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

Experiment No: 10

Aim: To perform Batch and Streamed Data Analysis using Apache Spark.

Theory:

1. What is Streaming? Explain Batch and Stream Data.

Streaming is a data processing technique where data is continuously generated, transmitted, and processed in real time or near real time. This method is essential in scenarios where data is produced constantly and decisions or actions need to be taken immediately. Examples include live video broadcasting, online gaming, financial market analysis, social media feeds, and sensor data in IoT systems. Streaming enables organizations to react quickly to new information, detect anomalies, monitor activities, or provide real-time analytics and insights.

In contrast, data processing generally falls into two main categories: batch processing and stream processing. Batch data refers to data that is collected over a period of time and processed all at once in a scheduled, delayed manner. It is often used for tasks where real-time processing isn't necessary, such as generating monthly sales reports, performing payroll processing, or running complex data transformations in data warehouses. Batch processing is efficient for handling large volumes of data and is typically simpler and less resource-intensive to implement.

Stream data, on the other hand, consists of individual records or events that are generated continuously. This data is processed incrementally, often within milliseconds or seconds of being produced. Stream processing is critical for real-time use cases like fraud detection in banking, monitoring user behavior on websites, managing logistics in supply chains, or processing logs and telemetry from cloud services. It allows for continuous computation and immediate insights, but it can be more complex to design and requires systems capable of handling and scaling with fast data flows.

Overall, the choice between batch and stream processing depends on the specific business needs. While batch is suitable for retrospective analysis and high-throughput jobs, streaming is ideal for time-sensitive, real-time decision-making processes. In modern data architectures, many systems use a combination of both to take advantage of their respective strengths.

2. How Data Streaming Takes Place Using Apache Spark.

Apache Spark Streaming enables real-time data processing by using a model called micro-batching, where incoming data streams are divided into small batches and processed at regular time intervals. This allows Spark to apply its powerful batch processing capabilities to live data, giving the effect of real-time streaming. The data can be ingested from various sources such as Apache Kafka, Flume, Amazon Kinesis, or even TCP sockets. Once received, Spark treats each batch as a Resilient Distributed Dataset (RDD), which can be transformed and analyzed using Spark's rich set of operations like map, reduce, join, and window-based aggregations. After processing, the results can be sent to output systems like HDFS, databases, or dashboards for further use. Apache Spark also offers Structured Streaming, an advanced streaming engine built on Spark SQL, which provides a more user-friendly and declarative API using DataFrames and SQL queries. It allows developers to build streaming applications just like batch jobs, with the system handling the continuous execution behind the scenes. Spark ensures fault tolerance through data lineage and checkpointing, and it scales easily across a cluster of machines, making it ideal for high-volume, real-time data processing tasks.

Conclusion:

Apache Spark efficiently handles both batch and stream processing through its unified engine. While batch processing is suitable for historical data analysis, stream processing enables real-time analytics on continuously incoming data. Spark’s Structured Streaming model simplifies real-time data handling using familiar APIs and ensures fault-tolerant and scalable performance. Through this experiment, the practical understanding of how Apache Spark processes both static and dynamic data was achieved, highlighting its significance in real-world big data applications.