

Gestor de Ventas y de Satisfacción del Cliente (Tienda de Celulares)

Integrantes:

Amaya Brenda

Giménez Florencia

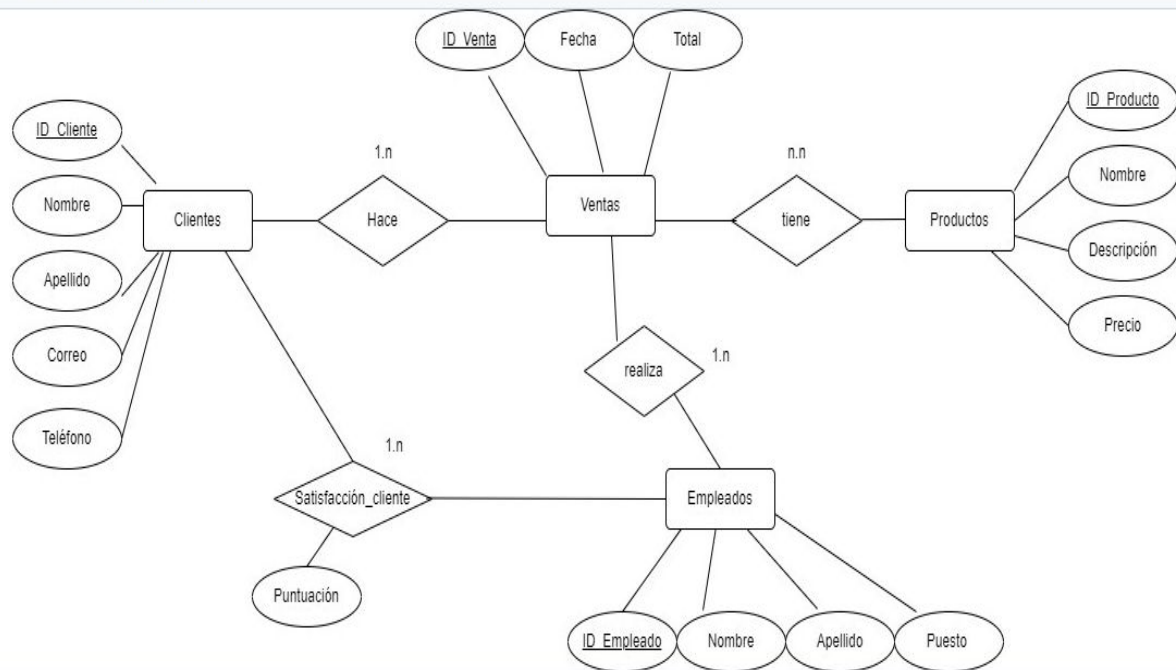
Lazarte José Augusto

Lelli Florencia

Linares Cecilia

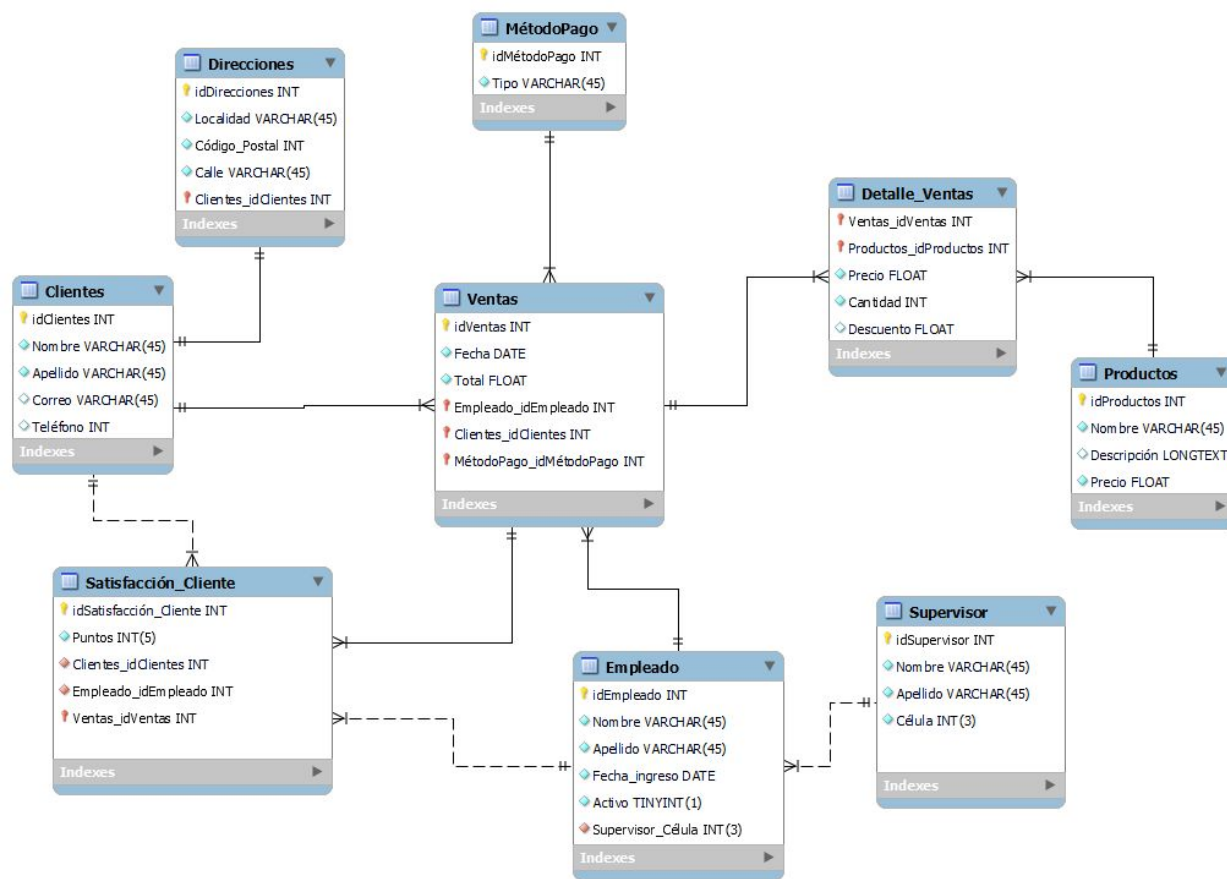
14 de Junio de 2024
Córdoba

DER



Iniciamos nuestro desarrollo de BBDD, mediante el diseño de un DER, notación de Chen con las entidades principales: Cliente, Ventas, Productos y Empleados, con sus respectivos atributos. De entre las relaciones merece una mención especial Satisfacción_cliente, la cual luego de redefiniciones tomaría centralidad de entidad, ya que la intención, no es solo llevar registros, si no medir el rendimiento en general de los empleados.

CROW FOOT



Adentrándonos en el traspaso del DER al Modelo Relacional, a partir de la definición de cardinalidades, definimos como tablas (antes entidades) del proyecto: Clientes, Direcciones, Ventas, Métodos de pago, Detalle Ventas, Productos, Supervisor, Empleado y Satisfacción_Cliente, con sus correspondientes Columnas (antes atributos) y las restricciones sobre cada uno de los datos que las sustentan, Claves primarias (PK), Claves foráneas (FK), not null, Unique

TABLAS DE LA BASE DE DATOS

SELECT * FROM clientes

	idClientes	Nombre	Apellido	Correo	Teléfono
▶	1	Olivia	Gonzalez	olivia@ejemplo.com	NULL
	2	Agustina	Molina	agusmolina@ejemplo.com	22211
	3	Juan	Perez	juan@ejemplo.com	NULL
	4	Victoria	Martinez	victoria@ejemplo.com	101010
	5	Santiago	Cepeda	santiago@ejemplo.com	555000
	6	Pedro	Garcia	pedro@ejemplo.com	NULL

SELECT * FROM direcciones

	idDirecciones	Localidad	Código_Postal	Calle	Clientes_idClientes
▶	1	Córdoba	5030	Belgrano 1012	1
	2	Córdoba	5000	Lima 800	2
	3	Bell Ville	2550	General Paz 745	3
	4	Córdoba	5002	Sam Martin 300	4
	5	Córdoba	5002	Caseros 4510	5
	6	Villa María	5900	9 de julio 1550	6
⬆	NULL	NULL	NULL	NULL	NULL

SELECT * FROM detalle_ventas

	Ventas_idVentas	Productos_idProductos	Precio	Cantidad	Descuento
▶	1	1	175999	1	NULL
	2	2	247999	2	NULL
	3	6	399999	1	NULL
	4	3	202499	3	NULL
	5	4	449999	1	NULL
	6	6	39999	3	NULL
*	NULL	NULL	NULL	NULL	NULL

SELECT * FROM empleado

	idEmpleado	Nombre	Apellido	Fecha_ingreso	Supervisor_Célula
▶	1	Eduardo	Lopez	2024-01-01	2
	2	Lucia	Acuña	2020-01-02	3
	3	Rocio	Pacheco	2019-05-08	1
	4	Diego	Zapata	2022-09-01	1
	5	Jorge	Suarez	2024-06-05	3
	6	Andrea	Romero	2019-05-08	2
*	NULL	NULL	NULL	NULL	NULL

TABLAS DE LA BASE DE DATOS

SELECT * FROM método_pago

	idMétodoPago	Tipo
▶	4	Efectivo
	1	Tarjeta de Crédito
	2	Tarjeta de Débito
	3	Transferencia
★	NULL	NULL

SELECT * FROM satisfacción_cliente

	idSatisfacción_Cliente	Puntos	Cientes_idClientes	Empleado_idEmpleado	Ventas_idVentas
▶	1	5	2	1	1
	2	1	4	2	2
	3	2	5	1	3
	4	5	1	3	4
	5	5	3	4	5
	6	1	2	2	6
★	NULL	NULL	NULL	NULL	NULL

SELECT * FROM productos

idProductos	Nombre	Descripción	Precio
1	Moto E22 64 GB	NULL	175999
2	Moto G23 128 GB	NULL	247999
3	Samsung Galaxy A04 64 GB	Pantalla Infinity-V de 6,5 pulgadas.	202499
4	Samsung Galaxy A34 5G 128 GB	NULL	449999
5	Moto G41 128 GB	Pantalla Full HD+ de 6,5 pulgadas.	279999
6	Samsung Galaxy A15 128GB	NULL	399999
7	Samsung Galaxy A54 5G 256GB	Características de última generación.	999999
8	Moto G13 128GB	NULL	269999
NULL	NULL	NULL	NULL

SELECT * FROM supervisor

	idSupervisor	Nombre	Apellido	Célula
▶	1	Verónica	Aguirre	1
	2	David	Arias	2
	3	Maria	Escalante	3
★	NULL	NULL	NULL	NULL

TABLAS DE LA BASE DE DATOS



SELECT * FROM ventas

	idVentas	Fecha	Total	Empleado_idEmpl	Clientes_idClie	MétodoPago_idMétodo
▶	1	2024-01-02	175999	1	2	2
	2	2024-01-04	495998	2	4	1
	3	2024-02-01	399999	1	5	3
	4	2024-02-10	607497	3	1	3
	5	2024-02-08	449999	4	3	4
	6	2024-01-10	1200000	2	2	1
*	NULL	NULL	NULL	NULL	NULL	NULL

INSERT

Comando utilizado para insertar nuevos registros en una tabla de una base de datos. Esta sentencia permite agregar datos a la tabla, creando nuevas filas con valores específicos en las columnas correspondientes.

La sintaxis de SQL:

```
INSERT INTO nombre_de_tabla (columna_1, columna_2, ...)  
VALUES (valor_1, valor_2, ...);
```

Un ejemplo que aplicamos a nuestra base de datos:

```
INSERT INTO clientes (Nombre, Apellido, Correo, Teléfono)  
VALUES  
    ("Olivia", "Gonzalez", "olivia@ejemplo.com", NULL),  
    ("Agustina", "Molina", "agusmolina@ejemplo.com", 22211),  
    ("Juan", "Perez", "juan@ejemplo.com", NULL),  
    ("Victoria", "Martinez", "victoria@ejemplo.com", 101010),  
    ("Santiago", "Cepeda", "santiago@ejemplo.com", 555000),  
    ("Pedro", "Garcia", "pedro@ejemplo.com", NULL);
```

Result Grid						Filter Rows:	Edit:		
	idClientes	Nombre	Apellido	Correo	Teléfono				
▶	1	Olivia	Gonzalez	olivia@ejemplo.com	NULL				
	2	Agustina	Molina	agusmolina@ejemplo.com	22211				
	3	Juan	Perez	juan@ejemplo.com	NULL				
	4	Victoria	Martinez	victoria@ejemplo.com	101010				
	5	Santiago	Cepeda	santiago@ejemplo.com	555000				
	6	Pedro	Garcia	pedro@ejemplo.com	NULL				
▲	NULL	NULL	NULL	NULL	NULL				

UPDATE

Comando utilizado para modificar o actualizar los registros existentes en una tabla de una base de datos.

La sintaxis de SQL:

UPDATE nombre_de_tabla

SET columna1 = nuevo_valor1, columna2 = nuevo_valor2, ...

WHERE condicion;

Un ejemplo que aplicamos a nuestra base de datos

UPDATE clientes

SET nombre = "Pablo", Correo = "pablo@ejemplo.com"

WHERE idClientes = 6 ;

	idClientes	Nombre	Apellido	Correo	Teléfono
▶	1	Olivia	Gonzalez	olivia@ejemplo.com	NULL
	2	Agustina	Molina	agusmolina@ejemplo.com	22211
	3	Juan	Perez	juan@ejemplo.com	NULL
	4	Victoria	Martinez	victoria@ejemplo.com	101010
	5	Santiago	Cepeda	santiago@ejemplo.com	555000
→	6	Pablo	Garcia	pablo@ejemplo.com	NULL
	NULL	NULL	NULL	NULL	NULL

UPDATE EMPLEADO

SET Fecha_ingreso = '2021-01-02'

WHERE idEmpleado = 2 ;

	idEmpleado	Nombre	Apellido	Fecha_ingreso	Supervisor_Célula
▶	1	Eduardo	Lopez	2024-01-01	2
	2	Lucia	Acuña	2021-01-02 ←	3
	3	Rocio	Pacheco	2019-05-08	1
	4	Diego	Zapata	2022-09-01	1
	5	Jorge	Suarez	2024-06-05	3
	6	Andrea	Romero	2019-05-08	2

DELETE

Comando utilizado para eliminar registros existentes en una tabla en una base de datos.
Permite eliminar una o varias filas.

La sintaxis de SQL:

DELETE FROM nombre_de_tabla

WHERE condicion,

Un ejemplo que aplicamos a nuestra base de datos:

DELETE FROM clientes

WHERE idClientes = 3;

Result Grid					
		Filter Rows:		Edit:	
	idClientes	Nombre	Apellido	Correo	Teléfono
▶	1	Olivia	Gonzalez	olivia@ejemplo.com	NULL
	2	Agustina	Molina	agusmolina@ejemplo.com	22211
	4	Victoria	Martinez	victoria@ejemplo.com	101010
	5	Santiago	Cepeda	santiago@ejemplo.com	555000
	6	Pablo	Garcia	pablo@ejemplo.com	NULL
✱	NULL	NULL	NULL	NULL	NULL

CONSULTAS

1. Una sola tabla. (mostrando todos los datos).

Descripción:

La consulta muestra una tabla con todos los registros en todas las columnas. Utiliza el “*” para que muestre todas las columnas. En este caso la consulta muestra todos los registros de la tabla “productos”.

Sintaxis SQL:

```
SELECT * FROM productos;
```

	idProductos	Nombre	Descripción	Precio
▶	1	Moto E22 64 GB	NULL	175999
	2	Moto G23 128 GB	NULL	247999
	3	Samgsung Galaxy A04 64 GB	Pantalla Infinity-V de 6,5 pulgadas.	202499
	4	Samsung Galaxy A34 5G 128 GB	NULL	449999
	5	Moto G41 128 GB	Pantalla Full HD+ de 6,5 pulgadas.	279999
	6	Samsung Galaxy A15 128GB	NULL	399999
	7	Samsung Galaxy A54 5G 256GB	Características de última generación.	999999
	8	Moto G13 128GB	NULL	269999
*	NULL	NULL	NULL	NULL

CONSULTAS

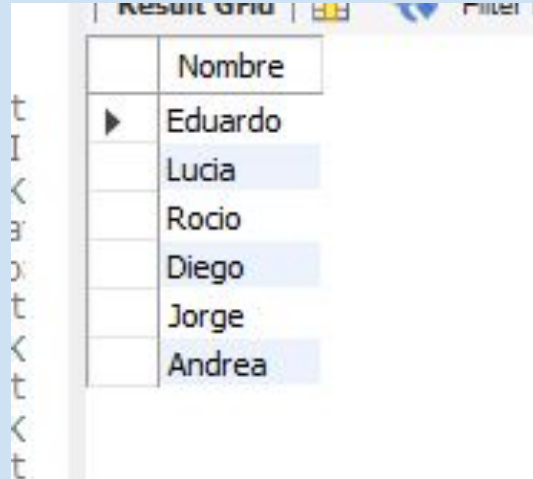
2. Una sola tabla (mostrando algunas columnas).

Descripción:

La consulta esta vez delimita las columnas que va a mostrar reemplazando el “*” por el paréntesis que indica cuáles son las columnas seleccionadas para mostrar. En este caso la consulta muestra de la tabla “empleado” los registros de la columna Nombre.

Sintaxis SQL:

```
SELECT Nombre FROM empleado;
```



The image shows a screenshot of a database application window titled "Result Grid". It displays the results of a SQL query. The table has one column labeled "Nombre" and six rows of data. The first row is "Eduardo", followed by "Lucia", "Rocio", "Diego", "Jorge", and "Andrea". The rows are alternatingly highlighted with light blue and white backgrounds. To the left of the table, there is a vertical list of letters: t, I, <, a, O, t, <, t, <, t.

Nombre
Eduardo
Lucia
Rocio
Diego
Jorge
Andrea

CONSULTAS

3. Una sola tabla con where.

Descripción:

El where funciona como un filtro para que muestre solo los registros que cumplan cierta condición. En este caso, la consulta va a mostrar de la tabla Productos los registros de la columna Nombre que contengan la palabra “Samsung”.

Sintaxis SQL:

```
SELECT * FROM productos
```

```
WHERE Nombre LIKE '%samsung%';
```

	idProductos	Nombre	Descripción	Precio
▶	3	Samsung Galaxy A04 64 GB	Pantalla Infinity-V de 6,5 pulgadas.	202499
	4	Samsung Galaxy A34 5G 128 GB	NULL	449999
	6	Samsung Galaxy A15 128GB	NULL	399999
	7	Samsung Galaxy A54 5G 256GB	Características de última generación.	999999
✱	NULL	NULL	NULL	NULL

CONSULTAS

4. Una sola tabla con where utilizando between.

Descripción:

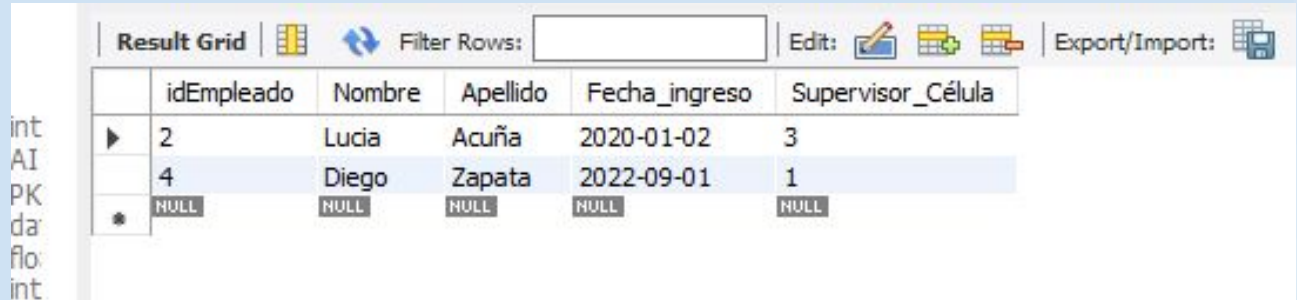
En nuestra consulta, la tabla muestra los empleados con fecha de ingreso superior a un año, las fechas para calcular el año usan el between.

Sintaxis SQL:

```
SELECT * FROM empleado
```

```
WHERE Fecha_ingreso BETWEEN '20-01-01' AND '23-06-12'
```

```
ORDER BY Fecha_ingreso;
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid displays the results of a SQL query. The columns are 'idEmpleado', 'Nombre', 'Apellido', 'Fecha_ingreso', and 'Supervisor_Célula'. The first two rows are highlighted in blue. The third row is highlighted in grey and contains NULL values. The interface also includes a 'Filter Rows' search bar, an 'Edit' button with a pencil icon, and an 'Export/Import' button with a document icon.

	idEmpleado	Nombre	Apellido	Fecha_ingreso	Supervisor_Célula
▶	2	Lucia	Acuña	2020-01-02	3
	4	Diego	Zapata	2022-09-01	1
*	NULL	NULL	NULL	NULL	NULL

CONSULTAS

5. Una sola tabla con where utilizando limit.

Descripción:

El limit pone un tope a la cantidad de registros que va a mostrar la consulta. En nuestra consulta muestra solo 3 de las ventas que cuentan con un total mayor a 200.000

Sintaxis SQL:

SELECT * FROM ventas

WHERE total > 200000

LIMIT 3;

Result Grid		Filter Rows:		Edit:		Export/Import:		Wrap Cell
	idVentas	Fecha	Total	Empleado_idEmpleado	Cientes_idClientes	MétodoPago_idMétodoPago		
	2	2024-01-04	495998	2	4	1		
	3	2024-02-01	399999	1	5	3		
	4	2024-02-10	607497	3	1	3		
	NULL	NULL	NULL	NULL	NULL	NULL		

CONSULTAS

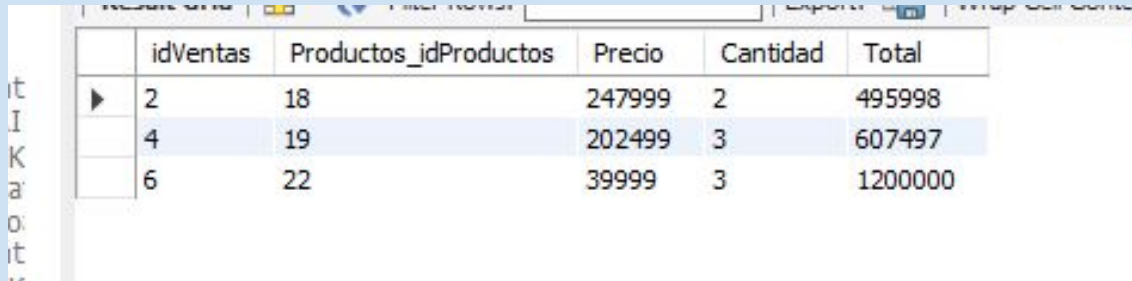
6. Más de una tabla con inner join.

Descripción:

El inner join permite consultar otras tablas. Nuestra consulta muestra las ventas con más de un producto comprado usando la tabla detalle_ventas.

Sintaxis SQL:

```
SELECT ventas.idVentas, detalle_ventas.Productos_idProductos, detalle_ventas.Precio, detalle_ventas.Cantidad, ventas.Total  
FROM detalle_ventas  
INNER JOIN ventas ON detalle_ventas.Ventas_idVentas = ventas.idVentas  
WHERE detalle_ventas.Cantidad > 1;
```



The screenshot shows a database query result window. The query is an inner join between the 'ventas' and 'detalle_ventas' tables. The result set contains three rows, each representing a sale with multiple products. The columns are: idVentas, Productos_idProductos, Precio, Cantidad, and Total. The first row shows sale 2 with product 18 at price 247999, quantity 2, and total 495998. The second row shows sale 4 with product 19 at price 202499, quantity 3, and total 607497. The third row shows sale 6 with product 22 at price 39999, quantity 3, and total 1200000.

	idVentas	Productos_idProductos	Precio	Cantidad	Total
▶	2	18	247999	2	495998
	4	19	202499	3	607497
	6	22	39999	3	1200000

CONSULTAS

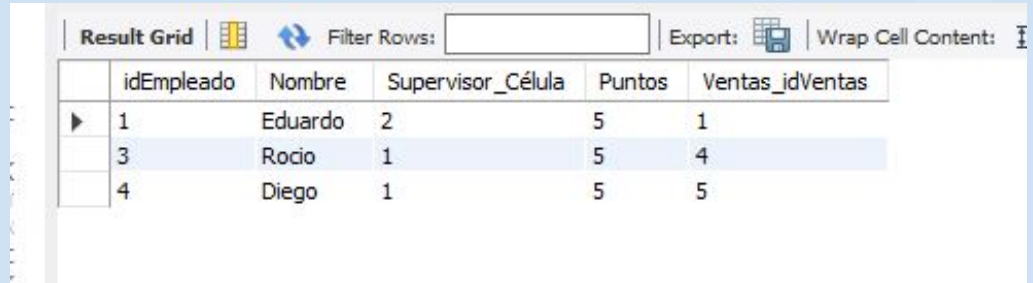
7. Más de una tabla con inner join y filtros.

Descripción:

Nuestra consulta muestra a los empleados por su id, nombre y célula pero que tienen un puntaje de 5.

Sintaxis SQL:

```
SELECT empleado.idEmpleado, empleado.Nombre, empleado.Supervisor_Célula, satisfacción_cliente.Puntos,  
satisfacción_cliente.Ventas_idVentas  
FROM satisfacción_cliente  
INNER JOIN empleado ON satisfacción_cliente.Empleado_idEmpleado = empleado.idEmpleado  
WHERE satisfacción_cliente.Puntos > 4  
ORDER BY empleado.idEmpleado;
```



The screenshot shows a 'Result Grid' window with a toolbar at the top containing icons for grid view, refresh, filter rows, export, and wrap cell content. The grid displays the results of the SQL query, showing three rows of employee data where the score is greater than 4. The columns are idEmpleado, Nombre, Supervisor_Célula, Puntos, and Ventas_idVentas.

	idEmpleado	Nombre	Supervisor_Célula	Puntos	Ventas_idVentas
▶	1	Eduardo	2	5	1
	3	Rocio	1	5	4
	4	Diego	1	5	5

CONSULTAS

8. Una sola tabla con group by usando alguna función agregada.

Descripción:

Nuestra consulta muestra la sumatoria de todas las ventas que se hicieron por mes. Establece que el mes de la fecha va a mostrar una columna llamada “Mes”, hace la sumatoria del total de las ventas y las muestra en una columna “Ventas_totales”. Group By agrupa las ventas por mes.

Sintaxis SQL:

```
SELECT MONTH(Fecha) AS Mes, SUM(Total) AS Ventas_totales
```

```
FROM ventas
```

```
GROUP BY MONTH(Fecha);
```



The screenshot shows a 'Result Grid' window with a toolbar at the top containing a grid icon, a refresh icon, and a 'Filter Rows:' label. The grid displays two columns: 'Mes' and 'Ventas_totales'. The first row shows '1' for the month and '1871994' for the total sales. The second row shows '2' for the month and '1457495' for the total sales.

	Mes	Ventas_totales
▶	1	1871994
	2	1457495