Spark SQL

In this section we cover:

- Spark SQL
- DataFrames
- Creating DataFrames from different data sources
- Operations on DataFrames

Spark SQL – Introduction

- Spark SQL is a Spark module for processing structured data.
- Spark SQL interfaces provide more information than RDD APIs about the structure of the data and the computation being performed. The additional information is used for better optimization of the operations on the data.
- DataFrame is the primary programming abstraction in SparkSQL.
- A DataFrame is a distributed collection of data organized into named columns.
- It can be considered equivalent to a table in a relational database or a data frame in Python/R at a conceptual level, but with richer optimizations.
- DataFrames can be constructed from a wide array of sources such as:
  - structured data files
  - tables in Hive
  - existing RDDs
- Once created, they can be manipulated using the various domain-specific-language (DSL) functions which include select, filter, group by, aggregations etc.
- In addition DataFrames also have a rich library of functions including string manipulation, date arithmetic, common math operations and so on.

Spark SQL – SparkSession

- The entry point into all functionality in Spark SQL is the `SparkSession` class.
- To create a basic SparkSession `SparkSession.builder()` is to be used

```
from pyspark.sql import SparkSession
spark = SparkSession.builder().appName("Spark SQL Example")
.getOrCreate()
```

- The sql function on a SparkSession enables applications to run SQL queries programmatically and returns the result as a DataFrame.

```
// Register the DataFrame as a SQL temporary view
df.createOrReplaceTempView("people")
sqlDF = spark.sql("SELECT * FROM people")
sqlDF.show()
```

- Let us now create DataFrames using the different sources mentioned. And use the functions and SQL queries on them with a couple of input datasets.

DataFrames – Data Source Formats

- SparkSQL supports a variety of input data formats in its DataFrame interface.
- We can read data from any of the following file formats and load data into DataFrame:
  - Structured delimited files such as comma-separated (CSV)
  - JSON files or
  - Parquet files
- Note that SparkSQL uses Parquet as the default format for reading and writing of DataFrames
  - Parquet is created by Apache for Hadoop ecosystem; it is a compressed, efficient columnar format that automatically preserves the schema of the original data
- We can apply any relational transformations required on the DataFrames
- We can register a DataFrame as a temporary view and run SQL queries on its data
- We can store the Dataframe into files of any of these formats as well.

DataFrames – RDDs

- Spark SQL supports two different methods for converting existing RDDs into Datasets.
- The first method uses reflection to infer the schema of an RDD that contains specific types of objects.
- This reflection based approach is simpler and more straightforward. It can be used when we already know the schema while writing the Spark application.
- The second method for creating DataFrames is through a programmatic interface. It allows constructing a schema and applying it to an existing RDD.
- This method can be used when the schema is not known beforehand but column names are available for example in a string.
- A DataFrame in this way can be created programmatically with three steps.
  - Create an RDD of tuples or lists from the original RDD
  - Create the schema represented by a StructType matching the structure of the tuples or lists in the RDD of the last step.
  - Apply the schema to the RDD via createDataFrame method provided by SparkSession

DataFrames – Hive Tables

- Spark SQL supports reading and writing data in Apache Hive.
- For using Hive tables with Spark we need to start Hadoop using the following commands.
  - `$HADOOP_HOME/sbin/start-dfs.sh`
  - `$HADOOP_HOME/sbin/start-yarn.sh`
- When working with Hive, we need to instantiate SparkSession with Hive support and ensure that Spark has access to Hive metastore.
- On the VM Spark is configured to have access to Hive's metastore.
- In SparkSQL we can use Hive's DDL commands such as create database, create table if required besides DML for querying.