Unit Testing T-SQL code with tSQLt

What can we test for?

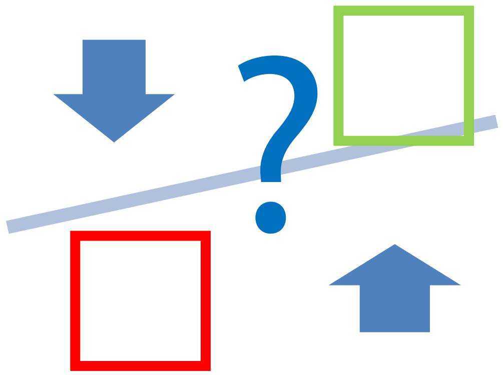
Dave Green
Twitter: @d_a_green
Email: dave@dgta.co.uk

pluralsight
hardcore developer training

Why Assert?

- **Third part of test flow:**
  - Assemble
  - Act
  - Assert

- **Tells tSQLt if the test fails**



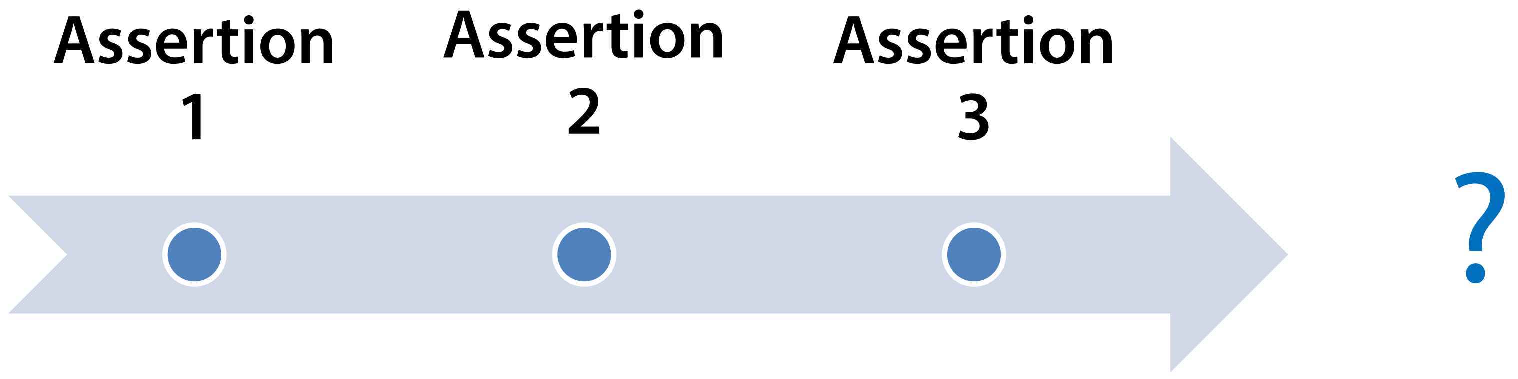
Assertions

- **The question we are testing determines which assertion we use**

- **Default test behavior is to pass**
  - If any assertion in the test fails, the test will fail

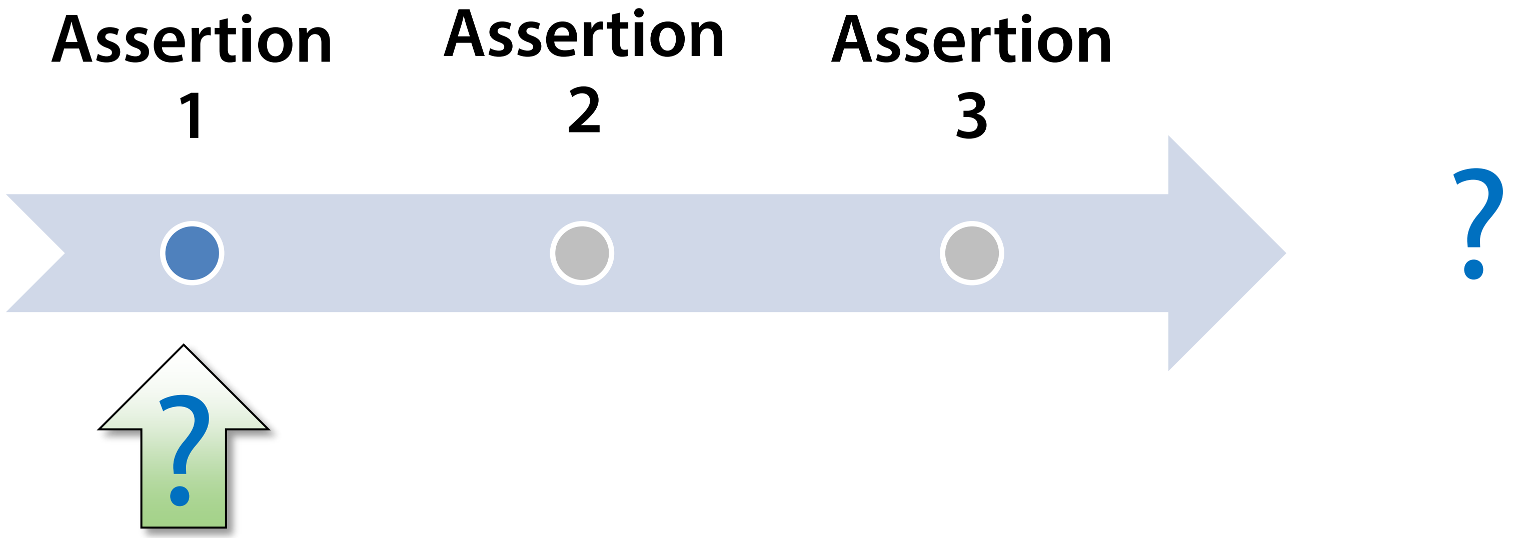
Assertions

- Assertions can be combined



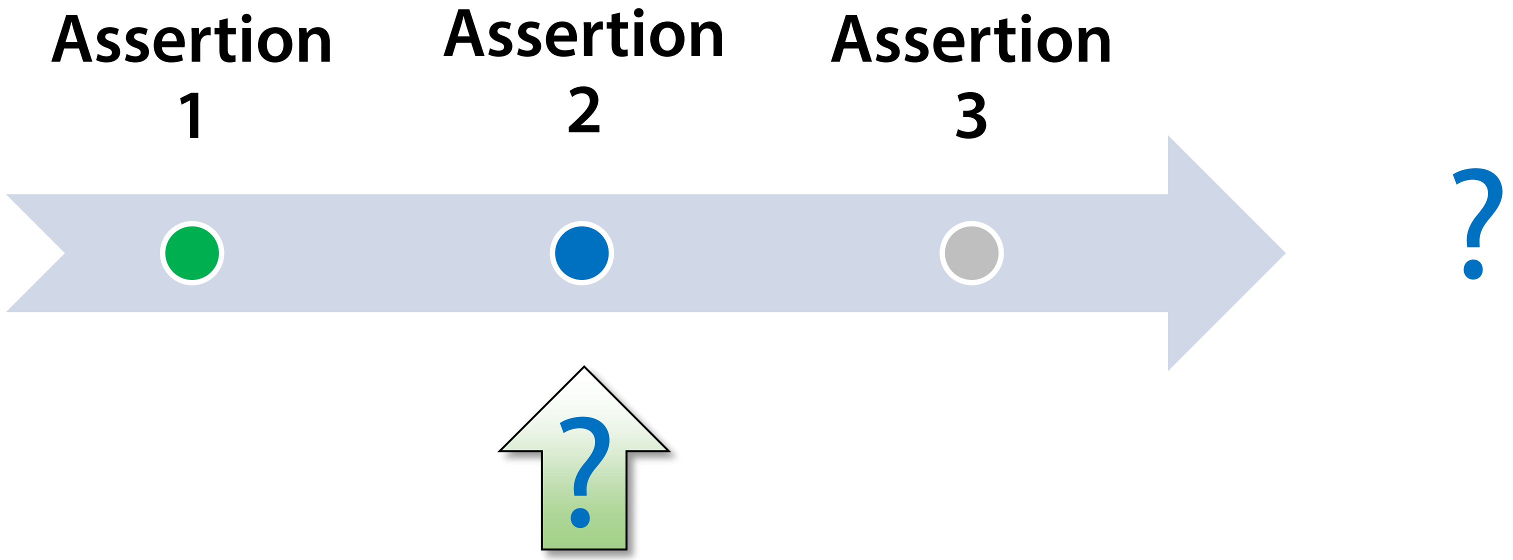
Assertions

- Assertions can be combined



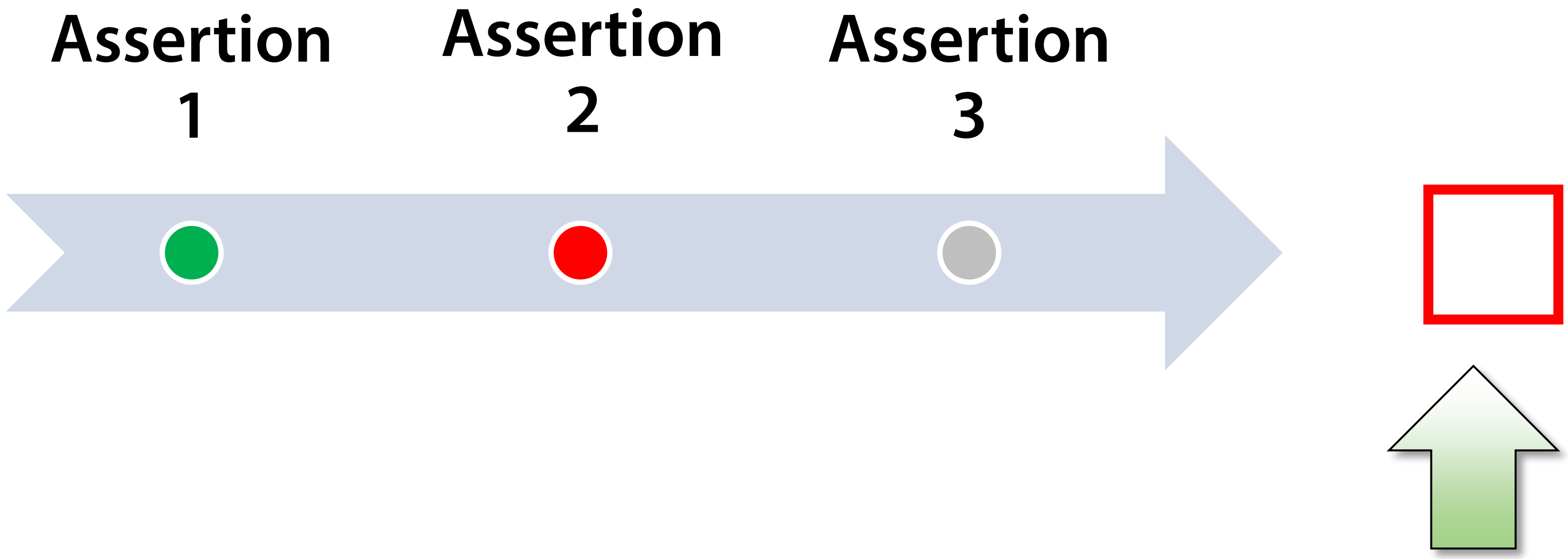
Assertions

- **Assertions can be combined**
  - Once an assertion passes, tSQLt moves to the next step in the test



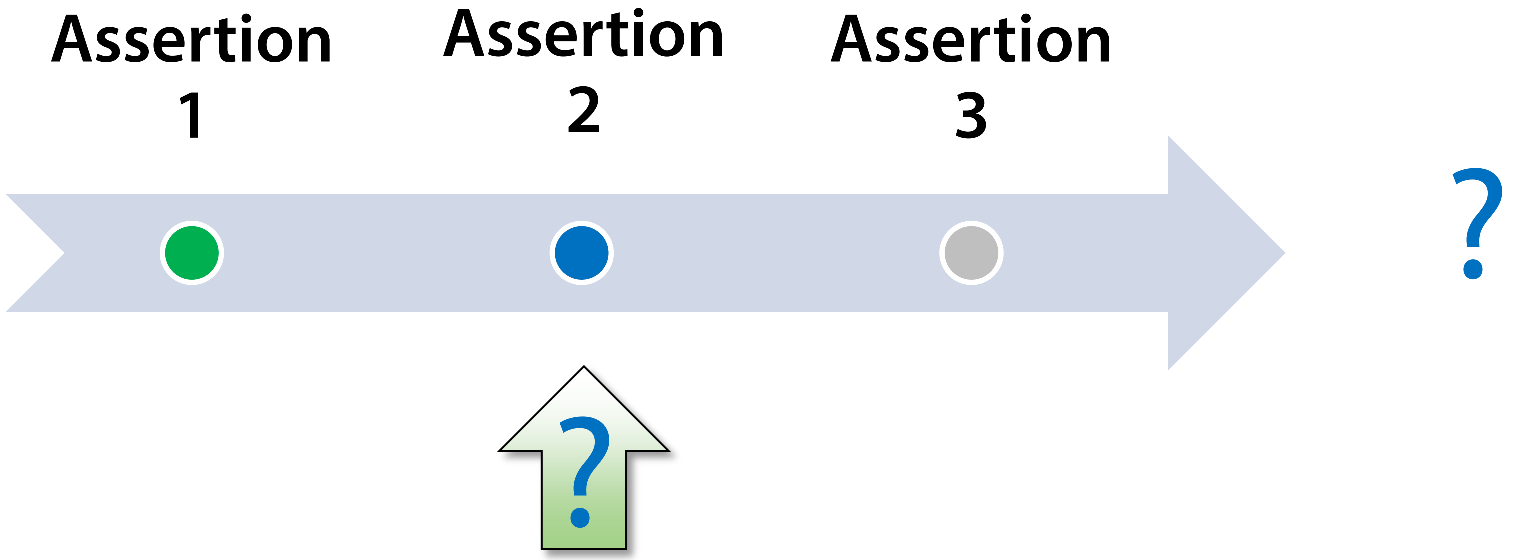
Assertions

- **Assertions can be combined**
  - Once an assertion passes, tSQLt moves to the next step in the test
  - If any assertion in the test fails, the test will fail



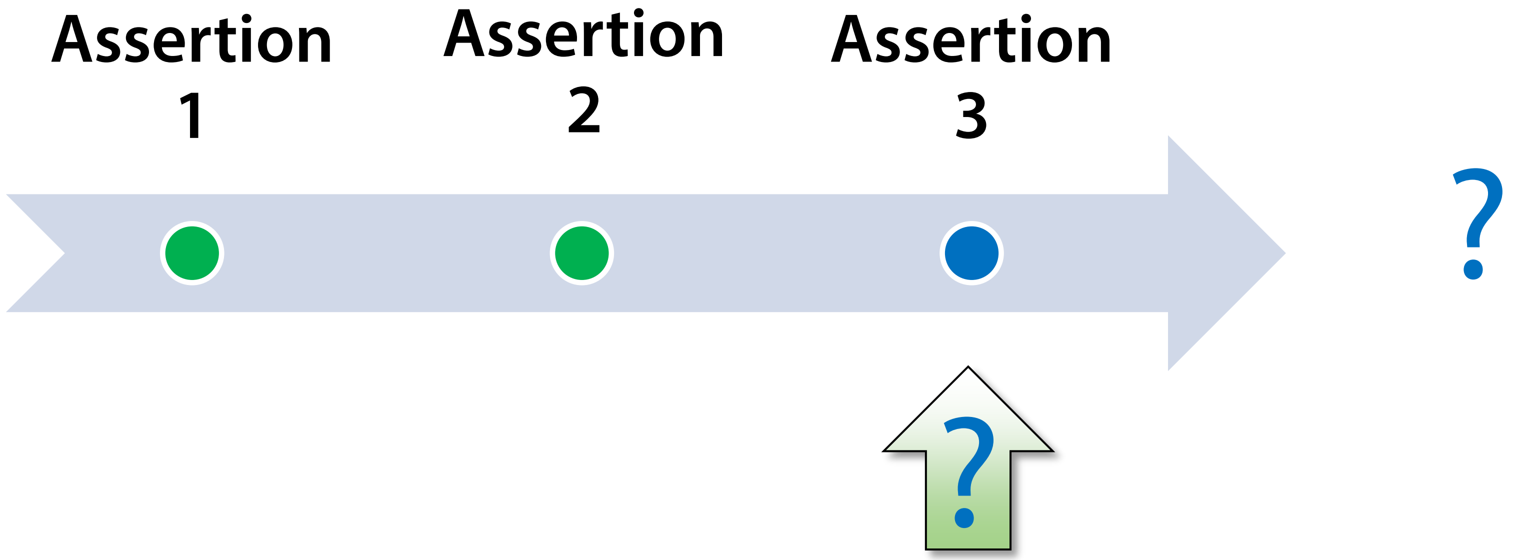
Assertions

- **Assertions can be combined**
  - Once an assertion passes, tSQLt moves to the next step in the test
  - If any assertion in the test fails, the test will fail



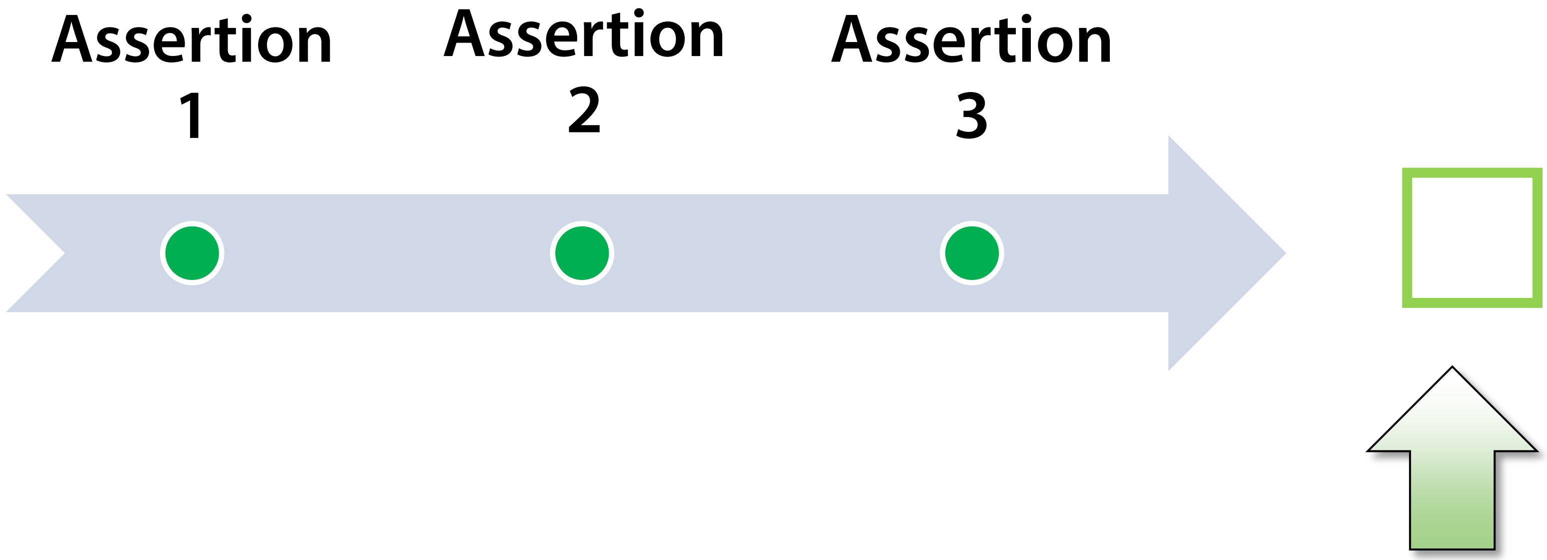
Assertions

- **Assertions can be combined**
  - Once an assertion passes, tSQLt moves to the next step in the test
  - If any assertion in the test fails, the test will fail
  - It is only when all assertions pass and we reach the end that the test passes



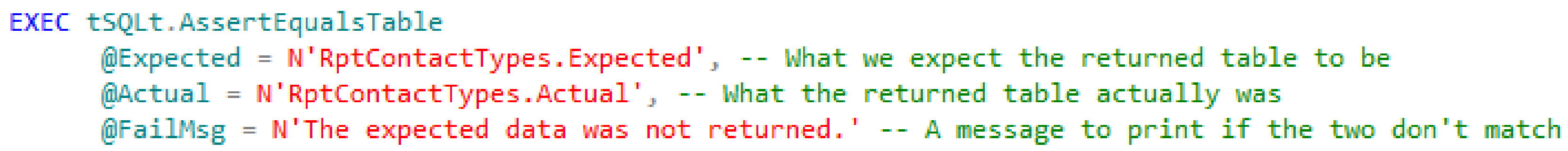
Assertions

- **Assertions can be combined**
  - Once an assertion passes, tSQLt moves to the next step in the test
  - If any assertion in the test fails, the test will fail
  - It is only when all assertions pass and we reach the end that the test passes



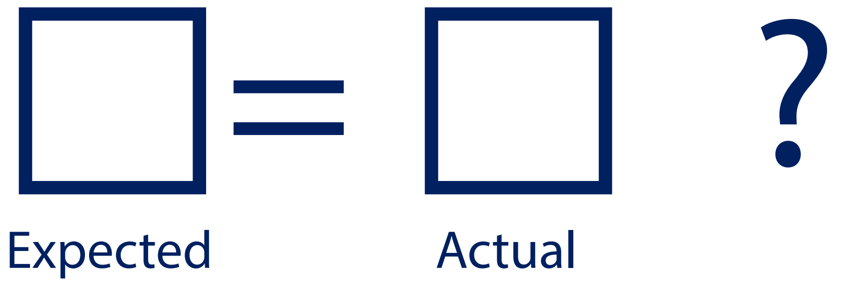
Checking data in tables

- **tSQLt.AssertEqualsTable**



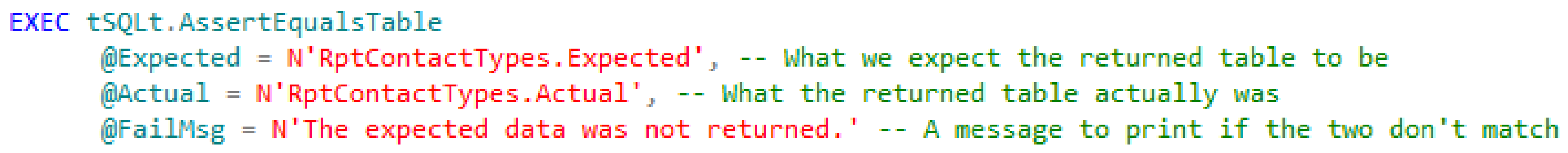
```
EXEC tSQLt.AssertEqualsTable
      @Expected = N'RptContactTypes.Expected', -- What we expect the returned table to be
      @Actual = N'RptContactTypes.Actual', -- What the returned table actually was
      @FailMsg = N'The expected data was not returned.' -- A message to print if the two don't match
```

- **Compares tables**
  - You can insert result sets into a table then compare



Checking data in tables

- tSQLt.AssertEqualsTable



```
EXEC tSQLt.AssertEqualsTable
      @Expected = N'RptContactTypes.Expected', -- What we expect the returned table to be
      @Actual = N'RptContactTypes.Actual', -- What the returned table actually was
      @FailMsg = N'The expected data was not returned.' -- A message to print if the two don't match
```

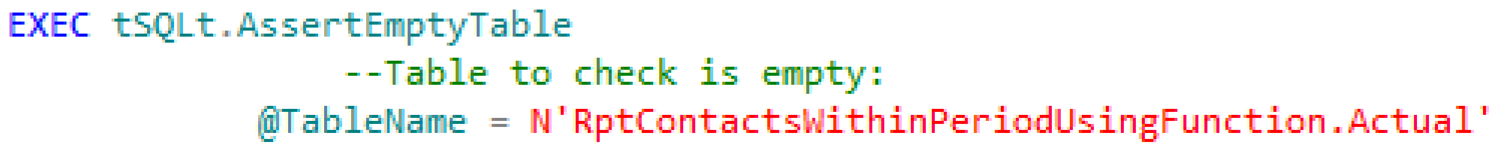
- Failure message is optional, but helps troubleshooting
- Not all datatypes are supported
- Only columns that are present in the expected table are compared

What if I don’t want any data returned?

Absence of data rows can be tested in two ways:

1. **Calling tSQLt.AssertEqualsTable with a blank expected table**
   - We saw this in the previous module
   - Supported in earlier versions of tSQLt

2. **tSQLt.AssertEmptyTable**
   - Raises an error if table is not empty
   - No need to set up expected table



```
EXEC tSQLt.AssertEmptyTable
                --Table to check is empty:
            @TableName = N'RptContactsWithinPeriodUsingFunction.Actual'
```

Checking Tables

What else needs checking?

- **Structure and shape of the data**
- **What columns exist in my source data?**
- **tSQLt.AssertEqualsTable does not check that columns exist, purely that data in the columns that do exist, match.**
- **This minimizes test revision when looking at wide tables**
    - No need to change each test when adding a new column
    - However test scenarios should be reviewed to check they are still valid

What does it matter what my data types are?

- What do we mean when we say “ the value should be 5”?

Value	Data Type	Storage Space	Range
“5”	Varchar(10)	Up to 12 bytes	-999999999 -> 9999999999
5	Int	4 bytes	-2,147,483,648 -> 2,147,483,647
5	Tinyint	1 byte	0 -> 255

Checking shape and structure of data

How can we check metadata?

- **tSQLt.AssertResultSetsHaveSameMetaData**
- **Compares columns and datatypes**
- **Not run as part of tSQLt.AssertEqualsTable**

Review

Assertion	Checks	Typical use
AssertEqualsTable	Table data	Check a result Set
AssertEmptyTable	Table contains no data	Testing for no output data
AssertResultSetsHaveSameMetaData	Table Metadata	Checking columns & data types

What if my code doesn’t return a table?

- **Straight string comparison with tSQLt.AssertEqualsString**

- **Useful for :**
  - Feedback from functions
  - Messages to user
  - Checking for specific, known messages

- **All inputs are NVARCHAR(MAX)**

- **NULL=NULL**

My code returns a string that changes each run!

- **Wildcard string comparison with tSQLt.AssertLike**
  - Uses LIKE Syntax

- **Useful for :**
  - Messages which vary with a predictable pattern

    "This proc ran in XX ms" (duration of run)

    "Query ran at XX:XX" (time of run)

- **All inputs are NVARCHAR(MAX)**

- **NULL=NULL**

Checking Values

Functions and output parameters frequently return specific values

- **These can be checked with tSQLt.AssertEquals**

- **Or tSQLt.AssertNotEquals**

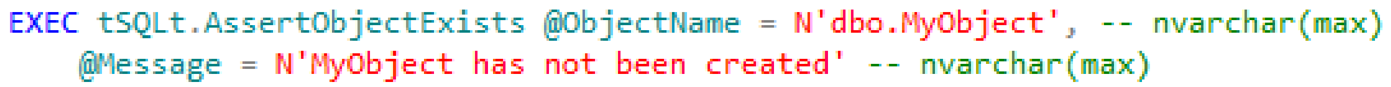
Review

Assertion	Checks	Typical use
AssertEqualsTable	Table data	Check a result Set
AssertEmptyTable	Table contains no data	Testing for no output data
AssertResultSetsHaveSameMetaData	Table Metadata	Checking columns & data types
AssertEqualsString	2 nvarchar(max) strings for exact match	Checking large strings are the same
AssertLike	One nvarchar string against a pattern	Checking strings against a pattern, e.g. message of time taken to execute
AssertEquals	2 values	Output from a function
AssertNotEquals	2 values (to ensure they are different)	Output from a function

Code that creates objects

- If your code creates an object, this needs to be tested
- You may need to remove existing objects first

You can check an object exists using tSQLt.AssertObjectExists:



```
EXEC tSQLt.AssertObjectExists @ObjectName = N'dbo.MyObject', -- nvarchar(max)
    @Message = N'MyObject has not been created' -- nvarchar(max)
```

Exceptions

Produced when :

- RAISERROR is used
- SQL Server encounters an error
- THROW is used

Severity	Meaning	Cancels Execution	tSQLt can test for
0-10	Warning	No	No (not an exception)
11-15	Error – special meanings	Sometimes	Yes
16	Error – user-correctable issue – Default	Sometimes	Yes
17-19	Errors that cannot be corrected by the user	Sometimes	Yes
20-24	System errors	Yes – closes connection	No

Exceptions

- **Sometimes we want to get the exception, other times we want to ensure we don't see it.**
- **Default behaviour is that an exception will cause a test to ERROR**
  - Can cause rollback issues
- **We need to anticipate the exception and plan our test accordingly**
- **Expectations not Assertions**

Review

Assertion	Checks	Typical use
AssertEqualsTable	Table data	Check a result Set
AssertEmptyTable	Table contains no data	Testing for no output data
AssertResultSetsHaveSameMetaData	Table Metadata	Checking columns & data types
AssertEqualsString	2 nvarchar(max) strings for exact match	Checking large strings are the same
AssertLike	One nvarchar string against a pattern	Checking strings against a pattern, e.g. message of time taken to execute
AssertEquals	2 values	Output from a function
AssertNotEquals	2 values (to ensure they are different)	Output from a function
AssertObjectExists	Object exists in the database	To test code that creates objects
ExpectException	An exception is raised	To check error condition
ExpectNoException	No exception is raised	To check error is handled

Review

Assertion	Checks	Typical use
AssertEqualsTable	Table data	Check a result Set
AssertEmptyTable	Table contains no data	Testing for no output data
AssertResultSetsHaveSameMetaData	Table Metadata	Checking columns & data types
AssertEqualsString	2 nvarchar(max) strings for exact match	Checking large strings are the same
AssertLike	One nvarchar string against a pattern	Checking strings against a pattern, e.g. message of time taken to execute
AssertEquals	2 values	Output from a function
AssertNotEquals	2 values (to ensure they are different)	Output from a function
AssertObjectExists	Object exists in the database	To test code that creates objects
ExpectException	An exception is raised	To check error condition
ExpectNoException	No exception is raised	To check error is handled

Summary

- **Assertions are what establish whether our criteria have been met (or not)**
- **Assertions can be chained together**
  - To answer one question
- **Assertions are available to check a variety of circumstances**
- **Exceptions can be checked too**