



QUERY TUNING SUPER-STARDOM ***for Beginners***

TIM FORD, SQL SERVER MVP

SQLAGENTMAN CONSULTANCY - & - SQL CRUISE EVENTS



EMAIL : tim@thesqlagentman.com

TWITER : [@sqlagentman](https://twitter.com/sqlagentman) -&- [@sqlcruise](https://twitter.com/sqlcruise)

CV : [linkedin.com/in/timothyford](https://www.linkedin.com/in/timothyford)



SQL Cruise



SQL Agent Man Consultancy
Beside you from start to finish.



Microsoft SQL Server MVP
Since 2009



theSQLAgentMan.com



Director, PASS SQLSaturday



Author

RE:

AGENDA

SELECT

TMI – TOO MUCH INFORMATION

FROM

ALL THE JOINS

WHERE

SARGABILITY

DISTINCT

EXPENSIVE, SLOPPY, LAZY

SCHEMA

FULLY-QUALIFIED OBJECT NAMES

SELECT : TMI – TOO MUCH INFORMATION

SELECT * <> SELECT SUPERSTAR

SELECT : TMI – TOO MUCH INFORMATION

WRONG

```
SELECT *  
FROM dbo.wait_history_int  
WHERE wait_id = 3298  
      AND running_pct < 80  
ORDER BY running_pct ASC  
OPTION (RECOMPILE);
```

SQL Server Execution Times:

CPU time = 0 ms, elapsed time = 109 ms.

SQL Server parse and compile time:

CPU time = 0 ms, elapsed time = 1 ms.

RIGHT

```
SELECT  date_stamp  
        , wait_type  
        , resource_wait_time_s  
        , avg_resource_wait_time_ms  
        , pct  
        , running_pct  
FROM    dbo.wait_history_int  
WHERE   wait_id = 3298  
      AND running_pct < 80  
ORDER BY running_pct ASC  
OPTION (RECOMPILE);
```

SQL Server Execution Times:

CPU time = 0 ms, elapsed time = 46 ms.

SQL Server parse and compile time:

CPU time = 0 ms, elapsed time = 0 ms.

FROM : ALL THE JOINS

Spot The Issue...

```
SELECT HEAD.SalesOrderID
      , CUST.PersonID AS CustomerID
      , PERS.LastName + ', ' + PERS.FirstName AS salesperson
      , DET.OrderQty
      , PRO.ProductNumber
      , DET.LineTotal
FROM sales.SalesOrderHeader HEAD
      INNER JOIN sales.SalesOrderDetail DET ON HEAD.SalesOrderID = DET.SalesOrderID
      LEFT JOIN Sales.Customer CUST ON HEAD.CustomerID = CUST.CustomerID
      LEFT JOIN Sales.SalesPerson SP ON HEAD.SalesPersonID = SP.BusinessEntityID
-- LEFT JOIN Sales.SalesTerritory ST ON SP.TerritoryID = ST.TerritoryID
      LEFT JOIN Person.Person PERS ON SP.BusinessEntityID = PERS.BusinessEntityID
      LEFT JOIN Production.Product PRO ON DET.ProductID = PRO.ProductID
WHERE HEAD.CustomerID = 29825
OPTION (RECOMPILE);
```


FROM : ALL THE JOINS

SQL Server Execution Times:

CPU time = 0 ms, elapsed time = 109 ms.

SQL Server parse and compile time:

CPU time = 0 ms, elapsed time = 1 ms.

SQL Server Execution Times:

CPU time = 0 ms, elapsed time = 46 ms.

SQL Server parse and compile time:

CPU time = 0 ms, elapsed time = 0 ms.

WHERE : **SARGA**

SARGABILITY!



WHERE : SARGABILITY – SEEKS YA'LL

**Columns Must Stand Alone
Searching the Phone Book**

WHERE : SARGABLE OPERATIONS

=

>

<

>=

<=

BETWEEN

LIKE without leading %

EXISTS

WHERE : SARGABLE-*ISH* OPERATIONS

<>

IN

OR

NOT IN

NOT EXISTS

NOT LIKE

WHERE : NON-SARGABLE OPERATIONS

LIKE with leading %

WHERE : SARGABILITY - LIKE

WRONG

```
SELECT DFH.[name]
FROM [Superstardom_Predicates].[dbo].[Database_Files_History] DFH
WHERE DFH.[name] LIKE '%Aegis%'
OPTION (RECOMPILE);
```

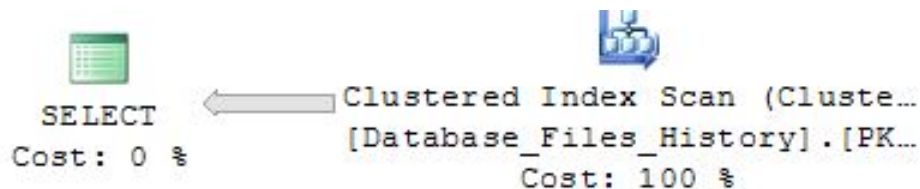
```
SELECT DFH.[name]
FROM [Superstardom_Predicates].[dbo].[Database_Files_History] DFH
WHERE LEFT(DFH.[name],5) = 'Aegis'
OPTION (RECOMPILE);
```

RIGHT

```
SELECT DFH.[name]
FROM [Superstardom_Predicates].[dbo].[Database_Files_History] DFH
WHERE DFH.[name] LIKE 'Aegis%'
OPTION (RECOMPILE);
```

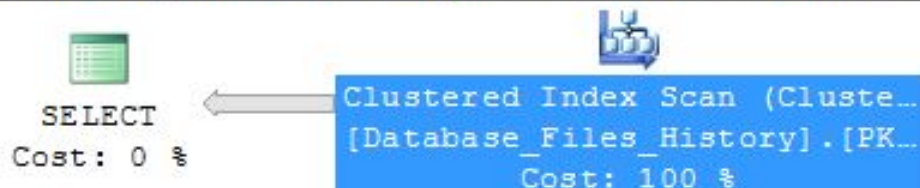

WHERE : SARGABILITY - LIKE

WRONG



RIGHT

Query 2: Query cost (relative to the b
SELECT DFH.DatabaseName FROM [Supersta
Missing Index (Impact 80.95): CREATE N



WHERE : SARGABILITY - DATES

WRONG

```
SELECT DFH.DatabaseName
FROM [Superstardom_Predicates].[dbo].[Database_Files_History] DFH
WHERE YEAR(DFH.date_stamp) = 2013
OPTION (RECOMPILE);
```

RIGHT

```
SELECT DFH.DatabaseName
FROM [Superstardom_Predicates].[dbo].[Database_Files_History] DFH
WHERE DFH.date_stamp >= '1/1/2013' AND DFH.date_stamp < '1/1/2014'
OPTION (RECOMPILE);
```


WHERE : SARGABILITY - DATES

WRONG

```
(1289948 row(s) affected)
Table 'Database_Files_History'. Scan count 1, logical reads 37266,
```

```
(1 row(s) affected)
```

```
SQL Server Execution Times:
```

```
  CPU time = 593 ms,  elapsed time = 6737 ms.
```

```
SQL Server parse and compile time:
```

```
  CPU time = 0 ms,  elapsed time = 1 ms.
```

```
(1289948 row(s) affected)
```

```
Table 'Database_Files_History'. Scan count 1, logical reads 37266,
```

```
(1 row(s) affected)
```

```
SQL Server Execution Times:
```

```
  CPU time = 390 ms,  elapsed time = 6896 ms.
```

```
SQL Server parse and compile time:
```

```
  CPU time = 0 ms,  elapsed time = 0 ms.
```

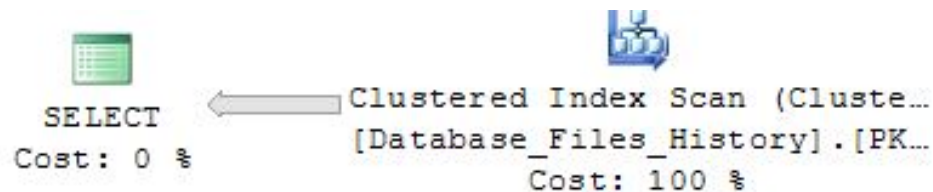
```
SQL Server Execution Times:
```

```
  CPU time = 0 ms,  elapsed time = 0 ms.
```

RIGHT

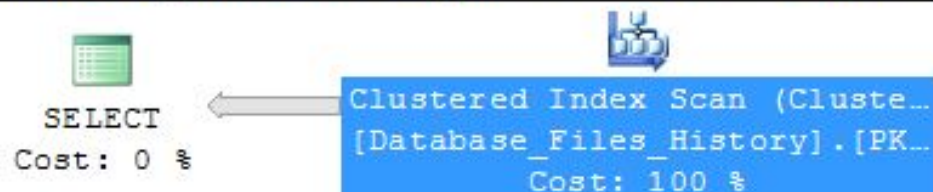
WHERE : SARGABILITY - DATES

WRONG



RIGHT

Query 2: Query cost (relative to the b...
SELECT DFH.DatabaseName FROM [Supersta...
Missing Index (Impact 80.95): CREATE N...



DISTINCT

EXPENSIVE, SLOPPY, LAZY

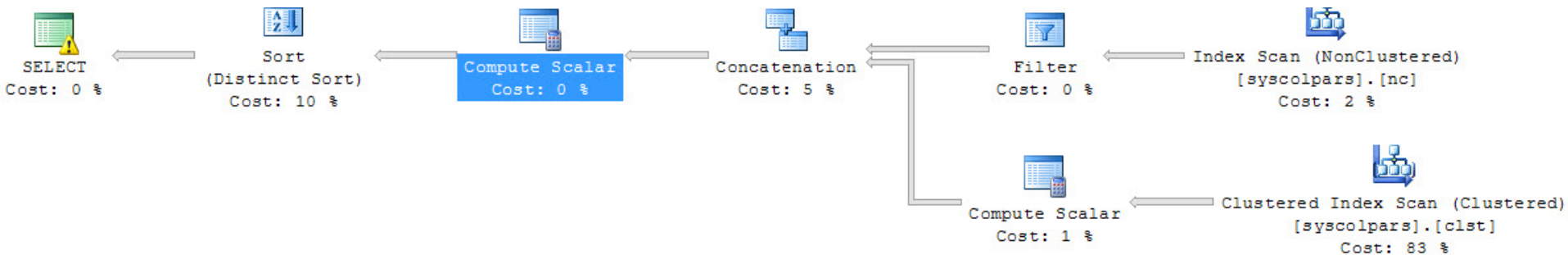
DISTINCT & User-Defined Functions

DISTINCT & 1-to-Many Joins

DISTINCT to Hide Bad/Lazy Joins

DISTINCT DISTINCT & User-Defined Functions

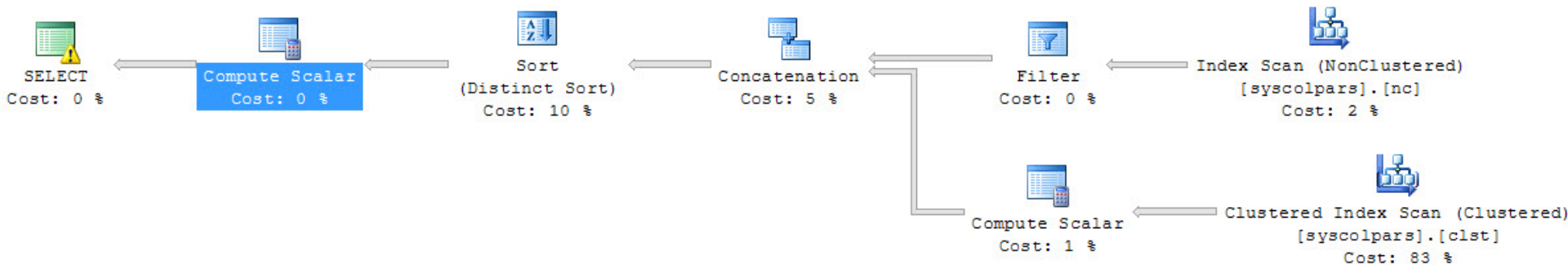
```
SELECT DISTINCT [name] AS [object_name]
      , Superstardom.dbo.fn_reversal([name]) AS [trimmed_name]
FROM sys.all_columns
OPTION (RECOMPILE);
```



WRONG

DISTINCT DISTINCT & User-Defined Functions

```
SELECT A.[name] AS [object_name]
      , Superstardom.dbo.fn_reversal(A.[name]) AS [trimmed_name]
FROM
    (
        SELECT DISTINCT [name]
        FROM sys.all_columns
    ) A
OPTION (RECOMPILE);
```



RIGHT

DISTINCT

DISTINCT & User-Defined Functions

WRONG

(3226 row(s) affected)

SQL Server Execution Times:

CPU time = 171 ms, elapsed time = 330 ms.

RIGHT

(3226 row(s) affected)

SQL Server Execution Times:

CPU time = 78 ms, elapsed time = 149 ms.

DISTINCT

DISTINCT & User-Defined Functions

WRONG

```
SELECT DISTINCT [name] AS [object_name]
      , Superstardom.dbo.fn_reversal([name]) AS [trimmed_name]
FROM sys.all_columns
OPTION (RECOMPILE);
```

RIGHT

```
SELECT A.[name] AS [object_name]
      , Superstardom.dbo.fn_reversal(A.[name]) AS [trimmed_name]
FROM
    (
        SELECT DISTINCT [name]
        FROM sys.all_columns
    ) A
OPTION (RECOMPILE);
```


DISTINCT DISTINCT & 1-to-Many Joins

WRONG

```
SELECT DISTINCT H.CustomerID, H.AccountNumber, H.CreditCardApprovalCode
FROM sales.SalesOrderHeader H
      INNER JOIN sales.SalesOrderDetail D
            ON H.SalesOrderID = D.SalesOrderID
OPTION (RECOMPILE);
```

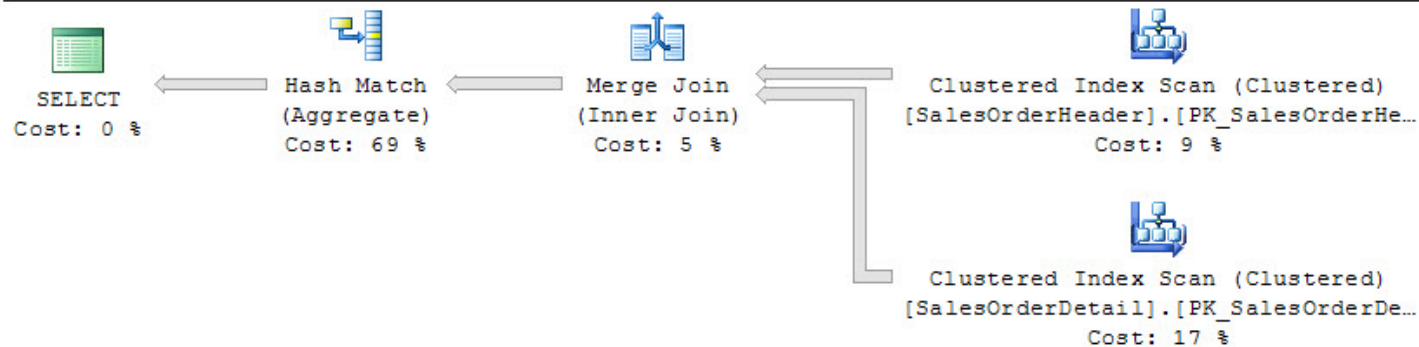
RIGHT

```
SELECT DISTINCT H.CustomerID, H.AccountNumber, H.CreditCardApprovalCode
FROM sales.SalesOrderHeader H
WHERE EXISTS
      (
        SELECT 1
        FROM sales.SalesOrderDetail D
        WHERE H.SalesOrderID = D.SalesOrderID
      )
OPTION (RECOMPILE);
```


DISTINCT DISTINCT & 1-to-Many Joins

Query 1: Query cost (relative to the batch): 68%

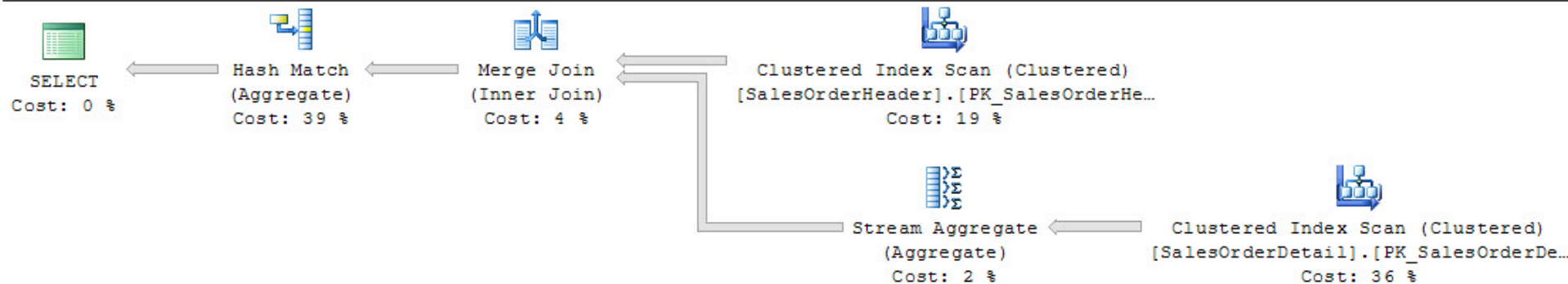
SELECT DISTINCT H.CustomerID, H.AccountNumber, H.CreditCardApprovalCode FROM sales.SalesOrderHeader H INNER JOIN sales.SalesOrderDetail



WRONG

Query 2: Query cost (relative to the batch): 32%

SELECT DISTINCT H.CustomerID, H.AccountNumber, H.CreditCardApprovalCode FROM sales.SalesOrderHeader H WHERE EXISTS (SELECT 1 FROM sa



RIGHT

DISTINCT DISTINCT & 1-to-Many Joins

WRONG

```
SELECT DISTINCT H.CustomerID, H.AccountNumber, H.CreditCardApprovalCode
FROM sales
INNER JOIN
OPTION (RECOMPILE);
```

SQL Server Execution Times:
CPU time = 125 ms, elapsed time = 387 ms.

SQL Server parse and compile time:
CPU time = 8 ms, elapsed time = 8 ms.

RIGHT

```
SELECT DIS
FROM sales
WHERE EXISTS
(
    SE
    FROM sales.SalesOrderDetail D
    WHERE H.SalesOrderID = D.SalesOrderID
)
OPTION (RECOMPILE);
```

SQL Server Execution Times:
CPU time = 63 ms, elapsed time = 271 ms.

SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 0 ms.

reditCardApprovalCode

DISTINCT

DISTINCT To Hide Bad/Lazy Joins

WRONG

```
SELECT DISTINCT
    H.SalesOrderID
FROM AdventureWorks2012.Sales.SalesOrderHeader H
    INNER JOIN AdventureWorks2012.Sales.SalesOrderDetail D
        ON H.SalesOrderID = D.SalesOrderID
OPTION (RECOMPILE);
```

RIGHT

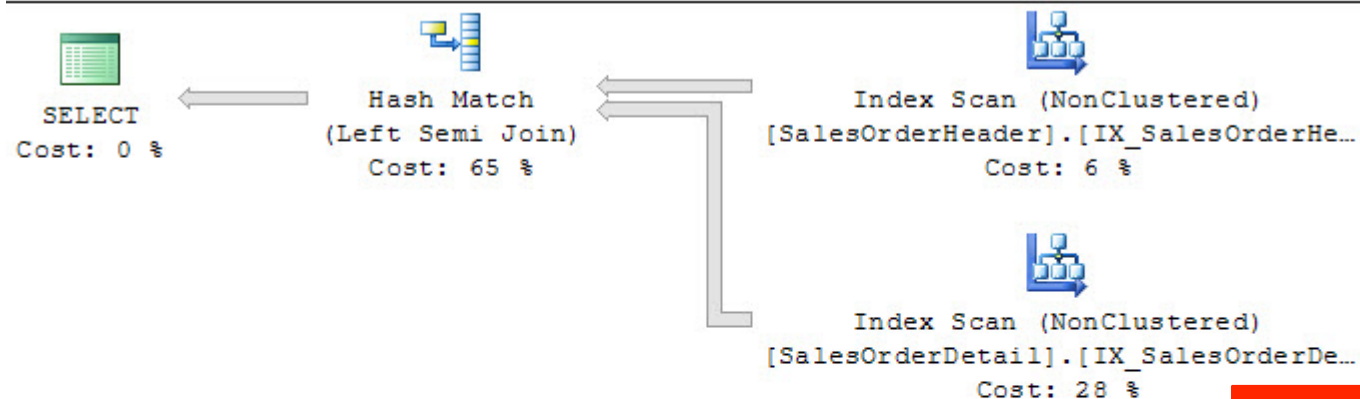
```
SELECT H.SalesOrderID
FROM AdventureWorks2012.Sales.SalesOrderHeader H
OPTION (RECOMPILE);
```


DISTINCT

DISTINCT To Hide Bad/Lazy Joins

Query 1: Query cost (relative to the batch): 94%

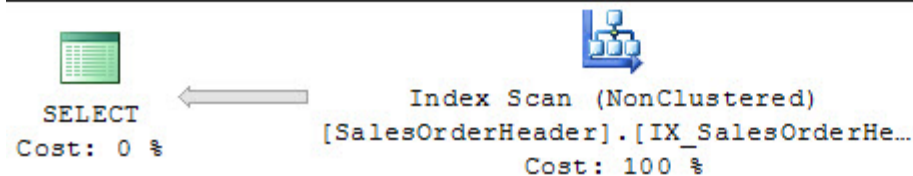
```
SELECT DISTINCT H.SalesOrderID FROM AdventureWorks2012.Sales.SalesOrderHeader H INNER JOIN I
```



WRONG

Query 2: Query cost (relative to the batch): 6%

```
SELECT H.SalesOrderID FROM AdventureWorks2012.Sales.SalesOrderHeader H OPTION(RECOMPILE);
```



RIGHT

DISTINCT DISTINCT To Hide Bad/Lazy Joins

(31465 row(s) affected)

Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

Table 'SalesOrderDetail'. Scan count 1, logical reads 339, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

Table 'SalesOrderHeader'. Scan count 1, logical reads 70, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:

CPU time = 47 ms, elapsed time = 241 ms.

SQL Server parse and compile time:

CPU time = 0 ms, elapsed time = 0 ms.

WRONG

(31465 row(s) affected)

Table 'SalesOrderHeader'. Scan count 1, logical reads 70, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

SQL Server Execution Times:

CPU time = 15 ms, elapsed time = 138 ms.

SQL Server parse and compile time:

CPU time = 0 ms, elapsed time = 0 ms.

RIGHT

SCHEMA

FULLY-QUALIFIED OBJECT NAMES

```
INSERT INTO dbo.sql_cruise_events(id, event_name, location, date_start, date_end)
VALUES (1, 'SQL Cruise', 'Caribbean', '8/10/2010', '8/14/2010')
, (2, 'SQL Cruise Caribbean 2011', 'Caribbean', '3/5/2011', '3/12/2011')
, (3, 'SQL Cruise Alaska 2011', 'Alaska', '5/25/2011', '6/1/2011')
, (4, 'SQL Cruise Caribbean 2012', 'Caribbean', '1/27/2012', '2/3/2012')
, (5, 'SQL Cruise Alaska 2012', 'Alaska', '5/25/2012', '6/1/2012')
, (6, 'SQL Cruise Caribbean 2013', 'Caribbean', '1/28/2013', '2/4/2013')
, (7, 'SQL Cruise Alaska 2013', 'Alaska', '5/27/2013', '6/2/2013')
, (8, 'SQL Cruise 2014', 'Caribbean', '1/25/2014', '2/2/2014')
, (9, 'SQL Cruise Caribbean 2015', 'Caribbean', '2/7/2015', '2/14/2015')
, (10, 'SQL Cruise Mediterranean 2015', 'Mediterranean', '6/14/2015', '6/21/2015');
```

GO

```
INSERT INTO amy.sql_cruise_events(id, event_name, location, date_start, date_end)
VALUES (1, 'SQL Cruise', 'Caribbean', '8/10/2010', '8/14/2010')
, (2, 'SQL Cruise Alaska 2011', 'Alaska', '5/25/2011', '6/1/2011')
, (3, 'SQL Cruise Caribbean 2012', 'Caribbean', '1/27/2012', '2/3/2012')
, (4, 'SQL Cruise Alaska 2012', 'Alaska', '5/25/2012', '6/1/2012')
, (5, 'SQL Cruise Caribbean 2013', 'Caribbean', '1/28/2013', '2/4/2013')
, (6, 'SQL Cruise 2014', 'Caribbean', '1/25/2014', '2/2/2014')
, (7, 'SQL Cruise Caribbean 2015', 'Caribbean', '2/7/2015', '2/14/2015')
, (8, 'SQL Cruise Mediterranean 2015', 'Mediterranean', '6/14/2015', '6/21/2015');
```

GO

SCHEMA

FULLY-QUALIFIED OBJECT NAMES

```
USE [Superstardom]  
GO
```

```
CREATE SCHEMA amy;  
GO
```

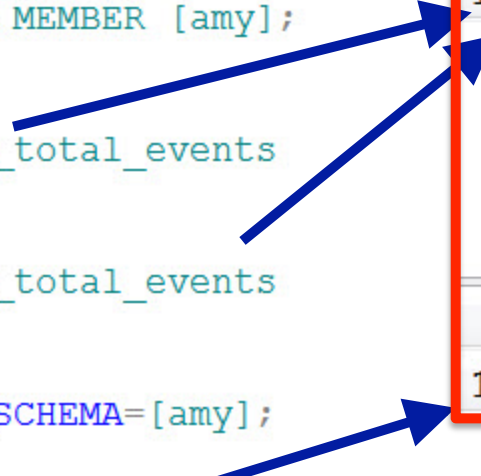
```
CREATE USER [amy] FOR LOGIN [amy]  
ALTER ROLE [db_datareader] ADD MEMBER [amy];  
GO
```

```
SELECT COUNT(id) AS sql_cruise_total_events  
FROM sql_cruise_events;
```

```
SELECT COUNT(id) AS sql_cruise_total_events  
FROM dbo.sql_cruise_events;
```

```
ALTER USER [amy] WITH DEFAULT_SCHEMA=[amy];
```

```
SELECT COUNT(id) AS sql_cruise_total_events  
FROM amy.sql_cruise_events;
```



	sql_cruise_total_events
1	10

	sql_cruise_total_events
1	8

SCHEMA

FULLY-QUALIFIED OBJECT NAMES

```
SELECT eST.[text] AS [QueryText]
      , CP.usecounts
      , CP.objtype
FROM sys.dm_exec_cached_plans AS CP
     CROSS APPLY sys.dm_exec_sql_text(CP.plan_handle) eST
WHERE eST.text LIKE '%AS sql_cruise_total_events%'
OPTION(RECOMPILE);
```

	QueryText	usecounts	objtype
1	SELECT COUNT(id) AS sql_cruise_total_events FROM dbo.sql_cruise_events;	30	Adhoc
2	SELECT COUNT(id) AS sql_cruise_total_events FROM sql_cruise_events;	13	Adhoc
3	SELECT COUNT(id) AS sql_cruise_total_events FROM sql_cruise_events;	10	Adhoc
4	SELECT COUNT(id) AS sql_cruise_total_events FROM sql_cruise_events;	9	Adhoc
5	SELECT COUNT(id) AS sql_cruise_total_events FROM sql_cruise_events;	11	Adhoc





SLIDE DECK :	thesqlagentman.com/go/presentations/
BLOG :	thesqlagentman.com
CONSULTING :	tim@thesqlagentman.com
SQL CRUISE EVENTS :	sqlcruise.com