Name:Sahil Motiramani Div:D15C Roll No:35

Experiment 9

Aim: To perform Exploratory data analysis using Apache Spark and Pandas

Theory:

1. What is Apache Spark and How It Works?

Ans:**Apache Spark** is a powerful, open-source distributed computing platform designed for processing large-scale data efficiently. Unlike the traditional Hadoop MapReduce model, Spark performs in-memory computations, making it significantly faster, especially for repetitive tasks like data analysis, machine learning, and graph processing.

Core Components of Apache Spark:

- **Spark Core**: The main engine responsible for essential distributed task execution.
- **Spark SQL**: A module used for structured data handling through SQL queries and DataFrames.
- **MLlib**: Spark's scalable machine learning library offering various algorithms and utilities.
- **GraphX**: A specialized API for graph-based computations.
- **Spark Streaming**: Designed for processing real-time data streams.

Working of Apache Spark:

- Spark operates on **RDDs** (Resilient Distributed Datasets) or **DataFrames** to represent and manipulate data.
- The **Driver Program** creates a **SparkContext**, which acts as the coordinator and connects to a **Cluster Manager**.
- Spark distributes work to multiple **Executors**, which process tasks in parallel across the cluster.
- Spark supports **lazy evaluation**, meaning transformations are only executed when an action (like .count() or .collect()) is triggered.

Name:Sahil Motiramani Div:D15C Roll No:35

2. How Data Exploration is Done in Apache Spark?

Exploratory Data Analysis (EDA) in Spark follows the same goals as in pandas—understanding and summarizing the dataset—but is built to scale across distributed systems for very large datasets.

Steps for Performing EDA in Spark:

1. **Set Up Spark**:
   Start by importing PySpark and creating a **SparkSession** via SparkSession.builder. This session is your main entry point to all Spark features.

2. **Load Data**:
   Use spark.read.csv() or spark.read.json() to load datasets into **Spark DataFrames**. Enable header=True and inferSchema=True for cleaner data ingestion.

3. **Examine Data Structure**:
   - .printSchema() shows column types.
   - .show() gives a quick snapshot of data rows.
   - .describe() provides summary stats like average, minimum, and maximum.

4. **Manage Missing Data**:
   Use methods like df.na.drop() to discard null rows or df.na.fill("value") to replace them with default values.

5. **Data Transformation**:
   Use methods like .withColumn(), .filter(), .groupBy() to reshape, filter, and summarize the data effectively.

6. **Visualizations**:
   Convert the Spark DataFrame to a pandas DataFrame using .toPandas() and visualize using **matplotlib** or **seaborn**.

7. **Correlation & Insights**:
   Use corr() in pandas or Correlation.corr() from MLlib to find relationships between variables.
   Use grouping and pivoting to analyze patterns and draw useful insights.

Name:Sahil Motiramani Div:D15C Roll No:35

Conclusion:

In this task, I gained hands-on experience with **Exploratory Data Analysis (EDA)** using Apache Spark alongside Python libraries like pandas. I learned how to create a SparkSession, efficiently load large datasets, and examine their structure using key Spark functions like .show(), .printSchema(), and .describe(). I also explored techniques for cleaning data, performing transformations, and visualizing results after converting Spark DataFrames to pandas. Lastly, I learned how to extract correlations and insights from the data using grouping and aggregation. This experiment enhanced my understanding of Spark’s ability to handle massive data workloads and its compatibility with traditional data science tools for in-depth analysis.