INSTITUTO POLITÉCTICO NACIONAL

UPIITA - Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas IPN

Base de Datos Distribuidas

practica 1 REPASO DE CONSULTAS SQL Y USO DE SERVIDORES VINCULADOS

Marco González Luna 2024640044

Olvera Valdivia Cristobal 2024640014

Grupo

3TM2

Profesor

Carlos De La Cruz Sosa

upiita-ipn

PRACTICA 1. BASE DE DATOS DISTRIBUIDAS 2026/2 3TM2 UPIITA – IPN

REPASO DE CONSULTAS SQL Y USO DE SERVIDORES VINCULADOS

Esta práctica utiliza la base de datos AdventureWorks de SQL Server para aplicar consultas multitablas, subconsultas, GROUP BY y conceptos avanzados. Los ejercicios requieren análisis de múltiples tablas como Sales.SalesOrderHeader, Sales.SalesOrderDetail, Production.Product, HumanResources.Employee y Person.Person.

Requisitos:

- Instala AdventureWorks desde el sitio oficial de Microsoft (versión OLTP).
- Usa SQL Server Management Studio (SSMS) conectado a la base de datos.
- Para cada consulta añade en el reporte el script de la consulta con discusión de la solución
- y captura de las 10 primeras filas del resultado.
- Subir al repositorio de cada alumno, reporte escrito de la práctica en el archivo
- practica1_equipoX.pdf y practica1_equipoX.SQL con scripts de las consultas

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

```sql
/*Ejercicio 1. Encuentra los 10 productos más vendidos en 2014,
                mostrando nombre del producto,
                cantidad total vendida y nombre del cliente.*/

WITH PROD2014 as (
select soh.SalesOrderID, sod.ProductID, sod.OrderQty, soh.CustomerID
from sales.SalesOrderHeader soh join sales.SalesOrderDetail sod
on soh.SalesOrderID = sod.SalesOrderID
where year(OrderDate) = '2014' ),

Top10 as (
select name, t.ProductID, cant
from Production.product p
    join
            (select top 10 productid , sum(p.OrderQty) cant
              from PROD2014 as p
              group by p.productid
              order by cant desc) as T
        on p.ProductID = t.ProductID),

ClienteConProducto as (
select t2.ProductID, t2.Name, p2.CustomerID, p2.OrderQty, t2.cant from
        PROD2014 as p2 JOIN Top10 as t2
        ON p2.ProductID = t2.ProductID
        )

Select ccp.ProductID, ccp.Name, ccp.CustomerID, ccp.OrderQty,
       per.FirstName, per.LastName, ccp.cant
    from ClienteConProducto ccp JOIN
        Sales.Customer cus
        ON cus.CustomerID = ccp.CustomerID

        JOIN Person.Person per
        ON cus.PersonID = per.BusinessEntityID
        order by ccp.cant desc;
```

Results | Messages

	ProductID	Name	CustomerID	OrderQty	FirstName	LastName	cant
1	870	Water Bottle - 30 oz.	15715	1	Isabella	Adams	2902
2	870	Water Bottle - 30 oz.	26073	1	Julia	Adams	2902
3	870	Water Bottle - 30 oz.	11869	1	Kaitlyn	Adams	2902
4	870	Water Bottle - 30 oz.	11869	1	Kaitlyn	Adams	2902
5	870	Water Bottle - 30 oz.	12663	1	Morgan	Adams	2902
6	870	Water Bottle - 30 oz.	16647	1	Anna	Alexander	2902
7	870	Water Bottle - 30 oz.	26281	1	Ian	Alexander	2902
8	870	Water Bottle - 30 oz.	17151	1	Jasmine	Alexander	2902
9	870	Water Bottle - 30 oz.	16102	1	Marcus	Alexander	2902
10	870	Water Bottle - 30 oz.	21837	1	Megan	Alexander	2902
11	870	Water Bottle - 30 oz.	14025	1	Robert	Alexander	2902
12	870	Water Bottle - 30 oz.	29240	1	Madeline	Allen	2902
13	870	Water Bottle - 30 oz.	23683	1	Dana	Alonso	2902
14	870	Water Bottle - 30 oz.	16686	1	Kristine	Alonso	2902
15	870	Water Bottle - 30 oz.	25009	1	Natasha	Alonso	2902
16	870	Water Bottle - 30 oz.	22692	1	Paula	Alonso	2902
17	870	Water Bottle - 30 oz.	24166	1	Renee	Alonso	2902
18	870	Water Bottle - 30 oz.	19464	1	Tyrone	Alonso	2902
19	870	Water Bottle - 30 oz.	23358	1	Alberto	Alvarez	2902
20	870	Water Bottle - 30 oz.	14235	1	Larry	Alvarez	2902
21	870	Water Bottle - 30 oz.	18221	1	Marc	Alvarez	2902
22	870	Water Bottle - 30 oz.	11986	1	Max	Alvarez	2902

Query executed successfully. | MARK\SQLEXPRESS (16.0 RTM) | sa (65) | BDDAdventureWorks2022 | 00:00:00 | 12,051 rows

1. Una vez resuelta la consulta: agrega el precio unitario promedio (AVG(UnitPrice)) y filtra solo productos con ListPrice > 1000.

```
/*1.Una vez resuelta la consulta:
agrega el precio unitario promedio (AVG(UnitPrice))
y filtra solo productos con ListPrice > 1000.*/

WITH PROD2014 as (
select soh.SalesOrderID, sod.ProductID, sod.OrderQty, soh.CustomerID, sod.UnitPrice, pr.ListPrice
from sales.SalesOrderHeader soh
        join sales.SalesOrderDetail sod
        on soh.SalesOrderID = sod.SalesOrderID
        join Production.Product pr
        on pr.ProductID = sod.ProductID
        where year(OrderDate) = '2014' ),
Top10 as (
select prod.name, t.ProductID, cant, precioUProm, prod.ListPrice
from Production.product prod
    join
            (select top 10 productid , sum(p.OrderQty) cant, avg(p.UnitPrice) precioUProm
              from PROD2014 as p
              where p.ListPrice>1000
              group by p.productid
              order by cant desc) as T
        on prod.ProductID = t.ProductID),

ClienteConProducto as (
select t2.ProductID, t2.Name, p2.CustomerID, p2.OrderQty, t2.cant, t2.precioUProm, t2.ListPrice from
        PROD2014 as p2 JOIN Top10 as t2
        ON p2.ProductID = t2.ProductID)

Select ccp.ProductID, ccp.Name, ccp.CustomerID, ccp.OrderQty,
       per.FirstName, per.LastName, ccp.cant, ccp.precioUProm, ccp.ListPrice
    from ClienteConProducto ccp JOIN
        Sales.Customer cus
        ON cus.CustomerID = ccp.CustomerID
        JOIN Person.Person per
        ON cus.PersonID = per.BusinessEntityID
        order by ccp.cant desc;
```

Results | Messages

	ProductID	Name	CustomerID	OrderQty	FirstName	LastName	cant	precioUProm	ListPrice
1	782	Mountain-200 Black, 38	29492	7	Jay	Adams	619	1981.7872	2294.99
2	782	Mountain-200 Black, 38	12950	1	Mary	Adams	619	1981.7872	2294.99
3	782	Mountain-200 Black, 38	29500	1	Anna	Albright	619	1981.7872	2294.99
4	782	Mountain-200 Black, 38	17423	1	Arianna	Alexander	619	1981.7872	2294.99
5	782	Mountain-200 Black, 38	12915	1	Kaitlyn	Alexander	619	1981.7872	2294.99
6	782	Mountain-200 Black, 38	18894	1	Eric	Allen	619	1981.7872	2294.99
7	782	Mountain-200 Black, 38	15588	1	Kristy	Alvarez	619	1981.7872	2294.99
8	782	Mountain-200 Black, 38	14929	1	Peter	Anand	619	1981.7872	2294.99
9	782	Mountain-200 Black, 38	19032	1	Katrina	Andersen	619	1981.7872	2294.99
10	782	Mountain-200 Black, 38	12885	1	Jonathan	Anderson	619	1981.7872	2294.99
11	782	Mountain-200 Black, 38	29522	2	Thomas	Armstrong	619	1981.7872	2294.99
12	782	Mountain-200 Black, 38	29523	2	John	Arthur	619	1981.7872	2294.99
13	782	Mountain-200 Black, 38	19030	1	Marco	Arun	619	1981.7872	2294.99
14	782	Mountain-200 Black, 38	16641	1	Julia	Bailey	619	1981.7872	2294.99
15	782	Mountain-200 Black, 38	12883	1	Stephanie	Bailey	619	1981.7872	2294.99
16	782	Mountain-200 Black, 38	12849	1	Adam	Baker	619	1981.7872	2294.99
17	782	Mountain-200 Black, 38	29540	2	Adam	Barr	619	1981.7872	2294.99
18	782	Mountain-200 Black, 38	29544	3	Shaun	Beasley	619	1981.7872	2294.99

Query executed s... | MARK\SQLEXPRESS (16.0 RTM) | sa (65) | BDDAdventureWorks2022 | 00:00:00 | 2,595 rows

2. Documenta la solución inicial y solución con las variantes solicitadas.

La diferencia entre la consulta base y la segunda consulta a realizar es que en la primera CTE que se realizó, se debió añadir un join con la tabla producto para así poder añadir el parámetro de ListPrice pues, de otra manera, no hubiera funcionado la segunda CTE Top10 al no poder filtrar los productos de manera correcta que superaran el ListPrice>1000, arrojando un conjunto vacío. Las demás operaciones se quedan iguales, sólo se debe añadir ListPrice en la CTE PROD2014 y filtrar por él en Top10.

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

1. Requisito adicional: aplicar subconsultas.

```sql
/*Ejercicio 2: Lista los empleados que han vendido más que el
promedio de ventas por empleado en el territorio 'Northwest'.*/
-- 1.Requisito adicional: aplicar subconsultas.

SELECT
    VentasEmp.SalesPersonID,
    VentasEmp.FirstName,
    VentasEmp.LastName,
    VentasEmp.ventas
FROM (
    SELECT  soh.SalesPersonID, pp.FirstName, pp.LastName , sum(sod.OrderQty * sod.UnitPrice) ventas
        FROM Sales.SalesOrderHeader soh
        JOIN Sales.SalesOrderDetail sod
        ON soh.SalesOrderID = sod.SalesOrderID
        JOIN Sales.SalesPerson ssp
        ON ssp.BusinessEntityID = soh.SalesPersonID and ssp.TerritoryID =1
        JOIN Person.Person pp
        ON soh.SalesPersonID = pp.BusinessEntityID
        where soh.TerritoryID = 1
        and soh.SalesPersonID is not null
        group by soh.SalesPersonID, pp.FirstName, pp.LastName) AS VentasEmp

WHERE VentasEmp.ventas > (
        SELECT AVG(PromedioVentas.ventas_totales)
    FROM (
        SELECT  soh.SalesPersonID, sum(sod.OrderQty * sod.UnitPrice) ventas_totales
        FROM Sales.SalesOrderHeader soh
        JOIN Sales.SalesOrderDetail sod
        ON soh.SalesOrderID = sod.SalesOrderID
        JOIN Sales.SalesPerson ssp
        ON ssp.BusinessEntityID = soh.SalesPersonID and ssp.TerritoryID =1
        JOIN Person.Person pp
        ON soh.SalesPersonID = pp.BusinessEntityID
        where soh.TerritoryID = 1
        and soh.SalesPersonID is not null
        group by soh.SalesPersonID, pp.FirstName, pp.LastName) AS PromedioVentas
);
```

	SalesPersonID	FirstName	LastName	ventas
1	280	Pamela	Ansman-Wolfe	3339713.6269
2	283	David	Campbell	3756861.6175

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

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

WITH salesEmp AS(
SELECT  soh.SalesPersonID, pp.FirstName, pp.LastName , sum(sod.OrderQty * sod.UnitPrice) ventas
        FROM Sales.SalesOrderHeader soh
        JOIN Sales.SalesOrderDetail sod
        ON soh.SalesOrderID = sod.SalesOrderID
        JOIN Sales.SalesPerson ssp
        ON ssp.BusinessEntityID = soh.SalesPersonID and ssp.TerritoryID =1
        JOIN Person.Person pp
        ON soh.SalesPersonID = pp.BusinessEntityID
        where soh.TerritoryID = 1
        and soh.SalesPersonID is not null
        group by soh.SalesPersonID, pp.FirstName, pp.LastName)


SELECT *
        FROM salesEmp se1 where se1.ventas > (SELECT avg(se.ventas) avgEmp FROM salesEmp se);
```

Results | Messages

	SalesPersonID	FirstName	LastName	ventas	avgEmp
1	280	Pamela	Ansman-Wolfe	3339713.6269	3141504.5522
2	283	David	Campbell	3756861.6175	3141504.5522

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

En este caso, las consultas son diferentes ya que, en este caso, la CTE sales Emp sí ahorra procesamiento a diferencia de realizar el query con subconsultas. En la CTE se obtienen los empleados del área de Northwest y las ventas que han hecho, luego se obtiene el promedio de ventas en una subconsulta escalar y se filtra la CTE base con el valor del promedio de ventas de los empleados.

Esto es diferente al query con la subconsulta, ya que se tuvo que ejecutar la misma instrucción 2 veces, una para obtener a los empleados y otra para el promedio, repitiendo una consulta que nos daría una CTE de manera más sencilla y entendible.

Ejercicio 3: Calcula ventas totales por territorio y año, mostrando solo aquellos con más de 5 órdenes y ventas > $1,000,000, ordenado por ventas descendente.

```
select
    t.name as Territorio,
    year(h.OrderDate) as Año,
    count(h.SalesOrderID) as NumeroOrdenes,
    sum(h.TotalDue) as VentasTotales

    from Sales.SalesOrderHeader h
        JOIN
    Sales.SalesTerritory t

    on h.TerritoryID = t.TerritoryID

        group by t.name, year(h.OrderDate)
        having count(h.SalesOrderID) > 5
        and sum(h.TotalDue) > 1000000
        order by sum(h.TotalDue) desc;
```

	Territorio	Año	NumeroOrdenes	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
18	Northeast	2013	138	2965567.0284
19	Germany	2013	1235	2869491.9712
20	Southeast	2013	180	2705730.9695
21	Canada	2014	1574	2681602.5941
22	Northwest	2011	224	2620943.826
23	Australia	2012	892	2347885.4611
24	United Kingdom	2014	1251	2335108.8971
25	Canada	2011	149	2106905.8728
26	France	2014	1039	1868973.7989
27	Southeast	2011	70	1847744.578
28	United Kingdom	2012	323	1769769.2149
29	France	2012	290	1743487.6538
30	Germany	2014	1058	1729718.5224
31	Australia	2011	463	1693032.7418
32	Central	2011	50	1126645.7497
33	Central	2014	54	1077449.2196

1. Una vez resuelta la consulta agrega desviación estándar de ventas

```
select
    t.name as Territorio,
    year(h.OrderDate) as Año,
    count(h.SalesOrderID) as NumeroOrdenes,
    sum(h.TotalDue) as VentasTotales,
    stdev(h.TotalDue) as DesviacionEstandarVentas

from Sales.SalesOrderHeader h
     join Sales.SalesTerritory t
     on h.TerritoryID = t.TerritoryID

            group by t.name, YEAR(h.OrderDate)
            having count(h.SalesOrderID) > 5
               and sum(h.TotalDue) > 1000000
            order by sum(h.TotalDue) desc;
```

	Territorio	Año	NumeroOrdenes	VentasTotales	DesviacionEstandarVentas
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
15	Northeast	2012	117	3272239.7992	28447.2048879228
16	Southwest	2011	339	3144713.0989	15928.8880820697
17	Australia	2014	2473	3071053.8419	3163.5920049902
18	Northeast	2013	138	2965567.0284	26359.6210570197
19	Germany	2013	1235	2869491.9712	8518.63607528887
20	Southeast	2013	180	2705730.9695	21131.9016426056
21	Canada	2014	1574	2681602.5941	8025.74655221896
22	Northwest	2011	224	2620943.826	20029.8439978845
23	Australia	2012	892	2347885.4611	1032.13497417915
24	United Ki...	2014	1251	2335108.8971	7517.65907120863
25	Canada	2011	149	2106905.8728	19763.2093198307
26	France	2014	1039	1868973.7989	7854.50892990388
27	Southeast	2011	70	1847744.578	29647.0788043396
28	United Ki...	2012	323	1769769.2149	13168.5936628185
29	France	2012	290	1743487.6538	16311.3089590187
30	Germany	2014	1058	1729718.5224	5849.0562018686
31	Australia	2011	463	1693032.7418	843.082115178095
32	Central	2011	50	1126645.7497	30731.5849951589
33	Central	2014	54	1077449.2196	21978.3749762531

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

La consulta base calcula las ventas totales por territorio y año, filtrando mediante la cláusula HAVING para mostrar solo aquellos registros con más de 5 órdenes y ventas superiores a $1,000,000.

La diferencia entre la consulta original y la variante solicitada cambia únicamente en agregar la desviación estándar de las ventas. Para lograr ocupo la función de agregación STDEV dentro de la instrucción SELECT. Al no requerir nuevos filtros ni nuevas tablas, las cláusulas de unión JOIN y de agrupación GROUP BY permanecen intactas, permitiéndonos observar qué tan dispersos están los montos de ventas dentro de esos mismos territorios y años sin afectar el rendimiento de la consulta base

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

```
WITH VENDEDORES AS(
select  soh.SalesPersonID idVendedor, count(distinct sod.ProductID) CANT
        from Sales.SalesOrderHeader soh
        JOIN Sales.SalesOrderDetail sod
        ON soh.SalesOrderID = sod.SalesOrderID
        JOIN Production.Product pp
        ON sod.ProductID = pp.ProductID
        JOIN Production.ProductSubcategory pps
        ON pp.ProductSubcategoryID = pps.ProductSubcategoryID
        JOIN Production.ProductCategory ppc
        ON pps.ProductCategoryID = ppc.ProductCategoryID
        where soh.SalesPersonID is not null
        AND ppc.ProductCategoryID = 1
        GROUP BY soh.SalesPersonID

        HAVING count(distinct sod.ProductID) = (

select count(pp.ProductID) CantProdBike
        FROM Production.Product pp
        JOIN Production.ProductSubcategory pps
        ON pp.ProductSubcategoryID = pps.ProductSubcategoryID
        JOIN Production.ProductCategory ppc
        ON pps.ProductCategoryID = ppc.ProductCategoryID
        WHERE ppc.ProductCategoryID = 1
        GROUP BY ppc.Name))

        SELECT  v.idVendedor, pp.FirstName, pp.LastName, v.CANT
            FROM VENDEDORES v
            JOIN
            Person.Person pp
            on v.idVendedor = pp.BusinessEntityID;
```

	idVendedor	FirstName	LastName	CANT
1	276	Linda	Mitchell	97
2	277	Jillian	Carson	97
3	279	Tsvi	Reiter	97
4	281	Shu	Ito	97
5	282	José	Saraiva	97

1. Cambia a categoría "Clothing" (ID=4).

```
WITH VENDEDORES AS(
select  soh.SalesPersonID idVendedor, count(distinct sod.ProductID) CANT
        from Sales.SalesOrderHeader soh
        JOIN Sales.SalesOrderDetail sod
        ON soh.SalesOrderID = sod.SalesOrderID
        JOIN Production.Product pp
        ON sod.ProductID = pp.ProductID
        JOIN Production.ProductSubcategory pps
        ON pp.ProductSubcategoryID = pps.ProductSubcategoryID
        JOIN Production.ProductCategory ppc
        ON pps.ProductCategoryID = ppc.ProductCategoryID
        where soh.SalesPersonID is not null
        AND ppc.ProductCategoryID = 3
        GROUP BY soh.SalesPersonID

        HAVING count(distinct sod.ProductID) = (

select count(pp.ProductID) CantProdBike
        FROM Production.Product pp
        JOIN Production.ProductSubcategory pps
        ON pp.ProductSubcategoryID = pps.ProductSubcategoryID
        JOIN Production.ProductCategory ppc
        ON pps.ProductCategoryID = ppc.ProductCategoryID
        WHERE ppc.ProductCategoryID = 1
        GROUP BY ppc.Name))

        SELECT  v.idVendedor, pp.FirstName, pp.LastName, v.CANT
            FROM VENDEDORES v
            JOIN
            Person.Person pp
            on v.idVendedor = pp.BusinessEntityID;
```

idVendedor	FirstName	LastName	CANT

2. Cuenta cuántos productos por categoría maneja cada vendedor.

```
WITH VendCat as(

select  soh.SalesPersonID, ppc.ProductCategoryID, count(distinct pp.ProductID) prodXCat

        from Sales.SalesOrderHeader soh
        JOIN Sales.SalesOrderDetail sod
        ON soh.SalesOrderID = sod.SalesOrderID
        JOIN Production.Product pp
        ON sod.ProductID = pp.ProductID
        JOIN Production.ProductSubcategory pps
        ON pp.ProductSubcategoryID = pps.ProductSubcategoryID
        JOIN Production.ProductCategory ppc
        ON pps.ProductCategoryID = ppc.ProductCategoryID
        where soh.SalesPersonID is not null
        group by  soh.SalesPersonID, ppc.ProductCategoryID
        )

        select vc.SalesPersonID,pp.FirstName, pp.LastName, vc.ProductCategoryID, vc.prodXCat
            from VendCat vc
            JOIN
            Person.Person pp
            ON vc.SalesPersonID = pp.BusinessEntityID
            order by  vc.SalesPersonID, vc.ProductCategoryID;
```

	SalesPersonID	FirstName	LastName	ProductCategoryID	prodXCat
35	282	José	Saraiva	3	32
36	282	José	Saraiva	4	10
37	283	David	Campbell	1	95
38	283	David	Campbell	2	104
39	283	David	Campbell	3	31
40	283	David	Campbell	4	10
41	284	Tete	Mensa-A...	1	78
42	284	Tete	Mensa-A...	2	91
43	284	Tete	Mensa-A...	3	29
44	284	Tete	Mensa-A...	4	9
45	285	Syed	Abbas	1	24
46	285	Syed	Abbas	2	22
47	285	Syed	Abbas	3	14
48	285	Syed	Abbas	4	8
49	286	Lynn	Tsoflias	1	37
50	286	Lynn	Tsoflias	2	57
51	286	Lynn	Tsoflias	3	15
52	286	Lynn	Tsoflias	4	8
53	287	Amy	Alberts	1	75
54	287	Amy	Alberts	2	82
55	287	Amy	Alberts	3	29
56	287	Amy	Alberts	4	10
57	288	Rachel	Valdez	1	68
58	288	Rachel	Valdez	2	80
59	288	Rachel	Valdez	3	25
60	288	Rachel	Valdez	4	9
61	289	Jae	Pak	1	79
62	289	Jae	Pak	2	102
63	289	Jae	Pak	3	30
64	289	Jae	Pak	4	10
65	290	Ranjit	Varkey C...	1	79
66	290	Ranjit	Varkey C...	2	101
67	290	Ranjit	Varkey C...	3	29
68	290	Ranjit	Varkey C...	4	10

3. Documenta la solución inicial y solución con las variantes solicitadas.

La consulta original tiene como objetivo encontrar a los vendedores que han vendido absolutamente TODOS los productos de la categoría "Bikes", utilizando un HAVING con una subconsulta para igualar el conteo de productos.

Variante de cambio de categoría: Se modifico la búsqueda para la categoría "Clothing" (ID=4). En este caso, se sustituye el identificador de la categoría en las cláusulas WHERE de la consulta principal y la subconsulta para la nueva categoría que nos pidió, sea entonces “Clothing”.

Variante de conteo de productos: Posteriormente, la lógica se modifica para contar cuántos productos por categoría maneja cada vendedor. A diferencia de la consulta original se elimina la restricción de la cláusula HAVING. En su lugar, se implementa una nueva CTE “VendCat” que agrupa directamente por el identificador del empleado y de la categoría, aplicando un COUNT(DISTINCT pp.ProductID). Esto simplifica la consulta y nos arroja un desglose general del portafolio de ventas de cada empleado, en lugar de un filtro excluyente.

Ejercicio 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.

```sql
WITH VentasPorCategoria AS (
    SELECT
        ppc.ProductCategoryID,
        ppc.Name,
        pp.ProductID,
        SUM(sod.OrderQty) AS TotalVendido
    FROM Sales.SalesOrderDetail sod -- Local (Servidor A)
    -- A partir de aquí, todo es remoto (Servidor B)
    JOIN SV_CRIS.AdventureWorks2022_2.Production.Product pp
        ON sod.ProductID = pp.ProductID
    JOIN SV_CRIS.AdventureWorks2022_2.Production.[ProductSubcategory] pps
        ON pp.ProductSubcategoryID = pps.ProductSubcategoryID
    JOIN SV_CRIS.AdventureWorks2022_2.Production.[ProductCategory] ppc
        ON pps.ProductCategoryID = ppc.ProductCategoryID
    GROUP BY ppc.ProductCategoryID, ppc.Name, pp.ProductID
)

SELECT * FROM VentasPorCategoria V1
WHERE TotalVendido = (
    SELECT MAX(TotalVendido)
    FROM VentasPorCategoria V2
    WHERE V2.ProductCategoryID = V1.ProductCategoryID
)
ORDER BY V1.ProductCategoryID;
```

```sql
EXEC sp_addlinkedserver
    @server = 'SV_CRIS',
    @srvproduct = 'SQLServer', -- opcional
    @provider = 'SQLOLEDB',
    @datasrc = '10.207.136.92,1433',
    @provstr = 'TrustServerCertificate=YES';

EXEC sp_addlinkedsrvlogin
    @rmtsrvname = 'SV_CRIS',
    @useself = 'false', -- valor false si se usarán credenciales distintas
    @rmtuser = 'sa',
    @rmtpassword = 'CristobalKari04*';

EXEC sp_testlinkedserver SV_CRIS;

select * from SV_CRIS.Escuela.Escuela.Alumno

INSERT INTO SV_CRIS.Escuela.Escuela.Alumno VALUES
('2020630021','Cristobal Olvera Valdivia',6,8.5,1,1);
```

	ProductCategoryID	Name	ProductID	TotalVendido
1	1	Bikes	782	2977
2	2	Components	738	1581
3	3	Clothing	712	8311
4	4	Accessories	870	6815

A diferencia de una consulta normal, esta consulta tablas del esquema Production de manera remota, esto aumenta brevemente el tiempo de realización de la consulta, pues debe pedir datos que no están en el equipo y se hacen joins con varias tablas que locales para generar la CTE. La cantidad de tiempo aumentaría si las tablas fueran más grandes o no estuvieran indexados los datos con los que se realiza la consulta.