SQL Expressive Powers

1. Relational Algebra or Calculus
2. Aggregation / Grouping
3. Deductive Logics / **Analytic Functions (Windowing)**
4. Data Mining Features

Analytic Functions

(Read Section 25.3.2 Window Query in SQL:1999)

2. Optional Reading: Oracle Data Warehousing Guide

Ch 19. SQL for Analysis and Report in Data Warehouses

https://docs.oracle.com/en/database/oracle/oracle-database/19/dwhsg/sql-analysis-reporting-data-warehouses.html#GUID-20EFBF1E-F79D-4E4A-906C-6E496EECA684

Comparison and Reporting!!

The Coolest SQL Thing Ever Since the SELECT Keyword

(Academic root: http://research.cs.wisc.edu/SEQ/)

Standardized as OLAP Amendment

Pitfalls in SQL: Comparisons

- Examples:
  - last year’s sales vs. this year’s sales for each product
    - ✓ requires a self-join

```
VIEW:
create or replace view v_sales as
select prod-id, year, sum(qty) as sale_sum
from sales
group by prod-id, year;

QUERY:
select cur.year cur_year, cur.sale_cur_sales, last.sum last_sales
from v_sales curr, v_sales last
where curr.year=(last.year+1)
```

Pitfalls in SQL: Reporting Features

- It was too complex to express
  - rank(top 10) and N_tile(“top 30%” of all products)
  - median, mode, …
  - running total, moving average, cumulative totals
- Examples:
  - a moving average (over 3 day window) of total sales for each product

```
VIEW: create or replace view v_sales as
      select prod-id, time-id, sum(qty) as sale_sum
      from sales
      group by prod-id, time-id;

QUERY: select end.time, avg(start.sale_sum)
       from v_sales start, v_sales end
       where end.time-id >= start.time-id and end.time-id <= start.time-id +2
       group by end.time
```

Why Analytic Functions?
Answer Important Business Questions

- Who are the top 10 salesmen in each region?
  - Rank
- What is the 90-day moving average of sales volume?
  - Moving window
- What is the percentage growth of Jan-00 sales over Jan-99?
  - Period-over-period comparison
- What are January’s sales as a percentage of the total annual sales?
  - Compare aggregates on different levels

Why Analytic Functions? (2)
Answer Important Business Questions

- Technical Benefits of analytical functions
  - Improved query performance
  - Enhanced developer productivity
  - Minimized learning effort
  - Standardized syntax
    - ✓ In particular, good for ISVs (Independent Software Vendors)

- "OLAP amendment" to the SQL Standard in 2000
  - Extremely fast adoption by ANSI due to its high value

- Many new analytic features have been added over the time into Oracle (since Oracle 8i until 12c)

Families of Analytic Functions

- Ranking functions
  - rank, dense_rank, cume_dist, percent_rank, ntile
- Window Aggregate functions (moving and cumulative)
  - avg, sum, min, max, count, variance, stddev, first_value, last_value
- LAG/LEAD functions
  - Direct inter-row reference using offsets
- Reporting Aggregate functions
  - sum, avg, min, max, variance, stddev, count, ratio_to_report
- *Statistical Aggregates*
  - *correlation, linear regression family, covariance*

(Many new features since Oracle 9i)

- Inverse Percentile
- FIRST/LAST Aggregates
- Hypothetical Rank and Distribution
- …..

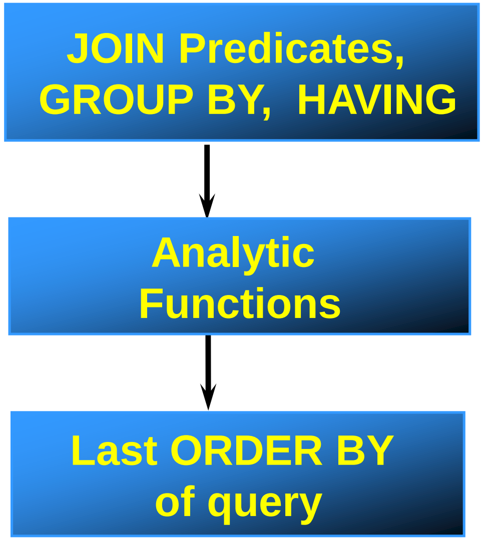
Key Concepts

Consistent across all analytic functions

- Second-Pass Grouping
  - Apply to results of joins, GROUP BY and HAVING
- Partitions
  - Grouping of rows within a query result set
  - Each function in a query defines its partitions
- Ordering
  - Rows may be ordered within a partition
  - Each function in a query defines its ordering
- Window
  - Moving group of rows within a partition
  - Determine the rows used in calculations
  - Great flexibility in setting window boundaries

Second-Pass Grouping

Apply to results of joins, GROUP BY, HAVING



```
SELECT job, deptno, sum(sal) s1,                              (2)
            RANK() OVER (PARTITION BY job
            ORDER BY SUM(sal) DESC) s2                        (3)
FROM emp
WHERE sal < 5000
GROUP BY job, deptno
HAVING sum(sal) > 1000                                        (1)
ORDER BY job, s2                                              (4)
```

Basic Syntax

<FUNCTION> (<argument>) OVER (<analytic clauses>)

- 3 analytic clauses
  1. Partitioning
  2. Ordering
  3. Windowing

Ranking - Order By

```
SELECT EMPNO, ENAME, SAL,
          RANK() OVER( ORDER BY SAL DESC) AS RANK
FROM   EMP
```

```
 EMPNO ENAME             SAL       RANK
------ ---------- ---------- ----------
  7839 KING             5000          1
  7788 SCOTT            3000          2
  7902 FORD             3000          2
  7566 JONES            2975          4
  7698 BLAKE            2850          5
  7782 CLARK            2450          6
  7499 ALLEN            1600          7
  7844 TURNER           1500          8
  7934 MILLER           1300          9
  7521 WARD             1250         10
  7654 MARTIN           1250         10
  7876 ADAMS            1100         12
  7900 JAMES             950         13
  7369 SMITH             800         14
```

Ranking - Order By

- When subquery is allowed in SELECT clause, how can we express the rank query?

```
SELECT empno, sal,  ( SELECT count(*)
                      FROM emp e2
                     WHERE e2.sal >= e1.sal) as rank
FROM emp e1;
```

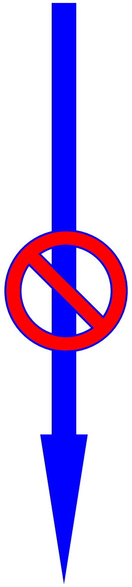
  - Check the internal "how"s (i.e. exec. Plan) for these two queries by turning on "set autotrace on" in SQL*Plus?
  - What about "Partition by Order By"?

- How can we express the rank queries without subquery and analytic function?

Order By

```
SELECT EMPNO, ENAME, SAL,
          RANK() OVER( ORDER BY SAL DESC) AS RANK
FROM   EMP
ORDER BY EMPNO
```

```
EMPNO ENAME             SAL        RANK
----- ---------- ---------- ----------
 7369 SMITH             800         14
 7499 ALLEN            1600          7
 7521 WARD             1250         10
 7566 JONES            2975          4
 7654 MARTIN           1250         10
 7698 BLAKE            2850          5
 7782 CLARK            2450          6
 7788 SCOTT            3000          2
 7839 KING             5000          1
 7844 TURNER           1500          8
 7876 ADAMS            1100         12
 7900 JAMES             950         13
 7902 FORD             3000          2
 7934 MILLER           1300          9
```



Partition By

```
SELECT DEPTNO, EMPNO, ENAME, SAL,
     RANK() OVER( PARTITION BY DEPTNO ORDER BY SAL DESC)
     AS DEPT_RANK
FROM   EMP
ORDER BY DEPTNO
```

```
DEPTNO      EMPNO ENAME             SAL  DEPT_RANK
------ ---------- ---------- ---------- ----------
    10       7839 KING             5000          1
    10       7782 CLARK            2450          2
    10       7934 MILLER           1300          3
    20       7788 SCOTT            3000          1
    20       7902 FORD             3000          1
    20       7566 JONES            2975          3
    20       7876 ADAMS            1100          4
    20       7369 SMITH             800          5
    30       7698 BLAKE            2850          1
    30       7499 ALLEN            1600          2
    30       7844 TURNER           1500          3
    30       7521 WARD             1250          4
    30       7654 MARTIN           1250          4
    30       7900 JAMES             950          6
```

Multiple Analytic Functions

```
SELECT EMPNO, ENAME, SAL, RANK() OVER(ORDER BY SAL DESC) AS SAL_RANK,
                    HIREDATE, RANK() OVER(ORDER BY HIREDATE) AS HIRE_RANK
FROM   EMP
ORDER BY ENAME
```

```
EMPNO ENAME             SAL   SAL_RANK HIREDATE  HIRE_RANK
----- ---------- ---------- ---------- -------- ----------
 7876 ADAMS            1100         12 87/05/23         14
 7499 ALLEN            1600          7 81/02/20          2
 7698 BLAKE            2850          5 81/05/01          5
 7782 CLARK            2450          6 81/06/09          6
 7902 FORD             3000          2 81/12/03         10
 7900 JAMES             950         13 81/12/03         10
 7566 JONES            2975          4 81/04/02          4
 7839 KING             5000          1 81/11/17          9
 7654 MARTIN           1250         10 81/09/28          8
 7934 MILLER           1300          9 82/01/23         12
 7788 SCOTT            3000          2 87/04/19         13
 7369 SMITH             800         14 80/12/17          1
 7844 TURNER           1500          8 81/09/08          7
 7521 WARD             1250         10 81/02/22          3
```

Top-N?

```
SELECT EMPNO, ENAME, SAL
FROM   EMP
WHERE RANK() OVER( ORDER BY SAL DESC) AS RANK <= 10
```

```
   *
3행에 오류:
ORA-30483: 윈도우 함수를 여기에 사용할 수 없습니다
```

Top-N? Work Around

```
SELECT *
FROM   (SELECT EMPNO, ENAME, SAL,
                    RANK() OVER( ORDER BY SAL DESC) AS RANK
          FROM EMP)
WHERE RANK <= 10;
```

```
 EMPNO ENAME             SAL       RANK
------ ---------- ---------- ----------
  7839 KING             5000          1
  7788 SCOTT            3000          2
  7902 FORD             3000          2
  7566 JONES            2975          4
  7698 BLAKE            2850          5
  7782 CLARK            2450          6
  7499 ALLEN            1600          7
  7844 TURNER           1500          8
  7934 MILLER           1300          9
  7521 WARD             1250         10
  7654 MARTIN           1250         10
```

What about Tie Value?

```
SELECT EMPNO, ENAME, SAL,
            RANK()       OVER(ORDER BY SAL DESC) AS RANK,
            DENSE_RANK() OVER(ORDER BY SAL DESC) AS DENSE_RANK,
            ROW_NUMBER() OVER(ORDER BY SAL DESC) AS ROW_NUMBER
FROM   EMP;
```

```
EMPNO ENAME             SAL       RANK DENSE_RANK ROW_NUMBER
----- ---------- ---------- ---------- ---------- ----------
 7839 KING             5000          1          1          1
 7788 SCOTT            3000          2          2          2
 7902 FORD             3000          2          2          3
 7566 JONES            2975          4          3          4
 7698 BLAKE            2850          5          4          5
 7782 CLARK            2450          6          5          6
 7499 ALLEN            1600          7          6          7
 7844 TURNER           1500          8          7          8
 7934 MILLER           1300          9          8          9
 7521 WARD             1250         10          9         10
 7654 MARTIN           1250         10          9         11
 7876 ADAMS            1100         12         10         12
 7900 JAMES             950         13         11         13
 7369 SMITH             800         14         12         14
```

What about rownum and row_number()?

```
SELECT EMPNO, ENAME, SAL, ROWNUM,
           ROW_NUMBER() OVER(ORDER BY SAL DESC) AS ROW_NUMBER
FROM   EMP
ORDER BY ENAME;  /* What about EMPNO ? */
```

```
 EMPNO ENAME             SAL     ROWNUM ROW_NUMBER
------ ---------- ---------- ---------- ----------
  7876 ADAMS            1100         11         12
  7499 ALLEN            1600          2          7
  7698 BLAKE            2850          6          5
  7782 CLARK            2450          7          6
  7902 FORD             3000         13          3
  7900 JAMES             950         12         13
  7566 JONES            2975          4          4
  7839 KING             5000          9          1
  7654 MARTIN           1250          5         11
  7934 MILLER           1300         14          9
  7788 SCOTT            3000          8          2
  7369 SMITH             800          1         14
  7844 TURNER           1500         10          8
  7521 WARD             1250          3         10
```

** ROWNUM assigned
as rows retrieved
before final ORDER BY

NULL sorts as the highest value

```
SELECT EMPNO, ENAME, COMM,
          RANK()      OVER(ORDER BY COMM DESC) AS RANK,
          DENSE_RANK() OVER(ORDER BY COMM DESC) AS DENSE_RANK,
          ROW_NUMBER() OVER(ORDER BY COMM DESC) AS ROW_NUMBER
FROM   EMP
```

```
EMPNO ENAME            COMM       RANK DENSE_RANK ROW_NUMBER
----- ---------- ---------- ---------- ---------- ----------
 7369 SMITH                          1          1          1
 7566 JONES                          1          1          2
 7782 CLARK                          1          1          3
 7698 BLAKE                          1          1          4
 7788 SCOTT                          1          1          5
 7839 KING                           1          1          6
 7900 JAMES                          1          1          7
 7934 MILLER                         1          1          8
 7902 FORD                           1          1          9
 7876 ADAMS                          1          1         10
 7654 MARTIN           1400         11          2         11
 7521 WARD              500         12          3         12
 7499 ALLEN             300         13          4         13
 7844 TURNER              0         14          5         14
```

NULLS FIRST vs. NULLS LAST

```
SELECT EMPNO, ENAME, COMM,
         RANK()      OVER(ORDER BY COMM DESC NULLS LAST) AS RANK,
         DENSE_RANK() OVER(ORDER BY COMM DESC NULLS LAST) AS DENSE_RANK,
         ROW_NUMBER() OVER(ORDER BY COMM DESC NULLS LAST) AS ROW_NUMBER
FROM   EMP
```

```
EMPNO ENAME            COMM       RANK DENSE_RANK ROW_NUMBER
----- ---------- ---------- ---------- ---------- ----------
 7654 MARTIN           1400          1          1          1
 7521 WARD              500          2          2          2
 7499 ALLEN             300          3          3          3
 7844 TURNER              0          4          4          4
 7369 SMITH                          5          5          5
 7566 JONES                          5          5          6
 7900 JAMES                          5          5          7
 7934 MILLER                         5          5          8
 7902 FORD                           5          5          9
 7876 ADAMS                          5          5         10
 7698 BLAKE                          5          5         11
 7782 CLARK                          5          5         12
 7788 SCOTT                          5          5         13
 7839 KING                           5          5         14
```

Partition By

```
SELECT DEPTNO, EMPNO, ENAME, SAL,
    RANK() OVER( PARTITION BY DEPTNO ORDER BY SAL DESC)
    AS DEPT_RANK
FROM   EMP
```

```
DEPTNO      EMPNO ENAME             SAL  DEPT_RANK
------ ---------- ---------- ---------- ----------
    10       7839 KING             5000          1
    10       7782 CLARK            2450          2
    10       7934 MILLER           1300          3
    20       7788 SCOTT            3000          1
    20       7902 FORD             3000          1
    20       7566 JONES            2975          3
    20       7876 ADAMS            1100          4
    20       7369 SMITH             800          5
    30       7698 BLAKE            2850          1
    30       7499 ALLEN            1600          2
    30       7844 TURNER           1500          3
    30       7521 WARD             1250          4
    30       7654 MARTIN           1250          4
    30       7900 JAMES             950          6
```

Partition by Multiple Columns

```
SELECT DEPTNO, JOB, EMPNO, ENAME, SAL,
    RANK() OVER( PARTITION BY DEPTNO, JOB ORDER BY SAL DESC)
    AS DEPT_RANK
FROM   EMP;
```

```
DEPTNO JOB            EMPNO ENAME             SAL  DEPT_RANK
------ --------- ---------- ---------- ---------- ----------
    10 CLERK           7934 MILLER           1300          1
    10 MANAGER         7782 CLARK            2450          1
    10 PRESIDENT       7839 KING             5000          1
    20 ANALYST         7788 SCOTT            3000          1
    20 ANALYST         7902 FORD             3000          1
    20 CLERK           7876 ADAMS            1100          1
    20 CLERK           7369 SMITH             800          2
    20 MANAGER         7566 JONES            2975          1
    30 CLERK           7900 JAMES             950          1
    30 MANAGER         7698 BLAKE            2850          1
    30 SALESMAN        7499 ALLEN            1600          1
    30 SALESMAN        7844 TURNER           1500          2
    30 SALESMAN        7521 WARD             1250          3
    30 SALESMAN        7654 MARTIN           1250          3
```

Multiple Functions with different Partitionings

```
SELECT DEPTNO, JOB, EMPNO, ENAME, SAL,
          RANK() OVER( PARTITION BY DEPTNO ORDER BY SAL DESC) AS DEPT_RANK,
          RANK() OVER( PARTITION BY JOB ORDER BY SAL DESC)    AS JOB_RANK,
          RANK() OVER(ORDER BY SAL DESC)                      AS TOTAL_RANK
FROM   EMP;
```

```
 DEPTNO JOB            EMPNO ENAME             SAL  DEPT_RANK   JOB_RANK TOTAL_RANK
------- --------- ---------- ---------- ---------- ---------- ---------- ----------
     10 PRESIDENT       7839 KING             5000          1          1          1
     20 ANALYST         7788 SCOTT            3000          1          1          2
     20 ANALYST         7902 FORD             3000          1          1          2
     20 MANAGER         7566 JONES            2975          3          1          4
     30 MANAGER         7698 BLAKE            2850          1          2          5
     10 MANAGER         7782 CLARK            2450          2          3          6
     30 SALESMAN        7499 ALLEN            1600          2          1          7
     30 SALESMAN        7844 TURNER           1500          3          2          8
     10 CLERK           7934 MILLER           1300          3          1          9
     30 SALESMAN        7521 WARD             1250          4          3         10
     30 SALESMAN        7654 MARTIN           1250          4          3         10
     20 CLERK           7876 ADAMS            1100          4          2         12
     30 CLERK           7900 JAMES             950          6          3         13
     20 CLERK           7369 SMITH             800          5          4         14
```

Aggregates

```
SELECT DEPTNO, AVG(SAL),
     RANK() OVER(ORDER BY AVG(SAL) DESC) AS DEPT_RANK
FROM   EMP
GROUP BY DEPTNO
```

```
DEPTNO    AVG(SAL)  DEPT_RANK
------ ---------- ----------
    10 2916.66667          1
    20       2175          2
    30 1566.66667          3
```

** NOTE:
Analytic Functions
after Group By

Aggregates + Partitioning

```
SELECT DEPTNO, JOB, AVG(SAL),
     RANK() OVER(PARTITION BY DEPTNO
                          ORDER BY AVG(SAL) DESC) AS DEPT_RANK
FROM   EMP
GROUP BY DEPTNO, JOB
```

```
 DEPTNO JOB         AVG(SAL)  DEPT_RANK
------- --------- ---------- ----------
     10 PRESIDENT       5000          1
     10 MANAGER         2450          2
     10 CLERK           1300          3
     20 ANALYST         3000          1
     20 MANAGER         2975          2
     20 CLERK            950          3
     30 MANAGER         2850          1
     30 SALESMAN        1400          2
     30 CLERK            950          3
```

Aggregates as Analytic Functions

```
SELECT EMPNO, ENAME, SAL,
         ROUND( AVG(SAL) OVER() ) AS AVGSAL,
         SAL - ROUND( AVG(SAL) OVER() ) AS DIFF
FROM   EMP
```

```
 EMPNO ENAME             SAL     AVGSAL       DIFF
------ ---------- ---------- ---------- ----------
  7369 SMITH             800       2073      -1273
  7499 ALLEN            1600       2073       -473
  7521 WARD             1250       2073       -823
  7566 JONES            2975       2073        902
  7654 MARTIN           1250       2073       -823
  7698 BLAKE            2850       2073        777
  7782 CLARK            2450       2073        377
  7788 SCOTT            3000       2073        927
  7839 KING             5000       2073       2927
  7844 TURNER           1500       2073       -573
  7876 ADAMS            1100       2073       -973
  7900 JAMES             950       2073      -1123
  7902 FORD             3000       2073        927
  7934 MILLER           1300       2073       -773
```

Aggregates as Analytic Functions + Partitions

```
SELECT DEPTNO, EMPNO, ENAME, SAL,
        ROUND( AVG(SAL) OVER(PARTITION BY DEPTNO) ) AS AVGSAL,
        SAL - ROUND( AVG(SAL) OVER(PARTITION BY DEPTNO) ) AS DIFF
FROM   EMP
```

** NOTE:
OVER() or
OVER(PARTITION BY)

** WE will visit this issue later

```
DEPTNO      EMPNO ENAME             SAL     AVGSAL       DIFF
------ ---------- ---------- ---------- ---------- ----------
    10       7782 CLARK            2450       2917       -467
    10       7839 KING             5000       2917       2083
    10       7934 MILLER           1300       2917      -1617
    20       7369 SMITH             800       2175      -1375
    20       7876 ADAMS            1100       2175      -1075
    20       7902 FORD             3000       2175        825
    20       7788 SCOTT            3000       2175        825
    20       7566 JONES            2975       2175        800
    30       7499 ALLEN            1600       1567         33
    30       7698 BLAKE            2850       1567       1283
    30       7654 MARTIN           1250       1567       -317
    30       7900 JAMES             950       1567       -617
    30       7844 TURNER           1500       1567        -67
    30       7521 WARD             1250       1567       -317
```

Percentiles

- Basic syntax

```
PERCENTILE_XXXX (<fraction> ) WITHIN GROUP ( ORDER BY < column > )
```

  - 0 < fraction < 1
  - column : the column to compute

```
SELECT PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY SAL) AS MEDIAN_C,
          PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY SAL) AS MEDIAN_D
FROM EMP
```

```
 MEDIAN_C   MEDIAN_D
---------- ----------
      1550       1500
```

Percentiles(2)

- Aggregate Example

```
SELECT DEPTNO,
        PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY SAL) AS MEDIAN_C,
        PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY SAL) AS MEDIAN_D
FROM EMP
GROUP BY DEPTNO;
```

```
  DEPTNO    MEDIAN_C    MEDIAN_D
-------- ----------- -----------
      10        2450        2450
      20        2975        2975
      30        1375        1250
```

Percentile (3)

- Compare each person’s salary to the median: Analytic Example

```
SELECT EMPNO, ENAME, SAL,
            PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY SAL) OVER() AS MEDIAN,
            SAL - PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY SAL) OVER() AS
DIFF
FROM   EMP;
```

```
 EMPNO ENAME             SAL     MEDIAN       DIFF
------ ---------- ---------- ---------- ----------
  7369 SMITH             800       1550       -750
  7900 JAMES             950       1550       -600
  7876 ADAMS            1100       1550       -450
  7521 WARD             1250       1550       -300
  7654 MARTIN           1250       1550       -300
  7934 MILLER           1300       1550       -250
  7844 TURNER           1500       1550        -50
  7499 ALLEN            1600       1550         50
  7782 CLARK            2450       1550        900
  7698 BLAKE            2850       1550       1300
  7566 JONES            2975       1550       1425
  7788 SCOTT            3000       1550       1450
  7902 FORD             3000       1550       1450
  7839 KING             5000       1550       3450
```

Percent_Rank

- Calculate the percent rank of a hypothetical entity in a group with the given values in the chosen fields

- Aggregate Example

```
SELECT PERCENT_RANK(2500) WITHIN GROUP (ORDER BY sal) "Percent-Rank“
FROM EMP;
```

```
Percent-Rank
------------
  .642857143
```

Percent_Rank(2)

- Analytic Example

```
SELECT DEPTNO, ENAME, SAL,
        PERCENT_RANK() OVER (PARTITION BY DEPTNO ORDER BY SAL ASC) as PR
FROM emp
ORDER BY DEPTNO, SAL;
```

```
    DEPTNO ENAME             SAL         PR
---------- ---------- ---------- ----------
        10 MILLER           1300          0
        10 CLARK            2450         .5
        10 KING             5000          1
        20 SMITH             800          0
        20 ADAMS            1100        .25
        20 JONES            2975         .5
        20 SCOTT            3000        .75
        20 FORD             3000        .75
        30 JAMES             950          0
        30 MARTIN           1250         .2
        30 WARD             1250         .2
        30 TURNER           1500         .6
        30 ALLEN            1600         .8
        30 BLAKE            2850          1
```

Percentile + Percent_Rank

- Analytic Example

```
SELECT ENAME, SAL, DEPTNO,
          PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY SAL ASC)
                         OVER (PARTITION BY DEPTNO) as PC,
          PERCENT_RANK() OVER (PARTITION BY DEPTNO ORDER BY SAL ASC) as PR
FROM emp
ORDER BY DEPTNO, SAL;
```

```
ENAME             SAL     DEPTNO         PC         PR
---------- ---------- ---------- ---------- ----------
MILLER           1300         10       2450          0
CLARK            2450         10       2450         .5
KING             5000         10       2450          1
SMITH             800         20       2975          0
ADAMS            1100         20       2975        .25
JONES            2975         20       2975         .5
SCOTT            3000         20       2975        .75
FORD             3000         20       2975        .75
JAMES             950         30       1375          0
MARTIN           1250         30       1375         .2
WARD             1250         30       1375         .2
TURNER           1500         30       1375         .6
ALLEN            1600         30       1375         .8
BLAKE            2850         30       1375          1
```

Ntiles

- NTILE divides the rows into intervals or buckets

```
SELECT  EMPNO, ENAME, SAL,
        NTILE(2) OVER(ORDER BY SAL DESC)  AS NTILE2,
        NTILE(4) OVER(ORDER BY SAL DESC)  AS NTILE4,
        NTILE(10) OVER(ORDER BY SAL DESC) AS NTILE10
FROM   EMP;
```

```
 EMPNO ENAME             SAL     NTILE2     NTILE4    NTILE10
------ ---------- ---------- ---------- ---------- ----------
  7839 KING             5000          1          1          1
  7788 SCOTT            3000          1          1          1
  7902 FORD             3000          1          1          2
  7566 JONES            2975          1          1          2
  7698 BLAKE            2850          1          2          3
  7782 CLARK            2450          1          2          3
  7499 ALLEN            1600          1          2          4
  7844 TURNER           1500          2          2          4
  7934 MILLER           1300          2          3          5
  7521 WARD             1250          2          3          6
  7654 MARTIN           1250          2          3          7
  7876 ADAMS            1100          2          4          8
  7900 JAMES             950          2          4          9
  7369 SMITH             800          2          4         10
```

** NOTE:

- 14 rows / 4 = 3
- each bucket has 3 rows
- the remaining 2 rows are spreaded to the first buckets

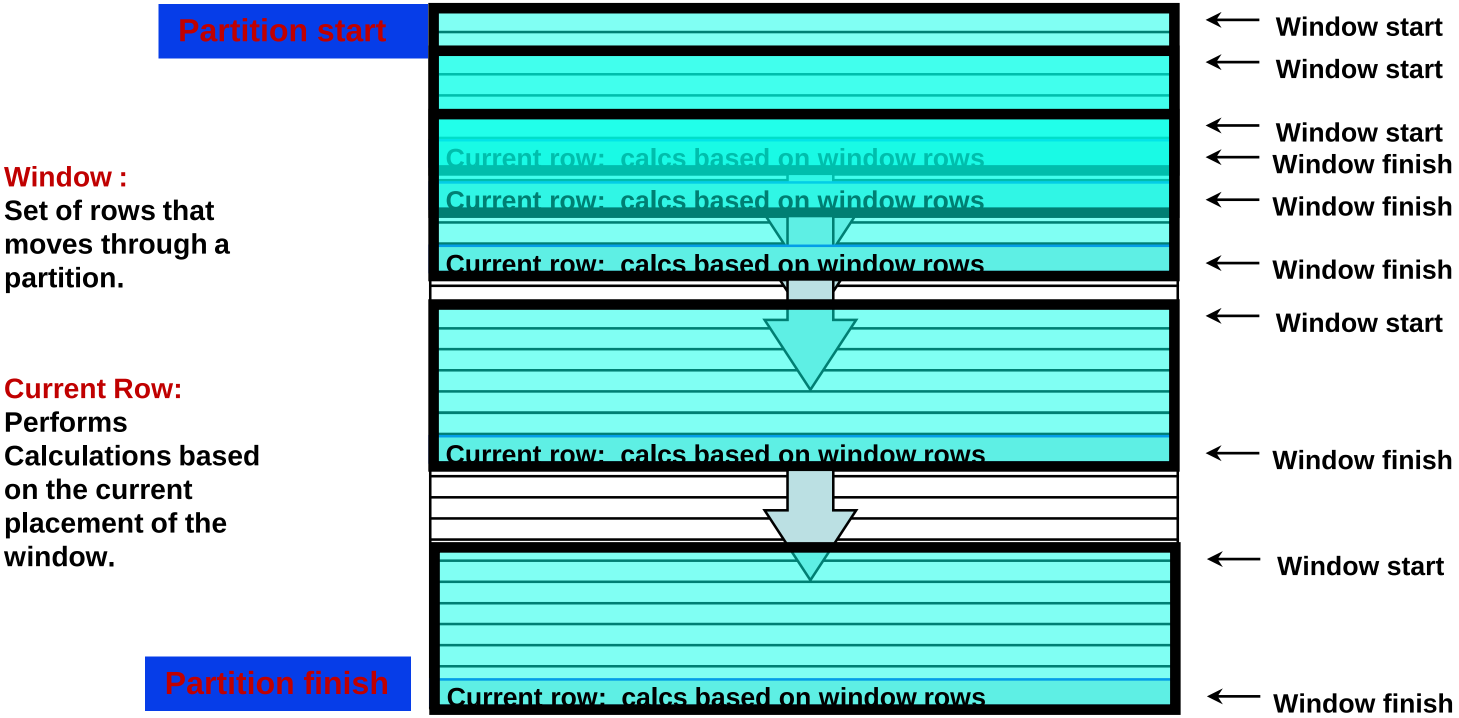
Basic Syntax

<FUNCTION> (<argument>) OVER (<analytic clauses>)

- 3 analytic clauses
  - partitioning
  - ordering
  - **windowing**

Window Functions

- Windowing analytic subclause
  - “sliding window”: a number of rows relative to the current row
  - when moving through the result set row by row, the window “slides” along for each row being processed


Partition start
Window start
Window start
Window start
Current row: calcs based on window rows
Window finish
Current row: calcs based on window rows
Window finish
Current row: calcs based on window rows
Window finish
Window start
Current row: calcs based on window rows
Window finish
Window start
Partition finish
Current row: calcs based on window rows
Window finish

Window :
Set of rows that moves through a partition.

Current Row:
Performs Calculations based on the current placement of the window.

Windowing

- Example: Compute the running total, i.e. the sum of the rows read so far. In this example, the window encompasses the rows from the start of the result set up to and including the current row.

```
SELECT EMPNO, ENAME, SAL, SUM(SAL)   OVER(ORDER BY EMPNO
                 ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
                                                  AS ACC_SAL
FROM   EMP
```

```
 EMPNO ENAME             SAL    ACC_SAL
------ ---------- ---------- ----------
  7369 SMITH             800        800
  7499 ALLEN            1600       2400
  7521 WARD             1250       3650
  7566 JONES            2975       6625
  7654 MARTIN           1250       7875
  7698 BLAKE            2850      10725
  7782 CLARK            2450      13175
                                 .......
```

Windowing (2)

```
SELECT EMPNO, ENAME, SAL, SUM(SAL) OVER( ORDER BY EMPNO
          ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING ) ACC_SAL
FROM   EMP
```

```
 EMPNO ENAME             SAL    ACC_SAL
------ ---------- ---------- ----------
  7369 SMITH             800       2400
  7499 ALLEN            1600       3650
  7521 WARD             1250       5825
  7566 JONES            2975       5475
  7654 MARTIN           1250       7075
  7698 BLAKE            2850       6550
  7782 CLARK            2450       8300
  7788 SCOTT            3000      10450
  7839 KING             5000       9500
  7844 TURNER           1500       7600
  7876 ADAMS            1100       3550
  7900 JAMES             950       5050
  7902 FORD             3000       5250
  7934 MILLER           1300       4300
```

Windowing (3)

- A “sliding window” is meaningful only if the rows are sorted

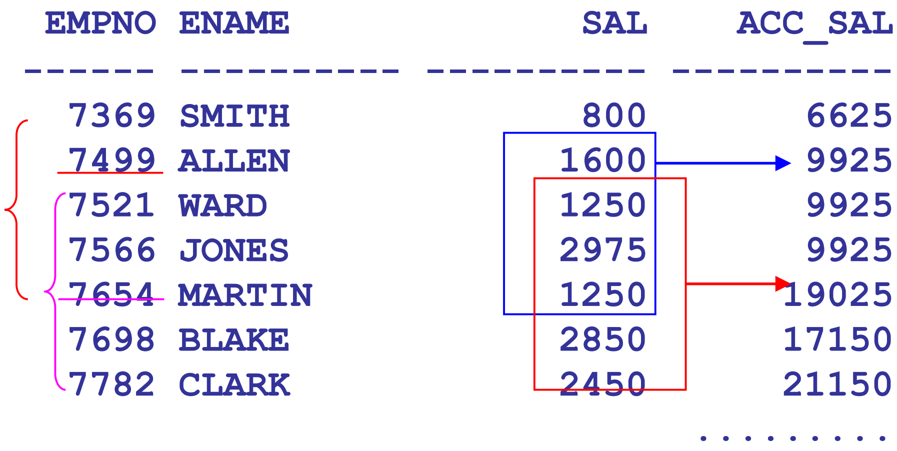
```
SELECT EMPNO, ENAME, SAL,SUM(SAL)
      OVER( ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
                                                     AS ACC_SAL
FROM   EMP
```

```
1행에 오류:
ORA-30485: 윈도우 지정에 ORDER BY 표현식이 없습니다
```

Physical and Logical Offset

- A sliding window is defined as an interval (or offset) with a specified start and end, relative to current row
  - Physical offset: ROWS (rows number)
  - Logical offset: RANGE (a value interval of the column)

```
SELECT EMPNO, ENAME, SAL,SUM(SAL) OVER(ORDER BY EMPNO
           RANGE BETWEEN 100 PRECEDING AND 200 FOLLOWING)
                                          AS ACC_SAL
FROM   EMP
```



```
EMPNO  ENAME                     SAL    ACC_SAL
------ ---------- ---------- ----------
  7369 SMITH                     800       6625
  7499 ALLEN                    1600       9925
  7521 WARD                     1250       9925
  7566 JONES                    2975       9925
  7654 MARTIN                   1250      19025
  7698 BLAKE                    2850      17150
  7782 CLARK                    2450      21150
                                      . . . . . . . . .
```

Physical and Logical Offset

- A subtle difference between RANGE and ROWS

```
SELECT EMPNO, ENAME, SAL,
       SUM(SAL) OVER(ORDER BY SAL
       ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)  ROWS_ACC,
       SUM(SAL) OVER(ORDER BY SAL
       RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)  RANGE_ACC
FROM   EMP;
```

```
EMPNO ENAME             SAL   ROWS_ACC  RANGE_ACC
----- ---------- ---------- ---------- ----------
 7369 SMITH             800        800        800
 7900 JAMES             950       1750       1750
 7876 ADAMS            1100       2850       2850
 7521 WARD             1250       4100       5350
 7654 MARTIN           1250       5350       5350
 7934 MILLER           1300       6650       6650
 7844 TURNER           1500       8150       8150
 7499 ALLEN            1600       9750       9750
 7782 CLARK            2450      12200      12200
 7698 BLAKE            2850      15050      15050
                                        .........
```

Defaults

```
SELECT EMPNO, ENAME, SAL,
       SUM(SAL) OVER(), /* NO ORDER BY, NO WINDOW → REPORTING FUNCTION */
       SUM(SAL) OVER( ORDER BY SAL ) /* DEFAULTS →
        RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)  RANGE_ACC */
FROM   EMP;
```

```
 EMPNO ENAME             SAL SUM(SAL)OVER() SUM(SAL)OVER(ORDERBYSAL)
------ ---------- ---------- -------------- ------------------------
  7369 SMITH             800          29025                      800
  7900 JAMES             950          29025                     1750
  7876 ADAMS            1100          29025                     2850
  7521 WARD             1250          29025                     5350
  7654 MARTIN           1250          29025                     5350
  7934 MILLER           1300          29025                     6650
  7844 TURNER           1500          29025                     8150
  7499 ALLEN            1600          29025                     9750
  7782 CLARK            2450          29025                    12200
  7698 BLAKE            2850          29025                    15050
  7566 JONES            2975          29025                    18025
  7788 SCOTT            3000          29025                    24025
  7902 FORD             3000          29025                    24025
  7839 KING             5000          29025                    29025
```

Partitioning and Windows

```
SELECT DEPTNO, EMPNO, ENAME, SAL,
     SUM(SAL) OVER( PARTITION BY DEPTNO
                    ORDER BY EMPNO
                    ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS ACC_SAL
FROM   EMP;
```

```
 DEPTNO      EMPNO ENAME             SAL    ACC_SAL
------- ---------- ---------- ---------- ----------
     10       7782 CLARK            2450       7450
     10       7839 KING             5000       8750
     10       7934 MILLER           1300       6300
     20       7369 SMITH             800       3775
     20       7566 JONES            2975       6775
     20       7788 SCOTT            3000       7075
     20       7876 ADAMS            1100       7100
     20       7902 FORD             3000       4100
     30       7499 ALLEN            1600       2850
     30       7521 WARD             1250       4100
     30       7654 MARTIN           1250       5350
     30       7698 BLAKE            2850       5600
     30       7844 TURNER           1500       5300
     30       7900 JAMES             950       2450
```

Time Interval

- So far, windows based on numeric intervals. Window can also be based on time interval

```
SELECT EMPNO, ENAME, HIREDATE, SAL, ROUND( AVG(SAL) OVER(ORDER BY HIREDATE
                                         RANGE BETWEEN INTERVAL '6' MONTH PRECEDING
                                        AND  INTERVAL '6' MONTH FOLLOWING) ) AS MOV_AVG
FROM   EMP;
```

```
EMPNO ENAME      HIREDATE        SAL    MOV_AVG
----- ---------- -------- ---------- ----------
 7369 SMITH      80/12/17        800 →     1988
 7499 ALLEN      81/02/20       1600       1988
 7521 WARD       81/02/22       1250       1988
 7566 JONES      81/04/02       2975       1834
 7698 BLAKE      81/05/01       2850       1834
 7782 CLARK      81/06/09       2450       2148
 7844 TURNER     81/09/08       1500       2364
 7654 MARTIN     81/09/28       1250       2364
                                      ..........
```

Additional Functions

- FIRST_VALUE() and LAST_VALUE()

```
SELECT EMPNO, ENAME, HIREDATE, SAL,
          FIRST_VALUE(SAL)   OVER(ORDER BY HIREDATE
                                     RANGE BETWEEN INTERVAL '6' MONTH PRECEDING
                                       AND INTERVAL '6' MONTH FOLLOWING) AS FIRST,
          LAST_VALUE(SAL) OVER(ORDER BY HIREDATE
                                     RANGE BETWEEN INTERVAL '6' MONTH PRECEDING
                                       AND INTERVAL '6' MONTH FOLLOWING) AS LAST
FROM EMP;
```

```
EMPNO ENAME      HIREDATE        SAL      FIRST       LAST
----- ---------- -------- ---------- ---------- ----------
 7369 SMITH      80/12/17        800        800       2450
 7499 ALLEN      81/02/20       1600        800       2450
 7521 WARD       81/02/22       1250        800       2450
 7566 JONES      81/04/02       2975        800       1250
 7698 BLAKE      81/05/01       2850        800       1250
 7782 CLARK      81/06/09       2450        800       3000
 7844 TURNER     81/09/08       1500       2975       1300
 7654 MARTIN     81/09/28       1250       2975       1300
                                                  ..........
```

Additional Functions (2)

- RATIO_TO_REPORT()

```
SELECT EMPNO, ENAME, SAL,
          SUM(SAL) OVER()                          AS TOTAL,
          SAL/SUM(SAL) OVER()                      AS FRACTION,
          RATIO_TO_REPORT(SAL) OVER()    AS FRACTION
FROM EMP;
```

```
EMPNO ENAME             SAL      TOTAL   FRACTION   FRACTION
----- ---------- ---------- ---------- ---------- ----------
 7369 SMITH             800      29025 .027562446 .027562446
 7499 ALLEN            1600      29025 .055124892 .055124892
 7521 WARD             1250      29025 .043066322 .043066322
 7566 JONES            2975      29025 .102497847 .102497847
 7654 MARTIN           1250      29025 .043066322 .043066322
 7698 BLAKE            2850      29025 .098191214 .098191214
                                                    ........
```

Additional Functions (3)

- LAG() and LEAD()
  - return values from preceding and following rows
  - default: immediate preceding/following row

```
SELECT EMPNO, ENAME, SAL,
          LAG(SAL) OVER (ORDER BY EMPNO)    AS LAG1,
          LEAD(SAL) OVER (ORDER BY EMPNO)   AS LEAD1,
          LAG(SAL,3) OVER (ORDER BY EMPNO)  AS LAG3,
          LEAD(SAL,3) OVER (ORDER BY EMPNO) AS LEAD3
FROM EMP;
```

```
EMPNO ENAME             SAL       LAG1      LEAD1       LAG3      LEAD3
----- ---------- ---------- ---------- ---------- ---------- ----------
 7369 SMITH             800                  1600                  2975
 7499 ALLEN            1600        800       1250                  1250
 7521 WARD             1250       1600       2975                  2850
 7566 JONES            2975       1250       1250        800       2450
 7654 MARTIN           1250       2975       2850       1600       3000
 7698 BLAKE            2850       1250       2450       1250       5000
 7782 CLARK            2450       2850       3000       2975       1500
 7788 SCOTT            3000       2450       5000       1250       1100
 7839 KING             5000       3000       1500       2850        950
                                                               ........
```

Additional Functions (4)

- LAG()/LEAD() and NULL

```
SELECT EMPNO, ENAME, SAL,
            LAG(SAL,1,0) OVER (ORDER BY EMPNO)    AS LAG1,
            LEAD(SAL,1, 0) OVER (ORDER BY EMPNO)   AS LEAD1,
            LAG(SAL,3, 0) OVER (ORDER BY EMPNO)  AS LAG3,
            LEAD(SAL,3, 0) OVER (ORDER BY EMPNO) AS LEAD3
FROM EMP;
```

```
EMPNO ENAME             SAL       LAG1      LEAD1       LAG3      LEAD3
----- ---------- ---------- ---------- ---------- ---------- ----------
 7369 SMITH             800          0       1600          0       2975
 7499 ALLEN            1600        800       1250          0       1250
 7521 WARD             1250       1600       2975          0       2850
 7566 JONES            2975       1250       1250        800       2450
 7654 MARTIN           1250       2975       2850       1600       3000
 7698 BLAKE            2850       1250       2450       1250       5000
 7782 CLARK            2450       2850       3000       2975       1500
                                                                ........
```

ListAggr

- http://www.oracle.com/technetwork/issue-archive/2017/17-jan/o17dba-mcdonald-3434673.html

Summary:
Oracle Analytic Functions

Several families of new functions enhancing decision support and OLAP processing

- Improve query performance through optimized processing
- Enhance developer productivity with simplified SQL
- Added to ANSI SQL standard

References

1. Oracle 11G Release 2 Data Warehousing Guide : Chapter 22, "SQL for Analysis and Reporting"
   - And, Googling with "Analytical SQL in Oracle Database 12c" (Oracle White Paper, Nov. 2013)

- If you are interested in more advanced SQL features in Oracle, then see
  - Chater 23. SQL for Modeling and Chapter 24. Advanced Business Intelligence Queries in Oracle 11G Rel 2. DW Guide.