BDTT

Week 4

Apache, Spark, Databricks & Spark DataFrames

2025

1. Learning Outcomes

- By the end of this week, you will be able to demonstrate a general understanding of Apache Spark and Databricks, including their core functionalities, architecture, and applications in big data processing and analytics.
- You will be able to explain what a DataFrame is, its structure, and its role in big data processing with PySpark.
- By the end of this week, you will be able to perform various DataFrame operations in PySpark, including filtering, transforming, and aggregating data.
- You will be able to convert a DataFrame to an RDD and vice versa, understanding the differences and appropriate use cases for each."

2. What is Apache

Apache is the umbrella name for a vast and multidimensional open-source project ecosystem, encompassing numerous frameworks and tools designed for big data processing, distributed computing, and software development.

Key components include Apache Spark, a powerful engine for large-scale data processing; Apache Hadoop, a framework for distributed storage and processing of massive datasets; Apache Hive, a data warehousing solution that provides SQL-like queries for big data; Apache Avro, a data serialization framework; Apache Ambari, a management tool for Hadoop clusters; and Apache Kafka, a distributed event streaming platform. Each of these projects serves a specific purpose, addressing various aspects of modern data engineering and analytics, making Apache a cornerstone for building scalable and efficient data-driven solutions.

3. Apache Project

The **Apache Project**, governed by the **Apache Software Foundation (ASF)**, is a global community-driven initiative that develops and maintains open-source software solutions for a wide range of use cases. Founded in 1999, the ASF provides a collaborative environment for developers to create innovative, freely available software that powers much of the internet and modern computing infrastructure. The

Apache Project is renowned for its modular approach, with each project operating independently under the "Apache Way"—a philosophy emphasising community, transparency, and meritocracy. With over 350 active projects, including widely used tools like Apache Spark, Apache Hadoop, and Apache Kafka, the Apache Project has become synonymous with scalable, reliable, and enterprise-grade open-source solutions.

4. What is Apache Spark

Apache Spark is an open-source, distributed computing system designed for fast and efficient processing of large-scale data. Originally developed at UC Berkeley's AMPLab, Spark is now one of the most popular big data frameworks, known for its speed, ease of use, and versatility. It supports a wide range of data processing tasks, including batch processing, streaming, machine learning, and graph analytics, all within a unified framework. Spark's in-memory computing capability dramatically speeds up data processing compared to traditional disk-based approaches like Hadoop MapReduce. Its APIs are available in multiple languages, including Python (PySpark), Scala, Java, and R, making it accessible to a broad audience. Spark is highly scalable and integrates seamlessly with other big data tools like Hadoop, Kafka, and Hive, making it a go-to choice for modern data engineering and analytics.

5. What is Databricks

Databricks is a cloud-based platform built specifically for big data analytics and artificial intelligence (AI) applications. Co-founded by the creators of Apache Spark, Databricks provides an integrated workspace for data engineering, data science, and machine learning, allowing teams to collaborate efficiently. It combines the scalability of cloud computing with the power of Apache Spark, enabling users to process large datasets, build machine learning models, and execute analytics workflows seamlessly. With features like collaborative notebooks, automated cluster management, and built-in support for popular programming languages (e.g., Python, R, SQL, and Scala), Databricks simplifies the end-to-end data lifecycle. Additionally, its support for integrations with various data sources, real-time streaming, and structured data pipelines makes Databricks an essential tool for organisations looking to accelerate data-driven decision-making and innovation.

6. What is a Dataframe

A **DataFrame** is a distributed collection of rows organised under named columns. It can be visualised as an Excel sheet with column headers or conceptualised as a table in a relational database. In programming terms, it resembles a DataFrame in R or Python.

DataFrames share three key characteristics with RDDs (Resilient Distributed Datasets).

First, they are **immutable**, meaning you can create a DataFrame but cannot modify it directly; instead, transformations can be applied to produce a new DataFrame. Second, they feature **lazy evaluation**, where tasks are not executed until an action, such as retrieving or saving data, is performed.

Lastly, they are **distributed**, like RDDs, ensuring scalability and efficient processing across multiple nodes in a cluster.

These features make DataFrames a powerful tool for handling large-scale data in distributed computing environments.

7. What is a Dataframe Schema

A **DataFrame schema** is a structural definition that describes the organisation of data within a DataFrame. It defines the column names, their data types (e.g., integer, string, double, etc.), and sometimes additional metadata about the data (e.g., nullable properties or constraints). In simple terms, the schema acts as a blueprint for the DataFrame, ensuring the data is well-structured and properly understood by the processing engine.

For example, in PySpark, the schema of a DataFrame provides an organised representation of the data, allowing users to understand how data is stored and enabling operations like querying, transformations, and validations. The schema can either be inferred automatically based on the input data or explicitly defined by the user. This explicit control helps prevent errors and ensures compatibility with downstream operations or applications.

An example in PySpark: If a DataFrame contains information about employees, its schema might look like this:

```
root
|-- name: string (nullable = true)
|-- age: integer (nullable = true)
|-- department: string (nullable = true)
|-- salary: double (nullable = true)
```

This schema tells us the DataFrame has four columns with their respective data types and indicates whether each column allows null values.

8. DataFrame operations

Let's see a brief overview of **transformation** and **action** operations for DataFrames in PySpark:

Transformations (Create a new DataFrame from an existing one):

- **filter()**: Filter rows based on a condition.
  - Example: Select rows where a column value is greater than a certain number.
- **select()**: Select specific columns from a DataFrame.
- **withColumn()**: Add or update a column.
- **groupBy()**: Group data by a column(s) for aggregation.
- **join()**: Combine two DataFrames based on a key.
- **distinct()**: Remove duplicate rows.
- **orderBy() / sort()**: Sort the DataFrame based on one or more columns.
- **drop()**: Remove specific columns from the DataFrame.
- **union()**: Combine rows of two DataFrames.

Actions (Trigger execution and return results):

- **show()**: Display the first few rows of the DataFrame.
- **collect()**: Retrieve all data as a list (use cautiously with large datasets).
- **count()**: Count the number of rows in the DataFrame.
- **first() / head()**: Return the first row or rows of the DataFrame.
- **take(n)**: Retrieve the first n rows as a list.

- **write()**: Save the DataFrame to an external storage format (e.g., CSV, JSON).
- **describe()**: Generate summary statistics for numeric columns.
- **printSchema()**: Display the schema of the DataFrame.

To conclude, In this week, you will learn how to create, handle, and transform DataFrames in PySpark, a powerful tool for big data processing and analysis. You will explore how to efficiently load data into DataFrames, perform operations such as filtering, aggregating, and transforming data, and leverage the flexibility of PySpark's API to process large-scale datasets. Additionally, you will gain an understanding of how to convert RDDs to DataFrames and vice versa, allowing you to seamlessly switch between the two structures and take advantage of their respective strengths for various use cases in data engineering and analytics.