The results are in! See what nearly 90,000 developers picked as their most loved, dreaded, and desired coding languages and more in the 2019 Developer Survey.

## LINQ - Summing numbers stored as string

Ask Question



I have the following data:

| PK | OrderNumber | USERDEFFIELD |
|----|-------------|--------------|
| 1  | 0001        | 10           |
| 2  | 0001        | 25           |
| 3  | 0002        | 20           |
| 4  | 0002        | 22           |
| 5  | 0002        | NULL         |
|    |             |              |

0003 ABC123

The UserDefField column is of VARCHAR type in the database. Using LINQ, how can I get the SUM(UserDefField) per order? NULL and non-numeric values for UserDefField are to be considered as zero. The result I'm trying to get:

| OrderNumber | <b>TotalQty</b> |
|-------------|-----------------|
| 0001        | 35              |
| 0002        | 42              |
| 0003        | 0               |

If UserDefField is strictly nullable numeric field I know I would do this inside a foreach loop:

```
TotalQtyForThisOrder = orders.Sum(w => w.UserDefField ?? 0 );
```

But for a string field, what should be done? Thank you very much for the help.



Home

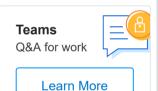
**PUBLIC** 

## Stack Overflow

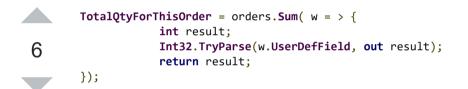
Tags

Users

Jobs



## 3 Answers





edited Jan 22 '14 at 12:45



Dionysos

**133** 1 2 14

answered Jan 18 '12 at 21:01



LiOliC

**10.3k** 26 50

1 @DavidHoerster: not true. You can initialize it, there's just no point doing so. – Igby Largeman Jan 18 '12 at 21:10 ✓

sorry - I was mistaken. Perhaps I need more sleep (or more caffeine). – David Hoerster Jan 18 '12 at 21:17



I would use the Aggregate function like so:

1

```
var result = orders.Aggregate(new Dictionary<OrderIdType, int>(), (a)
{
   int quantity;
   if (!int.TryParse(item.USERDEFFIELD, out quantity))
        quantity = 0;
   if (!accumulator.ContainsKey(item.OrderId))
        accumulator[item.OrderId] = quantity;
   else
        accumulator[item.OrderId] += quantity;
   return accumulator;
});
```

answered Jan 18 '12 at 21:16



Charles Lambert

**3,885** 18 42

updated, I forgot int.TryParse - Charles Lambert Jan 18 '12 at 21:18



Fundamentally, I'd say your schema is off: if you're treating a textual value as a number if it happens to be parsable that way, you're really fighting against sensible design.



I don't know how you'd be able to write a LINQ query to make that sum occur in the database. You'd probably want to write some custom conversion function, or perhaps a custom view which did the "try to parse it and return 0 otherwise" operation.

You can do it easily enough within LINQ to Objects:

```
x = orders.Select(y => y.UserDefField) // To just bring down the rigi
    .AsEnumerable()
    .Sum(text => {
         int result;
         // We don't care about the return value... and
```

```
// it even handles null input. (In any error cond
// "result" will be 0 afterwards.)
int.TryParse(text, out result);
return result;
});
```

... but I'd *really* recommend that you revisit your schema if you possibly can.

edited Jan 18 '12 at 21:12

answered Jan 18 '12 at 21:03



Jon Skeet 1099k 698 8006 8482

1 Thank you @Jon Skeet. I'd be the first to admit that the schema is terrible. But that's the card I've been dealt and expected to play with. It is a 11+ year old design and is not expected to change anytime soon. I will try your approach. — FMFF Jan 18 '12 at 21:10