Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

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?
Answer:
Introduction to Apache Spark
Apache Spark is an open-source, unified analytics engine specifically built for processing large volumes of data at high speed. Originally developed at UC Berkeley, Spark quickly gained popularity due to its ability to handle complex data tasks with ease and efficiency.

It is a big data tool used in situations where traditional tools (like Excel, Pandas, or SQL) are not enough — for example, when working with gigabytes or terabytes of data, or when data is stored across multiple systems like cloud servers, Hadoop clusters, or databases.

Apache Spark:

1. **Category:** Big Data Processing Engine
2. **Use Case:** Handles large datasets that exceed single-machine memory
3. **Working:** Executes tasks in parallel across a cluster
4. **Strengths:** Scalable, fast (in-memory), fault-tolerant

How Apache Spark Works
Imagine you're organizing a massive event with thousands of tasks to do. If one person tries to handle everything, it's slow and inefficient. But if you divide tasks among a team, each person can handle a portion, and things get done much faster. That's exactly what Apache Spark does with your data.

- **Cluster-Based Architecture**
  Spark works on a cluster, which is a group of machines. One machine acts as the driver (manager) and the rest are workers (task doers). The driver coordinates the tasks, and the workers process chunks of data in parallel.

- **RDD (Resilient Distributed Dataset)**
  Spark's basic unit of data is called an RDD. It's like a big list that's split and stored across multiple computers. Each piece of the RDD can be worked on separately without affecting the others, which is why it's so fast and fault-tolerant.

- **In-Memory Computation**
  Unlike other tools that constantly write and read from hard drives, Spark processes data in memory (RAM). This makes it incredibly fast — often 10 to 100 times faster

than traditional big data tools like Hadoop MapReduce.

- **Lazy Evaluation**
  Spark doesn't immediately perform operations as they're written. It waits until the final result is needed and then runs the most optimized path to get there. This avoids unnecessary computations.

- **Rich API Support**
  Spark supports multiple programming languages — Python (PySpark), Scala, Java, and R — making it widely accessible for data analysts and engineers.

Use Cases

1. **Social Media Analysis:** This involves collecting and analyzing massive amounts of posts, comments, and interactions to identify trending topics, popular hashtags, and user sentiment in real-time.
2. **Retail Sales Insights:** EDA can help uncover patterns in customer purchasing behavior, such as peak buying times, popular products, and customer segmentation, using transaction-level data.
3. **IoT Sensor Data Monitoring:** Spark processes real-time data coming from distributed sensors to detect anomalies, monitor performance, and alert in case of unusual patterns.
4. **Financial Transactions:** Huge volumes of transaction data can be analyzed to detect fraudulent activities, such as unauthorized transfers or abnormal spending behaviors.
5. **Healthcare Analytics:** Patient data, including medical histories and treatment responses, can be explored to improve diagnoses, predict outcomes, and personalize care.

2. How is Data Exploration Done in Apache Spark?
Answer:
Exploratory Data Analysis (EDA) is the first step in any data science or machine learning project. It's like getting to know your data — understanding what's inside, spotting errors, identifying trends, and deciding how to clean or transform it.

With Spark, EDA can be done on huge datasets that don't fit into memory using a distributed approach. Here's a deeper breakdown of how EDA is carried out using Apache Spark:

Steps To perform EDA On Apache Spark

Step 1: Initializing Spark Session
Initializing Spark Session then A SparkSession is started, which serves as the entry point for working with DataFrames and datasets in Spark.

Step 2: Reading and Loading Data
Data is loaded from various formats (CSV, JSON, Parquet, etc.) into Spark DataFrames. These DataFrames are distributed collections of data, similar to tables.

Step 3: Data Inspection and Schema Exploration
We examine the structure of the data using commands to display column names, data types, row counts, and sample records.

Step 4: Data Cleaning
This involves handling missing values, fixing data types, removing duplicates, and filtering out invalid records. This step ensures that the data is accurate and usable.

Step 5: Descriptive Statistics
We calculate summary statistics like mean, median, standard deviation, min, and max values to understand the distribution of data.

Step 6: Grouping and Aggregation
Data is grouped by categories and aggregated to analyze trends and patterns across different segments.

Step 7: Filtering and Sorting
Specific subsets of the data are extracted by applying filter conditions and sorting values for deeper insights.

Step 8: Data Sampling and Conversion
For visualization or detailed analysis, a small sample of the Spark DataFrame is converted into a Pandas DataFrame.

Conclusion:

This experiment aimed to understand how Exploratory Data Analysis (EDA) can be performed using both Apache Spark and Pandas. Spark, with its distributed and in-memory processing, is ideal for analyzing large-scale datasets, while Pandas is better suited for smaller, quick analysis tasks. We explored the step-by-step EDA process—loading, inspecting, cleaning, summarizing, and analyzing data. Each tool serves a specific purpose, and together they offer flexibility and efficiency across data sizes. Understanding how these tools work prepares data professionals to handle diverse analytical challenges with the right approach.