

DISEÑO DE UN DATA MART PARA EL DATASET ECOMMERCEFAKEDATA

SISTEMAS DE BASES DE DATOS AVANZADAS

INTEGRANTES DEL GRUPO

BAJAÑA FERNANDEZ LUIS ANDRES

CAÑARTE ORMAZA JOHN EDUARDO

COTERA MENDOZA WASHINGTON GENARO

MITE GUILLEN BRIAN ANDREI

Proceso de ETL, Diseño de Data Mart y Visualización en Power BI

1. Introducción

En este documento se presenta el proceso completo de construcción de un Data Mart de comercio electrónico a partir de datos transaccionales. El trabajo incluye la extracción, transformación y carga (ETL) de los datos en un modelo dimensional, su explotación mediante Power BI y la generación de indicadores claves de negocio (KPIs).

2. Dataset seleccionado.

El dataset proviene de Kaggle – [ECommerceFakeData](#).

Incluye información clave de un ecosistema de comercio electrónico:

- **categories.csv** → categorías de productos, con jerarquías (Electronics → Smartphones).
- **products.csv** → catálogo de productos con atributos de marca, precio, costo, peso, dimensiones, descripción.
- **customers.csv** → clientes con datos demográficos, geográficos, segmentación de marketing, fecha de registro y última conexión.
- **orders.csv y orders_part_00x.csv** → pedidos realizados por los clientes, con fechas, estado de la orden, método de pago, direcciones de envío/facturación, descuentos, impuestos, costos de envío y totales.
- **order_items.csv** → detalle de ítems dentro de cada pedido (qué producto, cuántas unidades, precio unitario, descuentos).
- **reviews.csv** → reseñas de clientes con calificaciones, comentarios y votos de utilidad.
- **inventory_logs.csv** → movimientos de inventario (entrada, salida, devoluciones, ajustes) con cantidades, razones y fechas.

Este dataset fue elegido porque:

- Contiene información transaccional, de clientes y de productos típica de un e-commerce real.
- Tiene un volumen considerable (>500MB) que permite probar procesos ETL y modelado dimensional a escala.

- Permite analizar no solo ventas, sino también satisfacción del cliente (reviews) y logística (inventario), ampliando el alcance del Data Mart.

The screenshot shows a SQL IDE with a dark theme. The main editor displays a script named '001_create_dm.sql'. The script starts with a comment '-- Base' and creates a database 'dm_ecommerce' with a character set of 'utf8mb4' and a collate of 'utf8mb4_0900_ai_ci'. It then uses the 'dm_ecommerce' database. Following a comment '-- STAGING (copia "raw/typed")', it drops the 'stg_categories' table if it exists and creates it with columns: 'category_id' (BIGINT), 'category_name' (VARCHAR(255)), 'parent_category' (VARCHAR(255)), and 'created_at' (DATETIME). It then drops the 'stg_customers' table if it exists and creates it with columns: 'customer_id' (BIGINT), 'email' (VARCHAR(255)), 'first_name' (VARCHAR(100)), 'last_name' (VARCHAR(100)), 'phone' (VARCHAR(50)), 'date_of_birth' (DATE), 'gender' (VARCHAR(20)), 'country' (VARCHAR(100)), 'city' (VARCHAR(100)), 'postal_code' (VARCHAR(20)), 'address' (VARCHAR(255)), 'registration_date' (DATE), 'last_login' (DATETIME), 'is_active' (TINYINT(1)), 'customer_segment' (VARCHAR(50)), and 'marketing_consent' (TINYINT(1)). Finally, it drops the 'stg_inventory_logs' table if it exists and creates it with columns: 'log_id' (BIGINT), 'product_id' (BIGINT), 'movement_type' (VARCHAR(50)), 'quantity_change' (INT), 'reason' (VARCHAR(255)), 'timestamp' (DATETIME), 'reference_id' (VARCHAR(100)), and 'notes' (VARCHAR(255)). The Explorer panel on the right shows the project structure for 'DM-ECOMMERCE', including folders for 'docker', 'docs', 'drivers', 'etl', 'mysql_data', 'powerbi', 'sql', and 'staging'. The 'sql' folder contains '001_create_dm.sql', '002_load_dim_date.sql', and '100_dwh_ddl.sql'. The 'staging' folder contains various CSV files like 'categories.csv', 'customers.csv', 'inventory_logs.csv', 'order_items.csv', 'orders_part_001.csv', 'orders_part_002.csv', 'orders_part_003.csv', 'orders_part_004.csv', 'orders.csv', 'products.csv', and 'reviews.csv'.

```

001_create_dm.sql
sql > 001_create_dm.sql
1  -- Base
2  CREATE DATABASE IF NOT EXISTS dm_ecommerce
3  CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci;
4  USE dm_ecommerce;
5
6
7  -- STAGING (copia "raw/typed")
8
9  DROP TABLE IF EXISTS stg_categories;
10 CREATE TABLE stg_categories (
11     category_id      BIGINT,
12     category_name    VARCHAR(255),
13     parent_category  VARCHAR(255),
14     created_at       DATETIME
15 );
16
17 DROP TABLE IF EXISTS stg_customers;
18 CREATE TABLE stg_customers (
19     customer_id      BIGINT,
20     email            VARCHAR(255),
21     first_name       VARCHAR(100),
22     last_name        VARCHAR(100),
23     phone            VARCHAR(50),
24     date_of_birth    DATE,
25     gender           VARCHAR(20),
26     country          VARCHAR(100),
27     city             VARCHAR(100),
28     postal_code      VARCHAR(20),
29     address          VARCHAR(255),
30     registration_date DATE,
31     last_login       DATETIME,
32     is_active        TINYINT(1),
33     customer_segment VARCHAR(50),
34     marketing_consent TINYINT(1)
35 );
36
37 DROP TABLE IF EXISTS stg_inventory_logs;
38 CREATE TABLE stg_inventory_logs (
39     log_id           BIGINT,
40     product_id       BIGINT,
41     movement_type    VARCHAR(50),
42     quantity_change  INT,
43     reason           VARCHAR(255),
44     `timestamp`      DATETIME,
45     reference_id     VARCHAR(100),
46     notes            VARCHAR(255)
47 );
48

```

3. Diseño del Data Mart.

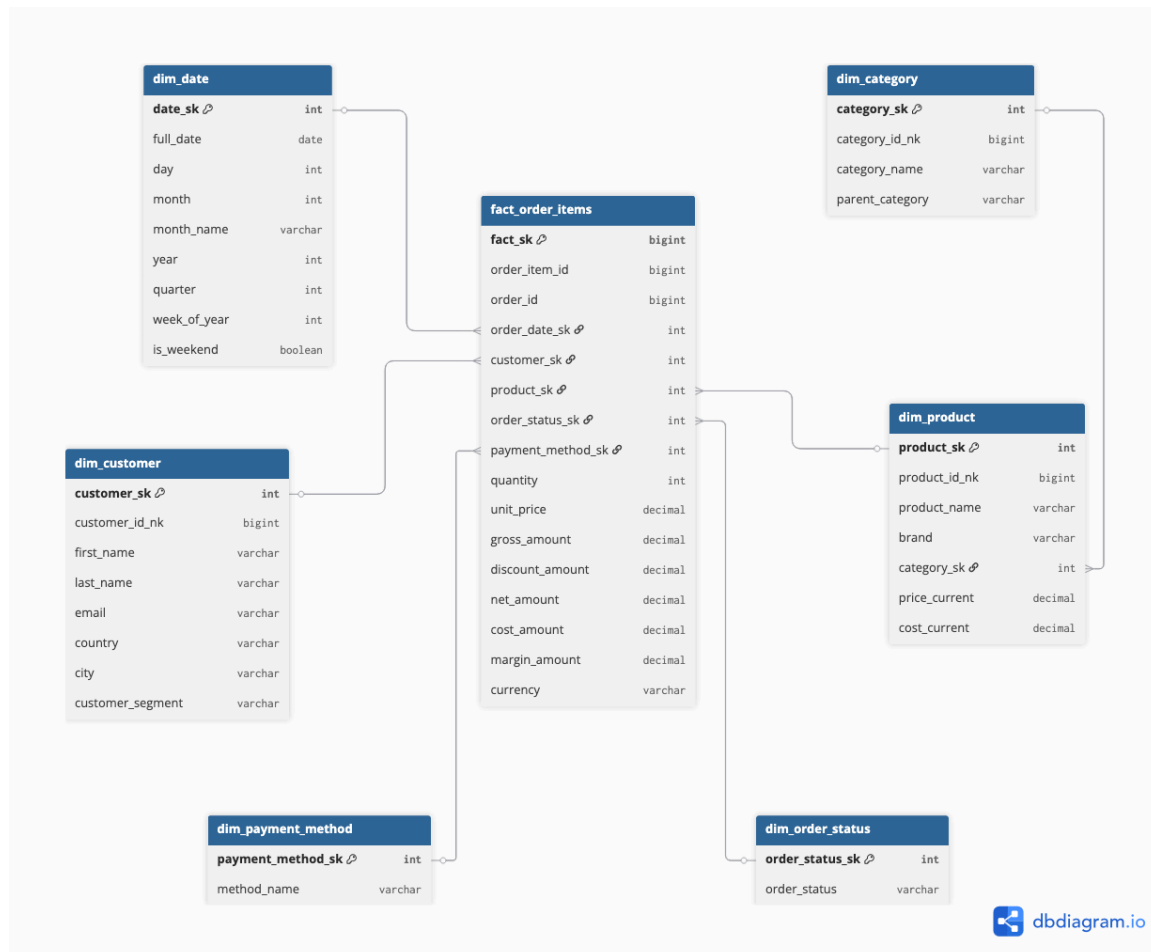
Se diseñó un Data Mart orientado a ventas, siguiendo un modelo dimensional en estrella, donde la tabla de hechos principal es fact_order_items (una fila por ítem de pedido).

La razones de este diseño son:

- Centralizar métricas de ventas en una única tabla de hechos.
- Permitir análisis por múltiples perspectivas (producto, categoría, cliente, tiempo, estado del pedido, método de pago).
- Definir la granularidad al nivel más bajo (detalle de ítem de pedido) para mantener flexibilidad analítica.

4. Diagramas del modelo dimensional

La granularidad definida es una fila por ítem de pedido (order_item_id).



5. Proceso ETL (Pentaho)

5.1 ¿Qué es un ETL?

Un ETL (Extract, Transform, Load) es un proceso que:

- Extrae datos desde diversas fuentes (archivos CSV, bases transaccionales, APIs).
- Transforma los datos para limpiarlos, enriquecerlos y adaptarlos al modelo destino.
- Carga los datos en un Data Warehouse o Data Mart, optimizados para análisis.

5.2 ETL en este proyecto

Se utilizó Pentaho Data Integration (PDI):

- Transformaciones de staging para normalizar y limpiar los datos.
- Uso de pasos como Text File Input, Replace in String, Strings Cut para corregir formatos de fechas.
- Database Lookup para asignar llaves sustitutas (SK) desde las dimensiones.
- Table Output para cargar los hechos en el Data Mart.

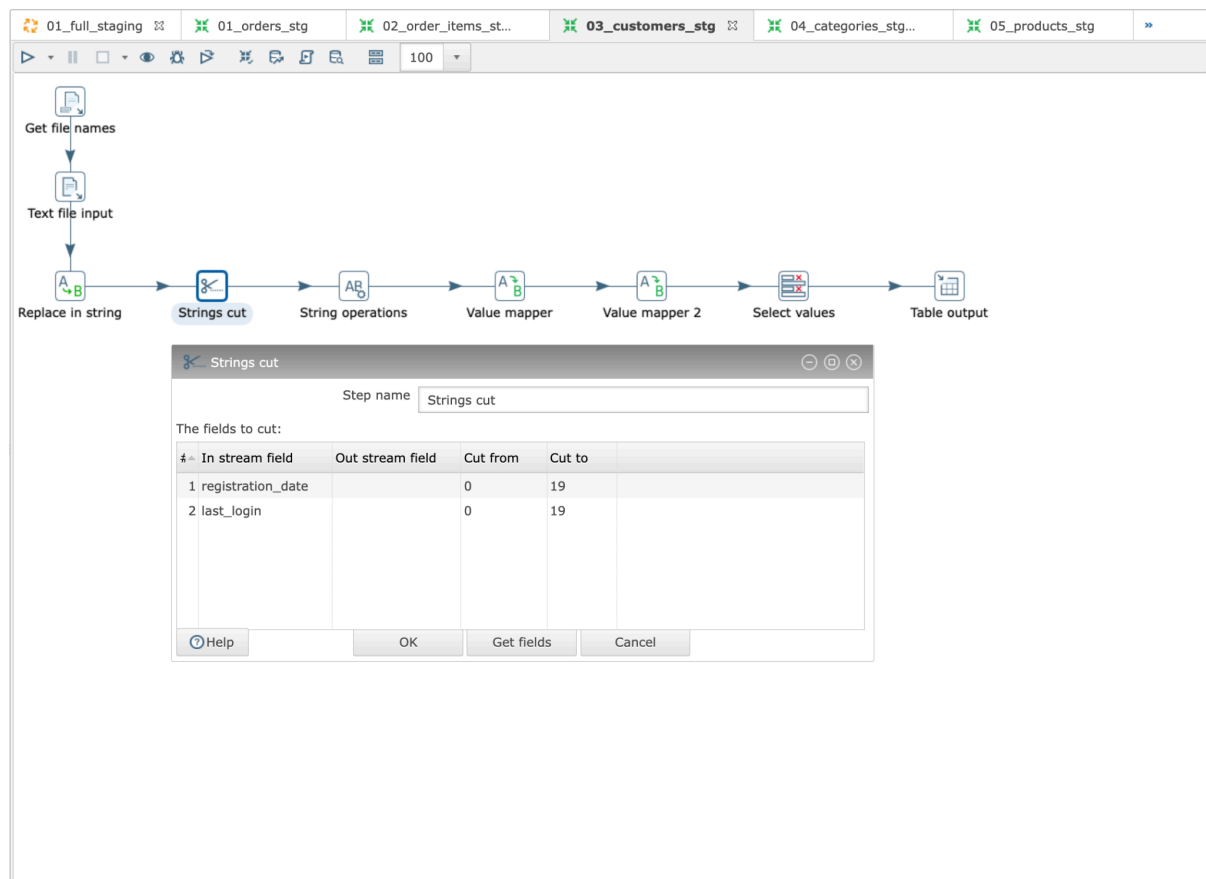
El proceso ETL fue implementado en Pentaho Data Integration (PDI - Spoon). Consta de tres fases principales:

- Carga a staging: ingestión de archivos CSV originales a tablas temporales en la base de datos.
- Construcción de dimensiones: normalización y carga de las tablas dim_customer, dim_product, dim_category, dim_order_status, dim_payment_method y dim_date.
- Carga de hechos: integración de los datos de staging y dimensiones para poblar fact_order_items.

Cada transformación .ktr y job .kjb fue versionado en la carpeta ETL del proyecto. Se implementaron pasos de limpieza como la normalización de fechas (YYYYMMDD) para asegurar consistencia entre la tabla de hechos y la dimensión fecha.

Algunas capturas del proceso en pentaho.

- Por la estructura de las fechas en el dataset fue necesario usar “Replace in string” y “String cut” en cada para cada tabla que las tuviera.



- Por seguridad se utilizó “Value mapper” en los casos donde se trabajó con datos booleanos.

The screenshot shows a data processing workflow in a tool like Alteryx Designer. The workflow starts with 'Get file names', followed by 'Text file input', then a sequence of steps: 'Replace in string', 'Strings cut', 'String operations', 'Value mapper' (highlighted), 'Value mapper 2', 'Select values', and finally 'Table output'. A dialog box for the 'Value mapper' step is open, showing configuration options for mapping source values to a target field named 'is_active'.

Value mapper configuration:

- Step name: Value mapper
- Fieldname to use: is_active
- Target field name (empty=overwrite):
- Default upon non-matching: 0

Field values:

#	Source value	Target value
1	1	1
2	true	1
3	t	1
4	yes	1
5	y	1
6	si	1
7	sí	1
8	0	0
9	false	0
10	f	0
11	no	0

- En la construcción de las dimensiones se usó “Table input” para cargar los datos de las tablas creadas en staging.

The screenshot displays a data pipeline interface with a sequence of steps: Table input, Value mapper, Database lookup, Select values, and Insert / update. The 'Table input' step is selected, and its configuration window is open. The window shows the step name 'Table input', the connection 'dm_mysql_local', and a SQL query. The query selects various columns from 'dm_ecommerce.stg_products' and includes a CASE statement for 'created_at'. The configuration also includes options for storing column info, enabling lazy conversion, replacing variables, inserting data from step, and a limit size of 0.

Step name: Table input

Connection: dm_mysql_local

SQL:

```
SELECT
    product_id          AS product_id_nk,
    product_name,
    brand,
    category_id,
    price,
    cost,
    weight_kg,
    dimensions,
    description,
    is_active,
    CASE
        WHEN created_at IS NULL THEN NULL
        ELSE DATE_FORMAT(created_at, '%Y%m%d')+0
    END                AS created_date_sk
FROM dm_ecommerce.stg_products;
```

Line 1 Column 0

Store column info in step meta data ☐

Enable lazy conversion ☐

Replace variables in script? ☐

Insert data from step

Execute for each row? ☐

Limit size: 0

Buttons: Help, OK, Preview, Cancel

- Para especificar donde se guardaran los datos de las dimensiones se usó "Insert/Update".

02_dim_build 10_dim_customer_1... 11_dim_category_1... 12_dim_product_1o... 13_dim_order_stat...

Table input Value mapper Database lookup Select values Insert / update

Insert / update

Step name: Insert / update

Connection: dm_mysql_local Edit... New... Wizard...

Target schema: dm_ecommerce Browse...

Target table: dim_product Browse...

Commit size: 10000

Don't perform any updates: ☐

The key(s) to look up the value(s):

#	Table field	Comparator	Stream field1	Stream field2
1	product_id_nk	=	product_id_nk	

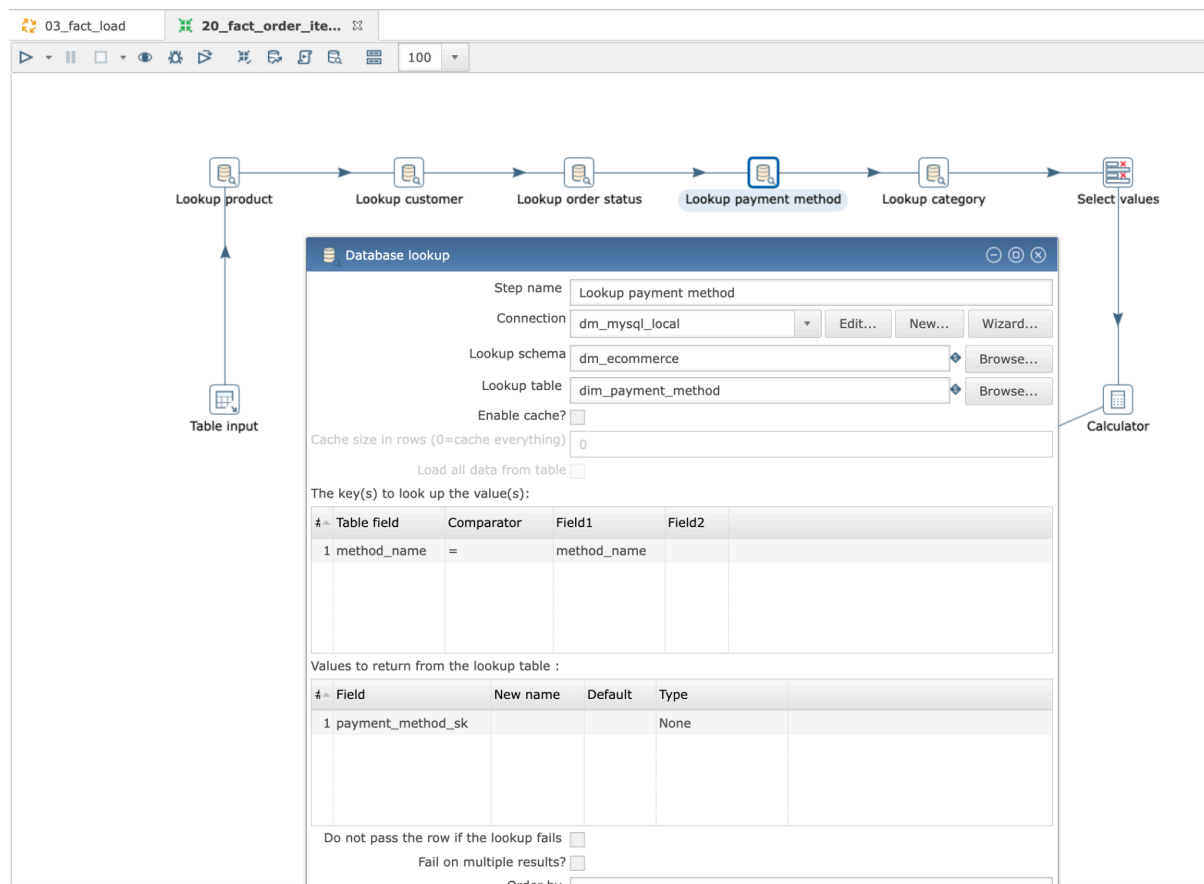
Get fields

Update fields:

#	Table field	Stream field	Update
1	product_id_nk	product_id_nk	Y
2	product_name	product_name	Y
3	brand	brand	Y
4	price_current	price_current	Y
5	cost_current	cost_current	Y
6	weight_kg	weight_kg	Y
7	dimensions	dimensions	Y
8	description	description	Y
9	is_active	is_active	Y

Get update fields Edit mapping

- Para cargar columnas de otras tablas se usó “Database Lookup”, la cual permite especificar qué campos necesitamos indicando las columnas por las que se relacionan.



6. Power BI

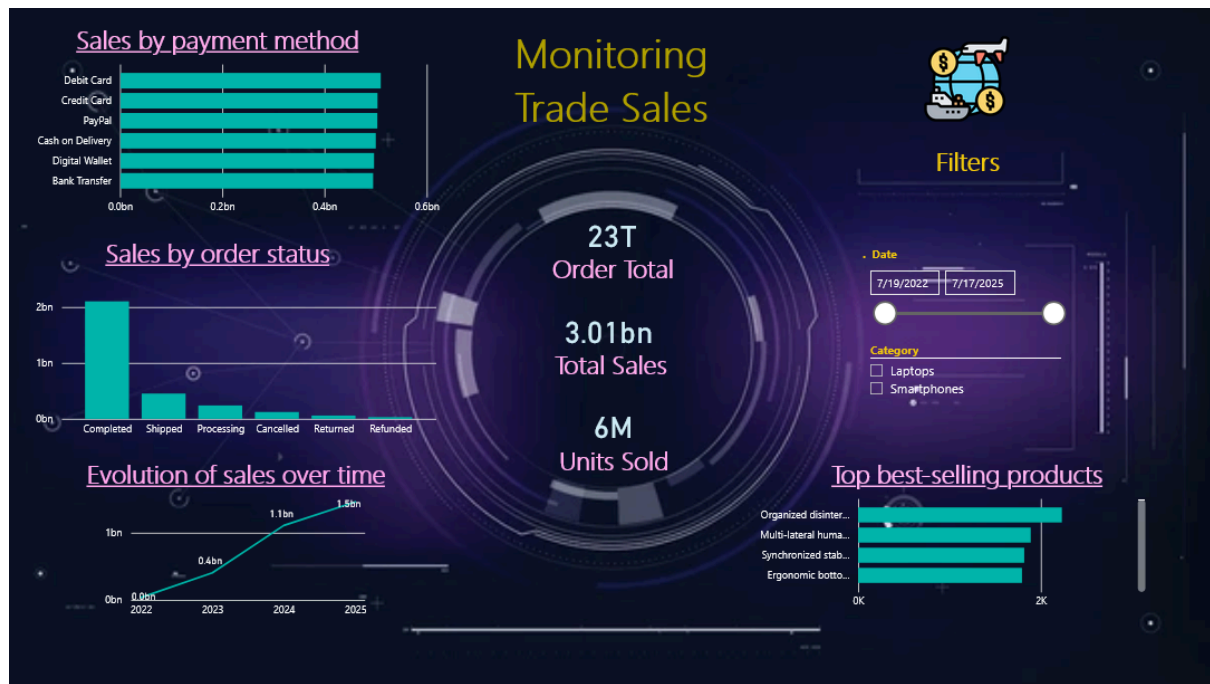
6.1 ¿Qué es Power BI?

Power BI es una herramienta de Microsoft para el análisis de datos y la construcción de dashboards interactivos. Permite a los usuarios finales explorar la información del Data Mart mediante visualizaciones gráficas dinámicas.

6.2 Dashboards creados

Se generaron reportes que permiten responder preguntas clave de negocio:

- Tendencia de ventas en el tiempo (gráfico de líneas).
- Ventas por categoría de producto (gráfico de barras).
- Estado de ventas realizadas (gráfico de barras).
- Métodos de pago más utilizados (gráfico circular).



Los indicadores se integraron en un único dashboard que resume la información clave y se puede filtrar de forma interactiva con los gráficos implementados y las opciones de filtros incluidas.

7. Tabla de requisitos del negocio

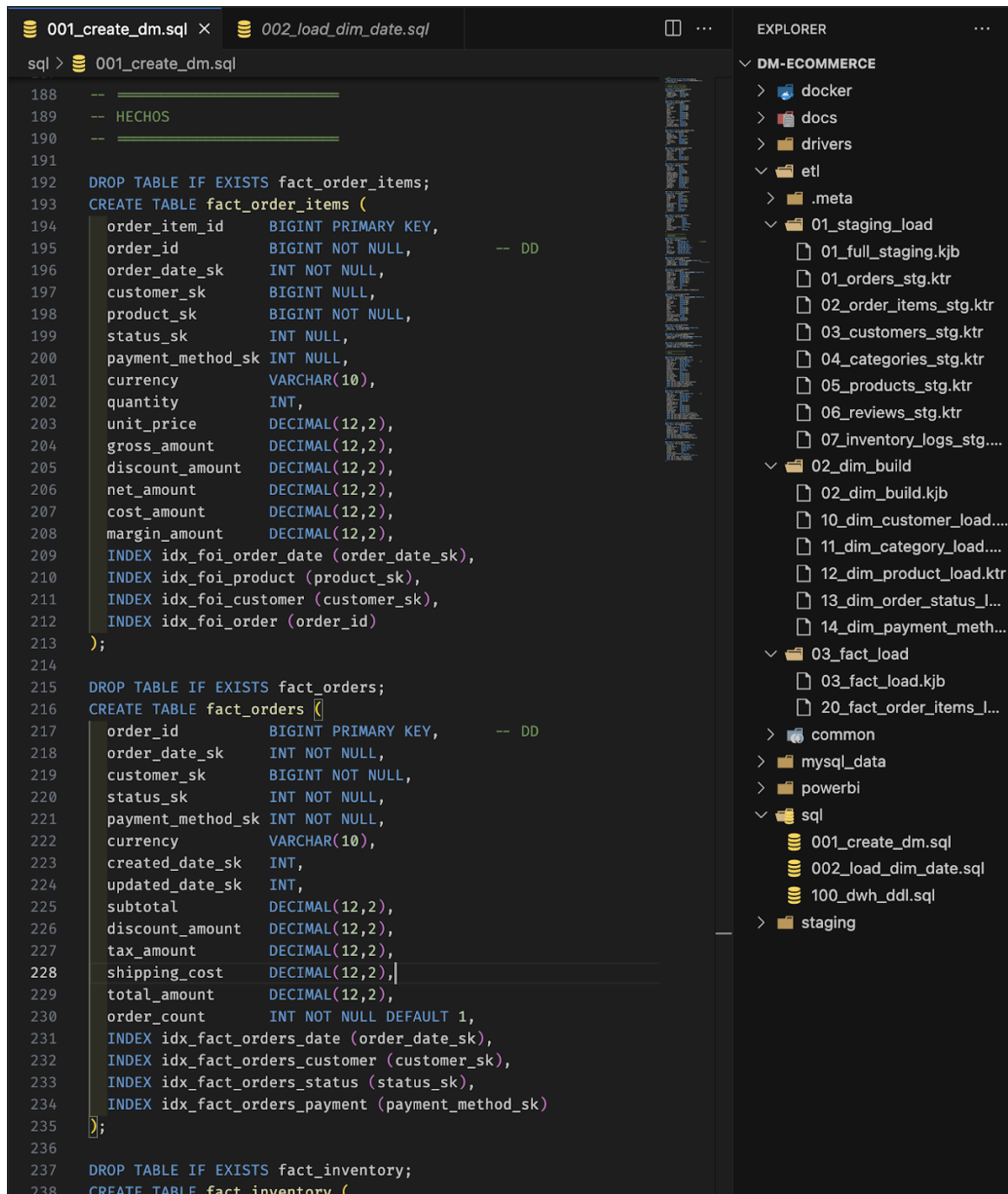
Indicador	Variables	Características	Proceso	Método	Visualización
Ventas netas	fact_order_items.quantity, fact_order_items.unit_price, fact_order_items.discount_amount, fact_order_items.net_amount	Cantidad monetaria (USD)	Proceso de venta	Ventas brutas = $\Sigma(\text{quantity} * \text{unit_price})$ Ventas netas = $\Sigma(\text{net_amount})$	Tarjeta KPI + tendencia (línea)
Ticket promedio (AOV)	fact_order_items.net_amount, fact_order_items.order_id	Ratio monetario	Proceso de venta	AOV = $\Sigma(\text{net_amount}) / \text{\#órdenes}$	Tarjeta + línea temporal
Margen bruto %	fact_order_items.net_amount, fact_order_items.cost_amount, fact_order_items.margin_amount	Porcentaje	Proceso de venta	Margen% = $(\Sigma(\text{margin_amount}) / \Sigma(\text{net_amount})) * 100$	Semáforo / barra por categoría
Tasa de recompra	fact_order_items.customer_sk, dim_customer.customer_id	Porcentaje de clientes	Proceso de venta	Clientes con ≥ 2 pedidos / Clientes totales (periodo)	Barra / línea
Rating promedio	fact_reviews.rating, fact_reviews.product_sk	Promedio 1–5	Postventa	Avg(rating) por producto/categoría	Gráfico de barras
Ítems con stock crítico	fact_inventory.stock_quantity,	Conteo / %	Inventario	% de productos con stock_quantity <	Tarjeta + barra

	dim_product.product_name, dim_category.category_name			umbral	
--	---	--	--	--------	--

8. Código y flujo del proceso ETL

Se incluyen ejemplos del código SQL y configuraciones en Pentaho para la creación de la tabla de hechos:

- Creación de tablas de dimensiones y hechos.



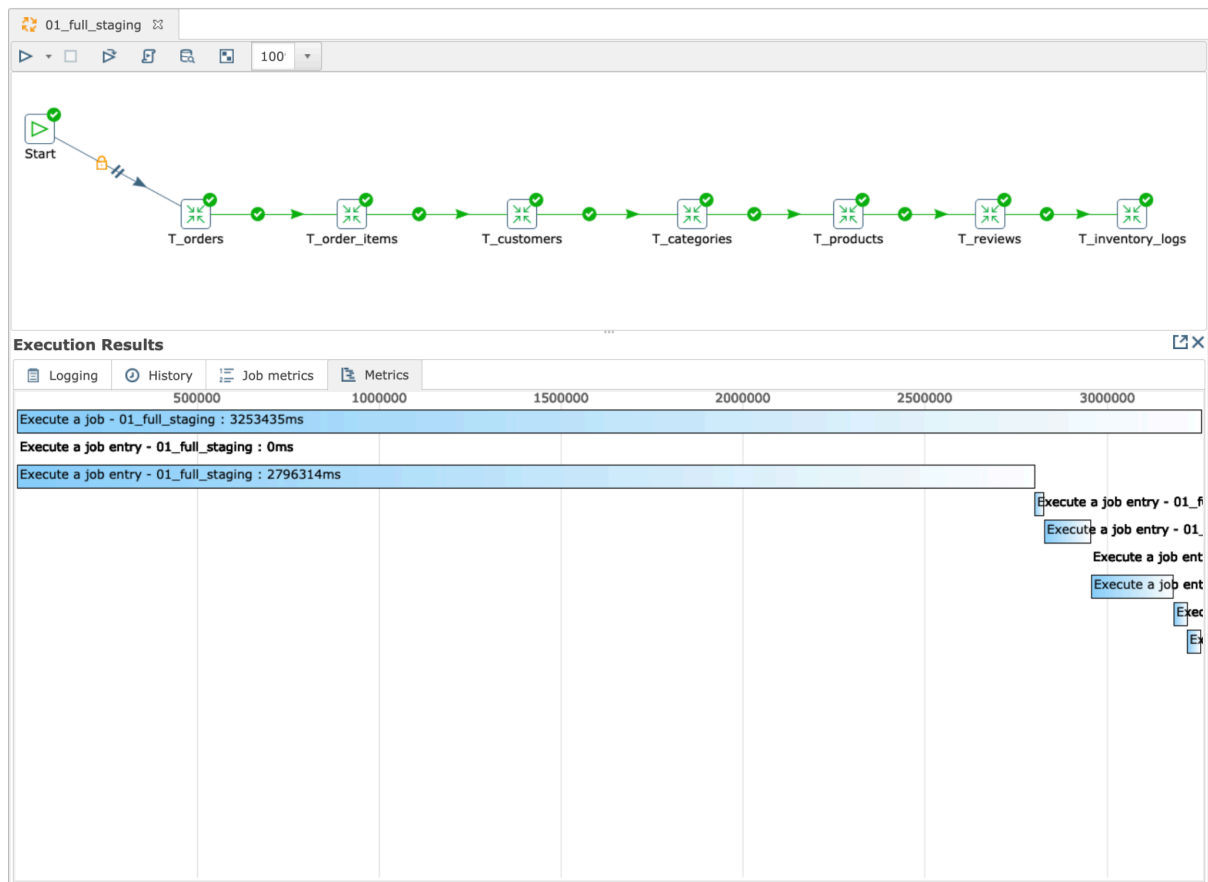
```
001_create_dm.sql | 002_load_dim_date.sql
sql > 001_create_dm.sql

188 -- =====
189 -- HECHOS
190 -- =====
191
192 DROP TABLE IF EXISTS fact_order_items;
193 CREATE TABLE fact_order_items (
194     order_item_id BIGINT PRIMARY KEY,
195     order_id BIGINT NOT NULL, -- DD
196     order_date_sk INT NOT NULL,
197     customer_sk BIGINT NULL,
198     product_sk BIGINT NOT NULL,
199     status_sk INT NULL,
200     payment_method_sk INT NULL,
201     currency VARCHAR(10),
202     quantity INT,
203     unit_price DECIMAL(12,2),
204     gross_amount DECIMAL(12,2),
205     discount_amount DECIMAL(12,2),
206     net_amount DECIMAL(12,2),
207     cost_amount DECIMAL(12,2),
208     margin_amount DECIMAL(12,2),
209     INDEX idx_foi_order_date (order_date_sk),
210     INDEX idx_foi_product (product_sk),
211     INDEX idx_foi_customer (customer_sk),
212     INDEX idx_foi_order (order_id)
213 );
214
215 DROP TABLE IF EXISTS fact_orders;
216 CREATE TABLE fact_orders (
217     order_id BIGINT PRIMARY KEY, -- DD
218     order_date_sk INT NOT NULL,
219     customer_sk BIGINT NOT NULL,
220     status_sk INT NOT NULL,
221     payment_method_sk INT NOT NULL,
222     currency VARCHAR(10),
223     created_date_sk INT,
224     updated_date_sk INT,
225     subtotal DECIMAL(12,2),
226     discount_amount DECIMAL(12,2),
227     tax_amount DECIMAL(12,2),
228     shipping_cost DECIMAL(12,2),
229     total_amount DECIMAL(12,2),
230     order_count INT NOT NULL DEFAULT 1,
231     INDEX idx_fact_orders_date (order_date_sk),
232     INDEX idx_fact_orders_customer (customer_sk),
233     INDEX idx_fact_orders_status (status_sk),
234     INDEX idx_fact_orders_payment (payment_method_sk)
235 );
236
237 DROP TABLE IF EXISTS fact_inventory;
238 CREATE TABLE fact_inventory (
```

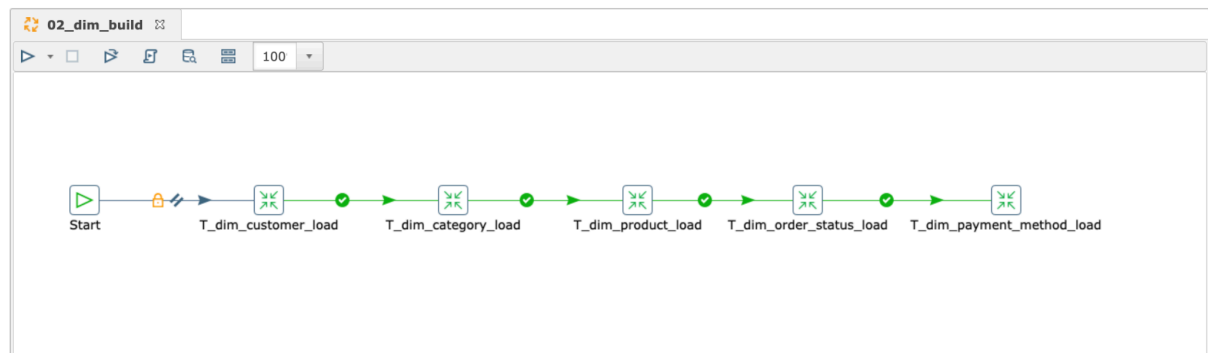
EXPLORER

- DM-ECOMMERCE
 - docker
 - docs
 - drivers
 - etl
 - .meta
 - 01_staging_load
 - 01_full_staging.kjb
 - 01_orders_stg.ktr
 - 02_order_items_stg.ktr
 - 03_customers_stg.ktr
 - 04_categories_stg.ktr
 - 05_products_stg.ktr
 - 06_reviews_stg.ktr
 - 07_inventory_logs_stg....
 - 02_dim_build
 - 02_dim_build.kjb
 - 10_dim_customer_load....
 - 11_dim_category_load....
 - 12_dim_product_load.ktr
 - 13_dim_order_status_l...
 - 14_dim_payment_meth...
 - 03_fact_load
 - 03_fact_load.kjb
 - 20_fact_order_items_l...
 - common
 - mysql_data
 - powerbi
 - sql
 - 001_create_dm.sql
 - 002_load_dim_date.sql
 - 100_dwh_ddl.sql
 - staging

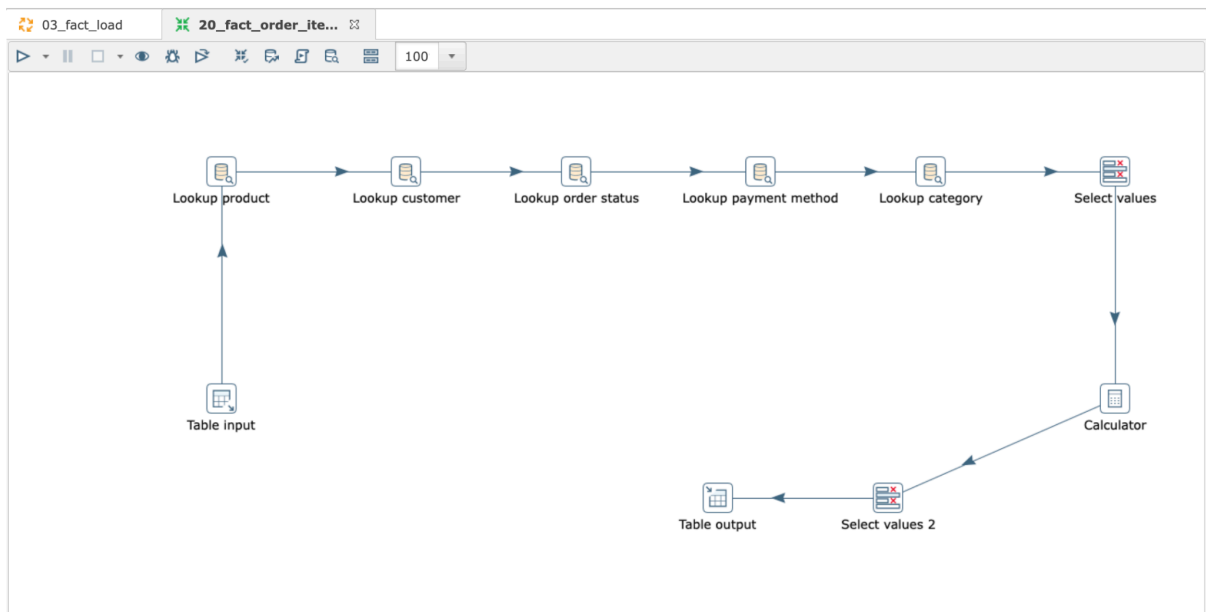
- Carga a staging.



- Construcción de dimensiones.



- Carga de hechos.



El código SQL (creación de Data Mart, carga de dimensiones, vistas finales) y las transformaciones Pentaho (.ktr, .kjb) se organizaron en carpetas según su etapa:

- /etl/01_staging_load: Cargas a staging desde CSV.
- /etl/02_dim_build: Construcción de dimensiones.
- /etl/03_fact_load: Carga de hechos.
- /sql: scripts SQL de creación, carga y vistas.

9. Conclusiones

- Se construyó un Data Mart con granularidad a nivel de ítem de pedido.
- El proceso ETL con Pentaho permitió limpiar y transformar los datos de staging a un esquema en estrella.
- Con Power BI se generaron dashboards interactivos que cumplen con los requisitos de indicadores del negocio.
- El modelo permite escalabilidad para añadir nuevos hechos como inventario o reseñas.