Final Assignment - Querying data in NoSQL databases

Estimated time needed: **90** minutes.

It is highly recommened that you finish the Setup and Practice Assignment Lab before you proceed with this Assignment.

About This SN Labs Cloud IDE

This Skills Network Labs Cloud IDE provides a hands-on environment for course and project related labs. It utilizes Theia, an open-source IDE (Integrated Development Environment) platform, that can be run on desktop or on the cloud. To complete this lab, we will be using the Cloud IDE based on Theia and Cassandra and Mongodb running in a Docker container. You will also need an instance of Cloudant running in IBM Cloud.

Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persisted. Every time you connect to this lab, a new environment is created for you. Any data you may have saved in the earlier session would get lost. Plan to complete these labs in a single session, to avoid losing your data.

Scenario

You are a data engineer at a data analytics consulting company. Your company prides itself in being able to efficiently handle data in any format on any database on any platform. Analysts in your office need to work with data on different databases, and data in different formats. While they are good at analyzing data, they count on you to be able to move data from external sources into various databases, move data from one type of database to another, and be able to run basic queries on various databases.

Objectives

In this assignment you will:

- replicate a Cloudant database.
- create indexes on a Cloudant database.
- query data in a Cloudant database.
- import data into a MongoDB database.
- query data in a MongoDB database.
- export data from MongoDB.

- import data into a Cassandra database.
- query data in a Cassandra database.

Note - Screenshots

Throughout this lab you will be prompted to take screenshots and save them on your own device. These screenshots will be needed to be uploaded for peer review in the next section of the course. You can use various free screengrabbing tools to do this or use your operating system's shortcut keys to do this (for example Alt+PrintScreen in Windows).

Exercise 1 - Check the lab environment

Before you proceed with the assignment :

- Check if you have the 'couchimport' tool installed on the lab, otherwise install it.
- Check if you have the 'mongoimport' and 'mongoexport' tools installed on the lab, otherwise install them.
- Check if the environment variable CLOUDANTURL is set, otherwise set it.

Right-click on the link and "open in a new tab" to obtain the content of the json file and copy the entire content: movies.json

Now, go to the lab environment by clicking on the Launch/Open Tool button, and create a new file named movie.json under the project directory and paste the entire content copied earlier into the newly created json file.

Export the json into your Cloudant Database

Go to your Cloudant Service created on IBM Cloud and then click on Launch Dashboard to launch the cloudant databases. Then, click on New database on the top-right corner and create a Non-patitioned database named movies.

Run the given command in the terminal to export the data of the "movies.json" file into your Cloudant database "movies":

```
curl -XPOST $CLOUDANTURL/movies/_bulk_docs -Hcontent-type:application/json -d @movies.json
```

Exercise 2 - Working with a Cloudant database

Task 1 - Replicate a local database into your Cloudant instance

Using the replication page in the Cloudant dashboard, replicate the below source database using one time replication,

Source

- Type : Local Database

- Name: movies
- Authentication : IAM Authentication

Add your Cloudant API key below.(You can find it in the service credentials tab)

Target

- Type: New local database
- Authentication : IAM Authentication

Add your Cloudant API key below.(You can find it in the service credentials tab)

Options

- Replication type: One time

Click on Start Replication

Once the replication is done take a screenshot of the database's dashboard (which shows database names, number of rows etc,).

Name the screenshot as `1-replication.jpg`. (images can be saved with either .jpg or .png extension)

Task 2 - Create an index for the "Director" key, on the 'movies' database using the HTTP API

Take a screenshot of the command you used and the output.

Name the screenshot as `2-index-director.jpg`. (images can be saved with either .jpg or .png extension)

Task 3 - Write a query to find all movies directed by 'Richard Gage' using the HTTP API

Take a screenshot of the command you used and the output.

Name the screenshot as `3-query-director.jpg`. (images can be saved with either .jpg or .png extension)

Task 4 - Create an index for the "title" key, on the 'movies' database using the HTTP API

Take a screenshot of the command you used and the output.

Name the screenshot as `4-index-title.jpg`. (images can be saved with either .jpg or .png extension)

Task 5 - Write a query to list only the "year" and "Director" keys for the 'Top Dog' movies using the HTTP API

Take a screenshot of the command you used and the output.

Name the screenshot as `5-query-title.jpg`. (images can be saved with either .jpg or .png extension)

Task 6 - Export the data from the 'movies' database into a file named 'movies.json'

Take a screenshot of the command you used and the output.

Name the screenshot as `6-couchexport.jpg`. (images can be saved with either .jpg or .png extension)

Exercise 3 - Working with a MongoDB database

Task 7 - Import 'movies.json' into mongodb server into a database named 'entertainment' and a collection named 'movies'

Take a screenshot of the command you used and the output.

Name the screenshot as `7-mongoimport.jpg`. (images can be saved with either .jpg or .png extension)

Note: Please use the movies.json (new file that gets created in the project directory as part of Task6) and not the movie.json (previous file that was downloaded as part of Exercise:1) for this task.

Task 8 - Write a mongodb query to find the year in which most number of movies were released

▼ Click here for Hint

The $group stage can be used to group documents by a certain field. You can calculate the count of movies within the group using the $sum aggregation operator.
Your query should:

- Group movies by their release year.
- Calculate the total count of movies for each year.
- Sort the years in descending order of movie count.
- Limit the output to 1 document (year) which has the highest movie count.

```
1
2
3
4
5
6
7
8
9
```

```
db.movies.aggregate([
{
    "$group":{
        "_id":"$director",
        "moviecount":{"$sum":1}
        }
}
])
```

Copied!

Take a screenshot of the command you used and the output.

Name the screenshot as `8-mongo-query.jpg`. (images can be saved with either .jpg or .png extension)

Task 9 - Write a mongodb query to find the count of movies released after the year 1999

Take a screenshot of the command you used and the output.

Name the screenshot as `9-mongo-query.jpg`. (images can be saved with either .jpg or .png extension)

Task 10. Write a query to find out the average votes for movies released in 2007

▼ Click here for Hint

use the $match operator to filter for movies released in 2007.
example usage: db.books.aggregate(
[{ $match : { author : “john” } }]
)
will filter out all the books written by the author `john`

Take a screenshot of the command you used and the output.

Name the screenshot as `10-mongo-query.jpg`. (images can be saved with either .jpg or .png extension)

Task 11 - Export the fields `_id`, “title”, “year”, “rating” and “director” from the ‘movies’ collection into a file named `partial_data.csv`

Take a screenshot of the command you used and the output.

Name the screenshot as `11-mongoexport.jpg`. (images can be saved with either .jpg or .png extension)

Exercise 4 - Working with a Cassandra database

Task 12 - Import ‘partial_data.csv’ into cassandra server into a keyspace named ‘entertainment’ and a table named ‘movies’

Note: While creating the table `movies` make the all the columns as text columns including the id column.

Take a screenshot of the command you used and the output.

Name the screenshot as `12-cassandra-import.jpg`. (images can be saved with either .jpg or .png extension)

Task 13 - Write a cql query to count the number of rows in the ‘movies’ table

Take a screenshot of the command you used and the output.

Name the screenshot as `13-cassandra-query.jpg`. (images can be saved with either .jpg or .png extension)

Task 14 - Create an index for the “rating” column in the ‘movies’ table using cql

Take a screenshot of the command you used and the output.

Name the screenshot as `14-cassandra-index.jpg`. (images can be saved with either .jpg or .png extension)

Task 15 - Write a cql query to count the number of movies that are rated 'G'.

Take a screenshot of the command you used and the output.

Name the screenshot as `15-cassandra-query.jpg`. (images can be saved with either .jpg or .png extension)

End of assignment.

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-02-27	0.8	Sapthashree	Updated task 8 to match rubric
2023-02-27	0.7	Lavanya Rajalingam	Minor Note added for clarity
2023-02-27	0.6	Lavanya Rajalingam	Minor correction related to case of letters to match with DB
2022-05-27	0.5	Pallavi Rai	Instructions and json file updated
2021-06-08	0.4	Malika Singla	Updated task 10 to match rubric
2021-04-23	0.3	Steve Ryan	Review pass
2021-04-16	0.2	Ramesh Sannareddy	Reorganised the Exercises and Tasks
2021-04-14	0.1	Ramesh Sannareddy	Created initial version