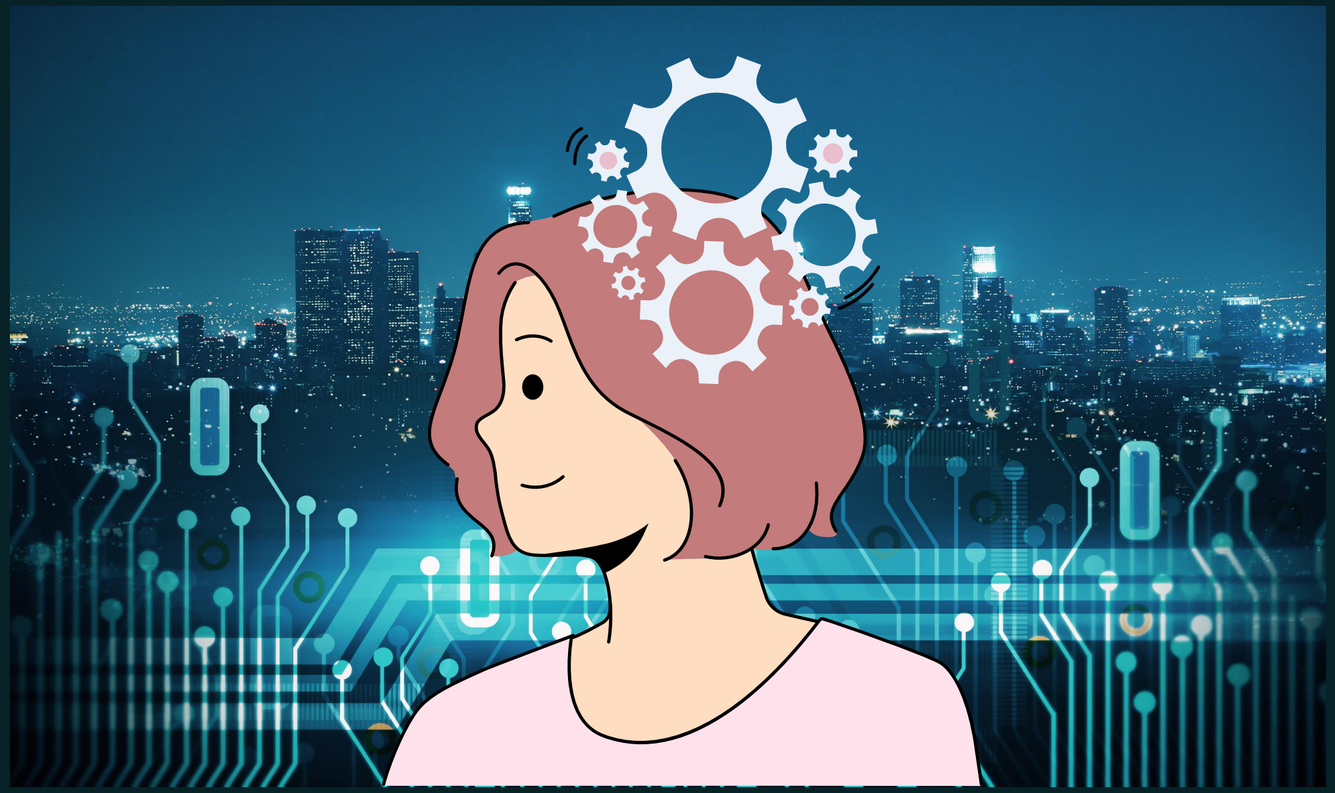
FUNCIONES, VISTAS Y TRIGGERS

BASE DE DATOS II

DOCENTE: WILLIAM BARRA

ESTUDIANTE: ILIA ARACELI SARZO LAURA

MANEJO DE CONCEPTO



Defina que es lenguaje procedural en MySQL

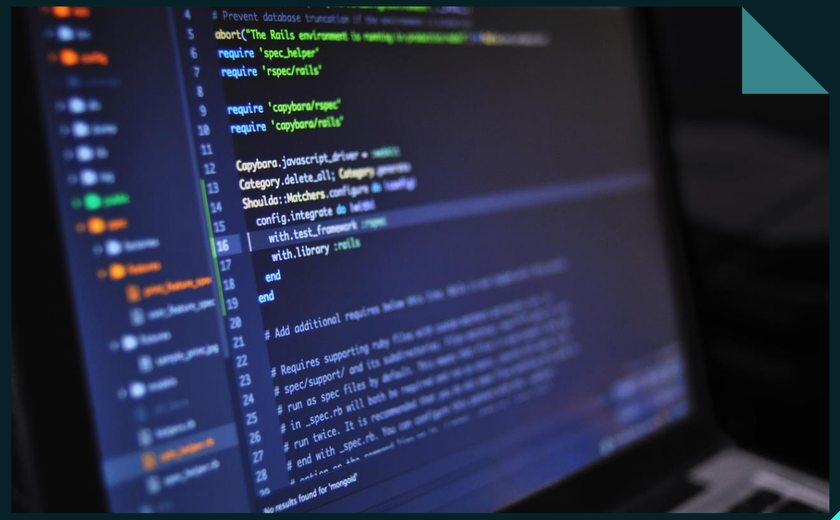
Los procedimientos almacenados MySQL, también conocidos como Stored Procedure, se presentan como conjuntos de instrucciones escritas en el lenguaje SQL. Su objetivo es realizar una tarea determinada, desde operaciones sencillas hasta tareas muy complejas.
Es el manejo de estructuras "bucles, sentencias, variables" dentro del gestor de base de datos

Defina que es una funCtion en MySQL

Una función de SQL se define en la base de datos sólo a través de sentencias SQL, incluida, como mínimo, una sentencia RETURN la cual nos dice que es una reutina creada que puede recibir parametros para realizar un proceso y devolver un resultado en concreto.

BASE DE DATOS II

Cual es la diferencia entre funciones y procedimiento almacenados



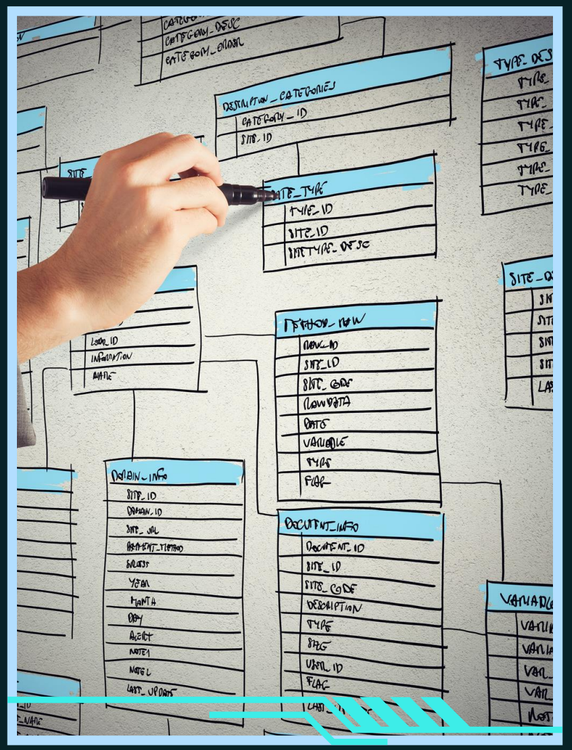
Una función y un procedimiento almacenado, son diferentes, aunque con ciertas similitudes. Pero cuando debemos de usar cada una. Generalmente se usa una función para calcular un valor y regresarlo para utilizarlo posteriormente en alguna expresión. Un procedimiento se usa para producir un efecto o una acción sin necesidad de regresar algún valor.

la diferencia mas notable es que una funcion no devuelve un valor unico; mientras que un procedimiento almacenado no devuelve ningin valor.

Como se ejecuta una funcion

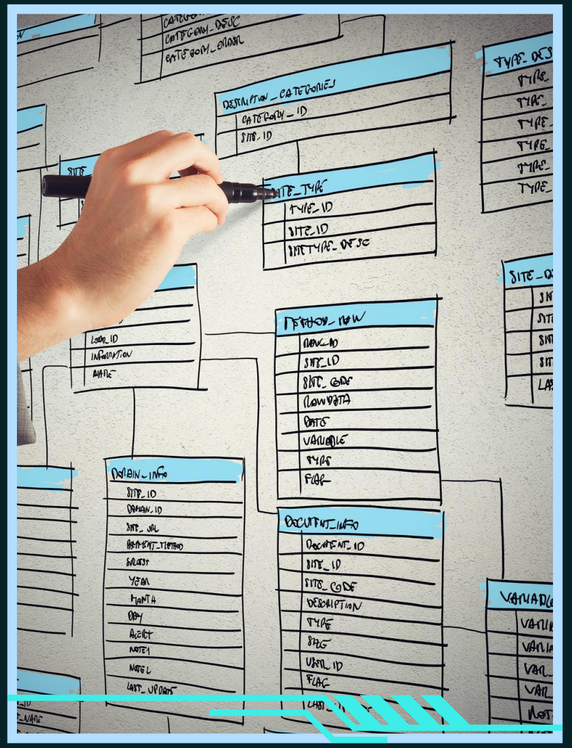
se lo invoca solo llamando a Select y al nombre de la funcion

select numeros_naturales(30);



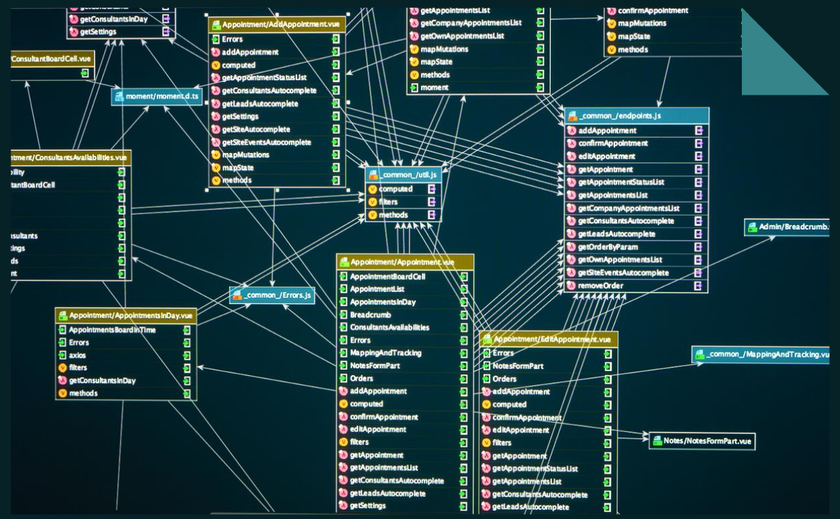
Como se ejecuta un procedimiento almacenado

un procedimiento almacenado se ejecuta con el comando CALL



TRIGGER

Define qu es un TRIGGER en MySQL

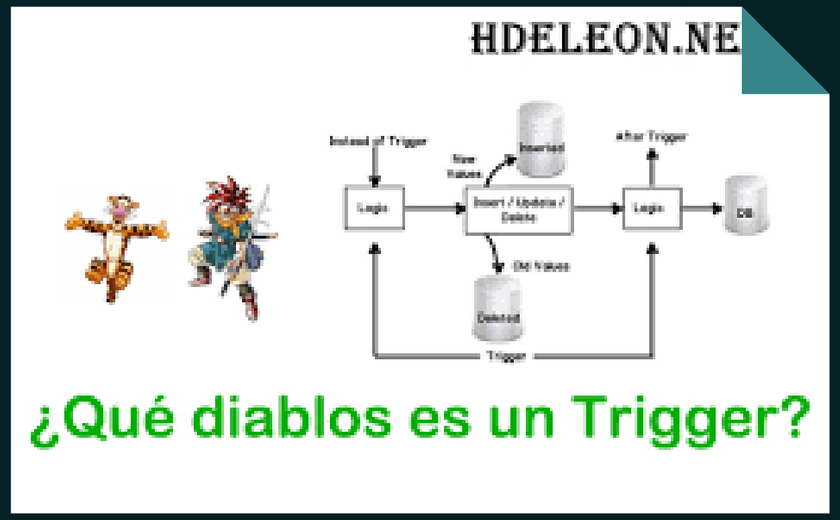


TRIGGER LOS TRIGGER SON PROGRAMAS ALMACENADOS, QUE SE EJECUTAN AUTOMATICAMENTE CUANDO OCURRE UN EVENTO LOS TRIGGER SE EJECUTAN DE MANERA AUTOMATICA.

- Inset- Updale - Dalete

TRIGGER

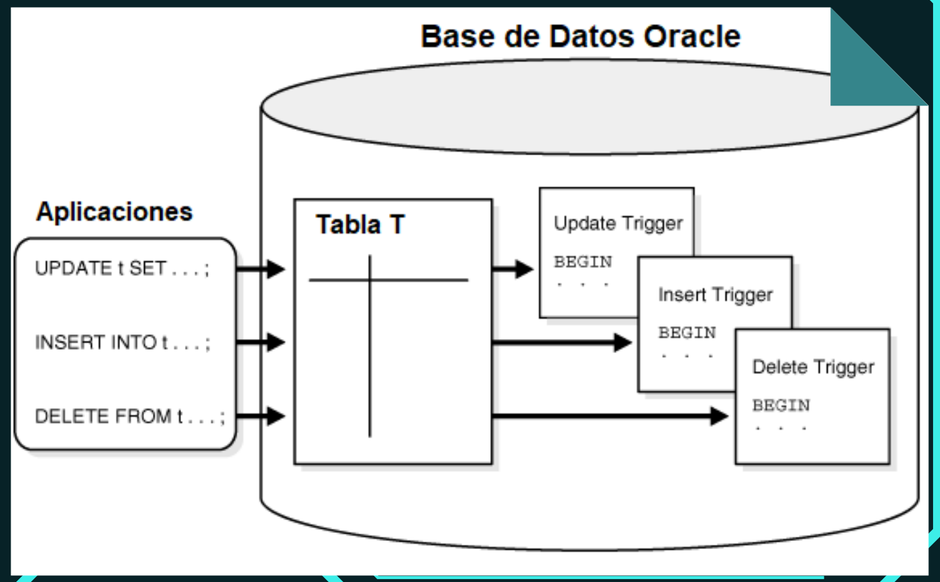
Define qu es un TRIGGER en MySQL



TRIGGER LOS TRIGGER SON PROGRAMAS ALMACENADOS, QUE SE EJECUTAN AUTOMATICAMENTE CUANDO OCURRE UN EVENTO LOS TRIGGER SE EJECUTAN DE MANERA AUTOMATICA.

- Inset- Updale - Dalete

TRIGGER

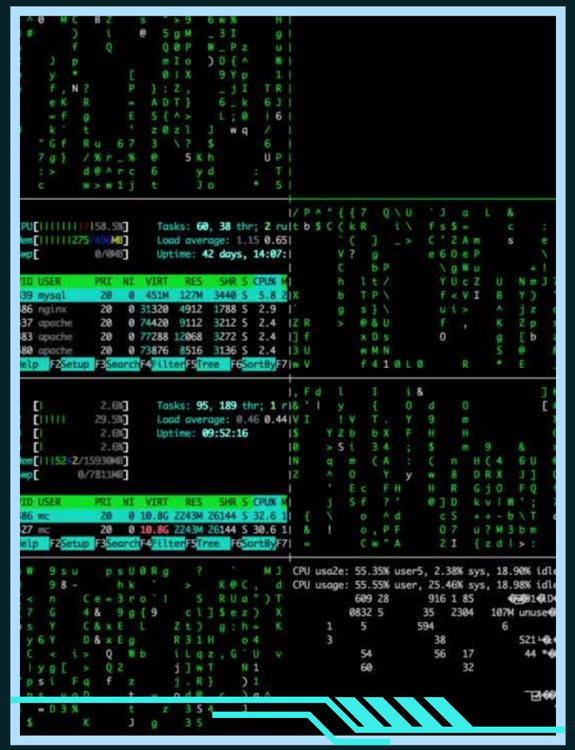


En un Trigger que papel juega las variablers OLD y NEW

La variable OLD nos permite acceder a la viaje y la sentencia NEW nos permite acceder a a nueva variable; antes y despues de un evento UPDATE

En un Trigger que papel juega las variablers OLD y NEW

La variable OLD nos permite acceder a la viaje y la sentencia NEW nos permite acceder a a nueva variable; antes y despues de un evento UPDATE



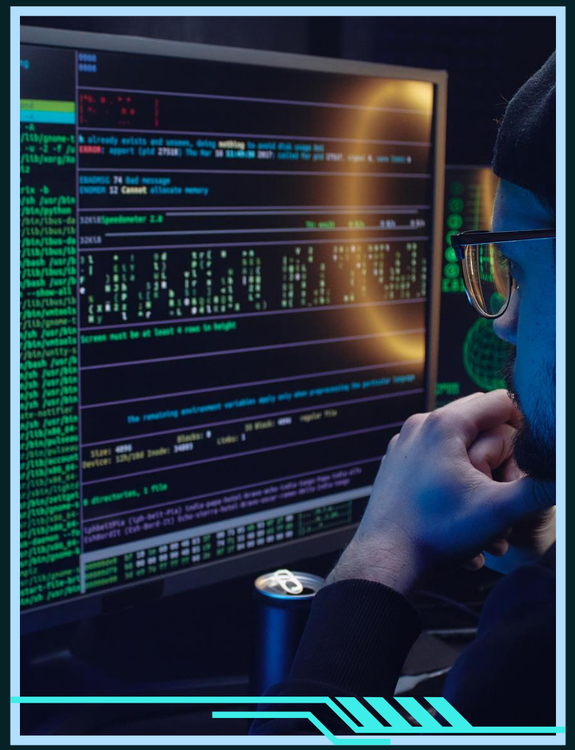
A que se refiere cuando se habla de eventos en TRIGGERS

Unos de los eventos son acciones que se realizan, en caso de los trigger se refiere escíticamente al

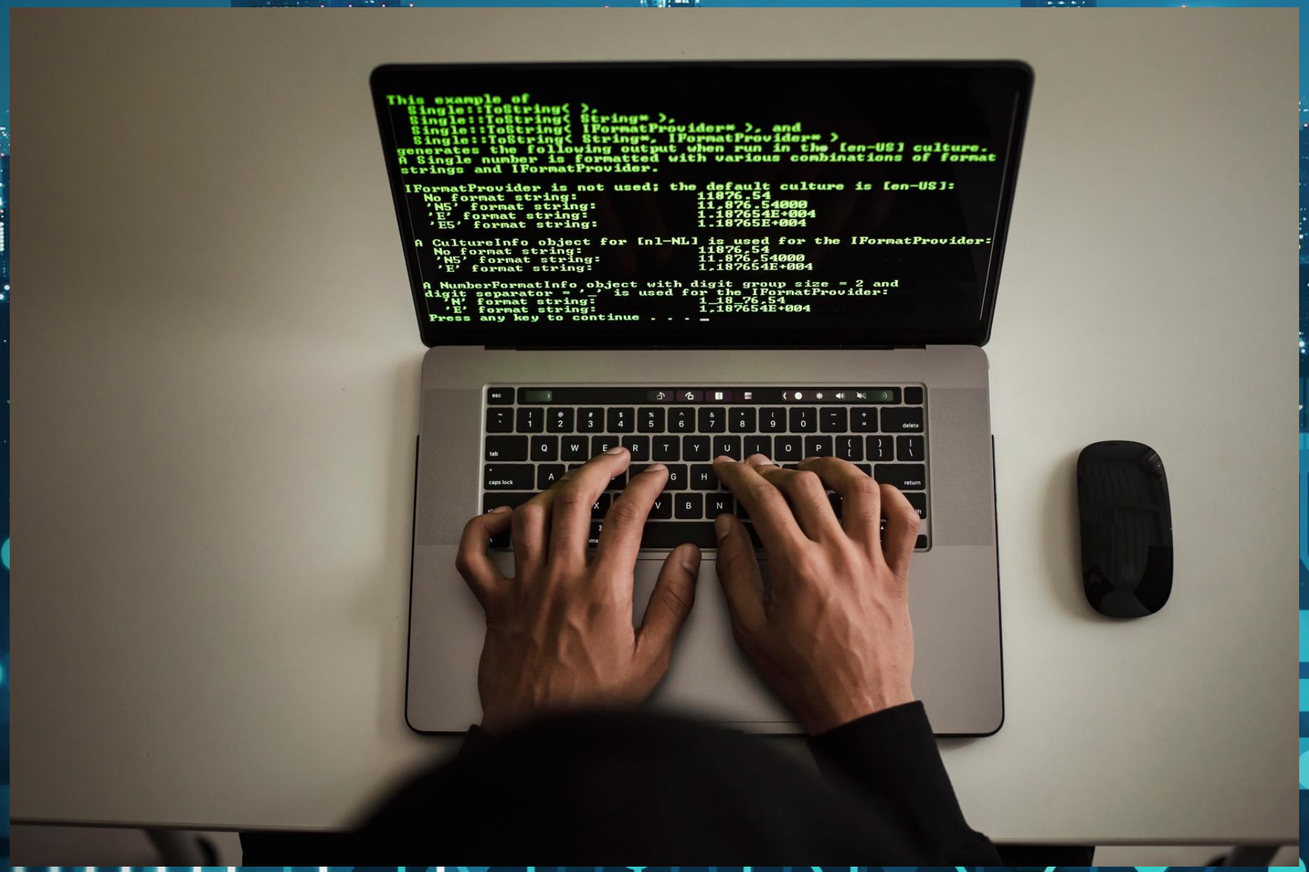
Update = Modificar

Delete = borrar

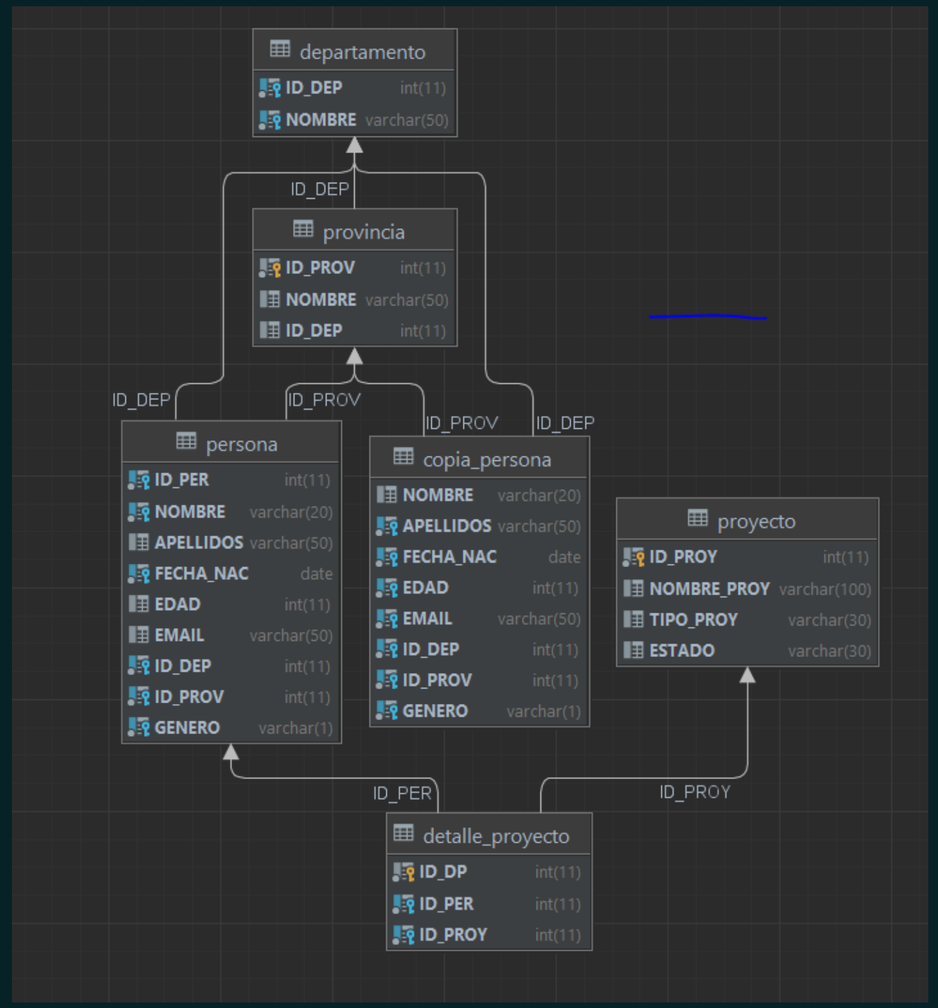
Insert = insertar

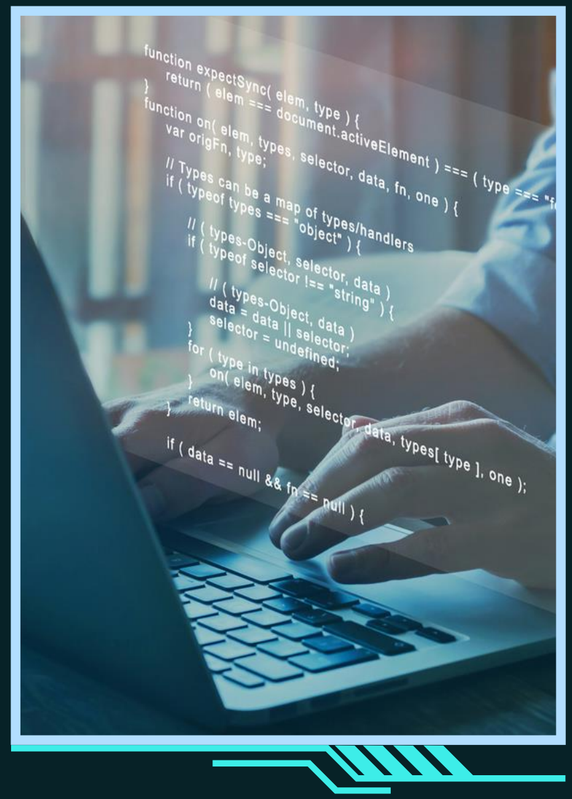


MANEJO PRACTICO



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





```sql
USE PRACTICA_H4;

CREATE TABLE  DEPARTAMENTO
(
    ID_DEP INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    NOMBRE VARCHAR(50)
);

CREATE TABLE PROVINCIA
(
    ID_PROV INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    NOMBRE VARCHAR(50),
    ID_DEP INT NOT NULL,
    FOREIGN KEY (ID_DEP) REFERENCES DEPARTAMENTO(ID_DEP)
);

CREATE TABLE PROYECTO
(
    ID_PROY INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    NOMBRE_PROY VARCHAR(100),
    TIPO_PROY VARCHAR(30)
);

CREATE TABLE PERSONA
(
  ID_PER INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  NOMBRE VARCHAR(20),
  APELLIDOS VARCHAR(50),
  FECHA_NAC DATE,
  EDAD INT,
  EMAIL VARCHAR(50),
  ID_DEP INT NOT NULL ,
  ID_PROV INT NOT NULL,
  GENERO VARCHAR(1),
  FOREIGN KEY (ID_PROV) REFERENCES  PROVINCIA(ID_PROV),
  FOREIGN KEY (ID_DEP) REFERENCES DEPARTAMENTO(ID_DEP)
```

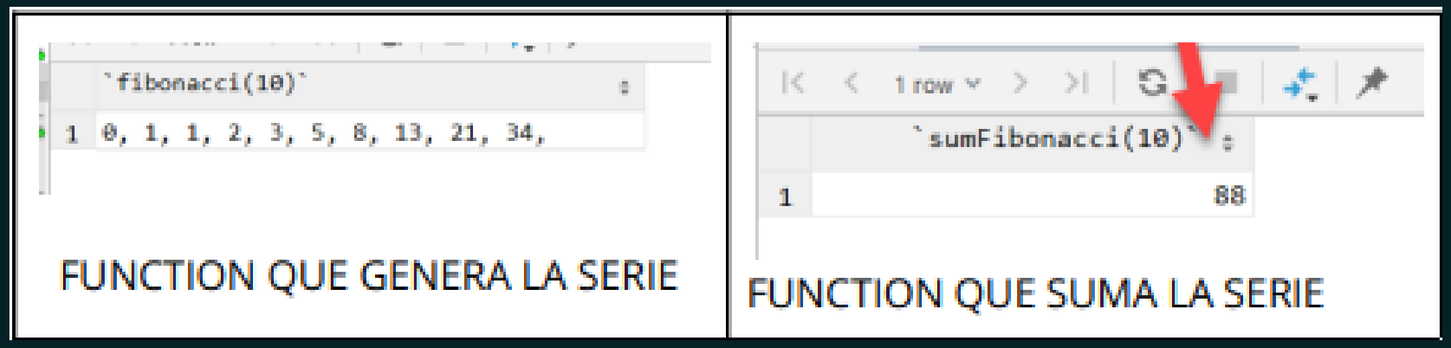
```sql
CREATE TABLE DETALLE_PROYECTO
(
    ID_DP INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
    ID_PER INT NOT NULL ,
    ID_PROY INT NOT NULL ,

    FOREIGN KEY (ID_PROY) REFERENCES PROYECTO(ID_PROY),
    FOREIGN KEY (ID_PER) REFERENCES PERSONA(ID_PER)
);



INSERT INTO  DEPARTAMENTO(NOMBRE)
VALUES ('LA PAZ'),
       ('SANTA CRUZ'),
       ('BENI'),
       ('ORURO'),
       ('CHUQUISACA');


INSERT INTO  PROVINCIA(NOMBRE, ID_DEP)
VALUES ('VIACHA',1),
       ('ROBORE',2),
       ('MAGDALENA',3),
       ('CHALLAPATA',4),
       ('TARABUCO',5);


INSERT INTO PERSONA (NOMBRE, APELLIDOS, FECHA_NAC, EDAD, EMAIL, I, ID_PROV, GENERO)
VALUES ('RODRIGO','MENODZA ','1999-11-21',23,'RODRIGO@GMAIL.COM',1,1,'M'),
       ('MARIA','LAURA ','1999-12-16',25,'MARIA@GAMIL.COM',2,2,'F'),
       ('AUGUSTO','MEDRANO ','1998-09-12',24,'AUGUSTO@GMAIL.COM',3,3,'M'),
       ('MARIANO','FERNANDEZ ','1995-09-12',27,'MARIANO@GMAIL.COM',4,4,'M'),
       ('LORENA','ZAMUDIO LLANOS','2000-09-12',22,'LORENA@GMAIL.COM',5,5,'F');
```

```sql
INSERT INTO  PROYECTO(NOMBRE_PROY, TIPO_PROY)
VALUES('ANIMALES','BIOLOGIA'),
      ('PREHISTORIA','ANTROPOLOGIA'),
      ('MICROBIOS','BIOLOGIA'),
      ('REDES Y SISTEMAS','TECNOLOGIA'),
      ('FLUJO MAGNETICO','FISICA');
INSERT INTO DETALLE_PROYECTO(ID_PER, ID_PROY)
VALUES (1,2),
       (2,1),
       (3,4),
       (4,3);
```

```sql
INSERT INTO  PROYECTO(NOMBRE_PROY, TIPO_PROY)
VALUES('ANIMALES','BIOLOGIA'),
      ('PREHISTORIA','ANTROPOLOGIA'),
      ('MICROBIOS','BIOLOGIA'),
      ('REDES Y SISTEMAS','TECNOLOGIA'),
      ('FLUJO MAGNETICO','FISICA');
INSERT INTO DETALLE_PROYECTO(ID_PER, ID_PROY)
VALUES (1,2),
       (2,1),
       (3,4),
       (4,3);
```

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.



10.Crear una función que sume los valores de la serie Fibonacci.

```
CREATE OR REPLACE FUNCTION SERIE_FIBONANCI(NUMBER INTEGER)

RETURNS TEXT
BEGIN
    DECLARE A INTEGER DEFAULT 0;
    DECLARE B INTEGER DEFAULT 1;
    DECLARE AUX INTEGER DEFAULT 0;
    DECLARE CONTADOR INTEGER DEFAULT 0;
    DECLARE CADENA TEXT DEFAULT '';
    SET CADENA =CONCAT(A,',',B);

IF NUMBER=1 THEN SET CADENA='0';
ELSEIF NUMBER=2 THEN SET CADENA='0,1';
ELSEIF NUMBER<=0 THEN SET CADENA='EL NUMERO DEBE SER MAYOR A CERO';
ELSE
            REPEAT

            SET AUX=A+B;
            SET CADENA = CONCAT(CADENA,',',AUX);
            SET A=B;
            SET B=AUX;
            SET CONTADOR=CONTADOR+1;
            UNTIL CONTADOR = NUMBER-2 END REPEAT;

END IF;
    RETURN CADENA;
END;
```

```
CREATE OR REPLACE FUNCTION CONTAR_FIBONANCI(SERIE TEXT)
RETURNS INTEGER
BEGIN
    DECLARE SUMA INTEGER DEFAULT 0;
    DECLARE CONT INTEGER DEFAULT 1;
    DECLARE FINAL INTEGER DEFAULT CHAR_LENGTH(SERIE);

  REPEAT
   SET SUMA =SUMA + SUBSTRING(SERIE,CONT,1);
   SET CONT=CONT+2;
  until  CONT> FINAL END REPEAT;

    RETURN SUMA;
end;

SELECT SERIE_FIBONANCI( NUMBER: 5);
SELECT CONTAR_FIBONANCI( SERIE: SERIE_FIBONANCI( NUMBER: 5));
```

10.Crear una función que sume los valores de la serie Fibonacci.

EXPECTATIVA

	`fibonacci(10)`
1	0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

FUNCTION QUE GENERA LA SERIE

1 row

	`sumFibonacci(10)`
1	88

FUNCTION QUE SUMA LA SERIE

REALIDAD

2 rows | Tx: Auto | DDL | CSV

Go to DDL

	NOMBRES_Y_APELLIDOS	EDAD	FECHA_DE... TO	NOMBRE_DEL_PROYECTO
1	CARMEN CALLE UGARTE	23	2000-10-10	REDES Y SISTEMAS
2	GABRIELA BARRA MENDOZA	23	2000-10-10	FLUJO MAGNETICO

11.Manejo de vistas.

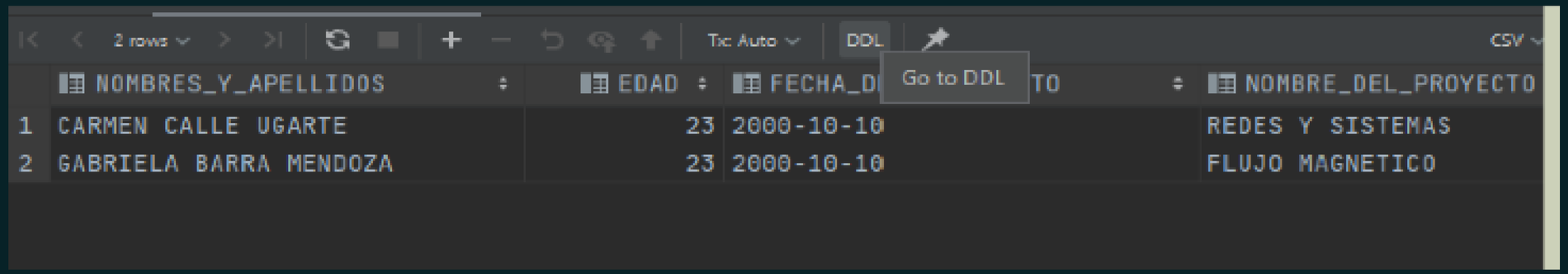
Crear una consulta SQL para lo siguiente.
La consulta de la vista debe reflejar como campos: 1. nombres y apellidos concatenados 2. la edad 3. fecha de nacimiento. 4. Nombre del proyecto
Obtener todas las personas del sexo femenino que hayan nacido en el departamento de El Alto en donde la fecha de nacimiento sea:
1. fecha_nac = '2000-10-10' LA CONSULTA GENERADA PREVIAMENTE CONVERTIR EN UNA VISTA

11.Manejo de vistas

```
INSERT INTO DEPARTAMENTO(NOMBRE)
VALUES('EL ALTO');

INSERT INTO PERSONA (NOMBRE, APELLIDOS, FECHA_NAC, EDAD, EMAIL, ID_DEP, ID_PROV, GENERO)
VALUES ('CARMEN','CALLE UGARTE','2000-10-10',23,'CARMEN@GMAIL.COM',10,1,'F'),
       ('GABRIELA','BARRA MENDOZA','2000-10-10',23,'GABRIELA@GMAIL.COM',10,1,'F');
INSERT INTO DETALLE_PROYECTO(ID_PER, ID_PROY)
VALUES (6,4),
       (7,5);


CREATE OR REPLACE VIEW BUSQUEDA AS
SELECT CONCAT(PERSONA.NOMBRE,' ',PERSONA.APELLIDOS) AS NOMBRES_Y_APELLIDOS,
           PERSONA.EDAD AS EDAD,
           PERSONA.FECHA_NAC AS FECHA_DE_NACIMIENTO,
        PROYECTO.NOMBRE_PROY AS NOMBRE_DEL_PROYECTO
FROM PERSONA
INNER JOIN DEPARTAMENTO 1..n<->1: ON PERSONA.ID_DEP = DEPARTAMENTO.ID_DEP
INNER JOIN DETALLE_PROYECTO 1<->1..n: ON PERSONA.ID_PER = DETALLE_PROYECTO.ID_PER
INNER JOIN PROYECTO 1..n<->1: ON DETALLE_PROYECTO.ID_PROY = PROYECTO.ID_PROY

WHERE PERSONA.GENERO='F' AND  DEPARTAMENTO.NOMBRE='EL ALTO' AND PERSONA.FECHA_NAC='2000-10-10' ;

SELECT * FROM BUSQUEDA;
```

11.Manejo de vistas

REALIDAD



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.

13.Manejo de Triggers II.

```
CREATE OR REPLACE TRIGGER AGREGAR_EDAD
BEFORE INSERT ON PERSONA
FOR EACH ROW
    BEGIN
        SET NEW.EDAD= TIMESTAMPDIFF(YEAR,NEW.FECHA_NAC,CURDATE());
    end;

INSERT INTO  PERSONA(NOMBRE, APELLIDOS, FECHA_NAC, EMAIL, ID_DEP, ID_PROV, GENERO)
VALUES  ('MARIA','GALARSA ORTEGA','1992-12-15','MARIA@GMAIL.COM',3,3,'F');

SELECT *FROM PERSONA;
```

13.Manejo de TRIGGERS II

	ID_PER	NOMBRE	APELLIDOS	FECHA_NAC	EDAD	EMAIL	ID_DEP	ID_PROV	GENERO
1	1	RODRIGO	MENODZA	1999-11-21	23	RODRIGO@GMAIL.COM	1	1	M
2	2	MARIA	LAURA	1999-12-16	25	MARIA@GAMIL.COM	2	2	F
3	3	AUGUSTO	MEDRANO	1998-09-12	24	AUGUSTO@GMAIL.COM	3	3	M
4	4	MARIANO	FERNANDEZ	1995-09-12	27	MARIANO@GMAIL.COM	4	4	M
5	5	LORENA	ZAMUDIO LLANOS	2000-09-12	22	LORENA@GMAIL.COM	5	5	F
6	6	CARMEN	CALLE UGARTE	2000-10-10	23	CARMEN@GMAIL.COM	10	1	F
7	7	GABRIELA	BARRA MENDOZA	2000-10-10	23	GABRIELA@GMAIL.COM	10	1	F
8	8	CARMEN	CALLE UGARTE	2000-10-10	23	CARMEN@GMAIL.COM	10	1	F
9	9	GABRIELA	BARRA MENDOZA	2000-10-10	23	GABRIELA@GMAIL.COM	10	1	F
10	10	CARMEN	CALLE UGARTE	2000-10-10	23	CARMEN@GMAIL.COM	10	1	F
11	11	GABRIELA	BARRA MENDOZA	2000-10-10	23	GABRIELA@GMAIL.COM	10	1	F
12	12	MARIA	GALARSA ORTEGA	1992-12-15	29	MARIA@GMAIL.COM	3	3	F
13	13	PDRO	ALIAGA ORTEGA	2000-12-15	21	ALEJANDRA@GMAIL.COM	5	5	M
14	14	CARMEN	CALLE UGARTE	2000-10-10	22	CARMEN@GMAIL.COM	10	1	F
15	15	GABRIELA	BARRA MENDOZA	2000-10-10	22	GABRIELA@GMAIL.COM	10	1	F
16	16	MARIA	GALARSA ORTEGA	1992-12-15	29	MARIA@GMAIL.COM	3	3	F

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.

```sql
CREATE TABLE COPIA_PERSONA
(
  NOMBRE VARCHAR(20),
  APELLIDOS VARCHAR(50),
  FECHA_NAC DATE,
  EDAD INT,
  EMAIL VARCHAR(50),
  ID_DEP INT NOT NULL ,
  ID_PROV INT NOT NULL,
  GENERO VARCHAR(1),
  FOREIGN KEY (ID_PROV) REFERENCES  PROVINCIA(ID_PROV),
  FOREIGN KEY (ID_DEP) REFERENCES DEPARTAMENTO(ID_DEP)
);
```

14.Manejo de TRIGGERS III.

```sql
CREATE OR REPLACE TRIGGER COPIAR_PERSONA
BEFORE INSERT ON PERSONA
FOR EACH ROW
    BEGIN
        INSERT INTO COPIA_PERSONA(NOMBRE, APELLIDOS, FECHA_NAC, EDAD, EMAIL, ID_DEP, ID_PROV, GENERO)
        VALUES(NEW.NOMBRE,NEW.APELLIDOS,NEW.FECHA_NAC,NEW.EDAD,NEW.EMAIL,NEW.ID_DEP,NEW.ID_PROV,NEW.GENERO);
    end;

INSERT INTO PERSONA(NOMBRE, APELLIDOS, FECHA_NAC, EDAD, EMAIL, ID_DEP, ID_PROV, GENERO)
VALUES ('PDRO','ALIAGA ORTEGA','2000-12-15',22,'ALEJANDRA@GMAIL.COM',5,5,'M');

SELECT*FROM COPIA_PERSONA;
```

Resultado

	NOMBRE	APELLIDOS	FECHA_NAC	EDAD	EMAIL	ID_DEP	ID_PROV	GENERO
1	PDRO	ALIAGA ORTEGA	2000-12-15	21	ALEJANDRA@GMAIL.COM	5	5	M
2	CARMEN	CALLE UGARTE	2000-10-10	22	CARMEN@GMAIL.COM	10	1	F
3	GABRIELA	BARRA MENDOZA	2000-10-10	22	GABRIELA@GMAIL.COM	10	1	F
4	MARIA	GALARSA ORTEGA	1992-12-15	29	MARIA@GMAIL.COM	3	3	F
5	PDRO	ALIAGA ORTEGA	2000-12-15	21	ALEJANDRA@GMAIL.COM	5	5	M

15.Crear una consulta SQL que haga uso de todas las tablas.

La consulta generada convertirlo a VISTA

```sql
CREATE OR REPLACE VIEW TODAS_LAS_TABLAS AS
SELECT
    CONCAT(PERSONA.NOMBRE,PERSONA.APELLIDOS) AS NOMBRE_Y_APELLIDOS,
    PERSONA.EDAD AS EDAD,
    DEPARTAMENTO.NOMBRE AS DEPARTAMENTO,
    PROVINCIA.NOMBRE AS PROVINCIA,
    CONCAT(PROYECTO.NOMBRE_PROY,': ',TIPO_PROY) AS PROYECTO

FROM PERSONA
INNER JOIN DEPARTAMENTO 1..n<->1: ON PERSONA.ID_DEP = DEPARTAMENTO.ID_DEP
INNER JOIN PROVINCIA 1..n<->1: ON PERSONA.ID_PROV = PROVINCIA.ID_PROV
INNER JOIN DETALLE_PROYECTO 1<->1..n: ON PERSONA.ID_PER = DETALLE_PROYECTO.ID_PER
INNER JOIN PROYECTO 1..n<->1: ON DETALLE_PROYECTO.ID_PROY = PROYECTO.ID_PROY;

SELECT * FROM (TODAS_LAS_TABLAS);
```

14.Manejo de TRIGGERS III.

Resultado

20 rows | Tx: Auto | DDL

	NOMBRE_Y_APELLIDOS	EDAD	DEPARTAMENTO	PROVINCIA	PROYECTO
1	RODRIGOMENODZA	23	LA PAZ	VIACHA	PREHISTORIA: ANTROPOLOGIA
2	CARMENCALLE UGARTE	23	LA PAZ	VIACHA	REDES Y SISTEMAS: TECNOLOGIA
3	CARMENCALLE UGARTE	23	LA PAZ	VIACHA	REDES Y SISTEMAS: TECNOLOGIA
4	CARMENCALLE UGARTE	23	LA PAZ	VIACHA	REDES Y SISTEMAS: TECNOLOGIA
5	CARMENCALLE UGARTE	23	LA PAZ	VIACHA	REDES Y SISTEMAS: TECNOLOGIA
6	CARMENCALLE UGARTE	23	LA PAZ	VIACHA	REDES Y SISTEMAS: TECNOLOGIA
7	CARMENCALLE UGARTE	23	LA PAZ	VIACHA	REDES Y SISTEMAS: TECNOLOGIA
8	CARMENCALLE UGARTE	23	LA PAZ	VIACHA	REDES Y SISTEMAS: TECNOLOGIA
9	CARMENCALLE UGARTE	23	LA PAZ	VIACHA	REDES Y SISTEMAS: TECNOLOGIA
10	GABRIELABARRA MENDOZA	23	LA PAZ	VIACHA	FLUJO MAGNETICO: FISICA
11	GABRIELABARRA MENDOZA	23	LA PAZ	VIACHA	FLUJO MAGNETICO: FISICA
12	GABRIELABARRA MENDOZA	23	LA PAZ	VIACHA	FLUJO MAGNETICO: FISICA
13	GABRIELABARRA MENDOZA	23	LA PAZ	VIACHA	FLUJO MAGNETICO: FISICA
14	GABRIELABARRA MENDOZA	23	LA PAZ	VIACHA	FLUJO MAGNETICO: FISICA
15	GABRIELABARRA MENDOZA	23	LA PAZ	VIACHA	FLUJO MAGNETICO: FISICA
16	GABRIELABARRA MENDOZA	23	LA PAZ	VIACHA	FLUJO MAGNETICO: FISICA
17	GABRIELABARRA MENDOZA	23	LA PAZ	VIACHA	FLUJO MAGNETICO: FISICA
18	MARIALAURA	25	SANTA CRUZ	ROBORE	ANIMALES: BIOLOGIA
19	AUGUSTOMEDRANO	24	BENI	MAGDALENA	REDES Y SISTEMAS: TECNOLOGIA
20	MARIANOFERNANDEZ	27	ORURO	CHALLAPATA	MICROBIOS: BIOLOGIA

GRACIAS POR SU ATENCION

ESTUDIANTE: Ilia Araceli Sarzo Laura