How Access Stores Queries – Part 2
Design vs SQL View

URL: https://isladogs.co.uk/explaining-queries-2/

The **first part** of this article explained how **Access stores query information using the MSysQueries** system table

In this article, I will explain how **Access retrieves the last saved view (design view/SQL view)** information for future use

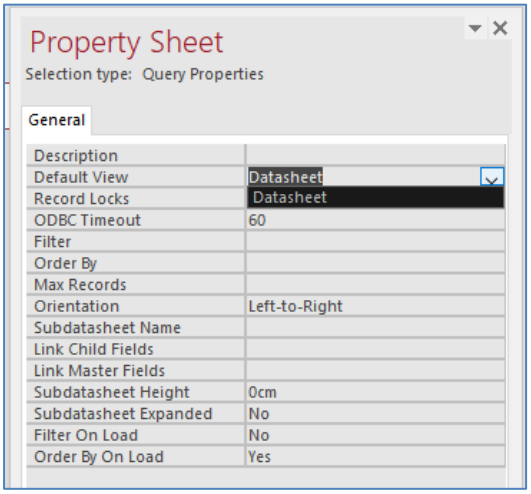
---

1. Introduction

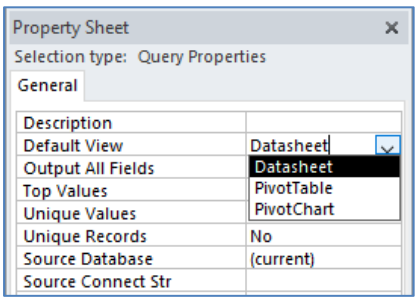
Almost all types of query can be created / edited in **Design View** or **SQL View** according to user preference.
However, **Union / Data Definition / Passthrough** query types are **SQL-specific** i.e. **SQL view ONLY**

When a query is opened, it will **ALWAYS open** in its **last saved view** – **Design or SQL view.**
BUT where does **Access** store this information for the **next time of use**?

The obvious place might appear to be the **query default view property.**
However, **in recent Access versions** (2013 onwards) , there is only **one option – Datasheet view.**



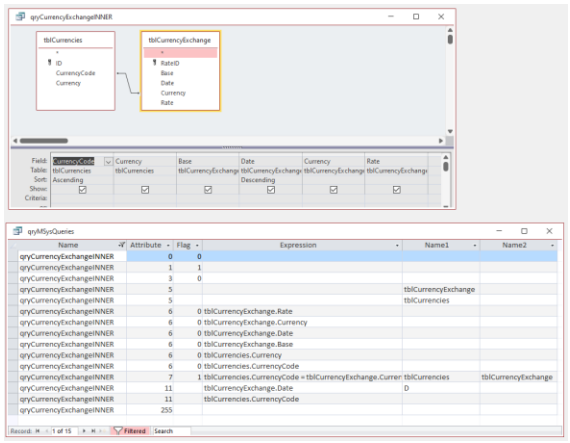
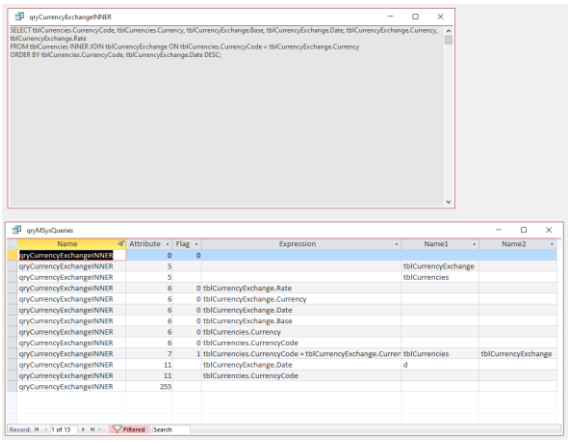
NOTE: Older versions of Access (up to 2010) had two other options – **Pivot Table & Pivot Chart**



However, in **all versions**, the **default view property** refers to the **view used when the query is run.**

The **last saved view** used in **query development** must be **stored somewhere else** . . .
Over the years, I have never seen any explanation of how this is done.

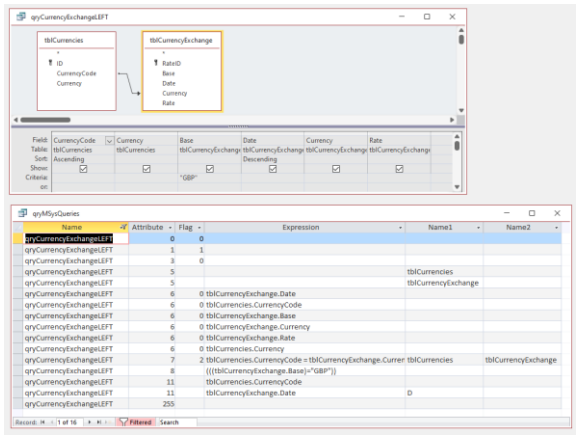
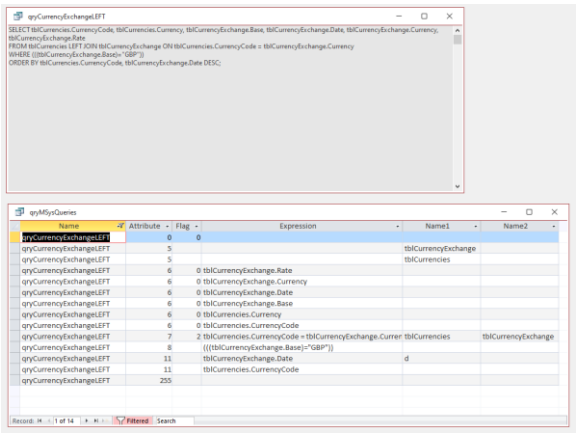
However, I noticed something unexpected when I was researching the **first part** of this article
The **MSysQueries** table data is different depending on whether a **SELECT query** is opened **in design view** or **SQL view.** For example:

Design View	**SQL View**
	

The **SQL view** is identical except it has **2 fewer records** with the **following items OMITTED**:

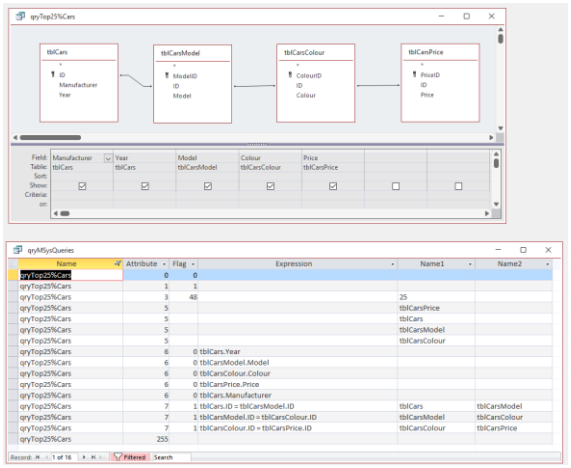
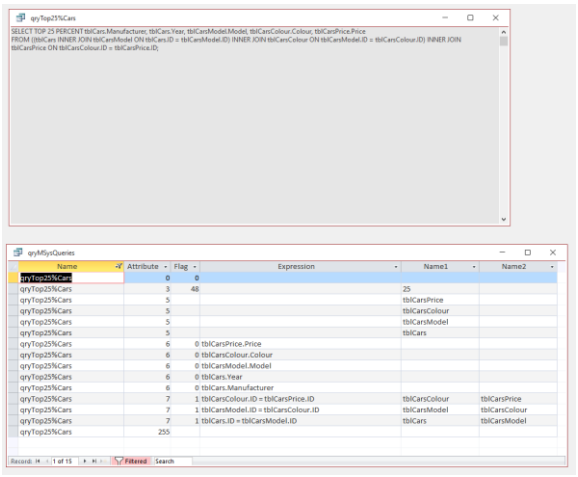
- **Attribute 1 – Flag = 1 (SELECT)**
- **Attribute 3 – Flag = 0 (No special attributes)**

This pattern is repeated for other **SELECT** queries e.g. LEFT join

Design View	**SQL View**
	

However, if the **SELECT** query uses options such as **DISTINCT / TOP / PERCENT / UNION** then **Attribute 3 <> 0**. In such cases, the **MSysQueries** data retains the **Attribute 3** record in **SQL view**.

For example, a **SELECT TOP PERCENT** query

Design View	SQL View
	

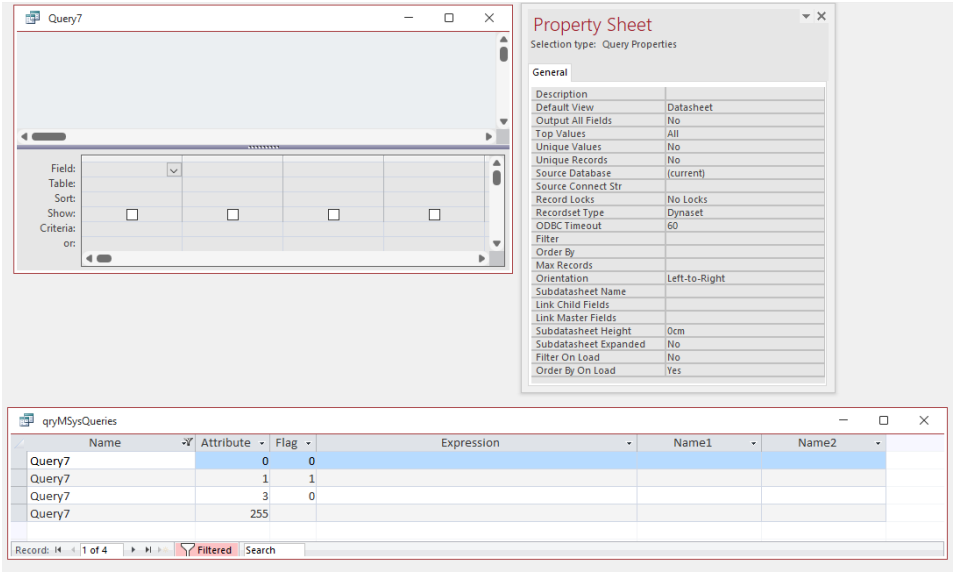
These **differences** seemed an **unlikely explanation** for the **last saved view** information.
Indeed, it appeared to be rather strange behaviour at first sight.
However, I now think it does make sense

To understand why, **create a new query** and go to **SQL view without adding a table**.
The **SQL window** shows this:

Until now, I had always assumed this was done just to help users start writing the query.
However, perhaps that's not the case.

Access won't let you save that **empty query in SQL view** so you can't yet view the **MSysQueries** records.

However, you can go to design view and save that query. The design window will of course be blank

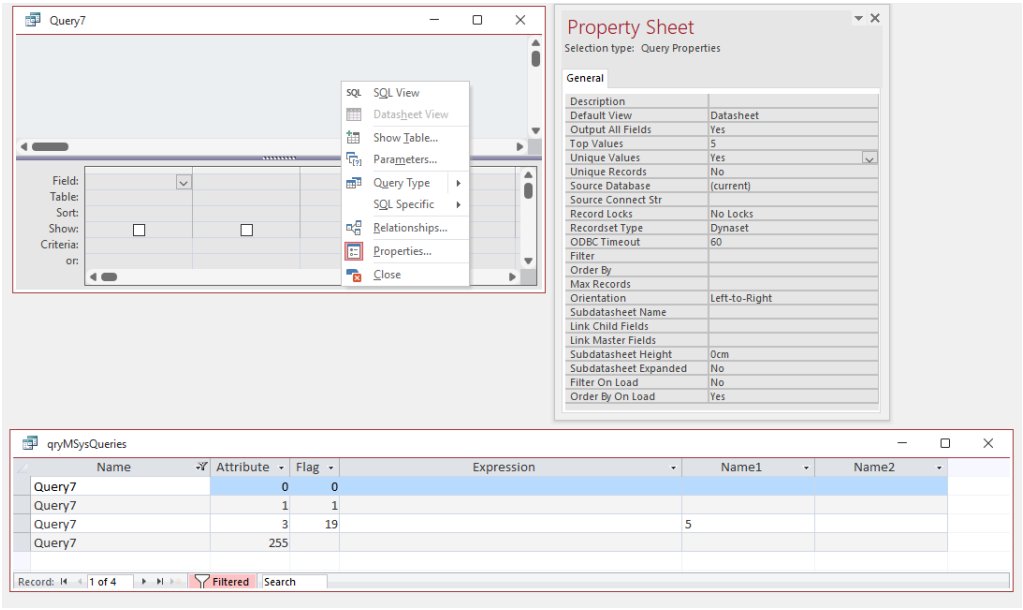


Notice the records for attributes 1 & 3

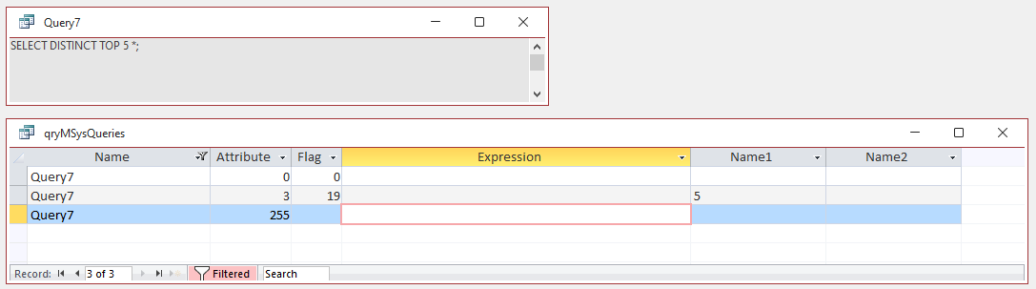
Still in **design view**, alter the **property sheet** as follows:

- **Unique values = Yes**
- **Output All Fields = Yes**
- **Top Values = 5**

The **query design window** will still be **blank** but the **MSysQueries** data has changed:



In **SQL view** we now see this:



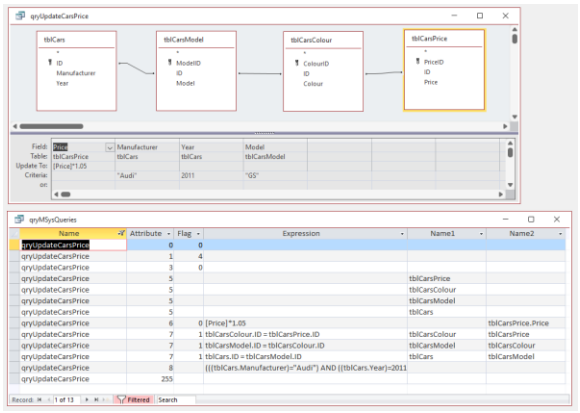
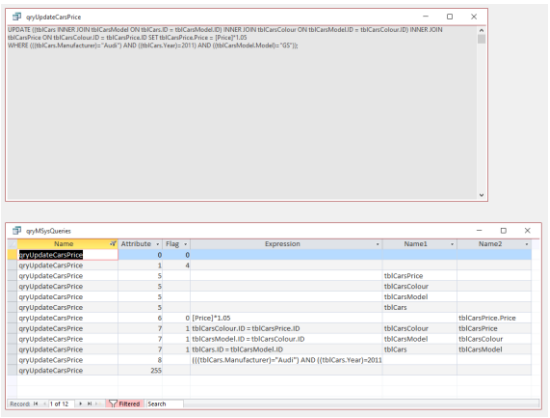
The values entered in the **property sheet** are of course shown in the **SQL window** but once again the record for **Attribute 1 is missing**. In this case, **Attribute 3** is shown as its **Flag is non-zero**

So it would appear that Access omits the Attribute 1 Flag = 1 record as SELECT is added by default in SQL view.

Similarly it **omits Attribute3 Flag = 0** as that **Flag value indicates the default SQL** and doesn't need to be altered

What about other types of query? **APPEND / UPDATE / DELETE / MAKE TABLE**
Attribute 1 will have **Flag >1** and will be shown.
Where **Attribute 3 = 0** it is again omitted.

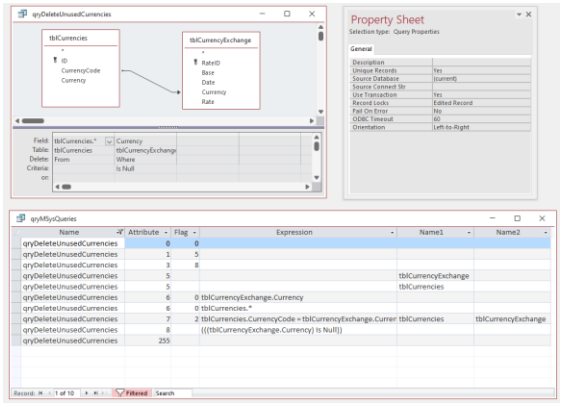
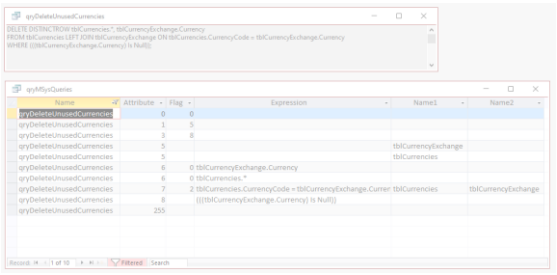
For example, a simple **UPDATE** query:

Design View	**SQL View**
	

Once again, the **MSysQueries** data is different when saved in each of the views

However, for an **action query** which does include options such as **Unique Records = Yes (DISTINCTROW)**, the **MSysQueries** records are **IDENTICAL**.

For example , a **DELETE DISTINCTROW** query:

Design View	SQL View
	

Conclusion:
Access is **NOT using** the **MSysQueries** records to **determine the last saved view**

I've checked the **query definition properties** and there are none which store this information.

Nevertheless, this information must be stored as part of the query itself so the next place to look at are the four **Lv** fields in the **MSysObjects** table: **Lv, LvModule, LvExtra & LvProp**

All 4 fields are **OLE Object** datatype.
Where the fields contain data, **Access** shows this as **Long binary data**

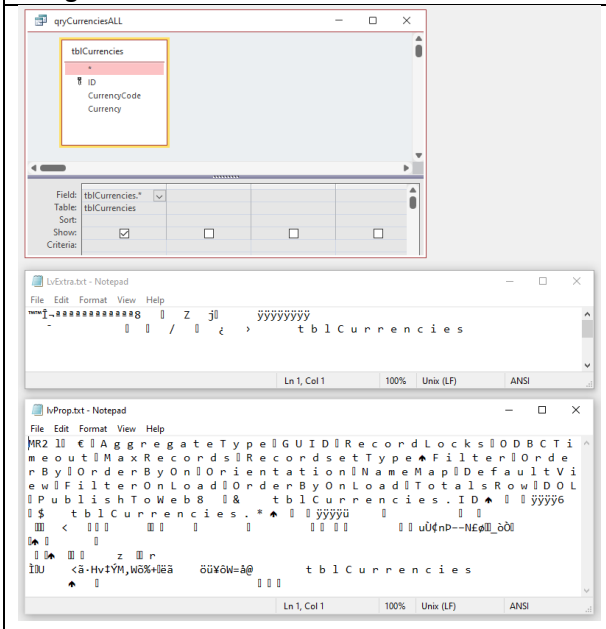
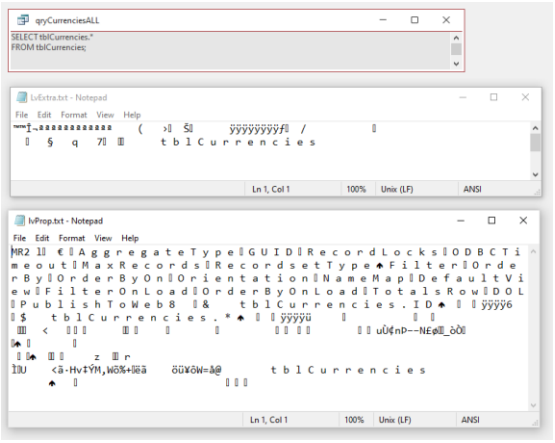
For queries, data is stored in the **LvProp** and **LvExtra** fields ONLY

qryMSysQueryObjects

Name	Type	Flags	Lv	LvExtra	LvModule	LvProp
qryAppendCars	5	64		Long binary data		Long binary data
qryAppendValues	5	64		Long binary data		Long binary data
qryCarColoursCount	5	0		Long binary data		Long binary data
qryCars	5	0		Long binary data		Long binary data
qryCarsCrosstab	5	16		Long binary data		Long binary data
qryCarsFiltered	5	0		Long binary data		Long binary data
qryCurrenciesALL	5	0		Long binary data		Long binary data
qryCurrencyExchangeINNER	5	0		Long binary data		Long binary data
qryCurrencyExchangeINNERFiltered	5	0		Long binary data		Long binary data
qryCurrencyExchangeLEFT	5	0		Long binary data		Long binary data
qryDeleteUnusedCurrencies	5	32		Long binary data		Long binary data
qryDistinctVehicles	5	0		Long binary data		Long binary data
qryMemo	5	0		Long binary data		Long binary data
qryMeterReadingsNonEquiJoin	5	0		Long binary data		Long binary data
qryMSysQueries	5	0		Long binary data		Long binary data
qryMSysQueryObjects	5	0		Long binary data		Long binary data
qryParameters	5	0		Long binary data		Long binary data
qryTop25%Cars	5	0		Long binary data		Long binary data
qryUpdateCarsPrice	5	48		Long binary data		Long binary data

Record: 1 of 26 No Filter Search

It is possible to view this **long binary data** e.g. by **exporting** the data to **text files**
The screenshots below show the contents of the **LvExtra** & **LvProp** fields for a very simple **SELECT** query saved in **design view** and again in **SQL view**

Design View	SQL View
	

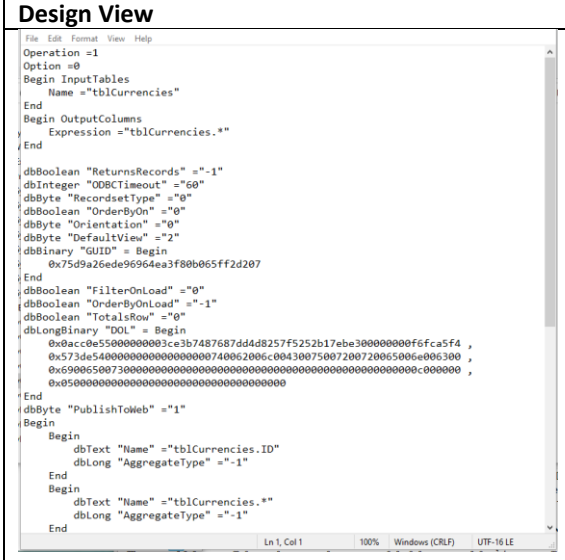
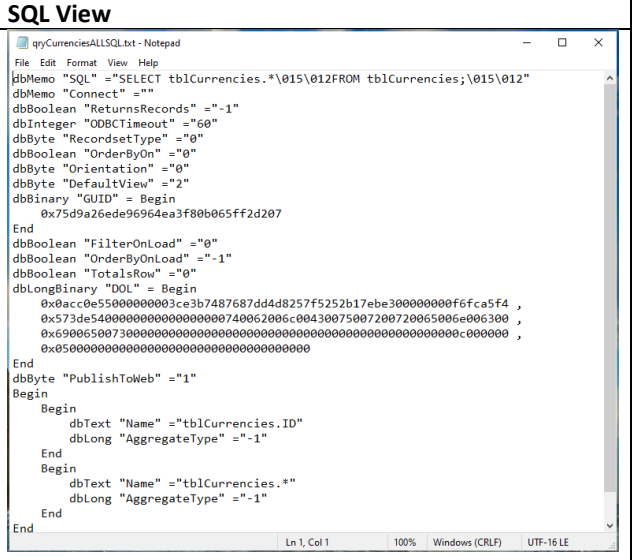
Even for a simple query, the information is hard to read.
However, there do appear to be differences in the **LvExtra** field

More complex queries contain additional information in these fields and are even harder to decipher.

However, there is an easier approach. We can use **Application.SaveAsText** to save the **entire query** to a **text file** in both **design & SQL views**

The text file output can be quite long but the differences between the two query views are obvious from the first line

For example, using the same simple SELECT query as above:

Design View	SQL View
	

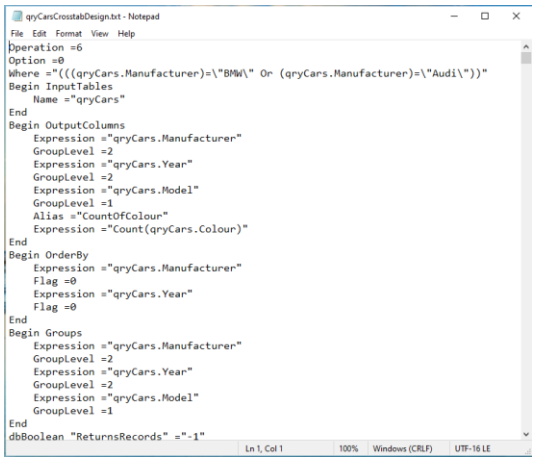
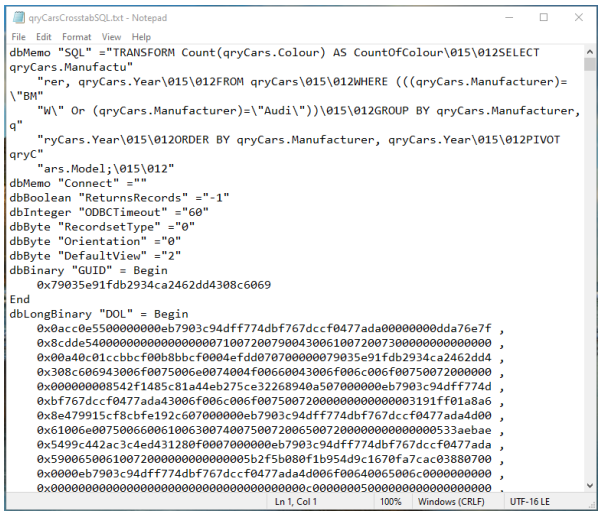
In **design view**, the **first 2 lines** ALWAYS correspond to the **Flag values** for **Attributes 1 and 3**

- **Operation = 1** corresponds to **Attribute 1 Flag = 1 (SELECT)**
- **Option = 0** corresponds to **Attribute 3 Flag = 0 (no special options)**

The rest of the file contains info about . . .

In **SQL view,** the **first item** ALWAYS starts with **dbMemo (long text datatype)** followed by the **query SQL**
The rest of the file contains info about . . .

Here's another example. This time for a **crosstab query:**

Design View	**SQL View**
	

Conclusion
To get the **last saved view** we just need to read the **first line** of the **query** when **output to a text file.**

The screenshot shows a form listing **all queries** together with the **query type**, **last saved view** and **last saved date** information

The form is based on this query:

SELECT MSysObjects.Name AS QueryName, tblSysObjectTypes.SubType AS QueryType, GetQueryLastSavedView([Name]) AS LastSavedView, GetDateLastUpdated([Name]) AS LastSavedDate
FROM MSysObjects INNER JOIN tblSysObjectTypes ON (MSysObjects.Flags = tblSysObjectTypes.Flags) AND (MSysObjects.Type = tblSysObjectTypes.Type)
WHERE (((MSysObjects.Flags)<>3) AND ((MSysObjects.Type)=5))
ORDER BY MSysObjects.Name;

The query uses the following to obtain this info:

- **QueryType** – obtained from table **tblSysObjectTypes** based on the **Flags** value in **MSysObjects**
- **LastSavedView** – obtained using the **GetQueryLastSavedView** function

  This uses **Application.SaveAsText** to **output the query** to a **text file** (UTF-16 format)
  It is then converted to **ANSI format** so **VBA** can read the first part of the file to identify it as **design view** or **SQL view**

```
Function GetQueryLastSavedView(strQuery As String)
    'save the query as a text file (UTF-16 format)
    Application.SaveAsText acQuery, strQuery, CurrentProject.Path & "\qryUTF.txt"

    'convert the text file to ANSI format so it can be read using VBA
    UTF16toANSI CurrentProject.Path & "\qryUTF.txt", CurrentProject.Path & "\qryANSI.txt"

    'get the first 11 characters of the ANSI text file
    Select Case Left(ReadTextFile(CurrentProject.Path & "\qryANSI.txt"), 11)

    Case "Operation ="
       GetQueryLastSavedView = "Design"
    Case "dbMemo ""SQL"
       GetQueryLastSavedView = "SQL"
    Case Else
       GetQueryLastSavedView = "--"
    End Select
End Function
```

The code for the **UTF16toANSI** procedure used above is in module **modQueryInfo:**

```
Option Compare Database
Option Explicit

Private Const adReadAll = -1
Private Const adSaveCreateOverWrite = 2
Private Const adTypeBinary = 1
Private Const adTypeText = 2
Private Const adWriteChar = 0

 Dim strText As String

‘=========================================

 'Adapted from code at https://stackoverflow.com/questions/5182102/vb6-vbscript-change-file-encoding-to-ansi

Private Sub UTF16toANSI(ByVal UTF16FName, ByVal ANSIFName)

   With CreateObject("ADODB.Stream")
      .Open
      .Type = adTypeBinary
      .LoadFromFile UTF16FName
      .Type = adTypeText
      .Charset = "utf-16"
      strText = .ReadText(adReadAll)
      .Position = 0
      .SetEOS
      .Charset = "_autodetect" 'Use current ANSI codepage.
      .WriteText strText, adWriteChar
      .SaveToFile ANSIFName, adSaveCreateOverWrite
      .Close
   End With
End Sub
```

- **LastSavedDate**- – obtained using the **GetDateLastUpdated** function:

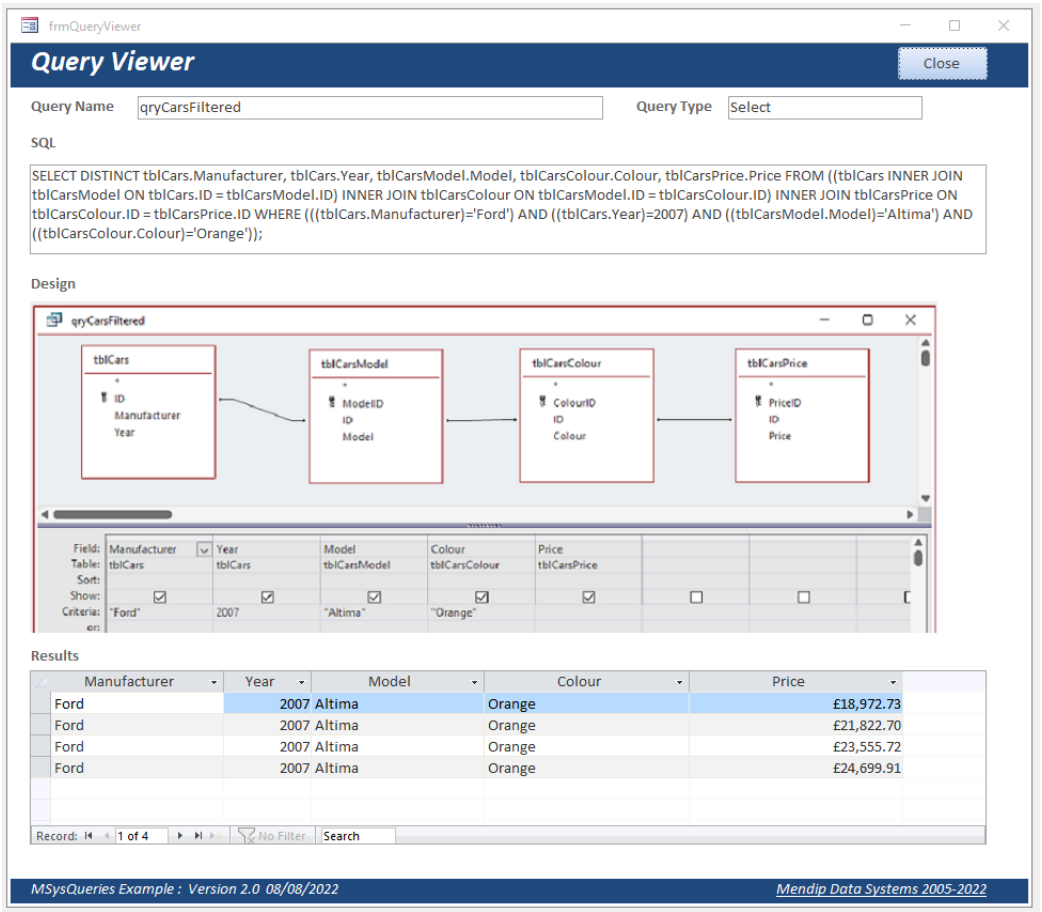
```
Function GetDateLastUpdated(strQuery As String)
    'gets the last updated property as shown in the navigation pane
  GetDateLastUpdated = CurrentDb.QueryDefs(strQuery).Properties("LastUpdated")
End Function
```

As the **data will change over time**, all the above **fields** are obtained at **runtime**
Obtaining all this info took a fraction of a second.

As an experiment I ran the above query on a very large FE database for schools with 1582 queries. It took about 12 seconds to open the query & move to the last record indicating the query had completed

Future Plans:
I hope to use the information in these **two articles** to display the **SQL, design view and results** for a **selected query** on the **same form**

The aim is to create something like this (similar to **SQL Server Management Studio**):



Colin Riddington ***Mendip Data Systems*** ***Last Updated 8 Aug 2022***