



MySQL

Comisión 81860
Julian Daud

PROYECTO: ECOMMERCE TIENDA DE MUSICA

1) Descripción de la temática

El proyecto desarrolla la base de datos de un e-commerce de instrumentos musicales. El sistema permite registrar clientes, administrar un catálogo de productos, generar órdenes de compra y registrar pagos. La base de datos organiza y centraliza toda la información necesaria para el funcionamiento del sitio.

2) Situación problemática

La tienda usa archivos sueltos para manejar datos, lo que provoca:

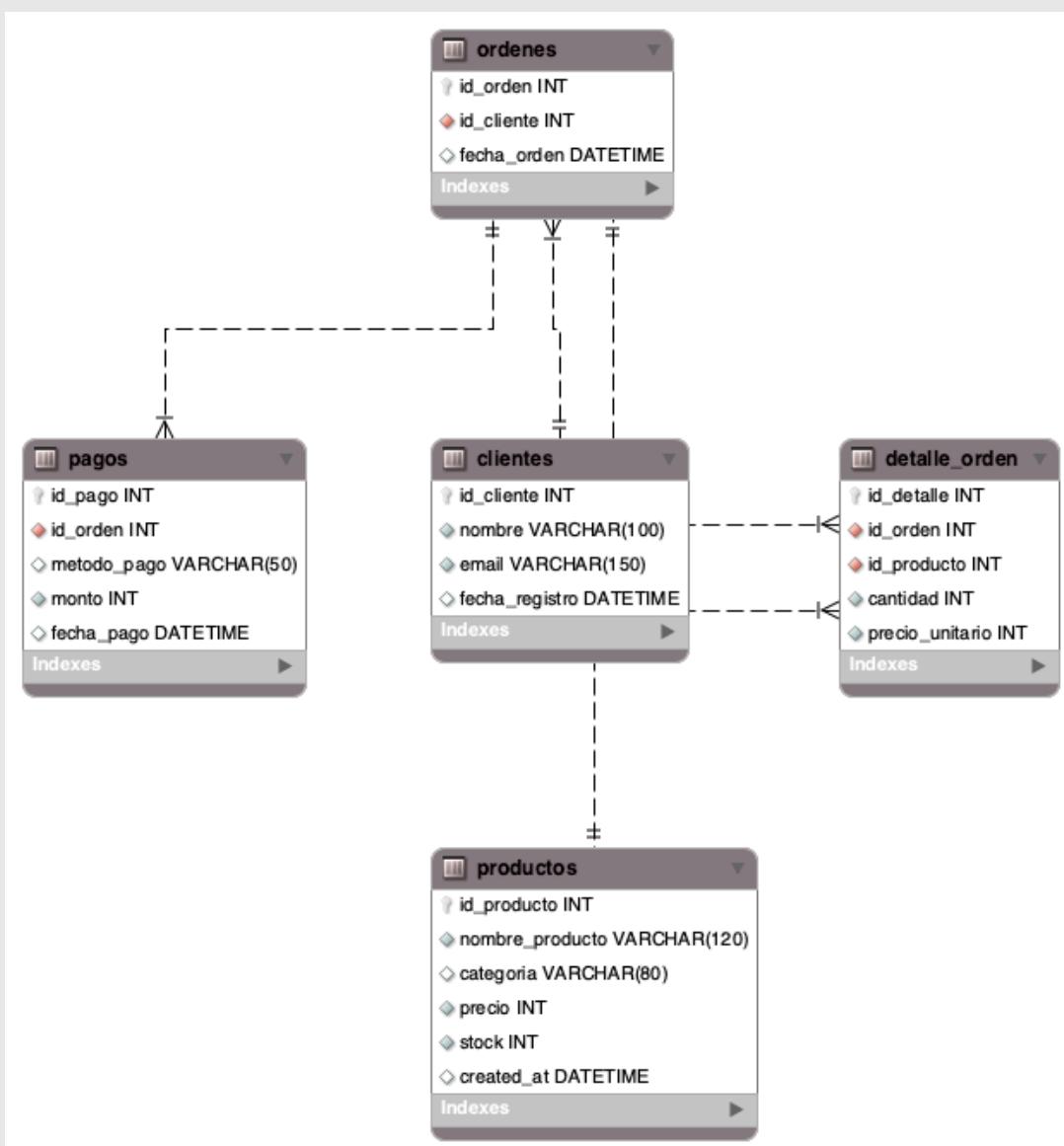
- 1.Dificultad para rastrear ventas.
- 2.Errores e inconsistencias en el stock.
- 3.Datos duplicados o incompletos.
- 4.Falta de un registro claro de pagos.

Se necesita una base de datos relacional que unifique y ordene toda la información.

3) Modelo de negocio

La empresa vende instrumentos musicales a través de un e-commerce. Ofrece productos con stock, clientes registrados, órdenes con múltiples ítems y un sistema de pagos. El negocio genera ingresos por venta directa y requiere una estructura de datos clara para operar de manera eficiente.

DIAGRAMA MODELO ER



LISTADO DE TABLAS

1) CLIENTES:

Registra los datos básicos de los clientes que se registran y compran en el e-commerce.

Nombre del campo	Abreviatura	Tipo de dato	Clave	Descripción
id_cliente	id_cliente	INT	PK	Identificador único del cliente
nombre	nombre	VARCHAR(100)	—	Nombre del cliente
email	email	VARCHAR(150)	—	Correo electrónico del cliente
fecha_registro	fecha_registro	DATETIME	—	Fecha en que el cliente se registró

2) PRODUCTOS:

Contiene el catálogo de instrumentos y accesorios disponibles para la venta

Nombre del campo	Abreviatura	Tipo de dato	Clave	Descripción
id_producto	id_producto	INT	PK	Identificador único del
nombre_produc to	nombre_produc to	VARCHAR(120)	—	Nombre del producto o
categoria	categoria	VARCHAR(80)	—	Tipo/categoría del instrumento
precio	precio	INT	—	Precio del producto
stock	stock	INT	—	Cantidad disponible en
created_at	created_at	DATETIME	—	Fecha en que se agregó el

3) ORDENES:

Registra cada compra realizada por los clientes

Nombre del campo	Abreviatura	Tipo de dato	Clave	Descripción
id_orden	id_orden	INT	PK	Identificador único de la orden
id_cliente	id_cliente	INT	FK	Cliente que realizó la compra
fecha_orden	fecha_orden	DATETIME	—	Fecha en que se generó la orden

4) DETALLE_ORDEN

Registra los productos incluidos en cada orden

Nombre del campo	Abreviatura	Tipo de dato	Clave	Descripción
id_detalle	id_detalle	INT	PK	Identificador único del detalle
id_orden	id_orden	INT	FK	Orden asociada
id_producto	id_producto	INT	FK	Producto incluido
cantidad	cantidad	INT	—	Cantidad comprada
precio_unitario	precio_unitario	INT	—	Precio del producto en el momento de la compra

5) PAGOS

Registra los pagos realizados por cada orden.

Nombre del campo	Abreviatura	Tipo de dato	Clave	Descripción
id_pago	id_pago	INT	PK	Identificador único del pago
id_orden	id_orden	INT	FK	Orden asociada al pago
metodo_pago	metodo_pago	VARCHAR(50)	—	Método de pago utilizado
monto	monto	INT	—	Importe del pago
fecha_pago	fecha_pago	DATETIME	—	Fecha en que se realizó el pago

CORRECCIONES 1RA ENTREGA

Se realizaron ajustes en la base de datos :

```
#== CORRECCIONES ==#  
  
72 • ALTER TABLE clientes  
73   ADD UNIQUE (email),  
74   ADD COLUMN ciudad VARCHAR (100),  
75   ADD COLUMN pais VARCHAR (100);  
76  
77  
78 • ALTER TABLE ordenes  
79   ADD COLUMN total DECIMAL(10,2);  
80  
81  
82 • ALTER TABLE productos  
83   MODIFY precio DECIMAL (10, 2);  
84  
85 • ALTER TABLE detalle_orden  
86   MODIFY precio_unitario DECIMAL (10, 2);
```

- Clientes: **email único** y nuevas columnas **ciudad y pais**.
- Órdenes: nueva columna **total** para el monto de la orden.
- **Productos / Detalle de Orden**: precios cambiados a **DECIMAL(10,2)** para mayor precisión.

Estos cambios mejoran la integridad de los datos y permiten las nuevas funcionalidades del proyecto.

CARGA DE DATOS A LA BASE

Se muestran los comandos **INSERT** utilizados para agregar datos a las tablas, junto con screenshots de la ejecución en MySQL.

1) PAGOS

```
262      # === PAGOS === #
264
265 •  INSERT INTO pagos (id_orden, metodo_pago, monto, fecha_pago) VALUES
266   (1, 'Tarjeta de crédito', 68.00, '2025-01-15 11:00:00'),
267   (2, 'Venmo', 79.00, '2025-02-03 15:00:00'),
268   (3, 'Zelle', 72.00, '2025-03-10 10:00:00'),
269   (4, 'Tarjeta de crédito', 87.50, '2025-04-22 17:00:00'),
270   (5, 'Venmo', 71.00, '2025-05-05 12:30:00'),
271   (6, 'Zelle', 57.50, '2025-06-18 14:00:00'),
272   (7, 'Tarjeta de crédito', 82.50, '2025-07-07 18:30:00'),
273   (8, 'Venmo', 70.00, '2025-08-12 13:30:00'),
274   (9, 'Zelle', 73.00, '2025-09-25 16:00:00'),
275   (10, 'Tarjeta de crédito', 54.00, '2025-10-03 11:00:00');
276
277
278      #== VISTAS ==#
279
280      #=1:##
281
282 •  CREATE VIEW vw_detalle_compra AS
283   SELECT p.categoría,
```

2) ORDENES

```
186      # === ORDENES === #
187
188
189 •  INSERT INTO ordenes (id_cliente, fecha_orden) VALUES
190   (1, '2025-01-15 10:20:00'),
191   (2, '2025-02-03 14:45:00'),
192   (3, '2025-03-10 09:30:00'),
193   (4, '2025-04-22 16:10:00'),
194   (5, '2025-05-05 11:50:00'),
195   (6, '2025-06-18 13:25:00'),
196   (7, '2025-07-07 17:40:00'),
197   (8, '2025-08-12 12:15:00'),
198   (9, '2025-09-25 15:30:00'),
199   (10, '2025-10-03 10:05:00'),
200   (11, '2025-11-11 14:20:00'),
201   (12, '2025-12-01 09:55:00'),
202   (13, '2025-01-28 16:45:00'),
203   (14, '2025-02-15 11:10:00'),
204   (15, '2025-03-30 13:50:00'),
205   (16, '2025-04-18 17:25:00'),
206   (17, '2025-05-27 10:40:00'),
```

3) PRODUCTOS

```
# == PRODUCTOS == #

INSERT INTO productos (nombre_producto, categoria, precio, stock, created_at) VALUES
  ('Vinilo "Midnight Echoes" - The Lunar Tides', 'Vinilo', 28.00, 40, '2025-01-05 10:00:00'),
  ('CD "Midnight Echoes" - The Lunar Tides', 'CD', 18.00, 50, '2025-01-10 11:00:00'),
  ('Camiseta "Midnight Echoes" - The Lunar Tides', 'Merchandising', 22.00, 80, '2025-01-15 12:00:00'),
  ('Pin "The Lunar Tides Logo"', 'Merchandising', 6.00, 150, '2025-01-20 13:00:00'),
  ('Poster "Midnight Echoes Album Art"', 'Poster', 12.00, 60, '2025-01-25 14:00:00'),

  ('Vinilo "Neon Shadows" - Desert Wolves', 'Vinilo', 20.00, 35, '2025-02-05 10:00:00'),
  ('CD "Neon Shadows" - Desert Wolves', 'CD', 20.00, 45, '2025-02-10 11:00:00'),
  ('Camiseta "Neon Shadows" - Desert Wolves', 'Merchandising', 24.00, 70, '2025-02-15 12:00:00'),
  ('Pin "Desert Wolves Logo"', 'Merchandising', 6.50, 140, '2025-02-20 13:00:00'),
  ('Poster "Neon Shadows Album Art"', 'Poster', 13.00, 55, '2025-02-25 14:00:00'),

  ('Vinilo "Crimson Horizon" - The Velvet Roads', 'Vinilo', 32.00, 30, '2025-03-05 10:00:00'),
  ('CD "Crimson Horizon" - The Velvet Roads', 'CD', 21.00, 40, '2025-03-10 11:00:00'),
  ('Camiseta "Crimson Horizon" - The Velvet Roads', 'Merchandising', 25.00, 60, '2025-03-15 12:00:00'),
  ('Pin "Velvet Roads Logo"', 'Merchandising', 7.00, 120, '2025-03-20 13:00:00'),
  ('Poster "Crimson Horizon Album Art"', 'Poster', 14.00, 50, '2025-03-25 14:00:00'),
  ('Vinilo "Echoes in Silence" - Midnight Signals', 'Vinilo', 29.00, 38, '2025-04-05 10:00:00'),
```

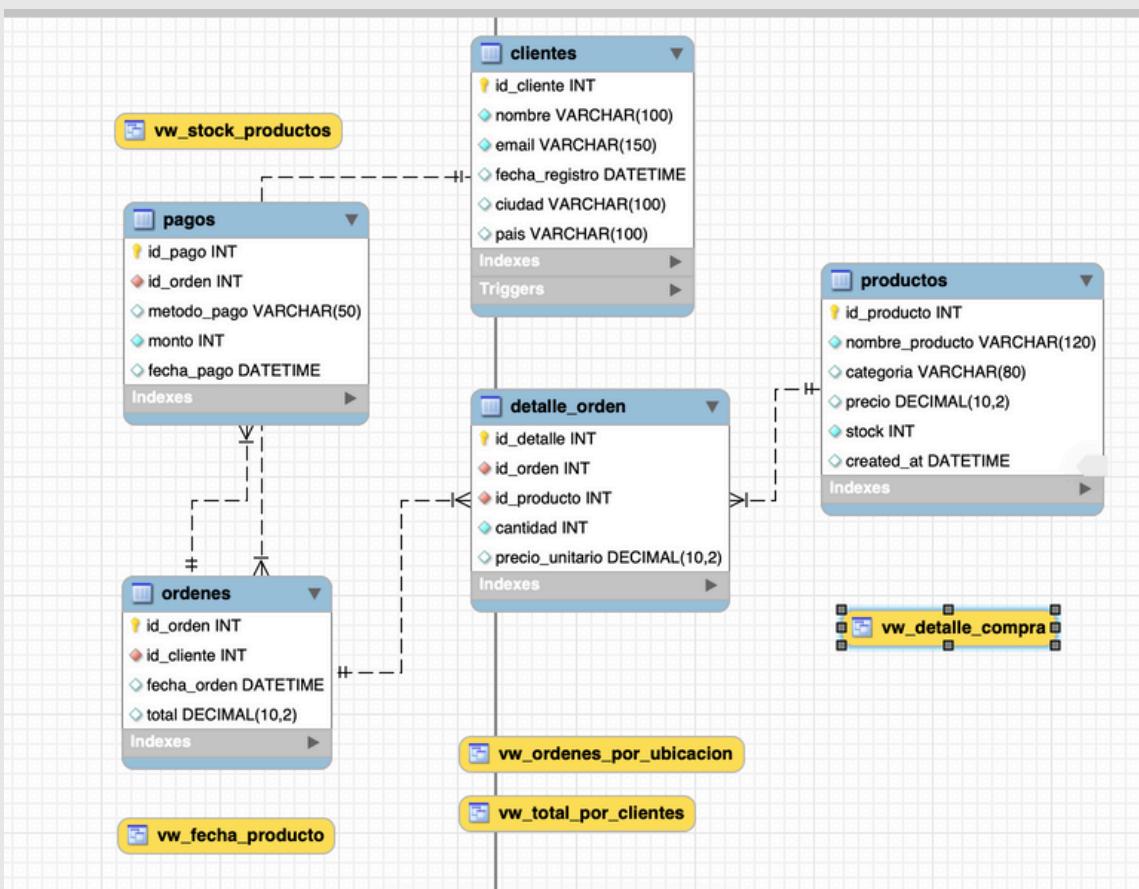
4) CLIENTES

```
INSERT INTO clientes (nombre, email, fecha_registro, ciudad, pais)
VALUES
  ('Paul Salvatierra', 'paul.salvatierra.35@gmail.com', NOW(), 'Brooklyn', 'USA'),
  ('Maria Perez', 'mariahotmail.com', NOW(), 'Buenos Aires', 'Argentina'),
  ('Carlos Lopez', 'carlos@gmail.com', NOW(), 'Madrid', 'España'),
  ('Gian Tenorio', 'gian.tenorio@gmail.com', NOW(), 'Call', 'Colombia'),
  ('Brian Scheller', 'bscheller@gmail.com', NOW(), 'Brooklyn', 'USA'),
  ('Juan Perez', 'juan.perez@gmail.com', NOW(), 'Buenos Aires', 'Argentina'),
  ('Maria Gonzalez', 'maria.gonzalez@yahoo.com', NOW(), 'Santiago', 'Chile'),
  ('Carlos Ramirez', 'carlos.ramirez@hotmail.com', NOW(), 'Lima', 'Peru'),
  ('Ana Fernandez', 'ana.fernandez@gmail.com', NOW(), 'Bogotá', 'Colombia'),
  ('Luis Martinez', 'luis.martinez@yahoo.com', NOW(), 'Quito', 'Ecuador'),
  ('Sofia Lopez', 'sofia.lopez@hotmail.com', NOW(), 'Montevideo', 'Uruguay'),
  ('Diego Sanchez', 'diego.sanchez@gmail.com', NOW(), 'Asunción', 'Paraguay'),
  ('Camila Torres', 'camila.torres@yahoo.com', NOW(), 'La Paz', 'Bolivia'),
  ('Jose Castillo', 'jose.castillo@hotmail.com', NOW(), 'Caracas', 'Venezuela'),
  ('Valentina Herrera', 'valentina.herrera@gmail.com', NOW(), 'Ciudad de Mexico', 'Mexico'),
  ('Ricardo Morales', 'ricardo.morales@yahoo.com', NOW(), 'Guadalajara', 'Mexico'),
  ('Fernanda Diaz', 'fernanda.diaz@hotmail.com', NOW(), 'Medellín', 'Colombia'),
  ('Gabriel Ortiz', 'gabriel.ortiz@gmail.com', NOW(), 'Santo Domingo', 'República Dominicana')
```

5) DETALLES ORDEN

```
215 •  INSERT INTO detalle_orden (id_orden, id_producto, cantidad, precio_unitario)
216   VALUES
217   (1, 1, 2, 28.00),
218   (1, 5, 1, 12.00),
219   (2, 2, 1, 18.00),
220   (2, 18, 2, 24.00),
221   (2, 15, 1, 13.00),
222   (3, 3, 1, 22.00),
223   (3, 7, 2, 25.00),
224   (4, 4, 1, 30.00),
225   (4, 12, 1, 14.00),
226   (4, 20, 2, 23.50),
227   (5, 8, 1, 26.00),
228   (5, 18, 2, 19.50),
229   (6, 6, 1, 31.00),
230   (6, 11, 2, 13.50),
231   (7, 9, 1, 21.00),
232   (7, 13, 1, 12.50),
233   (7, 17, 1, 24.50),
234   (8, 14, 2, 28.00),
```

MODELO ER ACTUALIZADO



Se ajustaron las tablas originales agregando columnas de ubicación y total, y se corrigieron tipos de datos para reflejar mejor los precios y montos, manteniendo las relaciones entre tablas y optimizando la integridad de la base de datos.

VIEWS

Se presentan los resultados de las **vistas** creadas, mostrando que **retornan correctamente la información esperada.**

VIEWS 1 Y 2

```
CREATE VIEW vw_detalle_compra AS
SELECT p.categoría,
       pa.método_pago,
       d.precio_unitario
FROM detalle_orden AS d
INNER JOIN productos AS p ON d.id_producto = p.id_producto
INNER JOIN ordenes AS o ON d.id_orden = o.id_orden
INNER JOIN pagos AS pa ON o.id_orden = pa.id_orden;

CREATE VIEW vw_total_por_clientes AS
SELECT c.nombre,
       p.categoría,
       o.fecha_orden,
       pa.monto
FROM clientes AS c
INNER JOIN ordenes AS o ON c.id_cliente = o.id_cliente
INNER JOIN detalle_orden AS do ON o.id_orden = do.id_orden
INNER JOIN productos AS p ON do.id_producto = p.id_producto
INNER JOIN pagos AS pa ON o.id_orden = pa.id_orden;
```

1. **vw_detalle_compra:** Muestra la categoría de los productos, el método de pago y el precio unitario de cada detalle de orden.

Tablas: **detalle_orden, productos, ordenes, pagos**

2. **vw_total_por_clientes:** Muestra el nombre del cliente, categoría de producto, fecha de orden y monto pagado por cada compra. Tablas: **clientes, ordenes, detalle_orden, productos, pagos**

VIEWS 3 Y 4

```
CREATE VIEW vw_ordenes_por ubicación AS
SELECT c.ciudad,
       c.pais,
       COUNT(o.id_orden) AS cantidad_ordenes
FROM clientes AS c
LEFT JOIN ordenes AS o ON c.id_cliente = o.id_cliente
GROUP BY c.ciudad, c.pais;

CREATE VIEW vw_fecha_producto AS
SELECT o.id_orden,
       o.fecha_orden,
       p.categoría,
       p.created_at
FROM ordenes o
INNER JOIN detalle_orden d ON o.id_orden = d.id_orden
INNER JOIN productos p ON d.id_producto = p.id_producto;
```

1. **vw_ordenes_por ubicación:** Agrupa las órdenes por ciudad y país, mostrando la cantidad de órdenes por ubicación. Tablas: **clientes, ordenes**

2. **vw_fecha_producto.** Muestra la fecha de cada orden junto con la categoría del producto y la fecha de creación del producto. Tablas: **ordenes, detalle_orden, productos**

VIEW 5

```
325      #=5=#  
326  
327 •  CREATE VIEW vw_stock_productos AS  
328   SELECT p.nombre_producto,  
329       p.stock,  
330       d.precio_unitario  
331   FROM productos p  
332   JOIN detalle_orden d ON p.id_producto = d.id_producto;  
333
```

1. vw_stock_productos. Muestra el stock disponible de cada producto en el inventario.
Tablas: **productos**

EJECUCIONES

```
326  
327 •  CREATE VIEW vw_stock_productos AS  
328   SELECT p.nombre_producto,  
329       p.stock,  
330       d.precio_unitario  
331   FROM productos p  
332   JOIN detalle_orden d ON p.id_producto = d.id_producto;  
333  
334  
335 # EJECUCIONES #  
336  
337 •  SELECT * FROM vw_detalle_compra  
338 WHERE metodo_pago IN ('Venmo');  
339  
340 •  SELECT * FROM vw_total_por_clientes  
341 WHERE categoria IN ('Vinilo');  
342  
343 •  SELECT * FROM vw_ordenes_por ubicacion  
344 WHERE pais IN ('Argentina');  
345
```

The screenshot shows two separate database queries running in a grid-based interface. Both queries are titled '1 •' and show results for 1000 rows. The first query, 'vw_ordenes_por ubicacion', displays data from a table with columns 'ciudad', 'pais', and 'cantidad_orden...'. The second query, 'vw_fecha_producto', displays data from a table with columns 'id_orden', 'fecha_orden', 'categoria', and 'created_at'. Both tables have several rows of data.

ciudad	pais	cantidad_orden...
Montevideo	Uruguay	3
Bogotá	2	
Buenos Aires	Argentina	2
Madrid	España	1
Cali	Colombia	1
Santiago	Chile	1
Lima	Perú	1
Bogotá	Colombia	1
Quito	Ecuador	1
Asunción	Paraguay	1
La Paz	Bolivia	1

id_orden	fecha_orden	categoria	created_at
1	2025-01-15 10:20:00	Vinilo	2025-12-18 20:40:14
1	2025-01-19 10:20:00	Poster	2025-12-18 20:40:14
2	2025-02-03 14:45:00	CD	2025-12-18 20:40:14
2	2025-02-03 14:45:00	Poster	2025-12-18 20:40:14
2	2025-02-03 14:45:00	Poster	2025-12-18 20:40:14
3	2025-03-10 09:30:00	Merchandising	2025-12-18 20:40:14
3	2025-04-22 16:10:00	Merchandising	2025-12-18 20:40:14
4	2025-04-22 16:10:00	CD	2025-12-18 20:40:14
4	2025-04-22 16:10:00	Poster	2025-12-18 20:40:14
5	2025-05-05 11:50:00	Merchandising	2025-12-18 20:40:14

Se muestran consultas de prueba sobre las **vistas, funciones y procedimientos almacenados**, verificando que devuelven correctamente los datos esperados según la lógica del proyecto.

FUNCIONES

MONTO TOTAL DE ORDENES

The screenshot shows the MySQL Workbench interface. On the left, the SQL editor contains the code for creating a function named 'monto_total_ordenes'. The code uses a cursor to sum the product of 'precio_unitario' and 'cantidad' from the 'detalle_orden' table where the order ID matches the parameter. The result is stored in the variable 'total'. The function returns this total. On the right, the results grid shows the output of the query 'select monto_total_ordenes(2);', which returns the value 79.00.

```
USE ecommerce_daud;
DELIMITER //
CREATE FUNCTION monto_total_ordenes (param_idOrden INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10,2);

    SELECT IFNULL(SUM(precio_unitario * cantidad), 0)
    INTO total
    FROM detalle_orden
    WHERE id_orden = param_idOrden;

    RETURN total;
END//
DELIMITER ;
SELECT monto_total_ordenes (2,3,8);

```

Calcula el **monto total de una orden** multiplicando precio unitario por cantidad de cada producto en detalle_orden.

Tablas: **detalle_orden**

Objetivo: **Permite obtener rápidamente el total de una orden específica, evitando sumar manualmente los productos.**

UBICACION DE CLIENTES

The screenshot shows the MySQL Workbench interface. On the left, the SQL editor contains the code for creating a function named 'ubicacion_clientes'. It concatenates the first name, last name, city, and country of a client into a single string. The function takes the client ID as a parameter. On the right, the results grid shows the output of the query 'select ubicacion_clientes(10);', which returns the string 'Ana Fernandez, Bogotá, Colombia'.

```
DELIMITER //
CREATE FUNCTION ubicacion_clientes(param_idCliente INT)
RETURNS VARCHAR(250)
DETERMINISTIC
BEGIN
    DECLARE info VARCHAR(250);
    SELECT CONCAT(nombre, ' ', apellido, ', ', ciudad, ', ', pais) INTO info
    FROM clientes
    WHERE id_cliente = param_idCliente;

    RETURN info;
END//
DELIMITER ;
SELECT ubicacion_clientes (2);
*** SP ***
# = 1 =

```

Devuelve el **nombre completo del cliente junto con su ciudad y país** en un solo string. Tablas: **clientes**

Objetivo: **Facilita mostrar información completa del cliente en reportes o vistas sin hacer múltiples joins o concatenaciones.**

STORED PROCEDURES

LISTAR ORDENES DE CLIENTES

The screenshot shows the MySQL Workbench interface. On the left, the SQL editor contains the code for creating a stored procedure:

```
CREATE PROCEDURE sp_listar_ordenes_cliente (IN p_id_cliente INT)
BEGIN
    SELECT
        o.id_orden,
        o.fecha_orden,
        monto_total_ordenes(o.id_orden) AS total
    FROM ordenes AS o
    WHERE o.id_cliente = p_id_cliente;
END//
```

Line numbers 409 to 419 are visible. Below the procedure definition, the code to call the procedure is shown:

```
DELIMITER //
USE ecommerce_daud;
CALL sp_listar_ordenes_cliente(2);
```

On the right, the results of the query are displayed in a result grid:

id_orden	fecha_orden	total
10	2025-10-03 10:05:00	46.50
...

Muestra todas las órdenes de un cliente específico junto con la fecha y el total de cada orden.

Tablas: **ordenes, detalle_orden (a través de la función monto_total_ordenes)**

Objetivo: **Permite consultar rápidamente el historial de compras de un cliente sin necesidad de joins complejos.**

ORDENAR CLIENTES

The screenshot shows the MySQL Workbench interface. On the left, the SQL editor contains the code for creating a stored procedure sp_ordenar_clientes:

```
DELIMITER //
CREATE PROCEDURE sp_ordenar_clientes(IN p_tabla VARCHAR(225), IN p_columna VARCHAR(225), IN p_orden VARCHAR(225))
BEGIN
    SET @sql_text = CONCAT("SELECT * FROM ", p_tabla, " ORDER BY ", p_columna, " ", p_orden, ";");
    PREPARE stat FROM @sql_text;
    EXECUTE stat;
    DEALLOCATE PREPARE stat;
END//
```

Line numbers 611 to 711 are visible. Below the procedure definition, the code to call the procedure is shown:

```
ALL sp_ordenar_clientes('clientes', 'fecha_registro', 'ASC');
```

On the right, the results of the query are displayed in a result grid:

id_cliente	nombre	email	fecha_registro	ciudad	país
1	Nuevo Cliente	nuevo@mail.com	2025-12-18 20:32:16	Montevideo	Uruguay
2	Raul Salvatierra	raul.salvatierra35@gmail.com	2025-12-18 20:36:16	Brooklyn	USA
3	Maria Hernandez	mari.hernandez99@gmail.com	2025-12-18 20:36:16	Buenos Aires	Argentina
4	Carlos Lopez	carlos@gmail.com	2025-12-18 20:36:16	Madrid	España
5	Gian Terorio	gian.terorio@gmail.com	2025-12-18 20:36:16	Call	Colombia
6	Brian Scheller	bsscheller@gmail.com	2025-12-18 20:36:16	Brooklyn	USA
7	Juan Perez	juan.perez@gmail.com	2025-12-18 20:36:16	Buenos Aires	Argentina
8	Alejandro Gomez	alejandro.gomez999@gmail.com	2025-12-18 20:36:16	Lima	Peru
9	Carlos Ramirez	carlos.ramirez@hotmail.com	2025-12-18 20:36:16	Bogotá	Colombia
10	Ana Fernandez	ana.fernandez@gmail.com	2025-12-18 20:36:16	Quito	Ecuador
11	Luis Martinez	luis.martinez@yahoo.com	2025-12-18 20:36:16

Ordena dinámicamente cualquier tabla por la columna y el orden especificado.

Tablas: **puede aplicarse a cualquier tabla que se pase como parámetro.**

Objetivo: **Permite listar datos de forma ordenada sin crear múltiples queries fijas, facilitando consultas flexibles y reutilizables.**

TRIGGERS

ACTUALIZAR STOCK

```
439
440    DELIMITER //
441
442 •  CREATE TRIGGER trg_actualizar_stock
443     AFTER INSERT ON detalle_orden
444     FOR EACH ROW
445     BEGIN
446         UPDATE productos
447             SET stock = stock - NEW.cantidad|
448             WHERE id_producto = NEW.id_producto;
449     END//
450
451     DELIMITER ;
452
453 •  SELECT stock FROM productos WHERE id_producto = 15;
454
455
456     DELIMITER //
```

⌚ | 37:447

Actualiza automáticamente el **stock de un producto** cada vez que se inserta un nuevo detalle de orden.

Tablas: **detalle_orden, productos**

Objetivo: **Mantener el inventario actualizado sin necesidad de cálculos manuales tras cada venta.**

MARCAR PAGO

```
DELIMITER //

CREATE TRIGGER trg_marcar_pago
AFTER UPDATE ON pagos
FOR EACH ROW
BEGIN
    IF NEW.estado_pago = 'Pagado' THEN
        UPDATE ordenes
            SET total = NEW.monto
            WHERE id_orden = NEW.id_orden;
    END IF;
END//

820
821     DELIMITER ;
822
823 •  SELECT * FROM pagos WHERE id_pago = 70;
824 •  SELECT * FROM ordenes WHERE id_orden = (SELECT id_orden FROM pagos WHERE id_pago = 70);
825
```

Trigger que se ejecuta al actualizar un registro en la **tabla pagos**. Cuando un pago cambia a estado “**Pagado**”, se actualiza el total correspondiente en la tabla **ordenes**, asegurando coherencia entre los pagos y las órdenes.

CORRECCIONES SEGUNDA ENTREGA

Se realizaron ajustes en la base de datos :



```
491 • ALTER TABLE pagos
492     MODIFY COLUMN estado_pago ENUM('Pagado', 'Pendiente', 'Rechazado');
493
494
495 • UPDATE pagos
496     SET estado_pago = 'Pagado'
497     WHERE id_pago >= 1
498         AND fecha_pago IS NOT NULL;
499
500
501 • UPDATE pagos
502     SET estado_pago = 'Pendiente'
503     WHERE id_pago IN (3, 7, 15);
504
505 • UPDATE pagos
506     SET estado_pago = 'Rechazado'
507     WHERE id_pago IN (9, 21);
508
```

- Se modificó la columna `estado_pago` en la tabla `pagos` a tipo **ENUM (Pagado, Pendiente, Rechazado)**.
- Se actualizaron los registros para reflejar el estado real de cada orden.
- Ajustes pensados para simplificar el seguimiento de pagos.



```
515 • INSERT INTO pagos (id_orden, metodo_pago, monto, fecha_pago, estado_pago) VALUES
516     (1, 'Venmo', 20.00, '2025-01-16 09:00:00', 'Pendiente'),
517     (2, 'Zelle', 30.00, '2025-02-04 10:30:00', 'Rechazado'),
518     (3, 'Tarjeta de crédito', 15.00, '2025-03-11 14:00:00', 'Pagado'),
519     (4, 'Venmo', 25.00, '2025-04-23 12:00:00', 'Pendiente'),
520     (5, 'Zelle', 10.00, '2025-05-06 16:00:00', 'Pagado'),
521     (6, 'Tarjeta de crédito', 18.00, '2025-06-19 10:00:00', 'Pendiente'),
522     (7, 'Venmo', 12.00, '2025-07-08 11:30:00', 'Rechazado'),
523     (8, 'Zelle', 22.00, '2025-08-13 09:45:00', 'Pendiente'),
524     (9, 'Tarjeta de crédito', 19.00, '2025-09-26 15:00:00', 'Pagado');
525
526
```

Se agregaron pagos adicionales usando distintos **id_orden** y **id_pago** para reflejar que una orden puede tener más de un pago, con estados variados (Pagado, Pendiente, Rechazado)

```
864  
865  
866 • ALTER TABLE pagos  
867     MODIFY monto DECIMAL(10,2);  
868  
869  
870  
871
```

Se modificó el campo monto de la tabla **pagos** a **DECIMAL(10,2)** para representar correctamente **valores monetarios**, permitir **decimales** y mantener consistencia en el modelo de datos.

```
...  
509 • CREATE OR REPLACE VIEW vw_stock_productos AS  
510     SELECT  
511         nombre_producto,  
512         stock  
513     FROM productos;
```

Se reemplazó la vista **vw_stock_productos** usando **CREATE OR REPLACE VIEW** para que seleccione únicamente **nombre_producto** y **stock** desde la tabla **productos**, eliminando la unión con **detalle_orden** y la columna **precio_unitario**

```
# === DATOS === #  
  
USE ecommerce_daud;  
  
#==== CLIENTES ===#  
  
USE ecommerce_daud;
```

Se corrigió el comando **USE** apuntando a la base de datos correcta **ecommerce_daud** antes de operar sobre la tabla **clientes**

Se modificó la llamada a la función **monto_total_ordenes** para ejecutar un **id_orden** por vez, respetando la definición de la función que acepta un solo parámetro

```
        RETURN total;  
END//  
  
386  
387     DELIMITER ;  
388  
389 •  SELECT monto_total_ordenes(2);  
390 •  SELECT monto_total_ordenes(3);  
391 •  SELECT monto_total_ordenes(8);  
392  
393
```

ACTUALIZACION DE TRIGGER

Se decidió remover el trigger **agregar clientes**, ya que requería la creación de una **nueva tabla** que no formaba parte del diseño original y podía **alterar la dinámica de la base de datos**. En su lugar, se implementó un **nuevo trigger** orientado a la gestión de pagos y órdenes, manteniendo una lógica **más simple y alineada** con el modelo definido.

ACTUALIZACION E INSERT DE DATOS

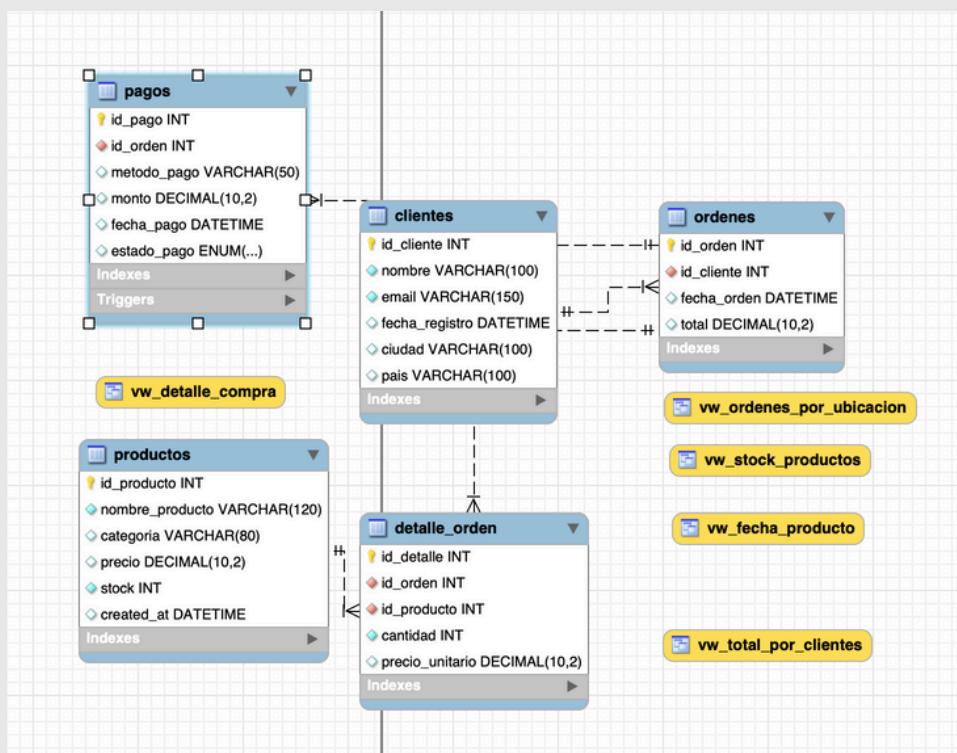
Se agregaron **nuevos registros** para lograr una **distribución más equilibrada y realista** de los datos, permitiendo probar mejor el funcionamiento del modelo. El dataset final cuenta con **70 clientes**, **115 órdenes**, **131 registros en detalle de orden**, **200 productos** y **115 pagos**.

ORDENES TOTAL

Result Grid				Filter Rows:	Search	Edit:	Export/Import:
id_orden	id_cliente	fecha_orden	total				
1	5	2025-01-01 10:00:00	21.00				
2	12	2025-01-02 11:15:00	12.00				
3	7	2025-01-03 12:30:00	24.00				
4	20	2025-01-04 13:45:00	20.00				
5	3	2025-01-05 14:00:00	7.00				
6	42	2025-01-06 15:10:00	23.00				
7	18	2025-01-07 16:20:00	14.50				
8	31	2025-01-08 17:30:00	13.50				
9	50	2025-01-09 09:45:00	32.00				
10	25	2025-01-10 10:50:00	22.00				
11	8	2025-01-11 11:05:00	14.00				
12	3	2025-01-12 12:15:00	22.00				
13	17	2025-01-13 13:25:00	6.50				
14	9	2025-01-14 14:35:00	6.50				
15	22	2025-01-15 15:45:00	22.00				

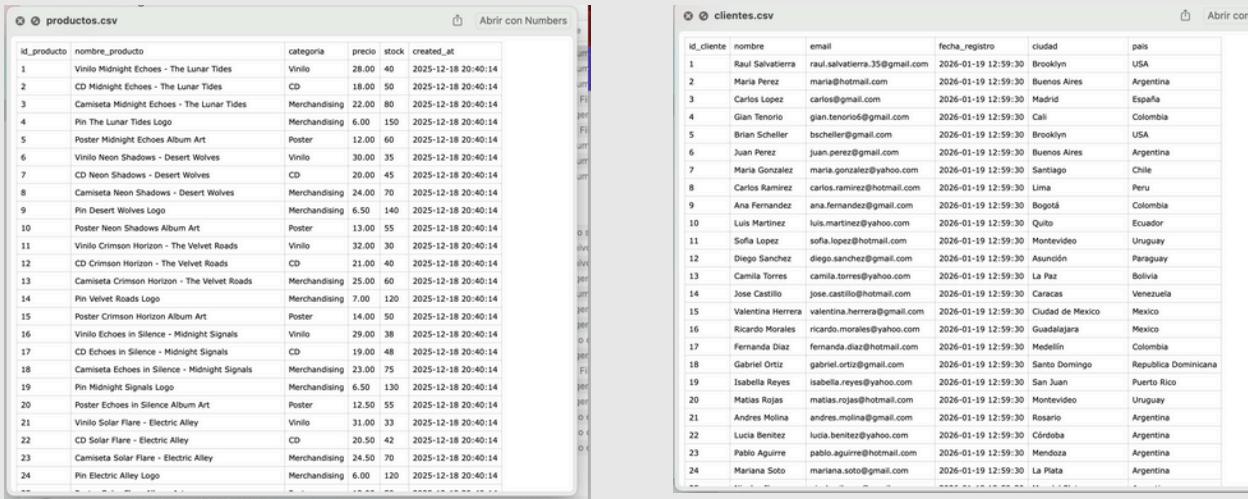
Se actualiza el campo **total** de la tabla **ordenes** tomando como base el precio de cada producto y la cantidad registrada en la tabla **detalle_orden**, permitiendo reflejar correctamente el monto total de cada orden.

MODELO ER FINAL



MATERIAL COMPLEMENTARIO

Se **exportaron las tablas desde MySQL Workbench** en formato **CSV**, las cuales fueron **abiertas y organizadas en Google Sheets** para conformar un **dataset de trabajo**. Posteriormente, el dataset fue utilizado en **Power BI** para **generar los gráficos de visualización**.

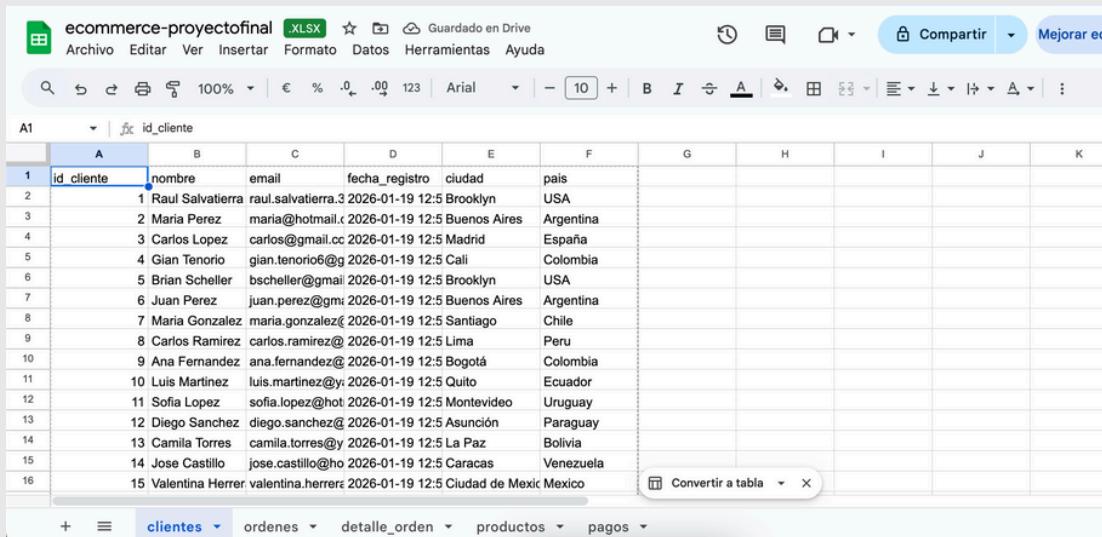


The image shows two Google Sheets side-by-side. The left sheet is titled 'productos.csv' and contains data for products. The right sheet is titled 'clientes.csv' and contains data for clients. Both sheets have columns for ID, name, price, stock, creation date, and other relevant details.

id_producto	nombre_producto	categoria	precio	stock	created_at
1	Vinilo Midnight Echoes - The Lunar Tides	Vinilo	28.00	40	2025-12-18 20:40:14
2	CD Midnight Echoes - The Lunar Tides	CD	18.00	50	2025-12-18 20:40:14
3	Camiseta Midnight Echoes - The Lunar Tides	Merchandising	22.00	80	2025-12-18 20:40:14
4	Pin The Lunar Tides Logo	Merchandising	6.00	150	2025-12-18 20:40:14
5	Poster Midnight Echoes Album Art	Poster	12.00	60	2025-12-18 20:40:14
6	Vinilo Neon Shadows - Desert Wolves	Vinilo	30.00	35	2025-12-18 20:40:14
7	CD Neon Shadows - Desert Wolves	CD	20.00	45	2025-12-18 20:40:14
8	Camiseta Neon Shadows - Desert Wolves	Merchandising	24.00	70	2025-12-18 20:40:14
9	Pin Desert Wolves Logo	Merchandising	6.50	140	2025-12-18 20:40:14
10	Poster Neon Shadows Album Art	Poster	13.00	55	2025-12-18 20:40:14
11	Vinilo Crimson Horizon - The Velvet Roads	Vinilo	32.00	30	2025-12-18 20:40:14
12	CD Crimson Horizon - The Velvet Roads	CD	21.00	48	2025-12-18 20:40:14
13	Camiseta Crimson Horizon - The Velvet Roads	Merchandising	25.00	60	2025-12-18 20:40:14
14	Pin Velvet Roads Logo	Merchandising	7.00	120	2025-12-18 20:40:14
15	Poster Crimson Horizon Album Art	Poster	14.00	50	2025-12-18 20:40:14
16	Vinilo Echoes in Silence - Midnight Signals	Vinilo	29.00	38	2025-12-18 20:40:14
17	CD Echoes in Silence - Midnight Signals	CD	19.00	48	2025-12-18 20:40:14
18	Camiseta Echoes in Silence - Midnight Signals	Merchandising	23.00	70	2025-12-18 20:40:14
19	Pin Midnight Signals Logo	Merchandising	6.50	130	2025-12-18 20:40:14
20	Poster Echoes in Silence Album Art	Poster	12.50	55	2025-12-18 20:40:14
21	Vinilo Solar Flare - Electric Alley	Vinilo	31.00	33	2025-12-18 20:40:14
22	CD Solar Flare - Electric Alley	CD	20.50	42	2025-12-18 20:40:14
23	Camiseta Solar Flare - Electric Alley	Merchandising	24.50	70	2025-12-18 20:40:14
24	Pin Electric Alley Logo	Merchandising	6.00	120	2025-12-18 20:40:14

id_cliente	nombre	email	fecha_registro	ciudad	pais
1	Raul Salvatierra	raul.salvatierra.3@gmail.com	2026-01-19 12:59:30	Brooklyn	USA
2	Maria Perez	maria@hotmail.com	2026-01-19 12:59:30	Buenos Aires	Argentina
3	Carlos Lopez	carlos@gmail.com	2026-01-19 12:59:30	Madrid	España
4	Gian Tenorio	gian.tenorio6@gmail.com	2026-01-19 12:59:30	Call	Colombia
5	Brian Scheller	bscheller@gmail.com	2026-01-19 12:59:30	Brooklyn	USA
6	Juan Perez	juan.perez@gmail.com	2026-01-19 12:59:30	Buenos Aires	Argentina
7	Maria Gonzalez	maria.gonzalez@yahoo.com	2026-01-19 12:59:30	Santiago	Chile
8	Carlos Ramirez	carlos.ramirez@hotmail.com	2026-01-19 12:59:30	Lima	Peru
9	Ana Fernandez	ana.fernandez@gmail.com	2026-01-19 12:59:30	Bogotá	Colombia
10	Luis Martinez	luis.martinez@yahoo.com	2026-01-19 12:59:30	Quito	Ecuador
11	Sofia Lopez	sofia.lopez@hotmail.com	2026-01-19 12:59:30	Montevideo	Uruguay
12	Diego Sanchez	diego.sanchez@gmail.com	2026-01-19 12:59:30	Asunción	Paraguay
13	Camila Torres	camila.torres@yahoo.com	2026-01-19 12:59:30	La Paz	Bolivia
14	Jose Castillo	jose.castillo@hotmail.com	2026-01-19 12:59:30	Caracas	Venezuela
15	Valentina Herrera	valentina.herrera@gmail.com	2026-01-19 12:59:30	Ciudad de Mexico	Mexico
16	Ricardo Morales	ricardo.morales@yahoo.com	2026-01-19 12:59:30	Guadalajara	Mexico
17	Fernanda Diaz	fernanda.diaz@hotmail.com	2026-01-19 12:59:30	Medellín	Colombia
18	Gabriel Ortiz	gabriel.ortiz@gmail.com	2026-01-19 12:59:30	Santa Domingo	República Dominicana
19	Isabella Reyes	isabella.reyes@yahoo.com	2026-01-19 12:59:30	San Juan	Puerto Rico
20	Matias Rojas	matias.rojas@hotmail.com	2026-01-19 12:59:30	Montevideo	Uruguay
21	Andres Molina	andres.molina@gmail.com	2026-01-19 12:59:30	Rosario	Argentina
22	Luria Benitez	luria.benitez@yahoo.com	2026-01-19 12:59:30	Córdoba	Argentina
23	Pablo Aguirre	pablo.aguirre@hotmail.com	2026-01-19 12:59:30	Mendoza	Argentina
24	Mariana Soto	mariana.soto@gmail.com	2026-01-19 12:59:30	La Plata	Argentina

Cientes y productos corresponden a dos de los cinco archivos CSV importados desde **MySQL Workbench**, utilizados para la carga de datos dentro del proyecto.



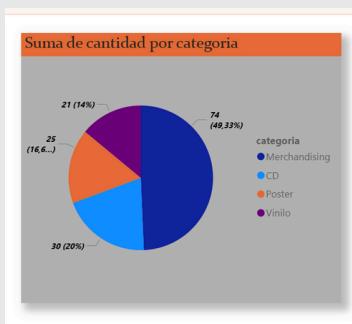
The image shows a Google Sheets spreadsheet titled 'ecommerce-proyectofinal'. The data is organized into several tabs at the bottom: 'clientes', 'ordenes', 'detalle_orden', 'productos', and 'pagos'. The 'clientes' tab is active and displays data from the 'clientes.csv' file. The columns include id_cliente, nombre, email, fecha_registro, ciudad, and pais.

A1	id_cliente	nombre	email	fecha_registro	ciudad	pais
1	1	Raul Salvatierra	raul.salvatierra.3@gmail.com	2026-01-19 12:5 Brooklyn	USA	
2	2	Maria Perez	maria@hotmail.com	2026-01-19 12:5 Buenos Aires	Argentina	
3	3	Carlos Lopez	carlos@gmail.cc	2026-01-19 12:5 Madrid	España	
4	4	Gian Tenorio	gian.tenorio6@gmail.com	2026-01-19 12:5 Cal	Colombia	
5	5	Brian Scheller	bscheller@gmail.com	2026-01-19 12:5 Brooklyn	USA	
6	6	Juan Perez	juan.perez@gmail.com	2026-01-19 12:5 Buenos Aires	Argentina	
7	7	Maria Gonzalez	maria.gonzalez@yahoo.com	2026-01-19 12:5 Santiago	Chile	
8	8	Carlos Ramirez	carlos.ramirez@gmail.com	2026-01-19 12:5 Lima	Peru	
9	9	Ana Fernandez	ana.fernandez@gmail.com	2026-01-19 12:5 Bogotá	Colombia	
10	10	Luis Martinez	luis.martinez@yahoo.com	2026-01-19 12:5 Quito	Ecuador	
11	11	Sofia Lopez	sofia.lopez@hotmail.com	2026-01-19 12:5 Montevideo	Uruguay	
12	12	Diego Sanchez	diego.sanchez@gmail.com	2026-01-19 12:5 Asunción	Paraguay	
13	13	Camila Torres	camila.torres@yahoo.com	2026-01-19 12:5 La Paz	Bolivia	
14	14	Jose Castillo	jose.castillo@hotmail.com	2026-01-19 12:5 Caracas	Venezuela	
15	15	Valentina Herrera	valentina.herrera@gmail.com	2026-01-19 12:5 Ciudad de Mexico	Mexico	

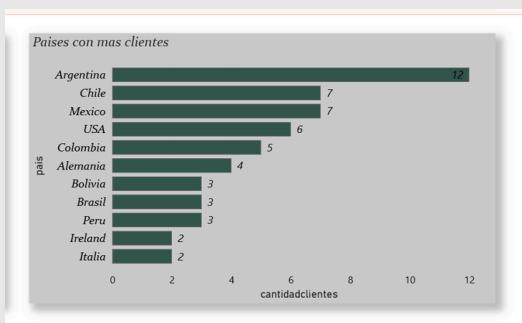
En el proyecto en **Excel** podemos visualizar el **dataset**, el cual luego fue importado a **Power BI** para la creación de gráficos y visualizaciones.

VISUALIZACIONES EN POWER BI

Se elaboraron **visualizaciones interactivas en Power BI**, utilizando **medidas DAX**, con el objetivo de mejorar la **visibilidad** y el **análisis de los datos**, permitiendo identificar patrones, tendencias y comparaciones relevantes a partir de la información del proyecto.



Mediante un **gráfico de torta**, se visualiza la **distribución de ventas por tipo de producto**, donde el **merchandising** representa casi el 50% del total, mientras que **CDs, pósters y vinilos** se reparten la mitad restante de forma equilibrada



El **gráfico de barras** presenta la **cantidad de clientes por país**, mostrando **únicamente los 11 países con mayor número de clientes**. **Argentina** lidera con 12 clientes, seguida por **Chile y México** con 7, **Estados Unidos** con 6 y **Colombia** con 5 clientes.

metodo_pago	Productos Vendidos
Paypal	34
Tarjeta	39
Venmo	40
Zelle	37
Total	150

En la **tabla** se presenta el **método de pago** junto con la cantidad de productos vendidos, donde **Venmo** lidera con 40 unidades, seguido por **Tarjeta** con 39, **Zelle** con 37 y **PayPal** con 34 productos vendidos.

LINK DE ACCESO A REPOSITORY

julesdaud/mysql-entrega-final

Final project using MySQL for database design and



<https://github.com/julesdaud/mysql-entrega-final>