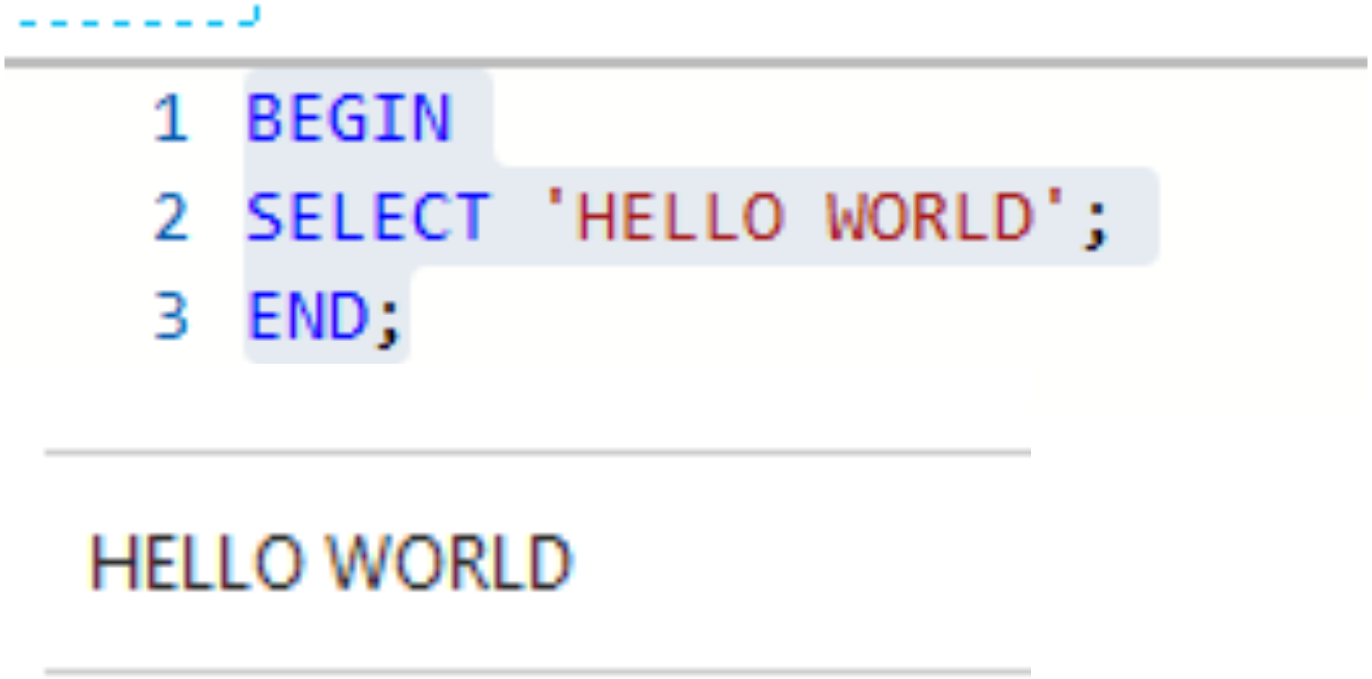
Transact-SQL (T-SQL)

- (Microsoft's) Extension of SQL Language. (Oracle has PLSQL)
- Allows procedural programming of databases.
- Enables the implementation of business logic at Database level.
- Includes:
  - Anonymous Blocks (NOT stored single use)
  - Stored procedures (stored – no return value)
  - Stored Functions (stored – has return value)
  - Triggers (stored – executed based on events)
- Full SSR (Sequence, Selection, Repetition)
- No Object Oriented Capability

Anonymous Blocks

- Starts with BEGIN, finishes with END;

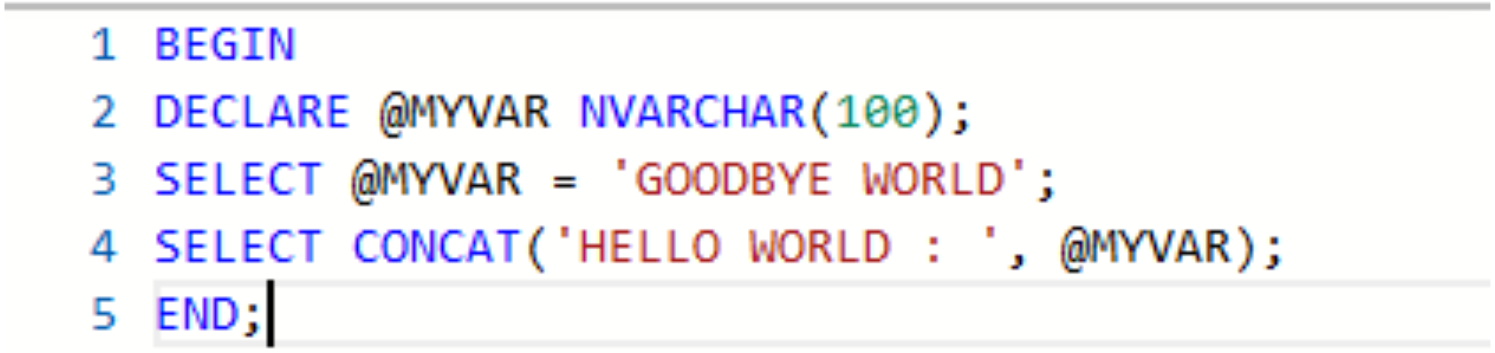


```
BEGIN
SELECT 'HELLO WORLD';
END;
```

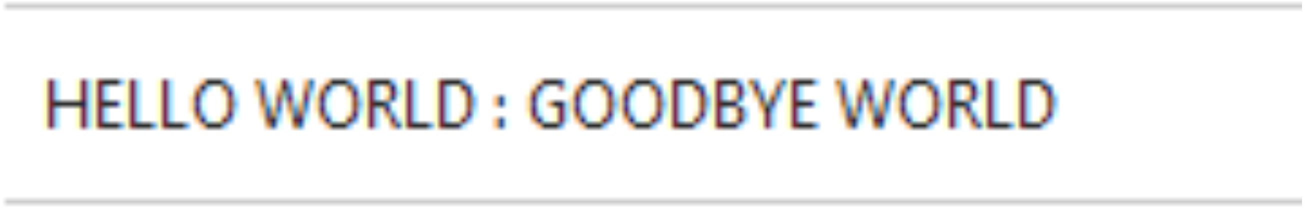
```
HELLO WORLD
```

Variables

- Have data types and store values just like C#



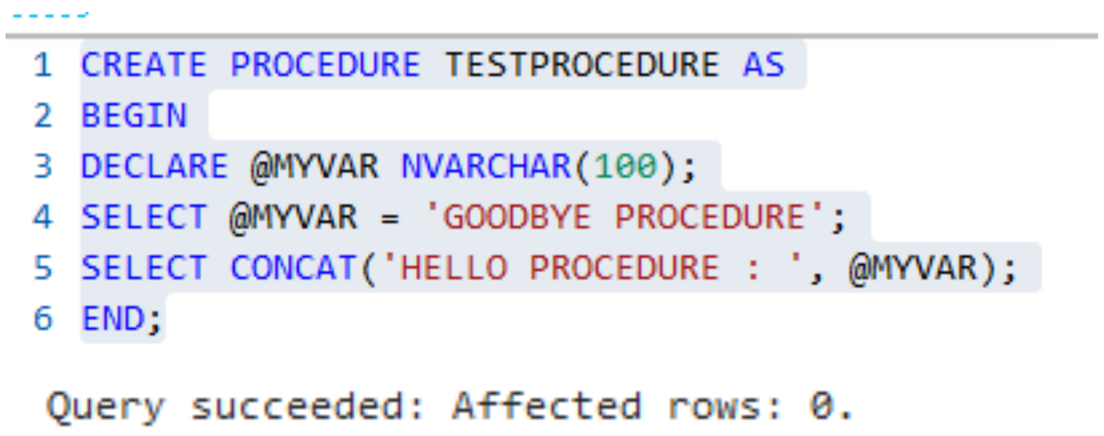
```
BEGIN
DECLARE @MYVAR NVARCHAR(100);
SELECT @MYVAR = 'GOODBYE WORLD';
SELECT CONCAT('HELLO WORLD : ', @MYVAR);
END;
```



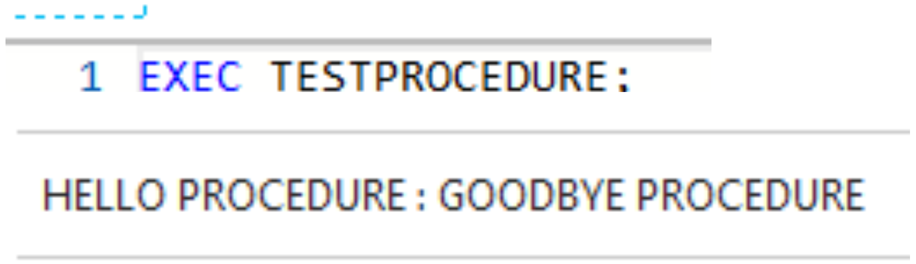
```
HELLO WORLD : GOODBYE WORLD
```

Stored Procedures

- *Have a name, stored in DB, can be called repeatedly*
- *no return value (stored functions do that)*



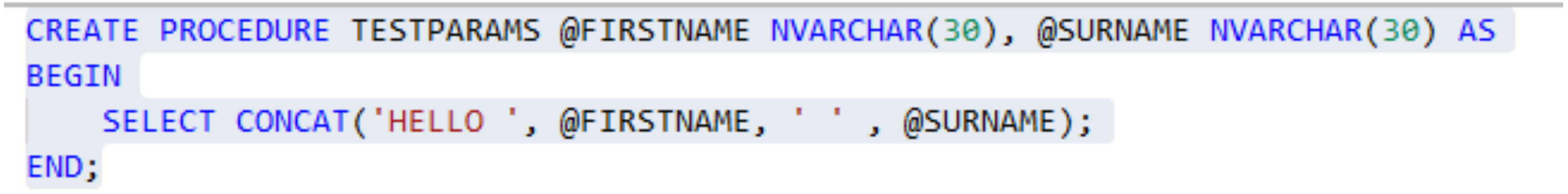
```
CREATE PROCEDURE TESTPROCEDURE AS
BEGIN
DECLARE @MYVAR NVARCHAR(100);
SELECT @MYVAR = 'GOODBYE PROCEDURE';
SELECT CONCAT('HELLO PROCEDURE : ', @MYVAR);
END;

Query succeeded: Affected rows: 0.
```



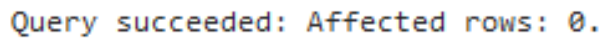
```
EXEC TESTPROCEDURE;

HELLO PROCEDURE : GOODBYE PROCEDURE
```

Parameters

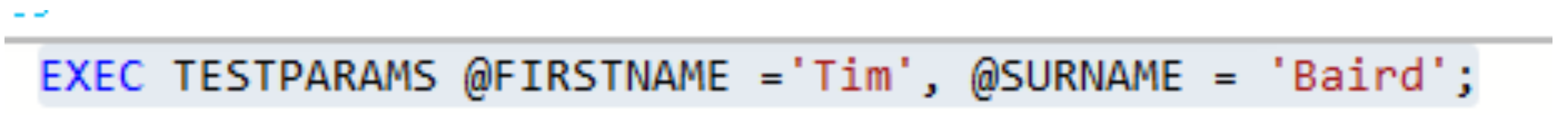
- Used to pass input into stored procedure (and functions)



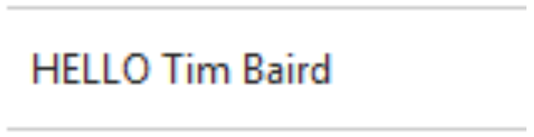
```sql
CREATE PROCEDURE TESTPARAMS @FIRSTNAME NVARCHAR(30), @SURNAME NVARCHAR(30) AS
BEGIN
    SELECT CONCAT('HELLO ', @FIRSTNAME, ' ' , @SURNAME);
END;
```



```
Query succeeded: Affected rows: 0.
```

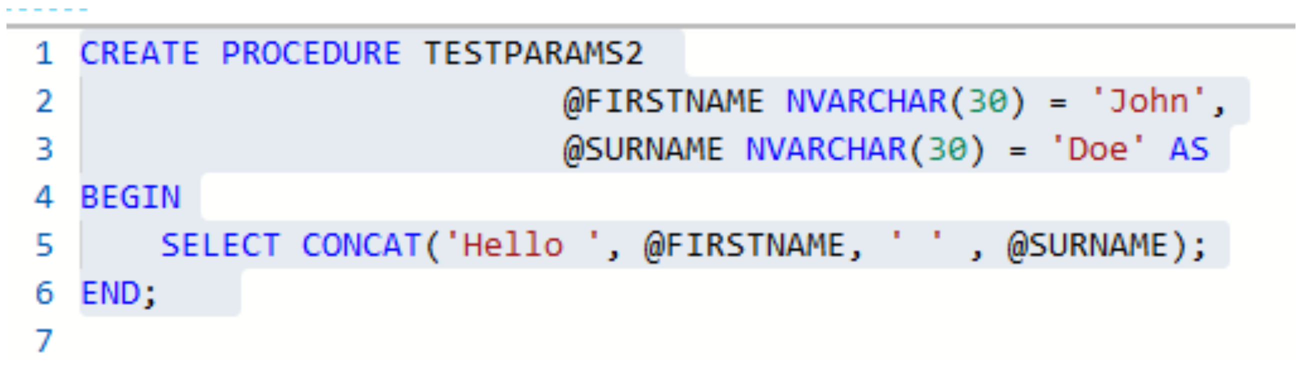


```sql
--
EXEC TESTPARAMS @FIRSTNAME ='Tim', @SURNAME = 'Baird';
```

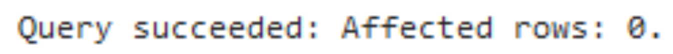


```
HELLO Tim Baird
```

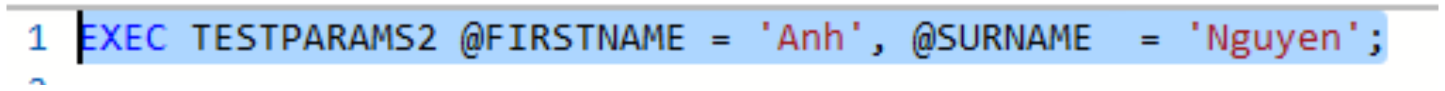
Default Parameter Values



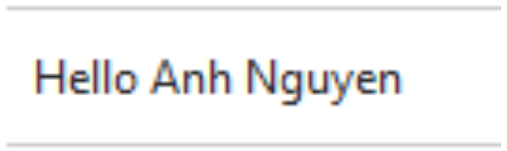
```
CREATE PROCEDURE TESTPARAMS2
                        @FIRSTNAME NVARCHAR(30) = 'John',
                        @SURNAME NVARCHAR(30) = 'Doe' AS
BEGIN
    SELECT CONCAT('Hello ', @FIRSTNAME, ' ' , @SURNAME);
END;
```



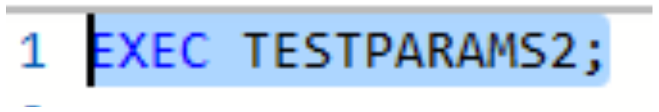
```
Query succeeded: Affected rows: 0.
```



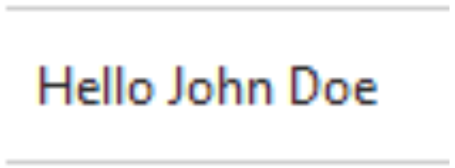
```
EXEC TESTPARAMS2 @FIRSTNAME = 'Anh', @SURNAME  = 'Nguyen';
```



```
Hello Anh Nguyen
```

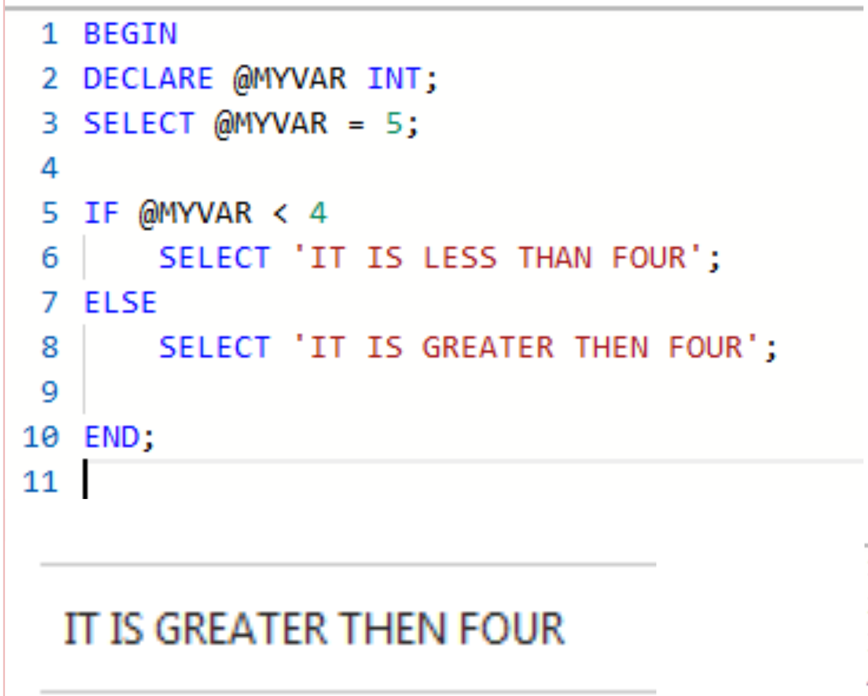


```
EXEC TESTPARAMS2;
```



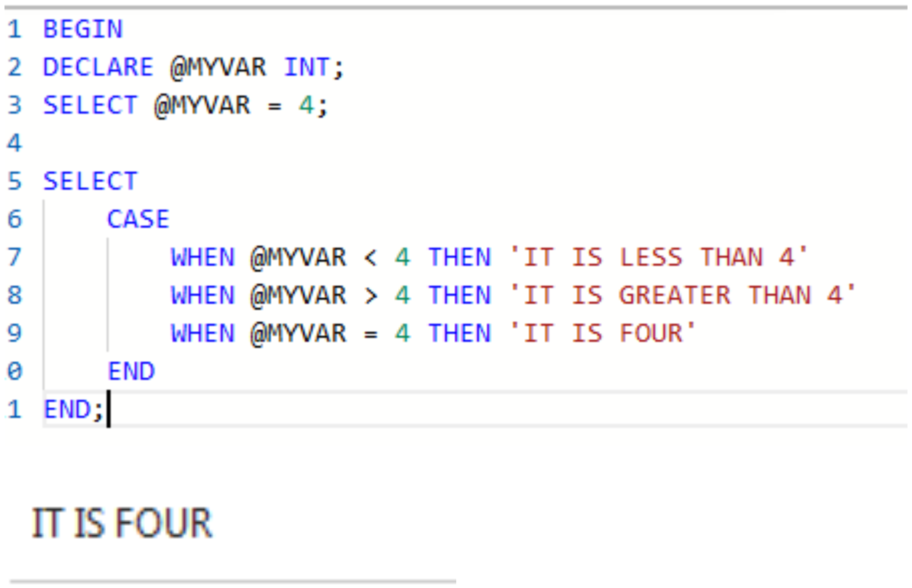
```
Hello John Doe
```

Conditional Logic



```sql
BEGIN
DECLARE @MYVAR INT;
SELECT @MYVAR = 5;

IF @MYVAR < 4
    SELECT 'IT IS LESS THAN FOUR';
ELSE
    SELECT 'IT IS GREATER THEN FOUR';

END;

```

```
IT IS GREATER THEN FOUR
```

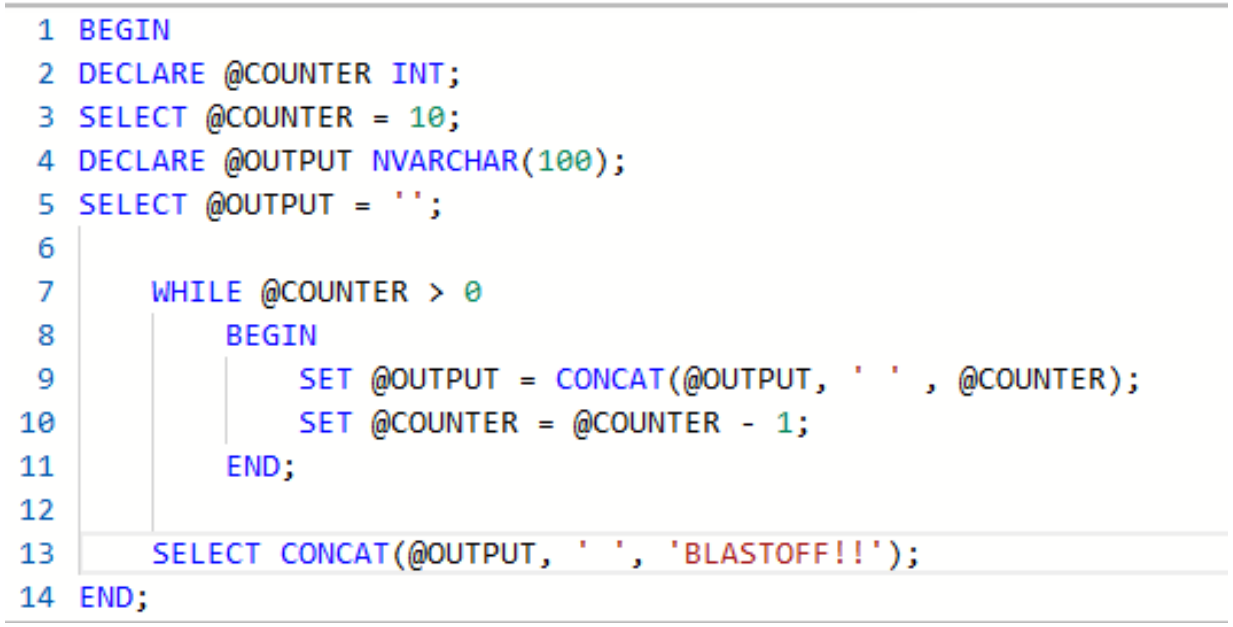


```sql
BEGIN
DECLARE @MYVAR INT;
SELECT @MYVAR = 4;

SELECT
    CASE
        WHEN @MYVAR < 4 THEN 'IT IS LESS THAN 4'
        WHEN @MYVAR > 4 THEN 'IT IS GREATER THAN 4'
        WHEN @MYVAR = 4 THEN 'IT IS FOUR'
    END
END;
```

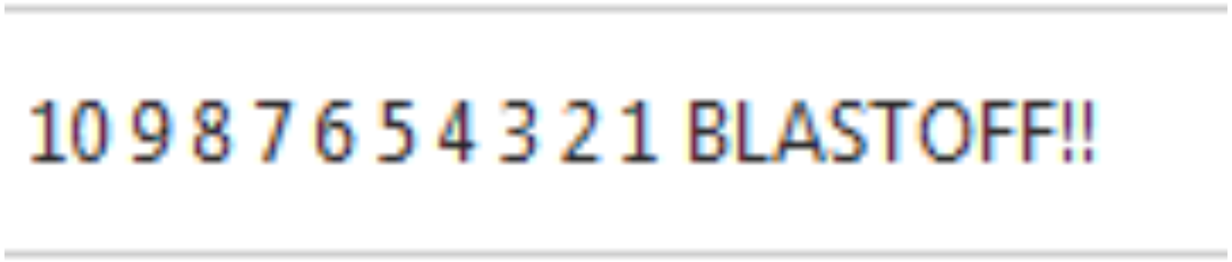
```
IT IS FOUR
```

Repetition (Loops)

- *No FOR loops – can simulate them with WHILE loops*



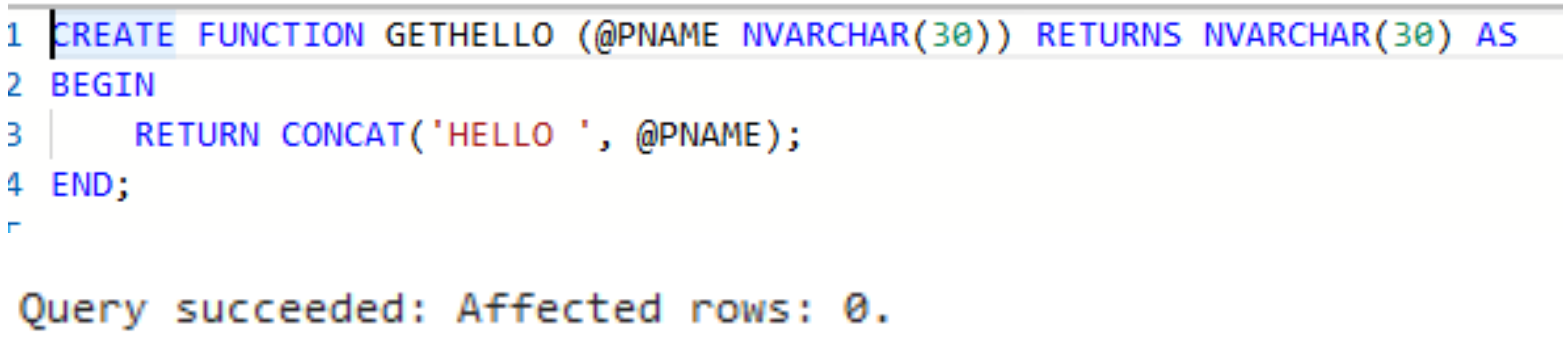
```sql
BEGIN
DECLARE @COUNTER INT;
SELECT @COUNTER = 10;
DECLARE @OUTPUT NVARCHAR(100);
SELECT @OUTPUT = '';

    WHILE @COUNTER > 0
        BEGIN
            SET @OUTPUT = CONCAT(@OUTPUT, ' ' , @COUNTER);
            SET @COUNTER = @COUNTER - 1;
        END;

    SELECT CONCAT(@OUTPUT, ' ', 'BLASTOFF!!');
END;
```



```
10 9 8 7 6 5 4 3 2 1 BLASTOFF!!
```

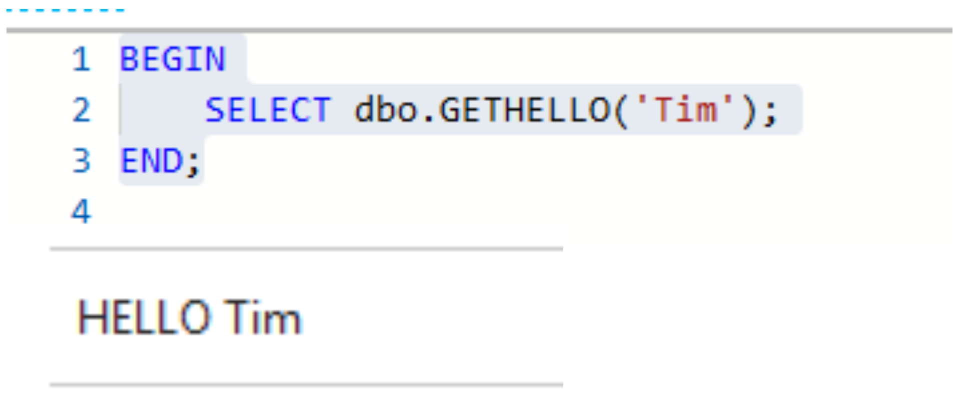
Stored Functions

- Returns a Value.
- This means you don't output the value from the function but rather have another block which calls the function and outputs its return value.



```
CREATE FUNCTION GETHELLO (@PNAME NVARCHAR(30)) RETURNS NVARCHAR(30) AS
BEGIN
    RETURN CONCAT('HELLO ', @PNAME);
END;
```

```
Query succeeded: Affected rows: 0.
```



```
BEGIN
    SELECT dbo.GETHELLO('Tim');
END;
```

```
HELLO Tim
```