Instituto Tecnológico de Nuevo Laredo

INGENIERIA EN SISTEMAS COMPUTACIONALES

TALLER DE BASE DE DATOS

Reporte Unidad 2

Iván Rosales Cancino 20100259

Guillermo Del Bosque Garcia 20100184

Ing. Humberto Peña Valle

Noviembre de 2024

Ave. Reforma #2007 Sur, Col. Fundadores, C.P. 88275, Nuevo Laredo, Tam.
Tels. (867) 7119061, Conmut. 7119050 Ext. 111.
e-mail: desarrolloacademico@nlaredo.tecnm.mx
www.tecnm.mx | www.nlaredo.tecnm.mx

Índice

- 6.9 REPALCE
- 6.10 RIGHT
- 6.11 RTRIM
- 6.12 SUBSTRING
- 6.13 UPPER

7. Funciones de agregación
   - 7.1 COUNT
   - 7.2 SUM
   - 7.3 AVG
   - 7.4 MIN
   - 7.5 MAX
   - 7.6 STDEV
   - 7.7 STDEVP
   - 7.8 VAR
   - 7.9 VARP
8. SELECT
   - 8.1 Todos los campos
   - 8.2 Campos específicos
   - 8.3 Renombrar encabezados de las columnas
   - 8.4 Valores sin repetición
   - 8.5 Primeros n registros
   - 8.6 Columnas calculadas
   - 8.7 Constantes
   - 8.8 Uso de calificadores
   - 8.9 Ordenamiento
     - 8.9.1 Ascendente
     - 8.9.2 Descendente
   - 8.10 Decisiones antes de proyectar
     - 8.10.1 CASE Simple
     - 8.10.2 CASE compuesto
   - 8.11 Funciones de agregación
   - 8.12 Agrupación de subconjuntos
   - 8.13 Unión de consultas
   - 8.14 WHERE
     - 8.14.1 Operadores relacionados básicos
     - 8.14.2 Operadores lógicos
       - 8.14.2.1 AND
       - 8.14.2.2 OR
       - 8.14.2.3 NOT
     - 8.14.3 Operadores relacionales especiales
       - 8.14.3.1 IS
       - 8.14.3.2 IN
       - 8.14.3.3 BETWEEN

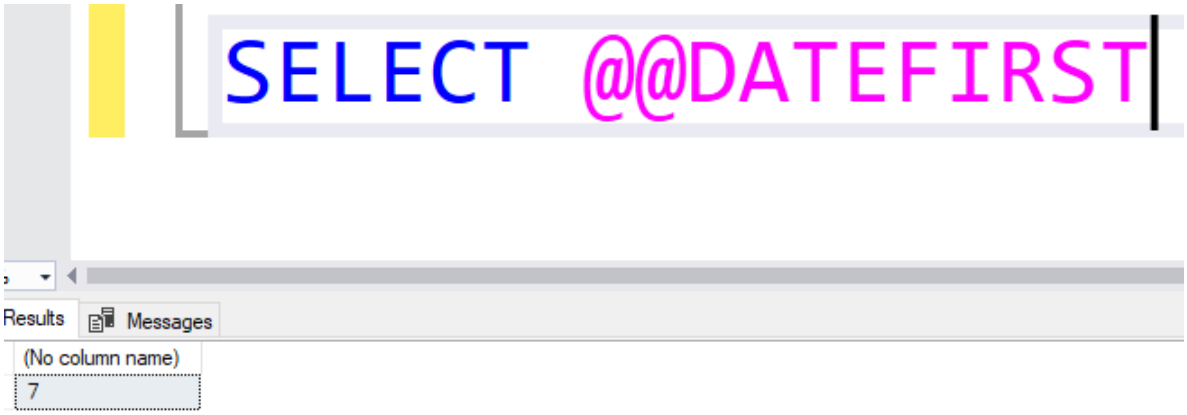
1. Introducción

En esta unidad se hará conocer sobre el uso de todas las funciones que contiene las bases de datos que nos permitirá agrupar, calcular, agregar, modificar entre tablas de una base de datos, al igual que aprenderemos agregar y eliminar datos en unas tablas de una base de datos, hay que tener en cuenta que necesitamos saber analizar los datos que nos piden para realizar alguna búsqueda especifica en la base de datos para comprobar si somos capaces de realizar cierta petición.

2. Funciones escalares

2.1 @@DATEFIRST

Determina el primer día de la semana para la configuración regional actual, por ejemplo, utilizando únicamente esta función, nos devolverá el número 7.



Otro ejemplo puede ser que queremos calcular el número de ventas realizadas en cada día de la semana, pero queremos que la semana comience el lunes en lugar del domingo (ya que el domingo es el primer día determinado).

```
SET DATEFIRST 1

SELECT
    DATEPART(WEEKDAY, OrderDate) AS DiaSemana,
    DATENAME(WEEKDAY, OrderDate) AS NombreDiaSemana,
    COUNT(*) AS VentasRealizadas
FROM
    Orders
GROUP BY
    DATEPART(WEEKDAY, OrderDate), DATENAME(WEEKDAY, OrderDate)
ORDER BY
    DiaSemana
```

	DiaSemana	NombreDiaSemana	VentasRealizadas
1	1	Monday	165
2	2	Tuesday	168
3	3	Wednesday	165
4	4	Thursday	168
5	5	Friday	164

2.2 @@LANGID

Este almacena el identificador de idioma predeterminado configurado en el servidor de base de datos. Se utiliza para determinar cómo se manejan ciertas operaciones relacionadas con el lenguaje, como el ordenamiento y la conversión de datos.

Ejemplo de su función:

```
SELECT @@LANGID AS 'LangID'
```

	LangID
1	0

En mi caso que no tengo algún idioma predeterminado, me devuelve 0.

Ejemplo con Northwind:

```
SET LANGUAGE Spanish

SELECT ProductName
FROM Products
ORDER BY ProductName
```

	ProductName
1	Alice Mutton
2	Aniseed Syrup
3	Boston Crab Meat
4	Camembert Pierrot
5	Camarvon Tigers
6	Chai
7	Chang
8	Chartreuse verte
9	Chef Anton's Cajun Seasoning

2.3 @@LANGUAGE

Determina el idioma actual establecido en la sesión de SQL server. Esto puede afectar cómo se presentan los resultados de las consultas, como el formato de fechas y la clasificación de cadenas.

Ejemplo de su uso:

```
SELECT @@LANGUAGE AS 'Language'
```

	Language
1	Español

Ejemplo con Northwind:

```
SET LANGUAGE Spanish

SELECT ProductName
FROM Products
```

	ProductName
1	Alice Mutton
2	Aniseed Syrup
3	Boston Crab Meat
4	Camembert Pierrot
5	Camarvon Tigers
6	Chai
7	Chang
8	Chartreuse verte

2.4 @@SERVERNAME

Devuelve el nombre del servidor de bases de datos actual, puede ser útil para identificar el servidor al que está conectada la sesión de SQL Server.

Ejemplo de su uso y con Northwind:

```
USE Northwind
SELECT @@SERVERNAME AS 'ServerName'
```

	ServerName
1	DESKTOP-M35LLQK

2.5 @@SERVICENAME

Devuelve el nombre del servicio de SQL Server en el sistema operativo, puede ser útil para identificar el servicio de SQL Server que está en funcionamiento en el sistema.

Ejemplo de su uso y con Northwind:

```
USE Northwind
SELECT @@SERVICENAME AS 'ServiceName'
```

	ServiceName
1	MSSQLSERVER

2.6 @@VERSION

Devuelve información detallada sobre la versión de SQL Server que se está ejecutando, incluye el número de versión, la edición, la compilación y otros detalles.

Ejemplo de su uso y con Northwind:

```
USE Northwind
SELECT @@VERSION AS 'Version'
```

	Version
1	Microsoft SQL Server 2022 (RTM) - 16.0.1000.6 (X...

3. Funciones de conversión

3.1 CAST

Se utiliza para convertir un valor de un tipo de datos a otro tipo de datos. Esto puede ser útil cuando se necesita realizar operaciones con valores de diferentes tipos o cuando se desea cambiar el formato de presentación de un valor.

CAST convierte el valor especificado al tipo de datos especificado, es útil cuando se necesita convertir un valor de un tipo de datos a otro tipo de datos compatible.

Ejemplo de conversión de tipo de datos:

```
SELECT CAST('123' AS INT) AS Numerico
```

	Numerico
1	123

Ejemplo de cambio de formato de fecha:

```
SELECT CAST(GETDATE() AS DATE) AS Fecha
```

	Fecha
1	2024-04-14

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductID, ProductName, CAST(UnitPrice AS DECIMAL(10,2)) AS UnitPriceDecimal
FROM Products
WHERE ProductID = 1
```

	ProductID	ProductName	UnitPriceDecimal
1	1	Chai	18.00

En este ejemplo, estamos convirtiendo el precio unitario (UnitPrice) de la tabla Products en la base de datos Northwind de VARCHAR a DECIMAL (10,2) utilizando la función CAST.

La función CAST es útil para realizar conversiones de tipo de datos de manera explícita y garantizar la compatibilidad de tipos en las operaciones realizadas en SQL Server.

3.2 CONVERT

Se utiliza para convertir un valor de un tipo de datos a otro tipo de datos. Al igual que CAST, CONVERT es útil cuando se necesita realizar operaciones con valores de diferentes tipos o cuando se desea cambiar el formato de presentación de un valor.

CONVERT convierte el valor especificado al tipo de datos especificado, además de la conversión de tipo de datos, CONVERT también puede aceptar un argumento opcional de estilo que controla el formato de la conversión, como el formato de fecha y hora.

Ejemplo de conversión de tipo de datos:

```
SELECT CONVERT(INT, '123') AS Numerico
```

	Numerico
1	123

Ejemplo de cambio de formato de fecha:

```
SELECT CONVERT(DATE, GETDATE()) AS Fecha
```

	Fecha
1	2024-04-14

Ejemplo con Northwind:

```
USE Northwind
SELECT OrderID, OrderDate, CONVERT(VARCHAR, OrderDate, 103) AS OrderDateFormatted
FROM Orders
WHERE OrderID = 10248
```

	OrderID	OrderDate	OrderDateFormatted
1	10248	1996-07-04 00:00:00.000	04/07/1996

En este ejemplo, estamos cambiando el formato de la fecha de la columna OrderDate de la tabla Orders en la base de datos Northwind utilizando la función CONVERT para que esté en formato DD/MM/YYYY.

3.3 PARCE

Se utiliza para convertir una cadena de caracteres en un tipo de datos de fecha y hora. Proporciona una forma más flexible de analizar y convertir cadenas de fecha en comparación con las funciones CAST y CONVERT.

PARSE analiza la cadena especificada y la convierte en el tipo de datos especificado, puede ser especialmente útil cuando se trabaja con formatos de fecha no estándar o cuando se necesita especificar la cultura para el análisis de la cadena.

Ejemplo de conversión de cadena a fecha:

```
SELECT PARSE('2022-04-19' AS DATE) AS Fecha
```

	Fecha
1	2022-04-19

En este ejemplo, estamos convirtiendo la cadena '2022-04-19' en una fecha (tipo de datos DATE). El resultado será la fecha representada por la cadena.

3.4 TRY_CAST

Se utiliza para intentar convertir un valor a un tipo de datos específico. Si la conversión es exitosa, devuelve el valor convertido; de lo contrario, devuelve NULL. Esto evita que se produzcan errores de conversión y permite manejarlos de manera más controlada en caso de fallo.

TRY_CAST intenta convertir el valor especificado al tipo de datos especificado, si la conversión es exitosa, devuelve el valor convertido; de lo contrario, devuelve NULL. Es útil para evitar errores de conversión y manejarlos de manera controlada.

Ejemplo de uso:

```
SELECT TRY_CAST('123' AS INT) AS Numerico
```

	Numerico
1	123

En este ejemplo, estamos intentando convertir la cadena '123' a un valor numérico (tipo de datos INT). Si la conversión es exitosa, el resultado será el número entero 123; de lo contrario, se devolverá NULL.

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductID, ProductName, TRY_CAST(UnitPrice AS DECIMAL(10,2)) AS UnitPriceDecimal
FROM Products
WHERE ProductID = 1
```

	ProductID	ProductName	UnitPriceDecimal
1	1	Chai	18.00

En este ejemplo, estamos utilizando TRY_CAST en la base de datos Northwind para intentar convertir el precio unitario (UnitPrice) de la tabla Products de tipo VARCHAR a DECIMAL (10,2). Si la conversión es exitosa, se devolverá el precio unitario convertido; de lo contrario, se devolverá NULL. Esto evita que se produzcan errores de conversión y permite manejarlos de manera controlada.

3.5 TRY_CONVERT

Se utilizar para intentar convertir un valor a un tipo de datos específico. La diferencia principal entre esta función y la función CONVERT es que, si la conversión falla, en lugar de arrojar un error, devolverán NULL.

TRY_CONVERT intenta convertir un valor a un tipo de datos específico y devuelve NULL si la conversión falla.

Ejemplo de uso:

```
SELECT TRY_CONVERT(INT, 'abc') AS ConvertedValue
```

	ConvertedValue
1	NULL

En este ejemplo, estamos intentando convertir la cadena 'abc' a un valor entero (INT). Como la conversión no es posible, la función TRY_CONVERT devuelve NULL.

Ejemplo con Northwind:

```
USE Northwind;
SELECT ProductID, ProductName, TRY_CONVERT(DECIMAL(10,2), UnitPrice) AS UnitPriceDecimal
FROM Products
WHERE ProductID = 1;
```

	ProductID	ProductName	UnitPriceDecimal
1	1	Chai	18.00

Aquí estamos usando TRY_CONVERT en la base de datos Northwind para intentar convertir el precio unitario (UnitPrice) de la tabla Products a tipo numérico (DECIMAL (10,2)). Si la conversión falla, se devolverá NULL.

3.6 TRY_PARCE

Se utilizan para intentar convertir un valor a un tipo de datos específico. La diferencia principal entre esta función y la función PARSE es que, si la conversión falla, en lugar de arrojar un error, devolverán NULL.

TRY_PARSE intenta analizar una cadena y convertirla a un tipo de datos de fecha y hora específico, devolviendo NULL si la conversión falla.

Ejemplo de uso:

```
SELECT TRY_PARSE('2022-04-abc' AS DATE) AS ParsedDate
```

	ParsedDate
1	NULL

En este ejemplo, estamos intentando analizar la cadena '2022-04-abc' como una fecha (DATE). Como la conversión no es posible, la función TRY_PARSE devuelve NULL.

4. Funciones de fecha y hora

4.1 GETDATE

Devuelve la fecha y hora actuales del sistema en el que se está ejecutando la consulta. Es útil para obtener la fecha y hora actual en consultas SQL y puede ser utilizada en diversas situaciones, como al insertar registros en una tabla con marca de tiempo o al realizar cálculos basados en la fecha y hora actual.

GETDATE devuelve la fecha y hora actuales del sistema en el que se está ejecutando la consulta, no toma ningún argumento y siempre devuelve la fecha y hora actual.

Ejemplo de uso:

```
SELECT GETDATE() AS FechaYHoraActual
```

	FechaYHoraActual
1	2024-04-14 21:35:43.850

Este ejemplo devolverá la fecha y hora actuales del sistema en el formato predeterminado de SQL Server.

Ejemplo con Northwind:

```
SELECT 'Hola', 1,  GETDATE(), 26+25, FIRSTNAME FROM Employees
```

	(No column name)	(No column name)	(No column name)	(No column name)	FIRSTNAME
1	Hola	1	2024-04-14 21:39:46.500	51	Nancy
2	Hola	1	2024-04-14 21:39:46.500	51	Andrew
3	Hola	1	2024-04-14 21:39:46.500	51	Janet
4	Hola	1	2024-04-14 21:39:46.500	51	Margaret
5	Hola	1	2024-04-14 21:39:46.500	51	Steven

En este ejemplo, sacamos las fechas de la tabla empleados.

4.2 DATENAME

Se utiliza para extraer partes específicas (como el nombre del mes, el nombre del día de la semana, etc.) de una fecha o hora dada.

DATENAME devuelve una cadena que representa la parte especificada de una fecha u hora. La parte puede ser 'year', 'quarter', 'month', 'day', 'weekday', 'week', 'hour', 'minute' o 'second'.

Ejemplo de uso:

```sql
SELECT DATENAME(month, GETDATE()) AS MesActual;
```

	MesActual
1	April

En este ejemplo, estamos obteniendo el nombre del mes actual (parte 'month') utilizando la función GETDATE() para obtener la fecha y hora actual del sistema.

Ejemplo con Northwind:

```sql
USE Northwind;
SELECT OrderID, OrderDate, DATENAME(month, OrderDate) AS NombreMes
FROM Orders
WHERE OrderID = 10248;
```

	OrderID	OrderDate	NombreMes
1	10248	1996-07-04 00:00:00.000	July

En este ejemplo, estamos extrayendo el nombre del mes de la columna OrderDate de la tabla Orders en la base de datos Northwind. Esto nos permite ver el nombre del mes en lugar de la fecha completa, lo que puede ser útil para ciertos informes o análisis.

4.3 DATEPART

Se utiliza para extraer una parte específica de una fecha o una hora, como el año, el mes, el día, la hora, el minuto, etc. Es útil cuando necesitas realizar operaciones basadas en componentes individuales de una fecha o una hora.

DATEPART devuelve un entero que representa la parte especificada de una fecha o una hora.

Ejemplo de uso:

```sql
SELECT DATEPART(year, '2024-04-19') AS Año;
```

	Año
1	2024

En este ejemplo, estamos utilizando DATEPART para extraer el año de la fecha '2024-04-19'. El resultado será el año 2024.

Ejemplo con Northwind:

```sql
USE Northwind;
SELECT OrderID, OrderDate, DATEPART(month, OrderDate) AS Mes
FROM Orders
WHERE OrderID = 10248;
```

	OrderID	OrderDate	Mes
1	10248	1996-07-04 00:00:00.000	7

En este ejemplo, estamos utilizando DATEPART para extraer el mes de la columna OrderDate de la tabla Orders en la base de datos Northwind. Esto nos permite obtener el mes de la fecha de pedido para el pedido con el ID 10248.

4.4 DAY

Se utiliza para extraer el día de una fecha dada. Es útil cuando solo se necesita el día de una fecha sin tener en cuenta el mes y el año, devuelve el día de la fecha especificada. El resultado será un entero que representa el día del mes (del 1 al 31).

Ejemplo de uso:

```
SELECT DAY(GETDATE()) AS Dia;
```

	Dia
1	14

Este ejemplo devolverá el día actual del mes.

Ejemplo con Northwind:

```
USE Northwind;
SELECT OrderID, OrderDate, DAY(OrderDate) AS DiaPedido
FROM Orders
WHERE OrderID = 10248;
```

	OrderID	OrderDate	DiaPedido
1	10248	1996-07-04 00:00:00.000	4

En este ejemplo, estamos extrayendo el día del mes de la columna OrderDate de la tabla Orders en la base de datos Northwind para un pedido específico (identificado por OrderID). Esto nos proporciona el día en que se realizó el pedido.

4.5 MONTH

Se utiliza para extraer el componente de mes de una fecha dada. Devuelve un valor entero que representa el mes (del 1 al 12) de la fecha especificada.

MONTH devuelve el componente de mes de la fecha especificada. El valor devuelto es un entero que representa el mes (del 1 al 12) de la fecha especificada.

Ejemplo de uso:

```
SELECT MONTH(GETDATE()) AS MesActual
```

	MesActual
1	4

En este ejemplo, estamos obteniendo el componente de mes de la fecha y hora actuales utilizando la función GETDATE().

Ejemplo con Northwind:

```sql
USE Northwind;
SELECT OrderID, OrderDate, MONTH(OrderDate) AS MesPedido
FROM Orders
WHERE OrderID = 10248;
```

	OrderID	OrderDate	MesPedido
1	10248	1996-07-04 00:00:00.000	7

En este ejemplo, estamos obteniendo el componente de mes de la columna OrderDate de la tabla Orders en la base de datos Northwind para el pedido con OrderID igual a 10248. Esto nos permite ver en qué mes se realizó el pedido.

4.6 YEAR

Se utiliza para extraer el año de una fecha especificada. Esto es útil cuando se necesita obtener solo el año de una fecha para realizar operaciones o análisis específicos basados en el año.

YEAR devuelve el año de la fecha especificada. Es útil cuando se necesita obtener el año de una fecha para realizar operaciones o análisis basados en el año.

Ejemplo de uso:

```sql
SELECT YEAR(GETDATE()) AS 'AñoActual'
```

	AñoActual
1	2024

Este ejemplo devolverá el año actual utilizando la función YEAR con la función GETDATE() para obtener la fecha y hora actuales.

Ejemplo con Northwind:

```sql
USE Northwind;
SELECT OrderID, ShippedDate, YEAR(ShippedDate) AS 'AñoEnvío'
FROM Orders
WHERE ShippedDate IS NOT NULL;
```

	OrderID	ShippedDate	AñoEnvio
1	10249	1996-07-10 00:00:00.000	1996
2	10252	1996-07-11 00:00:00.000	1996
3	10250	1996-07-12 00:00:00.000	1996
4	10251	1996-07-15 00:00:00.000	1996
5	10255	1996-07-15 00:00:00.000	1996
6	10248	1996-07-16 00:00:00.000	1996
7	10253	1996-07-16 00:00:00.000	1996

En este ejemplo, estamos utilizando la función YEAR para obtener el año de la columna ShippedDate de la tabla Orders en la base de datos Northwind. Esto nos

permite analizar los datos de envío basados en el año en que se enviaron los pedidos.

4.7 DATEFROMPARTS

Se utiliza para construir una fecha a partir de sus partes individuales, como el año, el mes y el día. Proporciona una forma conveniente de crear fechas a partir de valores numéricos.

DATEFROMPARTS crea una fecha a partir de las partes específicas proporcionadas (año, mes, día). Esta función es útil cuando se necesita construir una fecha utilizando valores numéricos para el año, mes y día.

Ejemplo de uso:

```
SELECT DATEFROMPARTS(2022, 4, 19) AS Fecha;
```

	Fecha
1	2022-04-19

En este ejemplo, estamos creando una fecha con el año 2022, el mes 4 y el día 19 utilizando la función DATEFROMPARTS.

Ejemplo con Northwind:

```
USE Northwind
SELECT OrderID, DATEFROMPARTS(YEAR(OrderDate), MONTH(OrderDate), DAY(OrderDate)) AS OrderDateConstructed
FROM Orders
WHERE OrderID = 10248
```

	OrderID	OrderDateConstructed
1	10248	1996-07-04

En este ejemplo, estamos utilizando DATEFROMPARTS en la base de datos Northwind para construir una fecha a partir de las partes del año, mes y día de la columna OrderDate de la tabla Orders. Esto nos permite crear una nueva columna llamada OrderDateConstructed que contiene las fechas construidas utilizando DATEFROMPARTS. Esto puede ser útil cuando necesitamos realizar operaciones o comparaciones con fechas utilizando partes individuales de las fechas existentes en la base de datos.

4.8 TIMEFROMPARTS

Se utiliza para construir un valor de tipo TIME utilizando sus partes individuales, como horas, minutos, segundos y milisegundos. Esta función es útil cuando se necesita crear manualmente un valor de tiempo a partir de sus componentes individuales.

TIMEFROMPARTS (hora, minuto, segundo, milisegundo, precisión) crea un valor de tipo TIME a partir de sus partes individuales. Los argumentos hora, minuto, segundo y milisegundo son enteros que representan las partes del tiempo.

Ejemplo de uso:

```
SELECT TIMEFROMPARTS(12, 30, 45, 500, 3) AS Tiempo;
```

	Tiempo
1	12:30:45.500

En este ejemplo, estamos utilizando TIMEFROMPARTS para crear un valor de tiempo que representa las 12 horas, 30 minutos, 45 segundos y 500 milisegundos, con una precisión de 3 dígitos fraccionarios.

Ejemplo con Northwind:

```
USE Northwind
SELECT OrderID, ShippedDate, TIMEFROMPARTS(DATEPART(HOUR, ShippedDate), DATEPART(MINUTE, ShippedDate),
DATEPART(SECOND, ShippedDate), 0, 0) AS TiempoEnvio
FROM Orders
WHERE OrderID = 10248
```

	OrderID	ShippedDate	TiempoEnvio
1	10248	1996-07-16 00:00:00.000	00:00:00

En este ejemplo, estamos utilizando TIMEFROMPARTS junto con funciones de fecha y hora (DATEPART) para obtener el tiempo de envío de un pedido específico en la base de datos Northwind. Esto nos permite construir un valor de tiempo a partir de las partes individuales de la columna ShippedDate.

4.9 DATEADD

Se utiliza para agregar una cantidad especificada de intervalos de tiempo a una fecha dada. Esto puede ser útil para calcular fechas futuras o pasadas basadas en una fecha inicial.

DATEADD agrega una cantidad especificada de intervalos de tiempo a una fecha. El argumento intervalo especifica el tipo de intervalo de tiempo que se va a agregar (por ejemplo, day, month, year, etc.). El argumento número especifica la cantidad de intervalos que se van a agregar. El argumento fecha es la fecha inicial a la que se le van a agregar los intervalos.

Ejemplo de uso:

```
SELECT DATEADD(day, 30, GETDATE()) AS FechaFutura
```

	FechaFutura
1	2024-05-15 07:10:17.507

Este ejemplo agrega 30 días a la fecha y hora actuales (obtenida mediante GETDATE ()) y devuelve la fecha resultante.

Ejemplo con Northwind:

```
USE Northwind
SELECT OrderID, OrderDate, DATEADD(day, 7, OrderDate) AS FechaEntrega
FROM Orders
WHERE OrderID = 10248
```

	OrderID	OrderDate	FechaEntrega
1	10248	1996-07-04 00:00:00.000	1996-07-11 00:00:00.000

En este ejemplo, estamos calculando la fecha de entrega agregando 7 días a la fecha de pedido (OrderDate) en la tabla Orders de la base de datos Northwind. Esto nos da la fecha en la que se espera que se entregue el pedido, basada en la fecha de pedido original.

4.10 EOMONTH

Se utiliza para obtener la última fecha del mes de una fecha dada. Es útil cuando necesitas calcular la fecha de finalización de un mes para una fecha específica.

EOMONTH (fecha, [meses]) devuelve la última fecha del mes de la fecha especificada. El parámetro opcional [meses] te permite especificar un número de meses adicionales para avanzar o retroceder desde la fecha dada antes de calcular el final del mes.

Ejemplo de uso:

```
SELECT EOMONTH(GETDATE()) AS UltimoDiaMesActual;
```

	UltimoDiaMesActual
1	2024-04-30

En este ejemplo, estamos utilizando GETDATE () para obtener la fecha actual y luego aplicamos la función EOMONTH para obtener la última fecha del mes actual.

Ejemplo con Northwind:

```
USE Northwind
SELECT OrderID, ShippedDate, EOMONTH(ShippedDate) AS UltimoDiaMesEnvio
FROM Orders
WHERE OrderID = 10248
```

	OrderID	ShippedDate	UltimoDiaMesEnvio
1	10248	1996-07-16 00:00:00.000	1996-07-31

En este ejemplo, estamos seleccionando el ID del pedido (OrderID), la fecha de envío (ShippedDate) y utilizando la función EOMONTH para calcular la última fecha del mes de la fecha de envío para un pedido específico (en este caso, el pedido con OrderID igual a 10248) en la base de datos Northwind. Esto nos permite obtener la fecha de finalización del mes en la que se realizó el envío del pedido.

5. Funciones matemáticas

5.1 ABS

Se utiliza para devolver el valor absoluto de un número, es decir, su valor sin tener en cuenta el signo.

ABS devuelve el valor absoluto del número especificado. Es útil cuando se necesita el valor numérico sin tener en cuenta su signo.

Ejemplo de uso:

```
SELECT ABS(-10) AS ValorAbsoluto
```

	ValorAbsoluto
1	10

En este ejemplo, la función ABS devuelve el valor absoluto de -10, que es 10.

Ejemplo con Northwind:

```
USE Northwind
SELECT OrderID, Discount, ABS(Discount) AS DiscountAbsoluto
FROM [Order Details]
WHERE OrderID = 10248
```

	OrderID	Discount	DiscountAbsoluto
1	10248	0	0
2	10248	0	0
3	10248	0	0

En este ejemplo, estamos utilizando la función ABS en la columna Discount de la tabla [Order Details] en la base de datos Northwind. Esto nos permite obtener el valor absoluto del descuento aplicado en una orden específica (OrderID = 10248), independientemente de si el descuento es positivo o negativo.

5.2 CEILING

Se utiliza para redondear un número hacia arriba al entero más cercano. Si el número ya es un entero, CEILING no tiene ningún efecto sobre él.

CEILING (numero) devuelve el entero más pequeño mayor o igual que el número especificado. Es útil cuando necesitas redondear un número hacia arriba al entero más cercano.

Ejemplo de uso:

```
SELECT CEILING(12.345) AS Resultado
```

	Resultado
1	13

En este ejemplo, el número 12.345 se redondea hacia arriba al entero más cercano, que es 13.

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductName, UnitPrice, CEILING(UnitPrice) AS UnitPriceRoundedUp
FROM Products
WHERE ProductID = 1
```

	ProductName	UnitPrice	UnitPriceRoundedUp
1	Chai	18.00	18.00

En este ejemplo, estamos utilizando CEILING en la base de datos Northwind para redondear hacia arriba el precio unitario (UnitPrice) del producto con ProductID igual a 1. Esto nos permite obtener el precio unitario redondeado hacia arriba al entero más cercano.

5.3 FLOOR

Se utiliza para redondear un número hacia abajo al entero más cercano. Es decir, elimina la parte decimal de un número y devuelve el entero resultante igual o menor al número original.

FLOOR (valor) redondea hacia abajo el valor especificado al entero más cercano. Devuelve el entero igual o menor al valor original, eliminando la parte decimal.

Ejemplo de uso:

```
SELECT FLOOR(3.7) AS Resultado;
```

	Resultado
1	3

En este ejemplo, FLOOR (3.7) devuelve 3, ya que redondea hacia abajo el número 3.7 al entero más cercano.

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductName, UnitPrice, FLOOR(UnitPrice) AS UnitPriceFloor
FROM Products
WHERE CategoryID = 1
```

	ProductName	UnitPrice	UnitPriceFloor
1	Chai	18.00	18.00
2	Chang	19.00	19.00
3	Guaraná Fantástica	4.50	4.00
4	Sasquatch Ale	14.00	14.00
5	Steeleye Stout	18.00	18.00
6	Côte de Blaye	263.50	263.00

En este ejemplo, estamos seleccionando el nombre del producto, el precio unitario original y el precio unitario redondeado hacia abajo (UnitPriceFloor). Esto nos permite ver cómo se redondean los precios unitarios de los productos hacia abajo en la categoría especificada (en este caso, la categoría con CategoryID = 1) en la base de datos Northwind.

5.4 POWER

Se utiliza para calcular una potencia específica de un número. Permite elevar un número a una potencia determinada.

POWER (valor, exponente) calcula el valor de elevar el número especificado a la potencia del exponente especificado.

Ejemplo de uso:

```
SELECT POWER(2, 3) AS Resultado
```

	Resultado
1	8

Este ejemplo devolverá el resultado de elevar el número 2 a la potencia de 3, que es 8.

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductID, ProductName, UnitPrice, POWER(UnitPrice, 2) AS PrecioCuadrado
FROM Products
WHERE ProductID = 1
```

	ProductID	ProductName	UnitPrice	PrecioCuadrado
1	1	Chai	18.00	324.00

En este ejemplo, estamos calculando el cuadrado del precio unitario (UnitPrice) de un producto en la tabla Products de la base de datos Northwind utilizando la función POWER. Esto nos dará el precio unitario elevado al cuadrado para el producto con ProductID igual a 1.

5.5 SQRT

Se utiliza para calcular la raíz cuadrada de un número. calcula la raíz cuadrada del número especificado.

Ejemplo de uso:

```
SELECT SQRT(25) AS RaizCuadrada
```

	RaizCuadrada
1	5

Este ejemplo calculará la raíz cuadrada del número 25, que es 5.

Ejemplo con Northwind:

```
USE Northwind
SELECT SQRT(COUNT(*)) AS RaizCuadradaProductos
FROM Products
```

	RaizCuadradaProductos
1	8.77496438739212

6. Funciones de cadena

6.1 ASCII

Se utiliza para devolver el valor ASCII del primer carácter de una cadena. Este valor ASCII representa el código numérico asociado al carácter en la tabla ASCII estándar.

ASCII (cadena) devuelve el valor ASCII del primer carácter de la cadena especificada. Es útil cuando se necesita trabajar con el código ASCII de un carácter en particular.

Ejemplo de uso:

```
SELECT ASCII('A') AS ValorASCII
```

	ValorASCII
1	65

En este ejemplo, estamos utilizando la función ASCII para obtener el valor ASCII del carácter 'A'. El resultado será 65, que es el valor ASCII para la letra 'A'.

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductID, ProductName, ASCII(LEFT(ProductName, 1)) AS PrimerCaracterASCII
FROM Products
WHERE ProductID = 1
```

	ProductID	ProductName	PrimerCaracterASCII
1	1	Chai	67

En este ejemplo, estamos utilizando la función ASCII en la base de datos Northwind para obtener el valor ASCII del primer carácter del nombre del producto. La función LEFT se utiliza para extraer el primer carácter de la cadena ProductName. Esto nos permite obtener el valor ASCII del primer carácter de los nombres de los productos en la tabla Products.

6.2 CHAR

Se utiliza para devolver el carácter correspondiente al valor entero especificado. Puede ser útil para generar caracteres específicos o para realizar manipulaciones de cadenas de caracteres.

CHAR(n) devuelve el carácter correspondiente al valor entero especificado por n. “n” debe ser un valor entero entre 0 y 255.

Ejemplo de uso:

```
SELECT CHAR(65) AS Character
```

	Character
1	A

En este ejemplo, estamos utilizando la función CHAR para devolver el carácter correspondiente al valor entero 65, que es la letra "A".

6.3 CHARINDEX

Se utiliza para encontrar la posición de una cadena de texto dentro de otra cadena. Devuelve la posición inicial de la primera aparición de la cadena buscada en la cadena de texto, o 0 si la cadena buscada no se encuentra.

CHARINDEX (subcadena, cadena[, inicio]) busca la subcadena dentro de la cadena y devuelve la posición inicial de la primera aparición de la subcadena en la cadena.

Ejemplo de uso:

```
DECLARE @cadena NVARCHAR(50) = 'Hello World';
DECLARE @subcadena NVARCHAR(20) = 'World';
SELECT CHARINDEX(@subcadena, @cadena) AS Posicion;
```

	Posicion
1	7

En este ejemplo, estamos buscando la posición de la subcadena 'World' dentro de la cadena 'Hello World'. La función CHARINDEX devuelve la posición 7, ya que 'World' comienza en la séptima posición de la cadena.

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductID, ProductName, CHARINDEX('$', UnitPrice) AS PosicionMoneda
FROM Products
WHERE CHARINDEX('$', UnitPrice) > 0
```

	ProductID	ProductName	PosicionMoneda

En este ejemplo, estamos utilizando CHARINDEX para buscar la posición del símbolo de moneda ('$') en la columna UnitPrice de la tabla Products en la base de datos Northwind. Esto nos ayuda a identificar los productos cuyos precios están

expresados en dólares. La condición WHERE se asegura de que solo se devuelvan los productos cuyos precios contienen el símbolo de moneda.

6.4 FORMAT

Se utiliza para formatear valores de fecha y hora, así como valores numéricos y de cadena, en una representación específica de formato de caracteres. Esto proporciona flexibilidad para presentar datos en diferentes formatos según los requisitos de presentación.

FORMAT (valor, formato) formatea el valor especificado según el formato de presentación especificado. El formato puede ser una cadena de formato estándar predefinida o una cadena de formato personalizada.

Ejemplo de uso:

```
SELECT FORMAT(GETDATE(), 'dd/MM/yyyy HH:mm:ss') AS FechaHoraFormateada
```

	FechaHoraFormateada
1	15/04/2024 08:58:10

En este ejemplo, estamos formateando la fecha y hora actual (obtenida mediante la función GETDATE ()) en un formato específico de fecha y hora (DD/MM/YYYY HH:mm:ss).

Ejemplo con Northwind:

```
USE Northwind
SELECT FORMAT(OrderDate, 'dd/MM/yyyy') AS OrderDateFormatted
FROM Orders
WHERE OrderID = 10248
```

	OrderDateFormatted
1	04/07/1996

En este ejemplo, estamos formateando la fecha de la columna OrderDate de la tabla Orders en la base de datos Northwind en un formato específico de fecha (DD/MM/YYYY). Esto nos permite presentar la fecha en el formato deseado sin cambiar el tipo de datos original.

6.5 LEFT

Se utiliza para devolver un número específico de caracteres desde el principio de una cadena de texto.

LEFT (cadena, longitud) devuelve los primeros longitud caracteres de la cadena especificada. Es útil cuando se necesita extraer una parte específica de una cadena, como un prefijo o un código de identificación.

Ejemplo de uso:

```
SELECT LEFT('Ejemplo de cadena', 3) AS Subcadena
```

	Subcadena
1	Eje

Da como resultado Eje.

Ejemplo con Northwind:

```
SELECT lEFT(LastName, 3) AS Letras FROM Employees
```

	Letras
1	Buc
2	Cal
3	Dav
4	Dod

En este ejemplo, utilizamos la funcion LEFT para la tabla de Employees con los datos de Firstname que son los nombres de los empleados, aquí se muestra las 3 inciales de cada empleado.

6.6 LEN

Se utiliza para obtener la longitud (número de caracteres) de una cadena de texto. Es útil para determinar la cantidad de caracteres en una cadena, lo que puede ser útil en diversas situaciones, como validar la longitud de una entrada de usuario o realizar cálculos basados en la longitud de una cadena.

LEN (cadena) devuelve la longitud de la cadena especificada.

Ejemplo de uso:

```
DECLARE @cadena VARCHAR(50) = 'Hola, mundo';
SELECT LEN(@cadena) AS LongitudCadena;
```

	LongitudCadena
1	11

En este ejemplo, la variable @cadena contiene la cadena 'Hola, mundo'. La función LEN se utiliza para obtener la longitud de esta cadena, que en este caso sería 11, ya que hay 11 caracteres en la cadena, incluidos los espacios y la coma.

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductID, ProductName, LEN(ProductName) AS LongitudNombre
FROM Products
WHERE ProductID = 1
```

	ProductID	ProductName	LongitudNombre
1	1	Chai	4

En este ejemplo, estamos utilizando la función LEN en la base de datos Northwind para obtener la longitud del nombre de un producto específico. Estamos seleccionando el ID del producto, el nombre del producto y la longitud del nombre del producto utilizando la función LEN en la tabla Products. Esto nos dará la longitud del nombre del producto en caracteres.

6.7 LOWER

Se utiliza para convertir una cadena de caracteres a minúsculas. Es útil cuando se desea comparar o buscar cadenas de caracteres sin importar si están escritas en mayúsculas o minúsculas.

LOWER (cadena) convierte todos los caracteres de una cadena de caracteres en minúsculas.

Ejemplo de uso:

```
SELECT LOWER('HELLO') AS LowerCaseString
```

	LowerCaseString
1	hello

Este ejemplo devolverá la cadena 'hello', ya que convierte todas las letras en mayúsculas a minúsculas.

Ejemplo con Northwind:

```
USE Northwind
SELECT LOWER(ContactName) AS LowerContactName
FROM Customers
WHERE Country = 'Germany'
```

	LowerContactName
1	maria anders
2	hanna moos
3	sven ottlieb
4	peter franken

En este ejemplo, estamos seleccionando el nombre de contacto (ContactName) de los clientes en Alemania de la tabla Customers en la base de datos Northwind y convirtiéndolos a minúsculas utilizando la función LOWER. Esto nos permite buscar clientes en Alemania sin importar si el nombre de contacto está escrito en mayúsculas o minúsculas.

6.8 LTRIM

Se utiliza para eliminar los espacios en blanco iniciales de una cadena de caracteres. Esto puede ser útil para limpiar y normalizar datos, especialmente cuando se trabaja con datos de texto almacenados en bases de datos.

LTRIM(cadena) devuelve la cadena de caracteres con los espacios en blanco iniciales eliminados.

Ejemplo de uso:

```
SELECT LTRIM('   Hello, world!   ') AS TrimmedString
```

	TrimmedString
1	Hello, world!

¡En este ejemplo, la función LTRIM elimina los espacios en blanco iniciales de la cadena ‘Hello, world! ', dejando 'Hello, world! ' como resultado.

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductName, LTRIM(ProductName) AS TrimmedProductName
FROM Products
WHERE ProductID = 1
```

	ProductName	TrimmedProductName
1	Chai	Chai

En este ejemplo, utilizamos LTRIM en la columna ProductName de la tabla Products en la base de datos Northwind para eliminar los espacios en blanco iniciales de los nombres de los productos. Esto puede ser útil para asegurar la coherencia en los datos y facilitar su análisis.

6.9 REPALCE

Se utiliza para reemplazar todas las ocurrencias de una subcadena dentro de una cadena de caracteres por otra subcadena especificada. Esta función es útil para realizar manipulaciones de cadenas, como cambiar un valor específico por otro en una columna.

REPLACE(cadena, subcadena_a_reemplazar, nueva_subcadena) reemplaza todas las ocurrencias de subcadena_a_reemplazar en cadena con nueva_subcadena.

Ejemplo de uso:

```
SELECT REPLACE('El cielo es rojo', 'rojo', 'azul') AS Resultado
```

	Resultado
1	El cielo es azul

En este ejemplo, estamos reemplazando todas las ocurrencias de la subcadena 'rojo' por 'azul' en la cadena 'El cielo es rojo'. El resultado será 'El cielo es azul'.

Ejemplo con Northwind:

```
USE Northwind
SELECT REPLACE(ShipName, 'Peacock', 'Leverling') AS NuevoShipName
FROM Orders
WHERE OrderID = 10248
```

	NuevoShipName
1	Vins et alcools Chevalier

En este ejemplo, estamos reemplazando todas las ocurrencias de la subcadena 'Peacock' por 'Leverling' en la columna ShipName de la tabla Orders en la base de datos Northwind. Esto nos permite actualizar valores en una columna específica según un criterio definido.

6.10 RIGHT

Se utiliza para devolver un número específico de caracteres desde el lado derecho de una cadena de caracteres. Es útil cuando se necesita extraer una parte específica de una cadena, como un sufijo o un código al final de una cadena.

RIGHT (cadena, n) devuelve los n caracteres desde el lado derecho de la cadena especificada.

Ejemplo de uso:

```
DECLARE @cadena VARCHAR(10) = 'Hola Mundo';
SELECT RIGHT(@cadena, 3) AS Sufijo;
```

	Sufijo
1	ndo

Ejemplo con Northwind:

```
USE Northwind
SELECT RIGHT(LastName, 3) AS Letras FROM Employees
```

	Letras
1	nan
2	han
3	lio

6.11 RTRIM

Se utiliza para eliminar los espacios en blanco del lado derecho de una cadena de caracteres. Esto puede ser útil para limpiar datos que puedan contener espacios en blanco adicionales al final.

RTRIM(valor) elimina los espacios en blanco del lado derecho de la cadena de caracteres especificada.

Ejemplo de uso:

```
SELECT RTRIM('Hello     ') AS TrimmedString;
```

	TrimmedString
1	Hello

En este ejemplo, la cadena 'Hello ' tiene espacios en blanco adicionales al final. La función RTRIM eliminará esos espacios en blanco, y el resultado será 'Hello'.

Ejemplo con Northwind:

```
USE Northwind
SELECT RTRIM(CompanyName) AS TrimmedCompanyName
FROM Customers
WHERE CustomerID = 'ALFKI'
```

	TrimmedCompanyName
1	Alfreds Futterkiste

En este ejemplo, estamos utilizando RTRIM en la columna CompanyName de la tabla Customers en la base de datos Northwind. Esto se hace para asegurarse de que cualquier espacio en blanco adicional al final del nombre de la empresa se elimine. Esto puede ser útil si los datos tienen inconsistencias en la entrada de datos y necesitamos limpiarlos para su posterior análisis.

6.12 SUBSTRING

Se utiliza para extraer una parte de una cadena de caracteres. Es útil cuando necesitas trabajar con partes específicas de una cadena, como extraer un subconjunto de caracteres o una subcadena de una cadena más larga.

SUBSTRING (cadena, inicio, longitud) devuelve una subcadena de la cadena especificada, comenzando en el índice de inicio y con la longitud especificada.

Ejemplo de uso:

```
SELECT SUBSTRING('Hello World', 7, 5) AS Subcadena
```

	Subcadena
1	World

Este ejemplo devuelve la subcadena de la cadena 'Hello World' que comienza en el séptimo carácter (que es 'W') y tiene una longitud de 5 caracteres. El resultado será 'World'.

Ejemplo con Northwind:

```
USE Northwind
SELECT ProductName, SUBSTRING(ProductName, 1, 10) AS ShortProductName
FROM Products
WHERE ProductID = 1
```

	ProductName	ShortProductName
1	Chai	Chai

En este ejemplo, estamos extrayendo los primeros 10 caracteres del nombre del producto de la tabla Products en la base de datos Northwind utilizando la función SUBSTRING. Esto nos permite obtener una versión abreviada del nombre del producto para su visualización en una aplicación o informe.

6.13 UPPER

Se utiliza para convertir una cadena de caracteres en mayúsculas. Esto puede ser útil cuando se desea realizar comparaciones de cadenas sin distinguir entre mayúsculas y minúsculas, o cuando se desea presentar la información en mayúsculas de manera uniforme.

UPPER (cadena) convierte todos los caracteres de una cadena en mayúsculas.

Ejemplo de uso:

```
SELECT UPPER('hola mundo') AS Mayusculas
```

	Mayusculas
1	HOLA MUNDO

Ejemplo con Northwind:

```
USE Northwind
SELECT UPPER(ProductName) AS ProductNameUpper
FROM Products
```

	ProductNameUpper
1	ALICE MUTTON
2	ANISEED SYRUP
3	BOSTON CRAB MEAT

En este ejemplo, estamos seleccionando los nombres de los productos de la tabla Products en la base de datos Northwind y convirtiéndolos en mayúsculas utilizando la función UPPER. Esto nos permite obtener los nombres de los productos en mayúsculas, independientemente de cómo estén almacenados en la base de datos.

7. Funciones de agregación

7.1 COUNT

se utiliza para contar el número de filas que cumplen ciertas condiciones en una tabla. Es una de las funciones de agregación más comunes y se puede utilizar en combinación con GROUP BY para contar el número de filas en grupos específicos. COUNT(expresión) devuelve el número de filas en una tabla que satisfacen la condición especificada en la expresión.

Ejemplo de uso en Northwind:

```
USE Northwind
SELECT COUNT(*) AS TotalProductos
FROM Products
```

	TotalProductos
1	77

En este ejemplo, estamos contando el número total de filas en la tabla Products de la base de datos.

7.2 SUM

Se utiliza para calcular la suma de los valores en una columna numérica de una tabla.

SUM (columna) calcula la suma de los valores en la columna especificada.

Ejemplo de uso con Northwind:

```
USE Northwind
SELECT CustomerID, SUM(OrderPrice) AS TotalOrderPrice
FROM (
    SELECT CustomerID, o.OrderID, SUM(UnitPrice * Quantity) AS OrderPrice
    FROM Orders o
    JOIN [Order Details] od ON o.OrderID = od.OrderID
    GROUP BY CustomerID, o.OrderID
) AS Subquery
GROUP BY CustomerID
```

	CustomerID	TotalOrderPrice
1	ALFKI	4596.20
2	ANATR	1402.95
3	ANTON	7515.35
4	AROUT	13806.50
5	BERGS	26968.15
6	BLAUS	3239.80

En este ejemplo, estamos calculando la suma de los importes de los pedidos (OrderPrice) para cada cliente en la tabla Customers. Primero, unimos las tablas Orders y Order Details para obtener los detalles de los pedidos y calcular el importe total de cada pedido. Luego, agrupamos los resultados por CustomerID y calculamos la suma de los importes para cada cliente.

7.3 AVG

Se utiliza para calcular el valor promedio de una columna numérica en un conjunto de datos. Proporciona una manera fácil y conveniente de calcular el promedio de valores numéricos en una consulta.

AVG(columna) calcula el promedio de los valores en la columna especificada. Se puede utilizar con cualquier columna numérica, como precios, cantidades, puntajes, etc.

Ejemplo de uso con Northwind:

```
USE Northwind
SELECT AVG(UnitPrice) AS PrecioPromedio
FROM Products
```

	PrecioPromedio
1	28.8663

En este ejemplo, estamos utilizando la función AVG para calcular el precio promedio de los productos en la tabla Products de la base de datos Northwind. La función AVG calculará automáticamente el promedio de los valores en la columna UnitPrice.

La función AVG es útil para obtener rápidamente el promedio de una columna numérica en una consulta SQL, lo que puede ser útil para análisis y toma de decisiones.

7.4 MIN

Se utiliza para obtener el valor mínimo en un conjunto de valores. Puede aplicarse a columnas numéricas, de fecha/hora o de texto para encontrar el valor más pequeño dentro del conjunto de datos especificado.

MIN(valor) devuelve el valor mínimo en un conjunto de valores.

Ejemplo de uso con Northwind:

```
USE Northwind
SELECT MIN(UnitPrice) AS MinPrecio
FROM Products
```

	MinPrecio
1	2.50

En este ejemplo, estamos utilizando la función MIN para encontrar el precio mínimo (UnitPrice) de los productos en la tabla Products.

7.5 MAX

Se utiliza para devolver el valor máximo de una columna en un conjunto de resultados.

MAX(columna) devuelve el valor máximo de la columna especificada en el conjunto de resultados.

Ejemplo de uso con Northwind:

```
USE Northwind
SELECT MAX(ShippedDate) AS FechaEnvioMaxima
FROM Orders
```

	FechaEnvioMaxima
1	1998-05-06 00:00:00.000

Aquí estamos seleccionando la fecha de envío más reciente (ShippedDate) de todos los pedidos en la base de datos Northwind. La función MAX se utiliza para encontrar

la fecha de envío máxima entre todos los pedidos. Esto nos dará la fecha más reciente en la que se ha enviado un pedido.

7.6 STDEV

Se utiliza para calcular la desviación estándar de un conjunto de valores en una columna. La desviación estándar es una medida de dispersión que indica cuánto varían los valores de un conjunto de datos con respecto a la media.

STDEV(valor) calcula la desviación estándar de un conjunto de valores en una columna.

Ejemplo de uso con Northwind:

```
USE Northwind
SELECT STDEV(UnitPrice) AS StandardDeviation
FROM Products
```

	StandardDeviation
1	33.8151114580251

En este ejemplo, estamos calculando la desviación estándar de los precios unitarios (UnitPrice) de los productos en la base de datos Northwind. Esto nos dará una idea de la variabilidad de los precios de los productos en la base de datos.

7.7 STDEVP

Se utiliza para calcular la desviación estándar de una población de valores numéricos. Es útil para medir la dispersión de los datos en una población específica.

STDEVP(valor) calcula la desviación estándar de una población de valores numéricos.

Ejemplo de uso con Northwind:

```
USE Northwind
SELECT STDEVP(UnitPrice) AS StandardDeviation
FROM Products
```

	StandardDeviation
1	33.5948152272562

En este ejemplo, estamos calculando la desviación estándar de la columna UnitPrice de la tabla Products en la base de datos Northwind. Esto nos dará una medida de la dispersión de los precios de los productos en toda la población.

7.8 VAR

Se utiliza para calcular la varianza de un conjunto de valores. Calcula la medida de dispersión de los datos en relación con su media. La varianza se utiliza comúnmente

en estadísticas para comprender la distribución de los datos y la dispersión de los valores en un conjunto de datos.

VAR (expresión) calcula la varianza de un conjunto de valores. La varianza se calcula como la suma de los cuadrados de las diferencias entre cada valor y la media, dividida por el número total de valores menos 1.

Ejemplo de uso con Northwind:

```
USE Northwind
SELECT VAR(UnitPrice) AS Variance
FROM Products
```

	Variance
1	1143.46176291866

En este ejemplo, estamos calculando la varianza de los precios unitarios (UnitPrice) en la tabla Products de la base de datos Northwind. Esto nos dará una medida de la dispersión de los precios unitarios en relación con su media.

La función VAR es útil para comprender la variabilidad de los datos en un conjunto de valores, lo que puede ser importante para el análisis estadístico y la toma de decisiones.

7.9 VARP

Se utiliza para calcular la varianza poblacional de un conjunto de valores. La varianza poblacional es una medida de dispersión que indica cuánto varían los valores de un conjunto de datos en relación con la media de esos datos.

VARP devuelve la varianza poblacional de un conjunto de valores. Es útil para medir la dispersión de los datos alrededor de la media en un conjunto de datos completo.

Ejemplo de uso con Northwind:

```
USE Northwind
SELECT VARP(UnitPrice) AS VarianzaPrecios
FROM Products
```

	VarianzaPrecios
1	1128.61161015348

En este ejemplo, estamos calculando la varianza poblacional de los precios de los productos en la tabla Products de la base de datos Northwind. Esto nos proporciona una medida de cuánto varían los precios de los productos en relación con la media de esos precios.

8. SELECT

8.1 Todos los campos

Para consultar todos los campos de una tabla de SQL Server se necesita utilizar el símbolo asterisco (*) con el selectivo SELECT. Esto te permitirá ver los datos que contiene alguna tabla de una base de datos, aquí un ejemplo para ver los campos:

```
SELECT * FROM Employees
```

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate
1	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000
2	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00:00.000
3	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00:00.000
4	4	Peacock	Margaret	Sales Representative	Mrs.	1937-09-19 00:00:00.000
5	5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00:00.000
6	6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00:00.000
7	7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00:00.000
8	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00:00.000
9	9	Dodsworth	Anne	Sales Representative	Ms.	1966-01-27 00:00:00.000

Como puedes observar se necesita escribir primero SELECT, después el asterisco (*), FROM ayuda a saber de qué tabla queremos ver los campos y después el nombre de la tabla, como resultado nos da la visión de los datos que tiene dicha tabla que en este caso es el de los empleados.

8.2 Campos específicos

Para poder ver un campo en específico, igual que cuando consultamos todos los campos de una tabla solamente sustituimos el asterisco con el nombre de una columna que deseamos visualizar de alguna tabla, aquí un ejemplo de como hacerlo:

```
SELECT FIRSTNAME FROM Employees
```

	FIRSTNAME
1	Nancy
2	Andrew
3	Janet
4	Margaret
5	Steven

En este caso hice llamar la columna de los apellidos de los empleados, ahora bien, si queremos consultar más de una columna solo es de colocar una coma (,) y escribir el encabezado o columna que queremos visualizar, como en el siguiente ejemplo:

```
SELECT EmployeeID, FIRSTNAME, LASTNAME FROM Employees
```

	EmployeeID	FIRSTNAME	LASTNAME
1	1	Nancy	Davolio
2	2	Andrew	Fuller
3	3	Janet	Leverling
4	4	Margaret	Peacock
5	5	Steven	Buchanan
6	6	Michael	Suyama
7	7	Robert	King
8	8	Laura	Callahan
9	9	Anne	Dodsworth

En este caso, hice llamar 3 columnas para ver que se puede consultar más de dos consultas.

8.3 Renombrar encabezados de las columnas

Cuando se requiere renombrar una columna para hacer una consulta es necesario utilizar la cláusula ‘AS’, con esto nos ayudara a cambiar el nombre de una columna ya existente pero solo cuando se consulta, esto no afectara en la documentación original, también se puede renombrar aquellas columnas que no tienen nombre y que fueron creadas por la consulta de datos que hagamos.

Por ejemplo, usaremos una tabla de empleados que manejaremos 3 datos que es el ID del empleado, nombre y apellido:

```
SELECT EMPLOYEEID, FISRTNAME, LASTNAME FROM EMPLOYEES
```

	EmployeeID	FirstName	LastName
1	1	Nancy	Davolio
2	2	Andrew	Fuller
3	3	Janet	Leverling
4	4	Margaret	Peacock
5	5	Steven	Buchanan
6	6	Michael	Suyama
7	7	Robert	King
8	8	Laura	Callahan
9	9	Anne	Dodsworth

Podemos observar que originalmente las columnas están en ingles ya que así fue creada la tabla, pero con ayuda de AS podemos arreglar eso de la siguiente manera:

```
SELECT EmployeeID, FirstName AS NOMBRE, LastName AS APELLIDO FROM Employees
```

	EmployeeID	NOMBRE	APELLIDO
1	1	Nancy	Davolio
2	2	Andrew	Fuller
3	3	Janet	Leverling
4	4	Margaret	Peacock
5	5	Steven	Buchanan
6	6	Michael	Suyama
7	7	Robert	King
8	8	Laura	Callahan
9	9	Anne	Dodsworth

Ahora las columnas Firstname y Lastname están en español, solo es de usar AS y el nombre o palabra por la que queremos renombrar el encabezado.

Otro ejemplo puede ser cuando hacemos operaciones que nosotros creamos y al momento de ejecutarlo sale el resultado, pero no tiene un encabezado como lo vemos en la siguiente imagen:

```
SELECT 'Hola', 1,  GETDATE(), 26+25, FIRSTNAME FROM Employees
```

	(No column name)	(No column name)	(No column name)	(No column name)	FIRSTNAME
1	Hola	1	2024-04-16 17:16:55.010	51	Nancy
2	Hola	1	2024-04-16 17:16:55.010	51	Andrew
3	Hola	1	2024-04-16 17:16:55.010	51	Janet

Para solucionar esto, nuevamente usamos la cláusula AS y quedara de la siguiente manera:

```
SELECT 'Hola' AS MENSAJE, 1 AS NUMERO,  GETDATE() AS 'FECHA Y HORA', 26+25 AS OPERACIONES, FIRSTNAME AS 'APELLIDO'
FROM Employees
```

	MENSAJE	NUMERO	FECHA Y HORA	OPERACIONES	APELLIDO
1	Hola	1	2024-04-16 22:19:26.110	51	Nancy
2	Hola	1	2024-04-16 22:19:26.110	51	Andrew
3	Hola	1	2024-04-16 22:19:26.110	51	Janet
4	Hola	1	2024-04-16 22:19:26.110	51	Margaret

8.4 Valores sin repetición

Cuando deseamos consultar valores, a veces nos topamos muchos datos repetidos como los que veras a continuación:

	EmployeeID	LastName	FirstName	Title
1	1	Davolio	Nancy	Sales Representative
2	2	Fuller	Andrew	Vice President, Sales
3	3	Leverling	Janet	Sales Representative
4	4	Peacock	Margaret	Sales Representative
5	5	Buchanan	Steven	Sales Manager
6	6	Suyama	Michael	Sales Representative
7	7	King	Robert	Sales Representative
8	8	Callahan	Laura	Inside Sales Coordinator
9	9	Dodsworth	Anne	Sales Representative

Utilizando la tabla de empleados, vemos que la mayoría de los empleados tienen un título similar que es “Sales Representative” o traducido a representantes de ventas, ahora sí solo queremos ver solo una vez el título o puesto de trabajo. Utilizaremos la cláusula DISTINC que nos ayudara a reducir los valores repetidos haciendo lo siguiente:

```
SELECT DISTINCT Title FROM Employees
```

	Title
1	Inside Sales Coordinator
2	Sales Manager
3	Sales Representative
4	Vice President, Sales

Como se puede ver ya no se repiten los encargos de los empleados, ahora si queremos consultar más columnas aun usando la cláusula DISTINC, no tendremos los mismos resultados ya que otras consultas pueden variar y requerirán la muestra del puesto del empleado correspondiente, como en el siguiente ejemplo que se agregara los apellidos de los empleados:

```
SELECT DISTINCT Title, LastName FROM Employees
```

	Title	LastName
1	Inside Sales Coordinator	Callahan
2	Sales Manager	Buchanan
3	Sales Representative	Davolio
4	Sales Representative	Dodsworth
5	Sales Representative	King
6	Sales Representative	Leverling
7	Sales Representative	Peacock
8	Sales Representative	Suyama
9	Vice President, Sales	Fuller

Como no hay apellidos que se repitan o que alguien tenga el mismo apellido y mismo puesto de trabajo, como resultado termina dando más resultado que el anterior ejemplo.

8.5 Primeros n registros

Para obtener los primeros N registros de una tabla en SQL Server, puedes utilizar la cláusula TOP en una consulta SELECT. Esto te permite limitar el número de filas que se devuelven en el resultado de la consulta. Aquí tienes un ejemplo de cómo se usa:

Supongamos que tenemos una tabla llamada Orders y queremos obtener los primeros 10 registros de esta tabla. Podemos hacerlo de la siguiente manera:

```
SELECT TOP 5 * FROM Orders
```

	OrderID	CustomerID	EmployeeID	OrderDate
1	10248	VINET	5	1996-07-04 00:00:00.000
2	10249	TOMSP	6	1996-07-05 00:00:00.000
3	10250	HANAR	4	1996-07-08 00:00:00.000
4	10251	VICTE	3	1996-07-08 00:00:00.000
5	10252	SUPRD	4	1996-07-09 00:00:00.000

Como ven la cláusula TOP me dio los primeros datos de la tabla orders que en este caso le pedí los primeros 5

8.6 Columnas calculadas

En SQL Server, las columnas calculadas son aquellas que se evalúan por valores de otras columnas de la misma tabla y de la misma fila. Estas columnas no se almacenan por ser una consulta, por lo que calcula dinámicamente cuando se accede a las consultas, un ejemplo puede ser que queremos calcular la columna UnitPrice (Precio unitario) con UnitsOnOrders (Unidades de pedidos) de la tabla Products quedando de la siguiente manera

```
SELECT UnitPrice, UnitsOnOrder, unitPrice + (UnitPrice * UnitsOnOrder) AS 'Calculo' FROM Products
```

	UnitPrice	UnitsOnOrder	Calculo
1	18.00	0	18.00
2	19.00	40	779.00
3	10.00	70	710.00
4	22.00	0	22.00
5	21.35	0	21.35
6	25.00	0	25.00
7	30.00	0	30.00
8	40.00	0	40.00
9	97.00	0	97.00

En esta ocasión multiplique las columnas y después lo que me saliera lo sume con la columna UnitPrice (Precio Unitario). Esto solo fue un ejemplo de cálculos entre columnas que tengan valores, ya que, si intenta hacer cálculos con un tipo de dato string, no les funcionara.

8.7 Constantes

En SQL Server, las constantes se utilizan para representar valores fijos que no cambian durante la ejecución de una consulta. Estos valores pueden utilizarse en muchas partes de una consulta, que pueden ser como condiciones, expresiones y asignaciones de columnas, un ejemplo puede ser que queremos ver los productos unitarios que cuesten más de 30.00, por que quedaría de la siguiente manera:

```
SELECT UnitPrice from Products where UnitPrice > 30
```

	UnitPrice
1	40.00
2	97.00
3	31.00
4	38.00
5	39.00
6	62.50

De esta forma podemos sacar valores específicos y fijos que nos ayudara ver mejor los resultados que nos solicitan.

8.8 Uso de calificadores

En SQL Server, los calificadores son utilizados para especificar a qué tabla o tabla derivada pertenece una columna en una consulta que involucra múltiples tablas. Esto es especialmente útil cuando las tablas tienen columnas con el mismo nombre. Aquí tienes un ejemplo de cómo se utilizan los calificadores en SQL Server:

```
SELECT Customers.CustomerID, Orders.OrderID
FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID
```

	CustomerID	OrderID
1	ALFKI	10643
2	ALFKI	10692
3	ALFKI	10702
4	ALFKI	10835
5	ALFKI	10952
6	ALFKI	11011
7	ALFKI	11078

En este ejemplo, utilizamos Customers.CustomerID y Orders.OrderID para especificar claramente de qué tabla provienen estas columnas. Esto evita cualquier ambigüedad que pueda surgir debido a las columnas con el mismo nombre en diferentes tablas.

También se pueden utilizar alias para abreviar los nombres de las tablas y hacer que la consulta sea más chica.

```
SELECT C.CustomerID, O.OrderID
FROM Customers C JOIN Orders O ON C.CustomerID = O.CustomerID
```

Da el mismo resultado, lo único que cambia es la abreviación del nombre de la tabla.

8.9 Ordenamiento

En SQL Server, alinear los datos de una columna ya sea de arriba hacia abajo o viceversa, esta función es útil cuando queremos saber que empleado tiene más sueldo o que producto cuesta más o cuesta menos, dependiendo lo que queremos buscar.

8.9.1 Ascendente

Comenzamos utilizando la cláusula ASC, que significa ascendente, esto nos ayudara a tener los datos de arriba hacia abajo, aquí un ejemplo:

```
select ORDERID, ORDERDATE, FREIGHT FROM ORDERS ORDER BY ORDERID ASC
```

	ORDERID	ORDERDATE	FREIGHT
1	10248	1996-07-04 00:00:00.000	32.38
2	10249	1996-07-05 00:00:00.000	11.61
3	10250	1996-07-08 00:00:00.000	65.83
4	10251	1996-07-08 00:00:00.000	41.34
5	10252	1996-07-09 00:00:00.000	51.30
6	10253	1996-07-10 00:00:00.000	58.17

Como los datos ya vienen con los datos ascendentes, no se reflejan los cambios.

8.9.2 Descendente

Ahora utilizamos la cláusula DESC, que significa descendente, esto nos ayudara a tener los datos de abajo hacia arriba, aquí un ejemplo:

```
select ORDERID, ORDERDATE, FREIGHT FROM ORDERS ORDER BY ORDERID DESC
```

	ORDERID	ORDERDATE	FREIGHT
1	11078	2024-04-14 21:37:01.413	0.00
2	11077	1998-05-06 00:00:00.000	8.53
3	11076	1998-05-06 00:00:00.000	38.28
4	11075	1998-05-06 00:00:00.000	6.19
5	11074	1998-05-06 00:00:00.000	18.44
6	11073	1998-05-05 00:00:00.000	24.95

A diferencia de ASC, con DESC si se aprecia el cambio de los datos viendo que los últimos datos de la tabla se encuentran en la parte superior.

8.10 Decisiones antes de proyectar

Se pueden tomar decisiones sobre los datos utilizando las declaraciones CASE. La estructura CASE permite realizar evaluaciones condicionales en los datos y realizar acciones en función de esas condiciones. Hay dos maneras de utilizar la declaración CASE que son el CASE simple y CASE compuesto.

8.10.1 CASE Simple

La forma más básica de la declaración CASE es la forma simple, esto evalúa las condiciones en orden. Cuando se encuentra una condición que se cumple, se devuelve resultado correspondiente. Si ninguna de las condiciones se cumple, se devuelve el resultado predeterminado especificado en la cláusula ELSE (si está presente) o NULL si no hay ninguna cláusula ELSE.

```
select productid, productname, unitprice, DISCONTINUED,
CASE
WHEN DISCONTINUED <= 0 AND UNITPRICE <= 50 THEN 'VIGENTE Y BARATO'
WHEN DISCONTINUED <= 1 AND UNITPRICE <= 50 THEN 'DESCONTINUADO Y BARATO'
WHEN DISCONTINUED <= 0 AND UNITPRICE <= 100 THEN 'VIGENTE E INTERMEDIO'
WHEN DISCONTINUED <= 1 AND UNITPRICE <= 100 THEN 'DESCONTINUADO E INTERMEDIO'
ELSE 'CARIÑOSO'
END UNITSINSTOCK
from products
```

	productid	productname	unitprice	DISCONTINUED	UNITSINSTOCK
1	1	Chai	18.00	0	VIGENTE Y BARATO
2	2	Chang	19.00	0	VIGENTE Y BARATO
3	3	Aniseed Syrup	10.00	0	VIGENTE Y BARATO
4	4	Chef Anton's Cajun Seasoning	22.00	0	VIGENTE Y BARATO
5	5	Chef Anton's Gumbo Mix	21.35	1	DESCONTINUADO Y BARATO
6	6	Grandma's Boysenberry Spread	25.00	0	VIGENTE Y BARATO
7	7	Uncle Bob's Organic Dried Pears	30.00	0	VIGENTE Y BARATO
8	8	Northwoods Cranberry Sauce	40.00	0	VIGENTE Y BARATO
9	9	Mishi Kobe Niku	97.00	1	DESCONTINUADO E INTERMEDIO

En este ejemplo, declaramos la discontinuidad de los productos unitarios de la tabla products buscando que productos están vigente, baratos, descontinuados, intermedios o caros.

8.10.2 CASE compuesto

La declaración CASE compuesto permite realizar evaluaciones más complejas mediante la combinación de múltiples condiciones. La estructura de la declaración CASE compuesto en SQL Server es similar a la declaración CASE simple, pero con múltiples condiciones en cada WHEN:

```
SELECT EMPLOYEEID, FIRSTNAME, LASTNAME, CITY, EXTENSION,
CASE WHEN EXTENSION >= 4000 THEN 'Tiene buena extension'
WHEN EXTENSION <= 3000 THEN 'No tiene buena extension'
WHEN EXTENSION > 3000 AND EXTENSION < 4000 THEN 'Extension pasable'
END AS PROMEDIO_EXTENSION FROM EMPLOYEES
```

	EMPLOYEEID	FIRSTNAME	LASTNAME	CITY	EXTENSION	PROMEDIO_EXTENSION
1	1	Nancy	Davolio	Seattle	5467	Tiene buena extension
2	2	Andrew	Fuller	Tacoma	3457	Extension pasable
3	3	Janet	Leverling	Kirkland	3355	Extension pasable
4	4	Margaret	Peacock	Redmond	5176	Tiene buena extension
5	5	Steven	Buchanan	London	3453	Extension pasable
6	6	Michael	Suyama	London	428	No tiene buena extension
7	7	Robert	King	London	465	No tiene buena extension

En este ejemplo, vemos la declaración de las extensiones que tienen los empleados por lo que se declara si tienen buena extensión, una extensión pasable o no tiene buena extensión.

8.11 Funciones de agregación

En SQL Server, la cláusula COUNT sirve para hacer un conteo de la cantidad de datos que tiene una columna, por lo verá a continuación:

```
SELECT COUNT(*) FROM PRODUCTS;
```

	(No column name)
1	77

Vemos que la tabla products tiene un total de 77 datos, ahora si queremos hacer un conteo más específico dentro de la cláusula COUNT escribimos el nombre de la columna que queremos hacer el conteo, a veces esto nos ayudara a contar valores:

```
SELECT COUNT(*) 'CANTIDAD DE EMPLEADOS', COUNT(REGION) AS REGIONES, COUNT(DISTINCT TITLE) AS TITULO
FROM EMPLOYEES;
```

	CANTIDAD DE EMPLEADOS	REGIONES	TITULO
1	9	5	4

En este ejemplo, hacemos el conteo del total de empleados, el conteo de las regiones que maneja la tabla y los puestos o títulos de los empleados.

Otra función de agregación es utilizando las clausulas MAX y MIN que su función es ver los valores que tengan mayor o menor valor, es como usar ASC y DESC, pero en vez de solo mover los datos de arriba hacia abajo o viceversa, estos ordenaran los valores que sean los más altos al más bajo o viceversa, aquí unos ejemplos:

```
SELECT MIN(UNITPRICE) AS 'Valor minimo' FROM PRODUCTS;
SELECT MAX(UNITPRICE) AS 'Valor maximo' FROM PRODUCTS;
```

	Valor minimo
1	2.50

	Valor maximo
1	263.50

Aquí podemos ver la diferencia entre MAX y MIN.

Otra función de agregación es AVG que nos permite hacer un promedio de los valores que tenga una columna:

```
SELECT AVG(UNITSINSTOCK) AS PROMEDIO FROM PRODUCTS
```

	PROMEDIO
1	40

Esta función es muy útil cuando queremos promediar valores.

Por último, tenemos la función SUM que prácticamente hace una suma del total de valores que se encuentran en una columna.

```
SELECT SUM(UNITPRICE) AS SUMA FROM PRODUCTS;
```

	SUMA
1	2222.71

8.12 Agrupación de subconjuntos

La agrupación de subconjuntos se refiere al proceso de dividir los datos en grupos basados en ciertos criterios y luego realizar cálculos o agregaciones dentro de cada grupo. Para realizar esto se utilizara la cláusula GROUP BY en combinación con funciones de agregación como SUM(), COUNT(), AVG(), etc. Aquí un ejemplo de la cláusula GRUOP BY:

```
SELECT TITLE, TITLEOFCOURTESY, COUNT(*), MIN(BIRTHDATE) FROM EMPLOYEES GROUP BY TITLE, TITLEOFCOURTESY
```

	TITLE	TITLEOFCOURTESY	(No column name)	(No column name)
1	Vice President, Sales	Dr.	1	1952-02-19 00:00:00.000
2	Sales Manager	Mr.	1	1955-03-04 00:00:00.000
3	Sales Representative	Mr.	2	1960-05-29 00:00:00.000
4	Sales Representative	Mrs.	1	1937-09-19 00:00:00.000
5	Inside Sales Coordinator	Ms.	1	1958-01-09 00:00:00.000
6	Sales Representative	Ms.	3	1948-12-08 00:00:00.000

En este ejemplo, vemos que se utilizan variables y datos con valores, esto es para que observen que se puede hacer una combinación entre estas consultas, gracias a la cláusula GRUOP BY, aquí se está haciendo el conteo que hay tanto el título de cortesía como el título de los empleados y la fecha de nacimiento más chica.

8.13 Unión de consultas

La operación UNION en SQL Server se utiliza para combinar los resultados de dos o más consultas SQL en un conjunto de resultados único. La operación UNION elimina automáticamente cualquier duplicado de filas del conjunto de resultados final.

Ejemplo:

```
SELECT EmployeeID FROM Employees
UNION
SELECT EmployeeID FROM Orders
```

	EmployeeID
1	1
2	2
3	3
4	4
5	5

En este ejemplo, unimos dos tablas que es employees y orders que ambos comparten una consulta que es el employeeID, con la operación UNION se evitó la duplicación de la columna.

8.14 WHERE

8.14.1 Operadores relacionados básicos

Las operaciones relacionadas básicas en SQL Server se utilizan para filtrar filas en una consulta basada en una condición específica. Estas operaciones incluyen:

Igualdad (=): Se utiliza para seleccionar filas donde el valor de una columna sea igual al valor especificado.

Diferencia (<>): Se utiliza para seleccionar filas donde el valor de una columna sea diferente al valor especificado.

Mayor que (>), Menor que (<): Se utilizan para seleccionar filas donde el valor de una columna sea mayor o menor que el valor especificado, respectivamente.

Mayor o igual que (>=), Menor o igual que (<=): Se utilizan para seleccionar filas donde el valor de una columna sea mayor o igual o menor o igual al valor especificado, respectivamente.

Ejemplo 1:

```
SELECT * FROM Customers WHERE Country = 'USA'
```

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country
1	GREAL	Great Lakes Food Market	Howard Snyder	Marketing Manager	2732 Baker Blvd.	Eugene	OR	97403	USA
2	HUNGC	Hungry Coyote Import Store	Yoshi Latimer	Sales Representative	City Center Plaza 516 Main St.	Elgin	OR	97827	USA
3	LAZYK	Lazy K Kountry Store	John Steel	Marketing Manager	12 Orchestra Terrace	Walla Walla	WA	99362	USA
4	LETSS	Let's Stop N Shop	Jaime Yorres	Owner	87 Polk St. Suite 5	San Francisco	CA	94117	USA
5	LONEP	Lonesome Pine Restaurant	Fran Wilson	Sales Manager	89 Chiaroscuro Rd.	Portland	OR	97219	USA
6	OLDWO	Old World Delicatessen	Rene Phillips	Sales Representative	2743 Bering St.	Anchorage	AK	99508	USA
7	RATTC	Rattlesnake Canyon Grocery	Paula Wilson	Assistant Sales Representative	2817 Milton Dr.	Albuquerque	NM	87110	USA

En este ejemplo, se utiliza la operación de igualdad para consultar los clientes de Estados Unidos.

Ejemmplo 2:

```
SELECT * FROM Products WHERE UnitPrice > 50
```

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice
1	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.00
2	18	Carnarvon Tigers	7	8	16 kg pkg.	62.50
3	20	Sir Rodney's Marmalade	8	3	30 gift boxes	81.00
4	29	Thüringer Rostbratwurst	12	6	50 bags x 30 sausgs.	123.79
5	38	Côte de Blaye	18	1	12 - 75 cl bottles	263.50
6	51	Manjimup Dried Apples	24	7	50 - 300 g pkgs.	53.00
7	59	Raclette Courdavault	28	4	5 kg pkg.	55.00

En este ejemplo, se utiliza el operador Mayor que, para consultar los precios unitarios que cuesten más de 50.

8.14.2 Operadores lógicos

Los operadores lógicos en SQL Server se utilizan para combinar múltiples condiciones en una sola expresión lógica.

8.14.2.1 AND

Se utiliza para combinar dos o más condiciones y devuelve true si todas las condiciones son verdaderas.

```
SELECT * FROM Orders WHERE ShipCountry = 'USA' AND ShipCity = 'New York'
```

En este ejemplo, se requiere una variable y después del AND otra variable, se requiere que estén las dos variables.

8.14.2.2 OR

Se utiliza para combinar dos o más condiciones y devuelve true si al menos una de las condiciones es verdadera.

```
SELECT * FROM Orders WHERE ShipCountry = 'USA' OR ShipCountry = 'Canada'
```

En este ejemplo, se requiere una variable y después del OR una variable, en caso de que no exista una de las dos.

8.14.2.3 NOT

Se utiliza para negar una condición y devuelve true si la condición es falsa.

```
SELECT * FROM Orders WHERE NOT ShipCountry = 'Canada'
```

En este ejemplo, Solo se requiere una variable e indicar con NOT que no queremos que esa consulta aparezca.

8.14.3 Operadores relacionales especiales

8.14.3.1 IS

Se utiliza para comparar un valor con NULL.

```
SELECT FIRSTNAME, REGION FROM EMPLOYEES WHERE REGION IS NOT NULL
```

	FIRSTNAME	REGION
1	Nancy	WA
2	Andrew	WA
3	Janet	WA
4	Margaret	WA
5	Laura	WA

En este ejemplo, se consulta Firstname y Region, donde se pide únicamente aquellos nombres que no tengan NULL en su región.

8.14.3.2 IN

Se utiliza para especificar múltiples valores en una condición de filtro.

```
SELECT FIRSTNAME, LEFT(FIRSTNAME, 1) AS VOCALES FROM EMPLOYEES
WHERE LEFT(FIRSTNAME, 1) IN ('A','E','I','O','U')
```

	FIRSTNAME	VOCALES
1	Andrew	A
2	Anne	A

En este ejemplo, se buscan los apellidos que inicien con una vocal, para esto se utiliza LEFT para indicar de qué lado queremos que condicione.

8.14.3.3 BETWEEN

Se utiliza para seleccionar filas donde el valor de una columna esté dentro de un rango específico de valores.

```
SELECT * FROM EMPLOYEES WHERE FIRSTNAME BETWEEN 'ANA' AND 'LUIS'
```

	EmployeeID	LastName	FirstName
1	2	Fuller	Andrew
2	3	Leverling	Janet
3	8	Callahan	Laura
4	9	Dodsworth	Anne

En este ejemplo, seleccionamos la fila de Firstname para buscar nombres similares a Ana y Luis, en si el rango tiene que ser que el nombre comience con ‘A’ y/o ‘L’.

8.14.3.4 LIKE

Se utiliza para seleccionar filas donde el valor de una columna coincida con un patrón específico. Esto se refiere a que podemos llamar un valor por una letras, por la cantidad de caracteres por letras que pueden diferenciar una palabra que se mencione lo mismo como el nombre de Nancy, que también la podemos encontrar como Nanci, Nansy, etc. Como se verá en los siguientes ejemplos:

```
SELECT * FROM EMPLOYEES WHERE FIRSTNAME LIKE 'NANC%';--SALE EL NOMBRE
SELECT * FROM EMPLOYEES WHERE FIRSTNAME LIKE 'NANC_';--AUN SALE EL NOMBRE
SELECT * FROM EMPLOYEES WHERE FIRSTNAME LIKE 'NAN___';--NO SALE NADA
```

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate
1	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate
1	1	Davolio	Nancy	Sales Representative	Ms.	1948-12-08 00:00:00.000

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City

Tenemos 3 ejemplos, con el primero usamos solo las primeras iniciales del nombre y el signo de porcentaje ayuda a completar el nombre que estamos buscando, el signo puede colocarse al principio, intermedio o final de la palabra.

Para el segundo ejemplo, se usa un guion bajo, esto se usa para especificar un valor su tamaño, dependiendo de la cantidad de guiones que use, es el tamaño de las características

Para el tercer ejemplo, se usa más de un carácter para buscar el nombre de Nancy pero como el nombre usa solo 5 caracteres, no lo considera por falta de letras.

8.15 HAVING

La cláusula HAVING se utiliza en combinación con la cláusula GROUP BY para filtrar filas de un conjunto de resultados basado en una condición de grupo. Se aplica después de que se hayan formado los grupos con la cláusula GROUP BY y antes de que se apliquen las funciones de agregación.

```
SELECT TITLEOFCOURTESY, COUNT(*)
FROM EMPLOYEES WHERE TITLE LIKE '%REPRESENTATIVE%'
GROUP BY TitleOfCourtesy HAVING COUNT(*) > 1
```

	TITLEOFCOURTESY	(No column name)
1	Mr.	2
2	Ms.	3

En este ejemplo, se busca los empleados hombres y mujeres que tengan el titulo de representativo, usando por último HAVING para hacer el conteo y que sea más de 1.

8.16 Subconsultas

Una subconsulta es una consulta que se encuentra dentro de otra consulta. Las subconsultas pueden ser SELECT, FROM, WHERE o HAVING de una consulta principal. Estos se utilizan para realizar operaciones más complejas o para buscar resultados de otras consultas.

Ejemplo:

```
SELECT * FROM ORDERS
WHERE EMPLOYEEID IN (
SELECT EMPLOYEEID
FROM EMPLOYEES
WHERE TITLEOFCOURTESY='MR.')
```

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate
1	10248	VINET	5	1996-07-04 00:00:00.000	1996-08-01 00:00:00.000
2	10249	TOMSP	6	1996-07-05 00:00:00.000	1996-08-16 00:00:00.000
3	10254	CHOPS	5	1996-07-11 00:00:00.000	1996-08-08 00:00:00.000
4	10264	FOLKO	6	1996-07-24 00:00:00.000	1996-08-21 00:00:00.000
5	10269	WHITC	5	1996-07-31 00:00:00.000	1996-08-14 00:00:00.000
6	10271	SPLIR	6	1996-08-01 00:00:00.000	1996-08-29 00:00:00.000

Este ejemplo, trata sobre los campos de la tabla orders dentro de esta consulta se buscará una columna que sea similar a una de las columnas de la tabla orders que en este caso se usará employeeID de la tabla Employees, ambas tablas comparten la misma consulta, para el selectivo interno solamente puede tener una cadena y que sea compatible con otra tabla.

8.17 Reunión de tablas

Para la reunión de tablas es la combinación entre dos o más tablas basadas en una condición específica, tenga en cuenta que para hacer esto, tienen que tener relación entre las tablas para que funcione. Esto se hace utilizando la cláusula JOIN en combinación con las cláusulas ON, USING o incluso WHERE.

Hay varios tipos de JOIN para alguna especificación más detallada comenzando con:

INNER JOIN: Devuelve solo las filas que tienen coincidencias en ambas tablas, también se puede usar sin la palabra INNER.

LEFT JOIN: Devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha. Si no hay coincidencias, devuelve NULL para las columnas de la tabla derecha.

RIGHT JOIN: Devuelve todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda. Si no hay coincidencias, devuelve NULL para las columnas de la tabla izquierda.

Ejemplos:

```
SELECT O.ORDERID, O.OrderDate, O.EMPLOYEEID, EM.FirstName
FROM ORDERS O INNER JOIN EMPLOYEES EM ON O.EmployeeID = EM.EmployeeID WHERE EM.TitleOfCourtesy = 'MR.'
```

	ORDERID	OrderDate	EMPLOYEEID	FirstName
1	10248	1996-07-04 00:00:00.000	5	Steven
2	10249	1996-07-05 00:00:00.000	6	Michael
3	10254	1996-07-11 00:00:00.000	5	Steven
4	10264	1996-07-24 00:00:00.000	6	Michael
5	10269	1996-07-31 00:00:00.000	5	Steven
6	10271	1996-08-01 00:00:00.000	6	Michael
7	10272	1996-08-02 00:00:00.000	6	Michael
8	10274	1996-08-06 00:00:00.000	6	Michael
9	10289	1996-08-26 00:00:00.000	7	Robert

En este ejemplo, solo hacemos la reunión entre la tabla orders y employees, utilizando la consulta employeeID para juntarlos y busca los empleados varones.

8.18 INTO

La cláusula INTO se utiliza para crear una nueva tabla y almacenar los resultados de una consulta dentro de las tablas creadas. Puede utilizarse en combinación con la cláusula SELECT para crear una tabla a partir de los resultados de una consulta.

8.18.1 Tablas permanentes

Las tablas permanentes son aquellas que existen de forma persistente en la base de datos. Se utilizan para almacenar datos de forma permanente y pueden ser accedidas y modificadas en cualquier momento.

```sql
SELECT * INTO NuevaTabla FROM Employees
```

Esto puede ayudar en caso de que necesitan modificar y que no estén %100 seguros de hacerlo y en caso de cometer un error, sería hecho en una tabla provisional creada por ustedes mismos..

8.18.2 Tablas temporales

Las tablas temporales son aquellas que existen solo durante la sesión de usuario o la ejecución de un procedimiento almacenado. Pueden ser de dos tipos: tablas temporales locales (denominadas #tabla_temporal) que están disponibles solo dentro del contexto de la sesión actual, y tablas temporales globales (denominadas ##tabla_temporal) que están disponibles para todas las sesiones de usuario en la base de datos.

Ejemplos:

```sql
-- Tabla temporal local
CREATE TABLE #Tablas (
    ID INT,
    Name VARCHAR(50)
);

-- Tabla temporal global
CREATE TABLE ##Tablaas (
    ID INT,
    Name VARCHAR(50)
);
```

8.19 Vistas

8.19.1 Creación

Las vistas en SQL Server son objetos de base de datos que almacenan consultas SQL y se comportan como tablas virtuales. Permiten a los usuarios acceder y manipular los datos de una o más tablas como si fueran una sola tabla.

8.19.2 Modificación

Puedes modificar una vista existente en SQL Server utilizando la instrucción ALTER VIEW, que te permite cambiar la definición de la vista.

9. INSERT

La instrucción INSERT se utiliza para agregar nuevas filas a una tabla en una base de datos SQL Server.

9.1 Sin especificar columnas

Cuando no especificas las columnas en la instrucción INSERT, debes proporcionar valores para todas las columnas de la tabla en el orden en que fueron definidas.

```
INSERT INTO TABLA2 VALUES (5, 'NANDO', 'GOMEZ')
```

9.2 Especificando columnas

Puedes especificar las columnas en la instrucción INSERT y proporcionar valores solo para esas columnas.

```
INSERT INTO TABLA2 (EmployeeID, NOMBRE, LastName) VALUES (5, 'NANDO', 'GOMEZ')
```

9.3 A partir de un SELECT

La instrucción INSERT a partir de un SELECT se utiliza para insertar datos en una tabla desde los resultados de una consulta SELECT.

```
INSERT INTO TABLA2 (EmployeeID, NOMBRE, LastName) VALUES (5, 'NANDO', 'GOMEZ')
SELECT EmployeeID*3, LASTNAME, NOMBRE FROM TABLA1 WHERE EmployeeID=4
```

Es importante asegurarse de que las columnas seleccionadas en el SELECT coincidan en tipo y número con las columnas en la tabla de destino.

10. UPDATE

La instrucción UPDATE se utiliza para modificar los datos existentes en una tabla en una base de datos SQL Server.

10.1 Sin condiciones

Cuando no especificas condiciones en la instrucción UPDATE, todos los registros de la tabla se actualizarán con los nuevos valores proporcionados.

```
UPDATE TABLA2 SET EMPLOYEEID=EMPLOYEEID*3
```

10.2 Con condiciones

Puedes especificar condiciones en la instrucción UPDATE para actualizar solo los registros que cumplan con ciertos criterios.

```
UPDATE TABLA2 SET EMPLOYEEID=EMPLOYEEID*3
WHERE EMPLOYEEID % 2 = 1
```

11. DELETE

La instrucción DELETE se utiliza para eliminar una o más filas de una tabla en una base de datos SQL Server.

11.1 Sin condiciones

Cuando no se especifican condiciones en la instrucción DELETE, se eliminan todas las filas de la tabla.

```
DELETE FROM TABLA2
```

11.2 Con condiciones

Puedes agregar una cláusula WHERE a la instrucción DELETE para especificar las filas que deseas eliminar en función de una condición.

```
DELETE FROM TABLA2 WHERE LEFT(NOMBRE,1) = 'N'
```

12.Ejercicios en clase

Insert.

```
SELECT * FROM TIPODONADOR

INSERT INTO TIPODONADOR
VALUES (1,  'TIPO1')

INSERT INTO TIPODONADOR (DESCRIPTIPO)
VALUES ('TIPO4')

INSERT INTO TIPODONADOR2
SELECT * FROM TIPODONADOR WHERE IDTIPODONADOR >= 2

SELECT DESCRIPTIPO AS DESCRIP, 5 AS NUM, 'ABC' AS CADENA
INTO TIPODONADOR4
FROM TIPODONADOR
WHERE IDTIPODONADOR IN (1, 3)\\

INSERT INTO TIPODONADOR
VALUES (5, 'TIPO5')

SELECT  FROM TIPODONADOR
```

Escalares

```
USE Northwind
SELECT COUNT(*), COUNT(LASTNAME), COUNT(REGION),
COUNT(DISTINCT TITLEOFCOURTESY), COUNT(DISTINCT REGION)
FROM Employees

SELECT COUNT(*) AS CONTEO, SUM(UNITSINSTOCK) AS SUMATORIA, AVG(UNITPRICE) AS
PROMEDIO,
MIN(UNITPRICE) AS MINIMO, MAX(UNITPRICE) AS MAXIMO
```

```
FROM Products


--GUTIERREZ DESEA SABER CUAL ES EL COBRO PROMEDIO POR CONCEPTO DE FLETES POR MES,
CONSIDERANDO TODA LA INFORMACIÓN

SELECT * FROM ORDERS
USE Northwind
SELECT YEAR(OrderDate) AS AÑO,
CASE MONTH(ORDERDATE)
WHEN 1 THEN 'ENERO'
WHEN 2 THEN 'FEBRERO'
WHEN 3 THEN 'MARZO'
WHEN 4 THEN 'ABRIL'
WHEN 5 THEN 'MAYO'
WHEN 6 THEN 'JUNIO'
WHEN 7 THEN 'JULIO'
WHEN 8 THEN 'AGOSTO'
WHEN 9 THEN 'SEPTIEMBRE'
WHEN 10 THEN 'OCTUBRE'
WHEN 11 THEN 'NOVIEMBRE'
WHEN 12 THEN 'DICIEMBRE'
END AS MES, AVG(FREIGHT) AS PROMEDIO
FROM ORDERS
GROUP BY YEAR(OrderDate), MONTH(ORDERDATE)
ORDER BY YEAR(OrderDate), MONTH(ORDERDATE)
```

13.Conclusión

Para esta unidad si se requiere utilizar muchas la lógica y comprensibilidad de los requisitos que nos llegan a pedir en algún trabajo que requiera de hacer consultas en una base de datos, por lo que es importante aprender el funcionamiento de cada cláusula, operaciones y entre otras herramientas que nos permitirán dar el resultado que nosotros esperamos, al igual de poder crear, modificar y eliminar información en tablas ya creadas.