Actividad de Evaluación: Funciones SQL

Nombre: Gustavo Martinez Guerra **Matrícula:** 21301268 **Materia:** Inteligencia de negocios **Fecha:** 9 de octubre de 2025

Objetivo

El alumno elaborará un documento que contenga teoría, sintaxis, ejemplos y resultados de ejecución sobre las siguientes funciones SQL:

- Funciones de cadenas
- Funciones de fechas
- Control de valores nulos
- Uso de MERGE
- Uso de CASE

1. Funciones de Cadenas

Teoría

Las **funciones de cadenas** permiten manipular y transformar valores de texto. Se utilizan para concatenar, extraer, reemplazar o convertir texto dentro de una consulta SQL.

Algunas funciones comunes son:

- `LEN()` → devuelve la longitud de una cadena.
- `UPPER()` y `LOWER()` → convierten texto a mayúsculas o minúsculas.
- `SUBSTRING()` → extrae una parte específica del texto.
- `REPLACE()` → sustituye un fragmento de texto por otro.

Sintaxis

- Longitud de una cadena

```
SELECT LEN('SQL Server');
```

(No column name)
10

- Convertir a mayúsculas

```
SELECT UPPER('cadena de texto');
```

(No column name)	(No column name)	(No column name)
2025	10	9

- Extraer parte del texto

```
SELECT SUBSTRING('Programación', 1, 7);
```

(No column name)
Program

- Reemplazar texto

```
SELECT REPLACE('Hola Mundo', 'Mundo', 'SQL');
```

(No column name)
Hola SQL

Ejemplo practico

```
SELECT
    FirstName,
    UPPER(LastName) AS ApellidoMayuscula,
    LEN(FirstName) AS LongitudNombre
FROM Employees;
```

	FirstName	ApellidoMayuscula	LongitudNombre
1	Nancy	DAVOLIO	5
2	Andrew	FULLER	6
3	Janet	LEVERLING	5
4	Margaret	PEACOCK	8
5	Steven	BUCHANAN	6
6	Michael	SUYAMA	7
7	Robert	KING	6
8	Laura	CALLAHAN	5
9	Anne	DODSWORTH	4
10	Bruno	QUINTANA	5

2. Funciones de Fechas

Teoría

Las funciones de fecha permiten manipular valores de tipo datetime, obteniendo partes específicas (día, mes, año) o realizando cálculos con fechas.

Funciones más comunes:

GETDATE() → devuelve la fecha y hora actual del sistema.

YEAR(), MONTH(), DAY() → extraen partes de la fecha.

DATEADD() → suma o resta intervalos a una fecha.

DATEDIFF() → calcula la diferencia entre dos fechas.

💻 Sintaxis

- Fecha y hora actual

```
SELECT GETDATE();
```

	(No column name)
1	2025-10-09 17:40:02.883

- Extraer año, mes y día

```
SELECT YEAR(GETDATE()), MONTH(GETDATE()), DAY(GETDATE());
```

	(No column name)	(No column name)	(No column name)
1	2025	10	9

- Sumar días a una fecha

```
SELECT DATEADD(DAY, 10, '2025-10-09');
```

(No column name)
2025-10-19 00:00:00.000

- Diferencia en días entre dos fechas

```
SELECT DATEDIFF(DAY, '2025-01-01', GETDATE());
```

(No column name)
281

🩹 Ejemplo practico

```
SELECT
    OrderID,
```

```
    OrderDate,
    DATEDIFF(DAY, OrderDate, GETDATE()) AS DiasTranscurridos
FROM Orders;
```

	OrderID	OrderDate	DiasTranscurridos
1	10248	1996-07-04 00:00:00.000	10689
2	10249	1996-07-05 00:00:00.000	10688
3	10250	1996-07-08 00:00:00.000	10685
4	10251	1996-07-08 00:00:00.000	10685
5	10252	1996-07-09 00:00:00.000	10684
6	10253	1996-07-10 00:00:00.000	10683
7	10254	1996-07-11 00:00:00.000	10682
8	10255	1996-07-12 00:00:00.000	10681
9	10256	1996-07-15 00:00:00.000	10678
10	10257	1996-07-16 00:00:00.000	10677
11	10258	1996-07-17 00:00:00.000	10676
12	10259	1996-07-18 00:00:00.000	10675
13	10260	1996-07-19 00:00:00.000	10674
14	10261	1996-07-19 00:00:00.000	10674
15	10262	1996-07-22 00:00:00.000	10671
16	10263	1996-07-23 00:00:00.000	10670
17	10264	1996-07-24 00:00:00.000	10669
18	10265	1996-07-25 00:00:00.000	10668
19	10266	1996-07-26 00:00:00.000	10667
20	10267	1996-07-29 00:00:00.000	10664

🚫 3. Control de Valores Nulos

📖 Teoría

Los valores NULL representan datos desconocidos o inexistentes. SQL Server ofrece funciones para tratarlos y evitar errores en operaciones o resultados.

Funciones principales:

ISNULL(valor, reemplazo) → reemplaza un valor nulo por otro.

COALESCE(valor1, valor2, ...) → devuelve el primer valor no nulo.

💻 Sintaxis

- Reemplazar NULL por un valor

```
SELECT
    CategoryID,
    CategoryName,
    ISNULL(Description, 'Sin descripción') AS Descripcion
FROM Categories;
```

```
INSERT INTO Categories (CategoryName, Description)
VALUES ('Postres', NULL);
```

	CategoryID	CategoryName	Descripcion
1	1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	2	Condiments	Sweet and savory sauces, relishes, spreads, and ...
3	3	Confections	Desserts, candies, and sweet breads
4	4	Dairy Products	Cheeses
5	5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	6	Meat/Poultry	Prepared meats
7	7	Produce	Dried fruit and bean curd
8	8	Seafood	Seaweed and fish
9	9	Fast Food	Comida chatarra
10	10	Postres	Sin descripción

- Devolver el primer valor no nulo

```sql
SELECT COALESCE(NULL, NULL, 'Valor por defecto');
```

(No column name)
Valor por defecto

Ejemplo Práctico

```sql
SELECT
    CustomerID,
    ISNULL(Region, 'No especificada') AS Region
FROM Customers;
```

	CustomerID	Region
1	ABCD3	No especificada
2	ABCD4	No especificada
3	ABCD8	No especificada
4	ABCD9	No especificada
5	ALFKI	No especificada
6	ANATR	No especificada
7	ANTON	No especificada
8	AROUT	No especificada
9	BERGS	No especificada
10	BLAUS	No especificada
11	BLONP	No especificada
12	BOLID	No especificada
13	BONAP	No especificada
14	BSBEV	No especificada
15	CACTU	No especificada
16	CENTC	No especificada
17	CHOPS	No especificada
18	CONSH	No especificada
19	DRACD	No especificada
20	DUMON	No especificada

4. Uso de MERGE

Teoría

La instrucción MERGE permite realizar operaciones INSERT, UPDATE o DELETE en una sola sentencia, dependiendo de si los registros coinciden o no entre dos tablas.

Es útil para sincronizar datos entre tablas.

Sintaxis

```
MERGE TablaDestino AS D
USING TablaOrigen AS O
ON D.ID = O.ID
WHEN MATCHED THEN
    UPDATE SET D.Campo = O.Campo
WHEN NOT MATCHED THEN
    INSERT (ID, Campo) VALUES (O.ID, O.Campo)
WHEN NOT MATCHED BY SOURCE THEN
    DELETE;
```

Ejemplo Práctico

- Actualizar/insertar algunos productos por NOMBRE

```
MERGE dbo.Products AS D
USING (VALUES
  ('Chai', 22.00, 0),      -- ProductName, UnitPrice, Discontinued
  ('Tofu', 18.50, 0)
) AS O(ProductName, UnitPrice, Discontinued)
ON D.ProductName = O.ProductName
WHEN MATCHED THEN
  UPDATE SET D.UnitPrice = O.UnitPrice,
             D.Discontinued = O.Discontinued
WHEN NOT MATCHED BY TARGET THEN
  INSERT (ProductName, SupplierID, CategoryID, QuantityPerUnit, UnitPrice,
          UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued)
  VALUES (O.ProductName, 1, 1, '1 box', O.UnitPrice, 0, 0, 0, O.Discontinued);
```

```
(2 rows affected)

Completion time: 2025-10-09T18:09:02.2367670-06:00
```

```
  SELECT ProductID, ProductName, UnitPrice, Discontinued
FROM Products
WHERE ProductName IN ('Chai', 'Tofu');
```

ProductID	ProductName	UnitPrice	Discontinued
1	Chai	22.00	0
14	Tofu	18.50	0

⚙️ 5. Uso de CASE

🕮 Teoría

La instrucción CASE permite realizar evaluaciones condicionales dentro de una consulta SQL. Es similar a una estructura if...else en otros lenguajes.

Se usa comúnmente para clasificar datos o mostrar resultados personalizados.

💻 Sintaxis

```
SELECT
    columna,
    CASE
        WHEN condición1 THEN resultado1
        WHEN condición2 THEN resultado2
        ELSE resultado_por_defecto
    END AS NuevaColumna
FROM tabla;
pendiente 2
```

🧹 Ejemplo Práctico

- Clasifica cada producto como Barato / Medio / Caro.

```
SELECT
    ProductID,
    ProductName,
    UnitPrice,
    CASE
        WHEN UnitPrice < 20 THEN 'Barato'
        WHEN UnitPrice BETWEEN 20 AND 50 THEN 'Medio'
        ELSE 'Caro'
    END AS RangoPrecio
FROM Products;
```

	ProductID	ProductName	UnitPrice	RangoPrecio
1	1	Chai	22.00	Medio
2	2	Chang	19.00	Barato
3	3	Aniseed Syrup	10.00	Barato
4	4	Chef Anton's Cajun Seasoning	22.00	Medio
5	5	Chef Anton's Gumbo Mix	21.35	Medio
6	6	Grandma's Boysenberry Spread	25.00	Medio
7	7	Uncle Bob's Organic Dried Pears	30.00	Medio
8	8	Northwoods Cranberry Sauce	40.00	Medio
9	9	Mishi Kobe Niku	97.00	Caro
10	10	Ikura	31.00	Medio
11	11	Queso Cabrales	21.00	Medio
12	12	Queso Manchego La Pastora	38.00	Medio
13	13	Konbu	6.00	Barato
14	14	Tofu	18.50	Barato
15	15	Genen Shouyu	15.50	Barato
16	16	Pavlova	17.45	Barato
17	17	Alice Mutton	39.00	Medio
18	18	Carnarvon Tigers	62.50	Caro
19	19	Teatime Chocolate Biscuits	9.20	Barato
20	20	Sir Rodney's Marmalade	81.00	Caro

☑ Conclusión

Las funciones SQL permiten manipular y analizar datos de forma eficiente. Cada tipo (cadenas, fechas, control de nulos, MERGE y CASE) amplía la capacidad de SQL Server para resolver problemas reales en el manejo de información empresarial.

🖩 Elaborado por:

Gustavo Martinez Guerra - UTTT – Ingeniería en Entornos Virtuales y Negocios Digitales