Public Sub ImportEstimates2R80() '<<< THIS DOES IMPORT IN 50 Seconds >>>

'called by MainForm Command33 to duplicate an estimate

'This is similar to MainForm Command33 to duplicate an estimate

Dim db As Database, qdf1 As QueryDef, lngRandom As Long, rst1 As Recordset, rstDbList As Recordset, frm1 As Form

Dim lngCountTot As Long, lngcount As Long, lngLevel As Long

'This sub will import estimates by...:

'1) copy records into Load2 tables (referential Integrity & Cascade Update & Cascade Delete is on

'2) update Primary Keys in Load2EstimateNos, Load2ZoneID & Load2CsiLines

'Eline & Components using cascade update - use random # & loop until all copied

'3) copy the load2 tables back into normal tables

On Error GoTo Errorhandler

DoCmd.Hourglass True

ShowMeter "Importing Estimates - 10%", 10

Set db = CurrentDb

Set frm1 = Forms!ImportOptions

'<<< set EXISTING estimate's <BondRequired> field to False. We later use this field to indicate that this EstimateNos

'was imported, by setting this field to True. The "Imported Estimate List" screen at the bottom of this sub uses

'this field.

Set qdf1 = db.CreateQueryDef("", "UPDATE TqEstimateNos " \_

& "SET TqEstimateNos.BondRequired = False;")

qdf1.Execute

If frm1!Check1 = False And frm1!Check13 = False Then

'not copying estimates or Dbs (must just be importing Gen Info)

GoTo cleanup

End If

ShowMeter "Importing Estimates - 15%", 15

'<<< Empty the Load2 tables

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2EstimateNos.\* " \_

& "FROM TqLoad2EstimateNos;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2dbList.\* " \_

& "FROM TqLoad2dbList;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2Zone.\* " \_

& "FROM TqLoad2Zone;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2CsiLines.\* " \_

& "FROM TqLoad2CsiLines;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2Elines.\* " \_

& "FROM TqLoad2Elines;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2Components.\* " \_

& "FROM TqLoad2Components;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2SubLines.\* " \_

& "FROM TqLoad2Sublines;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2Tlines.\* " \_

& "FROM TqLoad2Tlines;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2CsiDistributions.\* " \_

& "FROM TqLoad2CsiDistributions;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2AdjustmentsP.\* " \_

& "FROM TqLoad2AdjustmentsP;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2AdjustDistributions.\* " \_

& "FROM TqLoad2AdjustDistributions;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2plHeadings.\* " \_

& "FROM TqLoad2plHeadings;")

qdf1.Execute

Set qdf1 = db.CreateQueryDef("", "DELETE TqLoad2plMultipliers.\* " \_

& "FROM TqLoad2plMultipliers;")

qdf1.Execute

'1) <<< copy records into Load2 tables. (referential Integrity & Cascade Update & Cascade Delete is on)

'Load2 tables have referential intengrity, cascade update related records, & cascade delete turned on

'we can use cascade update to change foreign keys in related tables

ShowMeter "Importing Estimates - 18%", 18

'Copy EstimateNos

If frm1!Check1 And frm1!Check13 Then

'importing est & dbs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2EstimateNos " \_

& "SELECT ExportEstimateNos.\* " \_

& "FROM ExportEstimateNos;")

qdf1.Execute

Else

'Not importing both

If frm1!Check13 Then

'importing Dbs only

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2EstimateNos " \_

& "SELECT ExportEstimateNos.\* " \_

& "FROM ExportEstimateNos LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Not Null));")

qdf1.Execute

Else

'importing estimates only

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2EstimateNos " \_

& "SELECT ExportEstimateNos.\* " \_

& "FROM ExportEstimateNos LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

qdf1.Execute

End If

End If

'<<< set Load2EstimateNos's <BondRequired> field to True. This field indicates that this estimate was imported

'The "Imported Estimate List" screen at the bottom of this sub uses this field.

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2EstimateNos " \_

& "SET TqLoad2EstimateNos.BondRequired = True;")

qdf1.Execute

If frm1!Check13 Then

'importing dbs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2DbList " \_

& "SELECT ExportDbList.\* " \_

& "FROM ExportDbList;")

qdf1.Execute

End If

'Copy Zone

If frm1!Check1 And frm1!Check13 Then

'importing est & dbs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2Zone " \_

& "SELECT ExportZone.\* " \_

& "FROM ExportEstimateNos INNER JOIN ExportZone ON ExportEstimateNos.EstimateID = ExportZone.EstimateID;")

Else

'Not importing both

If frm1!Check13 Then

'importing Dbs only

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2Zone " \_

& "SELECT ExportZone.\* " \_

& "FROM (ExportEstimateNos INNER JOIN ExportZone ON ExportEstimateNos.EstimateID = ExportZone.EstimateID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Not Null));")

Else

'importing estimates only

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2Zone " \_

& "SELECT ExportZone.\* " \_

& "FROM (ExportEstimateNos INNER JOIN ExportZone ON ExportEstimateNos.EstimateID = ExportZone.EstimateID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

End If

End If

qdf1.Execute

'Copy CsiLines

If frm1!Check1 And frm1!Check13 Then

'importing est & dbs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2CsiLines " \_

& "SELECT ExportCsiLines.\* " \_

& "FROM ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID;")

Else

'Not importing both

If frm1!Check13 Then

'importing Dbs only

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2CsiLines " \_

& "SELECT ExportCsiLines.\* " \_

& "FROM (ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Not Null));")

Else

'importing estimates only

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2CsiLines " \_

& "SELECT ExportCsiLines.\* " \_

& "FROM (ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

End If

End If

qdf1.Execute

ShowMeter "Importing Estimates - 20%", 20

'Copy Elines

If frm1!Check1 And frm1!Check13 Then

'importing est & dbs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2Elines " \_

& "SELECT ExportElines.\* " \_

& "FROM (ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) INNER JOIN ExportElines ON ExportCsiLines.CsiLineID = ExportElines.CsiLineID;")

Else

'Not importing both

If frm1!Check13 Then

'importing Dbs only

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2Elines " \_

& "SELECT ExportElines.\* " \_

& "FROM ((ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) INNER JOIN ExportElines ON ExportCsiLines.CsiLineID = ExportElines.CsiLineID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Not Null));")

Else

'importing estimates only

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2Elines " \_

& "SELECT ExportElines.\* " \_

& "FROM ((ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) INNER JOIN ExportElines ON ExportCsiLines.CsiLineID = ExportElines.CsiLineID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

End If

End If

qdf1.Execute

ShowMeter "Importing Estimates - 25%", 25

'Copy Components

'delete temp table if it exists

On Error Resume Next

db.TableDefs.Refresh

db.TableDefs.Delete "TempComp1"

db.TableDefs.Refresh

On Error GoTo Errorhandler

If frm1!Check1 And frm1!Check13 Then

'importing est & dbs

Set qdf1 = db.CreateQueryDef("", "SELECT ExportComponents.\* " \_

& "INTO TempComp1 " \_

& "FROM ((ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) INNER JOIN ExportElines ON ExportCsiLines.CsiLineID = ExportElines.CsiLineID) INNER JOIN ExportComponents ON ExportElines.ElineID = ExportComponents.ElineID;")

Else

'Not importing both

If frm1!Check13 Then

'importing Dbs only

Set qdf1 = db.CreateQueryDef("", "SELECT ExportComponents.\* " \_

& "INTO TempComp1 " \_

& "FROM (((ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) INNER JOIN ExportElines ON ExportCsiLines.CsiLineID = ExportElines.CsiLineID) INNER JOIN ExportComponents ON ExportElines.ElineID = ExportComponents.ElineID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Not Null));")

Else

'importing estimates only

Set qdf1 = db.CreateQueryDef("", "SELECT ExportComponents.\* " \_

& "INTO TempComp1 " \_

& "FROM (((ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) INNER JOIN ExportElines ON ExportCsiLines.CsiLineID = ExportElines.CsiLineID) INNER JOIN ExportComponents ON ExportElines.ElineID = ExportComponents.ElineID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

End If

End If

qdf1.Execute

lngCountTot = qdf1.RecordsAffected

'rst1.Close 'this prevents table record locking in a looping situation

lngcount = 0

lngLevel = 0

ShowMeter "Importing Estimates - 25% " & lngcount & " of " & lngCountTot, 25

Do Until lngCountTot = lngcount

'you must repeat Components query since referential integrety prevents all records from being inserted at once.

Select Case True

Case lngLevel < 1

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2Components " \_

& "SELECT TempComp1.\* " \_

& "FROM TempComp1 " \_

& "WHERE (((TempComp1.Component2ID) Is Null));")

qdf1.Execute

lngLevel = lngLevel + 1

lngcount = lngcount + qdf1.RecordsAffected

ShowMeter "Importing Estimates - 25% " & lngcount & " of " & lngCountTot, 25

Case Else

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2Components " \_

& "SELECT TempComp1.\* " \_

& "FROM TempComp1 LEFT JOIN TqLoad2Components ON TempComp1.ComponentID = TqLoad2Components.ComponentID " \_

& "WHERE (((TqLoad2Components.ComponentID) Is Null));")

qdf1.Execute

lngcount = lngcount + qdf1.RecordsAffected

ShowMeter "Importing Estimates - 25% " & lngcount & " of " & lngCountTot, 25

End Select

Loop

'delete temp table if it exists

On Error Resume Next

db.TableDefs.Refresh

db.TableDefs.Delete "TempComp1"

db.TableDefs.Refresh

On Error GoTo Errorhandler

'Copy SubLines

If frm1!Check1 Then

'importing estimates

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2SubLines " \_

& "SELECT ExportSubLines.\* " \_

& "FROM ((ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) INNER JOIN ExportSubLines ON ExportCsiLines.CsiLineID = ExportSubLines.CsiLineID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

qdf1.Execute

End If

'Copy Tlines

If frm1!Check1 Then

'importing estimates

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2Tlines " \_

& "SELECT ExportTlines.\* " \_

& "FROM ExportDbList RIGHT JOIN (((ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) INNER JOIN ExportElines ON ExportCsiLines.CsiLineID = ExportElines.CsiLineID) INNER JOIN ExportTlines ON ExportElines.ElineID = ExportTlines.ElineID) ON ExportDbList.dbEstimateID = ExportEstimateNos.EstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

qdf1.Execute

End If

ShowMeter "Importing Estimates - 28%", 28

'Copy CsiDistributions

If frm1!Check1 Then

'importing estimates

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2CsiDistributions " \_

& "SELECT ExportCsiDistributions.\* " \_

& "FROM ((ExportEstimateNos INNER JOIN ExportCsiLines ON ExportEstimateNos.EstimateID = ExportCsiLines.EstimateID) INNER JOIN ExportCsiDistributions ON ExportCsiLines.CsiLineID = ExportCsiDistributions.CsiLineID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

qdf1.Execute

End If

'Copy AdjustmentsP

If frm1!Check1 Then

'importing estimates

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2AdjustmentsP " \_

& "SELECT ExportAdjustmentsP.\* " \_

& "FROM (ExportEstimateNos INNER JOIN ExportAdjustmentsP ON ExportEstimateNos.EstimateID = ExportAdjustmentsP.EstimateID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

qdf1.Execute

End If

'Copy AdjustDistibutions

If frm1!Check1 Then

'importing estimates

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2AdjustDistributions " \_

& "SELECT ExportAdjustDistributions.\* " \_

& "FROM ((ExportEstimateNos INNER JOIN ExportAdjustmentsP ON ExportEstimateNos.EstimateID = ExportAdjustmentsP.EstimateID) INNER JOIN ExportAdjustDistributions ON ExportAdjustmentsP.AdjustmentsPID = ExportAdjustDistributions.AdjustmentsPID) LEFT JOIN ExportDbList ON ExportEstimateNos.EstimateID = ExportDbList.dbEstimateID " \_

& "WHERE (((ExportDbList.dbEstimateID) Is Null));")

qdf1.Execute

End If

'Copy plHeadings

If frm1!Check13 Then

'importing Dbs

'copy all plHeadings into Load2 table so it's CsiLineID get's updated when random CsiLineIDs get assigned below

'This table get's copied to TqplHeadings in FinalCleanup2R80

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2plHeadings " \_

& "SELECT ExportPlHeadings.\* " \_

& "FROM ExportPlHeadings;")

qdf1.Execute

End If

'Copy plMultipliers

If frm1!Check13 Then

'importing Dbs

'copy all plMultipliers into Load2 table so it's dbEstimateID get's updated when random dbEstimateIDs get assigned below

'This table get's copied to TqplMultipliers in FinalCleanup2R80

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqLoad2plMultipliers " \_

& "SELECT ExportPlMultipliers.\* " \_

& "FROM ExportPlMultipliers;")

qdf1.Execute

End If

ShowMeter "Importing Estimates - 30%", 30

'Fix ClientIDs in Load2EstimateNos table if client exists in TqClients, or add new Client to TqClients table

UpdateClients2

'2) <<< update Primary Keys in Load2 Tables. (let cascade update fix the related IDs in other tables)

'we need to create unique Primary Keys that don't exist in the permanent tables.

'EstimateNos

'be sure copy field = False

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2EstimateNos " \_

& "SET TqLoad2EstimateNos.Copy = False;")

qdf1.Execute

'let's generate a random number between 0 & 2,147,483,600. If the existing ID is less than 0 then we

'add the random #, else we subtract the random number to get new ID's that are in the range of a long.

'Then, at that point we have changed all the Primary Key ID's in load2 tables. A small percent of these lines

'will conflict with existing ID's in permanent tables. Mark those that can be copied as Selected = Yes.

'Generate a new random number and repeat this process until all records have Selected = Yes.

'Now the entire Load2 table can be copied into the permanent table.

randomize

'dbList table - handle Imported Databases, that have MatchScreen matches.

'if importing more dbs than current dbs then those will get random IDs with rest of EstimateNos table (6" below here)

'if we are importing estimates only (no dbs) then Match Screen has already updated the dbList records

If frm1!Check13 Then

'importing dbs - treat them as special case...

'fix Load2EstimateNos & Load2DbList records which have MatchScreen matches, with MatchScreen data AND...

'so new EstimateID is recorded in Load2dbList table with it's matching MatchScreen\_ExportDbID. This will allow

'mapping of ExportDb fields to new Permanent Db fields like DbUsedID, etc.

'if importing more dbs than current db then those will get random IDs with rest of EstimateNos table (6" below here)

'if we are importing estimates only (no dbs) then Match Screen has already updated the dbList records

Set rstDbList = db.OpenRecordset("SELECT TqDbList.dbEstimateID, TqDbList.PrimaryDb, TqDbList.MatchScreen\_ExportDbID, TqDbList.MatchScreen\_ExistingDbID " \_

& "FROM TqDbList " \_

& "WHERE (((TqDbList.MatchScreen\_ExportDbID) Is Not Null));", dbOpenDynaset)

Do Until rstDbList.EOF

'we are on the 1st dbList record that has a matchScreen\_ExportDbID

'create a new random EstimateID

lngRandom = Int((CDbl(2147483600) - CDbl(-2147483600) + 1) \* Rnd + (-2147483600))

'verify this EstimateID does not exist in TqEstimateNos

Set rst1 = db.OpenRecordset("SELECT TqEstimateNos.EstimateID " \_

& "FROM TqEstimateNos " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngRandom & "));", dbOpenDynaset)

If rstIsEmpty(rst1) Then

'This estimateID is good. It does not exist in TqEstimateNos table

'change Load2EstimateNos (& Load2dbList) estimateID to this number

'note that referential update will fix IDs in other Load2 tables

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2EstimateNos " \_

& "SET TqLoad2EstimateNos.EstimateID =" & lngRandom & ", TqLoad2EstimateNos.Copy = True, TqLoad2EstimateNos.BondRequired = False " \_

& "WHERE (((TqLoad2EstimateNos.EstimateID)=" & rstDbList!MatchScreen\_ExportDbID & "));")

qdf1.Execute

'also fix other fields in Load2dbList (MatchScreen fields) so the Load2dbList record will contain all 3 ID's

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2DbList " \_

& "SET TqLoad2DbList.MatchScreen\_ExportDbID =" & rstDbList!MatchScreen\_ExportDbID & ", TqLoad2DbList.MatchScreen\_ExistingDbID =" & rstDbList!MatchScreen\_ExistingDbID & " " \_

& "WHERE (((TqLoad2DbList.dbEstimateID)=" & lngRandom & "));")

qdf1.Execute

'clear these fields in dbList (MatchScreen fields) so only the Load2dbList records will contain all 3 ID's

rstDbList.Edit

rstDbList!MatchScreen\_ExportDbID = Null

rstDbList!MatchScreen\_ExistingDbID = Null

rstDbList.Update

'move to the next database in DbList

rstDbList.MoveNext

Else

'ID exists - loop without advancing to next dblist record to try a different random number

End If

Loop

End If

'EstimateNos table (this will take care of any dbs not fixed above (if no matchList)

Do Until 1 = 2

'\*\*\* generate a random number between 0 and 2,147,483,600 \*\*\*

'To produce random integers in a given range, use this formula:

' Int((upperbound - lowerbound + 1) \* Rnd + lowerbound)

'Change every EstimateID where Copy = False

lngRandom = Int((2147483600 - 0 + 1) \* Rnd + 0)

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2EstimateNos " \_

& "SET TqLoad2EstimateNos.EstimateID =IIf([EstimateID]>0,[EstimateID]-" & lngRandom & ",[EstimateID]+" & lngRandom & ") " \_

& "WHERE (([Copy]=False));")

qdf1.Execute

If qdf1.RecordsAffected = 0 Then

'all ID's have Copy = Yes so exit Do loop

Exit Do

End If

'Mark Every ID that is unique (no duplicate in EstimateNos table) to Copy = Yes

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2EstimateNos LEFT JOIN TqEstimateNos ON TqLoad2EstimateNos.EstimateID = TqEstimateNos.EstimateID " \_

& "SET TqLoad2EstimateNos.Copy = Yes " \_

& "WHERE (((TqEstimateNos.EstimateID) Is Null));")

qdf1.Execute

Loop

'Zone

'be sure Load2Zone's copy field = False

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2Zone " \_

& "SET TqLoad2Zone.Copy = False;")

qdf1.Execute

ShowMeter "Importing Estimates - 33%", 33

Do Until 1 = 2

'\*\*\* generate a random number between 0 and 2,147,483,600 \*\*\*

'To produce random integers in a given range, use this formula:

' Int((upperbound - lowerbound + 1) \* Rnd + lowerbound)

'Change every ZoneID where Copy = False

lngRandom = Int((2147483600 - 0 + 1) \* Rnd + 0)

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2Zone " \_

& "SET TqLoad2Zone.ZoneID = IIf([ZoneID]>0,[ZoneID]-" & lngRandom & ",[ZoneID]+" & lngRandom & ") " \_

& "WHERE (([Copy]=False));")

qdf1.Execute

If qdf1.RecordsAffected = 0 Then

'all ZoneID's have Copy = Yes so exit Do loop

Exit Do

End If

'Mark Every Load2Zone.ZoneID that is unique (no duplicate in Zone table) to Copy = Yes

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2Zone LEFT JOIN TqZone ON TqLoad2Zone.ZoneID = TqZone.ZoneID " \_

& "SET TqLoad2Zone.Copy = Yes " \_

& "WHERE (((TqZone.ZoneID) Is Null));")

qdf1.Execute

Loop

ShowMeter "Importing Estimates - 35%", 35

'CsiLines

'be sure Load2CsiLines's copy field = False

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2CsiLines " \_

& "SET TqLoad2CsiLines.Copy = False;")

qdf1.Execute

Do Until 1 = 2

'\*\*\* generate a random number between 0 and 2,147,483,600 \*\*\*

'To produce random integers in a given range, use this formula:

' Int((upperbound - lowerbound + 1) \* Rnd + lowerbound)

'Change every CsiLineID where Copy = False

lngRandom = Int((2147483600 - 0 + 1) \* Rnd + 0)

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2CsiLines " \_

& "SET TqLoad2CsiLines.CsiLineID = IIf([CsiLineID]>0,[CsiLineID]-" & lngRandom & ",[CsiLineID]+" & lngRandom & ") " \_

& "WHERE (([Copy]=False));")

qdf1.Execute

If qdf1.RecordsAffected = 0 Then

'all ElineID's have Copy = Yes so exit Do loop

Exit Do

End If

'Mark Every Load2CsiLines.CsiLineID that is unique (no duplicate in CsiLines table) to Copy = Yes

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2CsiLines LEFT JOIN TqCsiLines ON TqLoad2CsiLines.CsiLineID = TqCsiLines.CsiLineID " \_

& "SET TqLoad2CsiLines.Copy = Yes " \_

& "WHERE (((TqCsiLines.CsiLineID) Is Null));")

qdf1.Execute

Loop

'AdjustmentsP

'be sure Load2AdjustmentsP's copy field = False

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2AdjustmentsP " \_

& "SET TqLoad2AdjustmentsP.Copy = False;")

qdf1.Execute

Do Until 1 = 2

'\*\*\* generate a random number between 0 and 2,147,483,600 \*\*\*

'To produce random integers in a given range, use this formula:

' Int((upperbound - lowerbound + 1) \* Rnd + lowerbound)

'Change every AdjustmentsPID where Copy = False

lngRandom = Int((2147483600 - 0 + 1) \* Rnd + 0)

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2AdjustmentsP " \_

& "SET TqLoad2AdjustmentsP.AdjustmentsPID = IIf([AdjustmentsPID]>0,[AdjustmentsPID]-" & lngRandom & ",[AdjustmentsPID]+" & lngRandom & ") " \_

& "WHERE (([Copy]=False));")

qdf1.Execute

If qdf1.RecordsAffected = 0 Then

'all ElineID's have Copy = Yes so exit Do loop

Exit Do

End If

'Mark Every TqLoad2AdjustmentsP.AdjustmentsPID that is unique (no duplicate in CsiLines table) to Copy = Yes

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2AdjustmentsP LEFT JOIN TqAdjustmentsP ON TqLoad2AdjustmentsP.AdjustmentsPID = TqAdjustmentsP.AdjustmentsPID " \_

& "SET TqLoad2AdjustmentsP.Copy = Yes " \_

& "WHERE (((TqAdjustmentsP.AdjustmentsPID) Is Null));")

qdf1.Execute

Loop

ShowMeter "Importing Estimates - 40%", 40

'ELINES

'be sure Load2Eline's copy field = False

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2Elines " \_

& "SET TqLoad2Elines.Copy = False;")

qdf1.Execute

Do Until 1 = 2

'\*\*\* generate a random number between 0 and 2,147,483,600 \*\*\*

'To produce random integers in a given range, use this formula:

' Int((upperbound - lowerbound + 1) \* Rnd + lowerbound)

'Change every ElineID where Copy = False

lngRandom = Int((2147483600 - 0 + 1) \* Rnd + 0)

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2Elines " \_

& "SET TqLoad2Elines.ElineID = IIf([ElineID]>0,[ElineID]-" & lngRandom & ",[ElineID]+" & lngRandom & ") " \_

& "WHERE (([Copy]=False));")

qdf1.Execute

ShowMeter "Importing Estimates - 40% " & qdf1.RecordsAffected, 40

If qdf1.RecordsAffected = 0 Then

'all ElineID's have Copy = Yes so exit Do loop

Exit Do

End If

'Mark Every Load2Eline.ElineID that is unique (no duplicate in Elines table) to Copy = Yes

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2Elines LEFT JOIN TqElines ON TqLoad2Elines.ElineID = TqElines.ElineID " \_

& "SET TqLoad2Elines.Copy = Yes " \_

& "WHERE (((TqElines.ElineID) Is Null));")

qdf1.Execute

Loop

ShowMeter "Importing Estimates - 45%", 45

'COMPONENTS

'be sure Load2Components's copy field = False

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2Components " \_

& "SET TqLoad2Components.Copy = False;")

qdf1.Execute

Do Until 1 = 2

'\*\*\* generate a random number between 0 and 2,147,483,600 \*\*\*

'To produce random integers in a given range, use this formula:

' Int((upperbound - lowerbound + 1) \* Rnd + lowerbound)

'Change every ComponentID where Copy = False

lngRandom = Int((2147483600 - 0 + 1) \* Rnd + 0)

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2Components " \_

& "SET TqLoad2Components.ComponentID = IIf([ComponentID]>0,[ComponentID]-" & lngRandom & ",[ComponentID]+" & lngRandom & ") " \_

& "WHERE (([Copy]=False));")

qdf1.Execute

ShowMeter "Importing Estimates - 45% " & qdf1.RecordsAffected, 45

If qdf1.RecordsAffected = 0 Then

'all ElineID's have Copy = Yes so exit Do loop

Exit Do

End If

'Mark Every Load2Components.ComponentID that is unique (no duplicate in Components table) to Copy = Yes

Set qdf1 = db.CreateQueryDef("", "UPDATE TqLoad2Components LEFT JOIN TqComponents ON TqLoad2Components.ComponentID = TqComponents.ComponentID " \_

& "SET TqLoad2Components.Copy = Yes " \_

& "WHERE (((TqComponents.ComponentID) Is Null));")

qdf1.Execute

Loop

ShowMeter "Importing Estimates - 50%", 50

'====================================

'<<< 2e) copy the load2 tables back into normal tables - this works because IDs are now unique & don't exist in destin table

'====================================

'At this point Load2EstimateNos, Load2Zone, Load2CsiLines, Load2Elines and Load2Components tables are ready to be

'copied back to Permanent Tables, since all IDs have been changed and are unique

'qdfs must list each field because you must eliminate copying "copy" & "select" fields that don't exist in Perm table

'EstimateNos

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqEstimateNos ( EstimateID, EstimateNo, ClientID, ProjectName, ProjectDescription, EstimateDescription, EstimateDate, Estimator, TaxPercent, FinalQuote, SquareFeet, DispSFSummary, DispSFDetail, Comments, SetDefTaxL, SetDefTaxM, SetDefTaxS, OverrideTaxable, SetDefTaxE, EstimateLocked, LockedBy, DbUsedID, DbZoneID, LabCityUsedID, RsmCityMethod, RsmCciCity, RsmCciState, DisCsiNoNew ) " \_

& "SELECT TqLoad2EstimateNos.EstimateID, TqLoad2EstimateNos.EstimateNo, TqLoad2EstimateNos.ClientID, [ProjectName] AS ProjectNameNew, TqLoad2EstimateNos.ProjectDescription, TqLoad2EstimateNos.EstimateDescription, TqLoad2EstimateNos.EstimateDate, TqLoad2EstimateNos.Estimator, TqLoad2EstimateNos.TaxPercent, TqLoad2EstimateNos.FinalQuote, TqLoad2EstimateNos.SquareFeet, TqLoad2EstimateNos.DispSFSummary, TqLoad2EstimateNos.DispSFDetail, TqLoad2EstimateNos.Comments, TqLoad2EstimateNos.SetDefTaxL, TqLoad2EstimateNos.SetDefTaxM, TqLoad2EstimateNos.SetDefTaxS, TqLoad2EstimateNos.OverrideTaxable, TqLoad2EstimateNos.SetDefTaxE, TqLoad2EstimateNos.EstimateLocked, TqLoad2EstimateNos.LockedBy, TqLoad2EstimateNos.DbUsedID, TqLoad2EstimateNos.DbZoneID, TqLoad2EstimateNos.LabCityUsedID, TqLoad2EstimateNos.RsmCityMethod, TqLoad2EstimateNos.RsmCciCity, TqLoad2EstimateNos.RsmCciState, TqLoad2EstimateNos.DisCsiNoNew " \_

& "FROM TqLoad2EstimateNos;")

qdf1.Execute

'dbList

If frm1.Check13 Then

'Importing dbs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqDbList ( dbEstimateID, PrimaryDb, LabMF, EqMF, MatMF, SubMF, MatchScreen\_ExistingDbID, MatchScreen\_ExportDbID ) " \_

& "SELECT TqLoad2dbList.dbEstimateID, TqLoad2dbList.PrimaryDb, TqLoad2dbList.LabMF, TqLoad2dbList.EqMF, TqLoad2dbList.MatMF, TqLoad2dbList.SubMF, TqLoad2dbList.MatchScreen\_ExistingDbID, TqLoad2dbList.MatchScreen\_ExportDbID " \_

& "FROM TqLoad2dbList;")

qdf1.Execute

Else

' not copying databases

End If

'Zone

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqZone ( ZoneID, ZoneName, EstimateID, LineNo, SF ) " \_

& "SELECT TqLoad2Zone.ZoneID, TqLoad2Zone.ZoneName, TqLoad2Zone.EstimateID, TqLoad2Zone.LineNo, TqLoad2Zone.SF " \_

& "FROM TqLoad2Zone;")

qdf1.Execute

ShowMeter "Importing Estimates - 55%", 55

'CsiLines

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqCsiLines ( CsiLineID, EstimateID, CsiNo, CsiDescription, CsiEstimator, Override1, Override2, Override1Desc, Override2Desc, Sub1Name, Sub2Name, Sub3Name, Sub4Name, Sub5Name, Sub6Name, Sub7Name, Sub8Name, Sub9Name, Sub10Name, cCodeOR1, cCodeOR2, ccDescriptionOR1, ccDescriptionOR2, ccCompanyOR1ID, ccCompanyOR2ID, CsiNoNew ) " \_

& "SELECT TqLoad2CsiLines.CsiLineID, TqLoad2CsiLines.EstimateID, TqLoad2CsiLines.CsiNo, TqLoad2CsiLines.CsiDescription, TqLoad2CsiLines.CsiEstimator, TqLoad2CsiLines.Override1, TqLoad2CsiLines.Override2, TqLoad2CsiLines.Override1Desc, TqLoad2CsiLines.Override2Desc, TqLoad2CsiLines.Sub1Name, TqLoad2CsiLines.Sub2Name, TqLoad2CsiLines.Sub3Name, TqLoad2CsiLines.Sub4Name, TqLoad2CsiLines.Sub5Name, TqLoad2CsiLines.Sub6Name, TqLoad2CsiLines.Sub7Name, TqLoad2CsiLines.Sub8Name, TqLoad2CsiLines.Sub9Name, TqLoad2CsiLines.Sub10Name, TqLoad2CsiLines.cCodeOR1, TqLoad2CsiLines.cCodeOR2, TqLoad2CsiLines.ccDescriptionOR1, TqLoad2CsiLines.ccDescriptionOR2, TqLoad2CsiLines.ccCompanyOR1ID, TqLoad2CsiLines.ccCompanyOR2ID, TqLoad2CsiLines.CsiNoNew " \_

& "FROM TqLoad2CsiLines;")

qdf1.Execute

ShowMeter "Importing Estimates - 60%", 60

'Elines

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqElines ( ElineID, CsiLineID, ZoneID, ElineNo, ElineDescription, ElineQty, ElineUnit, UnitL, UnitM, UnitS, Con1, Con2, Con3, Con1Label, Con2Label, Con3Label, Tqty1Label, Tqty2Label, Tqty3Label, Tqty4Label, Tqty5Label, SeeTakeoff, SeeComponents, TaxL, TaxM, TaxS, [Check], TaxE, UnitE, DailyOutput, DailyLabCost, DailyEqCost, CrewID, TradeQty, EquipQty, MTradeID, MEquipID, SeeDetails, Divider, ccLab, ccEquip, ccMat, ccSub, ccLabDesc, ccEquipDesc, ccMatDesc, ccSubDesc, ccLabCompanyID, ccEquipCompanyID, ccMatCompanyID, ccSubCompanyID, PdlcID, plItemID, plMultiplier, rsmMainLineNo, PercentHrsOnOT, MultiplierForOTRate, AssMhrsPunit, AssEqDaysPunit ) " \_

& "SELECT TqLoad2Elines.ElineID, TqLoad2Elines.CsiLineID, TqLoad2Elines.ZoneID, TqLoad2Elines.ElineNo, TqLoad2Elines.ElineDescription, TqLoad2Elines.ElineQty, TqLoad2Elines.ElineUnit, TqLoad2Elines.UnitL, TqLoad2Elines.UnitM, TqLoad2Elines.UnitS, TqLoad2Elines.Con1, TqLoad2Elines.Con2, TqLoad2Elines.Con3, TqLoad2Elines.Con1Label, TqLoad2Elines.Con2Label, TqLoad2Elines.Con3Label, TqLoad2Elines.Tqty1Label, TqLoad2Elines.Tqty2Label, TqLoad2Elines.Tqty3Label, TqLoad2Elines.Tqty4Label, TqLoad2Elines.Tqty5Label, TqLoad2Elines.SeeTakeoff, TqLoad2Elines.SeeComponents, TqLoad2Elines.TaxL, TqLoad2Elines.TaxM, TqLoad2Elines.TaxS, TqLoad2Elines.Check, TqLoad2Elines.TaxE, TqLoad2Elines.UnitE, TqLoad2Elines.DailyOutPut, TqLoad2Elines.DailyLabCost, TqLoad2Elines.DailyEqCost, TqLoad2Elines.CrewID, TqLoad2Elines.TradeQty, TqLoad2Elines.EquipQty, TqLoad2Elines.MTradeID, TqLoad2Elines.MEquipID, TqLoad2Elines.SeeDetails, TqLoad2Elines.Divider " \_

& ", TqLoad2Elines.ccLab, TqLoad2Elines.ccEquip, TqLoad2Elines.ccMat, TqLoad2Elines.ccSub, TqLoad2Elines.ccLabDesc , TqLoad2Elines.ccEquipDesc, TqLoad2Elines.ccMatDesc, TqLoad2Elines.ccSubDesc, TqLoad2Elines.ccLabCompanyID, TqLoad2Elines.ccEquipCompanyID, TqLoad2Elines.ccMatCompanyID, TqLoad2Elines.ccSubCompanyID, TqLoad2Elines.PdlcID, TqLoad2Elines.plItemID, TqLoad2Elines.plMultiplier, TqLoad2Elines.RsmMainLineNo, TqLoad2Elines.PercentHrsOnOT, TqLoad2Elines.MultiplierForOTRate, TqLoad2Elines.AssMhrsPunit, TqLoad2Elines.AssEqDaysPunit " \_

& "FROM TqLoad2Elines;")

qdf1.Execute

ShowMeter "Importing Estimates - 68% ", 68

'Components

'delete temp table if it exists

On Error Resume Next

db.TableDefs.Refresh

db.TableDefs.Delete "TempComp2"

db.TableDefs.Refresh

On Error GoTo Errorhandler

'make temp table

Set qdf1 = db.CreateQueryDef("", "SELECT TqLoad2Components.ComponentID, TqLoad2Components.ElineID, TqLoad2Components.Component, TqLoad2Components.LabC, TqLoad2Components.MatC, TqLoad2Components.SubC, TqLoad2Components.EquipC, TqLoad2Components.Qty, TqLoad2Components.Unit, TqLoad2Components.Component2ID, TqLoad2Components.OrderNo, TqLoad2Components.SeeComponents, TqLoad2Components.TaxL, TqLoad2Components.TaxE, TqLoad2Components.TaxM, TqLoad2Components.TaxS, TqLoad2Components.DailyOutput, TqLoad2Components.DailyLabCost, " \_

& "TqLoad2Components.DailyEqCost , TqLoad2Components.CrewID, TqLoad2Components.TradeQty, TqLoad2Components.EquipQty, TqLoad2Components.MTradeID, TqLoad2Components.MEquipID, TqLoad2Components.SeeDetails, TqLoad2Components.Divider, TqLoad2Components.ccLab, TqLoad2Components.ccEquip, TqLoad2Components.ccMat, TqLoad2Components.ccSub, TqLoad2Components.ccLabCompanyID, TqLoad2Components.ccEquipCompanyID, TqLoad2Components.ccMatCompanyID, TqLoad2Components.ccSubCompanyID, TqLoad2Components.PdlcID, TqLoad2Components.plItemID, TqLoad2Components.plMultiplier, TqLoad2Components.RsmMainLineNo, TqLoad2Components.PercentHrsOnOT, TqLoad2Components.MultiplierForOTRate, TqLoad2Components.AssMhrsPunit, TqLoad2Components.AssEqDaysPunit " \_

& "INTO TempComp2 " \_

& "FROM TqLoad2Components;")

qdf1.Execute

lngcount = 0

lngLevel = 0

ShowMeter "Importing Estimates - 70% " & lngcount & " of " & lngCountTot, 70

Do Until lngCountTot = lngcount

'you must repeat Components query since referential integrety prevents all records from being inserted at once.

'breaking the query into 2 parts speeds insert

Select Case True

Case lngLevel < 1

'part 1 - insert were component2ID is null

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqComponents " \_

& "SELECT TempComp2.\* " \_

& "FROM TempComp2 " \_

& "WHERE (((TempComp2.Component2ID) Is Null));")

qdf1.Execute

lngLevel = lngLevel + 1

lngcount = lngcount + qdf1.RecordsAffected

ShowMeter "Importing Estimates - 70% " & lngcount & " of " & lngCountTot, 70

Case Else

'part 2 - insert uninserted

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqComponents " \_

& "SELECT TempComp2.\* " \_

& "FROM TempComp2 LEFT JOIN TqComponents ON TempComp2.ComponentID = TqComponents.ComponentID " \_

& "WHERE (((TqComponents.ComponentID) Is Null));")

qdf1.Execute

lngcount = lngcount + qdf1.RecordsAffected

ShowMeter "Importing Estimates - 70% " & lngcount & " of " & lngCountTot, 70

'If qdf1.RecordsAffected = 0 Then Exit Do '<<this is slow method. See ImportPC.ImportEstmates2R80 for Do Until lngCountTot = lngCount method which is much faster

End Select

Loop

'delete temp table if it exists

On Error Resume Next

db.TableDefs.Refresh

db.TableDefs.Delete "TempComp2"

db.TableDefs.Refresh

On Error GoTo Errorhandler

ShowMeter "Importing Estimates - 85%", 85

'SubLines - don't copy SubLineID as we need to create new SubLineIDs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqSubLines ( CsiLineID, Item, Sub1, Sub2, Sub3, Sub4, Sub5, Sub6, Sub7, Sub8, Sub9, Sub10 ) " \_

& "SELECT TqLoad2SubLines.CsiLineID, TqLoad2SubLines.Item, TqLoad2SubLines.Sub1, TqLoad2SubLines.Sub2, TqLoad2SubLines.Sub3, TqLoad2SubLines.Sub4, TqLoad2SubLines.Sub5, TqLoad2SubLines.Sub6, TqLoad2SubLines.Sub7, TqLoad2SubLines.Sub8, TqLoad2SubLines.Sub9, TqLoad2SubLines.Sub10 " \_

& "FROM TqLoad2SubLines;")

qdf1.Execute

ShowMeter "Importing Estimates - 90%", 90

'Takeoff - don't copy TlineID as we need to create new TlineIDs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqTlines ( ElineID, Tqty1, Tqty2, Tqty3, Tqty4, Tqty5, Tcomment, TlineNo ) " \_

& "SELECT TqLoad2Tlines.ElineID, TqLoad2Tlines.Tqty1, TqLoad2Tlines.Tqty2, TqLoad2Tlines.Tqty3, TqLoad2Tlines.Tqty4, TqLoad2Tlines.Tqty5, TqLoad2Tlines.Tcomment, TqLoad2Tlines.TlineNo " \_

& "FROM TqLoad2Tlines;")

qdf1.Execute

ShowMeter "Importing Estimates - 92%", 92

'CsiDistributions - don't copy CsiDistributionID as we need to create new CsiDistributionIDs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqCsiDistributions ( CsiLineID, ZoneID, Distribution ) " \_

& "SELECT TqLoad2CsiDistributions.CsiLineID, TqLoad2CsiDistributions.ZoneID, TqLoad2CsiDistributions.Distribution " \_

& "FROM TqLoad2CsiDistributions;")

qdf1.Execute

ShowMeter "Importing Estimates - 95%", 95

'AdjustmentsP - MUST copy AdjustmentsPID as they have been adjusted above and match same field in AdjustDistributions

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqAdjustmentsP ( AdjustmentsPID, EstimateID, AdjustDescription, AmountA, PercentA, Manual, FormulaA, TotalLine, GTotalLine, LineNo, LineCalc, SFMethod, FormulaMethod, cCode, ccDescription, ccCompanyID ) " \_

& "SELECT TqLoad2AdjustmentsP.AdjustmentsPID, TqLoad2AdjustmentsP.EstimateID, TqLoad2AdjustmentsP.AdjustDescription, TqLoad2AdjustmentsP.AmountA, TqLoad2AdjustmentsP.PercentA, TqLoad2AdjustmentsP.Manual, TqLoad2AdjustmentsP.FormulaA, TqLoad2AdjustmentsP.TotalLine, TqLoad2AdjustmentsP.GTotalLine, TqLoad2AdjustmentsP.LineNo, TqLoad2AdjustmentsP.LineCalc, TqLoad2AdjustmentsP.SFMethod, TqLoad2AdjustmentsP.FormulaMethod, TqLoad2AdjustmentsP.cCode, TqLoad2AdjustmentsP.ccDescription, TqLoad2AdjustmentsP.ccCompanyID " \_

& "FROM TqLoad2AdjustmentsP;")

qdf1.Execute

ShowMeter "Importing Estimates - 97%", 97

'AdjustDistributions - don't copy AdjustDistributionID as we need to create new AdjustDistributionIDs

Set qdf1 = db.CreateQueryDef("", "INSERT INTO TqAdjustDistributions ( AdjustmentsPID, ZoneID, Distribution ) " \_

& "SELECT TqLoad2AdjustDistributions.AdjustmentsPID, TqLoad2AdjustDistributions.ZoneID, TqLoad2AdjustDistributions.Distribution " \_

& "FROM TqLoad2AdjustDistributions;")

qdf1.Execute

cleanup:

On Error Resume Next

'we are done duplicating - BE SURE to set SetDuplicatingRecordsToFalse in Sub that called this sub

DoCmd.Hourglass False

DoCmd.Close acForm, "Meter"

qdf1.Close

rst1.Close

rstDbList.Close

db.Close

Set qdf1 = Nothing

Set rst1 = Nothing

Set rstDbList = Nothing

Set db = Nothing

Exit Sub

Errorhandler:

Select Case Err.Number

' Case 3265

' 'table not found

' Resume Next

Case Else

MsgBox "ImportEstimates2R80 error " & Err.Number & " " & Err.Description

Resume Next

Resume cleanup

End Select

End Sub