Guía paso a paso: Evaluación Pentaho

Resumen del ejemplo

- Objetivo práctico: Construir un flujo ETL en Spoon que lea un **fact** (ventas) y 3 **dimensiones** (producto, cliente, tienda), haga joins entre ellos (sin BD) y genere agregados por **año**, **mes** y **categoría de producto**.
- Resultado: archivos CSV agregados listos para análisis (por año, mes y categoría).
- Herramienta: **Pentaho 7.1**.
- Archivos: FactSales.csv, DimProduct.csv, DimCustomer.csv, DimStore.csv.
- Técnica de unión: **Merge Join** (requiere ordenar las entradas con **Sort rows**).
- Operaciones clave: cálculo de sales_amount, extracción de año/mes, Group By para agregaciones, Text file output para la salida.

1) Archivos CSV — contenido completo (copiar/pegar)

Guarda estos archivos en tu carpeta local de ejercicios.

FactSales.csv

```
sale_id,date,product_id,customer_id,store_id,quantity,unit_price
1,2024-01-10,P001,C001,S001,2,15.50
2,2024-01-11,P002,C002,S002,1,45.00
3,2024-01-15,P001,C003,S001,3,15.50
4,2024-02-05,P003,C001,S003,5,8.00
5,2024-02-20,P004,C004,S002,2,120.00
6,2023-12-30,P002,C002,S001,1,45.00
7,2023-11-12,P005,C005,S003,10,2.50
8,2024-03-03,P001,C001,S001,1,15.50
9,2024-03-15,P003,C006,S002,4,8.00
10,2024-03-20,P002,C002,S001,2,45.00
11,2024-04-01,P004,C004,S003,1,120.00
12,2024-04-10,P005,C005,S002,6,2.50
```

DimProduct.csv

```
product_id,product_name,category,brand,unit_cost
```

P001,Classic Burger,Burgers,GrillMaster,9.00
P002,Double Burger,Burgers,GrillMaster,20.00
P003,Fries,Side,PotatoCo,1.50
P004,Milkshake,Beverage,SweetSip,40.00
P005,Salad,Healthy,GreenLeaf,1.00

DimCustomer.csv

customer_id,customer_name,gender,age_group,city,country
C001,Ana López,F,25-34,Managua,Nicaragua
C002,Carlos Ruiz,M,35-44,Juigalpa,Nicaragua
C003,María Gómez,F,18-24,Managua,Nicaragua
C004,José Pérez,M,45-54,Chontales,Nicaragua
C005,Lucía Torres,F,25-34,Managua,Nicaragua
C006,Pedro Díaz,M,30-34,Estelí,Nicaragua

DimStore.csv

store_id,store_name,region,city
S001,Central Store,Centro,Managua
S002,West Branch,Oeste,Juigalpa
S003,East Outlet,Este,Chontales

2) Estructura del flujo (visión general)

- **Single transformation**: ETL_Cubo_Ventas.ktr
- Lecturas: 4 pasos **Text file input** (Fact + 3 Dim)
- Uniones sucesivas con **Sort rows** + **Merge Join**:
    1. Fact ↔ DimProduct → resultado1
    2. resultado1 ↔ DimCustomer → resultado2
    3. resultado2 ↔ DimStore → resultado_final
- Añadir cálculo sales_amount = quantity * unit_price
- Extraer year y month desde date (cadena YYYY-MM-DD)
- Agregaciones con **Group By** (por year; por year-month; por year+category)
- Salidas con **Text file output** (CSV con encabezados)

3) Transformación detallada — pasos en Spoon (ordenado, con configuraciones clave)

Antes de empezar: Abre Pentaho. Crea una nueva **Transformation** y guarda como ETL_Cubo_Ventas.ktr en la misma carpeta del ejercicio.

Paso A — Lectura del hecho (FactSales)

1. Agrega **Text file input** → nómbralo Read_FactSales.
2. Pestaña *File*:
    - Filename: busca la ruta para el archivo FactSales.csv (ajusta ruta)

- Delimiter: ,
- Enclosure: " (si usa)
- Header: 1 (sí)
- Charset: UTF-8

3. Pestaña *Content*:
   - Record separator: \n
4. Pestaña *Fields*: define columnas (orden y tipo):
   - sale_id — Integer
   - date — String (o Date con formato yyyy-MM-dd) — recomiendo leer como **String** y convertir luego.
   - product_id — String
   - customer_id — String
   - store_id — String
   - quantity — Integer
   - unit_price — Number (format: 0.00)
5. Preview → confirma filas leídas.

Paso B — Lectura de dimensión Producto

1. **Text file input** → Read_DimProduct
2. File: DimProduct.csv, same delimiter/header config.
3. Fields:
   - product_id — String
   - product_name — String
   - category — String
   - brand — String
   - unit_cost — Number

Paso C — Preparar para Merge Join (Fact ↔ Product)

IMPORTANTE: Merge Join requiere **ambos** streams ordenados por la(s) llave(s) de join.

1. Añade **Sort rows** para Read_FactSales → Sort_Fact_product:
   - Sort fields: product_id (ascending)
   - Nota: Si te interesa multi-join posterior, puedes ordenar por la llave siguiente de cada join justo antes de ese join.
2. Añade **Sort rows** para Read_DimProduct → Sort_Product:
   - Sort fields: product_id (ascending)
3. Añade **Merge Join** → Join_Fact_Product:
   - Conecta Sort_Fact_product (primer input) y Sort_Product (segundo input) a Join_Fact_Product.
   - Configuración:
     - Join type: Left outer (asumimos queremos todas las facturas aunque falte producto)
     - Key fields: product_id (left) = product_id (right)

        - Selecciona campos a mantener (marca los campos de DimProduct que quieres agregar: product_name, category, brand, unit_cost)
    - Resultado: stream con columnas de fact + atributos de producto.

Paso D — Repetir patrón para Cliente

1. **Text file input** → Read_DimCustomer (si no lo hiciste antes).
2. **Sort rows** → Sort_Customer por customer_id.
3. **Sort rows** → Sort_joinResult_by_customer (ordena Join_Fact_Product salida por customer_id):
    - Para ordenar la salida del join: conecta salida de Join_Fact_Product a Sort_joinResult_by_customer.
4. **Merge Join** → Join_With_Customer:
    - Inputs: Sort_joinResult_by_customer (left) y Sort_Customer (right).
    - Key: customer_id
    - Tipo: Left outer
    - Selecciona campos de cliente (customer_name, city, country, age_group, gender).

Paso E — Repetir patrón para Store

1. **Text file input** → Read_DimStore
2. **Sort rows** → Sort_Store por store_id
3. **Sort rows** → Sort_joinResult_by_store (ordena salida de Join_With_Customer por store_id)
4. **Merge Join** → Join_With_Store:
    - Key: store_id (left=join result, right=store)
    - Tipo: Left outer
    - Selecciona campos store_name, region, city (si deseas diferenciarlos por alias, renómbralos con *Select values*).

Resultado final: flujo Join_With_Store con todos los atributos: fact + product + customer + store.

Paso F — Calcular sales_amount (cantidad * precio)

1. Usa **Modified Java Script Value** (o **Calculator** si prefieres) → Calc_sales_amount.
2. Script (ejemplo en JS):

```
// inputs: quantity (Integer), unit_price (Number)
var sales_amount = quantity * unit_price;
```

- Output field: sales_amount — Type: Number (precision 2).

*(Si usas Calculator: operación A * B con fields quantity y unit_price.)*

Paso G — Extraer year y month desde date

1. Usa **Modified Java Script Value** → Extract_date_parts.
2. Ejemplo de script (si date es string YYYY-MM-DD):

```
var d = date; // e.g., "2024-01-10"
var year = d.substring(0,4);
var month = d.substring(5,7);
var year_month = year + "-" + month;
var month_num = parseInt(month, 10);
var quarter = Math.floor((month_num + 2) / 3);
```

- Output fields:
  - `year` — String (o Integer si conviertes)
  - `month` — String
  - `year_month` — String (ej. `2024-01`)
  - `quarter` — Integer

Nota: si tu `date` viene como Date, usa las funciones de Date en JS o convierte adecuadamente.

Paso H — Limpieza y control de calidad (opcional pero recomendable)

- **Filter rows** → eliminar filas con quantity <= 0 o unit_price <= 0.
- **Trim fields** → usa Select values + *String operations* para recortar espacios.
- **Unique rows** si necesitas eliminar duplicados.

Paso I — Agregaciones (Group By)

Crea 3 pipelines de agregación (puedes duplicar el stream de salida final hacia cada Group By):

I.1 Agregado por AÑO

- Step: **Group by** → Agg_By_Year
  - Group fields: year
  - Aggregates:
    - Sum(sales_amount) → total_sales
    - Sum(quantity) → total_qty
    - Count(sale_id) → sale_count

- Output: **Text file output** sales_by_year.csv

I.2 Agregado por AÑO-MES

- Group fields: year_month
- Same aggregates → Output: sales_by_month.csv

I.3 Agregado por AÑO + CATEGORÍA

- Group fields: year, category
- Aggregates → Output: sales_by_category_year.csv

Configuración de Text file output

- Filename: output\sales_by_year.csv (ajusta ruta)
- Delimiter: ,
- Header: ON
- Enclosure: "
- Charset: UTF-8

---

4) Detalles clave y tips técnicos

- **Merge Join**: entradas deben estar **ordenadas por la llave**; usa Sort rows. Si el join falla, revisa que los tipos y longitudes de la llave sean idénticos (ej. String vs Integer).
- **Tipos**: Mantén product_id, customer_id, store_id como **String** para evitar casting.
- **Previsualizar**: usa Preview en cada step para verificar 10–50 filas antes de ejecutar todo.
- **Rendimiento**: con CSVs pequeños para clase no hay problema; en producción preferirías BD o archivos optimizados.
- **Decimales**: controla formato con Number y Select values (round) para outputs consistentes.
- **Errores comunes**:
  - Delimitador incorrecto (,) → el archivo se abre en una sola columna.
  - Headers mal definidos → fila header tratada como registro.
  - Merge Join sin Sort → error o resultados incorrectos.

---

5) Ejemplos de salidas (CSV de ejemplo generados por el flujo)

sales_by_year.csv (esperado)

year,total_sales,total_qty,sale_count
2024,675.00,27,10
2023,70.00,11,2

sales_by_month.csv (esperado; campo year_month)

```
year_month,total_sales,total_qty,sale_count
2023-11,25.00,10,1
2023-12,45.00,1,1
2024-01,122.50,6,3
2024-02,280.00,7,2
2024-03,137.50,7,3
2024-04,135.00,7,2
```

6) Orquestación con Jobs (opcional)

- Crea un **Job** (ETL_Cubo_Ventas.kjb) que ejecute la transformación:
    1. Start → Transformación ETL_Cubo_Ventas.ktr
    2. On success → Opcional: mover archivos de salida a carpeta de informes o comprimir.
- Esto permite ejecutarlo en lote o programarlo con el scheduler del servidor si tienes PDI en servidor.