Caso: NovaShop – Glue Crawler + ETL + Athena

Objetivo: Usar un Crawler de AWS Glue para catalogar dos archivos CSV (productos y ventas), crear un Job de ETL (PySpark) que genere una sola tabla en formato Parquet y consultar con Athena para responder 10 preguntas de analítica.

Datos entregados

1) novashop_products.csv

- product_id (INT), product_name (STRING), category (STRING), brand (STRING), unit_price (DECIMAL), unit_cost (DECIMAL)

2) novashop_sales.csv

- order_id (INT), order_date (DATE en formato YYYY-MM-DD), store_id (STRING), city (STRING), customer_id (INT), product_id (INT), qty (INT), unit_price (DECIMAL), discount_pct (DECIMAL), payment_method (STRING), channel (STRING)

Estructura recomendada en S3

s3://<tu-bucket>/novashop/raw/products/novashop_products.csv

s3://<tu-bucket>/novashop/raw/sales/novashop_sales.csv

s3://<tu-bucket>/novashop/curated/sales_merged/ (salida Parquet particionada por año/mes)

Pasos

1) Crear una base de datos en Glue Data Catalog, por ejemplo: novashop_db.

2) Crear dos Crawlers en Glue (o uno con múltiples paths) apuntando a las rutas RAW; base de datos objetivo: novashop_db.

3) Ejecutar los Crawlers y verificar que se creen las tablas externas products_raw y sales_raw.

4) Crear un Job de Glue (Spark) que lea las tablas RAW, haga el join y escriba Parquet particionado por año/mes.

5) En Athena, crear una tabla externa sobre la capa CURATED (o usar CTAS) y ejecutar las consultas.

Script de ejemplo (Glue PySpark)

```
# Glue ETL de ejemplo (simplificado)
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, year, month, round as sround, expr
```

```
spark = SparkSession.builder.getOrCreate()

products = spark.read.option("header", True).option("inferSchema", True) \
   .csv("s3://<tu-bucket>/novashop/raw/products/")

sales = spark.read.option("header", True).option("inferSchema", True) \
   .csv("s3://<tu-bucket>/novashop/raw/sales/")

joined = sales.join(products, on="product_id", how="left")

# Métricas calculadas
with_metrics = joined.withColumn("subtotal", col("qty") * col("unit_price")) \
   .withColumn("discount_amount", sround(col("subtotal") * col("discount_pct"), 2)) \
   .withColumn("total", sround(col("subtotal") - col("discount_amount"), 2)) \
   .withColumn("cogs", sround(col("qty") * col("unit_cost"), 2)) \
   .withColumn("profit", sround(col("total") - col("cogs"), 2)) \
   .withColumn("year", year(col("order_date"))) \
   .withColumn("month", month(col("order_date")))

# Escribir CURATED en Parquet, particionado por año/mes
with_metrics.write.mode("overwrite").partitionBy("year", "month") \
   .parquet("s3://<tu-bucket>/novashop/curated/sales_merged/")
```

DDL ejemplo en Athena (tabla CURATED)

```
CREATE EXTERNAL TABLE IF NOT EXISTS novashop_db.sales_curated (
  order_id bigint,
  order_date date,
  store_id string,
  city string,
  customer_id bigint,
  product_id bigint,
  qty int,
  unit_price double,
  discount_pct double,
  payment_method string,
  channel string,
  product_name string,
  category string,
  brand string,
  unit_price_catalog double,
```

```
  unit_cost double,
  subtotal double,
  discount_amount double,
  total double,
  cogs double,
  profit double
)
PARTITIONED BY (year int, month int)
STORED AS PARQUET
LOCATION 's3://<tu-bucket>/novashop/curated/sales_merged/'
TBLPROPERTIES ('parquet.compression'='SNAPPY');
```

Nota: Si tu columna unit_price del catálogo se llama igual que en ventas, puedes renombrar la del catálogo a unit_price_catalog en el ETL para evitar colisiones.

10 preguntas para resolver con Athena (con guía de consulta)

1) Ventas totales y margen bruto (suma de total y de profit) del periodo.

```
SELECT SUM(total) AS ventas_totales, SUM(profit) AS margen_bruto FROM novashop_db.sales_curated;
```

2) Ticket promedio (promedio de total por orden).

```
SELECT AVG(total) AS ticket_promedio FROM novashop_db.sales_curated;
```

3) Top 5 categorías por ventas.

```
SELECT category, SUM(total) AS ventas FROM novashop_db.sales_curated GROUP BY category ORDER BY ventas DESC LIMIT 5;
```

4) Tendencia mensual de ventas y margen.

```
SELECT year, month, SUM(total) AS ventas, SUM(profit) AS margen FROM novashop_db.sales_curated GROUP BY year, month ORDER BY year, month;
```

5) Ciudades con mejor rendimiento por ventas.

```
SELECT city, SUM(total) AS ventas FROM novashop_db.sales_curated GROUP BY city ORDER BY ventas DESC;
```

6) Marca con mayor contribución a la utilidad.

```
SELECT brand, SUM(profit) AS utilidad FROM novashop_db.sales_curated GROUP BY brand ORDER BY utilidad DESC;
```

7) Método de pago: distribución de ventas y conteo de órdenes.

SELECT payment_method, COUNT(DISTINCT order_id) AS ordenes, SUM(total) AS ventas FROM novashop_db.sales_curated GROUP BY payment_method ORDER BY ventas DESC;

8) Canal (Tienda vs Online): comparación de ventas, utilidad y ticket promedio.

SELECT channel, SUM(total) AS ventas, SUM(profit) AS utilidad, AVG(total) AS ticket_promedio FROM novashop_db.sales_curated GROUP BY channel;

9) Clientes únicos y tasa de recompra (clientes con más de 1 orden).

WITH c AS (SELECT customer_id, COUNT(DISTINCT order_id) AS n FROM novashop_db.sales_curated GROUP BY customer_id) SELECT COUNT(*) AS clientes_unicos, SUM(CASE WHEN n>1 THEN 1 ELSE 0 END) AS clientes_recompra, ROUND(100.0*SUM(CASE WHEN n>1 THEN 1 ELSE 0 END)/COUNT(*),2) AS pct_recompra FROM c;

10) Descuento total aplicado y su efecto en el margen (profit/ventas por nivel de descuento).

SELECT discount_pct, SUM(discount_amount) AS descuento_total, SUM(total) AS ventas, SUM(profit) AS margen, ROUND(100.0*SUM(profit)/NULLIF(SUM(total),0),2) AS margen_pct FROM novashop_db.sales_curated GROUP BY discount_pct ORDER BY discount_pct;

Entrega esperada

- Evidencia de la base de datos y tablas en Glue Data Catalog (capturas).
- Código del Job de Glue (archivo .py) y ruta de salida en S3.
- DDL/CTAS de Athena y resultados de las 10 consultas.
- Conclusiones: 3 insights accionables a partir del análisis.