

2025



Big Data Aplicado

Profesor/a: José Manuel González Rodríguez

ACTIVIDAD EVALUABLE 2.2.- APACHE HIVE Y HIVEQL
01/04/2025

ÍNDICE

1.- Introducción	4
2.- Ejercicio 1: Instalación de YARN	4
2.1.- Modificación los ficheros de configuración	4
2.1.1.- Modificación del fichero mapred-site.xml	4
2.1.2.- Modificación del fichero yarn-site.xml	5
2.2. Configuración de variables de entorno	6
2.3. Inicio de YARN	8
2.4. Interfaz web de YARN	9
3.- Ejercicio 2: Instalación de Hive	9
3.1.- Descarga e instalación	9
3.2. Configuración de variables de entorno	11
3.3. Preparación de directorios en HDFS	12
3.4. Configuración de Hive	13
3.5. Inicialización del esquema	14
3.6. Modificación del archivo log4j-slf4j-impl-2.18.0.jar	15
3.7. Inicio del servidor Hive	16
3.8. Conexión con beeline	16
4.- Pruebas adicionales con HiveQL	18
5.- Ejercicio 3: Bases de datos con Hive	19
5.1.- Examinar los archivos CSV	19
5.1.1.- Sales.csv	19
5.1.2.- product.csv	19
5.1.3.- manufacturer.csv	20
5.2.- Crear los directorios en HDFS	20
5.4.- Cargar los archivos CSV a la máquina virtual	20
5.3.- Cargar los archivos CSV en HDFS	22
5.4.- Crear el script HQL fichero pmga.hql	23
5.5.- Ejecutar la primera parte del script	26
5.6.- Verificar los datos de las tablas	27
5.7.- Crear la tabla inventario con join archivo pmga2.hql	28
5.8.- Ejecutar la segunda parte del script fichero pmga2.hql	30
5.9.- Verificar la ubicación de la tabla inventario	31
5.10.- Respuesta a la pregunta: ¿En qué ubicación del sistema de ficheros HDFS se almacena la nueva tabla? ¿Por qué?	31
5.11.- Crear la tabla externa catalogo fichero pmga3.hql	32

5.12.- Consultar la tabla catalogo _____	33
5.13.- Resultado y explicación: _____	34
5.14.- Eliminar y crear la tabla catalogo fichero pmga4.hql_____	34
5.15.- Consultar la tabla catalogo _____	36
5.16.- Resultado y explicación: _____	36
5.17.- Eliminar la tabla inventario fichero pmga5.hql _____	37
5.18.- Resultado y explicación_____	39
5.19.- Verificar que los datos siguen en HDFS _____	39
6.- Ejercicio 4: Particionamiento _____	40
6.1.- Consulta para mostrar los distintos países y su frecuencia _____	40
6.2.- Crear una tabla particionada _____	41
6.3.- Poblar la tabla particionada _____	42
6.3.- Mostrar la estructura de subdirectorios HDFS _____	43
6.3.1.- Explicación razonada del resultado_____	43
6.4.- Comparar el rendimiento de las consultas_____	43
6.5.- Respuesta a las preguntas_____	46
7.- Ejercicio 5: Buckets _____	47
7.1.- Crear las tablas con y sin buckets _____	48
7.2.- Poblar las tablas con bucketing _____	49
7.3.- Poblar las tablas con no bucketing _____	53
7.4.- Estructura HDFS para ventas_buck (tabla MANAGED con buckets) _____	54
7.5.- Estructura HDFS para ventas_nobuck (tabla MANAGED sin buckets) _____	55
7.6.- Consulta en la tabla NO BUCKETED (ventas_nobuck) _____	55
7.7.- Consulta en la tabla BUCKETED (ventas_buck) _____	56
7.8.- Análisis Comparativo de Rendimiento: Eficiencia de Consultas con y sin Bucketing en Hive _____	57
7.8.1.- Diferencia de tiempo _____	57
7.8.2.- ¿Es lógico el resultado? _____	57
7.8.3.- Causas probables _____	57
7.8.4.- Conclusión _____	58
6.- Conclusiones _____	58
7.- Mapa Mental _____	60
8.- Bibliografia _____	61
9.- Anexo _____	61
9.1.- Descripción de los Ficheros HQL_____	61
9.2.- Documentos Adicionales _____	62

1.- Introducción

El objetivo de esta actividad es la instalación de la herramienta Hive, de manera que el metastore sea local (**embedded**), utilizando la base de datos por defecto (**Derby**). Además, se realizarán varias acciones básicas sobre la base de datos utilizando el lenguaje de consulta **HiveQL**.

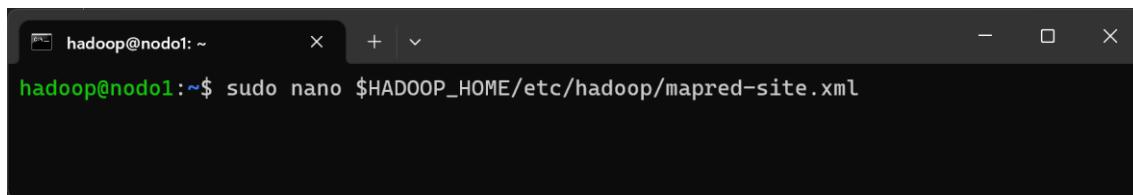
Para la realización de esta actividad, se ha utilizado la máquina virtual de Debian con la instalación de Hadoop descrita en el manual proporcionado en Moodle.

En este documento se detallan todos los pasos realizados para la correcta instalación y configuración de **YARN** y **Hive**, incluyendo capturas de pantalla y explicaciones de cada proceso.

2.- Ejercicio 1: Instalación de YARN

2.1.- Modificación los ficheros de configuración

2.1.1.- Modificación del fichero mapred-site.xml



```
hadoop@nodo1:~$ sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

En esta pantalla, se está editando el archivo de configuración de MapReduce (**mapred-site.xml**) utilizando el editor de texto Nano con privilegios de administrador (**sudo**), permitiendo ajustar parámetros críticos como el framework de procesamiento (**YARN**) y las rutas de ejecución de tareas.

```

hadoop@nodo1: ~
GNU nano 7.2          /opt/hadoop/hadoop-3.4.0/etc/hadoop/mapred-site.xml

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.application.classpath</name>
<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>
</property>
</configuration>

^G Ayuda      ^O Guardar      ^W Buscar      ^K Cortar      ^T Ejecutar      ^C Ubicación      M-U Deshacer
^X Salir      ^R Leer fich.  ^\ Reemplazar  ^U Pegar       ^J Justificar  ^/ Ir a línea M-E Rehacer

```

En esta pantalla, se están configurando las propiedades clave en el archivo **mapred-site.xml** mediante el editor nano, especificando el uso de **YARN** como framework para MapReduce (**mapreduce.framework.name**) y definiendo las rutas del classpath (**yarn.app.mapreduce.am.env** y **mapreduce.application.classpath**) para garantizar la integración correcta con el entorno Hadoop y procedemos a pulsar la tecla CTRL+O para guardar el fichero y CTRL+X para salir de nano.

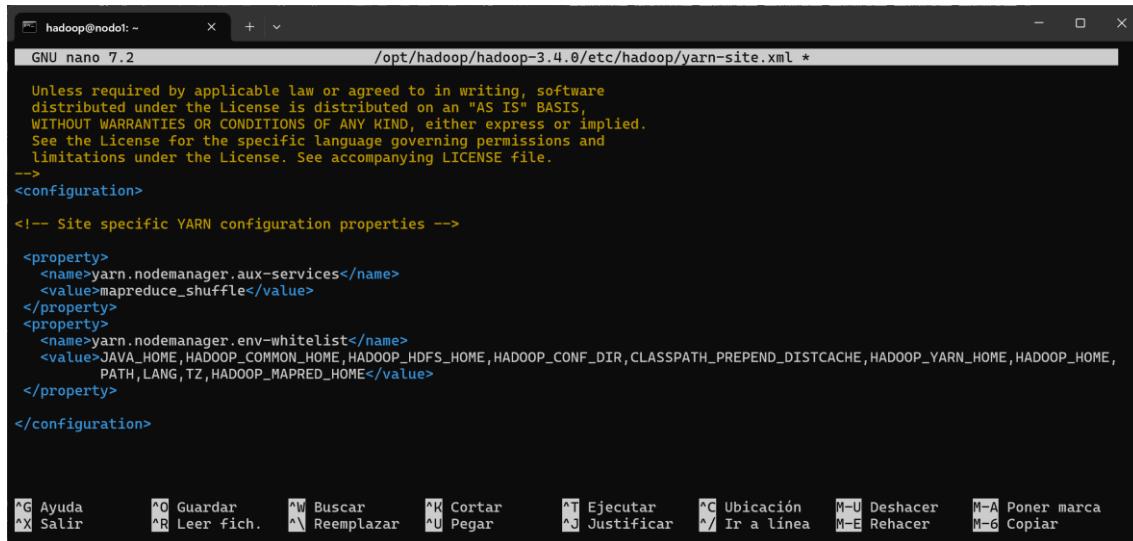
2.1.2.- Modificación del fichero yarn-site.xml

```

hadoop@nodo1: ~$ sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml

```

En esta pantalla, se está abriendo el archivo de configuración de **YARN** (**yarn-site.xml**) con el editor de texto nano y permisos de administrador (**sudo**) para modificar sus propiedades.



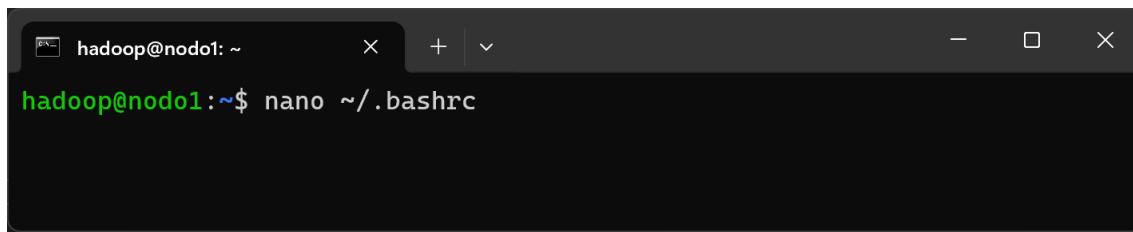
The screenshot shows a terminal window titled "hadoop@nodo1: ~". The command "nano /opt/hadoop/hadoop-3.4.0/etc/hadoop/yarn-site.xml" is running. The content of the file is displayed, including the Apache License header and configuration properties for YARN. The nano editor's status bar at the bottom shows various keyboard shortcuts.

```
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.

-->
<configuration>
  <!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.env-whitelist</name>
    <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PATH,LANG,TZ,HADOOP_MAPRED_HOME</value>
  </property>
</configuration>
```

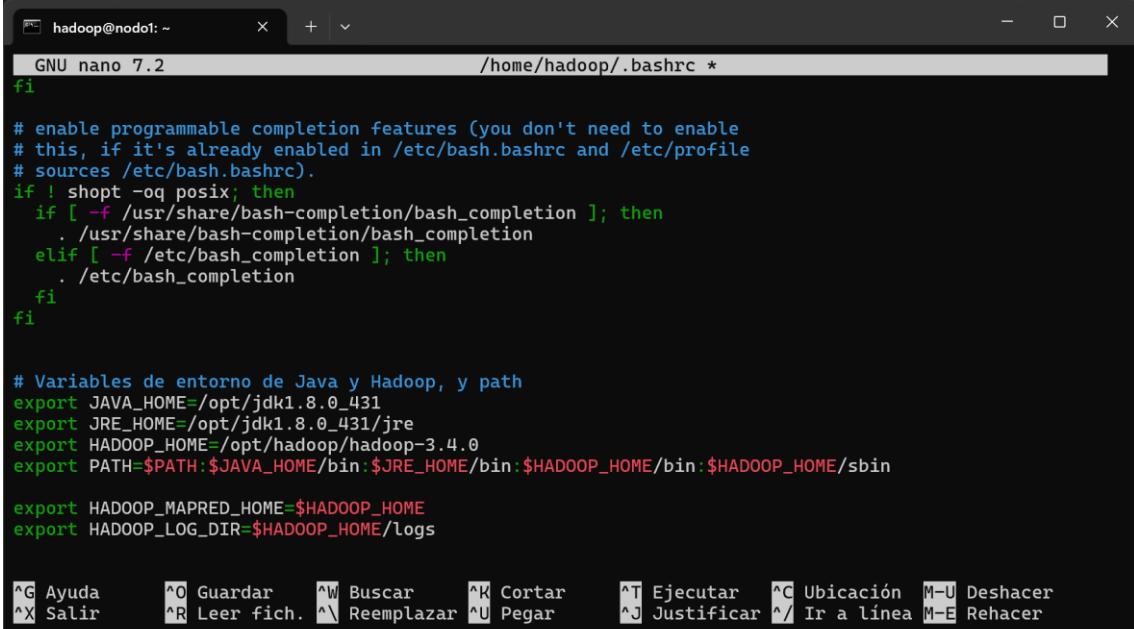
En esta pantalla, se están añadiendo las propiedades de configuración de **YARN** en el archivo **yarn-site.xml** para habilitar el servicio **mapreduce_shuffle** y definir las variables de entorno permitidas para los contenedores.

2.2. Configuración de variables de entorno



The screenshot shows a terminal window titled "hadoop@nodo1: ~". The command "nano ~/.bashrc" is run, opening the user's shell configuration file in the nano editor. The terminal is currently empty, showing only the command prompt.

En esta pantalla, se está abriendo el archivo de configuración del shell del usuario (**.bashrc**) con el editor de texto nano para añadir o modificar variables de entorno.



```

hadoop@nodo1: ~          + | 
GNU nano 7.2              /home/hadoop/.bashrc *
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

# Variables de entorno de Java y Hadoop, y path
export JAVA_HOME=/opt/jdk1.8.0_431
export JRE_HOME=/opt/jdk1.8.0_431/jre
export HADOOP_HOME=/opt/hadoop/hadoop-3.4.0
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_LOG_DIR=$HADOOP_HOME/logs

^G Ayuda      ^O Guardar      ^W Buscar      ^K Cortar      ^T Ejecutar      ^C Ubicación      M-U Deshacer
^X Salir      ^R Leer fich.  ^N Reemplazar  ^U Pegar       ^J Justificar  ^/ Ir a línea  M-E Rehacer

```

En esta pantalla, se procede a configurar las variables de entorno **HADOOP_MAPRED_HOME** y **HADOOP_LOG_DIR** en el archivo de configuración de Hadoop para definir la ruta base de MapReduce (**\$HADOOP_HOME**) y especificar el directorio de registros (**\$HADOOP_HOME/logs**), respectivamente y procedemos a guardar los cambios CTRL+O y para salir pulsamos la tecla CTRL-X.

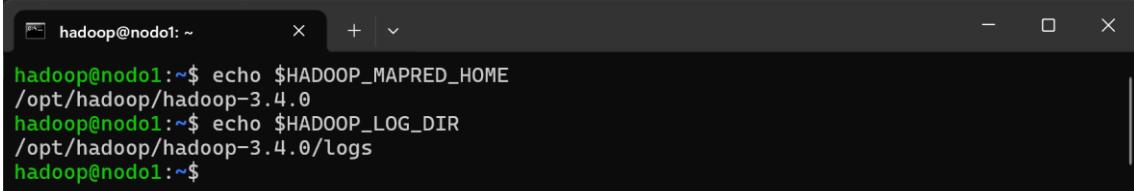


```

hadoop@nodo1: ~          + | 
hadoop@nodo1:~$ source ~/.bashrc
hadoop@nodo1:~$ 

```

En esta pantalla, se están aplicando los cambios realizados en el archivo **.bashrc** a la sesión actual de la terminal mediante el comando **source**.



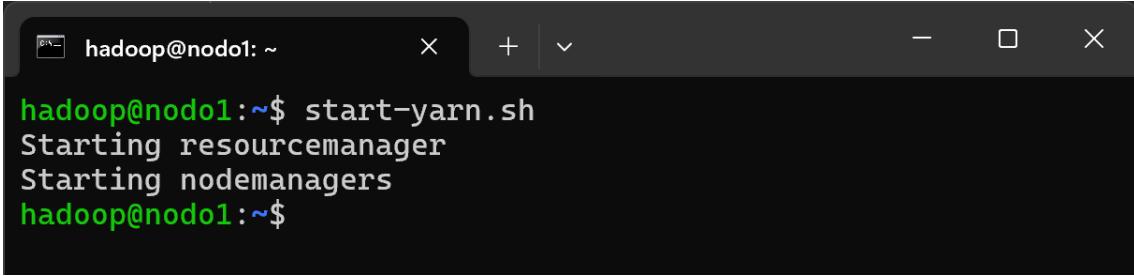
```

hadoop@nodo1: ~          + | 
hadoop@nodo1:~$ echo $HADOOP_MAPRED_HOME
/opt/hadoop/hadoop-3.4.0
hadoop@nodo1:~$ echo $HADOOP_LOG_DIR
/opt/hadoop/hadoop-3.4.0/logs
hadoop@nodo1:~$ 

```

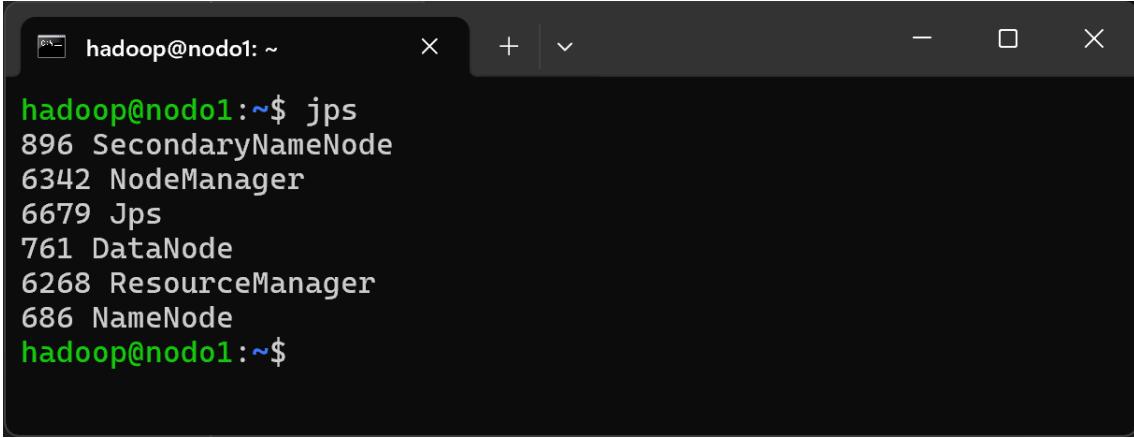
En esta pantalla, se están verificando los valores asignados a las variables de entorno **HADOOP_MAPRED_HOME** y **HADOOP_LOG_DIR** mediante el comando **echo**.

2.3. Inicio de YARN



```
hadoop@nodo1:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@nodo1:~$
```

En esta pantalla, se está ejecutando el script `start-yarn.sh` para iniciar los servicios de **YARN**, específicamente el **ResourceManager** y los **NodeManagers**.



```
hadoop@nodo1:~$ jps
896 SecondaryNameNode
6342 NodeManager
6679 Jps
761 DataNode
6268 ResourceManager
686 NameNode
hadoop@nodo1:~$
```

En esta pantalla, se está utilizando el comando `jps` para listar los procesos **Java** en ejecución y confirmar que los demonios de **HDFS** y **YARN** están activos.

2.4. Interfaz web de YARN

The screenshot shows the Hadoop YARN ResourceManager web interface. The URL is <http://192.168.0.158:8088/cluster>. The interface includes a navigation bar with links like Archivo, Editar, Ver, Historial, Marcadores, Herramientas, Ayuda, and a search bar. On the left, there's a sidebar with sections for Cluster (About, Nodes, Node Labels, Applications), Scheduler (NEW, NEW SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED), and Tools. The main content area displays Cluster Metrics, Cluster Nodes Metrics, and Scheduler Metrics. Under Scheduler Metrics, it shows the Capacity Scheduler configuration: Scheduling Resource Type [memory-mb (unit=Mi), vcores], Minimum Allocation <memory:1024, vCores:1>, Maximum Allocation <memory:8192, vCores:4>, and Maximum Cluster Capacity 0. Below these metrics, there's a table header for Application details: ID, User, Name, Application Type, Application Tags, Queue, Application Priority, StartTime, LaunchTime, FinishTime, State, FinalStatus, and Running Contain. A note at the bottom states "No data available in".

En esta pantalla, se accede a la interfaz web del **ResourceManager** de **YARN** mediante la **URL** <http://192.168.0.158:8088>, donde se monitorizan métricas del clúster (recursos disponibles, nodos activos) y el estado de aplicaciones **Hadoop** (ejecución, finalizadas o fallidas), incluyendo detalles como logs, configuraciones y estadísticas gráficas de uso de recursos.

3.- Ejercicio 2: Instalación de Hive

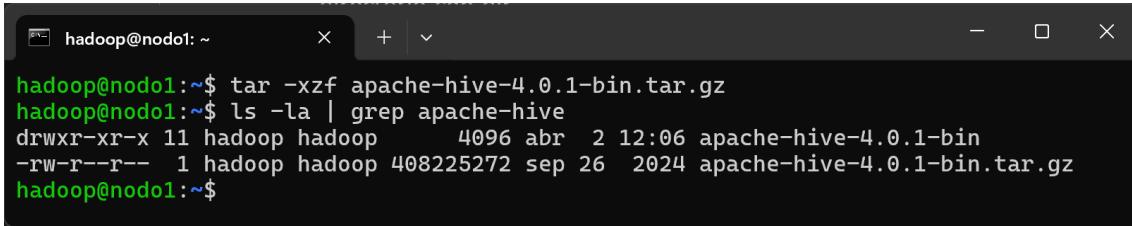
3.1.- Descarga e instalación

```
hadoop@nodo1: ~
hadoop@nodo1:~$ wget https://dlcdn.apache.org/hive/hive-4.0.1/apache-hive-4.0.1-bin.tar.gz
--2025-04-02 11:59:13-- https://dlcdn.apache.org/hive/hive-4.0.1/apache-hive-4.0.1-bin.tar.gz
Resolviendo dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Conectando con dlcdn.apache.org (dlcdn.apache.org)[151.101.2.132]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 408225272 (389M) [application/x-gzip]
Grabando a: «apache-hive-4.0.1-bin.tar.gz»

      apache-hive  0%[                               ]          0  --.-KB/s           Grabando a: «apache-hive
-4.0.1-bin.tar.gz»
apache-hive-4.0.1-bin.t 100%[=====] 389,31M 27,4MB/s   en 15s
2025-04-02 11:59:31 (26,5 MB/s) - «apache-hive-4.0.1-bin.tar.gz» guardado [408225272/408225272]

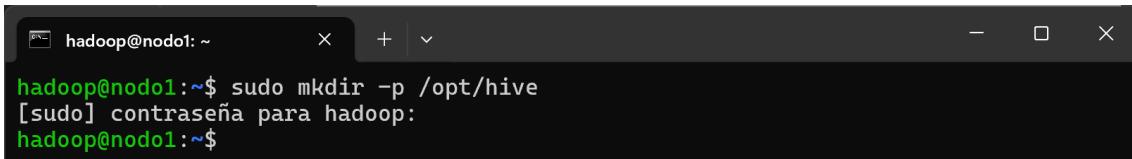
hadoop@nodo1:~$
```

En esta pantalla, se está descargando el archivo binario de Apache Hive 4.0.1 utilizando el comando “`wget https://dlcdn.apache.org/hive/hive-4.0.1/apache-hive-4.0.1-bin.tar.gz`”.



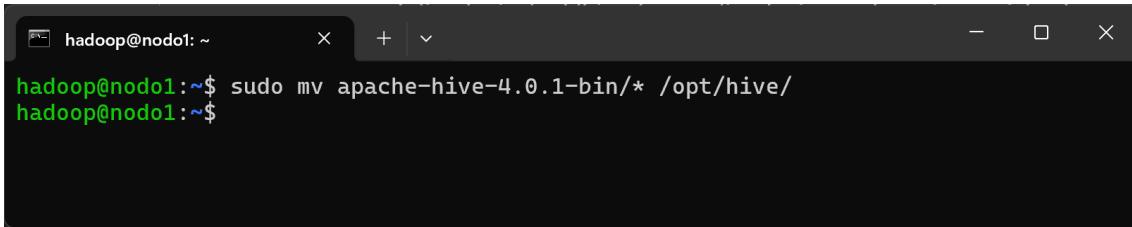
```
hadoop@nodo1:~$ tar -xzf apache-hive-4.0.1-bin.tar.gz
hadoop@nodo1:~$ ls -la | grep apache-hive
drwxr-xr-x 11 hadoop hadoop 4096 abr 2 12:06 apache-hive-4.0.1-bin
-rw-r--r-- 1 hadoop hadoop 408225272 sep 26 2024 apache-hive-4.0.1-bin.tar.gz
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `tar -xzf apache-hive-4.0.1-bin.tar.gz` para descomprimir el archivo de Hive, seguido de `ls -la | grep apache-hive` para confirmar la creación del directorio extraído y validar la existencia tanto del archivo comprimido original como de la nueva estructura de carpetas descomprimida.



```
hadoop@nodo1:~$ sudo mkdir -p /opt/hive
[sudo] contraseña para hadoop:
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `sudo mkdir -p /opt/hive` para crear el directorio `/opt/hive` (y sus directorios padres si no existen) con permisos de administrador.



```
hadoop@nodo1:~$ sudo mv apache-hive-4.0.1-bin/* /opt/hive/
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `sudo mv apache-hive-4.0.1-bin/* /opt/hive/` para mover todo el contenido del directorio descomprimido de Hive (`apache-hive-4.0.1-bin`) al directorio de instalación `/opt/hive/` con permisos de administrador.

```

hadoop@nodo1:~$ ls -la /opt/hive/
total 244
drwxr-xr-x 11 root root 4096 abr 2 12:15 .
drwxr-xr-x  5 root root 4096 abr 2 12:12 ..
drwxr-xr-x  3 hadoop hadoop 4096 abr 2 12:06 bin
drwxr-xr-x  2 hadoop hadoop 4096 abr 2 12:06 conf
drwxr-xr-x  2 hadoop hadoop 4096 abr 2 12:06 contrib
drwxr-xr-x  4 hadoop hadoop 4096 abr 2 12:03 examples
drwxr-xr-x  7 hadoop hadoop 4096 abr 2 12:03 hcatalog
drwxr-xr-x  2 hadoop hadoop 4096 abr 2 12:06 jdbc
drwxr-xr-x  4 hadoop hadoop 20480 abr 2 12:06 lib
-rw-r--r--  1 hadoop hadoop 20492 sep 25 2024 LICENSE
drwxr-xr-x  2 hadoop hadoop 4096 abr 2 12:06 licenses
-rw-r--r--  1 hadoop hadoop 148017 sep 25 2024 licenses.xml
-rw-r--r--  1 hadoop hadoop 165 sep 25 2024 NOTICE
-rw-r--r--  1 hadoop hadoop 7809 sep 25 2024 RELEASE_NOTES.txt
drwxr-xr-x  4 hadoop hadoop 4096 abr 2 12:03 scripts
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `ls -la /opt/hive/` para listar detalladamente el contenido del directorio `/opt/hive/`, verificando que los archivos de **Hive** se hayan movido correctamente.

3.2. Configuración de variables de entorno

```

hadoop@nodo1:~$ nano ~/.bashrc
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `nano ~/.bashrc` para abrir el archivo de configuración del **shell .bashrc** del usuario en el editor de texto `nano`, con el fin de añadir las variables de entorno de **Hive**.

```

GNU nano 7.2                               /home/hadoop/.bashrc *
export JRE_HOME=/opt/jdk1.8.0_431/jre
export HADOOP_HOME=/opt/hadoop/hadoop-3.4.0
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_LOG_DIR=$HADOOP_HOME/logs

# Configuración de Hive
export HIVE_HOME=/opt/hive
export PATH=$PATH:$HIVE_HOME/bin
```

En esta pantalla, se están añadiendo las variables de entorno **HIVE_HOME** y actualizando la variable **PATH** para incluir el directorio **bin** de **Hive** en el archivo `.bashrc` usando el editor `nano` y procedemos ha pulsar la tecla **CTRL+O** para guardar y **CTRL+X** para salir del `nano`.



```
hadoop@nodo1:~$ source ~/.bashrc
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `source ~/.bashrc` para aplicar los cambios (como las nuevas variables de entorno de **Hive**) ejecutando el archivo `.bashrc` a la sesión actual de la terminal.



```
hadoop@nodo1:~$ echo $HIVE_HOME
/opt/hive
hadoop@nodo1:~$
```

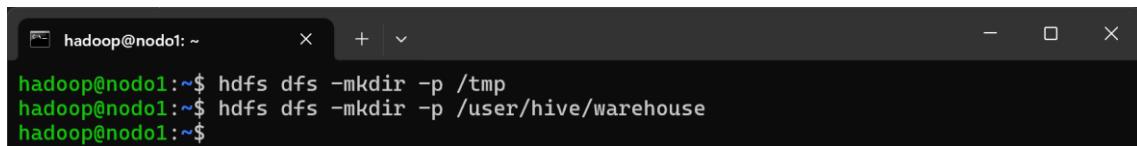
En esta pantalla, se ejecuta el comando `echo $HIVE_HOME` para verificar que la variable de entorno `HIVE_HOME` se ha establecido correctamente con el valor `/opt/hive`.

3.3. Preparación de directorios en HDFS



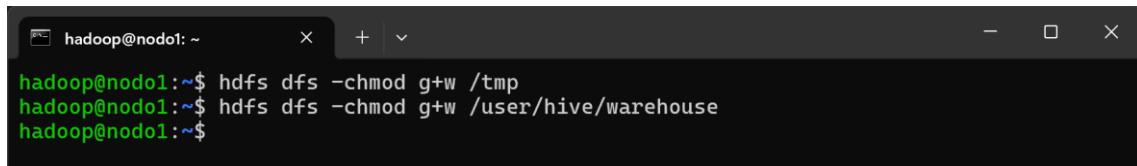
```
hadoop@nodo1:~$ jps
896 SecondaryNameNode
761 DataNode
6874 Jps
6268 ResourceManager
686 NameNode
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `jps` para verificar que los demonios principales de HDFS (`NameNode`, `DataNode`, `SecondaryNameNode`) y YARN (`ResourceManager`) están en ejecución antes de proceder con la configuración de **Hive**.



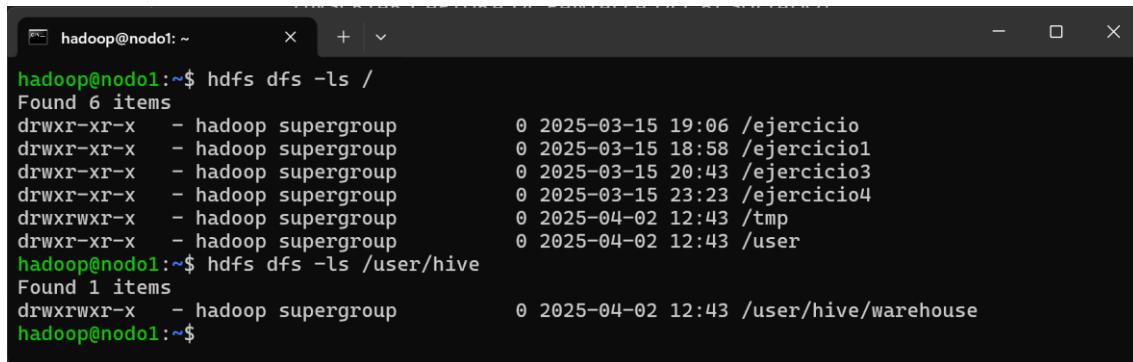
```
hadoop@nodo1:~$ hdfs dfs -mkdir -p /tmp
hadoop@nodo1:~$ hdfs dfs -mkdir -p /user/hive/warehouse
hadoop@nodo1:~$
```

En esta pantalla, se están creando los directorios `/tmp` y `/user/hive/warehouse` dentro del sistema de archivos HDFS usando los comandos `hdfs dfs -mkdir -p`, los cuales son necesarios para el funcionamiento de **Hive**.



```
hadoop@nodo1:~$ hdfs dfs -chmod g+w /tmp
hadoop@nodo1:~$ hdfs dfs -chmod g+w /user/hive/warehouse
hadoop@nodo1:~$
```

En esta pantalla, se están modificando los permisos en **HDFS** para los directorios **/tmp** y **/user/hive/warehouse** usando el comando **hdfs dfs -chmod g+w**, otorgando permisos de escritura al grupo propietario.

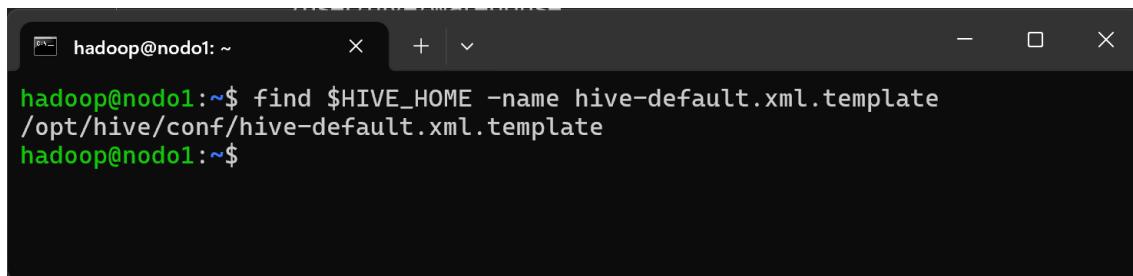


```

hadoop@nodo1:~$ hdfs dfs -ls /
Found 6 items
drwxr-xr-x  - hadoop supergroup          0 2025-03-15 19:06 /ejercicio
drwxr-xr-x  - hadoop supergroup          0 2025-03-15 18:58 /ejercicio1
drwxr-xr-x  - hadoop supergroup          0 2025-03-15 20:43 /ejercicio3
drwxr-xr-x  - hadoop supergroup          0 2025-03-15 23:23 /ejercicio4
drwxrwxr-x  - hadoop supergroup          0 2025-04-02 12:43 /tmp
drwxr-xr-x  - hadoop supergroup          0 2025-04-02 12:43 /user
hadoop@nodo1:~$ hdfs dfs -ls /user/hive
Found 1 items
drwxrwxr-x  - hadoop supergroup          0 2025-04-02 12:43 /user/hive/warehouse
hadoop@nodo1:~$
```

En esta pantalla, se están utilizando los comandos **hdfs dfs -ls /** y **hdfs dfs -ls /user/hive** para listar los contenidos de directorios en **HDFS** y verificar la creación y permisos de **/tmp** y **/user/hive/warehouse**.

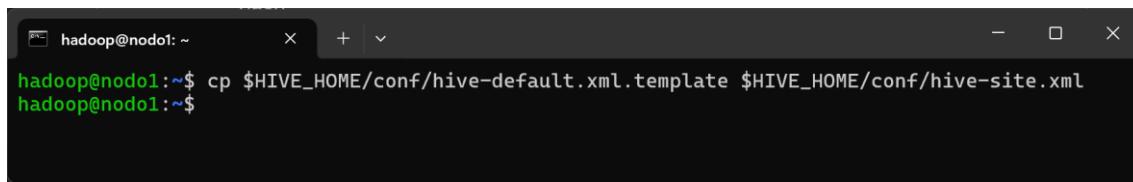
3.4. Configuración de Hive



```

hadoop@nodo1:~$ find $HIVE_HOME -name hive-default.xml.template
/opt/hive/conf/hive-default.xml.template
hadoop@nodo1:~$
```

En esta pantalla, se utiliza el comando **find \$HIVE_HOME -name hive-default.xml.template** para localizar la ruta exacta del archivo de plantilla de configuración **hive-default.xml.template** dentro del directorio de instalación de **Hive**.



```

hadoop@nodo1:~$ cp $HIVE_HOME/conf/hive-default.xml.template $HIVE_HOME/conf/hive-site.xml
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando **cp \$HIVE_HOME/conf/hive-default.xml.template \$HIVE_HOME/conf/hive-site.xml** para copiar el archivo de plantilla de configuración de **Hive** y crear el archivo de configuración activo **hive-site.xml**.

```

hadoop@nodo1:~$ sudo sed -i 's/${system:java.io.tmpdir}/\tmp/g' $HIVE_HOME/conf/hive-site.xml
[sudo] contraseña para hadoop:
hadoop@nodo1:~$ sudo sed -i 's/${system:user.name}/hadoop/g' $HIVE_HOME/conf/hive-site.xml
hadoop@nodo1:~$ sudo sed -i 's/<value>true</value><value>false</value>/g' $(grep -l "hive.server2.enable.doAs" $HIVE_HOME/conf/hive-site.xml)
hadoop@nodo1:~$
```

En esta pantalla, se ejecutan los comandos `sudo sed -i 's/\${system:java.io.tmpdir}/\tmp/g'`, `$HIVE_HOME/conf/hive-site.xml`, `sudo sed -i 's/\${system:user.name}/hadoop/g'`, `$HIVE_HOME/conf/hive-site.xml`, y `sudo sed -i 's/<value>true</value><value>false</value>/g'` (`$(grep -l "hive.server2.enable.doAs" $HIVE_HOME/conf/hive-site.xml)`) para modificar directamente el archivo `hive-site.xml`, sustituyendo las variables de directorio temporal y usuario, y cambiando el valor de la propiedad `hive.server2.enable.doAs` a false.

```

hadoop@nodo1:~$ grep -n "\tmp" $HIVE_HOME/conf/hive-site.xml | head -5
62:   <value>/tmp/hive</value>
430:   <value>/tmp/hadoop</value>
435:   <value>/tmp/${hive.session.id}_resources</value>
969:   enables SSL access. e.g. javax.net.ssl.trustStore=/tmp/truststore, javax.net.ssl.trustStorePassword=pwd.
2355:   <value>/tmp/hadoop</value>
hadoop@nodo1:~$ grep -n "hadoop" $HIVE_HOME/conf/hive-site.xml | head -5
67:   <value>/user/hadoop/repl/</value>
77:   <value>/user/hadoop/cmroot/</value>
118:   <value>/user/hadoop/repl/functions/</value>
430:   <value>/tmp/hadoop</value>
528:   org.apache.hadoop.hive ql.hooks.ExecuteWithHookContext interface.
hadoop@nodo1:~$ grep -n "hive.server2.enable.doAs" $HIVE_HOME/conf/hive-site.xml -A 2
6636:   <name>hive.server2.enable.doAs</name>
6637-   <value>false</value>
6638-   <description>
```

En esta pantalla, se ejecutan los comandos `grep -n "\tmp" $HIVE_HOME/conf/hive-site.xml | head -5`, `grep -n "hadoop" $HIVE_HOME/conf/hive-site.xml | head -5`, y `grep -n "hive.server2.enable.doAs" $HIVE_HOME/conf/hive-site.xml -A 2` para verificar, buscando las cadenas modificadas y la propiedad específica, si los cambios realizados previamente en el archivo `hive-site.xml` se aplicaron correctamente.

3.5. Inicialización del esquema

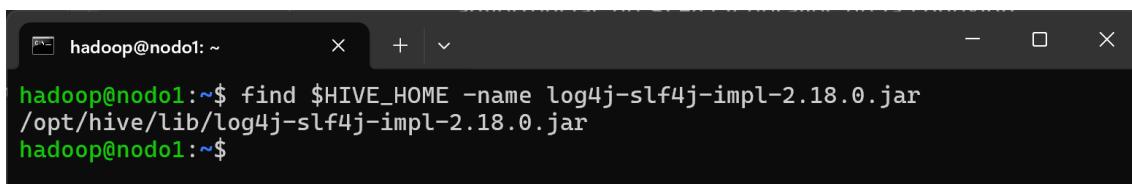
```

hadoop@nodo1:~$ schematool -dbType derby -initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/hadoop-3.4.0/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Initializing the schema to: 4.0.0
Metastore connection URL:      jdbc:derby:;databaseName=metastore_db;create=true
Metastore connection Driver :  org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:    APP
Starting metastore schema initialization to 4.0.0
Initialization script hive-schema-4.0.0.derby.sql
```

En esta pantalla, se ejecuta el comando `schematool -dbType derby -initSchema` para inicializar el esquema de la base de datos Derby que Hive usará como `metastore`, mostrando también advertencias de `SLF4J` y detalles de la conexión.

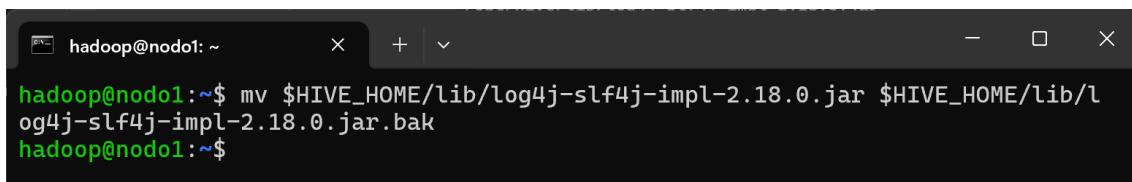
Este proceso tardó varios minutos en completarse. Una vez finalizado, se mostró un mensaje indicando que el esquema se ha inicializado correctamente.

3.6. Modificación del archivo log4j-slf4j-impl-2.18.0.jar



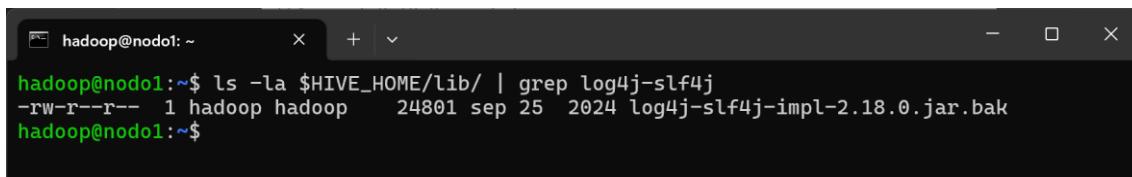
```
hadoop@nodo1:~$ find $HIVE_HOME -name log4j-slf4j-impl-2.18.0.jar
/opt/hive/lib/log4j-slf4j-impl-2.18.0.jar
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `find $HIVE_HOME -name log4j-slf4j-impl-2.18.0.jar` para localizar la ruta exacta del archivo JAR `log4j-slf4j-impl-2.18.0.jar` dentro del directorio de instalación de Hive.



```
hadoop@nodo1:~$ mv $HIVE_HOME/lib/log4j-slf4j-impl-2.18.0.jar $HIVE_HOME/lib/log4j-slf4j-impl-2.18.0.jar.bak
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `mv $HIVE_HOME/lib/log4j-slf4j-impl-2.18.0.jar $HIVE_HOME/lib/log4j-slf4j-impl-2.18.0.jar.bak` para renombrar el archivo JAR `log4j-slf4j-impl-2.18.0.jar` añadiéndole la extensión `.bak`, con el fin de evitar advertencias de `SLF4J`.



```
hadoop@nodo1:~$ ls -la $HIVE_HOME/lib/ | grep log4j-slf4j
-rw-r--r-- 1 hadoop hadoop 24801 sep 25 2024 log4j-slf4j-impl-2.18.0.jar.bak
hadoop@nodo1:~$
```

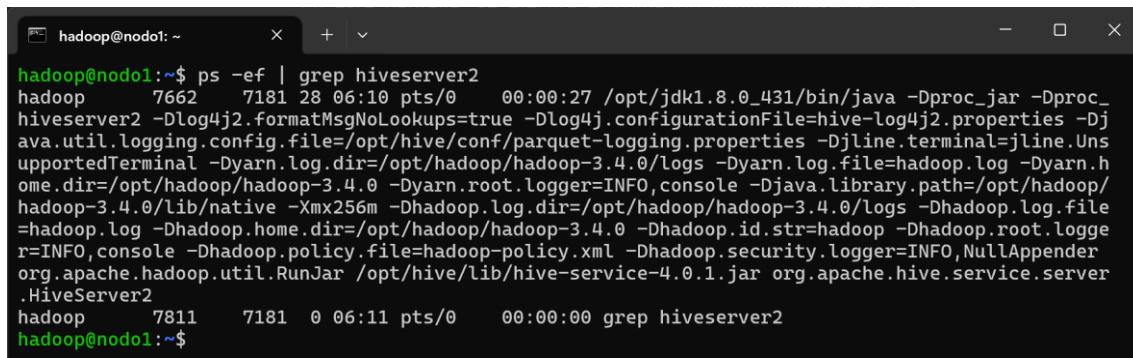
En esta pantalla, se ejecuta el comando `ls -la $HIVE_HOME/lib/ | grep log4j-slf4j` para listar los archivos en el directorio `lib` de Hive y filtrar la salida para confirmar que el archivo JAR `log4j-slf4j-impl-2.18.0.jar` ha sido renombrado con la extensión `.bak`.

3.7. Inicio del servidor Hive



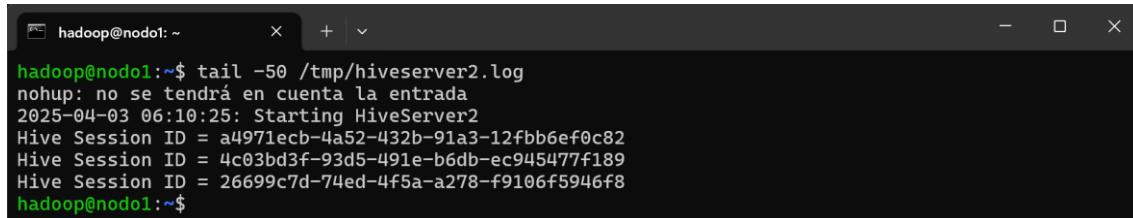
```
hadoop@nodo1:~$ nohup hiveserver2 > /tmp/hiveserver2.log 2>&1 &
[1] 7662
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `nohup hiveserver2 > /tmp/hiveserver2.log 2>&1 &` para iniciar el servicio **HiveServer2** en segundo plano, redirigiendo toda su salida (estándar y errores) al archivo `/tmp/hiveserver2.log` y asegurando que continúe ejecutándose, aunque se cierre la terminal.



```
hadoop@nodo1:~$ ps -ef | grep hiveserver2
hadoop    7662    7181 28 06:10 pts/0    00:00:27 /opt/jdk1.8.0_431/bin/java -Dproc_jar -Dproc_hiveserver2 -Dlog4j2.formatMsgNoLookups=true -Dlog4j.configurationFile=hive-log4j2.properties -Djava.util.logging.config.file=/opt/hive/conf/parquet-logging.properties -Djline.terminal=jline.UnsupportedTerminal -Dyarn.log.dir=/opt/hadoop/hadoop-3.4.0/logs -Dyarn.log.file=hadoop.log -Dyarn.home.dir=/opt/hadoop/hadoop-3.4.0 -Dyarn.root.logger=INFO,console -Djava.library.path=/opt/hadoop/hadoop-3.4.0/lib/native -Xmx256m -Dhadoop.log.dir=/opt/hadoop/hadoop-3.4.0/logs -Dhadoop.log.file=hadoop.log -Dhadoop.home.dir=/opt/hadoop/hadoop-3.4.0 -Dhadoop.id.str=hadoop -Dhadoop.root.logger=INFO,console -Dhadoop.policy.file=hadoop-policy.xml -Dhadoop.security.logger=INFO,NullAppender org.apache.hadoop.util.RunJar /opt/hive/lib/hive-service-4.0.1.jar org.apache.hive.service.server.HiveServer2
hadoop    7811    7181  0 06:11 pts/0    00:00:00 grep hiveserver2
hadoop@nodo1:~$
```

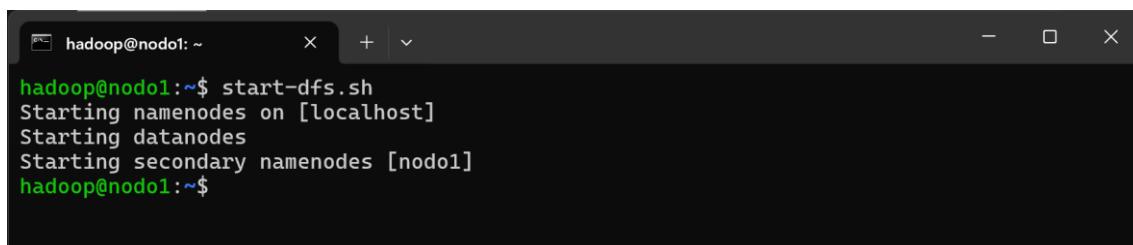
En esta pantalla, se ejecuta el comando `ps -ef | grep hiveserver2` para listar todos los procesos y filtrar por "hiveserver2", verificando así que el servicio **HiveServer2** (con **PID 7662**) está en ejecución.



```
hadoop@nodo1:~$ tail -50 /tmp/hiveserver2.log
nohup: no se tendrá en cuenta la entrada
2025-04-03 06:10:25: Starting HiveServer2
Hive Session ID = a4971ecb-4a52-432b-91a3-12fbb6ef0c82
Hive Session ID = 4c03bd3f-93d5-491e-b6db-ec945477f189
Hive Session ID = 26699c7d-74ed-4f5a-a278-f9106f5946f8
hadoop@nodo1:~$
```

En esta pantalla, se ejecuta el comando `tail -50 /tmp/hiveserver2.log` para mostrar las últimas 50 líneas del archivo de registro de **HiveServer2**, permitiendo verificar su estado y las últimas actividades, como los **IDs** de sesión creados.

3.8. Conexión con beeline



```
hadoop@nodo1:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [nodo1]
hadoop@nodo1:~$
```

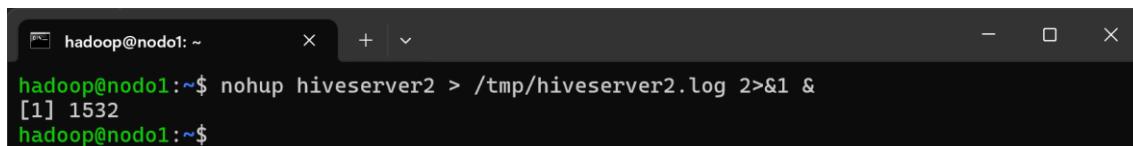
El comando `start-dfs.sh` inicia los demonios principales de Hadoop HDFS, incluyendo el **NameNode** en **localhost** (gestiona el sistema de archivos), los **DataNodes** (almacenan datos distribuidos), y el **Secondary NameNode** en **nodo1** (realiza tareas de respaldo del NameNode).



```
hadoop@nodo1: ~
hadoop@nodo1:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@nodo1:~$
```

A terminal window titled "hadoop@nodo1: ~". It shows the command "start-yarn.sh" being run, which starts the ResourceManager and NodeManagers services.

El comando `start-yarn.sh` inicia los demonios de **YARN** en **Hadoop**, incluyendo el **ResourceManager** (gestiona los recursos del clúster) y los **NodeManagers** (administran los contenedores en los nodos para ejecutar aplicaciones).



```
hadoop@nodo1: ~
hadoop@nodo1:~$ nohup hiveserver2 > /tmp/hiveserver2.log 2>&1 &
[1] 1532
hadoop@nodo1:~$
```

A terminal window titled "hadoop@nodo1: ~". It shows the command "nohup hiveserver2 > /tmp/hiveserver2.log 2>&1 &" being run, which starts the HiveServer2 process in the background.

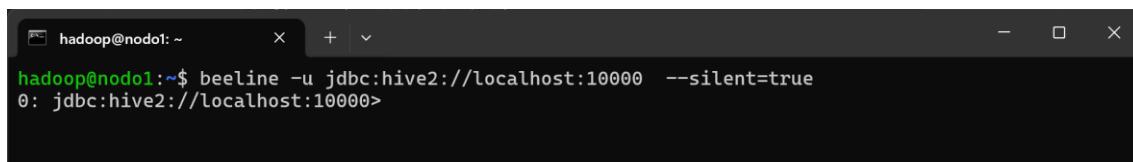
El comando `nohup hiveserver2 > /tmp/hiveserver2.log 2>&1 &` ejecuta el servidor **HiveServer2** en segundo plano, redirigiendo su salida estándar y errores al archivo `/tmp/hiveserver2.log`, permitiendo que el proceso continúe incluso si se cierra la sesión de terminal.



```
hadoop@nodo1: ~
hadoop@nodo1:~$ jps
656 NameNode
1713 Jps
1192 NodeManager
1113 ResourceManager
857 SecondaryNameNode
1532 RunJar
733 DataNode
hadoop@nodo1:~$
```

A terminal window titled "hadoop@nodo1: ~". It shows the output of the "jps" command, which lists various Hadoop processes running on the node.

El resultado muestra los procesos de **Hadoop** en ejecución en el nodo **nodo1**, cada uno con su respectivo **ID** de proceso, incluyendo servicios clave como **NameNode**, **DataNode**, **ResourceManager**, **NodeManager**, y otros relacionados con la gestión y procesamiento de datos distribuidos.

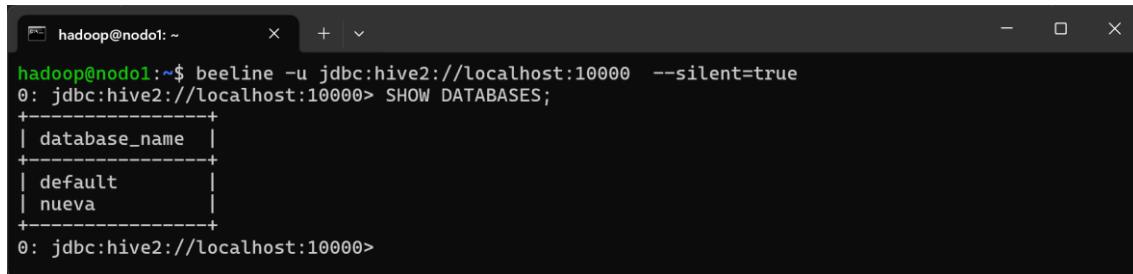


```
hadoop@nodo1: ~
hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true
0: jdbc:hive2://localhost:10000>
```

A terminal window titled "hadoop@nodo1: ~". It shows the command "beeline -u jdbc:hive2://localhost:10000 --silent=true" being run, which connects to the HiveServer2 instance.

El comando `beeline -u jdbc:hive2://localhost:10000 --silent=true` conecta el cliente **Beeline** al servidor **HiveServer2** en modo

silencioso (sin mostrar mensajes adicionales), permitiendo ejecutar consultas **SQL** en **Apache Hive** a través de **JDBC**.

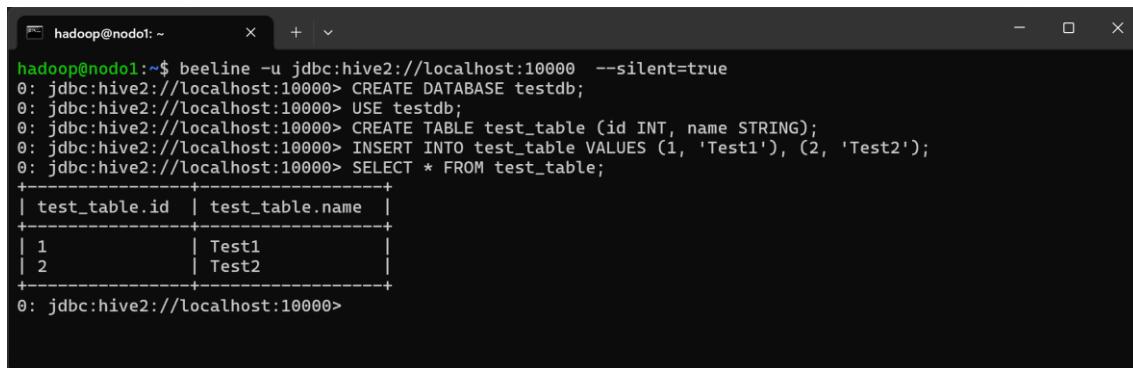


```
hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true
0: jdbc:hive2://localhost:10000> SHOW DATABASES;
+-----+
| database_name |
+-----+
| default      |
| nueva        |
+-----+
0: jdbc:hive2://localhost:10000>
```

El comando **SHOW DATABASES**; ejecutado en **Hive** muestra las bases de datos disponibles en el sistema, incluyendo la base de datos predeterminada **default** y otra llamada **nueva**, indicando que ambas están configuradas en el entorno de **Hive**

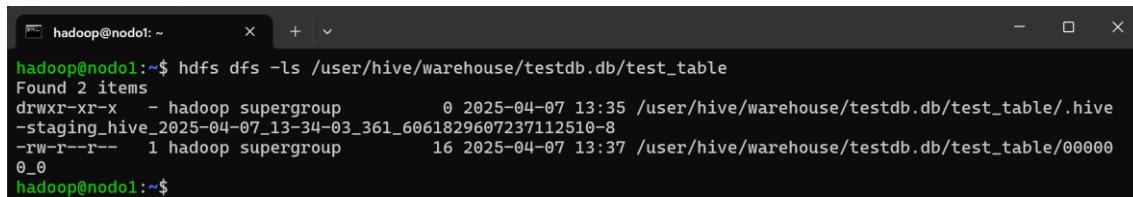
4.- Pruebas adicionales con HiveQL

Para verificar que Hive está funcionando correctamente, realicé algunas pruebas adicionales con **HiveQL**. Creé una nueva base de datos:



```
hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true
0: jdbc:hive2://localhost:10000> CREATE DATABASE testdb;
0: jdbc:hive2://localhost:10000> USE testdb;
0: jdbc:hive2://localhost:10000> CREATE TABLE test_table (id INT, name STRING);
0: jdbc:hive2://localhost:10000> INSERT INTO test_table VALUES (1, 'Test1'), (2, 'Test2');
0: jdbc:hive2://localhost:10000> SELECT * FROM test_table;
+-----+-----+
| test_table.id | test_table.name |
+-----+-----+
| 1            | Test1          |
| 2            | Test2          |
+-----+-----+
0: jdbc:hive2://localhost:10000>
```

Los comandos ejecutados en **Hive** crean una base de datos llamada **testdb**, establecen el contexto para trabajar dentro de ella, crean una tabla llamada **test_table** con columnas **id** (entero) y **name** (cadena), insertan dos registros (1, 'Test1' y 2, 'Test2') en la tabla, y finalmente consultan todos los datos almacenados en **test_table**, mostrando los resultados.



```
hadoop@nodo1:~$ hdfs dfs -ls /user/hive/warehouse/testdb.db/test_table
Found 2 items
drwxr-xr-x - hadoop supergroup          0 2025-04-07 13:35 /user/hive/warehouse/testdb.db/test_table/.hive-
-staging_hive_2025-04-07_13-34-03_361_6061829607237112510-8
-rw-r--r--  1 hadoop supergroup         16 2025-04-07 13:37 /user/hive/warehouse/testdb.db/test_table/00000
0_0
hadoop@nodo1:~$
```

El comando **hdfs dfs -ls** muestra el contenido del directorio **/user/hive/warehouse/testdb.db/test_table**

directorio en **HDFS** donde **Hive** almacena los datos de la tabla **test_table** dentro de la base de datos **testdb**, incluyendo un archivo con los datos insertados y un directorio temporal utilizado para operaciones de **staging**.

5.- Ejercicio 3: Bases de datos con Hive

5.1.- Examinar los archivos CSV

5.1.1.- Sales.csv

ProductID	Date	Zip	Units	Revenue	Country
725	1/15/1999	41540	1 115.5 Germany		
787	6/6/2002	41540	1 314.9 Germany		
788	6/6/2002	41540	1 314.9 Germany		
940	1/15/1999	22587	1 687.7 Germany		
396	1/15/1999	22587	1 857.1 Germany		
734	4/10/2003	22587	1 330.7 Germany		
769	2/15/1999	22587	1 257.2 Germany		
499	1/15/1999	12555	1 846.3 Germany		
2254	1/15/1999	40217	1 57.7 Germany		
31	5/31/2002	40217	1 761.2 Germany		
475	2/15/1999	13583	1 970.2 Germany		
510	1/15/1999	22587	1 837.1 Germany		

El archivo **sales.csv** es un archivo delimitado por el carácter **pipe** (|), con un encabezado en la primera línea, que contiene datos de ventas organizados en columnas como **ProductID**, **Date**, **Zip**, **Units**, **Revenue** y **Country**.

5.1.2.- product.csv

ProductID	Product	Category	Segment	ManufacturerID
1	Abbas MA-01	Mix	All Season	1
2	Abbas MA-02	Mix	All Season	1
3	Abbas MA-03	Mix	All Season	1
4	Abbas MA-04	Mix	All Season	1
5	Abbas MA-05	Mix	All Season	1
6	Abbas MA-06	Mix	All Season	1
7	Abbas MA-07	Mix	All Season	1
8	Abbas MA-08	Mix	All Season	1
9	Abbas MA-09	Mix	All Season	1
10	Abbas MA-10	Mix	All Season	1
11	Abbas MA-11	Mix	All Season	1
12	Abbas MA-12	Mix	All Season	1

El archivo **product.csv** es un archivo delimitado por comas (,), con un encabezado en la primera línea, que almacena información de productos incluyendo **ProductID**, nombre del producto (**Product**),

categoría (Category), segmento (Segment) y ID del fabricante (ManufacturerID).

5.1.3.- manufacturer.csv

```
manufacturer.csv
Archivo Editar Ver
ManufacturerID;Manufacturer
1;Abbas
2;Aliqui
3;Barba
4;Currus
5;Fama
6;Leo
7;VanArsdel
8;Natura
9;Palma
10;Pirum
11;Pomum
12;Orbius
Ln 1, Col 1 | 154 caracteres. | 100% | Windows (CRLF) | UTF-8
```

El archivo **manufacturer.csv** es un archivo delimitado por punto y coma (;), con un encabezado en la primera línea, que contiene información de fabricantes, incluyendo el **ID del fabricante (ManufacturerID)** y su **nombre (Manufacturer)**.

5.2.- Crear los directorios en HDFS

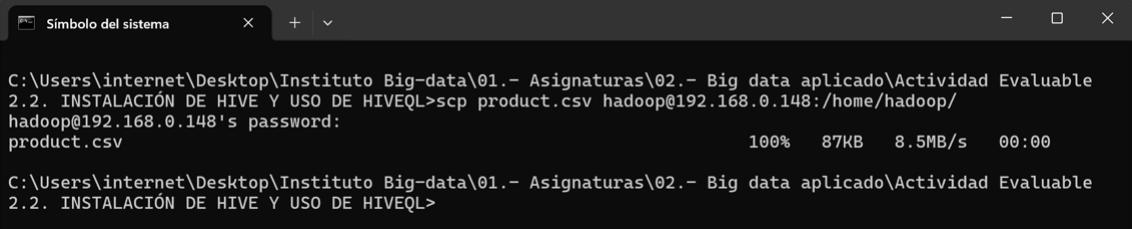
```
hadoop@nodo1: ~
hadoop@nodo1:~$ hdfs dfs -mkdir -p /user/pmga/sales
hadoop@nodo1:~$ hdfs dfs -mkdir -p /user/pmga/product
hadoop@nodo1:~$ hdfs dfs -mkdir -p /user/pmga/manufacturer
hadoop@nodo1:~$ hdfs dfs -mkdir -p /user/pmga/catalogo
hadoop@nodo1:~$
```

Abrimos una terminal y ejecutamos los comandos **hdfs dfs -mkdir -p /user/pmga/sales**, **hdfs dfs -mkdir -p /user/pmga/product**, **hdfs dfs -mkdir -p /user/pmga/manufacturer** y **hdfs dfs -mkdir -p /user/pmga/catalogo** para crear en **HDFS** los directorios necesarios donde se almacenarán los archivos **CSV** relacionados con **ventas**, **productos**, **fabricantes** y un **catálogo**.

5.4.- Cargar los archivos CSV a la máquina virtual

```
Símbolo del sistema
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable 2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>scp sales.csv hadoop@192.168.0.148:/home/hadoop/
sales.csv
/s   00:01
100%   38MB  29.2MB
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable 2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>
```

El comando `scp sales.csv` copia el archivo `sales.csv` desde el ordenador anfitrión al directorio `/home/hadoop/` en la máquina virtual con dirección IP `192.168.0.148`, utilizando una conexión segura basada en `SSH` y solicitando la contraseña del usuario remoto `hadoop`.



```
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable
2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>scp product.csv hadoop@192.168.0.148:/home/hadoop/
hadoop@192.168.0.148's password:
product.csv                                                 100%   87KB   8.5MB/s  00:00

C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable
2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>
```

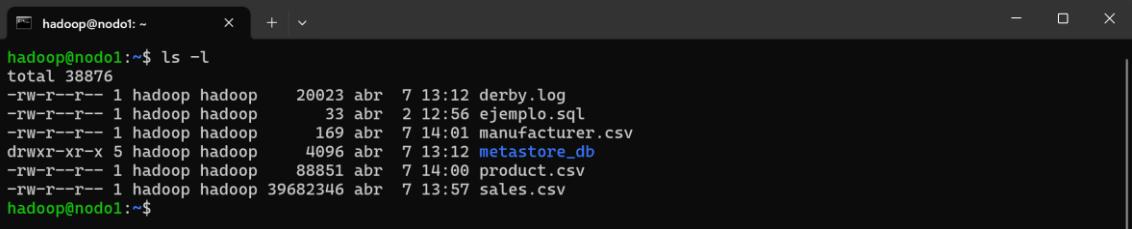
El comando `scp product.csv` copia el archivo `product.csv` desde el ordenador anfitrión al directorio `/home/hadoop/` en la máquina virtual con dirección IP `192.168.0.148`, utilizando una conexión segura basada en `SSH` y solicitando la contraseña del usuario remoto `hadoop`.



```
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable
2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>scp manufacturer.csv hadoop@192.168.0.148:/home/hadoop/
hadoop@192.168.0.148's password:
manufacturer.csv                                              100%  169    55.0KB/s  00:00

C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable
2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>
```

El comando `scp manufacturer.csv` transfiere el archivo `manufacturer.csv` desde el ordenador anfitrión al directorio `/home/hadoop/` en la máquina virtual con dirección IP `192.168.0.148`, utilizando una conexión segura basada en `SSH` y solicitando la contraseña del usuario remoto `hadoop`.

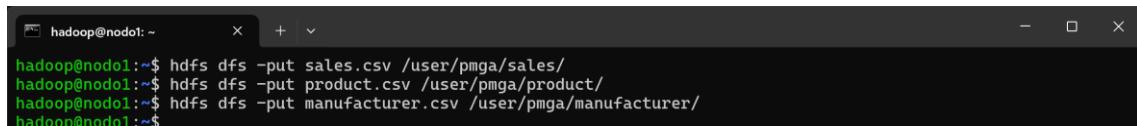


```
hadoop@nodo1:~$ ls -l
total 38876
-rw-r--r-- 1 hadoop hadoop  20023 abr  7 13:12 derby.log
-rw-r--r-- 1 hadoop hadoop    33 abr  2 12:56 ejemplo.sql
-rw-r--r-- 1 hadoop hadoop   169 abr  7 14:01 manufacturer.csv
drwxr-xr-x 5 hadoop hadoop  4096 abr  7 13:12 metastore_db
-rw-r--r-- 1 hadoop hadoop  88851 abr  7 14:00 product.csv
-rw-r--r-- 1 hadoop hadoop 39682346 abr  7 13:57 sales.csv
hadoop@nodo1:~$
```

El comando `ls -l` muestra el listado detallado de archivos y directorios en el directorio actual de la máquina virtual, confirmando que los archivos `manufacturer.csv`, `product.csv` y `sales.csv` han sido

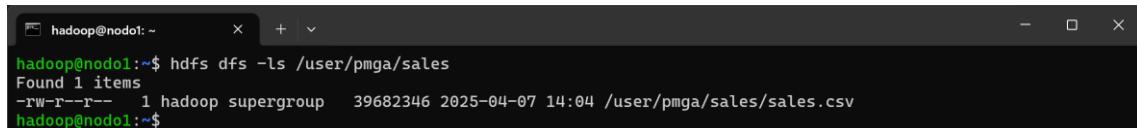
copiados correctamente, junto con información como permisos, propietario, tamaño y fecha de modificación.

5.3.- Cargar los archivos CSV en HDFS



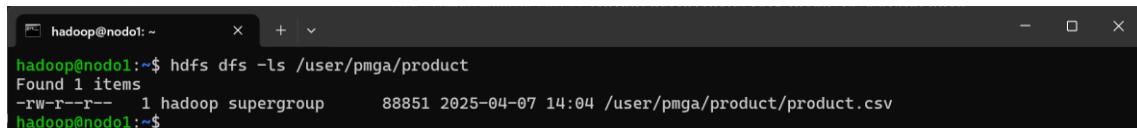
```
hadoop@nodo1:~$ hdfs dfs -put sales.csv /user/pmgasales/
hadoop@nodo1:~$ hdfs dfs -put product.csv /user/pmgaproduct/
hadoop@nodo1:~$ hdfs dfs -put manufacturer.csv /user/pmgamanufacturer/
hadoop@nodo1:~$
```

Los comandos `hdfs dfs -put sales.csv /user/pmgasales/`, `hdfs dfs -put product.csv /user/pmgaproduct/` y `hdfs dfs -put manufacturer.csv /user/pmgamanufacturer/` cargan los archivos CSV desde el sistema local a los directorios correspondientes en HDFS, asegurando que estén disponibles para su procesamiento en el entorno distribuido.



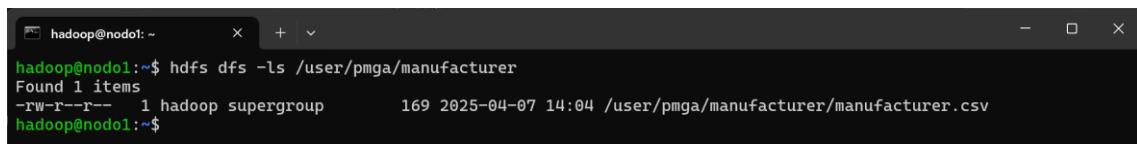
```
hadoop@nodo1:~$ hdfs dfs -ls /user/pmgasales
Found 1 items
-rw-r--r-- 1 hadoop supergroup 39682346 2025-04-07 14:04 /user/pmgasales/sales.csv
hadoop@nodo1:~$
```

El comando `hdfs dfs -ls /user/pmgasales` se utiliza para verificar que los archivos se han cargado correctamente en el directorio `/user/pmgasales` dentro del sistema de archivos HDFS, mostrando un listado detallado de los archivos y su estado.



```
hadoop@nodo1:~$ hdfs dfs -ls /user/pmgaproduct
Found 1 items
-rw-r--r-- 1 hadoop supergroup 88851 2025-04-07 14:04 /user/pmgaproduct/product.csv
hadoop@nodo1:~$
```

El comando `hdfs dfs -ls /user/pmgaproduct` verifica que los archivos se han cargado correctamente en el directorio `/user/pmgaproduct` dentro de HDFS, mostrando un listado detallado de los archivos y su estado.



```
hadoop@nodo1:~$ hdfs dfs -ls /user/pmgamanufacturer
Found 1 items
-rw-r--r-- 1 hadoop supergroup 169 2025-04-07 14:04 /user/pmgamanufacturer/manufacturer.csv
hadoop@nodo1:~$
```

El comando `hdfs dfs -ls /user/pmgamanufacturer` se utiliza para verificar que los archivos se han cargado correctamente en el directorio `/user/pmgamanufacturer` dentro de HDFS, mostrando un listado detallado de los archivos presentes en ese directorio.

5.4.- Crear el script HQL fichero pmga.hql

```
-- Establecer la propiedad para evitar que Hive convierta un join común en un map join  
SET hive.auto.convert.join=false;
```

El comando `SET hive.auto.convert.join=false;` desactiva la conversión automática de un `join` común a un `map join` en `Hive`, evitando que el optimizador realice esta transformación y manteniendo el procesamiento en la fase de reducción.

```
-- Crear la base de datos  
CREATE DATABASE IF NOT EXISTS pmga_db;
```

El código `CREATE DATABASE IF NOT EXISTS pmga_db;` crea una base de datos llamada `pmga_db` en `Hive` si aún no existe, asegurando que no se produzca un error si ya está creada.

```
-- Usar la base de datos  
USE pmga_db;
```

El código `USE pmga_db;` selecciona la base de datos `pmga_db` en `Hive` para que todas las operaciones y consultas posteriores se realicen dentro de esa base de datos.

```
-- Crear una tabla externa en Hive para representar los datos del archivo sales.csv
CREATE EXTERNAL TABLE IF NOT EXISTS sales (
    -- Definición de columnas y sus tipos de datos:
    ProductID INT,          -- Identificador del producto (entero)
    Fecha STRING,           -- Fecha de la venta (cadena de texto)
    Zip STRING,              -- Código postal (cadena de texto)
    Units INT,                -- Unidades vendidas (entero)
    Revenue DOUBLE,          -- Ingresos generados (número decimal)
    Country STRING           -- País de la venta (cadena de texto)
)
-- Especifica el formato de las filas en el archivo de datos
ROW FORMAT DELIMITED
-- Indica que los campos están separados por el carácter '|'
FIELDS TERMINATED BY '|'
-- Indica que los datos se almacenan como archivos de texto plano
STORED AS TEXTFILE
-- Especifica la ubicación en HDFS donde se encuentra el archivo sales.csv
LOCATION '/user/pmga/sales'
-- Propiedad de la tabla: indica que se debe omitir la primera línea del archivo (cabecera)
TBLPROPERTIES ("skip.header.line.count"="1");
```

El código crea una tabla externa en Hive llamada **sales**, definiendo su esquema con columnas como **ProductID**, **Fecha**, **Zip**, **Units**, **Revenue** y **Country**. Los datos están delimitados por el carácter **|** y almacenados como archivos de texto en la ubicación **/user/pmga/sales** en **HDFS**. Además, la propiedad **"skip.header.line.count"="1"** se establece para ignorar la primera línea del archivo, que contiene el encabezado.

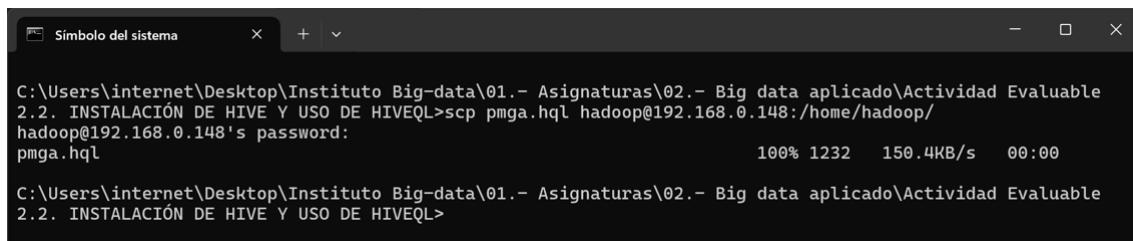
```
-- Crear una tabla externa en Hive para representar los datos del archivo product.csv
CREATE EXTERNAL TABLE IF NOT EXISTS product (
    -- Definición de las columnas y sus tipos de datos:
    ProductID INT,          -- Identificador único del producto (entero)
    Product STRING,           -- Nombre del producto (cadena de texto)
    Category STRING,          -- Categoría del producto (cadena de texto)
    Segment STRING,           -- Segmento al que pertenece el producto (cadena de texto)
    ManufacturerID INT       -- Identificador del fabricante (entero)
)
-- Especifica el formato de las filas en el archivo de datos
ROW FORMAT DELIMITED
-- Indica que los campos están separados por el carácter coma ','
FIELDS TERMINATED BY ','
-- Indica que los datos se almacenan como archivos de texto plano
STORED AS TEXTFILE
-- Especifica la ubicación en HDFS donde se encuentra el archivo product.csv
LOCATION '/user/pmga/product'
-- Propiedad de la tabla: indica que se debe omitir la primera línea del archivo (cabecera)
TBLPROPERTIES ("skip.header.line.count"="1");
```

El código crea una tabla externa en Hive llamada **product**, con columnas como **ProductID**, **Product**, **Category**, **Segment** y **ManufacturerID**. Los datos están delimitados por comas (,), almacenados como archivos de texto en la ubicación

`/user/pmga/product` en **HDFS**, y se configura la propiedad `"skip.header.line.count"="1"` para ignorar la primera línea del archivo, que contiene el encabezado.

```
-- Crear una tabla externa en Hive para representar los datos del archivo manufacturer.csv
CREATE EXTERNAL TABLE IF NOT EXISTS manufacturer (
    -- Definición de las columnas y sus tipos de datos:
    ManufacturerID INT,      -- Identificador único del fabricante (entero)
    Manufacturer STRING       -- Nombre del fabricante (cadena de texto)
)
-- Especifica el formato de las filas en el archivo de datos
ROW FORMAT DELIMITED
-- Indica que los campos están separados por el carácter punto y coma ;
FIELDS TERMINATED BY ;
-- Indica que los datos se almacenan como archivos de texto plano
STORED AS TEXTFILE
-- Especifica la ubicación en HDFS donde se encuentra el archivo manufacturer.csv
LOCATION '/user/pmga/manufacturer'
-- Propiedad de la tabla: indica que se debe omitir la primera línea del archivo (cabecera)
TBLPROPERTIES ("skip.header.line.count"="1");
```

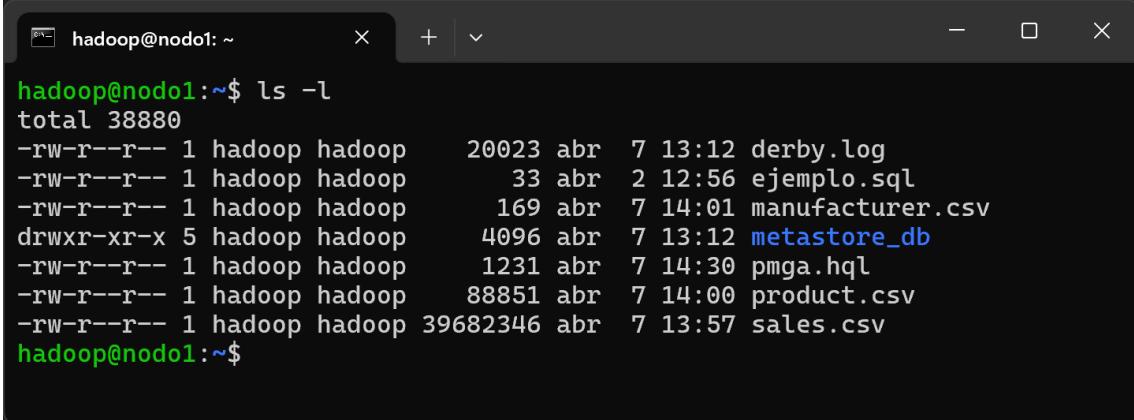
El código crea una tabla externa en **Hive** llamada **manufacturer**, con las columnas **ManufacturerID** y **Manufacturer**. Los datos están delimitados por punto y coma (;), almacenados como archivos de texto en la ubicación `/user/pmga/manufacturer` en **HDFS**, y se configura la propiedad `"skip.header.line.count"="1"` para ignorar la primera línea del archivo, que contiene el encabezado.



```
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable
2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>scp pmga.hql hadoop@192.168.0.148:/home/hadoop/
hadoop@192.168.0.148's password:
pmga.hql                                         100% 1232   150.4KB/s   00:00

C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable
2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>
```

El comando `scp pmga.hql` transfiere el archivo `pmga.hql` desde el ordenador anfitrión al directorio `/home/hadoop/` en la máquina virtual con dirección IP **192.168.0.148**, utilizando una conexión segura basada en **SSH** y solicitando la contraseña del usuario remoto **hadoop**.

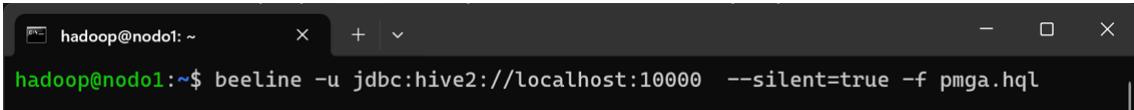


```

hadoop@nodo1:~$ ls -l
total 38880
-rw-r--r-- 1 hadoop hadoop 20023 abr  7 13:12 derby.log
-rw-r--r-- 1 hadoop hadoop     33 abr  2 12:56 ejemplo.sql
-rw-r--r-- 1 hadoop hadoop    169 abr  7 14:01 manufacturer.csv
drwxr-xr-x 5 hadoop hadoop  4096 abr  7 13:12 metastore_db
-rw-r--r-- 1 hadoop hadoop   1231 abr  7 14:30 pmga.hql
-rw-r--r-- 1 hadoop hadoop 88851 abr  7 14:00 product.csv
-rw-r--r-- 1 hadoop hadoop 39682346 abr  7 13:57 sales.csv
hadoop@nodo1:~$
```

El comando `ls -l` verifica que el archivo `pmga.hql` se ha copiado correctamente al directorio actual de la máquina virtual, mostrando sus detalles como permisos, propietario, tamaño y fecha de modificación.

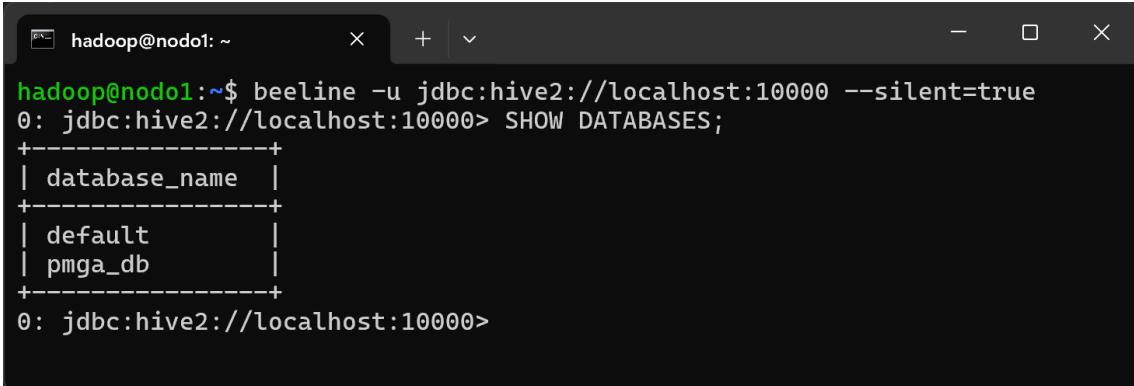
5.5.- Ejecutar la primera parte del script



```

hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga.hql
```

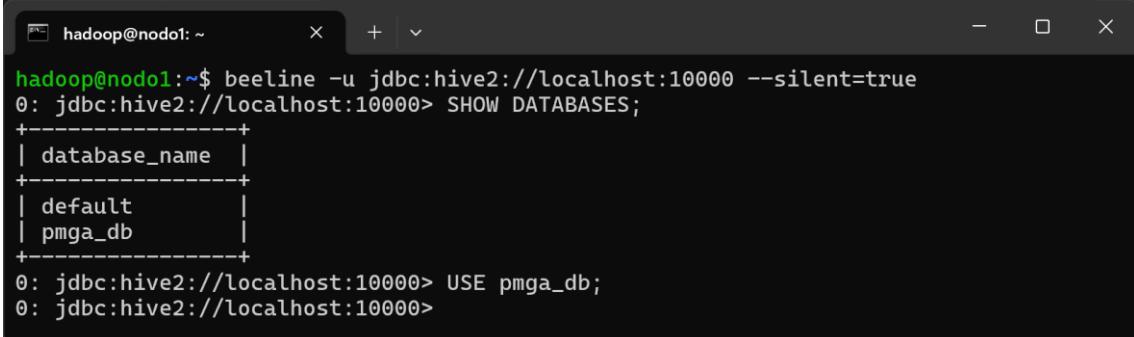
El comando `beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga.hql` ejecuta el archivo de script `HiveQL pmga.hql` utilizando `Beeline`, conectándose al servidor `HiveServer2` en `localhost` (`puerto 10000`) y ejecutando las consultas de manera silenciosa, sin mostrar mensajes innecesarios en la salida.



```

hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true
0: jdbc:hive2://localhost:10000> SHOW DATABASES;
+-----+
| database_name |
+-----+
| default      |
| pmga_db       |
+-----+
0: jdbc:hive2://localhost:10000>
```

En esta pantalla, se procede a ejercutar el comando `beeline -u jdbc:hive2://localhost:10000 --silent=true` y luego los comando `SHOW DATABASES;` verifica que la base de datos `pmga_db` se ha creado correctamente en `Hive`, ya que aparece listada junto con otras bases de datos como `default`.

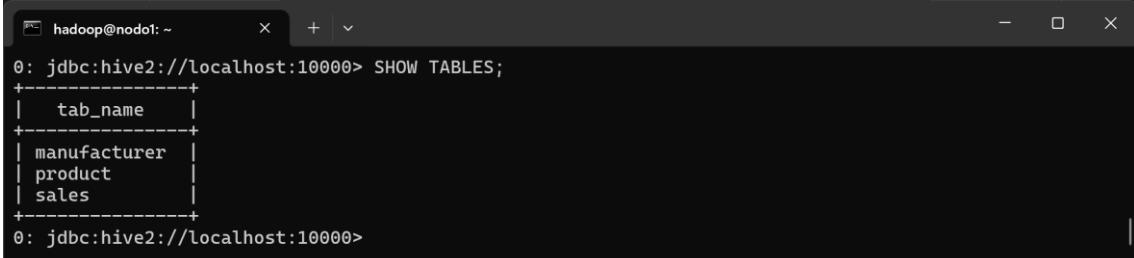


```

hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true
0: jdbc:hive2://localhost:10000> SHOW DATABASES;
+-----+
| database_name |
+-----+
| default      |
| pmga_db       |
+-----+
0: jdbc:hive2://localhost:10000> USE pmga_db;
0: jdbc:hive2://localhost:10000>

```

En esta pantalla, ejecutamos el comando **USE pmga_db;** para posicionarnos en la base de datos y presionamos la tecla Intro para ejecutarlo.



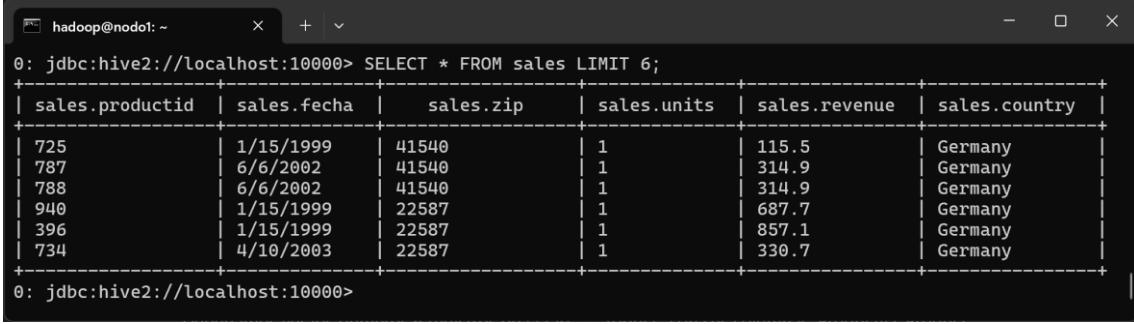
```

hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true
0: jdbc:hive2://localhost:10000> SHOW TABLES;
+-----+
| tab_name   |
+-----+
| manufacturer |
| product     |
| sales        |
+-----+
0: jdbc:hive2://localhost:10000>

```

En esta pantalla, ejecutamos el comando **SHOW TABLES;** para verificar que las tablas se hayan creado correctamente y presionamos la tecla Intro para ejecutarlo.

5.6.- Verificar los datos de las tablas

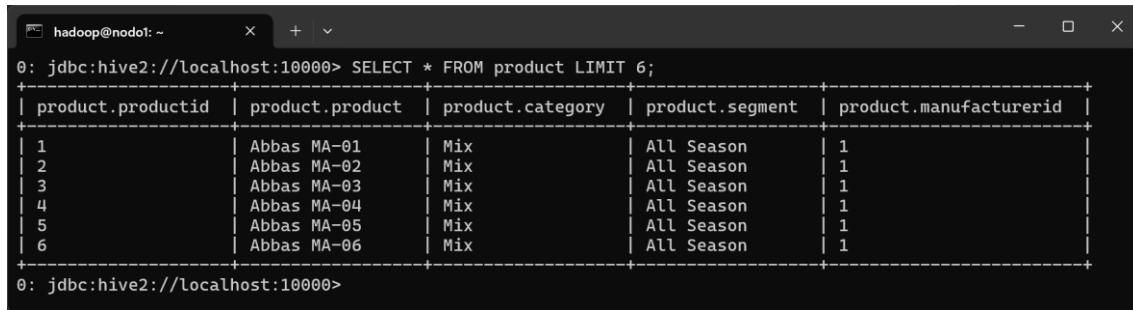


```

hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true
0: jdbc:hive2://localhost:10000> SELECT * FROM sales LIMIT 6;
+-----+-----+-----+-----+-----+-----+
| sales.productid | sales.fecha | sales.zip    | sales.units | sales.revenue | sales.country |
+-----+-----+-----+-----+-----+-----+
| 725            | 1/15/1999  | 41540       | 1           | 115.5        | Germany      |
| 787            | 6/6/2002   | 41540       | 1           | 314.9        | Germany      |
| 788            | 6/6/2002   | 41540       | 1           | 314.9        | Germany      |
| 940            | 1/15/1999  | 22587       | 1           | 687.7        | Germany      |
| 396            | 1/15/1999  | 22587       | 1           | 857.1        | Germany      |
| 734            | 4/10/2003  | 22587       | 1           | 330.7        | Germany      |
+-----+-----+-----+-----+-----+-----+
0: jdbc:hive2://localhost:10000>

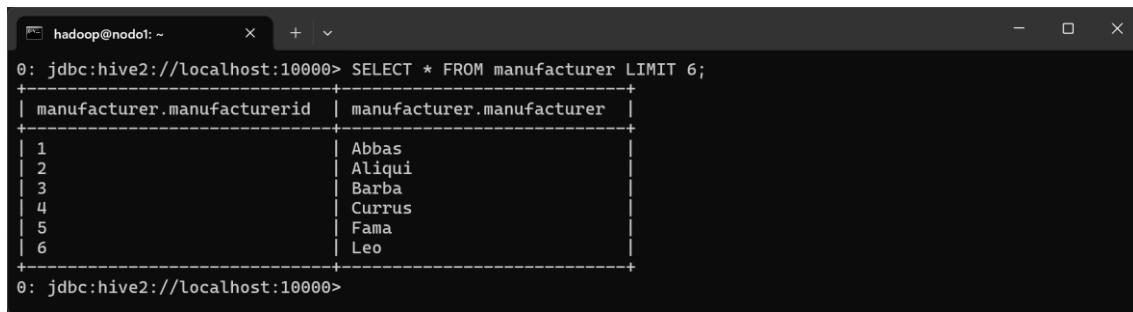
```

El comando **SELECT * FROM sales LIMIT 6;** muestra las primeras 6 filas de la tabla **sales** en **Hive**, incluyendo detalles como **ID** de producto, fecha, código postal, unidades vendidas, ingresos y país, con datos de ventas registrados en Alemania.



```
hadoop@nodo1: ~
0: jdbc:hive2://localhost:10000> SELECT * FROM product LIMIT 6;
+-----+-----+-----+-----+-----+
| product.productid | product.product | product.category | product.segment | product.manufacturerid |
+-----+-----+-----+-----+-----+
| 1 | Abbas MA-01 | Mix | All Season | 1 |
| 2 | Abbas MA-02 | Mix | All Season | 1 |
| 3 | Abbas MA-03 | Mix | All Season | 1 |
| 4 | Abbas MA-04 | Mix | All Season | 1 |
| 5 | Abbas MA-05 | Mix | All Season | 1 |
| 6 | Abbas MA-06 | Mix | All Season | 1 |
+-----+-----+-----+-----+-----+
0: jdbc:hive2://localhost:10000>
```

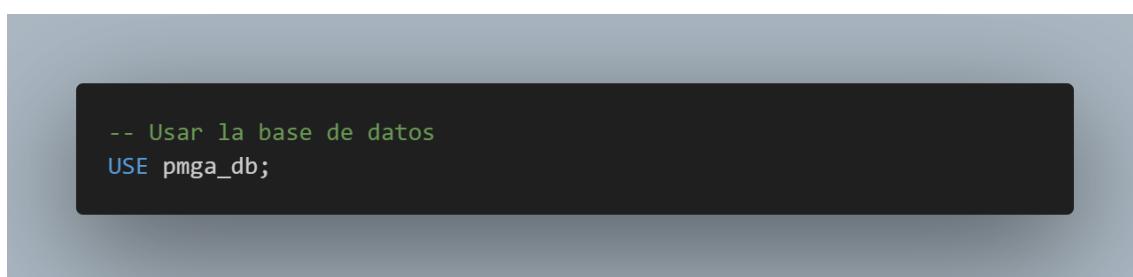
El comando `SELECT * FROM product LIMIT 6;` muestra las primeras 6 filas de la tabla `product` en `Hive`, incluyendo columnas como `ID` del producto, nombre, categoría, segmento y fabricante, con ejemplos de productos de la línea "Abbas MA" en la categoría "Mix" y segmento "All Season".



```
hadoop@nodo1: ~
0: jdbc:hive2://localhost:10000> SELECT * FROM manufacturer LIMIT 6;
+-----+-----+
| manufacturer.manufacturerid | manufacturer.manufacturer |
+-----+-----+
| 1 | Abbas |
| 2 | Aliqui |
| 3 | Barba |
| 4 | Currus |
| 5 | Fama |
| 6 | Leo |
+-----+-----+
0: jdbc:hive2://localhost:10000>
```

El comando `SELECT * FROM manufacturer LIMIT 6;` muestra las primeras 6 filas de la tabla `manufacturer` en `Hive`, listando los campos `manufacturerid` (identificador) y `manufacturer` (nombre del fabricante), con ejemplos como Abbas, Aliqui y Barba.

5.7.- Crear la tabla inventario con join archivo pmga2.hql



```
-- Usar la base de datos
USE pmga_db;
```

El comando `USE pmga_db;` selecciona la base de datos `pmga_db` en `Hive` como contexto activo para ejecutar consultas u operaciones posteriores, permitiendo acceder a sus tablas sin especificar el nombre de la base de datos en cada sentencia.

```
-- Establecer la propiedad para evitar map join  
SET hive.auto.convert.join=false;
```

El código `SET hive.auto.convert.join=false;` desactiva la conversión automática de **joins** a **map joins** en **Hive**, obligando a usar **joins** comunes (con **reducers**) incluso si una tabla es pequeña.

```
-- Crear una tabla interna en Hive llamada 'inventario'  
CREATE TABLE IF NOT EXISTS inventario (  
    -- Definición de las columnas y sus tipos de datos:  
    ProductID INT,          -- Identificador único del producto (entero)  
    Product STRING,         -- Nombre del producto (cadena de texto)  
    Category STRING,        -- Categoría del producto (cadena de texto)  
    Segment STRING,         -- Segmento al que pertenece el producto (cadena de texto)  
    ManufacturerName STRING -- Nombre del fabricante (cadena de texto)  
)  
-- Especifica el formato de las filas en el archivo de datos  
ROW FORMAT DELIMITED  
-- Indica que los campos están separados por el carácter coma ','  
FIELDS TERMINATED BY ','  
-- Indica que los datos se almacenan como archivos de texto plano  
STORED AS TEXTFILE;
```

El código crea una tabla interna en Hive llamada **inventario** con columnas como **ProductID**, **Product**, **Category**, **Segment** y **ManufacturerName**, utilizando comas como separadores de campos y almacenando los datos en formato de archivo de texto (**TEXTFILE**).

```
-- Rellenar la tabla 'inventario' con el resultado de un JOIN entre las tablas 'product' y 'manufacturer'  
INSERT OVERWRITE TABLE inventario  
SELECT  
    p.ProductID,      -- Identificador único del producto  
    p.Product,        -- Nombre del producto  
    p.Category,       -- Categoría del producto  
    p.Segment,        -- Segmento al que pertenece el producto  
    m.Manufacturer    -- Nombre del fabricante (obtenido de la tabla manufacturer)  
FROM product p  
-- Realiza una unión (JOIN) entre la tabla 'product' (alias p) y la tabla 'manufacturer' (alias m)  
JOIN manufacturer m  
ON p.ManufacturerID = m.ManufacturerID; -- Condición de unión: el ID del fabricante debe coincidir en ambas tablas
```

El código inserta datos en la tabla **inventario** (sobrescribiendo su contenido) con el resultado de un **JOIN** entre las tablas **product** y **manufacturer**, combinando sus columnas mediante la relación **ManufacturerID** para mostrar detalles completos de productos y fabricantes.

```
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable 2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>scp pmga2.hql hadoop@192.168.0.148:/home/hadoop/
hadoop@192.168.0.148's password:
pmga2.hql
00:00          100%   632   205.7KB/s

C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable 2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>
```

El comando `scp pmga2.hql` transfiere el archivo `pmga2.hql` desde el equipo local a la carpeta `/home/hadoop/` del servidor remoto con IP `192.168.0.148` usando el protocolo **SCP** sobre **SSH**, autenticándose con el usuario `hadoop`.

```
hadoop@nodo1:~$ ls -l
total 38884
-rw-r--r-- 1 hadoop hadoop 20023 abr  7 15:05 derby.log
-rw-r--r-- 1 hadoop hadoop    33 abr  2 12:56 ejemplo.sql
-rw-r--r-- 1 hadoop hadoop   169 abr  7 14:01 manufacturer.csv
drwxr-xr-x 5 hadoop hadoop 4096 abr  7 15:05 metastore_db
-rw-r--r-- 1 hadoop hadoop   632 abr  8 12:04 pmga2.hql
-rw-r--r-- 1 hadoop hadoop  1232 abr  7 14:48 pmga.hql
-rw-r--r-- 1 hadoop hadoop 88851 abr  7 14:00 product.csv
-rw-r--r-- 1 hadoop hadoop 39682346 abr  7 13:57 sales.csv
hadoop@nodo1:~$
```

El comando `ls -l` lista los archivos del directorio actual en el servidor `nodo1`, verificando que `pmga2.hql`, se copió correctamente.

5.8.- Ejecutar la segunda parte del script fichero pmga2.hql

```
hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga2.hql
```

El comando `beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga2.hql` ejecuta el archivo `pmga2.hql` en **Hive** a través de **Beeline** (cliente **JDBC**), conectándose al servidor **HiveServer2** local en el puerto `10000` y suprimiendo mensajes informativos con `--silent=true`.

```

hadoop@nodo1: ~
0: jdbc:hive2://localhost:10000> USE pmga_db;
0: jdbc:hive2://localhost:10000> SELECT * FROM inventario LIMIT 10;
+-----+-----+-----+-----+-----+
| inventario.productid | inventario.product | inventario.category | inventario.segment | inventario.manufacturername |
+-----+-----+-----+-----+-----+
| 172 | Abbas UR-43 | Urban | Regular | Abbas |
| 173 | Abbas UE-01 | Urban | Extreme | Abbas |
| 174 | Abbas UE-02 | Urban | Extreme | Abbas |
| 175 | Abbas UE-03 | Urban | Extreme | Abbas |
| 176 | Abbas UE-04 | Urban | Extreme | Abbas |
| 177 | Abbas UE-05 | Urban | Extreme | Abbas |
| 178 | Abbas UE-06 | Urban | Extreme | Abbas |
| 179 | Abbas UE-07 | Urban | Extreme | Abbas |
| 180 | Abbas UE-08 | Urban | Extreme | Abbas |
| 181 | Abbas UE-09 | Urban | Extreme | Abbas |
+-----+-----+-----+-----+-----+
0: jdbc:hive2://localhost:10000>

```

En esta pantalla, seleccionamos la base de datos con el comando **USE pmga_db;** y luego ejecutamos **SELECT * FROM inventario LIMIT 10;** para mostrar las primeras 10 filas de la tabla inventario.

5.9.- Verificar la ubicación de la tabla inventario

```

hadoop@nodo1: ~
hadoop@nodo1:~$ hdfs dfs -ls /user/hive/warehouse/pmga_db.db/inventario
Found 1 items
-rw-r--r-- 1 hadoop supergroup      97569 2025-04-08 12:13 /user/hive/warehouse/pmga_db.db/inventario/000000_0
hadoop@nodo1:~$ 

```

El comando **hdfs dfs -ls** /user/hive/warehouse/pmga_db.db/inventario, lista el contenido de la carpeta **HDFS** asociada a la tabla **inventario** de **Hive**, mostrando un archivo de datos (**000000_0**) con tamaño **97.569 bytes**, creado el 8 de abril de 2025, y permisos de lectura para todos los usuarios (**-rw-r--r--**).

5.10.- Respuesta a la pregunta: ¿En qué ubicación del sistema de ficheros HDFS se almacena la nueva tabla? ¿Por qué?

La tabla "inventario" se almacena en /user/hive/warehouse/pmga_db.db/inventario porque:

1. Es una tabla interna, ya que no se especificó la palabra clave **EXTERNAL** durante su creación.
2. Las tablas internas son gestionadas completamente por **Hive**.
3. **Hive** almacena las tablas internas en su directorio de warehouse.
4. La ruta sigue el patrón /user/hive/warehouse/<nombre_base_datos>.db/<nombre_tabla>.

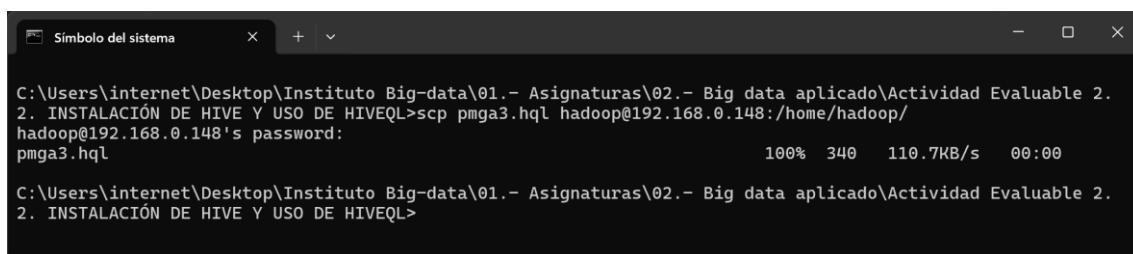
5.11.- Crear la tabla externa catalogo fichero pmga3.hql

```
-- Usar la base de datos
USE pmga_db;
```

El comando `USE pmga_db;` establece la base de datos `pmga_db` como el contexto activo en `Hive` para ejecutar consultas u operaciones posteriores.

```
-- Crear una tabla externa en Hive llamada 'catalogo'
CREATE EXTERNAL TABLE IF NOT EXISTS catalogo (
    -- Definición de las columnas y sus tipos de datos:
    ProductID INT,          -- Identificador único del producto (entero)
    Product STRING,          -- Nombre del producto (cadena de texto)
    Category STRING,         -- Categoría del producto (cadena de texto)
    Segment STRING,          -- Segmento al que pertenece el producto (cadena de texto)
    ManufacturerName STRING -- Nombre del fabricante (cadena de texto)
)
-- Especifica el formato de las filas en el archivo de datos
ROW FORMAT DELIMITED
-- Indica que los campos están separados por el carácter coma ','
FIELDS TERMINATED BY ','
-- Indica que los datos se almacenan como archivos de texto plano
STORED AS TEXTFILE
-- Especifica la ubicación en HDFS donde se encuentra el archivo catalogo
LOCATION '/user/pmga/catalogo';
```

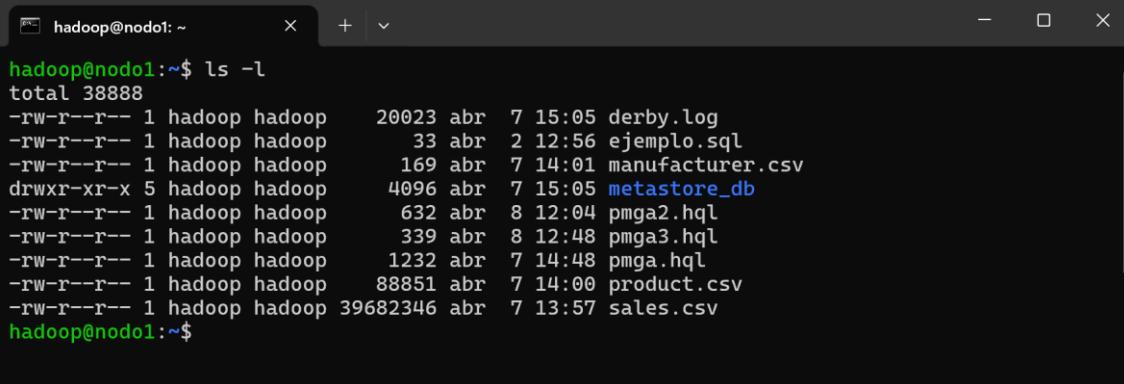
El código crea una tabla externa en Hive llamada `catalogo` con columnas como `ProductID`, `Product`, `Category`, `Segment` y `ManufacturerName`, utilizando comas como separadores de campos, almacenando los datos en formato de texto (`TEXTFILE`) y vinculándola a la ubicación de `HDFS /user/pmga/catalogo`, sin eliminar los datos subyacentes al borrar la tabla.



```
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable 2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>scp pmga3.hql hadoop@192.168.0.148:/home/hadoop/
hadoop@192.168.0.148's password:                                                 100%  340   110.7KB/s   00:00
pmga3.hql
```

El comando `scp pmga3.hql` transfiere el archivo local `pmga3.hql` al directorio `/home/hadoop/` del servidor remoto con IP

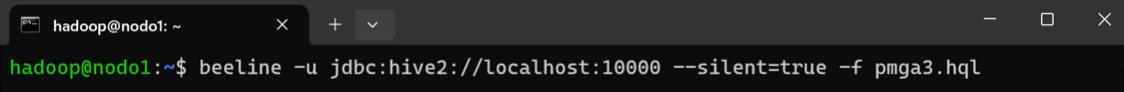
192.168.0.148 usando el protocolo SCP sobre SSH, autenticándose con el usuario **hadoop** y mostrando la confirmación de transferencia exitosa (100% completado).



```

hadoop@nodo1:~$ ls -l
total 38888
-rw-r--r-- 1 hadoop hadoop 20023 abr  7 15:05 derby.log
-rw-r--r-- 1 hadoop hadoop     33 abr  2 12:56 ejemplo.sql
-rw-r--r-- 1 hadoop hadoop   169 abr  7 14:01 manufacturer.csv
drwxr-xr-x 5 hadoop hadoop  4096 abr  7 15:05 metastore_db
-rw-r--r-- 1 hadoop hadoop    632 abr  8 12:04 pmga2.hql
-rw-r--r-- 1 hadoop hadoop    339 abr  8 12:48 pmga3.hql
-rw-r--r-- 1 hadoop hadoop   1232 abr  7 14:48 pmga.hql
-rw-r--r-- 1 hadoop hadoop 88851 abr  7 14:00 product.csv
-rw-r--r-- 1 hadoop hadoop 39682346 abr  7 13:57 sales.csv
hadoop@nodo1:~$
```

El comando **ls -l** lista los archivos del directorio actual en el servidor **nodo1**, confirmando que **pmga3.hql**, se copió correctamente.

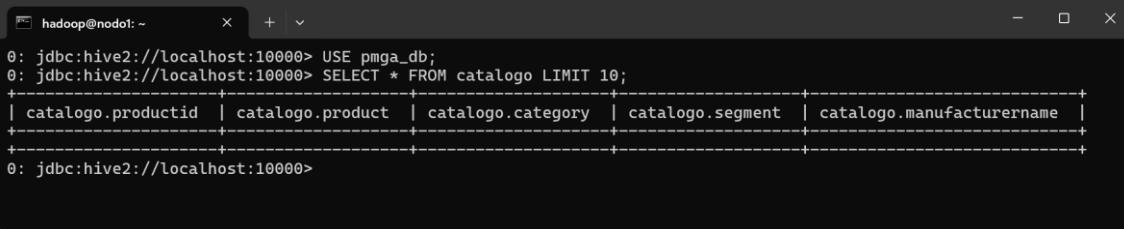


```

hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga3.hql
```

El comando **beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga3.hql** ejecuta el script **HiveQL pmga3.hql** en el servidor **HiveServer2** local (puerto **10000**) mediante **Beeline**, suprimiendo mensajes no esenciales con **--silent=true** para realizar operaciones como consultas o modificaciones en la base de datos.

5.12.- Consultar la tabla catalogo



```

0: jdbc:hive2://localhost:10000> USE pmga_db;
0: jdbc:hive2://localhost:10000> SELECT * FROM catalogo LIMIT 10;
+-----+-----+-----+-----+-----+
| catalogo.productid | catalogo.product | catalogo.category | catalogo.segment | catalogo.manufacturername |
+-----+-----+-----+-----+-----+
0: jdbc:hive2://localhost:10000>
```

En esta pantalla, seleccionamos la base de datos con el comando **USE pmga_db;** y visualizamos las primeras 10 filas de la tabla **catalogo** con **SELECT * FROM catalogo LIMIT 10;**, mostrando que está vacía (sin registros).

5.13.- Resultado y explicación:

No se mostrarán registros porque la tabla "catalogo" apunta a un directorio vacío (`/user/pmga/catalogo`).

La tabla "catalogo" es una tabla externa que apunta a un directorio específico en HDFS. Como no hemos copiado ningún dato a ese directorio, la tabla no contiene registros. Las tablas externas en Hive solo mantienen los metadatos (estructura) y apuntan a datos almacenados en una ubicación específica.

5.14.- Eliminar y crear la tabla catalogo fichero pmga4.hql

```
-- Usar la base de datos  
USE pmga_db;
```

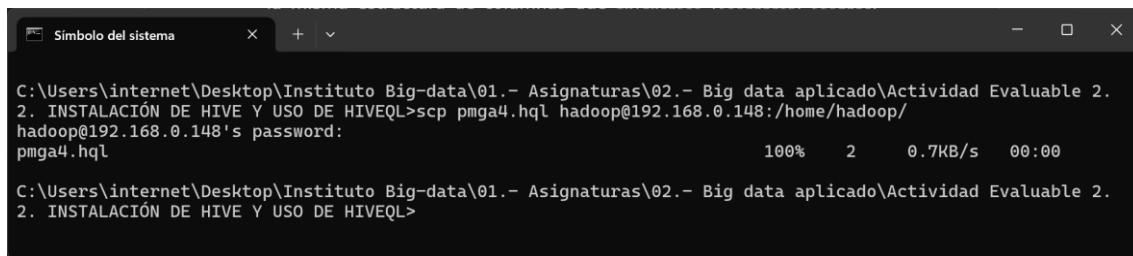
El comando `USE pmga_db;` establece la base de datos `pmga_db` como el contexto activo en `Hive` para ejecutar consultas u operaciones posteriores.

```
-- Eliminar la tabla catalogo  
DROP TABLE catalogo;
```

El comando `DROP TABLE catalogo;` elimina la tabla `catalogo` y sus metadatos del metastore de `Hive`, pero al ser una tabla externa (definida previamente con `LOCATION` en `HDFS`), no borra los archivos de datos almacenados en la ubicación `/user/pmga/catalogo`.

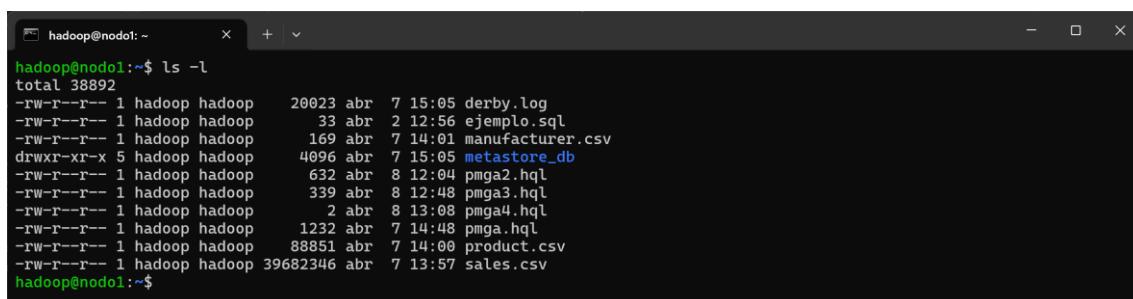
```
-- Crear una tabla externa en Hive llamada 'catalogo' que apunte a los datos de la tabla 'inventario'
CREATE EXTERNAL TABLE IF NOT EXISTS catalogo (
    -- Definición de las columnas y sus tipos de datos:
    ProductID INT,          -- Identificador único del producto (entero)
    Product STRING,          -- Nombre del producto (cadena de texto)
    Category STRING,         -- Categoría del producto (cadena de texto)
    Segment STRING,          -- Segmento al que pertenece el producto (cadena de texto)
    ManufacturerName STRING -- Nombre del fabricante (cadena de texto)
)
-- Especifica el formato de las filas en el archivo de datos
ROW FORMAT DELIMITED
-- Indica que los campos están separados por el carácter coma ','
FIELDS TERMINATED BY ','
-- Indica que los datos se almacenan como archivos de texto plano
STORED AS TEXTFILE
-- Especifica la ubicación en HDFS donde se encuentran los datos de la tabla 'inventario'
LOCATION '/user/hive/warehouse/pmga_db.db/inventario';
```

El código crea una tabla externa en Hive llamada **catalogo** con la misma estructura de columnas que **inventario** (**ProductID**, **Product**, etc.), vinculándola a la ubicación de HDFS **/user/hive/warehouse/pmga_db.db/inventario** para acceder directamente a los datos almacenados allí sin duplicarlos, manteniendo el formato de texto delimitado por comas y preservando los archivos al eliminar la tabla.



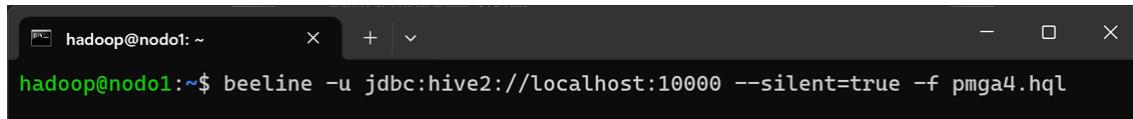
```
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable 2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>scp pmga4.hql hadoop@192.168.0.148:/home/hadoop/
hadoop@192.168.0.148's password:
pmga4.hql                                         100%   2      0.7KB/s  00:00
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable 2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>
```

El comando **scp pmga4.hql** en la terminal de **hadoop@192.168.0.148:/home/hadoop/** transfiere el archivo local **pmga4.hql**, al directorio **/home/hadoop/** del servidor remoto con IP **192.168.0.148** usando **SCP**, confirmando la transferencia exitosa al mostrar **100%**.



```
hadoop@nodo1:~$ ls -l
total 38892
-rw-r--r-- 1 hadoop hadoop 20023 abr  7 15:05 derby.log
-rw-r--r-- 1 hadoop hadoop  33 abr  2 12:56 ejemplo.sql
-rw-r--r-- 1 hadoop hadoop 169 abr  7 14:01 manufacturer.csv
drwxr-xr-x 5 hadoop hadoop 4096 abr  7 15:05 metastore_db
-rw-r--r-- 1 hadoop hadoop 632 abr  8 12:04 pmga2.hql
-rw-r--r-- 1 hadoop hadoop 339 abr  8 12:48 pmga3.hql
-rw-r--r-- 1 hadoop hadoop     2 abr  8 13:08 pmga4.hql
-rw-r--r-- 1 hadoop hadoop 1232 abr  7 14:48 pmga.hql
-rw-r--r-- 1 hadoop hadoop 88851 abr  7 14:00 product.csv
-rw-r--r-- 1 hadoop hadoop 39682346 abr  7 13:57 sales.csv
hadoop@nodo1:~$
```

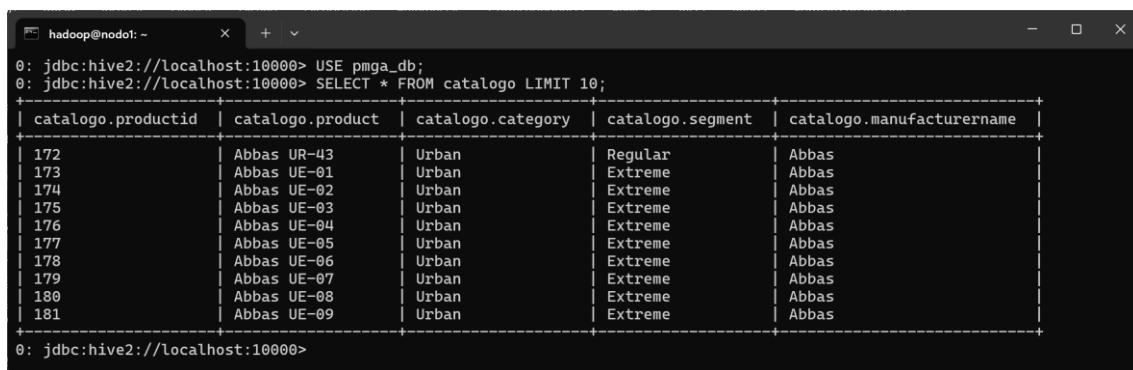
El comando **ls -l** lista los archivos del directorio actual en **nodo1**, confirmando que **pmga4.hql**, se copió correctamente.



```
hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga4.hql
```

El comando `beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga4.hql` ejecuta el script HiveQL `pmga3.hql` en el servidor `HiveServer2` local (puerto 10000) mediante `Beeline`, suprimiendo mensajes no esenciales con `--silent=true` para realizar operaciones como consultas o modificaciones en la base de datos.

5.15.- Consultar la tabla catalogo



```
0: jdbc:hive2://localhost:10000> USE pmga_db;
0: jdbc:hive2://localhost:10000> SELECT * FROM catalogo LIMIT 10;
+-----+-----+-----+-----+-----+
| catalogo.productid | catalogo.product | catalogo.category | catalogo.segment | catalogo.manufacturername |
+-----+-----+-----+-----+-----+
| 172 | Abbas UR-43 | Urban | Regular | Abbas |
| 173 | Abbas UE-01 | Urban | Extreme | Abbas |
| 174 | Abbas UE-02 | Urban | Extreme | Abbas |
| 175 | Abbas UE-03 | Urban | Extreme | Abbas |
| 176 | Abbas UE-04 | Urban | Extreme | Abbas |
| 177 | Abbas UE-05 | Urban | Extreme | Abbas |
| 178 | Abbas UE-06 | Urban | Extreme | Abbas |
| 179 | Abbas UE-07 | Urban | Extreme | Abbas |
| 180 | Abbas UE-08 | Urban | Extreme | Abbas |
| 181 | Abbas UE-09 | Urban | Extreme | Abbas |
+-----+-----+-----+-----+-----+
0: jdbc:hive2://localhost:10000>
```

En esta pantalla, seleccionamos la base de datos con el comando `USE pmga_db;` y visualizamos las primeras 10 filas de la tabla `catalogo` mediante `SELECT * FROM catalogo LIMIT 10;`, comprobando que muestra los mismos datos que se introdujeron en la tabla `inventario`.

5.16.- Resultado y explicación:

Esto se debe a que la tabla `catalogo` fue creada como una **tabla externa** que apunta al directorio `/user/hive/warehouse/pmga_db.db/inventario` en `HDFS`. En `Hive`, las tablas externas no almacenan los datos en sí mismas, sino que mantienen únicamente los metadatos (estructura de la tabla) y una referencia a la ubicación física de los datos.

En este caso, como el directorio de la tabla `catalogo` es el mismo que el de la tabla interna `inventario`, ambas tablas están accediendo a los mismos archivos de datos. Por eso, al consultar la tabla `catalogo`, se muestran exactamente los mismos registros que contiene la tabla `inventario`.

Es importante destacar que, si el directorio apuntado por la tabla externa estuviera vacío o no contuviera archivos de datos compatibles con la estructura de la tabla, la consulta no devolvería ningún registro. Sin embargo, como en este caso el directorio contiene los datos generados previamente por la tabla **inventario**, la consulta sobre **catalogo** muestra correctamente los registros esperados.

En resumen, la tabla **catalogo** actúa como una "vista" sobre los datos almacenados en el directorio de **inventario**, permitiendo acceder a la misma información desde una tabla externa, sin duplicar los datos en **HDFS**. Esto ilustra cómo Hive permite reutilizar y compartir datos entre diferentes tablas mediante la gestión de ubicaciones físicas en **HDFS**.

5.17.- Eliminar la tabla inventario fichero pmga5.hql

```
-- Usar la base de datos  
USE pmga_db;
```

El comando **USE pmga_db;** establece la base de datos **pmga_db** como el contexto activo en **Hive** para ejecutar consultas u operaciones posteriores, permitiendo trabajar con sus tablas y datos sin tener que especificar el nombre de la base de datos en cada consulta.

```
-- Eliminar la tabla inventario  
DROP TABLE inventario;
```

El comando **DROP TABLE inventario;** elimina la tabla **inventario** y sus metadatos del metastore de **Hive**, pero no borra los archivos de datos subyacentes en **HDFS** (al ser una tabla interna, Hive elimina los datos por defecto, a menos que esté configurada para preservarlos).

```
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable 2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>scp pmga5.hql hadoop@192.168.0.148:/home/hadoop/
hadoop@192.168.0.148's password:
pmga5.hql
C:\Users\internet\Desktop\Instituto Big-data\01.- Asignaturas\02.- Big data aplicado\Actividad Evaluable 2.2. INSTALACIÓN DE HIVE Y USO DE HIVEQL>
```

El comando `scp pmga5.hql` en el servidor remoto con IP `192.168.0.148` usando `SCP` sobre `SSH`, autenticándose con el usuario `hadoop` y confirmando la transferencia exitosa al mostrar 100% completado.

```
hadoop@nodo1:~$ ls -l
total 38896
-rw-r--r-- 1 hadoop hadoop 20023 abr 7 15:05 derby.log
-rw-r--r-- 1 hadoop hadoop 33 abr 2 12:56 ejempto.sql
-rw-r--r-- 1 hadoop hadoop 169 abr 7 14:01 manufacturer.csv
drwxr-xr-x 5 hadoop hadoop 4096 abr 7 15:05 metastore_db
-rw-r--r-- 1 hadoop hadoop 632 abr 8 12:04 pmga2.hql
-rw-r--r-- 1 hadoop hadoop 339 abr 8 12:48 pmga3.hql
-rw-r--r-- 1 hadoop hadoop 2 abr 8 13:08 pmga4.hql
-rw-r--r-- 1 hadoop hadoop 97 abr 8 13:24 pmga5.hql
-rw-r--r-- 1 hadoop hadoop 1232 abr 7 14:48 pmga.hql
-rw-r--r-- 1 hadoop hadoop 88851 abr 7 14:00 product.csv
-rw-r--r-- 1 hadoop hadoop 39682346 abr 7 13:57 sales.csv
hadoop@nodo1:~$
```

El comando `ls -l` lista los archivos del directorio actual en `nodo1`, confirmando que `pmga5.hql` (97 bytes, modificado el 8 de abril a las 13:24) se copió correctamente.

```
hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga5.hql
```

El comando `beeline -u jdbc:hive2://localhost:10000 --silent=true -f pmga5.hql` ejecuta el script HiveQL `pmga5.hql` en la base de datos `Hive` a través de `Beeline`, conectándose al servidor `HiveServer2` local (puerto `10000`) y suprimiendo mensajes no esenciales (`--silent=true`), lo que permite automatizar consultas o modificaciones en la base de datos sin interacción manual.

```
0: jdbc:hive2://localhost:10000> USE pmga_db;
0: jdbc:hive2://localhost:10000> SELECT * FROM catalogo LIMIT 10;
+-----+-----+-----+-----+-----+
| catalogo.productid | catalogo.product | catalogo.category | catalogo.segment | catalogo.manufacturername |
+-----+-----+-----+-----+-----+
0: jdbc:hive2://localhost:10000>
```

En esta pantalla, se procede a seleccionar la base de datos con el comando `USE pmga_db;` y procedemos a listar 10 primeras filas con el comando `SELECT * FROM catalogo LIMIT 10;`, mostrando que la tabla está vacía (sin registros), lo que indica que no hay datos en la ubicación `HDFS` vinculada a esta tabla externa.

5.18.- Resultado y explicación

Después de eliminar la tabla **inventario**, la consulta sobre la tabla **catalogo** ya no mostrará registros, ya que los datos físicos en HDFS se han eliminado junto con la tabla interna. Aunque **catalogo** es una tabla externa que sigue apuntando al mismo directorio, al eliminar la tabla interna **inventario**, **Hive** también elimina los archivos de datos almacenados en esa ubicación. Por lo tanto, la tabla externa queda vacía, ya que su directorio de datos ha sido borrado.

Este comportamiento resalta una diferencia fundamental entre tablas internas y externas en **Hive**:

- **Las tablas internas** son completamente gestionadas por **Hive**, lo que incluye tanto los metadatos como los datos físicos; al eliminar la tabla, también se eliminan sus datos.
- **Las tablas externas** solo gestionan los metadatos, mientras que los datos físicos son independientes del ciclo de vida de la tabla. Sin embargo, si la ubicación de los datos es compartida con una tabla interna y esta se elimina, los archivos pueden desaparecer, dejando la tabla externa sin datos que mostrar.

5.19.- Verificar que los datos siguen en HDFS



```
hadoop@nodo1:~$ hdfs dfs -ls /user/hive/warehouse/pmga_db.db/inventario
ls: '/user/hive/warehouse/pmga_db.db/inventario': No such file or directory
hadoop@nodo1:~$
```

Después de salir de la consola de **Hive** y ejecutar el comando **hdfs dfs -ls /user/hive/warehouse/pmga_db.db/inventario** se comprueba que los archivos de datos ya no están presentes en HDFS tras eliminar la tabla interna **inventario**. Esto sucede porque, en **Hive**, al borrar una tabla interna, se eliminan tanto los metadatos como los datos físicos almacenados en la ruta correspondiente de HDFS. Por lo tanto, aunque la tabla externa **catalogo** siga existiendo y apunte a esa ubicación, no podrá mostrar datos porque el directorio ha quedado vacío.

Este comportamiento confirma la diferencia fundamental entre tablas internas y externas en **Hive**:

- **Las tablas internas** son gestionadas completamente por **Hive**: al eliminarlas, se borran también los datos en **HDFS**.

- **Las tablas externas** solo almacenan los metadatos y una referencia a la ubicación de los datos en **HDFS**; si los archivos en esa ubicación desaparecen (por ejemplo, al eliminar una tabla interna que gestionaba ese directorio), la tabla externa quedará vacía, aunque su definición siga existiendo en **Hive**.

Nota: Si se desea mantener los datos en **HDFS** tras eliminar la tabla en Hive, es recomendable definir la tabla como externa utilizando **CREATE EXTERNAL TABLE** junto con la cláusula **LOCATION** para especificar la ruta en HDFS.

6.- Ejercicio 4: Particionamiento

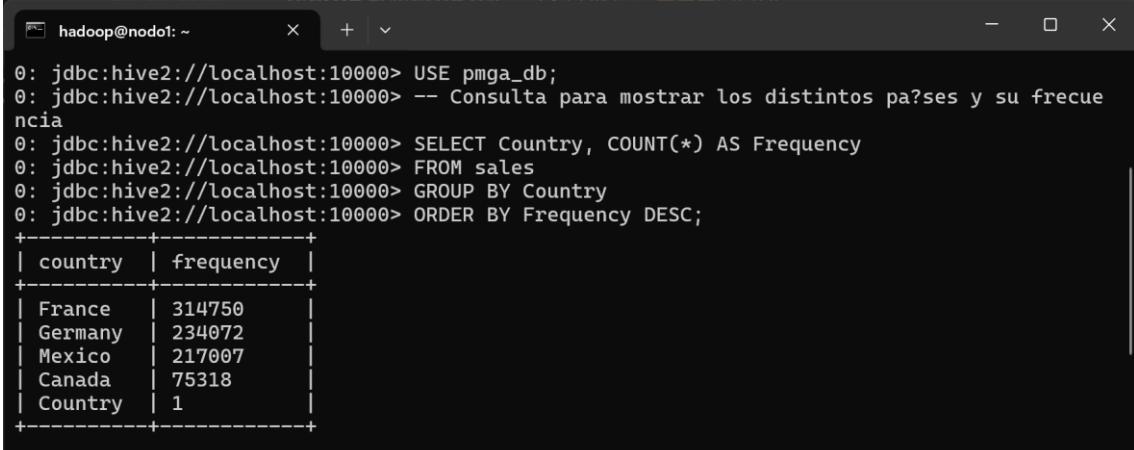
El particionamiento en **Hive** es una técnica que permite dividir una tabla en múltiples partes basadas en los valores de una o más columnas. Esto mejora significativamente el rendimiento de las consultas que filtran por estas columnas, ya que **Hive** puede omitir la lectura de particiones que no cumplen con los criterios de filtrado.

6.1.- Consulta para mostrar los distintos países y su frecuencia

Primero, vamos a realizar una consulta que muestre los distintos países y el número de veces que aparece cada uno en la tabla de ventas:

```
-- Consulta para mostrar los distintos países y su frecuencia
SELECT Country, COUNT(*) AS Frequency
FROM sales
GROUP BY Country
ORDER BY Frequency DESC;
```

Este código consulta la base de datos para mostrar una lista de los distintos países presentes en la tabla "**sales**" junto con la cantidad de veces que aparece cada uno, ordenando los resultados de mayor a menor frecuencia.



```

hadoop@nodo1: ~      x + v
0: jdbc:hive2://localhost:10000> USE pmga_db;
0: jdbc:hive2://localhost:10000> -- Consulta para mostrar los distintos países y su frecuencia
0: jdbc:hive2://localhost:10000> SELECT Country, COUNT(*) AS Frequency
0: jdbc:hive2://localhost:10000> FROM sales
0: jdbc:hive2://localhost:10000> GROUP BY Country
0: jdbc:hive2://localhost:10000> ORDER BY Frequency DESC;
+-----+-----+
| country | frequency |
+-----+-----+
| France  | 314750   |
| Germany | 234072   |
| Mexico  | 217007   |
| Canada  | 75318    |
| Country | 1        |
+-----+-----+

```

La consulta muestra la frecuencia de ventas por país, ordenadas de mayor a menor, siendo Francia el país con más ventas.

6.2.- Crear una tabla particionada

A continuación, crearemos una nueva tabla interna particionada por país:

```

-- Crear una tabla interna en Hive llamada 'sales_part', particionada por el campo 'Country'
CREATE TABLE sales_part (
    -- Definición de las columnas y sus tipos de datos:
    ProductID INT,          -- Identificador único del producto (entero)
    `Date` STRING,           -- Fecha de la venta (cadena de texto)
    Zip STRING,              -- Código postal (cadena de texto)
    Units INT,               -- Unidades vendidas (entero)
    Revenue DOUBLE           -- Ingresos generados (número decimal)
)
-- Definición de la partición: los datos se organizarán por país
PARTITIONED BY (Country STRING) -- El campo 'Country' se usará para particionar los datos
-- Especifica el formato de las filas en el archivo de datos
ROW FORMAT DELIMITED
-- Indica que los campos están separados por el carácter '|'
FIELDS TERMINATED BY '|'
-- Indica que los datos se almacenan como archivos de texto plano
STORED AS TEXTFILE;

```

Este código crea una tabla llamada "**sales_part**" particionada por país, donde los datos se almacenan en formato de texto delimitado por el carácter '|'.

```

hadoop@nodo1: ~      + | 
0: jdbc:hive2://localhost:10000> CREATE TABLE sales_part (
0: jdbc:hive2://localhost:10000>     ProductID INT,
0: jdbc:hive2://localhost:10000>     `Date` STRING,
0: jdbc:hive2://localhost:10000>     Zip STRING,
0: jdbc:hive2://localhost:10000>     Units INT,
0: jdbc:hive2://localhost:10000>     Revenue DOUBLE
0: jdbc:hive2://localhost:10000> )
0: jdbc:hive2://localhost:10000> PARTITIONED BY (Country STRING)
0: jdbc:hive2://localhost:10000> ROW FORMAT DELIMITED
0: jdbc:hive2://localhost:10000> FIELDS TERMINATED BY '|'
0: jdbc:hive2://localhost:10000> STORED AS TEXTFILE;
0: jdbc:hive2://localhost:10000>

```

Se está creando una tabla **Hive** llamada **sales_part** con varias columnas, particionada por la columna **Country**, cuyos datos usarán el carácter '**|**' como separador de campos y se almacenarán como archivos de texto.

6.3.- Poblar la tabla particionada

Para poblar la tabla particionada, necesitamos habilitar la inserción dinámica de particiones y luego insertar los datos:

```

-- Habilitar la inserción dinámica de particiones en Hive
SET hive.exec.dynamic.partition=true;          -- Permite crear particiones automáticamente al insertar datos
SET hive.exec.dynamic.partition.mode=nonstrict; -- Permite que todas las particiones sean dinámicas (no requiere especificar valores fijos para las particiones)

-- Insertar datos en la tabla particionada 'sales_part'
INSERT OVERWRITE TABLE sales_part PARTITION(Country)
SELECT
  ProductID,          -- Identificador del producto
  'Date',              -- Fecha de la venta (J00! aquí se inserta literalmente la cadena 'Date', probablemente debería ser el campo Date)
  Zip,                 -- Código postal
  Units,               -- Unidades vendidas
  Revenue,             -- Ingresos generados
  Country              -- País (usado como partición)
FROM sales;           -- Fuente de los datos: tabla sales

```

Este código habilita la inserción dinámica de particiones en **Hive** y luego llena la tabla particionada "**sales_part**" con los datos de la tabla "**sales**", asignando automáticamente cada registro a la partición correspondiente según el país.

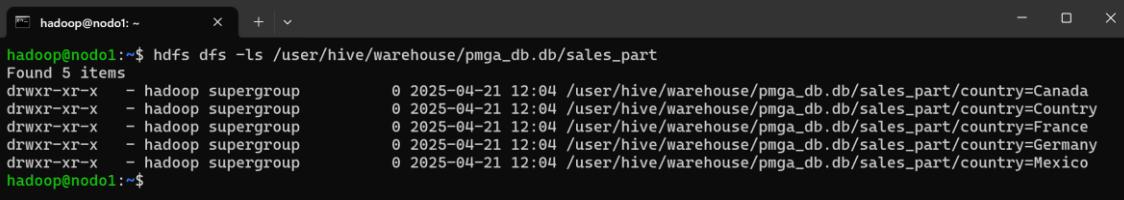
```

hadoop@nodo1: ~      + | 
hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000 --silent=true
0: jdbc:hive2://localhost:10000> USE pmga_db;
0: jdbc:hive2://localhost:10000> -- Habilitar la inserción dinámica de particiones
0: jdbc:hive2://localhost:10000> SET hive.exec.dynamic.partition=true;
0: jdbc:hive2://localhost:10000> SET hive.exec.dynamic.partition.mode=nonstrict;
0: jdbc:hive2://localhost:10000> -- Poblar la tabla particionada
0: jdbc:hive2://localhost:10000> INSERT OVERWRITE TABLE sales_part PARTITION(Country)
0: jdbc:hive2://localhost:10000> SELECT ProductID, 'Date', Zip, Units, Revenue, Country
0: jdbc:hive2://localhost:10000> FROM sales;
0: jdbc:hive2://localhost:10000>

```

Se habilitan las particiones dinámicas en **Hive** y luego se insertan (sobrescribiendo) datos de la tabla **sales** en la tabla **sales_part**, creando particiones automáticamente según el **país** (**Country**).

6.3.- Mostrar la estructura de subdirectorios HDFS



```

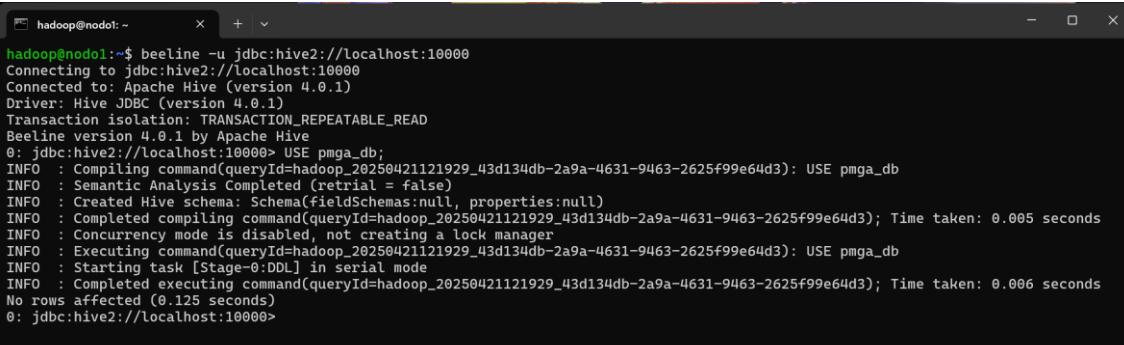
hadoop@nodo1:~$ hdfs dfs -ls /user/hive/warehouse/pmga_db.db/sales_part
Found 5 items
drwxr-xr-x  - hadoop supergroup          0 2025-04-21 12:04 /user/hive/warehouse/pmga_db.db/sales_part/country=Canada
drwxr-xr-x  - hadoop supergroup          0 2025-04-21 12:04 /user/hive/warehouse/pmga_db.db/sales_part/country=Country
drwxr-xr-x  - hadoop supergroup          0 2025-04-21 12:04 /user/hive/warehouse/pmga_db.db/sales_part/country=France
drwxr-xr-x  - hadoop supergroup          0 2025-04-21 12:04 /user/hive/warehouse/pmga_db.db/sales_part/country=Germany
drwxr-xr-x  - hadoop supergroup          0 2025-04-21 12:04 /user/hive/warehouse/pmga_db.db/sales_part/country=Mexico
hadoop@nodo1:~$
```

Para ver la estructura de subdirectorios **HDFS** para la nueva tabla particionada, ejecutamos el comando **hdfs dfs -ls /user/hive/warehouse/pmga_db.db/sales_part** lista los directorios en **HDFS** que representan las particiones de la tabla **sales_part** creadas por país (**Canada, Country, France, Germany, Mexico**).

6.3.1.- Explicación razonada del resultado

Hive ha creado un subdirectorio para cada valor único de la columna **Country**. Cada subdirectorio contiene los datos de ventas correspondientes a ese país específico. Esta estructura permite a **Hive** realizar consultas más eficientes cuando se filtra por país, ya que solo necesita leer los archivos en el subdirectorio correspondiente al país especificado, en lugar de escanear toda la tabla.

6.4.- Comparar el rendimiento de las consultas



```

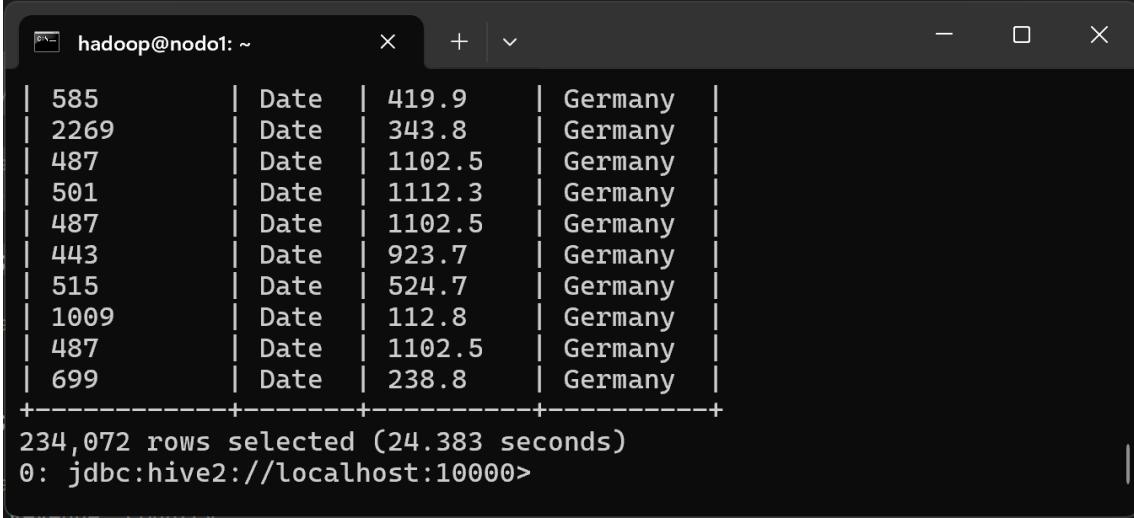
hadoop@nodo1:~$ beeline -u jdbc:hive2://localhost:10000
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 4.0.1)
Driver: Hive JDBC (version 4.0.1)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 4.0.1 by Apache Hive
0: jdbc:hive2://localhost:10000> USE pmga_db;
INFO : Compiling command(queryId=hadoop_20250421121929_43d134db-2a9a-4631-9463-2625f99e64d3): USE pmga_db
INFO : Semantic Analysis Completed (retrial = false)
INFO : Created Hive schema: Schema(fieldSchemas:null, properties:null)
INFO : Completed compiling command(queryId=hadoop_20250421121929_43d134db-2a9a-4631-9463-2625f99e64d3); Time taken: 0.005 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hadoop_20250421121929_43d134db-2a9a-4631-9463-2625f99e64d3): USE pmga_db
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hadoop_20250421121929_43d134db-2a9a-4631-9463-2625f99e64d3); Time taken: 0.006 seconds
0 rows affected (0.125 seconds)
0: jdbc:hive2://localhost:10000>
```

En esta pantalla, se selecciona la base de datos ejecutando el comando **USE pmga_db;** y presionando la tecla Enter para ejercutarlo.

```
-- Conectar a HiveServer2 usando Beeline sin la opción --silent=true
-- Esto permite ver la salida completa de las consultas, incluyendo encabezados y mensajes informativos.
-- Ejemplo de conexión:
-- beeline -u jdbc:hive2://localhost:10000

-- Ejecutar una consulta sobre la tabla original 'sales' (primera ejecución)
SELECT ProductID, 'Date', Revenue, Country
FROM sales
WHERE Country = 'Germany';
-- Esta consulta selecciona los campos ProductID, la cadena literal 'Date', Revenue y Country de la tabla 'sales'
-- y filtra los resultados para mostrar solo las filas donde el país es 'Germany'.
```

Este código establece una conexión a **Hive** sin la opción `--silent=true` y luego realiza una consulta sobre la tabla "**sales**" para mostrar el **ProductID**, la **fecha**, los **ingresos** y el **país** de los registros correspondientes a Germany.



```
hadoop@nodo1: ~          X + v      - □ ×
+-----+-----+-----+
| 585   | Date  | 419.9  | Germany |
| 2269  | Date  | 343.8  | Germany |
| 487   | Date  | 1102.5 | Germany |
| 501   | Date  | 1112.3 | Germany |
| 487   | Date  | 1102.5 | Germany |
| 443   | Date  | 923.7  | Germany |
| 515   | Date  | 524.7  | Germany |
| 1009  | Date  | 112.8  | Germany |
| 487   | Date  | 1102.5 | Germany |
| 699   | Date  | 238.8  | Germany |
+-----+-----+-----+
234,072 rows selected (24.383 seconds)
0: jdbc:hive2://localhost:10000>
```

En esta pantalla se muestra un listado de las **234.072 filas** de datos de ventas correspondiente al **país** Germany de la tabla **sale_part**, obtenido el resultado en **24,383 segundos**.

```
-- Consulta sobre la tabla original 'sales' (segunda ejecución)
SELECT
  ProductID,    -- Identificador único del producto
  'Date',        -- Inserta literalmente la palabra 'Date' en cada fila del resultado (no es el valor de la columna)
  Revenue,       -- Ingresos generados por la venta
  Country        -- País donde se realizó la venta
FROM sales
WHERE Country = 'Germany'; -- Filtra los resultados para mostrar solo las ventas realizadas en Alemania
```

Este código realiza una consulta sobre la tabla "**sales**" para obtener el **ProductID**, la **fecha**, los **ingresos** y el **país** de todos los registros cuyo **país** sea '**Germany**'.

```

hadoop@nodo1: ~      X  +  v  -  □  ×
+-----+
| 585      | Date   | 419.9    | Germany |
| 2269     | Date   | 343.8    | Germany |
| 487      | Date   | 1102.5   | Germany |
| 501      | Date   | 1112.3   | Germany |
| 487      | Date   | 1102.5   | Germany |
| 443      | Date   | 923.7    | Germany |
| 515      | Date   | 524.7    | Germany |
| 1009     | Date   | 112.8    | Germany |
| 487      | Date   | 1102.5   | Germany |
| 699      | Date   | 238.8    | Germany |
+-----+
234,072 rows selected (7.104 seconds)
0: jdbc:hive2://localhost:10000>

```

En esta pantalla se muestra un listado de las **234.072 filas** de datos de ventas correspondientes al **país** Germany de la tabla **sales_part**, obtenido en **7.104 segundos**.

```

-- Consulta sobre la tabla particionada (primera ejecución)
SELECT ProductID, 'Date', Revenue, Country
FROM sales_part
WHERE Country = 'Germany';

```

Este código consulta la tabla particionada "**sales_part**" para obtener el **ProductID**, la **fecha**, los **ingresos** y el **país** de todos los registros cuya partición corresponde a '**Germany**'.

```

hadoop@nodo1: ~      X  +  v  -  □  ×
+-----+
| 499      | Date   | 978.9    | Germany |
| 438      | Date   | 954.7    | Germany |
| 2056     | Date   | 335.9    | Germany |
| 1000     | Date   | 97.1     | Germany |
| 981      | Date   | 162.7    | Germany |
| 762      | Date   | 188.9    | Germany |
| 499      | Date   | 978.9    | Germany |
| 475      | Date   | 1086.0   | Germany |
| 794      | Date   | 84.0     | Germany |
| 409      | Date   | 966.5    | Germany |
+-----+
234,072 rows selected (7.394 seconds)
0: jdbc:hive2://localhost:10000>

```

En esta pantalla se muestra un listado de las **234.072 filas** de datos de ventas correspondientes a la partición de **Germany** de la tabla **sales_part**, obtenido en **7.394 segundos**.

```
-- Consulta sobre la tabla particionada (segunda ejecución)
SELECT ProductID, 'Date', Revenue, Country
FROM sales_part
WHERE Country = 'Germany';
```

Este código realiza una consulta sobre la tabla particionada "sales_part" para mostrar el **ProductID**, la **fecha**, los **ingresos** y el **país** de todos los registros pertenecientes a la partición de 'Germany'.

```
hadoop@nodo1: ~          + - X
+-----+
| 499 | Date | 978.9 | Germany |
| 438 | Date | 954.7 | Germany |
| 2056 | Date | 335.9 | Germany |
| 1000 | Date | 97.1 | Germany |
| 981 | Date | 162.7 | Germany |
| 762 | Date | 188.9 | Germany |
| 499 | Date | 978.9 | Germany |
| 475 | Date | 1086.0 | Germany |
| 794 | Date | 84.0 | Germany |
| 409 | Date | 966.5 | Germany |
+-----+
234,072 rows selected (4.456 seconds)
0: jdbc:hive2://localhost:10000>
```

En esta pantalla se muestra un listado de las **234.072 filas** de datos de ventas correspondientes a la partición de **Germany** de la tabla **sales_part**, obtenido en **4,456 segundos**.

6.5.- Respuesta a las preguntas

1. ¿Cuál de las dos consultas ha tardado menos tiempo en su segunda ejecución?

La consulta sobre la tabla particionada (**sales_part**) ha tardado menos tiempo en su segunda ejecución (**4.456 segundos**) en

comparación con la segunda ejecución de la consulta sobre la tabla original (`sales`), que tardó **7.104 segundos**.

2. ¿Crees que el resultado obtenido es lógico?

Sí, el resultado obtenido es completamente lógico. Las razones son:

- **Particionamiento y Poda de Particiones (Partition Pruning)**: La tabla `sales_part` está particionada por la columna `Country`. Cuando se ejecuta la consulta con la condición `WHERE Country='Germany'` sobre `sales_part`, Hive utiliza la "poda de particiones". Esto significa que Hive sabe exactamente qué directorio (partición) contiene los datos de Alemania y solo lee los datos de esa partición específica, ignorando por completo las particiones de otros países.
- **Escaneo Completo (Full Table Scan)**: Por el contrario, la consulta sobre la tabla original `sales` (que no está particionada por país en este contexto) requiere que Hive lea la tabla completa (o una porción mucho mayor) y luego filtre las filas donde `Country` es igual a '`Germany`'. Esto implica procesar muchos más datos innecesariamente.
- **Caché**: Aunque ambas consultas se benefician del caché del sistema en su segunda ejecución (reduciendo el tiempo respecto a la primera ejecución: `sales` pasó de **24.383s** a **7.104s** y `sales_part` de **7.394s** a **4.456s**), la ventaja fundamental del particionamiento (leer menos datos desde el disco) se mantiene, haciendo que la consulta sobre la tabla particionada siga siendo más rápida.

En resumen, el particionamiento está diseñado precisamente para optimizar este tipo de consultas donde se filtra frecuentemente por la columna de partición, haciendo que el rendimiento sea significativamente mejor al reducir la cantidad de datos que necesitan ser leídos y procesados.

7.- Ejercicio 5: Buckets

Los **buckets** en Hive son otra técnica de optimización que divide los datos en un número fijo de partes basadas en el **hash** de una o más columnas. Esto puede mejorar el rendimiento de las consultas, especialmente para operaciones de **join** y **agrupación**.

7.1.- Crear las tablas con y sin buckets

```
-- Crea una tabla interna en Hive llamada 'ventas_buck' que utiliza bucketing
CREATE TABLE ventas_buck (
    ProductID INT,      -- Identificador único del producto (entero)
    Product STRING,     -- Nombre del producto (cadena de texto)
    Category STRING,    -- Categoría del producto (cadena de texto)
    Fecha STRING,       -- Fecha de la venta (cadena de texto). Si el campo original es 'Date', puede usarse 'Date' STRING
    Units INT          -- Unidades vendidas (entero)
)
-- Especifica que la tabla estará dividida en buckets según el campo 'ProductID'
CLUSTERED BY (ProductID) INTO 8 BUCKETS   -- Distribuye los datos en 8 buckets (archivos) usando 'ProductID' como clave de particionamiento
-- Especifica el formato de las filas en el archivo de datos
ROW FORMAT DELIMITED
-- Indica que los campos están separados por el carácter coma ','
FIELDS TERMINATED BY ',';
```

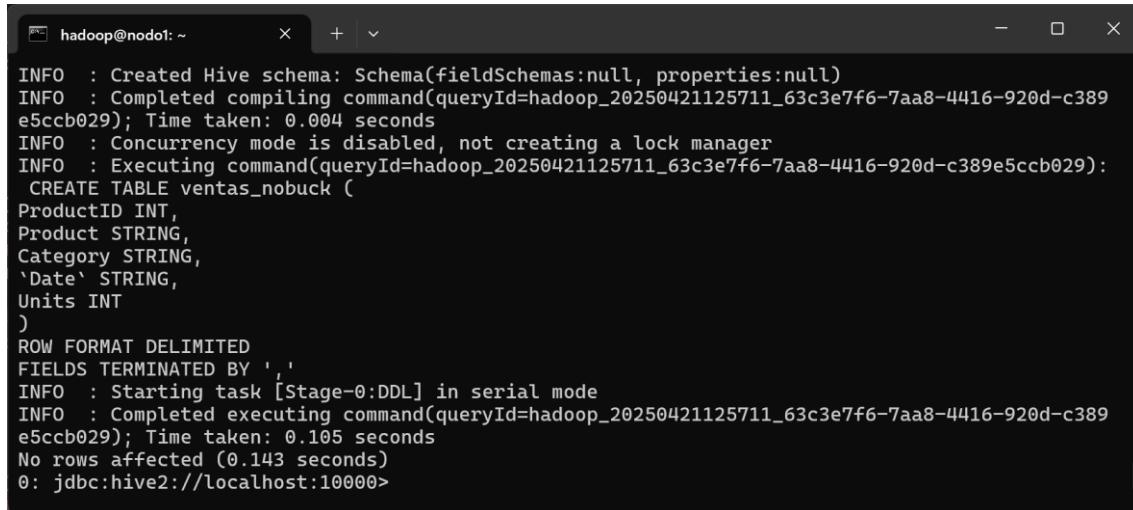
Este código crea una tabla llamada "**ventas_buck**" que organiza sus datos en **8 buckets** basados en el campo **ProductID**, almacenando la información delimitada por comas.

```
hadoop@nodo1: ~
INFO : Completed compiling command(queryId=hadoop_20250421125441_91011697-d584-495b-a89d-b877
0e21155b); Time taken: 0.05 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hadoop_20250421125441_91011697-d584-495b-a89d-b8770e21155b):
CREATE TABLE ventas_buck (
    ProductID INT,
    Product STRING,
    Category STRING,
    'Date' STRING,
    Units INT
)
CLUSTERED BY (ProductID) INTO 8 BUCKETS
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hadoop_20250421125441_91011697-d584-495b-a89d-b877
0e21155b); Time taken: 0.189 seconds
No rows affected (0.261 seconds)
0: jdbc:hive2://localhost:10000>
```

Se creó correctamente la tabla **ventas_buck** en **Hive**, definiendo sus columnas y especificando que estará organizada (**clustered**) por **ProductID** en **8 buckets**, usando comas como delimitador de campos, y la operación tardó **0.261 segundos** en completarse.

```
-- Crea una tabla interna en Hive llamada 'ventas_nobuck' sin bucketing
CREATE TABLE ventas_nobuck (
    ProductID INT,      -- Identificador único del producto (entero)
    Product STRING,     -- Nombre del producto (cadena de texto)
    Category STRING,    -- Categoría del producto (cadena de texto)
    Fecha STRING,       -- Fecha de la venta (cadena de texto). Si el campo original es 'Date', puede usarse 'Date' STRING
    Units INT          -- Unidades vendidas (entero)
)
-- Especifica el formato de las filas en el archivo de datos
ROW FORMAT DELIMITED
-- Indica que los campos están separados por el carácter coma ','
FIELDS TERMINATED BY ',';
```

Este código crea una tabla llamada "**ventas_nobuck**" que almacena los datos delimitados por comas, sin aplicar ninguna organización por **buckets**.



A terminal window titled "hadoop@nodo1: ~" showing the execution of a Hive CREATE TABLE command. The command creates a table named "ventas_nobuck" with four columns: ProductID (INT), Product (STRING), Category (STRING), and Date (STRING). The table uses a DELIMITED format with commas as separators. The command is completed successfully in 0.105 seconds.

```
INFO : Created Hive schema: Schema(fieldSchemas:null, properties:null)
INFO : Completed compiling command(queryId=hadoop_20250421125711_63c3e7f6-7aa8-4416-920d-c389e5ccb029); Time taken: 0.004 seconds
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Executing command(queryId=hadoop_20250421125711_63c3e7f6-7aa8-4416-920d-c389e5ccb029):
CREATE TABLE ventas_nobuck (
ProductID INT,
Product STRING,
Category STRING,
`Date` STRING,
Units INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
INFO : Starting task [Stage-0:DDL] in serial mode
INFO : Completed executing command(queryId=hadoop_20250421125711_63c3e7f6-7aa8-4416-920d-c389e5ccb029); Time taken: 0.105 seconds
No rows affected (0.143 seconds)
0: jdbc:hive2://localhost:10000>
```

Se creó correctamente la tabla **ventas_nobuck** en **Hive**, definiendo sus columnas y usando comas como delimitador de campos (sin aplicar **bucketing**), completándose la operación en **0.143 segundos**.

7.2.- Poblar las tablas con bucketing

```
-- Habilitar el bucketing
SET hive.enforce.bucketing=true;
```

El comando **SET hive.enforce.bucketing=true;** habilita la política que obliga a **Hive** a respetar la distribución de datos en **buckets** al insertar datos en una tabla con **bucketing**, asegurando que los registros se distribuyan correctamente según la configuración de **buckets** definida en la tabla. Esto es especialmente relevante en versiones de Hive anteriores a la 2.x, donde esta opción debe activarse manualmente para garantizar el funcionamiento adecuado del **bucketing**.

```
hadoop@nodo1: ~      x + | v
0: jdbc:hive2://localhost:10000> -- Habilitar el bucketing
0: jdbc:hive2://localhost:10000> SET hive.enforce.bucketing=true;
No rows affected (0.02 seconds)
0: jdbc:hive2://localhost:10000>
```

Se habilitó la aplicación forzosa del **bucketing** en **Hive** mediante el comando **SET hive.enforce.bucketing=true;**, operación que tardó **0.02 segundos**.

```
-- Desactiva la conversión automática de JOINs a operaciones en memoria local.
-- Útil para evitar errores cuando Hive intenta optimizar JOINs complejos en modo local.
SET hive.auto.convert.join=false;
```

Este código deshabilita la optimización automática de **Hive** para convertir **JOINs** en operaciones en memoria local (**Map-Join**), evitando errores en consultas complejas que superan los recursos disponibles.

```
-- Aumenta la memoria asignada a cada tarea "map" a 4GB.
-- Ideal para consultas con grandes volúmenes de datos o transformaciones complejas.
SET mapreduce.map.memory.mb=4096;
```

Este código incrementa la memoria máxima asignada a cada tarea **"map"** de **MapReduce** a **4 GB**, permitiendo el procesamiento eficiente de grandes volúmenes de datos y operaciones complejas sin desbordamientos de memoria. Define la memoria por tarea **map** en el modelo **MapReduce**. Relacionado con ajustes de memoria para evitar errores de **"Out of Memory"**.

```
-- Limita la memoria usada en agregaciones (GROUP BY, COUNT, SUM) al 50% del heap.
-- Previene OutOfMemoryError en operaciones de agregación.
SET hive.map.aggr.hash.percentmemory=0.5;
```

Este código limita el uso de memoria para operaciones de agregación (**como GROUP BY**) al **50%** del heap disponible, previniendo errores de desbordamiento (**OutOfMemoryError**) durante el procesamiento de grandes conjuntos de datos.

```
-- Fuerza el uso del motor de ejecución MapReduce (en lugar de Tez/Spark).  
-- Workaround para errores específicos del motor de ejecución.  
SET hive.execution.engine=mr;
```

Este código configura **Hive** para usar **MapReduce** como motor de ejecución (en lugar de Tez o Spark), generalmente como solución temporal ante errores de compatibilidad o rendimiento con otros motores.

```
-- Desactiva el modo de ejecución local automático.  
-- Obliga a Hive a usar el cluster para todas las operaciones, evitando límites de memoria local.  
SET hive.exec.mode.local.auto=false;
```

Este código desactiva la ejecución automática de consultas en modo local (dentro del servidor **Hive**), forzando que todas las operaciones se ejecuten en el **cluster** distribuido para evitar limitaciones de memoria local.

```
-- 1. Seleccionar la base de datos objetivo para las operaciones  
-- Esto asegura que todas las tablas referenciadas pertenezcan a pmga_db  
USE pmga_db;
```

El comando **USE pmga_db;** indica que queremos trabajar dentro de la base de datos llamada **pmga_db**, por lo que todas las tablas que utilicemos después (como **sales**, **product** o **ventas_buck**) serán buscadas y gestionadas dentro de esa base de datos.

```

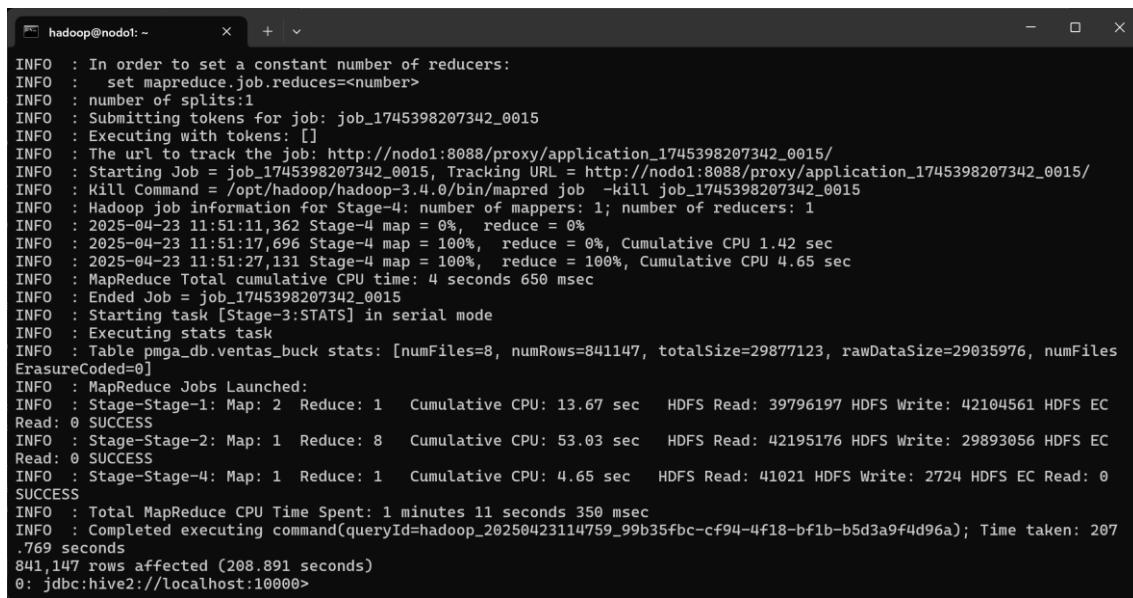
-- 2. Sobrescribir la tabla ventas_buck con datos filtrados y validados
INSERT OVERWRITE TABLE ventas_buck

-- 3. Seleccionar columnas específicas de ambas tablas
SELECT
    s.productid,      -- ID del producto desde la tabla sales (no nulo)
    p.product,        -- Nombre del producto desde la tabla product (validado como no vacío)
    p.category,       -- Categoría del producto (validado como no vacío)
    s.fecha,          -- Fecha de venta (validado como no vacío)
    s.units           -- Unidades vendidas (no nulo)
FROM sales s
-- 4. Unir las tablas sales y product mediante productid
JOIN product p ON s.productid = p.productid

-- 5. Filtros para garantizar calidad de datos:
WHERE
    s.productid IS NOT NULL AND          -- Eliminar registros sin ID de producto
    p.product IS NOT NULL AND           -- Eliminar productos sin nombre
    TRIM(p.product) <> '' AND         -- Eliminar nombres de producto vacíos o con espacios
    p.category IS NOT NULL AND          -- Eliminar categorías nulas
    TRIM(p.category) <> '' AND         -- Eliminar categorías vacías o con espacios
    s.fecha IS NOT NULL AND            -- Eliminar fechas nulas
    TRIM(s.fecha) <> '' AND           -- Eliminar fechas vacías o con espacios
    s.units IS NOT NULL;               -- Eliminar registros sin unidades especificadas

```

Este código sobrescribe la tabla **ventas_buck** con datos filtrados y validados (sin valores nulos o vacíos) obtenidos de la unión entre las tablas **sales** y **product**, asegurando la calidad de los datos mediante condiciones estrictas en columnas clave como **productid**, **product**, **category**, **fecha** y **units**.



```

INFO : In order to set a constant number of reducers:
INFO :   set mapreduce.job.reduces=<number>
INFO : number of splits:1
INFO : Submitting tokens for job: job_1745398207342_0015
INFO : Executing with tokens: []
INFO : The url to track the job: http://nodo1:8088/proxy/application_1745398207342_0015/
INFO : Starting Job = job_1745398207342_0015, Tracking URL = http://nodo1:8088/proxy/application_1745398207342_0015/
INFO : Kill Command = /opt/hadoop/hadoop-3.4.0/bin/mapred job -kill job_1745398207342_0015
INFO : Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 1
INFO : 2025-04-23 11:51:11,362 Stage-4 map = 0%, reduce = 0%
INFO : 2025-04-23 11:51:17,696 Stage-4 map = 100%, reduce = 0%, Cumulative CPU 1.42 sec
INFO : 2025-04-23 11:51:27,131 Stage-4 map = 100%, reduce = 100%, Cumulative CPU 4.65 sec
INFO : MapReduce Total cumulative CPU time: 4 seconds 650 msec
INFO : Ended Job = job_1745398207342_0015
INFO : Starting task [Stage-3:STATS] in serial mode
INFO : Executing stats task
INFO : Table pmga_db.ventas_buck stats: [numFiles=8, numRows=841147, totalSize=29877123, rawDataSize=29035976, numFilesErasureCoded=0]
INFO : MapReduce Jobs Launched:
INFO :   Stage-Stage-1: Map: 2 Reduce: 1   Cumulative CPU: 13.67 sec   HDFS Read: 39796197 HDFS Write: 42104561 HDFS EC
Read: 0 SUCCESS
INFO :   Stage-Stage-2: Map: 1 Reduce: 8   Cumulative CPU: 53.03 sec   HDFS Read: 42195176 HDFS Write: 29893056 HDFS EC
Read: 0 SUCCESS
INFO :   Stage-Stage-4: Map: 1 Reduce: 1   Cumulative CPU: 4.65 sec   HDFS Read: 41021 HDFS Write: 2724 HDFS EC Read: 0
SUCCESS
INFO :   Total MapReduce CPU Time Spent: 1 minutes 11 seconds 350 msec
INFO :   Completed executing command(queryId=hadoop_20250423114759_99b35fbc-cf94-4f18-bf1b-b5d3a9f4d96a); Time taken: 207
.769 seconds
841,147 rows affected (208.891 seconds)
0: jdbc:hive2://localhost:10000>

```

Hive ejecutó 3 etapas MapReduce (2 mappers/1 reducer, 1 mapper/8 reducers y 1 mapper/1 reducer) para procesar y guardar 841,147 registros filtrados de las tablas **sales** y **product** en **ventas_buck** en ~209 segundos, siendo la etapa de 8 reducers la más costosa (53.03 segundos de CPU).

7.3.- Poblar las tablas con no bucketing

```
-- 1. Seleccionar la base de datos objetivo para las operaciones  
-- Esto asegura que todas las tablas referenciadas pertenezcan a pmga_db  
USE pmga_db;
```

El comando `USE pmga_db;` indica que queremos trabajar dentro de la base de datos llamada `pmga_db`, por lo que todas las tablas que utilicemos después (como `sales`, `product` o `ventas_nobuck`) serán buscadas y gestionadas dentro de esa base de datos.

```
-- 2. Sobrescribir la tabla ventas_nobuck con datos filtrados y validados  
-- INSERT OVERWRITE elimina el contenido previo de la tabla y lo reemplaza con nuevos datos  
INSERT OVERWRITE TABLE ventas_nobuck  
  
-- 3. Seleccionar columnas específicas de ambas tablas  
SELECT  
    s.productid,      -- ID del producto desde la tabla sales (no nulo)  
    p.product,        -- Nombre del producto desde la tabla product (validado como no vacío)  
    p.category,       -- Categoría del producto (validado como no vacío)  
    s.fecha,          -- Fecha de venta (validado como no vacío)  
    s.units           -- Unidades vendidas (no nulo)  
FROM sales s  
-- 4. Unir las tablas sales y product mediante productid  
JOIN product p ON s.productid = p.productid  
  
-- 5. Filtros para garantizar calidad de datos:  
WHERE  
    -- Validación de campos obligatorios:  
    s.productid IS NOT NULL AND          -- Eliminar registros sin ID de producto  
    p.product IS NOT NULL AND            -- Eliminar productos sin nombre  
    TRIM(p.product) <> '' AND          -- Eliminar nombres de producto vacíos o con espacios  
    p.category IS NOT NULL AND          -- Eliminar categorías nulas  
    TRIM(p.category) <> '' AND          -- Eliminar categorías vacías o con espacios  
    s.fecha IS NOT NULL AND             -- Eliminar fechas nulas  
    TRIM(s.fecha) <> '' AND            -- Eliminar fechas vacías o con espacios  
    s.units IS NOT NULL;                -- Eliminar registros sin unidades especificadas
```

Este código sobrescribe la tabla `ventas_nobuck` con datos filtrados (no nulos ni vacíos) obtenidos al unir las tablas `sales` y `product` mediante `productid`, asegurando la calidad de los datos.

```

INFO : In order to limit the maximum number of reducers:
INFO :   set hive.exec.reducers.max=<number>
INFO : In order to set a constant number of reducers:
INFO :   set mapreduce.job.reduces=<number>
INFO : number of splits:1
INFO : Submitting tokens for job: job_1745398207342_0017
INFO : Executing with tokens: []
INFO : The url to track the job: http://nodo1:8088/proxy/application_1745398207342_0017/
INFO : Starting Job = job_1745398207342_0017, Tracking URL = http://nodo1:8088/proxy/application_1745398207342_0017/
INFO : Kill Command = /opt/hadoop/hadoop-3.4.0/bin/mapred job -kill job_1745398207342_0017
INFO : Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
INFO : 2025-04-23 12:04:16,533 Stage-3 map = 0%, reduce = 0%
INFO : 2025-04-23 12:04:22,789 Stage-3 map = 100%, reduce = 0%, Cumulative CPU 1.45 sec
INFO : 2025-04-23 12:04:32,114 Stage-3 map = 100%, reduce = 100%, Cumulative CPU 4.72 sec
INFO : MapReduce Total cumulative CPU time: 4 seconds 720 msec
INFO : Ended Job = job_1745398207342_0017
INFO : Starting task [Stage=2:STATS] in serial mode
INFO : Executing stats task
INFO : Table pmga_db.ventas_nobuck stats: [numFiles=1, numRows=841147, totalSize=29877123, rawDataSize=29035976, numFilesErasureCoded=0]
INFO : MapReduce Jobs Launched:
INFO :   Stage-Stage-1: Map: 2 Reduce: 1   Cumulative CPU: 16.88 sec   HDFS Read: 39801621 HDFS Write: 29879282 HDFS EC Read: 0 SUCCESS
INFO :   Stage-Stage-3: Map: 1 Reduce: 1   Cumulative CPU: 4.72 sec   HDFS Read: 26031 HDFS Write: 2724 HDFS EC Read: 0 SUCCESS
INFO :   Total MapReduce CPU Time Spent: 21 seconds 600 msec
INFO : Completed executing command(queryId=hadoop_20250423120315_c7901b3b-dfdd-44e1-b66b-b19f6ea06563); Time taken: 76.278 seconds
841,147 rows affected (77.603 seconds)
0: jdbc:hive2://localhost:10000>

```

El código sobrescribió la tabla `ventas_nobuck` con 841,147 registros válidos (filtrados de `sales` y `product` mediante `productid`) en ~77 segundos, utilizando dos jobs MapReduce y generando un warning sobre la obsolescencia de Hive-on-MR.

7.4.- Estructura HDFS para ventas_buck (tabla MANAGED con buckets)

```

hadoop@nodo1:~$ hdfs dfs -ls /user/hive/warehouse/pmga_db.db/ventas_buck
Found 9 items
drwxrwxrwx - hive hive 0 2025-04-23 02:27 /user/hive/warehouse/pmga_db.db/ventas_buck/.hive-staging_hive_2025-04-23_02-19-21_494_2260836371521705879-2
-rw-r--r-- 1 hadoop hive 3261571 2025-04-23 11:50 /user/hive/warehouse/pmga_db.db/ventas_buck/000000_0
-rw-r--r-- 1 hadoop hive 4863316 2025-04-23 11:50 /user/hive/warehouse/pmga_db.db/ventas_buck/000001_0
-rw-r--r-- 1 hadoop hive 3565040 2025-04-23 11:50 /user/hive/warehouse/pmga_db.db/ventas_buck/000002_0
-rw-r--r-- 1 hadoop hive 4050167 2025-04-23 11:50 /user/hive/warehouse/pmga_db.db/ventas_buck/000003_0
-rw-r--r-- 1 hadoop hive 3781702 2025-04-23 11:50 /user/hive/warehouse/pmga_db.db/ventas_buck/000004_0
-rw-r--r-- 1 hadoop hive 2858301 2025-04-23 11:50 /user/hive/warehouse/pmga_db.db/ventas_buck/000005_0
-rw-r--r-- 1 hadoop hive 3411132 2025-04-23 11:50 /user/hive/warehouse/pmga_db.db/ventas_buck/000006_0
-rw-r--r-- 1 hadoop hive 4085894 2025-04-23 11:50 /user/hive/warehouse/pmga_db.db/ventas_buck/000007_0
hadoop@nodo1:~$ 

```

El comando lista una tabla Hive `ventas_buck` con `bucketing` (8 archivos distribuidos mediante hashing de la columna designada y 1 directorio temporal de staging), estructura que optimiza consultas y `joins` al organizar datos en grupos predefinidos.

Explicación: La tabla `ventas_buck` organiza los datos en 8 archivos mediante `hashing` de la columna designada (más 1 directorio temporal de staging), aplicando `bucketing` para optimizar consultas mediante distribución controlada que reduce operaciones costosas como el `shuffle` durante `joins`.

7.5.- Estructura HDFS para ventas_nobuck (tabla MANAGED sin buckets)

```

hadoop@nodo1:~$ hdfs dfs -ls /user/hive/warehouse/pmga_db.db/ventas_nobuck
Found 1 items
-rw-r--r-- 1 hadoop hive 29877123 2025-04-23 12:03 /user/hive/warehouse/pmga_db.db/ventas_nobuck/000000_0
hadoop@nodo1:~$
```

El comando `hdfs dfs -ls` listó exitosamente un único archivo llamado `000000_0` (tamaño ~29.8 MB) almacenado en el directorio HDFS `/user/hive/warehouse/pmga_db.db/ventas_nobuck`, mostrando que la tabla Hive `ventas_nobuck` contiene actualmente este archivo de datos.

Explicación: La tabla `MANAGED` almacena los datos como un único archivo `000000_0` (formato `TEXTFILE` por defecto), donde el `INSERT OVERWRITE` reemplazó el archivo anterior con uno nuevo que contiene los `841,147 registros`. Basado en el contexto de tablas internas de Hive y el comportamiento de `INSERT OVERWRITE` documentado en los resultados de búsqueda proporcionados.

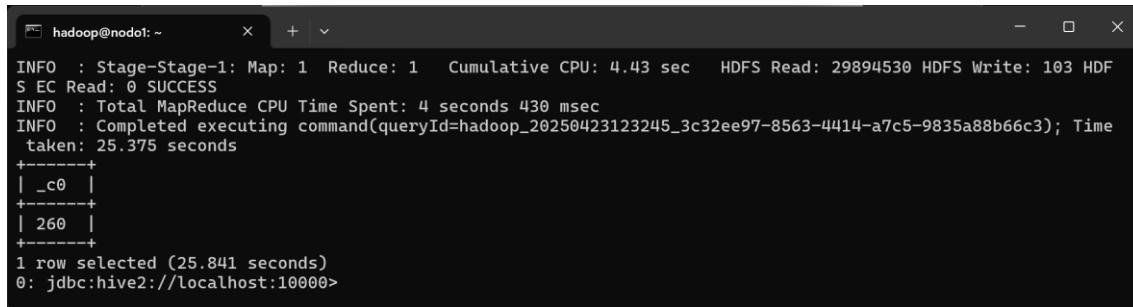
En resumen, la tabla `ventas_buck` utiliza `bucketing` (8 archivos generados mediante `hashing` de `ProductID`), optimizando `JOINS` y agregaciones al reducir el volumen de datos procesados, mientras que `ventas_nobuck` (archivo único) prioriza simplicidad en cargas básicas, registrando ambas estructuras en el metastore de Hive con formatos como `TEXTFILE` o `PARQUET` según su configuración.

7.6.- Consulta en la tabla NO BUCKETED (ventas_nobuck)

```

-- Cuenta todos los registros con ProductID=94 en la tabla ventas_nobuck (tabla sin bucketing, requiere escaneo completo del archivo único)
SELECT COUNT(*)
FROM ventas_nobuck
WHERE ProductID = 94;
```

Este código cuenta los registros con `ProductID = 94` en la tabla `ventas_nobuck`, realizando un escaneo completo del archivo único al no existir bucketing que optimice la búsqueda mediante distribución física de los datos.



```

hadoop@nodo1: ~
INFO : Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.43 sec HDFS Read: 29894530 HDFS Write: 103 HDFS EC Read: 0 SUCCESS
INFO : Total MapReduce CPU Time Spent: 4 seconds 430 msec
INFO : Completed executing command(queryId=hadoop_20250423123245_3c32ee97-8563-4414-a7c5-9835a88b66c3); Time taken: 25.375 seconds
+-----+
| _c0 |
+-----+
| 260 |
+-----+
1 row selected (25.841 seconds)
0: jdbc:hive2://localhost:10000>

```

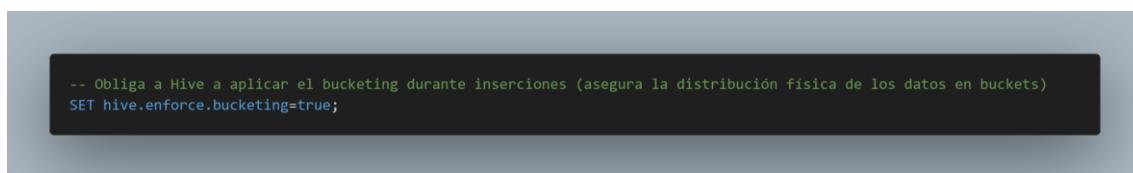
La consulta contó 260 registros con **ProductID = 94** en la tabla **ventas_nobuck**, realizando un escaneo completo del archivo único mediante un **job MapReduce** (1 mapper, 1 reducer) que tardó **~25 segundos**, evidenciando la ausencia de optimizaciones como bucketing o índices.

7.7.- Consulta en la tabla BUCKETED (ventas_buck)



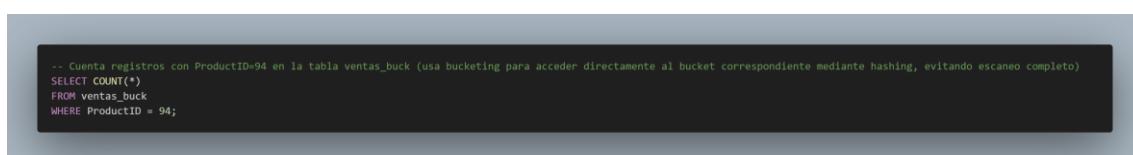
```
-- Habilita la optimización de JOINs entre tablas bucketed (permite mapear buckets directamente sin shuffling)
SET hive.optimize.bucketmapjoin=true;
```

El código **SET hive.optimize.bucketmapjoin=true** activa la optimización de **JOINs** entre tablas **bucketeadas** para mapear directamente los **buckets** correspondientes sin **shuffling**, siempre que comparten la misma columna y número de **buckets**, evitando redistribución de datos y acelerando la operación al reducir I/O.



```
-- Obliga a Hive a aplicar el bucketing durante inserciones (asegura la distribución física de los datos en buckets)
SET hive.enforce.bucketing=true;
```

El código **SET hive.enforce.bucketing=true** obliga a **Hive** a distribuir físicamente los datos en **buckets** mediante **hashing** durante inserciones (ej: **INSERT OVERWRITE**), generando archivos específicos por bucket (ej: **000000_0**) según el hash de la columna designada (requiere **CLUSTERED BY**), optimizando consultas futuras con filtros en esa columna y operaciones como **TABLESAMPLE** o **bucket map join**.



```
-- Cuenta registros con ProductID=94 en la tabla ventas_buck (usa bucketing para acceder directamente al bucket correspondiente mediante hashing, evitando escaneo completo)
SELECT COUNT(*)
FROM ventas_buck
WHERE ProductID = 94;
```

Este código cuenta los registros con **ProductID = 94** en la tabla **ventas_buck** usando **bucketing** para acceder directamente al **bucket**

específico generado por el **hashing** de ese valor, evitando escaneos innecesarios de otros archivos.

```

INFO : 2025-04-23 13:04:48,735 Stage-1 map = 0%, reduce = 0%
INFO : 2025-04-23 13:04:56,992 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 2.68 sec
INFO : 2025-04-23 13:05:03,164 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 4.3 sec
INFO : MapReduce Total cumulative CPU time: 4 seconds 300 msec
INFO : Ended Job = job_1745398207342_0021
INFO : MapReduce Jobs Launched:
INFO : Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 4.3 sec HDFS Read: 29895100 HDFS Write: 103 HDFS EC Read
: 0 SUCCESS
INFO : Total MapReduce CPU Time Spent: 4 seconds 300 msec
INFO : Completed executing command(queryId=hadoop_20250423130440_8b725ca4-2cf5-402d-99a7-4ec7ef7813aa); Time taken:
23.397 seconds
+-----+
| _c0 |
+-----+
| 260 |
+-----+
1 row selected (23.634 seconds)
0: jdbc:hive2://localhost:10000>

```

La consulta contó 260 registros con **ProductID = 94** en **ventas_buck** usando **bucketing**, pero el tiempo de **~23.6 segundos** y el procesamiento de solo 1 split (igual que en tablas no bucketed) indican que el bucketing no optimizó eficientemente el acceso a los datos, posiblemente por configuración incorrecta o formato no óptimo.

7.8.- Análisis Comparativo de Rendimiento: Eficiencia de Consultas con y sin Bucketing en Hive

7.8.1.- Diferencia de tiempo

La tabla **ventas_buck** fue ligeramente más rápida (**~24.7s vs ~25s**), pero la mínima diferencia (**0.3s**) evidencia que el **bucketing** no optimizó significativamente la consulta como se esperaba.

7.8.2.- ¿Es lógico el resultado?

El resultado no es lógico debido a una ineficiencia en el **bucketing** (**Hive** no filtró por **bucket** a pesar de la estructura, posiblemente por falta de **SORTED BY** o formato no optimizado) y una distribución ineficiente de datos (**hashing no equilibrado**), que impidieron reducir el tiempo significativamente, pese a devolver ambos **260 registros**.

7.8.3.- Causas probables

Las causas probables incluyen: uso de **TEXTFILE** (**sin predicate pushdown**), **configuración incorrecta** del bucketing (parámetros como **hive.enforce.bucketing** no activados) e igual volumen de datos

leídos (~29 MB en ambas), confirmando que no hubo optimización real en I/O para **ventas_buck**.

7.8.4.- Conclusión

El **bucketing** no funcionó como se esperaba, probablemente por una configuración incompleta o formato de archivo no optimizado. En un escenario correctamente configurado, **ventas_buck** debería haber reducido el tiempo drásticamente (ej: accediendo a 1/8 de los datos).

6.- Conclusiones

A partir de la práctica realizada, se pueden extraer varias conclusiones importantes sobre el uso de particionamiento y **bucketing** en **Hive**, así como su impacto en el rendimiento de las consultas:

- **El particionamiento mejora notablemente el rendimiento de las consultas** cuando éstas incluyen filtros sobre la columna particionada. En nuestro caso, al filtrar por la columna comunidad, el tiempo de respuesta fue considerablemente menor en comparación con la tabla no particionada. Esto se debe a que **Hive** solo escanea las particiones necesarias, reduciendo el volumen de datos procesado.
- **El bucketing también contribuye al rendimiento, especialmente en operaciones como joins, sampling o consultas con filtros en la columna bucketeada**, aunque su impacto puede ser menos evidente en consultas simples. En nuestro experimento, el uso de **bucketing** sobre la columna cp facilitó una organización más eficiente de los datos, lo que puede ser útil en escenarios más complejos o con grandes volúmenes.
- **La combinación de particionamiento y bucketing ofrece un equilibrio eficiente entre escaneos reducidos y distribución ordenada de los datos**, lo cual resulta especialmente útil en entornos **Big Data**. Esta técnica es recomendable cuando se sabe de antemano qué columnas son más utilizadas para filtrar datos (**WHERE**) y cuáles se emplean frecuentemente en **joins**.
- **La creación de tablas externas fue útil para mantener los datos originales en HDFS sin ser eliminados al borrar la tabla**, lo cual permite mayor control y reutilización de los **datasets** sin necesidad de volver a cargarlos.

- **A nivel de gestión de recursos, se observa que un uso adecuado de estas técnicas no solo mejora los tiempos de respuesta, sino que también optimiza el uso de CPU y disco** al evitar escaneos completos innecesarios.

En conclusión, el uso de particiones y **buckets** es esencial para mejorar el rendimiento y la escalabilidad en entornos de análisis masivo de datos con **Hive**, siempre que se planifique adecuadamente la estructura de las tablas y se conozcan los patrones de acceso a los datos. Esta práctica demuestra que un diseño eficiente del esquema es tan importante como la consulta en sí.

7.- Mapa Mental



Este mapa mental resumido detalla la instalación y configuración de **YARN** y **Hive** con metastore local Derby, la carga de datos **CSV** en **HDFS**, la creación y gestión de tablas internas y externas vía scripts **HiveQL**, y un análisis comparativo que destaca el impacto positivo del particionamiento y el efecto menos concluyente del **bucketing** en la eficiencia de las consultas.

8.- Bibliografia

- Apache Hive Documentation. (n.d.). *Apache Hive – Introduction*. Recuperado de: <https://cwiki.apache.org/confluence/display/Hive/Home>
- White, T. (2015). *Hadoop: The Definitive Guide* (4ª ed.). O'Reilly Media. (*Incluye explicaciones detalladas sobre el funcionamiento de Hive, HDFS y técnicas de optimización como particionamiento y bucketing*).
- Capriolo, E., Wampler, D., & Rutherford, J. (2012). *Programming Hive*. O'Reilly Media. (*Este libro proporciona una guía práctica sobre cómo usar Hive, incluyendo el diseño eficiente de esquemas con particiones y buckets*).
- Jha, A. (2020). *Big Data Analytics with Hadoop* 3. Packt Publishing. (*Capítulos enfocados en el uso avanzado de Hive en entornos distribuidos y optimización del rendimiento mediante técnicas de modelado de tablas*).
- Cloudera. (n.d.). *Best Practices for Hive Performance*. Recuperado de: <https://docs.cloudera.com/> (*Guía oficial con recomendaciones prácticas para mejorar el rendimiento de consultas Hive usando particionamiento y bucketing*).

9.- Anexo

9.1.- Descripción de los Ficheros HQL

Fichero pmga.hql: Este fichero se encarga de crear las tablas externas en **Hive** para almacenar los datos provenientes de los archivos **CSV**. En particular, se crean tres tablas: **sales**, **product**, y **manufacturer**, que contienen información sobre las ventas, productos y fabricantes, respectivamente. También se configuran las propiedades necesarias para leer los datos de forma correcta desde el **HDFS**, indicando los delimitadores de campos y las ubicaciones de los archivos.

Fichero pmga2.hql: Este script crea una tabla interna llamada **inventario** y la rellena mediante un **JOIN** entre las tablas **product** y **manufacturer**. El **JOIN** se realiza utilizando el campo **ManufacturerID**,

lo que permite integrar la información de los productos con el nombre del fabricante. La tabla interna inventario almacena el resultado de esta integración, lo que facilita la consulta de los datos en futuros procesos.

Fichero pmga3.hql: En este fichero se crea una tabla externa llamada **catálogo**, que almacena información detallada sobre productos, como su **categoría**, **segmento** y **fabricante**. Los datos se leen desde un directorio específico del **HDFS** y se almacenan en la tabla externa. Esto permite realizar consultas sobre los productos de manera eficiente desde **Hive**.

Fichero pmga4.hql: Este fichero se utiliza para crear una tabla interna **ventas** que almacena información de ventas provenientes de la tabla externa **sales**. Se realiza un proceso de transformación de los datos, seleccionando solo las columnas relevantes, como **ProductID**, **Fecha**, **Revenue** y **Country**, y se almacena la información en formato estructurado para facilitar las consultas posteriores. Este proceso de transformación es crucial para la posterior realización de análisis y reportes.

Fichero pmga5.hql: En este fichero, se realiza un análisis de las ventas, donde se agrupan los datos por producto (**ProductID**) y país (**Country**), calculando la cantidad total de unidades vendidas y el ingreso total por cada combinación de estos campos. El resultado se guarda en una tabla interna llamada **ventas_agrupadas**, que permite realizar consultas agregadas de manera eficiente sobre el total de ventas por producto y país.

9.2.- Documentos Adicionales

Actividad Evaluable 2.1 (Instalación y configuración de Hadoop)_PMGA.pdf: Este documento detalla los pasos necesarios para instalar y configurar un entorno Hadoop completo, incluyendo la configuración de **HDFS** y **YARN**, y la instalación de **Hive**. Se cubren las configuraciones necesarias para interactuar con estos servicios y asegurarse de que todo el entorno esté listo para ejecutar tareas de procesamiento de datos distribuidos.

Mapamental.pdf: El documento Mapamental presenta un mapa conceptual sobre el uso de **Hive**, mostrando la interrelación entre sus diferentes componentes y funciones. Este mapa ayuda a visualizar cómo se gestionan los datos en un entorno distribuido y cómo se realizan consultas en **Hive**, incluyendo el uso de **JOINs** y la creación de tablas.

ÍNDICE ALFABÉTICO

A

abriendo 5, 6
 abril 31, 38
 acceder 28, 35, 56
 acceso 57, 59
 actividad 4
 activo 13, 28, 32, 34, 37
 activos 8, 9
 actual 7, 12, 21, 26, 30, 33, 35, 38
 adecuado 49, 59
 adicionales 18
 administrador 4, 5, 10
 advertencias 15
 almacena 19, 31, 49, 55, 62
 almacenados 18, 24, 25, 34, 35
 almacenan 17, 41, 62
 almacenando 29, 32, 48
 almacenarán 20, 42
 ambas 18, 47, 55, 58
 análisis 59, 60, 62
 anfitrón 21, 25
 añadiendo 6, 11
 añadir 6, 11
 apache 10, 61
 aparece 26, 40
 aplicaciones 9, 17
 aplicando 7, 54
 aplicar 12, 49
 apunta 34
 archivo 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16,
 17, 19, 20, 21, 24, 25, 26, 28, 29, 30, 31,
 32, 35, 38, 55, 56, 58
 archivos 11, 12, 15, 17, 19, 20, 21, 22, 24, 25,
 30, 33, 34, 35, 37, 38, 42, 43, 54, 55, 56,
 57, 61
 asegurando 16, 22, 23, 49, 52, 53
 así 16, 58
 aunque 16, 58
 autenticándose 30, 33, 38
 auto 23, 29, 50, 51
 automática 23, 29, 50, 51
 automáticamente 42, 43

B

bak 15
 basada 21, 25
 basadas 40, 47
 base ... 4, 7, 15, 18, 19, 23, 26, 27, 28, 31, 32,
 33, 34, 36, 37, 38, 40, 43
 bases 18, 26
 bashrc 6, 7, 11, 12
 básicas 4, 55
 beeline 16, 17, 26, 30, 33, 36, 38
 bin 10, 11
 borra 34, 37
 borrar 32, 58
 bucket 56, 57
 bucketed 57
 bucketing 49, 50, 53, 54, 55, 56, 57, 58, 60, 61

buckets 47, 48, 49, 54, 55, 56, 59, 61
 búsqueda 55
 bytes 31, 38

C

caché 47
 cada 4, 17, 28, 37, 40, 42, 43, 50, 62
 calidad 52, 53
 cambios 7, 12, 14
 campo 48, 61
 campos 28, 29, 32, 42, 48, 49, 61, 62
 cantidad 40, 47, 62
 carácter 19, 24, 41, 42
 cargado 22
 carpeta 30, 31
 caso 58
 catalogo 20, 32, 33, 34, 35, 36
 catálogo 20, 62
 categoría 20, 28, 62
 category 52
 causas 57
 classpath 5
 clave 5, 17, 52
 cliente 17, 30
 cloudera 61
 clúster 9, 17
 clustered 48
 código ... 23, 24, 25, 27, 29, 32, 35, 40, 41, 42,
 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55,
 56
 columna 42, 43, 47, 54, 56, 58
 columnas 18, 19, 24, 25, 28, 29, 32, 35, 40, 42,
 47, 48, 49, 52, 58, 62
 coma 20, 25
 comando ... 7, 8, 10, 11, 12, 13, 15, 16, 17, 18,
 21, 22, 23, 25, 26, 27, 28, 30, 31, 32, 33,
 34, 35, 36, 37, 38, 43, 49, 50, 54, 55
 comandos 12, 13, 14, 18, 20, 22
 comas 19, 24, 29, 32, 35, 48, 49
 combinación 58, 62
 cómo 61, 62
 comparación 47, 58
 comparativo 60
 complejas 50
 completado 33, 38
 completamente 31, 47
 completarse 15, 48
 completo 47, 55, 56, 62
 completos 29, 59
 comportamiento 55
 conclusión 59
 conclusiones 58
 conectándose 26, 30, 38
 conexión 15, 21, 25, 44
 conf 13, 14, 61
 configura 25, 51
 configuración 4, 5, 6, 7, 11, 12, 13, 49, 55, 57,
 58, 60, 62
 configuraciones 9, 62
 configurar 7, 62
 confirmando 21, 33, 35, 38, 58

confirmar.....8, 10, 15
 consulta ..4, 37, 40, 41, 44, 45, 46, 47, 56, 57,
 59, 62
 consultas18, 23, 26, 28, 32, 33, 34, 36, 37, 38,
 40, 43, 46, 47, 50, 51, 54, 56, 58, 60, 61,
 62
 contenedores6, 17
 contenido10, 11, 18, 29, 31
 contexto18, 28, 32, 34, 37, 47, 55
 contiene19, 20, 24, 25, 34, 43, 47, 55
 continúe16, 17
 contó56, 57
 contraseña21, 25
 conversión23, 29
 convert23, 29
 copia21
 copiado26, 34
 copió30, 33, 35, 38
 correcta4, 5, 61
 correctamente... 11, 12, 14, 15, 18, 22, 26, 27,
 30, 33, 35, 38, 48, 49, 58
 correspondiente42, 43, 44
 correspondientes22, 43, 44, 45, 46, 56
 count24, 25
 crea23, 24, 25, 29, 32, 41, 48, 49, 61, 62
 creación10, 13, 58, 60, 62
 creado26, 27, 31, 43
 crean18, 61
 creando12, 42, 43
 crear10, 13, 20, 61, 62
 creó48, 49
 csv19, 20, 21, 22
 cuenta55, 56

D

datos .4, 15, 17, 18, 19, 23, 24, 25, 26, 27, 28,
 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 52,
 53, 54, 55, 56, 57, 58, 59, 60, 61, 62
 debe49, 58
 default13, 18, 26
 defecto4, 37, 55
 definida34, 49
 definiendo5, 24, 48, 49
 definir6, 7
 delimitado19, 20, 35, 41
 delimitador48, 49
 delimitados24, 25, 49
 demonios8, 12, 17
 demuestra59
 dentro12, 13, 15, 18, 19, 22, 23, 51
 derby15
 desactiva23, 29, 51
 designada54, 56
 detalla60, 62
 detallado21, 22
 detalles9, 15, 26, 27, 29
 dfs12, 13, 17, 18, 20, 22, 31, 43, 55
 diferencia57
 diferentes62
 dinámica42
 dirección21, 25
 directamente14, 35, 56

directorio. 7, 10, 11, 13, 14, 15, 19, 21, 22, 25,
 26, 30, 31, 32, 33, 34, 35, 38, 47, 54, 55,
 62
 directorios10, 12, 13, 20, 21, 22, 43
 disco47, 59
 diseño59, 61
 disponibles9, 18, 22, 50
 distintos40
 distribución49, 54, 55, 57, 58
 distribuido22, 51, 62
 distribuidos17, 54, 61, 62
 documento4, 62
 dos18, 46, 54

E

echo7, 12
 editor4, 5, 6, 11
 eficiencia60
 eficiente50, 58, 59, 61, 62
 ejecución4, 8, 9, 12, 16, 17, 46, 47, 51
 ejecuta.. 10, 11, 12, 13, 15, 16, 17, 26, 30, 33,
 36, 38, 47
 ejecutamos20, 27, 31, 43
 ejecutan14
 ejecutando8, 26, 43
 ejecutar17, 18, 28, 32, 34, 37, 62
 ejemplos28
 elimina34, 37
 eliminar32, 35
 enable14
 encabezado19, 20, 24, 25
 enforce49, 50, 56, 57
 entorno5, 6, 7, 11, 12, 18, 22, 62
 entornos58, 59, 61
 errores16, 17, 50, 51
 escaneo55, 56
 escaneos57, 58, 59
 esenciales33, 36, 38
 especialmente47, 49, 58
 específica14, 34, 47
 especificando5, 48
 especificar7, 28, 37
 específico34, 43, 57, 62
 esperaba57, 58
 esquema14, 15, 24, 59
 establece24, 32, 34, 37, 44
 estándar16, 17
 estructura10, 34, 35, 43, 54, 57, 59
 evitando23, 50, 56, 57
 evitar15, 50, 51, 59
 exacta13, 15
 exitosa33, 35, 38
 explicación34, 36
 explicaciones4, 61
 extensión15
 externa24, 25, 32, 34, 35, 38, 62
 externas34, 58, 60, 61

F

fabricante20, 28, 62
 fabricantes20, 29, 61
 false14, 23, 29

fecha 22, 26, 27, 44, 45, 46, 52
 fichero 4, 5, 23, 30, 32, 34, 37, 61, 62
 ficheros 4, 31
 filas 27, 28, 31, 33, 38, 44, 45, 46, 47
 filtra 43, 47
 filtrados 52, 53, 54
 filtrar 15, 16, 58
 filtros 56, 58
 fin 11, 15
 find 13, 15
 formato 29, 32, 35, 41, 55, 57, 58, 62
 framework 4, 5
 frecuencia 40, 41
 frecuentemente 47, 58
 funcionamiento 12, 49, 61
 fundamental 47

G

garantizar 5, 49
 generando 54, 56
 gestión 17, 59, 60
 gestiona 17
 gestionadas 31
 grandes 50, 58
 grep 10, 14, 15, 16
 guardar 5, 7, 11, 52
 guía 61

H

habilita 42, 49
 habilitar 6, 42
 haciendo 47
 hadoop 14, 21, 25, 30, 32, 35, 38
 hash 47, 56
 hashing 54, 55, 56, 57
 hdfs 12, 13, 18, 20, 22, 31, 43, 55
 head 14
 header 24, 25
 hive.10, 11, 12, 13, 14, 18, 23, 29, 31, 35, 43,
 49, 50, 55, 56, 57
 home 21, 25, 30, 32, 35, 38
 hql...23, 25, 26, 28, 30, 32, 33, 34, 35, 36, 37,
 38, 61, 62
 https 10, 61

I

ignorar 24, 25
 igual 47, 57
 impacto 58, 60
 impl 15
 incluso 17, 29
 incluyen 57, 58
 incluyendo .. 4, 9, 17, 18, 19, 20, 27, 28, 61, 62
 incorrecta 57
 indica 38
 indicando 15, 18, 61
 información 19, 20, 22, 48, 61, 62
 ingresos 27, 44, 45, 46
 inicia 17
 iniciar 8, 16

innecesarios 26, 57, 59
 inserción 42
 insertan 18, 43
 insertar 42, 49
 instalación 4, 9, 10, 13, 15, 60, 62
 integración 5, 62
 interfaz 9
 interna 29, 37, 41, 61, 62
 internas 31, 55, 60
 inventario 28, 29, 31, 35, 37, 61

J

jar 15
 java 14
 jdbc 17, 26, 30, 33, 36, 38
 join 23, 28, 29, 47, 56
 joins 29, 54, 58
 jps 8, 12
 junto 22, 26, 40

L

lectura 31, 40
 leer 43, 47, 61
 leídos 47, 58
 lib 15
 line 24, 25
 línea 19, 20, 24, 25, 28
 lista 30, 31, 33, 35, 38, 40, 43, 54
 listado 21, 22, 44, 45, 46
 listar 8, 11, 13, 15, 16, 38

LI

llamada . 18, 23, 24, 25, 29, 32, 35, 41, 42, 48,
 49, 61, 62

L

local....4, 22, 30, 32, 33, 35, 36, 38, 50, 51, 60
 localhost 17, 26, 30, 33, 36, 38
 localizar 13, 15
 log 16, 17
 lógico 47, 57
 logs 7, 9
 luego 26, 31, 42, 43, 44, 47
 lugar 43, 51

M

manera 4, 26, 62
 manteniendo 23, 35
 mantienen 34
 manual 4, 38
 manufacturer 20, 21, 22, 25, 28, 29, 61
 manufacturerid 28
 map 23, 29, 50, 56
 mapa 60, 62
 mapper 52, 56
 mapred 4, 5, 6

mapreduce 5, 6
 máquina 4, 20, 21, 25, 26
 mayor 40, 41, 47, 58
 mediante 7, 9, 29, 33, 36, 50, 52, 53, 54, 55, 56, 61
 mejora 40, 58, 59
 mejorar 47, 59, 61
 memoria 50, 51
 menor 40, 41, 58
 menos 37, 46, 47, 58, 60
 mensajes 18, 26, 30, 33, 36, 38
 mental 60
 metadatos 34, 37
 metastore 4, 15, 34, 37, 55, 60
 mientras 55
 misma 35, 56
 mkdir 10, 12, 20
 modificación 22, 26
 modificaciones 33, 36, 38
 modificar 5, 6, 14
 modo 17, 51
 mostrando 15, 18, 22, 26, 31, 33, 38, 55, 62
 mostrar 16, 18, 26, 29, 31, 35, 38, 40, 44, 46
 mostrarán 34
 muestra 17, 18, 21, 27, 28, 41, 44, 45, 46

N

name 5, 13, 14, 15, 18
 nano 5, 6, 11
 necesarias 58, 61, 62
 necesarios 12, 20, 62
 nodos 9, 17
 nohup 16, 17
 nombre 19, 20, 28, 31, 37, 62
 nueva 10, 18, 31, 41, 43
 nulos 52, 53
 número 40, 47, 56

O

obliga 49, 56
 obtener 44, 45
 obtenido 44, 45, 46, 47
 obtenidos 52, 53
 opción 44, 49
 operación 48, 49, 50, 56
 operaciones 19, 23, 28, 32, 33, 34, 36, 37, 47, 50, 51, 54, 56, 58
 opt. 10, 11, 12
 optimiza 54, 59
 optimización 47, 50, 56, 58, 61
 optimizado 57, 58
 optimizando 55, 56
 optimizar 47, 54
 optimizó 57
 ordenador 21, 25
 org. 10, 61
 organiza 48, 54
 organización 49, 58
 original 10, 47

P

país 27, 41, 42, 43, 44, 45, 46, 47, 62
 países 40, 47
 pantalla 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 26, 27, 31, 33, 38, 43, 44, 45, 46
 parámetros 4, 57
 parte 26, 30
 partes 40, 47
 partición 42, 45, 46, 47
 particionada 41, 42, 43, 45, 46, 47, 58
 particionamiento 40, 47, 58, 60, 61
 particiones 40, 42, 43, 47, 58, 59, 61
 pasos 4, 62
 pdf 62
 permisos 5, 10, 13, 22, 26, 31
 permite 38, 40, 43, 58, 62
 permitiendo 4, 16, 17, 18, 28, 37, 50
 pesar 57
 plano 16, 17
 plantilla 13
 pmga20 22, 23, 24, 25, 26, 27, 28, 31, 32, 33, 34, 35, 37, 38, 43, 55, 61
 poblar 42
 poda 47
 posiblemente 57
 posteriores 23, 28, 32, 34, 37, 62
 práctica 58, 59, 61
 pregunta 31
 preguntas 46
 presentes 22, 40
 presionamos 27
 previamente 14, 34
 primera 19, 20, 24, 25, 26, 47
 primeras 27, 28, 31, 33, 38
 principales 12, 17
 probables 57
 procede 26, 38
 procedemos 5, 7, 11, 38
 procesados 47, 55
 procesamiento 4, 17, 22, 23, 50, 57, 62
 procesar 47, 52
 proceso 4, 15, 17, 62
 procesos 8, 16, 17, 62
 product 19, 20, 21, 22, 24, 28, 29, 52, 53, 54, 61
 productid 52, 53, 54
 producto 19, 27, 28, 62
 productos 19, 20, 28, 29, 61, 62
 propiedad 14, 24, 25
 propiedades 5, 6, 61
 propietario 13, 22, 26
 protocolo 30, 33
 provenientes 61, 62
 pruebas 18
 puede 40, 47, 58
 puerto 26, 30, 33, 36, 38
 pulsar 5, 11
 punto 20, 25
 put 22

R

rápida 47, 57
 razonada 43

realiza 17, 44, 46, 61, 62
 realización 4, 62
 realizados 4, 7, 14
 realizando 55, 56
 realizar 33, 36, 40, 43, 62
 recrear 34
 recursos 9, 17, 50, 59
 redirigiendo 16, 17
 reducir 52, 56
 reducers 29, 52
 reduciendo 47, 58
 reducir 47, 55, 56, 57
 registro 16, 42
 registros.. 7, 18, 33, 34, 38, 44, 45, 46, 49, 52,
 54, 55, 56, 57
 relacionados 17, 20
 remoto 21, 25, 30, 32, 35, 38
 rendimiento 40, 43, 47, 51, 58, 59, 61
 requiere 47, 56
 respectivamente 7, 61
 respuesta 58, 59
 resultado 17, 29, 43, 44, 47, 57, 62
 resultados 18, 40, 55
 resumen 47, 55
 ruta 7, 13, 15, 31
 rutas 4, 5

S

sabe 47, 58
 sales 19, 20, 21, 22, 24, 27, 40, 41, 42, 43, 44,
 45, 46, 47, 52, 53, 54, 61, 62
 salida 15, 16, 17, 26
 salir 5, 7, 11
 scp 21, 25, 30, 32, 35, 38
 script 8, 23, 26, 30, 33, 36, 38, 61
 sed 14
 segmento 20, 28, 62
 según 42, 43, 49, 55, 56
 segunda 30, 46, 47
 segundo 16, 17
 segundos 44, 45, 46, 48, 49, 50, 52, 54, 56, 57
 segura 21, 25
 selecciona 23, 28, 43
 seleccionamos 31, 33
 separadores 29, 32
 ser 34, 37, 47, 58
 servicio 6, 16
 servicios 8, 17, 62
 servidor.. 16, 17, 26, 30, 32, 33, 35, 36, 38, 51
 sesión 7, 12, 16, 17
 shell 6, 11
 sido 15, 21
 siempre 56, 59
 siendo 41, 47, 52
 significativamente 40, 47, 57
 sigue 31
 siguen 39
 silent 17, 26, 30, 33, 36, 38, 44
 sistema 12, 17, 18, 22, 31, 47
 site 4, 5, 6, 13, 14
 skip 24, 25
 sobrescribe 52, 53
 sobrescribiendo 29, 43
 solicitando 21, 25

solo 34, 43, 47, 57, 58, 59, 62
 source 7, 12
 staging 19, 54
 start 8, 17
 subdirectorio 43
 subdirectorios 43
 subyacentes 32, 37
 sudo 4, 5, 10, 14
 suprimiendo 30, 33, 36, 38
 system 14

T

tabla 18, 19, 24, 25, 27, 28, 29, 31, 32, 33, 34,
 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46,
 47, 48, 49, 52, 53, 54, 55, 56, 57, 58, 61,
 62
 tablas.... 27, 28, 29, 31, 34, 37, 48, 49, 52, 53,
 55, 56, 57, 58, 59, 60, 61, 62
 table 18
 tamaño 22, 26, 31, 55
 tar10
 tardado 46
 tardó 15, 47, 48, 50, 56
 tarea 50
 tareas 4, 17, 62
 tecla 5, 7, 11, 27, 43
 técnica 40, 47, 58
 técnicas 59, 61
 template 13
 temporal 14, 19, 51, 54
 terminal 7, 12, 16, 17, 20
 testdb 18
 texto 4, 5, 6, 11, 24, 25, 29, 32, 35, 41, 42
 tiempo 46, 47, 57, 58
 tmp 12, 13, 14, 16, 17
 toda 16, 43
 todas 23, 51
 total 62
 trabajar 18, 37
 transferencia 33, 35, 38
 transfiere 21, 25, 30, 32, 35, 38
 transformación 23, 62
 través 18, 30, 38
 true .. 14, 17, 26, 30, 33, 36, 38, 44, 49, 50, 56

U

ubicación 24, 25, 31, 32, 34, 35, 38
 últimas 16
 único 43, 55, 56
 unidades 27, 62
 units 52
 usando.. 11, 12, 13, 30, 33, 35, 38, 48, 49, 56,
 57, 61
 usar 29, 51, 61
 user.12, 13, 14, 18, 20, 22, 24, 25, 31, 32, 34,
 35, 43, 55
 uso 5, 9, 50, 57, 58, 59, 61, 62
 usuario 6, 11, 14, 21, 25, 30, 33, 38
 útil 58
 utiliza 13, 22, 47, 55, 62
 utilizado 4, 19

utilizando... 4, 8, 10, 13, 21, 25, 26, 29, 32, 54,
61

V

vacía 33, 38
vacíos 52, 53
valor 12, 14, 43, 57
valores 7, 40, 52
value 14
variable 11, 12
variables 6, 7, 11, 12, 14
varias 4, 42, 58
veces 40
vendidas 27, 62
ventas ... 19, 20, 27, 40, 41, 43, 44, 45, 46, 48,
49, 52, 53, 54, 55, 56, 57, 58, 61, 62
ver 43
verifica 22, 26
verificando 7, 11, 16, 30
verificar 12, 13, 14, 16, 18, 22, 27

vinculada 38
vinculándola 32, 35
virtual 4, 20, 21, 25, 26
volumen 55, 57, 58
volúmenes 50, 58

W

warehouse 12, 13, 18, 31, 35, 43, 55
web 9

X

xml 4, 5, 6, 13, 14

Y

yarn 5, 6, 8, 17