

(P) Diseño y programación de una base de datos

PRODUCTO 3.

**CONSULTAS AVANZADAS, VISTAS,
ACTUALIZACIONES, ELIMINACIÓN**

COMPONENTES DEL GRUPO C2047 LAC

M^aCelia Garcia Molina

Liliana Díaz Ibáñez

Agnès Garcia Mateo

CONSULTORA

Rita de la Torre Chirivella

Descripción	2
Objetivos	2
Realizar las siguientes consultas y anotar la sentencia SQL y su salida en un documento:	2
Mostrar el nombre de cada plato y la última fecha en la que ha sido solicitado.	2
Ordenar el listado por tipo de plato.	3
Hacer una lista del coste de cada plato versus su PVP tomando en cuenta el coste de sus ingredientes. Ordenar el listado por el beneficio aportado.	4
Mostrar un listado de los proveedores de bebidas a quienes se les haya comprado en los últimos dos meses.	5
Inventar una consulta que contenga un group by + having + agregación (count).	6
Inventar una consulta que contenga una combinación externa + ordenación (order by).	7
Realizar una consulta de UNIÓN para mostrar los platos y productos solicitados en cada comanda.	8
Guardar la consulta anterior como una vista.	9
Realizar las siguientes modificaciones en los datos insertados y anotar la sentencia SQL y su salida en un documento:	10
Generar automáticamente una nueva tabla resumen a partir de la comanda de elaborados que indique: Código del producto, Nombre, unidad, total cantidad y PVP promedio.	10
Eliminar los proveedores a los cuales nunca se les ha comprado productos.	10
Se requiere	12
Indicaciones para la entrega de la actividad	12

Descripción

Continuamos con las consultas y modificaciones en nuestra base de datos, utilizando el lenguaje SQL (DML), pero ahora veremos consultas y modificaciones en las que intervienen más de una tabla, y tendremos que realizarlas con sintaxis más complicada que en el producto 2.

Objetivos

- Ampliar los conocimientos sobre consultas y modificaciones más complejas como las subconsultas y combinación de tablas.

1. Realizar las siguientes consultas y anotar la sentencia SQL y su salida en un documento:

A. Mostrar el nombre de cada plato y la última fecha en la que ha sido solicitado.

Para hacer esta consulta, hemos seleccionado nomPlato desde la tabla PLATO y MAX(fechaCo) usando el comando MAX para obtener el más reciente de los registros agrupados por nomPlato usando el comando group by. Para combinar las tablas hemos usado inner join con la tabla COMANDO_PLATO teniendo en cuenta las llaves de cada Tabla idPlato=idPIPla combinándola con la Tabla COMANDA y las llaves idCo=idCoPla.

Código:

```
SELECT PLATO.nomPlato, COMANDA.fechaCo FROM PLATO, COMANDA,  
COMANDA_PLATO WHERE (COMANDA.idCo = COMANDA_PLATO.idCoPla) AND  
(PLATO.idPlato = COMANDA_PLATO.idPIPla) and comanda.fechaCo = (select MAX(fechaCo)  
from comanda);
```


```
1  
2 • Select nomPlato, MAX(fechaCo) from PLATO  
3   inner join COMANDA_PLATO ON idPlato=idPIPla inner join COMANDA ON idCo=idCoPla  
4   group by nomPlato
```

nomPlato	MAX(fechaCo)
Gambas al ajillo	2022-04-10
Ensalada de tomate	2021-11-15
Mouse de chocolate blanco	2022-04-12




B. Ordenar el listado por tipo de plato.

Para ordenar usaremos el **ORDER BY** en idTipoPlato.

SQL File 3* x

 Limit to 11

```
1 • select nomPlato, idTipoPlato from PLATO  
2   order by idTipoPlato;  
3  
4  
5
```

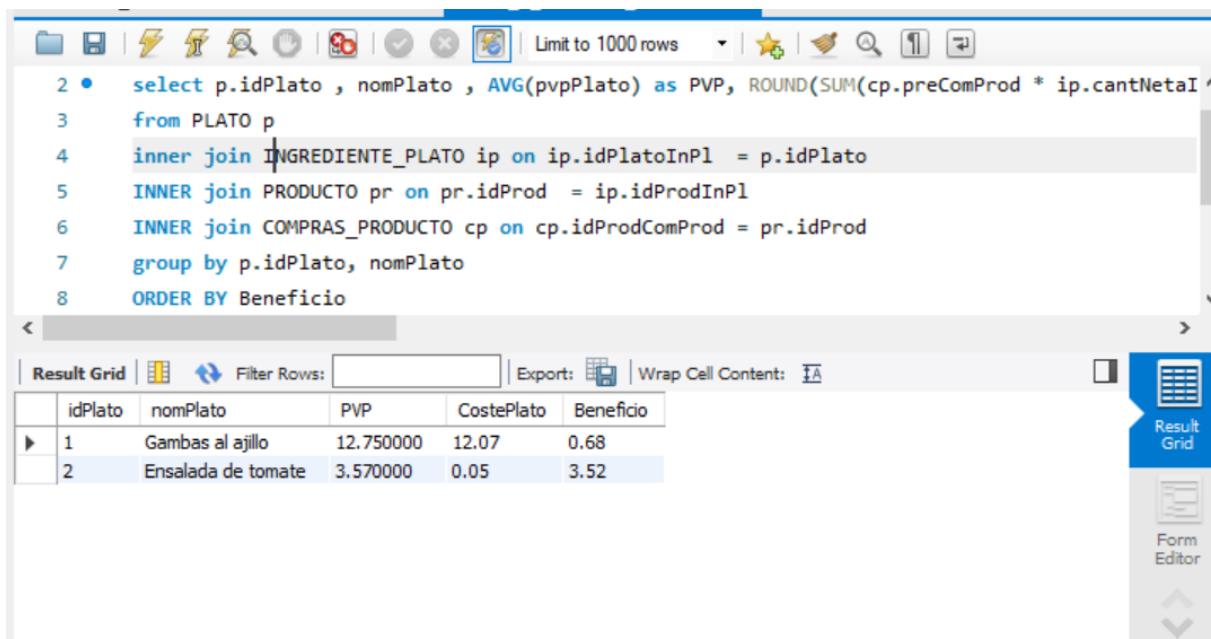
Result Grid   Filter Rows: Export: 

nomPlato	idTipoPlato
▶ Sopa castellana	1
Gambas al ajillo	2
Mouse de chocolate blanco	3
Ensalada de tomate	4

Código:

```
select nomPlato, idTipoPlato from PLATO order by idTipoPlato;
```

C. Hacer una lista del coste de cada plato versus su PVP tomado en cuenta el coste de sus ingredientes. Ordenar el listado por el beneficio aportado.



The screenshot shows a database query editor with a SQL query and its results in a grid. The query is as follows:

```
2 • select p.idPlato , nomPlato , AVG(pvpPlato) as PVP, ROUND(SUM(cp.preComProd * ip.cantNetaI  
3 from PLATO p  
4 inner join INGREDIENTE_PLATO ip on ip.idPlatoInPl = p.idPlato  
5 INNER join PRODUCTO pr on pr.idProd = ip.idProdInPl  
6 INNER join COMPRAS_PRODUCTO cp on cp.idProdComProd = pr.idProd  
7 group by p.idPlato, nomPlato  
8 ORDER BY Beneficio
```

The results are displayed in a grid with the following columns: idPlato, nomPlato, PVP, CostePlato, and Beneficio. The data is as follows:

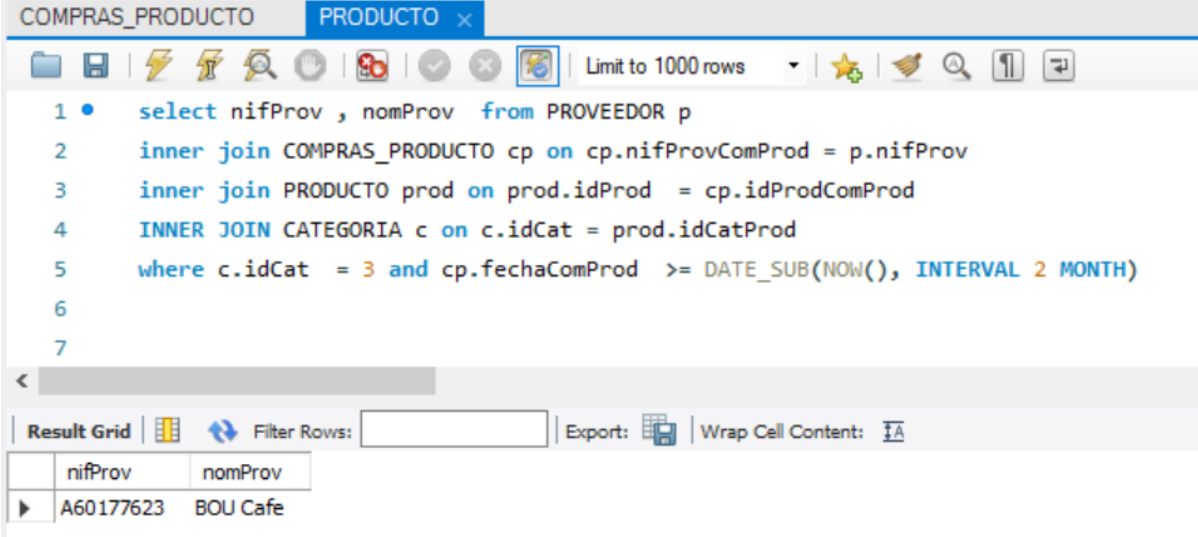
idPlato	nomPlato	PVP	CostePlato	Beneficio
1	Gambas al ajillo	12.750000	12.07	0.68
2	Ensalada de tomate	3.570000	0.05	3.52

Código:

```
select p.idPlato , nomPlato , AVG(pvpPlato) as PVP,  
ROUND(SUM(cp.preComProd * ip.cantNetaInPI / cp.cantComProd), 2) as  
CostePlato, ROUND(AVG(pvpPlato) - SUM(cp.preComProd *  
ip.cantNetaInPI / cp.cantComProd), 2) as Beneficio  
from PLATO p  
inner join INGREDIENTE_PLATO ip on ip.idPlatoInPI = p.idPlato  
INNER join PRODUCTO pr on pr.idProd = ip.idProdInPI  
INNER join COMPRAS_PRODUCTO cp on cp.idProdComProd =  
pr.idProd  
group by p.idPlato, nomPlato
```

ORDER BY Beneficio

D. Mostrar un listado de los proveedores de bebidas a quienes se les haya comprado en los últimos dos meses.



The screenshot shows a SQL query editor with a toolbar at the top containing icons for file operations, execution, and navigation. The query is as follows:

```
1 • select nifProv , nomProv from PROVEEDOR p
2 inner join COMPRAS_PRODUCTO cp on cp.nifProvComProd = p.nifProv
3 inner join PRODUCTO prod on prod.idProd = cp.idProdComProd
4 INNER JOIN CATEGORIA c on c.idCat = prod.idCatProd
5 where c.idCat = 3 and cp.fechaComProd >= DATE_SUB(NOW(), INTERVAL 2 MONTH)
6
7
```

Below the query editor, there is a 'Result Grid' section with a 'Filter Rows' input field and an 'Export' button. The results are displayed in a table:

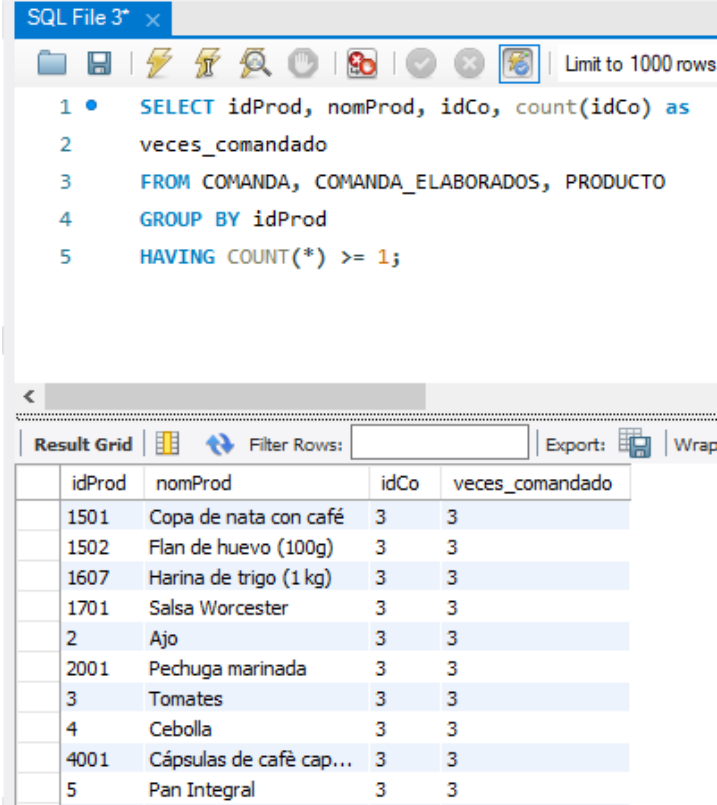
nifProv	nomProv
A60177623	BOU Cafe

Código:

```
select nifProv , nomProv from PROVEEDOR p
inner join COMPRAS_PRODUCTO cp on cp.nifProvComProd = p.nifProv
inner join PRODUCTO prod on prod.idProd = cp.idProdComProd
INNER JOIN CATEGORIA c on c.idCat = prod.idCatProd
where c.idCat = 3 and cp.fechaComProd >= DATE_SUB(NOW(),
INTERVAL 2 MONTH)
```

E. Inventar una consulta que contenga un group by + having + agregació (count).

Creemos una consulta que nos agrupara el número de veces que se ha pedido el Producto y sacamos los productos que se han pedido más de 1 vez. Para ello crearemos “Veces_Comandado”.



The screenshot shows a SQL IDE window titled "SQL File 3* x". The query editor contains the following SQL code:

```
1 • SELECT idProd, nomProd, idCo, count(idCo) as  
2 veces_comandado  
3 FROM COMANDA, COMANDA_ELABORADOS, PRODUCTO  
4 GROUP BY idProd  
5 HAVING COUNT(*) >= 1;
```

Below the query editor, the "Result Grid" tab is active, displaying the results of the query. The results are as follows:

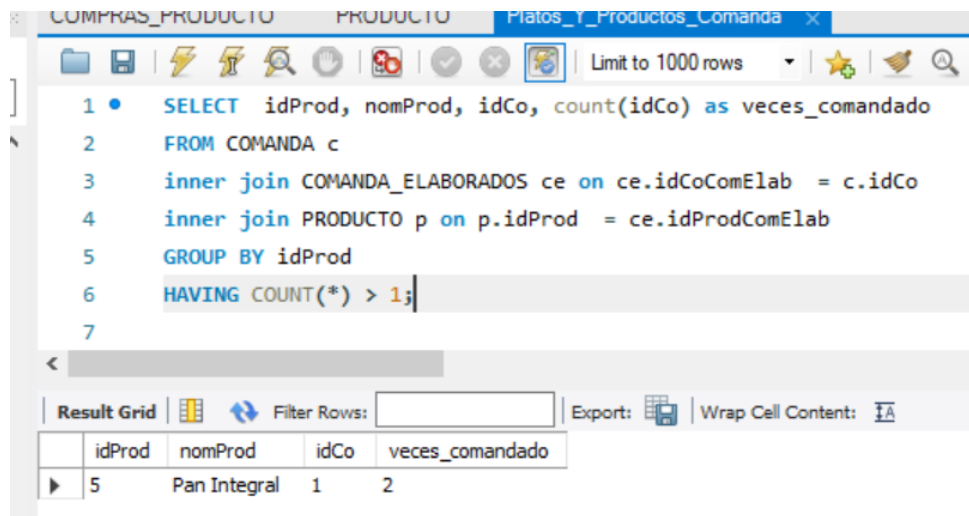
	idProd	nomProd	idCo	veces_comandado
	1501	Copa de nata con café	3	3
	1502	Flan de huevo (100g)	3	3
	1607	Harina de trigo (1 kg)	3	3
	1701	Salsa Worcester	3	3
	2	Ajo	3	3
	2001	Pechuga marinada	3	3
	3	Tomates	3	3
	4	Cebolla	3	3
	4001	Cápsulas de café cap...	3	3
	5	Pan Integral	3	3

Código:

```
SELECT idProd, nomProd, idCo, count(idCo) as veces_comandado  
FROM COMANDA, COMANDA_ELABORADOS, PRODUCTO  
GROUP BY idProd  
HAVING COUNT(*) >= 1;
```

Otra manera de hacer el código sería:

```
SELECT idProd, nomProd, idCo, count(idCo) as veces_comandado
FROM COMANDA c
inner join COMANDA_ELABORADOS ce on ce.idCoComElab = c.idCo
inner join PRODUCTO p on p.idProd = ce.idProdComElab
GROUP BY idProd
HAVING COUNT(*) > 1;
```



F. Inventar una consulta que contenga una combinación externa + ordenación (order by).

Supongamos que nos piden sacar la lista de todos los productos y si hacen parte de un ingrediente de plato queremos saber la cantidad total a usar en los platos que lo contengan, y queremos ver los productos que menos ingredientes requieren para no tener que almacenar tanto y ver cuales debemos descartar de la compra.

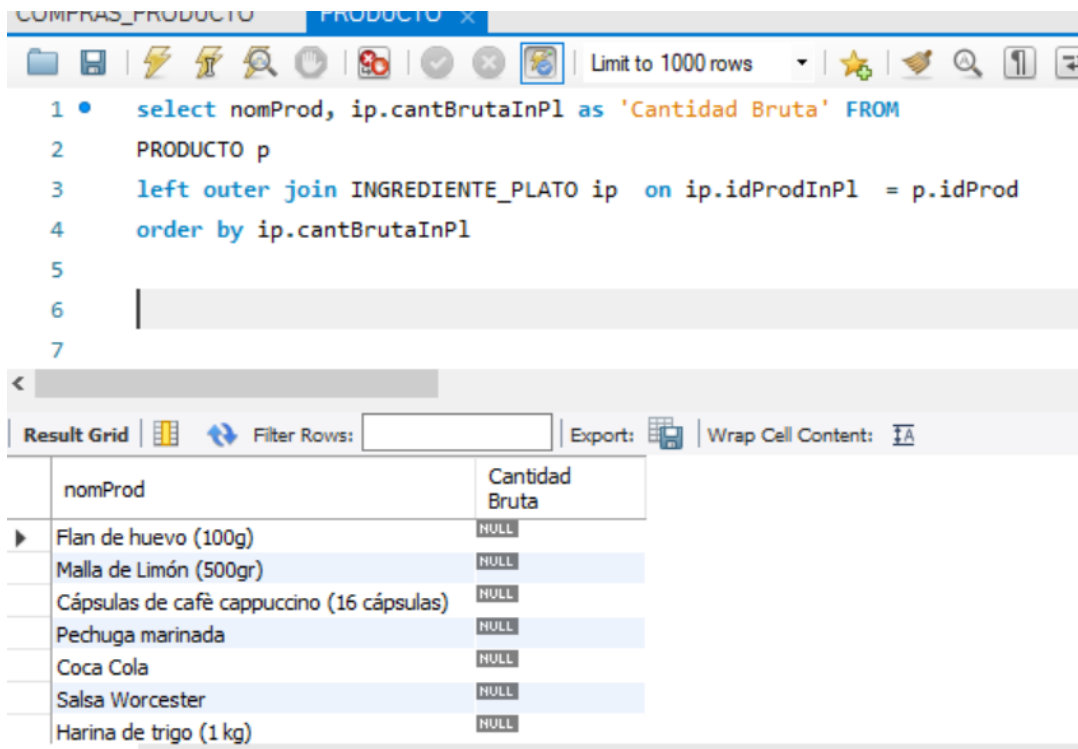
Código:

```
select idProd, nomProd, nifProv, nomProv
from PRODUCTO NATURAL LEFT OUTER JOIN PROVEEDOR
ORDER BY nomProd asc;
```

Otra opción:

```
select nomProd, ip.cantBrutaInPI as 'Cantidad Bruta' FROM
```


PRODUCTO p **left outer join** INGREDIENTE_PLATO ip **on** ip.idProdInPl =
p.idProd
order by ip.cantBrutaInPl



The screenshot shows a database query editor with a SQL query and its result grid. The query is as follows:

```

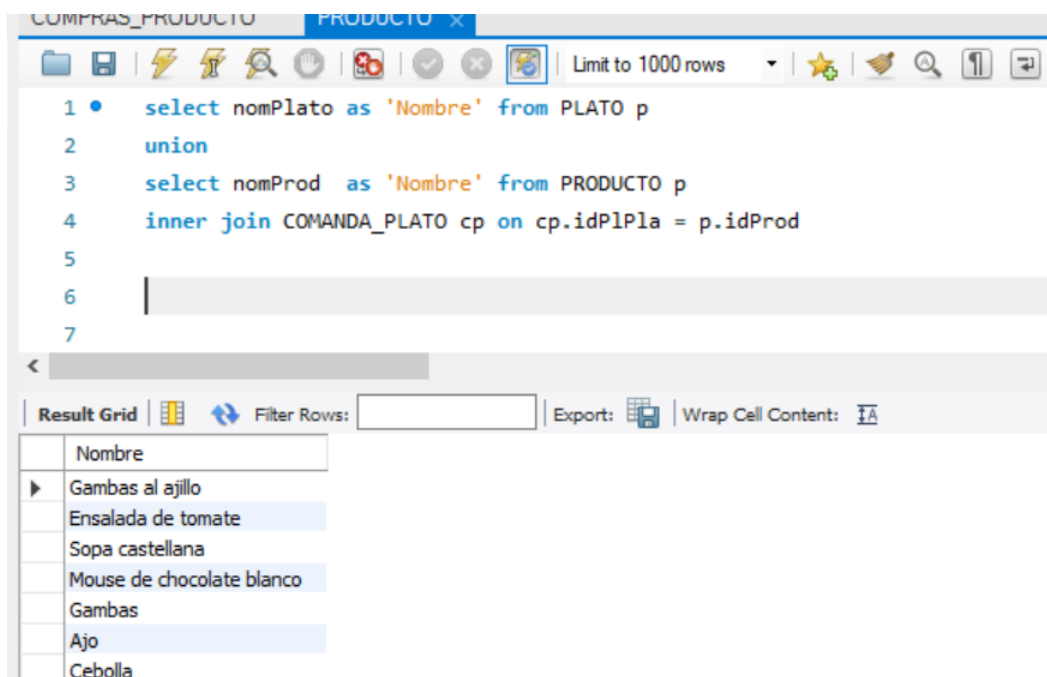
1 • select nomProd, ip.cantBrutaInPl as 'Cantidad Bruta' FROM
2 PRODUCTO p
3 left outer join INGREDIENTE_PLATO ip on ip.idProdInPl = p.idProd
4 order by ip.cantBrutaInPl
5
6
7

```

The result grid displays the following data:

nomProd	Cantidad Bruta
Flan de huevo (100g)	NULL
Malla de Limón (500gr)	NULL
Cápsulas de café cappuccino (16 cápsulas)	NULL
Pechuga marinada	NULL
Coca Cola	NULL
Salsa Worcester	NULL
Harina de trigo (1 kg)	NULL

G. Realizar una consulta de UNIÓN para mostrar los platos y productos solicitados en cada comanda.



The screenshot shows a database query editor with a SQL query and its result grid. The query is as follows:

```

1 • select nomPlato as 'Nombre' from PLATO p
2 union
3 select nomProd as 'Nombre' from PRODUCTO p
4 inner join COMANDA_PLATO cp on cp.idPlPla = p.idProd
5
6
7

```

The result grid displays the following data:

Nombre
Gambas al ajillo
Ensalada de tomate
Sopa castellana
Mouse de chocolate blanco
Gambas
Ajo
Cebolla

Código:

```
select nomPlato as 'Nombre' from PLATO p
union
select nomProd as 'Nombre' from PRODUCTO p
inner join COMANDA_PLATO cp on cp.idPIPla = p.idProd
```

H. Guardar la consulta anterior como una vista.

Código:

```
create view Platos_Y_Productos_Comanda as
select nomPlato as 'Nombre' from PLATO p
union
select nomProd as 'Nombre' from PRODUCTO p
inner join COMANDA_PLATO cp on cp.idPIPla = p.idProd
```

Ahora se puede usar la vista para hacer cualquier otra consulta:

```
select * from Platos_Y_Productos_Comanda pypc
```

The screenshot displays a database management interface with two main panels. The top panel shows the 'SCHEMAS' tree on the left, where 'ICX0_P3_7' is expanded to show 'Views'. The 'Platos_Y_Productos_Comanda' view is selected. The right panel shows the SQL editor with the following code:

```
1 • create view Platos_Y_Productos_Comanda as
2   select nomPlato as 'Nombre' from PLATO p
3   union
4   select nomProd as 'Nombre' from PRODUCTO p
5   inner join COMANDA_PLATO cp on cp.idPIPla = p.idProd
6
7
```

The bottom panel shows the same 'SCHEMAS' tree, but the 'Platos_Y_Productos_Comanda' view is now selected. The right panel shows the SQL editor with the following code:

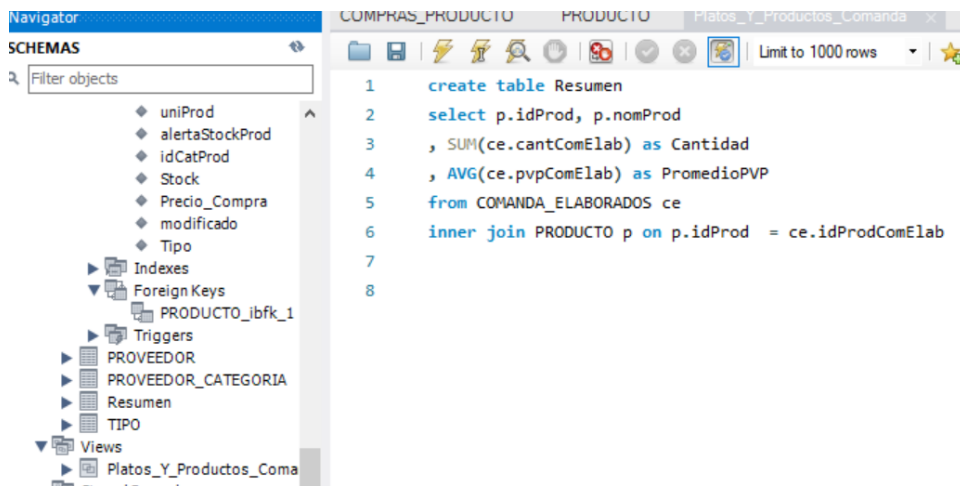
```
1 • SELECT * FROM ICX0_P3_7.Platos_Y_Productos_Comanda;
```

Below the SQL editor, the 'Result Grid' is visible, showing the output of the query. The grid has a header row 'Nombre' and several data rows:

Nombre
Gambas al ajillo
Ensalada de tomate
Sopa castellana
Mouse de chocolate blanco
Gambas
Ajo
Cebolla

2. Realizar las siguientes modificaciones en los datos insertados y anotar la sentencia SQL y su salida en un documento:

- A. Generar automáticamente una nueva tabla resumen a partir de la comanda de elaborados que indique: Código del producto, Nombre, unidad, total cantidad y PVP promedio.



Código:

```
create table Resumen
select p.idProd, p.nomProd
, SUM(ce.cantComElab) as Cantidad
, AVG(ce.pvpComElab) as PromedioPVP
from COMANDA_ELABORADOS ce
inner join PRODUCTO p on p.idProd = ce.idProdComElab
```

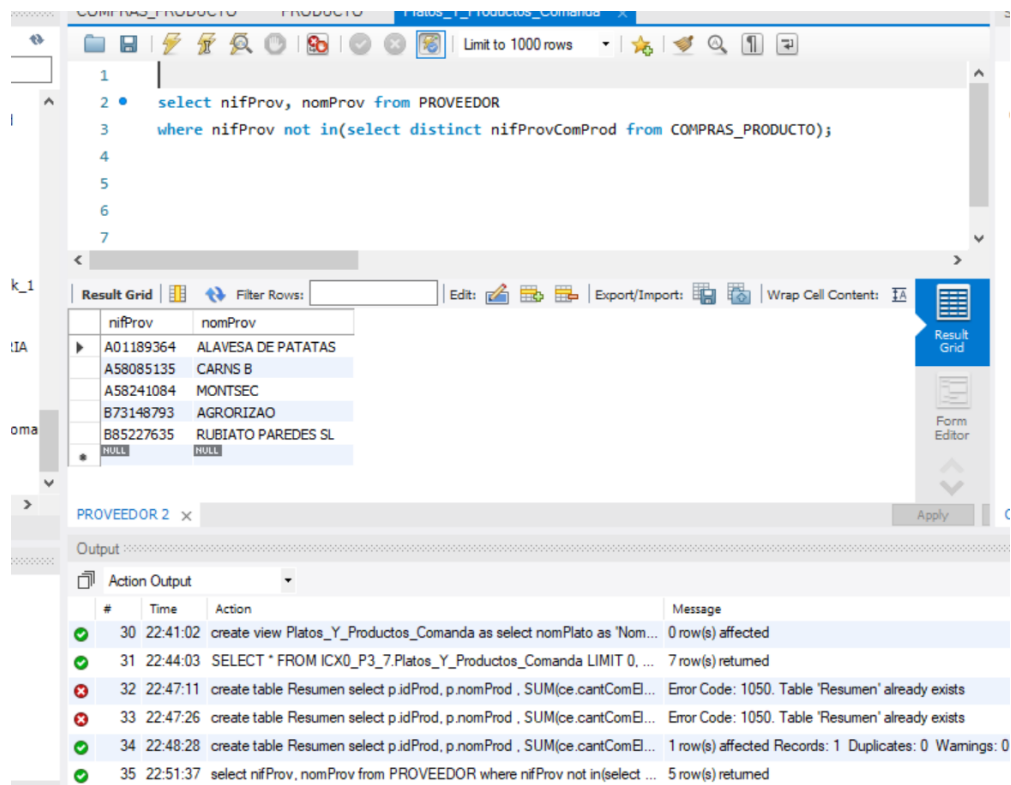
B. Eliminar los proveedores a los cuales nunca se les haya comprado productos.

Primero de todo comprobamos a cuáles de los proveedores nunca se les ha comprado producto:

Código:

```
select nifProv, nomProv from PROVEEDOR
where nifProv not in(select distinct nifProvComProd from
COMPRAS_PRODUCTO);
```

Una vez comprobado ya podemos usar el siguiente código para eliminarlos:



Código:

```
delete from PROVEEDOR
where nifProv not in (select distinct nifProvComProd from
COMPRAS_PRODUCTO);
```

Se han borrado 5 proveedores.

Automati disabled. I manually current ca toggle a

```

1
2 • delete from PROVEEDOR
3   where nifProv not in (select distinct nifProvComProd from COMPRAS_PRODUCTO);
4
5
6
7
8

```

Limit to 1000 rows

Context Help S

Output

Action Output

#	Time	Action	Message
32	22:47:11	create table Resumen select p.idProd, p.nomProd , SUM(ce.cantComEI...	Error Code: 1050. Table 'Resumen' already exists
33	22:47:26	create table Resumen select p.idProd, p.nomProd , SUM(ce.cantComEI...	Error Code: 1050. Table 'Resumen' already exists
34	22:48:28	create table Resumen select p.idProd, p.nomProd , SUM(ce.cantComEI...	1 row(s) affected Records: 1 Duplicates: 0 Warnings: 0
35	22:51:37	select nifProv, nomProv from PROVEEDOR where nifProv not in(select ...	5 row(s) returned
36	22:55:11	Código: delete from PROVEEDOR where nifProv not in (select distinct n...	Error Code: 1064. You have an error in your SQL syntax; check the man...
37	22:56:11	delete from PROVEEDOR where nifProv not in (select distinct nifProvC...	5 row(s) affected