Trabajo Final - Octubre 2025 - SQL Potenciado por IA - Instituto CPE

Nicolás Braga - Verónica Melogno

Ejercicio 1

Estructura - Tabla Producto		
Campo	**Comentario**	**Tipo Dato**
Producto_Descripción	Primary Key	varchar(20)
Producto_Cod		int
Color		varchar(20)
Fecha_factura		date
Fecha_creacion		varchar(20)
Cliente_Cod	Foreign key a Cliente	date

Aspectos a mejorar:

1) La **Primary Key** de la tabla Producto debería ser **Producto_Cod** que es un campo con registros únicos.
2) El campo Fecha_creacion es de tipo de dato date, no varchar. En tanto el campo **Cliente_Cod** (**Foreign Key**) debe ser tipo int y no date.
3) El campo Producto_Descripcion no debe llevar tilde para un buen funcionamiento general de la base de datos.
4) El campo Producto_Descripcion está bien que sea varchar pero debe tener un máximo de caracteres de por lo menos 100 para no truncar las descripciones.

Ejercicio 2

2.1)

```
SELECT c.StoreID, SUM(soh.TotalDue) AS 'Total'
FROM Sales.Customer AS c
LEFT JOIN Sales.SalesOrderHeader AS soh ON c.CustomerID=soh.CustomerID
GROUP BY c.StoreID
ORDER BY 2 DESC
```

a) Esta query lo que retorna es la suma de ingresos por ventas que tuvo cada tienda. Este valor está agrupado según cada tienda, según su ID.
b) La cláusula Left Join se utilizó para combinar las tablas Sales.Customer y Sales.SalesOrderHeader mediante el campo CustomerID. Left Join nos combina las tablas independientemente si hay coincidencia o no entre ellas. Por lo que encontraremos valores NULL cuando tengamos clientes que no hayan comprado en la respectiva tienda.
c) La cláusula Group By se usa para agrupar los resultados de la suma de TotalDue (total de ingreso por ventas) según cada tienda.

2.2)

```
SELECT sr.Name,sr.ReasonType, count(soh.SalesOrderID) AS 'Cantidad de Órdenes'
FROM Sales.SalesOrderHeader AS soh
INNER JOIN Sales.SalesOrderHeaderSalesReason AS sohr ON soh.SalesOrderID=sohr.SalesOrderID
INNER JOIN Sales.SalesReason AS sr ON sohr.SalesReasonID=sr.SalesReasonID
WHERE sr.ReasonType = 'Other'
GROUP BY sr.Name,sr.ReasonType
HAVING count(soh.SalesOrderID)  > 1400
ORDER BY 'Cantidad de Órdenes' DESC
```

a) Esta query retorna las razones (su nombre), el tipo de razón y la cantidad de órdenes (ventas realizadas). Además, se pide que retorne solo las razones cuyo tipo de razón sea “Other” y se quiere las razones que tengan más de 1400 órdenes.
b) Pondríamos la consulta de esta forma:

```
select top 1 sr.Name, max(soh.SalesOrderID) as 'Cantidad máxima de Ordenes'
from Sales.SalesOrderHeader as soh
inner join sales.SalesOrderHeaderSalesReason as sohr on soh.SalesOrderID=sohr.SalesOrderID
inner join Sales.SalesReason as sr on sr.SalesReasonID=sohr.SalesReasonID
group by sr.Name
order by [Cantidad máxima de Ordenes] desc
```

Le agregamos el operador **max** a SalesOrderID para obtener el valor máximo de órdenes por nombre de razón. Además, elegimos el valor máximo absoluto con top 1 y ordenando la columna de forma descendente. Por otro lado sacamos el campo ReasonType ya que no interesa en este caso.

c) Se usa la cláusula HAVING para trabajar con el campo “Cantidad de órdenes” ya que siempre que usamos una función de agregación (Count en este caso), debemos usar group by y having para condicionar nuestro campo agregado, no se usa where.

Ejercicio 3

a)

```
SELECT c.CompanyName, soh.SalesOrderID, soh.TotalDue
FROM SalesLT.Customer AS c
JOIN SalesLT.SalesOrderHeader AS soh
ON c.CustomerID = soh.CustomerID
```

Esta query está mal, las tablas SalesLT.Customer y SalesLT.SalesOrderHeader no existen en nuestra base de AdventureWorks2008. El nombre correcto de las tablas es Sales.Customer y Sales.SalesOrderHeader.
Tampoco existe un campo llamado CompanyName en la tabla Customer. La query correcta sería así:

```sql
select c.CustomerID, soh.SalesOrderID, soh.TotalDue
from Sales.Customer as c
Join Sales.SalesOrderHeader as soh
on c.CustomerID=soh.CustomerID
```

Esta query corregida nos retorna los ID de Cliente y de orden, así como el Total facturado para cada cliente y cada orden.

b)

```sql
SELECT OrderQty, Name, ListPrice
FROM Sales.SalesOrderHeader JOIN Sales.SalesOrderDetail
      ON SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID
            JOIN Production.Product
      ON SalesOrderDetail.ProductID = Product.ProductID
WHERE CustomerID = 30027
```

Esta query está bien formulada. Lo que sí se podría haber usado alias para las tablas y así acortar la escritura del código en la parte de los ON. Por otro lado, como las columnas solicitadas son únicas en las 3 tablas que unimos, no es necesario indicar a qué tabla pertenecen.
Esta query nos retorna los productos, su precio y la cantidad de veces que fue ordenado (comprado).

Ejercicio 9

a) Crear el prompt adecuado para la siguiente consulta

```sql
SELECT p.Name, p.Color
FROM Production.Product p
WHERE Color =
      (SELECT Color
      FROM Production.Product
      WHERE Name = 'AWC Logo Cap')
```

Tengo en SQL Server una tabla llamada Production. Product alias p con el campo color del cual necesito seleccionar el color cuyo nombre es AWC Logo Cap, quiero resolverlo con una subconsulta.

b) Dado el siguiente enunciado, generar el Prompt adecuado para resolverlo:
“Mostrar el nombre del territorio de venta en donde se venden las órdenes cuyo total sea mayor al promedio”

Dentro de SQL Server tengo una tabla llamada Sales.SalesTerritory con el campo Name cuya clave primaria es TerritoryID, la tabla Sales.SalesOrderHeader cuya clave primaria es SalesOrderID y clave Foránea TerritoryID, con el campo TotalDue.

Necesito obtener el nombre del territorio donde se venden las órdenes cuyo total sea mayor al promedio.

Ejercicio 10

a) Crear el prompt adecuado para la siguiente consulta:

```
SELECT p.Name
FROM Production.Product p
WHERE p.ProductID in
              (SELECT ProductID
              FROM Production.TransactionHistory
              WHERE year(TransactionDate)=2004)
and p.ListPrice > (SElect MIN(ListPrice)
                                  FROM Production.Product
                                  WHERE ListPrice <> 0)
```

Dentro de SQL Server tengo una tabla llamada Production.Product con los campos Name y ListPrice y una tabla llamada Production.TransactionHistory con el campo TransactionDate. Necesito el campo Name según año de transacción igual al 2004, y ListPrice mayor al seleccionar el valor mínimo de ListPrice desde la tabla Production.Product, con la condición que sea ListPrice distinto a cero. Resolverlo con subconsultas.

b) Dado el siguiente enunciado, generar el Prompt adecuado para resolverlo:

“Mostrar las órdenes de venta que vendieron el producto más caro en abril del año 2002”

En SQL Server tengo la tabla Sales.SalesOrderDetail, Production.product y Production.TransactionHistory necesito obtener las órdenes de venta con los productos más caros según precio de lista dentro de abril del año 2002.