Twice is Sometimes Nicer, but Not Always

I was asked by a manufacturing operations manager to help reduce his inventory costs by identifying custom made products, currently in inventory and with no scheduled delivery date, that could be substituted for items scheduled for future production. The idea was to avoid producing duplicate items when suitable ones already exist. When a substitution is made, the initial item is not re-scheduled for production without customer confirmation, leaving, at any given time, at most one item in inventory for a given configuration. The product was a particular type of cylinder and the constraining feature various holes formed at arbitrary locations into the cylinder walls. The geometry of cylinders in inventory and those scheduled for production were known, as was the delivery schedule and customer demand. The difficulty in matching demand to existing items was in identifying all possible combinations of pairs of holes (a pair consists of one hole from the demand configuration and one from a potential matching cylinder in inventory). Imagine a demand configuration and an inventoried item, both with three holes – three pairs are required and there are six distinct groupings: 1-1, 2-2, 3-3; 1-1, 2-3, 3-2; 1-2, 2-1, 3-3; 1-2, 2-3, 3-1; 1-3, 2-1, 3-2; and 1-3, 2-2, 3-1. The general rule for the number of pair groups is *the number of distinct combinations of required holes taken the number of inventoried holes at a time times the number of permutations of inventoried holes taken that number at a time*, which is

$$\binom{N_r}{N_i} \mathrm{P}_{N_i}^{N_i} = \frac{\mathrm{P}_{N_i}^{N_r}}{N_i!} N_i! = \mathrm{P}_{N_i}^{N_r} = N_r(N_r-1)(N_r-2)\ldots(N_r-N_i+1)$$

where N_r = the number of required (demand) holes and N_i = the number of holes in a potential inventoried item. Note that in this solution N_r is required to be greater than or equal to N_i since additional holes can be fabricated into a cylinder, but not removed, which conveniently accommodates the mathematical requirement that the top value in the combination expression be greater than the bottom value, that is, in $\binom{n}{m}$, n must be greater than or equal to m. I found enumerating the pairs to be easy (from the above formula) while generating the actual pairings to be much more challenging. In Probability, we are typically interested in quantities only, the number of combinations or distinct events, but in optimization we need a complete description of every population member so that each can be evaluated and optimal ones selected. One further complication in our problem is that by "matching" it is meant that the difference in hole position, measured in degrees from a selected reference point, for each pair is within a specified tolerance. And, of course, multiple deviations are obtained by choosing different reference points – a deviation of 0 is obtained by setting one inventoried hole location and one demand hole location to reference points of 0 degrees. The deviations for remaining holes are dependent on which holes are matched and matching is done with the objective of all deviations falling within the specified tolerance.

Following is the inventoried hole to requested hole combination setup SQL procedure.

```
ALTER proc [dbo].[InventoryLocatorHoleCombinationSetup] @nReq tinyint, @nInv tinyint as

-- Generate list of all unique combinatons of required holes to holes in inventoried
-- items
-- nReq = number of requested holes, nInv = holes of item in inventory, note nReq>=nInv
-- Results are saved in table InventoryLocatorHoleCombinations which must exist and have
-- columns
-- nReqHoles tinyint, nInvHoles tinyint, ReqCombination (smallint), InvCombination
-- (smallint), ReqHole (tinyint), InvHole (tinyint)
-- On return it will contain all unique combinations of required and inventory holes
-- Each group of rows identified by ReqCombination, InvCombination is one unique
-- combination of holes

set nocount on

-- Declare tables to group req/inv combination pairings
create table #a(h tinyint)
create table #ReqHoles(CombinationID int identity)
create table #InvHoles(CombinationID int identity)
```

```
declare @i tinyint, @j tinyint, @k tinyint, @sqltext varchar(1023), @sqltext2 varchar(1023)

-- Dynamically construct tables to contain distinct pairs of req holes to be mated to
-- inv holes
-- Note:  req hole combinations have h1<h2<...<hnReqHole
select @j=1
while(@j<=@nInv)
  begin
    select @sqltext='alter table #ReqHoles add h' + convert(varchar(2),@j) + ' tinyint'
    exec(@sqltext)
    select @sqltext='alter table #InvHoles add h' + convert(varchar(2),@j) + ' tinyint'
    exec(@sqltext)
    select @j=@j+1
  end

-- Setup unique combinations of request hole numbers
select @i=1
while(@i<=@nReq)
  begin
    insert into #a values(@i)
    select @i=@i+1
  end
select @sqltext='select * from #a a1', @sqltext2='where 1=1', @j=2
while(@j<=@nInv)
  begin
    select @sqltext=@sqltext + ' cross join #a a' + convert(varchar(2),@j),
           @sqltext2=@sqltext2 + ' and a' + convert(varchar(2),@j-1) + '.h<a' +
           convert(varchar(2),@j) + '.h'
    select @j=@j+1
  end
exec('insert into #ReqHoles ' + @sqltext + ' ' + @sqltext2)

-- Setup all combinations of inventory hole numbers
delete from #a
select @i=1
while(@i<=@nInv)
  begin
    insert into #a values(@i)
    select @i=@i+1
  end
select @sqltext='select * from #a a1', @j=2
while(@j<=@nInv)
  begin
    select @sqltext=@sqltext + ' cross join #a a' + convert(varchar(2),@j)
    select @j=@j+1
  end
exec('insert into #InvHoles ' + @sqltext)
-- Omit combinations with one or more holes specified more than once (since each can be
-- mated to at most one requested hole)
select @j=1, @sqltext2='delete #InvHoles where 1=0'
while(@j<=@nInv)
  begin
    select @k=1
    while(@k<@j)
      begin
        select @sqltext2=@sqltext2+' or
               h'+convert(varchar(2),@k)+'=h'+convert(varchar(2),@j)
        select @k=@k+1
      end
    select @j=@j+1
  end
exec(@sqltext2)

-- Save all combination groups, one req-inv hole combination per row, grouped by
-- ReqCombination-InvCombination
-- Note the repeated cross-joining of req to inv tables.  This is a little inefficient
-- and could be avoided by first saving
-- the cross join results to yet another temp table, but both nReq and nInv are expected
-- to be less than 5, which
-- yields small tables, and the processing saved by constructing an iterated 'create
```

```
-- table' statement to be later executed
-- is expected to be negligible.
select @j=1
while(@j<=@nInv)
  begin
    select @sqltext='insert into InventoryLocatorHoleCombinations(nReqHoles, nInvHoles,
                     ReqCombination, InvCombination, ReqHole, InvHole) '+
           'select '+convert(varchar(2),@nReq)+','+convert(varchar(2),@nInv)+',
            #ReqHoles.CombinationID, #InvHoles.CombinationID, #ReqHoles.h'+
            convert(varchar(2),@j)+', #InvHoles.h'+convert(varchar(2),@j)+' from
            #ReqHoles cross join #InvHoles'
    exec(@sqltext)
    select @j=@j+1
  end

--select * from #ReqHoles cross join #InvHoles

drop table #a
drop table #ReqHoles
drop table #InvHoles
```

Following is the Production Scheduling interface SQL procedure. This generates the trigger report that instructs the scheduler to investigate possible substitution.

```
ALTER proc [dbo].[InventoryItemLocator] @CPCItemID varchar(20), @ReqHoles varchar(1000),
@ReqDate smalldatetime, @NoInvOrderID smallint=0, @MfgSchedSearch varchar(5)='no',
@InvCodeAlternate varchar(5)='yes', @HeightTolerance real=99,
@NumberOfHolesIdentical varchar(5)='no', @HoleTypeIdentical varchar(5)='no',
@HoleSizeIdentical varchar(5)='no', @HoleLocAngleTolerance real=10,
@HoleLocVertToleranceAbove real=24, @HoleLocVertToleranceBelow real=24,
@DaysInInventory smallint=120, @SearchInactiveStock varchar(5)='yes' as

-- Identify inactive or aging in-stock items that can be substituted for items items on
-- mfg schedule on requested date

-- Parameters:

-- @CPCItemID                - Single item to search for inventory substitutes, use
--                             configuration parameters from Item Master and @ReqHoles
--                           - Empty skips single item parameter search
-- @ReqHoles                 - Required holes in single (@CPCItemID) item search –
--                             valid only when @CPCItemID not empty
--                           - Format '(HoleType, xDim, yDim, Angle, Up), (HoleType,
--                             xDim, yDim, Angle, Up), etc.'
--                           - xDim, yDim in feet; Angle in degrees; Up in inches
-- @ReqDate                  - Date that substitute is required on
--                           - Used to determine age in inventory (age=@ReqDate-
--                             InventoryDate)
-- @MfgSchedSearch           - Yes: search for possible inventory substitutes for all
--                             items scheduled on @ReqDate
-- @NoInvOrderID             - Order ID of custom BOM items to search for possible
--                             substitutes, only those with inventory qty <= 0 as of
--                             @ReqDate
--                           - 0 skips this search
-- @InvCodeAlternate         - Yes: allow alternate inv-codes as listed in
--                             InventoryLocatorInvCodeAlt table (view)
--                           - No: Limit search to inventory items with inv code
--                             identical to requested item(s)
-- @HeightTolerance          - Limit search to items with height within requested item
--                             height +- @HeightTolerance (in feet)
-- @NumberOfHolesIdentical   - Yes: limit search to inventory items with number of
--                             holes identical to number of required holes
--                           - Note that (as of development - in 2006) returned
--                             inventory items will never have more than the requested
--                             number of holes
-- @HoleTypeIdentical        - Yes: limit search to inventory items with hole types
--                             identical to required hole types
-- @HoleSizeIdentical        - Yes: limit search to inventory items with hole sizes
--                             identical to required hole sizes
```

```
-- @HoleLocAngleTolerance     - Return inventoried items with all hole locations within
--                              this number of degrees of corresponding requested
--                              locations
-- @HoleLocVertToleranceAbove - Return inventoried items with all hole locations within
--                              this number of inches above corresponding requested
--                              locations
-- @HoleLocVertToleranceBelow - Return inventoried items with all hole locations within
--                              this number of inches below corresponding requested
--                              locations
-- @DaysInInventory           - Limit search to items in active inventory for at least
--                              this number of days (0 eliminates this criterion)
-- @SearchInactiveStock       - Yes: search inventory transfer log for similar items
--                              remaining in inactive stock

-- No locks on any tables (uncommitted transactions also read)
set transaction isolation level read uncommitted
set nocount on

declare @ReqInvPair table(ReqBOMSerialNo int, InvBOMSerialNo int,
        nReqHoles tinyint, nInvHoles tinyint, ReqCombination int, InvCombination int,
        ReqHoleNumber tinyint, InvHoleNumber tinyint, InvStatus varchar(15),
        DaysInInventory real, ReqHoleType varchar(10), ReqHoleXDim real,
        ReqHoleYDim real, ReqHoleLoc real, ReqHoleUp real, InvHoleType varchar(10),
        InvHoleXDim real, InvHoleYDim real, InvHoleLoc real, InvHoleUp real)
declare @Holes table(HoleType varchar(10), xDim real, yDim real, Loc real, Up real,
        RecID int identity)
declare @i int, @j int, @nHoles int, @p1 smallint, @p2 smallint, @q1 smallint,
        @q2 smallint, @HolePar varchar(1000), @HoleType varchar(10), @xDim real,
        @yDim real, @Loc real, @Up real

-- Parse supplied holes when specific item ID requested
-- Format is '(HoleType, xDim, yDim, Angle, Up), (HoleType, xDim, yDim, Angle, Up),
-- etc.'
-- xDim, yDim, and Up are assumed to be in inches, Angle in degrees
if(@CPCItemID<>'')
  begin
    -- Locate first set of parameters [between "(" and ")"]
    select @p1=charindex('(',@ReqHoles), @p2=charindex(')',@ReqHoles)
    while(@p1>0 and @p2>@p1)
      begin
        -- Extract parameters - leave trailing ) to delimit final parameter
        select @HolePar=substring(@ReqHoles,@p1+1,@p2-@p1)
        -- Locate first comma and extract hole type from 1st pos to just before comma
        select @q1=1, @q2=charindex(',',@HolePar), @HoleType=
               substring(@HolePar,@q1,@q2-@q1)
        -- Locate next comma and extract x-dim
        select @q1=@q2+1, @q2=charindex(',',@HolePar,@q1),
               @xDim=convert(real,substring(@HolePar,@q1,@q2-@q1))
        -- Locate next comma and extract y-dim
        select @q1=@q2+1, @q2=charindex(',',@HolePar,@q1),
               @yDim=convert(real,substring(@HolePar,@q1,@q2-@q1))
        -- Locate next comma and extract angle-loc
        select @q1=@q2+1, @q2=charindex(',',@HolePar,@q1),
               @Loc=convert(real,substring(@HolePar,@q1,@q2-@q1))
        -- Locate ) and extract up-dimension
        select @q1=@q2+1, @q2=charindex(')',@HolePar,@q1),
               @Up=convert(real,substring(@HolePar,@q1,@q2-@q1))
        -- Save parameters
        if(@HoleType is not null and @xDim is not null and @yDim is not null
            and @Loc is not null and @Up is not null)
          insert into @Holes values(@HoleType, @xDim, @yDim, @Loc, @Up)
        -- Locate next set of parameters [between next ( and )]
        select @p1=charindex('(',@ReqHoles,@p2+1), @p2=charindex(')',@ReqHoles,@p1+1)
      end
    select @nHoles=count(1) from @Holes
  end

-- Get max number of req holes in scheduled items, when requested
-- Retain greater of max scheduled holes and static supplied holes
if(@MfgSchedSearch='yes')
  select @nHoles = case when(max(n)>isnull(@nHoles,0))then max(n) else @nHoles end
```

```
  from   ( select   count(1) as n
           from     MfgSchedule join BOMFeature on
MfgSchedule.BOMSerialNo=BOMFeature.BOMSerialNo and Feature='hole'
           where    MfgSchedDate=@ReqDate and @CPCItemID=''
           group by MfgSchedule.BOMSerialNo
         ) a

-- Get max number of req holes in custom BOM items for requested order that are not in
inventory as of requested date
-- Retain greater of non-inventory BOM holes and previous counts
if(@NoInvOrderID>0)
  select @nHoles = case when(max(n)>isnull(@nHoles,0))then max(n) else @nHoles end
  from   ( select   count(1) as n
           from     BOMItemInventoryQty join BOM on
BOMItemInventoryQty.BOMSerialNo=BOM.SerialNo
                    join ItemMaster on BOM.CPCItemID=ItemMaster.CPCItemID
                    join InvCodes on ItemMaster.InvCode=InvCodes.Code
           where    BOM.OrderID=@NoInvOrderID and ItemMaster.ItemType='mfg' and
InvCodes.Inventory=0
                    and BOMItemInventoryQty.InventoryDate<=@ReqDate
           group by BOMSerialNo
           having   sum(MfgReceiptQty-ShipQty+OtherQty)<0
         ) a

-- Populate pairing table to accomodate up to n,n req,inv holes
-- At time of development (2006) inv holes are restricted to <= req holes (LOOK INTO
-- WHETHER > WILL FUNCTION CORRECTLY)
-- Impose arbitrary limit of 8 holes, since the chances of finding a matching 9 hole or
-- above item in inventory is (what?) small?
-- InventoryLocatorHoleCombinations records increases greatly as the number of holes
-- increases
-- In fact, given n holes, the number of records is sum(i=1 to n)[i*n!/(n-i)!]
-- So, for n=9, the number of required records is 7,891,281; for n=10, 88,786,910; for
-- n=11, over 1 billion; and so forth
-- A practical limit is implied by the log file filling when generating records for
-- larger numbers of holes

if(isnull(@nHoles,0)>0)
  begin
    select @i=1
    while(@i<=@nHoles and @i<9)
      begin
        select @j=1
        while(@j<=@i)
          begin
            if(not exists(select *
                          from InventoryLocatorHoleCombinations
                          where nReqHoles=@i and nInvHoles=@j))
              exec InventoryLocatorHoleCombinationSetup @i, @j
            select @j=@j+1
          end
        select @i=@i+1
      end
  end

-- Match demand with candidate substitute items in inventory
-- Filter by end item dimensions, number of holes, and hole-height tolerance
-- Hole type and angular tolerance comparison is done after mapping of req holes to inv
-- holes
insert into @ReqInvPair
select Req.BOMSerialNo, Inv.BOMSerialNo, Req.nHoles, Inv.nHoles,
       isnull(InventoryLocatorHoleCombinations.ReqCombination,0),
       isnull(InventoryLocatorHoleCombinations.InvCombination,0),
       isnull(ReqHoleNumber.HoleNumber,0), isnull(InvHoleNumber.HoleNumber,0),
       Inv.Status, Inv.DaysInInventory,
       ReqHole.HoleType, ReqHole.xDim, ReqHole.yDim, ReqHole.Loc, ReqHole.Up,
       InvHole.FeatureDetail, InvHole.xDim, InvHole.yDim, InvHole.Loc, InvHole.Up
from   ( -- Demand
         select BOMSerialNo, ProdType, DimX, DimY, SubAsmCode, InvCode, Height, nHoles
         from ( -- Scheduled items when requested
                select MfgSchedule.BOMSerialNo, BOM.CPCItemID, isnull(nHoles,0)
```

```
                as nHoles
        from    MfgSchedule join BOM on MfgSchedule.BOMSerialNo=BOM.SerialNo
                -- Count holes in req items
                left join ( select   BOMSerialNo, count(1) as nHoles
                            from     BOMFeature
                            where    Feature='hole'
                            group by BOMSerialNo
                            having   count(1)<9
                          ) BOMFeature on
                            MfgSchedule.BOMSerialNo=BOMFeature.BOMSerialNo
        where   @MfgSchedSearch='yes' and MfgSchedDate=@ReqDate
                and SchedStatus<>'cancel'
                -- Exclude any with too many holes
                and isnull(nHoles,0)<9
        union
        -- Individual requested item
        select 0 as BOMSerialNo, @CPCItemID, count(1) as nHoles
        from   @Holes
        where  @CPCItemID<>''
        union
        -- Custom items on requested order with inventory qty <= 0 as of
        -- @ReqDate
        select   BOM.SerialNo, BOM.CPCItemID, isnull(nHoles,0) as nHoles
        from     BOM join BOMItemInventoryQty
                 on BOM.SerialNo=BOMItemInventoryQty.BOMSerialNo
                 join ItemMaster on BOM.CPCItemID=ItemMaster.CPCItemID
                 join InvCodes on ItemMaster.InvCode=InvCodes.Code
                 -- Count holes in req items
                 left join ( select   BOMSerialNo, count(1) as nHoles
                             from     BOMFeature
                             where    Feature='hole'
                             group by BOMSerialNo
                             having   count(1)<9
                           ) BOMFeature on BOM.SerialNo=BOMFeature.BOMSerialNo
        where    BOM.OrderID=@NoInvOrderID and ItemMaster.ItemType='mfg'
                 and InvCodes.Inventory=0
                 and BOMItemInventoryQty.InventoryDate<=@ReqDate
                 -- Exclude any with too many holes
                 and isnull(nHoles,0)<9
        group by BOM.SerialNo, BOM.CPCItemID, nHoles
        having   sum(MfgReceiptQty-ShipQty+OtherQty)<0
      ) d join ItemMaster on d.CPCItemID=ItemMaster.CPCItemID
) Req join
( -- Candidate substitute items
  select i.BOMSerialNo, ProdType, DimX, DimY, SubAsmCode, InvCode, Height,
         Status, DaysInInventory, isnull(nHoles,0) as nHoles
  from   ( -- All items in inventory as of requested date
           select   BOMSerialNo, 'Stock' as Status,
                    max(case when(MfgReceiptQty>0 or OtherQty>0)then
                         datediff(d,InventoryDate,@ReqDate) else 0 end)
                    as DaysInInventory
           from     BOMItemInventoryQty
           where    InventoryDate<=@ReqDate
           group by BOMSerialNo
           having   sum(MfgReceiptQty-ShipQty+OtherQty)>0
                    -- Apply age constraint if requested
                    and (max(case when(MfgReceiptQty>0 or OtherQty>0)then
                                  datediff(d,InventoryDate,@ReqDate)
                             else 0
                             end)>=@DaysInInventory or @DaysInInventory=0)
           union -- distinct
           -- Inactive inventory - mfg items moved into inactive stock but not
           -- yet removed
           select distinct a.FromBOMSerialNo, 'InactiveStock' as Status, 0
                  as DaysInInventory
           from   InventoryTransferLog a left join InventoryTransferLog b
                  on b.TransType='reactive'
                  and a.FromBOMSerialNo=b.FromBOMSerialNo
                  and a.TransDate<=b.TransDate
           where  a.TransType='inactive' and b.TransID is null
                  and @SearchInactiveStock='yes'
```

```
        ) i join BOM on i.BOMSerialNo=BOM.SerialNo
        join ItemMaster on BOM.CPCItemID=ItemMaster.CPCItemID
        -- Count holes in req items
        left join ( select   BOMSerialNo, count(1) as nHoles
                    from     BOMFeature
                    where    Feature='hole'
                    group by BOMSerialNo
                  ) BOMFeature on i.BOMSerialNo=BOMFeature.BOMSerialNo
) Inv on
-- Restrict to compatible product dimensions and configurations
Req.ProdType=Inv.ProdType and Req.SubAsmCode=Inv.SubAsmCode
-- Allow substitute inv codes if instructed
and (Req.InvCode=Inv.InvCode
     or Inv.InvCode in(select InvCodeAlt
                       from    InventoryLocatorInvCodeAlt
                       where InvCode=Req.InvCode) and @InvCodeAlternate='yes')
and Req.DimX=Inv.DimX and Req.DimY=Inv.DimY
and Inv.Height between Req.Height-@HeightTolerance
and Req.Height+@HeightTolerance
and (Req.nHoles=Inv.nHoles or Req.nHoles>Inv.nHoles
and @NumberOfHolesIdentical<>'yes')
-- Get holes for req items (use left joins on all hole related joins in case an
-- item has no holes)
left join ( select BOMSerialNo, FeatureSerialNo, FeatureDetail as HoleType, xDim,
                   yDim, Loc, Up
            from   BOMFeature
            where  Feature='hole' and @CPCItemID=''
            union
            select 0 as BOMSerialNo, RecID, HoleType, xDim, yDim, Loc, Up
            from   @Holes
          ) ReqHole on Req.BOMSerialNo=ReqHole.BOMSerialNo
left join ( -- Number holes for matching - 1, 2, 3, etc.  These correspond with
            -- hole numbers IDs matching patterns (below).
            -- Hole number is the number of records, by BOMSerialNo, with ID
            -- equal to or below primary (left hand record) ID
            -- Note that FeatureSerialNo is unique
            select   a.FeatureSerialNo, count(1) as HoleNumber
            from     BOMFeature a left join BOMFeature b
                     on a.BOMSerialNo=b.BOMSerialNo
                     and a.FeatureSerialNo>=b.FeatureSerialNo
                     and b.Feature='hole'
            where    a.Feature='hole' and @CPCItemID=''
            group by a.FeatureSerialNo
            union
            select   a.RecID, count(1) as HoleNumber
            from     @Holes a join @Holes b on a.RecID>=b.RecID
            group by a.RecID
          ) ReqHoleNumber on
            ReqHole.FeatureSerialNo=ReqHoleNumber.FeatureSerialNo
-- Get holes for inv items
left join BOMFeature InvHole on Inv.BOMSerialNo=InvHole.BOMSerialNo
and InvHole.Feature='hole'
left join ( -- Number holes for matching - 1, 2, 3, etc.  These correspond with
            -- hole numbers IDs matching patterns (below).
            -- Hole number is the number of records, by BOMSerialNo, with ID
            -- equal to or below primary (left hand record) ID
            -- Note that FeatureSerialNo is unique
            select   a.FeatureSerialNo, count(1) as HoleNumber
            from     BOMFeature a left join BOMFeature b
                     on a.BOMSerialNo=b.BOMSerialNo
                     and a.FeatureSerialNo>=b.FeatureSerialNo
                     and b.Feature='hole'
            where    a.Feature='hole'
            group by a.FeatureSerialNo
          ) InvHoleNumber
            on InvHole.FeatureSerialNo=InvHoleNumber.FeatureSerialNo
-- Get hole matching patterns and pair req holes with inv holes
left join InventoryLocatorHoleCombinations
on Req.nHoles=InventoryLocatorHoleCombinations.nReqHoles
and Inv.nHoles=InventoryLocatorHoleCombinations.nInvHoles
and ReqHoleNumber.HoleNumber=InventoryLocatorHoleCombinations.ReqHole
```

```
        and InvHoleNumber.HoleNumber=InventoryLocatorHoleCombinations.InvHole

-- Now, filter by hole constraints and calculate "closeness" of holes in inv items to
-- corresponding holes in req items
-- Note that angular deviations are calculated after req-inv hole mapping because hole
-- locations (degrees of rotation) have to be
-- adjusted to an offset from a basis; one req hole, the "sync" hole, is set to 0 deg
-- and the remaining req holes are adjusted by the
-- same amount as the sync hole - similarly, the inv holes are adjusted by using the
-- hole mapped to the req sync hole as basis;
-- so that angular comparisons (req to inv) are valid - all referenced from 0 degrees
-- Any differences in paired angles greater than 360 deg are converted to their
-- corresponding difference angle between 0 and 360 deg
-- Then, differences greater than 180 deg are converted to the corresponding angle at
-- 360 - the difference
-- This gives the minimum angular difference between holes
-- Note that each req hole, in turn, is used as the sync hole (with corresponding inv
-- hole used as inv sync hole)
-- This way, all possible deviations of req-inv mapped holes are measured, selecting a
-- different req-inv pair as reference angle for each
-- Consider choosing one req hole and one inventory hole as your "zero" degree holes
-- The deviations of remaining holes are determined by which hole you chose - choose
-- another pair and you get different deviations
-- By examining every possible pair of req-inv "zero" or "sync" holes we are able to
-- measure all deviation sets and choose the one
-- with smallest total deviation
select   @ReqDate as ReqDate, ReqBOMSerialNo, InvBOMSerialNo,
         ReqBOM.OrderID as ReqOrderID, ReqBOM.ItemLine as ReqItemLine,
         ReqBOM.ItemSubLine as ReqItemSubLine,
         ReqBOM.CPCItemID as ReqItemID, ReqItem.Description as ReqItemDesc,
         ReqBOM.Layout as ReqLayout,
         ReqMfgItemWorkcenter.Workcenter as Workcenter, ReqWorkcenter.Description
         as WorkcenterDesc, InvBOM.OrderID as InvOrderID, InvBOM.ItemLine
         as InvItemLine, InvBOM.ItemSubLine as InvItemSubLine,
         InvBOM.CPCItemID as InvItemID, InvItem.Description as InvItemDesc,
         InvBOM.Layout as InvLayout, InvStatus, DaysInInventory,
         min(abs(ReqItem.Height-InvItem.Height)) as ItemHeightDiff,
         min(abs(nReqHoles-nInvHoles)) as NumberHolesDiff,
         min(HoleTypeDiff) as HoleTypeDiff, min(HoleDimDiff) as HoleDimDiff,
         min(HoleUpDiff) as HoleUpDiff,
         isnull(ReqWorkcenter.AppearanceOrder,9999) as AppearanceOrder, 0 as Priority
         --sum(case when(aDiff>180.)then 360.-aDiff else aDiff end) as aDiff,
         -- Calculate sums of (flat-plane Cartesian) distances between paired requested-
         -- inventoried holes
         -- Sum of squared differences of hole locations, in inches, adjusted for sync
         -- hole angle
         -- Calculate proportion of circumference of round finished item, perimeter of
         -- rectangular finished item, between req-inv angles
         -- Round: square_root_of [ [pi*Finished_Diameter*(|i_angle-r_angle|)/360*12]**2
         -- + (difference in requested and inventoried Up dims)**2 ]
         -- Rectangular: square_root_of [ [2*(l+w)*(i_angle-r_angle)/360*12]**2
         -- (difference in requested and inventoried Up dims)**2 ]
         -- The up dims are adjusted to hole center - 1/2 hole dia for round holes, 1/2
         -- y dimension for rectangular
from     ( -- Calculate angular differences once, retain 2 decimal decimal digits
           select ReqSync.ReqBOMSerialNo, ReqSync.InvBOMSerialNo, ReqSync.InvStatus,
                  ReqSync.DaysInInventory, ReqSync.nReqHoles, ReqSync.nInvHoles,
                  ReqSync.ReqCombination, ReqSync.InvCombination,
                  -- Note that modulo is not available for real numbers so convert to
                  -- magnified integer, mod it, then scale back
                  abs(convert(int,100*(ReqInv.ReqHoleLoc-ReqSync.ReqHoleLoc-
                  ReqInv.InvHoleLoc+ReqSync.InvHoleLoc))%36000/100.) as aDiff,
                  HoleTypeDiff, HoleDimDiff, HoleUpDiff
           from   ( -- Get req-inv end item pairs with hole combinations that satisfy
                    -- hole type, hole dimension, and hole up location constraints
                    select   ReqBOMSerialNo, InvBOMSerialNo, nReqHoles, nInvHoles,
                             ReqCombination, InvCombination,
                             sum(case when(ReqHoleType<>InvHoleType)then 1 else 0 end)
                             as HoleTypeDiff,
                             sum(case when(ReqHoleXDim<>InvHoleXDim or
                                           ReqHoleYDim<>InvHoleYDim)then 1 else 0 end)
```

```
                        as HoleDimDiff,
                        sum(abs(InvHoleUp-ReqHoleUp)) as HoleUpDiff
                from    @ReqInvPair
                group by ReqBOMSerialNo, InvBOMSerialNo, nReqHoles, nInvHoles,
                        ReqCombination, InvCombination
                having  (sum(case when(ReqHoleType<>InvHoleType)then 1 else 0
                                end)=0 or @HoleTypeIdentical<>'yes')
                        and (sum(case when(ReqHoleXDim<>InvHoleXDim or
                        ReqHoleYDim<>InvHoleYDim)then 1 else 0 end)=0 or
                        @HoleSizeIdentical<>'yes')
                        and sum(case when(InvHoleUp<ReqHoleUp-
                        @HoleLocVertToleranceBelow or
                        InvHoleUp>ReqHoleUp+@HoleLocVertToleranceAbove)then 1
                        else 0 end)=0
            ) FiltPair
            join @ReqInvPair ReqSync
            on FiltPair.ReqBOMSerialNo=ReqSync.ReqBOMSerialNo
            and FiltPair.InvBOMSerialNo=ReqSync.InvBOMSerialNo
            and FiltPair.nReqHoles=ReqSync.nReqHoles
            and FiltPair.nInvHoles=ReqSync.nInvHoles
            and FiltPair.ReqCombination=ReqSync.ReqCombination
            and FiltPair.InvCombination=ReqSync.InvCombination
            join @ReqInvPair ReqInv
            on ReqSync.ReqBOMSerialNo=ReqInv.ReqBOMSerialNo
            and ReqSync.InvBOMSerialNo=ReqInv.InvBOMSerialNo
            and ReqSync.nReqHoles=ReqInv.nReqHoles
            and ReqSync.nInvHoles=ReqInv.nInvHoles
            and ReqSync.ReqCombination=ReqInv.ReqCombination
            and ReqSync.InvCombination=ReqInv.InvCombination
        ) SyncPair left join BOM ReqBOM on SyncPair.ReqBOMSerialNo=ReqBOM.SerialNo
          and SyncPair.ReqBOMSerialNo>0
        join ItemMaster ReqItem on SyncPair.ReqBOMSerialNo>0
        and ReqBOM.CPCItemID=ReqItem.CPCItemID or ReqItem.CPCItemID=@CPCItemID
        left join (select   CPCItemID, min(Workcenter) as Workcenter
                from    MfgItemWorkcenter
                group by CPCItemID) ReqMfgItemWorkcenter
        -- Investigate why there are two on clauses for the following join
        left join Workcenter ReqWorkcenter
        on ReqMfgItemWorkcenter.Workcenter=ReqWorkcenter.Workcenter
        on ReqItem.CPCItemID=ReqMfgItemWorkcenter.CPCItemID
        join BOM InvBOM on SyncPair.InvBOMSerialNo=InvBOM.SerialNo
        join ItemMaster InvItem on InvBOM.CPCItemID=InvItem.CPCItemID
group by ReqBOMSerialNo, InvBOMSerialNo,
        ReqBOM.OrderID, ReqBOM.ItemLine, ReqBOM.ItemSubLine,
        ReqBOM.CPCItemID, ReqItem.Description, ReqBOM.Layout,
        ReqMfgItemWorkcenter.Workcenter, ReqWorkcenter.Description,
        InvBOM.OrderID, InvBOM.ItemLine, InvBOM.ItemSubLine,
        InvBOM.CPCItemID, InvItem.Description, InvBOM.Layout, InvStatus,
        DaysInInventory,
        ReqWorkcenter.AppearanceOrder
having  -- Compare hole locations after adjustment for sync-angle, omit any with at
        -- least one inv angle outside of specified tolerance
        sum(case when(case when(aDiff>180.)then 360.-aDiff else aDiff
end>@HoleLocAngleTolerance)then 1 else 0 end)=0
```