

Procedimientos almacenados

Variables

Declarar variables y asignar valores

```
DECLARE @NOMBRE VARCHAR(30)
DECLARE @EDAD INT
SET @NOMBRE = 'ANGEL'
SET @EDAD = 27
```

Al colocar PRINT ejecutamos e imprime por pantalla el nombre:

```
DECLARE @NOMBRE VARCHAR(30)
DECLARE @EDAD INT
SET @NOMBRE = 'ANGEL'
SET @EDAD = 27
PRINT @NOMBRE
```

Mensajes

ANGEL

```
DECLARE @NOMBRE VARCHAR(30)
DECLARE @EDAD INT
SET @NOMBRE = 'ANGEL'
SET @EDAD = 27
SELECT @NOMBRE AS NOMBRE, @EDAD AS EDAD
```

Resultados

	NOMBRE	EDAD
1	ANGEL	27

DECLARE → asigna un valor solamente.

Al ejecutar esta consulta se ejecuta correctamente pero no te muestra los valores:

```
DECLARE @APELLIDO VARCHAR(50)
DECLARE @NOMBRE VARCHAR(50)
DECLARE @DNI VARCHAR(10)

SELECT @APELLIDO = APELLIDO, @NOMBRE = NOMBRE, @DNI = DNI
FROM PERSONAS
WHERE SEXO = 'F'
```

Mensajes

Comandos completados correctamente.

El 1er SELECT declara las variables con la tabla PERSONAS y el 2do SELECT muestra los resultados por pantalla:

- Quiere decir que a partir de una o varias tablas podemos asignar valores a nuestras variables.

```
DECLARE @APELLIDO VARCHAR(50)
DECLARE @NOMBRE VARCHAR(50)
DECLARE @DNI VARCHAR(10)

SELECT @APELLIDO = APELLIDO, @NOMBRE = NOMBRE, @DNI = DNI
FROM PERSONAS
WHERE SEXO = 'F'

SELECT @APELLIDO + ', ' + @NOMBRE AS APENOM, @DNI AS ID
```

Resultados

APENOM	ID
BENES, ELAINE	3333

DECISIÓN SIMPLE


```
DECLARE @ORDEN BIT
SET @ORDEN = 1
IF @ORDEN = 1 BEGIN
    SELECT * FROM PERSONAS ORDER BY APELLIDO ASC
END
ELSE BEGIN
    SELECT * FROM PERSONAS ORDER BY APELLIDO DESC
END
```

Resultados

IDPERSONA	APELLIDO	NOMBRE	SEXO	DNI	
1	3	BENES	ELAINE	F	3333
2	2	COSTANZA	GEORGE	M	2222
3	4	KRAMER	COSMO	M	4444
4	1	SEINFELD	JERRY	M	1111

Al declarar orden en 0 podemos ver que muestra en DESC:

```
DECLARE @ORDEN BIT
SET @ORDEN = 0
IF @ORDEN = 1 BEGIN
    SELECT * FROM PERSONAS ORDER BY APELLIDO ASC
END
ELSE BEGIN
    SELECT * FROM PERSONAS ORDER BY APELLIDO DESC
END
```



IDPERSONA	APELLIDO	NOMBRE	SEXO	DNI
1	SEINFELD	JERRY	M	1111
4	KRAMER	COSMO	M	4444
2	COSTANZA	GEORGE	M	2222
3	BENES	ELAINE	F	3333


Funciones Globales

@@ROWCOUNT: Se encarga de devolver la cantidad de filas que fueron afectadas por la consulta anterior

En este ejemplo muestra 4 por que son 4 filas afectadas (registros) de la tabla persona.

```
SQLQuery8.sql - localhost\....gel (52))"
--@@ROWCOUNT, @@ERROR, @@IDENTITY

SELECT * FROM PERSONAS
SELECT @@ROWCOUNT
```




IDPERSONA	APELLIDO	NOMBRE	SEXO	DNI
1	SEINFELD	JERRY	M	1111
2	COSTANZA	GEORGE	M	2222
3	BENES	ELAINE	F	3333
4	KRAMER	COSMO	M	4444

(Sin nombre de columna)
4

Al ejecutar esta consulta muestra que 0 filas fueron afectadas:

```
SQLQuery5.sql - localhost\....gel (52))"
--@@ROWCOUNT, @@ERROR, @@IDENTITY

UPDATE PERSONAS SET APELLIDO = 'PEREZ' WHERE DNI = 5555
SELECT @@ROWCOUNT
```




(Sin nombre de columna)
0

En cambio en esta consulta podemos ver que 1 sola fila fue afectada y de esta forma con ROWCOUNT también vamos a poder ver bien cuantas fueron afectadas:

```
SQLQuery8.sql - localhost\....gel (52)))*
--@@ROWCOUNT, @@ERROR, @@IDENTITY

UPDATE PERSONAS SET SEXO = 'F' WHERE DNI = 3333
SELECT @@ROWCOUNT
```

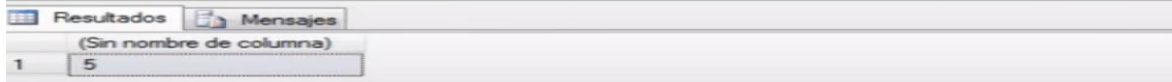


(Sin nombre de columna)	
1	1

@@IDENTITY: Obtiene el ultimo id autogenerado y por ejemplo al hacer el insert que vemos a continuación, nos muestra por pantalla el ID de ese insert creado:


```
--@@ROWCOUNT, @@ERROR, @@IDENTITY

INSERT INTO PERSONAS (APELLIDO, NOMBRE, SEXO, DNI)
VALUES ('SIMON', 'ANGEL', 'M', 5555)
SELECT @@IDENTITY
```



(Sin nombre de columna)	
1	5

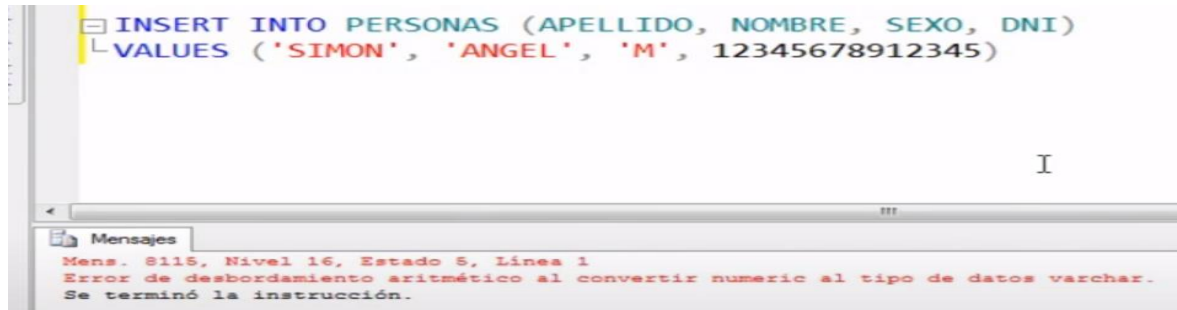
```
SELECT * FROM PERSONAS
```



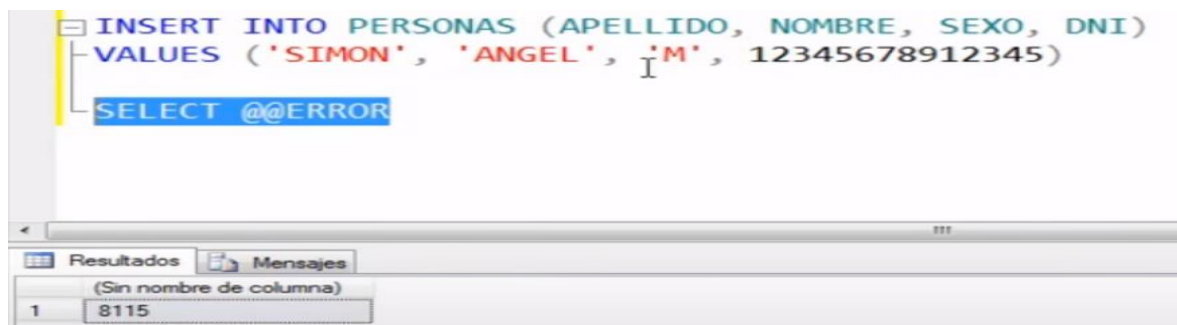
	IDPERSONA	APELLIDO	NOMBRE	SEXO	DNI
1	1	SEINFELD	JERRY	M	1111
2	2	COSTANZA	GEORGE	M	2222
3	3	BENES	ELAINE	F	3333
4	4	KRAMER	COSMO	M	4444
5	5	SIMON	ANGEL	M	5555

@@ERROR: Código de error de la consulta anterior

El DNI en este caso es mas largo de lo normal:

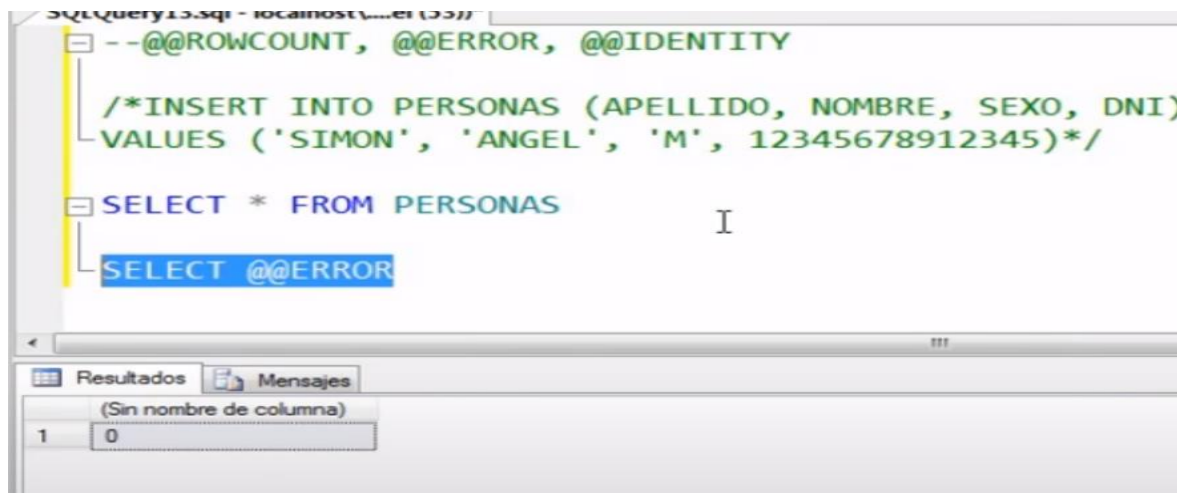


Al ejecutarlo con @@ERROR nos indica cual es el código de error del (error rojo img anterior):



Para que funcione, primero tenemos que ejecutar el insert erroneo y después tenemos que ejecutar SELECT @@ERROR solamente para que nos aparezca el código de error.

En este caso podemos ver que si ejecutamos select * from personas y luego ejecutamos select @@ERROR nos aparece 0 por no haber ningún error causado anteriormente:



TRY y CATCH

- ▼ El bloque Try..Catch sirve para realizar el manejo de errores.
- ▼ Se deben agrupar todas las instrucciones que pueden causar error dentro del bloque TRY.
- ▼ Si alguna de las instrucciones dentro del bloque TRY provocan algún tipo de error, se procede a ejecutar la primera instrucción del bloque CATCH.
- ▼ Si todas las instrucciones del bloque TRY se ejecutan sin errores, el bloque CATCH no se ejecuta.
- ▼ Se deben agrupar todas las instrucciones destinadas a gestionar los errores en el bloque CATCH.

Estructura general

```
[ BEGIN TRY
  -- instrucciones que pueden provocar errores
END TRY

BEGIN CATCH
  -- gestión de eventuales errores
END CATCH
```

La función RAISERROR

- ▼ La función RAISERROR se utiliza para generar un error

```
RAISERROR ({código | mensaje}, {gravedad, estado})
```

Código: Se puede establecer un código existente en la tabla sysmessages.

Mensaje: Alternativamente, se puede definir un mensaje personalizado de hasta 2047 caracteres.

Gravedad: Permite especificar la gravedad del error. Su valor oscila entre 0 y 24. Se recomienda un valor entre 11 y 16. Y siempre menor a 18, ya que de lo contrario se considera un error fatal y se cierra la conexión.

Estado: Permite establecer el origen/contexto del error. Útil para asociarlo con la documentación de la aplicación.

```
SQLQuery2.sql - localhost\.....el (52))*  
- DECLARE @VAL INT  
- SET @VAL = 1/0  
  
- BEGIN TRY  
  
- END TRY  
- BEGIN CATCH  
  
- END CATCH
```

Mensajes

Mens. 8134, Nivel 16, Estado 1, Línea 2
Error de división entre cero.

```
SQLQuery2.sql - localhost\.....el (52))*  
- BEGIN TRY  
-     DECLARE @VAL INT  
-     SET @VAL = 1/0  
- END TRY  
- BEGIN CATCH  
-     PRINT 'ERROR AL DIVIDIR. NO SE PUEDE DIVIDIR POR CERO'  
- END CATCH
```

Mensajes

ERROR AL DIVIDIR. NO SE PUEDE DIVIDIR POR CERO

SQLQuery2.sql - localhost\.....el (52))*

```

BEGIN TRY
    DECLARE @VAL INT
    SET @VAL = 1/0
END TRY
BEGIN CATCH
    RAISERROR ('ERROR AL DIVIDIR. NO SE PUEDE DIVIDIR POR CERO', 16, 10) WITH LOG
END CATCH

```

Mensajes

Mens. 50000, Nivel 16, Estado 10, Línea 6
ERROR AL DIVIDIR. NO SE PUEDE DIVIDIR POR CERO

ltar Ver

nsulta

ster

ry

EG

ND

EG

ND

ajes

SQL

AL

ERRORLOG: Bloc de notas

Archivo	Edición	Formato	Ver	Ayuda
2014-01-02 17:42:21.06	spid53	Starting up database	'PARTE1'.	
2014-01-02 17:42:21.17	spid53	Starting up database	'PARTE2'.	
2014-01-02 17:42:21.30	spid53	Starting up database	'PARTEa1'.	
2014-01-02 17:45:54.90	spid53	Starting up database	'DEMO'.	
2014-01-02 17:45:55.03	spid53	Starting up database	'DEMO2'.	
2014-01-02 17:45:55.17	spid53	Starting up database	'DEMO3'.	
2014-01-02 17:45:55.25	spid53	Starting up database	'PARCIAL2_PARTE2'.	
2014-01-02 17:45:55.44	spid53	Starting up database	'PARTE1'.	
2014-01-02 17:45:55.55	spid53	Starting up database	'PARTE2'.	
2014-01-02 17:45:55.65	spid53	Starting up database	'PARTEa1'.	
2014-01-02 17:47:18.02	spid53	Starting up database	'DEMO'.	
2014-01-02 17:47:18.17	spid53	Starting up database	'DEMO2'.	
2014-01-02 17:47:18.31	spid53	Starting up database	'DEMO3'.	
2014-01-02 17:47:18.40	spid53	Starting up database	'PARCIAL2_PARTE2'.	
2014-01-02 17:47:18.56	spid53	Starting up database	'PARTE1'.	
2014-01-02 17:47:18.68	spid53	Starting up database	'PARTE2'.	
2014-01-02 17:47:18.84	spid53	Starting up database	'PARTEa1'.	
2014-01-02 17:48:43.85	spid53	Starting up database	'DEMO'.	
2014-01-02 17:48:43.97	spid53	Starting up database	'DEMO2'.	
2014-01-02 17:48:44.13	spid53	Starting up database	'DEMO3'.	
2014-01-02 17:48:44.21	spid53	Starting up database	'PARCIAL2_PARTE2'.	
2014-01-02 17:48:44.37	spid53	Starting up database	'PARTE1'.	
2014-01-02 17:48:44.47	spid53	Starting up database	'PARTE2'.	
2014-01-02 17:48:44.63	spid53	Starting up database	'PARTEa1'.	
2014-01-02 17:48:49.61	spid52	Error: 50000, gravedad: 16, estado: 10.		
2014-01-02 17:48:49.61	spid52	ERROR AL DIVIDIR. NO SE PUEDE DIVIDIR POR CERO		

0

SQLQuery2.sql - localhost\.....el (52))*

```

BEGIN TRY
    INSERT INTO PERSONAS (APELLIDO, NOMBRE, SEXO, DNI)
    VALUES ('PEREZ', 'MARTIN', 'M', 3333)
END TRY
BEGIN CATCH
    PRINT 'NO SE PUDO ALMACENAR EL REGISTRO DE PERSONAS'
END CATCH

```

Mensajes

NO SE PUDO ALMACENAR EL REGISTRO DE PERSONAS


```

BEGIN TRY
    INSERT INTO PERSONAS (APELLIDO, NOMBRE, SEXO, DNI)
    VALUES ('PEREZ', 'MARTIN', 'M', 3333)
END TRY
BEGIN CATCH
    PRINT ERROR_MESSAGE()
END CATCH

```

Mensajes

Infracción de la restricción UNIQUE KEY 'UQ_PERSONAS_C035B8DD023D5A04'. No se puede insertar una clave duplicada en el objeto 'dbo.PERSONA'.

Procedimientos Almacenados

```

CREATE PROCEDURE SP_OBTENER_CLIENTES
AS
BEGIN
    SELECT * FROM CLIENTES
END

```

DROP PROCEDURE SP_OBTENER_CLIENTES

EXEC SP_OBTENER_CLIENTES

```

CREATE PROCEDURE SP_AGREGAR_STOCK (
    @IDARTICULO BIGINT,
    @CANTIDAD INT
)
AS
BEGIN
    UPDATE ARTICULOS SET STOCK = STOCK + @CANTIDAD
    WHERE IDARTICULO = @IDARTICULO
END

```

```

SELECT * FROM ARTICULOS WHERE IDARTICULO = 1

EXEC SP_AGREGAR_STOCK 1, 6

```

IDARTICULO	ARTICULO	PRECIO	STOCK	STOCK_MIN	ORIGEN
1	ROUTER	550.00	16	5	I

Al realizar esta query la toma igual afectando la cantidad de stock sacándole 1:

```
EXEC SP_AGREGAR_STOCK 1, -1
```

Para solucionar esto vamos a realizar:

```
ALTER PROCEDURE SP_AGREGAR_STOCK(  
    @IDARTICULO BIGINT,  
    @CANTIDAD INT  
)  
AS  
BEGIN  
    IF @CANTIDAD > 0 BEGIN  
        UPDATE ARTICULOS SET STOCK = STOCK + @CANTIDAD  
        WHERE IDARTICULO = @IDARTICULO  
    END  
END
```

Continuación:

```
BEGIN  
    IF @CANTIDAD > 0 BEGIN  
        UPDATE ARTICULOS SET STOCK = STOCK + @CANTIDAD  
        WHERE IDARTICULO = @IDARTICULO  
    END  
    ELSE BEGIN  
        RAISERROR('LA CANTIDAD NO ES VÁLIDA', 16, 1)  
    END  
END
```

Estado

Se transforma en una excepción

Ahora al ejecutar nos indica el mensaje de error:

```
QLQuery6.sql - ANGEL-YB...)) ANGEL-YBOX\SQL...P - Diagram_0  
EXEC SP_AGREGAR_STOCK 1, -1  
  
Mens 50000, Nivel 16, Estado 1, Procedimiento SP_AGREGAR_STOCK, Línea 12  
LA CANTIDAD NO ES VÁLIDA
```

Al realizar una cantidad valida nos lo ejecuta correctamente:

```
QLQuery6.sql - ANGEL-YB...)) ANGEL-YBOX\SQL...P - Diagram_0  
EXEC SP_AGREGAR_STOCK 1, 2
```

0

```
CREATE PROCEDURE SP_AGREGAR_VENTA(  
    @DNI VARCHAR(10),  
    @IDARTICULO BIGINT,  
    @CANTIDAD INT  
)  
AS  
BEGIN  
    INSERT INTO VENTAS(DNI, IDARTICULO, CANTIDAD, FECHA, IMPORTE)  
    VALUES(@DNI, @IDARTICULO, @CANTIDAD, GETDATE(), 0)  
END
```

Mejoramos el procedure multiplicando el **precio unitario * cantidad**:

```
ALTER PROCEDURE SP_AGREGAR_VENTA(
    @DNI VARCHAR(10),
    @IDARTICULO BIGINT,
    @CANTIDAD INT
)
AS
BEGIN
    DECLARE @PU MONEY
    SELECT @PU = PRECIO FROM ARTICULOS WHERE IDARTICULO = @IDARTICULO

    INSERT INTO VENTAS(DNI, IDARTICULO, CANTIDAD, FECHA, IMPORTE)
    VALUES(@DNI, @IDARTICULO, @CANTIDAD, GETDATE(), @PU*@CANTIDAD)
END
```

* Asterisco

Ahora tambien implementamos para que **descuento el stock del articulo vendido**:

```
ALTER PROCEDURE SP_AGREGAR_VENTA(
    @DNI VARCHAR(10),
    @IDARTICULO BIGINT,
    @CANTIDAD INT
)
AS
BEGIN
    BEGIN TRY
        DECLARE @PU MONEY
        SELECT @PU = PRECIO FROM ARTICULOS WHERE IDARTICULO = @IDARTICULO

        INSERT INTO VENTAS(DNI, IDARTICULO, CANTIDAD, FECHA, IMPORTE)
        VALUES(@DNI, @IDARTICULO, @CANTIDAD, GETDATE(), @PU*@CANTIDAD)
    END TRY
    BEGIN CATCH
        RAISERROR('NO SE PUDO AGREGAR LA VENTA', 16, 1)
    END CATCH
END
```

Y agregamos el UPDATE

```
INSERT INTO VENTAS(DNI, IDARTICULO, CANTIDAD, FECHA, IMPORTE)
VALUES(@DNI, @IDARTICULO, @CANTIDAD, GETDATE(), @PU*@CANTIDAD)

UPDATE ARTICULOS SET STOCK = STOCK - @CANTIDAD WHERE IDARTICULO = @IDARTICULO
END TRY
BEGIN CATCH
    RAISERROR('NO SE PUDO AGREGAR LA VENTA', 16, 1)
END CATCH
```

Con **transacciones** solucionamos ej:

articulo: Rollers cantidad: 6

Queremos hacer la venta pero con cantidad 10 que no tenemos!

Aparece mensaje de error correctamente pero crea una nueva fila con la venta de 10 incorrectamente.