Advanced Aggregate Functions in SQL

David Simpson
dsimpson@themisinc.com

Themis Education

- Most complete DB2 Curriculum in the industry
- Offerings include a complete mainframe curriculum in addition to Oracle, Java, .NET, Linux, UNIX, etc.
- Training Venues:
  - At your facility for groups
  - Public enrollment at various locations in the USA
  - Distance learning public enrollment with live instructors.
- Webinars:
  - Visit www.themisinc.com/webinars for upcoming schedule and replays of past webinars and to download today's slides.
- This material comes from our Db2 12 migration course
  - View Outline and Register

Themis

David Simpson

David Simpson is currently the Vice President of Themis Inc. He teaches courses on SQL, Application Programming, Database Administration as well as optimization, performance and tuning. He also installs and maintains the database systems used for training at Themis and works with our network of instructors to deliver high quality training solutions to our customers worldwide.

Since 1993 David has worked as a developer and DBA in support of very large transactional and business intelligence systems. David is a certified DB2 DBA on both z/OS and LUW. David was voted Best User Speaker and Best Overall Speaker at IDUG North America 2006. He was also voted Best User Speaker at IDUG Europe 2006 and is a member of the IDUG Speakers Hall of Fame. David is also an IBM Gold Consultant.

dsimpson@themisinc.com
www.themisinc.com
@ThemisDave
@ThemisTraining

Find my work:
www.themisinc.com/webinars
www.idug.org/content

Themis

Agenda

- OLAP Functions
- OLAP Specifications on Aggregate Functions
- The LEAD and LAG Functions

Rank Function with Partitioning

```
SELECT DEPTNO, LASTNAME, SALARY,
       RANK() OVER(PARTITION BY DEPTNO
                   ORDER BY SALARY DESC) AS R
FROM EMP
ORDER BY DEPTNO, SALARY DESC
```

DEPTNO	LASTNAME	SALARY	R
A00	HAAS	52750.00	1
A00	MOON	52750.00	1
A00	LUCCHESI	46500.00	3
A00	O'CONNEL	29250.00	4
B01	THOMPSON	41250.00	1
C01	KWAN	38250.00	1
C01	NICHOLS	28420.00	2
C01	QUINTANA	23800.00	3
…	…	…	…

Dense Rank Function

```
SELECT LASTNAME, SALARY,
  DENSE_RANK() OVER(ORDER BY SALARY DESC) AS R
FROM EMP
ORDER BY SALARY DESC
```

LASTNAME	SALARY	R
HAAS	52750.00	1
MOON	52750.00	1
LUCCHESI	46500.00	2
THOMPSON	41250.00	3
GEYER	40175.00	4
KWAN	38250.00	5
PULASKI	36170.00	6
...	...	...

Themis

Row Numbering

```
SELECT LASTNAME, SALARY,
  ROW_NUMBER() OVER(ORDER BY SALARY DESC) AS R
FROM EMP
ORDER BY SALARY DESC
```

LASTNAME	SALARY	R
HAAS	52750.00	1
MOON	52750.00	2
LUCCHESI	46500.00	3
THOMPSON	41250.00	4
GEYER	40175.00	5
KWAN	38250.00	6
PULASKI	36170.00	7
...	...	...

More Complex ORDER BY

```
SELECT LASTNAME, SALARY,
  ROW_NUMBER()
  OVER(ORDER BY SALARY DESC, BONUS DESC) AS R
FROM EMP
ORDER BY SALARY DESC, BONUS DESC
```

LASTNAME	SALARY	BONUS	R
HAAS	52750.00	1000.00	1
MOON	52750.00	900.00	2
LUCCHESI	46500.00	900.00	3
THOMPSON	41250.00	800.00	4
GEYER	40175.00	800.00	5
KWAN	38250.00	800.00	6
PULASKI	36170.00	700.00	7
...	...	...	...

Themis

Moving Aggregates

- Allows for a “running” total, average, etc
- Works with aggregate or “column” functions
  - AVG
  - SUM
  - MIN
  - MAX
- ORDER BY in the OVER should match the final for the statement

Moving Sum / Avg

```
SELECT EMPNO, DEPTNO, SALARY,
   SUM (SALARY)
        OVER (ORDER BY SALARY ASC) AS SUM_SAL
FROM EMP
ORDER BY SALARY ASC
```

EMPNO	DEPTNO	SALARY	SUM_SAL
000290	E11	15340.00	15340.00
000310	E11	15900.00	31240.00
000260	D21	17250.00	48490.00
000300	E11	17750.00	66240.00
000210	D11	18270.00	84510.00
000250	D21	19180.00	103690.00
000320	E21	19950.00	123640.00
000190	D11	20450.00	144090.00
000180	D11	21340.00	165430.00
000230	D21	22180.00	187610.00
…	…	…	…

Moving Sum / Avg

```
SELECT DEPTNO, EMPNO, SALARY,
   SUM  (SALARY)
            OVER (PARTITION BY DEPTNO
                  ORDER BY SALARY ASC) AS SUM_SAL
FROM EMP
ORDER BY DEPTNO, SALARY
```

DEPTNO	EMPNO	SALARY	SUM_SAL
A00	000120	29250.00	29250.00
A00	000110	46500.00	75750.00
A00	000010	52750.00	128500.00
A00	000011	52750.00	181250.00
B01	000020	41250.00	41250.00
C01	000130	23800.00	23800.00
C01	000140	28420.00	52220.00
C01	000030	38250.00	90470.00
…	…	…	…

SUM resets for each new value of partition clause

Moving Sum / Avg

```
SELECT DEPTNO, EMPNO, SALARY,
  AVG(SALARY)
  OVER (PARTITION BY DEPTNO
       ORDER BY SALARY ASC
       ROWS 2 PRECEDING AND CURRENT ROW
       ) AS AVG_SAL
FROM EMP
ORDER BY DEPTNO,
                SALARY
```

You control the window to be summarized

This topic is covered in an article on the IDUG Content Blog:
https://www.idug.org/p/bl/ar/blogaid=847

Themis

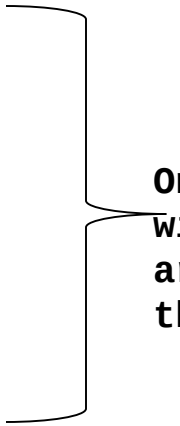
Windowing by RANGE

```
SELECT DEPTNO, EMPNO, SALARY,
  SUM  (SALARY)
          OVER (ORDER BY SALARY ASC
                RANGE BETWEEN 5000 PRECEEDING
                                  AND CURRENT ROW
                ) AS SUM_SAL
FROM EMP
ORDER BY DEPTNO
```

Themis

Result

DEPTNO	EMPNO	SALARY	SUM_SAL
E11	000290	15340.00	15340.00
E11	000310	15900.00	31240.00
D21	000260	17250.00	48490.00
E11	000300	17750.00	66240.00
D11	000210	18270.00	84510.00
D21	000250	19180.00	103690.00
E21	000320	19950.00	123640.00
D11	000190	20450.00	128750.00
D11	000180	21340.00	134190.00
D21	000230	22180.00	156370.00
…	…	…	…



Only values within $5000 are summed with the current row

Themis

A Short Case Study

The Data:

CUST_ID	START_DATE	IMPORTANT DATA
1	2017-01-01	abcdefg
1	2018-01-01	hijklmn
1	2019-01-01	opqrstu
2	2017-01-01	abcdefg
2	2017-04-10	vwxyzab
2	2019-01-01	zzzzzzz

The Desired Result:

CUST_ID	START_DATE	END_DATE	IMPORTANT DATA
1	2017-01-01	2018-01-01	abcdefg
1	2018-01-01	2019-01-01	hijklmn
1	2019-01-01	NULL	opqrstu
2	2017-01-01	2017-04-10	abcdefg
2	2017-04-10	2019-01-01	vwxyzab
2	2019-01-01	NULL	zzzzzzz

The SQL

```
SELECT CUST_ID
     , START_DATE
     , LEAD(START_DATE)
            OVER(PARTITION BY CUST_ID
            ORDER BY START_DATE ASC) AS END_DATE
     , IMPORTANT_DATA
  FROM IMPORTANT_THINGS
ORDER BY CUST_ID, START_DATE;
```

Take the value from the *next* row

The LAG function would pull the value from the *previous* row

Themis

The SQL (Db2 for z/OS)

```
SELECT CUST_ID
     , START_DATE
     , MIN(START_DATE)
        OVER(PARTITION BY CUST_ID
        ORDER BY START_DATE ASC
        ROWS BETWEEN 1 FOLLOWING AND 1 FOLLOWING)
            AS END_DATE
     , IMPORTANT_DATA
  FROM IMPORTANT_THINGS
ORDER BY CUST_ID, START_DATE;
```

Themis

David Simpson

Themis Training

dsimpson@themisinc.com

@ThemisDave

@ThemisTraining

Themis