Name: Shreyash Kamat Div: D15C Roll no: 22

Experiment 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 an open-source, distributed computing system designed for big data processing and analytics. It supports programming languages like Python, Scala, Java, and R. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

Key components and working:

- **Driver Program** : Coordinates the execution of Spark applications by creating SparkContext and handling task scheduling.
- **Cluster Manager** : Allocates resources to the Spark applications across the cluster.
- **Executors** : Run on worker nodes to execute tasks assigned by the driver.
- **RDD (Resilient Distributed Dataset)** : A fault-tolerant collection of elements that can be operated on in parallel.
- **Lazy Evaluation** : Spark doesn't execute transformations until an action is triggered, which optimizes execution.

Spark also includes high-level APIs for structured data processing via **DataFrames** and **Spark SQL** , enabling SQL-like queries on distributed datasets. It integrates well with other big data tools and supports various data sources like HDFS, Cassandra, HBase, and Amazon S3. Spark's ability to process data in-memory significantly reduces disk I/O, making it much faster than traditional frameworks like Hadoop MapReduce. This speed, combined with scalability and ease of use, makes Spark ideal for data exploration, machine learning, and real-time analytics on large datasets.

2. How is data exploration done in Apache Spark?

Step 1: Loading the Data

Use `spark.read.csv()` or similar methods to load data from different sources such as CSV, JSON, or Parquet.

Step 2: Inspecting the Schema

Use `.printSchema()` or `.dtypes` to view the structure and data types of each column.

Step 3: Viewing Sample Data

Use `.show()` or `.head()` to look at a few initial records and get an idea of the data.

Step 4: Summary Statistics

Apply `.describe()` or `.summary()` to generate basic statistics like mean, standard deviation, min, max, and count.

Step 5: Checking for Missing Values

Use `.filter()` or `.where()` with `.isNull()` to identify null or missing entries.

Step 6: Counting Unique Values

Use `.groupBy(column).count()` to see frequency distributions of categorical or numerical features.

Step 7: Correlation Analysis

Use `.corr()` method to measure the linear correlation between numerical variables.

Step 8: Visualization

Since Spark doesn't support advanced visualizations directly, convert Spark DataFrames to Pandas using `.toPandas()` and use libraries like Matplotlib or Seaborn for plotting.

Conclusion:

Apache Spark is a powerful tool for performing EDA on large datasets thanks to its distributed architecture and in-memory processing. When combined with Pandas and

Python’s visualization libraries, it becomes possible to efficiently analyze and visualize even large datasets. Spark enables scalable, fast, and interactive exploration that helps identify data quality issues, trends, and patterns before modeling. It supports parallel processing, making it ideal for handling large-scale data without performance bottlenecks. Overall, Spark greatly enhances the EDA process by accelerating insights and enabling more informed decision-making.