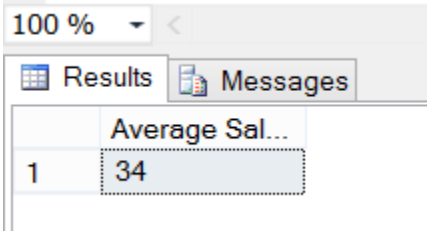
# Some Real World SQL Examples

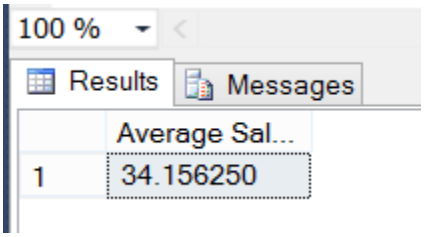
```
/*
  A useful way to check the quality of data is to COUNT 1.
  This example shows us many rows do not have a value in
  the AQSPreference Column.
*/

-- This example doesn't work with our sample database.

/*
   In this example, we JOIN two tables together. We want to check the
   data quality of the AQSPreference column for the records contained
   in the first table. By using COUNT(1), we can get the number
   Of records contained in the first table that have NULL
   in the matching AQSPreference column.
*/

SELECT AQSPreference, COUNT(1)
FROM OutlookProfile e
JOIN CIUserBase u
ON u.PassportID = e.Puid
GROUP BY AQSPreference
ORDER BY AQSPreference
```

| AQSPreference | Count |
|---|---|
| NULL | 203,596,488 |
| 0 | 188,158,220 |
| 1 | 84,101,835 |
| 9 | 8,309,077 |

```
-- Set the database context
USE AdventureWorks2008R2;
/*
CAUTION: In SQL Server, the numeric results of aggregate functions
         Will be in the same data type as the field on which the
         operation was performed. If the field is a whole number
         data type, then the results will be rounded as a whole
         number. To get decimal results, add .0 to the field inside
         the aggregate function and the result will include
         decimals.
*/
/*
Demo the AVG function and return the same data type as the argument
(whole number)
*/
SELECT AVG(OrderQty) AS [Average Sales]
FROM Sales.SalesOrderDetail
WHERE OrderQty BETWEEN 30 AND 50;

/* Query Result
```




```
*/
/*
Demo the AVG function including decimals. Add .0 to the argument to
change it from a whole number to a decimal.
*/
SELECT AVG(OrderQty + .0) AS [Average Sales]
FROM Sales.SalesOrderDetail
WHERE OrderQty BETWEEN 30 AND 50;
/* Query Result
```




```
*/
```

# -- An application of JOIN

```
/*
   In this example, we split a very large table into several
   small tables, each containing 30 million rows.
/*


-- Create a small table out of the large table
-- No row should be included in more than one small table
-- The column “split” is the splitting flag for avoiding duplicates
-- Use the TOP clause to get the first 30 million un-used rows
-- This example doesn’t work with our sample databases

SELECT TOP 30000000 *
INTO MSA_EN_US_1
FROM MSA_EN_US
WHERE split = 'N'


/*
   Flag the rows we have just included in the new small table
   so that we don’t have any duplicate.
*/

/*
    JOIN the new small table to the original large table
    to determine what rows in the large table we just used and
    need to be flagged to avoid duplicates.
*/

UPDATE MSA_EN_US
SET split = '1'
FROM MSA_EN_US u
     JOIN MSA_EN_US_1 e
     ON u.PUID = e.PUID
```

## -- An application of JOIN, IN, and Subquery

**-- Use JOIN to filter data and determine what we need to DELETE**

**-- IN and a SUBQUERY are used in a condition**

**-- This example doesn’t work with our sample databases**

```
DELETE OutlookProfilePromo
FROM OutlookProfilePromo e
JOIN CIUserBase u
ON u.PassportID = e.puid
WHERE (u.AQSPreference is null OR
       u.AQSPreference =0 OR
       u.AQSPreference =9 ) AND
      (e.puid not in (select dim_email_key_REAL from LIST_348932))
```

# Useful Links

**USE SQL Server Management Studio**

http://msdn.microsoft.com/en-us/library/ms174173.aspx

**Writing SQL Queries**

http://technet.microsoft.com/en-us/library/bb264565(v=sql.90).aspx

**SQL Aggregate Functions**

http://msdn.microsoft.com/en-us/library/ms173454.aspx

**Types of JOIN in SQL Server**

http://www.codeproject.com/Tips/712941/Types-of-Join-in-SQL-Server

**GROUP BY and HAVING**
http://technet.microsoft.com/en-us/library/ms180199.aspx

**Subquery Fundamentals**

http://technet.microsoft.com/en-us/library/ms189575(v=sql.105).aspx