FP055-Introduccion a las bases de datos.

AA4. Sentencias DQL y DML

Contenido

Caso Teórico.

Caso 1.
Repasar las sentencias de SQL de tipo Data Manipulation Language (DML) y Data Query Languaje (DQL) que permiten manipular los datos de las tablas, respondiendo a las siguientes preguntas:

Las instrucciones de SQL de tipo Data Manipulation Language (DML) y Data Query Language (DQL) se utilizan para manipular y consultar los datos en las bases de datos:

A. ¿Cuál es el objetivo de las instrucciones siguientes:

INSTRUCCIÓN (SENTENCIA)	OBJETIVO
SELECT	Recuperar datos de una o varias tablas. Permite especificar y filtrar exactamente qué datos se desean ver.
UPDATE	Modificar los valores de los datos existentes en una o varias tablas.
DELETE	Eliminar filas de una tabla.
INSERT	Agregar nuevas filas a una tabla.

SELECT: Se utiliza para consultar y extraer datos de una base de datos. Puedes seleccionar datos de una tabla o a través de varias tablas utilizando JOINs, y puedes filtrar y ordenar esos datos según sea necesario.

UPDATE: Esta sentencia se utiliza para actualizar o cambiar los datos existentes en una tabla. Puedes actualizar uno o varios registros a la vez y es posible aplicar condiciones para especificar exactamente qué registros deben ser actualizados.

DELETE: La instrucción DELETE se utiliza para borrar registros de una tabla. Puede eliminar un único registro o múltiples registros que coincidan con un criterio específico.

INSERT: La sentencia INSERT INTO se usa para insertar nuevos registros en una tabla. Puedes insertar valores en todas las columnas o solo en algunas especificando los nombres de las columnas y los valores correspondientes.

B. ¿Qué carácter se usa para delimitar los valores de los campos tipo texto y/o fecha durante su utilización en una consulta?

En SQL, los valores de los campos de tipo texto y fecha se delimitan comúnmente utilizando comillas simples ('). Por ejemplo:

- Para un campo de texto:

```
SELECT * FROM tabla WHERE campo_texto = 'Este es un texto';
```

- Para un campo de fecha:

```
SELECT * FROM tabla WHERE campo_fecha = '2023-01-01';
```

INNER JOIN es una cláusula en SQL que se utiliza para combinar filas de dos o más tablas, basándose en una columna relacionada entre ellas. Es fundamentalmente útil en situaciones donde tienes datos relacionados en distintas tablas y necesitas juntar estos datos en un solo conjunto de resultados.

Cuando se utiliza **INNER JOIN**, solo se incluyen en el resultado final aquellas filas que tienen un valor coincidente en ambas tablas. Esto es, si una fila en una tabla tiene un valor en la columna de unión que no existe en la otra tabla, esa fila no se incluirá en los resultados finales.

Ejemplo básico:
`SELECT tabla1.columna1, tabla2.columna2 FROM tabla1 INNER JOIN tabla2 ON tabla1.columnaComun = tabla2.columnaComun;`

C. ¿Qué pasa si un campo tipo texto contiene un apostrofe? Ejemplo: Apellido = D'Onofrio.
¿Cómo se manipula la información en estos casos?

Cuando un campo de texto contiene un apóstrofe (como en el apellido "D'Onofrio" o It"s), Se debe escapar el apóstrofe para evitar errores de sintaxis en tu consulta SQL. En la mayoría de los sistemas de gestión de bases de datos SQL, esto se hace duplicando el apóstrofe. Aquí tienes un ejemplo:

```
SELECT * FROM tabla WHERE apellido = 'D''Onofrio';
```

En esta consulta, 'D''Onofrio' se interpreta correctamente como D'Onofrio por el motor de la base de datos.

```
SELECT * FROM tabla WHERE campo_texto = 'It''s a single quote inside a text';
```

En este caso, **It''s** se interpretará como **It's** en la base de datos.

D. Un cualificador es la parte inicial de un identificador, compuesto por una estructura que indica la ruta donde se encuentra una tabla o columna en concreto, separando cada objeto por puntos.

Los cualificadores se utilizan en SQL para:

1. **Especificar con precisión la ubicación de una tabla o columna**: Esto es especialmente útil en consultas que involucran múltiples tablas que pueden tener columnas con el mismo nombre o cuando se hacen consultas que abarcan diferentes esquemas o bases de datos.
2. **Evitar ambigüedades**: Cuando se unen varias tablas, puede haber columnas con nombres idénticos en más de una tabla. Los cualificadores permiten al motor SQL saber exactamente a qué columna nos referimos.
3. **Mejorar la legibilidad**: En consultas complejas, los cualificadores ayudan a entender de qué tabla proviene cada columna.

El orden que se sigue al usar cualificadores en SQL es el siguiente:

1. **Nombre de la base de datos**: El nombre de la base de datos que contiene el esquema. Este es el nivel más alto de cualificación.
2. **Nombre del esquema**: El nombre del esquema dentro de la base de datos. Un esquema es una forma de agrupar objetos de base de datos relacionados, como tablas, vistas, y procedimientos almacenados.
3. **Nombre de la tabla**: El nombre de la tabla que contiene la columna a la que queremos referirnos.
4. **Nombre de la columna**: El nombre de la columna específica en la tabla.

La estructura sería así: **base_de_datos.esquema.tabla.columna**

Cuando es necesario delimitar los nombres de objeto, como cuando contienen caracteres especiales, espacios o coinciden con palabras reservadas de SQL, se utilizan delimitadores específicos que varían según el sistema de gestión de base de datos. En la mayoría de los sistemas SQL, se utilizan corchetes (**[ ]**), comillas dobles (**"**), o la comilla invertida/backtick (**`**). Por ejemplo:

- En SQL Server, utilizas corchetes:

```
SELECT [columna] FROM [base_de_datos].[esquema].[tabla];
```

- En MySQL y PostgreSQL, si es necesario, utilizas la comilla invertida o comillas dobles respectivamente:

```
-- MySQL SELECT `columna` FROM `base_de_datos`.`tabla`; -- PostgreSQL
SELECT "columna" FROM "esquema"."tabla";
```

Es importante recordar que el uso de delimitadores es opcional a menos que el identificador contenga caracteres especiales o sea una palabra reservada.

E. Las instrucciones o sentencias SQL están formadas por cláusulas (algunas de ellas obligatorias). Una cláusula tiene una estructura y realiza una función dentro de una instrucción. A este respecto, escribe el orden de escritura correcto en el cual se deben añadir a las instrucciones DML/DQL las cláusulas siguientes (no todas las cláusulas se incluyen en todas las sentencias).

Para organizar las cláusulas según el tipo de sentencia en SQL, creare dos tablas: una para las sentencias SELECT (DQL) y otra para las sentencias UPDATE y DELETE (DML). La cláusula INSERT es ligeramente diferente y normalmente no incluye cláusulas como **WHERE**, **ORDER BY**, etc.

SENTENCIA SELECT

CLÁUSULA	ORDEN	OBJETIVO
SELECT	1	Especifica las columnas que se van a recuperar en la consulta.
DISTINCT	2	Elimina duplicados en el resultado de la consulta.
FROM	3	Indica la tabla desde donde se van a recuperar los datos.
WHERE	4	Filtra las filas que se devuelven en la consulta.
GROUP BY	5	Agrupa las filas que tienen el mismo valor en las columnas especificadas.
HAVING	6	Filtra grupos de filas después de la agregación.
ORDER BY	7	Ordena las filas resultantes por una o más columnas.
LIMIT/TOP	8	Limita el número de filas devueltas en la consulta (LIMIT en MySQL, TOP en SQL Server).

SENTENCIA UPDATE

CLÁUSULA	ORDEN	OBJETIVO
UPDATE	1	Indica la tabla en la que se van a actualizar los datos.
SET	2	Especifica las columnas a actualizar y los valores correspondientes.
FROM	3	Especifica las tablas adicionales utilizadas en la consulta, si se une con otras tablas.
WHERE	4	Filtra las filas que se actualizarán.

SENTENCIA DELETE

CLÁUSULA	ORDEN	OBJETIVO
DELETE	1	Indica que se van a eliminar filas de una tabla.
FROM	2	Especifica la tabla de donde se van a eliminar los datos.
WHERE	3	Filtra las filas que se eliminarán.

La cláusula **INTO** se utiliza principalmente en las sentencias **SELECT INTO** para copiar datos de una consulta a una nueva tabla y en **INSERT INTO** para indicar la tabla a la cual se le insertarán datos. La sintaxis y el orden pueden variar dependiendo del sistema de gestión de base de datos que estés utilizando. Por ejemplo, **TOP** es específico de SQL Server y T-SQL, mientras que **LIMIT** se usa en MySQL y PostgreSQL. **ORDER BY** en una sentencia **UPDATE** es específico para algunos RDBMS y puede no ser válido en otros.

Caso 2.

Rellena la siguiente tabla de operadores de MySQL.

OPERADOR	OBJETIVO
| |, OR	Evalúa como verdadero si cualquiera de las condiciones es verdadera.
&&, AND	Evalúa como verdadero solo si todas las condiciones son verdaderas.
NOT	Niega una condición.
BETWEEN	Verifica si un valor está entre un rango de dos valores.
=	Igualdad.
<	Menor que.
<=	Menor o igual que.
>	Mayor que.
>=	Mayor o igual que.
!=	No igual o diferente.
IN	Verifica si un valor está dentro de un conjunto de valores.
+	Suma de valores.
-	Resta de valores o negación de un valor.
*	Multiplicación de valores.
/, DIV	División de valores.
%, MOD	Módulo - el resto de una división.
LIKE	Comparación de patrones (comodines % y _ pueden ser utilizados).

Caso 3.

Rellena la siguiente tabla de funciones de MySQL indicando a qué tipo de datos se aplica y dando un ejemplo.

FUNCTION	OBJETIVO, EJEMPLO	TIPO DE DATOS
UPPER	Convierte texto a mayúsculas. Ej: UPPER('hello') da como resultado 'HELLO'.	Cadena de caracteres
LEFT	Extrae los primeros caracteres de una cadena. Ej: LEFT ('Hello', 2) da como resultado 'He'.	Cadena de caracteres
CURRENT_DATE	Obtiene la fecha actual. Ej: CURRENT_DATE da la fecha de hoy.	Fecha
DATE_ADD	Añade un intervalo de tiempo a una fecha. Ej: DATE_ADD('2023-01-01', INTERVAL 1 DAY) añade un día a la fecha especificada.	Fecha
DAY	Obtiene el día del mes de una fecha. Ej: DAY('2023-01-01') da como resultado 1.	Fecha
MONTH	Obtiene el mes de una fecha. Ej: MONTH('2023-01-01') da como resultado 1.	Fecha
YEAR	Obtiene el año de una fecha. Ej: YEAR('2023-01-01') da como resultado 2023.	Fecha
ROUND	Redondea un número. Ej: ROUND(123.456, 2) da como resultado 123.46.	Número
CONCAT	Concatena dos o más cadenas. Ej: CONCAT('Hello ', 'World') da como resultado 'Hello World'.	Cadena de caracteres
FORMAT	Da formato a un número con un número específico de decimales. Ej: FORMAT(123.4567, 2) da como resultado '123.46'.	Número

Estas funciones son herramientas útiles para manipular y formatear datos en consultas SQL en MySQL. Dependiendo de la función, pueden aplicarse a diferentes tipos de datos como cadenas de texto, fechas o números.

Caso 4

¿Qué indica un valor NULL en una tabla? ¿Cómo se consulta si en un campo existen valores nulos?

Un valor **NULL** en una tabla de una base de datos indica que el campo no tiene un valor asignado. Es importante destacar que **NULL** no es lo mismo que un espacio en blanco o cero; es una ausencia de valor. En el contexto de las bases de datos, **NULL** puede representar información desconocida, inaplicable o que simplemente no se ha ingresado.

Para consultar si en un campo existen valores nulos, utilizas la cláusula **IS NULL** en una sentencia **SELECT**. Por ejemplo:

```
SELECT * FROM tabla WHERE columna IS NULL;
```

Este comando seleccionará todas las filas de la tabla donde el valor de "columna" es **NULL**.

Del mismo modo, si deseas seleccionar filas donde el campo no es **NULL**, puedes usar **IS NOT NULL**:

```
SELECT * FROM tabla WHERE columna IS NOT NULL;
```

Este comando seleccionará todas las filas de la tabla donde el valor de "columna" no es **NULL**.

Caso 5.

¿Qué son funciones agregadas y con qué cláusula se usan? ¿Cuáles funciones agregadas proporciona el SQL?
Las funciones agregadas en SQL son funciones que realizan un cálculo sobre un conjunto de valores y devuelven un único valor. Se utilizan comúnmente para realizar operaciones como sumar, contar, encontrar el máximo, etc., sobre un conjunto de filas. Estas funciones son especialmente útiles para análisis de datos y generación de informes.

Las funciones agregadas se utilizan típicamente con la cláusula **GROUP BY**, la cual agrupa filas que tienen el mismo valor en una columna especificada y permite que las funciones agregadas realicen cálculos en cada grupo.

Algunas de las funciones agregadas más comunes en SQL son:

COUNT: Cuenta el número de filas en un conjunto de resultados. Por ejemplo, **COUNT (*)** cuenta todas las filas, mientras que **COUNT (columna)** cuenta las filas donde **columna** no es **NULL**.

SUM: Suma los valores de una columna para todas las filas en un conjunto de resultados. Por ejemplo, **SUM (columna)** sumaría todos los valores de **columna**.

AVG: Calcula el promedio de los valores en una columna. Por ejemplo, **AVG (columna)** daría el promedio de todos los valores en **columna**.

MAX: Encuentra el valor máximo en una columna. Por ejemplo, **MAX (columna)** devolvería el valor más alto en **columna**.

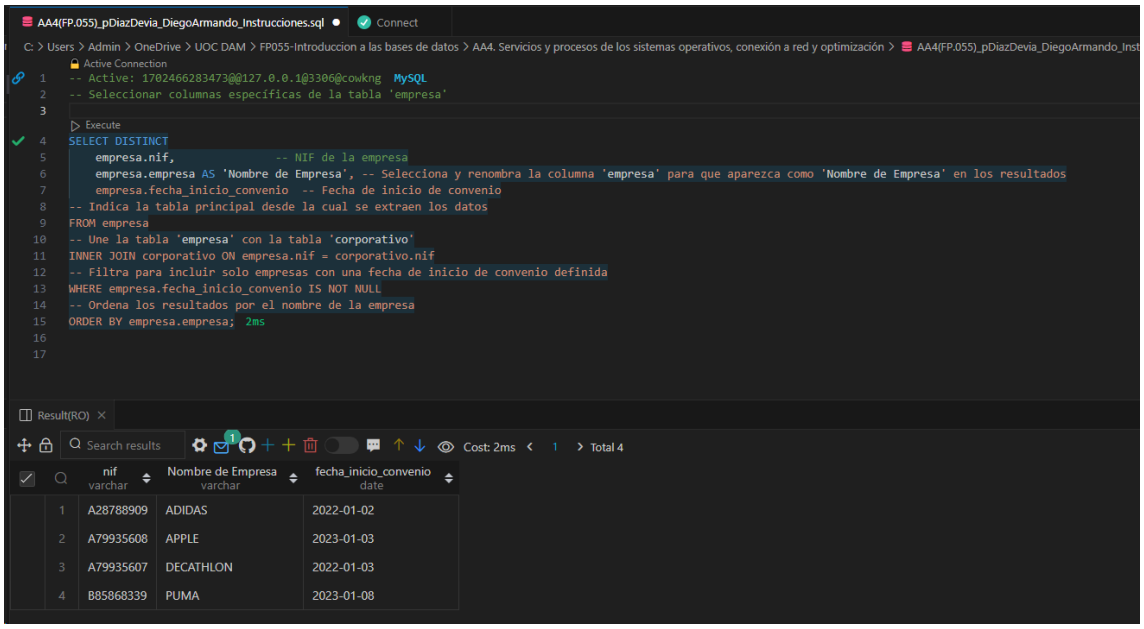
MIN: Encuentra el valor mínimo en una columna. Por ejemplo, **MIN (columna)** devolvería el valor más bajo en **columna**.

Estas funciones son cruciales para resumir y analizar grandes conjuntos de datos en bases de datos relacionales.

Caso Practico.

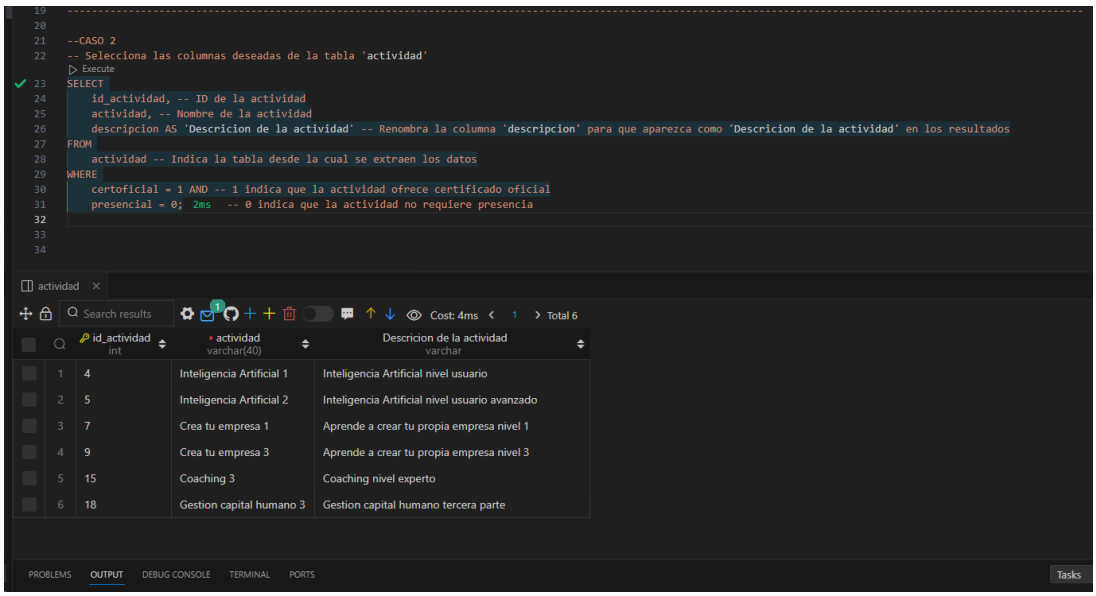
Caso 1.

Muestra un listado con el NIF, Nombre de Empresa y la Fecha de Inicio de convenio de las empresas que tienen convenio con el coWorking, ordenando el listado por el Nombre de la Empresa.



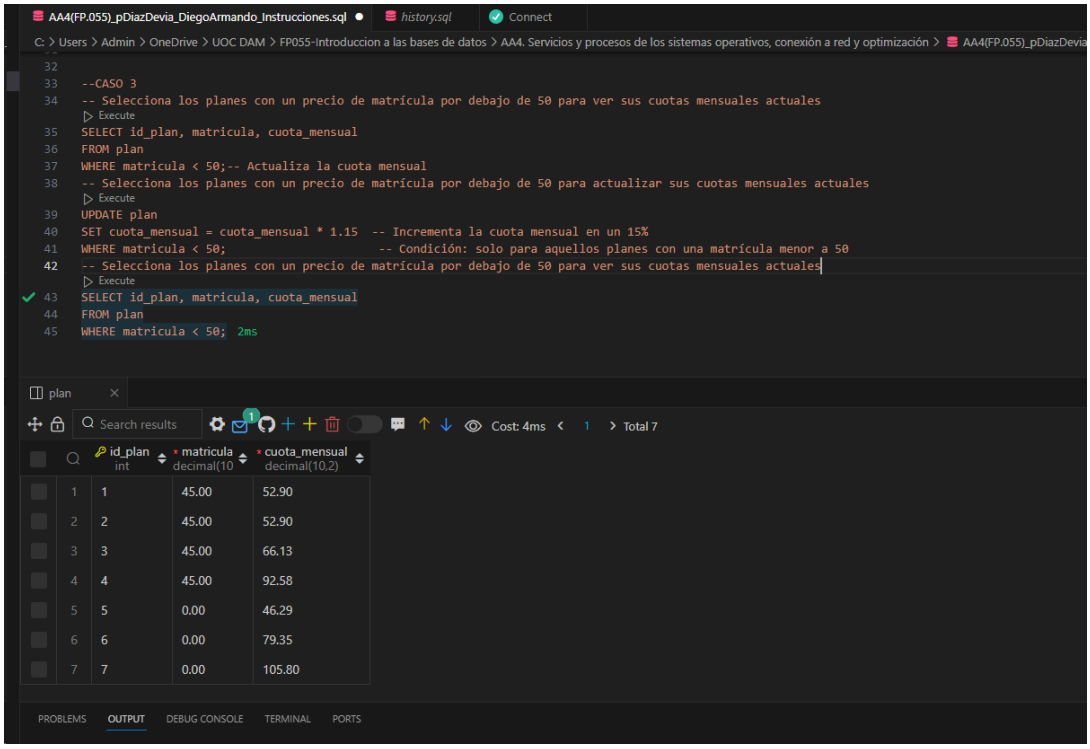
Caso 2.

Muestra un listado de las actividades impartidas en las que se obtenga certificado oficial y que no requieran presencia.



Caso 3

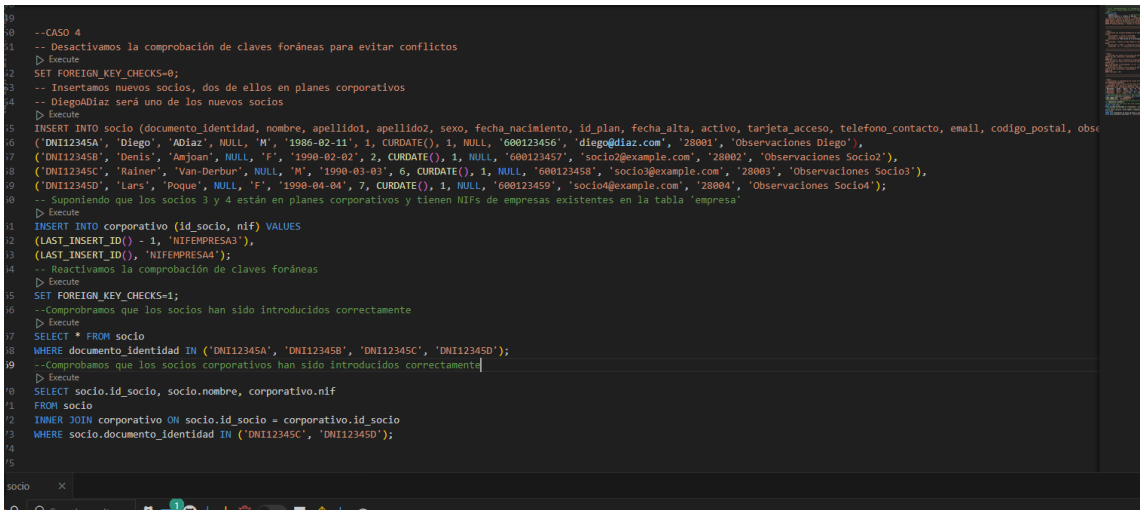
Actualiza la cuota mensual de aquellos planes cuyo precio de la matrícula este por debajo de 50. Incrementa esta cuota en un 15%.

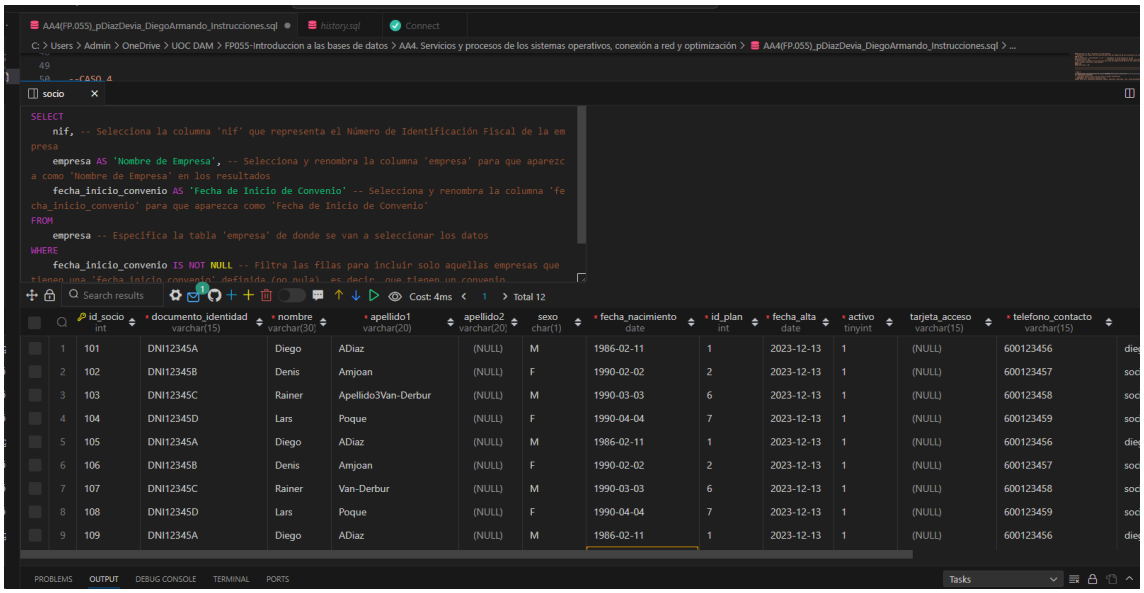


Caso 4.

Inserta 4 nuevos socios (uno de los cuales deberás ser tú). Dos (2) de ellos deberán estar en planes corporativos. Ten cuidado de rellenar todas las tablas pertinentes y los datos necesarios. Desactiva la FK si es necesario.

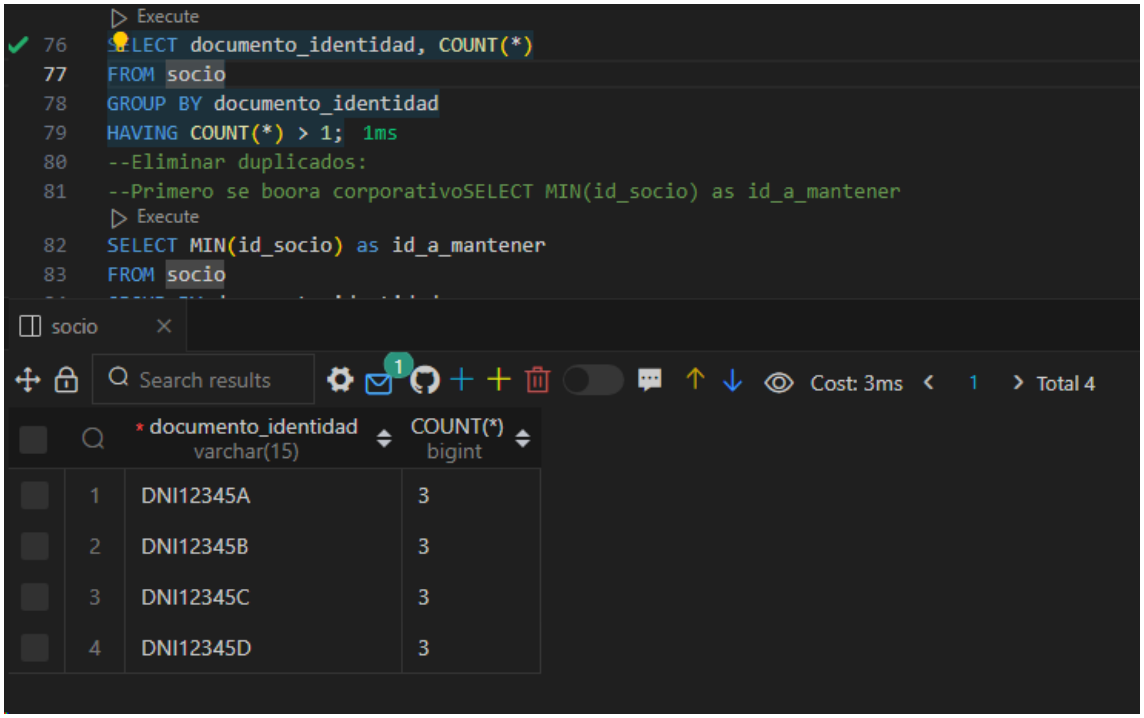
Al hacer la inserción de datos de esta manera, me percatado que se repiten los datos



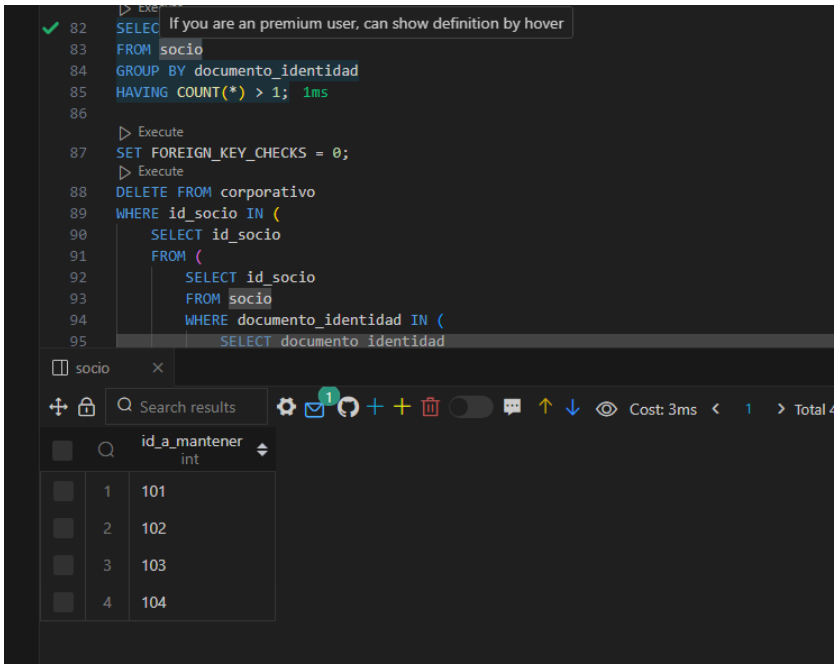


He comprobado que el código está mal e inserta los mismos usuarios varias veces la verificar los registros duplicados

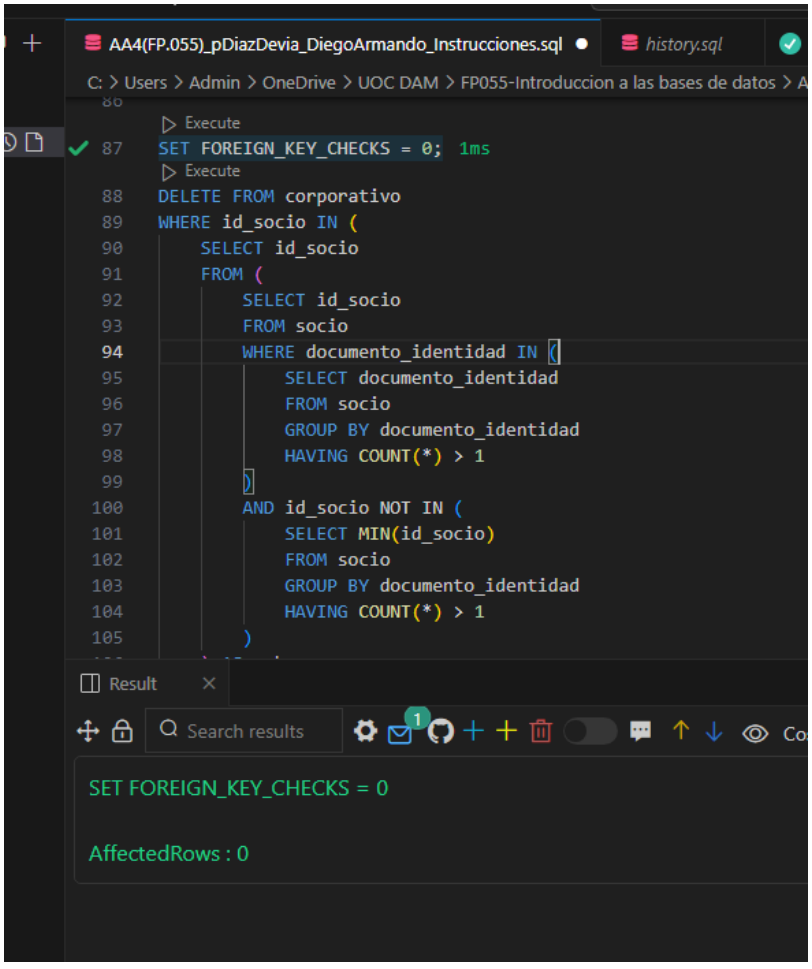
Esta consulta mostrará los documentos de identidad que están duplicados y cuántas veces se han insertado.



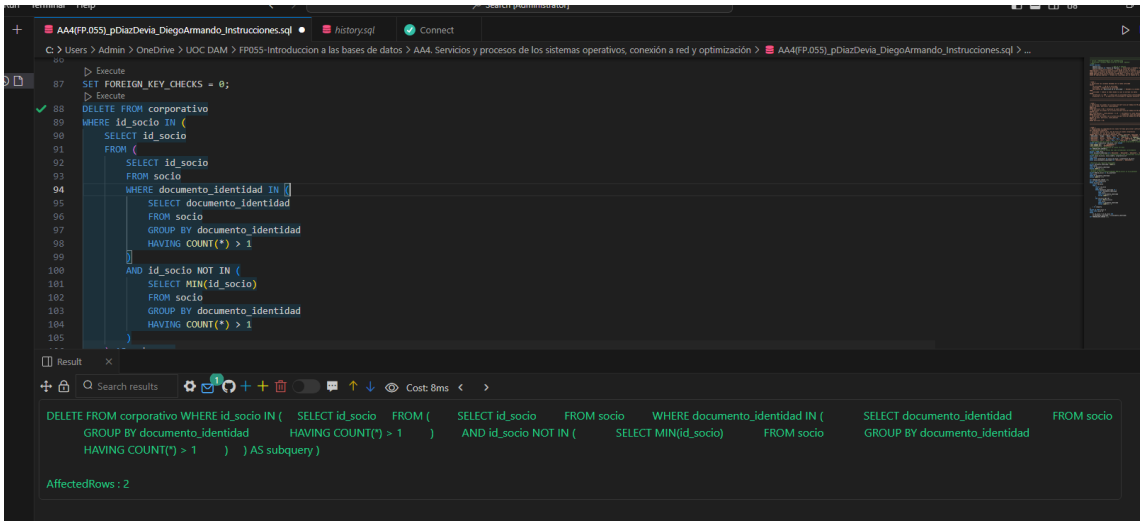
Primero, selecciona los IDs de todos los registros duplicados, exceptuando uno de ellos (por ejemplo, el de menor o mayor id_socio). Este paso es necesario para asegurarte de que no elimines todas las copias de un registro duplicado.



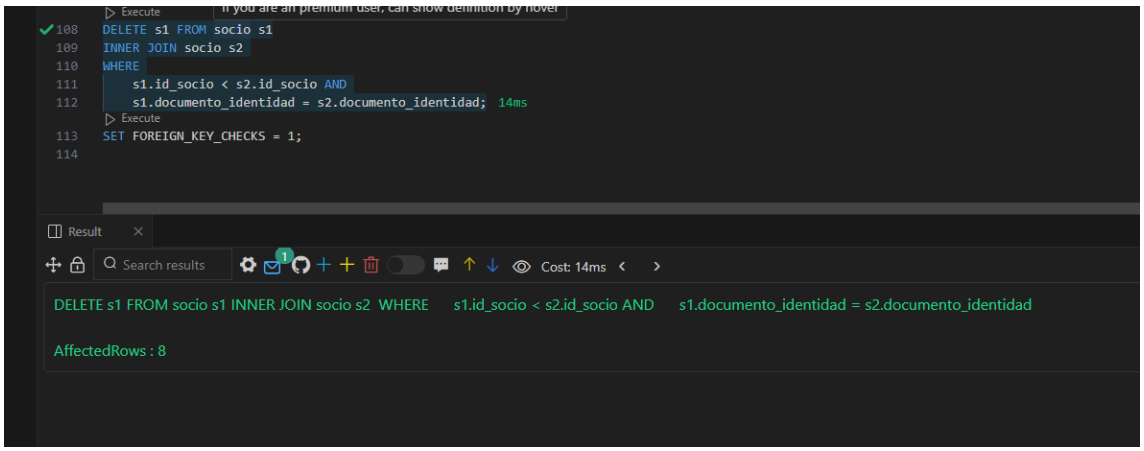
Desactivar la comprobación de claves foráneas temporalmente



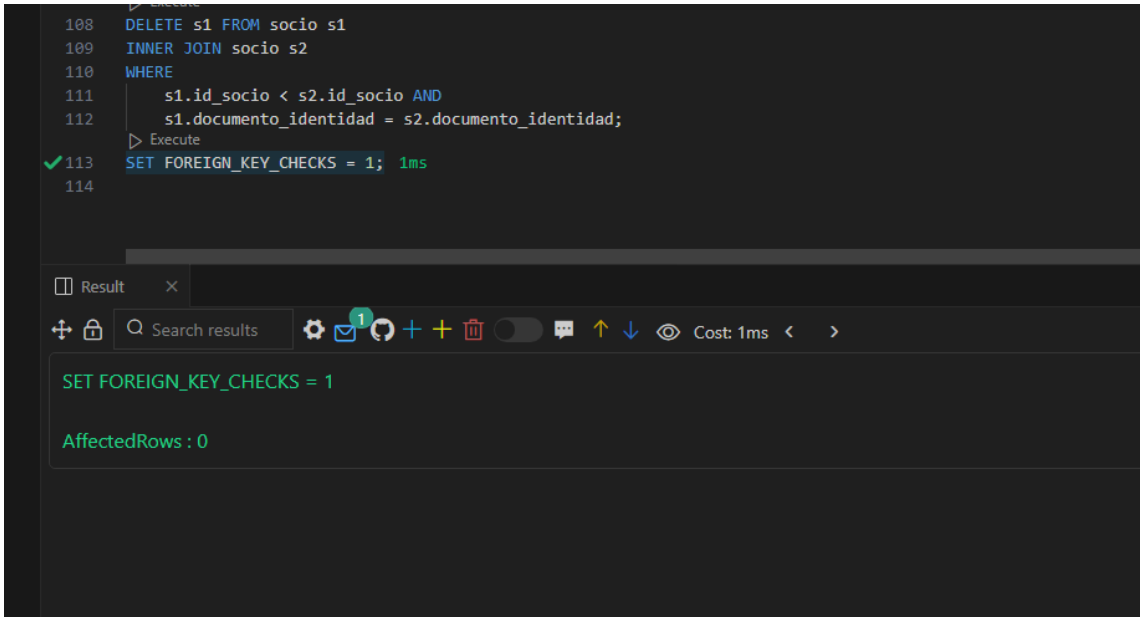
Eliminar los registros en la tabla corporativo que correspondan a los id_socio duplicados en la tabla socio.



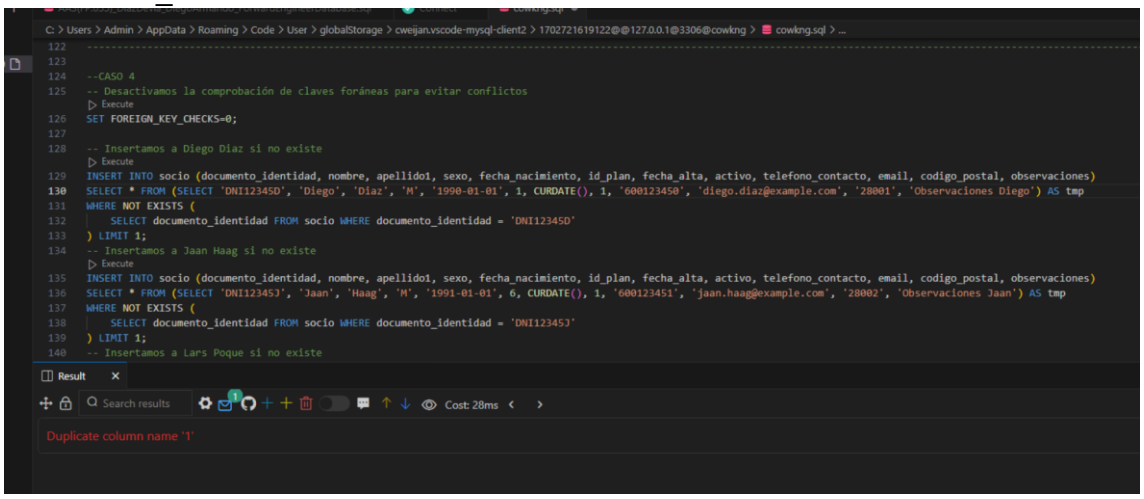
Eliminar registros duplicados de socio

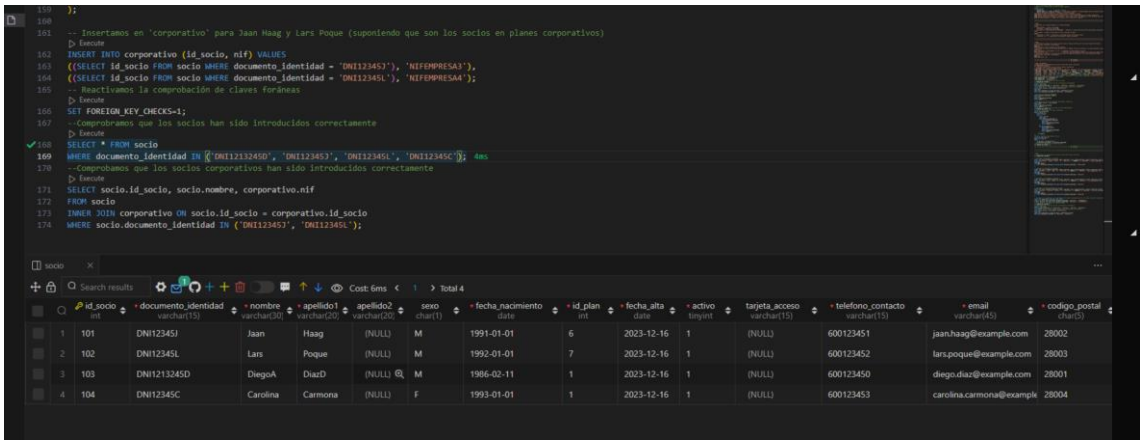


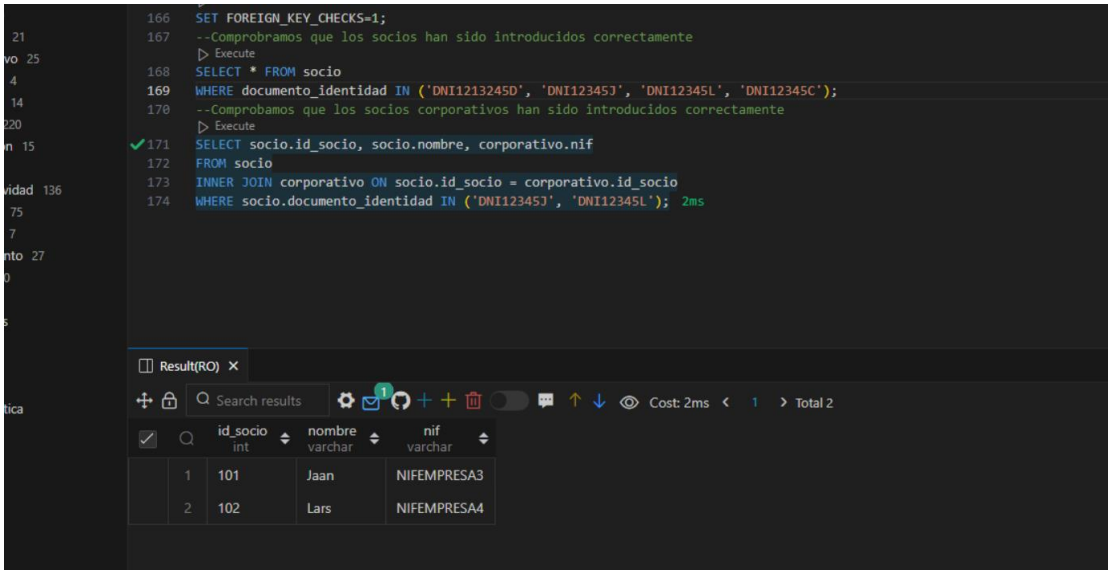
Reactivar la comprobación de claves foráneas



Volvemos a empezar, como ya habíamos creado un usuario con el documento_identidad devuelve error

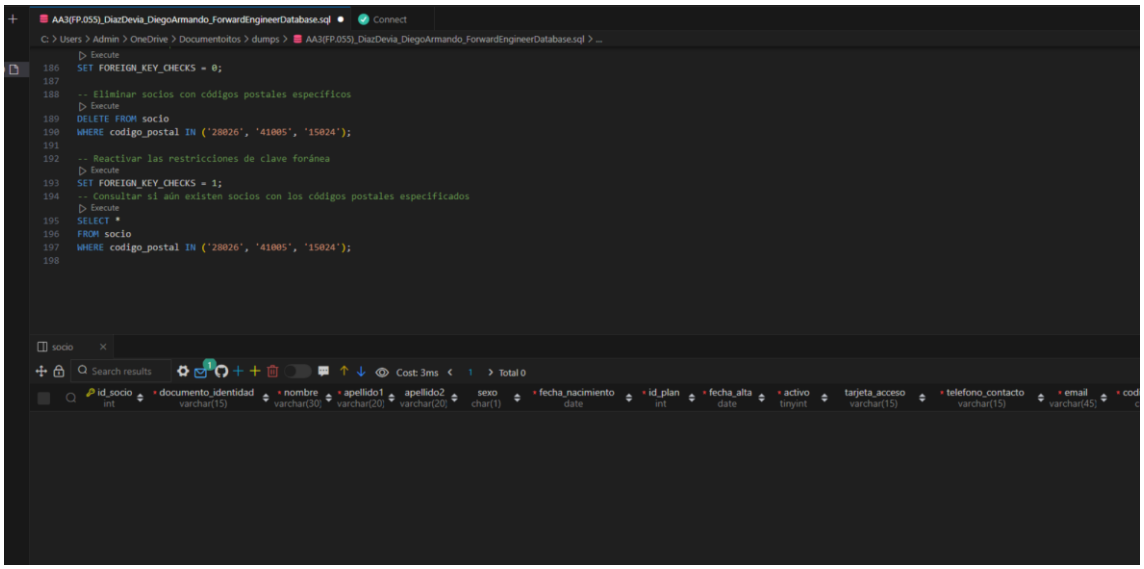






Caso 5.

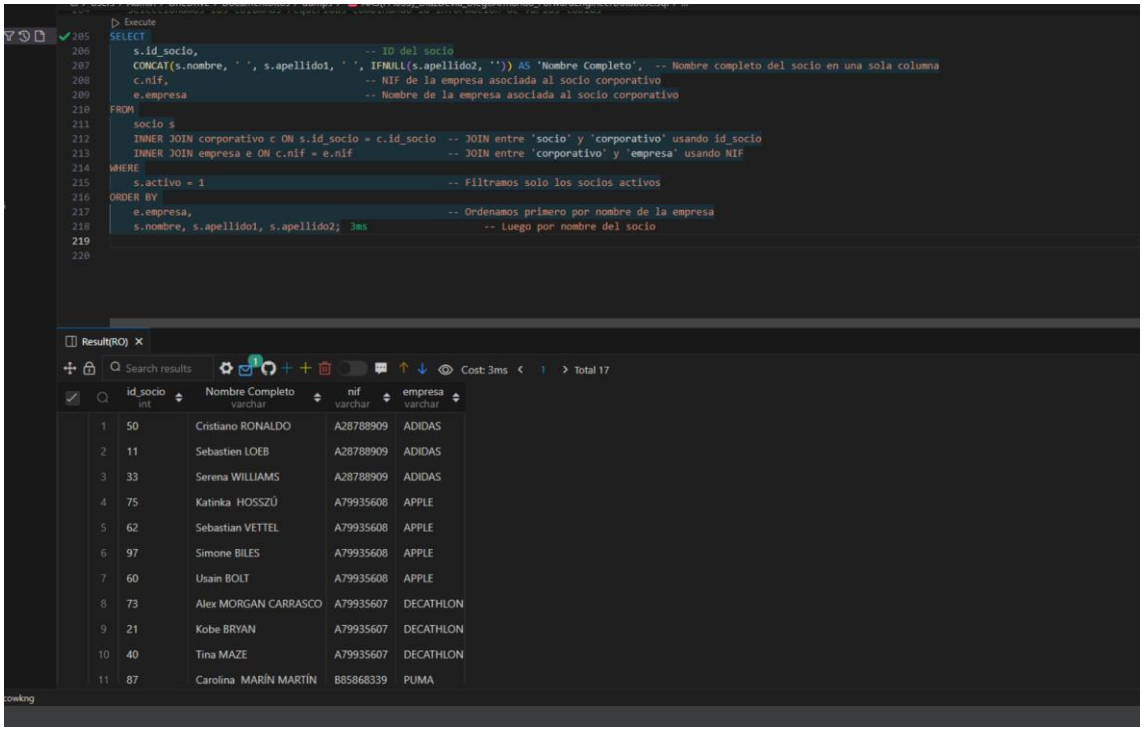
Elimina de la tabla socios aquellos socios que contengan los códigos postales 28026, 41005 y 15024. Desactiva la FK si es necesario.



Caso 6

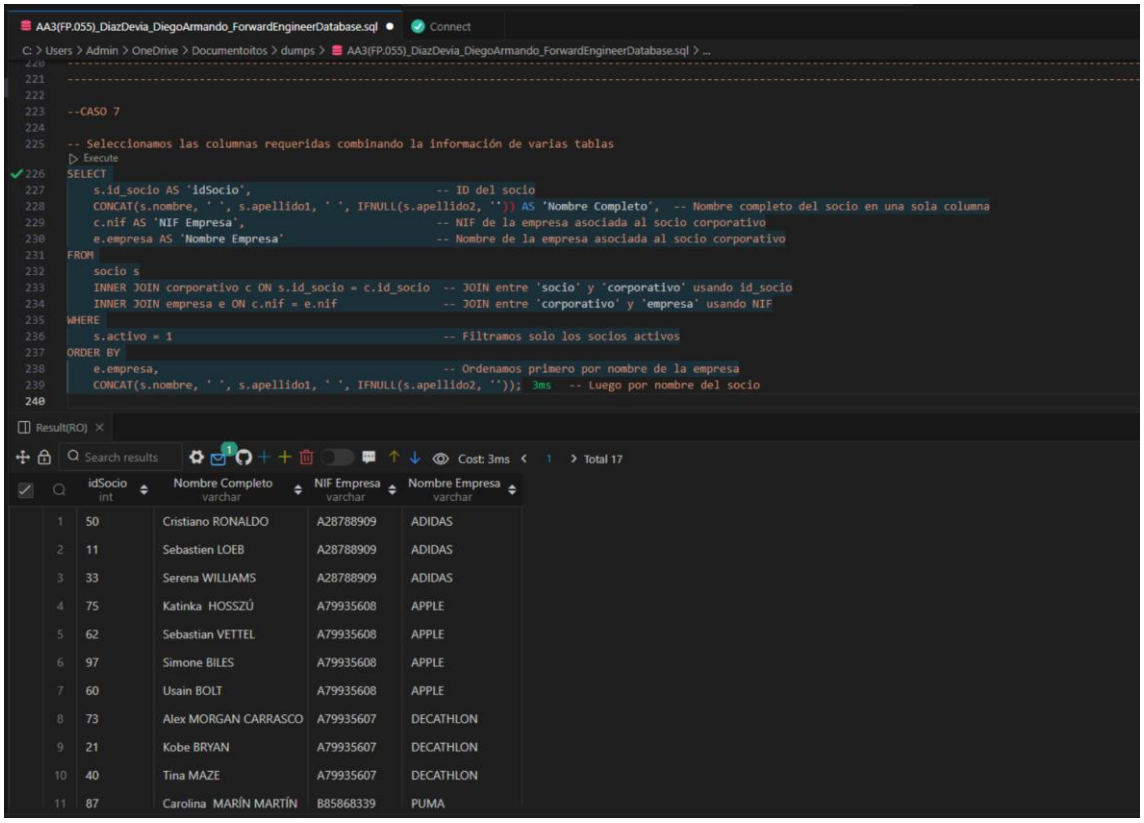
Muestra en un listado los siguientes datos para los socios corporativos: idSocio, Nombre completo (en una sola columna), NIF de la Empresa y Nombre de la Empresa. El listado deberá mostrarse ordenado por nombre de la empresa y nombre del socio.

Para realizar esta consulta en MySQL, utilizare un JOIN entre las tablas socio, corporativo y empresa. Esto nos permitirá obtener los datos necesarios de cada socio corporativo.



Caso 7.

Muestra en un listado los siguientes datos para los socios corporativos: idSocio, Nombre completo (en una sola columna), NIF de la Empresa y Nombre de la Empresa. El listado deberá mostrarse ordenado por nombre de la empresa y nombre del socio.



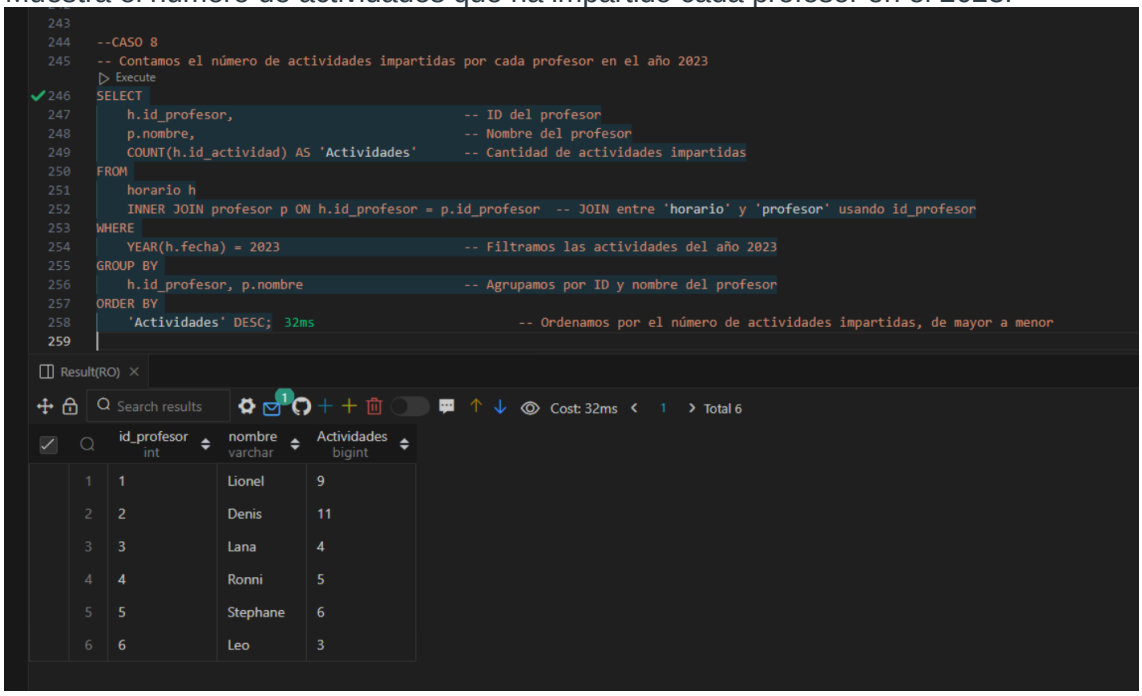
```sql
--CASO 7

-- Seleccionamos las columnas requeridas combinando la información de varias tablas
SELECT
    s.id_socio AS 'idSocio',                          -- ID del socio
    CONCAT(s.nombre, ' ', s.apellido1, ' ', IFNULL(s.apellido2, '')) AS 'Nombre Completo',  -- Nombre completo del socio en una sola columna
    c.nif AS 'NIF Empresa',                           -- NIF de la empresa asociada al socio corporativo
    e.empresa AS 'Nombre Empresa'                     -- Nombre de la empresa asociada al socio corporativo
FROM
    socio s
    INNER JOIN corporativo c ON s.id_socio = c.id_socio  -- JOIN entre 'socio' y 'corporativo' usando id_socio
    INNER JOIN empresa e ON c.nif = e.nif             -- JOIN entre 'corporativo' y 'empresa' usando NIF
WHERE
    s.activo = 1                                      -- Filtramos solo los socios activos
ORDER BY
    e.empresa,                                        -- Ordenamos primero por nombre de la empresa
    CONCAT(s.nombre, ' ', s.apellido1, ' ', IFNULL(s.apellido2, '')); 3ms  -- Luego por nombre del socio
```

Result(RO) — Cost: 3ms — Total 17

	idSocio int	Nombre Completo varchar	NIF Empresa varchar	Nombre Empresa varchar
1	50	Cristiano RONALDO	A28788909	ADIDAS
2	11	Sebastien LOEB	A28788909	ADIDAS
3	33	Serena WILLIAMS	A28788909	ADIDAS
4	75	Katinka HOSSZÚ	A79935608	APPLE
5	62	Sebastian VETTEL	A79935608	APPLE
6	97	Simone BILES	A79935608	APPLE
7	60	Usain BOLT	A79935608	APPLE
8	73	Alex MORGAN CARRASCO	A79935607	DECATHLON
9	21	Kobe BRYAN	A79935607	DECATHLON
10	40	Tina MAZE	A79935607	DECATHLON
11	87	Carolina MARÍN MARTÍN	B85868339	PUMA

Caso 8.

Muestra el número de actividades que ha impartido cada profesor en el 2023.



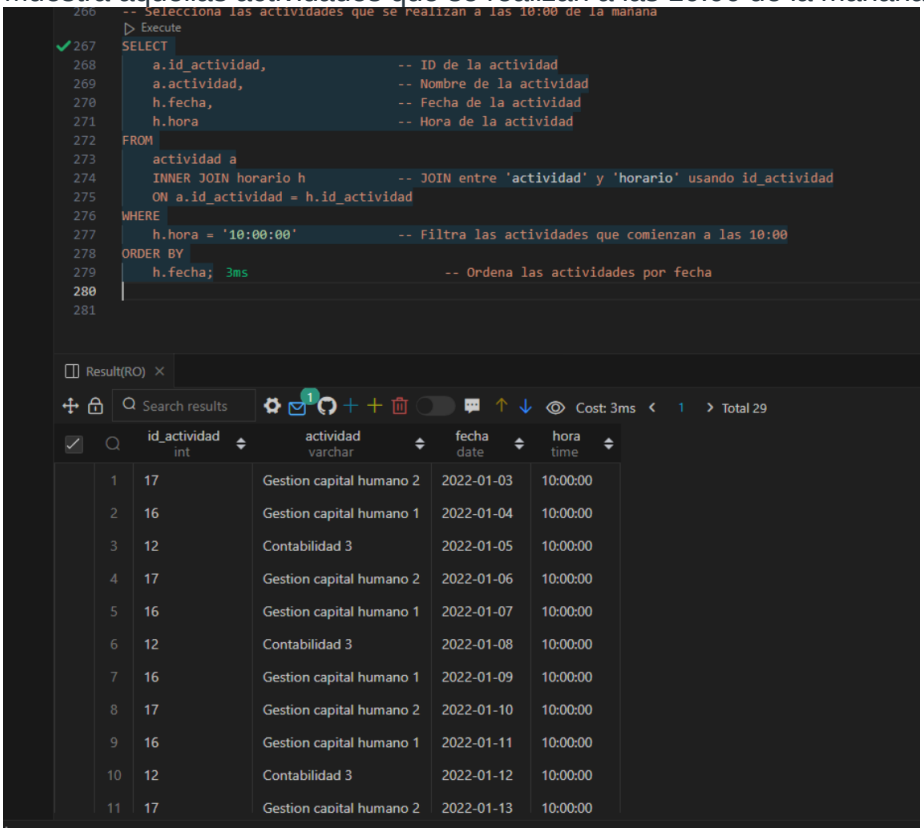
```sql
--CASO 8
-- Contamos el número de actividades impartidas por cada profesor en el año 2023
SELECT
    h.id_profesor,                                    -- ID del profesor
    p.nombre,                                         -- Nombre del profesor
    COUNT(h.id_actividad) AS 'Actividades'            -- Cantidad de actividades impartidas
FROM
    horario h
    INNER JOIN profesor p ON h.id_profesor = p.id_profesor  -- JOIN entre 'horario' y 'profesor' usando id_profesor
WHERE
    YEAR(h.fecha) = 2023                              -- Filtramos las actividades del año 2023
GROUP BY
    h.id_profesor, p.nombre                           -- Agrupamos por ID y nombre del profesor
ORDER BY
    'Actividades' DESC; 32ms                          -- Ordenamos por el número de actividades impartidas, de mayor a menor
```

Result(RO) — Cost: 32ms — Total 6

	id_profesor int	nombre varchar	Actividades bigint
1	1	Lionel	9
2	2	Denis	11
3	3	Lana	4
4	4	Ronni	5
5	5	Stephane	6
6	6	Leo	3

Caso 9.

Muestra aquellas actividades que se realizan a las 10:00 de la mañana.



```
-- Selecciona las actividades que se realizan a las 10:00 de la mañana
SELECT
    a.id_actividad,                 -- ID de la actividad
    a.actividad,                    -- Nombre de la actividad
    h.fecha,                        -- Fecha de la actividad
    h.hora                          -- Hora de la actividad
FROM
    actividad a
    INNER JOIN horario h            -- JOIN entre 'actividad' y 'horario' usando id_actividad
    ON a.id_actividad = h.id_actividad
WHERE
    h.hora = '10:00:00'             -- Filtra las actividades que comienzan a las 10:00
ORDER BY
    h.fecha;  3ms                       -- Ordena las actividades por fecha
```

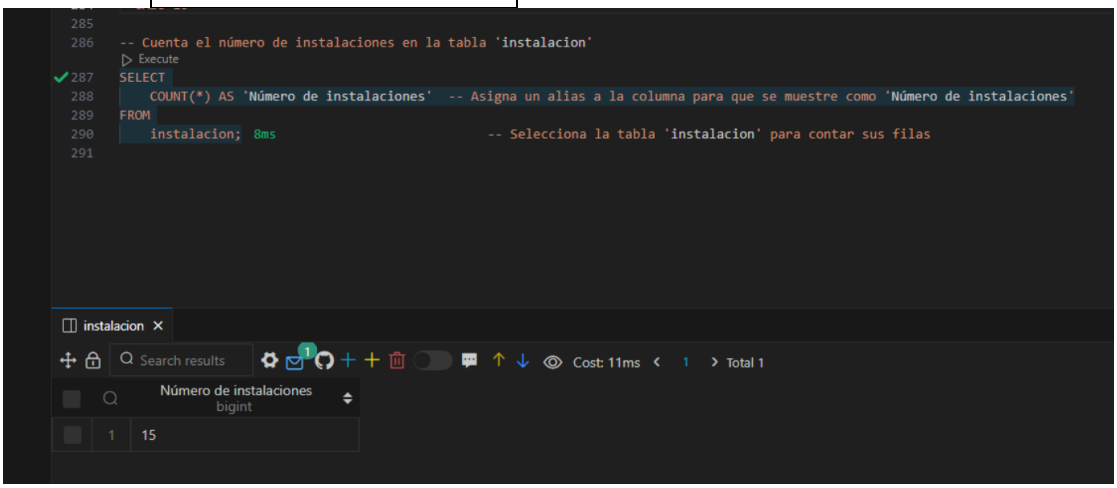
Result(RO) — Cost: 3ms — Total 29

	id_actividad int	actividad varchar	fecha date	hora time
1	17	Gestion capital humano 2	2022-01-03	10:00:00
2	16	Gestion capital humano 1	2022-01-04	10:00:00
3	12	Contabilidad 3	2022-01-05	10:00:00
4	17	Gestion capital humano 2	2022-01-06	10:00:00
5	16	Gestion capital humano 1	2022-01-07	10:00:00
6	12	Contabilidad 3	2022-01-08	10:00:00
7	16	Gestion capital humano 1	2022-01-09	10:00:00
8	17	Gestion capital humano 2	2022-01-10	10:00:00
9	16	Gestion capital humano 1	2022-01-11	10:00:00
10	12	Contabilidad 3	2022-01-12	10:00:00
11	17	Gestion capital humano 2	2022-01-13	10:00:00

Caso 10.

Contar cuántas instalaciones tiene registrado coWorking en la tabla de instalaciones. Al ejecutar la consulta, deberá presentar la siguiente salida, donde **n,** es el número de instalaciones.

Número de instalaciones
n



```
-- Cuenta el número de instalaciones en la tabla 'instalacion'
SELECT
    COUNT(*) AS 'Número de instalaciones'  -- Asigna un alias a la columna para que se muestre como 'Número de instalaciones'
FROM
    instalacion;  8ms                       -- Selecciona la tabla 'instalacion' para contar sus filas
```

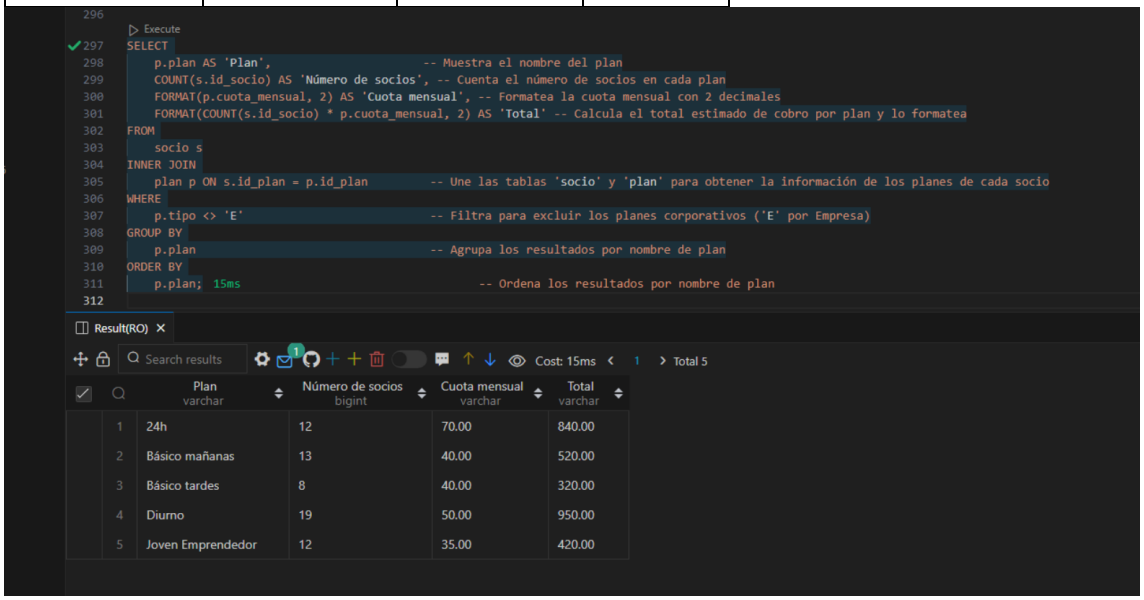
instalacion — Cost: 11ms — Total 1

	Número de instalaciones bigint
1	15

Caso 11.

Mostrar en un listado como el siguiente, el estimado de cobro del coWorking por concepto de cuota mensual para los planes no corporativos. Utiliza la función FORMAT para que los datos de Cuota Mensual y Total se muestren con 2 decimales y punto entre los miles.

Plan	Número de socios	Cuota mensual	Total
BÁSICO MAÑANAS			
BÁSICO TARDES			
DIURNO			
24h			
StartUp			
StartUp 24h			



Bibliografía.

1. Relaciones en SQL: Claves Primarias y Foráneas.
2. Llaves primarias y foráneas en bases de datos.
3. Claves primarias, foráneas y relaciones entre tablas en MySQL.
4. Guía de MySQL: Primary Key y Foreign Key explicado.
5. Video Tutorial SQL.
6. SQLZOO: SQLZOO es un recurso interactivo que permite practicar SQL en línea.
7. Tutorialspoint SQL Tutorial: Tutorialspoint ofrece una variedad de tutoriales sobre SQL y bases de datos.
8. Conexión Visual estudio con MySQL Workbench.