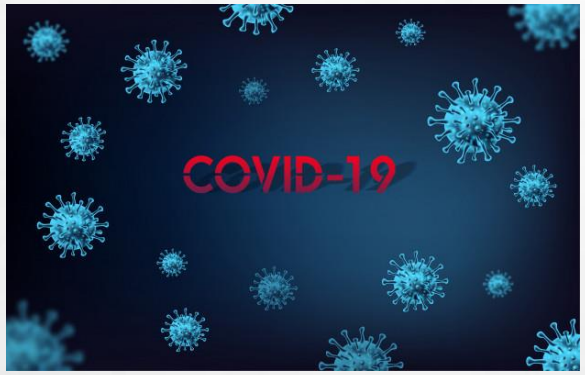
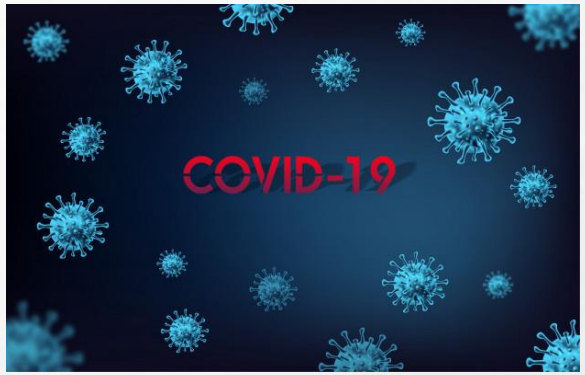




CORONAVIRUS

MICHELLE YUSDAELA GALARZA QUIROGA

BASE DE DATOS II

ING. WILLIAM RODDY BARRA

25 DE JUNIO DE 2020

INFORMACIÓN

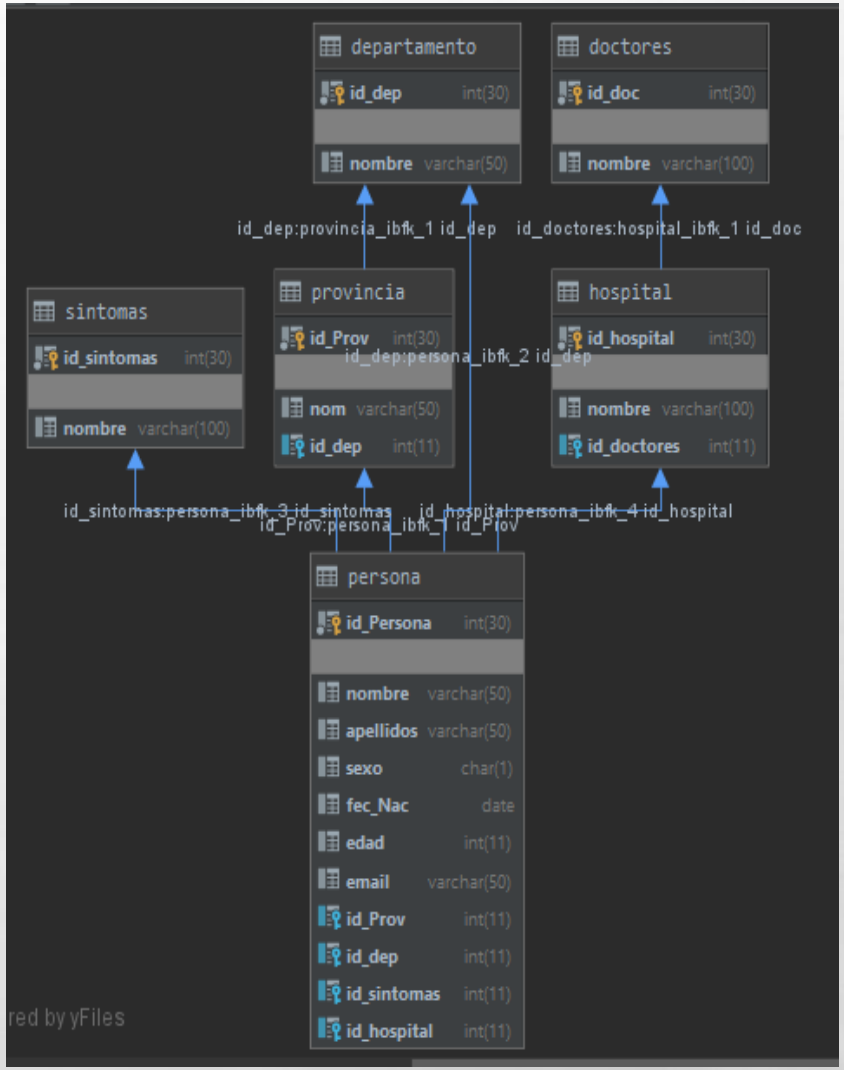
- CORONAVIRUS ES UNA GRAN FAMILIA DE VIRUS CONOCIDOS POR CAUSAR ENFERMEDADES QUE VAN DESDE UN RESFRIADO COMÚN HASTA MANIFESTACIONES CLÍNICAS MÁS SEVERAS COMO LAS OBSERVADAS EN EL SÍNDROME RESPIRATORIO POR EL CORONAVIRUS DE ORIENTE MEDIO (MERS) Y EL SÍNDROME ESPIRATORIO AGUDO GRAVE (SARS).
- UN NUEVO CORONAVIRUS (COVID-19) SE IDENTIFICÓ EN 2019 EN WUHAN, CHINA. ESTE ES UN NUEVO CORONAVIRUS QUE NO SE HA IDENTIFICADO PREVIAMENTE EN HUMANOS.

BASE DE DATOS

DESARROLLE UNA BASE DE DATOS CON EL NOMBRE “CORONAVIRUS”

LA CUAL CONSISTE EN 6 TABLAS

1. **PERSONA** CUENTA CON LOS SIGUIENTES ATRIBUTOS QUE SON : ID, NOMBRE, APELLIDO, SEXO, FECHA NACIMIENO, EDAD, EMAIL
1. **DEPARTAMENTO** CUENTA CON LOS SIGUIENTES ATRIBUTOS QUE SON: ID, NOMBRE
2. **PROVINCIA** CUENTA CON LOS SIGUIENTES ATRIBUTOS QUE SON: ID, NOMBRE
3. **SINTOMAS** CUENTA CON LOS SIGUIENTES ATRIBUTOS QUE SON: ID, NOMBRE
4. **HOSPITAL** CUENTA CON LOS SIGUIENTES ATRIBUTOS QUE SON: ID, NOMBRE
5. **DOCTORES** CUENTA CON LOS SIGUIENTES ATRIBUTOS QUE SON: ID, NOMBRE



```sql
create  database Coronavirus;
use Coronavirus;
create table persona
(
    id_Persona integer(30) AUTO_INCREMENT NOT NULL PRIMARY KEY,
    nombre varchar(50),
    apellidos varchar(50),
    sexo char(1),
    fec_Nac date,
    edad integer,
    email varchar(50),

    id_Prov integer,
    id_dep integer,
    id_sintomas integer,
    id_hospital integer,

    foreign key (id_Prov) references provincia(id_Prov),
    foreign key (id_dep) references departamento(id_dep),
    foreign key (id_sintomas) references sintomas(id_sintomas),
     foreign key (id_hospital) references hospital(id_hospital)
);
insert into  persona ( id_Persona,nombre , apellidos , fec_Nac , edad , email , sexo ,  id_prov, id_dep , id_sintomas, id_hospital )

VALUES ( 1, 'Michelle' , 'Galarza Quiroga' , '1990/12/10' , 28 , 'nombre1gmail.com' , 'F' , 1 , 1 ,3,5),

      (  2,'Juan' , 'Adubiri Colque' , '1991/12/10' , 27 , 'nombre2gmail.com' , 'M' , 1 , 1 ,3, 1),

      (  3,'Soledad' , 'Gamirde Tori' , '1992/12/10' , 26 , 'nombre3gmail.com' , 'F' , 1 , 1,1 ,4),

      ( 4,'Carlos' , 'Cespedes Rocha' , '1993/12/10' , 25 , 'nombre4gmail.com' , 'M' , 1 , 1 ,2,5),

      (  5, 'Manuel' , 'Gómez Quisbert' , '1994/12/10' , 24 , 'nombre5gmail.com' , 'M' , 1 , 1,4 ,3),

      ( 6, 'Carla' , 'Morelia Duya' , '1995/12/10' , 23 , 'nombre6gmail.com' , 'F' , 1 , 1 ,5,4);
```

CÓDIGO DE LA TABLA PERSONA

```sql
create table departamento
(
  id_dep  integer(30) AUTO_INCREMENT NOT NULL PRIMARY KEY,
  nombre varchar(50)
);
insert into departamento (id_dep, nombre)
VALUES ( 1, 'Cochabamba'),
      (2, 'La Paz'),
      (3, 'Santa Cruz'),
      (4, 'Beni'),
      (5, 'Chuquisaca') ,
      (6, 'Potosi'),
      (7, 'Tarija'),
      (8, 'Pando'),
      (9, 'Oruro');
create table provincia
(
    id_Prov  integer(30) AUTO_INCREMENT NOT NULL PRIMARY KEY,
    nom varchar(50),
    id_dep integer,

    foreign key (id_dep) references departamento(id_dep)
);
insert into provincia (id_prov, nom, id_dep)
values (1, 'Quillacollo', 1),
      (2, 'Cercado', 1),
      (3, 'Punata', 1);
```

CÓDIGO DE LA TABLA DEPARTAMENTO Y PROVINCIA

```sql
create  table sintomas
(
  id_sintomas  integer(30) AUTO_INCREMENT NOT NULL PRIMARY KEY,
  nombre varchar(100)
);
insert into sintomas (id_sintomas, nombre)
values (1, 'Tos seca'),
      (2, 'Fiebre'),
      (3, 'Cansancio'),
      (4, 'Dolor de Cabeza'),
      (5, 'Difcultad para respirar');
create  table hospital
(
  id_hospital  integer(30) AUTO_INCREMENT NOT NULL PRIMARY KEY,
  nombre varchar(100),
  id_doctores integer,
  foreign key (id_doctores) references doctores(id_doc)
);
insert into hospital (id_hospital, nombre, id_doctores)
values (1, 'Viedma', 1),
      (2, 'Tiquipaya', 1),
      (3, 'Los Olivos', 3),
      (4, 'Univalle', 2),
      (5, 'Copacabana', 2);
create  table doctores
(
  id_doc integer(30) AUTO_INCREMENT NOT NULL PRIMARY KEY,
  nombre varchar(100)
);
insert into doctores (id_doc, nombre)
values (1, 'Doctor_1'),
      (2, 'Doctor_2'),
      (3, 'Doctor_3'),
      (4, 'Doctor_4'),
      (5, 'Doctor_5');
```

CÓDIGO DE LA TABLA HOSPITAL, DOCTORES Y SINTOMAS

VISTAS

REALICE UNA VISTA PARA GENERAR UNA TABLA VIRTUAL LA CUAL MUSTRA TODOS LOS DATOS PERSONALES.

```
create view vista_Persona as

    select concat_ws(' ', p.nombre, p.apellidos) as NombreCompleto, edad as EdadPersona, fec_Nac as FechaNacimineto, d.nombre
         as Nobre_Ciudad,h.nombre as Nombre_Hospital
    from persona as p
        inner join departamento as d on p.id_dep = d.id_dep
        inner  join hospital h on p.id_hospital = h.id_hospital
    where sexo='F' and d.nombre='Cochabamba' and fec_Nac = '1990-12-10';

select *
from vista_Persona;
```

PRIMERA VISTA GENERADA

```sql
CREATE VIEW Saber_Nivel as
  SELECT concat(per.nombre, ' ', per.apellidos) as NOMBRE_COMPLETO,
         concat(d.nombre,' ', p.nom)  as 'UBICACION',
         getEdad(per.edad) as NIVEL_EDAD, s.nombre AS SINTOMAS

FROM persona as per
  inner join departamento d on per.id_dep = d.id_dep
  inner join provincia p on d.id_dep = p.id_dep
  INNER  JOIN sintomas s on per.id_sintomas = s.id_sintomas
WHERE d.nombre='Cochabamba'
order by per.edad;
select * from Saber_Nivel;
DROP FUNCTION  getEdad;
create function getEdad(edadPer INTEGER) returns text
begin
    declare resp text default '';
    case
            when  edadPer<=17  then set resp='ES MENOR 20% de probabilidad';
            when edadPer>=30 then set resp='ES  ADULTO 70% de probabilidad';
            else set resp = 'EDAD NO SE ENCUENTRA EN EL RANGO';
     end case;
    return resp;
end;
```

SEGUNDA VISTA GENERADA

TRIGGERS

REALICE TRIGGERS PARA CONTROLAR LOS DATOS INGRESADOS.

```
create trigger noInsertHospital
    before insert on hospital for each row
    begin
        declare  test text default '';
        if new.nombre= 'Salomon Klein'
            then
                signal sqlstate '45000' set message_text ='Hospital lleno ingrese otro nombre';

            else
                 set test='Se cargo correctamente';
        end if;
    end;

insert into hospital ( id_hospital, nombre, id_doctores) values ('8','Salomon Klein','1');
```

PRMIER TRIGGER GENERADO

```
create table autitoria_sintomas
(
  operation char(1) not null,
  stamp timestamp not null,
  userid text not null,
  hostname text not null,
  name text not null
);
create  trigger auditoria_sintomas
    before insert on sintomas
    for each row
begin
    insert into autitoria_sintomas(operation, stamp , userid , hostname ,name )select
      'I', now(), user(), @@hostname, new.nombre;
end;
```

SEGUNDO TRIGGER GENERADO

PROCEDIMIENTOS ALMACENADOS

REALICE UN PROCEDIMIENTO ALMACENADO PARA OUTIMIZAR LOS TRIGGERS O DICHAS FUNCIONES.

```
create table autitoria_Hospital
(
  operation char(1) not null,
  stamp timestamp not null,
  userid text not null,
  hostname text not null,
  name text not null
);
create procedure insertAudiHospital( in operacion char(1),
                                in fecTime timestamp,
                                in userid text,
                                in currentHost text,
                                in name text)
begin
    INSERT INTO autitoria_Hospital
        (operation, stamp, userid, hostname, name) SELECT
        operacion, fecTime, userId, currentHost, name;
end;
```

PRIMER PROCEDIMIENTO ALMACENADO

```
create  trigger auditoria_Hospital_insert
    before insert on hospital
    for each row
begin
    call insertAudiHospital('I',
        now(),
        user(),
        @@hostname,
        new.nombre);
end;
create  trigger auditoria_Hospital_update
     after update on hospital
    for each row
begin
    call insertAudiHospital('U',
        now(),
        user(),
        @@hostname,
        new.nombre);
end;
create  trigger auditoria_Hospital_delete
     after delete on hospital
    for each row
begin
    call insertAudiHospital('D',
        now(),
        user(),
        @@hostname,
        OLD.nombre);
end;
```

PRIMER PROCEDIMIENTO ALMACENADO

```
create table Auditoria_antes_despues
(
     operation  CHAR(1)   NOT NULL,
     stamp      TIMESTAMP NOT NULL,
    userid         TEXT   NOT NULL,
    hostname text not null,
    oldNombre text not null,
    newNombre text not null
);

create procedure ProcediAntesDesp( in operation char(1),
                                   in fecTime timestamp,
                                   in userid text,
                                   in currentHost text,
                                   in oldNombre text,
                                   in newNombre text)

begin
    INSERT INTO Auditoria_antes_despues
        (operation, stamp, userid, hostname,oldNombre, newNombre) SELECT
        operation, fecTime, userId,currentHost ,oldNombre, newNombre;
end;
```

SEGUNDO PROCEDIMIENTO ALMACENADO

```sql
CREATE TRIGGER backup_Insert
    AFTER INSERT ON doctores
    FOR EACH ROW
BEGIN
    DECLARE oldNombre TEXT DEFAULT 'No existe estado anterior';
    DECLARE newNombre TEXT DEFAULT NEW.nombre;

    CALL ProcediAntesDesp(
            'I',
        now(),
    user(),
        @@hostname,
            oldNombre,
            newNombre);
END;
drop trigger backud_update;

CREATE TRIGGER backud_update
    AFTER UPDATE ON doctores
    FOR EACH ROW
BEGIN
    DECLARE oldNombre TEXT DEFAULT OLD.nombre;
    DECLARE newNombre TEXT DEFAULT NEW.nombre;

    CALL ProcediAntesDesp(
            'U',
        now(),
    user(),
        @@hostname,
            oldNombre,
            newNombre
        );
END;
CREATE TRIGGER backud_delete
    AFTER delete ON doctores
    FOR EACH ROW
BEGIN
    DECLARE oldNombre TEXT DEFAULT 'No existe estado anterior';
    DECLARE newNombre TEXT DEFAULT OLD.nombre;

    CALL ProcediAntesDesp(
            'D',
        now(),
    user(),
        @@hostname,
            oldNombre,
            newNombre
        );
END;
```

SEGUNDO PROCEDIMIENTO ALMACENADO

GRACIAS