**NAME: Vedashree Bhalerao**

**CLASS: AIA-1**

**ROLL NO:2213191**

**BATCH: B**

# Big Data Analytics Experiment no. 04

**Aim:**  1) PySpark - Read CSV file into Data Frame

2) Create and query a HIVE table in PySpark.

**Theory:**

**PySpark Overview:**

PySpark is the Python API for Apache Spark, an open-source, distributed computing system. It provides a programming interface for entire clusters with implicit data parallelism and fault tolerance. PySpark allows you to write Spark applications using Python.

**PySpark DataFrame:**

A DataFrame is a distributed collection of data organized into named columns, similar to a table in a relational database or a data frame in R or Python (with pandas). It is one of the most commonly used abstractions in Spark. **Hive and Hive Tables:**

Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis. It provides an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop.

**Why Use PySpark with Hive?**

* **Scalability**: Handle large datasets efficiently.
* **SQL Compatibility**: Use familiar SQL queries for data processing.
* **Integration**: Easily integrate with existing Hadoop ecosystems.

**Program Details: Prerequisites:**

1. **Apache Spark**: Installed and configured.
2. **Hadoop and Hive**: Installed and configured if using a Hadoop-based deployment.
3. **Python**: Ensure Python 2.7 or 3.4 and later versions are installed.

**Steps:**

***Step 1: PySpark - Read CSV File into DataFrame***

1. **Initialize SparkSession**:

from pyspark.sql import SparkSession spark = SparkSession.builder \ .appName("ReadCSV") \

.getOrCreate()

1. **Read CSV File**:

df = spark.read.csv("path/to/your/csvfile.csv", header=True, inferSchema=True)

1. **Display DataFrame**:

**df.show()**

1. **Print Schema**:

df.printSchema()

*Step 2: Create and Query a Hive Table in PySpark*

1. **Enable Hive Support in SparkSession**: spark = SparkSession.builder \ .appName("HiveExample") \

.enableHiveSupport() \

.getOrCreate()

1. **Create Hive Database (if not exists)**:

spark.sql("CREATE DATABASE IF NOT EXISTS mydb") spark.sql("USE mydb")

1. **Create Hive Table**: spark.sql("""

CREATE TABLE IF NOT EXISTS my\_table ( id INT,

name STRING,

age INT

) ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

""")

1. **Load Data into Hive Table**:

spark.sql("LOAD DATA LOCAL INPATH 'path/to/your/csvfile.csv' INTO TABLE my\_table")

1. **Query Hive Table**:

result = spark.sql("SELECT \* FROM my\_table") result.show()

**Conclusion:**

In this lab assignment, we have learned how to perform two essential tasks using PySpark:

1. **Reading a CSV File into a DataFrame**:
   * You initialized a SparkSession and read a CSV file into a DataFrame.
   * You displayed the contents and schema of the DataFrame.
2. **Creating and Querying a Hive Table**:
   * You enabled Hive support in SparkSession and created a Hive database and table. o You loaded data from a CSV file into the Hive table and performed SQL queries on the table.

These skills are fundamental for working with large datasets in a distributed computing environment, allowing you to efficiently process and analyze data using the power of Apache Spark and Hive. By integrating these technologies, you can leverage the scalability of Spark and the SQL capabilities of Hive, making your data processing tasks more effective and streamlined.

# Code :-

import pandas as pd def read\_csv\_file(file\_path): return pd.read\_csv(file\_path)

df = read\_csv\_file('C:/Users/Hp/Documents/ds\_salaries.csv')

print(df.head())

# Example of reading a text file as input df\_txt = pd.read\_csv('C:/Users/Admin/Desktop/crow.txt', delimiter=' ') print(df\_txt.head()) df\_txt.head()

# OUTPUT:-





