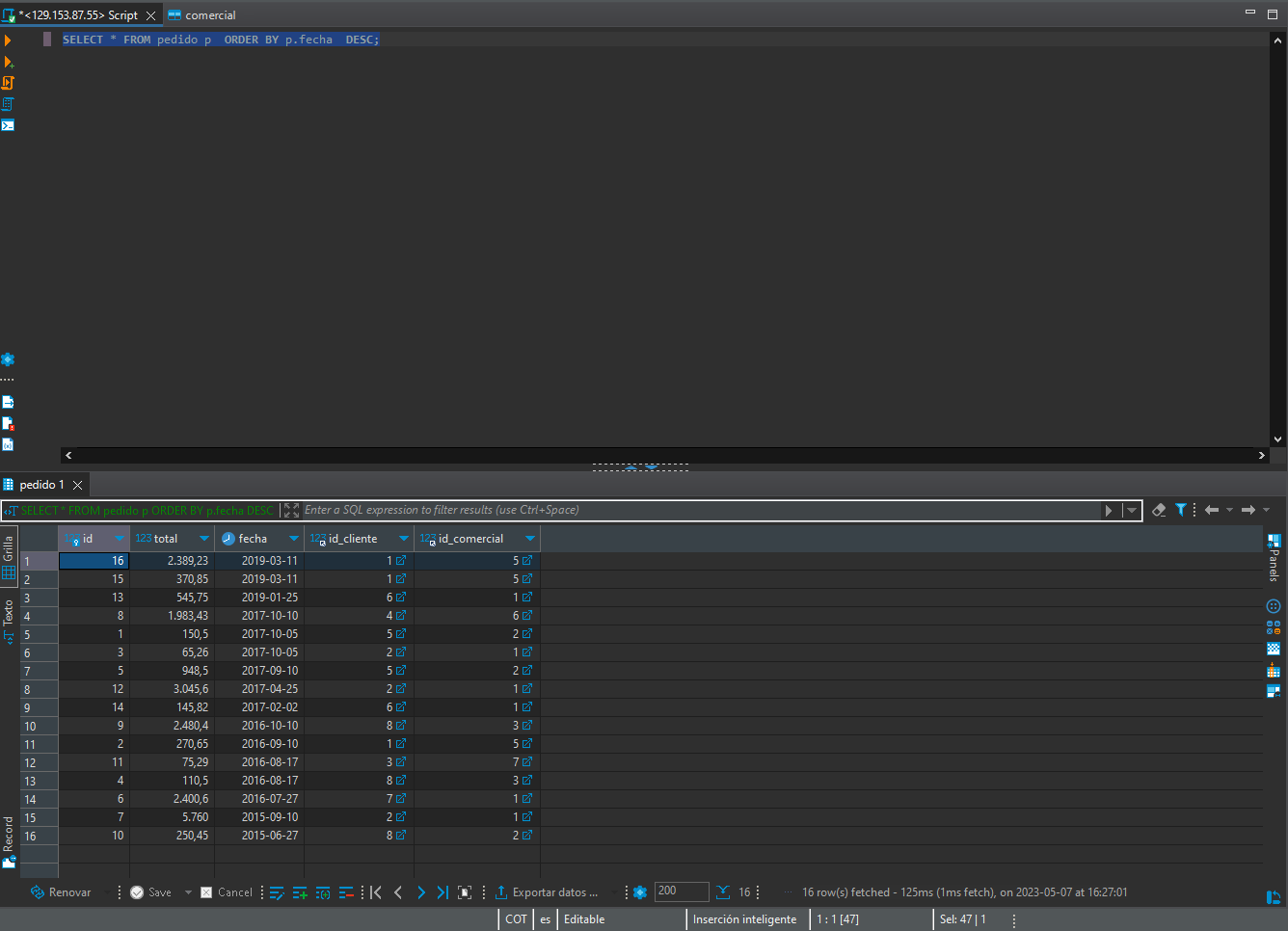
Taller

1. Gestion de Ventas

• Consultas sobre una tabla

1. Devuelve un listado con todos los pedidos que se han realizado. Los pedidos deben estar ordenados por la fecha de realización, mostrando en primer lugar los pedidos más recientes.

SELECT \* FROM pedido p ORDER BY p.fecha DESC;



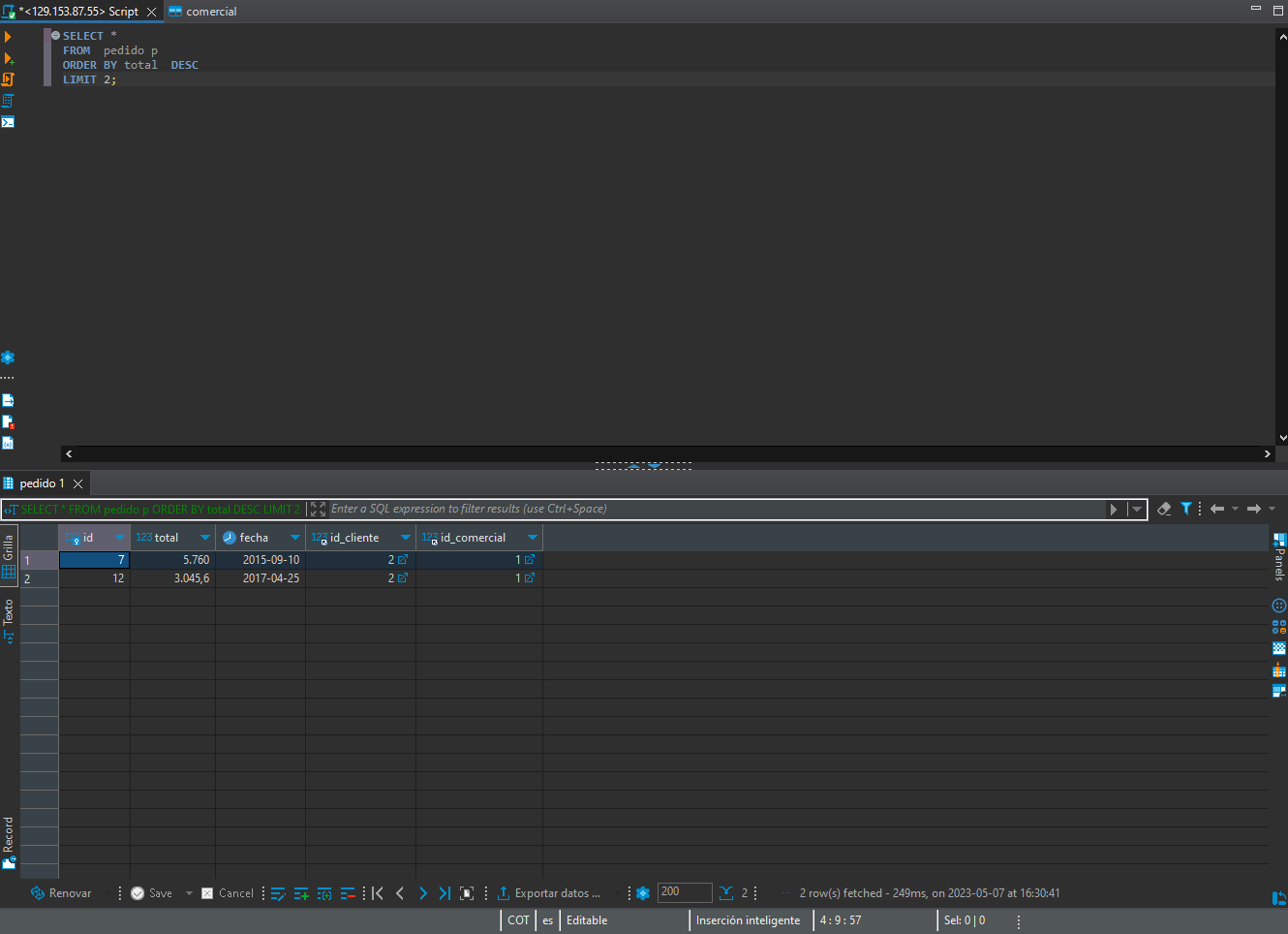
1. Devuelve todos los datos de los dos pedidos de mayor valor

SELECT \*

FROM pedido p

ORDER BY total DESC

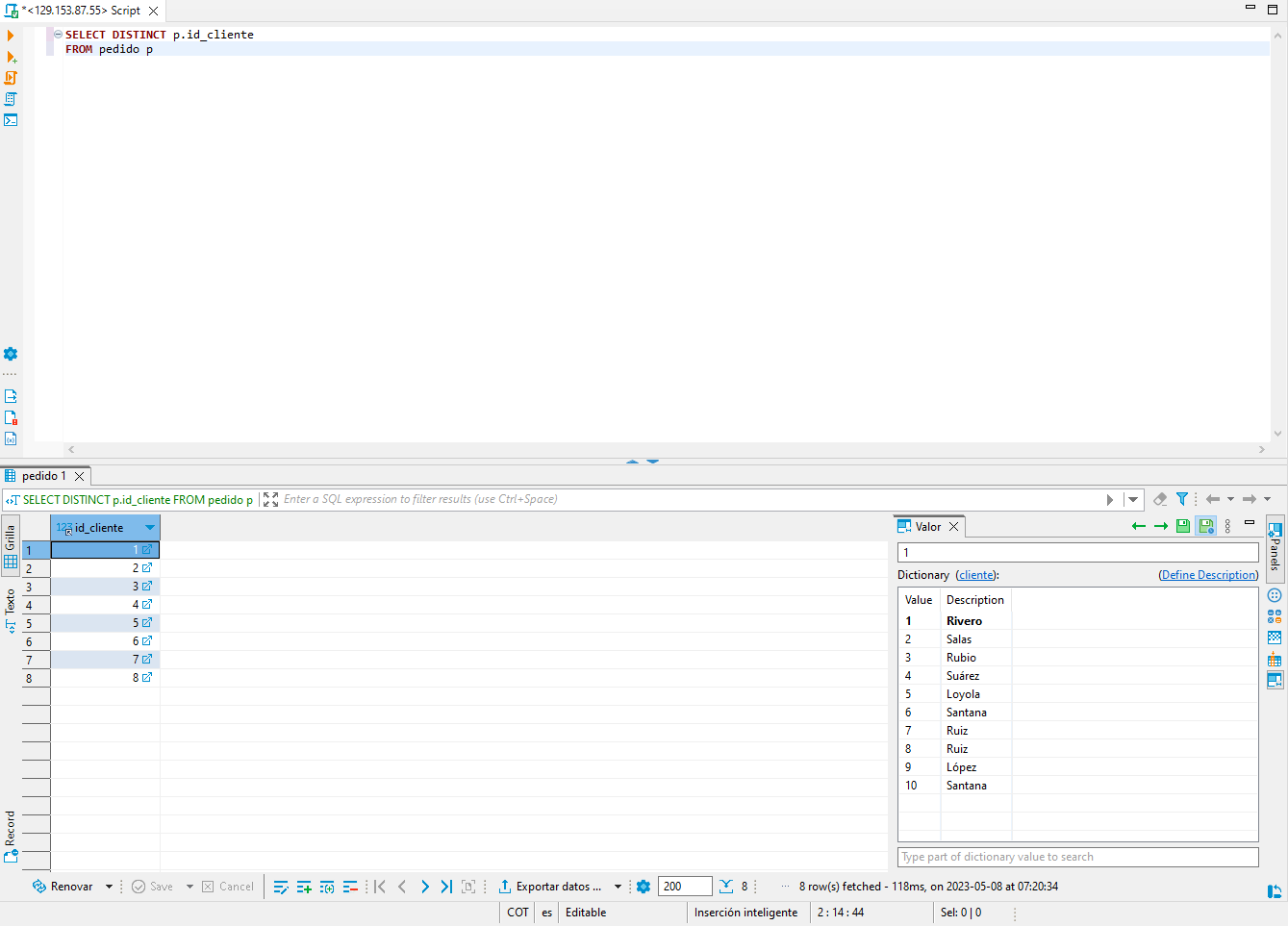
LIMIT 2;



1. Devuelve un listado con los identificadores de los clientes que han realizado algún pedido. Tenga en cuenta que no debe mostrar identificadores que estén repetidos.

SELECT DISTINCT p.id\_cliente

FROM pedido p



1. Devuelve un listado de todos los pedidos que se realizaron durante el año 2017, cuya cantidad total sea superior a 500€.

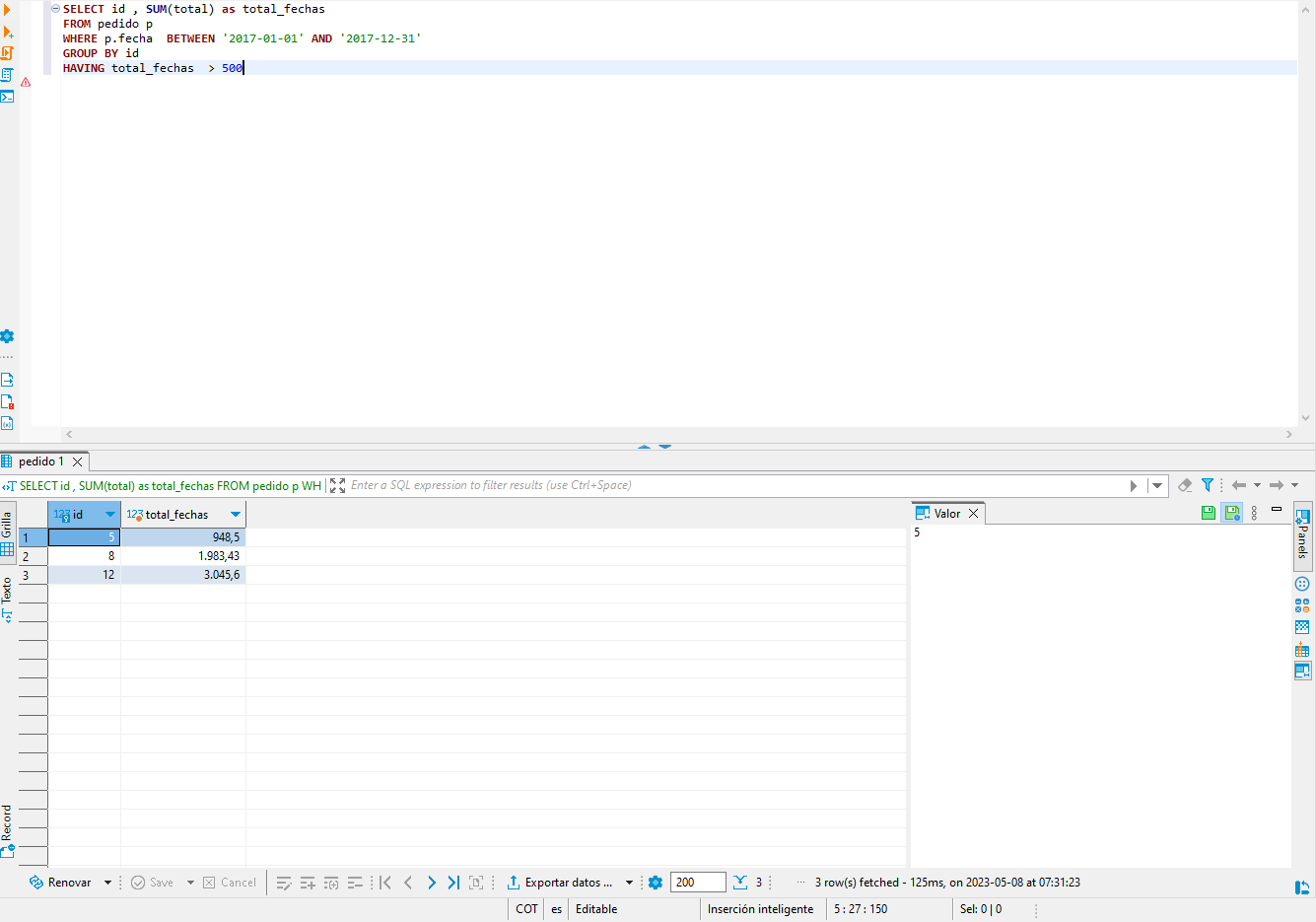
SELECT id , SUM(total) as total\_fechas

FROM pedido p

WHERE p.fecha BETWEEN '2017-01-01' AND '2017-12-31'

GROUP BY id

HAVING total\_fechas > 500



CREATE PROCEDURE proc\_sacar\_fechas(IN fecha\_inicio VARCHAR(10), IN fecha\_final VARCHAR(10))

BEGIN

SELECT id, SUM(total) AS total\_fechas

FROM pedido p

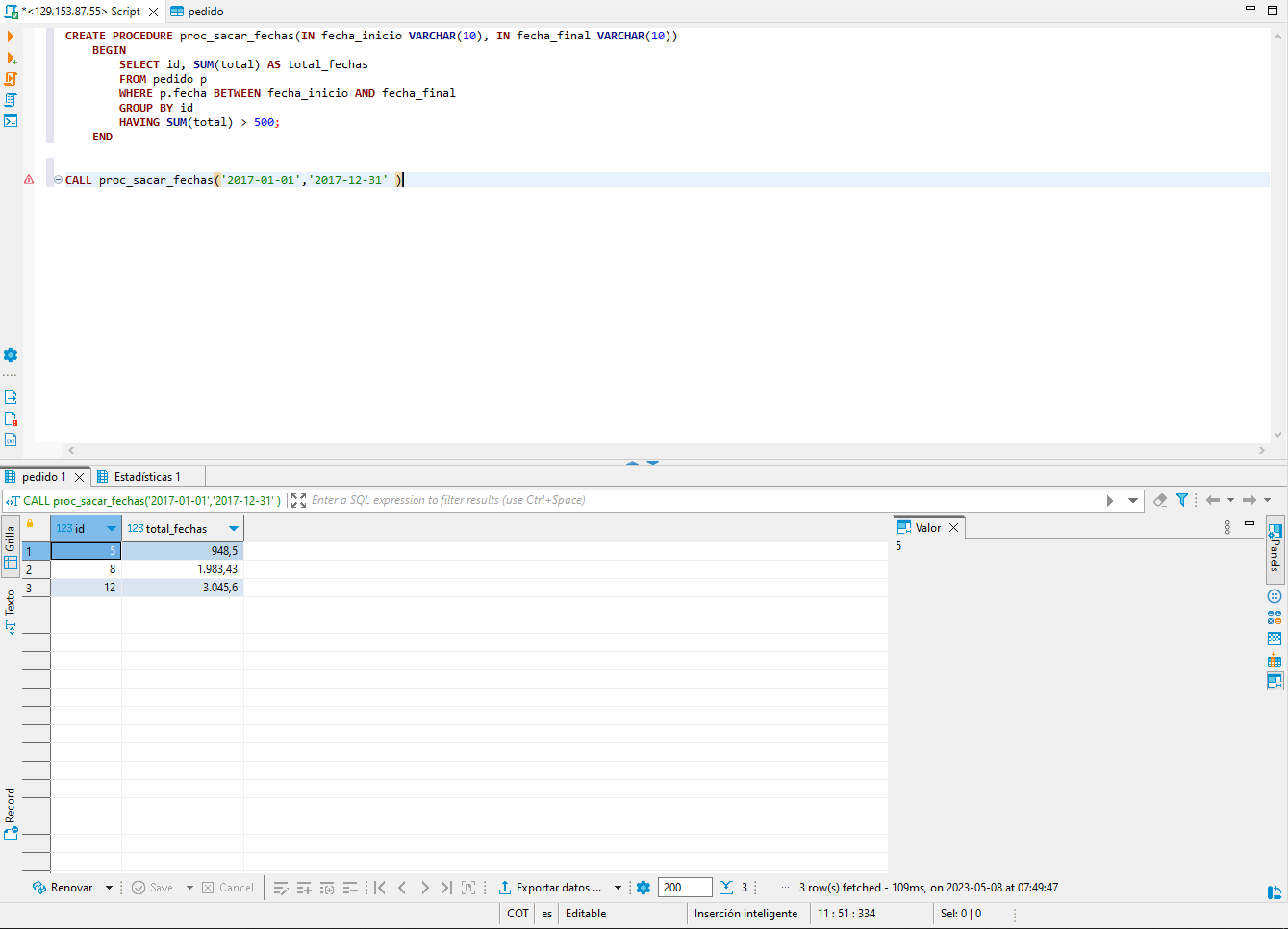
WHERE p.fecha BETWEEN fecha\_inicio AND fecha\_final

GROUP BY id

HAVING SUM(total) > 500;

END

CALL proc\_sacar\_fechas('2017-01-01','2017-12-31' )

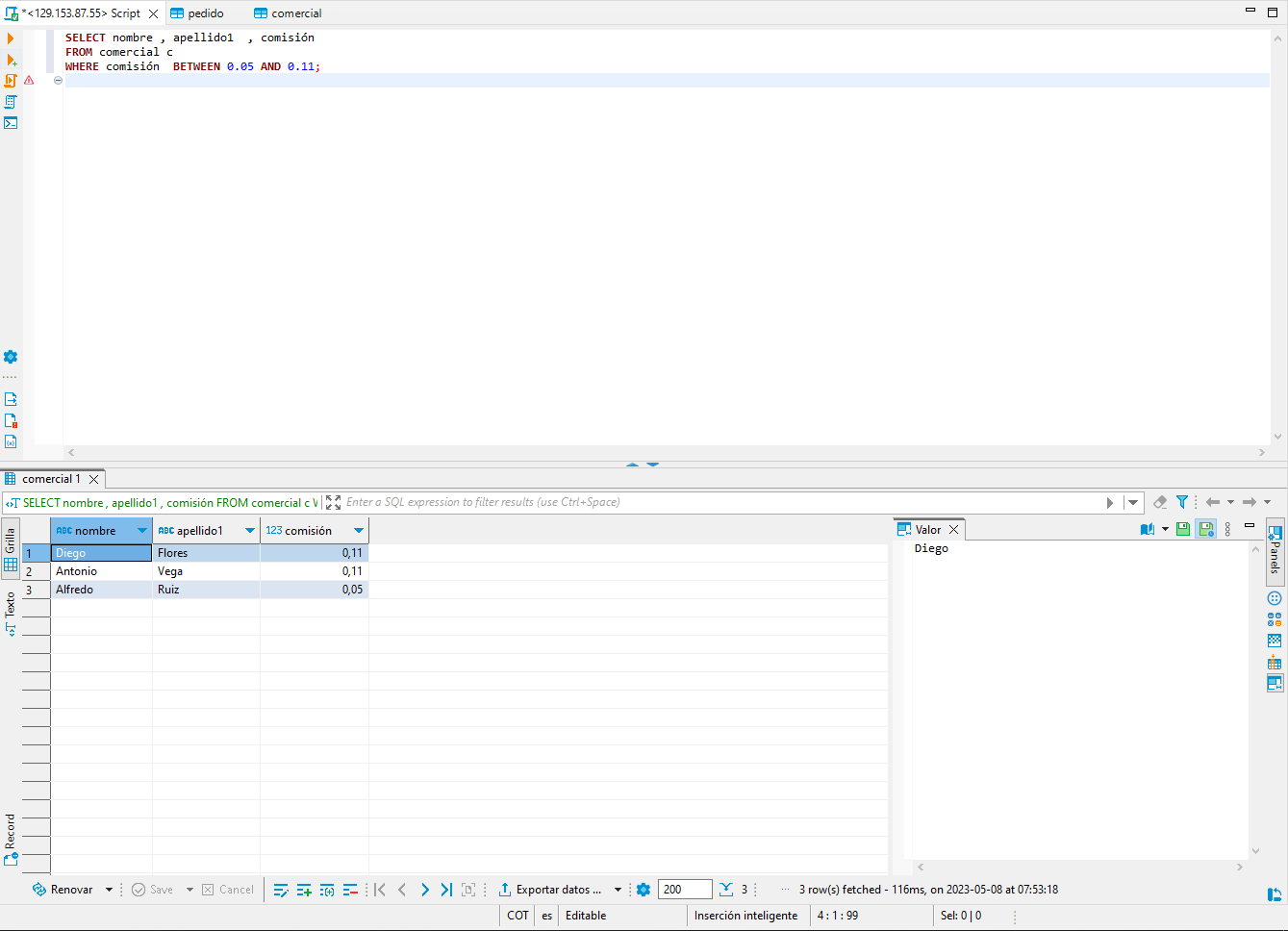


1. Devuelve un listado con el nombre y los apellidos de los comerciales que tienen una comisión entre 0.05 y 0.11.

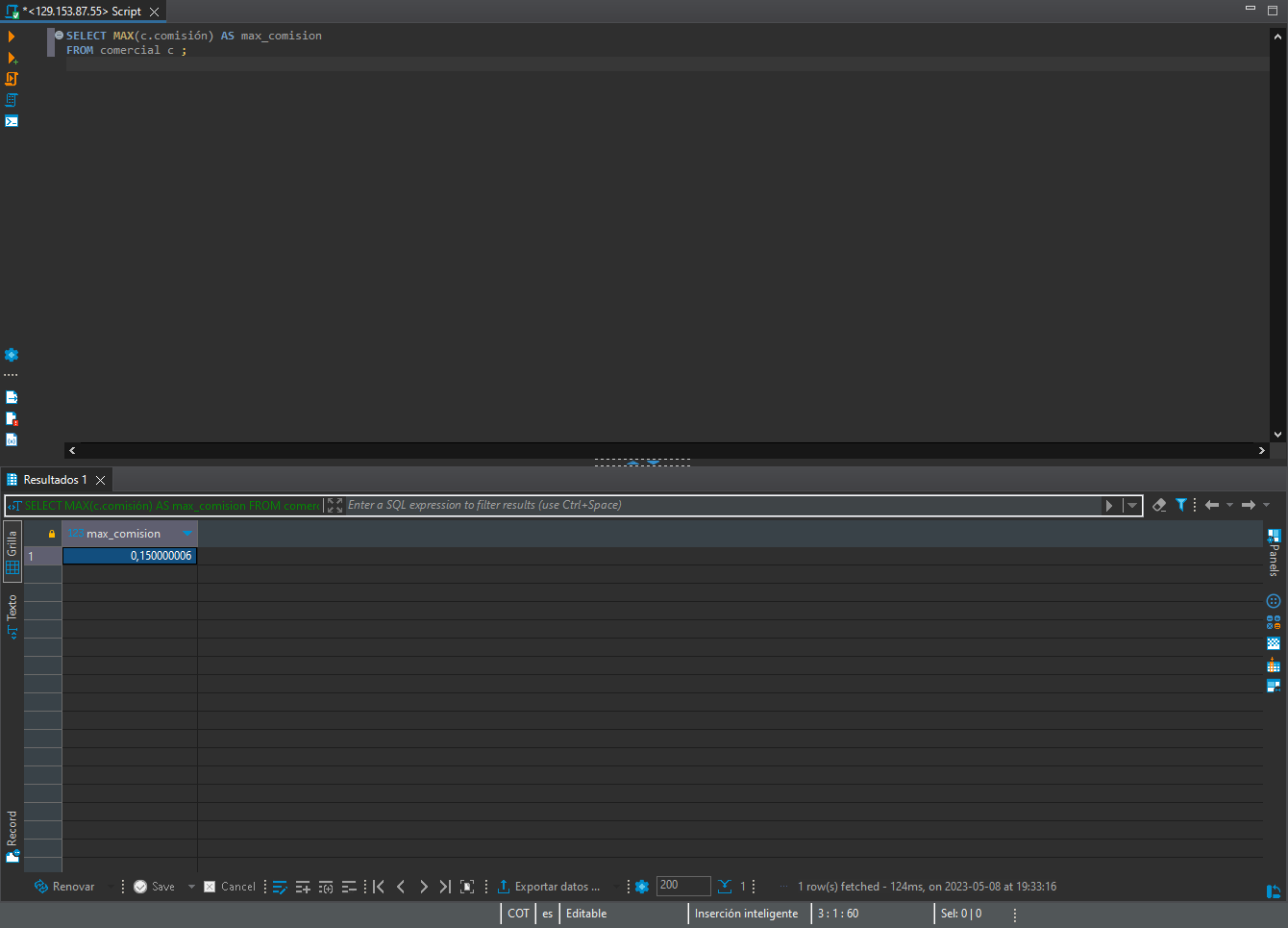
SELECT nombre , apellido1 , comisión

FROM comercial c

WHERE comisión BETWEEN 0.05 AND 0.11;



1. Devuelve el valor de la comisión de mayor valor que existe en la tabla comercia



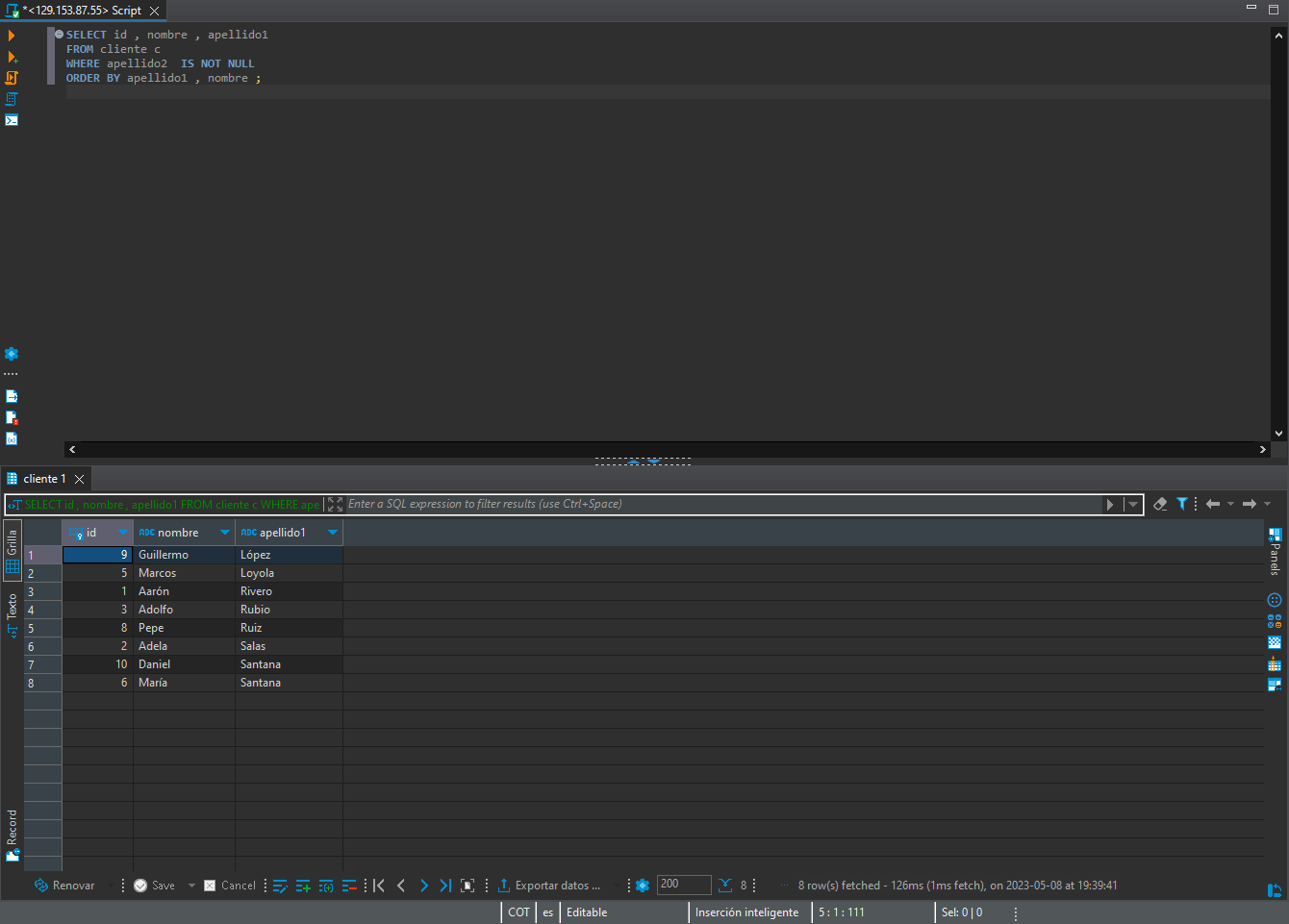
1. Devuelve el identificador, nombre y primer apellido de aquellos clientes cuyo segundo apellido no es NULL. El listado deberá estar ordenado alfabéticamente por apellidos y nombre

SELECT id , nombre , apellido1

FROM cliente c

WHERE apellido2 IS NOT NULL

ORDER BY apellido1 , nombre ;



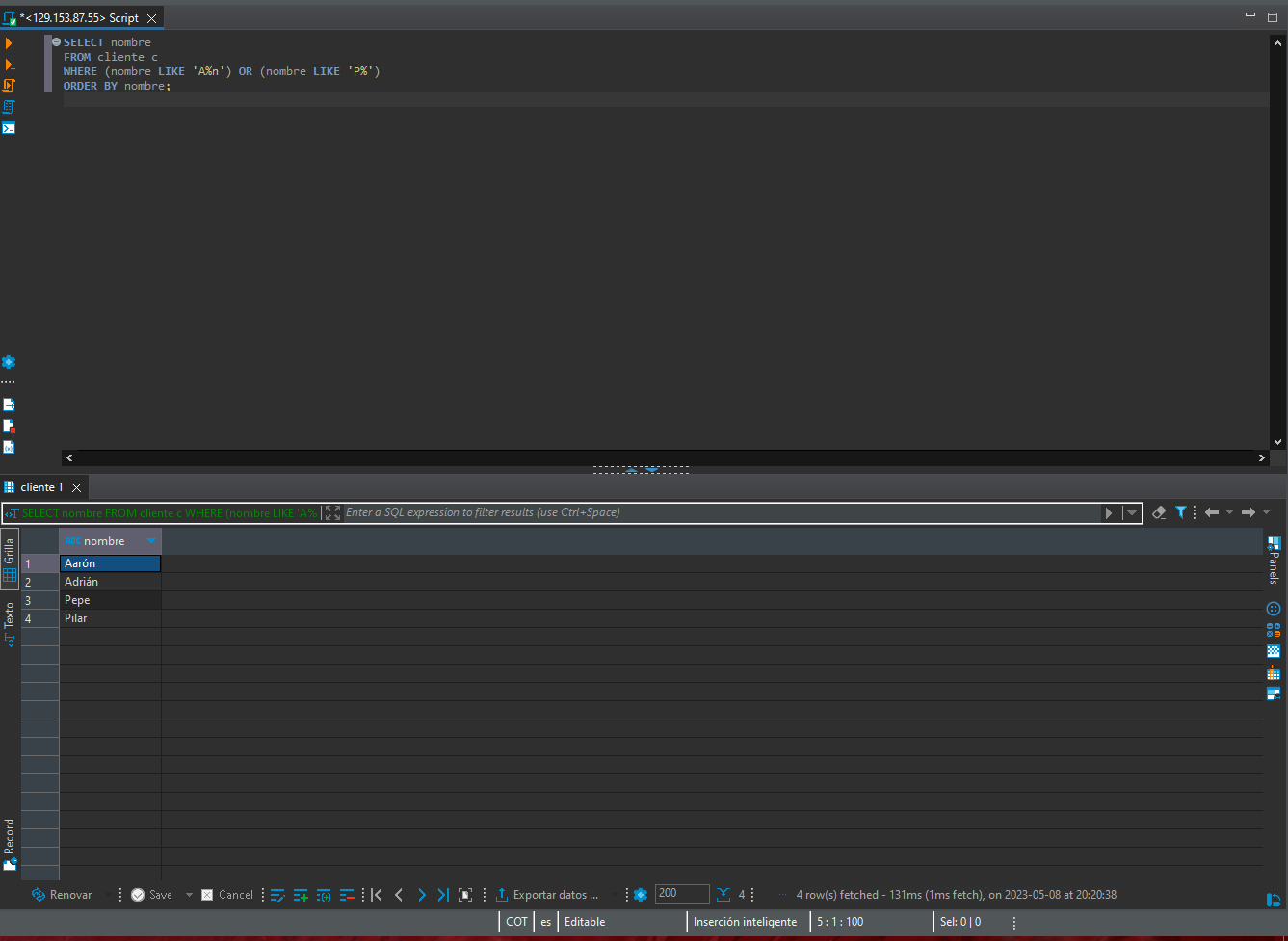
1. Devuelve un listado de los nombres de los clientes que empiezan por A y terminan por n y también los nombres que empiezan por P. El listado deberá estar ordenado alfabéticamente.

SELECT nombre

FROM cliente c

WHERE (nombre LIKE 'A%n') OR (nombre LIKE 'P%')

ORDER BY nombre;



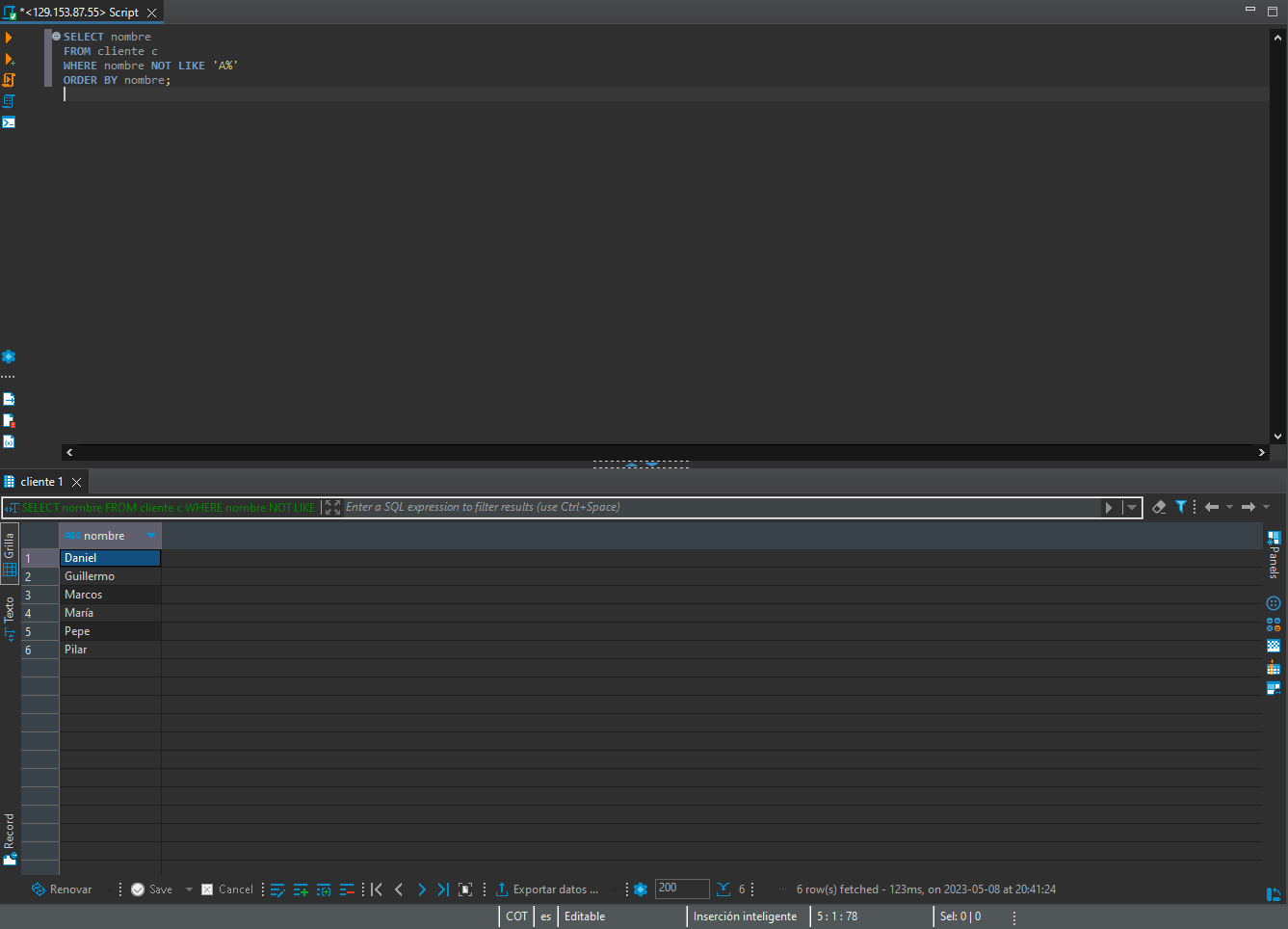
1. Devuelve un listado de los nombres de los clientes que no empiezan por A. El listado deberá estar ordenado alfabéticamente

SELECT nombre

FROM cliente c

WHERE nombre NOT LIKE 'A%'

ORDER BY nombre;

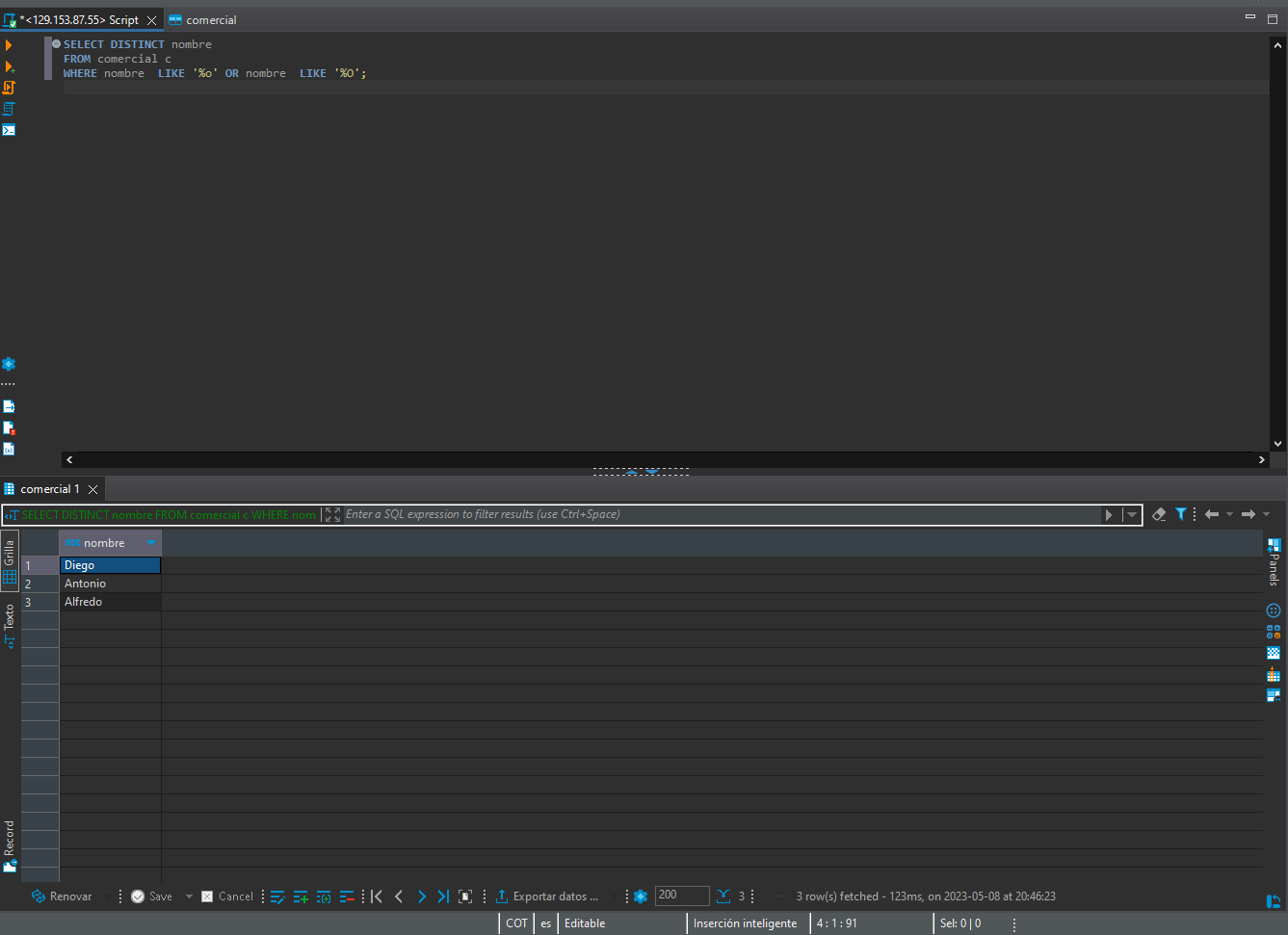


1. Devuelve un listado con los nombres de los comerciales que terminan por el o o. Tenga en cuenta que se deberán eliminar los nombres repetidos

SELECT DISTINCT nombre

FROM comercial c

WHERE nombre LIKE '%o' OR nombre LIKE '%O';



* Consultas multitabla

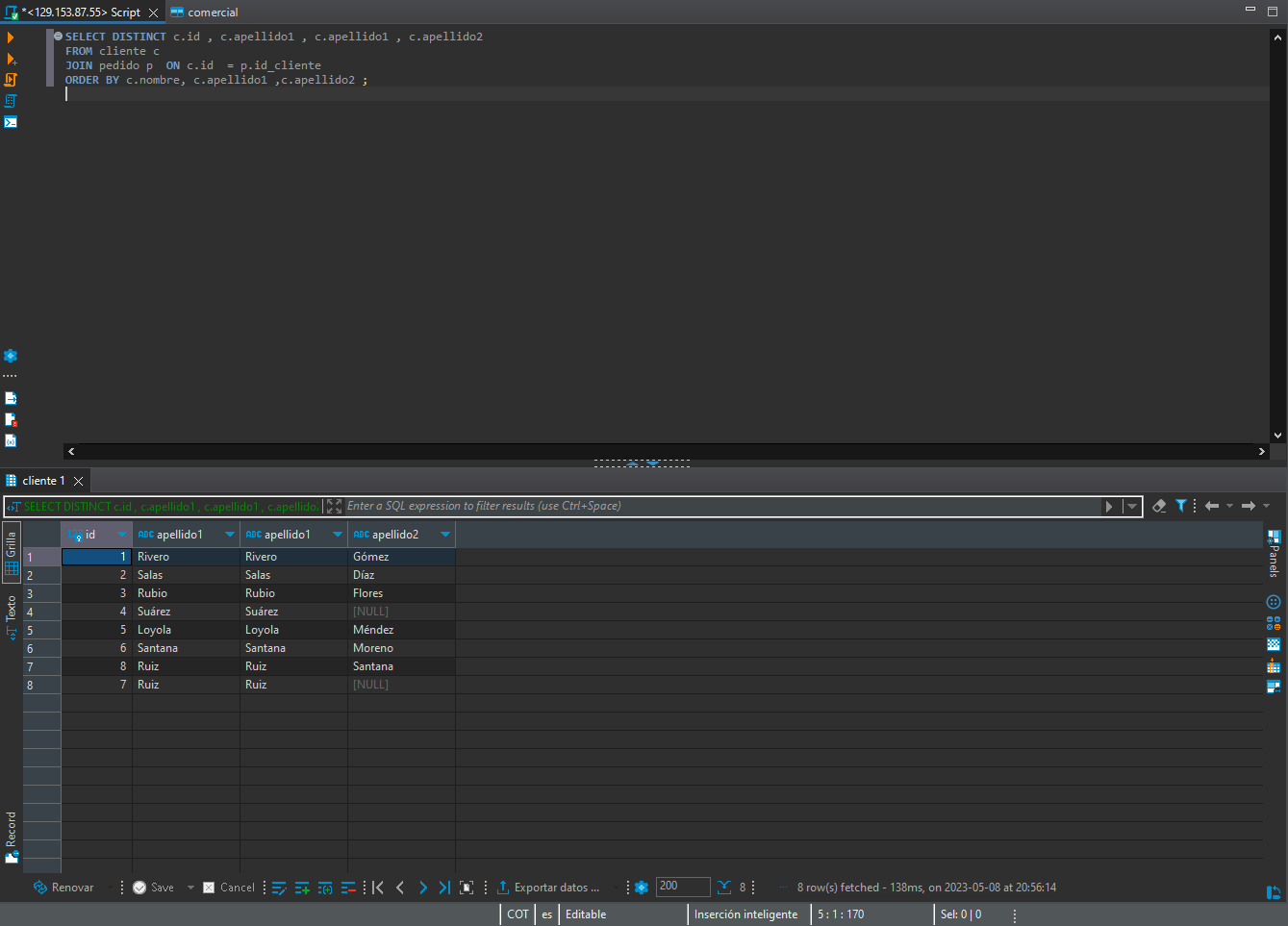
1. Devuelve un listado con el identificador, nombre y los apellidos de todos los clientes que han realizado algún pedido. El listado debe estar ordenado alfabéticamente y se deben eliminar los elementos repetidos

SELECT DISTINCT c.id , c.apellido1 , c.apellido1 , c.apellido2

FROM cliente c

JOIN pedido p ON c.id = p.id\_cliente

ORDER BY c.nombre, c.apellido1 ,c.apellido2 ;



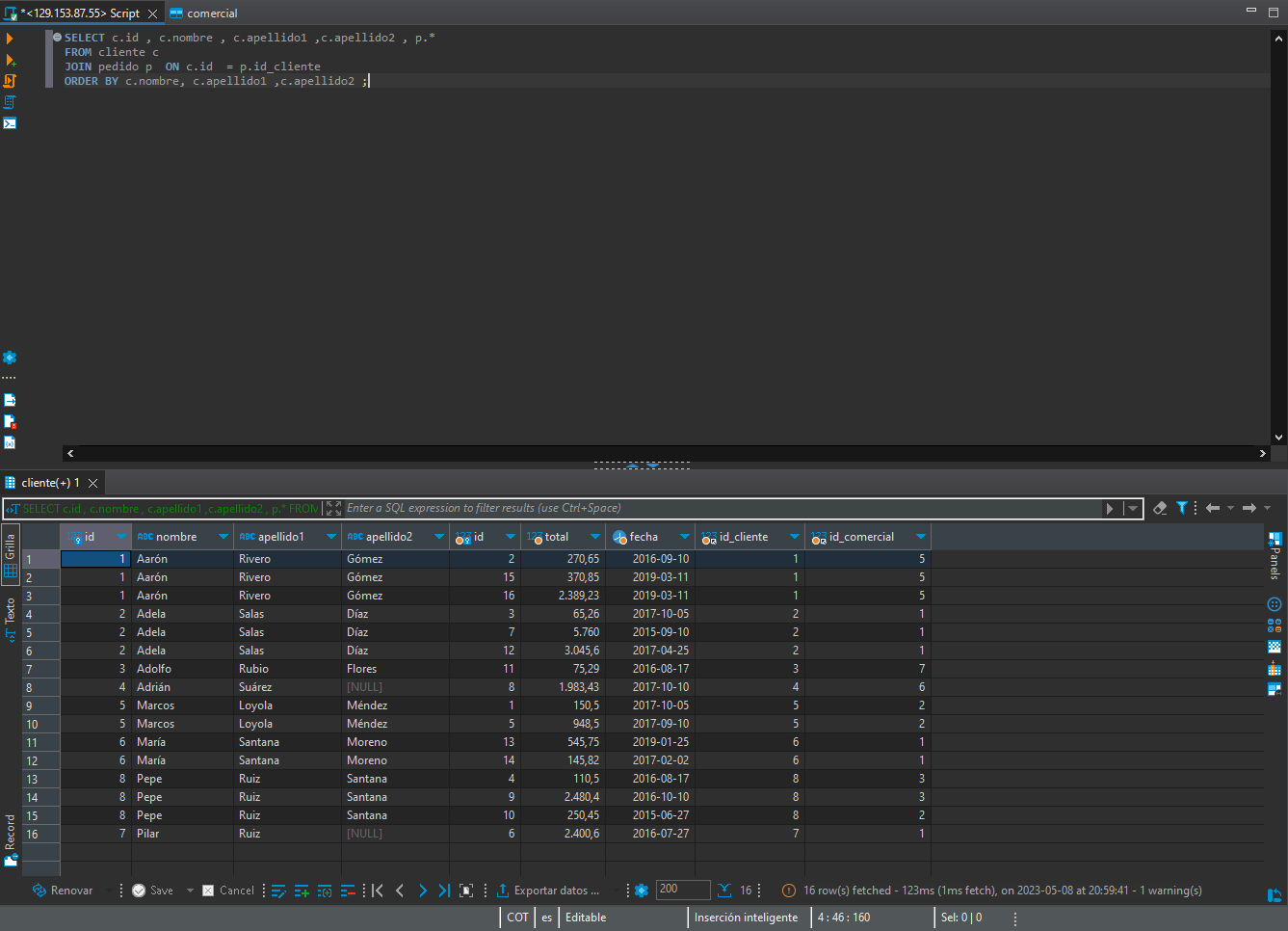
1. Devuelve un listado que muestre todos los pedidos que ha realizado cada cliente. El resultado debe mostrar todos los datos de los pedidos y del cliente. El listado debe mostrar los datos de los clientes ordenados alfabéticamente

SELECT c.id , c.nombre , c.apellido1 ,c.apellido2 , p.\*

FROM cliente c

JOIN pedido p ON c.id = p.id\_cliente

ORDER BY c.nombre, c.apellido1 ,c.apellido2 ;



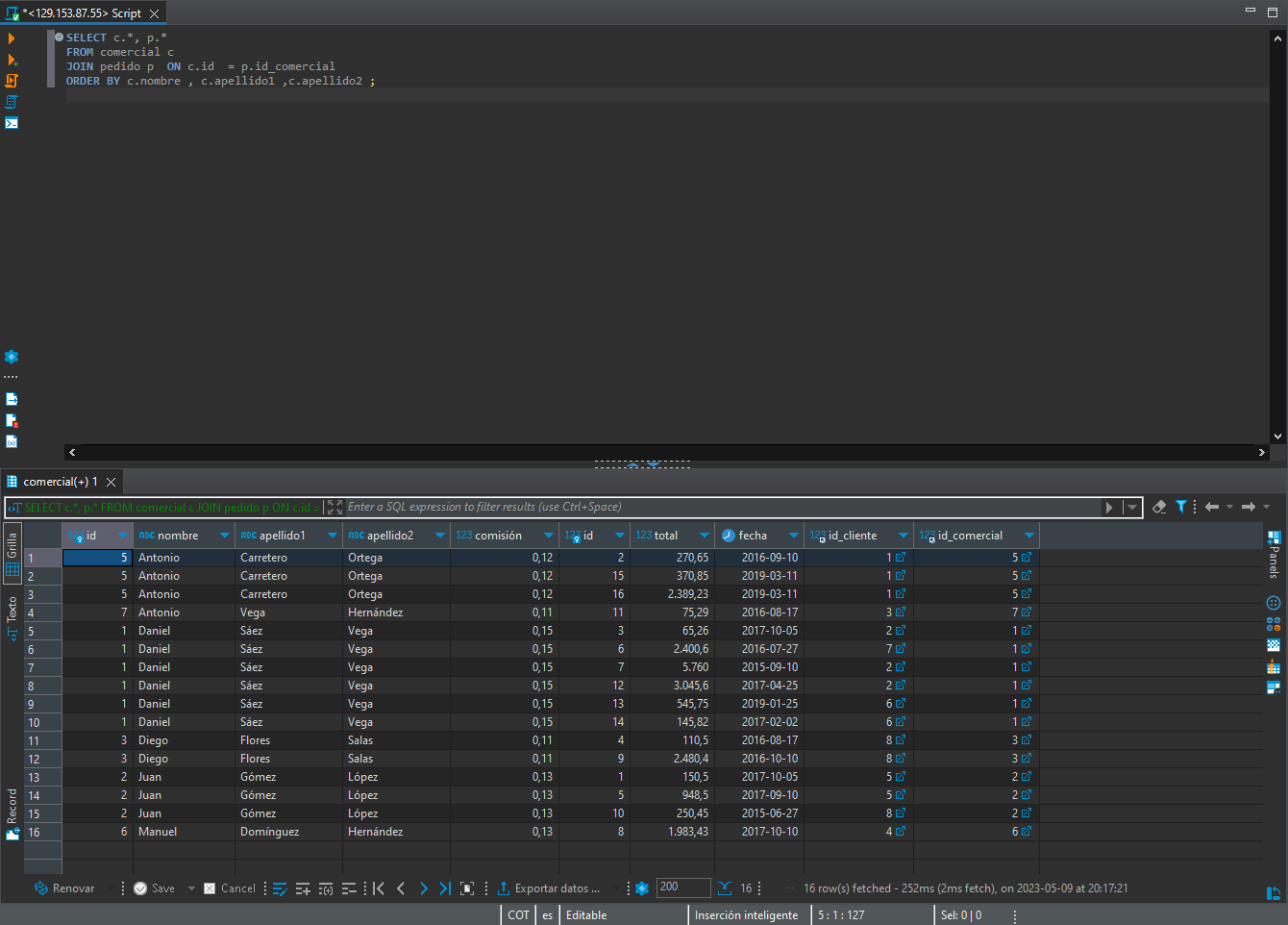
1. Devuelve un listado que muestre todos los pedidos en los que ha participado un comercial. El resultado debe mostrar todos los datos de los pedidos y de los comerciales. El listado debe mostrar los datos de los comerciales ordenados alfabéticamente.

SELECT c.\*, p.\*

FROM comercial c

JOIN pedido p ON c.id = p.id\_comercial

ORDER BY c.nombre , c.apellido1 ,c.apellido2 ;



1. Devuelve un listado que muestre todos los clientes, con todos los pedidos que han realizado y con los datos de los comerciales asociados a cada pedido

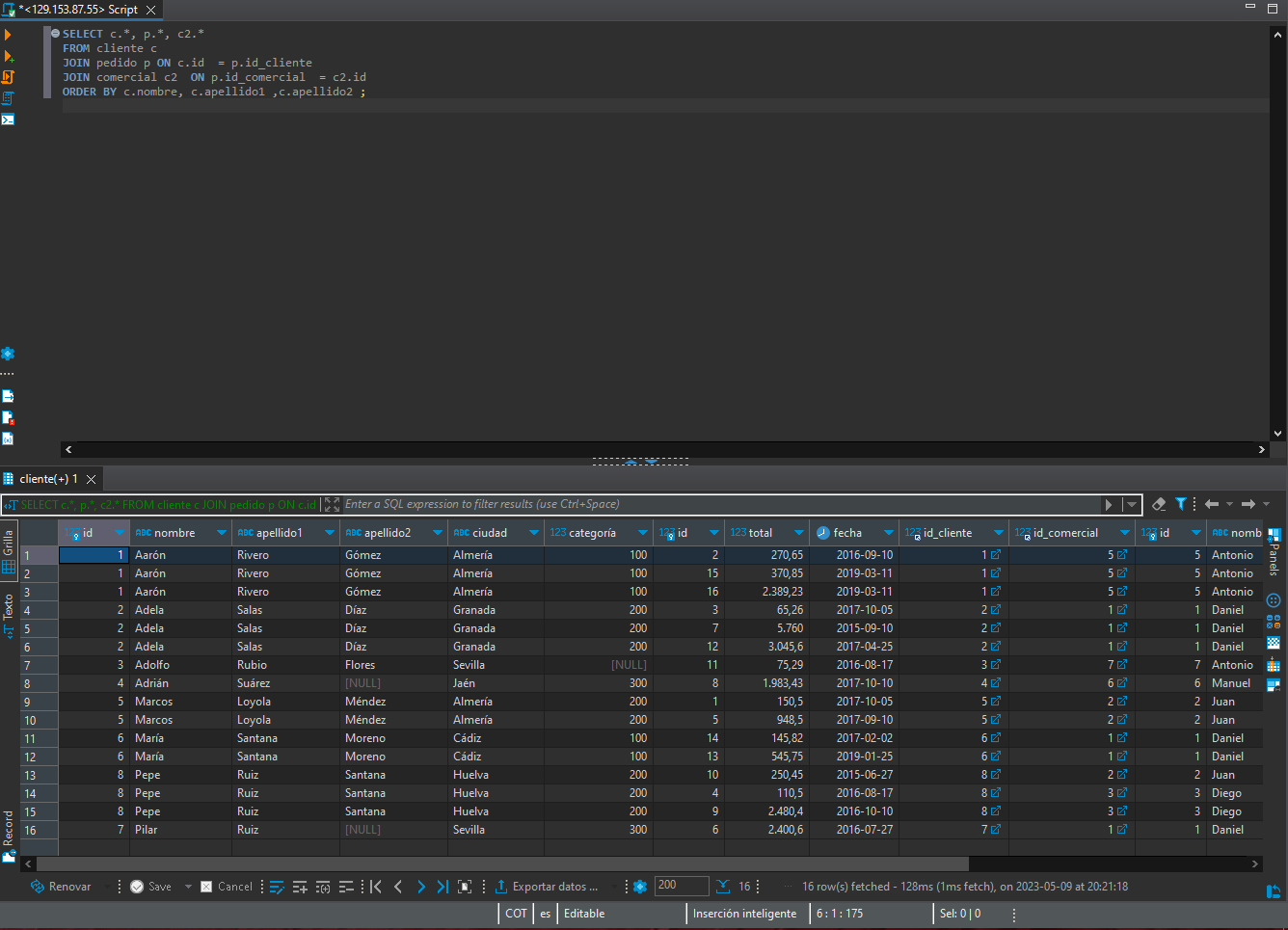
SELECT c.\*, p.\*, c2.\*

FROM cliente c

JOIN pedido p ON c.id = p.id\_cliente

JOIN comercial c2 ON p.id\_comercial = c2.id

ORDER BY c.nombre, c.apellido1 ,c.apellido2 ;



1. Devuelve un listado de todos los clientes que realizaron un pedido durante el año 2017, cuya cantidad esté entre 300 € y 1000 €.

SELECT DISTINCT c.\*,p.total

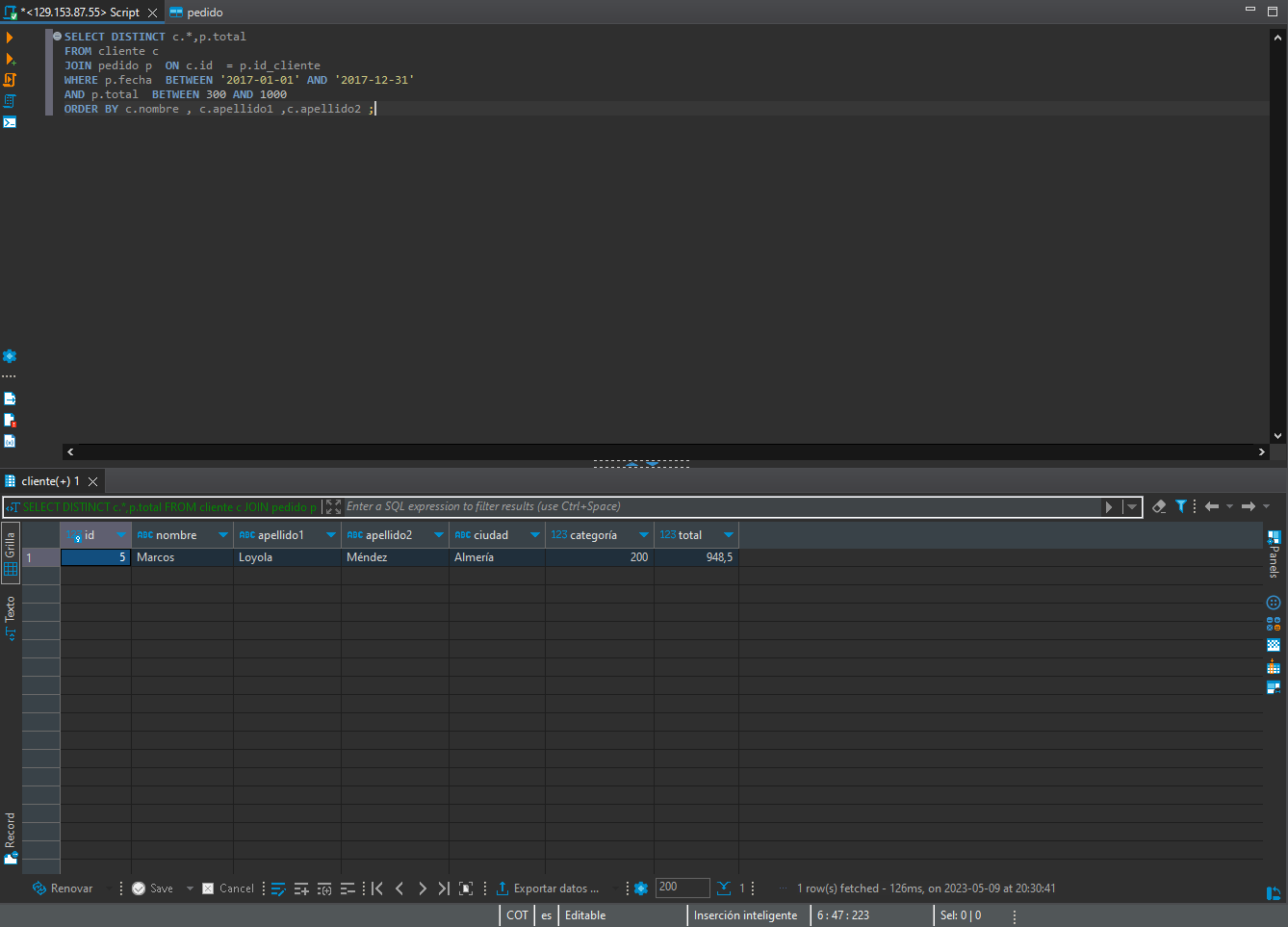
FROM cliente c

JOIN pedido p ON c.id = p.id\_cliente

WHERE p.fecha BETWEEN '2017-01-01' AND '2017-12-31'

AND p.total BETWEEN 300 AND 1000

ORDER BY c.nombre , c.apellido1 ,c.apellido2 ;



1. Devuelve el nombre y los apellidos de todos los comerciales que ha participado en algún pedido realizado por María Santana Moreno

SELECT DISTINCT c2.nombre, c2.apellido1 ,c2.apellido2

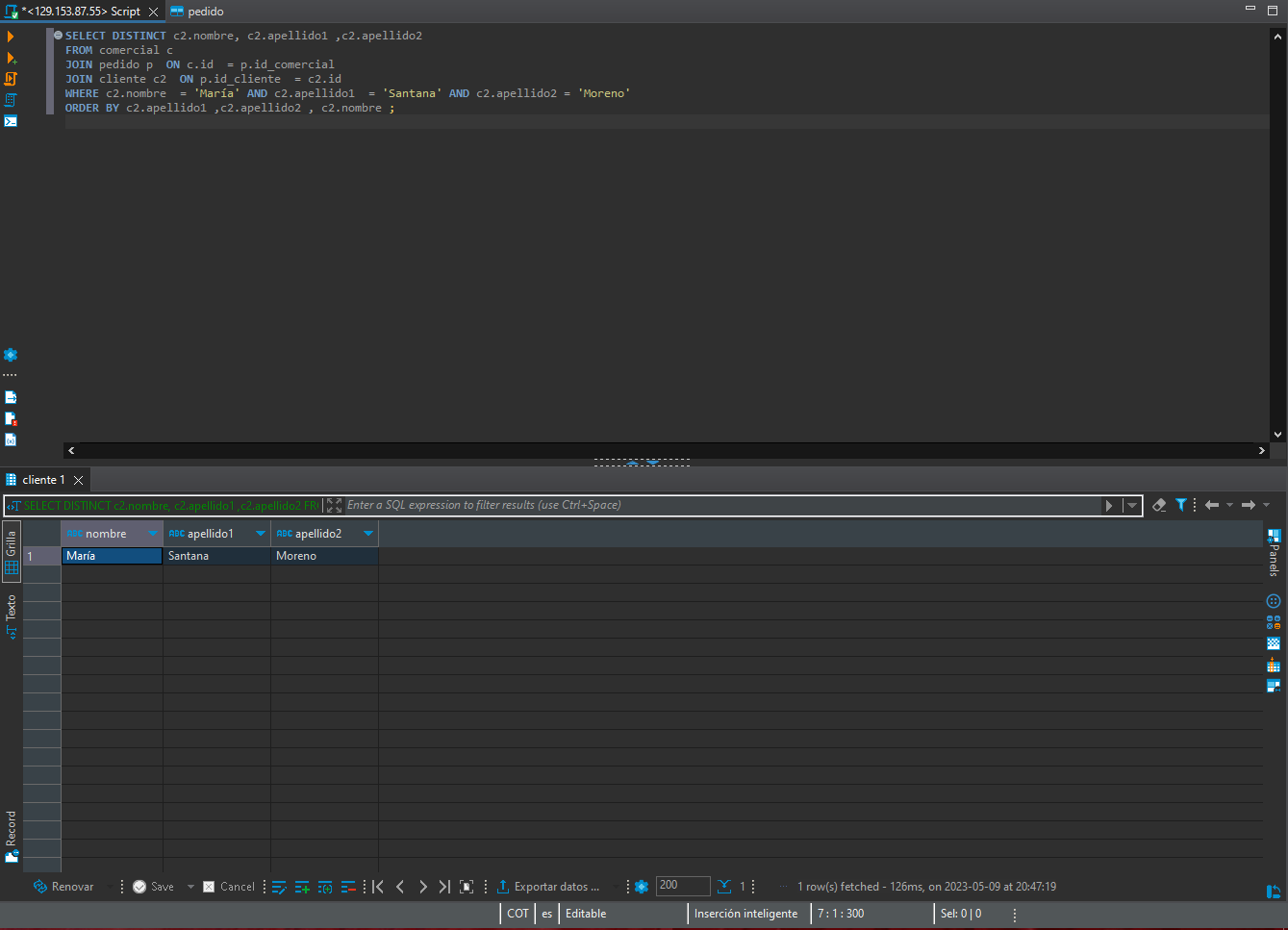
FROM comercial c

JOIN pedido p ON c.id = p.id\_comercial

JOIN cliente c2 ON p.id\_cliente = c2.id

WHERE c2.nombre = 'María' AND c2.apellido1 = 'Santana' AND c2.apellido2 = 'Moreno'

ORDER BY c2.apellido1 ,c2.apellido2 , c2.nombre ;



1. Devuelve el nombre de todos los clientes que han realizado algún pedido con el comercial Daniel Sáez Vega

SELECT DISTINCT c.nombre

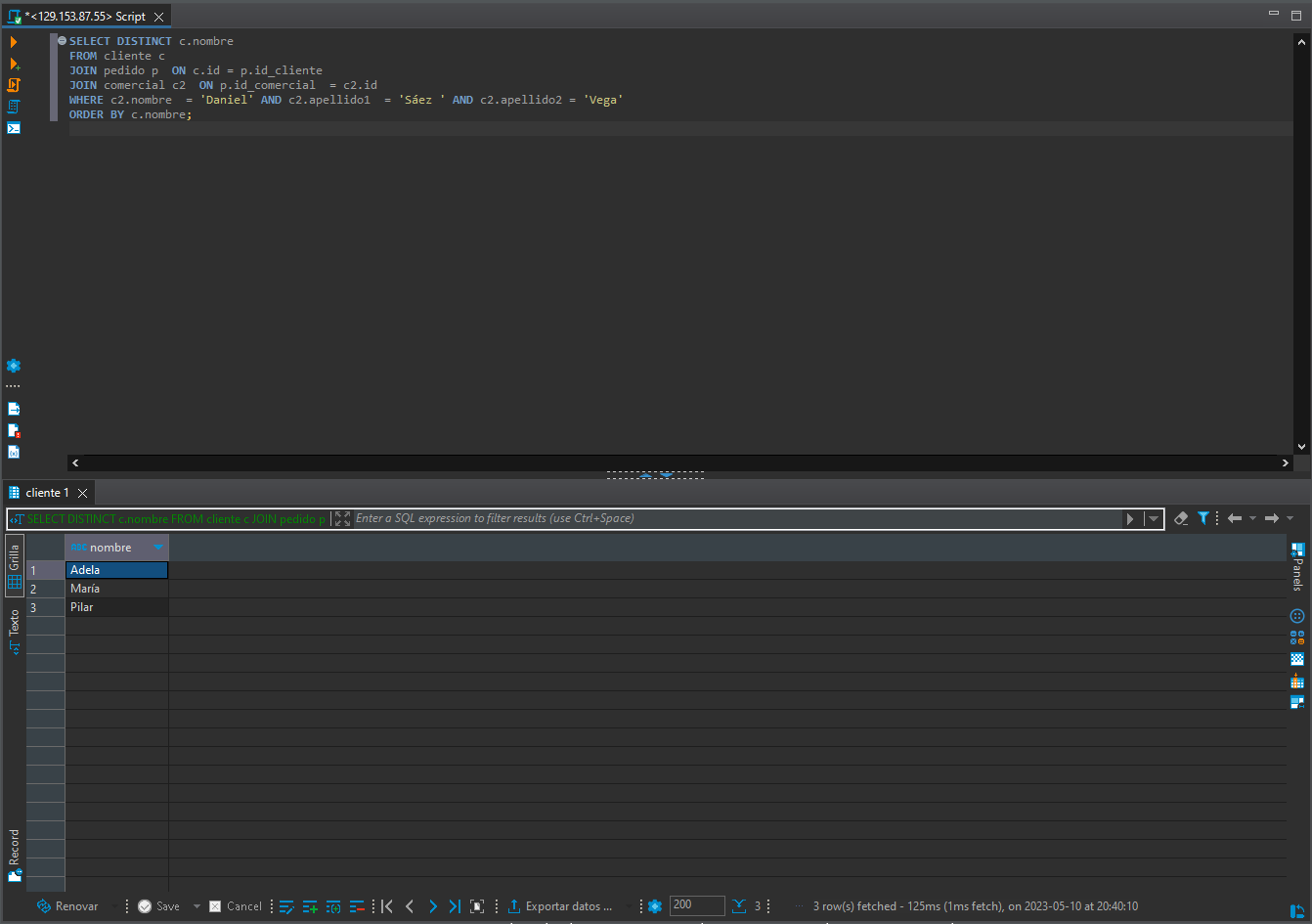
FROM cliente c

JOIN pedido p ON c.id = p.id\_cliente

JOIN comercial c2 ON p.id\_comercial = c2.id

WHERE c2.nombre = 'Daniel' AND c2.apellido1 = 'Sáez ' AND c2.apellido2 = 'Vega'

ORDER BY c.nombre;



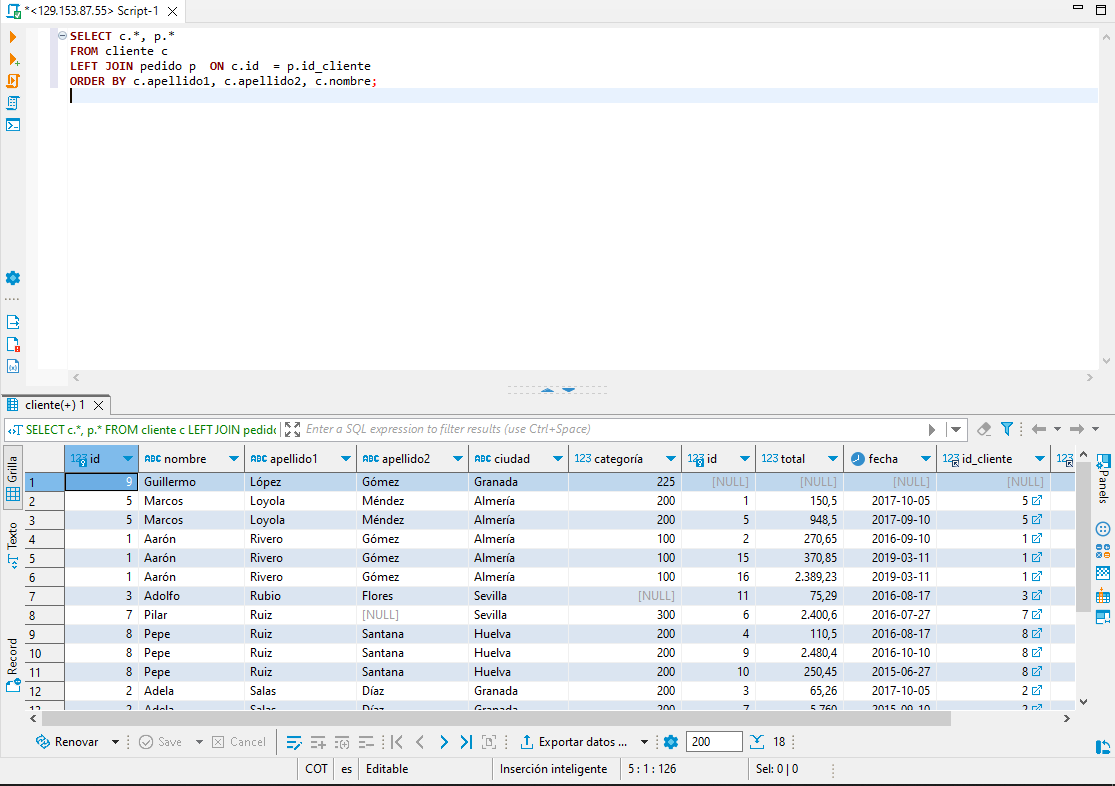
1. Devuelve un listado con todos los clientes junto con los datos de los pedidos que han realizado. Este listado también debe incluir los clientes que no han realizado ningún pedido. El listado debe estar ordenado alfabéticamente por el primer apellido, segundo apellido y nombre de los clientes.

SELECT c.\*, p.\*

FROM cliente c

LEFT JOIN pedido p ON c.id = p.id\_cliente

ORDER BY c.apellido1, c.apellido2, c.nombre



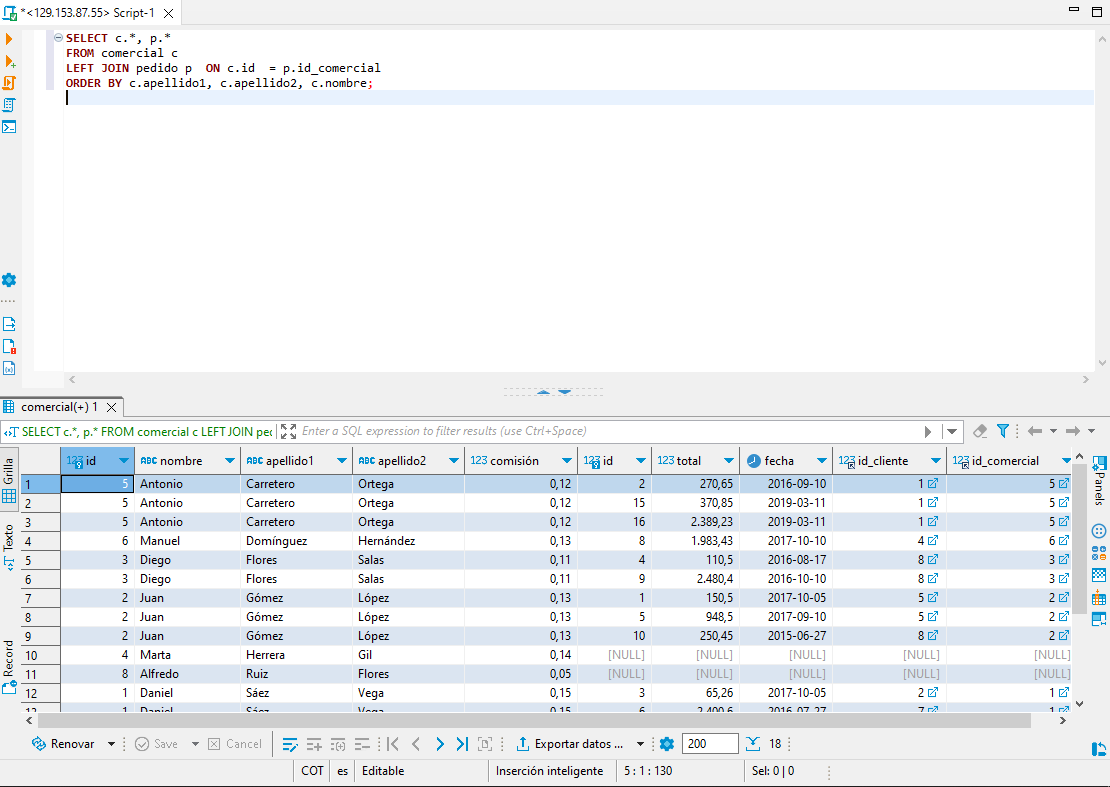
1. Devuelve un listado con todos los comerciales junto con los datos de los pedidos que han realizado. Este listado también debe incluir los comerciales que no han realizado ningún pedido. El listado debe estar ordenado alfabéticamente por el primer apellido, segundo apellido y nombre de los comerciales.

SELECT c.\*, p.\*

FROM comercial c

LEFT JOIN pedido p ON c.id = p.id\_comercial

ORDER BY c.apellido1, c.apellido2, c.nombre



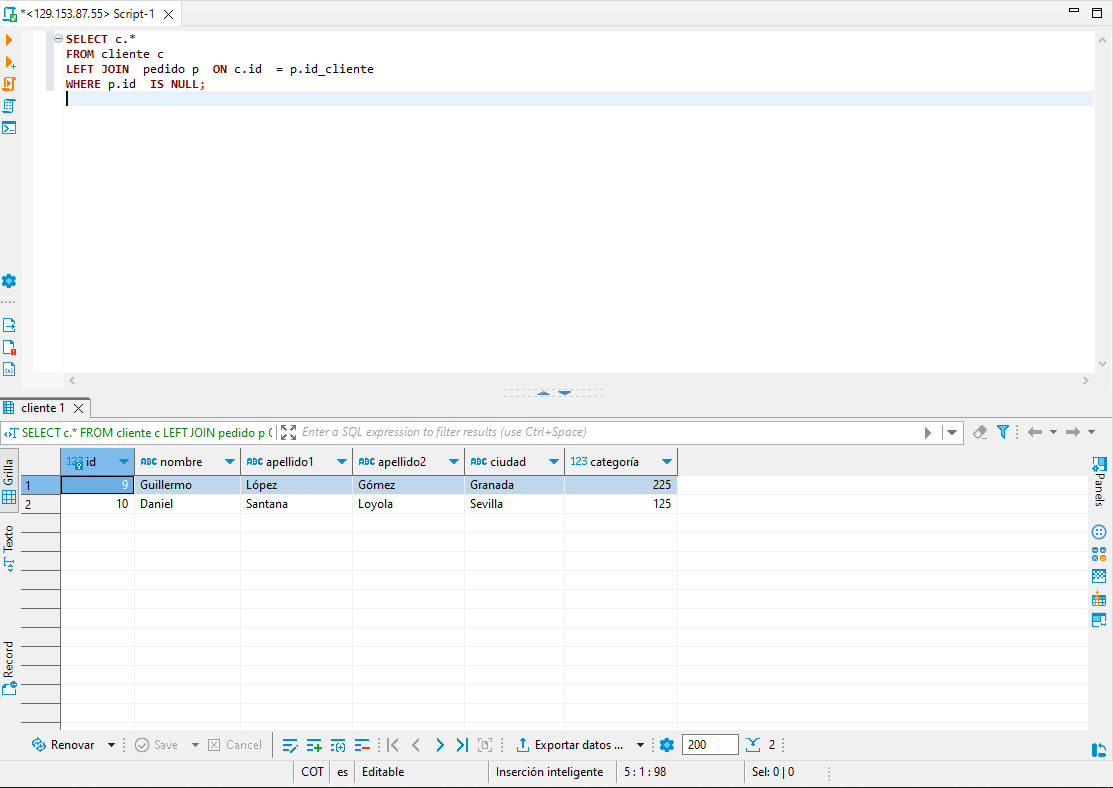
1. Devuelve un listado que solamente muestre los clientes que no han realizado ningún pedido.

SELECT c.\*

FROM cliente c

LEFT JOIN pedido p ON c.id = p.id\_cliente

WHERE p.id IS NULL;



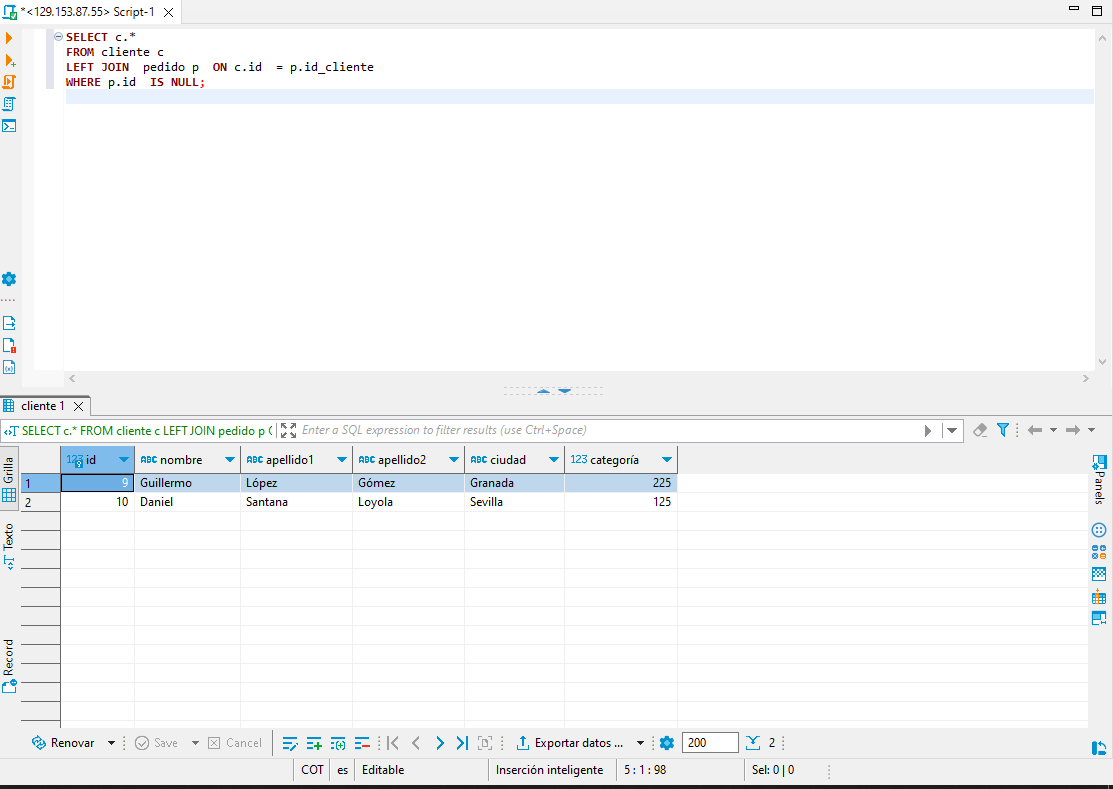
1. Devuelve un listado que solamente muestre los comerciales que no han realizado ningún pedido.

SELECT c.\*

FROM cliente c

LEFT JOIN pedido p ON c.id = p.id\_cliente

WHERE p.id IS NULL;



1. Devuelve un listado con los clientes que no han realizado ningún pedido y de los comerciales que no han participado en ningún pedido. Ordene el listado alfabéticamente por los apellidos y el nombre. En el listado deberá diferenciar de algún modo los clientes y los comerciales.

SELECT CONCAT(apellido1 , ', ', nombre) AS nombre\_completo, 'cliente' AS tipo

FROM cliente c

WHERE c.id NOT IN (SELECT DISTINCT p.id\_cliente FROM pedido p)

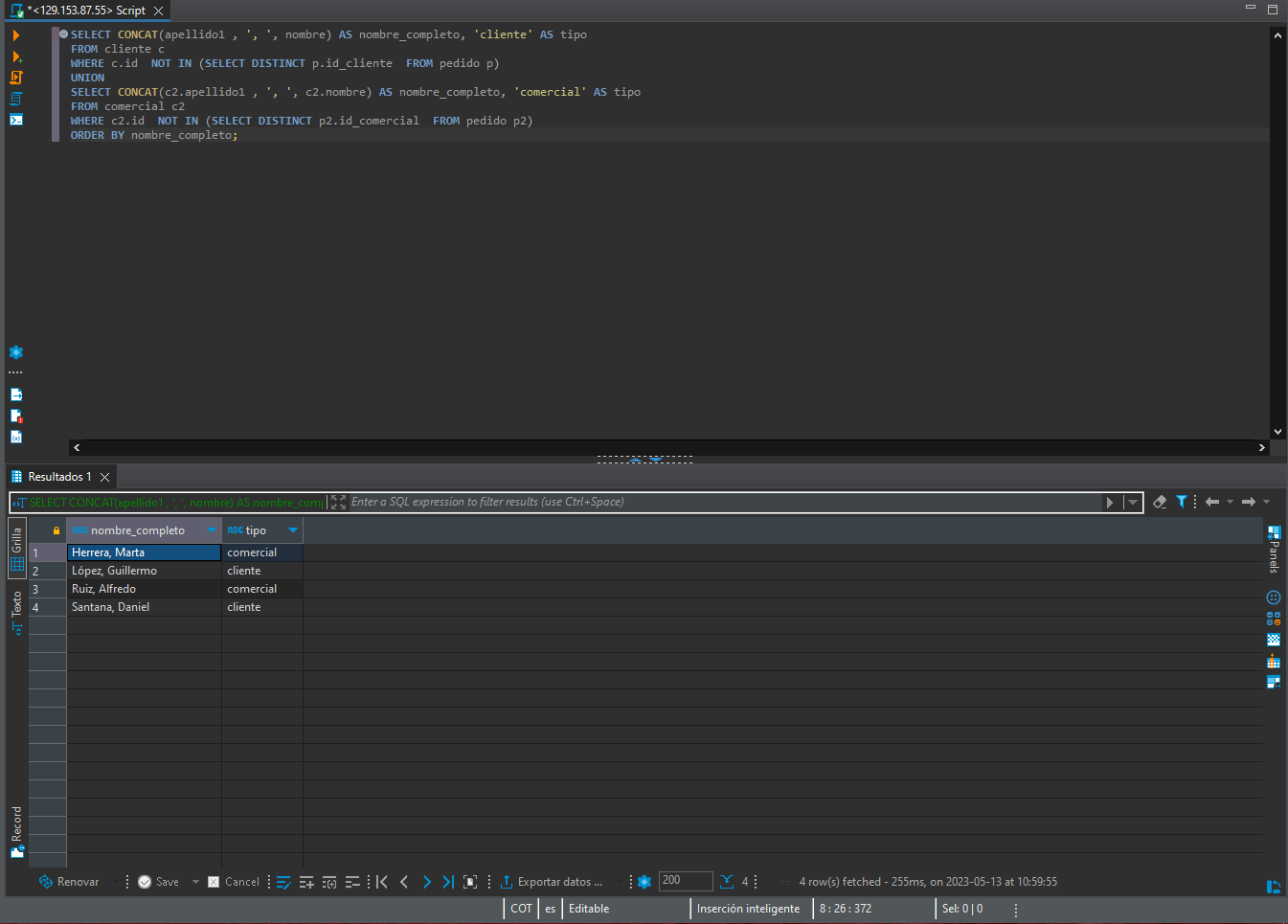
UNION

SELECT CONCAT(c2.apellido1 , ', ', c2.nombre) AS nombre\_completo, 'comercial' AS tipo

FROM comercial c2

WHERE c2.id NOT IN (SELECT DISTINCT p2.id\_comercial FROM pedido p2)

ORDER BY nombre\_completo

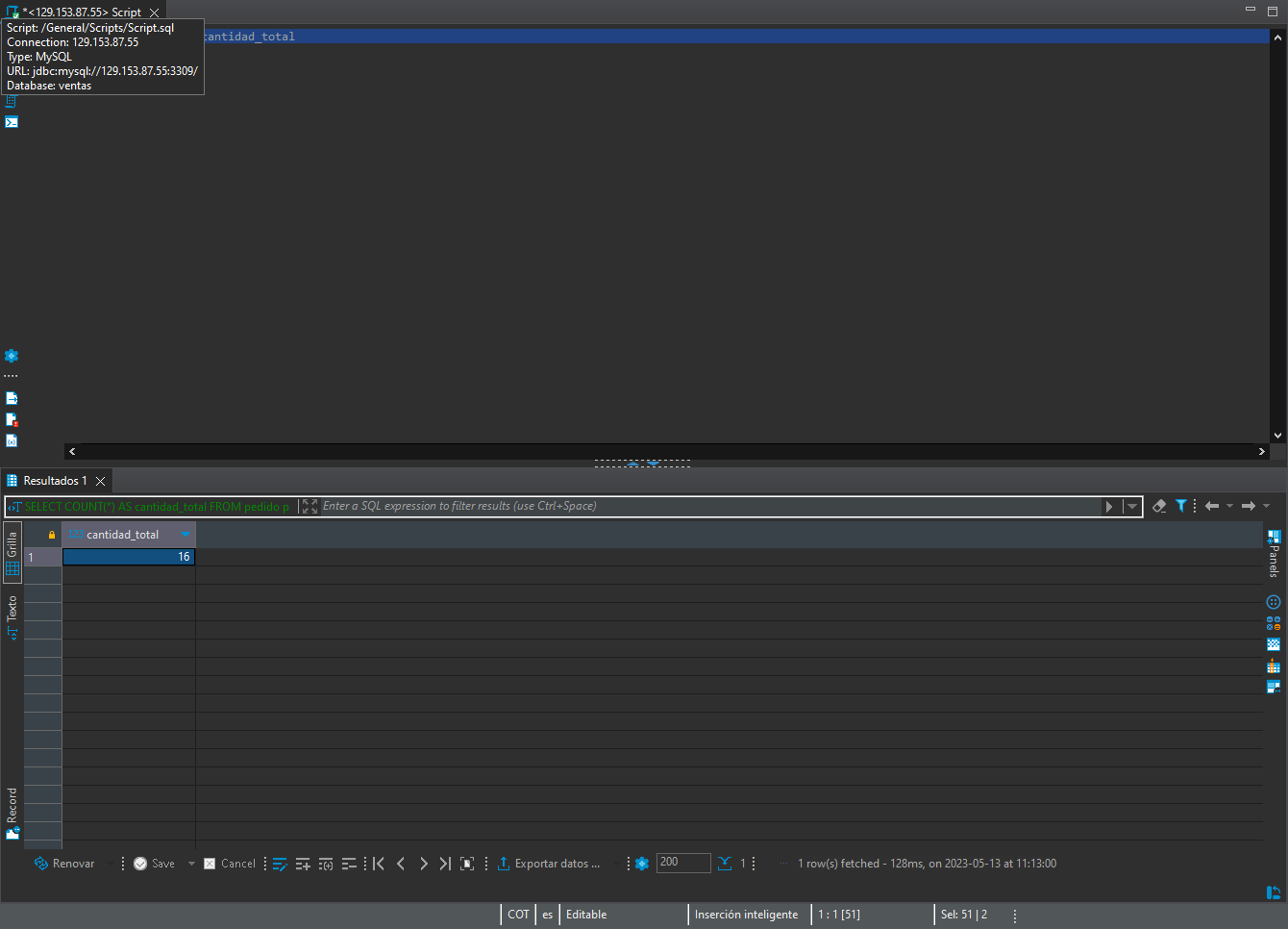


* Consultas resume

1. Calcula la cantidad total que suman todos los pedidos que aparecen en la tabla pedido

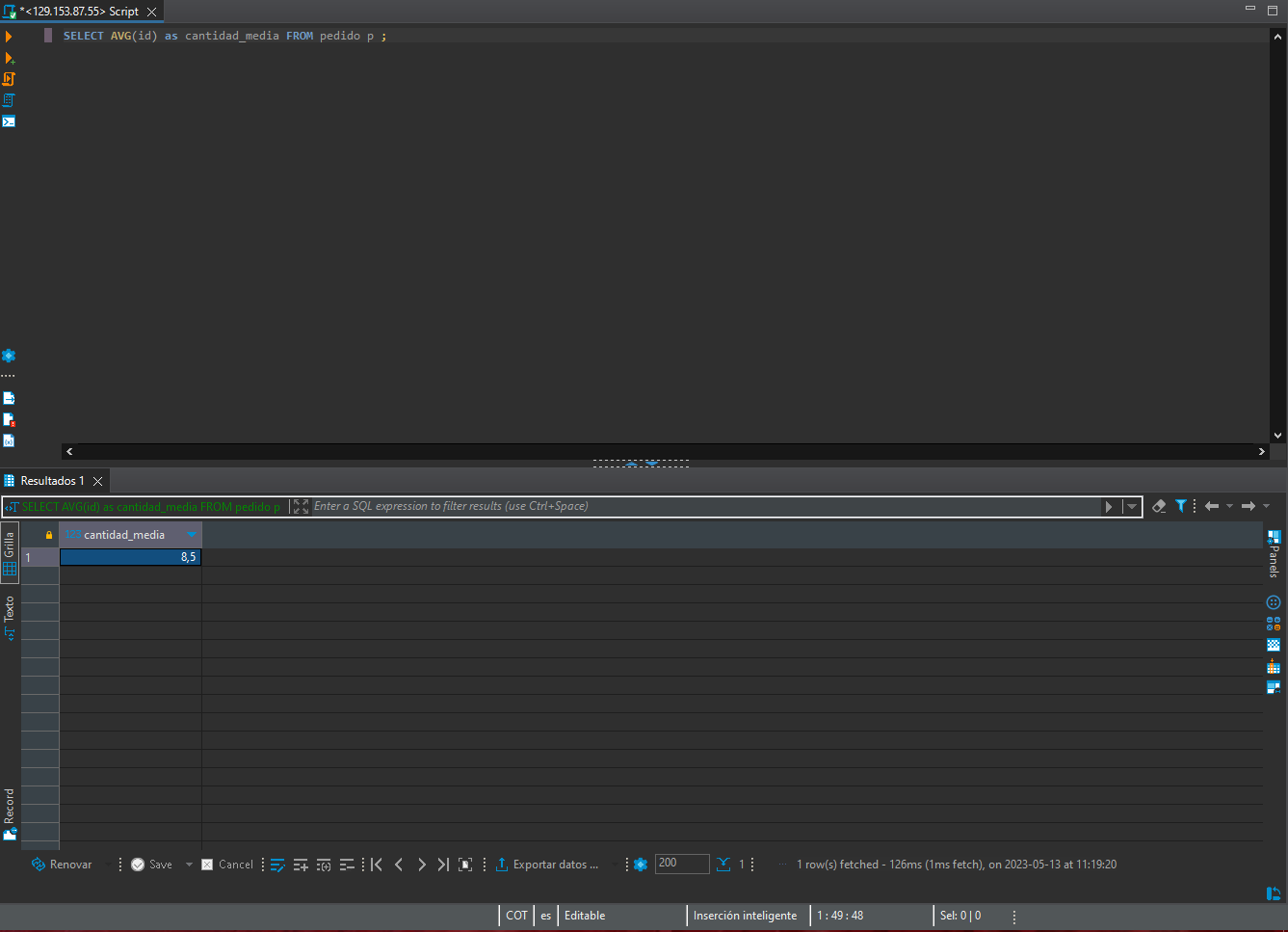
SELECT COUNT(\*) AS cantidad\_total

FROM pedido p ;



1. Calcula la cantidad media de todos los pedidos que aparecen en la tabla pedido.

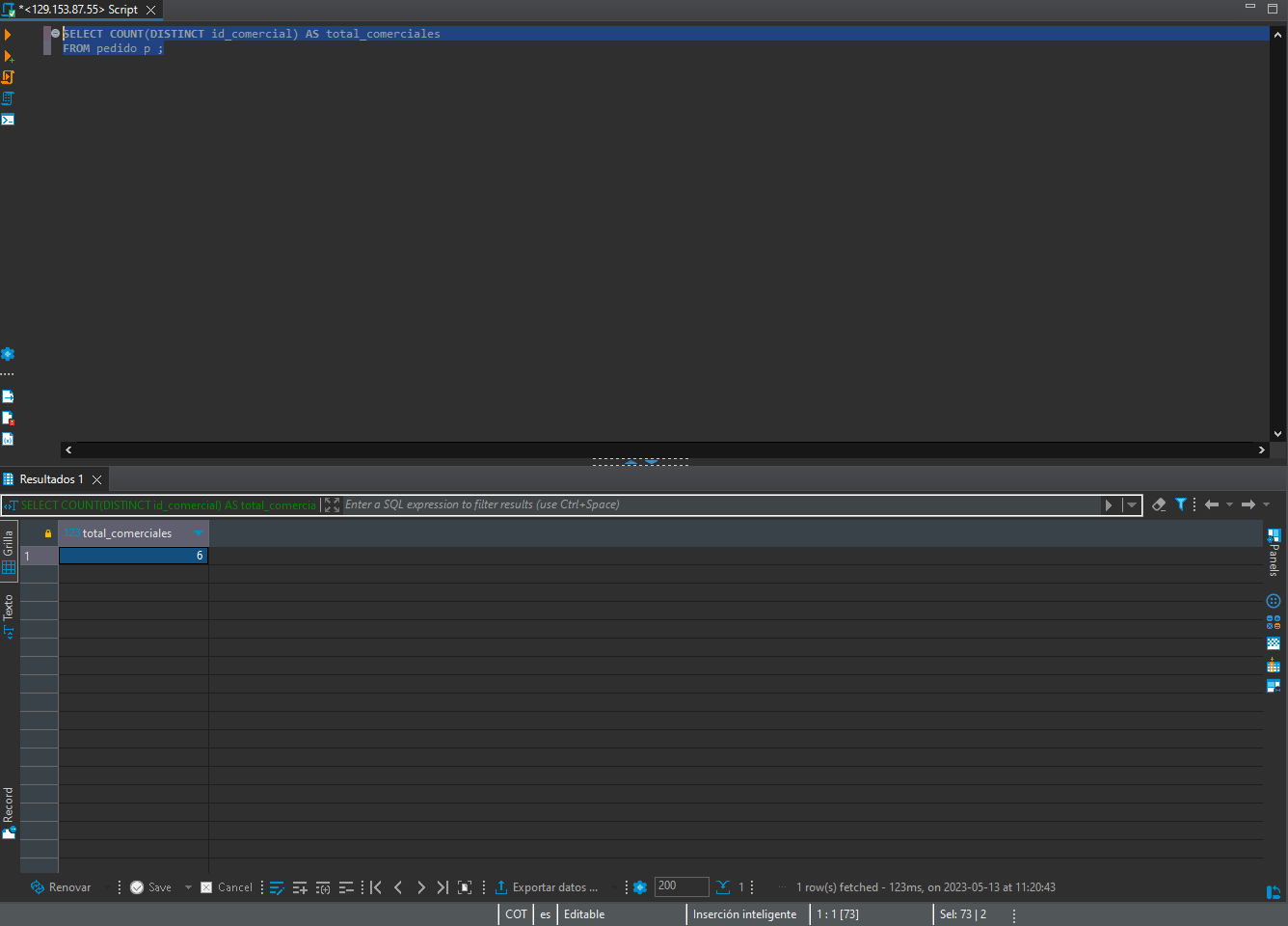
SELECT AVG(id) as cantidad\_media FROM pedido p ;



1. Calcula el número total de comerciales distintos que aparecen en la tabla pedido.

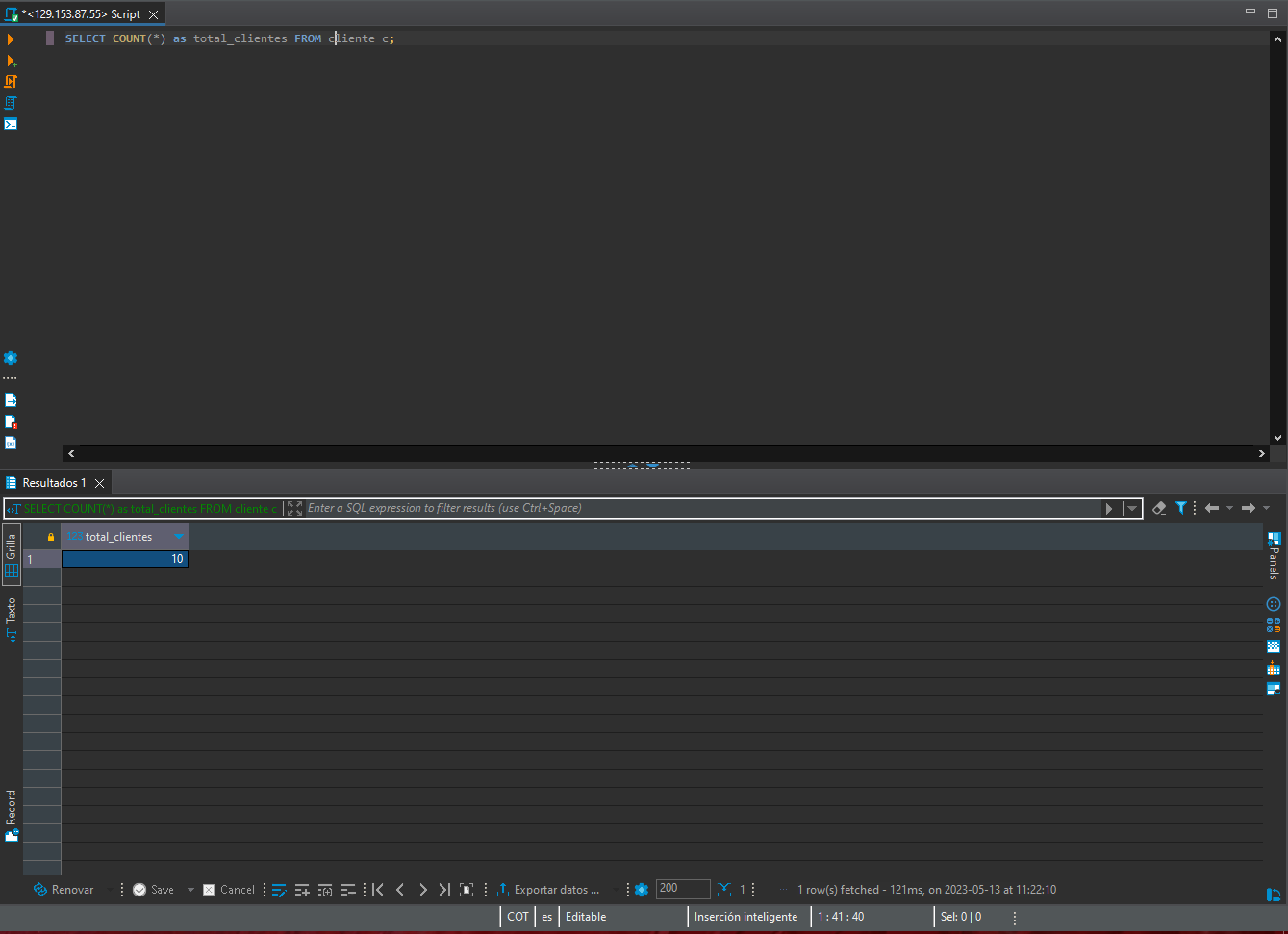
SELECT COUNT(DISTINCT id\_comercial) AS total\_comerciales

FROM pedido p ;



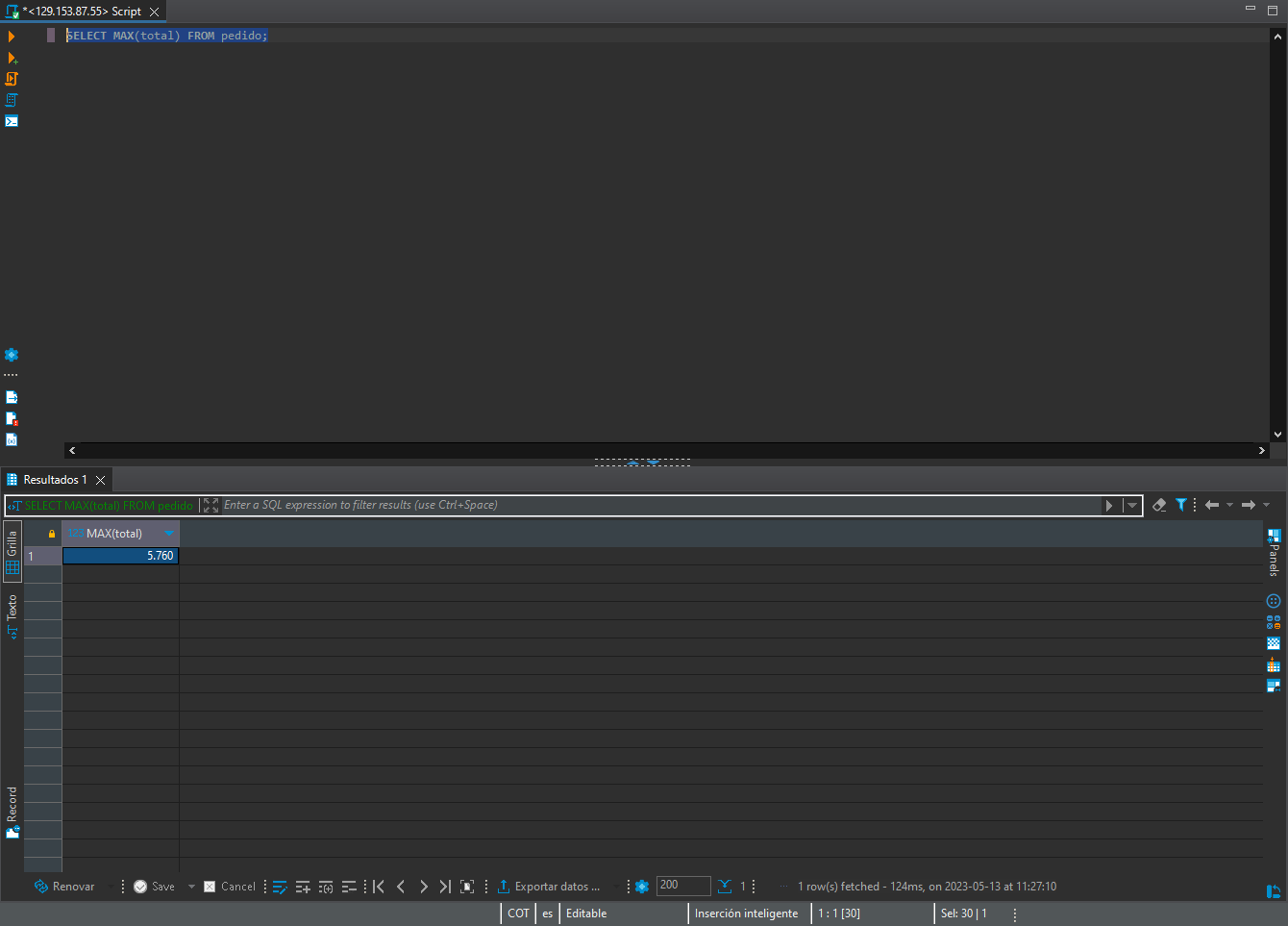
1. Calcula el número total de clientes que aparecen en la tabla cliente.

SELECT COUNT(\*) as total\_clientes FROM cliente c;



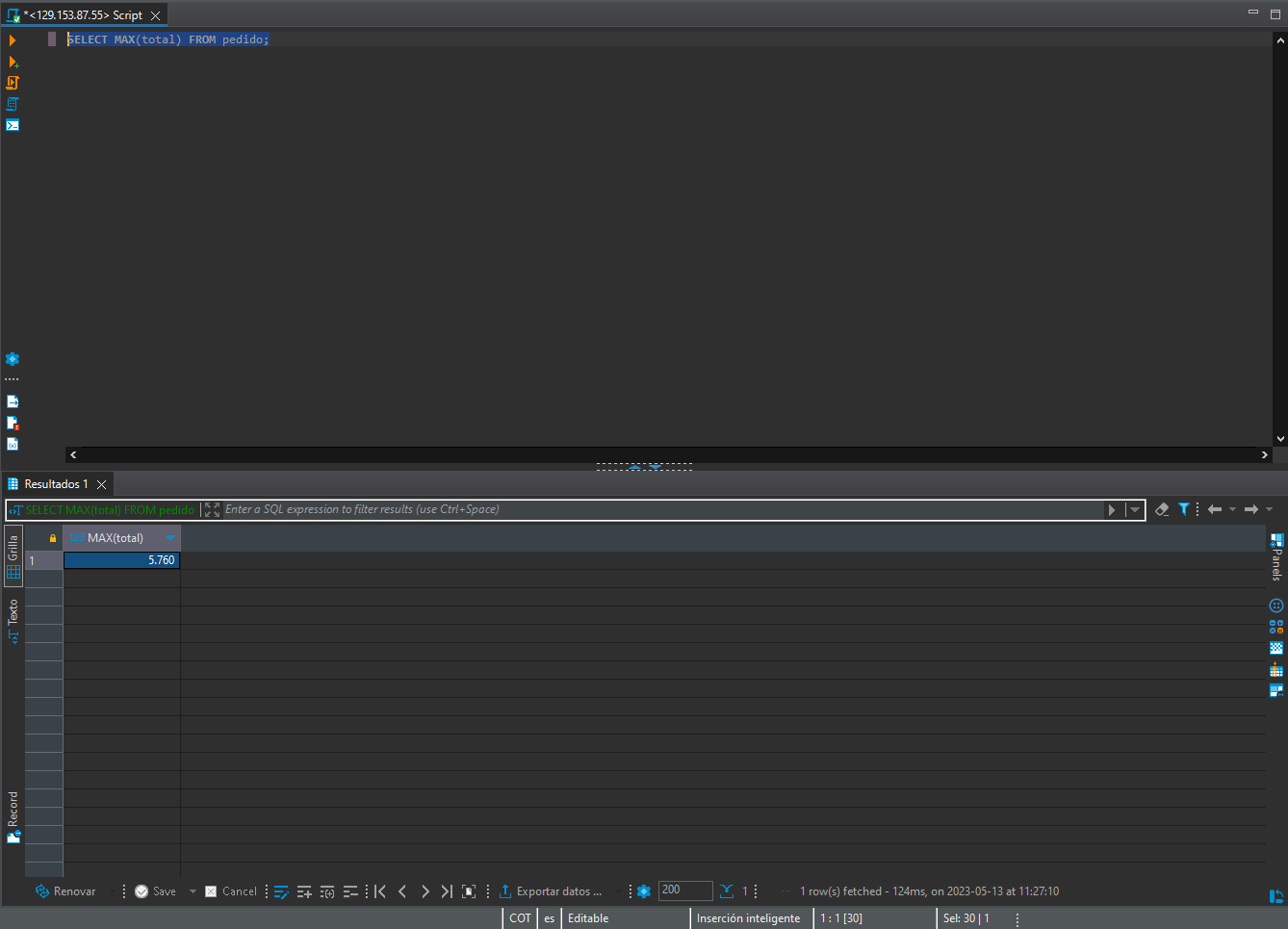
1. Calcula cuál es la mayor cantidad que aparece en la tabla pedido.

SELECT MAX(total) FROM pedido;



1. Calcula cuál es la menor cantidad que aparece en la tabla pedido.

SELECT MAX(total) FROM pedido;

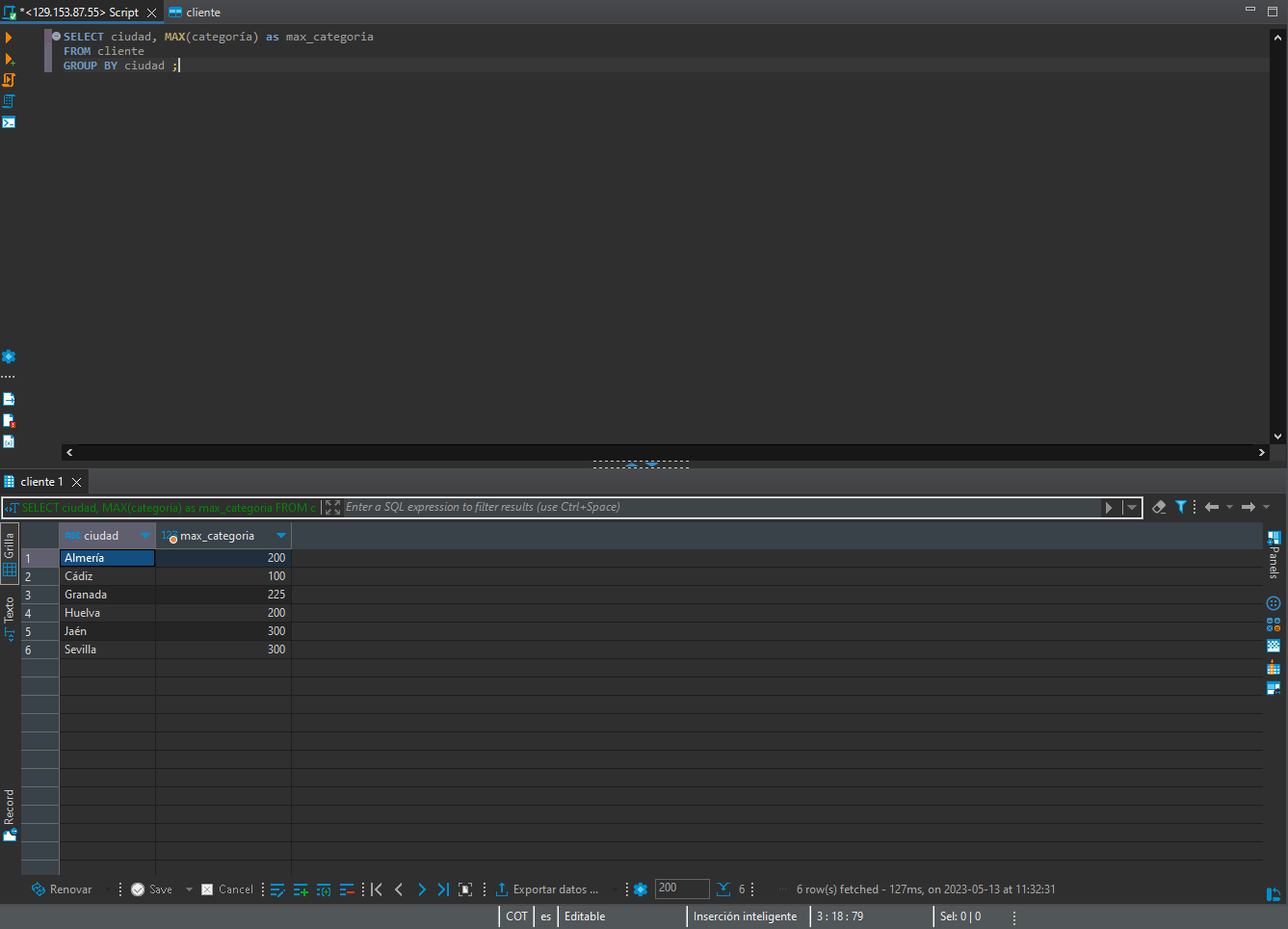


1. Calcula cuál es el valor máximo de categoría para cada una de las ciudades que aparece en la tabla cliente.

SELECT ciudad, MAX(categoría) as max\_categoria

FROM cliente

GROUP BY ciudad ;



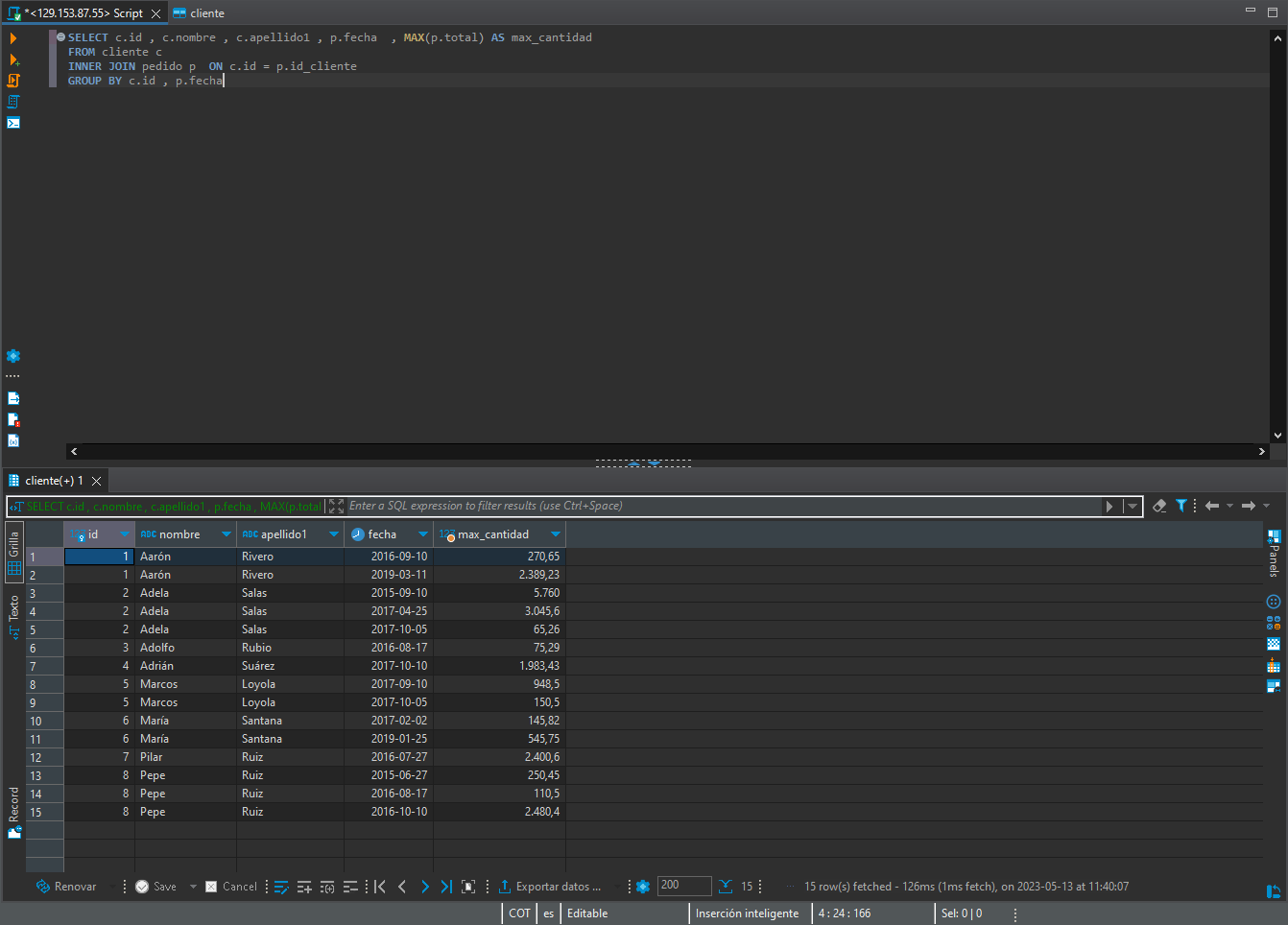
1. Calcula cuál es el máximo valor de los pedidos realizados durante el mismo día para cada uno de los clientes. Es decir, el mismo cliente puede haber realizado varios pedidos de diferentes cantidades el mismo día. Se pide que se calcule cuál es el pedido de máximo valor para cada uno de los días en los que un cliente ha realizado un pedido. Muestra el identificador del cliente, nombre, apellidos, la fecha y el valor de la cantida

SELECT c.id , c.nombre , c.apellido1 , p.fecha , MAX(p.total) AS max\_cantidad

FROM cliente c

INNER JOIN pedido p ON c.id = p.id\_cliente

GROUP BY c.id , p.fecha



1. Calcula cuál es el máximo valor de los pedidos realizados durante el mismo día para cada uno de los clientes, teniendo en cuenta que sólo queremos mostrar aquellos pedidos que superen la cantidad de 2000 €.

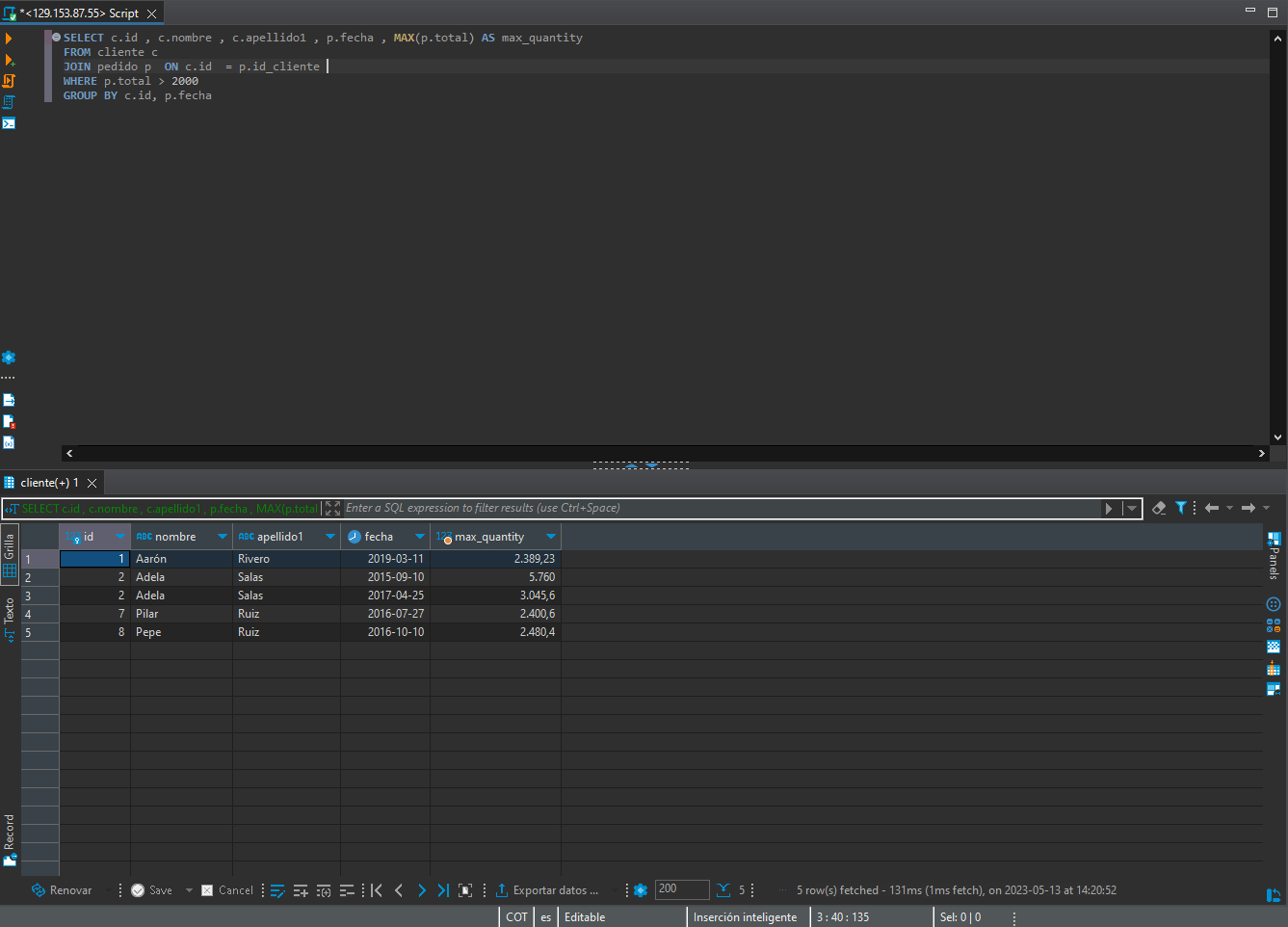
SELECT c.id , c.nombre , c.apellido1 , p.fecha , MAX(p.total) AS max\_quantity

FROM cliente c

JOIN pedido p ON c.id = p.id\_cliente

WHERE p.total > 2000

GROUP BY c.id, p.fecha



1. Calcula el máximo valor de los pedidos realizados para cada uno de los comerciales durante la fecha 2016-08-17. Muestra el identificador del comercial, nombre, apellidos y total

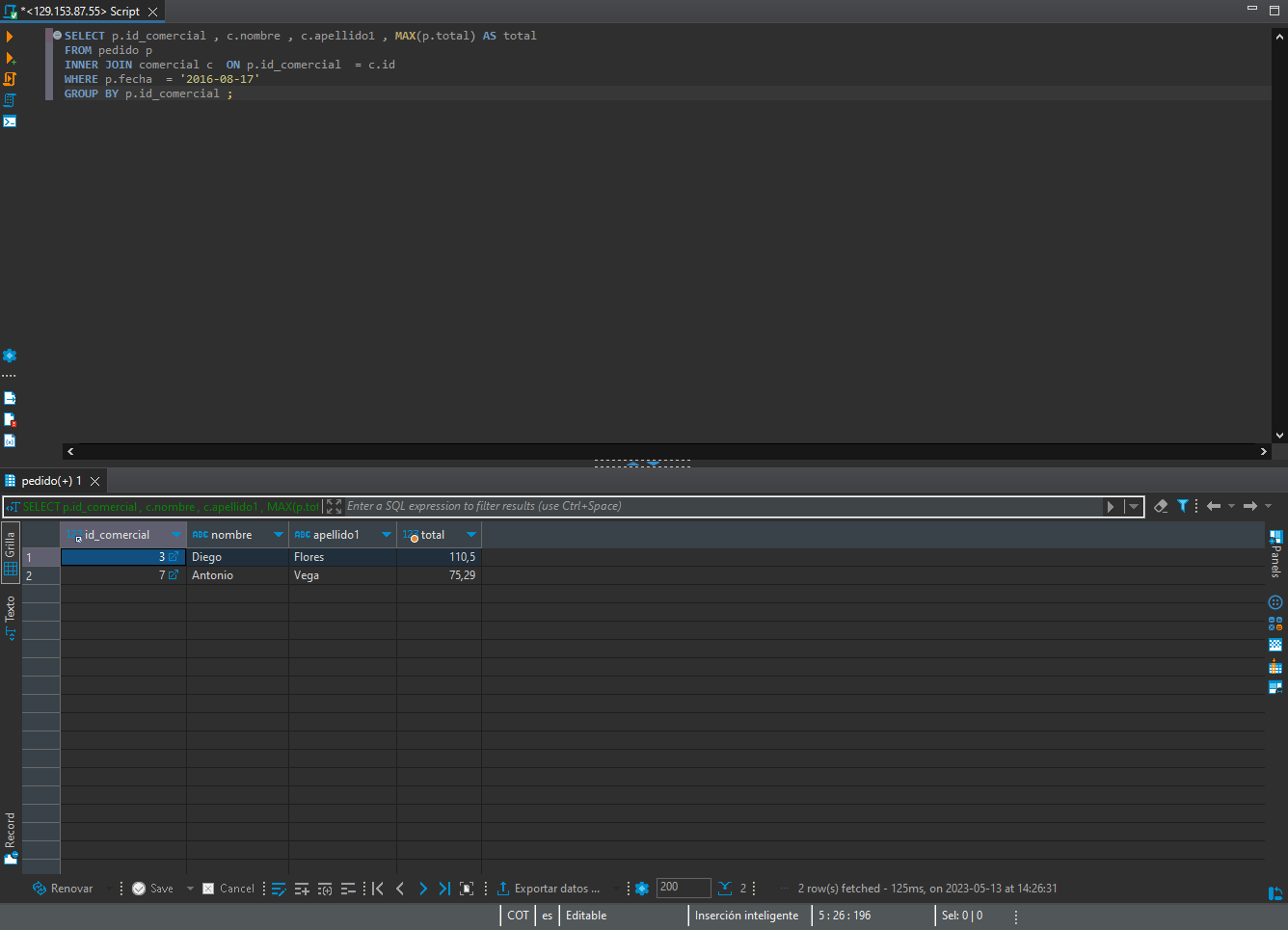
SELECT p.id\_comercial , c.nombre , c.apellido1 , MAX(p.total) AS total

FROM pedido p

INNER JOIN comercial c ON p.id\_comercial = c.id

WHERE p.fecha = '2016-08-17'

GROUP BY p.id\_comercial ;



1. Devuelve un listado con el identificador de cliente, nombre y apellidos y el número total de pedidos que ha realizado cada uno de clientes. Tenga en cuenta que pueden existir clientes que no han realizado ningún pedido. Estos clientes también deben aparecer en el listado indicando que el número de pedidos realizados es 0

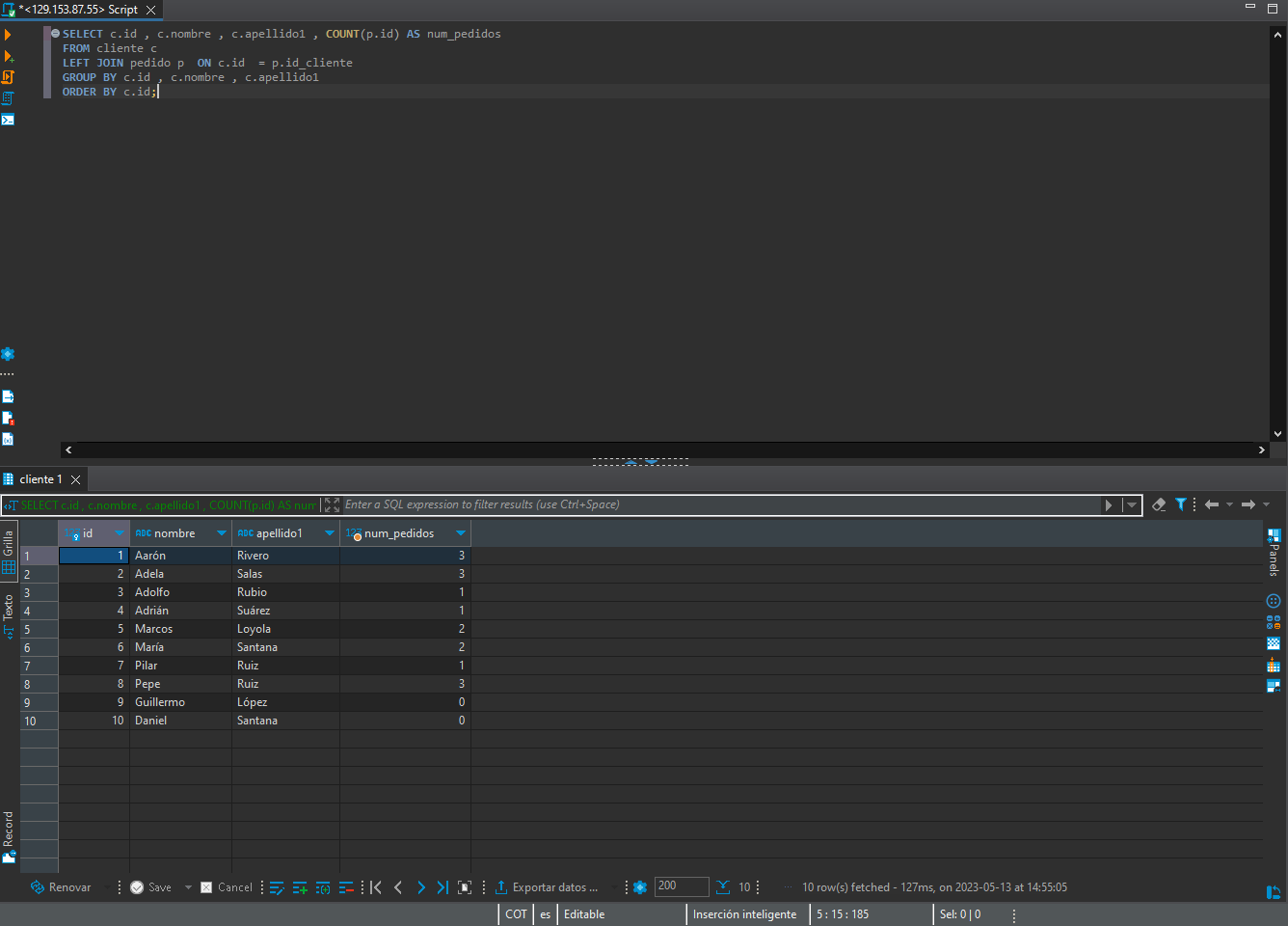
SELECT c.id , c.nombre , c.apellido1 , COUNT(p.id) AS num\_pedidos

FROM cliente c

LEFT JOIN pedido p ON c.id = p.id\_cliente

GROUP BY c.id , c.nombre , c.apellido1

ORDER BY c.id;



1. Devuelve un listado con el identificador de cliente, nombre y apellidos y el número total de pedidos que ha realizado cada uno de clientes durante el año 2017.

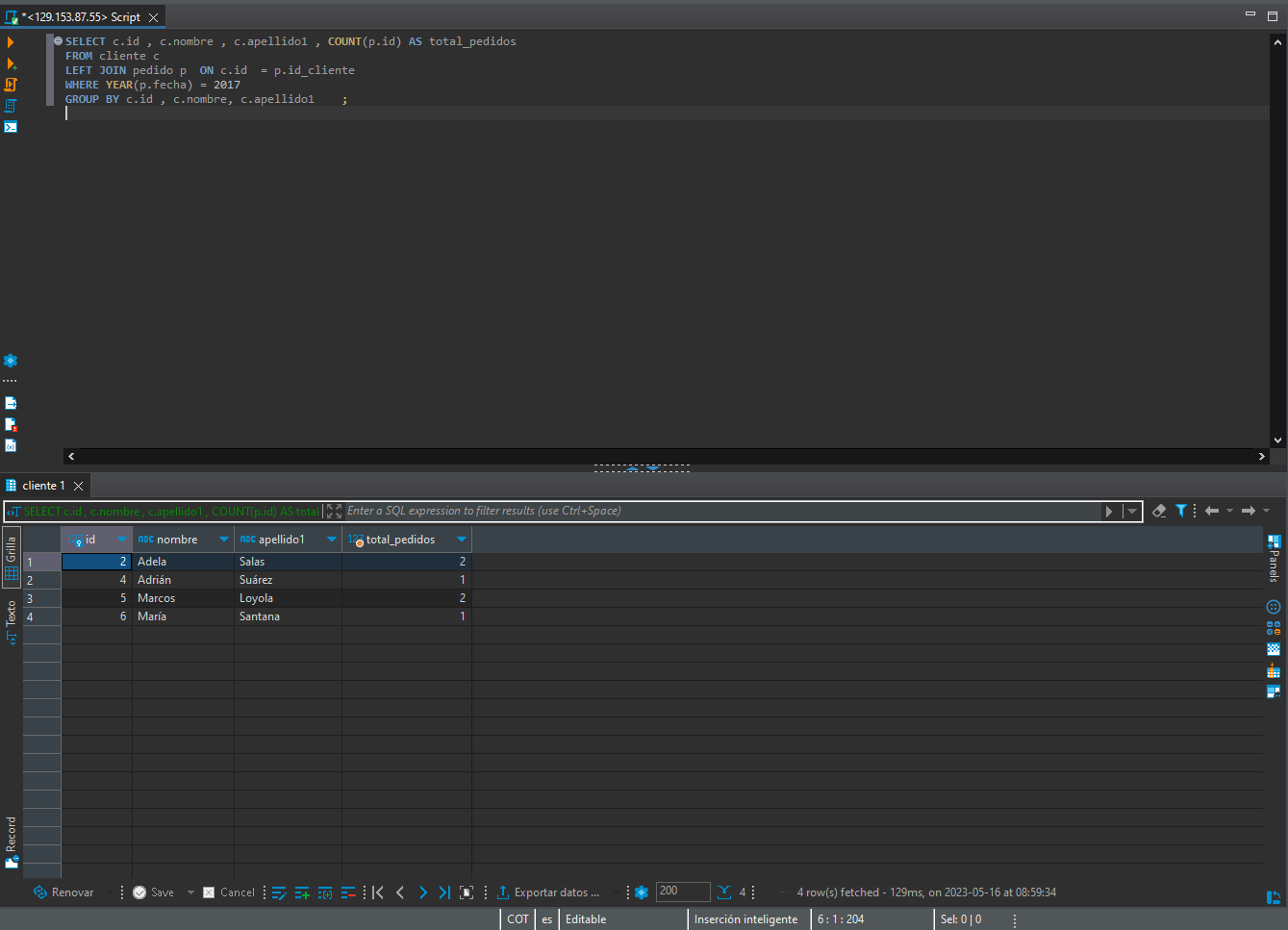
SELECT c.id , c.nombre , c.apellido1 , COUNT(p.id) AS total\_pedidos

FROM cliente c

LEFT JOIN pedido p ON c.id = p.id\_cliente

WHERE YEAR(p.fecha) = 2017

GROUP BY c.id , c.nombre, c.apellido1 ;



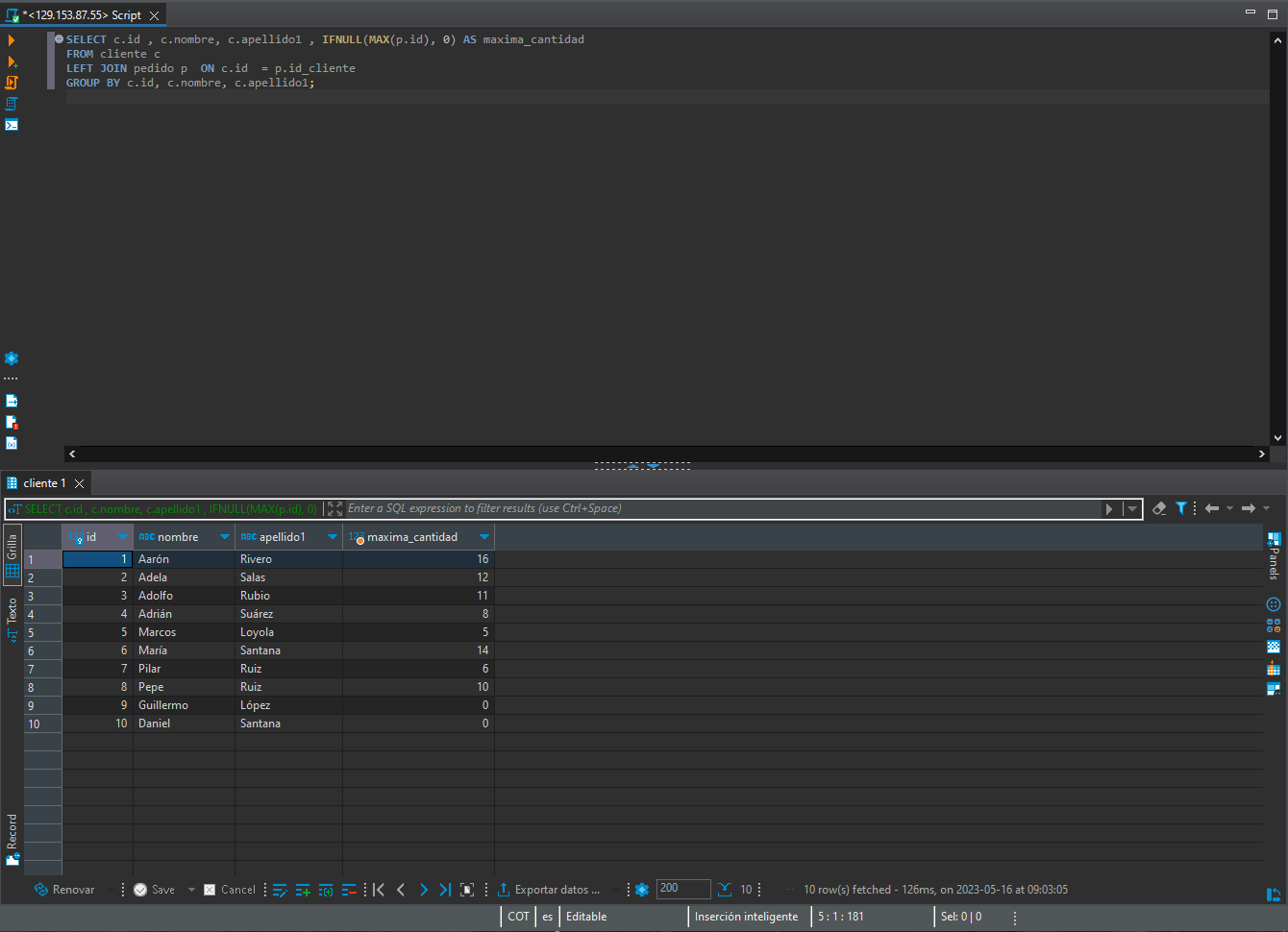
1. Devuelve un listado que muestre el identificador de cliente, nombre, primer apellido y el valor de la máxima cantidad del pedido realizado por cada uno de los clientes. El resultado debe mostrar aquellos clientes que no han realizado ningún pedido indicando que la máxima cantidad de sus pedidos realizados es 0. Puede hacer uso de la función IFNULL

SELECT c.id , c.nombre, c.apellido1 , IFNULL(MAX(p.id), 0) AS maxima\_cantidad

FROM cliente c

LEFT JOIN pedido p ON c.id = p.id\_cliente

GROUP BY c.id, c.nombre, c.apellido1;

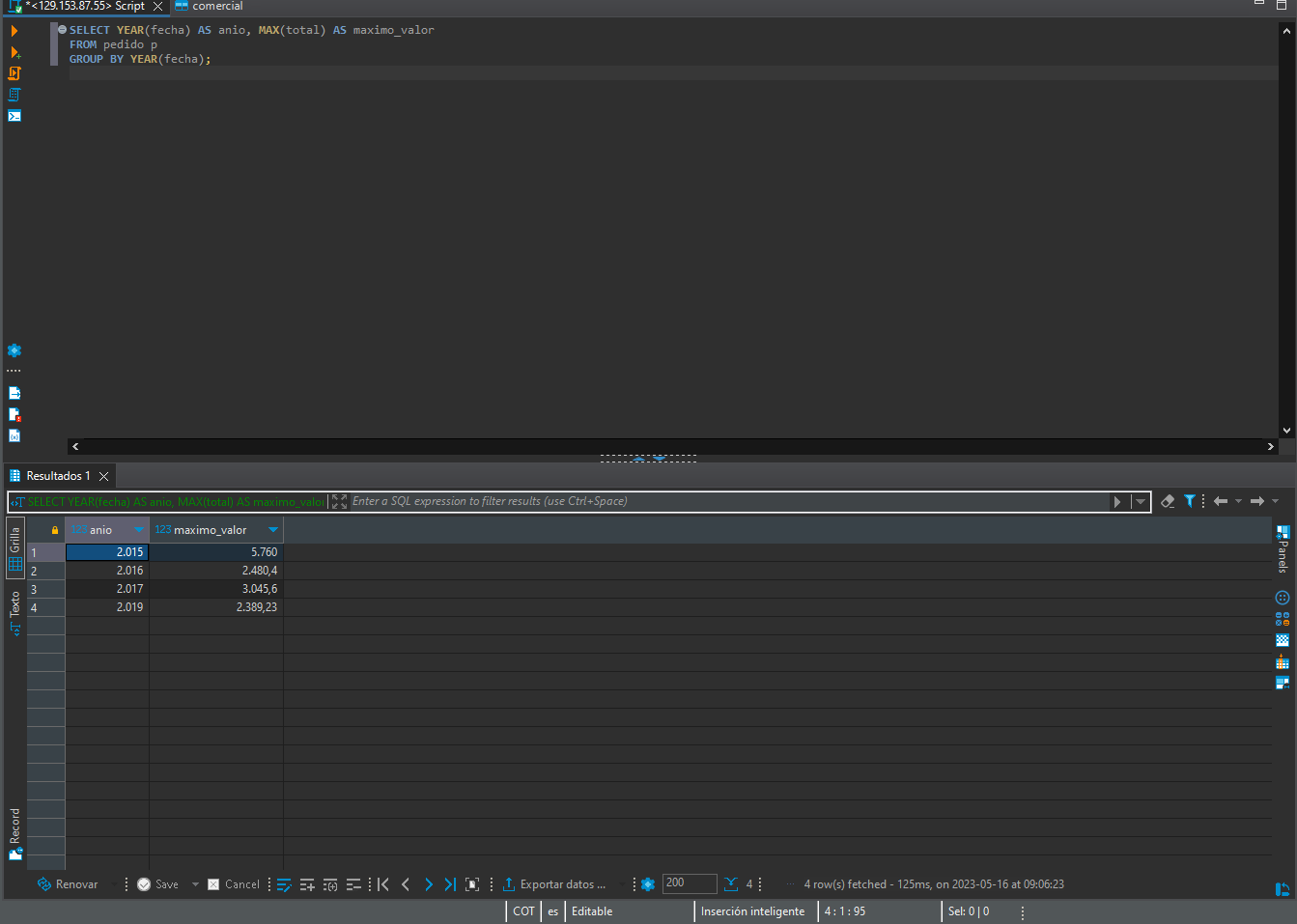


1. Devuelve cuál ha sido el pedido de máximo valor que se ha realizado cada año

SELECT YEAR(fecha) AS anio, MAX(total) AS maximo\_valor

FROM pedido p

GROUP BY YEAR(fecha);

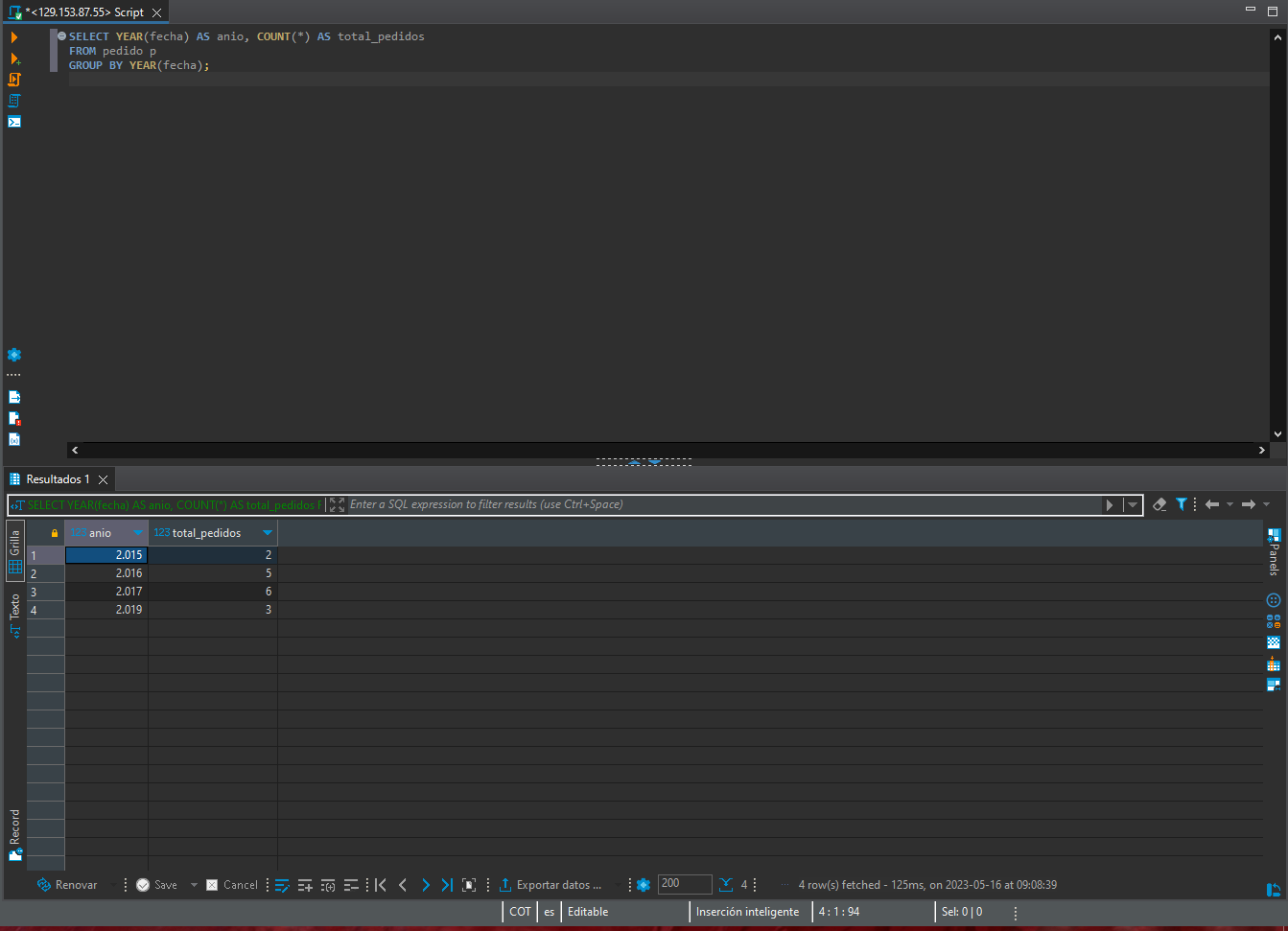


1. Devuelve el número total de pedidos que se han realizado cada año.

SELECT YEAR(fecha) AS anio, COUNT(\*) AS total\_pedidos

FROM pedido p

GROUP BY YEAR(fecha);



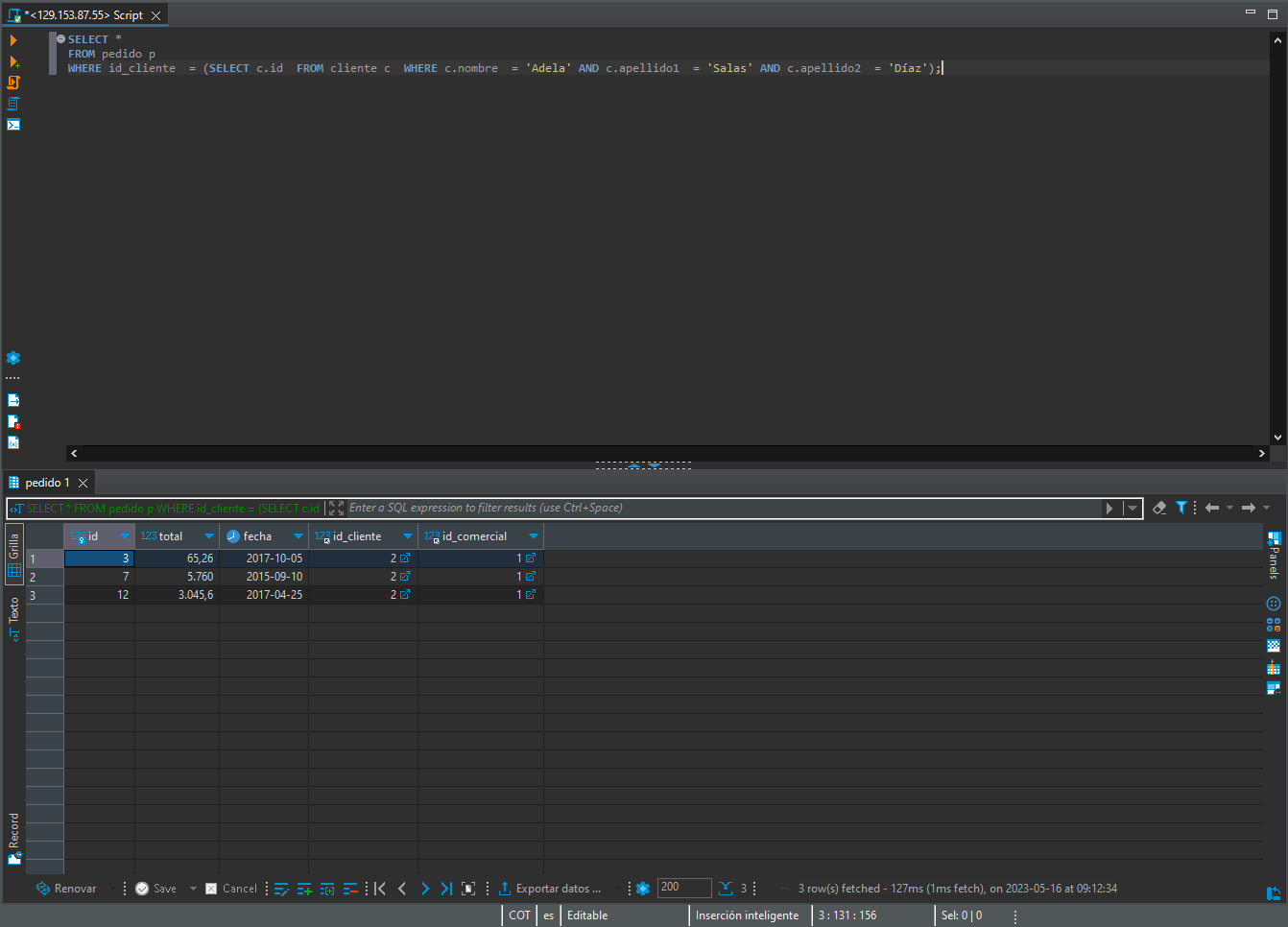
* Subconsultas

1. Devuelve un listado con todos los pedidos que ha realizado Adela Salas Díaz. (Sin utilizar INNER JOIN).

SELECT \*

FROM pedido p

WHERE id\_cliente = (SELECT c.id FROM cliente c WHERE c.nombre = 'Adela' AND c.apellido1 = 'Salas' AND c.apellido2 = 'Díaz');

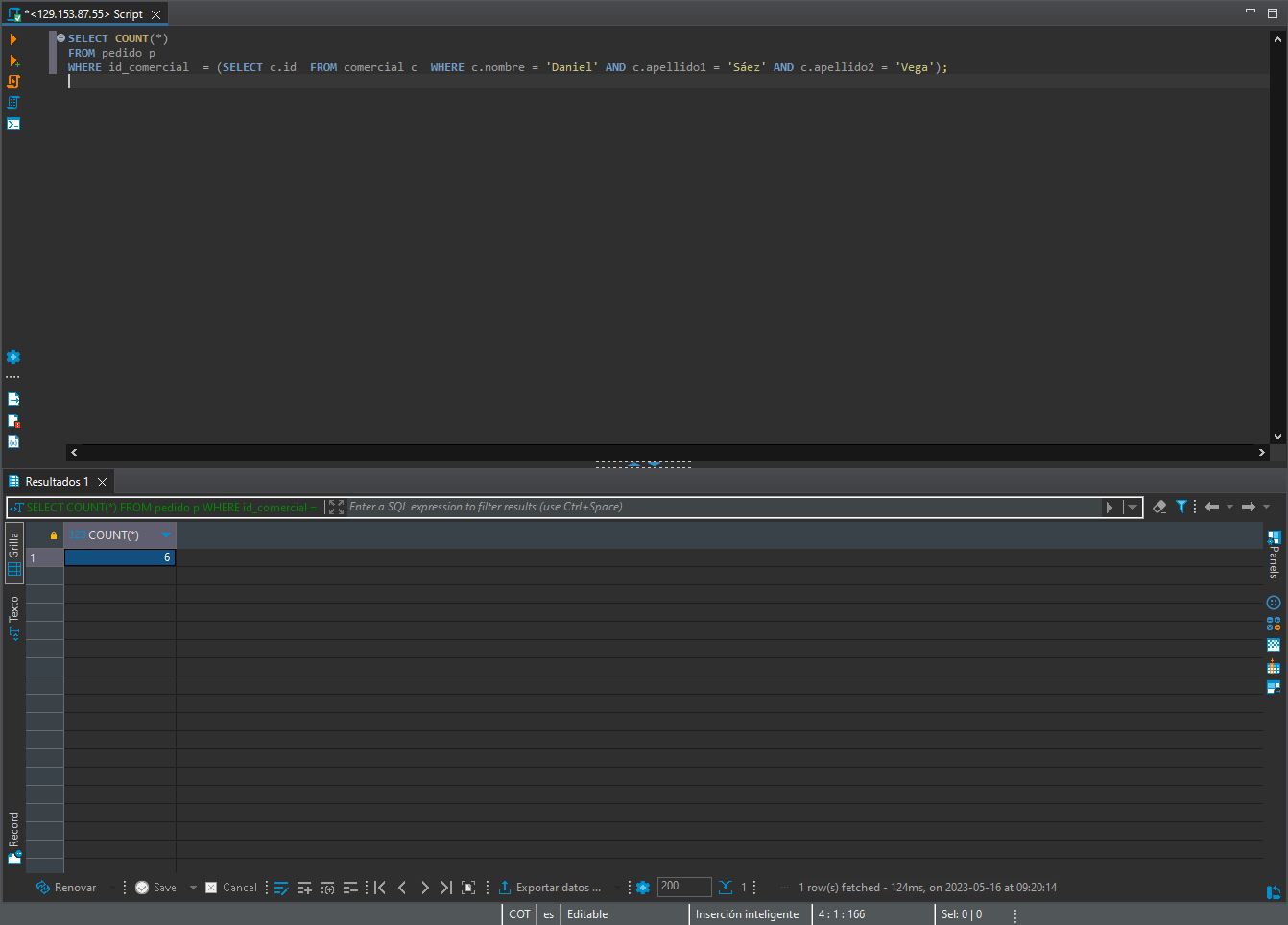


1. Devuelve el número de pedidos en los que ha participado el comercial Daniel Sáez Vega. (Sin utilizar INNER JOIN)

SELECT COUNT(\*)

FROM pedido p

WHERE id\_comercial = (SELECT c.id FROM comercial c WHERE c.nombre = 'Daniel' AND c.apellido1 = 'Sáez' AND c.apellido2 = 'Vega');



1. Devuelve los datos del cliente que realizó el pedido más caro en el año 2019. (Sin utilizar INNER JOIN

SELECT c.\*

FROM cliente c

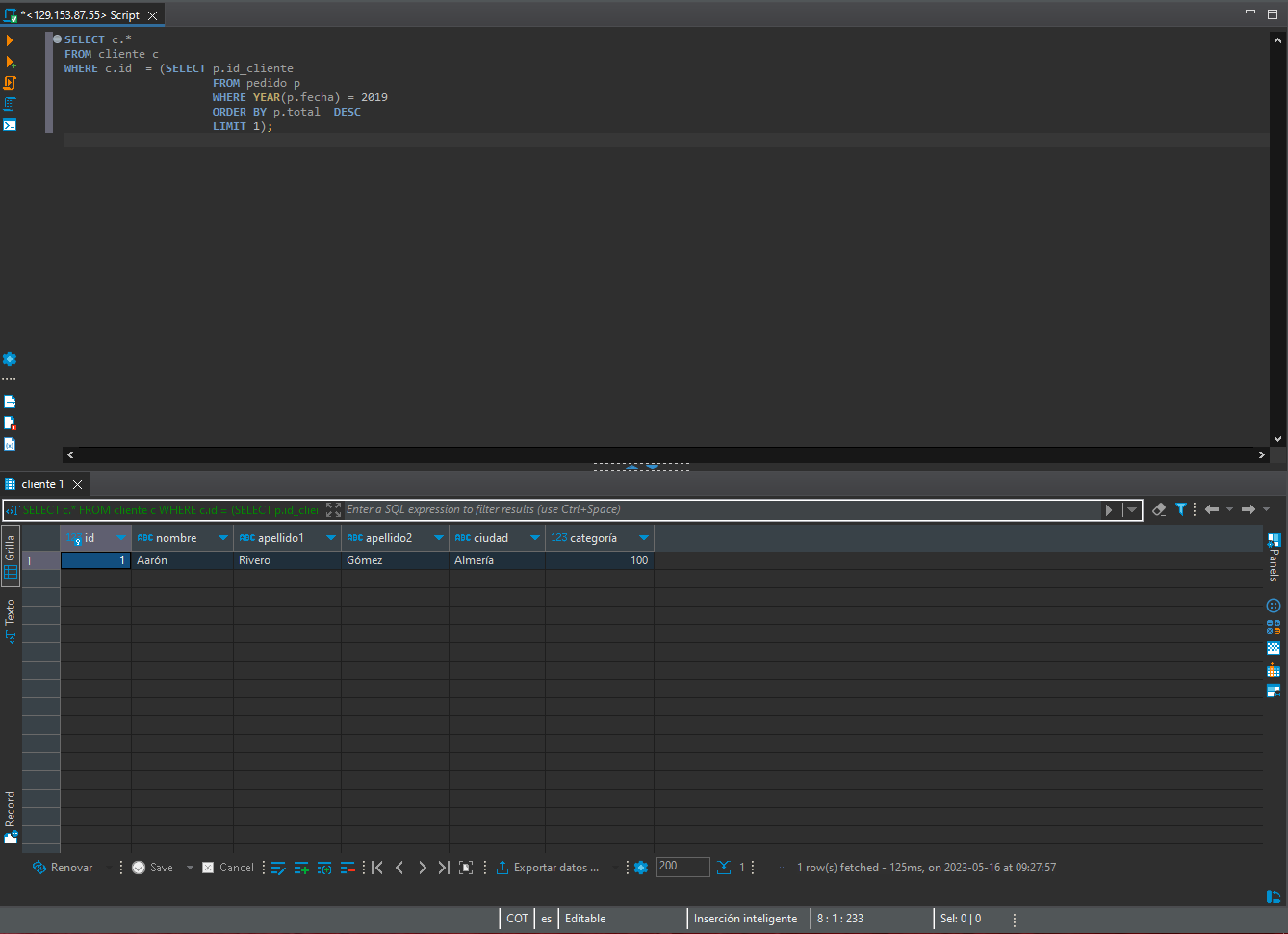
WHERE c.id = (SELECT p.id\_cliente

FROM pedido p

WHERE YEAR(p.fecha) = 2019

ORDER BY p.total DESC

LIMIT 1);



1. Devuelve la fecha y la cantidad del pedido de menor valor realizado por el cliente Pepe Ruiz Santana

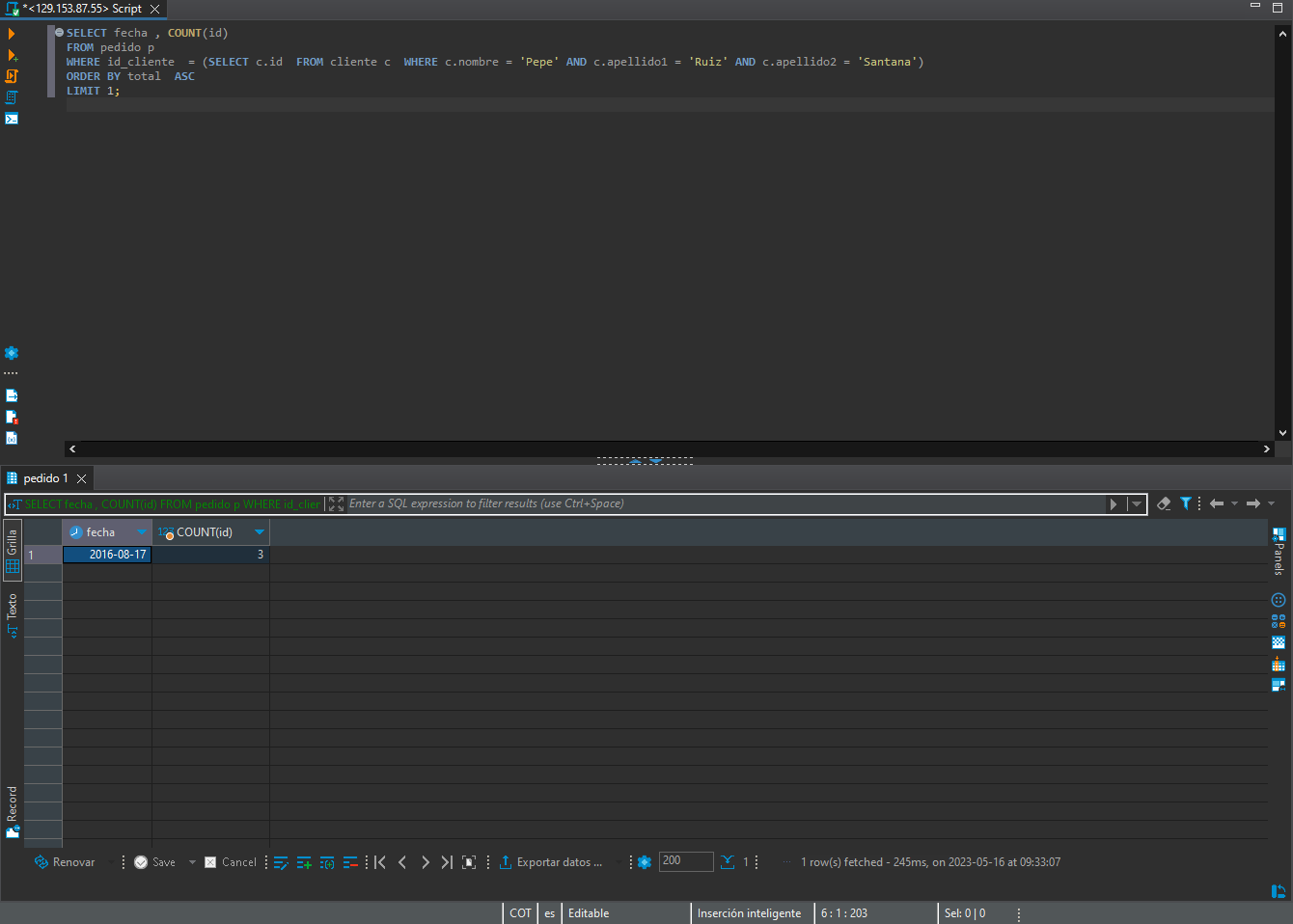
SELECT fecha , COUNT(id)

FROM pedido p

WHERE id\_cliente = (SELECT c.id FROM cliente c WHERE c.nombre = 'Pepe' AND c.apellido1 = 'Ruiz' AND c.apellido2 = 'Santana')

ORDER BY total ASC

LIMIT 1;



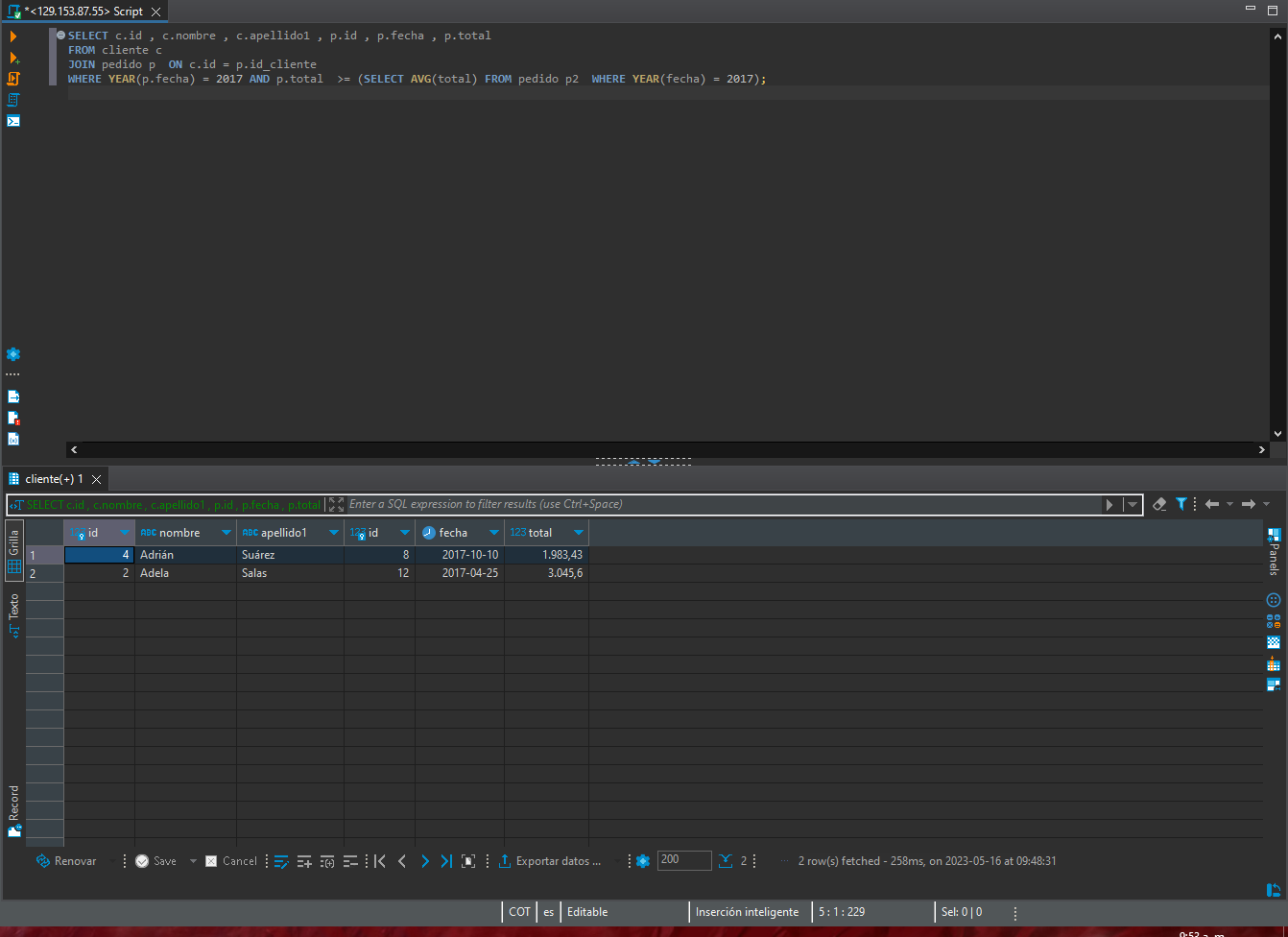
1. Devuelve un listado con los datos de los clientes y los pedidos, de todos los clientes que han realizado un pedido durante el año 2017 con un valor mayor o igual al valor medio de los pedidos realizados durante ese mismo año

SELECT c.id , c.nombre , c.apellido1 , p.id , p.fecha , p.total

FROM cliente c

JOIN pedido p ON c.id = p.id\_cliente

WHERE YEAR(p.fecha) = 2017 AND p.total >= (SELECT AVG(total) FROM pedido p2 WHERE YEAR(fecha) = 2017);

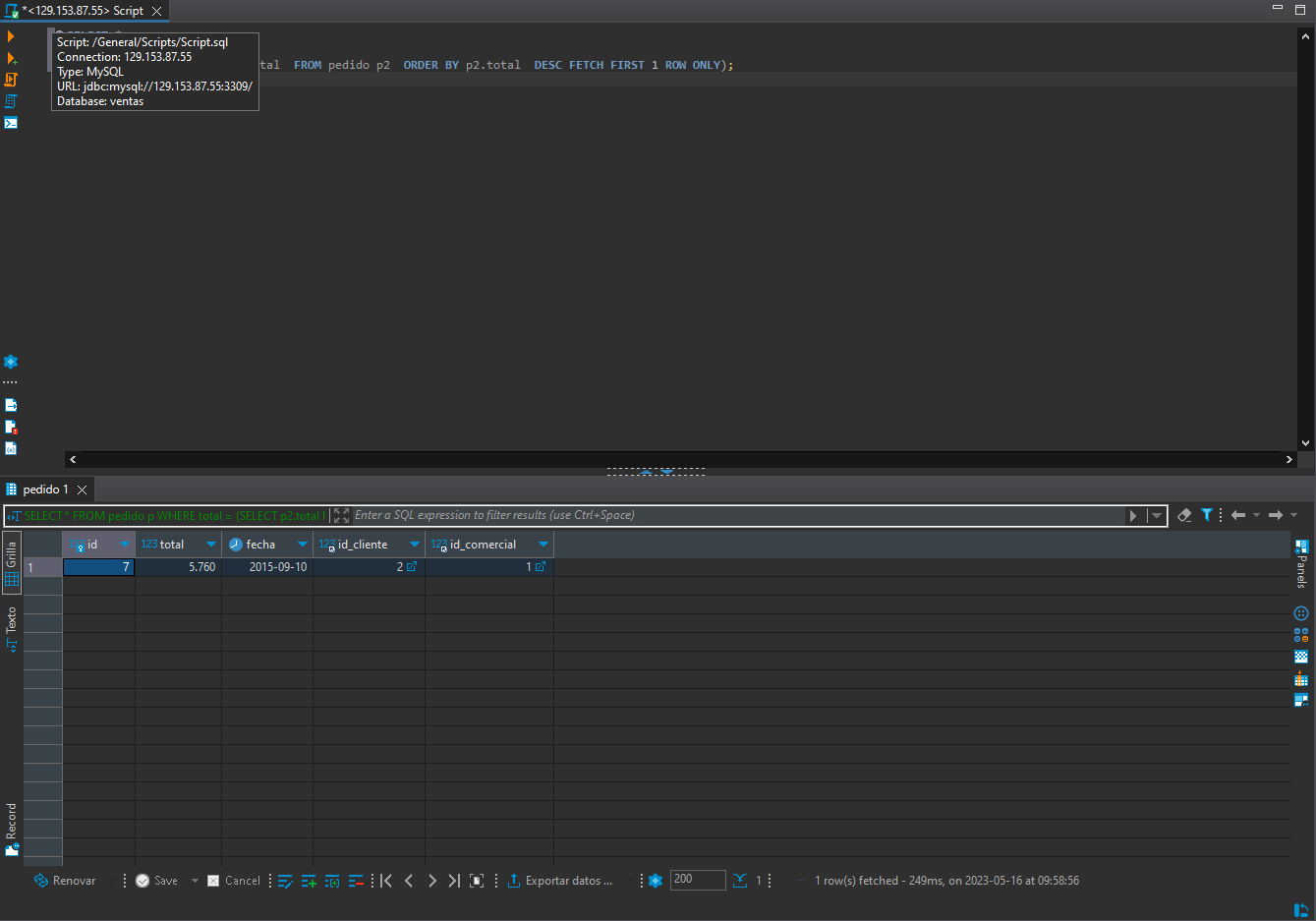


1. Devuelve el pedido más caro que existe en la tabla pedido si hacer uso de MAX, ORDER BY ni LIMIT

SELECT \*

FROM pedido p

WHERE total = (SELECT p2.total FROM pedido p2 ORDER BY p2.total DESC FETCH FIRST 1 ROW ONLY);



Calcula cuál es el máximo valor de los pedidos realizados durante el mismo día para cada uno de los

clientes, teniendo en cuenta que sólo queremos mostrar aquellos pedidos que superen la cantidad de

2000 €.

SELECT p.id\_cliente, c.nombre, c.apellido1, c.apellido2, p.fecha, MAX(p.total) AS

max\_valor

FROM pedido p

JOIN cliente c ON p.id\_cliente = c.id

WHERE (p.fecha, p.id\_cliente, p.total) IN (

SELECT fecha, id\_cliente, MAX(total)

FROM pedido

WHERE total > 2000

GROUP BY fecha, id\_cliente

)

GROUP BY p.fecha, p.id\_cliente;

10. Calcula el máximo valor de los pedidos realizados para cada uno de los comerciales durante la fecha 2016-

08-17. Muestra el identificador del comercial, nombre, apellidos y total.

SELECT p.id\_comercial, c.nombre, c.apellido1, c.apellido2, MAX(p.total) AS

max\_valor

FROM pedido p

JOIN comercial c ON p.id\_comercial = c.id

WHERE p.fecha = '2016-08-17'

GROUP BY p.id\_comercial;

11. Devuelve un listado con el identificador de cliente, nombre y apellidos y el número total de pedidos que

ha realizado cada uno de clientes. Tenga en cuenta que pueden existir clientes que no han realizado ningún

pedido. Estos clientes también deben aparecer en el listado indicando que el número de pedidos

realizados es 0.

SELECT c.id, c.nombre, c.apellido1, c.apellido2, COUNT(p.id) AS num\_pedidos

FROM cliente c

LEFT JOIN pedido p ON c.id = p.id\_cliente

GROUP BY c.id, c.nombre, c.apellido1, c.apellido2;

12. Devuelve un listado con el identificador de cliente, nombre y apellidos y el número total de pedidos que

ha realizado cada uno de clientes durante el año 2017.

SELECT c.id, c.nombre, c.apellido1, c.apellido2, COUNT(p.id) AS num\_pedidos

FROM cliente c

LEFT JOIN pedido p ON c.id = p.id\_cliente

WHERE YEAR(p.fecha) = 2017

GROUP BY c.id, c.nombre, c.apellido1, c.apellido2;

13. Devuelve un listado que muestre el identificador de cliente, nombre, primer apellido y el valor de la

máxima cantidad del pedido realizado por cada uno de los clientes. El resultado debe mostrar aquellos

clientes que no han realizado ningún pedido indicando que la máxima cantidad de sus pedidos realizados

es 0. Puede hacer uso de la función IFNULL.

SELECT c.id, c.nombre, c.apellido1, IFNULL(MAX(p.total), 0) AS max\_cantidad

FROM cliente c

LEFT JOIN pedido p ON c.id = p.id\_cliente

GROUP BY c.id, c.nombre, c.apellido1;

14. Devuelve cuál ha sido el pedido de máximo valor que se ha realizado cada año.

SELECT YEAR(fecha) AS año, MAX(total) AS maximo

FROM pedido

GROUP BY YEAR(fecha);

15. Devuelve el número total de pedidos que se han realizado cada año.

SELECT YEAR(fecha) AS año, COUNT(\*) AS total\_pedidos

FROM pedido

GROUP BY YEAR(fecha);

Subconsultas

1. Devuelve un listado con todos los pedidos que ha realizado Adela Salas Díaz. (Sin utilizar INNER JOIN).

SELECT \*

FROM pedido

WHERE id\_cliente = (

SELECT id

FROM cliente

WHERE nombre = 'Adela' AND apellido1 = 'Salas' AND apellido2 = 'Díaz'

);

2. Devuelve el número de pedidos en los que ha participado el comercial Daniel Sáez Vega. (Sin utilizar INNER

JOIN)

SELECT COUNT(\*)

FROM pedido

WHERE id\_comercial = (

SELECT id

FROM comercial

WHERE nombre = 'Daniel'

AND apellido1 = 'Sáez'

AND apellido2 = 'Vega'

)

3. Devuelve los datos del cliente que realizó el pedido más caro en el año 2019. (Sin utilizar INNER JOIN)

SELECT id, nombre, apellido1, apellido2

FROM cliente

WHERE id = (

SELECT id\_cliente

FROM pedido

WHERE total = (

SELECT MAX(total)

FROM pedido

WHERE YEAR(fecha) = 2019

)

)

4. Devuelve la fecha y la cantidad del pedido de menor valor realizado por el cliente Pepe Ruiz Santana.

SELECT fecha, total

FROM pedido

WHERE id\_cliente = (

SELECT id

FROM cliente

WHERE nombre = 'Pepe'

AND apellido1 = 'Ruiz'

AND apellido2 = 'Santana'

)

ORDER BY total ASC

LIMIT 1;

5. Devuelve un listado con los datos de los clientes y los pedidos, de todos los clientes que han realizado un

pedido durante el año 2017 con un valor mayor o igual al valor medio de los pedidos realizados durante

ese mismo año.

SELECT c.id, c.nombre, c.apellido1, c.apellido2, p.id AS id\_pedido, p.total,

p.fecha

FROM cliente c, pedido p

WHERE c.id = p.id\_cliente

AND YEAR(p.fecha) = 2017

AND p.total >= (SELECT AVG(total) FROM pedido WHERE YEAR(fecha) = 2017)

ORDER BY c.id, p.fecha;

6. Devuelve el pedido más caro que existe en la tabla pedido si hacer uso de MAX, ORDER BY ni LIMIT.

SELECT id, total

FROM pedido

WHERE total >= ALL (SELECT total FROM pedido);

7. Devuelve un listado de los clientes que no han realizado ningún pedido. (Utilizando ANY o ALL).

SELECT id, nombre, apellido1, apellido2

FROM cliente

WHERE id <> ALL (

SELECT DISTINCT id\_cliente

FROM pedido

)

8. Devuelve un listado de los comerciales que no han realizado ningún pedido. (Utilizando ANY o ALL).

SELECT \*

FROM comercial

WHERE NOT EXISTS (

SELECT 1

FROM pedido

WHERE pedido.id\_comercial = comercial.id

);

9. Devuelve un listado de los clientes que no han realizado ningún pedido. (Utilizando IN o NOT

IN).

SELECT id, nombre, apellido1, apellido2

FROM cliente

WHERE id NOT IN (SELECT id\_cliente FROM pedido);

10. Devuelve un listado de los comerciales que no han realizado ningún pedido.

(Utilizando IN o NOT IN).

SELECT id, nombre, apellido1, apellido2

FROM comercial

WHERE id NOT IN (SELECT DISTINCT id\_comercial FROM pedido);

11. Devuelve un listado de los clientes que no han realizado ningún

pedido. (Utilizando EXISTS o NOT EXISTS).

SELECT id, nombre, apellido1, apellido2

FROM cliente

WHERE NOT EXISTS (

SELECT \*

FROM pedido

WHERE cliente.id = pedido.id\_cliente

);

12. Devuelve un listado de los comerciales que no han realizado ningún pedido. (Utilizando EXISTS o NOT

EXISTS).

SELECT id, nombre, apellido1, apellido2, comisión

FROM comercial c

WHERE NOT EXISTS (

SELECT 1

FROM pedido p

WHERE p.id\_comercial = c.id

);

GESTION DE EMPLEADOS

Consultas sobre una tabla

1. Lista el primer apellido de todos los empleados.

SELECT apellido1 FROM empleado;

2. Lista el primer apellido de los empleados eliminando los apellidos que estén repetidos.

SELECT DISTINCT apellido1 FROM empleado;

3. Lista todas las columnas de la tabla empleado.

SELECT \* FROM empleado;

4. Lista el nombre y los apellidos de todos los empleados.

SELECT nombre, apellido1, apellido2 FROM empleado;

5. Lista el identificador de los departamentos de los empleados que aparecen en la tabla empleado.

SELECT id\_departamento FROM empleado;

6. Lista el identificador de los departamentos de los empleados que aparecen en la tabla empleado,

eliminando los identificadores que aparecen repetidos.

SELECT DISTINCT id\_departamento FROM empleado;

7. Lista el nombre y apellidos de los empleados en una única columna.

SELECT CONCAT(nombre, ' ', apellido1, ' ', apellido2) AS

nombre\_completo

FROM empleado;

8. Lista el nombre y apellidos de los empleados en una única columna, convirtiendo todos los caracteres en

mayúscula.

SELECT UPPER(CONCAT(nombre, ' ', apellido1, ' ', apellido2)) AS

nombre\_completo

FROM empleado;

9. Lista el nombre y apellidos de los empleados en una única columna, convirtiendo todos los caracteres en

minúscula.

SELECT LOWER(CONCAT(nombre, ' ', apellido1, ' ', apellido2)) AS

nombre\_completo

FROM empleado;

10. Lista el identificador de los empleados junto al nif, pero el nif deberá aparecer en dos columnas, una

mostrará únicamente los dígitos del nif y la otra la letra.

SELECT id, SUBSTR(nif, 1, 8) AS nif\_digitos, SUBSTR(nif, 9, 1) AS

nif\_letra

FROM empleado;

11. Lista el nombre de cada departamento y el valor del presupuesto actual del que dispone. Para calcular este

dato tendrá que restar al valor del presupuesto inicial (columna presupuesto) los gastos que se han

generado (columna gastos). Tenga en cuenta que en algunos casos pueden existir valores negativos. Utilice

un alias apropiado para la nueva columna columna que está calculando.

SELECT nombre, (presupuesto - gastos) AS presupuesto\_actual

FROM departamento;

12. Lista el nombre de los departamentos y el valor del presupuesto actual ordenado de forma ascendente.

SELECT nombre, (presupuesto - gastos) AS presupuesto\_actual

FROM departamento

ORDER BY presupuesto\_actual ASC;

13. Lista el nombre de todos los departamentos ordenados de forma ascendente.

SELECT nombre

FROM departamento

ORDER BY nombre ASC;

14. Lista el nombre de todos los departamentos ordenados de forma desscendente.

SELECT nombre

FROM departamento

ORDER BY nombre DESC;

15. Lista los apellidos y el nombre de todos los empleados, ordenados de forma alfabética teniendo en cuenta

en primer lugar sus apellidos y luego su nombre.

SELECT apellido1, apellido2, nombre

FROM empleado

ORDER BY apellido1 ASC, apellido2 ASC, nombre ASC;

16. Devuelve una lista con el nombre y el presupuesto, de los 3 departamentos que tienen mayor presupuesto.

SELECT nombre, presupuesto AS presupuesto\_actual

FROM departamento

ORDER BY presupuesto DESC

LIMIT 3;

17. Devuelve una lista con el nombre y el presupuesto, de los 3 departamentos que tienen menor

presupuesto.

SELECT nombre, presupuesto AS presupuesto\_actual

FROM departamento

ORDER BY presupuesto ASC

LIMIT 3;

18. Devuelve una lista con el nombre y el gasto, de los 2 departamentos que tienen mayor gasto.

SELECT nombre, gastos

FROM departamento

ORDER BY gastos DESC

LIMIT 2;

19. Devuelve una lista con el nombre y el gasto, de los 2 departamentos que tienen menor gasto.

SELECT nombre, gastos

FROM departamento

ORDER BY gastos ASC

LIMIT 2;

20. Devuelve una lista con 5 filas a partir de la tercera fila de la tabla empleado. La tercera fila se debe incluir

en la respuesta. La respuesta debe incluir todas las columnas de la tabla empleado.

SELECT \*

FROM empleado

LIMIT 2, 5;

21. Devuelve una lista con el nombre de los departamentos y el presupuesto, de aquellos que tienen un

presupuesto mayor o igual a 150000 euros.

SELECT nombre, presupuesto

FROM departamento

WHERE presupuesto >= 150000;

22. Devuelve una lista con el nombre de los departamentos y el gasto, de aquellos que tienen menos de 5000

euros de gastos.

SELECT nombre, gastos

FROM departamento

WHERE gastos < 5000;

23. Devuelve una lista con el nombre de los departamentos y el presupesto, de aquellos que tienen un

presupuesto entre 100000 y 200000 euros. Sin utilizar el operador BETWEEN.

SELECT nombre, presupuesto

FROM departamento

WHERE presupuesto > 100000 AND presupuesto < 200000;

24. Devuelve una lista con el nombre de los departamentos que no tienen un presupuesto entre 100000 y

200000 euros. Sin utilizar el operador BETWEEN.

SELECT nombre, presupuesto

FROM departamento

WHERE presupuesto < 100000 OR presupuesto > 200000;

25. Devuelve una lista con el nombre de los departamentos que tienen un presupuesto entre 100000 y 200000

euros. Utilizando el operador BETWEEN.

SELECT nombre FROM departamento WHERE presupuesto BETWEEN 100000 AND

200000;

26. Devuelve una lista con el nombre de los departamentos que no tienen un presupuesto entre 100000 y

200000 euros. Utilizando el operador BETWEEN.

SELECT nombre, presupuesto

FROM departamento

WHERE presupuesto NOT BETWEEN 100000 AND 200000;

27. Devuelve una lista con el nombre de los departamentos, gastos y presupuesto, de aquellos departamentos

donde los gastos sean mayores que el presupuesto del que disponen.

SELECT nombre, presupuesto, gastos

FROM departamento

WHERE gastos > presupuesto;

28. Devuelve una lista con el nombre de los departamentos, gastos y presupuesto, de aquellos departamentos

donde los gastos sean menores que el presupuesto del que disponen.

SELECT nombre, presupuesto, gastos

FROM departamento

WHERE gastos < presupuesto;

29. Devuelve una lista con el nombre de los departamentos, gastos y presupuesto, de aquellos departamentos

donde los gastos sean iguales al presupuesto del que disponen.

SELECT nombre, presupuesto, gastos

FROM departamento

WHERE presupuesto = gastos;

30. Lista todos los datos de los empleados cuyo segundo apellido sea NULL.

SELECT \* FROM empleado WHERE apellido2 IS NULL;

31. Lista todos los datos de los empleados cuyo segundo apellido no sea NULL.

SELECT \*

FROM empleado

WHERE apellido2 IS NOT NULL;

32. Lista todos los datos de los empleados cuyo segundo apellido sea López.

SELECT \* FROM empleado WHERE apellido2 = 'López';

33. Lista todos los datos de los empleados cuyo segundo apellido sea Díaz o Moreno. Sin utilizar el operador

IN.

SELECT \*

FROM empleado

WHERE apellido2 = 'Díaz' OR apellido2 = 'Moreno';

34. Lista todos los datos de los empleados cuyo segundo apellido sea Díaz o Moreno. Utilizando el operador

IN.

SELECT \* FROM empleado WHERE apellido2 IN ('Díaz', 'Moreno');

35. Lista los nombres, apellidos y nif de los empleados que trabajan en el departamento 3.

SELECT nombre, apellido1, apellido2, nif

FROM empleado

WHERE id\_departamento = 3;

36. Lista los nombres, apellidos y nif de los empleados que trabajan en los departamentos 2, 4 o 5.

SELECT nombre, apellido1, apellido2, nif

FROM empleado

WHERE id\_departamento IN (2, 4, 5);

• Consultas multitabla

1. Devuelve un listado con los empleados y los datos de los departamentos donde trabaja cada uno.

SELECT e.\*, d.\*

FROM empleado e

JOIN departamento d ON e.id\_departamento = d.id;

2. Devuelve un listado con los empleados y los datos de los departamentos donde trabaja cada uno. Ordena el

resultado, en primer lugar por el nombre del departamento (en orden alfabético) y en segundo lugar por los

apellidos y el nombre de los empleados.

SELECT e.nombre, e.apellido1, e.apellido2, d.nombre,

d.presupuesto, d.gastos

FROM empleado e

JOIN departamento d ON e.id\_departamento = d.id

ORDER BY d.nombre, e.apellido1, e.apellido2, e.nombre;

3. Devuelve un listado con el identificador y el nombre del departamento, solamente de aquellos departamentos que

tienen empleados.

SELECT DISTINCT d.id, d.nombre

FROM departamento d

INNER JOIN empleado e ON e.id\_departamento = d.id;

4. Devuelve un listado con el identificador, el nombre del departamento y el valor del presupuesto actual del que

dispone, solamente de aquellos departamentos que tienen empleados. El valor del presupuesto actual lo puede

calcular restando al valor del presupuesto inicial (columna presupuesto) el valor de los gastos que ha generado

(columna gastos).

SELECT d.id, d.nombre, d.presupuesto - COALESCE((SELECT SUM(d.gastos)

FROM empleado e WHERE e.id\_departamento = d.id), 0) as

presupuesto\_actual

FROM departamento d

WHERE EXISTS (SELECT 1 FROM empleado e WHERE e.id\_departamento = d.id)

5. Devuelve el nombre del departamento donde trabaja el empleado que tiene el nif 38382980M.

SELECT d.nombre

FROM empleado e

INNER JOIN departamento d ON e.id\_departamento = d.id

WHERE e.nif = '38382980M';

6. Devuelve el nombre del departamento donde trabaja el empleado Pepe Ruiz Santana.

SELECT d.nombre

FROM empleado e

INNER JOIN departamento d ON e.id\_departamento = d.id

WHERE e.nombre = 'Pepe' AND e.apellido1 = 'Ruiz' AND e.apellido2 =

'Santana';

7. Devuelve un listado con los datos de los empleados que trabajan en el departamento de I+D. Ordena el resultado

alfabéticamente.

SELECT \*

FROM empleado e

INNER JOIN departamento d ON e.id\_departamento = d.id

WHERE d.nombre = 'I+D'

ORDER BY e.apellido1, e.apellido2, e.nombre;

8. Devuelve un listado con los datos de los empleados que trabajan en el departamento de Sistemas, Contabilidad o

I+D. Ordena el resultado alfabéticamente.

SELECT \*

FROM empleado e

JOIN departamento d ON e.id\_departamento = d.id

WHERE d.nombre IN ('Sistemas', 'Contabilidad', 'I+D')

ORDER BY e.apellido1, e.apellido2, e.nombre;

9. Devuelve una lista con el nombre de los empleados que tienen los departamentos que no tienen un presupuesto

entre 100000 y 200000 euros.

SELECT e.nombre

FROM empleado e

WHERE e.id\_departamento NOT IN (

SELECT d.id

FROM departamento d

WHERE d.presupuesto BETWEEN 100000 AND 200000

)

10. Devuelve un listado con el nombre de los departamentos donde existe algún empleado cuyo segundo apellido sea

NULL. Tenga en cuenta que no debe mostrar nombres de departamentos que estén repetidos.

SELECT DISTINCT d.nombre

FROM empleado e

INNER JOIN departamento d

ON e.id\_departamento = d.id

WHERE e.apellido2 IS NULL;

11. Devuelve un listado con todos los empleados junto con los datos de los departamentos donde trabajan. Este listado

también debe incluir los empleados que no tienen ningún departamento asociado.

SELECT e.\*, d.nombre

FROM empleado e

LEFT JOIN departamento d ON e.id\_departamento = d.id;

12. Devuelve un listado donde sólo aparezcan aquellos empleados que no tienen ningún departamento asociado.

SELECT \*

FROM empleado e

LEFT JOIN departamento d ON e.id\_departamento = d.id

WHERE e.id\_departamento IS NULL;

13. Devuelve un listado donde sólo aparezcan aquellos departamentos que no tienen ningún empleado asociado.

SELECT d.id, d.nombre

FROM departamento d

LEFT JOIN empleado e ON d.id = e.id\_departamento

WHERE e.id IS NULL;

14. Devuelve un listado con todos los empleados junto con los datos de los departamentos donde trabajan. El listado

debe incluir los empleados que no tienen ningún departamento asociado y los departamentos que no tienen

ningún empleado asociado. Ordene el listado alfabéticamente por el nombre del departamento.

SELECT e.\*, d.\*

FROM empleado e

LEFT JOIN departamento d ON e.id\_departamento = d.id

ORDER BY d.nombre;

15. Devuelve un listado con los empleados que no tienen ningún departamento asociado y los departamentos que no

tienen ningún empleado asociado. Ordene el listado alfabéticamente por el nombre del departamento.

SELECT e.\*, d.\*

FROM empleado e

LEFT JOIN departamento d

ON e.id\_departamento = d.id

WHERE e.id\_departamento IS NULL OR d.id IS NULL

ORDER BY d.nombre;

• Consultas resumen

1. Calcula la suma del presupuesto de todos los departamentos.

SELECT SUM(presupuesto) AS suma\_presupuesto

FROM departamento;

2. Calcula la media del presupuesto de todos los departamentos.

SELECT AVG(presupuesto) as media\_presupuesto

FROM departamento;

3. Calcula el valor mínimo del presupuesto de todos los departamentos.

SELECT MIN(presupuesto) AS presupuesto\_minimo

FROM departamento;

4. Calcula el nombre del departamento y el presupuesto que tiene asignado, del departamento con menor

presupuesto.

SELECT d.nombre, d.presupuesto

FROM departamento d

INNER JOIN (

SELECT MIN(presupuesto) as min\_presupuesto

FROM departamento

) t

ON d.presupuesto = t.min\_presupuesto

5. Calcula el valor máximo del presupuesto de todos los departamentos.

SELECT MAX(presupuesto) AS max\_presupuesto

FROM departamento;

6. Calcula el nombre del departamento y el presupuesto que tiene asignado, del departamento con mayor

presupuesto.

SELECT nombre, presupuesto

FROM departamento

WHERE presupuesto = (

SELECT MAX(presupuesto)

FROM departamento

)

7. Calcula el número total de empleados que hay en la tabla empleado.

SELECT COUNT(\*) AS total\_empleados FROM empleado;

8. Calcula el número de empleados que no tienen NULL en su segundo apellido.

SELECT COUNT(\*) as Num\_Empleados\_Con\_Segundo\_Apellido

FROM empleado

WHERE apellido2 IS NOT NULL;

9. Calcula el número de empleados que hay en cada departamento. Tienes que devolver dos columnas, una con el

nombre del departamento y otra con el número de empleados que tiene asignados.

SELECT d.nombre, COUNT(e.id) AS num\_empleados

FROM departamento d

LEFT JOIN empleado e ON d.id = e.id\_departamento

GROUP BY d.nombre;

10. Calcula el nombre de los departamentos que tienen más de 2 empleados. El resultado debe tener dos columnas,

una con el nombre del departamento y otra con el número de empleados que tiene asignados.

SELECT d.nombre, COUNT(\*) as num\_empleados

FROM empleado e

RIGHT JOIN departamento d ON e.id\_departamento = d.id

GROUP BY d.nombre

HAVING COUNT(\*) > 2

11. Calcula el número de empleados que trabajan en cada uno de los departamentos. El resultado de esta consulta

también tiene que incluir aquellos departamentos que no tienen ningún empleado asociado.

SELECT d.nombre, COUNT(e.id) as numero\_empleados

FROM departamento d

LEFT JOIN empleado e ON d.id = e.id\_departamento

GROUP BY d.nombre

12. Calcula el número de empleados que trabajan en cada uno de los departamentos que tienen un presupuesto mayor

a 200000 euros.

SELECT d.nombre, COUNT(e.id) AS num\_empleados

FROM departamento d

LEFT JOIN empleado e ON d.id = e.id\_departamento

WHERE d.presupuesto > 200000

GROUP BY d.nombre;

• Subconsultas

1. Devuelve un listado con todos los empleados que tiene el departamento de Sistemas. (Sin utilizar INNER JOIN).

SELECT id, nif, nombre, apellido1, apellido2, id\_departamento

FROM empleado

WHERE id\_departamento = (SELECT id FROM departamento WHERE nombre =

'Sistemas');

2. Devuelve el nombre del departamento con mayor presupuesto y la cantidad que tiene asignada.

SELECT nombre, presupuesto

FROM departamento

WHERE presupuesto = (SELECT MAX(presupuesto) FROM departamento);

3. Devuelve el nombre del departamento con menor presupuesto y la cantidad que tiene asignada.

SELECT nombre, presupuesto

FROM departamento

WHERE presupuesto = (SELECT MIN (presupuesto) FROM departamento);

4. Devuelve el nombre del departamento con mayor presupuesto y la cantidad que tiene asignada. Sin hacer uso de

MAX, ORDER BY ni LIMIT.

SELECT nombre, presupuesto

FROM departamento

WHERE presupuesto = (SELECT presupuesto FROM departamento WHERE

presupuesto >= ALL(SELECT presupuesto FROM departamento));

5. Devuelve el nombre del departamento con menor presupuesto y la cantidad que tiene asignada. Sin hacer uso de

MIN, ORDER BY ni LIMIT.

SELECT nombre, presupuesto

FROM departamento

WHERE presupuesto = (SELECT presupuesto FROM departamento WHERE

presupuesto >= ALL(SELECT presupuesto FROM departamento));