UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERIA Y TECNOLOGIAS AVANZADAS

UPIITA

INSTITUTO POLITÉCNICO NACIONAL

Base De Datos Distribuidas

"Repaso de consultas y servidores vinculados"

Grupo: 1TM5

ALUMNO:

- Vargas Clemente Leonel Zajid

BOLETA:

-2024640128

1. Introducción

En la presente práctica se trabajó con la base de datos **AdventureWorks2022** en SQL Server Management Studio con el objetivo de aplicar consultas avanzadas utilizando múltiples tablas, subconsultas, CTE, funciones de agregación, agrupamientos, filtros con HAVING y el uso de servidores vinculados.

2. Desarrollo

Ejercicio 1

10 productos más vendidos en 2014

Objetivo

Obtener los 10 productos más vendidos en el año 2014 mostrando:

- Nombre del producto
- Cantidad total vendida
- Nombre del cliente

Análisis

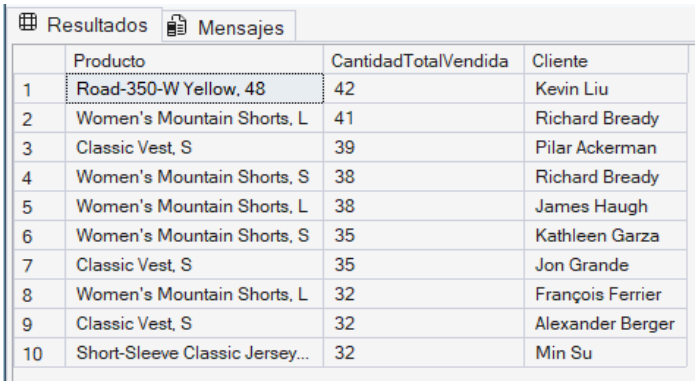
Se realizó una consulta multitabla uniendo:

- SalesOrderHeader (fecha y cliente)
- SalesOrderDetail (detalle del producto)
- Product (nombre del producto)
- Customer y Person (datos del cliente)

```sql
SELECT TOP 10
    p.Name AS Producto,
    SUM(sod.OrderQty) AS CantidadTotalVendida,
    CONCAT(pp.FirstName, ' ', pp.LastName) AS Cliente
FROM Sales.SalesOrderHeader soh
INNER JOIN Sales.SalesOrderDetail sod
    ON soh.SalesOrderID = sod.SalesOrderID
INNER JOIN Production.Product p
    ON sod.ProductID = p.ProductID
INNER JOIN Sales.Customer c
    ON soh.CustomerID = c.CustomerID
INNER JOIN Person.Person pp
    ON c.PersonID = pp.BusinessEntityID
WHERE YEAR(soh.OrderDate) = 2014
GROUP BY p.Name, pp.FirstName, pp.LastName
ORDER BY CantidadTotalVendida DESC;
```

Resultado

Se obtuvieron los 10 productos con mayor cantidad vendida durante 2014.



Resultados | Mensajes

	Producto	CantidadTotalVendida	Cliente
1	Road-350-W Yellow, 48	42	Kevin Liu
2	Women's Mountain Shorts, L	41	Richard Bready
3	Classic Vest, S	39	Pilar Ackerman
4	Women's Mountain Shorts, S	38	Richard Bready
5	Women's Mountain Shorts, L	38	James Haugh
6	Women's Mountain Shorts, S	35	Kathleen Garza
7	Classic Vest, S	35	Jon Grande
8	Women's Mountain Shorts, L	32	François Ferrier
9	Classic Vest, S	32	Alexander Berger
10	Short-Sleeve Classic Jersey...	32	Min Su

Variante

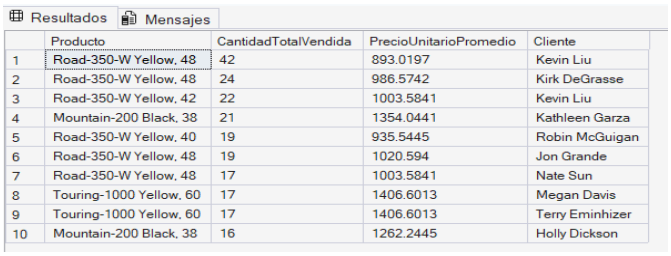
Se agregó:

- AVG(UnitPrice) para calcular precio promedio
- Filtro ListPrice > 1000

Esto permitió analizar únicamente productos de alto valor comercial.

```sql
SELECT TOP 10
    p.Name AS Producto,
    SUM(sod.OrderQty) AS CantidadTotalVendida,
    AVG(sod.UnitPrice) AS PrecioUnitarioPromedio,
    CONCAT(pp.FirstName, ' ', pp.LastName) AS Cliente
FROM Sales.SalesOrderHeader soh
INNER JOIN Sales.SalesOrderDetail sod
    ON soh.SalesOrderID = sod.SalesOrderID
INNER JOIN Production.Product p
    ON sod.ProductID = p.ProductID
INNER JOIN Sales.Customer c
    ON soh.CustomerID = c.CustomerID
INNER JOIN Person.Person pp
    ON c.PersonID = pp.BusinessEntityID
WHERE YEAR(soh.OrderDate) = 2014
AND p.ListPrice > 1000
GROUP BY p.Name, pp.FirstName, pp.LastName
ORDER BY CantidadTotalVendida DESC;
```

Resultado



Resultados | Mensajes

	Producto	CantidadTotalVendida	PrecioUnitarioPromedio	Cliente
1	Road-350-W Yellow, 48	42	893.0197	Kevin Liu
2	Road-350-W Yellow, 48	24	986.5742	Kirk DeGrasse
3	Road-350-W Yellow, 42	22	1003.5841	Kevin Liu
4	Mountain-200 Black, 38	21	1354.0441	Kathleen Garza
5	Road-350-W Yellow, 40	19	935.5445	Robin McGuigan
6	Road-350-W Yellow, 48	19	1020.594	Jon Grande
7	Road-350-W Yellow, 48	17	1003.5841	Nate Sun
8	Touring-1000 Yellow, 60	17	1406.6013	Megan Davis
9	Touring-1000 Yellow, 60	17	1406.6013	Terry Eminhizer
10	Mountain-200 Black, 38	16	1262.2445	Holly Dickson

Ejercicio 2

Empleados que vendieron más que el promedio en el territorio 'Northwest'

Objetivo

Identificar empleados cuyas ventas superan el promedio de ventas por empleado en ese territorio.

Análisis

Se utilizó:

- SUM(TotalDue) para calcular ventas totales por empleado
- Subconsulta para calcular el promedio general
- HAVING para comparar contra el promedio

```
SELECT
    e.BusinessEntityID,
    p.FirstName,
    p.LastName,
    SUM(soh.TotalDue) AS TotalVentas
FROM Sales.SalesOrderHeader soh
INNER JOIN Sales.SalesPerson sp
    ON soh.SalesPersonID = sp.BusinessEntityID
INNER JOIN HumanResources.Employee e
    ON sp.BusinessEntityID = e.BusinessEntityID
INNER JOIN Person.Person p
    ON e.BusinessEntityID = p.BusinessEntityID
INNER JOIN Sales.SalesTerritory st
    ON soh.TerritoryID = st.TerritoryID
WHERE st.Name = 'Northwest'
GROUP BY e.BusinessEntityID, p.FirstName, p.LastName
HAVING SUM(soh.TotalDue) >
(
    SELECT AVG(VentasPorEmpleado)
    FROM (
        SELECT SUM(soh2.TotalDue) AS VentasPorEmpleado
        FROM Sales.SalesOrderHeader soh2
        INNER JOIN Sales.SalesTerritory st2
            ON soh2.TerritoryID = st2.TerritoryID
        WHERE st2.Name = 'Northwest'
        GROUP BY soh2.SalesPersonID
    ) AS Promedio
);
```

Resultado

Resultados | Mensajes

	BusinessEntityID	FirstName	LastName	TotalVentas
1	284	Tete	Mensa-Annan	2608116.3755
2	280	Pamela	Ansman-Wolfe	3748246.1218
3	283	David	Campbell	4207894.6025

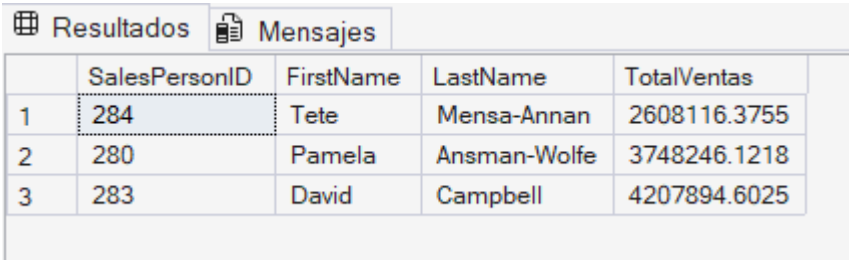
Variante

Posteriormente se resolvió usando una CTE (Common Table Expression), lo que mejora la legibilidad y organización del código.

```sql
WITH VentasEmpleado AS (
    SELECT
        soh.SalesPersonID,
        SUM(soh.TotalDue) AS TotalVentas
    FROM Sales.SalesOrderHeader soh
    INNER JOIN Sales.SalesTerritory st
        ON soh.TerritoryID = st.TerritoryID
    WHERE st.Name = 'Northwest'
    GROUP BY soh.SalesPersonID
)

SELECT
    ve.SalesPersonID,
    p.FirstName,
    p.LastName,
    ve.TotalVentas
FROM VentasEmpleado ve
INNER JOIN Person.Person p
    ON ve.SalesPersonID = p.BusinessEntityID
WHERE ve.TotalVentas > (SELECT AVG(TotalVentas) FROM VentasEmpleado);
```

Resultado


Resultados Mensajes

	SalesPersonID	FirstName	LastName	TotalVentas
1	284	Tete	Mensa-Annan	2608116.3755
2	280	Pamela	Ansman-Wolfe	3748246.1218
3	283	David	Campbell	4207894.6025

Ejercicio 3

Ventas totales por territorio y año

Objetivo

Mostrar ventas agrupadas por territorio y año con condiciones:

- Más de 5 órdenes
- Ventas mayores a 1,000,000
- Orden descendente

Análisis

Se utilizó:

- COUNT() para número de órdenes
- SUM() para total de ventas
- HAVING para aplicar filtros sobre agregados

- ORDER BY DESC

```sql
SELECT
    st.Name AS Territorio,
    YEAR(soh.OrderDate) AS Año,
    COUNT(soh.SalesOrderID) AS TotalOrdenes,
    SUM(soh.TotalDue) AS TotalVentas
FROM Sales.SalesOrderHeader soh
INNER JOIN Sales.SalesTerritory st
    ON soh.TerritoryID = st.TerritoryID
GROUP BY st.Name, YEAR(soh.OrderDate)
HAVING COUNT(soh.SalesOrderID) > 5
AND SUM(soh.TotalDue) > 1000000
ORDER BY TotalVentas DESC;
```

Resultado

Resultados | Mensajes

	Territorio	Año	TotalOrdenes	TotalVentas
1	Southwest	2013	2725	10239209.3403
2	Southwest	2012	777	9329154.3425
3	Canada	2013	1884	7010449.6994
4	Northwest	2013	2053	6759500.6713
5	Canada	2012	460	6599971.0217
6	Northwest	2012	510	5325813.0562
7	Australia	2013	3015	4702404.0504
8	Southwest	2014	2383	4437517.8076
9	France	2013	1273	4271019.2663
10	United Ki...	2013	1528	4068178.6672
11	Central	2013	151	3374336.2992
12	Northwest	2014	1807	3355402.8175
13	Southeast	2012	167	3344683.6085
14	Central	2012	130	3334867.9788
15	Northeast	2012	117	3272239.7992

Variante

En la variante se agregó:

- STDEV(TotalDue) para medir dispersión de ventas

```sql
SELECT
    st.Name AS Territorio,
    YEAR(soh.OrderDate) AS Año,
    COUNT(soh.SalesOrderID) AS TotalOrdenes,
    SUM(soh.TotalDue) AS TotalVentas,
    STDEV(soh.TotalDue) AS DesviacionEstandar
FROM Sales.SalesOrderHeader soh
INNER JOIN Sales.SalesTerritory st
    ON soh.TerritoryID = st.TerritoryID
GROUP BY st.Name, YEAR(soh.OrderDate)
HAVING COUNT(soh.SalesOrderID) > 5
AND SUM(soh.TotalDue) > 1000000
ORDER BY TotalVentas DESC;
```

Resultado

Resultados | Mensajes

	Territorio	Año	TotalOrdenes	TotalVentas	DesviacionEstandar
1	Southwest	2013	2725	10239209.3403	13470.4188703684
2	Southwest	2012	777	9329154.3425	23693.0432287992
3	Canada	2013	1884	7010449.6994	13359.6078579698
4	Northwest	2013	2053	6759500.6713	12654.1872740093
5	Canada	2012	460	6599971.0217	24167.4810564263
6	Northwest	2012	510	5325813.0562	20150.9169044465
7	Australia	2013	3015	4702404.0504	3719.04527457183
8	Southwest	2014	2383	4437517.8076	7343.9044753176
9	France	2013	1273	4271019.2663	12923.6957600484
10	United Ki...	2013	1528	4068178.6672	9889.93309207325
11	Central	2013	151	3374336.2992	29231.3553332069
12	Northwest	2014	1807	3355402.8175	8268.1738993602
13	Southeast	2012	167	3344683.6085	26445.9089480054
14	Central	2012	130	3334867.9788	29430.5755643707

Ejercicio 4

Vendedores que han vendido TODOS los productos de la categoría "Bikes"

Objetivo

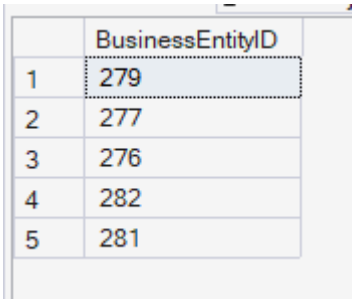
Encontrar vendedores que hayan vendido cada producto existente dentro de la categoría.

Análisis

Se utilizó doble NOT EXISTS, técnica común para resolver divisiones relacionales.

```sql
SELECT sp.BusinessEntityID
FROM Sales.SalesPerson sp
WHERE NOT EXISTS (
    SELECT p.ProductID
    FROM Production.Product p
    INNER JOIN Production.ProductSubcategory ps
        ON p.ProductSubcategoryID = ps.ProductSubcategoryID
    INNER JOIN Production.ProductCategory pc
        ON ps.ProductCategoryID = pc.ProductCategoryID
    WHERE pc.Name = 'Bikes'
    AND NOT EXISTS (
        SELECT 1
        FROM Sales.SalesOrderDetail sod
        INNER JOIN Sales.SalesOrderHeader soh
            ON sod.SalesOrderID = soh.SalesOrderID
        WHERE sod.ProductID = p.ProductID
        AND soh.SalesPersonID = sp.BusinessEntityID
    )
);
```

Resultado



	BusinessEntityID
1	279
2	277
3	276
4	282
5	281

Variante

Se modificó la categoría a "Clothing" para validar comportamiento dinámico.

```sql
SELECT sp.BusinessEntityID
FROM Sales.SalesPerson sp
WHERE NOT EXISTS (
    SELECT p.ProductID
    FROM Production.Product p
    INNER JOIN Production.ProductSubcategory ps
        ON p.ProductSubcategoryID = ps.ProductSubcategoryID
    INNER JOIN Production.ProductCategory pc
        ON ps.ProductCategoryID = pc.ProductCategoryID
    WHERE pc.Name = 'Clothing'
    AND NOT EXISTS (
        SELECT 1
        FROM Sales.SalesOrderDetail sod
        INNER JOIN Sales.SalesOrderHeader soh
            ON sod.SalesOrderID = soh.SalesOrderID
        WHERE sod.ProductID = p.ProductID
        AND soh.SalesPersonID = sp.BusinessEntityID
    )
);
```

Ejercicio 5

Producto más vendido por categoría usando Servidores Vinculados

Objetivo

Simular escenario distribuido donde:

- SALES está en Servidor A
- PRODUCTION está en Servidor B

Scrip de Creacion del Servidor Vinvulado

```
EXEC sp_addlinkedserver
    @server = 'ServidorRemoto',
    @provider = 'MSOLEDBSQL19',
    @srvproduct = '',
    @datasrc = 'DESKTOP-0I0029G\SQLEXPRESS',
    @provstr = 'Encrypt=Optional;TrustServerCertificate=Yes;';
GO

EXEC sp_addlinkedsrvlogin
    @rmtsrvname = 'ServidorRemoto',
    @useself = 'TRUE';
GO
```

Scrip de consultas

```
WITH VentasPorProducto AS (

    SELECT
        ProductID,
        SUM(OrderQty) AS TotalVendido
    FROM Sales.SalesOrderDetail
    GROUP BY ProductID
),
ProductosPorCategoria AS (

    SELECT
        P.ProductID,
        P.Name AS NombreProducto,
        PC.Name AS NombreCategoria,
        V.TotalVendido
    FROM [ServidorRemoto].[AdventureWorks2022].[Production].[Product] P
    INNER JOIN [ServidorRemoto].[AdventureWorks2022].[Production].[ProductSubcategory] PS
        ON P.ProductSubcategoryID = PS.ProductSubcategoryID
    INNER JOIN [ServidorRemoto].[AdventureWorks2022].[Production].[ProductCategory] PC
        ON PS.ProductCategoryID = PC.ProductCategoryID
    INNER JOIN VentasPorProducto V
        ON P.ProductID = V.ProductID
),
RankingProductos AS (
    SELECT
        NombreCategoria,
        NombreProducto,
        TotalVendido,
        RANK() OVER (PARTITION BY NombreCategoria ORDER BY TotalVendido DESC) AS Rnk
    FROM ProductosPorCategoria
)
SELECT
    NombreCategoria,
    NombreProducto,
    TotalVendido
FROM RankingProductos
WHERE Rnk = 1
ORDER BY TotalVendido DESC;
```

Conclusiones por Ejercicio

Ejercicio 1

En conclusión, el ejercicio reforzó el manejo correcto de agrupaciones y filtros por fecha, además de evidenciar cómo pequeños cambios en el agrupamiento modifican significativamente los resultados.

El uso de funciones agregadas como SUM() y AVG() permitió analizar tanto volumen de ventas como comportamiento de precios. También se observó la diferencia

conceptual entre UnitPrice (precio real de venta) y ListPrice (precio de catálogo), lo que impacta directamente en la interpretación de resultados.

Ejercicio 2

Este ejercicio permitió comprender cómo realizar comparaciones dinámicas dentro de una misma consulta y cómo estructurar consultas complejas de manera eficiente. En este ejercicio se profundizó en el uso de subconsultas y CTE para comparar datos agregados contra un promedio general.

La versión con CTE demostró ser más clara y organizada, facilitando la lectura del código. También se identificó la importancia de manejar correctamente valores NULL en SalesPersonID.

Ejercicio 3

Este ejercicio permitió aplicar filtros sobre funciones agregadas mediante la cláusula HAVING, diferenciándola claramente de WHERE. Se reforzó la importancia de agrupar correctamente por múltiples columnas (territorio y año) para evitar resultados incorrectos. La inclusión de STDEV() permitió analizar no solo el total de ventas, sino también la variabilidad de estas.

Ejercicio 4

Este ejercicio fortaleció el pensamiento lógico aplicado a consultas SQL avanzadas fue el más complejo desde el punto de vista lógico. Se utilizó la técnica de doble NOT EXISTS, que representa una división relacional en SQL. Se aprendió que esta estructura permite validar que un registro cumple con una condición universal

Ejercicio 5

En conclusión, este ejercicio demostró cómo SQL Server puede trabajar en arquitecturas distribuidas y cómo se integran datos de diferentes instancias en una sola consulta.