
# Mini projet HADOOP : Traitement des données de movies

**Réalisé par :**

**Lotfi HAMICHE**

**Ferroudja DJELLALI**

# Introduction

Dans ce projet, nous allons travailler sur des données relationnelles relatives aux films et series.

Nous allons utiliser trois sources de fichiers d'extension ".dat" dans le but de mettre en pratique les technologies de la Big Data et les modules de Hadoop vus en cours.

## DATASET

- Movies.dat:

Les informations sur les films sont contenues dans le fichier movies.dat. Chaque ligne de ce fichier représente un film, et a le format suivant :

MovieID::Titre::Genres

MovieID est l'identifiant réel de de la table movies.

- Users.dat:

Les informations sur les utilisateurs se trouvent dans le fichier "users.dat" et ont le format suivant :

**UserID::Gender::Age::Occupation::Zip-code**

Toutes les informations démographiques sont fournies volontairement par les utilisateurs et leur exactitude n'est pas vérifiée. Seuls les utilisateurs qui ont fourni certaines informations démographiques sont inclus dans cet ensemble de données.

- **Le Gender** est indiqué par un "M" pour homme et un "F" pour femme

- **L'âge** est choisi parmi les plages suivantes :

1 : "Moins de 18 ans"

18 : "18-24"

25 : "25-34"

35 : "35-44"

45 : "45-49"

50 : "50-55"

56 : "56+"

- **L'occupation:** profession de l'utilisateur est choisie parmi les choix suivants

0 : "autre" ou non spécifié

1 : "universitaire/éducateur

2 : "artiste

3 : "employé de bureau/administrateur".

4 : "université/étudiant de troisième cycle

5 : "service clientèle

6 : "médecin/soins de santé

7 : "cadre supérieur/management

8 : "agriculteur

9 : "femme au foyer

10 : "étudiant de la maternelle à la 12e année

11 : "avocat

12 : "programmeur

13 : "retraité

14 : "vente/marketing

15 : "scientifique

16 : "indépendant

17 : "technicien/ingénieur

18 : " commerçant/artisan ".

19 : "chômeur

20 : " écrivain ".

- Ratings.dat:

Toutes les évaluations sont contenues dans le fichier ratings.dat. Chaque ligne de ce fichier représente une évaluation d'un film par un utilisateur, et a le format suivant :
**UserID::MovieID::Rating::Timestamp**

Les lignes de ce fichier sont d'abord classées par **UserID,** puis, au sein de l'utilisateur, par **MovieID. Les notes** (rating): sont attribuées sur une échelle de 5 étoiles, avec des incréments d'une demi-étoile.

**La colonne Timestamp** représentent les dates pendant lesquelles l'utilisateur a noté un film.

## SOURCE DU DATASET

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/NRDJCB

# Commandes de base avec HDFS

**Objectif :** "Dans cette partie nous allons manipuler quelques commandes de bases du système de fichiers distribués de Hadoop (HDFS) notemment pour importer les fichiers de données, les afficher, créer les repertoires de travail, …

Nous allons traiter tout au long du projet les données que nous aurions importés avec les commandes de cette partie ."

1. Upload les fichiers de données movies.csv et ratings.csv depuis notre machine vers le compte student de la machine virtuelle

| |
|---|
| Depuis le **terminal** linux (ou **cmd** sous Windows) taper :<br>scp -P 2222 movies.csv student@20.101.89.109:movies.dat<br>scp -P 2222 ratings.csv student@20.101.89.109: ratings.dat<br>scp -P 2222 ratings.csv student@20.101.89.109: users.dat |

2. Créer un répertoire HDFS MiniProject :

hadoop fs -mkdir /user/student/ MiniProject

**Erreur**:

```
[root@sandbox-hdp ~]# hadoop fs -mkdir /user/student/MiniProject
mkdir: Cannot create directory /user/student/MiniProject. Name node is in safe mode.
```

**Solution proposée** : sudo –u hdfs dfsadmin -safemode leave

3. Mettre les fichiers chargés dans le repertoire MiniProject:

| |
|---|
| hadoop fs -put /home/student/movies.dat /user/student/data/MiniProject/<br>hadoop fs -put /home/student/users.dat /user/student/data/MiniProject/<br>hadoop fs -put /home/student/ratings.dat /user/student/data/MiniProject/<br><br>NB: ces trois commandes peuvent être remplacées par la commande<br>**hadoop fs -put /home/student/*.dat /user/student/data/MiniProject/** |

4. Afficher les informations à propos des fichiers que nous avons chargé :

hadoop fs -ls /user/student/MiniProject /movies.dat

**-rw-r--r-- 1 student student 171308 2022-02-04 23:48 /user/student/MiniProject/movies.dat**

hadoop fs -ls /user/student/MiniProject /ratings.dat

**-rw-r--r-- 1 student student 24594131 2022-02-04 23:48 /user/student/MiniProject/ratings.dat**

hadoop fs -ls /user/student/MiniProject /users.dat

**-rw-r--r-- 1 student student 134368 2022-02-04 23:49 /user/student/MiniProject/users.dat**

5. Déterminer la taille des fichiers HDFS :

hadoop fs –du –h /user/student/MiniProject /movies.dat

```
[student@sandbox-hdp ~]$ hadoop fs -du -h /user/student/MiniProject/movies.dat
167.3 K  /user/student/MiniProject/movies.dat
```

hadoop fs -du –h /user/student/MiniProject /ratings.dat

```
[student@sandbox-hdp ~]$ hadoop fs -du -h /user/student/MiniProject/ratings.dat
23.5 M  /user/student/MiniProject/ratings.dat
```

hadoop fs –du –h /user/student/MiniProject /users.dat

```
[student@sandbox-hdp ~]$ hadoop fs -du -h /user/student/MiniProject/users.dat
131.2 K  /user/student/MiniProject/users.dat
```

6. Déterminer le nombre de ligne dans chaque fichier HDFS

hadoop fs -cat /user/student/MiniProject /movies.dat | wc –l

**9743**

hadoop fs -cat /user/student/MiniProject /ratings.dat | wc –l

**100837**

hadoop fs -cat /user/student/MiniProject /users.dat | wc –l

**6040**

7. Exploration des 10 premières lignes du dataset ratings.dat :

```
[student@sandbox-hdp ~]$ cat ratings.dat | head -n 10
1::1193::5::978300760
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
1::594::4::978302268
1::919::4::978301368
[student@sandbox-hdp ~]$
```

# Traitements avec des jobs MapReduce en python

**Objectif :** "Dans cette partie nous allons manipuler quelques commandes de bases du système de fichiers distribués de Hadoop (HDFS) notamment pour importer les fichiers de données, les afficher, créer les répertoires de travail, …

Pour cela, nous allons utiliser le fichier ratings.dat :

Nous avons un fichier **mapNbRatings.py** dans lequel nous avons déclaré la fonction qui permet de créer une paire : clé, valeur (correspondant à film : nombre d'utilisateurs qui ont noté ce film) et un fichier **reduceNbRatings.py** "

1. Donner les droits d'exécution aux scripts python que nous venons de créer

```
Chmod u+x mapNbRatings.py
Chmod u+x reduceNbRatings.py
```

2. Exécuter le job MapReduce dans hadoop à la base des deux scripts pythons map et reduce

Hadoop jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-streaming.jar -mapper mapNbRatings.py -reducer reduceNbRatings.py -input /user/student/MiniProject -output test-dir -file reduceNbRatings.py -file mapNbRatings.py

3. Vérifier si le dossier **test-dir** est bien créé et afficher son contenu

Hadoop fs –ls test-dir

```
[student@sandbox-hdp ~]$ hadoop fs -ls test-dir
Found 2 items
-rw-r--r--   1 student student          0 2022-02-05 01:54 test-dir/_SUCCESS
-rw-r--r--   1 student student         44 2022-02-05 01:54 test-dir/part-00000
```

4. Visualiser le contenu du résultat de l'exécution des scripts MapReduce (généré automatiquement)

Hadoop fs –cat test-dir/part-00000

```
[student@sandbox-hdp ~]$ hadoop fs -cat test-dir/part-00000
1       56174
3       261197
2       107557
5       226310
4       348971
```

**Interprétation de ces résultats**

Le nombre d'utilisateurs qui ont donné la note **'1'** à un film est : 56174

Le nombre d'utilisateurs qui ont donné la note **'3'** à un film est : 261197

...

# Partie avec PySpark

## Tâche 1 : Calculer le nombre de valeurs par rating

**Objectif :** ”Dans cette partie, nous allons effectuer quelques taches pour traiter les données avec pyspark, vous allez retrouver l’application PySpark dans les fichiers script de ce rendu“

Dans cette tâche, nous allons calculer le nombre d’utilisateurs ayant noté un film avec une certaine note, nous allons avoir à la fin un dictionnaire qui contient les notes comme clés et le nombre de fois que cette note se répète dans le dataset comme valeur.

Ex : note 5 : 100, note 4 : 20, ….

Nous allons procéder comme suit

1. Lecture du fichier “**ratings.dat** “

```
dataRatings =sc.textFile("/user/student/MiniProject/ratings.dat")
```

- Décomposer lexicalement les lignes en colonnes par le séparateur “::”

```
>>> dataRatingSplitted = dataRatings.map(lambda line: line.split("::"))
```

1. Afficher les 5 premières lignes :

**dataRatingSplitted.take(5)**

```
[[u'1', u'1193', u'5', u'978300760'],
[u'1', u'661', u'3', u'978302109'],
[u'1', u'914', u'3', u'978301968'],
[u'1', u'3408', u'4', u'978300275'],
[u'1', u'2355', u'5', u'978824291'],
```

- Effectuer un mapping de la colonne 2 (3ème colonne (0..2), avec une émission de valeur 1 pour chaque ligne : (“5”,1), (“4”,1), (“3”,1), (“2”,1), (“1”,1)

```
rc = dataRatingSplitted.map(lambda line: (line[2],1))
```

result:

```
[(u'5', 1), (u'3', 1)]
```

- Effectuer un Reduce Sommant les valeurs (1) émises par le Map en regroupant par clef

```
>>> rc_reduce = rc.reduceByKey(lambda a,b: a+b)
```

- Afficher quelques lignes de résultat :

```
rc_reduce.take(5)
```

```
[(u'1', 56174),
(u'2', 107557),
(u'3', 261197),
(u'4', 348971),
(u'5', 226310)]
```

## Tache 2 : Calculer le nombre d'utilisateurs ayant noté un film

De la même manière, nous allons utiliser le fichier rating.dat

Nous allons maintenant grouper selon la colonne 2 (1) qui correspond à la colonne movieID

Aperçu (5 premières colonnes) :

```
[(u'3724', 195), (u'344', 766), (u'346', 120), (u'340', 84), (u'342', 518)]
```

Interprétation :

Le film ayant l'ID 3724 a eu 195 notes

766 utilisateurs ont noté le film avec l'ID 344

**Remarque :**

Nous avons créé l'application PySpark dans le fichier ratingsPySpark.py,

Ou vous allez retrouver toutes les commandes que nous avons exécuté précédemment.

L'application est ensuite soumise en mode cluster :

**spark-submit --master yarn --deploy-mode cluster ratingsPySpark.py**

Le résultat est stocké dans le fichier **hdfs:///user/student/output-RalingsVaue/**

# Partie avec HIVE

# Objectif : " Dans cette partie, nous allons utiliser le module Hive pour exploiter les données d'utilisateurs, de films et de notes de films par utilisateur."

Dans un premier temps, nous allons créer une base de données qui contiens trois tables (users, ratings et movies) dans lesquelles nous allons chargerles données provenant de trois fichiers source (extension .dat)

1. **Création de la base de données :**

```
CREATE DATABASE db_miniproject;
```

2. **Utiliser la base de données créée auparavant :**

```
Use db_miniproject;
```

3. **Création et alimentation des tables :**

Nous avons utilisé les expressions régulières afin de séparer les champs de nos **fichiers .dat** ou le séparateur était **'::'**

- **Table movies**:

```
CREATE TABLE movies (
MovieID STRING,
Title STRING,
Genres STRING) ROW FORMAT SERDE
'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES ('input.regex' = '(.*?)::(.*?)::(.*)')
stored as textfile;

LOAD DATA INPATH '/user/student/data/miniprojet/movies.dat' OVERWRITE INTO TABLE movies;
```

- **Table users** :

```
CREATE TABLE users (
UserID INT,
Gender STRING,
Age INT,
Occupation INT,
Zip_code STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES ('input.regex' = '(.*?)::(.*?)::(.*?)::(.*?)::(.*)') stored as textfile;
LOAD DATA INPATH '/user/student/data/miniprojet/users.dat' OVERWRITE INTO TABLE users;
```

- **Table ratings** :

```
CREATE TABLE ratings (
UserID INT,
MovieID INT,
Rating INT,
TimesRating INT)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES ('input.regex' = '(.*?)::(.*?)::(.*?)::(.*)')
stored as textfile
LOCATION '/miniprojet';

LOAD DATA INPATH '/user/student/data/miniprojet/ratings.dat' OVERWRITE INTO TABLE ratings;
```

4. Afficher la structure de la table movies :

```
0: jdbc:hive2://> DESCRIBE MOVIES;
OK
+-----------+------------+----------+--+
| col_name  | data_type  | comment  |
+-----------+------------+----------+--+
| movieid   | string     |          |
| title     | string     |          |
| genres    | string     |          |
+-----------+------------+----------+--+
3 rows selected (0.559 seconds)
```

5. Afficher les 10 premières lignes de la table movies :

```
0: jdbc:hive2://> SELECT * FROM MOVIES LIMIT 10;
OK
+-----------------+-----------------------------------+-------------------------------+--+
| movies.movieid  |           movies.title            |         movies.genres         |
+-----------------+-----------------------------------+-------------------------------+--+
| 1               | Toy Story (1995)                  | Animation|Children's|Comedy   |
| 2               | Jumanji (1995)                    | Adventure|Children's|Fantasy  |
| 3               | Grumpier Old Men (1995)           | Comedy|Romance                |
| 4               | Waiting to Exhale (1995)          | Comedy|Drama                  |
| 5               | Father of the Bride Part II (1995)| Comedy                        |
| 6               | Heat (1995)                       | Action|Crime|Thriller         |
| 7               | Sabrina (1995)                    | Comedy|Romance                |
| 8               | Tom and Huck (1995)               | Adventure|Children's           |
| 9               | Sudden Death (1995)               | Action                        |
| 10              | GoldenEye (1995)                  | Action|Adventure|Thriller     |
+-----------------+-----------------------------------+-------------------------------+--+
10 rows selected (0.172 seconds)
```

6. Afficher la structure de la table users :

```
0: jdbc:hive2://> DESCRIBE USERS;
OK
+-------------+------------+----------+--+
|  col_name   | data_type  | comment  |
+-------------+------------+----------+--+
| userid      | int        |          |
| gender      | string     |          |
| age         | int        |          |
| occupation  | int        |          |
| zip_code    | string     |          |
+-------------+------------+----------+--+
5 rows selected (0.493 seconds)
```

7. Afficher les 10 premières lignes de la table users :

```
0: jdbc:hive2://> SELECT * FROM USERS LIMIT 10;
OK
+---------------+---------------+------------+-------------------+-----------------+--+
| users.userid  | users.gender  | users.age  | users.occupation  | users.zip_code  |
+---------------+---------------+------------+-------------------+-----------------+--+
| 1             | F             | 1          | 10                | 48067           |
| 2             | M             | 56         | 16                | 70072           |
| 3             | M             | 25         | 15                | 55117           |
| 4             | M             | 45         | 7                 | 02460           |
| 5             | M             | 25         | 20                | 55455           |
| 6             | F             | 50         | 9                 | 55117           |
| 7             | M             | 35         | 1                 | 06810           |
| 8             | M             | 25         | 12                | 11413           |
| 9             | M             | 25         | 17                | 61614           |
| 10            | F             | 35         | 1                 | 95370           |
+---------------+---------------+------------+-------------------+-----------------+--+
10 rows selected (0.179 seconds)
```

8. Afficher la structure de la table ratings

```
0: jdbc:hive2://> DESCRIBE RATINGS;
OK
+--------------+------------+----------+--+
|   col_name   | data_type  | comment  |
+--------------+------------+----------+--+
| userid       | int        |          |
| movieid      | int        |          |
| rating       | int        |          |
| timesrating  | int        |          |
+--------------+------------+----------+--+
4 rows selected (0.518 seconds)
```

9. Afficher les 10 premières lignes de la table ratings :

```
0: jdbc:hive2://> SELECT * FROM RATINGS LIMIT 10;
OK
+-----------------+------------------+-----------------+----------------------+--+
| ratings.userid  | ratings.movieid  | ratings.rating  | ratings.timesrating  |
+-----------------+------------------+-----------------+----------------------+--+
| 1               | 1193             | 5               | 978300760            |
| 1               | 661              | 3               | 978302109            |
| 1               | 914              | 3               | 978301968            |
| 1               | 3408             | 4               | 978300275            |
| 1               | 2355             | 5               | 978824291            |
| 1               | 1197             | 3               | 978302268            |
| 1               | 1287             | 5               | 978302039            |
| 1               | 2804             | 5               | 978300719            |
| 1               | 594              | 4               | 978302268            |
| 1               | 919              | 4               | 978301368            |
+-----------------+------------------+-----------------+----------------------+--+
10 rows selected (0.163 seconds)
```

10. Afficher la moyenne des notes de chaque film (affichage limité à 10 lignes)

```
0: jdbc:hive2://> SELECT RATINGS.MOVIEID,AVG(RATING) AVGRATING
FROM RATINGS INNER JOIN MOVIES ON MOVIES.MOVIEID=RATINGS.MOVIEID
WHERE (INSTR(MOVIES.GENRES,"Action")>0)
GROUP BY RATINGS.MOVIEID
ORDER BY AVGRATING DESC
LIMIT 10;

OK
+------------------+---------------------+--+
| ratings.movieid  |      avgrating      |
+------------------+---------------------+--+
| 2905             | 4.608695652173913   |
| 2019             | 4.560509554140127   |
| 858              | 4.524966261808367   |
| 1198             | 4.477724741447892   |
| 260              | 4.453694416583082   |
| 1221             | 4.357565011820331   |
| 2028             | 4.337353938937053   |
| 2571             | 4.315830115830116   |
| 1197             | 4.3037100949094045  |
| 1233             | 4.302697302697303   |
+------------------+---------------------+--+
10 rows selected (22.082 seconds)
```

11. Afficher les titres et le genre des films notés par les hommes qui ont une note moyenne entre 4.4 et 4.7, et qui ont au moins le  genre Drama ou Action :

```
0: jdbc:hive2://>SELECT MOVIES.MOVIEID, MOVIES.TITLE, MOVIES.GENRES, AVG(RATINGS.RATING) AS AVGRATINGS
FROM USERS JOIN RATINGS ON USERS.USERID = RATINGS.USERID
JOIN MOVIES ON MOVIES.MOVIEID = RATINGS.MOVIEID
WHERE USERS.GENDER = "M" AND (INSTR(MOVIES.GENRES,"Action")>0 OR INSTR(MOVIES.GENRES,"Drama")>0)
GROUP BY MOVIES.MOVIEID,MOVIES.TITLE,MOVIES.GENRES
HAVING  AVG(RATING)  >= 4.4 AND  AVG(RATING)  <= 4.7;

OK
+----------------+-----------------------------------------------------------------+-----------------------------+--------------------+--+
| movies.movieid |                          movies.title                           |        movies.genres        |     avgratings     |
+----------------+-----------------------------------------------------------------+-----------------------------+--------------------+--+
| 1178           | Paths of Glory (1957)                                           | Drama|War                  | 4.485148514851486  |
| 1193           | One Flew Over the Cuckoo's Nest (1975)                          | Drama                       | 4.418423106947697  |
| 1198           | Raiders of the Lost Ark (1981)                                  | Action|Adventure           | 4.520597322348094  |
| 1221           | Godfather: Part II, The (1974)                                  | Action|Crime|Drama          | 4.437777777777778  |
| 1250           | Bridge on the River Kwai, The (1957)                            | Drama|War                   | 4.401505646173149  |
| 1664           | Ni¿½nette et Boni (1996)                                        | Drama                       | 4.5                |
| 1795           | Calle¡¿½n de los milagros, El (1995)                            | Drama                       | 4.5                |
| 2019           | Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954) | Action|Drama           | 4.576628352490421  |
| 2480           | Dry Cleaning (Nettoyage ¡¿½ sec) (1997)                         | Drama                       | 4.5                |
| 2503           | Apple, The (Sib) (1998)                                         | Drama                       | 4.6                |
| 260            | Star Wars: Episode IV - A New Hope (1977)                       | Action|Adventure|Fantasy|Sci-Fi | 4.495307167235495  |
| 2905           | Sanjuro (1962)                                                  | Action|Adventure           | 4.639344262295082  |
| 3030           | Yojimbo (1961)                                                  | Comedy|Drama|Western        | 4.402116402116402  |
| 318            | Shawshank Redemption, The (1994)                                | Drama                       | 4.560625           |
| 3410           | Soft Fruit (1999)                                               | Comedy|Drama                | 4.4                |
| 527            | Schindler's List (1993)                                         | Drama|War                   | 4.49141503848431   |
| 53             | Lamerica (1994)                                                 | Drama                       | 4.666666666666667  |
| 557            | Mamma Roma (1962)                                               | Drama                       | 4.5                |
| 578            | Hour of the Pig, The (1993)                                     | Drama|Mystery               | 4.5                |
| 858            | Godfather, The (1972)                                           | Action|Crime|Drama          | 4.583333333333333  |
| 912            | Casablanca (1942)                                               | Drama|Romance|War           | 4.461340206185567  |
| 923            | Citizen Kane (1941)                                             | Drama                       | 4.407894736842105  |
+----------------+-----------------------------------------------------------------+-----------------------------+--------------------+--+
22 rows selected (20.348 seconds)
```

12. Compter le nombre de films par genre :

```
0: jdbc:hive2://> SELECT Genres, COUNT(Genres) AS Nb_Genres FROM  movies GROUP BY Genres ORDER BY
Genres ASC LIMIT 10;
Query ID = student_20220212164722_74f7ec7b-f5f5-4042-9e26-2ee048ae3e56
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1644671321892_0014)

OK
+-----------------------------------------------+------------+--+
|                    genres                     | nb_genres  |
+-----------------------------------------------+------------+--+
| Action                                        | 65         |
| Action|Adventure                              | 25         |
| Action|Adventure|Animation                     | 1          |
| Action|Adventure|Animation|Children's|Fantasy  | 1          |
| Action|Adventure|Animation|Horror|Sci-Fi       | 1          |
| Action|Adventure|Children's                   | 1          |
| Action|Adventure|Children's|Comedy            | 2          |
| Action|Adventure|Children's|Fantasy           | 1          |
| Action|Adventure|Children's|Sci-Fi            | 1          |
| Action|Adventure|Comedy                       | 5          |
+-----------------------------------------------+------------+--+
10 rows selected (2.246 seconds)
```

13. Afficher le nombre de films par genre :

```
0: jdbc:hive2://> SELECT Title,Genres FROM ratings R INNER JOIN movies M ON R.MovieID=M. MovieID
WHERE R.Rating=5 LIMIT 10;
Query ID = student_20220212164845_7d6c3009-9eab-413d-a44f-782ea43195f2
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1644671321892_0014)

OK
+-----------------------------------------+-----------------------------------------+--+
|                  title                  |                 genres                  |
+-----------------------------------------+-----------------------------------------+--+
| One Flew Over the Cuckoo's Nest (1975)  | Drama                                   |
| Bug's Life, A (1998)                    | Animation|Children's|Comedy             |
| Ben-Hur (1959)                          | Action|Adventure|Drama                   |
| Christmas Story, A (1983)               | Comedy|Drama                            |
| Beauty and the Beast (1991)             | Animation|Children's|Musical            |
| Sound of Music, The (1965)              | Musical                                 |
| Awakenings (1990)                       | Drama                                   |
| Back to the Future (1985)               | Comedy|Sci-Fi                           |
| Schindler's List (1993)                 | Drama|War                               |
| Pocahontas (1995)                       | Animation|Children's|Musical|Romance   |
+-----------------------------------------+-----------------------------------------+--+
10 rows selected (11.056 seconds)
```

14. Afficher le nombre de films notés par année :

```
0: jdbc:hive2://> SELECT FROM_UNIXTIME(TIMESRATING,'YYYY') AS DATE_RATINGS,COUNT(RATING) AS
NB_RATINGS FROM RATINGS GROUP BY FROM_UNIXTIME(TIMESRATING,'YYYY') ORDER BY
DATE_RATINGS LIMIT 10;
Query ID = student_20220212165112_13b385f1-d7e4-47e2-b16e-e262adff7e07
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1644671321892_0014)

OK
+---------------+-------------+--+
| date_ratings  | nb_ratings  |
+---------------+-------------+--+
| 2000          | 902585      |
| 2001          | 69774       |
| 2002          | 24361       |
| 2003          | 3489        |
+---------------+-------------+--+
4 rows selected (11.401 seconds)
```

15. Créer une table partitionnée movies_partitioned dans Hive :

```
DROP TABLE IF EXISTS MOVIES_PARTITIONED;
CREATE TABLE IF NOT EXISTS
MOVIES_PARTITIONED(MOVIEID STRING,TITLE STRING,GENRES
STRING)PARTITIONED BY (YEARRATING INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '::'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE ;
set hive.exec.dynamic.partition.mode=nonstrict
```

16. Inserer et afficher dans la table partitionnée movies_partitioned les notes de l'année 2000 :

```
INSERT INTO MOVIES_PARTITIONED
PARTITION (YEARRATING = 2000)
SELECT RATINGS.MOVIEID, MOVIES.TITLE, MOVIES.GENRES
FROM RATINGS INNER JOIN MOVIES ON RATINGS.MOVIEID = MOVIES.MOVIEID
WHERE FROM_UNIXTIME(TIMESRATING,'YYYY') = 2000;


0: jdbc:hive2://> SELECT * FROM MOVIES_PARTITIONED WHERE YEARRATING=2000 LIMIT 10;
OK
+-----------------------------+-----------------------------------+----------------------------+-------------------------------+--+
| movies_partitioned.movieid  |       movies_partitioned.title    | movies_partitioned.genres  | movies_partitioned.yearrating |
+-----------------------------+-----------------------------------+----------------------------+-------------------------------+--+
| 1188                        | Strictly Ballroom (1992)          | Comedy|Romance            | 2000                          |
| 587                         | Ghost (1990)                      | Comedy|Romance|Thriller   | 2000                          |
| 1265                        | Groundhog Day (1993)              | Comedy|Romance            | 2000                          |
| 2712                        | Eyes Wide Shut (1999)             | Drama                      | 2000                          |
| 1198                        | Raiders of the Lost Ark (1981)    | Action|Adventure          | 2000                          |
| 593                         | Silence of the Lambs, The (1991)  | Drama|Thriller             | 2000                          |
| 597                         | Pretty Woman (1990)               | Comedy|Romance            | 2000                          |
| 1923                        | There's Something About Mary (1998) | Comedy                   | 2000                          |
| 2580                        | Go (1999)                         | Crime                      | 2000                          |
| 1784                        | As Good As It Gets (1997)         | Comedy|Drama               | 2000                          |
+-----------------------------+-----------------------------------+----------------------------+-------------------------------+--+
10 rows selected (0.32 seconds)
```

17. Insérer et afficher dans la table partitionnée movies_partitioned les notes de l'année 2001 :

```
INSERT INTO MOVIES_PARTITIONED
PARTITION (YEARRATING = 2001)
SELECT RATINGS.MOVIEID, MOVIES.TITLE, MOVIES.GENRES
FROM RATINGS INNER JOIN MOVIES ON RATINGS.MOVIEID = MOVIES.MOVIEID
WHERE FROM_UNIXTIME(TIMESRATING,'YYYY') = 2001;


0: jdbc:hive2://> SELECT * FROM MOVIES_PARTITIONED WHERE YEARRATING=2001 LIMIT 10;
OK
+-----------------------------+-----------------------------------+----------------------------+-------------------------------+--+
| movies_partitioned.movieid  |       movies_partitioned.title    | movies_partitioned.genres  | movies_partitioned.yearrating |
+-----------------------------+-----------------------------------+----------------------------+-------------------------------+--+
| 1193                        | One Flew Over the Cuckoo's Nest (1975) | Drama                 | 2001                          |
| 661                         | James and the Giant Peach (1996)  | Animation|Children's|Musical | 2001                          |
| 914                         | My Fair Lady (1964)               | Musical|Romance          | 2001                          |
| 3408                        | Erin Brockovich (2000)            | Drama                      | 2001                          |
| 2355                        | Bug's Life, A (1998)              | Animation|Children's|Comedy | 2001                          |
| 1197                        | Princess Bride, The (1987)        | Action|Adventure|Comedy|Romance | 2001                          |
| 1287                        | Ben-Hur (1959)                    | Action|Adventure|Drama      | 2001                          |
| 2804                        | Christmas Story, A (1983)         | Comedy|Drama               | 2001                          |
| 594                         | Snow White and the Seven Dwarfs (1937) | Animation|Children's|Musical | 2001                          |
| 919                         | Wizard of Oz, The (1939)          | Adventure|Children's|Drama|Musical | 2001                          |
+-----------------------------+-----------------------------------+----------------------------+-------------------------------+--+
10 rows selected (0.3 seconds)
```

18. Insérer et afficher dans la table partitionnée movies_partitioned les notes de l'année 2002 :

```
INSERT INTO MOVIES_PARTITIONED
PARTITION (YEARRATING = 2002)
SELECT RATINGS.MOVIEID, MOVIES.TITLE, MOVIES.GENRES
FROM RATINGS INNER JOIN MOVIES ON RATINGS.MOVIEID = MOVIES.MOVIEID
WHERE FROM_UNIXTIME(TIMESRATING,'YYYY') = 2002;
```

```
0: jdbc:hive2://> SELECT * FROM MOVIES_PARTITIONED WHERE YEARRATING=2002 LIMIT 10;
OK
22/02/12 19:05:16 [main]: WARN lazy.LazyStruct: Extra bytes detected at the end of the row! Ignoring similar problems.
+-----------------------------+-----------------------------+-----------------------------+-----------------------------+--+
| movies_partitioned.movieid  |   movies_partitioned.title  |    movies_partitioned.genres    | movies_partitioned.yearrating  |
+-----------------------------+-----------------------------+-----------------------------+-----------------------------+--+
| 3359                        | Breaking Away (1979)        | Drama                       | 2002                        |
| 581                         | Celluloid Closet, The (1995)| Documentary                 | 2002                        |
| 7                           | Sabrina (1995)              | Comedy|Romance              | 2002                        |
| 1191                        | Madonna                     |  Truth or Dare (1991)       | 2002                        |
| 1266                        | Unforgiven (1992)           | Western                     | 2002                        |
| 2713                        | Lake Placid (1999)          | Horror|Thriller            | 2002                        |
| 1912                        | Out of Sight (1998)         | Action|Crime|Romance       | 2002                        |
| 595                         | Beauty and the Beast (1991) | Animation|Children's|Musical | 2002                        |
| 247                         | Heavenly Creatures (1994)   | Drama|Fantasy|Romance|Thriller | 2002                        |
| 1283                        | High Noon (1952)            | Western                     | 2002                        |
+-----------------------------+-----------------------------+-----------------------------+-----------------------------+--+
10 rows selected (0.304 seconds)
```

19. Explorer la structure de stockage HDFS réalisé par Hive pour la table partitionnée movies_partitioned :

```
[student@sandbox-hdp ~]$ hadoop fs -ls /apps/hive/warehouse/db_miniprojet.db/movies_partitioned
Found 3 items
drwxrwxrwx   - student hadoop          0 2022-02-12 19:19
/apps/hive/warehouse/db_miniprojet.db/movies_partitioned/yearrating=2000
drwxrwxrwx   - student hadoop          0 2022-02-12 19:19
/apps/hive/warehouse/db_miniprojet.db/movies_partitioned/yearrating=2001
drwxrwxrwx   - student hadoop          0 2022-02-12 19:19
/apps/hive/warehouse/db_miniprojet.db/movies_partitioned/yearrating=2002
```

20. Explorer plus en profondeur la structure et le contenu du stockage réalisé par Hive pour la table partitionnée movies_partitioned sur HDFS:

```
[student@sandbox-hdp ~]$ hadoop fs -ls -R
/apps/hive/warehouse/db_miniprojet.db/movies_partitioned/yearrating\=2001/
-rwxrwxrwx   1 student hadoop    2508204 2022-02-12 18:57
/apps/hive/warehouse/db_miniprojet.db/movies_partitioned/yearrating=2001/000000_0
-rwxrwxrwx   1 student hadoop    2508204 2022-02-12 19:19
/apps/hive/warehouse/db_miniprojet.db/movies_partitioned/yearrating=2001/000000_0_copy_1
-rwxrwxrwx   1 student hadoop     521092 2022-02-12 18:57
/apps/hive/warehouse/db_miniprojet.db/movies_partitioned/yearrating=2001/000001_0
-rwxrwxrwx   1 student hadoop     521092 2022-02-12 19:19
/apps/hive/warehouse/db_miniprojet.db/movies_partitioned/yearrating=2001/000001_0_copy_1
[student@sandbox-hdp ~]$


[student@sandbox-hdp ~]$ hadoop fs -cat
/apps/hive/warehouse/db_miniprojet.db/movies_partitioned/yearrating\=2001/000000_0 | head -n 10
1193:One Flew Over the Cuckoo's Nest (1975):Drama
661:James and the Giant Peach (1996):Animation|Children's|Musical
914:My Fair Lady (1964):Musical|Romance
3408:Erin Brockovich (2000):Drama
2355:Bug's Life, A (1998):Animation|Children's|Comedy
1197:Princess Bride, The (1987):Action|Adventure|Comedy|Romance
1287:Ben-Hur (1959):Action|Adventure|Drama
2804:Christmas Story, A (1983):Comedy|Drama
594:Snow White and the Seven Dwarfs (1937):Animation|Children's|Musical
919:Wizard of Oz, The (1939):Adventure|Children's|Drama|Musical
cat: Unable to write to output stream.
```

# Partie avec PIG

**Objectif :** ”Dans cette partie nous avons utilisé le language **Apache PIG** donc pouvoir charger et manipuler des données avec **Pig** via **Hadoop** Command Line

(**Grunt Shell d'Apache Pig**). Comme **Pig** est adapté aux problématiques liées aux traitements **ETL** (Extract, Transform, Load) de gros volumes de données, nous avons la possibilité de traiter et stocker des données Hadoop structurées ou non-stucturé. Enfin voir les résultats des exécutions dans la file manger de **Web Ambari**"

1. Création et alimentation des tables :

- **Table movies**:

```
A = LOAD '/user/student/data/miniprojet/movies.dat' USING PigStorage('\n');
raw_movies = FOREACH A GENERATE FLATTEN(STRSPLIT($0, '::'));

movies_details = FOREACH raw_movies
GENERATE ((INT)$0) AS MovieID, $1 AS Title, $2 AS Genres;
```

- **Table users** :

```
A = LOAD '/user/student/data/miniprojet/users.dat' USING PigStorage('\n');
raw_users = FOREACH A GENERATE FLATTEN(STRSPLIT($0, '::'));

users_details = FOREACH raw_users
GENERATE ((INT)$0) AS UserID , $1 AS Gender , ((INT)$2) AS Age , ((INT)$3) AS Occupation , $4 AS
Zip_code ;
```

- **Table ratings** :

```
A = LOAD '/user/student/data/miniprojet/ratings.dat' USING PigStorage('\n');
raw_ratings = FOREACH A GENERATE FLATTEN(STRSPLIT($0, '::'));

ratings_details = FOREACH raw_ratings
GENERATE ((INT)$0) AS UserID , ((INT)$1) AS MovieID, ((INT)$2) AS Rating , ((INT)$3) AS
TimesRating ;
```

2. Afficher les 10 premières lignes de la table movies :

```
limited = LIMIT movies_details 10 ;
DUMP limited ;

Input(s):
Successfully read 10 records (131072 bytes) from: "/user/student/data/miniprojet/movies.dat"

Output(s):
Successfully stored 10 records (482 bytes) in: "hdfs://sandbox-
hdp.hortonworks.com:8020/tmp/temp1699679319/tmp1037103974"

2022-02-12 20:15:06,188 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat -
Total input paths to process : 1
2022-02-12 20:15:06,188 [main] INFO
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process
: 1
(1,Toy Story (1995),Animation|Children's|Comedy)
(2,Jumanji (1995),Adventure|Children's|Fantasy)
(3,Grumpier Old Men (1995),Comedy|Romance)
(4,Waiting to Exhale (1995),Comedy|Drama)
(5,Father of the Bride Part II (1995),Comedy)
(6,Heat (1995),Action|Crime|Thriller)
(7,Sabrina (1995),Comedy|Romance)
(8,Tom and Huck (1995),Adventure|Children's)
(9,Sudden Death (1995),Action)
(10,GoldenEye (1995),Action|Adventure|Thriller)
```

3. Afficher les 10 premières lignes de la table users:

```
limited = LIMIT users_details 10 ;
DUMP limited ;

Input(s):
Successfully read 10 records (131072 bytes) from: "/user/student/data/miniprojet/users.dat"

Output(s):
Successfully stored 10 records (216 bytes) in: "hdfs://sandbox-
hdp.hortonworks.com:8020/tmp/temp1699679319/tmp1757236790"

2022-02-12 20:31:35,423 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat -
Total input paths to process : 1
2022-02-12 20:31:35,423 [main] INFO
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process
: 1
(1,F,1,10,48067)
(2,M,56,16,70072)
(3,M,25,15,55117)
(4,M,45,7,02460)
(5,M,25,20,55455)
(6,F,50,9,55117)
(7,M,35,1,06810)
(8,M,25,12,11413)
(9,M,25,17,61614)
(10,F,35,1,95370)
```

4. Afficher les 10 premières lignes de la table ratings :

```
limited = LIMIT ratings_details 10 ;
DUMP limited ;

Input(s):
Successfully read 10 records (131072 bytes) from: "/user/student/data/miniprojet/ratings.dat"

Output(s):
Successfully stored 10 records (150 bytes) in: "hdfs://sandbox-
hdp.hortonworks.com:8020/tmp/temp1699679319/tmp-1893179374"

2022-02-12 20:33:25,900 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat -
Total input paths to process : 1
2022-02-12 20:33:25,900 [main] INFO
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process
: 1
(1,1193,5,978300760)
(1,661,3,978302109)
(1,914,3,978301968)
(1,3408,4,978300275)
(1,2355,5,978824291)
(1,1197,3,978302268)
(1,1287,5,978302039)
(1,2804,5,978300719)
(1,594,4,978302268)
(1,919,4,978301368)
```

5. Enregistrer le resultat de la varialble movies_rations_result dans un fichier et affichier les 10 premiere ligne :

```
STORE movies_rations_result INTO
'/user/student/data/miniprojet/movies_ratings_result';
```

```
[student@sandbox-hdp ~]$ hadoop fs -ls /user/student/data/miniprojet/movies_rations_result
Found 2 items
-rw-r--r--   1 student student          0 2022-02-12 22:55
/user/student/data/miniprojet/movies_rations_result/_SUCCESS
-rw-r--r--   1 student student       6188 2022-02-12 22:55
/user/student/data/miniprojet/movies_rations_result/part-v006-o000-r-00000
[student@sandbox-hdp ~]$ hadoop fs -cat
/user/student/data/miniprojet/movies_rations_result/part-v006-o000-r-00000 |head -n 10

5
{(5282,5,1,961096251),(1880,5,3,974700429),(3884,5,2,967597601),(4560,5,2,964633542),....}
```

6. Afficher les utilisateurs féminins dont l'âge se situe entre 20 et 30 ans:

```
femaleUsers = FILTER users_details BY (Gender == 'F') AND (Age <
30) AND (Age >= 20);
limited = LIMIT femaleUsers 10 ;
DUMP limited ;

Input(s):
Successfully read 96 records (131072 bytes) from:
"/user/student/data/miniprojet/users.dat"

Output(s):
Successfully stored 10 records (215 bytes) in: "hdfs://sandbox-
hdp.hortonworks.com:8020/tmp/temp1171754118/tmp186079344"

2022-02-13 19:13:56,458 [main] INFO
org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total
input paths to process : 1
2022-02-13 19:13:56,458 [main] INFO
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil -
Total input paths to process : 1
(11,F,25,1,04093)
(24,F,25,7,10023)
(28,F,25,1,14607)
(32,F,25,0,19355)
(37,F,25,9,66212)
(50,F,25,2,98133)
(69,F,25,1,02143)
(81,F,25,0,60640)
(83,F,25,2,94609)
(96,F,25,16,78028)
```

7. Afficher films par categorie Action et War:

```
actionWarMovies = FILTER movies_details BY (Genres matches
'.*Action.*') AND (Genres matches '.*War.*');
limited = LIMIT actionWarMovies 10;
DUMP limited;

Input(s):
Successfully read 1193 records (131072 bytes) from:
"/user/student/data/miniprojet/movies.dat"

Output(s):
Successfully stored 10 records (687 bytes) in: "hdfs://sandbox-
hdp.hortonworks.com:8020/tmp/temp-337545176/tmp-1463666974"

2022-02-13 19:30:16,060 [main] INFO
org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input
paths to process : 1
2022-02-13 19:30:16,060 [main] INFO
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil -
Total input paths to process : 1
(110,Braveheart (1995),Action|Drama|War)
(465,Heaven & Earth (1993),Action|Drama|War)
(466,Hot Shots! Part Deux (1993),Action|Comedy|War)
(688,Operation Dumbo Drop (1995),Action|Adventure|Comedy|War)
```

```
(780,Independence Day (ID4) (1996),Action|Sci-Fi|War)
(969,African Queen, The (1951),Action|Adventure|Romance|War)
(1196,Star Wars: Episode V - The Empire Strikes Back
(1980),Action|Adventure|Drama|Sci-Fi|War)
(1200,Aliens (1986),Action|Sci-Fi|Thriller|War)
(1205,Transformers: The Movie, The
(1986),Action|Animation|Children's|Sci-Fi|Thriller|War)
(1210,Star Wars: Episode VI - Return of the Jedi
(1983),Action|Adventure|Romance|Sci-Fi|War)
grunt>
```