EJERCICIOS
TRATAMIENTO DE DATOS

Alejandro Pérez Martín
1ºDAM

10/03/2022

ÍNDICE:

TEMA 5

Consultas --------

EJERCICIO 1:

```
CREATE TABLE facturas (
  numero INT(10) unsigned zerofill NOT NULL auto_increment DEFAULT NULL,
  numeroitem smallint unsigned NOT NULL DEFAULT NULL,
  descripcion varchar(30) DEFAULT NULL,
  precioportunidad decimal(5,2) unsigned DEFAULT NULL;
  cantidad tinyint unsigned DEFAULT NULL
);
ALTER TABLE facturas ADD CONSTRAINT pk_facturas PRIMARY KEY (numero,
numeroitem);
INSERT INTO facturas VALUES
(
"100","1","escuadra 20 cm.","2.50","20";
"100","2","escuadra 50 cm.","5.00","30";
"100","3","goma lapiz-tinta","0.35","100";
"102","1","lapices coloresx6","4.40","50";
"102","2","lapices coloresx12","8.00","60";
"225","1","lapices coloresx24","12.35","100";
"567","1","compas plastico","12.00","50";
"567","2","compas metal","18.90","80";
);

4.
CREATE TABLE montofacturas (
   SELECT numero AS numerodefactura,
   (precioporunidad*Cantidad) AS total
   FROM facturas
);

5.  DROP TABLE montofacturas(if exist);

6.
SELECT numero AS numerodefactura,
   SUM(precioporunidad*Cantidad) AS total
   FROM facturas
   GROUP BY numero;
7.
CREATE TABLE montofacturas (
   SELECT numero AS numerodefactura,
   SUM(precioporunidad)*Cantidad AS total
   FROM facturas
   GROUP BY numero
);
```

EJERCICIO 2:

4.

```
SELECT deportes .nombre,inscritos .año,COUNT(*)
FROM deportes JOIN inscritos
ON deportes.codigo=inscritos.codigodeporte
JOIN socios
ON inscritos.documento=socios.documento
GROUP BY deportes.nombre,inscritos.año;
```

5.

```
Drop table inscritosporaño (if exist);
```

6.

```
CREATE TABLE inscritosporaño
SELECT deportes.nombre,inscritos.año,COUNT(*)
FROM deportes join inscritos
ON deportes.codigo=inscritos.codigodeporte
JOIN socios
ON inscritos.documento=socios.documento
GROUP BY deportes.nombre,inscritos.año;
```

7.

```
SELECT * FROM inscritosporaño;
```

8.

```
SELECT socio.nombre,count(inscritos.documento)
FROM socios
LEFT JOIN inscritos
ON socios.documento=inscritos.documento
LEFT JOIN deportes
ON inscritos.codigodeporte=deportes.codigo
GROUP BY socios.nombre;
```

9.

```
DROP TABLE sociosdeporte (if exist);
```

10.

```
CREATE TABLE sociosdeporte
SELECT socios.nombre,COUNT(inscritos.documento) AS inscripciones
FROM socios
LEFT JOIN inscritos
ON socios.documento=inscritos.documento
LEFT JOIN deportes
ON inscritos.codigodeporte=deportes.codigo
GROUP BY socios.nombre;
```

EJERCICIO 3:

```
CREATE TABLE socios
(
 numero INT unsigned auto_increment,
 documento CHAR(8) not null,
 nombre VARCHAR(30) not null,
 domicilio VARCHAR(30),
 PRIMARY KEY (numero)
 );

CREATE TABLE inscritos
 (
 numerosocio INT unsigned,
 deporte VARCHAR(20),
 año YEAR not null,
 cuota CHAR(1),
 PRIMARY KEY(numerosocio,deporte,año)
 );

INSERT INTO socios VALUES
 (
 23,"22333444","Juan Perez","Colon 123";
 56,"23333444","Ana Garcia","Sarmiento 984";
 102,"24333444","Hector Fuentes","Sucre 293";
 150,"25333444","Agustin Perez","Avellaneda 1234";
 230,"26333444","Maria Perez","Urquiza 283";
 231,"29333444","Agustin Perez","Urquiza 283"
 );

INSERT INTO inscritos VALUES
 (
 23,"tenis","2005","s";
 23,"tenis","2006","s";
 23,"natacion","2005","s";
 102,"tenis","2005","s";
 102,"natacion","2006","s"
 );
```

4.

```
 INSERT INTO inscritos (numerosocio,deporte,año,cuota)
 SELECT numero,"tenis","2016","s"
 FROMsocios
 WHEREdocumento="23333444";
```

5.

6.

```
INSERT INTO inscritos (numerosocio,deporte,año,cuota)
SELECT numero,"basquet","2016","n"
FROM socios
WHERE nombre="Agustin Perez" AND  domicilio="Urquiza 283";
```

7.

```
INSERT INTO inscritos (numerosocio,deporte,año,cuota)
SELECT numero,"natacion","2016","n"
FROM socios
WHERE domicilio="Urquiza 283";
```

8.

```
SELECT socios.nombre,inscritos.deporte,inscritos.año
from socios join inscritos
on socios.numero=inscritos.numerosocio;
```

EJERCICIO 4:

1. DROP TABLE facturas (if exist);

2.

```
CREATE TABLE facturas (
  numero INT(10) unsigned zerofill NOT NULL auto_increment,
  numeroitem smallint unsigned NOT NULL,
  descripcion VARCHAR(30),
  precioporunidad decimal(5,2),
  cantidad tinyint unsigned
)
```

3.

```
INSERT INTO facturas VALUES
(
"100","1","escuadra 20 cm.","2.50","20";
"100","2","escuadra 50 cm.","5.00","30";
"100","3","goma lapiz-tinta","0.35","100";
"102","1","lapices coloresx6","4.40","50";
"102","2","lapices coloresx12","8.00","60";
"225","1","lapices coloresx24","12.35","100";
"567","1","compas plastico","12.00","50";
"567","2","compas metal","18.90","80";
);
```

4. DROP TABLE facturas (if exist);

5.

```
CREATE TABLE montofacturas (
   SELECT numero AS numerodefactura ,
   (precioporunidad*Cantidad) AS total
   FROM facturas;
   PRIMARY KEY (numero)
);
```

6. SELECT DISTINC numero AS numerodefactura ,
(precioporunidad*Cantidad) AS total
FROM facturas;
GROUP BY numero

7. INSERT INTO montofacturas (numero,cantidad)
SELECT DISTINC numero AS numerodefactura ,
(precioporunidad*Cantidad) AS total
FROM facturas;
GROUP BY numero;

8. SELECT * FROM montofacturas;

EJERCICIO 5:

1.

```
DROP TABLE socios;
DROP TABLE deportes;
DROP TABLE inscritos;
```

2.

```
CREATE TABLE socios (
        documento char(8) NOT NULL,
        nombre varchar(30) DEFAULT NULL,
        domicilio varchar(30) DEFAULT NULL,
        PRIMARY KEY (documento)
);

CREATE TABLE deportes (
        codigo tinyint unsigned NOT NULL auto_increment,
        nombre varchar(15) DEFAULT NULL,
        profesor varchar(30) DEFAULT NULL,
        PRIMARY KEY (codigo)
);

CREATE TABLE inscritos (
        documento char(8) NOT NULL,
        codigodeporte varchar(20) NOT NULL,
        ao year NOT NULL,
        cuota char(1) DEFAULT NULL,
        PRIMARY KEY (documento,codigodeporte,ao)
);
```

3.

```
INSERT INTO socios VALUES (
        "22333444","Juan Perez","Colon 123";
        "23333444","Ana Lopez","Caseros 984";
        "24333444","Marcelo Morales","Sucre 356";
        "25333444","Susana Juarez","Sarmiento 723";
);

INSERT INTO deportes VALUES (
        "1","tennis","Tadeo Torres";
        "2","natacion","Natalia Nores";
        "3","basquet","Bautista Pereyra";
        "4","paddle","Bautista "
);

INSERT INTO inscritos VALUES (
        "22333444","1","2015","s";
```

```
        "22333444","1","2016","n";
        "22333444","2","2015","s";
        "23333444","1","2015","s";
        "23333444","1","2016","s";
        "23333444","2","2016","s";
        "24333444","1","2016","s";
        "24333444","3","2016","n";
);
```

4.

```
SELECT ao,count(documento) AS alumnos,deportes.nombre
FROM inscritos LEFT JOIN deportes
ON deportes.codigo=inscritos.codigodeporte
GROUP BY codigodeporte;
```

5. DROP TABLE inscritospordeporteporaño;

6.

```
CREATE TABLE inscritospordeporteporaño (
        deporte varchar(15) DEFAULT NULL,
        ao year DEFAULT NULL,
        cantidad tinyint unsigned DEFAULT NULL
);
```

7.

```
INSERT INTO inscritospordeporteporaño (deporte,ao,cantidad)
SELECT ao,count(documento) AS alumnos,d.nombre
FROM inscritos AS i LEFT JOIN deportes AS d
ON d.codigo=i.codigodeporte
GROUP BY codigodeporte;
```

8.

```
SELECT s.nombre,d.nombre,ao
FROM inscritos AS i JOIN deportes AS d
ON d.codigo=i.codigodeporte JOIN socios AS s
ON s.documento=i.documento
WHERE i.cuota="s";
```

9. DROP TABLE sociosdeudores;

10.

```
CREATE TABLE sociosdeudores (
        socios varchar(30),
        deporte varchar(15),
        ao year
);
```

11.
```
INSETRT INTO sociosdeudores (socio,deporte)
SELECT s.nombre,d.nombre,ao
FROM inscritos as i JOIN deportes as d
ON d.codigo=i.codigodeporte JOIN socios as s
ON s.documento=i.documento
WHERE ii.cuota="s";
```

12.
```
INSETRT INTO sociosdeudores
SELECT s.nombre,d.nombre,ao
FROM inscritos as i JOIN deportes as d
ON d.codigo=i.codigodeporte JOIN socios as s
ON s.documento=i.documento
WHERE ii.cuota="s";
```

13.
```
SELECT *
FROM sociosdeudores;
```

14.
```
SELECT d.profesor,d.nombre,COUNT(s.nombre) AS Alumnos
FROM deportes AS d JOIN inscritos AS i
ON d.codigo=i.codigodeporte
JOIN socios AS s
ON s.documento=i.documento
GROUP by d.profesor;
```

15.
```
DROP TABLE alumnosporprofesor;
```

16.
```
CREATE TABLE alumnosoporprofesor(
        profesor VARCHAR(30),
        deporte VARCHAR(15),
        cantidad tinyint unsigned
);
```

17.
```
INSERT INTO alumnosporprofesor
SELECT d.profesor,d.nombre,COUNT(s.nombre) AS Alumnos
FROM deportes AS d JOIN inscritos AS i
ON d.codigo=i.codigodeporte
JOIN socios AS s
ON s.documento=i.documento
GROUP BY d.profesor;
```

18.
```
SELECT * FROM alumnosporprofesor;
```

EJERCICIO 6:

1. DROP TABLE clientes; DROP TABLE provincias;

2.

```
CREATE TABLE clientes (
        codigo INT UNSIGNED AUTO_INCREMENT,
        nombre VARCHAR(30),
        domicilio VARCHAR(30) NULL,
        ciudad VARCHAR(30) NULL,
        codigoprovincia TINYINT UNSIGNED NULL,
        telefono VARCHAR(11) NULL,
        PRIMARY KEY (codigo)
);

CREATE TABLE provincias(
        codigo TINYINT UNSIGNED AUTO_INCREMENT,
        nombre VARCHAR(20) NULL,
        codigoprueba TINYINT UNSIGNED,
        PRIMARY KEY (codigo)
);
```

3.

```
INSERT INTO clientes VALUES
(1,"Lopez Marcos","Colon 111","Cordoba",1,NULL),
(2,"Perez Ana","San Martin 222","Cruz del Eje",1,4578585),
(3,"Garcia Juan","Rivadavia 333","Villa Maria",1,4578445),
(4,"Perez Luis","Sarmiento 444","Rosario",2,NULL),
(5,"Pereyra Lucas","San Martin 555","Creuz del Eje",1,4253685),
(6,"Gomez Ines","San Martin 666","Santa Fe",2,0345252525),
(7,"Torres Fabiola","Alem 777","Villa del Rosario",1,4554455),
(8,"Lopez Carlos","Irigoyen 888","Cruz del Eje",1,NULL),
(9,"Ramos Betina","San Martin 999","Cordoba",1,4223366),
(10,"Lopez Lucas","San Martin 1010","Posadas",4,0457858745);

INSERT INTO provincias VALUES
(1,"Cordoba"),
(2,"Santa Fe"),
(3,"Corrientes"),
(4,"Misiones"),
(5,"Salta"),
(6,"Buenos Aires"),
(7,"Neuquen");
```

4.

```
ALTER  TABLE clientes ADD nombreprovincia varchar(30);

UPDATE clientes
JOIN provincias
ON provincias.codigo=clientes.codigoprovincia
SET clientes.nombreprovincia=provincias.nombre;

DROP TABLEprovincias;

ALTER TABLE clientes DROP codigoprovincia;
```

5. ALTER TABLE clientes ADD nombreprovincia varchar(30);

6.

```
UPDATE clientes
JOIN provincias
ON provincias.codigo=clientes.codigoprovincia
SET clientes.nombreprovincia=provincias.nombre;
```

7. DROP TABLE provincias; ALTER TABLE clientes DROP codigoprovincia;

EJERCICIO 7:

1.

```
DROP TABLE clientes; DROP TABLE provincias;
```

2.

```
CREATE TABLE clientes (
        codigo INT UNSIGNED AUTO_INCREMENT,
        nombre VARCHAR(30) NOT NULL,
        domicilio VARCHAR(30) DEFAULT NULL,
        ciudad VARCHAR(20) DEFAULT NULL,
        codigoprovincia TINYINT,
        telefono VARCHAR(11) DEFAULT NULL,
        PRIMARY KEY (codigo)
);

CREATE TABLE provincias (
        codigo TINYINT UNSIGNED AUTO_INCREMENT,
        nombre VARCHAR(20) DEFAULT NULL,
        PRIMARY KEY (codigo)
);
```

3.

```
INSERT INTO clientes VALUES
        (1,"Lopez Marcos","Colon 111","Cordoba",1,NULL),
        (2,"Perez Ana","San Martin 222","Cruz del Eje",1,4578585),
        (3,"Garcia Juan","Rivadavia 333","Villa Maria",1,4578445),
        (4,"Perez Luis","Sarmiento 444","Rosario",2,NULL),
        (5,"Pereyra Lucas","San Martin 555","Creuz del Eje",1,4253685),
        (6,"Gomez Ines","San Martin 666","Santa Fe",2,0345252525),
        (7,"Torres Fabiola","Alem 777","Villa del Rosario",1,4554455),
        (8,"Lopez Carlos","Irigoyen 888","Cruz del Eje",1,NULL),
        (9,"Ramos Betina","San Martin 999","Cordoba",1,4223366),
        (10,"Lopez Lucas","San Martin 1010","Posadas",4,0457858745);

INSERT INTO provincias VALUES
        (1,"Cordoba"),
        (2,"Santa Fe"),
        (3,"Corrientes"),
        (4,"Misiones"),
        (5,"Salta"),
        (6,"Buenos Aires"),
        (7,"Neuquen");
```

4.

```
UPDATE  clientes
JOIN provincias
ON provincias.codigo=clientes.codigoprovincia
SET provincias.codigo = 10, clientes.codigoprovincia = 10
WHERE provincias.nombre = "Cordoba";
```

5.

```
SELECT clientes.*, provincias.* FROM clientes,provincias;
```

EJERCICIO 8:

1.

```
DROP TABLE clientes; DROP TABLE provincias;
```

2.

```
CREATE TABLE clientes (
        codigo INT UNSIGNED AUTO_INCREMENT,
        nombre VARCHAR(30) NOT NULL,
        domicilio VARCHAR(30) DEFAULT NULL,
        ciudad VARCHAR(20) DEFAULT NULL,
        codigoprovincia TINYINT,
        telefono VARCHAR(11) DEFAULT NULL,
        PRIMARY KEY (codigo)
);

CREATE TABLE provincias (
        codigo TINYINT UNSIGNED AUTO_INCREMENT,
        nombre VARCHAR(20) DEFAULT NULL,
        PRIMARY KEY (codigo)
);
```

3.

```
INSERT INTO clientes VALUES
        (1,"Lopez Marcos","Colon 111","Cordoba",1,NULL),
        (2,"Perez Ana","San Martin 222","Cruz del Eje",1,4578585),
        (3,"Garcia Juan","Rivadavia 333","Villa Maria",1,4578445),
        (4,"Perez Luis","Sarmiento 444","Rosario",2,NULL),
        (5,"Pereyra Lucas","San Martin 555","Creuz del Eje",1,4253685),
        (6,"Gomez Ines","San Martin 666","Santa Fe",2,0345252525),
        (7,"Torres Fabiola","Alem 777","Villa del Rosario",1,4554455),
        (8,"Lopez Carlos","Irigoyen 888","Cruz del Eje",1,NULL),
        (9,"Ramos Betina","San Martin 999","Cordoba",1,4223366),
        (10,"Lopez Lucas","San Martin 1010","Posadas",4,0457858745);

INSERT INTO provincias VALUES
        (1,"Cordoba"),
        (2,"Santa Fe"),
        (3,"Corrientes"),
        (4,"Misiones"),
        (5,"Salta"),
        (6,"Buenos Aires"),
        (7,"Neuquen");
```

4.

```
DELETE clientes
FROM  clientes
JOIN  provincias
ON clientes.codigoprovincia=provincia.codigo
WHERE provincia.nombre='Santa Fe';
```

5.

```
SELECT clientes.*, provincias.* FROM clientes,provincias;
```

EJERCICIO 9:

1. DROP TABLE deportes; DROP TABLE profesores; DROP TABLE inscritos;

2.

```
CREATE TABLE deportes (
        codigo tinyint unsigned auto_increment,
        nombre varchar(15) DEFAULT NULL,
        profesor varchar(30) DEFAULT NULL,
        PRIMARY KEY (codigo)
);

CREATE TABLE profesores (
        codigo tinyint unsigned auto_increment,
        nombre varchar(30) DEFAULT NULL,
        domicilio varchar(30) DEFAULT NULL,
        PRIMARY KEY (codigo)
);

CREATE TABLE inscritos (
        numerosocio int unsigned NOT NULL,
        deporte tinyint unsigned NOT NULL,
        ao year NOT NULL,
        cuota char(1) DEFAULT NULL,
        PRIMARY KEY (numerosocio,deporte,ao)
);
```

3.

```
INSERT INTO deportes VALUES
        (1,"tenis",1),
        (2,"natacion",2),
        (3,"basquet",3),
        (4,"futbol",1);

INSERT INTO profesores VALUES
        (1,"Alfredo Perez","Sarcamiento 983"),
        (2,"Betina Molina","Sucre 356"),
        (3,"Carlos Garcia","Urquiza 209"),
        (4,"Daniel Morales","Salta 1234");

INSERT INTO inscritos VALUES
        (102,1,2016,"s"),
        (102,2,2016,"s"),
        (104,2,2016,"s"),
        (104,3,2016,"s"),
        (106,1,2016,"s"),
        (109,2,2016,"s");
```

4.

```
DELETE deportes
FROM deportes
LEFT JOIN inscritos
ON deportes.codigo=inscritos.deporte
WHERE inscritos.deporte IS null;
```

5.

```
DELETE profesores
FROM profesores
LEFT JOIN deportes
ON profesores.codigo=deportes.profesor
WHERE deportes.profesor IS null;
```

EJERCICIO 10:

1. DROP TABLE pacientes; DROP TABLE obrassociales;

2.

```
CREATE TABLE pacientes (
        documento char(8),
        nombre varchar(30),
        domicilio varchar(30),
        codigoobrasocial tinyint unsigned
);

CREATE TABLE obrassociales (
        codigo tinyint unsigned auto_increment,
        nombre varchar(15) DEFAULT NULL,
        domicilio varchar(30) DEFAULT NULL,
        PRIMARY KEY (codigo)
);
```

3.

```
INSERT INTO pacientes VALUES
        ("22333444","Juan Lopez","Guemes 235",1),
        ("23444555","Analia Juarez","Avellaneda 367",1),
        ("24555666","Juan Lopez","Guemes 235",2),
        ("25666777","Jose Ferrero","Urquiza 321",3),
        ("26777888","Hector Garcia","Caseros 411",3),
        ("27888999","Susana Duarte","Peru 211",3);

INSERT INTO obrassociales VALUES
        (1,"PAMI","Colon 345"),
        (2,"IPAM","Sucre 98"),
        (3,"OSDOP","Avellaneda 267");
```

4.

```
DELETE pacientes,obrassociales
FROM obrassociales
JOIN pacientes
ON pacientes.codigoobrasocial=obrassociales.codigo
WHERE obrassociales.nombre='PAMI';
```

5.

```
SELECT * FROM pacientes JOIN obrassociales;
```

TEMA 6

Vistas ----------

EJERCICIO 1:

```
DROP TABLE IF EXISTS empleados;
DROP TABLE IF EXISTS secciones;
```

1.

```
CREATE TABLE secciones(
        codigo int auto_increment primary key,
        nombre varchar(30),
        sueldo decimal(5,2)
);

CREATE TABLE empleados(
        legajo int primary key auto_increment,
        documento char(8),
        sexo char(1),
        apellido varchar(40),
        nombre varchar(30),
        domicilio varchar(30),
        seccion int not null,
        cantidadhijos int,
        estadocivil char(10),
        fechaingreso date
);
```

2.

```
        INSERT INTO secciones(nombre,sueldo) VALUES
                ('Administracion', 300),
                ('Contaduría', 400),
                ('Sistemas', 500)
        ;

        INSERT INTO empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechain
greso)   VALUES
('22222222','f','Lopez','Ana','Colon 123',1,2,'casado','1990-10-10'),
('23333333','m','Lopez','Luis','Sucre 235',1,0,'soltero','1990-02-10'),
('24444444','m','Garcia','Marcos','Sarmiento 1234',2,3,'divorciado','1998-07-12'),
('25555555','m','Gomez','Pablo','Bulnes 321',3,2,'casado','1998-10-09'),
('26666666','f','Perez','Laura','Peru 1254',3,3,'casado','2000-05-09')
;
```

3.

```
DROP VIEW if exists vista_empleados;
```

4.

```
CREATE VIEW vista_empleados AS
        SELECT CONCAT(apellido,' ',e.nombre) AS nombre,  sexo,
                s.nombre AS seccion, cantidadhijos
        FROM empleados AS e
        JOIN secciones AS s ON codigo=seccion;
```

5.

```
SELECT nombre, seccion, cantidadhijos FROM vista_empleados;
```

6.

```
SELECT seccion, COUNT(*) AS cantidad
FROM vista_empleados
GROUP BY seccion;
```

7.

```
DROP VIEW IF EXISTS vista_empleados_ingreso;
```

8.

```
CREATE VIEW vista_empleados_ingreso(fecingreso,cantidad) AS
SELECT  EXTRACT(year from fechaingreso) AS fecingreso,
COUNT(*) AS cantidad
FROM empleados
GROUP BY fecingreso;
```

9.

```
SELECT fecingreso,cantidad FROM vista_empleados_ingreso;
```

EJERCICIO 2:

```
DROP TABLE IF EXISTS empleados;
DROP TABLE IF EXISTS secciones;
```

1.

```
CREATE TABLE secciones(
        codigo int auto_increment primary key,
        nombre varchar(30),
        sueldo decimal(5,2)
);

CREATE TABLE empleados(
        legajo int primary key auto_increment,
        documento char(8),
        sexo char(1),
        apellido varchar(40),
        nombre varchar(30),
        domicilio varchar(30),
        seccion int not null,
        cantidadhijos int,
        estadocivil char(10),
        fechaingreso date
);
```

2.

```
        INSERT INTO secciones(nombre,sueldo) VALUES
                ('Administracion', 300),
                ('Contaduría', 400),
                ('Sistemas', 500)
        ;

        INSERT INTO empleados
(documento,sexo,apellido,nombre,domicilio,seccion,cantidadhijos,estadocivil,fechain
greso)   VALUES
('22222222','f','Lopez','Ana','Colon 123',1,2,'casado','1990-10-10'),
('23333333','m','Lopez','Luis','Sucre 235',1,0,'soltero','1990-02-10'),
('24444444','m','Garcia','Marcos','Sarmiento 1234',2,3,'divorciado','1998-07-12'),
('25555555','m','Gomez','Pablo','Bulnes 321',3,2,'casado','1998-10-09'),
('26666666','f','Perez','Laura','Peru 1254',3,3,'casado','2000-05-09')
;
```

3.

```
DROP VIEW if exists vista_empleados;
```

4.

```
CREATE VIEW vista_empleados AS
        SELECT CONCAT(apellido,' ',e.nombre) AS nombre,  sexo,
                s.nombre AS seccion, cantidadhijos
        FROM empleados AS e
        JOIN secciones AS s ON codigo=seccion;
```

5.

```
SELECT nombre, seccion, cantidadhijos FROM vista_empleados;
```

6.

```
CREATE VIEW vista_empleados_con_hijos AS
        SELECT nombre, sexo, seccion, cantidadhijos
        FROM vista_empleados
        WHERE cantidadhijos>0;
```

7.

```
SELECT * FROM vista_empleados_con_hijos;
```

EJERCICIO 2:

1.
```
DROP TABLE IF EXISTS alumnos;
DROP TABLE IF EXISTS profesores;
```

2.
```
CREATE TABLE alumnos(
        documento char(8),
        nombre varchar(30),
        nota decimal(4,2),
        codigoprofesor int,
        primary key(documento)
);

CREATE TABLE profesores (
        codigo int auto_increment,
        nombre varchar(30),
        primary key(codigo)
);
```

3.
```
INSERT INTO alumnos values('30111111','Ana Algarbe', 5.1, 1);
INSERT INTO alumnos values('30222222','Bernardo Bustamante', 3.2, 1);
INSERT INTO alumnos values('30333333','Carolina Conte',4.5, 1);
INSERT INTO alumnos values('30444444','Diana Dominguez',9.7, 1);
INSERT INTO alumnos values('30555555','Fabian Fuentes',8.5, 2);
INSERT INTO alumnos values('30666666','Gaston Gonzalez',9.70, 2);

INSERT INTO profesores(nombre) values ('Maria Luque');
INSERT INTO profesores(nombre) values ('Jorje Dante');
```

4.
```
DROP VIEW if exists vista_alumnosaprobados;
```

5.
```
CREATE VIEW vista_alumnosaprobados AS
        SELECT documento,
                a.nombre AS nombrealumno,
                p.nombre AS nombreprofesor,
                nota,
                codigoprofesor
            FROM alumnos AS a
            JOIN profesores AS p
            ON a.codigoprofesor=p.codigo
            WHERE nota>=7
            WITH CHECK OPTION;
```

6.

```
SELECT * FROM vista_alumnosaprobados;
```

7.

```
INSERT  vista_alumnosaprobados(documento, nombrealumno, nota,
codigoprofesor)
VALUES ('12345678','Juan Juanito', 8 , 1);
```

8.

```
SELECT * FROM vista_alumnosaprobados;
```

9.

```
SELECT* FROM alumnos;
```

10.

```
UPDATE vista_alumnosaprobados SET nota=1
WHERE documento='12345678';
```

11.

```
SELECT * FROM alumnos;
```

EJERCICIO 3:

1. DROP TABLE IF EXISTS empleados;

2.
```
CREATE TABLE empleados(
        documento char(8),
        nombre varchar(20),
        apellido varchar(20),
        sueldo decimal(6,2),
        cantidadhijos int,
        seccion varchar(20),
        primary key(documento)
);
```

3.

```
INSERT INTO empleados values('22222222','Juan','Perez', 300.00, 2,'Contaduria');
INSERT INTO empleados values('22333333','Luis','Lopez', 300.00, 1,'Contaduria');
INSERT INTO empleados values('22444444','Marta','Perez', 500.00, 1,'Sistemas');
INSERT INTO empleados values('22555555','Susana','Garcia', 400.00, 2,'Secretaria');
INSERT INTO empleados values('22666666','Jose Maria','Morales', 400.00, 3,'Secretaria');
```

4.
```
DROP PROCEDURE IF EXISTS pa_empleados_sueldo;
```

5.
```
DELIMITER //
CREATE PROCEDURE pa_empleados_sueldo()
        BEGIN
                SELECT nombre,sueldo FROM empleados;
        END //
DELIMITER ;
```
6.
```
CALL pa_empleados_sueldo();
```

7.
```
DELIMITER //
CREATE PROCEDURE pa_empleados_hijos()
        BEGIN
                SELECT nombre,apellido,cantidadhijos FROM empleados
                WHERE cantidadhijos>0;
        END //
DELIMITER ;
```
8.
```
CALL pa_empleados_hijos()
```

9.

```
UPDATE empleados SET cantidadhijos=0
WHERE documento='22222222';

CALL pa_empleados_hijos();
```

EJERCICIO 4:

```
DROP TABLE IF EXISTS empleados;
```

1.

```
CREATE TABLE empleados(
        documento char(8),
        nombre varchar(20),
        apellido varchar(20),
        sueldo decimal(6,2),
        cantidadhijos int,
        seccion varchar(20),
        primary key(documento)
);
```

2.

```
INSERT INTO empleados values('22222222','Juan','Perez', 300.00, 2,'Contaduria');
INSERT INTO empleados values('22333333','Luis','Lopez', 300.00, 0,'Contaduria');
INSERT INTO empleados values('22444444','Marta','Perez', 500.00, 1,'Sistemas');
INSERT INTO empleados values('22555555','Susana','Garcia', 400.00, 2,'Secretaria');
INSERT INTO empleados values('22666666','Jose Maria','Morales', 400.00, 3,'Secretaria');
```

3.

```
DROP PROCEDURE IF EXISTS pa_empleados_sueldo;
```

4.

```
DELIMITER //
CREATE PROCEDURE pa_empleados_sueldo(in p_sueldo decimal(6,2))
        BEGIN
                SELECT nombre,apellido,sueldo FROM empleados
                WHERE sueldo>p_sueldo;
        END //
DELIMITER ;
```

5.

```
CALL pa_empleados_sueldo(300.00);
CALL pa_empleados_sueldo(100.00);
```

6.

```
CALL pa_empleados_sueldo();
```

7. DROP PROCEDURE IF EXISTS pa_empleados_actualizar_sueldo;

8.

```
DELIMITER //
 CREATE PROCEDURE pa_empleados_actualizar_sueldo
(IN p_sueldo decimal(6,2), IN p_nuevosueldo decimal(6,2))
        BEGIN
                UPDATE empleados SET sueldo=p_nuevosueldo
                WHERE sueldo = p_sueldo;
        END //
DELIMITER ;
```

9.

```
CALL pa_empleados_actualizar_sueldo(400.00, 1000.00);
```

10.

```
CALL pa_empleados_actualizar_sueldo(400.00);
```

EJERCICIO 5:

```
DROP TABLE IF EXISTS empleados;
```

1.

```
CREATE TABLE empleados(
        documento char(8),
        nombre varchar(20),
        apellido varchar(20),
        sueldo decimal(6,2),
        cantidadhijos int,
        seccion varchar(20),
        primary key(documento)
);
```

2.

```
INSERT INTO empleados values('22222222','Juan','Perez', 300.00, 2,'Contaduria');
INSERT INTO empleados values('22333333','Luis','Lopez', 700.00, 0,'Contaduria');
INSERT INTO empleados values('22444444','Marta','Perez', 500.00, 1,'Sistemas');
INSERT INTO empleados values('22555555','Susana','Garcia', 400.00, 2,'Secretaria');
INSERT INTO empleados values('22666666','Jose Maria','Morales', 1200.00, 3,'Secretaria');
```

3.

```
DROP PROCEDURE IF EXISTS pa_seccion;
```

4.

```
DELIMITER //
 CREATE PROCEDURE pa_seccion(
   IN p_seccion varchar(30),
   OUT maximo decimal(6,2),
   OUT promedio decimal(6,2))
 BEGIN

   SELECT nombre,apellido
     FROM empleados
WHERE empleados.seccion = p_seccion;
   SELECT max(sueldo) INTO maximo
    WHERE empleados.seccion = p_seccion;
   SELECT avg(sueldo) INTO promedio
     FROM empleados
     WHERE empleados.seccion = p_seccion;

 END //
 DELIMITER ;
```

5.

```
CALL pa_seccion(“Contaduria”);
CALL pa_seccion(“Sistemas”);
```

EJERCICIO 6:

3.

```
CREATE PROCEDURE pa_cantidadProv(IN provincia1 VARCHAR(20), IN provincia2
VARCHAR(20))
        BEGIN
        DECLARE cant1 int;
        DECLARE cant2 int;
         SELECT count(*) FROM provincias INNER JOIN clientes ON
provincias.codigo=clientes.codigoprovincia
         WHERE provincias.nombre = provincia1
         INTO cant1;
         SELECT COUNT(*) FROM provincias INNER JOIN clientes ON
provincias.codigo=clientes.codigoprovincia
         WHERE provincias.nombre = provincia2
         INTO cant2;
         SELECT cant1, cant2;
         END //
        DELIMITER ;
```

4.

```
 CALL pa_cantidadProv("Cordoba", "Santa Fe");
```

EJERCICIO 7:

3.

```
CREATE PROCEDURE pa_cantidadProv(IN provincia1 VARCHAR(20), IN provincia2
VARCHAR(20))
        BEGIN
        DECLARE cant1 int;
        DECLARE cant2 int;
         SELECT count(*)  INTO cant1 FROM provincias INNER JOIN clientes ON
provincias.codigo=clientes.codigoprovincia
         WHERE provincias.nombre = provincia1;

         SELECT COUNT(*) INTO cant2 FROM provincias INNER JOIN clientes ON
provincias.codigo=clientes.codigoprovincia
         WHERE provincias.nombre = provincia2;

        IF cant1>cant2 THEN
        SELECT provincia1,cant1;
        ELSEIF  cant1<cant2 THEN
        SELECT provincia2,cant2;
        ELSE
        SELECT provincia1,cant1,provincia2,cant2;
        END IF;
         END //
        DELIMITER ;
```

4.

```
 CALL pa_cantidadProv("Cordoba", "Santa Fe");
```

EJERCICIO 8:

3.

```
CREATE PROCEDURE pa_cantidadProv(IN provincia1 VARCHAR(20), IN provincia2
VARCHAR(20))
        BEGIN
        DECLARE cant1 int;
        DECLARE cant2 int;
         SELECT count(*)  INTO cant1 FROM provincias INNER JOIN clientes ON
provincias.codigo=clientes.codigoprovincia
         WHERE provincias.nombre = provincia1;

         SELECT COUNT(*) INTO cant2 FROM provincias INNER JOIN clientes ON
provincias.codigo=clientes.codigoprovincia
         WHERE provincias.nombre = provincia2;

        CASE
        WHEN cant1>cant2 then
        SELECT provincia1,cant1;
        WHEN cant1<cant2 then
        SELECT provincia2,cant2;
        ELSE
        SELECT provincia1,cant1,provincia2,cant2;
        END CASE;
         END //
        DELIMITER ;
```

4.

```
 CALL pa_cantidadProv("Cordoba", "Santa Fe");
```

Ejercicios PL 1:

1) Escribir un procedimiento que reciba dos números y visualice su suma.

```
 create procedure pa_Suma(in p_num1 int, in p_num2 int)
 begin
 declare suma int;
 set suma=p_num1+p_num2;
select suma;
 end //
 delimiter ;

Call pa_Suma("3","2");
```

2) Codificar un procedimiento que reciba una cadena y la visualice al revés.

```
DELIMITER //
 CREATE PROCEDURE pa_Invertir(IN p_cadena VARCHAR(30))
 BEGIN
        SELECT reverse(p_cadena);
 END //
 DELIMITER ;

CALL pa_Invertir("Hola Mundo");
```

3) Escribir una función que reciba una fecha y devuelva el año, en número, correspondiente a esa fecha.

```
DELIMITER //
 CREATE FUNCTION f_anio(fecha DATE)
RETURNS INT
DETERMINISTIC
 BEGIN
        DECLARE anio INT;
SET anio  = year(fecha);
        RETURN anio;
 END //
 DELIMITER ;
```

4) Escribir un bloque PL/SQL que haga uso de la función anterior.

```
DELIMITER //
CREATE PROCEDURE pa_anio (IN p_fecha1 DATE, IN p_fecha2 DATE, OUT anio int)
BEGIN
DECLARE anio1 int;
DECLARE anio2 int;
SELECT f_anio(p_fecha1) INTO anio1;
SELECT f_anio(p_fecha2) INTO anio2;
CASE
WHEN anio1>anio2 then
        SET anio=anio1;
        SELECT anio1;
WHEN anio1<anio2 then
        SET anio = anio2;
        SELECT anio2;

ELSE
        SET anio = anio1;
        SELECT anio;
END CASE;
        END //
DELIMITER ;



CALL pa_anio('1753-01-01','1759-01-01',@resu);



DROP PROCEDURE pa_anio;
```

Ejercicios PL 2:

1) Escribir un bloque PL/SQL que haga uso de la función anterior (último ejercicios PL1).

Esta función realiza la comparación de los años que les pasas, te devuelve el mayor y a este le sumas un int para indicar el año en el que estarás al sumar ese entero.

Ej: Año mayor = 1900 -> Int introducido = 99 -> Resultado será 1999.

```
DELIMITER //
CREATE PROCEDURE pa_llamarPl1(IN p_fecha1 DATE, IN p_fecha2 DATE,IN p_edad
INT)
BEGIN
        DECLARE anioActual INT;
        DECLARE anio INT;
CALL pa_anio(p_fecha1,p_fecha2,@resu);
SET anio = @resu;
SET anioActual = anio + p_edad;
SELECT anioActual;
END //
DELIMITER ;
```

2) Desarrollar una función que devuelva el número de años completos que hay entre dos fechas que se pasan como argumentos.

```
DELIMITER //
CREATE PROCEDURE pa_anio2 (IN p_fecha1 DATE, IN p_fecha2 DATE, OUT nanio int)
BEGIN
DECLARE anio1 int;
DECLARE anio2 int;
SELECT f_anio(p_fecha1) INTO anio1;
SELECT f_anio(p_fecha2) INTO anio2;
CASE
WHEN anio1>anio2 then
        SET nanio=anio1-anio2;
WHEN anio1<anio2 then
        SET nanio = anio2-anio1;
ELSE
        SET nanio = 0;
END CASE;
SELECT nanio;
        END //
DELIMITER ;
```

3) Escribir una función que, haciendo uso de la función anterior devuelva los trienios que hay entre dos fechas. (Un trienio son tres años completos).

```
DELIMITER //

CREATE PROCEDURE pa_llamar2(IN p_fecha1 DATE, IN p_fecha2 DATE,OUT trienios
INT)

BEGIN

        DECLARE anio1 INT;

        DECLARE anio2 INT;

CALL pa_anio2(p_fecha1,p_fecha2,@resultado);

SET trienios = @resultado/3;

SELECT trienios;

END //

DELIMITER ;


CALL pa_llamar2('1753-01-01','1760-01-01',@resultado);


DROP PROCEDURE pa_llamar2;
```

Ejercicios PL 3:

1) Codificar un procedimiento que reciba una lista de hasta 5 números y visualice su suma.

```
DELIMITER //

CREATE PROCEDURE pa_sumar(IN p_num1 INT, IN p_num2 INT,IN p_num3 INT,IN
p_num4 INT,IN p_num5 INT,OUT resultado INT)

BEGIN

SET resultado = p_num1+p_num2+p_num3+p_num4+p_num5;

SELECT resultado;

END //

DELIMITER ;



CALL pa_sumar(12,23,45,10,9,@resultado);
```

2) Escribir una función que devuelva solamente caracteres alfabéticos sustituyendo cualquier otro carácter por blancos a partir de una cadena que se pasará en la llamada.

```
DELIMITER //
 CREATE FUNCTION f_sustituir(texto VARCHAR(100))
RETURNS VARCHAR(100)
DETERMINISTIC
 BEGIN
        DECLARE resultado VARCHAR(100);
        SELECT REGEXP_REPLACE(texto, '[^a-z]' ,' ') INTO resultado;
        RETURN resultado;
 END //
 DELIMITER ;
------------------

DELIMITER //
 CREATE FUNCTION f_sustituir(texto VARCHAR(100))
         RETURNS VARCHAR(100)
         DETERMINISTIC
        BEGIN
                DECLARE resultado VARCHAR(100);
        SELECT REGEXP_REPLACE(texto, '[^a-z]' ,' ') INTO resultado;
        RETURN resultado;
         END //
 DELIMITER ;
SELECT f_sustituir('buenas8tard.es');
```

3) Implementar un procedimiento que reciba un importe y visualice el desglose del cambio en unidades monetarias de 1, 5, 10, 25, 50, 100, 200, 500, 1000, 2000, 5000 Ptas. en orden inverso al que aparecen aquí enumeradas.

```
DELIMITER //

CREATE PROCEDURE pa_cambio(

   in dinero INT)

BEGIN

   DECLARE strCambio VARCHAR(200);

   DECLARE cambioActual INT;

   DECLARE contador INT;

   DECLARE contadorIrBajando INT;

   DECLARE fin INT;

   SET fin=0;

   SET strCambio="";

   SET cambioActual=5000;

   SET contador = 0;

   SET contadorIrBajando = 3;

   loop_for: LOOP

         WHILE dinero >= cambioActual DO

               CASE cambioActual

                     WHEN 20 THEN

                           SET contador=contador+1;

                           SET dinero=dinero-25;

                     WHEN 2 THEN

                           SET contador=dinero;

                           SET dinero=0;

                            SET cambioActual=1;

                     ELSE
```

```
                    SET contador=contador+1;

                    SET dinero=dinero-cambioActual;

            END CASE;

        END WHILE;

        IF contador > 0 THEN

            IF cambioActual = 20 THEN

                SELECT CONCAT(strCambio, contador, " monedas de ", 25, ".  ")
INTO strCambio;

            ELSE

                IF cambioActual > 500 THEN

                    SELECT CONCAT(strCambio, contador, " billetes de ",
cambioActual, ".  ") INTO strCambio;

                ELSE

                    SELECT CONCAT(strCambio, contador, " monedas de ",
cambioActual, ".  ") INTO strCambio;

                END IF;

            END IF;

            SET contador=0;

        END IF;

        IF cambioActual = 1 THEN

            LEAVE loop_for;

        END IF;

        IF contadorIrBajando%3 = 0 THEN

            SET cambioActual = cambioActual * 0.4;

            SET contadorIrBajando = contadorIrBajando +1;

        ELSE

            SET cambioActual = cambioActual / 2;

            SET contadorIrBajando = contadorIrBajando +1;
```

```
            END IF;
            ITERATE loop_for;
    END LOOP loop_for;
    SELECT strCambio;
END //
DELIMITER ;
```

Ejercicios PL 4:

1) Codificar un procedimiento que permita borrar un empleado cuyo número se pasará en la llamada.

```
DELIMITER //

CREATE PROCEDURE pa_borrar(IN p_codigoempleado INT)

BEGIN

DELETE empleados

  FROM empleados

  WHERE empleados.codigo=p_codigoempleado;

SELECT * FROM empleados;

END //

DELIMITER ;
```

2) Escribir un procedimiento que modifique la localidad de un departamento. El procedimiento recibirá como parámetros el número del departamento y la localidad nueva.

```
 CREATE TABLE departamento( codigo int,
        localidad varchar(30),
        primary key(codigo));
INSERT INTO departamento(codigo,localidad) VALUES
            (111 , 'La Victoria'),
            (222 , 'La Matanza'),
            (333 , 'Punta Brava');

DELIMITER //

    CREATE PROCEDURE pa_mod_localidad(

        IN p_departamento INT,

        IN p_localidad VARCHAR(30))

    BEGIN

        UPDATE departamento SET departamento.localidad = p_localidad

        WHERE departamento.codigo = p_departamento;

        SELECT * FROM departamento;

    END //

DELIMITER ;
```

Ejercicios Triggers 1

Creación de tablas -> Aquí.

1) Crear un Trigger que borre en cascada sobre la tabla relacionada cuando borremos una sala. Mostrar el registro borrado al ejecutar el Trigger.

```
DROP TRIGGER mostarregistroborrado;

DELIMITER//

CREATE TRIGGER mostarregistroborrado AFTER DELETE ON Sala

FOR EACH ROW

BEGIN

        DELETE FROM Plantilla INNER JOIN Sala

        ON Sala.Sala_Cod=Plantilla.Sala_Cod

        WHERE Sala.Sala_Cod=old.Sala_Cod;

SET resultado = CONCAT("Sala eliminada ",old.Sala_Cod);

END//

DELIMITER ;
```

```
DELIMITER//

CREATE TRIGGER mostarregistroborrado AFTER DELETE ON Sala

FOR EACH ROW

BEGIN

        DECLARE resultado VARCHAR(50);

        DELETE FROM Plantilla

        WHERE Sala.Sala_Cod=old.Sala_Cod;

SET resultado = CONCAT("Sala eliminada ",old.Sala_Cod);

END//

DELIMITER ;
```

2) Crear un Trigger que se active cuando Actualicemos alguna sala del hospital, modificando sus tablas relacionadas. Mostrar el registro Actualizado.

```
DROP TRIGGER mostrarregistroactualizado;

DELIMITER //

CREATE TRIGGER mostrarregistroactualizado AFTER UPDATE ON Sala

FOR EACH ROW

        BEGIN

                DECLARE mensaje varchar(50);

                UPDATE Plantilla

                INNER JOIN Sala

                ON Sala.Sala_Cod=Plantilla.Sala_Cod

                SET Plantilla.Sala_Cod=new.Sala_Cod

                WHERE Sala.Sala_Cod=old.Sala_Cod;

END//

DELIMITER;
```

```
DELIMITER //

CREATE TRIGGER mostrarregistroactualizado AFTER UPDATE ON Sala

FOR EACH ROW

        BEGIN

                UPDATE Plantilla

                SET Plantilla.Sala_Cod=new.Sala_Cod

                WHERE Sala.Sala_Cod=old.Sala_Cod;

END//

DELIMITER;
```

3) Crear un Trigger que se active al eliminar un registro en la tabla hospital y modifique las tablas correspondientes.

```
DROP TRIGGER mostarregistroborrado;

DELIMITER //

CREATE TRIGGER mostarregistroborrado BEFORE DELETE ON Hospital

FOR EACH ROW

        BEGIN

DECLARE mensaje VARCHAR(50);

DELETE FROM Plantilla

INNER JOIN Hospital

ON Hospital.Hospital_Cod=Plantilla.Hospital_Cod

WHERE Hospital.Hospital_Cod=old.Hospital_Cod;


DELETE FROM Doctor

INNER JOIN Hospital

ON Hospital.Hospital_Cod=Doctor.Hospital_Cod

WHERE Hospital.Hospital_Cod=old.Hospital_Cod;


DELETE FROM Sala

INNER JOIN Hospital

ON Hospital.Hospital_Cod=Sala.Hospital_Cod

WHERE Hospital.Hospital_Cod=old.Hospital_Cod;

SET mensaje= CONCAT("Sala eliminada ",old.Hospital_Cod);

END //

DELIMITER;
```

```
DELIMITER //
CREATE TRIGGER mostarregistroborrado BEFORE DELETE ON Hospital
FOR EACH ROW
        BEGIN
DELETE FROM Plantilla WHERE Plantilla.Hospital_Cod=old.Hospital_Cod;
DELETE FROM Doctor WHERE Doctor.Hospital_Cod=old.Hospital_Cod;
DELETE FROM Sala WHERE Sala.Hospital_Cod=old.Hospital_Cod;
END //
DELIMITER;
```

Ejercicios Triggers 2

4) Crear un Trigger para controlar la inserción de empleados, cuando insertemos un empleado se copiarán datos sobre la inserción en una tabla llamada Control_BD. Los datos que se copiarán son el Número de empleado, El usuario que está realizando la operación, la fecha y el tipo de operación.

```
CREATE  TABLE Control_BD (
        User_BD VARCHAR(20) NOT NULL,
        Fecha DATE NOT NULL,
        TipoOP_BD VARCHAR(10) NOT NULL,
        NEmp INT NOT NULL
);
DELIMITER//
CREATE TRIGGER mostarregistroborrado AFTER INSERT ON Emp
FOR EACH ROW
BEGIN
        INSERT INTO Control_BD VALUES (current_user(),NOW( ),'INSERT',new.Emp_No);
END//
DELIMITER ;
```

5) Crear un Trigger que actúe cuando se modifique la tabla hospital y sobre todas las tablas con las que esté relacionadas.

```
DROP TRIGGER mostarregistroborrado;
DELIMITER //
CREATE TRIGGER mostarregistroborrado BEFORE UPDATE ON Hospital
FOR EACH ROW
        BEGIN
                UPDATE FROM Plantilla
                INNER JOIN Hospital
                ON Hospital.Hospital_Cod=Plantilla.Hospital_Cod
                SET Plantilla.Hospital_Cod=new.Hospital_Cod
                WHERE Plantilla.Hospital_Cod=old.Hospital_Cod;
```

```
                UPDATE FROM Doctor
                INNER JOIN Hospital
                ON Hospital.Hospital_Cod=Doctor.Hospital_Cod
                SET Doctor.Hospital_Cod=new.Hospital_Cod
                WHERE Doctor.Hospital_Cod=old.Hospital_Cod;

                UPDATE FROM Sala
                INNER JOIN Hospital
                ON Hospital.Hospital_Cod=Sala.Hospital_Cod
                SET Sala.Hospital_Cod=new.Hospital_Cod
                WHERE Sala. Hospital_Cod=old.Hospital_Cod;
                END //
DELIMITER ;
```

6) Crear un Trigger en la tabla plantilla. Cuando actualicemos la tabla plantilla, debemos comprobar que el hospital que actualizamos existe, si intentamos actualizar el código de hospital, no podremos hacerlo si no existe relación con algún código de hospital. Realizar el mismo Trigger para las tablas relacionadas con Hospital.

```
DELIMITER//

CREATE TRIGGER actualizartablas AFTER UPDATE ON Plantilla

FOR EACH ROW

BEGIN


END//

DELIMITER ;
```

Ejercicio errores

Crea una **base de datos** llamada test que contenga una **tabla** llamada alumno. La tabla debe tener cuatro columnas:

- id: entero sin signo (clave primaria).
- nombre: cadena de 50 caracteres.
- apellido1: cadena de 50 caracteres.
- apellido2: cadena de 50 caracteres.

Una vez creada la base de datos y la tabla deberá **crear un procedimiento** llamado insertar_alumno con las siguientes características. El procedimiento recibe cuatro parámetros de **entrada** (id, nombre, apellido1, apellido2) y los insertará en la tabla alumno. El procedimiento devolverá como **salida** un parámetro llamado error que tendrá un valor igual a 0 si la operación se ha podido realizar con éxito y un valor igual a 1 en caso contrario.

Deberá manejar los errores que puedan ocurrir cuando se intenta insertar una fila que contiene una clave primaria repetida.

```
CREATE DATABASE ejErrores;

USE ejErrores


CREATE TABLE alumnos(
		id int unsigned,
		nombre varchar(50),
		apellido1 varchar(50),
		apellido2 varchar(50),
		primary key(id)
	);
```

```
DELIMITER //

CREATE PROCEDURE procedimientoConHandler(

                in id int,

                in varchar(50) nombre ,

                 in apellido1 varchar(50),

                in apellido2 varchar(50))

BEGIN

DECLARE EXIT HANDLER FOR SQLSTATE '1602' SET @salida='1'

BEGIN

INSERT INTO alumnos VALUES ('Time: ',current_date,'. Error de clave ajena
para el valor= ', parametro1));

SET @salida='0'

END ;

SELECT @salida;

END //

DELIMITER ;
```

Ejercicios Triggers 3

7) Modificar el Trigger del ejercicio 4, utilizando control de errores, si la operación es correcta, se insertará en la tabla de ControlTrigger la hora y campo insertado y que ha sido correcto, en caso contrario, incorrecto.

```
CREATE  TABLE ControlTrigger (

        User_BD VARCHAR(20) NOT NULL,

        Fecha DATE NOT NULL,

        TipoOP_BD VARCHAR(10) NOT NULL,

        NEmp INT NOT NULL

);

DELIMITER//

CREATE TRIGGER mostarregistroborrado AFTER INSERT ON Emp

FOR EACH ROW

BEGIN

DECLARE EXIT HANDLER FOR SQLSTATE '1602' SET @salida='1'

BEGIN

INSERT INTO alumnos VALUES ('Time: ',current_date,'. Error de clave ajena para el valor= ', parametro1));

SET @salida='0'

END ;


END//

DELIMITER ;
```

8) Crear un trigger que guarde los datos en la tabla ControlTrigger cuando se realice la baja de un empleado.

9) Crear un Trigger que guarde los datos en la tabla ControlTrigger cuando se relice una modificación en un empleado. Guardar la hora de la actualización en un campo aparte en la tabla ControlTrigger. (Añadir un campo).

Encriptar:

```
 drop table if exists clientes;


create table clientes(
  nombre varchar(50),
  mail varchar(70),
  tarjetacredito blob,
  primary key (nombre)
);


insert into clientes
  values ('Marcos
Luis','marcosluis@gmail.com',aes_encrypt('5390700823285988','xyz123'));


-- Si accedemos al campo tarjetacredito podemos comprobar que se encuentra
cifrado.
select tarjetacredito from clientes;


-- Para descifrar la tarjeta de crédito debemos conocer la clave de cifrado:
'xyz123':
select cast(aes_decrypt(tarjetacredito, 'xyz123') as char) from clientes;
```