BASE DE DATOS DISTRIBUIDAS **3TM2**
PRACTICA 1: REPASO DE CONSULTAS SQL Y USO DE SERVIDORES VINCULADOS

INTEGRANTES: UGALDE ESCOBAR DONALDO PATRICIO
MIRAMAR CARDOSO ALEYDI

CONSULTA 1: Encuentra los 10 productos más vendidos en 2014, mostrando nombre del producto, cantidad total vendida y nombre del cliente.

```
WITH ProductSales2024 AS (
    SELECT
        pr.ProductID,
        pr.Name AS ProductName,
        sod.SalesOrderID,
        soh.OrderDate,
        c.CustomerID,
        CONCAT(p.FirstName, ', ', p.LastName) AS CustomerName,
        sod.OrderQty
    FROM Sales.SalesOrderHeader soh
    JOIN Sales.SalesOrderDetail sod
        ON sod.SalesOrderID = soh.SalesOrderID
    JOIN Production.Product pr
        ON pr.ProductID = sod.ProductID
    JOIN Sales.Customer c
        ON c.CustomerID = soh.CustomerID
    JOIN Person.Person p
        ON p.BusinessEntityID = c.PersonID
    WHERE
        soh.OrderDate >= '20240101'
        AND soh.OrderDate <  '20250101'
        AND p.PersonType = 'IN' --cliente
),
Agg AS (
    SELECT
        ProductID,
        ProductName,
        SUM(OrderQty) AS AmountOfSales
    FROM ProductSales2024
    GROUP BY ProductID, ProductName
),
SalesByCustomer as (
    select
        distinct(CustomerName),
        sum(OrderQty) as st
    from ProductSales2024
    group by CustomerName
),
T10Products as (
    select top 10
    ProductID
    from Agg
    order by AmountOfSales desc
)
```

```
select
    a.ProductName,
    a.AmountOfSales,
    sbc.CustomerName,
    sbc.st
from Agg a
join ProductSales2024 ps on a.ProductID = ps.ProductID
join SalesByCustomer sbc on ps.CustomerName = sbc.CustomerName
where a.ProductID in (select * from T10Products)
group by a.ProductName, a.AmountOfSales, sbc.CustomerName, sbc.st
order by a.AmountOfSales desc, sbc.st desc;
```

Agregar el precio unitario promedio (AVG(UnitPrice)) y filtra solo productos con ListPrice > 1000.

```
WITH ProductSales2024 AS (
    SELECT
        pr.ProductID,
        pr.Name AS ProductName,
        pr.ListPrice,
        sod.SalesOrderID,
        soh.OrderDate,
        c.CustomerID,
        CONCAT(p.FirstName, ', ', p.LastName) AS CustomerName,
        sod.OrderQty,
        sod.UnitPrice
    FROM Sales.SalesOrderHeader soh
    JOIN Sales.SalesOrderDetail sod
        ON sod.SalesOrderID = soh.SalesOrderID
    JOIN Production.Product pr
        ON pr.ProductID = sod.ProductID
    JOIN Sales.Customer c
        ON c.CustomerID = soh.CustomerID
    JOIN Person.Person p
        ON p.BusinessEntityID = c.PersonID
    WHERE
        soh.OrderDate >= '20240101'
        AND soh.OrderDate <  '20250101'

),
Agg AS (
    SELECT
        ProductID,
        ProductName,
        ListPrice,
        SUM(OrderQty) AS AmountOfSales,
        avg(UnitPrice) as AvgUnitPrice
    FROM ProductSales2024
    GROUP BY ProductID, ProductName, ListPrice
),
```

```
SalesByCustomer as (
    select
        distinct(CustomerName),
        sum(OrderQty) as TotalBought
    from ProductSales2024
    group by CustomerName
),
T10Products as (
    select top 10
    ProductID
    from Agg
    where ListPrice >= 1000
    order by AmountOfSales desc
)
select
    a.ProductName,
    a.AmountOfSales,
    sbc.CustomerName,
    sbc.TotalBought,
    a.AvgUnitPrice,
    a.ListPrice
from Agg a
join ProductSales2024 ps on a.ProductID = ps.ProductID
join SalesByCustomer sbc on ps.CustomerName = sbc.CustomerName
where a.ProductID in (select * from T10Products)
group by a.ProductName, a.AmountOfSales, sbc.CustomerName,
sbc.TotalBought, a.AvgUnitPrice, a.ListPrice
order by a.AmountOfSales desc, sbc.TotalBought desc;
```

Conclusión:

La consulta 2 construye primero un conjunto de ventas del año 2024 con toda la información relevante: producto, cliente, cantidad vendida y también datos de precio como el ListPrice y el UnitPrice. Luego se agrupan esos datos para calcular cuánto se vendió por producto y el precio promedio real de venta. Después calcula cuánto compró cada cliente en total y finalmente selecciona los 10 productos más vendidos, pero solo aquellos cuyo precio de lista sea mayor o igual a 1000. El resultado muestra los productos más vendidos dentro del segmento “caro”, cuánto se vendieron y qué clientes compraron más, incluyendo información de precios.
La consulta 1 sigue prácticamente la misma lógica estructural: crea el conjunto base de ventas 2024, agrupa para obtener el total vendido por producto, calcula el total comprado por cliente y selecciona el top 10 de productos más vendidos. La diferencia es que es una versión más simple: no incluye análisis de precios (no usa ListPrice ni promedios), no filtra por precio mínimo y sí filtra por tipo de cliente individual.
En esencia, la consulta 1 es una variante más básica de la consulta 2. Ambas resuelven el problema con la misma estrategia (CTEs para organizar la información,

luego agregaciones y por último un top 10), pero la consulta 2 se analiza en un enfoque económico por el precio, mientras que la consulta 1 se enfoca solo en volumen de ventas.

CONSULTA 2: Lista los empleados que han vendido más que el promedio de ventas por empleado en el territorio 'Northwest'.

1. Requisito adicional: aplicar subconsultas.

```
--Listar los empleados que han vendido mas que el promedio (1)
 select BusinessEntityID, SUM(SalesOrderHeader.TotalDue) AS
TotalVendido
from Sales.SalesOrderHeader Join Person.Person on
SalesOrderHeader.SalesPersonID = Person.BusinessEntityID
where TerritoryID =1 and PersonType = 'SP'
Group By Person.BusinessEntityID
Having SUM (SalesOrderHeader.TotalDue) > (
     Select AVG(SumaPorEmpleado) AS PromedioVentasEmpNorthWest from (
               Select sum(TotalDue) as SumaPorEmpleado from
Sales.SalesOrderHeader   join Person.Person on
Sales.SalesOrderHeader.SalesPersonID =      Person.BusinessEntityID
where SalesOrderHeader.TerritoryID = 1 and      Person.PersonType =
'SP'
group by Person.BusinessEntityID) as Ventas)
```

2. Una vez resuelta la consulta convierte la subconsulta en un CTE (Common Table Expresión).

```
WITH SumaXEmpleado AS (
Select sum(TotalDue) as SumaPorEmpleado, Person.BusinessEntityID as
IDVendedor
from Sales.SalesOrderHeader join Person.Person
on Sales.SalesOrderHeader.SalesPersonID = Person.BusinessEntityID
where SalesOrderHeader.TerritoryID = 1
and Person.PersonType = 'SP'
group by Person.BusinessEntityID
),
PromedioEmpleado as (
select AVG(SumaPorEmpleado) as PromedioXEmpleado
from SumaXEmpleado)
Select  SumaPorEmpleado, IDVendedor
from SumaXEmpleado, PromedioEmpleado
where SumaPorEmpleado > PromedioXEmpleado
```

3. Documenta la solución inicial y solución con la variante solicitada.

La Solución (subconsultas) 1 Es más larga, pero es una buena base para arrancar con la logica es decir realizarlo paso a paso, pero te puedes perder fácilmente entre tantas líneas

sin embargo, la solución 2 te das cuenta cómo puedes manejar las tablas como una variable en programación que puedes manipular y mover más fácilmente ya conociendo el resultado que te arroja y sobre todo para ahorrar recursos si es una consulta que se realizara constantemente.

CONSULTA 3: Calcular ventas totales por territorio y año, mostrando solo aquellos con más de 5 órdenes y ventas mayores a 1 millón, ordenado por ventas descendente.

```
select
    st.Name,
    year(soh.OrderDate) as OrderYear,
    sum(sod.LineTotal) as totalSales
from Sales.SalesTerritory st
join Sales.SalesOrderHeader soh
    on soh.TerritoryID = st.TerritoryID
join Sales.SalesOrderDetail sod
    on sod.SalesOrderID = soh.SalesOrderID
group by
    st.Name,
    year(soh.OrderDate)
having
    count(distinct soh.SalesOrderID) >= 5
    and sum(sod.LineTotal) > 1000000
order by
    totalSales desc
    ;
```

Agregar desviación estándar por ventas:

```
with OrderTotals as (
    select
        st.Name,
        year(soh.OrderDate) as OrderYear,
        soh.SalesOrderID,
        sum(sod.LineTotal) as OrderTotal
    from Sales.SalesTerritory st
    join Sales.SalesOrderHeader soh
        on soh.TerritoryID = st.TerritoryID
    join Sales.SalesOrderDetail sod
        on sod.SalesOrderID = soh.SalesOrderID
    group by
```

```
        st.Name,
        year(soh.OrderDate),
        soh.SalesOrderID
)
select
    Name,
    OrderYear,
    sum(OrderTotal) as totalSales,
    count(*) as orderCount,
    stdev(OrderTotal) as stdevOrderSales
from OrderTotals
group by
    Name,
    OrderYear
having
    count(*) >= 5
order by
    stdevOrderSales desc;
```

Conclusión:

En la primera consulta se realizan varios JOIN para relacionar los territorios con las órdenes y sus detalles, usando los ID correspondientes, con el objetivo de obtener el nombre del territorio, el año de cada orden y el importe total de cada compra mediante la suma de LineTotal. Luego se agrupa por nombre del territorio y año, se cuentan las órdenes distintas y se filtran los resultados para mostrar solo aquellos grupos que tengan más de cinco órdenes y ventas superiores a un millón, ordenando finalmente de forma descendente según el total vendido.
En la segunda variante se utiliza una CTE para dividir el proceso en dos etapas: primero se calculan los totales por cada orden individual agrupando por territorio, año y número de orden; después, en la consulta principal, se agrupa nuevamente por territorio y año para obtener el total de ventas, el conteo de órdenes y la desviación estándar de las ventas mediante la función STDEV, filtrando los casos con más de cinco órdenes y ordenando los resultados de forma descendente.

CONSULTA 4: Encuentra vendedores que han vendido TODOS los productos de la categoría "Bikes".

```
WITH NoProdBikes AS (
select COUNT(Productos.ProductID) as TotalArticulosBikes
from Production.ProductSubcategory as SubCat
join Production.ProductCategory as Cat on SubCat.ProductCategoryID =
Cat.ProductCategoryID
JOIN Production.Product as Productos on Productos.ProductSubcategoryID=
SubCat.ProductSubcategoryID
where Cat.ProductCategoryID =1
),
```

```
ProdDistintXVende AS (SELECT CabezaVenta.SalesPersonID, COUNT(DISTINCT
Productos.ProductID) as ProdDistintos
from Sales.SalesOrderHeader as CabezaVenta
join Person.Person as Empleados on CabezaVenta.SalesPersonID =
Empleados.BusinessEntityID
join Sales.SalesOrderDetail as DetalleVenta on CabezaVenta.SalesOrderID
= DetalleVenta.SalesOrderID
join Production.Product as Productos on Productos.ProductID =
DetalleVenta.ProductID
join Production.ProductSubcategory as SubCat on
Productos.ProductSubcategoryID = SubCat.ProductSubcategoryID
join Production.ProductCategory as CategoriP on
CategoriP.ProductCategoryID = SubCat.ProductCategoryID
where CategoriP.ProductCategoryID=1
group by CabezaVenta.SalesPersonID)
Select *
from ProdDistintXVende, NoProdBikes
where ProdDistintXVende.ProdDistintos = NoProdBikes.TotalArticulosBikes
```

Cambia a categoría "Clothing" (ID=3 en la version AW2022).

```
WITH NoProdClothing AS (
select COUNT(Productos.ProductID) as TotalArticulosClothing
from Production.ProductSubcategory as SubCat
join Production.ProductCategory as Cat on SubCat.ProductCategoryID =
Cat.ProductCategoryID
JOIN Production.Product as Productos on Productos.ProductSubcategoryID=
SubCat.ProductSubcategoryID
where Cat.ProductCategoryID =3
),
ProdDistintXVende AS (SELECT CabezaVenta.SalesPersonID, COUNT(DISTINCT
Productos.ProductID) as ProdDistintos
from Sales.SalesOrderHeader as CabezaVenta
join Person.Person as Empleados on CabezaVenta.SalesPersonID =
Empleados.BusinessEntityID
join Sales.SalesOrderDetail as DetalleVenta on CabezaVenta.SalesOrderID
= DetalleVenta.SalesOrderID
join Production.Product as Productos on Productos.ProductID =
DetalleVenta.ProductID
join Production.ProductSubcategory as SubCat on
Productos.ProductSubcategoryID = SubCat.ProductSubcategoryID
join Production.ProductCategory as CategoriP on
CategoriP.ProductCategoryID = SubCat.ProductCategoryID
where CategoriP.ProductCategoryID=3
group by CabezaVenta.SalesPersonID)
Select *
from ProdDistintXVende, NoProdClothing
where ProdDistintXVende.ProdDistintos =
NoProdClothing.TotalArticulosClothing
```

Cuenta cuantos productos por categoría maneja cada vendedor.

```
SELECT CabezaVenta.SalesPersonID, COUNT(DISTINCT Productos.ProductID)
as ProdDistintos
from Sales.SalesOrderHeader as CabezaVenta
join Person.Person as Empleados on CabezaVenta.SalesPersonID =
Empleados.BusinessEntityID
join Sales.SalesOrderDetail as DetalleVenta on CabezaVenta.SalesOrderID
= DetalleVenta.SalesOrderID
join Production.Product as Productos on Productos.ProductID =
DetalleVenta.ProductID
join Production.ProductSubcategory as SubCat on
Productos.ProductSubcategoryID = SubCat.ProductSubcategoryID
join Production.ProductCategory as CategoriP on
CategoriP.ProductCategoryID = SubCat.ProductCategoryID
group by CabezaVenta.SalesPersonID
```

Conclusión:

Para el ejercicio 4:
Este Primer variante del ejercicio fue muy complejo de hacer debido a que lo tuve que dividir en dos conceptos, el primero de ellos una consulta que nos diera el # total de productos de la categoría Bikes, es decir contar las subcategorías con ID (1,2,3).
La segunda de ellas fue realizar una suma en la cual nos diera el # distinto de productos que ha vendido cada vendedor. Por último, juntarlas en una sola tabla usando WITH por practicidad y filtrar solo vendedores que tuvieran el mismo # de productos vendidos que el número de productos en categoría bikes.
Segunda variante:
Para esta consulta, fue solamente cambiar el id de la categoría 1 de bikes al 3 de Clothing retomando el anterior código de la consulta y realizando ese pequeño cambio.
Tercera variante:
Para esta consulta de igual forma fue retomar parte de la anterior consulta y modificar solamente el filtro que dejara de tomar solamente el ID 1 o 3, y simplemente no tuviera restricción y asi consultar las 4 categorías.
Explicación final o general del ejercicio 4:
Pues son consultas bastante complejas en las que es fácil de perderse, es mejor ir separando la información de a poco o, es decir, en consultas más sencillas, y al final realizar la o las últimas operaciones de modo que se trabaje con un concepto o una tabla y no directamente con una consulta completa (código), es mas fácil de visualizar y de entender, dado los visto en clase y el entendimiento que ha generado esta practica, estamos llevando a cabo el concepto de la división Relacional

CONSULTA 5: Determinar el producto más vendido de cada categoría de producto, considerando el escenario de que el esquema SALES se encuentra en una instancia (servidor) A y el esquema PRODUCTION en otra instancia (servidor) B.

```
    WITH ProductSales AS (
    SELECT
        pc.ProductCategoryID,
        pc.Name AS CategoryName,
        p.ProductID,
        p.Name AS ProductName,
        SUM(sod.OrderQty) AS TotalSold
    FROM SV_P1.Sales.SalesOrderDetail sod
    JOIN SV_P2.Production.Product p
        ON p.ProductID = sod.ProductID
    JOIN Production.ProductSubcategory ps
        ON ps.ProductSubcategoryID = p.ProductSubcategoryID
    JOIN Production.ProductCategory pc
        ON pc.ProductCategoryID = ps.ProductCategoryID
    GROUP BY
        pc.ProductCategoryID,
        pc.Name,
        p.ProductID,
        p.Name
),

RankedProducts AS (
    SELECT *,
           ROW_NUMBER() OVER (
               PARTITION BY ProductCategoryID
               ORDER BY TotalSold DESC
           ) AS rn
    FROM ProductSales
)

SELECT
    CategoryName,
    ProductName,
    TotalSold
FROM RankedProducts
WHERE rn = 1
ORDER BY CategoryName;
```