Strategies in Produmex Scan

This documentation describes the steps of configuring and customizing the incoming and replenishment strategies in Produmex Scan.

Incoming Strategy

The incoming strategy is a user query that is run periodically by the Produmex Scan server component. The output of the query creates Stock Transfer Draft documents if stock move is required.

Limitations:
The Stock Transfer Draft document is not created in the following cases:

- The item is managed by serial numbers, the management method is On Release Only and the serial number is not provided.
- The item is managed by batch numbers or serial numbers and on the Item Master Data window the Issue Primarily By setting is set to Bin Locations.

Configuration

Settings

Configure the following setting on the Produmex Scan Strategies tab.

Incoming strategies UQ name (default: bxmobilewh9_strategy_incoming)
Incoming strategies frequency (default: 300 seconds = 5 minutes)

Query input, output

By default, the user query must be named 'bxmobilewh9_strategy_incoming', but this can be changed in the configuration.

Input: the user query takes no input
Output: the user query must return a table with the recommended movement results.

The result table columns are:

- ItemCode
- BatchNumber (only applicable for batch items, otherwise leave empty)
- SerialNumber (only applicable for serial items, otherwise leave empty)
- Quantity (in inventory UoM)
- SourceLocation (where to move from)
- DestinationLocation (where to move to)
- GroupID

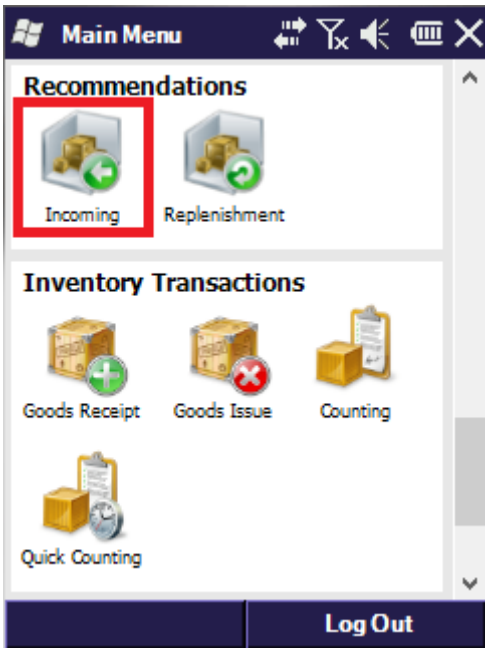
- Remarks

The GroupID result column can be used to group created output documents: for each different groupID, a different *Stock Transfer - Draft* document will be created.

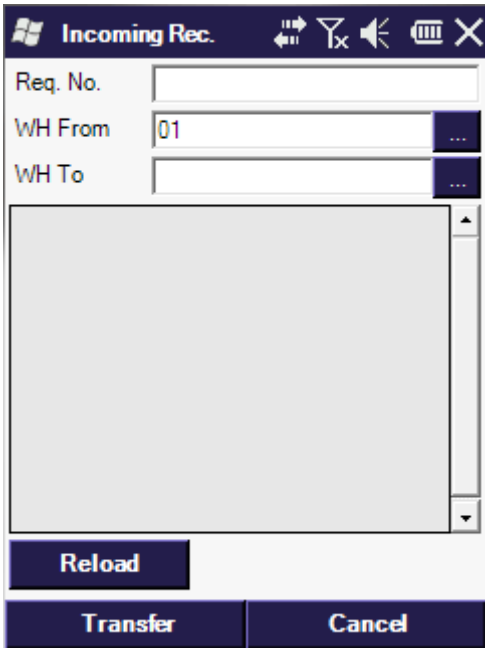
Logging, monitoring

Mobile interface

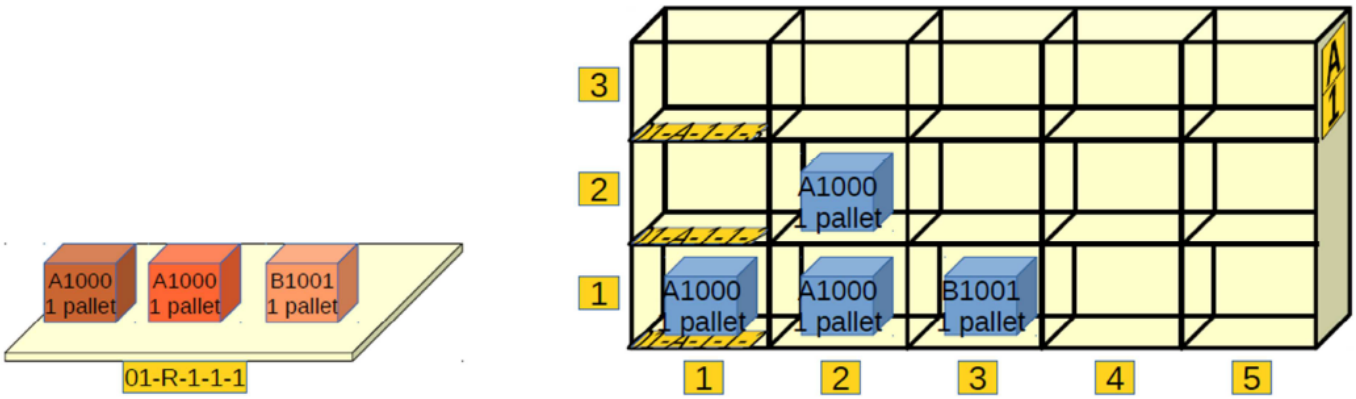
Incoming strategy recommendation results transfers can be accessed from the Produmex Scan main menu, with the 'Incoming Recommendations' icon.



On the incoming recommendations screen, you will see all the *Stock Transfer - Draft* documents, where the Transaction Type user-defined field is 'Incoming'. It is possible to filter by 'From Warehouse' or 'To Warehouse'.
After selecting the appropriate document, press the Transfer button to start working on it.

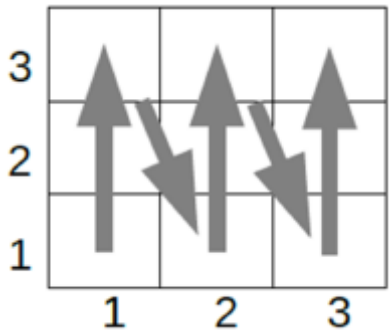


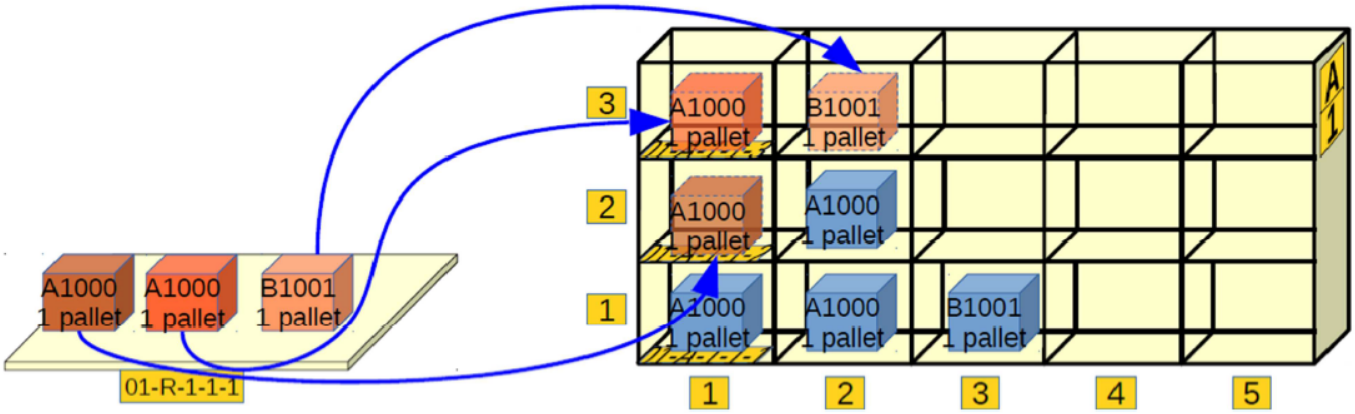
Example incoming strategy



In this example, we have an incoming location (01-R-1-1-1) where incoming trucks unload the goods. Our incoming strategy will check if there is any stock on this location, and automatically recommend moving it to empty locations in the warehouse shelves (locations 01-A-1-*-*).
The algorithm which is implemented in an SQL user query:

1. Check if incoming locations (01-R-1-1-1) have stock
2. Look at the stock, see if it has a recommendation already
3. If not, then find a locations to move to – the next empty location in 01-A-1-*-*. Order of looking for empty locations is by shelf column then level (see figure below).
4. Return results to create recommendations for moving the incoming the item





In this example scenario on the picture above, the algorithm will return the following recommendations:

Item	Batch	From bin	To bin	Quantity
A1000		01-R-1-1-1	01-A-1-1-2	1 (pallet)*
A1000		01-R-1-1-1	01-A-1-1-3	1 (pallet)*
B1001	B12345	01-R-1-1-1	01-A-1-2-3	

**Quantity is always in inventory units, pallet units are only for illustration purposes here.*

Replenishment Strategy

Configuration

Settings

Configure the following setting on the Produmex Scan Strategies tab.

Replenishment strategies UQ name (default: bxmobilewh9_strategy_replenishment)
Replenishment strategies frequency (default: 300 seconds = 5 minutes)

Query input, output

By default, the user query must be named 'bxmobilewh9_strategy_replenishment', but this can be

changed in the configuration.

Input: the user query takes no input
Output: the user query must return a table with the recommended movement results.

The result table columns are:

- ItemCode
- BatchNumber (only applicable for batch items, otherwise leave empty)
- SerialNumber (only applicable for serial items, otherwise leave empty)
- Quantity (in inventory UoM)
- SourceLocation (where to move from)
- DestinationLocation (where to move to)
- GroupID
- Remarks

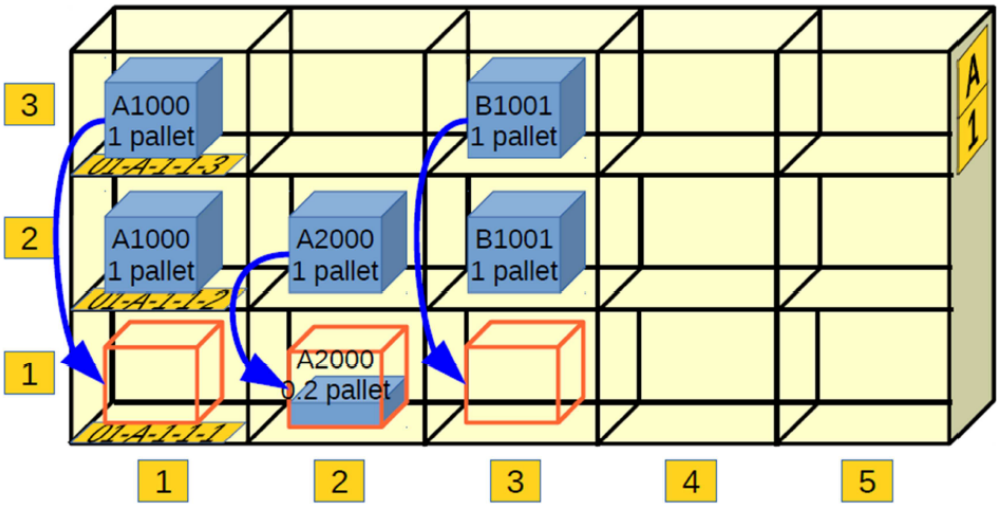
The GroupID result column can be used to group created output documents: for each different groupID, a different Stock Transfer Draft document will be created.

Logging, monitoring

Mobile interface

See the section 'Incoming Strategy – Mobile interface'. The work flow is the same, but you have to select the 'Replenishment recommendations' icon on the main screen.

Example replenishment strategy

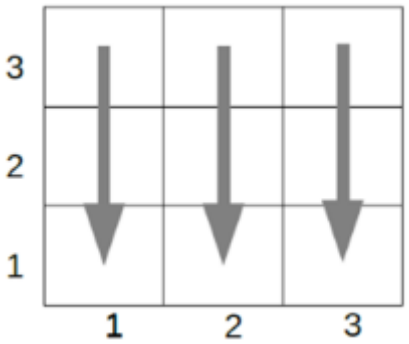


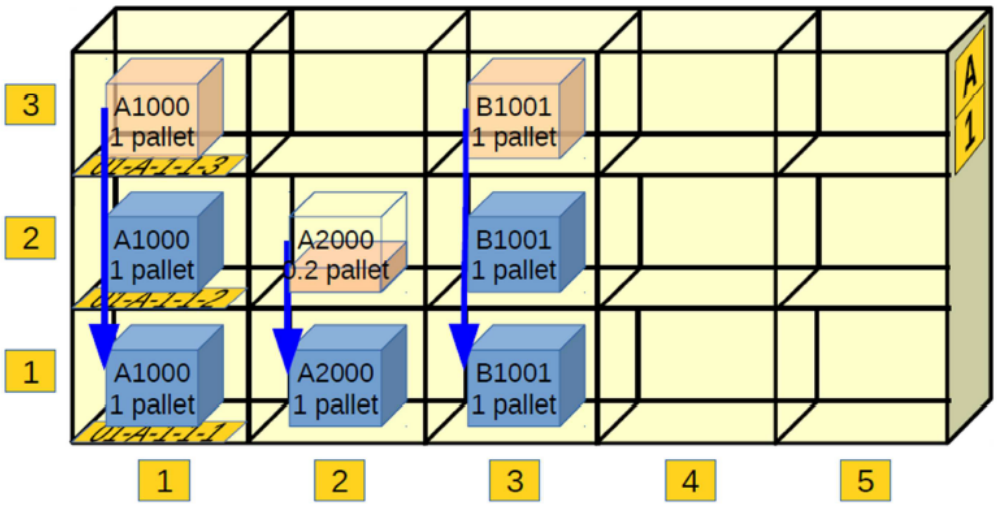
Replenishment strategy can be used to refill easily-reached locations in a warehouse system. In this example warehouse, the locations on the first level (01-A-1-*-1) are reachable by every warehouse worker, but higher levels (01-A-1-*-2, 3) are only reachable by fork lifts.
The refill algorithm, which is implemented in SQL query:

1. Check first level locations (01-A-1-*-1), see if it's empty or it's below minimum level (25% of

pallet).

2. Look for refill sources in the same column, but higher levels. If found, recommend moving items from upper levels to lower levels





In this example scenario on the picture above, the algorithm will return the following recommendations:

Item	Batch	From bin	To bin	Quantity
A1000		01-A-1-1-3	01-A-1-1-2	1 (pallet)*
A2000		01-A-1-2-3	01-A-1-1-3	0.8 (pallet)*
B1001	B12345	01-A-1-3-3	01-A-1-2-3	1 (pallet)*

**Quantity is always in inventory units, pallet units are only for illustration purposes here.*

Appendix

Helper view

MS SQL version

```
CREATE VIEW XXX_INVTRANSFER_DRAFT_LINESBIN
AS
SELECT DRF1.DocEntry, DRF1.ItemCode, DRF1.LineNum, fromBin.BinCode AS
```

```
FromBin,
fromBinLine.Quantity AS FromQuantity,
toBin.BinCode AS ToBin,
toBinLine.Quantity AS ToQuantity
FROM ODRF, DRF1
LEFT OUTER JOIN DRF19 fromBinLine ON (fromBinLine.ObjType = 67
AND fromBinLine.DocEntry = DRF1.DocEntry
AND fromBinLine.LineNum = DRF1.LineNum
AND fromBinLine.BinActTyp = 2)
LEFT OUTER JOIN DRF19 toBinLine ON (toBinLine.ObjType = 67
AND toBinLine.DocEntry = DRF1.DocEntry
AND toBinLine.LineNum = DRF1.LineNum
AND toBinLine.BinActTyp = 1)
LEFT OUTER JOIN OBIN fromBin ON (fromBin.AbsEntry = fromBinLine.BinAbs)
LEFT OUTER JOIN OBIN toBin ON (toBin.AbsEntry = toBinLine.BinAbs)
WHERE ODRF.ObjType = 67
AND DRF1.ObjType = 67
AND toBinLine.ObjType = 67
AND ODRF.DocEntry = DRF1.DocEntry
```

HANA version

```
CREATE VIEW "PMXSCAN19_2"."XXX_INVTRANSFER_DRAFT_LINESBIN" ( "DocEntry",
     "ItemCode",
     "LineNum",
     "fromBin",
     "FromQuantity",
     "ToBin",
     "ToQuantity" ) AS SELECT
     "DRF1"."DocEntry",
     "DRF1"."ItemCode",
     "DRF1"."LineNum",
     "fromBin"."BinCode" AS "fromBin",
     "fromBinLine"."Quantity" AS "FromQuantity",
     "toBin"."BinCode" AS "ToBin",
     "toBinLine"."Quantity" AS "ToQuantity"
FROM "ODRF",
     "DRF1"
LEFT OUTER JOIN "DRF19" AS "fromBinLine" ON ("fromBinLine"."ObjType" = 67
    AND "fromBinLine"."DocEntry" = "DRF1"."DocEntry"
    AND "fromBinLine"."LineNum" = "DRF1"."LineNum"
    AND "fromBinLine"."BinActTyp" = 2)
LEFT OUTER JOIN "DRF19" AS "toBinLine" ON ("toBinLine"."ObjType" = 67
    AND "toBinLine"."DocEntry" = "DRF1"."DocEntry"
    AND "toBinLine"."LineNum" = "DRF1"."LineNum"
    AND "toBinLine"."BinActTyp" = 1)
LEFT OUTER JOIN "OBIN" "fromBin" ON ("fromBin"."AbsEntry" =
"fromBinLine"."BinAbs")
LEFT OUTER JOIN "OBIN" "toBin" ON ("toBin"."AbsEntry" =
"toBinLine"."BinAbs")
WHERE "ODRF"."ObjType" = 67
```

```
AND "DRF1"."ObjType" = 67
AND "toBinLine"."ObjType" = 67
AND "ODRF"."DocEntry" = "DRF1"."DocEntry" WITH READ ONLY
```

Example Incoming Strategy

Algorithm:

1. Look for stock on RecBinLocation (eg. 01-Q), only non-serial, non-batch items
2. Subtract/ignore stock quantity already recommended for moving
3. Recommend moving remaining quantity to locations defined by OutLocationFilter eg. 01-F-%, look for empty locations. Only put one PurchaseUoM (eg. pallet) quantity on one out location, split moving to multiple out locations if needed.

Example Incoming Strategy SQL

```
DECLARE @RecBinLocation nvarchar(MAX)
DECLARE @OutLocationFilter nvarchar(MAX)
DECLARE @ItemCode nvarchar(MAX)
DECLARE @Quantity DECIMAL
DECLARE @SourceLocation nvarchar(MAX)
DECLARE @DestinationLocation nvarchar(MAX)
DECLARE @PurchaseUomQuantity nvarchar(MAX)
DECLARE @QuantityPartial DECIMAL
DECLARE @ExcludeBinList nvarchar(MAX)
DECLARE @QuantityAlready DECIMAL

/*
@RecBinLocation: receiving bin location
@OutLocationFilter: target bin location filter
*/
SET @RecBinLocation = '01-S'
SET @OutLocationFilter = '01-%'
SET @ExcludeBinList = ''

-- temp table for moving
DECLARE @RESULT TABLE (
       ItemCode nvarchar(MAX),
       BatchNumber nvarchar(MAX),
       SerialNumber nvarchar(MAX),
       Quantity DECIMAL,
       SourceLocation nvarchar(MAX),
       DestinationLocation nvarchar(MAX),
       GroupID nvarchar(MAX),
       Remarks nvarchar(MAX)
)

-- select stock on the receiving bin location
DECLARE curs CURSOR FOR
```

```
SELECT OIBQ.ItemCode, OIBQ.OnHandQty, OBIN.BinCode
FROM OBIN, OIBQ, OITM
WHERE OBIN.BinCode = @RecBinLocation
       AND OIBQ.BinAbs = OBIN.AbsEntry
       AND OITM.ItemCode = OIBQ.ItemCode
       AND OIBQ.OnHandQty > 0
       AND OITM.ManBtchNum ='N'
       AND OITM.ManSerNum = 'N'

-- cursor for checkin items one by one
OPEN curs
FETCH NEXT FROM curs INTO @ItemCode, @Quantity, @SourceLocation
WHILE @@FETCH_STATUS = 0
BEGIN
       SET @PurchaseUomQuantity = NULL
       SELECT @PurchaseUomQuantity = BaseQty
       FROM UGP1, OITM
       WHERE UGP1.UgpEntry = OITM.UgpEntry AND UGP1.UomEntry =
OITM.PUomEntry AND OITM.ItemCode = @ItemCode

       -- check the quantity that has already been added on a previous stock
transfer draft document
       SET @QuantityAlready = NULL
       SELECT @QuantityAlready = SUM(FromQuantity)
       FROM XXX_INVTRANSFER_DRAFT_LINESBIN
       WHERE ItemCode = @ItemCode AND FromBin = @SourceLocation


       IF @QuantityAlready IS NOT NULL BEGIN SET @Quantity = @Quantity -
@QuantityAlready END

       WHILE @Quantity > 0 BEGIN
              SET @QuantityPartial = @Quantity
              IF @PurchaseUomQuantity IS NOT NULL AND @QuantityPartial >
@PurchaseUomQuantity BEGIN
                     SET @QuantityPartial = @PurchaseUomQuantity
              END

              -- make some logic to get the target bin location
              SET @DestinationLocation = NULL
              SELECT TOP 1 @DestinationLocation = BinCode
                     FROM OBIN LEFT OUTER JOIN OIBQ ON (OBIN.AbsEntry =
OIBQ.AbsEntry)
                     WHERE OBIN.BinCode LIKE @OutLocationFilter
                           AND charindex(OBIN.BinCode, @ExcludeBinList) <= 0
                           AND OBIN.BinCode NOT IN
                           (SELECT ToBin FROM XXX_INVTRANSFER_DRAFT_LINESBIN
                                  WHERE ToBin LIKE @OutLocationFilter)
                                  GROUP BY BinCode HAVING SUM(OnHandQty) =
OR SUM(OnHandQty) IS NULL
              ORDER BY BinCode
```

```
            -- if no more places, DestinationLocation will be empty
            INSERT INTO @RESULT (ItemCode, Quantity, SourceLocation,
DestinationLocation)
            VALUES (@ItemCode, @QuantityPartial, @SourceLocation,
@DestinationLocation)

SET @Quantity = @Quantity - @QuantityPartial
SET @ExcludeBinList = @ExcludeBinList + ' ' + @DestinationLocation
END
FETCH NEXT FROM curs INTO @ItemCode, @Quantity, @SourceLocation
END
CLOSE curs
DEALLOCATE curs
SELECT * FROM @RESULT
```

Example Incoming Strategy HANA

```
CREATE PROCEDURE PMXSCAN_PMXSCAN_INCOMINGSTRATEGY (
)
LANGUAGE SQLSCRIPT
AS
BEGIN
    DECLARE RecBinLocation NVARCHAR(200);
    DECLARE OutLocationFilter NVARCHAR(200);
    DECLARE ItemCode NVARCHAR(200);
    DECLARE Quantity DECIMAL(21,6);
    DECLARE SourceLocation NVARCHAR(200);
    DECLARE DestinationLocation NVARCHAR(200);
    DECLARE PurchaseUomQuantity NVARCHAR(200);
    DECLARE QuantityPartial DECIMAL(21,6);
    DECLARE ExcludeBinList NVARCHAR(2000);
    DECLARE BatchNum NVARCHAR(200);
    DECLARE QuantityAlready NVARCHAR(200);
    DECLARE TableExists INT;
    DECLARE CURSOR curs FOR
        SELECT "OIBQ"."ItemCode", "OIBQ"."OnHandQty", "OBIN"."BinCode" FROM
"OBIN", "OIBQ", "OITM"
        WHERE "OBIN"."BinCode" = RecBinLocation AND "OIBQ"."BinAbs" =
"OBIN"."AbsEntry" AND
            "OITM"."ItemCode" = "OIBQ"."ItemCode" AND "OIBQ"."OnHandQty" > 0
            AND "OITM"."ManBtchNum" ='N' AND "OITM"."ManSerNum" = 'N';
    /*
    SELECT COUNT(*) INTO TableExists FROM "PUBLIC"."M_TEMPORARY_TABLES"
WHERE "TABLE_NAME" = '#RESULT';
    IF TableExists > 0 THEN
        DROP TABLE #result;
    END IF;
    */
    CREATE LOCAL TEMPORARY TABLE #result
```

```
    (
        "ItemCode" NVARCHAR(200),
        "BatchNumber" NVARCHAR(200),
        "SerialNumber" NVARCHAR(200),
        "Quantity" DECIMAL(21,6),
        "SourceLocation" NVARCHAR(200),
        "DestinationLocation" NVARCHAR(200),
        "GroupID" NVARCHAR(200),
        "Remarks" NVARCHAR(200)
    );
    RecBinLocation := '01-S';
    OutLocationFilter := '01-F%';
    ExcludeBinList := '01-SYSTEM-BIN-LOCATION';

    FOR c_ as curs DO
        ItemCode := c_."ItemCode";
        Quantity := c_."OnHandQty";
        SourceLocation := c_."BinCode";
        PurchaseUomQuantity := NULL;
        SELECT SUM("BaseQty") into PurchaseUomQuantity FROM "UGP1", "OITM"
            WHERE "UGP1"."UgpEntry" = "OITM"."UgpEntry" AND
"UGP1"."UomEntry" = "OITM"."PUoMEntry" AND "OITM"."ItemCode" = ItemCode;
        QuantityAlready := NULL;
        SELECT SUM("FromQuantity") into QuantityAlready FROM
"XXX_INVTRANSFER_DRAFT_LINESBIN"
            WHERE "ItemCode" = ItemCode AND "FromBin" = SourceLocation;
        IF QuantityAlready IS NOT NULL THEN
            Quantity := Quantity - QuantityAlready;
        END IF;
        WHILE Quantity > 0 DO
            QuantityPartial := Quantity;
            IF PurchaseUomQuantity IS NOT NULL AND QuantityPartial >
PurchaseUomQuantity THEN
                QuantityPartial := PurchaseUomQuantity;
            END IF;
            DestinationLocation := NULL;
            SELECT TOP 1 "BinCode" into DestinationLocation
                FROM "OBIN" LEFT OUTER JOIN "OIBQ" ON ("OBIN"."AbsEntry" =
"OIBQ"."BinAbs")
                WHERE "OBIN"."BinCode" LIKE OutLocationFilter
                AND LOCATE(ExcludeBinList, "OBIN"."BinCode") <= 0
                AND "OBIN"."BinCode" NOT IN (SELECT "ToBin" FROM
"XXX_INVTRANSFER_DRAFT_LINESBIN"
                WHERE "ToBin" LIKE OutLocationFilter)
                GROUP BY "BinCode" HAVING SUM("OnHandQty") = 0 OR
SUM("OnHandQty") IS NULL
                ORDER BY "BinCode";

            INSERT INTO #result VALUES (ItemCode, BatchNum, '',
QuantityPartial, SourceLocation, DestinationLocation, '', '');
            Quantity := Quantity - QuantityPartial;
```

```
            ExcludeBinList := ExcludeBinList || ' ' || DestinationLocation;
        END WHILE;
    END FOR;
    SELECT * FROM #result;
    DROP TABLE #result;
END;


call "BXINCOMINGSTRATEGY"()
```

Example Replenishment Strategy

Algorithm:

1. See what stocks are on floor (*-1) locations and on upper (*-2. *-3, ec) locations for the same itemcode. (Only normal items are considered, and which have purchase uom groups defined) Only look in bin locations 01-F-* (ScanArea0)
2. If floor quantity ⇐ 50% of pallet quantity, look for upper locations. Also consider Inventory Transfer Draft documents aleady recorded. Recommend moving quantities from upper locations until floor quantity reaches 100% of pallet quantity if possible.

Example Replenishment Strategy MS SQL

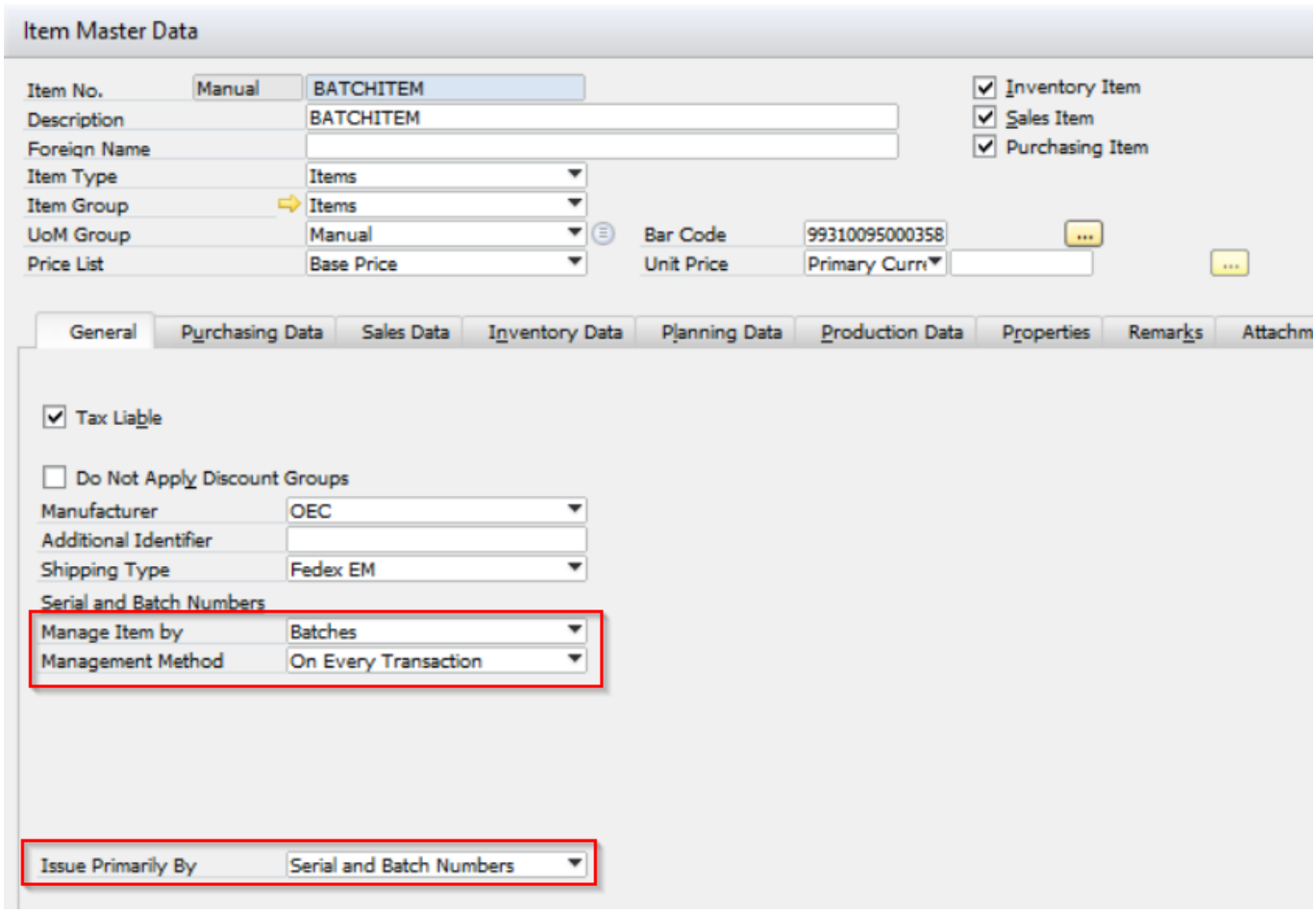
```
DECLARE @ScanArea nvarchar(MAX)
DECLARE @FloorLocation4 nvarchar(MAX)
SET @ScanArea = '01-F-%'
SET @FloorLocation4 = '1'

DECLARE @FloorBin nvarchar(MAX)
DECLARE @ItemCode nvarchar(MAX)
DECLARE @FloorQuantity DECIMAL
DECLARE @UpperBin nvarchar(MAX)
DECLARE @UpperQuantity DECIMAL
DECLARE @PalletQty DECIMAL
DECLARE @QuantityPartial DECIMAL
DECLARE @LastBin nvarchar(MAX)
DECLARE @LastBinFloorQuantity DECIMAL
DECLARE @QuantityNeeded DECIMAL
DECLARE @QuantityAlready DECIMAL
DECLARE @RESULT TABLE (
ItemCode nvarchar(MAX),
BatchNumber nvarchar(MAX),
SerialNumber nvarchar(MAX),
Quantity DECIMAL,
SourceLocation nvarchar(MAX),
DestinationLocation nvarchar(MAX),
GroupID nvarchar(MAX),
Remarks nvarchar(MAX)
)
```

```
DECLARE curs CURSOR FOR
SELECT
binupper.FloorBinCode2 AS FloorBin,
binupper.ItemCode AS ItemCode,
binfloor.OnHandQty AS FloorQuantity,
binupper.BinCode AS UpperBin,
binupper.OnHandQty AS UpperQuantity,
palletQuantities.BaseQty AS PalletQty
FROM
(
SELECT OBIN.WhsCode + '-' + OBIN.SL1Code + '-' + OBIN.SL2Code + '-' +
OBIN.SL3Code
AS BinPrefix,
OBIN.WhsCode + '-' + OBIN.SL1Code + '-' + OBIN.SL2Code + '-' + OBIN.SL3Code
+ '-' +
@FloorLocation4 AS FloorBinCode2,
OBIN.BinCode, OIBQ.ItemCode, OIBQ.OnHandQty FROM OIBQ, OBIN WHERE
OBIN.BinCode LIKE
@ScanArea AND OBIN.SL4Code <> @FloorLocation4 AND OBIN.AbsEntry =
OIBQ.BinAbs
) binupper
LEFT OUTER JOIN
(
SELECT OBIN.WhsCode + '-' + OBIN.SL1Code + '-' + OBIN.SL2Code + '-' +
OBIN.SL3Code
AS BinPrefix, OBIN.BinCode, OIBQ.ItemCode, OIBQ.OnHandQty
FROM OIBQ, OBIN
WHERE OBIN.BinCode LIKE @ScanArea
AND OBIN.SL4Code = @FloorLocation4
AND OBIN.AbsEntry = OIBQ.BinAbs
) binfloor ON (binfloor.ItemCode = binupper.ItemCode AND binfloor.BinPrefix
=
binupper.binprefix)
JOIN (
SELECT ItemCode, BaseQty FROM UGP1, OITM WHERE UGP1.UgpEntry = OITM.UgpEntry
AND
UGP1.UomEntry = OITM.PUomEntry
) palletQuantities ON (binupper.ItemCode = palletQuantities.ItemCode)
WHERE binfloor.OnHandQty IS NULL OR binfloor.OnHandQty <=
palletQuantities.BaseQty
/2 AND binupper.OnHandQty > 0
ORDER BY UpperBin
SET @LastBin = ''
SET @LastBinFloorQuantity = 0
OPEN curs
FETCH NEXT FROM curs INTO @FloorBin, @ItemCode, @FloorQuantity, @UpperBin,
@UpperQuantity, @PalletQty
WHILE @@FETCH_STATUS = 0
BEGIN
IF @FloorQuantity IS NULL BEGIN SET @FloorQuantity = 0 END
IF @LastBin <> @FloorBin BEGIN
```

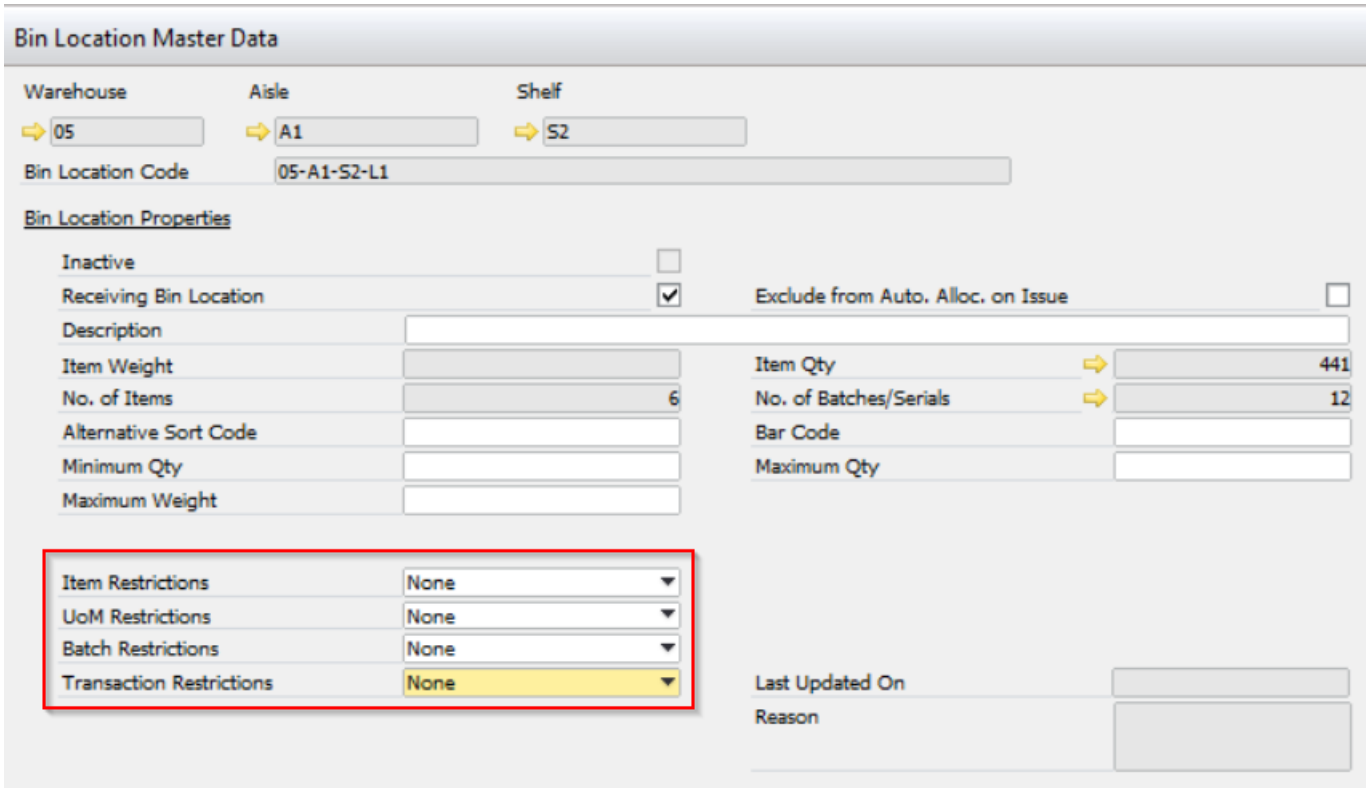
```
SET @LastBin = @FloorBin
SET @QuantityAlready = NULL
SELECT @QuantityAlready = SUM(ToQuantity) FROM
XXX_INVTRANSFER_DRAFT_LINESBIN WHERE ItemCode = @ItemCode AND ToBin =
@FloorBin
IF @QuantityAlready IS NULL BEGIN SET @QuantityAlready = 0 END
SET @LastBinFloorQuantity = @FloorQuantity + @QuantityAlready
END
IF @LastBinFloorQuantity <= @PalletQty / 2 BEGIN
SET @QuantityNeeded = @PalletQty - @LastBinFloorQuantity
SET @QuantityPartial = @QuantityNeeded
IF @QuantityPartial > @UpperQuantity BEGIN SET @QuantityPartial =
@UpperQuantity END
SET @LastBinFloorQuantity = @LastBinFloorQuantity + @QuantityPartial
INSERT INTO @RESULT (ItemCode, Quantity, SourceLocation,
DestinationLocation)
VALUES (@ItemCode, @QuantityPartial, @UpperBin, @FloorBin)
END
FETCH NEXT FROM curs INTO @FloorBin, @ItemCode, @FloorQuantity, @UpperBin,
@UpperQuantity, @PalletQty
END
CLOSE curs
DEALLOCATE curs
SELECT * FROM @RESULT
```

If you are NOT getting the expected *Replenishment Strategy* results, then it is recommend to check on the *Master Data Configuration*, the *Bin Restrictions* settings and make sure the *Batch is released*.

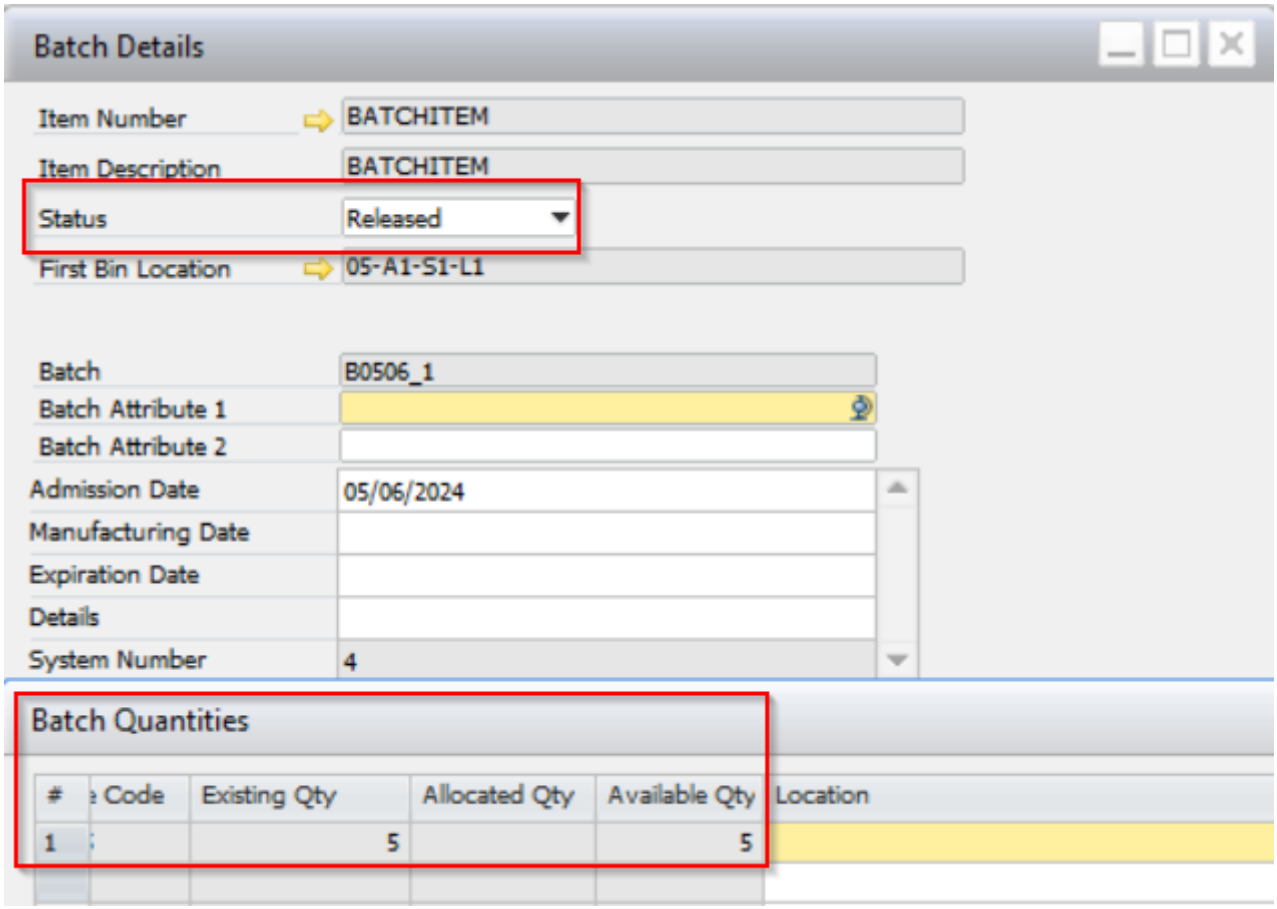
1. Check on your Item Master Data settings:



2. Review possible Bin Restrictions:



3. Make sure the status of the batch is released:



Example Replenishment Strategy HANA

The scan area and the floor location can be set in the calling of the procedure:

CALL “PMXSCAN_REPLENISHMENT” ('01-F-%', '1') ;

```
CREATE PROCEDURE PMXSCAN_REPLENISHMENT (
    IN
    ScanArea NVARCHAR(5000),
    FloorLocation4 NVARCHAR(5000)
)
LANGUAGE SQLSCRIPT
AS
BEGIN
    DECLARE FloorBin nvarchar(5000);
    DECLARE ItemCode nvarchar(5000);
    DECLARE FloorQuantity DECIMAL;
    DECLARE UpperBin nvarchar(5000);
    DECLARE UpperQuantity DECIMAL;
    DECLARE PalletQty DECIMAL;
    DECLARE QuantityPartial DECIMAL;
    DECLARE LastBin nvarchar(5000);
    DECLARE LastBinFloorQuantity DECIMAL;
    DECLARE QuantityNeeded DECIMAL;
    DECLARE QuantityAlready DECIMAL;
    DECLARE CURSOR curs FOR
```

```
    SELECT
        "binupper"."FloorBinCode2" AS "FloorBin",
        "binupper"."ItemCode" AS "ItemCode",
        "binfloor"."OnHandQty" AS "FloorQuantity",
        "binupper"."BinCode" AS "UpperBin",
        "binupper"."OnHandQty" AS "UpperQuantity",
        "palletQuantities"."BaseQty" AS "PalletQty"
    FROM
        (
        SELECT
            "OBIN"."WhsCode" || '-' || "OBIN"."SL1Code" || '-' ||
"OBIN"."SL2Code" || '-' || "OBIN"."SL3Code" AS "BinPrefix",
            "OBIN"."WhsCode" || '-' || "OBIN"."SL1Code" || '-' ||
"OBIN"."SL2Code" || '-' || "OBIN"."SL3Code" || '-' || FloorLocation4 AS
"FloorBinCode2",
            "OBIN"."BinCode", "OIBQ"."ItemCode", "OIBQ"."OnHandQty"
        FROM "OIBQ", "OBIN"
        WHERE "OBIN"."BinCode" LIKE ScanArea AND "OBIN"."SL4Code" <>
FloorLocation4 AND "OBIN"."AbsEntry" = "OIBQ"."BinAbs"
        ) "binupper"
        LEFT OUTER JOIN
        (
        SELECT
            "OBIN"."WhsCode" || '-' || "OBIN"."SL1Code" || '-' ||
"OBIN"."SL2Code" || '-' || "OBIN"."SL3Code" AS "BinPrefix",
            "OBIN"."BinCode", "OIBQ"."ItemCode", "OIBQ"."OnHandQty"
        FROM "OIBQ", "OBIN"
        WHERE "OBIN"."BinCode" LIKE ScanArea
            AND "OBIN"."SL4Code" = FloorLocation4
            AND "OBIN"."AbsEntry" = "OIBQ"."BinAbs"
        ) "binfloor"
        ON ("binfloor"."ItemCode" = "binupper"."ItemCode" AND
"binfloor"."BinPrefix" = "binupper"."BinPrefix")
        JOIN (
        SELECT "ItemCode", "BaseQty" FROM "UGP1", "OITM" WHERE
"UGP1"."UgpEntry" = "OITM"."UgpEntry"
        /*AND "UGP1"."UomEntry" = "OITM"."PUomEntry"*/
        ) "palletQuantities"
        ON ("binupper"."ItemCode" = "palletQuantities"."ItemCode")
    WHERE
        "binfloor"."OnHandQty" IS NULL OR "binfloor"."OnHandQty" <=
"palletQuantities"."BaseQty" AND "binupper"."OnHandQty" > 0
    ORDER BY "UpperBin";

    LastBin := '';
    LastBinFloorQuantity := 0;

    CREATE LOCAL TEMPORARY TABLE #result
    (
        "ItemCode" NVARCHAR(200),
        "BatchNumber" NVARCHAR(200),
```

```
        "SerialNumber" NVARCHAR(200),
        "Quantity" DECIMAL(21,6),
        "SourceLocation" NVARCHAR(200),
        "DestinationLocation" NVARCHAR(200),
        "GroupID" NVARCHAR(200),
        "Remarks" NVARCHAR(200)
    );

    FOR c_ AS curs DO
        FloorBin :=c_."FloorBin";
        ItemCode :=c_."ItemCode";
        FloorQuantity :=c_."FloorQuantity";
        UpperBin :=c_."UpperBin";
        UpperQuantity :=c_."UpperQuantity";
        PalletQty :=c_."PalletQty";


        IF FloorQuantity IS NULL THEN
            FloorQuantity := 0;
        END IF;
        IF LastBin <> FloorBin THEN
            LastBin := FloorBin;
            QuantityAlready := NULL;

            SELECT SUM("ToQuantity") INTO QuantityAlready FROM
            "XXX_INVTRANSFER_DRAFT_LINESBIN" WHERE "ItemCode" = ItemCode AND
"ToBin" = FloorBin;

            IF QuantityAlready IS NULL THEN
                QuantityAlready := 0;
            END IF;
            LastBinFloorQuantity := FloorQuantity + QuantityAlready;
        END IF;
        IF LastBinFloorQuantity <= PalletQty / 2 THEN
            QuantityNeeded := PalletQty - LastBinFloorQuantity;
            QuantityPartial := QuantityNeeded;
            IF QuantityPartial > UpperQuantity THEN
                QuantityPartial := UpperQuantity;
            END IF;
            LastBinFloorQuantity := LastBinFloorQuantity + QuantityPartial;

            INSERT INTO #result ("ItemCode", "Quantity", "SourceLocation",
"DestinationLocation")
            VALUES (ItemCode, QuantityPartial, UpperBin, FloorBin);
        END IF;

    END FOR;

    SELECT * FROM #result;
    DROP TABLE #result;
```

```
END;
```