Reset On-Order for Parts with Non-Zero Quantities

01/22/2026 2:51 pm EST

00152749

This is a generic script to reset Incorrect parts On Order quantity and out of repair quantity:

[Parts on order anomaly - post update]

```
/*============================================================================

Script Name : Reset On-Order for Parts with Non-Zero Quantities

Created On : 2026-01-21

Created By : Emeka D

Purpose  :

 1) Identify parts where IN_Inventory.On_Order_Quantity <> 0 OR Out_For_Repair_Qty <> 0

 2) Call Reset_On_Order once per Part_Code (WHILE loop, no cursor)

 3) Show ONLY parts whose On_Order_Quantity and/or Out_For_Repair_Qty changed after reset


Notes:

 - This script assumes Reset_On_Order updates inventory quantities by Part_Code.

 - Review the final output to confirm changes before/after.

============================================================================*/

SET NOCOUNT ON;

SET XACT_ABORT ON;

------------------------------------------------------------------------------

-- 1) Stage candidate parts + capture BEFORE snapshot
```

```
--------------------------------------------------------------------------------

IF OBJECT_ID('tempdb..#Parts') IS NOT NULL DROP TABLE #Parts;

IF OBJECT_ID('tempdb..#InvBefore') IS NOT NULL DROP TABLE #InvBefore;

IF OBJECT_ID('tempdb..#InvAfter') IS NOT NULL DROP TABLE #InvAfter;

CREATE TABLE #Parts

(

 Part_Code VARCHAR(50) NOT NULL,

 Processed BIT    NOT NULL CONSTRAINT DF_Parts_Processed DEFAULT (0),

 CONSTRAINT PK_Parts PRIMARY KEY CLUSTERED (Part_Code)

);

INSERT INTO #Parts (Part_Code)

SELECT DISTINCT

 p.Part_Code

FROM IN_Inventory i

JOIN IN_Part p

 ON p.Part_Id = i.Part_Id

WHERE ISNULL(i.On_Order_Quantity, 0) <> 0

 OR ISNULL(i.Out_For_Repair_Qty, 0) <> 0;

IF NOT EXISTS (SELECT 1 FROM #Parts)

BEGIN

 SELECT Message = 'No parts found with non-zero On_Order_Quantity or Out_For_Repair_Qty.';

 RETURN;

END;

SELECT

 p.Part_Code,

 i.Inventory_Id,
```

```
 On_Order_Quantity   = ISNULL(i.On_Order_Quantity, 0),
 Out_For_Repair_Qty  = ISNULL(i.Out_For_Repair_Qty, 0)
INTO #InvBefore
FROM #Parts p
JOIN IN_Part ip
 ON ip.Part_Code = p.Part_Code
JOIN IN_Inventory i
 ON i.Part_Id = ip.Part_Id;
--------------------------------------------------------------------------------
-- 2) Execute Reset_On_Order per Part_Code (WHILE loop)
--------------------------------------------------------------------------------
DECLARE @Part_Code VARCHAR(50);
WHILE EXISTS (SELECT 1 FROM #Parts WHERE Processed = 0)
BEGIN
 SELECT TOP (1)
  @Part_Code = Part_Code
 FROM #Parts
 WHERE Processed = 0
 ORDER BY Part_Code;
 IF @Part_Code IS NULL BREAK;
 EXEC Reset_On_Order @part_code=@Part_Code;
 UPDATE #Parts
 SET Processed = 1
 WHERE Part_Code = @Part_Code;
END;
--------------------------------------------------------------------------------
```

```
-- 3) Capture AFTER snapshot
-------------------------------------------------------------------------------
SELECT
 p.Part_Code,
 i.Inventory_Id,
 On_Order_Quantity   = ISNULL(i.On_Order_Quantity, 0),
 Out_For_Repair_Qty  = ISNULL(i.Out_For_Repair_Qty, 0)
INTO #InvAfter
FROM #Parts p
JOIN IN_Part ip
 ON ip.Part_Code = p.Part_Code
JOIN IN_Inventory i
 ON i.Part_Id = ip.Part_Id;
-------------------------------------------------------------------------------
-- 4) Show ONLY rows where quantities changed
-------------------------------------------------------------------------------
SELECT
 a.Part_Code,
 a.Inventory_Id,
 Before_On_Order_Quantity = b.On_Order_Quantity,
 After_On_Order_Quantity  = a.On_Order_Quantity,
 Before_Out_For_Repair_Qty = b.Out_For_Repair_Qty,
 After_Out_For_Repair_Qty = a.Out_For_Repair_Qty
FROM #InvAfter a
JOIN #InvBefore b
 ON b.Part_Code = a.Part_Code
```

```
 AND b.Inventory_Id = a.Inventory_Id
WHERE ISNULL(b.On_Order_Quantity, 0) <> ISNULL(a.On_Order_Quantity, 0)
 OR ISNULL(b.Out_For_Repair_Qty, 0) <> ISNULL(a.Out_For_Repair_Qty, 0)
ORDER BY a.Part_Code, a.Inventory_Id;
```