Curso SQL Server 😎

Para conocer los tipos de datos y estructura de las tablas, DESCRIBE de MySQL, En SQL Server ejecutar **EXEC sp_help tablename**
Anotación sobre creación de tablas: al anteponer el signo # en el nombre de la tabla, indico que es una TEMPORARY TABLE. Con solo 1 signo # (CREATE TABLE #Empleado) indico que es una tabla temporal **local,** al anteponer 2 signos # (CREATE TABLE ##Sucursal) estoy creando la tabla Sucursal que es temporal pero **global.** Todas estas se guardan dentro de las **tempdb** y se eliminan al cerrar la sesión/conexión a la BD.

Funciones de CONVERSIÓN

- Si tengo un campo 'inscription_date' de tipo DATETIME y lo quiero castear a DATE > **CAST**(inscription_date **AS** DATE) o castear el tipo MONEY A INTEGER > CAST(total_cost AS INT)
- Para castear un VARCHAR a DATE, se usa la función **PARSE()** . Si el campo modified_date viene como caracteres, por ejemplo 'Friday, 18 December 2015' > PARSE(modified_date **AS** DATE) me daria como output '2015-12-18'
- **CASE** Simil ¨IF¨ de python, dispone de valores según condiciones:

```
CASE Columna
WHEN valor1 THEN resultado1
WHEN valor2 THEN resultado2
ELSE resultadox
END AS nombre_columna
```

  Con comparaciones más elaboradas que no sean simplemente la igualdad de una única columna con algún valor:

```
CASE
WHEN columna BETWEEN valor1 AND valor2 THEN resultado
END AS nombre_col
```

Funciones de FECHA

- SELECT **GetDate()** > devuelve fecha actual en formato DATETIME (YYYY-MM-DD hh:mm:ss) , es el NOW() de MySQL
- Se puede extraer el año, mes o dia > MONTH(fecha_venta). Si fecha_venta = '2017-03-17 00:00:00', **mi output = 03**
- **DATEADD()** agrega o quita dias, meses o años a formatios de fecha: DATEADD(MONTH, -1, '2007-12-28') le RESTA un mes a esa fecha > '2007-11-28' DATEADD(DAY, 2, '2018-02-20') > '2018-02-22'.
- Para **convertir** a CARACTERES los formatos DATE usamos **DATENAME()**

DATENAME(MONTH, ‘2009-07-25’) > ‘July’

- **DATEPART()** es idéntico a DATENAME() pero *solo* retorna valores tipo INT
- **DATEDIFF()** Diferencia entre fechas, sobre todo para calcular la edad actual teniendo la fecha de nacimiento:

```
SELECT
    e.Name, AS Nombre
    DATEDIFF(YEAR,e.BirthDate ,GETDATE()) as EdadEmpleado
FROM HumanResources.Employee AS e;
```

Funciones con CARACTERES

- **CONCAT()** para concatenar caracteres: CONCAT(first_name, ‘ ‘, middle_name, ‘ ‘, last_name) AS NombreCompleto
- Mucho más práctico resulta el **CONCAT_WS()** ‘concat whit separator’, te deja concatenar sin tener que repetir los separadores, asignandolo en el primer parámetro: CONCAT_WS(‘ ‘, first_name, middle_name, last_name)
- Para el ‘largo’ o cantidad de char que tenga una cadena usamos **LEN()**, **igual que python.**
- **LEFT()** y **RIGHT()** selecciona una determinada cantidad de caracteres de la derecha(final) o izquierda(comienzo). Con el ejemplo, vamos a ver cuantos dominios DISTINTOS de email de clientes tenemos:

```
SELECT DISTINCT
    RIGHT(e.EmailAdress, 10) AS Dominio
FROM Client.Email AS e ;
```

- **LTRIM()** y **RTRIM()** quita caracteres en blanco de la derecha (RTRIM) o de la izq.
- **REPLACE()** cambia caracteres seleccionados por OTRO caracter/es. Ej: REPLACE(EmailAdress, ‘@yahoo.com.ar’, ‘@gmail.com’). **Recibe 3 argumentos**: el campo donde se ejecuta, el valor a reemplazar, y el/los nuevos caracteres
- **STUFF()** elimina una cierta cantidad de caracteres y en esa posición, inserta números o nuevos caracteres. Recibe 4 argumentos: el campo donde se ejecuta, el numero de caracter donde inicia a eliminar, la cantidad de caracteres a eliminar desde el caracter asignado en el 2do argumento, y el nuevo numero/s o caracter/es. Ej: STUFF(‘Argentina’, 3, 6, ‘PERRO’) > ‘ArPERROa’ y STUFF('Argentina',3,1,'PERRO') > ‘ArPERROentina’
- **LIKE()** encuentra coincidencias en cadenas de chr. Ej:
  WHERE name LIKE ‘Fernand[a_o]’ > Fernanda / Fernando
  WHERE country LIKE ‘a%’ > Argentina/Australia/Angola
  WHERE country LIKE ‘_o%’ > P**o**lonia / C**o**lombia
- **ROW_NUMBER() y PARTITION BY** detecta valores repetidos y los enumera:

```
SELECT
  empleado.Nombre,
  ROW_NUMBER() OVER (PARTITION BY empleado.Nombre) AS Contador
  FROM empleado ;
```

Nombre	Contador
Felipe	1
Felipe	2
Felipe	3
Virola	1
Virola	2

INSERCIÓN DE DATOS

INSERT INTO: INSERT INTO nombre_tabla VALUES (valor1,valor2 . . . . .)
Insertar el output de una query en una tabla nueva (tabla que no exista):

```
SELECT *
INTO tabla_solo_mujeres
FROM cliente
WHERE género = 'F'
```

Si la tabla ya existe y quiero insertar una consulta:

```
INSERT INTO tabla_solo_mujeres
SELECT *
FROM cliente
WHERE Pais = 'Argentina'
```

MERGE: Se usa al querer llevar data de una tabla a otra, evitando la duplicación de registros al usar un campo de comparacion donde se hace 'match'. Ej:
La tabla 2 está vacía, lo que hice por practicidad fue primero hacer un SELECT * INTO Tabla 2 FROM Tabla 1 y después la trunqué para que me quede la estructura con sus datatypes.

```
MERGE Tabla2 AS a
USING Tabla1 AS b
ON a.id = b.id
WHEN NOT MATCHED THEN
    INSERT VALUES (b.valor1, a.valor2, . . . ) ;
```

BULK INSERT Carga/importación de datos desde archivos planos (csv, txt, etc)

Tener cuidado sobre todo con los formatos de fecha. Por ejemplo, si en el archivo a importar la fecha esta seteada como dia/mes/anio, antes debería setear el formato:

```
SET DATEFORMAT day
```

Sintaxis:

```
BULK INSERT tabla_destino
FROM 'C:\Felipe\SQLServer\Compras.txt'
WITH (FIRSTROW = 2)                    para ignorar encabezados
```

Otros posibles parámetros de WITH :

```
            (
    [ [ , ] BATCHSIZE = batch_size ]
    [ [ , ] CHECK_CONSTRAINTS ]
    [ [ , ] CODEPAGE = { 'ACP' | 'OEM' | 'RAW' | 'code_page' } ]
    [ [ , ] DATAFILETYPE =
       { 'char' | 'native' | 'widechar' | 'widenative' } ]
    [ [ , ] DATA_SOURCE = 'data_source_name' ]
    [ [ , ] ERRORFILE = 'file_name' ]
    [ [ , ] ERRORFILE_DATA_SOURCE = 'errorfile_data_source_name' ]
    [ [ , ] FIRSTROW = first_row ]
    [ [ , ] FIRE_TRIGGERS ]
    [ [ , ] FORMATFILE_DATA_SOURCE = 'data_source_name' ]
    [ [ , ] KEEPIDENTITY ]
    [ [ , ] KEEPNULLS ]
    [ [ , ] KILOBYTES_PER_BATCH = kilobytes_per_batch ]
    [ [ , ] LASTROW = last_row ]
    [ [ , ] MAXERRORS = max_errors ]
    [ [ , ] ORDER ( { column [ ASC | DESC ] } [ ,...n ] ) ]
    [ [ , ] ROWS_PER_BATCH = rows_per_batch ]
    [ [ , ] ROWTERMINATOR = 'row_terminator' ]
    [ [ , ] TABLOCK ]

   -- input file format options
    [ [ , ] FORMAT = 'CSV' ]
    [ [ , ] FIELDQUOTE = 'quote_characters']
    [ [ , ] FORMATFILE = 'format_file_path' ]
    [ [ , ] FIELDTERMINATOR = 'field_terminator' ]
    [ [ , ] ROWTERMINATOR = 'row_terminator' ]
     )
```