Apache Spark & PySpark:

EXPLORING FRAMEWORKS FOR EFFICIENT BIG DATA PROCESSING

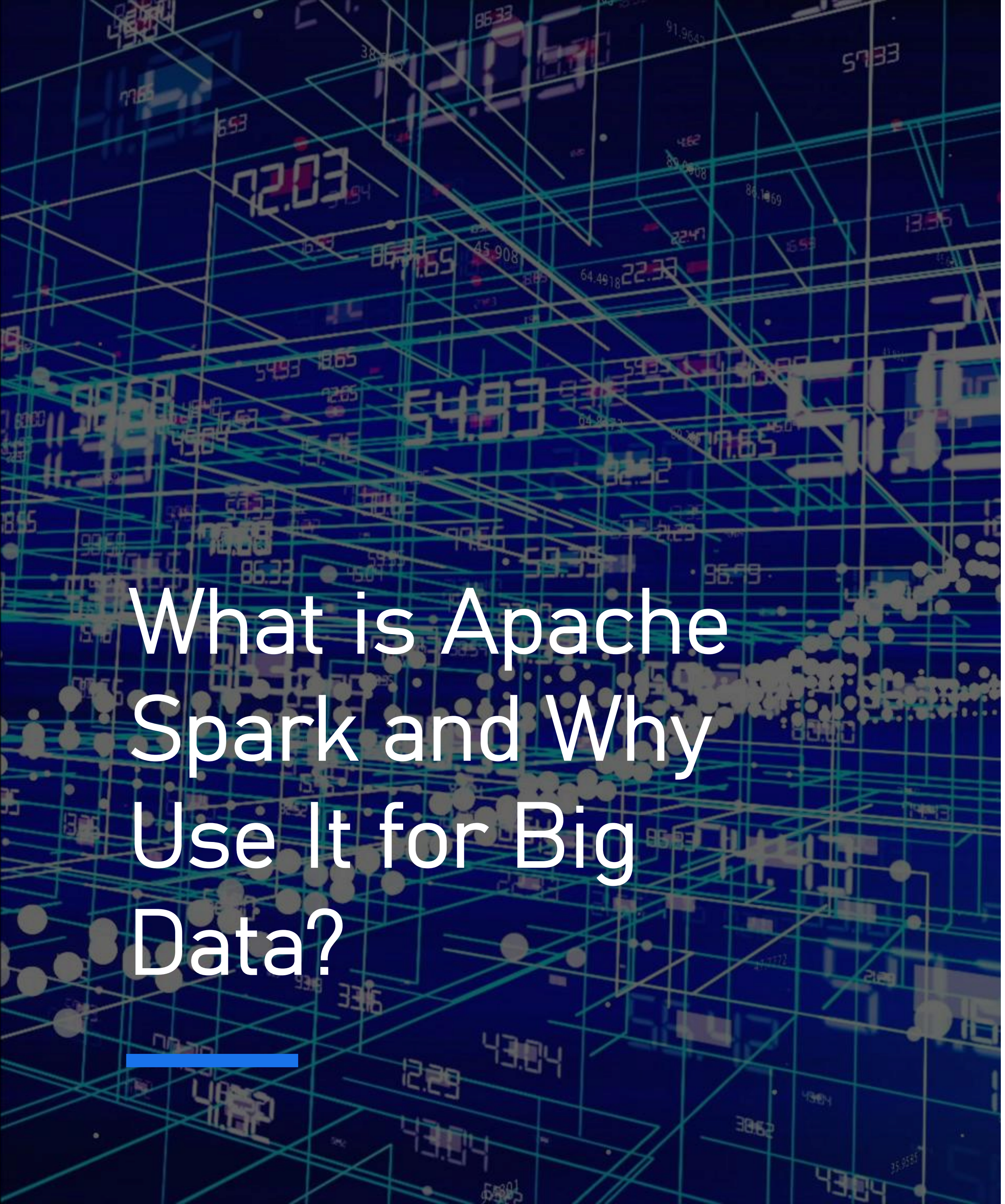
MC ID: 7124506 | PARVEEN KR | PARVEEN.R@HOTMAIL.COM

Presentation Agenda

- Understanding Apache Spark and Its Applications
- Core Architecture and Execution Flow
- Data Structures in Spark: RDDs, DataFrames, and Datasets
- Transformations and Actions in Spark
- Getting Started with PySpark
- Choosing the Right Tool: PySpark, Pandas, or Dask?
- Managing Data with PySpark: Schema and Formats
- Loading Data from Multiple Sources and Essential DataFrame Operations

Understanding Apache Spark and Its Applications

MC ID: 7124506 | PARVEEN KR | PARVEEN.R@HOTMAIL.COM



What is Apache Spark and Why Use It for Big Data?

Apache Spark Overview

Apache Spark is an open-source distributed system for fast, large-scale data processing across clusters.

Data Splitting Across Machines

Spark divides data and computation across multiple machines enabling the processing of terabytes to petabytes efficiently.

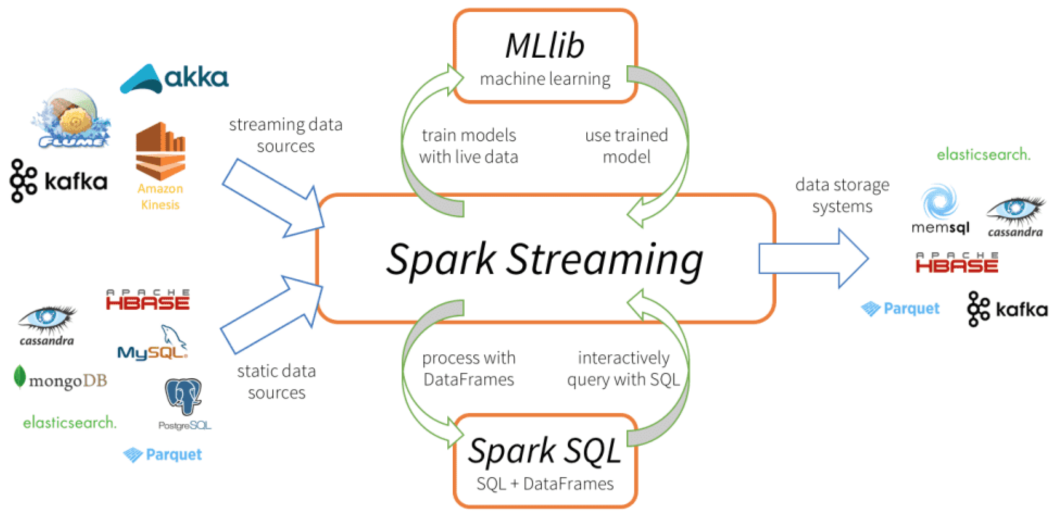
Limitations of Traditional Tools

Traditional tools like Excel and Pandas cannot handle data too large for memory or require fast processing speeds.

Common Use Cases

Spark is widely used for ETL jobs, data analysis, real-time processing, and large-scale machine learning tasks.

MC ID: 7124506 | PARVEEN KR | PARVEEN.R@HOTMAIL.COM



Spark in Action

Excel/Pandas Analogy

Excel and Pandas work like a single chef in a home kitchen, handling smaller data tasks efficiently.

Spark Analogy

Spark functions like a professional restaurant kitchen with many chefs collaborating on the same meal simultaneously.

Netflix Data Processing

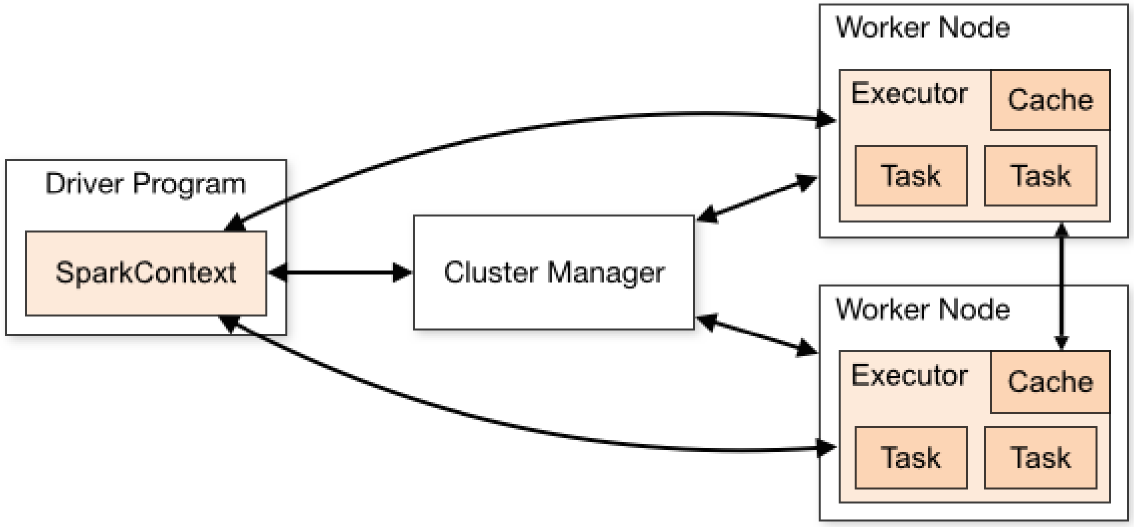
Netflix uses Spark to process billions of movie views daily for generating personalized recommendations efficiently.

Spark vs Excel Performance

Spark can process massive data logs in seconds, a feat not achievable with Excel or Pandas.

Core Architecture and Execution Flow

Key Components of Spark Architecture



Driver Program Role

Driver program coordinates the overall execution and manages the Spark application flow.

Cluster Manager Function

Cluster manager allocates computing resources across the cluster like a kitchen manager organizing staff.

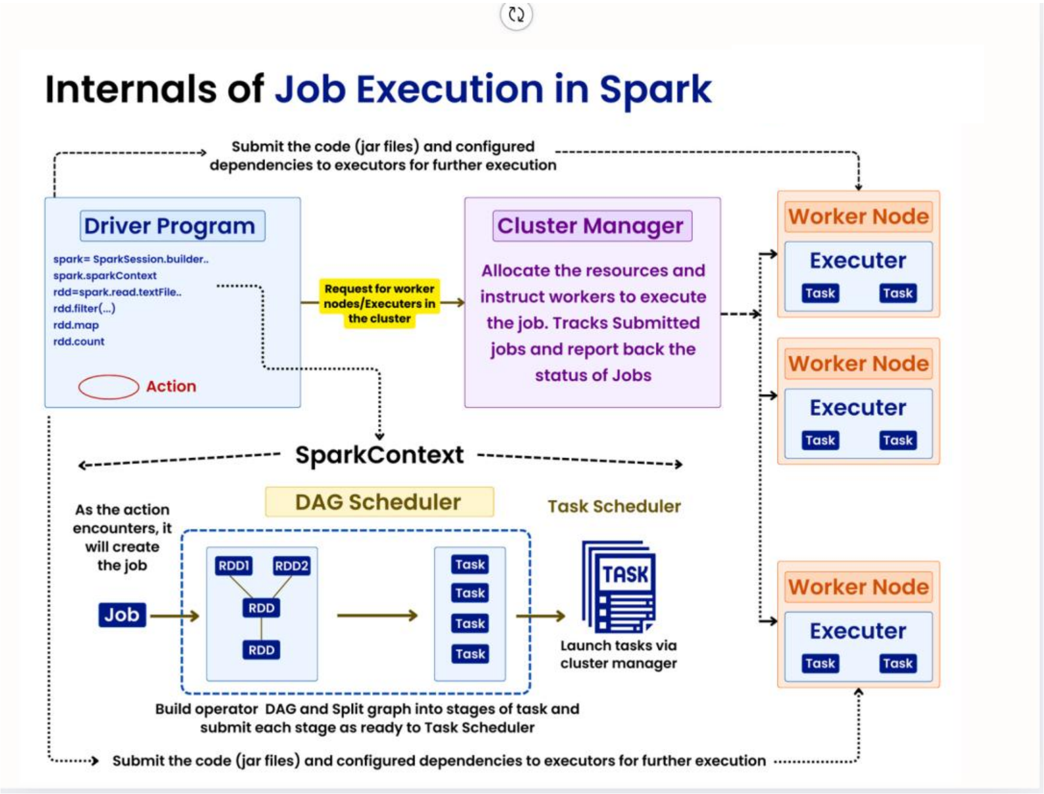
Executors Processing

Executors perform the actual data processing tasks assigned by the driver program.

Spark Context Interface

Spark Context acts as the interface connecting user code to the Spark cluster functionalities.

How Spark Executes Your Code



Code Writing and Task Creation

You write code in your notebook, which the Driver breaks down into smaller executable tasks.

Task Assignment by Cluster Manager

The Cluster Manager assigns individual tasks to various Executors in the cluster for parallel processing.

Executors Process and Return Results

Executors run assigned tasks on data and send the processed results back to the Driver for aggregation.

Analogy to Food Delivery

Like a food delivery app, where the Driver is the customer, Cluster Manager assigns delivery workers, who deliver food as data.

Data Structures in Spark: RDDs, DataFrames, and Datasets

Comparing RDDs, DataFrames, and Datasets

RDD Characteristics

RDDs are low-level data structures allowing custom transformations and raw data manipulation.

DataFrame Features

DataFrames provide a table-like structure with rows and columns, optimized for performance and ease of use.

Dataset Overview

Datasets are typed versions of DataFrames mainly used in Scala and Java for type safety and functional programming.

Use Case Comparison

RDDs are ideal for custom log parsing, whereas DataFrames excel in analyzing structured sales data efficiently.

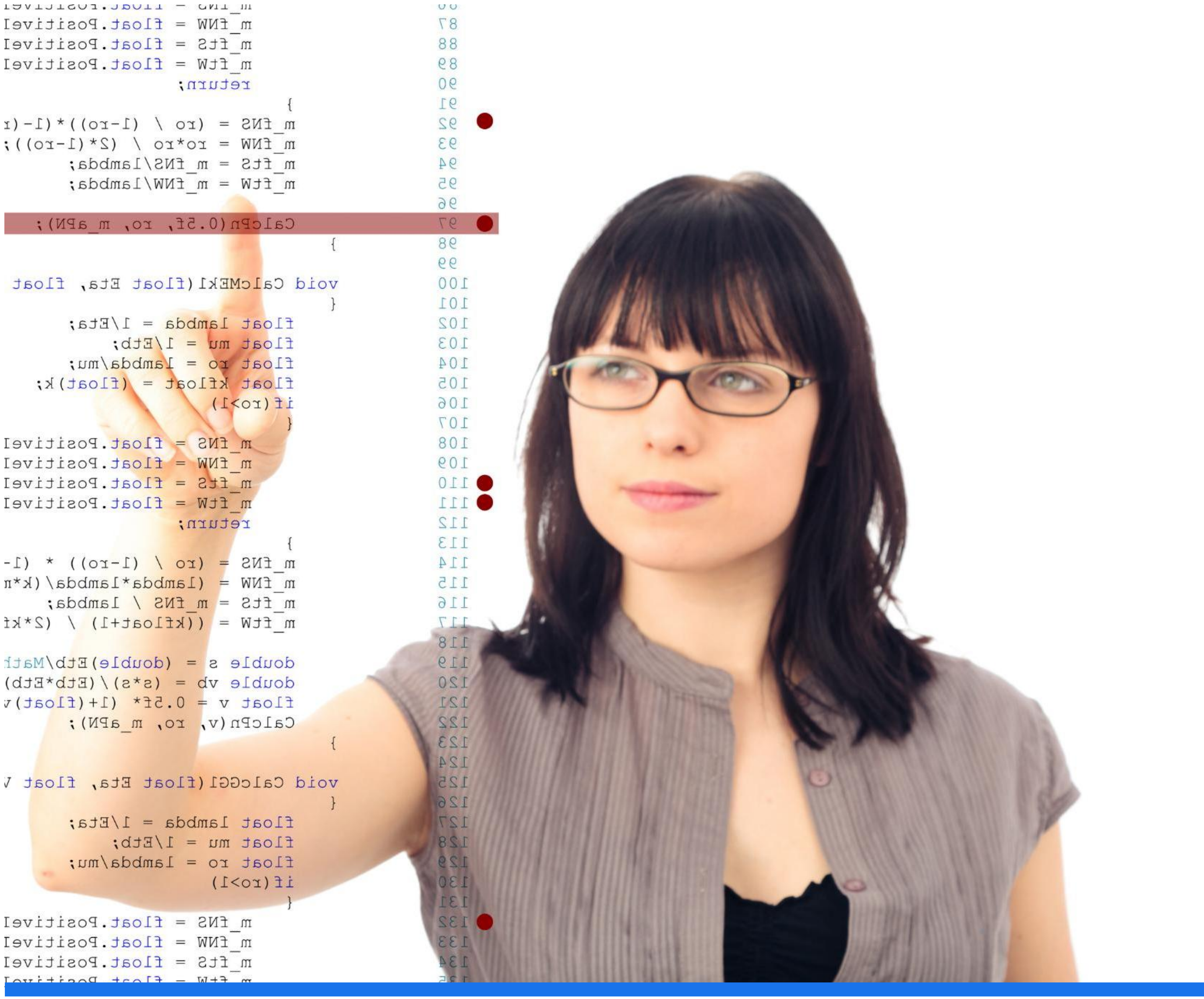
Practical Examples: Working with RDDs and DataFrames

RDD Text File Loading

Load text file into an RDD to perform distributed data processing and retrieve the first line.

DataFrame CSV Reading

Read CSV file as a DataFrame with headers for structured data manipulation and display first rows.



Transformations and Actions in Spark

Understanding Transformations vs. Actions

Transformations

map	join	union	distinct	repartition
mapPartitions	flatMap	intersection	pipe	coalesce
cartesian	cogroup	filter	sample	
sortByKey	groupByKey	reduceByKey	aggregateByKey	
mapPartitionswithIndex		repartitionAndSortWithinPartitions		

Actions

reduce	take	collect	takeSample	count
takeOrdered	countByKey	first	foreach	saveAsTextFile
saveAsSequenceFile		saveAsObjectFile		

Definition of Transformation

Transformations prepare data operations but do not execute computations immediately.

Definition of Action

Actions trigger computation and produce results by executing previously defined transformations.

Transformation vs Action Analogy

Transformations are like writing a recipe; actions are like cooking the meal.

Getting Started with PySpark

What is PySpark and Why Use It?

PySpark Overview

PySpark enables Python users to access Spark's distributed computing capabilities for scalable data processing.

Handling Large Datasets

PySpark works efficiently with huge datasets, unlike Pandas which struggles with big data.

Scalability Across Devices

The same PySpark code runs seamlessly on a laptop or a cluster of thousands of machines.

Real-World Application

PySpark is used for cleaning millions of sensor readings from smart meters in real-time data processing.

MC ID: 7124506 | PARVEEN KR | PARVEEN.R@HOTMAIL.COM

PySpark in Practice: Sensor Data Example

Reading CSV Data

PySpark reads sensor data from CSV files with header and schema inference for structured processing.

Grouping Data by Device

Data is grouped by device ID to aggregate readings from individual sensors efficiently.

Computing Average Readings

Average sensor readings per device are calculated to summarize data insights.

MC ID: 7124506 | PARVEEN KR | PARVEEN.R@HOTMAIL.COM

Choosing the Right Tool: PySpark, Pandas, or Dask?

When to Use Pandas, Dask, or PySpark

Pandas for Small Data

Pandas is ideal for small datasets that fit within a laptop's memory, typically under 1-2GB.

Dask for Medium Data

Dask scales to datasets larger than memory on a single machine or small cluster, handling medium data efficiently.

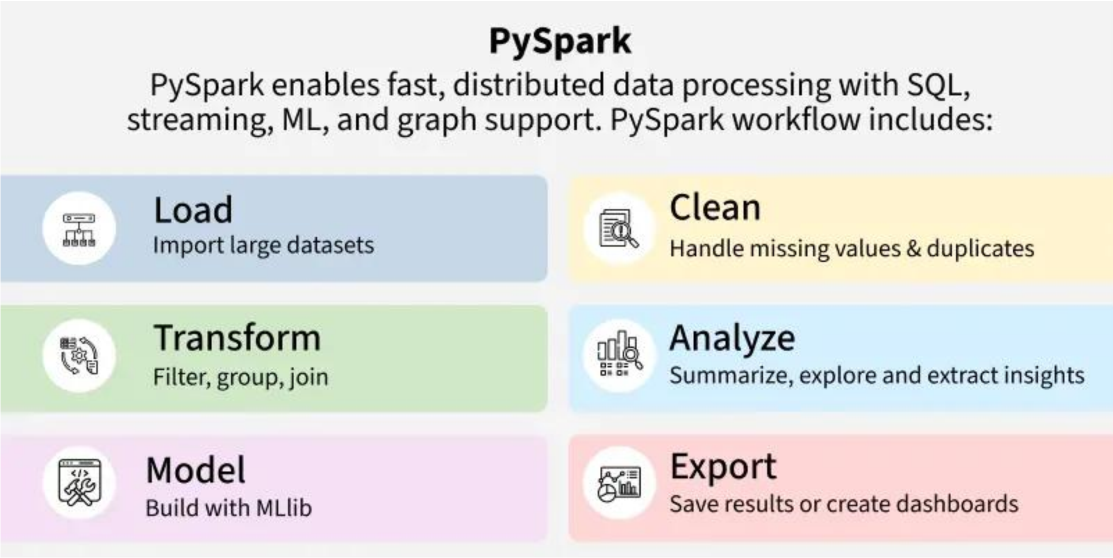
PySpark for Large Data

PySpark manages huge datasets distributed across many computers, suitable for petabyte-scale data processing.

PANDAS VS ALTERNATIVES

	What it is	Good for	Not best fit for
pandas	Go-to Python library for data analysis	Data manipulation and further analysis in different domains	Very large datasets, unstructured data
NumPy	Python library for numerical computing	Mathematical operations on arrays and matrices	Non-numerical data types, data manipulation tasks
PySpark	Python API for Apache Spark	Big data processing in distributed environment	Small-scale data tasks
dask	Python library for parallel and distributed computing	Processing of larger-than-memory datasets	Data manipulation tasks
MODIN	Python library for distributed computing of Pandas DataFrames	Manipulating datasets from 1MB to 1TB+	Small-scale data tasks
vaex.io	Python library for larger-than-memory Pandas DataFrames	Visualizing and exploring big tabular datasets	Data manipulation tasks
R	Statistical programming language	Data mining, data wrangling, data visualization, machine learning operations	Basic data manipulation tasks, big data projects

MC ID: 7124506 | PARVEEN KR | PARVEEN.R@HOTMAIL.COM



Loading Large Data

PySpark allows efficient loading of large CSV datasets using distributed cluster computing.

Counting Rows Efficiently

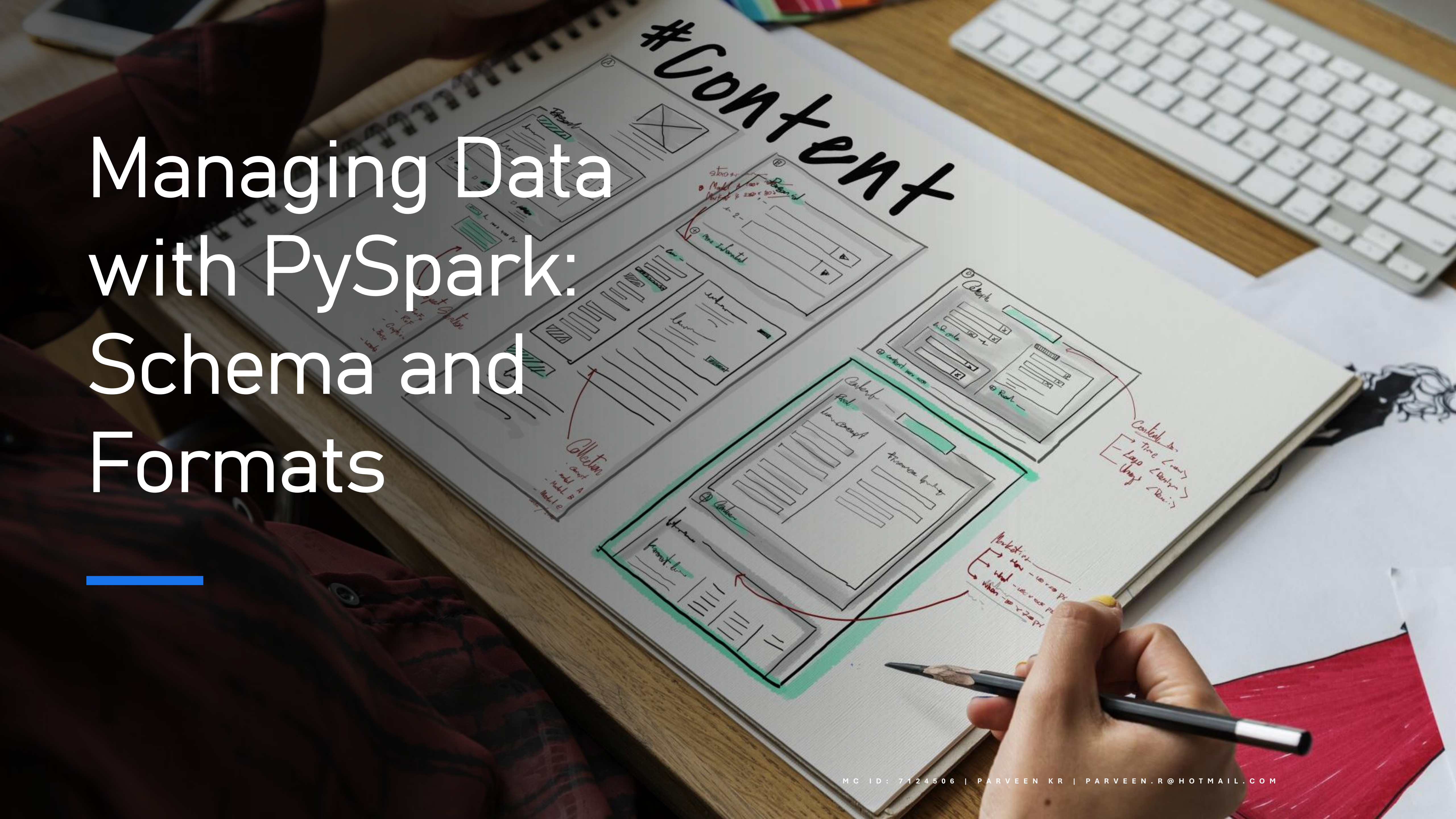
Counting rows in big dataframes is optimized in PySpark through distributed processing.

PySpark for Big Data

PySpark provides a scalable framework to handle and analyze large datasets with ease.

Handling Large Data: PySpark Example

Managing Data with PySpark: Schema and Formats



MC ID: 7124506 | PARVEEN KR | PARVEEN.R@HOTMAIL.COM

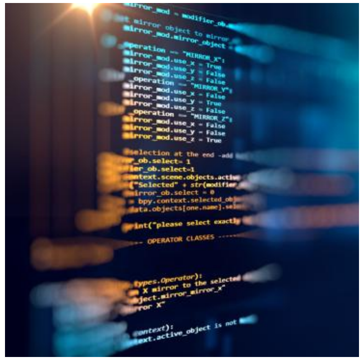
Schema Inference vs. Explicit Schema Definition

Role of Schema in Spark

Schema defines expected data types like string, integer, and date for Spark data processing.

Schema Inference Method

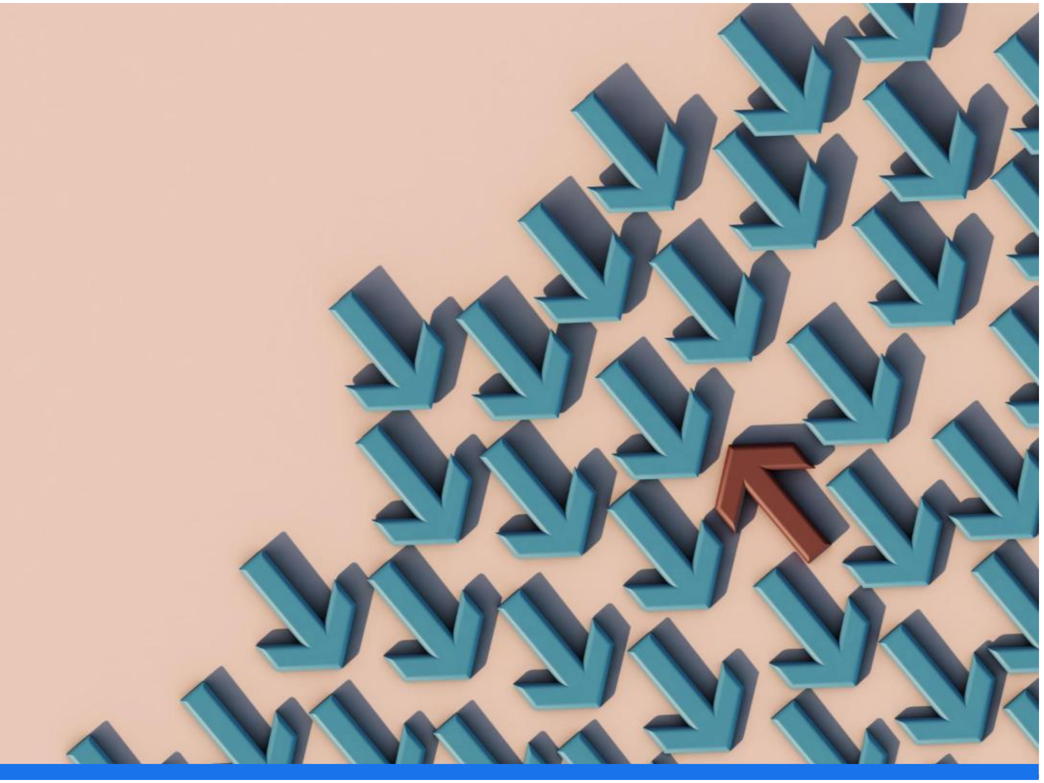
Spark guesses schema from data automatically, useful for quick checks but not always reliable.



Explicit Schema Definition

Explicitly defining schema ensures accuracy and is best practice for production environments.

Reading and Writing Data in Multiple Formats



Data Format Preferences

Different tools and business needs require various data formats for optimal performance and usability.

Human-Friendly Formats

CSV and JSON are easy to read and suitable for simple data handling and exchange.

Optimized Analytical Formats

Parquet and ORC formats are optimized for fast analytics and efficient data compression.

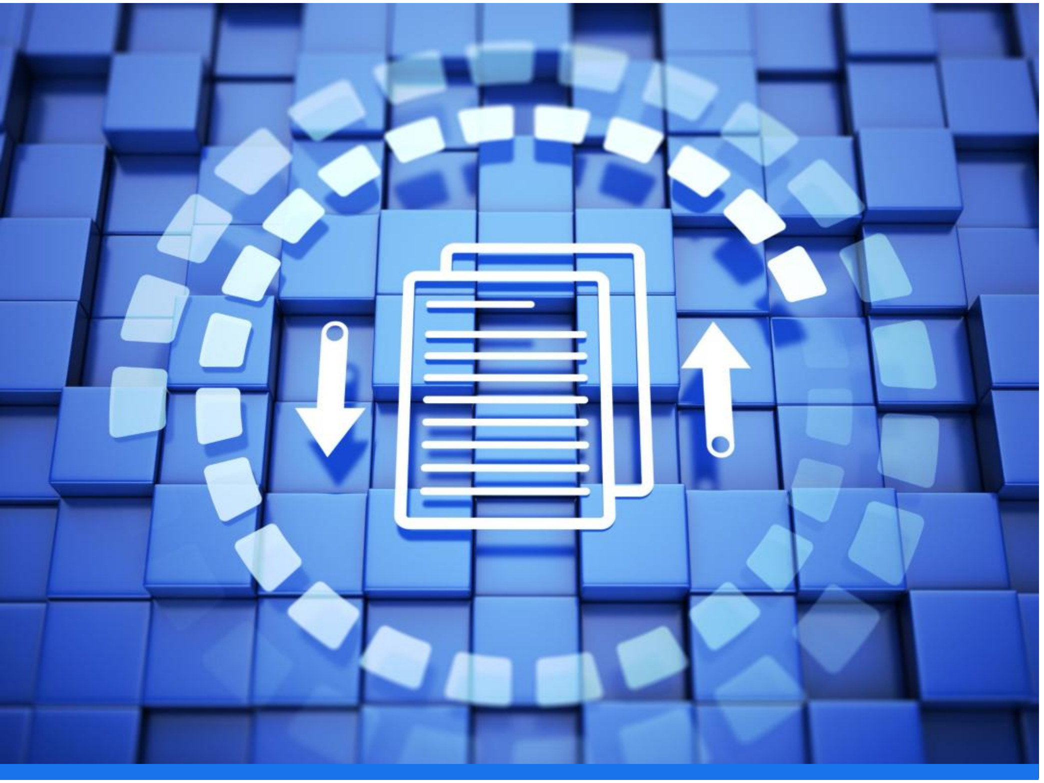
Delta Format Features

Delta format supports transactions and versioning, enhancing data reliability and management.

Loading Data from Multiple Sources and Essential DataFrame Operations

MC ID: 7124506 | PARVEEN KR | PARVEEN.R@HOTMAIL.COM

Creating DataFrames from Various Sources



DataFrames from RDD and Lists

RDDs handle text files or custom data, while lists and dictionaries enable quick demos and unit tests for DataFrames.

Loading Data from Databases

DataFrames can be created by loading production data directly from relational databases using JDBC connections.

Cloud Storage Integration

Cloud storage systems like S3, ADLS, and GCS provide scalable sources for DataFrames in big data processing.

Combining Multiple Data Sources

Combining CSVs, real-time database transactions, and JSON marketing data enables comprehensive analytics in DataFrames.

Essential DataFrame Operations: Select, Filter, Group, Sort, Join

Select Columns

Extract specific columns from a DataFrame to focus on relevant data attributes.

Filter Rows

Apply conditions to filter rows that meet specific criteria for targeted analysis.

Group and Aggregate Data

Group data by columns and perform aggregations like sum to summarize information.

Sort Results

Order data based on column values to identify top or bottom records efficiently.

Conclusion

Powerful Big Data Frameworks

Apache Spark and PySpark provide robust and scalable architectures for processing large datasets efficiently.

Core Concepts and Tools

Understanding key concepts and versatile tools in Spark and PySpark helps developers optimize data workflows.

Practical Applications

Applying Spark and PySpark enables efficient management and analysis of vast and complex datasets.