

Entrega 2:

Vistas:

1: Datos Clientes

- Tabla Clientes
- Uso: fácil acceso a traer info del cliente

2: Compra Proveedores

- Tabla detalles_compras_proveedores , Peliculas, Proveedores, Compra_proveedores
- Uso acceso rápido a compra por proveedor

3: Catalogo

- Tabla Peliculas
- Uso Catalogo

4: Join Venta / Cliente

- Tabla Proveedores, Cliente, Detalles_Transaccion
- Uso identificar las ventas a Clientes

5: Compra Proveedor/Pelicula

- Tabla Proveedores, detalles_compras_proveedores , Peliculas
- Uso tabla similar a Compra/Proveedores pero sin detalles de las formas de pago, fecha, etc centrándose en películas y proveedor nomas

Código + Descripciones y Pruebas:

USE VENTA_DE_PELICULAS;

-- VISTAS

-- PRIMER VISTA TRAE DATOS CLIENTES



```
CREATE VIEW `info_clientes` AS
```


```
SELECT nombre, apellido, direccion, celular, correo
```

FROM Clientes;

SELECT * FROM info_clientes;

Result Grid

  Filter Rows:

Export:  Wrap Cell Content

	nombre	apellido	direccion	celular	correo
▶	Carlos	Gutiérrez	Calle 123	123456789	carlos@example.com
	Ana	Martínez	Avenida 456	987654321	ana@example.com
	Carlos	Gutiérrez	Calle 123	123456789	carlos@example.com
	Ana	Martínez	Avenida 456	987654321	ana@example.com
	Carlos	Gutiérrez	Calle 123	123456789	carlos@example.com
	Ana	Martínez	Avenida 456	987654321	ana@example.com
	Carlos	Gutiérrez	Calle 123	123456789	carlos@example.com
	Ana	Martínez	Avenida 456	987654321	ana@example.com
	Juancito	Rodriguez	Calle 8	2224455	juan@mail.com
	Juancito	Rodriguez	Calle 8	2224455	juan@mail.com

-- SEGUNDA TRAE DETALLES COMPRA PROVEEDORES

```
CREATE VIEW `detalles_compras_proveedores` AS

SELECT cp.fecha, cp.metodo_pago, pr.nombre AS nombre_proveedor, pel.titulo, dcp.cantidad,
dcp.precio_unitario, (dcp.cantidad * dcp.precio_unitario) AS total

FROM Compra_Proveedores cp

JOIN Proveedores pr ON cp.id_proveedor = pr.id_proveedor

JOIN Detalles_Compra_Proveedores dcp ON cp.id_transaccion = dcp.id_transaccion

JOIN Peliculas pel ON dcp.id_pelicula = pel.id_pelicula;
```

SELECT * FROM detalles_compras_proveedores;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	fecha	metodo_pago	nombre_proveedor	titulo	cantidad	precio_unitario	total
▶	2024-03-22	Transferencia	Proveedor Uno	Back to the Future	5	2.50	12.50
	2024-03-23	Tarjeta	Proveedor Dos	The Breakfast Club	3	2.75	8.25
	2024-03-22	Transferencia	Proveedor Uno	Raiders of the Lost Ark	2	2.25	4.50
	2024-03-23	Tarjeta	Proveedor Dos	Back to the Future	4	3.00	12.00

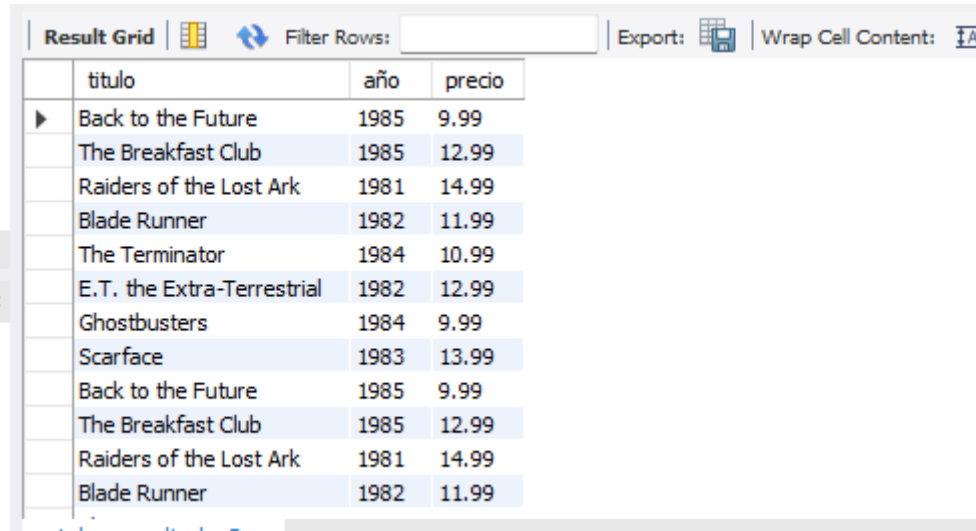
-- TERCERA TRAE CATALOGO

```
CREATE VIEW `catalogo_peliculas` AS
```

```
SELECT titulo, año, precio
```

```
FROM Peliculas;
```

```
SELECT * FROM catalogo_peliculas;
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid displays a list of movies with columns for 'titulo', 'año', and 'precio'. The data is as follows:

	titulo	año	precio
▶	Back to the Future	1985	9.99
	The Breakfast Club	1985	12.99
	Raiders of the Lost Ark	1981	14.99
	Blade Runner	1982	11.99
	The Terminator	1984	10.99
	E.T. the Extra-Terrestrial	1982	12.99
	Ghostbusters	1984	9.99
	Scarface	1983	13.99
	Back to the Future	1985	9.99
	The Breakfast Club	1985	12.99
	Raiders of the Lost Ark	1981	14.99
	Blade Runner	1982	11.99

```
-- CUARTA TRAE VENTA POR CLIENTE - mediante el uso de joins
```

```
CREATE VIEW `ventas_por_cliente_y_pelicula` AS
```

```
SELECT c.nombre, c.apellido, p.titulo, dt.cantidad, dt.precio_unitario, (dt.cantidad *  
dt.precio_unitario) AS total
```

```
FROM Clientes c
```

```
JOIN Transacciones t ON c.id_cliente = t.id_cliente
```

```
JOIN Detalles_Transaccion dt ON t.id_transaccion = dt.id_transaccion
```

```
JOIN Peliculas p ON dt.id_pelicula = p.id_pelicula;
```

```
SELECT * FROM ventas_por_cliente_y_pelicula;
```

Result Grid						
Filter Rows: <input type="text"/>						
Export: <input type="button" value="Export"/> Wrap Cell Content: <input type="button" value="Wrap"/>						
	nombre	apellido	titulo	cantidad	precio_unitario	total
▶	Carlos	Gutiérrez	Back to the Future	1	9.99	9.99
	Ana	Martínez	The Breakfast Club	2	12.99	25.98
	Ana	Martínez	Raiders of the Lost Ark	1	14.99	14.99
	Carlos	Gutiérrez	Back to the Future	1	9.99	9.99
	Ana	Martínez	The Breakfast Club	2	12.99	25.98
	Ana	Martínez	Raiders of the Lost Ark	1	14.99	14.99
	Carlos	Gutiérrez	Back to the Future	1	9.99	9.99
	Ana	Martínez	The Breakfast Club	2	12.99	25.98
	Ana	Martínez	Raiders of the Lost Ark	1	14.99	14.99
	Carlos	Gutiérrez	Back to the Future	1	9.99	9.99
	Ana	Martínez	The Breakfast Club	2	12.99	25.98
	Ana	Martínez	Raiders of the Lost Ark	1	14.99	14.99

ventas_por_cliente_y_pelicula6 x

-- QUINTA TRAE COMPRA CON PROVEEDOR + PELICULA

```
CREATE VIEW `compras_por_proveedor_y_pelicula` AS
```

```
SELECT pr.nombre AS nombre_proveedor, p.titulo, dcp.cantidad, dcp.precio_unitario,
(dcp.cantidad * dcp.precio_unitario) AS total
```

```
FROM Proveedores pr
```

```
JOIN Compra_Proveedores cp ON pr.id_proveedor = cp.id_proveedor
```

```
JOIN Detalles_Compra_Proveedores dcp ON cp.id_transaccion = dcp.id_transaccion
```

```
JOIN Peliculas p ON dcp.id_pelicula = p.id_pelicula;
```

```
SELECT * FROM compras_por_proveedor_y_pelicula;
```

Result Grid					
Filter Rows: <input type="text"/>					
Export: <input type="button" value="Export"/> Wrap Cell Content: <input type="button" value="Wrap"/>					
	nombre_proveedor	titulo	cantidad	precio_unitario	total
▶	Proveedor Uno	Back to the Future	5	2.50	12.50
	Proveedor Dos	The Breakfast Club	3	2.75	8.25
	Proveedor Uno	Raiders of the Lost Ark	2	2.25	4.50
	Proveedor Dos	Back to the Future	4	3.00	12.00

FUNCIONES:

1: anio_pelicula :

Uso: Recibe un año y retorna un título de película de ese año

2: CalcularPrecio :

Uso: Recibe un id_detalle lo busca en la tabla detalles_compra_proveedores y retorna el costo total de esa transacción multiplicando la cantidad de películas por el costo unitario

Código:

Prueba 1:

```
1 CREATE DEFINER='root'@'localhost' FUNCTION `anio_pelicula`(numero INT) RETURNS varchar(255) CHARSET utf8mb4
2 READS SQL DATA
3 BEGIN
4     DECLARE pelicula VARCHAR(255);
5
6     SELECT titulo INTO pelicula
7     FROM Peliculas
8     WHERE año = numero
9     LIMIT 1;
10
11     RETURN pelicula;
12 END
```

Todas las transacciones:

Result Grid					
Filter Rows:					
Edit: Export/Import: Wrap Cell Content: I A					
	id_detalle	id_transaccion	id_pelicula	cantidad	precio_unitario
▶	1	1	1	5	2.50
	2	2	2	3	2.75
	3	3	3	2	2.25
	4	4	1	4	3.00
✖	NULL	NULL	NULL	NULL	NULL

Result 1 detalles_compra_proveedores 2 x		
Output		
Action Output		
#	Time	Action
180	16:53:11	SELECT * FROM catalogo_peliculas LIMIT 0, 1000
181	16:53:46	SELECT * FROM ventas_por_cliente_y_pelicula LIMIT 0, 1000
182	16:54:25	SELECT * FROM compras_por_proveedor_y_pelicula LIMIT 0, 1000
183	17:06:41	SELECT CalcularPrecio(3) AS PrecioTotal LIMIT 0, 1000
184	17:06:41	SELECT * FROM detalles_compra_proveedores LIMIT 0, 1000

```
1 • USE VENTA_DE_PELICULAS;
2
3
4 -- Se ejecuta la funcion calcula Precio
5
6 • SELECT CalcularPrecio(3) AS PrecioTotal;
```

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

PrecioTotal
4.50

Result 6 x

Output

Action Output

#	Time	Action	Message
✓ 184	17:06:41	SELECT * FROM detalles_compra_proveedores LIMIT 0, 1000	4 row(s) returned
✓ 185	17:07:13	SELECT anio_pelicula(1985) AS PeliculaAño LIMIT 0, 1000	1 row(s) returned
✓ 186	17:07:13	SELECT * FROM peliculas LIMIT 0, 1000	25 row(s) returned
✓ 187	17:08:08	SELECT CalcularPrecio(3) AS PrecioTotal LIMIT 0, 1000	1 row(s) returned
✓ 188	17:08:34	SELECT CalcularPrecio(3) AS PrecioTotal LIMIT 0, 1000	1 row(s) returned

Prueba 2:

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `CalcularPrecio`(numero INT) RETURNS decimal(10,2)
2     READS SQL DATA
3     BEGIN
4         DECLARE total DECIMAL(10,2);
5
6         SELECT cantidad * precio_unitario INTO total
7         FROM detalles_compra_proveedores
8         WHERE numero = id_detalle
9         LIMIT 1; -- Limitar la consulta a un solo resultado
10
11     RETURN total;
12 END
```

Todas las películas:

9

10 • `SELECT anio_pelicula(1985) AS PeliculaAño;`

11 • `SELECT * FROM peliculas;`

12

13

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id_pelicula	titulo	genero	director	año	precio
▶	1	Back to the Future	Ciencia ficción	Robert Zemeckis	1985	9.99
	2	The Breakfast Club	Comedia dramática	John Hughes	1985	12.99
	3	Raiders of the Lost Ark	Aventura	Steven Spielberg	1981	14.99
	4	Blade Runner	Ciencia ficción	Ridley Scott	1982	11.99
	5	The Terminator	Acción y ciencia ficción	James Cameron	1984	10.99
	6	E.T. the Extra-Terrestrial	Ciencia ficción	Steven Spielberg	1982	12.99
	7	Ghostbusters	Comedia de ciencia ficción	Ivan Reitman	1984	9.99
	8	Scarface	Drama criminal	Brian De Palma	1983	13.99
	9	Back to the Future	Ciencia ficción	Robert Zemeckis	1985	9.99
	10	The Breakfast Club	Comedia dramática	John Hughes	1985	12.99
	11	Raiders of the Lost Ark	Aventura	Steven Spielberg	1981	14.99
	12	Blade Runner	Ciencia ficción	Ridley Scott	1982	11.99

Result 3 `peliculas 4` ×

Output

Action Output

#	Time	Action	Message
✓ 182	16:54:25	SELECT * FROM compras_por_proveedor_y_pelicula LIMIT 0, 1000	4 row(s) returned
✓ 183	17:06:41	SELECT CalcularPrecio(3) AS PrecioTotal LIMIT 0, 1000	1 row(s) returned
✓ 184	17:06:41	SELECT * FROM detalles_compra_proveedores LIMIT 0, 1000	4 row(s) returned
✓ 185	17:07:13	SELECT anio_pelicula(1985) AS PeliculaAño LIMIT 0, 1000	1 row(s) returned
✓ 186	17:07:13	SELECT * FROM peliculas LIMIT 0, 1000	25 row(s) returned

Solo la de 1985:

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

PeliculaAño

▶ Back to the Future

Result 7 ×

Output

Action Output

#	Time	Action	Message
✓ 185	17:07:13	SELECT anio_pelicula(1985) AS PeliculaAño LIMIT 0, 1000	1 row(s) returned
✓ 186	17:07:13	SELECT * FROM peliculas LIMIT 0, 1000	25 row(s) returned
✓ 187	17:08:08	SELECT CalcularPrecio(3) AS PrecioTotal LIMIT 0, 1000	1 row(s) returned
✓ 188	17:08:34	SELECT CalcularPrecio(3) AS PrecioTotal LIMIT 0, 1000	1 row(s) returned
✓ 189	17:09:01	SELECT anio_pelicula(1985) AS PeliculaAño LIMIT 0, 1000	1 row(s) returned

Solo la del 1982:

10 • `SELECT anio_pelicula(1982) AS PeliculaAño;`
 11 • `SELECT * FROM peliculas;`
 12
 13

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

PeliculaAño
 ▶ Blade Runner

Result 8 x

Output

Action Output

#	Time	Action	Message
✓ 186	17:07:13	SELECT * FROM peliculas LIMIT 0, 1000	25 row(s) returned
✓ 187	17:08:08	SELECT CalcularPrecio(3) AS PrecioTotal LIMIT 0, 1000	1 row(s) returned
✓ 188	17:08:34	SELECT CalcularPrecio(3) AS PrecioTotal LIMIT 0, 1000	1 row(s) returned
✓ 189	17:09:01	SELECT anio_pelicula(1985) AS PeliculaAño LIMIT 0, 1000	1 row(s) returned
✓ 190	17:09:20	SELECT anio_pelicula(1982) AS PeliculaAño LIMIT 0, 1000	1 row(s) returned

STORE PROCEDURES:

- 1: Procedimiento insert_cliente , uso: fácil inserción de usuarios
- 2: Procedimiento insert_pelicula , uso: fácil inserción de películas

CODIGO:

```

1 • CREATE DEFINER='root'@'localhost' PROCEDURE `insertar_cliente`(
2     IN p_nombre VARCHAR(100),
3     IN p_apellido VARCHAR(100),
4     IN p_edad INT,
5     IN p_direccion VARCHAR(255),
6     IN p_celular VARCHAR(15),
7     IN p_correo VARCHAR(100)
8 )
9 BEGIN
10     INSERT INTO Clientes (nombre, apellido, edad, direccion, celular, correo)
11     VALUES (p_nombre, p_apellido, p_edad, p_direccion, p_celular, p_correo);
12 END
  
```



```
DDL:
1 CREATE DEFINER='root'@'localhost' PROCEDURE `insertar_pelicula` (
2     IN p_titulo VARCHAR(255),
3     IN p_genero VARCHAR(100),
4     IN p_director VARCHAR(100),
5     IN p_año int,
6     IN p_precio DECIMAL(10,2)
7 )
8 BEGIN
9     INSERT INTO Peliculas (titulo, genero, director, año, precio)
10    VALUES (p_titulo, p_genero, p_director, p_año, p_precio);
11 END
```

Pruebas:

Call stored procedure venta_de_peliculas.insertar_cliente

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

p_nombre	<input type="text" value="Pedro"/>	[IN] VARCHAR(100)
p_apellido	<input type="text" value="Rodriguez"/>	[IN] VARCHAR(100)
p_edad	<input type="text" value="21"/>	[IN] INT
p_direccion	<input type="text" value="Calle o"/>	[IN] VARCHAR(255)
p_celular	<input type="text" value="111222"/>	[IN] VARCHAR(15)
p_correo	<input type="text" value="pedro@mail.com"/>	[IN] VARCHAR(100)

191 17:12:17 call venta_de_peliculas.insertar_cliente('Pedro', 'Rodriguez', 21, 'Calle o', '111222', 'pedro@mail.com') 1 row(s) affected

Call stored procedure venta_de_peliculas.insertar_pelicula

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

p_titulo	Agente X	[IN]	VARCHAR(255)
p_genero	Drama	[IN]	VARCHAR(100)
p_director	Juancito	[IN]	VARCHAR(100)
p_año	2028	[IN]	int
p_precio	8,02	[IN]	DECIMAL(10,2)

Execute Cancel

193 17:13:07 call venta_de_peliculas.insertar_pelicula('Agente X', 'Drama', 'Juancito', 2028, 8.20)

1 row(s) affected

INSERCIÓN DE DATOS SCRIPT:

USE VENTA_DE_PELICULAS;

-- Inserción de datos en la tabla Peliculas

```
INSERT INTO Peliculas (titulo, genero, director, año, precio) VALUES
('Back to the Future', 'Ciencia ficción', 'Robert Zemeckis', 1985, 9.99),
('The Breakfast Club', 'Comedia dramática', 'John Hughes', 1985, 12.99),
('Raiders of the Lost Ark', 'Aventura', 'Steven Spielberg', 1981, 14.99),
('Blade Runner', 'Ciencia ficción', 'Ridley Scott', 1982, 11.99),
('The Terminator', 'Acción y ciencia ficción', 'James Cameron', 1984, 10.99),
('E.T. the Extra-Terrestrial', 'Ciencia ficción', 'Steven Spielberg', 1982, 12.99),
('Ghostbusters', 'Comedia de ciencia ficción', 'Ivan Reitman', 1984, 9.99),
('Scarface', 'Drama criminal', 'Brian De Palma', 1983, 13.99);
```

-- Inserción de datos en la tabla Clientes

```
INSERT INTO Clientes (nombre, apellido, edad, direccion, celular, correo) VALUES  
( 'Carlos', 'Gutiérrez', 35, 'Calle 123', '123456789', 'carlos@example.com'),  
( 'Ana', 'Martínez', 28, 'Avenida 456', '987654321', 'ana@example.com');
```

-- Inserción de datos en la tabla Transacciones

```
INSERT INTO Transacciones (id_cliente, fecha, metodo_pago) VALUES  
(1, '2024-03-22', 'Tarjeta de crédito'),  
(2, '2024-03-21', 'Efectivo');
```

-- Inserción de datos en la tabla Detalles_Transaccion

```
INSERT INTO Detalles_Transaccion (id_transaccion, id_pelicula, cantidad, precio_unitario)  
VALUES  
(1, 1, 1, 9.99),  
(2, 2, 2, 12.99),  
(2, 3, 1, 14.99);
```

-- Inserciones para la tabla Proveedores

```
INSERT INTO Proveedores (rut, nombre, telefono, correo) VALUES  
(12345678, 'Proveedor Uno', '123456789', 'proveedor1@example.com'),  
(87654321, 'Proveedor Dos', '987654321', 'proveedor2@example.com');
```

-- Inserciones para la tabla Compra_Proveedores

```
INSERT INTO Compra_Proveedores (id_proveedor, fecha, metodo_pago) VALUES  
(1, '2024-03-22', 'Transferencia'),  
(2, '2024-03-23', 'Tarjeta');
```

-- Inserciones para la tabla Detalles_Compra_Proveedores

```
INSERT INTO Detalles_Compra_Proveedores (id_transaccion, id_pelicula, cantidad,  
precio_unitario) VALUES  
(1, 1, 5, 2.50),
```

(2, 2, 3, 2.75),
(3, 3, 2, 2.25),
(4, 1, 4, 3.00);

TRIGGER:

1: Tal cual vimos en clase implemente un trigger que al intentar insertar un cliente con mail repetido devuelve un error con el mensaje “El mail ya esta registrado”

CODIGO:

```
-- tTRIGGERS

USE VENTA_DE_PELICULAS;

-- Validar que el email sea único:

DELIMITER //

CREATE TRIGGER validar_correo

BEFORE INSERT ON CLIENTES

FOR EACH ROW

BEGIN

    DECLARE correo_existente INT;

    SELECT COUNT(*) INTO correo_existente

    FROM CLIENTES

    WHERE correo = NEW.correo;

    IF correo_existente > 0 THEN

        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El email ya está registrado';

    END IF;

END;

//
```

DELIMITER ;

```
call venta_de_peliculas.insertar_cliente('Pedro', 'Segundo', 2, 'aca la vuelta', '11233',  
'carlos@example.com');
```

SELECT * FROM CLIENTES;

SHOW TRIGGERS LIKE 'validar_correo';

PRUEBAS:

```
call venta_de_peliculas.insertar_cliente('Pedro', 'Segundo', 2, 'aca la vuelta', '11233', 'carlos@example.com');
```

9 17:49:29 call venta_de_peliculas.insertar_cliente('Pedro', 'Segundo', 2, 'aca la vuelta', '11233', 'carlos@example.com')

Error Code: 1644. El email ya está registrado

