DataCo Real-time Data Pipeline - Brief Report

Approach Taken:
The objective of this project was to build a real-time data pipeline for DataCo to ingest, process, and analyze clickstream data from a web application. The data pipeline was designed to handle data ingestion from Apache Kafka, store the ingested data in AWS S3, perform periodic processing using Apache Spark, and index the processed data in Elasticsearch for easy searching and analysis.

The following steps were followed to implement the data pipeline:

1. Data Ingestion from Kafka:
  - A Kafka consumer script was developed using the `kafka-python` library to consume data from the designated Kafka topic.
  - The consumed messages were parsed, and relevant fields such as user ID, timestamp, URL, country, city, and user agent were extracted.
  - The extracted data was stored in AWS S3 using a suitable storage format like CSV or JSON.

2. Periodic Processing of Clickstream Data in AWS S3:
  - Apache Spark was utilized to process the stored clickstream data in AWS S3. The `pyspark` library was used for this purpose.
  - A Spark job was developed to read the clickstream data from AWS S3 and perform aggregations.
  - Aggregations included grouping the data by URL and country, calculating the number of clicks, unique users, and average time spent on each URL by users from each country.
  - The processed data was stored in a suitable format like CSV or JSON.

3. Indexing Processed Data in Elasticsearch:
  - Elasticsearch was employed as the indexing and searching tool for the processed clickstream data.
  - A script was developed to read the processed data from AWS S3.
  - The data was indexed in Elasticsearch using the appropriate Elasticsearch client or library, such as the `elasticsearch` library in Python.

Assumptions Made:
During the implementation of the data pipeline, the following assumptions were made:

1. Kafka Setup: It was assumed that Apache Kafka was already installed and properly configured on the system. The Kafka topic for clickstream data ingestion was assumed to be created.

2. AWS S3 Setup: It was assumed that an AWS S3 bucket was available and accessible. The necessary credentials and permissions for reading and writing data to the bucket were assumed to be in place.

3. Apache Spark and Elasticsearch Setup: It was assumed that Apache Spark and Elasticsearch were properly installed and configured on the system. The necessary dependencies and libraries, such as `pyspark` and `elasticsearch`, were assumed to be available.

4. Data Formats: The specific data formats for storing the ingested and processed data in AWS S3 were not mentioned in the requirements. It was assumed that suitable formats like CSV or JSON would be used based on their efficiency and compatibility with Spark.

5. AWS S3 Data Partitioning: Data partitioning in AWS S3 was not explicitly mentioned. It was assumed that partitioning strategies, such as partitioning by date or any other relevant field, would be implemented for efficient data retrieval and processing in Spark.

Conclusion:
In conclusion, the data pipeline implemented for DataCo successfully achieves the ingestion, storage, processing, and indexing of clickstream data in a real-time manner. The approach taken utilizes Apache Kafka for data ingestion, AWS S3 for data storage, Apache Spark for data processing, and Elasticsearch for data indexing and searching. Assumptions were made regarding the setup of Kafka, AWS S3, Apache Spark, and Elasticsearch, as well as the data formats and partitioning strategies. The implemented pipeline meets the requirements of the tasks, and further enhancements can be made based on specific use cases and environment considerations.