

ANALYTICAL SQL

DECEMBER 2021

Agenda

❖ Day 1

- Introduction to Analytical SQL
- Analytic window functions
- Ranking functions
- LAB

❖ Day 2

- Ranking functions (cont.)
- Windowing
- Aggregate Analytical functions
- LAB

❖ Day 3

- Pivoting operations
- Statistical Aggregates
- Case Study

Ranking Functions

- RANK , DENSE_RANK and ROW_NUMBER Function
- FIRST_VALUE and LAST_VALUE Function
- PERCENT_RANK Function
- NTILE Function
- CUME_DIST Function

PERCENT_RANK

- The PERCENT_RANK() is a window function that calculates the percentile ranking of rows in a result set.
- For a specific row, PERCENT_RANK() uses the following formula to calculate the percentile rank:

$$(\text{rank} - 1) / (\text{total_rows} - 1)$$

- The PERCENT_RANK analytic function is order sensitive so the ORDER BY clause is mandatory.
- The first row of the ordered set is assigned 0 and the last row of the set is assigned 1
- If there is a single row in the set it is assigned 0
- Ties are assigned the same value
- The PERCENT_RANK calculates the relative position in the set.

```

SELECT deptno, sal,
       RANK() OVER (PARTITION BY deptno ORDER BY sal) AS rank,
       PERCENT_RANK()
         OVER (PARTITION BY deptno ORDER BY sal) AS percent_rank_sal,
       ROUND(PERCENT_RANK()
         OVER (PARTITION BY deptno ORDER BY sal)*100) AS percent_rank_sal_pct
FROM   emp
ORDER BY deptno, sal;

```

DEPTNO	SAL	RANK	PERCENT_RANK_SAL	PERCENT_RANK_SAL_PCT
10	1300	1	0	0
10	2450	2	.5	50
10	5000	3	1	100
20	800	1	0	0
20	1100	2	.25	25
20	2975	3	.5	50
20	3000	4	.75	75
20	3000	4	.75	75
30	950	1	0	0
30	1250	2	.2	20
30	1250	2	.2	20

HANDS ON

Find The top 30% of employees in the company based on their salaries

ENAME character varying	SAL numeric	%mk double precision
KING	5000	0
SCOTT	3000	10
JONES	2975	20
BLAKE	2850	30

HANDS ON

Find The top 30% of employees in the company based on their salaries

```
select *  
from (Select "ENAME" , "SAL" , PERCENT_RANK() OVER(ORDER BY "SAL" desc)*100 AS "%rnk"  
      from "General_schema".emp) AS rnk  
where "%rnk" <= 30
```

CUME_DIST

- The CUME_DIST() function calculates the cumulative distribution of a value within a group of values. Simply put, it calculates the relative position of a value in a group of values.
- For a specific row, CUME_DIST() uses the following formula to calculate the percentile rank:

$$(\text{rank}) / (\text{total_rows})$$

- The result of CUME_DIST() is greater than 0 and less than or equal to 1.

```
CUME_DIST() OVER (  
    [PARTITION BY partition_expression, ... ]  
    ORDER BY sort_expression [ASC | DESC], ...  
)
```


Calculates the sales percentile for each sales staff in 2017

```
SELECT
    full_name,
    net_sales,
    CUME_DIST() OVER (
        ORDER BY net_sales DESC
    ) cume_dist
FROM
    sales.vw_staff_sales t
INNER JOIN sales.staffs m on m.staff_id = t.staff_id
WHERE
    year = 2017;
```

full_name	net_sales	cume_dist
Marcelene Boyer	1370320.0000	0.166666666666667
Venita Daniel	1109368.0000	0.333333333333333
Genna Serrano	285771.0000	0.5
Mireya Copeland	277137.0000	0.666666666666667
Layla Terrell	222740.0000	0.833333333333333
Kali Vargas	181872.0000	1

Calculates the sales percentile for each sales staff in 2016 and 2017

```
SELECT
    full_name,
    net_sales,
    year,
    CUME_DIST() OVER (
        PARTITION BY year
        ORDER BY net_sales DESC
    ) cume_dist
FROM
    sales.vw_staff_sales t
INNER JOIN sales.staffs m on m.staff_id = t.staff_id
WHERE
    year IN (2016,2017);
```

full_name	net_sales	year	cume_dist
Venita Daniel	856904.0000	2016	0.166666666666667
Marcelene Boyer	733695.0000	2016	0.333333333333333
Genna Serrano	320342.0000	2016	0.5
Mireya Copeland	245152.0000	2016	0.666666666666667
Kali Vargas	146934.0000	2016	0.833333333333333
Layla Terrell	124353.0000	2016	1
Marcelene Boyer	1370320.0000	2017	0.166666666666667
Venita Daniel	1109368.0000	2017	0.333333333333333
Genna Serrano	285771.0000	2017	0.5
Mireya Copeland	277137.0000	2017	0.666666666666667
Layla Terrell	222740.0000	2017	0.833333333333333
Kali Vargas	181872.0000	2017	1

NTILE

- The SQL NTILE() is a window function that allows you to break the result set into a specified number of approximately equal groups, or buckets.
- It assigns each group a bucket number starting from one.
- For each row in a group, the NTILE() function assigns a bucket number representing the group to which the row belongs
- If the number of rows in the set is smaller than the number of buckets specified, the number of buckets will be reduced so there is one row per bucket.
- Unlike some other analytic functions, it doesn't support the windowing clause.

```
SELECT ename, deptno, sal,  
       NTILE(5) OVER (ORDER BY sal) AS quintile
```

```
FROM emp;
```

ENAME	DEPTNO	SAL	QUINTILE
-----	-----	-----	-----
SMITH	20	800	1
JAMES	30	950	1
ADAMS	20	1100	1
WARD	30	1250	2
MARTIN	30	1250	2
MILLER	10	1300	2
TURNER	30	1500	3
ALLEN	30	1600	3
CLARK	10	2450	3
BLAKE	30	2850	4
JONES	20	2975	4
SCOTT	20	3000	4
FORD	20	3000	5
KING	10	5000	5

Windowing

- WINDOW FRAME Clause
- ROWS Vs. RANGE Clause
- Interval

LAB1 Q1 : Query Expectation Vs Query Results

```
select userId, sessionId,  
first_value(song) over (partition by sessionId order by ts) as first_song,  
last_value(song) over (partition by sessionId order by ts ) as last_song  
from events  
order by userId, sessionId;
```

userid bigint	sessionId bigint	first_song character varying	last_song character varying
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	323	Macarena	Macarena
2	354	Runaway (Album Version)	Pet Semetary
2	354	Runaway (Album Version)	Pet Semetary
2	354	Runaway (Album Version)	Pet Semetary
3	112	Adios	Strasbourg
3	112	Adios	Strasbourg
3	112	Adios	Strasbourg
4	3	Read My Mind	Read My Mind
4	572	Something Happened On The Way To Heaven	Bracelets (LP Version)
4	572	Something Happened On The Way To Heaven	Bracelets (LP Version)

Vs.

userid bigint	sessionId bigint	first_song character varying	last_song character varying
2	126	Self Control (Laurent Wolf & Anton Wick)	Self Control (Laurent Wolf & Anton Wick)
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Love Stinks
2	126	Self Control (Laurent Wolf & Anton Wick)	The Lady Is A Tramp (From 'Babes In Arms')
2	126	Self Control (Laurent Wolf & Anton Wick)	Tell Me When The Party's Over/Prequiem
2	126	Self Control (Laurent Wolf & Anton Wick)	Hurt Me Soul (Explicit Album Version)
2	323	Macarena	Macarena
2	354	Runaway (Album Version)	Pet Semetary
2	354	Runaway (Album Version)	Runaway (Album Version)
2	354	Runaway (Album Version)	Spark My Soul (feat. Substantial)
3	112	Adios	Adios
3	112	Adios	Pop Corn
3	112	Adios	Strasbourg

Over clause

The OVER clause is used to determine

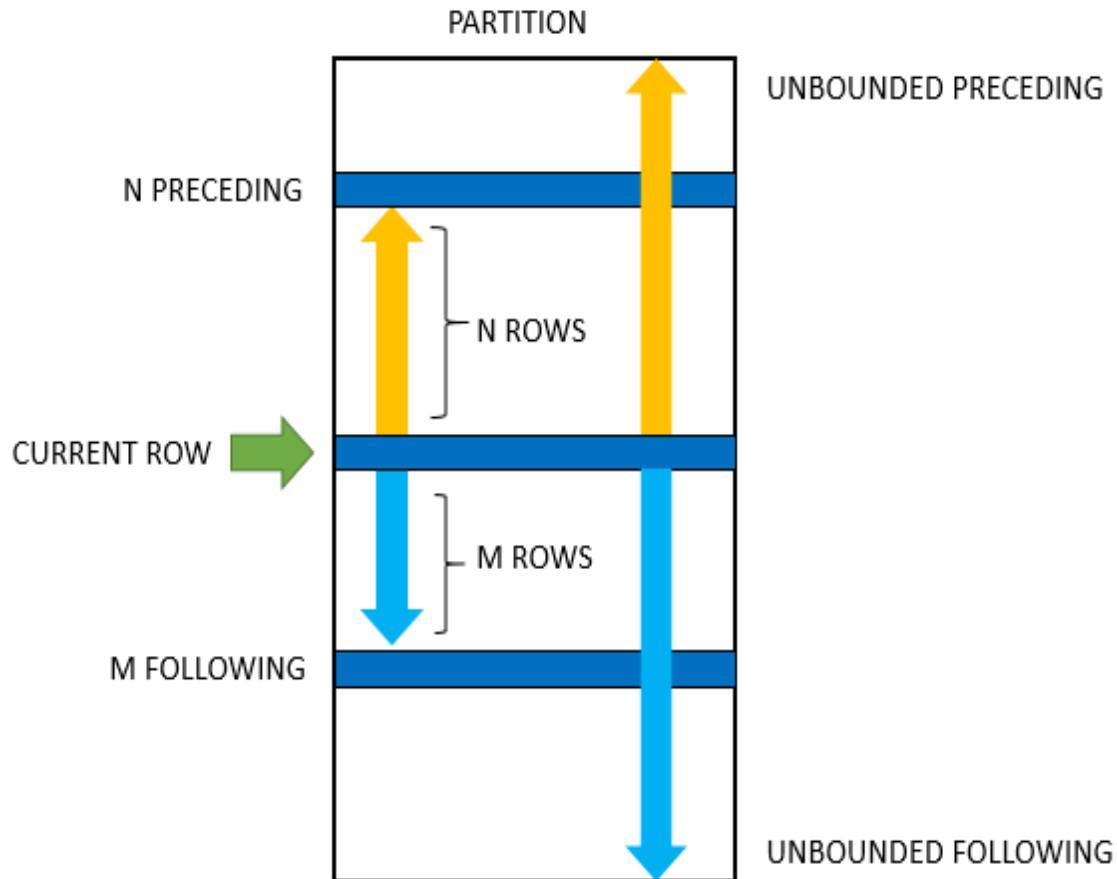
when the function's calculations should restart ==> (PARTITION BY)

what order they are evaluated in by that function ==> (ORDER BY)

which rows from the query are applied to the function ==> (ROWS or RANGE)

```
<function> OVER (
    [PARTITION BY clause]
    [ORDER BY clause]
    [ROWS or RANGE clause]) ←
```

Windowing



UNBOUNDED PRECEDING:

the frame starts at the first row of the partition.

N PRECEDING:

the frame starts at Nth rows before the current row.

CURRENT ROW:

is the current row that is being processed.

UNBOUNDED FOLLOWING:

the frame ends at the final row of the partition.

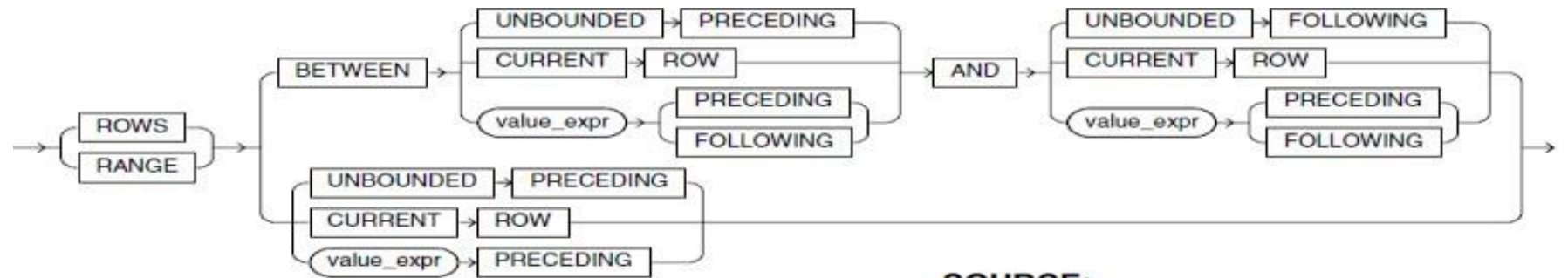
M FOLLOWING:

the frame ends at the Mth row after the current row.

By default, window functions use “**RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW**”

Windowing

- Selects a smaller subset than the partition based on a number of records before/after or a time period before/after.
- Syntax Diagram:



SOURCE:

Oracle Database SQL Language Reference
11g Release 2 (E10592-04)
Page 5-12

```

select userId, sessionId,
first_value(song) over (partition by sessionId order by ts) as first_song,
last_value(song) over (partition by sessionId order by ts
                      rows between unbounded preceding and unbounded following ) as last_song
from events
order by userId, sessionId;

```

userid bigint	sessionId bigint	first_song character varying	last_song character varying
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	323	Macarena	Macarena
2	354	Runaway (Album Version)	Pet Semetary
2	354	Runaway (Album Version)	Pet Semetary
2	354	Runaway (Album Version)	Pet Semetary
3	112	Adios	Strasbourg
3	112	Adios	Strasbourg
3	112	Adios	Strasbourg
4	3	Read My Mind	Read My Mind
4	572	Something Happened On The Way To Heaven	Bracelets (LP Version)
4	572	Something Happened On The Way To Heaven	Bracelets (LP Version)

```

SELECT deptno, ename, sal
, SUM ( sal ) OVER ( ) sum1
, SUM ( sal ) OVER ( ORDER BY ename
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING ) sum2
, SUM ( sal ) OVER ( ORDER BY ename
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW ) sum3
FROM emp
WHERE deptno = 10;

```

DEPTNO	ENAME	SAL	SUM1	SUM2	SUM3
10	CLARK	2450	8750	8750	2450
10	KING	5000	8750	8750	7450
10	MILLER	1300	8750	8750	8750

HANDS ON

Find for each employee his salary ,Highest salary in his job and Lowest salary in his job using LAST_VALUE function

	Data Output	Messages	Notifications	Explain	
	EMPNO integer	JOB character varying	SAL numeric	HIGHEST_SAL_IN_JOB numeric	LOWEST_SAL_IN_JOB numeric
1	7788	ANALYST	3000	3000	3000
2	7369	CLERK	800	1100	800
3	7876	CLERK	1100	1100	800
4	7782	MANAGER	2450	2975	2450
5	7698	MANAGER	2850	2975	2450
6	7566	MANAGER	2975	2975	2450
7	7839	PRESIDENT	5000	5000	5000
8	7654	SALESMAN	1250	1600	1250
9	7521	SALESMAN	1250	1600	1250
10	7844	SALESMAN	1500	1600	1250
11	7499	SALESMAN	1600	1600	1250

Find for each employee his salary ,Highest salary in his job and Lowest salary in his job using LAST_VALUE function

```
SELECT "EMPNO" , "JOB" ,"SAL"  
,last_VALUE("SAL") OVER(PARTITION BY "JOB"  
                        ORDER BY "SAL"  
                        rows between unbounded preceding  
                        and unbounded following) AS "HIGHEST_SAL_IN_JOB"  
  
,last_value("SAL") OVER(PARTITION BY "JOB"  
                        ORDER BY "SAL" desc  
                        rows between unbounded preceding  
                        and unbounded following) AS "LOWEST_SAL_IN_JOB"  
  
FROM "General_schema".emp  
ORDER BY "JOB" , "SAL"
```

Shortcut

- If you omit **BETWEEN** and **AND** then the windowing value is \leq CURRENT ROW
- The second argument is assumed to be CURRENT ROW

ROWS UNBOUNDED PRECEDING



ROWS BETWEEN UNBOUNDED PRECEDING
AND CURRENT ROW

ROWS 10 PRECEDING



ROWS BETWEEN 10 PRECEDING
AND CURRENT ROW

ROWS CURRENT ROW



ROWS BETWEEN CURRENT ROW
AND CURRENT ROW

```

SELECT deptno, ename, sal
, SUM ( sal ) OVER ( ORDER BY ename
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING ) sum1
, SUM ( sal ) OVER ( PARTITION BY deptno ORDER BY ename
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING ) sum2
FROM emp;

```

DEPTNO	ENAME	SAL	SUM1	SUM2
20	ADAMS	1100	2700	4100
30	ALLEN	1600	5550	4450
30	BLAKE	2850	6900	5400
10	CLARK	2450	8300	7450
20	FORD	3000	6400	7075
30	JAMES	950	6925	5050
20	JONES	2975	8925	8975
10	KING	5000	9225	8750
30	MARTIN	1250	7550	3700
10	MILLER	1300	5550	6300
20	SCOTT	3000	5100	6775

Windowing clause comparison

- Rows

Restricts window by records based on ORDER BY

```
ROWS BETWEEN 10 PRECEDING  
AND 10 FOLLOWING
```

Analytic function will include the 10 records just before this record and the 10 records after

- Range (Oracle SQL)

Restricts window by a value references field used in ORDER BY

```
RANGE BETWEEN 200 PRECEDING  
AND 200 FOLLOWING
```

Analytic function will include all records within \$200 of the record in question


```
select "ENAME" , "SAL" ,  
SUM("SAL") OVER(ORDER BY "SAL" ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW ) AS SUM1,  
SUM("SAL") OVER(ORDER BY "SAL" RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW ) AS SUM2  
from "General_schema".emp
```

ENAME character varying	SAL numeric	sum1 numeric	sum2 numeric
SMITH	800	800	800
ADAMS	1100	1900	1900
WARD	1250	3150	4400
MARTIN	1250	4400	4400
TURNER	1500	5900	5900
ALLEN	1600	7500	7500
CLARK	2450	9950	9950
BLAKE	2850	12800	12800
JONES	2975	15775	15775
SCOTT	3000	18775	18775
KING	5000	23775	23775

HANDS ON

Find for each employee the number of employees who **nearly** gets the same salary - 200\$ more or less-

ENAME	SAL	EMPS_200_SAL
SMITH	800	2
JAMES	950	3
ADAMS	1100	5
WARD	1250	4
MARTIN	1250	4
MILLER	1300	5
TURNER	1500	3
ALLEN	1600	2
CLARK	2450	1
BLAKE	2850	4
JONES	2975	4

HANDS ON

```
SELECT ename, sal
, COUNT(*) OVER ( ORDER BY sal RANGE BETWEEN 200 PRECEDING
                  AND 200 FOLLOWING ) emps_200_sal
FROM emp
ORDER BY sal;
```

Consider only those records within \$200 of the value from the current record

Which field? SAL: The field that is used in the ORDER BY

ENAME	SAL	EMPS_200_SAL
-----	-----	-----
SMITH	800	2
JAMES	950	3
ADAMS	1100	5
WARD	1250	4
MARTIN	1250	4
MILLER	1300	5
TURNER	1500	3
ALLEN	1600	2
CLARK	2450	1
BLAKE	2850	4
JONES	2975	4