

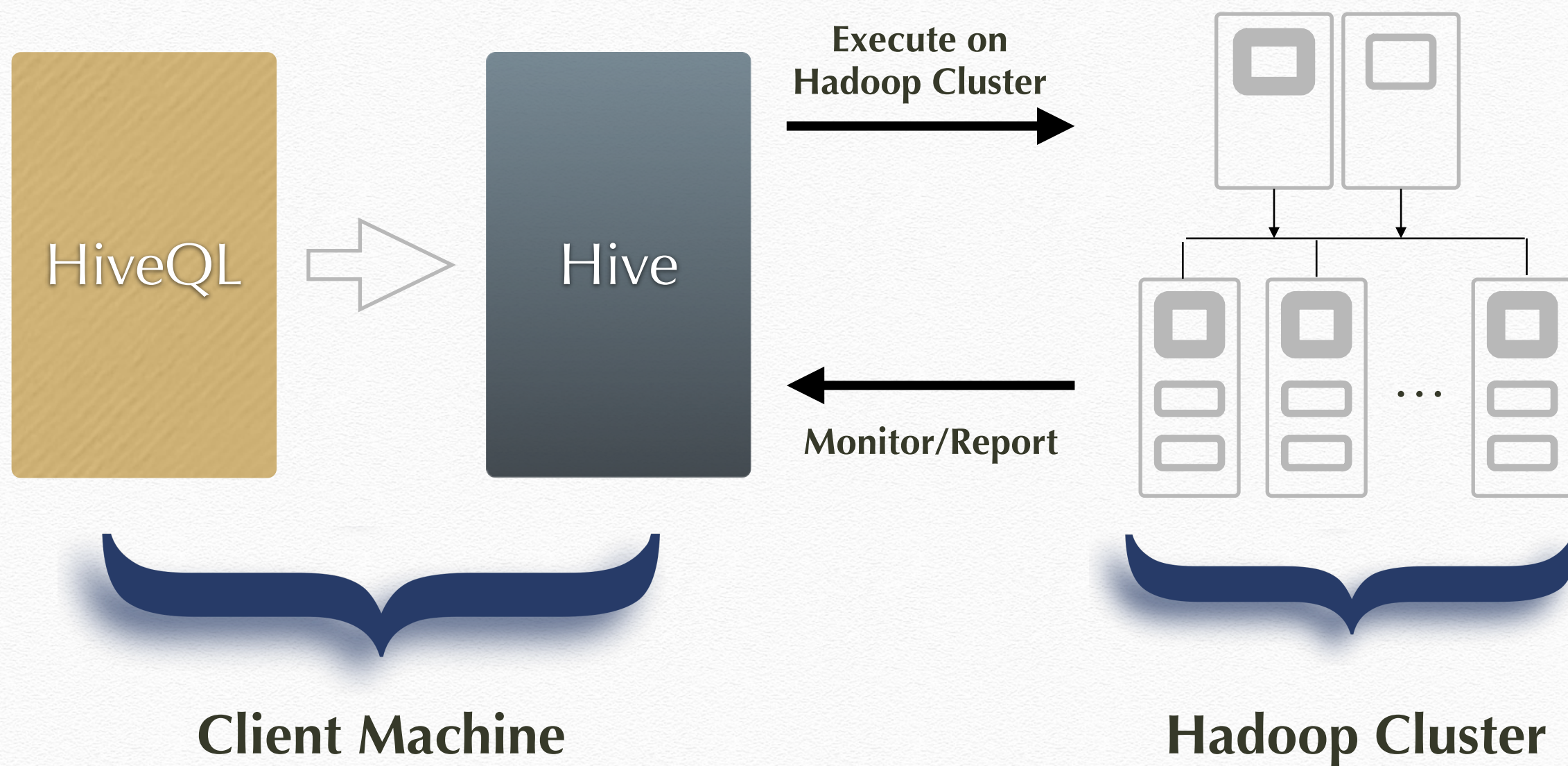
Apache Hive

Big Data - 15/04/2020



Hive Configuration

- ❖ Translates **HiveQL** statements into a set of **MapReduce jobs** which are then executed on a **Hadoop Cluster**













Hive Configuration




- ❖ Download a **binary release** of **apache Hive**: <https://mirror.nohup.it/apache/hive/>
- ❖ **apache-hive-2.3.6-bin.tar.gz**

Index of /apache/hive

<u>Name</u>	<u>Last modified</u>
 Parent Directory	
 hive-1.2.2/	2018-05-04 22:51
 hive-2.3.6/	2019-08-22 20:53
 hive-3.1.2/	2019-08-26 22:21
 hive-standalone-metastore-3.0.0/	2018-06-07 20:12
 hive-storage-2.6.1/	2018-05-12 00:26
 hive-storage-2.7.1/	2019-12-01 18:42
 stable-2/	2019-08-22 20:53



Index of /apache/hive/hive-2.3.6

<u>Name</u>	<u>Last modified</u>	<u>Size</u>
 Parent Directory		-
 apache-hive-2.3.6-bin.tar.gz	2019-08-22 20:53	221M
 apache-hive-2.3.6-src.tar.gz	2019-08-22 20:53	20M



Environment variables

- ❖ In the **bash_profile** export all needed **environment variables**

```
Air-di-Roberto:~ roberto$ cd  
Air-di-Roberto:~ roberto$ nano .bash_profile
```



```
GNU nano 2.0.6 File: .bash_profile  
  
export PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin  
export JAVA_HOME=$(/usr/libexec/java_home)  
export HADOOP_HOME=/Users/roberto/Documents/hadoop-3.2.1  
export HIVE_HOME=/Users/roberto/Documents/apache-hive-2.3.6-bin  
export PATH=$PATH:$HADOOP_HOME/bin:$HIVE_HOME/bin
```



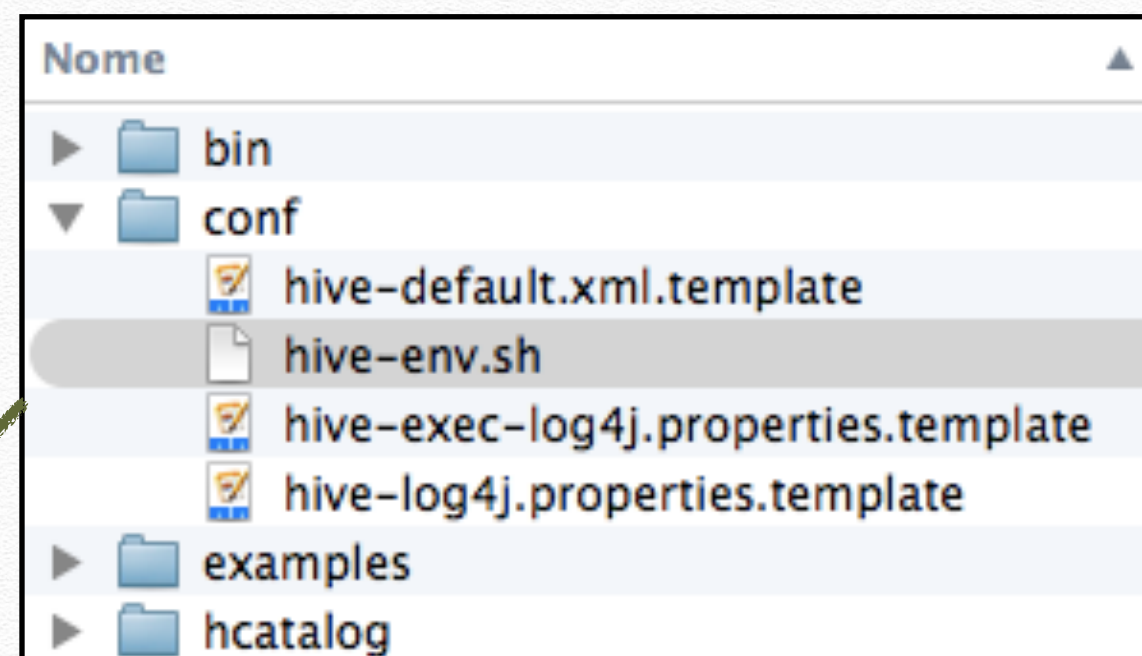
Mac OS X



Hive Configuration

- ❖ In the **conf** directory of hive-home directory set **hive-env.sh** file

set the **HADOOP_HOME**



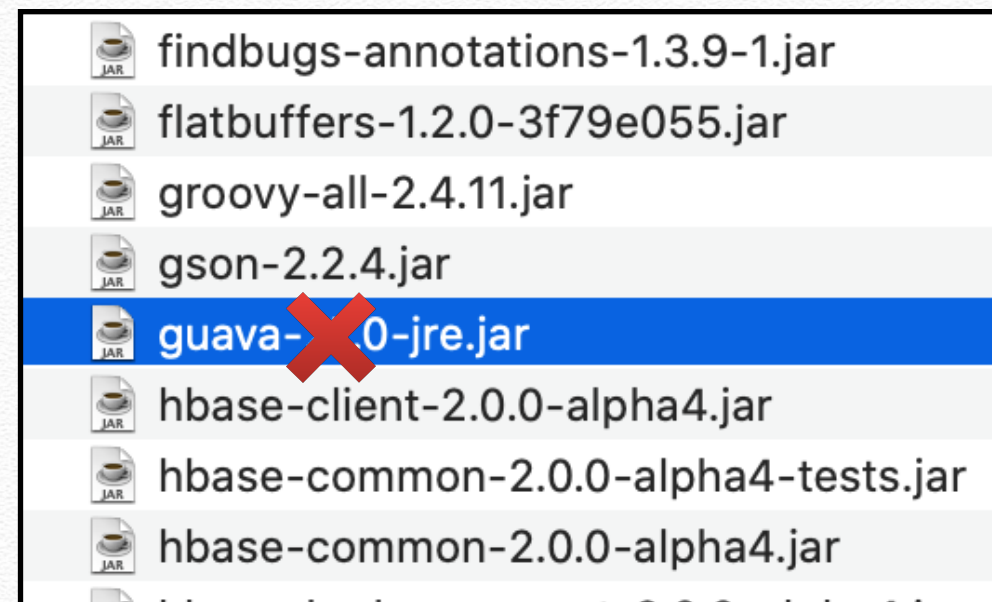
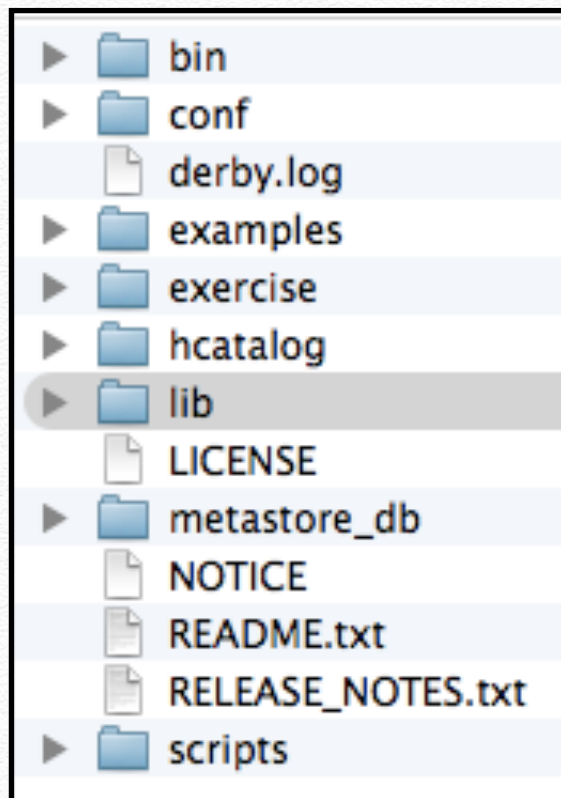
```
# Set HADOOP_HOME to point to a specific hadoop install directory
HADOOP_HOME=/Users/roberto/Documents/hadoop-3.2.1
# Hive Configuration Directory can be controlled by:
export HIVE_CONF_DIR=/Users/roberto/Documents/apache-hive-2.3.6-bin

export JAVA_HOME=$(/usr/libexec/java_home)
```




Hive Configuration

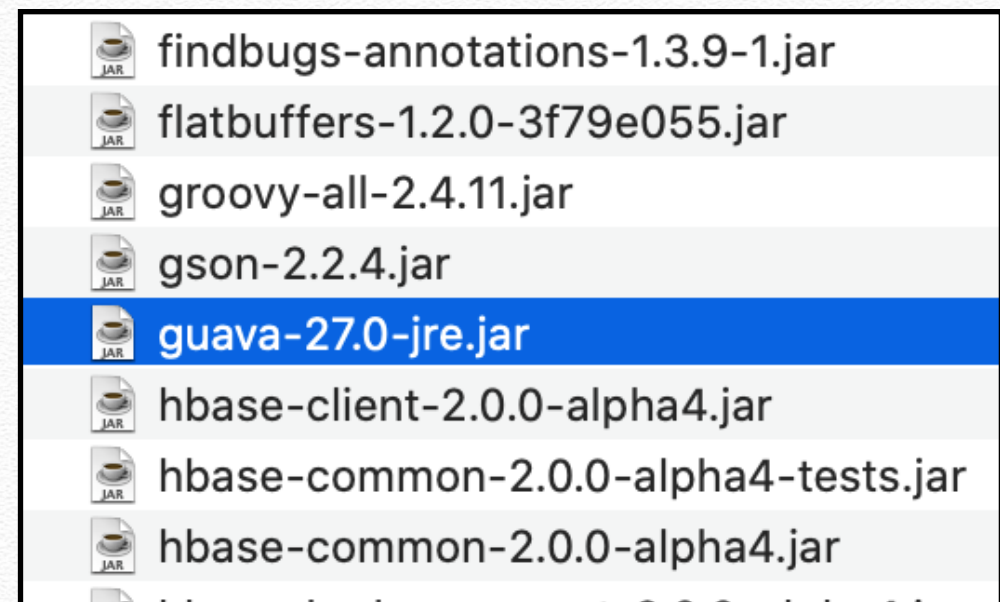
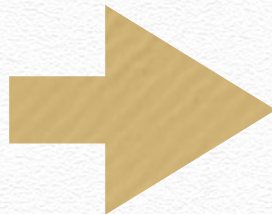
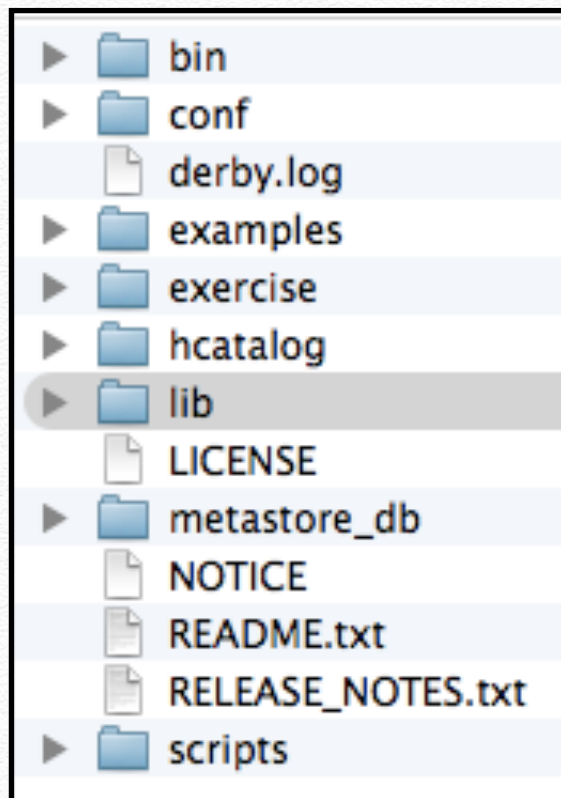
- ❖ In the **lib** directory of hive-home directory **remove** **guava-14.0.jar** file





Hive Configuration

- ❖ In the **lib** directory of hive-home directory **copy** **guava-27.0.jar** file from **share/hdfs/lib** of your hadoop-home directory





Hive Configuration

- ❖ Hive by default uses **Derby database**. Use the below command to initialize the Derby database.

```
$:~$HIVE_HOME/bin/schematool -dbType derby -initSchema
```




Hive Configuration

- ❖ Hive uses **Hadoop** `$:~ hadoop-*/sbin/start-dfs.sh`
- ❖ In addition, you must create **/tmp** and **/user/hive/warehouse** and set them **chmod g+w** in **HDFS** before you can create a table in Hive.
- ❖ Commands to perform this setup:

```
$:~$HADOOP_HOME/bin/hdfs dfs -mkdir /tmp
$:~$HADOOP_HOME/bin/hdfs dfs -mkdir /user/hive
$:~$HADOOP_HOME/bin/hdfs dfs -mkdir /user/hive/warehouse
$:~$HADOOP_HOME/bin/hdfs dfs -chmod g+w /tmp
$:~$HADOOP_HOME/bin/hdfs dfs -chmod g+w /user/hive/warehouse
```




Hive Console

- ❖ Usually, we run **Hive** by using a command line console; **Hive** supports two Hive clients: the **Hive CLI** and **Beeline**. The primary difference between the two involves how the clients connect to Hive.
- ☑ The **Hive CLI**, which connects directly to HDFS and the Hive Metastore, and can be used only on a host with access to those services.
- ☑ **Beeline**, which connects to *HiveServer2* and requires access to only one .jar file: *hive-jdbc-<version>-standalone.jar*.



Hive Console

- ☑ The **Hive CLI** is actually deprecated but yet working
- ☑ **Beeline** is recommended, that is using HiveServer2 and a JDBC client, as the primary way to access Hive. This approach uses SQL standard-based authorization or Ranger-based authorization.
- ☑ However, some users may wish to access Hive data from other applications, such as **Pig**. For these use cases, use the Hive CLI and storage-based authorization.
- ☑ Both clients allows **HiveQL** to query Hive but they have different approaches to start a working session.



Hive CLI Console

✓ The **Hive CLI** is called by the following command

```
$:~hive-*/bin/hive -h -p
```

✓ **-h** indicates the host

✓ **-p** indicates the TCP port

Hive Local Running with CLI



❖ Running Hive:

```
$:~hive-*/bin/hive <parameters>
```

❖ Try the following command to access to Hive shell:

```
$:~hive-*/bin/hive
```

❖ Hive Shell

```
Logging initialized using configuration in jar:file:/Users/  
mac/Documents/hive-0.11.0-bin/lib/hive-common-0.11.0.jar!/hive-log4j.properties
```

```
Hive history file=/tmp/mac/hive_job_log_mac_4426@MacBook-  
Air-di-mac.local_201404091440_1786371657.txt
```

```
hive>
```




Hive CLI Console

The **Hive CLI** can receive three types of commands (terminated by “;”)

- ✓ **HiveQL** statement to query Hive by using a **SQL**-Like language (see later).
- ✓ **shell-commands** preceded by an exclamation mark “!”

```
hive> !ls -la;

total 216

drwxr-xr-x 17 bigdata bigdata 4096 May 11 11:30 .
-rw-rw-r-- 1 bigdata bigdata  39 Apr 13 07:32 appunti.txt

...
```




Hive CLI Console

The **Hive CLI** can receive three types of commands (terminated by “;”)

- ✓ **HiveQL** statement to query Hive by using a SQL-Like language (see later).
- ✓ **shell-commands** preceded by an exclamation mark “!”
- ✓ **HDFS-commands** introduced by the command *dfs*

```
hive> dfs -ls / ;
```

```
Found 2 items
```

```
drwx-wx-wx    - hduser supergroup    0 2020-04-03 22:35 /tmp
```

```
drwxr-xr-x    - hduser supergroup    0 2020-04-10 13:37 /warehouse
```




Hive CLI Console

To close the **Hive CLI** you have to use the command *quit*

```
hive> quit;
```

```
$:~
```


Hive Running



❖ In the Hive Shell you can call any **HiveQL** statement:

▶ *create a table*

```
hive> CREATE TABLE pokes (foo INT, bar STRING);
```

```
OK
```

```
Time taken: 0.354 seconds
```

```
hive> CREATE TABLE invites (foo INT, bar STRING) PARTITIONED BY (ds STRING);
```

```
OK
```

```
Time taken: 0.051 seconds
```

▶ *browsing through Tables: lists all the tables*

```
hive> SHOW TABLES;
```

```
OK
```

```
invites
```

```
pokes
```

```
Time taken: 0.093 seconds, Fetched: 2 row(s)
```


Hive Running



- ***browsing through Tables:** lists all the tables that end with 's'.*

```
hive> SHOW TABLES `.*s`;  
  
OK  
  
invites  
  
pokes  
  
Time taken: 0.063 seconds, Fetched: 2 row(s)
```

- ***browsing through Tables:** shows the list of columns of a table.*

```
hive> DESCRIBE invites;  
  
OK  
  
foo                int                None  
bar                string             None  
ds                 string             None  
  
# Partition Information  
  
# col_name          data_type          comment  
ds                  string             None  
  
Time taken: 0.191 seconds, Fetched: 8 row(s)
```


Hive Running



► *altering tables*

```
hive> ALTER TABLE events RENAME TO 3koobecaf;  
  
hive> ALTER TABLE pokes ADD COLUMNS (new_col INT);  
  
hive> ALTER TABLE invites ADD COLUMNS (new_col2 INT COMMENT 'a comment');  
  
hive> ALTER TABLE invites REPLACE COLUMNS (foo INT, bar STRING, baz INT  
                                             COMMENT 'baz replaces new_col2');
```

► *dropping Tables*

```
hive> DROP TABLE pokes;
```


Hive Running



► *DML operations*

takes file from local file system

```
hive> LOAD DATA LOCAL INPATH './examples/files/kv1.txt'  
      OVERWRITE INTO TABLE pokes;
```

takes file from HDFS file system

```
hive> LOAD DATA INPATH '/user/hive/files/kv1.txt'  
      OVERWRITE INTO TABLE pokes;
```

► *SQL query*

```
hive> SELECT * FROM pokes;
```


Hive Running



- ❖ Running Hive “**One Shot**” command:

```
$:~hive-*/bin/hive -e <command>
```

- ❖ For instance:

```
$:~hive-*/bin/hive -e "SELECT * FROM mytable LIMIT 3"
```

Result

```
OK
```

```
name1 10
```

```
name2 20
```

```
name3 30
```


Hive Running



- ❖ Executing Hive queries from **file**:

```
$:~hive-*/bin/hive -f <file>
```

- ❖ For instance:

```
$:~hive-*/bin/hive -f query.hql
```

query.hql

```
SELECT *
```

```
FROM mytable
```

```
LIMIT 3
```


Hive Running



- ❖ Executing Hive queries from **file** inside the **Hive Shell**

```
$:~ cat /path/to/file/query.hql
```

```
SELECT * FROM mytable LIMIT 3
```

```
$:~hive-*/bin/hive
```

```
hive> SOURCE /path/to/file/query.hql;
```

```
...
```


HiveQL - Select-Where



❖ SYNTAX

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...  
  
FROM table_reference  
  
[WHERE where_condition]  
  
[GROUP BY col_list]  
  
[HAVING having_condition]  
  
[CLUSTER BY col_list | [DISTRIBUTE BY col_list] [SORT BY col_list]]  
  
[LIMIT number];
```


Hive in Local: Examples



Word Count using Hive



❖ Count words in a text file

words_1.txt

sopra la panca la capra campa, sotto la panca la capra crepa

Hive in Local: Examples



Word Count using Hive



wordcounts.hql

```
CREATE TABLE docs (line STRING);

LOAD DATA LOCAL INPATH './data/words_1.txt'

                                OVERWRITE INTO TABLE docs;

CREATE TABLE word_counts AS

SELECT w.word, count(1) AS count FROM

    (SELECT explode(split(line, ' ')) AS word FROM docs) w

GROUP BY w.word

ORDER BY w.word;
```

words.txt

```
program
program
pig
pig
program
pig
hadoop
pig
latin
latin
```





Hive in Local: Examples





Word Count using Hive

```
$:~hive-*/bin/hive -f wordcounts.hql
```

Browse Directory

/user/hive/warehouse Go!   


Show 25 entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxrwxr-x	roberto	supergroup	0 B	Apr 14 14:10	0	0 B	docs	
<input type="checkbox"/>	drwxrwxr-x	roberto	supergroup	0 B	Apr 14 14:11	0	0 B	word_counts	

Showing 1 to 2 of 2 entries

Previous 1 Next

Size Name

000000_0 

Previous 1 Next

File contents

campa, 1
capra 2
crepa 1
la 4
panca 2
sopra 1
sotto 1

Hive in Local: Examples



Word Count using Hive



- ❖ **Count words** in a **text file** separated by **lines** and **spaces**

words.txt

```
program
program
pig
pig
program
pig
hadoop
pig
latin
latin
```


Hive in Local: Examples



Word Count using Hive



wordcounts.hql

```
CREATE TABLE docs (line STRING);

LOAD DATA LOCAL INPATH './exercise/data/words.txt'

                                OVERWRITE INTO TABLE docs;

CREATE TABLE word_counts AS

SELECT w.word, count(1) AS count FROM

    (SELECT explode(split(line, '\s')) AS word FROM docs) w

GROUP BY w.word

ORDER BY w.word;
```

words.txt

```
program
program
pig
pig
program
pig
hadoop
pig
latin
latin
```





Hive in Local: Examples





Word Count using Hive

```
$:~hive-*/bin/hive -f wordcounts.hql
```

Browse Directory

/user/hive/warehouse Go!   


Show 25 entries Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	drwxrwxr-x	roberto	supergroup	0 B	Apr 14 14:10	0	0 B	docs	
<input type="checkbox"/>	drwxrwxr-x	roberto	supergroup	0 B	Apr 14 14:11	0	0 B	word_counts	

Showing 1 to 2 of 2 entries

Previous 1 Next

Size Name

000000_0 

Previous 1 Next

File contents

Hadoop 1
Latin 2
Pig 4
Program 3

Hive in Local: Examples



Computing the number of log entries for each user

- ❖ Logs of user querying the web consists of (**user,time,query**)
- ❖ Fields of the log are tab separated and in text format



excite-small.log

user	time	query
2A9EABFB35F5B954	970916105432	foods
BED75271605EBD0C	970916001949	yahoo chat
BED75271605EBD0C	970916001954	yahoo chat
BED75271605EBD0C	970916003523	yahoo
824F413FA37520BF	970916184809	spider
824F413FA37520BF	970916184818	calg

result

user	#log entries
2A9EABFB35F5B954	1
BED75271605EBD0C	3
824F413FA37520BF	2

Hive in Local: Examples



Computing the number of log entries for each user



logs.hql

```
CREATE TABLE logs (user STRING, time STRING, query STRING);

LOAD DATA LOCAL INPATH './exercise/data/excite-small.log'

OVERWRITE INTO TABLE logs;

CREATE TABLE result AS

SELECT user, count(1) AS log_entries

FROM logs

GROUP BY user

ORDER BY user;
```


Hive in Local: Examples



Computing the average number of page visits by user



- ❖ Logs of user visiting a webpage consists of (**name,url,time**)
- ❖ Fields of the log are tab separated and in text format

- ❖ **Basic idea:**

- ★ **Load** the log file
- ★ **Group** based on the **user** field
- ★ **Count** the group
- ★ **Calculate** average for all users
- ★ **Visualize** result

visits.log

Name	url	time
Amy	www.cnn.com	8:00
Amy	www.crap.com	8:05
Amy	www.myblog.com	10:00
Amy	www.flickr.com	10:05
Fred	cnn.com/index.htm	12:00
Fred	cnn.com/index.htm	13:00

Hive in Local: Examples



Computing the average number of page visits by user

avg_visits.hql

```
CREATE TABLE pages (name STRING, url STRING, time STRING)
```

```
    ROW FORMAT DELIMITED
```

```
    FIELDS TERMINATED BY ' ';
```

```
LOAD DATA LOCAL INPATH './data/visits.log'
```

```
    OVERWRITE INTO TABLE pages;
```

```
CREATE TABLE avg_visit AS
```

```
SELECT AVG(num_pages) FROM
```

```
    (SELECT name, count(1) AS num_pages
```

```
    FROM pages
```

```
    GROUP BY name) np;
```



HiveQL - Select-Joins



❖ SYNTAX

`join_table:`

`table_reference JOIN table_factor [join_condition]`

`| table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference`

`join_condition`

`| table_reference LEFT SEMI JOIN table_reference join_condition`

`| table_reference CROSS JOIN table_reference [join_condition]`

HiveQL - Select-Joins



❖ There are different types of joins given as follows:

☒ *JOIN*

☒ *LEFT OUTER JOIN*

☒ *RIGHT OUTER JOIN*

☒ *FULL OUTER JOIN*

Hive in Local: Examples



Identify users who visit “Good Pages” on average

visits.log

name	url	time
Amy	www.cnn.com	8:00
Amy	www.crap.com	8:05
Amy	www.myblog.com	10:00
Amy	www.flickr.com	10:05
Fred	cnn.com/index.htm	12:00
Fred	cnn.com/index.htm	13:00

pages.log

url	pagerank
www.cnn.com	0.9
www.flickr.com	0.9
www.myblog.com	0.7
www.crap.com	0.2
cnn.com/index.htm	0.1

- ❖ **Good pages** are those pages visited by users whose **page rank** is greater than **0.5**
- ❖ **Basic idea:**
 - ★ **Join** table based on **url**
 - ★ **Group** based on **user**
 - ★ **Calculate** average **page rank** of user visited pages
 - ★ **Filter** user who has average page rank greater than **0.5**
 - ★ **Store** the result



Hive in Local: Examples



Computing the average number of page visits by user

rank.hql



```
set hive.auto.convert.join = false;

CREATE TABLE visits (name STRING, url STRING, time STRING);

LOAD DATA LOCAL INPATH './data/visits.log'

OVERWRITE INTO TABLE visits;

CREATE TABLE pages (url STRING, pagerank DECIMAL);

LOAD DATA LOCAL INPATH './data/pages.log'

OVERWRITE INTO TABLE pages;
```


Hive in Local: Examples



Computing the average number of page visits by user

rank.hql

```
CREATE TABLE rank_result AS

SELECT pr.name FROM

    (SELECT V.name, AVG(P.pagerank) AS prank

    FROM visits V JOIN pages P ON (V.url = P.url)

    GROUP BY name) pr

WHERE pr.prank > 0.5;
```



Hive in Local: Examples



ID, Name of ALL customers and amount of corresponding orders

customers.txt

Id	Name	Age	Address	Salary
1	Ramesh	32	Rome	2000
2	Bob	25	Milan	1500
3	Mary	23	Paris	2000
4	Mark	25	Rome	6500
5	John	27	Milan	8500
6	Carl	29	Milan	3400

Orders.txt

Id	c_id	Amount
101	3	3000
102	3	4000
103	2	8000
104	1	10000
105	5	1300

❖ We have to include also customers that have not orders

❖ Basic idea:

★ **Left outer join** table based on **id** and **c_id**



Hive in Local: Examples



Computing the average number of page visits by user

no_order.hql

```
set hive.auto.convert.join = false;
```

```
CREATE TABLE customers (id int, name STRING, age int, address STRING,  
salary int);
```

```
LOAD DATA LOCAL INPATH './data/customers.txt'
```

```
OVERWRITE INTO TABLE customers;
```

```
CREATE TABLE orders (id int, c_id int, amount int);
```

```
LOAD DATA LOCAL INPATH './data/orders.txt'
```

```
OVERWRITE INTO TABLE orders;
```



Hive in Local: Examples



Computing the average number of page visits by user

`no_order.hql`



File contents

```
1 Ramesh 10000
2 Bob 8000
3 Mary 4000
3 Mary 3000
4 Mark \N
5 John 1300
6 Carl \N
```


Hive in Local: Examples



Computing the average number of page visits by user

no_order.hql



```
CREATE TABLE no_order_result AS  
  
SELECT c.ID, c.NAME, o.AMOUNT  
  
FROM CUSTOMERS c LEFT OUTER JOIN ORDERS o  
  
ON (c.ID = o.C_ID) ;
```


Hive in Local with **User Defined Functions**: Examples



Convert unixtime to a regular time date format

subscribers.txt

name, department, email, time

Frank Black, 1001, frankdaman@eng.example.com, -72710640

Jolie Guerms, 1006, jguerms@ga.example.com, 1262365200

Mossad Ali, 1001, mali@eng.example.com, 1207818032

Chaka Kaan, 1006, ckhan@ga.example.com, 1130758322

Verner von Kraus, 1007, verner@example.com, 1341646585

Lester Dooley, 1001, ldooley@eng.example.com, 1300109650



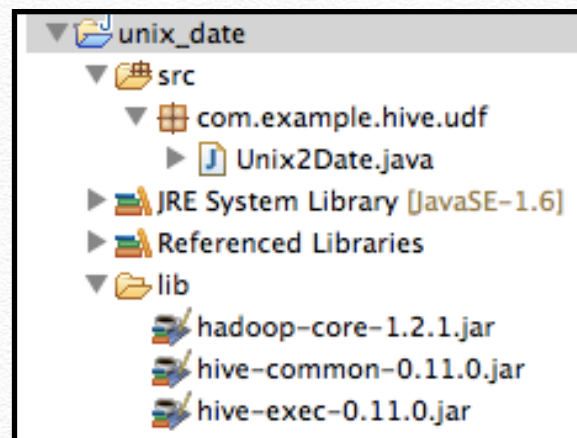
❖ Basic idea:

★ **Define** a User Defined Function (**UDF**)

★ **Convert** time field using **UDF**

Hive in Local with User Defined Functions: Examples

Convert unixtime to a regular time date format



```
package com.example.hive.udf;
```

```
import java.util.Date;
import java.util.TimeZone;
import java.text.SimpleDateFormat;
import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

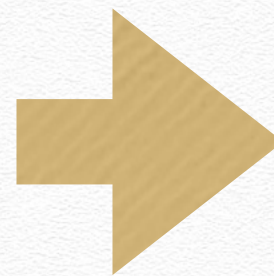
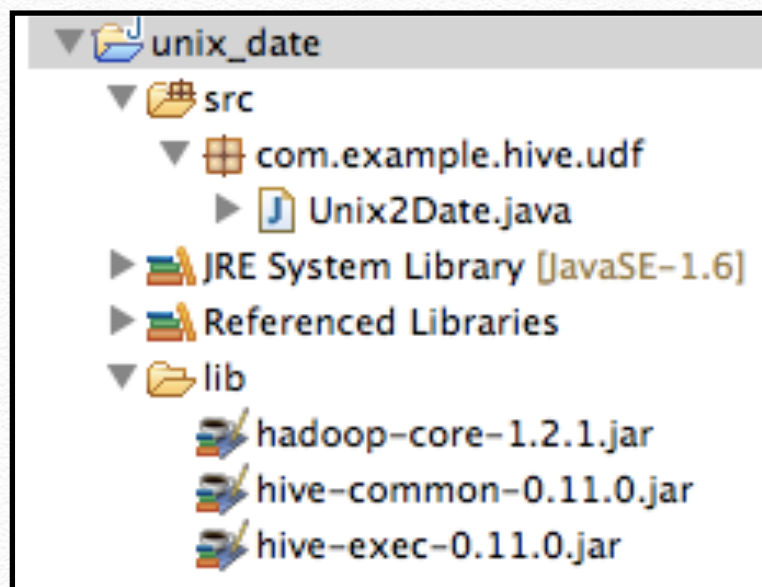
public class Unix2Date extends UDF{
    public Text evaluate(Text text) {
        if(text == null) return null;
        long timestamp = Long.parseLong(text.toString());
        // timestamp*1000 is to convert seconds to milliseconds
        Date date = new Date(timestamp*1000L);
        // the format of your date
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss z");
        sdf.setTimeZone(TimeZone.getTimeZone("GMT+2"));
        String formattedDate = sdf.format(date);
        return new Text(formattedDate);
    }
}
```

Unix2Date.java



Hive in Local with **User Defined Functions**: Examples

Convert unixtime to a regular time date format



unix_date.jar



Hive in Local with User Defined Functions: Examples

Convert unixtime to a regular time date format



```
$:~hive-*/bin/hive -f time_conversion.hql
```

time_conversion.hql



```
CREATE TABLE IF NOT EXISTS subscriber (  
  
    username STRING,  
  
    dept STRING,  
  
    email STRING,  
  
    provisioned STRING)  
  
    ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';  
  
LOAD DATA LOCAL INPATH './exercise/data/subscribers.txt' INTO TABLE subscriber;  
  
add jar ./exercise/jar_files/unix_date.jar;  
  
CREATE TEMPORARY FUNCTION unix_date AS 'com.example.hive.udf.Unix2Date';  
  
SELECT username, unix_date(provisioned) FROM subscriber;
```


Hive in Local with User Defined Functions: Examples

Convert unixtime to a regular time date format



```
$:~hive-*/bin/hive -f time_conversion.hql
```



```
Frank Black 12-09-1967 12:36:00 GMT+02:00  
Jolie Guerms 01-01-2010 19:00:00 GMT+02:00  
Mossad Ali 10-04-2008 11:00:32 GMT+02:00  
Chaka Kaan 31-10-2005 13:32:02 GMT+02:00  
Verner von Kraus 07-07-2012 09:36:25 GMT+02:00  
Lester Dooley 14-03-2011 15:34:10 GMT+02:00  
Time taken: 9.12 seconds, Fetched: 6 row(s)
```


Hive in Local: Examples



Aggregate items by category and calculate counting and average

tbl

item	category	price
banana	fruit	1
apple	fruit	1.2
desk	furniture	420
TV	electronics	500
iPad	electronics	120



result

category	cnt	avg_price
fruit	2	1.1
furniture	1	420
electronics	2	310

Hive in Local: Examples



Aggregate items by category and calculate counting and average

```
SELECT
    category,
    COUNT(1) AS cnt,
    AVG(price) AS avg_price
FROM tbl
GROUP BY category
```


Hive in Local: Examples



The average number of page views per user in each day

access

time	user_id
1416435585	2
1416435586	3
1416435587	5
1416435588	1
...	...

```
TD_TIME_FORMAT(  
    time,  
    'yyyy-MM-dd',  
    'PST' )
```



result

day	average_pv
2014-10-01	6.21
2014-10-02	1.21
2014-10-03	7.1
...	...

Hive in Local: Examples



The average number of page views per user

```
SELECT
```

```
    TD_TIME_FORMAT(time, 'yyyy-MM-dd', 'PST') AS day,  
    COUNT(1) / COUNT(DISTINCT(user_id)) AS average_pv
```

```
FROM access
```

```
GROUP BY TD_TIME_FORMAT(time, 'yyyy-MM-dd', 'PST')
```

```
ORDER BY day ASC
```


Hive in Local: Examples



Market Basket Analysis:

How many people bought both Product A and Product B?

purchase

receipt_id	item_id
1	31231
1	13212
2	312
3	2313
3	9833
4	1232
...	...



result

item1	item2	cnt
583266	621056	3274
621056	583266	3274
31231	13212	2077
...

Hive in Local: Examples



Market Basket Analysis:

How many people bought both Product A and Product B?

```
SELECT
  t1.item_id AS item1,
  t2.item_id AS item2,
  COUNT(1) AS cnt
FROM
  (
    SELECT DISTINCT receipt_id, item_id
    FROM purchase
  ) t1
JOIN
  (
    SELECT DISTINCT receipt_id, item_id
    FROM purchase
  ) t2
ON (t1.receipt_id = t2.receipt_id)
GROUP BY t1.item_id, t2.item_id
HAVING t1.item_id != t2.item_id
ORDER BY cnt DESC
```


Hive in Local: Examples



First Paid Users:

The number of people who first paid the money to the product

pay

time	user_id	price
1416435585	2	100
1416435586	3	500
1416435587	5	100
1416435588	1	100
...	...	

`TD_TIME_FORMAT(
time,
'yyyy-MM-dd',
'PST')`



result

day	first_paid_users
2014-10-01	321
2014-10-02	364
2014-10-03	343
...	...

Hive in Local: Examples



First Paid Users:

The number of people who first paid the money to the product

```
SELECT
  t1.day, COUNT(1) AS first_paid_users
FROM
  (SELECT user_id,
    TD_TIME_FORMAT(MIN(time), 'yyyy-MM-dd', 'PST') AS day
  FROM pay
  GROUP BY user_id
  ) t1
GROUP BY t1.day
ORDER BY t1.day ASC
```


Hive in Local: Examples



Gold Inflow / Outflow:

In each day how many losses and how many winnings

winners

time	user_id	value
1416435585	2	300
1416435586	3	500
1416435587	5	1000
...	...	

losers

time	user_id	value
1416435585	2	100
1416435586	3	900
1416435587	5	3000
...	...	

`TD_TIME_FORMAT(
time,
'yyyy-MM-dd',
'PST')`



result

day	user_win	user_lose
2014-10-01	3010	2010
2014-10-02	2030	4200
2014-10-03	1000	10200
...	...	

Hive in Local: Examples

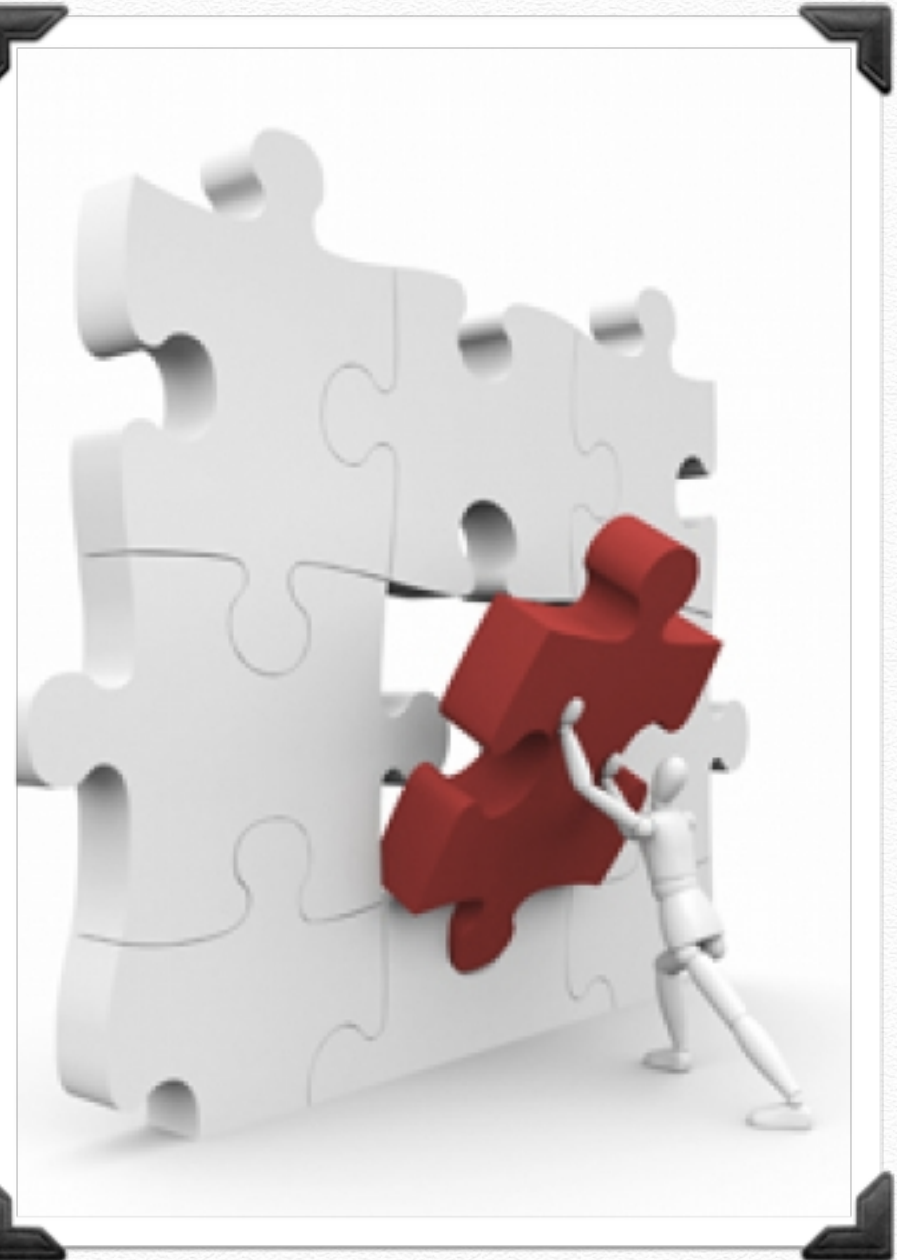


Gold Inflow / Outflow:

In each day how many losses and how many winnings

```
SELECT
  t1.d AS day,
  user_win,
  user_loss
FROM
  (SELECT
    TD_TIME_FORMAT(time, 'yyyy-MM-dd', 'PDT') AS d,
    SUM(value) AS user_win,
  FROM winners
  GROUP BY
    TD_TIME_FORMAT(time, 'yyyy-MM-dd', 'PDT')
  ) t1
JOIN
  (SELECT
    TD_TIME_FORMAT(time, 'yyyy-MM-dd', 'PDT') AS d,
    SUM(value) AS user_lose,
  FROM losers
  GROUP BY
    TD_TIME_FORMAT(time, 'yyyy-MM-dd', 'PDT')
  ) t2
ON (t1.d=t2.d)
ORDER BY d ASC
```


Quickly Wrap Up





What Hive allows

- ❖ Hive **generates map-reduce jobs** from a **query** written in **higher level language**.
- ❖ Hive **frees users** from **knowing** all the **little secrets** of **Map-Reduce & HDFS**.

Language



❖ **HiveQL:** Declarative SQLish language

- `SELECT * FROM 'mytable' ;`

language = user



❖ **Hive:** More popular among

- analysts

users = usage pattern



❖ **Hive:**

- `analysts:` Generating daily reports

Usage pattern



Data Collection



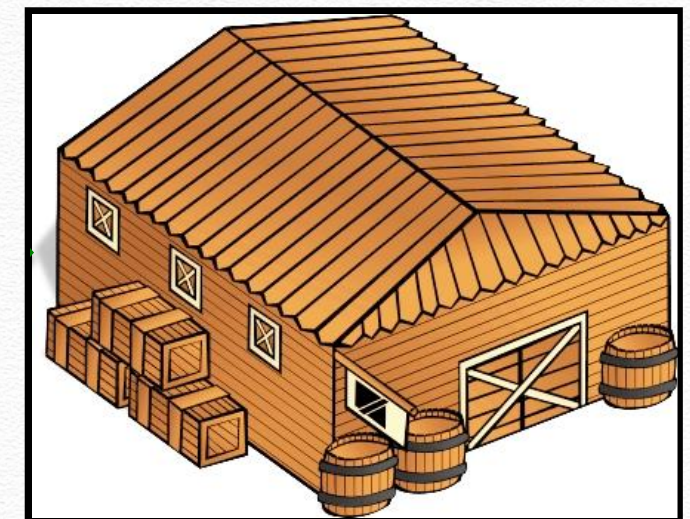
Data Factory

Scripting/Programming

-Pipeline

-Iterative Processing

-Research



Data Warehouse

Hive

-BI tools

-Analysis

usage pattern = future directions



- ❖ **Hive is evolving towards Data-warehousing solution**
 - **Users** are asking for better **integration** with other systems (**O/JDBC**)



Apache Hive

Big Data - 15/04/2020