UDACITY

< Return to Classroom

DISCUSS ON STUDENT HUB

Data Modeling with Cassandra

REVIEW

HISTORY

Requires Changes

2 specifications require changes

Great work!

You have almost completed this project!

You should be very proud of your accomplishments in building a Data Model with Cassandra. You'll find feedback and some tips to help you continue to improve on your project. Your project code gave correct output as expected and I would recommend you to go through the official documentation always.

Few important things which I want you to check is the order of columns and use correct partition columns. Please check the specification comments and update the code accordingly. If you still face issues or have any questions, please ask using the Knowledge Hub platform.

Once done, please submit the project. Looking forward to your submission. All the best!

ETL Pipeline Processing

Student creates `event_data_new.csv` file.

Nice job! I see the event_data_new.csv file, which indicates you followed the ETL pipeline to create the csv file.

file.

(The file was generated after running your code in the Udacity workspace)

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

Nice work! You have used appropriate datatype within the Create Statement.

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 job! You followed the one table per query rule of Apache Cassandra. You are not replicating the same table for all three queries, which defies that rule. You have three distinct tables with unique tables names and uses appropriate CREATE table statements.

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.

Awesome work! You have correctly used select statements. You are not using `select *` in queries.

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.

We are looking for table names that provide a good general sense of what this query will generate. For e.g., for query 2, an appropriate table name should reflect song playlist in session (e.g., name could be song_playlist_session). You should not be using table names like query 1 or project 1, etc.

You have used table names that reflect the query and the result it will generate. Table names include alphanumeric characters and underscores, and table names start with a letter.

We are always concerned about correct table schemas. It becomes a lot difficult to manage and work on tables if the names are like table1,table2 and table3. One of my day-to-day work also includes creating multiple tables in production. Having thousands of tables, it's really handy to give them some name that we can understand and that relates to the query/business requirement.

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

This is what I really want you to focus on!

The sequence of the columns in the CREATE and INSERT statements should follow the order of the COMPOSITE PRIMARY KEY and CLUSTERING columns. The data should be inserted and retrieved in the same

order as how the COMPOSITE PRIMARY KEY is set up.

Apache Cassandra is a partition row store, which means the partition key determines which node a particular row is stored on. With the Primary key (which includes the Partition Key and any clustering columns), the partitions are distributed across the nodes of the cluster. It determines how data are chunked for write purposes. Any clustering column(s) would determine the order in which the data is sorted within the partition.

Here is some DataStax documentation you can refer to:
https://docs.datastax.com/en/dse/5.1/cql/cql/cql_using/whereClustering.html

For first table, instead of :

```
query_create = query_create + "(artist text, itemInSession int, length float, ses
sionId int, \
                               songTitle text, PRIMARY KEY (sessionId, itemInSes
sion))"
```

write:

```
query_create = query_create + "(artist text,  sessionId int, itemInSession int, l
ength float, \
                               songTitle text, PRIMARY KEY (sessionId, itemInSes
sion))"
```

See the order of sessionid and iteminsession!

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.

Excellent work at understanding and then implementing the PRIMARY KEY with a COMPOSITE Partition for both the CREATE and INSERT statements. Just a minor change required for the third query.

For e.g., in Query 3, you should not only use song as PARTITION KEY. Similarly, you should not need to user both firstName and lastName along with userId for query 3 clustering columns, as **song and userId** together will uniquely identify each row. You should include clustering columns as part of the COMPOSITE PRIMARY KEY and understand that a COMPOSITE PRIMARY KEY uniquely identifies each row

Presentation

Presentation

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

Good work including headers to denote descriptions of the query! It was clear how each query was being addressed.

Formatting (Optional/For reference)

You might want to look into using PrettyTable. For example:

```
# use pretty table to display
data in tabular form and include headings
t = PrettyTable(['Artist', 'Song', 'Length'])
for row in rows:
    t.add_row([row.artist, row.song, row.length])
print(t)
```

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.

The code has been correctly organised into different queries correctly! You have removed all the TO-DO's which gave the notebook professional look. Nice.

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH

Rate this review

START