Problem Set F Submission Form

# Overview

|  |  |
| --- | --- |
| Your Name | Jaya Varshini Prabakar |
| Your SU Email | jprabaka@syr.edu |

# Instructions

Put your name and SU email at the top. Answer these questions all from the lab. When asked to include screenshots, please follow the screen shot guidelines from the first homework.

Remember as you complete the homework it is not only about getting it right / correct. We will discuss the answers in class so it’s important to articulate anything you would like to contribute to the discussion in your answer:

* If you feel the question is vague, include any assumptions you've made.
* If you feel the answer requires interpretation or justification provide it.
* If you do not know the answer to the question, articulate what you tried and how you are stuck.
* Highlight any doubts or questions you would like me to review.

This how you receive credit for answering questions which might not be correct. In addition, you must complete the reflection portion of the homework assignment for full credit. Since most answers will be similar this is an important part of your individual submission.

Complete Part II of this document first, then go back and complete the Reflection in Part I.

# Part I - Reflection

Use this section to reflect on your learning. To achieve the highest grade on the assignment you must be as descriptive and personal as possible with your reflection.

1. As you completed this assignment, identify what you learned.

MongoDB with Drill

Spark to MongoDB operations

1. What barriers or challenges did you encounter while completing this assignment?

Question 5 was hard to figure out in Drill, I had to do option changes with mongo and spark in order to make the query work. Even after drill read the double type as Character so had to perform casting in order to get the query working.

1. How prepared were you to complete this assignment? What can you do to be better prepared?

Well prepared

1. Rate your comfort level with this week’s material. Use the rubric provided.

3  
4 ==> I understand this material and can explain it to others.  
3 ==> I understand this material.  
2 ==> I somewhat understand the material but sometimes need guidance from others.  
1 ==> I understand very little of this material and need extra help.

# Part II – 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**.  
   **Code:**

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()

**Screenshots:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

test labf

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

**Screenshots:**

A screenshot of a computer program

Description automatically generated

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.   
   **Code:**

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

db.senators.getIndexes()

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

**Explanation:**

Without Indexing we can see that the query examines all 99 rows and then filters the party with ‘Republican’ in it.

When we create an Index on party, the query seeks the Republican party only which is a best query practice and also improves query performance.

**Screenshots:**

A computer screen shot of a program

Description automatically generated

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

**Code:**

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()

**Screenshots:**

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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 a average rating under 7.   
     
   **Code:**

select n.name,n.premiered,cast(n.rating.`average` as DECIMAL(28,1)) from mongo.labf.nfcan n

where cast(n.rating.`average` as DECIMAL(28,1)) < cast (7 as DECIMAL(28,1) )

**Screenshot:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

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.  
   **Code:**

With gn as(

select n.name, n.premiered, flatten(n.genres) as genre

from mongo.labf.nfcan n

)

select \* from gn where genre in ('Family','History')

**Screenshot:**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

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).

**Code:**

from pyspark.sql.functions import col, explode

temp = df2.select(col("name").alias("showname"),

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

)

temp.select("showname","episode.name","episode.season","episode.number", "episode.airdate","episode.rating.average").show()

**Screenshot:**

A screenshot of a computer

Description automatically generated

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.

**Code**:

query = '''

With src as (

select showname, name, season, number, airdate, average, min(average) over

(partition by showname, season) as lowest\_rated\_in\_season from ep\_df

)

select \* from src where lowest\_rated\_in\_season = average

order by showname, season

'''

spark.sql(query).show()

**Screenshot:**

A screenshot of a computer

Description automatically generated

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.

**Code:**

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>STAUS:<b>{info['status']}</b> RATING:<b>{info['average']}</b> </p>"))

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

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

**Screenshot:**

A screenshot of a computer

Description automatically generated