Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria

En Ingeniería Y Tecnologías Avanzadas.

Tarea:

Practica 1

Alumnos:

Mendoza Xicohtencatl Hector Jair

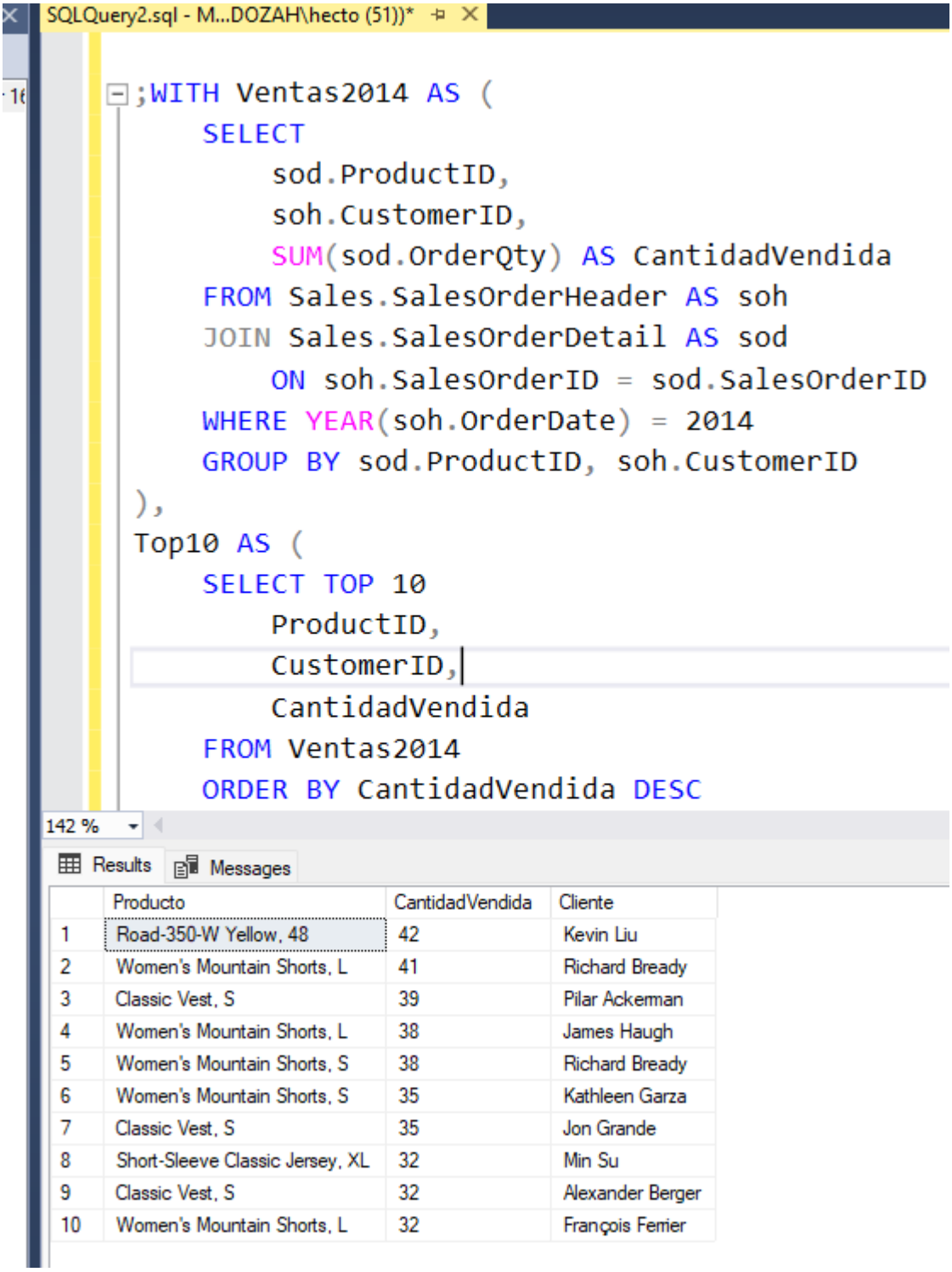
González Vázquez Rodrigo

Profesor: De la Cruz Sosa Carlos

Grupo: 3TM2

Fecha de entrega: 26/02/2026

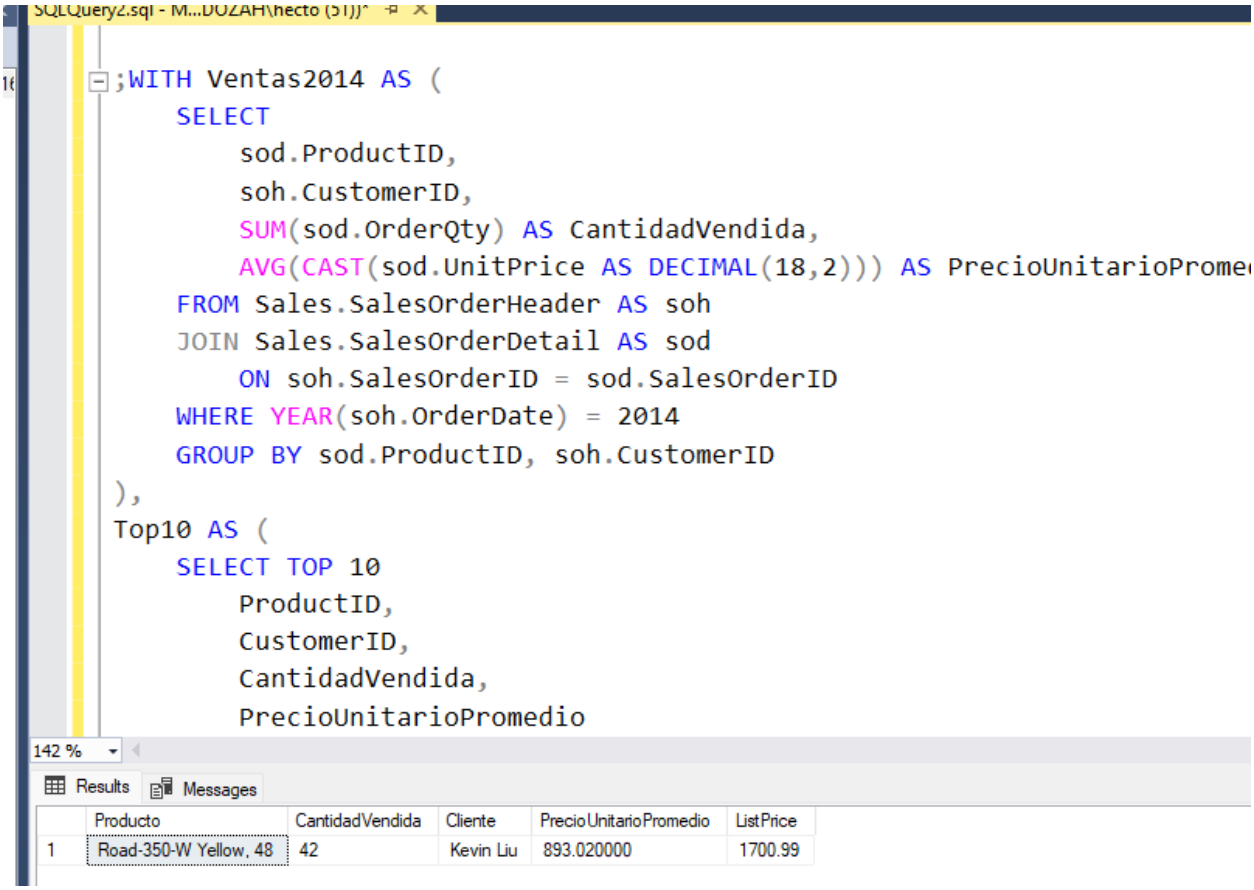
1)



	Producto	CantidadVendida	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, L	38	James Haugh
5	Women's Mountain Shorts, S	38	Richard Bready
6	Women's Mountain Shorts, S	35	Kathleen Garza
7	Classic Vest, S	35	Jon Grande
8	Short-Sleeve Classic Jersey, XL	32	Min Su
9	Classic Vest, S	32	Alexander Berger
10	Women's Mountain Shorts, L	32	François Ferrier

Se filtraron las órdenes del año 2014 y se sumó OrderQty para obtener cuántas piezas se vendieron por producto. Con eso se eligieron los 10 productos con mayor cantidad vendida. Luego se unieron tablas para mostrar el nombre del producto y el nombre del cliente (persona o tienda).

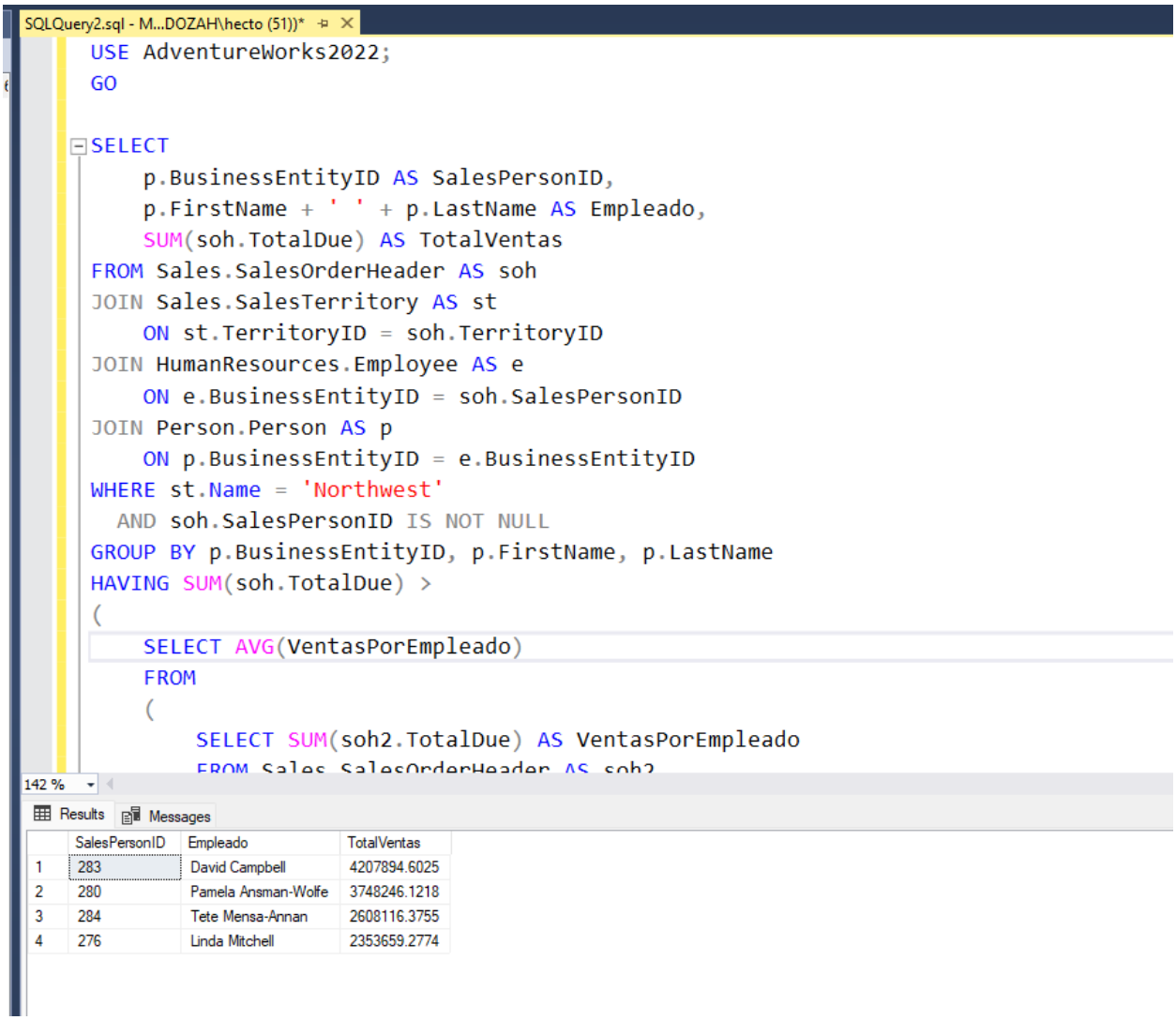
b)



A la misma consulta se le agregó el promedio del precio unitario usando AVG(UnitPrice). También se filtraron los resultados para mostrar solo productos con ListPrice > 1000.

2.

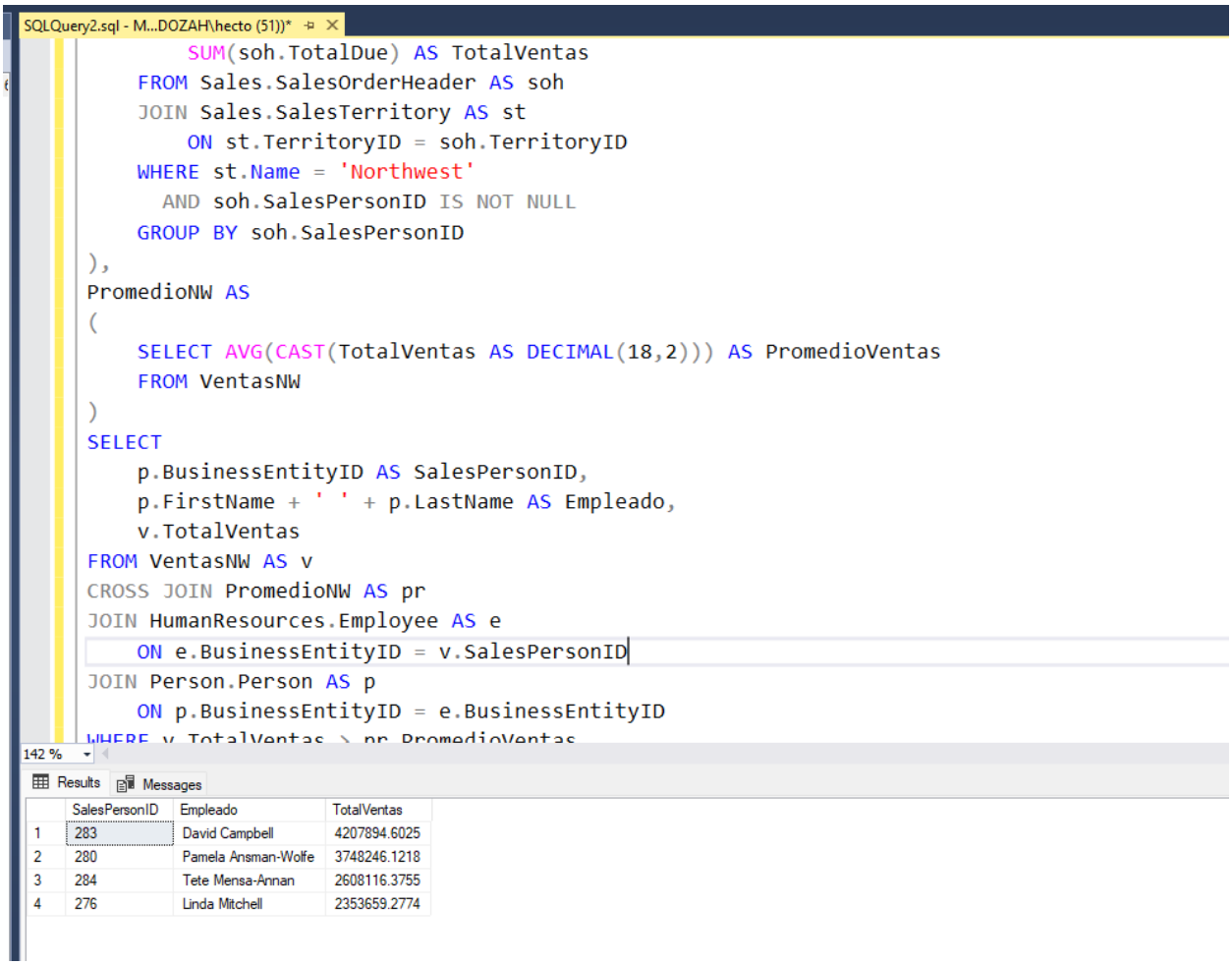
a)



En esta consulta se calcularon las ventas totales por empleado en el territorio “Northwest”. Después se obtuvo el promedio de ventas por empleado en ese mismo territorio usando una subconsulta.

Finalmente, se comparó el total de cada empleado contra ese promedio y se mostraron únicamente los que vendieron más que la media. También se unieron las tablas necesarias para mostrar el nombre completo del empleado.

b)



En esta versión se utilizó una CTE para organizar mejor la consulta. Primero se calcularon las ventas por empleado en “Northwest”, y después se obtuvo el promedio a partir de esos resultados.

Al final, se filtraron los empleados que superan ese promedio. Esta forma hace que la consulta sea más clara y fácil de entender.

3.

a)

```
SELECT
    st.Name AS Territorio,
    YEAR(soh.OrderDate) AS Anio,
    COUNT(*) AS NumOrdenes,
    SUM(soh.TotalDue) AS VentasTotales
FROM Sales.SalesOrderHeader AS soh
JOIN Sales.SalesTerritory AS st
    ON st.TerritoryID = soh.TerritoryID
GROUP BY
    st.Name,
    YEAR(soh.OrderDate)
HAVING
    COUNT(*) > 5
    AND SUM(soh.TotalDue) > 1000000
ORDER BY
```

142 %

Results | Messages

	Territorio	Anio	NumOrdenes	VentasTotales
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 Kingdom	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
16	Southwest	2011	339	3144713.0989
17	Australia	2014	2473	3071053.8419

En esta consulta se agruparon las ventas por territorio y por año. Se utilizó COUNT(*) para obtener el número de órdenes y SUM(TotalDue) para calcular el total vendido.

Después se aplicó un filtro con HAVING para mostrar únicamente los casos con más de 5 órdenes y ventas mayores a $1,000,000. Finalmente, se ordenaron los resultados de mayor a menor según las ventas.

b)

```sql
SELECT
    st.Name AS Territorio,
    YEAR(soh.OrderDate) AS Año,
    COUNT(*) AS NumOrdenes,
    SUM(soh.TotalDue) AS VentasTotales,
    STDEV(CAST(soh.TotalDue AS DECIMAL(18,2))) AS DesvStdVentas
FROM Sales.SalesOrderHeader AS soh
JOIN Sales.SalesTerritory AS st
    ON st.TerritoryID = soh.TerritoryID
GROUP BY
    st.Name,
    YEAR(soh.OrderDate)
HAVING
    COUNT(*) > 5
    AND SUM(soh.TotalDue) > 1000000
ORDER BY
    VentasTotales DESC;
```

142 %

Results | Messages

	Territorio	Año	NumOrdenes	VentasTotales	DesvStdVentas
1	Southwest	2013	2725	10239209.3403	13470.4190076418
2	Southwest	2012	777	9329154.3425	23693.0433086733
3	Canada	2013	1884	7010449.6994	13359.6079162581
4	Northwest	2013	2053	6759500.6713	12654.187349953
5	Canada	2012	460	6599971.0217	24167.4811154
6	Northwest	2012	510	5325813.0562	20150.9168435972
7	Australia	2013	3015	4702404.0504	3719.04539329966
8	Southwest	2014	2383	4437517.8076	7343.90444035755
9	France	2013	1273	4271019.2663	12923.6955837112
10	United Kingdom	2013	1528	4068178.6672	9889.93325606799
11	Central	2013	151	3374336.2992	29231.3556892111
12	Northwest	2014	1807	3355402.8175	8268.17397338434
13	Southeast	2012	167	3344683.6085	26445.9087300123
14	Central	2012	130	3334867.9788	29430.5757523319
15	Northeast	2012	117	3272239.7992	28447.2046291298
16	Southwest	2011	339	3144713.0989	15928.88792577
17	Australia	2014	2473	3071053.8419	3163.59210482511

En esta versión se agregó la función STDEV() para calcular la desviación estándar de las ventas dentro de cada territorio y año.

Esto permite ver qué tanto varían los montos de las órdenes en cada grupo, además de mantener los mismos filtros y el mismo ordenamiento que en la consulta anterior.

Ejercico 4

El propósito de esta consulta es encontrar a los vendedores que han vendido TODOS los productos pertenecientes a la categoría "Bikes". Para lograr esto y mostrar nombres reales, se integró el análisis de múltiples tablas solicitadas en la práctica, específicamente: Sales.SalesOrderHeader, Sales.SalesOrderDetail, Production.Product, HumanResources.Employee y Person.Person.

2. Lógica Implementada: La "División Relacional" *En el lenguaje SQL no existe un comando nativo directo para solicitar "los elementos A que tengan absolutamente todos los elementos B". Para resolver este problema, aplicamos un principio de álgebra de bases de datos conocido como* ***División Relacional****.*

La estrategia matemática consiste en comparar dos conteos:

- ***Conteo del Catálogo (Total):*** *Se calcula cuántos productos únicos existen en total dentro de la categoría "Bikes".*
- ***Conteo del Vendedor (Ventas Únicas):*** *Se calcula cuántas bicicletas diferentes ha vendido cada empleado de forma individual.*
- ***Condición:*** *Si los productos distintos vendidos por un empleado son exactamente iguales en cantidad al total de productos en el catálogo, podemos asegurar que dicho empleado ha vendido todos y cada uno de los modelos.*

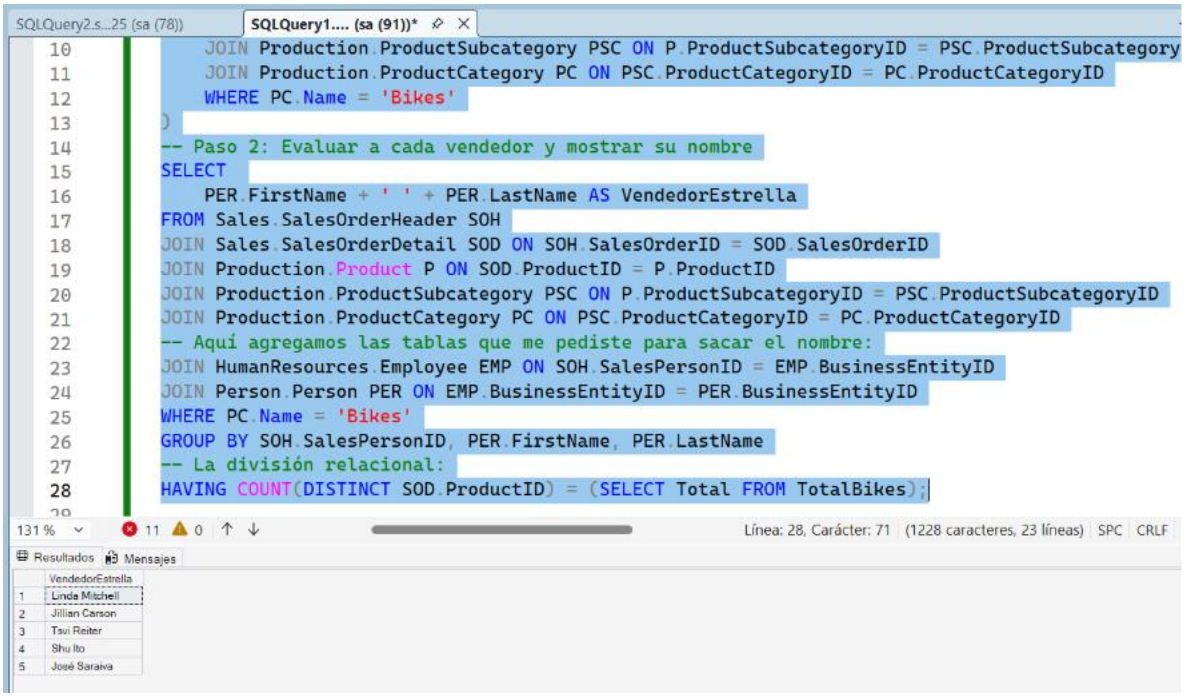
3. Explicación del Código (Flujo de Ejecución)

- ***Paso 1 (CTE TotalBikes):*** *Para optimizar el rendimiento, primero aislamos el cálculo del "Conteo del Catálogo" utilizando una Expresión de Tabla Común (CTE). Se une la tabla Production.Product con sus respectivas subcategorías y categorías, filtrando por el nombre 'Bikes', y obtenemos un único número almacenado en el alias Total.*
- ***Paso 2 (Recopilación y Cruce de Datos):*** *En la consulta principal, partimos del historial de ventas (Sales.SalesOrderHeader y Sales.SalesOrderDetail) para saber qué se vendió. Cruzamos con las tablas de Recursos Humanos y Personas (HumanResources.Employee y Person.Person) para transformar el ID numérico del vendedor en un nombre y apellido legible (VendedorEstrella).*
- ***Paso 3 (Agrupación):*** *Utilizamos GROUP BY sobre el ID y el nombre del vendedor para agrupar todas sus ventas en una sola "cubeta" por persona.*
- ***Paso 4 (Aplicación de la División Relacional):*** *Finalmente, la cláusula HAVING aplica el filtro definitivo post-agrupación. A través de la*

función COUNT(DISTINCT SOD.ProductID), contamos la variedad de bicicletas vendidas por el grupo y la igualamos a la subconsulta (SELECT Total FROM TotalBikes). Quien no alcance ese número exacto, es descartado del reporte.

- -- Paso 1: Contar el total de productos que son 'Bikes'
- WITH TotalBikes AS (
- SELECT COUNT(P.ProductID) AS Total
- FROM Production.Product P
- JOIN Production.ProductSubcategory PSC ON P.ProductSubcategoryID = PSC.ProductSubcategoryID
- JOIN Production.ProductCategory PC ON PSC.ProductCategoryID = PC.ProductCategoryID
- WHERE PC.Name = 'Bikes'
-)
- -- Paso 2: Evaluar a cada vendedor y mostrar su nombre
- SELECT
- PER.FirstName + ' ' + PER.LastName AS VendedorEstrella
- FROM Sales.SalesOrderHeader SOH
- JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID = SOD.SalesOrderID
- JOIN Production.Product P ON SOD.ProductID = P.ProductID
- JOIN Production.ProductSubcategory PSC ON P.ProductSubcategoryID = PSC.ProductSubcategoryID
- JOIN Production.ProductCategory PC ON PSC.ProductCategoryID = PC.ProductCategoryID
- -- Aquí agregamos las tablas que me pediste para sacar el nombre:
- JOIN HumanResources.Employee EMP ON SOH.SalesPersonID = EMP.BusinessEntityID
- JOIN Person.Person PER ON EMP.BusinessEntityID = PER.BusinessEntityID

- WHERE PC.Name = 'Bikes'
- GROUP BY SOH.SalesPersonID, PER.FirstName, PER.LastName
- -- La división relacional:
- HAVING COUNT(DISTINCT SOD.ProductID) = (SELECT Total FROM TotalBikes);



Parte 2

1. Objetivo de la Variante *El objetivo de esta variante es adaptar la lógica de la "División Relacional" previamente construida para buscar a los vendedores que hayan vendido el 100% de los productos de la categoría "Clothing", cuyo identificador en la base de datos es el ID 4.*

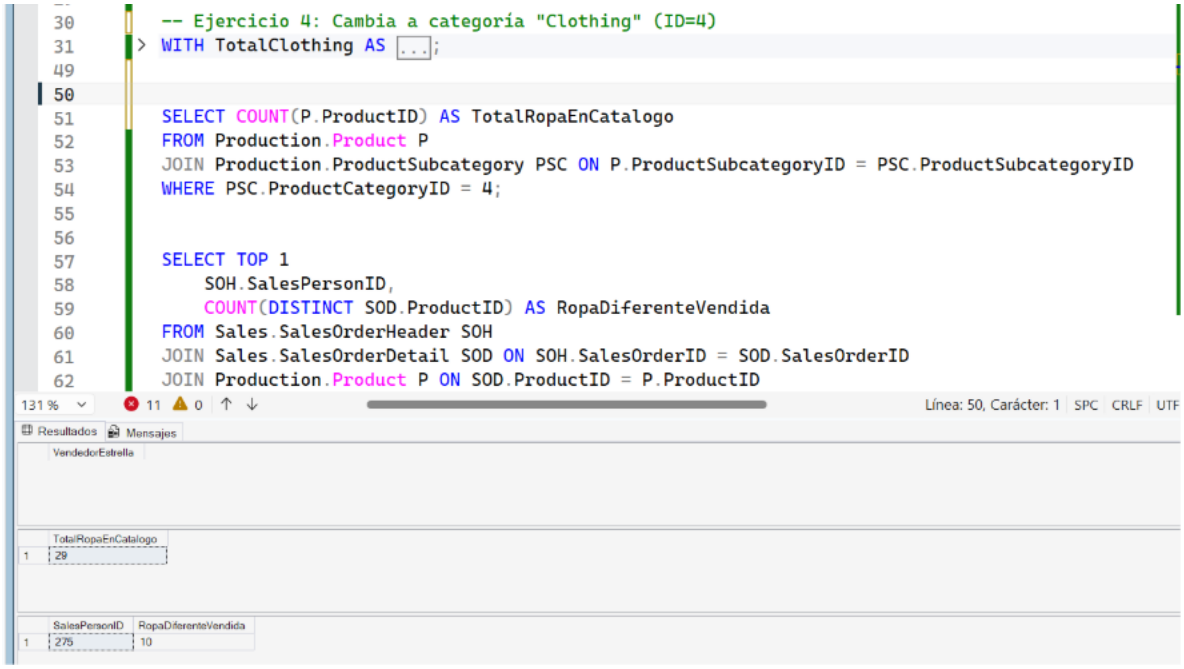
2. Ajustes Técnicos y Optimización *Para esta adaptación, la estructura general (la Expresión de Tabla Común TotalClothing y la validación matemática en la cláusula HAVING) se mantuvo idéntica. Sin embargo, se aplicó una optimización en el rendimiento:*

- *Al contar con el ID numérico de la categoría (ProductCategoryID = 4), se eliminó la necesidad de hacer un JOIN adicional con la tabla Production.ProductCategory. La validación numérica se realizó directamente desde la*

tabla Production.ProductSubcategory, haciendo que la ejecución de la consulta sea más directa y consuma menos recursos.

3. Análisis de Resultados (Validación del Conjunto Vacío) *Al ejecutar la consulta principal de esta variante, el motor de SQL Server devuelve un* ***conjunto de resultados vacío (0 filas)****. Lejos de ser un error de sintaxis, esto representa la respuesta correcta al escenario planteado por la base de datos AdventureWorks. Para sustentar matemáticamente este resultado y demostrar la rigurosidad de la División Relacional, se ejecutaron dos consultas de validación adicionales:*

- ***Validación A (Total de Ropa en Catálogo):*** *Mediante la consulta SELECT COUNT(...) AS TotalRopaEnCatalogo, se determinó el universo total de productos únicos que pertenecen a la categoría 4.*
- ***Validación B (Máximo de Ropa Vendida por Empleado):*** *A través de la consulta con SELECT TOP 1 ... ORDER BY RopaDiferenteVendida DESC, se localizó al vendedor que mayor variedad de artículos de ropa ha logrado colocar en el mercado.*



Para este ejercicio, se requirió simular un entorno distribuido donde el esquema SALES reside en el Servidor A (local) y el esquema PRODUCTION en el Servidor B (remoto). Para resolverlo en un entorno real, configuramos un Servidor Vinculado (Linked Server) denominado SV_SELF, el cual apunta a una instancia remota de SQL Server a través de una conexión de red celular externa.

Debido a que la conexión entre los nodos se realiza por una red celular, la latencia y el ancho de banda son factores críticos. Si intentáramos cruzar la tabla de detalles de ventas directamente con el catálogo remoto, la consulta saturaría la red y probablemente fallaría por *Timeout*.

Para evitar esto, estructuramos la consulta utilizando Expresiones de Tabla Comunes (CTEs):

1. **Agrupación Local:** Primero, sumamos las cantidades vendidas (OrderQty) agrupando por ProductID únicamente con los datos locales del Servidor A.
2. **Consulta Distribuida:** Posteriormente, utilizamos la nomenclatura de 4 partes ([SV_SELF].[AdventureWorks].[Production].[Product]) para llamar exclusivamente a los catálogos del Servidor B.
3. **Cruce y Ranking:** Finalmente, unimos el resumen local con el catálogo remoto y aplicamos la función de ventana ROW_NUMBER() OVER(PARTITION BY...) para aislar al producto #1 en ventas de cada categoría.

Esta arquitectura de consulta garantiza que el cruce de datos masivos se procese localmente, enviando a través del Servidor Vinculado únicamente el conjunto de resultados estrictamente necesario, optimizando así el rendimiento en un entorno de bases de datos distribuidas.