Bases de datos VIAJES: Escribir la sentencia SQL Server para obtener lo que se solicita (20% cada una):

Destinos
id_destino (pk)
Destino

Paquetes
id_paquete (pk)
id_destino
valor_cuota
cant_cuotas
cant_Dias

Pagos
id_pago (pk)
id_pasajero
id_paquete
monto_cuota
Fecha_pago
Fecha_vto
id_f_pago

Pasajeros
id_pasajero (pk)
nombre
apellido
fec_nac
telefono
mail
id_barrio

Barrios
id_barrio (pk)
barrio

F_pagos
id_f_pago (pk)
f_pago

1. Crear una función escalar que devuelva el monto a pagar de acuerdo con el monto_cuota y un recargo de un 1% por día si pagó después de la fecha de vencimiento (la función recibirá tres parámetros: uno de tipo moneda y dos fechas).

```
Create function f_m_pagado
  (@monto_cta money,f_vto datetime, f_pago datetime)
 returns money
 as
  begin
   declare @monto_p money=@monto_cta
   if int(@f_vto) < int(@f_pago)
     set @monto_p = @monto_p *(1+int(@f_pago-@f_vto)/100)
   return @monto_p
  end;
```

2. Crear una vista que muestre el monto total pagado, cantidad de pagos, promedio de monto pagado mensualmente por paquete en los últimos 5 años. Para calcular el monto pagado utilice la función del punto 1

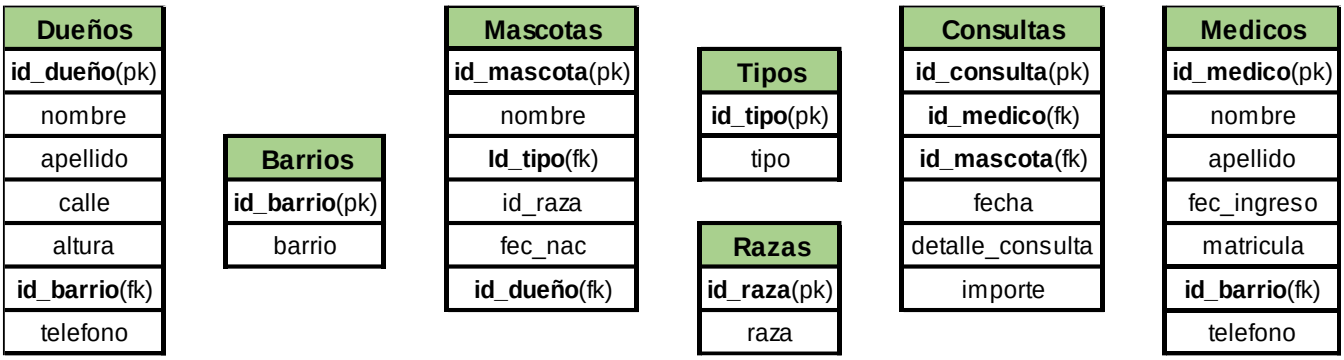
```
Create view v_punto2
as
  select month(fecha_pago) mes, year(fecha_pago) año, id_paquete paquete
     sum(dbo.f_m_pagado(monto_cuota,fecha_vto,fecha_pago)) Total_pagado,
     count(*) Cantidad_pagos,
     avg(dbo.f_m_pagado(monto_cuota,fecha_vto,fecha_pago)) Promedio_pagado
  from pagos
  where year(fecha_pago)>year(getdate())-5
  group by month(fecha_pago), year(fecha_pago), id_paquete
```

3. Cree un procedimiento almacenado que consulte la vista anterior y muestre los totales pagados y la cantidad de pagos mensualmente por paquete dentro del año que se ingresará por parámetro. Verifique que el año ingresado por parámetro esté dentro de los años considerados en la vista, en caso contrario dé un mensaje informando la situación. Escriba la sentencia que ejecuta el procedimiento almacenado.

```
Create proc pa_punto3
@anio int
As
  if @anio>year(getdate())-5
    select mes, año, paquete, total_pagado, cantidad_pagos
    from v_punto2
    where año=@anio
  else
    select 'El año ingresado no es correcto' Error

Execute pa_punto3 2020
```

Bases de datos PRODUCCIÓN: Escribir la sentencia SQL Server para obtener lo que se solicita (20% cada una):



1. Crear una función que devuelva como parámetros de salida la edad y el próximo cumpleaños de la mascota recibiendo como parámetro de entrada la fecha de nacimiento.
2. Crear un procedimiento almacenado que reciba el dueño como parámetro entrada y cree una vista para mostrar el dueño y el listado de todas las mascotas que forman parte de su familia. Ejecute el procedimiento almacenado. Muestre el contenido de la vista.
3. Crear un procedimiento almacenado que reciba el médico como parámetro de entrada y cree una tabla temporal local que contenga todas las mascotas atendidas por ese médico.
4. Crear un disparador que evite crear una mascota sin dueño, cree un error en caso de suceder tal situación y avise al usuario que intento realizarlo.

TEÓRICO: explique, diferencie y ejemplifique a los Disparadores Instead Off y After.

Punto 1

```
create function edad_prox (
@fec_ date
)
returns varchar(50)
begin
   declare @retorno varchar(50)
   declare @edad int
   declare @prox date
   set @edad =  datediff(year, @fec_, getdate())
   set @prox = DATEADD(year, @edad + 1, @fec_)
   set @retorno = concat( cast(@edad as varchar(10)) , ' ', cast(@prox as varchar(10)))
   return @retorno
end

select dbo.edad_prox(fec_nac) 'edad' from mascotas
```

Punto 2

```
create view vi_dos as SELECT d.apellido+' '+d.nombre 'dueño', m.nombre 'mascota' , d.id_dueño 'id'
FROM dueños d, mascotas m
WHERE d.id_dueño = m.id_dueño

create proc punto_dos @id_due int
as
select * from vi_dos where id=@id_due

exec punto_dos 1
```

Punto 3

```
create proc punto_tres @id_med int
as
create table #atendidos (dueño varchar(100), mascota varchar(100))
insert into #atendidos
SELECT med.apellido+' '+med.nombre 'dueño', m.nombre 'mascota'
FROM medicos med, mascotas m , consultas c
WHERE med.id_medico = @id_med and c.id_medico = @id_med and c.id_mascota = m.id_mascota

select * from #atendidos
```

Punto 4

```
CREATE TRIGGER masc_due_prue
 on mascotas
   AFTER INSERT
 AS BEGIN
   IF (NOT EXISTS(
      SELECT i.id_dueño
      FROM dueños d, inserted i, mascotas m
      WHERE d.id_dueño = i.id_dueño and m.id_dueño = i.id_dueño ))
    raiserror(15600,-1,-1,'no hay dueño')
 END
```

Punto 5 **Teórico: Disparador Instead Of y After**

Hasta el momento se ha mostrado triggers que se crean sobre una tabla específica para un evento de inserción, eliminación o actualización; pero, también se puede especificar el momento en que se ejecutan. Este momento de disparo indica que las acciones (sentencias) del trigger se ejecuten después (after) de la acción (insert, delete o update) que dispara el trigger o en lugar (instead of) de la acción.

La sintaxis básica es:

```
create trigger NOMBREDISPARADOR
 on NOMBRETABLA o VISTA
 MOMENTODEDISPARO-- after o instead of
 ACCION-- insert, update o delete
as
 SENTENCIAS
```

Si no se especifica el momento de disparo en la creación del trigger, por defecto se establece como "after", es decir, las acciones que el disparador realiza se ejecutan luego del suceso disparador.

Los disparadores "instead of" se ejecutan en lugar de la acción desencadenante, es decir, cancelan la acción desencadenante (suceso que disparó el trigger) reemplazándola por otras acciones.

Por ejemplo. Se crea un trigger que inserte un registro en la tabla detalles_facturas si la cantidad vendida es menor al stock del artículo sino da un mensaje informando la situación:

```
create trigger dis_clie_actualizar
  on clientes
  instead of update
  as
   if update(cod_cliente)
   begin
     raiserror('Los códigos no pueden modificarse', 10, 1)
     rollback transaction
   end
   else
   begin
     update clientes
        set clientes.nom_cliente=inserted.nom_cliente,
            clientes.ape_cliente=inserted.ape_cliente
       from clientes join inserted
         on clientes.cod_cliente =inserted.cod_cliente
   end
```

Consideraciones:

Se pueden crear disparadores "instead of" en vistas y tablas.

No se puede crear un disparador "instead of" en vistas definidas "with check option".

No se puede crear un disparador "instead of delete" y "instead of update" sobre tablas que tengan una "foreign key" que especifique una acción "on delete cascade" y "on update cascade" respectivamente.

Los disparadores "after" no pueden definirse sobre vistas.

No pueden crearse disparadores "after" en vistas ni en tablas temporales; pero pueden referenciar vistas y tablas temporales.

Si existen restricciones en la tabla del disparador, se comprueban DESPUES de la ejecución del disparador "instead of" y ANTES del disparador "after". Si se infringen las restricciones, se revierten las acciones del disparador "instead of"; en el caso del disparador "after", no se ejecuta.