Additional Spark Material

Rosa Filgueira

The structured spectrum

Strcutured	Semi-structured	Unstructured
• Relational Databases • Parquet • Formatted Messages	• HTML • XML • JSON	• Plain text • Generic media

Parsing JSON data

```
import json

#parse multi-line json file
parsed = sc.textFile("data.json").map(lambda x:
json.loads(x))

#parse directory of json documents
parsed = sc.wholeTextFiles("data/").map(lambda x:
json.loads(x))

#write out json data
data.map(lambda x:json.dumps(x)).saveAsTextFile(output)
```

Parsing CSV data

```
import csv
from StringIO import StringIO

# Read from CSV
def load_csv(contents):
    return csv.reader(StringIO(contents[1]))

data = sc.wholetextFile("data/").flatMap(load_csv)

# Write to CSV
def write_csv(recods):
   output = StringIO()
   writer = csv.writer()
   for record in records:
        writer.writerow(record)
   return [output.get_value()]

data.mapPartitions(wrute_csv).saveAsTextFile("output/")
```

Parsing Structured Objects

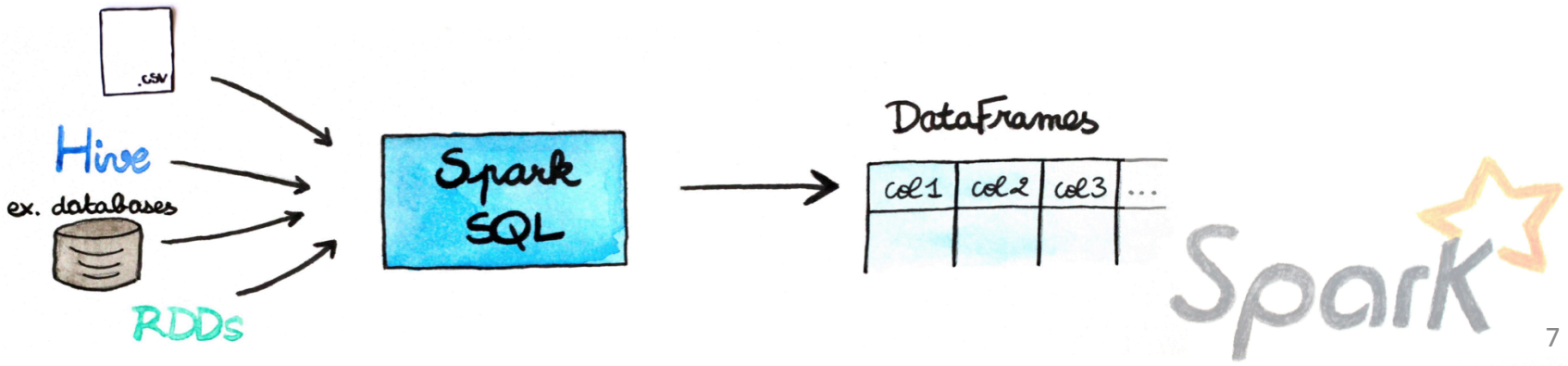
```
import csv

from datetime import datetime
from StringIO import StringIO
from collections import namedtuple

DATE_FMT = "%Y-%m-%d %H:%M:%S" # 2013-09-16 12:23:33
Customer = namedtuple('Customer', ('id', 'name', 'registered'))

def parse(row):
   row[0] = int(row[0]) # Parse ID to an integer
   row[4] = datetime.strptime(row[4], DATE_FMT)
   return Customer(*row)

def split(line):
    reader = csv.reader(StringIO(line))
    return reader.next()

customers = sc.textFile("customers.csv").map(split).map(parse)
```

About Spark SQL

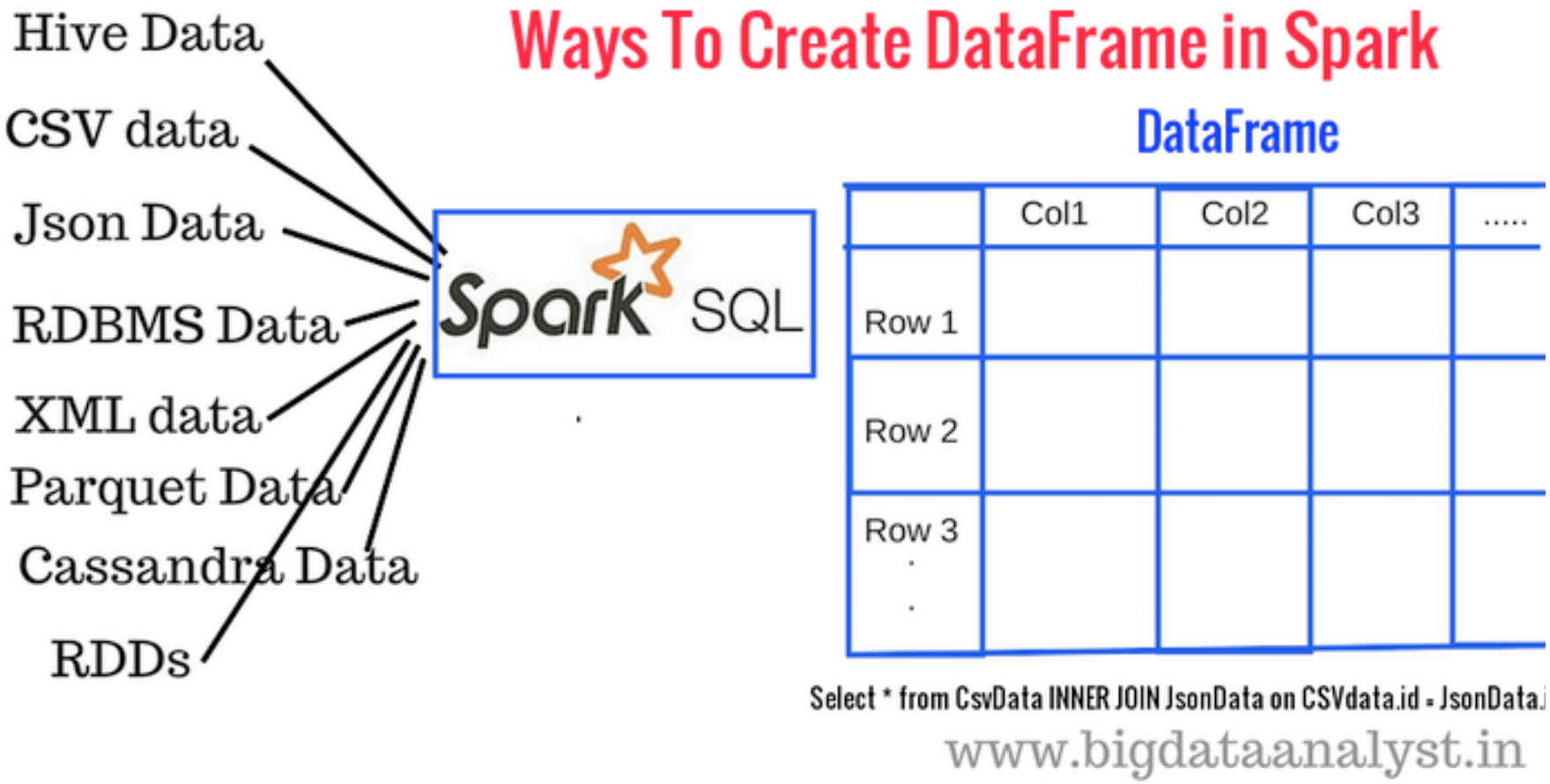
Spark SQL is a component on top of **Spark Core** that facilitates processing of structured and semi-structured data and the integration of several data formats as source (Hive, Parquet, JSON).

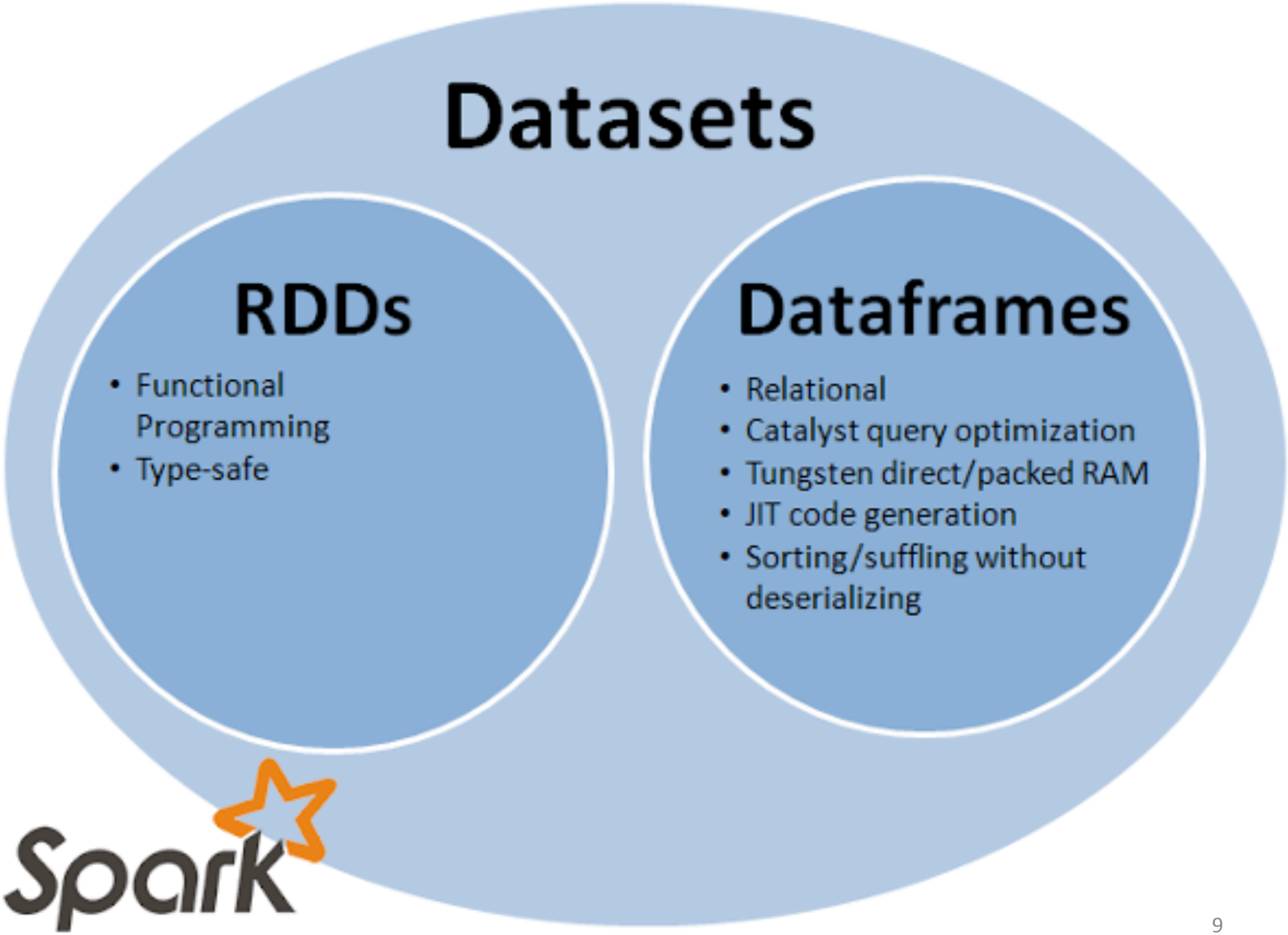
DataFrames

- DataFrame is an immutable distributed collection of data.
- Unlike an RDD, data is organized into named columns, like a table in a relational database or a data frame in R/Python
- Distributed collection of data grouped into named columns:
  - DataFrames = RDD + Schema
- Designed to make large data sets processing even easier.
- Allows developers to impose a structure onto a distributed collection of data, allowing higher-level abstraction;



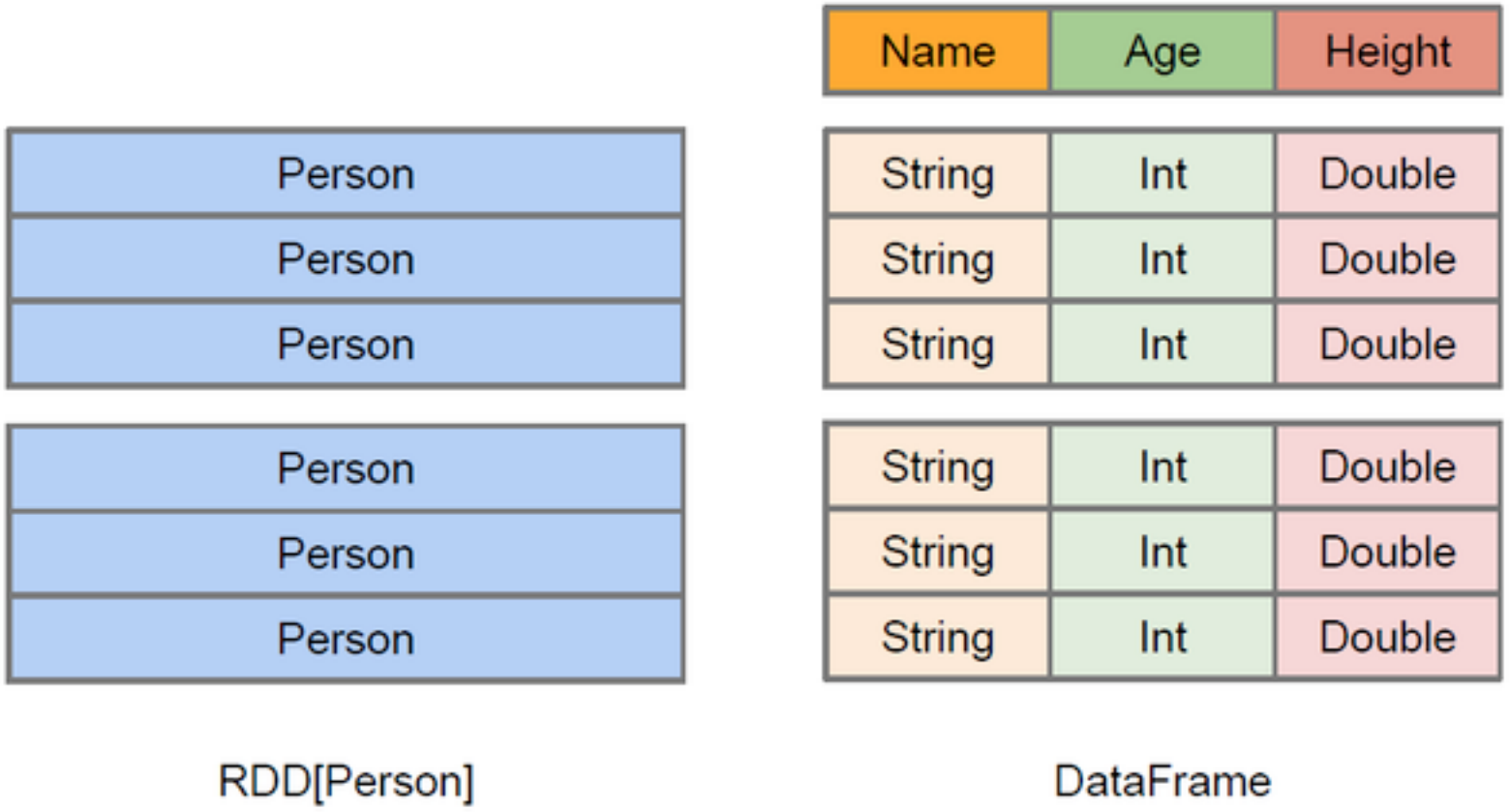
DataFrames


Hive Data
CSV data
Json Data
RDBMS Data
XML data
Parquet Data
Cassandra Data
RDDs
Spark SQL
Ways To Create DataFrame in Spark
DataFrame
Col1
Col2
Col3
.....
Row 1
Row 2
Row 3
Select * from CsvData INNER JOIN JsonData on CSVdata.id = JsonData.
www.bigdataanalyst.in


Datasets
RDDs
• Functional Programming
• Type-safe
Dataframes
• Relational
• Catalyst query optimization
• Tungsten direct/packed RAM
• JIT code generation
• Sorting/suffling without deserializing
Spark

RDD vs DataFrames

DataFrames are composed of Row objects,
along with a schema that describes the data types of each column in the row.



Write Less Code: Compute an Average

DataFrames

dept	age	name
Bio	48	H Smith
CS	54	A Turing
Bio	43	B Jones
Chem	61	M Kennedy

Data grouped into named columns

RDD API

```
pdata.map(lambda x: (x.dept, [x.age, 1])) \
  .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
  .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
  .collect()
```

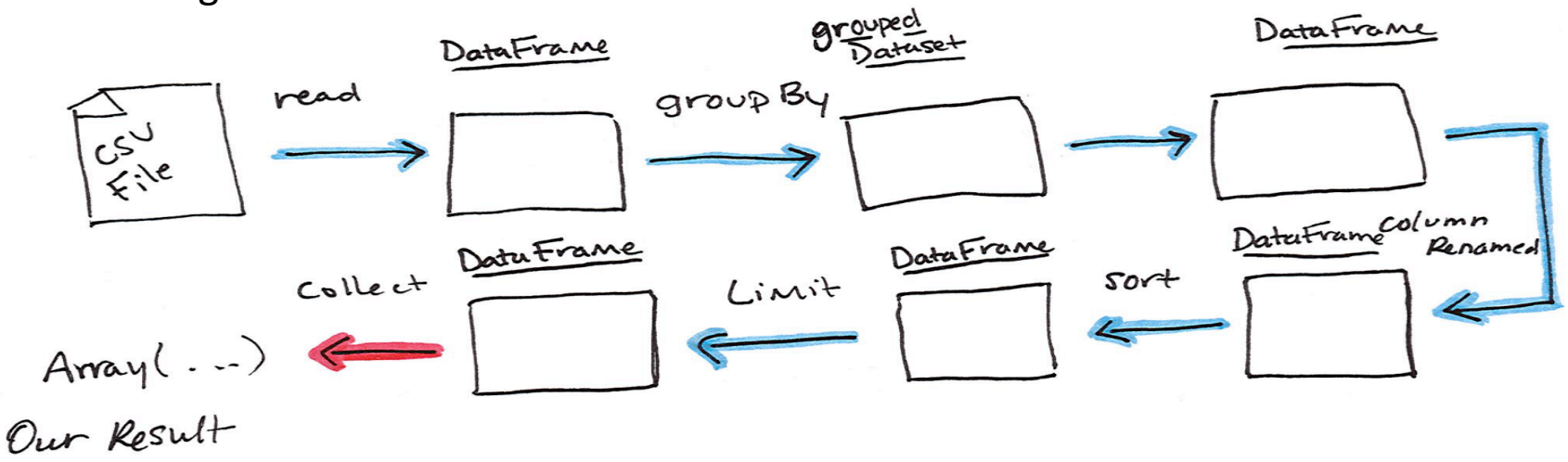
DataFrame API

```
data.groupBy("dept").avg("age")
```

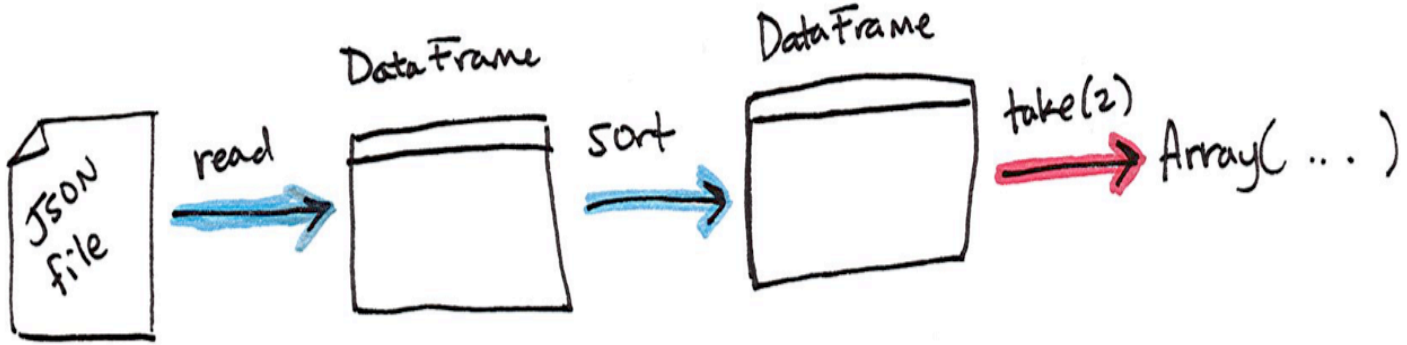
databricks

Dataframes

Working with CSV files



Working with Json files



https://www.safaribooksonline.com/library/view/spark-the-definitive/9781491912201/ch04.html

Dataframes and RDDs

- Create from RDD of tuples

```
>> rdd = sc.parallelize([(”a”, 1), (”b”, 2), (”c”, 3)])
>> df = sqlContext.createDataFrame(rdd,[”name”, ”id”])
>> df.show()
```

```
+----+--+
|name|id|
+----+--+
|   a| 1|
|   b| 2|
|   c| 3|
+----+--+
```

Dataframes and RDDs

- Create from RDD of Rows

```
>> from pyspark.sql  import Row
>> ExampleRow = Row(”name”, ”id”)
>> rdd = sc.parallelize([ ExampleRow(”a”, 1),
   ExampleRow(”b”, 2), ExampleRow(”c”, 3) ])
>> df = sqlContext.createDataFrame(rdd)
>> df.show()
```

```
+----+--+
|name|id|
+----+--+
|   a| 1|
|   b| 2|
|   c| 3|
+----+--+
```

```
>> df.printSchema()
```

```
root
 |-- name: string (nullable = false)
 |-- id: integer (nullable = false)
```

More information at: http://semantica.cs.lth.se/pyspark/slides/Lecture%202.pdf

Dataframe – read/write formats

- sqlContext.read.[format]

```
>> sqlContext.read.parquet(path)
>> sqlContext.read.json(path, [schema])
>> sqlContext.read.jdbc(url, table)
>> sqlContext.read.load(path, [format])
```

- sqlContext.write.[format]

```
>> sqlContext.write.parquet(path)
>> sqlContext.write.json(path, [mode])
>> sqlContext.write.jdbc(url, table, [mode])
>> sqlContext.write.save(path, [format], [mode])
```

More information at:
https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrameWriter

Examples of Spark SQL

```
from pyspark import SparkContext
from pyspark.sql import SQLContext

sc = SparkContext('local', 'Spark SQL')
sqlc = SQLContext(sc)
#We can read aJSON file and create a DataFrame (Spark SQL json reader)

players = sqlc.read.json(get(1)) # Print the schema in a tree format
players.printSchema()

" Select only the "FullName" column players.select("FullName").show(4)
```

```
+----------------------+
| FullName             |
 +----------------------+
| Ãngel Bossio       |
| Juan Botasso       |
| Roberto Cherro   |
| Alberto Chividini|
+----------------------+
```

Examples of Spark SQL

```
#Then we can create a view of our DataFrame. The lifetime of this
temporary table is tied to the SparkSession that was used to create
this DataFrame.


players.registerTempTable("players")


#We can then query our view; for instance to get the names of all
the Teams


sqlc.sql("select distinct Team from players").show(5)
+----------+
| Team   |
 +----------+
|England |
|Paraguay|
| POL    |
 | Russia  |
 | BRA    |
+-----------+
```

Pandas DataFrame

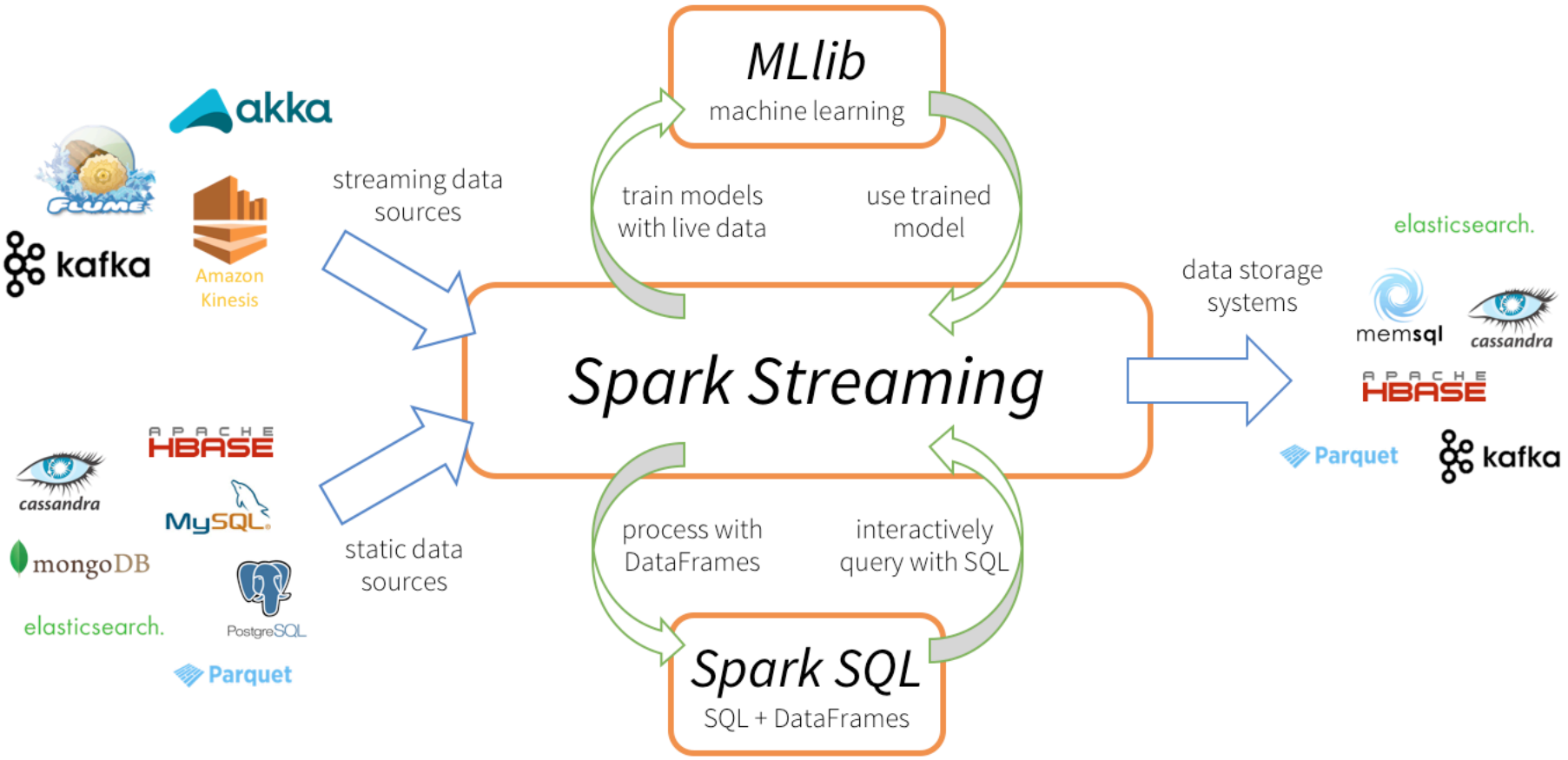
- When in PySpark, there is also an easy option to convert Spark DataFrame to Pandas dataframe.

```
# Convert Spark DataFrame to Pandas
pandas_df = spark_df.toPandas()

# Create a Spark DataFrame from Pandas
spark_df = context.createDataFrame(pandas_df)
```

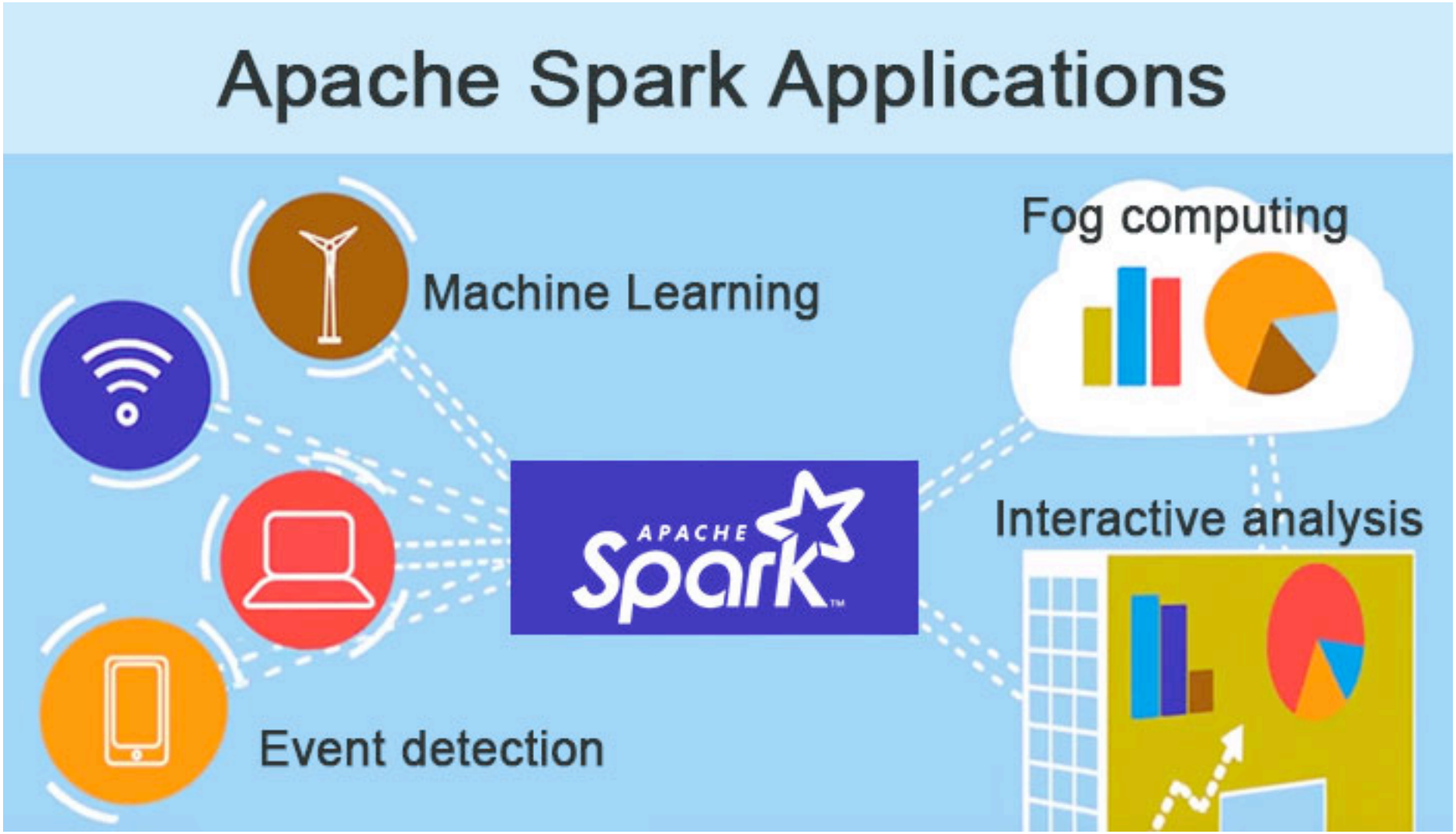
- One powerful and easy way to visualize data is

```
dataframe.toPandas().plot()
```

Complex Applications - Working with all the Spark libraries



Type of applications



Uses Cases

NETFLIX

Uses Spark Streaming to provide the best-in-class movie streaming and recommendation tool to its users.

Uses Spark to collect TBs of raw and unstructured data every day from its users to convert it into structured data. This makes it ready for further complex analytics.

Feeds real-time data into Spark via Spark Streaming to get instant insights on how users are engaging with Pins globally. This makes Pinterest's recommendations (i.e. to show Pins) to be accurate.

Uses Cases

Spark Use Cases

edureka!

Twitter Sentiment Analysis With Spark

Trending Topics can be used to create campaigns and attract larger audience

Sentiment helps in crisis management, service adjusting and target marketing

NYSE ICE

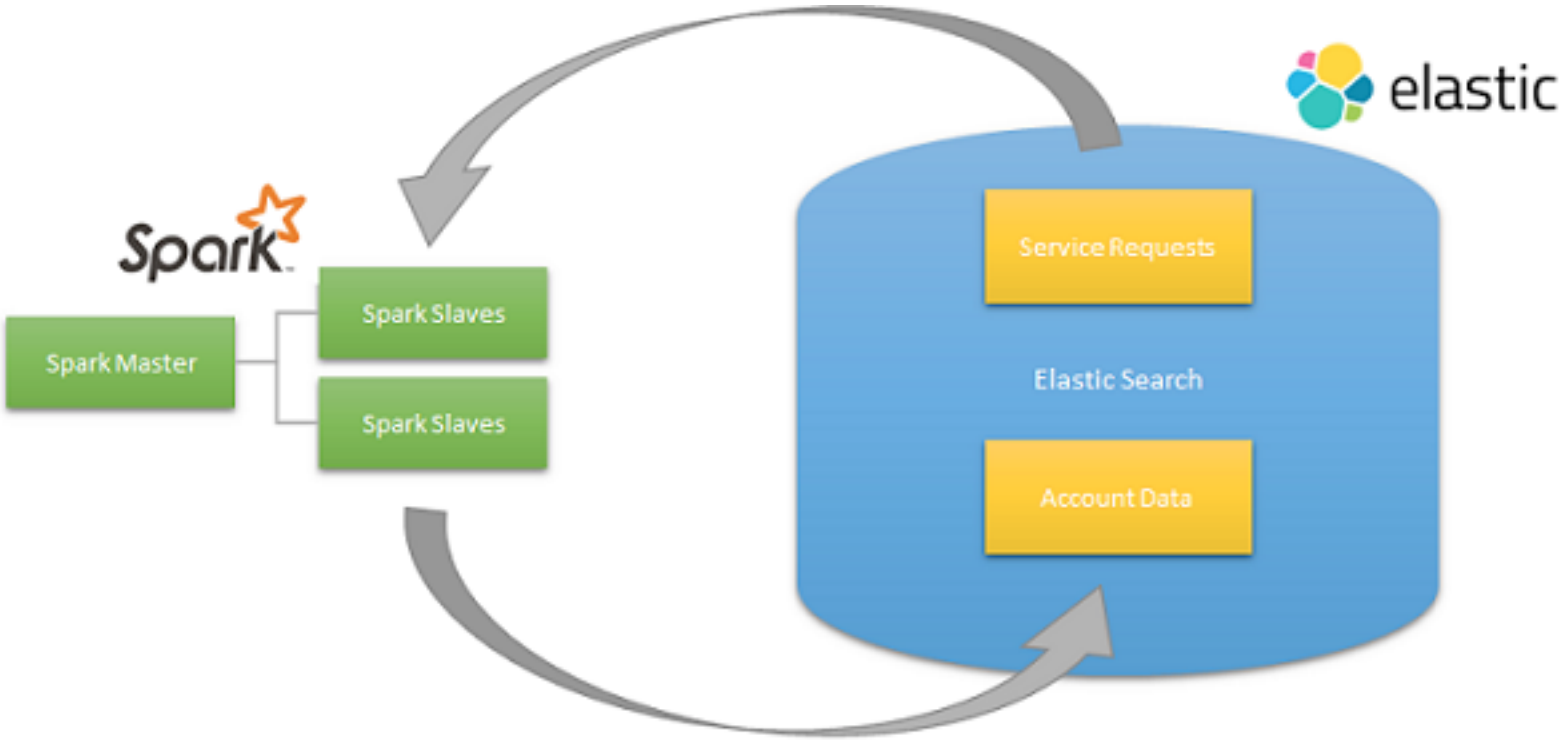
NYSE: Real Time Analysis of Stock Market Data

Banking: Credit Card Fraud Detection

Genomic Sequencing

EDUREKA SPARK CERTIFICATION TRAINING www.edureka.co/apache-spark-scala-training

Spark + Elastic Search



Apache + Elastic Search