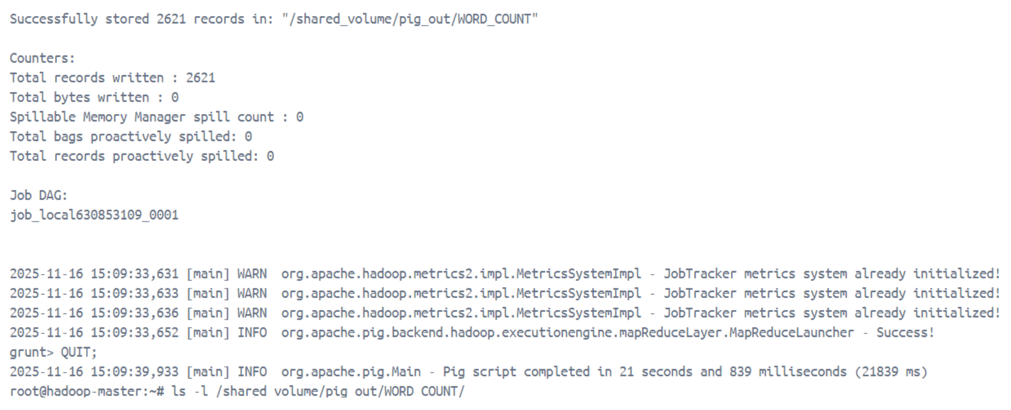
TP5 : PIG

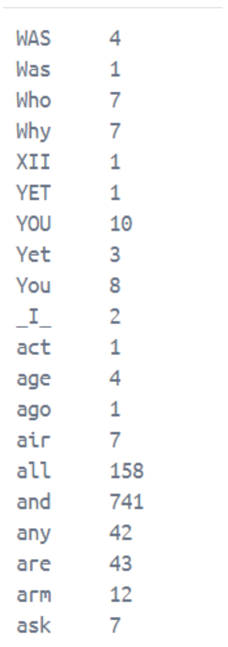
Ahmed QAIS

II. Premier exemple



```
Successfully stored 2621 records in: "/shared_volume/pig_out/WORD_COUNT"

Counters:
Total records written : 2621
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local630853109_0001


2025-11-16 15:09:33,631 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-11-16 15:09:33,633 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-11-16 15:09:33,636 [main] WARN  org.apache.hadoop.metrics2.impl.MetricsSystemImpl - JobTracker metrics system already initialized!
2025-11-16 15:09:33,652 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
grunt> QUIT;
2025-11-16 15:09:39,933 [main] INFO  org.apache.pig.Main - Pig script completed in 21 seconds and 839 milliseconds (21839 ms)
root@hadoop-master:~# ls -l /shared volume/pig out/WORD COUNT/
```

Exemple de résultats



```
WAS     4
Was     1
Who     7
Why     7
XII     1
YET     1
YOU     10
Yet     3
You     8
_I_     2
act     1
age     4
ago     1
air     7
all     158
and     741
any     42
are     43
arm     12
ask     7
```

III. employés d’une entreprise

-- Chargement des données

Employees = LOAD '/input/employees.txt' USING PigStorage(',')

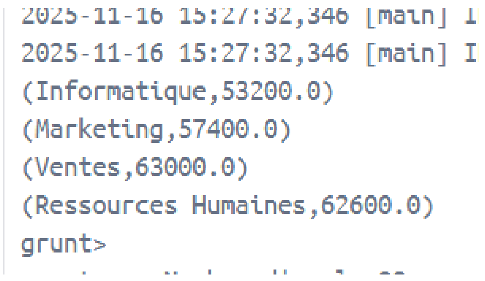
AS (ID:chararray, Nom:chararray, Genre:chararray, Salaire:chararray, depno:chararray, Region:chararray);

-- Chargez les départements

```
Departments = LOAD '/input/department.txt' USING PigStorage(',') AS (depno:int,
name:chararray);
```

• Quel est le salaire moyen des employés dans chaque département ?

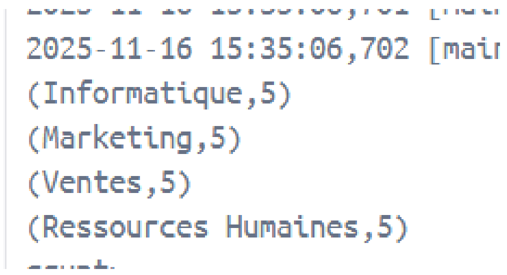
```
EmpByDept = GROUP Employees BY depno;

AvgSalary = FOREACH EmpByDept GENERATE group AS depno, AVG(Employees.Salaire) AS
average_salary;

-- Jointure pour afficher le nom

AvgSalaryNames = JOIN AvgSalary BY depno, Departments BY depno;

FinalAvgSalary = FOREACH AvgSalaryNames GENERATE Departments::name,
AvgSalary::average_salary;

DUMP FinalAvgSalary;
```



```
2025-11-16 15:27:32,346 [main] I
2025-11-16 15:27:32,346 [main] I
(Informatique,53200.0)
(Marketing,57400.0)
(Ventes,63000.0)
(Ressources Humaines,62600.0)
grunt>
```

• Combien d'employés travaillent dans chaque département ?

```
EmpByDeptCount = GROUP CleanedEmployees BY depno;

CountByDept = FOREACH EmpByDeptCount GENERATE

        group AS depno,

        COUNT(CleanedEmployees) AS employee_count;

-- Jointure avec le nom du département

CountByDeptNames = JOIN CountByDept BY depno, Departments BY depno;

FinalCount = FOREACH CountByDeptNames GENERATE Departments::name,
CountByDept::employee_count;

DUMP FinalCount;
```



```
2025-11-16 15:35:06,702 [mai
(Informatique,5)
(Marketing,5)
(Ventes,5)
(Ressources Humaines,5)
```

• Lister tous les employés avec leurs départements respectifs.

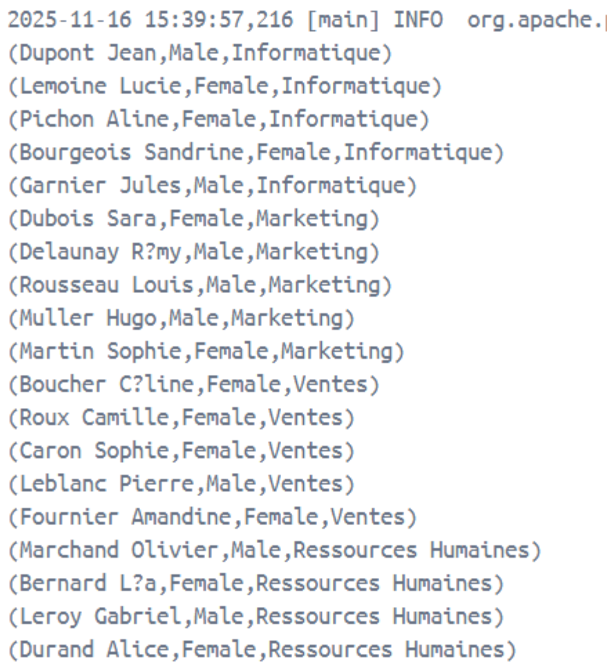
ListEmpDept = FOREACH JoinedEmpDept GENERATE

Employees::Nom,

Employees::Genre AS Genre,

Departments::name AS Department_Name;

DUMP ListEmpDept;

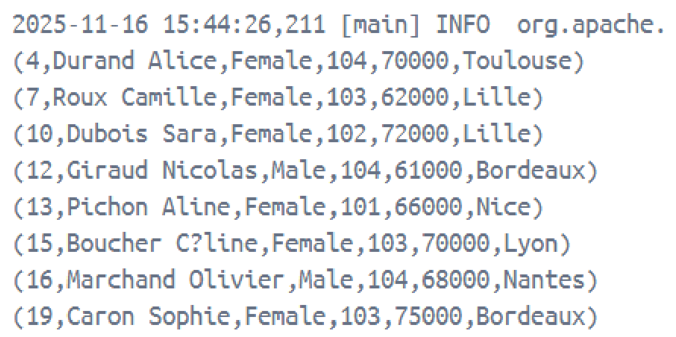


```
2025-11-16 15:39:57,216 [main] INFO  org.apache.
(Dupont Jean,Male,Informatique)
(Lemoine Lucie,Female,Informatique)
(Pichon Aline,Female,Informatique)
(Bourgeois Sandrine,Female,Informatique)
(Garnier Jules,Male,Informatique)
(Dubois Sara,Female,Marketing)
(Delaunay R?my,Male,Marketing)
(Rousseau Louis,Male,Marketing)
(Muller Hugo,Male,Marketing)
(Martin Sophie,Female,Marketing)
(Boucher C?line,Female,Ventes)
(Roux Camille,Female,Ventes)
(Caron Sophie,Female,Ventes)
(Leblanc Pierre,Male,Ventes)
(Fournier Amandine,Female,Ventes)
(Marchand Olivier,Male,Ressources Humaines)
(Bernard L?a,Female,Ressources Humaines)
(Leroy Gabriel,Male,Ressources Humaines)
(Durand Alice,Female,Ressources Humaines)
```

• Quels sont les employés ayant un salaire supérieur à 60 000 ?

HighSalary = FILTER Employees BY Salaire > 60000;

DUMP HighSalary;

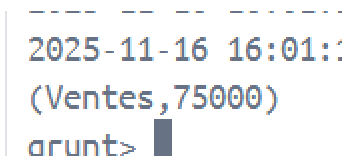


```
2025-11-16 15:44:26,211 [main] INFO  org.apache.
(4,Durand Alice,Female,104,70000,Toulouse)
(7,Roux Camille,Female,103,62000,Lille)
(10,Dubois Sara,Female,102,72000,Lille)
(12,Giraud Nicolas,Male,104,61000,Bordeaux)
(13,Pichon Aline,Female,101,66000,Nice)
(15,Boucher C?line,Female,103,70000,Lyon)
(16,Marchand Olivier,Male,104,68000,Nantes)
(19,Caron Sophie,Female,103,75000,Bordeaux)
```

• Quel est le département avec le salaire le plus élevé ?

DeptWithMaxSalary = FOREACH HighestPaidDept GENERATE Departments::name, HighestPaidEmp::CleanedEmployees::Salaire AS Salaire;

DUMP DeptWithMaxSalary;



```
2025-11-16 16:01:
(Ventes,75000)
grunt>
```

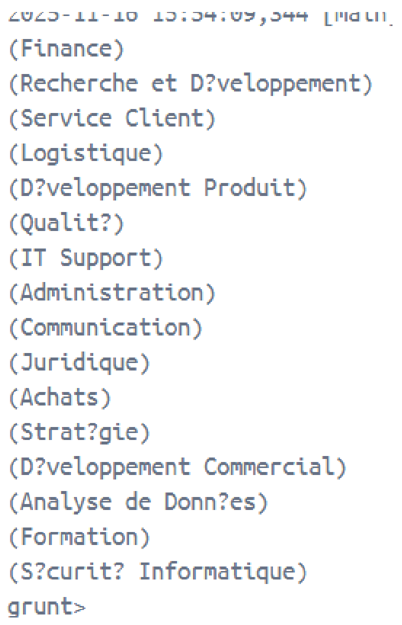
• Lister tous les départements sans employés.

DeptLeftJoin = JOIN Departments BY depno LEFT OUTER, Employees BY depno;

DeptWithoutEmp = FILTER DeptLeftJoin BY Employees::ID IS NULL;

FinalDeptWithoutEmp = FOREACH DeptWithoutEmp GENERATE Departments::name;

DUMP FinalDeptWithoutEmp;



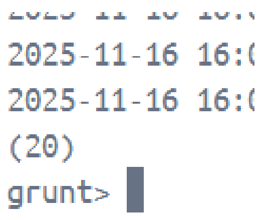
```
2025-11-16 15:54:09,344 [main
(Finance)
(Recherche et D?veloppement)
(Service Client)
(Logistique)
(D?veloppement Produit)
(Qualit?)
(IT Support)
(Administration)
(Communication)
(Juridique)
(Achats)
(Strat?gie)
(D?veloppement Commercial)
(Analyse de Donn?es)
(Formation)
(S?curit? Informatique)
grunt>
```

• Quel est le nombre total d'employés dans l'entreprise ?

TotalEmp = GROUP Employees ALL;

TotalCount = FOREACH TotalEmp GENERATE COUNT(Employees) AS Total_Employees;

DUMP TotalCount;

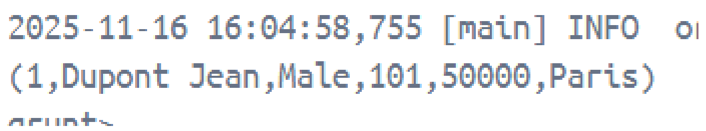


```
2025-11-16 16:(
2025-11-16 16:(
(20)
grunt>
```

• Lister tous les employés de la ville de Paris.

EmpParis = FILTER Employees BY Region MATCHES 'Paris';

DUMP EmpParis;



```
2025-11-16 16:04:58,755 [main] INFO  o
(1,Dupont Jean,Male,101,50000,Paris)
```

• Quel est le salaire total des employés dans chaque ville ?

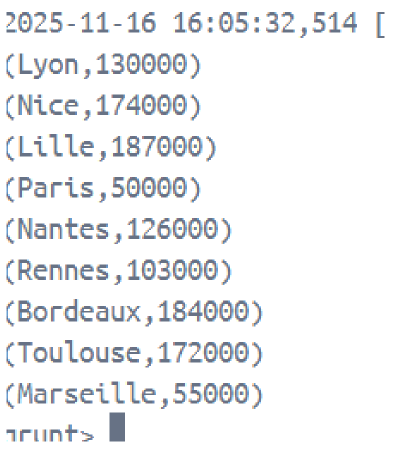
EmpByCity = GROUP Employees BY Region;

TotalSalaryCity = FOREACH EmpByCity GENERATE

group AS City,

SUM(Employees.Salaire) AS Total_Salary;

DUMP TotalSalaryCity;



```
2025-11-16 16:05:32,514 [
(Lyon,130000)
(Nice,174000)
(Lille,187000)
(Paris,50000)
(Nantes,126000)
(Rennes,103000)
(Bordeaux,184000)
(Toulouse,172000)
(Marseille,55000)
grunt>
```

• Quels sont les départements qui ont des femmes employées ?

Enregistrer le dernier résultat sur hdfs dans le dossier pigout/employes_femmes

EmpWomen = FILTER CleanedEmployees BY Genre == 'Female';

WomenDeptProj = FOREACH EmpWomen GENERATE depno;

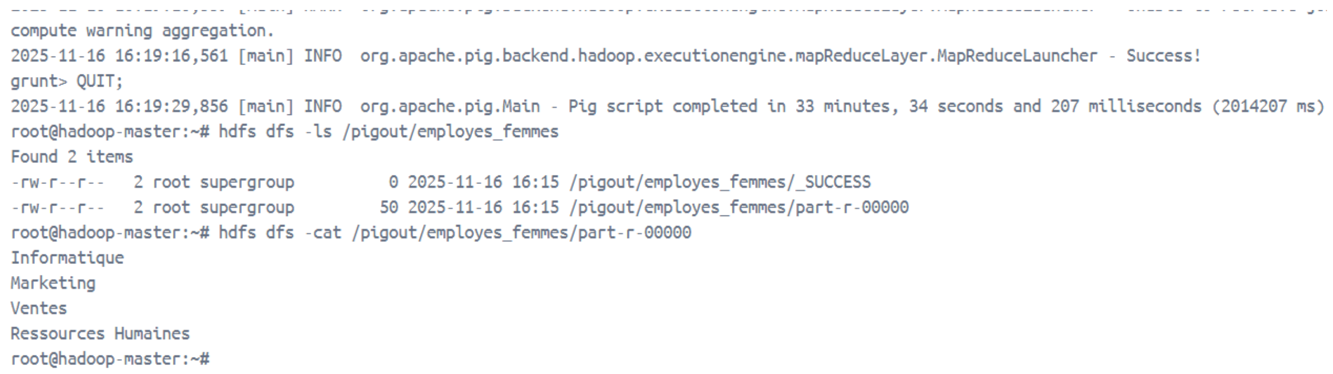
WomenDept = DISTINCT WomenDeptProj;

WomenDeptNames = JOIN WomenDept BY depno, Departments BY depno;

FinalWomenDept = FOREACH WomenDeptNames GENERATE Departments::name;

##enregistrer sous HDFS

STORE FinalWomenDept INTO '/pigout/employes_femmes';

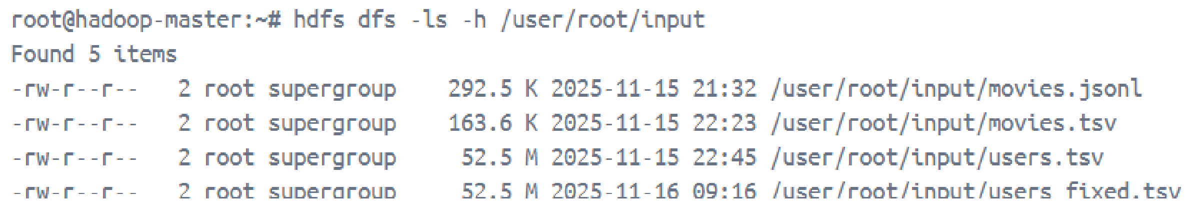


```
compute warning aggregation.
2025-11-16 16:19:16,561 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
grunt> QUIT;
2025-11-16 16:19:29,856 [main] INFO  org.apache.pig.Main - Pig script completed in 33 minutes, 34 seconds and 207 milliseconds (2014207 ms)
root@hadoop-master:~# hdfs dfs -ls /pigout/employes_femmes
Found 2 items
-rw-r--r--   2 root supergroup          0 2025-11-16 16:15 /pigout/employes_femmes/_SUCCESS
-rw-r--r--   2 root supergroup         50 2025-11-16 16:15 /pigout/employes_femmes/part-r-00000
root@hadoop-master:~# hdfs dfs -cat /pigout/employes_femmes/part-r-00000
Informatique
Marketing
Ventes
Ressources Humaines
root@hadoop-master:~#
```

IV. Analyse des films

La Partie IV (Analyse JSON) est malheureusement irréalisable dans cet environnement (pig 0.17). Les méthodes standard pour gérer le format JSON Lines et les structures imbriquées sont soit bloquées par l'absence d'UDF (JsonLoader ou JSONDECODE), soit par des erreurs de résolution de fonction (REGEX_EXTRACT). La réalisation de l’exercice en utilisant le format JSON m’était impossible à ce point. Du coup, j’ai essayé d’utiliser le format JSONL mais j’ai fait face au même problème.

La solution était de transformer les fichiers en TSV et les traiter de cette façon :



```
root@hadoop-master:~# hdfs dfs -ls -h /user/root/input
Found 5 items
-rw-r--r--   2 root supergroup    292.5 K 2025-11-15 21:32 /user/root/input/movies.jsonl
-rw-r--r--   2 root supergroup    163.6 K 2025-11-15 22:23 /user/root/input/movies.tsv
-rw-r--r--   2 root supergroup     52.5 M 2025-11-15 22:45 /user/root/input/users.tsv
-rw-r--r--   2 root supergroup     52.5 M 2025-11-16 09:16 /user/root/input/users_fixed.tsv
```

Pour loader les données sur PIG, on utilise la commande suivante :

- *movies = LOAD '/tmp/movies.tsv' USING PigStorage('\t') AS (c0:chararray, c1:chararray, c2:chararray, c3:chararray, c4:chararray);*
- *users = LOAD '/tmp/users_fixed.tsv' USING PigStorage('\t') AS (u0:chararray, u1:chararray, u2:chararray, u3:chararray, u4:chararray, u5:chararray, u6:chararray, u7:chararray, u8:chararray);*

De plus, les éléments de nos fichiers des films actuels ne sont pas les mêmes que ceux dans le Lab. Du coup, on ne peut pas faire les mêmes traitements. Comme alternative, voici quelques traitements réalisés sur nos fichiers.

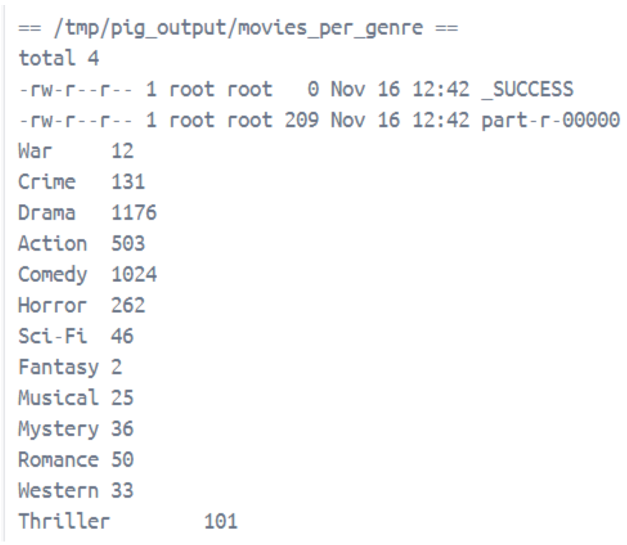
- **Films par genre**

movies_genre_split = FOREACH movies_with_year GENERATE movieId, title, year, FLATTEN(STRSPLIT((genres is null ? '' : genres),'\\|')) AS genre;

grp_genre = GROUP movies_genre_split BY genre;

movies_per_genre = FOREACH grp_genre GENERATE group AS genre, COUNT(movies_genre_split.movieId) AS cnt;

STORE movies_per_genre INTO '/tmp/pig_output/movies_per_genre' USING PigStorage('\t');



```
== /tmp/pig_output/movies_per_genre ==
total 4
-rw-r--r-- 1 root root   0 Nov 16 12:42 _SUCCESS
-rw-r--r-- 1 root root 209 Nov 16 12:42 part-r-00000
War     12
Crime   131
Drama   1176
Action  503
Comedy  1024
Horror  262
Sci-Fi  46
Fantasy 2
Musical 25
Mystery 36
Romance 50
Western 33
Thriller        101
```

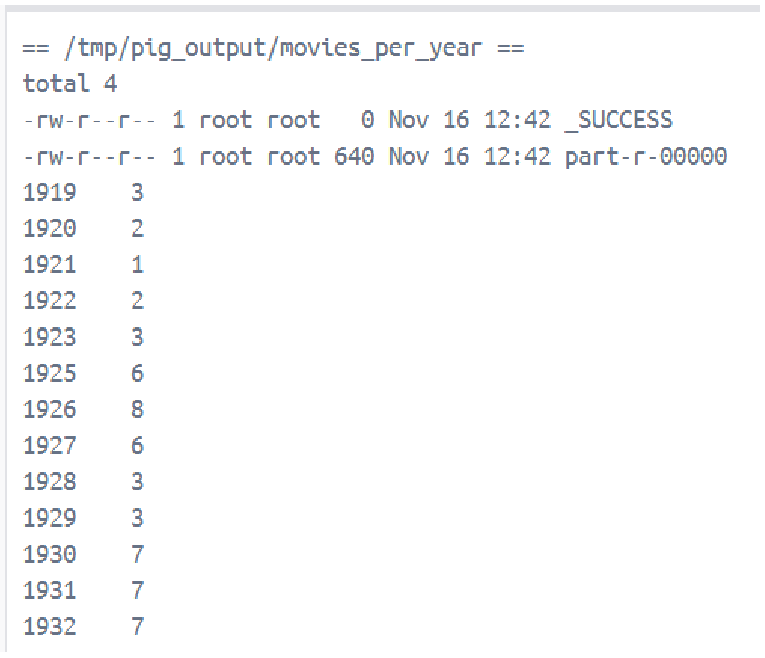
- **Films par année**

  movies_valid = FILTER movies_with_year BY year IS NOT NULL AND year > 0;

  grp_by_year = GROUP movies_valid BY year;

  movies_per_year = FOREACH grp_by_year GENERATE group AS year, COUNT(movies_valid.movieId) AS cnt;

  STORE movies_per_year INTO '/tmp/pig_output/movies_per_year' USING PigStorage('\t');



```
== /tmp/pig_output/movies_per_year ==
total 4
-rw-r--r-- 1 root root   0 Nov 16 12:42 _SUCCESS
-rw-r--r-- 1 root root 640 Nov 16 12:42 part-r-00000
1919    3
1920    2
1921    1
1922    2
1923    3
1925    6
1926    8
1927    6
1928    3
1929    3
1930    7
1931    7
1932    7
```

Pour les traitements actuels dans le lab, voici les codes :

- **Créez une collection mUSA_annee groupant les films américains par année (code du pays: US).**

MoviesUSA = FILTER Movies BY country == 'US';

mUSA_annee = GROUP MoviesUSA BY year;

- **Créez une collection mUSA_director groupant les films américains par metteur en scène.**

mUSA_director = GROUP MoviesUSA BY director._id;

- **Créez une collection mUSA_acteurs contenant des triplets (idFilm, idActeur, role). Chaque film apparaît donc dans autant de documents qu'il y a d'acteurs**

Aide: il faut « aplatir » la collection actors imbriquée dans chaque film.

```
mUSA_acteurs_raw = FOREACH MoviesUSA GENERATE
                _id AS idFilm,
                FLATTEN(actors) AS (idActeur: chararray, role: chararray);
mUSA_acteurs = FOREACH mUSA_acteurs_raw GENERATE idFilm, idActeur, role;
```

- **Créez une collection moviesActors associant l'identifiant du film à la description complète de l'acteur.**

Aide: utiliser une jointure . Consultez le schéma de mUSA_actors pour connaître le nom des colonnes.

moviesActors = JOIN mUSA_acteurs BY idActeur, Artists BY _id;

- **créez une collection fullMovies associant la description complète du film à la description complète de tous les acteurs.**

Aide: soit une jointure entre moviesActors et movies, puis un regroupement par film, ce qui un contenu correct mais très compliqué , soit un cogroup entre moviesUSA et moviesActors

fullMovies = COGROUP MoviesUSA BY _id, moviesActors BY mUSA_acteurs::idFilm;

- **Créer une collection ActeursRealisateurs donnant pour chaque artiste la liste des films où il/elle a joué (éventuellement vide), et des films qu'il/elle a dirigé. On peut se contenter d'afficher l'identifiant de l'artiste : (artist:24,{},{(movie:10,Blade Runner,artist:24,Deckard), (movie:34,Le retour du Jedi,artist:24,Han Solo)})**

Enregistrer le dernier résultat sur hdfs dans le dossier pigout/ ActeursRealisateur

-- 1. Préparer les réalisateurs (ID Artiste, Film Dirigé)

Directors = FOREACH MoviesUSA GENERATE director._id AS ArtistID, _id AS movieID_dir, title AS title_dir;

-- 2. Préparer les acteurs (ID Artiste, Film Joué)

ActorsPlayed = FOREACH mUSA_acteurs GENERATE idActeur AS ArtistID, idFilm AS movieID_act, role;

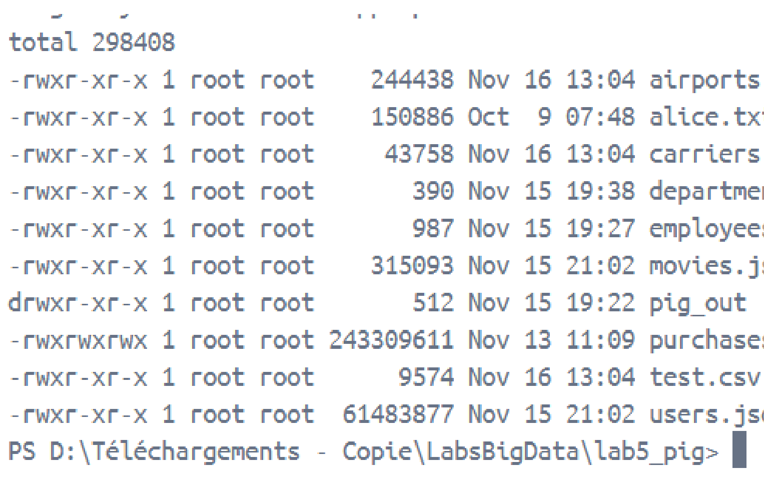
-- 3. COGROUP pour regrouper les rôles et réalisations par artiste

ArtisteMovies = COGROUP Directors BY ArtistID, ActorsPlayed BY ArtistID;

-- 4. Générer la collection finale (ID Artiste, Bag Films Dirigés, Bag Films Joués)

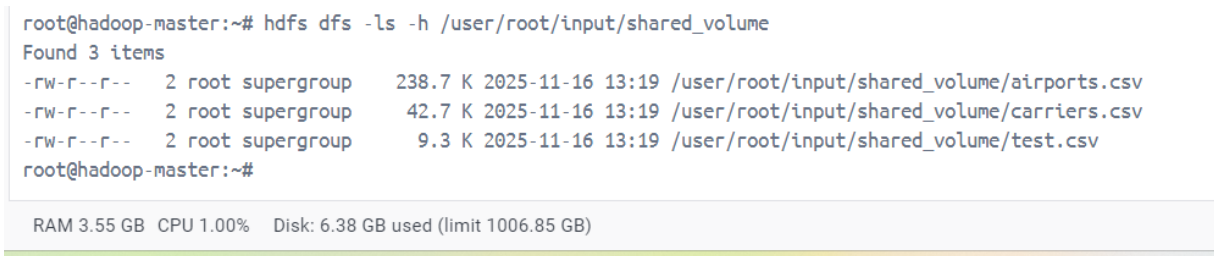
```
ActeursRealisateurs = FOREACH ArtisteMovies GENERATE
            group AS ArtisteID,
            Directors AS Diriges,
            ActorsPlayed AS Joues;
STORE ActeursRealisateurs INTO '/pigout/ActeursRealisateurs';
```

V. Analyse des vols



```
total 298408
-rwxr-xr-x 1 root root    244438 Nov 16 13:04 airports.csv
-rwxr-xr-x 1 root root    150886 Oct  9 07:48 alice.txt
-rwxr-xr-x 1 root root     43758 Nov 16 13:04 carriers.csv
-rwxr-xr-x 1 root root       390 Nov 15 19:38 department.txt
-rwxr-xr-x 1 root root       987 Nov 15 19:27 employees.txt
-rwxr-xr-x 1 root root    315093 Nov 15 21:02 movies.json
drwxr-xr-x 1 root root       512 Nov 15 19:22 pig_out
-rwxrwxrwx 1 root root 243309611 Nov 13 11:09 purchases2.txt
-rwxr-xr-x 1 root root      9574 Nov 16 13:04 test.csv
-rwxr-xr-x 1 root root  61483877 Nov 15 21:02 users.json
PS D:\Téléchargements - Copie\LabsBigData\lab5_pig>
```

Dans hdfs



```
root@hadoop-master:~# hdfs dfs -ls -h /user/root/input/shared_volume
Found 3 items
-rw-r--r--   2 root supergroup    238.7 K 2025-11-16 13:19 /user/root/input/shared_volume/airports.csv
-rw-r--r--   2 root supergroup     42.7 K 2025-11-16 13:19 /user/root/input/shared_volume/carriers.csv
-rw-r--r--   2 root supergroup      9.3 K 2025-11-16 13:19 /user/root/input/shared_volume/test.csv
root@hadoop-master:~#
```

RAM 3.55 GB CPU 1.00%   Disk: 6.38 GB used (limit 1006.85 GB)

Tout d'abord, on charge les données des 3 fichiers sur PIG :

```
Flights = LOAD '/user/root/input/shared_volume/test.csv' USING PigStorage(',') AS (
   Year: int, Month: int, DayofMonth: int, DayOfWeek: int,
   DepTime: chararray, CRSDepTime: chararray, ArrTime: chararray, CRSArrTime: chararray,
   UniqueCarrier: chararray, FlightNum: int, TailNum: chararray,
   ActualElapsedTime: int, CRSElapsedTime: int, AirTime: int,
   ArrDelay: int, DepDelay: int, Origin: chararray, Dest: chararray,
   Distance: int, TaxiIn: int, TaxiOut: int,
   Cancelled: int, CancellationCode: chararray, Diverted: int,
   CarrierDelay: int, WeatherDelay: int, NASDelay: int,
   SecurityDelay: int, LateAircraftDelay: int
);
-- Filtrer les vols valides (non annulés et non détournés)
ValidFlights = FILTER Flights BY (Cancelled == 0 AND Diverted == 0);
```

Airports = LOAD '/user/root/input/shared_volume/airports.csv' USING PigStorage(',')

AS (iata: chararray, airport: chararray, city: chararray, state: chararray, country: chararray, lat: float, long: float);

Carriers = LOAD '/user/root/input/shared_volume/carriers.csv' USING PigStorage(',')

AS (Code: chararray, Description: chararray);

• Top 20 des aéroports par volume total de vols

Departures = FOREACH ValidFlights GENERATE Origin AS Airport;

Arrivals = FOREACH ValidFlights GENERATE Dest AS Airport;

AllFlights = UNION Departures, Arrivals;

VolumeByAirport = GROUP AllFlights BY Airport;

TotalVolume = FOREACH VolumeByAirport GENERATE

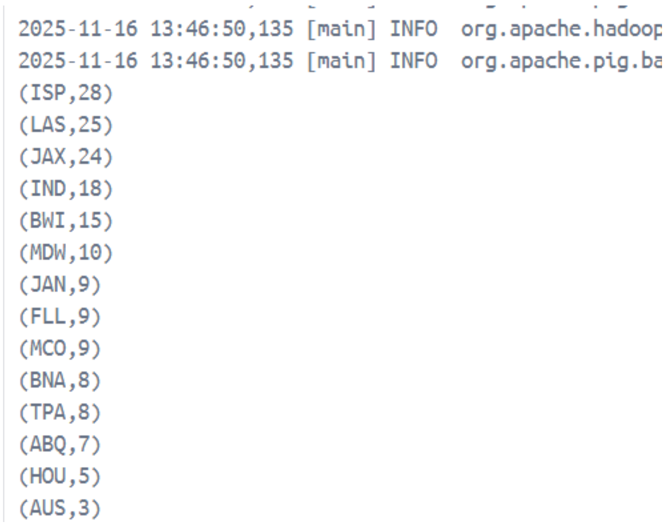
group AS Airport,

COUNT(AllFlights) AS TotalFlights;

SortedVolume = ORDER TotalVolume BY TotalFlights DESC;

Top20Airports = LIMIT SortedVolume 20;

DUMP Top20Airports ;



```
2025-11-16 13:46:50,135 [main] INFO  org.apache.hadoop
2025-11-16 13:46:50,135 [main] INFO  org.apache.pig.ba
(ISP,28)
(LAS,25)
(JAX,24)
(IND,18)
(BWI,15)
(MDW,10)
(JAN,9)
(FLL,9)
(MCO,9)
(BNA,8)
(TPA,8)
(ABQ,7)
(HOU,5)
(AUS,3)
```

Variation : Calculs Entrants, Sortants, et Totaux

-- Compter les vols sortants (Origin)

Outbound = GROUP ValidFlights BY Origin;

CountOut = FOREACH Outbound GENERATE group AS Airport, COUNT(ValidFlights) AS OutboundCount;

-- Compter les vols entrants (Dest)

Inbound = GROUP ValidFlights BY Dest;

CountIn = FOREACH Inbound GENERATE group AS Airport, COUNT(ValidFlights) AS InboundCount;

-- Joindre les comptes (FULL OUTER pour inclure les aéroports qui n'ont que des départs/arrivées)

AirportTotals = JOIN CountOut BY Airport FULL OUTER, CountIn BY Airport;

-- Générer le rapport final

AirportReport = FOREACH AirportTotals GENERATE

(CountOut::Airport IS NULL ? CountIn::Airport : CountOut::Airport) AS Airport_Code,

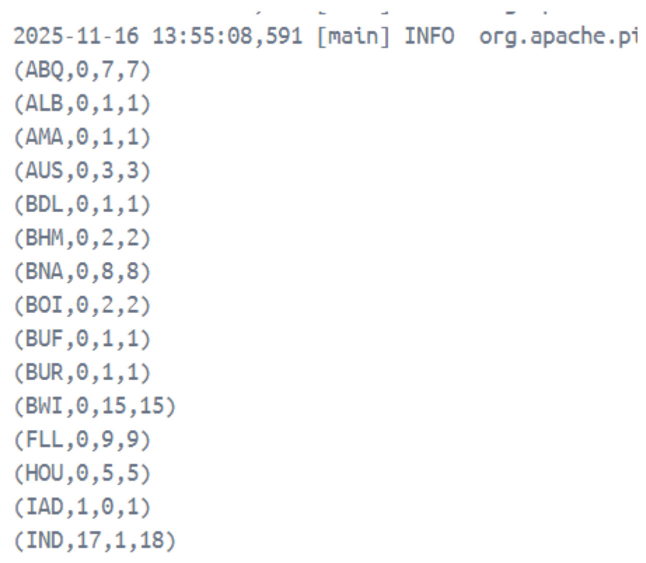
(OutboundCount IS NULL ? 0 : OutboundCount) AS Sortants,

(InboundCount IS NULL ? 0 : InboundCount) AS Entrants,

(long)((OutboundCount IS NULL ? 0 : OutboundCount) + (InboundCount IS NULL ? 0 : InboundCount)) AS Total;

DUMP AirportReport;

Les colonnes sont Airport, sortant, entrants, nombre entrants et sortants

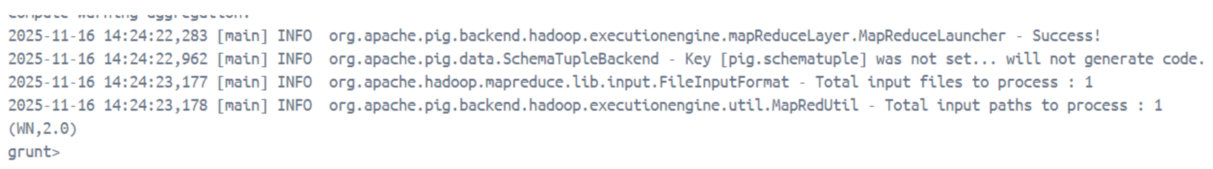


```
2025-11-16 13:55:08,591 [main] INFO  org.apache.pi
(ABQ,0,7,7)
(ALB,0,1,1)
(AMA,0,1,1)
(AUS,0,3,3)
(BDL,0,1,1)
(BHM,0,2,2)
(BNA,0,8,8)
(BOI,0,2,2)
(BUF,0,1,1)
(BUR,0,1,1)
(BWI,0,15,15)
(FLL,0,9,9)
(HOU,0,5,5)
(IAD,1,0,1)
(IND,17,1,18)
```

• Popularité des transporteurs

```
-- 1. Compter le volume total par transporteur (UniqueCarrier) et par Année

CarrierVolume_Group = GROUP ValidFlights BY (UniqueCarrier, Year);

CarrierVolume = FOREACH CarrierVolume_Group GENERATE

            group.UniqueCarrier AS Carrier,

            group.Year AS Year,

            COUNT(ValidFlights) AS TotalFlights;

-- 2. Calculer le volume Logarithmique (Log base 10)

CarrierLogVolume = FOREACH CarrierVolume GENERATE

              Carrier,

              Year,

              LOG10(TotalFlights) AS LogVolume;

-- 3. Grouper par Transporteur pour simuler la médiane (classement basé sur la Moyenne, car
MEDIAN n'est pas native)

CarrierAvgLogVolume = GROUP CarrierLogVolume BY Carrier;

AvgLogVolume = FOREACH CarrierAvgLogVolume GENERATE

           group AS Carrier,

           AVG(CarrierLogVolume.LogVolume) AS AvgLogVolume;

-- 4. Classer les transporteurs par leur volume log moyen (proxy pour la médiane)

RankedCarriers = ORDER AvgLogVolume BY AvgLogVolume DESC;

 DUMP RankedCarriers;
```

Un seul carrier

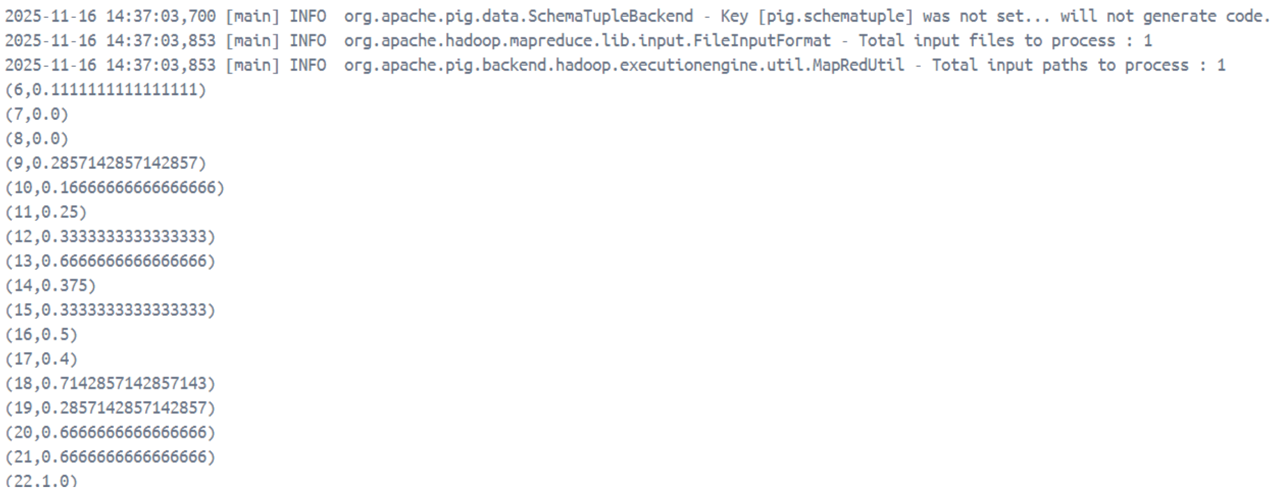


```
2025-11-16 14:24:22,283 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2025-11-16 14:24:22,962 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2025-11-16 14:24:23,177 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2025-11-16 14:24:23,178 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(WN,2.0)
grunt>
```

• Proportion de vols retardés

```
-- Préparer la relation avec le marqueur de retard et l'heure de départ

DelayedFlights = FOREACH ValidFlights GENERATE

             (ArrDelay > 15 ? 1 : 0) AS Delayed, -- 1 si retard > 15 min, 0 sinon
```

```
            Year, Month, DayofMonth, DayOfWeek,
            (int)(DepTime / 100) AS Hour; -- Heure de départ (HH)
-- Calculer la proportion de retard par Heure (Exemple)
DelayByHour_Group = GROUP DelayedFlights BY Hour;
DelayByHour = FOREACH DelayByHour_Group GENERATE
        group AS Hour,
        AVG(DelayedFlights.Delayed) AS DelayProportion;
-- DUMP DelayByHour;
```



```
2025-11-16 14:37:03,700 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2025-11-16 14:37:03,853 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2025-11-16 14:37:03,853 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(6,0.1111111111111111)
(7,0.0)
(8,0.0)
(9,0.2857142857142857)
(10,0.16666666666666666)
(11,0.25)
(12,0.3333333333333333)
(13,0.6666666666666666)
(14,0.375)
(15,0.3333333333333333)
(16,0.5)
(17,0.4)
(18,0.7142857142857143)
(19,0.2857142857142857)
(20,0.6666666666666666)
(21,0.6666666666666666)
(22,1.0)
```

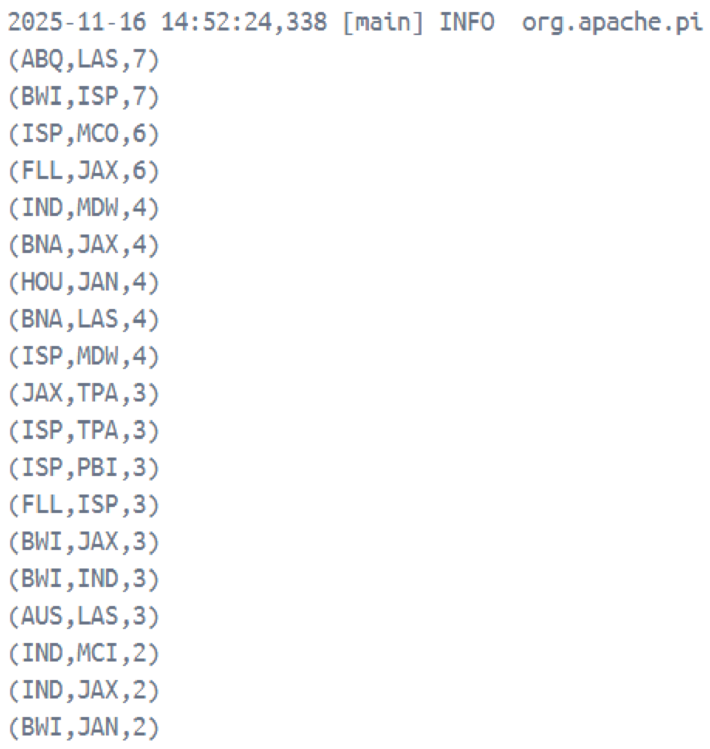
• Retards des transporteurs

```
CarrierDelayByMonth = FOREACH CarrierDelayByMonth_Group GENERATE
            group.UniqueCarrier AS Carrier,
            group.Month AS Month,
            AVG((CarrierDelayedFlights::ValidFlights::ArrDelay > 15 ? 1 : 0)) AS
DelayProportion;
SortedCarrierDelay = ORDER CarrierDelayByMonth BY Carrier, Month;
DUMP SortedCarrierDelay;
```

• Itinéraires les plus fréquentés

```
-- 1. Créer une paire d'aéroports (Origin, Dest)
AirportPairs = FOREACH ValidFlights GENERATE Origin, Dest;
-- 2. Normaliser la paire pour qu'elle soit non ordonnée
NormalizedPairs = FOREACH AirportPairs GENERATE
```

```
        (Origin < Dest ? Origin : Dest) AS Airport1,
        (Origin < Dest ? Dest : Origin) AS Airport2;
-- 3. Grouper par la paire normalisée
ItinerairesGroup = GROUP NormalizedPairs BY (Airport1, Airport2);
-- 4. Compter la fréquence et classer
Frequences = FOREACH ItinerairesGroup GENERATE
      group.Airport1,
      group.Airport2,
      COUNT(NormalizedPairs) AS Frequency;
TopItineraires = ORDER Frequences BY Frequency DESC;
DUMP TopItineraires;
```



```
2025-11-16 14:52:24,338 [main] INFO  org.apache.pi
(ABQ,LAS,7)
(BWI,ISP,7)
(ISP,MCO,6)
(FLL,JAX,6)
(IND,MDW,4)
(BNA,JAX,4)
(HOU,JAN,4)
(BNA,LAS,4)
(ISP,MDW,4)
(JAX,TPA,3)
(ISP,TPA,3)
(ISP,PBI,3)
(FLL,ISP,3)
(BWI,JAX,3)
(BWI,IND,3)
(AUS,LAS,3)
(IND,MCI,2)
(IND,JAX,2)
(BWI,JAN,2)
```