Group B
Assignment 11

PROBLEM STATEMENT:

Create databases and tables, insert small amounts of data, and run simple queries using Impala

OBJECTIVE:

- To Learn and understand the steps to create databases and tables in Impala, insert data into these tables, and execute simple queries.
- To learn and understand how to use Impala to analyze data.

Theory:

Impalas

- A tool which we use to overcome the slowness of Hive Queries is what we call Impala.
- Syntactically Impala queries run very faster than Hive Queries even after they are more or less same as Hive Queries.
- It offers high-performance, low-latency SQL queries.
- Impala is the best option while we are dealing with medium-sized datasets and we expect the real-time response from our queries. However, make sure Impala is available only in Hadoop distribution.
- Since MapReduce store intermediate results in the file system, Impala is not built on MapReduce. Hence, it is very slow for real-time query processing.

In addition, Impala has its own execution engine. Basically, that stores the intermediate results in In-memory. Therefore, when compared to other tools which use MapReduce its query execution is very fast.

Some Key Points

- It offers high-performance, low-latency SQL queries.
- Moreover, to share databases and tables between both Impala and Hive it integrates very well with the Hive Metastore.
- Also, it is Compatible with HiveQL Syntax

Impala is an open-source SQL query engine allows you to query data stored in Apache

Hadoop clusters. It's a fast, distributed, and highly scalable system that can run complex queries on large datasets in near real-time. In this manual, we will guide you through the process of creating databases and tables, inserting small amounts of data, and running simple queries using Impala.

Impala uses a distributed architecture, which means that data is stored across multiple nodes in a cluster. Impala also uses a query engine that allows you to write SQL queries that are executed in parallel across the nodes in the cluster. This allows for fast query execution, even on large datasets.

Impala uses a metadata store to keep track of the databases and tables that are created. The metadata store is used to store information about the location of the data, the schema of the tables, and other metadata. Impala supports a wide range of data formats, including Parquet, Avro, and RCFile.

Data Types:

BIGINT :- This datatype stores numerical values and the range of this data type is -9223372036854775808 to 9223372036854775807. This datatype is used in create table and alter table statements.

BOOLEAN:-This data type stores only true or false values and it is used in the column definition of create table statement.

CHAR:- This data type is a fixed length storage, it is padded with spaces, you can store up to the maximum length of 255.

DECIMAL:- This data type is used to store decimal values and it is used in create table and alter table statements.

DOUBLE:- This data type is used to store the floating point values in the range of positive or negative 4.94065645841246544e-324d -1.79769313486231570e+308.

FLOAT :- This data type is used to store single precision floating value datatypes in the range of positive or negative 1.40129846432481707e-45 .. 3.40282346638528860e+38.

Example:

To create a database in Impala, you can use the following SQL command:

```
CREATE DATABASE my_database;
```

This will create a new database called "my_database". To create a table within this database, you can use the following SQL command:

```
CREATE TABLE my_table ( id INT,   name STRING,  age INT );
```

This will create a new table called "my_table" with three columns: id, name, and age. To insert data into this table, you can use the following SQL command:

INSERT INTO my_table VALUES (1, 'John', 25), (2, 'Jane', 30), (3, 'Bob', 40);

This will insert three rows into the table, each with a unique id, name, and age. To run a simple query on this table, you can use the following SQL command:

SELECT * FROM my_table;

This will return all the rows in the table. You can also use more complex queries, such as aggregations and joins, to analyze the data in the table.

Conclusion:

Impala is a powerful SQL query engine that can be used to analyze large datasets stored in Apache Hadoop clusters. We have seen how to create databases and tables, insert data, and execute simple queries in Impala.

Group B

Assignment No: 12

Problem Statemen:

Write a simple program in SCALA using Apache Spark framework

Theory:

- Scala is an acronym for “Scalable Language”.
- It is a general-purpose programming language designed for the programmers who want to write programs in a concise, elegant, and typesafe way.
- Scala enables programmers to be more productive. Scala is developed as an object-oriented and functional programming language.
- If you write a code in Scala, you will see that the style is similar to a scripting language.
- Even though Scala is a new language, it has gained enough users and has a wide community support. It is one of the most user-friendly languages.

About Scala

- Scala is pure Object-Oriented programming language
- Scala is an object-oriented programming language.
- Everything in Scala is an object and any operations you perform is a method call.
- Scala, allow you to add new operations to existing classes with the help of implicit classes.
- One of the advantages of Scala is that it makes it very easy to interact with Java code.
- You can also write a Java code inside Scala class.
- The Scala supports advanced component architectures through classes and traits.

Scala is a functional language

- Scala is a programming language that has implemented major functional programming concepts.
- In Functional programming, every computation is treated as a mathematical function which avoids states and mutable data.
- The functional programming exhibits following characteristics:
- Power and flexibility
- Simplicity
- Suitable for parallel processing

- Scala is not a pure functional language. Haskell is an example of a pure functional language.

Scala is a compiler based language (and not interpreted)

- Scala is a compiler based language which makes Scala execution very fast if you compare it with Python (which is an interpreted language).
- The compiler in Scala works in similar fashion as Java compiler. It gets the source code and generates Java byte-code that can be executed independently on any standard JVM (Java Virtual Machine).
- There are more important points about Scala which I have not covered. Some of them are:
- Scala has thread based executors
- Scala is statically typed language
- Scala can execute Java code
- You can do concurrent and Synchronized processing in Scala
- Scala is JVM based languages

Installing Scala

- Scala can be installed in any Unix or windows-based system. Below are the steps to install for Ubuntu (14.04) for scala version 2.11.7.
- I am showing the steps for installing Scala (2.11.7) with Java version 7. It is necessary to install Java before installing Scala. You can also install latest version of Scala(2.12.1) as well. Step 0: Open the terminal Step 1: Install Java
- $ sudo apt-add-repository ppa:webupd8team/java $ sudo apt-get update $ sudo apt-get install oracle-java7installer
- If you are asked to accept Java license terms, click on “Yes” and proceed. Once finished, let us check whether Java has installed successfully or not. To check the Java version and installation, you can type:
- $ java -version
- Step 2: Once Java is installed, we need to install Scala
- $ cd ~/Downloads $ wget http://www.scala-lang.org/files/archive/scala-2.11.7.deb $ sudo dpkg -i scala2.11.7.deb $ scala –version

Scala Basics Terms

- Object: An entity that has state and behavior is known as an object. For example: table, person, car etc.
- Class: A class can be defined as a blueprint or a template for creating different objects which defines its properties and behavior.
- Method: It is a behavior of a class. A class can contain one or more than one method. For example: deposit can be considered a method of bank class.
- Closure: Closure is any function that closes over the environment in which it’s defined. A closure returns value depends on the value of one or more variables which is declared outside this closure.
- Traits: Traits are used to define object types by specifying the signature of the supported methods. It is like interface in java.

Variable declaration in Scala

- In Scala, you can declare a variable using ‘var’ or ‘val’ keyword. The decision is based on whether it is a constant or a variable.
- If you use ‘var’ keyword, you define a variable as mutable variable. On the other hand, if you use ‘val’, you define it as immutable. Let’s first declare a variable using “var” and then using “val”.

  Declare using var

  var Var1 : String = "Ankit"

- In the above Scala statement, you declare a mutable variable called “Var1” which takes a string value. You can also write the above statement without specifying the type of variable. Scala will automatically identify it. For example:

  var Var1 = “Joshi"

Operations on Variables

- You can perform various operations on variables. There are various kinds of operators defined in Scala. For example: Arithmetic Operators, Relational Operators, Logical Operators, Bitwise Operators, Assignment Operators.
- Lets see “+” , “==” operators on two variables ‘Var4’, “Var5”. But, before that, let us first assign values to “Var4” and “Var5”.

  scala> var Var4 = 2 Output: Var4: Int = 2 scala> var Var5 = 3

  Output: Var5: Int = 3

  Now, let us apply some operations using operators in Scala.

  Apply ‘+’ operator Var4+Var5

  Output: res1: Int = 5 Apply “==” operator

  Var4==Var5 Output: res2: Boolean = false

- In Scala, if-else expression is used for conditional statements.
- YouThe ifcan write-oneelse expression in Scalaor more conditions inside “if”.
- Let’s declare a variable called “Var3” with a value 1 and then compare “Var3” using if-else expression.

- In the above snippet, the condition evaluates to True and hence True will be printed in the output.

Iteration in Scala

Like most languages, Scala also has a FOR-loop which is the most widely used method for iteration. It has a simple syntax too.

- Declare a simple function in Scala and call it by passing value
- You can define a function in Scala using “def” keyword. Let’s define a function called
- “mul2” which will take a number and multiply it by 10.
- You need to define the return type of function, if a function not returning any value you should use the “Unit” keyword.
- In the below example, the function returns an integer value. Let’s define the function

“mul2”:

def mul2(m: Int): Int = m * 10

Output: mul2: (m: Int)Int

Example:

Here is an example of a simple Scala program that prints "Hello, World!" to the console:

```
object HelloWorld {
  def main(args: Array[String]) {
    println("Hello, World!")
  }
}
```

In this program, we define an object called "HelloWorld". Inside the object, we define a method called "main" that takes an array of strings as an argument. The method simply prints the string "Hello, World!" to the console using the println() method.

To compile and run this program on Ubuntu, you will need to follow these steps:

Install the Scala compiler by running the following command in the terminal:

sudo apt-get install scala

Save the above code as "HelloWorld.scala" in a directory of your choice.

Open a terminal in the directory where you saved the file and run the following command to compile the code:

scalac HelloWorld.scala

This will generate a class file called "HelloWorld.class" in the same directory.

Run the program by entering the following command in the terminal:

scala HelloWorld

This will execute the "main" method of the "HelloWorld" object and print "Hello, World!" to the console.

Assignment Questions

1. What are the features of scala?
2. Explain the program execution steps in scala