	UNIVERSIDAD DON BOSCO FACULTAD DE ESTUDIOS TECNOLÓGICOS ESCUELA DE COMPUTACIÓN
CICLO 1-2016	GUÍA DE LABORATORIO Nº 7
	Nombre de la práctica: Uso de funciones SQL, Agrupando y sumando datos Lugar de ejecución: Laboratorio de Informática Tiempo estimado: 2 horas y 30 minutos Materia: Base de datos

I. Objetivos

Que el estudiante sea capaz de:

1. Implementar las diferentes tipos de funciones en la selección de datos almacenados en una base de datos.
2. Combinar las funciones de agregado con las diferentes cláusulas de agrupación de datos.

II. Introducción Teórica

Funciones de agregado

Las funciones de agregado realizan un cálculo sobre un conjunto de valores y devuelven un solo valor. Si exceptuamos la función COUNT, todas las funciones de agregado ignoran los valores NULL. Las funciones de agregado se suelen utilizar con la cláusula GROUP BY de la instrucción SELECT. Todas las funciones de agregado son deterministas, esto significa que las funciones de agregado devuelven el mismo valor cada vez que se las llama con un conjunto específico de valores de entrada.

Las funciones de agregado sólo se pueden utilizar como expresiones en:

- La lista de selección de una instrucción SELECT
- Cláusulas COMPUTE o COMPUTE BY.
- Cláusulas HAVING.
- En una subconsulta o en la consulta externa.

Transact-SQL proporciona las siguientes funciones de agregado:

Función de agregado	Descripción
AVG	Devuelve el promedio de los valores de un grupo.
COUNT	Devuelve el número de elementos de un grupo.
COUNT_BIG	Devuelve el número de elementos de un grupo. COUNT_BIG funciona como COUNT.
MAX	Devuelve el valor máximo de la expresión. MAX pasa por alto los valores NULL.
MIN	Devuelve el valor mínimo de la expresión. MIN pasa por alto los valores NULL.
SUM	Devuelve la suma de todos los valores o solo de los valores DISTINCT de la expresión.
STDEV	Devuelve la desviación típica estadística de todos los valores de la expresión especificada. Si STDEV se utiliza en todos los elementos de una instrucción SELECT, en el cálculo se incluirán todos los valores del conjunto de resultados. STDEV solo puede utilizarse con columnas numéricas. Los valores NULL se pasan por alto.
STDEVP	Devuelve la desviación estadística estándar para la población de todos los valores de la expresión especificada. Si se utiliza STDEVP en todos los elementos de una instrucción SELECT, se incluirán en el cálculo todos los valores del conjunto de resultados. STDEVP solo puede utilizarse con columnas numéricas. Los valores NULL se pasan por alto.
VAR	Devuelve la varianza estadística de todos los valores de la expresión especificada. Si se utiliza VAR en todos los elementos de una instrucción SELECT, cada valor del conjunto de resultados se incluye en el cálculo. VAR solo se puede utilizar con columnas numéricas. Los valores NULL se pasan por alto.
VARP	Devuelve la varianza estadística de la población para todos los valores de la expresión especificada. Si se utiliza VARP en todos los elementos de una instrucción SELECT, cada valor del conjunto de resultados se incluye en el cálculo. VARP solo se puede utilizar con columnas numéricas. Los valores NULL se pasan por alto.
CHECKSUM_AGG	Devuelve la suma de comprobación de los valores de un grupo. Se omiten los valores NULL.
GROUPING	Indica si una expresión de columna especificada en una lista GROUP BY es agregada o no. GROUPING devuelve 1 para agregado y 0 para no agregado, en el conjunto de resultados. GROUPING solo se puede usar en la lista de SELECT <selección>, cláusulas HAVING y ORDER BY cuando se especifica GROUP BY.

Diferencia entre los tipos de datos int y bigint

Tipo de datos	Intervalo	Almacenamiento
bigint	De -2^{63} (-9.223.372.036.854.775.808) a $2^{63}-1$ (9.223.372.036.854.775.807)	8 bytes
int	De -2^{31} (-2.147.483.648) a $2^{31}-1$ (2.147.483.647)	4 bytes

Cuando se ejecuta una función de agregado, SQL Server resume los valores de toda una tabla o de grupos de columnas de una tabla, y produce un valor por cada conjunto de filas para las columnas especificadas:

- Las funciones de agregado se pueden utilizar en la instrucción SELECT o en combinación con la cláusula GROUP BY.

- Con la excepción de la función COUNT(*), todas las funciones de agregado devuelven NULL si ninguna fila cumple la cláusula WHERE. La función COUNT(*) devuelve el valor cero si ninguna fila cumple la cláusula WHERE.

Sintaxis:

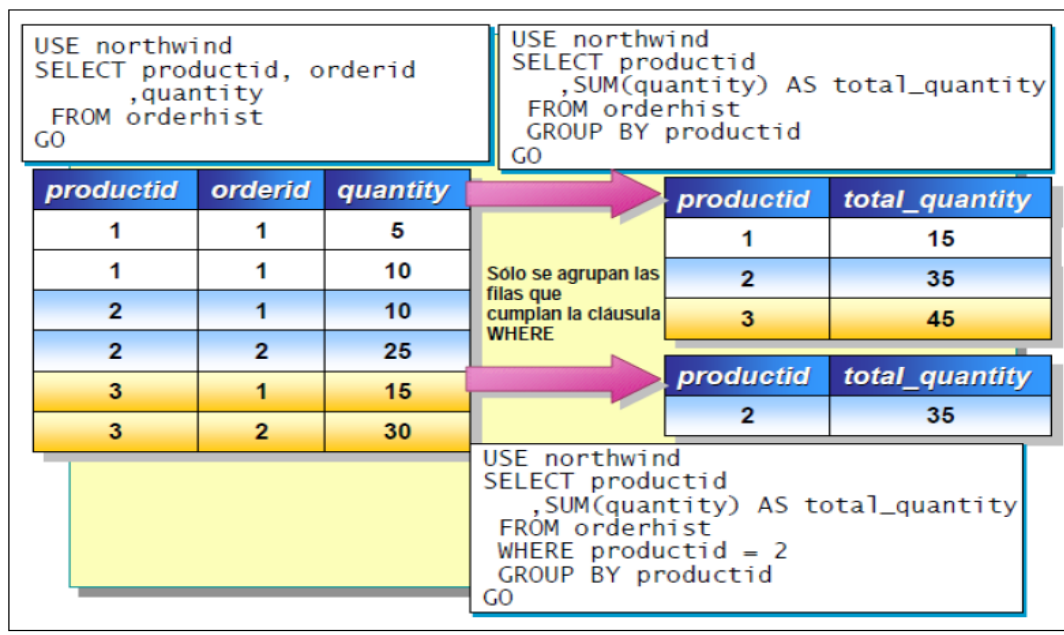
```
SELECT [ ALL | DISTINCT ]
[ TOP n [PERCENT] [ WITH TIES ] <listaSelección>
[ INTO nuevaTabla ] [ FROM <tablasOrigen > ]
[ WHERE <condicionesBúsqueda> ]
[ [ GROUP BY [ALL] expresiónAgrupación [, ...n]]
[ HAVING <condicionesBúsqueda> ]
[ WITH { CUBE | ROLLUP } ] ]
[ ORDER BY { columna [ ASC | DESC ] } [, ...n] ]
[ COMPUTE
{ { AVG | COUNT | MAX | MIN | SUM } (expresión) } [, ...n]
[ BY expresión [, ...n] ]
```

Uso de la cláusula GROUP BY

Utilizar la cláusula GROUP BY en columnas o expresiones para organizar filas en grupos y para resumir dichos grupos. Por ejemplo, utilice la cláusula GROUP BY para determinar la cantidad de pedido de cada producto en todos los pedidos registrados. Cuando utilice la cláusula GROUP BY, considerar los hechos e instrucciones siguientes:

- SQL Server produce una columna de valores por cada grupo definido.
- SQL Server sólo devuelve filas por cada grupo especificado; no devuelve información de detalle.
- Todas las columnas que se especifican en la cláusula GROUP BY tienen que estar incluidas en la lista de selección.
- Si incluye una cláusula WHERE, SQL Server sólo agrupa las filas que cumplen las condiciones de la cláusula WHERE.
- No utilice la cláusula GROUP BY en columnas que contengan varios valores nulos, porque los valores nulos se procesan como otro grupo.
- Utilice la palabra clave ALL con la cláusula GROUP BY para presentar todas las filas que tengan valores nulos en las columnas de agregado, independientemente de si las filas cumplen la condición de la cláusula WHERE.

Ejemplo:



En la consulta SELECT se pueden usar más de una función de agregado

Con las filas de la información de proceso correspondiente a una instrucción **SELECT** se pueden establecer grupos.

En cada uno de estos grupos, mediante las funciones de grupo, se pueden efectuar ciertos cálculos.

- Encontrar cuántos artículos hay registrados, el máximo y el mínimo precio unitario, el precio unitario medio y la valoración del almacén.

COUNT(*)	Nº de filas que componen el grupo.
COUNT(campo)	Nº de filas con valor asignado al campo (nulos no cuentan).
SUM(exp)	Suma de valores obtenidos con la expresión en cada fila.
AVG(exp)	Media.
MAX(exp)	Máximo.
MIN(exp)	Mínimo.
STDEV(exp)	Desviación típica.
VAR(exp)	Varianza.

```
SELECT COUNT(codigart) AS Cantidad, MAX(preunart) AS Max,
MIN(preunart) AS Min, AVG(preunart) AS Precio_medio,
SUM(preunart*stockart) AS Valoración
```

FROM Articulos

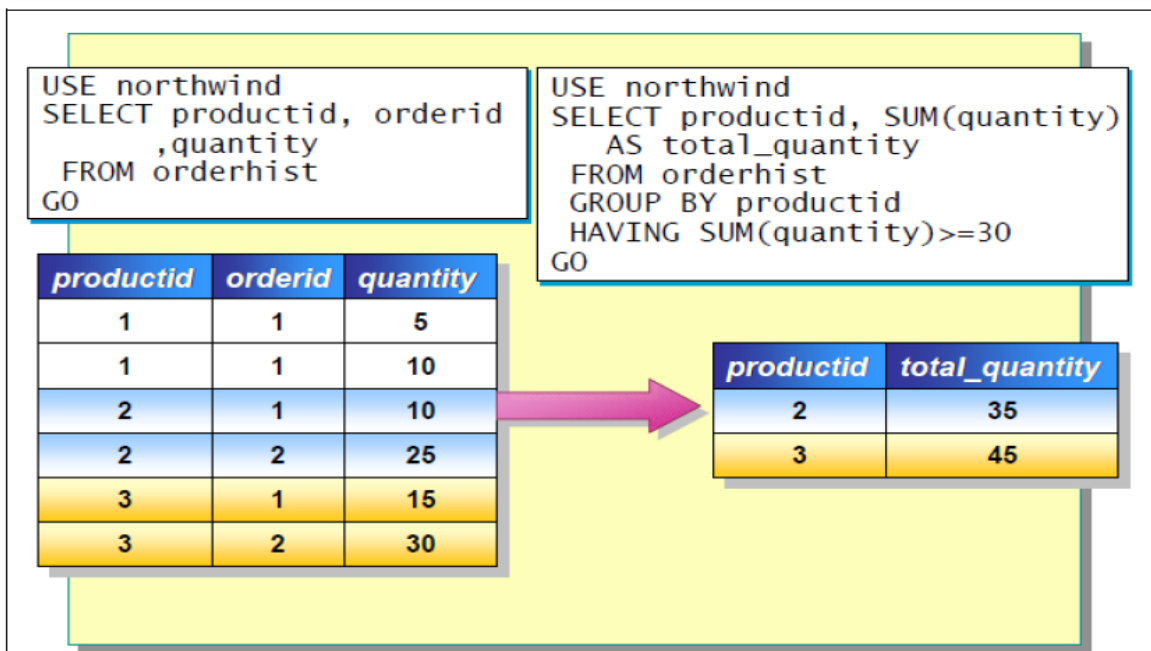
Cantidad	Max	Min	Precio_medio	Valoración
4	300,00	120,00	206,25	26640,00

Uso de la cláusula GROUP BY con la cláusula HAVING

Utilizar la cláusula HAVING en columnas o expresiones para establecer condiciones en los grupos incluidos en un conjunto de resultados. La cláusula HAVING establece condiciones en la cláusula GROUP BY de una forma muy similar a como interactúa la cláusula WHERE con la instrucción SELECT. Cuando utilice la cláusula HAVING, considere los hechos e instrucciones siguientes:

- Utilice la cláusula HAVING sólo con la cláusula GROUP BY para restringir los agrupamientos. El uso de la cláusula HAVING sin la cláusula GROUP BY no tiene sentido.
- En una cláusula HAVING puede haber hasta 128 condiciones. Cuando utilice varias condiciones, tiene que combinarlas con operadores lógicos (AND, OR o NOT).
- Puede hacer referencia a cualquiera de las columnas que aparezcan en la lista de selección.
- No utilice la palabra clave ALL con la cláusula HAVING, porque la cláusula HAVING pasa por alto la palabra clave ALL y sólo devuelve los grupos que cumplen la cláusula HAVING.

Ejemplo 1:



Ejemplo 2:

Mostrar el importe, sin aplicar el IVA, de los pedidos que tienen más de un registro o fila

```

SELECT numped, SUM((preunlin*unilin)*(1-desculin/100)) as Importe
FROM Lineas
GROUP BY numped
HAVING COUNT(*) > 1

```

numped	Importe
1	810
2	1534,8
4	2280

Generación de valores de agregado dentro de los conjuntos de resultados.

Utilice la cláusula GROUP BY con los operadores ROLLUP y CUBE para generar valores de agregado dentro de los conjuntos de resultados. Los operadores ROLLUP o CUBE pueden ser útiles para obtener información de referencias cruzadas dentro de una tabla sin tener que escribir secuencias de comandos adicionales. Cuando utilice los operadores ROLLUP o CUBE, use la función GROUPING para identificar los valores de detalle y de resumen dentro del conjunto de resultados.

Uso de la cláusula GROUP BY con el operador ROLLUP

Ejemplo 1:

<pre> USE northwind SELECT productid, orderid, SUM(quantity) AS total_quantity FROM orderhist GROUP BY productid, orderid WITH ROLLUP ORDER BY productid, orderid GO </pre>				
productid	orderid	total_quantity	Descripción	
NULL	NULL	95	Total general	
1	NULL	15	Resume sólo las filas para productid 1	
1	1	5	Detalla el valor para productid 1, orderid 1	
1	2	10	Detalla el valor para productid 1, orderid 2	
2	NULL	35	Resume sólo las filas para productid 2	
2	1	10	Detalla el valor para productid 2, orderid 1	
2	2	25	Detalla el valor para productid 2, orderid 2	
3	NULL	45	Resume sólo las filas para productid 3	
3	1	15	Detalla el valor para productid 3, orderid 1	
3	2	30	Detalla el valor para productid 3, orderid 2	

Utilizar la cláusula GROUP BY con el operador ROLLUP para resumir valores de grupos. La cláusula GROUP BY y el operador ROLLUP proporcionan datos en un formato relacional estándar. Cuando utilice la cláusula GROUP BY con el operador ROLLUP, considere los hechos e instrucciones siguientes:

- SQL Server procesa los datos de derecha a izquierda en la lista de columnas especificadas en la cláusula GROUP BY. Después, SQL Server aplica la función de agregado a cada grupo.
- SQL Server agrega al conjunto de resultados una fila que presenta los cálculos acumulados, como un total o un promedio acumulado. Dichos cálculos acumulados se indican en el conjunto de resultados con un valor NULL.
- Cuando utiliza el operador ROLLUP puede tener hasta 10 expresiones de agrupación.
- Con el operador ROLLUP no se puede utilizar la palabra clave ALL.
- Cuando utilice el operador CUBE, asegúrese de que las columnas que siguen a la cláusula GROUP BY son las que se han utilizado en la instrucción SELECT menos las que se han utilizado en cualquier función de agregado

Ejemplo 2:

```
SELECT Country,[State],City,
SUM ([Population (in Millions)]) AS [Population (in Millions)] FROM tblPopulation
GROUP BY Country,[State],City
WITH ROLLUP
GO
```

	Country	State	City	Population (in Millions)
1	India	Delhi	East Delhi	9
2	India	Delhi	North Delhi	5
3	India	Delhi	South Delhi	8
4	India	Delhi	West Delhi	7
5	India	Delhi	NULL	29
6	India	Karnataka	Bangalore	9
7	India	Karnataka	Belur	2
8	India	Karnataka	Manipal	1
9	India	Karnataka	NULL	12
10	India	Maharashtra	Mumbai	30
11	India	Maharashtra	Nagpur	11
12	India	Maharashtra	Nashik	6
13	India	Maharashtra	Pune	20
14	India	Maharashtra	NULL	67
15	India	NULL	NULL	108
16	NULL	NULL	NULL	108

Ejemplo 3:

```
SELECT PRODUCTID, ORDERID,
SUM(QUANTITY) AS CANTIDAD_TOTAL
FROM [ORDER DETAILS]
WHERE ORDERID < 10250
GROUP BY PRODUCTID, ORDERID
WITH ROLLUP
ORDER BY PRODUCTID, ORDERID
```

	PRODUCTID	ORDERID	CANTIDAD_TOTAL
1	NULL	NULL	76
2	11	NULL	12
3	11	10248	12
4	14	NULL	9
5	14	10249	9
6	42	NULL	10
7	42	10248	10
8	51	NULL	40
9	51	10249	40
10	72	NULL	5
11	72	10248	5

Uso de la cláusula GROUP BY con el operador CUBE

Utilizar la cláusula GROUP BY con el operador CUBE para crear y resumir todas las combinaciones posibles de los grupos en función de la cláusula GROUP BY. Utilice la cláusula GROUP BY con el operador ROLLUP para proporcionar datos en un formato relacional estándar.

Ejemplo 1:

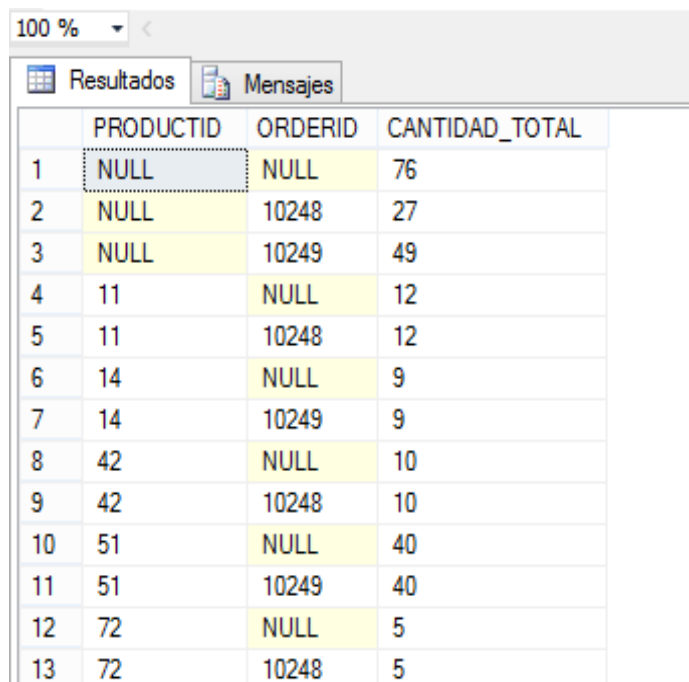
<pre>USE northwind SELECT productid,orderid,SUM(quantity) AS total_quantity FROM orderhist GROUP BY productid,orderid WITH CUBE ORDER BY productid,orderid GO</pre>			
<p>El operador CUBE produce dos resúmenes más de valores que el operador ROLLUP</p>	productid	orderid	total_quantity
	NULL	NULL	95
	NULL	1	30
	NULL	2	65
	1	NULL	15
	1	1	5
	1	2	10
	2	NULL	35
	2	1	10
	2	2	25
	3	NULL	45
	3	1	15
	3	2	30
Descripción			
Total general			
Resume todas las filas para orderid 1			
Resume todas las filas para orderid 2			
Resume sólo las filas para productid 1			
Detalla el valor para productid 1, orderid 1			
Detalla el valor para productid 1, orderid 2			
Resume sólo las filas para productid 2			
Detalla el valor para productid 2, orderid 1			
Detalla el valor para productid 2, orderid 2			
Resume sólo las filas para productid 3			
Detalla el valor para productid 3, orderid 1			
Detalla el valor para productid 3, orderid 2			

Cuando utilice la cláusula GROUP BY con el operador CUBE, considere los hechos e instrucciones siguientes:

- Si tiene n columnas o expresiones en la cláusula GROUP BY, SQL Server devuelve las $2^n - 1$ combinaciones posibles en el conjunto de resultados.
- Los valores NULL del conjunto de resultados indican que dichas filas concretas son el resultado del operador CUBE.
- Cuando utilice el operador CUBE, puede incluir hasta 10 expresiones de agrupamiento.
- Con el operador CUBE no se puede utilizar la palabra clave ALL.
- Cuando utilice el operador CUBE, asegúrese de que las columnas que siguen a la cláusula GROUP BY son las que se han utilizado en la instrucción SELECT menos las que se han utilizado en cualquier función de agregado.

Ejemplo 2:

```
SELECT PRODUCTID, ORDERID,  
SUM(QUANTITY) AS CANTIDAD_TOTAL  
FROM [ORDER DETAILS]  
WHERE ORDERID < 10250  
GROUP BY PRODUCTID, ORDERID  
WITH CUBE  
ORDER BY PRODUCTID, ORDERID
```



	PRODUCTID	ORDERID	CANTIDAD_TOTAL
1	NULL	NULL	76
2	NULL	10248	27
3	NULL	10249	49
4	11	NULL	12
5	11	10248	12
6	14	NULL	9
7	14	10249	9
8	42	NULL	10
9	42	10248	10
10	51	NULL	40
11	51	10249	40
12	72	NULL	5
13	72	10248	5

Funciones de fecha y hora

En la tabla siguiente se enumeran los tipos de datos de fecha y hora de Transact-SQL.

Tipo de datos	Formato
Time	hh:mm:ss[. nnnnnnn]
date	AAAA-MM-DD
smalldatetime	AAAA-MM-DD hh:mm:ss
datetime	AAAA-MM-DD hh:mm:ss[. nnn]

Funciones que obtienen partes de fecha y hora

Función	Sintaxis	Valor devuelto	Tipo de datos devuelto
DATENAME	DATENAME (datepart , date)	Devuelve una cadena de caracteres que representa el datepart especificado de la fecha (date) especificada	nvarchar
DATEPART	DATEPART (datepart , date)	Devuelve un entero que representa el datepart especificado de la fecha (date) especificada	int
DAY	DAY (date)	Devuelve un entero que representa la parte del día de la fecha (date) especificada	int
MONTH	MONTH (date)	Devuelve un entero que representa el mes de una fecha (date) especificada.	int
YEAR	YEAR (date)	Devuelve un entero que representa el año de una fecha (date) especificada	int

Funciones que obtienen diferencias de fecha y hora

Función	Sintaxis	Valor devuelto	Tipo de datos devuelto
DATEDIFF	DATEDIFF (datepart , fechainicio , fechafin)	Devuelve el número de límites datepart de fecha y hora entre dos fechas especificadas.	Int

datepart

Es una parte de la fecha (date) que se devuelve. En la siguiente tabla se describen los argumentos válidos de datepart.

Datepart	Abreviaturas
year	yy, yyyy
quarter	qq, q
month	mm, m
dayofyear	dy, y
day	dd, d
week	wk, ww
weekday	dw, w

hour	hh
minute	mi, n
second	ss, s
millisecond	ms

Funciones de cadenas

Algunas funciones de cadena se describen a continuacion:

Función	Descripción	Sintaxis
CONCAT	Devuelve una cadena que es el resultado de concatenar dos o más valores de cadena.	CONCAT (string_value1, string_value2 [, string_valueN])
LEFT	Devuelve la parte izquierda de una cadena de caracteres con el número de caracteres especificado.	LEFT (character_expression , integer_expression)
RIGHT	Devuelve la parte derecha de una cadena de caracteres con el número de caracteres especificado.	RIGHT (character_expression , integer_expression)
LTRIM	Devuelve una expresión de caracteres tras quitar todos los espacios iniciales en blanco.	LTRIM (character_expression)
RTRIM	Devuelve una cadena de caracteres después de truncar todos los espacios en blanco finales.	RTRIM (character_expression)
LOWER	Devuelve una expresión de caracteres después de convertir en minúsculas los datos de caracteres en mayúsculas.	LOWER (character_expression)
UPPER	Devuelve una expresión de caracteres con datos de caracteres en minúsculas convertidos a mayúsculas.	UPPER (character_expression)
SUBSTRING	Devuelve parte de una expresión de caracteres, binaria, de texto o de imagen en SQL Server 2012.	SUBSTRING (expression ,start , length)

III. Requerimientos

- Máquina con SQL Server 2012
- Guía Número 7 de base de datos

IV. Procedimiento

Parte 1: Iniciando sesión desde SQL Server Managment Studio

1. Hacer clic en el botón **Inicio**
2. Hacer clic en la opción **Todos los programas** y hacer clic en **Microsoft SQL Server 2012**

Para conectarse con el servidor de base de datos elija los siguientes parámetros de autenticación:

- **Tipo de servidor:** Database Engine
- **Nombre del servidor:** Colocar el nombre del servidor local, por ejemplo PCNumMaquina-SALA2
Nota: NumMaquina es el número de la maquina local
- **Autenticación:** SQL Server Authentication
- **Login:** sa

- **Password:** 123456

3. Luego, presione Ctrl+N o de clic en **“New Query”** en la barra de trabajo estándar.

Parte 2. Uso de las funciones de SQL Server

1. Haciendo uso siempre de la base de datos **Northwind**

```
USE NORTHWND
GO
```

FUNCIONES DE AGREGADO

Ejercicio 1. Función AVG

1. Función AVG, devuelve el promedio de los valores de un grupo. Los valores NULL se pasan por alto.
2. En este ejemplo se calcula el precio promedio de todos los productos de la tabla Products, tomando el dato almacenado en el campo UnitPrice

```
SELECT AVG(UnitPrice) AS [Promedio Precio Unitario]
FROM Products
GO
```

Ejercicio 2. Función SUM

1. Función SUM, devuelve la suma de todos los valores o sólo de los valores DISTINCT de la expresión. SUM sólo puede utilizarse con columnas numéricas. Los valores Null se pasan por alto.
2. En este ejemplo se suman todos los datos almacenados en la columna Quantity de la tabla Order Details.

```
SELECT SUM(Quantity) AS [Suma Cantidad]
FROM [Order Details]
GO
```

Ejercicio 3. Función COUNT

COUNT(*)	Devuelve el número de elementos de un grupo. Se incluyen valores NULL y duplicados.
COUNT(ALL expression)	Evalúa expression en todas las filas del grupo y devuelve el número de valores no NULL.
COUNT(DISTINCT expression)	Evalúa expression en todas las filas del grupo y devuelve el número de valores no NULL únicos.

Función COUNT_BIG

COUNT_BIG(*)	Devuelve el número de elementos de un grupo. Esto incluye los valores NULL y los duplicados.
COUNT_BIG(ALL expression)	Evalúa expression en todas las filas del grupo y devuelve el número de valores no NULL.
COUNT_BIG(DISTINCT expression)	Evalúa expression en todas las filas del grupo y devuelve el número de valores no NULL únicos.

1. Función COUNT, devuelve el número de elementos de un grupo. COUNT siempre devuelve un valor de tipo de datos INT. Si los valores devueltos son superiores a $2^{31}-1$, COUNT genera un error. En su lugar, se debe utilizar COUNT_BIG.
2. En este ejemplo se presenta el número de empleados de la tabla Employees.

```
SELECT COUNT(*) AS [Total Empleados]
FROM Employees
GO
```

Se obtendrá el siguiente resultado:

	Total Empleados
1	9

3. Ahora digite la siguiente consulta:

```
SELECT EmployeeID, Region
FROM Employees
GO
```

4. Al ejecutar la consulta se obtendrá los siguientes resultados:

	EmployeeID	Region
1	1	WA
2	2	WA
3	3	WA
4	4	WA
5	5	NULL
6	6	NULL
7	7	NULL
8	8	WA
9	9	NULL

Como observa se muestran 4 resultados NULOS en el campo Region

5. Este ejemplo presenta el número de empleados de la tabla Employees en donde el dato almacenado en el campo Region sea diferente a NULL.

```
SELECT COUNT(ALL Region) AS [Total Empleados]
FROM Employees
GO
```

Al ejecutar la consulta se obtiene el siguiente resultado:

	Total Empleados
1	5

Al final solo cuenta los empleados que tienen asignado una región en este caso WA, el mismo resultado, se obtendría a ejecutar la siguiente consulta:

```
SELECT COUNT(Region) AS [Total Empleados]
FROM Employees
GO
```

6. En el siguiente ejemplo se quiere contar a los empleados que tienen asignado una región, la cual esta no debe repetirse

```
SELECT COUNT(DISTINCT Region) AS [Total Empleados]
FROM Employees
GO
```

Al ejecutar la consulta se obtiene el siguiente resultado:

	Total Empleados
1	1

Ejercicio 4. Función MAX y MIN

1. La función MAX, devuelve el valor máximo de la expresión. Y la función MIN, devuelve el valor mínimo de la expresión. Para las columnas de caracteres, MAX busca el valor más alto de la secuencia de intercalación y MIN busca el valor más bajo en la secuencia de ordenación
2. En el siguiente ejemplo, se muestra como obtener el precio mayor de un producto almacenado en el campo UnitPrice de la tabla Products

```
SELECT MAX(UnitPrice) AS [Precio mas alto]
FROM Products
GO
```

3. Al ejecutar la consulta se obtiene el siguiente resultado:

	Precio mas alto
1	263.50

4. En el siguiente ejemplo, se muestra como obtener el precio menor de un producto almacenado en el campo UnitPrice de la tabla Products

```
SELECT MIN(UnitPrice) AS [Precio mas bajo]
FROM Products
GO
```

5. Al ejecutar la consulta se obtiene el siguiente resultado:

	Precio mas bajo
1	2.50

Ejercicio 5. Uso del GROUP BY

1. En este ejercicio se devuelve información acerca de los pedidos de la tabla Order Details. La consulta agrupa y presenta cada identificador de producto (ProductID) y calcula la cantidad total de pedido por cada producto. La cantidad total se calcula con la función de agregado SUM y presenta un valor para cada producto del conjunto de resultados, y al final se ordena la información con el ORDER BY.

```
SELECT ProductID, SUM(Quantity) AS [Total de Productos]
FROM [Order Details]
GROUP BY ProductID
ORDER BY ProductID ASC
GO
```

2. Este ejemplo agrega una cláusula WHERE a la consulta del ejercicio anterior. Esta consulta restringe las filas al producto cuyo identificador (ProductID), está dentro del rango 10 y 25 y, después, agrupa dichas filas y calcula la cantidad total del pedido. Compare este conjunto de resultados con el ejercicio anterior.

```

SELECT ProductID, SUM(Quantity) AS [Total de Productos]
FROM [Order Details]
WHERE ProductID BETWEEN 10 AND 25
GROUP BY ProductID
ORDER BY ProductID ASC
GO

```

3. Calcular y mostrar la cantidad de clientes que se encuentran por cada país (Country) y región (Region) los cuales están almacenados en la tabla Customers, los resultados se ordenaran por país (Country)

```

SELECT Country, Region, COUNT(*) AS [Total de Clientes]
FROM Customers
GROUP BY Country, Region
ORDER BY Country
GO

```

Este ejemplo muestra la agrupación de 2 campos (Country y Region)

Ejercicio 6. Uso de la cláusula HAVING

1. En este ejemplo se calcula el total de productos (Quantity) que se han realizado en los pedidos almacenados en la tabla Order Details y donde esa cantidad tiene que ser mayor o igual a 100 unidades y las filas se ordenan en forma ascendente según la suma de las cantidades.

```

SELECT ProductID, SUM(Quantity) AS [Total de Productos]
FROM [Order Details]
GROUP BY ProductID
HAVING SUM(Quantity) >= 100
ORDER BY SUM(Quantity)
GO

```

Ejercicio 7. Uso de la cláusula ROLLUP

1. En este ejemplo se presentan todos los registros de la tabla Order Details y los valores de cantidades de resumen por cada producto.

```

SELECT ProductID, OrderID, SUM(Quantity) AS [Total de Productos]
FROM [Order Details]
GROUP BY ProductID, OrderID
WITH ROLLUP
ORDER BY ProductID, OrderID
GO

```


Genera filas de agregado mediante la cláusula GROUP BY simple, más filas de subtotal o de superagregado, así como una fila de total general por la cláusula ROLLUP.

	ProductID	OrderID	Total de Productos
1	NULL	NULL	51317
2	1	NULL	828
3	1	10285	45
4	1	10294	18
5	1	10317	20

Con el ejemplo anterior es similar al uso de las siguientes dos consultas:

Consulta 1:

```
SELECT SUM(Quantity) AS [Total de Productos]
FROM [Order Details]
```

Al ejecutar se obtienen los siguientes resultados:

	Total de Productos
1	51317

La cual muestra la suma de todas las cantidades vendidas registradas en la tabla Orders Details

Consulta 2:

```
SELECT ProductID, SUM(Quantity) AS [Total de Productos]
FROM [Order Details]
GROUP BY ProductID
ORDER BY ProductID
```

Al ejecutar se obtienen los siguientes resultados:

	ProductID	Total de Productos
1	1	828
2	2	1057
3	3	328
4	4	453
5	5	298
6	6	301
7	7	763
8	8	372
9	9	95
10	10	742
11	11	706

Se obtiene un total de 77 registros aproximadamente

En esta consulta se muestra la suma de las cantidades vendidas por cada producto, registradas en la tabla Orders Details

2. En este ejemplo se devuelve información acerca de los pedidos almacenados en la tabla Order Details. Esta consulta contiene una instrucción SELECT con una cláusula GROUP BY sin el operador ROLLUP. El ejemplo devuelve la lista de las cantidades totales de cada producto solicitadas en cada pedido, para los pedidos cuyo orderid sea menor que 10250.

```
SELECT OrderID, ProductID, SUM(Quantity) AS [Total de Productos]
FROM [Order Details]
WHERE OrderID < 10250
GROUP BY OrderID, ProductID
ORDER BY ProductID, OrderID
GO
```

3. En este ejemplo se agrega el operador ROLLUP a la instrucción del ejemplo anterior. El conjunto de resultados incluye la cantidad total para:
- Cada producto en cada pedido (también devuelto por la cláusula GROUP BY sin el operador ROLLUP).
 - Todos los productos de cada pedido.
 - Todos los productos de todos los pedidos (total final).

```
SELECT OrderID, ProductID, SUM(Quantity) AS [Total de Productos]
FROM [Order Details]
WHERE OrderID < 10250
GROUP BY OrderID, ProductID
WITH ROLLUP
ORDER BY ProductID, OrderID
GO
```

Observe en el conjunto de resultados que la fila que contiene NULL en las columnas ProductID y OrderID representa la cantidad total final de todos los pedidos para todos los productos.

	OrderID	ProductID	Total de Productos
1	NULL	NULL	76
2	10248	NULL	27
3	10249	NULL	49

Las filas que contienen NULL en la columna ProductID representan la cantidad de unidades vendidas de un producto por cada OrderID.

	OrderID	ProductID	Total de Productos
1	NULL	NULL	76
2	10248	NULL	27
3	10249	NULL	49

Ejercicio 8. Uso de la cláusula CUBE

1. Digitar la siguiente consulta para comprobar la cláusula CUBE, los datos a mostrar son :

- a. ProductID
- b. OrderID
- c. La suma de las unidades vendidas (Quantity) por cada producto

```
SELECT ProductID, OrderID, SUM(Quantity) AS [Total de Productos]
FROM [Order Details]
GROUP BY OrderID, ProductID
WITH CUBE
ORDER BY ProductID, OrderID
GO
```

2. Al ejecutar la consulta se obtiene el siguiente resultado:

- a. La cantidad total final de todos los productos en todos los pedidos.

	ProductID	OrderID	Total de Productos
1	NULL	NULL	51317
2	NULL	10248	27
3	NULL	10249	49

- b. La cantidad de todos los productos en cada pedido, independientemente que producto sea

	ProductID	OrderID	Total de Productos
1	NULL	NULL	51317
2	NULL	10248	27
3	NULL	10249	49

- c. La cantidad total de cada producto en todos los pedidos, independientemente en que pedido se encuentre el producto

	ProductID	OrderID	Total de Productos
828	NULL	11074	14
829	NULL	11075	42
830	NULL	11076	50
831	NULL	11077	72
832	1	NULL	828
833	1	10285	45
834	1	10294	18
835	1	10317	20
836	1	10348	15
837	1	10354	12

d. La cantidad total de cada producto por cada pedido

	ProductID	OrderID	Total de Productos
828	NULL	11074	14
829	NULL	11075	42
830	NULL	11076	50
831	NULL	11077	72
832	1	NULL	828
833	1	10285	45
834	1	10294	18
835	1	10317	20
836	1	10348	15
837	1	10354	12

Ejercicio 9. Uso de la cláusula GROUPING SETS

Una cláusula GROUP BY que utiliza GROUPING SETS puede generar un conjunto de resultados equivalente al generado por una cláusula UNION ALL (COMBINA los resultados de dos o más consultas en un SOLO CONJUNTO de resultados), o varias cláusulas GROUP BY simples.

1. El siguiente ejemplo muestra el total de proveedores por cada País y por cada Ciudad

```
SELECT Country, City, COUNT(*) AS [Total de Proveedores]
FROM Suppliers
GROUP BY GROUPING SETS ((Country),(City))
GO
```

	Country	City	Total de Proveedores
1	NULL	Ann Arbor	1
2	NULL	Annecy	1
3	NULL	Bend	1
4	NULL	Berlin	1
5	NULL	Boston	1
6	NULL	Cuxhaven	1
7	NULL	Frankfurt	1

	Country	City	Total de Proveedores
30	Australia	NULL	2
31	Brazil	NULL	1
32	Canada	NULL	2
33	Denmark	NULL	1
34	Finland	NULL	1
35	France	NULL	3
36	Germany	NULL	3
37	Italy	NULL	2

2. Ahora digite la siguiente consulta y compare con los resultados de la anterior

```
SELECT Country, NULL AS City, COUNT(*) AS [Total de Proveedores]
FROM Suppliers
GROUP BY Country
UNION ALL
SELECT NULL AS Country, City, COUNT(*) AS [Total de Proveedores]
FROM Suppliers
GROUP BY City
GO
```

Verifique el total de proveedores para el país USA y compárelo con el resultado que se obtuvo en la consulta del punto 1, es el mismo o diferente ¿?

Funciones de Fecha y Hora

Ejercicios.

1. En el siguiente ejemplo se quiere obtener el año de una fecha específica

```
SELECT YEAR('2014-03-12 17:06:00.390') AS Año
GO
```

2. Utilizando la misma funcion YEAR se desea obtener el año de la fecha actual, utilizando la funcion GETDATE()

```
SELECT YEAR(GETDATE()) AS Año  
GO
```

3. Obteniendo el mes de la fecha actual

```
SELECT MONTH(GETDATE()) AS Mes;  
GO
```

4. Obteniendo el dia de la fecha actual

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

5. Obteniendo la hora de la fecha actual

```
SELECT DATENAME(HOUR,GETDATE()) AS Hora  
GO
```

6. Obteniendo el minuto de la fecha actua

```
SELECT DATENAME(MINUTE,GETDATE()) AS Minuto  
GO
```

7. Obteniendo el segundo de la fecha actual

```
SELECT DATENAME(SECOND,GETDATE()) AS Segundo  
GO
```

Funciones de cadena

1. Para los siguientes ejercicios se utilizara la base de datos **AdventureWorks2012**

```
USE AdventureWorks2012
```

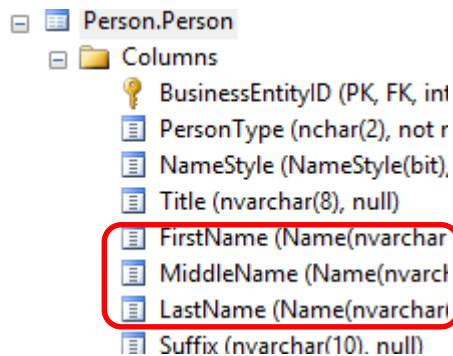
2. En el ejemplo siguiente se devuelven los cinco primeros caracteres situados más a la izquierda de cada nombre de producto.

```
SELECT LEFT(Name, 5) AS [Nombre Producto]
FROM Production.Product
ORDER BY ProductID;
GO
```

3. En el ejemplo siguiente se devuelven los cinco caracteres situados más a la derecha (últimos cinco) del nombre de cada producto

```
SELECT RIGHT(Name, 5) AS [Nombre Producto]
FROM Production.Product
ORDER BY ProductID;
GO
```

4. En el siguiente ejemplo se muestran en una sola columna la unión de tres campos, la tabla que se ha tomado es Person.Person



Los campos a concatenar son: FirstName, MiddleName (en muchos registros este dato es NULL) y LastName, digitar la siguiente consulta

```
SELECT CONCAT (FirstName, ' ', MiddleName, ' ', LastName, ' ')
AS [Nombre Completo]
FROM Person.Person
GO
```

5. En el siguiente ejemplo se muestran los nombres de los productos cuyos precios sean entre 11 y 20 , estos nombres se muestran en minúscula y mayúscula respectivamente

```
SELECT LOWER(Name) AS [Nombre en minuscula],
UPPER(Name) AS [Nombre en mayuscula]
FROM Production.Product
WHERE ListPrice between 11.00 and 20.00
GO
```

6. En el siguiente ejemplo se muestra cómo devolver únicamente una parte de una cadena de caracteres, se está obteniendo el primero carácter del dato almacenado en el campo FirstName

```
SELECT FirstName, SUBSTRING(FirstName, 1, 1) AS Inicial
FROM Person.Person
ORDER BY LastName
GO
```

Colocar como nombre al script **Guia7_Procedimiento_SuCarnet.sql**

V. Ejercicio complementario

Haciendo uso de la base de datos **Northwind** realice las siguientes consultas:

1. Mostrar cuantos Clientes hay por cada Compañía (CompanyName)
2. Seleccionar los campos: OrderID, CustomerID, EmployeeID , OrderDate y ShippedDate de la tabla Orders, donde la fecha del pedido (OrderDate) sea del año 1996.

Implemente la función YEAR

3. Se desea conocer cuántos empleados hay por cada Territorio, utilice la tabla EmployeeTerritories
4. Tomando como base de la consulta del punto 3, crear una consulta donde implemente la instrucción ROLLUP.
5. Se desea conocer cuántos territorios hay por cada región (RegionID), utilice la tabla Territories
6. Mostrar de la tabla Order Details aquellos pedidos en donde las unidades sumen más de 50 y ordenar los datos en forma descendente según la suma de esas cantidades.

Haciendo uso de la base de datos **AdventureWorks2012** realice las siguientes consultas:

Utilizando la tabla **Sales.SalesOrderDetail** realice las siguientes consultas:

1. Mostrar la suma de las unidades vendidas (OrderQty) por cada orden (SalesOrderID)
2. Mostrar el promedio de ventas (LineTotal) por cada orden (SalesOrderID)
3. Mostrar la venta (LineTotal) máxima por cada orden (SalesOrderID)

- Mostrar la venta (LineTotal) mínima por cada orden (SalesOrderID)

Utilizando la tabla **Sales.Store** realice las siguientes consultas:

- Mostrar los datos de la tabla y ordenarlos por el campo SalesPersonID
- Mostrar en una sola columna el nombre de la tienda (Name) y el código del vendedor encargado de la tienda (SalesPersonID)
- Implementar la función UPPER o LOWER (ud seleccione el campo)

Colocar como nombre al script **Guia7_EjercicioComplementario_SuCarnet.sql**

VI. Investigación complementaria

Tarea individual:

- Creación de base de datos:
Nombre de la base de datos: **Control_de_libros**
- Crear las tablas tomando en cuenta:
 - Crear las relaciones entre las tablas (llaves primarias y llaves foráneas).
 - En las relaciones de las tablas implementar las instrucciones ON DELETE CASCADE y ON UPDATE CASCADE
 - Deberá implementar también las restricciones (CHECK, UNIQUE y DEFAULT) en los campos de cada tabla que ud. cree necesarias.
- El formato de las tablas son:

Tabla Autor:

CodigoAutor	PrimerNombre	PrimerApellido	FechaNacimiento	Nacionalidad	Edad
PL001	Pablo	López	19/08/1960	Colombiana	54
CM002	Claudia	Martínez	10/06/1970	Salvadoreña	45
PM003	Patricio	Murry	12/12/1967	Española	47
NH004	Nuria	Hernández	03/09/1980	Colombiana	34
HM005	Helen	Martínez	22/11/1980	Española	34
JR006	José	Roldan	13/09/1967	Colombiano	54

Tabla Editorial:

CodigoEditorial	Nombre	Pais
ED001	Omega 2000	Colombia
ED002	Anaya Multimedia	España
ED003	McGrawHill	Inglaterra
ED004	Reyes	México
ED005	Prentice Hall	Inglaterra

Tabla Libro:

CodigoLibro	Título	ISBN	AñoEdicion	CodigoEditorial
BDCOL00001	Fundamentos de Base de datos	12333-8999988	2004	ED001
BDESP00002	La Biblia de SQL Server 2008	3444-99888-88	2008	ED002
PRCOL00002	Programación orientada a objetos	8999-9999444	2011	ED001
DWING00003	Diseño Web y Hojas de estilo	300096-99999	2010	ED003
PRING00004	Programación en C/C++	45667-87878	2009	ED005
HJMEX00005	Uso de hojas de estilo con JavaScript	0990-87878787	2008	ED004
ABESP00006	Administración de Base de datos	585885-88484848	2010	ED002

Tabla Detalle_AutorLibro

CodigoAutor	CodigoLibro
PL001	BDCOL00001
NH004	BDCOL00001
CM002	PRCOL00002
PM003	BDESP00002
PM003	DWING00003
HM005	PRING00005
CM002	ABESP00006
NH004	HJMEX00005
JR006	DWING00003

4. Agregar los registros a las tablas
5. Realizar las siguientes consultas, implementando funciones de agregado o funciones de cadenas:
 - a. Mostrar cuantos autores hay por cada nacionalidad
 - b. Calcular cuántos libros hay por cada editorial
 - c. Mostrar la cantidad de editoriales que hay por cada país
 - d. Mostrar el libro donde el año de edición sea el más actual

- e. Mostrar el libro donde el año de edición sea el menos actual
- f. Calcular el promedio de las edades de los autores
- g. Mostrar cuántos libros ha escrito cada autor
- h. Mostrar cuantos autores nacieron en el mismo mes
- i. Mostrar el nombre y apellido del AUTOR , haciendo uso de la función CONCAT
- j. Contar cuantos libros tienen en su título la palabra PROGRAMACION
- k. Mostrar las iniciales del Primer apellido de cada AUTOR y ordenarlos de forma descendente
- l. Mostrar los autores que nacieron antes del año 1980
- m. Se desea conocer aquellos autores que han escrito más de un libro, mostrar el autor y la cantidad de libros
- n. Mostrar cuantos autores comienzan con la inicial P en su primer apellido
- o. Mostrar cuantos libros finalizan con la palabra Base de datos en su título

VII. Fuente de consulta

1. Microsoft SQL Server 2008: Guía del Administrador
Madrid, España: ANAYA, 2009
Autor: William Stanek
Biblioteca UDB – Clasificación: 005.361 S784 2009
2. [https://msdn.microsoft.com/es-es/library/ms177677\(v=sql.110\).aspx](https://msdn.microsoft.com/es-es/library/ms177677(v=sql.110).aspx)
3. [https://msdn.microsoft.com/es-es/library/ms181984\(v=sql.110\).aspx](https://msdn.microsoft.com/es-es/library/ms181984(v=sql.110).aspx)
4. <http://blog.sqlauthority.com/2012/04/02/sql-server-use-roll-up-clause-instead-of-compute-by/>
5. <http://www.ingenieriasystems.com/2015/05/Procedimientos-para-agrupar-y-resumir-datos-en-SQL-Server.html>
6. <http://es.slideshare.net/osmar201291/sql-server-express-edition-2012-libro>