Posgrados CUCEA
El mejor lugar para el talento

MCD | MAESTRÍA EN CIENCIA DE LOS DATOS

Maestría en Ciencia de Datos (MCD)

Procesamiento de Grandes Bases de Datos
(Soluciones Empresariales)

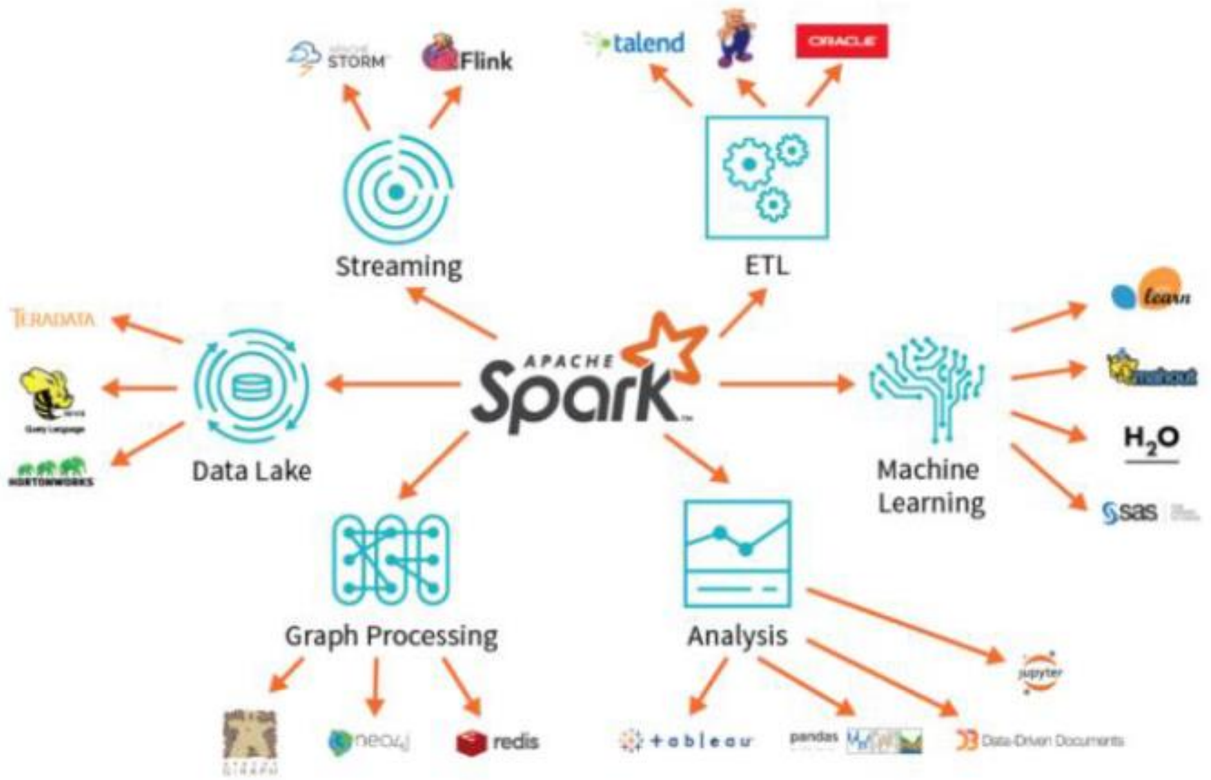
Unidad I

Soluciones Empresariales

Hadoop
Apache Spark
Instalación Apache Spark

Spark es una solución **Big Data** de **código abierto**. Desarrollado por el laboratorio RAD de **UC Berkeley** (2009).

Se ha convertido en una **herramienta de referencia** en el campo del Big Data.

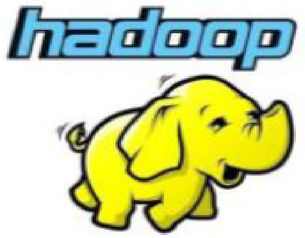


Apache Spark vs MapReduce

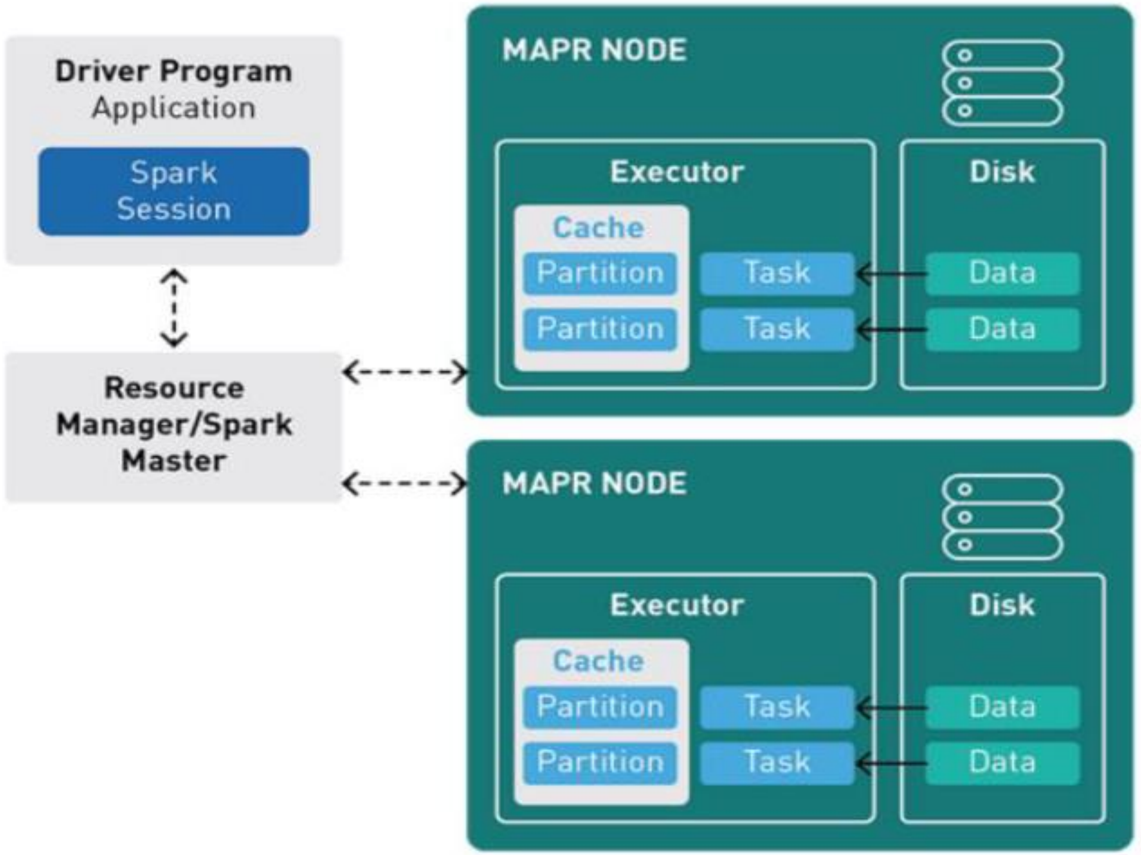
Más fácil y rápida que Hadoop MapReduce.

Diferencias:

- **Spark** mucho **más rápido** al almacenar en caché los datos en la **memoria** vs **MapReduce** en el **disco duro** (más lectura y escritura)
- Spark optimizado para un mejor **paralelismo**, utilización **CPU** e inicio más rápido
- Spark tiene modelo de **programación funcional** más rico
- Spark es especialmente útil para **algoritmos iterativos**

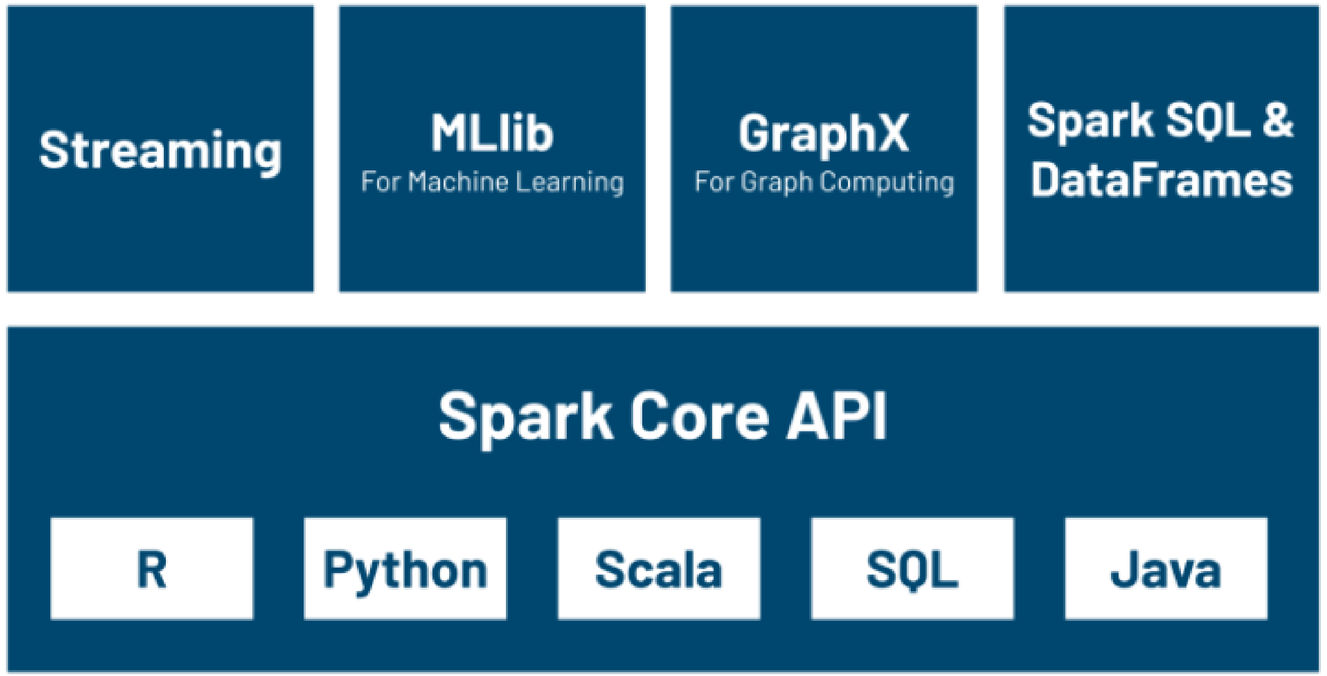


Cómo se Spark en un clúster



Componentes de Spark

Spark contiene un **ecosistema** de herramientas **muy completo.**

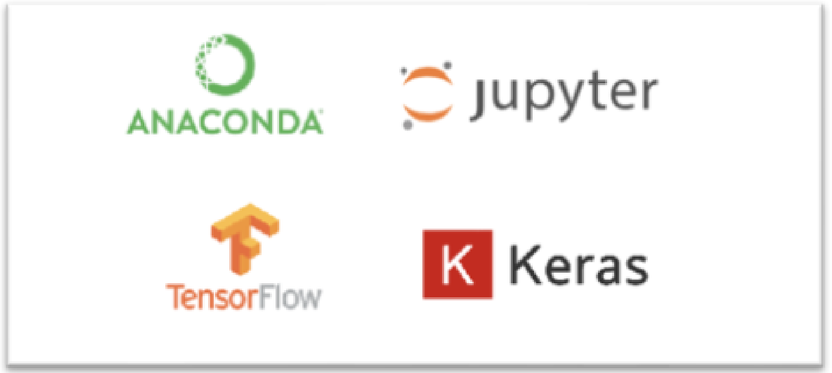


PySpark

PySpark es una biblioteca Spark **escrita en Python** para ejecutar la aplicación Python usando las **capacidades de Apache Spark**.

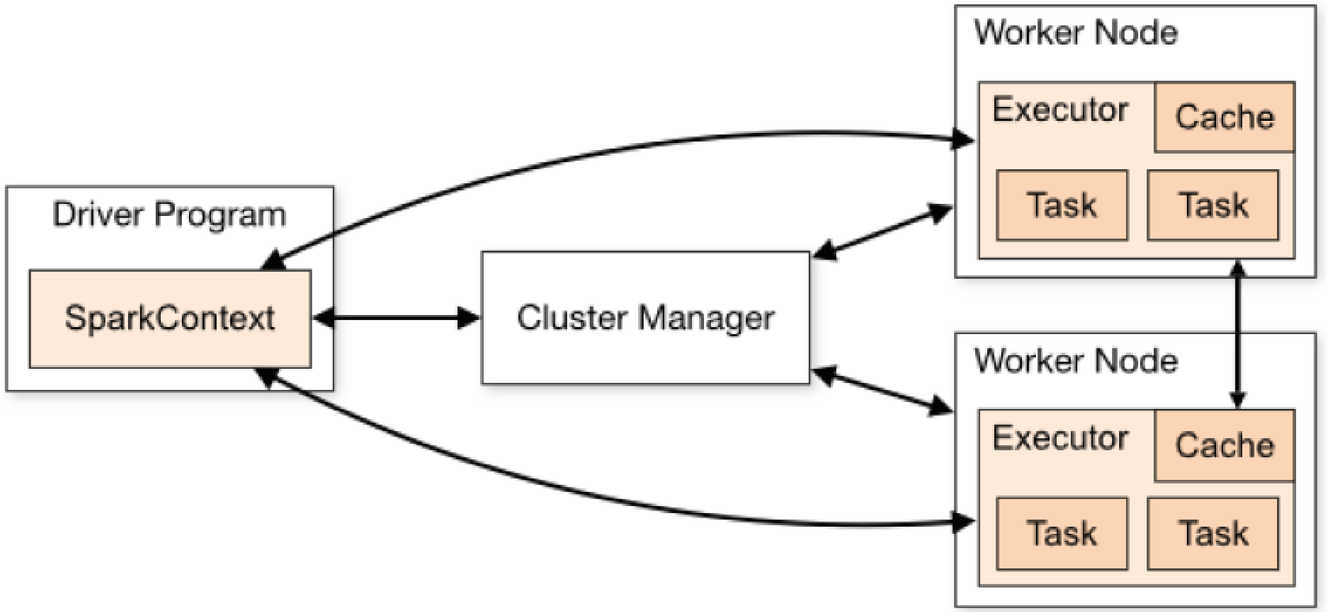
Ventajas de PySpark:

- **Fácil** de aprender
- Amplio conjunto de librerías para **ML y DS**
- Gran apoyo de la **comunidad**



Arquitectura de PySpark

Apache Spark funciona en una **arquitectura maestro-esclavo**. Las **operaciones** se ejecutan en los **trabajadores**, y el **Cluster Manager** administra los recursos.



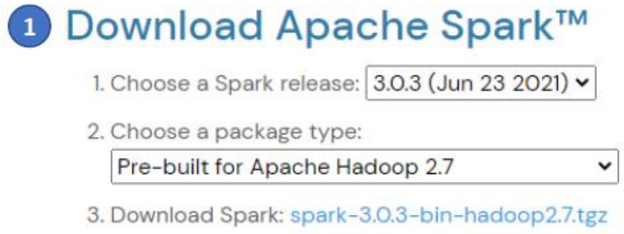
Tipos de administradores de clústeres

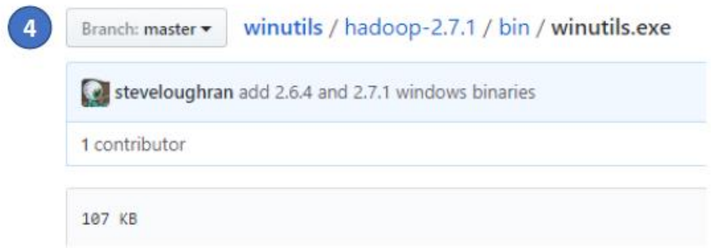
Spark admite los siguientes administradores de clústeres:

- **Standalone** : administrador de clúster simple
- **Apache Mesos** : es un administrador de clústeres que puede ejecutar también Hadoop MapReduce y PySpark.
- **Hadoop YARN** : el administrador de recursos en Hadoop 2
- **Kubernetes**: para automatizar la implementación y administración de aplicaciones en contenedores.

Pasos para instalar Spark (1)

- Descarga **Spark**de https://spark.apache.org/downloads.html
- Modifica el **log4j.properties.template** pon en log4j.rootCategory=**ERROR**en vez de INFO.
- Instala **Anaconda**de https://www.anaconda.com/
- Descarga **winutils.exe**. Es un binario de Hadoop para Windows -del repositorio de GitHub de https://github.com/steveloughran/winutils/. Vaya a la versión de Hadoop correspondiente con la distribución de Sparky busque winutils.exe en /**bin**.





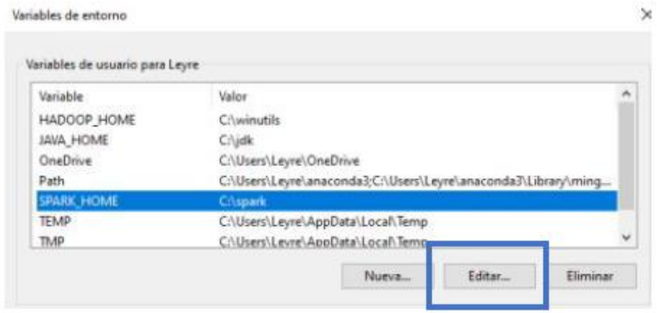
Pasos para instalar Spark (2)

- Si no tienes **Java**o la versión de Java es 7.x o menos, descargue e instale Java desde Oracle https://www.oracle.com/java/technologies/downloads/

- Descomprime Spark en **C:\spark**
- Añade el winutils.exe descargado a una carpeta de winutilsen C:. Debe quedar así: **C:\winutils\bin\winutils.exe**.
- Desde **cmd** ejecuta: “cd C:\winutils\bin” y después: winutils.exe chmod777 \tmp\hive

- Añade las variables de entorno:HADOOP_HOME ->C:\winutils

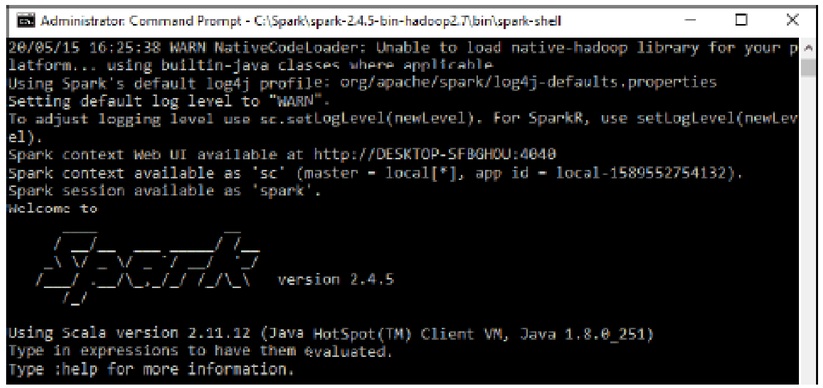
- SPARK_HOME -> C:\spark
- JAVA_HOME -> C:\jdk
- Path-> %SPARK_HOME%\bin
- Path-> %JAVA_HOME%\bin



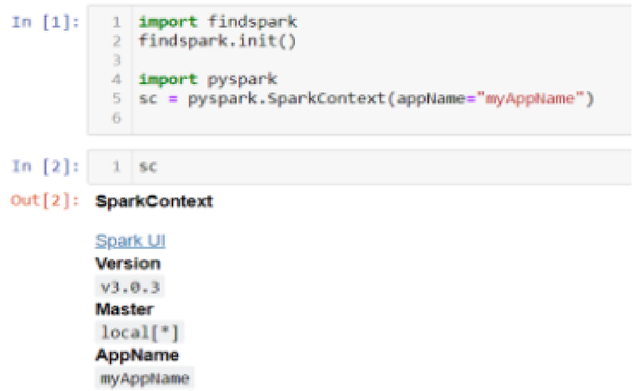
1. Desde el **prompt** de Anaconda ejecuta: “cd C:\spark” y después “pyspark”. Deberías ver algo como lo de la imagen 1.
2. Desde **jupyter notebook** instala findspark con “pip install findspark” y ejecuta el siguiente código.

```
import findspark
findspark.init()
import pyspark
sc = pyspark.SparkContext(appName="myAppName")
sc
```

1



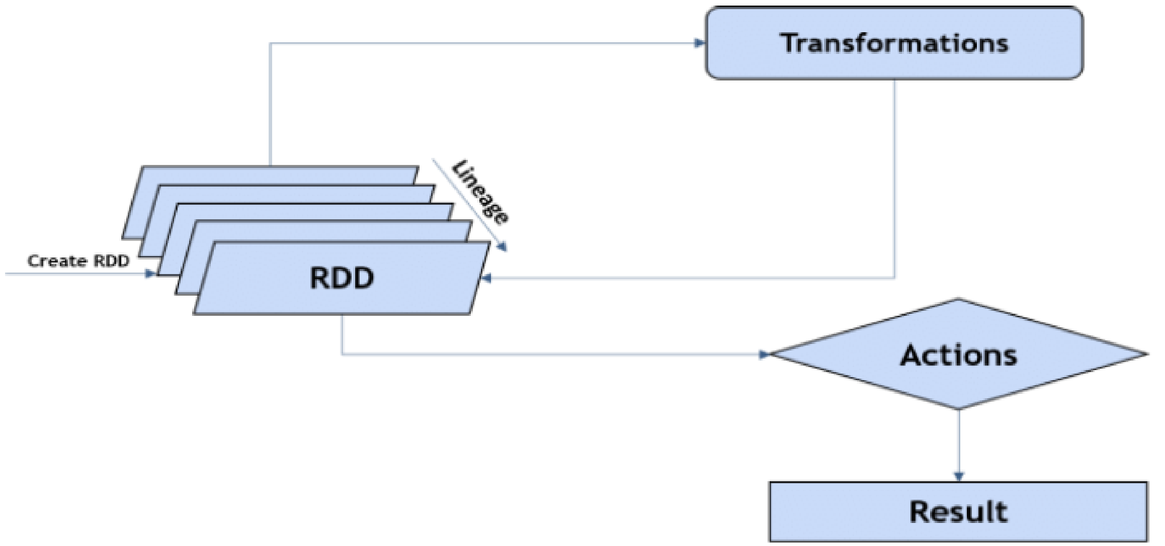
2



Apache Spark RDDs

Los RDD son los componentes básicos de cualquier aplicación Spark. RDD significa:

- **Resiliente**: es tolerante a fallos y es capaz de reconstruir datos en caso de fallo.
- **Distribuido**: los datos se distribuyen entre los múltiples nodos de un clúster.
- **Conjunto de datos**: colección de datos particionados con valores.



Operaciones en RDDs

Con los RDD, puede realizar dos tipos de operaciones:

- **Transformaciones:** estas operaciones se aplican para crear un nuevo RDD.
- **Acciones:** estas operaciones se aplican en un RDD para indicarle a Apache Spark que aplique el cálculo y devuelva el resultado al controlador.

Map: Aplica una función a cada elemento en el RDD y retorna un nuevo RDD.

```
rdd2 = rdd map(lambda x: x * 2)
```

Filter: Retorna un nuevo RDD conteniendo solo los elementos que satisfacen una condición dada.

```
rdd2 = rdd.filter(lambda x: x % 2 == 0)
```

FlatMap: Similar a map, pero cada entrada puede ser mapeada a 0 o más elementos de salida.

```
rdd2 = rdd.flatMap(lambda x: x.split(" "))
```

Union: Retorna un nuevo RDD que contiene los elementos de dos RDDs.

```
rdd3 = rdd.union(otherRDD)
```

Intersection: Retorna un nuevo RDD que contiene solo los elementos que están presentes en ambos RDDs.

```
rdd3 = rdd.intersection(otherRDD)
```

Distinct: Retorna un nuevo RDD que contiene los elementos distintos del RDD original.

```
rdd2 = rdd.distinct()
```

GroupByKey: Agrupa los valores de un RDD de pares clave-valor por clave.

```
rdd2 = rdd.groupByKey()
```

ReduceByKey: Combina los valores de un RDD de pares clave-valor por clave usando una función de reducción.

```
rdd2 = rdd.reduceByKey(lambda x, y: x + y)
```

SortByKey: Retorna un RDD de pares clave-valor ordenado por claves.

```
rdd2 = rdd.sortByKey()
```

Join: Une dos RDDs basados en las claves de cada RDD.

```
rdd3 = rdd.join(otherRDD)
```

Introducción a DataFrames

Los **DataFrames** son de naturaleza **tabular.** Permiten varios formatos dentro de una misma tabla (**heterogéneos**), mientras que cada variable suele tener valores con un único formato (**homogéneos**). Similares a las tablas SQL o a las hojas de calculo.

Column Index

Row Index

	2018	2019	2020
English	85	60	90
Math	73	80	64
Science	98	58	74
French	88	96	87

- Esquema
- Lazy Evaluation
- Optimización
- SQL Queries
- Interoperabilidad
- APIs
- Operaciones
- Integración con Machine Learning

DATA SCIENCE
PARICHAY

```
+-------+---------+-------+-----+
|Book_Id|Book_Name| Author|Price|
+-------+---------+-------+-----+
|      1|      PHP| Sravan|  250|
|      2|      SQL|Chandra|  300|
|      3|   Python| Harsha|  250|
|      4|        R| Rohith| 1200|
|      5|   Hadoop| Manasa|  700|
+-------+---------+-------+-----+
```

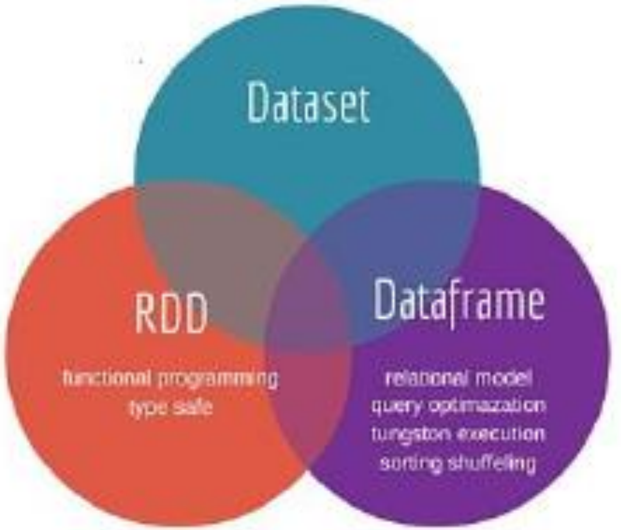
```
root
 |-- Book_Id: long (nullable = true)
 |-- Book_Name: string (nullable = true)
 |-- Author: string (nullable = true)
 |-- Price: long (nullable = true)
```

PySpark Print Schema

Ventajas de los DataFrames

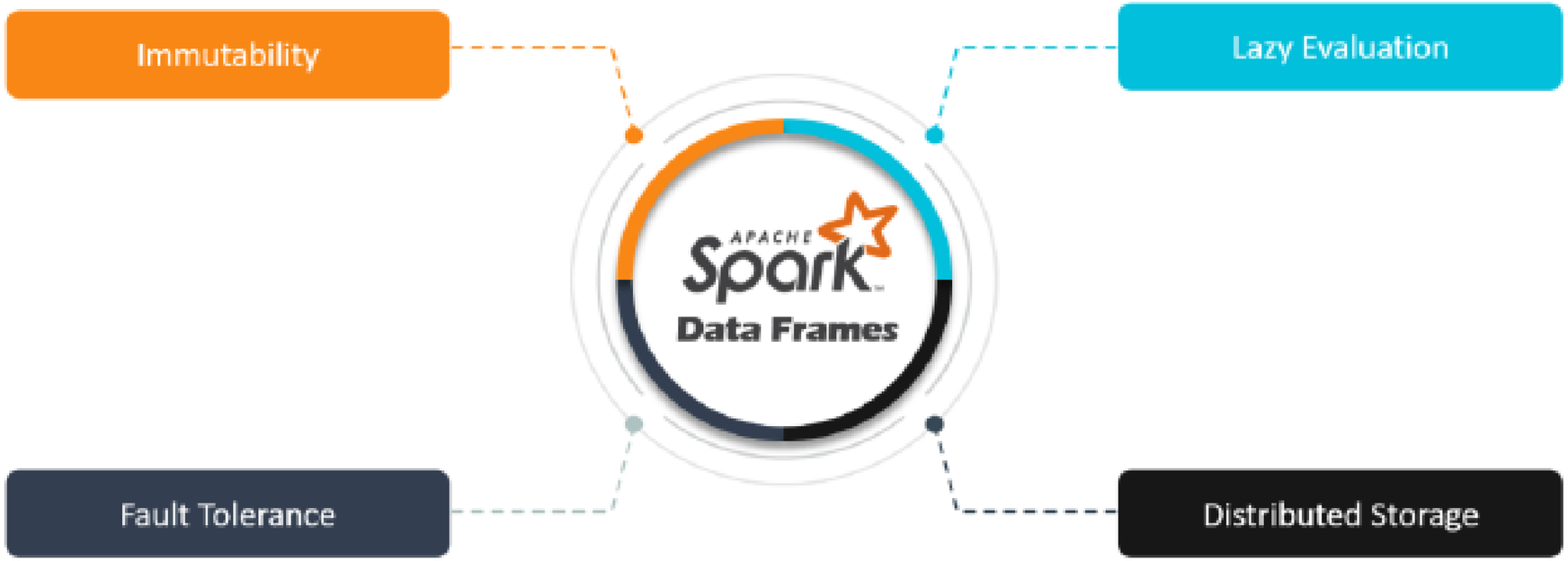
Algunas de las ventajas de trabajar con Dataframes en Spark son:

- Capacidad de procesar una **gran cantidad de datos** estructurados o semiestructurados
- Fácil **manejo de datos** e imputación de valores faltantes
- Múltiples formatos como **fuentes de datos**
- Compatibilidad con **múltiples lenguajes**



Características de los DataFrames

Los **DataFrames** de Spark **se caracterizan** por: ser distribuidos, evaluación perezosa, inmutabilidad y tolerancia a fallos.



Fuentes de datos de DataFrames

Los marcos de datos en Pyspark se pueden crear de varias formas: a través de archivos, utilizando RDDs o a través de bases de datos.