Not Using Analytics?

Shame on You!

Travis R. Rogers

SrVP of Database Development at
CDG Management LLC

travis@jerseyrogers.com

Session Goals

1. Define Oracle Analytics.
2. Learn why analytics should be used.
3. Learn enough to get started.
4. See some examples.

What is Oracle Analytics?

Part 1

Oracle Docs Definition #1

Analytic functions compute an aggregate value based on a group of rows. They differ from aggregate functions in that they return multiple rows for each group.

Oracle® Database SQL Reference Ch5

Oracle Docs Definition #2

Analytic functions (formerly called **window**, or windowing functions) enable you to compute various cumulative, moving, and centered aggregates over a set of rows called a window.

Oracle® Database Data Cartridge Developer's Guide Ch 11

Oracle Docs Definition #3

It provides access to more than one row of a table at the same time without a self join.

Various places in Oracle Docs

Oracle Docs Definition #4

Chapter 21 of **Oracle® Database Data Warehousing Guide** has a pretty thorough discussion and is a good place to start.

Why is it called Analytics?

I have no idea but I have some theories:

- Window Functions is a boring name.
- Analytics is a cool name.
- Analytics was/is a hot topic.
- Businesses pay for Analytics but hate paying for windows ;).
- This functionality is useful in OLAP analysis.
- Analytics is a free word.

Wikipedia says:

...In reality, the word "Analytics" has not been properly defined by the professional community and may mean different things to different people...

...Common applications of Analytics include the study of business data using statistical analysis in order to discover and understand historical patterns with an eye to predicting and improving business performance in the future...

What is Oracle Analytics?

- A SQL Extension.
- Available since v8.1.6.
- Part of ANSI SQL 2003 OLAP Functions (Non-Core Feature ID T611).
- AKA
  - Analytic Functions.
  - Window or Windowing Functions (ANSI)

What is Oracle Analytics?

- A great set of functionality.
- Poorly defined.
- Poorly named.

So we need a new name and a new definition.

Uhhh…I got nothin’.

Why are Analytics not used more often?

Part 2

Why not used?

- My app/project must connect to multiple databases therefore the SQL must be generic.

OR

- My company/manager says I have to use standard/generic SQL.

Why not used?

- I’m not doing “analytical” work.
- It’s hard.

OR

- It’s hard to read.
- I can do the same thing using other SQL and/or PLSQL so there is no reason.
- I’m waiting on the Tom Kyte book.

Why use Analytics?

Part 3

Why use?

- Faster.
- Easier.
- More readable.
- If you don’t, somebody else will.

The Basics

Part 4

How I see Oracle Analytics

- The ability to query a subset of the data from the primary result set (the subset is referred to as the window).
- The subset query will return a single value per row and is presented as a column in the final query output.
- This can be done multiple times in a single query.

Primary Result Set Example

```
SELECT   empno
FROM     emp;
```

Primary Result Set Example

```
SELECT   e.empno, d.loc
FROM     emp e
         ,dept d
WHERE    e.deptno = d.deptno
AND e.job = 'CLERK';
```

e.* and d.*

AFTER filter is applied

Subset Query Example (ie Analytic Function)

```
SELECT empno, sal, deptno
      ,SUM(sal) over
       (PARTITION BY deptno)
       AS dept_sal
FROM   emp
```

emp.*

Translates to: get the salary for every employee in the same department as this employee and return the sum.

Subset Query Example (ie Analytic Function)

EMPNO	SAL	DEPTNO	DEPT_SAL
7782	2450.00	10	8750
7839	5000.00	10	8750
...			
7566	2975.00	20	10875
7902	3000.00	20	10875
...			
7521	1250.00	30	9400
7844	1500.00	30	9400
...			

Alternative 1 – Inline View

```
SELECT
  e1.empno, e1.sal, e1.deptno
  ,e2.d_sal
FROM
  emp e1,
  (SELECT deptno, sum(sal) AS d_sal
   FROM emp
   GROUP BY deptno) e2
WHERE
  e1.deptno = e2.deptno
```

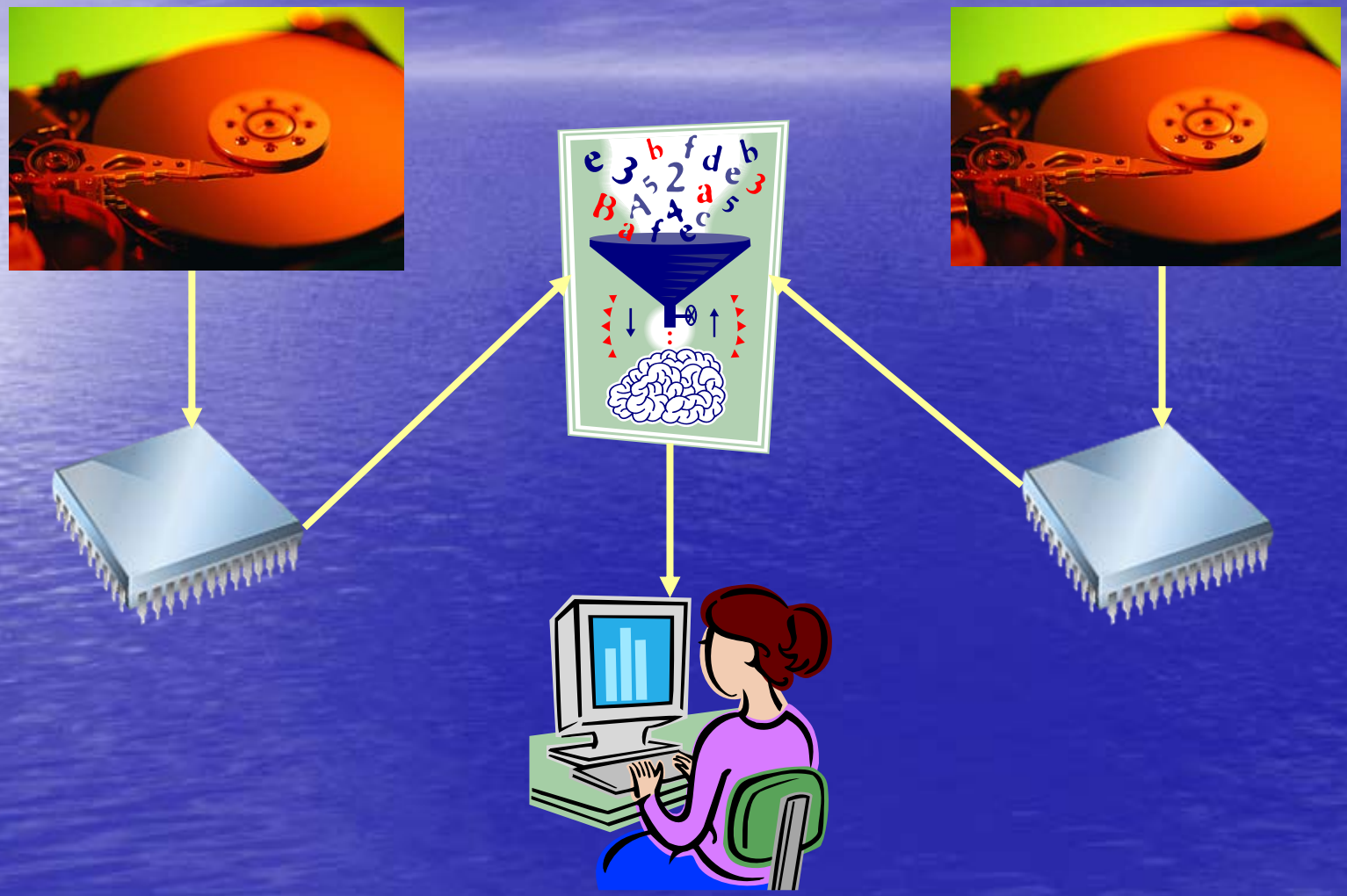
Alternative 2 – Scalar SubQuery

```
SELECT
 e1.empno, e1.sal, e1.deptno
 ,(SELECT sum(sal)
   FROM   emp
   WHERE  deptno = e1.deptno)
   AS dept_sal
FROM
 emp e1
```

Alternative 3 – PLSQL

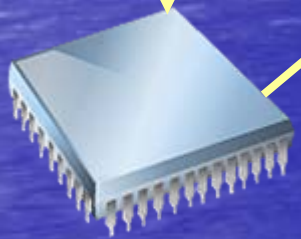
TOO MUCH CODE FOR ONE PAGE...

What happens – Sub Query

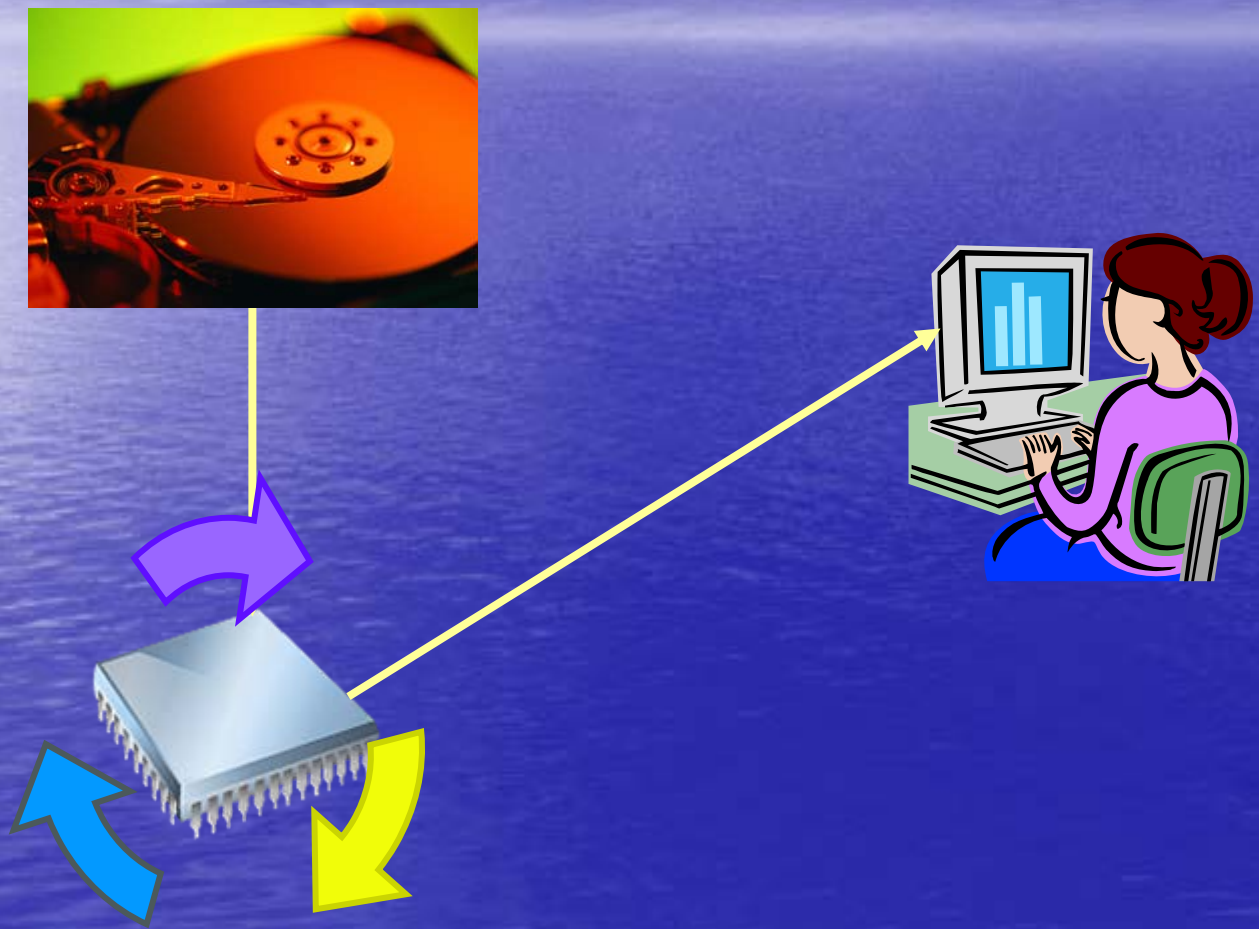


What happens – PL/SQL

```
...
BEGIN
   FOR c in (SELECT...
   LOOP
      ...
      LOOP
```



What happens – Analytics

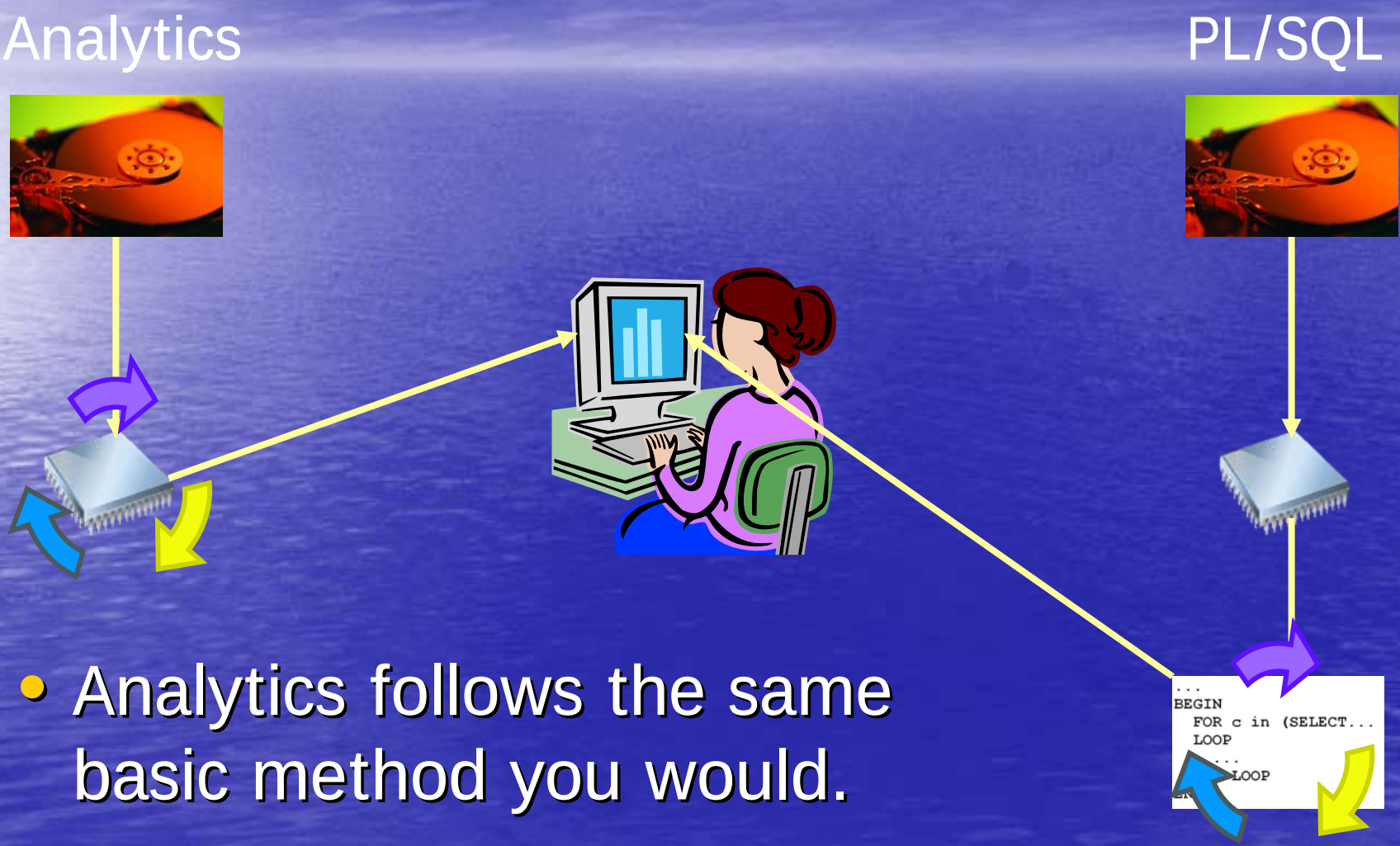


Review of Example

```
SELECT
  empno, sal, deptno
  ,SUM(sal) over
    (PARTITION BY deptno) AS
   dept_sal
FROM
  emp
```

- No Group By
- Simple query
- Analytic function fills one column per row

Review of Example



- Analytics follows the same basic method you would.

Parts of an Analytic Function

<function> OVER (<subset>)

Parts - <function>

- Aggregate and other analytic specific functions to be used on a subset of data.
- AVG, CORR, COVAR_POP, COVAR_SAMP, COUNT, CUME_DIST, DENSE_RANK, FIRST, FIRST_VALUE, LAG, LAST, LAST_VALUE, LEAD, MAX, MIN, NTILE, PERCENT_RANK, PERCENTILE_CONT, PERCENTILE_DISC, RANK, RATIO_TO_REPORT, REGR_(Linear Regression), ROW_NUMBER, STDDEV, STDDEV_POP, STDDEV_SAMP, SUM, VAR_POP, VAR_SAMP, VARIANCE

Parts - <subset>

- To get started, probably the best way to think about the subset is by thinking of SQL.
- Ask yourself “What SQL is required to get the value I need for this column/row from the primary result set?”
- Then figure out how to apply the correct subset syntax.

Parts - <subset>

PARTITION BY <expr>

ORDER BY <expr>

<window>

Windows into the Data

Part 4 – Digression 1

Windows into the Data

- This presentation is about getting started.
- We've talked about the efficiency of analytic functions.
- And we've discussed how it can be thought of as a sort of embedded SQL.
- To get to the real value requires visualizing the data in windows.

Basic Data Set

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

PARTITION BY deptno

Basic Data Set Partitioned

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

ORDER BY mgr

Basic Data Set + Ordered

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

Sum(sal)

Basic Data Set + Function

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

ROWS BETWEEN
1 preceding
AND
1 following

8750
8750
8750
10875
9400
9400
9400
9400
9400

Basic Data Set + Window

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

3750
8750
7450
6000
7100
7075
4875
3775
2850
4350
4050
3700
5050
4100

The Query with Window

```
SELECT
 empno, ename, job, mgr, deptno
 ,sal
 ,SUM(sal) OVER
  (PARTITION BY deptno
   ORDER      BY mgr
   ROWS BETWEEN 1 preceding
            AND 1 following) AS
 dept_sal
FROM
 emp;
```

Examples

Part 5

Example 1 - Problem Statement

- There is a product registration table which is sparsely populated.
- We need to find
  - all of the entries
  - from the most recent X months (4)
  - prior to the current month (12/2007)
  - given the user specified Type (1)

Example 1 – Need to Know

DENSE_RANK computes the rank of a row in an ordered group of rows and returns the rank as a NUMBER. The ranks are consecutive integers beginning with 1. The largest rank value is the number of unique values returned by the query. Rank values are not skipped in the event of ties. Rows with equal values for the ranking criteria receive the same rank.

Example 1 - Base Data

PID	REG_DT	TYPE
93	4/17/2007	2
88	4/30/2007	1
90	5/1/2007	1
92	6/10/2007	1
95	6/22/2007	3
94	8/9/2007	1
97	9/20/2007	3
99	11/1/2007	2
96	11/8/2007	1
98	11/15/2007	1
100	12/1/2007	1

Example 1 - Step 1 SQL

```
SELECT
  a.*
FROM
  prod_p1 a
WHERE
  a.reg_dt < trunc(SYSDATE, 'MM')
  AND a.TYPE = 1;
```

Example 1 - Step 2 SQL

```
SELECT
  a.*
  ,trunc(a.reg_dt,'MM') AS mm_dt
  ,dense_rank() over
    (ORDER BY trunc(a.reg_dt, 'MM')
     DESC) AS rank
FROM
  prod_p1 a
WHERE
  a.reg_dt < trunc(SYSDATE,'MM')
  AND a.TYPE = 1;
```

Example 1 - Step 2 Data

PID	REG_DT	TYPE	MM_DT	RANK
98	11/15/2007	1	11/1/2007	1
96	11/8/2007	1	11/1/2007	1
94	8/9/2007	1	8/1/2007	2
92	6/10/2007	1	6/1/2007	3
90	5/1/2007	1	5/1/2007	4
88	4/30/2007	1	4/1/2007	5

Example 1 - Final SQL

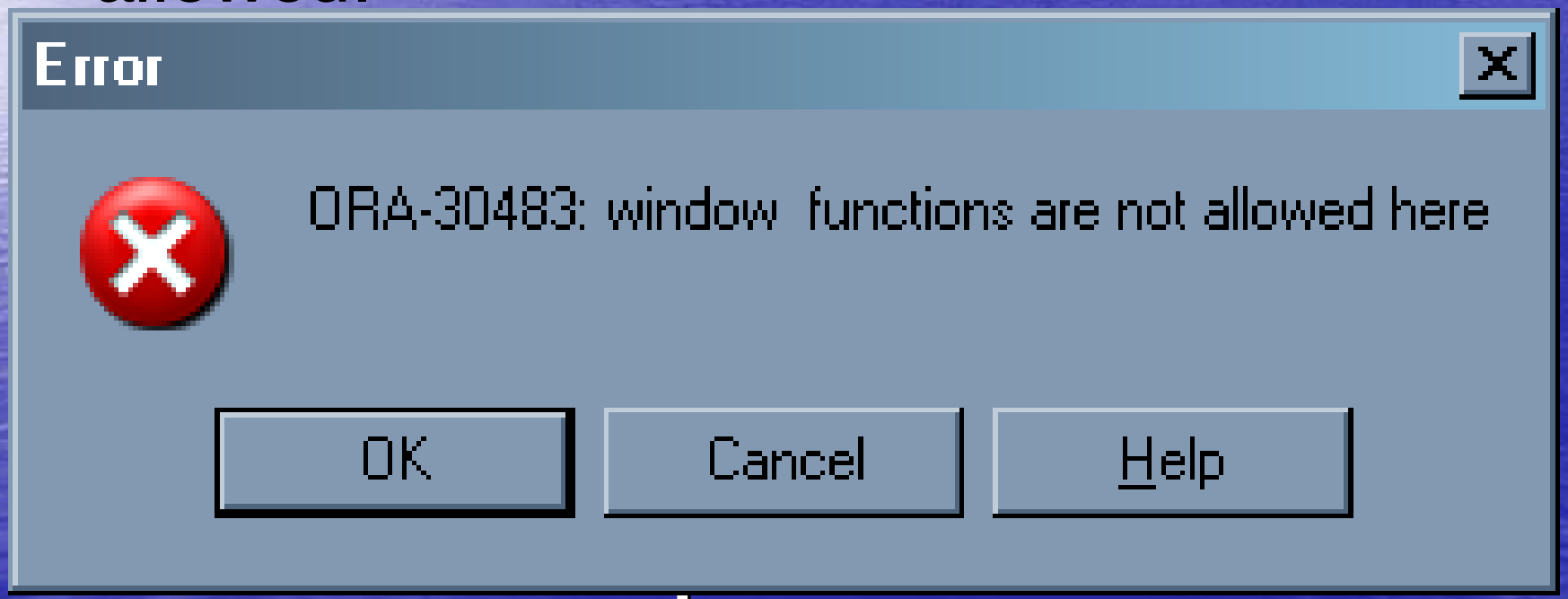
```
SELECT
  b.*
FROM
  (SELECT
    a.*
    ,dense_rank() over
     (ORDER BY trunc(a.reg_dt, 'MM')
      DESC) AS rank
   FROM   prod_p1 a
   WHERE  a.reg_dt < trunc(SYSDATE, 'MM')
          AND a.TYPE = 1) b
WHERE       b.rank <= 4;
```

Example 2 - Problem Statement

- There is a table that records sales for each employee and includes the $ amount of each sale.
- There is a need to see the employee, count of sales, sum of $, $ per sale and the $ per sale for the department.

Example 2 – Need to Know

Aggregating analytic functions is not allowed.



Example 2 - Step 1 SQL

```
SELECT
  s.empno, s.amt, e.deptno
  ,COUNT(*) over
   (PARTITION BY e.deptno) AS d_sls
  ,SUM(s.amt) over
   (PARTITION BY e.deptno) AS d_amt
FROM
  sales_p2 s, emp e
WHERE
  s.empno = e.empno
```

Example 2 - Step 1 Data

EMPNO	AMT	DEPTNO	D_SALES	D_AMT
7369	13,227	20	87	1,262,099
7369	26,445	20	87	1,262,099
7369	6,963	20	87	1,262,099
...	...	...	...	...
7499	32,333	30	111	1,799,562
7499	21,826	30	111	1,799,562
7499	28,219	30	111	1,799,562
...	...	...	...	...
7566	23,598	20	87	1,262,099
7566	12,269	20	87	1,262,099
...	...	...	...	...
7782	6,660	10	54	917,949
7782	6,562	10	54	917,949
7782	1,482	10	54	917,949

This data is easy.

Sum amt and count by empno.

Hmmm…

This data is already summed so if I sum it again that would be wrong!

Example 2 - Final SQL

```
SELECT
  x.empno
  ,COUNT(*) AS e_sls
  ,sum(x.amt) AS e_amt
  ,MAX(x.d_sls) AS d_sls
  ,MAX(x.d_amt) AS d_amnt
FROM
  (SELECT  s.empno, s.amt
           ,COUNT(*) over (PARTITION BY e.deptno) AS d_sls
           ,SUM(s.amt) over (PARTITION BY e.deptno) AS d_amt
   FROM    sales_p2 s, emp e
   WHERE   s.empno = e.empno) x
GROUP BY
  x.empno
```

Example 3 - Problem Statement

- In a large production line there are 10's of thousands of devices each throwing a variety of events regularly.
- One of these events indicates potential failure; however it has been determined that there are often false positives (ie indications of a failure when there was really no failure).

Example 3 - Problem Statement

- It has been theorized that there is a pattern of events which indicate real failure.
- In order to determine this, the event data needs to be analyzed resulting in:

Example 3 - Problem Statement

- For each instance where a device has issued the first error (status = 0) then
  - How many times did any event follow?
  - If an event followed how many times was it an error.
- Repeat this data up to 3 consecutive errors.
  - ie was there a 4^{th} event and was it an error?

Example 3 – Need to Know

LEAD is an analytic function... [which]... provides access to a row at a given physical offset beyond that position.

In other words you can peek ahead in the subset.

Example 3 – Table Structure

NAME	TYPE
dev_id	number(10)
evt_dt	date
status	number(2)

Example 3 - Step 1 SQL

```
SELECT
  a.*
  ,lead(a.status,1) over
   (PARTITION BY a.dev_id
    ORDER BY a.evt_dt ASC) AS next1_stat
  ,lead(a.status,2) over
   (PARTITION BY a.dev_id
    ORDER BY a.evt_dt ASC) AS next2_stat
  ,lead(a.status,3) over
   (PARTITION BY a.dev_id
    ORDER BY a.evt_dt ASC) AS next3_stat
  ,lead(a.status,4) over
   (PARTITION BY a.dev_id
    ORDER BY a.evt_dt ASC) AS next4_stat
```

Example 3 - Step 1 SQL

```
 ,MIN(CASE WHEN a.status = 0
           THEN a.evt_dt
           ELSE to_date('1-jan-2099')
      END) over
  (PARTITION BY a.dev_id) AS min_err_dt
FROM
  dev_evt a;
```

Example 3 - Step 2 SQL

```
SELECT      x.*
FROM        (blah) x
WHERE       x.evt_dt = min_err_dt
```

D_ID	EVT_DT	ST	N1	N2	N3	N4	MIN_ERR_DT
1	7/25/01 15:16:48	0	0	0	0	0	7/25/01 15:16:48
2	8/7/02 19:09:17	0	99	99	99	99	8/7/02 19:09:17
3	5/30/01 14:22:41	0	90	22	22	90	5/30/01 14:22:41
4	6/6/01 17:38:39	0	22	22	0		6/6/01 17:38:39
7	11/22/00 14:38:17	0	44				11/22/00 14:38:17

The hard work is done
You can do the rest!

Example 3 – The real world!

- All the basics are the same but the names were changed to protect the innocent.
- 290 million rows of data.
- Someone tried this in PL/SQL.
- It ran for 24 hrs and never finished.
- I did it with analytics and it took less then 2.
- I’m pretty sure it took me less time to write the query then it took them to write the PL/SQL.

So…

- You won't truly "get" analytic functions till you use them so get started.
- I'll help, and if they are really interesting I may use them in a more advanced presentation.

Thanks!

Travis R. Rogers

SrVP of Database Development at
CDG Management LLC

travis@jerseyrogers.com