Lightning Guide to MS Access & SQL

A practical guide to Database Design
with MS Access and SQL

Prof. Arturo Azcorra

Lightning Guide to Databases with Microsoft Access and SQL

© Prof. Dr. Arturo Azcorra

August 2021 – First edition v1.0

Buy me at: lightningguide.net

PARTS IN THIS LIGHTNING GUIDE

PART A. CREATING MY FIRST DATABASE WITH MS-ACCESS 1
PART B. BRIEFING ON MS-ACCESS USER INTERFACE.................................... 24
PART C. CONCEPTS AND INTERNALS OF DATABASES 92
PART D. DESIGNING MY DATABASES WITH MS-ACCESS............................ 133
PART E. ENTERING, MODIFYING AND DELETING MY DATABASE DATA . 202
PART F. WRITING SQL QUERIES TO USE MY DATABASE 231
PART G. WRITING EXPRESSIONS 398
PART H. CUSTOMIZING THE APPEARANCE OF A QUERY/TABLE/FORM IN “DATASHEET VIEW”.. 439
PART I. EVOLVING MY DATABASE DESIGN... 469
PART J. DEBUGGING MY SQL QUERIES.. 495
PART K. USEFUL DESIGN ADVICE 555
PART L. FIXING DATABASE ERRORS 669
PART M. LIST OF BUILT-IN FUNCTIONS 720
PART N. CONTENTS AND ACKNOWLEDGEMENTS I

Copyright warning!

This book is protected by international copyright laws, and you are not authorized to distribute it to any other person. ISBN: 978-84-09-33305-9

This file has been specifically produced for you and may be watermarked with a SHA-256 hash of your personal data, as provided to the purchase agent company. In case a copy of this file with your personal data appears in public servers, sharing file networks, or is found in the possession of any person other than you, we will take legal action against you. Legal actions will be taken in the courts of your country and in the courts of the state of California.

If you want to recommend this book to a friend/colleague, just send him/her the link:

https://lightningguide.net

PART N. CONTENTS AND ACKNOWLEDGEMENTS

Table of Contents

PART A. CREATING MY FIRST DATABASE WITH MS-ACCESS 1
A.1 How do I use this Lightning Guide?.. 1
A.2 What version of MS-Access is this Guide for? 2
A.3 How do I create my first database?.. 2
A.3.1 What is the MS-Access user interface? 2
A.3.2 How do I create my first database file? 2
A.3.3 How do I create my first Table? 3
A.3.4 How do I input my first data into my Table?.. 5
A.4 How do I write and run my first SQL Query? 5
A.4.1 How do I write my first SQL Query? 6
A.4.2 How do I run my first SQL Query? 7
A.4.3 What does it mean “case insensitive”? 7
A.5 How do I add a Table to my first database? 8
A.5.1 How do I add the Table “T_Capital_Rainfall_Q” to my first database?.......... 8
A.5.2 How do I input data into the Table “T_Capital_Rainfall_Q”?......................... 8
A.6 How do I write and run my first Select Query with record aggregation?............. 9
A.6.1 How do I write my first Select Query with record aggregation?.................... 9
A.6.2 How do I run my first Select Query with record aggregation?..................... 10
A.7 How do I configure my Tables in my first database?... 11
A.7.1 How do I configure my Table “T_Capital_Cities”? 11
A.7.1.1 How do I configure the Primary Key field in Table “T_Capital_Cities”? 11
A.7.1.2 How do I configure no zero-length in all my text fields?................................. 12
A.7.1.3 How do I check that the “Required” property really works? 12
A.7.1.4 How do I check that the Key field really works? 13
A.7.2 How do I configure my Table “T_Capital_Rainfall_Q”? 13
A.7.2.1 How do I configure the Primary Key fields in Table “T_Capital_Rainfall_Q”? 14
A.7.2.2 How do I configure “Allow Zero Length=No” in all my text fields? 15
A.7.2.3 How do I configure a field validation rule in my “Quart” field? 15
A.7.2.4 How do I check that the composite Key fields really work?............................. 15
A.7.2.5 How do I check that the field validation rule for the field “Quart” really works? 16
A.8 How do I write and run my first Union Query?.. 16
A.8.1 How do I write my first Union Query?.. 16

A.8.2 How do I run my first Union Query? 18
A.9 How do I create a Relationship in my first database? 18
A.9.1 How do I create a Relationship from “T_Capital_Cities” to “T_Capital_Rainfall_Q”? 18
A.9.2 How do I check that the Relationship really works? 19
A.10 How do I write and run my first Join Query? 20
A.10.1 How do I write my first Join Query? 20
A.10.2 How do I run my first Join Query? 21
A.11 How do I write and run my first Transform Query? 21
A.11.1 How do I write my first Transform Query? 21
A.11.2 How do I run my first Transform Query? 23
PART B. BRIEFING ON MS-ACCESS USER INTERFACE 24
B.1 What options should I set in MS-Access? 24
B.1.1 What “Current Database” options should I set? 25
B.1.2 What “Object Designers” options should I set? 26
B.1.3 What “Client Settings” options should I set? 26
B.1.4 What “Quick Access Toolbar” options should I set? 26
B.2 How is the MS-Access window structured? 27
B.2.1 What is the top window frame? 28
B.2.2 What is the “Quick Access Toolbar”? 28
B.2.3 What is the “Ribbon-bar”? 28
B.2.4 What are Ribbon names? 28
B.2.5 What is the Ribbon Area? 29
B.2.6 What is the visible Ribbon? 29
B.2.7 What is the “Navigation Pane”? 29
B.2.8 What is the Object Area? 30
B.2.9 What are the object tabs? 30
B.2.10 What is the Visible Object and the object panes? 30
B.2.11 What is the “Status-bar”? 31
B.3 What are the Ribbons? 31
B.3.1 What is a Ribbon group? 31
B.3.2 Why do Ribbons change how they look? 32
B.3.3 What permanent Ribbon can I display? 33
B.3.4 What contextual Ribbon can I display? 34
B.3.4.1 What are contextual Ribbons? 34
B.3.4.2 What are contextual Ribbon labels? 34

B.3.4.3 What are contextual Ribbon names? 34
B.3.4.4 What contextual Ribbon can I display, depending on context? 34
B.4 What is the "Navigation Pane" and how it works? 36
B.4.1 How do I open, close and do other actions on objects in the "Navigation Pane"? 38
B.4.1.1 How do I search for an object name in the "Navigation Pane"? 38
B.4.1.2 What is the object menu in the "Navigation Pane"? 38
B.4.1.3 How do I open an object with the "Navigation Pane"? 41
B.4.1.4 How do I change the view-type of an opened object? 41
B.4.1.5 How do I view another of the opened objects? 43
B.4.1.6 How do I save the layout/design of an opened object? 43
B.4.1.7 How do I close an opened object? 43
B.4.1.8 How do I rename an object? 44
B.4.1.9 How do I run a Query? 44
B.4.1.10 How do I select one or more objects? 45
B.4.1.11 How do I edit a Query? 45
B.4.1.12 How do I create an object? 45
B.4.1.13 How do I delete objects? 46
B.4.1.14 How do I copy/cut and paste objects? 46
B.4.2 How do I configure the way object names are grouped, ordered, displayed or hidden in the "Navigation Pane"? 46
B.4.2.1 How do I configure the way object names are grouped in the "Navigation Pane"? 47
B.4.2.2 How do I configure the way object names are ordered in the "Navigation Pane"? 48
B.4.2.3 How do I configure the way object names are displayed in the "Navigation Pane"? 49
B.4.2.4 How do I hide/unhide whole object name groups in the "Navigation Pane"? 49
B.4.2.5 How do I hide/unhide an individual object name in its object group? 50
B.4.2.6 How do I hide/unhide all system object names? 51
B.5 What is a Table/Query/Form in "Datasheet View"? 51
B.5.1 What is the "Navigation bar" in "Datasheet View"? 53
B.5.2 How do I select a range of fields/records in "Datasheet View"? 53
B.5.3 How do I select one field's value in "Datasheet View"? 57
B.5.4 What are the differences between selecting one field and one field's value in "Datasheet View"? 59
B.5.5 How do I find a record in a Table/Query/Form "Datasheet View" 60

B.5.6 How do I manage a Table/Query/Form “Property Sheet” in “Datasheet View”? 60
B.6 What is a Table in “Design View”? 60
B.6.1 How do I view, add, delete and configure my Table fields? 62
B.6.1.1 How do I view the Table fields in “Design View”? 62
B.6.1.2 What are a Field’s “General” tab properties in “Design View”? 63
B.6.1.3 What are a Field’s “Lookup” tab properties in “Design View”? 63
B.6.1.4 How do I select one or more Table fields in “Design View”? 63
B.6.1.5 How do I add Table fields in “Design View”? 64
B.6.1.6 How do I copy/cut and paste Table fields in “Design View”? 65
B.6.1.7 How do I reconfigure my Table fields in “Design View”? 66
B.6.1.8 How do I delete Table fields in “Design View”? 66
B.6.1.9 How do I reorder Table fields in “Design View”? 67
B.6.2 How do I manage Table indexes in “Design View”? 67
B.6.2.1 How do I view Table indexes in “Design View”? 67
B.6.2.2 How do I select fields in Table indexes in “Design View”? 69
B.6.2.3 How do I delete, insert or move fields in Table indexes in “Design View”? 70
B.6.3 How do I unhide/hide a Table “Property Sheet” in “Design View”? 70
B.6.4 How do I configure a Table “Property Sheet” in “Design View”? 71
B.7 What is a Query in “Design View”? 72
B.7.1 How do I unhide/hide a Query’s “Property Sheet” in “Design View”? 73
B.7.2 How do I configure a Query’s “Property Sheet” in “Design View”? 74
B.8 What is a Form in “Design View”? 76
B.8.1 How do I unhide/hide a Form’s “Property Sheet” in “Datasheet View” or “Design View”? 76
B.8.2 How do I configure a Form’s “Property Sheet” in “Datasheet View” or “Design View”? 77
B.9 What is a Query in “SQL View”? 79
B.10 What is the “Relationships” pane? 80
B.10.1 How do I open the “Relationships” pane? 80
B.10.2 How do I view my Relationships? 80
B.10.3 How do I view a Relationship’s properties? 82
B.10.4 How do I change the layout of the “Relationships” pane? 85
B.10.4.1 How do I select/unselect Table-boxes? 85
B.10.4.2 How do I unhide/hide Relationships? 86
B.10.4.3 How do I unhide/hide the “Add Tables” sub-pane? 86

B.10.4.4 How do I unhide or add Table-boxes? 87
B.10.4.5 How do I hide or delete a Table-box? 87
B.10.4.6 What is the difference between unhide/hide and add/delete a Table-box? 88
B.10.4.7 How do I move a Table-box? 88
B.10.4.8 How do I resize a Table-box? 89
B.10.4.9 How do I front-display a Table-box? 89
B.10.4.10 How do I save the “Relationships” pane layout? 89
B.10.5 How do I edit an existing Relationship? 89
B.10.6 How do I create a new Relationship? 90
B.10.7 How do I delete a Relationship? 90
B.10.8 How do I close the “Relationships” pane? 90
B.11 Can I use a drop-down/expression menu even if its icon is not shown? 91
PART C. CONCEPTS AND INTERNALS OF DATABASES 92
C.1 What are the main concepts of databases? 92
C.1.1 What is a database, a record and a field? 92
C.1.2 What is the difference between a database and a spreadsheet? 93
C.1.3 What is a database design? 94
C.1.4 What is an SQL Query? 94
C.1.5 What is the Structured Query Language (SQL) 94
C.1.6 How is a database used? 94
C.2 What are objects, names, keywords, data types, constants, variables, operators, functions and expressions? 95
C.2.1 What is an object and a name? 95
C.2.2 What is a qualified field name? 95
C.2.3 What is a keyword? 96
C.2.4 What is a data type? 96
C.2.5 What is a constant? 96
C.2.6 What is a variable? 97
C.2.7 What is an operator? 97
C.2.8 What is a function? 98
C.2.9 What is an expression? 99
C.2.10 What is an SQL operation? 99
C.2.11 What is expression syntax? 100
C.3 What are fields, field value-lists, records and record-lists? 100
C.3.1 What are fields, field names and field types? 101
C.3.2 What is a record and a record type? 101

C.3.3 What is a record-list? 101
C.3.4 What is a field value-list? 102
C.4 What are database Tables? 103
C.5 What is a pointer? 103
C.6 What is a Null? 104
C.7 What are duplicate records and duplicate field values? 105
C.7.1 What are duplicate records? 105
C.7.2 What are duplicate field values? 106
C.8 What is indexing? 107
C.8.1 Why is indexing useful? 107
C.8.2 Can I create one index over a list of field names (called a composite index)? 110
C.8.3 What different types of indexes are there? 112
C.8.3.1 What are simple indexes and composite indexes? 112
C.8.3.2 What are indexes with duplicate values and without duplicate values? 112
C.8.3.3 What are indexes that ignore Nulls and not ignore Nulls? 113
C.8.3.4 What are indexes with Nulls and without Nulls? 113
C.8.3.5 What is an index without duplicate values and without Nulls? 113
C.8.4 What indexes can I configure in a given Table? 114
C.8.5 Why should I use indexing in my database? 116
C.8.5.1 Why indexing improves performance? 116
C.8.5.2 Why indexing prevents duplicate records in a Table? 116
C.8.5.3 Why indexing allows establishing Relationships between Tables? 117
C.9 How do I prevent duplicate field values and duplicate records? 117
C.9.1 How I do I prevent duplicate field values over a group of field name(s) in my Tables? 117
C.9.2 How I do I prevent duplicate records in my Tables? 118
C.9.3 How do I prevent duplicate records in my Query's record-lists? 119
C.10 What are the Table Key(s) and how should I handle them? 120
C.10.1 What is the Primary Key of a Table? 120
C.10.2 What is a simple Key and a composite Key? 121
C.10.3 What are the candidate Keys of a Table? 122
C.11 What is a Relationship? 122
C.11.1 What is a Relationship with referential integrity? 125
C.11.2 What are "one-to-many" and "one-to-one" Relationships? 127
C.11.3 What is an indeterminate Relationship? 130
C.11.4 What is a many-to-many Relationship? 130

C.11.5 Why should I configure Relationships? 132
PART D. DESIGNING MY DATABASES WITH MS-ACCESS 133
How do I design my database? 133
How do I use my database? 134
How do I evolve my database design? 135
D.1 How do I create, close and open a database file? 135
D.1.1 How do I create a database file? 135
D.1.2 How do I close a database file? 136
D.1.3 How do I open an existing database file? 136
D.2 How do I carefully assign good names from the very beginning? 137
D.2.1 Why should I make field names short but clear? 138
D.2.2 Why should I use homogeneous field names across Tables and Queries? . 139
D.2.3 Why should I use structured names for Tables and Queries? 139
D.2.4 Why should I avoid certain elements in names? 140
D.2.4.1 Why should I avoid special characters in names? 140
D.2.4.2 Why should I avoid "<>" as a field name? 141
D.2.4.3 Why should I avoid non-English letters in names? 141
D.2.4.4 Why must I avoid control characters in names? 141
D.2.4.5 Why must I avoid using keywords as names? 142
D.2.4.6 Why should I avoid the suffix "_n" in Table names? 142
D.2.5 What are the MS-Access formal rules for identifiers? 143
D.2.6 When can I assign the same name to different objects and/or properties? . 144
D.3 How do I create and design a Table and its fields? 145
D.3.1 How do I create a Table? 145
D.3.2 How do I create my Table fields? 146
D.4 How do I configure a Table field data type and size? 146
D.4.1 What is the "Short Text" field type? 147
D.4.2 What is the "Long Text" field type? 148
D.4.3 What are the "Number" field types? 148
D.4.3.1 What is the "Number-Byte" field type-size? 148
D.4.3.2 What is the "Number-Integer" field type-size? 149
D.4.3.3 What is the "Number-Long Integer" field type-size? 149
D.4.3.4 What is the "Number-Single" field type-size? 149
D.4.3.5 What is the "Number-Double" field type-size? 150
D.4.3.6 What is the "Number-Replication ID" field type-size? 150
D.4.3.7 What is the "Number-Decimal" field type-size? 150

D.4.4 What is the “Currency” field type? 150
D.4.5 What is the “Date/Time” field type? 151
D.4.6 What is the “Date/Time extended” field type? 152
D.4.7 What is the “Yes/No” field type? 153
D.4.8 What is the “Calculated” field type? 153
D.4.9 What is the “Large Number” field type? 154
D.4.10 What is the “AutoNumber” field type? 155
D.4.11 What is the “OLE Object” field type? 155
D.4.12 What is the “Hyperlink” field type? 156
D.4.13 What is the “Attachment” field type? 156
D.4.14 What is the “Lookup Wizard...”? 157
D.5 How do I configure a Table field validation rule, indexing, and other properties? 157
D.5.1 What common Table field properties should I set? 158
D.5.1.1 What is the “Description” Table field property? 158
D.5.1.2 What is the “Format” Table field property? 158
D.5.1.3 What is the “Caption” Table field property? 159
D.5.1.4 What is the “Default Value” Table field property? 159
D.5.1.5 What is the field “Validation Rule” Table field property? 159
D.5.1.6 What is the field “Validation Text” Table field property? 161
D.5.1.7 What is the “Required” Table field property? 161
D.5.1.8 What is the “Indexed” Table field property? 162
D.5.1.9 What is the “Text Align” Table field property? 162
D.5.2 What data-type specific Table field properties should I set? 162
D.5.2.1 What is the “Field Size” Table field property? 163
D.5.2.2 What is the “Allow Zero Length” Table field property? 163
D.5.2.3 What is the “Decimal Places” Table field property? 163
D.5.2.4 What is the “Input Mask” Table field property? 163
D.5.2.5 What is the “Result Type” Table field property? 164
D.6 How do I configure the Primary Key field(s) of a Table? 164
D.6.1 How do I configure a Primary Key with only one field? 164
D.6.2 How do I configure a Primary Key with several fields? 165
D.6.3 How do I change the Primary Key of a Table? 165
D.6.4 What checks are done on Key fields when saving my Table design? 166
D.6.5 How do I identify the Primary Key field(s) of a Table? 166
D.6.6 Why should I define the Primary Key field(s) in Tables? 168

D.7 How do I add simple and/or composite index(es) to a Table? 168
D.7.1 How do I add simple indexes to a Table?.. 169
D.7.2 How do I add composite (and simple) indexes to a Table?........................ 169
D.7.3 How do I reconfigure the indexes of a Table?... 170
D.8 How do I configure the properties of a Table?... 171
D.8.1 How do I configure a record validation rule?.. 172
D.8.2 How do I configure the record validation text?.. 173
D.8.3 How do I configure the “Subdatasheet Name” property? 173
D.9 How do I create and configure my Table Relationships?................................. 174
D.9.1 How do I create a new Relationship? 175
D.9.1.1 How do I create a new Relationship in the most frequent case?..................... 175
D.9.1.2 How do I create a new Relationship in less frequent cases? 176
D.9.2 How do I configure a Relationship?.. 177
D.9.3 What is the effect of “Enforce Referential Integrity”? 180
D.9.4 What is the effect of “Cascade Update Related Fields”?............................ 180
D.9.5 What is the effect of “Cascade Delete Related Records”? 181
D.9.6 What Relationships should I configure?.. 181
D.10 How do I design MS-Access Forms? 182
D.10.1 What are MS-Access Forms? 182
D.10.2 How do I create an MS-Access Form?.. 182
D.10.3 How do I add/delete/update a field of a Form? 183
D.10.3.1 How do I add a field to a Form?.. 183
D.10.3.2 How do I delete a field from a Form? 183
D.10.3.3 How do I update a field of a Form?... 184
D.10.4 How do I configure VBA action code associated to Form Events? 184
D.10.4.1 How do I configure a Form to have an associated VBA Module?................ 184
D.10.4.2 How do I enter/edit the VBA subroutine associated to a Form Event?.......... 185
D.10.4.3 How does my VBA code modify field values in the Form record being edited? 186
D.10.4.4 Why should I write options in the VBA Modules of my Forms?.................. 187
D.10.4.5 How do I close the VBA editor? 187
D.11 How do I configure the way to enter data (e.g., a drop-down menu) in a Table/Form field? 188
D.11.1 How do I configure a drop-down menu to enter data in a Table field?.... 189
D.11.1.1 How do I configure a Table drop-down menu that takes its values from a Table/Query?.. 190

D.11.1.2 How do I configure a Table drop-down menu that takes its values from a Value List? ... 191
D.11.1.3 How do I configure a Table drop-down menu that takes its values from a Field List? ... 192
D.11.2 How do I configure a date-picker to enter a date in a Table field? ... 193
D.11.3 How do I configure type-in or checkbox to enter data in a Table/Form field? ... 193
D.11.4 How do I configure a drop-down menu or a date-picker to enter data in a Form field? ... 194
D.11.4.1 How do I configure a new drop-down menu directly in a Form field? ... 194
D.11.4.2 How do I configure an existing drop-down menu in a Form field? ... 196
D.11.4.3 How do I configure a new date-picker directly in a Form field? ... 197
D.11.5 What options can I set in a “Combo Box” Table/Form drop-down menu? ... 198
D.11.5.1 What are useful options in a “Combo Box” Table/Form drop-down menu? ... 198
D.11.5.2 What are the interactions between slave fields, “Limit to List” property and the field validation rule? ... 200
D.12 How do I use MS-Access Reports? ... 200
D.13 How do I share a database, having multiple concurrent users? ... 201
PART E. ENTERING, MODIFYING AND DELETING MY DATABASE DATA . 202
E.1 How do I edit one new or existing record? ... 203
E.1.1 How do I go to the new-record row? ... 204
E.1.2 How do I enter the record under edition? ... 205
E.2 How do I edit the field’s values of the record under edition? ... 206
E.2.1 How do I choose a field value? ... 207
E.2.2 How do I type-in a value in a field? ... 208
E.2.2.1 How do I type-in a value in a Yes/No field? ... 209
E.2.2.2 How do I type-in a value in a Short Text field? ... 209
E.2.2.3 How do I type-in a value in a Number, Currency or Large Number field? ... 209
E.2.2.4 How do I type-in a value in a Date/Time field? ... 209
E.2.2.5 How do I type-in a zero-time value in a Date/Time field? ... 210
E.2.2.6 How do I type-in a zero-date value in a Date/Time field? ... 212
E.2.2.7 What happens if the field “Format” property does not match the field type? ... 212
E.2.3 Why is a value shown in a different way when the Table/Query/Form field’s value has been selected? ... 213
E.2.3.1 How is the formatting of a Number or Currency field changed when I select its field’s value? ... 213

E.2.3.2 How is the formatting of a Yes/No field changed when I select its field's value? 214
E.2.3.3 How is the formatting of a Date/Time field changed when I select its field's value? 214
E.3 How do I interactively delete existing records? 215
E.4 What is different about entering, modifying or deleting records from a Table or a Form? 215
E.5 How do I copy/cut and paste data between MS-Access and other applications? 216
E.5.1 How do I copy/cut data from an MS-Access Table, Form or Query result? 216
E.5.2 How do I paste data into MS-Access? 217
E.5.2.1 How do I paste into a field's value? 217
E.5.2.2 How do I paste over a rectangle of fields? 217
E.5.2.3 How do I paste as new records? 218
E.5.2.4 What are the fields and field order when pasting? 219
E.5.3 How do I paste data copied from MS-Access into other applications? 219
E.5.3.1 How do I paste as plain text into Excel? 219
E.5.3.2 How do I paste as formatted data into Excel? 221
E.5.3.3 How do I paste as plain text into Word? 222
E.5.3.4 How do I paste as formatted text into Word? 222
E.5.3.5 What are the additional rows when pasting into Excel or Word? 223
E.5.3.6 How do I paste into a plain-text processor? 223
E.5.3.7 How do I paste back into MS-Access? 223
E.6 What checks are done when saving a field value, or entering or modifying a record? 223
E.6.1 What checks are done when saving a field value? 223
E.6.2 What checks are done when entering or modifying a record? 224
E.7 How do I bulk-change my Table/Form's data? 225
E.7.1 How do I bulk-modify my data using the "Find/Replace" tool? 225
E.7.2 How do I bulk-change my data with an external application? 227
E.7.3 How do I bulk-change my data using data-changing Queries? 227
E.7.4 Why should I backup my data before a bulk-change? 228
E.8 How do I upload my pre-existing data into my database? 228
E.9 Can I get inconsistent results out of my initial data in my database? 229
E.10 Why should I use "Compact and Repair Database"? 229
PART F. WRITING SQL QUERIES TO USE MY DATABASE 231
F.1 What is the Structured Query Language (SQL)? 231

F.2 What version of SQL is this guide for? 232
F.3 Why should I write and run Queries? 232
F.4 What is an SQL operation and an SQL Query? 232
F.4.1 What is an SQL operation? 233
F.4.2 What is an SQL Query? 233
F.4.3 Why should I write my Queries in SQL? 233
F.4.4 What SQL Query editor should I use? 234
F.4.5 How do I create a Query? 234
F.4.6 How do I edit my SQL Queries without a plug-in Query editor? 235
F.5 How do I edit my SQL Queries with the plug-in "Access SQL Editor"? 235
F.5.1 How do I start with the plug-in "Access SQL Editor"? 236
F.5.1.1 How do I get and install the plug-in "Access SQL Editor"? 236
F.5.1.2 How do I open the plug-in "Access SQL Editor"? 236
F.5.1.3 What options should I set in the "Access SQL Editor"? 237
F.5.2 What is the user interface of the plug-in "Access SQL Editor"? 238
F.5.3 What are the toolbar commands in the "Access SQL Editor"? 239
F.5.4 How do I manage the Queries/Tables with the "Access SQL Editor"? 240
F.5.4.1 How do I open a Query/Table from the "Access SQL Editor"? 240
F.5.4.2 How do I select a Query/Table name in the "Access SQL Editor"? 241
F.5.4.3 How do I show a query pane in the "Access SQL Editor"? 242
F.5.4.4 How do I select a query pane in the "Access SQL Editor"? 242
F.5.4.5 How do I run a Query from the "Access SQL Editor"? 243
F.5.4.6 How do I save a Query from the "Access SQL Editor"? 244
F.5.4.7 How do I close a query pane in the "Access SQL Editor"? 244
F.5.4.8 How do I create a new Query from the "Access SQL Editor"? 245
F.5.4.9 How do I rename a Query from the "Access SQL Editor"? 245
F.5.4.10 How do I copy a Query from the "Access SQL Editor"? 246
F.5.4.11 How do I delete a Query from the "Access SQL Editor"? 246
F.5.5 How do I edit a Query in its query pane in the "Access SQL Editor"? 246
F.5.5.1 What are the specific editing functionalities of a query pane from the "Access SQL Editor"? 247
F.5.5.2 What are the conventional editing functionalities of a query pane from the "Access SQL Editor"? 248
F.5.6 How do I configure the layout of the "Access SQL Editor"? 249
F.5.6.1 What are the query panes in the "Access SQL Editor"? 250
F.5.6.2 What is the "Queries" pane in the "Access SQL Editor"? 250

F.5.6.3 What is the “Tables” pane in the “Access SQL Editor”? 251
F.5.6.4 What is the “Status” pane in the “Access SQL Editor”? 252
F.5.6.5 What is the “Results” pane in the “Access SQL Editor”? 252
F.5.6.6 What is a pane-space in the “Access SQL Editor”? 252
F.5.6.7 What is a sub-window in the “Access SQL Editor”? 253
F.5.6.8 What is a self-standing window of the “Access SQL Editor”? 255
F.5.6.9 How do I move an object pane, a sub-window or a self-standing window of the “Access SQL Editor”? 256
F.5.6.10 How do I convert between an object pane, a pane-space, a sub-window, or a self-standing window in the “Access SQL Editor”? 257
F.5.6.11 What can be a good layout for the “Access SQL Editor”? 258
F.6 What are the SQL operators I use to write my Queries? 259
F.6.1 What are the SQL consulting operations? 259
F.6.2 What are the SQL data-changing operators? 261
F.7 What is a Select operation and how do I write it? 261
F.7.1 What is the Select operator? 262
F.7.2 What are the three types of Select? 265
F.7.3 What is the dataflow of a Select? 267
F.7.4 What is the input record-list (“FROM” clause) of a Select? 270
F.7.5 What are the output fields (“SELECT” clause) of a Select? 271
F.7.6 What is the output record-list of a Select? 273
F.7.6.1 What are the output field values of a Select-no_aggreg? 273
F.7.6.2 What are the output field values of a Select-group_by_aggreg? 274
F.7.6.3 What are the output field values of a Select-total_aggreg? 276
F.7.6.4 What are the output records of a Select? 277
F.7.6.5 How many output records does a Select produce? 279
F.7.7 What is the “WHERE” clause of a Select? 280
F.7.8 What is the “DISTINCTROW” clause of a Select? 281
F.7.9 What is the “GROUP BY” clause of a Select-group_by_aggreg? 281
F.7.10 What is the “HAVING” clause of Select-group_by_aggreg or Select-total_aggreg? 282
F.7.11 What is the “DISTINCT” clause of a Select? 283
F.7.12 How do I use “ORDER BY” to order the output records of a Select? 284
F.7.12.1 How are the different data/field types ordered by the “ORDER BY” clause?. 287
F.7.13 What is the “TOP” clause of a Select? 288
F.7.14 How do I write a correct (syntax) Select? 289

F.7.15 How do I write a correct (syntax) Select-no_aggreg? 290
F.7.15.1 What is a syntax-example of a Select-no_aggreg? 291
F.7.15.2 What are the formal rules (syntax) to write a Select-no_aggreg? 292
F.7.16 How do I write a correct (syntax) Select-group_by_aggreg? 295
F.7.16.1 What is a syntax-example of a Select-group_by_aggreg? 296
F.7.16.2 What are the formal rules (syntax) to write a Select-group_by_aggreg? 298
F.7.17 How do I write a correct (syntax) Select-total_aggreg? 303
F.7.17.1 What is a syntax-example of a Select-total_aggreg? 304
F.7.17.2 What are the formal rules (syntax) to write a Select-total_aggreg? 305
F.7.18 What is an SQL aggregate function? 308
F.7.18.1 What is the "`Count(*)`" SQL aggregate function? 310
F.7.18.2 What is the "`Count()`" SQL aggregate function? 310
F.7.18.3 What are the "`First()`" and "`Last()`" SQL aggregate functions? 310
F.7.18.4 What are the "`Min()`" and "`Max()`" SQL aggregate functions? 310
F.7.18.5 What are the "`Sum()`" and "`Avg()`" SQL aggregate functions? 311
F.7.18.6 What are the "`StDev()`", "`StDevP()`", "`Var()`" and "`VarP()`" SQL aggregate functions? 312
F.7.18.7 What is a summary and grouping of aggregate functions? 313
F.8 What is a Join operation and how do I write it? 314
F.8.1 What is "joining" two ordered records? 314
F.8.2 What are the Join operators? 316
F.8.2.1 How do the four Join operations compare? 319
F.8.2.2 Are Join operations commutative? 320
F.8.3 What are the input record-lists of a Join? 322
F.8.4 What are the output fields of a Join? 323
F.8.5 What is the output record-list of a "`,`" Cross-Join? 324
F.8.5.1 What are the output field values of a "`,`" Cross-Join? 324
F.8.5.2 What are the output records of a "`,`" Cross-Join? 324
F.8.5.3 How many output records does a "`,`" Cross-Join produce? 326
F.8.6 What is the output record-list of an "`INNER JOIN`"? 326
F.8.6.1 What are the output field values of an "`INNER JOIN`"? 327
F.8.6.2 What are the output records of an "`INNER JOIN`"? 327
F.8.6.3 How many output records does an "`INNER JOIN`" produce? 332
F.8.7 What is the output record-list of an Outer-Join ("`LEFT JOIN`" or "`RIGHT JOIN`")? 332

F.8.7.1 What are the output field values of an Outer-Join ("LEFT JOIN" or "RIGHT JOIN")? 333
F.8.7.2 What are the output records of an Outer-Join ("LEFT JOIN" or "RIGHT JOIN")? 333
F.8.7.3 How many output records does an Outer-Join ("LEFT JOIN" or "RIGHT JOIN") produce? 336
F.8.8 What is the output record-list of a Full-Outer-Join? 337
F.8.8.1 What are the output field values of a Full-Outer-Join? 337
F.8.8.2 What are the output records of a Full-Outer-Join? 337
F.8.8.3 How many output records does a Full-Outer-Join produce? 341
F.8.9 What is the "ON" clause of "INNER JOIN" and Outer-Join ("LEFT JOIN" and "RIGHT JOIN")? 341
F.8.10 How do I write a correct (syntax) Join? 343
F.8.10.1 What is a syntax-example of a Join? 343
F.8.10.2 What are the formal rules (syntax) to write a Join? 344
F.8.10.3 How do I nest Joins? 346
F.9 What is a Union operation and how do I write it? 347
F.9.1 What are the Union operators? 348
F.9.2 What are the input record-lists of a Union? 349
F.9.3 What are the output fields of a Union? 349
F.9.4 What is the output record-list of a Union? 351
F.9.4.1 What are the output field values of a Union? 351
F.9.4.2 What are the output records of a Union? 351
F.9.4.3 How many output records does a Union produce? 351
F.9.5 How do I write a correct (syntax) Union? 352
F.9.5.1 What is a syntax-example of a Union? 352
F.9.5.2 What are the formal rules (syntax) to write a Union? 353
F.9.5.3 How do I nest Unions? 353
F.9.6 Why do I find misleading the names of the Union operators? 354
F.10 What is a Transform operation and how do I write it? 354
F.10.1 What is the Transform operator? 356
F.10.2 What is the input record-list ("FROM" clause) of a Transform? 359
F.10.3 What are the output fields of a Transform? 359
F.10.3.1 What is the number of output fields of a Transform? 359
F.10.3.2 What are the output field names of a Transform? 360
F.10.3.3 What is the output field order of a Transform? 362
F.10.3.4 What are the output data/field types of a Transform? 363

F.10.4 What is the output record-list of a Transform? 364
F.10.4.1 What are the output field values of a Transform? 364
F.10.4.2 What are the output records of a Transform? 367
F.10.4.3 How many output records does a Transform produce? 368
F.10.5 Can I see an example of a Transform operation? 368
F.10.6 What is the “TRANSFORM” clause of a Transform? 371
F.10.7 What is the “SELECT” clause of a Transform? 371
F.10.8 What is the “ORDER BY” clause of a Transform? 371
F.10.9 What is the “WHERE” clause of a Transform? 371
F.10.10 What is the “GROUP BY” clause of a Transform? 372
F.10.11 What is the “PIVOT” clause of a Transform? 372
F.10.12 What is the “IN” clause of a Transform? 373
F.10.13 How do the clauses from Transform and Select compare? 374
F.10.14 How do I write a correct (syntax) Transform? 374
F.10.14.1 What is a syntax-example of a Transform? 375
F.10.14.2 What are the formal rules (syntax) to write a Transform? 377
F.10.14.3 Can I nest Transform operations? 382
F.11 What are the SQL clauses, their expression’s elements and color codes? 382
F.11.1 Given an SQL clause, what are its expression’s elements? 382
F.11.2 What are the SQL color codes used in this Guide? 385
F.12 How do I add parameters (type-in variables) to my Queries? 386
F.13 How do I write a Query that changes my Table data? 388
F.13.1 What is a Delete operation and how do I write it? 389
F.13.2 What is an Insert operation and how do I write it? 389
F.13.2.1 How do I write a Query that inserts many Table records? 390
F.13.2.2 How do I write a Query that inserts only one Table record? 391
F.13.2.3 What are the common characteristics to all Insert operations? 392
F.13.2.4 What is the summary of Insert operations? 395
F.13.3 What is an Update operation and how do I write it? 396
F.13.4 Can I write a VBA function that deletes, inserts or updates Table records? 396
F.13.5 When should I use SQL operations that change records from my Tables? 396
F.14 How do I write and debug my SQL Queries? 396
PART G. WRITING EXPRESSIONS 398
G.1 What are the main differences between the three expression scopes? 398

G.1.1 How available field names depend on expression scopes? 399
G.1.2 How writing field names depends on expression scopes?.......................... 400
G.1.3 How data types depend on expression scopes? 400
G.1.4 How constants depend on expression scopes?... 400
G.1.5 How value operators depend on expression scopes?................................. 400
G.1.6 How non-aggregate built-in functions depend on expression scopes?....... 401
G.1.7 How domain aggregate functions depend on expression scopes?............. 402
G.1.8 How SQL aggregate functions depend on expression scopes? 402
G.1.9 How user-defined functions depend on the expression scopes?................ 402
G.1.10 How using SQL operations ("Subqueries") depends on the expression scopes?.. 402
G.2 How do I manage VBA data types and Table field types-sizes?....................... 403
G.2.1 What VBA data types vs. Table field types-sizes are equivalent? 403
G.2.2 What VBA data types vs. Table field types-sizes are not equivalent?....... 404
G.2.3 What is the result of combining different data/field types in expressions? 405
G.2.4 How are data/field types grouped for easier reference? 406
G.2.5 How do I force a value to belong to a specific data type?.......................... 408
G.2.6 Why should I force a value into a specific data type? 409
G.3 What is the data type returned by an expression?.. 410
G.3.1 What is the data type of a constant? 410
G.3.2 What is the data type returned by a non-arithmetic operator?................... 410
G.3.3 What is the data type returned by an arithmetic operator? 411
G.3.4 What is the data type returned by a function? 411
G.4 How do I write a constant? 411
G.4.1 How do I write *Boolean* constants?.. 412
G.4.2 How do I write *String* constants?... 412
G.4.3 How do I write *Byte*, *Integer*, *Long Integer*, *LongLong*, *Currency*, *Single* and *Double* constants?.. 413
G.4.4 How do I write *Date* constants? 413
G.4.5 What are the restrictions when writing constants in VBA expressions?.... 413
G.5 How do I use value operators in an expression?.. 414
G.5.1 What are the Arithmetic operators?... 415
G.5.2 What are the Text string operators?... 416
G.5.3 What are the Comparison operators?... 417
G.5.4 What are the Pattern operators?.. 418
G.5.5 What are the Logical (*Boolean*) operators?... 419
G.5.6 What are the "`IS NULL`" and "`IS NOT NULL`" Miscellaneous operators?

.... 420
G.5.7 What are the "IN" and "NOT IN" Miscellaneous operators? 421
G.5.8 What are the "BETWEEN AND" and "NOT BETWEEN AND" Miscellaneous operators? 422
G.6 How do I use functions in an expression? 423
G.6.1 How do I use non-aggregate built-in functions in an expression? 424
G.6.2 How do I use domain aggregate functions in an expression? 426
G.6.3 How do I use SQL aggregate functions in an expression? 428
G.6.4 How do I use user-defined VBA functions in an expression? 429
G.7 What is the evaluation order of an expression? 429
G.8 How do I use an SQL operation in an expression? 430
G.8.1 How do I use a Subquery in an SQL expression? 431
G.8.2 What are the "EXISTS", "ANY" and "ALL" Subquery operators? 432
G.8.2.1 What is the "EXISTS" Subquery operator? 432
G.8.2.2 What is the "ANY" Subquery operator? 433
G.8.2.3 What is the "ALL" Subquery operator? 433
G.8.3 What is a correlated Subquery? 434
G.8.4 What is an uncorrelated Subquery? 434
G.8.5 How do I use an SQL operation in a VBA variable assignment? 434
G.9 How are numeric-like values internally represented and processed? 435
G.9.1 How are numeric-like values internally represented? 435
G.9.2 What is decimal/binary conversion? 435
G.9.3 What are decimal/binary conversion rounding errors? 436
G.9.4 What is the cause of decimal/binary conversion rounding errors? 437
PART H. CUSTOMIZING THE APPEARANCE OF A QUERY/TABLE/FORM IN "DATASHEET VIEW" 439
H.1 How do I change the column width, or freeze/unfreeze the columns in a Table/Query/Form? 439
H.1.1 How do I change the column width in a Table/Query/Form? 439
H.1.2 How do I freeze/unfreeze the columns in a Table/Query/Form? 440
H.1.2.1 How do I freeze the columns in a Table/Query/Form? 441
H.1.2.2 How do I unfreeze all the columns in a Table/Query/Form? 441
H.2 How do I change row height, hide rows or change row order in a Table/Query/Form? 442
H.2.1 How do I change the height of all the rows in a Table/Query/Form? 442
H.2.2 How do I hide rows in a Table/Query/Form? 443
H.2.3 How do I change the sorting of rows in a Table/Query/Form? 443

H.3 How do I change the order of columns that I see in a Table/Query/Form?....... 445
H.3.1 How do I change the order of columns in a Table/Query/Form? 445
H.3.2 How is it related a Table's column order and its field order in "Design View"? 446
H.3.3 How is it related a Query's column order and its field order in "Design View" and "SQL View"? 446
H.3.4 How is it related a Forms' column order and shown/hidden columns and the ones of its associated Table? 447
H.4 How do I hide/unhide columns in a Table/Query/Form? 447
H.4.1 How do I hide the columns that I see in a Table/Query/Form?.................. 448
H.4.2 How do I unhide columns in a Table/Query/Form? 448
H.4.3 How are related a Table/Query's shown/hidden columns and its fields in other views? 449
H.4.4 How is it related a Forms' column order and shown/hidden columns and the ones of its associated Table? 449
H.5 How do I change the column headings in a Table/Query/Form? 450
H.5.1 How do I change a Table's column headings? 450
H.5.2 How do I change a Query's column headings? 451
H.5.3 How do I change a Form's column headings?.. 451
H.5.4 How are related a Form's column headings and the ones of its associated Table? 452
H.6 How do I configure the formatting of column values in a Table/Query/Form? 452
H.6.1 How do I configure predefined column formatting in a Table/Query/Form? 453
H.6.1.1 What "Format" options are available in Short Text and Long Text fields? 453
H.6.1.2 What "Format" options are available in Yes/No fields?................................. 453
H.6.1.3 What "Format" options are available in Number, Large Number, Currency and Date/Time fields?... 454
H.6.1.4 What "Format" options are available in Calculated fields? 456
H.6.2 How do I configure custom column formatting in a Table/Query/Form?.. 457
H.6.2.1 How do I configure custom formatting for all field types? 457
H.6.2.2 How do I configure custom formatting for Short Text fields? 458
H.6.2.3 How do I configure custom formatting for Number, Large Number and Currency fields? 459
H.6.2.4 How do I configure custom formatting for Date/Time field type? 460
H.6.3 How do I find the column formatting properties in a Table/Query/Form? 461
H.6.3.1 How do I find a Table's column formatting properties? 462
H.6.3.2 How do I find a Query's column formatting properties? 462

H.6.3.3 How do I find a Form's column formatting properties?.................................. 463
H.7 How do I configure the column text alignment in a Table/Query/Form? 464
H.7.1 How do I configure a Table/Form's column text alignment?..................... 464
H.7.2 How do I configure a Query's column text alignment? 465
H.8 How do I show aggregate values (e.g., totals) in a Table/Query/Form? 465
H.8.1.1 How do I show aggregate values in Short Text and Yes/No fields?................ 466
H.8.1.2 How do I show aggregate values in Date/Time field? 466
H.8.1.3 How do I show aggregate values in Number, Large Number and Currency fields?... 467
H.9 How do I configure colors, fonts and other features of a Table/Query/Form?.. 467
PART I. EVOLVING MY DATABASE DESIGN... 469
I.1 Why would I want to improve/modify my database design?............................. 469
I.2 Why should I be so careful with any change to the database design? 469
I.2.1 What are database element dependencies?.. 470
I.2.2 What are side effects? 471
I.3 How do I find all the dependent objects on a given database object?................. 472
I.4 What are the side effects of modifying my Table fields? 472
I.4.1 What are the side effects of adding a field to a Table? 472
I.4.2 What are the side effects of deleting a Table field? 474
I.4.3 What are the side effects of changing the name of a Table field? 476
I.4.4 What are the side effects of modifying the properties of a Table field? 477
I.4.4.1 What are the side effects of changing the "Field Type" and/or "Field Size" properties of a Table field?.. 477
I.4.4.2 What are the side effects of changing the "Required", "Allow zero length", "Validation rule" or "Indexing" properties of a Table field?.. 481
I.4.4.3 What are the side effects of changing a drop-down menu of a Table field?...... 481
I.4.5 What are the side effects of changing the order of Table fields?................. 481
I.5 What are the side effects of modifying my Tables?... 482
I.5.1 What are the side effects of adding a new Table?....................................... 483
I.5.2 What are the side effects of deleting a Table? 483
I.5.3 What are the side effects of changing the name of a Table?........................ 484
I.5.4 What are the side effects of modifying a Table field? 484
I.5.5 What are the side effects of modifying the order of fields in the Table?..... 484
I.5.6 What are the side effects of modifying the indexes or the Key fields of a Table? 484
I.5.6.1 What are the side effects of changing an index with duplicate values?............ 485
I.5.6.2 What are the side effects of adding an index without duplicate values?........... 485

I.5.6.3 What are the side effects of changing an index without duplicate values? 485
I.5.6.4 What are the side effects of adding or changing the Key fields of a Table? 486
I.5.7 What are the side effects of modifying the record validation rule of a Table? 486
I.6 What are the side effects of modifying my Queries? 486
I.6.1 What are the side effects of adding a new Query? 487
I.6.2 What are the side effects of deleting a Query? 487
I.6.3 What are the side effects of modifying the functionality of a Query? 487
I.6.4 What are the side effects of changing the name of a Query? 488
I.6.5 What are the side effects of adding a new field to a Query? 489
I.6.6 What are the side effects of deleting a Query field? 490
I.6.7 What are the side effects of changing the name of a Query field? 491
I.6.8 What are the side effects of changing the data type of a Query field? 491
I.6.9 What are the side effects of modifying the order of fields in a Query? 492
I.7 What are the side effects of modifying my user-defined VBA functions? 493
I.8 What are the side effects of modifying my Relationships, Forms and/or Reports? 494
PART J. DEBUGGING MY SQL QUERIES 495
J.1 How do I fix an error/crash in a test-and-proven Query? 495
J.2 How do I fix an error/crash in a non-test-and-proven Query? 497
J.3 How do I debug by commenting/uncommenting? 497
J.4 How do I debug in progressive steps? 498
J.5 How do debug inside out? 499
J.6 How do I debug my same-level code linearly? 499
J.7 How do I debug the current uncommented SQL operation at each step? 500
J.7.1 How do I debug the current uncommented SQL operation so it can be saved? 501
J.7.2 How do I debug the current uncommented SQL operation so it runs? 504
J.7.3 How do I debug the current uncommented SQL operation so it does not produce defective results? 506
J.7.4 How do I debug the current uncommented SQL operation's functionality? 507
J.7.5 How do I check the results of the current uncommented SQL operation? .. 509
J.8 How do I fix a syntax error that prevents saving a Query? 510
J.9 How do I fix a crash from a syntax error? 511
J.10 How do I fix a crash from a run-time error? 517
J.10.1 How do I fix the crash "*Divide by zero.*"? 517
J.10.2 How do I fix the crash "*Overflow.*"? 518

J.10.3 How do I fix the crash "*Data type mismatch in criteria expression.*"? 519
J.10.4 How do I fix the crash "*Data type mismatch.*"? 520
J.10.5 How do I fix the crash "*Invalid Procedure Call or Argument.*"? 521
J.10.6 How do I fix the crash "*Invalid use of Null.*"? 521
J.10.7 How do I fix the crash "*The expression cannot be used in a calculated column.*"? 521
J.10.8 How do I fix the crash "*Cannot have … in aggregate argument.*"? 522
J.10.9 How do I fix the crash "*Could not find field…*"? 522
J.10.10 How do I fix the crash "*The expression cannot be used in a Calculated column*"? 522
J.10.11 How do I fix the crash "*Expression is too complex.*"? 523
J.10.12 How do I fix the crash "*Query is too complex.*"? 523
J.10.13 How do I fix the crash "*Cannot open any more databases.*"? 523
J.10.14 How do I fix a crash from my user-defined VBA functions? 523
J.11 How do I fix defective Query results? 524
J.11.1 How do I fix a Query showing a wrong numeric-like value? 525
J.11.2 How do I fix a Query showing a wrong Short Text value? 525
J.11.3 How do I fix a Query erroneously considering two different values as equal? 525
J.11.4 How do I fix a Query erroneously considering two equal values as different? 526
J.11.5 How do I fix a Query erroneously ordering records? 527
J.11.6 How do I fix a Query requesting a non-declared parameter? 527
J.11.7 How do I fix a Query requesting a non-existing parameter? 528
J.11.8 How do I fix a Query requesting the same parameter twice (or more times)? 528
J.11.9 How do I fix a Query showing "########" in a field? 529
J.11.10 How do I fix a Query producing "#Num!", "#Div/0!", "#Error", "#Type!" or "#Func!" in a field? 529
J.11.11 How do I fix a Query producing "#Invalid" or "#Deleted" in a field? 530
J.11.12 How do I fix a Query showing a black square in a field? 531
J.11.13 How do I fix a Query producing "#Name?" in a field? 531
J.11.14 How do I fix a Query showing a blank field that should not be blank? 531
J.11.15 How do I fix a Query producing defective values? 532
J.11.16 How do I fix a Query's defective "ON" expression? 533
J.11.17 How do I fix a Transform Query producing wrong field names? 533
J.11.18 How do I fix a Query that stalls/freezes? 534

J.11.19 How do I fix a Query making arithmetic errors? 534
J.11.20 How do I fix a Query making rounding errors? 535
J.11.21 How do I fix a Query that I cannot open in "Design View"? 539
J.11.22 How do I fix my VBA functions comparing text strings case sensitive? 539
J.12 What do I do when I just cannot fix a Query? 539
J.13 Why should I always compare the results of an existing Query? 540
J.13.1 What is a record-list comparator tool? 541
J.14 What Null-related bugs can I get? 542
J.14.1 What effect does Null cause in Query results? 542
J.14.2 What effects does Null cause in Arithmetic, Comparison and Pattern value operators? 542
J.14.3 What effects does Null cause in Logical value operators? 542
J.14.4 What effects does Null cause in Text string value operators? 543
J.14.5 What effects does Null cause in the "`IN`" and "`NOT IN`" Miscellaneous value operators? 543
J.14.6 What effects does Null cause in SQL operators that remove duplicate records? 543
J.14.7 What effect does Null cause in Union SQL operators? 543
J.14.8 What effects does Null cause in aggregate functions? 544
J.14.9 What effects does Null cause as an argument of a VBA function? 544
J.15 What exception-value bugs can I get? 545
J.15.1 What is an exception-value? 545
J.15.2 What operations produce number-overflow "#Num!"? 546
J.15.3 How do I prevent exception-values? 547
J.15.4 What is the effect of an exception-value in value operators? 548
J.15.5 What is the effect of an exception-value in function arguments? 548
J.15.6 What is the effect of an exception-value in expressions? 548
J.15.7 When is an exception-value crashing the Query? 548
J.16 What data type bugs can I get? 549
J.16.1 What data type bugs can I get with constants? 549
J.16.2 What data type bugs can I get with value operators? 550
J.16.2.1 What data type problems can I get with Arithmetic operators? 550
J.16.2.2 What data type problems can I get with Text string operators? 551
J.16.2.3 What data type problems can I get with Comparison operators? 551
J.16.2.4 What data type problems can I get with Pattern operators? 552
J.16.2.5 What data type problems can I get with Logical operators? 552
J.16.2.6 What data type problems can I get with Miscellaneous operators? 552

J.16.3 What data type bugs can I get with the Union operator? 552
J.16.4 What data type bugs can I get with aggregate functions? 553
J.16.5 What data type bugs can I get with VBA functions? 553
PART K. USEFUL DESIGN ADVICE 555
K.1 What are good practices in my Table design? 555
K.1.1 Why should I write field descriptions in my Table fields? 555
K.1.2 Why should I add validation rules to my Tables? 556
K.1.2.1 Why should I add field validation rules to my Table fields? 556
K.1.2.2 Why should I add record validation rules to my Tables? 557
K.1.3 Why should I prevent Nulls in my Table fields? 557
K.1.4 Why should I add a “Comments” field to my Tables? 558
K.1.5 Why should I add at least one Date/Time field to my Tables? 558
K.1.6 How to prevent errors in Short Text fields? 560
K.1.7 Why should I configure drop-down menus to enter data? 561
K.1.8 What are good practices in configuring my drop-down menus? 561
K.1.8.1 What “Row Source Type” and “Row Source” should I use in my drop-down menus? 562
K.1.8.2 Should I create a Relationship from an auxiliary Table used in drop-down menus? 563
K.1.8.3 Should I configure “Limit to List=Yes” in my drop-down menus? 564
K.1.9 How do I configure a drop-down menu in a Date/Time field? 565
K.2 What are other good practices in my database design? 566
K.2.1 Why should I hide unfrequently used objects? 566
K.2.2 Why should I add a “T_Numbers” Table with integer numbers? 567
K.2.3 What are the interactions between Nulls, duplicates, indexing, and Key field(s)? 567
K.2.4 What are the interactions between Relationships and “PrimaryKey”, “Required” and Table indexes? 569
K.2.5 What are the actual fields and the actual field order in a Table/Query/Form? 569
K.2.6 What is the difference between a Calculated field and automatically introducing a value using a Form? 570
K.2.7 How should I back-up MS-Access database files? 571
K.2.8 How do I store a list of values, instead of a single value, in a Table field? 571
K.2.9 Why and how should I maximize Table data correctness and coherence?. 572
K.2.10 How do I restrict the values introduced in my Table fields? 572
K.3 How do I structure and optimize a distributed database? 577

K.3.1 How do I organize my database files? 577
K.3.2 How do I create user-specific views of my shared database? 579
K.3.3 What are frontend files, backend files and source files? 580
K.3.4 What are local Tables, linked Tables and source Tables? 580
K.3.5 How do I get a split database? 582
K.3.6 How do I create a linked Table? 583
K.3.7 How do I refresh a Table link? 584
K.3.8 How do I convert a linked Table to a local Table? 584
K.3.9 How do I view a Table link? 585
K.3.10 How do I view and manage Table links? 585
K.3.11 How do I delete a linked Table? 588
K.3.12 How do I manage Relationships with linked Tables? 588
K.3.13 Can I change the file path or name of a source file? 589
K.3.14 What are the options to lock records in a shared database? 589
K.3.15 What is the delay of network access to a database? 590
K.3.16 Why should I make temporal Tables local? 591
K.4 What Query design principles should I follow? 591
K.4.1 How do I write my SQL Queries? 592
K.4.2 Why should I incrementally run my SQL code while I write a Query? 592
K.4.3 How do I write readable (maintainable) SQL Queries? 593
K.4.4 Why should I disable changing Table data from Query results? 599
K.4.5 Why should I qualify all the outermost output field names of my Queries? 600
K.4.6 When should I remove duplicate records? 600
K.4.7 Why should I enclose the outermost Union operation in a Select operation? 601
K.4.8 Why should I restrict the usage of “`INNER JOIN`”? 602
K.4.9 Why should I avoid using “`SELECT *`”? 602
K.4.10 Why should I avoid using the same name of an input field name for an output expression? 603
K.4.11 When are “`SELECT`” expressions evaluated along Query processing? ... 603
K.5 Why and how should I carefully handle Nulls in my Queries? 604
K.5.1 What is a Null? 604
K.5.2 What problems can Null produce? 604
K.5.3 How is a Null produced? 604
K.5.4 How do I handle Nulls in my Queries? 605
K.5.5 Where do I handle Nulls in my Queries? 607

K.5.6 Why is MS-Access not handling Null fields as I indicated? 607
K.5.6.1 How are Null fields created in “LEFT JOIN” and “RIGHT JOIN”? 608
K.5.6.2 Why should I handle Null fields in all the innermost SQL operations? 609
K.6 What are some useful models of SQL code? 613
K.6.1 How do I generate/create record-lists? 614
K.6.2 How do I replicate record-lists? 615
K.6.3 How do I produce totals in addition to individual results? 616
K.6.4 How do I convert rows into columns? 618
K.6.5 How do I convert columns into rows? 621
K.6.6 How do I use a Transform operation to show a cross table? 622
K.6.7 How do I produce the non-matching records of a Join operation? 624
K.6.8 How do I write a Full-Outer-Join? 627
K.6.9 How do I “merge” two record-lists? 629
K.6.10 How do I produce a weighted average? 632
K.6.11 How do I design a data check Query? 634
K.6.12 How do I get the exact record ordering I want? 636
K.6.12.1 How do I order over several expressions? 636
K.6.12.2 How do I order over only one expression? 636
K.6.12.3 How do I order over user-defined functions? 637
K.6.12.4 How do I order over “GROUP BY” expressions? 637
K.6.12.5 How do the four ordering approaches compare among themselves? 637
K.6.13 How do I use *String* fields to display different data types in the same Query column? 638
K.6.14 What are useful tricks with SQL aggregate functions? 639
K.6.14.1 How do I use the “AllNull()” SQL aggregate function? 640
K.6.14.2 How do I use the “AllEqual()” SQL aggregate function? 640
K.6.14.3 How do I use the “Or()” or “And()” SQL aggregate functions? 641
K.7 Why and how do I design a fast database and fast Queries? 642
K.7.1 How do I design a fast database? 643
K.7.2 How do I design faster Select operations over Queries? 644
K.7.2.1 Why should I use “DISTINCT”, “UNION” and “ORDER BY” only if needed? 644
K.7.2.2 Why should I use the most restrictive “WHERE” and “HAVING” expressions in my Select operations? 645
K.7.2.3 Why should I mainly use comparison and logical operators in my “WHERE” and “HAVING” expressions? 646
K.7.3 How do I design faster Select operations over Tables? 647

K.7.3.1 Why should I avoid bringing unnecessary data from Tables?.......................... 647
K.7.3.2 Why should I consider using uncorrelated Subqueries and/or domain aggregate functions in "`WHERE`" expressions over Tables? 649
K.7.4 How do I design faster Union operations? 650
K.7.5 How do I design faster Join operations?.. 650
K.7.5.1 Why should I always restrict in inner Selects?... 650
K.7.5.2 Why should I mainly use comparison and logical operators in my "`ON`" expressions? 651
K.7.6 How do I design a faster Select-group_by when I have bound values in all my records?.. 652
K.8 Why should I avoid using Decimal data types?.. 653
K.8.1 What are the Decimal data types? 653
K.8.2 Why should I avoid using the Number-Decimal Table field type? 653
K.8.3 Why should I avoid using the *Variant-Decimal* VBA data type? 654
K.8.4 How do Decimal data types work?... 655
K.9 How do I write my user-defined VBA functions and database Subroutines? ... 655
K.9.1 How do I create VBA modules and open the VBA editor?....................... 656
K.9.2 How do I write a VBA function that inserts/updates/deletes Table's records? 658
K.9.3 How do I write a VBA function that reads database records?................... 659
K.9.4 How do I write a VBA function that requests a parameter to the user? 660
K.9.5 How do I test my user-defined VBA functions? 660
K.9.6 How do I write VBA Subroutines associated to Form Events?................. 662
K.9.7 How do I close the VBA editor? 662
K.10 What elements/concepts are explained in various places? 663
K.10.1 Where are names and identifiers explained? 663
K.10.2 Where are drop-down menus explained? 663
K.10.3 Where are duplicates explained? 664
K.10.4 Where are indexes explained? 664
K.10.5 Where are Key fields explained?.. 664
K.10.6 Where are Relationships explained?.. 665
K.10.7 Where is Query results ordering explained?.. 665
K.10.8 Where are data types explained? 665
K.10.9 Where are zero-length and invisible strings explained?.......................... 666
K.10.10 Where are exception-values explained? 666
K.10.11 Where are aggregate functions explained?.. 666
K.10.12 Where is remote database access explained?... 667

K.10.13 Where is VBA code explained? 667
K.10.14 Where are Nulls explained? 667
PART L. FIXING DATABASE ERRORS 669
L.1 Why can I get an error/crash in a test-and-proven database? 669
L.2 How do I fix errors with my Table/Form design? 670
L.2.1 How do I fix data integrity errors when saving my Table design? 670
L.2.1.1 How do I fix "*Nulls in Required field*" when saving my Table design? 673
L.2.1.2 How do I fix "field validation rule violated" when saving my Table design? 673
L.2.1.3 How do I fix "record validation rule violated" when saving my Table design? 673
L.2.1.4 How do I fix field Type-size change errors when saving my Table design? 674
L.2.1.5 How do I fix that I cancelled the Table data integrity check? 674
L.2.2 How do I fix orphan Calculated fields when saving my Table design? 675
L.2.3 How do I fix "*Calculated column cannot be saved…*" when saving my Table design? 675
L.2.4 How do I fix "*Could not find field…*" when writing the expression of a Calculated field? 675
L.2.5 How do I fix "*Could not find field…*" when saving my Table design? 675
L.2.6 How do I fix "*Key/Index with duplicate values*" when saving my Table design? 676
L.2.7 How do I fix "*…primary key cannot contain a Null…*" when saving my Table design? 677
L.2.8 How do I fix "*You can't change the primary key*" when saving my Table design? 677
L.2.9 How do I fix "*Cannot delete this index…*" when saving my Table design? 678
L.2.10 How do I fix "*There is no primary key…*" when saving my Table design? 678
L.3 How do I fix errors in my Relationship configuration? 679
L.3.1 How do I fix "different number of fields" Relationship error? 679
L.3.2 How do I fix "different field types" Relationship error? 680
L.3.3 How do I fix "different field sizes" Relationship error? 680
L.3.4 How do I fix "violates referential integrity" Relationship error? 681
L.3.5 How do I fix "missing index" Relationship error? 681
L.3.6 How do I fix "…could not lock table…" Relationship error? 682
L.3.7 How do I fix a removed Relationship when I create a new one? 682
L.3.8 How do I fix showing a wrong Relationship type? 682
L.3.9 How do I fix "reversed Relationship Tables and fields" error? 683
L.3.10 How do I fix that I cannot configure a Relationship between a Table and itself? 684

L.3.11 How do I fix a malfunctioning Table slave record?.................................. 684
L.3.12 How do I fix a malfunctioning Relationship?... 684
L.4 How do I fix errors when entering, modifying or deleting records?.................. 685
L.4.1 How do I fix various field value errors? 685
L.4.1.1 How do I fix that I cannot type-in a text string in full? 685
L.4.1.2 How do I fix that I cannot paste into a field's value? 685
L.4.1.3 How do I fix a changed numeric value? 686
L.4.1.4 How do I fix a changed Short Text value that I typed-in? 687
L.4.1.5 How do I fix a Table Short Text field configured as "Required=Yes" that accepts a Null? 687
L.4.1.6 How do I fix typing-in invisible characters into a Short Text Table field?....... 687
L.4.2 How do I fix error messages when saving a field value?............................ 688
L.4.2.1 How do I fix an invalid value"?.. 688
L.4.2.2 How do I fix trying to save Null in a "Required" field?.................................. 689
L.4.2.3 How do I fix violating "Allow Zero Length=No"? 689
L.4.2.4 How do I fix that I cannot paste a text string in full?....................................... 689
L.4.2.5 How do I fix a value rejected because it is not in the list?............................... 690
L.4.2.6 How do I fix a value violating a field validation rule? 690
L.4.3 How do fix error messages when entering a record? 690
L.4.3.1 How do I fix a record violating a record validation rule?................................ 690
L.4.3.2 How do I fix erroneous records because of duplicate values?.......................... 691
L.4.3.3 How do I fix a slave record without a master record? 691
L.4.3.4 How do I fix an erroneous slave record? 692
L.4.4 How do I fix errors when pasting records into a Table/Form? 692
L.4.4.1 What pasting errors can I get from invisible characters in Text fields?............ 694
L.4.4.2 What pasting errors can I get from data copied from Excel?............................ 695
L.4.4.3 What pasting errors can I get from Windows' cut/paste buffer? 696
L.4.5 How do I fix a record I cannot delete?... 696
L.5 How do I fix errors in Table/Form data? 696
L.5.1 How do I fix a Table/Form showing "########" in a field? 697
L.5.2 How do I fix a Table/Form showing "#Num!", "#Div/0!", "#Type!" or "#Func!" in a field?.. 698
L.5.3 How do I fix a Table/Form showing "#Invalid" in a field?......................... 698
L.5.4 How do I fix a Table/Form showing "#Deleted" in a field?....................... 699
L.5.5 How do I fix a Form showing "#Name?" in a field? 699
L.5.6 How do I fix a Table/Form showing a black square in a checkbox field?.. 699
L.5.7 How do I fix a Table/Form showing a wrong numeric-like value in a field?

.... 700
L.5.8 How do I fix a Table/Form showing a wrong Short Text value in a field?. 700
L.5.9 How do I fix a Table/Form erroneously ordering records? 700
L.5.10 How do I fix a value not in the drop-down menu in a Table/Form field? 701
L.5.11 How do I fix a Null in a Table/Form field configured as "Required=Yes"? 701
L.5.12 How do I fix an apparent Null in a Table/Form Short Text field configured as "Required=Yes"? 701
L.5.13 How do I fix an apparent zero-length string in a Table/Form field configured as "Allow Zero Length=No"? 702
L.5.14 How do I fix field value(s) that violate(s) the field/record validation rule(s)"? 703
L.5.15 How do I fix spontaneously changing Table/Form field values? 703
L.6 How do I fix a Table/Form that I cannot open? 703
L.6.1 How do I fix "*The record source…does not exist*"? 704
L.6.2 How do I fix "*…database engine could not find the object.*"? 704
L.6.3 How do I fix "*Could not find file…*"? 704
L.6.4 How do I fix "*…is not a valid path*"? 705
L.7 How do I fix errors with Short Text or *String* fields? 705
L.7.1 Why text strings can be different from what is shown? 706
L.7.2 What unexpected results can I get because text strings are different from what is shown? 706
L.7.3 How do I fix text strings and insufficient field width? 707
L.7.4 How do I fix text strings and insufficient field height? 708
L.7.5 How do I fix text strings and invisible characters? 708
L.7.6 What are invisible characters? 709
L.7.7 What is the zero-length text string? 710
L.7.8 What are invisible text strings? 711
L.7.9 Why can I have invisible characters in my Short Text and *String* fields? .. 711
L.7.10 What apparently defective results can invisible characters produce? 711
L.8 How do I fix errors with the user interface? 712
L.8.1 How do I fix a missing or disappearing command/tool? 713
L.8.2 How do I fix a shaded new-record button? 713
L.8.3 How do I fix a Query opening in "SQL View" when I selected "Design View"? 713
L.8.4 How do I fix the "Access SQL Editor" marking a Query as having unsaved changes? 713
L.8.5 How do I fix an error when opening a Query in "Design View"? 714

L.8.6 How do I fix an extra column in my Tables in “Datasheet View”? 715
L.8.7 How do I fix an extra row in my Query results?.. 715
L.8.8 How do I fix the database users modifying Table values by editing Query results? 715
L.8.9 How do I fix missing columns/rows in my Query results?......................... 715
L.8.10 How do I fix “*Could not find field…*” when doing “Compact and Repair” or “Save As”?... 716
L.8.11 How do I fix the VBA editor changing the value of the constants I write? 716
L.8.12 How do I fix foreign-language issues of MS-Access?.............................. 716
PART M. LIST OF BUILT-IN FUNCTIONS 720
M.1 ActiveX functions 721
M.2 Application functions.. 721
M.3 Array functions 721
M.4 Conversion functions 721
M.5 Database functions 722
M.6 Date and Time functions.. 722
M.7 Domain Aggregate functions 722
M.8 Error Handling functions 723
M.9 File Input/Output functions.. 723
M.10 Financial functions.. 723
M.11 Inspection functions 723
M.12 Mathematical functions.. 724
M.13 Message functions... 724
M.14 Miscellaneous functions... 724
M.15 Program Flow functions.. 725
M.16 SQL aggregate functions... 725
M.17 File Management functions... 725
M.18 Text Processing functions 725
PART N. CONTENTS AND ACKNOWLEDGEMENTS I

PREVIEW OF SECTION F.10

F.10 What is a Transform operation and how do I write it?

This chapter also answers the questions:

- **How do I create cross tables?**

[57] There could be some type conversion combinations that make **Union** operators non-associative.

- **What is a crosstab Query?**

A **Transform operation** is an SQL operation performed with the **Transform** operator (click *F.10.1*) plus its corresponding operands. Therefore, a **Transform** operation is the complete SQL code associated to the **Transform** operator.

A **Transform** operation produces cross tables in a very flexible way. In its **simplest** form, a **Transform** operation **generates new field names** from the **values** of the **field** written in the "**PIVOT**" clause, **and also**, places the **values** of the **expression** written in the "**TRANSFORM**" clause in the **correct** places **under** the **newly generated field names**.

Transform is **extremely** useful to **display** your Query **results** as a **cross table**. Imagine you have the following record-list (in this case it is a Table[58], but most frequently it is a Query result or an inner SQL operation):

T_Capital_Rainfall		
Capital	**Cal_Year**	**Rainfall**
Beijing	2018	25
Beijing	2019	41
Washington	2018	22
Washington	2019	17
Beijing	2020	27
Beijing	2021	38
Washington	2020	26
Washington	2021	9

Now you want to display this information as a **cross table**, with **fields** "**Capital**" (renamed to "**Cap_City**"), "**2018**", "**2019**", "**2020**" and "**2021**", and the **value** of "**Rainfall**" in the **corresponding cell under the fields** "**2018**", "**2019**", "**2020**" and "**2021**". In summary, what you want is:

F_Transform_1				
Cap_City	**2018**	**2019**	**2020**	**2021**
Beijing	25	41	27	38
Washington	22	17	26	9

Then, this is **exactly** what you would get with the following **Transform** operation[59]:

```
TRANSFORM First(Rainfall) AS GenVals
SELECT Capital AS Cap_City
FROM T_Capital_Rainfall
GROUP BY Capital
PIVOT Cal_Year ;
```

The "**n**" **leftmost output fields** of a **Transform** are called the "**SELECT**" **fields**, because they are determined by the "**SELECT**" clause. The **output fields** placed to the **right** of the "**SELECT**" **fields** are called the "**PIVOT**" **fields**, because they are

[58] This is the Table "**T_Capital_Rainfall**" in file "**Company_Database.accdb**".
[59] This is the Query "**F_Transform_1**" from the "**Company_Database.accdb**" file.

generated by converting to ***String*** the **returned values** of the "**PIVOT**" **expression** (click *F.10.11*), subject to the modifications of the optional "**IN**" list (click *F.10.12*). In this example, we have one "**SELECT**" **field** (field name "**Cap_City**") and four "**PIVOT**" **fields** (with field **names** "**2018**", "**2019**", "**2020**" and "**2021**").

Transform operations **cannot** be nested. The **Transform** operation must be **the first** operation in a Query, and a Query **cannot contain** any other **Transform** operation. Consequently, a **Transform** operation **cannot** be used as an **input record-list** in any other Query or SQL operation. A Query that has a **Transform** operation **cannot** be used in any other Query. The **Transform** operation can therefore only be the **very last** operation over a record-list.

If you want to know more about a **Transform** operation, you may click:

- "*F.10.1 What is the Transform operator?*"
- "*F.10.2 What is the input record-list ("FROM" clause) of a Transform?*"
- "*F.10.3 What are the output fields of a Transform?*"
- "*F.10.4 What is the output record-list of a Transform?*"
- "*F.10.5 Can I see an example of a Transform operation?*"
- "*F.10.6 What is the "TRANSFORM" clause of a Transform?*"
- "*F.10.7 What is the "SELECT" clause of a Transform?*"
- "*F.10.8 What is the "ORDER BY" clause of a Transform?*"
- "*F.10.9 What is the "WHERE" clause of a Transform?*"
- "*F.10.10 What is the "GROUP BY" clause of a Transform?*"
- "*F.10.11 What is the "PIVOT" clause of a Transform?*"
- "*F.10.12 What is the "IN" clause of a Transform?*"
- "*F.10.13 How do the clauses from Transform and Select compare?*"
- "*F.10.14 How do I write a correct (syntax) Transform?*"

F.10.1 What is the Transform operator?

Transform is a **consulting** (click *F.6*) operator that works over one single **input record-list**, plus other operands that are not record-lists.

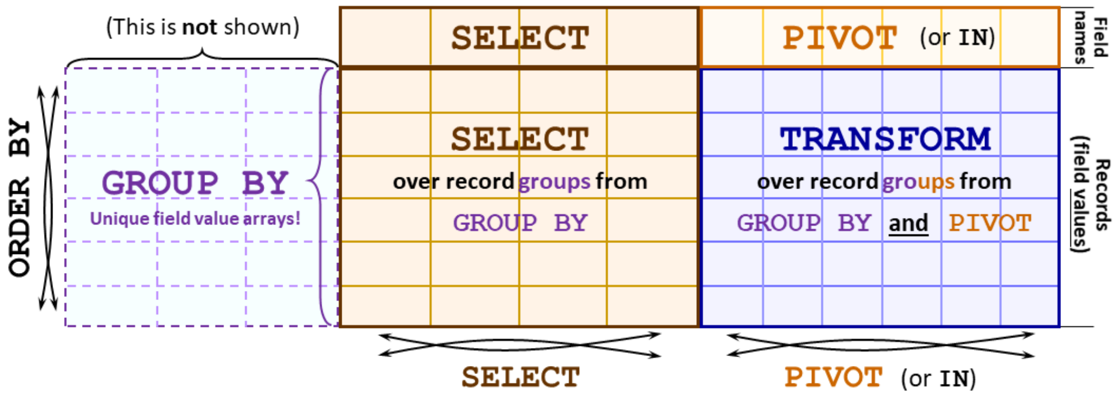
The **Transform** operator includes the clauses "**SELECT**", "**DISTINCT**" (optional and irrelevant), "**DISTINCTROW**" (optional and not advisable), "**FROM**", "**WHERE**" (optional), "**GROUP BY**", "**ORDER BY**" (optional) and "**IN**" (optional).

A simplified view of writing (syntax) a **Transform operation** is:

```
  TRANSFORM PIVOT-field-values-expression()
  SELECT [DISTINCT] [DISTINCTROW] Output-expression() 1 to n
  FROM Input-record-list
[ WHERE Where-Boolean-expression(Input-field-names) ]
  GROUP BY Group_by-expression(Input-field-names) 1 to k
[ ORDER BY Order_by-expression() 1 to w ]
  PIVOT PIVOT-field-names-expression(Input-field-names) ;
[ IN ( List-of-PIVOT-values ) ]
```

The clauses enclosed between square brackets "`[]`" are optional.

The following picture shows the **output** of the **Transform** operation:



The **Transform** operator creates a **cross table** by converting the **distinct values** of the "**PIVOT**" **expression** (click *F.10.11*) into **additional** **generated** **field names** (i.e., "**columns**") and presenting the **results** of the "**TRANSFORM**" **expression** (click *F.10.6*) in the corresponding **cells** (**row** x **column**).

The **Transform** operator produces:

- The **rows** by doing record **aggregation** over the (mandatory) "**GROUP BY**" **expressions**.
- The **fixed** leftmost **columns** (the "**SELECT**" **fields names** and **values**) with the conventional "**SELECT**" **expressions**.
- The **cross table column names** (the "**PIVOT**" **field names**) by doing record **aggregation** over the (mandatory) "**PIVOT**" **expression**,
- The **cross table column values** by doing record **aggregation** of the "**TRANSFORM**" **expression** over the "**GROUP BY**" **expressions** **jointly with** the "**PIVOT**" **expression**.

Remind that record aggregation produces **only one value** from **each group** of records. I now explain the above bullet points in more detail.

The **number** of **rows** is determined by the **1** to **k** "**GROUP BY**" **expressions** (click *F.10.10*). There will be **as many rows** as **groups** of (retained) **input records** (see below). **Each group** contains the (retained) **input records** that produce the **same results** in **all** the **1** to **k** "**GROUP BY**" **expressions**. Notice that the resulting **arrays** of "**k**" **results** from the "**GROUP BY**" **expressions** are **all distinct**. Notice also that these **result arrays** are **not shown** in the **output** of the **Transform operation** and are just internal for its processing. The "**GROUP BY**" clause of a **Transform** works exactly the same as the "**GROUP BY**" clause (click *F.7.9*) of a **Select**.

The "**n**" **leftmost output fields** of a **Transform** are called the "**SELECT**" **fields**, because they are determined by the "**SELECT**" clause. The **output fields** placed to the **right** of the "**SELECT**" **fields** are called the "**PIVOT**" **fields**, because they are **generated** by converting to ***String*** the **returned values** of the "**PIVOT**" **expression** (click *F.10.11*), subject to the modifications of the optional "**IN**" **list** (click *F.10.12*).

The number of **select columns** "**n**", their **field names**, the **field order** and the **field values** are determined by the "**SELECT**" clause (click *F.10.7*) with the "**GROUP BY**" clause (click *F.10.10*). The **value** of **field** "**j**" from the **row** "**i**" is produced by applying the corresponding "**SELECT**" **expression** "**j**", to **all** the records in the **group** "**i**" of (retained) **input records** corresponding to **this row**. These **groups** of (retained) **input records** are **the same** as the ones indicated in the previous paragraph.

The **number** of "**PIVOT**" **columns** and their **field names** are determined by the **distinct values** returned by the "**PIVOT**" **expression** computed over **all** the (retained) **input records**. The **field order** is (click *F.10.3.3*), left to right, **ascending alphanumeric** by the **value** returned by the "**PIVOT**" **expression**. However, if the optional "**IN**" clause is used, then the **number** of "**PIVOT**" **columns**, the **field names** and the **field order** are **all** determined by the "**IN**" **list** (click *F.10.12*).

The **field value** of "**PIVOT**" **field** "**j**" from the **row** "**i**" is produced by applying the "**TRANSFORM**" **expression**, to **all** the records in the **group** "**(i, j)**" of (retained) **input records** corresponding to **row** "**i**" and **column** "**j**". Each **group** "**(i, j)**" contains the (retained) **input records** that produce the **same results** in **all** the **1** to **k** "**GROUP BY**" **expressions** as the **values** in **row** "**i**" **and the same result** in the "**PIVOT**" **expression** as the **field name** of column "**j**". If there is a **field name collision** with a "**SELECT**" **field**, or the "**IN**" clause is used, this becomes slightly more complex (click *F.10.4.1*).

The order of records is **unknown**, unless the optional "**ORDER BY**" clause (click *F.10.9*) is used. The ordering works the same as the "**ORDER BY**" clause (click *F.7.12*) from the **Select** operator, with just one restriction: the "**ORDER BY**" **expressions must** be a **list** of **any number** of **exactly the same** "**GROUP BY**" **expressions** or the "**PIVOT**" **expression**.

The "**Input-record-list**" is stated in the "**FROM**" clause. The "**FROM**" clause in the **Transform** operator works exactly the same as the "**FROM**" clause (click *F.7.4*) in the **Select** operators.

If the optional "**WHERE**" clause is used, the **input records** that produce ***True*** in the "**Where-Boolean-expression()**" are **retained**, while the other ones are **discarded**. The "**WHERE**" clause (click *F.10.9*) of a **Transform** operator works exactly the same as the "**WHERE**" clause (click *F.7.7*) of a **Select** operator. If against my advice you use the optional "DISTINCTROW" clause, you may click *F.7.8*.

Very important to highlight that if the **input record-list** is a Table name, no records will be **deleted** from the Table, and the Table remains **unmodified**. This is so because all the SQL **consulting** operators work over an **image** of Table's records, and **not** over the Table records themselves. Remind that the **Transform** operator is a **consulting** SQL operator (i.e., one that can only consult Tables) and it is not a **data-changing** SQL operator. If you want to actually **modify** your Table's records, you may click

"*F.6.2 What are the SQL data-changing operators?*".

If you want to know more about **Transform**, you may click:

- "*F.10.2 What is the input record-list ("FROM" clause) of a Transform?*"
- "*F.10.3 What are the output fields of a Transform?*"
- "*F.10.4 What is the output record-list of a Transform?*"
- "*F.10.5 Can I see an example of a Transform operation?*"
- "*F.10.13 How do the clauses from Transform and Select compare?*"
- "*F.10.14 How do I write a correct (syntax) Transform?*"

If you want to know the SQL **color codes** used in this **Lightning Guide**, you may click "*F.11.2 What are the SQL color codes used in this Guide?*".

F.10.2 What is the input record-list ("FROM" clause) of a Transform?

In a **Transform** operation (click *F.10.1*), the mandatory "**FROM**" clause indicates its **input record-list**, as follows:

```
FROM {    [[(] {Table-name or Query-name} [)]                               ]
       or [     {Table-name or Query-name}     [AS Input-record-list-name] ]
       or [ (   {Select-opr or Union-opr }   ) [AS Input-record-list-name] ]
       or [[(]   Inner-or-Outer-Join-opr   [)] or Cross-Join-opr            ] }
```

The **input record-list** is indicated in its "**FROM**" clause, **exactly the same** as the one from the **Select** operation.

If you want to write (syntax) a correct "**FROM**" clause, you may click *F.10.14*.

F.10.3 What are the output fields of a Transform?

You may click:

- "*F.10.3.1 What is the number of output fields of a Transform?*"
- "*F.10.3.2 What are the output field names of a Transform?*"
- "*F.10.3.3 What is the output field order of a Transform?*"
- "*F.10.3.4 What are the output data/field types of a Transform?*"
- "*F.10.4 What is the output record-list of a Transform?*"

F.10.3.1 What is the number of output fields of a Transform?

In a **Transform** operation (click *F.10.1*), the **number** of **output fields** is indicated in the "**SELECT**", "**PIVOT**" and "**IN**" clauses as follows:

```
  TRANSFORM PIVOT-field-values-exp()
  SELECT [DISTINCT] [TOP int [PERCENT]] [DISTINCTROW or ALL]
         {    * or
              Output-exp_1(exp-elements) [AS Output-field-name_1]
          [,  ...
           ,  Output-exp_n(exp-elements) [AS Output-field-name_n] ] }
   ...
  PIVOT PIVOT-field-names-exp(Input-field-names)
 [IN ( List-of-PIVOT-values ) ]
```

The "**n**" **leftmost output fields** of a **Transform** are called the "**SELECT**" **fields**, because they are determined by the "**SELECT**" clause. The **output fields** placed to the **right** of the "**SELECT**" **fields** are called the "**PIVOT**" **fields**, because they are **generated** by converting to ***String*** the **returned values** of the "**PIVOT**" **expression** (click *F.10.11*), subject to the modifications of the optional "**IN**" **list** (click *F.10.12*).

What is the number of output "SELECT" fields of a Transform?

The **number** "**n**" of "**SELECT**" **fields** is the **number** of "**SELECT**" **expressions**. This is **exactly the same** as in a **Select** operation (click *F.7.5*).

What is the number of output "PIVOT" fields of a Transform?

If the optional "**IN**" clause is **not** used, the **number** of "**PIVOT**" **fields** is the **number** of **distinct values** returned by the "**PIVOT**" **expression** computed over **all** the (retained) **input records**.

Else, if the optional "**IN**" clause **is used**, then the **number** of "**PIVOT**" **fields** is the **number** of **values** in the "**IN**" **list**.

F.10.3.2 What are the output field names of a Transform?

In a **Transform** operation (click *F.10.1*), the **output field names** are indicated in the "**SELECT**", "**PIVOT**" and "**IN**" clauses as follows:

```
   SELECT [DISTINCT] [TOP int [PERCENT]] [DISTINCTROW or ALL]
          {    * or
               Output-exp_1(exp-elements) [AS Output-field-name_1]
           [,  ...
            ,  Output-exp_n(exp-elements) [AS Output-field-name_n] ] }
    ...
   PIVOT PIVOT-field-names-exp(Input-field-names)
  [IN ( List-of-PIVOT-values ) ]
```

The "**n**" **leftmost output fields** of a **Transform** are called the "**SELECT**" **fields**, because they are determined by the "**SELECT**" clause. The **output fields** placed to the **right** of the "**SELECT**" **fields** are called the "**PIVOT**" **fields**, because they are **generated** by converting to ***String*** the **returned values** of the "**PIVOT**" **expression** (click *F.10.11*), subject to the modifications of the optional "**IN**" **list** (click *F.10.12*).

What are the names of output "SELECT" fields of a Transform?

The **names** of the "**n**" "**SELECT**" **fields** are the identifiers "**Output-field-name_i**" from the "**SELECT**" clause. This is **exactly the same** as in a **Select** operation (click *F.7.5*).

What are the names of output "PIVOT" fields of a Transform?

Regarding the **names** of the "**PIVOT**" **fields**, we have the following three cases:

- The "**IN**" clause is **not** used, **and** the **distinct value** (converted to ***String***) returned by the "**PIVOT**" **expression** is **different** from **all** the "**SELECT**" **field names**. Then, the **name** of this "**PIVOT**" **field** is the **distinct value** (converted to ***String***) returned by the "**PIVOT**" **expression**.
- The "**IN**" clause **is used**, **and** the **value** (converted to ***String***) from the "**IN**" **list** is

different from **all** the "**SELECT**" **field names**.
Then, the **name** of this "**PIVOT**" **field** is the **value** (converted to ***String***) from "**IN**" **list**.

- The **value** (converted to ***String***), which is either a **distinct value** returned by the "**PIVOT**" **expression** or a **value** from the "**IN**" **list**, is the **same as** one of the "**SELECT**" **field names**.
  Then, the **name** of this "**PIVOT**" **field** is "**FieldN**", where "**N**" is an integer value assigned by MS-Access.

The data type of the "**PIVOT**" **expression** is **usually *String***, but it can also be any other data type. In case the data type of the **expression is not *String***, then MS-Access will **convert its result** to a ***String*** (to become a **field name**). The values **True/Yes/On** or **ticked** and **False/No/Off** or **unticked** are converted to the **field names** "**-1**" and "**0**", respectively. An **integer-like** data type value is converted to the **field name** of the equivalent ***String*** (e.g., "**-12**", "**14**"). A ***Date*** value is converted to the **field name** of the equivalent ***String*** (e.g., "**04/05/2003**"). A fractional data type is converted to the **field name** of the equivalent ***String***, but, replacing the period "**.**" with underscore "**_**". The reason for this is that names **cannot** contain a period "**.**" (click *D.2.5*). Therefore, the number "**-0.45**" is converted to the **field name** "**-0_45**".

In case that one or more **input records** make the expression in the "**PIVOT**" clause produce **Null**, these **Nulls** will produce a **valid** "**PIVOT**" **field name** which is "**<>**". I strongly recommend that you **avoid** using the string "**<>**" in the "**List-of-PIVOT-values**" of the "**IN**" clause, to avoid confusion with a "**PIVOT**" field arising from a **Null** (which is most likely not intentional).

A **Null**, either returned by the "**PIVOT**" **expression** or written in the "**IN**" **list**, will produce "**<>**" as the "**PIVOT**" **field name**. If both a **Null** and an explicit "**<>**" happen, the field corresponding to **Null** will be named "**<>**", while the field corresponding to "**<>**" will be named "**FieldN**", where "**N**" is an integer value assigned by MS-Access.

Field names **cannot** include the period "**.**" character (click *D.2.5*). For this reason, every period "**.**" character in a **value** from the "**PIVOT**" **expression** or in a **value** from the "**IN**" **list**, will be converted to the underscore "**_**" character in the resulting "**PIVOT**" **field name**. For example, the returned **values** numeric "**3.4**" and string "**Oh.No**" from the "**PIVOT**" **expression** produce the "**PIVOT**" **field names** "**3_4**" and "**Oh_No**" respectively; likewise, the **values** "**54.6**" and "**Hi.there**" found in the "**IN**" **list** produce the "**PIVOT**" **field name** "**54_6**" and "**Hi.there**", respectively.

If you use the "**IN**" clause, the **data type** returned by the "**PIVOT**" **expression must** be the same as the one of **all** the elements in the "**IN**" **list**, either **directly** or through **type conversion**. Otherwise, the Query will **crash**. Curiously, a **Null**, either **returned** by the "**PIVOT**" **expression** or explicitly **written** in "**IN**" **list**, is always considered as having a compatible data type. I now present a few examples to clarify this paragraph:

- The following works, because "**Cal_Year**" is a **number** and **all** the elements in the "**IN**" **list** are **numbers**, or **can be** converted to a **number**, or are **Null**.

```
PIVOT Cal_Year
IN (2016, 2020, "2018", "34", Null)
```

- The following **crashes**, because "**`Cal_Year`**" is a **number** and the **string** "**`Hello`**" **cannot** be converted to a **number**.

  ```
  PIVOT Cal_Year
  IN (2016, 2020, "2018", "Hello", Null)
  ```

- The following works, because "**`Date_Time`**" is a ***Date/Time*** and **all** the elements in the "**`IN`**" list are either ***Date/Time***, or **can be** converted to a ***Date/Time***, or are **Null**.

  ```
  PIVOT Date_Time
  IN (#2018-1-1#, "3/January/2020", Null)
  ```

- The following **crashes**, because "**`Date_Time`**" is a ***Date/Time*** and the **string** "**`Hello`**" **cannot** be converted to a **number**.

  ```
  PIVOT Date_Time
  IN (#2018-1-1#, "Hello", Null)
  ```

F.10.3.3 What is the output field order of a Transform?

In a **Transform** operation (click *F.10.1*), the **output field order** is indicated in the "**`SELECT`**", "**`PIVOT`**" and "**`IN`**" clauses as follows:

```
SELECT [DISTINCT] [TOP int [PERCENT]] [DISTINCTROW or ALL]
       {    * or
            Output-exp_1(exp-elements) [AS Output-field-name_1]
         [, ...
          , Output-exp_n(exp-elements) [AS Output-field-name_n] ] }
 ...
PIVOT PIVOT-field-names-exp(Input-field-names)
[IN ( List-of-PIVOT-values ) ]
```

The "**n**" **leftmost output fields** of a **Transform** are called the "**`SELECT`**" **fields**, because they are determined by the "**`SELECT`**" clause. The **output fields** placed to the **right** of the "**`SELECT`**" **fields** are called the "**`PIVOT`**" **fields**, because they are **generated** by converting to ***String*** the **returned values** of the "**`PIVOT`**" **expression** (click *F.10.11*), subject to the modifications of the optional "**`IN`**" list (click *F.10.12*).

What is the order of output "`SELECT`" fields of a Transform?

The **order** of the "**n**" "**`SELECT`**" **fields** is the same as the one of the "**`SELECT`**" **expressions**. This is **exactly the same** as in a **Select** operation (click *F.7.5*).

What is the order of output "`PIVOT`" fields of a Transform?

If the optional "**`IN`**" clause is **not** used, the **order** of "**`PIVOT`**" **fields** is, left to right, **ascending** by the **value** produced by the "**`PIVOT`**" **expression**. Notice that if the "**`PIVOT`**" **expression** produces a **numeric value**, the **field** ordering will be ascending **numeric**, while if it produces a ***String*** **value** the field ordering will be ascending **alphabetical**, which is **different**. You can easily change between both by enclosing the "**`PIVOT`**" **expression** in a type conversion function (click *G.2.5*). If a **Null** is produced, it will **always** be the **first** (leftmost) "**`PIVOT`**" **field**, with **field name** "**<>**".

Else, if the optional "**`IN`**" clause **is used**, then the "**`PIVOT`**" **field order** is the one you wrote in the "**`IN`**" list (click *F.10.12*).

Notice that the field order just described is **maintained** even when a "**`PIVOT`**" **field name** (either converted from the "**`PIVOT`**" **expression** or from the "**`IN`**" list) is **the**

same as a "**SELECT**" **field name**. In this case, the "**PIVOT**" field name is changed to "FieldN", but its **position** within the "**PIVOT**" fields is **not** changed.

The "**PIVOT**" field name corresponding to a **Null** value (i.e., the field name "<>") can **also** be reordered by including it in the "**IN**" list. I strongly recommend that you **never use** the string "**<>**" neither as a "**SELECT**" **field name**, nor as a name in the "**IN**" list, nor as a name produced by the "**PIVOT**" expression. If you use "**<>**" in either of these three cases, you create the risk of **mistaking** that field with a "**PIVOT**" field arising from **Null** (which is most likely not intentional).

F.10.3.4 What are the output data/field types of a Transform?

In a **Transform** operation (click *F.10.1*), the **output** data/field **types** are indicated in the "**TRANSFORM**", "**SELECT**", "**PIVOT**" and "**IN**" clauses as follows:

```
   TRANSFORM PIVOT-field-values-exp()
   SELECT [DISTINCT] [TOP int [PERCENT]] [DISTINCTROW or ALL]
          {    * or
               Output-exp_1(exp-elements) [AS Output-field-name_1]
            [, ...
             , Output-exp_n(exp-elements) [AS Output-field-name_n] ] }
    ...
   PIVOT PIVOT-field-names-exp(Input-field-names)
  [IN ( List-of-PIVOT-values ) ]
```

The "**n**" **leftmost output fields** of a **Transform** are called the "**SELECT**" **fields**, because they are determined by the "**SELECT**" clause. The **output fields** placed to the right of the "**SELECT**" **fields** are called the "**PIVOT**" fields, because they are **generated** by converting to ***String*** the returned values of the "**PIVOT**" expression (click *F.10.11*), subject to the modifications of the optional "**IN**" list (click *F.10.12*).

What is the data/field type of output "SELECT" fields of a Transform?

The **data/field type** of the "**n**" "**SELECT**" **fields** are the ones of the corresponding "**SELECT**" **expressions**. This is **exactly the same** as in a **Select** operation (click *F.7.5*).

What is the data/field type of output "PIVOT" fields of a Transform?

Regarding the data/field type of the "**PIVOT**" fields, we have the following three cases:

- The "**IN**" clause is **not** used, **and** the **distinct** value (converted to ***String***) returned by the "**PIVOT**" expression is **different** from **all** the "**SELECT**" **field names**.
  Then, the **data type** of this "**PIVOT**" field is the one of the "**TRANSFORM**" **expression** "**PIVOT-field-values-exp()**".

- The "**IN**" clause **is used**, **and** the value (converted to ***String***) from the "**IN**" list is **different** from **all** the "**SELECT**" **field names**.
  Then, the **data type** of this "**PIVOT**" field is **unknown** (and all this field's values are **Null**).

- The value (converted to ***String***), which is either a **distinct** value returned by the "**PIVOT**" expression or a value from the "**IN**" list, is the **same as** one of the "**SELECT**" **field names**.
  Then, the **data type** of this "**PIVOT**" field is the one of the **same-name** "**SELECT**" **field**.

If you want to know what are the **output** field **values** and/or what is the **output record-list** of a **Transform**, you may click "*F.10.4 What is the output record-list of a Transform?*".

F.10.4 What is the output record-list of a Transform?

You may click:

- "*F.10.4.1 What are the output field values of a Transform?*"
- "*F.10.4.2 What are the output records of a Transform?*"
- "*F.10.4.3 How many output records does a Transform produce?*"

F.10.4.1 What are the output field values of a Transform?

In a **Transform**, the **output field values** are determined by the "**TRANSFORM**", "**SELECT**", "**PIVOT**" and "**IN**" clauses as follows:

```
TRANSFORM
 PIVOT-field-values-exp     (  Output-field-names
                             , PIVOT-field-names-exp(Input-field-names)
                             , Group_by-exp_1(Input-field-names)
                             , ...
                             , Group_by-exp_k(Input-field-names)
                             , SQL_agg_func(exp_t11(INOUT-field-names))
                             , ...
                             , SQL_agg_func(exp_t1d(INOUT-field-names))
                            ) [AS Output-values-Identifier]

 SELECT [DISTINCT] [DISTINCTROW or ALL]
                Output-exp_1(  Output-field-names
                             , PIVOT-field-names-exp(Input-field-names)
                             , Group_by-exp_1(Input-field-names)
                             , ...
                             , Group_by-exp_k(Input-field-names)
                             , SQL_agg_func(exp_o11(INOUT-field-names))
                             , ...
                             , SQL_agg_func(exp_o1y(INOUT-field-names))
                            ) [ AS Output-field-name_1 ]
              [ , ...
                , Output-exp_n(  Output-field-names
                             , PIVOT-field-names-exp(Input-field-names)
                             , Group_by-exp_1(Input-field-names)
                             , ...
                             , Group_by-exp_k(Input-field-names)
                             , SQL_agg_func(exp_on1(INOUT-field-names))
                             , ...
                             , SQL_agg_func(exp_onz(INOUT-field-names))
                            ) [ AS Output-field-name_n ] ]
   ...
   PIVOT PIVOT-field-names-exp(Input-field-names)
  [IN ( List-of-PIVOT-values ) ]
```

The "**n**" **leftmost output fields** of a **Transform** are called the "**SELECT**" **fields**, because they are determined by the "**SELECT**" clause. The **output fields** placed to the **right** of the "**SELECT**" **fields** are called the "**PIVOT**" **fields**, because they are **generated** by converting to ***String*** the **returned values** of the "**PIVOT**" **expression** (click *F.10.11*), subject to the modifications of the optional "**IN**" **list** (click *F.10.12*).

What are the values of output "SELECT" fields of a Transform?

The **values** of the "**n**" "**SELECT**" **fields** are the result of the "**SELECT**" **expressions**, computed over the groups of (retained) **input records** produced by the "`GROUP BY`" **expressions**. This is the same as in a **Select-group_by_aggreg** operation (click *F.7.6.2*), with only one difference. The one difference is that in a **Transform**, the "`SELECT`" **expressions 1** to **n** can **also** use as an **element** the "`PIVOT`" **expression**. You may see this in the SQL code above. This is somehow **surprising**, because the "`PIVOT`" expression produces **multiple** values (**different** in the general case) when computed over the (retained) **input records** in the group corresponding to a given **output** record (remind that we are using a "`GROUP BY`" clause). The way this works is that when evaluating a "`SELECT`" **expression** that includes the "`PIVOT`" expression, it returns the result **as if** the "`PIVOT`" expression was **enclosed** in the "`Min()`" SQL aggregate function. This is, whenever you write the the "`PIVOT`" expression "`PIVOT-field-names-expression()`" within a "`SELECT`" **expression**, it works **as if** you had written "`Min(PIVOT-field-names-expression())`".

Notice that if a "`PIVOT`" field is discarded because the "`IN`" clause is used, and the corresponding result from the "`PIVOT`" expression is not in the "`IN`" list, the **result** of using the "`PIVOT`" expression within the "`SELECT`" **expression** will remain unaffected. This is, you will still get the "`Min()`" value over **all** the values produced, even if the "`Min()`" value is **not** in the "`IN`" list.

Finally, as a rather strange case, if you use the "`PIVOT`" expression as an **element** of a "`SELECT`" **expression**, and you also use the "`IN`" clause, the corresponding "`SELECT`" **expression** will produce the exception value "**#Error**". Curiously, if the "`SELECT`" **expression** is **exactly the same** as the "`PIVOT`" expression, this works fine. This may be an MS-Access **bug**.

What are the values of output "`PIVOT`" fields of a Transform?

Regarding the **values** of the "`PIVOT`" fields, we have the following three cases:

- The "`IN`" clause is **not** used, **and** the **distinct** value (converted to ***String***) returned by the "`PIVOT`" expression is **different** from **all** the "`SELECT`" **field names**.
  Then, the **value** of this "`PIVOT`" field is the result of the "`TRANSFORM`" **expression** computed over the groups of (retained) **input records** produced by the "`GROUP BY`" **expressions** **jointly with** the "`PIVOT`" expression.
  In more detail, the **value** of field "j" and row "i" is produced by applying the "`TRANSFORM`" **expression**, to **all** the records in the group "(i, j)" of (retained) **input records** corresponding to row "i" and column "j". Each group "(i, j)" contains the (retained) **input records** that produce the **same results** in **all** the **1** to **k** "`GROUP BY`" **expressions** as the **values** in row "i" **and the same result** in the "`PIVOT`" expression as the field name of column "j". Since these groups of **input records** depend on the result of the "`PIVOT`" expression, they will be different (in the general case) for **each** "`PIVOT`" field, and therefore, the "`TRANSFORM`" **expression** will produce **different values** (i.e., different columns of **values**) below **each** of the "`PIVOT`" fields.

- The "`IN`" clause **is used**, **and** the value (converted to ***String***) from the "`IN`" list is **different** from **all** the "`SELECT`" **field names**.
  Then, the **value** of this "`PIVOT`" field is **Null** in **all** the **output** records.

- The value (converted to ***String***), which is either a **distinct** value returned by the

"**PIVOT**" **expression** or a **value** from the "**IN**" **list**, is the **same as** one of the "**SELECT**" **field names**.
Then, the **value** of this "**PIVOT**" **field** is **the same as** the one in the **same-name** "**SELECT**" **field** in **each and every output** record. This is, the "**column**" of **values** under this **field** is the same as the "**column**" of **values** under the **same-name** "**SELECT**" **field**.

In the first bullet above, notice that it is **not surprising** to be able to use the "**PIVOT**" **expression** as one of the **elements** of the "**TRANSFORM**" **expression**, because in this case the "**PIVOT**" **expression** produces **the same** **value** in **every** record of **each** **group**. However, in the explanation above in "***What are the values of output "SELECT" fields of a Transform?***" it was surprising that you could use the "**PIVOT**" **expression** as an **element** of the "**SELECT**" **expressions**, because in that case the "**PIVOT**" **expression** **may** produce **different** values in the **different** records of **each** **group**.

Why does the same expression produce different values within the "SELECT" and "TRANSFORM" expressions?

Because in a **Transform**, the "**SELECT**" **expressions** and the "**TRANSFORM**" **expression** are computed over **different** **groups** of **input records**. The "**SELECT**" **expressions** are computed over **groups** of **input records** that produce the same results in **all** the "**GROUP BY**" **expressions**. However, the "**TRANSFORM**" **expression** is computed over **groups** of **input records** that produce the same result in **all** the "**GROUP BY**" **expressions** **and also** in the "**PIVOT**" **expression**.

Let me show this with an example over the Table "**T_Capital_Rainfall_District**".

T_Capital_Rainfall_District			
Capital	**District**	**Cal_Year**	**Rainfall**
Beijing	Dongcheng	2018	14
Beijing	Xicheng	2018	11
Beijing	Dongcheng	2019	18
Beijing	Xicheng	2019	23
Washington	Downtown	2018	10
Washington	Bloomingdale	2018	12
Washington	Downtown	2019	8
Washington	Bloomingdale	2019	9

If you now run the Query[60]:

```
TRANSFORM Sum(Rainfall) AS GenVals
SELECT Capital AS Cap_City, Sum(Rainfall) AS Total
FROM T_Capital_Rainfall_District
GROUP BY Capital
PIVOT Cal_Year ;
```

[60] This is the Query "**F_Transform_sum**" from the "**Company_Database.acccb**" file.

you then get the result:

F_Transform_Sum			
Cap_City	**Total**	**2018**	**2019**
Beijing	32	14	18
Beijing	34	11	23
Washington	18	10	8
Washington	21	12	9

Notice how the "**TRANSFORM**" **expression** "`Sum(Rainfall)`" and the "**SELECT**" **expression** "`Sum(Rainfall)`", that are **exactly the same**, produce **different** results under the field names "**Total**", "**2018**" and "**2019**". The reason is that the "**SELECT**" **expression** "`Sum(Rainfall)`" is computed over the record **groups** produced by the "**GROUP BY**" **expressions**, while **the same** "**TRANSFORM**" **expression** "`Sum(Rainfall)`" is computed over the record **groups** produced by the "**GROUP BY**" **expressions jointly with** the "**PIVOT**" **expression**. In other words, the "**SELECT**" **expression** "`Sum(Rainfall)`" is the **sum** for each "`Capital`" (i.e., adding **all districts** and **all years**) while the "**TRANSFORM**" **expression** "`Sum(Rainfall)`" is the **sum** for each "`Capital`" **and** each "`Cal_Year`" (i.e., adding **all districts**). Notice how the "**TRANSFORM**" **expression** produces a **different column** of **values** for each "**PIVOT**" **field** (i.e., for "**2018**" and "**2019**").

Since in this case we are using the "`Sum()`" aggregate function, the result under "**Total**" is the addition of the results under "**2018**" and "**2019**". However, notice that this **does not** happen with some other SQL aggregate functions (see the last bullet point of *F.10.5*).

F.10.4.2 What are the output records of a Transform?

In a **Transform**, the **output records** are determined by the optional "**WHERE**" clause, the "**GROUP BY**" clause and the optional "**ORDER BY**" clause as follows:

```
[WHERE Where-Boolean-exp(Input-field-names) ]
 GROUP BY      Group_by-exp_1(Input-field-names)
           [ , ...
             , Group_by-exp_k(Input-field-names) ]

[ORDER BY      Group_by-exp_x(Input-field-names)  [DESC]
           [ , ...
             , Group_by-exp_y(Input-field-names)  [DESC] ] ]
```

Transform produces **as many records** as **groups** of (retained) **input records** from the "**GROUP BY**" **expressions**. **Each group** contains the (retained) **input records** that produce the **same results** in **all** the "**GROUP BY**" **expressions**. Notice that the resulting **field value arrays** from "`Group_by-expression()`" **1** to **k** are **all distinct**. Notice also that these resulting **field value arrays** are **not shown** in the result of the **Transform operation**.

For each such **output record** its field **values** will be the ones indicated in the previous subsection *F.10.4.1*.

The order of records is **unknown**, unless the optional "**ORDER BY**" clause (click *F.10.12*) is used. The ordering works the same as the "**ORDER BY**" clause (click *F.7.12*) from the **Select** operator, with just one restriction: the expressions in the **Transform** "**ORDER BY**" clause **must** be a **list** of **any number** of **exactly the same** "**GROUP BY**" **expressions**.

F.10.4.3 How many output records does a Transform produce?

Knowing how many records a **Transform** produces is **very** useful when **debugging** your Queries.

A **Transform** produces **as many records** as **groups** of (retained) **input records** are produced by the "**GROUP BY**" **expressions**. **Each group** contains the (retained) **input records** that produce the **same results** in **all** the "**GROUP BY**" **expressions**.

Notice that the **number** of **output** records is determined only by the (retained) **input records** (i.e., the "**WHERE**" ***Boolean* expression**) and the "**GROUP BY**" **expressions**.

F.10.5 Can I see an example of a Transform operation?

A simple but quite complete example of a **Transform operation**[61] is:

```
TRANSFORM StDev(Temp_Max) AS GenVals
SELECT  Capital AS Cap_City, Cal_Year AS C_Year
      , StDev(Temp_Max) AS StDev_T_Max_Year
FROM T_Capital_temps
GROUP BY Capital, Cal_Year
PIVOT Quart
```

The Table "**T_Capital_Temps**" used in this example has the following structure and values:

T_Capital_Temps					
City	**District**	**Cal_Year**	**Quart**	**Temp_max**	**Temp_min**
Beijing	Dongcheng	2018	Q1	12.3	0
Beijing	Dongcheng	2018	Q2	4	1
Beijing	Dongcheng	2018	Q3	7.8	6.7
Beijing	Dongcheng	2018	Q4	17	15
Beijing	Xicheng	2018	Q1	2.26	-3.25
Beijing	Xicheng	2018	Q2	5.6	-4.5
Beijing	Xicheng	2018	Q3	30	25
Beijing	Xicheng	2018	Q4	18	13
Brasilia	Asa_Norte	2019	Q1	7.4	-0.96
Brasilia	Asa_Norte	2019	Q2	7.57	-1.05
Brasilia	Asa_Norte	2019	Q3	17.5	7.4
Brasilia	Asa_Norte	2019	Q4	10.04	2.35
Brasilia	Guara_I	2019	Q1	10.62	3.17
Brasilia	Guara_I	2019	Q2	11.43	4.21
Brasilia	Guara_I	2019	Q3	12.4	5.52
Brasilia	Guara_I	2019	Q4	11.12	3.81

[61] This is the Query "**F_Transform_examp**" from the "**Company_Database.accdb**" file.

T_Capital_Temps					
City	**District**	**Cal_Year**	**Quart**	**Temp_max**	**Temp_min**
Washington	Anacostia	2018	Q1	6.67	-0.57
Washington	Anacostia	2018	Q2	7.55	-1.24
Washington	Anacostia	2018	Q3	7.65	-1.07
Washington	Anacostia	2018	Q4	7.68	-0.95
Washington	Downtown	2018	Q1	12.13	-9.55
Washington	Downtown	2018	Q2	5.67	-3.25
Washington	Downtown	2018	Q3	2.26	-4.3
Washington	Downtown	2018	Q4	17.57	-2.23
Washington	Bloomingdale	2018	Q1	3.15	2.13
Washington	Bloomingdale	2018	Q2	7.16	-1.92
Washington	Bloomingdale	2018	Q3	7.53	-1.58
Washington	Bloomingdale	2018	Q4	8.85	-0.9

Then, the **Transform** operation above would produce the following **output** record-list:

F_Transform_examp						
Cap_City	**C_Year**	**StDev_T_Max_Year**	**Q1**	**Q2**	**Q3**	**Q4**
Beijing	2018	9.28	7.10	1.13	15.70	0.71
Brasilia	2019	3.17	2.28	2.73	3.61	0.76
Washington	2018	3.98	4.52	0.99	3.08	5.40

Let me explain in more detail why are you getting this **output** record-list.

<u>What is</u> the input record-list of this example?

The **input record-list** is the Table "`T_Capital_Temps`" in the "**`FROM`**" clause.

Since in this example there is no "**`WHERE`**" clause, the **retained input records** are **<u>all</u>** the records in the Table "`T_Capital_Temps`".

<u>What is</u> the output record-list of this example?

The **output** record-list of is determined by the "**`GROUP BY`**" clause. The "**`GROUP BY`**" **expressions** are "`Capital`" and "`Cal_Year`". Therefore, the **output** records correspond to the **distinct values** produced by the (retained) **input records** in the "**`GROUP BY`**" **expressions** "`Capital`" and "`Cal_Year`". If you check the Table "`T_Capital_Temps`" you will see that there are **three** distinct **arrays** of values for "`Capital`" and "`Cal_Year`":

Capital	Cal_Year
Beijing	2018
Brasilia	2019
Washington	2018

<u>What are</u> the output field names of this example?

The **names** of the "**`SELECT`**" **fields** are "`Cap_City`", "`C_Year`" and

"StDev_T_Max_Year", as indicated in the "SELECT" clause.

The **names** of the "PIVOT" **fields** are "Q1", "Q2", "Q3" and "Q4", because these are the **distinct** results of the "PIVOT" **expression** "Quart", when computed over **all** the (retained) **input records**.

What is the output field order of this example?

The "SELECT" **fields** are produced in that order because this is how they appear in the "SELECT" clause.

The "PIVOT" **fields** are produced in that order because the data type of the "PIVOT" **expression** is ***Strings***, and ***Strings*** are ordered in **alphabetical** order.

What are the output field values of this example?

The **values** of the "SELECT" **fields** are the result of the "SELECT" **expressions** "Capital", "Cal_Year" and "StDev(Temp_Max)" computed over the **groups** of (retained) **input records** produced by the "GROUP BY" **expressions**.

The (retained) **input records** in each **group** are the ones that produce the same values in the "GROUP BY" **expressions**. In this example, there are three **groups**, each characterized by the **results** "(**Beijing**, **2018**)", "(**Brasilia**, **2019**)" and "(**Washington**, **2018**)". You may check that the **values** of the "SELECT" **fields** above correspond to the explanation I have just given.

The **values** of the "PIVOT" **fields** are the result of the "TRANSFORM" **expression** "StDev(Temp_max)" computed over the **groups** of (retained) **input records** produced by the "GROUP BY" **expressions** **jointly with** the "PIVOT" **expression**.

In more detail, the **value** of **field** "**j**" from the **output record** "**i**" is produced by applying the "TRANSFORM" **expression** "Avg(Temp_max)", to **all** the records in the **group** "**(i, j)**" of (retained) **input records** corresponding to **record** "**i**" and **field** "**j**". Each **group** "**(i, j)**" contains the (retained) **input records** that produce the **same results** in **all** the **1** to **k** "GROUP BY" **expressions** as the **values** in **row** "**i**" **and the same result** in the "PIVOT" **expression** as the **field name** of column "**j**".

I want to point out **two relevant** aspects from this example, that apply to **all Transform** operations:

- Notice that the "SELECT" **expression** "StDev(Temp_Max)" is **exactly the same** as the "TRANSFORM" **expression** "StDev(Temp_Max)". However, being the same expression it produces **different results** because it is computed over **different groups** of (retained) **input records**. The "SELECT" **expression** is computed over **groups** produced by the "GROUP BY" **expressions**, while the "TRANSFORM" **expression** is computed over **groups** produced the "GROUP BY" **expressions** **jointly with** the "PIVOT" **expression**.

- Notice that the field **values** of "StDev_T_Max_Year" are **not** the result of SQL aggregate function "StDev()" over the **values** of **fields** "Q1", "Q2", "Q3" and "Q4", in **each row**. If they were computed like that, the result would be: "**7.00**", "**1.19**" and "**1.93**, which is different from the actual **output** of the Query: "**9.28**", "**3.17**" and "**3.98**".

The reason is that the field **values** of "`StDev_T_Max_Year`" are the result of the "`SELECT`" **expression** "`StDev(Temp_Max)`" computed over the **groups** of (retained) **input records** produced by the "`GROUP BY`" **expressions**. In this example, this means computing the function "`StDev()`" over **all** the values of "`Temp_Max`" for each "`Capital`" and each "`Cal_Year`".

F.10.6 What is the "TRANSFORM" clause of a Transform?

In a **Transform** operation (click *F.10.1*), the mandatory "**TRANSFORM**" clause determines the "**PIVOT**" data/field **types** (click *F.10.3.4*) and the "**PIVOT**" **field values** (click *F.10.4.1*), as follows:

```
TRANSFORM PIVOT-field-values-expression()
```

If you want to write (syntax) a correct "**TRANSFORM**" clause, you may click *F.10.14*.

F.10.7 What is the "SELECT" clause of a Transform?

In a **Transform** operation (click *F.10.1*), the mandatory "**SELECT**" clause determines the "**SELECT**" **fields** (click *F.10.3*), the "**SELECT**" **field names** (click *F.10.3.2*), the "**SELECT**" **field order** (click *F.10.3.3*), the "**SELECT**" data/field **types** (click *F.10.3.4*) and the "**SELECT**" **field values** (click *F.10.4.1*), as follows:

```
SELECT    Output-expression_1() [ AS Output-field-name_1 ]
       [ , ...
         , Output-expression_n() [ AS Output-field-name_n ] ]
```

If you want to write (syntax) a correct "**SELECT**" clause, you may click *F.10.14*.

F.10.8 What is the "ORDER BY" clause of a Transform?

In a **Transform** operation (click *F.10.1*), the optional "**ORDER BY**" clause determines the **order** of the **output** records, as follows:

```
ORDER BY [    Group_by-exp_x(Input-field-names)  [DESC]
           ,  ...
           ,  Group_by-exp_y(Input-field-names)  [DESC] ]
         [[,] PIVOT-field-names-exp(Input-field-names)  [DESC]]
```

Each "**ORDER BY**" **expression** must be **exactly the same** as one of the "**GROUP BY**" **expressions** or as the "**PIVOT**" **expression**, which is a very strong restriction.

Aside from this restriction on the **Transform** "**ORDER BY**" **expressions**, the "**ORDER BY**" of a **Transform** works exactly the same as the "**ORDER BY**" of a **Select**: you may click "*F.7.12 How do I use "ORDER BY" to order the output records of a Select?*".

If you want to write (syntax) a correct "**ORDER BY**" clause, you may click *F.10.14*.

F.10.9 What is the "WHERE" clause of a Transform?

In a **Transform** operation (click *F.10.1*), the optional "**WHERE**" clause indicates what are the **retained input records**, as follows:

```
WHERE Where-Boolean-expression(Input-field-names)
```

The "**WHERE**" ***Boolean* expression** is built using the "`Input-field-names`" combining them with functions (excluding SQL aggregate), value operators and constants.

The "**WHERE**" clause of a **Transform** works **exactly the same** as the "**WHERE**" clause of a **Select**. You may see "*F.7.7 What is the "WHERE" clause of a Select?*".

If you want to write (syntax) a correct "**WHERE**" clause, you may click *F.10.14*.

F.10.10 What is the "GROUP BY" clause of a Transform?

In a **Transform** operation (click *F.10.1*), the mandatory "**GROUP BY**" clause determines the **output records** (click *F.10.4.2*), as follows:

```
GROUP BY      Group_by-expression_1(Input-field-names)
          [ , ...
            , Group_by-expression_k(Input-field-names) ]
```

Each "**GROUP BY**" **expression** is built using the "**Input-field-names**" combining them with functions (excluding SQL aggregate), value operators and constants.

The "**GROUP BY**" of a **Transform** works **exactly the same** as the "**GROUP BY**" of a **Select**. You may click "*F.7.9 What is the "GROUP BY" clause of a Select-group_by_aggreg?*".

There is also a very subtle, but very interesting, difference: the "**GROUP BY**" clause is optional in a **Select**, but it is mandatory in a **Transform**. This is because you can only do a cross table if there is **only one value** for each **row** and **column**. If you had **several values** for **each row** and **column**, there would be **no criterion** to choose one of them to be displayed. Therefore, the **Transform** operation **guarantees** that there is **only one value** for each **row** and **column**. The way to **guarantee** this is producing the cross table **values** from record **aggregation** of the **rows** and **columns**. Remind that record aggregation produces **only one value** from **each group** of records.

If you want to write (syntax) a correct "**GROUP BY**" clause, you may click *F.10.14*.

F.10.11 What is the "PIVOT" clause of a Transform?

In a **Transform** operation (click *F.10.1*), the mandatory "**PIVOT**" clause determines the "**PIVOT**" **fields** (click *F.10.3*), the "**PIVOT**" **field names** (click *F.10.3.2*) and the "**PIVOT**" **field order** (click *F.10.3.3*), as follows:

```
PIVOT PIVOT-field-names-expression(Input-field-names)
```

The "**PIVOT**" clause contains the **expression** that produces the "**PIVOT**" **field names** (click *F.10.3.2*), **unless** the optional "**IN**" clause (click *F.10.12*) is used. If the optional "**IN**" clause is used, then the "**PIVOT**" **field names** are produced by the **list** of **values** in the "**IN**" clause.

The "**PIVOT**" **expression** is built using the "**Input-field-names**" combining them with functions (excluding SQL aggregate), value operators and constants.

If you want to write (syntax) a correct "**PIVOT**" clause, you may click *F.10.14*.

F.10.12 What is the "IN" clause of a Transform?

In a **Transform** operation (click *F.10.1*), the optional "**IN**" clause determines the "**PIVOT**" **fields**, the "**PIVOT**" **field names** (click *F.10.3.2*) and the "**PIVOT**" **field order** (click *F.10.3.3*), as follows:

```
IN ( List-of-PIVOT-values )
```

The "**IN**" keyword is followed by a **list** of **constants**. I will call this **list** the "**IN**" **list**. The "**IN**" **list** is enclosed between parentheses and the **constants** are separated with commas.

The "**IN**" clause is most frequently used to specify the **order** of the "**PIVOT**" **fields** and/or to **discard** some of them. The way to do this is just by writing in the "**IN**" **list** the "**PIVOT**" **field names** that you want (i.e., excluding some if you do not want them), and writing them in the **specific order** that you want them.

In the general case, when the "**IN**" clause is used, the "**PIVOT**" **fields**, their **field names** and their **field order** will be **exactly the ones** and with the **same order** as they appear in the "**IN**" clause (except if they are the same as a "**SELECT**" **field name**). I now explain the possible cases in more detail:

- If the "**IN**" **list does not** include a **value** that is produced by the "**PIVOT**" **expression**, then the "**PIVOT**" **field** corresponding to that **value does not** appear in the **output** of the **Transform**.
- If the "**IN**" **list includes** a **value** that is produced by the "**PIVOT**" **expression**, **and** that **value** (converted to ***String***) is **not** one of the "**SELECT**" **field names**, then the corresponding "**PIVOT**" **field** will be in the **output** of the **Transform**, in the order indicated in the "**IN**" **list**, with the corresponding **field values** produced by the "**TRANSFORM**" **expression**.
- If the "**IN**" **list includes** a **value** that is **not** produced by the "**PIVOT**" **expression**, **and** that **value** (converted to ***String***) is **not** one of the "**SELECT**" **field names**, then that **value** (converted to ***String***) will be a "**PIVOT**" **field**, in the order indicated in the "**PIVOT**" **list**, with all its values being **Null**.
- If the "**IN**" **list includes** a **value** that (converted to ***String***) **is one of** the "**SELECT**" **field names**, then a "**PIVOT**" **field** will be produced with the name "**FieldN**", where "**N**" is an integer value assigned by MS-Access. The order of that field is the one of the corresponding **value** in the "**IN**" **list**. The **values** of this field will be exactly the same ones as the ones of "**SELECT**" **field** whose **name** was the same.

The "**PIVOT**" **field name** corresponding to a **Null** value (i.e., "**PIVOT**" **field name** "**<>**") can also be reordered by including it in the "**IN**" clause. I strongly recommend that you **do not use** a "**PIVOT**" **expression** that produces the string "**<>**", to avoid confusion with a "**PIVOT**" **field name** arising from **Null** (which most likely will be unintentional).

The "**IN**" **list cannot** contain **duplicated** values: if it does, the Query will **crash** with a syntax error message.

As I indicated at the beginning of this section, the "**IN**" clause is most frequently used to specify the **order** of the "**PIVOT**" **fields** and/or to **discard** some of them. As I have explained just above, the "**IN**" clause can **also** be used to add **copies** of "**SELECT**" **fields** and/or to **create** **new** "**PIVOT**" **fields**. However, the two features in the previous sentence do not seem very useful to me, because the added fields would have either replicated **values** or **Null** (respectively), in **all** the **output** records.

If you want to write (syntax) a correct "**IN**" clause, you may click *F.10.14*.

F.10.13 How do the clauses from Transform and Select compare?

To better understand **Select** and **Transform**, I think it is useful to compare the characteristics of the clauses of a **Select** operation and of a **Transform** operation:

- In the **Transform** (mandatory) "**SELECT**" clause, each "**SELECT**" **expression** may include as its **elements** the "**PIVOT**" **expression**, in addition to the **elements** in the (mandatory) "**SELECT**" clause of a **Select-group_by_aggreg**. This allows to use the "**PIVOT**" **field names** in the "**SELECT**" **expressions**.
  Remind that in a **Select-group_by_aggreg**, the "**SELECT**" **expressions** can contain **other** "**Output-field-names**" (as long as you **do not** create a **circular** reference, click *F.7.14*), any number of "**GROUP BY**" **expressions**, and any number of **SQL aggregate functions** each having as argument its specific **expression** over the "**Input-field-names**" and **other** "**Output-field-names**" (as long as you **do not** create a **circular** reference, **nor** a **nested** SQL aggregate function, click *F.7.14*).
- The **Transform** (optional) "**DISTINCT**" clause does not produce any effect in the results of the **Transform** operation, while in the **Select** operation the (optional) "**DISTINCT**" clause removes **all** the redundant **output** duplicate records.
- The **Transform** (optional) "**ORDER BY**" clause can only use as its **expressions** the "**GROUP BY**" **expressions** (over the **Input-field-names**) or **the** "**PIVOT**" **expression** (over the **Input-field-names**). Aside from this, its functionality is the same as the (optional) "**GROUP BY**" clause from a **Select**.
- **Transform** does **not** have the (optional) "**HAVING**" and "**TOP**" clauses, that the **Select** operator has.
- **Select** does **not** have the (mandatory) "**TRANSFORM**" and "**PIVOT**" clauses nor the (optional) "**IN**" clause, that the **Transform** operator has.
- Both **Transform** and **Select** have the (**not advisable**) (optional) "**DISTINCTROW**" clause.

F.10.14 How do I write a correct (syntax) Transform?

You may click:

- "*F.10.14.1 What is a syntax-example of a Transform?*"
- "*F.10.14.2 What are the formal rules (syntax) to write a Transform?*"
- "*F.10.14.3 Can I nest Transform operations?*"