

Sintaxis SQL

1. Crear la tabla **estudiantes**, con las siguientes características.

La tabla debería de tener los siguientes campos.

ci => cadena de 12 y ademas llave primaria

nombres => una cadena de 20

apellidos => una cadena de 40

edad => un valor numérico

fono => un valor numérico

email => una cadena de 100

```
DROP TABLE IF EXISTS estudiantes;
```

```
CREATE TABLE estudiantes
```

```
(
```

```
  ci VARCHAR(12) PRIMARY KEY NOT NULL,
```

```
  nombres VARCHAR(20),
```

```
  apellidos VARCHAR(40),
```

```
  edad INTEGER,
```

```
  fono INTEGER,
```

```
  email VARCHAR(100)
```

```
);
```

2. Insertando a un estudiante con el correo electrónico(campo email) vacío.

- a. Notemos que es lo que pasa cuando mandamos un campo vacío

```
INSERT INTO estudiantes (ci, nombres, apellidos, edad, fono)
VALUES ('7886733SC', 'nombre1', 'apellidos1', 20, 78789898);
```

ci	nombres	apellidos	edad	fono	email
7886733SC	nombre1	apellidos1	20	78789898	<null>

El campo **email** está con un valor **null**, eso ocurre cuando en el diseño de la tabla ese campo admite valores nulos.

3. Rediseñando la tabla **estudiante**.

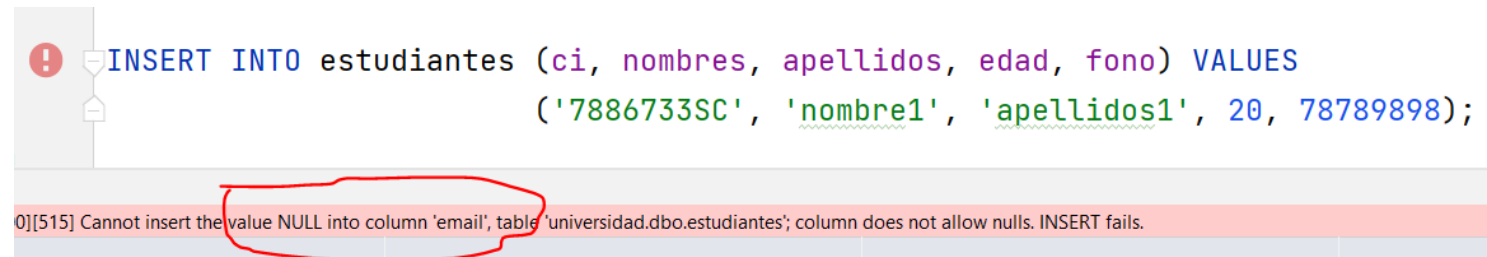
- Se agregara NOT NULL a todos los campos.
- Con esto evitaremos valores vacíos o nulos.
- Eliminar la tala estudiante para crear nuevamente

```
CREATE TABLE estudiantes
(
  ci VARCHAR(12) PRIMARY KEY NOT NULL,
  nombres VARCHAR(100) NOT NULL,
  apellidos VARCHAR(100) NOT NULL,
  edad INTEGER NOT NULL,
  fono INTEGER NOT NULL,
  email VARCHAR(100) NOT NULL
);
```

- Intentemos insertar nuevamente el registro insertado a anteriormente

```
INSERT INTO estudiantes (ci, nombres, apellidos, edad, fono)
VALUES ('7886733SC', 'nombre1', 'apellidos1', 20, 78789898);
```

Se generará un error.



- e. Eliminando todos los registros de la tabla **estudiantes** para agregar nuevos

```
TRUNCATE TABLE estudiantes;
```

Cuando ejecutamos **TRUNCATE** elimina todos los registros de la tabla.

Nota. Solo elimina los registros y no así la tabla

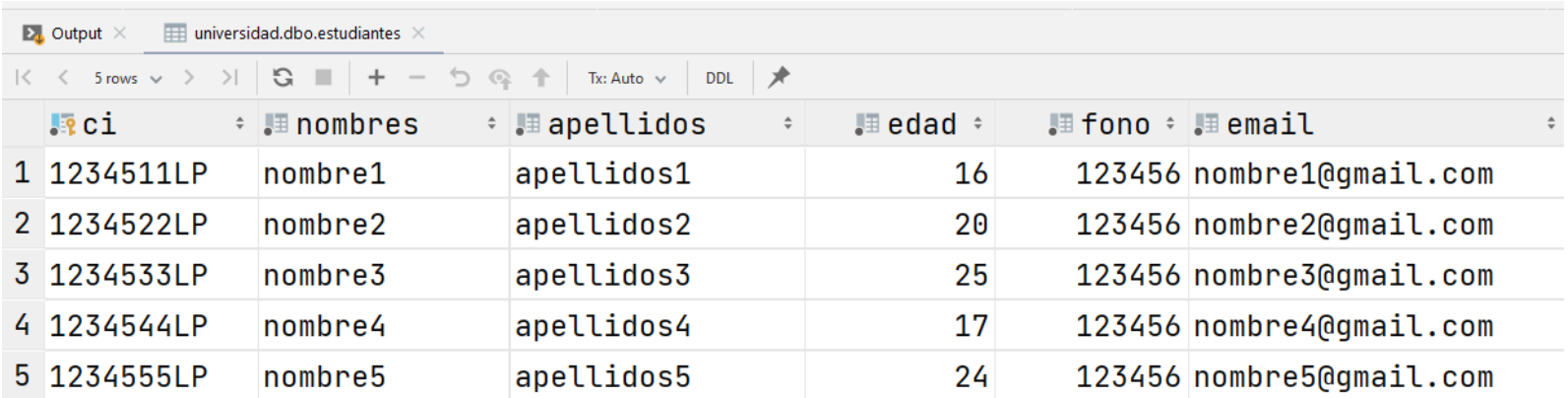
4. Agregar nuevos registros a la tabla **estudiantes** en la base de datos universidad.

```
INSERT INTO estudiantes (ci, nombres, apellidos, edad, fono, email)
VALUES ('1234511LP', 'nombre1', 'apellidos1', 16, 123456, 'nombre1@gmail.com');
INSERT INTO estudiantes (ci, nombres, apellidos, edad, fono, email)
VALUES ('1234522LP', 'nombre2', 'apellidos2', 20, 123456, 'nombre2@gmail.com');
INSERT INTO estudiantes (ci, nombres, apellidos, edad, fono, email)
```

```
VALUES ('1234533LP', 'nombre3', 'apellidos3', 25, 123456, 'nombre3@gmail.com');
INSERT INTO estudiantes (ci, nombres, apellidos, edad, fono, email)
VALUES ('1234544LP', 'nombre4', 'apellidos4', 17, 123456, 'nombre4@gmail.com');
INSERT INTO estudiantes (ci, nombres, apellidos, edad, fono, email)
VALUES ('1234555LP', 'nombre5', 'apellidos5', 24, 123456, 'nombre5@gmail.com');
```

5. Seleccionar todos los registros de la tabla estudiantes.

```
SELECT * FROM estudiantes;
```



	ci	nombres	apellidos	edad	fono	email
1	1234511LP	nombre1	apellidos1	16	123456	nombre1@gmail.com
2	1234522LP	nombre2	apellidos2	20	123456	nombre2@gmail.com
3	1234533LP	nombre3	apellidos3	25	123456	nombre3@gmail.com
4	1234544LP	nombre4	apellidos4	17	123456	nombre4@gmail.com
5	1234555LP	nombre5	apellidos5	24	123456	nombre5@gmail.com

6. Agregar un nuevo campo a la tabla estudiantes.

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

Output x universidad.dbo.estudiantes x

5 rows

	ci	÷	nombres	÷	apellidos	÷	edad	÷	fono	÷	email	÷	direccion
1	1234511LP		nombre1		apellidos1		16		123456		nombre1@gmail.com		<null>
2	1234522LP		nombre2		apellidos2		20		123456		nombre2@gmail.com		<null>
3	1234533LP		nombre3		apellidos3		25		123456		nombre3@gmail.com		<null>
4	1234544LP		nombre4		apellidos4		17		123456		nombre4@gmail.com		<null>
5	1234555LP		nombre5		apellidos5		24		123456		nombre5@gmail.com		<null>

7. Agregar 2 nuevos campos con una sola instrucción.

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

DDL

	edad	÷	fono	÷	email	÷	direccion	÷	fax	÷	sexo
	16		123456		nombre1@gmail.com		<null>		<null>		<null>
	20		123456		nombre2@gmail.com		<null>		<null>		<null>
	25		123456		nombre3@gmail.com		<null>		<null>		<null>

8. Modificar el tipo de un campo que ya existe.

a. Es decir vamos a modificar el tipo de un campo que ya existe.

- b. En este caso modificaremos el campo de nombre SEXO.
- c. Inicialmente era de tipo VARCHAR(10), ahora sera un char(1). Es decir una sola letra

```
ALTER TABLE estudiantes
  ALTER COLUMN sexo CHAR(1);
```

```
tax      varchar(10),
sexo     char
```

9. Eliminar el campo FAX de la tabla estudiantes.

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

nombres	apellidos	edad	fono	email	direccion	sexo
nombre1	apellidos1	16	123456	nombre1@gmail.com	<null>	<null>
nombre2	apellidos2	20	123456	nombre2@gmail.com	<null>	<null>
nombre3	apellidos3	25	123456	nombre3@gmail.com	<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 != "

10. Mostrar todos los registros de la tabla estudiantes.

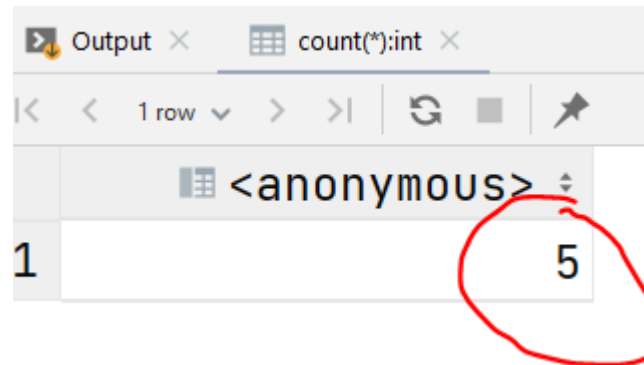
```
SELECT * FROM estudiantes;
```

ci	nombres	apellidos	edad	fono	email	direccion	sexo
1 1234511LP	nombre1	apellidos1	16	123456	nombre1@gmail.com	<null>	<null>
2 1234522LP	nombre2	apellidos2	20	123456	nombre2@gmail.com	<null>	<null>
3 1234533LP	nombre3	apellidos3	25	123456	nombre3@gmail.com	<null>	<null>
4 1234544LP	nombre4	apellidos4	17	123456	nombre4@gmail.com	<null>	<null>
5 1234555LP	nombre5	apellidos5	24	123456	nombre5@gmail.com	<null>	<null>

11. Determinar cuántos registros hay en la tabla estudiantes.

```
SELECT count(*)  
FROM estudiantes;
```

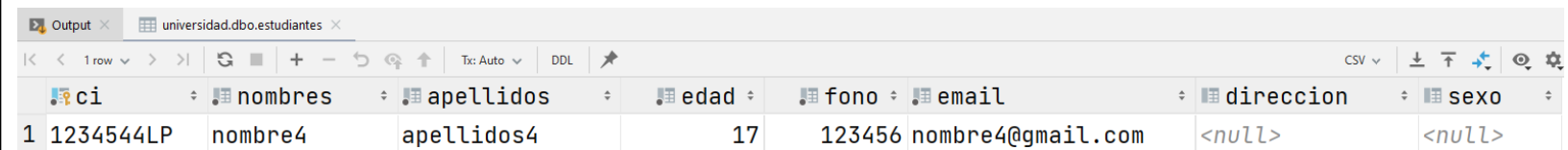
#user primary key para contar



	<anonymous>
1	5

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

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



	ci	nombres	apellidos	edad	fono	email	direccion	sexo
1	1234544LP	nombre4	apellidos4	17	123456	nombre4@gmail.com	<null>	<null>

13. Mostrar los registros cuya edad sea igual a 15 o igual a 20.

```
SELECT *
FROM estudiantes
WHERE estudiantes.edad = 15 OR estudiantes.edad = 20;
```

Output × universidad.dbo.estudiantes ×

1 row

Tx: Auto

DDL

CSV

↓

ci	nombres	apellidos	edad	fono	email	direccion
1 1234522LP	nombre2	apellidos2	20	123456	nombre2@gmail.com	<null>

14. Mostrar los registros cuya edad sea mayor a 18.

	Nombre Completo
1	Kevin Michel
2	
3	
4	
5	

15. Mostrar aquellos registros en donde el campo SEXO sea vacío o nulo.

	Nombre Completo
1	
2	
3	
4	
5	

16. Mostrar los registros en donde cuya EDAD sea PAR. (o IMPAR).

a. Para el caso de los PARES debería mostrar

	ci	nombres	apellidos	edad	fono	email	direccion
1	1234511LP	nombre1	apellidos1	16	123456	nombre1@gmail.com	<null>
2	1234522LP	nombre2	apellidos2	20	123456	nombre2@gmail.com	<null>
3	1234555LP	nombre5	apellidos5	24	123456	nombre5@gmail.com	<null>

b. Para el caso de los IMPARES debería mostrar

	ci	nombres	apellidos	edad	fono	email	direccion	sexo
1	1234533LP	nombre3	apellidos3	25	123456	nombre3@gmail.com	<null>	<null>
2	1234544LP	nombre4	apellidos4	17	123456	nombre4@gmail.com	<null>	<null>

	Nombre Completo
1	
2	
3	
4	
5	

Manejo de Primary Key y Foreign Key

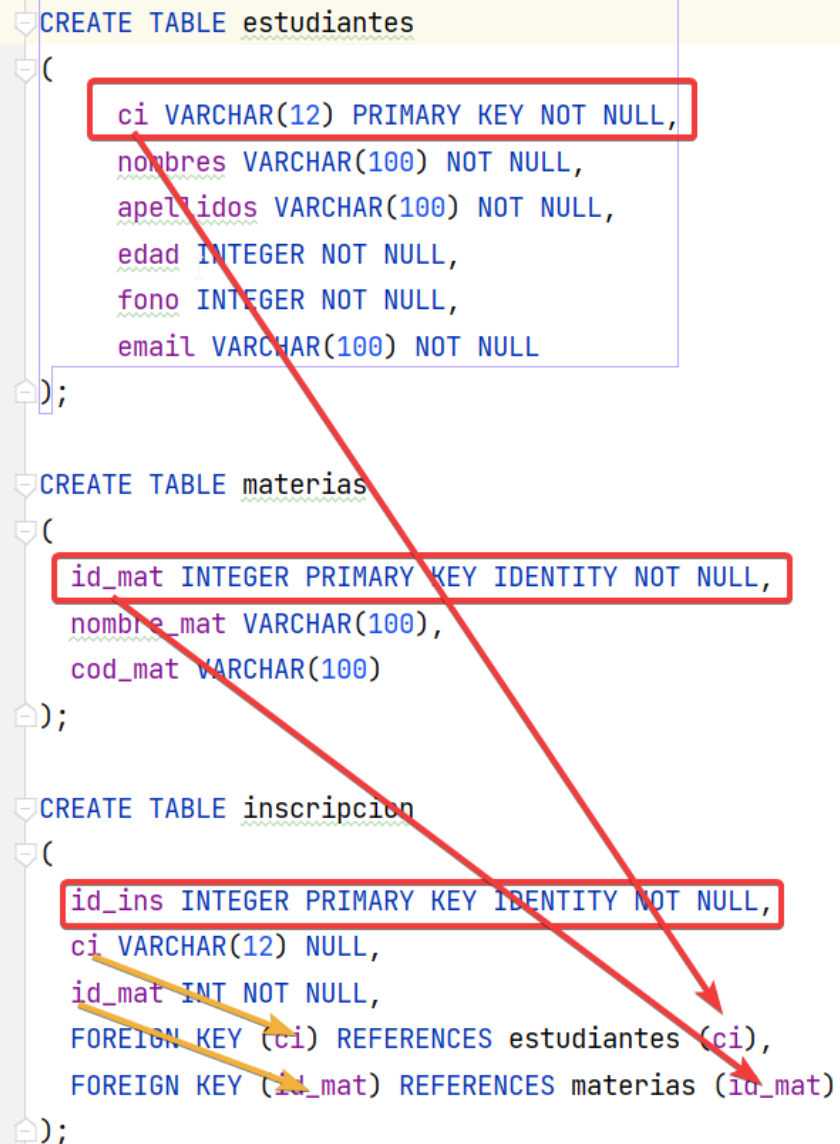
17. Crear las siguientes tablas.

Crear la tabla estudiantes.	<pre>CREATE TABLE estudiantes (ci VARCHAR(12) PRIMARY KEY NOT NULL, nombres VARCHAR(100) NOT NULL, apellidos VARCHAR(100) NOT NULL, edad INTEGER NOT NULL, fono INTEGER NOT NULL, email VARCHAR(100) NOT NULL);</pre>
Crear la tabla materias	<pre>CREATE TABLE materias (id_mat INTEGER PRIMARY KEY IDENTITY 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 PRIMARY KEY IDENTITY NOT NULL, ci VARCHAR(12) NULL, id_mat INT NOT NULL, FOREIGN KEY (ci) REFERENCES estudiantes (ci), FOREIGN KEY (id_mat) REFERENCES materias (id_mat));</pre>

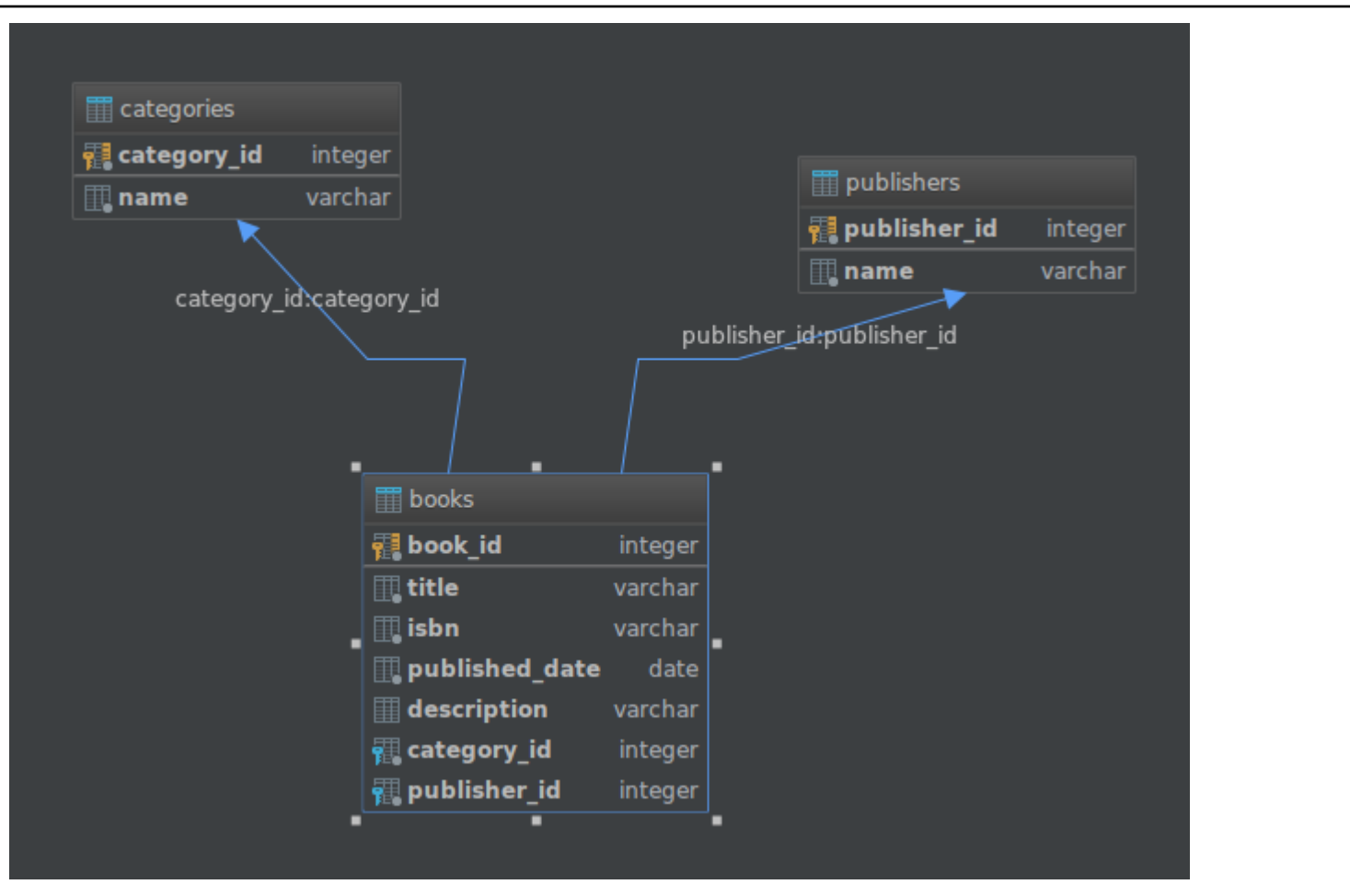
```
CREATE TABLE estudiantes
(
    ci VARCHAR(12) PRIMARY KEY NOT NULL,
    nombres VARCHAR(100) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
    edad INTEGER NOT NULL,
    fono INTEGER NOT NULL,
    email VARCHAR(100) NOT NULL
);

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

CREATE TABLE inscripcion
(
    id_ins INTEGER PRIMARY KEY IDENTITY NOT NULL,
    ci VARCHAR(12) NULL,
    id_mat INT NOT NULL,
    FOREIGN KEY (ci) REFERENCES estudiantes (ci),
    FOREIGN KEY (id_mat) REFERENCES materias (id_mat)
);
```



18. Generar las tablas de acuerdo a la siguiente imagen.
- Crear una base de datos para el esquema dado.
 - Crear mínimamente un registro por cada tabla.



	Nombre Completo
1	
2	
3	
4	
5	
6	
7	

Pregunta de Diseño de Base Datos.

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

Sugerencias.

- Crear una nueva consola para este ejercicio.
- Agregar mínimamente un registro a las tablas.

	Nombre Completo	Diseño	Query
1			
2			
3			
4			
5			
6			

