

# What is Big Data?

- Big Data refers to datasets that are too large or complex for traditional data-processing tools to handle.
- These datasets typically require advanced technologies to store, process, and analyze. Big Data is characterized by the 5 V's: Volume, Velocity, Variety, Veracity, and Value.

# Big Data: The 5 V's

V	Description
<b>Volume</b>	Refers to the large scale of data generated. Big Data systems can handle terabytes or even petabytes of data.
<b>Velocity</b>	The speed at which data is generated and processed in real-time from various sources, like social media, IoT devices, etc.
<b>Variety</b>	The different forms of data, including structured, semi-structured, and unstructured (e.g., text, images, videos).
<b>Veracity</b>	The uncertainty or quality of the data, ensuring data accuracy despite potential inconsistencies or incompleteness.
<b>Value</b>	The benefit or insights derived from Big Data, driving business decisions, enhancing efficiency, and creating opportunities.

# What is PySpark?

- PySpark is the Python API for Apache Spark, an open-source, distributed computing framework.
- PySpark enables parallel processing on large datasets by using Python to write Spark applications, making it accessible to developers familiar with Python while leveraging Spark's power.

# Difficulty level of questions

- Easy : Focus on basic concepts, definitions
- Medium: joins, aggregations, general scenarios
- Hard: performance optimization, Spark's architecture

# Analyzing Titanic Dataset with Spark

- This tutorial demonstrates how to analyze the Titanic dataset using Spark, covering basic operations like `groupBy()`, `aggregateByKey()`, `sortByKey()`, joins, and more.
- Notebook link to be provided

# What is RDD and what is its importance?

- **RDD** stands for **Resilient Distributed Dataset**.
- It is the fundamental data structure in Apache Spark.
- **Resilient**: If part of the data is lost (due to node failure), the RDD can recover that lost data using the original transformations (thanks to lineage).

- **Distributed:** RDDs are divided into partitions that are distributed across the nodes in a cluster. This allows Spark to process the data in parallel, making it much faster than traditional systems.
- **Dataset:** RDDs represent datasets that can be created from data in external storage systems like HDFS, local files, or existing data in memory.

# Why is RDD Important for Spark?

- **Fault Tolerance:** RDDs can automatically recover lost data. If a node fails, Spark uses the **lineage** (a record of transformations applied) to recreate the lost data without rerunning the entire job.
- **Parallel Processing:** Since RDDs are distributed across multiple nodes, Spark can process them in parallel, which speeds up data processing.
- **In-memory Computation:** RDDs allow Spark to cache data in memory, reducing disk I/O operations. This leads to faster execution, especially in iterative algorithms (like machine learning).
- **Immutable:** Once created, RDDs cannot be modified, but you can apply transformations (like map and filter) to create new RDDs.
- **Lazy Evaluation:** RDD transformations (like map() or filter()) are not executed immediately. Instead, Spark builds an execution plan (called a DAG) and only runs it when an action (like collect(), count()) is triggered. This allows Spark to optimize and combine operations.



# What are transformations and actions in PySpark? Give examples.

- **Transformations:** Operations that define how data is transformed but do not execute immediately. They are **lazy**.
  - Example: `map()`, `filter()`, `groupBy()`
  - Example code: `titanic_df.filter(titanic_df.Age > 30)`
- **Actions:** Operations that trigger the actual execution of transformations and return results.
  - Example: `collect()`, `count()`, `show()`
  - Example code: `titanic_df.count()`

# What are the different file formats supported by PySpark

- PySpark supports various file formats for reading and writing data, including:
  - **CSV**: Comma-separated values.
  - **Parquet**: Columnar storage format that is efficient for large datasets.
  - **JSON**: JavaScript Object Notation, useful for semi-structured data.
  - **ORC**: Optimized Row Columnar format, commonly used in the Hadoop ecosystem.
  - **Avro**: A compact, fast, and efficient data serialization format.

# What is the difference between caching and persistence?

- **Caching** and **persistence** allow you to store a DataFrame or RDD in memory (or memory/disk) so that future operations don't need to recompute the data.
- **Caching**: Stores the data in memory only. It's shorthand for `persist()` with the **memory-only** option.
- **Persistence** ( `persist()` ): Allows you to specify different storage levels, like memory, disk, or a combination of both.
- **Why useful:**
  - If you're reusing the same DataFrame or RDD multiple times, caching/persistence saves time by avoiding the need to re-compute the same transformations.
  - **Improves performance** for iterative algorithms (like machine learning), where the same data is processed repeatedly.

- Question: What does `agg()` do?
- Answer: `agg()` performs aggregate operations such as sum, average, min, and max on grouped data. In this case, it calculates the mean survival rate by gender.

Thank you!