Spark SQL Overview and Architecture

**Spark SQL Overview**

* **Definition**: Spark SQL is a component of Apache Spark that provides a programming interface for working with structured and semi-structured data.
* **Features**:
  + Unified data processing
  + Integration with Hive
  + Support for SQL queries
  + DataFrames and Datasets API
* **Example**:

SELECT \* FROM employees WHERE salary > 50000

**Spark SQL Architecture**

* **Components**:
  + **Spark SQL Engine**: Executes queries using Spark’s execution engine.
  + **Catalyst Optimizer**: Optimizes query plans.
  + **Data Sources API**: Connects to various data sources (e.g., Hive, Parquet, JSON).
* **Diagram**: [Include a diagram showing the architecture components]
* **Example**:

val df = spark.read.json("path/to/json")

df.createOrReplaceTempView("people")

val results = spark.sql("SELECT name FROM people WHERE age BETWEEN 13 AND 19")

**Catalyst Optimizer**

* **Definition**: Catalyst is the query optimization framework in Spark SQL.
* **Features**:
  + **Rule-based Optimizations**: E.g., constant folding, predicate pushdown.
  + **Cost-based Optimizations**: Estimating the cost of different query plans.
* **Example**:

val df = spark.sql("SELECT name, COUNT(\*) FROM people GROUP BY name")

// Catalyst will optimize this aggregation query for better performance.

**Plan Optimization & Execution**

* **Logical Plan**: Represents the query as it is written.
* **Physical Plan**: Represents the execution strategy chosen by the optimizer.
* **Example**:

val logicalPlan = df.queryExecution.logical

val physicalPlan = df.queryExecution.executedPlan

**ROW API**

* **Definition**: Provides an API for working with rows of data in DataFrames.
* **Methods**:
  + getAs[T](fieldName: String): T
  + schema: StructType
* **Example**:

val row = df.first()

val name = row.getAs[String]("name")

**Querying Using Temporary Views**

* **Definition**: Temporary views allow SQL queries to be run against DataFrames.
* **Creating and Using Views**:
  + Create View: df.createOrReplaceTempView("viewName")
  + Query: spark.sql("SELECT \* FROM viewName")
* **Example**:

df.createOrReplaceTempView("employees")

val highEarners = spark.sql("SELECT \* FROM employees WHERE salary > 50000")

**Loading Files and Views into DataFrames Using Spark SQL**

* **Data Sources**: JSON, Parquet, ORC, etc.
* **Loading Data**:
  + From File: spark.read.json("path/to/file.json")
  + From SQL Table: spark.read.table("tableName")
* **Example**:

val df = spark.read.parquet("path/to/file.parquet")

df.createOrReplaceTempView("parquetTable")

**Saving to Persistent Tables + Spark 2 Known Issue**

* **Saving Data**:
  + To a Table: df.write.saveAsTable("tableName")
  + To a File: df.write.parquet("path/to/save")
* **Spark 2 Known Issue**: [Briefly describe a known issue, e.g., issues with table schema evolution or performance]
* **Example**:

df.write.mode("overwrite").saveAsTable("employees")

**Hive Support and External Databases**

* **Integration with Hive**:
  + Allows querying of Hive tables.
  + Support for Hive UDFs and custom SerDe.
* **Connecting to External Databases**:
  + Use JDBC to connect and query databases.
* **Example**:

val jdbcDF = spark.read

.format("jdbc")

.option("url", "jdbc:mysql://localhost/db")

.option("dbtable", "employees")

.option("user", "root")

.option("password", "password")

.load()

**Aggregating, Grouping, and Joining**

* **Aggregating**: Using functions like count(), sum(), avg()
* **Grouping**: Use groupBy()
* **Joining**: Different types of joins (inner, outer, etc.)
* **Example**:

val result = df.groupBy("department").agg(sum("salary").alias("total\_salary"))

**User Defined Functions (UDFs) on Spark SQL**

* **Definition**: UDFs allow you to create custom functions for use in Spark SQL.
* **Creating a UDF**:
  + Define the function.
  + Register it with Spark.
* **Example**:

val myUDF = udf((x: String) => x.toUpperCase())

spark.udf.register("toUpperCase", myUDF)

val result = spark.sql("SELECT toUpperCase(name) FROM employees")

**Summary**

* **Recap of Key Points**:
  + Spark SQL Overview and Architecture
  + Catalyst Optimizer and Plan Execution
  + ROW API and Temporary Views
  + Data Loading, Saving, and Hive Integration
  + Aggregating, Joining, and UDFs
* **Q&A**: Open the floor for questions.