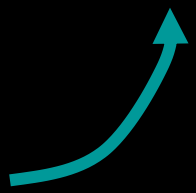
NOTE

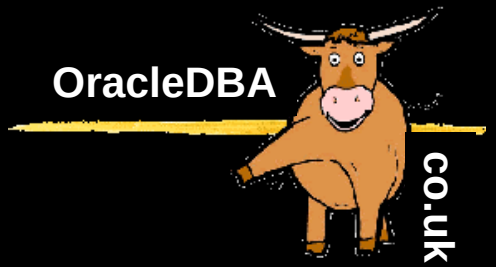
itty bitty fonts in this presentation

```
SQL> exec sample_font
```



Can you read this ?

Connor McDonald



The “look how I smart I am” slide…

this page intentionally left blank

sneak peek

12g

moral of the story

don’t believe anything

try, try, try


ANALYTICS

awesome

sometimes

do

really
silly
things

Application Development

You can develop applications using many operating systems, lang…nt parts of the application can run inside the database, on an application serv…

This book introduces you to various aspects of application develop… typical duties of a developer.

Application Developer's Guide - Fundamentals	

Java

The Java programming language is used for database code that runs … tier, or on a client system. Many database components have Java AP… Javadoc format.

Java Developer's Guide	
JDBC Developer's Guide and Reference	
JPublisher User's Guide	
Data Cartridge Java API Reference (Javadoc)	
Data Mining Java API Reference (Javadoc)	
OLAP Java API Reference (Javadoc)	
SQLJ Developer's Guide and Reference	
Ultra Search Java API Reference (Javadoc)	HTML
XML Java API Reference (Javadoc)	HTML
Streams Advanced Queuing Java API Reference (Javadoc)	HTML
Spatial Java API Reference (Javadoc)	HTML
OLAP Analytic Workspace Java API Reference (Javadoc)	HTML
Globalization Development Kit Java API Reference (Javadoc)	HTML
interMedia Java Classes API Reference (Javadoc)	HTML
interMedia Java Classes for Servlets and JSP API Reference (Javadoc)	HTML

XML

You can store XML content in the database, and write applications to process it and use it as a data interchange format:

XML DB Developer's Guide	HTML	PDF
XML C++ API Reference	HTML	PDF
XML C API Reference	HTML	PDF
XML Developer's Kit Programmer's Guide	HTML	PDF
XML Java API Reference (Javadoc)	HTML	

Quick Search

Search | All books | Books on this tab

…and PL/SQL

…uing language for all database programming. PL/SQL is tightly integrated with SQL, …side the database such as triggers, stored procedures and functions, and …he command-line tool for running SQL and compiling PL/SQL.

…opment, there is an online quick reference that

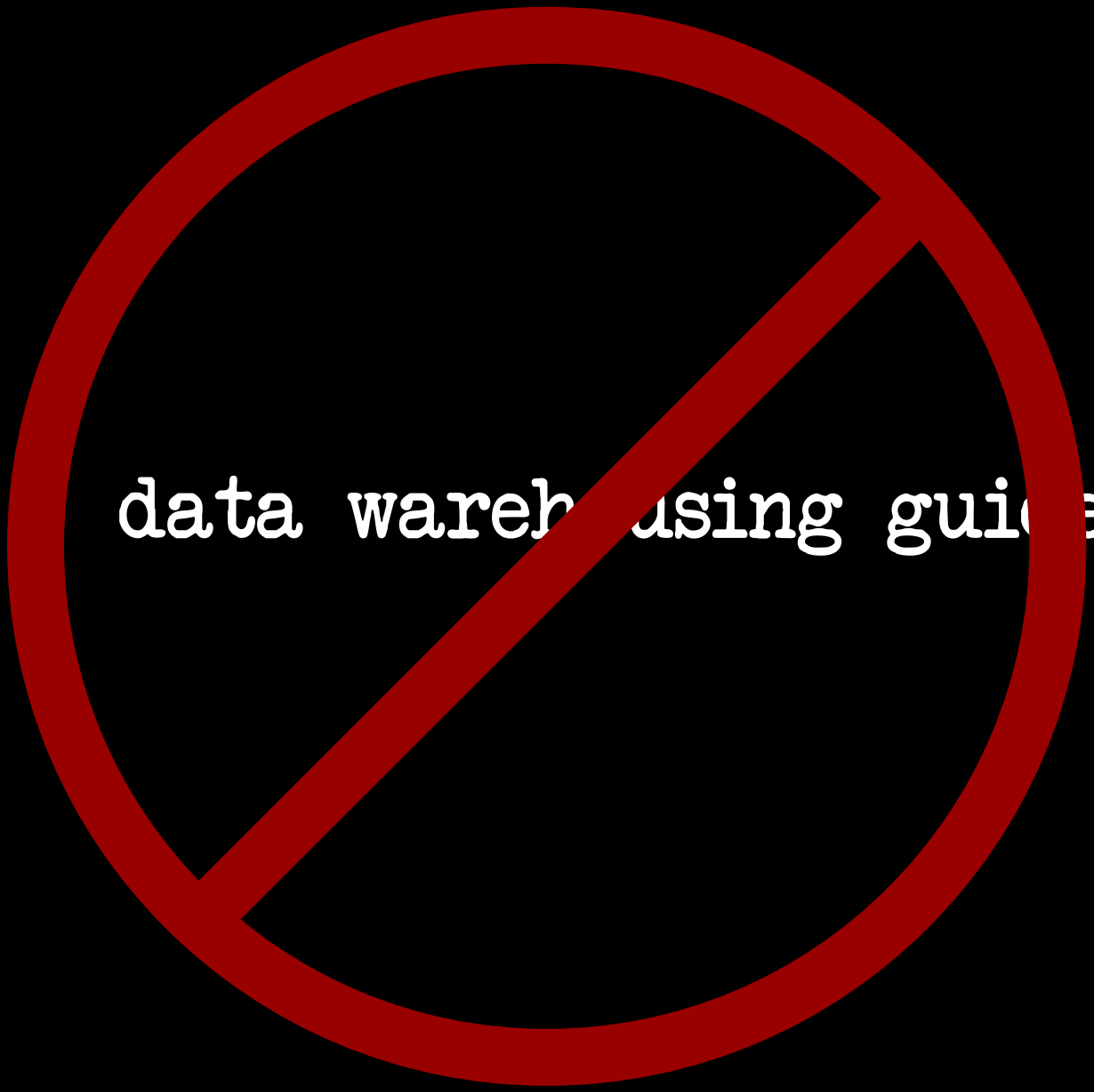
	HTML	PDF
	HTML	PDF
	HTML	PDF
	HTML	PDF
	HTML	PDF
	HTML	PDF
Oracle Call Interface Programmer's Guide	HTML	PDF
Oracle C++ Call Interface Programmer's Guide	HTML	PDF

Application Developer Guides

These books cover specific aspects of developing with Oracle:

Application Developer's Guide - Large Objects	HTML	PDF
Application Developer's Guide - Object-Relational Features	HTML	PDF
Application Developer's Guide - Rules Manager and Expression Filter	HTML	PDF
Application Developer's Guide - Fundamentals	HTML	PDF
Application Developer's Guide - Workspace Manager	HTML	PDF
Data Mining Application Developer's Guide	HTML	PDF
HTML DB 2 Day Developer	HTML	PDF
HTML DB User's Guide	HTML	PDF
OLAP Application Developer's Guide	HTML	PDF
Sample Schemas	HTML	PDF
Text Application Developer's Guide	HTML	PDF

Application Development


data wareh using gui e

Oracle® Database Data Warehousing Guide
10*g* Release 2 (10.2)
Part Number B14223-01


Next

Contents

Title and Copyright Information

Send Us Your Comments

Preface

- Intended Audience
- Documentation Accessibility
- Organization
- Related Documentation
- Conventions

21 SQL for Analysis and Reporting

- Overview of SQL for Analysis and Reporting
- Ranking Functions
  - RANK and DENSE_RANK Functions
    - Ranking Order
    - Ranking on Multiple Expressions
    - RANK and DENSE_RANK Difference
    - Per Group Ranking
    - Per Cube and Rollup Group Ranking
    - Treatment of NULLs
  - Bottom N Ranking
  - CUME_DIST Function
  - PERCENT_RANK Function
  - NTILE Function
  - ROW_NUMBER Function

simple syntax

```
<function> ( <arg>,<arg>,… )
OVER (
   <partition clause>
   <sorting clause>
   <windowing clause>
)
```

that's it !

quick example #1

employees by salary

```
SQL> select empno, ename, job, hiredate, sal
  2  from emp
  3  order by sal;

     EMPNO ENAME      JOB       HIREDATE          SAL
---------- ---------- --------- --------- ----------
      7369 SMITH      CLERK     17-DEC-80         800
      7900 JAMES      CLERK     03-DEC-81         950
      7876 ADAMS      CLERK     12-JAN-83        1100
      7521 WARD       SALESMAN  22-FEB-81        1250
      7654 MARTIN     SALESMAN  28-SEP-81        1250
      7934 MILLER     CLERK     23-JAN-82        1300
      7844 TURNER     SALESMAN  08-SEP-81        1500
      7499 ALLEN      SALESMAN  20-FEB-81        1600
      7782 CLARK      MANAGER   09-JUN-81        2450
      7698 BLAKE      MANAGER   01-MAY-81        2850
      7566 JONES      MANAGER   02-APR-81        2975
      7902 FORD       ANALYST   03-DEC-81        3000
      7788 SCOTT      ANALYST   09-DEC-82        3000
      7839 KING       PRESIDENT 17-NOV-81        5000
```

”hiring sequence”

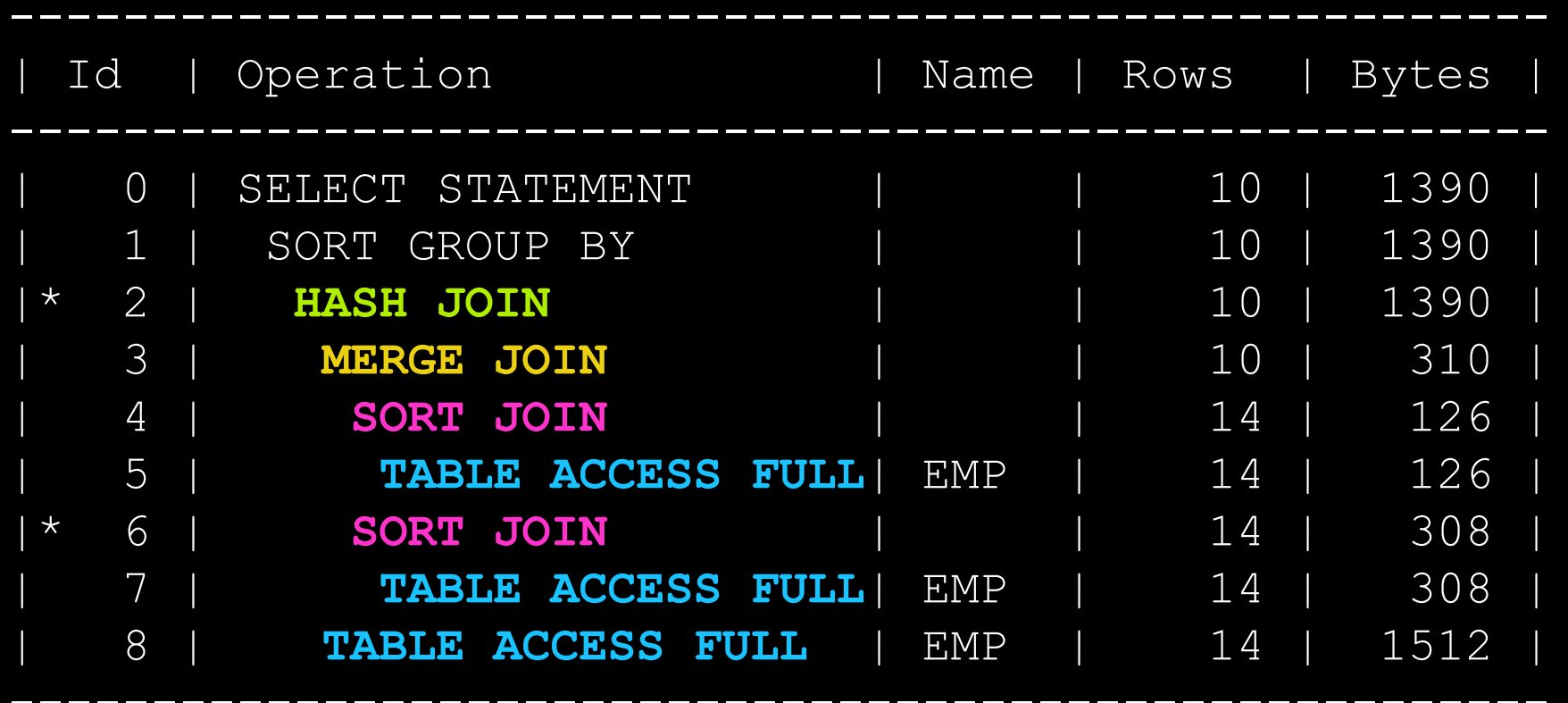
SMITH was hired "first"

EMPNO	ENAME	JOB	HIREDATE	SAL	HIRE_SEQ
7369	SMITH	CLERK	17-DEC-80	800	1
7900	JAMES	CLERK	03-DEC-81	950	10
7876	ADAMS	CLERK	12-JAN-83	1100	14
7521	WARD	SALESMAN	22-FEB-81	1250	3
7654	MARTIN	SALESMAN	28-SEP-81	1250	8
7934	MILLER	CLERK	23-JAN-82	1300	12
7844	TURNER	SALESMAN	08-SEP-81	1500	7
7499	ALLEN	SALESMAN	20-FEB-81	1600	2
7782	CLARK	MANAGER	09-JUN-81	2450	6
7698	BLAKE	MANAGER	01-MAY-81	2850	5
7566	JONES	MANAGER	02-APR-81	2975	4
7902	FORD	ANALYST	03-DEC-81	3000	10
7788	SCOTT	ANALYST	09-DEC-82	3000	13
7839	KING	PRESIDENT	17-NOV-81	5000	9

ADAMS was hired "last"

Oracle 8 and below

```
SQL> select e.empno, e.ename, e.job,
  2         e.hiredate, e.sal, x.seq
  3  from emp e,
  4    (  select e2.empno, count(*) seq
  5       from   emp e1, emp e2
  6       where  e1.hiredate <= e2.hiredate
  7       group by e2.empno
  8    ) x
  9  where e.empno = x.empno
 10  order by sal;
```


Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		10	1390
1	SORT GROUP BY		10	1390
* 2	HASH JOIN		10	1390
3	MERGE JOIN		10	310
4	SORT JOIN		14	126
5	TABLE ACCESS FULL	EMP	14	126
* 6	SORT JOIN		14	308
7	TABLE ACCESS FULL	EMP	14	308
8	TABLE ACCESS FULL	EMP	14	1512

```
SQL> select empno, ename, job, hiredate, sal,
  2           rank() OVER (order by hiredate) as hire_seq
  3  from emp
  4  order by sal;
```

EMPNO	ENAME	JOB	HIREDATE	SAL	HIRE_SEQ
7369	SMITH	CLERK	17-DEC-80	800	1
7900	JAMES	CLERK	03-DEC-81	950	10
7876	ADAMS	CLERK	12-JAN-83	1100	14
7521	WARD	SALESMAN	22-FEB-81	1250	3
7654	MARTIN	SALESMAN	28-SEP-81	1250	8
7934	MILLER	CLERK	23-JAN-82	1300	12
7844	TURNER	SALESMAN	08-SEP-81	1500	7
7499	ALLEN	SALESMAN	20-FEB-81	1600	2
7782	CLARK	MANAGER	09-JUN-81	2450	6
7698	BLAKE	MANAGER	01-MAY-81	2850	5
7566	JONES	MANAGER	02-APR-81	2975	4
7902	FORD	ANALYST	03-DEC-81	3000	10
7788	SCOTT	ANALYST	09-DEC-82	3000	13
7839	KING	PRESIDENT	17-NOV-81	5000	9

function

```
rank() OVER (
  order by hire_date) as hire_seq
```

sorting clause

functions
for ranking

RANK 1, 2, 3, 3, 5, …

functions
for ranking

RANK 1, 2, 3, 3, 5, …

DENSE_RANK 1, 2, 3, 3, 4, …

functions
for ranking

RANK 1, 2, 3, 3, 5, …

DENSE_RANK

CUME_DIST

```
SQL> select ename, sal,
  2    100*cume_dist() over ( order by sal ) as pct
  3  from emp
  4  order by sal;

ENAME              SAL     PCT
---------- ---------- -------
SMITH              800    7.14
JAMES              950   14.29
ADAMS             1100   21.43
WARD              1250   35.71
MARTIN            1250   35.71
MILLER            1300   42.86
TURNER            1500   50.00
ALLEN             1600   57.14
CLARK             2450   64.29
BLAKE             2850   71.43
JONES             2975   78.57
FORD              3000   92.86
SCOTT             3000   92.86
KING              5000  100.00
```

RANK 1, 2, 3, 3, 5, …

DENSE_RANK

CUME_DIST

PERCENT_RANK

```
SQL> select ename, sal,
  2   100*percent_rank() over ( order by sal ) pct
  3  from emp
  4  order by ename;

ENAME                SAL     PCT
---------- ---------- -------
ADAMS              1100   15.38
ALLEN              1600   53.85
BLAKE              2850   69.23
CLARK              2450   61.54
FORD               3000   84.62
JAMES               950    7.69
JONES              2975   76.92
KING               5000  100.00
MARTIN             1250   23.08
MILLER             1300   38.46
SCOTT              3000   84.62
SMITH               800     .00
TURNER             1500   46.15
WARD               1250   23.08
```

RANK 1, 2, 3, 3, 5, …

DENSE_RANK

CUME_DIST

PERCENT_RANK

NTILE

```
SQL> select ename, sal,
  2    ntile(4) over ( order by sal ) as quartile
  3  from emp
  4  order by ename;

ENAME                SAL   QUARTILE
---------- ---------- ----------
ADAMS              1100          1
ALLEN              1600          2
BLAKE              2850          3
CLARK              2450          3
FORD               3000          4
JAMES               950          1
JONES              2975          3
KING               5000          4
MARTIN             1250          2
MILLER             1300          2
SCOTT              3000          4
SMITH               800          1
TURNER             1500          2
WARD               1250          1
```

RANK 1, 2, 3, 3, 5, …

DENSE_RANK 1, 2, 3, 3, 4, …

CUME_DIST

PERCENT_RANK

functions
for ranking

NTILE

ROW_NUMBER 1, 2, 3, 4, 5, …

functions
for aggregation

SUM
AVERAGE
MIN
MAX
COUNT

quick example #2

```
SQL> select deptno, empno, ename, job, sal
  2  from emp
  3  order by deptno, empno;
```

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

"department salaries,
running total
by employee name"

```
SQL> select deptno, empno, ename, job, sal,
  2    sum(sal) OVER (
  3      partition by deptno
  4      order by ename) as running_total
  5  from emp
  6  order by deptno, ename;
```

DEPTNO	EMPNO	ENAME	JOB	SAL	RUNNING_TOTAL
10	7782	CLARK	MANAGER	2450	2450
10	7839	KING	PRESIDENT	5000	7450
10	7934	MILLER	CLERK	1300	8750
20	7876	ADAMS	CLERK	1100	1100
20	7902	FORD	ANALYST	3000	4100
20	7566	JONES	MANAGER	2975	7075
20	7788	SCOTT	ANALYST	3000	10075
20	7369	SMITH	CLERK	800	10875
30	7499	ALLEN	SALESMAN	1600	1600
30	7698	BLAKE	MANAGER	2850	4450
30	7900	JAMES	CLERK	950	5400
30	7654	MARTIN	SALESMAN	1250	6650
30	7844	TURNER	SALESMAN	1500	8150
30	7521	WARD	SALESMAN	1250	9400

```
SQL> select deptno, empno, ename, job, sal,
  2    sum(sal) OVER (
  3      partition by deptno
  4      order by ename) as running_total
  5  from emp
  6  order by deptno, ename;
```

DEPTNO	EMPNO	ENAME	JOB	SAL	RUNNING_TOTAL
10	7782	CLARK	MANAGER	2450	2450
10	7839	KING	PRESIDENT	5000	7450
10	7934	MILLER	CLERK	1300	8750
20	7876	ADAMS	CLERK	1100	1100
20	7902	FORD	ANALYST	3000	4100
20	7566	JONES	MANAGER	2975	7075
20	7788	SCOTT	ANALYST	3000	10075
20	7369	SMITH	CLERK	800	10875
30	7499	ALLEN	SALESMAN	1600	1600
30	7698	BLAKE	MANAGER	2850	4450
30	7900	JAMES	CLERK	950	5400
30	7654	MARTIN	SALESMAN	1250	6650
30	7844	TURNER	SALESMAN	1500	8150
30	7521	WARD	SALESMAN	1250	9400

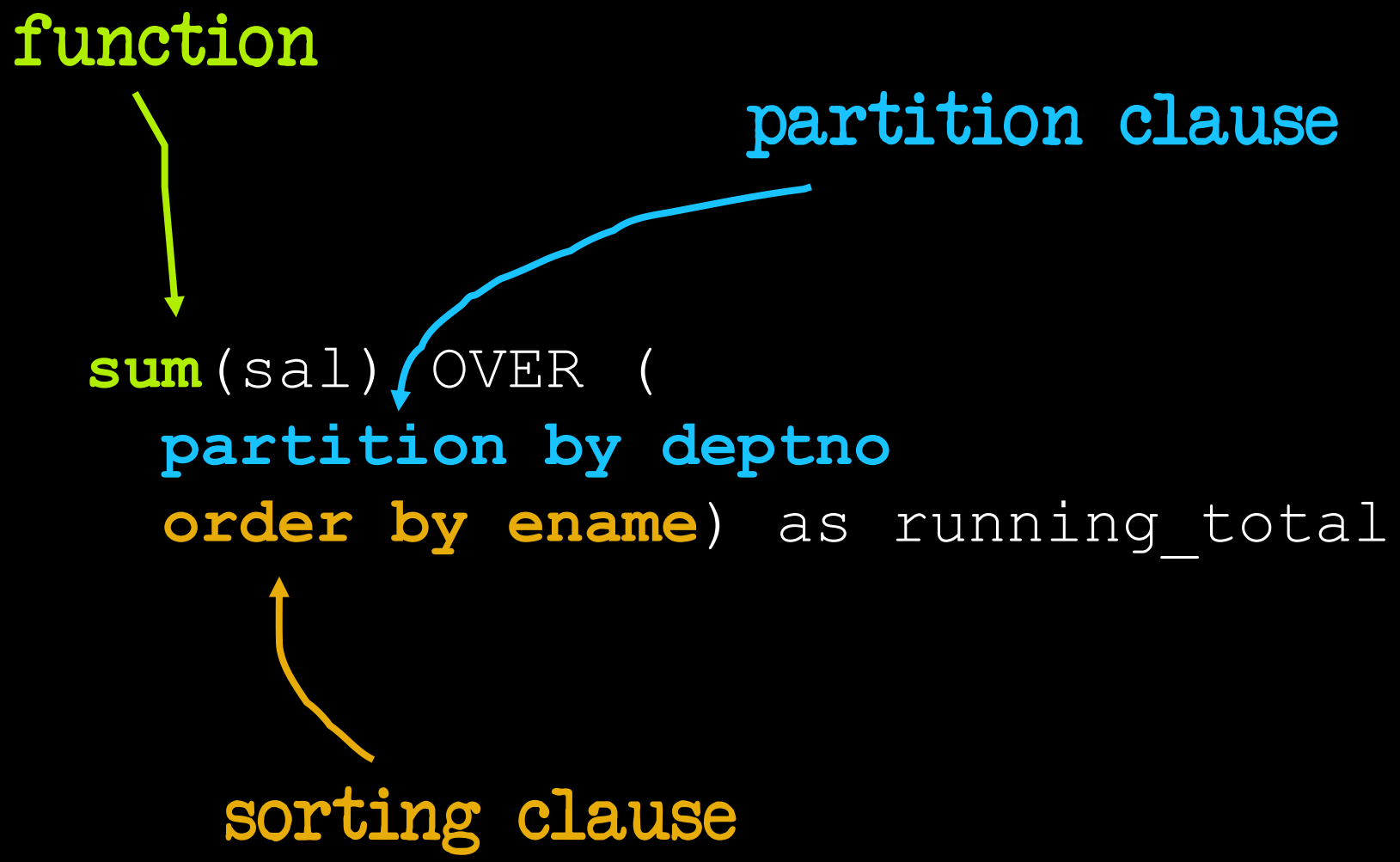
```
SQL> select deptno, empno, ename, job, sal,
  2    sum(sal) OVER (
  3      partition by deptno
  4      order by ename) as running_total
  5  from emp
  6  order by deptno, ename;
```

DEPTNO	EMPNO	ENAME	JOB	SAL	RUNNING_TOTAL
10	7782	CLARK	MANAGER	2450	2450
10	7839	KING	PRESIDENT	5000	7450
10	7934	MILLER	CLERK	1300	8750
20	7876	ADAMS	CLERK	1100	1100
20	7902	FORD	ANALYST	3000	4100
20	7566	JONES	MANAGER	2975	7075
20	7788	SCOTT	ANALYST	3000	10075
20	7369	SMITH	CLERK	800	10875
30	7499	ALLEN	SALESMAN	1600	1600
30	7698	BLAKE	MANAGER	2850	4450
30	7900	JAMES	CLERK	950	5400
30	7654	MARTIN	SALESMAN	1250	6650
30	7844	TURNER	SALESMAN	1500	8150
30	7521	WARD	SALESMAN	1250	9400

```
SQL> select deptno, empno, ename, job, sal,
  2    sum(sal) OVER (
  3      partition by deptno
  4      order by ename) as running_total
  5  from emp
  6  order by deptno, ename;

    DEPTNO      EMPNO ENAME      JOB              SAL RUNNING_TOTAL
---------- ---------- ---------- --------- ---------- -------------
        10       7782 CLARK      MANAGER        2450          2450
        10       7839 KING       PRESIDENT      5000          7450
        10       7934 MILLER     CLERK          1300          8750
        20       7876 ADAMS      CLERK          1100          1100
        20       7902 FORD       ANALYST        3000          4100
        20       7566 JONES      MANAGER        2975          7075
        20       7788 SCOTT      ANALYST        3000         10075
        20       7369 SMITH      CLERK           800         10875
        30       7499 ALLEN      SALESMAN       1600          1600
        30       7698 BLAKE      MANAGER        2850          4450
        30       7900 JAMES      CLERK           950          5400
        30       7654 MARTIN     SALESMAN       1250          6650
        30       7844 TURNER     SALESMAN       1500          8150
        30       7521 WARD       SALESMAN       1250          9400
```

```
SQL> select deptno, empno, ename, job, sal,
  2    sum(sal) OVER (
  3      partition by deptno
  4      order by ename) as running_total
  5  from emp
  6  order by deptno, ename;
```

DEPTNO	EMPNO	ENAME	JOB	SAL	RUNNING_TOTAL
10	7782	CLARK	MANAGER	2450	2450
10	7839	KING	PRESIDENT	5000	7450
10	7934	MILLER	CLERK	1300	8750
20	7876	ADAMS	CLERK	1100	1100
20	7902	FORD	ANALYST	3000	4100
20	7566	JONES	MANAGER	2975	7075
20	7788	SCOTT	ANALYST	3000	10075
20	7369	SMITH	CLERK	800	10875
30	7499	ALLEN	SALESMAN	1600	1600
30	7698	BLAKE	MANAGER	2850	4450
30	7900	JAMES	CLERK	950	5400
30	7654	MARTIN	SALESMAN	1250	6650
30	7844	TURNER	SALESMAN	1500	8150
30	7521	WARD	SALESMAN	1250	9400


function
partition clause
sum(sal) OVER (
partition by deptno
order by ename) as running_total
sorting clause

```
SQL> select deptno, empno, ename, job, sal,
  2    sum(sal) OVER (
  3      partition by deptno
  4      order by ename) as running_total
  5  from emp
  6  order by deptno, ename;
```

DEPTNO	EMPNO	ENAME	JOB	SAL	RUNNING_TOTAL
10	7782	CLARK	MANAGER	2450	2450
10	7839	KING	PRESIDENT	5000	7450
10	7934	MILLER	CLERK	1300	8750
20	7876	ADAMS	CLERK	1100	1100
20	7902	FORD	ANALYST	3000	4100
20	7566	JONES	MANAGER	2975	7075
20	7788	SCOTT	ANALYST	3000	10075
20	7369	SMITH	CLERK	800	10875
30	7499	ALLEN	SALESMAN	1600	1600
30	7698	BLAKE	MANAGER	2850	4450
30	7900	JAMES	CLERK	950	5400
30	7654	MARTIN	SALESMAN	1250	6650
30	7844	TURNER	SALESMAN	1500	8150
30	7521	WARD	SALESMAN	1250	9400

```
SQL> select deptno, empno, ename, job, sal,
  2    sum(sal) OVER (
  3      partition by deptno
  4      order by ename) as running_total
  5  from emp
  6  order by deptno, empno
```

DEPTNO	EMPNO	ENAME	JOB	SAL	RUNNING_TOTAL
10	7782	CLARK	MANAGER	2450	2450
10	7839	KING	PRESIDENT	5000	7450
10	7934	MILLER	CLERK	1300	8750
20	7369	SMITH	CLERK	800	10875
20	7566	JONES	MANAGER	2975	7075
20	7788	SCOTT	ANALYST	3000	10075
20	7876	ADAMS	CLERK	1100	1100
20	7902	FORD	ANALYST	3000	4100
30	7499	ALLEN	SALESMAN	1600	1600
30	7521	WARD	SALESMAN	1250	9400
30	7654	MARTIN	SALESMAN	1250	6650
30	7698	BLAKE	MANAGER	2850	4450
30	7844	TURNER	SALESMAN	1500	8150
30	7900	JAMES	CLERK	950	5400

lots of power

```
SQL> select deptno, job, ename, sal,
  2         sum(sal) over () total_sal,
  3         sum(sal) over
  4             ( partition by deptno) sal_by_dept,
  5         sum(sal) over
  6             ( partition by deptno, job) sal_by_dept_job,
  7  from emp
  8  order by deptno, job;
```

DEPTNO	JOB	ENAME	SAL	TOTAL_SAL	SAL_BY_DEPT	SAL_BY_DEPT_JOB
10	CLERK	MILLER	1300	29025	8750	1300
10	MANAGER	CLARK	2450	29025	8750	2450
10	PRESIDENT	KING	5000	29025	8750	5000
20	ANALYST	SCOTT	3000	29025	10875	6000
20	ANALYST	FORD	3000	29025	10875	6000
20	CLERK	ADAMS	1100	29025	10875	1900
20	CLERK	SMITH	800	29025	10875	1900
20	MANAGER	JONES	2975	29025	10875	2975
30	CLERK	JAMES	950	29025	9400	950
30	MANAGER	BLAKE	2850	29025	9400	2850
30	SALESMAN	TURNER	1500	29025	9400	5600
30	SALESMAN	MARTIN	1250	29025	9400	5600
30	SALESMAN	WARD	1250	29025	9400	5600
30	SALESMAN	ALLEN	1600	29025	9400	5600

STDDEV_SAMP

VAR_POP

STATS_T_TEST_…

COVAR_SAMP

GROUP_ID

MEDIAN

STATS_MODE

STDDEV

STATS_BINOMIAL_TEST

CORR

REGR_ …

STATS_CROSSTAB

GROUPING_ID

COLLECT

and a whole lot more…

STATS_MW_TEST

PERCENTILE_CONT

STATS_WSR_TEST

STDDEV_POP

COVAR_POP

VARIANCE

GROUPING

STATS_ONE_WAY_ANOVA

STATS_KS_TEST

PERCENTILE_DISC

STATS_F_TEST

VAR_SAMP

important note !!!

aggregations

calculated

not

restrictive

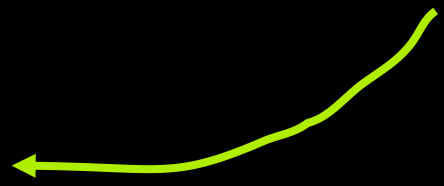
conventional
aggregation

```
SQL> select deptno, sum(sal)
  2  from   emp
  3  group  by deptno;

    DEPTNO   SUM(SAL)
---------- ----------
        10       8750
        20      10875
        30       9400
```

```
SQL> select ename, deptno,
  2     sum(sal) over
  3         ( partition by deptno) as deptsal
  4  from emp
  5  order by deptno;
```

ENAME	DEPTNO	DEPTSAL
CLARK	10	8750
KING	10	8750
MILLER	10	8750
JONES	20	10875
FORD	20	10875
ADAMS	20	10875
SMITH	20	10875
SCOTT	20	10875
WARD	30	9400
TURNER	30	9400
ALLEN	30	9400
JAMES	30	9400
BLAKE	30	9400
MARTIN	30	9400

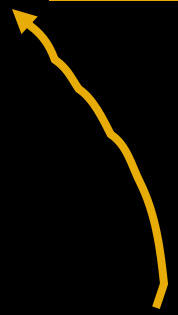
there is still 14 rows !!!



two kinds

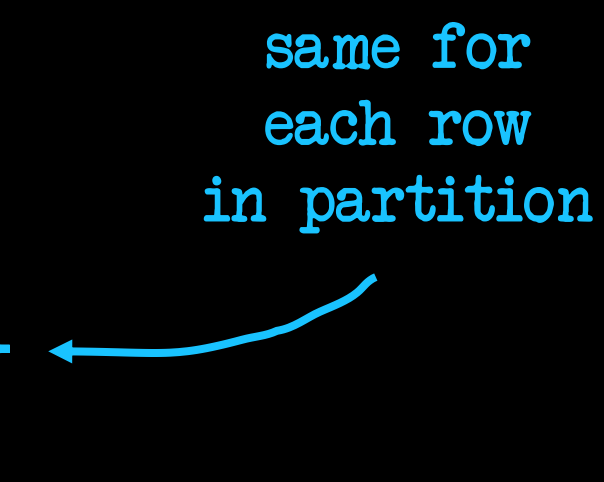
aggregation

”same aggregate for each row in a partition”

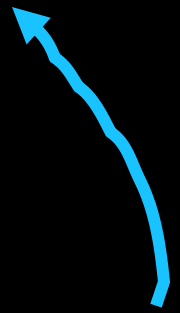


reporting
aggregation

```
SQL> select ename, deptno,
  2    sum(sal) over ( partition by deptno) as deptsal
  3  from emp
  4  order by deptno;

ENAME          DEPTNO    DEPTSAL
---------- ---------- ----------
CLARK              10       8750
KING               10       8750
MILLER             10       8750
JONES              20      10875
FORD               20      10875
ADAMS              20      10875
SMITH              20      10875
SCOTT              20      10875
WARD               30       9400
TURNER             30       9400
ALLEN              30       9400
JAMES              30       9400
BLAKE              30       9400
MARTIN             30       9400
```

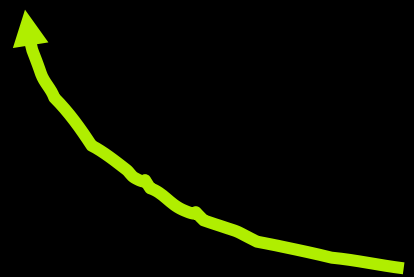


”changing aggregate for each row in a partition”



windowing
aggregation

```
<function> ( <arg>,<arg>,… )
OVER (
   <partition clause>
   <sorting clause>
   <windowing clause>
)
```



defines how ”broadly” the aggregating function applies

”salary cumulative total”

```
SQL> select
  2    empno, ename, sal,
  3    sum(sal)
  4     over ( order by empno
  5      rows between unbounded preceding and current row ) as cumtot
  6  from   emp
  7  order by empno;

     EMPNO ENAME             SAL     CUMTOT
---------- ---------- ---------- ----------
      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
      7788 SCOTT            3000      16175
      7839 KING             5000      21175
      7844 TURNER           1500      22675
      7876 ADAMS            1100      23775
      7900 JAMES             950      24725
      7902 FORD             3000      27725
      7934 MILLER           1300      29025
```

```
SQL> select
  2    empno, ename, sal,
  3    sum(sal)
  4     over ( order by empno
  5      rows between unbounded preceding and current row ) as cumtot
  6  from   emp
  7  order by empno;

     EMPNO ENAME             SAL     CUMTOT
---------- ---------- ---------- ----------
      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
      7788 SCOTT            3000      16175
      7839 KING             5000      21175
      7844 TURNER           1500      22675
      7876 ADAMS            1100      23775
      7900 JAMES             950      24725
      7902 FORD             3000      27725
      7934 MILLER           1300      29025
```

```
SQL> select
  2    empno, ename, sal,
  3    sum(sal)
  4     over ( order by empno
  5      rows between unbounded preceding and current row ) as cumtot
  6  from   emp
  7  order by empno;

     EMPNO ENAME             SAL     CUMTOT
---------- ---------- ---------- ----------
      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
      7788 SCOTT            3000      16175
      7839 KING             5000      21175
      7844 TURNER           1500      22675
      7876 ADAMS            1100      23775
      7900 JAMES             950      24725
      7902 FORD             3000      27725
      7934 MILLER           1300      29025
```

```
SQL> select
  2    empno, ename, sal,
  3    sum(sal)
  4     over ( order by empno
  5      rows between unbounded preceding and current row ) as cumtot
  6  from   emp
  7  order by empno;

     EMPNO ENAME             SAL     CUMTOT
---------- ---------- ---------- ----------
      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
      7788 SCOTT            3000      16175
      7839 KING             5000      21175
      7844 TURNER           1500      22675
      7876 ADAMS            1100      23775
      7900 JAMES             950      24725
      7902 FORD             3000      27725
      7934 MILLER           1300      29025
```

"sum across 3 rows"

```
SQL> select deptno, ename, hiredate, sal,
  2    sum(sal) over (
  3      partition by deptno
  4      order by hiredate
  5      rows between 1 preceding and 1 following) as x
  6   from emp
  7  order by deptno, hiredate;

    DEPTNO ENAME      HIREDATE         SAL          X
---------- ---------- --------- ---------- ----------

        10 CLARK      09-JUN-81       2450       7450
        10 KING       17-NOV-81       5000       8750
        10 MILLER     23-JAN-82       1300       6300
        20 SMITH      17-DEC-80        800       3775
        20 JONES      02-APR-81       2975       6775
        20 FORD       03-DEC-81       3000       8975
        20 SCOTT      09-DEC-82       3000       7100
        20 ADAMS      12-JAN-83       1100       4100
        30 ALLEN      20-FEB-81       1600       2850
        30 WARD       22-FEB-81       1250       5700
        30 BLAKE      01-MAY-81       2850       5600
        30 TURNER     08-SEP-81       1500       5600
        30 MARTIN     28-SEP-81       1250       3700
        30 JAMES      03-DEC-81        950       2200
```

```
SQL> select deptno, ename, hiredate, sal,
  2    sum(sal) over (
  3      partition by deptno
  4      order by hiredate
  5      rows between 1 preceding and 1 following) as x
  6   from emp
  7  order by deptno, hiredate;

    DEPTNO ENAME      HIREDATE         SAL          X
---------- ---------- --------- ---------- ----------

        10 CLARK      09-JUN-81       2450       7450
        10 KING       17-NOV-81       5000       8750
        10 MILLER     23-JAN-82       1300       6300
        20 SMITH      17-DEC-80        800       3775
        20 JONES      02-APR-81       2975       6775
        20 FORD       03-DEC-81       3000       8975
        20 SCOTT      09-DEC-82       3000       7100
        20 ADAMS      12-JAN-83       1100       4100
        30 ALLEN      20-FEB-81       1600       2850
        30 WARD       22-FEB-81       1250       5700
        30 BLAKE      01-MAY-81       2850       5600
        30 TURNER     08-SEP-81       1500       5600
        30 MARTIN     28-SEP-81       1250       3700
        30 JAMES      03-DEC-81        950       2200
```

```
SQL> select deptno, ename, hiredate, sal,
  2    sum(sal) over (
  3      partition by deptno
  4      order by hiredate
  5      rows between 1 preceding and 1 following) as x
  6   from emp
  7  order by deptno, hiredate;
```

DEPTNO	ENAME	HIREDATE	SAL	X
10	CLARK	09-JUN-81	2450	7450
10	KING	17-NOV-81	5000	8750
10	MILLER	23-JAN-82	1300	6300
20	SMITH	17-DEC-80	800	3775
20	JONES	02-APR-81	2975	6775
20	FORD	03-DEC-81	3000	8975
20	SCOTT	09-DEC-82	3000	7100
20	ADAMS	12-JAN-83	1100	4100
30	ALLEN	20-FEB-81	1600	2850
30	WARD	22-FEB-81	1250	5700
30	BLAKE	01-MAY-81	2850	5600
30	TURNER	08-SEP-81	1500	5600
30	MARTIN	28-SEP-81	1250	3700
30	JAMES	03-DEC-81	950	2200

```
SQL> select deptno, ename, hiredate, sal,
  2    sum(sal) over (
  3      partition by deptno
  4      order by hiredate
  5      rows between 1 preceding and 1 following) as x
  6   from emp
  7  order by deptno, hiredate;

    DEPTNO ENAME      HIREDATE         SAL          X
---------- ---------- --------- ---------- ----------

        10 CLARK      09-JUN-81       2450       7450
        10 KING       17-NOV-81       5000       8750
        10 MILLER     23-JAN-82       1300       6300
        20 SMITH      17-DEC-80        800       3775
        20 JONES      02-APR-81       2975       6775
        20 FORD       03-DEC-81       3000       8975
        20 SCOTT      09-DEC-82       3000       7100
        20 ADAMS      12-JAN-83       1100       4100
        30 ALLEN      20-FEB-81       1600       2850
        30 WARD       22-FEB-81       1250       5700
        30 BLAKE      01-MAY-81       2850       5600
        30 TURNER     08-SEP-81       1500       5600
        30 MARTIN     28-SEP-81       1250       3700
        30 JAMES      03-DEC-81        950       2200
```

”6 month moving average”

```
SQL> select deptno, ename, hiredate, sal,
  2   avg(sal) over (
  3     partition by deptno
  4     order by hiredate
  5     range between interval '6' month preceding and current row ) x
  6   from emp
  7  order by deptno, hiredate;

    DEPTNO ENAME      HIREDATE         SAL        X
---------- ---------- --------- ---------- --------
        10 CLARK      09-JUN-81       2450     2450
        10 KING       17-NOV-81       5000     3725
        10 MILLER     23-JAN-82       1300     3150
        20 SMITH      17-DEC-80        800      800
        20 JONES      02-APR-81       2975     1888
        20 FORD       03-DEC-81       3000     3000
        20 SCOTT      09-DEC-82       3000     3000
        20 ADAMS      12-JAN-83       1100     2050
        30 ALLEN      20-FEB-81       1600     1600
        30 WARD       22-FEB-81       1250     1425
        30 BLAKE      01-MAY-81       2850     1900
        30 TURNER     08-SEP-81       1500     2175
        30 MARTIN     28-SEP-81       1250     1867
        30 JAMES      03-DEC-81        950     1233
```

```
SQL> select deptno, ename, hiredate, sal,
  2   avg(sal) over (
  3     partition by deptno
  4     order by hiredate
  5     range between interval '6' month preceding and current row ) x
  6   from emp
  7  order by deptno, hiredate;

    DEPTNO ENAME      HIREDATE         SAL          X
---------- ---------- --------- ---------- --------
        10 CLARK      09-JUN-81       2450     2450
        10 KING       17-NOV-81       5000     3725
        10 MILLER     23-JAN-82       1300     3150
        20 SMITH      17-DEC-80        800      800
        20 JONES      02-APR-81       2975     1888
        20 FORD       03-DEC-81       3000     3000
        20 SCOTT      09-DEC-82       3000     3000
        20 ADAMS      12-JAN-83       1100     2050
        30 ALLEN      20-FEB-81       1600     1600
        30 WARD       22-FEB-81       1250     1425
        30 BLAKE      01-MAY-81       2850     1900
        30 TURNER     08-SEP-81       1500     2175
        30 MARTIN     28-SEP-81       1250     1867
        30 JAMES      03-DEC-81        950     1233
```

```
SQL> select deptno, ename, hiredate, sal,
  2   avg(sal) over (
  3     partition by deptno
  4     order by hiredate
  5     range between interval '6' month preceding and current row ) x
  6   from emp
  7  order by deptno, hiredate;

    DEPTNO ENAME      HIREDATE         SAL          X
---------- ---------- --------- ---------- --------
        10 CLARK      09-JUN-81       2450     2450
        10 KING       17-NOV-81       5000     3725
        10 MILLER     23-JAN-82       1300     3150
        20 SMITH      17-DEC-80        800      800
        20 JONES      02-APR-81       2975     1888
        20 FORD       03-DEC-81       3000     3000
        20 SCOTT      09-DEC-82       3000     3000
        20 ADAMS      12-JAN-83       1100     2050
        30 ALLEN      20-FEB-81       1600     1600
        30 WARD       22-FEB-81       1250     1425
        30 BLAKE      01-MAY-81       2850     1900
        30 TURNER     08-SEP-81       1500     2175
        30 MARTIN     28-SEP-81       1250     1867
        30 JAMES      03-DEC-81        950     1233
```

```
SQL> select deptno, ename, hiredate, sal,
  2   avg(sal) over (
  3     partition by deptno
  4     order by hiredate
  5     range between interval '6' month preceding and current row ) x
  6   from emp
  7  order by deptno, hiredate;

    DEPTNO ENAME      HIREDATE         SAL          X
---------- ---------- --------- ---------- --------
        10 CLARK      09-JUN-81       2450     2450
        10 KING       17-NOV-81       5000     3725
        10 MILLER     23-JAN-82       1300     3150
        20 SMITH      17-DEC-80        800      800
        20 JONES      02-APR-81       2975     1888
        20 FORD       03-DEC-81       3000     3000
        20 SCOTT      09-DEC-82       3000     3000
        20 ADAMS      12-JAN-83       1100     2050
        30 ALLEN      20-FEB-81       1600     1600
        30 WARD       22-FEB-81       1250     1425
        30 BLAKE      01-MAY-81       2850     1900
        30 TURNER     08-SEP-81       1500     2175
        30 MARTIN     28-SEP-81       1250     1867
        30 JAMES      03-DEC-81        950     1233
```

dynamic windows

”sum sales from previous close of processing”

```
SQL> create or replace
  2  function LAST_CLOSE(p_purchase_date date)
  3  return number is
  4  begin
  5    return
  6      case to_char(p_purchase_date,'DY')
  7       when 'SUN' then 2
  8       when 'MON' then 3
  9       else 1
 10      end;
 11  end;
 12  /

Function created.

SQL> select
  2    prod_id, cust_id,
  3    sum(amount_sold)
  4     over ( order by purchase_date
  5     range between LAST_CLOSE(purchase_date) preceding) as bus_tot
  6  from   sales
  7  /
```

window boundaries

first_value / last_value

”compare each salary with lowest in organisation and department”

```
SQL> select  deptno, empno, ename, sal,
  2    first_value(sal) over ( order by sal
  3      range unbounded preceding ) lo_sal,
  4    first_value(sal) over ( partition by deptno
  5                            order by sal
  6      range unbounded preceding) lo_dept_sal
  7  from   emp
  8  order by deptno, sal;

    DEPTNO      EMPNO ENAME             SAL     LO_SAL LO_DEPT_SAL
---------- ---------- ---------- ---------- ---------- -----------
        10       7782 CLARK            2450        800        2450
        10       7839 KING             5000        800        2450
        20       7369 SMITH             800        800         800
        20       7876 ADAMS            1100        800         800
        20       7566 JONES            2975        800         800
        20       7902 FORD             3000        800         800
        20       7788 SCOTT            3000        800         800
        30       7900 JAMES             950        800         950
        30       7521 WARD             1250        800         950
        30       7654 MARTIN           1250        800         950
        30       7844 TURNER           1500        800         950
        30       7499 ALLEN            1600        800         950
        30       7698 BLAKE            2850        800         950
        40       7934 MILLER           1300        800        1300
```

```
SQL> select
  2    deptno, empno, ename, sal,
  3    100 * sal / first_value(sal)
  4                  over ( order by sal ) lo_sal_pct,
  5    100 * sal / first_value(sal)
  6                  over ( partition by deptno
  7                         order by sal ) lo_dept_sal_pct
  8  from   emp
  9  order by deptno, sal;
```

DEPTNO	EMPNO	ENAME	SAL	LO_SAL_PCT	LO_DEPT_SAL_PCT
10	7782	CLARK	2450	306.25	100.00
10	7839	KING	5000	625.00	204.08
20	7369	SMITH	800	100.00	100.00
20	7876	ADAMS	1100	137.50	137.50
20	7566	JONES	2975	371.88	371.88
20	7902	FORD	3000	375.00	375.00
20	7788	SCOTT	3000	375.00	375.00
30	7900	JAMES	950	118.75	100.00
30	7521	WARD	1250	156.25	131.58
30	7654	MARTIN	1250	156.25	131.58
30	7844	TURNER	1500	187.50	157.89
30	7499	ALLEN	1600	200.00	168.42
30	7698	BLAKE	2850	356.25	300.00
40	7934	MILLER	1300	162.50	100.00

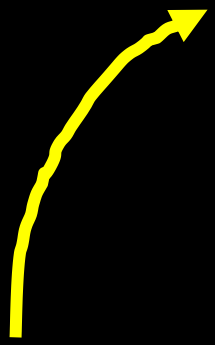
implicit windows

recall

```
SQL> select ename, deptno,
  2    sum(sal) over ( partition by deptno ) as deptsal
  3  from emp
  4  order by deptno;

ENAME          DEPTNO    DEPTSAL
---------- ---------- ----------
CLARK              10       8750
KING               10       8750
MILLER             10       8750
JONES              20      10875
FORD               20      10875
ADAMS              20      10875
SMITH              20      10875
SCOTT              20      10875
WARD               30       9400
TURNER             30       9400
ALLEN              30       9400
JAMES              30       9400
BLAKE              30       9400
MARTIN             30       9400
```

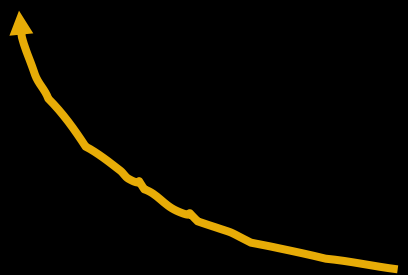
```
SQL> select deptno, empno, ename, job, sal,
  2    sum(sal) OVER (
  3      partition by deptno
  4      order by ename) as running_total
  5  from emp
```

DEPTNO	EMPNO	ENAME	JOB	SAL	RUNNING_TOTAL
10	7782	CLARK	MANAGER	2450	2450
10	7839	KING	PRESIDENT	5000	7450
10	7934	MILLER	CLERK	1300	8750
20	7876	ADAMS	CLERK	1100	1100
20	7902	FORD	ANALYST	3000	4100
20	7566	JONES	MANAGER	2975	7075
20	7788	SCOTT	ANALYST	3000	10075
20	7369	SMITH	CLERK	800	10875
30	7499	ALLEN	SALESMAN	1600	1600
30	7698	BLAKE	MANAGER	2850	4450
30	7900	JAMES	CLERK	950	5400
30	7654	MARTIN	SALESMAN	1250	6650
30	7844	TURNER	SALESMAN	1500	8150
30	7521	WARD	SALESMAN	1250	9400

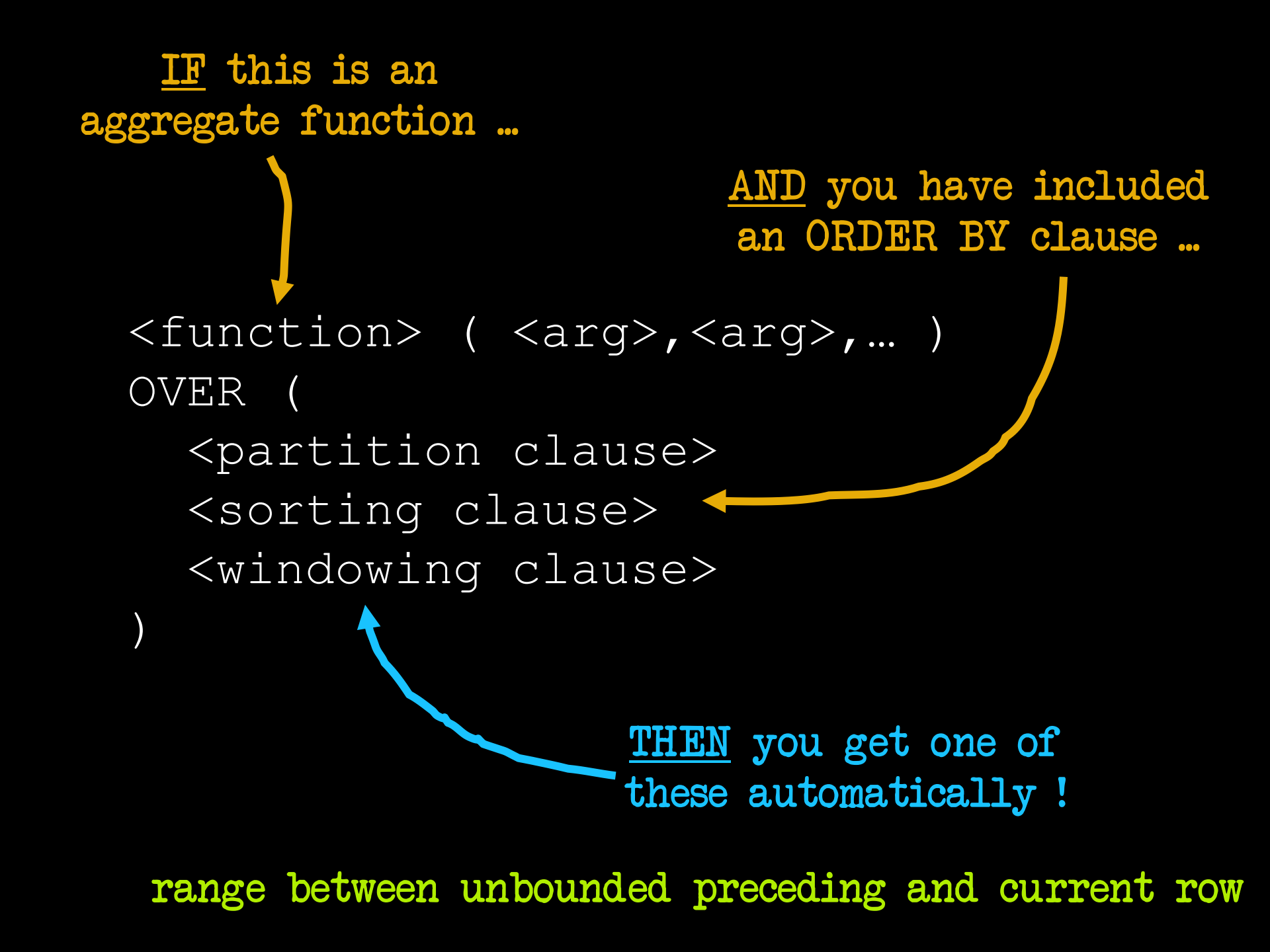


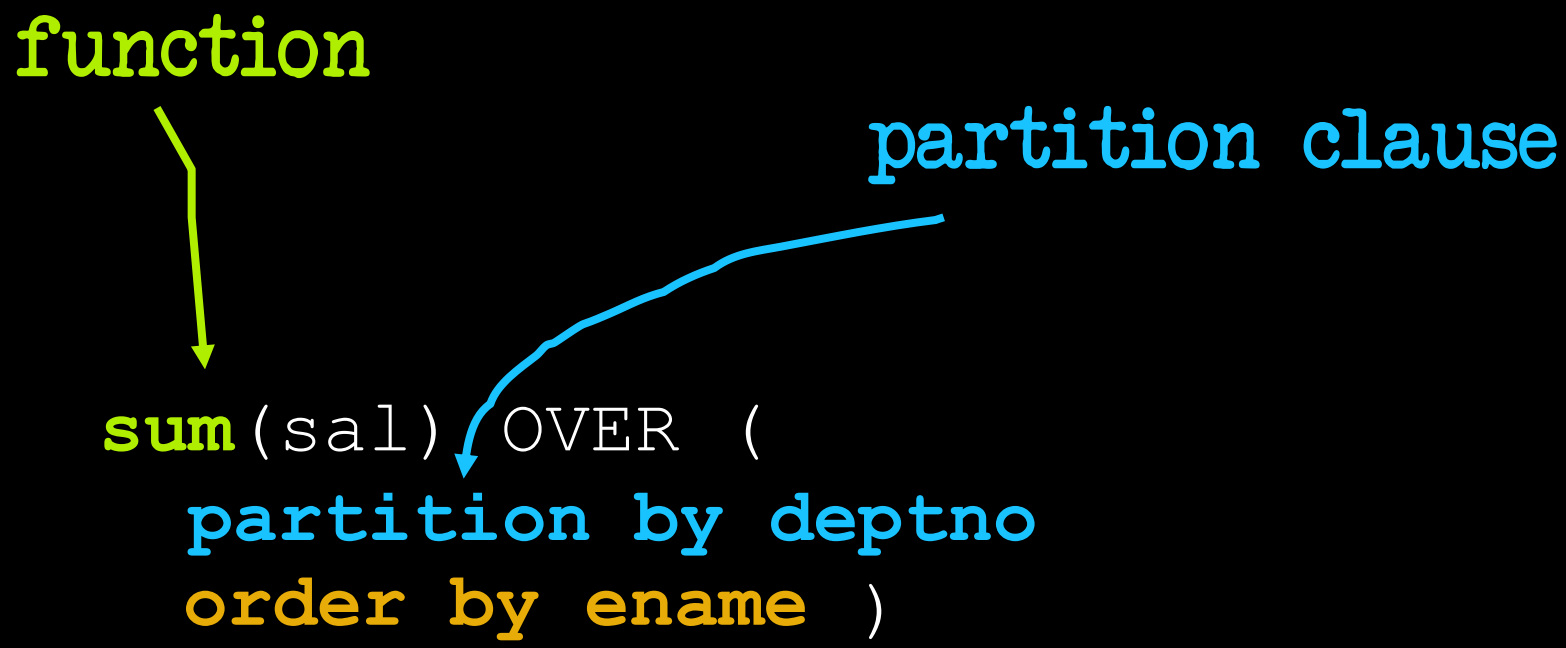
windowing aggregate

```
<function> ( <arg>,<arg>,… )
OVER (
   <partition clause>
   <sorting clause>
   <windowing clause>
)
```



but I didn't specify
one of these ?


IF this is an
aggregate function …
AND you have included
an ORDER BY clause …
<function> (<arg>,<arg>,…)
OVER (
<partition clause>
<sorting clause>
<windowing clause>
)
THEN you get one of
these automatically !
range between unbounded preceding and current row

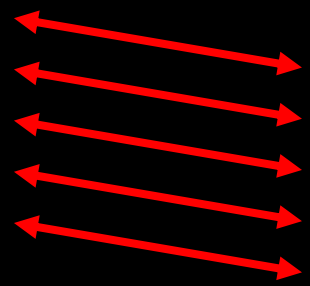

function
partition clause
sum(sal) OVER (
partition by deptno
order by ename)

”OVER”

lag / lead

```
SQL> select
  2    empno, ename, hiredate, sal,
  3    lag(sal,1)
  4      over ( order by hiredate ) prev_hiree_sal
  5  from   emp
  6  order by hiredate;
```

EMPNO	ENAME	HIREDATE	SAL	PREV_HIREE_SAL
7369	SMITH	17-DEC-80	800	
7499	ALLEN	20-FEB-81	1600	800
7521	WARD	22-FEB-81	1250	1600
7566	JONES	02-APR-81	2975	1250
7698	BLAKE	01-MAY-81	2850	2975
7782	CLARK	09-JUN-81	2450	2850
7844	TURNER	08-SEP-81	1500	2450
7654	MARTIN	28-SEP-81	1250	1500
7839	KING	17-NOV-81	5000	1250
7900	JAMES	03-DEC-81	950	5000
7902	FORD	03-DEC-81	3000	950
7934	MILLER	23-JAN-82	1300	3000
7788	SCOTT	09-DEC-82	3000	1300
7876	ADAMS	12-JAN-83	1100	3000



and then....

“imagination is more important than knowledge”

- Albert Einstein

NOT just warehousing

classical problems
made simple

”remove the duplicates”

```
SQL> select * from BAD_EMP;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL
---------- ---------- --------- ---------- --------- ----------
      7788 SCOTT      ANALYST         7566 09-DEC-82       3000
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600
      7369 SMITH      CLERK           7902 17-DEC-80        800
      7839 KING       PRESIDENT            17-NOV-81       5000
      7566 JONES      MANAGER         7839 02-APR-81       2975
      7876 ADAMS      CLERK           7788 12-JAN-83       1100
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600
      7369 SMITH      CLERK           7902 17-DEC-80        800
      7902 FORD       ANALYST         7566 03-DEC-81       3000
      7900 JAMES      CLERK           7698 03-DEC-81        950
      7521 WARD       SALESMAN        7698 22-FEB-81       1250
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250
      7934 MILLER     CLERK           7782 23-JAN-82       1300
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850
      7782 CLARK      MANAGER         7839 09-JUN-81       2450
```

```
SQL> select * from BAD_EMP;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL
---------- ---------- --------- ---------- --------- ----------
      7788 SCOTT      ANALYST         7566 09-DEC-82       3000
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600
      7369 SMITH      CLERK           7902 17-DEC-80        800
      7839 KING       PRESIDENT            17-NOV-81       5000
      7566 JONES      MANAGER         7839 02-APR-81       2975
      7876 ADAMS      CLERK           7788 12-JAN-83       1100
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600
      7369 SMITH      CLERK           7902 17-DEC-80        800
      7902 FORD       ANALYST         7566 03-DEC-81       3000
      7900 JAMES      CLERK           7698 03-DEC-81        950
      7521 WARD       SALESMAN        7698 22-FEB-81       1250
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250
      7934 MILLER     CLERK           7782 23-JAN-82       1300
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850
      7782 CLARK      MANAGER         7839 09-JUN-81       2450
```

```
SQL> select empno, rowid,
  2         row_number() over
  3          ( partition by empno order by rowid ) as r
  4  from BAD_EMP
  5  /

     EMPNO ROWID                  R
---------- ------------------ ----------
      7369 AAARXwAAEAAATlMAAA          1
      7369 AAARXwAAEAAATlOAAA          2
      7499 AAARXwAAEAAATlMAAB          1
      7499 AAARXwAAEAAATlOAAB          2
      7521 AAARXwAAEAAATlMAAC          1
      7566 AAARXwAAEAAATlMAAD          1
      7654 AAARXwAAEAAATlMAAE          1
      7698 AAARXwAAEAAATlMAAF          1
      7782 AAARXwAAEAAATlMAAG          1
      7788 AAARXwAAEAAATlMAAH          1
      7839 AAARXwAAEAAATlMAAI          1
      7844 AAARXwAAEAAATlMAAJ          1
      7876 AAARXwAAEAAATlMAAK          1
      7900 AAARXwAAEAAATlMAAL          1
      7902 AAARXwAAEAAATlMAAM          1
      7934 AAARXwAAEAAATlMAAN          1
```

```
SQL> delete from BAD_EMP
  2  where ROWID in
  3  ( select rowid
  4    from
  5    ( select rowid,
  6             row_number() over
  7               ( partition by empno
  8                 order by rowid) as r
  9      from BAD_EMP
 10    )
 11    where r > 1
 12  )
 13  /

2 rows deleted.
```

"5 highest salaries from each department"

```
SQL> select deptno, salary
  2  from
  3  ( select
  4      deptno,
  5      salary,
  6      rank() over (
  7        partition by deptno
  8        order by salary) top_5
  9    from EMPLOYEES
 10  )
 11  where top_5 <= 5
```

“mind the gap”

```
SQL> select X from T;

         X
----------
         2
         3
         4
         7
         8
        12
        13
        15
        16
        17
        19
        20
```

2-4

7-8

12-13

15-17

19-20

```
SQL> select
  2    x,
  3    lag(x) over ( order by x) prev
  4  from T  ;

         X       PREV
---------- ----------
         2
         3          2
         4          3
         7          4
         8          7
        12          8
        13         12
        15         13
        16         15
        17         16
        19         17
        20         19
```

```
SQL> select
  2    x,
  3    lag(x) over ( order by x) prev
  4  from T  ;

         X       PREV
---------- ----------
         2
         3          2
         4          3
         7          4
         8          7
        12          8
        13         12
        15         13
        16         15
        17         16
        19         17
        20         19
```

```
SQL> select
  2    x,
  3    lag(x) over ( order by x) prev
  4  from T  ;

         X       PREV
---------- ----------
         2
         3          2
         4          3
         7          4
         8          7
        12          8
        13         12
        15         13
        16         15
        17         16
        19         17
        20         19
```

```
SQL> select
  2    x,
  3    lag(x) over ( order by x) prev
  4  from T  ;

         X       PREV
---------- ----------
         2
         3          2
         4          3
         7          4
         8          7
        12          8
        13         12
        15         13
        16         15
        17         16
        19         17
        20         19
```

```
SQL> select
  2    x,
  3    lag(x) over ( order by x) prev
  4  from T  ;

         X       PREV
---------- ----------
         2
         3          2
         4          3
         7          4
         8          7
        12          8
        13         12
        15         13
        16         15
        17         16
        19         17
        20         19
```

```
SQL> select
  2    x,
  3    case
  4      when nvl(lag(x) over (order by x),x) != x-1
  5         then x end loval
  6  from t;

         X      LOVAL
---------- ----------
         2          2
         3
         4
         7          7
         8
        12         12
        13
        15         15
        16
        17
        19         19
        20
```

```
SQL> select
  2    x,
  3    case
  4      when nvl(lag(x) over (order by x),x) != x-1
  5         then x end loval
  6  from t;

         X      LOVAL
---------- ----------
         2          2
         3
         4
         7          7
         8
        12         12
        13
        15         15
        16
        17
        19         19
        20
```

```
SQL> select
  2    x,
  3    case
  4      when nvl(lag(x) over (order by x),x) != x-1
  5         then x end loval
  6  from t;

         X      LOVAL
---------- ----------
         2          2
         3
         4
         7          7
         8
        12         12
        13
        15         15
        16
        17
        19         19
        20
```

```
SQL> select
  2    x,
  3    case
  4      when nvl(lag(x) over (order by x),x) != x-1
  5         then x end loval
  6  from t;

         X      LOVAL
---------- ----------
         2          2
         3
         4
         7          7
         8
        12         12
        13
        15         15
        16
        17
        19         19
        20
```

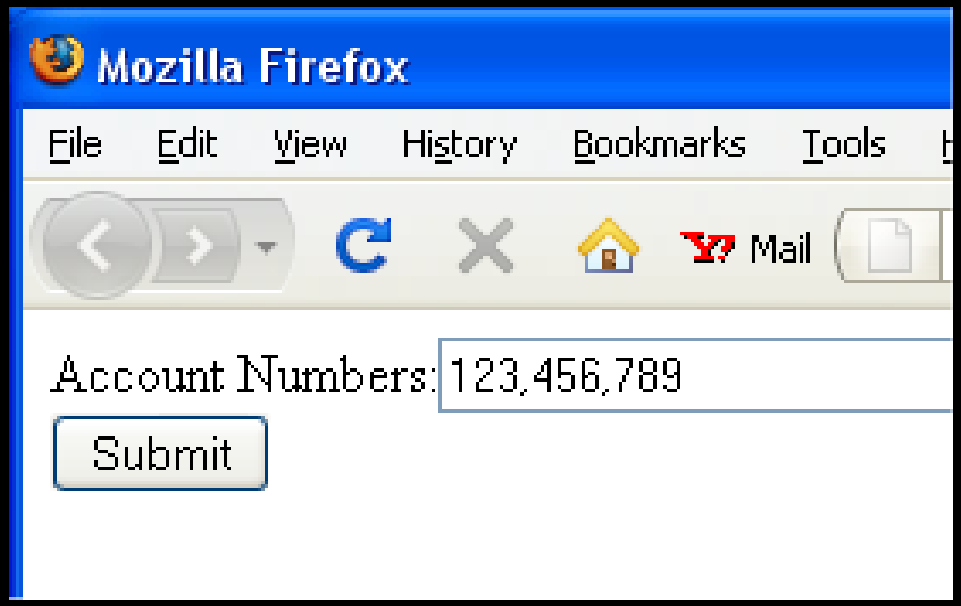
```
SQL> select x, max(loval) over (order by x) loval
  2  from (
  3  select x,
  4         case
  5           when nvl(lag(x) over (order by x),x) != x-1
  6             then x end loval
  7  from t );

         X      LOVAL
---------- ----------
         2          2
         3          2
         4          2
         7          7
         8          7
        12         12
        13         12
        15         15
        16         15
        17         15
        19         19
        20         19
```

```
SQL> select x, max(loval) over (order by x) loval
  2  from (
  3  select x,
  4         case
  5           when nvl(lag(x) over (order by x),x) != x-1
  6             then x end loval
  7  from t );

         X      LOVAL
---------- ----------
         2          2
         3          2
         4          2
         7          7
         8          7
        12         12
        13         12
        15         15
        16         15
        17         15
        19         19
        20         19
```

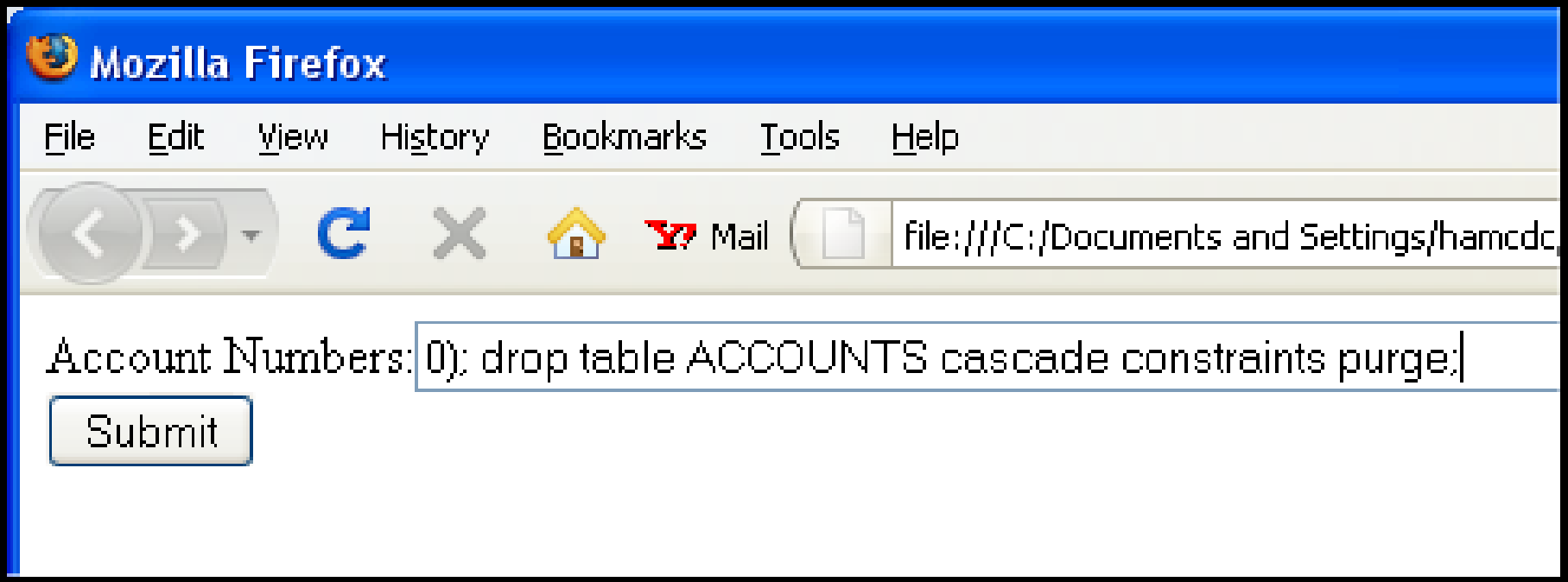
```
SQL> select min(x)||'-'||max(x) ranges from (
  2  select x,max(loval) over (order by x) loval
  3  from (
  4  select x,
  5         case
  6           when nvl(lag(x) over (order by x),x) != x-1
  7             then x end loval
  8  from t))
  9  group by loval;

RANGES
--------------------
2-4
7-8
12-13
15-17
19-20
```

```
SQL> select
  2    min(x)||'-'||
  3    case when min (x) = max (x)
  4         then min(x)
  5         else max(x)
  6    end rng
  7  from
  8    (select X
  9          , row_number() over (order by X) rn
 10       from t
 11    )
 12  group by x - rn
 13  order by min(x);

RNG
---------------------------------------------------
2-4
7-8
12-13
15-17
19-20
```

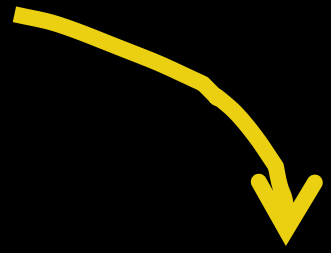
in-list processing




```
sql_string =
  "select * from ACCOUNTS where ACCT_NO in ( " + :acct_input + ")"


EXEC SQL PREPARE sql_string;
```


Mozilla Firefox
File Edit View History Bookmarks Tools Help
Mail
file:///C:/Documents and Settings/hamcdc
Account Numbers: 0); drop table ACCOUNTS cascade constraints purge;
Submit

123,456,789



```
sql_string =
  "select * from ACCOUNTS where ACCT_NO in ( :bindvar  )"


EXEC SQL PREPARE sql_string;
```

ORA-01722: invalid number

```
SQL> ticket = '123,456,789'
SQL> select substr(:ticket,
  2           loc+1,nvl(
  3           lead(loc) over ( order by loc ) - loc-1,
  4           length(:ticket)-loc)
  5         ) list_as_rows
  6  from (
  7    select distinct (instr(:ticket,',',1,level)) loc
  8    from dual
  9    connect by level < length(:ticket)-
 10         length(replace(:ticket,','))+1
 11    );

LIST_AS_ROWS
--------------------
123
456
789
```

```
SQL> with MY_LIST as select substr(:ticket,
  2           loc+1,nvl(
  3           lead(loc) over ( order by loc ) - loc-1,
  4           length(:ticket)-loc)
  5         ) val
  6  from (
  7    select distinct (instr(:ticket,',',1,level)) loc
  8    from dual
  9    connect by level < length(:ticket)-
 10         length(replace(:ticket,','))+1
 11    )
 12  select *
 13  from   ACCOUNTS
 14  where  ACCT_NO in ( select val from MY_LIST)
```

other goodies

FIRST / LAST extension

"Show me lowest salary for each department…"

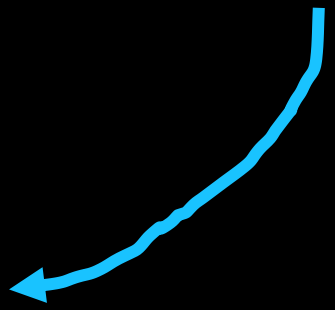
```
SQL> select deptno, min(sal)
  2  from emp
  3  group by deptno;
```

"…and I need to know who has that lowest salary as well"

```
SQL> select deptno, empno, min(sal)
  2  from emp
  3  group by deptno;

ORA-00979: not a GROUP BY expression
```

```
SQL> select deptno, min(sal), min(empno)
  2    KEEP ( dense_rank FIRST order by sal) empno
  3  from emp
  4  group by deptno
  5  /
```

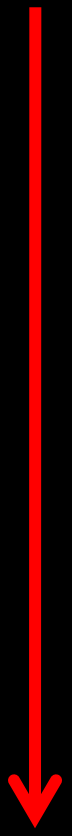
Emp 7934 has the lowest salary in dept 10



DEPTNO	MIN(SAL)	EMPNO
10	1300	7934
20	800	7369
30	950	7900

ignore nulls extension

```
SQL> select empno, ename, sal, deptno
  2  from   emp
  3  order by sal;

     EMPNO ENAME             SAL     DEPTNO
---------- ---------- ---------- ----------
      7369 SMITH             800         10
      7900 JAMES             950
      7876 ADAMS            1100
      7521 WARD             1250         20
      7654 MARTIN           1250
      7934 MILLER           1300
      7844 TURNER           1500
      7499 ALLEN            1600         30
      7782 CLARK            2450
      7698 BLAKE            2850
      7566 JONES            2975
      7788 SCOTT            3000
      7902 FORD             3000
      7839 KING             5000         40
```



```
SQL> select empno, ename, sal, deptno,
  2         last_value(deptno IGNORE NULLS)
  3           over (order by sal) as last_dept
  4  from emp
  5  order by sal

     EMPNO ENAME             SAL     DEPTNO  LAST_DEPT
---------- ---------- ---------- ---------- ----------
      7369 SMITH             800         10         10
      7900 JAMES             950                    10
      7876 ADAMS            1100                    10
      7521 WARD             1250         20         20
      7654 MARTIN           1250                    20
      7934 MILLER           1300                    20
      7844 TURNER           1500                    20
      7499 ALLEN            1600         30         30
      7782 CLARK            2450                    30
      7698 BLAKE            2850                    30
      7566 JONES            2975                    30
      7788 SCOTT            3000                    30
      7902 FORD             3000                    30
      7839 KING             5000         40         40
```

```
SQL> select last_dept, count(*)
  2  from
  3  ( select
  4      last_value(deptno ignore nulls)
  5        over (order by sal) as last_dept
  6    from emp2
  7  )
  8  group by last_dept;

 LAST_DEPT   COUNT(*)
---------- ----------
        30          6
        20          4
        40          1
        10          3
```

inverse analytics

recall

cume_dist

```
SQL> select ename, sal,
  2    100*cume_dist() over ( order by sal ) as pct
  3  from emp
  4  order by sal;

ENAME             SAL     PCT
---------- ---------- -------
SMITH             800    7.14
JAMES             950   14.29
ADAMS            1100   21.43
WARD             1250   35.71
MARTIN           1250   35.71
MILLER           1300   42.86
TURNER           1500   50.00
ALLEN            1600   57.14
CLARK            2450   64.29
BLAKE            2850   71.43
JONES            2975   78.57
FORD             3000   92.86
SCOTT            3000   92.86
KING             5000  100.00
```

"what is the 60th percentile salary ?"

```
SQL> select ename, sal,
  2    100*cume_dist() over ( order by sal ) as pct
  3  from emp
  4  order by sal;

ENAME             SAL     PCT
---------- ---------- -------
SMITH             800    7.14
JAMES             950   14.29
ADAMS            1100   21.43
WARD             1250   35.71
MARTIN           1250   35.71
MILLER           1300   42.86
TURNER           1500   50.00
ALLEN            1600   57.14
CLARK            2450   64.29
BLAKE            2850   71.43
JONES            2975   78.57
FORD             3000   92.86
SCOTT            3000   92.86
KING             5000  100.00
```

```
SQL> select ename, sal,
  2    100*cume_dist() over ( order by sal ) as pct
  3  from emp
  4  order by sal;

ENAME             SAL     PCT
---------- ---------- -------
SMITH             800    7.14
JAMES             950   14.29
ADAMS            1100   21.43
WARD             1250   35.71
MARTIN           1250   35.71
MILLER           1300   42.86
TURNER           1500   50.00
ALLEN            1600   57.14
CLARK            2450   64.29
BLAKE            2850   71.43
JONES            2975   78.57
FORD             3000   92.86
SCOTT            3000   92.86
KING             5000  100.00
```

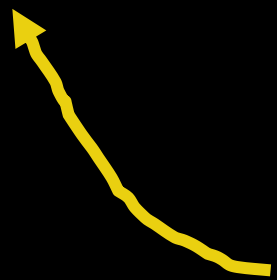
”WTIHIN GROUP”

```
SQL> select
  2       percentile_disc(0.6)
  3          within group
  4              (order by sal) as dicrete_pct,
  5       percentile_cont(0.6)
  6          within group
  7              (order by sal) as continuous_pct
  8  from emp;

DICRETE_PCT CONTINUOUS_PCT
----------- --------------
       2450           2280
```

classical problems
made simple

”median salary for each department?”

```
SQL> select deptno, avg(distinct sal) median
  2  from
  3    (select cp1.deptno, cp1.sal
  4     from emp cp1, emp cp2
  5     where cp1.deptno = cp2.deptno
  6     group by cp1.deptno, cp1.sal
  7     having sum(decode(cp1.sal, cp2.sal, 1, 0)) >=
  8                  abs(sum(sign(cp1.sal - cp2.sal))))
  9  group by deptno
 10  /

    DEPTNO     MEDIAN
---------- ----------
        10       3725
        20       2975
        30       1375
        40       1300
```



huh?

```
SQL> select
  2       deptno,
  3       percentile_cont(0.5)
  4          within group (order by sal) as median
  5  from emp
  6  group by deptno;

SQL> select
  2       deptno,
  3       median(sal)
  4  from emp
  5  group by deptno;

    DEPTNO MEDIAN(SAL)
---------- -----------
        10        3725
        20        2975
        30        1375
        40        1300
```

hypothetical analytics

"if I was paid $3000, where would I rank in each department?"

```
SQL> select
  2       deptno,
  3       rank(3000) within group
  4         ( order by sal ) as ranking
  5  from emp
  6  group by deptno;

    DEPTNO    RANKING
---------- ----------
        10          2
        20          4
        30          7
        40          2
```

ratio_to_report

"salary percentage breakdown across employees"

```
SQL> select
  2       empno,
  3       ename,
  4       sal,
  5       100*ratio_to_report(sal) over () as pct
  6  from emp;

     EMPNO ENAME                SAL        PCT
---------- ---------- ---------- -------
      7521 WARD             1250    4.69
      7566 JONES            2975   11.17
      7654 MARTIN           1250    4.69
      7698 BLAKE            2850   10.70
      7782 CLARK            2450    9.20
      7788 SCOTT            3000   11.27
      7839 KING             5000   18.78
      7844 TURNER           1500    5.63
      7876 ADAMS            1100    4.13
      7900 JAMES             950    3.57
      7902 FORD             3000   11.27
      7934 MILLER           1300    4.88
```

partitioned outer join

not really analytic ?

```
SQL> select * from bookings;

        HR ROOM       WHO
---------- ---------- -------
         8 Room2      PETE
         9 Room1      JOHN
        11 Room1      MIKE
        14 Room2      JILL
        15 Room2      JANE
        16 Room1      SAM
```

```
SQL> select * from hrs;

HR
--
 8
 9
10
11
12
13
14
15
16
```

bookings by hour

(conventional outer join)_

```
SQL> SELECT hrs.hr, t1.room, t1.who
  2  from hrs, bookings t1
  3  where  hrs.hr = t1.hr(+)

        HR ROOM       WHO
---------- ---------- ----------
         8 Room2      PETE
         9 Room1      JOHN
        10
        11 Room1      MIKE
        12
        13
        14 Room2      JILL
        15 Room2      JANE
        16 Room1      SAM
```

room occupancy

(*partitioned* outer join)_

```
SQL> SELECT hrs.hr, t1.room, t1.who
  2  FROM   bookings t1
  3  PARTITION BY  (t1.room)
  4  RIGHT OUTER JOIN hrs ON (hrs.hr = t1.hr);

        HR ROOM       WHO
---------- ---------- ----------
         8 Room1
         9 Room1      JOHN
        10 Room1
        11 Room1      MIKE
        12 Room1
        13 Room1
        14 Room1
        15 Room1
        16 Room1      SAM
         8 Room2      PETE
         9 Room2
        10 Room2
        11 Room2
        12 Room2
        13 Room2
        14 Room2      JILL
        15 Room2      JANE
        16 Room2
```

things to note

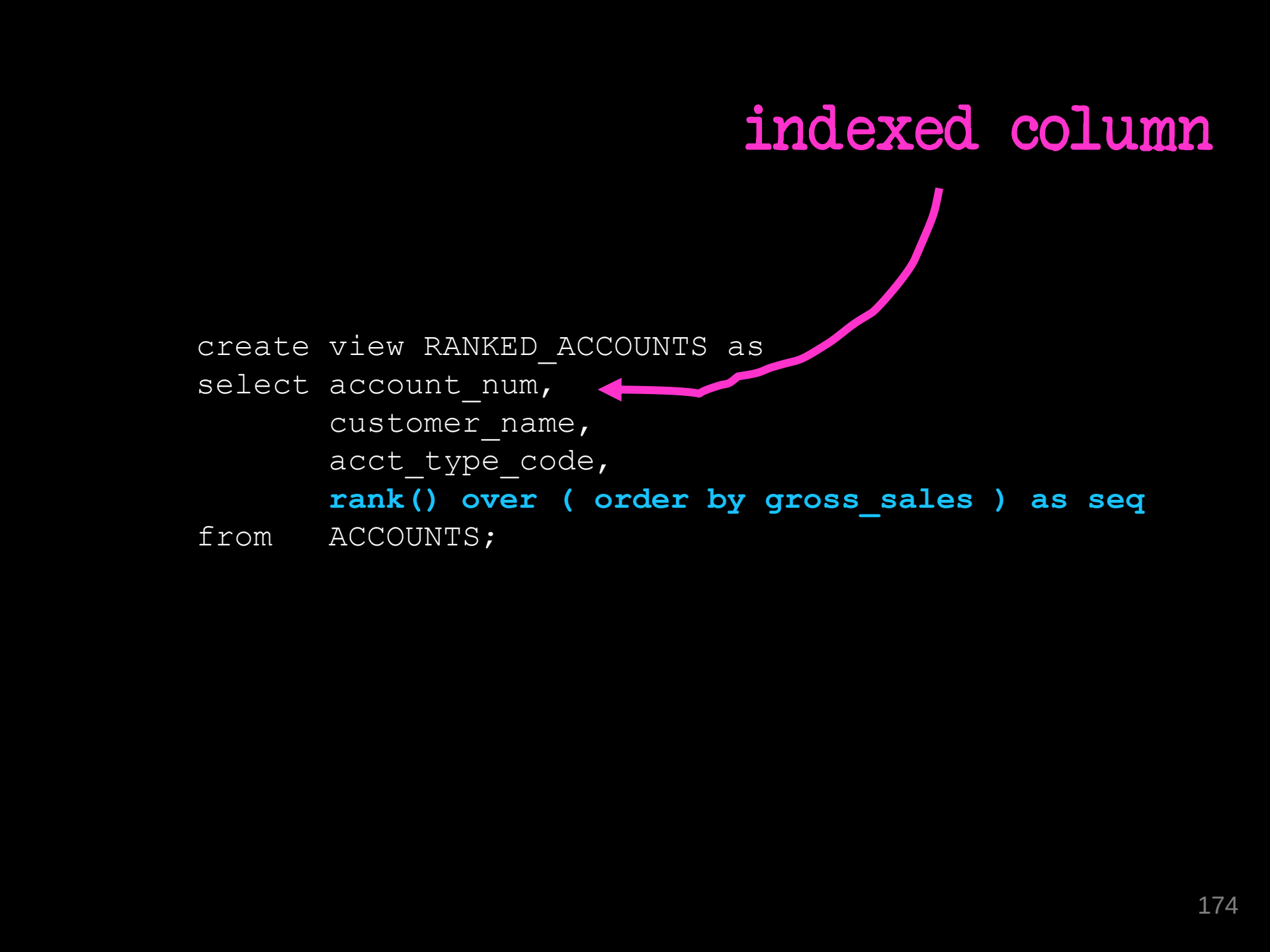
cannot be a predicate

```
SQL> select ename, deptno, sal
  2  from emp
  3  where
  4    sum(sal) over
  5      ( partition by deptno) > 10;
  sum(sal) over
  *
ERROR at line 4:
ORA-00934: group function is not allowed here
```

inline view

```
SQL> select ename, deptno, sal
  2  from (
  3    select ename, deptno, sal,
  4      sum(sal) over
  5           ( partition by deptno) as deptsal
  6    from emp
  7    )
  8  where deptsal > 10;
```

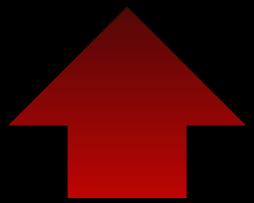
careful with views…


indexed column
create view RANKED_ACCOUNTS as
select account_num,
customer_name,
acct_type_code,
rank() over (order by gross_sales) as seq
from ACCOUNTS;

```
SQL> select * from RANKED_ACCOUNTS
  2  where  ACCOUNT_NUM = 12345

-------------------------------------------
| Id  | Operation           | Name        |
-------------------------------------------
|   0 | SELECT STATEMENT    |             |
|   1 |  VIEW               | RANKED_ACCOUNTS |
|   2 |   WINDOW SORT       |             |
|   3 |    TABLE ACCESS FULL| ACCOUNTS    |
-------------------------------------------
```

a more holistic view

question



solution

frequent itemsets

```
SQL> select DEMO
  2  from   DEMOGRAPHIC;

DE
--
Ch
Te
Ad
```

```
SQL> select SEX
  2  from   GENDER;

SE
--
Ma
Fe
Un
```

```
SQL> select CEREAL_NAME
  2  from   BRANDS;

CEREAL_NAME
------------
CornFlakes
Weetbix
CocoPops
FrootLoops
NutriGrain
AllBran
SultanaBran
Porridge
JustRight
Rice Bubbles
```

```
SQL> desc BREAKFAST_FOOD

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 CUST_ID                                            NUMBER(38)
 DEMOGRAPHIC                                        VARCHAR2(20)
 GENDER                                             VARCHAR2(20)
 BRAND                                              VARCHAR2(20)
 RATING                                             VARCHAR2(20)
```

```
SQL> select *
  2  from BREAKFAST_FOOD;

 CUST_ID DEMOGRAPHIC       GENDER         BRAND           RATING
-------- ----------------- ------------- --------------- --------
       1 Child             Male          Weetbix         Hate
       2 Teenager          Female        AllBran         Hate
       3 Child             Male          FrootLoops      Love
       4 Adult             Female        AllBran         Love
       5 Child             Male          Weetbix         Hate
       6 Adult             Male          FrootLoops      Love
       7 Child             Female        SultanaBran     Hate
       8 Child             Female        Weetbix         Hate
       9 Teenager          Male          Weetbix         Hate
      10 Child             Female        NutriGrain      OK
      11 Teenager          Male          Rice Bubbles    OK
      12 Child             Male          CornFlakes      Hate
```

```
SQL> create view CUSTOMER_ATTRIBUTES as
  2  SELECT cust_id, demographic
  3  FROM   breakfast_food
  4  UNION ALL
  5  SELECT cust_id, brand
  6  FROM   breakfast_food
  7  UNION ALL
  8  SELECT cust_id, rating
  9  FROM   breakfast_food
 10  UNION ALL
 11  SELECT cust_id, gender
 12  FROM   breakfast_food
 13  /

View created.
```

```
SQL> create or replace type VC_LIST
  2  as table of varchar2(30);
  3  /

Type created.
```

```
SQL> SELECT
  2    CAST (itemset as vc_list) itemset
  3   ,round(100*support/total_tranx,2) pct
  4  FROM
  5    TABLE(DBMS_FREQUENT_ITEMSET.FI_TRANSACTIONAL
  6              (  cursor
  7                  ( SELECT * FROM CUSTOMER_ATTRIBUTES  )
  8                , 0.03 -- threshold
  9                , 3
 10                , 4
 11                , cursor
 12                   (SELECT cereal_name FROM brands)
 13                , NULL
 14                )
 15             )
 16  order by 2 desc
 17  /
```

```
ITEMSET                                                  PCT
-------------------------------------------------- ----------
VC_LIST('AllBran', 'Child', 'Hate')                        6.27
VC_LIST('AllBran', 'Hate', 'Male')                         4.65
VC_LIST('Child', 'Hate', 'JustRight')                      4.40
VC_LIST('CornFlakes', 'Hate', 'Male')                      4.27
VC_LIST('AllBran', 'Child', 'Male')                        4.20
VC_LIST('Child', 'Hate', 'SultanaBran')                    4.15
VC_LIST('Adult', 'FrootLoops', 'Hate')                     4.12
VC_LIST('Child', 'CornFlakes', 'Hate')                     4.06
VC_LIST('Child', 'Hate', 'Weetbix')                        4.04
VC_LIST('Adult', 'CocoPops', 'Hate')                       3.81
VC_LIST('AllBran', 'Child', 'Hate', 'Male')                3.78
VC_LIST('Adult', 'AllBran', 'Love')                        3.75

[snip]

28 rows selected.
```

“what will my child like
for breakfast that is not CocoPops”

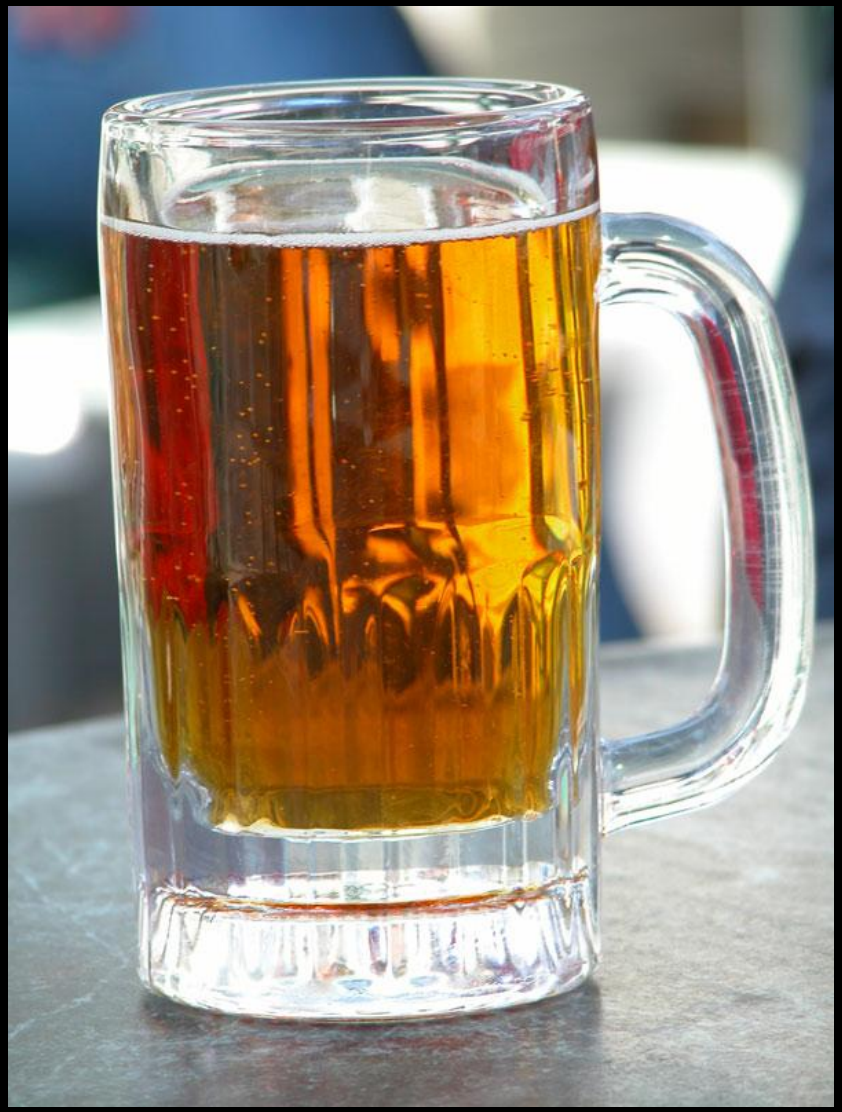
```
SQL> SELECT CAST (itemset as vc_list) itemset
  2  FROM
  3    table( DBMS_FREQUENT_ITEMSET.FI_TRANSACTIONAL
  4               (  cursor
  5                   ( SELECT * FROM CUSTOMER_ATTRIBUTES  )
  6                  , 0.03 -- threshold
  7                  , 3
  8                  , 4
  9                  , cursor
 10                    (SELECT cereal_name FROM brands)
 11                  , NULL))
 12  where 'Child' member of itemset
 13  and 'CocoPops' not member of itemset
 14  and 'Love' member of itemset
 15  /

ITEMSET
--------------------------------------------------
VC_LIST('Child', 'FrootLoops', 'Love')
```

wrap up

analytics

cool

less code



ora-3113

www.oracledba.co.uk