Perform JOIN in order to get CalendarYear value:

Query:

```
109 SELECT
110     q.CalendarYear,
111     SUM(UnitCost) AS TotalUnitCost,
112     f.Column1,
113     f.Column2,
114     f.Column3,
115     f.Column4
116 FROM
117     dbo.FactProductInventory_DST AS f
118     INNER JOIN v_CalendarQuarters AS q ON q.QuarterRange = f.QuarterRange
119 GROUP BY
120     q.CalendarYear,
121     f.Column1,
122     f.Column2,
123     f.Column3,
124     f.Column4;
```

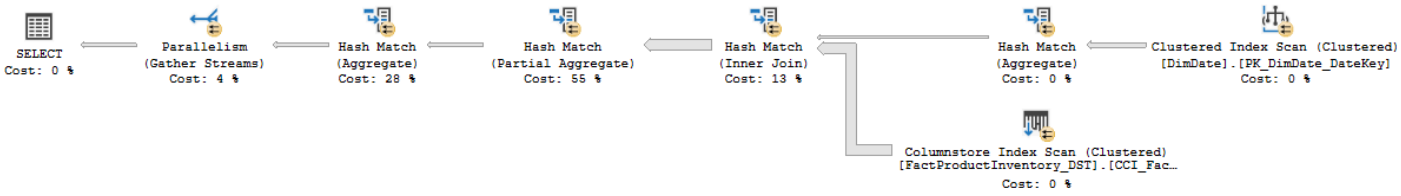
Statistics: Time + IO:

Totals:

Table	Scan Count	Logical Reads	Physical Reads	Read-Ahead Reads	LOB Logical Reads	LOB Physical Reads	LOB Read-Ahead Reads	% Logical Reads of Total Reads
DimDate	5	340	0	0	0	0	0	100.000
FactProductInventory_DST	4	0	0	0	19,800	0	0	0.000
Worktable	0	0	0	0	0	0	0	0.000
Total	**9**	**340**	**0**	**0**	**19,800**	**0**	**0**	

	CPU	Elapsed
SQL Server parse and compile time:	00:00:00.000	00:00:00.000
SQL Server Execution Times:	00:00:43.282	00:00:11.576
Total	**00:00:43.282**	**00:00:11.576**

Actual Execution Plan:



Use LEFT() function in order to get CalendarYear

Query:

```sql
SELECT
    LEFT(f.QuarterRange, 4) AS CalendarYear,
    SUM(UnitCost) AS TotalUnitCost,
    f.Column1,
    f.Column2,
    f.Column3,
    f.Column4
FROM
    dbo.FactProductInventory_DST AS f
GROUP BY
    LEFT(f.QuarterRange, 4),
    f.Column1,
    f.Column2,
    f.Column3,
    f.Column4;
```

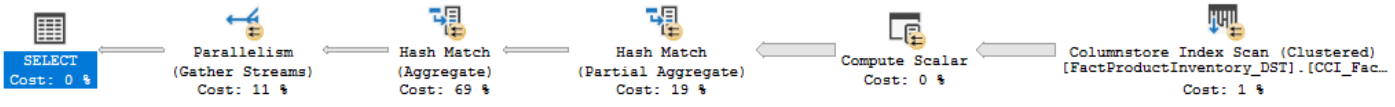
Statistics: Time + IO:

Totals:

Table	Scan Count	Logical Reads	Physical Reads	Read-Ahead Reads	LOB Logical Reads	LOB Physical Reads	LOB Read-Ahead Reads	% Logical Reads of Total Reads
FactProductInventory_DST	4	0	0	0	19,800	0	0	NaN
Worktable	0	0	0	0	0	0	0	NaN
Total	**4**	**0**	**0**	**0**	**19,800**	**0**	**0**	

	CPU	Elapsed
SQL Server parse and compile time:	00:00:00.000	00:00:00.000
SQL Server Execution Times:	00:00:55.282	00:00:14.221
Total	**00:00:55.282**	**00:00:14.221**

Actual Execution Plan:



Summary:

Source table has the following definition:

```
DROP TABLE IF EXISTS dbo.FactProductInventory_DST;
GO

CREATE TABLE dbo.FactProductInventory_DST
(
    ProductKey INT NOT NULL,
    DateKey INT NOT NULL,
    MovementDate DATE NOT NULL,
    UnitCost MONEY NOT NULL,
    UnitsIn INT NOT NULL,
    UnitsOut INT NOT NULL,
    UnitsBalance INT NOT NULL,
    QuarterRange VARCHAR(16) NOT NULL,
    Column1 VARCHAR(64) NULL,
    Column2 VARCHAR(64) NULL,
    Column3 VARCHAR(64) NULL,
    Column4 VARCHAR(64) NULL
);

CREATE CLUSTERED COLUMNSTORE INDEX CCI_FactProductInventory ON dbo.FactProductInventory_DST;
```

And is populated by 77 500 000 records.

The JOIN version has a lower execution and CPU time than LEFT() function. The difference in CPU time can be even more significant when we work with a larger data set.