HrOUG Autumn 2022

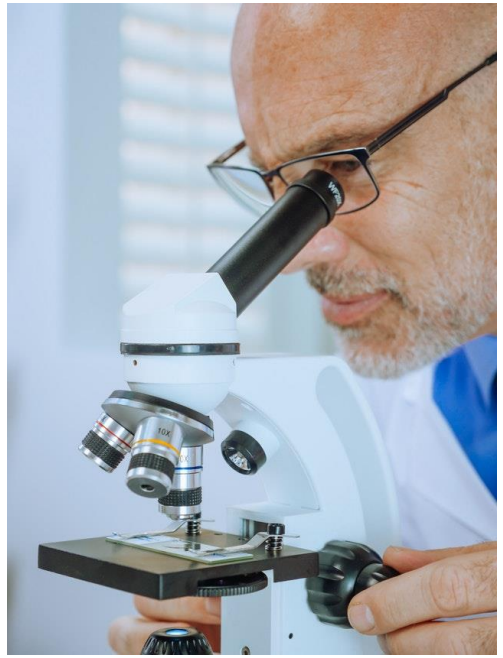
Back to the future
With Analytic Functions

Erik van Roon

EvROCS
COMPLETING THE PUZZLE

Who Am I?

Erik van Roon



1995

ORACLE DATABASE
>=Oracle5

2009

EvROCS
COMPLETING THE PUZZLE

SYM42
Member of Symposium 42
https://sym42.org/

MASH
Core team
MASH Program

Oracle ACE Pro

: erik.van.roon@evrocs.nl
: www.evrocs.nl
: @evrocs_nl

Mentor and Speaker Hub

Our goal is to *connect* speakers with mentors to assist in *preparing* technical sessions and *improving* presentation skills

Interested? Read more and get in touch

https://mashprogram.wordpress.com

Oracle ACE Program

500+ technical experts helping peers globally

The **Oracle ACE Program** recognizes and rewards community members for their technical and community contributions to the Oracle community

3 membership tiers

Oracle ACE Director | Oracle ACE Pro | Oracle ACE Associate

For more details on Oracle ACE Program:
ace.oracle.com

Oracle ACE

Nominate

yourself or someone you know:

ace.oracle.com/nominate

Connect: **aceprogram_ww@oracle.com** Facebook.com/OracleACEs @oracleace

Let's Start

Analytic Functions

- Calculated based on group of rows
- Similar to aggregate, but return all rows
- No need to use "GROUP BY"
- Most aggregate functions have analytic versions

Analytic Functions Concept

ID	Name	Sal	Avg
206	Gietz	8300	**6411**
198	Oconnell	2600	**6411**
204	Baer	10000	**6411**
101	Kochhar	17000	**6411**
127	Landry	2400	**6411**
131	Marlow	2500	**6411**
112	Urman	7800	**6411**
116	Baida	2900	**6411**
107	Lorentz	4200	**6411**

- Used in select clause
- Works for each row
- Input is the result set

Analytic Functions Concept

ID	Name	Sal	Avg
206	Gietz	8300	**9460**
204	Baer	10000	**9460**
101	Kochhar	17000	**9460**
112	Urman	7800	**9460**
107	Lorentz	4200	**9460**

- Used in select clause
- Works for each row
- Input is the result set
  So, after: where/group by/etc.
  But before: order by

Analytic Functions Concept

ID	Name	Sal	Avg
206	Gietz	8300	6411
198	Oconnell	2600	6411
204	Baer	10000	6411
101	Kochhar	17000	6411
127	Landry	2400	6411
131	Marlow	2500	6411
112	Urman	7800	6411
116	Baida	2900	6411
107	Lorentz	4200	6411

Use data from other rows without:

- Group by
- Self join

Analytic Functions Concept

DEPT	Name	Sal	Avg
A	Gietz	8300	6411
A	Oconnell	2600	6411
Q	Baer	10000	6411
Q	Kochhar	17000	6411
Q	Landry	2400	6411
M	Marlow	2500	6411
Z	Urman	7800	6411
Z	**Baida**	**2900**	**6411**
Z	Lorentz	4200	6411

analytic functions can calculate based on:

the entire resultset …

Analytic Functions Concept

DEPT	Name	Sal	Avg
A	Gietz	8300	5450
A	Oconnell	2600	5450
Q	Baer	10000	9800
Q	Kochhar	17000	9800
Q	Landry	2400	9800
M	Marlow	2500	2500
Z	Urman	7800	4967
Z	**Baida**	**2900**	**4967**
Z	Lorentz	4200	4967

… or just the rows by their group

Analytic Functions Concept

ID	Name	Sal	Avg
101	Kochhar	17000	6411
107	Lorentz	4200	6411
112	Urman	7800	6411
116	Baida	2900	6411
127	Landry	2400	6411
131	**Marlow**	**2500**	**6411**
198	Oconnell	2600	6411
204	Baer	10000	6411
206	Gietz	8300	6411

The rows can be relative to the current row:

All rows -
from first to last

Analytic Functions Concept

ID	Name	Sal	Avg
101	Kochhar	17000	17000
107	Lorentz	4200	10600
112	Urman	7800	9667
116	Baida	2900	7975
127	Landry	2400	6860
131	**Marlow**	**2500**	**6133**
198	Oconnell	2600	5629
204	Baer	10000	6175
206	Gietz	8300	6411

The rows can be relative to the current row:

From first row
To current row

Analytic Functions Concept

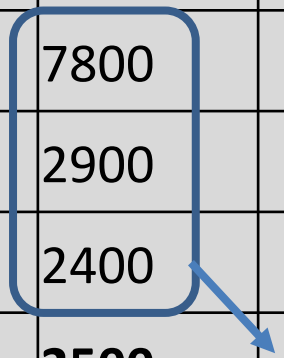
ID	Name	Sal	Avg
101	Kochhar	17000	9667
107	Lorentz	4200	7975
112	Urman	7800	4325
116	Baida	2900	3900
127	Landry	2400	2600
131	**Marlow**	**2500**	**4375**
198	Oconnell	2600	5850
204	Baer	10000	6967
206	Gietz	8300	9150

The rows can be relative to the current row:

From 1 row before
To 2 rows after

Analytic Functions Concept

ID	Name	Sal	Avg
101	Kochhar	17000	
107	Lorentz	4200	17000
112	Urman	7800	10600
116	Baida	2900	9667
127	Landry	2400	4967
131	**Marlow**	**2500**	**4367**
198	Oconnell	2600	2600
204	Baer	10000	2500
206	Gietz	8300	5033

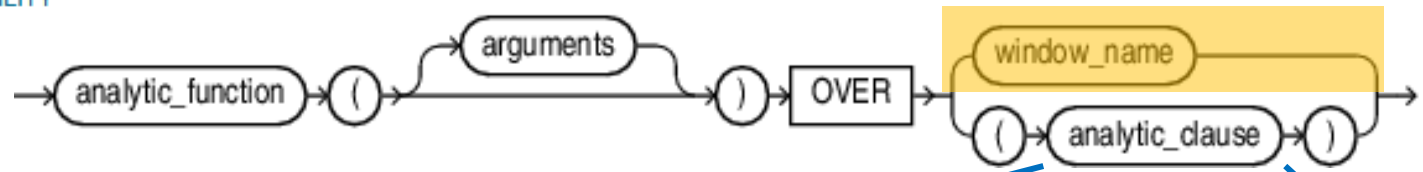


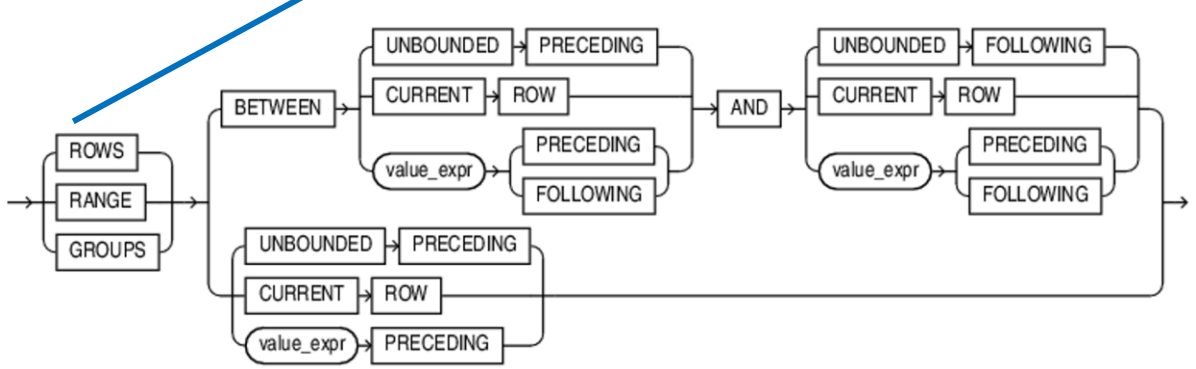
The rows can be relative to the current row:

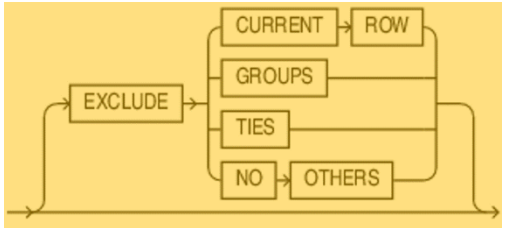
From 3 rows before
To 1 row before

AVG *
CLUSTER_DETAILS
CLUSTER_DISTANCE
CLUSTER_ID
CLUSTER_PROBABILITY
CLUSTER_SET
CORR *
COUNT *
COVAR_POP *
COVAR_SAMP *
CUME_DIST
DENSE_RANK
FEATURE_DETAILS
FEATURE_ID
FEATURE_SET
FEATURE_VALUE
FIRST
FIRST_VALUE *
LAG
LAST
LAST_VALUE *
LEAD
LISTAGG
MAX *
MIN *
NTH_VALUE *
NTILE
PERCENT_RANK
PERCENTILE_CONT
PERCENTILE_DISC
PREDICTION
PREDICTION_COST
PREDICTION_DETAILS
PREDICTION_PROBABILITY
PREDICTION_SET
RANK
RATIO_TO_REPORT

Full Syntax







= New in 21c

Syntax - order by

ID	Name	Sal	Row_number
		1000	**3**
		1500	**5**
		500	**1**
		3000	**9**
		1600	**6**
		1700	**7**
		1200	**4**
		900	**2**
		2000	**8**

```
select
    ...
    row_number()
    over(order by sal)
from
    ...
```

Syntax - partition by

ID	Name	Dept	Count
		10	**5**
		10	**5**
		10	**5**
		10	**5**
		10	**5**
		20	**3**
		20	**3**
		20	**3**
		30	**1**

```
select
    ...
    count(*)
    over(partition by dept)
from
    ...
```

Syntax - window (rows)

ID	Name	Sal	Avg
		500	**700**
		900	**800**
		1000	**1033**
		1200	**1233**
		1500	**1433**
		1600	**1600**
		1700	**1767**
		2000	**2233**
		3000	**2500**

```
select
    ...
    avg(sal)
    over(order by sal
         rows between
             1 preceding and
             1 following)
from
    ...
```

Syntax – window (range)

ID	Name	Sal	Avg
		1000	**1100**
		1200	**1250**
		1400	**1400**
		1400	**1400**
		1500	**1533**
		1500	**1533**
		1700	**1600**
		1700	**1600**
		2000	**2000**

```
select
    ...
    avg(sal)
    over(order by sal
         range between
           200 preceding and
           200 following)
from
    ...
```

Without "window" default is:
RANGE BETWEEN
UNBOUNDED PRECEDING AND CURRENT ROW

Syntax – window (groups) >= 21c

ID	Name	Sal	Avg
		1000	**1250**
		1200	**1333**
		1400	**1486**
		1400	**1486**
		1500	**1600**
		1500	**1600**
		1700	**1680**
		1700	**1680**
		2000	**1800**

```
select
    ...
    avg(sal)
    over(order by sal
         groups between
              1 preceding and
              2 following)
from
    ...
```

New in 21c - Window Clause (Named Windows)

Define a window

Give it a name

Reference it in analytic functions

```
select
    ...
    avg(sal) over my_window
from
    ...
where
    ...
window my_window as
    (partition by dept_id
     order by sal
          rows between
              1 preceding and
              2 following)
```

New in 21c - Exclude Clause

Exclude Result-Set rows

From the function !!
Not from the resultset!!

EXCLUDE CURRENT ROW

- The current row
  (if it's in the window)

EXCLUDE GROUP

- All rows in the window with the same value as the current row

EXCLUDE TIES

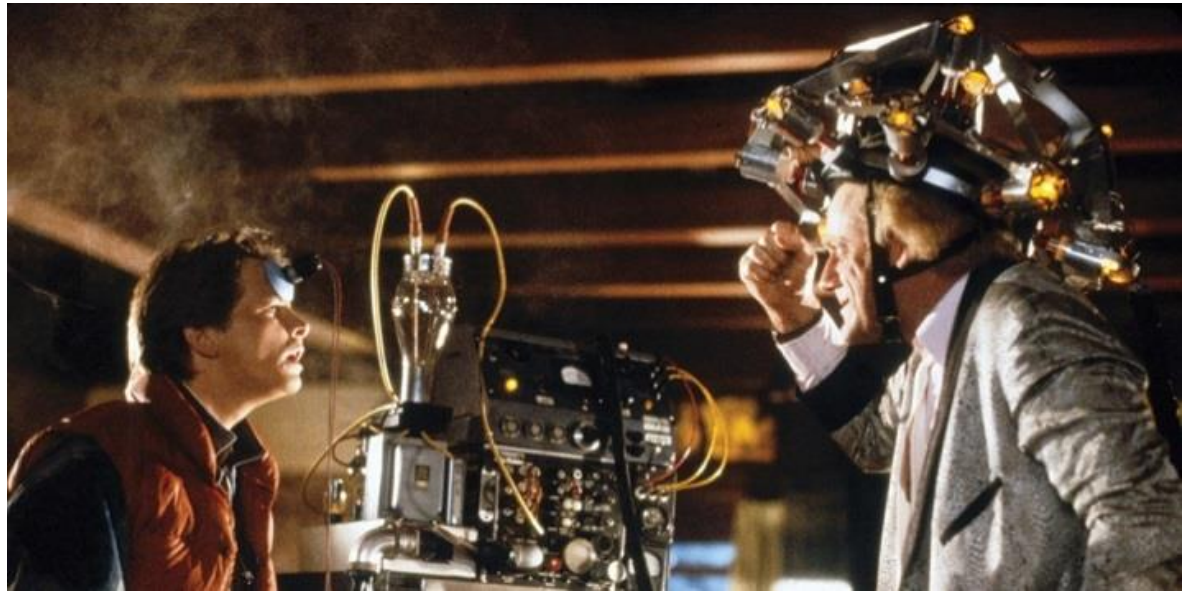
- All ****other**** rows with same value as current row

EXCLUDE NO OTHERS

- No rows are excluded (default)

Usage

- In the SELECT or ORDER BY clauses
- Can't be used in the where clause
- Can't be nested



Benefits

- Use just one table access
- So, performance
- Readability/maintainability
  (provided you master analytic functions)

Examples

first_value / last_value

- For each employee show the person in the same department who earns the most

DEPARTMENT_ID	FIRST_NAME	LAST_NAME	SALARY	HIGH_SAL	HIGH_SAL_NAME
10	Jennifer	Whalen	4400	4400	Jennifer Whalen
20	Michael	Hartstein	13000	13000	Michael Hartstein
20	Pat	Fay	6000	13000	Michael Hartstein
30	Den	Raphaely	11000	11000	Den Raphaely
30	Alexander	Khoo	3100	11000	Den Raphaely
30	Shelli	Baida	2900	11000	Den Raphaely
30	Sigal	Tobias	2800	11000	Den Raphaely
30	Guy	Himuro	2600	11000	Den Raphaely
30	Karen	Colmenares	2500	11000	Den Raphaely
40	Susan	Mavris	6500	6500	Susan Mavris
50	Matthew	Weiss	8000	8200	Adam Fripp
50	Adam	Fripp	8200	8200	Adam Fripp
50	Payam	Kaufling	7900	8200	Adam Fripp
50	Shanta	Vollman	6500	8200	Adam Fripp

first_value - Doc

You first get the max salary per department

```
(select nvl(department_id,-1),max(salary) max_sal
 from employees
 group by department_id)
```

first_value - Doc

```
(select
    department_id, salary
    first_name || ' ' || last_name name,
from
   employees
where (nvl(department_id,-1),salary) in
    (select nvl(department_id,-1),max(salary) max_sal
     from employees
     group by department_id)
) high_sal,
```

Then the employees that earn that max salary in their department

first_value - Doc

```
select
    e.department_id, e.first_name, e.last_name,
    e.salary, high_sal.salary, high_sal.name
from
    (select
        department_id, salary
        first_name || ' ' || last_name name,
    from
       employees
    where (nvl(department_id,-1),salary) in
       (select nvl(department_id,-1),max(salary) max_sal
        from employees
        group by department_id)
    ) high_sal,
    employees e
where nvl(e.department_id,-1) = nvl(high_sal.department_id,-1)
order by e.department_id, e.employee_id;
```

Finally, you join that to "employees" by department_id to get all employees with the employee with highest salary

first_value - Marty

```
first_value (e.salary)
   over (partition by e.department_id order by e.salary desc)

first_value (e.first_name|| ' ' ||last_name)
   over (partition by e.department_id order by e.salary desc)
```

Actually Doc, why not use the first_value function to partition by department, order by salary desc and get the first_value of salary and name?

first_value - Marty

```
select e.department_id, e.first_name, e.last_name, e.salary,
       first_value (e.salary)
          over (partition by e.department_id order by e.salary desc)
          as high_sal,
       first_value (e.first_name|| ' ' ||last_name)
          over (partition by e.department_id order by e.salary desc)
          as high_sal_name
from   employees    e
order by e.department_id,
          e.employee_id;
```

Lag / Lead

- I need a list of orders with the number of days since the previous order of the customer

CUSTOMER_ID	CUST_FIRST_NAME	CUST_LAST_NAME	ORDER_DATE	DAYS_SINCE_LAST
101	Constantin	Welles	2007-08-16 14:34:12	
101	Constantin	Welles	2007-10-02 05:18:36	47
101	Constantin	Welles	2008-03-29 12:34:04	179
101	Constantin	Welles	2008-07-27 07:59:10	120
102	Harrison	Pacino	2006-09-14 06:03:04	
102	Harrison	Pacino	2007-03-29 13:22:40	196
102	Harrison	Pacino	2007-09-14 08:53:40	169
102	Harrison	Pacino	2007-11-19 15:41:54	66
103	Manisha	Taylor	2006-03-29 14:34:50	
103	Manisha	Taylor	2006-09-01 07:59:15	156
103	Manisha	Taylor	2007-09-13 09:19:00	377
103	Manisha	Taylor	2007-10-02 16:49:34	19
104	Harrison	Sutherland	2006-01-26 10:22:51	
104	Harrison	Sutherland	2007-03-29 15:41:20	427
104	Harrison	Sutherland	2007-09-01 08:53:26	156
104	Harrison	Sutherland	2008-07-14 17:18:23	317
105	Matthias	MacGraw	2007-03-20 16:49:10	
105	Matthias	MacGraw	2007-08-31 09:19:37	164
105	Matthias	MacGraw	2008-01-08 18:03:12	130
105	Matthias	MacGraw	2008-01-26 10:22:41	18
106	Matthias	Hannah	2004-03-20 17:18:21	
106	Matthias	Hannah	2006-01-08 22:34:13	659
106	Matthias	Hannah	2008-05-14 19:59:08	857

Lag - Doc

Simple!
Self-join orders to older orders

Group on customer

Get most recent (older) order date

Subtract from order date

Done

But, Doc??
You go to the table twice, then disregard what you don't need!

Lag - Marty

```
select


,

            over (partition by customer_id

                  )

from   orders

;
```

Just partition on the customer

Lag - Marty

```
select


,
            lag (order_date)
            over (partition by customer_id
                  order by order_date
                 )

from   orders


;
```

Then order by order date

Get the previous order date

Lag - Marty

```
select customer_id
,      cust_first_name
,      cust_last_name
,      order_date
,      round(order_date
             -
             lag (order_date)
             over (partition by customer_id
                    order by order_date
                   )
           )     days_since_last
from   orders
order by customer_id
,        order_date
;
```

Subtract from order date, done.

Lag - Plans

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT		105	7140	4	00:00:01
1	SORT GROUP BY		105	7140	4	00:00:01
2	NESTED LOOPS OUTER		105	7140	3	00:00:01
3	TABLE ACCESS FULL	ORDERS	105	4830	3	00:00:01
* 4	TABLE ACCESS BY INDEX ROWID BATCHED	ORDERS	1	22	0	00:00:01
* 5	INDEX RANGE SCAN	OE_ORDER_CUSTOMER_I	2		0	00:00:01

Look Doc, this is the plan for your statement

...and this is the one for mine

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT		105	4830	3	00:00:01
1	WINDOW SORT		105	4830	3	00:00:01
2	TABLE ACCESS FULL	ORDERS	105	4830	3	00:00:01

Lag / Lead

- My status history table is not allowed to have overlaps or gaps in status validity

ACTION_ID	STATUS	START_DATE	END_DATE
1	initialized	2021-04-01 14:02:36	2021-04-21 09:33:42
1	started	2021-04-21 09:33:42	
2	initialized	2021-04-03 00:58:42	2021-04-17 10:08:04
2	started	2021-04-17 10:08:04	2021-05-11 07:57:20
2	active	2021-05-11 07:57:20	
3	initialized	2021-04-28 12:09:42	2021-05-12 05:07:52
3	started	2021-05-12 05:07:52	
4	initialized	2021-04-10 19:50:00	
5	initialized	2021-05-09 09:37:52	2021-05-17 20:58:04
5	started	2021-05-17 20:58:04	2021-06-15 09:30:44
5	active	2021-06-15 09:30:44	2021-07-04 16:15:44
5	terminated	2021-07-04 16:15:44	
6	initialized	2021-04-20 02:17:05	2021-05-02 06:04:39
6	started	2021-05-02 06:04:39	2021-05-03 12:55:56
6	active	2021-05-03 12:55:56	2021-05-05 05:06:08
6	terminated	2021-05-05 05:06:08	
7	initialized	2021-03-29 07:13:35	
8	initialized	2021-03-24 04:50:25	2021-04-04 09:22:22
8	started	2021-04-04 09:22:22	
9	initialized	2021-04-25 15:30:46	
10	initialized	2021-04-15 02:48:59	

Lead - Doc

Table must have start-date/end-date columns

insert trigger sets end date of previous row

Watch out: mutating table

Update/delete trigger denies or 'fixes' the gaps it creates

Again: mutating table

And of course, pray to great Scott that nobody will temporarily disable the triggers, ever.

Lead - Marty

Why not
ONLY a start-date?

And then determine
the end date in a view

```
create or replace force
view v_status_hist
as
select action_id
,      status
,      start_date
,      lead(start_date)
       over (partition by action_id
             order by      start_date
            )          end_date
from   status_hist
;
```

rank / dense_rank

- Show podium winners in a competition (1st, 2nd, and 3rd places for Gold, Silver, and Bronze medals)

Podium for 100m freestyle swimming men (Rio 2016):

PLACE	COUNTRY	NAME	TIME
1	Australia	Kyle Chalmers	47.58
2	Belgium	Pieter Timmers	47.8
3	USA	Nathan Adrian	47.85

rank - Doc

```
select
    rownum place, country, name, time
from
    (select
        country, name, time
     from olympics
     order by time
    )
where rownum <= 3;
```

rank - Doc

But what about a tie?

If Nathan Adrian got 47.8s as well?

PLACE	COUNTRY	NAME	TIME
1	Australia	Kyle Chalmers	47.58
2	Belgium	Pieter Timmers	47.8
3	USA	Nathan Adrian	47.8

rank - Marty

```
select *
from
    (select
        rank() over (order by time) place,
        country, name, time
     from olympics
    )
where place <= 3;
```

PLACE	COUNTRY	NAME	TIME
1	Australia	Kyle Chalmers	47.58
2	Belgium	Pieter Timmers	47.8
2	USA	Nathan Adrian	47.8
4	Canada	Santo Condorelli	47.88

dense_rank - Marty

```
select *
from
    (select dense_rank() over (order by time) place, country, name, time
     from olympics
    )
where place <= 3;
```

PLACE	COUNTRY	NAME	TIME
1	Australia	Kyle Chalmers	47.58
2	Belgium	Pieter Timmers	47.8
2	USA	Nathan Adrian	47.8
3	Canada	Santo Condorelli	47.88

max / min

- How much does every employee in the IT and Finance departments earn more than the lowest and less than the highest salary in their department

DEPARTMENT_ID	FIRST_NAME	LAST_NAME	SALARY	LESS_THAN_HIGHEST	MORE_THAN_LOWEST
60	Diana	Lorentz	4200	4800	0
60	David	Austin	4800	4200	600
60	Valli	Pataballa	4800	4200	600
60	Bruce	Ernst	6000	3000	1800
60	Alexander	Hunold	9000	0	4800
100	Luis	Popp	6900	5108	0
100	Ismael	Sciarra	7700	4308	800
100	Jose Manuel	Urman	7800	4208	900
100	John	Chen	8200	3808	1300
100	Daniel	Faviet	9000	3008	2100
100	Nancy	Greenberg	12008	0	5108

max / min - Doc

```
with
  department_extremes as
  (
    select department_id
    ,      min(salary)          lowest_salery
    ,      max(salary)          highest_salary
    from   emp_details_view
    where  department_id  in (60,100)
    group by department_id
  )
```

Okay, get the highest and lowest salary

DEPARTMENT_ID	LOWEST_SALERY	HIGHEST_SALARY
60	4200	9000
100	6900	12008

max / min - Doc

Then self-join that to the original table

And do some calculation

```
...
select vw.department_name
,      vw.first_name
,      vw.last_name
,      vw.salary
,      ex.highest_salary - vw.salary  less_than_highest
,      vw.salary - ex.lowest_salery   more_than_lowest
from   emp_details_view       vw
join   department_extremes    ex
  on   ex.department_id = vw.department_id
order by vw.department_name
,        vw.salary
;
```

max / min - Doc

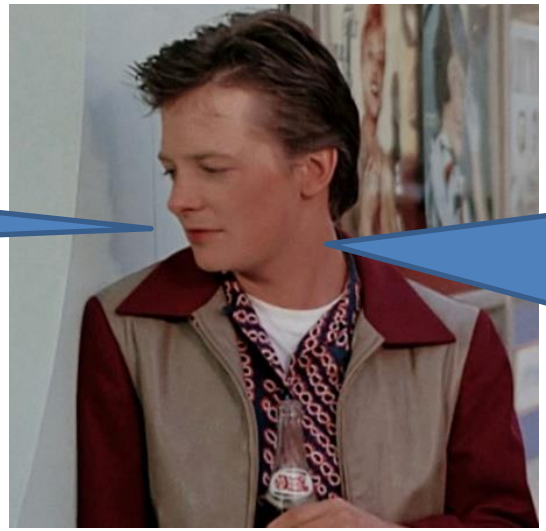
DEPARTMENT_NAME	FIRST_NAME	LAST_NAME	SALARY	LESS_THAN_HIGHEST	MORE_THAN_LOWEST
Finance	Luis	Popp	6900	5108	0
Finance	Ismael	Sciarra	7700	4308	800
Finance	Jose Manuel	Urman	7800	4208	900
Finance	John	Chen	8200	3808	1300
Finance	Daniel	Faviet	9000	3008	2100
Finance	Nancy	Greenberg	12008	0	5108
IT	Diana	Lorentz	4200	4800	0
IT	Valli	Pataballa	4800	4200	600
IT	David	Austin	4800	4200	600
IT	Bruce	Ernst	6000	3000	1800
IT	Alexander	Hunold	9000	0	4800

There!!
Doesn't get much simpler than that

Hold my beer.....

max / min - Marty

Just use the max and min analytic functions



Other aggregates like SUM/COUNT/AVG also available as analytic functions

```
select department_id
,      first_name
,      last_name
,      salary
,      max(salary) over (partition by department_id)
       - salary                                          less_than_highest
,      salary -
       min(salary) over (partition by department_id)     more_than_lowest
from   employees
where  department_id  in (60,100)
order by department_id
,        salary
;
```

Ntile

- The nightly invoicing job runs too long. We need it to evenly distribute the work over 10 subjobs. Determine the work to be done by each subjob

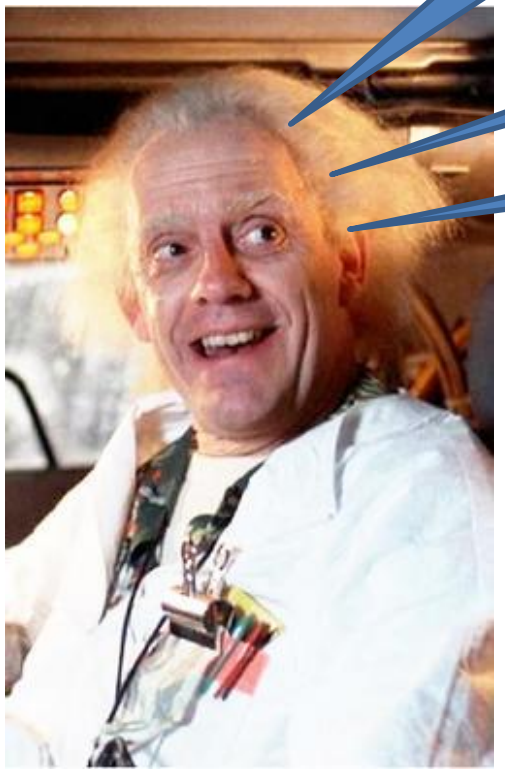
```
ERO@EVROCS>desc invoices

             Name       Null?            Type
_________________ ____________ _______________
ID                NOT NULL     NUMBER(10)
CUSTOMER_ID       NOT NULL     NUMBER(10)
READY_TO_SEND                  VARCHAR2(1)
```

Ntile - Doc

Hah!
Simple.

Add a rownum to the select

A mod(rownum,10) will give each record the jobnr that should process it



```
select id
,      mod(rownum,10)      job_id
from   invoices
where  ready_to_send = 'Y'
;
```

ID	JOB_ID
1	1
2	2
4	3
7	4
10	5
12	6
13	7
14	8
15	9
16	0
17	1
18	2
19	3
20	4
21	5
22	6
23	7
25	8
26	9
27	0
28	1

Ntile - Marty

Hold on, Doc!
It's still not fast enough.

Because ID's that are close to each other are in the same block.

All jobs are constantly competing for the same block

We need:
Job 1: first 10% of the rows
Job 2: second 10% of the rows
Etc.

ID	JOB_ID
1	1
2	1
4	1
7	1
10	1

1009	1
1010	1
1011	1
1012	2
1013	2
1015	2

Ntile - Doc

Determine number of rows in each job

Without decimals (floor)

Determine number of 'leftovers'

Hmm tough one!

ordered by ID
Get a row number

calculate row number for first and last row in each job

Spread out 'leftovers' over the first jobs

Assign job ID to each row, based on row number and first/last row number of jobs

Ntile - Doc

```
with
  table_dimensions as
  (
    select floor(count(*) / 10)   min_rows_per_job
    ,      mod(count(*),10)       leftover_rows
    from   invoices
    where  ready_to_send = 'Y'
  )
, job_sizes as
  (
    select level         job_id
    ,      min_rows_per_job
           +
           case
             when level <= leftover_rows
             then 1
             else 0
           end           rows_to_process
    from   table_dimensions
    connect by level <= 10
  )
, job_dimensions as
  (
    select job.job_id
    ,      nvl((select sum(prv.rows_to_process)
                from   job_sizes   prv
                where  prv.job_id < job.job_id
               )
              ,0
              ) + 1            start_row_seq
    ,      (select sum(prv.rows_to_process)
            from   job_sizes   prv
            where  prv.job_id <= job.job_id
           )                   end_row_seq
    from   job_sizes   job
  )
, row_sequence as
  (
    select id
    ,      rownum     row_seq
    from   (
             select id
             from   invoices
             where  ready_to_send = 'Y'
             order by id
           )
  )
, row_jobs as
  (
    select rs.id
    ,      jd.job_id
    from   row_sequence    rs
    join   job_dimensions  jd
      on   rs.row_seq between jd.start_row_seq
                      and     jd.end_row_seq
  )
select job_id
,      min(id)   job_start_id
,      max(id)   job_end_id
,      count(*)  job_rows
from   row_jobs
group by job_id
order by job_id
;
```

JOB_ID	JOB_START_ID	JOB_END_ID	JOB_ROWS
1	1	1011	751
2	1012	2012	751
3	2013	2996	751
4	2998	4039	751
5	4042	5035	751
6	5036	6014	751
7	6015	7018	751
8	7019	7992	751
9	7993	8998	751
10	8999	10000	750

Yes!

An unnecessarily very complex query!!

Ntile - Marty

```
select id
,      ntile (10)
       over (order by id)  job_id
from   invoices
where  ready_to_send = 'Y'
;
```

ID	JOB_ID
1	1
2	1
4	1
7	1
1009	1
1010	1
1011	1
1012	2
1013	2

Just use NTILE to assign a job ID to every row

Ntile - Marty

```
select job_id
,      min(id)    job_start_id
,      max(id)    job_end_id
,      count(*)   job_rows
from   (
        select id
        ,      ntile (10)
               over (order by id)  job_id
        from   invoices
        where  ready_to_send = 'Y'
       )
group by job_id
order by job_id
;
```

And then group that

JOB_ID	JOB_START_ID	JOB_END_ID	JOB_ROWS
1	1	1011	751
2	1012	2012	751
3	2013	2996	751
4	2998	4039	751
5	4042	5035	751
6	5036	6014	751
7	6015	7018	751
8	7019	7992	751
9	7993	8998	751
10	8999	10000	750

Mix Different Functions

- Find the top 2 earners in each department, but we're only interested in the 3 largest departments

FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_ID
Den	Raphaely	11000	30
Alexander	Khoo	3100	30
Adam	Fripp	8200	50
Matthew	Weiss	8000	50
John	Russell	14000	80
Karen	Partners	13500	80

Mix Different Functions - Doc

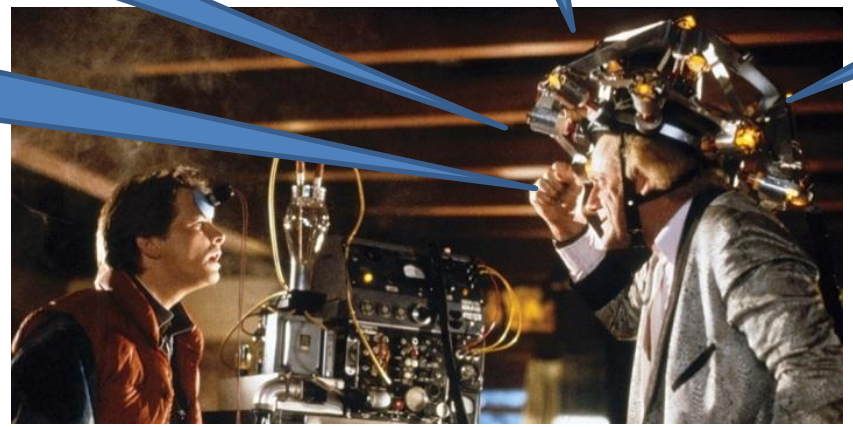
Oh, to many round trips?

Then a few self joins

Then query each department for top 2 earners

Or PL/SQL

Easy! Let's get the largest departments based on count(*)



Mix Different Functions - Marty

```
select first_name,
       last_name,
       salary,
       department_id,
       count(*) over (partition by department_id) dep_size,
       row_number()
           over (partition by department_id
                 order by salary desc) sal_rank
from employees
```

dep_size is using count(*) to count how many people there are in each department

sal_rank is ranking each employee in their own department based on salary

All together - Marty

```
select first_name, last_name, salary, department_id,
       dep_size, sal_rank
       dense_rank() over (order by dep_size desc) dep_rank
from
    (select first_name,
            last_name,
            salary,
            department_id,
            count(*) over (partition by department_id) dep_size,
            row_number()
                  over (partition by department_id
                        order by salary desc) sal_rank
     from employees) e
```

dep_rank is ranking the department based on their size (using the dep_size from previous step)

All together - Marty

```
select first_name, last_name, salary, department_id
from
    (select first_name, last_name, salary, department_id,
            dep_size, sal_rank,
            dense_rank() over (order by dep_size desc) dep_rank
     from
        (select first_name,
                last_name,
                salary,
                department_id,
                count(*) over (partition by department_id) dep_size,
                row_number()
                      over (partition by department_id
                            order by salary desc) sal_rank
         from employees) e
    )
where dep_rank<=3 and sal_rank<=2
order by department_id,salary desc;
```

Now only take the employees that their dep_rank is up to 3, while their sal_rank in the department is 1 or 2

Until next time (sometime in the future)

“Stupid questions do exist.
But it takes a lot more time and energy to correct a stupid mistake than it takes to answer a stupid question, so please ask your stupid questions.”

a wise teacher who taught me more than just physics

Thanks !!