



**BASE DE DATOS II
TAREA HITO 4**

**ESTUDIANTE: BRITTANY IBLING MARINO
QUISPE**

CODIGO: SIS13181271

DOCENTE: LIC. WILLIAM BARRA PAREDES

CARRERA: INGENIERIA DE SISTEMAS

SEMESTRE 3

FECHA: 15-06-2022

MANEJO DE CONCEPTOS

1.- Defina que es lenguaje procedural en MySQL.

R.- El usuario da órdenes para que se realicen las tareas pertinentes con el objetivo de recuperar los datos requeridos. Es la base del lenguaje de consulta SQL.

2.- Defina que es una FUNCTION en MySQL.

R.- Las funciones son piezas de código que reciben datos de entrada, realizan operaciones con ellos y luego devuelven un resultado.

Mediante las funciones que nos provee el motor o mediante funciones definidas por nosotros, podemos también manipular los datos antes de recuperarlos o guardarlos.

3.-Cuál es la diferencia entre funciones y procedimientos almacenados.

R.- Cuando llama al procedimiento almacenado, se debe especificar que es un parámetro externo. Una ventaja de los procedimientos almacenados es que puede obtener varios parámetros mientras que, en las funciones, solo se puede devolver una variable o una tabla.

4.- Cómo se ejecuta una función y un procedimiento almacenado.

R.- Hay dos formas diferentes de ejecutar un procedimiento almacenado. El primer método y más común es que una aplicación o un usuario llame al procedimiento. El segundo método consiste en establecer el procedimiento para que se ejecute automáticamente cuando se inicie una instancia de SQL Server .

Una función acepta entradas en forma de parámetros y devuelve un valor. SQL Server viene con un conjunto de funciones integradas que realizan una variedad de tareas.

5.- Defina que es una TRIGGER en MySQL.

R.- Un trigger, también conocido como disparador (Por su traducción al español) es un conjunto de sentencias SQL las cuales se ejecutan de forma automática cuando ocurre algún evento que modifique a una tabla.

6.- En un trigger que papel juega las variables OLD y NEW

R.- En un trigger disparado por un "insert", se puede acceder al campo ":new" unicamente, el campo ":old" contiene "null". En una inserción se puede emplear ":new" para escribir nuevos valores en las columnas de la tabla. En un trigger que se dispara con "update", se puede acceder a ambos campos. En una actualización, se pueden comparar los valores de ":new" y ":old".

7.- En un trigger que papel juega los conceptos(cláusulas) BEFORE o AFTER

R.- Puede ser BEFORE (antes) o AFTER (después), para indicar que el disparador se ejecute antes o después que la sentencia que lo activa.

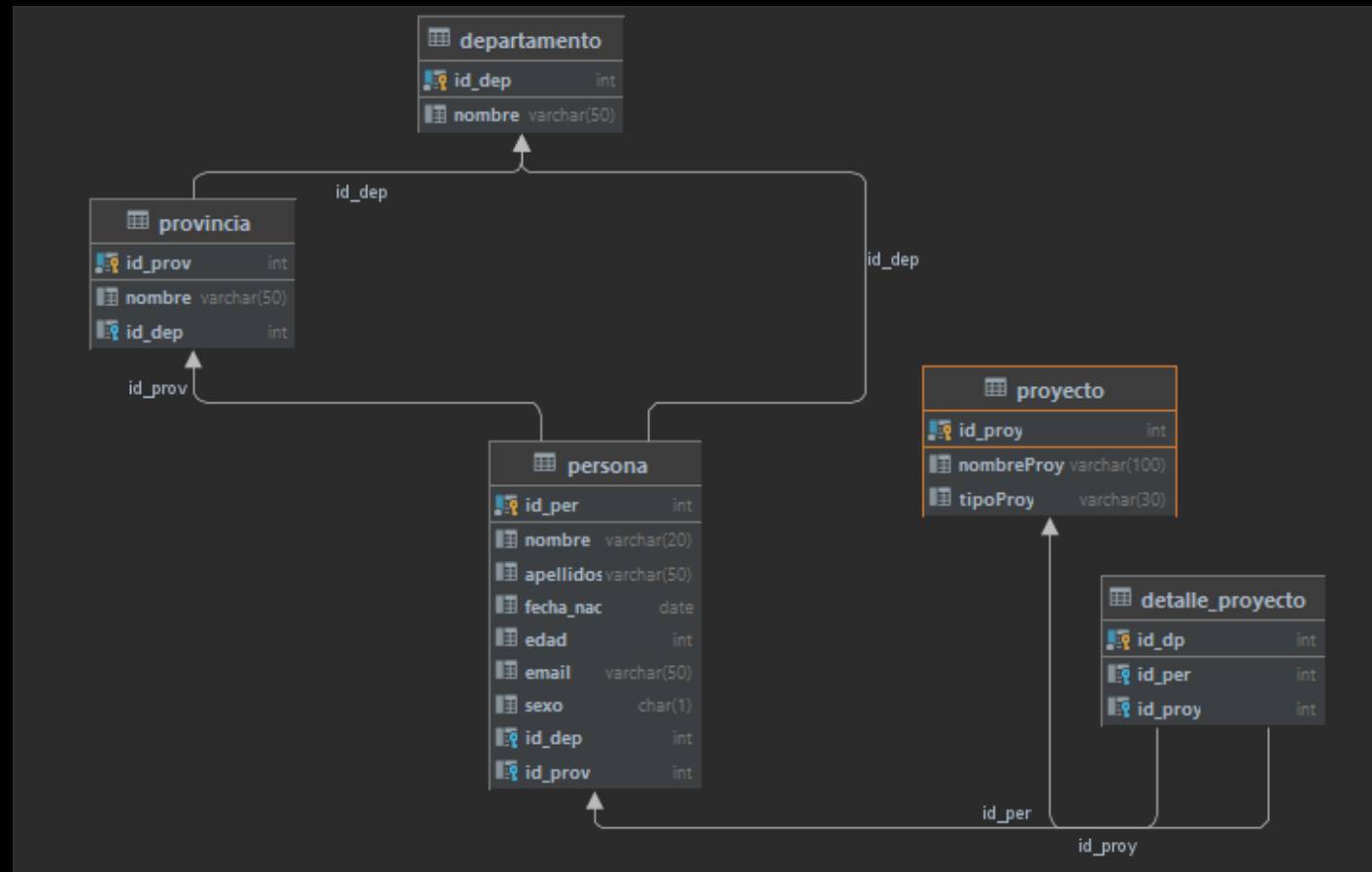
El modificador BEFORE o AFTER indica que el trigger se ejecutará antes o después de ejecutarse la sentencia SQL definida por DELETE, INSERT o UPDATE.

8.- A que se refiere cuando se habla de eventos en TRIGGERS

R.- Un "trigger", es un bloque de código que se ejecuta automáticamente cuando ocurre algún evento sobre una determinada tabla; es decir, cuando se intenta modificar los datos de una tabla asociada al disparador.

PARTE PRACTICA

9.- Crear la siguiente Base de datos y sus registros.




```

1  create database procesual_hito_4
2  use procesual_hito_4
3
4  create table proyecto(
5      id_proy INT IDENTITY(1,1) PRIMARY KEY,
6      nombreProy varchar(100),
7      tipoProy varchar(30)
8  )
9  |
10 create table departamento(
11     id_dep INT IDENTITY(1,1) PRIMARY KEY,
12     nombre varchar(50)
13 )
14
15 create table provincia(
16     id_prov INT IDENTITY(1,1) PRIMARY KEY,
17     nombre varchar(50),
18     id_dep int,
19     foreign key (id_dep) references departamento(id_dep)
20 )

```

```

22 create table persona(
23     id_per INT IDENTITY(1,1) PRIMARY KEY,
24     nombre varchar(20),
25     apellidos varchar(50),
26     fecha_nac date,
27     edad int,
28     email varchar(50),
29     sexo char,
30     id_dep int,
31     id_prov int,
32     foreign key (id_dep) references departamento(id_dep),
33     foreign key (id_prov) references provincia(id_prov)
34 )
35
36 create table detalle_proyecto(
37     id_dp INT IDENTITY(1,1) PRIMARY KEY,
38     id_per int,
39     id_proy int,
40     foreign key (id_per) references persona(id_per),
41     foreign key (id_proy) references proyecto(id_proy)
42 )

```

```

44 insert into proyecto(nombreProy, tipoProy)
45 values ('cargador inalambrico', 'Robotica');
46 insert into proyecto(nombreProy, tipoProy)
47 values ('crecimiento de plantas', 'Forestacion');
48 insert into proyecto(nombreProy, tipoProy)
49 values ('imagen al futuro', 'Arte');
50 insert into proyecto(nombreProy, tipoProy)
51 values ('robot que juegan ajedrez', 'Deporte');
52
53 insert into departamento(nombre)
54 values ('La Paz');
55 insert into departamento(nombre)
56 values ('Cochabamba');
57 insert into departamento(nombre)
58 values ('Tarija');
59 insert into departamento(nombre)
60 values ('pando');

```

```

61
62 insert into provincia(nombre, id_dep)
63 values ('Nombre1', 1);
64 insert into provincia(nombre, id_dep)
65 values ('Nombre2', 2);
66 insert into provincia(nombre, id_dep)
67 values ('Nombre3', 3);
68 insert into provincia(nombre, id_dep)
69 values ('Nombre4', 4);
70
71 insert into persona(nombre, apellidos, fecha_nac, edad, email, sexo, id_dep, id_prov)
72 values ('Abel', 'Lopez', '2000-10-10', 22, 'abel@gmail.com', 'M', 1, 1);
73 insert into persona(nombre, apellidos, fecha_nac, edad, email, sexo, id_dep, id_prov)
74 values ('Andres', 'Zavaleta', '2001-10-10', 21, 'andres@gmail.com', 'M', 2, 2);
75 insert into persona(nombre, apellidos, fecha_nac, edad, email, sexo, id_dep, id_prov)
76 values ('Daniel', 'Javiery', '2002-10-10', 20, 'daniel@gmail.com', 'M', 3, 3);
77 insert into persona(nombre, apellidos, fecha_nac, edad, email, sexo, id_dep, id_prov)
78 values ('Micaela', 'Cruz', '2003-10-10', 19, 'micaela@gmail.com', 'F', 4, 4);

```

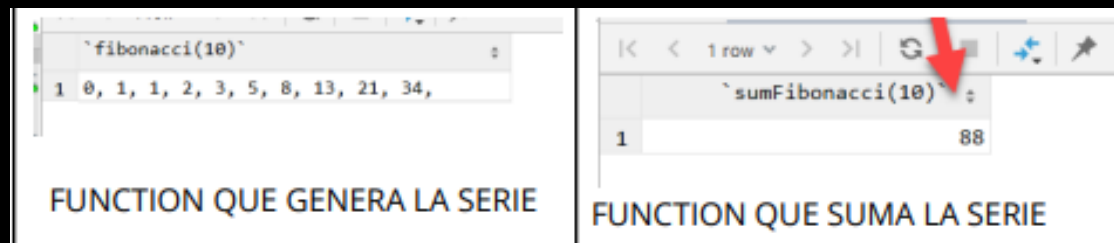
```

80 insert into detalle_proyecto(id_per, id_proy)
81 values (1,1);
82 insert into detalle_proyecto(id_per, id_proy)
83 values (2,2);
84 insert into detalle_proyecto(id_per, id_proy)
85 values (3,3);
86 insert into detalle_proyecto(id_per, id_proy)
87 values (4,4);

```

10. CREAR UNA FUNCIÓN QUE SUME LOS VALORES DE LA SERIE FIBONACCI.

- El objetivo es sumar todos los números de la serie fibonacci desde una cadena.
- Es decir usted tendrá solo la cadena generada con los primeros N números de la serie fibonacci y a partir de ellos deberá sumar los números de esa serie.
- Ejemplo: `suma_serie_fibonacci(mi_metodo_que_retorna_la_serie(10))`
 - Note que previamente deberá crear una función que retorne una cadena con la serie fibonacci hasta un cierto valor. 1. Ejemplo: 0,1,1,2,3,5,8,.....
 - Luego esta función se deberá pasar como parámetro a la función que suma todos los valores de esa serie generada.



Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.


```

create function fibonacci(@ENTRADA int)
returns text
begin
    declare a int default 0;
    declare b int default 1;
    declare cont int default 0;
    declare aux int default 0;
    declare unir text default '';

    while cont < entrada
    do
        if cont = 0
        then
            set unir = '0 ';
        else
            set unir = concat(unir, b, ' ');
            set aux = a;
            set a = b;
            set b = aux + a;
        end if;
        set cont = cont + 1;
    end while;
    return unir;
end;

select fibonacci(10)

```

```

create function suma_fibonacci(entrada text)
returns int
begin
    declare espacio text default ' ';
    declare x int default 1;
    declare nVeces int default 0;
    declare letra char default '';
    declare limite int default char_length(entrada);
    declare a int default 0;
    declare b int default 1;
    declare cont int default 0;
    declare aux int default 0;
    declare sumar int default 0;

    while x <= limite do
        set letra = substring(entrada, x, 1);
        if letra = espacio
        then
            set nVeces = nVeces + 1;
        end if;
        set x = x + 1;
    end while;
    while cont < nVeces
    do
        if cont = 0
        then
            set sumar = 0;
        else
            set sumar = sumar + b;
            set aux = a;
            set a = b;
            set b = aux + a;
        end if;
        set cont = cont + 1;
    end while;
    return sumar;
end;

select suma_fibonacci(fibonacci(10)) as suma_fibonacci

```

11.MANEJO DE VISTAS

-Crear una consulta SQL para lo siguiente.

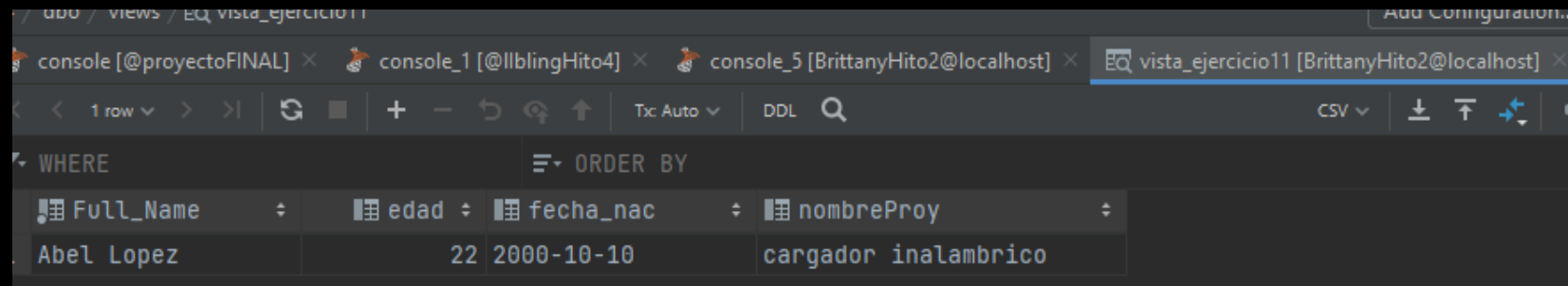
■ La consulta de la vista debe reflejar como campos:

1.- nombres y apellidos concatenados 2.- la edad 3.- fecha de nacimiento.

4.- Nombre del proyecto

○ Obtener todas las personas del sexo femenino que hayan nacido en el departamento de El Alto en donde la fecha de nacimiento sea: 1. fecha_nac = '2000-10-10'

```
create view vista_ejercicio11 as
select concat(per.nombre, ' ', per.apellidos) as Full_Name, per.edad,
per.fecha_nac, pro.nombreProy
from persona as per
inner join detalle_proyecto as dep on per.id_per = dep.id_per
inner join proyecto as pro on dep.id_proy = pro.id_proy
inner join departamento as depa on per.id_dep = depa.id_dep
where per.sexo = 'M' and depa.nombre = 'La Paz' and per.fecha_nac = '2000-10-10'
```



The screenshot shows a SQL IDE interface with a query editor at the top and a results pane at the bottom. The query editor contains the SQL code for creating the view 'vista_ejercicio11'. The results pane shows the output of the query, which is a single row of data. The columns are 'Full_Name', 'edad', 'fecha_nac', and 'nombreProy'. The values are 'Abel Lopez', '22', '2000-10-10', and 'cargador inalambrico' respectively.

Full_Name	edad	fecha_nac	nombreProy
Abel Lopez	22	2000-10-10	cargador inalambrico

12.-MANEJO DE TRIGGERS I.

- Crear TRIGGERS Before or After para INSERT y UPDATE aplicado a la tabla PROYECTO
 - Debera de crear 2 triggers minimamente.
- Agregar un nuevo campo a la tabla PROYECTO.
 - El campo debe llamarse ESTADO
- Actualmente solo se tiene habilitados ciertos tipos de proyectos.
 - EDUCACION, FORESTACION y CULTURA
- Si al hacer insert o update en el campo tipoProy llega los valores EDUCACION, FORESTACIÓN
- CULTURA, en el campo ESTADO colocar el valor ACTIVO. Sin embargo se llegat un tipo de proyecto distinto colocar INACTIVO
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
create trigger insertar_datos
before insert on proyecto
for each row
begin
  case
    when new.tipoProy = 'Robotica' or new.tipoProy = 'Forestacion' or new.tipoProy = 'Arte' then
      set new.estado = 'Activo';
    else
      set new.estado = 'Inactivo';
    end case;
end;
```

```

create trigger actualizar_datos
before update on proyecto
for each row
begin
    case
        when new.tipoProy = 'Robotica' or new.tipoProy = 'Forestacion' or
new.tipoProy = 'Arte' then
            set new.estado = 'Activo';
        else
            set new.estado = 'Inactivo';
        end case;
    end;

alter table proyecto add estado varchar(8) not null;

update proyecto as proy
set proy.estado = 'Activo'
where proy.tipoProy = 'Robotica' or proy.tipoProy = 'Forestacion' or proy.tipoProy =
'Arte';

update proyecto as proy
set proy.estado = 'Inactivo'
where proy.tipoProy = 'Deporte' or proy.tipoProy = 'Forestacion';

```

```

select *
from proyecto

insert into proyecto(nombreProy, tipoProy)
values ('Nombreproy5', 'Robotica')

insert into proyecto(nombreProy, tipoProy)
values ('Nombreproy6', 'Deporte')

UPDATE proyecto as proy
SET proy.tipoProy = 'Deporte'
WHERE proy.nombreProy= 'Nombreproy5';

UPDATE proyecto as proy
SET proy.tipoProy = 'Robotica'
WHERE proy.nombreProy= 'Nombreproy6';

```

13.MANEJO DE TRIGGERS II.

- ◦ El trigger debe de llamarse calculaEdad.
- ◦ El evento debe de ejecutarse en un BEFORE INSERT.
- ◦ Cada vez que se inserta un registro en la tabla PERSONA, el trigger debe de calcular la edad en función a la fecha de nacimiento.
- ◦ Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
create or replace trigger calculaEdad
before insert on persona
for each row
begin
    declare anio int default 0;
    declare anio_actual int default 0;

    set anio = (select max(substr(new.fecha_nac, 1, 4))
from persona as per);
    set anio_actual = (SELECT substr(curdate(),1, 4));

    set new.edad = anio_actual - anio;
end;

insert into persona (nombre, apellidos, fecha_nac, email, sexo, id_dep, id_prov)
values ('Nombre7', 'Apellidos7', '1998-10-10', 'nombr76@gamil.com', 'F', 1, 1)

select per.*
from persona as per
```


14.MANEJO DE TRIGGERS III

Crear otra tabla con los mismos campos de la tabla persona(Excepto el primary key id_per).

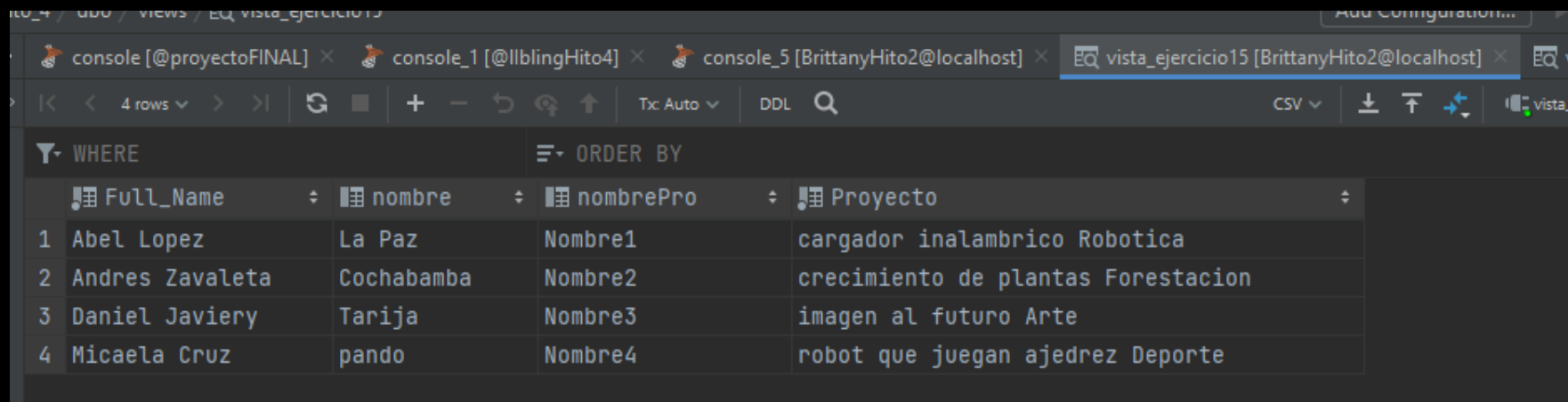
- No es necesario que tenga PRIMARY KEY.
- Cada vez que se haga un INSERT a la tabla persona estos mismos valores deben insertarse a la tabla copia.
- Para resolver esto deberá de crear un trigger before insert para la tabla PERSONA.
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
create table datos(  
  nombres varchar(50),  
  apellidos varchar(50),  
  fecha_nac date,  
  edad int,  
  email varchar(50),  
  sexo char  
);  
  
create trigger datos  
  before insert on persona  
  for each row  
  begin  
    insert into datos (nombres, apellidos, fecha_nac, edad, email, sexo)  
    values (new.nombre, new.apellidos, new.fecha_nac, new.edad, new.email, new.sexo);  
  end;  
  
insert into persona(nombre, apellidos, fecha_nac, edad, email, sexo, id_dep, id_prov)  
values ('Nombre6', 'Apellido6', '1999-10-10', 23, 'email6@gmail.com', 'F', 1, 1)  
  
select da.*  
from datos as da
```

15. CREAR UNA CONSULTA SQL QUE HAGA USO DE TODAS LAS TABLAS.

- La consulta generada convertirlo a VISTA

```
create view vista_ejercicio15 as
select concat(per.nombre, ' ', per.apellidos) as Full_Name,
       depa.nombre, pro.nombre as nombrePro, concat(proy.nombreProy, ' ', proy.tipoProy) as Proyecto
from departamento as depa
inner join provincia as pro on depa.id_dep = pro.id_dep
inner join persona as per on depa.id_dep = per.id_dep
inner join detalle_proyecto as dep on per.id_per = dep.id_per
inner join proyecto as proy on dep.id_proy = proy.id_proy
```



The screenshot shows a database management tool interface. At the top, there are several tabs: 'console [@proyectoFINAL]', 'console_1 [@lblingHito4]', 'console_5 [BrittanyHito2@localhost]', and 'EQ vista_ejercicio15 [BrittanyHito2@localhost]'. The 'EQ vista_ejercicio15' tab is active, displaying a table with 4 rows and 4 columns. The columns are labeled 'Full_Name', 'nombre', 'nombrePro', and 'Proyecto'. The rows contain the following data:

	Full_Name	nombre	nombrePro	Proyecto
1	Abel Lopez	La Paz	Nombre1	cargador inalambrico Robotica
2	Andres Zavaleta	Cochabamba	Nombre2	crecimiento de plantas Forestacion
3	Daniel Javiery	Tarija	Nombre3	imagen al futuro Arte
4	Micaela Cruz	pando	Nombre4	robot que juegan ajedrez Deporte