La finalidad de esta práctica es familiarizarse con el lenguaje PIG.

Vete haciendo capturas de pantalla de todos los pasos que vayas dando, acompañándolas de comentarios descriptivos de los mismos.

CONTENIDO

APARTADO A

Práctica con PIG

Trabajaremos sobre el fichero u.user. Las consultas han de hacerse con *mapreduce,* es decir, con los ficheros en HDFS, no en local.

u.user -- Demographic information about the users; this is a tab separated list of
user id | age | gender | occupation | zip code
The user ids are the ones used in the u.data data set.

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -get /user/maría_dev/archive.zip
get: `/user/maría_dev/archive.zip': No such file or directory
[maria_dev@sandbox-hdp ~]$ hdfs dfs -get /user/maria_dev/archive.zip /tmp/
[maria_dev@sandbox-hdp ~]$ unzip /tmp/archive.zip -d /tmp/archive
Archive:  /tmp/archive.zip
  inflating: /tmp/archive/ml-100k/README
  inflating: /tmp/archive/ml-100k/allbut.pl
  inflating: /tmp/archive/ml-100k/mku.sh
  inflating: /tmp/archive/ml-100k/u.data
  inflating: /tmp/archive/ml-100k/u.genre
  inflating: /tmp/archive/ml-100k/u.info
  inflating: /tmp/archive/ml-100k/u.item
  inflating: /tmp/archive/ml-100k/u.occupation
  inflating: /tmp/archive/ml-100k/u.user
  inflating: /tmp/archive/ml-100k/u1.base
  inflating: /tmp/archive/ml-100k/u1.test
  inflating: /tmp/archive/ml-100k/u2.base
  inflating: /tmp/archive/ml-100k/u2.test
  inflating: /tmp/archive/ml-100k/u3.base
  inflating: /tmp/archive/ml-100k/u3.test
  inflating: /tmp/archive/ml-100k/u4.base
  inflating: /tmp/archive/ml-100k/u4.test
  inflating: /tmp/archive/ml-100k/u5.base
  inflating: /tmp/archive/ml-100k/u5.test
  inflating: /tmp/archive/ml-100k/ua.base
  inflating: /tmp/archive/ml-100k/ua.test
```

```
hdfs dfs -ls /user/maría_dev[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls /user/maria_dev
Found 4 items
drwx------   - maria_dev hdfs          0 2025-11-12 09:39 /user/maria_dev/.Trash
-rw-r--r--   1 maria_dev hdfs    4998818 2025-11-12 09:34 /user/maria_dev/archive.zip
drwxr-xr-x   - maria_dev hdfs          0 2025-11-10 08:39 /user/maria_dev/datos
drwxr-xr-x   - maria_dev hdfs          0 2025-11-12 09:42 /user/maria_dev/ml-100k
[maria_dev@sandbox-hdp ~]$
```

1.- Muestra el total de hombres y mujeres que hay en el archivo *u.user*.

```
grunt> usuarios = LOAD '/user/maria_dev/ml-100k/u.user' USING PigStorage('|') AS (user_id:int, age:int, gender:chararray, occupation:chararray, zip:chararray);
grunt>
grunt>  grp_gender = GROUP usuarios BY gender;
grunt> count_gender = FOREACH grp_gender GENERATE group AS gender, COUNT(usuarios) AS total;
grunt> DUMP count_gender;
```

```
2025-11-12 10:21:26,478 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
2025-11-12 10:21:26,479 [main] INFO  org.apache.hadoop.yarn.client.AHSProxy - Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
2025-11-12 10:21:26,482 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2025-11-12 10:21:26,518 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
2025-11-12 10:21:26,518 [main] INFO  org.apache.hadoop.yarn.client.AHSProxy - Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
2025-11-12 10:21:26,522 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2025-11-12 10:21:26,556 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
2025-11-12 10:21:26,557 [main] INFO  org.apache.hadoop.yarn.client.AHSProxy - Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
2025-11-12 10:21:26,560 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server
2025-11-12 10:21:26,598 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2025-11-12 10:21:26,604 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2025-11-12 10:21:26,633 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2025-11-12 10:21:26,634 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(F,273)
(M,670)
grunt>
```

273 mujeres

670 hombres

2.- Mediante instrucciones de PIG encontrar las 10 ocupaciones más frecuentes entre los usuarios.

```
grunt> grp_occ = GROUP usuarios BY occupation;
grunt> occ_count = FOREACH grp_occ GENERATE group AS occupation, COUNT(usuarios) AS n;
grunt> occ_sorted = ORDER occ_count BY n DESC;
grunt> top10_occ = LIMIT occ_sorted 10;
grunt> DUMP top10_occ;
```

```
2025-11-12 10:51:02,953 [main] INFO  org.apache.hadoop.mapreduce.lib.input.Fi
2025-11-12 10:51:02,953 [main] INFO  org.apache.pig.backend.hadoop.executione
(student,196)
(other,105)
(educator,95)
(administrator,79)
(engineer,67)
(programmer,66)
(librarian,51)
(writer,45)
(executive,32)
(scientist,31)
grunt>
```

3.- Muestra la edad media por géneros.

```
grunt> avg_age_media = FOREACH grp_gender GENERATE group AS gender, AVG(usuarios.age) AS avg_age;
grunt> DUMP avg_age_media;
```

```
2025-11-12 10:57:41,268 [main] INFO  org.apache.hadoop.yarn.c
2025-11-12 10:57:41,269 [main] INFO  org.apache.hadoop.yarn.c
0200
2025-11-12 10:57:41,277 [main] INFO  org.apache.hadoop.mapred
g to job history server
2025-11-12 10:57:41,299 [main] INFO  org.apache.pig.backend.h
2025-11-12 10:57:41,301 [main] INFO  org.apache.pig.data.Sche
2025-11-12 10:57:41,307 [main] INFO  org.apache.hadoop.mapred
2025-11-12 10:57:41,307 [main] INFO  org.apache.pig.backend.h
(F,33.81318681318681)
(M,34.149253731343286)
grunt>
```

4.- Muestra la edad media por ocupaciones.

```
grunt> avg_age_media = FOREACH grp_gender GENERATE group AS gender, AVG(usuarios.age) AS avg_age;
grunt> DUMP avg_age_media;
```

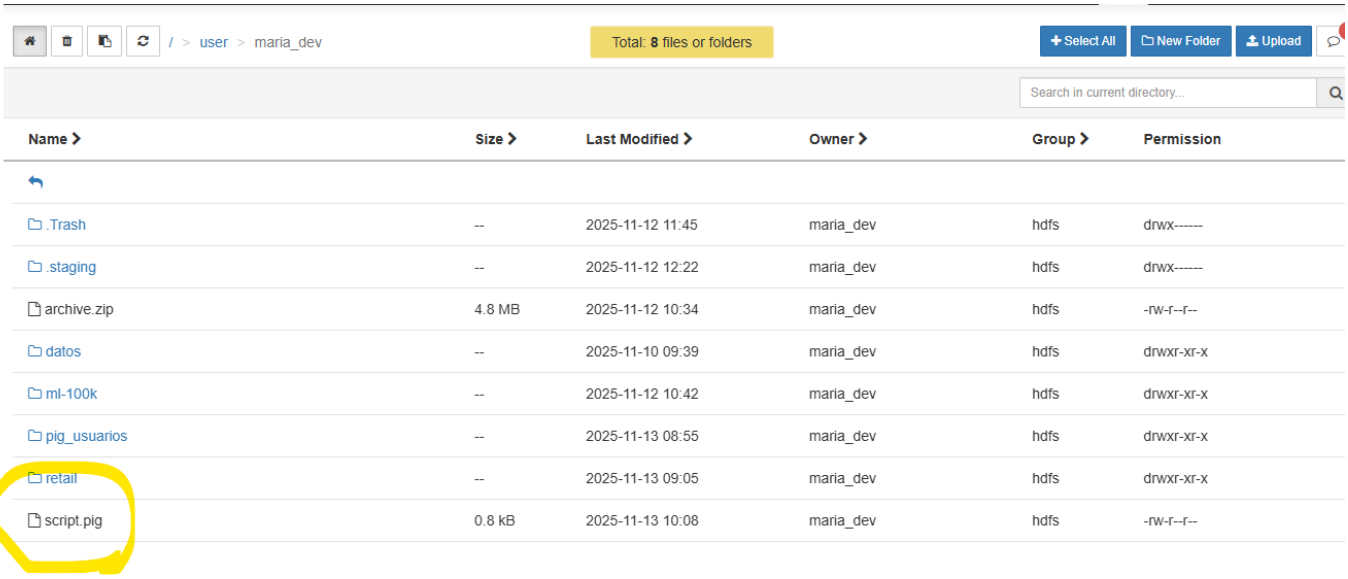
```
2025-11-12 11:02:11,692 [main] INFO  org.apache.hadoo
2025-11-12 11:02:11,692 [main] INFO  org.apache.pig.b
(retired,63.07142857142857)
(doctor,43.57142857142857)
(educator,42.010526315789 48)
(healthcare,41.5625)
(librarian,40.0)
(administrator,38.746835443037 97)
(executive,38.71875)
(marketing,37.615384615384 61)
(lawyer,36.75)
(engineer,36.388059701492 54)
(writer,36.311111111111 11)
(salesman,35.666666666666 64)
(scientist,35.548387096774 19)
(other,34.523809523809526)
(technician,33.148148148148 145)
(programmer,33.121212121212 125)
(homemaker,32.57142857142857)
(artist,31.392857142857142)
(entertainment,29.222222222222 22)
(none,26.555555555555557)
(student,22.081632653061224)
grunt>
```

5.- Guarda el resultado de las cuatro consultas anteriores en un script de extensión "*.pig*". Ejecútalo. (recuerda, siempre en la carpeta `/user/maria_dev`)

Anadimos la línea del cmando STORE para que

STORE count_gender INTO '/user/maria_dev/pig_usuarios/count_gender' USING PigStorage('\t');

```
usuarios = LOAD '/user/maria_dev/ml-100k/u.user' USING PigStorage('|') AS (user_id:int, age:int, gender:chararray, occupation:chararray, zip:chararray);
grp_gender = GROUP usuarios BY gender;


grp_gender = GROUP usuarios BY gender;

count_gender = FOREACH grp_gender GENERATE group AS gender, COUNT(usuarios) AS total;

STORE count_gender INTO '/user/maria_dev/pig_usuarios/count_gender' USING PigStorage('\t');

grp_occ = GROUP usuarios BY occupation;
occ_count = FOREACH grp_occ GENERATE group AS occupation, COUNT(usuarios) AS n;
occ_sorted = ORDER occ_count BY n DESC;
top10_occ = LIMIT occ_sorted 10;

STORE top10_occ INTO '/user/maria_dev/pig_usuarios/top10_occ' USING PigStorage('\t');

avg_age_media = FOREACH grp_gender GENERATE group AS gender, AVG(usuarios.age) AS avg_age;


STORE avg_age_media INTO '/user/maria_dev/pig_usuarios/avg_age_media' USING PigStorage('\t');

avg_age_occ = FOREACH grp_occ GENERATE group AS occupation, AVG(usuarios.age) AS avg_age;
avg_age_occ_sorted = ORDER avg_age_occ BY avg_age DESC;


STORE avg_age_occ_sorted INTO '/user/maria_dev/pig_usuarios/avg_age_occ_sorted' USING PigStorage('\t');
```



6.- Almacena la salida de las cuatro consultas anteriores en una carpeta de HDFS llamada *pig_usuarios*.

avg_age_by_media	--	2025-11-13 08:56
count_gender	--	2025-11-12 12:16
top10_occupationes	--	2025-11-12 12:20

INTRODUCCIÓN

A.- Ahora trabajaremos sobre el conjunto de datos ***Retail Sales Dataset*** de **Kaggle:**

https://www.kaggle.com/datasets/mohammadtalib786/retail-sales-dataset Investiga su formato.

CONTENIDO
APARTADO B

1. Carga y descripción del *dataset*

```
[maria_dev@sandbox-hdp ~]$ hdfs dfs -put retail_sales_dataset.csv /user/maria_dev/retail/retail_sales_dataset.csv

[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls
Found 7 items
drwx------   - maria_dev hdfs          0 2025-11-12 10:45 .Trash
drwx------   - maria_dev hdfs          0 2025-11-12 11:22 .staging
-rw-r--r--   1 maria_dev hdfs    4998818 2025-11-12 09:34 archive.zip
drwxr-xr-x   - maria_dev hdfs          0 2025-11-10 08:39 datos
drwxr-xr-x   - maria_dev hdfs          0 2025-11-12 09:42 ml-100k
drwxr-xr-x   - maria_dev hdfs          0 2025-11-13 07:55 pig_usuarios
drwxr-xr-x   - maria_dev hdfs          0 2025-11-13 08:05 retail
[maria_dev@sandbox-hdp ~]$ hdfs dfs -ls retail
Found 1 items
-rw-r--r--   1 maria_dev hdfs      51673 2025-11-13 08:05 retail/retail_sales_dataset.csv
[maria_dev@sandbox-hdp ~]$
```

- **Carga el archivo (*retail_sales_dataset.csv*) usando PigStorage(',') y define un esquema correctamente para cada tipo de campo.**

  EL CSV lo asignamos ventas

```
Transaction ID,Date,Customer ID,Gender,Age,Product Category,Quantity,Price per Unit,Total Amount
1,2023-11-24,CUST001,Male,34,Beauty,3,50,150
2,2023-02-27,CUST002,Female,26,Clothing,2,500,1000
3,2023-01-13,CUST003,Male,50,Electronics,1,30,30
```

```
grunt> ventas = LOAD '/user/maria_dev/retail/retail_sales_dataset.csv' USING PigStorage(',') AS (transaction_id:chararra
y, date:chararray, customer_id:chararray, age:int, product_category:chararray, quantiti:int, price_per_unit:double, tota
l_amount:double);
```

transaction_id:chararray

date:chararray

customer_id:chararray

age:int

product_category:chararray

quantiti:int

price_per_unit:double

total_amount:double)

- **Usa DESCRIBE para ver el esquema y DUMP para ver las primeras tuplas.**

```
grunt> DESCRIBE ventas;
ventas: {transaction_id: chararray,date: chararray,customer_id: chararray,age: int,product_category: c
hararray,quantiti: int,price_per_unit: double,total_amount: double}
grunt>
```

```
grunt> DUMP ventas LIMIT 10;
```

```
2025-11-13 08:49:33,822 [main] INFO  org.ap
ut paths to process : 1
2025-11-13 08:49:33,822 [main] INFO  org.ap
otal input paths to process : 1
(Transaction ID,Date,Customer ID,,Age,,,)
(1,2023-11-24,CUST001,,34,,3.0,50.0)
(2,2023-02-27,CUST002,,26,,2.0,500.0)
(3,2023-01-13,CUST003,,50,,1.0,30.0)
(4,2023-05-21,CUST004,,37,,1.0,500.0)
(5,2023-05-06,CUST005,,30,,2.0,50.0)
(6,2023-04-25,CUST006,,45,,1.0,30.0)
(7,2023-03-13,CUST007,,46,,2.0,25.0)
(8,2023-02-22,CUST008,,30,,4.0,25.0)
(9,2023-12-13,CUST009,,63,,2.0,300.0)
(10,2023-10-07,CUST010,,52,,4.0,50.0)
```

- **Calcula cuántas transacciones totales tiene el *dataset* (COUNT).**

```
grunt> total = GROUP ventas ALL;
grunt> count_total = FOREACH total GENERATE COUNT(ventas) AS total_transactions;
grunt> STORE count_total INTO '/user/maría_dev/ventas_analisis/total_transactions' USING PigStorage('\
t');
```

```
ut paths to process : 1
2025-11-13 08:56:27,530 [main] INFO
otal input paths to process : 1
(1001)
grunt>
```

2. **Filtrado por rango de edad**
   - Filtra los clientes con edad mayor de 30 años y guarda en alias `clientes_mayores30`.

```
grunt> clientes_mayores30 = FILTER ventas BY age > 30;
grunt> STORE clientes_mayores30 INTO '/user/maria_dev/ventas_analisis/clientes_mayores30' USING PigSto
rage(',');
```

   - Utiliza `LIMIT` para ver los primeros 10 resultados.

```
grunt> primeros10 = LIMIT clientes_mayores30 10;
grunt> DUMP primeros10;
```

   - ¿Qué porcentaje del total de transacciones corresponde a clientes mayores de 30?
     Total

```
grunt> total = GROUP ventas ALL;
grunt> count_total = FOREACH total GENERATE COUNT(ventas) AS total_transacciones;
grunt> DUMP count_total;
```

```
2025-11-14
o process :
(1001)
grunt>
```

Mayores de 30

```
grunt> g_mayores = GROUP clientes_mayores30 ALL;
grunt> count_mayores = FOREACH g_mayores GENERATE COUNT(clientes_mayores30) AS trans_mayores30;
grunt> DUMP count_mayores;
```

3. **Transformación de campos** ○ A partir del conjunto original, crea un alias

   donde generes:

   1. el género en mayúsculas (`UPPER(gender)`),

2. una nueva columna `importe_descuento` que calcule, por ejemplo, `price_per_unit * quantity * 0.90` (aplicando un 10% de "descuento ficticio").

```
grunt> ventas_trans = FOREACH ventas GENERATE    transaction_id,    date,    customer_id,    age,    product_category,
  quantiti,    price_per_unit,    (quantiti * price_per_unit) AS total_calc,    (quantiti * price_per_unit * 0.90) AS to
tal_descuento;
```

- Muestra los primeros 20 registros resultantes.

4. Agrupación y agregación por categoría de producto

- Agrupa por `product_category`.

```
grunt> cat_stats = FOREACH grp_cat GENERATE    group AS categoria,    COUNT(ventas) AS n_transacciones,    SUM(ventas.to
tal_amount) AS suma_total,    AVG(ventas.age) AS edad_media;
```

```
Vertex Stats:
VertexId Parallelism TotalTasks   InputRecords   ReduceInputRecords  OutputRecords  FileBytesRead FileBytesWritten  Hdfs
BytesRead HdfsBytesWritten Alias        Feature Outputs
scope-47           1          1           1001                   0           1001            32             615
    51673                0 cat_stats,grp_cat,ventas
scope-48           1          1              0                  48             96           647            1103
        0                0 cat_stats,cat_stats_sorted   GROUP_BY,SAMPLER
scope-57           1          1              0                  48              1           527             139
        0                0
scope-67           1          1             49                   0             48           747             570
        0                0 cat_stats_sorted
scope-69          -1          1              0                  48             48           570               0
        0              666      ORDER_BY        /user/maria_dev/retail/category_stats,

Input(s):
Successfully read 1001 records (51673 bytes) from: "/user/maria_dev/retail/retail_sales_dataset.csv"

Output(s):
Successfully stored 48 records (666 bytes) in: "/user/maria_dev/retail/category_stats"
```

- Para cada categoría calcula: número de transacciones (`COUNT`), suma de `total_amount` (`SUM`), edad promedio de cliente (`AVG(age)`).

```
grunt> cat_stats_sorted = ORDER cat_stats BY suma_total DESC;
```

- Ordena el resultado por la suma de `total_amount` descendente.

```
grunt> STORE cat_stats_sorted INTO '/user/maria_dev/retail/category_stats' USING PigStorage(',');
```

5. Extracción de categorías distintas

- En este *dataset* extrae las categorías de producto distintas (`DISTINCT product_category`).

```
grunt> distinct_cats = DISTINCT (FOREACH ventas GENERATE product_category);
grunt> DUMP distinct_cats;
```

- Pregunta: ¿Cuántas categorías diferentes hay?

```
grunt> distinct_cats = DISTINCT (FOREACH ventas GENERATE product_category);
grunt> grp_cats = GROUP distinct_cats ALL;
grunt> num_categorias = FOREACH grp_cats GENERATE COUNT(distinct_cats) AS total_categorias;
grunt> DUMP num_categorias;
```

```
Output(s):
Successfully stored 1 records (6 byt

2025-11-18 09:13:34,431 [main] INFO
s : 1
2025-11-18 09:13:34,431 [main] INFO
o process : 1
(48)
```

Hay 48 Categorias

6. Ordenación y obtención de *top*-transacciones

- Ordena todas las transacciones por `total_amount` descendente.

```
grunt> ordenado = ORDER ventas BY total_amount DESC;
```

- Usa `LIMIT` para extraer, por ejemplo, las 5 transacciones con mayor `total_amount`.

```
grunt> ordenado = ORDER ventas BY total_amount DESC;
grunt> top5 = LIMIT ordenado 5;
grunt>
```

- Muestra: `transaction_id`, `customer_id`, `product_category`, `total_amount`.

```
grunt> resultado_top5 =    FOREACH top5 GENERATE transaction_id, customer_id, product_category, total_amount;
grunt> DUMP resultado_top5;
```

```
Output(s):
Successfully stored 5 records (170 bytes) in: "hdfs://sandbox-hdp

2025-11-18 09:15:56,388 [main] INFO  org.apache.hadoop.mapreduce.
s : 1
2025-11-18 09:15:56,388 [main] INFO  org.apache.pig.backend.hadoo
o process : 1
(161,CUST161,64,500.0)
(164,CUST164,47,500.0)
(805,CUST805,30,500.0)
(808,CUST808,33,500.0)
(166,CUST166,34,500.0)
```

7. Uso de funciones de cadena

- Añade una nueva columna al alias original donde el `product_category` se recorte a los primeros 3 caracteres (`SUBSTRING(product_category,0,3)`) y otra que sea la longitud del `product_category` (`SIZE(product_category)`).

```
grunt> ventas_strings = FOREACH ventas GENERATE    transaction_id,    product_category,    SUBSTRING(product_category,0,
3) AS cat3,    SIZE(product_category) AS cat_length;
```

- Muestra los primeros 15 registros resultantes.

```
grunt> primeros15 = LIMIT ventas_strings 15;
grunt> DUMP primeros15;
s : 1
2025-11-18 09:17:43,244 [mai
o process : 1
(Transaction ID,Age,Age,3)
(1,34,34,2)
(2,26,26,2)
(3,50,50,2)
(4,37,37,2)
(5,30,30,2)
(6,45,45,2)
(7,46,46,2)
(8,30,30,2)
(9,63,63,2)
(10,52,52,2)
(11,23,23,2)
(12,35,35,2)
(13,22,22,2)
(14,64,64,2)
grunt>
```

8. Filtrado por fecha y condiciones combinadas

- Filtra primero las transacciones que se han hecho **antes de** una determinada fecha, por ejemplo, `date < '2023-07-01'`. (Suponiendo que el campo `date` es tipo `chararray` con formato `'YYYY-MM-DD'`).

```
grunt> ventas_fecha = FILTER ventas BY date < '2023-07-01';
grunt>
```

- De ese conjunto, filtra adicionalmente las transacciones con `total_amount > 500`.

```
grunt> ventas_fecha_mayor = FILTER ventas_fecha BY total_amount > 500;
2025-11-18 09:18:47,615 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_D
LE 1 time(s).
grunt> DUMP ventas_fecha_mayor;
```

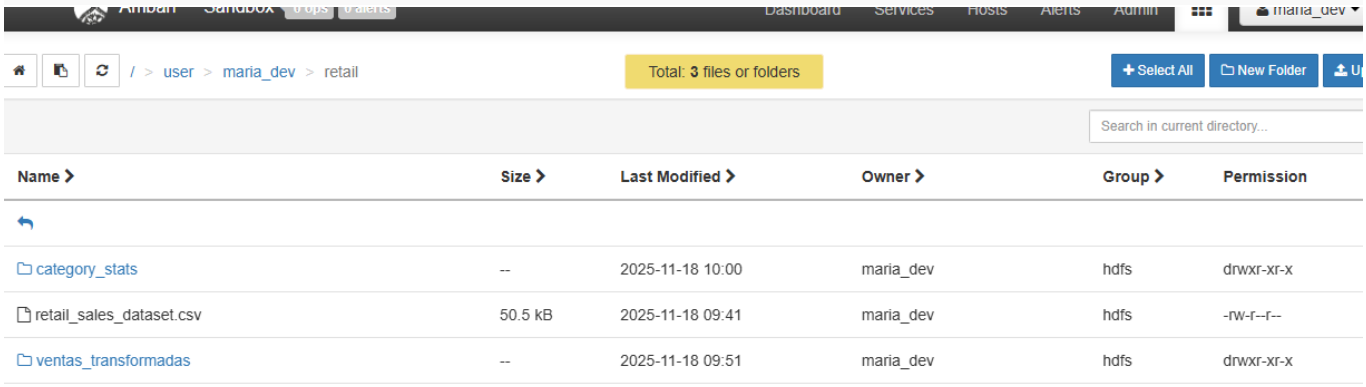
- Muestra el resultado, y calcula la edad promedio (`AVG(age)`) de los clientes que cumplen estas condiciones.

```
grunt> grp_age = GROUP ventas_fecha_mayor ALL;
grunt> edad_promedio = FOREACH grp_age GENERATE AVG(ventas_fecha_mayor.age) AS avg_age;
```

9. *Script* completo + almacenamiento

- Crea un script `.pig` que contenga los pasos: carga, filtrado, transformación, agrupación, ordenación, y finalmente almacenamiento (`STORE`) del resultado final en un directorio (por ejemplo `/usr/maria_dev/ventas_analisis`). Debes crear tu los filtros, transforamciones, etc. que desees.
- Asegúrate de comentar la operación de cada bloque del *script* con `--comentario`.
- Ejecuta el script en modo MapReduce estándar (`pig script.pig`). ○ Verifica los archivos de salida y comprueba que los resultados tienen sentido.

```
ventas = LOAD '/user/maria_dev/retail/retail_sales_dataset.csv'
USING PigStorage(',') AS (    transaction_id:chararray, date:chararray, customer_id:chararray, age:int, product_category:chararray, quantiti:int, price_per_unit:double, total_amount:double);


-- ============================================
-- FILTRO: Clientes mayores de 30
-- ============================================
clientes_mayores30 = FILTER ventas BY age > 30;


-- ============================================
-- TRANSFORMACIÓN: calcular total y total con descuento
-- ============================================
ventas_trans = FOREACH ventas GENERATE transaction_id, customer_id,product_category,quantiti,price_per_unit,(quantiti * price_per_unit) AS total_calculado, (quantiti * price_per_unit * 0.90) AS total_descuento;

-- ============================================
-- AGRUPACIÓN POR CATEGORÍA
-- ============================================
grp_cat = GROUP ventas BY product_category;

cat_stats = FOREACH grp_cat GENERATE group AS categoria, COUNT(ventas) AS n_transacciones, SUM(ventas.total_amount) AS suma_total, AVG(ventas.age) AS edad_media;


-- ============================================
-- ORDENAR POR SUMA TOTAL Y QUEDARSE EL TOP 5
-- ============================================
ordenado = ORDER ventas BY total_amount DESC;
top5 = LIMIT ordenado 5;


-- ============================================
-- GUARDAR RESULTADOS EN HDFS
-- ============================================
STORE clientes_mayores30 INTO '/user/maria_dev/ventas_analisis/clientes_mayores30' USING PigStorage(',');
STORE ventas_trans INTO '/user/maria_dev/ventas_analisis/ventas_transformadas' USING PigStorage(',');
STORE cat_stats INTO '/user/maria_dev/ventas_analisis/category_stats' USING PigStorage(',');
STORE top5 INTO '/user/maria_dev/ventas_analisis/top5_transactions' USING PigStorage(',');
```



CONTENIDO

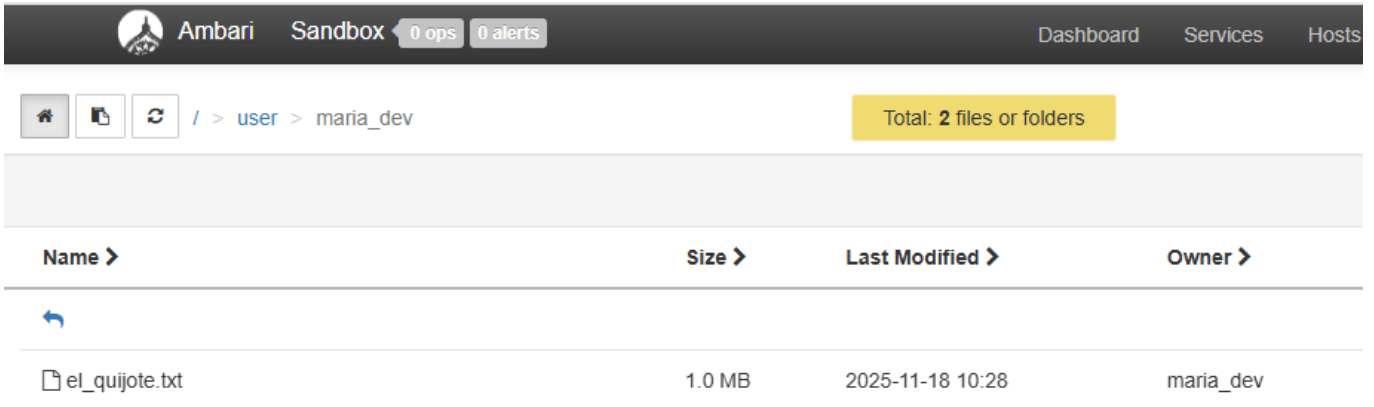
APARTADO C

INVESTIGA

Implementa un contador de palabras (cuantas veces aparece cada palabra en un texto)

1. Localiza en Internet una versión del ***Quijote*** en formato texto. Descárgala y cópiala a en tu sistema HDFS.

```
[maria_dev@sandbox-hdp retail]$ hdfs dfs -put el_quijote.txt /user/maria_dev/el_quijote.txt
```



- **Implementa en PIG el script necesario para hacer dicha operación. o Muestra un ejemplo de ejecución sobre El Quijote en pantalla.**

```
-- Cargar líneas del texto
quijote = LOAD '/user/maria_dev/el_quijote.txt' USING PigStorage('\n') AS (line:chararray);

-- Tokenizar
palabras = FOREACH quijote GENERATE FLATTEN(TOKENIZE(LOWER(line))) AS palabra;

-- Quitar tokens vacíos
limpias = FILTER palabras BY palabra IS NOT NULL AND palabra != '';

-- Agrupar y contar
grp = GROUP limpias BY palabra;
conteo = FOREACH grp GENERATE group AS palabra, COUNT(limpias) AS total;

-- Ordenar descendente
ordenado = ORDER conteo BY total DESC;

-- Guardar
STORE ordenado INTO '/user/maria_dev/pig_quijote' USING PigStorage('\t');
```

- Almacena la salida en una carpeta de HDFS llamada `/usr/maria_dev/pig_quijote`.

```
Input(s):
Successfully read 2186 records (1060259 bytes) from: "/user/maria_dev/el_quijote.txt"

Output(s):
Successfully stored 18581 records (208020 bytes) in: "/user/maria_dev/pig_quijote"
```

```
grunt> quijote = LOAD '/user/maria_dev/el_quijote.txt' USING PigStorage('\n') AS (line:chararray);
grunt> palabras = FOREACH quijote GENERATE FLATTEN(TOKENIZE(LOWER(line))) AS palabra;
2025-11-18 09:32:33,102 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (
PS Old Gen) of size 699400192 to monitor. collectionUsageThreshold = 489580128, usageThreshold = 48958
0128
grunt> limpias = FILTER palabras BY palabra IS NOT NULL AND palabra != '';
grunt> grp = GROUP limpias BY palabra;
grunt> conteo = FOREACH grp GENERATE group AS palabra, COUNT(limpias) AS total;
grunt> ordenado = ORDER conteo BY total DESC;
grunt> STORE ordenado INTO '/user/maria_dev/pig_quijote' USING PigStorage('\t');
```

el_quijote.txt	1.0 MB	2025-11-18 10:28	maria_dev	hdfs	-rw-r--r--
pig_quijote	--	2025-11-18 10:33	maria_dev	hdfs	drwxr-xr-x
retail	--	2025-11-18 10:00	maria_dev	hdfs	drwxr-xr-x