WIZELINE ACADEMY

Grow your career:
Free courses in Artificial Intelligence, Software Development, User Experience and More

WIFI: WizelineAcademy
Password: academyGDL
Slack Channel: #

@WizelineAcademy

academy.wizeline.com

/WizelineAcademy

Get notified about courses:
tinyurl.com/WL-academy

Week 4 -
Session 2/2

Spark SQL & Beyond

Spark Details

WIZELINE

Agenda

- **Spark SQL**
  - Data Schemas
  - SQL Support
- **Advanced Features**
  - User-Defined Functions (UDFs)
  - Window Functions
- **Beyond this Certification**
  - Other Spark Components
  - Technologies Around Spark

Spark SQL

SQL

Structured Query Language

[...] domain-specific **language** used in programming and designed for **managing data** held in a **relational database** management system.

Wikipedia

Do You Know SQL?

Supported syntax

```
SELECT [DISTINCT] [column names]|[wildcard]
FROM [keyspace name.]table name
[JOIN clause table name ON join condition]
[WHERE condition]
[GROUP BY column name]
[HAVING conditions]
[ORDER BY column names [ASC | DSC]]
```

Spark SQL and Spark DataFrames

Side by side similarities

Spark SQL	DataFrames
SELECT * FROM orders **LIMIT** 20	orders.**limit**(20)
SELECT **DISTINCT**(product_id) FROM orders	orders.select("product_id") .**distinct**
SELECT * FROM orders **WHERE** month(timestamp) == 2	orders.**where**(month($"timestamp") === 2)
SELECT client_id, count(*) FROM orders **GROUP BY** client_id	orders.**groupBy**(“client_id”) .count
SELECT * FROM orders **JOIN** clients ON orders.client_id == clients.id	orders.**join**(clients, orders("client_id") === clients("id"))

Spark SQL

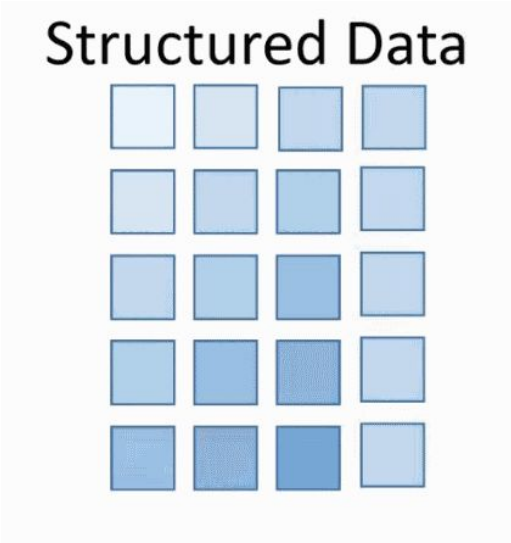
Not just parsing a string!

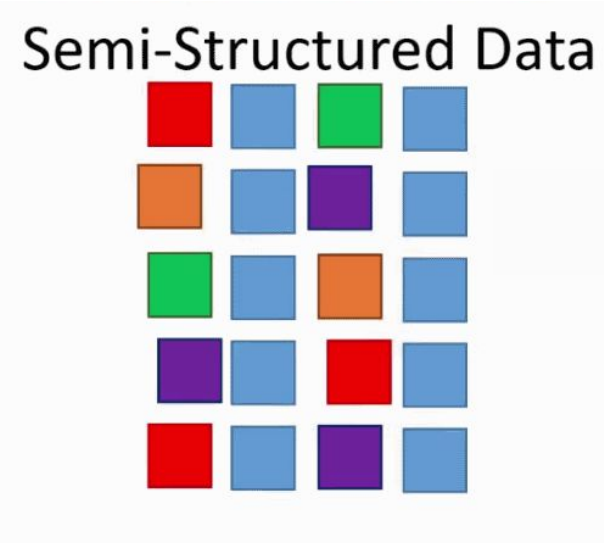
It uses Catalyst too!

- Actually... Catalyst was born for Spark SQL.
  - DataFrames were shortly adapted.
  - Datasets came later on.

Spark SQL Works On...

Anything you can somehow describe





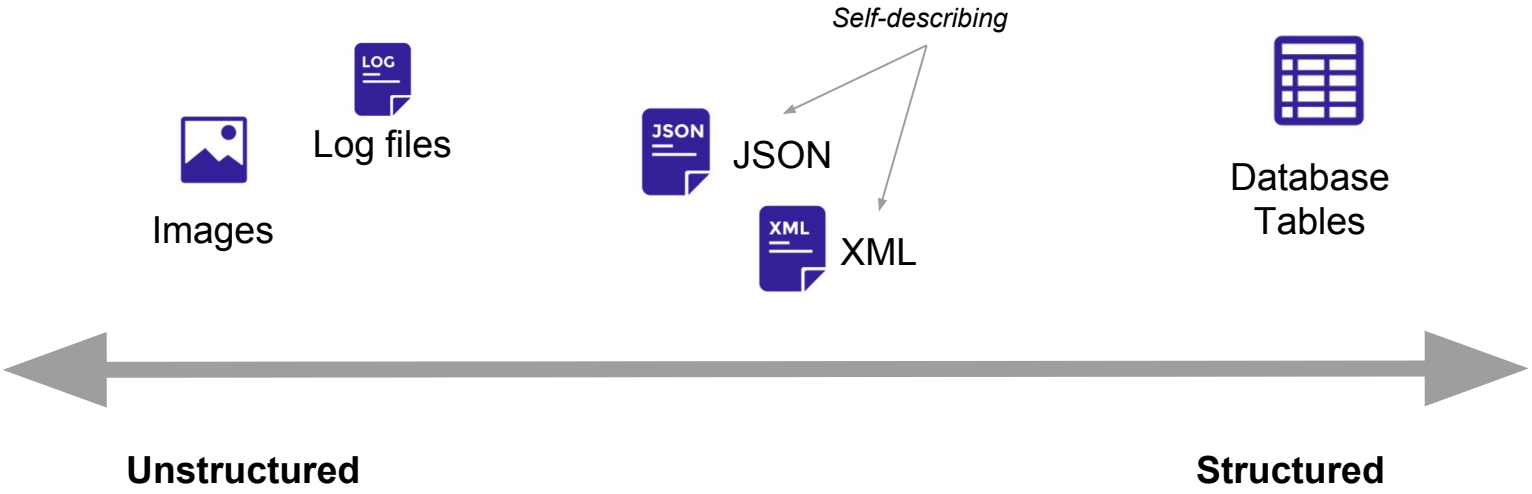
BUT... **Data Schema** is a MUST

Data Schema

[...] Database **structure** described in a **formal language**. The term "schema" refers to the organization of data as a **blueprint** of how the database is constructed [...]

Wikipedia

Friendly Reminder



Data Schemas

Metadata for data structures

- Model that shows the possible arrangement of a data source.
  - **Structured** ⇒ Columns and relationships
  - **Semi-structured** ⇒ Tree-like / key-value pairs. It can grow easily.

Data Schema Example

For structured data

Example

year	week_num	prod_cat	total_qty_top5	total_spent_top5
2018	35	d5	17,403	$235,768.21
2018	34	h5	12	$5,127.00
2018	33	zp	729	$5,689,141.00
2018	32	77	6,123	$781,902.00

Schema

name	type	notes
year	int	Year of data point
week_num	int	Week number
prod_cat	string	Product category
total_qty_top5	long	Total items purchased from top 5 items
total_spent_top5	double	Total money spent in top 5 items

Data Schema Example

For structured data

Example

year	week_num	prod_cat	total_qty_top5	total_spent_top5
2018	35	d5	17,403	$235,768.21
2018	34	h5	12	$5,127.00
2018	33	zp	729	$5,689,141.00
2018	32	77	6,123	$781,902.00

Schema

```
val schema = StructType(List(
    StructField("year", IntegerType, false),
    StructField("week_num", IntegerType, false),
    StructField("prod_cat", StringType, false),
    StructField("total_qty_top5", LongType, false),
    StructField("total_spent_top5", DoubleType, false)))
```

Data Schema Example

For semi-structured data

```json
{
  "firstName": "Nadine",
  "lastName": "Booth",
  "isAlive": false,
  "age": 80,
  "address": {
    "streetAddress": "Woodbine Street",
    "city": "Gibsonia",
    "state": "Georgia",
    "postalCode": 7139
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "(897) 480-3587"
    }
  ]
}
```

Example

```
val schema = StructType(List(
      StructField("firstName", StringType, false),
      StructField("lastName", StringType, false),
      StructField("isAlive", BooleanType, false),
      StructField("age", LongType, false),
      StructField("address",StructType(List(
            StructField("streetAddress", StringType, false),
            StructField("city", StringType, false),
            StructField("state", StringType, false),
            StructField("postalCode", LongType, false))),
        false),
      StructField("phoneNumbers", ArrayType(StructType(List(
          StructField("type", StringType, false),
          StructField("number", StringType, false)))),
        false)))
```

Spark Schema

Characteristics of a Data Schema

Expected fields

- All fields must have a **name.**
- All fields must have a defined **type.**
- (Optional) Fields can be **nullable** by design (may not exist).
- (Optional) Fields might contain **metadata** (`Map[String, Any]`).

How to SQL in Spark?

Using Dataset and Dataframes as tables

1) Firstly, Datasets and DataFrames must be registered as a **TempView.**

2) Then, use SQL.

```
val orders = spark.read.json("gs://de-training-input/alimazon/50000/client-orders/")

// register DataFrame/Dataset to be used by SQL
orders.createTempView("orders")
orders.createOrReplaceTempView("orders")
orders.createGlobalTempView("orders")
orders.createOrReplaceGlobalTempView("orders")

// Remove from catalog once they are not used anymore
spark.catalog.dropTempView("orders")
spark.catalog.dropGlobalTempView("orders")
```

Q&A

Spark SQL Recap

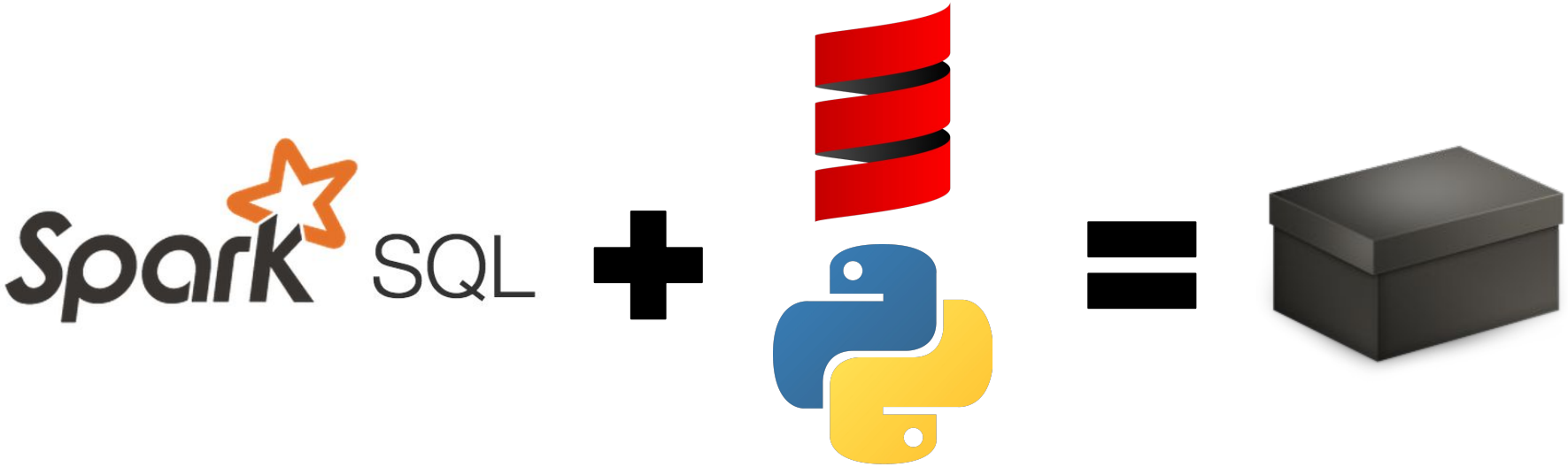
You should be able to answer

- What operations does Spark SQL support?
- What are data schemas?
- How are data schemas implemented in Spark?
- How to register Spark DataFrames or Datasets, so that you can use SQL on them?

Advanced Features

Spark UDFs

Extending functionality through code



This is a UDF

Simple, yet elegant declaration

Used in SQL statement

```
val incDefinition = (x:Long) => x + 1
spark.udf.register("inc", incDefinition)
sql("SELECT id, inc(quantity) FROM orders").show
```

Used in Scala code

```
import org.apache.spark.sql.functions.udf
val incDefinition = (x:Long) => x + 1
val incUDF = udf(incDefinition)
orders.select($"id", incUDF($"quantity")).show
```

Spark UDF: Costs

It is not a go-to solution for everything

- As a **black box**, Catalyst can’t optimize them much.
- **PySpark UDFs** require **data serialization** (CPU overhead):
  - Prefer Scala UDFs in PySpark code.

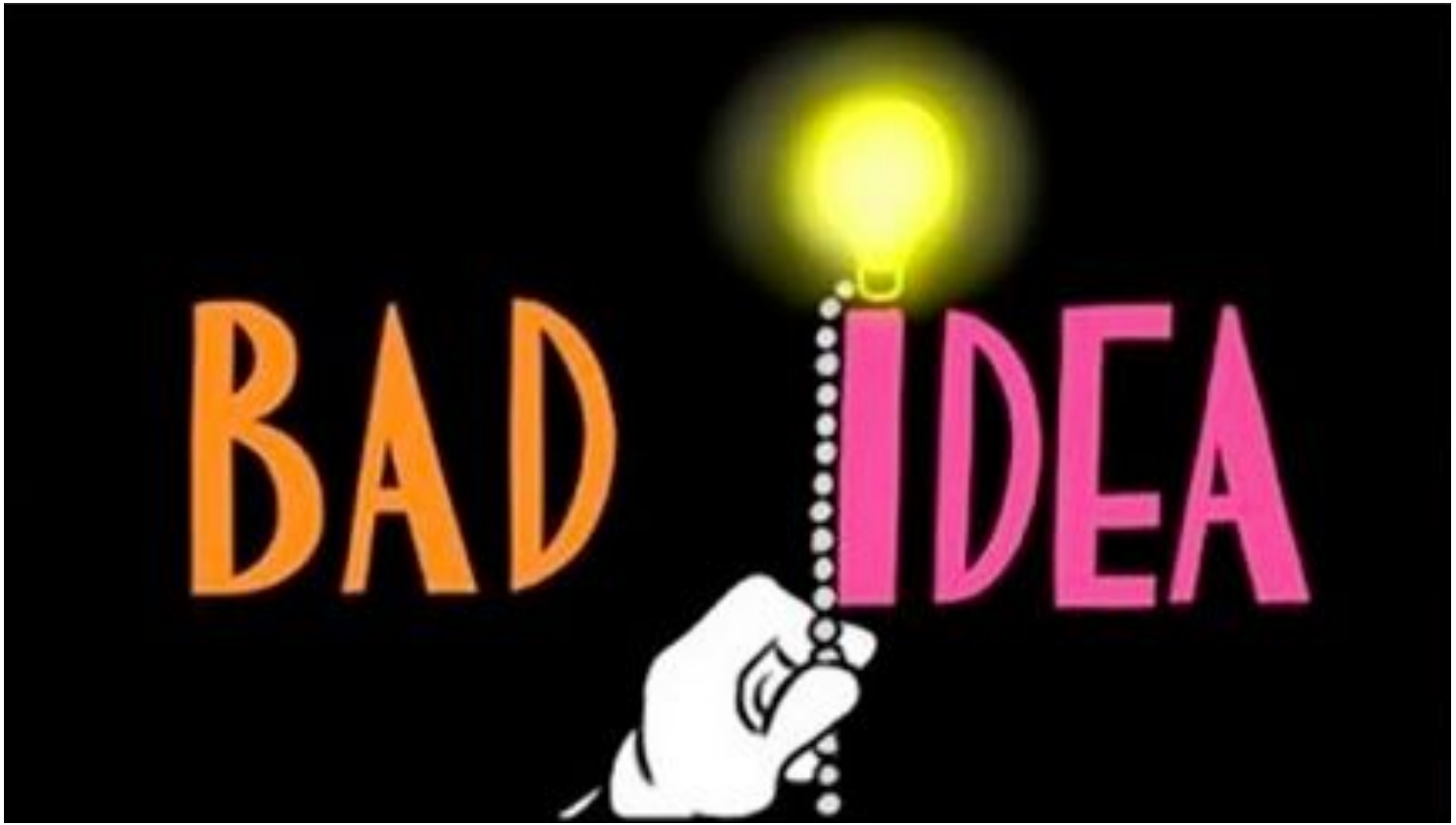
Spark UDFs - Good Ideas

Spark UDFs - Good Ideas

- Call external web services or APIs.
- Embedded cache system from external data source.
- Read mutable configuration to change behavior.

In essence, supplementing Spark capabilities...

Spark UDFs - Bad Ideas



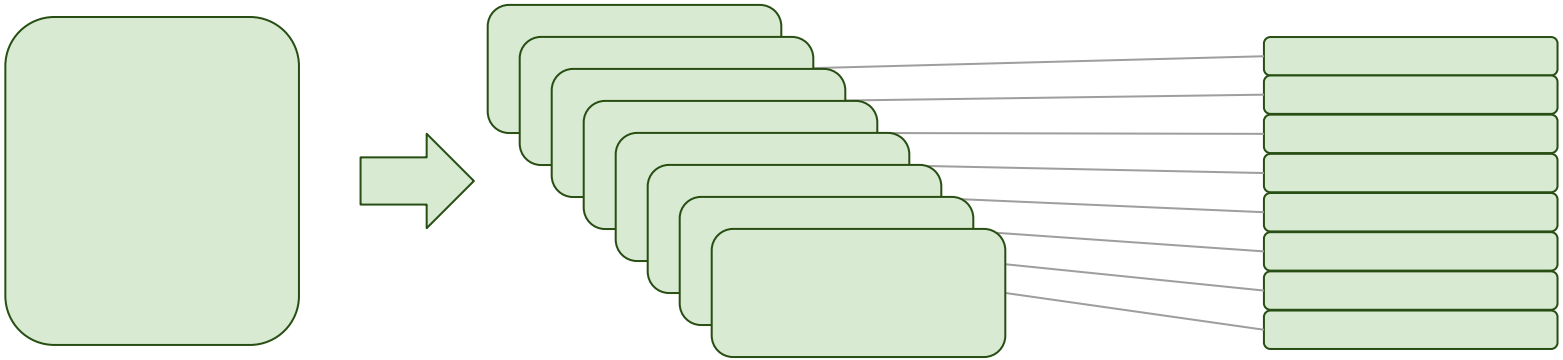
Spark UDFs - Bad Ideas

- Make modifications to an external database.
- Implement functions already defined by an API.
- UDFs that are too costly

In essence, they are not magic functions!

Window Functions

Another interesting option for Spark



Window builders

Helping to define the window you will work on

Methods

- `partitionBy`
- `orderBy`
- `rowsBetween`
- `rangeBetween`

Attributes

- `unboundedPreceding`
- `unboundedFollowing`
- `currentRow`

Window Functions: An Example

Example through code

Scala

```
import org.apache.spark.sql.expressions.Window
val window = Window.
     partitionBy(weekofyear($"timestamp")).
     orderBy($"timestamp").
     rowsBetween(-1, Window.currentRow)
orders.withColumn("avgWindow", avg($"total") over window).show
```

SQL

```
sql("SELECT *, avg(total) over (
           PARTITION BY weekofyear(timestamp)
           ORDER BY timestamp
           ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) as avgTotal
     FROM orders").show
```

Q&A

Advanced Features Recap

You Should Be Able to Answer

- What is a Spark UDF?
- What is the cost of a Spark UDF?
- What are Spark Window functions?
- How to define Windows from a DataFrame?

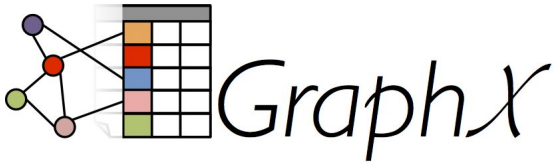
Beyond Spark Core

Other Spark Components

Spark Streaming

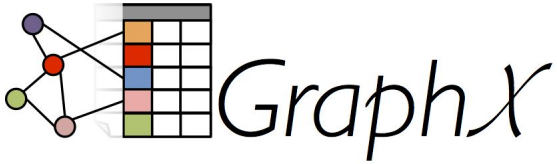
Spark Streaming — Continuous Data Processing

GraphX/GraphFrames — Large-Scale Graph Analysis



ML/MLlib — Machine Learning Algorithms Implementations

Other Spark Components



Spark Streaming

Streaming vs Batching

- **Batch Processing** takes a “static” dataset and processes it in a single swoop.
- **Stream Processing** assumes your dataset is a continuous stream of data.

Spark Streaming

Example of inputs and outputs

Spark Streaming Use Cases

Powerful real-time applications

- Immediate notifications on **social media:**
  - Tags, keywords, mentions, topics, weather conditions
- Alerts upon **malicious activity:**
  - Network, accounts, credit cards
- **Internet of Things (IoT):**
  - Traffic cameras, distributed sensors

Spark Streaming Use Cases

Powerful real-time applications

- Financial **market trends:**
  - Stocks, products, services, price hikes, price drops
- Historical data **compared against live data:**
  - Weather patterns, website bounce rates

Spark Streaming

Who's using this?

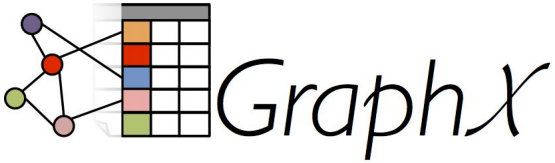
CERN, the biggest physics laboratory in the world, generates large volumes of data that must go through complex tests and models.

Apple collects data from their platform, services, and devices to understand their market share. It also enhances their media store experience.

Every single search is **processed and enriched** in near real-time to monitor **user engagement** and find revenue **opportunities**.

Graphx and GraphFrames

Why are graph problems important?



- Graph models show **interactions between entities.**

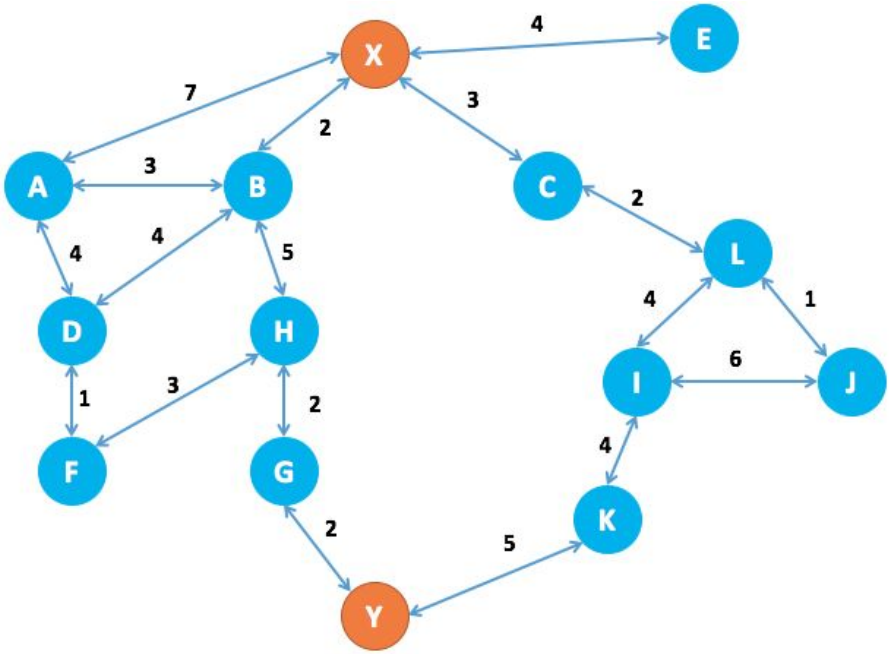
- **Real-world** problems can be **modeled** with graph-related algorithms. The solutions are relevant for many **industries**:
  - Transportation
  - Manufacturing
  - Information retrieval
  - Social media
  - Entertainment

Graphx and GraphFrames

Use cases

Transportation

- **How do you get from point A to point B efficiently?**
  - *Dijkstra's Shortest Paths algorithm*
- **How do you get to all points in a route efficiently?**
  - *Traveling Salesman Problem approximation algorithms*

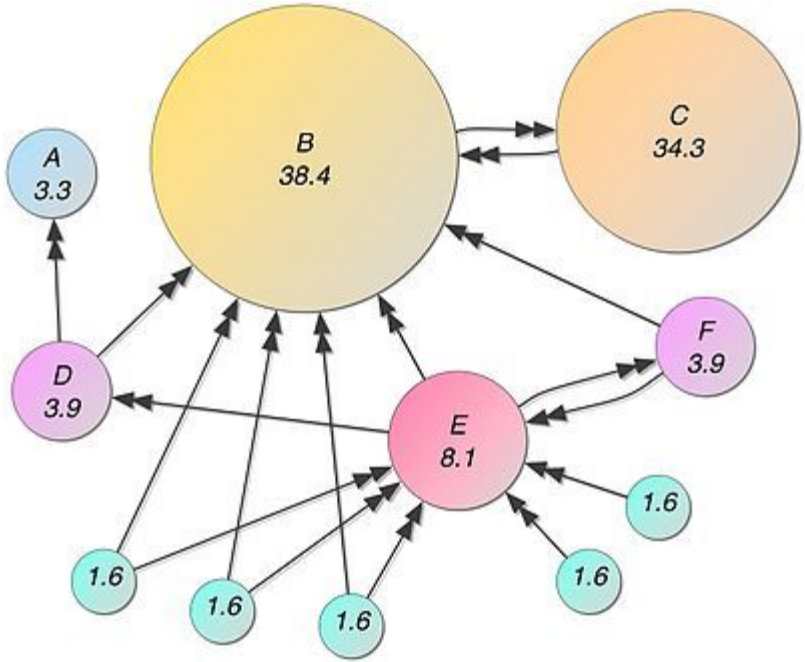


Graphx and GraphFrames

Use cases

Skill Ranking, Document Relevance Ranking, ...

- **How do you find experts in a community without *a priori* scores?**
  - *PageRank algorithm*
- **How do you figure out which web pages are most relevant for a query?**
  - *HITS algorithm*

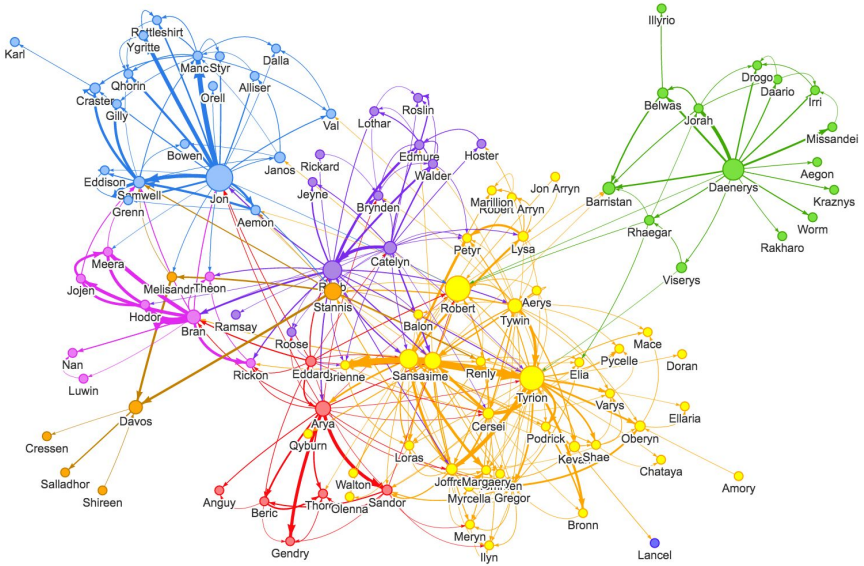


Graphx and GraphFrames

Use cases

Social Networks Analysis

- **Who are the key people that join sub-communities in a larger community?**
  - *Articulation Points algorithm*
- **Which are the tightly connected groups in a large community?**
  - *Strongly Connected Components algorithm*



Graphx and GraphFrames

Who's using this?

Large probabilistic clustering and graph diffusion algorithms are in the works. GraphX is making this analysis possible at the Netflix scale.

Uses intent data to find new customers for advertisers. They present a propagation-based model that uses GraphX and Pregel to identify possible customers for each advertiser.

Leverages GraphX to efficiently find top LinkedIn feed influencers in different communities, and by different actions.

Spark ML/MLlib

Why is ML for Big Data important?

Spark MLlib

- Datasets for many **real-world problems** today are humongous, and **cannot be fit in a single machine.**
- State-of-the-art **machine learning** models can use these huge datasets to improve their generalization capabilities.
- However, model **training can no longer happen on a single machine**. It requires distributed clusters of machines, instead.

Spark ML/MLlib

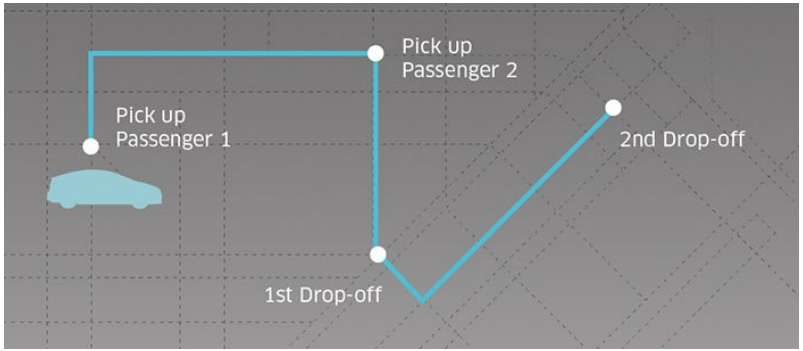
Use cases

Retail Industry

- Demand prediction
- Ad placement for maximum revenue

Transportation Industry

- Matching drivers with riders to minimize pickup time and maximize revenue for cab hailing companies

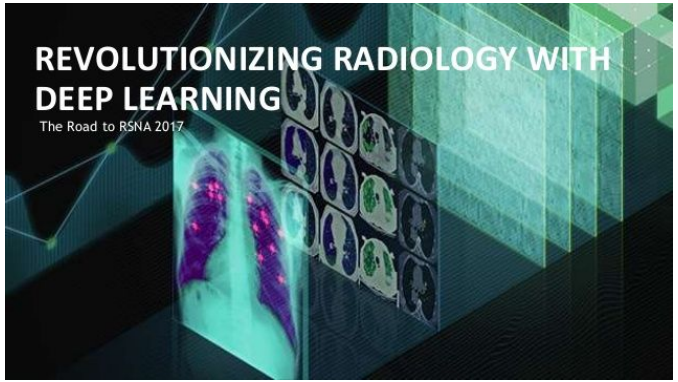


Spark ML/MLlib

Use cases

Health Industry

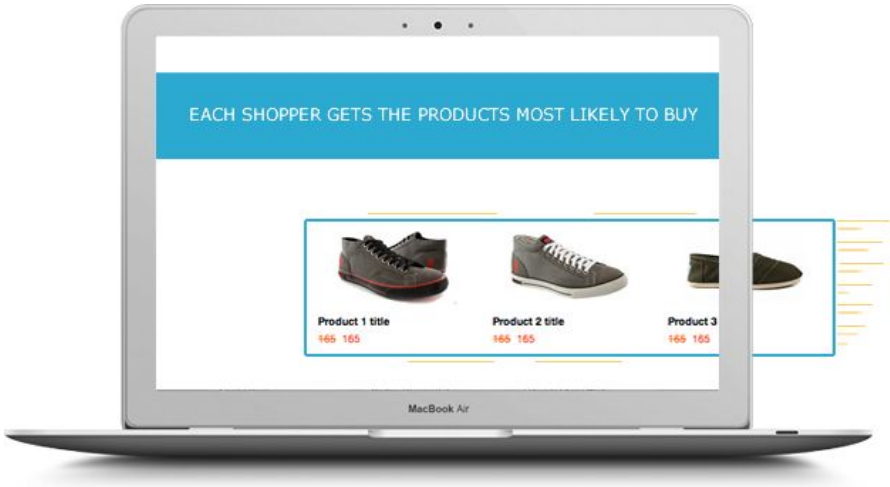
- Diagnosis using medical imaging
- New drugs discovery



eCommerce

- Recommending products to buy based on personal preferences

- Detecting when is the best time to make an offer to a client based on observed habits



Spark ML/MLlib

Who's using this?

UberEATS is using it to improve relevance and reliability of their recommendation system and serves to find trade-offs between different business metrics.

BBVA, the second largest bank in Spain, uses it to classify 700K daily transfers. It also uses it for pattern recognition in anomalous transactions.

Verizon uses it for near real-time predictive model building. This works on top of their cellular network.

The One Tech to Rule Them All....

See what we did there?

Is Spark a Swiss Army knife?

The One Tech to Rule Them All....

See what we did there?

...Not Really!

Question Time!

Do you know the answer?

What is the **REAL CHALLENGE**

of working with Big Data?

Question Time!

What else could it be?

Dealing with

HUMONGOUS AMOUNTS

of Data?

Question Time!

Something is missing...

Processing

HUMONGOUS AMOUNTS

of Data

Question Time!

Now it is clearer!

Processing

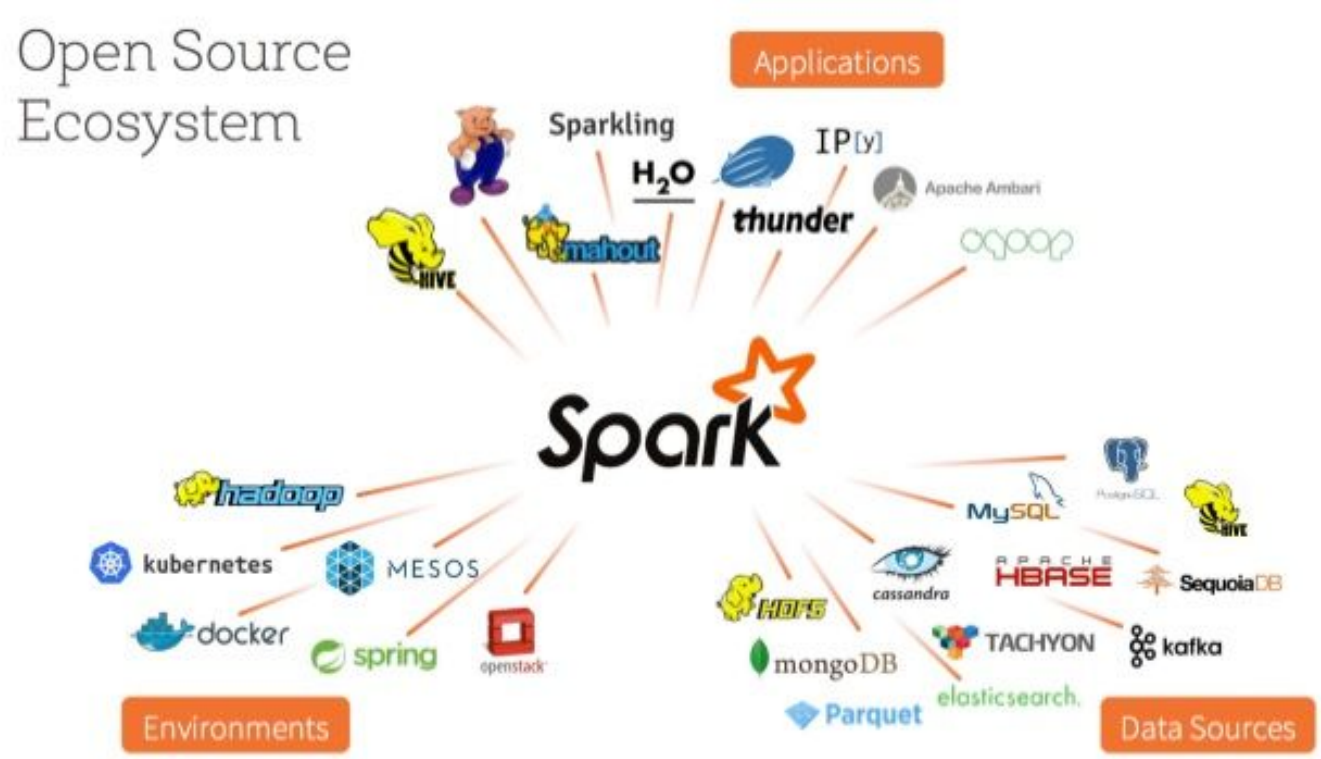
HUMONGOUS AMOUNTS

of Data

at high speed

Far More About Spark

Technology explosion around it...



.... And even **MORE**!

Technology Areas

A quick review list

- **Traditional RDBMS through JDBC:** MySQL, PostgreSQL
- **NoSQL Databases**: Cassandra, MongoDB, Redis
- **In-Disk Distributed Storage:** HDFS / S3 / GCS
- **In-Memory Distributed Storage:** Alluxio (Tachyon), Apache Ignite
- **Distributed File Format:** Parquet
- **Distributed Message Queues:** Apache Kafka
- **Resource Managers:** Yarn, Mesos, Kubernetes
- **Workflow Schedulers:** Airflow, Luigi

Traditional RDBMS through JDBC

Classic database systems

- Databases that **store structured data.**
- **Easy manipulation** of the data is easy with common SQL.

  Location, processing, and extraction become simple tasks.
- Tables are **related** to each other through **keys.**
- They use the **JDBC** connector, which is easier to integrate.

NoSQL Databases

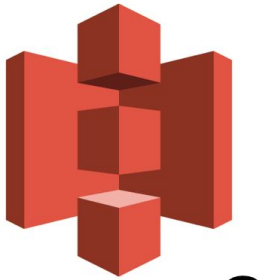
Flexible non-tabular structures

- Database design that can accommodate various including **key-value, document, columnar, and graph formats**.
- Compared to relational databases, NoSQL databases are **more scalable** and have **superior performance**.
- Used for large volumes of **rapidly changing structures**.
- Work with **semi-structured** and **unstructured** data.

In-Disk Distributed Storage

Distributing partitions of files on disk

"Storing data on multitude of standard servers, which behave as one storage system although data is distributed between these servers."



- By design, it solves these issues:
  - It is flexible to **add and remove nodes on the fly**.
  - It is reliable with **replication and sanity checks.**
  - It is scalable with **no upper limit in capacity.**
  - Its **cost is prohibitively low.**

In-Memory Distributed Storage

Distributing partitions of files in memory

- **In-Memory File System** as an interface to its in-memory data.
- Similar functionality to **Hadoop HDFS, but in memory**.
- Each file is split on separate **data blocks** and stored in **cache**.
- Co-locates **data with compute power,** resulting in run time of jobs improving by several orders of magnitude.

Distributed File Format

File formatting for Big Data era

- Free and open-source **column-oriented** data store of the Apache Hadoop ecosystem.
- **Nested data structures** stored in a flat columnar format.
- Traditional approaches store data as row-oriented, making Parquet more **efficient in terms of storage and performance.**

Distributed Message Queues

Communication is important at every level

- Message queues are software components for **inter-process or inter-thread communication**.
- Spreading the queues **across a cluster of nodes** and **replicating** them provides fault tolerance and high availability.
- Their implementations yields systems that let you have distributed, **decoupled architectures**.

Resource Managers

Making “hardware” fault tolerant too

- These technologies are responsible for **tracking the resources** in a cluster and **scheduling the applications**.

- For every application to be executed on the cluster, **resources are assigned** and tracked by the manager.

- Resource managers can **work in different manner**; some use message queues, schedulers, or a combination of both.

Workflow Schedulers

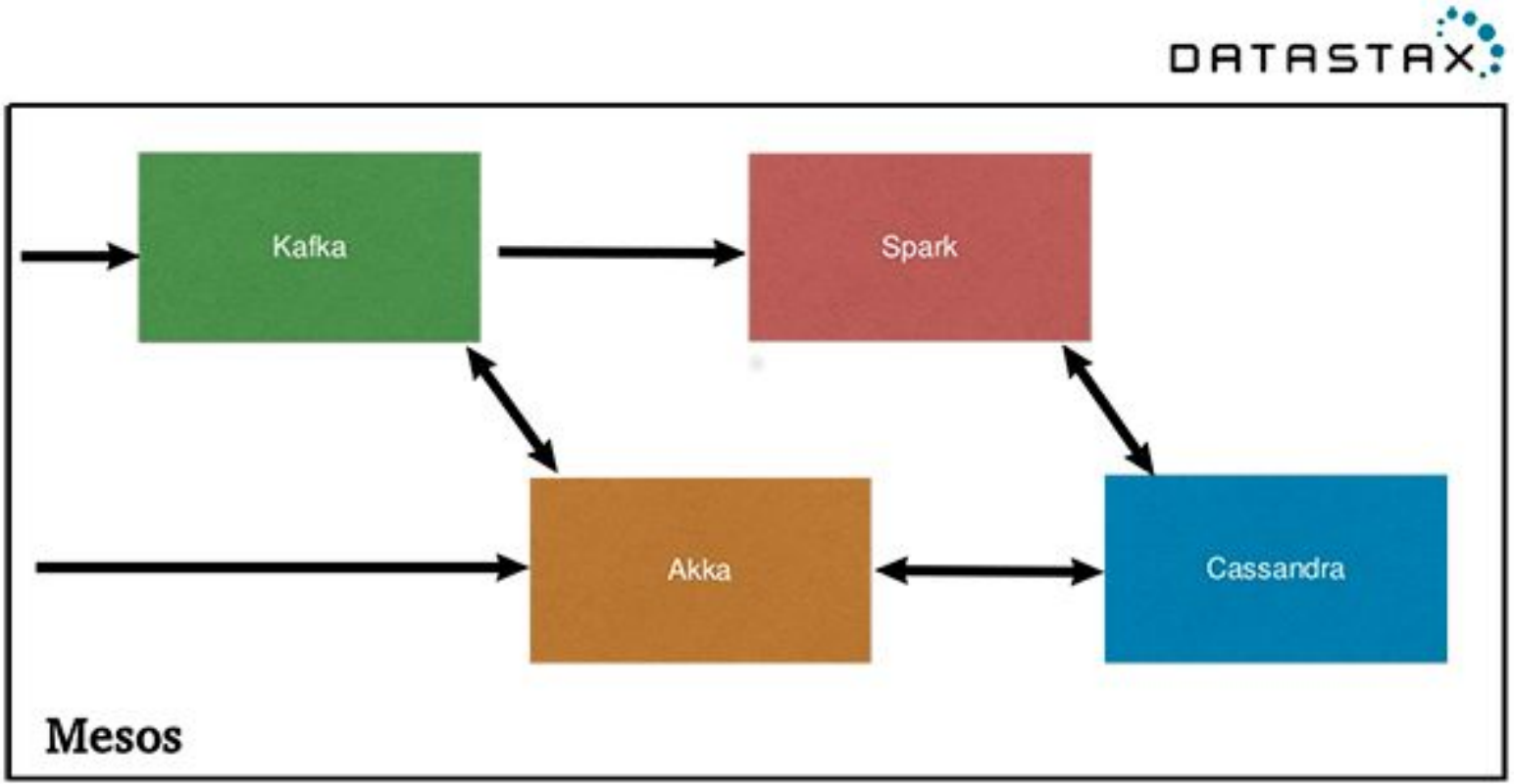
Planning your ETL pipeline with care and scalability

- **Design** and execute **computationally heavy workflows.**
- In general, it lets you create **complex task execution plans (DAGs)** by linking or triggering desired behaviors.
- It is possible to specify **synchronous and asynchronous tasks**, create **conditional paths**, parallelize events, etc.

The Big Data Proposed Stack

S M A C K

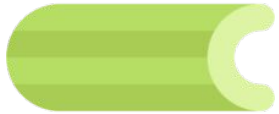
The Big Data Proposed Stack



A project at Wizeline

Wizeline is creating a **Big Data pipeline** for a multi-national media company that uses

- **Airflow** to schedule hourly, daily and weekly DAGs
  - Internally uses **Celery**, **PostgreSQL**, and **Redis**
- **PySpark** to clean ~200 GBs of daily files
- **S3/GCS** to store files throughout the process
- **Kubernetes** to control the platform resources
- **BigQuery** to run SQL analysis over datasets
- **Keybase** to have version control over credentials



Q&A

FURTHER READING

Spark at Apple Inc
https://databricks.com/session/apache-spark-at-apple

Microsoft using Spark for Bing
https://databricks.com/session/five-lessons-learned-in-building-streaming-applications-at-microsoft-bing-scale

Window Functions in Spark SQL
https://databricks.com/blog/2015/07/15/introducing-window-functions-in-spark-sql.html

Scala UDFs in PySpark
https://sigdelta.com/blog/scala-spark-udfs-in-python/

For the Knowledge Hungry...

No-SQL
http://nosql-database.org/

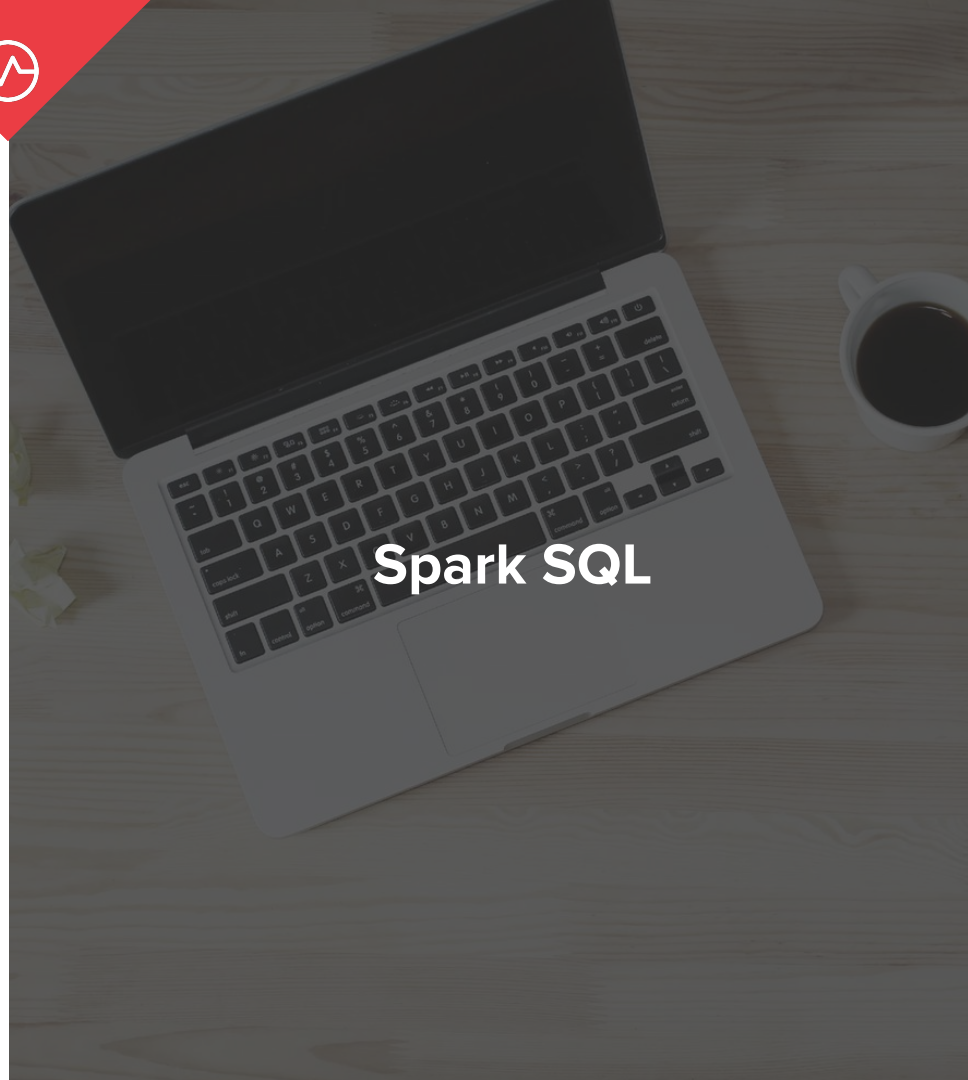
Tolkien for Spark
https://www.amazon.com/Fellowship-Ring-Being-First-Rings/dp/0547928211/ref=pd_lpo_sbs_14_img_0?_encoding=UTF8&psc=1&refRID=WNBJR5XJGF1SXNPN85GR

Distributed Data Storage
http://www.googlinux.com/understanding-distributed-data-storage/index.html

Using Scala UDFs in PySpark
https://medium.com/wbaa/using-scala-udfs-in-pyspark-b70033dd69b9

Assignment

To Work on Your Own at Home



Assignment

Images MetaData

In this assignment you will be working with the profiles dataset on this URI

gs://de-training-input/profiles/

It contains JSON objects that contain **profile information along with a profile picture**.

The picture will be a **PPM** format image encoded in **base64.**

TASK: Retrieve the metadata information (magic identifier, width, height, maximum value of colour) from the header of each profile picture. Store the metadata as additional fields.

To accomplish this task you should use UDFs

PPM Files Specifications

PPM files have two parts, **the header and the image data.**

The header has **at least** three parts. It's **normally delineated by carriage returns and/or linefeeds** but the PPM specification **only requires white space to be considered valid**.

Part one, a "magic" PPM identifier. Either "P3" or "P6" (not including the double quotes!)
Part two, the width and height of the image as ASCII numbers.
Part three, the maximum value of the colour components for the pixels.

The last part allows the format to **describe more than single byte (0..255)** colour values

PPM Files Specifications

These are VALID examples of PPM headers you might observe.

Header example 1

```
P6 1024 788 255
```

Header example 2

```
# Not a P3 image
P6
1024 788
# Invalid value 250
255
```

Header example 3

```
P3
1024 # the image width
788 # the image height
# A random comment
1023
```

Comments can be **placed anywhere** with a "#". They extend **until the end of the line.**

Expected Input (Each line in the dataset)

```
{
  "firstName": "Nadine",
  "lastName": "Booth",
  "isAlive": false,
  "age": 80,
  "address": {
    "streetAddress": "Woodbine Street",
    "city": "Gibsonia",
    "state": "Georgia",
    "postalCode": 7139
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "(897) 480-3587"
    }
  ],
  "image": {
    "data": "iVBORw0KGgoAAAANSUhEUgAAAgA....",
   }
}
```

Remember, each line in the dataset is a complete JSON string.

This is just an expanded version.

Expected Output (Each new line in your results)


```
{
  "firstName": "Nadine",
  "lastName": "Booth",
  "isAlive": false,
  "age": 80,
  "address": {
    "streetAddress": "Woodbine Street",
    "city": "Gibsonia",
    "state": "Georgia",
    "postalCode": 7139
  },
  ...
  "image": {
    "data": "iVBORw0KGgoAAAANSUhEUgAAAgA.....",
    "metadata": {
        "magic": "P6",
        "width": 512,
        "height": 625,
        "max_value": 255
    }
  }
}
```


You will keep the rest of the JSON intact.

Just add a `metadata` sub-object to the `image` entry with the processed data as shown here.

Use the field names `magic`, `width`, `height` and `max_value`.

Where Can I Get this Presentation?

PDF is on the Slack Channel!

C8 - Data Engineering Academy

Your feedback is very valuable!

https://goo.gl/forms/qAX2xZEWZYDjdYbe2

Survey Answers == Class Attendance

CONGRATULATIONS!

You've completed your first step towards becoming a Data Engineer!

We hope to see you back in future courses!


THANK
YOU
WIZELINE®