



A Blog Article by

Geek Must Have

Crib Sheets

John HR Schuster

Version 2.1b, 02/06/2020

Table of Contents

1. MSAccess	1
1.1. Substring (MID)	1
1.2. Date Difference (DateDif)	1
1.3. Format values	2
1.4. Check if folder or file exists	4
1.5. Copy file	5
1.6. Hide second column in Combo box	7
1.7. Refresh Subform with new QueryDef SQL	7
1.8. Convert expression to data type (C)	7
1.9. Replace	7
1.10. SQL LIKE	8
1.11. C data conversion	8
1.12. Creating User Properties (Tables / Queries)	9
1.13. Turn off datasheet Sort/Filter	9
1.14. Determine the Current Object	9
1.15. Update LastUpdated and UpdatedBy	10
1.16. Determine Current subroutine	10
1.17. Requery a form from another form	10
1.18. Relinking table VBA	10
1.19. Sort Hidden properties of datasheet	11
1.20. Change Hide / Show of a field on datasheet	13
1.21. Select Case (If-Then-Else)	13
1.22. Access columns of Combo control	14
1.23. Set Filter on Subform from Main Form	14
1.24. Reset query Column Order setting	14
1.25. Open Dynaset, change SQL	15
1.26. Open recordset, add row	15
1.27. Select Cased Statement (VBA)	16
1.28. Coalesce Function	16
1.29. IsNull	16
2. MySQL	17
2.1. Like with Join	17
2.2. Update with Where	17
2.3. Update with Join	17
2.4. Count Distinct	17
2.5. Count nulls	18
2.6. Add a Auto Increment column to a table	18
2.7. Case (Select)	18

2.8. DBA Display all columns of Tables in DB	18
3. SQL Server	19
3.1. Concat	19
3.2. Cast Number into string	19
3.3. Round	19
3.4. Find string within a string	20
3.5. Create Table Select	20
3.6. IF then else	20
3.7. Immediate If (IIF)	21
3.8. Variables	21
3.9. Substring (Substr)	21
3.10. Update with a Join	22
3.11. Charindex (InStr)	22
3.12. Len (Length) of string	22
3.13. Copy Table	23
3.14. Parameters in SP	23
3.15. CharIndex (Substring)	24
3.16. Declare (Variables)	24
3.17. Case then	24
4. SQLite	25
4.1. DSNLess ODBC connection	25
4.2. Get last ID of a table	27
4.3. Copy Text Box to Clipboard	27
5. Powershell	27
5.1. Extract x Top rows of text file	27
6. Document History	28

These are crib notes. Every time I look something up more than once I try to add it to this document. Something simple.

The PDF version of this document is available at this [Link](#)

1. MSAccess

1.1. Substring (MID)

The Microsoft Access Mid function extracts a substring from a string (starting at any position).

```
Mid ( text, start_position, [number_of_characters] )
```

<https://www.techonthenet.com/access/functions/string/mid.php>

1.2. Date Difference (DateDiff)

The DateDiff() function returns the difference between two dates.

DateDiff

```
DateDiff(datepart, date1, date2, firstdayofweek, firstweekofyear)
```

```
'*--- Return the difference between two dates, in years:  
SELECT DateDiff("yyyy", #13/01/1998#, #09/05/2017#);
```

Parameter	Description
datepart	Required. The part to return. Can be one of the following values: * yyyy = Year . q = Quarter . m = month . y = Day of the year . d = Day . w = Weekday . ww = Week . h = hour . n = Minute . s = Second

date1 and date2 Required. The two dates to calculate the difference between firstdayofweek Optional. Specifies the first day of the week. Can be one of the following values: 0 = Use the NLS API setting 1 = Sunday (this is default) 2 = Monday 3 = Tuesday 4 = Wednesday 5 = Thursday 6 = Friday 7 = Saturday firstdayofyear Optional. Specifies the first week of the year. Can be one of the following values: 0 = Use the NLS API setting 1 = Use the first week that includes Jan 1st (default) 2 = Use the first week in the year that has at least 4 days 3 = Use the first full week of the year

https://www.w3schools.com/sql/func_msaccess_datediff.asp

Format	Explanation
--------	-------------

General Date	<p>Displays date based on your system settings</p> <p>Long Date Displays date based on your system's long date setting</p> <p>Medium Date Displays date based on your system's medium date setting</p> <p>Short Date Displays date based on your system's short date setting</p> <p>Long Time Displays time based on your system's long time setting</p> <p>Medium Time Displays time based on your system's medium time setting</p> <p>Short Time Displays time based on your system's short time setting</p> <p>https://www.techonthenet.com/access/functions/date/format.php</p>
--------------	---

1.3. Format values

The Format function does exactly the same thing as formatting a number or a date within a cell in a spreadsheet, except it does so from within the code itself. If you wish to display a number in a message box or on a user form, this function is very useful for making it readable, particularly if it is a large number

Examples

```
Format(1234567.89, "#,###.#")

format(([On_Hand]-[Qty_Needed]), "#,##0[Black];(#,##0)[Red];0;0")

sReturn = Format(sValueIS, "$###,###,###,##0[Black];($###,###,###,##0)[Red];;")
```

Table 1. Predefined Formats

Format Name	Description
General Number	Display the number as is.
Currency	Display the number with currency symbol. Use thousand separator. Enclose in brackets if negative. Display to two decimal places.
Fixed	Display at least one digit to the left and two digits to the right of the decimal point.
Standard	Display number with thousand separator. Display to two decimal places.
Percent	Display number multiplied by 100 with a percent sign (%) appended after. Display to two decimal places.
Scientific	Use standard scientific notation.
Yes/No	Display No if number is 0; otherwise, display Yes.
True/False	Display False if number is 0; otherwise, display True.
On/Off	Display Off if number is 0; otherwise, display On.

A number of characters can be used to define a user-defined format, as shown below. The format string can have up to four sections separated by semicolons (;). This is so different formats can be

applied to different values, such as to positive and negative numbers. For example, you may wish to show brackets/parentheses around a negative value

Table 2. Section Detail

Number of Sections	Formatting
One section only	Applies to all values
Two sections	First section for positive values, second section for negative values
Three sections	First section for positive values, second section for negative values, third section for zeros
Four sections	First section for positive values, second section for negative values, third section for zeros, fourth section for null values

Table 3. User-Defined Formats

Character	Description
Null String	No formatting.
0	Digit placeholder. Displays a digit or a zero. If there is a digit for that position, then it displays the digit; otherwise, it displays 0. If there are fewer digits than zeros, you will get leading or trailing zeros. If there are more digits after the decimal point than there are zeros, then the number is rounded to the number of decimal places shown by the zeros. If there are more digits before the decimal point than zeros, these will be displayed normally.
#	Digit placeholder. This displays a digit or nothing. It works the same as the preceding zero placeholder, except that leading and trailing zeros are not displayed. For example, 0.75 would be displayed using zero placeholders, but this would be .75 using # placeholders.
.Decimal point.	Only one permitted per format string. This character depends on the settings in the Windows Control Panel.
%	Percentage placeholder. Multiplies number by 100 and places % character where it appears in the format string.
,	Thousand separator. This is used if 0 or # placeholders are used and the format string contains a comma. One comma to the left of the decimal point means to round to the nearest thousand (e.g., 0,). Two adjacent commas to the left of the thousand separator indicate rounding to the nearest million (e.g., 0,,).
E- E+	Scientific format. This displays the number exponentially.
:	Time separator-used when formatting a time to split hours, minutes, and seconds.
/	Date separator-this is used when specifying a format for a date.
- + \$ ()	Displays a literal character. To display a character other than listed here, precede it with a backslash (\).

Table 4. Predefined Date and Time Formats

Format Name	Description
General Date	Display a date and/or time. For real numbers, display date and time. Integer numbers display time only. If there is no integer part, then display only time.
Long Date	Displays a long date as defined in the international settings of the Windows Control Panel.
Medium Date	Displays a date as defined in the short date settings of the Windows Control Panel, except it spells out the month abbreviation.
Short Date	Displays a short date as defined in the International settings of the Windows Control Panel.
Long Time	Displays a long time as defined in the International settings of the Windows Control Panel.
Medium Time	Displays time in a 12-hour format using hours, minutes, and seconds and the AM/PM format.
Short Time	Displays a time using 24-hour format (e.g., 18:10).

<https://sourcedaddy.com/ms-access/format-function.html>

1.4. Check if folder or file exists

```
Function FileExists(ByVal strFile As String, Optional bFindFolders As Boolean) As Boolean
    'Purpose:    Return True if the file exists, even if it is hidden.
    'Arguments: strFile: File name to look for. Current directory searched if no path
    included.
    '
    bFindFolders. If strFile is a folder, FileExists() returns False
    unless this argument is True.
    'Note:      Does not look inside subdirectories for the file.
    'Author:    Allen Browne. http://allenbrowne.com June, 2006.
    Dim lngAttributes As Long

    'Include read-only files, hidden files, system files.
    lngAttributes = (vbReadOnly Or vbHidden Or vbSystem)

    If bFindFolders Then
        lngAttributes = (lngAttributes Or vbDirectory) 'Include folders as well.
    Else
        'Strip any trailing slash, so Dir does not look inside the folder.
        Do While Right$(strFile, 1) = "\"
            strFile = Left$(strFile, Len(strFile) - 1)
        Loop
    End If

    'If Dir() returns something, the file exists.
    On Error Resume Next
    FileExists = (Len(Dir(strFile, lngAttributes)) > 0)
End Function

Function FolderExists(strPath As String) As Boolean
    On Error Resume Next
    FolderExists = ((GetAttr(strPath) And vbDirectory) = vbDirectory)
End Function

Function TrailingSlash(varIn As Variant) As String
    If Len(varIn) > 0 Then
        If Right(varIn, 1) = "\" Then
            TrailingSlash = varIn
        Else
            TrailingSlash = varIn & "\"
        End If
    End If
End Function
```

1.5. Copy file

```
'-----
```



```
--
' Procedure : CopyFile
' Author    : Daniel Pineault, CARDA Consultants Inc.
' Website   : http://www.cardaconsultants.com
' Purpose   : Copy a file
'             Overwrites existing copy without prompting
'             Cannot copy locked files (currently in use)
' Copyright : The following is release as Attribution-ShareAlike 4.0 International
'             (CC BY-SA 4.0) - https://creativecommons.org/licenses/by-sa/4.0/
' Req'd Refs: None required
'
' Input Variables:
' ~~~~~
' sSource - Path/Name of the file to be copied
' sDest - Path/Name for copying the file to
'
' Revision History:
' Rev      Date(yyyy/mm/dd)      Description
'
~~~~~
' 1          2007-Apr-01          Initial Release
'-----
--

Public Function CopyFile(sSource As String, sDest As String) As Boolean
On Error GoTo CopyFile_Error

    FileCopy sSource, sDest
    CopyFile = True
    Exit Function

CopyFile_Error:
    If Err.Number = 0 Then
    ElseIf Err.Number = 70 Then
        MsgBox "The file is currently in use and therefore is locked and cannot be
copied at this" & _
            " time. Please ensure that no one is using the file and try again.",
vbOKOnly, _
            "File Currently in Use"
    ElseIf Err.Number = 53 Then
        MsgBox "The Source File '" & sSource & "' could not be found. Please validate
the" & _
            " location and name of the specifed Source File and try again",
vbOKOnly, _
            "File Currently in Use"
    Else
        MsgBox "MS Access has generated the following error" & vbCrLf & vbCrLf &
"Error Number: " & _
            Err.Number & vbCrLf & "Error Source: CopyFile" & vbCrLf & _
            "Error Description: " & Err.Description, vbCritical, "An Error has
Occurred!"
    End If
End Function
```

Exit Function
End Function

<https://www.devhut.net/2010/09/29/ms-access-vba-copy-a-file/>

1.6. Hide second column in Combo box

In Visual Basic, the ColumnWidth property setting is an Integer value that represents the column width in twips. You can specify a width or use one of the following predefined settings.



When you use a 0 as a ColumnWidth, that columns is not available in vba.

Table 5. Special Column Width Values

Setting	Description
0	Hides the column.
1	(Default) Sizes the column to the default width.

The Alternative method which allows the column to be used is setting the first columns width to something like 5"

1.7. Refresh Subform with new QueryDef SQL

You can't requery, you have to refresh the subform source object:

```
MySubformControl.SourceObject = ""  
MySubformControl.SourceObject = "Query.MyQuery"
```

1.8. Convert expression to data type (C)

The function name determines the return type as shown in the following:

Cast and Convert

```
-- CAST Syntax:  
CAST ( expression AS data_type [ ( length ) ] )  
  
-- CONVERT Syntax:  
CONVERT ( data_type [ ( length ) ] , expression [ , style ] )
```

1.9. Replace

The REPLACE() function replaces all occurrences of a substring within a string, with a new substring.

Function	Return Type	Range for expression argument
CVar	Variant	Same range as Double for numerics. Same range as String for non-numerics.

1.12. Creating User Properties (Tables / Queries)

You can create user-defined properties for persistent DAO objects, such as tables and queries. You can't create properties for nonpersistent objects, such as recordsets. To create a user-defined property, you must first create the property, using the Database's CreateProperty method. You then append the property using the Properties collection's Append method. That's all there is to it.

Using the example of a field's Description property, the following code demonstrates just how easy it is:

```
Public Sub SetFieldDescription(strTableName As String, _
    strFieldName As String, _
    varValue As Variant, _
)
    Dim dbs As DAO.Database
    Dim prop As DAO.Property
    Set dbs = CurrentDb

    'Create the property
    Set prop = dbs.CreateProperty("Description", dbText, varValue)

    'Append the property to the object Properties collection
    dbs(strTableName)(strFieldName).Properties.Append prop
    Debug.Print dbs(strTableName)(strFieldName).Properties("Description")

    'Clean up
    Set prop = Nothing
    Set dbs = Nothing
End Sub
```

<https://sourcedaddy.com/ms-access/setting-and-retrieving-built-in-object-properties.html>

1.13. Turn off datasheet Sort/Filter

On design view go to the properties page. Under the "Other" or "All" tab find Shortcut Menu. Change that property from Yes to No and save. Be warned though that this will disable all shortcuts for the form and not just the drop down filter/sort menus on column headings in datasheet view.

1.14. Determine the Current Object

The CurrentObjectName property is set by Microsoft Access to a string expression containing the name of the active object.

Current Object

```
intCurrentType = Application.CurrentObjectType  
strCurrentName = Application.CurrentObjectName
```

1.15. Update LastUpdated and UpdatedBy

These two fields are used in most my code to show when the row was last touched by someone.

Update Audit

```
Private Sub Form_BeforeUpdate(Cancel As Integer)  
    '*--- TimeStanp any change  
    Me.LastUpdated = Now()  
    Me.UpdatedBy = SetUserName()  
End Sub
```

1.16. Determine Current subroutine

Current sub

```
msgbox Application.VBE.ActiveCodePane.CodeModule)  
'*--- will return something like  
'* Form_frmIMMTemplateImport
```

1.17. Requery a form from another form

```
e.dirty = false  
Forms!frmLegacy.Requery
```

Source: <http://www.utteraccess.com/forum/Requery-Form-Form-t2001669.html>

1.18. Relinking table VBA

```

Function ReLinkTable(strTable As String, strPath As String) As Boolean
' Comments: Re-links the named table to the named path
' Params  : strTable      Table name of the linked table
' strPath : full path name of the database containing the real table
' Returns : True if successful, False otherwise

Dim fOK As Boolean
Dim dbs As DAO.Database
Dim tdf As DAO.TableDef
Dim strPrefix As String
Dim strNewConnect As String

fOK = False

On Error GoTo PROC_ERR

Set dbs = CurrentDb()
Set tdf = dbs.TableDefs(strTable)

strPrefix = Left$(tdf.Connect, InStr(tdf.Connect, "="))
strNewConnect = strPrefix & strPath

tdf.Connect = strNewConnect
tdf.RefreshLink

fOK = True

PROC_EXIT:
dbs.Close
ReLinkTable = fOK
Exit Function

PROC_ERR:
Resume PROC_EXIT
End Function

```

<http://www.fmsinc.com/microsoftaccess/databasesplitter/>

1.19. Sort Hidden properties of datasheet

The properties in Access related to datasheet.

Table 6. Hidden Properties

Property	Meaning and Usage
ColumnHidden	Exists on columns in the datasheet, controls whether the column is visible or not.
ColumnWidth	Exists on columns in the datasheet, controls the width of the column.

Property	Meaning and Usage
DatasheetBackColor	Exists on the datasheet itself, specifies the background color for the whole datasheet.
DatasheetCellsEffect	Exists on the datasheet itself, handles whether special effects are used for the cells (flat, raised, or sunken are the only effects supported).
DatasheetFontHeight	Exists on the datasheet itself, this unfortunately named property specifies the font size.
DatasheetFontItalic	Exists on the datasheet itself, controls whether all of the text is italic.
DatasheetFontName	Exists on the datasheet itself, controls the name of the font.
DatasheetFontUnderline	Exists on the datasheet itself, controls whether all of the text is underlined.
DatasheetFontWeight	Exists on the datasheet itself, controls whether the text is bolded.
DatasheetForeColor	Exists on the datasheet itself, specifies the foreground color for the whole datasheet.
DatasheetGridlinesBehavior	Exists on the datasheet itself, controls which gridlines will be displayed (if any).
DatasheetGridlinesColor	Exists on the datasheet itself, specifies the color of the gridlines.
FrozenColumns	Exists on the datasheet itself, specifies how many columns have been frozen by the user (discussed later in the article).
ShowGrid	Exists on the datasheet itself, but has been superseded by the DatasheetGridlinesBehavior property.
SubdatasheetExpanded	Exists on the datasheet itself, specifies whether all subdatasheets should be expanded. (Access 2000 only)
SubdatasheetHeight	Exists on the datasheet itself, specifies the number of records to display for subdatasheets (a scrollbar appears if there are more records than this property allows). (Access 2000 only)
SubdatasheetName	Exists on the datasheet itself, specifies the name of the table's subdatasheet. (Access 2000 only)
TabularCharSet	Exists on the datasheet itself, and is hidden. It specifies the font character set and can often cause bad things to happen if it's set to an incorrect value. It's best not to set it, or to set it to 1 (which uses the DEFAULT_CHARSET for the machine).

With the exception of the Subdatasheet properties, you have no direct design-time access to these properties: None of these properties show up in the datasheet property sheet. As a result, they can only be set at runtime from VBA code in order to make changes. Interestingly, none of the properties are exposed by ADO or ADOX, so if you want to change them, you'll have to use DAO.

While you can't access these properties through property sheets, many of them can be set in the user interface. They are, for example, what's changed when you set the font of a datasheet from the Format menu. For full control over the datasheet, though, you'll want to explicitly set the properties

in code and save the object when you're done.

It's worth noting that a datasheet is a form it says so right in the object browser. The object browser considers the datasheet columns to be the controls on the form. As a result, a datasheet can consist of any control that can be displayed, which means all TextBox, ComboBox, and CheckBox controls.

Source: [https://docs.microsoft.com/en-us/previous-versions/office/developer/office-2003/aa217449\(v=office.11\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/office/developer/office-2003/aa217449(v=office.11)?redirectedfrom=MSDN)

1.20. Change Hide / Show of a field on datasheet

The code to show and hide the columns is in a routine called ShowHideColumn.

```
Private Function ShowHideColumn()  
    Dim sfrm As SubForm  
    Dim ctl As Control  
    Dim stCtl As String  
  
    Set sfrm = Me.sfrmHideShowColumns  
    For Each ctl In Me.Controls  
        If TypeOf ctl Is Access.CheckBox Then  
            stCtl = "tb" & Mid$(ctl.Name, 3)  
            sfrm.Form(stCtl).ColumnHidden = _  
                Not ctl.Value  
        End If  
    Next ctl  
End Function
```

1.21. Select Case (If-Then-Else)

The Microsoft Access Case statement can only be used in VBA code. It has the functionality of an IF-THEN-ELSE statement.


```
Select Case test_expression
```

```
    Case condition_1
```

```
        result_1
```

```
    Case condition_2
```

```
        result_2
```

```
    ...
```

```
    Case condition_n
```

```
        result_n
```

```
[ Case Else
```

```
    result_else ]
```

```
End Select
```

Source: <https://www.techonthenet.com/access/functions/advanced/case.php>

1.22. Access columns of Combo control

Use 0 to refer to the first column, 1 to refer to the second column, and so on. Use 0 to refer to the first row, 1 to refer to the second row, and so on. For example, in a list box containing a column of customer IDs and a column of customer names, you could refer to the customer name in the second column (1) and fifth (4) row as:

If the user has made no selection when you refer to a column in a combo box or list box, the Column property setting will be Null. You can use the IsNull function to determine if a selection has been made

```
Forms!Contacts!Customers.Column(1, 4)
```

```
'*--- Empty combo selection
```

```
If IsNull(Forms!Customers!Country)
```

```
    Then MsgBox "No selection."
```

```
End If
```

1.23. Set Filter on Subform from Main Form

```
'*--- lstBoxSheets is the subForm
```

```
Me.LstBoxSheets.Form.Filter = "prjCategory='General'"
```

```
Me.LstBoxSheets.Form.FilterOn = True
```

1.24. Reset query Column Order setting

So, when you open a query in Datasheet view, and the column order has not been messed with and saved, the column order displayed is determined by the OrdinalPosition and the value of that

property corresponds to the order in which your columns appear in the query design grid (OrdinalPosition is 0 based, so 0 is the first column).

Then ... when you move the column while viewing the query in Datasheet view, and subsequently save that change in the column order, Access creates the ColumnOrder property for each of the columns in the query. This property is not visible in the query design grid, but is definately there.

```
Public Sub ResetColumnOrder(strQueryName)
    Dim fld As DAO.Field
    Dim qdf As DAO.QueryDef

    Set qdf = CurrentDb.QueryDefs(strQueryName)

    For Each fld In qdf.Fields
        On Error Resume Next
        fld.Properties.Delete "ColumnOrder"
    Next fld

End Sub
```

1.25. Open Dynaset, change SQL

1.26. Open recordset, add row

```
Dim dbCurrent As Database
Dim rsNotes As Recordset
Dim sSQL As String

sSQL = "Select * from PrePos where PostType = 'Help';"
Set dbCurrent = CurrentDb
Set rsNotes = dbCurrent.OpenRecordset(sSQL, dbOpenDynaset, dbSeeChanges)
With rsNotes
    If .EOF Then
        .AddNew
        ![PostIMMTable] = gsNewTableName
        ![PostIMMField] = gsNewFieldName
        ![PostPtype] = gsPtype
        ![Notes] = Me.txtNotes
        ![UpdatedDate] = Now()
        ![UpdatedBy] = gsUserName
        .Update
        .Close
    End If
End With
Set rsNotes = Nothing
Set dbCurrent = Nothing
```

1.27. Select Cased Statement (VBA)

```
Select Case test_expression

    Case condition_1
        result_1
    Case condition_2
        result_2
    ...
    Case condition_n
        result_n

[ Case Else
    result_else ]

End Select
```

1.28. Coalesce Function

Access does not have Coalesce function, this quick VBA equivalent. You pass it an array of values.

Coalesce Function

```
Function Coalesce(ParamArray varValues()) As Variant
'returns the first non null value, similar to SQL Server Coalesce() function
'Patrick Honorez --- www.idevelop.com
    Dim i As Long
    Coalesce = Null
    For i = LBound(varValues) To UBound(varValues)
        If Not IsNull(varValues(i)) Then
            Coalesce = varValues(i)
            Exit Function
        End If
    Next
End Function
```

<https://stackoverflow.com/questions/247858/coalesce-alternative-in-access-sql>

1.29. IsNull

The MS Access IsNull() function returns TRUE (-1) if the expression is a null value, otherwise FALSE (0):

```
SELECT ProductName, UnitPrice * (UnitsInStock + IIF(IsNull(UnitsOnOrder), 0,
UnitsOnOrder))
FROM Products;
```

Source: https://www.w3schools.com/sql/sql_isnull.asp

2. MySQL

2.1. Like with Join

```
SELECT table1.*, table2.z
FROM table1
INNER JOIN table2
    ON table2.name LIKE CONCAT('%', table1.name, '%')
AND table1.year = table2.year
```

2.2. Update with Where

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

2.3. Update with Join

```
UPDATE T1, T2,
[INNER JOIN | LEFT JOIN] T1 ON T1.C1 = T2. C1
SET T1.C2 = T2.C2,
    T2.C3 = expr
WHERE condition
```

<https://www.mysqltutorial.org/mysql-update-join/>

2.4. Count Distinct

You can use the DISTINCT clause within the COUNT function. For example, the SQL statement below returns the number of unique departments where at least one employee makes over \$55,000 / year.

```
SELECT COUNT(DISTINCT department) AS "Unique departments"
FROM employees
WHERE salary > 55000;
```

2.5. Count nulls

```
select sum(case when FirstName IS NULL then 1 else 0 end) as NUMBER_OF_NULL_VALUE from
DemoTable;
```

Source: <https://www.tutorialspoint.com/how-to-count-null-values-in-mysql>

2.6. Add a Auto Increment column to a table

Add Auto Increment

```
ALTER TABLE ThreeSeasons ADD column id INT NOT NULL AUTO_INCREMENT unique first
```

2.7. Case (Select)

The CASE statement goes through conditions and return a value when the first condition is met (like an IF-THEN-ELSE statement). So, once a condition is true, it will stop reading and return the result.

If no conditions are true, it will return the value in the ELSE clause.

If there is no ELSE part and no conditions are true, it returns NULL.

Case

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN "The quantity is greater than 30"
    WHEN Quantity = 30 THEN "The quantity is 30"
    ELSE "The quantity is under 30"
END
FROM OrderDetails;
```

2.8. DBA Display all columns of Tables in DB

```
select
    tab.name as table_name,
    col.column_id,
    col.name as column_name,
    t.name as data_type,
    col.max_length,
    col.precision
from sys.tables as tab
    inner join sys.columns as col
        on tab.object_id = col.object_id
    left join sys.types as t
        on col.user_type_id = t.user_type_id
```

3. SQL Server

3.1. Concat

```
CONCAT(string1, string2, ....., string_n)
```

3.2. Cast Number into string

```
CAST(expression AS datatype(length))
```

Table 7. Parameter Values

Value	Description
expression	Required. The value to convert
datatype	Required. The datatype to convert expression to. Can be one of the following: bigint, int, smallint, tinyint, bit, decimal, numeric, money, smallmoney, float, real, datetime, smalldatetime, char, varchar, text, nchar, nvarchar, ntext, binary, varbinary, or image
(Length)	Optional. The length of the resulting data type (for char, varchar, nchar, nvarchar, binary and varbinary) Technical Details

3.3. Round

Round

```
ROUND(number, decimals, operation)
```

```
-- Round number to nearest decimal  
round(DIHTA.ValueIS, 0, 1)
```

Table 8. Round arguments

Parameter	Description
number	Required. The number to be rounded
decimals	Required. The number of decimal places to round number to
operation	Optional. If 0, it rounds the result to the number of decimal. If another value than 0, it truncates the result to the number of decimals. Default value is 0

3.4. Find string within a string

```
SELECT CHARINDEX('t', 'Customer') AS MatchPosition;
```

3.5. Create Table Select

The SELECT INTO statement copies data from one table into a new table.

```
SELECT *  
INTO newtable [IN externaldb]  
FROM oldtable  
WHERE condition;
```

Practical Example

```
select distinct FieldName  
    into New_DealPathway_B2  
from dbo.Pre_Migration_Guide
```

3.6. IF then else

```
IF Boolean_expression  
    { sql_statement | statement_block }  
[ ELSE  
    { sql_statement | statement_block } ]
```

3.7. Immediate If (IIF)

```
IIF(condition, value_if_true, value_if_false)
```

3.8. Variables

```
-- Declare a variable with a data type
DECLARE @model_year SMALLINT;

-- Set a variable to a value
SET @model_year = 2018;

-- Use variable in query
SELECT
    product_name,
    model_year,
    list_price
FROM
    production.products
WHERE
    model_year = @model_year
ORDER BY
    product_name;

-- Set Variable in query
SELECT
    @product_name = product_name,
    @list_price = list_price
FROM
    production.products
WHERE
    product_id = 100;
```

3.9. Substring (Substr)


```
SELECT
    email,
    SUBSTRING(
        email,
        CHARINDEX('@', email)+1,
        LEN(email)-CHARINDEX('@', email)
    ) domain
FROM
    sales.customers
ORDER BY
    email;
```

3.10. Update with a Join

```
UPDATE
    t1
SET
    t1.c1 = t2.c2,
    t1.c2 = expression,
    ...
FROM
    t1
    [INNER | LEFT] JOIN t2 ON join_predicate
WHERE
    where_predicate;
```

3.11. Charindex (InStr)

```
SELECT
    email,
    SUBSTRING(
        email,
        CHARINDEX('@', email)+1,
        LEN(email)-CHARINDEX('@', email)
    ) domain
FROM
    sales.customers
ORDER BY
    email;
```

3.12. Len (Length) of string

```

SELECT
    email,
    SUBSTRING(
        email,
        CHARINDEX('@', email)+1,
        LEN(email)-CHARINDEX('@', email)
    ) domain
FROM
    sales.customers
ORDER BY
    email;

```

3.13. Copy Table

The SELECT INTO statement copies data from one table into a new table.

```

SELECT *
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;

```

Source: https://www.w3schools.com/sql/sql_select_into.asp

https://www.w3schools.com/sql/sql_select_into.asp

3.14. Parameters in SP

Create a query to repeatedly get the data for different sales people, you could instead parameterize the query and turn it into a stored procedure like:

```

create procedure getSalesperson
@sp varchar(25)
as
select SalesPerson, Mon, amount
from SalesData
where SalesPerson = @sp;
Go

-- Run the SP
declare @sp varchar(25)
set @sp = 'Jack'
exec getSalesperson @sp

```

<https://www.mssqltips.com/sqlservertip/2981/using-parameters-for-sql-server-queries-and-stored-procedures/>

3.15. CharIndex (Substring)

The CHARINDEX() function searches for a substring in a string, and returns the position. If the substring is not found, this function returns 0. Note: This function performs a case-insensitive search.

CharIndex

```
CHARINDEX(substring, string, start)
```

Source: https://www.w3schools.com/sql/func_sqlserver_charindex.asp

3.16. Declare (Variables)

In SQL Server (Transact-SQL), a variable allows a programmer to store data temporarily during the execution of code.

Declare Variable

```
-- Declare the variable to be used.
DECLARE @MyCounter int;

-- Initialize the variable.
SET @MyCounter = 0;

-- Alternative example
DECLARE @techonthenet VARCHAR(50);

SET @techonthenet = 'Example showing how to declare variable';
```

3.17. Case then

The CASE statement goes through conditions and returns a value when the first condition is met (like an IF-THEN-ELSE statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

Case then

```
CASE expression
  WHEN value_1 THEN result_1
  WHEN value_2 THEN result_2
  WHEN value_n THEN result_n
  ELSE result
END

-- Or

CASE
  WHEN condition_1 THEN result_1
  WHEN condition_2 THEN result_2
  WHEN condition_n THEN result_n
  ELSE result
END
```

Source: https://www.w3schools.com/sql/sql_case.asp

Source: https://www.techonthenet.com/sql_server/functions/case.php

4. SQLite

4.1. DSNLess ODBC connection

```
DRIVER=SQLite3 ODBC
Driver;Database=c:\mydb.db;LongNames=0;Timeout=1000;NoTXN=0;SyncPragma=NORMAL;StepAPI=
0;

-- Zortero Example
ODBC;DSN=Zotero;Database=C:\Users\{userDirectory}\Zotero\link_zotero.sqlite;StepAPI=0;
SyncPragma=OFF;NoTXN=0;Timeout=;ShortNames=0;LongNames=0;NoCreat=0;NoWCHAR=0;FKSupport
=0;JournalMode=;OEMCP=0;LoadExt=;BigInt=0;JDConv=0;;TABLE=collections
```

This is some VB code to create the DSNLess Connection

```

'//Name      :   AttachDSNLessTable
'//Purpose   :   Create a linked table to SQL Server without using a DSN
'//Parameters
'//      stLocalTableName: Name of the table that you are creating in the current
database
'//      stRemoteTableName: Name of the table that you are linking to on the SQL Server
database
'//      stServer: Name of the SQL Server that you are linking to
'//      stDatabase: Name of the SQL Server database that you are linking to
'//      stUsername: Name of the SQL Server user who can connect to SQL Server, leave
blank to use a Trusted Connection
'//      stPassword: SQL Server user password
Function AttachDSNLessTable(stLocalTableName As String, stRemoteTableName As String,
stServer As String, stDatabase As String, Optional stUsername As String, Optional
stPassword As String)
    On Error GoTo AttachDSNLessTable_Err
    Dim td As TableDef
    Dim stConnect As String

    For Each td In CurrentDb.TableDefs
        If td.Name = stLocalTableName Then
            CurrentDb.TableDefs.Delete stLocalTableName
        End If
    Next

    If Len(stUsername) = 0 Then
        '//Use trusted authentication if stUsername is not supplied.
        stConnect = "ODBC;DRIVER=SQL Server;SERVER=" & stServer & ";DATABASE=" &
stDatabase & ";Trusted_Connection=Yes"
    Else
        '//WARNING: This will save the username and the password with the linked table
information.
        stConnect = "ODBC;DRIVER=SQL Server;SERVER=" & stServer & ";DATABASE=" &
stDatabase & ";UID=" & stUsername & ";PWD=" & stPassword
    End If
    Set td = CurrentDb.CreateTableDef(stLocalTableName, dbAttachSavePWD,
stRemoteTableName, stConnect)
    CurrentDb.TableDefs.Append td
    AttachDSNLessTable = True
    Exit Function

AttachDSNLessTable_Err:

AttachDSNLessTable = False
    MsgBox "AttachDSNLessTable encountered an unexpected error: " & Err.Description

End Function

```

4.2. Get last ID of a table

Last Record

```
Dim lastID As Integer  
lastID = DMax("IDField", "YourTable")
```

4.3. Copy Text Box to Clipboard

The following example illustrates how to copy the contents of a text box named txtNotes to the Clipboard. .Copy to Clipboard

```
Private Sub cmdCopy_Click()  
    Me!txtNotes.SetFocus  
    DoCmd.RunCommand acCmdCopy  
End Sub
```

<https://docs.microsoft.com/en-us/office/vba/access/concepts/windows-api/send-information-to-the-clipboard>

5. Powershell

5.1. Extract x Top rows of text file

```
get-content input.txt|select-object -first 10 >output.txt
```

<https://stackoverflow.com/questions/28908638/extract-only-the-first-10-lines-of-a-csv-file-in-powershell>

6. Document History

Table 9. Document History

Date	Version	Author	Description
06/14/2020	V2.1d	JHRS	Charindex, isnull, Variables
06/08/2020	V2.1c	JHRS	added MySQL DBVA
06/04/2020	V2.1b	JHRS	Initial version