SQL para Análisis de Datos

Módulo IV

Temario

- SQL – Funciones
  - Numéricas (De Agregado)
  - De Texto
  - Fecha y Hora
  - Lógicas
  - De Existencia
  - Agregado
    - GROUP BY
    - HAVING

SQL > Funciones > Numéricas

Función	Significado
COUNT()	Cuenta los registros que no sean nulos
SUM()	Suma campos numéricos
MAX()	Retorna el valor máximo
MIN()	Retorna el valor mínimo
AVG()	Retorna el promedio
ABS()	Retorna el valor absoluto

SQL > Funciones > Numéricas

Función	Significado
ROUND()	Realiza el 'redondeo' de una cifra
CEILING()	Retorna el próximo valor entero de una cifra
FLOOR()	Retorna el valor entero anterior de una cifra
CAST()	Convierte un tipo de dato a otro *
SQRT()	Retorna la raíz cuadrada de un número
RAND()	Retorna un número pseudo-aleatorio

* Funciona con otros tipos de datos (no numéricos)

COUNT():

- Cuenta los registros de una tabla o los registros no nulos de un cierto campo
- Ejemplo:

```
SELECT COUNT(*) AS 'Cantidad de Empleados'
FROM HumanResources.Employee
```

	Cantidad de Empleados
1	290

- En este caso se obtiene la cantidad de empleados (registros) de la tabla HumanResources.Employee

COUNT():

- Ejemplo:

```
SELECT COUNT(*) AS 'Cantidad de Productos'
FROM Production.Product
```

	Cantidad de Productos
1	504

```
SELECT COUNT(Color) AS 'Productos con Colores (no NULL)'
FROM Production.Product
```

	Productos con Colores (no NULL)
1	256

- En este caso se nota la diferencia en contar el total de productos y contar los productos en base a un campo (no se cuentan los valores nulos)

SUM():

- Suma campos numéricos
- Ejemplo:

```
SELECT SUM(ListPrice) AS SUMA
FROM Production.Product
WHERE Color IS NOT NULL
      AND ListPrice != 0.00
      AND Name LIKE 'Mountain%'
```

	SUMA
1	53886,68

- En este caso se obtiene la suma de los precios de lista de los productos que tienen un color en la tabla, su precio de lista es distinto de 0 (cero) y su nombre comienza con 'Mountain'

MAX():

- Retorna el valor máximo de un campo (funciona también con campos de texto)
- Ejemplo:

```
SELECT MAX(ListPrice) AS MayorPrecio
FROM Production.Product
```

	MayorPrecio
1	3578.27

- En este caso se obtiene el mayor precio de lista entre todos los productos

AVG():

- Retorna el valor promedio de todos los valores de una cierta columna
- Ejemplo:

```
SELECT AVG(VacationHours)AS 'Promedio de horas de vacaciones',
SUM(SickLeaveHours) AS 'Total de horas enfermos'
FROM HumanResources.Employee
```

	Promedio de horas de vacaciones	Total de horas enfermos
1	50	13139

- En este caso se obtiene el promedio de las horas de vacaciones de todos los empleados y la suma de todas las horas que se registraron de ausencias por enfermedad

ROUND():

- Realiza el 'redondeo' de una cifra en base a 2 parámetros:
  - Precisión y tipo de redondeo (opcional)
- Ejemplos:

```
SELECT ROUND(123.9994, 3), ROUND(123.9995, 3)
```

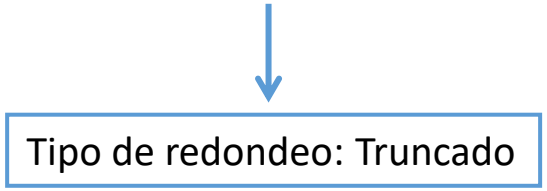
1	123.9990	124.0000

```
SELECT ROUND(123.9994, -1), ROUND(123.9995, -2)
```

1	120.0000	100.0000

```
SELECT ROUND(150.75, 0), ROUND(150.75, 0, 1);
```

1	151.00	150.00



CEILING():

- Retorna el próximo valor entero de una cifra
- Ejemplo:

```
SELECT CEILING(123.45), CEILING(-123.45)
```

1	124	-123

FLOOR():

- Retorna el valor entero anterior de una cifra
- Ejemplo:

```
SELECT FLOOR(123.45), FLOOR(-123.45)
```

1	123	-124

CAST():

- Sirve para convertir tipos de datos
- Ejemplos:

```
SELECT CAST(10.6496 AS int)
```

1	10

```
SELECT Name, ListPrice
FROM Production.Product
WHERE CAST(ListPrice AS int) LIKE '3%'
```

	Name	ListPrice
1	Sport-100 Helmet, Red	34,99
2	Sport-100 Helmet, Black	34,99
3	Sport-100 Helmet, Blue	34,99
4	LL Road Frame - Black, 58	337,22
5	LL Road Frame - Black, 60	337,22
6	LL Road Frame - Black, 62	337,22
7	LL Road Frame - Red, 44	337,22
8	LL Road Frame - Red, 48	337,22

SQL > Funciones > Texto

Función	Significado
CONCAT()	Concatena cadenas de texto
LTRIM() - RTRIM()	Suprime espacios en blanco en un campo de texto
REPLACE()	Remplaza caracteres dentro de una cadenas de texto
SUBSTRING()	Devuelve parte de una cadena de texto
LOWER()	Pasa una cadena de texto a minúscula
UPPER()	Pasa una cadena de texto a mayúscula
LEN()	Obtiene el largo de una cadena

CONCAT():

- Concatena cadenas de texto
- Ejemplo:

```
SELECT CONCAT(LastName, ', ' ,FirstName) AS Name
FROM Person.Person
ORDER BY LastName ASC, FirstName ASC
```

	Name
1	Abbas, Syed
2	Abel, Catherine
3	Abercrombie, Kim
4	Abercrombie, Kim
5	Abercrombie, Kim

- Funciona de manera similar al operador '+' con texto

LTRIM() – RTRIM():

- LTRIM(): suprime espacios en blanco delante del texto (left)
- RTRIM(): suprime espacios en blanco detrás de texto (right)

- Ejemplos:

```
SELECT '     hola     '

SELECT LTRIM('     hola     ')
```

	(No column name)
1	hola

	(No column name)
1	hola

LTRIM() – RTRIM():

- Ejemplos:

```
SELECT *
FROM (SELECT ' Pedro ' as nombre) as Newtable
WHERE nombre LIKE 'p%'
```

nombre

```
SELECT *
FROM (SELECT ' Pedro ' as nombre) as Newtable
WHERE LTRIM(nombre) LIKE 'p%'
```

	nombre
1	Pedro

REPLACE():

- Reemplaza todas las ocurrencias de una cadena especificada con otra cadena
- Ejemplo:

```
SELECT REPLACE('abcdefghicde','cde','xxx')
```

1	abxxxfghixxx

SUBSTRING():

- Devuelve parte de una cadena de texto, especificando comienzo y largo de la cadena requerida
- Ejemplo:

```
SELECT LastName, SUBSTRING(FirstName, 1, 1) AS Initial
FROM Person.Person
```

	LastName	Initial
1	Abbas	S
2	Abel	C
3	Abercrombie	K
4	Abercrombie	K
5	Abercrombie	K

```
SELECT x = SUBSTRING('abcdef', 2, 3)
```

	x
1	bcd

- SUBSTRING(texto, comienzo, largo)

UPPER():

- Pasa una cadena de texto a mayúscula
- Ejemplo:

```
SELECT LastName, UPPER(FirstName) as FirstUpperName
FROM Person.Person
```

	LastName	FirstUpperName
1	Abbas	SYED
2	Abel	CATHERINE
3	Abercrombie	KIM
4	Abercrombie	KIM
5	Abercrombie	KIM

LEN():

- Obtiene el largo de una cadena de caracteres
- Ejemplo:

```
SELECT LastName, LEN(LastName) AS Largo
FROM Person.Person
```

	LastName	Largo
1	Abbas	5
2	Abel	4
3	Abercrombie	11
4	Abercrombie	11
5	Abercrombie	11
6	Abolrous	8

SQL > Funciones > Fecha y Hora

Función	Significado
YEAR()	Retorna un entero que representa el año de una fecha
MONTH()	Retorna un entero que representa el mes de una fecha
DAY()	Retorna un entero que representa el día de una fecha
GETDATE()	Retorna el día de la fecha del sistema
DATEDIFF()	Retorna la diferencia entre 2 fechas
DATEADD()	Agrega a una fecha el intervalo especificado

YEAR():

- Retorna un entero que representa el año de una fecha
- Ejemplo:

```
SELECT BirthDate, YEAR(BirthDate) AS Año
FROM HumanResources.Employee
WHERE YEAR(BirthDate) > 1970
```

	BirthDate	Año
1	1977-03-27	1977
2	1976-07-06	1976
3	1975-01-01	1975
4	1979-06-29	1979
5	1977-06-03	1977

GETDATE():

- Retorna el día de la fecha del sistema
- Ejemplo:

```
SELECT GETDATE() AS Ahora
```

	Ahora
1	2016-09-12 02:27:27.863

- Obtiene la Fecha y la Hora del momento en que se ejecuta la consulta

DATEDIFF():

- Retorna la diferencia entre 2 fechas, especificando la unidad que se desea (años, meses, días, horas, minutos, segundos, etc.)
- Ejemplo:

```
SELECT DATEDIFF(day, '2016-09-11 23:59:59.9999999' ,
'2016-09-15 00:00:00.0000000')
```

	(No column name)
1	4

- DATEDIFF(Unidad, FechaInicio, FechaFinal) -> FechaFinal - FechaInicio

DATEADD():

- Agrega a una fecha el intervalo especificado
- Ejemplo:

```
SELECT DATEADD(month, 3, '2016-08-30')
```

	(No column name)
1	2016-11-30 00:00:00.000

- DATEADD(Unidad, Incremento, FechaInicial)

SQL > Funciones > Lógicas

Función	Significado
IIF()	Evalúa una expresión y retorna lo que se le especifica en el caso que sea Verdadera o Falsa
CHOOSE()	Retorna el elemento en la posición indicada

IIF():

- Evalúa una expresión y retorna lo que se le especifica en el caso que sea Verdadera o Falsa
- Ejemplo:

```
SELECT BirthDate, IIF(YEAR(BirthDate)<1950, 'Mayor',
'Joven') AS Es
FROM HumanResources.Employee
```

	BirthDate	Es
1	1959-03-02	Joven
2	1961-09-01	Joven
3	1964-12-13	Joven
4	1965-01-23	Joven
5	1942-10-29	Mayor
6	1949-04-11	Mayor
7	1977-03-27	Joven

- IIF(Expresión, Cuando_es_V, Cuando_es_F)

CHOOSE():

- Retorna el elemento en la posición indicada
- Ejemplo:

```
SELECT CHOOSE ( 3, 'Manager', 'Director', 'Developer',
'Tester' ) AS Result
```

	Result
1	Developer

- CHOOSE (index, val_1, val_2 [, val_n])

SQL > Funciones > De Existencia

Función	Significado
ISNULL()	Evalúa si un campo es nulo, permitiendo remplazar los valores nulos por un valor predeterminado

ISNULL():

- Evalúa si un campo es nulo, permitiendo remplazar los valores nulos por un valor predeterminado
- Ejemplo:

```
SELECT AVG(ISNULL(Weight, 50))  AS Promedio
FROM Production.Product
```

	Promedio
1	59.790059

- Se utiliza mucho cuando se aplican funciones que no toman en cuenta valores nulos en los campos, pero se considera necesario que estos valores nulos repersenten un cierto valor (no nulo)

GROUP BY:

- Se utiliza para agrupar el resultado de una consulta por el campo que necesitemos
- Se utiliza generalmente en las funciones de agregado
- Su sintaxis es:

```
SELECT "nombre_columna1", FUNCIÓN_DE_ AGREGADO("nombre_columna2")
FROM "nombre_tabla"
GROUP BY "nombre-columna1"
```

GROUP BY:

- Se utiliza generalmente con funciones de agregado:
  - **SUM, COUNT, MIN, MAX, AVG**
- Ejemplo:

```
SELECT CustomerID ,MAX(TotalDue) AS 'Máxima Compra'
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
ORDER BY CustomerID
```

- Se obtiene la compra de mayor valor por Cliente

	CustomerID	Máxima Compra
1	11000	3756,989
2	11001	3729,364
3	11002	3756,989
4	11003	3756,989
5	11004	3756,989
6	11005	3729,364

GROUP BY:

- Ejemplo:

```
SELECT CustomerID ,SUM(TotalDue) AS 'Suma de Compras'
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
ORDER BY CustomerID
```

	CustomerID	Suma de Compras
1	11000	9115,1341
2	11001	7054,1875
3	11002	8966,0143
4	11003	8993,9155
5	11004	9056,5911
6	11005	8974,0698

- Se Obtiene la Suma de todas las compras por Cliente

GROUP BY:

- Ejemplo:

```
SELECT MONTH(DueDate) AS MES, CustomerID ,SUM(TotalDue)
AS 'Suma de Compras'
FROM Sales.SalesOrderHeader
GROUP BY CustomerID, MONTH(DueDate)
ORDER BY CustomerID
```

	MES	CustomerID	Suma de Compras
1	8	11000	6344,8659
2	11	11000	2770,2682
3	6	11001	650,8008
4	7	11001	3729,364
5	8	11001	2674,0227
6	7	11002	6292,953

- Se obtiene la suma de todas las compras por cliente y por mes (sin importar el año)

GROUP BY:

- Ejemplo sin función de agregado:

```
SELECT ListPrice
FROM Production.Product
GROUP BY ListPrice
```

	ListPrice
1	0,00
2	2,29
3	3,99
4	4,99
5	7,95
6	8,99

- En este ejemplo se obtienen los mismos resultados que si se utiliza la consulta SELECT DISTINCT

HAVING:

- Especifica una condición de búsqueda para un grupo o agregado
- Solo aparecen en el resultado de la consulta los grupos que cumplen las condiciones HAVING
- Solo puede aplicar una cláusula HAVING a las columnas que también aparecen en la cláusula GROUP BY o en una función de agregado
- HAVING solo se puede utilizar con la instrucción SELECT

HAVING:

- Sintaxis completa de una consulta con HAVING:

```
SELECT "nombre_columna1", FUNCIÓN_DE_AGREGADO("nombre_columna2")
FROM "nombre_tabla"
WHERE "nombre_columna1"   operador   valor
GROUP BY "nombre-columna1"
HAVING FUNCIÓN_DE_AGREGADO("nombre_columna2")   operador   valor
ORDER BY FUNCIÓN_DE_AGREGADO("nombre_columna2") o "nombre_columna1"
```

HAVING:

- Ejemplo:

```
SELECT CustomerID ,SUM(TotalDue) AS 'Suma de Compras'
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
HAVING SUM(TotalDue) > 10000
ORDER BY CustomerID
```

	CustomerID	Suma de Compras
1	11237	11675,85
2	11241	12673,4532
3	11242	12230,1493
4	11245	11691,2869
5	11246	11685,7397
6	11249	10928,7485
7	11412	10917,3559

- Se Obtiene la Suma de todas las compras por Cliente, de los que hayan comprado más de $10000

HAVING:

- Ejemplo:

```
SELECT CustomerID ,SUM(TotalDue) AS 'Suma de Compras'
FROM Sales.SalesOrderHeader
WHERE CustomerID LIKE '1142%'
GROUP BY CustomerID
HAVING SUM(TotalDue) > 10000
ORDER BY CustomerID
```

	CustomerID	Suma de Compras
1	11420	12376,8506
2	11421	10806,9444
3	11423	10944,992
4	11425	11634,1343
5	11427	10959,3791
6	11428	10999,4574
7	11429	11567,5595

- Se Obtiene la Suma de todas las compras por Cliente, de los que hayan comprado más de $10000 y su ID comience con 1142

Resumen Módulo IV

- SQL – Funciones
  - Numéricas (De Agregado)
  - De Texto
  - Fecha y Hora
  - Lógicas
  - De Existencia
  - Agregado
    - GROUP BY
    - HAVING