

# Proyecto final

- Autor: Hector Nicolas Atencio
- Curso: SQL
- Academia: Coderhouse

## INDICE

### 1.0 Primera entrega:

- 1.1 Descripción de la temática página 3.
- 1.2 Diagrama entidad relación página 4.
- 1.3 Reverse engineer página 5.
- 1.4 Listado de tablas página 6.

### 2.0 Segunda entrega:

- 2.1 Vistas página 10.
- 2.2 Funciones página 13.
- 2.3 Stored procedures página 14.
- 2.4 Triggers página 16.
- 2.5 Link script página 18.

### 3.0 Tercera entrega:

- 3.1 Informes página 19
- 3.2 Herramientas utilizadas página 21.

# PRIMERA ENTREGA

## 1.1 Descripción de la temática

### Introducción

El proyecto consiste en una base de datos que almacena la información de una fábrica de pastas. Esta dividido en tres partes, primera entrega, segunda entrega y tercera entrega, las cuales debieron cumplir requisitos para su aprobación.

### Objetivos

El objetivo del proyecto es almacenar toda la información necesaria para el correcto funcionamiento de la fábrica de pastas y demostrar el conocimiento adquirido durante la cursada.

### Situación problemática

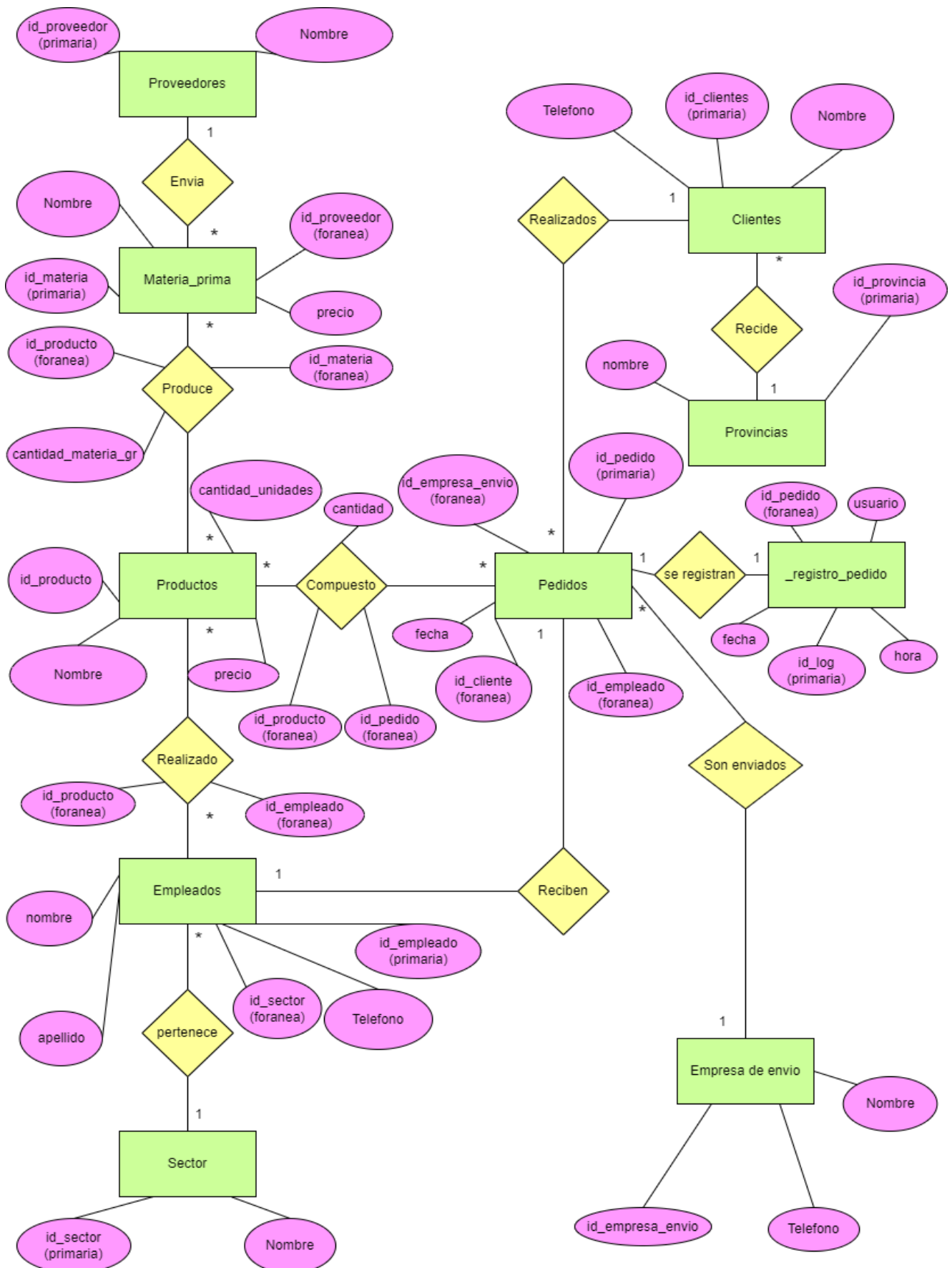
La base de datos implementada funciona para obtener información y saber sobre:

- El precio de la materia prima y proveedores que la venden.
- Saber que vendedores son los responsables de cada venta.
- Saber cuáles son los productos más vendidos.
- Saber cuáles productos generan más ganancias.
- Saber cuánto dinero se recauda con cada venta.
- Saber que empleados son los responsables de la realización de los distintos productos.
- Almacenar la información de los clientes
- Conocer que empresa de envío es la responsable del envío de cada venta.

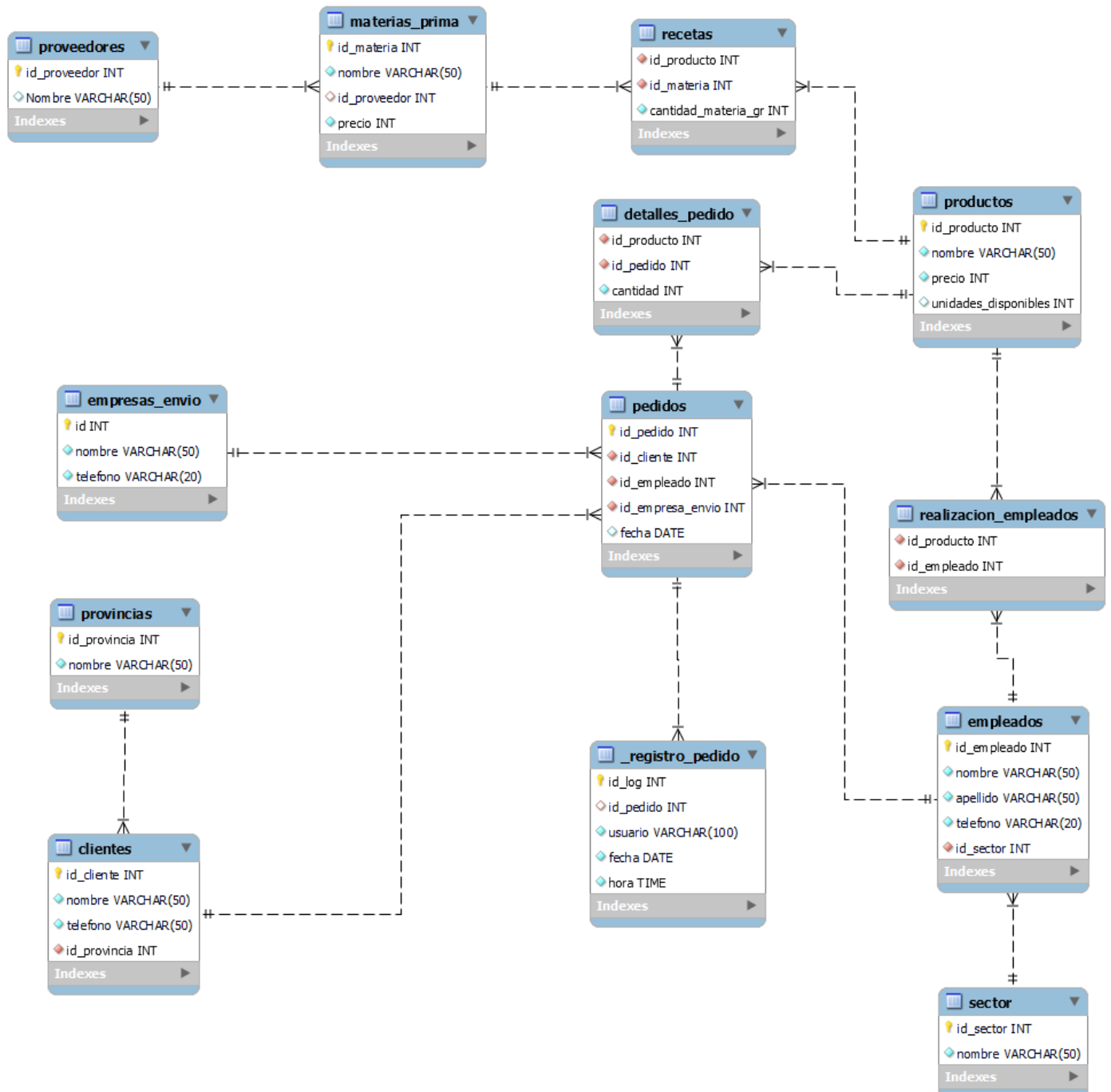
### Modelo de negocio

El negocio es una fábrica de pastas que se encarga de la realización de diferentes tipos de pastas como fideos, ñoquis, rabioles y más productos, además se encarga de la venta a distintos comercios y super mercados de las distintas provincias en Argentina. Para el envío de los productos se contratan empresas que se dediquen a ello.

## 1.2 Diagrama entidad relación



## 1.3 Reverse Engineer



## 1.4 Listado de tablas

Tabla	proveedor				
Descripción	Contiene información sobre los proveedores.				
Column	Type	Not null	Unique	Notes	Key
Id_proveedor	int	verdadero	verdadero	Id del proveedor	PK
nombre	varchar	verdadero		Nombre del proveedor	
teléfono	varchar	verdadero		Contacto del proveedor	

Tabla	materia_prima				
Descripción	Contiene información sobre la materia prima comprada a los proveedores				
Column	Type	Not null	Unique	Notes	Key
Id_materia	int	verdadero	verdadero	Id de materia prima	PK
nombre	varchar	verdadero	verdadero	Nombre de la materia prima	
precio	int	verdadero		Costo de la materia prima	
Id_proveedor	int	verdadero		Id del proveedor	FK

Tabla	receta				
Descripción	Tabla intermedia entre MATERIA_PRIMA y PRODUCTOS donde se detalla cuales materias primas se utilizan en la preparación de los productos.				
Column	Type	Not null	Unique	Notes	Key
Id_producto	int	verdadero		Id del producto	FK
Id_materia	int	verdadero		Id de materia prima	FK
cantidad	int	verdadero		Cantidad en gramos de materia nesositada	

Tabla	productos				
Descripción	Contiene información sobre los productos que se realizan en la fábrica.				
Column	Type	Not null	Unique	Notes	Key
Id_producto	int	verdadero	verdadero	Id del producto	PK
nombre	varchar	verdadero	verdadero	Nombre del producto	
precio	int	verdadero		Precio en el momento de vender el producto.	
unidades_disponibles	int			Cantidad disponibles del producto.	

Tabla	realización_empleados				
Descripción	Tabla intermedia entre PRODUCTOS y EMPLEADOS donde informa cuales empleados se encargan de la produccion de los distintos productos.				
Column	Type	Not null	Unique	Notes	Key
Id_producto	int	verdadero		Id del producto	FK
Id_empleado	int	verdadero		Id del empleado	FK

Tabla	empleados				
Descripción	Contiene informacion sobre los empleados de la fabrica.				
Column	Type	Not null	Unique	Notes	Key
Id_empleado	int	verdadero	verdadero	Id de empleado	PK
nombre	varchar	verdadero		Nombre empleado	
apellido	varchar	verdadero		Apellido empleado	
teléfono	varchar	verdadero	verdadero	Contacto del empleado	
Id_sector	int	verdadero		Id del sector	FK

Tabla	sector				
Descripción	Contiene información sobre los sectores de la empresa.				
Column	Type	Not null	Unique	Notes	Key
Id_sector	int	verdadero	verdadero	Id del sector	PK
nombre	varchar	verdadero	verdadero	Nombre del sector	

Tabla	detalles_pedido				
Descripción	Tabla intermedia entra la tabla PRODUCTOS y PEDIDOS donde se detalla la información de un pedido.				
Column	Type	Not null	Unique	Notes	Key
Id_pedido	int	verdadero		Id pedido	FK
Id_producto	int	verdadero		Id del producto	FK
cantidad	int	verdadero		Cantidad de producto	

Tabla	pedidos				
Descripción	Contiene informacion sobre los pedidos.				
Column	Type	Not null	Unique	Notes	Key
Id_pedido	int	verdadero	verdadero	Id del pedido	PK
Id_cliente	int	verdadero		Id del cliente	FK
Id_empleado	int	verdadero		Id empleado que realizo la venta	FK
Id_empresa_envio	int	verdadero		Id empresa a cargo del envio	FK
fecha	date			Fecha de la realización de la venta	

Tabla	clientes				
Descripción	Contiene información sobre los clientes.				
Column	Type	Not null	Unique	Notes	Key
Id_cliente	int	verdadero	verdadero	Id del cliente	PK
nombre	varchar	verdadero		Nombre cliente	
teléfono	varchar	verdadero		Contacto de cliente	
Id_provincia	int	verdadero		Id de provincia	FK

Tabla	Provincia				
Descripción	Contiene los id de cada provincia				
Column	Type	Not null	Unique	Notes	Key
Id_provincia	int	verdadero	verdadero	Id provincia	PK
nombre	varchar	verdadero	verdadero	Nombre de provincia	

Tabla	empresa_envio				
Descripción	Contiene información sobre las empresa que se encarga de los envíos.				
Column	Type	Not null	Unique	Notes	Key
Id_empresa	int	verdadero	verdadero	Id de empresa a cargo del envío.	PK
nombre	varchar	verdadero	verdadero	Nombre de la empresa	
teléfono	varchar	verdadero	verdadero	Contacto de la empresa	



Tabla	<u>_registro_pedido</u>				
Descripción	Tabla de auditoria, registra que usuario registró un nuevo pedido.				
Column	Type	Not null	Unique	Notes	Key
Id_log	int	verdadero	verdadero	Id de auditoria	PK
Id_pedido	int	verdadero	verdadero	Id del pedido	
usuario	varchar	verdadero		Usuario que registró el pedido	
fecha	date	verdadero		Fecha de la inserción del pedido	
hora	time	verdadero		Hora de la inserción del pedido	

## SEGUNDA ENTREGA

### 2.1 Vistas

- Vista 1: recaudado\_vendedores

El objetivo de la vista es obtener una tabla donde se muestra cuanto dinero a sido recaudado por cada vendedor al realizar ventas.

```
209 • CREATE OR REPLACE VIEW recaudado_vendedores AS
210   (SELECT em.nombre,em.apellido,SUM(pr.precio * de.cantidad) as recaudado
211     FROM empleados em
212     INNER JOIN pedidos pe ON (pe.id_empleado = em.id_empleado)
213     INNER JOIN detalles_pedido de ON (de.id_pedido = pe.id_pedido)
214     INNER JOIN productos pr ON (pr.id_producto = de.id_producto)
215     GROUP BY em.id_empleado
216   );
217 • select * from recaudado_vendedores;
```

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

	nombre	apellido	recaudado
▶	Lucas	Aindriu	615300
	Mariana	Jhonson	592600
	Fields	Sapson	315580

- Vista 2: cantidad\_productos\_vendidos

El objetivo es tener una tabla donde se muestre la cantidad total vendida de cada producto.

```
228 • CREATE OR REPLACE VIEW cantidad_productos_vendidos AS
229   (
230     SELECT p.id_producto,p.nombre,SUM(d.cantidad) as cantidad_vendidos
231     FROM productos p
232     INNER JOIN detalles_pedido d ON(d.id_producto = p.id_producto)
233     GROUP BY p.id_producto
234   );
235 • select * from cantidad_productos_vendidos;
```

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

	id_producto	nombre	cantidad_vendidos
▶	1	Fideos	990
	2	Fideos de morron	542
	3	ñoquis	1024
	4	ravioles espinaca	698
	5	ravioles jamon y queso	790

- Vista 3: recaudado\_por\_producto

El objetivo es obtener una tabla que muestre lo recaudado en ventas de cada producto.

```

237 • CREATE OR REPLACE VIEW recaudado_por_producto AS
238 (
239     SELECT p.id_producto, p.nombre, SUM(d.cantidad*p.precio) as recaudado
240     FROM productos p
241     INNER JOIN detalles_pedido d ON(d.id_producto = p.id_producto)
242     GROUP BY p.id_producto
243 );
244 • select * from recaudado_por_producto;

```

Result Grid

	id_producto	nombre	recaudado
▶	1	Fideos	297000
	2	Fideos de morron	189700
	3	ñoquis	327680
	4	ravioles espinaca	314100
	5	ravioles jamon y queso	395000

- Vista 4: gastos\_por\_producto

El propósito de la vista es obtener una tabla donde muestre cuanto ha costado las ventas totales de los productos.

```

246 • CREATE OR REPLACE VIEW gastos_por_producto AS
247 (
248     SELECT p.id_producto, p.nombre, SUM(d.cantidad*((r.cantidad_materia_gr*m.precio)/1000)) as gastos
249     FROM productos p
250     INNER JOIN detalles_pedido d ON(d.id_producto = p.id_producto)
251     INNER JOIN recetas r ON (r.id_producto = p.id_producto)
252     INNER JOIN materias_prima m ON (m.id_materia = r.id_materia)
253     GROUP BY p.id_producto
254 );
255 • select * from gastos_por_producto;

```

Result Grid

	id_producto	nombre	gastos
▶	1	Fideos	127215.0000
	2	Fideos de morron	58265.0000
	3	ñoquis	17920.0000
	4	ravioles espinaca	26175.0000
	5	ravioles jamon y queso	71100.0000

- Vista 5 beneficio\_netoventa\_producto

El objetivo de la vista es obtener una tabla donde muestre el beneficio económico neto de la venta de cada producto, donde al total de lo recaudado por su venta se le resta lo que cuesta generar esa cantidad de producto.

```

257 • CREATE OR REPLACE VIEW beneficio_netoventa_producto AS
258 (
259     SELECT re.id_producto, re.nombre, re.recaudado - ga.gastos as ingreso_netov
260     FROM recaudado_por_producto re
261     INNER JOIN gastos_por_producto ga ON (re.id_producto = ga.id_producto)
262     -- gastos_por_producto no es una tabla, es una vista creada con anterioridad
263 );
264 • select * from beneficio_netoventa_producto;

```

Result Grid

	id_producto	nombre	ingreso_netov
▶	1	Fideos	169785.0000
	2	Fideos de morron	131435.0000
	3	ñoquis	309760.0000
	4	ravioles espinaca	287925.0000
	5	ravioles jamon y queso	323900.0000

- Vista 6: provincia\_ventas

El objetivo de la vista es obtener una tabla que muestre la cantidad de ventas en cada provincia.

```

219 • CREATE OR REPLACE VIEW provincia_ventas AS
220 (SELECT p.nombre, COUNT(DISTINCT pe.id_pedido) as ventas_provincia
221 FROM provincias p
222 INNER JOIN clientes c ON (c.id_provincia = p.id_provincia)
223 INNER JOIN pedidos pe ON (pe.id_cliente = c.id_cliente)
224 GROUP BY p.id_provincia
225 );
226 • select * from provincia_ventas;

```

Result Grid

	nombre	ventas_provincia
▶	Mendoza	5
	Buenos Aires	2
	San Juan	3
	San Luis	2

## 2.2 Funciones

- Función 1: recaudado\_por\_vendedor

La función recibe como parámetro el id de un empleado y devuelve la cantidad de ventas totales realizadas por dicho empleado.

```
270 DELIMITER $$
271 • CREATE FUNCTION recaudado_por_vendedor(id INT) RETURNS INT
272 READS SQL DATA
273 BEGIN
274     DECLARE recaudado_vendedor INT;
275     SELECT re.recaudado INTO recaudado_vendedor
276     FROM empleados e
277     INNER JOIN recaudado_vendedores re ON re.nombre = e.nombre AND re.apellido = e.apellido
278     -- recaudado_vendedores no es una tabla, es una vista.
279     WHERE id_empleado = id;
280     RETURN recaudado_vendedor;
281 END $$
282 select recaudado_por_vendedor(6) as total_ventas_empleado_id_6;
```

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

total_ventas_empleado_id_6
592600

- Función 2: sector\_del\_empleado

Recibe como parámetro el id de un empleado y devuelve el sector en donde desempeña sus tareas.

```
286 DELIMITER $$
287 • CREATE FUNCTION sector_del_empleado(id INT) RETURNS VARCHAR(50)
288 READS SQL DATA
289 BEGIN
290     DECLARE nombre_sector VARCHAR(50);
291     SELECT sector.nombre INTO nombre_sector
292     FROM empleados
293     INNER JOIN sector on sector.id_sector = empleados.id_sector
294     WHERE id_empleado = id;
295     RETURN nombre_sector;
296 END $$
297 select sector_del_empleado(2) as sector;
```

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

sector
Recursos humanos

## 2.3 Stored Procedures

- Procedure 1 nuevo\_empleado

Insertar un nuevo empleado sin la necesidad de usar INSERT.

```
304 DELIMITER $$
305 • CREATE PROCEDURE `nuevo_empleado` (
306     IN nombre_e VARCHAR(50),
307     IN apellido_e VARCHAR(50),
308     IN telefono_e VARCHAR(20),
309     IN id_sector_e INT
310 )
311 BEGIN
312     INSERT INTO empleados(nombre,apellido,telefono,id_sector)
313     values (nombre_e,apellido_e,telefono_e,id_sector_e);
314 END$$
```

- Procedure 2 nuevo\_pedido\_mas\_detalle

El objetivo es agregar datos en dos tablas, la tabla pedidos y la tabla detalles\_pedido, para ello se pasa como parametro al stored nuevo\_pedido\_mas\_detalle el id del cliente, el id del empleado a cargo de la venta, el id de la empresa encargada del envío, todos los id de los productos para el detalle de ventas en formato de texto separados por una coma y en el último parámetro donde pasamos la cantidad de cada producto también en el orden que pasamos los id\_productos en forma de texto separado por coma. Lo primero que hace es insertar en la tabla pedidos los tres primeros parámetro, luego obtiene el id del pedido realizado y finalmente a través de un bucle while inserta el id del pedido, el id del producto y la cantidad de ese producto en la tabla detalles\_pedido.

```

317 DELIMITER $$
318 CREATE PROCEDURE `nuevo_pedido_mas_detalle`(
319     IN id_cliente_env INT,
320     IN id_empleado_env INT,
321     IN id_empresa_env INT,
322     IN productos_ids TEXT,
323     IN cantidades TEXT)
324 BEGIN
325     DECLARE pedido_id INT;
326     DECLARE i INT DEFAULT 1;
327     DECLARE producto_id INT;
328     DECLARE cantidad_producto INT;
329
330     INSERT INTO pedidos(id_cliente,id_empleado,id_empresa_envio)
331     VALUES(id_cliente_env,id_empleado_env,id_empresa_env);
332
333     SELECT MAX(id_pedido) INTO pedido_id
334     FROM pedidos;
335
336     WHILE i <= LENGTH(REPLACE(productos_ids','')) DO
337         -- Esto da como resultado la cantidad de id_productos pasados como parametro sin contar la coma.
338         SET producto_id = CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(productos_ids','i'),',',-1) AS SIGNED);
339         SET cantidad_producto = CAST(SUBSTRING_INDEX((SUBSTRING_INDEX(cantidades','i')),',',-1) AS SIGNED);
340         -- En ambos casos obtengo el numero de la posicion i del texto donde los numeros estan
341         -- separados por coma y lo convierto a un INT.
342
343         INSERT INTO detalles_pedido(id_producto,id_pedido,cantidad)
344         VALUES(producto_id,pedido_id,cantidad_producto);
345
346         SET i = i + 1;
347     END WHILE;
348 END$$

```

Por ejemplo realizamos la siguiente acción

```

443 CALL nuevo_pedido_mas_detalle(2,1,1,'1,2,3,4,5','100,24,54,78,100');
444 -- Segundo Stored procedures que agrega un venta mas el detalle de ella.

```

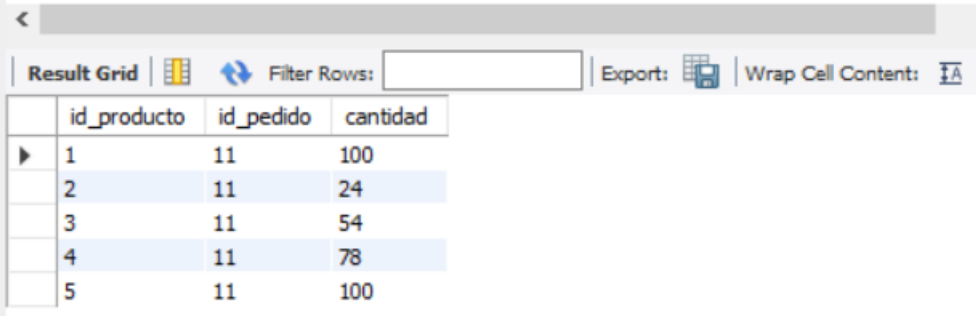
Esto lo que hace es insertar en la tabla pedidos en id\_cliente 2, el id\_empleado 1 y el id\_empresa\_envio 1. Luego al obtener el id de la venta inserta en la tabla detalles\_pedido el producto 1 con la cantidad 100, el producto 2 con la cantidad 24, el producto 3 con la cantidad 54 y así sucesivamente como podemos ver a continuación.

1 `select * from pedidos where id_pedido = 11;`

	id_pedido	id_cliente	id_empleado	id_empresa_envio	fecha
▶	11	2	1	1	2023-09-22
*	NULL	NULL	NULL	NULL	NULL

Como se puede ver se agregó con éxito a la tabla pedidos, ahora veremos como se agrego los datos a la tabla detalles\_pedido.

```
1 select * from detalles_pedido where id_pedido = 11;
```



The screenshot shows a database interface with a query result grid. The query executed is 'select \* from detalles\_pedido where id\_pedido = 11;'. The result grid displays five rows of data. The columns are 'id\_producto', 'id\_pedido', and 'cantidad'. The first row has values 1, 11, and 100. The second row has values 2, 11, and 24. The third row has values 3, 11, and 54. The fourth row has values 4, 11, and 78. The fifth row has values 5, 11, and 100. The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, and buttons for 'Export' and 'Wrap Cell Content'.

	id_producto	id_pedido	cantidad
▶	1	11	100
	2	11	24
	3	11	54
	4	11	78
	5	11	100

## 2.4 Triggers

- Trigger 1 trigger\_registrar\_pedido

Luego de que se inserta un nuevo pedido, la función del trigger es agregar a la tabla \_registro\_pedido quien insertó el pedido, que día, a qué hora y el id del pedido.

```
362 DELIMITER $$
363 • CREATE TRIGGER trigger_registrar_pedido
364 AFTER INSERT ON pedidos
365 FOR EACH ROW
366 BEGIN
367     INSERT INTO _registro_pedido(id_pedido,usuario,fecha,hora)
368     VALUES (NEW.id_pedido,USER(),CURDATE(),CURTIME());
369 END $$
```



- Trigger 2 trigger\_revizar\_cantidad\_producto

Antes de insertar detalles de una venta, corrobora que la cantidad de un producto no sea negativo, cero o nula, en caso de que suceda le asigna un valor de 50 a la cantidad.

```
374 DELIMITER $$
375 • CREATE TRIGGER trigger_revizar_cantidad_producto
376 BEFORE INSERT ON detalles_pedido
377 FOR EACH ROW
378 BEGIN
379     IF NEW.cantidad <= 0 OR NEW.cantidad IS NULL THEN
380         SET NEW.cantidad = 50;
381     END IF;
382 END $$
```

- Trigger 3 descontar stock

El propósito del de trigger es que al detallar la cantidad de un producto vendido en la tabla detalles\_pedido se descuenta la cantidad de stock en la tabla producto.

```
386 DELIMITER $$
387 • CREATE TRIGGER descontar_stock
388 AFTER INSERT ON detalles_pedido
389 FOR EACH ROW
390 BEGIN
391     DECLARE cantidad_pedido INT;
392     DECLARE id_producto_pedido INT;
393
394     SET cantidad_pedido = NEW.cantidad;
395     SET id_producto_pedido = NEW.id_producto;
396
397     UPDATE productos
398     SET unidades_disponibles = unidades_disponibles - cantidad_pedido
399     WHERE id_producto = id_producto_pedido;
400 END $$
```

- Trigger 4 insertar\_fecha

El objetivo del trigger es que en caso de que no se ingrese una fecha a la tabla pedido se inserte automáticamente la fecha del momento de la inserción de los datos.

```
404 DELIMITER $$
405 • CREATE TRIGGER insertar_fecha
406 BEFORE INSERT ON pedidos
407 FOR EACH ROW
408 BEGIN
409     IF NEW.fecha IS NULL THEN
410         SET NEW.fecha = CURDATE();
411     END IF;
412 END $$
```

## 2.5 Link script base de dato.

[https://github.com/NicoAtencio/Proyecto\\_SQL\\_Coderhouse](https://github.com/NicoAtencio/Proyecto_SQL_Coderhouse)

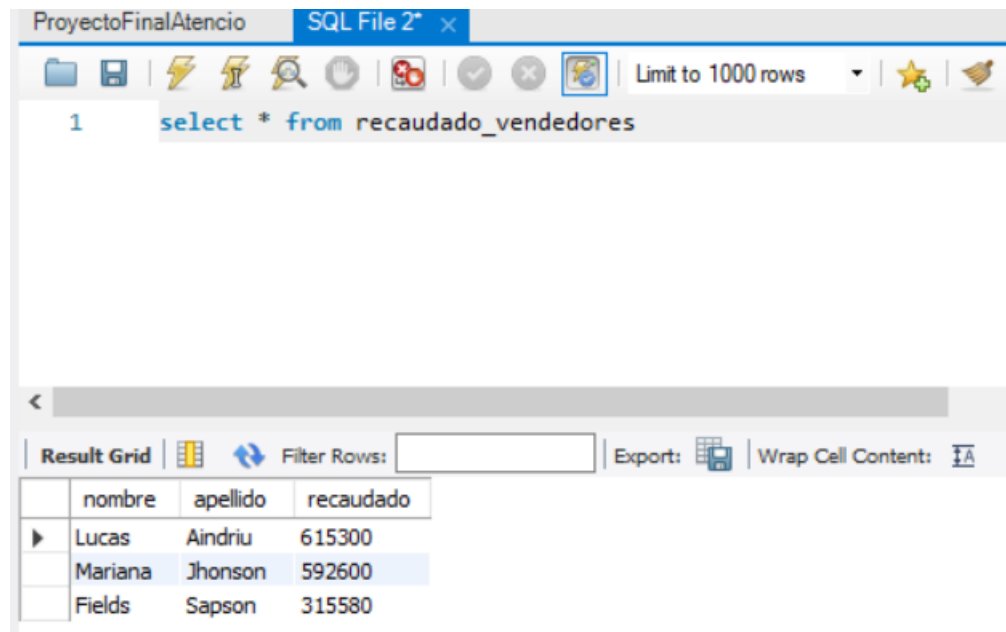
Dentro del repositorio se encuentra un archivo llamado ProyectoFinalAtencio.sql que es donde se encuentra el script con la creación de la base de dato y sus objetos, otro archivo llamado documentacionProyecto.pdf que es el archivo donde se encuentra en este momento y un archivo readme.md para que las personas interesadas es saber sobre el proyecto tengan una guía de los pasos a seguir.

## TERCER ENTREGA

### 3.1 Informes

- Informe 1

Primero se presentará la consulta y luego se redactará las conclusiones.



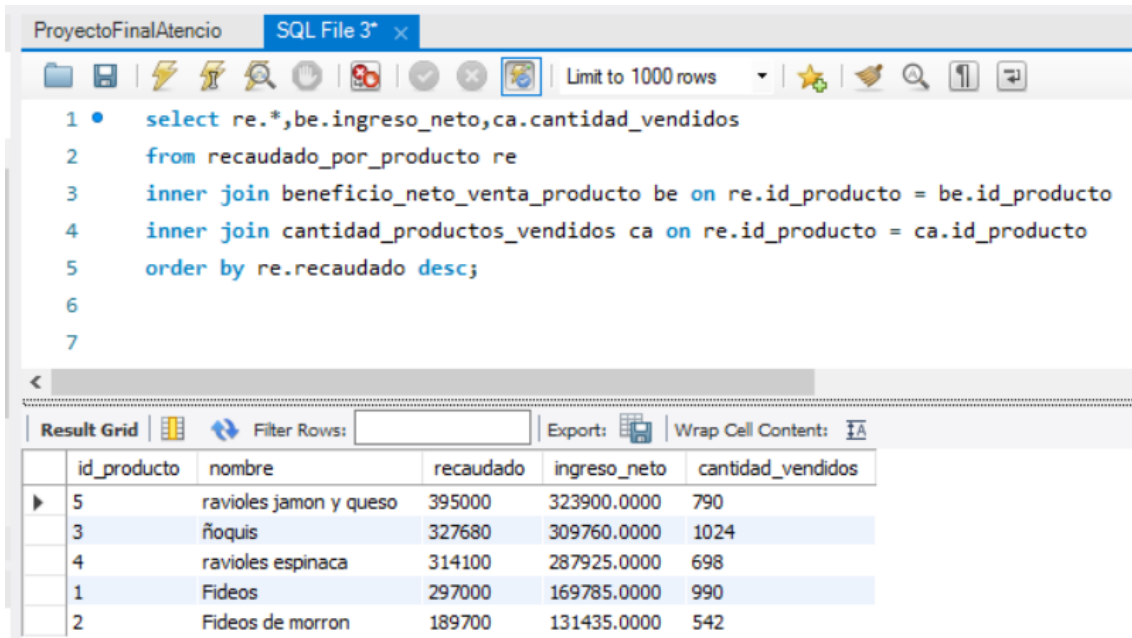
The screenshot shows a SQL IDE window titled 'ProyectoFinalAtencio' with a tab for 'SQL File 2\*'. The query editor contains the SQL statement: `1 select * from recaudado_vendedores`. Below the editor, the 'Result Grid' displays the query results in a table format. The table has three columns: 'nombre', 'apellido', and 'recaudado'. There are three rows of data: Lucas Aindriu with 615300, Mariana Jhonson with 592600, and Fields Sapson with 315580. The interface includes various icons for file operations, a 'Limit to 1000 rows' dropdown, and options for 'Filter Rows', 'Export', and 'Wrap Cell Content'.

	nombre	apellido	recaudado
▶	Lucas	Aindriu	615300
	Mariana	Jhonson	592600
	Fields	Sapson	315580

En base a la consulta realizada podemos observar la cantidad recaudada por las ventas de cada vendedor desde que comenzó la empresa. Lo que podemos notar es que Lucas Aindriu es el vendedor con mas ventas lo cual se podría ganar un premio por ello o liderar un equipo de ventas, por el contrario Fields Sapson es el vendedor que menos ha recaudado, habría que ver todas las variables posibles para dicha razón pero vamos a simular que todos los vendedores llevan el mismo tiempo en la empresa por lo cual se tendría que tomar una determinación con Fields, como por ejemplo capacitarlo para que mejore sus ventas o ubicarlo en un puesto donde sea más eficiente.

- Informe 2

Primero se presentará la consulta y luego se redactará las conclusiones.



```

1 • select re.*,be.ingreso_netto,ca.cantidad_vendidos
2   from recaudado_por_producto re
3   inner join beneficio_neto_venta_producto be on re.id_producto = be.id_producto
4   inner join cantidad_productos_vendidos ca on re.id_producto = ca.id_producto
5   order by re.recaudado desc;
6
7

```

	id_producto	nombre	recaudado	ingreso_netto	cantidad_vendidos
▶	5	ravioles jamon y queso	395000	323900.0000	790
	3	ñoquis	327680	309760.0000	1024
	4	ravioles espinaca	314100	287925.0000	698
	1	Fideos	297000	169785.0000	990
	2	Fideos de morron	189700	131435.0000	542

En la consulta recaudado\_por\_producto y beneficio\_neto\_venta\_producto son vistas.

Las conclusiones que podemos sacar con esta consulta es que el producto que más dinero recaudó con ventas también es el que más dinero neto ha hecho que ingrese a la empresa.

Otra observación es que a pesar de que los ravioles con jamón y queso sea el producto con más recaudación e ingreso neto de dinero a la empresa el costo de fabricarlo es del 18% de su recaudación, mientras que en los ñoquis el costo de fabricarlo es del 5,5% de su recaudación y el segundo producto que más dinero hizo ingresar a la empresa, por lo que se debería tratar de bajar el costo de fabricar ravioles de jamón y queso y buscar la forma de aumentar las ventas de los ñoquis. También están los ravioles de espinaca que para fabricarlos se utiliza el 9% su recaudación, los fideos que se utiliza el 43% de su recaudación por lo que lo hace el producto mas caro de producir en base al beneficio obtenido y por último tenemos los fideos de morrón que para fabricarlos se debe utilizar el 30% de su recaudación.

Hay que buscar la forma de disminuir el costo de fabricar los fideos y los fideos de morrón ya que son los que menos dinero recaudan y los productos más costosos de realizar, o eliminarlos y reemplazarlos por productos que obtengan mejores ventas, mas dinero recaudado y menor costo de fabricación.

Por último, podemos observar que mas productos vendidos no significa mayores ganancias, ya que los fideos son el segundo producto más vendido pero el cuarto que obtiene mayores ganancias netas.

### 3.2 Herramientas utilizadas

Para la realización de proyecto final de SQL de Coderhouse las herramientas utilizadas fueron:

- Workbench: Se utilizo para la creación de la base de datos.
- Word: Se utilizo para la creación del informe.
- Draw.io: Se utilizo para la creación del diagrama entidad-relación.

