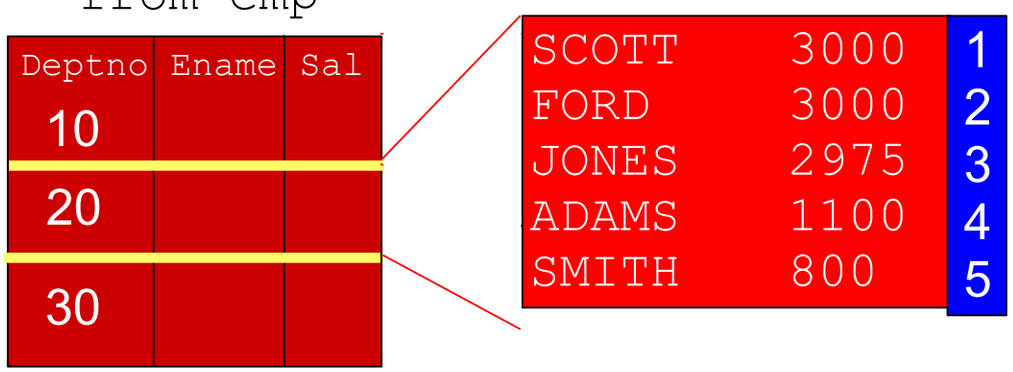
Analytics

Analytics

Ordered Array Semantics in SQL queries

```
Select deptno,ename,sal
  Row_number() over (partition by deptno
                     Order by sal desc )
   from emp
```



Why Analytics

- A running total (demo001.sql)
- Percentages within a group (demo002.sql)
- Top-N queries (demo003.sql)
- Moving Averages (demo004.sql)
- Ranking Queries (demo005.sql)
- Medians (med.sql)
- And the list is infinitely long
  - *"Analytics are the coolest thing to happen to SQL since the keyword Select"*
  - *Lets look at a complex example*

Why Analytics

```
scott%ORA11GR2> Select deptno, ename, sal,
  2         sum(sal) over (partition by deptno order by sal) running_total1,
  3         sum(sal) over (partition by deptno order by sal, rowid) running_total2
  4    from emp order by deptno, sal;


    DEPTNO ENAME             SAL RUNNING_TOTAL1 RUNNING_TOTAL2
---------- ---------- ---------- -------------- --------------
        10 MILLER           1300           1300           1300
           CLARK            2450           3750           3750
           KING             5000           8750           8750

        20 SMITH             800            800            800
           ADAMS            1100           1900           1900
           JONES            2975           4875           4875
           SCOTT            3000          10875           7875
           FORD             3000          10875          10875

        30 JAMES             950            950            950
           WARD             1250           3450           2200
….
```

Why Analytics

```
scott%ORA11GR2> select deptno, ename, sal,
  2         to_char( round(
  3               ratio_to_report(sal) over (partition by deptno)
  4                   *100, 2 ), '990.00' )||'%' rtr
  5    from emp
  6   order by deptno, sal
  7  /
```

DEPTNO	ENAME	SAL	RTR
10	MILLER	1300	14.86%
	CLARK	2450	28.00%
	KING	5000	57.14%
20	SMITH	800	7.36%
	ADAMS	1100	10.11%
	JONES	2975	27.36%
	SCOTT	3000	27.59%
	FORD	3000	27.59%

…

Why Analytics

```
scott%ORA11GR2> select emp.deptno, ename, sal,
  2         to_char( round(
  3                         sal/tot_dept
  4                       *100, 2 ), '990.00' )||'%' rtr
  5    from emp, (select deptno, sum(sal) tot_dept from emp group by deptno) d
  6   where emp.deptno = d.deptno
  7   order by deptno, sal
  8  /


    DEPTNO ENAME             SAL RTR
---------- ---------- ---------- --------
        10 MILLER           1300   14.86%
           CLARK            2450   28.00%
           KING             5000   57.14%

        20 SMITH             800    7.36%
           ADAMS            1100   10.11%
           JONES            2975   27.36%
           SCOTT            3000   27.59%
           FORD             3000   27.59%
```

Why Analytics

```
scott%ORA11GR2> select deptno, ename, sal, rn
  2    from (
  3  Select deptno, ename, sal,
  4         row_number() over (partition by deptno order by sal desc) rn
  5    from emp
  6         )
  7   where rn <= 3
  8  /

    DEPTNO ENAME             SAL         RN
---------- ---------- ---------- ----------
        10 KING             5000          1
           CLARK            2450          2
           MILLER           1300          3

        20 SCOTT            3000          1
           FORD             3000          2
           JONES            2975          3

        30 BLAKE            2850          1
           ALLEN            1600          2
           TURNER           1500          3
9 rows selected.
```

Why Analytics

```
scott%ORA11GR2> select deptno, ename, sal, rank
  2    from (
  3  Select deptno, ename, sal,
  4         rank() over (partition by deptno order by sal desc) rank
  5    from emp
  6         )
  7   where rank <= 3
  8  /

    DEPTNO ENAME              SAL       RANK
---------- ---------- ---------- ----------
        10 KING             5000          1
           CLARK            2450          2
           MILLER           1300          3

        20 SCOTT            3000          1
           FORD             3000          1
           JONES            2975          3

        30 BLAKE            2850          1
           ALLEN            1600          2
           TURNER           1500          3
9 rows selected.
```

Why Analytics

```
scott%ORA11GR2> select deptno, ename, sal, dr
  2    from (
  3  Select deptno, ename, sal,
  4         dense_rank() over (partition by deptno order by sal desc) dr
  5    from emp
  6         )
  7   where dr <= 3
  8  /

    DEPTNO ENAME              SAL         DR
---------- ---------- ---------- ----------
        10 KING             5000          1
           CLARK            2450          2
           MILLER           1300          3

        20 SCOTT            3000          1
           FORD             3000          1
           JONES            2975          2
           ADAMS            1100          3

        30 BLAKE            2850          1
           ALLEN            1600          2
           TURNER           1500          3
```

Why Analytics

```
scott%ORA11GR2> Select ename, sal,
  2         round(
  3               avg(sal) over (order by sal
  4                              rows 5 preceding)
  5                   ) avg_sal,
  6         rtrim(
  7         lag(sal) over (order by sal) || ',' ||
  8         lag(sal,2) over (order by sal) || ',' ||
  9         lag(sal,3) over (order by sal) || ',' ||
 10         lag(sal,4) over (order by sal) || ',' ||
 11         lag(sal,5) over (order by sal),',') last_5
 12    from emp
 13   order by sal;

ENAME              SAL    AVG_SAL LAST_5
---------- ---------- ---------- ------------------------------
SMITH              800        800
JAMES              950        875 800
ADAMS             1100        950 950,800
WARD              1250       1025 1100,950,800
MARTIN            1250       1070 1250,1100,950,800
MILLER            1300       1108 1250,1250,1100,950,800    …
```

Why Analytics

```
ops$tkyte%ORA11GR2> select object_type, round( avg( diff ), 2 ) avg_diff
  2    from (
  3  select object_type,
  4         created - lag(created) over (partition by object_type order by created) diff
  5    from all_objects
  6         )
  7   group by object_type
  8   order by 2 desc
  9  /


OBJECT_TYPE           AVG_DIFF
------------------- ----------
RULE
LOB
EDITION                390.21
MATERIALIZED VIEW       90.34
CLUSTER                 42.89
DIRECTORY                13.9
EVALUATION CONTEXT       5.71
RULE SET                 3.63
TABLE SUBPARTITION       3.43
PROCEDURE                 2.7
TABLE PARTITION          1.98
SEQUENCE                 1.84
TYPE BODY                1.84
…
```

Analytics To Tune

```
insert into t ( c1, c2, .... )
 select c1, c2, ....
   from t1, t2, t3, t4, ....
  where ....;

 loop
    delete from t
     where (c1,c2) in ( select c1, min(c2)
                          from t
                         group by c1
                        having count(1) > 1 );
    exit when sql%rowcount = 0;
 end loop;
```

remove rows with the older C2 having the same C1. Additionally -- IF all of the rows for a given C1 have the same C2, these rows are to be removed as well.

Analytics

```
insert into t ( c1, c2, .... )
 select c1, c2, ....
   from t1, t2, t3, t4, ....
  where ....;
 loop
    delete from t
     where (c1,c2) in ( select c1, min(c2)
                          from t
                         group by c1
                        having count(1) > 1 );
    exit when sql%rowcount = 0;
 end loop;
```

The query took the same amount of time, the delete went from 3/6/9 hours to 0

```
 insert into t ( c1, c2, .... )
  select c1, c2, ......
    from
( select c1, c2, .... ,
         max(c2) OVER ( partition by c1 ) max_c2
         --count(c2) OVER ( partition by c1,c2 ) cnt
             from t1, t2, t3, t4, ....
            where .... )
   where c2 = max_c2
     --and cnt = 1;
```

Now for the boring stuff -- syntax

- Straightforward looking perhaps
- But looks are deceiving

```
  FUNCTION_NAME( <arg>,<arg>,… )
  OVER (
  <partition clause>
  <order by clause>
  <windowing clause>
  )
```

- Lets look at each

Functions

- 40-50 of them
- Not including all of the linear regression functions
- Favorites
  - LAG/LEAD (look back/forwards)
  - FIRST/LAST or MIN/MAX
  - ROW_NUMBER, RANK, DENSE_RANK, RATIO_TO_REPORT
  - AVG, SUM
  - NTILE
  - PERCENTILE_CONT/DISC

Partition Clause

- Breaks entire result set up into "mini result sets"
- Functions work only on the rows within each partition
- If you omit a partition clause -- entire result set is processed by the analytic function
- Each analytic function may have its own partition clause
  - Caveat Emptor!!
- Demo008.sql

Why Analytics

```
scott%ORA11GR2> select deptno, job, ename, sal,
  2         sum(sal) over () total_sal,
  3         sum(sal) over ( partition by deptno) sal_by_dept,
  4         sum(sal) over ( partition by deptno, job) sal_by_dept_job,
  5         sum(sal) over ( partition by job) sal_by_job
  6    from emp
  7   order by deptno, job
  8
scott%ORA11GR2> /
```

DEPTNO	JOB	ENAME	SAL	TOTAL_SAL	SAL_BY_DEPT	SAL_BY_DEPT_JOB	SAL_BY_JOB
10	CLERK	MILLER	1300	29025	8750	1300	4150
	MANAGER	CLARK	2450	29025	8750	2450	8275
	PRESIDENT	KING	5000	29025	8750	5000	5000
20	ANALYST	SCOTT	3000	29025	10875	6000	6000
	ANALYST	FORD	3000	29025	10875	6000	6000
	CLERK	ADAMS	1100	29025	10875	1900	4150
	CLERK	SMITH	800	29025	10875	1900	4150
	MANAGER	JONES	2975	29025	10875	2975	8275

Why Analytics

```
scott%ORA11GR2> select deptno, job, ename, sal,
  2         sum(sal) over () total_sal,
  3         sum(sal) over ( partition by deptno) sal_by_dept,
  4         sum(sal) over ( partition by deptno, job) sal_by_dept_job,
  5         sum(sal) over ( partition by job) sal_by_job
  6    from emp
  7   order by deptno, job
  8
scott%ORA11GR2> /


Execution Plan
----------------------------------------------------------
Plan hash value: 4086863039

-------------------------------------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |    14 |   294 |     4  (25)| 00:00:01 |
|   1 |  WINDOW SORT        |      |    14 |   294 |     4  (25)| 00:00:01 |
|   2 |   WINDOW SORT       |      |    14 |   294 |     4  (25)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| EMP  |    14 |   294 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Order by Clause

- Some analytics require it -- e.g. row_number()
- Some analytics do not -- e.g. sum()
  - Causes change in behavior!
  - Creates an implicit window (more on that in a moment) from the current row back up to the first row in the partition
  - Beware of DUPLICATES - the are considered "the same". This can be a big "gotcha" depending on need.
  - ORDER BY expression <asc|desc> <*NULLS FIRST*| *NULLS LAST*>
  - Demo009.sql

Why Analytics

```
scott%ORA11GR2> select empno, row_number() over ()
  2    from emp;
select empno, row_number() over ()
              *
ERROR at line 1:
ORA-30485: missing ORDER BY expression in the window specification
```

Why Analytics

```
scott%ORA11GR2> select empno,
  2         sum(sal) over () tot,
  3             sum(sal) over (order by sal) rt1,
  4             sum(sal) over (order by sal, rowid) rt2,
  5             sum(sal) over (order by sal)
  6             - sum(sal) over (order by sal, rowid) diff
  7    from emp
  8   order by sal
  9  /

     EMPNO        TOT        RT1        RT2       DIFF
---------- ---------- ---------- ---------- ----------
      7369      29025        800        800          0
      7900      29025       1750       1750          0
      7876      29025       2850       2850          0
      7521      29025       5350       4100       1250
      7654      29025       5350       5350          0
      7934      29025       6650       6650          0
      7844      29025       8150       8150          0
      7499      29025       9750       9750          0
      7782      29025      12200      12200          0
      7698      29025      15050      15050          0
      7566      29025      18025      18025          0
      7788      29025      24025      21025       3000
      7902      29025      24025      24025          0
      7839      29025      29025      29025          0
14 rows selected.
```

Window Clause

- This looks "tricky". 'range between unbounded preceding and current row'
  - That’s the default window when you use ORDER BY with sum for example!
- Lets us create a window in the current partition
  - Works on a single partition, the window will never span two partitions (won't span DEPTNO's for example)
  - Can be SLIDING
  - Or ANCHORED
  - How about a diagram for that….

```
scott@ORA920> select empno, hiredate, sal,
  2         sum(sal) over
  3         ( order by hiredate
  4           rows between 1 preceding and 1 following) sumsal,
  5         count(sal) over
  6         ( order by hiredate
  7           rows between 1 preceding and 1 following) cntsal
  8    from emp
  9   order by hiredate
 10  /

     EMPNO HIREDATE         SAL      SUMSAL      CNTSAL
---------- --------- ---------- ---------- ----------
      7369 17-DEC-80        800       2400          2
      7499 20-FEB-81       1600       3650          3
      7521 22-FEB-81       1250       5825          3
      7566 02-APR-81       2975       7075          3
      7698 01-MAY-81       2850       8275          3
      7782 09-JUN-81       2450       6800          3
      7844 08-SEP-81       1500       5200          3
      7654 28-SEP-81       1250       7750          3
      7839 17-NOV-81       5000       7200          3
      7900 03-DEC-81        950       8950          3
      7902 03-DEC-81       3000       5250          3
      7934 23-JAN-82       1300       7300          3
      7788 09-DEC-82       3000       5400          3
      7876 12-JAN-83       1100       4100          2
```

Sliding
demo010

```
scott@ORA920> select ename, sal, hiredate, hiredate-100 window_top,
  2         first_value( ename )
  3             over ( order by hiredate asc
  4                    range 100 preceding ) ename_prec,
  5         first_value( hiredate )
  6             over ( order by hiredate asc
  7                    range 100 preceding ) hiredate_prec
  8    from emp
  9   order by hiredate asc
 10  /

ENAME              SAL HIREDATE  WINDOW_TO ENAME_PREC HIREDATE_
---------- ---------- --------- --------- ---------- ---------
SMITH              800 17-DEC-80 08-SEP-80 SMITH      17-DEC-80
ALLEN             1600 20-FEB-81 12-NOV-80 SMITH      17-DEC-80
WARD              1250 22-FEB-81 14-NOV-80 SMITH      17-DEC-80
JONES             2975 02-APR-81 23-DEC-80 ALLEN      20-FEB-81
BLAKE             2850 01-MAY-81 21-JAN-81 ALLEN      20-FEB-81
CLARK             2450 09-JUN-81 01-MAR-81 JONES      02-APR-81
TURNER            1500 08-SEP-81 31-MAY-81 CLARK      09-JUN-81
MARTIN            1250 28-SEP-81 20-JUN-81 TURNER     08-SEP-81
KING              5000 17-NOV-81 09-AUG-81 TURNER     08-SEP-81
FORD              3000 03-DEC-81 25-AUG-81 TURNER     08-SEP-81
JAMES              950 03-DEC-81 25-AUG-81 TURNER     08-SEP-81
MILLER            1300 23-JAN-82 15-OCT-81 KING       17-NOV-81
SCOTT             3000 09-DEC-82 31-AUG-82 SCOTT      09-DEC-82
ADAMS             1100 12-JAN-83 04-OCT-82 SCOTT      09-DEC-82
```

Sliding
demo011

```
scott@ORA920> select ename, sal,
  2         sum(sal) over
  3         (order by sal
  4          range between unbounded preceding
  5                    and current row) sum_back,
  6         sum(sal) over
  7         (order by sal
  8          range between current row
  9                    and unbounded following
 10          ) sum_forward
 11    from emp order by sal
 12  /
ENAME              SAL   SUM_BACK SUM_FORWARD
---------- ---------- ---------- -----------
SMITH             800        800       29025
JAMES             950       1750       28225
ADAMS            1100       2850       27275
WARD             1250       5350       26175
MARTIN           1250       5350       26175
MILLER           1300       6650       23675
TURNER           1500       8150       22375
ALLEN            1600       9750       20875
CLARK            2450      12200       19275
BLAKE            2850      15050       16825
JONES            2975      18025       13975
SCOTT            3000      24025       11000
FORD             3000      24025       11000
KING             5000      29025        5000
```

Anchor
demo012

Caveats

- In 8i -- PL/SQL did not recognize analytics (or order by in subquery, or …)
  - Use native dynamic SQL or
  - A view (but, see caveat about views first!)
- Analytics cannot be used in a where clause
  - They are evaluated second to last, right before "order by"
  - Inline views are crucial for this
- Nulls sort "higher" than non-nulls
  - So you must be aware of NULLS LAST!

Caveats

- Performance
  - Each partition can cause a temporary allocation or sort
  - Each order by could as well

```
Select ename, deptno,
       sum(sal) over (order by ename, deptno ),
       sum(sal) over (order by deptno, ename ),
       sum(sal) over (partition by job),
       sum(sal) over (partition by deptno)
  from emp;
```

  - Think of what that query must do to get the answer!
  - But also think of how “regular SQL” would perform

Caveats

- Views
  - Remember, "select * from view where x = 5" does not mean the predicate will be pushed!
  - Because doing so may change the answer!
  - demo013

Why Analytics

```
ops$tkyte%ORA11GR2> create index job_idx on emp(job);

Index created.

ops$tkyte%ORA11GR2>
ops$tkyte%ORA11GR2> create or replace view v
  2  as
  3  select ename, sal, job,
  4         sum(sal) over (partition by job) sal_by_job,
  5             sum(sal) over (partition by deptno) sal_by_deptno
  6    from emp
  7  /

View created.
```

Why Analytics

```
ops$tkyte%ORA11GR2> select *
  2    from v
  3   where job = 'CLERK'
  4   order by ename
  5  /

ENAME           SAL JOB       SAL_BY_JOB SAL_BY_DEPTNO
---------- ------- --------- ---------- -------------
ADAMS          1100 CLERK            4150         10875
JAMES           950 CLERK            4150          9400
MILLER         1300 CLERK            4150          8750
SMITH           800 CLERK            4150         10875
```

Why Analytics

```
ops$tkyte%ORA11GR2> select * from table(dbms_xplan.display_cursor());


PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
select *   from v  where job = 'CLERK'  order by ename

Plan hash value: 2851695225

-------------------------------------------------
| Id  | Operation             | Name | Cost (%CPU)|
-------------------------------------------------
|   0 | SELECT STATEMENT      |      |            |
|   1 |  SORT ORDER BY        |      |     0   (0)|
|*  2 |   VIEW                | V    |            |
|   3 |    WINDOW SORT        |      |     0   (0)|
|   4 |     WINDOW SORT       |      |     0   (0)|
|   5 |      TABLE ACCESS FULL| EMP  |            |
-------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter("JOB"='CLERK')
```

Why Analytics

```
ops$tkyte%ORA11GR2> select ename, sal, job,
  2         sum(sal) over (partition by job) sal_by_job,
  3             sum(sal) over (partition by deptno) sal_by_deptno
  4    from emp
  5   where job = 'CLERK' order by ename;


ops$tkyte%ORA11GR2> select * from table(dbms_xplan.display_cursor());
…
-----------------------------------------------------------------
| Id  | Operation                     | Name    | Cost (%CPU)|
-----------------------------------------------------------------
|   0 | SELECT STATEMENT              |         |            |
|   1 |  SORT ORDER BY                |         |     0   (0)|
|   2 |   WINDOW SORT                 |         |     0   (0)|
|   3 |    WINDOW SORT                |         |     0   (0)|
|   4 |     TABLE ACCESS BY INDEX ROWID| EMP     |            |
|*  5 |      INDEX RANGE SCAN         | JOB_IDX |            |
-----------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   5 - access("JOB"='CLERK')
```

Why Analytics

```
ops$tkyte%ORA11GR2> select * from v where job = 'CLERK' order by ename;

ENAME          SAL JOB       SAL_BY_JOB SAL_BY_DEPTNO
---------- ------- --------- ---------- -------------
ADAMS         1100 CLERK           4150         10875
JAMES          950 CLERK           4150          9400
MILLER        1300 CLERK           4150          8750
SMITH          800 CLERK           4150         10875

ops$tkyte%ORA11GR2> select ename, sal, job,
  2         sum(sal) over (partition by job) sal_by_job,
  3             sum(sal) over (partition by deptno) sal_by_deptno
  4    from emp
  5   where job = 'CLERK' order by ename;


ENAME          SAL JOB       SAL_BY_JOB SAL_BY_DEPTNO
---------- ------- --------- ---------- -------------
ADAMS         1100 CLERK           4150          1900
JAMES          950 CLERK           4150           950
MILLER        1300 CLERK           4150          1300
SMITH          800 CLERK           4150          1900
```

Why Analytics

```
ops$tkyte%ORA11GR2> select ename, sal, sal_by_job
  2    from v
  3   where job = 'CLERK'
  4   order by ename
  5  /

ENAME          SAL SAL_BY_JOB
---------- ------- ----------
ADAMS         1100       4150
JAMES          950       4150
MILLER        1300       4150
SMITH          800       4150
```

Why Analytics

```
ops$tkyte%ORA11GR2> select * from table(dbms_xplan.display_cursor());

…

-----------------------------------------------------------------------
| Id  | Operation                      | Name    | Cost (%CPU)|
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT               |         |            |
|   1 |  SORT ORDER BY                 |         |     0   (0)|
|   2 |   VIEW                         | V       |            |
|   3 |    WINDOW BUFFER               |         |     0   (0)|
|   4 |     TABLE ACCESS BY INDEX ROWID| EMP     |            |
|*  5 |      INDEX RANGE SCAN          | JOB_IDX |            |
-----------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   5 - access("JOB"='CLERK')
```

Other Examples

- Analytic1.html – detecting overlaps
- Analytic2.html – complex running totals
- Analytic3.html – look at performance
- Analytic4.html – ranking/numbering
- Analytic5.html – filtering
- Analytic6.html – getting related columns with min/max/whatever
- Search for this on asktom for more examples
  - Analytics rock and roll

Questions and Answers

ORACLE