**COVID-19 Data Processing with Spark**

For this part, we will store the COVID-19 dataset in Amazon S3 and leverage the Databricks environment for data processing. Using Apache Spark’s core APIs, we will perform data transformations. This is partially Spark implementation of the Covid19 project on Databricks.

**Apache Spark Ecosystem and Architecture**

Apache Spark is a unified compute engine designed for large-scale, distributed data processing across clusters of machines. It provides a core execution engine (Spark Core) responsible for running all submitted jobs, along with several key libraries to extend its functionality:

* **Spark SQL:** Enables executing SQL queries within Spark applications, facilitating structured data processing.
* **Spark Streaming:** Supports real-time data processing, powering applications such as live maps and ride-sharing platforms.
* **MLlib:** A scalable machine learning library for building and deploying ML models on big data.
* **GraphX:** A library for graph processing, useful for social network analysis and other graph-structured data.

At its core, Spark breaks down massive datasets into smaller partitions, distributes them across a cluster of machines, processes them in parallel, and then combines the results. This approach enables efficient handling of petabytes of data that cannot be stored or processed on a single machine.

A **cluster** is a group of interconnected computers (nodes) that work together as a single system. While multiple machines provide the necessary resources, Apache Spark serves as the framework that manages and coordinates task execution across the cluster, ensuring seamless distributed computation.

In summary, Apache Spark offers a powerful framework to manage, process, and analyze vast volumes of data quickly and efficiently by leveraging parallel processing across clusters, supported by specialized libraries for SQL, streaming, machine learning, and graph processing.

**Architecture of an Apache Spark Application**

An Apache Spark application primarily involves two key components managed by a **cluster manager**:

* **Driver Process:** The central coordinator that manages the Spark application’s lifecycle and execution.
* **Executor Processes:** Distributed worker nodes that perform the actual data processing tasks.

**Core Concepts:**

1. **Spark Session:**  
   The entry point to any Spark application is the **Spark session** (commonly instantiated as spark). It represents the driver program and provides APIs to create DataFrames, execute SQL queries, and manage the Spark application workflow. Every Spark application starts with this session.
2. **Spark DataFrame:**  
   A DataFrame is a distributed collection of data organized into named columns, similar to a table or spreadsheet. DataFrames abstract the distributed nature of the data, allowing users to work with structured data without managing the underlying partitions manually.
3. **Transformations:**  
   Transformations define a logical plan to process data (e.g., filtering, mapping, adding columns). They are **lazy**, meaning Spark does not execute these operations immediately but builds an execution plan.
4. **Actions:**  
   Actions trigger the actual computation and execute the transformations planned by Spark (e.g., count(), collect()). Only when an action is called does Spark process the data and return results.

**Summary:**

* Spark applications are driven by a **driver** coordinating multiple **executors** across a cluster.
* The **Spark session** acts as the application’s entry point for data processing.
* DataFrames provide a distributed, structured data abstraction.
* Transformations build up the execution plan lazily, while actions trigger execution.

This architecture enables Spark to efficiently process large-scale data in a distributed and fault-tolerant manner.

**Databricks Overview**

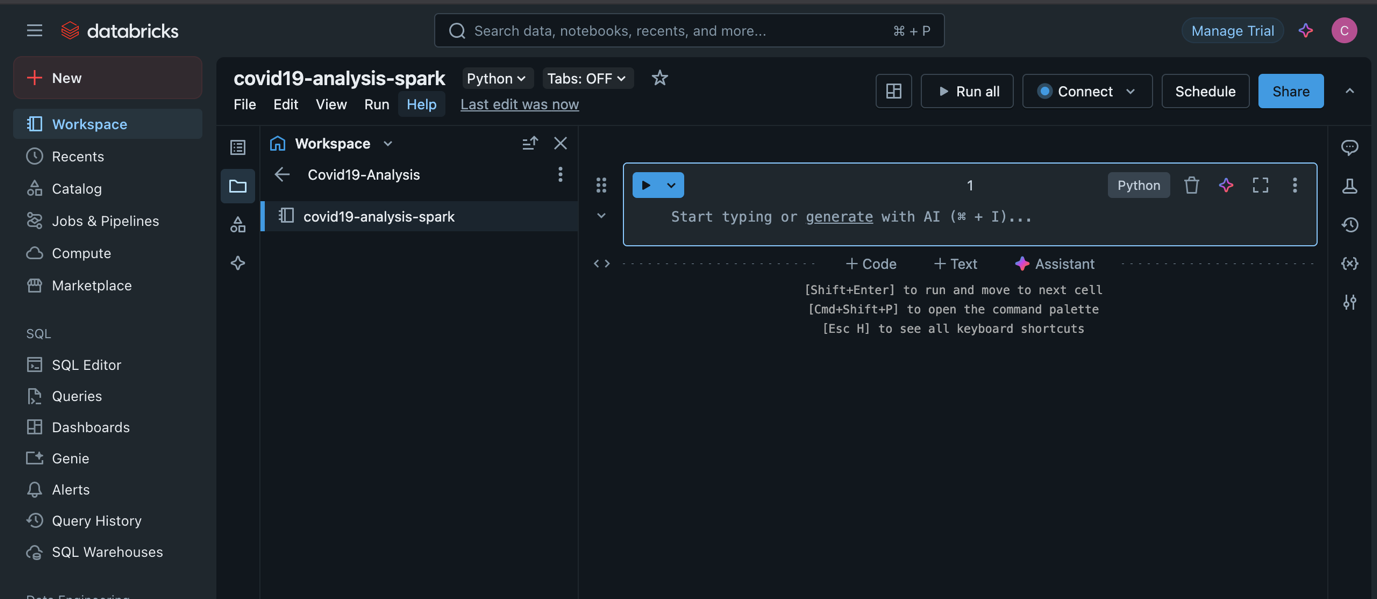
Databricks is a unified platform that simplifies the deployment and management of Apache Spark environments. Setting up Apache Spark on a local machine involves installing the Java Virtual Machine (JVM), configuring numerous packages, setting environment variables, and troubleshooting errors. Scaling this setup to an on-premises data center with thousands of machines adds significant complexity, requiring considerations for infrastructure such as storage, networking, and resource management.

Databricks eliminates these challenges by providing a fully managed, cloud-based Apache Spark environment. With just a few clicks, users can launch scalable Spark clusters without worrying about underlying infrastructure. This allows data professionals to focus entirely on solving business problems rather than managing technical operations.

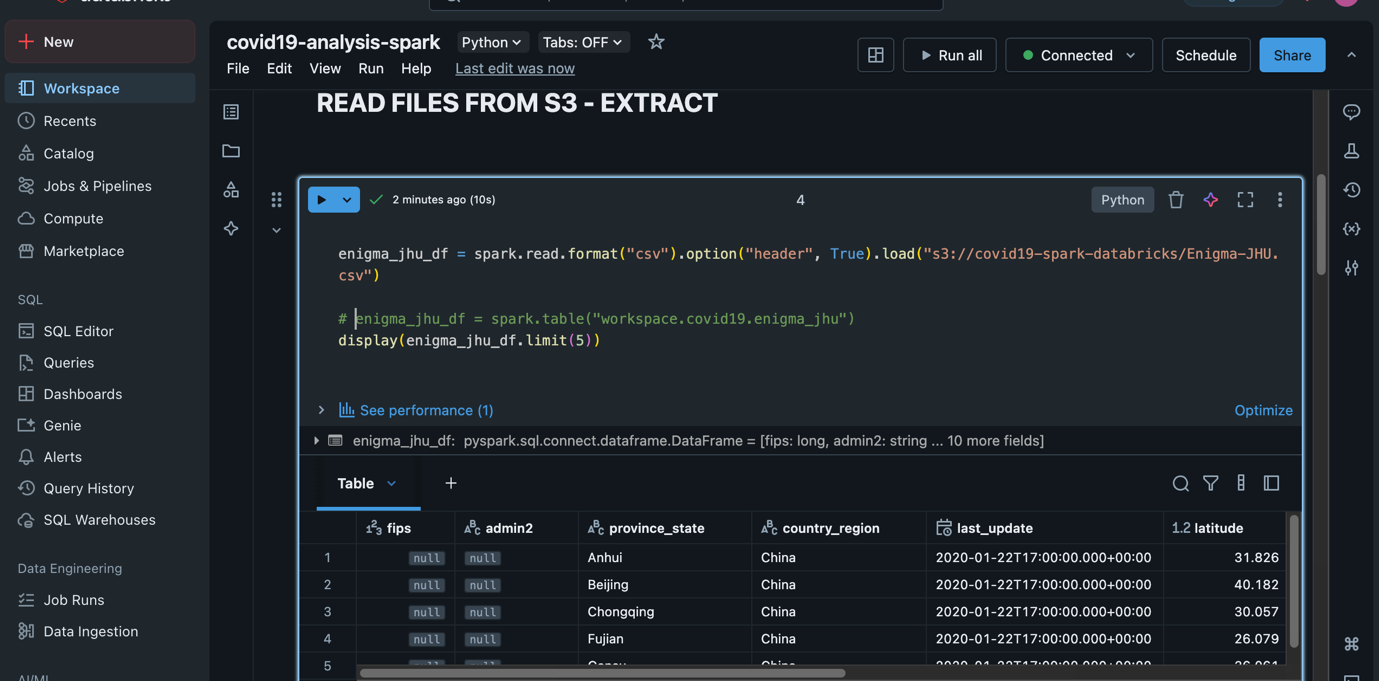
Beyond Spark, Databricks offers additional capabilities including a Lakehouse architecture, integrated SQL data warehousing, and machine learning tools, making it a comprehensive platform for data engineering, analytics, and AI workloads. At its core, however, Databricks is designed to deliver a seamless, scalable environment to run Apache Spark workloads efficiently.

Databricks-Spark

1. Create a new notebook connect to cluster



1. Read Data from S3: Create a publicly available S3 bucket ([covid19-spark-databricks](https://us-east-1.console.aws.amazon.com/s3/buckets/covid19-spark-databricks?region=us-east-1&bucketType=general)) and upload your files.



Check *covid19-etl-spark.ipynb* under scripts