Oracle-SQL Server Integration

The following is a report source that integrated SQL data from a remote Oracle EBS (Enterprise Business Suite) server with local ERP (Enterprise Resource Planning) data to produce a record source for LEED (Leadership in Energy and Efficiency Design) reporting of construction projects. Of note is the proper interpretation and conversion of Oracle sales and inventory units of measure. Also, note the construction of dynamic queries, tailored to the user supplied reporting requirements, to retrieve and stage remote data by executing Oracle SQL statements through a linked server and openquery(). This enabled efficient generation of a large cross-server data set, with necessary data available at each processing step. Note, also the geospacial calculations (near “Accumulate summary information, “ toward the end) that supply job site distances as a simple SQL column.

```
create proc OracleLEEDData @OracleOrderID varchar(100), @IgnoreMissingBOM varchar(5)='no', @SubstitutePlantBOM varchar(5)='no',
           @IncludeCastings varchar(5)='no' as

-- Retrieve Oracle bill of material data for requested sales order(s)
-- If multiple orders requested, separate individual order IDs by a comma

declare @sqltext varchar(8000)

create table #SalesLine(OrderID int, LineID int, OrgID int, ItemID varchar(50), ItemCategory varchar(50), QtyReq real, UOM varchar(20),
            QtyShipped real, Price real, ItemPrimaryUOM varchar(20), ItemWeight real, ItemWeightUOM varchar(20))
create table #BOM(LineID int, OrgID int,
            ItemIDLevel1 varchar(50), ItemCategoryLevel1 varchar(50), QtyReqLevel1 real,
            UOMLevel1 varchar(20), WeightLevel1 real, WeightUOMLevel1 varchar(20),
            ItemIDLevel2 varchar(50), ItemCategoryLevel2 varchar(50), QtyReqLevel2 real,
            UOMLevel2 varchar(20), WeightLevel2 real, WeightUOMLevel2 varchar(20),
            ItemIDLevel3 varchar(50), ItemCategoryLevel3 varchar(50), QtyReqLevel3 real,
            UOMLevel3 varchar(20), WeightLevel3 real, WeightUOMLevel3 varchar(20))
create table #SalesUOMConversion(LineID int, SalesToPrimaryUnitConversion real)
create table #Plant(OrgID int, PlantID smallint, PlantTitle varchar(50))
create table #CustomerJobData(OrderID int, CustomerName varchar(100), JobName varchar(100), ZipCode varchar(20))
declare @Results table(OracleOrderID int, CustomerName varchar(100), JobName varchar(100), JobLocation varchar(50), JobLocationState
varchar(10),
                       OrderSalesValue real, PlantCount tinyint, PlantName varchar(50), PlantSalesValue real, MaterialPlantTotal real,
                       MaterialType varchar(10), MaterialQtyReq real, MaterialProportion real, RecycledProportionPreConsumer real,
                       RecycledProportionPostConsumer real, PlantLocation varchar(50), PlantLocationState varchar(50), Supplier
varchar(30),
                       SupplierLocation varchar(50), SupplierLocationState varchar(10), DistJobPlant real, DistJobSupplier real,
                       mtlAppearanceOrder tinyint, ErrMessage varchar(255))
declare @CastingCategory table(Category varchar(50))
if(@IncludeCastings='yes')
  begin
    insert into @CastingCategory values('raw resale.ring')
    insert into @CastingCategory values('raw resale.covers')
    insert into @CastingCategory values('raw resale.frame&cover')
    insert into @CastingCategory values('raw resale.frame&grate')
    insert into @CastingCategory values('raw resale.frames')
    insert into @CastingCategory values('raw resale.grates')
    insert into @CastingCategory values('raw resale.hatches')
```

```
    insert into @CastingCategory values('raw resale.ring&cover')
    insert into @CastingCategory values('raw resale.ring&grate')
  end

-- Get sales line data, BOM data, and UOM conversions in separate steps
-- 1.  Because items may appear on a sales order credited to a plant that does not produce it (BOM does not exist)
--     In this case, an alternate plant for a (sales order, item) combination can be specified (in the OraclePlantSubstitution table)
--     To make data available for substitution, all BOMs (from all producing plants) for an item are retrieved and later joined to
--     sales lines
-- 2.  SQL Server 2000 varchars are limited to 8000 characters, so retrieve UOM conversions in separate step, then join later

-- Sales line data

select @sqltext='insert into #SalesLine select * from openquery(OracleProduction,''
select Header.Order_Number, Line.Line_ID, Line.Ship_From_Org_ID, Item.Segment1 as ItemID,
       nvl(ItemCategory.Segment1||''''.''''||ItemCategory.Segment2,''''na''''),
       case when(lower(Line.Line_Category_Code)<>''''return'''')then Line.Ordered_Quantity else -Line.Ordered_Quantity end,
       Line.Order_Quantity_UOM,
       case when(lower(Line.Line_Category_Code)<>''''return'''')then Line.Shipped_Quantity else -Line.Shipped_Quantity end,
       Line.Unit_Selling_Price, Item.Primary_UOM_Code, Item.Unit_Weight, Item.Weight_UOM_Code
from   ont.oe_Order_Headers_All Header join ont.oe_Order_Lines_All Line on Header.Header_ID=Line.Header_ID
       join inv.mtl_System_Items_B Item on Line.Ship_From_Org_ID=Item.Organization_ID and
Line.Inventory_Item_ID=Item.Inventory_Item_ID
       left join inv.mtl_Category_Sets_tl ItemCategorySet on ItemCategorySet.Language=userenv(''''lang'''')
       and lower(ItemCategorySet.Category_Set_Name)=''''inventory''''
       left join inv.mtl_Item_Categories ItemCategoryHeader on ItemCategorySet.Category_Set_ID=ItemCategoryHeader.Category_Set_ID
       and Line.Ship_From_Org_ID=ItemCategoryHeader.Organization_ID and Line.Inventory_Item_ID=ItemCategoryHeader.Inventory_Item_ID
       left join inv.mtl_Categories_B ItemCategory on ItemCategoryHeader.Category_ID=ItemCategory.Category_ID
       and lower(ItemCategory.Enabled_Flag)=''''y'''' and nvl(ItemCategory.Disable_Date,sysdate)>=sysdate
where  Header.Order_Number in(' + @OracleOrderID + ') and lower(Line.Flow_Status_Code)<>''''cancelled''''
       and Line.Ordered_Quantity<>0'')'
exec(@sqltext)

-- BOM data

select @sqltext='insert into #BOM select * from openquery(OracleProduction,''
select Line.Line_ID, Item.Organization_ID,
       BOMComponentLevel1.Segment1 as BOMItemIDLevel1,
       nvl(ItemCategoryLevel1.Segment1||''''.''''||ItemCategoryLevel1.Segment2,''''na''''),
       BOMLineLevel1.Component_Quantity, BOMComponentLevel1.Primary_UOM_Code, BOMComponentLevel1.Unit_Weight,
       BOMComponentLevel1.Weight_UOM_Code, BOMComponentLevel2.Segment1 as BOMItemIDLevel2,
       nvl(ItemCategoryLevel1.Segment2||''''.''''||ItemCategoryLevel2.Segment2,''''na''''),
       BOMLineLevel2.Component_Quantity, BOMComponentLevel2.Primary_UOM_Code, BOMComponentLevel2.Unit_Weight,
       BOMComponentLevel2.Weight_UOM_Code, BOMComponentLevel3.Segment1 as BOMItemIDLevel3,
       nvl(ItemCategoryLevel1.Segment3||''''.''''||ItemCategoryLevel3.Segment2,''''na''''),
       BOMLineLevel3.Component_Quantity, BOMComponentLevel3.Primary_UOM_Code, BOMComponentLevel3.Unit_Weight,
       BOMComponentLevel3.Weight_UOM_Code
from   -- Get requested sales lines
       ont.oe_Order_Headers_All Header join ont.oe_Order_Lines_All Line on Header.Header_ID=Line.Header_ID
       -- Get Item records for all plants (so that substitutions can be made later)
       join inv.mtl_System_Items_B Item on Line.Inventory_Item_ID=Item.Inventory_Item_ID
```

```
-- One time item sub org 1514 to 1515
-- and Line.Inventory_Item_ID<>2355701 or Item.Inventory_Item_ID=2022221
-- Get first level BOM components
join bom.BOM_Structures_b BOMHeaderLevel1 on Item.Organization_ID=BOMHeaderLevel1.Organization_ID
and Item.Inventory_Item_ID=BOMHeaderLevel1.Assembly_Item_ID
-- Default BOMs only (no alternates)
and BOMHeaderLevel1.Alternate_BOM_Designator is null
join bom.BOM_Components_b BOMLineLevel1 on BOMHeaderLevel1.Bill_Sequence_ID=BOMLineLevel1.Bill_Sequence_ID
and BOMLineLevel1.Disable_Date is null and BOMLineLevel1.Component_Quantity<>0
join inv.mtl_System_Items_b BOMComponentLevel1 on BOMHeaderLevel1.Organization_ID=BOMComponentLevel1.Organization_ID
and BOMLineLevel1.Component_Item_ID=BOMComponentLevel1.Inventory_Item_ID
-- Get second level BOM components (subcomponents of first level components)
left join inv.mtl_System_Items_B ItemLevel2 on BOMComponentLevel1.Organization_ID=ItemLevel2.Organization_ID
and BOMComponentLevel1.Inventory_Item_ID=ItemLevel2.Inventory_Item_ID
left join bom.BOM_Structures_b BOMHeaderLevel2 on ItemLevel2.Organization_ID=BOMHeaderLevel2.Organization_ID
and ItemLevel2.Inventory_Item_ID=BOMHeaderLevel2.Assembly_Item_ID
-- Default BOMs only (no alternates)
and BOMHeaderLevel2.Alternate_BOM_Designator is null
left join bom.BOM_Components_b BOMLineLevel2 on BOMHeaderLevel2.Bill_Sequence_ID=BOMLineLevel2.Bill_Sequence_ID
and BOMLineLevel2.Disable_Date is null and BOMLineLevel2.Component_Quantity<>0
left join inv.mtl_System_Items_b BOMComponentLevel2 on BOMHeaderLevel2.Organization_ID=BOMComponentLevel2.Organization_ID
and BOMLineLevel2.Component_Item_ID=BOMComponentLevel2.Inventory_Item_ID
-- Get third level BOM components (subcomponents of second level components)
left join inv.mtl_System_Items_B ItemLevel3 on BOMComponentLevel2.Organization_ID=ItemLevel3.Organization_ID
and BOMComponentLevel2.Inventory_Item_ID=ItemLevel3.Inventory_Item_ID
left join bom.BOM_Structures_b BOMHeaderLevel3 on ItemLevel3.Organization_ID=BOMHeaderLevel3.Organization_ID
and ItemLevel3.Inventory_Item_ID=BOMHeaderLevel3.Assembly_Item_ID
-- Default BOMs only (no alternates)
and BOMHeaderLevel3.Alternate_BOM_Designator is null
left join bom.BOM_Components_b BOMLineLevel3 on BOMHeaderLevel3.Bill_Sequence_ID=BOMLineLevel3.Bill_Sequence_ID
and BOMLineLevel3.Disable_Date is null and BOMLineLevel3.Component_Quantity<>0
left join inv.mtl_System_Items_b BOMComponentLevel3 on BOMHeaderLevel3.Organization_ID=BOMComponentLevel3.Organization_ID
and BOMLineLevel3.Component_Item_ID=BOMComponentLevel3.Inventory_Item_ID
-- Get item category codes
-- BOM level 1
left join inv.mtl_Category_Sets_tl ItemCategorySet1 on ItemCategorySet1.Language=userenv('''lang''')
and lower(ItemCategorySet1.Category_Set_Name)=''''inventory''''
left join inv.mtl_Item_Categories ItemCategoryHeader1 on ItemCategorySet1.Category_Set_ID=ItemCategoryHeader1.Category_Set_ID
and BOMHeaderLevel1.Organization_ID=ItemCategoryHeader1.Organization_ID
and BOMHeaderLevel1.Assembly_Item_ID=ItemCategoryHeader1.Inventory_Item_ID
left join inv.mtl_Categories_B ItemCategoryLevel1 on ItemCategoryHeader1.Category_ID=ItemCategoryLevel1.Category_ID
and lower(ItemCategoryLevel1.Enabled_Flag)=''''y'''' and nvl(ItemCategoryLevel1.Disable_Date,sysdate)>=sysdate
-- BOM level 2
left join inv.mtl_Category_Sets_tl ItemCategorySet2 on ItemCategorySet2.Language=userenv('''lang''')
and lower(ItemCategorySet2.Category_Set_Name)=''''inventory''''
left join inv.mtl_Item_Categories ItemCategoryHeader2 on ItemCategorySet2.Category_Set_ID=ItemCategoryHeader2.Category_Set_ID
and BOMHeaderLevel2.Organization_ID=ItemCategoryHeader2.Organization_ID
and BOMHeaderLevel2.Assembly_Item_ID=ItemCategoryHeader2.Inventory_Item_ID
left join inv.mtl_Categories_B ItemCategoryLevel2 on ItemCategoryHeader2.Category_ID=ItemCategoryLevel2.Category_ID
and lower(ItemCategoryLevel2.Enabled_Flag)=''''y'''' and nvl(ItemCategoryLevel2.Disable_Date,sysdate)>=sysdate
-- BOM level 3
```

```
       left join inv.mtl_Category_Sets_tl ItemCategorySet3 on ItemCategorySet3.Language=userenv(''''lang'''')
       and lower(ItemCategorySet3.Category_Set_Name)=''''inventory''''
       left join inv.mtl_Item_Categories ItemCategoryHeader3 on ItemCategorySet3.Category_Set_ID=ItemCategoryHeader3.Category_Set_ID
       and BOMHeaderLevel3.Organization_ID=ItemCategoryHeader3.Organization_ID
       and BOMHeaderLevel3.Assembly_Item_ID=ItemCategoryHeader3.Inventory_Item_ID
       left join inv.mtl_Categories_B ItemCategoryLevel3 on ItemCategoryHeader3.Category_ID=ItemCategoryLevel3.Category_ID
       and lower(ItemCategoryLevel3.Enabled_Flag)=''''y'''' and nvl(ItemCategoryLevel3.Disable_Date,sysdate)>=sysdate
where  Header.Order_Number in(' + @OracleOrderID + ') and lower(Line.Flow_Status_Code)<>''''cancelled''''
       and Line.Ordered_Quantity<>0'')'
exec(@sqltext)

-- UOM conversion rates from sales units to item primary units

-- Notes on Oracle sales units to item primary units conversion
-- Experimentation has demonstrated that sales line units must be one of:  item primary units, an intra-class uom code
-- (mtl_uom_conversions.uom_code where class = class of item primary units),
-- or in the same class as an inter-class uom code [(mtl_uom_class_conversions.from_uom_code, .to_uom_code) where item ID =
-- sales item ID]
-- With one exception (RLNG0000003), one and only one of (mtl_uom_class_conversions.from_uom, .to_uom) are of the same class as
-- that of the
-- primary units of an item
-- Unit conversion method:
-- if(sales units = item primary units)then
--    conversion = 1
--  if(exists an intra-class conversion)then
--    note: all uoms in class are specified in terms of the class uom (tons for weight, each for quantity, etc)
--    conversion = sales_units_class_conversion_rate/primary_units_class_conversion_rate
--  if(exists an inter-class conversion)then
--    if(sales_uom class = from_uom class)then
--      note: to_uom must be in same class as primary units since one and only one of (from_uom, to_uom) share class with pri_uom
--            and, from observation, from_uom in same class as sales_uom
--            convert using inter-class rate, then intra-class rates for sales_uom and to_uom
--      conversion = sales qty / inter-class-conversion-rate * sales-uom-intra-rate / to-uom-intra-rate / pri-uom-intra-class-
conversion-rate
--    else
--      note: from_uom in same class as pri_uom, to_uom in same class as sales_uom
--      conversion = sales qty * inter-class-conversion-rate * sales-uom-intra-rate / from-uom-intra-rate / pri-uom-intra-class-
conversion-rate

select @sqltext='insert into #SalesUOMConversion select * from openquery(OracleProduction,''
select Line.Line_ID,
       case when(Line.Order_Quantity_UOM=Item.Primary_UOM_Code)then
              -- Ordered units same as primary units - no conversion
              1
            when(UOMConversionIntra.Inventory_Item_ID is not null and UOMConversionPri.Conversion_Rate>0)then
              -- Exists an intra-class conversion - resulting conversion is sales_rate / primary_rate
              -- It is assumed that both sales_uom and pri_uom are actually in the same class
              -- Note that all conversion rates within a class are with respect to the base units for the class (each for quantity,
etc)
              nvl(UOMConversionIntra.Conversion_Rate,-1)/UOMConversionPri.Conversion_Rate
            when(UOMConversionInter.Inventory_Item_ID is not null)then
```

```
            -- Exists an inter-class conversion
            -- On clause requires one of:  sales_uom=from_uom and item_pri_uom=to_uom or sales_uom=to_uom and item_pri_uom=from_uom
            case when(Line.Order_Quantity_UOM=UOMConversionInter.FromClassUOM and UOMConversionInter.Conversion_Rate>0
                      and UOMConversionSales.Conversion_Rate>0 and UOMConversionInter.ToRate>0
                      and UOMConversionPri.Conversion_Rate>0)then
                   -- Sales_uom=from_uom, to_uom=pri_uom, so resulting conversion=1/intra_rate*sales_rate/to_rate/pri_rate
                   1/UOMConversionInter.Conversion_Rate*UOMConversionSales.Conversion_Rate/
                   UOMConversionInter.ToRate/UOMConversionPri.Conversion_Rate
                 when(Line.Order_Quantity_UOM=UOMConversionInter.ToClassUOM and UOMConversionInter.Conversion_Rate>0
                      and UOMConversionSales.Conversion_Rate>0 and UOMConversionInter.FromRate>0
                      and UOMConversionPri.Conversion_Rate>0)then
                   -- Sales_uom=to_uom, from_uom=pri_uom, so resulting conversion=intra_rate*sales_rate/from_rate/pri_rate
                   UOMConversionInter.Conversion_Rate*UOMConversionSales.Conversion_Rate/
                   UOMConversionInter.FromRate/UOMConversionPri.Conversion_Rate
                 else
                   -1
            end
          else
            -1
       end as SalesToPrimaryUnitConversion
from   -- Get sales lines
       ont.oe_Order_Headers_All Header join ont.oe_Order_Lines_All Line on Header.Header_ID=Line.Header_ID
       join inv.mtl_System_Items_B Item on Line.Ship_From_Org_ID=Item.Organization_ID
       and Line.Inventory_Item_ID=Item.Inventory_Item_ID
       left join -- Check for intra-class UOM conversion (tons->pounds, ounces->gallons, etc)
                 -- It is assumed that primary keys prevent multiple records by item, units
                 -- it is also assume that sales units and item primary are actually in the same class
                 inv.mtl_UOM_Conversions UOMConversionIntra on Line.Order_Quantity_UOM<>Item.Primary_UOM_Code
                 and Item.Inventory_Item_ID=UOMConversionIntra.Inventory_Item_ID
       left join -- Get in-class item primary unit conversion rate (item ID = 0)
                 -- Note selection of base conversion for class (item ID = 0) since pri_uom is what we are converting to
                 -- UOMs by item ID are unique, so at most one record here
                 inv.mtl_UOM_Conversions UOMConversionPri on Line.Order_Quantity_UOM<>Item.Primary_UOM_Code
                 and Item.Primary_UOM_Code=UOMConversionPri.UOM_Code and UOMConversionPri.Inventory_Item_ID=0
       left join ( -- Check for inter-class UOM conversion (tons->each, ounces->feet, etc)
                   -- One of (from_uom, to_uom) in same class as sales_uom, the other in same class as item pri_uom
                   -- Get intra-class unit conversions for from_uom and to_uom
                   -- It is assumed that primary keys prevent multiple records by item, from units, and to units
                   select InterClass.Inventory_Item_ID, InterClass.Conversion_Rate,
                          IntraFrom.UOM_Code as FromClassUOM, IntraFrom.Conversion_Rate as FromRate,
                          IntraTo.UOM_Code as ToClassUOM, IntraTo.Conversion_Rate as ToRate
                   from   inv.mtl_UOM_Class_Conversions InterClass
                          -- Collect all possible conversion rates for from_uom (same class as from_uom) - this is a unique list
                          join inv.mtl_UOM_Conversions IntraFrom on InterClass.From_UOM_Class=IntraFrom.UOM_Class
                          and IntraFrom.Inventory_Item_ID=0
                          -- Collect all possible conversion rates for to_uom (same class) - this is a unique list
                          join inv.mtl_UOM_Conversions IntraTo on InterClass.To_UOM_Class=IntraTo.UOM_Class
                          and IntraTo.Inventory_Item_ID=0
                   where  (InterClass.Disable_Date is null or InterClass.Disable_Date>SysDate)
                          and (IntraFrom.Disable_Date is null or IntraFrom.Disable_Date>SysDate)
                          and (IntraTo.Disable_Date is null or IntraTo.Disable_Date>SysDate)
```

```
                ) UOMConversionInter on Line.Order_Quantity_UOM<>Item.Primary_UOM_Code
                  and UOMConversionIntra.Inventory_Item_ID is null
                  and Item.Inventory_Item_ID=UOMConversionInter.Inventory_Item_ID
                  -- One and only one of (from_uom, to_uom) in same class as sales_uom, other in same class as item pri_uom
                  and (Line.Order_Quantity_UOM=UOMConversionInter.FromClassUOM
                       and Item.Primary_UOM_Code=UOMConversionInter.ToClassUOM
                       or Line.Order_Quantity_UOM=UOMConversionInter.ToClassUOM
                          and Item.Primary_UOM_Code=UOMConversionInter.FromClassUOM)
       left join -- Get in-class conversion rate for sales units
                 -- For inter-class conversions, sales units are in same class as one of (inter.from_uom, inter.to_uom)
                 -- UOMs unique by item ID, so at most one record returned here
                 inv.mtl_UOM_Conversions UOMConversionSales on UOMConversionInter.Inventory_Item_ID is not null
                 and Line.Order_Quantity_UOM=UOMConversionSales.UOM_Code and UOMConversionSales.Inventory_Item_ID=0
where  Header.Order_Number in(' + @OracleOrderID + ') and lower(Line.Flow_Status_Code)<>''''cancelled''''''')'
exec(@sqltext)

-- Plant IDs and titles
select @sqltext='insert into #Plant select * from openquery(OracleProduction,''
select HROrg.Organization_ID, InvOrg.Organization_Code, HROrg.Name
from   hr.hr_All_Organization_Units HROrg join inv.mtl_Parameters InvOrg on HROrg.Organization_ID=InvOrg.Organization_ID'')'
exec(@sqltext)

-- Require valid sales item UOM conversions
insert into @Results(ErrMessage)
select 'Error - Invalid UOM conversion: ' + PlantTitle + '; ' + ItemID + '; ' + UOM + ' to ' + ItemPrimaryUOM
from   #SalesLine join #SalesUOMConversion on #SalesLine.LineID=#SalesUOMConversion.LineID
       left join #Plant on #SalesLine.OrgID=#Plant.OrgID
where  isnull(SalesToPrimaryUnitConversion,-1)<=0

if(not exists(select * from @Results where ErrMessage is not null))
  begin

    -- Convert sales units where applicable
    update #SalesLine set QtyReq=QtyReq*#SalesUOMConversion.SalesToPrimaryUnitConversion,
           QtyShipped=QtyShipped*#SalesUOMConversion.SalesToPrimaryUnitConversion,
           -- Note that div by 0 avoided by unit conversion requirement > 0 in where clause
           Price=Price/#SalesUOMConversion.SalesToPrimaryUnitConversion
    from   #SalesLine join #SalesUOMConversion on #SalesLine.LineID=#SalesUOMConversion.LineID
    where  #SalesUOMConversion.SalesToPrimaryUnitConversion>0

    -- Substitute plants where applicable
    if(@SubstitutePlantBOM='yes')
      update #SalesLine set OrgID=isnull(SubOrgPri.OrgID, SubOrgDef.OrgID)
      from   #SalesLine join #Plant on #SalesLine.OrgID=#Plant.OrgID
             -- Get substitute plant for requested order, sales order plant, and sales item
             left join OracleItemPlantSubstitute SubPri on #Plant.PlantID=SubPri.PlantID
             and #SalesLine.OrderID=SubPri.OrderID and #SalesLine.ItemID=SubPri.ItemID
             -- Get Oracle ID for substitute plant
             left join #Plant SubOrgPri on SubPri.SubstitutePlantID=SubOrgPri.PlantID
             -- Get default substitution for plant and item (all orders)
             left join OracleItemPlantSubstitute SubDef on #Plant.PlantID=SubDef.PlantID
```

```
        and SubDef.OrderID=0 and #SalesLine.ItemID=SubDef.ItemID
        left join #Plant SubOrgDef on SubDef.SubstitutePlantID=SubOrgDef.PlantID
  where  #SalesLine.OrgID<>isnull(SubOrgPri.OrgID, SubOrgDef.OrgID)

-- **********************************
-- component mods
-- update #bom set WeightUOMLevel1='lbs' where ItemIDLevel1='RBAR0400M01'
-- select * from #SalesLine where ItemID='SR144R14P036N000'
-- select * from #BOM where LineID in(9689633, 9729288) order by LineID
-- **********************************

-- Require valid material UOMs (after plant substitution so that appropriate BOM is verified)
-- Component units can be "tons", 'lbs", "kg" (2.2406 lbs), "mt" (metric ton, 2204.623 lbs), "cbm" (cubic meter), or "each"
-- If "each" or "cbm" then use weight (weight UOM must be "lbs", "tons", "kg", or "mt" and weight>0)
-- Castings must have primary UOM of "each", have weight UOM of "lbs", "tons", "kg", or "mt", and have weight>0
-- Components on sales items
insert into @Results(ErrMessage)
select 'Error - Unknown UOM or Invalid Weight: ' + PlantTitle + '; ' + ItemID  + '; ' + lower(UOM) + '; Weight=' +
       convert(varchar(10),isnull(ItemWeight,0)) + ' ' + isnull(lower(ItemWeightUOM),'Unknown Wt UOM')
from   #SalesLine left join #Plant on #SalesLine.OrgID=#Plant.OrgID
where  left(ItemID,4) in('rcem','rfly','rsnd','rrck','rmce','rmfl','rmsn','rmrc','rbar','rmsh','rwir')
       and (UOM not in('ton','lbs','kg','mt','cbm','ea')
            or UOM in('cbm','ea') and (isnull(ItemWeight,0)<=0 or ItemWeightUOM not in('lbs','ton','kg','mt')))
       or ItemCategory in(select Category from @CastingCategory)
       and (UOM<>'ea' or isnull(ItemWeight,0)<=0 or ItemWeightUOM not in('lbs','ton','kg','mt'))
-- First level BOM components
insert into @Results(ErrMessage)
select 'Error - Unknown UOM or Invalid Weight: ' + PlantTitle + '; ' + #SalesLine.ItemID + '(' + #BOM.ItemIDLevel1 + '); ' +
       lower(#BOM.UOMLevel1) + '; Weight=' +
       convert(varchar(10),isnull(#BOM.WeightLevel1,0)) + ' ' + isnull(lower(#BOM.WeightUOMLevel1),'Unknown Wt UOM')
from   #SalesLine join #BOM on #SalesLine.LineID=#BOM.LineID and #SalesLine.OrgID=#BOM.OrgID
       left join #Plant on #SalesLine.OrgID=#Plant.OrgID
where  left(#BOM.ItemIDLevel1,4) in('rcem','rfly','rsnd','rrck','rmce','rmfl','rmsn','rmrc','rbar','rmsh','rwir')
       and (#BOM.UOMLevel1 not in('ton','lbs','kg','mt','cbm','ea')
            or #BOM.UOMLevel1 in('cbm','ea') and (isnull(#BOM.WeightLevel1,0)<=0
            or #BOM.WeightUOMLevel1 not in('lbs','ton','kg','mt')))
       or #BOM.ItemCategoryLevel1 in(select Category from @CastingCategory)
       and (#BOM.UOMLevel1<>'ea' or isnull(#BOM.WeightLevel1,0)<=0 or #BOM.WeightUOMLevel1 not in('lbs','ton','kg','mt'))
-- Second level BOM components
insert into @Results(ErrMessage)
select 'Error - Unknown UOM or Invalid Weight: ' + PlantTitle + '; ' + #SalesLine.ItemID + '(' + #BOM.ItemIDLevel1 + ')( ' +
       #BOM.ItemIDLevel2 + '); ' + lower(#BOM.UOMLevel2) + '; Weight=' +
       convert(varchar(10),isnull(#BOM.WeightLevel2,0)) + ' ' + isnull(lower(#BOM.WeightUOMLevel2),'Unknown Wt UOM')
from   #SalesLine join #BOM on #SalesLine.LineID=#BOM.LineID and #SalesLine.OrgID=#BOM.OrgID
       left join #Plant on #SalesLine.OrgID=#Plant.OrgID
where  left(#BOM.ItemIDLevel2,4) in('rcem','rfly','rsnd','rrck','rmce','rmfl','rmsn','rmrc','rbar','rmsh','rwir')
       and (#BOM.UOMLevel2 not in('ton','lbs','kg','mt','cbm','ea')
            or #BOM.UOMLevel2 in('cbm','ea') and (isnull(#BOM.WeightLevel2,0)<=0
            or #BOM.WeightUOMLevel2 not in('lbs','ton','kg','mt')))
       or #BOM.ItemCategoryLevel2 in(select Category from @CastingCategory)
       and (#BOM.UOMLevel2<>'ea' or isnull(#BOM.WeightLevel2,0)<=0 or #BOM.WeightUOMLevel2 not in('lbs','ton','kg','mt'))
```

```
-- Third level BOM components
insert into @Results(ErrMessage)
select 'Error - Unknown UOM or Invalid Weight: ' + PlantTitle + '; ' + #SalesLine.ItemID + '(' + #BOM.ItemIDLevel1 + ')( ' +
       #BOM.ItemIDLevel2 + ')(' + #BOM.ItemIDLevel3 + '); ' + lower(#BOM.UOMLevel3) + '; Weight=' +
       convert(varchar(10),isnull(#BOM.WeightLevel3,0)) + ' ' + isnull(lower(#BOM.WeightUOMLevel3),'Unknown Wt UOM')
from   #SalesLine join #BOM on #SalesLine.LineID=#BOM.LineID and #SalesLine.OrgID=#BOM.OrgID
       left join #Plant on #SalesLine.OrgID=#Plant.OrgID
where  left(#BOM.ItemIDLevel3,4) in('rcem','rfly','rsnd','rrck','rmce','rmfl','rmsn','rmrc','rbar','rmsh','rwir')
       and (#BOM.UOMLevel3 not in('ton','lbs','kg','mt','cbm','ea')
            or #BOM.UOMLevel3 in('cbm','ea') and (isnull(#BOM.WeightLevel3,0)<=0
            or #BOM.WeightUOMLevel3 not in('lbs','ton','kg','mt')))
       or #BOM.ItemCategoryLevel3 in(select Category from @CastingCategory)
       and (#BOM.UOMLevel3<>'ea' or isnull(#BOM.WeightLevel3,0)<=0 or #BOM.WeightUOMLevel3 not in('lbs','ton','kg','mt'))

if(not exists(select * from @Results where ErrMessage is not null))
  begin

    -- Check for finished items with no material (non-raw material, no BOM records)
    -- Use @IgnoreMissingBOM to over-ride
    if(@IgnoreMissingBOM='no')
      insert   into @Results(ErrMessage)
      select   'Error - Oracle Item with No BOM: ' + PlantTitle + '; ' + #SalesLine.ItemID
      from     #SalesLine left join #BOM on #SalesLine.LineID=#BOM.LineID and #SalesLine.OrgID=#BOM.OrgID
               and (left(#BOM.ItemIDLevel1,4) in('rcem','rfly','rbar','rmsh','rwir','rsnd','rrck','rmce','rmfl','rmsn','rmrc')
                    or left(#BOM.ItemIDLevel2,4) in('rcem','rfly','rbar','rmsh','rwir','rsnd','rrck','rmce','rmfl','rmsn','rmrc')
                    or left(#BOM.ItemIDLevel3,4) in('rcem','rfly','rbar','rmsh','rwir','rsnd','rrck','rmce','rmfl','rmsn','rmrc'))
               left join #Plant on #SalesLine.OrgID=#Plant.OrgID
      where    #SalesLine.ItemID not like 'r%' and #BOM.LineID is null
      group by PlantTitle, #SalesLine.ItemID

    if(not exists(select * from @Results where ErrMessage is not null))
      begin

        -- Accumulate material quantities (in lbs) by plant and material type
        -- Use staging table for further tests and accumulation
        declare  @mtl table(OrgID smallint, MaterialType varchar(10), QtyReq real)
        insert   into @mtl
        select   OrgID, MaterialType, sum(QtyReq) as QtyReq
        from     ( select #SalesLine.OrgID,
                          -- Identify material type sales line takes precedece, then BOM level 1, then BOM level 2
                          -- Use same sequence for quantity case statement so that accumulation identified by correct
                          -- material type
                          -- Do not combine case clauses (#SalesLine like 'rcem%' or #BOM like 'rcem%', for instance)
                          -- so that sales line checked first then BOM level 1, then BOM level 2
                          -- in same sequence as in quantity case statement
                          case when(left(#SalesLine.ItemID,4) in('rcem','rmce'))then
                                 'Cement'
                               when(left(#SalesLine.ItemID,4) in('rfly','rmfl'))then
                                 'Flyash'
                               when(#SalesLine.ItemID like 'rbar%')then
                                 'Rebar'
```

```
when(left(#SalesLine.ItemID,4) in('rmsh','rwir'))then
  'Mesh'
when(left(#SalesLine.ItemID,4) in('rsnd','rmsn'))then
  'Sand'
when(left(#SalesLine.ItemID,4) in('rrck','rmrc'))then
  'Aggregate'
when(CastingCategorySalesLine.Category is not null)then
  'Castings'
when(left(#BOM.ItemIDLevel1,4) in('rcem','rmce'))then
  'Cement'
when(left(#BOM.ItemIDLevel1,4) in('rfly','rmfl'))then
  'Flyash'
when(#BOM.ItemIDLevel1 like 'rbar%')then
  'Rebar'
when(left(#BOM.ItemIDLevel1,4) in('rmsh','rwir'))then
  'Mesh'
when(left(#BOM.ItemIDLevel1,4) in('rsnd','rmsn'))then
  'Sand'
when(left(#BOM.ItemIDLevel1,4) in('rrck','rmrc'))then
  'Aggregate'
when(CastingCategoryBOMLevel1.Category is not null)then
  'Castings'
when(left(#BOM.ItemIDLevel2,4) in('rcem','rmce'))then
  'Cement'
when(left(#BOM.ItemIDLevel2,4) in('rfly','rmfl'))then
  'Flyash'
when(#BOM.ItemIDLevel2 like 'rbar%')then
  'Rebar'
when(left(#BOM.ItemIDLevel2,4) in('rmsh','rwir'))then
  'Mesh'
when(left(#BOM.ItemIDLevel2,4) in('rsnd','rmsn'))then
  'Sand'
when(left(#BOM.ItemIDLevel2,4) in('rrck','rmrc'))then
  'Aggregate'
when(CastingCategoryBOMLevel2.Category is not null)then
  'Castings'
when(left(#BOM.ItemIDLevel3,4) in('rcem','rmce'))then
  'Cement'
when(left(#BOM.ItemIDLevel3,4) in('rfly','rmfl'))then
  'Flyash'
when(#BOM.ItemIDLevel3 like 'rbar%')then
  'Rebar'
when(left(#BOM.ItemIDLevel3,4) in('rmsh','rwir'))then
  'Mesh'
when(left(#BOM.ItemIDLevel3,4) in('rsnd','rmsn'))then
  'Sand'
when(left(#BOM.ItemIDLevel3,4) in('rrck','rmrc'))then
  'Aggregate'
when(CastingCategoryBOMLevel3.Category is not null)then
  'Castings'
else
```

```
       'Other'
end as MaterialType,
case when(left(#SalesLine.ItemID,4) in('rcem','rfly','rbar','rmsh','rwir','rsnd',
                                       'rrck','rmce','rmfl','rmsn','rmrc'))then
       #SalesLine.QtyReq *
       case when(#SalesLine.ItemPrimaryUOM='lbs')then 1
            when(#SalesLine.ItemPrimaryUOM='ton')then 2000
            when(#SalesLine.ItemPrimaryUOM='kg')then 2.204623
            when(#SalesLine.ItemPrimaryUOM='mt')then 2204.623
            when(#SalesLine.ItemPrimaryUOM in('cbm','ea'))then
              #SalesLine.ItemWeight *
              case when(#SalesLine.ItemWeightUOM='lbs')then 1
                   when(#SalesLine.ItemWeightUOM='ton')then 2000
                   when(#SalesLine.ItemWeightUOM='kg')then 2.204623
                   when(#SalesLine.ItemWeightUOM='mt')then 2204.623
                   else 0
              end
            else 0
       end
     when(left(#BOM.ItemIDLevel1,4) in('rcem','rfly','rbar','rmsh','rwir','rsnd',
                                       'rrck','rmce','rmfl','rmsn','rmrc'))then
       #SalesLine.QtyReq*#BOM.QtyReqLevel1 *
       case when(#BOM.UOMLevel1='lbs')then 1
            when(#BOM.UOMLevel1='ton')then 2000
            when(#BOM.UOMLevel1='kg')then 2.204623
            when(#BOM.UOMLevel1='mt')then 2204.623
            when(#BOM.UOMLevel1 in('cbm','ea'))then
              #BOM.WeightLevel1 *
              case when(#BOM.WeightUOMLevel1='lbs')then 1
                   when(#BOM.WeightUOMLevel1='ton')then 2000
                   when(#BOM.WeightUOMLevel1='kg')then 2.204623
                   when(#BOM.WeightUOMLevel1='mt')then 2204.623
                   else 0
              end
            else 0
       end
     when(left(#BOM.ItemIDLevel2,4) in('rcem','rfly','rbar','rmsh','rwir','rsnd',
                                       'rrck','rmce','rmfl','rmsn','rmrc'))then
       #SalesLine.QtyReq*#BOM.QtyReqLevel1*#BOM.QtyReqLevel2 *
       case when(#BOM.UOMLevel2='lbs')then 1
            when(#BOM.UOMLevel2='ton')then 2000
            when(#BOM.UOMLevel2='kg')then 2.204623
            when(#BOM.UOMLevel2='mt')then 2204.623
            when(#BOM.UOMLevel2 in('cbm','ea'))then
              #BOM.WeightLevel2 *
              case when(#BOM.WeightUOMLevel2='lbs')then 1
                   when(#BOM.WeightUOMLevel2='ton')then 2000
                   when(#BOM.WeightUOMLevel2='kg')then 2.204623
                   when(#BOM.WeightUOMLevel2='mt')then 2204.623
                   else 0
              end
```

```
      else 0
  end
when(left(#BOM.ItemIDLevel3,4) in('rcem','rfly','rbar','rmsh','rwir','rsnd',
                                  'rrck','rmce','rmfl','rmsn','rmrc'))then
  #SalesLine.QtyReq*#BOM.QtyReqLevel1*#BOM.QtyReqLevel2*#BOM.QtyReqLevel3 *
  case when(#BOM.UOMLevel3='lbs')then 1
       when(#BOM.UOMLevel3='ton')then 2000
       when(#BOM.UOMLevel3='kg')then 2.204623
       when(#BOM.UOMLevel3='mt')then 2204.623
       when(#BOM.UOMLevel3 in('cbm','ea'))then
         #BOM.WeightLevel3 *
         case when(#BOM.WeightUOMLevel3='lbs')then 1
              when(#BOM.WeightUOMLevel3='ton')then 2000
              when(#BOM.WeightUOMLevel3='kg')then 2.204623
              when(#BOM.WeightUOMLevel3='mt')then 2204.623
              else 0
         end
       else 0
  end
when(CastingCategorySalesLine.Category is not null)then
  #SalesLine.QtyReq*#SalesLine.ItemWeight *
  case when(#SalesLine.ItemWeightUOM='lbs')then 1
       when(#SalesLine.ItemWeightUOM='ton')then 2000
       when(#SalesLine.ItemWeightUOM='kg')then 2.204623
       when(#SalesLine.ItemWeightUOM='mt')then 2204.623
       else 0
  end
when(CastingCategoryBOMLevel1.Category is not null)then
  #SalesLine.QtyReq*#BOM.QtyReqLevel1*#BOM.WeightLevel1 *
  case when(#BOM.WeightUOMLevel1='lbs')then 1
       when(#BOM.WeightUOMLevel1='ton')then 2000
       when(#BOM.WeightUOMLevel1='kg')then 2.204623
       when(#BOM.WeightUOMLevel1='mt')then 2204.623
       else 0
  end
when(CastingCategoryBOMLevel2.Category is not null)then
  #SalesLine.QtyReq*#BOM.QtyReqLevel2*#BOM.WeightLevel2 *
  case when(#BOM.WeightUOMLevel2='lbs')then 1
       when(#BOM.WeightUOMLevel2='ton')then 2000
       when(#BOM.WeightUOMLevel2='kg')then 2.204623
       when(#BOM.WeightUOMLevel2='mt')then 2204.623
       else 0
  end
when(CastingCategoryBOMLevel3.Category is not null)then
  #SalesLine.QtyReq*#BOM.QtyReqLevel2*#BOM.WeightLevel2*#BOM.WeightLevel3 *
  case when(#BOM.WeightUOMLevel3='lbs')then 1
       when(#BOM.WeightUOMLevel3='ton')then 2000
       when(#BOM.WeightUOMLevel3='kg')then 2.204623
       when(#BOM.WeightUOMLevel3='mt')then 2204.623
       else 0
  end
```

```
                     else
                       0
                end as QtyReq
           from   #SalesLine left join #BOM on #SalesLine.LineID=#BOM.LineID and #SalesLine.OrgID=#BOM.OrgID
                  left join @CastingCategory CastingCategorySalesLine
                  on #SalesLine.ItemCategory=CastingCategorySalesLine.Category
                  left join @CastingCategory CastingCategoryBOMLevel1
                  on #BOM.ItemCategoryLevel1=CastingCategoryBOMLevel1.Category
                  left join @CastingCategory CastingCategoryBOMLevel2
                  on #BOM.ItemCategoryLevel2=CastingCategoryBOMLevel2.Category
                  left join @CastingCategory CastingCategoryBOMLevel3
                  on #BOM.ItemCategoryLevel3=CastingCategoryBOMLevel2.Category
           where  left(#SalesLine.ItemID,4) in('rcem','rfly','rbar','rmsh','rwir','rsnd',
                                              'rrck','rmce','rmfl','rmsn','rmrc')
                  or isnull(left(#BOM.ItemIDLevel1,4),'') in('rcem','rfly','rbar','rmsh','rwir','rsnd',
                                              'rrck','rmce','rmfl','rmsn','rmrc')
                  or isnull(left(#BOM.ItemIDLevel2,4),'') in('rcem','rfly','rbar','rmsh','rwir','rsnd',
                                             'rrck','rmce','rmfl','rmsn','rmrc')
                  or isnull(left(#BOM.ItemIDLevel3,4),'') in('rcem','rfly','rbar','rmsh','rwir','rsnd',
                                              'rrck','rmce','rmfl','rmsn','rmrc')
                  or CastingCategorySalesLine.Category is not null
                  or CastingCategoryBOMLevel1.Category is not null
                  or CastingCategoryBOMLevel2.Category is not null
                  or CastingCategoryBOMLevel3.Category is not null
         ) a
group by OrgID, MaterialType

-- Verify existence of Oracle plant record for all plants with recorded material on requested order
insert into @Results(ErrMessage)
select distinct case when(OraclePlant.PlantID is null)then
                       'Error - Missing Oracle Plant Record: '
                     when(OracleSupplier.PlantID is null)then
                       'Error - Missing Oracle Supplier Record: '
                     else
                       ''
                end
       + #Plant.PlantTitle + '; Material Type ' +
       case when(OraclePlant.PlantID is null)then
              'All'
            when(OracleSupplier.PlantID is null)then
              mtl.MaterialType
            else
              ''
       end
from   @mtl mtl left join #Plant on mtl.OrgID=#Plant.OrgID
       left join OraclePlant on #Plant.PlantID=OraclePlant.PlantID
       left join OracleSupplier on #Plant.PlantID=OracleSupplier.PlantID and mtl.MaterialType=OracleSupplier.MaterialType
where  OraclePlant.PlantID is null or OracleSupplier.PlantID is null

if(not exists(select * from @Results where ErrMessage is not null))
  begin
```

```
-- Get customer and job info
-- Note that "and rownum=1" returns at most one record
-- If multiple orders requested, job info from least ranked order ID returned
select @sqltext='insert into #CustomerJobData select * from openquery(oracleproduction, ''
select   OrderHeader.Order_Number, substr(CustomerLocation.Party_Name,1,99), substr(JobLocation.Address1,1,99),
         substr(JobLocation.Postal_Code,1,19)
from     ont.oe_Order_Headers_all OrderHeader left join ar.hz_Cust_Accounts Customer
         on OrderHeader.Sold_To_Org_ID=Customer.Cust_Account_ID
         left join hz_Parties CustomerLocation on Customer.Party_ID=CustomerLocation.Party_ID
         left join ar.hz_Cust_Site_Uses_All Site1 on OrderHeader.Ship_To_Org_ID=Site1.Site_Use_ID
         left join ar.hz_Cust_Acct_Sites_All Site2 on Site1.Cust_Acct_Site_ID=Site2.Cust_Acct_Site_ID
         left join ar.hz_Party_Sites Site3 on Site2.Party_Site_ID=Site3.Party_Site_ID
         left join ar.hz_Locations JobLocation on Site3.Location_ID=JobLocation.Location_ID
where    OrderHeader.Order_Number in(' + @OracleOrderID + ') and rownum=1
order by OrderHeader.Order_Number'')'
exec(@sqltext)

-- Accumulate summary information
insert into @Results
select #CustomerJobData.OrderID as OracleOrderID, #CustomerJobData.CustomerName, #CustomerJobData.JobName,
       GeoJob.Location as JobLocation, GeoJob.State as JobLocationState, OrderSummary.SalesValue,
       OrderSummary.PlantCount, OraclePlant.PlantName, PlantSales.Value, mtlPlantTotal.QtyReq as MaterialPlantTotal,
       mtl.MaterialType, mtl.QtyReq,
       case when(mtlPlantTotal.QtyReq>0)then mtl.QtyReq/mtlPlantTotal.QtyReq else 0 end as MaterialProportion,
       OracleSupplier.RecycledProportionPreConsumer, OracleSupplier.RecycledProportionPostConsumer,
       GeoPlant.Location as PlantLocation, GeoPlant.State as PlantLocationState,
       Supplier, GeoSupplier.Location as SupplierLocation, GeoSupplier.State as SupplierLocationState,
       -- Calculate spherical distance between job-site and plant locations and job-site and supplier locations
       -- Get (x,y,z) coordinates of requested location by rotating vector through (0,0,0) [earth's center]
       -- and (r,0,0) by longitude, latitude, r=earth radius
       -- Note that the x, y, and z planes all have origin at the center of the earth and (r,0,0) lies in the x-y
       -- plane
       -- which intersects the sphere at the equator
       -- Rotation of (r,0,0) by longitude then latitude locates the requested point on the sphere's (earth's) surface
       -- Note that the surface coordinates are calculated in a trigger on GeographicData to avoid continual
       -- recalculation
       -- At time of development, the radius used in the trigger and in this proc were identical at 3959
       -- Rotate job-site and plant location vectors
       -- Calculate length of straight line in 3 dimensions that connects the surface points of the vectors
       -- The length is the hypotenuse of a 3-dimensional triangle
       -- This line is the base of an isosceles triangle with legs equal the sphere radius and angle opposite base
       -- [at (0,0,0) the center of the earth] corresponding to the arc between the rotated vectors
       -- The length of that arc is the distance along the earth's surface between the two points where
       -- the vectors and the surface intersect
       -- That length, of course, is a*r, where a is the angle between the vectors and r is the earth's radius
       -- The angle opposite base = 2*arcsine(b/2r), since a line through (0,0,0) that bisects the base, one half of
       -- the base,
       -- and one vector through (0,0,0) and one location point (connected to the half-base) form a right triangle
       -- Here we go!
       -- Distances between plants and job-site
```

```
        3959 -- Radius
        -- times angle opposite base
        *2*asin( -- here's the base; length = square root of [(x-displacement)^2 + (y-displacement)^2
        +(z-displacement)^2] after rotation
                 sqrt( (GeoJob.x-GeoPlant.x)*(GeoJob.x-GeoPlant.x)+(GeoJob.y-GeoPlant.y)*(GeoJob.y
                       -GeoPlant.y)+(GeoJob.z-GeoPlant.z)*(GeoJob.z-GeoPlant.z) )
                 /7918 ),  -- divide by 2r
        -- And now, distances between job-site and supplier locations
        3959 -- times angle opposite base
        *2*asin( -- here's the base; length = square root of [(x-displacement)^2 + (y-displacement)^2 +(z-
                 -- displacement)^2] after rotation
                 sqrt( (GeoJob.x-GeoSupplier.x)*(GeoJob.x-GeoSupplier.x)+(GeoJob.y-GeoSupplier.y)*(GeoJob.y
                       -GeoSupplier.y)+(GeoJob.z-GeoSupplier.z)*(GeoJob.z-GeoSupplier.z) )
                 /7918 ),  -- divide by 2r
        case when(mtl.MaterialType='cement')then 1
             when(mtl.MaterialType='flyash')then 2
             when(mtl.MaterialType='rebar')then 3
             when(mtl.MaterialType='mesh')then 4
             when(mtl.MaterialType='sand')then 5
             when(mtl.MaterialType='aggregate')then 6
             else 9
        end as mtlAppearanceOrder,
        -- Error message, if applicable
        case when(GeoJob.x is null)then
               'Error - Missing Geographic Data for Job Location - Zip Code = '
               + isnull(#CustomerJobData.ZipCode,'null')
             when(GeoPlant.x is null)then
               'Error - Missing Geographic Data for Plant Location - Plant = ' + isnull(#Plant.PlantTitle,'null')
             when(GeoSupplier.x is null)then
               'Error - Missing Geographic Data for Supplier Location - Plant, Material = '
               + isnull(#Plant.PlantTitle,'null') + ', ' + isnull(mtl.MaterialType,'null')
             else
               'ok'
        end as ErrMessage
from    ( select   OrgID, sum(QtyReq*Price) as Value
          from     #SalesLine
          group by OrgID
        ) PlantSales join @mtl mtl on PlantSales.OrgID=mtl.OrgID
        join ( select   OrgID, sum(QtyReq) as QtyReq
               from     @mtl
               group by OrgID
             ) mtlPlantTotal on mtl.OrgID=mtlPlantTotal.OrgID
        left join ( -- Count number of plants - to instruct whether or not to display summary table on report
                    select sum(QtyReq*Price) as SalesValue, count(distinct OrgID) as PlantCount
                    from   #SalesLine
                  ) OrderSummary on 1=1
        left join #Plant on PlantSales.OrgID=#Plant.OrgID
        left join OraclePlant on #Plant.PlantID=OraclePlant.PlantID
        left join OracleSupplier on OraclePlant.PlantID=OracleSupplier.PlantID
        and mtl.MaterialType=OracleSupplier.MaterialType
        left join #CustomerJobData on 1=1
```

```
                    left join ( -- Get a single spherical coordinate for each zip code
                                select ZipCode, Location, State, x, y, z
                                from   GeographicData join ( select   min(GeographicID) as GeographicID
                                                             from     GeographicData
                                                             group by ZipCode
                                                           ) g on GeographicData.GeographicID=g.GeographicID
                              ) GeoJob on #CustomerJobData.ZipCode=GeoJob.ZipCode
                    left join GeographicData GeoPlant on OraclePlant.GeographicID=GeoPlant.GeographicID
                    left join GeographicData GeoSupplier on OracleSupplier.GeographicID=GeoSupplier.GeographicID

            end
        end
    end
  end

if(exists(select * from @Results where ErrMessage<>'ok'))
  select distinct * from @Results where ErrMessage<>'ok'
else
  select * from @Results order by PlantName, mtlAppearanceOrder

drop table #SalesLine
drop table #BOM
drop table #SalesUOMConversion
drop table #Plant
drop table #CustomerJobData
```