Calculate the Running Total

```
/*
I have a table like this.

Date            Item            BuyItem
-------------------------------------------
20150101        Mouse            10
20150101        Keyboard        100
20150202        Mouse            20
20150202        Keyboard        200

I want to query like this.

Date            Item            RunningTotal
-------------------------------------------------
20150101        Mouse            10
20150202        Mouse            30
20150101        Keyboard        100
20150202        Keyboard        300

Try using CROSS APPLY (BEST WAY) or Correlated sub-query.
*/
;WITH cte AS
(
SELECT
 * FROM
 (VALUES (20150101,'Mouse',10),
             (20150101,'Keyboard',100),
             (20150202,'Mouse',20),
             (20150202,'Keyboard',200) )tc([Date], Item, BuyItem)
)

SELECT *
FROM   cte a
     CROSS APPLY(SELECT SUM(BuyItem) AS running_total
             FROM   cte b
             WHERE  a.Item = b.Item
                  AND a.[Date] >= b.[Date]) cs
/*
Result:
Date       Item           BuyItem   Running_Total
---------------------------------------------------------
20150101   Mouse         10            10
20150202   Mouse         20            30
20150101   Keyboard    100           100
20150202   Keyboard    200           300
```

```
Recursive CTE method using ROW_NUMBER function and PARTITION:
*/
;WITH cte
   AS (SELECT ROW_NUMBER()OVER(PARTITION BY Item
              ORDER BY [date] ) AS rn,*
      FROM   (VALUES (20150101,'Mouse',10),
                     (20150101,'Keyboard',100),
                     (20150202,'Mouse',20),
                     (20150202,'Keyboard',200) )tc([Date], Item, BuyItem)),
   CTE_RunningTotal
   AS (SELECT [Date],Item,BuyItem,BuyItem AS running_total,rn
      FROM   cte
      WHERE  rn = 1
      UNION ALL
      SELECT T.[Date],T.Item,t.BuyItem,
          T.BuyItem + C.running_total AS running_total,
          t.rn
      FROM   CTE_RunningTotal AS C
          INNER JOIN cte AS T
                 ON T.Item = c.Item
                   AND t.rn = C.rn + 1)
SELECT [Date],
     Item,
     BuyItem,
     running_total
FROM   CTE_RunningTotal AS C

/*
Note: Better to update your server to 2012 which can use
sum() over(order by) method to calculate running total
which much faster than these methods
*/

--Using CROSS APPLY:

SELECT [Date],
     item,
     running_total
FROM   #yourtable a
     CROSS APPLY(SELECT SUM(BuyItem) AS running_total
              FROM   #yourtable b
              WHERE  a.Item = b.Item
                  AND a.[Date] >= b.[Date]) ca
ORDER  BY BuyItem
```

```
--Make a use of windowing function. in SQL 2012+

DECLARE @Items TABLE
(
DATE NVARCHAR(MAX),
Item NVARCHAR(MAX),
BuyItem INT
)

INSERT INTO @Items([DATE], Item, BuyItem) VALUES('20150101', 'Mouse', 10)
INSERT INTO @Items([DATE], Item, BuyItem) VALUES('20150101', 'Keyboard', 100)
INSERT INTO @Items([DATE], Item, BuyItem) VALUES('20150202', 'Mouse', 20)
INSERT INTO @Items([DATE], Item, BuyItem) VALUES('20150202', 'Keyboard', 200)

SELECT [DATE], Item, SUM(BuyItem) OVER (PARTITION BY Item ORDER BY BuyItem) AS RunningTotal FROM
@Items ORDER BY Item DESC

/* Result:
Date       Item              Running_Total
---------------------------------------------------
20150101   Mouse              10
20150202   Mouse              30
20150101   Keyboard          100
20150202   Keyboard          300



In aggregate function, with running total, using Window functions
are very good performance In SQL 2012+:
*/


SELECT *,
  SUM() OVER(PARTITION BY Item, ORDER BY [Date]) AS RunningTotal
FROM
Your_Table
ORDER BY Item DESC

--Faster than when add window frame:

SELECT *,
  SUM() OVER(PARTITION BY Item, ORDER BY [Date] ROWS UNBOUNDED PRECEDING) AS RunningTotal
FROM
Your_Table
ORDER BY Item DESC
```