# Guía de ejercicios - Bases de datos relacionales

¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

## ¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

## Tabla de contenidos



**¡Comencemos!**

## Actividad guiada: Simulación de base de datos para gimnasio

En esta actividad realizaremos la simulación de registros en una base de datos para gimnasios. El enfoque será básico, es decir, nos enfocaremos únicamente en registrar clientes y además una tabla donde se almacenen las matrículas de las personas.

Para la tabla de clientes es importante almacenar información primordial como el nombre, apellido, RUT y correo electrónico. En la tabla de matrículas añadiremos los campos de monto y estado. Para el campo estado utilizaremos un tipo de dato boolean. Además, la tabla matrícula estará conectada con clientes a través de un Primary Key, la cual estará definida en el RUT del cliente.

Iniciemos entonces el desarrollo de este ejercicio, esto lo haremos directamente desde la terminal o consola.

- **Paso 1:** Creamos la base de datos con el nombre `bbdd_gimnasios`.

```
create database bbdd_gimnasios;
```

- **Paso 2:** Nos conectamos a la base de datos una vez que haya sido creada.

```
 \c bbdd_gimnasios;
```

- **Paso 3:** Creamos la tabla clientes con los siguientes campos:
  - Nombre
  - Apellido
  - Rut
  - Email

```
create table clientes(nombre varchar(50), apellido varchar(50), rut int,
email varchar(50));
```

- **Paso 4:** Modificamos la tabla clientes y definimos que el rut será la clave primaria (Primary Key)

```
alter table clientes add primary key(rut);
```

- **Paso 5:** Creamos la tabla matrículas con los siguientes campos:
  - Monto
  - Estado
  - Asignamos la clave foránea para la integración de ambas tablas con el Rut.

Considera que para efectos de redacción estamos respetando las reglas gramaticales y acentuamos la palabra matrículas, sin embargo para que no se generen problemas en el código de la sintaxis SQL omite las acentuaciones.

```
create table matriculas(monto varchar(50), estado boolean, cliente_rut
int references clientes(rut));
```

Resultados de ejecución.

```
 Column   |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
nombre    | character varying(50) |           |          |
apellido  | character varying(50) |           |          |
rut       | integer               |           | not null |
email     | character varying(50) |           |          |
```

Imagen 1. Tabla clientes
Fuente: Desafío Latam.

```
  Column    |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
monto       | character varying(50) |           |          |
estado      | boolean               |           |          |
cliente_rut | integer               |           |          |
```

Imagen 2. Tabla matrículas
Fuente: Desafío Latam.

- **Paso 6:** Insertamos 5 registros en la tabla clientes.

```
insert into clientes values ('Cliente 1', 'Apellido cliente 1',
'999999999', 'cliente1@email.com');
insert into clientes values ('Cliente 2', 'Apellido cliente 2',
'888888888', 'cliente2@email.com');
insert into clientes values ('Cliente 3', 'Apellido cliente 3',
'777777777', 'cliente3@email.com');
insert into clientes values ('Cliente 4', 'Apellido cliente 4',
'666666666', 'cliente4@email.com');
insert into clientes values ('Cliente 5', 'Apellido cliente 5',
'555555555', 'cliente5@email.com');
```

- **Paso 7:** Insertamos 5 registros en la tabla matriculas y los asociamos mediante su rut a cada cliente.

```
insert into matriculas values ('40000', True, '999999999');
insert into matriculas values ('40000', False, '888888888');
insert into matriculas values ('55000', True, '555555555');
insert into matriculas values ('35000', True, '777777777');
insert into matriculas values ('60000', False, '666666666');
```

Resultados de la inserción de datos.

```
  nombre   |      apellido       |    rut    |       email
-----------+---------------------+-----------+--------------------
 Cliente 1 | Apellido cliente 1 | 999999999 | cliente1@email.com
 Cliente 2 | Apellido cliente 2 | 888888888 | cliente2@email.com
 Cliente 3 | Apellido cliente 3 | 777777777 | cliente3@email.com
 Cliente 4 | Apellido cliente 4 | 666666666 | cliente4@email.com
 Cliente 5 | Apellido cliente 5 | 555555555 | cliente5@email.com
(5 rows)
```

Imagen 3. Inserción en clientes
Fuente: Desafío Latam

```
 monto | estado | cliente_rut
-------+--------+-------------
 40000 | t      |   999999999
 40000 | f      |   888888888
 55000 | t      |   555555555
 35000 | t      |   777777777
 60000 | f      |   666666666
(5 rows)
```

Imagen 4. Inserción de matrículas
Fuente: Desafío Latam

Hasta este punto ya tenemos la base de datos con sus respectivas tablas y registros. En este punto ya tenemos la opción de realizar algunas operaciones como por ejemplo unificación. Para ello, aplicaremos los conceptos vistos en la sesión anterior acerca de Joins y sus distintos tipos.

- **Paso 8:** Utilizar el inner join para que se muestren todos los registros que estén relacionados. Recordemos que esto se ejecuta siempre que exista una columna que relacione nuestras dos tablas.

```
select email, rut, monto, estado from clientes inner join matriculas on
clientes.rut = cliente_rut;
```

Con esta consulta de SQL estamos realizando las siguientes acciones:

1. Seleccionamos el email y rut de la tabla clientes, además queremos mostrar el monto y el estado del pago almacenado en la tabla matrículas.
2. Hacemos la unión de los datos con inner join y seleccionamos los datos que deseamos unir, en este caso de matrículas.
3. Definimos condicionalmente que si clientes.rut es igual a `cliente_rut` (clave foránea), entonces retorne los resultados.

Si verificamos el resultado en la consola o terminal veremos el siguiente resultado.

```
      email         |    rut    | monto | estado
--------------------+-----------+-------+--------
 cliente1@email.com | 999999999 | 40000 | t
 cliente2@email.com | 888888888 | 40000 | f
 cliente5@email.com | 555555555 | 55000 | t
 cliente3@email.com | 777777777 | 35000 | t
 cliente4@email.com | 666666666 | 60000 | f
(5 rows)
```

Imagen 5. Resultado de inner join
Fuente: Desafío Latam.

- **Paso 9:** Podemos incluso agregar funciones anidadas a la consulta, en este caso supongamos que queremos ordenar según el monto de cada cliente. El orden será de manera ascendente, es decir del monto menor al mayor, para ello utilizaremos `order by`.

```
select email, rut, monto, estado from clientes inner join matriculas on
clientes.rut = cliente_rut order by matriculas.monto;
```

```
      email         |    rut    | monto | estado
--------------------+-----------+-------+--------
 cliente3@email.com | 777777777 | 35000 | t
 cliente1@email.com | 999999999 | 40000 | t
 cliente2@email.com | 888888888 | 40000 | f
 cliente5@email.com | 555555555 | 55000 | t
 cliente4@email.com | 666666666 | 60000 | f
(5 rows)
```

Imagen 6. Orden ascendente de montos.
Fuente: Desafío Latam

- **Paso 10:** Agrupar consultas, para ello añadiremos una matrícula nueva a un RUT ya existente. Esta agrupación consistirá en obtener aquellos clientes que tienen más de una matrícula generada.

```
select monto, count(monto) from matriculas group by monto having
count(monto) >= 2;
```

## Having

La cláusula having nos permite filtrar por el resultado de funciones de agregado. Por ejemplo, usando los datos de la tabla de matrículas podemos filtrar los resultados por aquellos que tienen 2 o más registros.

## Where Vs Having

Una confusión típica es saber cuándo se debe ocupar **where** y cuándo **having**. Having se ocupa cuando la condición es sobre una función de agregado.

Por ejemplo, si queremos filtrar por un correo, o por montos inferiores a cierto valor entonces ocuparemos **where**, si queremos filtrar por la suma o conteo de los valores deberemos ocupar **having**.

- **Paso 11:** Hasta el paso 10 recibimos los datos del registro que tiene dos matrículas asignadas, pero estos datos están incompletos, no sabemos el nombre de la persona, su rut ni el email. Para solucionarlo debemos primero agrupar a partir de aquellos campos que tienen una función de agregado implementada, veamos el código.

```
select email, rut, monto, estado, count(matriculas.*) from clientes
inner join matriculas on clientes.rut = cliente_rut group by email,
monto, rut, estado;
```

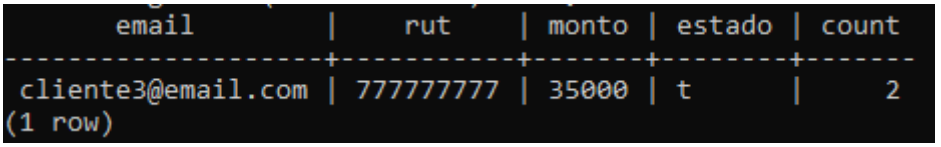
```
      email        |    rut    | monto | estado | count
-------------------+-----------+-------+--------+-------
 cliente4@email.com | 666666666 | 60000 | f      |     1
 cliente3@email.com | 777777777 | 35000 | t      |     2
 cliente2@email.com | 888888888 | 40000 | f      |     1
 cliente5@email.com | 555555555 | 55000 | t      |     1
 cliente1@email.com | 999999999 | 40000 | t      |     1
(5 rows)
```

Imagen 7. Agrupando los datos
Fuente: Desafío Latam

Nótese en la imagen que el cliente con el RUT “777777777” tiene dos matrículas registradas. Supongamos que ahora queremos mostrar únicamente ese registro.

- **Paso 12:** Ya que tenemos la agrupación de los datos ahora podemos usar nuevamente having para obtener únicamente aquellos registros con los datos completos que tengan dos o más matrículas, recuerda la cláusula del `having`.

```
select email, rut, monto, estado, count(matriculas.*) from clientes
inner join matriculas on clientes.rut = cliente_rut group by email,
monto, rut, estado having count(matriculas.*) >= 2;
```



```
       email        |    rut    | monto | estado | count
--------------------+-----------+-------+--------+-------
 cliente3@email.com | 777777777 | 35000 | t      |     2
(1 row)
```

Imagen 8. Cliente con más de 1 matrícula
Fuente: Desafío Latam.

## ¡Manos a la obra! - Agregando integridad referencial

A continuación, deberás agregar el campo correspondiente en la tabla cliente para asegurar la integridad referencial. Recuerda que este campo lo puedes definir como ID, cliente_id, matriculas_id, indistintamente de cómo lo llames lo importante es garantizar la integridad de la información.

Para lograrlo ejecuta los siguientes pasos:

1. Elimina los registros de ambas tablas con `delete from nombre_tabla;`.
2. Utiliza `ALTER` para incorporar el nuevo campo en cada tabla.

Ejemplo

```
ALTER TABLE nombre_tabla(
  ADD COLUMN nombre_campo
)
```

3. Inserta registros en ambas tablas al menos 2.
4. Asigna más de una matrícula a algún cliente y realiza la consulta del paso 12 del ejercicio anterior.
5. Comparte con tus compañeros/as y docente el resultado de tus consultas.