



# DIGITAL TALENT SCHOLARSHIP 2019

Big Data Analytics



## Pig, dan Hive

Oleh: Imam Cholissodin | [imamcs@ub.ac.id](mailto:imamcs@ub.ac.id), Putra Pandu Adikara, Sufia Adha Putri

Asisten: Guedho, Sukma, Anshori, Aang dan Gusti

**Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya (UB)**

# Pokok Pembahasan

- Review Hadoop MapReduce
- Tentang Pig
- Tentang Hive
- Perbedaan Pig dan Hive
- Tugas



# Hadoop MapReduce

- Kita semua tahu, Hadoop menggunakan MapReduce untuk memproses dan menganalisis big data.
- Processing big data menghabiskan banyak waktu jika menggunakan cara-cara tradisional.
- Hadoop MapReduce telah terbukti dapat digunakan untuk memproses data dengan cepat.

Before



Pemrosesan Big Data  
Membutuhkan waktu lama

After



Pemrosesan Big Data  
Dgn MapReduce Lebih Cepat

# Hadoop MapReduce

- Tetapi MapReduce implementasinya harus menggunakan Java, yang kodingnya cukup panjang dan komplek.
- Sehingga MapReduce tidak cocok untuk non-programmer atau ketika programmer butuh waktu cepat dalam menuliskan kodenya. Untuk mengatasi problem tersebut digunakanlah Hive dan Pig.

Before



Pemrosesan Big Data  
Membutuhkan waktu lama

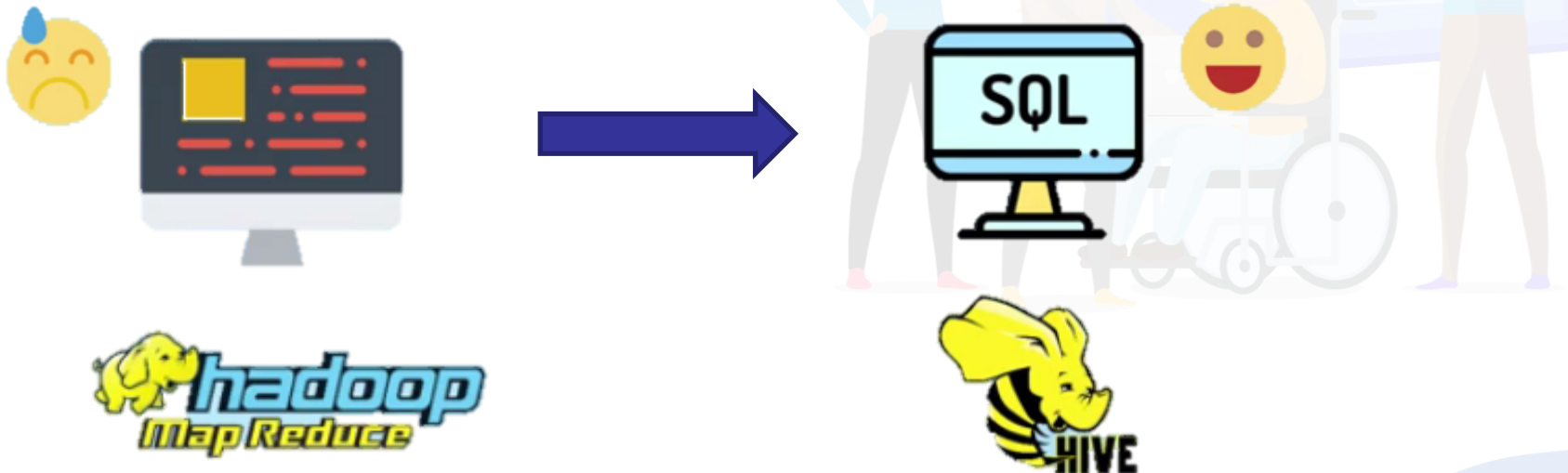
After



Pemrosesan Big Data  
Dgn MapReduce Lebih Cepat

# Kenapa Butuh Hive?

- Problem base, misal pada facebook, tidak semua pekerjaanya paham dengan high-level programming. Dan ketika butuh memproses big data akan sangat kesulitan.
- Solusi: Pekerja yang tidak familiar dengan high-level programming biasanya lebih familiar dengan bahasa seperti SQL, karena lebih mudah untuk dituliskan. Dan Hive didesain dengan tujuan kemudahan dalam akses struktur tabel, seperti SQL pada umumnya dari big data.





# Kenapa Butuh Pig?

- Problem base, misal pada Yahoo, tidak semua pekerjaanya paham MapReduce, serta java programming yang baris kodenya panjang dan kompleks. Sehingga kebanyakan pekerja tersebut kesulitan.
- Solusi: Maka dalam hal tersebut dibutuhkan proses data yang menggunakan bahasa yang lebih mudah dari Java. Kemudian Peneliti di Yahoo menggunakan Pig untuk pengolahan data yang lebih cepat dan mudah.





# DIGITAL TALENT SCHOLARSHIP 2019

Big Data Analytics



## Apache Spark

Oleh: Imam Cholissodin | [imamcs@ub.ac.id](mailto:imamcs@ub.ac.id), Putra Pandu Adikara, Sufia Adha Putri

Asisten: Guedho, Sukma, Anshori, Aang dan Gusti

**Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya (UB)**

# Pokok Pembahasan

- Review Hadoop
- Tentang Spark & RDD
- Cara menjalankan Kode Spark
- Tugas

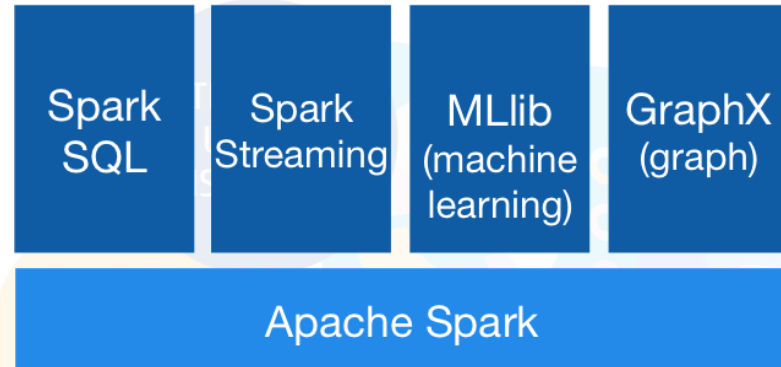




# Review Hadoop

- **Apache Hadoop Framework:**
  - **HDFS**, untuk menyimpan data.
  - **MapReduce**, untuk mengolah data secara **batch**.
  - Pada Perkembangannya ada **YARN** yaitu seperti OS dalam hadoop yang mengatur “**Resource Manager**” dan “**Node Manager**”
- Hadoop dapat ditanam pada (pseudo/virtualisasi atau non-virtualisasi, standalone, maupun full distributed):
  - **Single Node**.
  - **Multi Node**.
- Apache Hadoop sangat **mudah** untuk **dikombinasi** dengan misal Apache Spark (sehingga dapat mengolah data secara **interaktif/ streaming**), maupun lainnya.
- Dalam MapReduce terdapat 2 komponen, yaitu “**Job Tracker**” pada PC Master dan “**Task Tracker**” pada PC Slaves. Jika pada Spark PC Slaves disebut sebagai Workers
- Beberapa **contoh Hadoop Distribution**: Cloudera Distribution Including Apache Hadoop (CDH), Hortonworks Data Platform (HDP), MapR.

# Pengertian Spark



- **Spark** adalah salah satu project Apache, free dan open-source untuk suatu *engine* umum (*general engine*) yang cepat dalam pemrosesan Big Data.
- Spark disebut juga dengan “*Lightning Fast Cluster Computing*”.
- Spark 100x lebih cepat dari Hadoop MapReduce pada memory, dan 10x lebih cepat pada disk.

# Pengertian Spark

- Spark dapat dijalankan di Hadoop, Mesos, standalone, atau di cloud. Dan dapat mengakses beragam sumber data termasuk HDFS, Cassandra, HBase, dan S3.



**Apache Mesos** adalah proyek open-source untuk mengelola cluster komputer.

**Spark Core** adalah mesin komputasi yang bertanggung jawab terhadap proses penjadwalan (*scheduling*), distribusi dan pemantauan (*monitoring*) dari suatu aplikasi yang dijalankan, yang terdiri dari **banyak tugas komputasi** pada banyak mesin pekerja (**multi-node**) dalam komputasional cluster sampai 1000 node dan terus berkembang.

# Pengertian Spark

- **Spark** adalah *engine* berkelompok untuk tujuan umum (***general purpose cluster engine***) yang mendukung konsep sistem terdistribusi dengan *application programming interface* (APIs) dalam **Java**, **Scala**, **Python**, dan **R** serta beberapa library untuk *streaming*, *graph* dan juga *machine learning* (mesin pembelajaran yang merupakan sekumpulan dari banyak algoritma didalamnya).
- Spark menawarkan suatu fungsional dari pemrograman API untuk memanipulasi **Resilient Distributed Datasets** (RDDs).

## Bahasa Pemrograman



# Resilient Distributed Datasets (RDDs)

- **RDDs** merepresentasikan suatu **logical plan** untuk melakukan komputasi suatu dataset.
- **RDDs** mendukung toleransi kesalahan (fault-tolerant), sehingga sistem dapat *me-recover* data yang hilang (*lost*) atau gagal saat diproses menggunakan **lineage graph** RDDs (dengan *me-running* kembali operasi pada *input* data untuk *me-rebuild* kembali partisi yang hilang).
- **RDDs** memiliki 2 tipe operasi:
  - **Transformation**, mengkonstruksi/membangun RDD baru dari satu atau lebih dari yang sebelumnya. Contoh: Map, Reduce, Filter.
  - **Actions**, Menghitung hasil dari suatu komputasi berdasarkan RDD dan memberikan return/kembalian kepada program driver atau simpan ke penyimpanan eksternal.

# 3 APIs Pada Spark

- **RDDs** (low-level: map, reduce, filter, etc). RDDs dapat di-*convert* menjadi DataFrames.
- **DataFrames(df)** (higher-level: SQL, etc), DataFrames hanya seperti Datasets[Row] atau termasuk bagian **ds**.

```

In [1]: import findspark
findspark.init() # call SPARK_HOME

In [2]: import pyspark # run setelah memanggil findspark.init()
from pyspark.sql import SparkSession # panggil SQL pada Spark
# spark = SparkSession.builder.getOrCreate() # aktifkan spark session
spark = SparkSession\
    .builder\
    .appName("Use SQL Spark")\
    .getOrCreate()

In [3]: df = spark.sql('select 'spark' as kolomDummy ')
df.show() # df adalah dataframe, yang merupakan bagian dari API-nya Spark
# API pada Spark ada RDDs (low-level => filter, etc), DataFrames/df (higher-level => SQL, etc),
# DataFrames hanya seperti Datasets[Row], dan Datasets/ds

+-----+
|kolomDummy|
+-----+
|      spark|
+-----+

In [ ]:

```

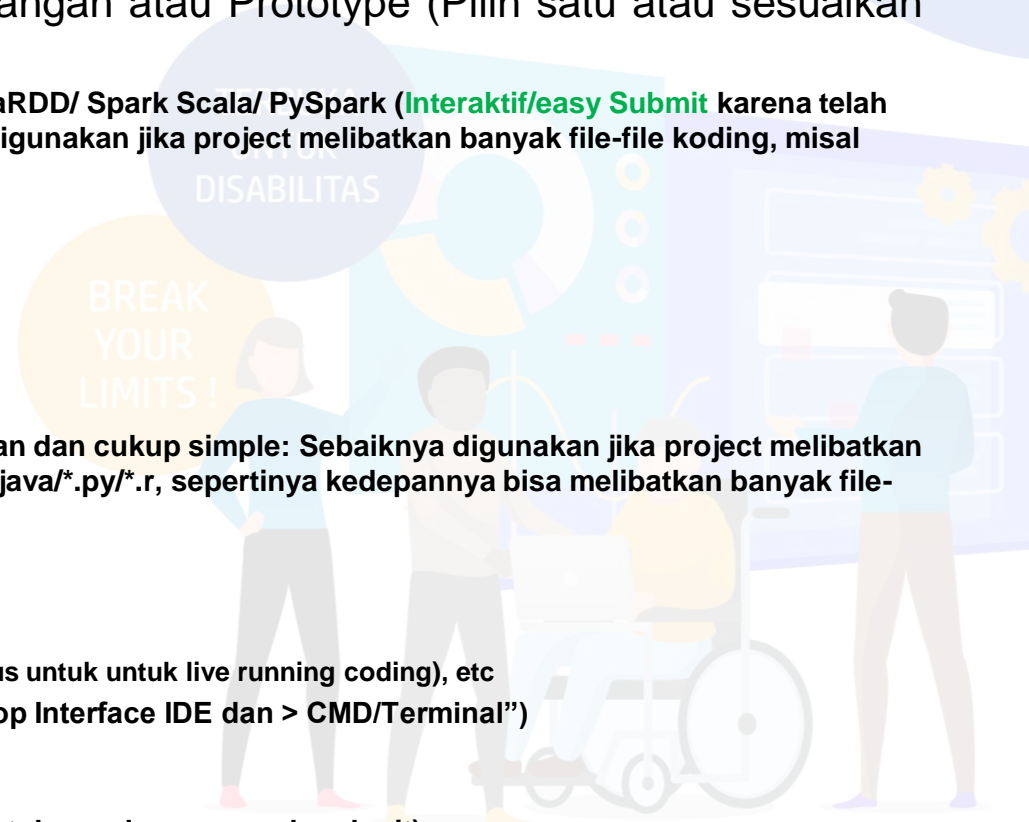
- **Datasets(ds).**





# Cara menjalankan Kode Spark

- Run Kode Program untuk Pengembangan atau Prototype (Pilih satu atau sesuaikan dengan kebutuhan):
  - Desktop Interface/IDE/GUI For Spark JavaRDD/ Spark Scala/ PySpark (**Interaktif/easy Submit** karena telah dkonfigurasi dalam IDE-nya: Sebaiknya digunakan jika project melibatkan banyak file-file koding, misal \*.scala/\*.java/\*.py/\*.r):
    - Eclipse + Spark Standalone (Java EE)
    - Eclipse + Spark + Scala IDE + Maven
    - Eclipse + Spark + Scala IDE + SBT
    - Eclipse + PySpark + PyDev
    - Pycharm + PySpark
    - IntelliJ IDEA + SBT, etc
  - Web Interface (**Interaktif/** mudah digunakan dan cukup simple: Sebaiknya digunakan jika project melibatkan hanya file koding tunggal, misal \*.scala/\*.java/\*.py/\*.r, sepertinya kedepannya bisa melibatkan banyak file-file koding):
    - Jupyter Notebook
    - Apache Zeppelin
    - Hue
    - databricks.com (notebook online, bagus untuk untuk live running coding), etc
  - Editor Interface (versi simple dari “Desktop Interface IDE dan > CMD/Terminal”)
    - VS Code
    - Sublime, Notepad++, etc
  - CMD atau Terminal, etc (**hard Submit**, contoh run dengan spark-submit) (atau disebut sebagai stand-alone file kode program python/scala/java/r/etc.)
- Untuk membuat App Big Data (Product siap pakai untuk End User):
  - Buat GUI Desktop App dari Desktop Interface + (**Hadoop, Spark/etc**)
  - Buat GUI Web atau Hybrid App dari Desktop Interface/etc + (**Hadoop, Spark/etc** + Django/Vue/etc), etc



# Koding Dasar dan Tingkat Lanjut di Spark

TERBUKA  
UNTUK  
DISABILITAS

BREAK  
YOUR  
LIMITS

# Koding Dasar dan Tingkat Lanjut di Spark



- Koding Spark (Dasar dan Tingkat Lanjut), di Python & Scala
  - Map
  - Filter
  - Reduce
  - Lambda
- Langsung Live Koding dari Jupyter Notebook, link kode “<http://bit.ly/2MXHzZ9>”

```
// Ref. Big Data Community dan https://spark.apache.org/docs/latest/ml/bayes-naive.html MK  
// Analisis Big Data Filkom UB (Imam Cholissodin | imams@ub.ac.id)  
  
Latihan Dasar Scala  
  
In [280]: %scala  
//import java.text.SimpleDateFormat  
//import java.util.{Calendar, Date}  
  
// Membuat class dan fungsi  
class Opt {  
  // deklarasi method add secara umum  
  def add(a: Int, b: Int): Int = a + b  
  
  // nama method sama, tetapi dengan men-set tipe return-nya  
  //def add(a: Int, b: Int): Int = a + b  
  
  // men-definisi method body dalam suatu blok dalam kurung kurawal  
  /*def add(a: Int, b: Int): Int = {  
    a + b  
  }  
}
```

```
Koding Map Spark  
  
In [329]: %python  
def add1(x):  
    return x*5  
  
raw_data = sc.parallelize([1,2,3])  
rdd = raw_data.map(lambda x: add1(x))  
#print(rdd.take(3))  
print(rdd.collect())  
  
[6, 7, 8]  
  
In [136]: %scala  
def add1(x: Double) = x*5  
  
//val rdd = sc.parallelize(1 to 3).map(i => add1(i)).count()  
val rdd = sc.parallelize(1 to 3).map(i => add1(i)).count()  
//val rdd = sc.parallelize(1 to 3).map(i => add1(i)).count()  
print(rdd.count()+"\n")  
print(rdd.sum()+"\n")  
print(rdd.mean())  
rdd.collect()  
  
4  
33.0  
8.25  
Out[136]: add1: (x: Double)Double  
rdd: org.apache.spark.rdd.RDD  
res62: Array[Double] = Array(5, 10, 15)
```

TERBUKA  
UNTUK  
DISABILITAS

BREAK  
YOUR  
LIMIT

# Koding Dasar dan Tingkat Lanjut di Spark

## ■ Koding Spark + Python (Tambahan 1 of 4)

### Contoh Koding PySpark (Tambahan)

```
In [48]: %%python
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Latihan Tambahan").getOrCreate()
sc = spark.sparkContext

data = sc.parallelize([('Mhs 1', 95), ('Mhs 2', 70), ('Mhs 3', 85), ('Mhs 4', 100)])

# menampilkan data awal, yaitu mahasiswa dan nilai uas-nya
print("Menampilkan data awal, yaitu mahasiswa dan nilai uas-nya:")
print(data.collect())

print("\n")

print("Menampilkan nama mahasiswa dan nilai uas-nya:")
hasil1=data.reduceByKey(lambda x,y: x+y).collect()
print(hasil1)

print("\n")

print("Menampilkan hanya nilai uas semua mahasiswa-nya:")
hasil2=data.map(lambda t: t[1]).collect()
print(hasil2)

print("\n")

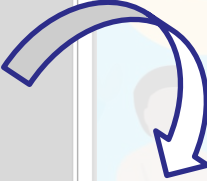
print("Menampilkan total nilai uas semua mahasiswa-nya:")
hasil3=data.map(lambda t: (1,int(t[1]))).reduceByKey(lambda x,y: x+y).collect()
print(hasil3)

print("\n")

print("Menampilkan nilai minimum dari semua nilai uas mahasiswa:")
hasil4=data.map(lambda t: (1,int(t[1]))).reduceByKey(lambda x,y: min(x,y)).collect()
print(hasil4)

print("\n")

print("Menampilkan nilai maksimum dari semua nilai uas mahasiswa:")
hasil5=data.map(lambda t: (1,int(t[1]))).reduceByKey(lambda x,y: max(x,y)).collect()
print(hasil5)
```



```
Menampilkan data awal, yaitu mahasiswa dan nilai uas-nya:
[('Mhs 1', 95), ('Mhs 2', 70), ('Mhs 3', 85), ('Mhs 4', 100)]

Menampilkan nama mahasiswa dan nilai uas-nya:
[('Mhs 2', 70), ('Mhs 3', 85), ('Mhs 4', 100), ('Mhs 1', 95)]

Menampilkan hanya nilai uas semua mahasiswa-nya:
[95, 70, 85, 100]

Menampilkan total nilai uas semua mahasiswa-nya:
[(1, 350)]

Menampilkan nilai minimum dari semua nilai uas mahasiswa:
[(1, 70)]

Menampilkan nilai maksimum dari semua nilai uas mahasiswa:
[(1, 100)]
```

# Koding Dasar dan Tingkat Lanjut di Spark

## ■ Koding Spark + Python (Tambahan 2 of 4)

```
print("\n")

print("Menampilkan banyak mahasiswa:")
hasil6=data.map(lambda t: (1,1)).reduceByKey(lambda x,y: x+y).collect()
print(hasil6)

print("\n")

print("Menampilkan nilai rata-rata dari semua nilai uas mahasiswa:")
import numpy as np
hasil7=np.array(hasil3) / np.array(hasil6)
print(hasil7)
```

Menampilkan banyak mahasiswa:  
[(1, 4)]

Menampilkan nilai rata-rata dari semua nilai uas mahasiswa:  
[[ 1. 87.5]]

# Koding Dasar dan Tingkat Lanjut di Spark

## ■ Koding Spark + Python (Tambahan 3 of 4)

```
In [88]: %%python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
#from pyspark.sql import functions as F    dimana min => F.min, max => F.max
from pyspark.sql.functions import min
from pyspark.sql.functions import max

spark = SparkSession.builder.appName("Latihan Tambahan").getOrCreate()
sc = spark.sparkContext

rdd = sc.parallelize [("Mhs 1", 95), ("Mhs 2", 70), ("Mhs 3", 85), ("Mhs 4", 100)]

# cara convert rdd ke df
rdd2df=rdd.toDF(["Nama","Nilai UAS"])

print("Menampilkan data awal, yaitu mahasiswa dan nilai uas-nya dari hasil rdd2df:")
rdd2df.show()

df = spark.createDataFrame([("Mhs 1", 95), ("Mhs 2", 70), ("Mhs 3", 85), ("Mhs 4", 100)], ["Nama", "Nilai UAS"])

# menampilkan data awal, yaitu mahasiswa dan nilai uas-nya
print("Menampilkan data awal, yaitu mahasiswa dan nilai uas-nya:")
df.show()

print("\n")

print("Menampilkan nilai uas yang paling minimal cara ke-1:")
minNilaiUAS = df \
    .groupBy() \
    .min('Nilai UAS') \
    .select(col("min(Nilai UAS)").alias("min Nilai UAS"))

minNilaiUAS.show()

print("\n")

print("Menampilkan nilai uas yang paling minimal cara ke-2:")
hasil=df.groupBy().agg(min(col('Nilai UAS')))
hasil.show()
```

Menampilkan data awal, yaitu mahasiswa dan nilai uas-nya dari hasil rdd2df:

+-----+	
Nama	Nilai UAS
+-----+	
Mhs 1	95
Mhs 2	70
Mhs 3	85
Mhs 4	100
+-----+	

Menampilkan data awal, yaitu mahasiswa dan nilai uas-nya:

+-----+	
Nama	Nilai UAS
+-----+	
Mhs 1	95
Mhs 2	70
Mhs 3	85
Mhs 4	100
+-----+	

Menampilkan nilai uas yang paling minimal cara ke-1:

+-----+	
min Nilai UAS	
+-----+	
	70
+-----+	

Menampilkan nilai uas yang paling minimal cara ke-2:

+-----+	
min(Nilai UAS)	
+-----+	
	70
+-----+	



# Koding Dasar dan Tingkat Lanjut di Spark

## ■ Koding Spark + Python (Tambahan 4 of 4)

```
In [95]: %%python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.sql.functions import mean
from pyspark.sql.functions import sum

spark = SparkSession.builder.appName("Latihan Tambahan").getOrCreate()
#sc = spark.sparkContext

# data nilai uas mahasiswa
df = spark.createDataFrame(
[("Mhs 1", "P", 95), ("Mhs 2", "L", 70),
("Mhs 3", "P", 85), ("Mhs 4", "L", 100)],
["Nama", "Jenis Kelamin", "Nilai UAS"])

print("Data nilai uas mahasiswa:")
df.show()

print("\n")

print("Menampilkan rata-rata nilai uas berdasarkan JK:")
df.groupBy('Jenis Kelamin').agg(mean(col('Nilai UAS'))).show()

# data iuran mahasiswa
df2 = spark.createDataFrame(
[("Mhs 1", "P", 1000), ("Mhs 2", "L", 500),
("Mhs 3", "P", 10000), ("Mhs 4", "L", 1000)],
["Nama", "Jenis Kelamin", "Iuran (Rp)"])

print("Data iuran mahasiswa:")
df2.show()

print("\n")

print("Menampilkan jumlah iuran berdasarkan JK:")
df2.groupBy("Jenis Kelamin").agg(sum(col("Iuran (Rp)"))).show()
```

Data nilai uas mahasiswa:

Nama	Jenis Kelamin	Nilai UAS
Mhs 1	P	95
Mhs 2	L	70
Mhs 3	P	85
Mhs 4	L	100

Menampilkan rata-rata nilai uas berdasarkan JK:

Jenis Kelamin	avg(Nilai UAS)
L	85.0
P	90.0

Data iuran mahasiswa:

Nama	Jenis Kelamin	Iuran (Rp)
Mhs 1	P	1000
Mhs 2	L	500
Mhs 3	P	10000
Mhs 4	L	1000

Menampilkan jumlah iuran berdasarkan JK:

Jenis Kelamin	sum(Iuran (Rp))
L	1500
P	11000



# Latihan Koding Dasar dan Tingkat Lanjut di Spark



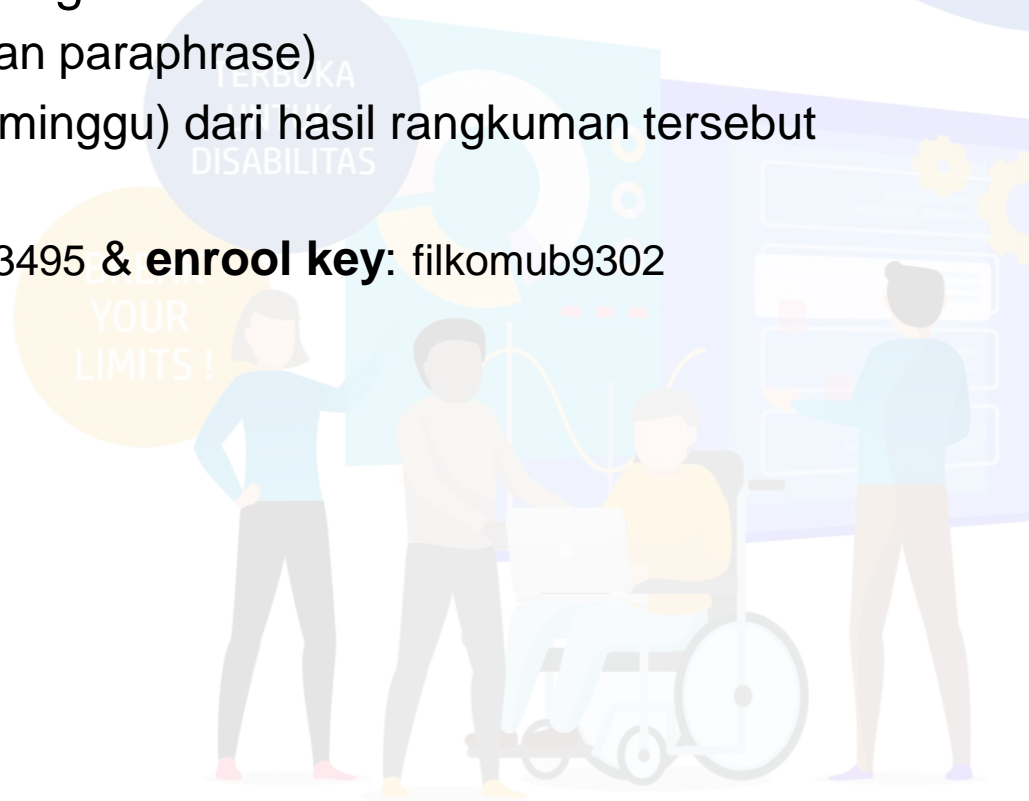
- Diberikan himpunan data berikut:  $a = [(1,2), (10,4), (5,6)]$ 
  - a. Buat suatu koding Python dan Scala dengan suatu ekspresi untuk mendapatkan hanya element kedua dari tiap tuple dari a
  - b. Buat suatu koding Python dan Scala untuk menghitung hasil penjumlahan elemen kedua
  - c. Buat suatu koding Python dan Scala untuk menghitung hasil penjumlahan elemen pertama yang bernilai ganjil
  - d. Buat RDD dari himpunan data a, lalu convert menjadi DataFrame
- Diberikan data berikut:

```
df = spark.createDataFrame(  
    [("Mhs 1", "4", 1000), ("Mhs 2", "1", 500),  
    ("Mhs 3", "6", 10000), ("Mhs 4", "5", 1000), ("Mhs 5", "2", 2000)],  
    ["Nama", "Semester", "Tabungan (x Rp 1000)"])
```

  - a. Buat suatu koding PySpark untuk mencari Mhs yang jumlah tabungannya paling Besar
  - b. Buat suatu koding PySpark untuk mencari banyaknya Mhs yang jumlah tabungannya diatas 1 juta
  - c. Buat suatu koding PySpark untuk mencari semua Mhs diatas semester 3, lalu totalkan semua jumlah tabungannya

# Tugas Individu

1. Buatlah rangkuman materi dengan cara berikut:
  - Rangkum yang pokok (bukan paraphrase)
  - Cek plagiasi di turnitin (tiap minggu) dari hasil rangkuman tersebut
    - > Register ke turnitin
    - > Masukkan **id class**: 21563495 & **enroll key**: filkomub9302





# DIGITAL TALENT SCHOLARSHIP 2019

Big Data Analytics



## Terimakasih

Oleh: Imam Cholissodin | [imamcs@ub.ac.id](mailto:imamcs@ub.ac.id), Putra Pandu Adikara, Sufia Adha Putri

Asisten: Guedho, Sukma, Anshori, Aang dan Gusti

**Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya (UB)**