"El producto más valioso que conozco es la información"

02

TALLER DE BASE DE DATOS - INF 272 SEMESTRE 2/2022

SQL
Structured Query Language
Lenguaje de Consulta Estructurada

Surgió a finales de los 70 cuando apareció todo lo referido al modelo relacional, lenguaje estándar promulgado por ANSI. Lenguaje sencillo en inglés fácil de aprender.

Instrucciones SQL – DML
Data Manipulation Language

Empleadas para SELECCIONAR, INSERTAR, ACTUALIZAR, BORRAR registros en una tabla de una base de datos.

SELECT

INSERT

UPDATE

DELETE

MERGE

Instrucciones SQL – DDL
Data Definition Language

Empleadas para CREAR, MODIFICAR o BORRAR objetos en una base de datos como tablas, vistas, esquemas, dominios, activadores y almacenar procedimientos, usuarios.

CREATE

ALTER

DROP

RENAME

TRUNCATE

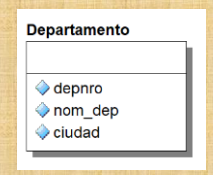
Instrucciones SQL – DCL
Data Control Language

Permite crear y gestionar permisos y accesos a los datos.

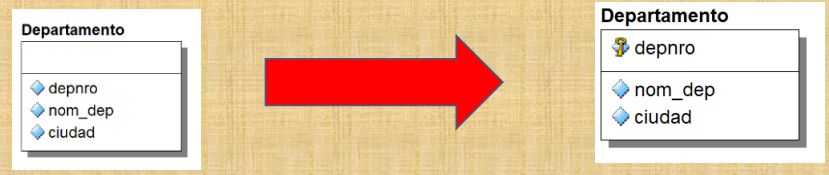
GRANT

REVOKE

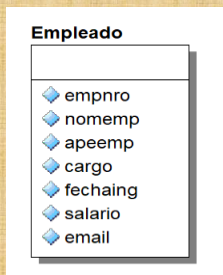
Tenemos a la empresa "PAPELITO S.R.L.", misma que opera en toda Bolivia, la empresa posee en su estructura organizacional varios **departamentos,** cada departamento funciona en diferentes departamentos de Bolivia y cuenta con diferentes **empleados**, de los cuales tenemos sus datos personales, su fecha de ingreso y su salario.



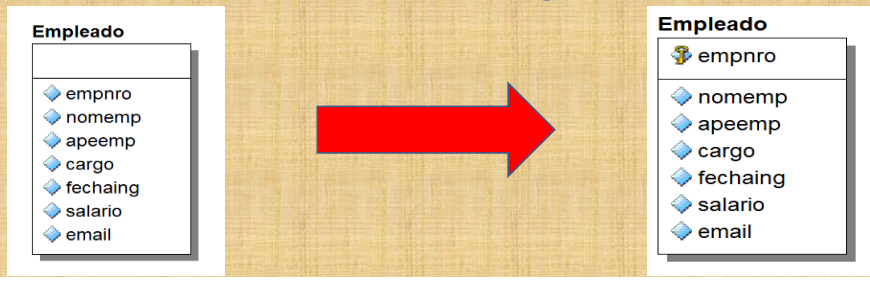
Tenemos a la empresa "PAPELITO S.R.L.", misma que opera en toda Bolivia, la empresa posee en su estructura organizacional varios **departamentos**, cada departamento funciona en diferentes departamentos de Bolivia y cuenta con diferentes **empleados**, de los cuales tenemos sus datos personales, su fecha de ingreso y su salario.



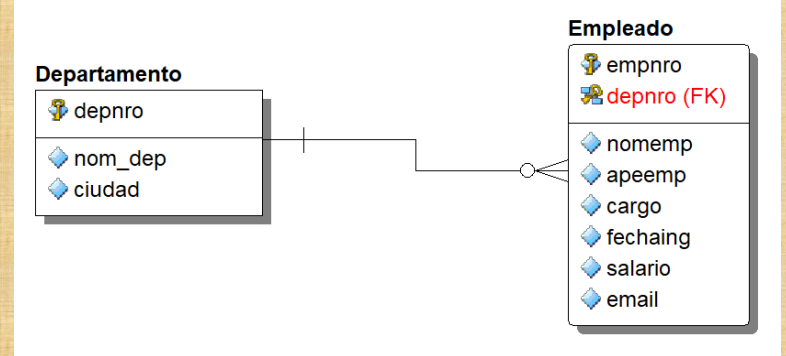
Tenemos a la empresa "PAPELITO S.R.L.", misma que opera en toda Bolivia, la empresa posee en su estructura organizacional varios **departamentos**, cada departamento funciona en diferentes departamentos de Bolivia y cuenta con diferentes **empleados**, de los cuales tenemos sus datos personales, su fecha de ingreso y su salario.



Tenemos a la empresa "PAPELITO S.R.L.", misma que opera en toda Bolivia, la empresa posee en su estructura organizacional varios **departamentos**, cada departamento funciona en diferentes departamentos de Bolivia y cuenta con diferentes **empleados**, de los cuales tenemos sus datos personales, su fecha de ingreso y su salario.



Tenemos a la empresa "PAPELITO S.R.L.", misma que opera en toda Bolivia, la empresa posee en su estructura organizacional varios **departamentos**, cada departamento funciona en diferentes departamentos de Bolivia y cuenta con diferentes **empleados**, de los cuales tenemos sus datos personales, su fecha de ingreso y su salario.



```
CREATE TABLE nombre_tabla
(definición-de-columna   tipo
[,definición de columna ] ……
[, definición-de-clave-primaria]
[, definición-de-clave-foranea] …..);
```

Tipos de Datos

TIPO	DESCRIPCION
CHAR(N)	CADENA DE CARATERES DE LONGITUD FIJA, TIENE UN TAMAÑO DE N BYTES. SINO SE ESPECIFICA EL TAMAÑO ASUME QUE ES 1 BYTE. EL TAMAÑO MAXIMO QUE PUEDE TENER ES DE 2000 BYTES.
VARCHAR2(N)	CADENA DE CARACTERES DE LONGITUD VARIABLE, TIENE UN TAMAÑO MAXIMO DE N BYTES. SE DEBE ESPECIFICAR NECESARIAMENTE EL TAMAÑO. MINIMO 1 Y MAXIMO 32767.

Tipos de Datos

TIPO	DESCRIPCION
NUMBER(P,S)	NUMERO, COMPUESTA POR P DIGITOS DE LOS CUALES S SON DECIMALES. NO ES OBLIGATORIO ESPECIFICAR EL TAMAÑO. P VARIA DE 1 A 38 Y S DE -84 HASTA 127
DATE	TIPO FECHA, QUE EN ORACLE ES DATETIME, CONTIENE FECHA Y HORA HASTA SEGUNDOS. SE USA DESDE EL 1 DE ENERO DE -4712 HASTA 9999.

```
create table DEPARTAMENTOS (
  depnro      number,
  nom_dep     varchar2(50) not null,
  ciudad      varchar2(50) not null,
  PRIMARY KEY (depnro)
);
```

```
create table EMPLEADOS (
  empnro       number,
  nomemp       varchar2(50) not null,
  apeemp       varchar2(50) not null,
  cargo        varchar2(50),
  fecha_ing    date,
  salario      number(8,2),
  depnro       number,
  constraint pk_empleados primary key (empnro),
  constraint fk_emp_dept foreign key (depnro)
     references DEPARTAMENTOS (deptnro)
);
```

INSERT
INTO tabla (campo1, campo2, ...,campo n);
VALUES (valor1, valor2, ...valor n);

```
insert into departamentos (depnro,nom_dep,ciudad)
values (1,'Marketing','Santa Cruz de la Sierra');
```

```
insert into empleados
(empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (1,'Jorge', Choque','Vendedor',TO_DATE('15/01/2022','dd/mm/yyyy'),
2000,'jchoquemorales@gmail.com',1);
```

SELECT (* ó CAMPOS)
FROM tabla;

SELECT DISTINCT (* ó CAMPOS)
FROM tabla;

```
select * from empleados;

select nomemp,apeemp,cargo from empleados;

select cargo from empleados;

select disctinct cargo from empleados;
```

```
create table DEPARTAMENTOS (
  depnro       number,
  nom_dep       varchar2(50) not null,
  ciudad        varchar2(50),
  PRIMARY KEY (depnro)
);
```

```
create table EMPLEADOS (
  empnro          number,
  nomemp          varchar2(50) not null,
  apeemp      varchar2(59) not null,
  cargo           varchar2(50),
  fechaing       date,
  salario         number(8,2),
  email           varchar2(50),
  depnro          number,
  constraint pk_empleados primary key (empnro),
  constraint fk_emp_dept foreign key (depnro)
  references DEPARTAMENTOS(depnro)
);
```

```
insert into departamentos
(depnro,nom_dep,ciudad)
values
(1,'Marketing','Santa Cruz de la Sierra');


insert into empleados
(empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values
(1,'Jorge','Choque','Vendedor',TO_DATE('15/01/2021','dd/mm/yyyy'),2000,'jchoquemorales@gmail.com',1);
```

```
insert into departamentos (depnro,nom_dep,ciudad)
values (1,'Marketing','Santa Cruz de la Sierra');

insert into departamentos (depnro,nom_dep,ciudad)
values (2,'Dirección General CEO','La Paz');

insert into departamentos (depnro,nom_dep,ciudad)
values (3,'Sistemas','Cochabamba');

insert into departamentos (depnro,nom_dep,ciudad)
values (4,'Producción','La Paz');

insert into departamentos (depnro,nom_dep,ciudad)
values (5,'Administración y Contabilidad','Potosí');

insert into departamentos (depnro,nom_dep,ciudad)
values (6,'Control de Calidad','Cochabamba');

insert into departamentos (depnro,nom_dep,ciudad)
values (7,'Recursos Humanos','Oruro');

insert into departamentos (depnro,nom_dep,ciudad)
values (8,'Almacen','Santa Cruz de la Sierra');

insert into departamentos (depnro,nom_dep,ciudad)
values (9,'Investigación y Desarrollo I+D','Tarija');

insert into departamentos (depnro,nom_dep,ciudad)
values (10,'Finanzas y Control de Gestión','Santa Cruz de la Sierra');

insert into departamentos (depnro,nom_dep,ciudad)
values (11,'Logistica','Chuquisaca');
```

```
insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (1,'Jorge','Choque Morales','Vendedor',TO_DATE('15/01/2021','dd/mm/yyyy'),2000,'jchoquemorales@gmail.com',1);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (2,'Grover','Quispe','Analista Programador',TO_DATE('15/01/2019','dd/mm/yyyy'),5000,'groverq@fcpn.edu.bo',3);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (3,'Estefania','Agostopa Machaca','Gerente General',TO_DATE('25/09/2018','dd/mm/yyyy'),20000,'eagostopam@fcpn.edu.bo',2);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (4,'Ramiro','Alvarez Cuaretti','Investigador',TO_DATE('12/02/2020','dd/mm/yyyy'),10000,'halvarezc@fcpn.edu.bo',9);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (5,'Mariana','Condori Apaza','Administrador Base de Datos',TO_DATE('20/05/2020','dd/mm/yyyy'),7000,'mncondori6@umsa.bo',3);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (6,'Leonor','Condori Trujillo','Inventariador',TO_DATE('15/12/2020','dd/mm/yyyy'),4500,'econdorit3@fcpn.edu',8);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (7,'Alexander','Cundir Arratia','Administrador Portal WEB',TO_DATE('31/03/2019','dd/mm/yyyy'),5500,'acundira@fcpn.edu.bo',3);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (8,'Roger','Tancara Suxo','Contador',TO_DATE('5/03/2020','dd/mm/yyyy'),3800,'code01error@gmail.com',5);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (9,'Itati','Torrez Mendez','Vendedor',TO_DATE('18/05/2021','dd/mm/yyyy'),2000,'itorrezm@fcpn.edu.bo',8);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (10,'Andrea','Salas Collanqui','Vendedor',TO_DATE('18/05/2021','dd/mm/yyyy'),2000,'elizabethandreasalasc@gmail.com',8);

insert into empleados (empnro,nomemp,apeemp,cargo,fechaing,salario,email,depnro)
values (12,'Antonio','Laura Lino','Gestor de Expediciones',TO_DATE('18/12/2019','dd/mm/yyyy'),5500,null,11);
```

SQL Worksheet

```
1  SELECT * from departamentos;
```

DEPNRO	NOM_DEP	CIUDAD
1	Marketing	Santa Cruz de la Sierra
2	Dirección General CEO	La Paz
3	Sistemas	Cochabamba
4	Producción	La Paz
5	Administración y Contabilidad	Potosí
6	Control de Calidad	Cochabamba
7	Recursos Humanos	Oruro
8	Almacen	Santa Cruz de la Sierra
9	Investigación y Desarrollo I+D	Tarija
10	Finanzas y Control de Gestión	Santa Cruz de la Sierra
11	Logistica	Chuquisaca
12	Comercial	Chuquisaca

SQL Worksheet

```
1  SELECT * from empleados;
```

EMPNRO	NOMEMP	APEEMP	CARGO	FECHAING	SALARIO	EMAIL	DEPNRO
1	Jorge	Choque Morales	Vendedor	15-JAN-21	2000	jchoquemorales@gmail.com	1
2	Grover	Quispe	Analista Programador	15-JAN-19	5000	groverq@fcpn.edu.bo	3
3	Estefania	Agostopa Machaca	Gerente General	25-SEP-18	20000	eagostopam@fcpn.edu.bo	2
4	Ramiro	Alvarez Cuaretti	Investigador	12-FEB-20	10000	halvarezc@fcpn.edu.bo	9
5	Mariana	Condori Apaza	Administrador Base de Datos	20-MAY-20	7000	mncondori6@umsa.bo	3
6	Leonor	Condori Trujillo	Inventariador	15-DEC-20	4500	econdorit3@fcpn.edu	8
7	Alexander	Cundir Arratia	Administrador Portal WEB	31-MAR-19	5500	acundira@fcpn.edu.bo	3
8	Roger	Tancara Suxo	Contador	05-MAR-20	3800	code01error@gmail.com	5
9	Itati	Torrez Mendez	Vendedor	18-MAY-21	2000	itorrezm@fcpn.edu.bo	8
10	Andrea	Salas Collanqui	Vendedor	18-MAY-21	2000	elizabethandreasalasc@gmail.com	8
12	Antonio	Laura Lino	Gestor de Expediciones	18-DEC-19	5500	-	11
13	Jenny	Ichuta Triguero	Responsable de Campañas	28-JAN-22	3500	-	12
14	Lizeth	Quispe Chambi	Responsable Transformación Digital	01-FEB-20	5500	-	10
15	Abigail	Vargas Blanco	Responsable Egresos	05-FEB-19	4800	svargasb@fcpn.edu.bo	5

Para ver las tablas creadas en la Base de Datos: select * from all_tables;

Para ver la estructura de una tabla creadas: describe departamento;

Para eliminar una tabla: drop table bd_prueba;

Para seleccionar registros:

select * from departamento;

select * from departamento where nom_dep = 'Sistemas';

select nom_dep from departamento;

Operadores

Los operadores son símbolos que permiten realizar operaciones matemáticas, concatenar cadenas, hacer comparaciones.

1) relacionales (o de comparación)

2) aritméticos

3) de concatenación

4) lógicos

Operadores Relacionales

= igual
<> distinto
> mayor
< menor
>= mayor o igual
<= menor o igual

is null
is not null

Operadores Aritméticos y de concatenación

Los operadores aritméticos permiten realizar cálculos con valores numéricos:
multiplicación (*),
división (/),
suma (+)
resta (-).
Es posible obtener salidas en las cuales una columna sea el resultado de un cálculo y no un campo de una tabla.
Operador de concatenación: ||

Funciones String

chr(x): retorna un caracter equivalente al código enviado como argumento "x".
Ejemplo:

```
select chr(65) from dual;          -- retorna 'A'.
select chr(100) from dual;         -- retorna 'd'.
```

concat(cadena1,cadena2): concatena dos cadenas de caracteres; es equivalente al operador ||. Ejemplo:

```
select concat('Buenas',' tardes') from dual;--retorna 'Buenas tardes'.
```

La tabla DUAL es una tabla especial de una sola columna presente de manera predeterminada en todas las instalaciones de bases de datos de Oracle. Se utiliza cuando queremos hacer un select que no necesita consultar tablas. La tabla tiene una sola columna VARCHAR2(1) llamada DUMMY que tiene un valor de 'X'

Funciones String

lower(cadena): retorna la cadena enviada como argumento en minúsculas.

upper(cadena): retorna la cadena con todos los caracteres en mayúsculas.

lpad(cadena,longitud,cadenarelleno): retorna la cantidad de caracteres especificados por el argumento "longitud", de la cadena enviada como primer argumento (comenzando desde el primer caracter).

rpad(cadena,longitud,cadenarelleno): retorna la cantidad de caracteres especificados por el argumento "longitud", de la cadena enviada como primer argumento (comenzando desde el primer caracter).

Funciones String

ltrim(cadena1,cadena2): borra todas las ocurrencias de "cadena2" en "cadena1", si se encuentran al comienzo; si se omite el segundo argumento, se eliminan los espacios.
rtrim(cadena1,cadena2): borra todas las ocurrencias de "cadena2" en "cadena1", si se encuentran por la derecha (al final de la cadena); si se omite el segundo argumento, se borran los espacios.
trim(cadena): retorna la cadena con los espacios de la izquierda y derecha eliminados.
replace(cadena,subcade1,subcade2): retorna la cadena con todas las ocurrencias de la subcadena de reemplazo (subcade2) por la subcadena a reemplazar (subcade1).

Funciones String

length(cadena): retorna la longitud de la cadena enviada como argumento.
instr (cadena,subcadena): devuelve la posición de comienzo (de la primera ocurrencia) de la subcadena especificada en la cadena enviada como primer argumento. Si no la encuentra retorna 0.
translate(cadena, cadena1, cadena2): reemplaza cada ocurrencia de una serie de caracteres **cadena1** con otra serie de caracteres **cadena2**. La diferencia con "**replace**" es que aquella trabaja con cadenas de caracteres y reemplaza una cadena completa por otra, en cambio "**translate**" trabaja con caracteres simples y reemplaza varios.

Funciones Matemáticas

abs(x): retorna el valor absoluto del argumento "x".
ceil(x): redondea a entero, hacia arriba, el argumento "x".
floor(x): redondea a entero, hacia abajo, el argumento "x".
mod(x,y): devuelve el resto de la división x/y.
power(x,y): retorna el valor de "x" elevado a la "y" potencia.
round(n,d): retorna "n" redondeado a "d" decimales; si se omite el segundo argumento, redondea todos los decimales. Si el segundo argumento es positivo, el número de decimales es redondeado según "d"; si es negativo, el número es redondeado desde la parte entera según el valor de "d".

Funciones Matemáticas

sign(x): si el argumento es un valor positivo, retorna 1, si es negativo, devuelve -1 y 0 si es 0.

trunc(n,d): trunca un número a la cantidad de decimales especificada por el segundo argumento. Si se omite el segundo argumento, se truncan todos los decimales. Si "d" es negativo, el número es truncado desde la parte entera.

sqrt(x): devuelve la raíz cuadrada del valor enviado como argumento. **Ejemplo: select sqrt(9) from dual;--retorna 3**

Operadores Lógicos

Son los siguientes:

and, significa "y",

p	q	p ^ q
V	V	V
V	F	F
F	V	F
F	F	F

or, significa "y/o",

p	q	(pvq)
F	F	F
F	V	V
V	F	V
V	V	V

not, significa "no", invierte el resultado

(), paréntesis

Los operadores lógicos se usan para combinar condiciones.

Trabajar con ALIAS

Un ALIAS es un nombre alternativo para una columna cuando se trabaja con la sentencia SELECT.

```
SELECT nomemp,apeemp,salario FROM empleados;
```

```
SELECT nomemp AS NOMBRES,apeemp AS APELLIDOS,salario
FROM empleados;
```

```
SELECT nomemp AS NOMBRES,apeemp AS APELLIDOS,salario,
salario*12 AS “Salario Anual” FROM empleados;
```

Operador de comparación BETWEEN

Otro operador relacional es "between", trabajan con intervalos de valores.

```
select * from empleados where salario between 2000 and 4000

select * from empleados where fechaing between '1-JAN-2020' and '31-MAR-2020';

select * from empleados where nomemp between 'Gabriela' and 'Miriam';
```

IS NULL – IS NOT NULL

Operador relacional para poder ponerlo en la condiocionante y poder seleccionar si el valor de una columna contiene valores nulos o no nulos.

```
SELECT * FROM empleados WHERE email is null;

SELECT * FROM empleados WHERE email is not null;
```

Operador IN

Se utiliza "in" para averiguar si el valor de un campo está incluido en una lista de valores especificada. con "not" antecediendo la condición, invertimos el resultado.

```
SELECT * FROM empleados where depnro IN (3,5);

SELECT * FROM departamentos WHERE ciudad IN ('La Paz', 'Potosí');
```

Operador LIKE

Existe un operador relacional que se usa para realizar comparaciones exclusivamente de cadenas, "like" y "not like".

El símbolo "%" (porcentaje) reemplaza cualquier cantidad de caracteres (incluyendo ningún carácter).
El guion bajo "_" reemplaza un carácter, es otro carácter comodín.

```
SELECT * FROM empleados WHERE nomemp LIKE 'J%';

SELECT * FROM empleados WHERE nomemp LIKE '_i%';

SELECT * FROM empleados WHERE apeemp LIKE '%Mamani%';
```

Cláusula ORDER BY

Permite generar las salidas ordenadas por una o mas columnas en orden ascendente o descendente.

```
SELECT * FROM empleados ORDER BY salario ASC;

SELECT * FROM empleados ORDER BY salario DESC;

SELECT * FROM empleados ORDER BY nomemp,apeemp;

SELECT nomemp,apeemp,fechaing,salario FROM empleados
ORDER BY 3;
```

Cláusula FETCH

Permite limitar el número de registros a ser generadas como resultado de la consulta.

```
SELECT * FROM empleados ORDER BY salario FETCH FIRST 3
ROWS ONLY;

SELECT * FROM empleados ORDER BY salario FETCH FIRST 3
ROWS WITH TIES;

SELECT * FROM empleados ORDER BY salario OFFSET 5 ROWS
FETCH FIRST 3 ROWS ONLY;

SELECT * FROM empleados ORDER BY salario FETCH FIRST 20
PERCENT ROWS ONLY;
```

Funciones de GRUPOS

Las funciones de grupo permiten generar un único resultado de un conjunto de filas. Diferente a las funciones simples que generan un resultado para cada fila.
Se puede agrupar los datos de una tabla para realizar operaciones de grupo (GROUP BY)

```
39  select max(salario) from empleados;
40
```

MAX(SALARIO)
20000

Download CSV

```
43  select depnro,max(salario) from empleados group by depnro;
```

DEPNRO	MAX(SALARIO)
1	2000
2	20000
8	4500
5	3800
9	10000
3	7000

Download CSV
6 rows selected.

Funciones AVG-MAX-MIN

Para averiguar el valor máximo o mínimo de un campo usamos las funciones "max()" y "min()" respectivamente.
La función "avg()" retorna el valor promedio de los valores del campo especificado.
Excluye los valores nulos de los campos.

SELECT AVG(salario) FROM empleados

SELECT AVG(salario),MAX(salario),MIN(salario) FROM empleados;

SELECT AVG(salario),MAX(salario),MIN(salario) FROM empleados WHERE depnro=9;

Funciones COUNT

La función "count()" cuenta la cantidad de registros de una tabla, excluyendo los que tienen valor nulo.

```
SELECT COUNT(*) FROM empleados;

SELECT COUNT(EMAIL) FROM empleados;

SELECT COUNT(DISTINCT DEPNRO) FROM empleados;
```

Funciones SUM

La función "SUM()" permite sumar valores de una columna numérica.

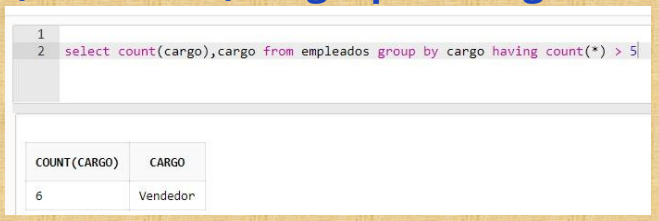
```
SELECT SUM(salario) FROM empleados;

SELECT SUM(salario) FROM empleados WHERE depnro=3;
```

Usamos la clausula "GROUP BY" para agrupar registros para consultas detalladas. Esta siempre se emplea con "SELECT".

Entonces, para saber la cantidad de departamentos que tenemos por cada ubicación, utilizamos la función "count()", agregamos "group by" (que agrupa registros) y el campo por el que deseamos que se realice el agrupamiento.

Así como la cláusula "where" permite seleccionar (o rechazar) registros individuales; la cláusula "having" permite seleccionar (o rechazar) un grupo de registros..



Determinar la cantidad de cargos que tenemos en los empleados, que sean mayores a 5, utilizamos la función "count()", agregamos "group by" (que agrupa registros) y el campo por el que deseamos que se realice el agrupamiento.

No debemos confundir la cláusula "where" con la cláusula "having"; la primera establece condiciones para la selección de registros de un "select"; la segunda establece condiciones para la selección de registros de una salida "group by".