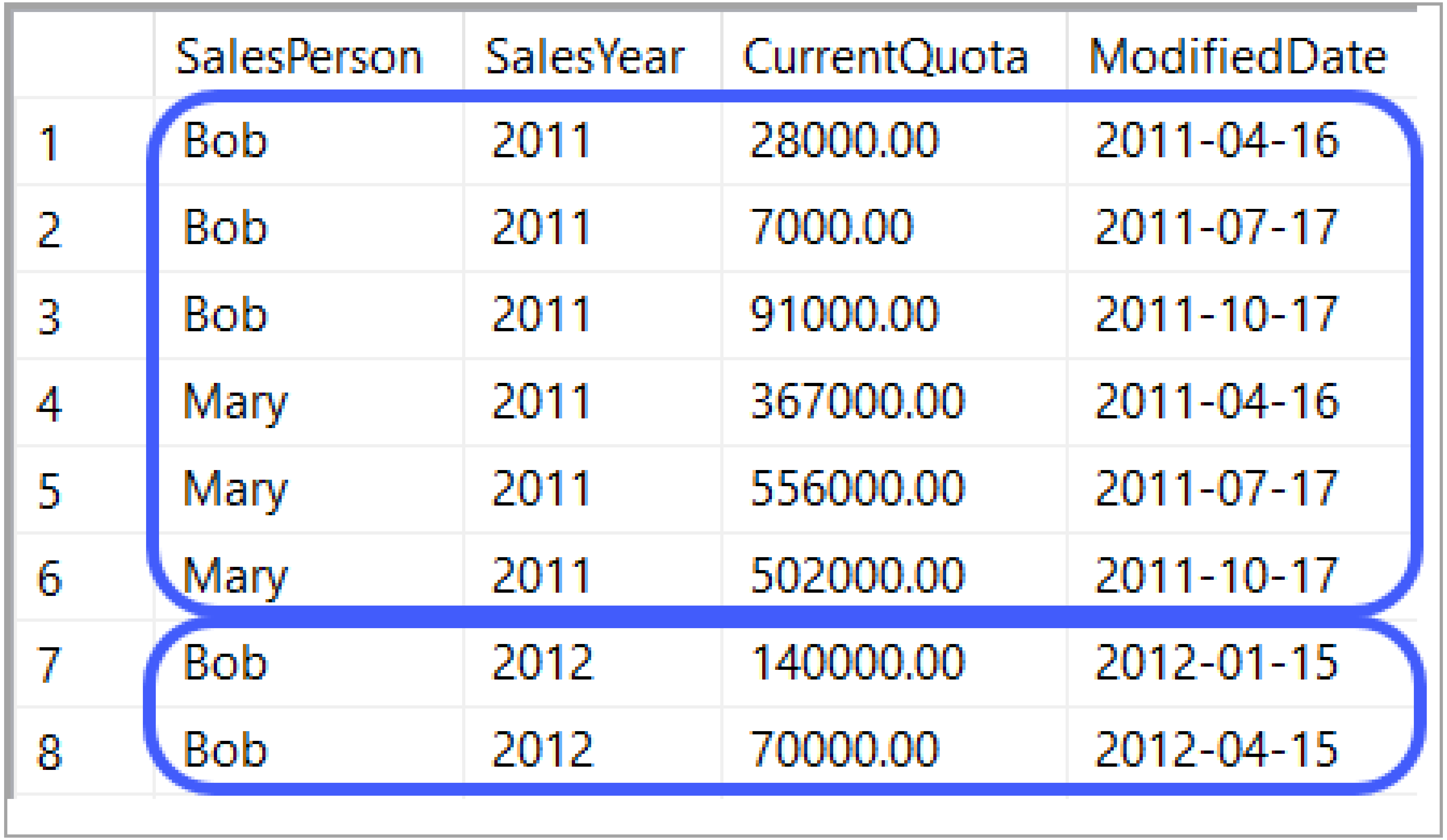
Window functions

INTERMEDIATE SQL SERVER

It provides the ability to create and analyze groups of data. With windowing functions, you can look at the current row, the next row, and the previous row all at the same time very efficiently.

Ginger Grant
Instructor



	SalesPerson	SalesYear	CurrentQuota	ModifiedDate
1	Bob	2011	28000.00	2011-04-16
2	Bob	2011	7000.00	2011-07-17
3	Bob	2011	91000.00	2011-10-17
4	Mary	2011	367000.00	2011-04-16
5	Mary	2011	556000.00	2011-07-17
6	Mary	2011	502000.00	2011-10-17
7	Bob	2012	140000.00	2012-01-15
8	Bob	2012	70000.00	2012-04-15

Using windowing functions, data within a table is processed as a group, allowing each group to be evaluated separately. In the example shown here, the data is broken into windows based upon the SalesYear column.

Grouping data in T-SQL

```sql
SELECT SalesPerson, SalesYear,
       CurrentQuota, ModifiedDate
FROM SaleGoal
WHERE SalesYear = 2011
```

```
+-----------+----------+---------------+----------------+
|SalesPerson |SalesYear | CurrentQuota  | ModifiedDate   |
+-----------+----------+---------------+----------------+
| Bob        | 2011     | 28000.00      | 2011-04-16     |
| Bob        | 2011     |  7000.00      | 2011-07-16     |
| Bob        | 2011     | 91000.00      | 2011-10-16     |
| Mary       | 2011     |367000.00      | 2011-04-16     |
| Mary       | 2011     |556000.00      | 2011-07-16     |
| Mary       | 2011     |502000.00      | 2011-10-16     |
+-----------+----------+---------------+----------------+
```

Window syntax in T-SQL

- Create the window with `OVER` clause
- `PARTITION BY` creates the frame

  create the window boundary based on the specified columns
- If you do not include `PARTITION BY` the frame is the entire table
- To arrange the results, use `ORDER BY`
- Allows aggregations to be created at the same time as the window

```
. . .
-- Create a Window data grouping
OVER (PARTITION BY SalesYear ORDER BY SalesYear)
```

Window functions (SUM)

After you create the window, you can add new functions.

```
SELECT SalesPerson, SalesYear, CurrentQuota,
       SUM(CurrentQuota)
       OVER (PARTITION BY SalesYear) AS YearlyTotal,
       ModifiedDate AS ModDate
FROM SaleGoal
```

We partition the table by SalesYear and use the windowing function SUM to add up every row of the CurrentQuota column in the window to provide a total for the entire window in the YearlyTotal column. When the year changes, the value in the YearlyTotal changes showing the total for the next year because the window is being partitioned by SalesYear.

```
+------------+----------+-------------+-----------+----------+
|SalesPerson |SalesYear |CurrentQuota |YearlyTotal | ModDate  |
+------------+----------+-------------+-----------+----------+
|Bob         |2011      |28000.00     |1551000.00  |2011-04-16|
|Bob         |2011      |7000.00      |1551000.00  |2011-07-17|
|Mary        |2011      |367000.00    |1551000.00  |2011-04-16|
|Mary        |2011      |556000.00    |1551000.00  |2011-07-15|
|Bob         |2012      |70000.00     |1859000.00  |2012-01-15|
|Bob         |2012      |154000.00    |1859000.00  |2012-04-16|
|Bob         |2012      |107000.00    |1859000.00  |2012-07-16|
...          |          |             |            |          |
+------------+----------+-------------+-----------+----------+
```

Window functions (COUNT)

```sql
SELECT SalesPerson, SalesYear, CurrentQuota,
       COUNT(CurrentQuota)
       OVER (PARTITION BY SalesYear) AS QuotaPerYear,
       ModifiedDate AS ModDate
FROM SaleGoal
```

```
+-----------+---------+-----------+-------------+---------+
|SalesPerson |SalesYear |CurrentQuota|QuotaPerYear  | ModDate  |
+-----------+---------+-----------+-------------+---------+
|Bob         |2011      |28000.00    |4              |2011-04-16|
|Bob         |2011      |7000.00     |4              |2011-07-17|
|Mary        |2011      |367000.00   |4              |2011-04-16|
|Mary        |2011      |556000.00   |4              |2011-07-15|
|Bob         |2012      |70000.00    |8              |2012-01-15|
|Bob         |2012      |154000.00   |8              |2012-04-15|
|Bob         |2012      |107000.00   |8              |2012-10-16|
...
+-----------+---------+-----------+-----------+---------+
```

- Notice the count starts over for each window in column `QuotaPerYear`

Let's practice!

INTERMEDIATE SQL SERVER

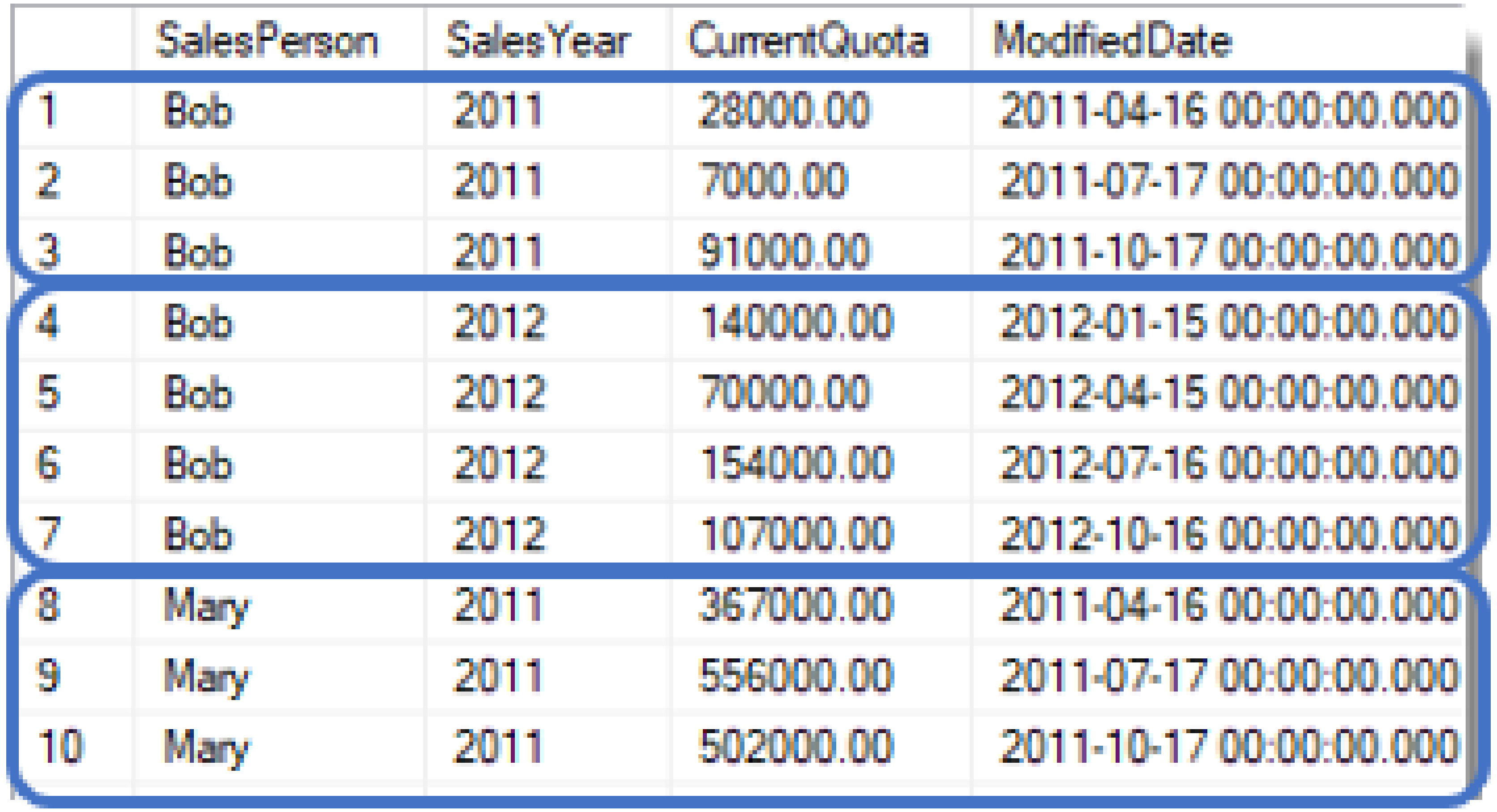
Common window functions

INTERMEDIATE SQL SERVER

Ginger Grant
Instructor

FIRST_VALUE() and LAST_VALUE()

- `FIRST_VALUE()` returns the first value in the window
- `LAST_VALUE()` returns the last value in the window



	SalesPerson	SalesYear	CurrentQuota	ModifiedDate
1	Bob	2011	28000.00	2011-04-16 00:00:00.000
2	Bob	2011	7000.00	2011-07-17 00:00:00.000
3	Bob	2011	91000.00	2011-10-17 00:00:00.000
4	Bob	2012	140000.00	2012-01-15 00:00:00.000
5	Bob	2012	70000.00	2012-04-15 00:00:00.000
6	Bob	2012	154000.00	2012-07-16 00:00:00.000
7	Bob	2012	107000.00	2012-10-16 00:00:00.000
8	Mary	2011	367000.00	2011-04-16 00:00:00.000
9	Mary	2011	556000.00	2011-07-17 00:00:00.000
10	Mary	2011	502000.00	2011-10-17 00:00:00.000

FIRST_VALUE() and LAST_VALUE() in T-SQL

- Note that for FIRST_VALUE and LAST_VALUE the ORDER BY command is required

```
-- Select the columns
SELECT SalesPerson, SalesYear, CurrentQuota,
    -- First value from every window
       FIRST_VALUE(CurrentQuota)
       OVER (PARTITION BY SalesYear ORDER BY ModifiedDate) AS StartQuota,
    -- Last value from every window
       LAST_VALUE(CurrentQuota)
       OVER (PARTITION BY SalesYear ORDER BY ModifiedDate) AS EndQuota,
       ModifiedDate as ModDate
FROM SaleGoal
```

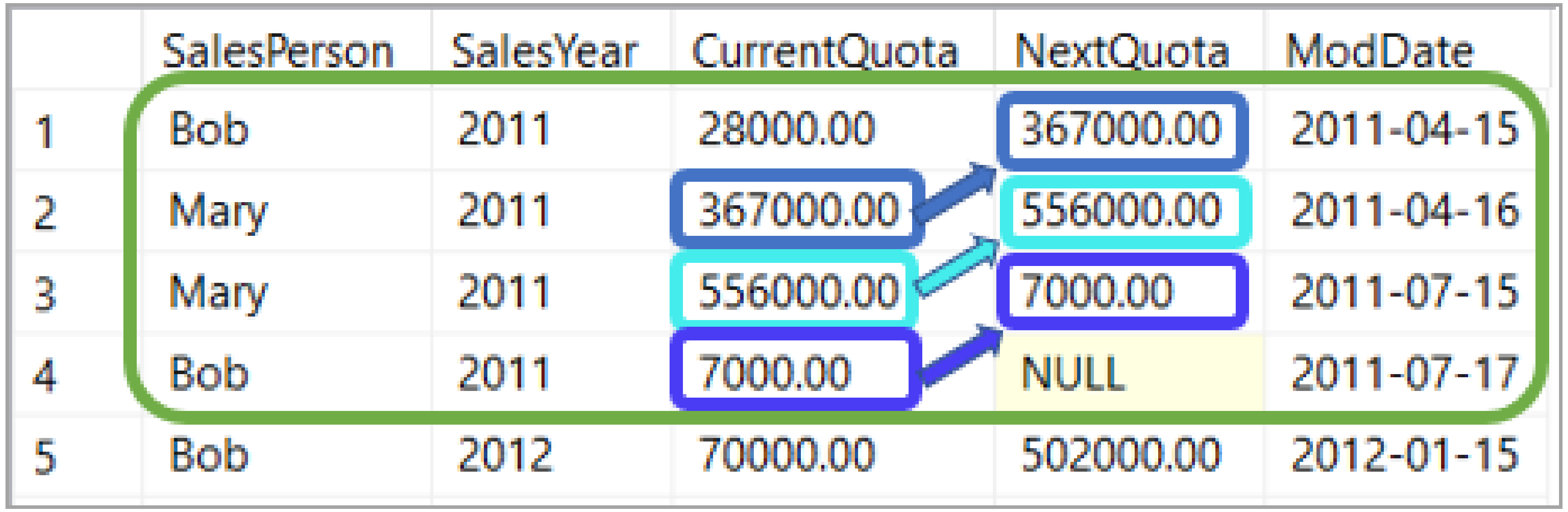
The window is sorted based upon the values in the ModifiedDate field.

Results

```
+------------+----------+------------+----------+----------+----------+
|SalesPerson |SalesYear |CurrentQuota|StartQuota| EndQuota |ModDate   |
+------------+----------+------------+----------+----------+----------+
|Bob         |2011      |28000.00    |28000.00  |91000.00  |2011-04-16|
|Bob         |2011      |7000.00     |28000.00  |91000.00  |2011-07-17|
|Bob         |2011      |91000.00    |28000.00  |91000.00  |2011-10-17|
|Bob         |2012      |140000.00   |140000.00 |107000.00 |2012-01-15|
|Bob         |2012      |70000.00    |140000.00 |107000.00 |2012-04-15|
|Bob         |2012      |154000.00   |140000.00 |107000.00 |2012-07-16|
|Bob         |2012      |107000.00   |140000.00 |107000.00 |2012-10-16|
...
+------------+----------+------------+----------+----------+----------+
```

Getting the next value with LEAD()

- Provides the ability to query the value from the next row
- NextQuota column is created by using `LEAD()` Using LEAD(), you can compare the value of the current row to the value of the next row in the window.
- Requires the use of `ORDER BY` to order the rows



	SalesPerson	SalesYear	CurrentQuota	NextQuota	ModDate
1	Bob	2011	28000.00	367000.00	2011-04-15
2	Mary	2011	367000.00	556000.00	2011-04-16
3	Mary	2011	556000.00	7000.00	2011-07-15
4	Bob	2011	7000.00	NULL	2011-07-17
5	Bob	2012	70000.00	502000.00	2012-01-15

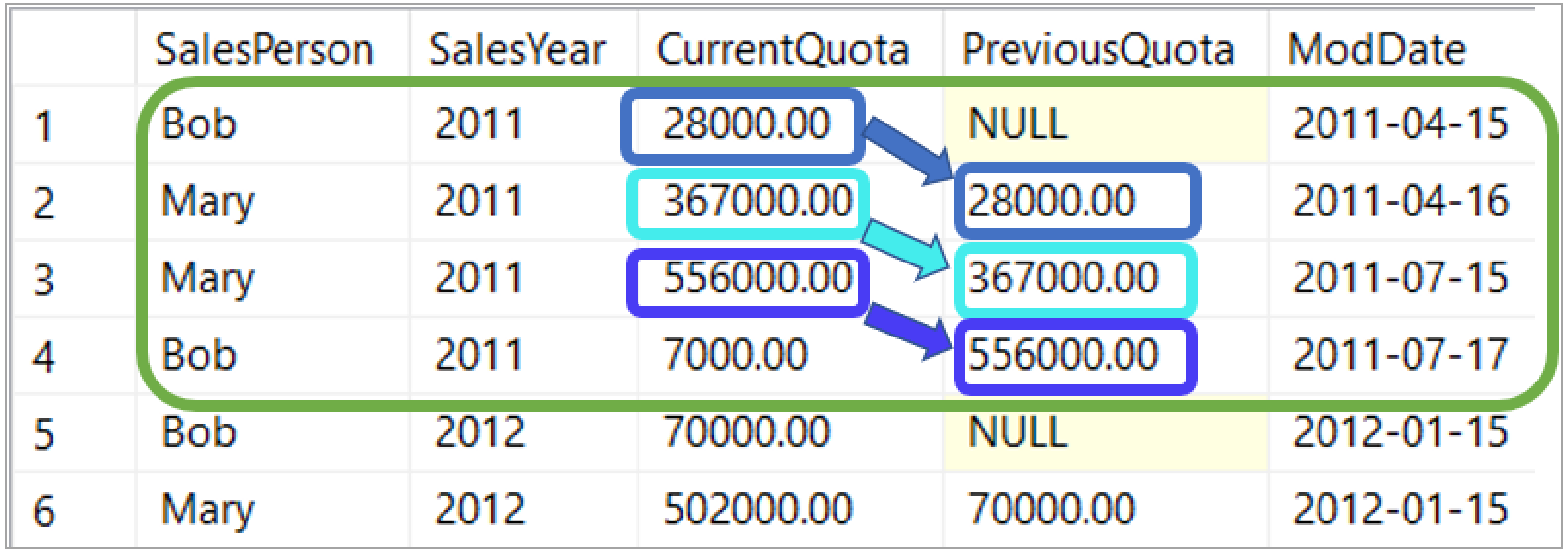
LEAD() in T-SQL

```sql
SELECT SalesPerson, SalesYear, CurrentQuota,
-- Create a window function to get the values from the next row
       LEAD(CurrentQuota)
       OVER (PARTITION BY SalesYear ORDER BY ModifiedDate) AS NextQuota,
       ModifiedDate AS ModDate
FROM SaleGoal
```

```
+------------+----------+------------+------------+----------+
|SalesPerson |SalesYear |CurrentQuota|NextQuota   | ModDate  |
+------------+----------+------------+------------+----------+
|Bob         |2011      |28000.00    |367000.00   |2011-04-15|
|Mary        |2011      |367000.00   |556000.00   |2011-04-16|
|Mary        |2011      |556000.00   |7000.00     |2011-07-15|
|Bob         |2011      |7000.00     |NULL        |2011-07-17|
|Bob         |2012      |70000.00    |502000.00   |2012-01-15|
|Mary        |2012      |502000.00   |154000.00   |2012-01-16|
...
+------------+----------+------------+------------+----------+
```

Getting the previous value with LAG()

- Provides the ability to query the value from the previous row
- PreviousQuota column is created by using `LAG()`
- Requires the use of `ORDER BY` to order the rows



	SalesPerson	SalesYear	CurrentQuota	PreviousQuota	ModDate
1	Bob	2011	28000.00	NULL	2011-04-15
2	Mary	2011	367000.00	28000.00	2011-04-16
3	Mary	2011	556000.00	367000.00	2011-07-15
4	Bob	2011	7000.00	556000.00	2011-07-17
5	Bob	2012	70000.00	NULL	2012-01-15
6	Mary	2012	502000.00	70000.00	2012-01-15

LAG() in T-SQL

```
SELECT SalesPerson, SalesYear, CurrentQuota,
-- Create a window function to get the values from the previous row
       LAG(CurrentQuota)
       OVER (PARTITION BY SalesYear ORDER BY ModifiedDate) AS PreviousQuota,
       ModifiedDate AS ModDate
FROM SaleGoal
```

```
+------------+----------+------------+------------+----------+
|SalesPerson |SalesYear |CurrentQuota|PreviousQuota |ModDate   |
+------------+----------+------------+------------+----------+
|Bob         |2011      |28000.00    |NULL         |2011-04-15|
|Mary        |2011      |367000.00   |28000.00     |2011-04-16|
|Mary        |2011      |556000.00   |367000.00    |2011-07-15|
|Bob         |2011      |7000.00.00  |556000.00    |2011-07-17|
|Bob         |2012      |7000.00     |NULL         |2012-01-15|
|Mary        |2012      |502000.00   |7000.00      |2012-01-16|
...
+------------+----------+------------+------------+----------+
```

Let's practice !

INTERMEDIATE SQL SERVER

Increasing window complexity

INTERMEDIATE SQL SERVER

Ginger Grant
Instructor

Reviewing aggregations

```sql
SELECT SalesPerson, SalesYear, CurrentQuota,
       SUM(CurrentQuota)
       OVER (PARTITION BY SalesYear) AS YearlyTotal,
       ModifiedDate as ModDate
FROM SaleGoal
```

```
+------------+----------+------------+-----------+----------+
|SalesPerson |SalesYear |CurrentQuota|YearlyTotal | ModDate  |
+------------+----------+------------+-----------+----------+
|Bob         |2011      |28000.00    |1551000.00  |2011-04-16|
|Bob         |2011      |7000.00     |1551000.00  |2011-07-17|
|Bob         |2011      |91000.00    |1551000.00  |2011-10-17|
|Mary        |2011      |140000.00   |1551000.00  |2012-04-15|
|Mary        |2011      |70000.00    |1551000.00  |2012-07-15|
|Mary        |2011      |154000.00   |1551000.00  |2012-01-15|
|Mary        |2012      |107000.00   |1859000.00  |2012-01-16|
...
+------------+----------+------------+-----------+----------+
```

Adding ORDER BY to an aggregation

```
SELECT SalesPerson, SalesYear, CurrentQuota,
       SUM(CurrentQuota)
       OVER (PARTITION BY SalesYear ORDER BY SalesPerson) AS YearlyTotal,
       ModifiedDate as ModDate
FROM SaleGoal
```

```
+------------+----------+------------+-----------+----------+
|SalesPerson |SalesYear |CurrentQuota|YearTotal  | ModDate  |
+------------+----------+------------+-----------+----------+
|Bob         |2011      |28000.00    |35000.00   |2011-04-16|
|Bob         |2011      |7000.00     |35000.00   |2011-07-17|
|Mary        |2011      |367000.00   |958000.00  |2011-10-17|
|Mary        |2011      |556000.00   |958000.00  |2012-04-15|
|Bob         |2012      |70000.00    |401000.00  |2012-07-15|
|Bob         |2012      |154000.00   |401000.00  |2012-10-16|
...
+------------+----------+------------+-----------+----------+
```

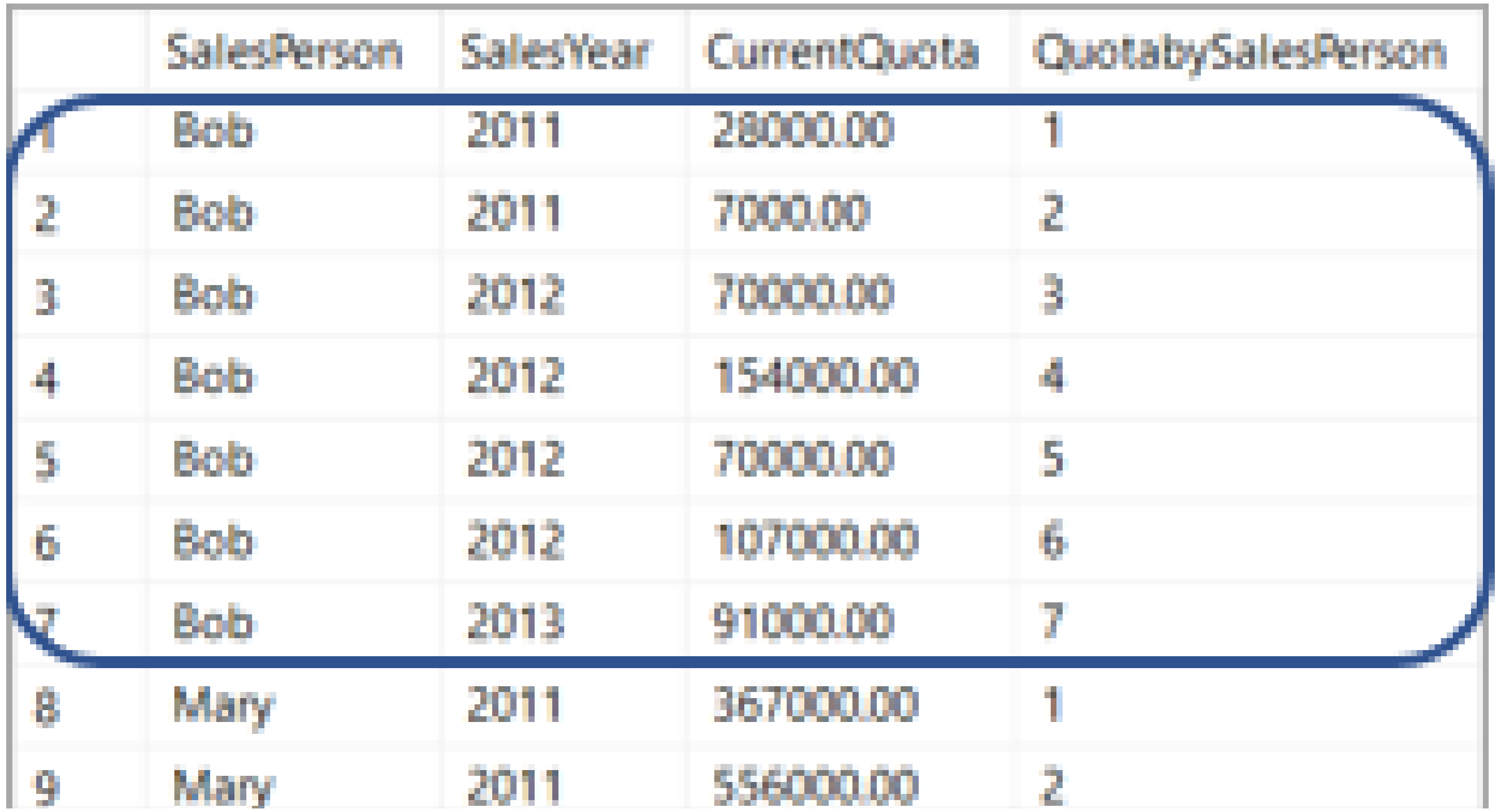
Creating a running total with ORDER BY

```
SELECT SalesPerson, SalesYear, CurrentQuota,
       SUM(CurrentQuota)
       OVER (PARTITION BY SalesYear ORDER BY ModifiedDate) as RunningTotal,
       ModifiedDate as ModDate
FROM SaleGoal
```

```
+------------+----------+------------+-----------+----------+
|SalesPerson |SalesYear |CurrentQuota|RunningTotal| ModDate   |
+------------+----------+------------+-----------+----------+
|Bob         |2011      |28000.00    |28000.00    |2011-04-16|
|Mary        |2011      |367000.00   |395000.00   |2011-07-17|
|Mary        |2011      |556000.00   |951000.00   |2011-10-17|
|Bob         |2011      |7000.00     |958000.00   |2012-04-15|
|Bob         |2012      |70000.00    |70000.00    |2012-01-15|
|Mary        |2012      |502000.00   |572000.00   |2012-01-16|
...
+------------+----------+------------+-----------+----------+
```

Adding row numbers

- `ROW_NUMBER()` sequentially numbers the rows in the window
- `ORDER BY` is required when using `ROW_NUMBER()`



	SalesPerson	SalesYear	CurrentQuota	QuotabySalesPerson
1	Bob	2011	28000.00	1
2	Bob	2011	7000.00	2
3	Bob	2012	70000.00	3
4	Bob	2012	154000.00	4
5	Bob	2012	70000.00	5
6	Bob	2012	107000.00	6
7	Bob	2013	91000.00	7
8	Mary	2011	367000.00	1
9	Mary	2011	556000.00	2

Adding row numbers in T-SQL

```
SELECT SalesPerson, SalesYear, CurrentQuota,
       ROW_NUMBER()
       OVER (PARTITION BY SalesPerson ORDER BY SalesYear) AS QuotabySalesPerson
FROM SaleGoal
```

```
+------------+----------+------------+------------------+
|SalesPerson |SalesYear |CurrentQuota|QuotabySalesPerson|
+------------+----------+------------+------------------+
|Bob         |2011      |28000.00    |1                 |
|Bob         |2011      |7000.00     |2                 |
|Bob         |2011      |70000.00    |3                 |
|Bob         |2011      |154000.00   |4                 |
|Bob         |2012      |70000.00    |5                 |
|Bob         |2012      |107000.00   |6                 |
|Bob         |2012      |91000.00    |7                 |
|Mary        |2011      |367000.00   |1                 |
...
+------------+----------+------------+------------------+
```

Let's practice!

INTERMEDIATE SQL SERVER

Using windows for calculating statistics

INTERMEDIATE SQL SERVER

Ginger Grant
Instructor

Calculating the standard deviation

- Calculate standard deviation either for the entire table or for each window
- `STDEV()` calculates the standard deviation

Calculating the standard deviation for the entire table

```
SELECT SalesPerson, SalesYear, CurrentQuota,
       STDEV(CurrentQuota)
       OVER () AS StandardDev,
       ModifiedDate  AS ModDate
FROM SaleGoal
```

Since no columns are listed in OVER, we did not include PARTITION BY, only one window is created for the entire table. As a result, only one value is repeated across the entire StandardDev column.

```
+------------+----------+------------+-----------------+----------+
|SalesPerson |SalesYear |CurrentQuota|StandardDev      | ModDate  |
+------------+----------+------------+-----------------+----------+
|Bob         |2011      |28000.00    |267841.370964233 |2011-04-16|
|Bob         |2011      |7000.00     |267841.370964233 |2011-07-17|
|Bob         |2011      |91000.00    |267841.370964233 |2011-10-17|
|Bob         |2012      |140000.00   |267841.370964233 |2012-01-15|
|Bob         |2012      |70000.00    |267841.370964233 |2012-04-15|
...
```

Calculating the standard deviation for each partition

```sql
SELECT SalesPerson, SalesYear, CurrentQuota,
       STDEV(CurrentQuota)
       OVER (PARTITION BY SalesYear ORDER BY SalesYear) AS StDev,
       ModifiedDate AS ModDate
FROM SaleGoal
```

```
+------------+----------+------------+-------------+----------+
|SalesPerson |SalesYear |CurrentQuota|StDev        | ModDate  |
+------------+----------+------------+-------------+----------+
|Bob         |2011      |28000.00    |267841.54080 |2011-04-16|
|Bob         |2011      |7000.00     |267841.54080 |2011-07-17|
|Mary        |2011      |91000.00    |267841.54080 |2011-04-16|
|Mary        |2011      |140000.00   |267841.54080 |2011-07-15|
|Bob         |2012      |70000.00    |246538.86248 |2012-01-15|
|Bob         |2012      |154000.00   |246538.86248 |2012-04-15|
|Bob         |2012      |107000.00   |246538.86248 |2012-07-16|
...
+------------+----------+------------+------------+-----------+
```

Calculating the mode

- Mode is the value which appears the most often in your data
- To calculate mode:
  - Create a CTE containing an ordered count of values using ROW_NUMBER
  - Write a query using the CTE to pick the value with the highest row number

Calculating the mode in T-SQL (I)

```sql
WITH QuotaCount AS (
SELECT SalesPerson, SalesYear, CurrentQuota,
       ROW_NUMBER()
       OVER (PARTITION BY CurrentQuota ORDER BY CurrentQuota) AS QuotaList
FROM SaleGoal
)
SELECT * FROM QuotaCount
```

```
+------------+----------+------------+-------------+
|SalesPerson |SalesYear |CurrentQuota|QuotaList    |
+------------+----------+------------+-------------+
|Bob         |2011      |7000.00     |1            |
|Bob         |2011      |28000.00    |1            |
|Bob         |2011      |70000.00    |1            |
|Bob         |2012      |70000.00    |2            |
|Mary        |2012      |73000.00    |1            |
...
+------------+----------+-----------+------------+
```

- Notice there are two values for 70,000.00

Calculating the mode in T-SQL (II)

```
WITH QuotaCount AS (
SELECT SalesPerson, SalesYear, CurrentQuota,
       ROW_NUMBER()
       OVER (PARTITION BY CurrentQuota ORDER BY CurrentQuota) AS QuotaList
FROM SaleGoal
)

SELECT CurrentQuota, QuotaList AS Mode
FROM QuotaCount
WHERE QuotaList IN (SELECT  MAX(QuotaList) FROM QuotaCount)
```

```
+-----------+----------+
|CurrentQuota|Mode      |
+-----------+----------+
|70000.00    |2          |
+-----------+----------+
```

Let's practice!

INTERMEDIATE SQL SERVER