Solutions for the Module 6 Assignment

This document shows a suggested solution for each problem. Solutions may vary by join styles (cross product style with tables in the FROM clause and join conditions in the WHERE clause or join operator style with join operations in the FROM clause). Solutions can also vary by alias names, order of columns in the SELECT clause, CTE names, and renaming of result columns. The suggested solutions execute on both Oracle and PostgreSQL except where noted.

Problem 1: Baskets containing item sets of two items

Write a SELECT statement to generate baskets identified by the combination of customer vendor key, date key, and branch plant key for shipments (TransTypeKey = 5). Each row should contain the basket identifying columns and a combination of two items containing item master key values. Eliminate permutations of basket items. Order the result by customer vendor key, date key, branch plant key, and the first item.

```
SELECT IF1.CustVendorKey, IF1.DateKey, IF1.BranchPlantKey,
      IF1.ItemMasterKey ItemId1, IF2.ItemMasterKey ItemId2
FROM Inventory_Fact IF1, Inventory_Fact IF2
WHERE IF1.CustVendorKey = IF2.CustVendorKey
  AND IF1.DateKey = IF2.DateKey
  AND IF1.BranchPlantKey = IF2.BranchPlantKey
  AND IF1.TransTypeKey = 5
  AND IF2.TransTypeKey = 5
  AND IF1.ItemMasterKey < IF2.ItemMasterKey
ORDER BY IF1.CustVendorKey, IF1.DateKey, IF1.BranchPlantKey,
       IF1.ItemMasterKey;
```

Problem 2: Baskets containing item sets of three items

Write a SELECT statement to generate baskets identified by the combination of customer vendor key, date key, and branch plant key for shipments (TransTypeKey = 5). Each row should contain the basket identifying columns and a combination of three items containing item master key values. Eliminate permutations of basket items. Order the result by customer vendor key, date key, and branch plant key.

```
SELECT IF1.CustVendorKey, IF1.DateKey,  IF1.BranchPlantKey,
      IF1.ItemMasterKey ItemId1, IF2.ItemMasterKey ItemId2, IF3.ItemMasterKey ItemId3
FROM Inventory_Fact IF1, Inventory_Fact IF2, Inventory_Fact IF3
WHERE IF1.CustVendorKey = IF2.CustVendorKey
  AND IF1.DateKey = IF2.DateKey
  AND IF1.BranchPlantKey = IF2.BranchPlantKey
  AND IF2.CustVendorKey = IF3.CustVendorKey
  AND IF2.DateKey = IF3.DateKey
  AND IF2.BranchPlantKey = IF3.BranchPlantKey
```

```
  AND IF1.TransTypeKey = 5
  AND IF2.TransTypeKey = 5
  AND IF3.TransTypeKey = 5
  AND IF1.ItemMasterKey < IF2.ItemMasterKey
  AND IF2.ItemMasterKey < IF3.ItemMasterKey
ORDER BY IF1.CustVendorKey, IF1.DateKey, IF1.BranchPlantKey;
```

Problem 3: Association rules of size 2 with evaluation measures (support, confidence, and lift)

Write an SQL statement with three CTEs and a SELECT statement using the CTEs to generate association rules of size 2 along with evaluation measures of support, confidence, and lift. An association rule indicates the LHS and RHS of a rule. A rule in the result contains a permutation of a combination to generate the LHS and RHS of the rule. Each row should contain the rule text (concatenation of LHS -> RHS) and evaluation measures (support, confidence, and lift) for the rule. Create three CTEs for the pairs (query for problem 1), rules, and counts followed by a SELECT statement using the CTEs. As in problem 1, baskets are identified by a combination of customer vendor key, date key, and branch plant key. Only consider shipments TransTypeKey = 5. Sort the result by descending lift and confidence measures.

```
WITH PairsCTE AS (
SELECT IF1.CustVendorKey, IF1.DateKey, IF1.BranchPlantKey,
      IF1.ItemMasterKey ItemId1, IF2.ItemMasterKey ItemId2
FROM Inventory_Fact IF1, Inventory_Fact IF2
WHERE IF1.CustVendorKey = IF2.CustVendorKey
  AND IF1.DateKey = IF2.DateKey
  AND IF1.BranchPlantKey = IF2.BranchPlantKey
  AND IF1.TransTypeKey = 5
  AND IF2.TransTypeKey = 5
  AND IF1.ItemMasterKey <> IF2.ItemMasterKey
),
RulesCTE AS (
SELECT ItemId1 || ' -> ' || ItemId2 as TheRule,
      ItemId1, ItemId2, COUNT(*) AS SupportCnt
 FROM PairsCTE
 GROUP BY ItemId1, ItemId2
),
CountProductCTE AS (
SELECT ItemMasterKey ItemId, COUNT(*) AS ProductCount
 FROM Inventory_Fact
 WHERE TransTypeKey = 5
 GROUP BY ItemMasterKey
)
-- SELECT statement using the CTEs
-- CAST function optional
```

```
SELECT R.TheRule,
 Cast(100.00 * R.SupportCnt / A.NumOrders as NUMERIC(5, 2)) AS SupportPercentage,
 Cast(100.00 * R.SupportCnt / C1.ProductCount as NUMERIC(5, 2)) AS Confidence,
 Cast((1.0 * R.SupportCnt / A.NumOrders) / ((1.0 * C1.ProductCount / A.NumOrders) *
     (1.0 * C2.ProductCount / A.NumOrders)) As NUMERIC(5, 2)) AS Lift
 FROM RulesCTE R INNER JOIN CountProductCTE C1 ON R.ItemId1 = C1.ItemId
     INNER JOIN CountProductCTE C2 ON R.ItemId2 = C2.ItemId
     CROSS JOIN (
       SELECT COUNT(*) AS NumOrders
        FROM
          ( SELECT DISTINCT CustVendorKey, DateKey, BranchPlantKey
             FROM Inventory_Fact
             WHERE TransTypeKey = 5 ) AS X) A
 ORDER BY Lift DESC, Confidence DESC;
```

Problem 4: Association rule input as a cross product of baskets and items

Write a SELECT statement to generate baskets identified by customer vendor key, date key, and branch plant key. Generate baskets containing two or more items for customers residing in CA, calendar year of 2022, and company key = 1. The result should contain customer vendor key, date key, branch plant key, item master key, and a basket indicator (1 if item is in the basket, 0 otherwise). Order the result by customer vendor key, date key, and branch plant key in ascending order.

```
SELECT C1.CustVendorKey, D1.DateKey, B1.BranchPlantKey, I1.ItemMasterKey,
 CASE WHEN (C1.CustVendorKey, D1.DateKey, B1.BranchPlantKey, I1.ItemMasterKey) IN
  ( SELECT CustVendorKey, DateKey, BranchPlantKey, ItemMasterKey
     FROM Inventory_Fact )
    THEN 1 ELSE 0 END AS Basket
 FROM Cust_Vendor_Dim C1, Date_Dim D1, Branch_Plant_Dim B1, Item_Master_Dim I1
 WHERE (C1.CustVendorKey, D1.DateKey, B1.BranchPlantKey) IN
  ( SELECT IF1.CustVendorKey, IF1.DateKey, IF1.BranchPlantKey
     FROM Inventory_Fact IF1
     GROUP BY IF1.CustVendorKey, IF1.DateKey, IF1.BranchPlantKey
     HAVING COUNT(DISTINCT IF1.ItemMasterKey) > 1 )
   AND D1.CalYear = 2022
   AND C1.State = 'CA'
   AND B1.CompanyKey = 1
 ORDER BY C1.CustVendorKey, D1.DateKey, B1.BranchPlantKey;
```

Problem 5: Association rule input as a nested list of items in each basket

Write a SELECT statement to generate shipment baskets (TransType = 5) with baskets identified by a combination of customer vendor key date key, and branch plant key. Only generate baskets with two or more items. The result should contain customer vendor key, date key, branch plant key, item master key, and an array of item master keys. Order the result by customer vendor key, date key, and branch plant key in ascending order.

```
-- Using ARRAY_AGG aggregate function
-- PostgreSQL only
-- ORDER BY clause in ARRAY_AGG is optional to order values in the result array

SELECT CustVendorKey, DateKey, BranchPlantKey,
            ARRAY_AGG(DISTINCT ItemMasterKey) AS ItemMasterKeys
FROM inventory_fact
WHERE TransTypeKey = 5
GROUP BY CustVendorKey, DateKey, BranchPlantKey
HAVING COUNT(DISTINCT ItemMasterKey) >= 2
ORDER BY CustVendorKey, DateKey, BranchPlantKey;

SELECT CustVendorKey, DateKey, BranchPlantKey,
           ARRAY_AGG(DISTINCT ItemMasterKey
            ORDER BY ItemMasterKey) AS ItemMasterKeys
FROM inventory_fact
WHERE TransTypeKey = 5
GROUP BY CustVendorKey, DateKey, BranchPlantKey
HAVING COUNT(DISTINCT ItemMasterKey) >= 2
ORDER BY CustVendorKey, DateKey, BranchPlantKey ASC;

-- Oracle LISTAGG function
-- LISTAGG can be used as an aggregate function or analytic
-- Used as an aggregate function in this solution.
-- ORDER BY clause in LISTAGG is optional to order values in the result array
-- The argument after the column is a separator.
-- Oracle only

SELECT CustVendorKey, DateKey, BranchPlantKey,
      LISTAGG(DISTINCT ItemMasterKey, ', ')
  WITHIN GROUP (ORDER BY ItemMasterKey) AS ItemMasterKeys
 FROM inventory_fact
 WHERE TransTypeKey = 5
 GROUP BY CustVendorKey, DateKey, BranchPlantKey
 HAVING COUNT(DISTINCT ItemMasterKey) >= 2
 ORDER BY CustVendorKey, DateKey, BranchPlantKey;
```

Query 6: CTE using the ROW_NUMBER analytic function for event history ordering

Write a CTE with a SELECT statement to generate shipments for a combination of customer vendor key and branch plant key. The entity in the input for a classification algorithm is the combination of customer vendor key and branch plant key. ItemMasterKey identifies items. Unit cost is the weight in descending order. Only generate rows for shipments (TransTypeKey = 5), CompanyKey = 5, and first quarter of 2022. The result should contain customer vendor key, branch plant key, customer state, customer zip, item master key, item unit cost, and the row number of the item. Partition the analytic function by customer vendor key and branch plant key. Order the analytic function by descending item unit cost. After the CTE, write a simple SELECT statement to retrieve all rows and columns of the CTE. Sort by customer vendor key and branch plant key.

```
WITH CTERankedShipments AS (
SELECT C1.CustVendorKey, IF.BranchPlantKey, C1.State, C1.Zip,
            IF.ItemMasterKey, IF.UnitCost,
 ROW_NUMBER() OVER ( PARTITION BY C1.CustVendorKey, IF.BranchPlantKey
  ORDER BY IF.UnitCost DESC ) AS UnitCostRank
 FROM Cust_Vendor_Dim C1 INNER JOIN Inventory_Fact IF
     ON C1.CustVendorKey = IF.CustVendorKey
   INNER JOIN Branch_Plant_Dim BP1 ON BP1.BranchPlantKey = IF.BranchPlantKey
   INNER JOIN Date_Dim ON Date_Dim.DateKey = IF.DateKey
 WHERE  TransTypeKey = 5
   AND BP1.CompanyKey = 5
   AND CalYear = 2022
   AND CalQuarter = 1 )

SELECT * FROM CTERankedShipments
ORDER BY CustVendorKey, BranchPlantKey;
```

Problem 7: Classification algorithm input using a CTE and a query for entities with only one event

Write a SELECT statement to generate rows with only one shipment (TransType = 5) for company key 5 in first quarter 2022. Eliminate rows not having exactly 1 shipment. Essentially, this query flattens the result of query 6 to entities having one event with default values for events 2 and 3. Use the SELECT statement from problem 6 as a CTE except for the ORDER BY clause. The result should contain the customer vendor key, branch plant key, customer state, customer zip, item master key, unit cost, and default values (0) for items 2 and 3 (both item master key and unit cost). The combination of the customer vendor key and branch plant key represent the entity in a row. The item number and unit cost in a row should be the values of the item with maximum unit cost related to the entity (combination of customer vendor key and branch plant key). Order the result by customer vendor key and branch plant key.

```
WITH CTERankedShipments AS (
SELECT C1.CustVendorKey, IF.BranchPlantKey, C1.State, C1.Zip,
            IF.ItemMasterKey, IF.UnitCost,
 ROW_NUMBER() OVER ( PARTITION BY C1.CustVendorKey, IF.BranchPlantKey
  ORDER BY IF.UnitCost DESC ) AS UnitCostRank
 FROM Cust_Vendor_Dim C1 INNER JOIN Inventory_Fact IF ON C1.CustVendorKey =
IF.CustVendorKey
   INNER JOIN Branch_Plant_Dim BP1 ON BP1.BranchPlantKey = IF.BranchPlantKey
   INNER JOIN Date_Dim ON Date_Dim.DateKey = IF.DateKey
 WHERE  TransTypeKey = 5
   AND BP1.CompanyKey = 5
   AND CalYear = 2022
   AND CalQuarter = 1 )

SELECT CustVendorKey, BranchPlantKey, State, Zip, ItemMasterKey AS Item1,
           UnitCost AS UnitCost1, 0 AS Item2, 0 AS UnitCost2, 0 AS Item3, 0 AS UnitCost3
 FROM CTERankedShipments
 WHERE ( CustVendorKey, BranchPlantKey ) IN
  ( SELECT CustVendorKey, BranchPlantKey
     FROM CTERankedShipments
     GROUP BY CustVendorKey, BranchPlantKey
     HAVING MAX(UnitCostRank) = 1 )
 ORDER BY CustVendorKey, BranchPlantKey;
```

Problem 8: Classification algorithm input using a CTE and query for entities with exactly two events

Write a SELECT statement to generate rows with exactly two shipments (TransType = 5) for company key 5 in first quarter 2022. Eliminate rows not having exactly 2 shipments. Essentially, this query flattens the result of query 6 to entities having exactly two shipments. Use the SELECT statement from problem 6 as a CTE except for the ORDER BY clause. The result should contain the customer vendor key, branch plant key, customer state, customer zip, item master key for item 1, unit cost for item 1, item master key for item 2, unit cost for item 2, and default values (0) for item 3 (both item master key and unit cost). The combination of the customer vendor key and branch plant key represent the entity in a row. The item number and unit cost values in a row should be the values of the items with row numbers 1 and 2. Order the result by customer vendor key and branch plant key.

```
WITH CTERankedShipments AS (
SELECT C1.CustVendorKey, IF.BranchPlantKey, C1.State, C1.Zip, IF.ItemMasterKey,
        IF.UnitCost,
 ROW_NUMBER() OVER ( PARTITION BY C1.CustVendorKey, IF.BranchPlantKey
  ORDER BY IF.UnitCost DESC ) AS UnitCostRank
```

```
 FROM Cust_Vendor_Dim C1 INNER JOIN Inventory_Fact IF ON C1.CustVendorKey =
IF.CustVendorKey
  INNER JOIN Branch_Plant_Dim BP1 ON BP1.BranchPlantKey = IF.BranchPlantKey
  INNER JOIN Date_Dim ON Date_Dim.DateKey = IF.DateKey
 WHERE  TransTypeKey = 5
  AND BP1.CompanyKey = 5
  AND CalYear = 2022
  AND CalQuarter = 1 )

SELECT CTE1.CustVendorKey, CTE1.BranchPlantKey, CTE1.State, CTE1.Zip,
     CASE 1 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
               ELSE CTE2.ItemMasterKey END AS Item1,
     CASE 1 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
                ELSE CTE2.UnitCost END AS UnitCost1,
     CASE 2 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
               ELSE CTE2.ItemMasterKey END AS Item2,
     CASE 2 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
               ELSE CTE2.UnitCost END AS UnitCost2,
     0 AS ItemId3, 0 AS UnitCost3
FROM CTERankedShipments CTE1, CTERankedShipments CTE2
WHERE ( CTE1.CustVendorKey, CTE1.BranchPlantKey ) IN
  ( SELECT CustVendorKey, BranchPlantKey
    FROM CTERankedShipments
    GROUP BY CustVendorKey, BranchPlantKey
    HAVING MAX(UnitCostRank) = 2 )
  AND CTE1.CustVendorKey = CTE2.CustVendorKey
  AND CTE1.BranchPlantKey = CTE2.BranchPlantKey
  AND CTE1.ItemMasterKey < CTE2.ItemMasterKey
ORDER BY CustVendorKey, BranchPlantKey;

-- CASE functions can also be written with the column name before the WHEN clause

     CASE CTE1.UnitCostRank WHEN 1 THEN CTE1.ItemMasterKey
               ELSE CTE2.ItemMasterKey END AS Item1,
     CASE CTE1.UnitCostRank WHEN 1 THEN CTE1.UnitCost
                ELSE CTE2.UnitCost END AS UnitCost1,
     CASE CTE1.UnitCostRank WHEN 2 THEN CTE1.ItemMasterKey
               ELSE CTE2.ItemMasterKey END AS Item2,
     CASE CTE1.UnitCostRank WHEN 2 THEN CTE1.UnitCost
               ELSE CTE2.UnitCost END AS UnitCost2,
     0 AS ItemId3, 0 AS UnitCost3
```

Problem 9: Classification algorithm input using a CTE and query for entities with exactly three events

Write a SELECT statement to generate rows with 3 or more shipments (TransType = 5) for company key 5 in first quarter 2022. The result should contain only the largest 3 values for unit cost for these shipments. Use the SELECT statement from problem 6 as a CTE. The result should contain the customer vendor key, branch plant key, customer state, customer zip, item master key for item 1, unit cost for item 1, item master key for item 2, unit cost for item 2, and item master key for item 3, unit cost for item 3. The combination of the customer vendor key and branch plant key represent the entity in a row. The item number and unit cost values in a row should be the values of the shipments with row numbers 1 to 3. Order the result by customer vendor key and branch plant key.

```
WITH CTERankedShipments AS (
SELECT C1.CustVendorKey, IF.BranchPlantKey, C1.State, C1.Zip, IF.ItemMasterKey,
          IF.UnitCost,
  ROW_NUMBER() OVER ( PARTITION BY C1.CustVendorKey, IF.BranchPlantKey
   ORDER BY IF.UnitCost DESC ) AS UnitCostRank
 FROM Cust_Vendor_Dim C1 INNER JOIN Inventory_Fact IF ON C1.CustVendorKey =
IF.CustVendorKey
   INNER JOIN Branch_Plant_Dim BP1 ON BP1.BranchPlantKey = IF.BranchPlantKey
   INNER JOIN Date_Dim ON Date_Dim.DateKey = IF.DateKey
 WHERE  TransTypeKey = 5
   AND BP1.CompanyKey = 5
   AND CalYear = 2022
   AND CalQuarter = 1 )

SELECT CTE1.CustVendorKey, CTE1.BranchPlantKey, CTE1.State, CTE1.Zip,
      CASE 1 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
                ELSE CTE2.ItemMasterKey END AS Item1,
      CASE 1 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
                ELSE CTE2.UnitCost END AS UnitCost1,
      CASE 2 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
                ELSE CTE2.ItemMasterKey END AS Item2,
      CASE 2 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
               ELSE CTE2.UnitCost END AS UnitCost2,
      CASE 3 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
                WHEN CTE2.UnitCostRank THEN CTE2.ItemMasterKey
                 ELSE CTE3.ItemMasterKey END AS Item3,
      CASE 3 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
                WHEN CTE2.UnitCostRank THEN CTE2.UnitCost
                 ELSE CTE3.UnitCost END AS UnitCost3
 FROM CTERankedShipments CTE1, CTERankedShipments CTE2,
          CTERankedShipments CTE3
WHERE ( CTE1.CustVendorKey, CTE1.BranchPlantKey ) IN
  ( SELECT CustVendorKey, BranchPlantKey
     FROM CTERankedShipments
     GROUP BY CustVendorKey, BranchPlantKey
     HAVING MAX(UnitCostRank) >= 3 )
```

```
  AND CTE1.CustVendorKey = CTE2.CustVendorKey
  AND CTE1.BranchPlantKey = CTE2.BranchPlantKey
  AND CTE1.ItemMasterKey < CTE2.ItemMasterKey
  AND CTE2.CustVendorKey = CTE3.CustVendorKey
  AND CTE2.BranchPlantKey = CTE3.BranchPlantKey
  AND CTE2.ItemMasterKey < CTE3.ItemMasterKey
  AND CTE1.UnitCostRank < 4
  AND CTE2.UnitCostRank < 4
  AND CTE3.UnitCostRank < 4
ORDER BY CustVendorKey, BranchPlantKey;

-- CASE functions can also be written with the column name before the WHEN clause

    CASE CTE1.UnitCostRank WHEN 1 THEN CTE1.ItemMasterKey
            ELSE CTE2.ItemMasterKey END AS Item1,
    CASE CTE1.UnitCostRank WHEN 1 THEN CTE1.UnitCost
            ELSE CTE2.UnitCost END AS UnitCost1,
    CASE CTE1.UnitCostRank WHEN 2 THEN CTE1.ItemMasterKey
            ELSE CTE2.ItemMasterKey END AS Item2,
    CASE CTE1.UnitCostRank WHEN 2 THEN CTE1.UnitCost
           ELSE CTE2.UnitCost END AS UnitCost2,
    CASE CTE1.UnitCostRank WHEN 3 THEN CTE1.ItemMasterKey
            WHEN CTE2.UnitCostRank THEN CTE2.ItemMasterKey
             ELSE CTE3.ItemMasterKey END AS Item3,
    CASE CTE1.UnitCostRank WHEN 3 THEN CTE1.UnitCost
            WHEN CTE2.UnitCostRank THEN CTE2.UnitCost
             ELSE CTE3.UnitCost END AS UnitCost3
```

Problem 10: Classification algorithm input using a CTE and union of queries for entities with a range of event sizes (1 to 3)

The result should contain rows with one to three shipments. The result should contain columns for the customer vendor key, branch plant key, customer state, customer zip, item master key for item 1, unit cost for item 1, item master key for item 2, unit cost for item 2, and item master key for item 3, and unit cost for item 3. The statement should use the SELECT statement from problem 6 as a CTE along with a UNION of SELECT statements from problems 7 to 9. Order the result by customer vendor key and branch plant key. Note that a SELECT statement can only contain a single ORDER BY clause at the end of the statement. Thus, your statement should remove the ORDER BY clauses in statements for problems 6 to 8.

```
WITH CTERankedShipments AS (
SELECT C1.CustVendorKey, IF.BranchPlantKey, C1.State, C1.Zip, IF.ItemMasterKey,
        IF.UnitCost,
        ROW_NUMBER() OVER ( PARTITION BY C1.CustVendorKey, IF.BranchPlantKey
        ORDER BY IF.UnitCost DESC ) AS UnitCostRank
```

```
 FROM Cust_Vendor_Dim C1 INNER JOIN Inventory_Fact IF
          ON C1.CustVendorKey = IF.CustVendorKey
   INNER JOIN Branch_Plant_Dim BP1 ON BP1.BranchPlantKey = IF.BranchPlantKey
   INNER JOIN Date_Dim ON Date_Dim.DateKey = IF.DateKey
 WHERE  TransTypeKey = 5
   AND BP1.CompanyKey = 5
   AND CalYear = 2022
   AND CalQuarter = 1 )

SELECT CustVendorKey, BranchPlantKey, State, Zip, ItemMasterKey AS Item1,
           UnitCost AS UnitCost1, 0 AS Item2, 0 AS UnitCost2, 0 AS Item3, 0 AS UnitCost3
 FROM CTERankedShipments
 WHERE ( CustVendorKey, BranchPlantKey ) IN
   ( SELECT CustVendorKey, BranchPlantKey
     FROM CTERankedShipments
     GROUP BY CustVendorKey, BranchPlantKey
     HAVING MAX(UnitCostRank) = 1 )

UNION

SELECT CTE1.CustVendorKey, CTE1.BranchPlantKey, CTE1.State, CTE1.Zip,
      CASE 1 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
                ELSE CTE2.ItemMasterKey END AS Item1,
      CASE 1 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
                 ELSE CTE2.UnitCost END AS UnitCost1,
      CASE 2 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
                ELSE CTE2.ItemMasterKey END AS Item2,
      CASE 2 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
                ELSE CTE2.UnitCost END AS UnitCost2,
      0 AS ItemId3, 0 AS UnitCost3
FROM CTERankedShipments CTE1, CTERankedShipments CTE2
WHERE ( CTE1.CustVendorKey, CTE1.BranchPlantKey ) IN
   ( SELECT CustVendorKey, BranchPlantKey
     FROM CTERankedShipments
     GROUP BY CustVendorKey, BranchPlantKey
     HAVING MAX(UnitCostRank) = 2 )
   AND CTE1.CustVendorKey = CTE2.CustVendorKey
   AND CTE1.BranchPlantKey = CTE2.BranchPlantKey
   AND CTE1.ItemMasterKey < CTE2.ItemMasterKey

UNION

SELECT CTE1.CustVendorKey, CTE1.BranchPlantKey, CTE1.State, CTE1.Zip,
      CASE 1 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
                ELSE CTE2.ItemMasterKey END AS Item1,
      CASE 1 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
```

```
            ELSE CTE2.UnitCost END AS UnitCost1,
      CASE 2 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
            ELSE CTE2.ItemMasterKey END AS Item2,
      CASE 2 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
           ELSE CTE2.UnitCost END AS UnitCost2,
      CASE 3 WHEN CTE1.UnitCostRank THEN CTE1.ItemMasterKey
            WHEN CTE2.UnitCostRank THEN CTE2.ItemMasterKey
             ELSE CTE3.ItemMasterKey END AS Item3,
      CASE 3 WHEN CTE1.UnitCostRank THEN CTE1.UnitCost
            WHEN CTE2.UnitCostRank THEN CTE2.UnitCost
             ELSE CTE3.UnitCost END AS UnitCost3
 FROM CTERankedShipments CTE1, CTERankedShipments CTE2,
        CTERankedShipments CTE3
WHERE ( CTE1.CustVendorKey, CTE1.BranchPlantKey ) IN
  ( SELECT CustVendorKey, BranchPlantKey
    FROM CTERankedShipments
    GROUP BY CustVendorKey, BranchPlantKey
    HAVING MAX(UnitCostRank) >= 3 )
  AND CTE1.CustVendorKey = CTE2.CustVendorKey
  AND CTE1.BranchPlantKey = CTE2.BranchPlantKey
  AND CTE1.ItemMasterKey < CTE2.ItemMasterKey
  AND CTE2.CustVendorKey = CTE3.CustVendorKey
  AND CTE2.BranchPlantKey = CTE3.BranchPlantKey
  AND CTE2.ItemMasterKey < CTE3.ItemMasterKey
  AND CTE1.UnitCostRank < 4
  AND CTE2.UnitCostRank < 4
  AND CTE3.UnitCostRank < 4
ORDER BY CustVendorKey, BranchPlantKey;
```