INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA EN TECNOLOGÍAS AVANZADAS

Ingeniería Telemática

Bases de Datos Distribuidas

Practica 2:

Planes de ejecución.

- **Docente**: De la Cruz Sosa Carlos
- **Alumnos:** De Jesús Lucio Oscar Daniel
  Pérez Iturbe Carolina
- **Grupo:** 3TM3

- **Fecha de entrega**: 29 de mayo 2025
- **Ciclo escolar:** 2025-2

Practica 2 "PLANES DE EJECUCIÓN"

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
        select *
        into practicaPE.dbo.SalesOrderHeader
        from
AdventureWorks2019.Sales.SalesOrderHeader;
    --// select * from SalesOrderHeader --(31,465 rows)
```

```
---/ b) Sales.SalesOrderHeader
        select *
        into practicaPE.dbo.SalesOrderDetail
        from
AdventureWorks2019.Sales.SalesOrderDetail;
    --// select * from SalesOrderDetail --(121,317 rows)
```

```
---/ c) Sales.Customer
        select *
        into practicaPE.dbo.SalesCustomer
        from AdventureWorks2019.Sales.Customer;
    --// select * from SalesCustomer --(19,820 rows)
```

```
---/ d) Sales.Sales.Territory
        select *
        into practicaPE.dbo.SalesTerritory
        from AdventureWorks2019.Sales.SalesTerritory;
    --// select * from SalesTerritory --(10 rows)
```

```
---/ e) Production.Product
        select *
        into practicaPE.dbo.ProductionProduct
        from AdventureWorks2019.Production.Product;
    --// select * from ProductionProduct --(504 rows)
```

```
---/ f) Production.ProductCategory
        select *
        into practicaPE.dbo.ProductionProductCategory
        from
AdventureWorks2019.Production.ProductCategory;
    --// select * from ProductionProductCategory --(4 rows)
```

```
---/ g) Production.ProductSubcategory
        select *
        into
practicaPE.dbo.ProductionProductSubcategory
        from
AdventureWorks2019.Production.ProductSubcategory;
```

```
--// select * from ProductionProductSubcategory --(37 rows)
```

---/ h) PersonPerson

```
select BusinessEntityID, FirstName, LastName
into practicaPE.dbo.PersonPerson
from AdventureWorks2019.Person.Person; --(additionalcotactinfo is type with a shema collection)
--// select * from PersonPerson --(19,972 rows)
```

3.- Codificar las siguientes consultas :

Descripción:	*a. Listar el producto más vendido de cada una de las categorías registradas en la base de datos.*
Catálogos:	-ProductSubcategory -ProductCategory -SalesOrderDetail
Comentarios:	En esta consulta no se ocupó nada que no se haya visto en clase, únicamente fue necesario buscar la lógica para poner los Join y que estos arrojaran el resultado deseado, y esto se guardara en una tabla para poder satisfacer la consulta

```sql
WITH VentasProducto AS (
    SELECT
        pc.Name AS Categoria,
        p.ProductID,
        p.Name AS Producto,
        SUM(od.OrderQty) AS CantidadVendida
    FROM SalesorderDetail od
    JOIN Product p ON od.ProductID = p.ProductID
    JOIN ProductSubcategory ps ON p.ProductSubcategoryID = ps.ProductSubcategoryID
    JOIN ProductCategory pc ON ps.ProductCategoryID = pc.ProductCategoryID
    GROUP BY pc.Name, p.ProductID, p.Name),
ProductoMaximo AS (
    SELECT
        Categoria,
        MAX(CantidadVendida) AS MaxCantidad
    FROM VentasProducto
    GROUP BY Categoria)
SELECT vp.Categoria, vp.Producto, vp.CantidadVendida
FROM VentasProducto vp
JOIN ProductoMaximo pm
  ON vp.Categoria = pm.Categoria AND vp.CantidadVendida = pm.MaxCantidad
ORDER BY vp.Categoria;
```

Descripción:	*b. Listar el nombre de los clientes con más ordenes por cada uno de los territorios registrados en la base de datos.*
Catálogos:	-Customer -Person -SalesOrderHeader
Comentarios:	En esta consulta fue necesario crear una tabla común para poder hallar el resultado de la consulta, en está fue necesario hacer un JOIN entre las tres tablas, hacer un conteo y un Group By posterior para poder obtener a los clientes con más órdenes y después sean agrupados por territorio.

```sql
WITH Customer_Orders AS (
    SELECT
        soh.SalesOrderID,
        c.CustomerID,
        soh.TerritoryID,
        p.FirstName,
        p.LastName
    FROM SalesOrderHeader soh
    JOIN Customer c ON soh.CustomerID = c.CustomerID
    JOIN Person p ON c.PersonID = p.BusinessEntityID),
OrdenesPorCliente AS (
    SELECT
        TerritoryID,
        CustomerID,
        FirstName,
              LastName,
        COUNT(*) AS Orders
    FROM Customer_Orders
    GROUP BY TerritoryID, CustomerID, FirstName, LastName),
MaxOrdenesPorTerritorio AS (
    SELECT
        TerritoryID,
        MAX(Orders) AS MaxOrders
    FROM OrdenesPorCliente
    GROUP BY TerritoryID)
SELECT
    o.TerritoryID,
    o.FirstName,
    o.LastName,
    o.Orders
FROM OrdenesPorCliente o
JOIN MaxOrdenesPorTerritorio m
    ON o.TerritoryID = m.TerritoryID AND o.Orders = m.MaxOrders
ORDER BY o.TerritoryID;
```

Descripción:	*Listar los datos generales de las ordenes que tengan al menos los mismos productos de la orden con salesorderid = 43676.*
Catálogos:	-SalesOrderDetail
Comentarios:	Esta consulta fue la más difícil de las anteriores ya que se tuvo que utilizar la división relacional, un concepto que fue nuevo y algo compliado de utilizar

```sql
SELECT distinct  SalesOrderID,OrderQty, ProductID
FROM SalesOrderDetail AS OD
WHERE NOT EXISTS
(SELECT *
FROM ( select ProductID
             from SalesOrderDetail
             where SalesOrderID = 43676) as P
WHERE NOT EXISTS(
             (select *
                              From SalesOrderDetail as OD2
                              where OD.salesorderid =OD2.SalesOrderID
                              and OD2.ProductID = P.ProductID)));
```

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

Sin índices:

A)

Índices propuestos:

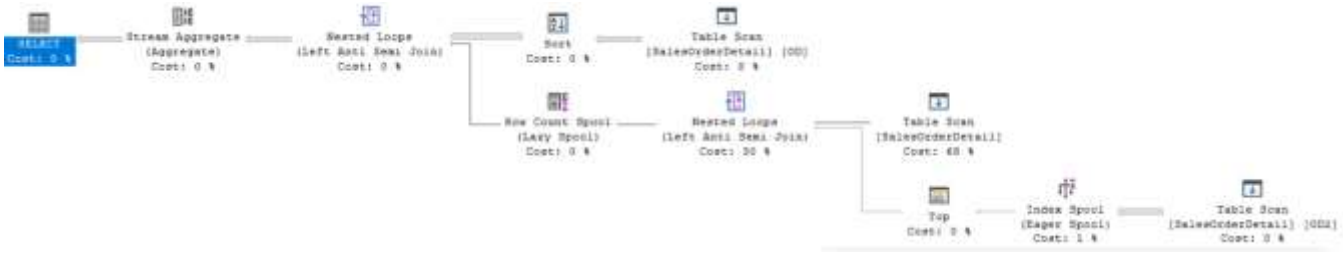
```
-- Índice para acelerar la agregación y búsquedas por ProductID
CREATE NONCLUSTERED INDEX IX_SalesOrderDetail_ProductID_OrderQty
ON SalesOrderDetail (ProductID)
INCLUDE (OrderQty, SalesOrderID);
-- Índice para las uniones en Product
CREATE NONCLUSTERED INDEX IX_Product_ProductSubcategoryID
ON Product (ProductSubcategoryID)
INCLUDE (ProductID, Name);
-- Índice para unir ProductSubcategory con ProductCategory
CREATE NONCLUSTERED INDEX IX_ProductSubcategory_ProductCategoryID
ON ProductSubcategory (ProductCategoryID)
INCLUDE (ProductSubcategoryID);
-- Índice para buscar por nombre en ProductCategory
CREATE NONCLUSTERED INDEX IX_ProductCategory_Name
ON ProductCategory (ProductCategoryID)
INCLUDE (Name);
```

B)

Índices propuestos:

```
-- Índice para acelerar la búsqueda por CustomerID y TerritoryID
CREATE NONCLUSTERED INDEX IX_SalesOrderHeader_CustomerID_TerritoryID
ON SalesOrderHeader (CustomerID, TerritoryID)
INCLUDE (SalesOrderID);


-- Índice para unir Customer con Person
CREATE NONCLUSTERED INDEX IX_Customer_PersonID
ON Customer (PersonID)
INCLUDE (CustomerID);


-- Índice para buscar por BusinessEntityID en Person
CREATE NONCLUSTERED INDEX IX_Person_BusinessEntityID
ON Person (BusinessEntityID)
INCLUDE (FirstName, LastName);
```

C)



Índices propuestos:

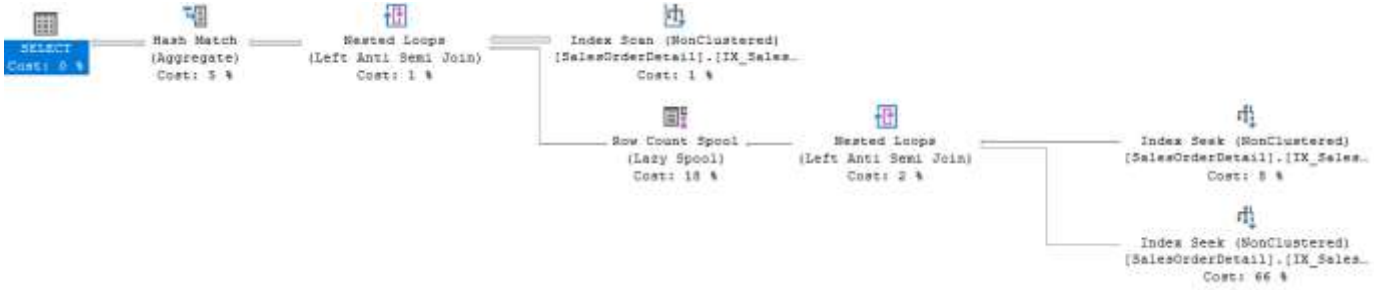
```
-- Índice para acelerar las subconsultas que buscan por SalesOrderID y ProductID
CREATE NONCLUSTERED INDEX IX_SalesOrderDetail_SalesOrderID_ProductID
ON SalesOrderDetail (SalesOrderID, ProductID)
INCLUDE (OrderQty);


-- Acelerar las búsquedas por solo SalesOrderID
CREATE NONCLUSTERED INDEX IX_SalesOrderDetail_SalesOrderID
ON SalesOrderDetail (SalesOrderID)
INCLUDE (ProductID, OrderQty);
```

Con índices:

A)

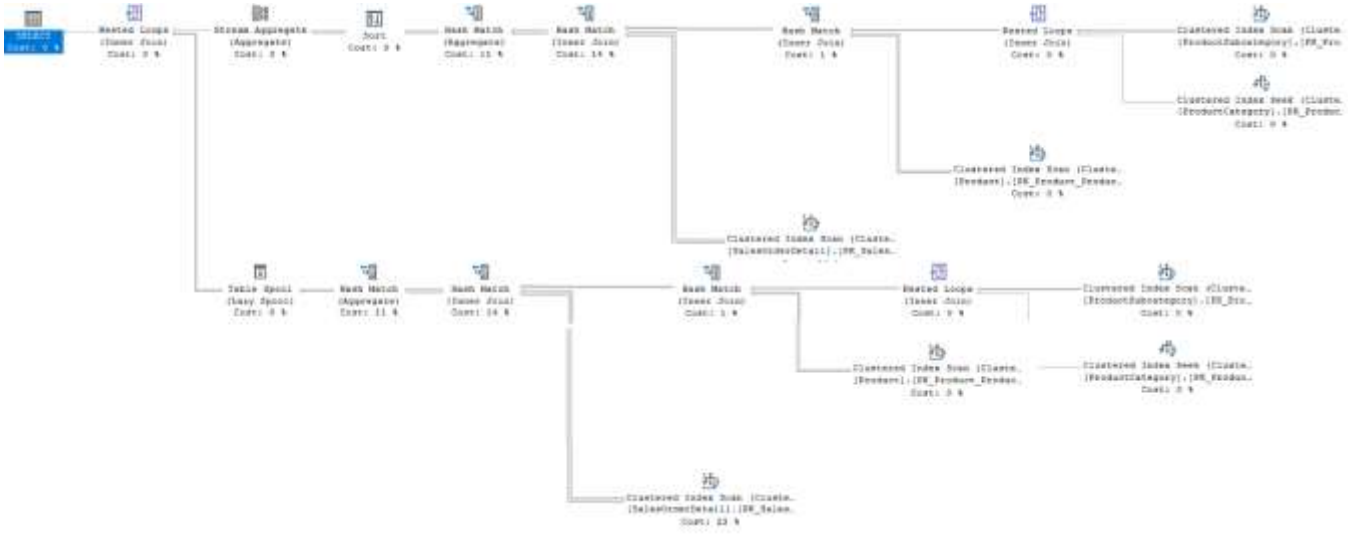
B)

C)



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)



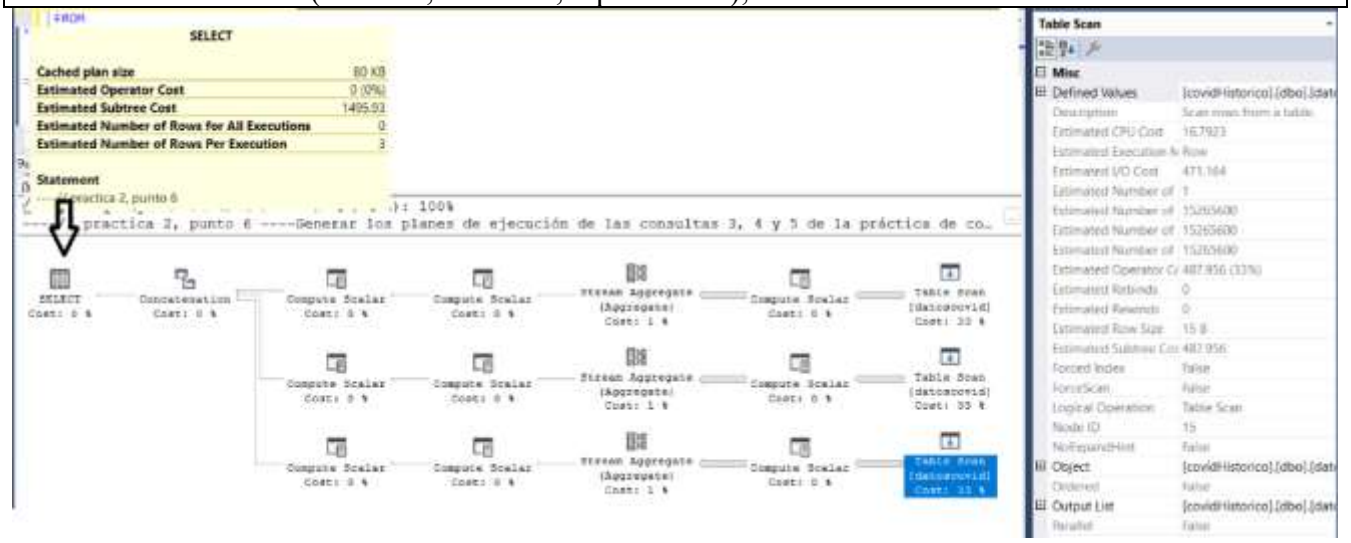
b)

c)

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

-- Consulta 3; indice propuesto;

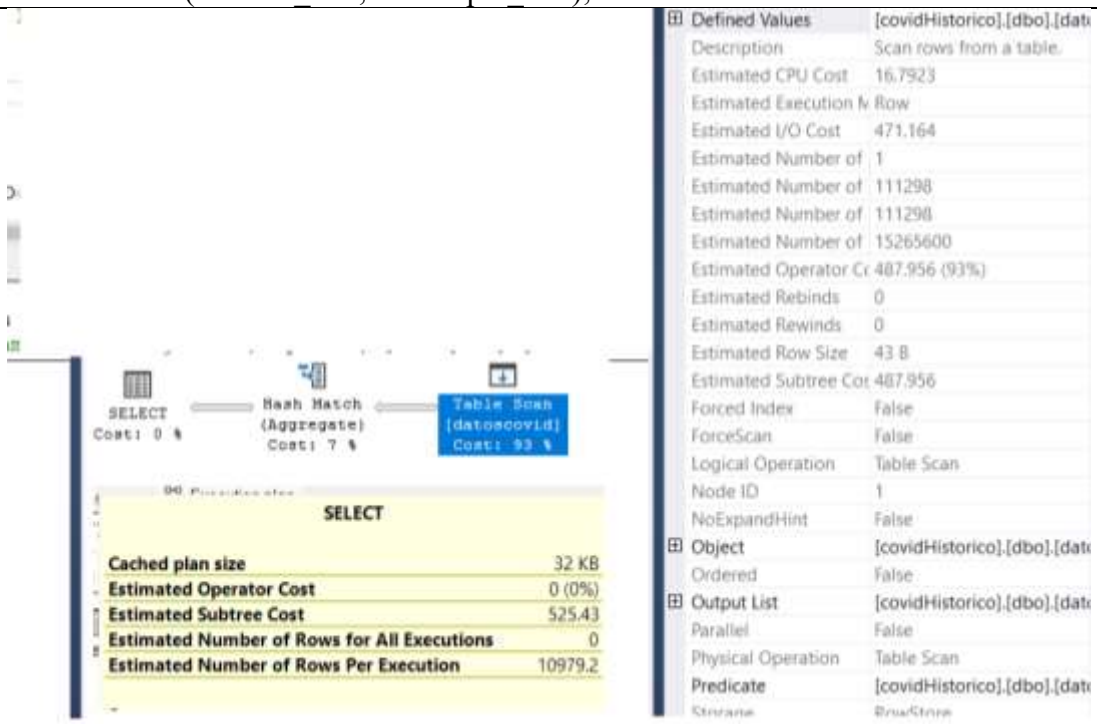
```
CREATE INDEX idx_morbilidades_confirmados
        ON datoscovid (clasificacion_final)
        INCLUDE (diabetes, obesidad, hipertension);
```



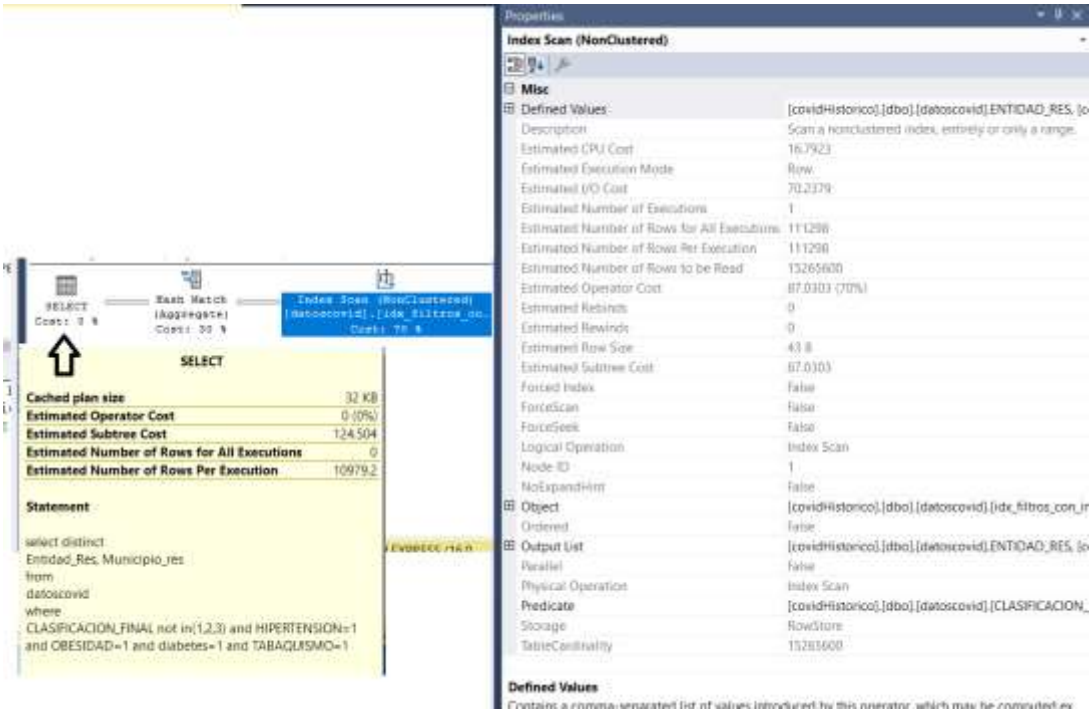
Con índice;

-- Consulta 4; indice propuesto;

```
CREATE INDEX idx_filtros_con_include
        ON datoscovid (CLASIFICACION_FINAL, HIPERTENSION, OBESIDAD, DIABETES, TABAQUISMO)
        INCLUDE (Entidad_Res, Municipio_Res);
```

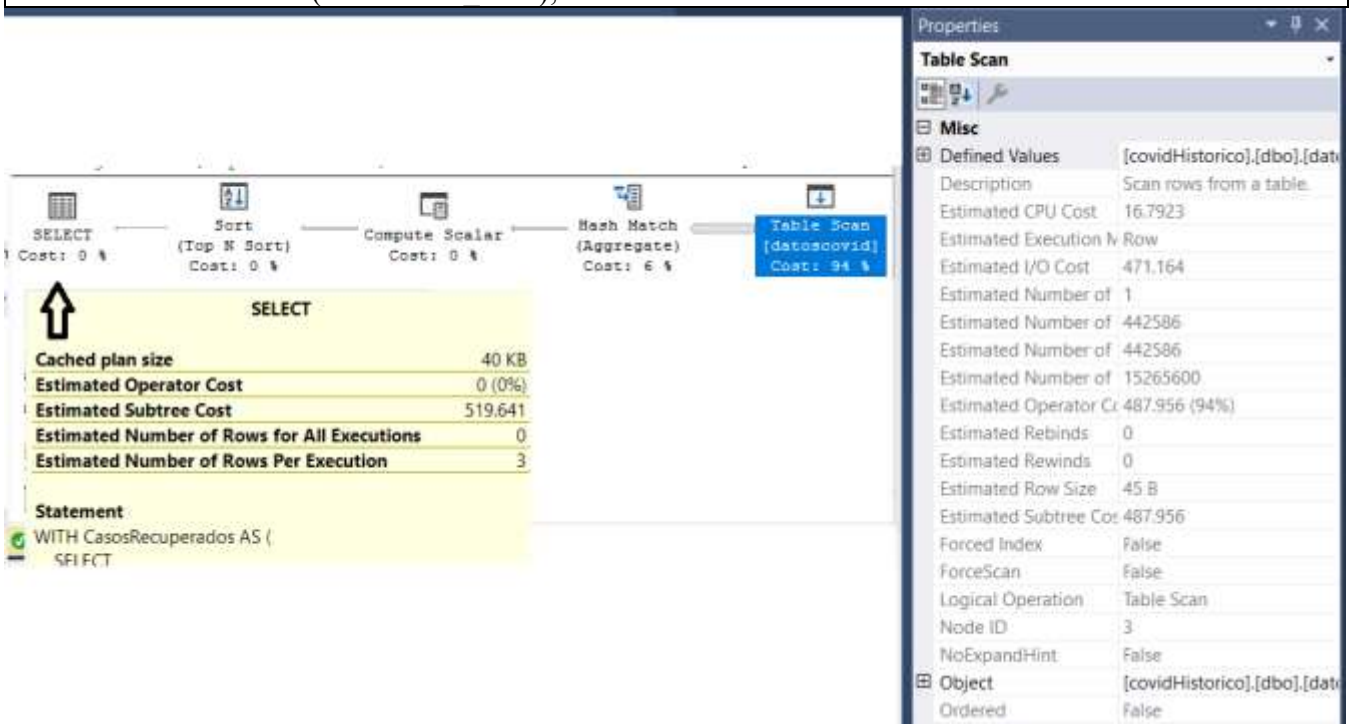


Con índice

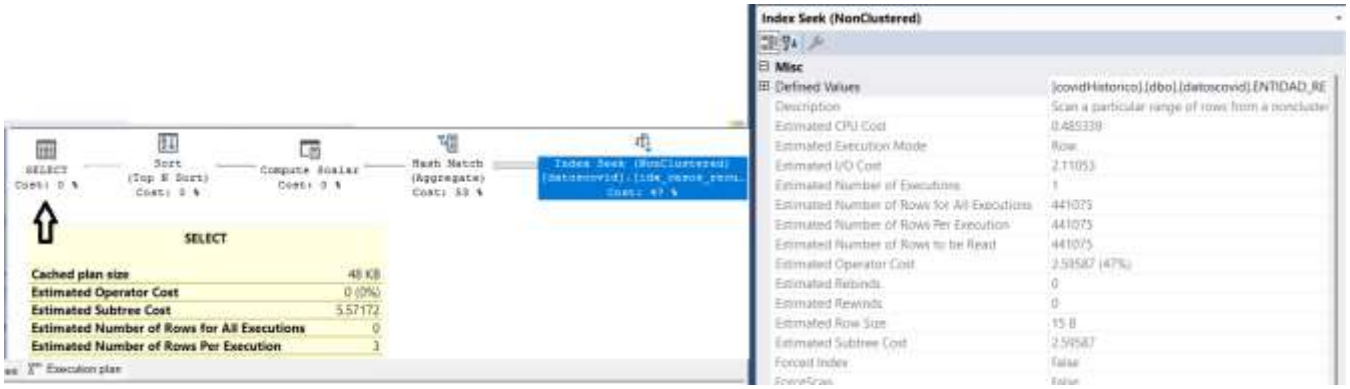


-- Consulta 5; indice propuesto;

```
CREATE INDEX idx_casos_recuperados_neumonia_inc
ON datoscovid (CLASIFICACION_FINAL, FECHA_DEF, NEUMONIA)
INCLUDE (ENTIDAD_RES);
```



Con índices



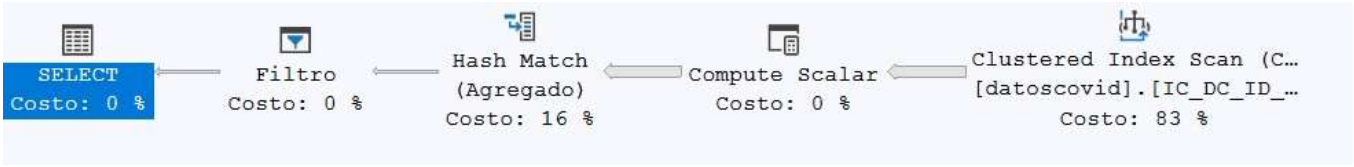
--// 7.- Comparar los planes de ejecución del punto 6 con los planes de ejecución de otro equipo.

Planes de ejecución equipo 6:

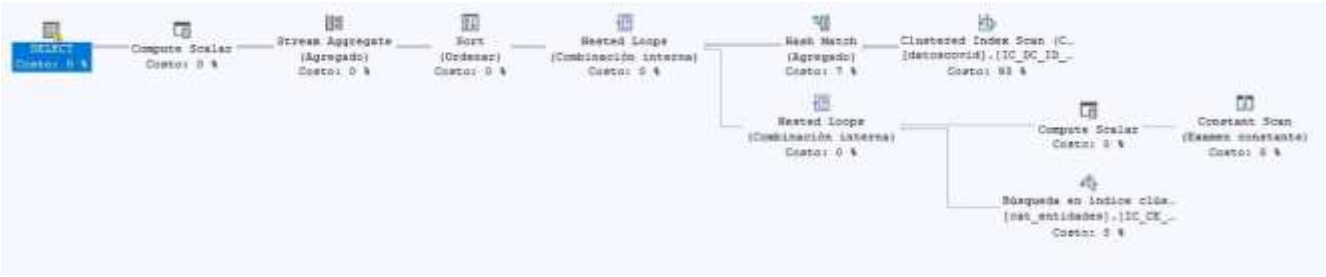
Sin índices:

3)

4)

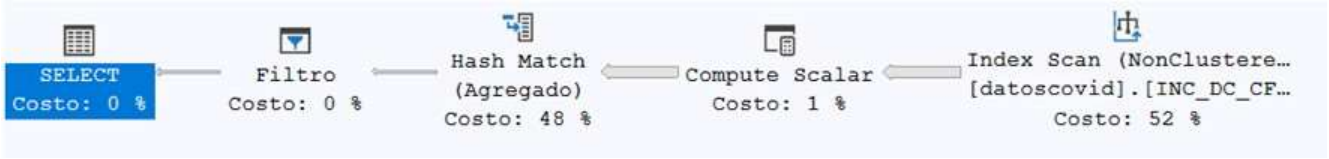


5)

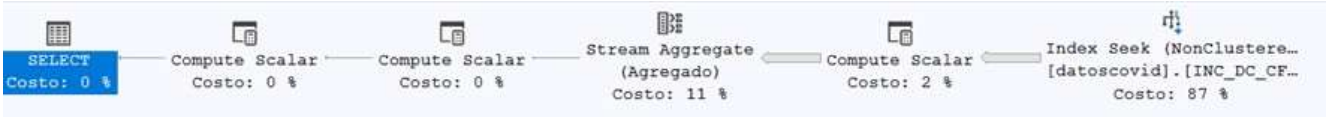


Con índices

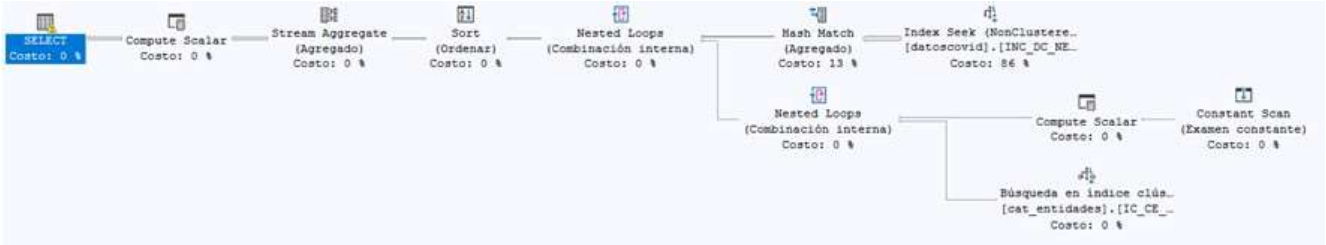
3)



4)



5)



8.- Conclusiones:

Los índices son herramientas del algebra relacional para acelerar los procesos de consultas, porque de no usarlos el motor de la base escanea toda la tabla para consultar, hay índices únicos [garantizan valores distintos], índices filtrados [solo indexan una parte de la tabla], índices de columnas incluidas [para cubrir consultas sin agregar al índice principal], cada uno implementado para su caso particular.

Lo primero que consideramos es la información que queremos copiar, si la estructura o los datos, o ambos, en nuestro caso nos enfocamos más en los datos que en la estructura ya que habríamos de implementar

métodos para mejorar las consultas, enfocándonos en el uso de SELECT INTO nos permitió tener una DIVIOS EXACTA en tablas “planas” y tomando en cuenta las columnas donde más se usan en condicionales (where, join, orden by, grup by), las que buscan igualdades, rangos o listas y observando la proporción de datos obtenidos en relación a los datos totales es que consideramos la implementación de los ÍNDICES y su orden. Otras consideraciones para implementar índices son las columnas que se usan con frecuencia, las que tienen alta cardinalidad (por que permiten filtrar más fácilmente) y evitar columnas booleanas o con pocos valores distintos.

En cuanto a los planes de ejecución hemos observado que debemos reducir los “table scan” y buscar más “index seek”, en cuanto al filtrado columnas con muchos valores distintos ayudan a filtrar mejor, evitar usar ORDEN BY y mejor implementar un índice ordenado. Lo que deberíamos buscar para mejorar el plan de ejecución son; “hash match”, “sort”, “spool”, “computer scalar”, “key lookup” ya que son señales de que el motor está trabajando más de lo necesario.

Aunque el uso de índices es idóneo, siempre se debe volver a considerar reescribir las consultas pensándolo como una transformación en etapas

I. From .- ¿de dónde vienen los datos?
II. Where.- ¿qué filas se necesitan?
III. Group by .- ¿cómo agruparlos?
IV. Having.- ¿qué grupos son de interés?
V. Selecy.- ¿qué columnas se quieren mostrar?
VI. Filtrar lo antes posible. - cuantos menos datos pasen a etapas posteriores, más rápido será el procesamiento
VII. Usar alias claros y consistentes
VIII. Documentar las consultas

Con esta práctica se permitió el acercarnos mas a las elecciones de índices y como estos afectan en los planes de ejecución como aun cuando esperas que cambien de alguna manera estos pueden cambiar de una manera completamente diferente, ya sea para mejor o para peor también al comparar con otros equipos se pudo apreciar como par una misma consulta se puede cambiar de índice y el porque elegirlo dependiendo del enfoque que le de la persona o los costos que desee mejorar.