UDACITY

‹ Return to Classroom

Data Modeling with Cassandra

REVIEW

HISTORY

Requires Changes

5 specifications require changes

Dear student,

That was an excellent submission! 👏

There are still some minor adjustments to be made, but you have almost finished this project.
We also need to make sure that you include a description of how you modeled each table and remove unnecessary comments/instructions.

Check my comments and implement the necessary changes.
If you have any questions or need some help in fixing errors, check out the Knowledge Forum here.

Good luck with your next submission! 😃

ETL Pipeline Processing

Student creates `event_data_new.csv` file.

✅ **Good work here**. The `event_datafile_new.csv` was created correctly and it has 6821 lines which means that you followed all of the notebook's guidelines.

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

✅ **Good job!** You chose the correct data type for each column.
Be sure to keep this link for future reference in case you want to check the available Cassandra data types.

Data Modeling

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

include the appropriate table.

✅ **Great work.** You've created one table for each query, it's usually one of the best approaches for Apache Cassandra. Since it's not possible to perform table JOINs, a good approach is to model every table according to the queries you're going to perform.

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.

❌ This item requires some attention.

You used `SELECT *` for all queries, however, you should only select the **requested** columns for each.

This is very important because Cassandra usually contains lots of data and also receives many concurrent INSERTs. And since we're talking about a lot of data, selecting **all** columns every time will put an unnecessary load in the Cassandra cluster. Try to avoid that and select only what you need from each table.

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.

✅ **Good job here**. You've used one different name for each table that greatly represents its content.
It's important to choose names that best reflect each table's data as it may help other users that will use the same tables.

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

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 the `CREATE` statements corresponds to the `PRIMARY KEY` definitions.

You should do the same for the **INSERT** statement of the third table.

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 ✅

- For Table 2, it's not an error but an improvement. Since we're filtering by two columns `(user_id, session_id)` and the result **has to be sorted** by `item_in_session`, you can create a composite partition key like this `PRIMARY KEY ( (user_id, session_id), item_in_session))`. It'll improve overall performance because the userId data will be spread among more than just one node and it'll be much faster to look for a specific session_id, which is the case of the query we're making here. If you want to learn more about **composite partition keys**, check this link.

Presentation

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

❌ You must add a description for each CREATE TABLE explaining how/why you modeled your table and chose the partition keys / clustering columns
For example:

```
For table session_library, the column_a was used as a partition key because the queries will f
ilter by this column. column_b and column_c were used as clustering columns to help make up a
 unique key.
```

Try to answer questions like which columns will be primary keys and why? What clustering keys you are using and why?

Use Jupyter's markdown cells to create the headers and give your notebook a more professional look.

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.

❌ Even though we're working with a Jupyter notebook, it's very common to share them between coworkers or present your results to someone.

The notebook is already well organized, but I suggest you remove any unnecessary `#TO-DO` comments that are project's instructions like the ones below:

```
# We have provided part of the code to set up the CSV file. Please complete the Apache Cassand
ra code below#

##Assign the INSERT statements into the `query` variable for song table
```

```
## TO-DO: Assign t...


## Assign ...
```

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH

Rate this review

START