



PROYECTO FINAL:

“BASE DE DATOS DE TIENDA ONLINE PARA VENTAS EN MARKETPLACE”

Alumno: Luciano Di Naso

Institución: Coderhouse

Curso: SQL

Comisión: 9780

Tutor: Jennifer Goldfeld

Profesora: Lucia Blanc

INDICE:

Instrucciones de navegación:

Con CTRL + click sobre Pág, va al título deseado.

Para regresar al índice, hay un botón arriba a la derecha de cada página, con CTRL + click sobre el botón 'ir al índice'.

1. Descripción de la temática.....	Pág. 3
2. Objetivo del trabajo.....	Pág. 3
3. Descripción de las tablas.....	Pág. 4
<ul style="list-style-type: none">• Ventas• Compradores• Estados• Productos• Costos• Costos comercialización• Publicación• _COMPRADORES_UPDATE_AUD• _VENTAS_INSERT_AUD	
4. Diagrama de Entidad Relación.....	Pág. 7
5. Vistas.....	Pág. 8
6. Funciones.....	Pág. 12
7. Stored Procedures.....	Pág. 14
8. Triggers.....	Pág. 17
9. Informes.....	Pág. 19
10. Futuras líneas.....	Pág. 21
11. Script Final.....	Pág. 21
12. Herramientas utilizadas.....	Pág. 22

1. Descripción de la temática

La temática seleccionada corresponde a una tienda online que utiliza como único canal de ventas un Marketplace.

La principal fuente de datos proviene del Marketplace, estos brindan muchos datos e informes, sobre la operatoria de la tienda online, la problemática es que los informes no son a medida de la necesidad del titular de la tienda y son estáticos.

2. Objetivo del trabajo

Para el presente trabajo, se busca crear métricas e informe a medidas que puedan responder las preguntas de negocios, sobre márgenes, comportamiento de las publicaciones y reputación.

Con la creación de una base de datos propias se busca utilizar la información brindada por el Marketplace, y que esta se pueda manipular de acuerdo a la necesidad propia, y así poder realizar informes y métricas a nuestra medida.

Se buscará generar un panel de control para el dueño a nivel global, donde pueda obtener informes con gráficas, donde pueda hacer un seguimiento de las ventas, la utilidad, el punto de equilibrio, monitorear los costos, ver si hay modificaciones mes a mes en costos y utilidad, mostrar los productos que más se venden mes a mes.

3. Descripción de las tablas

DESCRIPCIÓN:

NOMBRE DE LA TABLA:

TIPOS DE CLAVES:

Contiene las ventas realizadas.

VENTAS		
KEY	NOMBRE CAMPO	TIPO DE DATO
PK	Id_venta	INT
	cod_venta_market	INT
	fecha_venta	DATETIME
	cantidad	INT
	precio_venta	INT
FK	Id_costos_com	INT
FK	Id_compradores	INT
FK	Id_estado	INT
FK	Id_Productos	INT
FK	Id_Publicacion	INT

DESCRIPCIÓN:

NOMBRE DE LA TABLA:

TIPOS DE CLAVES:

Contiene los datos de los compradores.

COMPRADORES		
KEY	NOMBRE CAMPO	TIPO DE DATO
PK	Id_compradores	INT
	nombre	VARCHAR(25)
	apellido	VARCHAR(25)
	dni	INT
	ciudad	VARCHAR(25)
	provincia	VARCHAR(25)
	codigo_postal	int
	domicilio	VARCHAR(50)
	email	VARCHAR(50)
	telefono	INT

DESCRIPCIÓN:

NOMBRE DE LA TABLA:

TIPOS DE CLAVES:

Contiene el estado de la venta, pendiente o entregado.

ESTADOS		
KEY	NOMBRE CAMPO	TIPO DE DATO
PK	Id_estado	INT
	nombre_estado	VARCHAR(20)
	fecha_en_camino	DATETIME
	fecha_entregado	DATETIME

DESCRIPCIÓN:

Contiene de los productos que posee la tienda.

NOMBRE DE LA TABLA:

TIPOS DE CLAVES:

PRODUCTOS		
KEY	NOMBRE CAMPO	TIPO DE DATO
PK	Id_productos	INT
	nombre_producto	VARCHAR(50)
	descripcion	VARCHAR(100)
	cod_barras	INT
	variantes	VARCHAR(25)
FK	Id_costos	INT

DESCRIPCIÓN:

Contiene el costo de compra de los productos, hasta estar disponible en el almacén.

NOMBRE DE LA TABLA:

TIPOS DE CLAVES:

COSTOS		
KEY	NOMBRE CAMPO	TIPO DE DATO
PK	Id_costos	INT
	fecha	DATE
	precio_compra	INT
	costo_reposicion	INT
	proveedor	VARCHAR(50)

DESCRIPCIÓN:

Contiene el costo de compra de los productos, hasta estar disponible en el almacén.

NOMBRE DE LA TABLA:

TIPOS DE CLAVES:

COSTOS COMERCIALES		
KEY	NOMBRE CAMPO	TIPO DE DATO
PK	Id_costos_com	INT
	cargos_venta	INT
	costo_envio	INT

DESCRIPCIÓN:

Contiene los datos de las publicaciones

NOMBRE DE LA TABLA:

TIPOS DE CLAVES:

PUBLICACION		
KEY	NOMBRE CAMPO	TIPO DE DATO
PK	Id_publicacion	INT
	cod_publicacion_market	VARCHAR(25)
	titulo_publicacion	VARCHAR(100)
	tipo_publicacion	BOOLEANO
FK	Id_vistas	INT

DESCRIPCIÓN: Tabla de auditoría, contiene los datos de los compradores antes de ser actualizado el registro, contiene el detalle del usuario y hora de ingreso del registro.

NOMBRE DE LA TABLA:

TIPOS DE CLAVES:

_COMPRADORES_UPDATE_AUD		
KEY	NOMBRE CAMPO	TIPO DE DATO
PK	Id_compradores_aud	INT
	entidad	VARCHAR(25)
	nombre	VARCHAR(25)
	apellido	VARCHAR(25)
	dni	INT
	ciudad	VARCHAR(25)
	provincia	VARCHAR(25)
	codigo_postal	int
	domicilio	VARCHAR(50)
	email	VARCHAR(50)
	telefono	INT
	usuario	VARCHAR(50)
	fecha	DATETIME

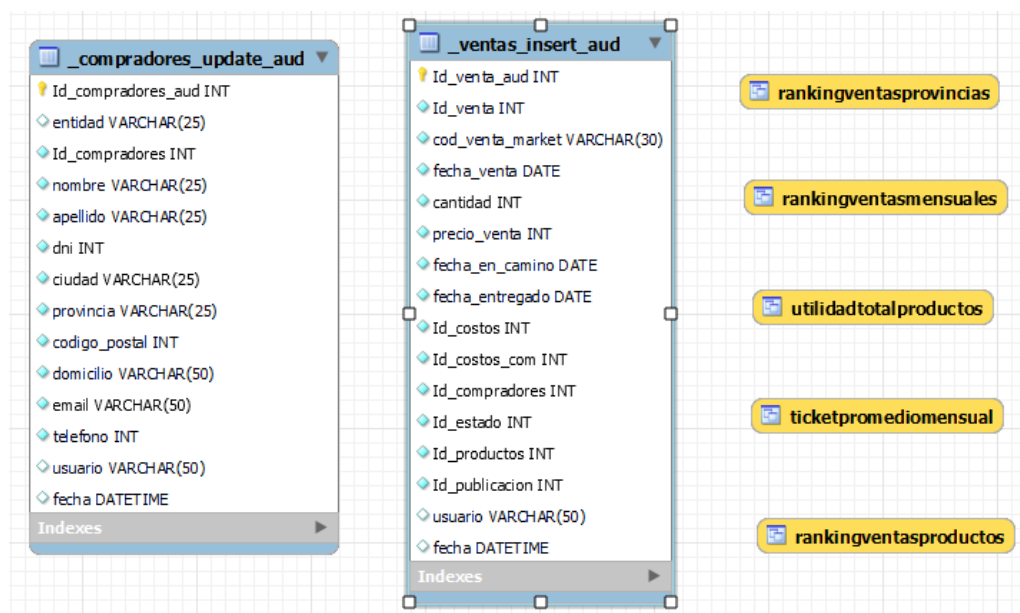
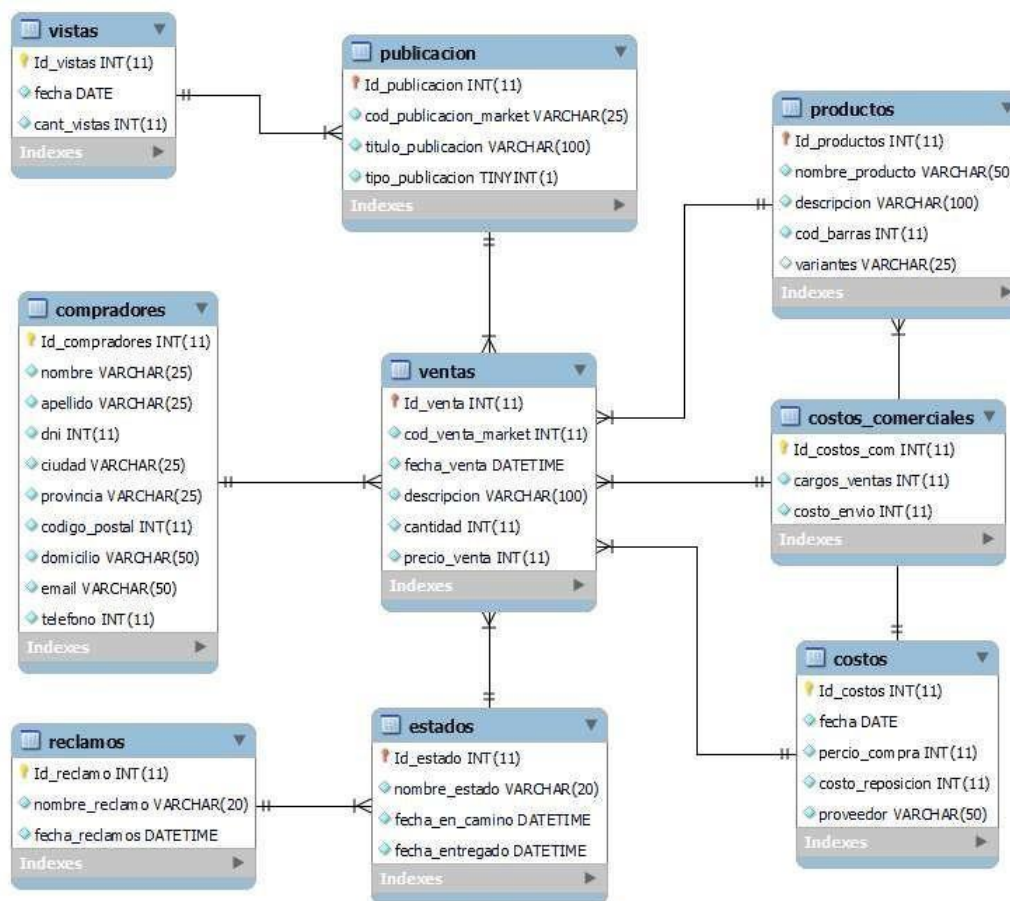
DESCRIPCIÓN: Tabla de auditoría, es una tabla espejo de los registros insertados en las ventas, contiene el detalle del usuario y hora de ingreso del registro.

NOMBRE DE LA TABLA:

TIPOS DE CLAVES:

_VENTA_INSERT_AUD		
KEY	NOMBRE CAMPO	TIPO DE DATO
PK	Id_venta	INT
	cod_venta_market	INT
	fecha_venta	DATETIME
	cantidad	INT
	precio_venta	INT
	Id_costos_com	INT
	Id_compradores	INT
	Id_estado	INT
	Id_Productos	INT
	Id_Publicacion	INT
	usuario	VARCHAR(50)
	fecha	DATETIME

4. Diagrama de Entidad Relación



5. Vistas

[Link a GitHub - Vistas](#)

Vista 1: 'rankingventasmensuales'

Descripción: La tabla muestra un detalle de ventas totales en pesos por mes.

Objetivo: Mostrar una tabla con el total de ventas en pesos por mes.

Tablas/Datos: Ventas

```
1 • SELECT * FROM desafio_sql_dinaso_triggers.rankingventasmensuales;
```

MES	TOTAL
January	3889648
February	4207791
March	276037

Vista 2: 'rankingventasproductos'

Descripción: Se muestra una tabla con el ranking de ventas totales en pesos por productos, ordenados de mayor a menor.

Objetivo: Ver el ranking de ventas en pesos por productos, ordenados de mayor a menor en ventas.

Tablas/Datos: productos.

```
1 • SELECT * FROM desafio_sql_dinaso_triggers.rankingventasproductos;
```

nombre_producto	Total_Ventas
Termo Classic 1.9 L	1478947
Termo Classic 1.4L	1243942
Termo Classic 1.1L	1217100
Termo Classic 1 L	983330
Termo Classic 0.87 L	875840
Vaso Stanley 354 Ml Camp Mug	618407
Jarra De Cerveza Adventure 700ml	573132
Vaso termico Stanley Classic 591m	569057

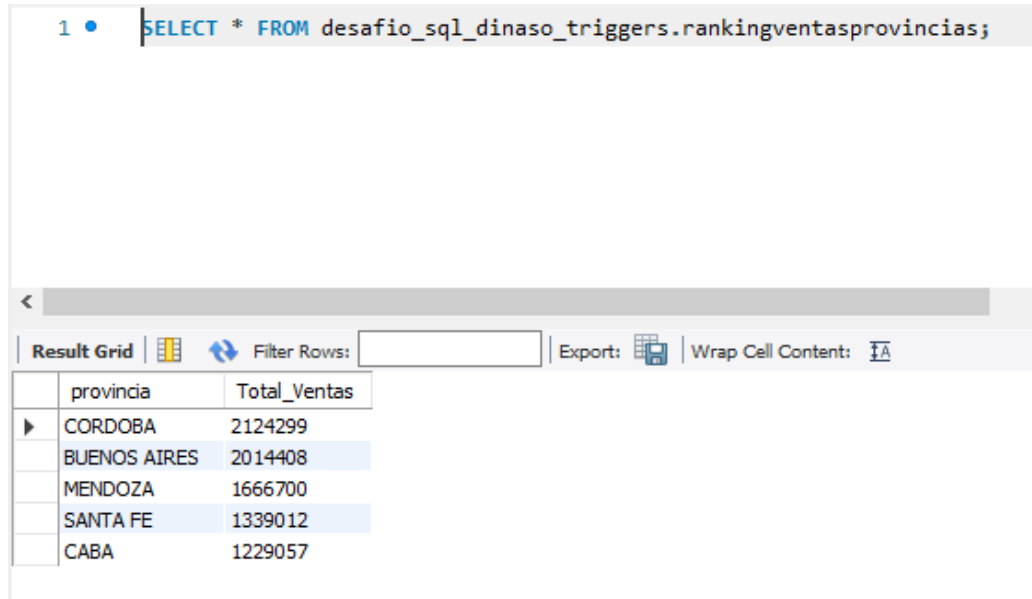
Vista 3: `rankingventasprovincias`

Descripción: Se muestra en una tabla las ventas totales en pesos por provincia ordenadas de mayor a menor en ventas.

Objetivo: Obtener el ranking de ventas en pesos por provincia ordenadas de mayor a menor.

Tablas/Datos: ventas y compradores.

```
1 • SELECT * FROM desafio_sql_dinaso_triggers rankingventasprovincias;
```



provincia	Total_Ventas
CORDOBA	2124299
BUENOS AIRES	2014408
MENDOZA	1666700
SANTA FE	1339012
CABA	1229057

Vista 4: `ticketpromediomensual`

Descripción: Se obtiene una tabla con el detalle mensual y el dato del ticket promedio de cada mes.

Objetivo: Obtener el ticket promedio de ventas por mes.

Tablas/Datos: Ventas

```
1 • SELECT * FROM desafio_sql_dinaso_triggers.ticketpromediomensual;
```

monthname(venta_fecha)	ticket_promedio
January	19448
February	22030
March	2509

Vista 5: 'utilidadtotalproductos'

Descripción: Se obtiene una tabla con la utilidad total en pesos obtenida por producto, ordenada de mayor a menor, para saber que producto obtuvo mayor ganancia.

Objetivo: El objetivo es tener un ranking de los productos según la utilidad que aportaron.

Tablas/Datos: Ventas, Productos y Costos.

```
1 • SELECT * FROM desafio_sql_dinaso_triggers.utilidadtotalproductos;
```

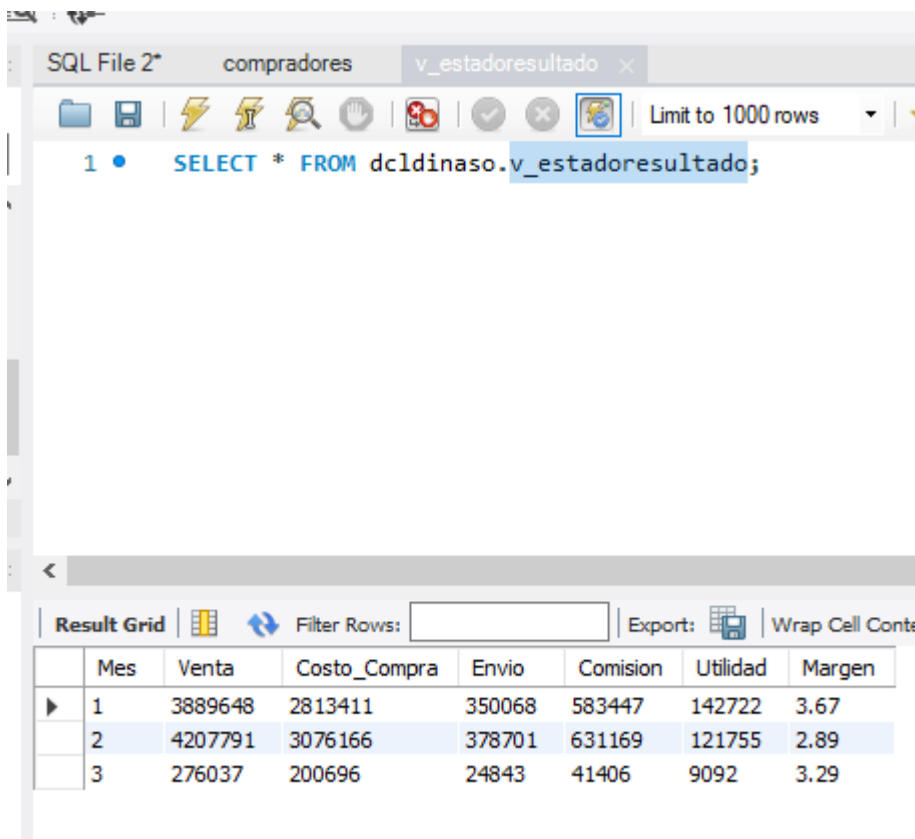
nombre_producto	Utilidad
Termo Classic 1.9 L	18192
Termo Classic 1.4L	14362
Termo Classic 1.1L	13405
Termo Classic 1 L	11490
Termo Classic 0.87 L	9575
Vaso Stanley 354 Ml Camp Mug	8820
Vaso termico Stanley Classic 591m	7717
Jarra De Cerveza Adventure 700ml	6063
Vaso Termico Stanley Starbucks	4851
Vaso termico Stanley Classic 887mL	4299

Vista 6: 'v_estadore resultado'

Descripción: Se obtiene una tabla con las, ventas, costos de compra, comisiones, envíos, utilidad total en pesos, y el margen de ganancia en tanto por 1, obtenida mes a mes, para saber la evolución del Estado de Resultado

Objetivo: El objetivo es obtener un Estado de Resultado y el Ratio de Margen de ganancia, mes a mes en una línea.

Tablas/Datos: Ventas, Costos.



The screenshot shows a SQL IDE window with a tab labeled 'v_estadore resultado'. The query editor contains the following SQL statement:

```
1 • SELECT * FROM dcldinaso.v_estadore resultado;
```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has 8 columns: Mes, Venta, Costo_Compra, Envio, Comision, Utilidad, and Margen. The results are as follows:

	Mes	Venta	Costo_Compra	Envio	Comision	Utilidad	Margen
▶	1	3889648	2813411	350068	583447	142722	3.67
	2	4207791	3076166	378701	631169	121755	2.89
	3	276037	200696	24843	41406	9092	3.29

6. Funciones

[Link a GitHub - Funciones](#)

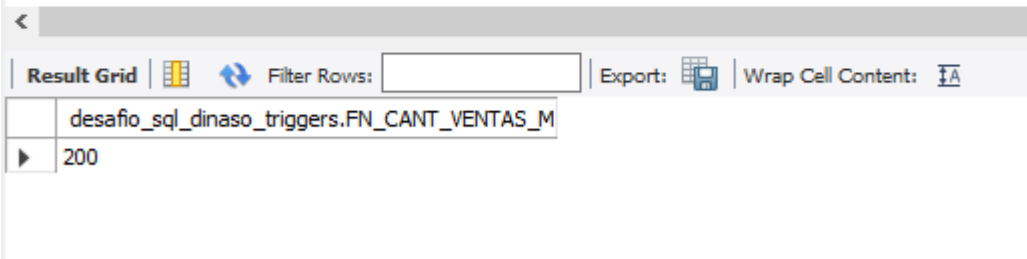
Función 1: `FN_CANT_VENTAS_MES`

Descripción: Se envía por parámetro el número del mes que se desea obtener la información, y devuelve la suma total de ventas en cantidades por mes.

Objetivo: El objetivo es obtener la suma total de ventas en cantidades de un mes determinado.

Tablas/Datos: Ventas

```
1 • select desafio_sql_dinaso_triggers.FN_CANT_VENTAS_MES('01');  
2
```



The screenshot shows a database interface with a query result grid. The query executed is `select desafio_sql_dinaso_triggers.FN_CANT_VENTAS_MES('01');`. The result grid displays a single row with the value 200.

desafio_sql_dinaso_triggers.FN_CANT_VENTAS_M
200

Función 2: `FN_TOTAL_PROVINCIA`

Descripción: Se envía por parámetro la provincia que se desea obtener la información y devuelve la suma total de ventas en cantidades por provincia.

Objetivo: El objetivo es obtener la suma total de ventas en cantidades según la provincia solicitada.

Tablas/Datos: Ventas

```
1 • select desafio_sql_dinaso_triggers.FN_TOTAL_PROVINCIA('santa fe');
2
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	desafio_sql_dinaso_triggers.FN_TOTAL_PROVINCIA('santa fe')
▶	104

Función 3: `FN TOTALVENTAS POR PRODUCTO`

Descripción: Se envía por parámetro el nombre del producto que se desea obtener la información y devuelve la suma total de ventas en pesos por producto.

Objetivo: El objetivo es obtener la suma total en pesos por producto.

Tablas/Datos: Ventas

```
1 • select desafio_sql_dinaso_triggers.FN_TOTALVENTAS_POR_PRODUCTO('Termo Classic 0.87 L');
2
3
4
```

Limit to 1000 rows | | |

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	desafio_sql_dinaso_triggers.FN_TOTALVENTAS_POR_PRODUCTO('Termo Classic 0.87 L')
▶	875840

7. Stores Procedures

[Link a GitHub - StoredProcedure](#)

Procedure 1: 'sp_get_compradores_order'

Descripción: Este procedimiento consiste en enviar por parámetro, el campo por el cual se desea ordenar y como segundo parámetro ASC/DESC, para determinar el tipo de orden ascendente o descendente.

Objetivo: El objetivo es que colocando el campo y tipo de orden asc/desc, se pueda obtener la tabla ordenada por el campo colocado, en el orden ascendente o descendente, según lo colocado.

Tablas/Datos: Compradores.

Ejemplo de ejecución:

```
call desafio_sql_dinaso_triggers.sp_get_compradores_order('nombre', 'asc');
```

Resultado: Se obtiene una tabla ordenada por el campo nombre el orden ascendente.

The screenshot shows a database client interface. At the top, there's a toolbar with icons for file operations, search, and execution. Below the toolbar, a command window shows the execution of the stored procedure: `call desafio_sql_dinaso_triggers.sp_get_compradores_order('nombre', 'asc');`. Below the command window, a table grid displays the results. The table has columns: Id_compradores, nombre, apellido, dni, ciudad, provincia, codigo_postal, domicilio, email, and telefono. The data is sorted by 'nombre' in ascending order.

Id_compradores	nombre	apellido	dni	ciudad	provincia	codigo_postal	domicilio	email	telefono
70	Nombre 0	Apellido 23	7428942	CORDOBA	CORDOBA	5000	CORDOBA16513840	16513840@hotmail.com	16513840
117	Nombre 0	Apellido 56	16206173	SANTA FE	SANTA FE	2000	SANTA FE62206396	62206396@hotmail.com	62206396
428	Nombre 0	Apellido 4	14672617	SANTA FE	SANTA FE	2000	SANTA FE13174609	13174609@hotmail.com	13174609
69	Nombre 0	Apellido 19	30529041	CORDOBA	CORDOBA	5000	CORDOBA70089361	70089361@hotmail.com	70089361
412	Nombre 0	Apellido 31	20152829	SANTA FE	SANTA FE	2000	SANTA FE53443184	53443184@hotmail.com	53443184
145	Nombre 0	Apellido 50	79125301	CABA	CABA	1000	CABA28220763	28220763@hotmail.com	28220763
364	Nombre 0	Apellido 68	27767267	CORDOBA	CORDOBA	5000	CORDOBA61963883	61963883@hotmail.com	61963883
327	Nombre 0	Apellido 15	1991338	MENDOZA	MENDOZA	5500	MENDOZA80646136	80646136@hotmail.com	80646136
252	Nombre 0	Apellido 43	38966374	BUENOS AIRES	BUENOS AIRES	1600	BUENOS AIRES52253322	52253322@hotmail.com	52253322
495	Nombre 0	Apellido 22	59299738	BUENOS AIRES	BUENOS AIRES	1600	BUENOS AIRES63797492	63797492@hotmail.com	63797492
31	Nombre 0	Apellido 5	19667750	MFNDO7A	MFNDO7A	5500	MFNDO7A88765839	88765839@hotmail.com	88765839

Procedure 2: sp_insert_products

Descripción: Este procedimiento realiza un control al insertar datos en una tabla, sino se envía el parámetro de nombre o variantes vacíos, no ingresa el registro, En todos los casos devuelve en una variable el resultado del procedimiento.

Objetivo: Controla que, al insertar un producto, se ingrese el nombre y la variante.

Tablas/Datos: Productos

Ejemplo de ejecución: (Ingreso de producto, campo de nombre se envía vacío)

```
set @mensaje = '0';
call desafio_sql_dinaso_triggers.sp_insert_products(' ', 'termin', '234324', 'Azul', @mensaje);
select @mensaje;
```

Resultado: Se llama al stored rocedures, y al enviarse un parámetro vacío para el campo nombre, no permite ingresar el producto, y devuelve un mensaje de error.

The screenshot displays a SQL client window with a toolbar at the top. The main area contains the following SQL script:

```
1 • set @mensaje = '0';
2 • call desafio_sql_dinaso_triggers.sp_insert_products(' ', 'termin', '234324', 'Azul', @mensaje);
3 • select @mensaje;
4
```

Below the script, the 'Result Grid' is visible. It shows a single row with the column name '@mensaje' and the value 'No se puede agregar el producto, sin nombre'.



@mensaje
No se puede agregar el producto, sin nombre

Ejemplo de ejecución:

```
set @mensaje = '0';  
call desafio_sql_dinaso_triggers.sp_insert_products('Termo ', 'tres', '234324', 'Azul', @mensaje);  
select @mensaje;
```

Resultado: Se inserta correctamente el producto en la tabla productos, y devuelve un mensaje diciendo que se agregó exitosamente el producto.

```
1 • set @mensaje = '0';  
2 • call desafio_sql_dinaso_triggers.sp_insert_products(' Termo ', 'termin', '234324', 'Azul', @mensaje);  
3 • select @mensaje;  
4
```

<	
Result Grid	Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 
@mensaje	
▶	Producto agregado exitosamente

8. Triggers

[Link a GitHub - Triggers](#)

Trigger 1: 'tr_ventas_ai_aud'

Descripción: Este trigger corresponde a la tabla de ventas, al suceder la inserción de un registro en esta tabla, se ejecuta el trigger, este se ejecutan después de la inserción de los datos y realiza una inserción de datos en una nueva tabla de auditoria llamada '_ventas_insert_aud', donde se replican los datos ingresados y además se registra el usuario que realizo la operación, la fecha y hora.

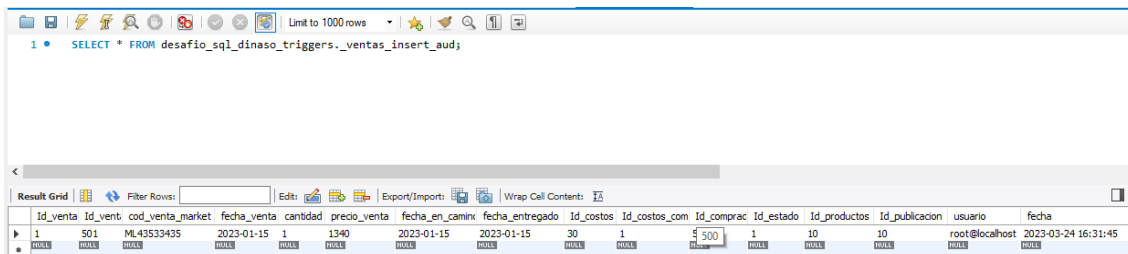
Objetivo: El objetivo es obtener una tabla espejo, que sirva de resguardo de la tabla más importante de la BD, además obtener una trazabilidad de los usuarios que han insertado los datos.

Tablas/Datos: Ventas y '_ventas_insert_aud'

Ejemplo de ejecución:

```
INSERT INTO ventas
VALUES ( 501, "ML43533435", '2023-01-15',1,1340,'2023-01-15','2023-01-15',30,1,500,1,10,10
);
```

Resultado: Obtenemos como resultado, en la nueva tabla, registros autogenerada a partir del trigger:



Id_venta	Id_venti	cod_venta_market	fecha_venta	cantidad	precio_venta	fecha_en_camin	fecha_entregado	Id_costos	Id_costos_com	Id_comprac	Id_estado	Id_productos	Id_publicacion	usuario	fecha
1	501	ML43533435	2023-01-15	1	1340	2023-01-15	2023-01-15	30	1	500	1	10	10	root@localhost	2023-03-24 16:31:45

Trigger 2: “tr_compradores_bu_aud”

Descripción: Este trigger corresponde a la tabla de compradores, al suceder la actualización de un registro en esta tabla, se ejecuta el trigger, este se ejecuta antes de la inserción de los datos y realiza una inserción de datos en una nueva tabla de auditoria llamada ‘_compradores_update_aud’, donde se registran los datos antes del ingreso de la actualización, para tener un historial de los datos de los compradores, esto sirve para casos que se desee recuperar datos anteriores a la actualización del registro y además se registra el usuario que realizo la operación, la fecha y hora.

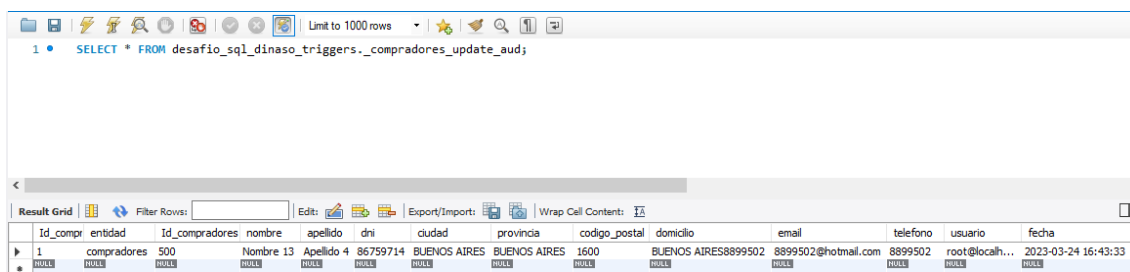
Objetivo: El objetivo es obtener una tabla historial de los cambios que se efectuaron sobre la tabla compradores, que sirva para volver a un paso anterior en caso de ser necesario, además obtener una trazabilidad de los usuarios que han actualizado los datos.

Tablas/Datos: Compradores y ‘_compradores_update_aud’

Ejemplo de ejecución:

```
UPDATE compradores
SET nombre = "Pepe", apellido = "Mujica"
WHERE Id_compradores = 500;
```

Resultado: Obtenemos como resultado, en la nueva tabla, registros autogenerada a partir del trigger:



The screenshot shows a database client interface. At the top, a SQL query is entered: `SELECT * FROM desafio_sql_dinaso_triggers._compradores_update_aud;`. Below the query, a 'Result Grid' displays the results of the query. The grid has 13 columns: `Id_compr`, `entidad`, `Id_compradores`, `nombre`, `apellido`, `dni`, `ciudad`, `provincia`, `codigo_postal`, `domicilio`, `email`, `telefono`, `usuario`, and `fecha`. The first row of data shows the following values: 1, compradores, 500, Nombre 13, Apellido 4, 86759714, BUENOS AIRES, BUENOS AIRES, 1600, BUENOS AIRES8899502, 8899502@hotmail.com, 8899502, root@localh..., and 2023-03-24 16:43:33.

	Id_compr	entidad	Id_compradores	nombre	apellido	dni	ciudad	provincia	codigo_postal	domicilio	email	telefono	usuario	fecha
1	1	compradores	500	Nombre 13	Apellido 4	86759714	BUENOS AIRES	BUENOS AIRES	1600	BUENOS AIRES8899502	8899502@hotmail.com	8899502	root@localh...	2023-03-24 16:43:33

9. INFORMES

Se creo una vista, la cual se exporto a Power BI, para los informes de gráficas.

Scripts Estado de Resultado

[Vinculo a GitHub - Vista_EstadoResultado](#)

SQL File 2* x compradores v_estadoresultado

Limit to 1000 rows

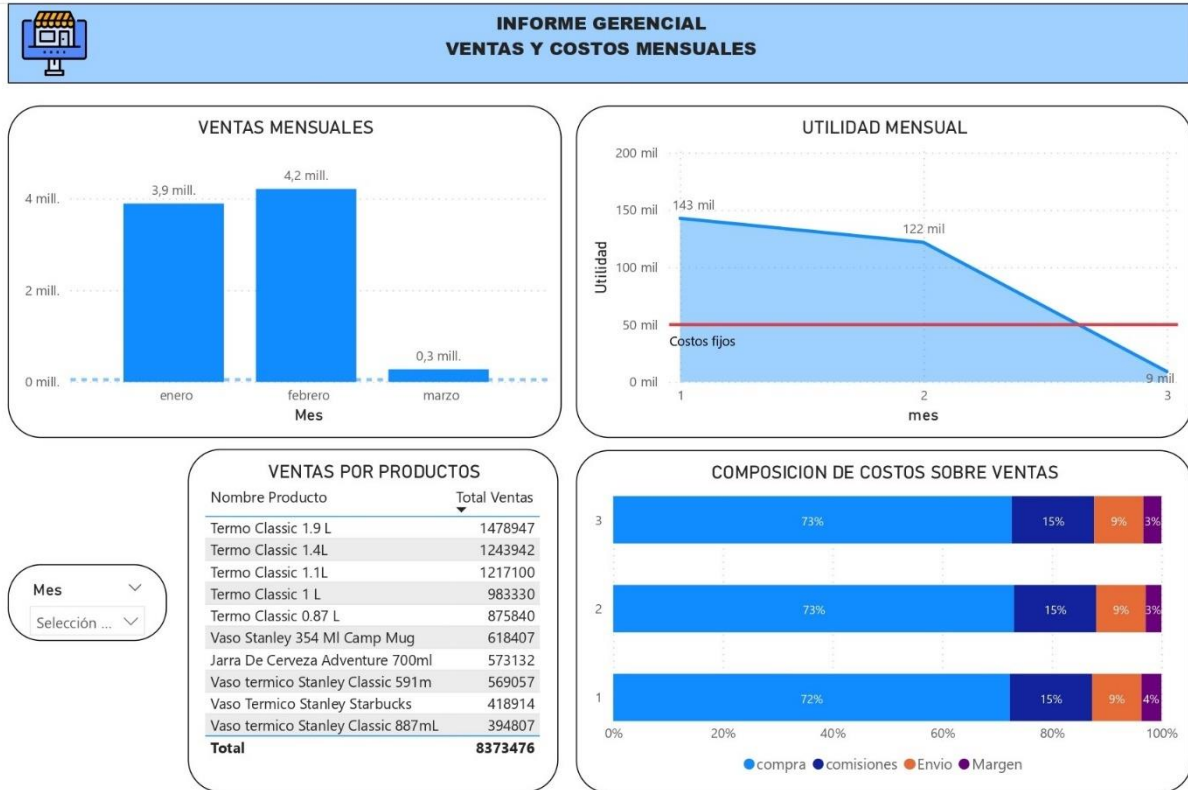
```

15
16 • create view v_EstadoResultado as
17   select month(v.fecha_venta) as Mes,
18   sum(v.precio_venta) as Venta,
19   sum(c.costos_reposicion ) as Costo_Compra,
20   round(sum(v.precio_venta)*0.09) as Envio,
21   round(sum(v.precio_venta)*0.15) as Comision,
22   #Calculo de utilidad
23   ⊕ round(
24   #Calculo de Margen = (Utilidad - Ventas)*100
25   ⊕ round(((round(
26   as Margen
27   from ventas as v
28   inner join costos as c
29   on v.Id_costos = c.Id_costos
30   group by month(v.fecha_venta);
31
32
33
34
35
36
37
38
39
40
41
42
43

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	Mes	Venta	Costo_Compra	Envio	Comision	Utilidad	Margen
▶	1	3889648	2813411	350068	583447	142722	3.67
	2	4207791	3076166	378701	631169	121755	2.89
	3	276037	200696	24843	41406	9092	3.29



Descripción de las gráficas:

A partir de las queries se generó el siguiente informe a nivel gerencia.

En la primera gráfica se muestra la evolución de las ventas mes a mes, mediante un gráfico de barras, con sus respectivas etiquetas que muestran en términos monetarios las ventas.

En la segunda gráfica arriba a la derecha, se muestra mediante un gráfico de líneas cual es la evolución de la utilidad mes a mes, la línea roja, representa los costos fijos, el objetivo es que a simple vista se pueda identificar los meses que se consiguieron utilidades por encima de los costos fijos.

Abajo a la izquierda se muestra un ranking de ventas por productos, ordenados de mayor a menos, posee un filtro, donde se puede obtener este ranking de un mes en particular, o de varios de ellos.

En la última gráfica, abajo a la derecha, se muestra la composición de los costos y la utilidad sobre las ventas, las barras comparativas de mes a mes, permite identificar si hay modificaciones de algún ítem de los costos de manera considerable en relación a otro mes.

10. FUTURAS LINEAS

Este modelo de negocios, entiende que el cliente no es propio, sino del marketplace, por lo cual para próximos informes se va a poner énfasis en los siguientes ejes:

- Comportamiento de las publicaciones
- Márgenes
- Reputación
 - Cantidad de ventas
 - Tiempos de entregas.
 - Reclamos

Para poder hacer un seguimiento operativo se buscará analizar márgenes de ganancias, cantidad de visitas que tienen las publicaciones, cantidad de ventas por publicación, tiempos entre que se realiza la venta y se envía y tipos de reclamos.

11. Script Final

[Link a GitHub - ScriptFinal](#)

12. HERRAMIENTAS UTILIZADAS

- Excel, Word.
- MySQL Workbench
- SQL Server
- SQL Server Migration Assistant for MySQL
- Power BI
- GitHub.com
- GitHub Desktop