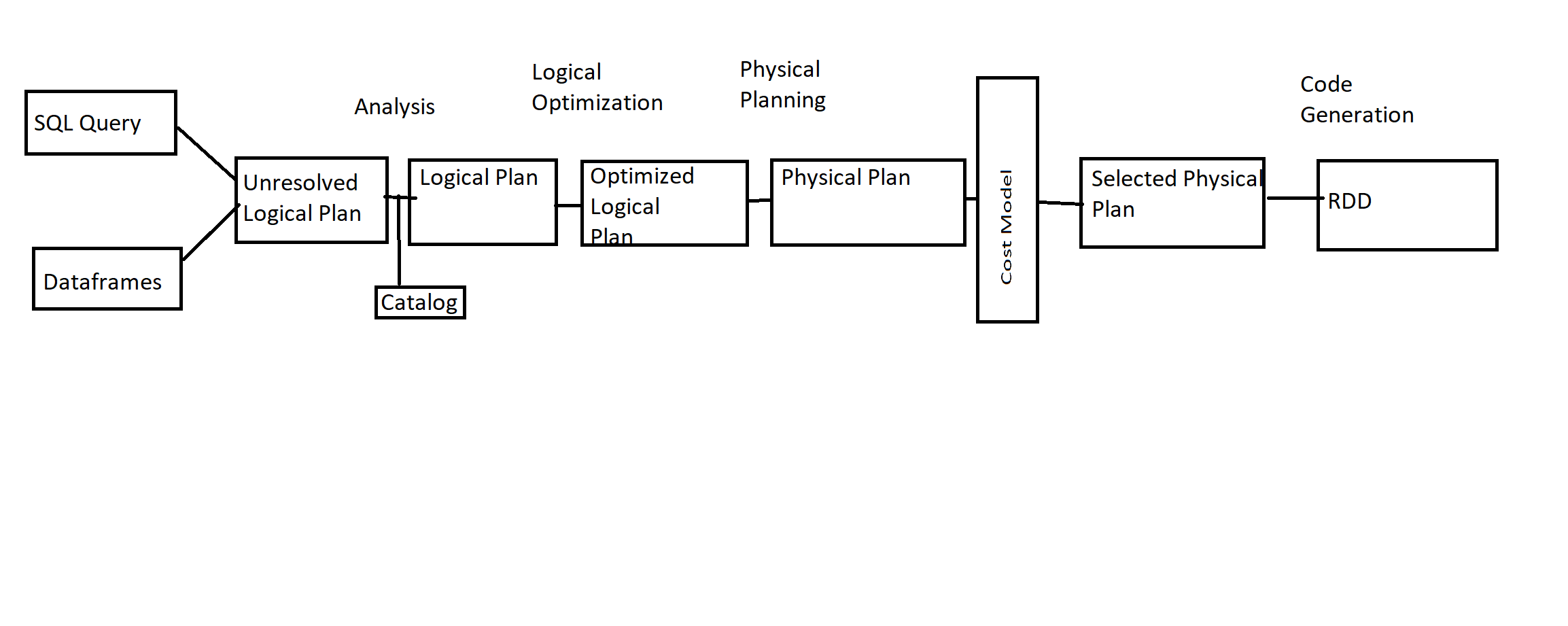
|  |  |  |
| --- | --- | --- |
| **UMBC – CSEE Department** |  | Dr. Waleed Youssef |
| **Data Science Program** |  | youssef1@umbc.edu |
| **Spring 2023** |  |  |
| **DATA 603 – Big Data Platforms** | | |
|  |  |  |
| **Homework #9 – Spark SQL** | | |
|  |  |  |

1. **[10 Points]** Explain how SQL is applied to a typical RDD? What components are needed to perform this task?



**SQL Query**: The SQL is the input query provided by user.

**Data Frame**: The SQL query is distributed into a dataframe.

**Unresolved logical plan**: The Dataframe is parsed and analyzed to create an unreolved logical plan. This plan includes logical operations that need to be performed on the data to execute the query, but does not specify how they should be implemented.

**Catalog**: It is a metadata repository that contains information about the data sources, tables, and views used in the query. It is consulted during the parsing and analysis of the query to resolve table and column names.

**Logical Plan**: The unresolved logical plan is resolved and transformed into logical plan, which specifies the operations to be performed on the data in a structured form. The logical plan is a Directed Acyclic Graph(DAG) that represents the sequence of operations needed to execute the query.

**Optimized Logical Plan**: The logical plan in optimized using a set of rules that transform the plan into more efficient form.

**Physical Plan**: The optimized logical plan is transformed into a physical plan, which specifies how operations in the plan will be executed on the cluster.

**Cost Model**: The cost model is used to estimate the cost of executing the physical paln. It considers characteristics of data, the query, and the cluster and determines the optimal way to execute the plan.

**Selected Physical Plan**: The physical plan with lowest cost is selected for execution. It is compiled into set of executable tasks that are sent to the cluster for execution.

RDD: The physical plan is executed on the cluster and the result is returned as RDD.

**Reference:** Leturgez, L. (2020, July 23). *Spark's Logical and Physical plans … When, Why, How and Beyond.* Medium. Retrieved from <https://medium.com/datalex/sparks-logical-and-physical-plans-when-why-how-and-beyond-8cd1947b605a>

1. **[10 Points]** Describe how Spark SQL supports Hive and Mongo DB? How do you connect to these data sources?

**Spark SQL with Hive:**

Spark SQL supports Apache Hive using HiveContext. It uses the Spark SQL execution engine to work with data stored in Hive.

Working with Hive requires additional configuration setup, since Hive has large number of dependencies. These dependencies are not included in the default Spark distribution. If Hive dependencies can be found on ClassPath, Spark will load them automatically. The Hive dependencies must also be present on all the worker nodes. The worker nodes need access to the Hive serialization and deserialization library(SerDe) in order to access data stored in Hive.

Configuration of Hive is done by placing the followingconfigurations files in the “conf” directory:

– hive-site.xml

– core-site.xml (for security configuration)

– hdfs-site.xml (for HDFS configuration)

Must instantiate SparkSession with Hive support towork with Hive

– We need to use spark.sql.warehouse.dir to specify the default location of database in warehouse.

Eg:

spark = SparkSession \

.builder \

.appName("Python Spark SQL Hive integration example") \ .config("spark.sql.warehouse.dir", warehouse\_location) \

.enableHiveSupport() \

.getOrCreate()

**SparkSQL with MongoDB**

When pyspark is started, a SparkSession object called Spark is created by default. For MongoDB, if the spark.mongodb.input.uri and spark.mongodb.output.uri configuration options is specified when pyspark is started, then the default SparkSessionobject uses them.

To create a SparkSession object from within pyspark, use SparkSession.builder with different options

from pyspark.sql import SparkSession

my\_spark = SparkSession \

.builder \

.appName("my MongoDB Example App") \

.config("spark.mongodb.input.uri", "mongodb://127.0.0.1/test.coll") \ .config("spark.mongodb.output.uri", "mongodb://127.0.0.1/test.coll") \

.getOrCreate()

1. **[10 Points]** Why do we need file formats like Parquet? How is data stored in Parquet?

File formats like Parquet aredesigned to optimize data storage and retrieval for big data processing systems like Apache Spark. Data is stored in columnar format which is supported by many data processing systems. Here each column is divided into pages. Pages are compressed and encoded using techniques like run-length encoding and dictionary encoding to reduce the amount of space required to store the data.

Advantages of using Parquet files are:

Columnar Storage: Parquet uses columnar storage, which means that data for each column is stored together, rather than storing the entire row. This allows for more efficient querying and processing of large datasets, as only the columns needed for a given query are read from disk.

Compression: Parquet supports different compression algorithms, which can reduce the amount of disk space required to store data.

Schema Evolution: Parquet supports schema evolution, which allows for changes to the structure of the data over time without requiring all data to be rewritten or converted. This can help to reduce downtime and improve system availability.

Apache Parquet is built from the ground up. Hence it can support advanced nested data structures, like arrays and maps, which can be stored as separate columns or as repeated fields within a column. This allows for more complex data structures to be stored and queried efficiently. The layout of Parquet data files is optimized for queries that process large volumes of data.

1. **[10 Points]** Describe DataFrames? How it is used in Spark? Explain its benefits in ML?

In Spark, a DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations. DataFrames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs. DataFrames are the primary API in Spark SQL and can be used to read and manipulate structured data from various sources like Hive tables, Parquet files, CSV files, and JSON files. DataFrames in Spark are designed to provide optimized performance and usability for data processing tasks. They support a rich set of operations and transformations like filtering, grouping, aggregation, joins, and machine learning operations.

DataFrames are used in Spark for a variety of tasks, including:

* Data exploration: DataFrames can be used to quickly and easily explore large datasets.
* Data cleaning: DataFrames can be used to clean and transform data, making it ready for analysis.
* Machine learning: DataFrames can be used to build and train machine learning models.

Benefits of DataFrames:

* They are easy to use: DataFrames can be manipulated using a variety of tools, making them easy to use for both beginners and experienced users.
* DataFrames enable easy integration with other machine learning libraries like scikit-learn and TensorFlow through Spark's MLlib API.
* They are scalable: DataFrames can be used to work with large datasets, making them ideal for machine learning tasks that require a lot of data.
* DataFrames support a wide range of data sources, including structured and semi-structured data formats, which makes it easy to incorporate various data sources into machine learning workflows.
* DataFrames provide support for various machine learning algorithms like linear regression, logistic regression, decision trees, random forests, and gradient boosting.

**Reference**: Xin, R., Armbrust, M., & Liu, D. (2015, February 17). Apache Spark DataFrames for Large Scale Data Science. Databricks. Retrieved April 19, 2023, from <https://www.databricks.com/blog/2015/02/17/introducing-dataframes-in-spark-for-large-scale-data-science.html>

1. **[20 Points]** **Spark SQL:**

Write a Spark program to load students’ data into a DataFrame(Student ID, Student Name, Student Phone Number, GPA)

**Data file:**

**543, John Smith, 301-304-4044,3.3,325, Nancy Thomas, 240-340-3444, 3.95, etc.**

Query the data to categorize students by grade (A, B, C, D, …). Output results should be something like this:

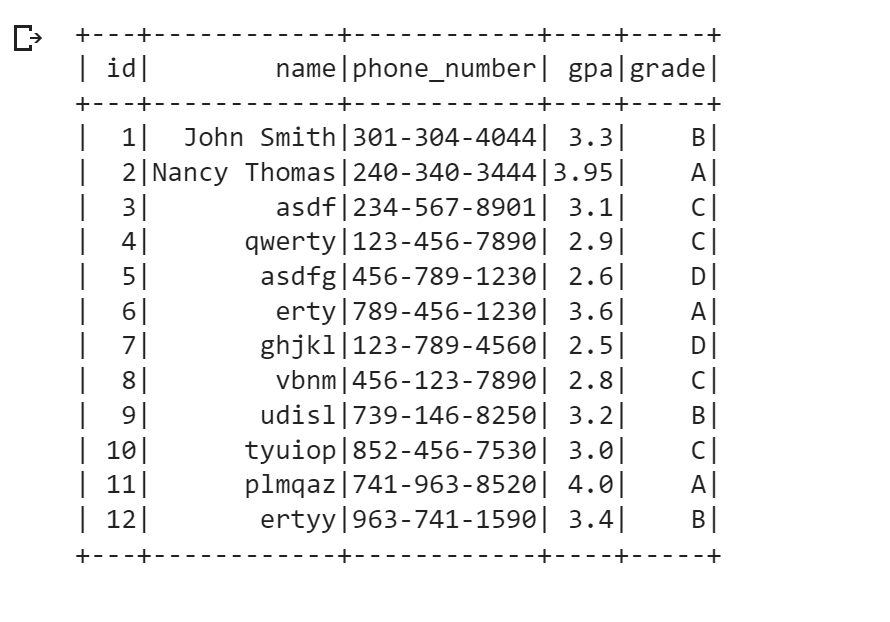
**Student ID Student Name Phone Number Grade**

**543 John Smith 301-304-4044 B**

**325 Nancy Thomas 240-340-3444 A**

**…..**

******

******

***Hints***

* *Grade A is for GPA 3.6 or higher, Grade B for GPA 3.2 to less than 3.6, Grade C for GPA 2.8 to less than 3.2.*
* *Enter at least 10 records in your data file, so your technique can be illustrated correctly.*
* *Focus on visualization and presentation of your results.*
* *You may use <<DataFrame>>.withColumn if needed*