WS : Manipulation et analyse de données avec Hive



**Correction**

# Création d’une base de données et des tables

Dans cette partie, nous allons voir comment créer une base de données ainsi que les tables sous Hive à travers HiveQL. Ce dernier est un langage qui utilisé pour adresser des requêtes très proche de SQL et permet également aux développeurs d'intégrer des fonctions Map et Reduce qui tournent en arrière-plan afin de couvrir une plus large palette de problèmes de gestion de données.

Il faut savoir que Hive gère deux types de tables : internes et externes. Principalement, Hive représente un entrepôt de données sous l’écosystème d’Hadoop, ainsi toutes les tables crées seront stockées dans HDFS.

### Démarrage avec Hive

Pour travailler avec Hive, il suffit de définir un schéma qui est associé aux données et stocké dans une bdd MySQL, appelée Metastore. Ce schéma donne les noms et types des colonnes, et structure les informations en tables exploitables par HiveQL.

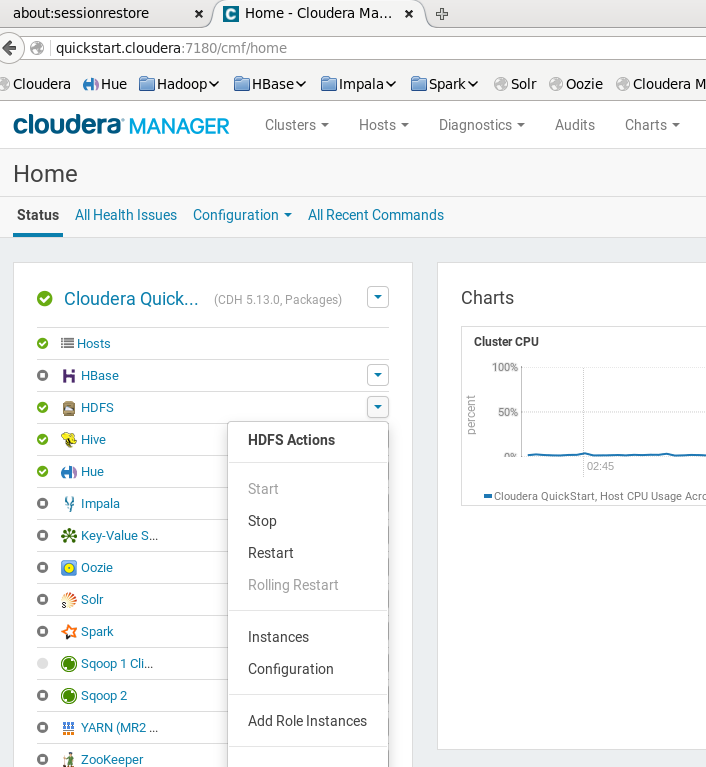
Par défault, hive stocke les bases de données avec les métadonnées des tables dans le répertoire /user/hive/warehouse.

Pour être sûr d’avoir tous les droits sur ce répertoire, autorisez toutes les opérations en E/S comme ci-dessous:

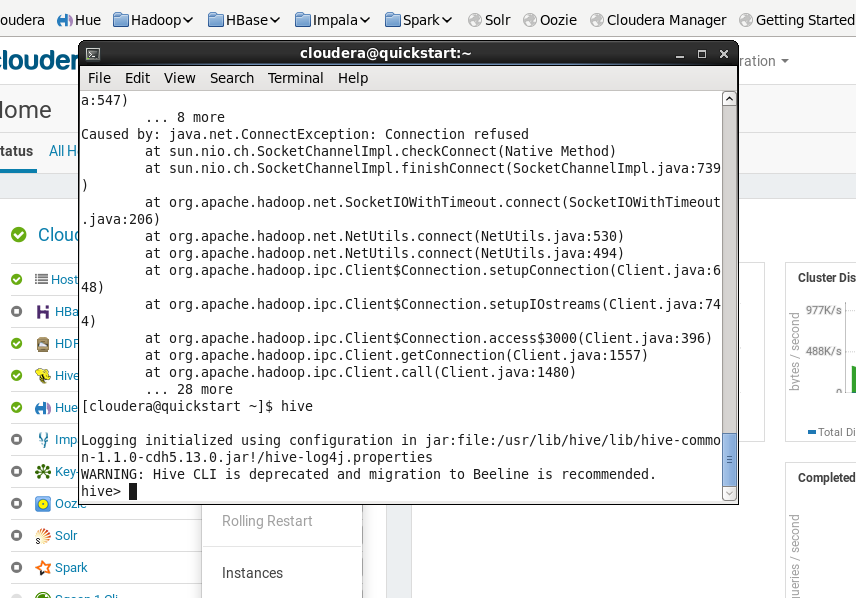
hdfs dfs -chmod 777 /user/hive/warehouse

Vous avez la possibilité de faire des requêtes sous Hive à travers l’interface graphique « Hue » ou bien en console « Hive CLI » (ligne de commande en tapant Hive). Ici, nous allons créer la base de données en ligne de commande et par la suite, nous pouvons travailler sur l’interface Hue.

Avant d’ouvrir la console, il faut démarrer les services de HDFS et de Hive sur Cloudera Manager (voir la figure suivante)



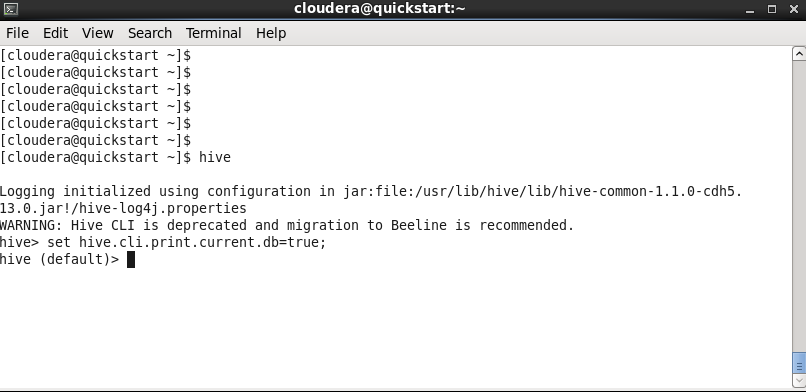
Ensuite, il faut ouvrir le terminal et tapez l’instruction *hive* pour lancer le Hive CLI.



Ce dernier permet de se connecter directement à HDFS ainsi que le Metastore de Hive, et peut être utilisé seulement en mode Host en accédant à ses services.

### Création de bases de données

Avant de créer une nouvelle base de données en console, nous allons voir celle qui est utilisée par défaut. Tapez l’instruction set hive.cli.print.current.db=true;

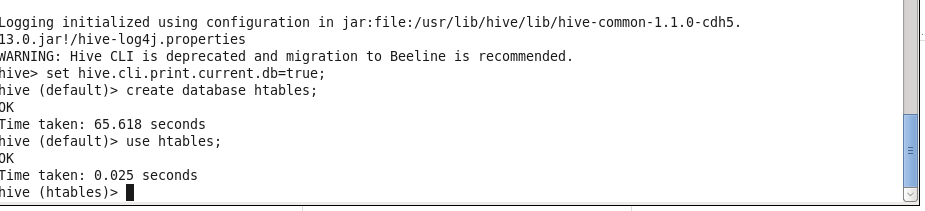
**

Nous pouvons voir maintenant que Hive utilise la base de données « default » (elle est dans le catalogue HCatalogue).

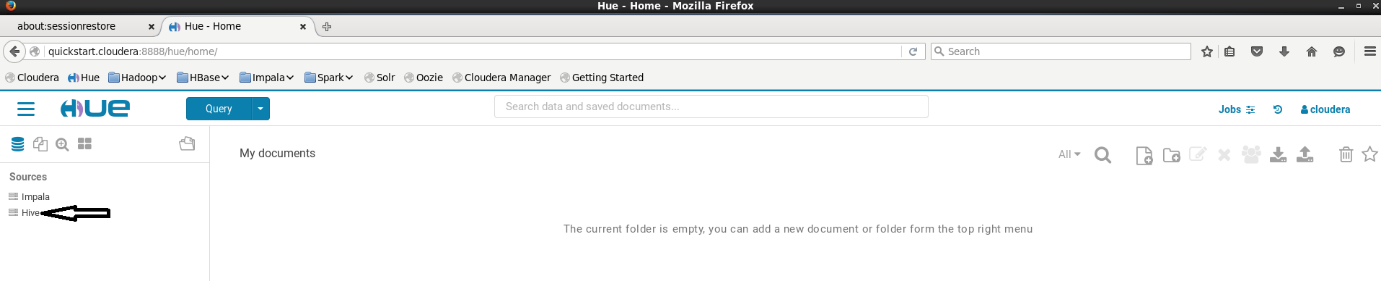
Tapez maintenant cette requête pour créer une nouvelle qui sera nommée « htables » :

CREATE DATABASE htables;

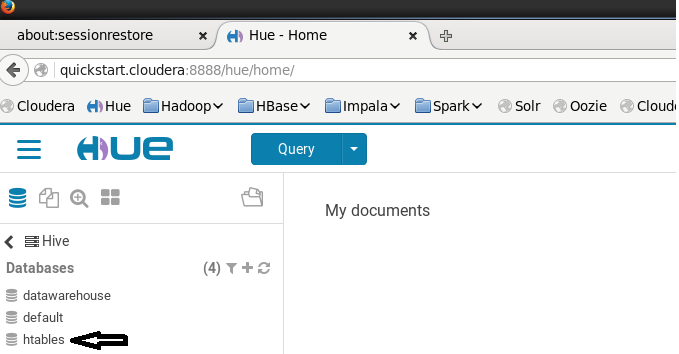
Ensuite pour l’utiliser, il faut le mentionner à Hive avec l’instruction USE htables;



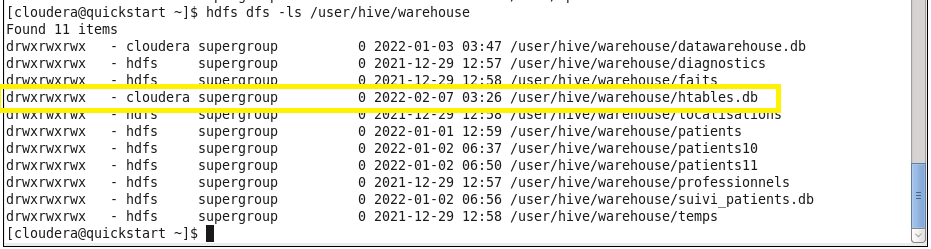
Maintenant, nous pourrions voir sur l’interface Hue, la base de données « htables » déjà crée en choisissant la source Hive.



Toutes les bases de données déjà existantes apparaissent dans la partie gauche comme suit :



Nous pourrions voir aussi que l’emplacement par défaut des bases de données créées sont dans le répertoire de HDFS : */user/hive/warehouse:*

****

Une autre manière de créer une base de données sous hive est d’identifier un autre répertoire sur HDFS est de spécifier la location ainsi que d’autres informations additionnelles:

CREATE DATABASE IF NOT EXISTS htables

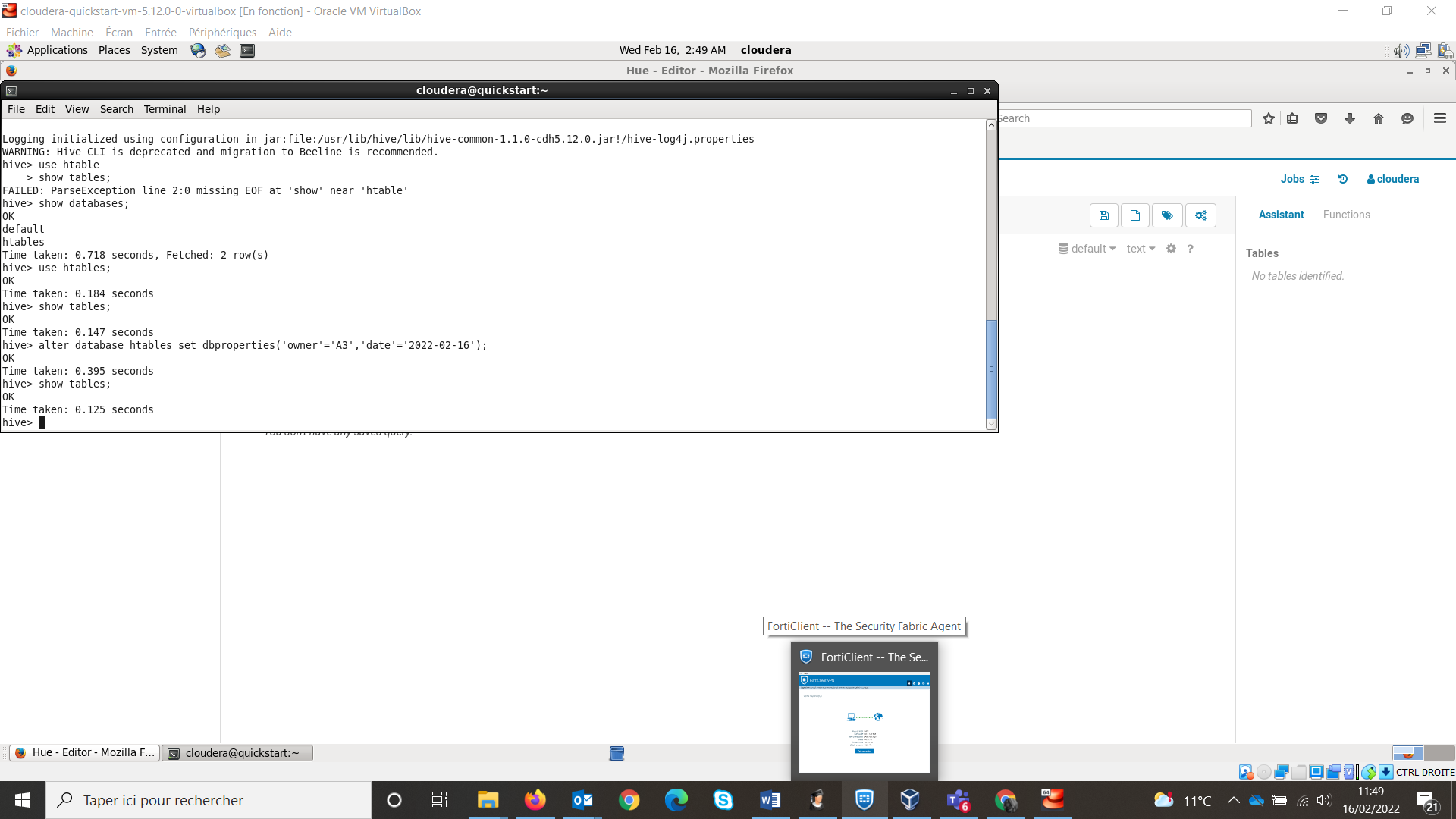
COMMENT 'Base de données du WS Hive'

LOCATION '/user/hive/data/htables';

 Utilisez ALTER pour ajouter des propriétés à la base de données « htable » :

ALTER DATABASE HTABLES SET DBPROPERTIES ('owner'='A3', 'date'='2022-02-03');

Testez l’existence de cette table avec SHOW TABLES; pour afficher la liste des tables (elles sont dans le metastore) ainsi que la description de la base de données pour voir l’effet :



DESCRIBE DATABASE EXTENDED htables;

### Création des tables (externes/internes)

Maintenant que nous avons notre base de données crée, on passe à la création des tables sous Hive.

Pour les besoins de ce ws, vous avez trois fichiers de données qui font références aux employés des certains secteurs d’activités avec les salaires de France:

Copiez le fichier de données « emploi\_population.txt » dans votre répertoire linux dans Cloudera ( ex. /home/cloudera), ensuite créez un répertoire pour héberger le fichier de données Hive:

hdfs dfs -mkdir /user/hive/data

Donnez à ce répertoire les autorisations nécessaires :

hdfs dfs -chmod 777 /user/hive/data

Copiez le fichier de données dans HDFS:

hadoop fs -copyFromLocal /home/cloudera/emploi\_population.txt /user/hive/data

* **Création des tables externes**

La première étape dans la création des tables sous Hive consiste souvent à créer une table intermédiaire (staging table) dans la base de catalogue. Cette table va pointer vers les données du fichier stocké dans HDFS.

Pour ce faire, vous utiliserez l’instruction suivante pour la création d’une table externe. Le choix de cette dernière est justifiée par le fait que les données devront être dupliquées pour une utilisation ultérieure):

CREATE EXTERNAL TABLE IF NOT EXISTS log\_emploi\_population (

genre STRING,

date\_naissance DATE,

fonction STRING,

salaire STRING,

code\_postal STRING,

id INT

)

COMMENT 'table intermediaire'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE

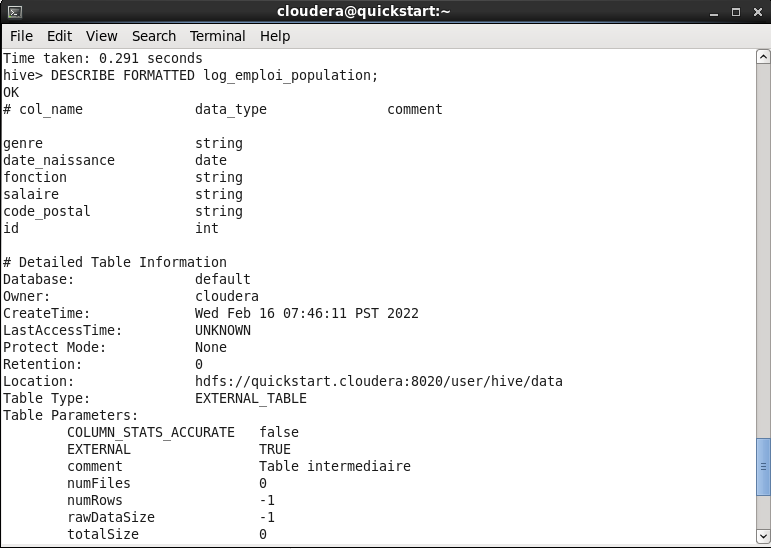
LOCATION '/user/hive/data';

Nous avons choisi une table externe afin de garder les données dans HDFS après la suppression de la table log\_salaire\_population.

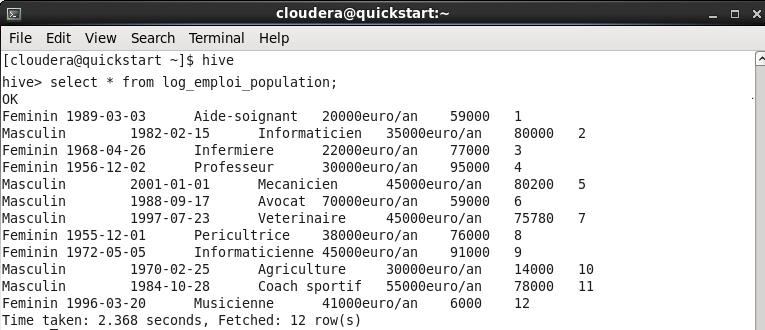
Cela va créer la table log\_emploi\_population remplie automatiquement toutes les colonnes du fichier source. L’instruction *ROW FORMAT DELIMITED FIELDS TERMINATED BY* permet de préciser le code ANSI du séparateur de colonne (ici c’est « ; »). L’instruction STORED AS permet de spécifier le format sur lequel le fichier est sérialisé dans HDFS. Nous remarquons aussi que le début de la requête est classique, sauf les contraintes d’intégrité : il n’y en a pas !

Faites une description de cette table avec l’instruction : DESCRIBE FORMATTED table\_name;



****

Vérifiez que votre table log\_emploi\_population contient bien les données insérées depuis le fichier emploi\_population.txt avec la commandeSELECT \* FROM log\_emploi\_population ;

**

Notez que dans ce mode d'utilisation, la commande DROP TABLE supprimera la table dans Hive, mais pas le fichier de données. Vous pouvez le vérifier en supprimant puis recréant la table « log\_emploi\_population ».

* **Affichage des propriétés d’une table**

Pour connaitre les propriétés de cette table, tapez l’instruction suivante :

SHOW TBLPROPERTIES table\_name;

* **Création d’une table à partir d’une table existante**

Maintenant, nous pourrions construire la table finale dont nous aurons besoin pour nos analyses. Pour cela, il faut identifier à partir de la table intermédiaire les traitements nécessaires pour obtenir les colonnes finales.

Pour ce faire, on choisit de créer une table dont les données sont stockées dans HDFS ainsi que les métadonnées dans la bdd de Hive. Le choix de la table externe est justifié par le fait que les données sont permanentes et pourraient être utilisées par un programme existant.

L’instruction suivant permet de créer une table interne sous Hive (DDL):

CREATE TABLE IF NOT EXISTS emploi\_population AS

SELECT

SUBSTR(genre, 1, 1) AS sexe,

ROUND(DATEDIFF(CURRENT\_TIMESTAMP(), date\_naissance) / 365.2425) AS age,

fonction,

salaire,

code\_postal,

id AS Id\_empl

FROM log\_emploi\_population;

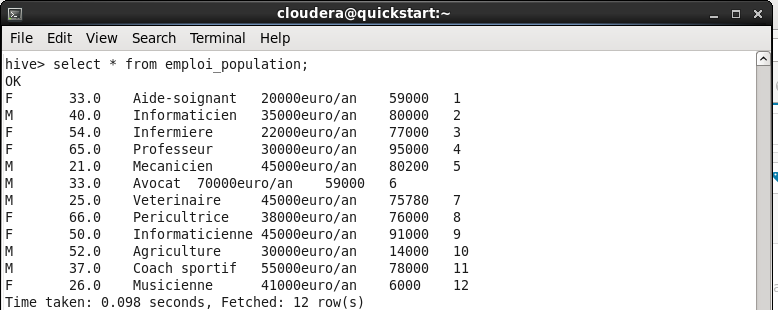
Dans ce cas, la colonne « sexe » est obtenue par l’extraction de la première lettre de la colonne « genre ». La colonne « age » est obtenue en faisant la différence entre la date de la colonne « date\_naissance » et la date d’aujourd’hui. Nous avons traité la colonne id\_empl obtenue à partir de la colonne id.

Il faut s'assurer que la table a été bien chargée avec une requête de select.

Vous pouvez tester aussi l’instruction « DESCRIBE EXTENDED emploi\_population; »pour voir plus de détails sur la table crée (son schéma).

Vérifiez que votre table emploi\_population contient les données souhaitées avec la commande

SELECT \* FROM emploi\_population ;



* **Utilisation d’une table interne**

Nous allons maintenant voir l’utilisation d’une table interne. Avant cela, il faut copiez le fichier « personne.txt » dans HDFS. Les données des tables internes (ou gérée) sont copiées dans l’entrepôt de Hive *(/user/hive/warehouse*)

hadoop fs -copyFromLocal /home/cloudera/personne.txt /user/hive/warehouse/

Pour vérifier que le fichier a bien été copié :

hdfs dfs -ls /user/hive/warehouse

Puis, créez la table interne nommée « personne » avec deux champs : l’identifiant et le nom, sachant que la syntaxe est la même pour la création d’une table externe sans le mot-clé EXTERNAL TABLE et LOCATION:

CREATE TABLE IF NOT EXISTS name\_table (

-- Liste des colonnes et de leurs types de données ici

)

COMMENT 'Description de la table'

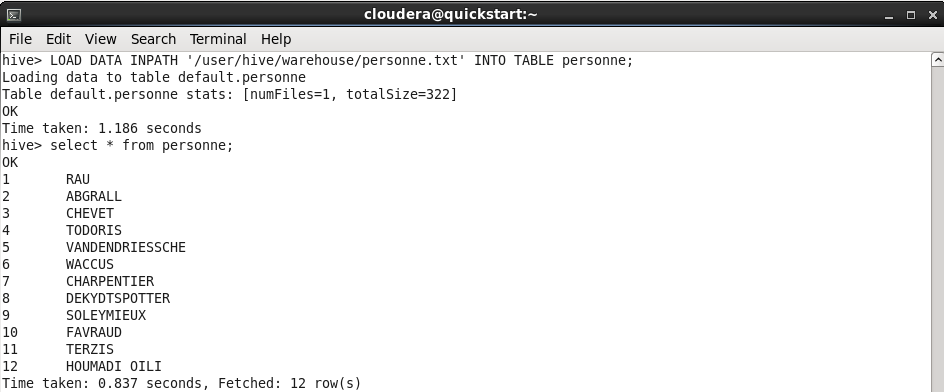
ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE;

Ensuite, il faut charger les données dans cette table :

LOAD DATA INPATH '/user/hive/warehouse/personne.txt' OVERWRITE INTO TABLE personne;



Les données de la table interne sont temporaires, car si nous supprimons cette table, nous perdons aussi le fichier de données dans HDFS. Testez cela en supprimant la table personne et vérifiez sur HDFS l’absence du fichier peronne.txt.

### Partitionnement et Buckets

Apache Hive organise les tables en partitions pour regrouper le même type de données en fonction d’une clé de colonne ou de partition. Ce concept est très utile dans Hive car cela permet d’optimiser les requêtes d’interrogation et de se concentrer uniquement sur une partition au lieu d’une analyse complète d’une table, ainsi améliorer les performances.

Dans Hive, nous avons deux types de partition : statiques et dynamiques. La syntaxe générale est :

CREATE [EXTERNAL] TABLE table\_name (

column1 data\_type,

column2 data\_type,

...

)

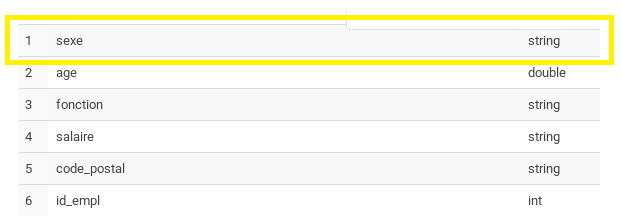
PARTITIONED BY (partition1 data\_type, partition2 data\_type, ...);

Les champs de partitionnement ne doivent pas figurer parmi les champs de la table.

Dans ce ws, nous nous concentrons que sur les partitions dynamiques qui sont configurées par default dans Hive. Cela nous permettra de contrôler toutes les opérations réalisées dessus, et surtout cela permet aussi de gagner du temps lors d’un chargement d’un gros fichier dans les tables de Hive.

Pour pratiquer tout cela, nous allons utiliser la table externe déjà crée (emploi\_population).

Faites *describe emploi\_population* pour visualiser les champs à partitionner :

****

Ensuite sélectionnez la table par sexe : select distinct(sexe) from emploi\_population ;

Nous savons que les valeurs du genre sont le féminin et le masculin. Ainsi, nous créons une partition qui se base sur ces deux valeurs distinctes. Mais avant cela, assurez-vous des valeurs de ces deux paramètres dans hue en tapant ces deux instructions (partitions dynamiques):

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

Redémarrez ensuite la machine Cloudera, et relancez tous les services.

Puis tapez cette instruction afin de créer la partition :

CREATE EXTERNAL TABLE population\_genre (

age INT,

fonction STRING,

salaire STRING,

code\_postal STRING,

id INT

)

PARTITIONED BY (sexe STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',';

Nous remarquons dans ce script, que la partition n’est pas présente dans les colonnes de la table population\_genre.

Il faut ensuite peupler la partition avec l’instruction suivante en respectant l’ordre des colonnes dans la table initiale (emploi\_population) :

INSERT OVERWRITE TABLE population\_genre

PARTITION (sexe)

SELECT age, fonction, salaire, code\_postal, id\_empl, sexe FROM emploi\_population;

Il est important ici de précisier que la partition est insérée en derniere colonne dans la clause « select »

Maintenent, assurez-vous que les partitions apparaissent bien sur l’interface Hue :

SHOW PARTITIONS population\_genre;

Pour observer l’interet des partitions, exécutez la même requête suivante sur les deux tables emploi\_population et population\_genre :

SELECT \* FROM table\_name WHERE sexe = 'F';

Que constatez-vous ?

Réponse : Le temps de réponse de la requête sur la table emploi\_population est de 11s et celle sur la partition et de moins de 1s.



Afin de visualiser les partitions dans HDFS, accédez depuis le navigateur à l’interface du namenode en tapant quickstart.cloudera :50070 puis cliquez sur Utilities-Browse the file system.

* **Bucket**

Dans Hive, les partitions sont subdivisées en Buckets pour donner une structure supplémentaire aux données qui peuvent être utilisées pour des requêtes plus efficaces. Ils sont utilisés afin de résoudre le problème de sur-partitionnement (taille de partitions très grande et variable) et ainsi permet de décomposer les partitions ou/et les tables en parties plus gérables.

Pour alimenter la table Bucket, nous devrons définir la propriété *suivante :*

set hive.enforce.bucketing=true;

Voici maintenant la syntaxe pour créer des Buckets :

CREATE [EXTERNAL] TABLE table\_name

PARTITIONED BY (partition1 data\_type, partition2 data\_type, ...)

CLUSTERED BY (column\_name [ASC|DESC], ...) INTO nombre\_buckets BUCKETS;

* CLUSTERED BY : permet de deviser une table en buckets. Ces derniers représentent des fichiers dans Hive au sein d’un répertoire de la table (non partitionnée) ou bien dans le répertoire de la partition.

Passons maintenant à la pratique. Assurez-vous que les valeurs de ces propriétés sont les suivantes :

SET hive.exec.dynamic.partition=true;

SET hive.exec.dynamic.partition.mode=nonstrict;

SET hive.conf.validation=false;

SET hive.enforce.bucketing=true;

* Nous allons maintenant déplacer le fichier source (internautes\_log.txt) fourni dans ce ws dans le répertoire /user/hive/data/internautes :

hdfs dfs -mkdir /user/hive/data/internautes

hdfs dfs -chmod 777 /user/hive/data/internautes

hdfs dfs -copyFromLocal /home/cloudera/Desktop/internautes\_log.txt /user/hive/data/internautes

Il faut faire attention de ne pas stocker le fichier de données internautes\_log dans le répertoire /user/hive/data car il existe déjà d’autres fichiers de données et au moment de la création d’une table externe, toutes les données de ces fichiers vont alimenter cette table !

Ensuite, nous créons une table intermédiaire nommée log\_internautes et elle sera peuplée automatiquement avec la clause « LOCATION »:

CREATE EXTERNAL TABLE log\_internautes (

id\_internaute INT,

prenom STRING,

site\_consulte STRING,

date\_consultation STRING,

pays STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE

LOCATION '/user/hive/data/internautes';

Si vous analysez le fichier de données des internautes, vous remarquez qu’il y a trois dates différentes ainsi que les identifiants des internautes qui se répètent. Pour cela, vous allez créer des partitions dynamiques basées sur la date de consultation ainsi que des classes par identifiant des internautes regroupés en 4 fichiers pour ces partitions. Si nous résumons cela :

CREATE TABLE log\_internautes\_buckets (

id\_internaute INT,

prenom STRING,

site\_consulte STRING,

pays STRING

)

PARTITIONED BY (date\_consultation STRING)

CLUSTERED BY (id\_internaute) into 4 BUCKETS

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE;

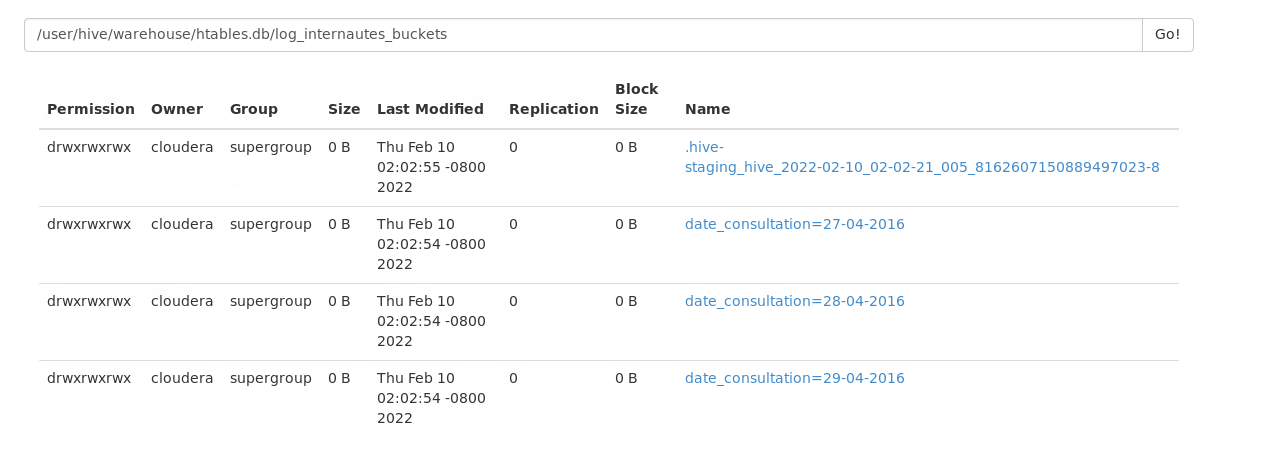
Maintenant que les partitions ainsi que les buckets sont créés, il faudrait les peupler:

INSERT OVERWRITE TABLE log\_internautes\_buckets PARTITION (date\_consultation)

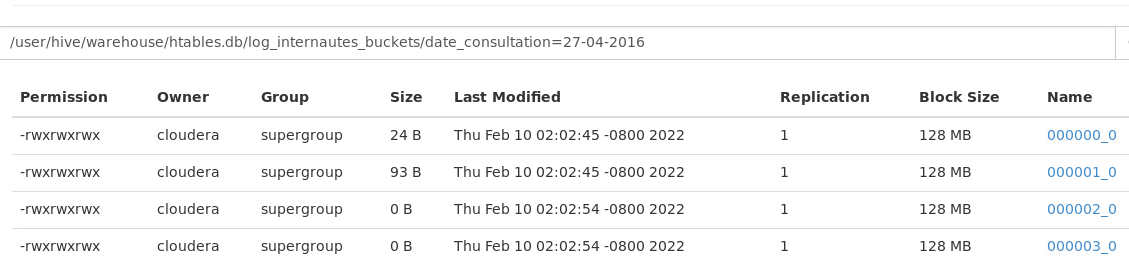
SELECT id\_internaute, prenom, site\_consulte, pays, date\_consultation FROM log\_internautes;

Visualisez les buckets avec DESCRIBE.

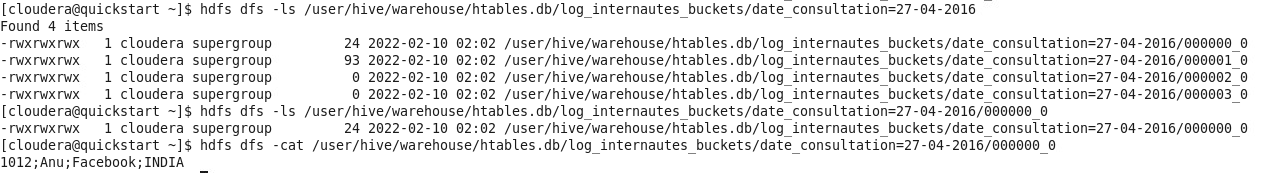
Si nous résumons tout cela, pour chaque date distincte, il y a 4 fichiers de données regroupés par identifiant, comme le montre la figure ci-dessous :



Nous avons 3 valeurs distinctes des dates de consultation, et dans chaque partition, nous trouvons les Buckets :



Si vous remarquez, les deux derniers Buckets sont vides pour la partition du 27-04-2016, car nous avons forcé Hive pour créer 4 Buckets !



Nous avons une autre manière de créer des Buckets, mais cette fois-ci à partir d’une table existante sans les partitions. Si vous remarquez dans le fichier de données, il y a 6 pays mentionnés. Nous allons créer 4 Buckets, et Hive s’organise pour répartir les données sur 4 fichiers :

CREATE TABLE log\_internautes\_buckets2 (

id\_internaute INT,

prenom STRING,

site\_consulte STRING,

date\_consultation STRING,

pays STRING

)

CLUSTERED BY (pays) into 4 BUCKETS

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

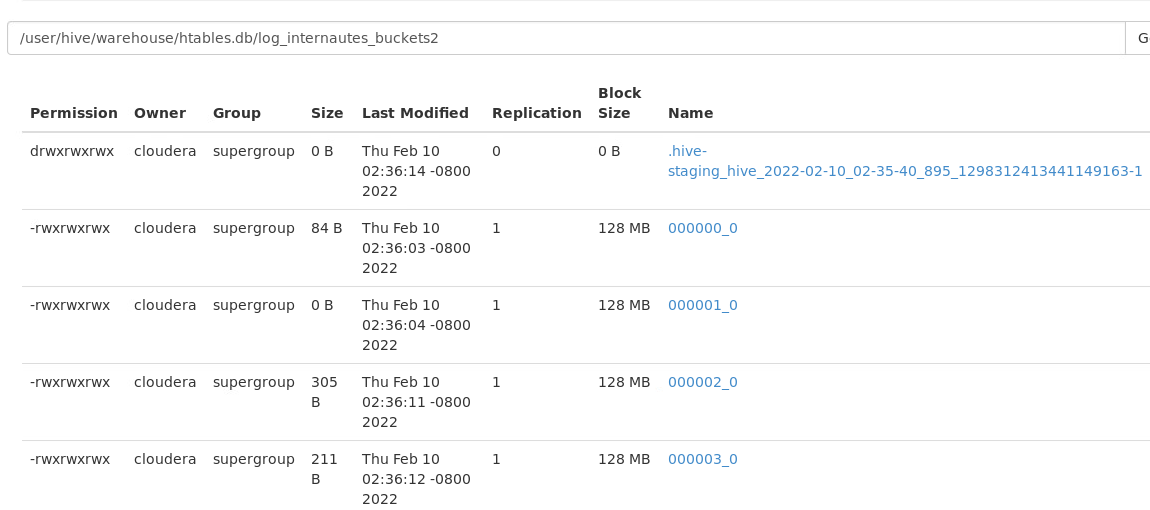
STORED AS TEXTFILE;

Il faudrait les peupler avec cette instruction:

INSERT OVERWRITE TABLE log\_internautes\_buckets2

SELECT id\_internaute, prenom, site\_consulte, date\_consultation, pays FROM log\_internautes;

Visualisons maintenant les Buckets sur HDFS :



Remarque : N'hésitez pas à répéter les instructions liées aux partitions et aux seaux (buckets), sinon il en créera un seul :

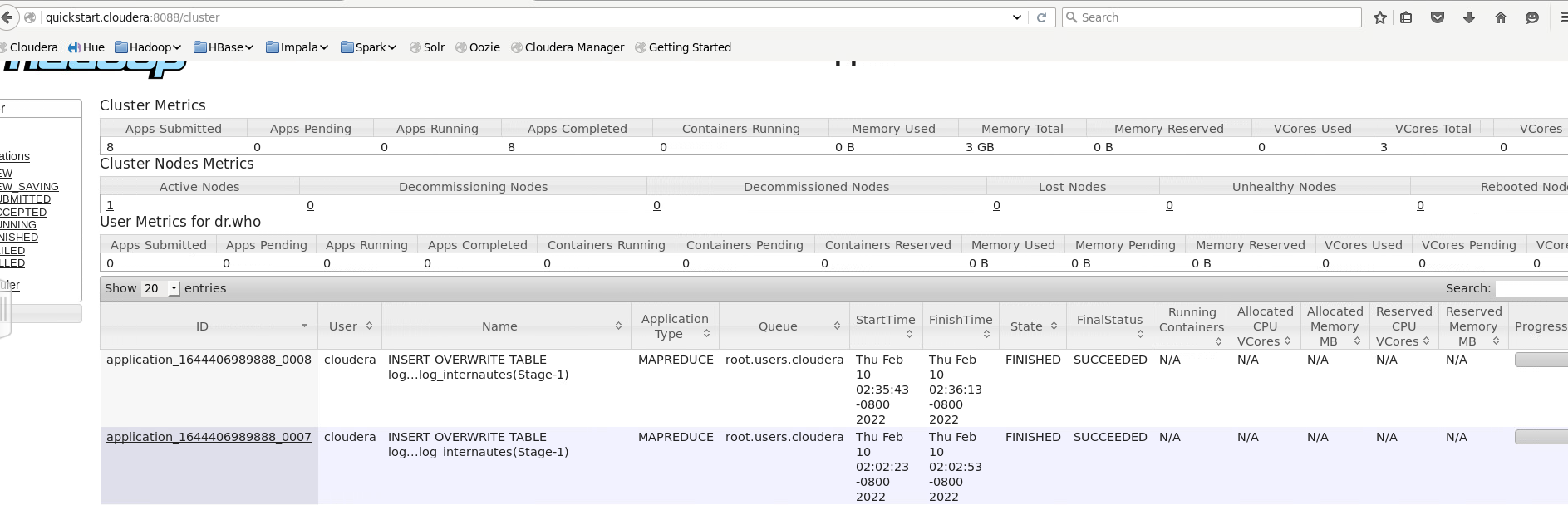
SET hive.exec.dynamic.partition=true;

SET hive.exec.dynamic.partition.mode=nonstrict;

SET hive.conf.validation=false;

SET hive.enforce.bucketing=true;

Vous pouvez visualiser aussi les jobs MapReduce qui créer ces charges de travail sur la page Web de Yarn (quickstart.cloudera :8088):



### Analyse de données (Manipulation des données (Tri, Agrégat, jointure, vue, …)

A ce stade, vous avez utilisé le langage de définition et de manipulation de données (DDL et DML) avec HiveQL et dans cette deuxième partie, vous allez faire de l’analyse de données avec HiveQL, à savoir le requetage de données.

Vous allez travailler avec trois fichiers différents employes.txt, fonction\_employes.txt et salaire\_employes.txt. Toutes les requêtes peuvent être faites sur Hue.

1. Créez une nouvelle base de donnée nommée analyse\_salaires.

Réponse

CREATE DATABASE analyse\_salaires;

USE analyse\_salaires;

1. Créer ensuite trois tables externes qui devront intégrées les données des trois fichiers. N’oubliez pas de créer les dossiers dans HDFS (/user/hive/data/salaire, /user/hive/data/fonction, /user/hive/data/employes) puis copier les trois fichiers de données dans les dossiers respectifs.

-- Assurez-vous de définir les autorisations correctes sur les répertoires HDFS

hdfs dfs -mkdir /user/hive/data/employes

hdfs dfs -mkdir /user/hive/data/salaire

hdfs dfs -mkdir /user/hive/data/fonction

hdfs dfs -chmod 777 /user/hive/data/employes

hdfs dfs -chmod 777 /user/hive/data/salaire

hdfs dfs -chmod 777 /user/hive/data/fonction

-- Copier les fichiers dans hdfs :

hadoop fs -copyFromLocal /home/cloudera/employes1.csv /user/hive/data/employes/

hadoop fs -copyFromLocal /home/cloudera/salaire\_employes.csv /user/hive/data/salaire/

hadoop fs -copyFromLocal /home/cloudera/fonction\_employes.csv /user/hive/data/fonction/

-- Création des tables externes :

CREATE EXTERNAL TABLE salaire\_employes (

sexe STRING,

secteur STRING,

salaire STRING,

code\_postal INT,

id\_empl INT

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE

LOCATION '/user/hive/data/salaire'

TBLPROPERTIES ("skip.header.line.count"="1");

CREATE EXTERNAL TABLE fonction\_employes (

id\_empl INT,

annee\_embauche INT,

fonction STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE

LOCATION '/user/hive/data/fonction'

TBLPROPERTIES ("skip.header.line.count"="1");

CREATE EXTERNAL TABLE employes (

id\_empl INT,

nom STRING,

prenom STRING,

date\_naissance STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ';'

STORED AS TEXTFILE

LOCATION '/user/hive/data/employes';

1. Combien d’employés ont été embauchés après 2019 ?

Réponse :

SELECT count(id\_empl) AS nombre\_employes FROM fonction\_employes WHERE annee\_embauche > '2019';

1. Quelles paires d’employés-secteurs ont un salaire supérieur à 35000 euro/année ?

Réponse :

SELECT EM.nom, EM.prenom, ES.secteur

FROM employes EM

JOIN salaire\_employes ES ON EM.id\_empl = ES.id\_empl

WHERE CAST(SUBSTR(ES.salaire, 1, 5) AS INT) > 35000;

1. Lister les employés du secteur éducation recrutés après l’année 2010.

Indication : créez une vue pour stocker la requête de jointure entre les deux tables employes et fonction\_employes avec la condition sur l’année 2010.

Réponse :

-- Créez une vue pour stocker la requête de jointure avec la condition sur l'année 2010

CREATE VIEW vue1 AS

SELECT

employes.id\_empl AS empID,

employes.nom AS empNOM,

employes.prenom AS empPRENOM,

fonction\_employes.annee\_embauche AS anneeRecrutement,

fonction\_employes.id\_empl AS empID\_fonction

FROM employes

JOIN fonction\_employes ON employes.id\_empl = fonction\_employes.id\_empl

WHERE fonction\_employes.annee\_embauche > '2010';

-- Sélectionnez les employés du secteur de l'éducation

SELECT empID, empNOM, empPRENOM

FROM vue1 v1

JOIN salaire\_employes SE ON v1.empID = SE.id\_empl

WHERE SE.secteur = 'Education';

1. Quel employé a le plus d’expérience professionnelle ?

Réponse

SELECT e1.id\_empl, e1.nom, e1.prenom, CAST(YEAR(CURRENT\_DATE()) AS INT) - f1.annee\_embauche AS experience

FROM employes e1

JOIN fonction\_employes f1 ON e1.id\_empl = f1.id\_empl

ORDER BY experience DESC

LIMIT 1;

1. Quelle est le salaire moyen des employés ?

Réponse:

SELECT AVG(CAST(SUBSTRING(salaire\_employes.salaire, 1, LENGTH(salaire\_employes.salaire) - 1) AS DECIMAL(10, 2))) AS salaire\_moyen

FROM salaire\_employes;

1. Quelle est l’ensemble des fonctions des employées femmes par année d’embauche ?

Réponse :

SELECT fe.id\_empl, fe.fonction, fe.annee\_embauche from fonction\_employes fe join salaire\_employes se on (fe.id\_empl=se.id\_empl) and se.sexe='F' order by fe.annee\_embauche DESC;

1. Lister les paires <prénom, fonction> des employés qui résident dans le département de paris ?

Réponse :

-- Créez une vue pour stocker les informations des employés et de leurs fonctions

CREATE VIEW vue4 AS

SELECT

employes.id\_empl AS empID,

employes.nom AS empNOM,

employes.prenom AS empPRENOM,

fonction\_employes.fonction AS empFONCTION

FROM employes

JOIN fonction\_employes ON employes.id\_empl = fonction\_employes.id\_empl;

-- Sélectionnez les paires <prénom, fonction> des employés résidant à Paris (code postal 75000)

SELECT empNOM, empPRENOM, empFONCTION

FROM vue4 v4

JOIN salaire\_employes se ON v4.empID = se.id\_empl

WHERE se.code\_postal = 75000;

1. Y-t-il une différence entre le salaire des employés hommes et femmes ?

Indication : calculez le ratio des femmes et des hommes par rapport à la somme des salaires/par la somme des expériences professionnelles, et comparez les deux ratios.

Réponse :

select sum(substr(se.salaire, 1, 5))/sum((cast(year(current\_date()) as int)-fe.annee\_embauche)) FROM salaire\_employes se join fonction\_employes fe on (se.id\_empl=fe.id\_empl) and se.sexe='F';

select sum(substr(se.salaire, 1, 5))/sum((cast(year(current\_date()) as int)-fe.annee\_embauche)) FROM salaire\_employes se join fonction\_employes fe on (se.id\_empl=fe.id\_empl) and se.sexe='M'

On peut l’optimiser cette requête comme suit :

SELECT

SUM(

CASE

WHEN se.sexe = 'F' THEN CAST(SUBSTRING(se.salaire, 1, 5) AS DECIMAL(10, 2))

ELSE 0

END

) / SUM(

CASE

WHEN se.sexe = 'F' THEN CAST(YEAR(CURRENT\_DATE()) - fe.annee\_embauche AS DECIMAL(10, 2))

ELSE 0

END

) AS ratio\_femmes,

SUM(

CASE

WHEN se.sexe = 'M' THEN CAST(SUBSTRING(se.salaire, 1, 5) AS DECIMAL(10, 2))

ELSE 0

END

) / SUM(

CASE

WHEN se.sexe = 'M' THEN CAST(YEAR(CURRENT\_DATE()) - fe.annee\_embauche AS DECIMAL(10, 2))

ELSE 0

END

) AS ratio\_hommes

FROM salaire\_employes se

JOIN fonction\_employes fe ON se.id\_empl = fe.id\_empl;

On calcule le ratio des salaires des femmes et des hommes par rapport à la somme des expériences professionnelles pour chaque groupe, en utilisant la clause "CASE" pour différencier les sexes. Les ratios des femmes et des hommes seront calculés dans le même résultat.

1. Le nombre d’années d’expérience influence-t-il le salaire des employés ? Indication : calculez le coefficient de corrélation avec la fonction corr() pour connaitre la relation de dépendance entre le nombre d’expérience et le salaires des employés.

Réponse:

select corr(cast(year(current\_date()) as int)-fe.annee\_embauche,cast(substr(se.salaire, 1, 5) as int)) from salaire\_employes se join fonction\_employes fe on (se.id\_empl=fe.id\_empl);

On pouvait la reformuler autrement :

SELECT

corr(

CAST(YEAR(CURRENT\_DATE()) - fe.annee\_embauche AS INT),

CAST(SUBSTRING(se.salaire, 1, 5) AS INT)

) AS coefficient\_correlation

FROM salaire\_employes se

JOIN fonction\_employes fe ON se.id\_empl = fe.id\_empl;

1. Le secteur d’activité influence-t-il le salaire des employés ? Conclure ? Indication : utiliser la solution de la question 10.

Réponse

create view vue12 as

select sum(substr(se.salaire, 1, 5)) / sum(cast(year(current\_date()) as int)-fe.annee\_embauche) as ratio, se.secteur as secteur from

fonction\_employes fe join salaire\_employes se on (fe.id\_empl=se.id\_empl) group by se.secteur;

select vue12.ratio, vue12.secteur from vue12;

Encore mieux (en une seule requête !):

SELECT AVG(CAST(SUBSTRING(salaire, 1, 5) AS DECIMAL(10, 2))) AS salaire\_moyen, secteur

FROM salaire\_employes

GROUP BY secteur;

Les salaires varient effectivement selon le secteur d'activité et le nombre d’expérience. Les résultats obtenus reflètent bien le dataset que nous avons en possession.