use [AdventureWorks]

--Clase 04

--Subconsultas--

--En el SELECT

select top 3 ProductID Codigo, Name Nombre, ListPrice Precio,

(select AVG(ListPrice) from Production.Product) [Precio Promedio],

(select AVG(UnitPrice) from Sales.SalesOrderDetail) [Precio Promedio de Venta],

(select AVG(ListPrice) from Production.Product) - ListPrice [Diferencia de Precio]

from Production.Product where ListPrice > 0

--En el FROM

select top 3 p.ProductID Codigo, Name Nombre, ListPrice Precio, Precio\_Promedio

from Production.Product p join

(select ProductID, AVG(UnitPrice) Precio\_Promedio

from Sales.SalesOrderDetail group by ProductID) Ventas

on p.ProductID = Ventas.ProductID

--En el WHERE

select ProductID Codigo, Name Nombre, ListPrice Precio,

(select AVG(ListPrice) from Production.Product) [Precio Promedio],

(select AVG(UnitPrice) from Sales.SalesOrderDetail) [Precio Promedio de Venta],

(select AVG(ListPrice) from Production.Product) - ListPrice [Diferencia de Precio]

from Production.Product where ListPrice > 0 and

ListPrice > (select AVG(ListPrice) from Production.Product)

order by ListPrice desc

--Subconsultas correlacionadas--

--La subconsulta utiliza un campo que pernetece a la consulta original

select ProductSubcategoryID Subcategoria, Name Nombre, ListPrice Precio

from Production.Product p where ListPrice =

(select min(ListPrice) from Production.Product p1

where p.ProductSubcategoryID = p1.ProductSubcategoryID)

--IN

select ProductId Codigo, Name Nombre, ListPrice Precio from Production.Product

where ProductId in (995,996,997)

select ProductId Codigo, Name Nombre, ListPrice Precio from Production.Product

where ProductId between 500 and 999

-- probamos equivalentes --

select ProductId Codigo, Name Nombre, ListPrice Precio from Production.Product

where ProductId in (select ProductID from Sales.SalesOrderDetail) --

--equivalentes en cuanto a la información, pero no a la cantidad de resultados

--al hacer un join devuelve todos los registros de ambas tablas, si quisiera

--que sean iguales, deberia aplicar distinct en p.ProductID en la segunda

--o agrupo por p.ProductId (group by p.ProductID, Name, ListPrice)

select p.ProductId Codigo, Name Nombre, ListPrice Precio from Production.Product p

join Sales.SalesOrderDetail v on p.ProductID = v.ProductID

select distinct p.ProductId Codigo, Name Nombre, ListPrice Precio from Production.Product p

join Sales.SalesOrderDetail v on p.ProductID = v.ProductID

--equivalentes

select p.ProductId Codigo, Name Nombre, ListPrice Precio from Production.Product p

join Sales.SalesOrderDetail v on p.ProductID = v.ProductID

group by p.ProductID, Name, ListPrice

--equivalentes

select ProductID from Production.Product

intersect

select ProductId from Sales.SalesOrderDetail

---

--NOT IN

select ProductId Codigo, Name Nombre, ListPrice Precio from Production.Product

where ProductId not in (select ProductID from Sales.SalesOrderDetail)

--equivalente

select ProductID from Production.Product

except

select ProductId from Sales.SalesOrderDetail

--EXISTS (sería lo mismo que el IN, INTERSECT)

select ProductId Codigo, Name Nombre, ListPrice Precio from Production.Product p

where exists

(select ProductID from Sales.SalesOrderDetail v where p.ProductID = v.ProductID)

--NOT EXISTS (sería lo mismo que el NOT IN, EXCEPT)

select ProductId Codigo, Name Nombre, ListPrice Precio from Production.Product p

where not exists

(select ProductID from Sales.SalesOrderDetail v where p.ProductID = v.ProductID)

--es lo mismo lo que pongamos en el select de la subconsulta, podría ir 1, \*, etc

--SOME / ANY >> equivalente a EXISTS

--En esta query traemos los productos que sean mayores a cualquiera de los primeros 5 productos

select ProductID Codigo, Name Nombre, ListPrice Precio from Production.Product

where ListPrice > any

(select top 5 ListPrice from Production.Product where ListPrice > 0)

order by ListPrice

--ALL >> equivalente a NOT EXISTS

--En este caso pedimos que sea mayor a todos los primeros 5 productos

select ProductID Codigo, Name Nombre, ListPrice Precio from Production.Product

where ListPrice > all

(select top 5 ListPrice from Production.Product where ListPrice > 0)

order by ListPrice

-- Ejercitación --

--Laboratorio: AdventureWorks - Subconsultas

--1. Listar todos los productos cuyo precio sea inferior al precio promedio de todos los productos.

--Tablas: Production.Product

--Campos: Name, ListPrice

select Name, ListPrice [Precio], (select AVG(ListPrice) from Production.Product) [Precio Promedio]

from Production.Product

where ListPrice < (select AVG(ListPrice) from Production.Product)

and ListPrice > 0

order by ListPrice desc

--2. Listar el nombre, precio de lista, precio promedio y diferencia de precios entre

--cada producto y el valor promedio general.

--Tablas: Production.Product

--Campos: Name, ListPrice

select Name, ListPrice, (select avg(ListPrice) from Production.Product) [PrecioPromedio],

(select avg(ListPrice) from Production.Product) - ListPrice [DiferenciaDePrecio]

from Production.Product where ListPrice > 0 order by ListPrice desc

--3. Mostrar el o los códigos del producto más caro.

--Tablas: Production.Product

--Campos: ProductID,ListPrice

select ProductId, ListPrice from Production.Product

where (select max(ListPrice) from Production.Product) = ListPrice

--4. Mostrar el producto más barato de cada subcategoría. mostrar subcategoría, código

--de producto y el precio de lista más barato ordenado por subcategoría.

--Tablas: Production.Product

--Campos: ProductSubcategoryID, ProductID, ListPrice

select ProductID, ProductSubcategoryID, ListPrice from Production.Product p1

where ListPrice = (select min(ListPrice) from Production.Product p2

where p1.ProductSubcategoryID = p2.ProductSubcategoryID)

--Laboratorio: Exists / Not Exists--

--1. Mostrar los nombres de todos los productos presentes en la subcategoría de ruedas.

--Tablas: Production.Product, Production.ProductSubcategory

--Campos: ProductSubcategoryID, Name

select ProductSubcategoryID, Name from Production.Product p1

where exists (select ProductSubcategoryID from Production.ProductSubcategory p2

where p1.ProductSubcategoryID = p2.ProductSubcategoryID and Name = 'Wheels')

--Con join (si quiero traer un campo de la subconsulta, debo usar join)

select p.Name, sc.Name from Production.Product p join Production.ProductSubcategory sc

on p.ProductSubcategoryID = sc.ProductSubcategoryID and sc.Name = 'wheels'

--Con IN

select Name from Production.Product where ProductSubcategoryID

in (select ProductSubcategoryID from Production.ProductSubcategory

where Name = 'Wheels')

--2. Mostrar todos los productos que no fueron vendidos.

--Tablas: Production.Product, Sales.SalesOrderDetail

--Campos: Name, ProductID

select Name, ProductID from Production.Product p

where not exists (select \* from Sales.SalesOrderDetail sod

where p.ProductID = sod.ProductID)

--3. Mostrar la cantidad de personas que no son vendedores.

--Tablas: Person.Person, Sales.SalesPerson

--Campos: BusinessEntityID

select count(\*) from Person.Person p where

not exists (select \* from Sales.SalesPerson sl

where p.BusinessEntityID = sl.BusinessEntityID)

--4. Mostrar todos los vendedores (nombre y apellido) que

--no tengan asignado un territorio de ventas.

--Tablas: Person.Person, Sales.SalesPerson

--Campos: BusinessEntityID, TerritoryID, LastName, FirstName

select FirstName, LastName from Person.Person p

where exists (select \* from Sales.SalesPerson sp

where p.BusinessEntityID = sp.BusinessEntityID and

TerritoryID is null)

--Laboratorio: IN / Not IN--

--1. Mostrar las órdenes de venta que se hayan

--facturado en territorio de estado unidos únicamente 'us'.

--Tablas: Sales.SalesOrderHeader, Sales.SalesTerritory

--Campos: CountryRegionCode, TerritoryID

select \* from Sales.SalesOrderHeader soh where TerritoryID

in (select TerritoryID from Sales.SalesTerritory where CountryRegionCode = 'US')

--2. Al ejercicio anterior agregar ordenes de Francia e Inglaterra.

--Tablas: Sales.SalesOrderHeader, Sales.SalesTerritory

--Campos: CountryRegionCode, TerritoryID

select \* from Sales.SalesOrderHeader soh where TerritoryID

in (select TerritoryID from Sales.SalesTerritory where CountryRegionCode in ('US', 'FR', 'DE'))

--3. Mostrar los nombres de los diez productos más caros.

--Tablas: Production.Product

--Campos: ListPrice

select top 10 Name, ListPrice from Production.Product

order by ListPrice desc

--

select top 10 Name, ListPrice from Production.Product where ListPrice in

(select top 10 ListPrice from Production.Product order by ListPrice desc)

--4. Mostrar aquellos productos cuya cantidad de pedidos de venta sea igual o superior a 20.

--Tablas: Production.Product, Sales.SalesOrderDetail

--Campos: Name, ProductID , OrderQty

select Name, ProductID from Production.Product

where ProductID in (select ProductID from Sales.SalesOrderDetail where OrderQty >=20)

--Laboratorio: All / Any--

--1. Mostrar los nombres de todos los productos de ruedas

--que fabrica adventure works cycles

--Tablas: Production.Product, Production.ProductSubcategory

--Campos: Name, ProductSubcategoryID

select Name from Production.Product where ProductSubcategoryID = any

(select ProductSubcategoryID from Production.ProductSubcategory where Name = 'Wheels')

--2. Mostrar los clientes ubicados en un territorio

--no cubierto por ningún vendedor

--Tablas: Sales.Customer, Sales.SalesPerson

--Campos: TerritoryID

select \* from Sales.Customer where TerritoryID <> all

(select TerritoryID from Sales.SalesPerson)

--3. Listar los productos cuyos precios de venta sean mayores

--o iguales que el precio de venta máximo de cualquier

--subcategoría de producto.

--Tablas: Production.Product

--Campos: Name, ListPrice, ProductSubcategoryID

select Name, ListPrice from Production.Product

where ListPrice >= any (select max(ListPrice)

from Production.Product group by ProductSubcategoryID)

--Operaciones DML--

--INSERT--

drop table if exists sectores

create table sectores(

codigo int not null identity,

sector varchar(25),

area varchar (50) default 'Sin determinar'

)

select \* from sectores

--Manera de insertar valores por defecto

insert into sectores default values

--Manera de insertar valores de modo manual

insert sectores (sector, area)

--o insert sectores values

values ('Licencias', 'RRHH'),

('Vacaciones', 'RRHH'),

('ART', 'Salud')

--Le agrego una primary key

alter table sectores add primary key (codigo)

--Le agrego cantidad de caracteres a la/s variable/s

alter table sectores alter column sector varchar(50)

--Inserto valores desde otra tabla (agregué caracteres anteriormente

--porque no podría insertarlos por la limitada capacidad inicial)

insert sectores (sector, area)

select Name, GroupName from HumanResources.Department

--Manera de eliminar

delete from sectores where codigo > 6

select \* from sectores

--Reseteo valor de identidad con:

--dbcc checkident ('nombre tabla', reseed, desde que valor)

dbcc checkident('dbo.sectores', reseed, 6)

--Para cargar campos con valor de identidad a elección (n numero)

--debo setear IDENTITY\_INSERT en ON

set identity\_insert dbo.sectores on

insert sectores (codigo, sector, area) values (23, 'Creditos', 'Ventas')

--Lo vuelvo a dejar en off por precaución,

--y así seguira incrementando el IDENTITY automaticamente

set identity\_insert dbo.sectores off

insert sectores (sector, area) values ('Fidelización', 'Ventas')

--Inserto valores en una tabla temporal

drop table if exists #temp\_sector

select \* into #temp\_sector from sectores

select \* from #temp\_sector

set identity\_insert #temp\_sector on

insert #temp\_sector (codigo, sector, area) values (23, 'Creditos', 'Ventas')

--Elimino/vacío datos de la tabla sectores, no toca la serie

delete from sectores

select \* from sectores

--Recupero los datos desde la tabla temporal

insert sectores (sector, area)

select sector, area from #temp\_sector

--truncate vacía la tabla, y RESETEA/FORMATEA la serie

truncate table sectores

--UPDATE--

select \* from sectores

alter table sectores add fechaCreacion date

alter table sectores add Luna varchar(50)

--Formas de updatear

update sectores set area = 'Recursos Humanos'

where area = 'RRHH'

update sectores set area = 'Manufactura'

where area = 'Manufacturing'

update sectores set sector = 'Ventas'

where sector = 'Sales'

update sectores set area = 'Recursos Humanos'

update sectores set fechaCreacion = getdate()

update sectores set fechaCreacion = null

where codigo between 1 and 6

update sectores set fechaCreacion = null

update sectores set Luna = 'menguante'

where codigo between 1 and 3

update sectores set fechaCreacion = ModifiedDate

from sectores s join HumanResources.Department d

on s.sector = d.Name

--Ejemplo de cómo no es posible acceder de esta manera (IN)--

update sectores set fechaCreacion = HumanResources.Department.ModifiedDate

where sector in (select name from HumanResources.Department)

--

--DELETE--

--IN

delete from sectores where codigo not in

(select codigo from #temp\_sector where codigo between 7 and 22)

--equivalente

delete from sectores where codigo in

(select codigo from #temp\_sector where codigo not between 7 and 22)

select \* from sectores

--JOIN

delete sectores from sectores s join HumanResources.Department d

on s.sector = d.Name

--Laboratorio: AdventureWorks--

--Operaciones DML--

--1. Clonar estructura y datos de los campos nombre, color y

--precio de lista de la tabla Production.Product en una tabla

--llamada Productos.

--Tablas: Production.Product

--Campos: ProductID, Name, Color, ListPrice

drop table if exists Productos

select ProductID, Name, Color, ListPrice into Productos

from Production.Product--

select ListPrice, ListPrice \* 1.20 [ListPrice20+]

from Productos where ListPrice > 0 order by ListPrice

select \* from Productos

--2. Aumentar un 20% el precio de lista de todos los productos.

--Tablas: Productos

--Campos: ListPrice

select ProductID, Name, Color, ListPrice

from Productos where ListPrice > 0 order by ListPrice

select ProductID, Name, Color, ListPrice

from Productos where ProductID between 438 and 444

update Productos set ListPrice = ListPrice \* 1.20--

--3. Aumentar un 20% el precio de lista de los productos del proveedor 1540.

--Tablas: Productos, Purchasing.ProductVendor

--Campos: ProductID, ListPrice, BusinessEntityID

update Productos set ListPrice = ListPrice \* 1.20

from Productos where ProductID in (select ProductID

from Purchasing.ProductVendor where BusinessEntityID = 1540)--

update Productos set ListPrice = ListPrice \* 1.20

from Productos p join Purchasing.ProductVendor pv

on p.ProductID = pv.ProductID where BusinessEntityID = 1540--

select \* from Productos

select \* from Purchasing.ProductVendor where BusinessEntityID = 1540

--4. Eliminar los productos cuyo precio sea igual a cero.

--Tablas: Productos

--Campos: precio

delete Productos where ListPrice = 0

--5. Insertar un producto dentro de la tabla Productos.

--Tener en cuenta los siguientes datos: el color de producto

--debe ser rojo, el nombre debe ser "bicicleta mountain bike",

--y el precio de lista debe ser $4000.

--Tablas: Productos

--Campos: Color, Nombre, Precio

insert into Productos (Name, Color, ListPrice) values(

'Bicicleta Mountain Bike', 'Red', 4000)

select \* from Productos

--6. Aumentar en un 15% el precio de los pedales de bicicleta.

--Tablas: Productos

--Campos: Nombre, Precio

update Productos set ListPrice = ListPrice \* 1.15 where Name like '%pedal%'

--7. Eliminar los productos cuyo nombre comience con la letra m.

--Tablas: Productos

--Campos: Nombre

delete Productos where Name like 'm%'

select \* from Productos

--8. Borrar todo el contenido de la tabla

--producto sin utilizar la instrucción Delete.

--Tablas: Productos

truncate table Productos

--9. Eliminar tabla Productos

--Tablas: Productos

drop table if exists Productos

--MERGE--

drop table if exists alumnos

create table alumnos(

legajo int primary key,

nombre varchar(50),

nota int)

insert alumnos values

(1, 'Ariel MF', 10),

(2, 'Pedro Gomez', 9),

(3, 'Julio Bossero', 6),

(4, 'Pablo Antico', 10),

(5, 'Gabriel Perez', 5)

drop table if exists alumnos\_final

create table alumnos\_final(

codigo int primary key,

alumno varchar(50),

resultado int)

insert alumnos\_final values

(1, 'Ariel Mercado Fernandez', 10),

(2, 'Pedro Gomez', 7),

(4, 'Pablo Horacio Antico', 6),

(5, 'Gabriel Perez', 8),

(6, 'Leonardo Frade', 8)

select \* from alumnos

select \* from alumnos\_final

--Comando Merge

merge alumnos A --Tabla de destino / target

using alumnos\_final AF --Tabla Origen / source

on A.legajo = AF.codigo

when matched and A.nombre <> AF.alumno or A.nota <> AF.resultado

then update set A.nombre = AF.alumno, A.nota = AF.resultado

when not matched by target

then insert (legajo, nombre, nota) values (codigo, alumno, resultado)

when not matched by source

then delete;