Víctor Manuel Ridao Chaves
07/03/2025
1º DAW ADAits

CONSULTAS DE BASES DE DATOS

ÍNDICE

Alejandro Márquez Romero

1. AES_ENCRYPT :

La función AES_ENCRYPT en SQL se utiliza para cifrar datos usando el algoritmo 'AES'. Este cifrado convierte información sensible en un formato que no puede ser leída sin la clave de cifrado correspondiente. Es útil para proteger datos como contraseñas o información personal.

EJEMPLO:

```
SELECT AES_ENCRYPT('Pepe', 'clave_secreta')
AS nombre_secreto
FROM usuario;
```

En el ejemplo justo arriba:

***'Pepe'** es el dato que queremos cifrar.*
***'clave_secreta'** es la cadena que se utiliza para cifrar **'Pepe'** , y esta cadena también es necesaria para descifrar **'Pepe'** utilizando otra función.*
***'nombre_secreto'** es el alias para llamar al resultado como queramos, en mi caso, quería llamarlo **'nombre_secreto'**.*

2. RTRIM Y LTRIM :

- La función RTRIM elimina los espacios en blanco al final de una cadena de texto.

- La función LTRIM hace justo lo contrario que la función RTRIM. LTRIM elimina los espacios en blanco al inicio de una cadena de texto.

EJEMPLO CON 'RTRIM':

```
SELECT RTRIM('     hola mundo     ')
AS resultado;
```

En el ejemplo justo arriba el resultado sería el siguiente:

'   hola mundo'

EJEMPLO CON 'LTRIM':

```
SELECT LTRIM('     hola mundo     ')
AS resultado;
```

En el ejemplo justo arriba el resultado sería:
'hola mundo   '

3. AES_DECRYPT :

Esta función en sql se utiliza para desencriptar datos que fueron previamente cifrados con el algoritmo AES.
Esta función toma el dato cifrado y lo convierte nuevamente a su formato original, usando la misma clave de cifrado que se utilizó al encriptarlo.

EJEMPLO:

```
SELECT AES_DECRYPT('Pepe', 'clave_secreta')
AS nombre_secreto
FROM usuario;
```

En el ejemplo justo arriba:

***'Pepe'** es el dato que ya está cifrado en la tabla usuario y queremos descifrar.*
***'clave_secreta'** es la clave que se utilizó para cifrar el dato.*
***'nombre_secreto'** es el resultado de la función, que contiene el correo electrónico en su formato original.*

4. CAST :

La función CAST en SQL se utiliza para convertir un valor de un tipo de dato a otro.

EJEMPLO:

```
SELECT CAST('123' AS INT)
AS numero;
```

En el ejemplo justo arriba:

***'123'** es un valor de tipo String.*

***'AS INT'** indica que queremos convertir ese valor a un tipo entero.*
*'**número**' será el resultado, que en este caso es el valor numérico **'123'**.*

5. INSTR :

La función INSTR en SQL se utiliza para buscar la posición de una subcadena dentro de una cadena más grande.
Devuelve la posición en la que comienza la primera aparición de la subcadena, y si no se encuentra, devuelve 0 .

EJEMPLO:

```
SELECT INSTR('correo@deEjemplo.com', '@')
AS posición;
```

En el ejemplo justo arriba:

***'correo@deEjemplo.com'** es la cadena donde se busca la subcadena **'@'**.*
***INSTR** devolverá la posición en la que aparece la subcadena **'@'** en la cadena.*
*El resultado sería **'7'** , porque el símbolo **'@'** está en la octava posición de la cadena.*

6. Pi :

Esta función devuelve el número 'Pi'.

EJEMPLO:

```
SELECT PI();
```

En el ejemplo justo arriba:

*En Heidi, redondea el número PI y solo devuelve **'3,141593'**.*

Bartłomiej Krzysztof Chudy

1. RAND:

Esta consulta genera un número entero aleatorio entre 1 y 100:

```
SELECT FLOOR(1 + RAND() * 100)
AS numero_Aleatorio;
```

2. DATEDIFF:

Esta función calcula la diferencia entre dos fechas:

```
SELECT DATEDIFF(&#39;2025-03-10&#39;, &#39;2025-03-01&#39;)
AS
dias_diferencia;
```

3. BIT_COUNT( ):

Esta función cuenta el número de bits en 1 en la representación binaria de un número:

```
SELECT BIT_COUNT(13)
AS bits_en_uno;
```

4. PERIOD_DIFF( ):

Esta función calcula la diferencia en meses entre dos fechas en formato AAAAMM(Año y Mes):

```
SELECT PERIOD_DIFF(202503, 202401)
AS meses_diferencia;
```

5. SLEEP( ):

Esta función pausa la ejecución de la consulta durante el número de segundos que se indique:

```
SELECT SLEEP(5);
```

6. DATE_SUB( ):

Esta función resta un intervalo de tiempo de una fecha dada:

```
SELECT DATE_SUB(&amp;#39;2025-02-10&amp;#39;, INTERVAL 3 DAY)
AS
nueva_fecha;
```

7. ARRAY_AGG( ):

Se utiliza para agrupar valores de una columna en un array.

```
SELECT ARRAY_AGG(nombre)
FROM empleados;
```

8. SHA1( ):

Genera un valor hash de tipo SHA-1 (Secure Hash Algorithm 1) para una cadena de texto dada.

```
SELECT SHA1(&amp;#39;miContraseñaSecreta&amp;#39;)
AS hash_valor;
```

9. TIME( ):

Se utiliza para extraer la parte de la hora de una fecha o convertir un valor a solo hora, minutos y segundos.

```
SELECT TIME(&amp;#39;2025-02-16 12:34:56&amp;#39;)
AS solo_hora;
```

10. BIT_LENGTH( ):

Devuelve la longitud en bits de una cadena de texto, es decir, cuántos bits se necesitan para almacenar esa cadena en su formato binario.

```
SELECT BIT_LENGTH(&#39;Hola&#39;)
 AS longitud_en_bits;
```

11. UUID( ):

Genera un Identificador Único Universal.

```
SELECT UUID( )
AS identificador_unico;
```

Francisco Mercado Aguilar

1. REPEAT( ):

Repite un string varias veces.

```
SELECT REPEAT(&#39;SQL&#39;, 3);
```

2. CONV( ):

Convierte un número de una base a otra (decimal, binario, hexadecimal, etc.)

```
SELECT CONV(255, 10, 16);
```

3. DAYOFYEAR( ):

Devuelve el número de día en el año (1-366).

```
SELECT DAYOFYEAR(&#39;2025-02-05&#39;);
```

4. MAKEDATE( ):

Crea una fecha a partir de un año y un número de día en el año.

```
SELECT MAKEDATE(2025, 36);
```

5. ST_DISTANCE( ):

Calcula la distancia entre dos puntos geoespaciales.

```
SELECT ST_Distance(

   ST_GeomFromText(&#39;POINT(0 0)&#39;),
   ST_GeomFromText(&#39;POINT(3 4)&#39;)
);
```

6. LAST_DAY( ):

Devuelve el último día del mes para la fecha dada.

SELECT LAST_DAY('2025-02-05');

7. ST_AREA( ):

Calcula el área de una figura geométrica.

SELECT ST_Area(ST_GeomFromText('POLYGON((0 0, 4 0, 0 4, 4 4,))'));

8. ELT( ):

Devuelve el valor de una lista según el índice dado.

SELECT ELT(2, 'Rojo', 'Verde', 'Azul');

Hernán Pedraza Torres

1. ROUND():

Esta función se utiliza para redondear un número a un número específico de decimales.

```
SELECT ROUND(4.68778, 2);
```

2. ASCII():

Esta función devuelve el valor de la letra en la tabla ASCII del primer carácter de un string.

```
SELECT ASCII(&#39;C&#39;);
```

3. SPACE():

La función genera un string con un número determinado de espacios.

```
SELECT SPACE(8);
```

4. CONCAT_WS():

Esta función concatena strings utilizando cualquier símbolo introducido como separador.

```
SELECT CONCAT_WS(&#39;**&#39;, &#39;Trabajo&#39;, &#39;SQL&#39;,
&#39;!&#39;);
```

5. FIELD():

Esta función devuelve la posición de una cadena específica en una lista de cadenas.

```
SELECT FIELD(&#39;cabra&#39;, &#39;a&#39;, &#39;cabra&#39;, &#39;c&#39;,
&#39;d&#39;);
```

6. CONVERT():

La función se utiliza para convertir datos a otros tipos de datos.

```
SELECT CONVERT(&#39;165&#39;, UNSIGNED);
```

Javier Manzano Oliveros

1. NTILE( ):

Esta función divide el conjunto de resultados en grupos casi iguales.
Los datos se ordenan según la cláusula 'ORDER BY' dentro de la ventana 'OVER( )', y cada fila se asigna a un grupo.

```
SELECT id, nombre, salario, NTILE(3)
OVER(ORDER BY salario)
AS grupo
FROM empleados;
```

Ejemplo:

Imagina que tenemos una tabla empleados con los siguientes datos:

ID	NOMBRE	SALARIO
1	ANA	3000
2	LUIS	2500
3	MARÍA	4000
4	JUAN	5000
5	SOFÍA	3500
6	PEDRO	2800

Si ejecutamos:

```
SELECT id, nombre, salario, NTILE(3)
OVER(ORDER BY salario)
AS grupo
FROM empleados;
```

Los empleados se ordenan por salario, y se dividen en 3 grupos aproximadamente iguales:

ID	NOMBRE	SALARIO	GRUPO
2	LUIS	2500	1
6	PEDRO	2800	1
1	ANA	3000	2
5	SOFÍA	3500	2
3	MARÍA	4000	3
4	JUAN	5000	3

- ☐ *Grupo 1: Salarios más bajos*
- ☐ *Grupo 2: Salarios intermedios*
- ☐ *Grupo 3: Salarios más altos*

El resultado dependerá del número total de filas y de cuántos grupos (NTILE(n)) especificamos.

JOSÉ LUIS CARRILLO GONZÁLEZ

1. CEIL( ):

*La función **'CEIL( )'** devuelve el valor entero más pequeño que sea mayor o igual a un número.*

```
SELECT CEIL(5.4), CEIL(5.5), CEIL(5.6);
```

2. MOD( ):

*La función '**MOD( )'** devuelve el resto de un número dividido por otro número. Se puede usar para mostrar los pares e impares por ejemplo.*

```
SELECT MOD (5,2);
```

3. EXTRACT( ):

*La función '**EXTRACT( )'** extrae una parte de una fecha determinada.*

```
SELECT EXTRACT (day from date &amp;#39;2017-1-1&amp;#39;),
        EXTRACT (month from date &amp;#39;2017-1-1&amp;#39;),
                EXTRACT (year from date &amp;#39;2017-1-1&amp;#39;
);
```

4. SIGN( ):

*La función '**SIGN( )'** devuelve el signo de un número.*

```
SELECT SIGN (-5), SIGN (0) ,SIGN (5);
```

Esta función devolverá uno de los siguientes:

- ☐ *Si el número > 0, devuelve 1*
- ☐ *Si el número = 0, devuelve 0*
- ☐ *Si el número < 0, devuelve -1*

5. IIF( ):

*La función **'IIF( )'** devuelve un valor si una condición es VERDADERA, u otro valor si una condición es FALSA.*

```
SELECT OrderID, Quantity IIF (Quantity &amp;gt; 10, 'More', 'Less')
FROM OrderDetails;
```

6. CHARINDEX( ):

*La función '**CHARINDEX( )'** busca una subcadena en una cadena y devuelve la posición. Si no se encuentra la subcadena, esta función devuelve 0.*

```
SELECT CHARINDEX ('OM' , 'Customer')
 AS MatchPosition;
```

***Nota:** esta función realiza una búsqueda sin distinguir entre mayúsculas y minúsculas.*

7. RPAD( ):

*La función '**RPAD( )'** rellena una cadena con otra cadena hasta una longitud determinada.*

```
SELECT RPAD(&amp;quot;Pepe&amp;quot;, 20, &amp;quot;ABC&amp;quot;);
```

***Nota**: observe también la función '**LPAD( )'**.*

8. DATE_TRUNC( ):

*La función '**DATE_TRUNC( )'** trunca un valor de fecha, hora o indicación de fecha y hora en la unidad de tiempo especificada. Si tienes una fecha, pero quieres tener sólo el primer día del mes de esa fecha, truncamos el valor de la fecha a un parámetro '**"mes"**'.*

```
SELECT DATE_TRUNC ('MONTH', date '2017-04-02');
```

9. NULLIF( ):

La función NULLIF() compara dos expresiones y devuelve NULL si son iguales. De lo contrario, se devuelve la primera expresión.

```
SELECT NULLIF(25, 25);
```

10. LAST_INSERT_ID( ):

***'LAST_INSERT_ID( )'** (sin argumentos) devuelve el primer valor generado automáticamente que se insertó correctamente en una columna '**AUTO_INCREMENT'** como resultado de la última instrucción '**INSERT'** ejecutada. El valor de '**LAST_INSERT_ID( )'** permanece sin cambios si no se insertan filas correctamente.*

*Si se le da un argumento a '**LAST_INSERT_ID( )**', se devolverá el valor de la expresión y la siguiente llamada a '**LAST_INSERT_ID( )'** devolverá el mismo valor.*

```
CREATE TABLE t (
id INTEGER UNSIGNED AUTO_INCREMENT PRIMARY KEY,
f VARCHAR(1))
ENGINE = InnoDB;
INSERT INTO t(f) VALUES(&amp;#39;a&amp;#39;);
SELECT LAST_INSERT_ID( );
1
INSERT INTO t(f) VALUES(&amp;#39;b&amp;#39;);
INSERT INTO t(f) VALUES(&amp;#39;c&amp;#39;);
SELECT LAST_INSERT_ID( );
3
```