



INSTITUTO POLITÉCNICO NACIONAL



**UNIDAD PROFESIONAL INTERDISCIPLINARIA EN
INGENIERÍA Y TECNOLOGÍAS AVANZADAS**

UPIITA

PROFESOR: Carlos De La Cruz Sosa

ASIGNATURA: Bases de Datos Distribuidas

GRUPO: 3TM3

Práctica II

EQUIPO 5:

- Bernal Aguilar Yuvia Abigail
- Contreras Jimenez Mariana Montserrat
- Medina Gómez Jimena Zarahí

Introducción:

Cuando enviamos una consulta en lenguaje SQL, el motor de base de datos elabora una estrategia detallada: un mapa de ruta interno conocido como plan de ejecución. Estos planes actúan como una hoja de ruta que guía al motor en cómo acceder a las tablas, aplicar filtros, realizar uniones y, finalmente, entregar el conjunto de resultados deseado.

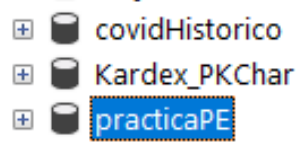
Al analizar estos planes, es posible comprender cómo se ejecuta realmente una consulta dentro del sistema. Esto permite identificar ineficiencias, optimizar el uso de recursos, detectar qué operaciones consumen más tiempo y aplicar mejoras clave en el rendimiento. Uno de los elementos más importantes en esta optimización son los índices.

Los índices están basados en estructuras que permiten localizar filas dentro de las tablas de forma más rápida y eficiente. Funcionan almacenando una copia ordenada de las columnas clave de una tabla, lo cual acelera significativamente el acceso a los datos más relevantes y mejora el rendimiento general de las consultas.

En conclusión, entender y aprovechar adecuadamente los planes de ejecución y los índices no solo permite optimizar el rendimiento de nuestras consultas, sino que también contribuye al diseño de bases de datos más extensas, escalables y eficientes a largo plazo.

Desarrollo:

1. Crear una base de datos con el nombre: practicaPE.



2. Copiar a la base de datos practicaPE las siguientes tablas de la base de datos AdventureWorks:

a) Sales.SalesOrderHeader

```
--- Tabla SalesOrderHeader
select * into order_header
from AdventureWorks2019.sales.SalesOrderHeader;
```

b) Sales.SalesOrderDetail

```
--- Tabla SalesOrderDetail
select * into order_detail
from AdventureWorks2019.sales.SalesOrderDetail;
```

c) Sales.Customer

```
-- Tabla Customer
select * into customer
from AdventureWorks2019.sales.Customer;
```

d) Sales.SalesTerritory

```
-- Tabla SalesTerritory
select * into SalesTerritory
from AdventureWorks2019.Sales.SalesTerritory;
```

e) Production.Product

```
-- Tabla Product
select * into products
from AdventureWorks2019.Production.Product;
```

f) Production.ProductCategory

```
-- Tabla ProductCategory
select * into ProductCategory
from AdventureWorks2019.Production.ProductCategory;
```

g) Production.ProductSubcategory

```
-- Tabla ProductSubCategory
select * into ProductSubCategory
from AdventureWorks2019.Production.ProductSubCategory;
```

h) Person.Person

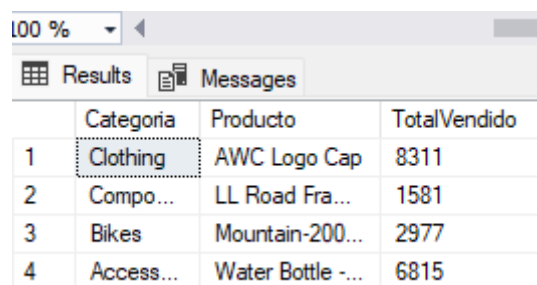
```
-- Tabla Person
--Lo hacemos de este modo ya que existen datos XML

CREATE TABLE person (
    BusinessEntityID int NOT NULL,
    PersonType nchar(2) NOT NULL,
    NameStyle bit NOT NULL,
    Title nvarchar(8),
    FirstName nvarchar(50) NOT NULL,
    MiddleName nvarchar(50),
    LastName nvarchar(50) NOT NULL,
    Suffix nvarchar(10),
    EmailPromotion int NOT NULL,
    AdditionalContactInfo xml,
    Demographics xml,
    rowguid uniqueidentifier NOT NULL,
    ModifiedDate datetime NOT NULL
);

INSERT INTO person
SELECT * FROM AdventureWorks2019.Person.Person;
```

3. Codificar las siguientes consultas:

a) Listar el producto más vendido de cada una de las categorías registradas en la base de datos.



	Categoria	Producto	TotalVendido
1	Clothing	AWC Logo Cap	8311
2	Compo...	LL Road Fra...	1581
3	Bikes	Mountain-200...	2977
4	Access...	Water Bottle -...	6815

b) Listar el nombre de los clientes con más ordenes por cada uno de los territorios registrados en la base de datos.

	Territorio	Cliente	CustomerID	Ordenes
1	Northwest	François Fierier	29497	12
2	Central	Kim Abercrombie	29486	12
3	Australia	Pilar Ackeman	29488	4
4	Southwest	Frances Adams	29489	12
5	Northeast	Michael Allen	29509	12
6	Australia	Sandra Altamirano	29512	4
7	Australia	Mae Anderson	29516	4
8	Northwest	Tom Johnston	29521	12
9	Southwest	Thomas Armstrong	29522	12
10	Southwest	John Arthur	29523	12
11	Australia	Phillip Bacalzo	29529	4
12	Central	Douglas Baldwin	29533	12
13	Australia	Brenda Barlow	29538	4
14	United Ki...	Christopher Beck	29546	8
15	Australia	Scot Bent	29556	4
16	Southwest	Michael Blythe	29570	12
17	Northwest	Gabriel Bocken...	29571	12
18	Northwest	Richard Bready	29580	12
19	Germany	David Brink	29586	8
20	United Ki...	John Brooks	29587	8

	Territorio	Cliente	CustomerID	Ordenes
20	United Ki...	John Brooks	29587	8
21	Northeast	Kevin Browne	29592	12
22	Australia	Bridget Browgett	29595	4
23	Australia	Nancy Buchanan	29602	4
24	Northwest	Lindsey Camacho	29617	12
25	United Ki...	Gustavo Camargo	29618	8
26	Australia	Richard Carey	29628	4
27	Southwest	Donna Carreras	29637	12
28	Southwest	Stacey Cereghino	29646	12
29	Australia	Sean Chai	29648	4
30	Central	Lee Chapla	29651	12
31	Australia	Neil Chamey	29652	4
32	Australia	Susan Chestnut	29656	4
33	Australia	James Clark	29665	4
34	Australia	Eric Coleman	29670	4
35	Northeast	Aaron Con	29675	12
36	Southeast	Pamela Cox	29685	12
37	Northwest	Conor Cunningh...	29690	12
38	Australia	Barbara Decker	29700	4
39	Southwest	Aidan Delaney	29702	12

	Territorio	Cliente	CustomerID	Ordenes
39	Southwest	Aidan Delaney	29702	12
40	Australia	Bruno Deniut	29706	4
41	Southeast	Andrew Dixon	29715	12
42	Southwest	Blaine Dockter	29716	12
43	Southeast	Gary Drury	29721	12
44	Northeast	Bernard Duerr	29724	12
45	Northeast	Carla Eldridge	29731	12
46	Germany	Marc Faeber	29745	8
47	Southwest	Garth Fort	29757	12
48	United Ki...	Leo Giakoumakis	29778	8
49	Northwest	Jean Handley	29810	12
50	Southeast	Mark Hanson	29811	12
51	Germany	William Hapke	29812	8
52	Northwest	Roger Harui	29818	12
53	Australia	Neal Hasty	29821	4
54	Australia	Jeff Hay	29823	4
55	Southeast	James Hendergart	29825	12
56	Central	Valerie Hendricks	29827	12
57	Australia	Jeff Henshaw	29829	4
58	Southwest	Cheryl Hering	29834	12

	Territorio	Cliente	CustomerID	Ordenes
59	Germany	Onetha Higgs	29837	8
60	Northeast	Nancy Hirota	29844	12
61	Southeast	Douglas Hite	29846	12
62	Southwest	Lucio Iallo	29865	12
63	Southeast	Samuel Johnson	29880	12
64	Southeast	Brannon Jones	29888	12
65	Southwest	Kay Krane	29897	12
66	Southwest	John Kelly	29901	12
67	United Ki...	Mary Kesslep	29904	8
68	Southwest	Frank Campbell	29938	12
69	Central	David Liu	29955	12
70	Australia	Kok-Ho Loh	29960	4
71	Southeast	Spencer Low	29963	12
72	Northeast	Richard Lum	29966	12
73	Australia	Sharon Lynn	29970	4
74	Australia	Jenny Lysaker	29972	4
75	United Ki...	Linda Martin	29985	8
76	Southwest	Sandra Maynard	29992	12
77	Australia	Ramesh Meyyap...	30015	4
78	United Ki...	Matthew Miller	30018	8

	Territorio	Cliente	CustomerID	Ordenes
74	Australia	Jenny Lysaker	29972	4
75	United Ki...	Linda Martin	29985	8
76	Southwest	Sandra Maynard	29992	12
77	Australia	Ramesh Meyyap...	30015	4
78	United Ki...	Matthew Miller	30019	8
79	United Ki...	Stefano Stefani	30030	8
80	Australia	Judy Storjohann	30038	4
81	United Ki...	Krishna Sunkam...	30050	8
82	Australia	Denis Taylor	30059	4
83	Central	Diane Tibbott	30076	12
84	Australia	Gracia Tuell	30092	4
85	Southwest	Sunil Uppal	30095	12
86	Australia	Rachel Valdez	30097	4
87	Northwest	Margaret Vander...	30107	12
88	United Ki...	Raja Venugopal	30113	8
89	Southwest	Robert Vessa	30117	12
90	Canada	Dalton Perez	11091	28
91	Canada	Mason Roberts	11176	28
92	France	April Shan	11566	25

c) Listar los datos generales de las ordenes que tengan al menos los mismos productos de la orden con salesorderid = 43676.

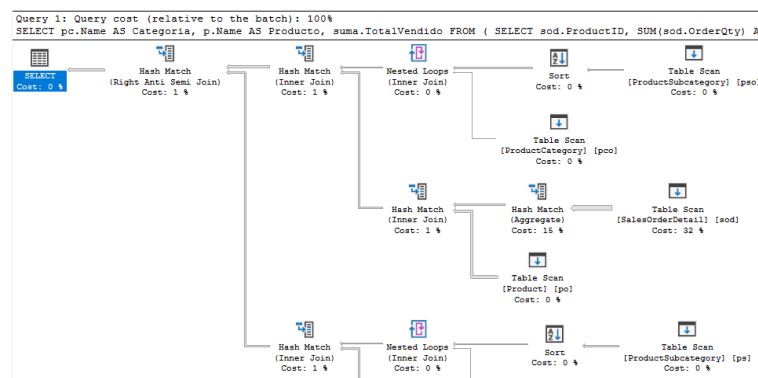
	productid
1	776
2	778
3	710
4	775
5	709

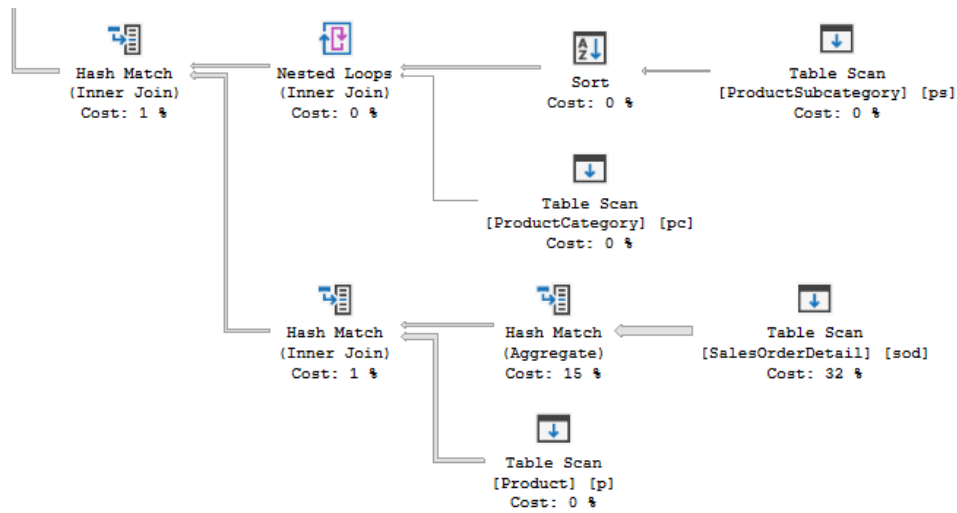
	salesorderid	SalesOrderDetailID	productid	OrderQty
1	43659	1	776	1
2	43659	2	777	3
3	43659	3	778	1
4	43659	4	771	1
5	43659	5	772	1
6	43659	6	773	2
7	43659	7	774	1
8	43659	8	714	3
9	43659	9	716	1
10	43659	10	709	6
11	43659	11	712	2
12	43659	12	711	4
13	43660	13	762	1

4. Generar los planes de ejecución de las consultas en las bases de datos practicaPE y proponer índices para mejorar el rendimiento de las consultas.

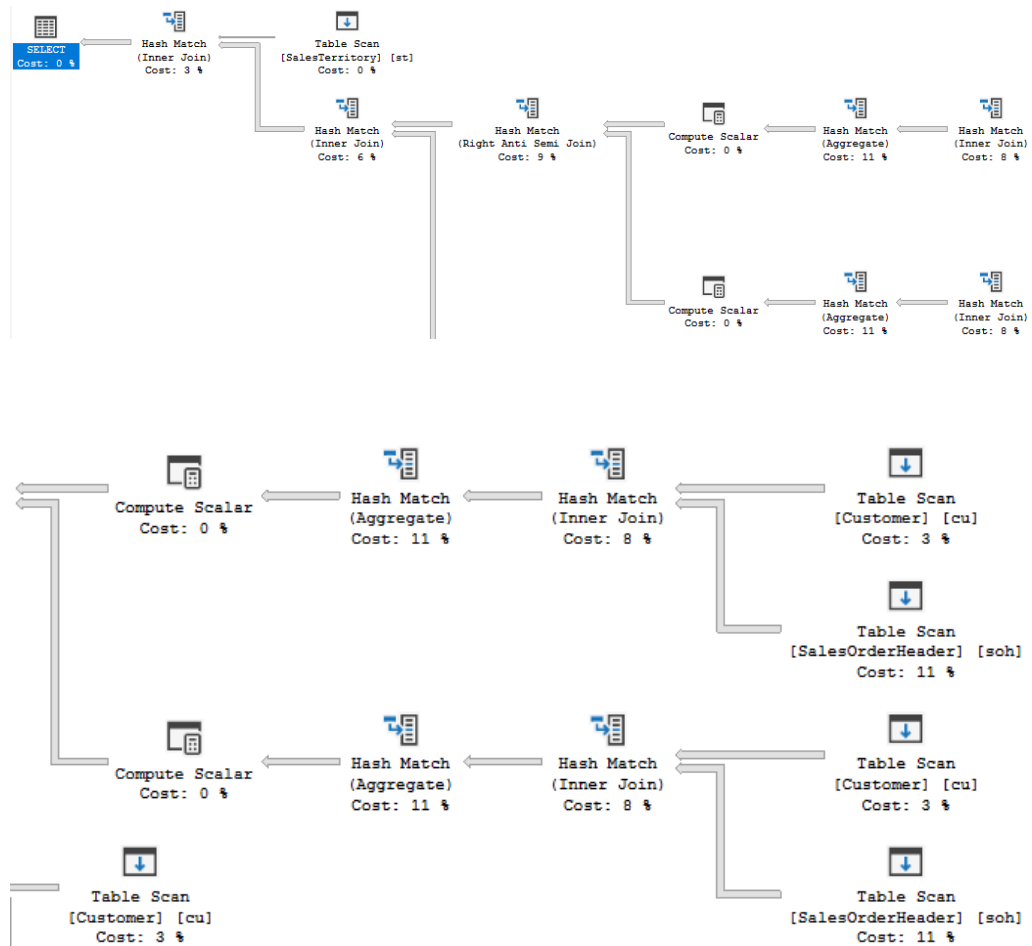
-Planes de Ejecuciones:

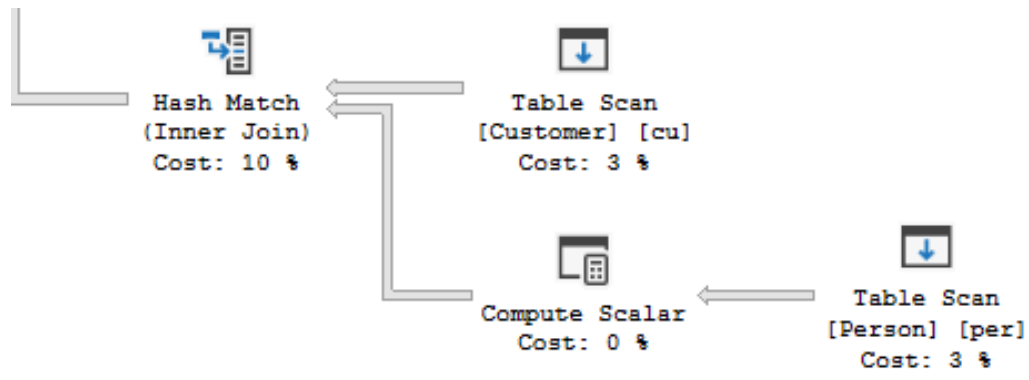
a)



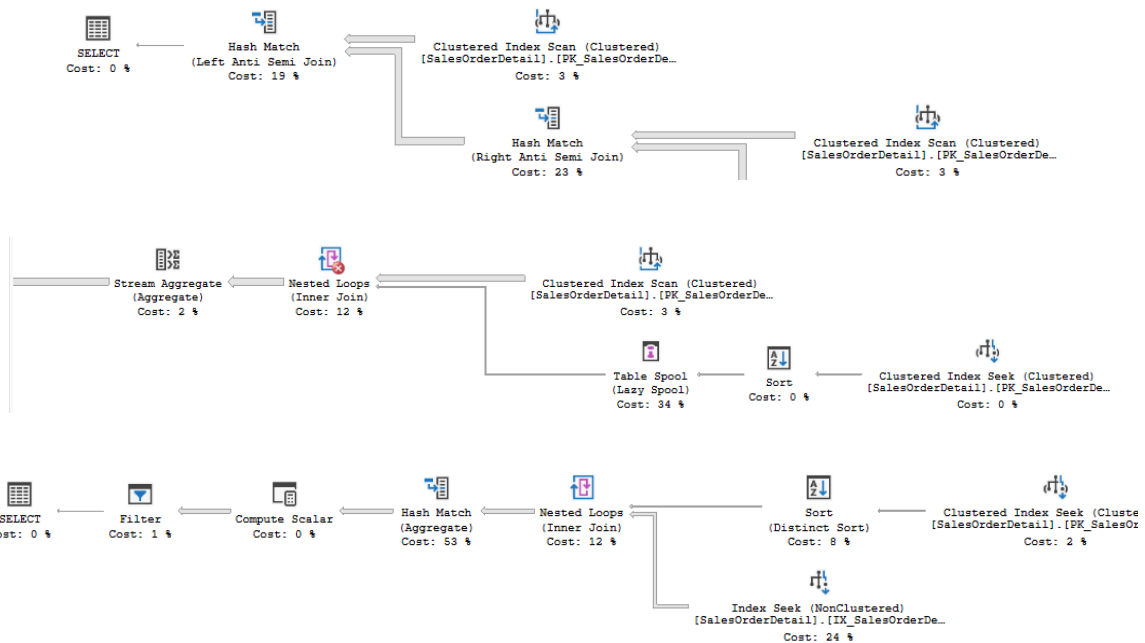
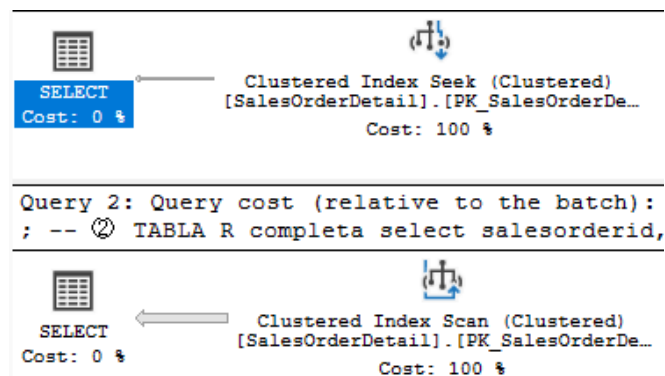


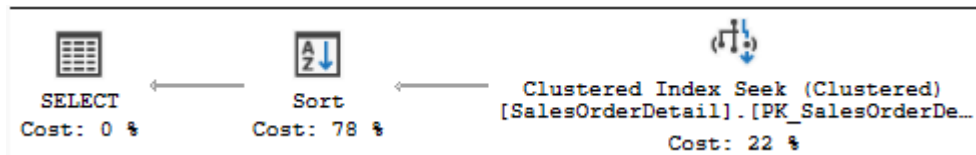
b)





c)





Los índice propuestos son:

b) 1. CREATE NONCLUSTERED

INDEX IX_Person_BEID

ON dbo.Person (BusinessEntityID)

INCLUDE (FirstName, LastName);

GO

2. CREATE NONCLUSTERED

INDEX IX_SOH_CustomerID

ON dbo.SalesOrderHeader (CustomerID);

GO

→ Con el uso de estos índices ayudará a que a la hora de la ejecución no se recorran las tablas completas y solo se recorra lo que se utilizara. Esto se observo porque en las consultas se utilizaron comandos como "Join", "Group by" & "where"

SELECT	
Cached plan size	200 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	4.28936
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	18342.7
Statement	
SELECT	
st.Name AS Territorio,	
per.FirstName + ' ' + per.LastName AS Cliente,	
cu.CustomerID,	
conteo.Ordenes	
FROM (
SELECT	
soh.CustomerID,	
cu.TerritoryID,	
COUNT(*) AS Ordenes	
FROM SalesOrderHeader soh	
JOIN Customer cu ON cu.CustomerID = soh.CustomerID	
GROUP BY soh.CustomerID, cu.TerritoryID	
) AS conteo	
JOIN Customer cu ON cu.CustomerID = conteo.CustomerID	
JOIN Person per ON per.BusinessEntityID = cu.PersonID	
JOIN SalesTer...	

Nota: En el inciso **a) y c)** no hay necesidad de proponer o en su defecto agregar algún otro índice ya que ambas consultas estan optimizadas en su funcionamiento.

Comprobación:

a)

SELECT	
Cached plan size	24 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0032917
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	8.8125
Statement	
-- ① PRODUCTOS del pedido 43676	
select productid	
from AdventureWorks2022.sales.SalesOrderDetail	
where salesorderid = 43676	

Query 1: Query cost (relative to the batch): 0%
 -- ① PRODUCTOS del pedido 43676 select productid from AdventureWorks2022.sales.SalesOrderDetail where salesorderid = 43676

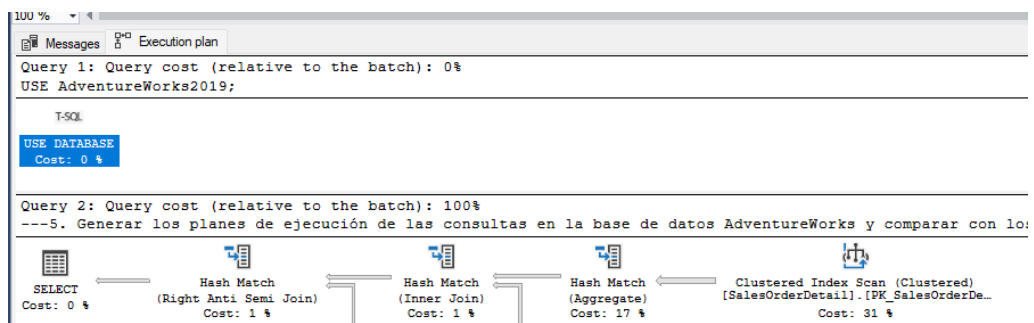
c)

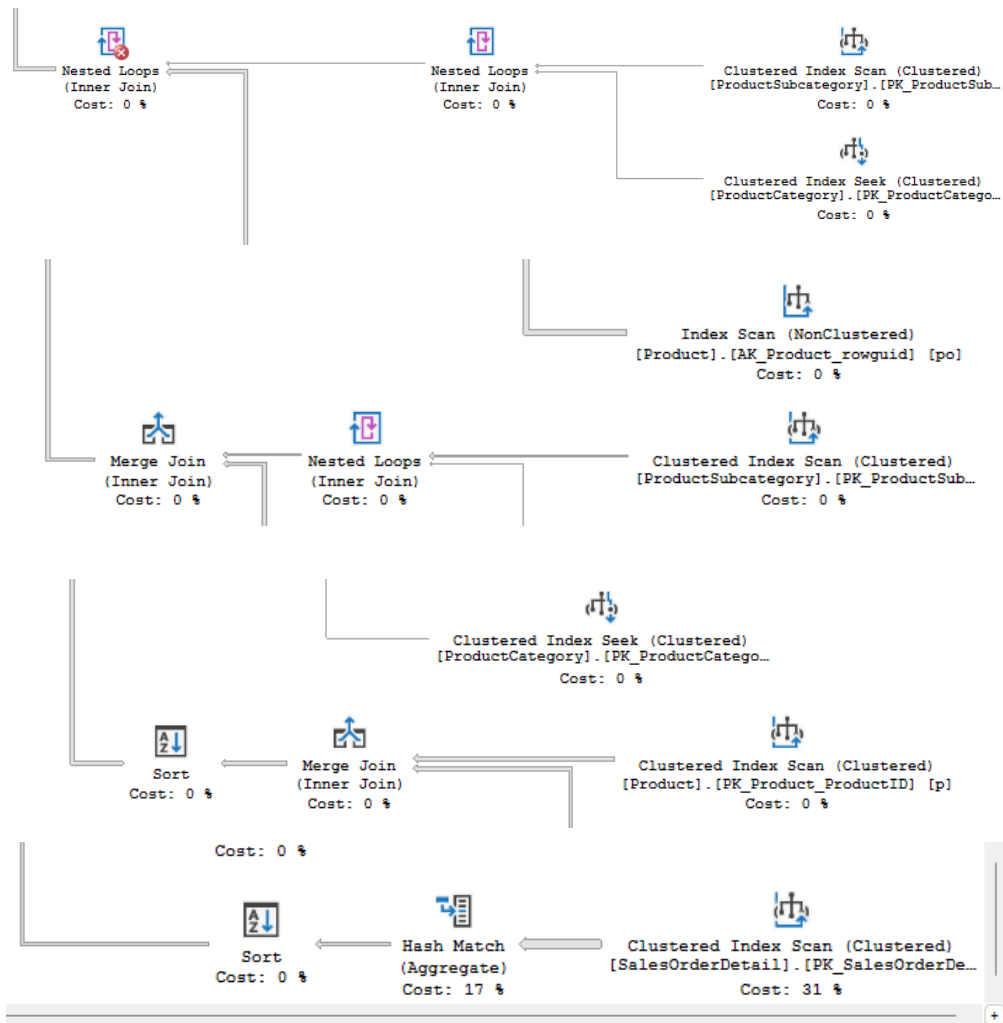
SELECT	
Cached plan size	24 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0148551
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	25.3052
Statement	
;	
-- ② Productos de pedidos específicos (para comparación visual)	
select salesorderid, productid	
from AdventureWorks2022.Sales.SalesOrderDetail	
where SalesOrderID in (43676, 46052, 43891)	
order by SalesOrderID, ProductID	

Query 1: Query cost (relative to the batch): 100%
 SELECT st.Name AS Territorio, per.FirstName + ' ' + per.LastName AS Cliente, cu.CustomerID, conteo.Ordenes FROM (SELECT soh.Cust...

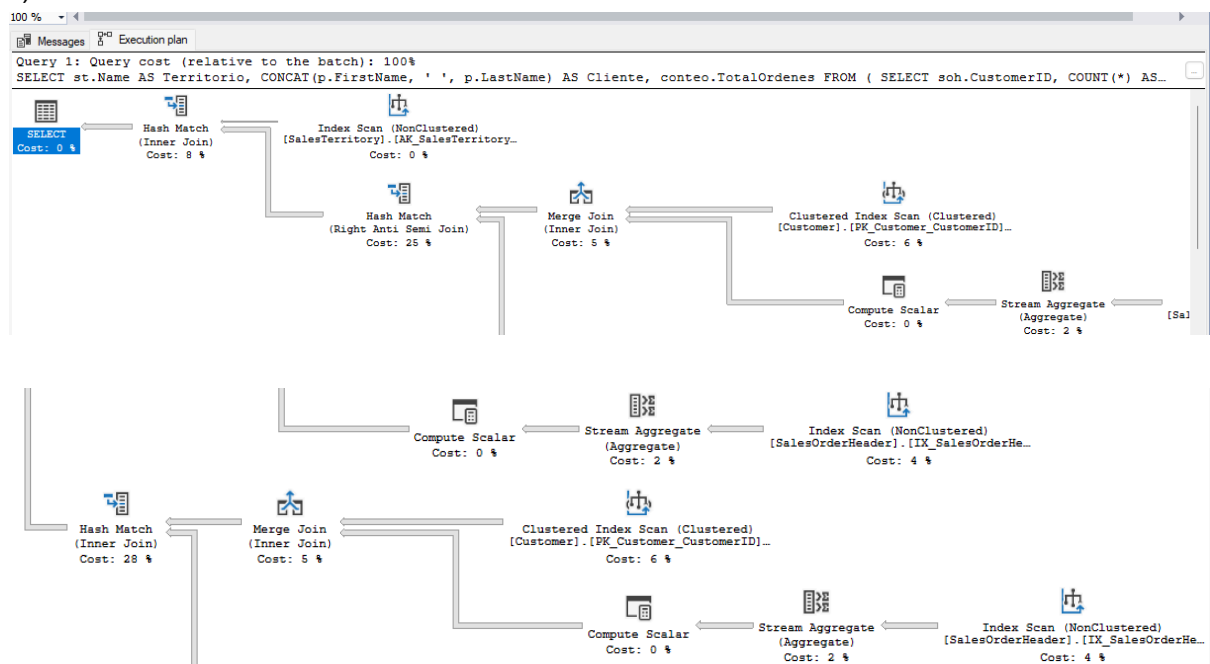
5. Generar los planes de ejecución de las consultas en la base de datos AdventureWorks y comparar con los planes de ejecución del punto 4.

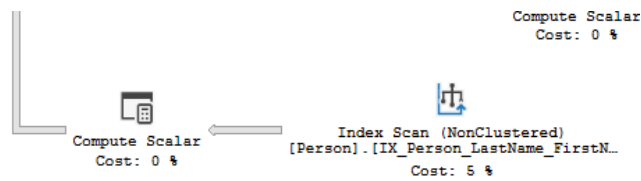
a)



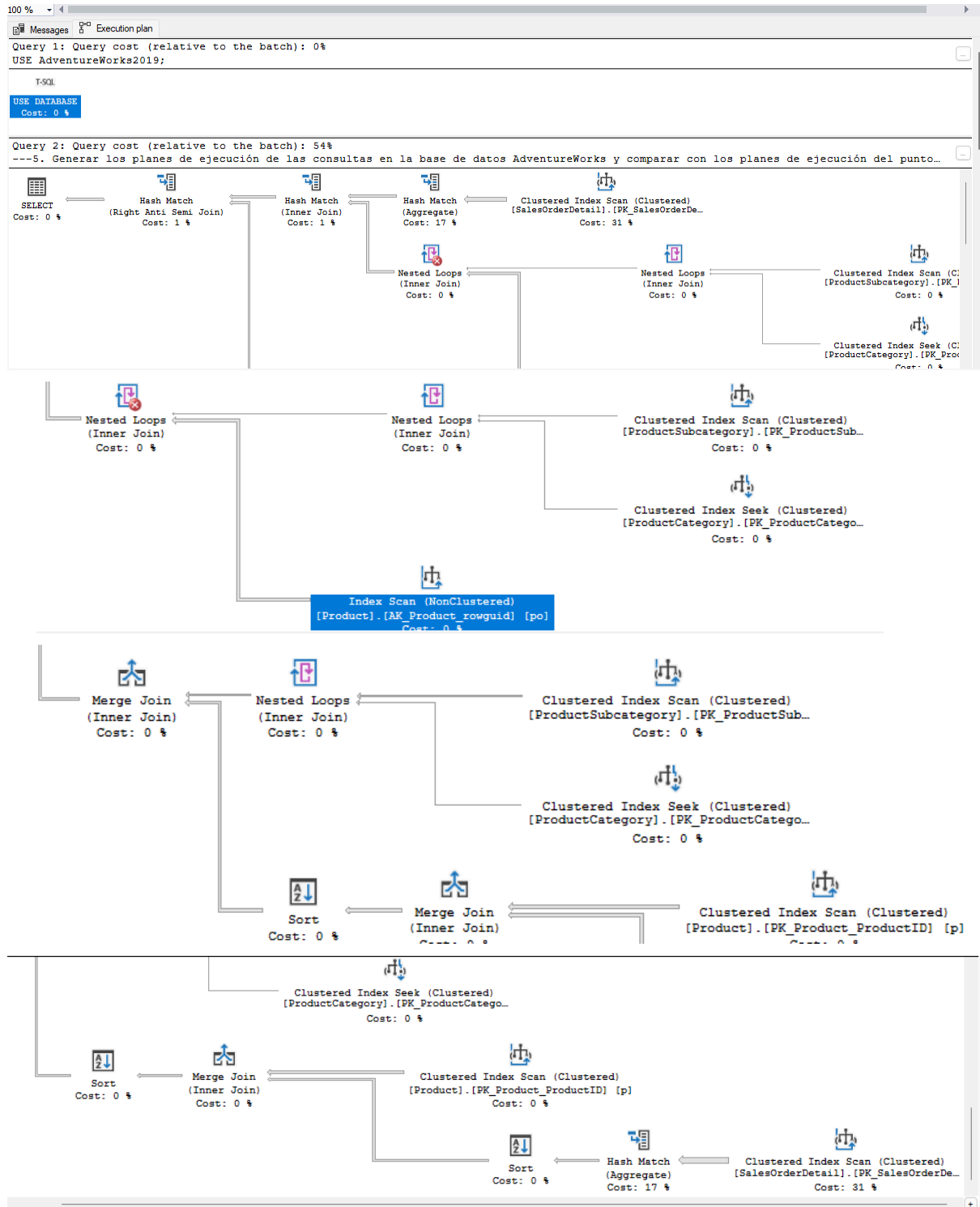


b)



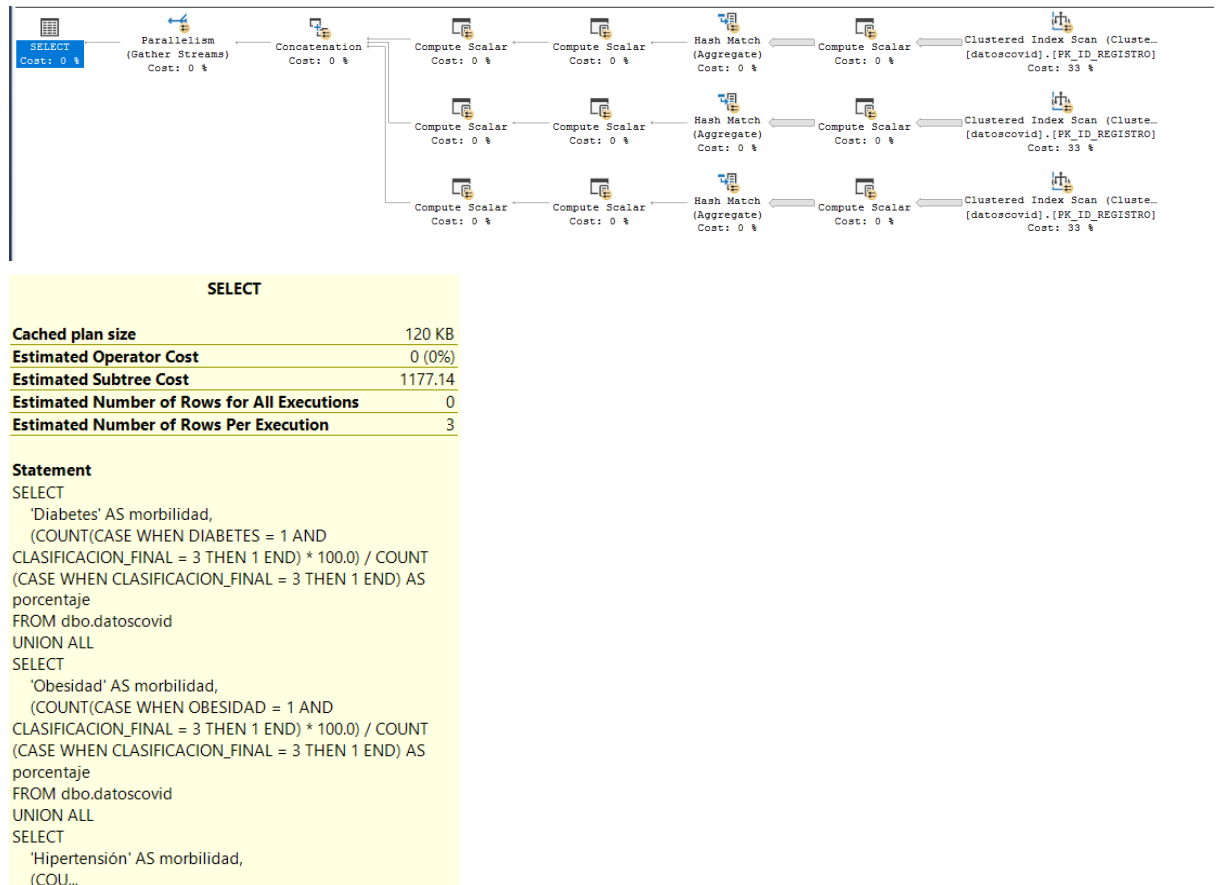


c)



6. Generar los planes de ejecución de las consultas 3, 4 5 de la práctica de consultas en la base de datos Covid y propone índices para mejorar el rendimiento.

- **Consulta 3:**



Los índice propuestos son:

1. CREATE NONCLUSTERED INDEX
INDEX_CLASIFICACION_DIABETES on
datoscovid(CLASIFICACION_FINAL, DIABETES)
2. CREATE NONCLUSTERED INDEX
INDEX_CLASIFICACION_OBESIDAD on
datoscovid(CLASIFICACION_FINAL, OBESIDAD)
3. CREATE NONCLUSTERED INDEX
INDEX_CLASIFICACION_HIPERTENSION on
datoscovid(CLASIFICACION_FINAL, HIPERTENSION)

→ Se propusieron estos índices porque el programa buscara primero la clasificacion_final=3 seguido de las enfermedades especificadas: obesidad, diabetes e hipertension.

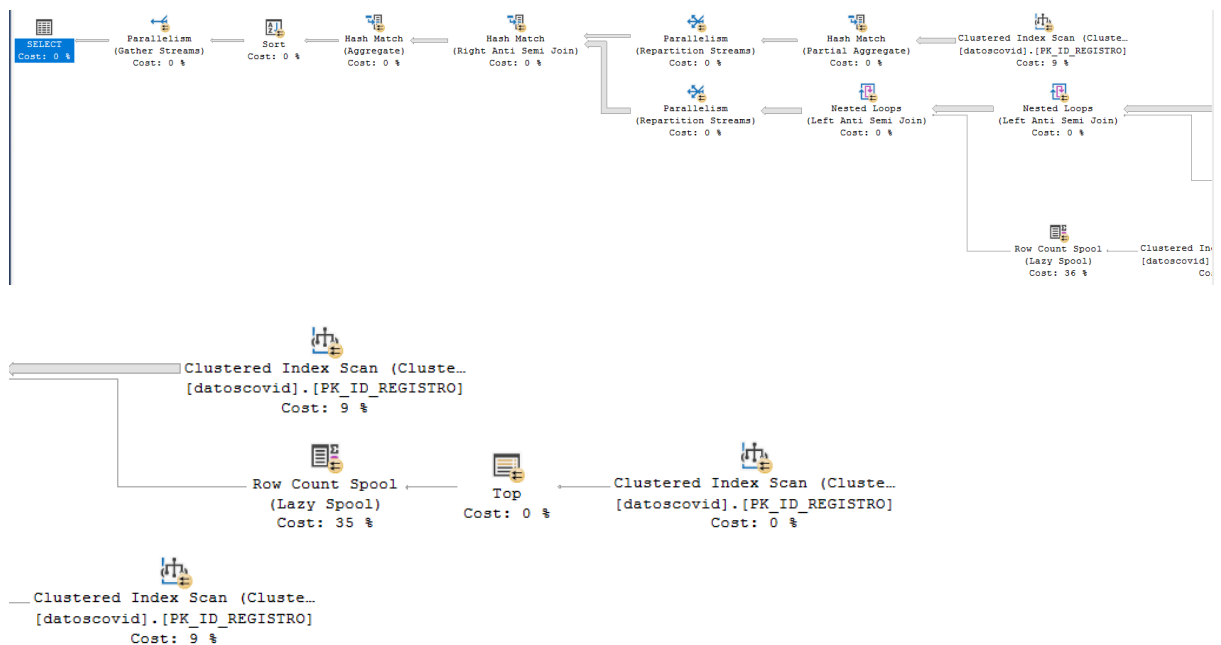
Con índices propuestos:

SELECT	
Cached plan size	120 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	107.623
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	3

Statement

```

SELECT
    'Diabetes' AS morbilidad,
    (COUNT(CASE WHEN DIABETES = 1 AND
CLASIFICACION_FINAL = 3 THEN 1 END) * 100.0) / COUNT
(CASE WHEN CLASIFICACION_FINAL = 3 THEN 1 END) AS
porcentaje
FROM dbo.datoscovid
UNION ALL
SELECT
    'Obesidad' AS morbilidad,
    (COUNT(CASE WHEN OBESIDAD = 1 AND
CLASIFICACION_FINAL = 3 THEN 1 END) * 100.0) / COUNT
(CASE WHEN CLASIFICACION_FINAL = 3 THEN 1 END) AS
porcentaje
FROM dbo.datoscovid
UNION ALL
SELECT
    'Hipertensión' AS morbilidad,
    (COU...
```



El indice propuesto es:

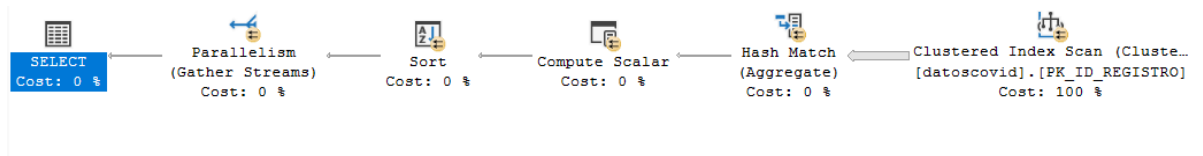
1. CREATE NONCLUSTERED INDEX **INDEX_CLASIFICACION_FINAL** on datoscovid (CLASIFICACION_FINAL) include (MUNICIPIO_RES, DIABETES, OBESIDAD, TABAQUISMO)

→ Lo elegimos porque se necesita filtrar de la columna clasificacion final y ademas, debe incluirse los casos de diabetes, obesidad, tabaquismo y sus respectivos municipios.

Plan de ejecucion con indices:

SELECT	
Cached plan size	112 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	3209.64
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	347.137
Statement	
SELECT DISTINCT MUNICIPIO_RES FROM dbo.datoscovid WHERE MUNICIPIO_RES NOT IN (SELECT DISTINCT MUNICIPIO_RES FROM dbo.datoscovid WHERE CLASIFICACION_FINAL = 3 -- Casos confirmados AND (DIABETES = 1 OR OBESIDAD = 1 OR TABAQUISMO = 1)) ORDER BY MUNICIPIO_RES	

- Consulta 5:



SELECT	
Cached plan size	40 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	393.947
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	33
Statement	
SELECT ENTIDAD_NAC AS estado, COUNT(*) AS total_casos_con_neumonia FROM dbo.datoscovid WHERE CLASIFICACION_FINAL = 3 -- Casos confirmados AND NEUMONIA = 1 -- Pacientes con neumonia GROUP BY ENTIDAD_NAC ORDER BY total_casos_con_neumonia DESC	

Los indice propuesto es:

1. CREATE NONCLUSTERED INDEX **INDEX_NEUMONIA** on datoscovid (neumonia, clasificacion_final) include (entidad_nac)

→ Fue propuesto porque se encontrara con mayor facilidad los casos de neumonia seguido de un filtrado de clasificacion final= 3

Plan de ejecucion con indices:

SELECT	
Cached plan size	48 KB
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	2.20621
Estimated Number of Rows for All Executions	0
Estimated Number of Rows Per Execution	33
Statement	
SELECT	
ENTIDAD_NAC AS estado,	
COUNT(*) AS total_casos_con_neumonia	
FROM dbo.datoscovid	
WHERE CLASIFICACION_FINAL = 3 -- Casos confirmados	
AND NEUMONIA = 1 -- Pacientes con neumonia	
GROUP BY ENTIDAD_NAC	
ORDER BY total_casos_con_neumonia DESC	

7. Comprobar los planes de ejercicio del punto 6 con los planes de ejecucion de otro equipo.

Comprobacion con el equipo 1.

- Consulta 3:

The screenshot displays the Microsoft SQL Server Management Studio interface. The top pane shows a query in the 'SQLQuery1.sql - ARTURO.covidHistorico (sa (64))' window. The query is as follows:

```
responsable: palacios reyes leslie noemi
comentarios:
-----/
select
  'diabetes' as morbilidad,
  100.0 * sum(case when diabetes = 1 then 1 else 0 end) / count(*) as porcentaje
from datoscovid
where clasificacion_final in (1, 2, 3)
union all
select
  'obesidad' as morbilidad,
  100.0 * sum(case when obesidad = 1 then 1 else 0 end) / count(*) as porcentaje
from datoscovid
where clasificacion_final in (1, 2, 3)
```

The bottom pane shows the 'Execution plan' for 'Query 1: Query cost (relative to the batch): 100%'. The plan includes a 'Missing Index' warning: 'Missing Index (Impact 32.6354): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[datoscovid] ([CLASIFICACION_FINAL]) INCLUDE ([HIPERTEN...]'.

The execution plan diagram shows the following operators and their costs:

- Parallelism (Gather Streams): Cost: 0 %
- Concatenation: Cost: 0 %
- Compute Scalar: Cost: 0 %
- Hash Match (Aggregate): Cost: 0 %
- Clustered Index Scan (Clustered) [datoscovid].[PK_ID_REGISTRO]: Cost: 33 %

The status bar at the bottom indicates 'Query executed successfully.' and 'ARTURO (16.0 RTM) sa (64) covidHistorico 00:00:00 0 rows'.

- Consulta 4:

SQLQuery1.sql - ARTURO.covidHistorico (sa (64)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Quick Launch (Ctrl+Q)

covidHistorico

Execute

SQLQuery1.sql - ART...Historico (sa (64))

```
CREATE NONCLUSTERED INDEX idx_tabaq
ON datoscovid (TABAQUSMO);

SELECT MUNICIPIO_RES
FROM datoscovid
WHERE CLASIFICACION_FINAL IN (1, 2, 3) -- Casos confirmados
GROUP BY MUNICIPIO_RES
HAVING SUM(CASE WHEN HIPERTENSION = 1 THEN 1 ELSE 0 END) = 0
AND SUM(CASE WHEN OBESIDAD = 1 THEN 1 ELSE 0 END) = 0
AND SUM(CASE WHEN DIABETES = 1 THEN 1 ELSE 0 END) = 0
AND SUM(CASE WHEN TABAQUSMO = 1 THEN 1 ELSE 0 END) = 0
```

Consulta No.5 Listar los estados con más casos recuperados con neumonía

100 %

Messages Execution plan

Query 1: Query cost (relative to the batch): 100%

SELECT MUNICIPIO_RES FROM datoscovid WHERE CLASIFICACION_FINAL IN (1, 2, 3) -- Casos confirmados GROUP BY MUNICIPIO_RES HAVING SUM(CASE WHEN HIPERTENSION = 1...

Missing Index (Impact 94.6329): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[datoscovid] ([CLASIFICACION_FINAL]) INCLUDE ([MUNICIPIO...

Execution Plan:

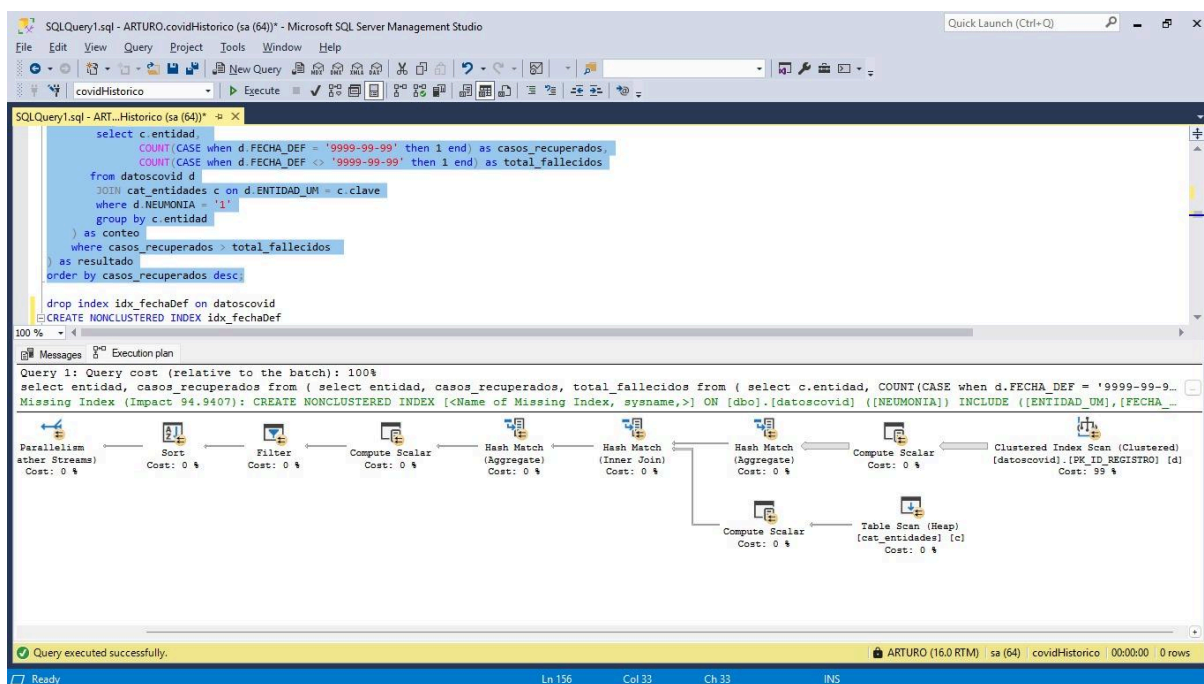
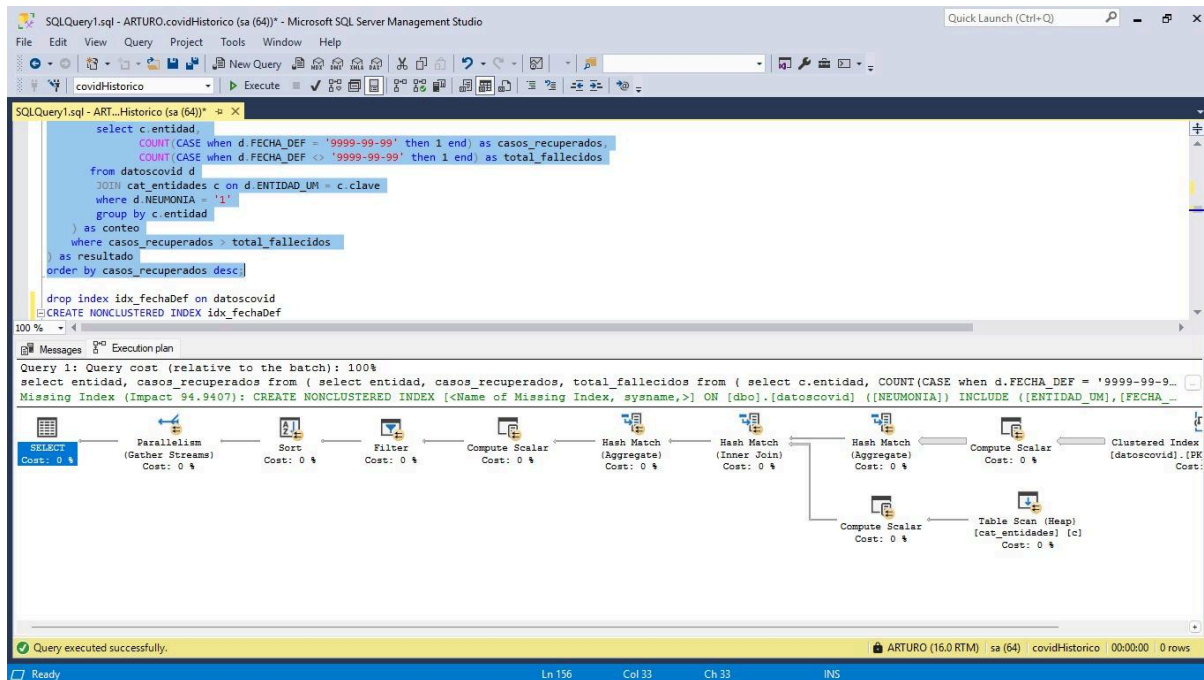
- SELECT (Cost: 0 %)
- Parallelism (Gather Streams) (Cost: 0 %)
- Filter (Cost: 0 %)
- Hash Match (Aggregate) (Cost: 1 %)
- Compute Scalar (Cost: 0 %)
- Clustered Index Scan (Clustered) [datoscovid].[PW_ID_REGISTRO] (Cost: 99 %)

Query executed successfully.

ARTURO (16.0 RTM) sa (64) covidHistorico 00:00:01 0 rows

Ready Ln 117 Col 1 Ch 1 INS

- Consulta 5:



Conclusiones:

- Bernal Aguilar Yuvia Abigail

Podemos confirmar que los planes de ejecución son muchas veces fundamentales para poder conocer mejor sobre el proceso que se lleva a cabo al realizar una consulta en una base de datos ya que nos ayudaron para saber como es el proceso, que parte de la consulta se consume más recursos, problemas de estadísticas y de identificar la falta de índices, los cuales son indispensables en una base de datos que contienen miles de información y es necesario acceder a esta información de forma optima. Entre estos planes, observamos unos conceptos como lo son nested loop,

hash match y merge join. El primero es una unión de bucle anidado en donde se escanea la primera tabla y después se unen las filas coincidentes de la segunda tabla, el segundo término, hash match, que se crea una tabla hash para que después, de igual forma buscar coincidencias con la otra tabla. Por último, tenemos merge join ésta es diferente a las anteriores ya que se necesita que las dos tablas deberán estar ordenadas para después escanearse al mismo tiempo y encontrar términos iguales para combinarlas.

- Medina Gómez Jimena Zarahí

Con esta práctica comprendí la utilidad de consultar los planes de ejecución, su importancia y los distintos datos que se pueden obtener de ellos. Estos planes facilitan la identificación de mejoras que no son evidentes al momento de compilar los queries. Todo este análisis se realiza con el objetivo de aumentar la velocidad de cada consulta y determinar si se requieren índices adicionales que permitan optimizar su ejecución.

Además, al comparar los distintos planes de ejecución en diferentes equipos, puede establecer estrategias que demuestran cómo muchas consultas pueden simplificarse sin alterar sus resultados, logrando una ejecución más eficiente y un menor consumo de recursos.

- Contreras Jiménez Mariana Montserrat

En esta práctica comprendí la importancia de ejecutar consultas junto con la revisión de sus planes de ejecución, ya que esto no solo permite verificar su funcionamiento, sino también evaluar su eficiencia y el uso de recursos, especialmente al trabajar con grandes volúmenes de datos. A través del análisis de cada plan, fue posible identificar operaciones costosas y detectar oportunidades para optimizar el rendimiento mediante el uso de índices. Los índices, correctamente aplicados, ayudan a dirigir las búsquedas de forma más precisa, especialmente en consultas que utilizan comandos como join o select. Incorporarlos de manera adecuada permite obtener respuestas más rápidas, claras y concisas, lo cual mejora significativamente la experiencia en el manejo de bases de datos.

Conclusión General:

Durante esta práctica comprendimos que los planes de ejecución son herramientas fundamentales para analizar y mejorar el rendimiento de las consultas en bases de datos. Estos planes nos permiten observar con detalle cómo se procesa una consulta, qué partes consumen más recursos, y nos ayudan a identificar problemas como estadísticas desactualizadas o la falta de índices, lo cual es crucial en bases de datos con grandes volúmenes de información. A través del análisis, detectamos conceptos técnicos como nested loop, hash match y merge join, cuya comprensión nos permitió conectar la teoría con el código SQL: por ejemplo, entendimos que nested loop escanea fila por fila para encontrar coincidencias, hash match crea una tabla hash para optimizar la comparación, y merge join requiere que las tablas estén ordenadas para unirlos eficientemente. Al comparar los planes generados en distintos escenarios, también identificamos cómo reestructurar consultas sin cambiar sus

resultados, logrando una ejecución más simple, rápida y con menor carga de recursos. En definitiva, analizar los planes de ejecución no sólo valida el funcionamiento del código, sino que lo lleva a un nivel más eficiente y profesional.