Instituto Politécnico Nacional

Escuela Superior de Computo

Sistemas Distribuidos

Tarea extra 1

Procesamiento de tablas agregadas utilizando MapReduce

Nombre del alumno:

García Quiroz Gustavo Ivan

Grupo: 7CV4

Nombre del profesor: Guerrero Carlos Pineda

Fecha de entrega: 17/11/2025

ÍNDICE

1 Introducción

Este trabajo implementa, de extremo a extremo, un flujo de procesamiento de datos para obtener tablas agregadas mediante MapReduce sobre una VM de Azure. Se creó una máquina virtual Ubuntu en la región Canadá con el nombre TE1-2022630278; se instaló y configuró MySQL, se creó la base de datos practica_olap, se ejecutó el script practica_olap.sql y se cargaron datos desde sales_data.csv. A partir de la fact_table se generó el archivo country_category_product.csv, que sirvió como entrada para una aplicación Hadoop en Java (AggregationMapper, AggregationReducer y AggregationDriver) ejecutada en modo local, la cual acumuló las ventas por id_country, id_category e id_product. El resultado del proceso (part-r-00000) se transformó a CSV y se cargó en la tabla agregada country_category_product. Finalmente, se consultaron los cubos requeridos (ventas por country; por country y category; y por country, category y product) apoyándose exclusivamente en la tabla agregada. Todo el proceso se realizó desde Windows 11 usando SFTP únicamente cuando fue necesario transferir archivos.

En esta tarea se usó la IA generativa de GitHub copilot para guiarse mejor y completar las instrucciones más rápido. El enlace se encuentra al final de este reporte además de los resultados: tabla agregada country_category_product poblada con ventas acumuladas por país, categoría y producto.

> NOTA: La plataforma de Moodle no permitió subir el reporte en buena calidad debido al límite de 2 MB por archivo. Sugiero revisar el siguiente documento de Google drive con el reporte en buena calidad: https://drive.google.com/file/d/1adBCfOKGC740AYOSHVSum924Mud-BB1O/view?usp=sharing

2 Objetivos

2.1 Objetivo general

Implementar un flujo completo de aprovisionamiento, carga, agregación con MapReduce y consulta OLAP que produzca y utilice la tabla agregada country_category_product para responder cubos de ventas con eficiencia.

2.2 Objetivos específicos

- Provisionar una VM Ubuntu en Azure for Students (región Canada) con la denominación TE1-2022630278 y las especificaciones solicitadas.
- Instalar y configurar MySQL; crear la base practica_olap; ejecutar practica_olap.sql; cargar sales_data.csv.
- Poblar las tablas de dimensiones y la fact_table a partir de sales_data.
- Generar el archivo country_category_product.csv como entrada al proceso de MapReduce.
- Instalar y configurar Apache Hadoop (modo local) y Java, incluyendo variables de entorno y hadoop-env.sh.
- Desarrollar, compilar y ejecutar un job MapReduce en Java que acumule ventas por id_country, id_category e id_product.
- Convertir el resultado del job a CSV y cargarlo en la tabla agregada country_category_product.
- Ejecutar las consultas OLAP solicitadas (por country; por country y category; por country, category y product) usando únicamente la tabla agregada.
- Documentar el proceso con evidencias y capturas puntuales para cada hito técnico.

3 Requerimientos y materiales

Para ejecutar la práctica de principio a fin se utilizó una máquina virtual en Azure for Students (región Canada) con Ubuntu y el stack MySQL + Hadoop en modo local. El entorno se mantuvo deliberadamente simple para enfocarse en la generación de la tabla agregada y su consulta.

En términos de cómputo, se empleó una VM con 2 vCPU, 4 GB de RAM y disco SSD estándar, suficiente para ejecutar MySQL y un job de Hadoop MapReduce en modo standalone. El sistema operativo fue Ubuntu 22.04 LTS, lo que garantiza compatibilidad con los paquetes de MySQL y OpenJDK. Desde el equipo cliente (Windows 11), se usó SSH para administración y SFTP únicamente para transferir los archivos proporcionados por el curso (practica_olap.sql y sales_data.csv). En Azure, se expuso únicamente el puerto 22 (SSH) y el resto de la operación se realizó dentro de la VM.

Materiales utilizados (software y archivos):

- Ubuntu Server 22.04 LTS (VM Azure: TE1-2022630278, región Canada)
- MySQL Server y cliente (con soporte de LOAD DATA INFILE/SELECT INTO OUTFILE mediante secure-file-priv)
- OpenJDK (Java 17 recomendado; Java 16 aceptable si el entorno lo requiere)
- Apache Hadoop 3.4.0 en modo local (standalone)
- Herramientas de sistema: nano, wget, tar, unzip
- Cliente SFTP/SSH en Windows 11 (OpenSSH integrado)
- Archivos de la plataforma:
    - practica_olap.sql (definición de tablas: sales_data, dimensiones, fact_table y agregadas)
    - sales_data.csv (datos de ventas con encabezados)

Requisitos de cuentas y permisos:

- Acceso SSH a la VM con usuario con privilegios sudo
- Acceso a MySQL como root (o usuario con permisos de creación/carga)
- Permisos sobre el directorio de secure-file-priv de MySQL (se usó /var/lib/mysql-files)

Notas prácticas:

- secure-file-priv: se trabajó con /var/lib/mysql-files para INFILE/OUTFILE y evitar errores de permisos
- Arquitectura de la VM: x86_64 (usual en Azure); si la VM fuera ARM, se usaría el tarball aarch64 de Hadoop
- Transferencia de archivos: SFTP se utilizó solo para subir practica_olap.sql y sales_data.csv

4 Arquitectura y entorno de trabajo

La arquitectura implementada sigue un flujo lineal: los datos se ingieren en MySQL, se normalizan en dimensiones y fact_table, se exportan a CSV y se procesan con Hadoop MapReduce para obtener agregados, que se reimportan a MySQL en una tabla agregada optimizada para consultas. Todo corre en una sola VM (sin HDFS/YARN), aprovechando Hadoop en modo standalone y el sistema de archivos local.

Descripción del flujo:

- Ingesta y modelado: sales_data.csv se carga en MySQL; a partir de ella se pueblan tablas de dimensiones, fechas y fact_table
- Extracción a CSV: se genera country_category_product.csv mediante SELECT … INTO OUTFILE
- Procesamiento MapReduce: un job en Java (Mapper, Reducer, Driver) acumula ventas por id_country, id_category e id_product
- Postproceso y carga: el resultado part-r-00000 se convierte a CSV (tabulador → coma) y se carga en la tabla agregada country_category_product
- Consultas OLAP: los cubos se ejecutan exclusivamente sobre la tabla agregada para reducir costos de cómputo y latencia

Componentes y configuración clave:

- Host de cómputo: Azure VM TE1-2022630278 (Ubuntu 22.04, 2 vCPU, 4 GB RAM, SSD estándar, región Canada)
- Base de datos: MySQL con directorio de intercambio /var/lib/mysql-files (secure-file-priv)
- Hadoop: 3.4.0 en modo local, sin HDFS; entrada/salida en el FS local (por ejemplo, prueba/input y prueba/output)
- Java: OpenJDK (JAVA_HOME definido); Hadoop configurado en hadoop-env.sh

- Cliente: Windows 11 para acceso SSH y SFTP (solo cuando es necesario copiar archivos)

Variables de entorno (referenciales):

- JAVA_HOME (por ejemplo, /usr/lib/jvm/java-17-openjdk-amd64)
- HADOOP_HOME (/home/usuario/hadoop-3.4.0)
- PATH extendido con $JAVA_HOME/bin y $HADOOP_HOME/bin

Consideraciones de red y seguridad:

- Puerto 22 abierto para administración (SSH/SFTP)
- Descargas de paquetes y binarios desde repositorios oficiales (apt y Apache)
- No se exponen puertos de MySQL al exterior; se opera localmente dentro de la VM

Estructura de trabajo (local):

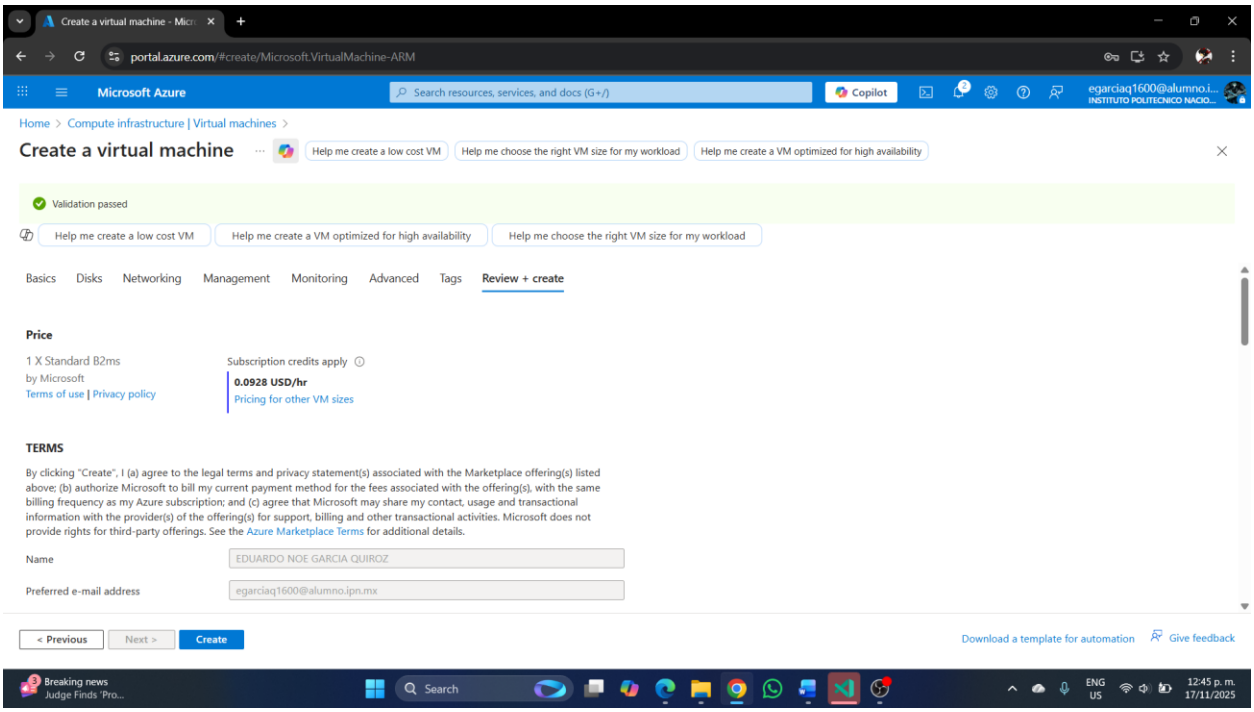
- Directorio del proyecto MapReduce (por ejemplo, ~/prueba) con fuentes Java, build y Aggregation.jar
- Directorios input y output para el job de Hadoop
- Archivos CSV intermedios y finales ubicados en /var/lib/mysql-files para facilitar INFILE/OUTFILE

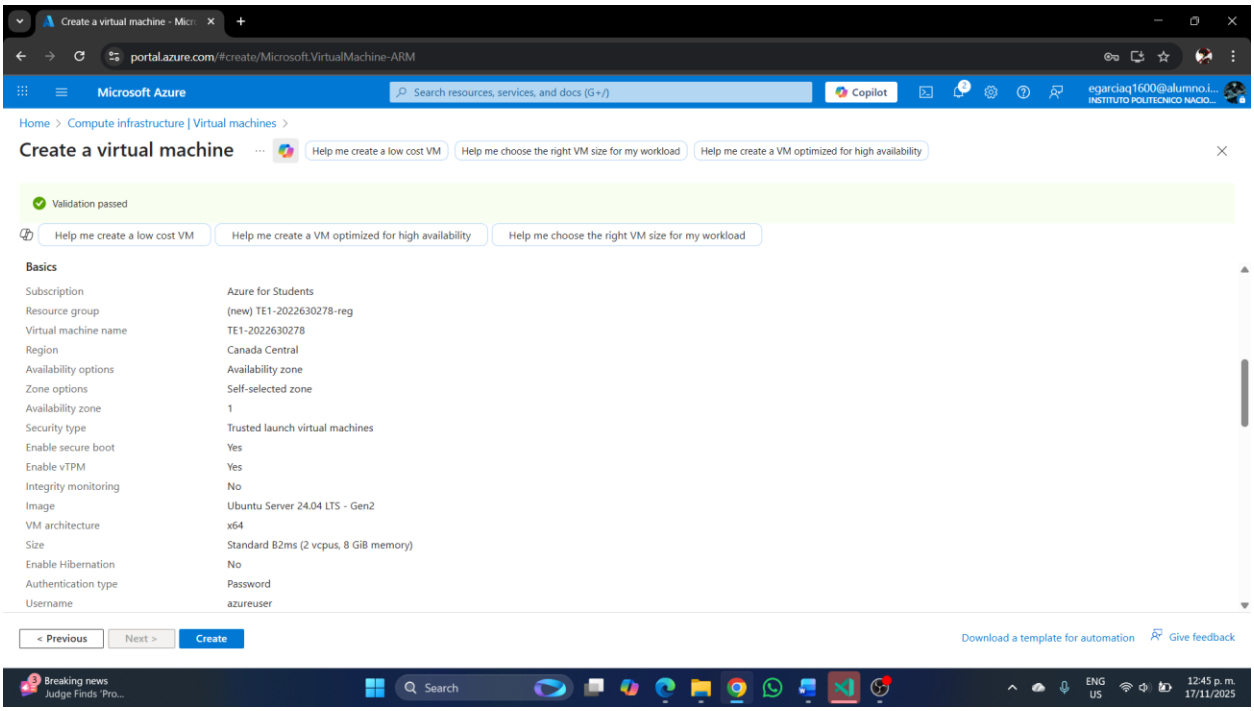
5 Procedimiento

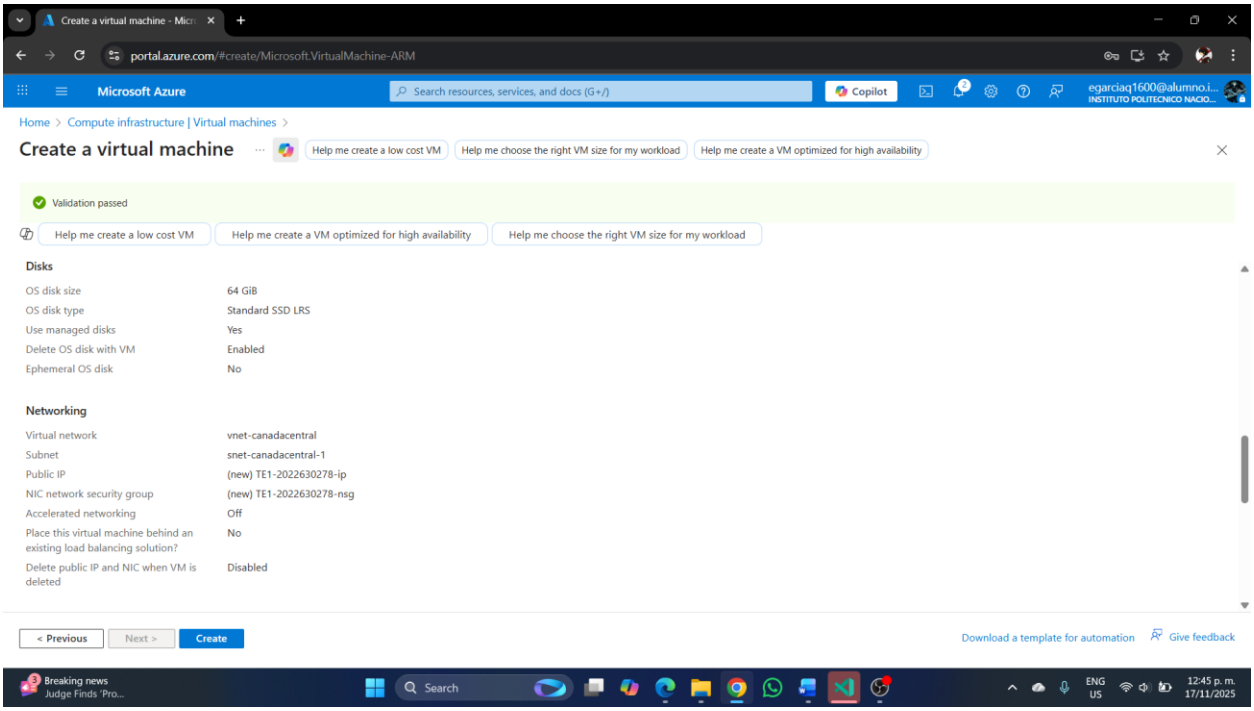
5.1 Creación de la máquina virtual en Azure

Se creó una máquina virtual en Azure for Students con Ubuntu 22.04 LTS en la región Canada. Se asignó el nombre TE1-2022630278, 2 vCPU, 4 GB de RAM y disco SSD estándar. Se habilitó únicamente el puerto 22 para SSH. Al finalizar, se verificó la IP pública desde el panel de la VM.

- Parámetros clave: Nombre TE1-2022630278, Región Canada, Tamaño Standard_B2s, Disco Standard SSD, Puerto 22 (SSH).




Create a virtual machine - Micr
portal.azure.com/#create/Microsoft.VirtualMachine-ARM
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
egarciaq1600@alumno.i...
INSTITUTO POLITECNICO NACIO...
Home > Compute infrastructure | Virtual machines >
Create a virtual machine
Help me create a low cost VM
Help me choose the right VM size for my workload
Help me create a VM optimized for high availability
Validation passed
Help me create a low cost VM
Help me create a VM optimized for high availability
Help me choose the right VM size for my workload
Basics
Subscription Azure for Students
Resource group (new) TE1-2022630278-reg
Virtual machine name TE1-2022630278
Region Canada Central
Availability options Availability zone
Zone options Self-selected zone
Availability zone 1
Security type Trusted launch virtual machines
Enable secure boot Yes
Enable vTPM Yes
Integrity monitoring No
Image Ubuntu Server 24.04 LTS - Gen2
VM architecture x64
Size Standard B2ms (2 vcpus, 8 GiB memory)
Enable Hibernation No
Authentication type Password
Username azureuser
< Previous
Next >
Create
Download a template for automation
Give feedback
Breaking news
Judge Finds 'Pro...
Search
ENG
US
12:45 p. m.
17/11/2025


Create a virtual machine - Micr
portal.azure.com/#create/Microsoft.VirtualMachine-ARM
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
egarciaq1600@alumno.i...
INSTITUTO POLITECNICO NACIO...
Home > Compute infrastructure | Virtual machines >
Create a virtual machine
Help me create a low cost VM
Help me choose the right VM size for my workload
Help me create a VM optimized for high availability
Validation passed
Help me create a low cost VM
Help me create a VM optimized for high availability
Help me choose the right VM size for my workload
Disks
OS disk size 64 GiB
OS disk type Standard SSD LRS
Use managed disks Yes
Delete OS disk with VM Enabled
Ephemeral OS disk No
Networking
Virtual network vnet-canadacentral
Subnet snet-canadacentral-1
Public IP (new) TE1-2022630278-ip
NIC network security group (new) TE1-2022630278-nsg
Accelerated networking Off
Place this virtual machine behind an existing load balancing solution? No
Delete public IP and NIC when VM is deleted Disabled
< Previous
Next >
Create
Download a template for automation
Give feedback
Breaking news
Judge Finds 'Pro...
Search
ENG
US
12:45 p. m.
17/11/2025

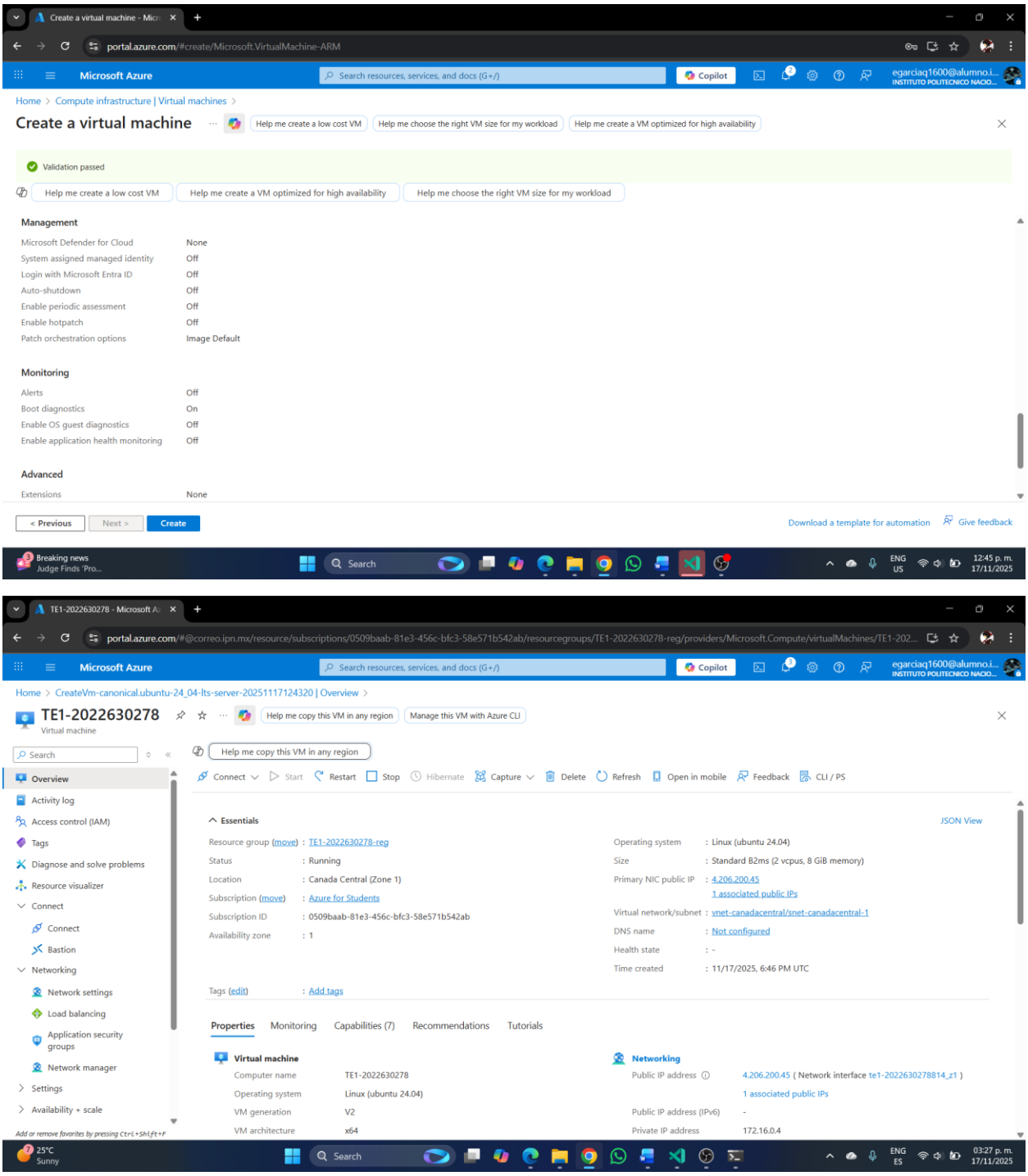


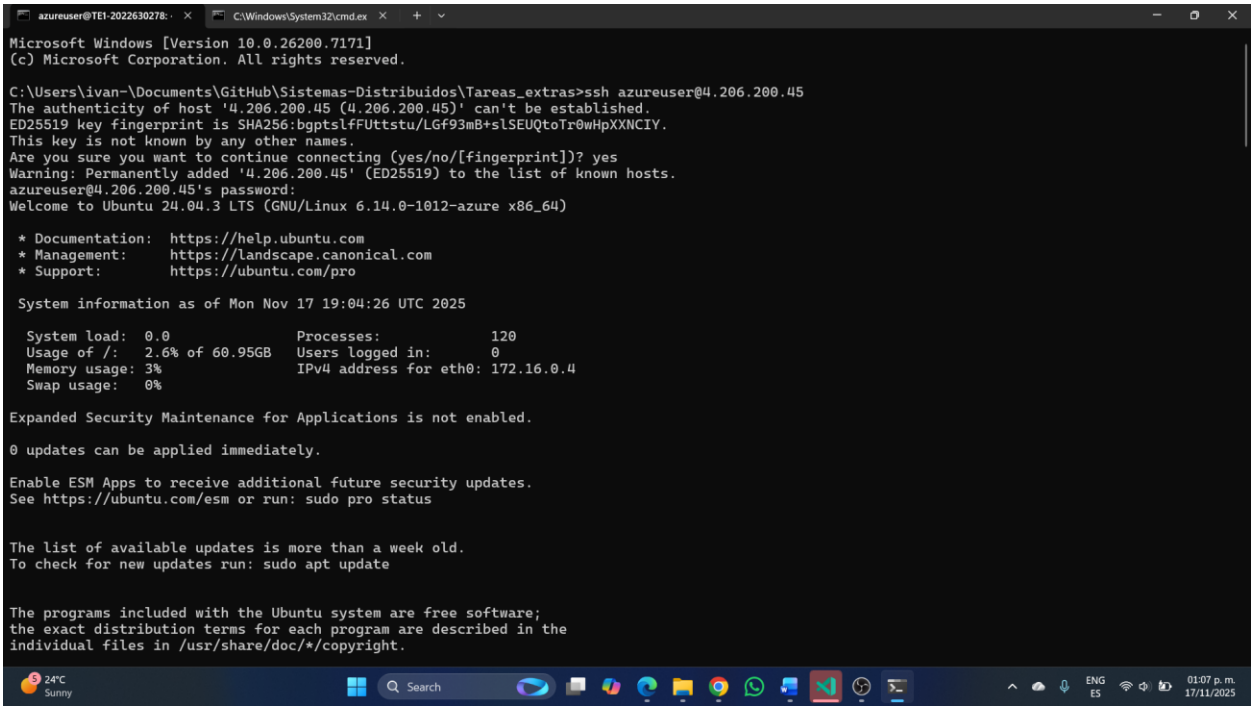
Imagen 7: Resumen de la VM creada (Overview en Azure Portal mostrando nombre TE1-2022630278, tamaño, región y IP pública).

5.2 Instalación y configuración de MySQL

Se accedió por contraseña desde Windows 11 y se actualizó el sistema. Posteriormente, se instaló MySQL Server y herramientas básicas de administración.

ssh azureuser@4.206.200.45

sudo apt update && sudo apt -y upgrade



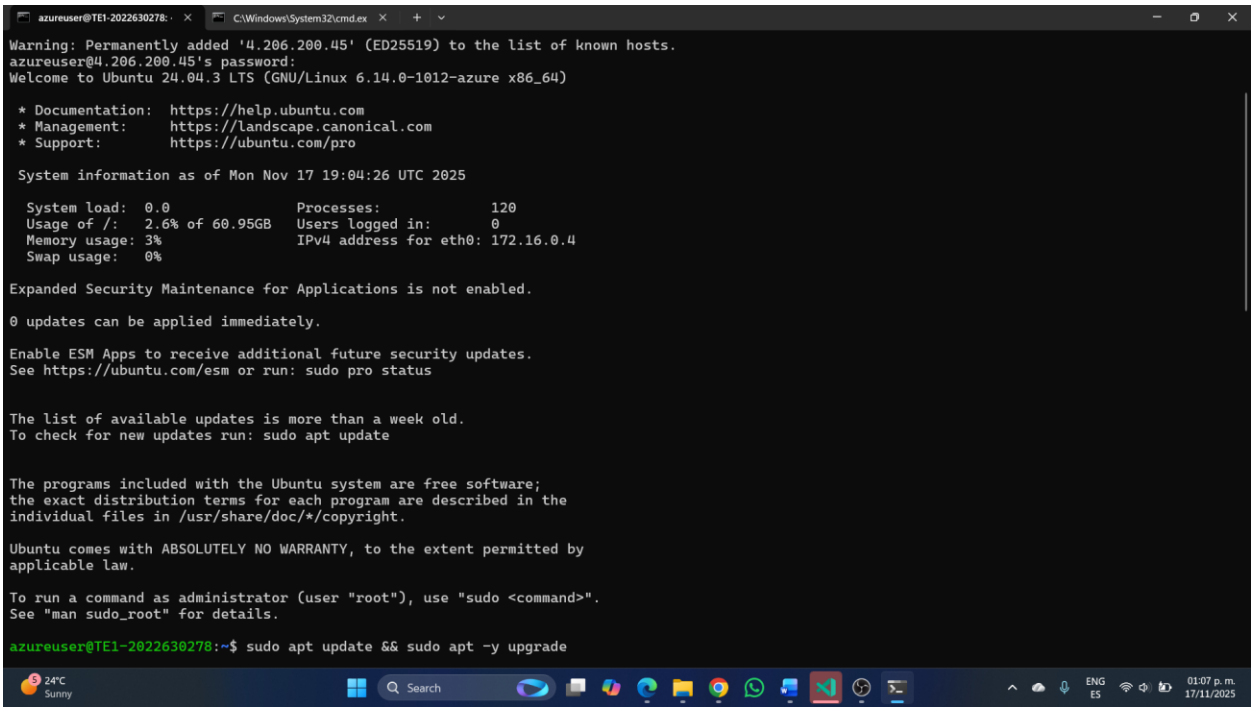


Imagen 7.1 Inicio de sesión en la teminal.

sudo apt -y install mysql-server mysql-client nano wget tar unzip

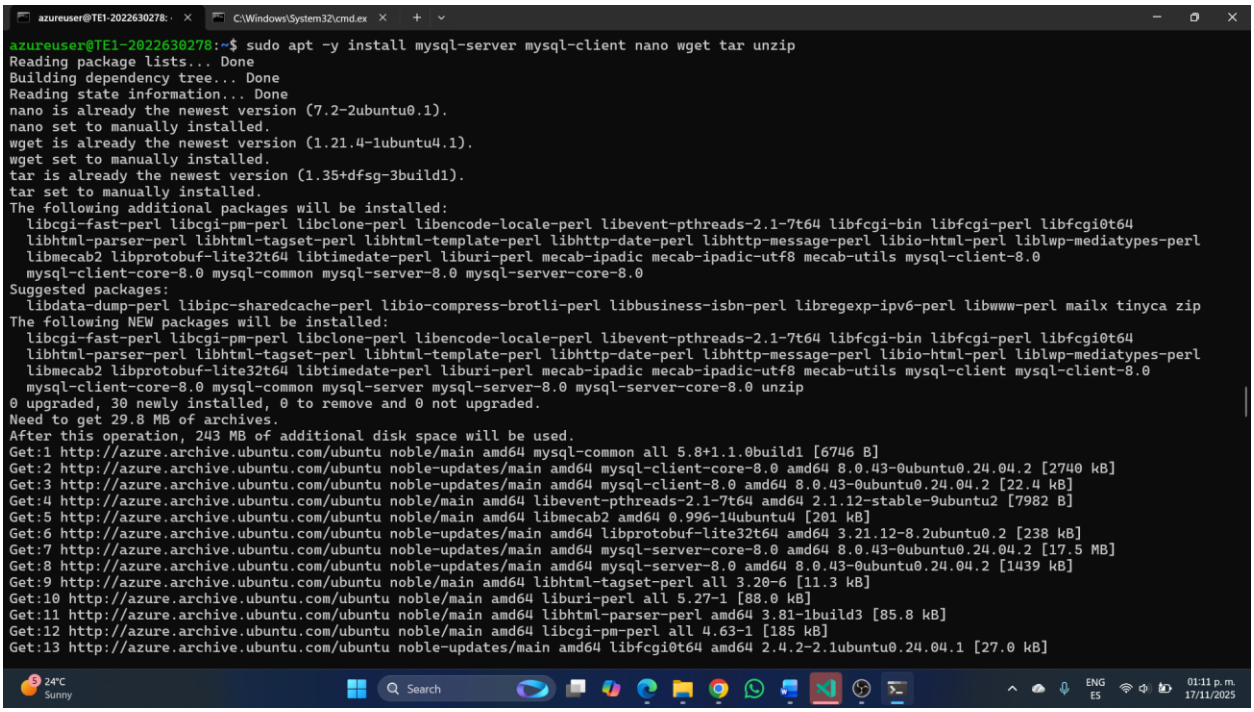


Imagen 7.2 Instalación de MySQL.

Se verificó que el servicio MySQL quedara activo:

sudo systemctl status mysql

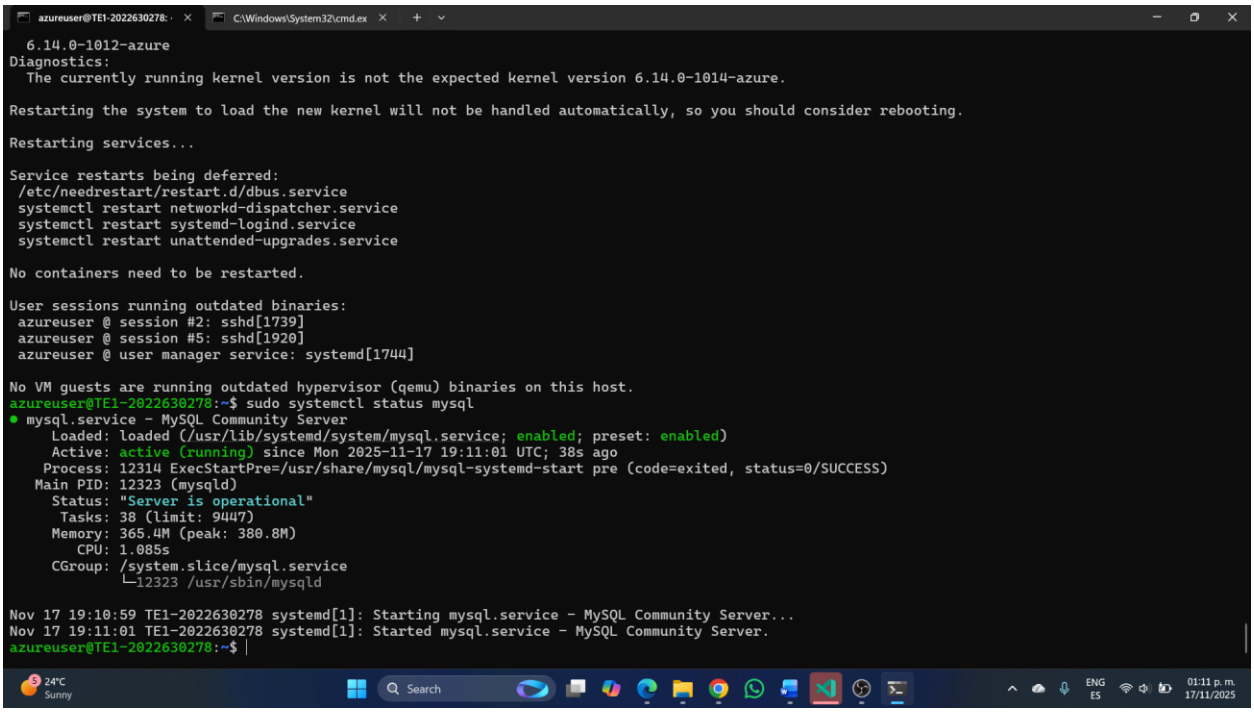


Imagen 7.3 Verificación de MySQL.

Se preparó el directorio permitido por secure-file-priv para operaciones INFILE/OUTFILE:

sudo mkdir -p /var/lib/mysql-files

sudo chown mysql:mysql /var/lib/mysql-files

sudo chmod 750 /var/lib/mysql-files

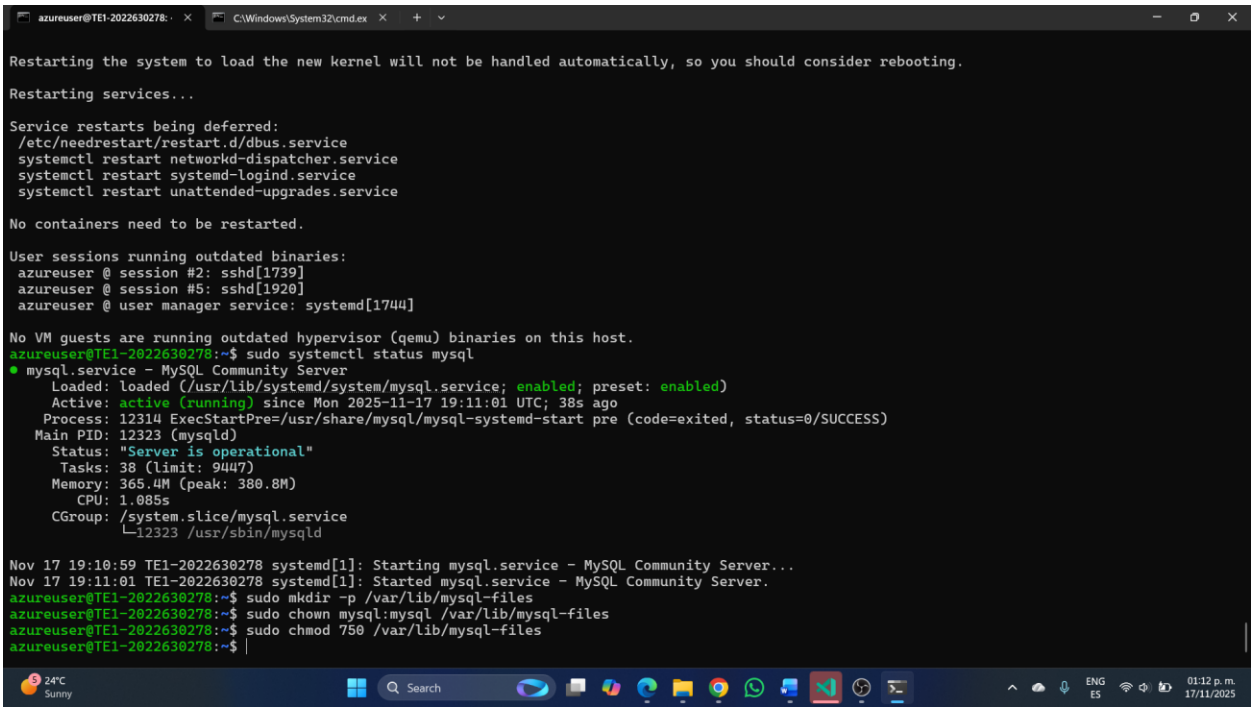


Imagen 8: MySQL instalado y servicio activo (systemctl status mysql en “active (running)”).

5.3 Creación de la base de datos y ejecución del script practica_olap.sql

Se transfirió el archivo practica_olap.sql desde Windows 11 a la VM usando SFTP.

- Desde Windows 11 (PowerShell o Windows Terminal)

sftp azureuser@4.206.200.45

put practica_olap.sql

exit

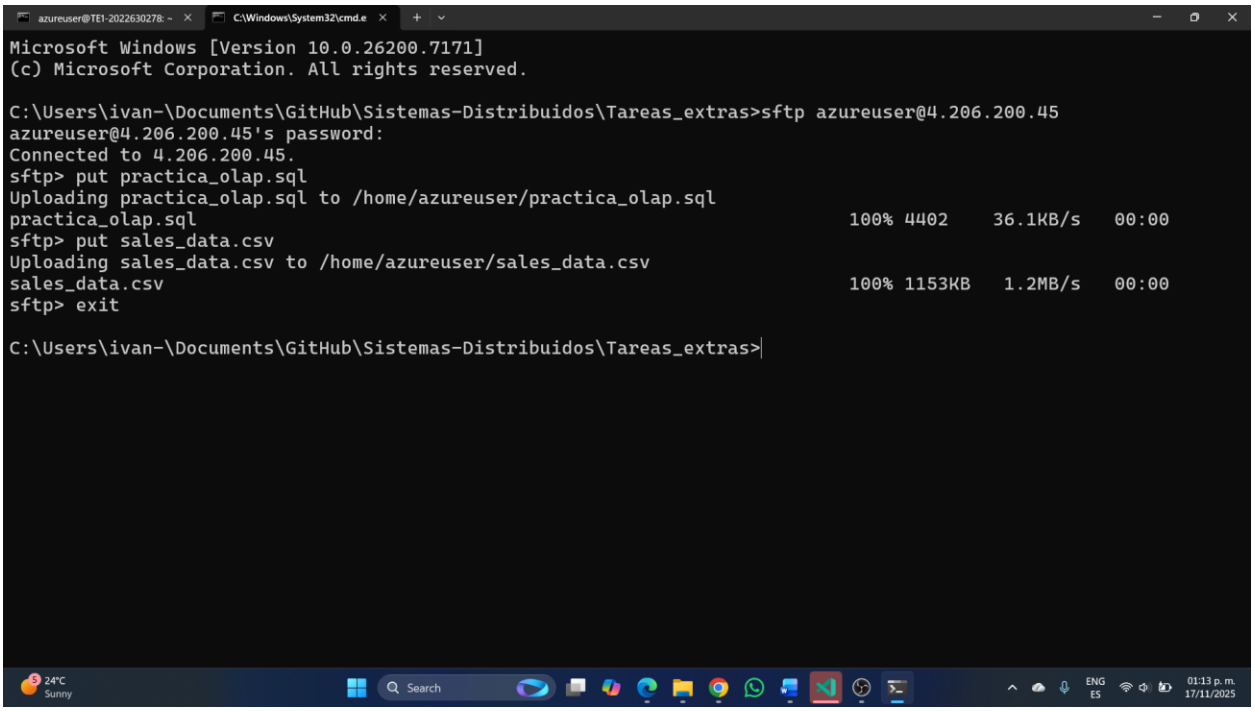


Imagen 8.1 Copiar y pegar archivos a las VM's.

Se creó la base de datos practica_olap y se ejecutó el script con la definición de tablas (incluyendo dimensiones, fact_table y tablas agregadas).

sudo mysql

CREATE DATABASE practica_olap;

USE practica_olap;

SOURCE /home/azureuser/practica_olap.sql;

SHOW TABLES;

EXIT;

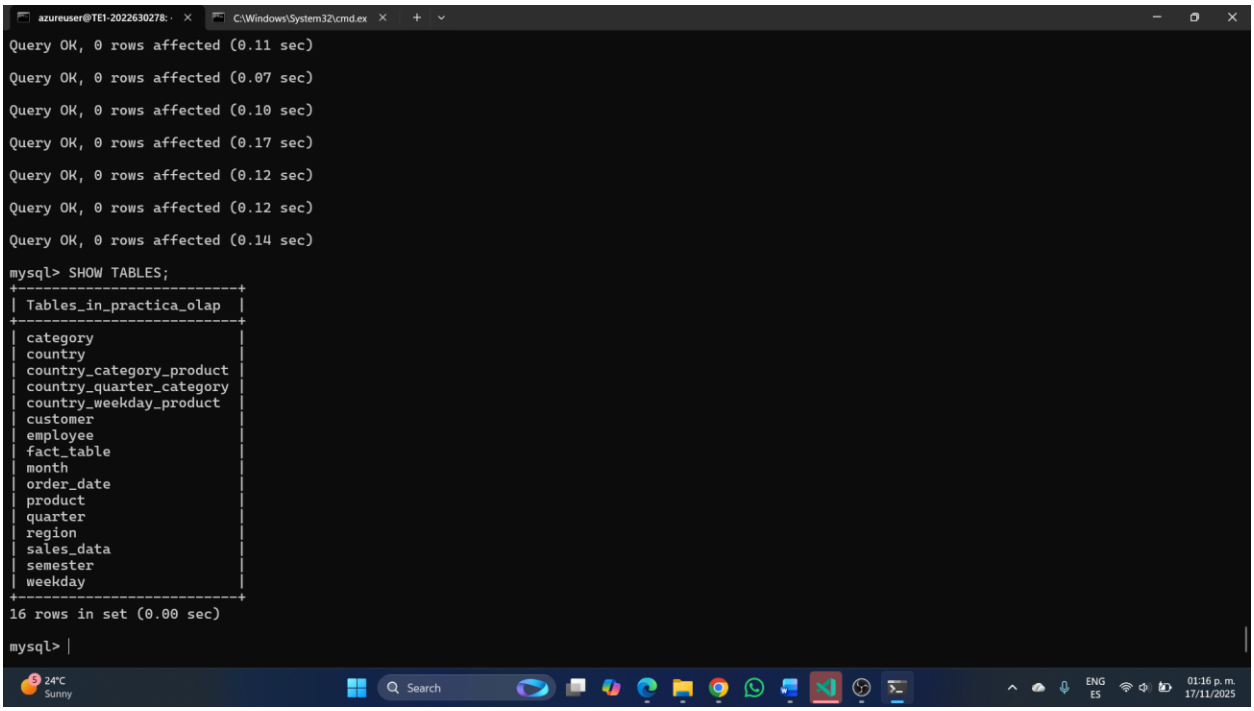


Imagen 9: Evidencia de creación de la base (SHOW TABLES mostrando sales_data, dimensiones, fact_table y tablas agregadas).

5.4 Carga de datos a la tabla sales_data

Se transfirió sales_data.csv mediante SFTP y se colocó en el directorio seguro de MySQL para la carga con LOAD DATA INFILE.

- Desde Windows 11

sftp azureuser@<IP_PUBLICA>

put sales_data.csv

exit

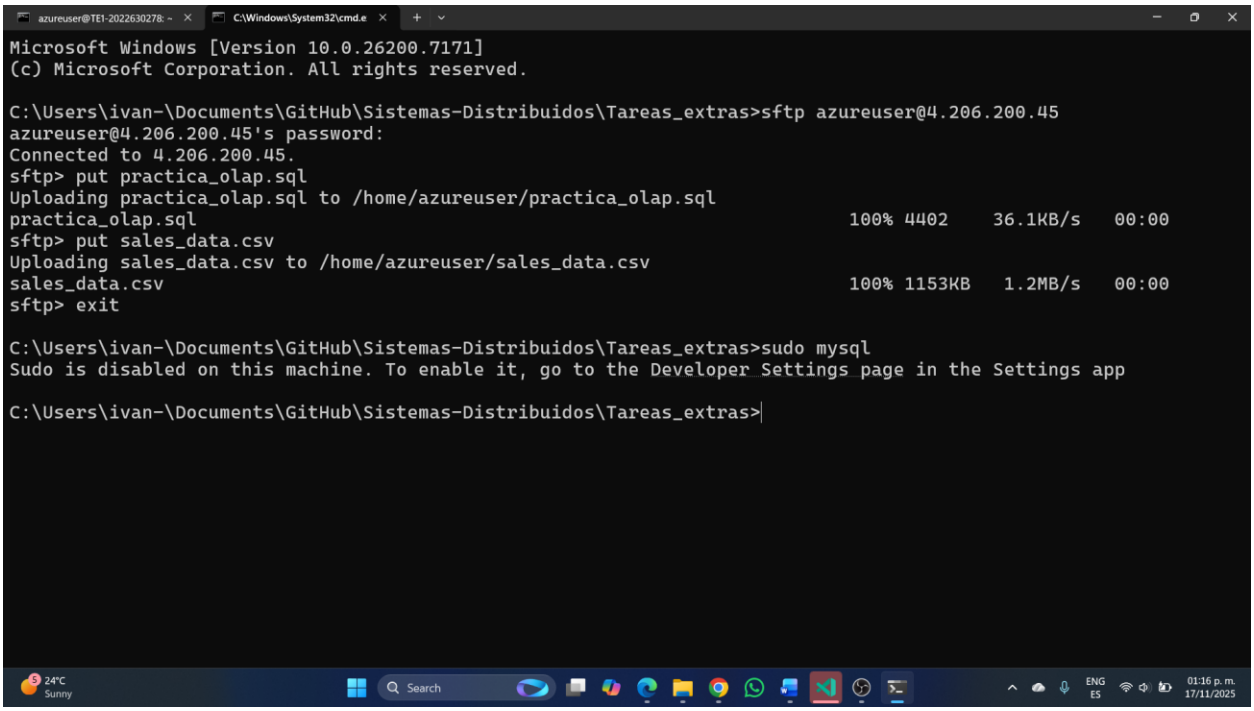


Imagen 9.1: Pasar archivos a la VM.

- En la VM

sudo mv /home/azureuser/sales_data.csv /var/lib/mysql-files/

sudo chmod 644 /var/lib/mysql-files/sales_data.csv

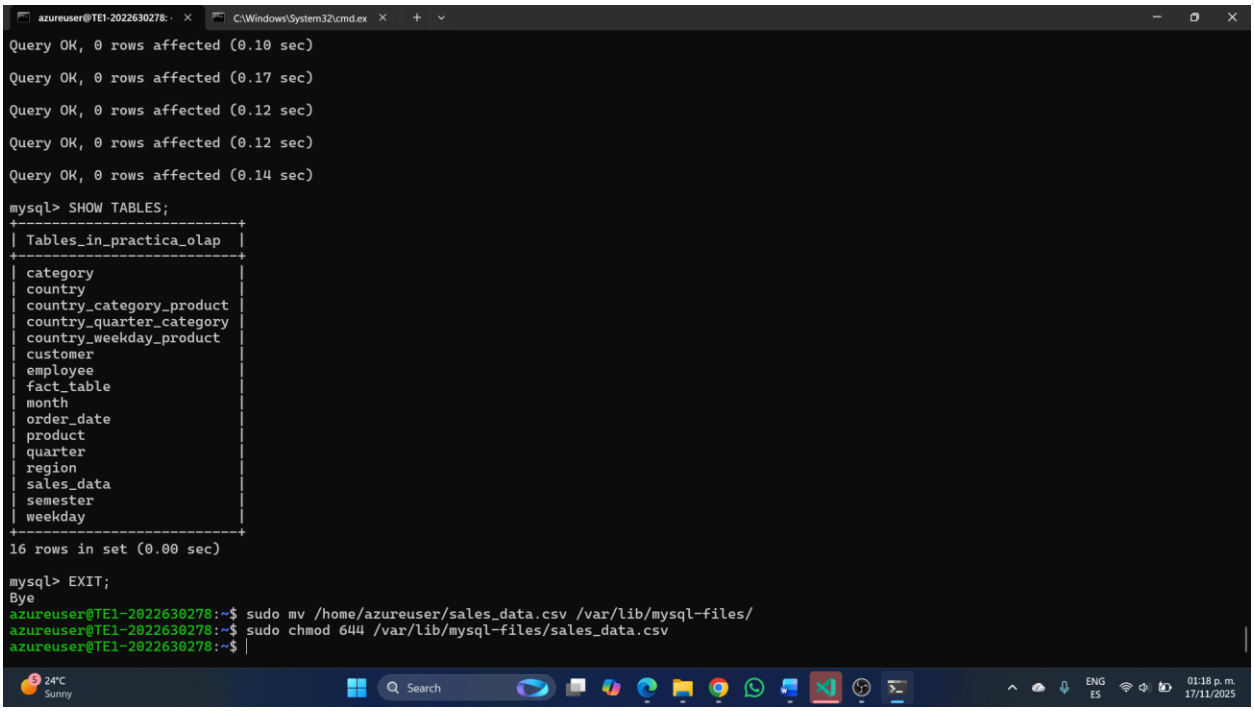


Imagen 9.2: Mostrar tablas de la VM.

Se realizó la carga a la tabla sales_data (omitiendo la línea de encabezados):

sudo mysql

USE practica_olap;

LOAD DATA INFILE '/var/lib/mysql-files/sales_data.csv'

INTO TABLE sales_data

FIELDS TERMINATED BY ',' ENCLOSED BY ""

IGNORE 1 LINES

(sales, order_date, product, customer, country, region, employee, category, weekday, month, quarter, semester);

SELECT COUNT(*) AS filas_en_sales_data FROM sales_data;

EXIT;

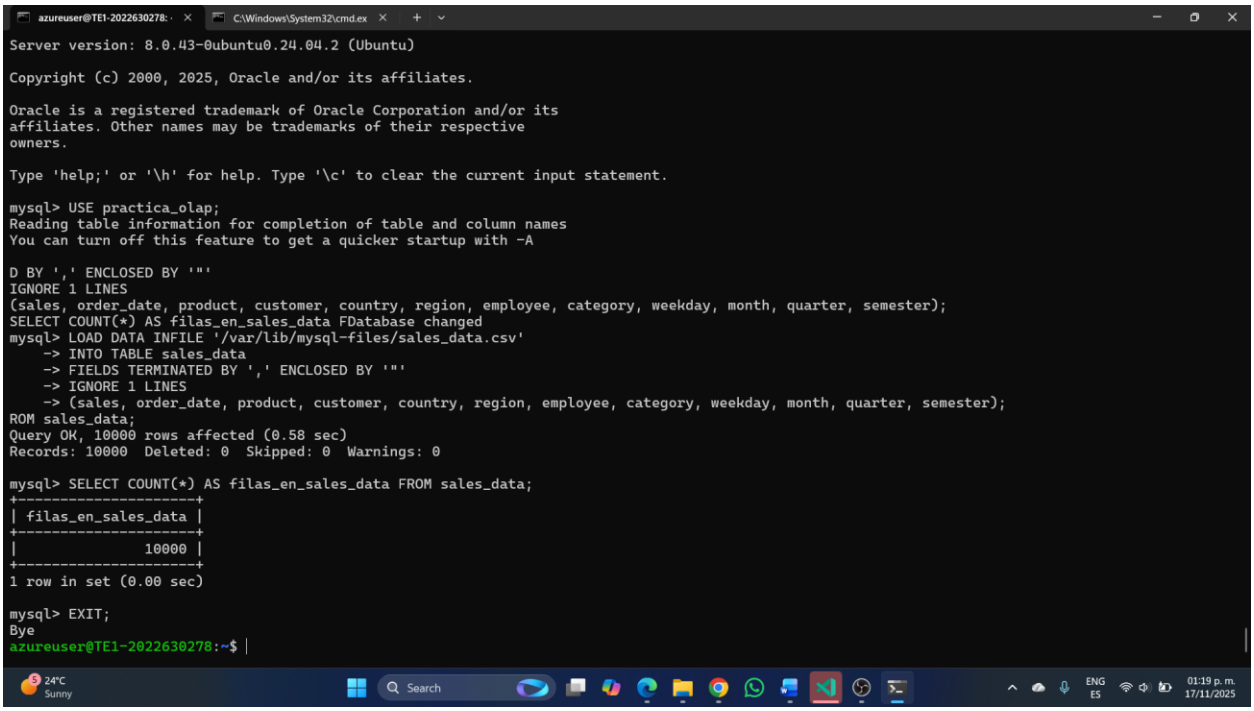


Imagen 10: Resultado de la carga (SELECT COUNT(*) FROM sales_data mostrando la cantidad de filas cargadas).

5.5 Población de tablas de dimensiones y fact_table

Se poblaron las tablas de dimensiones a partir de sales_data y, posteriormente, la fact_table con los IDs correspondientes.

- Dentro de MySQL (USE practica_olap;)

INSERT INTO region (region)

SELECT DISTINCT region FROM sales_data WHERE region IS NOT NULL;

INSERT INTO country (country, id_region)

SELECT DISTINCT sd.country, r.id_region

FROM sales_data sd

JOIN region r ON r.region = sd.region;

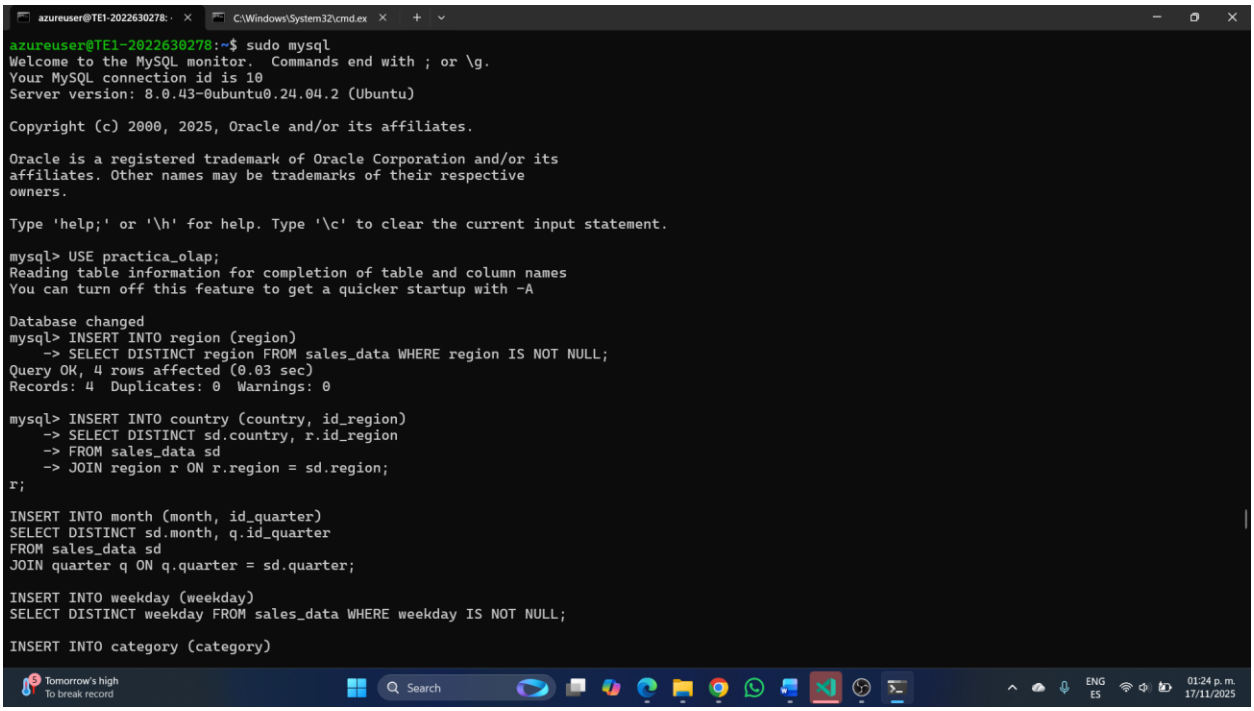
```
INSERT INTO customer (customer)
SELECT DISTINCT customer FROM sales_data WHERE customer IS NOT NULL;

INSERT INTO employee (employee)
SELECT DISTINCT employee FROM sales_data WHERE employee IS NOT NULL;

INSERT INTO semester (semester)
SELECT DISTINCT semester FROM sales_data WHERE semester IS NOT NULL;

INSERT INTO quarter (quarter, id_semester)
SELECT DISTINCT sd.quarter, s.id_semester
FROM sales_data sd
JOIN semester s ON s.semester = sd.semester;

INSERT INTO month (month, id_quarter)
SELECT DISTINCT sd.month, q.id_quarter
FROM sales_data sd
JOIN quarter q ON q.quarter = sd.quarter;

INSERT INTO weekday (weekday)
SELECT DISTINCT weekday FROM sales_data WHERE weekday IS NOT NULL;
```

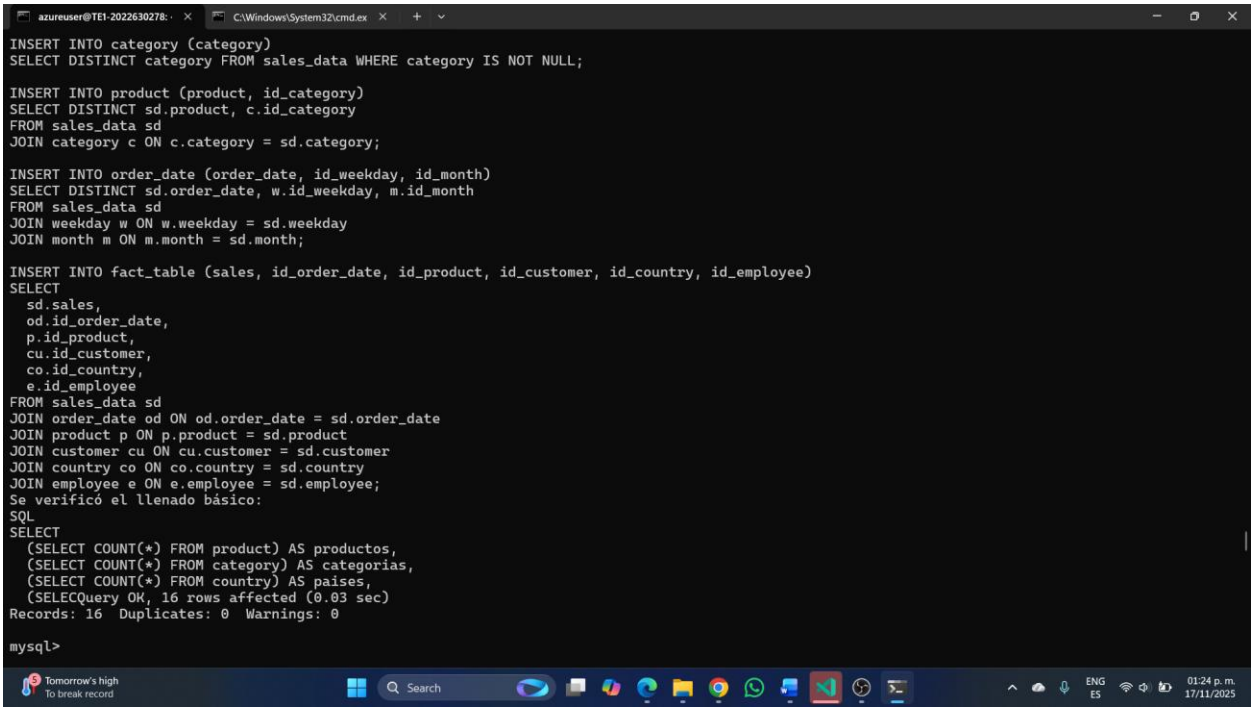
```
INSERT INTO category (category)
SELECT DISTINCT category FROM sales_data WHERE category IS NOT NULL;

INSERT INTO product (product, id_category)
SELECT DISTINCT sd.product, c.id_category
FROM sales_data sd
JOIN category c ON c.category = sd.category;

INSERT INTO order_date (order_date, id_weekday, id_month)
SELECT DISTINCT sd.order_date, w.id_weekday, m.id_month
FROM sales_data sd
JOIN weekday w ON w.weekday = sd.weekday
JOIN month m ON m.month = sd.month;

INSERT INTO fact_table (sales, id_order_date, id_product, id_customer, id_country, id_employee)
SELECT
  sd.sales,
  od.id_order_date,
  p.id_product,
  cu.id_customer,
  co.id_country,
  e.id_employee
```

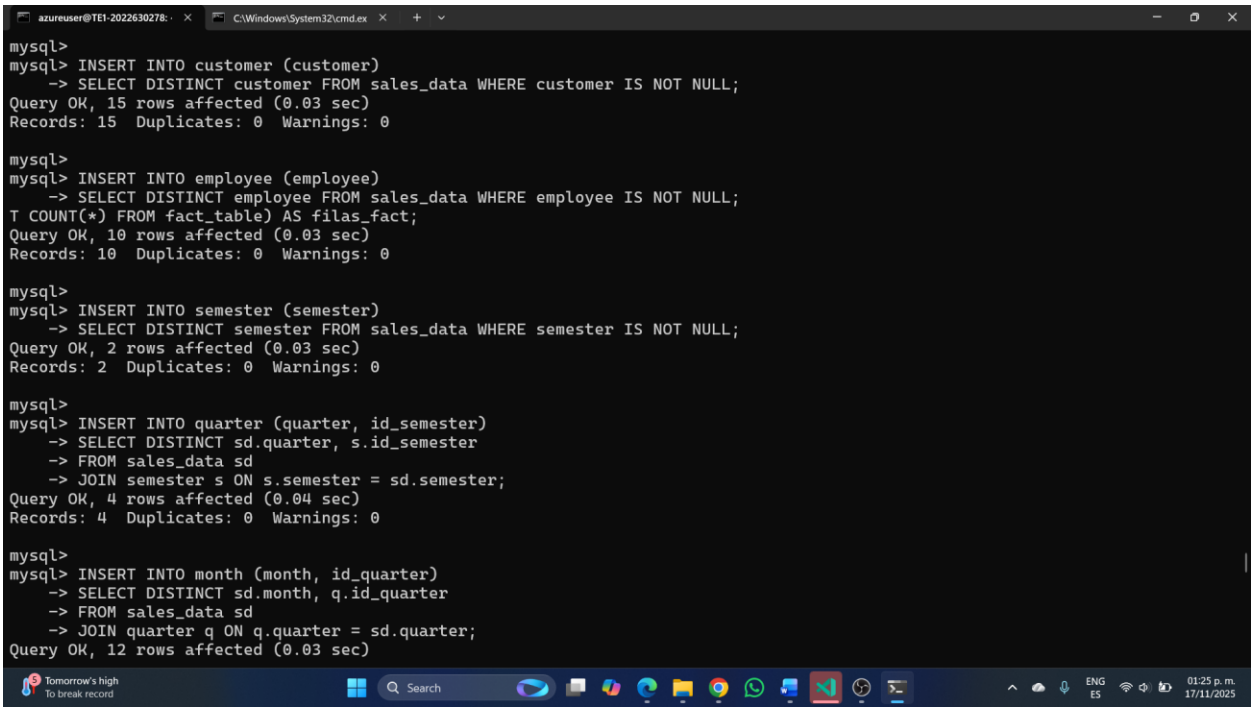
```
FROM sales_data sd
JOIN order_date od ON od.order_date = sd.order_date
JOIN product p ON p.product = sd.product
JOIN customer cu ON cu.customer = sd.customer
JOIN country co ON co.country = sd.country
JOIN employee e ON e.employee = sd.employee;
```

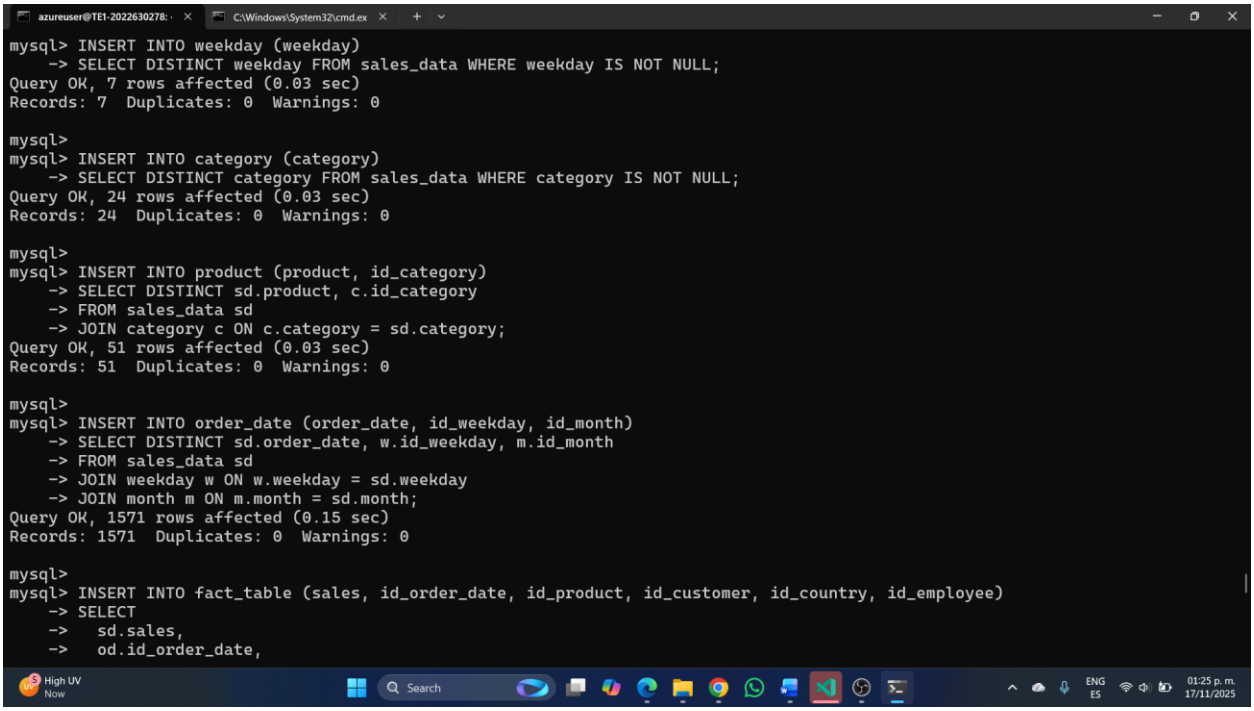
Se verificó el llenado básico:

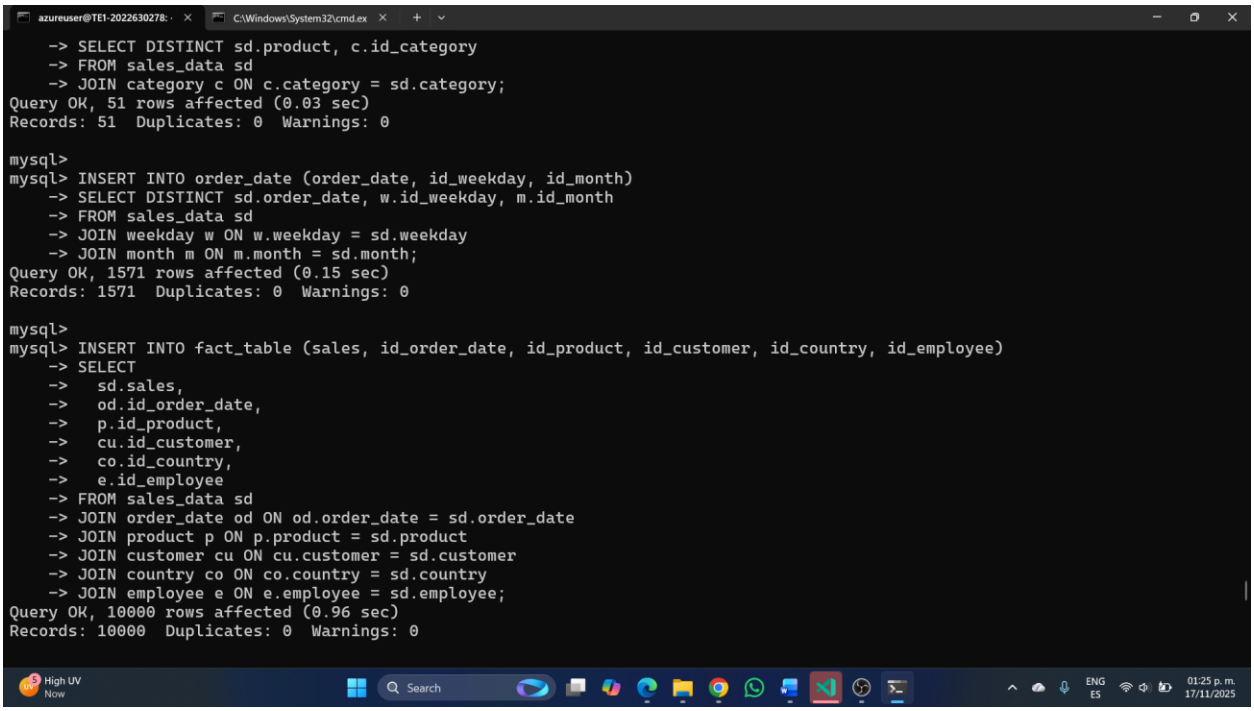
```
SELECT
  (SELECT COUNT(*) FROM product) AS productos,
  (SELECT COUNT(*) FROM category) AS categorias,
  (SELECT COUNT(*) FROM country) AS paises,
  (SELECT COUNT(*) FROM fact_table) AS filas_fact;
```


azureuser@TE1-2022630278:~$ sudo mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.43-0ubuntu0.24.04.2 (Ubuntu)
Copyright (c) 2000, 2025, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> USE practica_olap;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
mysql> INSERT INTO region (region)
-> SELECT DISTINCT region FROM sales_data WHERE region IS NOT NULL;
Query OK, 4 rows affected (0.03 sec)
Records: 4 Duplicates: 0 Warnings: 0
mysql> INSERT INTO country (country, id_region)
-> SELECT DISTINCT sd.country, r.id_region
-> FROM sales_data sd
-> JOIN region r ON r.region = sd.region;
r;
INSERT INTO month (month, id_quarter)
SELECT DISTINCT sd.month, q.id_quarter
FROM sales_data sd
JOIN quarter q ON q.quarter = sd.quarter;
INSERT INTO weekday (weekday)
SELECT DISTINCT weekday FROM sales_data WHERE weekday IS NOT NULL;
INSERT INTO category (category)


INSERT INTO category (category)
SELECT DISTINCT category FROM sales_data WHERE category IS NOT NULL;
INSERT INTO product (product, id_category)
SELECT DISTINCT sd.product, c.id_category
FROM sales_data sd
JOIN category c ON c.category = sd.category;
INSERT INTO order_date (order_date, id_weekday, id_month)
SELECT DISTINCT sd.order_date, w.id_weekday, m.id_month
FROM sales_data sd
JOIN weekday w ON w.weekday = sd.weekday
JOIN month m ON m.month = sd.month;
INSERT INTO fact_table (sales, id_order_date, id_product, id_customer, id_country, id_employee)
SELECT
sd.sales,
od.id_order_date,
p.id_product,
cu.id_customer,
co.id_country,
e.id_employee
FROM sales_data sd
JOIN order_date od ON od.order_date = sd.order_date
JOIN product p ON p.product = sd.product
JOIN customer cu ON cu.customer = sd.customer
JOIN country co ON co.country = sd.country
JOIN employee e ON e.employee = sd.employee;
Se verificó el llenado básico:
SQL
SELECT
(SELECT COUNT(*) FROM product) AS productos,
(SELECT COUNT(*) FROM category) AS categorias,
(SELECT COUNT(*) FROM country) AS paises,
(SELECQuery OK, 16 rows affected (0.03 sec)
Records: 16 Duplicates: 0 Warnings: 0
mysql>


azureuser@TE1-2022630278:
C:\Windows\System32\cmd.ex
mysql>
mysql> INSERT INTO customer (customer)
-> SELECT DISTINCT customer FROM sales_data WHERE customer IS NOT NULL;
Query OK, 15 rows affected (0.03 sec)
Records: 15 Duplicates: 0 Warnings: 0
mysql>
mysql> INSERT INTO employee (employee)
-> SELECT DISTINCT employee FROM sales_data WHERE employee IS NOT NULL;
T COUNT(*) FROM fact_table) AS filas_fact;
Query OK, 10 rows affected (0.03 sec)
Records: 10 Duplicates: 0 Warnings: 0
mysql>
mysql> INSERT INTO semester (semester)
-> SELECT DISTINCT semester FROM sales_data WHERE semester IS NOT NULL;
Query OK, 2 rows affected (0.03 sec)
Records: 2 Duplicates: 0 Warnings: 0
mysql>
mysql> INSERT INTO quarter (quarter, id_semester)
-> SELECT DISTINCT sd.quarter, s.id_semester
-> FROM sales_data sd
-> JOIN semester s ON s.semester = sd.semester;
Query OK, 4 rows affected (0.04 sec)
Records: 4 Duplicates: 0 Warnings: 0
mysql>
mysql> INSERT INTO month (month, id_quarter)
-> SELECT DISTINCT sd.month, q.id_quarter
-> FROM sales_data sd
-> JOIN quarter q ON q.quarter = sd.quarter;
Query OK, 12 rows affected (0.03 sec)
Tomorrow's high
To break record
Search
ENG
ES
01:25 p. m.
17/11/2025


azureuser@TE1-2022630278:
C:\Windows\System32\cmd.ex
mysql> INSERT INTO weekday (weekday)
-> SELECT DISTINCT weekday FROM sales_data WHERE weekday IS NOT NULL;
Query OK, 7 rows affected (0.03 sec)
Records: 7 Duplicates: 0 Warnings: 0
mysql>
mysql> INSERT INTO category (category)
-> SELECT DISTINCT category FROM sales_data WHERE category IS NOT NULL;
Query OK, 24 rows affected (0.03 sec)
Records: 24 Duplicates: 0 Warnings: 0
mysql>
mysql> INSERT INTO product (product, id_category)
-> SELECT DISTINCT sd.product, c.id_category
-> FROM sales_data sd
-> JOIN category c ON c.category = sd.category;
Query OK, 51 rows affected (0.03 sec)
Records: 51 Duplicates: 0 Warnings: 0
mysql>
mysql> INSERT INTO order_date (order_date, id_weekday, id_month)
-> SELECT DISTINCT sd.order_date, w.id_weekday, m.id_month
-> FROM sales_data sd
-> JOIN weekday w ON w.weekday = sd.weekday
-> JOIN month m ON m.month = sd.month;
Query OK, 1571 rows affected (0.15 sec)
Records: 1571 Duplicates: 0 Warnings: 0
mysql>
mysql> INSERT INTO fact_table (sales, id_order_date, id_product, id_customer, id_country, id_employee)
-> SELECT
-> sd.sales,
-> od.id_order_date,
High UV
Now
Search
ENG
ES
01:25 p. m.
17/11/2025



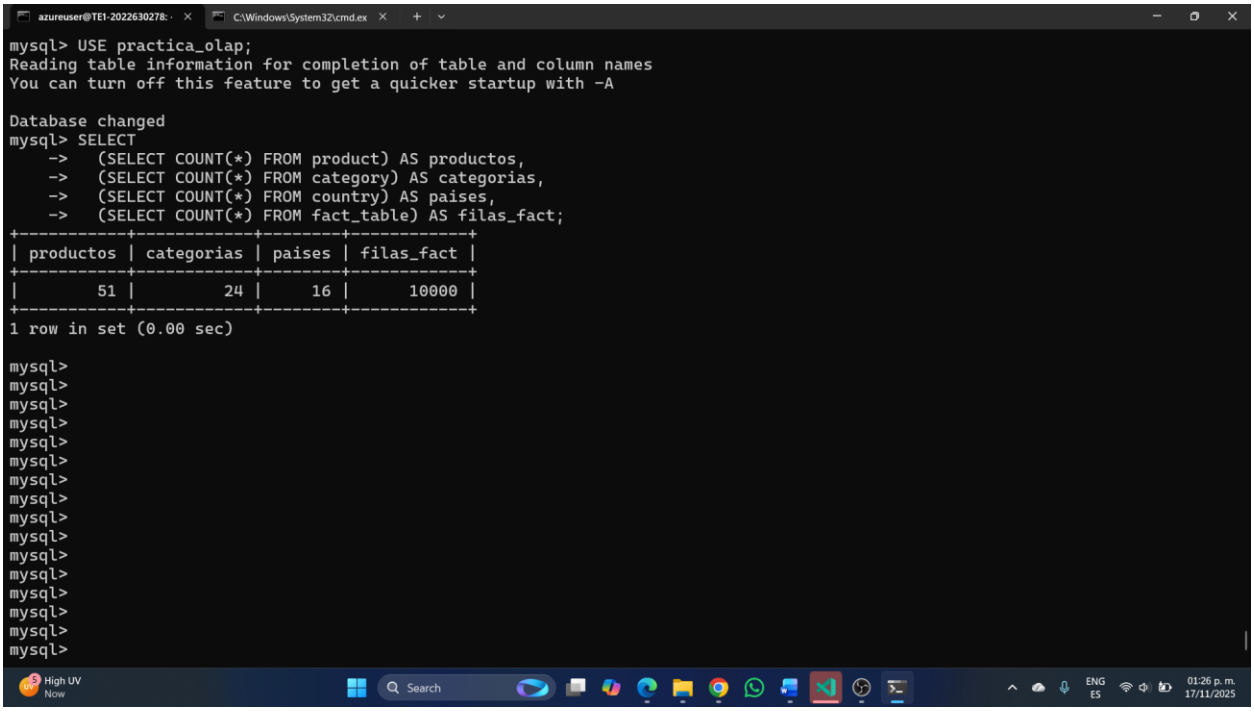


Imagen 11: Conteos de dimensiones y fact_table (SELECT con totales de productos, categorías, países y filas_fact).

5.6 Generación del archivo country_category_product.csv

Se generó el archivo CSV de entrada para Hadoop mediante SELECT … INTO OUTFILE en el directorio seguro de MySQL y se copió al HOME del usuario para facilitar su uso con Hadoop.

- En MySQL

```
USE practica_olap;

SELECT a.id_country, b.id_category, a.id_product, a.sales

INTO OUTFILE '/var/lib/mysql-files/country_category_product.csv'

FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

FROM fact_table a

JOIN product b ON a.id_product = b.id_product;
```

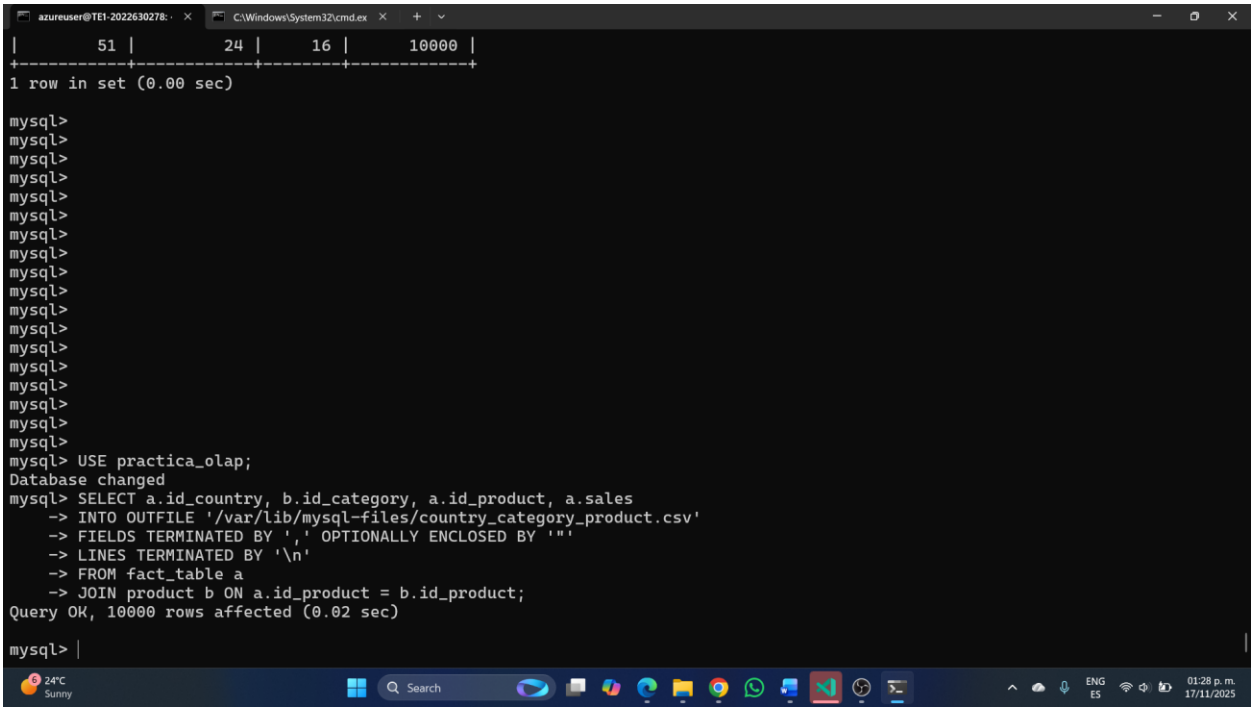


Imagen 11.1: Consulta a la base de datos.

- En la VM:

cp /var/lib/mysql-files/country_category_product.csv /home/azureuser/

ls -lh /var/lib/mysql-files/country_category_product.csv

head -n 5 /home/azureuser/country_category_product.csv

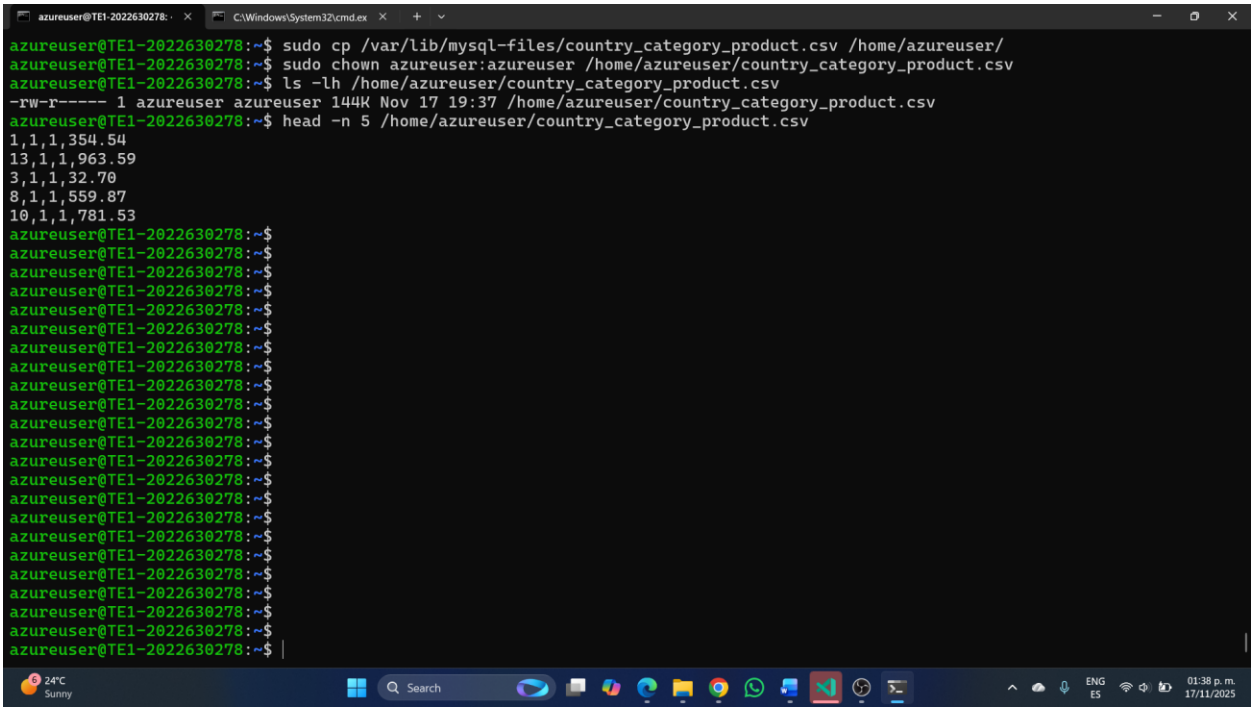


Imagen 12: Evidencia del archivo country_category_product.csv (ls -lh y primeras líneas con head).

5.7 Instalación y configuración de Apache Hadoop

Se verificó la arquitectura de la VM y se descargó la distribución adecuada de Hadoop 3.4.0. Si la arquitectura fue x86_64, se utilizó el tarball genérico; si fue ARM, se usó el aarch64 indicado en la guía.

uname -m

cd /home/azureuser

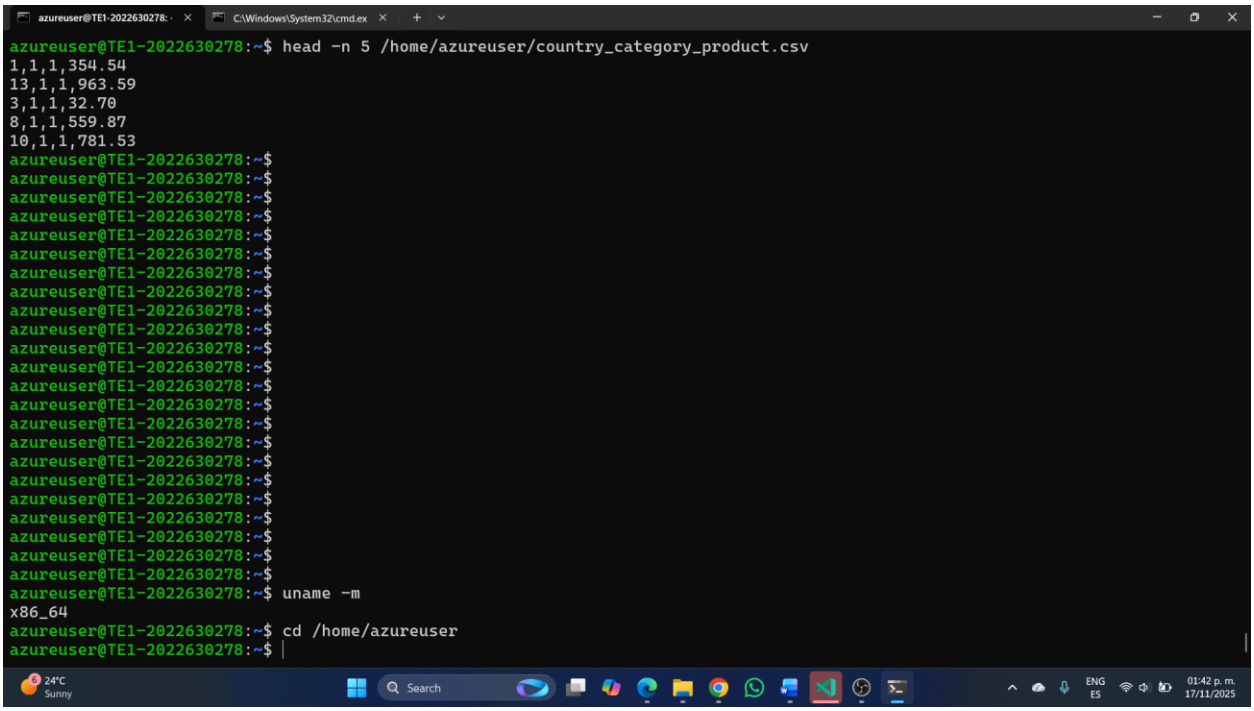


Imagen 12.1: Arquitectura x86_64.

- x86_64 (típico en Azure)

wget https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz

tar -xzf hadoop-3.4.0.tar.gz

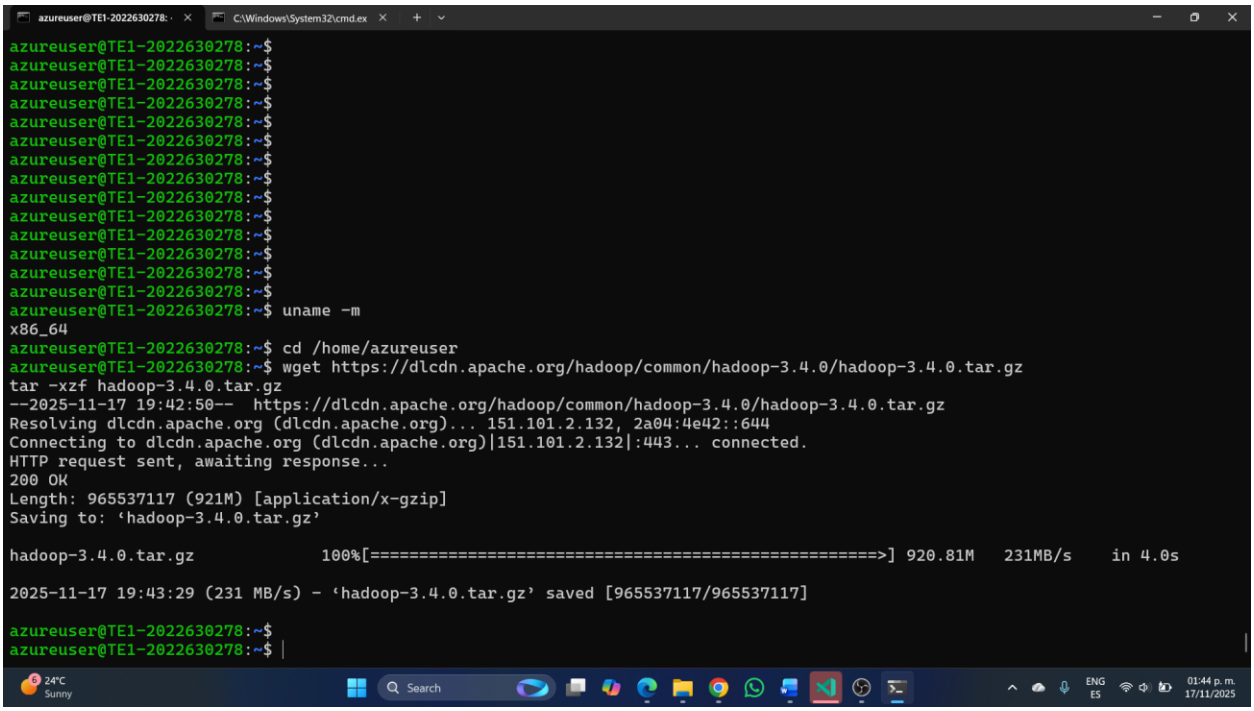


Imagen 12. 2: Instalar hadoop

Se instaló Java según lo requerido por Hadoop (OpenJDK 16 o, en su defecto, 17), y se configuró JAVA_HOME en hadoop-env.sh.

sudo apt -y install openjdk-16-jdk-headless || sudo apt -y install openjdk-17-jdk-headless

readlink -f $(which java)

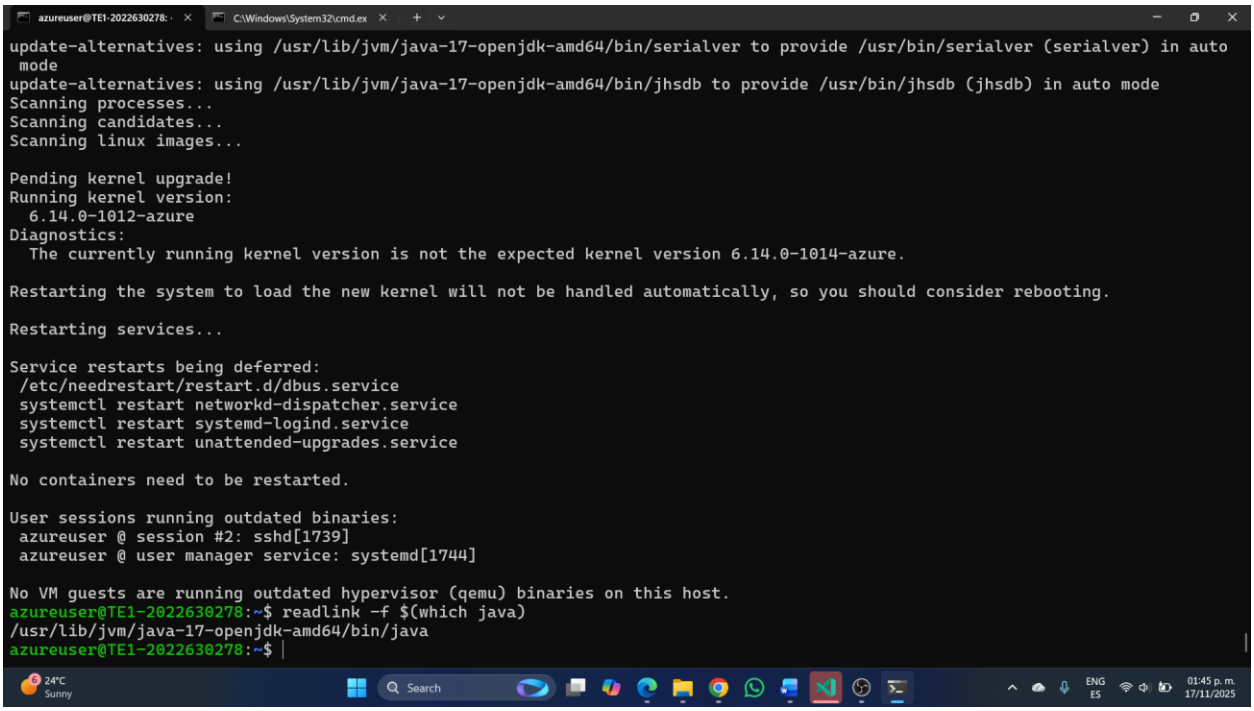


Imagen 12.3: Mostrar ubicación de java

Ejemplo típico: /usr/lib/jvm/java-17-openjdk-amd64/bin/java

nano /home/azureuser/hadoop-3.4.0/etc/hadoop/hadoop-env.sh

Se actualizó la línea: export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64

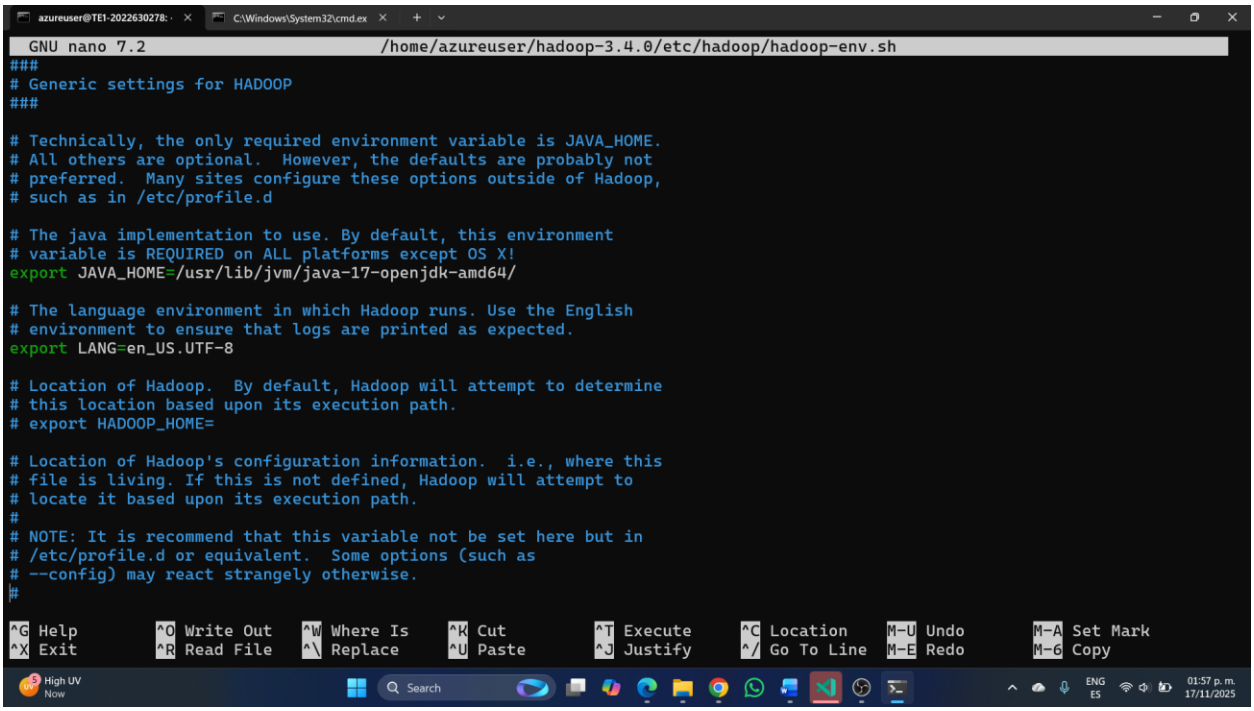


Imagen 12.4: Editar JAVA_HOME

Se exportaron variables de entorno para la sesión y para futuras sesiones:

echo 'export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64' >> ~/.bashrc

echo 'export HADOOP_HOME=/home/azureuser/hadoop-3.4.0' >> ~/.bashrc

echo 'export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin' >> ~/.bashrc

source ~/.bashrc

hadoop version

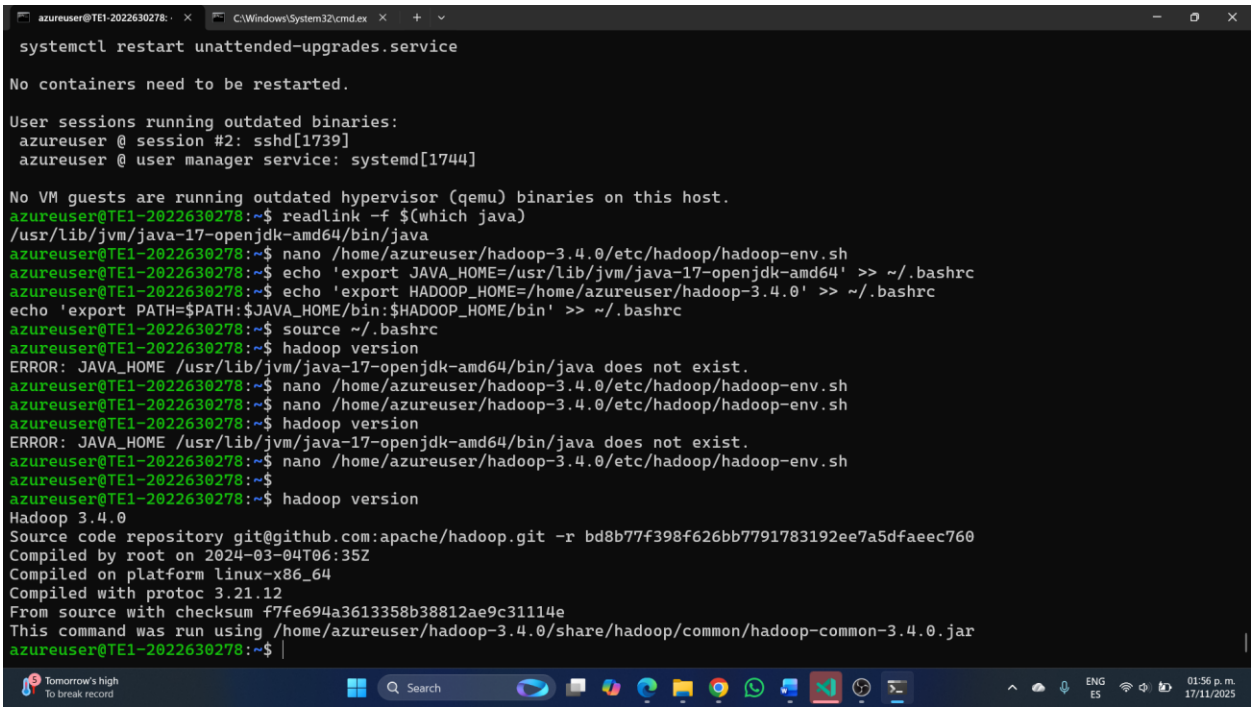


Imagen 13: Configuración de Hadoop y Java (salida de hadoop version y valor de JAVA_HOME; captura de hadoop-env.sh con export JAVA_HOME).

5.8 Implementación del job MapReduce

Se creó el directorio de trabajo y se implementaron las clases Java del Mapper, Reducer y Driver, conforme a lo visto en clase. Se compiló y se construyó el JAR de la aplicación.

mkdir -p /home/azureuser/prueba

cd /home/azureuser/prueba

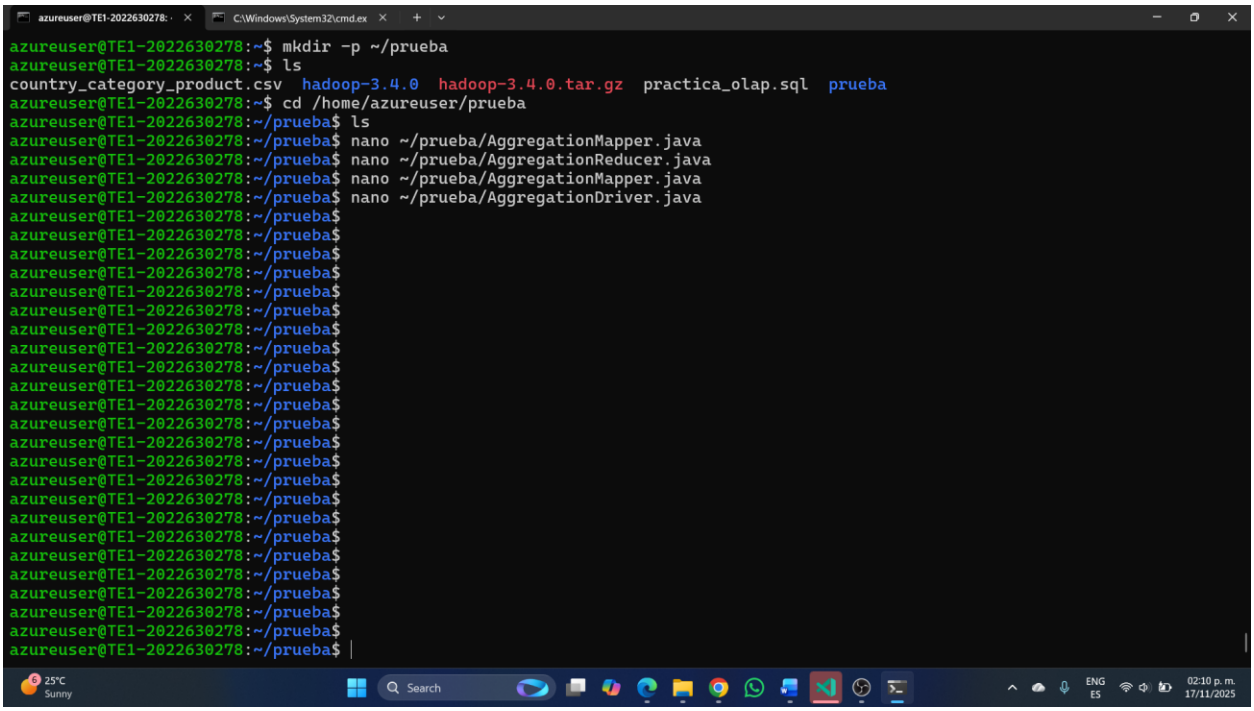


Imagen 13.1: Creación de archivos java

AggregationMapper.java

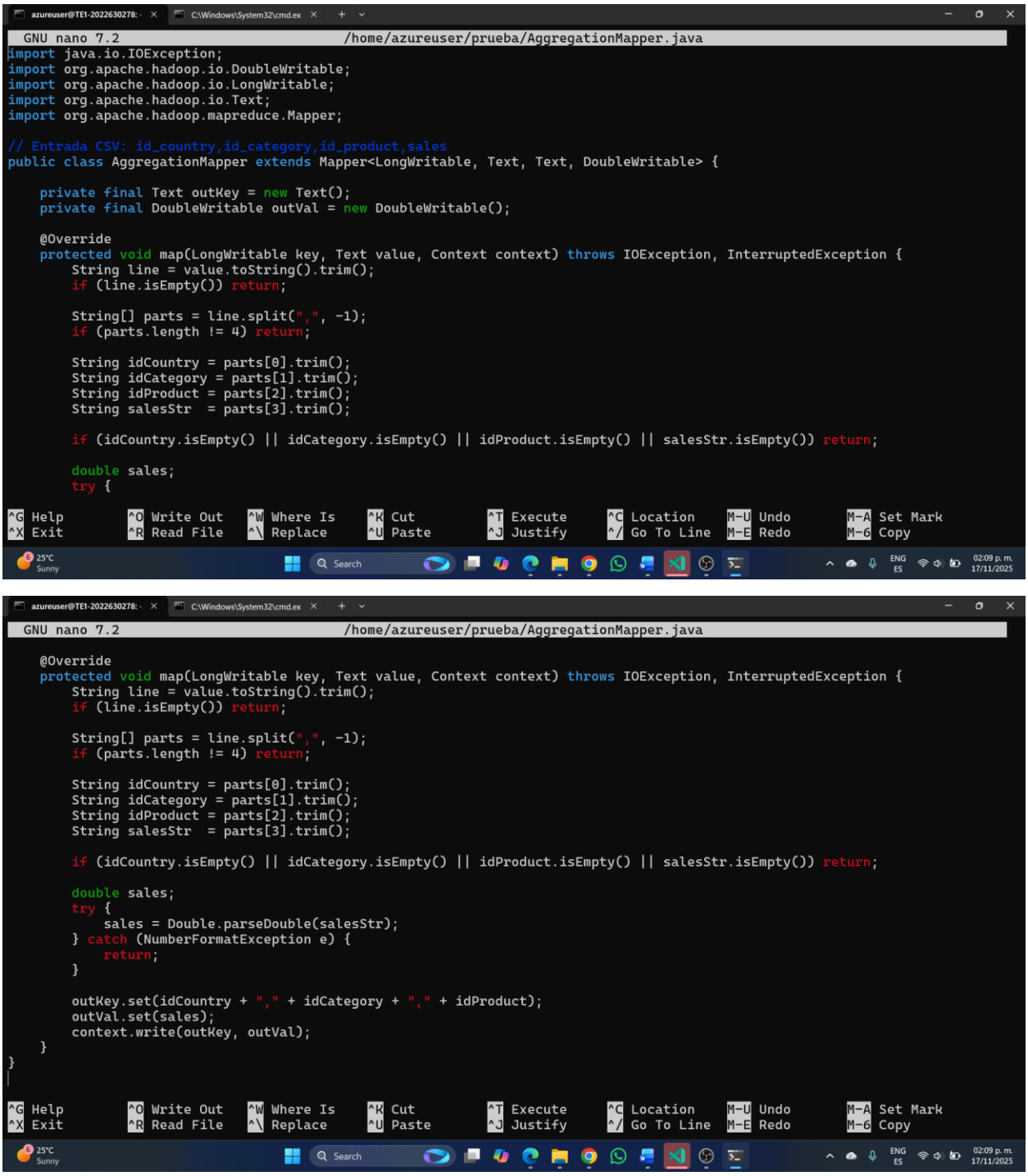


Imagen 13.1: AggregationMapper.java

AggregationReducer.java

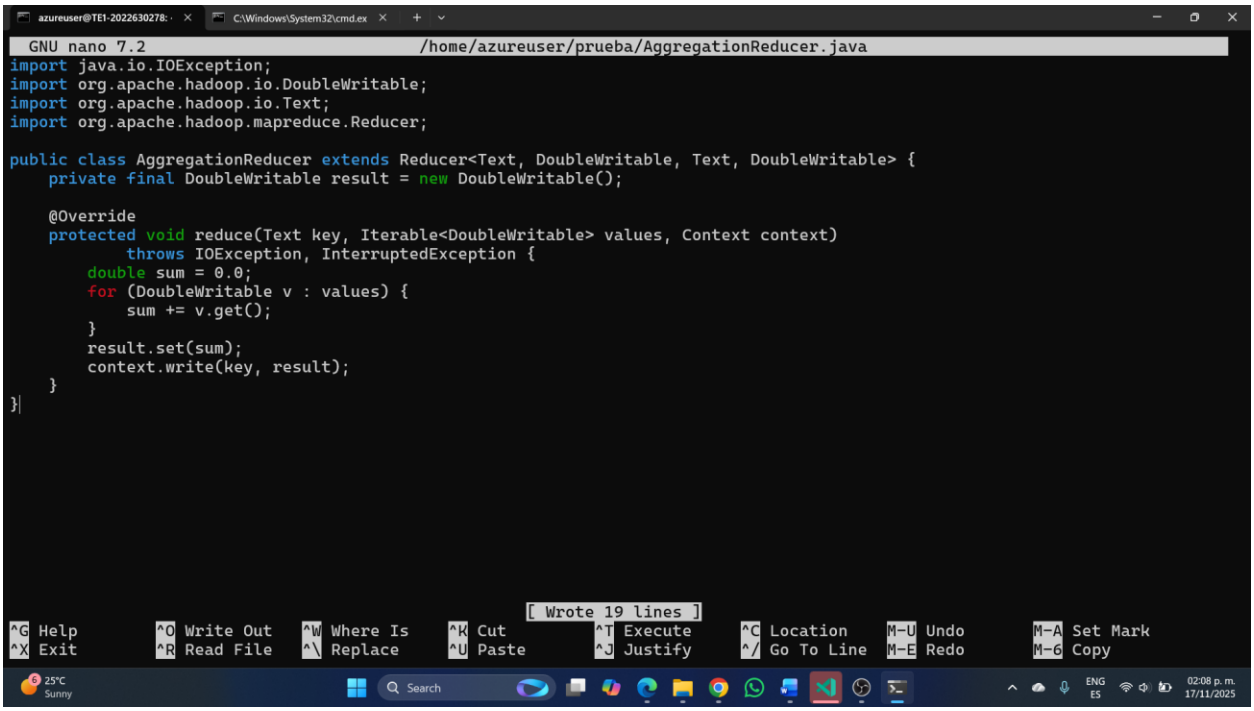


Imagen 13. 2: AggregationReducer.java

AggregationDriver.java

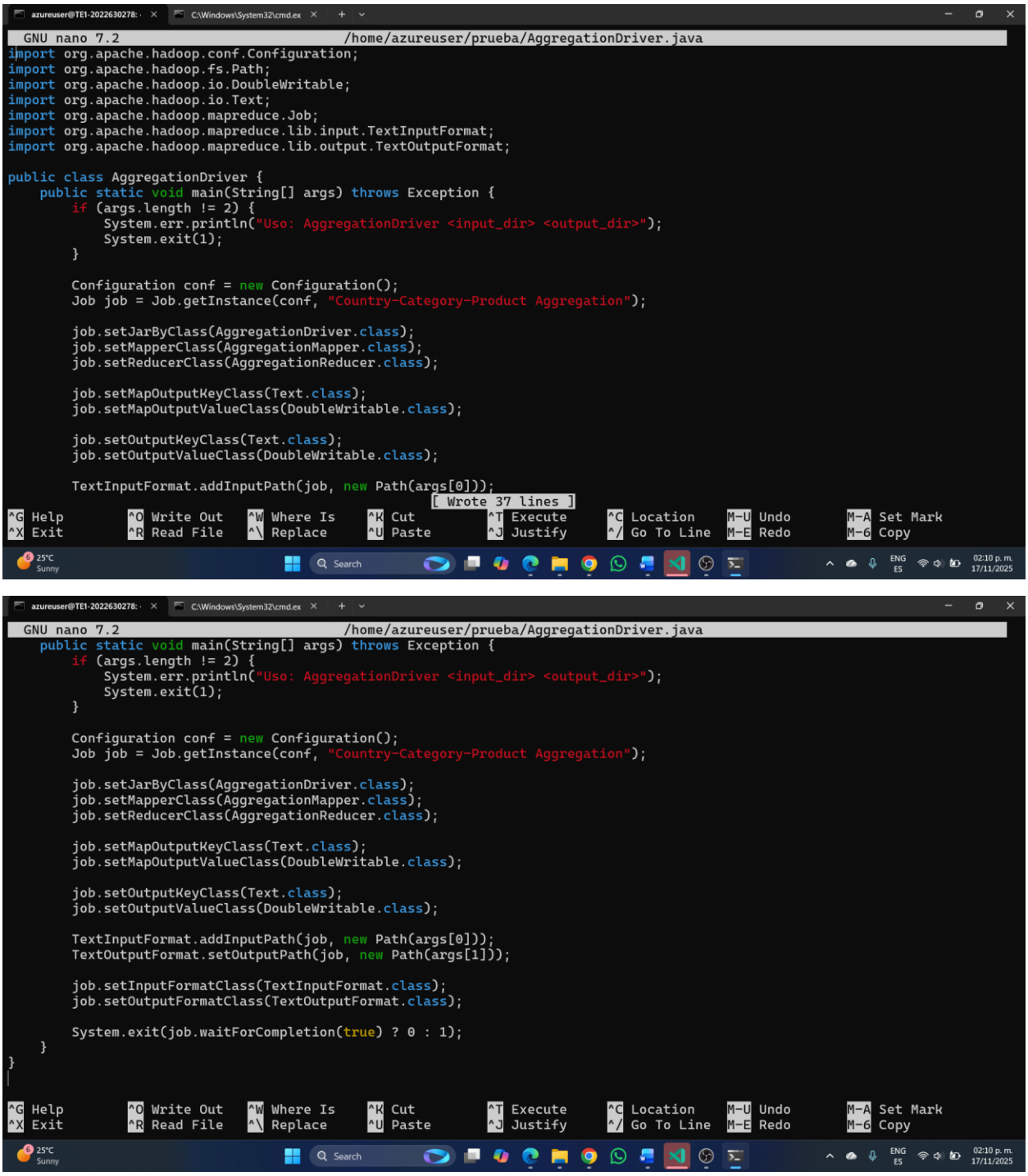


Imagen 13.3: AggregationDriver.java

mkdir -p build

javac -classpath "$(hadoop classpath)" -d build *.java

jar -cvf Aggregation.jar -C build .

ls -lh Aggregation.jar

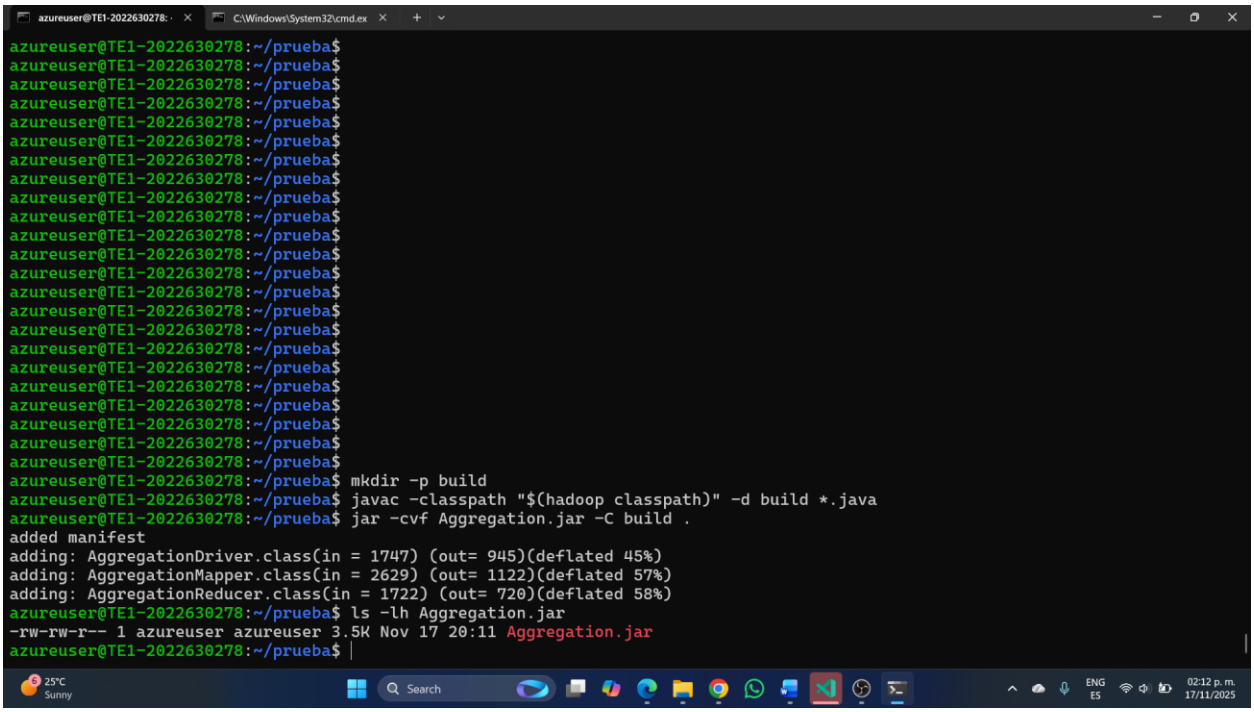


Imagen 14: Compilación y empaquetado exitosos (salida de javac/jar y listado mostrando Aggregation.jar).

5.9 Ejecución del job MapReduce y verificación del resultado

Se preparó el directorio de entrada con el CSV generado desde MySQL y se ejecutó el job en modo local (standalone). Se visualizó la salida en part-r-00000 y se normalizó el separador a coma para su posterior carga.

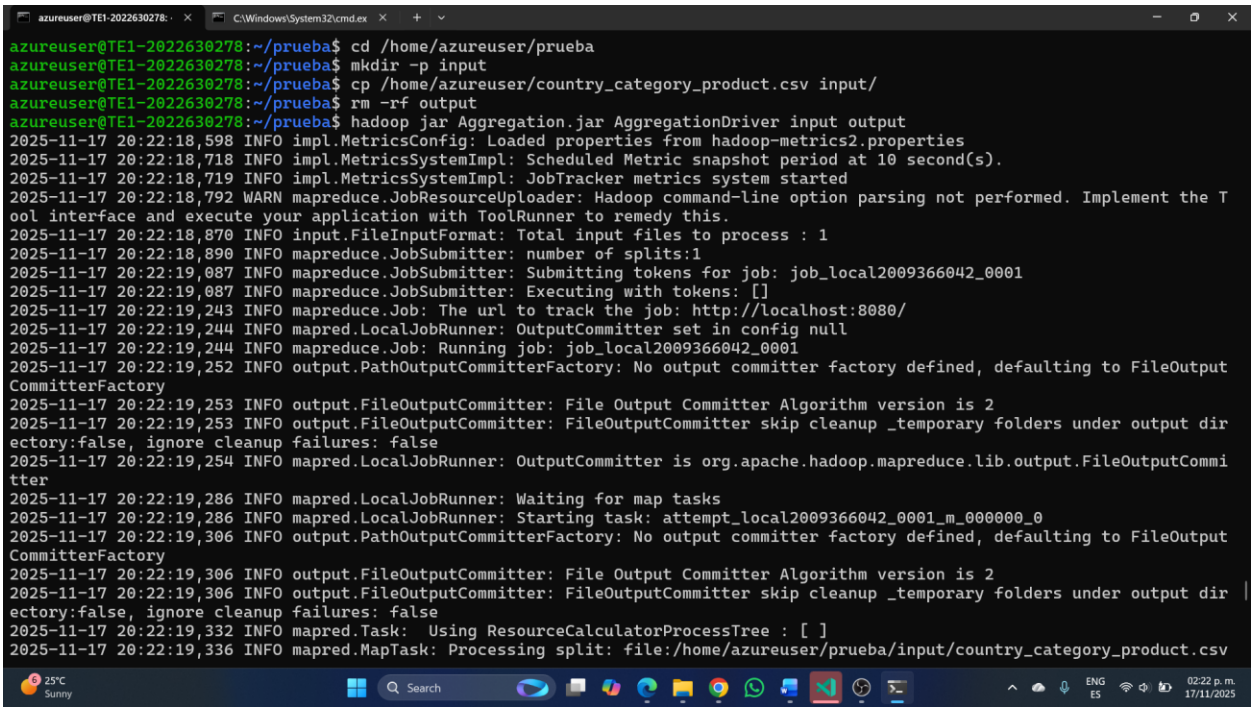
cd /home/azureuser/prueba

mkdir -p input

cp /home/azureuser/country_category_product.csv input/

rm -rf output

hadoop jar Aggregation.jar AggregationDriver input output



ls -lh output

head -n 10 output/part-r-00000

Se convirtió el tabulador a coma:

sed -i 's/\t/,/g' output/part-r-00000

head -n 10 output/part-r-00000

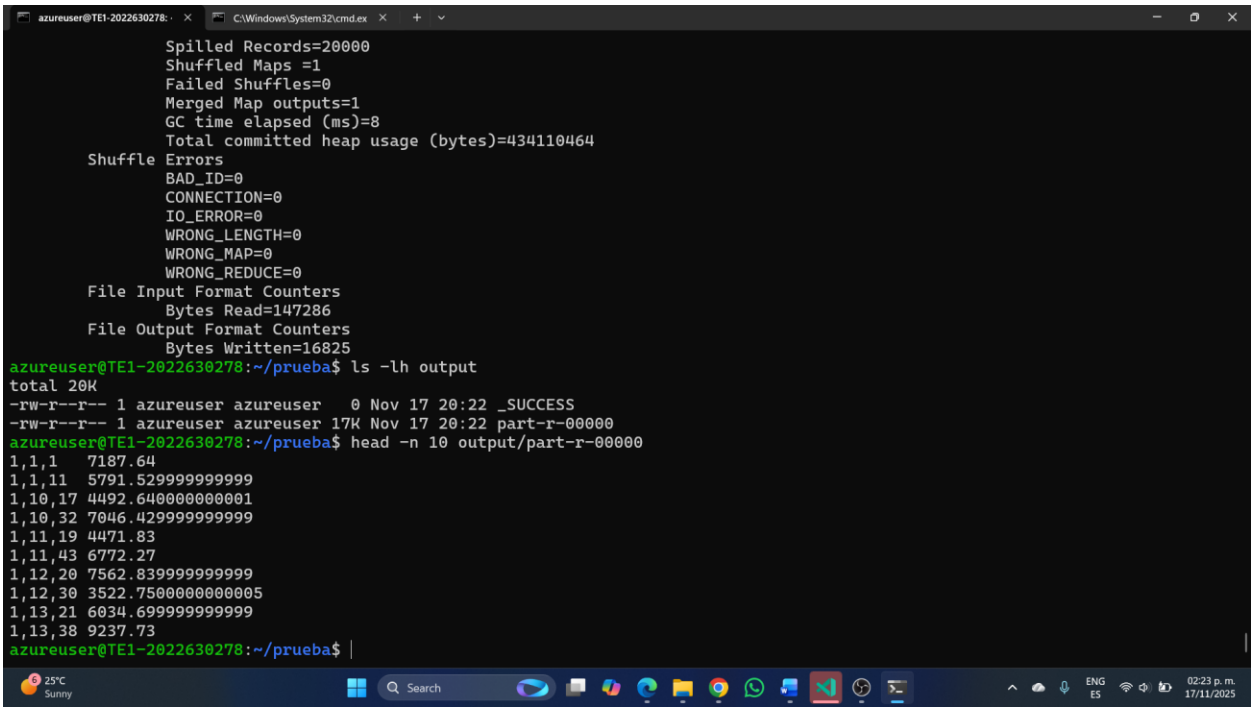


Imagen 15: Ejecución del job (salida con counters/resumen y existencia de output/part-r-00000).

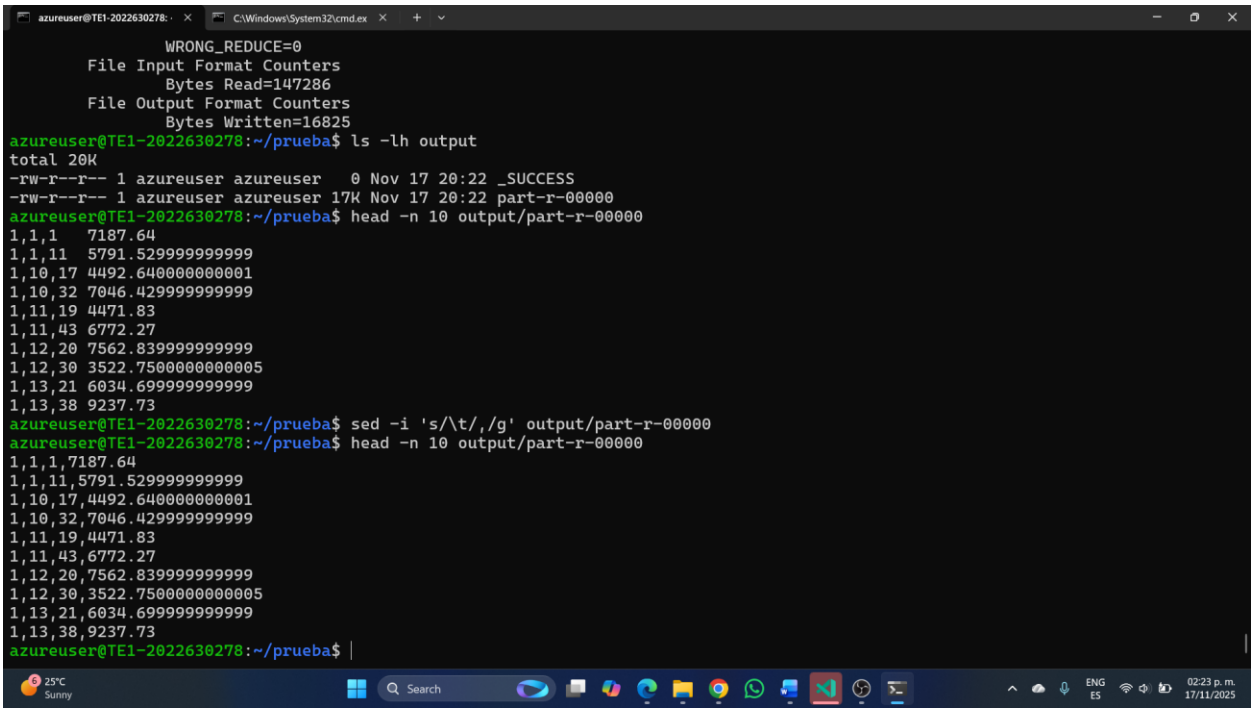


Imagen 16: Contenido de part-r-00000 antes y después de sed (dos capturas mostrando el cambio de tabulador a coma).

5.10 Carga del resultado a la tabla agregada country_category_product

Se colocó el archivo final en el directorio seguro de MySQL y se cargó a la tabla agregada. Se verificó la cantidad de filas.

sudo cp /home/azureuser/prueba/output/part-r-00000 /var/lib/mysql-files/ccp_part.csv

sudo chown mysql:mysql /var/lib/mysql-files/ccp_part.csv

sudo chmod 644 /var/lib/mysql-files/ccp_part.csv

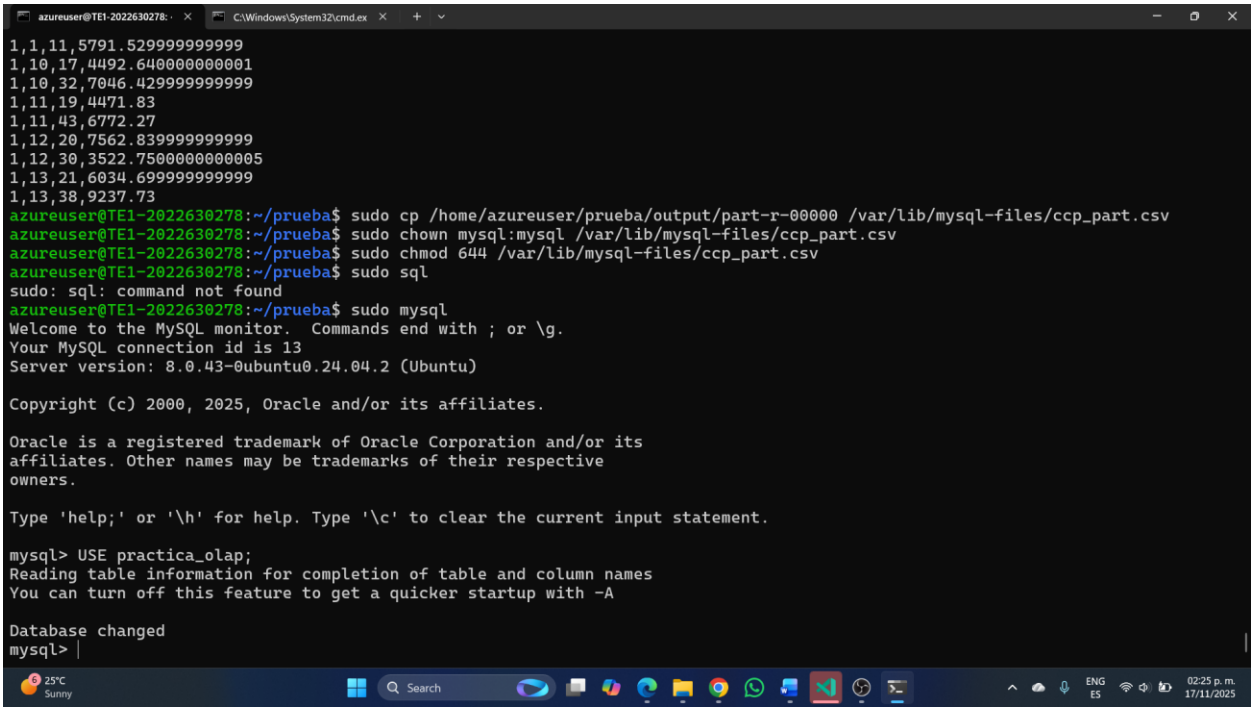


Imagen 16: Carga del resultado a la tabla agregada

- En MySQL

USE practica_olap;

TRUNCATE TABLE country_category_product;

LOAD DATA INFILE '/var/lib/mysql-files/ccp_part.csv'

INTO TABLE country_category_product

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

(id_country, id_category, id_product, sales);

SELECT COUNT(*) AS filas_en_country_category_product FROM country_category_product;

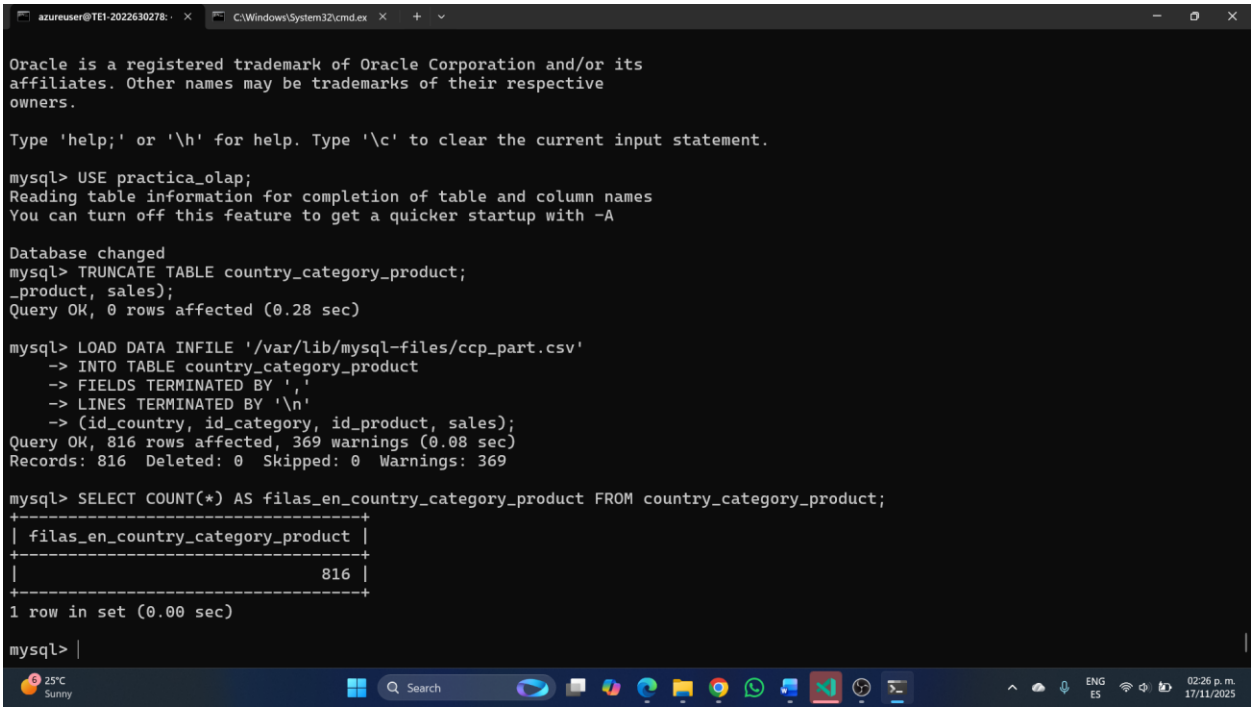


Imagen 17: Carga a la tabla agregada y verificación (SELECT COUNT(*) FROM country_category_product mostrando el total de filas).

6 Consultas OLAP (usando la tabla agregada)

En esta sección se ejecutan los tres cubos solicitados, exclusivamente sobre la tabla agregada country_category_product. Cada consulta muestra la suma de ventas (sales) y se acompaña de su evidencia. Toma la captura de la “primera pantalla” de resultados en MySQL para cada caso.

Sugerencia: para que la salida sea legible, ordena por el total y limita las primeras filas si tu terminal imprime demasiadas líneas. Se usa ROUND(...,2) para mostrar dos decimales.

6.1 Acumulado de sales por country

Consulta que suma las ventas por país, aprovechando el join entre la tabla agregada y la dimensión country.

```
SELECT c.country,
    ROUND(SUM(ccp.sales), 2) AS total_sales
FROM country_category_product ccp
JOIN country c
  ON c.id_country = ccp.id_country
GROUP BY c.country
ORDER BY total_sales DESC;
```

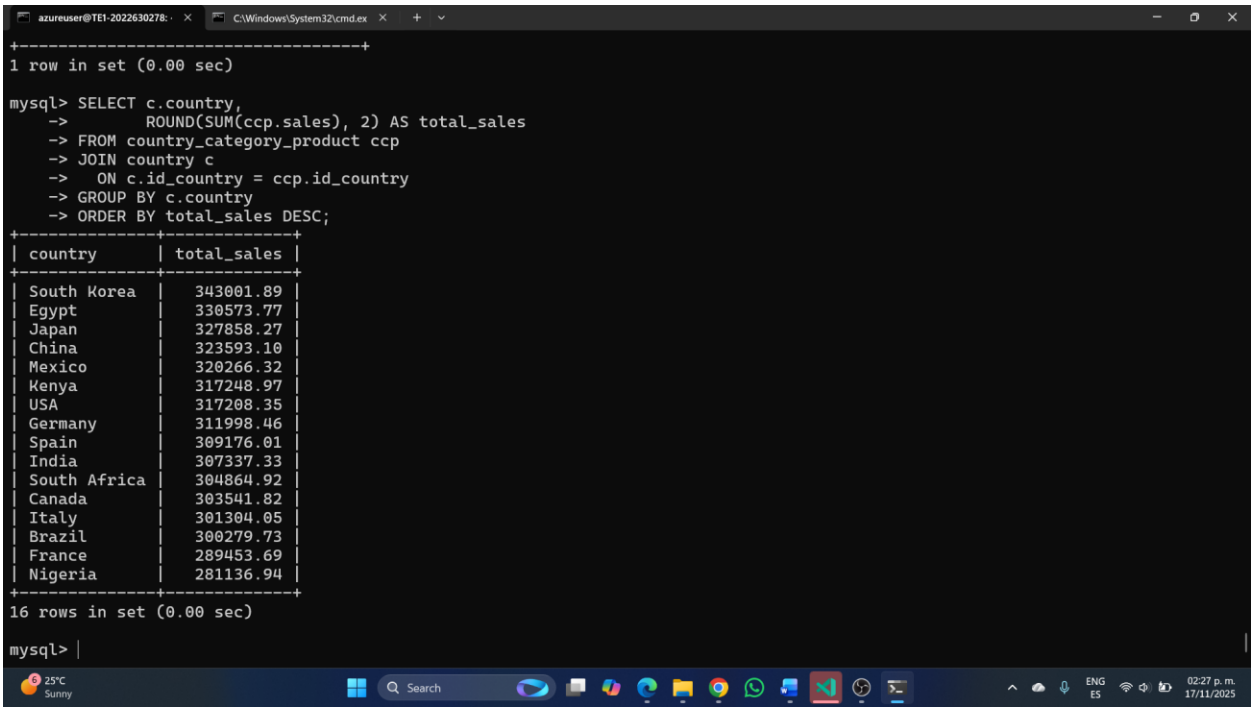


Imagen 18. Resultado de la consulta de acumulado de ventas por país (primera pantalla).

6.2 Acumulado de sales por country y category

Consulta que suma las ventas por país y categoría, uniendo la tabla agregada con las dimensiones country y category.

```
SELECT c.country,
    cat.category,
    ROUND(SUM(ccp.sales), 2) AS total_sales
FROM country_category_product ccp
JOIN country c
  ON c.id_country = ccp.id_country
JOIN category cat
  ON cat.id_category = ccp.id_category
```

GROUP BY c.country, cat.category

ORDER BY c.country ASC, total_sales DESC;

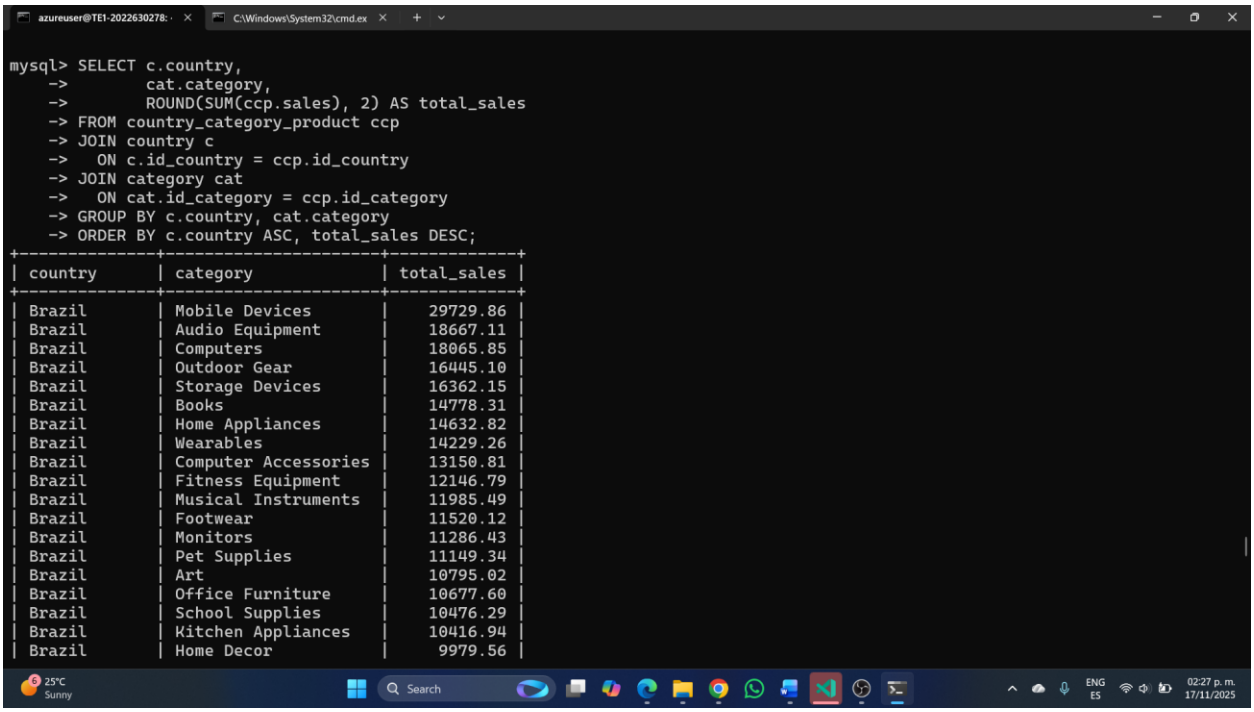


Imagen 19. Resultado de la consulta de acumulado de ventas por país y categoría (primera pantalla).

6.3 Acumulado de sales por country, category y product

Consulta que desglosa las ventas hasta nivel de producto, combinando las tres dimensiones con la tabla agregada.

SELECT c.country,

cat.category,

p.product,

ROUND(SUM(ccp.sales), 2) AS total_sales

FROM country_category_product ccp

JOIN country c

```
  ON c.id_country = ccp.id_country
JOIN category cat
  ON cat.id_category = ccp.id_category
JOIN product p
  ON p.id_product = ccp.id_product
GROUP BY c.country, cat.category, p.product
ORDER BY c.country ASC, cat.category ASC, total_sales DESC;
```

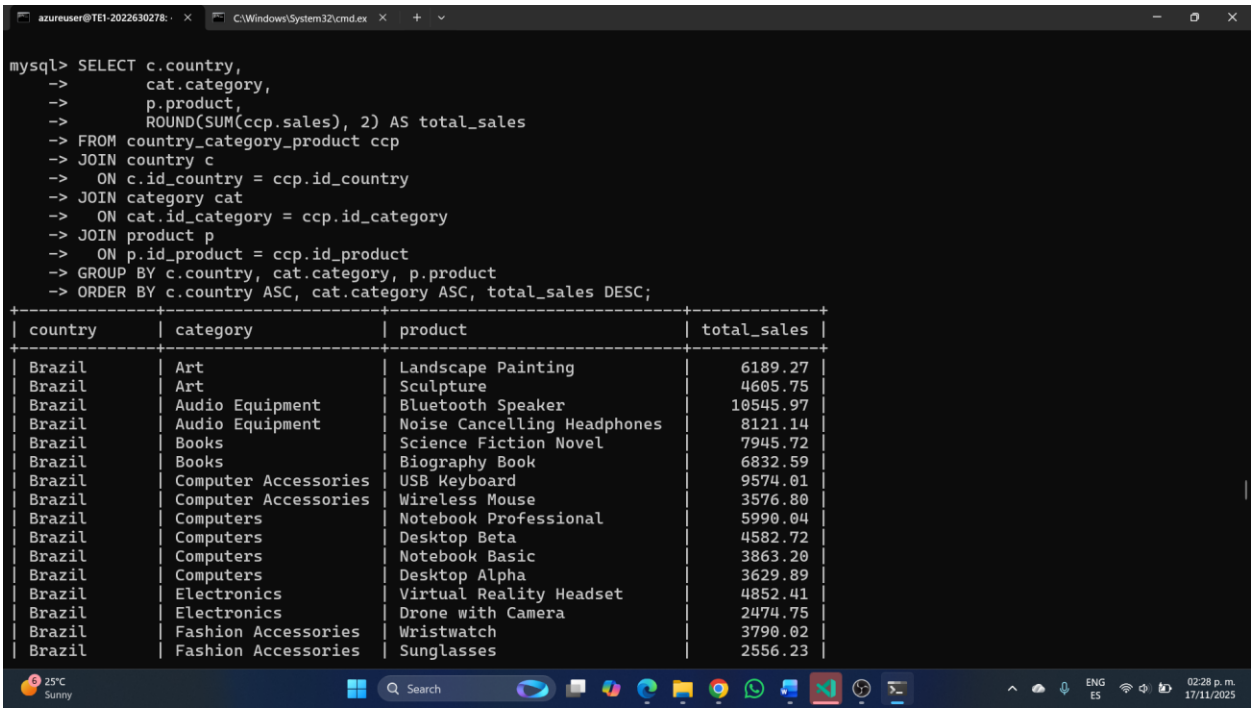


Imagen 20. Resultado de la consulta de acumulado de ventas por país, categoría y producto (primera pantalla).

7 Enlace al chat de la IA GitHub Copilot

El enlace al chat de la IA GitHub Copilot que usamos para el desarrollo de esta práctica es el siguiente:

- https://github.com/copilot/share/881c0286-0a44-8ca7-8811-a00604304183

8 Conclusiones

La práctica demostró un flujo completo y reproducible de generación de tablas agregadas apoyado en MySQL y Hadoop MapReduce, desplegado en una sola VM de Azure. Se verificó que, para consultas analíticas recurrentes, operar sobre una tabla agregada reduce el volumen de datos a escanear y simplifica las uniones, mejorando la latencia y el consumo de recursos frente a consultar directamente la fact_table.

La integración entre MySQL y Hadoop funcionó de forma fluida usando archivos intermedios CSV, y la ejecución en modo local de Hadoop fue suficiente para los objetivos del laboratorio, evitando la complejidad de HDFS/YARN. La etapa crítica consistió en cuidar el secure-file-priv (para INFILE/OUTFILE) y la correcta configuración del JAVA_HOME en hadoop-env.sh; ambas resultaron determinantes para evitar errores de permisos y de clase en tiempo de ejecución.

En términos de aprendizaje, se consolidaron:

- El diseño dimensional básico (dimensiones, fechas, fact_table) y su relación con tablas agregadas para OLAP.
- La implementación de un job MapReduce en Java (Mapper/Reducer/Driver) para agregaciones por claves compuestas.

9 Anexos

- **AggregationMapper.java**

```
import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;


// Entrada CSV: id_country,id_category,id_product,sales

public class AggregationMapper extends Mapper<LongWritable, Text, Text,
DoubleWritable> {


    private final Text outKey = new Text();

    private final DoubleWritable outVal = new DoubleWritable();


    @Override

    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {

        String line = value.toString().trim();

        if (line.isEmpty()) return;


        String[] parts = line.split(",", -1);

        if (parts.length != 4) return;


        String idCountry = parts[0].trim();

        String idCategory = parts[1].trim();

        String idProduct = parts[2].trim();
```

```
        String salesStr  = parts[3].trim();

        if (idCountry.isEmpty() || idCategory.isEmpty() || idProduct.isEmpty() ||
salesStr.isEmpty()) return;

        double sales;
        try {
            sales = Double.parseDouble(salesStr);
        } catch (NumberFormatException e) {
            return;
        }

        outKey.set(idCountry + "," + idCategory + "," + idProduct);
        outVal.set(sales);
        context.write(outKey, outVal);
    }
}
```

- **AggregationReducer.java**

```
import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class AggregationReducer extends Reducer<Text, DoubleWritable, Text,
DoubleWritable> {
    private final DoubleWritable result = new DoubleWritable();

    @Override
```

```
    protected void reduce(Text key, Iterable<DoubleWritable> values, Context context)
            throws IOException, InterruptedException {
        double sum = 0.0;
        for (DoubleWritable v : values) {
            sum += v.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

- **AggregationDriver.java**

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class AggregationDriver {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Uso: AggregationDriver <input_dir> <output_dir>");
            System.exit(1);
        }

        Configuration conf = new Configuration();
```

```
        Job job = Job.getInstance(conf, "Country-Category-Product Aggregation");

        job.setJarByClass(AggregationDriver.class);

        job.setMapperClass(AggregationMapper.class);
        job.setReducerClass(AggregationReducer.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(DoubleWritable.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DoubleWritable.class);

        TextInputFormat.addInputPath(job, new Path(args[0]));
        TextOutputFormat.setOutputPath(job, new Path(args[1]));

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

- **olap_workflow.sql**

```
-- Asume: CREATE DATABASE practica_olap; y SOURCE practica_olap.sql ya ejecutados
USE practica_olap;
```

```
-- Carga del CSV a sales_data (archivo colocado en /var/lib/mysql-files)
LOAD DATA INFILE '/var/lib/mysql-files/sales_data.csv'
INTO TABLE sales_data
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
IGNORE 1 LINES
(sales, order_date, product, customer, country, region, employee, category, weekday,
month, quarter, semester);

-- Dimensiones
INSERT INTO region (region)
SELECT DISTINCT region FROM sales_data WHERE region IS NOT NULL;

INSERT INTO country (country, id_region)
SELECT DISTINCT sd.country, r.id_region
FROM sales_data sd
JOIN region r ON r.region = sd.region;

INSERT INTO customer (customer)
SELECT DISTINCT customer FROM sales_data WHERE customer IS NOT NULL;

INSERT INTO employee (employee)
SELECT DISTINCT employee FROM sales_data WHERE employee IS NOT NULL;

INSERT INTO semester (semester)
SELECT DISTINCT semester FROM sales_data WHERE semester IS NOT NULL;

INSERT INTO quarter (quarter, id_semester)
```

```
SELECT DISTINCT sd.quarter, s.id_semester
FROM sales_data sd
JOIN semester s ON s.semester = sd.semester;

INSERT INTO month (month, id_quarter)
SELECT DISTINCT sd.month, q.id_quarter
FROM sales_data sd
JOIN quarter q ON q.quarter = sd.quarter;

INSERT INTO weekday (weekday)
SELECT DISTINCT weekday FROM sales_data WHERE weekday IS NOT NULL;

INSERT INTO category (category)
SELECT DISTINCT category FROM sales_data WHERE category IS NOT NULL;

INSERT INTO product (product, id_category)
SELECT DISTINCT sd.product, c.id_category
FROM sales_data sd
JOIN category c ON c.category = sd.category;

INSERT INTO order_date (order_date, id_weekday, id_month)
SELECT DISTINCT sd.order_date, w.id_weekday, m.id_month
FROM sales_data sd
JOIN weekday w ON w.weekday = sd.weekday
JOIN month m ON m.month = sd.month;

-- Fact table
```

```
INSERT INTO fact_table (sales, id_order_date, id_product, id_customer, id_country,
id_employee)
SELECT
  sd.sales,
  od.id_order_date,
  p.id_product,
  cu.id_customer,
  co.id_country,
  e.id_employee
FROM sales_data sd
JOIN order_date od ON od.order_date = sd.order_date
JOIN product p ON p.product = sd.product
JOIN customer cu ON cu.customer = sd.customer
JOIN country co ON co.country = sd.country
JOIN employee e ON e.employee = sd.employee;

-- Exportación a CSV para MapReduce
SELECT a.id_country, b.id_category, a.id_product, a.sales
INTO OUTFILE '/var/lib/mysql-files/country_category_product.csv'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ""
LINES TERMINATED BY '\n'
FROM fact_table a
JOIN product b ON a.id_product = b.id_product;

-- Importación del resultado de MapReduce a la tabla agregada
-- (asume que /var/lib/mysql-files/ccp_part.csv ya existe y usa comas)
TRUNCATE TABLE country_category_product;
```

```
LOAD DATA INFILE '/var/lib/mysql-files/ccp_part.csv'
INTO TABLE country_category_product
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
(id_country, id_category, id_product, sales);

-- Consultas de cubos
-- 1) Ventas por country
SELECT c.country, ROUND(SUM(ccp.sales), 2) AS total_sales
FROM country_category_product ccp
JOIN country c ON c.id_country = ccp.id_country
GROUP BY c.country
ORDER BY total_sales DESC;

-- 2) Ventas por country y category
SELECT c.country, cat.category, ROUND(SUM(ccp.sales), 2) AS total_sales
FROM country_category_product ccp
JOIN country c ON c.id_country = ccp.id_country
JOIN category cat ON cat.id_category = ccp.id_category
GROUP BY c.country, cat.category
ORDER BY c.country, total_sales DESC;

-- 3) Ventas por country, category y product
SELECT c.country, cat.category, p.product, ROUND(SUM(ccp.sales), 2) AS total_sales
FROM country_category_product ccp
JOIN country c ON c.id_country = ccp.id_country
JOIN category cat ON cat.id_category = ccp.id_category
```

```
JOIN product p ON p.id_product = ccp.id_product
GROUP BY c.country, cat.category, p.product
ORDER BY c.country, cat.category, total_sales DESC;
```

- **hadoop-env-snippet.sh**

```
# Ruta ejemplo en Ubuntu 22.04 con OpenJDK 17
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
```

- **commands.sh**

```
# Conexión y transferencia
ssh azureuser@<IP_PUBLICA>
sftp azureuser@<IP_PUBLICA>
put practica_olap.sql
put sales_data.csv
exit

# Paquetes
sudo apt update && sudo apt -y upgrade
sudo apt -y install mysql-server mysql-client openjdk-17-jdk-headless wget tar nano

# MySQL secure-file-priv
sudo mkdir -p /var/lib/mysql-files
sudo chown mysql:mysql /var/lib/mysql-files
sudo chmod 750 /var/lib/mysql-files

# Hadoop 3.4.0 (x86_64)
cd ~
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz
tar -xzf hadoop-3.4.0.tar.gz
```

```
# Variables de entorno
echo 'export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64' >> ~/.bashrc
echo 'export HADOOP_HOME=$HOME/hadoop-3.4.0' >> ~/.bashrc
echo 'export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin' >> ~/.bashrc
source ~/.bashrc

# Compilación y ejecución MapReduce
mkdir -p ~/prueba
cd ~/prueba
javac -classpath "$(hadoop classpath)" -d build *.java
jar -cvf Aggregation.jar -C build .
mkdir -p input
cp ~/country_category_product.csv input/
rm -rf output
hadoop jar Aggregation.jar AggregationDriver input output
sed -i 's/\t/,/g' output/part-r-00000
sudo cp output/part-r-00000 /var/lib/mysql-files/ccp_part.csv
```

10 Referencias (formato IEEE)

[1] Oracle and/or its affiliates, “MySQL 8.0 Reference Manual,” MySQL Documentation. [En línea]. Disponible en: https://dev.mysql.com/doc/refman/8.0/en/ (Accedido: 17-nov-2025).

[2] Apache Software Foundation, “Apache Hadoop 3.4 – Documentation,” Apache Hadoop. [En línea]. Disponible en: https://hadoop.apache.org/docs/r3.4.0/ (Accedido: 17-nov-2025).

[3] Eclipse Adoptium, “Temurin 17 (OpenJDK 17) Documentation,” Adoptium. [En línea]. Disponible en: https://adoptium.net/temurin/releases/?version=17 (Accedido: 17-nov-2025).

[4] Canonical Ltd., “Ubuntu 22.04 LTS Documentation,” Ubuntu. [En línea]. Disponible en: https://help.ubuntu.com/ (Accedido: 17-nov-2025).

[5] Microsoft, “Create a Linux virtual machine in the Azure portal,” Microsoft Learn. [En línea]. Disponible en: https://learn.microsoft.com/azure/virtual-machines/linux/quick-create-portal (Accedido: 17-nov-2025).

[6] Apache Software Foundation, “MapReduce Tutorial,” Apache Hadoop. [En línea]. Disponible en: https://hadoop.apache.org/docs/r3.4.0/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html (Accedido: 17-nov-2025).

[7] Oracle and/or its affiliates, “MySQL Server System Variables – secure_file_priv,” MySQL Documentation. [En línea]. Disponible en: https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_secure_file_priv (Accedido: 17-nov-2025).

[8] The GNU Project, “sed, a stream editor,” GNU sed Manual. [En línea]. Disponible en: https://www.gnu.org/software/sed/manual/ (Accedido: 17-nov-2025).

Si quieres, puedo entregarte estas secciones en un archivo Markdown listo para imprimir o integrarlas al documento completo con portada e índice.