```
strGreeting = "Dear someone" & vbNewLine
= = =
wdDoc.Range.InsertBefore strGreeting
Sheet1.Activate
Range("A1").CurrentRegion.Copy
wdDoc.Range (Len(strGreeting),
             Len(strGreeting)).Paste
End With
End Sub
```

---

choosing a specific signature :-

~~Alt~~ outlook → Message → Signature → | → personal
| → work
signatures.

→ No methods to switch signatures

→ so, ~~use~~ open up contains the signature text.

option Explicit

sub sendBasicEmail()

Tools -> References -> ☑ Microsoft Scripting Runtime

Dim olApp As outlook.Application

Dim olEmail As outlook.MailItem

Dim fso As scripting.FilesystemObject

Dim ts As scripting.TextStream

Dim sigpath As String, sigText As String

Views -> Immediate window

? Environ("AppData")

C:\Users\Andrew.Gould\AppData\Roaming

sigpath = Environ("AppData") & _

"\Microsoft\signatures\personal.htm"

set fso = New scripting.FileSystemObject

set ts = fso.OpenTextFile(sigpath)

(or) work.htm

\Microsoft\Outlook\

```
sigText = ts.ReadAll
ts.close

set fso = Nothing
set olAPP = New outlook.Application
set olEmail = olAPP.CreateItem(olMailItem)
with olEmail
  .BodyFormat = olFormatHTML
  .Display
  .HTMLBody = "Dear someone" & "<br>" &
                                  sigText
  .Attachments.Add environ("userprofile") &
    "\Desktop\movie Repost.docx"
  .TO = "sgkuo29u@gmail.com
  subject = "movie repost"
End with
End sub
```

~~Function GetsigText() AS String~~

~~Function GetsigText(~~

View → Object Browser → Outlook →

const
↑ olFormatHTML = 2 ⇒ constant

member of OutlookObject ⇓ Enumeration . olBodyFormat → olFormatHTML
→ olFormatPlain
→ olFormatRichText
→ olFormatUnspecified

Function ~~Greg~~ GetsigText (sigType As olBodyFormat,
Optional sigName As String = "work") As String

Dim fso As scripting.FilesystemObject
Dim ts As scripting.TextStream
Dim sigpath As String, sigText As String,
sigExt As String

```
select case sigType
    case olFormatHTML
        sigExt = ".htm"
    case olFormatRichText
        sigExt = ".rtf"
    case olFormatPlain
        sigExt = ".txt"
end select

sigPath = Environ("Appdata") & "\Microsoft\Signatures\"
          & sigName & sigExt

set fso = New Scripting.FileSystemObject
set ts = fso.OpenTextFile(sigPath, , , TristateMixed)
sigText = ts.ReadAll
ts.Close
```

TristateFalse → Ascii
TristateTrue → Unicode.
TristateMixed
HTML
plain

```
    set fso = Nothing
    GetsigText = sigText
End Function
```

```
Sub sendBasicEmail()
Dim olApp As outlook.Application
Dim olEmail As outlook.mailItem

set olApp = New outlook.Application
set olEmail = olApp.CreateItem(olMailItem)
with olEmail
    .BodyFormat = olFormatHTML
    'Display
    .HTMLBody = "Dear Someone" & "<b><b>" &
            GetsigText(olFormatHTML, "personal")
    .Attachments.Add Environ("Userprofile") &
        "\Desktop\Movie Report.docx"
```

wrong name

"work" runtime error "53"

```
'.To = "someone@somewhere.com"
.To = "sgkwoaqu@gmail.com"
.Subject = "movie report"
End With
End Sub
```

```
Sub sendBasicEmailPlain()
Dim ...
With olEmail
    .BodyFormat = olFormatPlain
    ...
    ' Body = "Dear someone" & vbNewLine &
        GetSigText(.BodyFormat, "personal")
                                    (oo)
                    (.BodyFormat)
    .Attachments ...
...
End Sub
```

Function

```
set fso = New scripting.FilesystemObject
If Not fso.FileExists(sigpath) Then
  sigpath = Environ("APPData") & "\microsoft\
            signatures\work" & sigExt
  End If
set ts = fso.openTextFile(sigpath,,,TristateMixed)
End Function.
```

Query a Database Using ActiveX Data Objects (ADO): (34)

Querying Databases from Excel VBA

- Referencing the ActiveX Data Object Library
- Creating a New Connection
- Setting the Connection String

- Creating a New Recordset
- Loading Records into a Recordset
- Copying Data into Excel
- writing sql queries in VBA
- Getting User input to modify Queries

'Querying a Database from Excel VBA :-

module -> modUsingADO

option Explicit

sub copyDataFromDatabase()

'The ActiveX Data Object library

'Tools -> References -> ☑ Microsoft Active X Data objects 6.1 Library -> OK

'view -> objectBrowser -> ADODB -> classes & constants & etc.....

'Declaring a Variable for a Connection :-

Dim MoviesConn AS ADODB.Connection

' Early Binding

'creating a New Instance of a Connection:-

set MoviesConn = New ADODB.Connection.

'MoviesConn.ConnectionString = " "

End sub

'Auto-Instancing Variables:-

sub copyDataFromDatabaseAutoInstancing()

Dim MoviesconnAs New ADODB.Connection

MoviesConn.ConnectionString = " "

↘ VB editor checks & creates new instance

End sub

Downsides:-

① checks every time it encounters moviesconn. => creates & this if never happens

② If MoviesConn is Nothing Then

End If

Using the CreateObject Function :-

```
sub CopyDataFromDataBaseUsingCreateObject()
'Late Binding
Dim MoviesConn AS object

set MoviesConn = CreateObject("ADODB.Connection")

MoviesConn.connectionstring = ""
        ↓
    No = Intelligence

End sub
```

'setting an Access Connection String :-

Google search → Connection strings →

www.connectionstrings.com
(first one Mostly)

Connect :-
Access ✓

//Access DB created in Access 2007

Copy Microsoft ACE OLEDB 12.0
standard security :- & modify.

```
option Explicit
sub copyDataFromDatabase()
 Dim moviesConn As ADODB.connection
set moviesConn = New ADODB.Connection
moviesConn.connectionstring= _
"provider=Microsoft.ACE.OLEDB.12.0; Data
 Source=C:\myFolder\myAccessFile.accdb;
 persist / Security =nfo=False;"
 C:\Users\Andrew.Gould\Desktop\Database\movies.accdb;
'Opening and closing a Connection:-
moviesconn.open
moviesConn.close
end sub
'setting a sql server Connection strings-
sql server ->
 SQL Server Native Client 11.0 OLE DB provider ->
```

Connecting to an sql server instance.

```
sub copyDatafromDatabase()
Dim Moviescom AS ADODB.Connection
set Moviescom = New ADODB.Connection
Moviescom.ConnectionString = _
"provider=SQLNCLI11; Server=DESKTOP-UPB90TU\
    SQLEXPRESS; Database=Movies;
    Trusted_connection=Yes;"
```

```
provider=SQLNCLI11; Server=myServerName\
  theInstanceName; Database=myDataBase;
  Trusted_connection=Yes;
```

```
Moviescom.open
Moviescom.close
End sub
```

'Connection strings and wizards :-

Before that :-
MoviesConn.ConnectionString = ""

other ways.

Data -> Get External Data -> From Access ->

select Movies.accdb -> tblActor (table) -> OK

Import Data -> properties -> Definition ->

Connection String

copy and paste in MoviesConn.ConnectionString.

But if we found " then add "" (" equals "")

for Access

Data -> Get External Data -> From Other Sources ->

From SQL Server -> Data Connection Wizard ->

Server Name: Desktop-UPB90TU\SQLExpress ->

Next -> Movies -> tblActor table -> Next ->

FileName: MovieConnection.odc → save here

Finish → Import Data → properties →

Definition → Connection string

For sql server

'switching Between Connections :-

option Explicit

Const ConstsAccess As string = " "

Const ConstsSql As string = " "

sub copyDatafromDatabase()

Dim MoviesConn As ADODB.Connection

set MoviesConn = New ADODB.Connection

MoviesConn.ConnectionString = constsAccess / or constsql

MoviesConn.open

Moviesconn.close

End sub

'creating a Recordset :-

```
sub---
Dim---
Dim MoviesData As ADODB.Recordset
set Moviesconn ---
set MoviesData=New ADODB.Recordset

'setting the Active Connection :-
Moviesconn.ConnectionString = constrAccess
Movies conn.open
 with MoviesData
    .ActiveConnection = Movies conn
'setting the Recordset Source :-
    .Source = "tblActor"
'setting the Recordset.locktype :-
'Google→ Ado recordset locktype→ microsoft.com.
    .Locktype = adlockReadonly
```

→ adlockBatchoptimistic
adlockoptimistic
adlock pessimistic
adlockReadonly

'setting the Recordset cursor type :-

• cursorType = adOpenForwardOnly / adOpenDynamic
adOpenForwardOnly
adOpenKeyset
adOpenStatic

'Google -> Ado recordset cursortype
-> Microsoft.com / w3 schools
Better

'Opening and closing a Recordset :-

• Open
End With

MoviesData.close => closes recordset
MoviesData.close => closes connection
conn
End Sub

'Copying Recordset Data into Excel :-

Sub ---
End With
Worksheets.Add // we dont get column Heading.
// Birth Date format
// Column width
Range("A2").CopyFromRecordset MoviesData
MoviesData.close // check with Both
MoviesData.close connection strings.
End Sub

'Testing Different Databases:-

Test By changing connections string

'Looping over the Fields collection:-

```
sub - -
Dim - -
Dim - - -
Dim MoviesField As ADODB.Field  -> Deplicated ~~method~~ property
- - -
Worksheets.Add
For Each MoviesField In MoviesData.Fields
  Activecell.Value = MoviesField.Name
  ~~Activecell.value~~
  Activecell.Offset(0,1).Select
Next MoviesField
Range("A1").Select
Range("A2").CopyFromRecordset MoviesData
- - -
- - -
End sub.   Range("A1").CurrentRegion.Entirecolumn.
- - -        AutoFit.
- - -
End sub
```

' Using late Binding Techniques :-

```
sub copyDataFromDataBaseLateBinding()
  Dim MoviesConn As object
  Dim MoviesData As object
  Dim MoviesField As object

set MoviesConn = CreateObject("ADODB.connection")
set MoviesData = CreateObject("ADODB.Recordset")
// object Browser / F8 & step in and find values.
---
---
 .LockType = 1
 .CursorType = 0  // Not supports intelligence.
---
---
End sub.
```

' Adding Error Handling :-

```
sub---
---
on Error GoTo closeConnection
---
---
---
```

```
MoviesData.close

CloseConnection:
    MoviesData.close
End sub.
```

```
sub---
End with
on Error GoTo closeRecordset
--
On Error GoTo 0
closeRecordset:
    MoviesData.close

closeConnection:
    MoviesData.close.
End sub.
```

```
' writing a simple SELECT statement:-
sub--
source = "SELECT * FROM tblActor;"
--
End sub      //In sql server ; is optional
             In other DB, ; is compulsory
```

'specifying with Columns to Return:-

```
sub--
.Source = "SELECT ActorName, ActorDOB, ActorGender
                FROM tblActor;"
--
End sub
```

'Sorting the Query Results:-

```
sub--
.Source = "SELECT ActorName, ActorDOB, ActorGender
        From tblActor ORDER BY ActorName (or) ASC;"
                                    ActorName
                                    DESC;"
---
--
End sub.
```

'Using the Access Query Designer:-

MS Access → create tab → Query Design → Show Table → tblDirector, tblFilm → Close → Run/View → View (SQL view) → copy & paste in Source. (keep in single line & spacing).

Using the Sql server Query Designer:-

SSMS → select Movies → New query (tab) → Right click in Background of page → Design Query in editor → Add Table → tblActor, tblCast, tblFilm → close → check & ~~d~~operation like sorting & de.... → copy query → paste in source.

Adding Text criteria to a query:-

Access → CREATE TAB → query Design → tblFilm → close.

~~FilmName~~

Field : FilmName

Table : tblFilm

Criteria: A* hit enter (Like "A*") → Run / view (Sql View) → copy paste.

Source: syntax Error : ("A*") solution

(or)

'A*' (prefered).

(No Records)

Because of * so, replace with %

'A%' ✓ (In VBA Editor).

'Date criteria in Access :-

Access → create tab → query design → tblfilm → close.

Field: FilmName FilmReleaseDate

Table: tblfilm tblfilm

Criteria: > 31/12/1999 & enter

>#31/12/1999# ⇒ UK format

Run → View (SQL view) → copy query.

VBA editor Automatically switches the positions of Date and month (In VBA Editor format).

>#10/31/1999# → US format

If we change to >#31/12/1999#, we will get exactly same result because 31 can be only date. If date is ~~#day/month/year~~ = #03/08/2007# Then No Records.

because VBA Editor treats as ~~3rd of~~ 8th of march 2007

Instead of 3rd of August.

so; #08/03/2007# => US format in VBA Editor.

UK format in Access DB.

(OR) Use International standard way :-

= #2007-08-03#

year month day

ISO 8601

Run in Access & sql connection

'Date Criteria in sql server :-

= '2007-03-08'

This fails in sql server because we don't use #.

Use single code (') instead.

'Getting User input to modify queries :-

Access → Create tab → Query Design → tblFilm → close

Field:	FilmName	FilmReleaseDate	FilmRunTimeMinutes
Table:	tblFilm	tblFilm	tblFilm
Sort:			Ascending
criteria:			between 100 and 180

→ Run → View (Sql View) → copy query.

```
Function GetSqlString() As String
  Dim SqlString As String
  Dim MinLength As Integer, MaxLength As Integer
MinLength = Application.InputBox("Enter the shortest RunTime", type:=1)
MaxLength = Application.InputBox("enter the longest RunTime", Type:=1)
SqlString = "SELECT FilmName, FilmReleaseDate, FilmRunTimeMinutes" & _
```

FI

// we can restrict the return type to a number but not possible with InputBox.

// A number. Acts like validation.

space

```
"From tblFilm" & _
"WHERE FilmRunTimeMinutes BETWEEN " &
        MinLength & _
"AND" & MaxLength & _
"ORDER BY FilmRunTimeMinutes ASC;"

GetSQLString = SQLString

End Function.
```

```
Sub ---
--- -
•Source = GetSQLString
---
---
End Sub:
```

Modifying Data using ADO Recordsets:- (35)

Inserting, Updating and Deleting Records:-

- Recap of the ADO library, Connections & Recordsets

- Recordset LockTypes
- Inserting New Records
- changing Field Values
- modifying Existing Records
- Looping Through a Recordset
- Deleting Records
- Transactions
- Dealing with Related Tables

'modifying Data using ADO :-

Module -> UsingADO

```
'A Recap of Creating a Connection :-
sub ConnectToDB()
 'Dim MoviesConn AS Object
'set Moviesconn = Createobject ("ADODB.Connection")

'MoviesConn.
```

```
Dim MoviesConn As ADODB.Connection
set MoviesConn = New ADODB.connection
MoviesConn.ConnectionString = ""
'Connection strings for Access:-
'Connection strings for MSSQL:-
'Opening and closing Connections:-
MoviesConn.open
MoviesConn-close
set MoviesConn= Nothing
End sub.
```

'Modifying Data using Recordsets :-

↓

2 ways

Recordset objects

command objects.

```
option Explicit
Const AccessConstr As string = " "
const sqlconstr As string = " "
sub ConnectToDB()
Dim MoviesConn As ADODB.Connection
Dim MoviesData As ADODB.Recordset
set MoviesConn = New ADODB.Connection
set MoviesData = New ADODB.Recordset
MoviesConn.ConnectionString = AccessConstr
MoviesConn.Open
'Basic Recordset properties:-
with MoviesData
    .ActiveConnection = MoviesConn
    .Source = "tblFilm"
'Recordset Lock Types:-
    .LockType = adLockOptimistic
```

'Record Cursor Types :-

- CursorType = adOpenForwardOnly

'Opening and closing a Record set

- Open

End with

MoviesData.close => Recordset

MoviesData.close => Connection

Set MoviesData = Nothing } Crud practise.
Set MoviesConn = Nothing

End sub.

'Including Error Handling :-

sub ---
— — —
— —

On Error GoTo CloseConnection

'End with

On Error GoTo closeRecordset

end with

closeRecordset:
    MoviesData.close

closeConnection:
    MoviesData.close

End sub.

' Adding New Records :-

sub --- Access DB

    .open
    On Error GoTo closeRecordset
' .AddNew_space (or) seperate fields by Fields
' .AddNew
    => Not casesensitive.
    .Fields("FilmName").Value = "Gravity"
    .Fields("FilmReleaseDate").Value = #07/Nov/2013# (or) us format.
                                    #11/7/2013#
    .Fields("FilmRunTimeMinutes").value = 91
    .Update => it will commit data to Database.
End with
End sub.

'Adding new Records to sql Server :-

```
sub - -
 .AddNew
 .Fields("FilmID").value = 266
end sub.
```

'Cancelling the Recordset Update :-

```
sub - -
on error GOTO closeRecordset
.AddNew                          => Duplicate and etc - - -
.Fields("FilmID").value = 266
.Fields - -
 . Update => jumps to closeRecordset
end with
closeRecordset:
 MoviesData.close                => Runtime error : 3219 becoz we didn't commit database & we are at the middle of 3 records - (we haven't cancelled our update.)
End sub.
```

```
sub---
  .Update
end with
closeRecordSet:
  MoviesData.CancelUpdates
  MoviesData.close
---
---
end sub.
```

'Calculating the Next ID Number:-

```
Sub---
Dim---
Dim---
Dim NextID As Long
---
---
.CursorType = adOpenKeyset
.Open
.MoveLast
NextID = .Fields("FilmID").value + 1
```

On Error GOTO closeRecordset

.AddNew

.Fields("FilmID").value = NextID

--

--

End sub.

'Adding Records Using cell values :-

=> To add multiple Records.

Sub --

Dim --

Dim --

Dim --

Dim x As Range

--

--

on Error GOTO closeRecordset

Sheet1.Activate

For Each x In Range("A3", Range("A3").End(xlDown))

.AddNew

.Fields("FilmID").value = NextID => In Access DB. commit it.

.Fields("FilmName").value = x.offset(0,1).value

sqlserver

```
.Fields("FilmReleaseDate").Value = x.Offset(0,2).Value
.Fields("FilmRunTimeMinutes").Value = x.Offset(0,3).Value
.Update
NextID = NextID + 1
Next x
End With
End Sub
```

```
' Altering Data in a Recordset
Sub ModifyExistingData()
  Dim MoviesConn AS ADODB.Connection
  Dim MoviesData AS ADODB.Recordset
  Set MoviesConn = New ADODB.Connection
  Set MoviesData = New ADODB.Recordset
MoviesConn.ConnectionString = AccessConstr
MoviesConn.Open
```

On Error GoTo closeConnection

With MoviesData

- ActiveConnection = MoviesConn
- Source = "tblFilm"
- LockType = adLockOptimistic
- CursorType = adOpenKeyset
- Open

' Looping over a Recordset ⇒ .bof = Beginning of first Record
⇒ .eof = After the last Record

On Error GoTo closeRecordset

Do until .EOF

'checking the value of a field :-

If .Fields("FilmRunTimeMinutes").Value > 180 Then
Debug.Print - - - -
.Fields("FilmRunTimeMinutes").Value = 180
.Update
End If

.MoveNext

Loop

End with

CloseRecordset:

MoviesData.CancelUpdate => Run-Time Error : 3021 Either BOF or EOF is True.

MoviesConn.close

MoviesData.close

CloseConnection:

MoviesConn.close

set MoviesData = Nothing

set MoviesConn = Nothing

End sub

' Avoiding the Cancel Update Method

---

sub..

...

Loop

.MoveFirst

End with

CloseRecordset:

MoviesData.CancelUpdate -> No runtime error.

End sub

sub - -

loop

'.movefirst Run in Both DB's

End with

closeRecordset:

If (Not MoviesData.BOF) And

(Not MoviesData.EOF) Then

MoviesData.CancelUpdate

End If

End sub.

'selecting Power Records

sub - -

.Source = "SELECT FilmName, FilmRunTimeMinutes

From tblFilm WHERE FilmRunTimeMinutes>150"

- -

- -

.open

on Error GoTo closeRecordset

```
Do until .EOF
 .Fields("FilmRuntimeMinutes").value=150
 .Update
 .MoveNext
loop
```

end sub.

'Deleting Records :-

```
sub DeleteExistingData()
Do until .EOF
  .Delete
 1.Update =>optional
  .MoveNext
loop
end loop.
```

'Using Transactions :-

sub ---:

on Error GoTo CloseRecordset

MoviesConn.BeginTrans

Do until .EOF

.Delete

'.Update =>optional

Loop

'MoviesConn.CommitTrans

MoviesConn.RollbackTrans

End sub.

'Dealing with Related Tables:-

Access -> DataBaseTools -> Relationships -> first select tbFilms.

Right click:

Edit Relationship -> ☑ Cascade delete Related Records.

and Run this

AccessDB

writing sql statements using ActiveX Data objects

• Using the ADO command object in VBA :-

- Recap of ActiveX Data objects and Connections
- creating a Command Object
- Some Examples of sql statements (Insert, update, Delete)
- setting the CommandText property
- Executing a Command
- Getting sql statements Automatically
- Reading cell contents into sql statements
- Dangers of Dynamic sql
- Using Transactions
- Updating and Deleting Data.

Sql statements in Excel VBA :-

```
'A Recap of Creating Connections :-
'Creating Connection Strings :-
option explicit
 const AccessConstr As string = "__"
 const SqlConstr As string = "_"
sub ConnectToDB()
Dim MoviesConn As ADODB.Connection
set MoviesConn = New ADODB.Connection
MoviesConn.ConnectionString = AccessConstr/
                              &SqlConstr.

MoviesConn.open

MoviesConn.close
set MoviesConn = Nothing
End sub.
```

'creating an ADO command object :-

```
sub --
 Dim
 Dim MoviesCmd As ADODB.Command
 set
 set MoviesCmd = New ADODB.command
 ==
MoviesConn.open
MoviesCmd.ActiveConnection = MoviesConn
'MoviesCmd.CommandText = "" ---> =inserting
                                 updating
                                 Deleting.
'Examples of sql statements :-
==
End sub.
```

'using the insert statement :-

```
Function getInsertTextSql() As string
 Dim sqlSts As string
 sqlSts = "INSERT INTO tblFilm(" &
        "FilmID,filmName,FilmReleaseDate,
        FilmRunTimeMinutes)" &
```

"VALUES (" &_

year-mm-dd

" 999, """ Gravity """, '2013-11-07', 91)"

(or)

'Gravity'

GetInsertText = sqlstr

End Function

Sub --

MoviesCmd.CommandText = GetInsertTextSql

'Executing a command. :-

MoviesCmd.Execute

End sub

Function GetInsertText(Access) As String

==

sqlstr = _

"INSERT INTO tblFilm (" & _

"FilmName, FilmReleaseDate, FilmRuntimeMinutes)" & _

"VALUES (" & _

"'Gravity', #2013-11-07#, 91)"

End Function

'creating sql statements Automatically:-

In sql server:-

Tables -> script Table as -> insert to -> update -> delete

New query Editor window.

In Access:-

'creating sql statements in Access:-

create -> query Design -> ~~show tables~~ ->

Append (+!) -> tblFilm -> OK ->

view -> sql view ->

Field: "Gravity"

Append To: FilmName

view -> sql view.

insert into tblFilm (FilmName)
select "Gravity" AS Expr1;
values ('Gravity')

(i) is optional.

// update
delete.

Reading cell values into sql statements:-

Adding parameters to a Function:-

~~Fun~~
Function GetInsertTextAccess(FilmName as string,
FilmDate as Date,
FilmTime as Integer) As
String

Dealing with Single-Quote characters.
Dim sqlstr As string.

```
sqlStr = _
"INSERT INTO tblFilm (" & _
"FilmName, FilmReleaseDate, FilmRunTimeMinutes)" & _
"VALUES (" & _
"'" & Replace(FilmName, "'", "''") & "', " & _
'Formatting Dates using ISO 8601:-
"#" & Format(FilmDate, "yyyy-mm-dd") & "#, " & _
CStr(FilmTime) & ")"
```

!!

Explicitly converts integer to string

```
GetInsertTextAccess = sqlStr
End Function.
```

'Generating sql for Each Row of cells:-

```
Sub connectToDB()
  Dim
  Dim
  Dim r As Range
  --
  MoviesCmd.ActiveConnection = Moviescomm

  For each r in Range("A3", Range("A3").End(xlDown))
    MoviesCmd.CommandText = _
    GetInsertTextAccess( _
    r.offset(0,1).value, _
    r.offset(0,2).value, _
    r.offset(0,3).value)
    Moviescmd.Execute
  Next r
  Moviescomm.close
  --
  End sub.
```

Access

'Executing the Commands in Access :-

'changing the statement for Sql Server

```
function GetInsertTextSql(FilmName As string,
                          FilmDate As Date,
                          FilmTime As integer) As
                          string
Dim sqlstring As string
sqlsts = _
  "INSERT INTO tblFilm (" & _
"FilmID, FilmName, FilmReleaseDate, FilmRunTimeMinutes)" & _
"VALUES(" & _
"(SELECT MAX(FilmID) FROM tblFilm) +1, " & _
"'" & Replace(FilmName, "'", "''") & "', " & _
"'" & Format(FilmDate, "yyyy-mm-dd") & "', " & _
Cstr(FilmTime) & ");"
```

↓↓ optional but it is Terminator.

```
GetInsertTextSql = sqlstr
End Function.
```

'Executing multiple statements in sql :-

```
Sub - - -
- - - -
For Each - - -
MoviesCmd.CommandText = MoviesCmd.CommandText &
GetInsertTextSql(_
    &. offset(0,1).value,_
    &. offset(0,2).value,_
    &. offset(0,3).value)

Next &
'Debug.print MoviesCmd.CommandText
MoviesCmd.Execute
MoviesConn.close
set MoviesConn = Nothing
End sub.
```

Executing Multiple statements in Access:-

=> same as sql server will cause errors.

Dangers of Dynamic sql:-

in Excel;

Length

143); DROP TABLE tblTest;-- → comments in sql

change FilmTime As string => parameter.

Validating sql statements:-

Function GetInsertTextSql ----

If INSTR(1, sqlstr, "drop", vbTextCompare) <> 0 Then

=> or "drop" → (or)

"; drop " and etc....

"drop"

Raising Errors in Functions:-

Err.Raise
||
Custom
||
Error

range
0-65535

reserved
0-512.

```
Err.Raise vbObject Error + 600, , "Naughty words used"
End If
GetInsertTextSel = sel$
End Function.
```

Sub - -

```
For each x - - -
On Error GoTo ErrHandler
- - -
- - -
ErrHandler:
On Error GoTo 0
Next x
```

```
moviescmd.execute
Moviesconn.close
set Moviesconn = Nothing
End sub
ErrHandler:
  MsgBox "Error number=" & (Err.Number-
                    vbObject-Error) &
          vbnewline & Err.Description
Moviesconn.close
End sub.
```

```
'Using Transactions:-

'Begin, Commit, Rollback, Transactions:-
  Sub
  - -
  - -
  - -
```

```
Moviescmd.ActiveConnection = MoviesConn

MoviesConn.BeginTrans

For Each x In Range("A3", Range("A3").
                        End(xlDown))

On Error GoTo Errorhandler
 MoviesCmd.CommandText = _
   GetInsertTextSql(_
     x.offset(0,1).value, _
     x.offset(0,2).value, _
     x.offset(0,3).value)

On Error GoTo 0
 Moviescmd.Execute

Next x
 MoviesConn.CommitTrans

Movieconn.close
Set Moviesconn = Nothing
```

```
Exit sub
Esssss Handler:
msgBox "Esssss Number=" & (Ess.Number -
                    vbobject Esssss) &
        vbNewLine & Ess.Description

MovieSconn.RollbackTrans
Moviesconn.close
end sub.
```

'Generating other sql statements:-

'Creating an Update statement:-

```
Function GetUpdateTextAccess(FilmName As
    string, FilmTitle As Integer) As string
  Dim sqlstr As string
  sqlstr = _
    "UPDATE tblFilm" & _
    "  SET FilmRuntimeMinutes=" & filmtime & _
```

"WHERE FilmName = '" & FilmName & "';"

GetUpdateTextAccess = sqlstr

End Function.

& Replace (FilmName, "'", "''") &

Sub ---

For Each r ---

MoviesCmd.commandText = _

GetUpdateTextAccess (_

r.offset(0,1).value, _

r.offset(0,3).value)

MoviesCmd.Execute

Next r

End Sub.

' Creating a Delete statement :-

```
function getDeleteTextAccess(FilmName As string)
                                    As string.

Dim sqlstr As string

sqlstr = _
"DELETE FROM tblFilm" & _
"WHERE FilmName = '" & Replace(FilmName, "'", "''")
                                    & "';"

getDeleteTextAccess = sqlstr

End Function.
```

```
Sub...
_ _ _
For Each r...
    MoviesCmd.commandText = _
      getDeleteTextAccess(r.offset(0,1).value)
MoviesCmd.Execute
Next r
End sub
```

```
Sub ---
--- -
For Each x ----
  If Lcase(x.offset(0,4).value) = "awful" Then
  MoviesCmd.CommandText = _
  GetDeleteTextAccess(x.offset(0,1).value)
  MoviesCmd.Execute
  End If
  Next x
  End Sub.
```

' A quick Note on Database Relationships :-

Access Access.

Database Tools → Relationships.

like one to one | many to one
one to many | many to many.
and etc---

⇒ Cascade (edit)