Funciones para manipulación de números

Es habitual en una base de datos que almacenemos números de cierta manera pero luego necesitemos realizar operaciones con los mismos. Transformándolos o haciendo operaciones matemáticas con ellos.
Este apunte tiene como objetivo ejemplificar estos casos y mostrar los operadores y funciones integradas al lenguaje más habituales.

Base de Datos

Para contar con un contexto que sea entendible y uniforme para todos los ejemplos, en este apunte utilizaremos la siguiente base de datos. La misma cuenta con el siguiente script para su creación en SQL Server:

```
Create Database EjemploFuncionesNumericas
Go
Use EjemploFuncionesNumericas
Go
Create Table Cuentas(
  ID_Cuenta bigint not null primary key identity (1, 1),
  DNI varchar(10) not null,
  Saldo decimal(18,2),
  Limite_Descubierto decimal(10,2),
  Interes_Mensual decimal(5,2),
  Cuota_Mensual decimal(10,2)
)
```

ℹ️ Como se mencionó anteriormente, esto es una simplificación. El saldo de una cuenta debe estar respaldado por transacciones (movimientos de crédito y débito) que deberían representarse en otra tabla. Los valores de interés mensual y cuota mensual deberían tener su histórico en respectivas tablas que aquí se representan de manera sencilla en columnas de la tabla cuentas.

Aclaraciones del contexto de la tabla

La tabla Cuentas representa, de una manera muy simplificada, cuentas bancarias incluyendo datos del cliente, el saldo y elementos adicionales como el límite de descubierto, interés mensual y costos de mantenimiento:

Nombre del campo	Descripción
ID_Cuenta	La clave primaria y código único de la cuenta bancaria
DNI	El dueño de la cuenta bancaria. Podría ser ID_Cliente si existiera una tabla de Clientes
Saldo	El saldo actual de la cuenta bancaria.
Limite_Descubierto	El dinero que puede llegar a adeudar una cuenta bancaria. Podría ser una restricción CHECK si queremos que el saldo jamás pueda estar por debajo del Limite_Descubierto.
Interes_Mensual	El porcentaje de interés que otorga el banco por depositar dinero en la cuenta.
Cuota_Mensual	El costo mensual de la cuota de mantenimiento de la cuenta bancaria.

Operaciones matemáticas básicas

Todas las columnas numéricas tienen la capacidad de ser operandos en cálculos utilizando operadores matemáticos. Los operadores matemáticos más comunes son:

Operador	Descripción	Ejemplo
+	Suma	`Saldo + 5000`
-	Resta	`Saldo - Cuota_Mensual`
*	Producto	`Saldo * 0.15`
/	División	`Saldo / 2`
%	Resto de división	`Saldo % 2`

Funciones Integradas de SQL para manejo de números

En SQL, existen funciones específicas para manipular valores numéricos. Estas funciones son útiles para realizar cálculos, redondeos o evaluar características de valores. A continuación, se presenta una breve descripción y un ejemplo en contexto con la tabla Cuentas para cada función.

ABS()

Propósito:

Devuelve el valor absoluto de un número (elimina el signo).

Ejemplo en la Tabla Cuentas:

```
SELECT id_cuenta, saldo, ABS(saldo) AS saldo_absoluto
FROM Cuentas;
```

Explicación en Contexto:

Esta consulta devuelve el valor absoluto del saldo de cada cuenta, útil para calcular totales sin importar si el saldo es positivo o negativo.

Resultado Genérico:

Si saldo es -200.50, entonces ABS(saldo) devolverá 200.50.

SIGN()

Propósito:

Devuelve el signo de un número:

- 1 si el número es positivo.
- 0 si es cero.
- -1 si es negativo.

Ejemplo en la Tabla Cuentas:

```
SELECT id_cuenta, saldo, SIGN(saldo) AS signo_saldo
FROM Cuentas;
```

Explicación en Contexto:

Devuelve el signo del saldo, permitiendo identificar rápidamente cuentas con saldo deudor (negativo).

Resultado Genérico:

Si saldo es 300.00, el resultado será 1; si es -50.00, el resultado será -1.

CEILING()

Propósito:

Redondea un número hacia arriba al entero más cercano.

Ejemplo en la Tabla Cuentas:

```
SELECT id_cuenta, interes_mensual, CEILING(interes_mensual) AS
interes_redondeado_arriba
FROM Cuentas;
```

Explicación en Contexto:

Devuelve el interés mensual redondeado hacia arriba, útil para sumar cálculos más conservadores en reportes o análisis.

Resultado Genérico:

Si interes_mensual es 3.25, CEILING(interes_mensual) devuelve 4.

FLOOR()

Propósito:

Redondea un número hacia abajo al entero más cercano.

Ejemplo en la Tabla Cuentas:

```
SELECT id_cuenta, interes_mensual, FLOOR(interes_mensual) AS
interes_redondeado_abajo
```

```
FROM Cuentas;
```

Explicación en Contexto:

Devuelve el interés mensual redondeado hacia abajo, ideal para cálculos más conservadores al otorgar beneficios.

Resultado Genérico:

Si interes_mensual es 3.75, FLOOR(interes_mensual) devuelve 3.

ROUND()

Propósito:

Redondea un número según el número de decimales especificados.

Ejemplo en la Tabla Cuentas:

```
SELECT id_cuenta, saldo, ROUND(saldo, 1) AS saldo_redondeado
FROM Cuentas;
```

Explicación en Contexto:

Redondea el saldo al primer decimal, útil para representaciones más simples en reportes.

Resultado Genérico:

Si saldo es 123.456, ROUND(saldo, 1) devuelve 123.5.

CAST()

Propósito:

Convierte un valor de un tipo de datos a otro. No es exclusivo para valores numéricos. Sirve para transformar un tipo de dato a otro siempre que la conversión tenga sentido.

Ejemplo en la tabla Cuentas:

```
Select ID_Cuenta, Cast(Saldo as Integer) As SaldoSinCentavos From
Cuentas;
```

Explicación en Contexto:

En este caso, CAST(saldo AS INT) transforma el valor de saldo (que es de tipo DECIMAL(18,2)) a un número entero (INT), truncando los decimales. Esto podría ser útil para reportes donde no se necesiten los valores decimales del saldo, o en cálculos simplificados.

Resultado Genérico:

Si el valor de saldo es 1234.56, el resultado será 1234.
Si el valor de saldo es -567.89, el resultado será -567.

Esto sería ideal, por ejemplo, para preparar un informe en el que solo interesen los valores enteros del saldo, sin los centavos.

Anexo: Datos para la tabla de Cuentas

```
Use EjemploFuncionesNumericas;

INSERT INTO Cuentas (dni, saldo, limite_descubierto, interes_mensual,
cuota_mensual)
VALUES
      ('123456789', 1500.75, 500.00, 1.25, 10.00),
      ('987654321', -250.50, 1000.00, 2.00, 15.00),
      ('562348791', 0.00, 200.00, 1.50, 12.50),
      ('741258963', 3000.00, 750.00, 1.75, 20.00),
      ('951357468', -100.25, 600.00, 1.00, 8.00),
      ('354789621', 0.00, 300.00, 2.50, 10.50),
      ('246813579', -500.00, 800.00, 1.25, 15.75),
      ('159487263', 200.00, 150.00, 1.00, 5.00),
      ('753961842', 5000.50, 1000.00, 2.25, 25.00),
      ('369258147', -300.00, 1200.00, 1.75, 18.50);
```