Name:Sahil Ramrakhyani **D15C** **Roll No:42**

EXPERIMENT-9

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

Theory:

1. What is Apache Spark and it works?

Apache Spark is an open-source, distributed computing framework designed for fast and scalable processing of large datasets. It works by dividing big data tasks into smaller chunks and executing them in parallel across multiple nodes in a cluster, which significantly boosts performance. One of Spark's key strengths is its ability to perform in-memory computation, meaning it stores intermediate data in RAM rather than writing it to disk, resulting in much faster processing compared to traditional tools like Hadoop MapReduce. At the core of Spark is the concept of Resilient Distributed Datasets (RDDs), which are fault-tolerant collections of data that can be operated on in parallel. Spark also supports lazy evaluation, where operations are not executed until a final action is called, allowing for optimized execution plans. It includes a rich set of libraries for various data tasks, including Spark SQL for structured data, Spark Streaming for real-time data, MLlib for machine learning, and GraphX for graph processing. Spark runs on different cluster managers such as YARN, Mesos, Kubernetes, or its own standalone manager, making it highly flexible for deployment. Overall, Apache Spark is a powerful tool for big data analytics, offering speed, versatility, and ease of use for handling complex data workloads.

2. How data exploration done in Apache spark? Explain steps.

Data exploration in Apache Spark involves examining and understanding a dataset to uncover patterns, spot anomalies, check assumptions, and gain insights before performing any advanced analysis or modeling. Spark, especially with Spark SQL and DataFrames, makes this process efficient and scalable for large datasets. Here's how data exploration is typically done in Apache Spark, explained step-by-step:

1. **Load the Data:**
    - First, the dataset is loaded into a DataFrame using Spark’s built-in functions.
2. **View Schema and Structure:**
    - Check the structure of the dataset to understand data types and columns.
3. **Preview the Data:**
    - Display a few rows to get a feel for the content.

4. **Check for Missing or Null Values:**
    - Identify missing data in each column.
5. **Summary Statistics:**
    - Get descriptive statistics (count, mean, min, max, etc.) for numerical columns.
6. **Check Unique Values / Distinct Counts:**
    - Useful for categorical columns.
7. **Filter and Query Data:**
    - Use SQL or DataFrame operations to filter and explore subsets of the data.
8. **Group and Aggregate:**
    - Analyze trends or summaries using group-by and aggregation.
9. **Correlation and Relationships:**
    - Analyze relationships between features (e.g., correlation).
10. **Data Visualization:**
    - Spark doesn't directly support plotting, but results can be collected into Pandas for visualization using libraries like Matplotlib or Seaborn

Conclusion:

In conclusion, Apache Spark proves to be a robust and scalable framework for handling large-scale data processing tasks with high efficiency. Its ability to perform in-memory computation, distributed processing, and support for multiple libraries makes it a versatile tool for data science and big data analytics. When it comes to data exploration, Spark offers powerful functionalities through its DataFrame and SQL APIs, allowing users to efficiently inspect, clean, and analyze large datasets. From loading and understanding the data structure to performing aggregations and identifying trends, Spark simplifies the exploration process and sets a strong foundation for further analysis and decision-making. Together, Spark’s processing power and exploration capabilities make it an invaluable tool in modern data workflows.