quiz 3

wbg231

January 2023

1 Introduction

part 1

question 1

- how would you wtite the following in spark sql as opposed to using the object interface? sailors.filter(sailors.age ¿ 40).select(sailors.sid, sailors.sname, sailors.age)
- spark.sql('SELECT sid, sname, age from sailors where age ¿ 40; ')

question 2

- How would you express the following using the object interface instead of SQL? spark.sql('SELECT sid, COUNT(bid) from reserves WHERE bid != 101 GROUP BY sid')
- $query - 2 = reserves.where("bid! = 101").groupBy("sid").count()$

question 3

- question: Using a single SQL query, how many distinct boats did each sailor reserve? The resulting DataFrame should include the sailor's id, name, and the count of distinct boats. (Hint: you may need to use first(...) aggregation function on some columns.) Provide both your query and the resulting DataFrame in your response to this question.
- query-3=spark.sql("select sailors.sid, first(sailors.sname) as name, count(distinct reserves.bid) from sailors Left JOIN reserves ON sailors.sid=reserves.sid group by sailors.sid")

question 4

- question : Implement a query using Spark transformations which finds for each artist term, compute the median year of release, maximum track duration, and the total number of artists for that term (by ID). What are the results for the ten terms with the shortest average track durations? Include both your query code and resulting DataFrame in your response.
- querry-4=artist-term.join(tracks,artist-term["artistID"] == tracks["artistID"],"right").groupby("term").agg(F.expr('percentile-approx(year, 0.5)').alias('median year'), count(artist-term["artistID"]).alias("count id "), max("duration").alias("max-duration")).orderBy(mean(tracks["duration"])).limit(10)

question 5

question: Question 5: Create a query using Spark transformations that finds the number of distinct tracks associated (through artistID) to each term. Modify this query to return only the top 10 most popular terms, and again for the bottom 10. Include each query and the tables for the top 10 most popular terms and the 10 least popular terms in your response.

- querry_5_2=artist_term.join(tracks,artist_term["artistID"] == tracks["artistID"],"right").groupby("term").agg(countDistinct(artist_term["artistID"]).alias("number distinct artists")).orderBy(countDistinct(artist_term["artistID"]).desc()).limit(10)
- for top 10

part 2

- stuff on partitioing in spark
- replication factor in spark
- description:In this part of the assignment, you will be comparing the speed of Spark queries against DataFrames backed by either CSV or Parquet file stores, and optimizing the storage to speed up queries.

big spender

- task : This function returns a uncomputed dataframe that will contains users with at least 100 orders but do not yet have a rewards card.
- basic querry on csv Orders.filter(orders.rewards === False).filter(orders.orders >= 100).select(firstname, lastname)
- i think it is a good idea to partion by orders, reward

- then sort by first name and last name
- there is no reaosn to increase replicaiton factor

sum orders

- task : will compute the total orders grouped by zipcode
- basic querry on csv Orders.groupby(orders.zipcode).agg(count(orders))
- may want to sort by zipcode
- there is no reason to sort on orders sine we are just summing
- then repartion to a good number
- do not want to partion by zip code col since there could be many zipcodes and this could result in key skew if some zipcodes are more populos
- we are grouping by zipcodes which i think by and large are small so raising the replication factor is not important