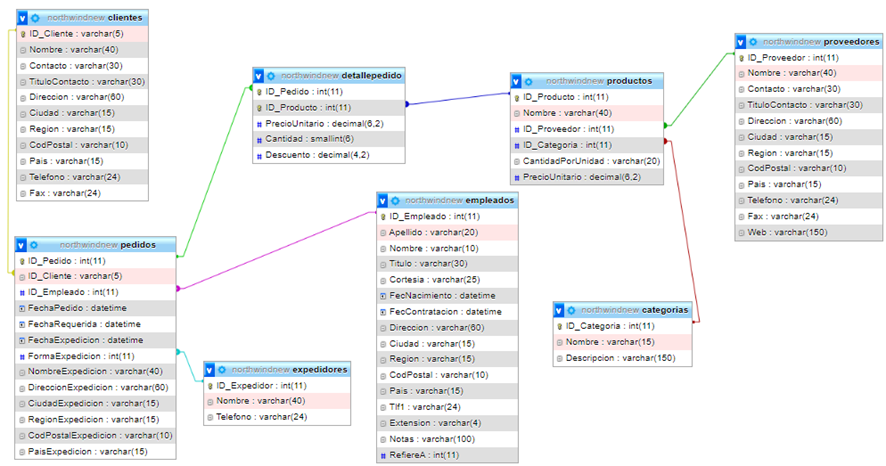
|  |  |  |
| --- | --- | --- |
| Bases de datos | | CASO PRACTICO 1 UD 5 |
|  | | |
|  | | |
|  |  | |
| alumno cesur 24/25  Alejandro Muñoz de la Sierra | PROFESOR  Inmaculada Morales Quesada | |

# CASO PRÁCTICO 1

**CONSULTAS AVANZADAS Y TRANSACCIONES**

## Contexto

En tu empresa, realizas múltiples consultas en bases de datos internas, y muchas de ellas son consultas avanzadas. Esta semana se ha incorporado a tu departamento un desarrollador nuevo, y te han pedido que lo ayudes. Para ello, y dada la siguiente estructura de base de datos:



Para crear las tablas e insertar los registros para la tarea, puedes ejecutar el script adjunto: [Recurso caso](https://cesurformacion0.sharepoint.com/:u:/s/RepoRecursosContenidos/EfqIzkhoPw1PlqGekhi4w00Bu_2kQs-e-ySD56dV3luClw?e=BsMooS)

Te piden que resuelvas los siguientes ejercicios:

1) Mostrar: código de la categoría, el nombre de la categoría, cod.Producto, nombre del producto y precio.

SELECT

c.ID\_Categoria AS Codigo\_Categoria,

c.Nombre AS Nombre\_Categoria,

p.ID\_Producto AS Codigo\_Producto,

p.Nombre AS Nombre\_Producto,

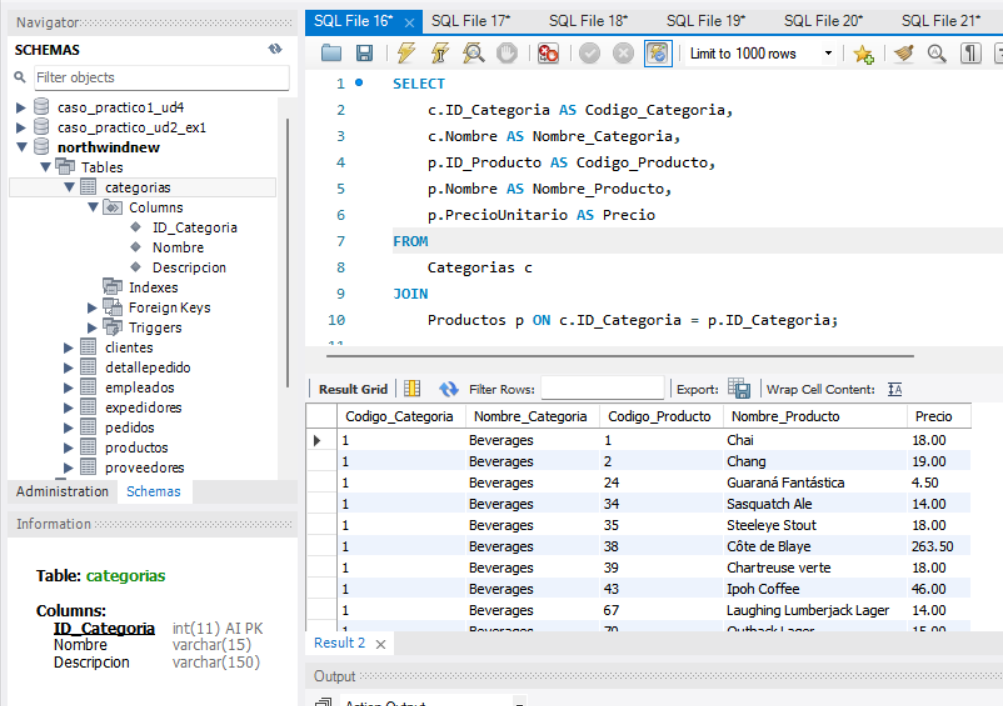
p.PrecioUnitario AS Precio

FROM

Categorias c

JOIN

Productos p ON c.ID\_Categoria = p.ID\_Categoria;



2) Mostrar: código, nombre, ciudad y país de proveedor, código, nombre y precio del producto.

SELECT

pr.ID\_Proveedor AS Codigo\_Proveedor,

pr.Nombre AS Nombre\_Proveedor,

pr.Ciudad AS Ciudad\_Proveedor,

pr.Pais AS Pais\_Proveedor,

p.ID\_Producto AS Codigo\_Producto,

p.Nombre AS Nombre\_Producto,

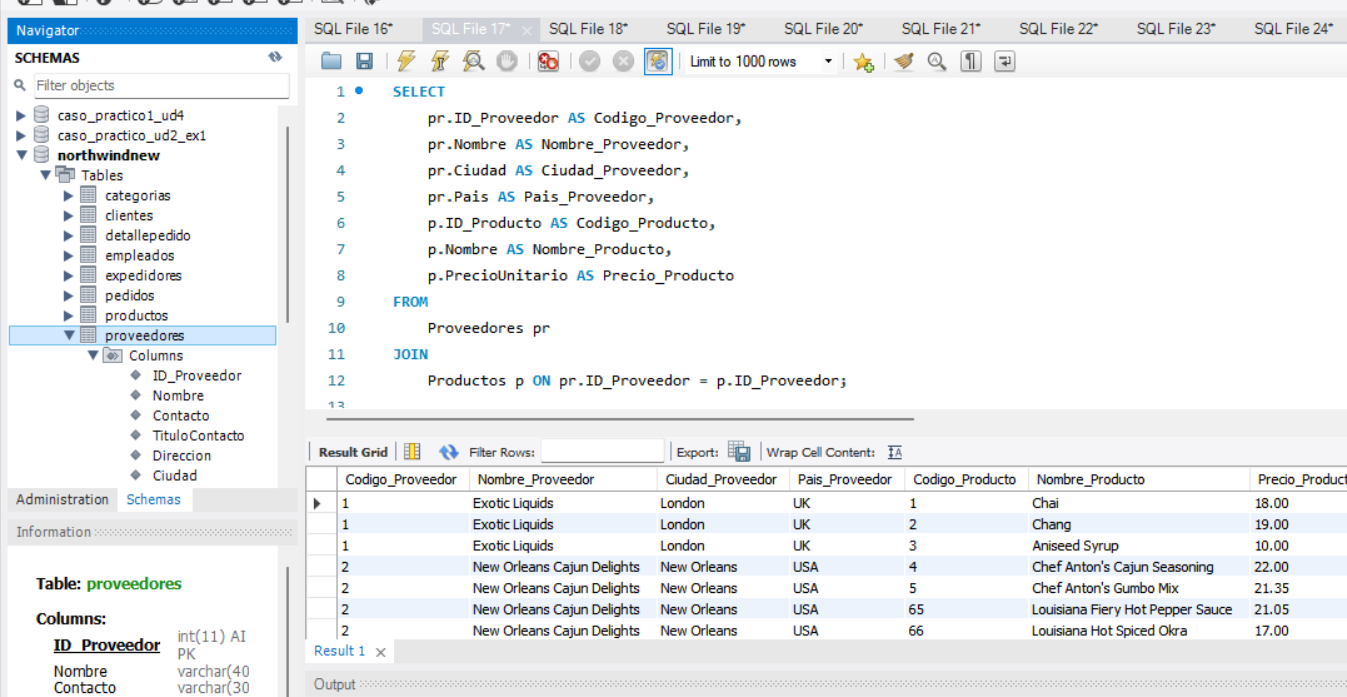
p.PrecioUnitario AS Precio\_Producto

FROM

Proveedores pr

JOIN

Productos p ON pr.ID\_Proveedor = p.ID\_Proveedor;



3) Mostrar: núm. de la orden, fecha, persona de contacto del cliente y el nombre y apellidos del empleado

SELECT

pe.ID\_Pedido AS Numero\_Orden,

pe.FechaPedido AS Fecha,

c.Contacto AS Persona\_Contacto\_Cliente,

e.Nombre AS Nombre\_Empleado,

e.Apellido AS Apellido\_Empleado

FROM

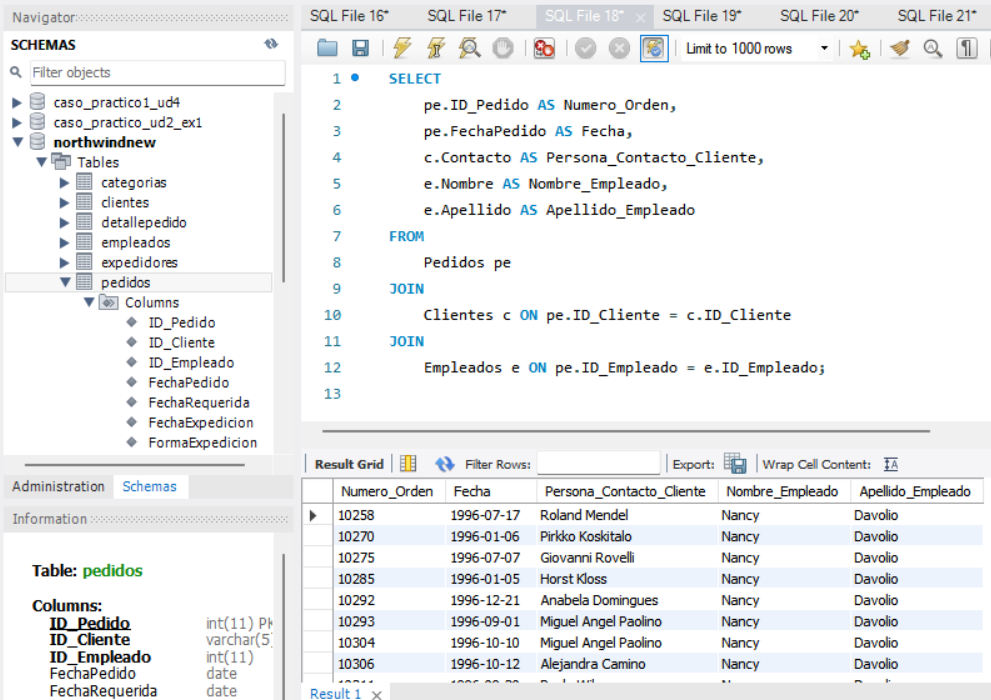
Pedidos pe

JOIN

Clientes c ON pe.ID\_Cliente = c.ID\_Cliente

JOIN

Empleados e ON pe.ID\_Empleado = e.ID\_Empleado;



4) Visualizar el mínimo precio de los productos, mostrar el nombre de la categoría y el producto

SELECT

c.Nombre AS Nombre\_Categoria,

p.Nombre AS Nombre\_Producto,

MIN(p.PrecioUnitario) AS Precio\_Minimo

FROM

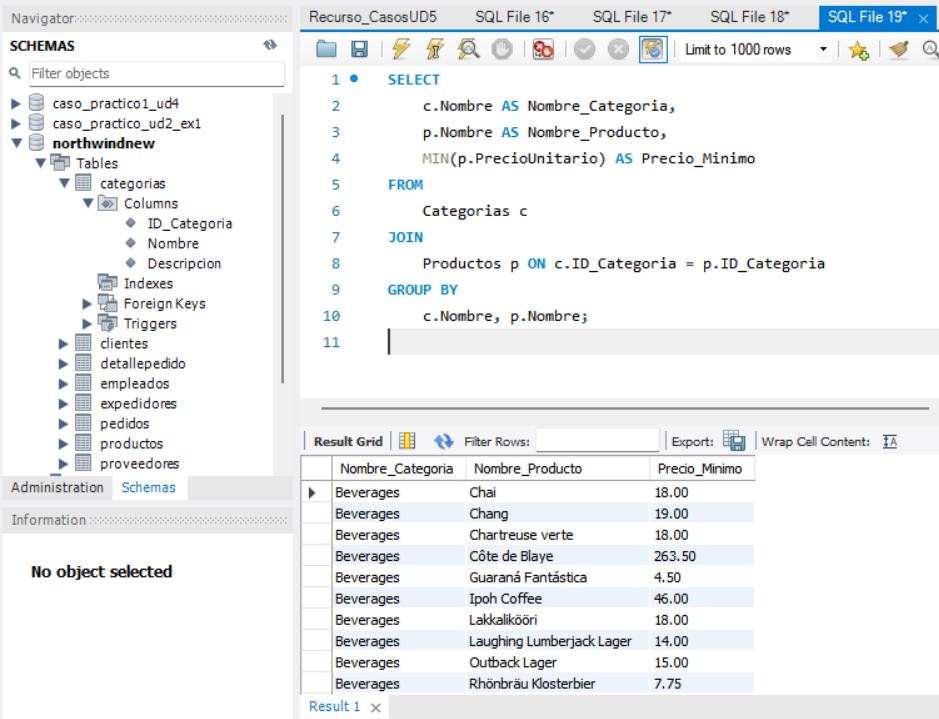
Categorias c

JOIN

Productos p ON c.ID\_Categoria = p.ID\_Categoria

GROUP BY

c.Nombre, p.Nombre;



5) Seleccionar las categorías que tengan más de 5 productos. Mostrar el nombre de la categoría y el número de productos

SELECT

c.Nombre AS Nombre\_Categoria,

COUNT(p.ID\_Producto) AS Numero\_Productos

FROM

Categorias c

JOIN

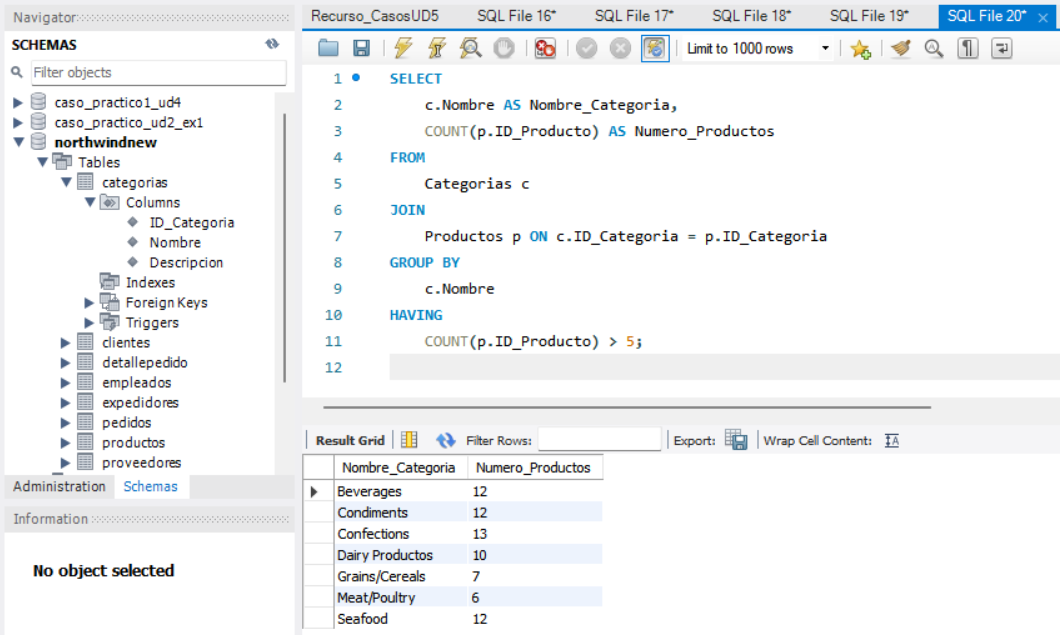
Productos p ON c.ID\_Categoria = p.ID\_Categoria

GROUP BY

c.Nombre

HAVING

COUNT(p.ID\_Producto) > 5;



6) Mostrar el nombre de todos los empleados y el número de órdenes que han hecho.

SELECT

e.Nombre AS Nombre\_Empleado,

e.Apellido AS Apellido\_Empleado,

COUNT(pe.ID\_Pedido) AS Numero\_Ordenes

FROM

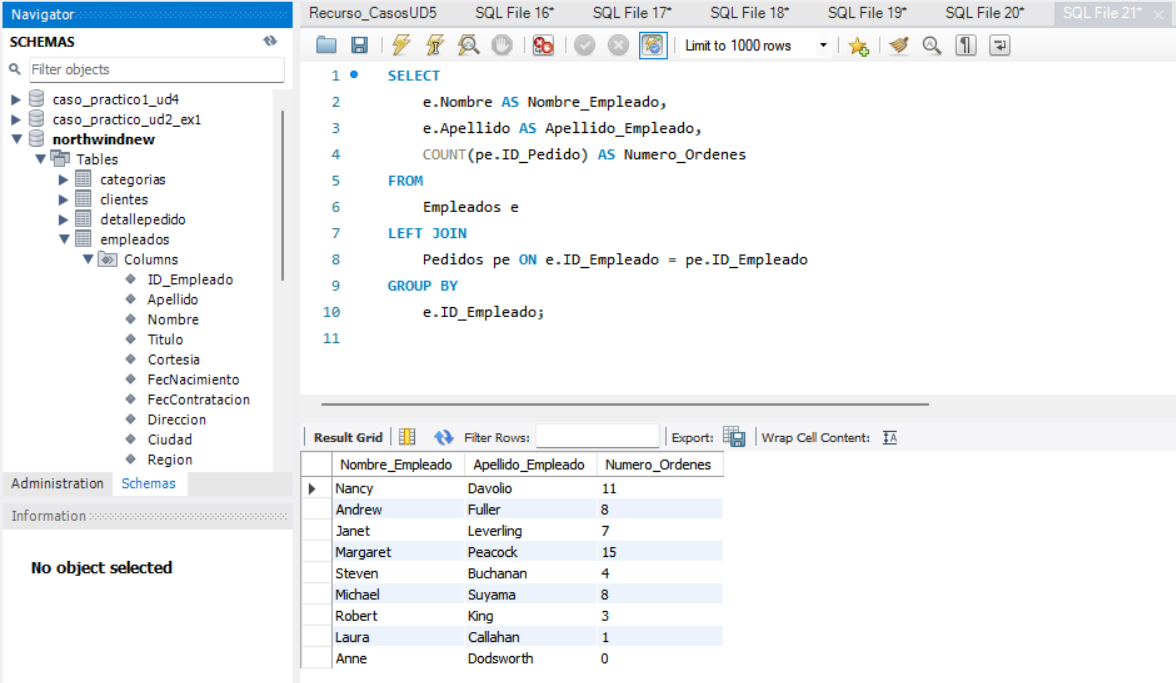
Empleados e

LEFT JOIN

Pedidos pe ON e.ID\_Empleado = pe.ID\_Empleado

GROUP BY

e.ID\_Empleado;



7) Mostrar el nombre y el precio del producto sólo cuando dicho precio sea igual o superior al precio medio de los productos.

SELECT

p.Nombre AS Nombre\_Producto,

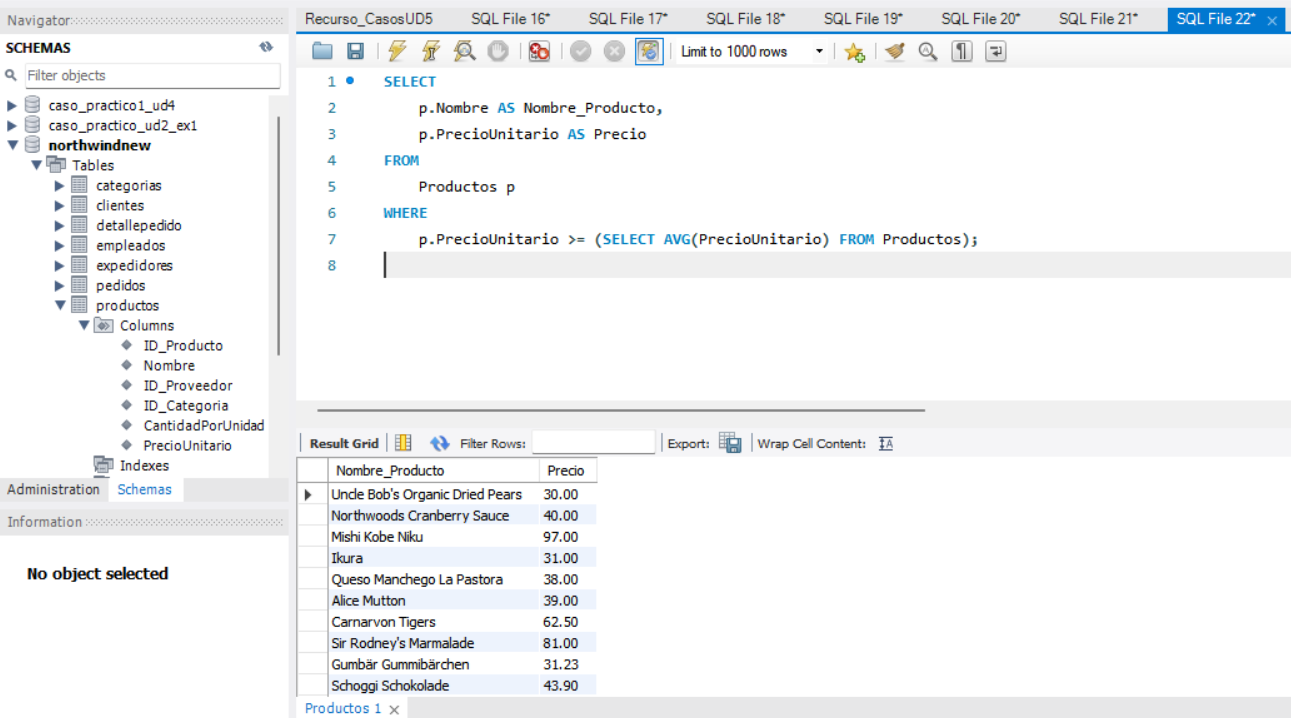
p.PrecioUnitario AS Precio

FROM

Productos p

WHERE

p.PrecioUnitario >= (SELECT AVG(PrecioUnitario) FROM Productos);



8) Mostrar el precio medio de los productos agrupados por categorías solo cuando dicho precio medio sea mayor que el precio medio de los productos de la categoría 4. Hazlo usando la cláusula HAVING

SELECT

c.Nombre AS Nombre\_Categoria,

AVG(p.PrecioUnitario) AS Precio\_Medio

FROM

Categorias c

JOIN

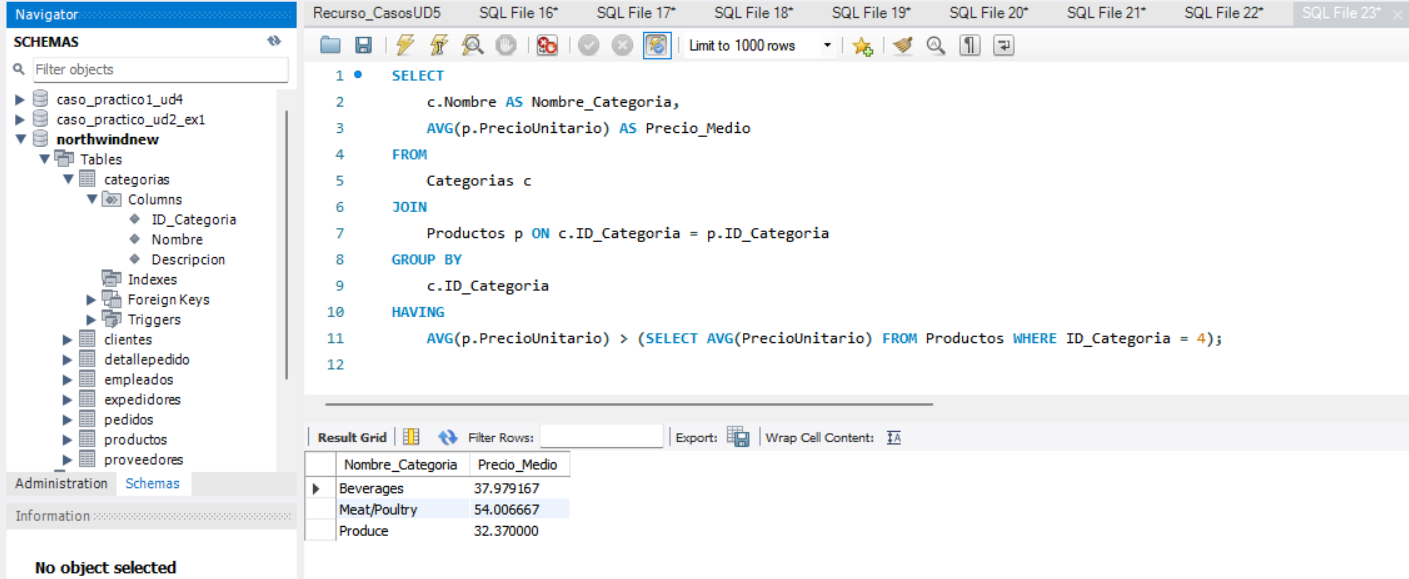
Productos p ON c.ID\_Categoria = p.ID\_Categoria

GROUP BY

c.ID\_Categoria

HAVING

AVG(p.PrecioUnitario) > (SELECT AVG(PrecioUnitario) FROM Productos WHERE ID\_Categoria = 4);



9) Obtener el Id del empleado, el nombre y el campo ID\_Pedido para los empleados cuyo id de empleado sea mayor que 5. Ten en cuenta que en el caso de que algún empleado no haya registrado ningún pedido también querremos que aparezca. Para estos casos, haz que en el campo ORDER\_ID aparezca un 0 en lugar de NULL.

SELECT

e.ID\_Empleado,

e.Nombre,

COALESCE(pe.ID\_Pedido, 0) AS ID\_Pedido

FROM

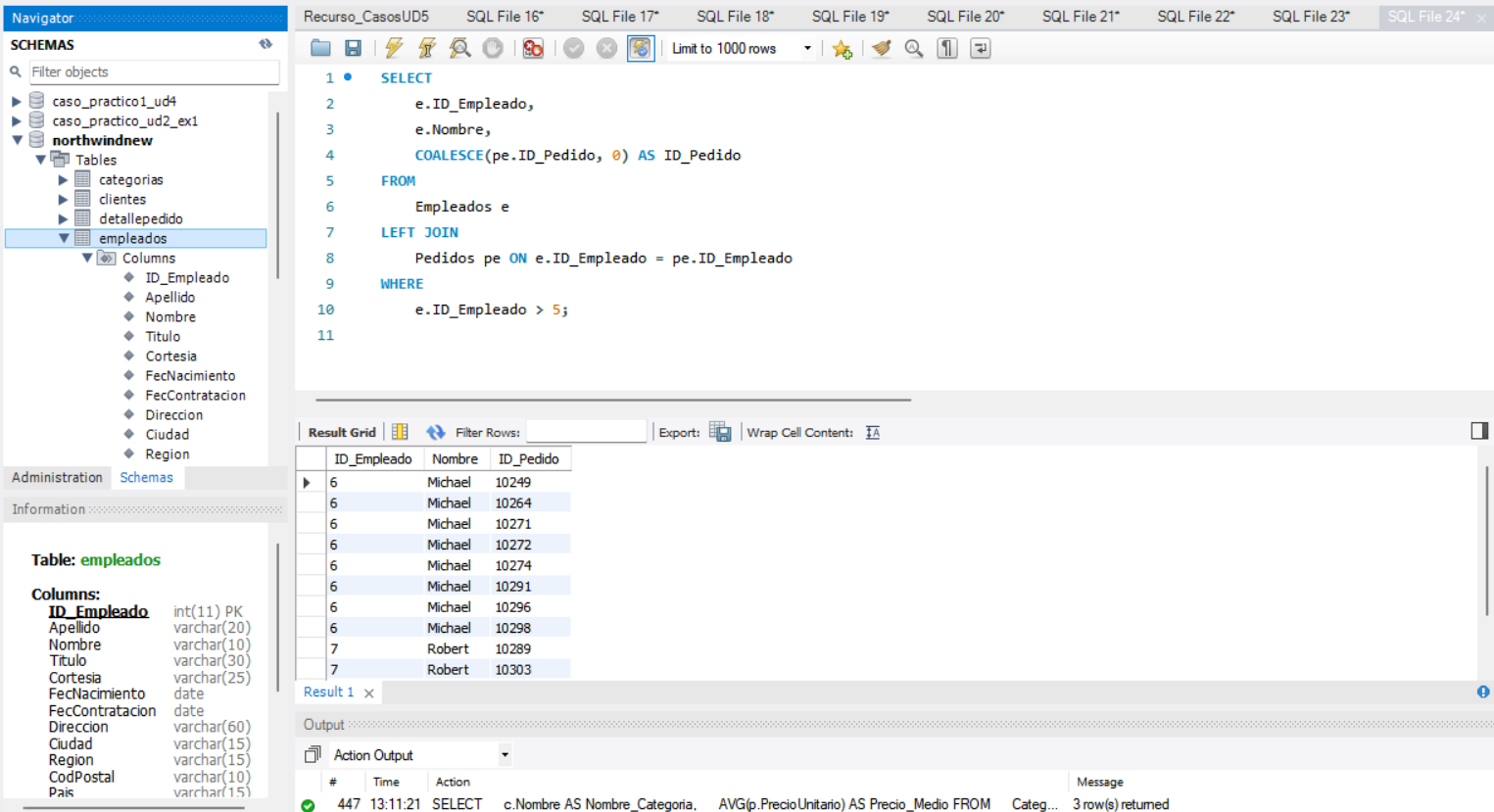
Empleados e

LEFT JOIN

Pedidos pe ON e.ID\_Empleado = pe.ID\_Empleado

WHERE

e.ID\_Empleado > 5;



10) Haz lo mismo que en el apartado anterior, pero para los clientes que no han realizado ningún pedido. Haz que en lugar de mostrar el campo ORDER\_ID muestre el valor “SIN PEDIDO”

SELECT

c.ID\_Cliente,

c.Nombre,

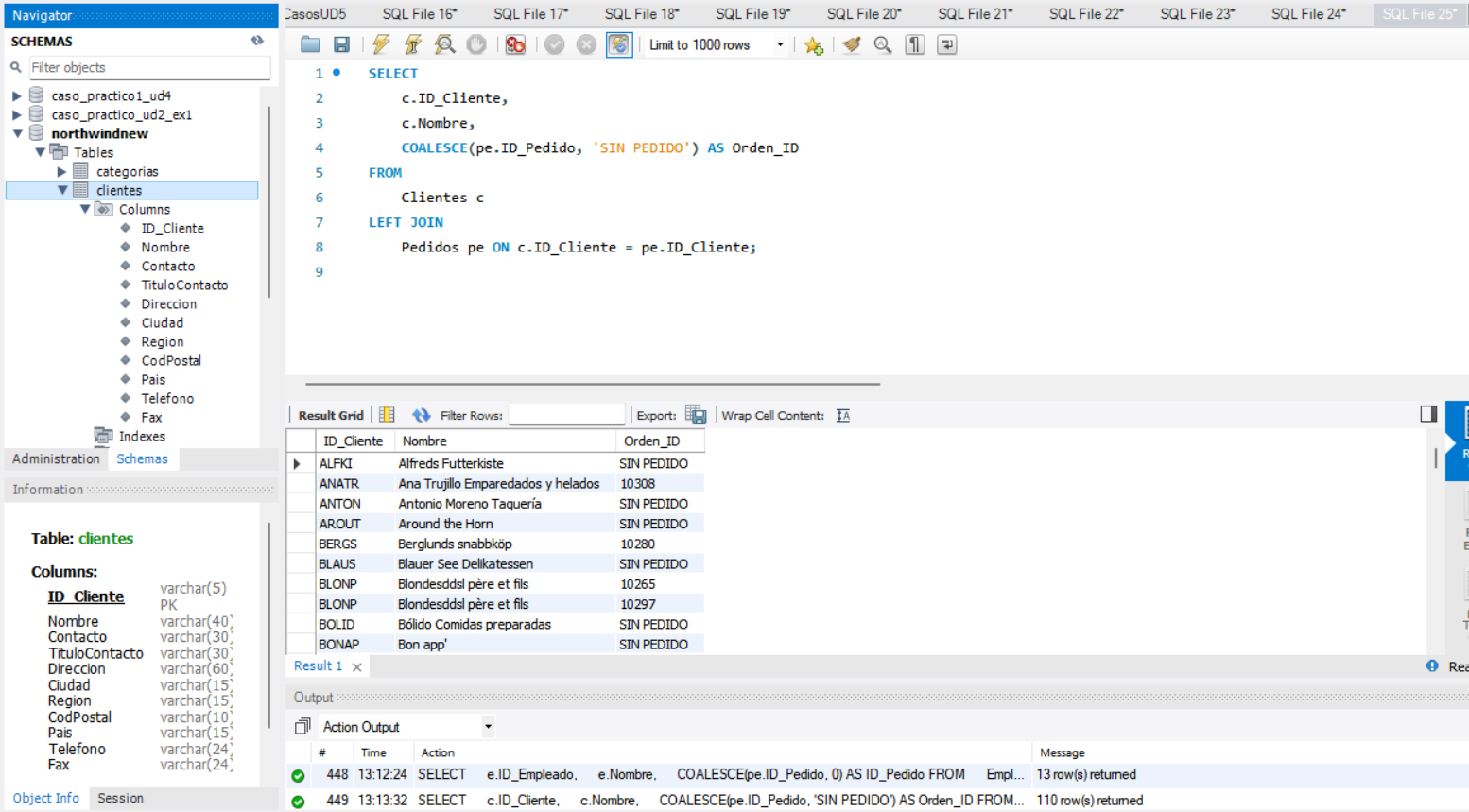
COALESCE(pe.ID\_Pedido, 'SIN PEDIDO') AS Orden\_ID

FROM

Clientes c

LEFT JOIN

Pedidos pe ON c.ID\_Cliente = pe.ID\_Cliente;



11) Crea una transacción en la que, sobre la tabla empleados, incrementes en 100 el salario del empleado con ID = 4 y reduzcas en 100 el salario del empleado con ID = 8. Por último, deshaz los cambios para finalizar la transacción.

START TRANSACTION;

UPDATE Empleados

SET Salario = Salario + 100

WHERE ID\_Empleado = 4;

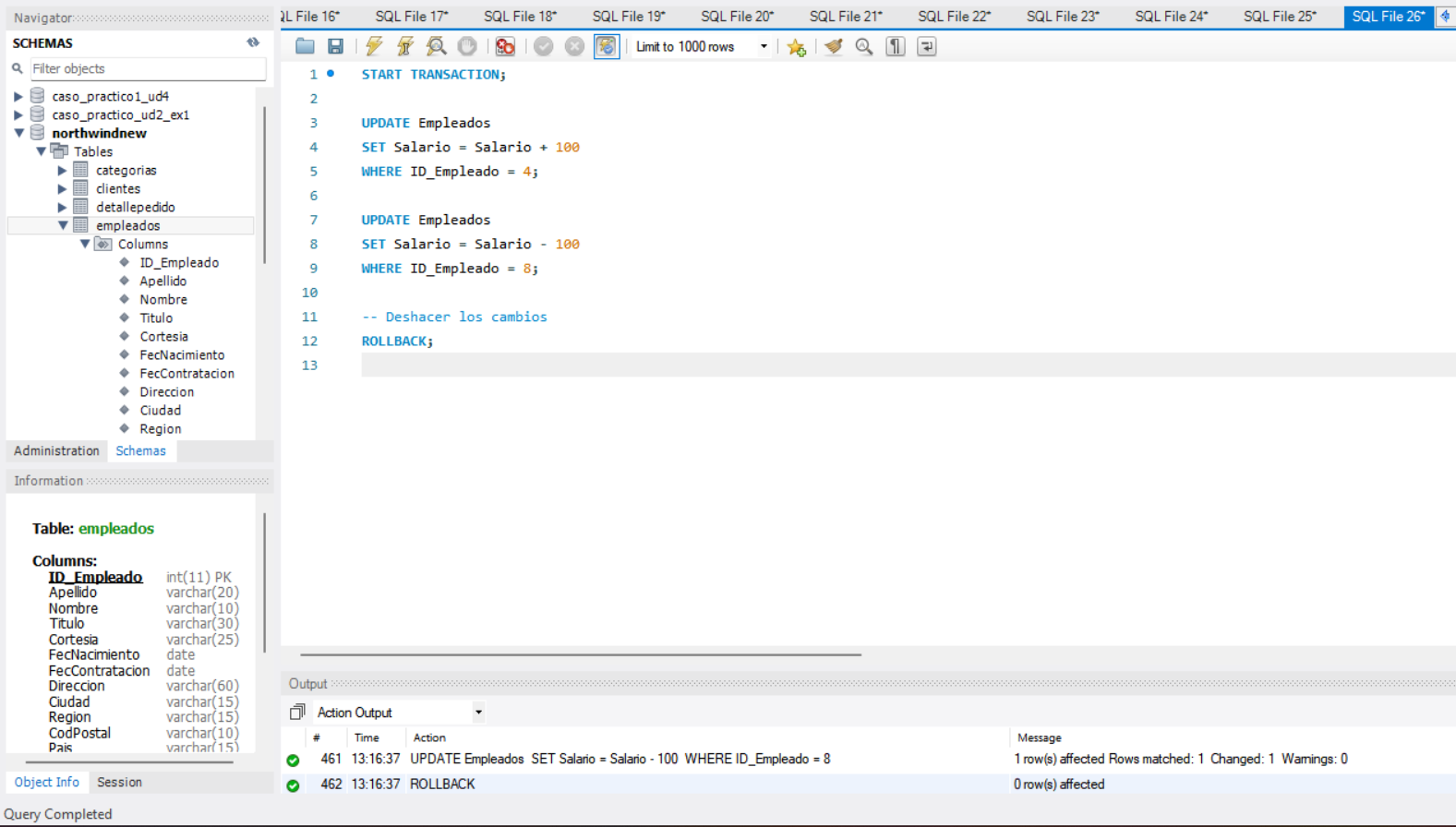
UPDATE Empleados

SET Salario = Salario - 100

WHERE ID\_Empleado = 8;

-- Deshacer los cambios

ROLLBACK;



**Conclusiones**

Este caso práctico nos ha dado una buena visión de cómo manejar bases de datos mediante varias tareas clave, como crear tablas, agregar datos, hacer consultas y controlar transacciones. A lo largo de los ejercicios, hemos visto cómo diseñar una base de datos efectiva usando claves primarias y foráneas, manejar datos con las instrucciones INSERT, UPDATE y DELETE, y hacer consultas con SELECT para obtener información.

Un punto importante ha sido la aplicación de transacciones, usando las órdenes START TRANSACTION, COMMIT y ROLLBACK. Esto nos ha ayudado a entender cómo gestionar operaciones atómicas, asegurando la integridad de los datos y que los cambios se hagan de manera ordenada y segura. También hemos revisado los resultados y aprendido a verificar y controlar los cambios en la base de datos para evitar errores.

En conclusión, este caso práctico ha sido crucial para afianzar los conceptos básicos de manejo y administración de bases de datos, dándonos una base sobre la que podemos seguir aprendiendo más sobre el manejo de datos a medida que avanzamos. Las habilidades adquiridas aquí nos ayudarán a enfrentar retos más difíciles en la administración de bases de datos.

**Referencias**

<https://dev.mysql.com/doc/>

<https://sqlzoo.net/wiki/SQL_Tutorial>

<https://www.w3schools.com/sql/default.asp?PHPSESSID=300ae3404d5fa2612f238abeebb8869c>

<https://www.youtube.com/watch?v=U8lgFaLW_Qg>

<https://www.youtube.com/watch?v=6TaZtzojx6A>

<https://www.apuntesinformaticafp.com/cursos/consultas_avanzadas.html>