



# Apache Hive

<https://hive.apache.org/>

- Le projet mené par des bénévoles de l'Apache Software Foundation
- Logiciel de Data Warehouse
- Il permet d'effectuer facilement et rapidement des requêtes « SQL-like » (HiveQL) pour extraire efficacement des données enregistrées sous HDFS de **Apache Hadoop**

1

## Avantages de HIVE

- Il peut être utilisé comme un ETL (Extract Transform Load), processus d'extraction des données des systèmes sources et de leur transfert dans l'entrepôt de données
- Offre les possibilités de requêtes et d'analyse
- Peut manipuler des ensembles de données larges
- SQL (filters, joins, group by)

3

## Apache HIVE

L'utilisateur envoie ses requêtes Hive. Ces dernières seront converties en des tâches MapReduce.



2

## Quand HIVE ne doit pas être utilisé?

- Quand la base de données est petite (utiliser dans ce cas les outils sql traditionnels)
- S'il n'est pas possible d'avoir un schéma pour la base de données
- Si l'application nécessite un temps de réponse rapide
- Si les bases de données relationnelles peuvent résoudre le problème, ne pas utiliser HIVE

4

## Autres écosystèmes utilisant SQL

Hive + Mapreduce----->This is the first avenger-->Damn slow  
Hive + Tez----->Hortonworks----->Interactive  
Impala----->Cloudera----->Interactive  
SparkSQL----->Spark----->Almost real time

Pig + Map Reduce

Phoenix - SQL interface on top of HBASE

5

## Différentes versions de HIVE

**HIVE:** version originale de HIVE, c'est juste un client qui permet d'interroger une base de données avec un shell interactif

**HiveServer 1:** permet des connexions JDBC/ODBC à partir de clients SQL réguliers. Mais il n'y a pas de concurrence. Il n'est pas possible d'avoir plusieurs sessions de HIVE, plusieurs connexions utilisateurs

**HiveServer 2+ Beeline:** permet des connexions JDBC/ODBC. Permet la concurrence (c'est possible d'avoir plusieurs connexions JDBC/ODBC) et a un nouveau outil ligne de commande client appelé Beeline. CLI Beeline peut être installé comme un client SQL, à côté du CLI initial. Le serveur HIVE peut être installé à l'intérieur ou à l'extérieur du cluster.

6

## Métadata- Service metastore

**Métastore intégré:** par défaut HIVE utilise un métastore intégré Apache derby. Il enregistre les métadonnées dans la base de données Apache derby. Derby n'autorise qu'un seul utilisateur pour y accéder.

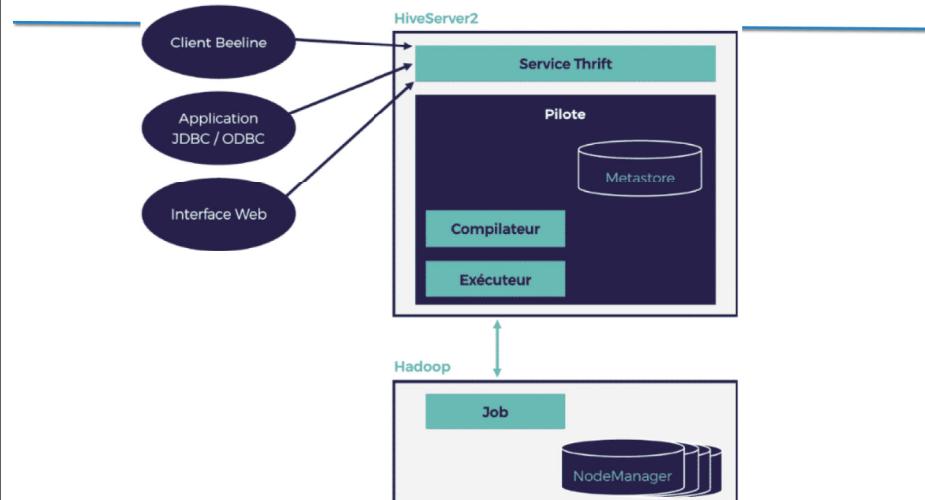
Il est possible de configurer une base de données personnelle comme stockage metadata pour être utilisée par HIVE. Exemple, MySQL.

Le stockage est distribué, qui est HDFS.

Les dernières versions de HIVE permettent les modifications de la base de données.

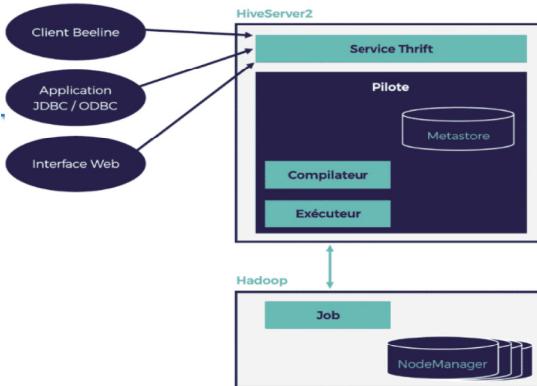
7

## Architecture de HIVE



<https://meritis.fr/big-data-analyse-donnees-apache-hive/>

8

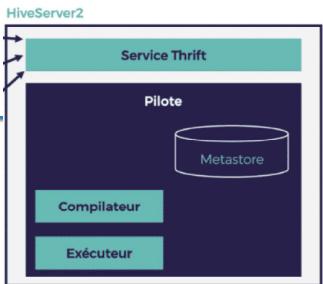


## Architecture de HIVE

L'interaction Hive/Hadoop s'effectue selon les **3** étapes suivantes:

- 1. Envoi de la requête HiveQL :** en utilisant un client Hive (le client shell, un client JDBC (Java Database Connectivity) /ODBC (Open Database Connectivity) ou une interface web), la requête est envoyée au serveur Hive,
- 2. Planification de la requête :** la requête est reçue par le driver (pilote). Elle est compilée, optimisée et planifiée comme un job,
- 3. Exécution du job :** le job est exécuté sur le cluster Hadoop.

9



## Architecture de HIVE

### Partie (2) : la partie serveur

- **HiveServer2** succède à HiveServer (qui est devenu deprecated à partir de la version 1.0.0). Il s'agit du **conteneur du moteur d'exécution de Hive** et appelé couramment **pilote** (ou driver). Il se compose du **metastore**, du **compilateur** et de **l'exécuteur**.
- **HiveServer2** assure deux nouvelles fonctionnalités : la gestion de l'authentification client et la gestion des requêtes concurrentes.

11



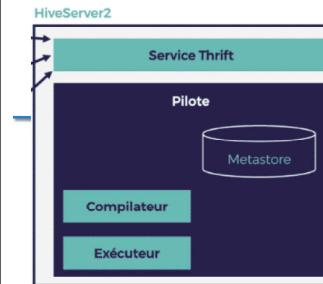
## Architecture de HIVE

### Partie (1) : la partie client

Il est possible de **soumettre des requêtes au serveur Hive de différentes manières**. En utilisant :

- Le **client Hive CLI** (Hive Command Line Interface) qui permet d'entrer des commandes directement depuis le shell hive ou d'exécuter un ensemble de commandes Hive écrites dans un fichier texte. Ce client n'est pas compatible avec la nouvelle version de Hive (HiveServer2) et a été remplacé par **Beeline** qui est le nouveau client en mode ligne de commande de Hive. Il communique avec HiveServer2 via thrift,
- Le **client JDBC/ODBC**,
- Le **client web**.

10



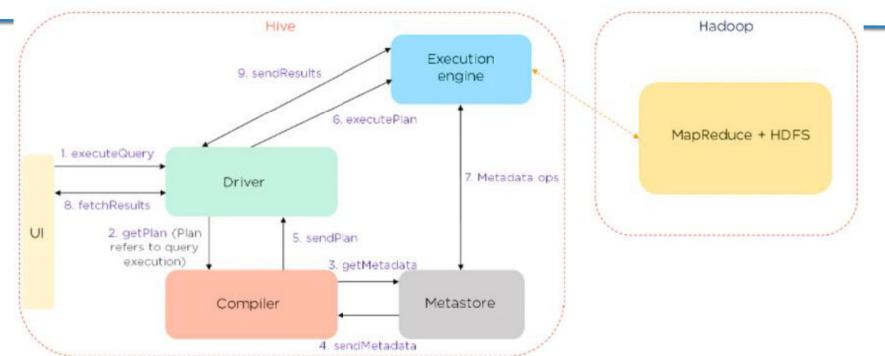
## Architecture de HIVE

### Partie (2) : la partie serveur

Pour chaque connexion client, HiveServer2 crée un nouveau contexte d'exécution (connexion + session). La nouvelle interface RPC de HiveServer2 permet au serveur d'associer le contexte d'exécution Hive avec le thread qui sert la requête client. Cette interface implémente un service thrift pour communiquer avec les clients et exécuter leurs requêtes.

12

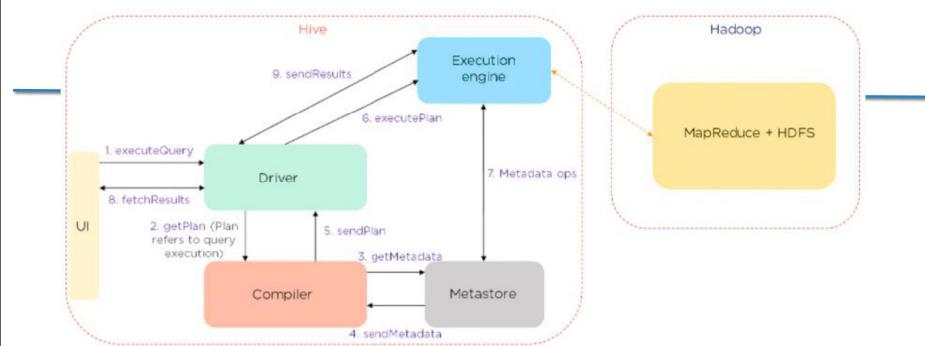
## Les flux de données dans HIVE



Le flux de données dans les séquences suivantes:

1. Le client exécute une requête, qui va au driver
2. Le driver demande le plan d'exécution de la requête
3. Le compilateur demande les métadonnées depuis metastore
4. Le compilateur reçoit les métadonnées depuis metastore

13



5. Le compilateur rassemble ses informations et envoie le plan au driver
6. Le driver envoie le plan d'exécution au Execution engine
7. Le moteur d'exécution exécute comme un pont entre Hive et Hadoop pour exécuter la requête.
8. Le moteur d'exécution communique avec metastore pour effectuer des différentes opérations, comme créer et effacer les tables
9. Afficher les résultats au client

14

## Modélisation de données dans HIVE

Les données de Hive sont organisées dans des:

**Bases de données:** espace de nom qui sépare les tables et les autres unités de données

**Tables:** Les tables dans Hive sont créées de la même manière que dans RDBMS

**Partitions :** Ici, les tables sont organisées en partitions pour regrouper des types de données similaires en fonction de la clé de partition

**Buckets (clusters):** Les données présentes dans les partitions peuvent être divisées en compartiments pour une interrogation efficace.

15

## Les types de données dans HIVE

### Type primitive

### Type numérique

- **TINYINT** (1 bytes signed integer)
- **SMALLINT** (2 bytes signed integer)
- **INT/INTEGER** (4 bytes signed integer)
- **BIGINT** (8 bytes signed integer)
- **FLOAT** (4 bytes single precision floating point number)
- **DOUBLE** (8 bytes double precision floating point number)
- **DOUBLE PRECISION** (alias for DOUBLE, only available starting with Hive [2.2.0](#))
- **DECIMAL**
  - Introduced in Hive [0.11.0](#) with a precision of 38 digits
  - Hive [0.13.0](#) introduced user-definable precision and scale
- **NUMERIC** (same as DECIMAL, starting with [Hive 3.0.0](#))

<https://cwiki.apache.org/confluence/display/hive/languagemanual+types>

16

## Type primitive

### Date/Time Types

- [TIMESTAMP](#) (Note: Only available starting with Hive [0.8.0](#))
- [DATE](#) (Note: Only available starting with Hive [0.12.0](#))
- [INTERVAL](#) (Note: Only available starting with Hive [1.2.0](#))

### String Types

- [STRING](#)
- [VARCHAR](#) (Note: Only available starting with Hive [0.12.0](#))
- [CHAR](#) (Note: Only available starting with Hive [0.13.0](#))

### Types de données divers

- BOOLEAN
- [BINARY](#) (Note: Only available starting with Hive [0.8.0](#))

17

## Types complexes

- arrays: ARRAY<data\_type> (Note: negative values and non-constant expressions are allowed as of [Hive 0.14.](#))
- maps: MAP<primitive\_type, data\_type> (Note: negative values and non-constant expressions are allowed as of [Hive 0.14.](#))
- structs: STRUCT<col\_name : data\_type [COMMENT col\_comment], ...>
- union: UNIONTYPE<data\_type, data\_type, ...> (Note: Only available starting with Hive [0.7.0.](#))

## Types Column

### Integral Types (TINYINT, SMALLINT, INT/INTEGER, BIGINT)

Par défaut le littéral Integral est INT, sauf si le nombre dépasse l'intervalle de INT auquel cas il est interprété comme BIGINT, ou un des postfixes suivants s'il est présent dans le nombre.

18

Type	Postfixe	Exemple
TINYINT	Y	100Y
SMALLINT	S	100S
BIGINT	L	100L

## Fonctions prédéfinies dans HIVE

### Fonctions mathématiques

round(16.39)=16	ceil(19.37)=20
round(16.39,1)=16.4	pow(2,3)=8
round(4.5)=5	abs(-9)=9
round(-4.5)=-5	sqrt(4)=2
floor(19.37)=19	

19

### Fonctions pour String

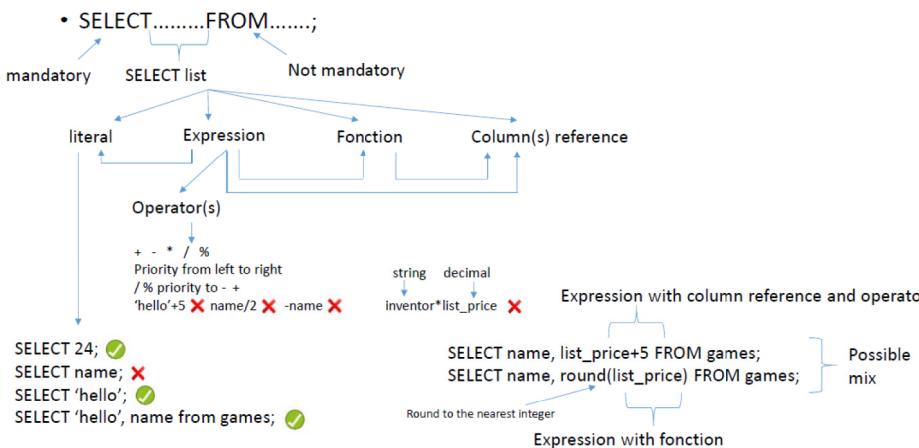
length(str)  
 reverse(str)  
 upper(str)  
 concat(str1,str2)  
 concat\_ws(sep,str1,str2,...) etc...

### Fonctions d'agrégation

max(col reference)  
 min(col reference)  
 count(\*)

20

## SELECT



21

## SELECT

La table toys a les colonnes: id (integer), name (string), price (decimal), and maker\_id (integer). La quelles des instructions select sont valides?

- ✓ SELECT name, price FROM toys;
- ✗ SELECT toys;
- ✓ SELECT 1000;
- ✗ SELECT FROM toys: name, price;
- ✗ SELECT FROM toys COLUMNS name, price;
- ✓ SELECT 'Lite-Brite';
- ✗ SELECT toys.name, toys.price;
- ✓ SELECT \* FROM toys;

23

## SELECT

La table toys a les colonnes: id (integer), name (string), price (decimal), and maker\_id (integer). La quelles des instructions liste un littéral string?

- ✗ SELECT Lite-Brite FROM toys
- ✓ SELECT 'toys'
- ✗ SELECT \* FROM toys
- ✓ SELECT 'name' FROM toys
- ✗ SELECT toys
- ✗ SELECT name FROM toys

21	Lite-Brite	14.47	105
22	Mr. Potato Head	11.50	105
23	Etch A Sketch	29.99	106

22

## SELECT

L'expression suivante cause une erreur lorsqu'elle est utilisée dans une expression SELECT: "-7.5" % 3.2

Quelle est l'erreur?

- ✓ Les types de données sont incompatibles.
- ✓ Les opérations indiquées sont applicables avec les entiers.
- ✗ Les opérations indiquées sont applicables avec les nombres négatifs.
- ✗ Un opérateur invalide est utilisé.

24

## SELECT

Quels sont les résultats des instructions SELECT suivantes?

SELECT round(3.47);

3

SELECT ceil(-2.47);

-2

25

## SELECT

Quelle est l'instruction SELECT qui est équivalente à  $3 * 3 * 3 * 3$  dans Hive?

✗ SELECT 3^4;

✓ SELECT pow(3,4);

✗ SELECT abs(3,4);

✗ SELECT round(3,4);

26

## SELECT

La base de données courante est par défaut, la table card\_rank existe dans la base de données fun. Quelles instructions permettent de lister les données de la table card\_rank?

✓ use fun; SELECT \* FROM card\_rank;

✗ SELECT card\_rank FROM fun;

✗ SELECT card\_rank.\* FROM fun;

✗ use card\_rank; SELECT \* FROM fun;

✓ SELECT \* FROM fun.card\_rank;

✗ SELECT \* FROM card\_rank;

Ici on choisit les instructions qui sont syntaxiquement correctes et permettent d'utiliser la table card\_rank qui se trouve dans la DB fun

27

## SELECT

### Alias

- Soit la requête suivante :

Select name, 5, list\_price+5 from games;

- il est possible de donner un nom de colonne (alias) résultats en utilisant as :

Select name, 5 as shipping\_fee, list\_price+5 as price\_with\_shipping from games;

- Alias peut contenir des lettres, chiffres, tirets bas (\_)

- Alias ne peut pas être uniquement des chiffres ou des mots réservés

28

## SELECT

Soit un extrait de la table games.csv

1 Monopoly	Elizabeth Magie	1903	8	2	6	19.99
2 Scrabble	Alfred Mosher Butts	1938	8	2	4	17.99
3 Clue	Anthony E. Pratt	1944	8	2	6	9.99
4 Candy Land	Eleanor Abbott	1948	3	2	4	7.99
5 Risk	Albert Lamorisse	1957	10	2	5	29.99

## cast

Select concat(name, ' is for', min\_age, ' or older' from games);

Hive fait le cast implicitement, Il est possible de faire le cast de min\_age comme suit:

cast(min\_age as string)

## distinct

Select distinct min\_age from games;

min_age
8
3
10

Les résultats sont distincts deux à deux (pas de répétitions)

29

## SELECT

Soit un extrait de la table games.csv

1 Monopoly	Elizabeth Magie	1903	8	2	6	19.99
2 Scrabble	Alfred Mosher Butts	1938	8	2	4	17.99
3 Clue	Anthony E. Pratt	1944	8	2	6	9.99
4 Candy Land	Eleanor Abbott	1948	3	2	4	7.99
5 Risk	Albert Lamorisse	1957	10	2	5	29.99

**Application.** Quels sont les jeux dans la tables games qui ont fêté leurs anniversaire de 100 années? Et dans quelles années?

Ecrire une instruction SELECT qui répond à cette question et l'exécuter pour avoir le résultat.

SELECT name, cast(year AS INT) + 100  
as century\_year FROM games;      → Le jeu est monopoly, a fêté en 2003

Dans une instruction SELECT exécutée avec Hive, les identifiants (tels que les noms de tables et de colonnes) fonctionneront quelle que soit leur casse (majuscule, minuscule ou mixte).

31

## SELECT

Soit un extrait de la table games.csv

1 Monopoly	Elizabeth Magie	1903	8	2	6	19.99
2 Scrabble	Alfred Mosher Butts	1938	8	2	4	17.99
3 Clue	Anthony E. Pratt	1944	8	2	6	9.99
4 Candy Land	Eleanor Abbott	1948	3	2	4	7.99
5 Risk	Albert Lamorisse	1957	10	2	5	29.99

- Possible avec divers colonnes, possible avec \* et possible avec des fonctions

Select distinct min\_age, max\_players  
FROM games;

min_age	Max_players
8	6
3	4
10	5

Select distinct  
concat(substring(year,1,3), «0s»)  
FROM games;

Concat(substring(year,1,3), «0s»)
1900s
1930s
1940s
1950s

30

## WHERE

Teste une expression booléenne et retourne TRUE, les lignes d'enregistrement

WHERE (opérande(s))

Opérandes :

- Deux colonnes
- Une colonne et littéral
- Une expression et un littéral (red+blue>650)
- Alias dans SELECT n'est pas autorisé avec WHERE (le moteur commence l'exécution en exécutant WHERE)
- Les éléments d'une opérande doivent être dans la même famille (numeric, character string).
- Utiliser round pour éliminer les conflits : 1/3 Vs 0,333 par exemple

Les tests avec les valeurs NULL retournent NULL!!

32

## SELECT

Soit un extrait de la table games.csv

1 Monopoly	Elizabeth Magie	1903	8	2	6	19.99
2 Scrabble	Alfred Mosher Butts	1938	8	2	4	17.99
3 Clue	Anthony E. Pratt	1944	8	2	6	9.99
4 Candy Land	Eleanor Abbott	1948	3	2	4	7.99
5 Risk	Albert Lamorisse	1957	10	2	5	29.99

**Application.** Quelles sont les jeux dans la tables games qui ont fêté leurs anniversaire de 100 années? Et dans quelles années?

```
SELECT name, cast(year AS INT) + 100  
as century_year FROM games;
```

Le résultat de la requête précédente n'est pas exact, i.e. donne une liste contenant les jeux qui ont fêtés et ceux qui n'ont pas encore fêté leurs anniversaire. On ajoute where:

33

## Where

Une table contient 100 lignes. Quelle instruction à utiliser dans SELECT avec WHERE pour mentioner le nombre de lignes à avoir dans le résultat?

- 100 or fewer;
- 100 or more;
- More than 100;

35

## SELECT

Soit un extrait de la table games.csv

1 Monopoly	Elizabeth Magie	1903	8	2	6	19.99
2 Scrabble	Alfred Mosher Butts	1938	8	2	4	17.99
3 Clue	Anthony E. Pratt	1944	8	2	6	9.99
4 Candy Land	Eleanor Abbott	1948	3	2	4	7.99
5 Risk	Albert Lamorisse	1957	10	2	5	29.99

Sachant que [Current\\_date\(\)](#) (Hive 1.2.0) permet de déterminer la date courante et [YEAR](#) permet d'avoir l'année de la date, la requête sera:

```
SELECT name  
FROM games  
WHERE cast(year AS INT)+100 <= YEAR(CURRENT_DATE());
```

34

## Where

Pour utiliser WHERE qui filtre une table en se basant sur les valeurs de la column\_x, SELECT list doit inclure column\_x.

True

False

36

## Where

Ce ci un extrait de quelques ligne de la table students dans une ecole. (GPA: grade point average, ou 4.0 signifie que l'étudiant a le Meilleur score. Absences est le nombre de jours d'absences de l'étudiant, et detention est une punition pour un mauvais comportement.)

id	name	age	gpa	absences	detentions
930	Olufunmilayo Aytон	16	4.00	3	2
667	Vincent Michaelson	15	2.53	12	0
907	Asa Quigg	15	3.57	1	0
168	Kiran Patil	17	3.28	0	3

Quelles instructionq trouvent le meilleur l'étudiant dédié pour representer l'école à une reunion, même si les résultats sont différentes?

- ✓ SELECT name ... WHERE gpa >= 3.5;
- ✓ SELECT name ... WHERE detentions = 0;
- ✗ SELECT name ... WHERE id < 200;
- ✓ SELECT name ... WHERE absences = 0;
- ✗ SELECT name ... WHERE absences > detentions;
- ✗ SELECT name ... WHERE age = 17;

37

Ici on choisit les instructions qui ont un sens pour le choix de l'étudiant. Par exemple l'âge ne peut pas être un critère de sélection.

id	name	age	gpa	absences	detentions
930	Olufunmilayo Aytон	16	4.00	3	2
667	Vincent Michaelson	15	2.53	12	0
907	Asa Quigg	15	3.57	1	0
168	Kiran Patil	17	3.28	0	3

## Where

Lister les étudiants qui ont un GPA aumoins 3.5, et ayant 3 detentions auplus (detention <=3) ou plus de 5 absences (abcences>5)

- ✗ SELECT \* FROM students WHERE gpa >= 3.5 AND NOT (detentions > 3 OR absences > 5)
- ✓ SELECT \* FROM students WHERE gpa >= 3.5 AND (detentions <= 3 OR absences > 5)
- ✗ SELECT \* FROM students WHERE (gpa >= 3.5 AND NOT detentions > 3) OR absences > 5
- ✓ SELECT \* FROM students WHERE gpa >= 3.5 AND (NOT detentions > 3 OR absences > 5)
- ✗ SELECT \* FROM students WHERE gpa >= 3.5 AND NOT detentions > 3 OR absences > 5
- ✗ SELECT \* FROM students WHERE (gpa >= 3.5 AND detentions <= 3) OR absences > 5
- ✗ SELECT \* FROM students WHERE gpa >= 3.5 AND detentions <= 3 OR absences > 5

38

## Interrogation de Bases de données

```
create database office; // créer une base de données appelée office
show databases; // afficher la base de données créée
drop database office; // effacer la base de données office si elle est vide
drop database office cascade; // effacer les table de la base de données
//office si elle n'est pas vide
create database office; // recréer la base de données office
use office; // utiliser office comme la base de données courante
```

39

## Bases de données

```
hive> create table employee
  > (Id INT, Name STRING, Dept STRING, Yoj INT, salary INT)
  > row format delimited fields terminated by ','
  > tblproperties ("skip.header.line.count"="1");
```

- La dernière ligne mentionne qu'il faut ignorer la ligne d'entête
- Il est préférable d'ajouter la ligne: STORED AS TEXTFILE
- Ne mettre le point virgule qu'à la fin de la création des champs de table.  
Mettre un point virgule informe hive d'exécuter la ligne.

>show tables;

Montre les tables existantes y compris  
la table employee

```
hive> show tables;
OK
employee
Time taken: 0.056 seconds, Fetched: 1 row(s)
```

> describe employee;

Montre le schema de la table employee

```
hive> describe employee;
OK
id          int
name        string
dept        string
yoj         int
salary      int
Time taken: 0.201 seconds, Fetched: 5 row(s)
```

40

## Bases de données

Supposons qu'il existe un fichier employee.csv qui contient les données des employées. Il est possible de transférer les données du fichier dans la table employee.

```
hive> LOAD DATA LOCAL INPATH  
> '/home/u1/employee.csv'  
> INTO TABLE employee;
```

```
Loading data to table office.employee  
Table office.employee stats: [numFiles=1, totalSize=116]  
OK  
Time taken: 1.147 seconds
```

**Remarque:** Quand une table est créée dans Hive, elle sera stockée dans le dossier de warehouse. Il est possible de créer des tables externes qui ne seront pas dans le dossier warehouse

41

## Données HIVE sur Hadoop

Ce qui se passe sur hadoop:

Ouvrir un nouveau terminal et taper:

```
hdfs dfs -ls /user/hive
```

```
Found 1 items  
drwxrwxrwx - hive supergroup 0 2017-03-30 22:20 /user/hive/warehouse
```

Lors de l'installation de hive, le dossier warehouse sera installé automatiquement sur hadoop

Afficher le contenu de user/hive/warehouse

```
Hdfs dfs -ls /user/hive/warehouse
```

43

## Bases de données

- Il est possible de faire des requêtes sur la table employee.

```
select * from employee; // affiche tout le contenu de la table  
select count (*) from employee; // affiche le nombre de lignes de employee  
select * from office.employee WHERE Salary>25000; // affiche le résultat:
```

```
hive> select * from office.employee WHERE Salary>25000;  
OK  
1 Rose IT 2012 26000  
3 Mike HR 2013 30000  
Time taken: 0.296 seconds, Fetched: 2 row(s)
```

- Hive permet aussi de renommer la table.

```
hive> alter table office.employee RENAME TO office.employees;  
OK  
Time taken: 0.496 seconds  
hive> show tables;  
OK  
employees  
Time taken: 0.024 seconds, Fetched: 1 row(s)
```

42

## Données HIVE sur Hadoop

```
ui@machine1:~$ hdfs dfs -ls /user/hive/warehouse  
  
22/04/12 01:50:57 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... Using the generic classloader instead.  
Found 16 items  
drwxr-xr-x - u1 supergroup 0 2022-04-12 01:28 /user/hive/warehouse/customers  
drwxrwxr-x - u1 supergroup 0 2019-01-26 12:37 /user/hive/warehouse/db.db  
drwxrwxr-x - u1 supergroup 0 2022-04-06 17:08 /user/hive/warehouse/employee  
drwxr-xr-x - u1 supergroup 0 2020-10-18 18:33 /user/hive/warehouse/employees  
drwxr-xr-x - u1 supergroup 0 2020-12-16 21:15 /user/hive/warehouse/fixed  
drwxrwxr-x - u1 supergroup 0 2020-10-18 19:25 /user/hive/warehouse/fly.db  
drwxrwxr-x - u1 supergroup 0 2020-12-15 10:41 /user/hive/warehouse/fun.db  
drwxr-xr-x - u1 supergroup 0 2020-10-18 18:33 /user/hive/warehouse/investors  
drwxr-xr-x - u1 supergroup 0 2020-10-18 18:34 /user/hive/warehouse/investors_parquet  
drwxr-xr-x - u1 supergroup 0 2020-10-18 18:34 /user/hive/warehouse/offices  
drwxr-xr-x - u1 supergroup 0 2020-10-18 18:34 /user/hive/warehouse/orders  
drwxr-xr-x - u1 supergroup 0 2020-10-18 18:35 /user/hive/warehouse/salary_grades  
drwxr-xr-x - u1 supergroup 0 2020-10-18 18:35 /user/hive/warehouse/subscribers  
drwxrwxr-x - u1 supergroup 0 2020-12-15 10:53 /user/hive/warehouse/toy.db  
drwxrwxr-x - u1 supergroup 0 2022-04-04 10:41 /user/hive/warehouse/userdb.db  
drwxrwxr-x - u1 supergroup 0 2020-12-15 09:53 /user/hive/warehouse/wax.db
```

Toutes les bases de données créées seront affichées. Les db (eg., toy.db) sont des dossiers créés dans le dossier warehouse dans hadoop

En ouvrant les db, on trouve les tables sous forme de dossiers, en les ouvrant on trouve les fichiers que nous avons copié dans les tables.

44

## Données HIVE sur Hadoop

```
ui@machine1:~$ hdfs dfs -ls /user/hive/warehouse
22/04/12 01:50:57 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Found 16 items
drwxr-xr-x  - u1 supergroup          0 2022-04-12 01:28 /user/hive/warehouse/customers
drwxrwxr-x  - u1 supergroup          0 2019-01-26 12:37 /user/hive/warehouse/db.db
drwxrwxr-x  - u1 supergroup          0 2022-04-06 17:08 /user/hive/warehouse/employee
drwxr-xr-x  - u1 supergroup          0 2020-10-18 18:33 /user/hive/warehouse/employees
drwxr-xr-x  - u1 supergroup          0 2020-12-18 21:15 /user/hive/warehouse/fixed
drwxrwxr-x  - u1 supergroup          0 2020-10-18 19:25 /user/hive/warehouse/fly.db
drwxrwxr-x  - u1 supergroup          0 2020-12-15 16:41 /user/hive/warehouse/fun.db
drwxr-xr-x  - u1 supergroup          0 2020-10-18 18:33 /user/hive/warehouse/investors
drwxr-xr-x  - u1 supergroup          0 2020-10-18 18:34 /user/hive/warehouse/investors_parquet
drwxr-xr-x  - u1 supergroup          0 2020-10-18 18:34 /user/hive/warehouse/offices
drwxr-xr-x  - u1 supergroup          0 2020-10-18 18:34 /user/hive/warehouse/orders
drwxr-xr-x  - u1 supergroup          0 2020-10-18 18:35 /user/hive/warehouse/salary_grades
drwxr-xr-x  - u1 supergroup          0 2020-10-18 18:35 /user/hive/warehouse/subscribers
drwxrwxr-x  - u1 supergroup          0 2020-12-15 16:53 /user/hive/warehouse/toy.db
drwxrwxr-x  - u1 supergroup          0 2022-04-04 10:41 /user/hive/warehouse/userdb.db
drwxrwxr-x  - u1 supergroup          0 2020-12-15 09:53 /user/hive/warehouse/wax.db
```

Exemple: on ouvre la db toy.db

```
ui@machine1:~$ hdfs dfs -ls /user/hive/warehouse/toy.db
22/04/12 09:18:18 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Found 2 items
drwxrwxr-x  - u1 supergroup          0 2020-12-15 16:55 /user/hive/warehouse/toy.db/makers
drwxrwxr-x  - u1 supergroup          0 2020-12-15 16:56 /user/hive/warehouse/toy.db/toys
```

45

## Données HIVE sur Hadoop

Exemple: on ouvre la table makers

```
ui@machine1:~$ hdfs dfs -ls /user/hive/warehouse/toy.db/makers
22/04/12 09:22:54 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform...
Found 1 items
-rw-r--r--  1 u1 supergroup      79 2020-12-15 16:55 /user/hive/warehouse/toy.db/makers.makers.tsv
```

On trouve le fichier makers.tsv utilisé pour la copie de données dans la table makers

HIVE juste fait une projection sur les données

46

## Types de tables dans HIVE

Il y'a deux types de tables dans HIVE:

- Table gérée
- Table externe

Par défaut, chaque table créée est une table gérée

Exemple: tapez show tables;

```
hive> show tables;
customers
employee
office
orders
```

47

## Types de tables dans HIVE

Exemple: 2) afficher les détails de la table employee  
hive> describe formatted employee;

```
> describe formatted employee;
OK
# col_name          data_type          comment
id                  int
name                string
dept                string
yoj                 int
salary               int

# Detailed Table Information
Database:           default
Owner:              u1
CreateTime:         Wed Apr 06 17:08:13 CEST 2022
LastAccessTime:     UNKNOWN
Retention:          0
Location:           hdfs://localhost:9000/user/hive/warehouse/employee
Table Type:         MANAGED TABLE
```

48

## Types de tables dans HIVE

Managed table: signifie la table est gérée par HIVE

Remarquer la location: la table est enregistrée sous /user/hive/warehouse et il n'est pas possible de la changer.

```
> describe formatted employee;
OK
# col_name          data_type      comment
id                  int
name                string
dept                string
yoj                 int
salarly             int

# Detailed Table Information
Database:          default
Owner:              u1
CreateTime:         Wed Apr 06 17:08:13 CEST 2022
LastAccessTime:     UNKNOWN
Retention:          0
Location:           hdfs://localhost:9000/user/hive/warehouse/employee
Table Type:         MANAGED_TABLE
```

49

## Types de tables dans HIVE

Table externe:

```
hive> create external table records2(num INT, date STRING, custNum INT,
   amount DOUBLE, category STRING, product STRING, city STRING, state
   STRING, spendBy STRING)
> row format delimited
> fields terminated by ','
> location '/home/u1/mycustomer'
```

La syntaxe de création est la même que celle d'une table interne, à l'exception qu'il faut ajouter le mot 'external' après create, et location 'chemin'. Ce dernier existera sur hadoop. Cette création permet de créer un dossier mycustomer vide ( car il n'y a pas des données)

50

## Types de tables dans HIVE

Pour avoir des données dans la table il faut copier le fichier de données (txt ou csv, ..) dans le dossier mycustomer.

C'est possible maintenant de faire des requêtes sql sur la table records2.

- Lorsqu'on efface une table gérée , elle sera effacée avec ses données.
- Lorsqu'on efface une table externe, les données ne seront pas effacées.

Exemple: effacer la table employee

```
drop table employee
drop table records2
```

- Vérifier si les tables sont effacées:

```
show tables
```

51

## Gestion de tables

**Exercice** soit deux fichiers: customer.csv et order.csv

customer.csv

Id	Name	Age	Address	Salary
1	Rony	32	New York	2000
2	Kate	25	Florida	1500
3	Kim	23	Seattle	2000
4	Clay	25	Boston	6500
5	Henry	27	California	8500
6	Kit	22	Chicago	4500
7	Muffy	24	New York	10000

order.csv

Oid	Date	Customer id	Amount
102	2018-08-08	3	3000
100	2018-08-03	3	1500
101	2018-11-02	2	1560
103	2018-11-08	4	2060

- Pour visualiser le contenu d'un fichier, taper: vi nomFichier.csv
- Créer une table customer et une table orders avec les champs nécessaires puis les remplir à partir les deux fichiers ci-dessus qui existent dans le dossier home/u1/documents/

52

## Gestion de tables

```
Oid,Date,Customer_id,Amount  
102,2018-08-08,3,3000  
100,2018-08-03,3,1500  
101,2018-11-02,2,1560  
103,2018-11-08,4,2060
```

```
hive> create table order  
> (id INT, Date DATE, Customer_id INT, Amount FLOAT)  
> row format delimited fields terminated by ','  
> tblproperties ("skip.header.line.count"="1");
```

```
hive> LOAD DATA LOCAL INPATH  
> 'home/u1/order.csv'  
> INTO TABLE order;
```

53

## Gestion de tables

customer.csv

```
Id,Name,Age,Address,Salary  
1,Rony,32,New York,2000  
2,Kate,25,Florida,1500  
3,Kim,23,Seattle,2000  
4,Clay,25,Boston,6500
```

```
hive> create table order  
> (id INT, Date DATE, Customer_id INT, Amount FLOAT)  
> row format delimited fields terminated by ','  
>tblproperties ("skip.header.line.count"="1");
```

```
hive> create table customer
```

```
> (id INT, name STRING, age INT, address STRING, salary FLOAT)  
> row format delimited fields terminated by ','  
>tblproperties ("skip.header.line.count"="1");
```

54

## Gestion de tables

customer.csv

```
Id,Name,Age,Address,Salary  
1,Rony,32,New York,2000  
2,Kate,25,Florida,1500  
3,Kim,23,Seattle,2000  
4,Clay,25,Boston,6500
```

```
hive> create table order  
> (id INT, Date DATE, Customer_id INT, Amount FLOAT)  
> row format delimited fields terminated by ','  
>tblproperties ("skip.header.line.count"="1");
```

```
hive> LOAD DATA LOCAL INPATH  
> 'home/u1/customer.csv'  
> INTO TABLE customer;
```

55

## Gestion de tables

Trouver les noms des customers qui ont passé des ordres et trouver le montant de ces ordres

```
hive> select c.Id, c.Name, c.Age, o.Amount  
> FROM customer c JOIN order o  
> ON (c.id = o.Customer_id);  
Query ID = cloudera_20190328093636_7cledblb-785c-4163-bfb3-c6e346d088b4  
Total jobs = 1
```

Résultat:

```
MapReduce Jobs Launched:  
Stage-Stage-3: Map: 1 Cumulative CPU: 3.42 sec HDFS Read: 7026 HDFS Write: 66 SUCCESS  
Total MapReduce CPU Time Spent: 3 seconds 420 msec  
OK  
2      Kate    25      1560.0  
3      Kim     23      3000.0  
3      Kim     23      1500.0  
4      Clay    25      2060.0  
Time taken: 53.151 seconds, Fetched: 4 row(s)
```

56

## Gestion de tables

### L'instruction case en hive

```
CASE
WHEN condition_1 THEN result_1
WHEN condition_2 THEN result_2
ELSE result_n
END
```

#### Exemple:

Soit la table Etudiants ayant les colonnes suivantes:

Etudiant_Id	Nom	Note_math
23512	Saleh	12
25847	Ali	8
58476	Salem	10
86957	Mohsen	9
47589	Jamil	17

Ecrire une requête permettant d'afficher les Id et les noms des étudiants qui ont réussis la matière mathématiques et ceux qui ont échoué en affichant respectivement: « passe », « échoue » respectivement. Sinon « pas de notes ». Donner le nom « result » pour la nouvelle colonne.

```
SELECT etudiant_id, nom,
CASE
    WHEN Note_math >=10 THEN "passe"
    WHEN Note_math <10 THEN "échoue "
    ELSE "pas de notes "
END
as result
```

```
FROM Etudiants;
```

57

## Gestion de tables

```
hive> select * from patient_details;
OK
patient_id      name      oxygen_level
P09821        Kevin      98
P03892        Peter      83
P07837        Robert     97
P06832        Jessica    81
P02791        Emma       94
Time taken: 0.055 seconds, Fetched: 5 row(s)
```

Si Oxygen\_level <85, le patient a besoin de ventilator support. Ecrire une requête permettant d'afficher pour chaque patient, « yes » s'il a un manque d'oxygène et a besoin d'un ventilator support, « no » sinon. Donner le nom « ventilator\_support » à la colonne résultat.

```
select patient_id, name, oxygen_level,
if(oxygen_level<85, 'Yes', 'No')
as ventilator_support
from patient_details;
```

59

## Gestion de tables

23512	Saleh, passe
25847	Ali, échoue
58476	Salem, passe
86957	Mohsen, échoue
47589	Jamil, passe

### L'instruction If en hive

```
if(boolean testCondition, valueTrue, valueFalseOrNull)
```

#### Exemple:

Soit la table Patient\_details ayant les colonnes suivantes:

```
hive> select * from patient_details;
OK
patient_id      name      oxygen_level
P09821        Kevin      98
P03892        Peter      83
P07837        Robert     97
P06832        Jessica    81
P02791        Emma       94
Time taken: 0.055 seconds, Fetched: 5 row(s)
```

58

## Gestion de tables

```
hive> select patient_id, name, oxygen_level,
> if(oxygen_level<85, 'Yes', 'No')
> as ventilator_support
> from patient_details;
OK
```

patient_id	name	oxygen_level	ventilator_support
P09821	Kevin	98	No
P03892	Peter	83	Yes
P07837	Robert	97	No
P06832	Jessica	81	Yes
P02791	Emma	94	No

60

## La fonction SUM() en hive

### Exemple

Soit la table **item\_purchase** in avec les détails suivants:

```
hive> select * from item_purchase;
OK
item_id product_category      amount   purchase_date
E-4897 Electronics            1200    2021-06-01
C-4897 Cosmetics              700     2021-06-02
C-3890 Cosmetics              350     2021-06-01
G-7832 Grocery                590     2021-06-05
E-9898 Electronics             820     2021-06-04
E-6773 Electronics             100     2021-06-02
Time taken: 0.051 seconds, Fetched: 6 row(s)
```

Trouver le [montant des ventes](#) du produit [Electronics](#) en donnant le nom [total\\_amount](#) au colonne résultat.

61

```
hive> select * from item_purchase;
OK
item_id product_category      amount   purchase_date
E-4897 Electronics            1200    2021-06-01
C-4897 Cosmetics              700     2021-06-02
C-3890 Cosmetics              350     2021-06-01
G-7832 Grocery                590     2021-06-05
E-9898 Electronics             820     2021-06-04
E-6773 Electronics             100     2021-06-02
Time taken: 0.051 seconds, Fetched: 6 row(s)
```

```
hive> select sum(amount) as total_amount from item_purchase
> where product_category='Electronics';
```

Résultat.

```
total_amount
2120
Time taken: 45.01 seconds, Fetched: 1 row(s)
```

62

## La fonction SUM() avec l'instruction If en hive

La function [SUM\(\)](#) retourne un double, qui est la somme de toutes les valeurs d'une colonne. Si elle est utilisée ave group, elle fait la somme de chaque groupe. Elle ignore les duplications si DISTINCT est utilisée.

Hive supporte d'aggréer les fonctions comme la fonction [SUM\(\)](#) pour ajouter les valeurs rentrées à une instruction if.

### Exemple:

Retournons à l'exemple précédent. Ecrire la requête permettant de trouver le nombre de personnes nécessitant un ventilator\_support,

63

### Exemple:

Retournons à l'exemple précédent. Ecrire la requête permettant de trouver le nombre de personnes nécessitant un ventilator\_support,

```
select
sum(if(oxygen_level<85,1,0))
as num_of_ventilators
from patient_details;
```

En agrément les valeurs de retour avec la condition if, la valeur quand la condition est true sera 1 et quand elle est false, la valeur sera 0. Donner le nom "num\_ventilators" à la nouvelle colonne.

64

## Gestion de tables

```
hive> select * from patient_details;
OK
patient_id      name    oxygen_level
P09821  Kevin   98
P03892  Peter   83
P07837  Robert  97
P06832  Jessica 81
P02791  Emma    94
Time taken: 0.055 seconds, Fetched: 5 row(s)
```

```
select
sum(if(oxygen_level<85,1,0))
as num_of_ventilators
from patient_details;
```

```
num_of_ventilators
2
Time taken: 147.885 seconds, Fetched: 1 row(s)
hive>
```

65

## Gestion de tables

### La fonction count() avec l'instruction group by

Soit la table suivante:

```
hive> select * from student_results;
OK
student_id      name    subject marks
1980    Alex    Maths   98
1980    Alex    Physics 90
8390    Chris   Chemistry 59
8390    Chris   Biology  91
1980    Alex    Chemistry 65
1980    Alex    Biology  99
8938    Smith   Maths   95
8938    Smith   Physics  54
8938    Smith   Chemistry 60
8938    Smith   Biology  94
8390    Chris   Maths   62
8390    Chris   Physics  90
Time taken: 0.052 seconds, Fetched: 12 row(s)
```

66

```
hive> select * from student_results;
OK
student_id      name    subject marks
1980    Alex    Maths   98
1980    Alex    Physics 90
8390    Chris   Chemistry 59
8390    Chris   Biology  91
1980    Alex    Chemistry 65
1980    Alex    Biology  99
8938    Smith   Maths   95
8938    Smith   Physics  54
8938    Smith   Chemistry 60
8938    Smith   Biology  94
8390    Chris   Maths   62
8390    Chris   Physics  90
Time taken: 0.052 seconds, Fetched: 12 row(s)
```

## Gestion de tables

Trouver, pour chaque sujet, le nombre des étudiants qui ont une note supérieure ou égale à 90. Donner le nom "greater\_than\_90" au colonne résultat.

```
select subject,count(*) as greater_than_90
from student_results
where marks>=90
group by subject;
```

subject	greater_than_90
Biology	3
Maths	2
Physics	2

67

```
hive> select * from student_results;
```

```
OK
student_id      name    subject marks
1980    Alex    Maths   98
1980    Alex    Physics 90
8390    Chris   Chemistry 59
8390    Chris   Biology  91
1980    Alex    Chemistry 65
1980    Alex    Biology  99
8938    Smith   Maths   95
8938    Smith   Physics  54
8938    Smith   Chemistry 60
8938    Smith   Biology  94
8390    Chris   Maths   62
8390    Chris   Physics  90
Time taken: 0.052 seconds, Fetched: 12 row(s)
```

## Gestion de tables

Trouver, pour chaque sujet, le nombre des étudiants qui ont une note inférieure ou égale à 65. Donner le nom "lesser\_than\_65" au colonne résultat.

```
select subject,count(*) as lesser_than_65
from student_results
where marks<=65
group by subject;
```

subject	lesser_than_65
Chemistry	3
Maths	1
Physics	1

68

## La fonction sum() avec l'instruction case

On reprend le même exemple précédent. On utilise la fonction SUM pour répondre aux deux questions précédentes. Une seule instruction sera employée.

```
hive> select subject,
> sum(case when marks>=90 then 1 else 0 end) as greater_than_90,
> sum(case when marks<=65 then 1 else 0 end) as lesser_than_65
> from student_results
> group by subject;
```

subject	greater_than_90	lesser_than_65
Biology	3	0
Chemistry	0	3
Maths	2	1
Physics	2	1

69

customer.csv

Id	Name	Age	Address	Salary
1	Rony	32	New York	2000
2	Kate	25	Florida	1500
3	Kim	23	Seattle	2000
4	Clay	25	Boston	6500
5	Henry	27	California	8500
6	Kit	22	Chicago	4500
7	Muffy	24	New York	10000

order.csv

Oid	Date	Customer_id	Amount
102	2018-08-08	3	3000
100	2018-08-03	3	1500
101	2018-11-02	2	1560
103	2018-11-08	4	2060

1) Trouver les : Id, name, age des customers qui ont passé des orders et trouver les montants de ces orders. Insérer le résultat dans une table out1.

1) Créer la table.

```
Create table out1(custId int, custName String, custAge int, amount float)
```

```
Row format delimited fields terminated by ';
```

2) Créer la requête et insérer le résultat dans la table.

```
Insert overwrite table out1
```

```
Select c.id, c.name, c.age, o.amount
```

```
From customer c JOIN order o ON (c.id = o.customer_id);
```

71

## Sauvegarde du résultat d'une requête dans une table

Il est possible de sauvegarder le résultat d'une requête dans une table. Il faut tout d'abord créer la table avec les champs désirés puis écrire la requête et insérer le résultat dans la table.

**Exemple:** reprenons les tables customer et order.

customer.csv

Id	Name	Age	Address	Salary
1	Rony	32	New York	2000
2	Kate	25	Florida	1500
3	Kim	23	Seattle	2000
4	Clay	25	Boston	6500
5	Henry	27	California	8500
6	Kit	22	Chicago	4500
7	Muffy	24	New York	10000

order.csv

Oid	Date	Customer_id	Amount
102	2018-08-08	3	3000
100	2018-08-03	3	1500
101	2018-11-02	2	1560
103	2018-11-08	4	2060

70

Out1

custId	custName	custAge	amount
2	Kate	25	1560
3	Kim	23	3000
3	kim	23	1500
4	Clay	25	2060

2) Classifier les customers de la table out1 suivant leurs âges (age<30:jeune, 30≤ age<50: moyen, age ≥ 50: vieux. Insérer le résultat dans une table out2.

1) Créer la table out2.

```
Create table out2(custId int, custName String, custAge int, order float, level string)
```

```
Row format delimited fields terminated by ';
```

2) Créer la requête et insérer le résultat dans la table out2.

```
Insert overwrite table out2
```

```
Select *, case
```

```
When custAge <30 then 'jeune'
```

```
When custAge >=30 and age <50 then 'moyen'
```

```
When custAge >=50 then 'vieux'
```

```
End
```

```
From out1;
```

72

Out1

	custId	custName	custAge	amount		custId	custName	custAge	order	level
2	Kate	25	1560			2	Kate	25	1560	jeune
3	Kim	23	3000			3	Kim	23	3000	jeune
3	kim	23	1500			3	kim	23	1500	jeune
4	Clay	25	2060			4	Clay	25	2060	jeune

3) Faire la somme des "order" par "level". Insérer le résultat dans une table

out3.

1) Créer la table out3.

```
Create table out3(level string, amount float)
```

```
Row format delimited fields terminated by '';
```

2) Créer la requête et insérer le résultat dans la table out3.

```
Insert overwrite table out3
```

```
select level, sum(amount) from out2 group by level;
```

73

## Types de tables dans HIVE

### Utiliser beeline



La commande pour connecter beeline au serveur HIVE2 à partir du terminal:

```
beeline -u jdbc:hive2://  
-n <user name>  
-p <password>
```

Pour quitter beeline:

```
!q ou !quit
```

74

Beeline est un client il doit être connecter au serveur HIVE pour créer / interroger les bases de données.

### Utiliser beeline

```
u1@machine1:~$ beeline -u jdbc:hive2:// -n u1 -p hadoop  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/usr/local/hive/apache-hive-2.3.4-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/usr/local/hadoop/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.  
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]  
Connecting to jdbc:hive2://  
22/04/13 16:54:59 [main]: WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-Java classes where applicable  
22/04/13 16:55:02 [main]: WARN session.SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFactory.  
Connected to: Apache Hive (version 2.3.4)  
Driver: Hive JDBC (version 2.3.4)  
Transaction Isolation: TRANSACTION_REPEATABLE_READ  
Beeline version 2.3.4 by Apache Hive  
0: jdbc:hive2://>
```

Taper: show databases;

```
0: jdbc:hive2://> show databases;  
OK  
+-----+  
| database_name |  
+-----+  
| db          |  
| default     |  
| fly         |  
| fun         |  
| toy         |  
| userdb     |  
| wax         |  
+-----+  
7 rows selected (0.3 seconds)  
0: jdbc:hive2://>
```

75