Hadoop Ecosystem

Bernard Lee Kok Bang

Hadoop ecosystem

- A powerful tool for transforming and analyzing massive data sets on a **cluster of computers**
- Consists of hundreds of distinct technologies
- Hive, Hbase, Spark, Flume, Kafka, Zookeeper, Mesos, Hortonworks, etc…
- Import and export both relational and non relational data stores

Hadoop installation

- **Requirements**
- 64 bit Windows, MacOSX
- 4GB of RAM for virtual machine (more the better)
- Download and install virtual machine

Hadoop Installation: Oracle VM VirtualBox

1. On PC/Mac

- Download virtual machine: **VirtualBox** (https://www.virtualbox.org)
- Choose the appropriate platform packages
- Require VirtualBox in order to run Hadoop

Horthonworks data platform (hdp) Sandbox

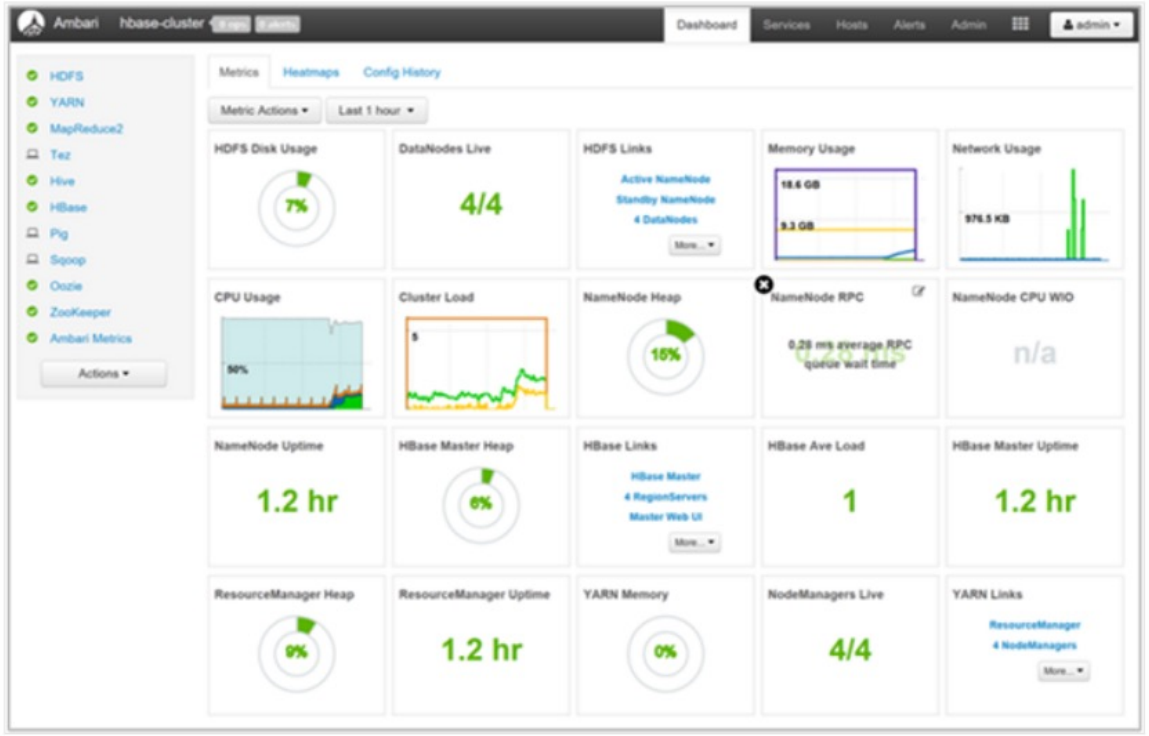
- a ***virtual environment*** provided by Hortonworks for ***learning***, ***testing***, and ***experimenting*** with Hadoop and associated technologies
- allows users to ***explore Hadoop ecosystem features*** without the need for a full-scale deployment
- HDP Sandbox comes preconfigured with a range of ***Hadoop components*** and related technologies commonly used in ***big data processing and analytics***
- ***Hadoop components (you will come across with lots of animals!!!):***
  - ***HDFS (Hadoop Distributed File System)***,
  - ***YARN (Yet Another Resource Negotiator)***,
  - ***MapReduce***,
  - ***Hive***,
  - ***Pig***,
  - ***Spark***,
  - ***HBase***

Download hdp sandbox (v. 2.6.5) here: https://bit.ly/3Kd8i53
*** later version no longer an open source**

What is a sandbox?

- **isolated testing environment** that enables users to run programs or open files **without affecting** the application, system or platform on which they run.
- Software developers use sandboxes to **test new programming code**
- Cybersecurity professionals use sandboxes to test potentially malicious software.
- business : a controlled environment supervised by a regulatory authority within which existing regulations are relaxed or removed to allow businesses to more **freely experiment with new products and services**

Apache Ambari



https://shorturl.at/vIMU4

1. open-source management platform designed to simplify the ***management***, ***monitoring***, and ***provisioning of Apache Hadoop clusters***.
2. provides an intuitive ***web interface*** and ***RESTful APIs*** for managing, monitoring, and securing Apache Hadoop clusters, as well as other related software components.

What is an API?

- stands for ***A*pplication *P*rogramming *I*nterface**
  →allows ***two software applications to talk to each other***.

- Think of it as a ***menu in a restaurant***:
  - The menu provides a list of ***dishes (functions)*** you can order.

- The ***waiter (API)*** takes your **order (request)** and brings back the ***food (response)*** from the ***kitchen (server)***.

What is REST?

- **REST** stands for ***Re*presentational *S*tate *T*ransfer**.
- a ***set of rules*** and ***architectural principles*** used when designing ***web APIs***

- RESTful API

  →An ***API*** that follows ***REST principles*** to ***allow systems to communicate*** over the ***web using HTTP***

Imagine this:

You have a **robot friend** who lives in another room. This robot can:

- Give you your toys (`GET`)
- Add new toys to your shelf (`POST`)
- Replace an old toy with a new one (`PUT`)
- Throw away a toy you don't want anymore (`DELETE`)

But... you can't go into the robot's room. So instead, you **send it notes** (messages) with very clear instructions, like:

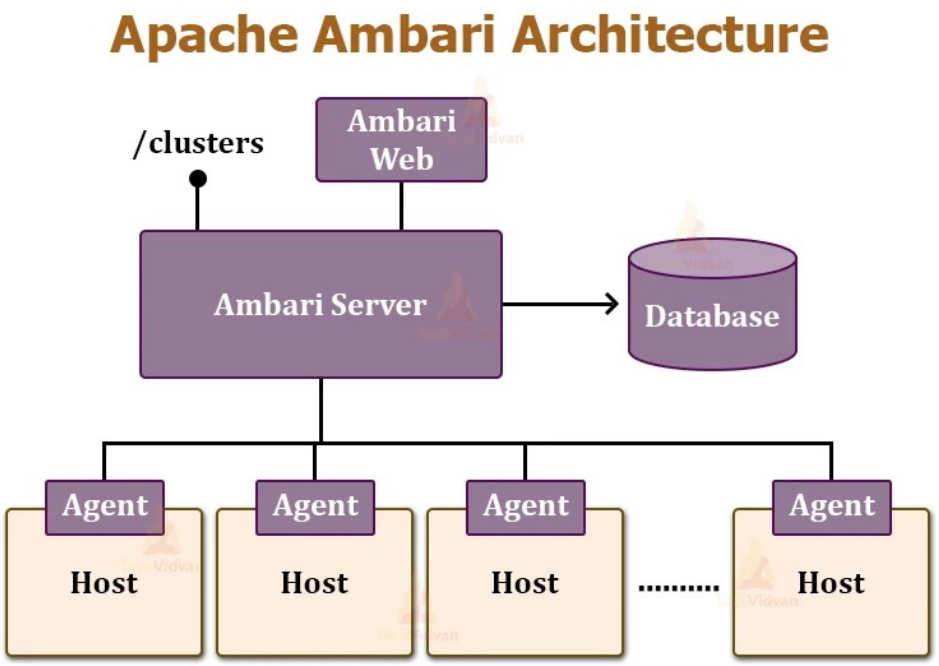
> "Hey robot, give me my toy car!"
> or
> "Hey robot, please add a teddy bear to my shelf!"

The robot reads your note, does what you ask, and sends you a note back, like:

> "Here's your toy car!"
> or
> "Teddy bear added!"

- ***You*** = the computer asking for something
- ***The robot*** = another computer that knows how to do stuff
- ***The note*** = the ***RESTful API request***
- ***The reply*** = the ***RESTful API response***

Ambari Architecture



https://shorturl.at/kIZ68

Ambari

- Use to visualize Hadoop cluster
- don’t need to worry whether running on one system or multiple nodes of entire clusters of computers
- Limiting factor: how much data we can handle…

1st: Start any web browser
2nd: Key in this address: http://localhost:1080

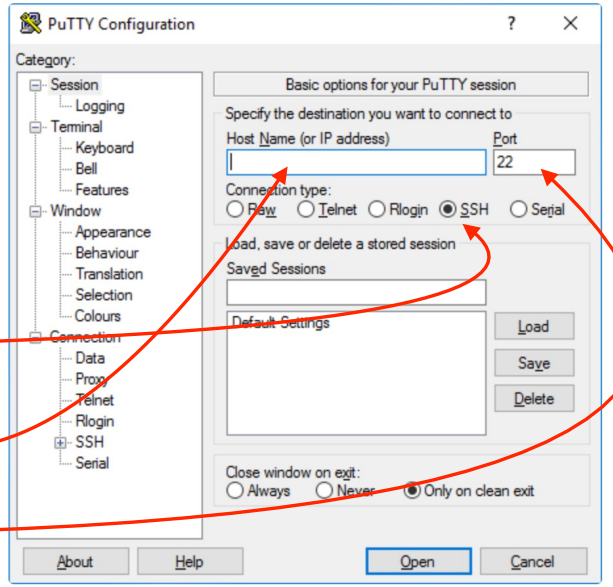
Ambari view

http://localhost:1080; or
http://127.0.0.1:1080

- Go straight to Ambari view: **127.0.0.1:8080**
- Username & p/w: **maria_dev**
- All the stuff installed on VirtualBox can be viewed using Ambari View
- More information can be viewed by clicking individual link; e.g., HDFS, Files View...

Install PuTTY

- Windows: Using command line interface [**https://www.putty.org**]
- Mac: Using **Terminal**
- Configuration:
    1. Connection type: **SSH**
    2. Host Name: **maria_dev@127.0.0.1**
    3. Port: **2222**
    4. Saved Session: **HDP** [click to reuse for future session]



Let’s play some Hadoop command line

- To list all files available in Hadoop cluster: ***hadoop fs -ls***
- To create folder in Hadoop: ***hadoop fs –mkdir ml-100k***
- To download file to local: ***wget http://media.sundog-soft.com/hadoop/ml-100k/u.data***
- To upload data from local to hadoop: ***hadoop fs –copyFromLocal u.data ml-100k/u.data***
- To delete file on hadoop: ***hadoop fs –rm ml-100k/u.data***
- To delete folder on hadoop: ***hadoop fs –rmdir ml-100k***
- List out other commands: ***hadoop fs***

“local” == a virtual machine like VirtualBox

Why need to use VirtualBox instead of running things directly on Windows?

🧁 Imagine Your Computer is a Bakery

Your **Windows computer** is like a bakery that makes **chocolate cakes** really well. 🍫🎂
But now you want to **make a special kind of cake**, like a **green tea cake** (let's pretend that green tea cake is Hadoop on Linux).

But here's the problem:

> Your bakery (Windows) **doesn't know how to make green tea cakes!** 😟

👜 VirtualBox to the Rescue!

VirtualBox is like bringing in a **tiny second bakery** that works **inside your bakery**.
This new mini-bakery can be **any type you want** — like a **Japanese bakery** that knows how to make **green tea cakes** (aka Linux for Hadoop). 🎌🍵🎂

Now, instead of changing your whole bakery, you just say:

> "Hey VirtualBox, please run this special mini-bakery (Linux), so I can make green tea cakes (run Hadoop) without messing up my chocolate cake setup (Windows)."

Download some data to play around

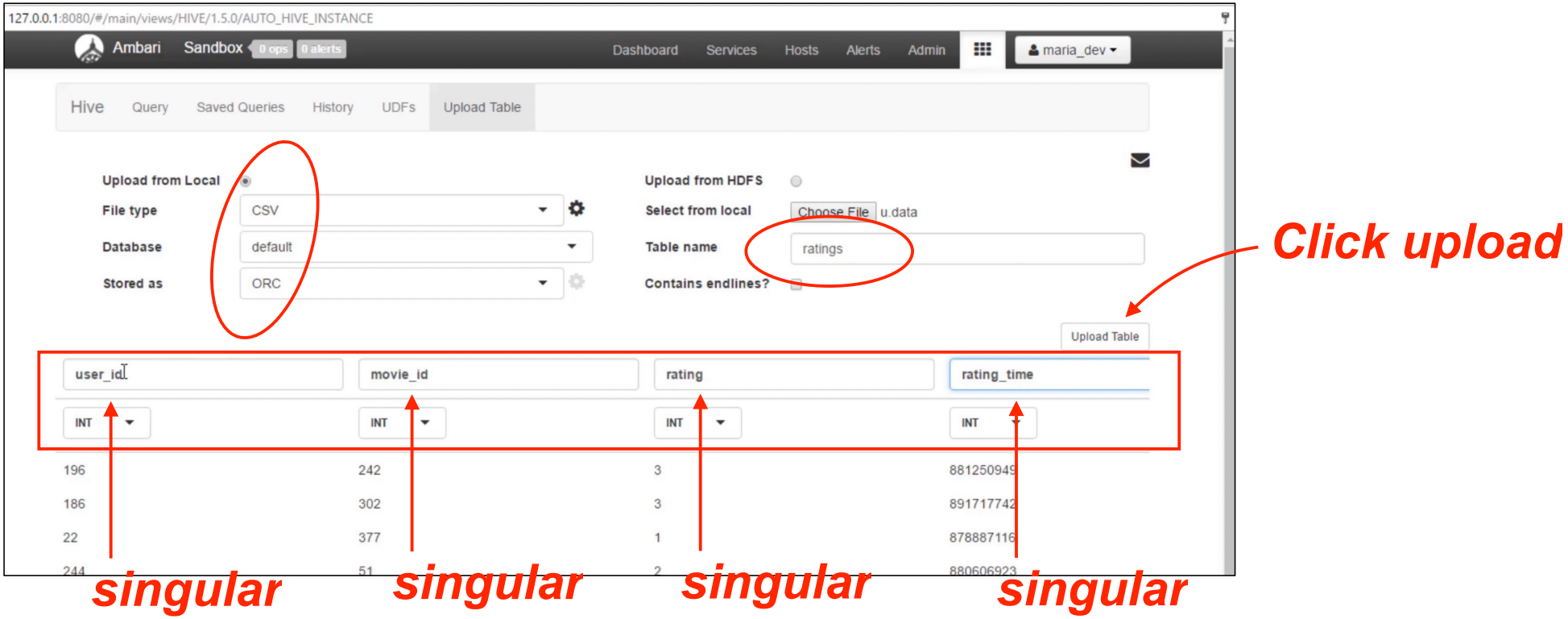
- Log on to grouplens.org (https://grouplens.org)
- Choose datasets button
- Choose MovieLens 100K Dataset
- Unzip and look for two datasets:
  →1. u.data; 2. u.item
- Load these two datasets into VirtualBox **[Ambari]**
- Choose browser interface by typing the associated address: **http://localhost:1080** [**depends on individual machine**]
- Launch the dashboard and we are good to go!!!
- Username & Password: **maria_dev**

Upload here: Dashboard/Files View/user/maria_dev

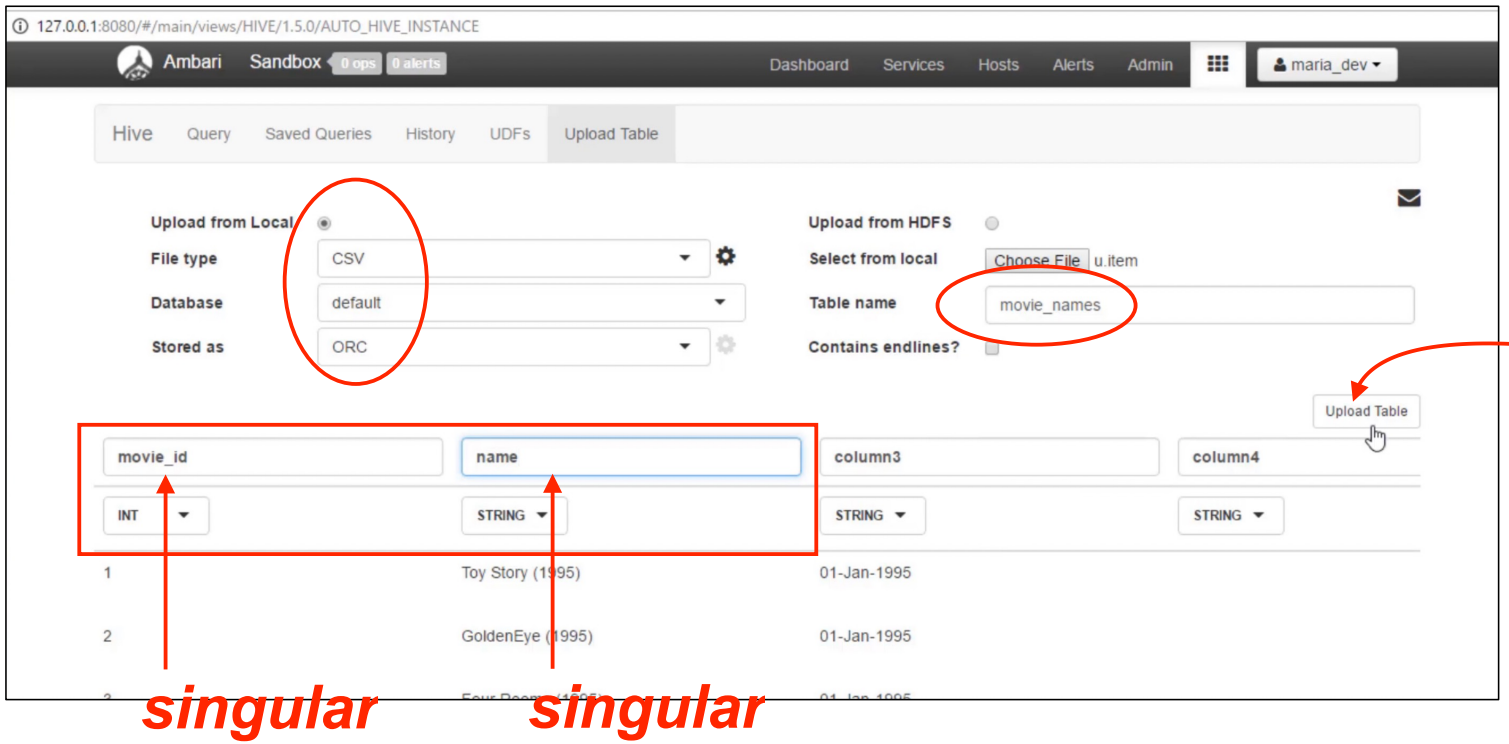
Some housekeeping naming conventions

- Table name → in plural forms, lowercase
- Field name → in singular form, lowercase
- ***camelCase*** naming system → e.g. ***numOfDonut, favDrink***
- ***snakecase*** naming system → e.g. ***num_of_Donut, fav_drink***

Hive View Example 1: u.data

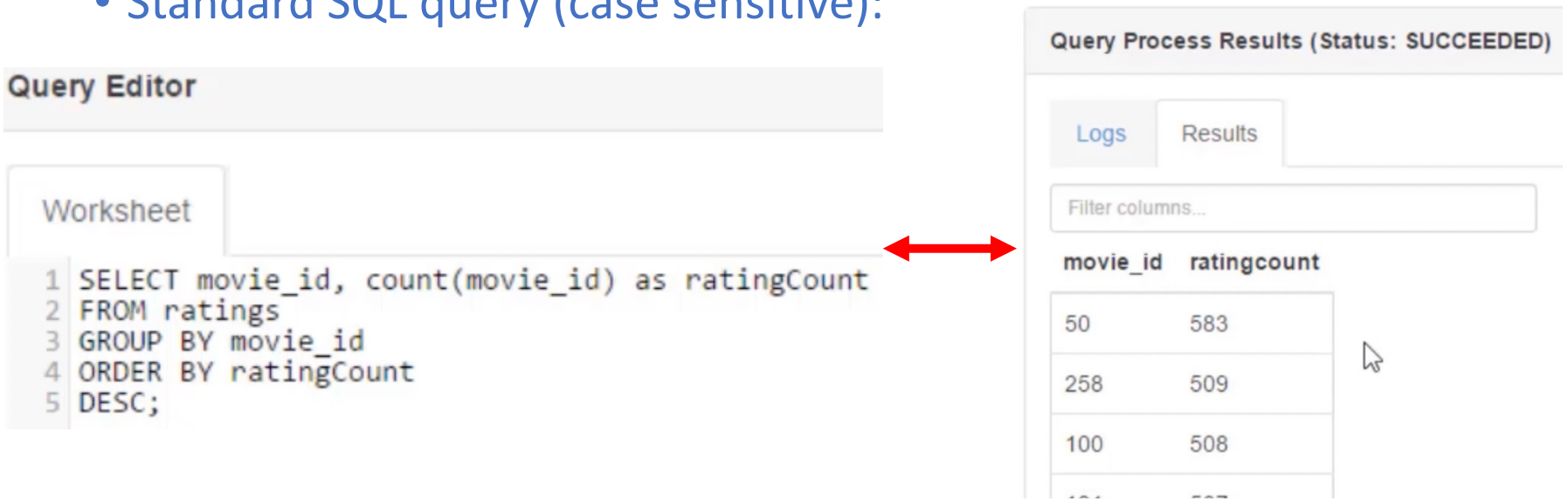


Hive View Example 2: u.item



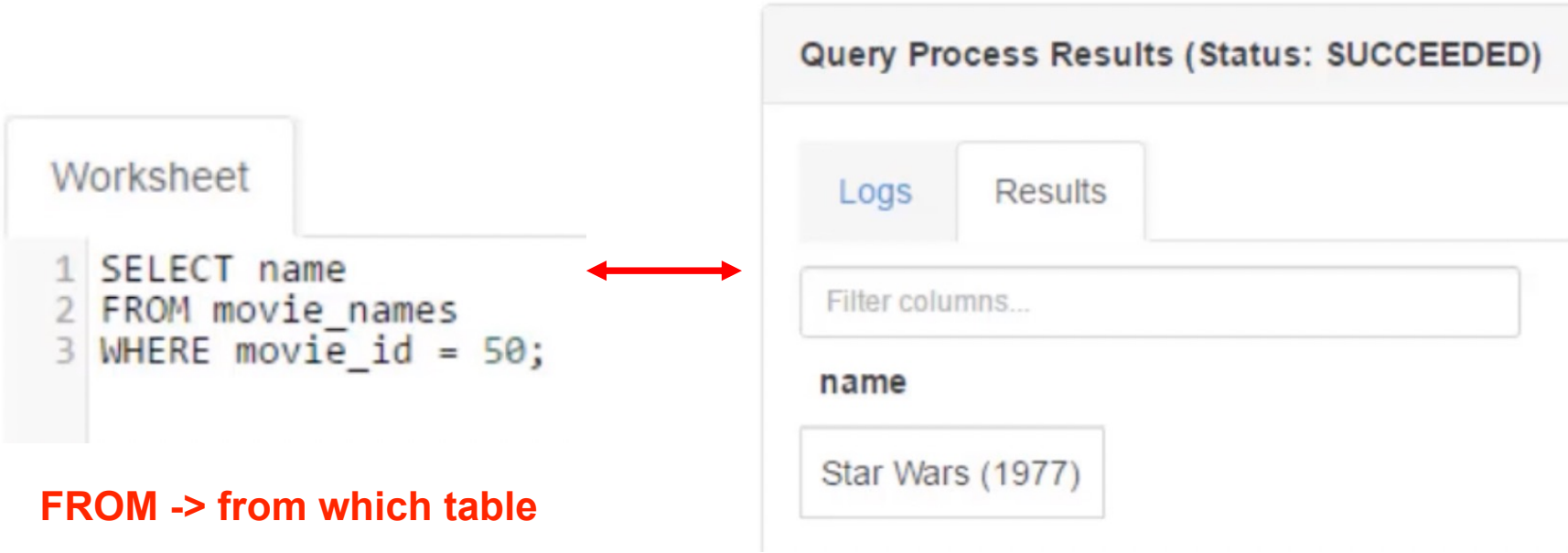
Hands on demo using MovieLens 100K data

- Using Hive to execute SQL-like queries on Movielens 100K data
- **Q: Find the most popular movie** **-> user rating: the higher the more popular**
- Standard SQL query (case sensitive):



Hands on demo (cont…)

- What's the movie name with movie_id = 50?



FROM -> from which table

Under the hood…

- How to optimize the **MapReduce** query
- figuring out the optimal way to divide the **supposed "very big data"** across **clusters of computers**
- process **in parallel**
- looks like SQL query; but it isn't [it is in **distributed manner**]
- **SQL is meant for relational database**

What is Hadoop

(not limited by single hard drive)

"an open-source **software platform** for **distributed storage** and **distributed processing** of **very large data sets** on **computer clusters** built from community hardware" – Hortonworks

Distributed processing -> transform data in other format, other system, or aggregate in some other way
-> all done in a parallel manner

- Can keep on adding **more and more computers** so the clusters and the hard drive will become part of the data storage
- A way to view all the data distributed across all the hard drive **as one single file system**
- Very **redundant** – Hadoop keep backup copies of all the data on other places in the clusters and it can automatically recover and make the data very resilient

• Community hardware -> Amazon Web Services, Google [cloud service providers]

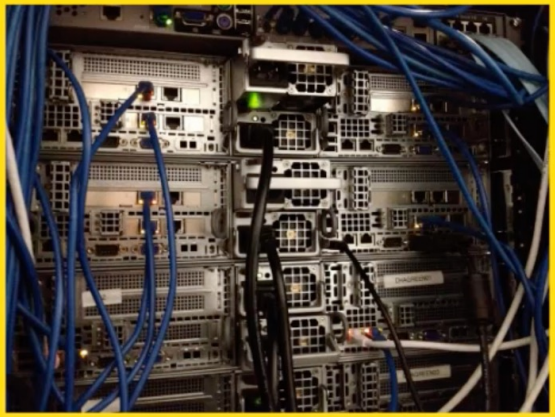
Hadoop History

1. Google file system -> foundation of Hadoop distributed storage idea

2. MapReduce -> inspired Hadoop distributed processing

- Google published GFS and MapReduce papers in 2003-2004
- Yahoo! was building "Nutch," an open source web search engine at the same time
- Hadoop was primarily driven by Doug Cutting and Tom White in 2006
- It's been evolving ever since...

Why Hadoop?

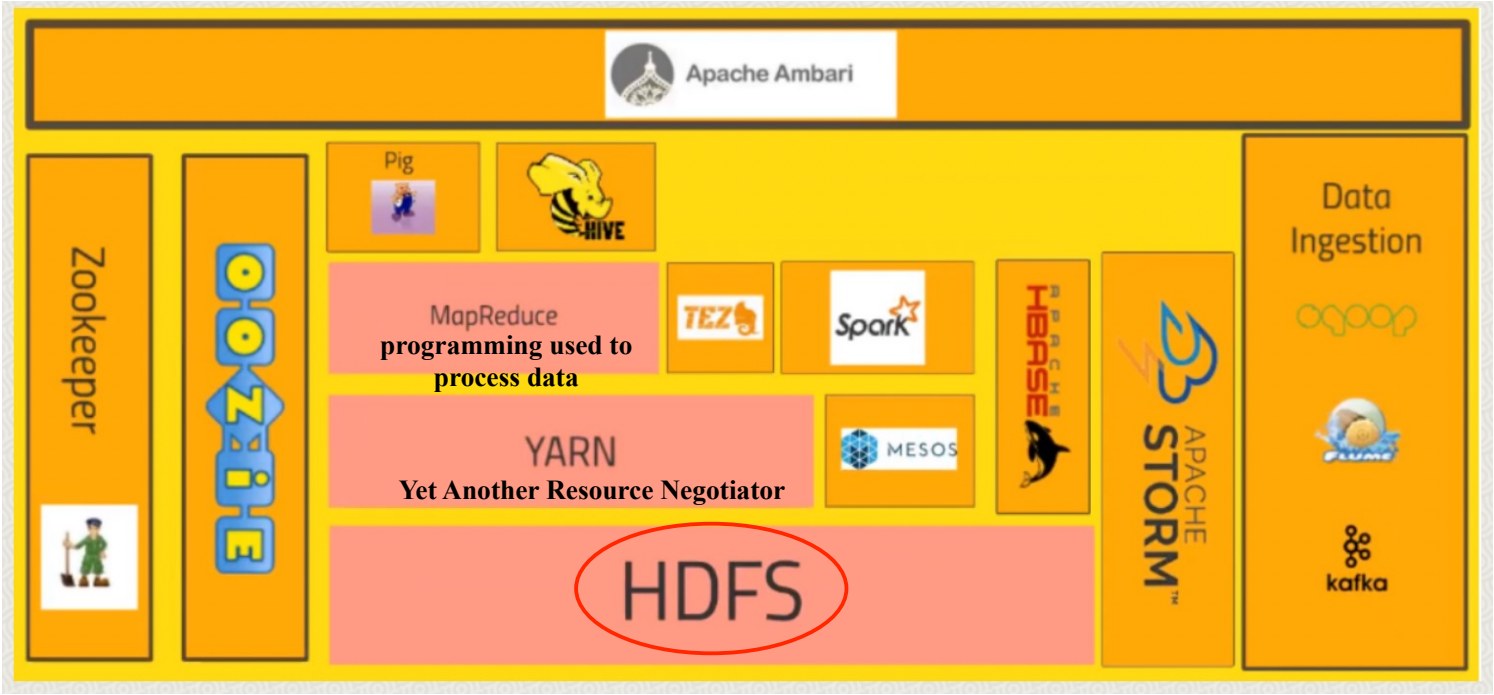


- Data's too darn big - terabytes per day
- Vertical scaling doesn't cut it
  - Disk seek times
  - Hardware failures
  - Processing times

Avoid single point of failure

- Horizontal scaling is linear
- Hadoop: It's not just for batch processing anymore

Core Hadoop Ecosystem



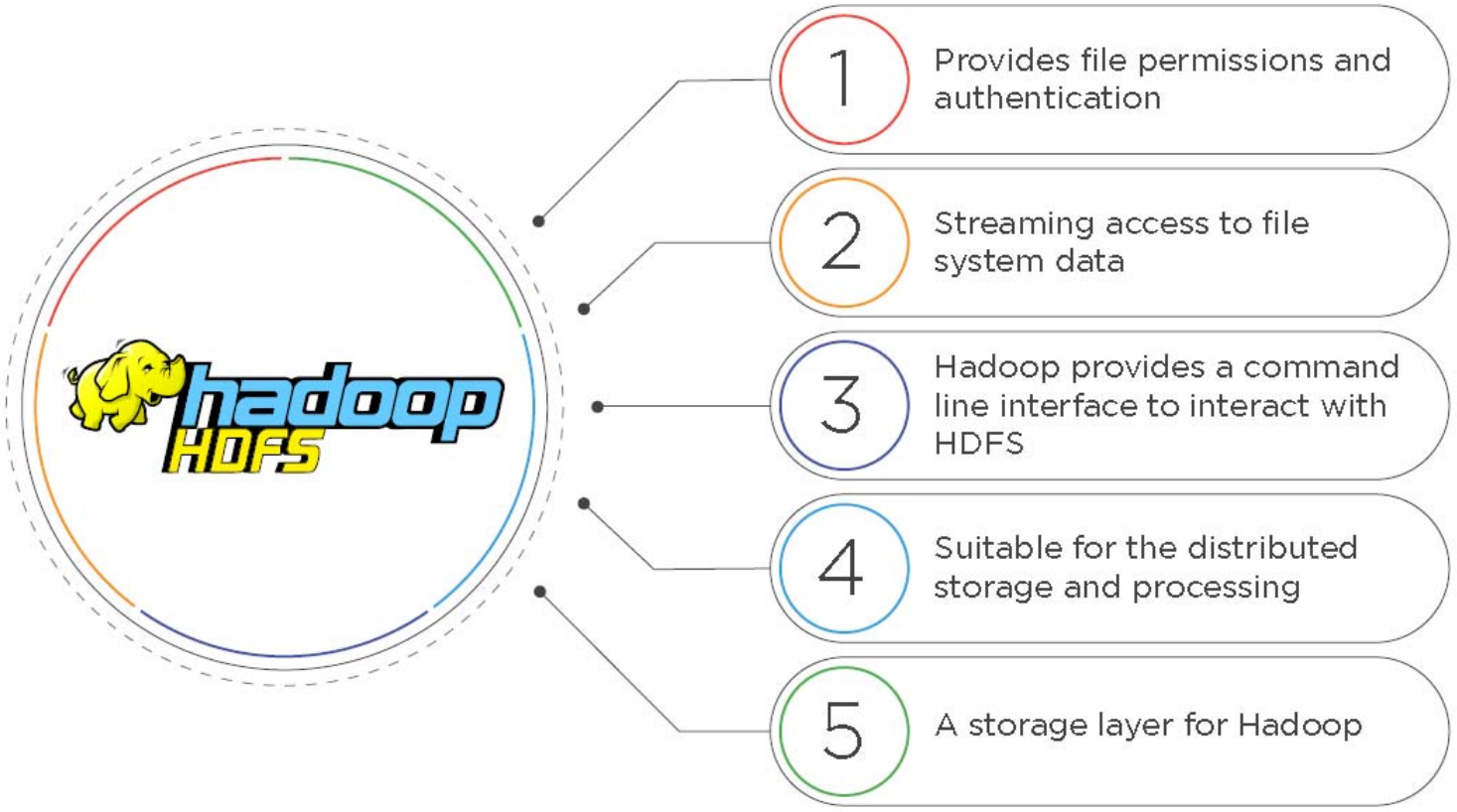
https://bit.ly/3nW58K9

External Data Storage

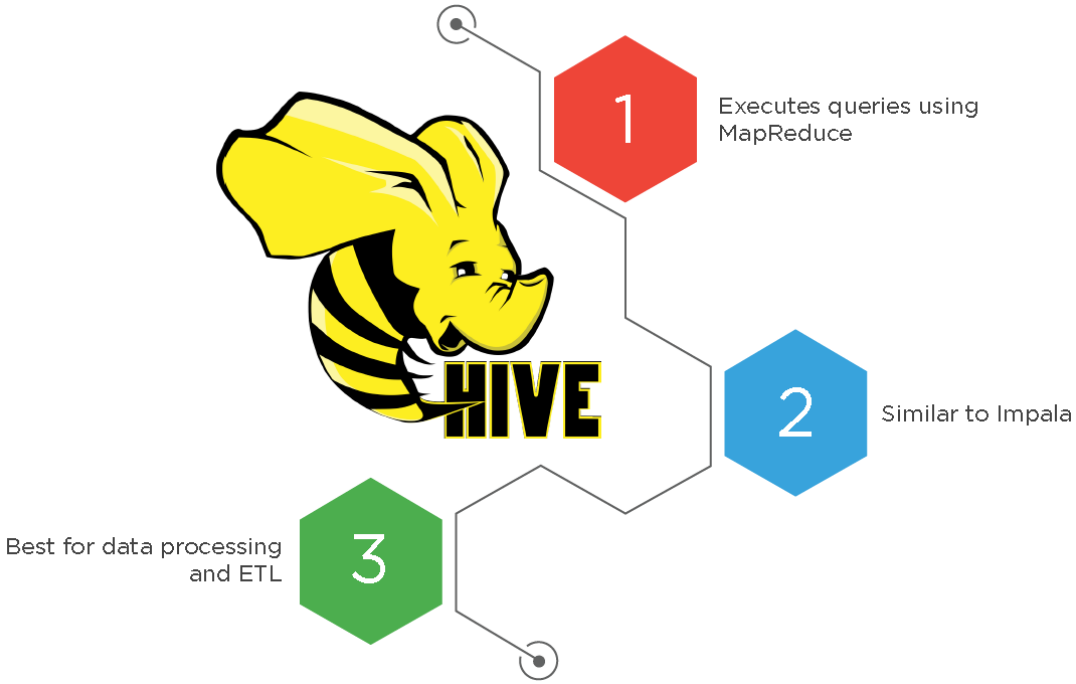
Query Engines

Apache Zeppelin

HDFS: Hadoop Distributed File System



HIVE


HIVE
1
Executes queries using MapReduce
2
Similar to Impala
3
Best for data processing and ETL

What Does ETL Stand For?

- ETL = ***E*xtract, *T*ransform, *L*oad**

Step	What it Means (Simple)	Like in a Burger Shop 🍔
Extract	Take the data from somewhere (like a database, file, website)	Get raw ingredients from the farm
Transform	Clean, fix, and prepare the data so it makes sense	Wash, chop, and cook the ingredients
Load	Put the ready data into a new place (like a data warehouse or report system)	Put the ingredients into a burger bun and serve it

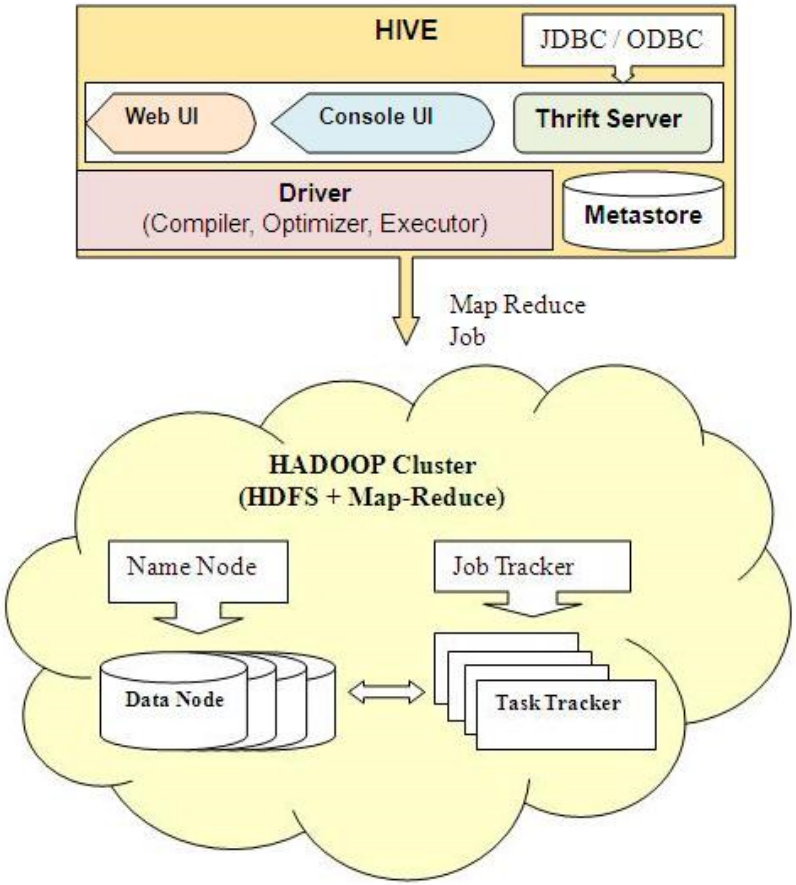
Introduction to Hive

- Hive is data warehouse infrastructure that is built on top of Apache Hadoop. It is configured to use Derby as the metadata storage (metastore). The primary responsibility is to provide **data summarization**, **query** and **analysis**.

- Hive provides mechanism to project structure onto the data and query the data using SQL-like language called **HiveQL**

- Developed by Facebook, now open source.

Hive Architecture



Apache Hive Command cheat sheet

Query

Function	MySQL	HiveQL
Retrieving information	`SELECT from_columns FROM table WHERE conditions;`	`SELECT from_columns FROM table WHERE conditions;`
All values	`SELECT * FROM table;`	`SELECT * FROM table;`
Some values	`SELECT * FROM table WHERE rec_name = "value";`	`SELECT * FROM table WHERE rec_name = "value";`
Multiple criteria	`SELECT * FROM table WHERE rec1="value1" AND rec2="value2";`	`SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";`
Selecting specific columns	`SELECT column_name FROM table;`	`SELECT column_name FROM table;`
Retrieving unique output records	`SELECT DISTINCT column_name FROM table;`	`SELECT DISTINCT column_name FROM table;`
Sorting	`SELECT col1, col2 FROM table ORDER BY col2;`	`SELECT col1, col2 FROM table ORDER BY col2;`
Sorting backward	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`	`SELECT col1, col2 FROM table ORDER BY col2 DESC;`
Counting rows	`SELECT COUNT(*) FROM table;`	`SELECT COUNT(*) FROM table;`
Grouping with counting	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`	`SELECT owner, COUNT(*) FROM table GROUP BY owner;`
Maximum value	`SELECT MAX(col_name) AS label FROM table;`	`SELECT MAX(col_name) AS label FROM table;`
Selecting from multiple tables (Join same table using alias w/"AS")	`SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;`	`SELECT pet.name, comment FROM pet JOIN event ON (pet.name = event.name);`

Metadata

Function	MySQL	HiveQL
Selecting a database	`USE database;`	`USE database;`
Listing databases	`SHOW DATABASES;`	`SHOW DATABASES;`
Listing tables in a database	`SHOW TABLES;`	`SHOW TABLES;`
Describing the format of a table	`DESCRIBE table;`	`DESCRIBE (FORMATTED|EXTENDED) table;`
Creating a database	`CREATE DATABASE db_name;`	`CREATE DATABASE db_name;`
Dropping a database	`DROP DATABASE db_name;`	`DROP DATABASE db_name (CASCADE);`

What is a Data Warehouse in Hive?

- the **data warehouse** is the ***location where all our data tables are stored*** inside Hadoop
- It's like the **main library room** where:
  - All our data tables (like "students", "marks", "teachers") live.
  - Each table is stored as files in the Hadoop system (HDFS).
- Hive organizes and manages those files so we can ***query them like a database***.

Pig


An open-source dataflow system
An alternative to writing Map-Reduce code
Converts pig script to Map-Reduce code
Best for ad-hoc queries such as join and filter

Pig

- **Pig** is a high-level platform for creating MapReduce programs used with **Hadoop**.
- Developed by Yahoo!, 40% of all its Hadoop jobs in Pig. Other user: Twitter
- The language for this platform is called **Pig** Latin.
- **Pig** Latin abstracts the programming from the Java MapReduce idiom into a notation which makes MapReduce programming high level, similar to that of SQL for RDBMSs.

Differences between Hive & Pig

- **Hive** Hadoop Component is used mainly by data analysts whereas **Pig** Hadoop Component is generally used by Researchers and Programmers.

- **Hive** Hadoop Component is used for completely structured Data whereas **Pig** Hadoop Component is used for semi structured data.

Differences Between Hive and Pig

Feature	Hive (🐝)	Pig (🐷)
Language Used	**HiveQL** (similar to SQL)	**Pig Latin** (a scripting language)
Best For	Users who know SQL (analysts, BI people)	Programmers and data engineers
Data Flow Style	Declarative (you tell what you want)	Procedural (you tell how to get it step by step)
Use Case	Running reports, data analysis, summarization	Data cleaning, transformation, and pipelines
Type of User	Business analysts, report writers	Developers, engineers
Optimization	Query optimizer under the hood (like SQL DBs)	Requires manual tuning or script control
Data Format Support	Tables with schema	Flexible with nested data and custom formats
Execution Engine	Originally MapReduce, now supports Tez & Spark too	Originally MapReduce, now supports Tez & Spark

Pig Command cheat sheet

		computation
SUM	SUM(expression)	Computes the sum of the numeric values in a single-column bag.
TOKENIZE	TOKENIZE(expression [, 'field_delimiter'])	Splits a string and outputs a bag of words.

Load/Store Functions

Function	Syntax	Description
Handling Compression	A = load 'myinput.gz'; store A into 'myoutput.gz';	PigStorage and TextLoader support gzip and bzip compression for both read (load) and write (store). BinStorage does not support compression.

Load/Store Functions

Function	Syntax	Description
BinStorage	A = LOAD 'data' USING BinStorage();	Loads and stores data in machine-readable format.
JsonLoader, JsonStorage	A = load 'a.json' using JsonLoader();	Load or store JSON data.
PigDump	STORE X INTO 'output' USING PigDump();	Stores data in UTF-8 format.
PigStorage	A = LOAD 'student' USING PigStorage('\t') AS (name: chararray, age:int, gpa: float);	Loads and stores data as structured text files.

Hadoop implementation using Hive

- Create employee dataset in excel
- save as ***employees.csv***

1	ID	Name	Salary	Position
2	1201	Cot	45000	Technical Manager
3	1202	Shasha	45000	Proof Reader
4	1203	Chunli	40000	Technical Writer
5	1204	Shafaat	40000	HR admin
6	1205	Cocot	30000	Op admin

Hadoop Implementation using Hive

- Filter data using SELECT…WHERE
  - SELECT -> retrieve data from table
  - WHERE -> condition

```
SELECT *
FROM employees
WHERE salary > 40000;
```

- Result:

Results | Query | Log | Columns

	id	name	salary	position
0	1201	Cot	45000	Technical Manager
1	1202	Shasha	45000	Proof Reader

Hadoop Implementation using Hive

- Filter data using SELECT…ORDER BY
  - ORDER BY -> retrieve data based on the selected column and sort the data result by ascending/descending order

- Result :

Results | Query | Log | Columns

	id	name	salary
0	1203	Chunli	40000
1	1205	Cocot	30000
2	1201	Cot	45000
3	1204	Shada	40000
4	1202	Shasha	45000

```
SELECT id, name, salary
FROM employees
ORDER BY name DESC;
```

Hadoop Implementation using Hive

- Filter data using SELECT…GROUP BY
  - GROUP BY -> Grouping the records based on the selected column

- Result:

Results | Query | Log | Columns

	salary	_c1
0	30000	1
1	40000	2
2	45000	2

```
SELECT salary, count(*)
FROM employees
GROUP BY salary
```

Hadoop – WordCount Exercise

- Run **Pig scripting** on the "Pig Shell (Grunt)"
- **To find out counts for each word**

```
-- Load input data
input_data = LOAD 'input.txt' AS (line:chararray);

-- Tokenize each line into words
words = FOREACH input_data GENERATE FLATTEN(TOKENIZE(line)) AS word;

-- Group by word and count occurrences
word_count = FOREACH (GROUP words BY word) GENERATE group AS word, COUNT(words) AS count;

-- Store word count results
STORE word_count INTO 'output';
```