# Advanced SQL in Oracle and SQL Server

## Analytic Functions – Part II

Scott L. Hecht
http://www.sheepsqueezers.com
@sheepsqueezers

pluralsight
hardcore developer training

## Module Contents

- **Analytic Functions**
  - Data used in Module
  - ORDER BY Clause
  - Summary

## Data Used in Module

- **Table**
  - CHILDSTAT
- **Columns**
  - FIRSTNAME – child's first name
  - GENDER – child's gender (M=Male, F=Female)
  - BIRTHDATE – child's date of birth
  - HEIGHT – child's height (inches)
  - WEIGHT – child's weight (pounds)
- **Data**

| FIRSTNAME | GENDER | BIRTHDATE | HEIGHT | WEIGHT |
|---|---|---|---|---|
| LAUREN | F | 10-JUN-00 | 54 | 876 |
| ROSEMARY | F | 08-MAY-00 | 35 | 123 |
| ALBERT | M | 02-AUG-00 | 45 | 150 |
| BUDDY | M | 02-OCT-98 | 45 | 189 |
| FARQUAR | M | 05-NOV-98 | 76 | 198 |
| SIMON | M | 03-JAN-99 | 87 | 256 |
| TOMMY | M | 11-DEC-98 | 78 | 167 |

## ORDER BY Clause

- **What is the ORDER BY Clause?**
  - Imposes ordering on incoming data
  - Required by some analytic functions (ROW_NUMBER(), LEAD(), LAG(),…)
  - Does not make sense on others (COUNT(), MIN(), …)
- **Syntax**

```
function(…) OVER ( … ORDER BY col3,col4, … )
```

- **Functions**
  - ROW_NUMBER() – ever-increasing integral value
  - LISTAGG() – row data as a delimited text string
  - LEAD()/LAG() – peek forward and look back
  - FIRST_VALUE()/LAST_VALUE()/NTH_VALUE() – access first, last or nth row's data

# ROW_NUMBER() Function

- **ROW_NUMBER() Analytic Function**
  - Creates an ever-increasing integral value, starting at 1
  - Subsequent rows get next higher value
  - Can use with PARTITION BY
  - Resets to 1 when crossing partition boundary
  - Similar to Oracle's ROWNUM Pseudo-Column
  - Takes no parameter!
  - Availability:
    - Oracle: 8i
    - SQL Server: 2005
- **Syntax**

```
ROW_NUMBER() OVER ( … ORDER BY var1,var2, … )
```

## Example #7

- **Task: Create a column containing row numbers ordered by ascending name.**

```
SELECT A.*,
       ROW_NUMBER() OVER (ORDER BY A.FIRSTNAME) AS RNUM
 FROM CHILDSTAT A
```

```
FIRSTNAME  GENDER BIRTHDATE  HEIGHT  WEIGHT  RNUM
ALBERT     M      02-AUG-00  45      150     1
BUDDY      M      02-OCT-98  45      189     2
FARQUAR    M      05-NOV-98  76      198     3
LAUREN     F      10-JUN-00  54      876     4
ROSEMARY   F      08-MAY-00  35      123     5
SIMON      M      03-JAN-99  87      256     6
TOMMY      M      11-DEC-98  78      167     7
```

# Example #8

- **Task: Create a column containing row numbers within gender.**

```
SELECT A.*,
       ROW_NUMBER() OVER (PARTITION BY A.GENDER
                                  ORDER BY A.FIRSTNAME) AS RNUM
 FROM CHILDSTAT A
 ORDER BY A.GENDER,A.FIRSTNAME
```

```
FIRSTNAME  GENDER BIRTHDATE  HEIGHT  WEIGHT  RNUM
LAUREN     F      10-JUN-00  54      876     1
ROSEMARY   F      08-MAY-00  35      123     2
-------------------------------------------------
ALBERT     M      02-AUG-00  45      150     1
BUDDY      M      02-OCT-98  45      189     2
FARQUAR    M      05-NOV-98  76      198     3
SIMON      M      03-JAN-99  87      256     4
TOMMY      M      11-DEC-98  78      167     5
```

# LISTAGG() Function

- **LISTAGG() Analytic Function**
  - Concatenates values appearing in a single column
  - Returns a string of delimited values
  - Can sort the data within the column
  - Can be used in an aggregate sense
  - Availability:
    - Oracle: 11g/R2
    - SQL Server: N/A
- **Syntax**

```
LISTAGG(column-name,'delimiter')
  WITHIN GROUP (order-by-clause)
  OVER ( … )
```

  - WITHIN GROUP specifies data order
  - OVER indicates using LISTAGG() in an analytic sense
  - If omit OVER, using in an aggregate sense

## Example #9

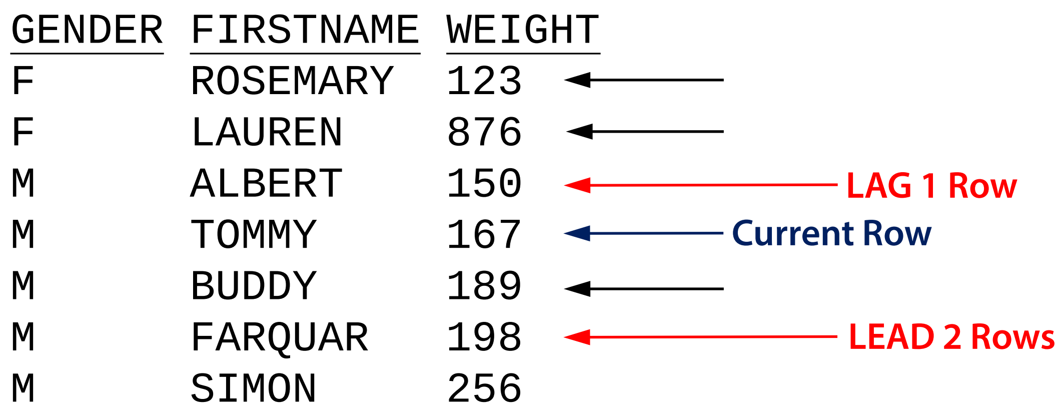
- **Task: Create a string of first names by gender ordered by descending weight.**

```
SELECT A.FIRSTNAME,A.GENDER,A.HEIGHT,A.WEIGHT,
       LISTAGG(A.FIRSTNAME,', ')
         WITHIN GROUP (ORDER BY A.WEIGHT DESC)
           OVER (PARTITION BY A.GENDER) AS NAMELIST
 FROM CHILDSTAT A
```

```
FIRSTNAME GENDER HEIGHT   WEIGHT   NAMELIST
LAUREN     F      54       876      LAUREN, ROSEMARY
ROSEMARY   F      35       123      LAUREN, ROSEMARY
------------------------------------------------------------------
SIMON      M      87       256      SIMON, FARQUAR, BUDDY, TOMMY, ALBERT
FARQUAR    M      76       198      SIMON, FARQUAR, BUDDY, TOMMY, ALBERT
BUDDY      M      45       189      SIMON, FARQUAR, BUDDY, TOMMY, ALBERT
TOMMY      M      78       167      SIMON, FARQUAR, BUDDY, TOMMY, ALBERT
ALBERT     M      45       150      SIMON, FARQUAR, BUDDY, TOMMY, ALBERT
```

# LEAD()/LAG() Functions

- **LEAD()/LAG() Analytic Functions**
  - LEAD() – peek forward a number of rows
  - LAG() – look back a number of rows
  - Have access to that row's data
  - Based off of current row!
  - Availability:
    - Oracle: 8i
    - SQL Server: 2012



```
GENDER FIRSTNAME WEIGHT
------ --------- ------
F      ROSEMARY  123
F      LAUREN    876
M      ALBERT    150
M      TOMMY     167
M      BUDDY     189
M      FARQUAR   198
M      SIMON     256
```

## LEAD()/LAG() Functions

- **Syntax**

```
LEAD(column-name,nbr-rows-to-lead,def-value) OVER (…)
LAG(column-name,nbr-rows-to-lag,def-value) OVER (…)
```

- **ORDER BY Clause required**
- **PARTITION BY Clause not required**
- ***def-value* returned if *nbr-rows-to-lead* or *nbr-rows-to-lag*:**
  - crosses partition boundary
  - goes off top of table
  - goes off bottom of table
- ***column-name* does not have to appear in the ORDER BY Clause**

## Example #10

- **Task: Create two additional columns using the weight:**
  - the next heaviest weight
  - the previous lightest weight

```
SELECT A.FIRSTNAME,A.WEIGHT,
       LEAD(A.WEIGHT,1,-1) OVER (ORDER BY A.WEIGHT) AS LEAD_1_WT,
       LAG(A.WEIGHT,2,-1)  OVER (ORDER BY A.WEIGHT) AS LAG_2_WT
 FROM CHILDSTAT A
 ORDER BY A.WEIGHT
```

```
FIRSTNAME WEIGHT   LEAD_1_WT  LAG_2_WT
ROSEMARY  123      150        -1
ALBERT    150      167        -1
TOMMY     167      189        123
BUDDY     189      198        150
FARQUAR   198      256        167
SIMON     256      876        189
LAUREN    876      -1         198
```

# Example #11

- **Task: Create two columns using the weight within gender:**
  - the next heaviest weight
  - the previous lightest weight

```
SELECT A.FIRSTNAME,A.GENDER,A.WEIGHT,
       LEAD(A.WEIGHT,1,-1) OVER (PARTITION BY A.GENDER
                                 ORDER BY A.WEIGHT) AS LEAD_1_WT,
       LAG(A.WEIGHT,2,-1) OVER (PARTITION BY A.GENDER
                                ORDER BY A.WEIGHT) AS LAG_2_WT
 FROM CHILDSTAT A
 ORDER BY A.GENDER,A.WEIGHT
```

```
FIRSTNAME GENDER WEIGHT  LEAD_1_WT  LAG_2_WT
ROSEMARY  F      123     876        -1
LAUREN    F      876     -1         -1
--------------------------------------------
ALBERT    M      150     167        -1
TOMMY     M      167     189        -1
BUDDY     M      189     198        150
FARQUAR   M      198     256        167
SIMON     M      256     -1         189
```

## RANK()/DENSE_RANK() Functions

- **RANK()/DENSE_RANK() Analytic Functions**
  - Provide ranks based on ORDER  BY column
  - Recall that runners finish a race ranked as 1, 2, 3,…
  - Tied for first?
    - RANK() returns 1, 1, 3, 4,…
    - DENSE_RANK() returns 1, 1, 2, 3,…

## RANK()/DENSE_RANK() Functions

- **RANK()/DENSE_RANK() Analytic Functions**
- **Syntax**

```
RANK() OVER ( … ORDER BY … )
DENSE_RANK() OVER ( … ORDER BY … )
```

  - No arguments required
  - ORDER BY Clause required
  - PARTITION BY Clause not required
  - Availability:
    - Oracle: 8i
    - SQL Server: 2005

# Example #12

- **Task: Create ranks using ascending height within gender.**

```
SELECT A.FIRSTNAME,A.GENDER,A.HEIGHT,
       RANK() OVER (PARTITION BY A.GENDER
                    ORDER BY A.HEIGHT) AS HT_RANK,
       DENSE_RANK() OVER (PARTITION BY A.GENDER
                          ORDER BY A.HEIGHT) AS HT_DENSERANK
 FROM CHILDSTAT A
 ORDER BY A.GENDER,A.HEIGHT
```

```
FIRSTNAME GENDER HEIGHT HT_RANK  HT_DENSERANK
ROSEMARY  F      35     1        1
LAUREN    F      54     2        2
---------------------------------------------
ALBERT    M      45     1        1
BUDDY     M      45     1        1
FARQUAR   M      76     3        2
TOMMY     M      78     4        3
SIMON     M      87     5        4
```

## FIRST_VALUE()/LAST_VALUE() Functions

- FIRST_VALUE()/LAST_VALUE() Analytic Functions
  - Retrieves the first or last value within a column
  - Takes a column name as the sole parameter
  - Honor partition boundaries!
- Syntax

```
FIRST_VALUE(column-name) OVER ( … ORDER BY … )
LAST_VALUE(column-name) OVER ( … ORDER BY … )
```

  - ORDER BY clause required
  - PARTITION BY clause not required
  - Availability:
    - Oracle: 8i
    - SQL Server: 2012

# Example #13

- **Task: Retrieve names of the heaviest/lightest male/female child.**

```
SELECT A.FIRSTNAME,A.GENDER,A.WEIGHT,
       FIRST_VALUE(A.FIRSTNAME) OVER (PARTITION BY A.GENDER
                                      ORDER BY A.WEIGHT) AS LT_CHILD,
       LAST_VALUE(A.FIRSTNAME) OVER (PARTITION BY A.GENDER
                                     ORDER BY A.WEIGHT) AS HV_CHILD
 FROM CHILDSTAT A
 ORDER BY A.GENDER,A.WEIGHT
```

| FIRSTNAME | GENDER | WEIGHT | LT_CHILD | HV_CHILD |
|---|---|---|---|---|
| ROSEMARY | F | 123 | ROSEMARY | ROSEMARY |
| LAUREN | F | 876 | ROSEMARY | LAUREN |
| ALBERT | M | 150 | ALBERT | ALBERT |
| TOMMY | M | 167 | ALBERT | TOMMY |
| BUDDY | M | 189 | ALBERT | BUDDY |
| FARQUAR | M | 198 | ALBERT | FARQUAR |
| SIMON | M | 256 | ALBERT | SIMON |

**The HV_CHILD column is *incorrect*. This brings us the *Window Clause*.**

## Summary

- **ORDER BY Clause gives order to your data**
- **Can be used with the PARTITION BY Clause**
- **ROW_NUMBER(), LEAD(), LAG(), etc.**
- ***…but something was wrong with that last example…***