Public Sub CopyEstimateFast(lngOldEstimateID As Long, lngNewEstimateID As Long)

'called by MainForm Command33 to duplicate an estimate

'This is similar to ProjectInfo Command421Now (which is an even more complex copying of selected records from a db)

Dim db As Database, qdf1 As QueryDef, lngRandom As Long, rst1 As Recordset

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

'This sub will duplicate this estimate by...:

'1) copy records into Load2 tables (referential Integrity & Cascase 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 "Duplicating Estimate", 10

'see if someone else is duplicating an estimate and pause until they are done

'otherwise record locking causes Duplicating to fail

CheckIfDuplicatingRecords

'Be sure to call SetDuplicatingRecordsToFalse at the end of the sub that called this sub (when records are done duplicating)

Set db = CurrentDb

'Empty the Load2 tables

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

& "FROM TqLoad2EstimateNos;")

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

'1) copy records into Load2 tables (referential Integrity & Cascase 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

'Copy EstimateNos

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

& "SELECT TqEstimateNos.\* " \_

& "FROM TqEstimateNos " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

'Copy Zone

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

& "SELECT TqZone.\* " \_

& "FROM TqEstimateNos INNER JOIN TqZone ON TqEstimateNos.EstimateID = TqZone.EstimateID " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

'Copy CsiLines

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

& "SELECT TqCsiLines.\* " \_

& "FROM TqEstimateNos INNER JOIN TqCsiLines ON TqEstimateNos.EstimateID = TqCsiLines.EstimateID " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

ShowMeter "Duplicating Estimate", 15

'Copy Elines

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

& "SELECT TqElines.\* " \_

& "FROM (TqEstimateNos INNER JOIN TqCsiLines ON TqEstimateNos.EstimateID = TqCsiLines.EstimateID) INNER JOIN TqElines ON TqCsiLines.CsiLineID = TqElines.CsiLineID " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

ShowMeter "Duplicating Estimate", 18

'Copy Components

Set rst1 = db.OpenRecordset("SELECT Count(TqComponents.ComponentID) AS CountOfComponentID " \_

& "FROM ((TqEstimateNos INNER JOIN TqCsiLines ON TqEstimateNos.EstimateID = TqCsiLines.EstimateID) INNER JOIN TqElines ON TqCsiLines.CsiLineID = TqElines.CsiLineID) INNER JOIN TqComponents ON TqElines.ElineID = TqComponents.ElineID " \_

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

lngCountTot = rst1!CountOfComponentID

' lngcount = 0

' Do Until lngCountTot = lngcount

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

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

' & "SELECT TqComponents.\* " \_

' & "FROM ((TqEstimateNos INNER JOIN TqCsiLines ON TqEstimateNos.EstimateID = TqCsiLines.EstimateID) INNER JOIN TqElines ON TqCsiLines.CsiLineID = TqElines.CsiLineID) INNER JOIN TqComponents ON TqElines.ElineID = TqComponents.ElineID " \_

' & "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

' qdf1.Execute

' lngcount = lngcount + qdf1.RecordsAffected

' Loop

'delete temp table if it exists

On Error Resume Next

db.TableDefs.Refresh

db.TableDefs.Delete "TempComp3"

db.TableDefs.Refresh

On Error GoTo Errorhandler

'make temp table

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

& "INTO TempComp3 " \_

& "FROM ((TqEstimateNos INNER JOIN TqCsiLines ON TqEstimateNos.EstimateID = TqCsiLines.EstimateID) INNER JOIN TqElines ON TqCsiLines.CsiLineID = TqElines.CsiLineID) INNER JOIN TqComponents ON TqElines.ElineID = TqComponents.ElineID " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

lngcount = 0

lngLevel = 0

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 TqLoad2Components " \_

& "SELECT TempComp3.\* " \_

& "FROM TempComp3 " \_

& "WHERE (((TempComp3.Component2ID) Is Null));")

qdf1.Execute

lngLevel = lngLevel + 1

lngcount = lngcount + qdf1.RecordsAffected

Case Else

'part 2 - insert uninserted

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

& "SELECT TempComp3.\* " \_

& "FROM TempComp3 LEFT JOIN TqLoad2Components ON TempComp3.ComponentID = TqLoad2Components.ComponentID " \_

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

qdf1.Execute

lngcount = lngcount + qdf1.RecordsAffected

End Select

Loop

'delete temp table if it exists

On Error Resume Next

db.TableDefs.Refresh

db.TableDefs.Delete "TempComp3"

db.TableDefs.Refresh

On Error GoTo Errorhandler

ShowMeter "Duplicating Estimate", 24

'Copy SubLines

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

& "SELECT TqSubLines.\* " \_

& "FROM (TqEstimateNos INNER JOIN TqCsiLines ON TqEstimateNos.EstimateID = TqCsiLines.EstimateID) INNER JOIN TqSubLines ON TqCsiLines.CsiLineID = TqSubLines.CsiLineID " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

'Copy Tlines

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

& "SELECT TqTlines.\* " \_

& "FROM ((TqEstimateNos INNER JOIN TqCsiLines ON TqEstimateNos.EstimateID = TqCsiLines.EstimateID) INNER JOIN TqElines ON TqCsiLines.CsiLineID = TqElines.CsiLineID) INNER JOIN TqTlines ON TqElines.ElineID = TqTlines.ElineID " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

ShowMeter "Duplicating Estimate", 27

'Copy CsiDistributions

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

& "SELECT TqCsiDistributions.\* " \_

& "FROM (TqEstimateNos INNER JOIN TqCsiLines ON TqEstimateNos.EstimateID = TqCsiLines.EstimateID) INNER JOIN TqCsiDistributions ON TqCsiLines.CsiLineID = TqCsiDistributions.CsiLineID " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

'Copy AdjustmentsP

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

& "SELECT TqAdjustmentsP.\* " \_

& "FROM TqEstimateNos INNER JOIN TqAdjustmentsP ON TqEstimateNos.EstimateID = TqAdjustmentsP.EstimateID " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

'Copy AdjustDistibutions

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

& "SELECT TqAdjustDistributions.\* " \_

& "FROM (TqEstimateNos INNER JOIN TqLoad2AdjustmentsP ON TqEstimateNos.EstimateID = TqLoad2AdjustmentsP.EstimateID) INNER JOIN TqAdjustDistributions ON TqLoad2AdjustmentsP.AdjustmentsPID = TqAdjustDistributions.AdjustmentsPID " \_

& "WHERE (((TqEstimateNos.EstimateID)=" & lngOldEstimateID & "));")

qdf1.Execute

ShowMeter "Duplicating Estimate", 30

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

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

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

'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 these 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

'EstimateNos

'be sure copy field = False

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

& "SET TqLoad2EstimateNos.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 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

Set rst1 = db.OpenRecordset("SELECT TqLoad2EstimateNos.EstimateID " \_

& "FROM TqLoad2EstimateNos;", dbOpenDynaset)

lngNewEstimateID = rst1!EstimateID

'Zone

'be sure Load2Zone's copy field = False

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

& "SET TqLoad2Zone.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 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 "Duplicating Estimate", 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

ShowMeter "Duplicating Estimate", 40

'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

'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

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 "Duplicating Estimate", 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

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 "Duplicating Estimate", 50

'2e) copy the load2 tables back into normal tables

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

'copied back to Permanent Tables

'you must list each field because you must eliminate copying copy & select fields that do not 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, 'Copy of ' & [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

'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

'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

'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 "Duplicating Estimate", 70

'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 "Duplicating Estimate", 90

'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

'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

'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

'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

'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

'Empty the Load2 tables

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

& "FROM TqLoad2EstimateNos;")

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

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

db.Close

Set qdf1 = Nothing

Set rst1 = Nothing

Set db = Nothing

Exit Sub

Errorhandler:

Select Case Err.Number

Case 3265

'table not found

Resume Next

Case Else

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

Resume cleanup

End Select

End Sub