use AdventureWorks

--Clase 05--

--Variables--

--El go no es necesario, pero mostramos como inicia y termina la sentencia

--Lote/Batch (lo que integra go)

--Debo ejecutar la siguiente sentencia (lote/batch) seleccionando todo

go

declare @alumno varchar(50)

set @alumno = 'Ariel MF'

select @alumno Alumno

--print @alumno --Otra forma de mostrar el msj (tabular)

go

--Puedo declarar variables y asignarles valores en la misma línea

declare @alumno varchar(50)='Ariel MF', @edad int = 45, @empresa varchar(50)='Educación IT'

select @alumno, @edad, @empresa

--Forma de asignar valores dinámicos

declare @promedio\_venta money

select @promedio\_venta = avg(TotalDue) from Sales.SalesOrderHeader

select @promedio\_venta

--Forma que solíamos utilizar

select top 5 SalesOrderID, TotalDue from Sales.SalesOrderHeader

where TotalDue > (select avg(TotalDue) from Sales.SalesOrderHeader) order by TotalDue

go

--Forma que ahora podemos utilizar

declare @promedio\_venta money

select @promedio\_venta = avg(TotalDue) from Sales.SalesOrderHeader

select top 5 SalesOrderID, TotalDue from Sales.SalesOrderHeader

where TotalDue > @promedio\_venta

order by TotalDue

go

--Variable tipo table (debo declararle también su estructura)

declare @ventas\_mayores table(id\_factura int, total money)

insert into @ventas\_mayores select SalesOrderID, TotalDue

from Sales.SalesOrderHeader where TotalDue > 5000

select \* from @ventas\_mayores order by total

select avg(total) Promedio from @ventas\_mayores

select count(\*) Cantidad from @ventas\_mayores

go

--Variables globales

select @@DEFAULT\_LANGID, @@SERVERNAME, @@SERVICENAME, @@CPU\_BUSY,

SUSER\_ID(), SUSER\_NAME()

--Control--

--IF

declare @promedio\_venta money

select @promedio\_venta = avg(TotalDue) from Sales.SalesOrderHeader

if @promedio\_venta < 3000

print 'Promedio de venta menor al objetivo'

if @promedio\_venta >= 3000

print 'Promedio de venta mayor al objetivo'

go

--IF - ELSE

declare @promedio\_venta money

select @promedio\_venta = avg(TotalDue) from Sales.SalesOrderHeader

if @promedio\_venta < 3000

print 'Promedio de venta menor al objetivo'

else

print 'Promedio de venta mayor al objetivo'

go

--Ciclos--

--While

declare @puntos int = 0

while(@puntos < 35)

begin

print @puntos

set @puntos = @puntos + 3

end

print 'Valor actual de la variable @puntos'

print @puntos

go

--Laboratorio: AdventureWorks--

--Manejo de variables--

--1. Obtener el total de ventas del año 2014 y guardarlo en una

--variable llamada @TotalVentas, luego imprimir el resultado.

--Tablas: Sales.SalesOrderDetail

--Campos: LineTotal

declare @TotalVentas money

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

where year(ModifiedDate) = 2005

select @TotalVentas [Total ventas del año]--

go

select sum(LineTotal) from Sales.SalesOrderDetail

where year(ModifiedDate) = 2005

--2. Obtener el promedio de precios y guardarlo en una variable

--llamada @Promedio luego hacer un reporte de todos los productos

--cuyo precio de venta sea menor al Promedio.

--Tablas: Production.Product

--Campos: ListPrice, ProductID

declare @Promedio money

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

select @Promedio

select ProductID, ListPrice from Production.Product

where ListPrice < @Promedio and ListPrice > 0 order by ListPrice desc--

go

select avg(ListPrice) from Production.Product

--3. Utilizando la variable @Promedio incrementar en un 10%

--el valor de los productos sean inferior al promedio.

--Tablas: Production.Product

--Campos: ListPrice

declare @Promedio money

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

update Production.Product set ListPrice = ListPrice \* 1.10

where ListPrice < @Promedio--

go

select ListPrice from Production.Product where ListPrice > 0

--4. Crear un variable de tipo tabla con las categorías y

--subcategoría de productos y reportar el resultado.

--Tablas: Production.ProductSubcategory, Production.ProductCategory

--Campos: Name

declare @Categorias table (Categoria varchar(100), Subcategoria varchar(100))

insert into @Categorias --(Categoria, Subcategoria)

select pc.Name, ps.Name from Production.ProductCategory pc

join Production.ProductSubcategory ps

on ps.ProductCategoryID = pc.ProductCategoryID

select \* from @Categorias order by Categoria, Subcategoria

go

--5. Analizar el promedio de la lista de precios de productos, si

--su valor es menor 500 imprimir el mensaje el PROMEDIO

--BAJO de lo contrario imprimir el mensaje PROMEDIO ALTO.

--Tablas: Production.Product

--Campos: ListPrice

declare @Promedio money

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

print @Promedio

if @Promedio < 500

print 'PROMEDIO BAJO'

else

print 'PROMEDIO ALTO'

go

-------------------------------------------

---Procedimientos Almacenados / Stored Procedure---

-------------------------------------------

--Todas equivalentes (pide info de las db)

execute sp\_helpdb

exec sp\_helpdb

sp\_helpdb

go

--Creo un Stored Procedure

create procedure ConsultarPersonas

as

select BusinessEntityID Codigo, FirstName Nombre, LastName Apellido

from Person.Person

go

--Llamo al SP

ConsultarPersonas

go

--Modifico el Stored Procedure ya creado

alter procedure ConsultarPersonas @pApellido varchar(50), @pNombre varchar(50)

as

select BusinessEntityID Codigo, FirstName Nombre, LastName Apellido

from Person.Person

where @pNombre = FirstName and LastName = @pApellido

go

--Llamo al SP aclarando los parámetros (en el mismo orden que fueron creados)

ConsultarPersonas 'Duffy', 'Terri'

go

--Parámetros con nombre / nombrados (aclaro variables y no importa orden)

ConsultarPersonas @pNombre = 'Terri', @pApellido = 'Duffy'

go

--Parámetros predeterminados / Default

alter procedure ConsultarPersonas @pApellido varchar(50) = 'Duffy', @pNombre varchar(50) = 'Terri'

as

select BusinessEntityID Codigo, FirstName Nombre, LastName Apellido

from Person.Person

where @pNombre = FirstName and LastName = @pApellido

go

--Llamo al SP y ya no necesito asignarle parámetros

--porque tiene valores por default

ConsultarPersonas

go

--Aún así puedo seguir usándolo con parámetros

ConsultarPersonas 'Achong', 'Gustavo'

go

--Modificamos y agremos IF y ELSE

alter procedure ConsultarPersonas @pApellido varchar(50), @pNombre varchar(50)

as

if(lower(@pApellido) = 'no' and lower(@pNombre) = 'no')

select BusinessEntityID Codigo, FirstName Nombre, LastName Apellido

from Person.Person order by FirstName

else

select BusinessEntityID Codigo, FirstName Nombre, LastName Apellido

from Person.Person

where @pNombre = FirstName and LastName = @pApellido

order by FirstName

go

--Consultamos

ConsultarPersonas 'no', 'no'

go

ConsultarPersonas 'Achong', 'Gustavo'

go

--Parámetros de salida (output)

drop procedure if exists TraerSectorEmpleado

go

create procedure TraerSectorEmpleado @pApellido varchar(50), @pNombre varchar(50),

@pSector varchar(100) output

as

select @pSector = Department from HumanResources.vEmployeeDepartmentHistory

where LastName = @pApellido and FirstName = @pNombre

go

--Lo invoco (tengo que ejecutarlo con execute/exec)

declare @pSector varchar(100)

exec TraerSectorEmpleado 'Chen', 'John', @pSector output

select @pSector Sector

select \* from HumanResources.vEmployeeDepartmentHistory where Department = @pSector

go

--Laboratorio: AdventureWorks—

--Procedimientos almacenados--

--1. Crear un procedimiento almacenado en el esquema HumanResources que dada una

--determinada inicial, devuelva codigo, nombre, apellido y dirección de correo de

--los empleados cuyo nombre coincida con la inicial ingresada.

--Vistas: HumanResources.vEmployee

--Campos: BusinessEntityID, FirstName, LastName, EmailAddress

drop procedure if exists HumanResources.TraerDatosEmpleados

go

create procedure HumanResources.TraerDatosEmpleados @Inicial char(1)

as

select BusinessEntityID, FirstName, LastName, EmailAddress from HumanResources.vEmployee

where left(FirstName, 1) = @Inicial

go

HumanResources.TraerDatosEmpleados 'j'

go

--2. Crear un procedimiento almacenado llamado ProductoVendido que permita ingresar

--un producto como parámetro, si el producto ha sido vendido imprimir el mensaje

--“El PRODUCTO HA SIDO VENDIDO” de lo contrario imprimir “El PRODUCTO NO HA SIDO VENDIDO”

--Tablas: Sales.SalesOrderDetail

--Campos: ProductID

drop procedure if exists ProductoVendido

go

create procedure ProductoVendido @Producto int

as

if exists(select \* from Sales.SalesOrderDetail where ProductID = @Producto)

print 'El PRODUCTO HA SIDO VENDIDO'

else

print 'El PRODUCTO NO HA SIDO VENDIDO'

go

ProductoVendido 771

go

--3. Crear un procedimiento almacenado en el esquema dbo para la actualización de

--precios llamado ActualizaPrecio recibiendo como parámetros el producto y el precio.

--Tablas: Production.Product

--Campos: ProductID, Name, ListPrice

drop procedure if exists dbo.ActualizaPrecio

go

create procedure dbo.ActualizaPrecio @Producto int, @Precio money

as

update Production.Product set ListPrice = @Precio

where ProductID = @Producto and ListPrice > 0

go

select ProductID, Name, ListPrice from Production.Product where ListPrice > 0

go

dbo.ActualizaPrecio 1, 100

go

--4. Crear un procedimiento almacenado llamado ProveedorProducto que devuelva

--los proveedores, el nombre del producto y el número de cuenta, y el código de unidad

--de medida que proporciona el producto especificado por parámetro.

--Tablas: Purchasing.Vendor, Purchasing.ProductVendor, Production.Product

--Campos: Name, AccountNumber, UnitMeasureCode

drop procedure if exists ProveedorProducto

go

create procedure ProveedorProducto @Producto int

--alter procedure ProveedorProducto @Producto int --ej modificación

as

select v.Name from Production.Product p join Purchasing.ProductVendor pv

on p.ProductID = pv.ProductID join Purchasing.Vendor v

on pv.BusinessEntityID = v.BusinessEntityID

and p.ProductID = @Producto

go

ProveedorProducto 1

go

--5. Crear un procedimiento almacenado llamado EmpleadoSector que devuelva

--nombre, apellido y sector del empleado que le pasemos como argumento. No es

--necesario pasar el nombre y apellido exactos al procedimiento.

--Vistas: HumanResources.vEmployeeDepartmentHistory

--Campos: FirstName, LastName, Department

drop procedure if exists EmpleadoSector

go

create procedure EmpleadoSector @Apellido varchar(50) = '%'

as

select FirstName, LastName, Department from HumanResources.vEmployeeDepartmentHistory

where LastName like @Apellido

go

select FirstName, LastName, Department from HumanResources.vEmployeeDepartmentHistory

go

EmpleadoSector 'Cha%'

go

EmpleadoSector 'Chai'

go

EmpleadoSector '%ha'

go

--Procedimientos almacenados transaccionales simples--

drop table if exists Personas

select BusinessEntityID Legajo, FirstName Nombre, LastName Apellido

into Personas from Person.Person

where BusinessEntityID <= 500

select \* from Personas

--Alta

drop procedure if exists AltaPersona

go

create procedure AltaPersona @pNombre varchar(50), @pApellido varchar(50)

as

insert Personas (Nombre, Apellido) values (@pNombre, @pApellido)

go

AltaPersona 'TheFirstName', 'TheLastName'

go

--Baja

drop procedure if exists BajaPersona

go

create procedure BajaPersona @pLegajo int

as

delete Personas where Legajo = @pLegajo

go

BajaPersona 1

go

select \* from Personas

go

--Actualizar

drop procedure if exists ModificarPersona

go

create procedure ModificarPersona @pLegajo int, @pNombre varchar(50), @pApellido varchar(50)

as

update Personas set Nombre = @pNombre, Apellido = @pApellido where Legajo = @pLegajo

go

ModificarPersona 2, 'Elsa', 'Payo'

go

select \* from Personas