Name: Divesh Lulla **Class: D15_C** **Roll no: 31**

Experiment No: 9

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

Theory -

1. What is Apache Spark, and how does it work?

Apache Spark is a fast, in-memory distributed computing framework designed for large-scale data processing. It is open-source and built to support batch as well as real-time analytics through a unified engine. Spark provides libraries and APIs for a range of tasks, including:

- Spark SQL – for structured data queries
- Spark Streaming – for real-time stream processing
- MLlib – for machine learning algorithms
- GraphX – for graph computations

Working:

- Spark breaks down large datasets into smaller pieces called partitions, which it processes in parallel across multiple machines. This makes it fast and efficient for handling big data.
- At the core of Spark is a smart data structure called an RDD (Resilient Distributed Dataset). It helps Spark manage data safely and reliably; even if something goes wrong on one of the machines, your data is still safe.
- Unlike older systems like Hadoop MapReduce, which write temporary results to disk after every step (which slows things down), Spark tries to keep as much data as possible in memory (RAM). This means tasks get done a lot faster.
- You can write Spark programs in several languages—Python (using PySpark), Scala, Java, or R—which makes it super flexible and accessible whether you're a data engineer or a data scientist.

2. How is Data Exploration Done in Apache Spark? Explain Steps.

Data exploration in Apache Spark is a key step in understanding and preparing your data for further analysis or machine learning. It involves loading, inspecting, and performing basic operations on the data to uncover patterns, detect anomalies, or test hypotheses. Below are the main steps typically involved in data exploration using Spark (especially with PySpark, the Python API for Spark):

Load the Data: First, bring your data into Spark. You can load it from various sources like CSV or JSON files, databases, or even cloud storage like AWS S3 or HDFS.

Check the Schema: Use .printSchema() to see the column names, data types (string, int, etc) and whether any columns can be empty.

Peek at the Data: Use .show(n) to display the first few rows. This gives you a quick look at what kind of values you're dealing with—perfect for spotting issues like weird formatting or unexpected entries

Get Some Stats of the data: Use .describe() to get basic statistics like mean, min, max, count, and standard deviation for each column.

Column-Wise Inspection: To closely examine specific parts of the data, the .select() function is used to focus on individual or multiple columns. For filtering rows based on certain conditions—such as missing values, threshold limits, or outliers—.filter() or .where() functions are applied. These tools help isolate relevant data and perform targeted inspections efficiently.

Grouping and Aggregations: Spark provides `.groupBy()` and `.agg()` functions to perform aggregation operations like counting, averaging, or summing values across grouped categories. This step is essential for identifying patterns, summarizing data by category, and uncovering trends within the dataset.

Handling Missing or Anomalous Data: Data cleaning is done using functions like `.dropna()` to remove rows with missing values or `.fillna()` to replace them with default or calculated values. Conditional logic and filters can be applied to detect and handle anomalies or outliers, ensuring the dataset is clean and ready for further processing or modeling.

Conclusion:

Apache Spark is a high-performance tool for big data analytics, combining the speed of in-memory computation with the scalability of cluster processing. It simplifies exploratory data analysis through its intuitive DataFrame API, robust support for distributed systems, and integrations with other tools like Pandas and visualization platforms. By leveraging Spark's capabilities, data professionals can efficiently clean, analyze, and derive actionable insights from massive datasets—laying the groundwork for successful data-driven decision-making and machine learning applications.