



Tecnológico
de Monterrey

| Educación
Continua

Plataformas y Máquinas Inteligentes en Big Data

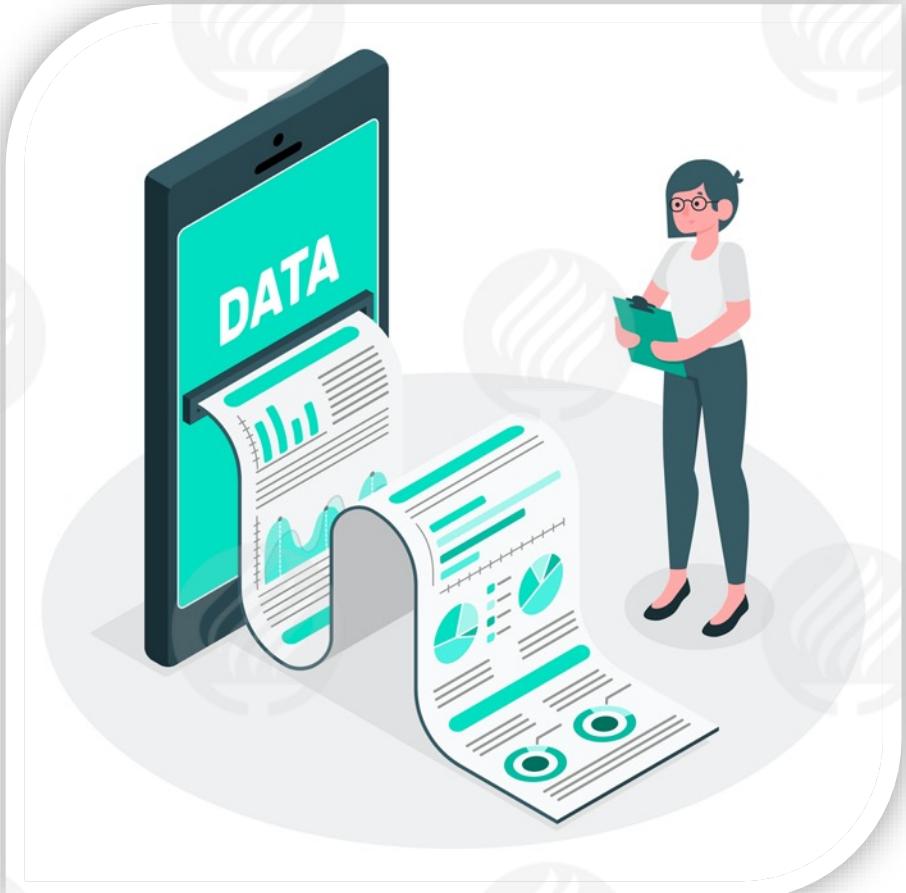
Programas LIVE

SS1 | Aprender

Bienvenida y presentación

- Bienvenidos al Módulo: Plataformas y Máquinas Inteligentes en Big Data
- Y bienvenidos al mundo del procesamiento Big Data, ¡finalmente!

Bienvenida



- Cuando la gente navega en la web, accede a redes sociales, hace compras en línea, realiza transacciones financieras u operaciones en su banca en línea o bien, cuando se realizan transacciones de entre proveedores y manufactureros etc., se generan **enormes cantidades de información** las cuales son **recolectadas y almacenadas**
- Dicha información puede ser utilizada por las organizaciones para lograr **ventajas competitivas** en los mercados y mejoras en su productividad
- Pero ¿de qué manera puede beneficiar a las organizaciones el uso de esta información?, ¿Cómo puede una compañía usar esta información?

Objetivo principal del módulo



Desarrollarás **modelos inteligentes** para el procesado de grandes volúmenes de datos utilizando **Spark**, seleccionando el **modelo adecuado** y analizando la **calidad del modelo**

Objetivos particulares de la sesión



- Analizarás las características del **Big Data** y del **entorno Spark**
- Utilizarás el manejo de Datos en el entorno Spark usando **RDDs** y **Dataframes**

Agenda

1

Sesión Síncrona 1

Aprender

2

Trabajo asíncrono 1

Profundizar | Ruta de Aprendizaje

3

Sesión Síncrona 2

Preparar para Aplicar

4

Trabajo asíncrono 2

Aplicar en el trabajo | Reto



2



3



4



Dinámica de participación y convivencia

Sesión grupal:

- Las dudas durante la sesión se manejarán a través del chat. Éstas serán respondidas en los momentos definidos para ello
- Participar en las dinámicas propuestas durante la sesión de acuerdo con las instrucciones dadas para cada una

Rooms:

- Los equipos se harán de manera aleatoria y serán sólo para las actividades de práctica
- Estar atentos a la notificación de zoom para unirse al room
- Seguir al pie de la letra las instrucciones para la dinámica en rooms
- Informar al moderador si te encuentras sólo en el room para reasignarte
- Te pedimos no salir del room que tienes asignado. Cualquier reasignación será realizada por parte del tutor o staff

Conociendo más de los participantes

- Vamos a crear una nube de ideas sobre lo que representan para ti los conceptos de **Big Data y Spark**
- Escribe en un párrafo lo que te viene a la mente cuando escuchas las palabras **Big Data y Spark**



Introducción

“Procesar información en una computadora poderosa es bueno...

Utilizar muchas de ellas es mejor”

-Raúl Ramírez Velarde

Panorámica de la sesión

1

Sesión Síncrona 1
Aprender

Tema 1: Introducción al Big Data y al Entorno Spark

Tema 2: Manejo de datos en Spark: Dataframes

Cierre de sesión

Panorámica de la sesión

1

Sesión Síncrona 1
Aprender

Tema 1: Introducción al Big Data y al Entorno Spark

Tema 2: Manejo de datos en Spark: Dataframes

Cierre de sesión

Propósito | Tema 1



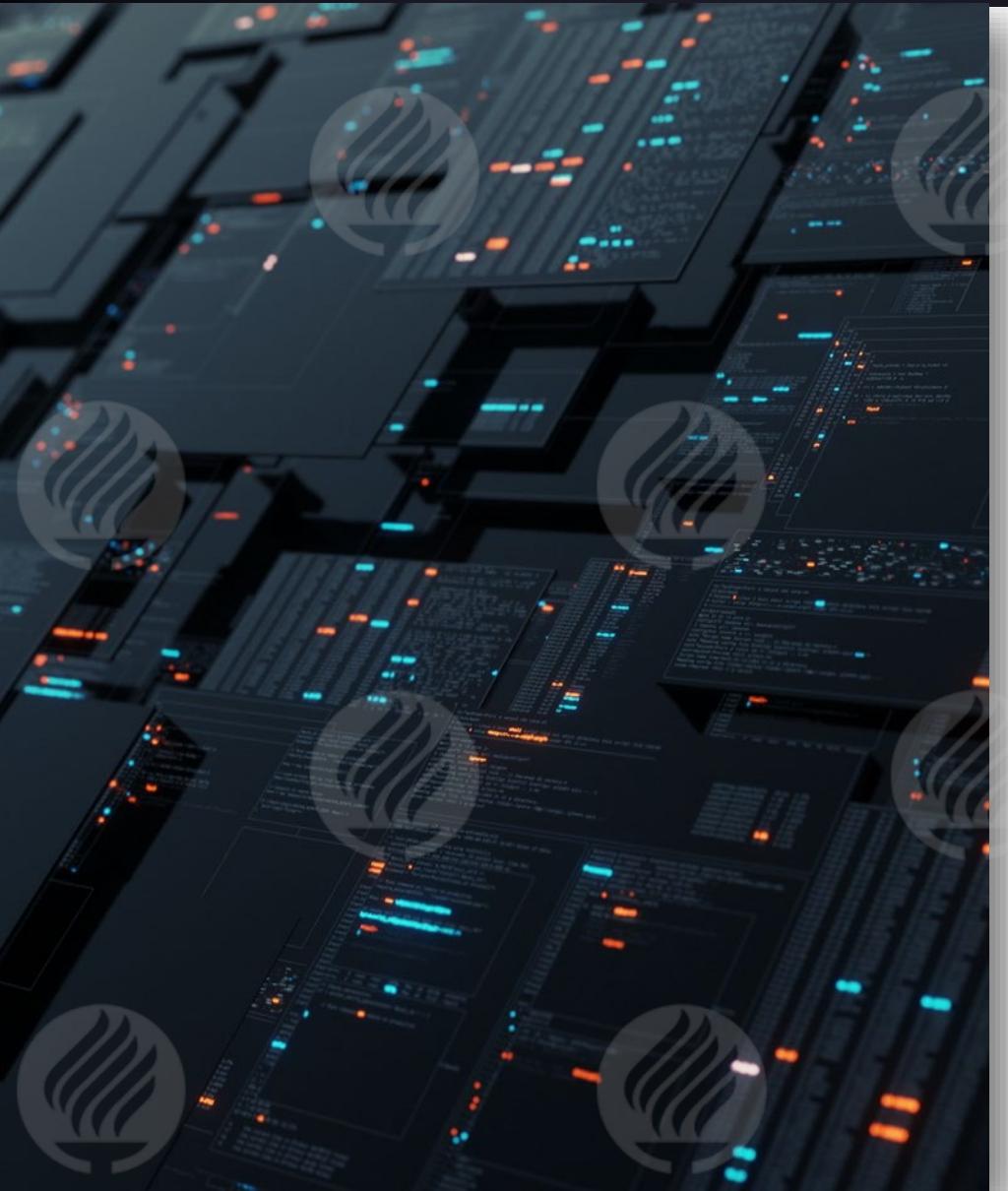
- En Big Data, el procesamiento de datos **no se puede realizar** en una sola estación de trabajo. Es necesario utilizar todo un conjunto de estaciones lo que agrega una **gran dificultad para diseñar, instalar, mantener y administrar** la infraestructura
- Es necesario **comprender los componentes** del ecosistema Tecnológico Big Data, principalmente el **entorno Spark**, para tomar decisiones que tengan sentido desde el punto de vista operativo de la organización. No comprender esta complejidad le **negará los beneficios** que espera recibir de su inversión

Introducción al Big Data y Spark | Tema 1



- Se denomina **Big Data** cuando la información no puede ser procesada con las tecnologías tradicionales de manejo de datos
- La definición de Big Data es una colección de datos que:
 1. Es de gran **volumen**
 2. Llega a gran **velocidad**
 3. Y es muy **variada**
- A esto se llama las **3 V's**

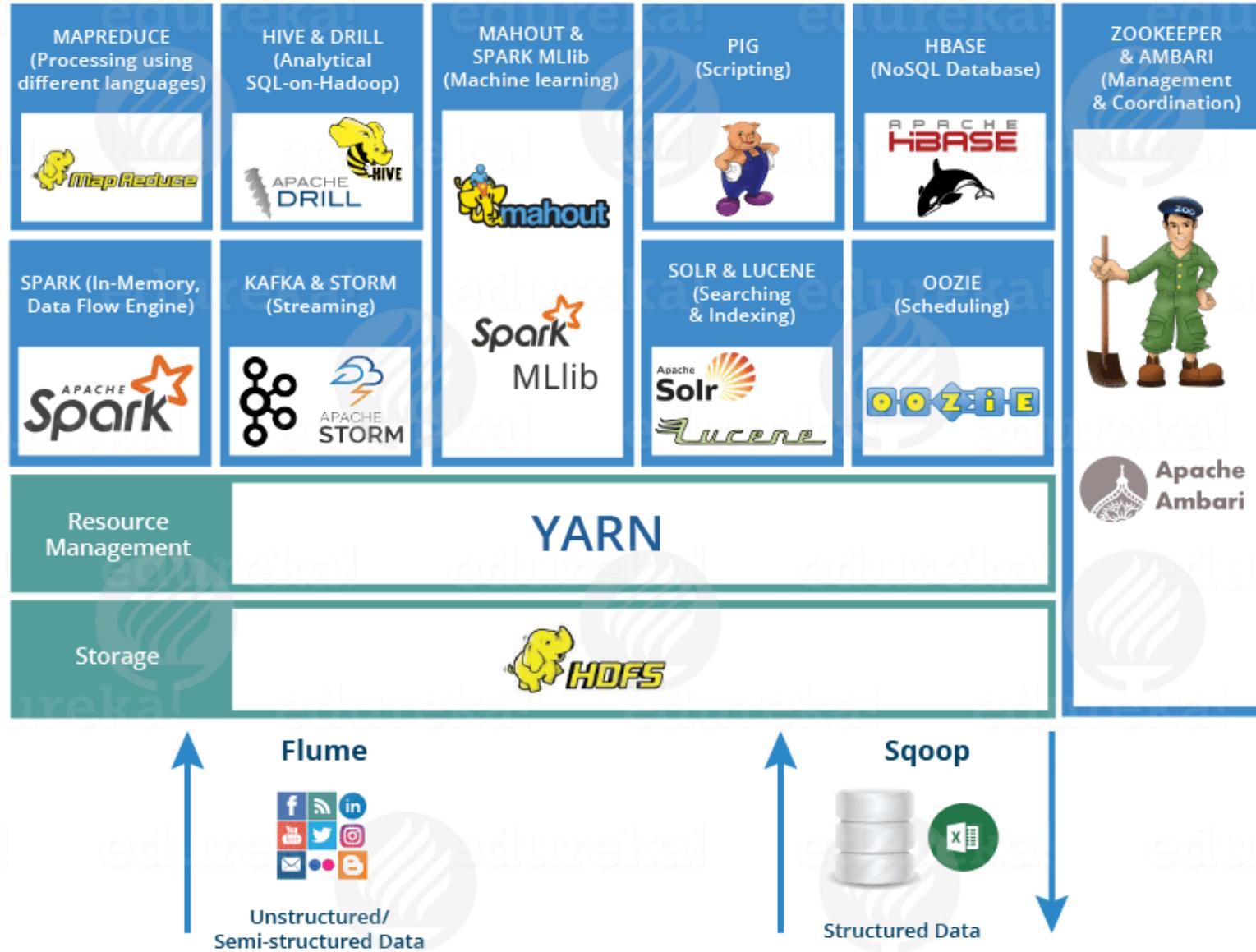
Introducción al Big Data y Spark | Tema 1



Las **tecnologías Big Data** se refieren a las herramientas de software y hardware especialmente diseñadas para **extraer, transformar, procesar y analizar información de conjuntos de datos complejos**

- Las **tecnologías operacionales** se refieren al **tratamiento de información que se genera día con día**. Esta información puede generarse en transacciones en línea, redes sociales, datos particulares de alguna organización, etc.
- Las **tecnologías Big Data analíticas** son más complicadas y avanzadas que las operacionales. Con estas tecnologías se toman las decisiones cruciales de respuesta de **tiempo real y de predicción**

Introducción al Big Data y Spark | Tema 1



El Ecosistema
Big Data

Tecnologías para la administración de *big data*

La **administración del *big data*** se lleva a cabo en plataformas manejadas por vendedores de tecnología de información que combinan muchas de las tecnologías en un solo paquete, **construidos sobre la nube**:

- Amazon EMR (antes Elastic MapReduce)
- Cloudera Data Platform
- Google Cloud Dataproc
- HPE Ezmeral Data Fabric (formerly MapR Data Platform)
- Microsoft Azure HDInsight
- IBM Cloud

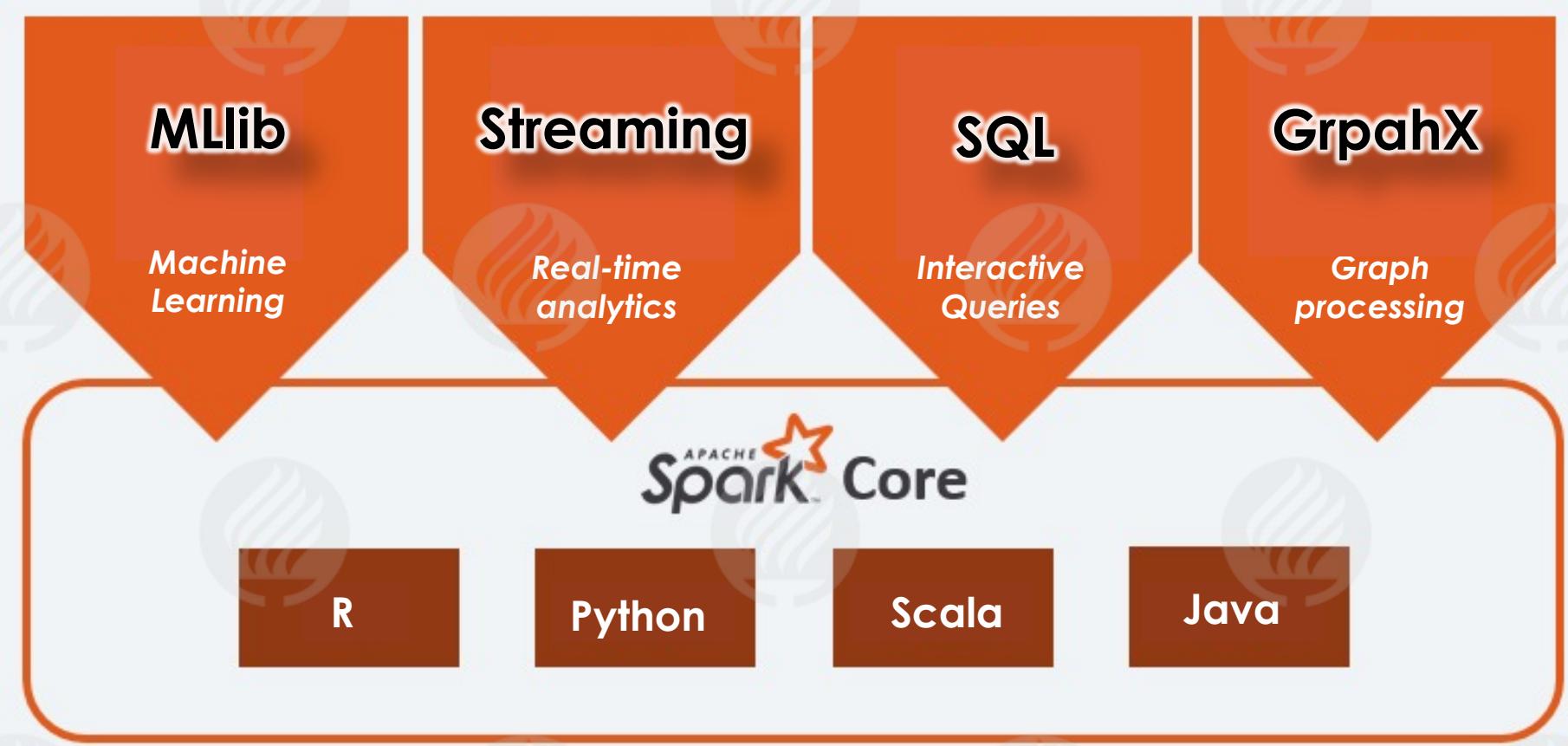
Para aquellas compañías que desean administrar sus propios sistemas, ya sea en sus propias instalaciones como en la nube, pueden implantar un conjunto diverso de tecnologías utilizando servicios de **administración de clusters de procesamiento** como **Databricks, Kubernetes, Mesos y Yarn**.

Spark

- Es un **conjunto de herramientas** de procesamiento de big data desarrollado por Apache que incluye opciones de **streaming**, uso del lenguaje **SQL** en datos **no estructurados**, herramientas de **analítica** y creación y procesamiento de **grafos** y más
- Las aplicaciones Spark se pueden programar usando la mayoría de los lenguajes de desarrollo más conocidos, como: **R, Python, Scala y Java**
- Aunque Spark funciona sobre Hadoop, **Spark es muchas veces más rápido** de MapReduce



Introducción al Big Data y Spark | Tema 1



El
Ecosistema
Spark

Introducción al Entorno Spark | Tema 1



- La principal característica de Spark es su **velocidad** de procesamiento
 - ✓ Para operaciones en disco, puede ser 10 veces más veloz que Hadoop, y para operaciones en memoria puede ser hasta 100 veces más rápido
- Puede ser usado por sí solo o en plataformas como **Hadoop, EC2, YARN y Mesos**, así como acceder a datos desde **Cassandra, Alluxio, HDFS, Hive** y cientos de otras bases de datos
- Apache Spark responde a una **arquitectura jerárquica de maestro/esclavo**
 - ✓ El **Spark driver** es el nodo **maestro** que convierte el código del usuario en diversas tareas que pueden ser distribuidas entre los nodos **workers** y entrega los resultados de los datos

Núcleo (Spark Core)

Es la **base de ejecución** para todos los procesamientos de **datos paralelos**. Provee **cómputo en memoria rápida** y de **referenciar las bases de datos** localizadas en almacenamientos externos. Se ocupa de las siguientes funciones:

- Administración de la memoria y recuperación de fallas
- Planificación, distribución y monitoreo de tareas en un cluster
- Interacción con sistemas de almacenamiento
- Abstracción de datos y RDD

El núcleo de Spark también es la **base funcional** de las bibliotecas como **Spark SQL, Spark Streaming, ML, MLlib y GraphX**.



Introducción al Entorno Spark | Tema 1



RDD

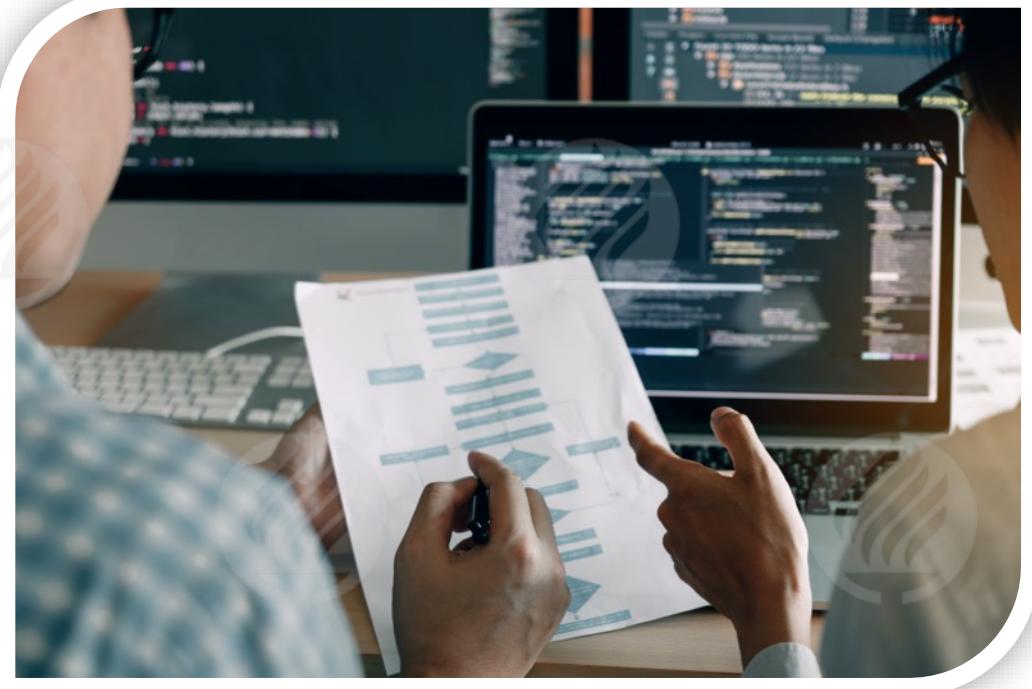
- Los **Resilient Distributed Datasets** o RDD es un concepto que podría traducirse como **“conjuntos de datos resilientes distribuidos”**, son una estructura fundamental de Spark para **Big Data**
- Es una **abstracción de programación** que consiste de **colecciones de elementos almacenados** en memoria con tolerancia a fallas que pueden ser distribuidos entre múltiples nodos de un **cluster** y **procesador paralelamente**, para después volverlos a unir en el nodo núcleo. Esto permite que su procesamiento sea **rápido y escalable**

Introducción al Entorno Spark | Tema 1



Spark SQL

- Quizá es la **interfaz más utilizada** por desarrolladores de Spark para crear aplicaciones
- Se centra en el **procesamiento de datos estructurados** y permite **consultar macrodatos desde otras fuentes**, como nos enseña José Manuel Incera Rosas en su curso de SQL para el análisis de datos



Spark Streaming

- Permite procesar **flujos de datos escalables y tolerantes** a fallas casi en tiempo real. Los datos pueden ser digeridos de Kafka, Flume, and HDFS que es Hadoop Distributed File System
- Los datos pueden ser procesados usando **algoritmos complejos** y enviados a sistemas de archivos, bases de datos y tableros de tiempo real

ML/MLIB

- Son unas **bibliotecas de algoritmos** para realizar operaciones enfocadas al **Machine Learning**
 - ✓ **MLIB** es para procesar **RDDs**
 - ✓ **ML** es para procesar **Spark Dataframes**
- Contiene una gran cantidad de algoritmos de **clasificación, regresión, clustering y filtrado colaborativo**
- **ML** en Python, es similar a la muy conocida **Scikit-Learn**
 - ✓ También incluye herramientas para crear, evaluar y ajustar ensamblajes (pipelines) de machine learning, es decir, construir un proceso o workflow que puede ser exportado a otras plataformas y puesto disponible en un servicio Web (URL)



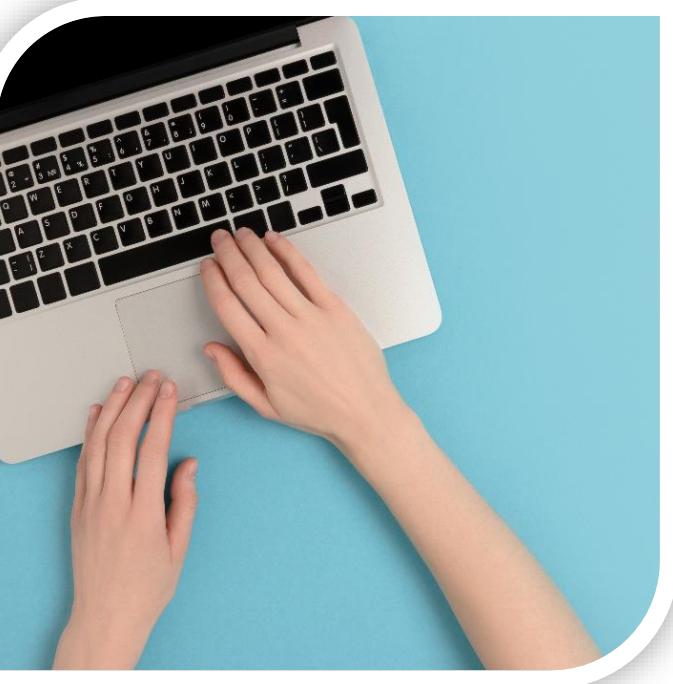
Introducción al Entorno Spark | Tema 1



GraphX

- Además de ofrecer una **serie de operaciones** para la **manipulación de grafos**, provee algunos algoritmos de grafos como PageRank. GraphX **unifica el proceso ETL** (Extract, Transform and Load), **análisis exploratorio, y cómputos iterativos dentro de un mismo sistema**

Actividad | Aprendizaje activo



- Inicio de sesión **Spark** en Google Colab y Kaggle:

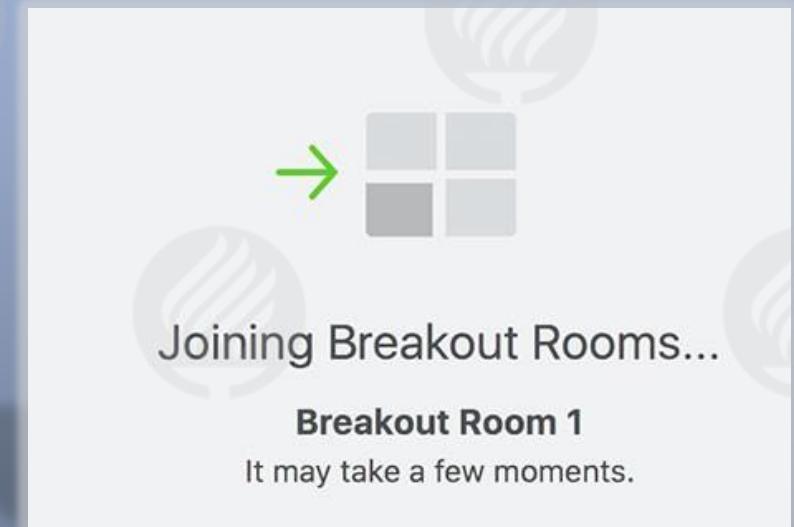
```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://downloads.apache.org/spark/spark-2.4.8/spark-
2.4.8-bin-hadoop2.7.tgz
!tar xf spark-2.4.8-bin-hadoop2.7.tgz
!pip install -q findspark

#All imports
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-2.4.8-bin-hadoop2.7"

import findspark
findspark.init()
from pyspark import SparkContext, SparkConf,SQLContext
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
spark.version
```

Actividad | BreakOut Rooms (BOR)

- 1. Al unirte,
aparecerá un
mensaje**
- 2. Después estarás
con un grupo
pequeño de
compañeros**



Reflexión | Tema 1



- Ahora que conoces un poco más acerca de **Spark**, ¿crees que aprender a utilizar **tecnologías big data** como Spark te abre puertas en tu carrera profesional?
- ¿Crees que las **tecnologías big data** puedan ayudar a resolver los retos que enfrenta tu organización?, ¿Qué reto escogerías? ¿Cómo aplicarías las tecnologías big data en este reto?



- Big data permite procesar **grandes volúmenes** de información que provienen de diversas fuentes en formatos variados, estructurados y no estructurados, de una forma **veloz y escalable**
- El principal conjunto de herramientas para el aprovechamiento de big data es el ecosistema **Spark**.
- Spark es **rápido, escalable y resistente**
- Spark es un conjunto de herramientas que permite la **minería, transformación, manipulación, búsqueda de información estática y dinámica**

Cierre: Concepto clave | Tema 1

**Spark es el ecosistema de big data
más usado actualmente.**

Panorámica de la sesión

1

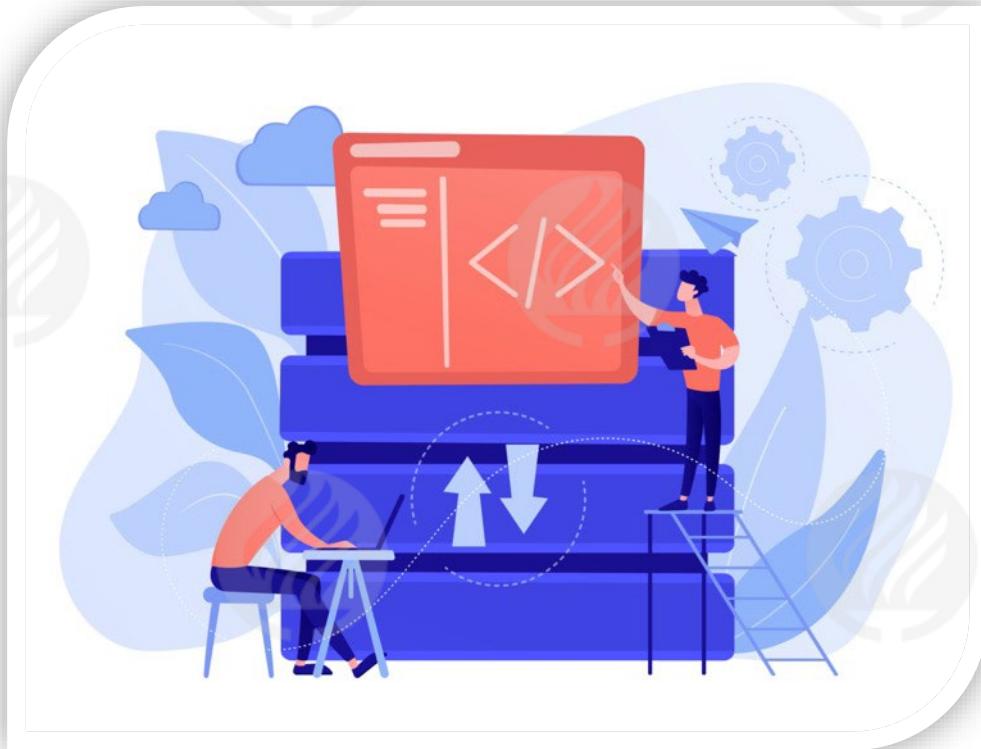
Sesión Síncrona 1
Aprender

Tema 1: Introducción al Big Data y al Entorno Spark

Tema 2: Manejo de datos en Spark: Dataframes

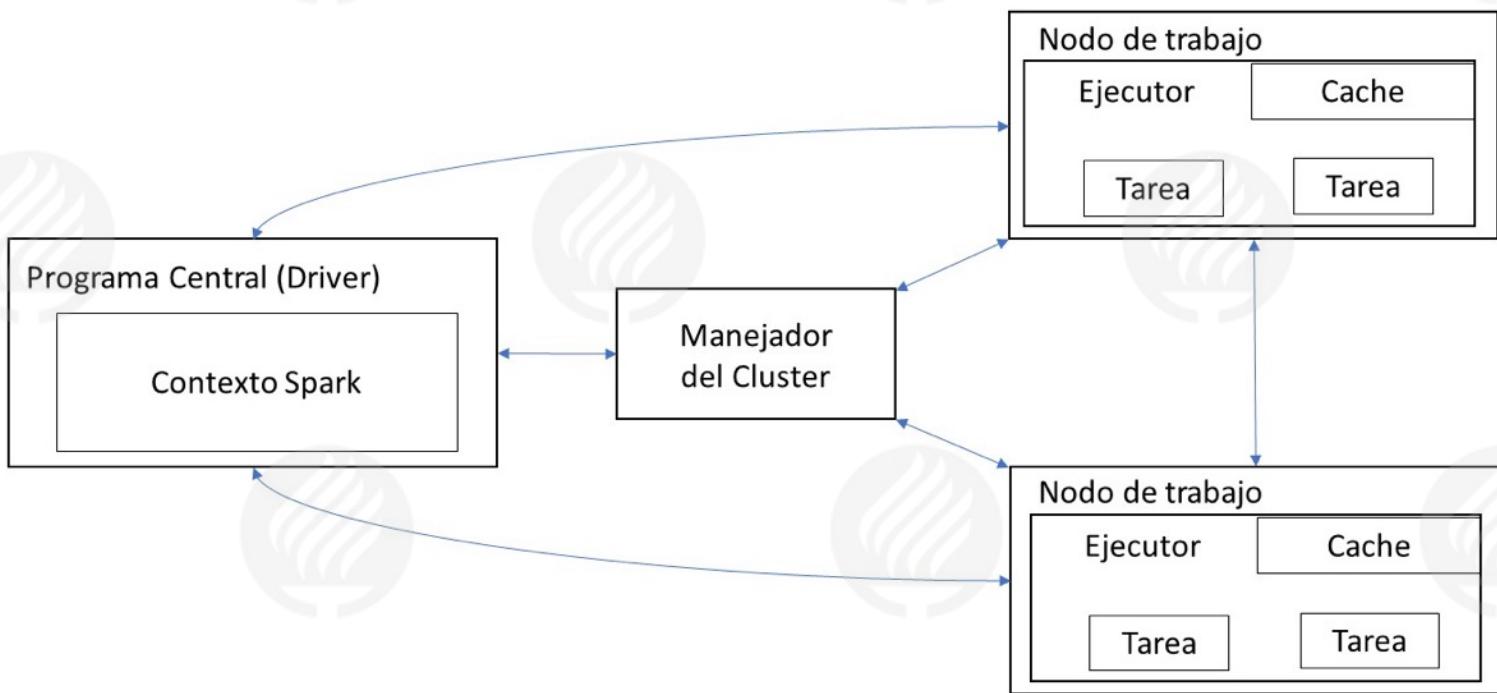
Cierre de sesión

Propósito | Tema 2



- **Spark** contiene bibliotecas y sistemas que permiten realizar búsquedas **SQL**, aplicar algoritmos de **machine learning**, realizar cálculos sobre **grafos** y procesar y reproducir datos en **stream**.
- Spark se programa en **R, Scala, Python y Java**
- Es necesario comprender las características de Spark para poder desarrollar aplicaciones que permitan extraer el mayor valor posible a los recursos que se tienen disponibles y la inversión que implican

Manejo de Datos en Spark | Tema 2



Arquitectura Spark

- Un entorno Spark consiste en un programa llamado **Central o Driver** y un grupo de nodos de trabajo llamados **Ejecutores**. Crea un **SparkContext** que coordina las tareas a ejecutar
- El **manejador del cluster** se encarga de administrar el software instalado en el cluster y llevar a cabo las actualizaciones necesarias



Inicio de Sesión Spark

- En Spark 1.x se tenían tres puntos de entrada: **SparkContext**, **SQLContext** y **HiveContext**. A partir de la versión 2.x se creó un nuevo punto de acceso llamado **SparkSession** que, en esencia, combina las funcionalidades de todas las anteriores
- **SparkContext** habilita al proceso central de Spark a establecer una comunicación con el cluster y el administrador de recursos para coordinar las acciones y ejecutar tareas. **SparkContext** también habilita la creación de los contextos **SQL** y **Hive**



Inicio de Sesión Spark

- **SQLContext** es un punto de entrada para el módulo **SparkSQL** que se utiliza para el manejo de información estructurada. Cuando se inicializa este contexto es posible realizar búsquedas o queries usando **SQL** en los conjuntos de datos
- **HiveContext** se utilizaba para comunicarse con el sistema de almacenamiento Hive. Sin embargo, a partir de la versión 2.x de Spark. **HiveContext** es redundante porque el soporte a **Hive** ya viene incluido en **SparkSession**

Manejo de Datos en Spark | Tema 2

Inicio de Sesión Spark

- **SparkSession** reemplaza **SQLContext** y **HiveContext** al mismo tiempo que da el acceso inmediatamente a **SparkContext**
- Para crear un **SparkSession** se utiliza el siguiente código:

```
spark = SparkSession \  
.builder \  
.getOrCreate()
```

- Si se desean un **SparkContext** y un **SQLContext** se puede utilizar el siguiente código:

```
sc = SparkContext.getOrCreate(SparkConf().setMaster("local[*]"))  
  
sqlContext = SQLContext(sc)
```

Manejo de Datos en Spark | Tema 2

Funciones Lambda

- Son funciones que no tienen nombre y que se utilizan, junto con la sentencias “**map**” y “**filter**”
- La sentencia “**map**” transforma los datos. La sentencia “**filter**” selecciona subconjuntos de datos
- Dado un arreglo de números llamado “**my_list**”, usamos una función lambda para obtener el cuadrado de cada uno de los elementos en la secuencia de números:

```
my_list = [1, 2, 3, 4, 5]
squared_my_list = list(map(lambda x: x*x, my_list))
squared_my_list
```

[1, 4, 9, 16, 25]

Manejo de Datos en Spark | Tema 2

Funciones Lambda

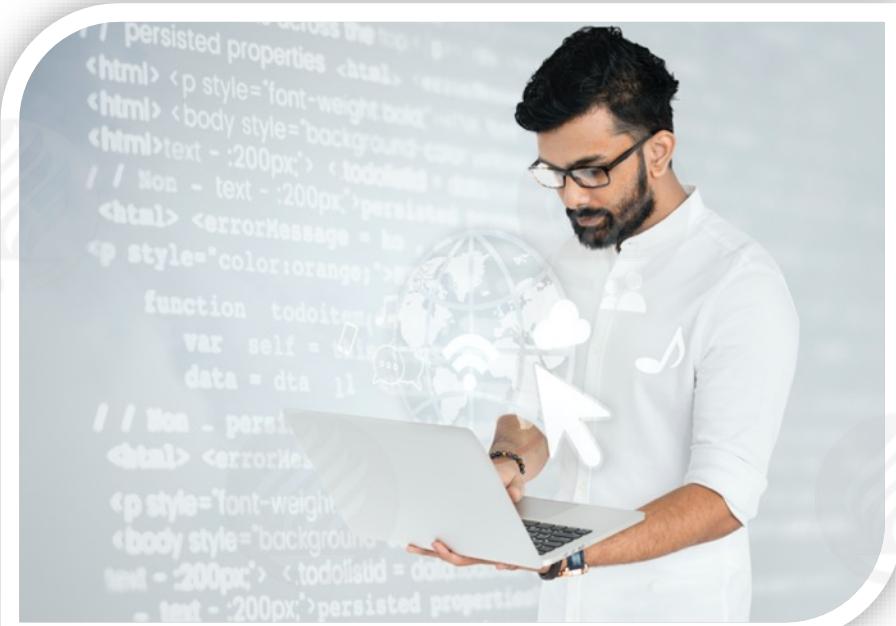
- En el siguiente ejemplo, se extraen sólo los datos pares de la misma secuencia de números “**my_list**”. Recuerda que “**x%2 == 0**” indica que el residuo de dividir el número entre 2 debe ser cero:

```
filtered_my_list = list(filter(lambda x: (x%2 != 0),  
    my_list))  
filtered_my_list
```

[1, 3, 5]

PySpark Dataframes:

- Se usa una sesión Spark (**SparkSession**) para **crear** dataframes, **registrar** dataframes, **leer** dataframes del almacenamiento y **ejecutar** comandos **SQL**
- Un Spark Dataframe es una **colección distribuida** de datos organizados en columnas con nombre
- Es **conceptualmente equivalente** a tabla en una base de datos relacional o a un **dataframe de R o Pandas en Python**, pero con una rica colección de **optimizaciones** para el funcionamiento distribuido que son completamente **transparentes** para el programador



Manejo de Datos en Spark | Tema 2

Creando un Dataframe de un RDD:

- Los Spark Dataframes se pueden crear usando **métodos parallelize y spark.createDataFrame**. Los dataframes usan el método “**show**” para mostrar su contenido

```
sample_list = [ ("Mona", 23), ("Lisa", 29), ('Leonardo', 37),  
('Piero', 41)]  
rdd = sc.parallelize(sample_list)  
df_names = spark.createDataFrame(rdd, schema=[ 'Name', 'Age'])  
type(df_names)  
  
pyspark.sql.dataframe.DataFrame
```

Manejo de Datos en Spark | Tema 2

Crear un Spark Dataframe de un archivo externo:

- Se utiliza “**spark.read.csv**”
- Por defecto, el parámetro “**header**” está puesto como verdadero, lo que indica que el primer renglón del archivo será tomado como encabezado y de ahí se obtendrán los nombres de las columnas
- Por defecto también, todas las columnas son leídas como StringType, pero al poner como verdadero el valor “**inferSchema**” se deriva el verdadero tipo de datos de las columnas y la estructura de los datos, lo que crea un conjunto de datos Spark SQL
- Por ejemplo, siguiendo con el código anterior:

```
path_people="drive/My Drive/Colab Notebooks/.../people.csv"  
df_people = spark.read.csv(path_people, header=True, inferSche  
ma=True)
```

Manejo de Datos en Spark | Tema 2

Crear un Spark Dataframe de un archivo externo:

- Veamos el contenido:

```
df_people.show(5)
```

```
+---+-----+-----+-----+-----+
| _c0|person_id| name| sex|date of birth|
+---+-----+-----+-----+-----+
| 0| 100| Penelope Lewis| female| 1990-08-31|
| 1| 101| David Anthony| male| 1971-10-14|
| 2| 102| Ida Shipp| female| 1962-05-24|
| 3| 103| Joanna Moore| female| 2017-03-10|
| 4| 104| Lisandra Ortiz| female| 2020-08-05|
| 5| 105| David Simmons| male| 1999-12-30|
+---+-----+-----+-----+
only showing top 10 rows
```

Manejo de Datos en Spark | Tema 2

Crear un Spark Dataframe de un archivo externo:

- Veamos las características de los datos:

```
df_people.count()
```

```
100000
```

```
print(len(df_people.columns))
```

```
5
```

```
df_people.printSchema()
```

```
root
```

```
| -- _c0: integer (nullable = true)
```

```
| -- person_id: integer (nullable = true)
```

```
| -- name: string (nullable = true)
```

```
| -- sex: string (nullable = true)
```

```
| -- date of birth: string (nullable = true)
```

Manejo de Datos en Spark | Tema 2

Eliminando valores nulos y eliminando y seleccionando columnas:

- Para eliminar todos los renglones en lo que hay valores no numéricos (**llamados nulos**):

```
df.na.drop()
```

- Para eliminar columnas también se utilizan de la sentencia “**drop()**”, poniendo los nombres de las columnas entre comillas:

```
df_people.drop("name", "date of birth").printSchema()
```

- Y también es buena práctica **eliminar** todos los renglones duplicados:

```
df_people_nodup = df_people.dropDuplicates()
```

Manejo de Datos en Spark | Tema 2

Eliminando valores nulos y eliminando y seleccionando columnas:

- Si se desea hacer un nuevo **Spark Dataframe** copiando sólo algunas columnas se utiliza el comando **select**:

```
df_people_sub = df_people.select('name', "sex", 'date of birth')
```

- Ya hemos visto, para filtrar los datos en base a una condición se utiliza la sentencia **“filter”**:

```
df_people_female = df_people.filter(df_people.sex == "female")
```

```
df_people_male = df_people.filter(df_people.sex == "male")
```

Manejo de Datos en Spark | Tema 2

Agrupando Valores:

- También es posible **agrupar** según los valores de alguna columna:

```
df_people_sex = df_people.groupby("sex")
```

```
df_people_sex.count().show()
```

```
+-----+  
| sex|count|  
+-----+  
| null| 1920|  
| female|49014|  
| male|49066|  
+-----+
```

Manejo de Datos en Spark | Tema 2

Ordenando Valores:

- Ordenar los valores de alguna columna:

```
df_people.orderBy("date of birth").show(3)
```

```
+-----+-----+-----+-----+
| _c0|person_id| name| sex|date of birth|
+-----+-----+-----+-----+
| 57359| 57459| Sharon Perez| female| 1899-08-28|
| 62233| 62333| Martina Morison| female| 1901-04-21|
| 96318| 96418| Lisa Garrett| female| 1901-05-09|
+-----+-----+-----+-----+
only showing top 3 rows
```

Manejo de Datos en Spark | Tema 2



pySpark SQL:

- Es posible utilizar los métodos **select, filter, select, count, groupby y orderby** en Spark Dataframes para **manipular los datos**
- Existen operaciones **equivalentes** usando **SparkSQL en Python**

Manejo de Datos en Spark | Tema 2

PySpark SQL:

- Una vez que se ha leído un **archivo de datos** dentro de un Spark Dataframe, se requiere **crear una vista** que se convertirá en nuestra tabla de base de datos:

```
path_people="/content/drive/MyDrive/Colab Notebooks/.../people.csv"
```

```
df_people = spark.read.csv(path_people, header=True, inferSchema=True)
```

```
df_people.createOrReplaceTempView("people")
```

- Nótese que no fue necesario crear un **SQLContext**

Manejo de Datos en Spark | Tema 2

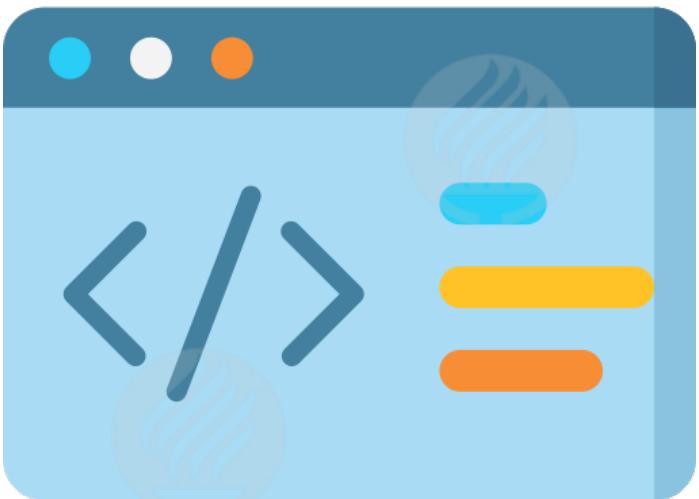
PySpark SQL:

- Para ver la **estructura de la tabla y los tipos de datos** que contiene se usa “**describe**”:

```
query='DESCRIBE people'  
spark.sql(query).show(10)
```

```
+-----+-----+-----+  
| col_name|data_type|comment|  
+-----+-----+-----+  
| _c0| int| null|  
| person_id| int| null|  
| name| string| null|  
| sex| string| null|  
| date of birth| string| null|  
+-----+-----+-----+
```

Manejo de Datos en Spark | Tema 2



pySpark SQL:

- Los **queries simples** se pueden encasillar con **comillas dobles** o **comillas sencillas**
- Queries mas complicados se encasillan en **triple comilla**
- Esto se hace para utilizar las comillas dobles o comillas sencillas para **valores constantes texto** y usar las comillas invertidas para **columnas que tienen espacios en blanco**

Manejo de Datos en Spark | Tema 2

PySpark SQL:

- En este ejemplo se busca el nombre y la fecha de nacimiento de todos los hombres y que se ordenen por fecha de nacimiento

```
query="""SELECT name, `date of birth` FROM people WHERE sex=="male" ORDER BY `date of birth`"""
df_people_names = spark.sql(query)
df_people_names.show(5)
```

```
+-----+-----+
| name|date of birth|
+-----+-----+
| Tyler Walton| 1903-07-14|
| Daniel Naiman| 1903-11-07|
| John Merritt| 1906-11-04|
| Roger Watkin| 1907-12-08|
| Tim Makris| 1909-07-11|
+-----+-----+
only showing top 10 rows
```

Manejo de Datos en Spark | Tema 3

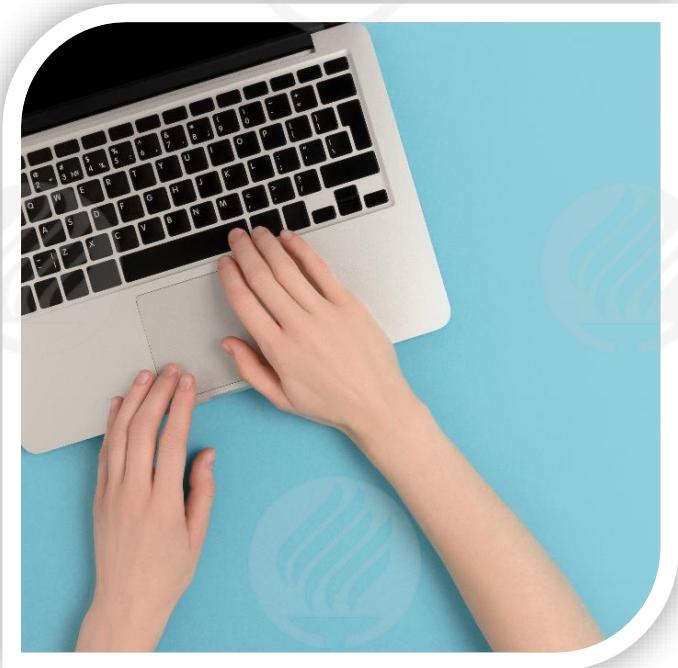
PySpark SQL:

- Para obtener una **tabla de frecuencias**, se utiliza la función interna “**COUNT**” y la sentencia SQL “**GROUP BY**”. El siguiente query encuentra cuántos hombres y cuántas mujeres nacieron entre enero de 1903 y diciembre de 1911:

```
query='SELECT sex,COUNT(sex) FROM people WHERE birth BETWEEN "  
1903-01-01" AND "1911-12-31" GROUP BY sex'  
df_people_1903_1906_sex = spark.sql(query)  
df_people_1903_1906_sex.show()
```

```
+-----+  
| sex|count (sex) |  
+-----+  
| female| 7|  
| male| 10|  
+-----+
```

Actividad | Aprendizaje activo



- Indica cuál es el resultado de los siguientes dos ejercicios. Explícalos con tus propias palabras:
 1. Se tiene el siguiente código:

```
year_report = [2011, 2012, 2013, 2013, 2014, 2018, 2019, 2020]
years_passed = list(map(lambda x: 2021-x, year_report))
```

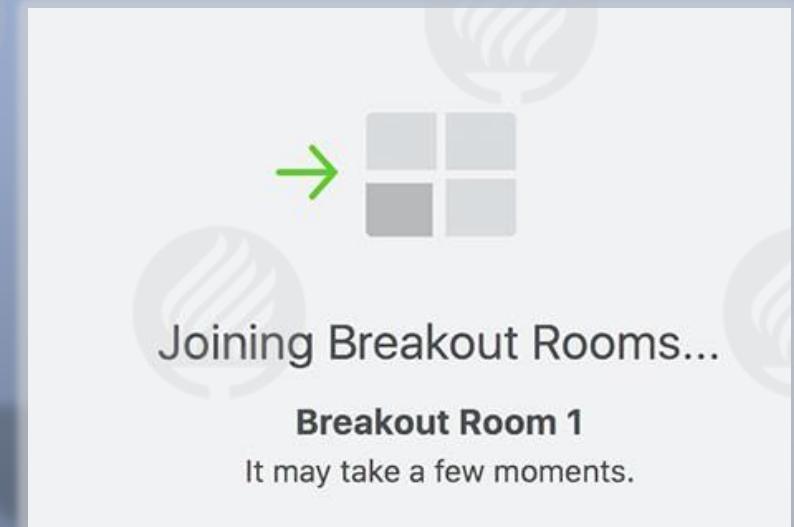
years_passed
 2. Se tiene el siguiente código:

```
year_report = [2011, 2012, 2013, 2013, 2014, 2018, 2019, 2020]
newer_reports = list(filter(lambda x: x>2015, year_report))
```

newer_reports
- Convierte los datos en un **dataframe** de Spark y realiza las mismas operaciones (usando funciones de los dataframes)
- Ahora **utiliza SQL** para obtener los mismos resultados

Actividad | BreakOut Rooms (BOR)

- 1. Al unirte,
aparecerá un
mensaje**
- 2. Despues estarás
con un grupo
pequeño de
compañeros**



Reflexión | Tema 2



- ¿Cómo se comparan los **dataframes de Spark** con los **dataframes de Pandas y R**?
- ¿Crees que los dataframes de Spark te **facilitarán el desarrollo o lo dificultarán**?



- Los RDD son **colecciones de objetos distribuidos** que son procesados por una colección de computadoras simultáneamente
- Las **funciones lambda** de Python se pueden aplicar a los RDDs, logrando que transformaciones sofisticadas se puedan realizar en **paralelo**
- Spark ha desarrollado una forma especializada de objetos distribuidos llamados **Spark Dataframes** que comparten muchas de las características y usos de los **dataframes encontrados en R y en Pandas de Python**
- Spark permite realizar operaciones **SQL** sobre conjunto de **datos distribuidos**

Cierre: Concepto clave | Tema 2

Los dataframes de Spark permiten procesar datos de forma distribuida usando técnicas de programación comunes y bien conocidas.

Cierre de la sesión

- Reflexión: **Big data y mi entorno de trabajo. Big data y mi carrera profesional**
- Preguntas y Respuestas

Cierre | Sesión Sincrónica 1 [Aprender]



- Es importante **conocer** las características, los alcances de la tecnología big data, sus usos y sus aplicaciones
- Es muy relevante conocer las **formas** en que se puede construir una infraestructura big data: comprarla o arrendarla
- El procesamiento de datos en paralelo presenta **retos prácticos y conceptuales**. Sin embargo, una vez superados ofrece **oportunidades** redituables para una organización

Cierre: Concepto clave

El procesamiento paralelo de datos es fundamental para big data

Siguientes pasos | Semana de trabajo asíncrono [Profundizar]



1 Sesión Síncrona 1
Aprender

2 Trabajo asíncrono 1
Profundizar | Ruta de Aprendizaje

3 Sesión Síncrona 2
Preparar para Aplicar

4 Trabajo asíncrono 2
Aplicar en el trabajo | Reto

- En la siguiente sección realizaremos **análisis estadísticos numéricos y gráficos** en Spark y crearemos nuestros primeros **modelos predictivos**



Tecnológico de Monterrey | 2021

Prohibida la reproducción total o parcial de esta
obra sin expresa autorización del Tecnológico
de Monterrey

Gracias | Programas LIVE