

```
-- G450_G3
-- Carlos Andrés Píriz Scognamiglio G3
-- Francisco Sosa G3
-- Indira Sophie G3
--
```

```
-----
/*
```

Ejercicio 1: 10 pts.

Comente al menos 4 aspectos a mejor de la estructura de la siguiente tabla:

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

```
*/
```

```
/*
```

A nuestro parecer 4 aspectos a mejorar de la estructura, podrian ser:

1 - Producto\_Cod sea autoincremental y se establezca como clave primaria, mas alla de que el campo Producto\_Descripción fuera de valores unicos( e indexado), pudiendo no ser clave primaria.

2 - Campo Color, sea numerico, y sea foreign key a una tabla Color (relacionada por ese id de color numerico)

3 - Que el campo Fecha\_creacion sea de tipo date o datetime

4 - El campo Cliente\_Cod no deberia de ser parte de esta tabla ya que entendemos que no corresponde para la tabla Producto

```
*/
```

```
--
```

```
-----
/*
```

Ejercicio 2: 10 pts.

1.1 Dada la siguiente consulta:

```
SELECT
    p.ProductNumber,
    Name,
    SUM(sod.LineTotal) AS Total
FROM
    Production.Product AS p
    RIGHT JOIN Sales.SalesOrderDetail AS sod ON p.ProductID = sod.ProductID
WHERE
    ProductNumber LIKE 'BK-[^R%][0-9][0-9]'
GROUP BY
    ProductNumber, Name
```

- Explique conceptualmente qué retorna la misma (información)
- Explique con qué intención se utilizó la cláusula RIGHT JOIN
- Explique por qué se utilizó la cláusula GROUP BY

a) Esta consulta devuelve una lista donde podemos ver el nombre del producto (Name), su numero asignado (ProductNumber) y el total de ventas de cada uno (SUM(sod.LineTotal) AS Total) cabe aclarar que puede ser nulo ese Total. Este resultado nos permite identificar las ventas de los productos que comiencen con BK no sigan con R y terminen con un guion seguido de dos digitos numericos.

b) El RIGHT JOIN nos permite asegurar que todos los registros de Sales.SalesOrderDetail aparezcan en el resultado, incluso aquellos que no tienen productos asociados en la tabla Production.Product. Sin embargo, solo se incluirán los productos de Production.Product que tengan ventas asociadas.

c) Si no utilizamos el GROUP BY la consulta no puede ser realizada ya que no se esta definiendo una agrupacion para las lineas de pedido. La clausula GROUP BY se utilizo para no traer todas las lineas de detalles de las ordenes de los productos seleccionados, sino que se quiso considerar el total de LineTotal para cada producto(de manera resumida).

```

*/
--
-----
/*
1.2 Dada la siguiente consulta:
    SELECT
        soh.OrderDate AS Fecha,
        p.Name AS Producto,
        COUNT(soh.SalesOrderID) AS CantOrders
    FROM
        Production.Product AS p
        INNER JOIN Sales.SalesOrderDetail AS sod ON P.ProductID = sod.ProductID
        INNER JOIN Sales.SalesOrderHeader AS soh ON sod.SalesOrderID =
soh.SalesOrderID
    GROUP BY
        soh.OrderDate, p.Name
    HAVING
        COUNT(soh.SalesOrderID) > 1
    ORDER BY
        COUNT(soh.SalesOrderID) DESC
a) Explique conceptualmente qué retorna la misma (información)
b) Explique con qué intención se utilizaron las cláusulas INNER JOIN
c) Explique para qué se utilizó la cláusula HAVING y por qué esto no fue realizado en la
cláusula WHERE

a) Entendemos que se quiso traer la cantidad de ordenes que se hicieron por producto y fecha
de orden
    Además solo considera los resultados que para el mismo producto y fecha tengas mas de
una orden
    Y finalmente los ordena por la cantidad de ordenes de forma descendente
b) Se utilizaron para asegurar que sean productos que tengan ordenes para las fechas que
devuelva. Porque podrian haber productos que no tuvieran ordenes relacionadas.
    Y además se consideran las ordenes que si tengan lineas de detalles.
c) Having se utilizo para filtrar que solo devolviera los registros de productos que tengan
mas de una orden para esa fecha.
    Y no se hizo en el where porque por sintaxis, el having debe ir luego del group by,
ya que trabaja sobre los resultados agrupados.
*/
--
-----
/*
Ejercicio 3: 10 pts.
Dadas las siguientes consultas resuelva:
- Si son correctas o no, justifique en caso de no serla y proceda a corregirlas
- Explique conceptualmente qué retornan (qué información)
a)
    Entendemos que la sentencia es incorrecta ya que:
        los esquemas SalesLT no existen, deberían ser Sales.
        el campo CompanyName no existe en la tabla Customer, podría ser el campo
CustomerID
    De esta manera, una posible solución, que devuelva el CustomerID, su ordenes
asociadas y el total de la deuda para cada orden, podría ser:
    SELECT
        c.CustomerID,
        soh.SalesOrderID,
        SOH.TotalDue
    FROM
        Sales.Customer AS c
        JOIN Sales.SalesOrderHeader AS soh ON c.CustomerID = soh.CustomerID
    order by
        c.CustomerID, soh.SalesOrderID;
*/
SELECT

```

```

        c.CustomerID,
        soh.SalesOrderID,
        SOH.TotalDue
FROM
    Sales.Customer AS c
    JOIN Sales.SalesOrderHeader AS soh ON c.CustomerID = soh.CustomerID
order by
    c.CustomerID, soh.SalesOrderID;
/*
b)
    Entendemos que la sentencia es correcta.
    Conceptualmente, devolveria los nombres de los productos que fueron ordenados con su
    cantidad ordenada y el precio de lista para los pedidos del cliente 30027
*/
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;
--
-----
/*

```

Ejercicio 4: 10 pts.

Desarrolle la consulta SQL que permita generar como resultado:

- Obtener todos los datos de los productos que tienen precio de lista entre 25 y 250, son exclusivamente de color Rojo, Azul, o Negro, y que el tamaño sea M o XL. Ordénelos por precio
- Muestre el resultado retornado por MSSQL Server Management Studio

```

/*
SELECT
    P.*
FROM
    Production.Product AS P
WHERE
    P.Color IN ('Red', 'Blue', 'Black')
    AND P.ListPrice BETWEEN 25 AND 250
    AND P.Size IN ('M', 'XL')
ORDER BY
    P.ListPrice;

```

```

/*
Ejercicio 4: 10 pts.
Desarrolle la consulta SQL que permita generar como resultado:
• Obtener todos los datos de los productos
  que tienen precio de lista entre 25 y 250,
  son exclusivamente de color Rojo, Azul, o Negro,
  y que el tamaño sea M o XL.
Ordénelos por precio

```

ProductID	Name	ProductNumber	Color	ReorderPoint	StandardCost	ListPrice	Size	Weight	ProductLine	Class	Style	ProductSubcategoryID	ProductModelID	rowguid	ModifiedDate
852	Full Finger Gloves, M	GL-F110-M	Black	3	15.6709	37.99	M	NULL	M	NULL	U	20	3	1084221E-1890-443E-9D87-AFCAD6358355	2004-03-11 10:01:36.827
849	Men's Sports Shorts, M	SH-M897-M	Black	3	24.7459	59.99	M	NULL	S	NULL	M	22	13	D837B435-74B9-43D3-B363-ABBEAD107BC4	2004-03-11 10:01:36.827
851	Men's Sports Shorts, XL	SH-M897-X	Black	3	24.7459	59.99	XL	NULL	S	NULL	M	22	13	B03FFE40-FE2E-44CB-A4ED-81786C3A751F	2004-03-11 10:01:36.827
865	Classic Vest, M	VE-C304-M	Blue	3	23.749	63.50	M	NULL	S	NULL	U	25	1	2E53F98E-64A1-4069-911C-E3FD5E094A1E	2004-03-11 10:01:36.827
868	Women's Mountain Shorts, M	SH-W890-M	Black	3	26.1763	69.99	M	NULL	M	NULL	W	22	37	968E3610-E583-42E8-8AB6-48A799B1774	2004-03-11 10:01:36.827
853	Women's Tights, M	TG-W091-M	Black	3	30.9334	74.99	M	NULL	S	NULL	W	24	38	4D8E186C-B8C9-4C64-B411-4995DD87E316	2004-03-11 10:01:36.827

Query executed successfully.

MIWIN10\SQLEXPRESS (13.0 SP2) | MIWIN10\carlos (53) | AdventureWorks2008 | 00:00:00 | 6 rows

```

--
-----
/*
Ejercicio 5: 10 pts.
Se necesita obtener un reporte que nos muestre
el total de ventas por cliente
desplegando Nombre del contacto y ciudad del cliente en un mismo campo con el siguiente
formato:
*/
SELECT
    -- NO ENCUENTRO UNA COLUMNA DE CITY O CIUDAD
    CONCAT(P.FirstName, ' ', P.LastName, ' de ', ST.Name) AS 'Cliente',
    SUM(TOTALES.TOTAL) AS 'Total de ventas ($)'
FROM
    Person.Person AS P
    JOIN Sales.Customer AS C ON P.BusinessEntityID = C.PersonID
    JOIN Sales.SalesTerritory AS ST ON ST.TerritoryID = C.TerritoryID
    JOIN (
        SELECT
            SOH.CustomerID,
            SUM(SOH.TotalDue) AS 'TOTAL'
            -- Podria ser subtotal si solo se quiere tener en cuenta segun
            -- SUM(SOH.SubTotal) AS 'TOTAL'
        FROM
            SALES.SalesOrderHeader AS SOH
        GROUP BY
            SOH.CustomerID
    ) AS TOTALES ON C.CustomerID = TOTALES.CustomerID
GROUP BY
    CONCAT(P.FirstName, ' ', P.LastName, ' de ', ST.Name)
ORDER BY
    2 DESC;

--
-----
/*
Ejercicio 6: 10 pts.
Se desea obtener un reporte con
el nombre de la categoría y el nombre del producto de los productos que no tienen una
orden de venta asociada.
*/
SELECT
    PC.Name AS 'Categoria',
    P.Name AS 'Producto'
FROM
    Production.Product P
    JOIN Production.ProductSubcategory PSC ON P.ProductSubcategoryID =
PSC.ProductSubcategoryID
    JOIN Production.ProductCategory PC ON PSC.ProductCategoryID = PC.ProductCategoryID
    LEFT JOIN (
        SELECT
            SOD.ProductID
        FROM
            Sales.SalesOrderDetail SOD
        GROUP BY
            SOD.ProductID
    ) AS ORDENADOS ON P.ProductID = ORDENADOS.ProductID
WHERE
    ORDENADOS.ProductID IS NULL
ORDER BY
    1, 2;

```

```

--
-----
-----
/*
Ejercicio 7: 10 pts. MAL,QUE TENGA SOLO UNA CANTIDAD
Nos piden un listado que muestre
todas aquellas órdenes que contienen un solo producto ordenado,
su precio de lista y
el id de cliente.
*/
SELECT
    DISTINCT
    ORD_UN_PRODUCTO.SalesOrderID AS 'Numero_Orden',
    P.ListPrice AS 'Precio_Lista',
    SOH.CustomerID AS 'Id_Cliente'
FROM
    Production.Product P
    JOIN
    (
        /*
        -- ORDENES QUE SOLO TIENEN UNA UNIDAD EN LA CANTIDAD ORDENADA, PERO PUEDEN
TENER VARIOS PRODUCTOS ORDENADOS, ORDERQTY
        SELECT
            SOD.SalesOrderID,
            SOD.ProductID
        FROM
            Sales.SalesOrderDetail SOD
        GROUP BY
            SOD.SalesOrderID,
            SOD.ProductID
        HAVING
            MAX(SOD.OrderQty)=1
        */

        -- ORDENES QUE TIENEN ORDENADOS UN SOLO PRODUCTO, PRODUCTID
        SELECT
            SOD.SalesOrderID,
            MIN(SOD.ProductID) AS ProductID
        FROM
            Sales.SalesOrderDetail SOD
        GROUP BY
            SOD.SalesOrderID
        HAVING
            COUNT(SOD.ProductID)=1
    ) AS ORD_UN_PRODUCTO ON P.ProductID = ORD_UN_PRODUCTO.ProductID
    JOIN Sales.SalesOrderHeader SOH ON SOH.SalesOrderID = ORD_UN_PRODUCTO.SalesOrderID
order by
    3,2,1;
-- No hay dos registros de un mismo producto que repita en una misma orden
-- Select sod.SalesOrderID,sod.ProductID,count(sod.ProductID)
-- from Sales.SalesOrderDetail sod
-- group by sod.SalesOrderID,sod.ProductID
-- order by 3 desc;

--
-----
-----
/*
Ejercicio 8: 10 pts.
Desarrolle la consulta SQL que permita generar como resultado según la tabla adjunta:
• Cantidad de órdenes y su valor total (total ventas) según el rango de ventas
(para cada rango de ventas, indicar la cantidad de órdenes y el total que suman dichas
ventas).
• Muestre el resultado retornado por MSSQL Server Management Studio.
*/

```

```

SELECT
CASE
    WHEN SOH.TotalDue>=0 AND SOH.TotalDue<100
    THEN
        '1 - 0 -- 99.9999'
    WHEN SOH.TotalDue>=100 AND SOH.TotalDue<500
    THEN
        '2 - 100 -- 499.9999'
    WHEN SOH.TotalDue>=500 AND SOH.TotalDue<1000
    THEN
        '3 - 500 -- 999.9999'
    WHEN SOH.TotalDue>=1000 AND SOH.TotalDue<10000
    THEN
        '4 - 1000 -- 9999.9999'
    ELSE
        '5 - 10000 en adelante'
END AS 'Rango',
COUNT(SOH.SalesOrderID) AS 'Cantidad de ordenes',
sum(SOH.TotalDue) AS 'Valor total'
FROM
    Sales.SalesOrderHeader SOH
GROUP BY
CASE
    WHEN SOH.TotalDue>=0 AND SOH.TotalDue<100
    THEN
        '1 - 0 -- 99.9999'
    WHEN SOH.TotalDue>=100 AND SOH.TotalDue<500
    THEN
        '2 - 100 -- 499.9999'
    WHEN SOH.TotalDue>=500 AND SOH.TotalDue<1000
    THEN
        '3 - 500 -- 999.9999'
    WHEN SOH.TotalDue>=1000 AND SOH.TotalDue<10000
    THEN
        '4 - 1000 -- 9999.9999'
    ELSE
        '5 - 10000 en adelante'
END
ORDER BY
    1;

```

G450\_G3\_OBLIGATOR\_IN10(carlos (53))

```

    THEN
        '2 - 100 -- 499.9999'
    WHEN SOH.TotalDue>=500 AND SOH.TotalDue<1000
    THEN
        '3 - 500 -- 999.9999'
    WHEN SOH.TotalDue>=1000 AND SOH.TotalDue<10000
    THEN
        '4 - 1000 -- 9999.9999'
    ELSE
        '5 - 10000 en adelante'
END
ORDER BY
    1;

```

/\*  
Ejercicio 9: 10 pts.  
Se desea obtener un reporte con el nombre de los clientes que compraron algún producto cuyo nombre del modelo contenga la palabra "Mountain". Realizarlo sólo con subconsultas (no usar JOIN).  
\*Tip: tabla ProductModel.  
\*/

100 %

Resultado Messages

	Rango	Cantidad de ordenes	Valor total
1	1 - 0 - 99.9999	11093	497802.774
2	2 - 100 - 499.9999	1765	286618.1907
3	3 - 500 - 999.9999	4080	3062299.6919
4	4 - 1000 - 9999.9999	12563	34023002.1614
5	5 - 10000 en adelante	1964	102837862.0066

Query executed successfully. MIWIN10\SQLEXPRESS (13.0 SP2) MIWIN10\carlos (53) AdventureWorks2008 00:00:00 5 rows

```

--
-----
/*
Ejercicio 9: 10 pts.
Se desea obtener un reporte con el nombre de los clientes que compraron algún producto cuyo
nombre del modelo
contenga la palabra "Mountain". Realizarlo sólo con sub-consultas (no usar JOIN).
*Tip: tabla ProductModel.
*/
SELECT
    P.FirstName,
    P.LastName
FROM
    Person.Person AS P
WHERE
    P.BusinessEntityID IN
    (
        -- Asumo que cada persona tiene un id de cliente unico, no mas de uno
        SELECT
            C.PersonID
        FROM
            Sales.Customer AS C
        WHERE
            C.CustomerID IN
            (
                -- Selecciono los id de clientes unicos que esten asociados a
esas ids de ordenes unicas
                SELECT
                    SOH.CustomerID
                FROM
                    Sales.SalesOrderHeader SOH
                WHERE
                    SOH.SalesOrderID IN
                    (
                        -- Selecciono los numeros de ordenes unicos que
contengan alguno de esos productos
                        SELECT
                            SOD.SalesOrderID
                        FROM
                            Sales.SalesOrderDetail SOD
                        WHERE
                            SOD.ProductID IN
                            (
                                -- Selecciono los Ids de productos
que en modelo contengan 'Mountain'
                                SELECT
                                    P.ProductID
                                FROM
                                    Production.Product AS P
                                WHERE
                                    P.ProductModelID IN
                                    (
                                        SELECT
                                            PM.ProductModelID
                                        FROM
                                            Production.ProductModel AS PM
                                        WHERE
                                            PM.Name LIKE
'%Mountain%'
                                    )
                                )
                            )
                        )
                    )
                )
            )
        )
    )
GROUP BY
    SOD.SalesOrderID

```

```

        )
        GROUP BY
        SOH.CustomerID
    )
ORDER BY 1,2
;
--

```

```

/*

```

Ejercicio 10: 10 pts.

-Se detectó una persona que tiene una tarjeta fraudulenta y está registrada como una persona en la Base de datos de nuestra empresa. Necesitamos comunicar a la dirección si podemos saber el nombre y el número de tarjeta de dicha persona.

Los datos que poseemos son los siguientes:

- La tarjeta en cuestión sabemos que comienza con el número 7 y termina con 105
- Sabemos que la tarjeta no vence en el segundo semestre del año.
- Y por último sabemos que el Apellido de la persona es menor a 4 caracteres.

Debemos formular la consulta que nos devuelva todos los datos concatenados, tal que cumpla con el siguiente formato

de ejemplo:

```

*/

```

```

SELECT

```

```

    CONCAT('Nombre Completo: ',P.FirstName,' ',p.LastName,' - Tarjeta: ',cc.CardNumber,'
Ven: ',CC.ExpMonth,'/',CC.ExpYear) as 'Datos_Tarjeta'

```

```

FROM

```

```

    Sales.CreditCard AS CC

```

```

    JOIN Sales.PersonCreditCard PCC ON CC.CreditCardID = PCC.CreditCardID

```

```

    JOIN Person.Person P ON P.BusinessEntityID = PCC.BusinessEntityID

```

```

WHERE

```

```

    CC.CardNumber LIKE '7%105'

```

```

    AND NOT(CC.ExpMonth BETWEEN 7 AND 12)

```

```

    AND LEN(RTRIM(LTRIM(P.LastName)))<4

```

```

ORDER BY

```

```

    1;

```