NAME: SHIVAM MAKHIJA ROLL NO:32 D15C

Experiment No: 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 system designed for fast and scalable big data processing. It provides a unified engine for working with large-scale data through modules like Spark SQL, Spark Streaming, MLlib (machine learning), and GraphX (graph processing). Spark is known for its **in-memory processing**, which allows it to perform operations much faster than traditional systems like Hadoop MapReduce that rely heavily on disk storage.

Spark works by dividing data into small chunks called **partitions**, which are processed in parallel across a cluster of machines. It uses a concept called **RDD (Resilient Distributed Dataset)** to manage data fault-tolerantly across nodes. Users write Spark applications in languages like Python, Scala, or Java, and Spark's engine takes care of scheduling tasks, distributing data, and recovering from failures. This makes it highly efficient for processing both batch and real-time streaming data across massive datasets.

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

Data exploration in Apache Spark is the process of understanding, summarizing, and preparing data for further analysis or machine learning. Spark provides powerful APIs (like PySpark for Python) that allow users to explore large datasets efficiently. The process typically starts by **loading the dataset** into a DataFrame from sources such as CSV, JSON, databases, or distributed file systems like HDFS or AWS S3.

Once the data is loaded, users perform **basic exploratory steps** such as checking the schema with `.printSchema()`, viewing sample data using `.show()`, and understanding data types. Spark also offers functions like `.describe()` to generate summary statistics (mean, count, stddev, min, max), `.select()` to view specific columns, and `.filter()` or `.where()` to inspect rows with specific conditions. Missing values and outliers can be identified using conditional queries and aggregate functions.

In deeper exploration, Spark enables the use of **grouping and aggregation operations** (`groupBy`, `agg`) to analyze trends and patterns across categories. Visualization can be done by exporting processed subsets of data into tools like Pandas or external BI dashboards. Overall, Apache Spark allows scalable and interactive exploration of big data, making it ideal for both data engineers and data scientists working on large datasets.

Conclusion:

Apache Spark serves as a powerful platform for both big data processing and exploration, offering a unified engine that supports batch and real-time analytics through its in-memory computing capabilities. Its ability to handle large datasets efficiently makes it ideal for data exploration, where users can load, inspect, summarize, and analyze data using intuitive APIs. With features like distributed processing, fault tolerance, and support for multiple data sources and formats, Spark empowers users to gain meaningful insights and prepare data at scale, setting a strong foundation for advanced analytics and machine learning.