Poblar datos csv y crear consultas con SQL

Archivos csv

product_dim.csv

product_id,product_name,category
1,Widget A,Hardware
2,Widget B,Hardware
3,Gadget C,Electronics
4,Gizmo D,Electronics
5,Thing E,Accessories
6,Item F,Accessories
7,Product G,Hardware
8,Product H,Electronics
9,Product I,Accessories
10,Product J,Hardware

store_dim.csv

store_id,store_name,city,region
1,Tienda Centro,Managua,Central
2,Tienda Norte,León,Occidente
3,Tienda Sur,Masaya,Central
4,Tienda Este,Bluefields,Caribe
5,Tienda Oeste,Estelí,Occidente
6,Tienda Playa,Corinto,Pacífico
7,Tienda Pueblo,Chinandega,Occidente
8,Tienda Capital,Managua,Central

date_dim.csv

date,year,month,day,month_name
2025-01-05,2025,1,5,enero
2025-02-12,2025,2,12,febrero
2025-03-03,2025,3,3,marzo
2025-04-10,2025,4,10,abril
2025-05-21,2025,5,21,mayo
2025-06-30,2025,6,30,junio
2025-07-14,2025,7,14,julio

2025-08-08,2025,8,8,agosto
2025-09-01,2025,9,1,septiembre
2024-12-25,2024,12,25,diciembre
2024-11-11,2024,11,11,noviembre
2024-10-31,2024,10,31,octubre

sales_fact.csv

sale_id,product_id,store_id,sale_date,quantity,unit_price,total_price
1,1,1,2025-01-05,2,10.00,20.00
2,2,1,2025-01-05,1,15.00,15.00
3,3,2,2025-02-12,3,7.50,22.50
4,4,3,2025-03-03,1,50.00,50.00
5,5,4,2025-04-10,5,3.00,15.00
6,1,2,2025-05-21,2,10.00,20.00
7,6,5,2025-06-30,4,8.00,32.00
8,7,6,2025-07-14,1,12.00,12.00
9,8,7,2025-08-08,2,20.00,40.00
10,9,8,2025-09-01,3,6.00,18.00
11,10,1,2024-12-25,1,25.00,25.00
12,2,3,2024-11-11,2,15.00,30.00
13,3,4,2024-10-31,1,7.50,7.50
14,1,1,2025-05-21,1,10.00,10.00
15,4,2,2025-03-03,2,50.00,100.00
16,5,3,2025-04-10,1,3.00,3.00
17,6,4,2025-06-30,2,8.00,16.00
18,7,5,2025-07-14,3,12.00,36.00
19,8,6,2025-08-08,1,20.00,20.00
20,9,7,2025-09-01,5,6.00,30.00

Paso a paso en Spoon (PDI) — crear DB HSQL local, tablas y cargar CSV
Nota: usamos HSQLDB en modo archivo (todo se guarda local). No necesitas MySQL/Postgres.
Tiempo estimado: 15–25 minutos.

1. Abrir Spoon (PDI) (Pentaho Data Integration).
2. Crear una conexión de base de datos:
    - Menú: View > Database connections → New.
    - Nombre: HSQL_sales

- Type: Generic database
- URL (recomendado file-mode):
  - Linux/mac:

    jdbc:hsqldb:file:/home/miusuario/pentaho_ejercicios/hsqldb/sales;shutdown=true
  - Windows:

    jdbc:hsqldb:file:C:/pentaho_ejercicios/hsqldb/sales;shutdown=true

  Driver Class: org.hsqldb.jdbcDriver
- User: SA Password: (dejar vacío)
- Test → should pass. (Si falla, crea la carpeta indicada y vuelve a probar.)
- **(Obligatorio)** Guardar conexión.

3. Crear transformación nueva (File > New > Transformation).
4. Paso A — Ejecutar DDL para crear tablas:
   - Añadir step Execute SQL script (del grupo Scripting).
   - Double-click > Database: HSQL_sales.
   - En SQL script, pegar este DDL:

```
DROP TABLE IF EXISTS product_dim;
CREATE TABLE product_dim (
  product_id INTEGER NOT NULL,
  product_name VARCHAR(100),
  category VARCHAR(50),
  PRIMARY KEY (product_id)
);

DROP TABLE IF EXISTS store_dim;
CREATE TABLE store_dim (
  store_id INTEGER NOT NULL,
  store_name VARCHAR(100),
  city VARCHAR(50),
  region VARCHAR(50),
  PRIMARY KEY (store_id)
);

DROP TABLE IF EXISTS date_dim;
CREATE TABLE date_dim (
  date DATE NOT NULL,
```

```
  year INTEGER,
  month INTEGER,
  day INTEGER,
  month_name VARCHAR(20),
  PRIMARY KEY (date)
);

DROP TABLE IF EXISTS sales_fact;
CREATE TABLE sales_fact (
  sale_id INTEGER NOT NULL,
  product_id INTEGER,
  store_id INTEGER,
  sale_date DATE,
  quantity INTEGER,
  unit_price DECIMAL(10,2),
  total_price DECIMAL(12,2),
  PRIMARY KEY (sale_id)
);
```

- Ejecuta el step (Run single step) o ejecuta la transformación; confirmas que las tablas se crean (mensaje OK).

5. Paso B — Cargar cada CSV (hacer 4 flujos paralelos o secuenciales):
    - Para product_dim.csv:
        - Añadir Text File Input → configurar ruta al CSV (C:/pentaho_ejercicios/csv/product_dim.csv).
        - Definir campos: product_id (Integer), product_name (String), category (String).
        - Añadir Table output → Database: HSQL_sales, Table: product_dim.
        - En Table output mapear campos a columnas y activar Specify database fields? si es necesario.
    - Repetir para store_dim.csv → store_dim, date_dim.csv → date_dim.
    - Para sales_fact.csv:
        - Text File Input configurar y mapear sale_id (Integer), product_id (Integer), store_id (Integer), sale_date (Date) —

especifica formato yyyy-MM-dd, quantity (Integer), unit_price (Number), total_price (Number).

- Table output → sales_fact.

6. Guardar transformación como load_sales.ktr.
7. Ejecutar transformación completa (Run) y confirmar en el log que filas insertadas (rows written).
8. Verificar en Spoon con un Table input + Preview o con SQL select en Execute SQL para confirmar datos:
    - SELECT COUNT(*) FROM sales_fact; debe devolver 20.

🔹 **Consultas sobre product_dim**

1. Ver todos los productos:

```
SELECT * FROM product_dim;
```

2. Solo productos de categoría **Hardware**:

```
SELECT * FROM product_dim WHERE category = 'Hardware';
```

3. Contar productos por categoría:

```
SELECT category, COUNT(*) AS total
FROM product_dim
GROUP BY category;
```

4. Listar productos ordenados alfabéticamente:

```
SELECT * FROM product_dim
ORDER BY product_name ASC;
```

🔹 **Consultas sobre store_dim**

5. Ver todas las tiendas en la región Central:

```
SELECT * FROM store_dim WHERE region = 'Central';
```

6. Número de tiendas por región:

```
SELECT region, COUNT(*) AS total_tiendas
FROM store_dim
GROUP BY region;
```

7. Listar tiendas por ciudad:

```
SELECT city, store_name
FROM store_dim
ORDER BY city;
```

🔹 **Consultas sobre date_dim**

8. Ver todos los registros del año 2025:

```
SELECT * FROM date_dim WHERE year = 2025;
```

9. Fechas agrupadas por mes:

```
SELECT month, COUNT(*) AS dias_registrados
FROM date_dim
GROUP BY month
ORDER BY month;
```

10. Listar fechas en diciembre:

```
SELECT * FROM date_dim WHERE month_name = 'diciembre';
```

🔹 **Consultas sobre sales_fact**

11. Ver todas las ventas:

```
SELECT * FROM sales_fact;
```

12. Total de ventas ($) por producto:

```
SELECT product_id, SUM(total_price) AS total_ventas
FROM sales_fact
GROUP BY product_id
ORDER BY total_ventas DESC;
```

13. Total de ventas por tienda:

```
SELECT store_id, SUM(total_price) AS total_ventas
FROM sales_fact
GROUP BY store_id
ORDER BY total_ventas DESC;
```

14. Promedio de cantidad vendida:

```
SELECT AVG(quantity) AS promedio_cantidad
FROM sales_fact;
```

15. Ventas mayores a $50:

```
SELECT * FROM sales_fact WHERE total_price > 50;
```

16. Ventas en el año 2025:

```
SELECT *
FROM sales_fact sf
JOIN date_dim d ON sf.sale_date = d.date
WHERE d.year = 2025;
```

17. Top 3 productos con mayor ingreso:

```
SELECT product_id, SUM(total_price) AS total_ventas
FROM sales_fact
GROUP BY product_id
ORDER BY total_ventas DESC
LIMIT 3;
```

18. Ventas agrupadas por mes:

```
SELECT d.month, SUM(sf.total_price) AS ventas_mes
FROM sales_fact sf
JOIN date_dim d ON sf.sale_date = d.date
GROUP BY d.month
ORDER BY d.month;
```

19. Ventas por región:

```
SELECT s.region, SUM(sf.total_price) AS ventas_region
FROM sales_fact sf
JOIN store_dim s ON sf.store_id = s.store_id
GROUP BY s.region;
```

20. Ticket promedio (total / cantidad):

```
SELECT SUM(total_price) / SUM(quantity) AS ticket_promedio
```

```
FROM sales_fact;
```

Ejercicios de práctica

🔹 Básico

1. **Filtrar productos de categoría "Electronics".**
    - Tabla: product_dim.
    - Instrucción: mostrar product_id, product_name de la categoría "Electronics".
2. **Listar todas las tiendas de la región "Occidente".**
    - Tabla: store_dim.
    - Ordenar por city.
3. **Obtener las ventas cuyo total sea mayor a 30 dólares.**
    - Tabla: sales_fact.
    - Mostrar sale_id, total_price.
4. **Contar cuántos registros de fecha hay en el año 2024.**
    - Tabla: date_dim.
5. **Mostrar los productos en orden alfabético.**
    - Tabla: product_dim.
    - Campos: product_name.

🔹 Intermedio (joins + agregaciones)

6. **Total de ventas por cada producto.**
    - Joins: sales_fact con product_dim.
    - Campos: product_name, SUM(total_price).
7. **Ventas totales por tienda.**
    - Joins: sales_fact con store_dim.
    - Campos: store_name, SUM(total_price).
8. **Cantidad promedio de productos vendidos por categoría.**
    - Joins: sales_fact con product_dim.
    - Campos: category, AVG(quantity).
9. **Ventas por mes en 2025.**
    - Joins: sales_fact con date_dim.
    - Agrupar por month_name.
10. **Top 3 regiones por ventas.**

- Joins: sales_fact con store_dim.
- Agrupar por region, ordenar y limitar a 3.

🔹 Nivel Avanzado (consultas de análisis tipo OLAP)

11. **Ticket promedio de ventas (Venta / Cantidad).**
- Usar SUM(total_price)/SUM(quantity).
12. **Ventas por año y categoría.**
- Joins: sales_fact + date_dim + product_dim.
- Agrupar por year, category.
13. **Detectar la tienda con mayor venta en 2025.**
- Joins: sales_fact + store_dim + date_dim.
- Filtrar year=2025.
14. **Ventas acumuladas mes a mes.**
- Tabla: sales_fact + date_dim.
- Usar función SUM con agrupación acumulativa.
15. **Comparar ventas entre 2024 y 2025.**
- Mostrar total por año.
- Pregunta: ¿qué año vendió más?