use AdventureWorks

--Clase 06--

--Funciones--

--Una función puede recibir o no parámetros

--Funciones escalares

--Creo una función escalar

drop function if exists TotalVentasIVAIncluido

go

create function TotalVentasIVAIncluido()

returns money

as

begin

declare @Total money

select @Total = sum(LineTotal) from Sales.SalesOrderDetail

set @Total = @Total \* 1.21

return @Total

end

go

select dbo.TotalVentasIVAIncluido()

select Name, ListPrice, dbo.TotalVentasIVAIncluido() from Production.Product

--Altero la función

go

alter function TotalVentasIVAIncluido(@pProducto int)

returns money

as

begin

declare @Total money

select @Total = sum(LineTotal) from Sales.SalesOrderDetail

where ProductID = @pProducto

set @Total = @Total \* 1.21

return @Total

end

go

select Name, ListPrice, dbo.TotalVentasIVAIncluido(ProductID) from Production.Product

select Name, ListPrice, dbo.TotalVentasIVAIncluido(ProductID),

dbo.TotalVentasIVAIncluido(ProductID) \* 1.05 [Retención extra]

from Production.Product

select dbo.TotalVentasIVAIncluido(775)

print dbo.TotalVentasIVAIncluido(777)

--Funciones tabla

drop function if exists ListarVentasCliente

go

create function ListarVentasCliente(@pCliente int)

returns table

as

return

(

select SalesOrderID Orden, CustomerID Cliente, OrderDate Fecha, TotalDue Total

from Sales.SalesOrderHeader where CustomerID = @pCliente

)

go

select \* from ListarVentasCliente(29825)

select count(\*) [Cantidad operaciones], avg(Total) [Monto Promedio]

from ListarVentasCliente(29825)

go

--Hago lo mismo que la función con un stored procedure, parece que

-- hace lo mismo, pero a la hora de hacer un count por ej, no resulta

drop procedure if exists Sales.ListarVentasCliente

go

create procedure Sales.ListarVentasCliente @pCliente int

as

select SalesOrderID Orden, CustomerID Cliente, OrderDate Fecha, TotalDue Total

from Sales.SalesOrderHeader where CustomerID = @pCliente

go

select \* from ListarVentasCliente(29825)

exec Sales.ListarVentasCliente 29825

--Se puede (Función)

select count(\*) [Cantidad operaciones], avg(Total) [Monto Promedio]

from ListarVentasCliente(29825)

--No se puede (Stored Procedure)

select count(\*) [Cantidad operaciones], avg(Total) [Monto Promedio]

from Sales.ListarVentasCliente 29825

--Una función y un stored procedure no pueden llamarse igual porque ambos

--son objetos, a no ser que pertenezcan a esquemas distintos (ej dbo.xx, Sales.xx)

--

--Tambien puedo hacer esto desde la función, no así de sp

declare @venta\_del\_cliente money

select @venta\_del\_cliente = sum(Total) from ListarVentasCliente(29825)

select @venta\_del\_cliente

--

select \* from Personas

drop function if exists ListarPersonas

go

create function ListarPersonas(@criterio varchar(10))

returns @Listado table(Legajo int, Nombre varchar(50), Apellido varchar(50))

as

begin

if @criterio = 'nombre'

begin

insert into @Listado select legajo, nombre, apellido

from Personas order by Nombre

end

if @criterio = 'apellido'

begin

insert into @Listado select legajo, nombre, apellido

from Personas order by Apellido

end

return

end

go

select \* from ListarPersonas('apellido')

--La modifico

go

alter function ListarPersonas(@criterio varchar(10))

returns @Listado table(Legajo int, Nombre varchar(100))

as

begin

if @criterio = 'nombre'

insert into @Listado select legajo, nombre +' '+ apellido

from Personas order by nombre +' '+ apellido

if @criterio = 'apellido'

insert into @Listado select legajo, apellido +' '+nombre

from Personas order by apellido +' '+nombre

return

end

go

select \* from ListarPersonas('apellido')

select \* from ListarPersonas('nombre')

--

--Laboratorio: AdventureWorks--

--Funciones Escalares--

--1. Crear una función que devuelva el promedio de los productos.

--Tablas: Production.Product

--Campos: ListPrice

drop function if exists PrecioPromedioProducto

go

create function PrecioPromedioProducto()

returns decimal (10, 2)

as

begin

declare @promedio decimal (10,2)

select @promedio = avg(ListPrice) from Production.Product

return @promedio

end

go

select dbo.PrecioPromedioProducto()

--

drop function if exists PrecioPromedioProducto1

go

create function PrecioPromedioProducto1()

returns money

as

begin

declare @promedio money

select @promedio = avg(ListPrice) from Production.Product

return @promedio

end

go

select dbo.PrecioPromedioProducto1()

--2. Crear una función que dado un código de producto devuelva el total de ventas para

--dicho producto luego, mediante una consulta, traer código y total de ventas.

--Tablas: Sales.SalesOrderDetail

--Campos: ProductID, LineTotal

drop function if exists TotalVentasProductos

go

create function TotalVentasProductos(@pProducto int)

returns money

as

begin

declare @Total\_Venta money

select @Total\_Venta = sum(LineTotal) from Sales.SalesOrderDetail

where @pProducto = ProductID

return @Total\_Venta

end

go

select dbo.TotalVentasProductos(777)

select ProductID, LineTotal from Sales.SalesOrderDetail

--3. Crear una función que dado un código devuelva la cantidad

--de productos vendidos o cero si no se ha vendido.

--Tablas: Sales.SalesOrderDetail

--Campos: ProductID, OrderQty

drop function if exists CantidadProductosVendidos

go

create function CantidadProductosVendidos(@pProducto int)

returns int

as

begin

declare @Cantidad int

select @Cantidad = sum(OrderQty) from Sales.SalesOrderDetail where @pProducto = ProductID

if @Cantidad is null

set @Cantidad = 0

return @Cantidad

end

go

select dbo.CantidadProductosVendidos(9999)

--4. Crear una función que devuelva el promedio total de venta, luego

--obtener los productos cuyo precio sea inferior al promedio.

--Tablas: Sales.SalesOrderDetail, Production.Product

--Campos: ProductID, ListPrice

drop function if exists PromedioVenta

go

create function PromedioVenta()

returns money

as

begin

declare @Promedio money

select @Promedio = avg(UnitPrice) from Sales.SalesOrderDetail

if @Promedio is null

set @Promedio = 0

return @Promedio

end

go

select dbo.PromedioVenta()

--Menos perfomante, tarda muchos segundos

select Name, ListPrice from Production.Product

where ListPrice < dbo.PromedioVenta()

order by ListPrice desc

go

--Mas performante, tarda 1 seg

declare @Valor money

select @Valor = dbo.PromedioVenta()

select Name, ListPrice from Production.Product

where ListPrice < @Valor

order by ListPrice desc

go

--5. Crear una función que dado un año, devuelva nombre y

--apellido de los empleados que ingresaron ese año.

--Tablas: Person.Person, HumanResources.Employee

--Campos: FirstName, LastName,HireDate, BusinessEntityID

drop function if exists NombreCompleto

go

create function NombreCompleto(@pYear int)

returns table

as

return

(

select FirstName, LastName, HireDate, p.BusinessEntityID from Person.Person p

join HumanResources.Employee e on p.BusinessEntityID = e.BusinessEntityID

where year(HireDate) = @pYear

)

go

select \* from dbo.NombreCompleto(2003)

where month(HireDate) > 2

select \* from Person.Person

select \* from HumanResources.Employee

--6. Crear una función que reciba un parámetro correspondiente a un precio y

--nos retorna una tabla con código, nombre, color y precio de todos los

--productos cuyo precio sea inferior al parámetro ingresado.

--Tablas: Production.Product

--Campos: ProductID, Name, Color, ListPrice

drop function if exists PreciosInferiores

go

create function PreciosInferiores(@Precio money)

returns table

as

return

(

select ProductID, Name, Color, ListPrice from Production.Product

where ListPrice < @Precio

)

go

select \* from dbo.PreciosInferiores(300) order by ListPrice desc

--Laboratorio: AdventureWorks--

--Funciones de Tabla en Línea--

--7. Realizar el mismo pedido que en el punto anterior, pero

--utilizando este tipo de función.

--Tablas: Production.Product

--Campos: ProductID, Name, Color, ListPrice

drop function if exists PreciosInferioresMulti

go

create function PreciosInferioresMulti(@Precio money)

returns @Lista table(Nombre varchar(50), Color varchar(25), Precio money)

as

begin

if @Precio <= 300

begin

insert into @Lista select Name, Color, ListPrice from Production.Product

where ListPrice < @Precio order by ListPrice desc

end

if @Precio > 300

begin

insert into @Lista select Name, Color, ListPrice from Production.Product

where ListPrice > @Precio order by ListPrice

end

return

end

go

select \* from dbo.PreciosInferioresMulti(250) order by Precio desc

select \* from dbo.PreciosInferioresMulti(350) order by Precio

--

--Triggers/Desencadenadores--

--Se van a ejecutar en respuesta a un evento

--(eventos a nivel de tabla (que podrucen modificaciones en las mismas:

--inserciones, borrado de registros, y actualizaciones de datos),

--(y a nivel de base de datos: creación, modificaciones a partir de alter,

--y eliminación de objetos a partir de drop)

--(El evento se produce siempre que yo hago una inserción de un registro en una tabla)

--Los triggers son objetos propios de una tabla, cuando los creo los tengo que asociar

--a una tabla, y a un evento de la misma.

--Hay dos tipos de triggers para los eventos de tabla (instead of y after)

--after insert

drop table if exists alumnos

go

create table alumnos (codigo int, nombre varchar(100))

go

drop table if exists alumnos\_log

go

create table alumnos\_log (codigo int, nombre varchar(100), fecha\_actualizacion datetime)

go

select \* from alumnos

select \* from alumnos\_log

drop trigger if exists AltaAlumno

go

create trigger AltaAlumno on dbo.alumnos

after insert

as

begin

insert into alumnos\_log (codigo, nombre, fecha\_actualizacion)

select codigo, nombre, getdate() from inserted

end

go

insert into alumnos values

(1, 'Jon Snow'),

(2, 'Daenerys Targaryen'),

(3, 'Arya Stark'),

(4, 'Tyrion Lannister')

--Cursores

--select \* from deleted

--select \* from inserted

--after update (posibilidad de crear tablas históricas)

drop trigger if exists ActualizaAlumno

go

create trigger ActualizaAlumno on dbo.alumnos

after update

as

begin

--update al set nombre = i.nombre, fecha\_actualizacion = getdate() from alumnos\_log al

--from alumnos\_log al join inserted i on al.codigo = i.codigo --no

insert into alumnos\_log (codigo, nombre, fecha\_actualizacion)

select codigo, nombre, getdate() from deleted

end

go

select \* from alumnos

select \* from alumnos\_log

update alumnos set nombre = 'Jon Stark' where codigo = 1

update alumnos set nombre = 'Jon Lannister' where codigo = 1

--after delete

drop trigger if exists EliminaAlumno

go

create trigger EliminaAlumno on dbo.alumnos

after delete

as

begin

insert into alumnos\_log (codigo, nombre, fecha\_actualizacion)

select codigo, nombre, getdate() from deleted

end

go

select \* from alumnos

select \* from alumnos\_log

delete from alumnos where codigo = 1

--

alter table alumnos\_log add Operacion char(1)

--En el trigger insert (Operacion = 'A')

--En el trigger update (Operacion = 'M')

--En el trigger delete (Operacion = 'B')

go

alter trigger EliminaAlumno on dbo.alumnos

after delete

as

begin

insert into alumnos\_log (codigo, nombre, fecha\_actualizacion, operacion)

select codigo, nombre, getdate(), 'B' from deleted

end

go

delete from alumnos where codigo = 3

select \* from alumnos

select \* from alumnos\_log

--

--Laboratorio: AdventureWorks--

--Triggers--

--1. Clonar estructura (ProductID, ListPrice) y datos de la

--tabla Production.Product en una tabla llamada Productos.

drop table if exists Productos

go

select ProductID Producto, ListPrice Precio into Productos from Production.Product

go

select \* from Productos

go

--2. Crear un trigger sobre la tabla Productos llamado TR\_ActualizaPrecios

--dónde actualice la tabla #HistoricoPrecios con los cambios de precio.

--Tablas: Productos

--Campos: ProductID, ListPrice

drop table if exists HistoricoPrecios

go

--despues de crearla tuve que entrar a la tabla:

--(scrip table as -> create to -> new query editor window)

--copié el create table de abajo y le quité el identity

CREATE TABLE [dbo].[HistoricoPrecios](

[Producto] [int],

[Precio] [money] NOT NULL,

[Ultima\_Modificacion] [datetime] NOT NULL

) ON [PRIMARY]

GO

insert into HistoricoPrecios

select Producto, Precio, getdate()

from Productos

go

select \* from HistoricoPrecios

go

drop trigger if exists TR\_ActualizaPrecios

go

create trigger TR\_ActualizaPrecios on Productos

after update

as

begin

insert into HistoricoPrecios (Producto, Precio, Ultima\_Modificacion)

select Producto, Precio, getdate() from deleted

end

go

update Productos set Precio = 0 where Producto <= 4

select \* from Productos

select \* from HistoricoPrecios

go

--3. Adaptar el trigger del punto anterior donde valide que

--el precio no pueda ser negativo.

alter trigger TR\_ActualizaPrecios on Productos

after update

as

begin

declare @precio\_nuevo money

select @precio\_nuevo = Precio from inserted

if @precio\_nuevo >= 0

insert into HistoricoPrecios (Producto, Precio, Ultima\_Modificacion)

select Producto, Precio, getdate() from deleted

else

rollback transaction

end

go

update Productos set Precio = -50 where Producto <= 4

select \* from Productos

select \* from HistoricoPrecios

---

--Funciones integradas

--Fecha

--(Convertir determinados tipos de datos de valores fecha, a valores de español)

--(utilizando los distintos tipos de conversión)

--date

--Fecha actual

declare @fecha\_actual date = getdate()

select @fecha\_actual [Formato estandar], convert(varchar, @fecha\_actual, 101) [Formato 101],

convert(varchar, @fecha\_actual, 102) [Formato 102], convert(varchar, @fecha\_actual, 103) [Formato 103],

convert(varchar, @fecha\_actual, 105) [Formato 105], convert(varchar, @fecha\_actual, 106) [Formato 106],

convert(varchar, @fecha\_actual, 107) [Formato 107], convert(varchar, @fecha\_actual, 111) [Formato 111],

convert(varchar, @fecha\_actual, 112) [Formato 112]

go

--datetime

--Hora

declare @fecha\_actual datetime = getdate()

select @fecha\_actual [Formato estandar], convert(varchar, @fecha\_actual, 8) [Formato 8],

convert(varchar, @fecha\_actual, 14) [Formato 14], convert(varchar, @fecha\_actual, 24) [Formato 24],

convert(varchar, @fecha\_actual, 108) [Formato 108], convert(varchar, @fecha\_actual, 114) [Formato 114]

go

--Fecha y Hora, aclaración AM-PM

declare @fecha\_actual datetime = getdate()

select @fecha\_actual [Formato estandar], convert(varchar, @fecha\_actual, 0) [Formato 0],

convert(varchar, @fecha\_actual, 9) [Formato 9]

go

--date --datename

--sumamos los campos que necesitamos

declare @fecha\_actual date = getdate()

select @fecha\_actual [Formato estandar], datename(year, @fecha\_actual) [Año],

datename(month, @fecha\_actual) [Mes], datename(day, @fecha\_actual) [Día del mes],

datename(dayofyear, @fecha\_actual) [Día del año],

datename(weekday, @fecha\_actual) [Día de la semana]

go

--date --datepart

--sumamos los campos que necesitamos

declare @fecha\_actual date = getdate()

select @fecha\_actual [Formato estandar], datepart(year, @fecha\_actual) [Año],

datepart(month, @fecha\_actual) [Mes], datepart(day, @fecha\_actual) [Día del mes],

datepart(dayofyear, @fecha\_actual) [Día del año],

datepart(weekday, @fecha\_actual) [Día de la semana]

go

--date --

--formato más entandar

declare @fecha\_actual date = getdate()

select @fecha\_actual [Formato estandar], year(@fecha\_actual) [Año],

month(@fecha\_actual) [Mes], day(@fecha\_actual) [Día del mes]

go

--datediff (diferencia entre rangos de fechas)

declare @fecha\_actual date = getdate(), @fechaInicio date = '1980-12-31'

select datediff(year, @fechaInicio, @fecha\_actual) años,

datediff(month, @fechaInicio, @fecha\_actual) meses,

datediff(day, @fechaInicio, @fecha\_actual) días

go

--dateadd (me permite sumar valores a las fechas,

--por ende puedo operar aritméticamente con ellas)

declare @fecha\_actual date = getdate()

select @fecha\_actual [Fecha actual], dateadd(year, 1, @fecha\_actual) Sumo1Año,

dateadd(month, 12, @fecha\_actual) Sumo12Meses,

dateadd(day, 365, @fecha\_actual) Sumo365Días

go

--

--Casteo

select Name, ListPrice, ListPrice \* 1.21 [Precio con IVA],

ROUND(ListPrice \* 1.21, 2), ROUND(ListPrice \* 1.21, 2, 0),

CAST(ListPrice \* 1.21 as decimal(7, 2))

from Production.Product

where ListPrice > 0 order by ListPrice desc

declare @precio decimal(7,2) = 99999.99 --

--(decimal ('max numeros posibles incluyendo decimales 7-2=5', 'decimales')

select @precio