

-- a. Insertar varios registros

```
INSERT INTO tabla (campo1, campo2 ....)
VALUES (campo1, campo2 ....) ,
      (campo1, campo2 ....) ,
      (campo1, campo2 ....) ,
      etc ;
```

-- b. Eliminar de la tabla todos los registros que cumplan con una condición

```
DELETE FROM tabla
WHERE condición ;
```

-- c. Actualizar todos los registros que cumplan con una condición

```
UPDATE tabla
SET campoQueSeActualiza = lo nuevo que quiero poner
WHERE campoDeLaCondicion = condición ;
```

-- d. Constraint dentro de la tabla (SIEMPRE PONERLE NOMBRE A LAS CONSTRAINT)

```
PK: CONSTRAINT tabla_pk PRIMARY KEY (campos) ;
```

```
FK: CONSTRAINT tabla_fk FOREIGN KEY (campo) REFERENCES tabla(campo) ;
```

-- e. Modificación de la tabla

```
COLUMNA: ALTER TABLE tabla ADD COLUMN campo tipoDato restricción ;
          ALTER TABLE tabla DROP COLUMN campo ;
```

```
CAMPO: ALTER TABLE tabla CHANGE campoViejo campoNuevo tipoDato restricción ;
```

```
PK: ALTER TABLE tabla ADD PRIMARY KEY (campos)
    ALTER TABLE tabla DROP PRIMARY KEY
```

```
FK: ALTER TABLE tabla ADD CONSTRAINT tabla_fk FOREIGN KEY (campo) REFERENCES tabla(campo) ;
    ALTER TABLE tabla DROP FOREIGN KEY nombreFK ;
```

-- f. Vistas: Al usarlas sólo se van a poder seleccionar los campos que hayan estado en el select al momento de crearlas

```
CREATE VIEW nombreDeLaVista AS
SELECT .....
```

Usar una vista

```
SELECT campoQueEstáEnLaVista
FROM nombreDeLaVista
```

RECORDAR PARA CONSULTAS:

-- Campo que esté en un intervalo:

Los que están dentro: WHERE campo BETWEEN ____ and ____

Los que están afuera: WHERE campo NOT BETWEEN ____ and ____

-- Campo que es/no es nulo (NO SE USA EL IGUAL):

WHERE campo IS NULL

WHERE campo IS NOT NULL

-- Comparar contra un rango de valores:

WHERE campo IN (__, __, __);

WHERE campo NOT IN (__, __, __);

-- Que no muestre repetidos:

SELECT DISTINCT campo

-- Cuidado si en una tabla puede haber nulos y necesito conservarlos -> LEFT / RIGHT JOIN

-- Consulta con unaria:

-- Listar el nombre de todos los empleados junto al nombre de su jefe (si tiene)

SELECT E.nombre AS 'Empleado' , J.nombre AS 'Jefe'

FROM empleado E

LEFT JOIN empleado J ON J.nro_jefe = E.nro ;

-- Listar los códigos de los materiales que sí provea el proveedor 10 y no los provea el proveedor 15.

select M.cod_mat

from material M

where **exists** (select 1

from provisto_por PP

where PP.cod_mat = M.cod_mat

and PP.cod_prov = 10)

and **not exists** (select 1

from provisto_por PP

where PP.cod_mat = M.cod_mat

and PP.cod_prov = 15);

-- Listar los nombres de proveedores de Capital Federal que sean únicos proveedores de algún material.

select p.nombre

from proveedor P

join provisto_por pv on pv.cod_prov = p.cod_prov

join ciudad c on c.cod_ciu = p.cod_ciu

where c.nombre = 'caba'

and exists(select T.cod_mat

from (select pv.cod_mat, count(*) cantidad

from provisto_por pv

group by pv.cod_mat) T

where **T.cantidad = 1**

and pv.cod_mat = T.cod_mat);

-- Listar los nombres de almacenes que almacenan la mayor cantidad de artículos.

```
select a.nombre
from almacen a
join contiene c on c.nro = a.nro
group by c.nro, a.nombre
having count(*) = (select max(T.cantidad)
                  from (select count(*) cantidad
                        from contiene c
                        group by c.nro) T);
```

-- Hallar los tipos de avión que no son utilizados en algún vuelo que pase por B.

-- traduccion: dame los tipos de aviones que no existen en la (lista de vuelos que pasan por B)

```
select A.tipo_avion
from avion A
where not exists (select 1
                 from vuelo V
                 join avion A1 on V.nro_avion = A1.nro_avion
                 where (V.desde = 'B' or V.hasta = 'B') -- ojo con los paréntesis, sino no da
                 and A1.tipo_avion = A.tipo_avion);
```

-- Listar las ciudades en las que todos los trabajadores que vienen en ellas ganan más de \$1000.

-- traduccion: listar las ciudades en donde nadie ganas 1000 o menos.

-- traducción : listar las ciudades que no existan en (la lista de ciudades donde se gane <= 1000)

```
select distinct P.ciudad
from persona P
where not exists (select P1.ciudad
                 from Persona P1
                 join Trabaja T on T.nro_persona = P1.nro_persona
                 where T.salario <= 1000
                 and P.ciudad = P1.ciudad);
```

-- Listar el nombre de los empleados que hayan ingresado en mas de 4 Empresas en el periodo **01-01-2000 y 31-03-2004**

-- traducción: dame los empleados que tengan fecha entre 01-01-2000 y 31-03-2004 y decime quien aparece más de 4 veces

```
select P.nombre
from persona P
join trabaja T on T.nro_persona = P.nro_persona
where T.fecha_ingreso between '2000-01-01' and '2004-03-31'
group by T.nro_persona, P.nombre
having count(*) > 4;
```

1° WHERE

2° GROUP BY

3° HAVING

LO QUE ESTÁ EN EL SELECT,
TIENE QUE IR AL GROUP BY

EL HAVING SE APLICA SOBRE
LA TABLA AGRUPADA