IST769 Lab F

# Document Model: MongoDb

### In this lab, we will explore the document database model and MongoDb. We will learn to import / export data into MongoDb and query the database using Spark, Drill and MQL – The mongo query language.

### Learning Outcomes

At the end of this lab you should be able to:

* Use Apache Spark to import and export data into MongoDb
* Perform CRUD operations with the MongoDb Query Language (MQL)
* Query MongoDb with Drill SQL, Spark SQL and MQL.

### Pre-Requisites

Before you begin:

* Open a terminal window in the lab environment
* Set the current working directory to **advanced-databases**
* Start the following services required by the lab Drill, Spark and MongoDb:   
  **jupyter drill mongo mongoapp mongoexpress**

### Tools Used In this Lab

The following tools will be used in this lab:

1. To access Jupyter Lab from your Windows host:  
   [http://localhost:8888](http://localhost:8888/)   
   The password is **SU2orange!**
2. Log-in to the drill Web UI from your windows host:  
   [https://localhost:8047](https://localhost:8047/)
3. To access the Mongo Express Admin UI:  
   [http://localhost:8881](http://localhost:8881/)
4. To access the Mongo Web App Example:   
   [http://localhost:5081](http://localhost:5081/)
5. To access the MongoDb Shell:

PS> docker-compose exec mongo mongosh -u admin -p mongopw   
 --authenticationDatabase=admin

1. Drill Storage plugins can be found in the **drill-storage-plugins** folder. Run   
   PS: advanced-databases> **code .**    
   from the command line to open the code editor.

# Lab Problem Set

**QUESTIONS:   
  
For each question, include a copy of the code required to complete the question along with a screenshot of the code and a screenshot of the output.**

1. Use spark to load the **/datasets/json-samples/US-Senators.jso**n into the MongoDb database **labf** under the collection **senators**.

# Spark init

import pyspark

from pyspark.sql import SparkSession

mongo\_uri = "mongodb://admin:mongopw@mongo:27017/admin?authSource=admin"

spark = SparkSession \

.builder \

.master("local") \

.appName('jupyter-pyspark') \

.config("spark.mongodb.input.uri", mongo\_uri) \

.config("spark.mongodb.output.uri", mongo\_uri) \

.config("spark.jars.packages","org.mongodb.spark:mongo-spark-connector\_2.12:3.0.1")\

.getOrCreate()

sc = spark.sparkContext

sc.setLogLevel("ERROR")

#1

**df = spark.read.json("file:///home/jovyan/datasets/json-samples/US-Senators.json")**

**df.write.format("mongo") \**

**.mode("overwrite") \**

**.option("database", "labf") \**

**.option("collection", "senators") \**

**.save()**

1. From the Mongo Client, retrieve the firstname lastname , state and party for those senators in either the “Republican”, “Democrat, or “Independent” party.

**db.senators.find( {party:'Independent'},{ party:1, person: { firstname:2, lastname:3 }, state:4 })**

1. From the Mongo Client, write an in MQL index to improve the query performance of question 2. Run the getIndexes() function on the collection to prove you created the index. Then use explain to prove the index is being used.

**db.senators.createIndex( {party:1} )**

**db.senators.getIndexes()**

**db.senators.find( {party:'Independent'},{ party:1, person: { firstname:2, lastname:3 }, state:4 }).explain('executionStats')**

1. Use Spark to Load in the **/datasets/netflix-canceled-2021/\*.json** into Mongodb database **labf** under the collection **nfcan**

**nfcan = spark.read.option("multiline", True).json("file:///home/jovyan/datasets/netflix-canceled-2021/\*.json")**

**nfcan.write.format("mongo") \**

**.mode("overwrite") \**

**.option("database", "labf") \**

**.option("collection", "nfcan") \**

**.save()**

1. Using Drill, write a query to list the name premier date and average rating of cancelled Netflix 2021 shows. Limit the shows to those with an average rating under 7.

**WITH table1 AS (**

**SELECT n.name, n.premiered, n.rating.average as rating**

**FROM mongo.labf.nfcan n**

**)**

**SELECT \***

**FROM table1**

**WHERE rating < 7;**

1. Using Drill, write a query to list the name, premier date, and genre for only those shows in the History or Family genres. Make sure to include a column to display the name of the Genre.

**WITH table1 AS (**

**SELECT n.name, n.premiered, flatten(n.genres) AS genre**

**FROM mongo.labf.nfcan n**

**)**

**SELECT \***

**FROM table1**

**WHERE genre IN ('Family', 'History');**

1. Using Spark or Spark SQL, create a DataFrame or view from the Netflix Cancellations MongoDb data, consisting of show name, season number, episode, number, episode name, airdate, and average rating (for the episode).

**nfcan = spark.read.format("mongo").option("database", "labf").option("collection", "nfcan").load()**

**from pyspark.sql.functions import col, explode**

**tmp = nfcan.select( col("name").alias("showname"),**

**explode("\_embedded.episodes").alias("episode") )**

**episodes = tmp.select("showname",**

**col("episode.name").alias("episode\_name"),**

**"episode.season",**

**"episode.number",**

**"episode.airdate",**

**"episode.rating.average")**

**episodes.show(5)**

1. Using the query you wrote in question 7 (if you want), write a Spark or Spark SQL query to get the lowest rated episodes of each season for the cancelled shows. Display show name, season number, episode number, episode name, and rating for that episode.   
   NOTE: some shows have more than one episode with the lowest rating.

**query = '''**

**with table1 as (**

**select**

**showname,**

**episode\_name,**

**season,**

**number,**

**airdate,**

**average,**

**min(average) over (partition by showname, season) as lowest\_rated\_season**

**from episodes**

**)**

**select \***

**from**

**table1**

**where**

**lowest\_rated\_season = average**

**order by**

**showname,**

**season**

**'''**

**spark.sql(query).show(5)**

1. CHALLENGE YOURSELF! Display Name of show, a picture of the show, and show summary. Make it interactive so you can select the show and see the details.

**from IPython.display import display, HTML, Image**

**from ipywidgets import interact, interact\_manual**

**display(HTML("<h1>Netflix Cancelled Shows of 2021</h1>"))**

**shows = nfcan.select("name").distinct().sort("name").toPandas()["name"].values**

**@interact(show = shows)**

**def onchange(show):**

**info = nfcan.select("name", "summary", "image.medium", "status", "rating.average").where(nfcan.name == show).toPandas().iloc[0]**

**display(HTML(f"<h3>{info['name']}</h3>"))**

**display(HTML(f"<p>STATUS: <b>{info['status']}</b> RATING: <b>{info['average']}</b>"))**

**display(HTML(info['summary']))**

**display(Image(url = info['medium']))**

**IMPORTANT:** When you are finished with the lab, execute:

PS:> docker-compose stop

To turn off all running services, then shut down your Azure Lab instance.