



# SQL Fast Track

Sample manual - first two chapters



**Wise Owl**  
Training

## TABLE OF CONTENTS (1 of 9)

1	DESIGNING DATABASES	Page
1.1	The Four Stages of Database Design	11
	<i>Stage 1 – Deciding what to Include</i>	11
	<i>Stage 2 – Dividing Data into Tables</i>	12
	<i>Stage 3 – Choosing a Primary Key for each Table</i>	13
	<i>Stage 4 – Creating Relationships and a Database Diagram</i>	14
1.2	Many-to-Many Relationships	15

2	SQL SERVER MANAGEMENT STUDIO	Page
2.1	Starting to Use Management Studio	16
2.2	Object Explorer	17
	<i>Useful Start-up Options</i>	17

3	CREATING A DATABASE AND TABLES	Page
3.1	Creating the Database	18
	<i>The Files Created</i>	18
3.2	Creating Tables	19
3.3	Setting an Identity Primary Key	20
3.4	Creating Columns	21
	<i>Data Types Explained in this Chapter</i>	21
	<i>Other Data Types in SQL Server</i>	21
3.5	Whole Numbers	22
	<i>Integer Field Types</i>	22
	<i>Logical Field Types</i>	22
3.6	Other Numerical Fields	23
	<i>Decimal and Numeric Field Types</i>	23
	<i>Float and Real Data Types</i>	23
3.7	Character Data Types	24
	<i>Types of Character Storage</i>	24
	<i>Variable Length Data Types</i>	24
	<i>Fixed Length Data Types</i>	24
3.8	Date/Time Data Types	25
	<i>Entering Dates into a Table</i>	25
3.9	Default and Null Values	26
	<i>Allowing Nulls</i>	26
	<i>Default Values</i>	26
3.10	Database Diagrams	27
	<i>Creating a Database Diagram</i>	27
	<i>Creating Relationships</i>	28
	<i>Database Diagram Support Objects Error</i>	28

4	QUERIES	Page
4.1	Basic SELECT Statements	29
	<i>Where to Put your Commas</i>	29
4.2	Creating Queries	30
	<i>Starting a New Query</i>	30
	<i>Choosing the Right Database</i>	30
4.3	Running Queries	31
	<i>Parsing a Query</i>	31
	<i>Executing a Query</i>	31
	<i>Viewing Information on a Query's Execution</i>	32
	<i>Cancelling a Running Query</i>	32
	<i>Redirecting Query Output</i>	32
4.4	Dealing with Errors	33
	<i>Displaying Line Numbers</i>	33
4.5	Using IntelliSense	34
	<i>Refreshing IntelliSense</i>	34
4.6	Multiple SQL Commands	35
4.7	Saving, Opening and Closing Queries	36
	<i>Saving Queries</i>	36
	<i>Opening Queries</i>	36
	<i>Closing Queries</i>	37

5	LAYING OUT QUERIES	Page
5.1	Using Case	38
5.2	Indentation and Word Wrap	39
	<i>Changing Tab Settings</i>	39
	<i>Word Wrap</i>	39
5.3	Comments	40
	<i>Commenting Out Blocks of Code</i>	40
5.4	Colours in SQL	41
	<i>Changing the Default Colours</i>	41
5.5	Auto-formatting SQL	42

## TABLE OF CONTENTS (2 of 9)

6	THE SELECT STATEMENT	Page
6.1	SELECT Statement Syntax	43
	<i>Mnemonic for Order of Commands</i>	43
6.2	Qualified Tables and Columns	44
	<i>Dragging Tables/Columns onto a Query</i>	44
	<i>Specifying the DBO Schema and Database</i>	44
6.3	Table Aliases	44
	<i>Reason 1 for Aliases – Easier to Refer to Field Names</i>	45
	<i>Reason 2 for Aliases – Joins</i>	46
	<i>Changing a Table Alias</i>	46
6.4	Column Aliases	47
	<i>Basic Column Aliases</i>	47
	<i>Other Ways to Create Column Aliases</i>	47
	<i>Aliases in WHERE and ORDER BY Clauses</i>	48
6.5	Ordering Rows	49
	<i>Simple Sorting</i>	49
	<i>Sorting by Multiple Columns</i>	49
6.6	Miscellaneous SELECT Tricks	50
	<i>Selecting All Columns Using *</i>	50
	<i>Selecting Unique Rows</i>	50
	<i>Showing Top and Bottom Rows</i>	51
	<i>Including Ties</i>	51
6.7	Using UNION to Combine Results	52

7	QUERY DESIGNER	Page
7.1	Starting Query Designer	53
	<i>What Query Designer is and does</i>	53
	<i>Starting Query Designer</i>	53
7.2	Using Query Designer	54
	<i>Choosing Tables</i>	54
	<i>Adding/Removing Tables</i>	54
	<i>The Parts of Query Designer</i>	55
	<i>The Non-Existent Results Pane/Execute Button</i>	55
	<i>Working with Columns</i>	56
	<i>Finishing Work in Query Designer</i>	56
7.3	Editing Generated SQL	57
7.4	Advanced Features	58
	<i>Inner and Outer Joins</i>	58
	<i>Grouping</i>	58

8	CRITERIA USING WHERE	Page
8.1	The WHERE Clause	59
	<i>Relational Operators</i>	59
8.2	Criteria with Numbers	60
	<i>Using Comparisons</i>	60
	<i>Finding Numbers in a Given Range</i>	60
8.3	Criteria using Text	61
	<i>Exact Matches</i>	61
	<i>Wildcard Matches using LIKE</i>	61
	<i>Using Special Characters with LIKE</i>	62
	<i>Ranges and Wildcards</i>	62
	<i>Using Relational Operators with Text</i>	63
	<i>Case Sensitivity</i>	63
8.4	Criteria for Dates	64
	<i>Using Dates in Criteria</i>	64
	<i>Using Dates with Wildcards</i>	64
8.5	Combining Criteria	65
8.6	Nulls	66
	<i>An Example of Testing for Nulls</i>	66
	<i>Entering Nulls into a Table</i>	66

9	EXPORTING TO EXCEL	Page
9.1	Copying and Pasting	67
	<i>Copying Column Headers by Default</i>	67
9.2	Exporting Data	68
	<i>Step 1 – Getting the Query</i>	68
	<i>Step 2 – Starting to Export Data</i>	68
	<i>Step 3 – Choosing the Source</i>	68
	<i>Step 4 – Choosing the Destination</i>	69
	<i>Step 5 – Specifying the Data to Export</i>	70
	<i>Step 6 – Specifying how to Export</i>	70
	<i>Step 7 – Finishing the Export</i>	70

## TABLE OF CONTENTS (3 of 9)

10	CALCULATIONS	Page
10.1	Creating Calculated Columns	72
	<i>Giving Calculated Columns Aliases</i>	72
	<i>Using Column Aliases in ORDER BY Clauses</i>	72
	<i>Column Aliases Don't Work in WHERE Criteria</i>	73
10.2	Using SQL Functions	74
	<i>Typing an SQL Function</i>	74
	<i>Getting the Full List of Functions</i>	74
10.3	Casting Data Types	75
	<i>The Need for Casting</i>	75
	<i>Data Type Precedence</i>	76
	<i>The CAST Function</i>	77
	<i>The CONVERT Function</i>	77
10.4	Numerical Calculations	78
	<i>Mathematical Symbols and BODMAS</i>	78
	<i>The Modulus Operator (%)</i>	78
	<i>Mathematical Functions</i>	79
	<i>The Importance of Casting Numbers for Calculations</i>	80
	<i>A Short-Cut to Forcing the Right Number Type</i>	80
10.5	Text Calculations	81
	<i>Concatenating Text using the CONCAT Function</i>	81
	<i>Concatenating Text Using the Plus Sign (with Data Conversion)</i>	81
	<i>Functions to Turn Numbers into Text</i>	83
	<i>Functions to Search for and Replace Text</i>	83
	<i>Functions for Extracting Text</i>	84
	<i>Changing the Case of Text</i>	84
	<i>Functions for Trimming Text</i>	85
	<i>Other Text Functions</i>	85
	<i>Worked Example – 1</i>	86
	<i>Worked Example – 2</i>	86
	3	86
10.6	Dealing with Nulls	87
	<i>The ISNULL Function</i>	87
	<i>The COALESCE Function</i>	88
10.7	Testing Conditions using IIF	89

11	THE CASE EXPRESSION	Page
11.1	The Searched Case Expression	90
	<i>Example: Film Bands</i>	90
	<i>Example: Film Era</i>	91
	<i>Using CASE in WHERE Criteria</i>	91
11.2	The Simple Case Statement	92
11.3	Nested CASE Statements	93

12	DATE CALCULATIONS	Page
12.1	How Dates and Times Work	94
	<i>How SQL Server Stores Dates and Times</i>	94
	<i>Displaying Dates/Times</i>	94
	<i>GETDATE – the Current Date/Time</i>	95
	<i>Dates Prefer American Format</i>	95
12.2	Formatting Dates using FORMAT	96
	<i>The Available Codes</i>	96
	<i>Using the Culture Argument</i>	97
	<i>How Slow is the Format Function?</i>	97
12.3	Formatting Dates using CONVERT	98
12.4	Parts of a Date: DATEPART and DATENAME	99
	<i>Displaying a Day Suffix</i>	100
12.5	Getting the Difference between Dates	100
	<i>Subtracting One Date from Another</i>	101
	<i>The DATEDIFF Function</i>	101
12.6	Calculating Ages Correctly	102
	<i>Using DateDiff</i>	102
	<i>Dividing Someone's Age in Days by 365</i>	102
	<i>Getting the Exact Age</i>	103
12.7	Adding Dates using DATEADD	104

## TABLE OF CONTENTS (4 of 9)

13	JOINS	Page
13.1	Overview of Joins	105
	<i>What is a Join?</i>	105
	<i>The Types of Join</i>	105
13.2	Understanding your Database	106
	<i>How Relationships Work (Reminder)</i>	106
13.3	Easy Joins, using Query Designer	107
13.4	Inner Joins	108
	<i>The Syntax of an Inner Join</i>	108
	<i>Our Example – Joining the Film and Director Tables</i>	108
	<i>Joining more than One Table</i>	109
	<i>Variations on Inner Join Syntax</i>	109
	<i>Composite Joins</i>	110
	<i>Joining by Expressions</i>	110
13.5	Outer Joins	112
	<i>Outer Joins using Query Designer</i>	112
	<i>Outer Joins in SQL</i>	113
	<i>Left and Right Outer Joins</i>	113
	<i>Picking Out Unmatched Rows</i>	114
	<i>Full Outer Joins</i>	114
13.6	Cross Joins	115
	<i>A Practical Example of Cross Joins</i>	115
13.7	Self-Joins	116

14	SUMMARISING DATA	Page
14.1	Simple Summarising	117
	<i>Syntax of a Simple Summary</i>	117
14.2	Counting	118
	<i>Counting All of a Table's Rows</i>	118
	<i>Counting Non-Null Columns</i>	118
	<i>Counting Unique Values</i>	119
14.3	Grouping	120
	<i>Why you Need GROUP BY</i>	120
	<i>The GROUP BY Clause</i>	120
	<i>Grouping by Multiple Columns</i>	121
	<i>Grouping by Expressions</i>	121
	<i>Grouping without Aggregating</i>	122
14.4	Filtering Results using HAVING	123
14.5	Casting Data for (eg) Averages	124
14.6	Dealing with Nulls	125
	<i>The Default Treatment of Nulls</i>	125
	<i>Forcing SQL to Include Nulls</i>	125
14.7	Additional Options when Grouping	126
	<i>Using ALL to Show Missing Rows</i>	126
	<i>Using CUBE to Show All Combinations</i>	126
	<i>Using GROUPING to Show Levels</i>	127

15	VIEWS	Page
15.1	Why Views are Useful	128
	<i>Use 1: Pre-Joining Tables</i>	128
	<i>Use 2: Virtually Renaming Columns</i>	129
15.2	Views using the Designer	130
	<i>Starting the Designer</i>	130
	<i>Choosing Columns</i>	130
	<i>Sorting and Filtering</i>	131
	<i>Adding Grouping</i>	131
	<i>Executing a View</i>	132
	<i>Saving and Closing Views</i>	133
	<i>Seeing your View in Object Explorer</i>	133
	<i>Running a View</i>	134
	<i>Changing a View</i>	134
15.3	Scripting Views	135
	<i>Creating a New View</i>	135
	<i>Changing an Open View in Script</i>	136
	<i>Changing a View's Script from Object Explorer</i>	136
15.4	Switching between the Designer and Scripting	137

## TABLE OF CONTENTS (5 of 9)

16	CTES AND DERIVED TABLES	Page
16.1	Multi-Stage Queries	138
16.2	Derived Tables	139
16.3	Single CTEs (Common Table Expressions)	140
	<i>Syntax of Single CTEs</i>	140
	<i>The CTE for our Example</i>	140
16.4	Multiple CTEs	141
	<i>Syntax of Multiple CTEs</i>	141
	<i>Example of a Multiple CTE</i>	142

17	SUBQUERIES	Page
17.1	Single-Value Subqueries	143
	<i>Example: Showing the Name of the Longest Film</i>	143
17.2	ANY, ALL, IN and NOT IN	144
17.3	Correlated Subqueries	145
	<i>Correlated Subqueries: Definition and Example</i>	145
	<i>Alternatives to Correlated Subqueries</i>	145
	<i>Considering Speed</i>	146
	<i>Using EXISTS to Check whether Rows are Returned</i>	146

18	RANKING AND PERCENTILES	Page
18.1	Ranking and Numbering	147
	<i>Simple Row Numbering</i>	147
18.2	Leading and Lagging	148
	<i>Example of LAG: Actors Born One Week Apart</i>	148
18.3	Percentiles	149
	<i>Percentile Rankings</i>	149

19	STORED PROCEDURES	Page
19.1	Overview	150
	<i>What is a Stored Procedure?</i>	150
	<i>Advantages and Disadvantages</i>	150
19.2	Creating Stored Procedures	151
	<i>Typing in a Stored Procedure</i>	151
	<i>Creating a Stored Procedure using a Template</i>	151
	<i>Executing the Query to Create your Stored Procedure</i>	152
	<i>Viewing your Stored Procedure</i>	152
19.3	Altering a Stored Procedure	153
	<i>Altering an Open Stored Procedure</i>	153
	<i>Altering a Procedure in a Database</i>	153
19.4	Executing Stored Procedures	154
	<i>Refreshing your Local Cache</i>	154
	<i>Executing a Procedure</i>	154
	<i>Altering and Executing a Stored Procedure Together</i>	155
	<i>Selecting a Stored Procedure Name to Run It</i>	155
19.5	Renaming and Deleting Stored Procedures	156
	<i>Renaming/Deleting a Procedure with the Menu</i>	156
	<i>Deleting a Procedure in Script</i>	156
	<i>Renaming a Procedure in Script</i>	156
19.6	System Stored Procedures	157
	<i>Listing System Stored Procedures</i>	157
	<i>Useful System Stored Procedures</i>	158
19.7	Getting Help on SQL	159
	<i>Context-Sensitive Help</i>	159
	<i>Tips on Googling</i>	159

## TABLE OF CONTENTS (6 of 9)

20	VARIABLES	Page
20.1	Declaring Variables	160
	<i>Syntax for Declaring a Variable</i>	160
20.2	Using Variables	161
	<i>Setting the Value of a Variable</i>	161
	<i>Showing the Values of Variables</i>	161
	<i>Scope of Variables</i>	162
	<i>Incrementing and Concatenating Variables</i>	162
	<i>The Importance of Casting</i>	163
20.3	Using Variables with Subqueries	164
	<i>An Alternative Approach: Aggregate Functions</i>	164
20.4	Storing Column Values in Variables	165
	<i>Storing a Single Row's Values</i>	165
	<i>Accumulating Numbers</i>	165
	<i>Accumulating Text</i>	166
20.5	Global Variables	167
	<i>Special Considerations when using @@ROWCOUNT</i>	167

22	STORED PROCEDURE PARAMETERS	Page
22.1	Overview	171
	<i>Syntax of Parameters</i>	171
22.2	Simple Parameters	172
	<i>Step 1 – Specifying the Parameters</i>	172
	<i>Step 2 – Coding the Parameters</i>	172
	<i>Step 3 – Referencing the Parameters</i>	173
	<i>Using Text Wildcards as Parameters</i>	173
22.3	Running Procedures using Parameters	174
	<i>Positional Arguments</i>	174
	<i>Named Arguments</i>	175
	<i>Right-clicking to Execute a Procedure</i>	175
22.4	Default Parameter Values	176
	<i>Setting Default Values to Null</i>	177
	<i>The Perfect Stored Procedure?</i>	177
22.5	The RETURN Statement	178
22.6	Output Parameters	179

21	VARIABLE AND PARAMETER DATA TYPES	Page
21.1	Numeric Data Types	168
	<i>Integer Variable/Parameter Types</i>	168
	<i>Decimal and Numeric Types</i>	168
21.2	Character Data Types	169
	<i>Types of Character Storage</i>	169
	<i>Variable Length Data Types</i>	169
	<i>Fixed Length Data Types</i>	169
21.3	Date/Time Data Types	170

23	CONDITIONS AND LOOPS	Page
23.1	IF Conditions	180
	<i>Simple Conditions</i>	180
	<i>Using BEGIN ... END</i>	180
	<i>Using ELSE</i>	181
	<i>Nesting Conditions and Indentation</i>	181
	<i>Using CASE to Avoid IF</i>	182
23.2	Looping using WHILE	183
	<i>The Syntax of WHILE Loops</i>	183
	<i>Breaking out of Loops</i>	184



## TABLE OF CONTENTS (7 of 9)

24	SCALAR FUNCTIONS	Page
24.1	Overview	185
	<i>Syntax of a Scalar Function</i>	185
24.2	Writing a Scalar Function	186
	<i>Specifying Input Parameters and Return Types</i>	186
	<i>Writing the Function Itself</i>	186
24.3	Running a Function	187
	<i>Calling a Function on its Own</i>	187
	<i>Calling a Function within a SELECT Statement</i>	187
24.4	Worked Examples	188
	<i>Example One – Returning a Person’s Status</i>	188
	<i>Example Two – Profitability</i>	189
	<i>Example Three – Categorisation (by Oscar Type)</i>	190
24.5	Limitations of Functions	191
	<i>Assessing Function Speed</i>	191

25	ERROR TRAPPING	Page
25.1	About Errors	192
25.2	TRY / CATCH	193
	<i>Syntax of TRY / CATCH</i>	193
	<i>Example of a Simple Error Trap</i>	193
	<i>Nesting TRY Statements</i>	194
25.3	Error Functions	195
	<i>T-SQL Error Functions</i>	195
	<i>Error Severity Levels</i>	196
	<i>Showing Errors within a TRY / CATCH Block</i>	196
25.4	Customising Error Messages	197
	<i>Viewing the Full List of Error Messages</i>	197
	<i>Creating your Own Errors</i>	198
	<i>Customising your own Error Messages</i>	198

26	DELETING DATA	Page
26.1	Deleting (Dropping) Tables	199
	<i>Dropping a Table if it Exists</i>	199
	<i>Using Error Trapping to Check Existence</i>	199
	<i>Using SYS.OBJECTS and OBJECT_ID</i>	200
26.2	Deleting Rows	201
	<i>Differences between TRUNCATE and DELETE FROM</i>	201

27	UPDATING DATA	Page
27.1	The UPDATE Command	202
	<i>An Example – Changing Genres for Films</i>	202
27.2	Updating using JOIN	203
	<i>The Obvious Answer doesn’t Work</i>	203
	<i>The Correct Syntax</i>	204

28	INSERTING DATA	Page
28.1	Three Possible Ways to Insert	205
28.2	Creating Tables from Existing Data (SELECT INTO)	206
	<i>Step 1 – Getting the Data for your New Table</i>	206
	<i>Step 2 – Making a New Table</i>	206
	<i>Step 3 – Checking the Table Created</i>	207
28.3	Inserting Multiple Rows into an Existing Table	208
	<i>Step 1 – Understanding the Syntax</i>	208
	<i>Step 2 – Working out what to do</i>	208
	<i>Step 3 – Mapping the Columns</i>	209
	<i>Step 4 – Creating the Query</i>	209
28.4	Inserting Single Rows	210
	<i>Syntax of INSERT INTO ... VALUES</i>	210
	<i>Example Code to Insert a New Row</i>	210
	<i>Inserting a Batch of Single Rows</i>	211
28.5	INSERT INTO – More Possibilities	212
	<i>Missing out Columns</i>	212
	<i>Using a Stored Procedure’s Output</i>	212
	<i>Outputting Inserted Rows</i>	213
	<i>Getting Inserted Row Numbers with @@IDENTITY</i>	213



## TABLE OF CONTENTS (8 of 9)

29	CREATING TABLES	Page
29.1	Setting Up our Example	214
	<i>The Example Used in this Chapter</i>	214
	<i>Creating and Dropping Databases</i>	214
29.2	Creating Tables	215
29.3	Setting Primary Keys	216
	<i>Creating a Primary Key when Creating Tables</i>	216
	<i>Creating a Primary Key Afterwards</i>	216
29.4	Setting a Default Value for a Column	217
29.5	Preventing Null Values in a Column	218
29.6	Putting Checks or Constraints on a Column	219
29.7	Foreign Keys and Relationships	220
	<i>Our Example</i>	220
	<i>Foreign Keys</i>	221
	<i>Creating a Foreign Key Constraint</i>	221
29.8	Two Reasons/Ways to Index a Column	222
	<i>Creating an Index to Speed Up Queries</i>	222
	<i>Enforcing Uniqueness with an Index</i>	222
29.9	A Complete Example	223

30	TRANSACTIONS	Page
30.1	The Concept	224
	<i>Syntax of a Transaction</i>	224
30.2	A Simple Example	225
30.3	Case Study – Recategorising Films	226
	<i>The Problem</i>	226
	<i>The Algorithm</i>	226
	<i>The Procedure</i>	227
30.4	Errors and Transactions	228

31	TEMPORARY TABLES	Page
31.1	Overview of Temporary Tables	229
	<i>Local and Global Temporary Tables</i>	229
	<i>How Temporary Tables are Stored</i>	229
31.2	Creating and Deleting Temporary Tables	230
	<i>Creating Temporary Tables</i>	230
	<i>Deleting Temporary Tables</i>	230
31.3	Scope of Temporary Tables	231
	<i>Temporary Tables are Tied to the Queries Creating Them</i>	231
	<i>Visibility of Temporary Tables</i>	232
	<i>Scope of Temporary Tables in Stored Procedures</i>	233
31.4	Case Study – Successful People	234
	<i>Step 1 – Busy Actors (Creating the Table)</i>	235
	<i>Step 2 – Busy Directors (Inserting Rows)</i>	235
	<i>Final Answer with Problems Solved</i>	236

32	TABLE VARIABLES	Page
32.1	About Table Variables	237
32.2	Case Study Revisited	238

33	COMPARING TABLE TYPES	Page
33.1	Differences between Table Variables and Temporary Tables	239
	<i>Speed</i>	239
	<i>Limitations of Table Variables</i>	240
	<i>Limitations of Temporary Tables</i>	240

34	TABLE-VALUED FUNCTIONS	Page
34.1	The Two Types of Table-Valued Functions	241
	<i>Types of Table-Valued Functions</i>	241
	<i>Where to Find Them</i>	241
34.2	In-line Table-Valued Functions	242
	<i>Syntax of In-Line TVFs</i>	242
	<i>Where Stored Procedures Fall Short</i>	242
	<i>The In-Line TVF Solution</i>	243
	<i>Joins with Table-Valued Functions</i>	243
34.3	Multi-Statement Table-Valued Functions	244
	<i>Syntax of an MSTVF</i>	244
	<i>Example of an MSTVF</i>	245

## TABLE OF CONTENTS (9 of 9)

35	DYNAMIC SQL	Page
35.1	The EXEC Command and Dynamic SQL	246
	<i>Why not to Use Dynamic SQL</i>	246
35.2	Example –Parameterising Row Selection	247

36	PIVOTING DATA	Page
36.1	Overview	248
36.2	The Two Stages of Creating a Pivot Query	249
	<i>Step 1 – Assembling the Data</i>	249
	<i>Step 2 – Pivoting the Assembled Data</i>	250
36.3	Varying the Number of Row Fields	251
	<i>Pivot Queries with no Row Headings</i>	251
	<i>Pivot Queries with Multiple Row Headings</i>	252
36.4	Queries Based on Pivot Queries	253
36.5	Getting and Using Dynamic Columns	254
	<i>Step 1 – Get a Comma-Delimited List</i>	254
	<i>Step 2 – Build up the SQL Statement</i>	255
	<i>Step 3 – Test the SQL</i>	255
	<i>Step 4 – Execute the SQL</i>	255

37	TRIGGERS	Page
37.1	Overview of Triggers	256
	<i>Syntax of a Trigger</i>	256
37.2	Working with Triggers	257
	<i>Creating a Trigger</i>	257
	<i>Viewing Triggers</i>	257
	<i>Enabling and Disabling Triggers</i>	258
	<i>Deleting Triggers</i>	258
37.3	More Sophisticated Triggers	259
	<i>Tables Created by Triggers</i>	259
37.4	A Case Study: Transactions in Triggers	260

## CHAPTER 1 - DESIGNING DATABASES

The world runs on relational databases. If you understand the principles upon which these are built, you'll find it much easier to write SQL to get information out of them!



*This manual gives an overview only of database design principles. If you want to delve deeper, try Googling phrases like **Third Normal Form**, **Database Normalisation** or **Entity Diagram**. If nothing else, this will give you an impressive search history in your browser!*

### 1.1 The Four Stages of Database Design

There are four stages to designing a relational database, shown below (using the example of creating a simple database to hold films; or movies, if you must).

#### Stage 1 – Deciding what to Include

A good way to do this is to create a spreadsheet of the data you want to include for each film:

Title	Oscars	Director	Date of birth	Studio
Armageddon	0	Michael Bay	17/02/1965	Touchstone Pictures
Bad Boys	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Bad Boys II	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Dead Poets Society	1	Peter Weir	21/08/1944	Touchstone Pictures
Master and Commander ...	2	Peter Weir	21/08/1944	20th Century Fox
Pearl Harbor	1	Michael Bay	17/02/1965	Touchstone Pictures
The Rock	0	Michael Bay	17/02/1965	Hollywood Pictures
The Truman Show	0	Peter Weir	21/08/1944	Scott Rudin Productions

We want to assign each film to a director, but we don't want to have to type each director's name in over and over again!

The aim of designing a relational database is to ensure that you don't hold information twice:

Title	Oscars	Director
Dead Poets Society	1	Peter Weir
Master and Commander ..	2	Peter Wier
The Truman Show	0	Peter Weird

Not only is holding duplicate information inefficient, but it also means that spelling mistakes will creep in. Here listing out films directed by **Peter Weir** would miss out the last two films, as his name has been misspelt.

## Stage 2 – Dividing Data into Tables

Having decided what data you want to include, the next stage of database design is to decide which table each bit of information belongs to:

Title	Oscars	Director	Date of birth	Studio
Armageddon	0	Michael Bay	17/02/1965	Touchstone Pictures
Bad Boys	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Bad Boys II	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Dead Poets Society	1	Peter Weir	21/08/1944	Touchstone Pictures
Master and Commander ...	2	Peter Weir	21/08/1944	20th Century Fox
Pearl Harbor	1	Michael Bay	17/02/1965	Touchstone Pictures
The Rock	0	Michael Bay	17/02/1965	Hollywood Pictures
The Truman Show	0	Peter Weir	21/08/1944	Scott Rudin Productions

↑

These are all details to do with the film itself.

↑

These are to do with the director (name / birthday).

↑

These are details to do with the studio.



*There's no magic wand to make this easier, other than bitter experience of getting it wrong and having to start again! A good guideline is that if you find yourself typing in something twice, it probably belonged in a different table.*

For our example above, there are clearly 3 separate entities: films, the directors who made them and the studios which produced them. Here are the fields that each table could contain:

Table	Fields
<i>Film</i>	<b>Title</b> and <b>Oscars Won</b> , plus something to identify which director and which studio made it
<i>Director</i>	<b>Director name</b> and <b>Date of birth</b> , plus some unique identifier for the director
<i>Studio</i>	<b>Studio name</b> , plus some unique identifier for the studio

What you need to do next is to decide what form these unique identifiers should take.

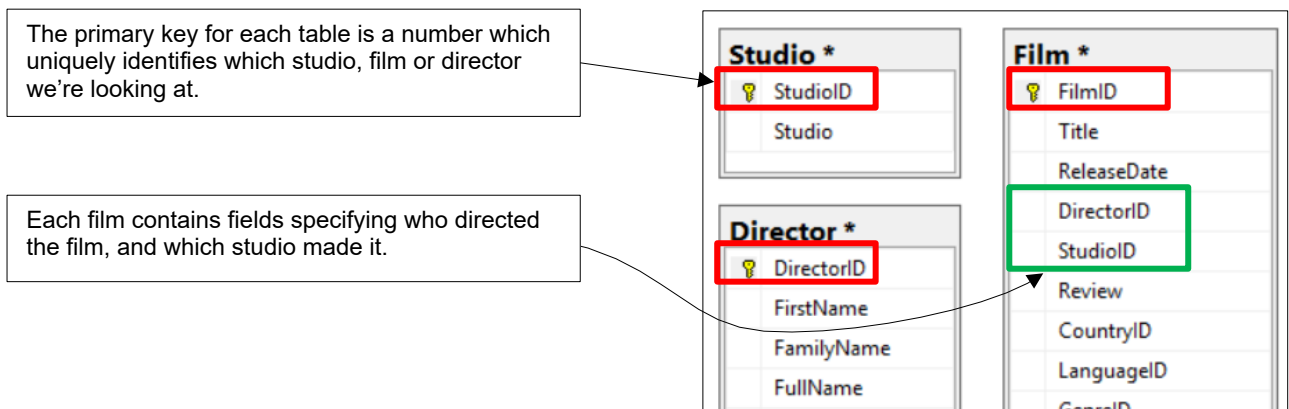
### Stage 3 – Choosing a Primary Key for each Table

The *primary key* for a table is a field which tells you exactly which record you're considering (for example, if you know a film's **DirectorID** you can look up all of the director's other details).



From the above definition, it follows that two records in a table can't have the same value for the primary key field – the field is unique.

For our example, we could use the director and studio names as our primary keys, but SQL Server works most efficiently if the primary key is as short as possible, so we'll create new fields instead:



Here's what **The Sound of Music** would now look like:

FilmID	Title	OscarWins	DirectorID	StudioID
638	The Sound of Music	5	89	4

Including the director's unique number allows us to look up all their other details:

DirectorID	FullName	DoB	Gender
89	Robert Wise	10/09/1914	Male

Including the studio's unique number allows us to look up its name:

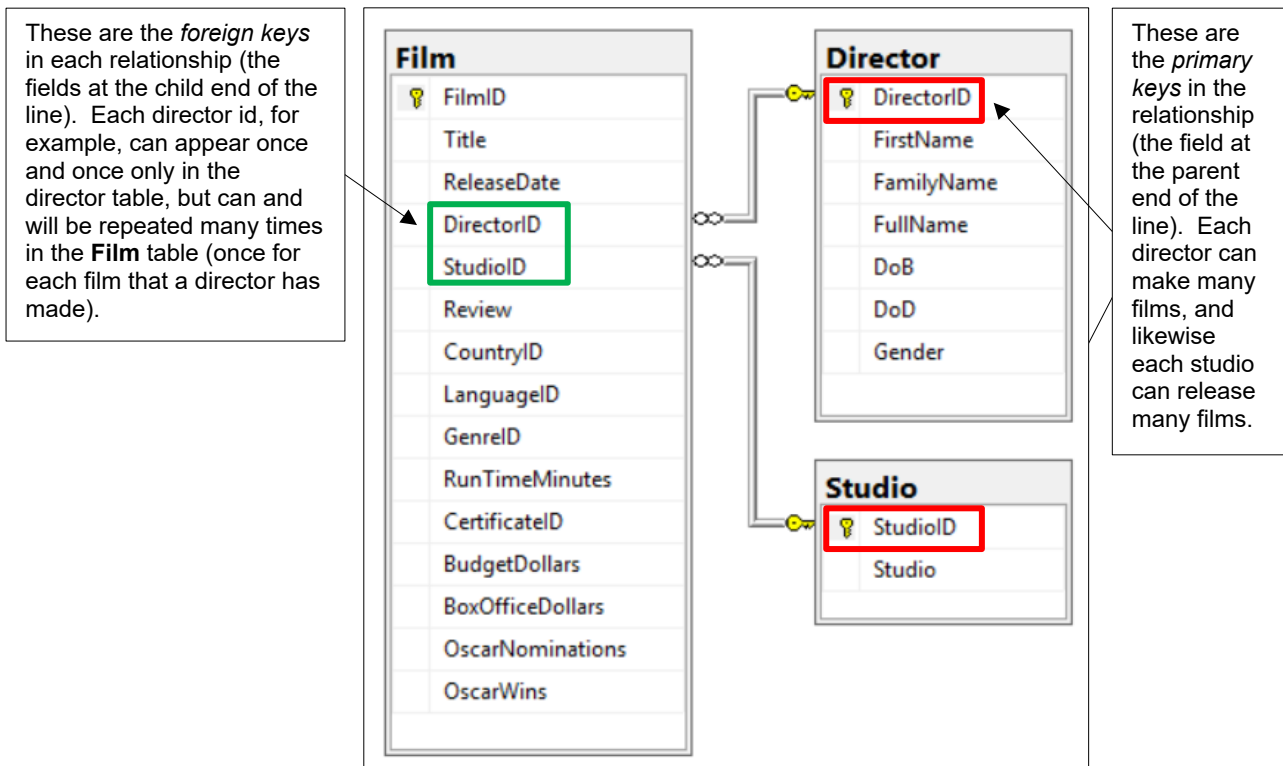
StudioID	Studio
4	20th Century Fox



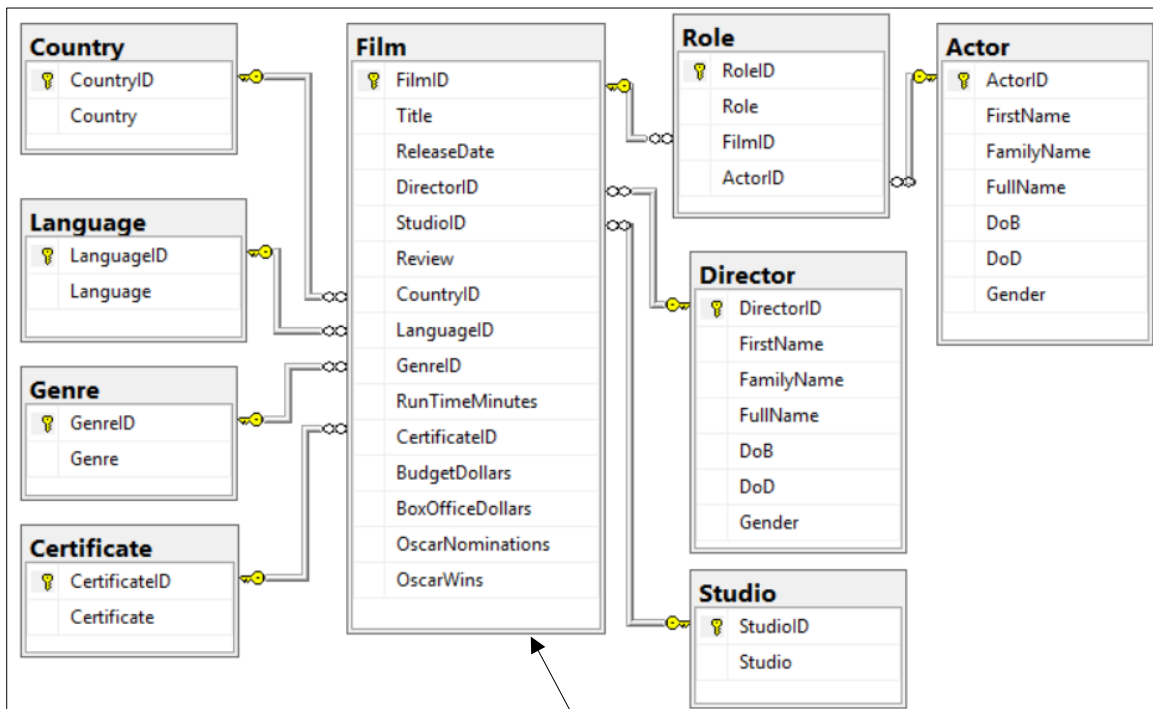
If you're beginning to think that relational databases are just like a lot of spreadsheets joined together with a more efficient version of a **VLOOKUP** formula in Excel, you're absolutely right!

## Stage 4 – Creating Relationships and a Database Diagram

The last step in designing a database is to decide for each relationship that you create whether it is *one-to-many* or *many-to-one* (parent-child or child-parent):




Database diagrams often involve hundreds of tables:



The **Movies** database we'll use in this courseware contains just 9 tables, and hence is untypically simple.

## 1.2 Many-to-Many Relationships


There's no such thing as a many-to-many relationship in SQL Server, but they do exist in real life:



Spider-Man
Jurassic Park
Mission Impossible
Superman Returns
Top Gun
Rain Man
Titanic
Waterworld
Pearl Harbor
Transformers

Sam Neill
Tom Cruise
Laura Dern
Jeff Goldblum
Jon Voight
Vanessa Redgrave
Kirsten Dunst
Naomi Watts
Jack Black
Adrien Brody

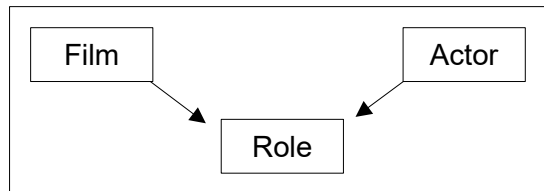
*Tom Cruise has appeared in lots of films, but equally **Mission: Impossible** has lots of actors in it.*



Spider-Man
Jurassic Park
Mission Impossible
Superman Returns
Top Gun
Rain Man
Titanic
Waterworld
Pearl Harbor
Transformers

Sam Neill
Tom Cruise
Laura Dern
Jeff Goldblum
Jon Voight
Vanessa Redgrave
Kirsten Dunst
Naomi Watts
Jack Black
Adrien Brody

The solution to this problem is to create a table that is a child to both of the two parent tables, as here:



Here's what the database would look like:

Film	
FilmID	PK
Title	
ReleaseDate	
DirectorID	
StudioID	
Review	
CountryID	
LanguageID	

Role	
RoleID	PK
Role	
FilmID	FK
ActorID	FK

Actor	
ActorID	PK
FirstName	
FamilyName	
FullName	
DoB	
DoD	
Gender	

The **Role** table links the **Film** and **Actor** tables. Each film can contain many roles (otherwise a film could only have a single actor), but likewise each actor can have many roles (otherwise they would never work again after completing their first film).

Here are 3 rows from the **Role** table:

RoleID	Role	FilmID	ActorID
1	Ray Ferrier	33	1
2	Dr. Alan Grant	1	2
3	Dr. Ellie Sattler	1	3
202	Nathan Algren	41	1

Here are the films and actors who are represented by these rows of data (the duplicate film name was **Jurassic Park**, and duplicated actor turns out to be **Tom Cruise**).

Film number 1 appears twice in this list, as does actor number 1.

RoleID	Role	FilmID	ActorID	Title	FullName
1	Ray Ferrier	33	1	War of the Worlds	Tom Cruise
2	Dr. Alan Grant	1	2	Jurassic Park	Sam Neill
3	Dr. Ellie Sattler	1	3	Jurassic Park	Laura Dern
202	Nathan Algren	41	1	The Last Samurai	Tom Cruise

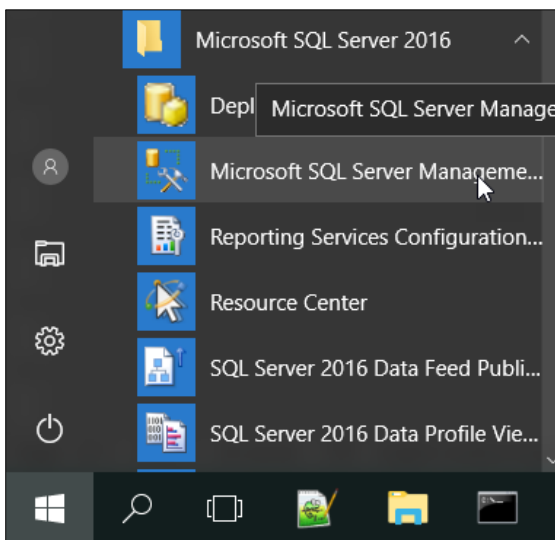


## CHAPTER 2 - SQL SERVER MANAGEMENT STUDIO

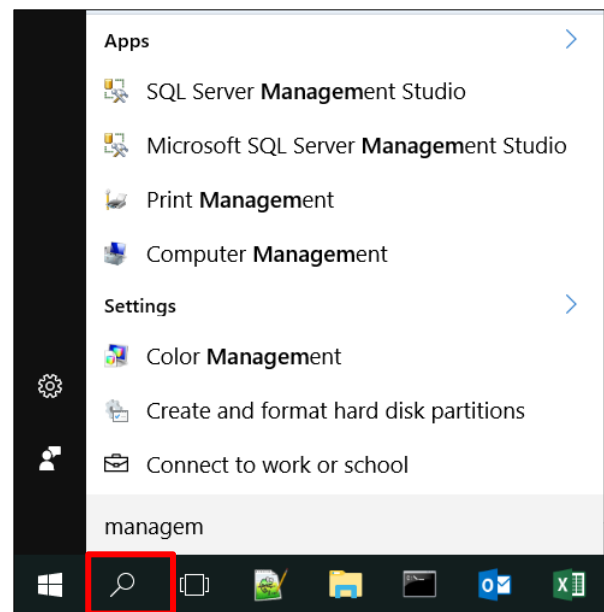
If you're writing SQL to get information out of a database created using SQL Server, the chances are that you'll use *SSMS (SQL Server Management Studio)* as your authoring tool.

### 2.1 Starting to Use Management Studio

You can start SSMS like any other application – here are a couple of ways using Windows 10:



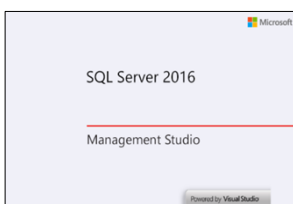
Click on the Windows icon, and choose the program that you want to run ...



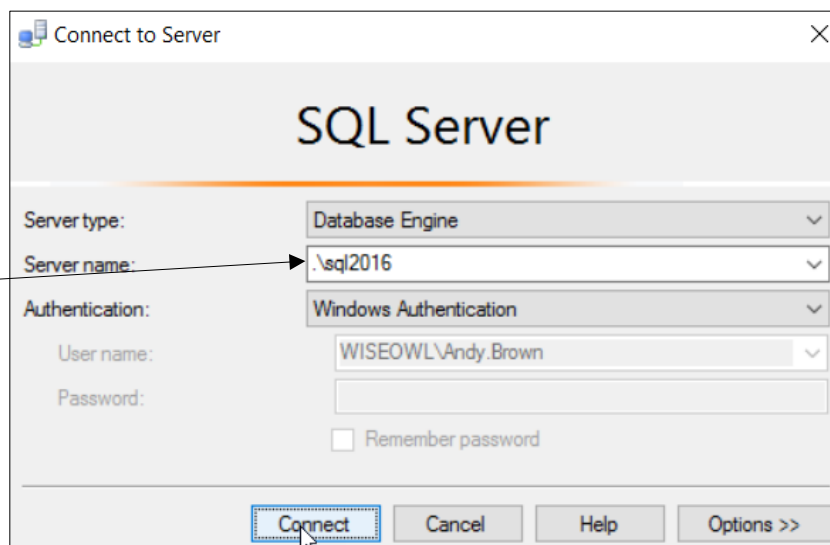
... or click on this tool and type in part of the program name (here typing **managem** has been enough to bring up the program name in the list).

You can then choose a database to use (or *connect to*):

You will see the SQL Server logo on screen while it loads:



You can then choose from the dropdown list which of your company's servers you want to connect to (your IT people should be able to advise on which to choose). If you use Windows authentication, you won't have to type in any more user names or passwords.



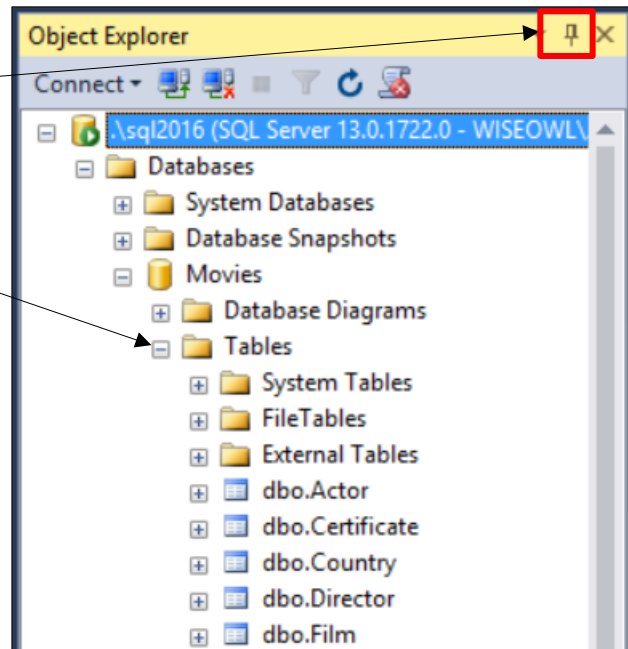
## 2.2 Object Explorer

When Management Studio loads, you should see the **Object Explorer** window (if it's not visible, press **F8** to show it):

If you think Object Explorer is taking up too much room, click on this pin (it will go from vertical to horizontal, and the window will collapse until needed). Click again on the pin to make the window permanently visible again.

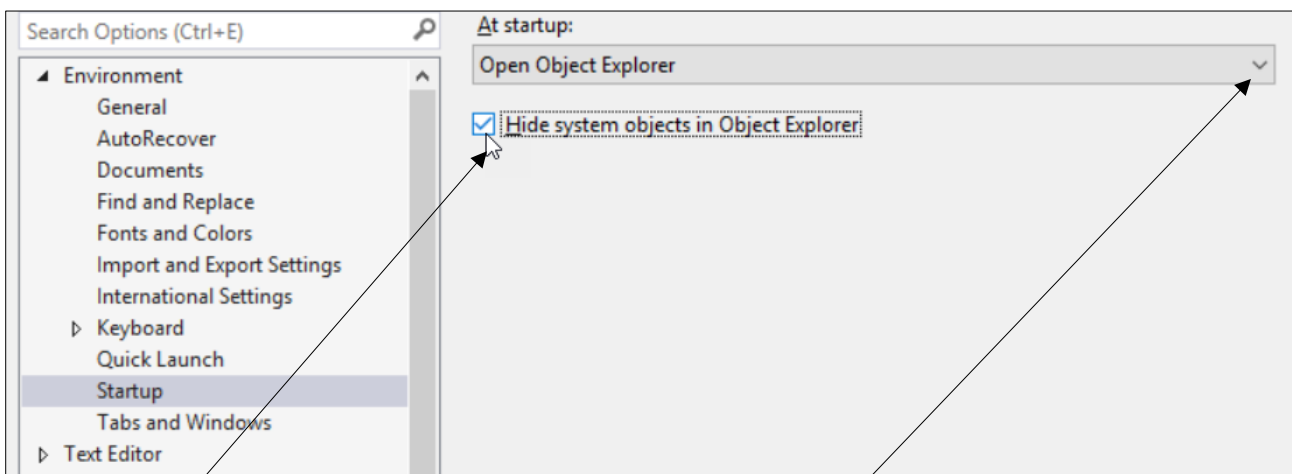
You can click on the **+** symbol to expand:

- **Databases**, to see the **Movies** database; then
- **Movies**, to see what it contains; then
- **Tables**, to see what tables there are.



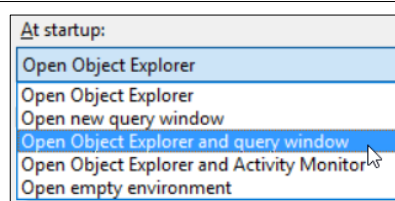
### Useful Start-up Options

You can control what happens when you start Management Studio. To do this, from the menu select **Tools** → **Options**, then complete the dialog box which appears as follows:







a) System objects clutter up SQL Server, and (in this owl's opinion) are best hidden, although you won't see a huge amount of difference.

b) You can click on the drop arrow and choose (for example) to show a blank query as well as Object Explorer whenever you open Management Studio.



## WHAT WE DO

	 <b>ONLINE TRAINING</b>	 <b>MANCHESTER OR LONDON</b>	 <b>AT YOUR OFFICE</b>	 <b>BESPOKE CONSULTANCY</b>	
<b>OFFICE 365</b>	Microsoft Excel	✓	✓	✓	✓
	VBA macros	✓	✓	✓	✓
	Office Scripts	✓		✓	
	Microsoft Access				✓
<b>POWER PLATFORM</b>	Power BI and DAX	✓	✓	✓	✓
	Power Apps	✓		✓	
	Power Automate	✓	✓	✓	✓
<b>SQL SERVER</b>	Reporting Services	✓	✓	✓	✓
	Report Builder	✓		✓	✓
	Integration Services	✓	✓	✓	✓
	Analysis Services	✓		✓	
<b>CODING LANGUAGES</b>	SQL	✓	✓	✓	✓
	Visual C#	✓	✓	✓	✓
	Python	✓	✓	✓	✓



**WiseOwl**  
Training

