REVATURE WEEK - 7 REVIEW QUESTIONS

1. **What is Spark SQL?**

   Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called Data Frames and can also act as a distributed SQL query engine. It enables unmodified Hadoop Hive queries to run up to 100x faster on existing deployments and data.

2. **How does a broadcast join work in Spark?**

   A broadcast is a join that joins on two relations by first broadcasting the smaller one to all Spark executors, then evaluating the join criteria with each executor's partitions of the other relation.

3. **Why are broadcast joins significantly faster than shuffle joins?**

   When the broadcasted relation is small enough, broadcast joins are fast, as they require minimal data shuffling. Above a certain threshold however, broadcast joins tend to be less reliable or performant than shuffle-based join algorithms

4. **How does Spark SQL evaluate a SQL query?**

   It uses a catalyst optimizer- which transforms a Spark SQL query to create a logical plan, which is then translated into a bunch of potential physical plans, one of which is selected for final evaluation based on a cost model

5. **What is the catalyst optimizer?**
   it's used in Spark SQL to translate the initial query into a logical plan, and finally a workable physical plan that is then evaluated.

6. **Why are there multiple APIs to work with Spark SQL?**

   We have three ways of using Spark SQL: SQL queries, DataFrames, and DataSets. DataFrames are like tables stored in SQL. They contain rows/records and have columns. The columns are stored by name, with datatypes.

   DataSets we can think of as between DataFrames and RDDs. DataSets contain strongly typed data, we use case classes for them in Scala. Both of these are distributed collections, and in Scala they are quite similar

7. **What are DataFrames?**

   They're like tables stored in SQL. They contain rows/records and have columns. The columns are stored by name, with datatypes

8. **What are DataSets?**

   Dataset is a data structure in SparkSQL which is strongly typed and is a map to a relational schema. It represents structured queries with encoders.

9. **How are DataFrames and DataSets "unified" in Spark 2.0?**

   type DataFrame = Dataset[Row]

10. **What is the SparkSession?**

    SparkSession is the entry point to Spark SQL. It is one of the very first objects you create while developing a Spark SQL application. As a Spark developer, you create a SparkSession using the SparkSession.

11. **Can we access the SparkContext via a SparkSession?**

    Yes, SparkContext can be accessed from within a SparkSession

12. **What other contexts are superseded by SparkSession?**

    SparkContext, SqlContext and HiveContext

13. **What are some data formats we can query with Spark SQL?**

    JSON, CSV, Parquet, Hive Tables, and SQL Tables

14. **Are Dataframes lazily evalauted, like RDDs?**

    Yes, Datasets, RDD's and Dataframes are lazy. it is necessary to call an action, such as .show to evaluate a DataSet.

15. **List functions available to us when using DataFrames?**

    .select, .filter, .groupBy, .agg, .drop, .write

16. **What's the difference between aggregate and scalar functions?**

    Aggregate functions operate against a collection of values and return a single summarizing value. Scalar functions return a single value based on scalar input arguments

17. How do we convert a DataFrame to a DataSet?

via a call to **.as** and the providing of a defined case class for typing, i.e. *ds = df.as[caseclass]*

18. How do we provide structure to the data contained in a DataSet?

via providing a pre-defined case class as type

19. How do we make a Dataset queryable using SQL strings?

A temp view must be created via a call to .createOrReplaceTempView, i.e. ds.createOrReplaceTempView("view name")

20. What is the return type of spark.sql("SELECT * FROM mytable") ?

Dataframe

21. How do we see the logical and physical plans produced to evaluate a DataSet?

We make a call to .explain(true). The true tells Spark to provide an extended explanation which includes the physical plan, otherwise, only the logical plan will be shown.

22. How do you add a new coloumn in Dataframes?

withColumn

23. How do you rename column in dataframe?

withColumnRename

24. What is the difference between inner, outer left, outer right, and outer full joins?

Inner join will only return the rows where the join condition is true for both tables.

Outer left and outer right joins return the rows in which the join condition is true, as well as the additional rows where it is not for either the table being joined to (left) or the table being joined (right).

Outer joins will include the records where the join condition is true, as well as all additional records from both tables.

25. What is a cross join / cartesian join?

All records from one table are joined with all records from another table, creating an output table that contains all possible combinations of records.

26. If I join two datasets with 10 records each, what is the maximum possible number of records in the output?

Using an outer full join, if no records match in either table, the maximum number of possible output records is 20, i.e. all records from both tables.

27. How many records would be in the output of a cross join/cartesian join?

AxB where A is the number of records in table A and B is the number of records in table B. For this example, 100 records will be in the output table.

28. What is Parquet?

Apache Parquet is a free and open-source column-oriented data storage format of the Apache Hadoop

29. What does it mean that parquet is columnar storage?

Columnar storage like Apache Parquet is designed to bring efficiency compared to row-based files like CSV. When querying, columnar storage you can skip over the non-relevant data very quickly. As a result, aggregation queries are less time consuming compared to row-oriented databases.

30. How can we partition files in Spark?

Coalesce, Repartition

31. What are some benefits of storing your data in partitions?

Storing data in partitions allows for faster queries based on those partitions.
(Potentially) Improved Security.
Better File Organization