

Introducción al manejo de Base de Datos Comandos Básicos I con DataGrip

1. Revisando Consola de DataGrip
 - a. Insertar 2 registros y mostrar esos valores ingresados.
 - b. Puede crear cualquier tabla.
 - c. Todo esto debe ser trabajador desde Datagrip

```
2.CREATE DATABASE example;

create database example2;

use example;

drop database example2;

create database UNIVERSIDAD;
use UNIVERSIDAD;
CREATE TABLE Estudiantes
(
    CI INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    NOMBRE VARCHAR(40) NOT NULL ,
    APELLIDOS VARCHAR(40) NOT NULL
);

insert into estudiantes (NOMBRE, APELLIDOS) VALUES
('CARLOS', 'Perez');

create database UNIVERSIDAD;
use UNIVERSIDAD;

create table personas
(
```

```

    id persona INTEGER AUTO_INCREMENT not null primary key,
    nombre varchar(20) not null,
    apellidos varchar(30) not null,
    edad integer not null,
    ci varchar(15) not null
);

drop table personas;

insert into personas(nombre, apellidos, edad, ci) VALUES
('Pepito', 'Pepito1', 20, '5664522 LP');

insert into personas(nombre, apellidos, edad, ci) VALUES
('Pepito2', 'Pepito2', 20, '5664522 LP');

insert into personas(nombre, apellidos, edad, ci) VALUES
('Pepito3', 'Pepito3', 20, '5664522 LP');

insert into personas(nombre, apellidos, edad, ci) values
('Pepito', 'Pepito1', 20, '5664522 LP'),
('Pepito2', 'Pepito2', 20, '5664522 LP'),
('Pepito3', 'Pepito3', 20, '5664522 LP');

SELECT per.*
FROM personas AS per;

```

Eliminar la base de datos universidad y sus tablas.

Eliminación de bases de datos	DROP DATABASE IF EXISTS universidad;
Eliminación de tablas	DROP TABLE IF EXISTS materias;

```
DROP DATABASE IF EXISTS universidad;  
DROP TABLE IF EXISTS estudiantes;
```

Creación de una base de datos. (Crear la base de datos universidad)

Desde la interfaz de usuario con DATAGRIP	CREATE DATABASE universidad;
---	------------------------------

```
CREATE DATABASE universidad;  
use universidad;
```

Crear la tabla estudiantes, con las siguientes características.

```
CREATE TABLE estudiantes  
( id_est INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,  
  nombres VARCHAR(100), apellidos VARCHAR(100),  
  edad INTEGER,  
  fono INTEGER,  
  email  
  VARCHAR(50)  
);
```

```
CREATE TABLE estudiantes  
(  
id_est INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,  
nombres VARCHAR(100),  
apellidos VARCHAR(100),  
edad INTEGER,
```

```
fono INTEGER,  
email VARCHAR(50)  
);
```

Agregar nuevos registros a la tabla creada estudiantes en la base de datos universidad.

```
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email) VALUES ('nombre1',  
'apellidos1', 12, 123456, 'nombre1@gmail.com');
```

```
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email) VALUES ('nombre2',  
'apellidos2', 15, 123456, 'nombre2@gmail.com');
```

```
INSERT INTO estudiantes  
(nombres, apellidos, edad, fono, email) VALUES ('nombre3', 'apellidos3', 19, 123456,  
'nombre3@gmail.com');
```

```
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email) VALUES ('nombre1',  
'apellidos1', 12, 123456, 'nombre1@gmail.com');  
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email) VALUES ('nombre2',  
'apellidos2', 15, 123456, 'nombre2@gmail.com');  
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email) VALUES ('nombre3', 'apellidos3', 19,  
123456,  
'nombre3@gmail.com');
```

Seleccionar todos los registros de la tabla estudiantes.

```
SELECT * FROM estudiantes;
```

Output **estudiantes** x

3 rows

	id	÷ nombres	÷ apellidos	÷ edad	÷ fono	÷ email	÷
1	1	nombre1	apellidos1	20	123456	nombre1@gmail.com	
2	2	nombre3	apellidos3	20	123456	nombre3@gmail.com	
3	3	nombre3	apellidos3	20	123456	nombre3@gmail.com	

```
select est.*
from estudiantes as est;
```

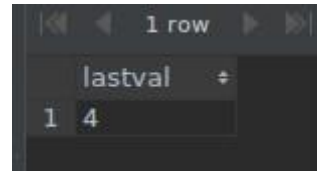
Output **universidad.estudiantes** x

3 rows

	id_est	÷ nombres	÷ apellidos	÷ edad	÷ fono	÷ email
1	1	nombre1	apellidos1	12	123456	nombre1@gn
2	2	nombre2	apellidos2	15	123456	nombre2@gn
3	3	nombre3	apellidos3	19	123456	nombre3@gn

Mostrar el ID del último valor agregado.

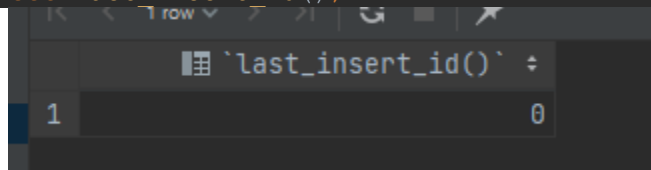
```
select last_insert_id();
```



A screenshot of a database query result. At the top, it says "1 row". Below that, there is a column header "lastval" with a dropdown arrow. The result table has one row with the value "4".

lastval
4

```
select last_insert_id();
```



A screenshot of a database query result. At the top, it says "1 row". Below that, there is a column header "`last_insert_id()`" with a dropdown arrow. The result table has one row with the value "0".

`last_insert_id()`
0

Agregar un nuevo campo a la tabla estudiantes.

```
ALTER TABLE estudiantes  
ADD COLUMN direccion VARCHAR(200);
```

4 rows								
	id	÷ nombres	÷ apellidos	÷ edad	÷ fono	÷ email	÷ direccion	÷
1	1	nombre1	apellidos1	20	123456	nombre1@gmail.com	<null>	
2	2	nombre3	apellidos3	20	123456	nombre3@gmail.com	<null>	
3	3	nombre3	apellidos3	20	123456	nombre3@gmail.com	<null>	
4	4	nombre4	apellidos4	20	123456	nombre3@gmail.com	<null>	

```
ALTER TABLE estudiantes
ADD COLUMN direccion VARCHAR(200);
```

CSV	↓
÷	direccion
	<null>
	<null>
	<null>

Agregar 2 nuevos campos con una sola instrucción.

```
ALTER TABLE estudiantes
ADD COLUMN fax VARCHAR(10),
ADD COLUMN sexo VARCHAR(10);
```

÷ edad	÷ fono	÷ email	÷ direccion	÷ fax	÷ sexo
20	123456	nombre1@gmail.com	<null>	<null>	<null>
20	123456	nombre3@gmail.com	<null>	<null>	<null>
20	123456	nombre3@gmail.com	<null>	<null>	<null>
20	123456	nombre3@gmail.com	<null>	<null>	<null>

```
ALTER TABLE estudiantes
ADD COLUMN fax VARCHAR(10),
ADD COLUMN sexo VARCHAR(10);
```

DDL	CSV	↓	»
÷	÷	÷	÷
<null>	<null>		
<null>	<null>		
<null>	<null>		

```
alter table estudiantes
add column ciudad varchar(50) default 'El alto';
```

÷
El alto
El alto
El alto

Eliminar el campo FAX de la tabla estudiantes.


```
ALTER TABLE estudiantes  
DROP COLUMN fax;
```

nombres	apellidos	edad	fono	email	direccion	sexo
nombre1	apellidos1	20	123456	nombre1@gmail.com	<null>	<null>
nombre3	apellidos3	20	123456	nombre3@gmail.com	<null>	<null>
nombre3	apellidos3	20	123456	nombre3@gmail.com	<null>	<null>
nombre4	apellidos4	20	123456	nombre3@gmail.com	<null>	<null>

```
ALTER TABLE estudiantes  
DROP COLUMN fax;
```

direccion	sexo
<null>	<null>
<null>	<null>
<null>	<null>

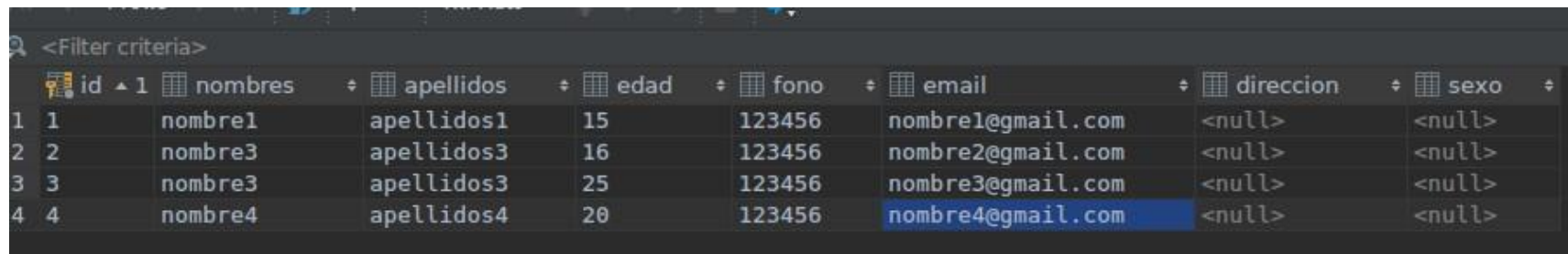
Ejercicios Consultas I

SELECT aqui debe ir lo que quieres mostrar **(CAMPOS)**
FROM aqui debe de ir de donde obtener esos valores **(TABLAS)**
WHERE aqui debe de ir las condiciones **(CONDITIONS)**

SELECT nombre
FROM estudiantes
WHERE estudiantes.nombres != ''

11. Mostrar todos los registros de la tabla estudiantes.

SELECT * FROM estudiantes



	id	nombres	apellidos	edad	fono	email	direccion	sexo
1	1	nombre1	apellidos1	15	123456	nombre1@gmail.com	<null>	<null>
2	2	nombre3	apellidos3	16	123456	nombre2@gmail.com	<null>	<null>
3	3	nombre3	apellidos3	25	123456	nombre3@gmail.com	<null>	<null>
4	4	nombre4	apellidos4	20	123456	nombre4@gmail.com	<null>	<null>

```
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email) VALUES ('nombre1', 'apellidos1', 12, 123456, 'nombre1@gmail.com');
```

	est	nombres	apellidos	edad	fono	email	cuidad
1	1	nombre1	apellidos1	12	123456	nombre1@gmail.com	El alto
2	2	nombre2	apellidos2	15	123456	nombre2@gmail.com	El alto
3	3	nombre3	apellidos3	19	123456	nombre3@gmail.com	El alto
4	4	nombre4	apellidos4	20	123456	nombre4@gmail.com	El alto

Mostrar los registros de aquellos estudiantes que su nombre sea igual a 'nombre4'.

```
SELECT *
```

```
FROM estudiantes
WHERE estudiantes.nombres = 'nombre4';
```

	id	nombres	apellidos	edad	fono	email	direccion	sexo
1	4	nombre4	apellidos4	20	123456	nombre4@gmail.com	<null>	<null>

```
select est.*
from estudiantes as est
where nombres='nombre4';
```

est	nombres	apellidos	edad	fono	email
1	4 nombre4	apellidos4	20	123456	nombre4@gmail.co

Mostrar los registros de los estudiantes donde la edad sea mayor a 18 años.

```
select est.*
from estudiantes as est
where edad>18;
```

est	nombres	apellidos	edad	fono	email
1	3 nombre3	apellidos3	19	123456	nombre3@gmail.co
2	4 nombre4	apellidos4	20	123456	nombre4@gmail.co

Mostrar los registros donde cuyo ID sea PAR. (o IMPAR).

```
select est.*
from estudiantes as est
where id_est%2=0;
```

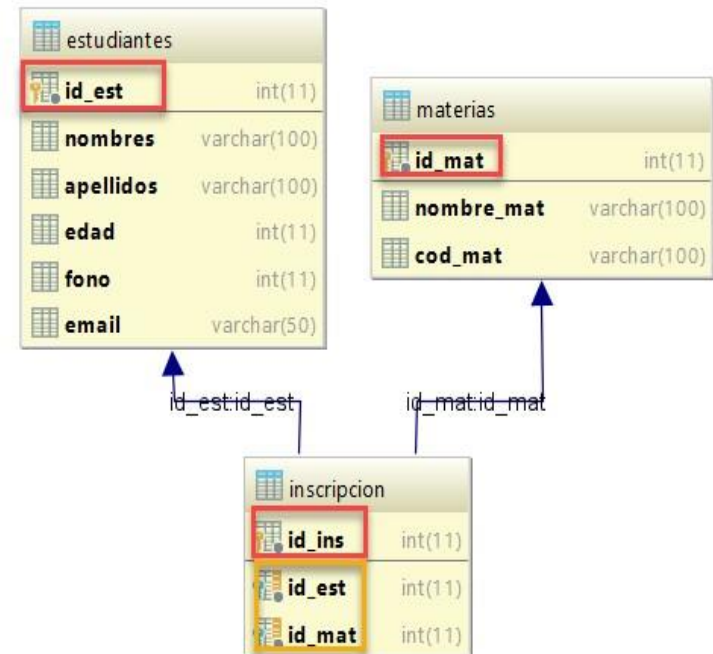
est	nombres	apellidos	edad	fono	email
1	2 nombre2	apellidos2	15	123456	nombre2@gmail.co
2	4 nombre4	apellidos4	20	123456	nombre4@gmail.co

Manejo de Primary Key y ForeignKey

```
CREATE TABLE estudiantes
(
    id_est INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombres VARCHAR(100),
    apellidos VARCHAR(100),
    edad INTEGER,
    fono INTEGER,
    email VARCHAR(50)
);

CREATE TABLE materias
(
    id_mat INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    nombre_mat VARCHAR(100),
    cod_mat VARCHAR(100)
);

CREATE TABLE inscripcion
(
    id_ins INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
    id_est INT NOT NULL,
    id_mat INT NOT NULL,
    FOREIGN KEY (id_est) REFERENCES estudiantes (id_est),
    FOREIGN KEY (id_mat) REFERENCES materias (id_mat)
);
```



Crear la tabla estudiantes.	<pre>CREATE TABLE estudiantes (</pre>
	<pre> id_est INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL, nombres VARCHAR(100), apellidos VARCHAR(100), edad INTEGER, fono INTEGER, email VARCHAR(50));</pre>
Crear la tabla materias	<pre>CREATE TABLE materias (id_mat INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL, nombre_mat VARCHAR(100), cod_mat VARCHAR(100));</pre>
Crear la tabla que relaciona a los 2.	<pre>CREATE TABLE inscripcion (id_ins INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL, id_est INT NOT NULL, id_mat INT NOT NULL, FOREIGN KEY (id_est) REFERENCES estudiantes (id_est), FOREIGN KEY (id_mat) REFERENCES materias (id_mat));</pre>

```

CREATE TABLE estudiantes
(
id_est INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
nombres VARCHAR(100),
apellidos VARCHAR(100),
edad INTEGER,
fono INTEGER,
email VARCHAR(50)
);
CREATE TABLE materias
(
id_mat INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
nombre_mat VARCHAR(100),
cod_mat VARCHAR(100)
);
CREATE TABLE inscripcion
(
id_ins INTEGER AUTO_INCREMENT PRIMARY KEY NOT NULL,
id_est INT NOT NULL,
id_mat INT NOT NULL,
FOREIGN KEY (id_est) REFERENCES estudiantes (id_est),
FOREIGN KEY (id_mat) REFERENCES materias (id_mat)
);

```

Generar la siguiente tabla utilizando la interfaz gráfica de Datagrip.

```

create table ejemplo
(
id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
nombre varchar(20) NOT NULL,
descripcion varchar(100) NOT NULL
);

```

Create New Table

Table:

table_name

Comment:

Columns

Keys

Indices

Foreign Keys

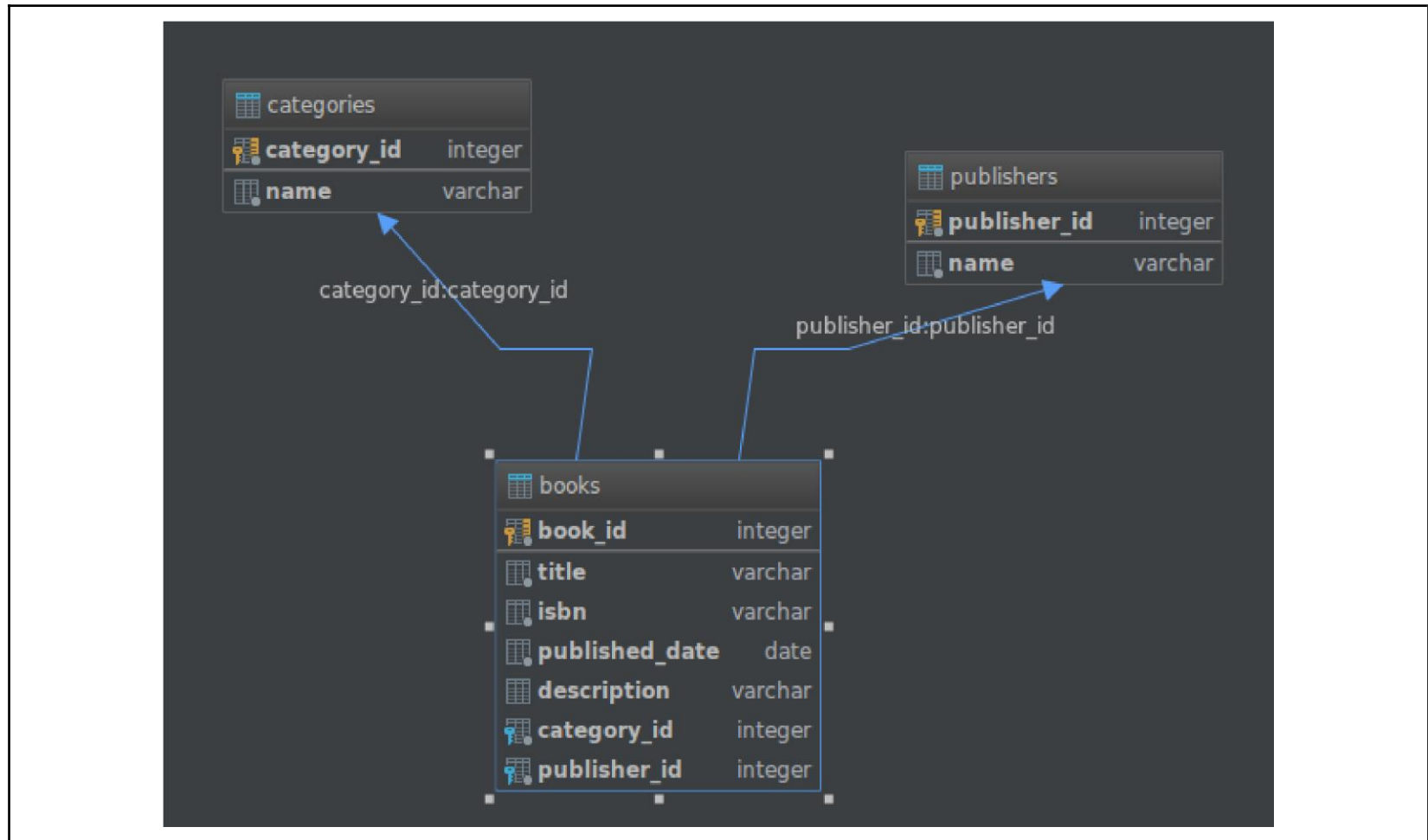
No columns

```
create table ejemplo
(
id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
nombre varchar(20) NOT NULL,
descripcion varchar(100) NOT NULL
);
```

ejemplo	
id	int(11)
nombre	varchar(20)
descripcion	varchar(100)

Generar las tablas de acuerdo a la siguiente imagen.

- Crear una base de datos denominada Librería.
- Hacer uso de esa base de datos.



Soluciones.

```
create database libreria;
use libreria;

create table categorias(
    category_id integer auto_increment primary key,
    name varchar(80) not null
);

create table publishers(
    publisher_id integer auto_increment primary key,
    name varchar(80) not null
);




create table books(
    book_id integer auto_increment primary key,
    title varchar(80) not null,
    isbn varchar(1000) not null,
    published_date date not null,
    description varchar(7000) not null,
    category_id integer not null,
    publisher_id integer not null,
    foreign key (category_id) references categorias(category_id),
    foreign key (publisher_id) references publishers(publisher_id)
);
```

Pregunta de Diseño de Base Datos.




- Dado un escenario de empleados que trabajan en departamentos dentro de una empresa, como debería ser su sistema de base de datos relacional.
 - ***El objetivo es saber en qué empresa y en qué área trabaja una persona.***
 - Debe de crear una base de datos relacional denominada EMPRESA.
 - Debería de crear 3 tablas mínimamente. (Empleado - Empresa - Area).

```
create table empresa(  
  id_emp integer auto_increment primary key not null,  
  nombre_emp varchar(50) not null,  
  tipo_emp varchar(50) not null  
);  
  
create table area(  
  id_area integer auto_increment primary key not null,  
  nombre_area varchar(50) not null,  
  empresa varchar(50) not null  
);  
  
create table empleado(  
  id_empleado integer auto_increment primary key not null,  
  nombres_empleado varchar(40) not null,  
  apellidos_empleado varchar(50) not null,  
  nombre_emp varchar(50) not null,  
  nombre_area varchar(50) not null,  
  id_emp integer not null,  
  id_area integer not null,  
  foreign key (id_emp) references empresa(id_emp),
```

```
foreign key (id_area) references area(id_area)
);
```

WHERE		ORDER BY	
	 id_area	 nombre_area	 empresa
1	1	Markenting	C&R
2	2	Produccion	Coca Cola

WHERE		ORDER BY			
	id_empleado	nombres_empleado	apellidos_empleado	nombre_emp	nombi
1	1	Adrian	Fernandez	C&R	Marketi
2	2	Carla	Cordoba	Coca Cola	Producc

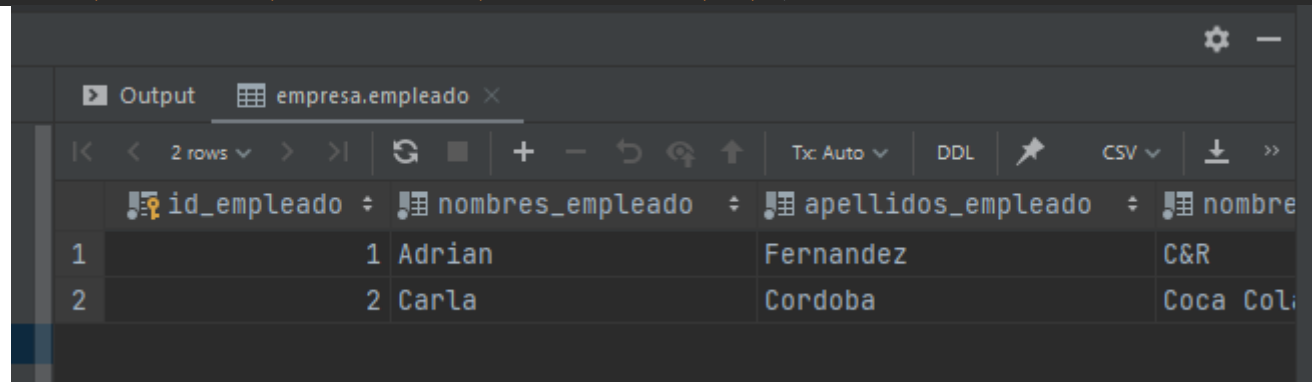
WHERE		ORDER BY	
	 id_emp	 nombre_emp	 tipo_emp
1	1	C&R	Entretenimiento
2	2	Coca Cola	Gaseosas

```
insert into empresa( nombre_emp, tipo_emp) values ('C&R', 'Entretenimiento'),
('Coca Cola', 'Gaseosas');

insert into area(nombre_area, empresa) values('Markenting', 'C&R'),
('Produccion', 'Coca Cola');

insert into empleado(nombres_empleado, apellidos_empleado, nombre_emp, nombre_area, id_emp, id_area)
```

```
values
('Adrian', 'Fernandez', 'C&R', 'Marketing', 1,1),
('Carla', 'Cordoba', 'Coca Cola', 'Produccion', 2,2);
```



	id_empleado	nombres_empleado	apellidos_empleado	nombre_empresa	departamento
1	1	Adrian	Fernandez	C&R	Marketing
2	2	Carla	Cordoba	Coca Cola	Produccion

```
select *
from empleado;
```