NAME:Himanshu Naik **CLASS:D15C** **ROLL.NO:63**

Experiment-10

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

Theory:

What is Streaming? Explain Batch and Stream Data:

Streaming refers to the real-time processing of data as it is generated. This approach is essential for applications that require instant insights or decisions, such as fraud detection systems, stock trading platforms, and live analytics dashboards. Streaming data is continuous, time-sensitive, and typically comes in an unbounded form that never stops.

On the other hand, **batch processing** involves collecting and storing data over a fixed interval, and then processing it in bulk. It is commonly used in traditional data processing scenarios such as ETL (Extract, Transform, Load), report generation, and data aggregation tasks. The data is processed in defined blocks or batches at scheduled times.

Examples:

- **Batch:** Compiling quarterly revenue reports.
- **Stream:** Analyzing live website user activity.

How Data Streaming Works with Apache Spark:

Apache Spark supports stream processing using its **Structured Streaming** module. This engine enables developers to process real-time data streams using the same high-level APIs as used for batch data, making it easier to build consistent and maintainable pipelines.

Structured Streaming conceptualizes data streams as continuously growing tables and performs computations incrementally as new data arrives. Spark can pull data from various streaming sources such as Apache Kafka, TCP/IP sockets, file systems, or cloud storage platforms.

After ingestion, Spark processes the data using familiar transformations like `filter`, `select`, and `groupBy`, along with advanced operations such as **windowing**, **watermarking** for handling late data, and **checkpointing** to ensure resilience and data recovery in case of failure.

Spark processes streaming data in **micro-batches**, meaning it treats small chunks of streaming data like mini batch jobs, ensuring both high throughput and low latency. The processed results can be stored in sinks like HDFS, databases, or displayed on live dashboards.

Key Highlights:

- Unified API for both real-time and batch data
- Ability to manage application state over time
- Compatible with structured data sources like Kafka
- Designed for scalability and fault tolerance

Real-World Use Cases:

- Detecting anomalies in banking transactions
- Analyzing server logs as they’re generated
- Monitoring trends on social media in real-time

Conclusion:

Through this exploration, I gained a clear understanding of the contrast between batch and streaming data models. Batch is well-suited for scheduled, large-scale jobs, while streaming is ideal for situations requiring immediate analysis. Apache Spark’s Structured Streaming provides a streamlined and unified solution to handle both styles of data processing. I learned how to ingest live data, transform it, and deliver insights in real time, making Spark a highly effective tool for building robust and intelligent data pipelines in modern data ecosystems.