

FP055-Introduccion a **las bases de datos.**

AA4. Sentencias DQL y DML

Contenido

Caso Teórico.	3
Caso 1.	3
Caso 2.	6
Caso 3.	7
Caso 4.	7
Caso 5.	8
Caso Pràctico.	9
Caso 1.	9
Caso 2.	9
Caso 3.	10
Caso 4.	10
Caso 5.	15
Caso 6.	15
Caso 7.	16
Caso 8.	16
Caso 9.	17
Caso 10.	17
Caso 11.	18
Bibliografia.	18

Caso Teórico.

Caso 1.

Repasar las sentencias de SQL de tipo Data Manipulation Language (DML) y Data Query Language (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 apóstrofe? 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, creame 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	Agrupar 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.

```

AA4(FP.055)_pDiazDevia_DiegoArmando_Instrucciones.sql • Connect
C: > Users > Admin > OneDrive > UOC DAM > FP055-Introducción a las bases de datos > AA4. Servicios y procesos de los sistemas operativos, conexión a red y optimización > AA4(FP.055)_pDiazDevia_DiegoArmando_Inst
Active Connection
1 -- Seleccionar columnas específicas de la tabla 'empresa'
2
3
4 Execute
5 SELECT DISTINCT
6     empresa.nif, -- NIF de la empresa
7     empresa.empresa AS 'Nombre de Empresa', -- Selecciona y renombra la columna 'empresa' para que aparezca como 'Nombre de Empresa' en los resultados
8     empresa.fecha_inicio_convenio -- Fecha de inicio de convenio
9 -- Indica la tabla principal desde la cual se extraen los datos
10 FROM empresa
11 -- Une la tabla 'empresa' con la tabla 'corporativo'
12 INNER JOIN corporativo ON empresa.nif = corporativo.nif
13 -- Filtra para incluir solo empresas con una fecha de inicio de convenio definida
14 WHERE empresa.fecha_inicio_convenio IS NOT NULL
15 -- Ordena los resultados por el nombre de la empresa
16 ORDER BY empresa.empresa; 2ms
17
Result(RO) X
Search results
nif varchar Nombre de Empresa varchar fecha_inicio_convenio date
1 A28788909 ADIDAS 2022-01-02
2 A79935608 APPLE 2023-01-03
3 A79935607 DECATHLON 2022-01-03
4 B85868339 PUMA 2023-01-08
Cost: 2ms < 1 > Total 4

```

Caso 2.

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

```

19
20
21 --CASO 2
22 -- Selecciona las columnas deseadas de la tabla 'actividad'
23 Execute
24 SELECT
25     id_actividad, -- ID de la actividad
26     actividad, -- Nombre de la actividad
27     descripcion AS 'Descripción de la actividad' -- Renombra la columna 'descripcion' para que aparezca como 'Descripción de la actividad' en los resultados
28 FROM
29     actividad -- Indica la tabla desde la cual se extraen los datos
30 WHERE
31     certificado = 1 AND -- 1 indica que la actividad ofrece certificado oficial
32     presencial = 0; 2ms -- 0 indica que la actividad no requiere presencia
33
34
actividad X
Search results
id_actividad int actividad varchar(40) Descripción de la actividad varchar
1 4 Inteligencia Artificial 1 Inteligencia Artificial nivel usuario
2 5 Inteligencia Artificial 2 Inteligencia Artificial nivel usuario avanzado
3 7 Crea tu empresa 1 Aprende a crear tu propia empresa nivel 1
4 9 Crea tu empresa 3 Aprende a crear tu propia empresa nivel 3
5 15 Coaching 3 Coaching nivel experto
6 18 Gestión capital humano 3 Gestión capital humano tercera parte
Cost: 4ms < 1 > Total 6
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Tasks

```

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%.

```

AA4(FP.055)_pDiazDevia_DiegoArmando_Instrucciones.sql • history.sql • Connect
C:\Users\Admin> OneDrive > UOC DAM > FP055-Introducción a las bases de datos > AA4. Servicios y procesos de los sistemas operativos, conexión a red y optimización > AA4(FP.055)_pDiazDevia

32
33 --CASO 3
34 -- Selecciona los planes con un precio de matrícula por debajo de 50 para ver sus cuotas mensuales actuales
35 > Execute
36 SELECT id_plan, matricula, cuota_mensual
37 FROM plan
38 WHERE matricula < 50;-- Actualiza la cuota mensual
39 -- Selecciona los planes con un precio de matrícula por debajo de 50 para actualizar sus cuotas mensuales actuales
40 > Execute
41 UPDATE plan
42 SET cuota_mensual = cuota_mensual * 1.15 -- Incrementa la cuota mensual en un 15%
43 WHERE matricula < 50; -- Condición: solo para aquellos planes con una matrícula menor a 50
44 -- Selecciona los planes con un precio de matrícula por debajo de 50 para ver sus cuotas mensuales actuales
45 > Execute
46 SELECT id_plan, matricula, cuota_mensual
47 FROM plan
48 WHERE matricula < 50; 2ms

```

id_plan	matricula	cuota_mensual
1	1	45.00
2	2	45.00
3	3	45.00
4	4	45.00
5	5	0.00
6	6	0.00
7	7	0.00

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

```

--CASO 4
-- Desactivamos la comprobación de claves foráneas para evitar conflictos
-- Execute
SET FOREIGN_KEY_CHECKS=0;
-- Insertamos nuevos socios, dos de ellos en planes corporativos
-- DiegoADiaz será uno de los nuevos socios
-- Execute
INSERT INTO socio (documento_identidad, nombre, apellido1, apellido2, sexo, fecha_nacimiento, id_plan, fecha_alta, activo, tarjeta_acceso, telefono_contacto, email, codigo_postal, observaciones)
VALUES ('DNI12345A', 'Diego', 'ADiaz', NULL, 'M', '1986-02-11', 1, CURDATE(), 1, NULL, '600123456', 'diego@adiaz.com', '28001', 'Observaciones Diego'),
('DNI12345B', 'Denis', 'Amjoan', NULL, 'F', '1990-02-02', 2, CURDATE(), 1, NULL, '600123457', 'socio@example.com', '28002', 'Observaciones Socio2'),
('DNI12345C', 'Rainer', 'Van-Derbur', NULL, 'M', '1990-03-03', 6, CURDATE(), 1, NULL, '600123458', 'socio3@example.com', '28003', 'Observaciones Socio3'),
('DNI12345D', 'Lars', 'Poque', NULL, 'F', '1990-04-04', 7, CURDATE(), 1, NULL, '600123459', 'socio4@example.com', '28004', 'Observaciones Socio4');
-- Suponiendo que los socios 3 y 4 están en planes corporativos y tienen NIFs de empresas existentes en la tabla 'empresa'
-- Execute
INSERT INTO corporativo (id_socio, nif) VALUES
(LAST_INSERT_ID() - 1, 'NIFEMPRESA3'),
(LAST_INSERT_ID(), 'NIFEMPRESA4');
-- Reactivamos la comprobación de claves foráneas
-- Execute
SET FOREIGN_KEY_CHECKS=1;
-- Comprobamos que los socios han sido introducidos correctamente
-- Execute
SELECT * FROM socio
WHERE documento_identidad IN ('DNI12345A', 'DNI12345B', 'DNI12345C', 'DNI12345D');
-- Comprobamos que los socios corporativos han sido introducidos correctamente
-- Execute
SELECT socio.id_socio, socio.nombre, corporativo.nif
FROM socio
INNER JOIN corporativo ON socio.id_socio = corporativo.id_socio
WHERE socio.documento_identidad IN ('DNI12345C', 'DNI12345D');

```

AA4(FP.053)_pDiazDevia_DiegoArmando_Instrucciones.sql | history.sql | Connect

C:\Users\Admin> OneDrive > UOC DAM > FP053-introducción a las bases de datos > AA4. Servicios y procesos de los sistemas operativos, conexión a red y optimización > AA4(FP.053)_pDiazDevia_DiegoArmando_Instrucciones.sql > ...

49

SQL

socio

```

SELECT
  nif, -- Selecciona la columna 'nif' que representa el Número de Identificación Fiscal de la em
  empresa AS 'Nombre de Empresa', -- Selecciona y renombra la columna 'empresa' para que aparezca como 'Nombre de Empresa' en los resultados
  fecha_inicio_convenio AS 'Fecha de Inicio de Convenio' -- Selecciona y renombra la columna 'fecha_inicio_convenio' para que aparezca como 'Fecha de Inicio de Convenio'
FROM
  empresa -- Especifica la tabla 'empresa' de donde se van a seleccionar los datos
WHERE
  fecha_inicio_convenio IS NOT NULL -- Filtra las filas para incluir solo aquellas empresas que tienen una 'fecha_inicio_convenio' definida (no nula), es decir, que tienen un convenio.
  
```

Search results

Cost: 4ms < 1 > Total 12

	id_socio int	documento_identidad varchar(15)	nombre varchar(30)	apellido1 varchar(20)	apellido2 varchar(20)	sexo char(1)	fecha_nacimiento date	id_plan int	fecha_alta date	activo tinyint	tarjeta_acceso varchar(15)	telefono_contacto varchar(15)	
1	101	DNI12345A	Diego	ADiaz	(NULL)	M	1986-02-11	1	2023-12-13	1	(NULL)	600123456	die
2	102	DNI12345B	Denis	Amjoan	(NULL)	F	1990-02-02	2	2023-12-13	1	(NULL)	600123457	soc
3	103	DNI12345C	Rainer	Apellido3Van-Derbur	(NULL)	M	1990-03-03	6	2023-12-13	1	(NULL)	600123458	soc
4	104	DNI12345D	Lars	Poque	(NULL)	F	1990-04-04	7	2023-12-13	1	(NULL)	600123459	soc
5	105	DNI12345A	Diego	ADiaz	(NULL)	M	1986-02-11	1	2023-12-13	1	(NULL)	600123456	die
6	106	DNI12345B	Denis	Amjoan	(NULL)	F	1990-02-02	2	2023-12-13	1	(NULL)	600123457	soc
7	107	DNI12345C	Rainer	Van-Derbur	(NULL)	M	1990-03-03	6	2023-12-13	1	(NULL)	600123458	soc
8	108	DNI12345D	Lars	Poque	(NULL)	F	1990-04-04	7	2023-12-13	1	(NULL)	600123459	soc
9	109	DNI12345A	Diego	ADiaz	(NULL)	M	1986-02-11	1	2023-12-13	1	(NULL)	600123456	die

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Tasks

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.

Execute

```

76 SELECT documento_identidad, COUNT(*)
77 FROM socio
78 GROUP BY documento_identidad
79 HAVING COUNT(*) > 1; 1ms
80 --Eliminar duplicados:
81 --Primero se boora corporativoSELECT MIN(id_socio) as id_a_mantener
82 SELECT MIN(id_socio) as id_a_mantener
83 FROM socio
  
```

socio

Search results

Cost: 3ms < 1 > Total 4

	documento_identidad varchar(15)	COUNT(*) bigint
1	DNI12345A	3
2	DNI12345B	3
3	DNI12345C	3
4	DNI12345D	3

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.

```

82 SELECT id_socio FROM socio
83 GROUP BY documento_identidad
84 HAVING COUNT(*) > 1; 1ms
85
86
87 SET FOREIGN_KEY_CHECKS = 0;
88
89 DELETE FROM corporativo
90 WHERE id_socio IN (
91     SELECT id_socio
92     FROM (
93         SELECT id_socio
94         FROM socio
95         WHERE documento_identidad IN (
96             SELECT documento_identidad
97             FROM socio
98             GROUP BY documento_identidad
99             HAVING COUNT(*) > 1
100         )
101     )
102 )

```

Resultados de la consulta:

id_a_mantener	int
1	101
2	102
3	103
4	104

Desactivar la comprobación de claves foráneas temporalmente

```

87 SET FOREIGN_KEY_CHECKS = 0; 1ms
88
89 DELETE FROM corporativo
90 WHERE id_socio IN (
91     SELECT id_socio
92     FROM (
93         SELECT id_socio
94         FROM socio
95         WHERE documento_identidad IN (
96             SELECT documento_identidad
97             FROM socio
98             GROUP BY documento_identidad
99             HAVING COUNT(*) > 1
100         )
101     )
102 )

```

Resultados de la consulta:

```

SET FOREIGN_KEY_CHECKS = 0
AffectedRows : 0

```

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

```

87 SET FOREIGN_KEY_CHECKS = 0;
88 DELETE FROM corporativo
89 WHERE id_socio IN (
90     SELECT id_socio
91     FROM (
92         SELECT id_socio
93         FROM socio
94         WHERE documento_identidad IN (
95             SELECT documento_identidad
96             FROM socio
97             GROUP BY documento_identidad
98             HAVING COUNT(*) > 1
99         )
100         AND id_socio NOT IN (
101             SELECT MIN(id_socio)
102             FROM socio
103             GROUP BY documento_identidad
104             HAVING COUNT(*) > 1
105         )
106     ) AS subquery

```

Result

DELETE FROM corporativo WHERE id_socio IN (SELECT id_socio FROM (SELECT id_socio FROM socio WHERE documento_identidad IN (SELECT documento_identidad FROM socio GROUP BY documento_identidad HAVING COUNT(*) > 1)) AS subquery)

AffectedRows: 2

Eliminar registros duplicados de socio

```

108 DELETE s1 FROM socio s1
109 INNER JOIN socio s2
110 WHERE
111     s1.id_socio < s2.id_socio AND
112     s1.documento_identidad = s2.documento_identidad; 14ms
113 SET FOREIGN_KEY_CHECKS = 1;
114

```

Result

DELETE s1 FROM socio s1 INNER JOIN socio s2 WHERE s1.id_socio < s2.id_socio AND s1.documento_identidad = s2.documento_identidad

AffectedRows: 8

Reactivar la comprobación de claves foráneas

```

108 DELETE s1 FROM socio s1
109 INNER JOIN socio s2
110 WHERE
111     s1.id_socio < s2.id_socio AND
112     s1.documento_identidad = s2.documento_identidad;
113 SET FOREIGN_KEY_CHECKS = 1; 1ms
114

```

Result

SET FOREIGN_KEY_CHECKS = 1

AffectedRows: 0

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

```

C:\Users\Admin\AppData\Roaming\Code\User\globalStorage\cweijan.vscode-mysql-client2> 1702721619122@127.0.0.1@3306@cowking > cowking.sql > ...

122
123
124 --CASO 4
125 -- Desactivamos la comprobación de claves foráneas para evitar conflictos
126 > Execute
127 SET FOREIGN_KEY_CHECKS=0;
128
129 -- Insertamos a Diego Diaz si no existe
130 > Execute
131 INSERT INTO socio (documento_id, nombre, apellido1, sexo, fecha_nacimiento, id_plan, fecha_alta, activo, telefono_contacto, email, codigo_postal, observaciones)
132 SELECT * FROM (SELECT 'DNI12345D', 'Diego', 'Diaz', 'M', '1998-01-01', 1, CURDATE(), 1, '600123450', 'diego.diaz@example.com', '28001', 'Observaciones Diego') AS tmp
133 WHERE NOT EXISTS (
134   SELECT documento_id FROM socio WHERE documento_id = 'DNI12345D'
135 ) LIMIT 1;
136
137 -- Insertamos a Jaan Haag si no existe
138 > Execute
139 INSERT INTO socio (documento_id, nombre, apellido1, sexo, fecha_nacimiento, id_plan, fecha_alta, activo, telefono_contacto, email, codigo_postal, observaciones)
140 SELECT * FROM (SELECT 'DNI12345J', 'Jaan', 'Haag', 'M', '1991-01-01', 6, CURDATE(), 1, '600123451', 'jaan.haag@example.com', '28002', 'Observaciones Jaan') AS tmp
141 WHERE NOT EXISTS (
142   SELECT documento_id FROM socio WHERE documento_id = 'DNI12345J'
143 ) LIMIT 1;
144
145 -- Insertamos a Lars Poque si no existe
  
```

Result X

Search results

Duplicate column name '1'

```

159
160
161 -- Insertamos en 'corporativo' para Jaan Haag y Lars Poque (suponiendo que son los socios en planes corporativos)
162 > Execute
163 INSERT INTO corporativo (id_socio, nif) VALUES
164 ((SELECT id_socio FROM socio WHERE documento_id = 'DNI12345J'), 'NIFEMPRESA3'),
165 ((SELECT id_socio FROM socio WHERE documento_id = 'DNI12345L'), 'NIFEMPRESA4');
166
167 -- Reactivamos la comprobación de claves foráneas
168 SET FOREIGN_KEY_CHECKS=1;
169
170 -- Comprobamos que los socios han sido introducidos correctamente
171 > Execute
172 SELECT * FROM socio
173 WHERE documento_id IN ('DNI123245D', 'DNI12345J', 'DNI12345L', 'DNI12345C'); 4ms
174
175 -- Comprobamos que los socios corporativos han sido introducidos correctamente
176 > Execute
177 SELECT socio.id_socio, socio.nombre, corporativo.nif
178 FROM socio
179 INNER JOIN corporativo ON socio.id_socio = corporativo.id_socio
180 WHERE socio.documento_id IN ('DNI12345J', 'DNI12345L');
  
```

socio X

Search results

	id_socio	documento_id	nombre	apellido1	apellido2	sexo	fecha_nacimiento	id_plan	fecha_alta	activo	tarjeta_acceso	telefono_contacto	email	codigo_postal
1	101	DNI12345J	Jaan	Haag	(NULL)	M	1991-01-01	6	2023-12-16	1	(NULL)	600123451	jaan.haag@example.com	28002
2	102	DNI12345L	Lars	Poque	(NULL)	M	1992-01-01	7	2023-12-16	1	(NULL)	600123452	lars.poque@example.com	28003
3	103	DNI1213245D	Diego	Diaz	(NULL)	M	1986-02-11	1	2023-12-16	1	(NULL)	600123450	diego.diaz@example.com	28001
4	104	DNI12345C	Carolina	Carmona	(NULL)	F	1993-01-01	1	2023-12-16	1	(NULL)	600123453	carolina.carmona@example.com	28004

```

166 SET FOREIGN_KEY_CHECKS=1;
167 --Comprobamos que los socios han sido introducidos correctamente
168 > Execute
169 SELECT * FROM socio
170 WHERE documento_id IN ('DNI1213245D', 'DNI12345J', 'DNI12345L', 'DNI12345C');
171
172 --Comprobamos que los socios corporativos han sido introducidos correctamente
173 > Execute
174 SELECT socio.id_socio, socio.nombre, corporativo.nif
175 FROM socio
176 INNER JOIN corporativo ON socio.id_socio = corporativo.id_socio
177 WHERE socio.documento_id IN ('DNI12345J', 'DNI12345L'); 2ms
  
```

Result(RO) X

Search results

	id_socio	nombre	nif
1	101	Jaan	NIFEMPRESA3
2	102	Lars	NIFEMPRESA4

Caso 5.

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

```

+ AA3(FP.055)_DiazDevia_DiegoArmando_ForwardEngineDatabase.sql • Connect
C:\Users\Admin> OneDrive\Documentos\AA3(FP.055)_DiazDevia_DiegoArmando_ForwardEngineDatabase.sql > ...
D> Execute
186 SET FOREIGN_KEY_CHECKS = 0;
187
188 -- Eliminar socios con códigos postales específicos
189 D> Execute
190 DELETE FROM socio
191 WHERE codigo_postal IN ('28026', '41005', '15024');
192
193 -- Reactivar las restricciones de clave foránea
194 D> Execute
195 SET FOREIGN_KEY_CHECKS = 1;
196 -- Consultar si aún existen socios con los códigos postales especificados
197 D> Execute
198 SELECT *
199 FROM socio
200 WHERE codigo_postal IN ('28026', '41005', '15024');
201

```

socio X

Search results

Cost: 3ms

Total 0

id_socio int documento_identities varchar(15) nombre varchar(30) apellido1 varchar(20) apellido2 varchar(20) sexo char(1) fecha_nacimiento date id_plan int fecha_alta date activo tinyint tarjeta_acceso varchar(15) telefono_contacto varchar(15) email varchar(45) cod

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.

```

205 SELECT
206     s.id_socio, -- ID del socio
207     CONCAT(s.nombre, ' ', s.apellido1, ' ', IFNULL(s.apellido2, '')) AS 'Nombre Completo', -- Nombre completo del socio en una sola columna
208     c.nif, -- NIF de la empresa asociada al socio corporativo
209     e.empresa -- Nombre de la empresa asociada al socio corporativo
210 FROM
211     socio s
212     INNER JOIN corporativo c ON s.id_socio = c.id_socio -- JOIN entre 'socio' y 'corporativo' usando id_socio
213     INNER JOIN empresa e ON c.nif = e.nif -- JOIN entre 'corporativo' y 'empresa' usando NIF
214 WHERE
215     s.activo = 1 -- Filtramos solo los socios activos
216 ORDER BY
217     e.empresa, -- Ordenamos primero por nombre de la empresa
218     s.nombre, s.apellido1, s.apellido2; 3ms -- Luego por nombre del socio
219
220

```

Result(RO) X

Search results

Cost: 3ms

Total 17

id_socio	Nombre Completo	nif	empresa
50	Cristiano RONALDO	A28788909	ADIDAS
11	Sebastien LOEB	A28788909	ADIDAS
33	Serena WILLIAMS	A28788909	ADIDAS
75	Katinka HOSSZÚ	A79935608	APPLE
62	Sebastian VETTEL	A79935608	APPLE
97	Simone BILES	A79935608	APPLE
60	Usain BOLT	A79935608	APPLE
73	Alex MORGAN CARRASCO	A79935607	DECATHLON
21	Kobe BRYAN	A79935607	DECATHLON
40	Tina MAZE	A79935607	DECATHLON
87	Carolina MARÍN MARTÍN	B85868339	PUMA

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.

```

AA3(FP.055)_DiazDevia_DiegoArmando_ForwardEngineerDatabase.sql
C:\Users\Admin>OneDrive\Documentos\AA3(FP.055)_DiazDevia_DiegoArmando_ForwardEngineerDatabase.sql
221
222
223 --CASO 7
224
225 -- Seleccionamos las columnas requeridas combinando la información de varias tablas
226
227 SELECT
228     s.id_socio AS 'idSocio', -- ID del socio
229     CONCAT(s.nombre, ' ', s.apellido1, ' ', IFNULL(s.apellido2, '')) AS 'Nombre Completo', -- Nombre completo del socio en una sola columna
230     c.nif AS 'NIF Empresa', -- NIF de la empresa asociada al socio corporativo
231     e.empresa AS 'Nombre Empresa' -- Nombre de la empresa asociada al socio corporativo
232 FROM
233     socio s
234     INNER JOIN corporativo c ON s.id_socio = c.id_socio -- JOIN entre 'socio' y 'corporativo' usando id_socio
235     INNER JOIN empresa e ON c.nif = e.nif -- JOIN entre 'corporativo' y 'empresa' usando NIF
236 WHERE
237     s.activo = 1 -- Filtramos solo los socios activos
238 ORDER BY
239     e.empresa, -- Ordenamos primero por nombre de la empresa
240     CONCAT(s.nombre, ' ', s.apellido1, ' ', IFNULL(s.apellido2, '')); 3ms -- Luego por nombre del socio
241

```

	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.

```

243
244 --CASO 8
245 -- Contamos el número de actividades impartidas por cada profesor en el año 2023
246
247 SELECT
248     h.id_profesor, -- ID del profesor
249     p.nombre, -- Nombre del profesor
250     COUNT(h.id_actividad) AS 'Actividades' -- Cantidad de actividades impartidas
251 FROM
252     horario h
253     INNER JOIN profesor p ON h.id_profesor = p.id_profesor -- JOIN entre 'horario' y 'profesor' usando id_profesor
254 WHERE
255     YEAR(h.fecha) = 2023 -- Filtramos las actividades del año 2023
256 GROUP BY
257     h.id_profesor, p.nombre -- Agrupamos por ID y nombre del profesor
258 ORDER BY
259     'Actividades' DESC; 32ms
260

```

	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.

```

265 -- Selecciona las actividades que se realizan a las 10:00 de la mañana
266 > Execute
267 SELECT
268     a.id_actividad, -- ID de la actividad
269     a.actividad, -- Nombre de la actividad
270     h.fecha, -- Fecha de la actividad
271     h.hora -- Hora de la actividad
272 FROM
273     actividad a
274     INNER JOIN horario h -- JOIN entre 'actividad' y 'horario' usando id_actividad
275     ON a.id_actividad = h.id_actividad
276 WHERE
277     h.hora = '10:00:00' -- Filtra las actividades que comienzan a las 10:00
278 ORDER BY
279     h.fecha; 3ms
280
281

```

Result(RO) X

Search results

Cost: 3ms < 1 > 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

```

285
286 -- Cuenta el número de instalaciones en la tabla 'instalacion'
287 > Execute
288 SELECT
289     COUNT(*) AS 'Número de instalaciones' -- Asigna un alias a la columna para que se muestre como 'Número de instalaciones'
290 FROM
291     instalacion; 8ms
292
293

```

instalacion X

Search results

Cost: 11ms < 1 > 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			

```

296
297 > Execute
298 SELECT
299     p.plan AS 'Plan', -- Muestra el nombre del plan
300     COUNT(s.id_socio) AS 'Número de socios', -- Cuenta el número de socios en cada plan
301     FORMAT(p.cuota_mensual, 2) AS 'Cuota mensual', -- Formatea la cuota mensual con 2 decimales
302     FORMAT(COUNT(s.id_socio) * p.cuota_mensual, 2) AS 'Total' -- Calcula el total estimado de cobro por plan y lo formatea
303 FROM
304     socio s
305 INNER JOIN
306     plan p ON s.id_plan = p.id_plan -- Une las tablas 'socio' y 'plan' para obtener la información de los planes de cada socio
307 WHERE
308     p.tipo <> 'E' -- Filtra para excluir los planes corporativos ('E' por Empresa)
309 GROUP BY
310     p.plan -- Agrupa los resultados por nombre de plan
311 ORDER BY
312     p.plan; 15ms
313

```

Result(RO) X

Search results Cost: 15ms < 1 > Total 5

	Plan varchar	Número de socios bigint	Cuota mensual varchar	Total varchar
1	24h	12	70.00	840.00
2	Básico mañanas	13	40.00	520.00
3	Básico tardes	8	40.00	320.00
4	Diurno	19	50.00	950.00
5	Joven Emprendedor	12	35.00	420.00

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.