Experiment 9

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

Theory:

1. What is Apache Spark and How It Works?

Apache Spark is an **open-source distributed computing framework** designed for big data processing, faster than traditional Hadoop MapReduce. It enables **in-memory computation**, making operations much quicker for iterative tasks like machine learning, data analysis, and graph processing.

Key Components of Apache Spark:

- **Spark Core**: The base engine for large-scale parallel data processing.
- **Spark SQL**: Module for structured data processing using DataFrames and SQL.
- **MLlib**: Machine Learning library for scalable learning algorithms.
- **GraphX**: For graph computations.
- **Spark Streaming**: For real-time stream data processing.

How Spark Works:

- Spark processes data in **RDDs (Resilient Distributed Datasets)** or **DataFrames**.
- The **Driver Program** initiates a SparkContext, connecting to a **Cluster Manager**. ● Tasks are distributed across **Executors** for parallel execution.
- Supports **lazy evaluation**—transformations are only computed when an action is called.

2. How Data Exploration is Done in Apache Spark?

EDA in Apache Spark follows similar principles to pandas but is designed to scale to massive datasets across clusters.

Steps of EDA in Spark:

1. Initialization:

Import pyspark and create a SparkSession using SparkSession.builder.
This session acts as the entry point to Spark functionalities.

2. Load Dataset:

Use spark.read.csv() or .json() to load data into a Spark DataFrame.
Enable header=True and inferSchema=True for cleaner loading.

3. Understand Data Schema:

Use .printSchema() to view column types and .show() for a data preview. .describe() provides summary statistics like mean, min, and max.

4. Handle Missing Values:

Use df.na.drop() to remove nulls or df.na.fill("value") to fill them.
This step is crucial to clean data for accurate analysis.

5. Data Transformation:

Apply .withColumn(), .filter(), .groupBy() to reshape and summarize data.
These functions help in refining the dataset before analysis.

6. Data Visualization:

Convert Spark DataFrame to Pandas using .toPandas() for plotting.
Then visualize with tools like matplotlib or seaborn.

7. Correlation & Insights:

Use .corr() in Pandas or MLlib's Correlation.corr() for relationships.
Group, pivot, and analyze data patterns for meaningful insights.

Conclusion:

In this experiment, I learned how to perform Exploratory Data Analysis using Apache Spark and Pandas. I understood how to initialize a SparkSession, load large datasets efficiently, and explore their structure using Spark functions like .show(), .printSchema(), and .describe(). I also learned how to handle missing values, transform data using Spark DataFrame operations, and convert data to Pandas for visualization. Additionally, I explored how to compute correlations and derive insights through grouping and aggregation. This experiment helped me grasp the scalability and power of Spark in handling big data and how it complements traditional Python libraries like Pandas and Seaborn for insightful data analysis.