WINDOW FUNCTIONS IN SQL

SQL Series Part 11

-Mayuri Dandekar

WINDOW FUNCTIONS

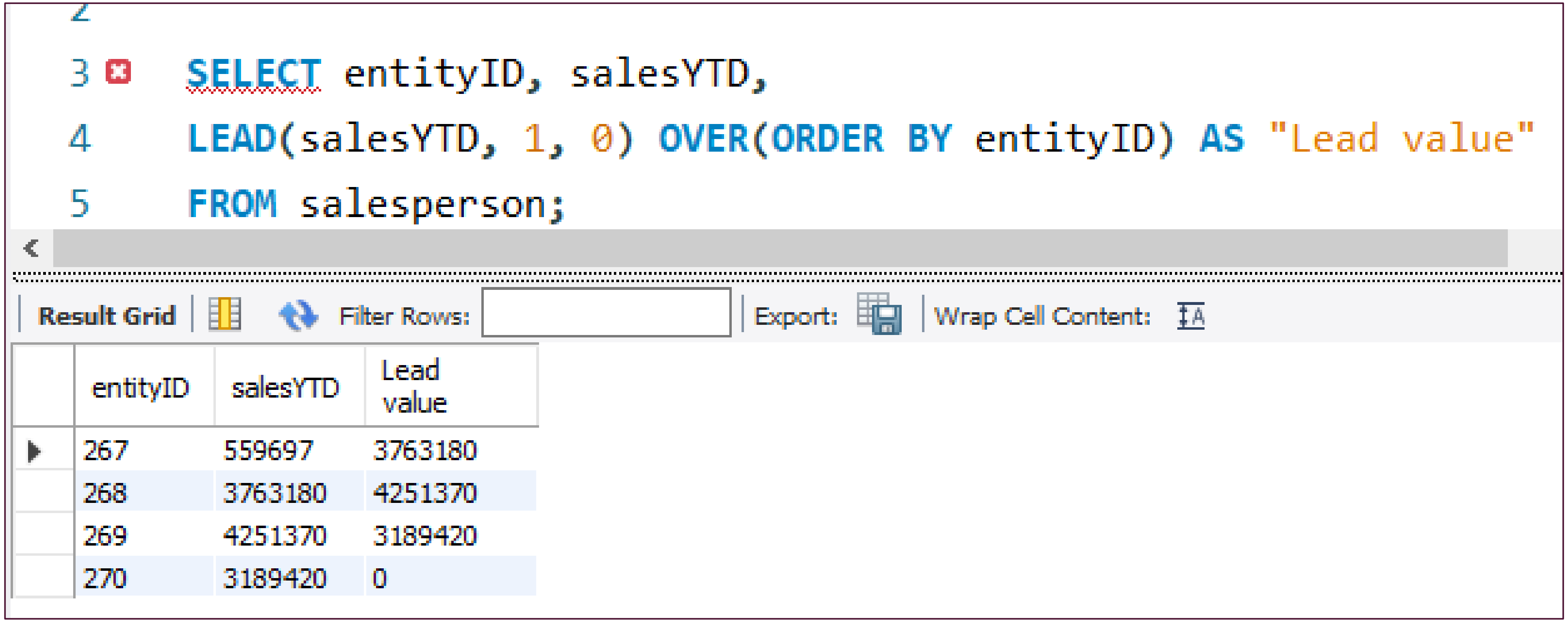
Window functions applies **aggregate, ranking and analytic functions** over a particular window (set of rows).

And **OVER clause** is used with window functions to define that window.

SYNTAX—

```
SELECT column_name(s),
fun( ) OVER ( [ <PARTITION BY Clause> ]
              [ <ORDER BY Clause> ]
              [ <ROW or RANGE Clause> ] )
FROM table_name;
```

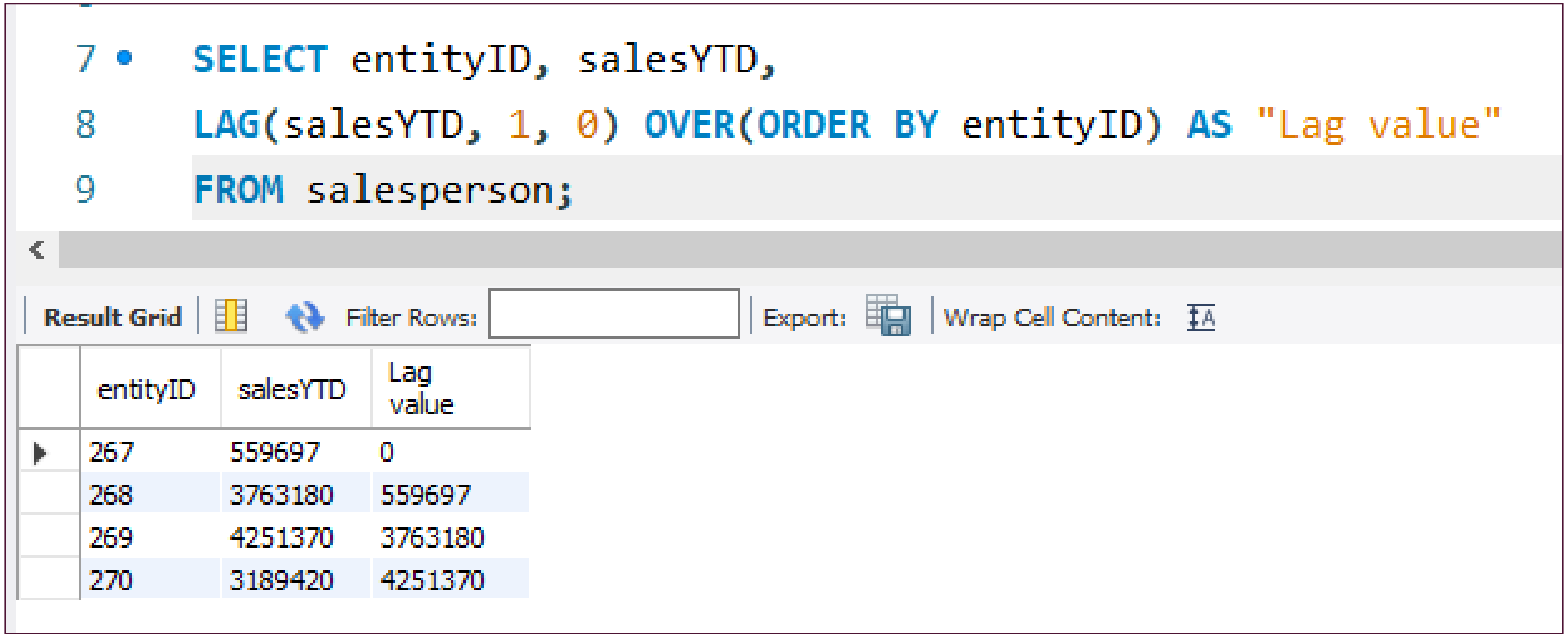
LEAD



```
SELECT entityID, salesYTD,
LEAD(salesYTD, 1, 0) OVER(ORDER BY entityID) AS "Lead value"
FROM salesperson;
```

entityID	salesYTD	Lead value
267	559697	3763180
268	3763180	4251370
269	4251370	3189420
270	3189420	0

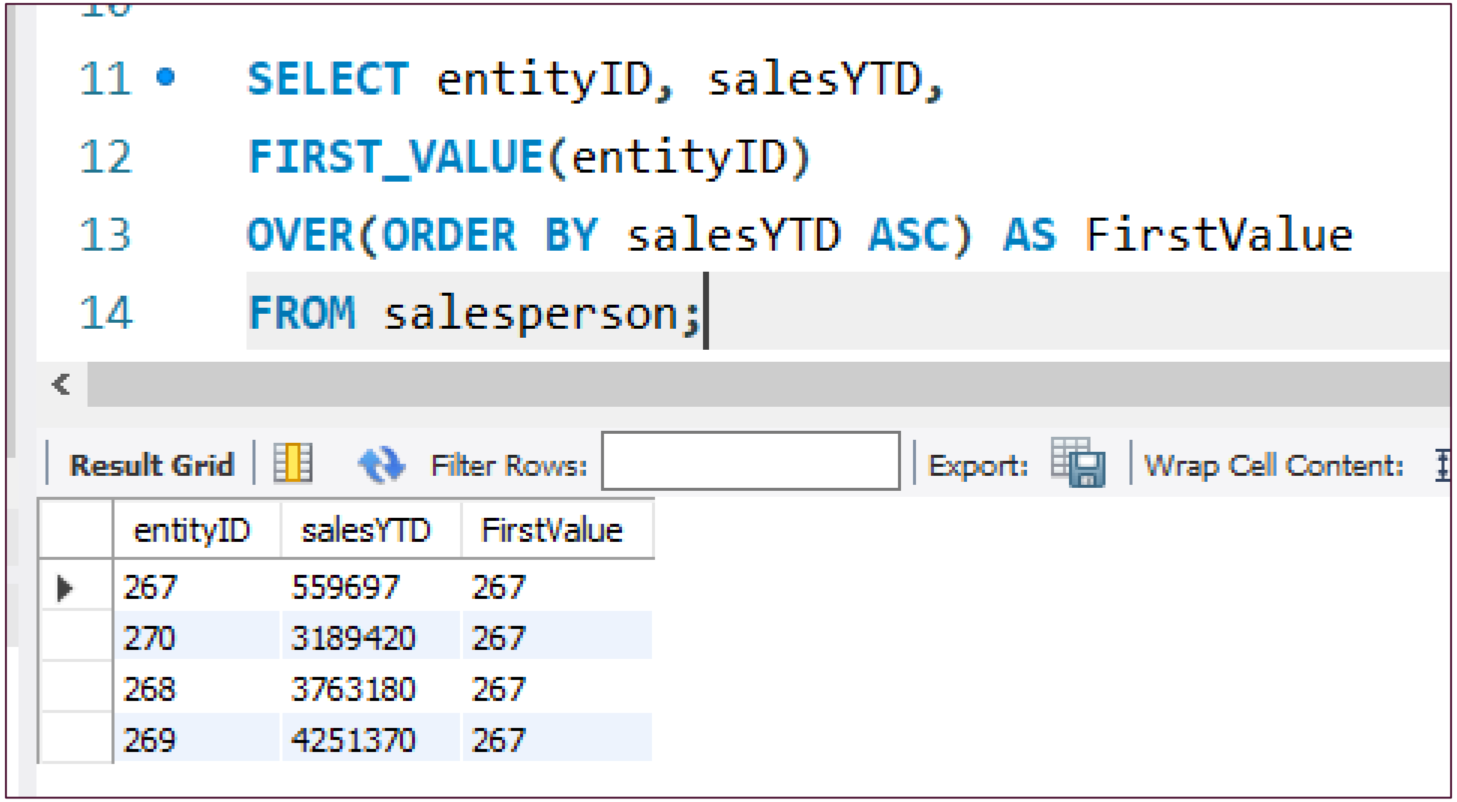
LAG



```
SELECT entityID, salesYTD,
LAG(salesYTD, 1, 0) OVER(ORDER BY entityID) AS "Lag value"
FROM salesperson;
```

entityID	salesYTD	Lag value
267	559697	0
268	3763180	559697
269	4251370	3763180
270	3189420	4251370

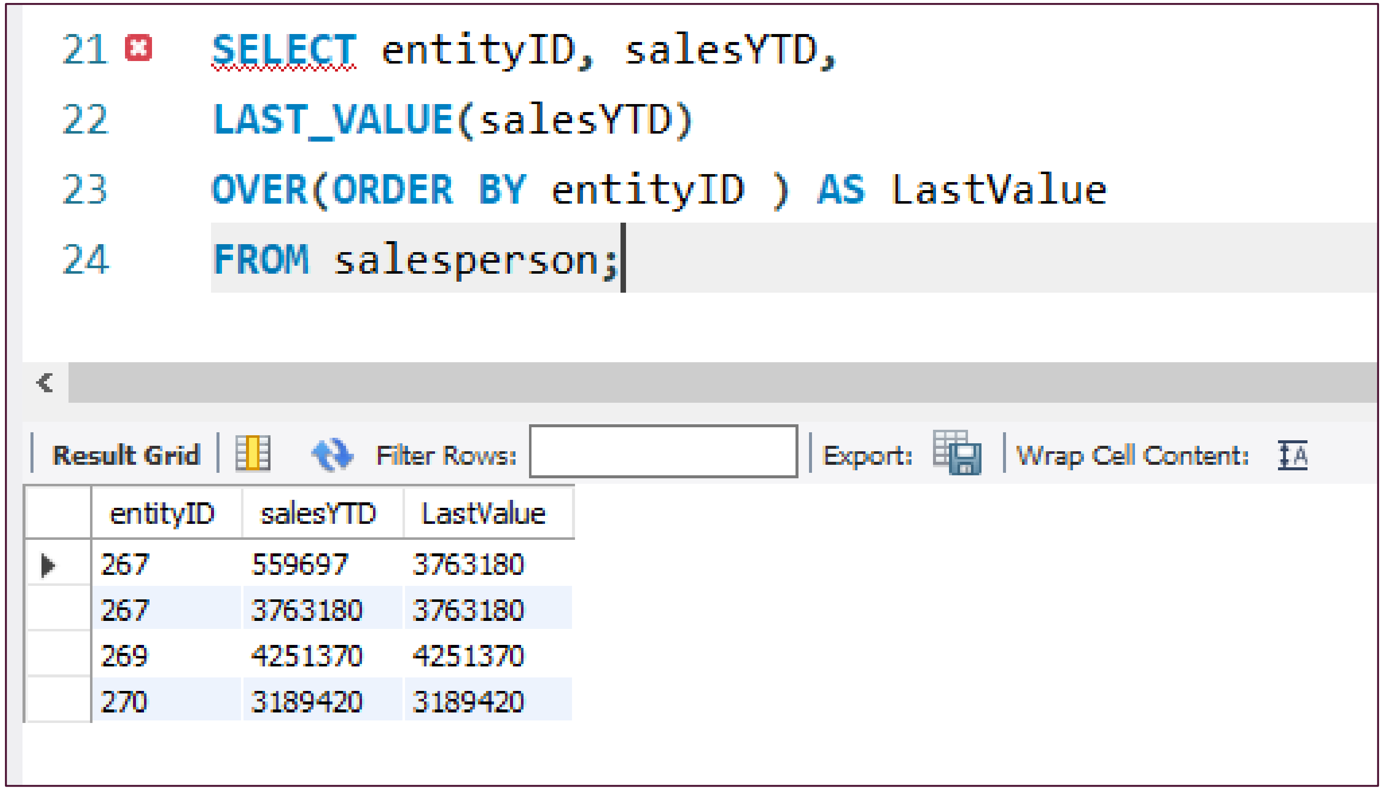
FIRST_VALUE



```
SELECT entityID, salesYTD,
FIRST_VALUE(entityID)
OVER(ORDER BY salesYTD ASC) AS FirstValue
FROM salesperson;
```

entityID	salesYTD	FirstValue
267	559697	267
270	3189420	267
268	3763180	267
269	4251370	267

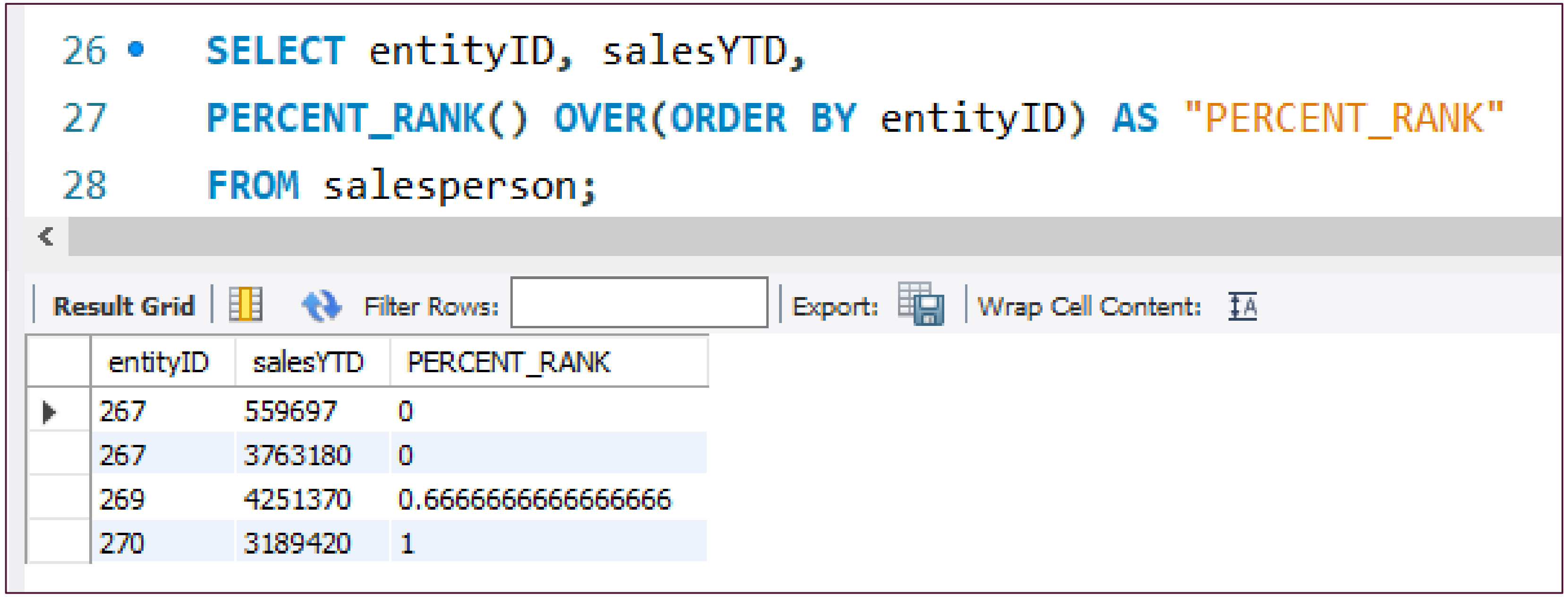
LAST_VALUE



```sql
SELECT entityID, salesYTD,
LAST_VALUE(salesYTD)
OVER(ORDER BY entityID ) AS LastValue
FROM salesperson;
```

entityID	salesYTD	LastValue
267	559697	3763180
267	3763180	3763180
269	4251370	4251370
270	3189420	3189420

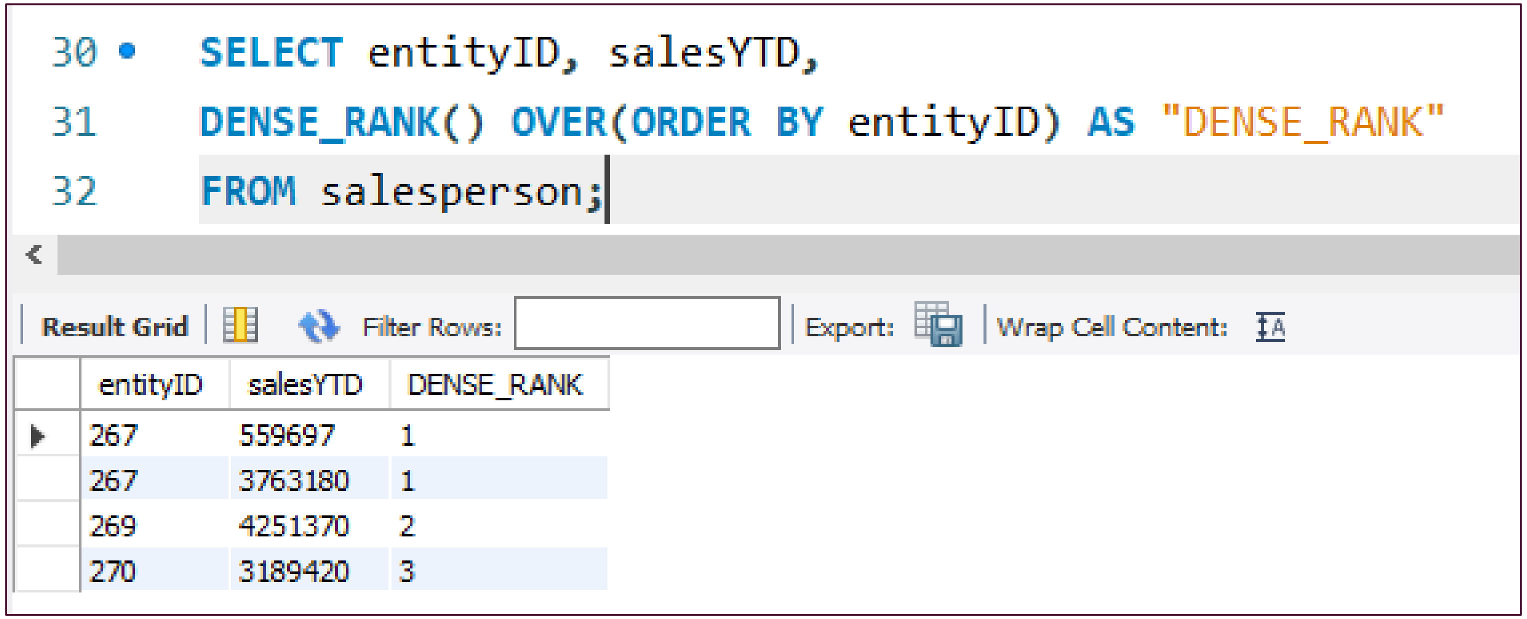
PERCENT_RANK



```sql
SELECT entityID, salesYTD,
PERCENT_RANK() OVER(ORDER BY entityID) AS "PERCENT_RANK"
FROM salesperson;
```

entityID	salesYTD	PERCENT_RANK
267	559697	0
267	3763180	0
269	4251370	0.6666666666666666
270	3189420	1

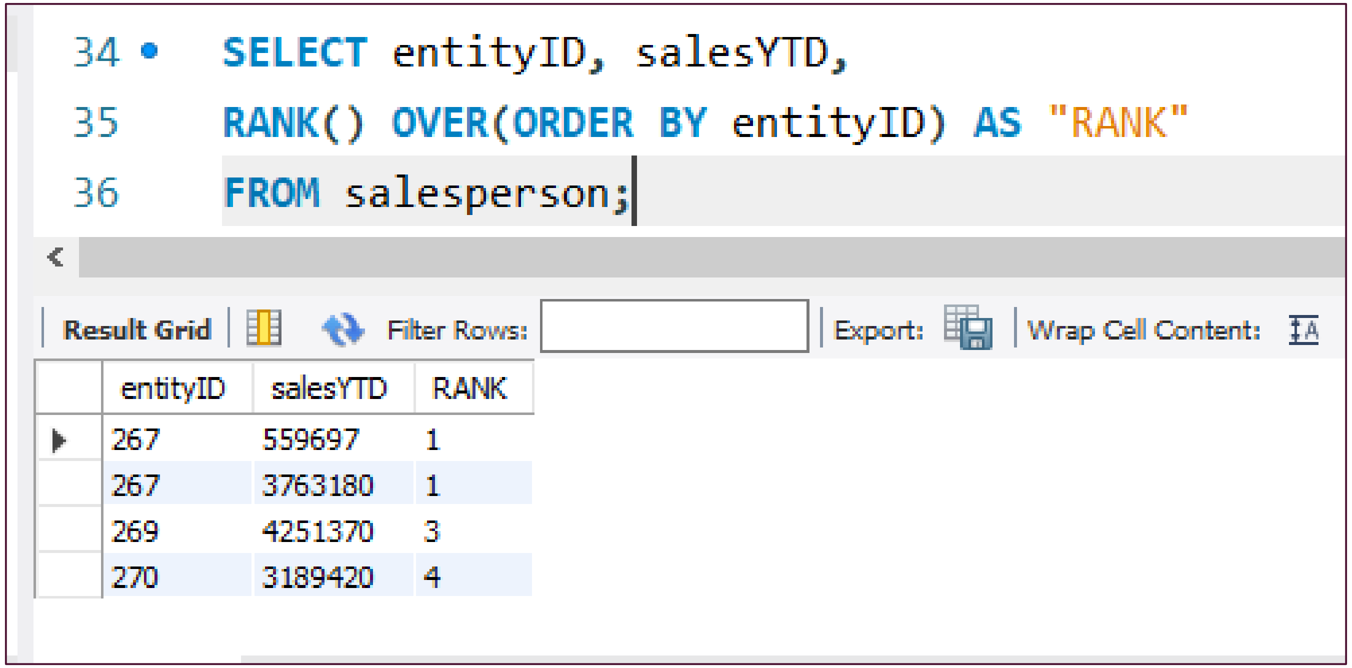
DENSE_RANK



```sql
SELECT entityID, salesYTD,
DENSE_RANK() OVER(ORDER BY entityID) AS "DENSE_RANK"
FROM salesperson;
```

entityID	salesYTD	DENSE_RANK
267	559697	1
267	3763180	1
269	4251370	2
270	3189420	3

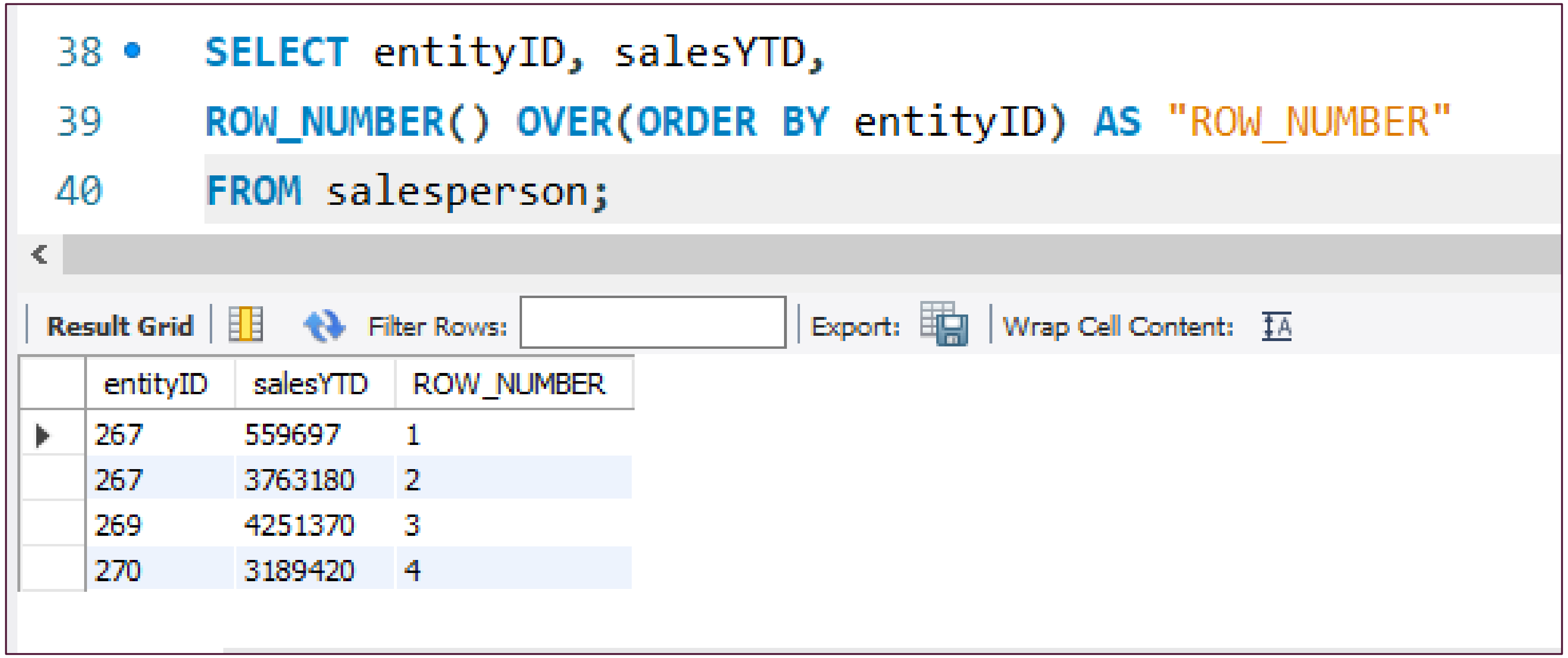
RANK



```sql
SELECT entityID, salesYTD,
RANK() OVER(ORDER BY entityID) AS "RANK"
FROM salesperson;
```

entityID	salesYTD	RANK
267	559697	1
267	3763180	1
269	4251370	3
270	3189420	4

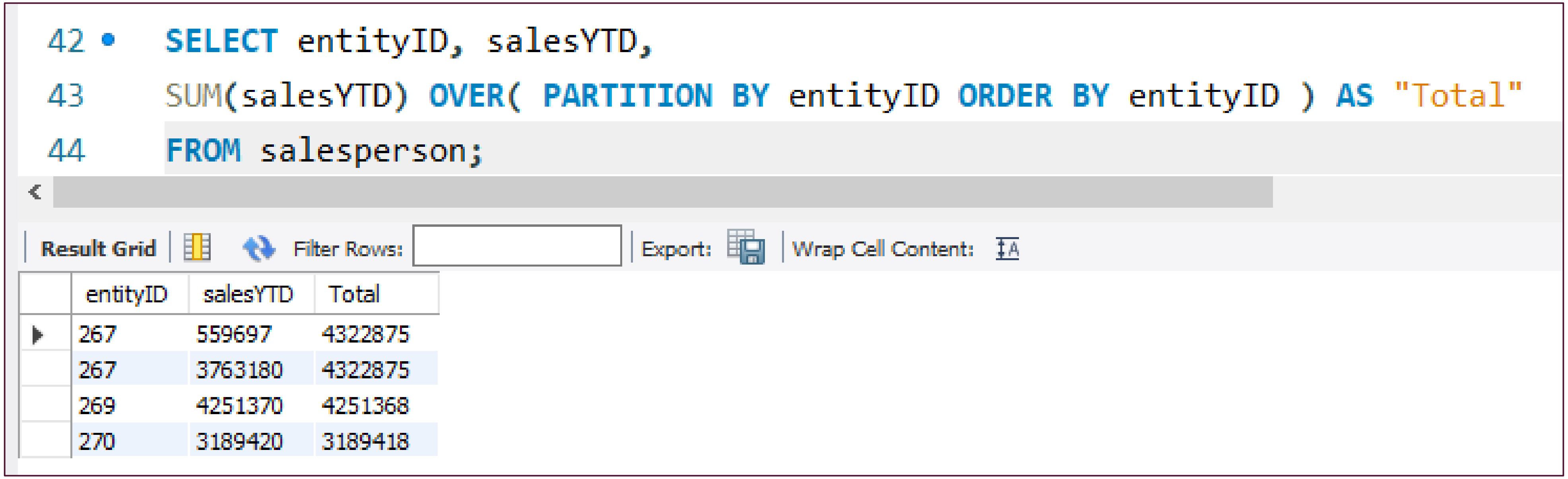
ROW_NUMBER



```sql
SELECT entityID, salesYTD,
ROW_NUMBER() OVER(ORDER BY entityID) AS "ROW_NUMBER"
FROM salesperson;
```

entityID	salesYTD	ROW_NUMBER
267	559697	1
267	3763180	2
269	4251370	3
270	3189420	4

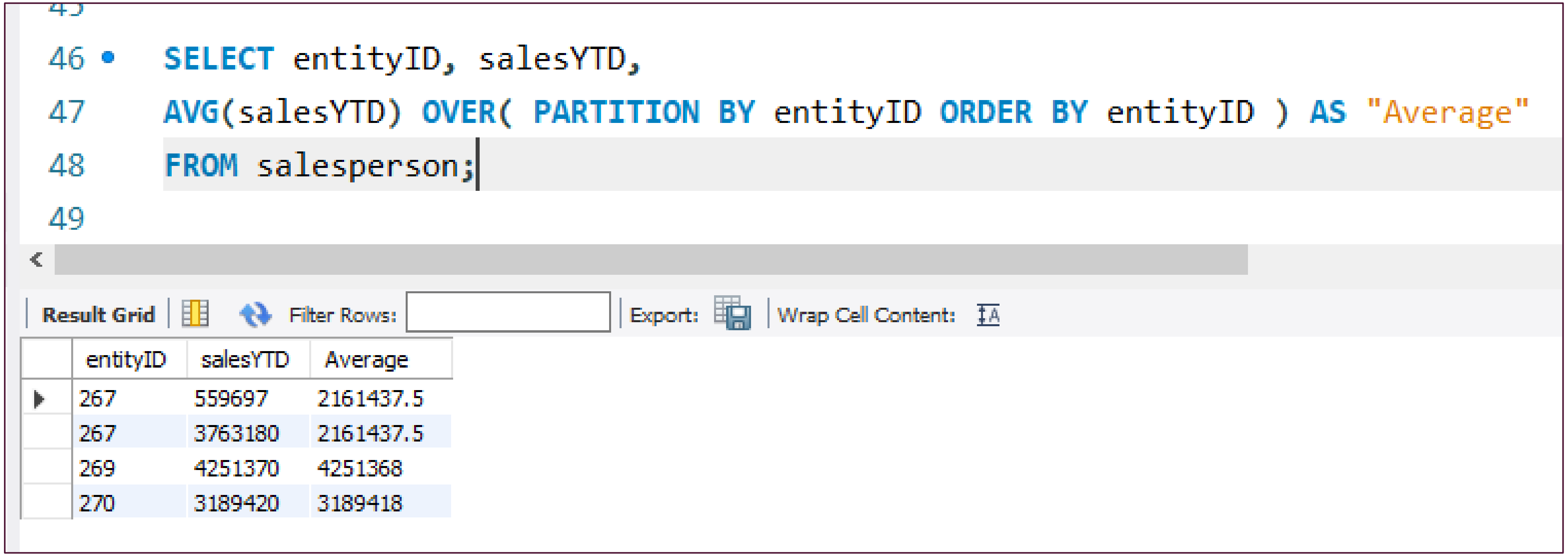
SUM



```sql
SELECT entityID, salesYTD,
SUM(salesYTD) OVER( PARTITION BY entityID ORDER BY entityID ) AS "Total"
FROM salesperson;
```

entityID	salesYTD	Total
267	559697	4322875
267	3763180	4322875
269	4251370	4251368
270	3189420	3189418

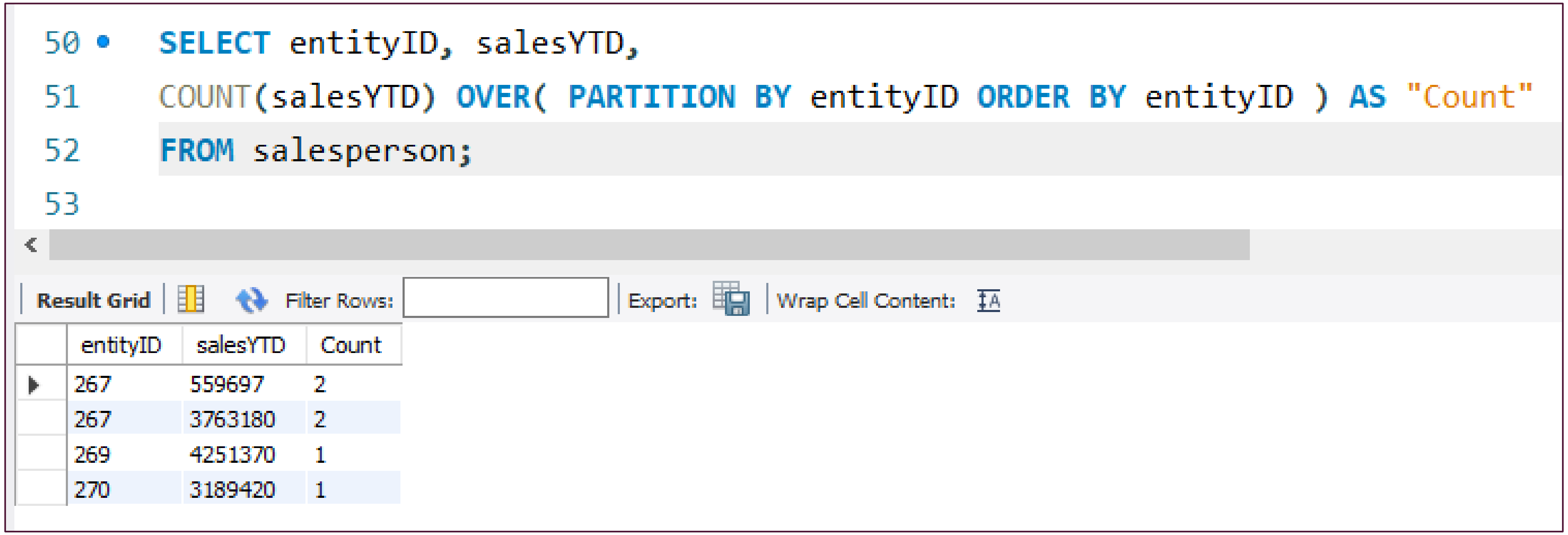
AVERAGE



```sql
SELECT entityID, salesYTD,
AVG(salesYTD) OVER( PARTITION BY entityID ORDER BY entityID ) AS "Average"
FROM salesperson;
```

entityID	salesYTD	Average
267	559697	2161437.5
267	3763180	2161437.5
269	4251370	4251368
270	3189420	3189418

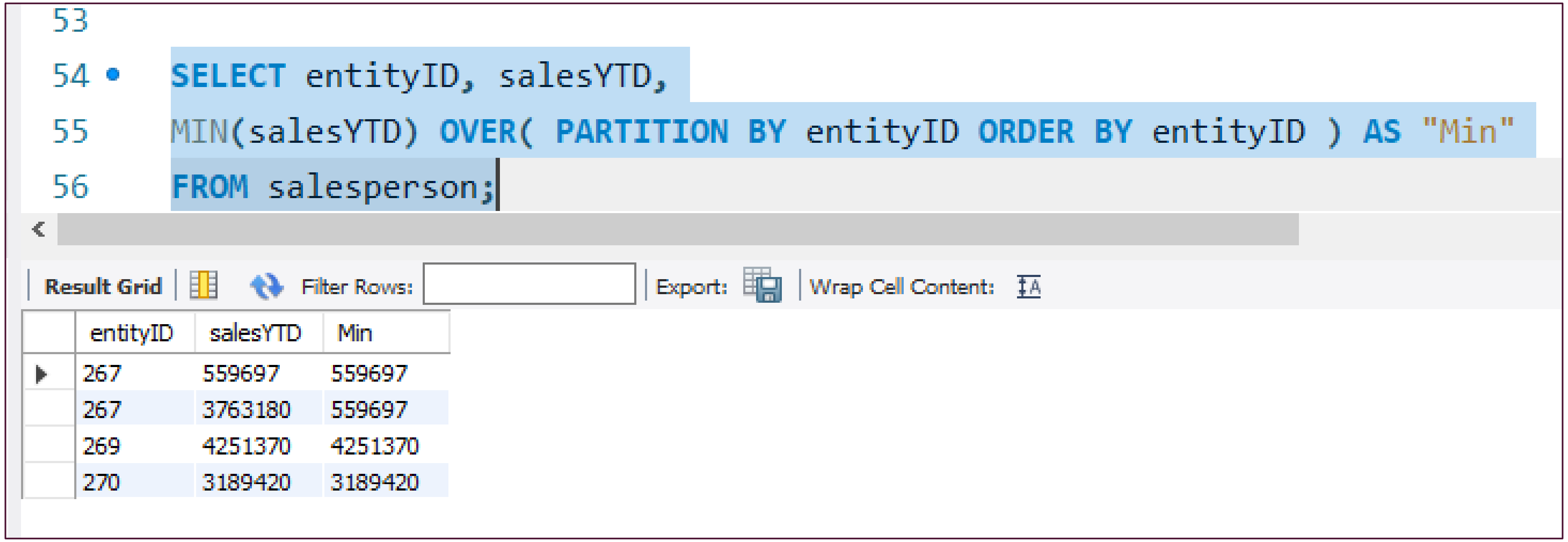
COUNT



```
SELECT entityID, salesYTD,
COUNT(salesYTD) OVER( PARTITION BY entityID ORDER BY entityID ) AS "Count"
FROM salesperson;
```

entityID	salesYTD	Count
267	559697	2
267	3763180	2
269	4251370	1
270	3189420	1

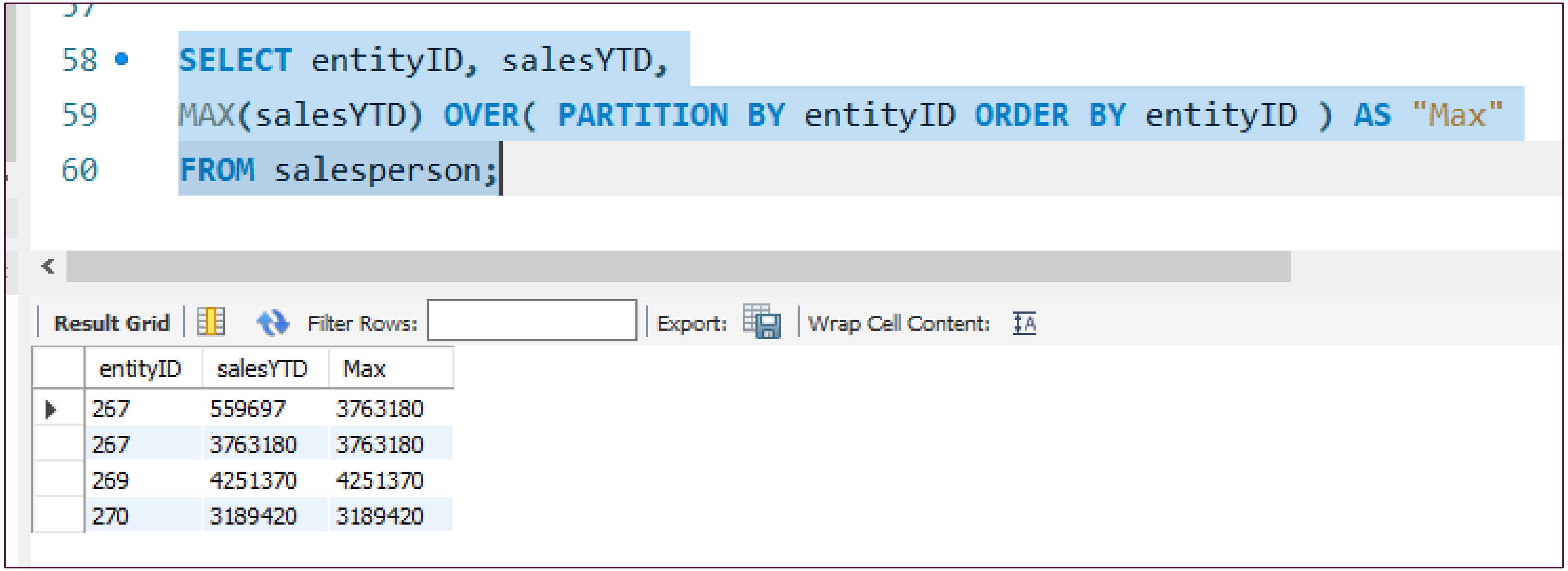
MIN



```sql
SELECT entityID, salesYTD,
MIN(salesYTD) OVER( PARTITION BY entityID ORDER BY entityID ) AS "Min"
FROM salesperson;
```

entityID	salesYTD	Min
267	559697	559697
267	3763180	559697
269	4251370	4251370
270	3189420	3189420

MAX



```sql
SELECT entityID, salesYTD,
MAX(salesYTD) OVER( PARTITION BY entityID ORDER BY entityID ) AS "Max"
FROM salesperson;
```

entityID	salesYTD	Max
267	559697	3763180
267	3763180	3763180
269	4251370	4251370
270	3189420	3189420

THANK YOU!!!

SQL Series Part 11

-Mayuri Dandekar