, you can visit Learn SQL: Naming Conventions. In this post you will learn how you should formulate your naming convention, maybe even more important, why should you do it and what is the overall benefit of using it.

✓ **The sequence in which columns appear should reflect how the data is partitioned and the order of the data within the partitions.**

Great Job!

The column order in your `CREATE` and `INSERT` statements follow the order to the `COMPOSITE PRIMARY KEY` and `CLUSTERING` columns. ✅

You Know!

Clustering columns order data within a partition. When a table has **multiple clustering columns** the data is stored in **nested sort order**. The database uses the clustering information to identify where the

data is within the partition. Use logical statements for clustering columns to identify the clustering segment and return slices of the data. For further learning, you can visit Clustering columns.

PRIMARY KEYS

✓ **The combination of the PARTITION KEY alone or with the addition of CLUSTERING COLUMNS should be used appropriately to uniquely identify each row.**

Table 1 ✅
Table 2 ✅
Table 3 ✅

You Know!

Each table requires a **unique primary key**. The first field listed is the **partition key** since its hashed value is used to determine the node to store the data. If those fields are **wrapped in parentheses then the partition key is composite**. Otherwise, the **first field is the partition key**. Any fields listed **after the partition key are called clustering columns**. These store data in **ascending** or **descending** order within the partition for the fast retrieval of similar values. All the fields together are the **primary key**. For further learning, you can visit Partition Key vs Composite Key vs Clustering Columns in Cassandra. In this post, you learn the differences between `partition key`, `composite key`, and `clustering key` in Cassandra.

Presentation

✓ The notebooks should include a description of the query the data is modeled after.

Suggestion

If you want to know about **markdown syntax** for Jupiter you can visit Markdown for Jupyter notebooks cheatsheet

✓ Code should be organized well into the different queries. Any in-line comments that were clearly part of the project instructions should be removed so the notebook provides a professional look.

Well Done!

You had removed all the `#TODO` comments from the juypter notebook that are part of the project instructions. ✅

Suggestion

If you want to know the basic tips to use juypter you can visit 28 Jupyter Notebook Tips, Tricks, and Shortcuts

DOWNLOAD PROJECT

RETURN TO PATH