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

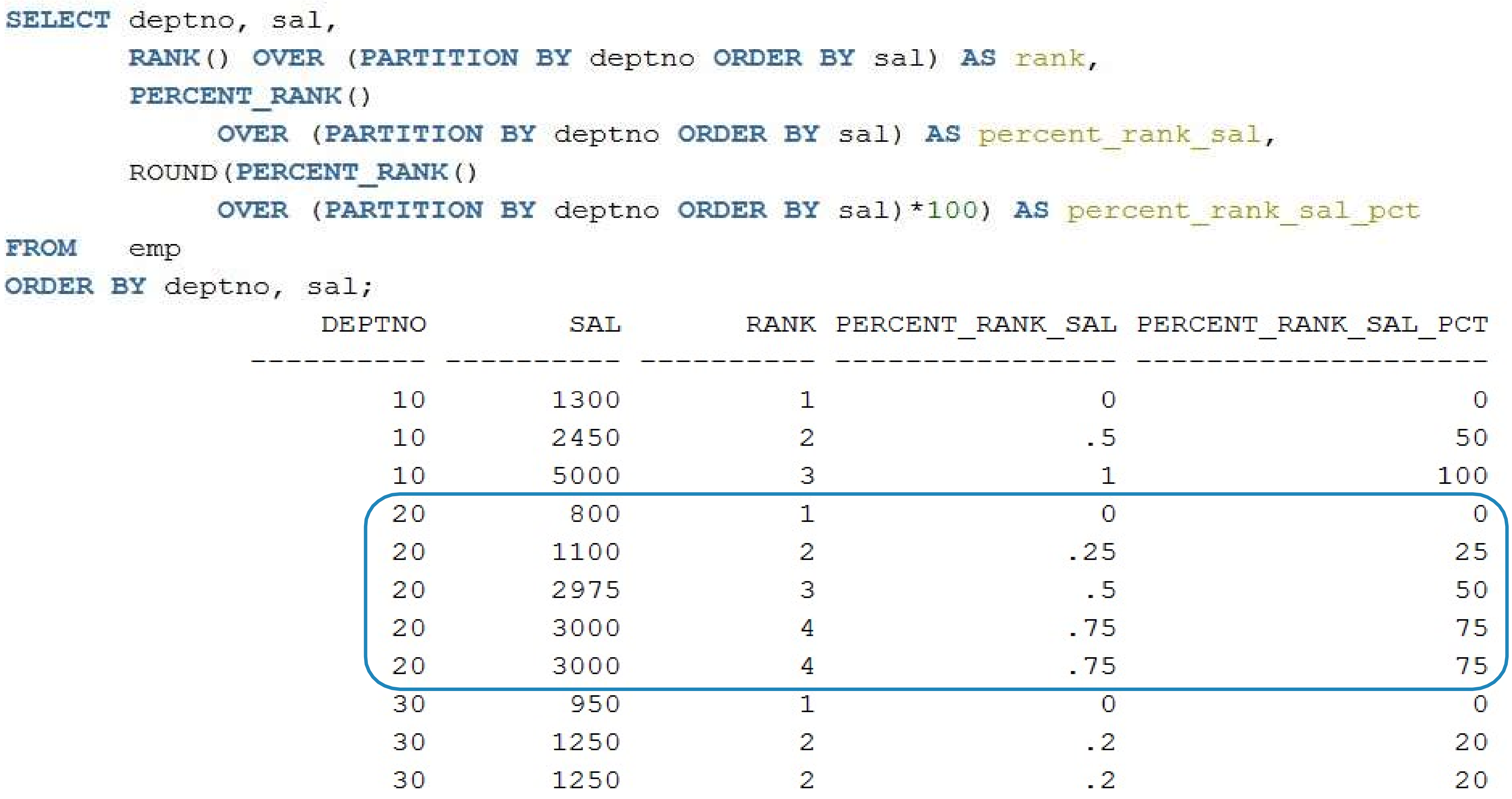
Ranking Functions

- 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:

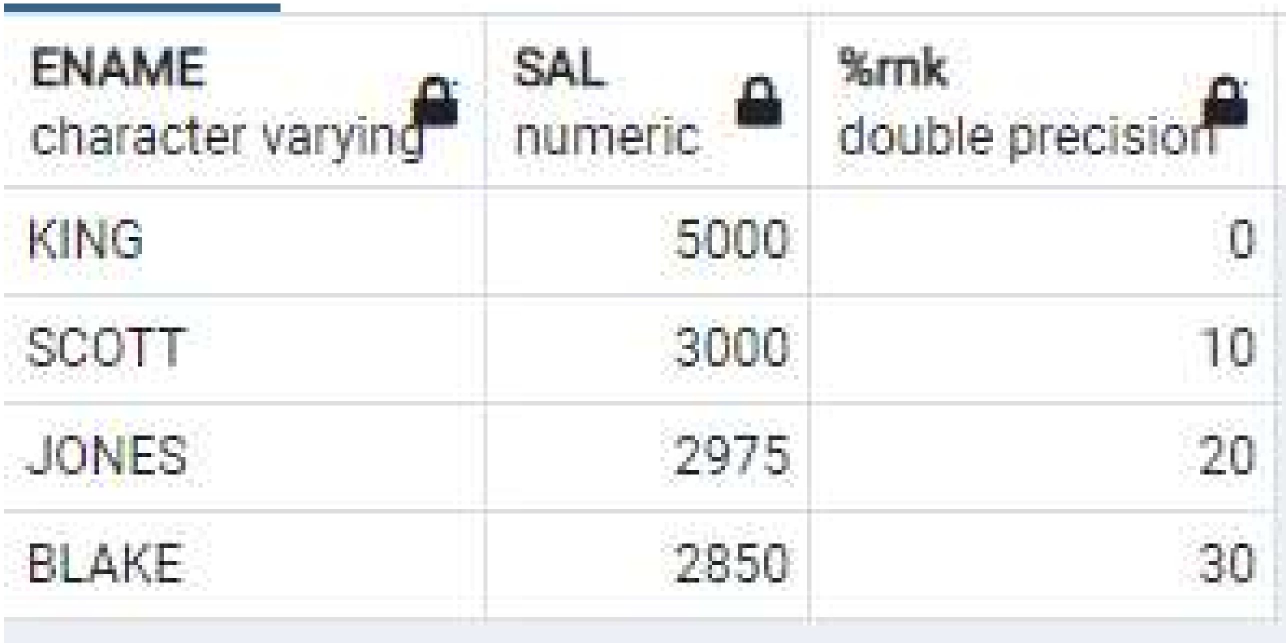
  (rank - 1) / (total_rows - 1)
- The PERCENT_RANK analytic function is order sensitive so the ORDER BY clause is mandatory.
- The first row of the ordered set is assigned 0 and the last row of the set is assigned 1
- If there is a single row in the set it is assigned 0
- Ties are assigned the same value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

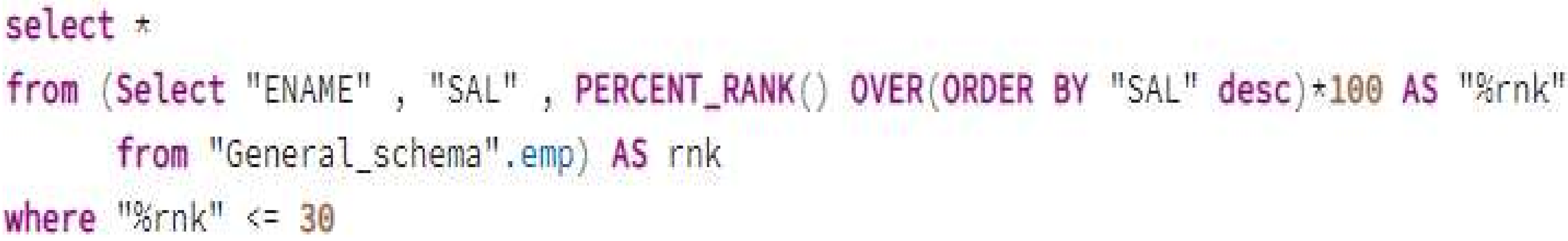
Find The top 30% of employees in the company based on their salaries



ENAME character varying	SAL numeric	%rnk 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



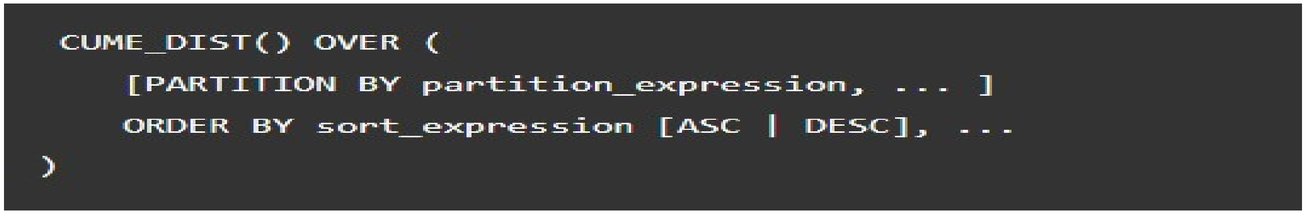
```
select *
from (Select "ENAME" , "SAL" , PERCENT_RANK() OVER(ORDER BY "SAL" desc)*100 AS "%rnk"
      from "General_schema".emp) AS rnk
where "%rnk" <= 30
```

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:

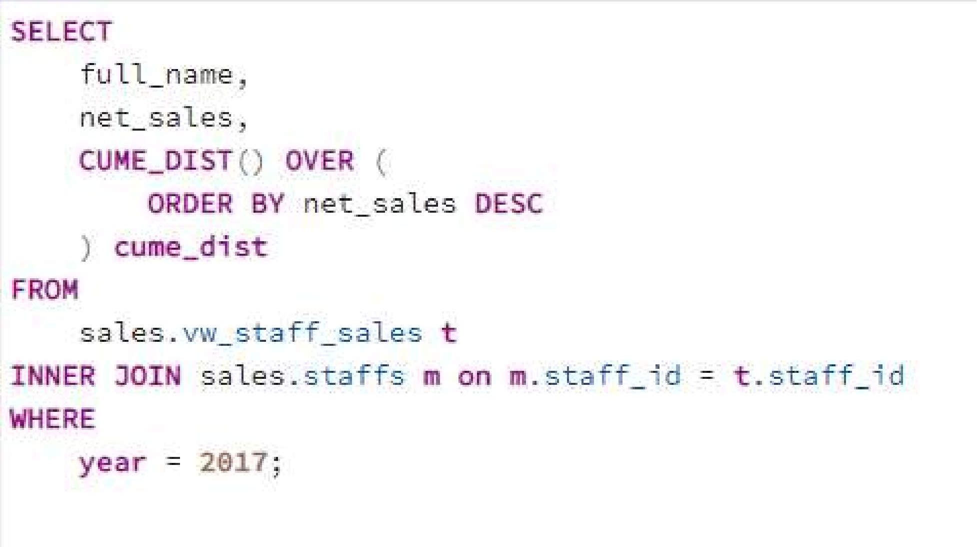
  (rank) / (total_rows)

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

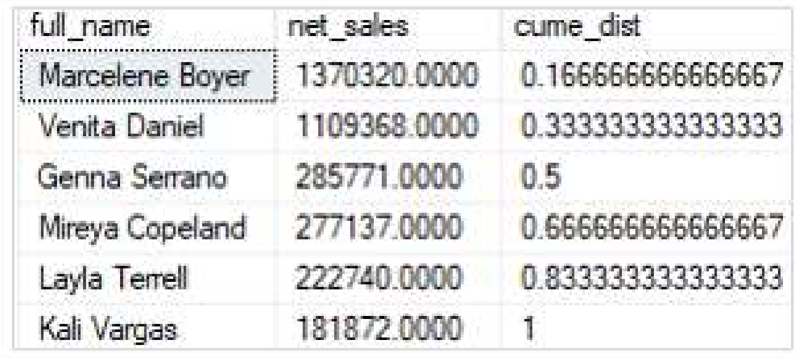


```
CUME_DIST() OVER (
    [PARTITION BY partition_expression, ... ]
    ORDER BY sort_expression [ASC | DESC], ...
)
```

Calculates the sales percentile for each sales staff in 2017

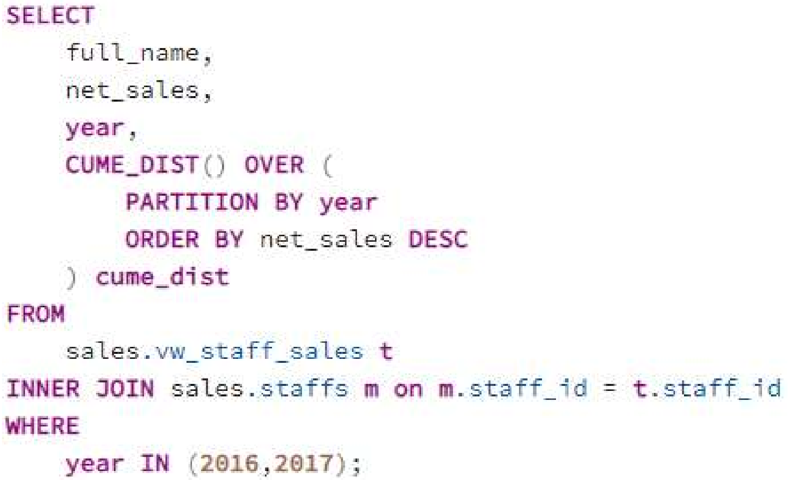


```
SELECT
    full_name,
    net_sales,
    CUME_DIST() OVER (
        ORDER BY net_sales DESC
    ) cume_dist
FROM
    sales.vw_staff_sales t
INNER JOIN sales.staffs m on m.staff_id = t.staff_id
WHERE
    year = 2017;
```

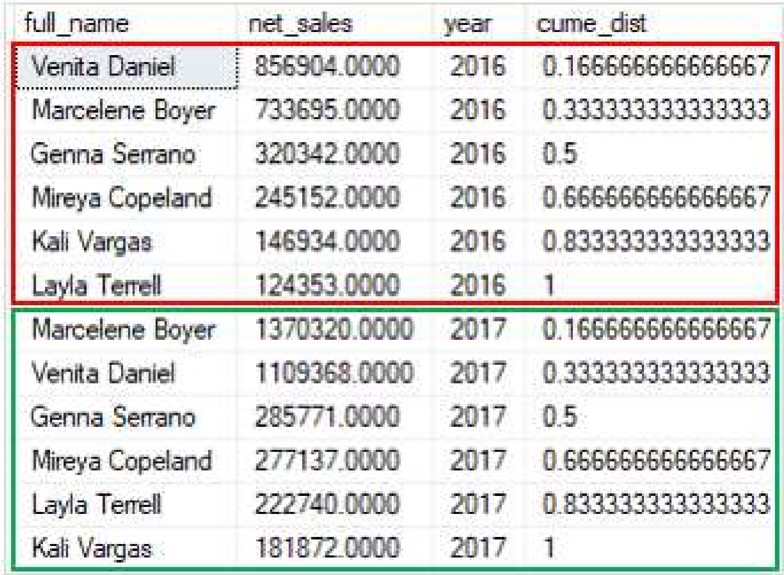


full_name	net_sales	cume_dist
Marcelene Boyer	1370320.0000	0.166666666666667
Venita Daniel	1109368.0000	0.333333333333333
Genna Serrano	285771.0000	0.5
Mireya Copeland	277137.0000	0.666666666666667
Layla Terrell	222740.0000	0.833333333333333
Kali Vargas	181872.0000	1

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;
```

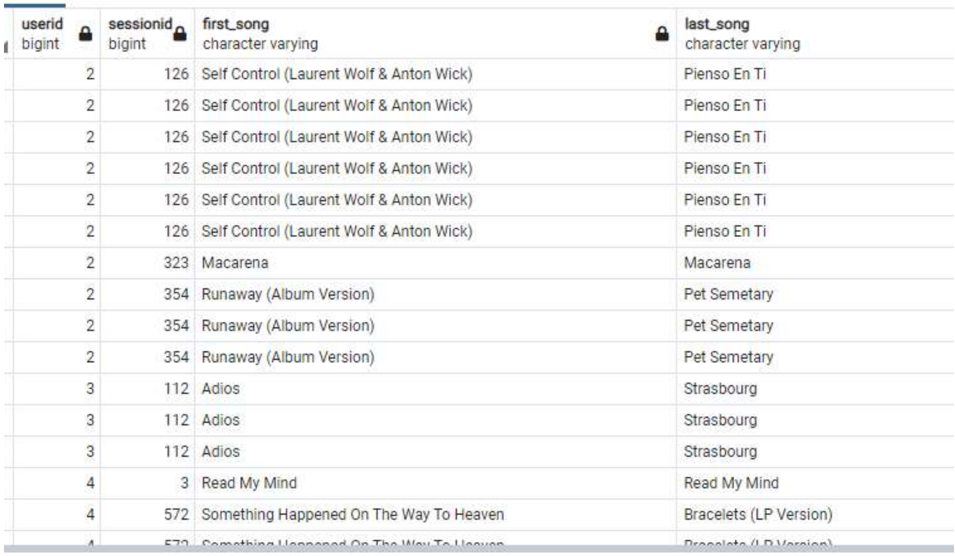
```
ENAME          DEPTNO        SAL   QUINTILE
---------- ---------- ---------- ----------
SMITH              20        800          1
JAMES              30        950          1
ADAMS              20       1100          1
WARD               30       1250          2
MARTIN             30       1250          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

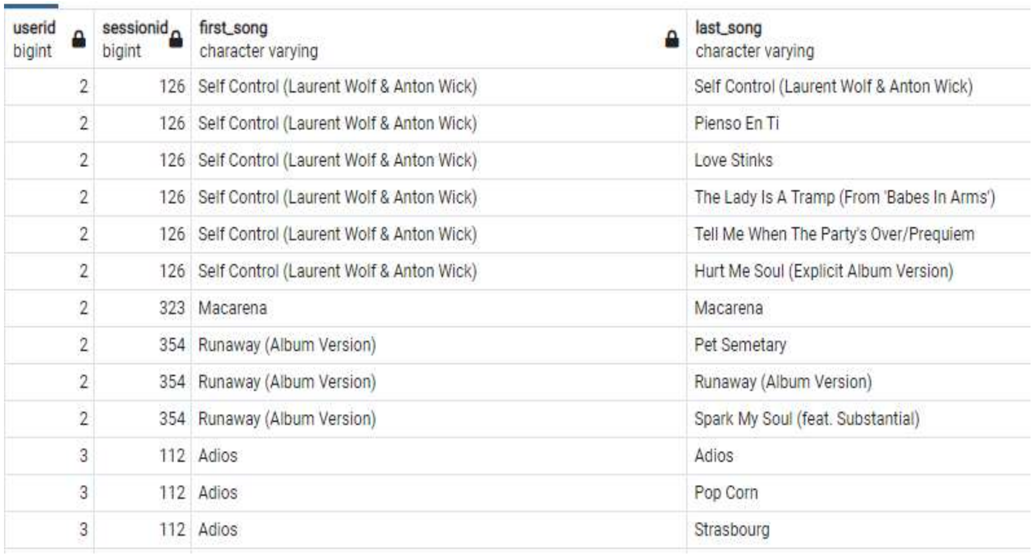
LAB1 Q1 : Query Expectation Vs Query Results

```
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

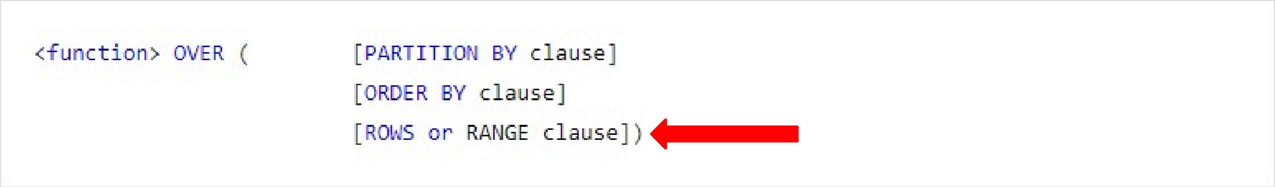
Over clause

The OVER clause is used to determine

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

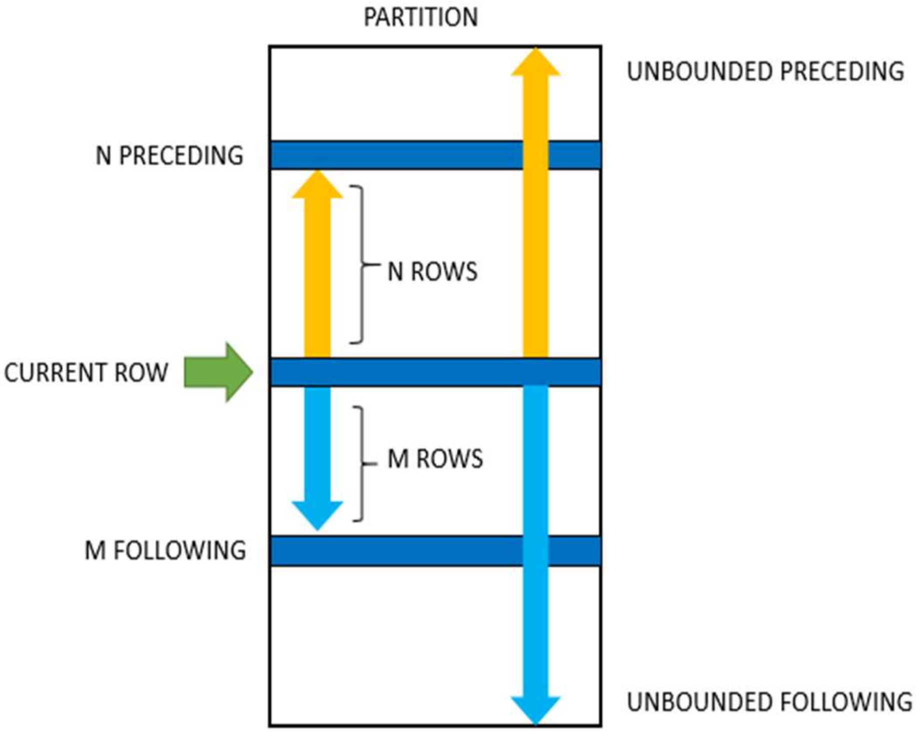
what order they are evaluated in by that function ==> (ORDER BY)

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.

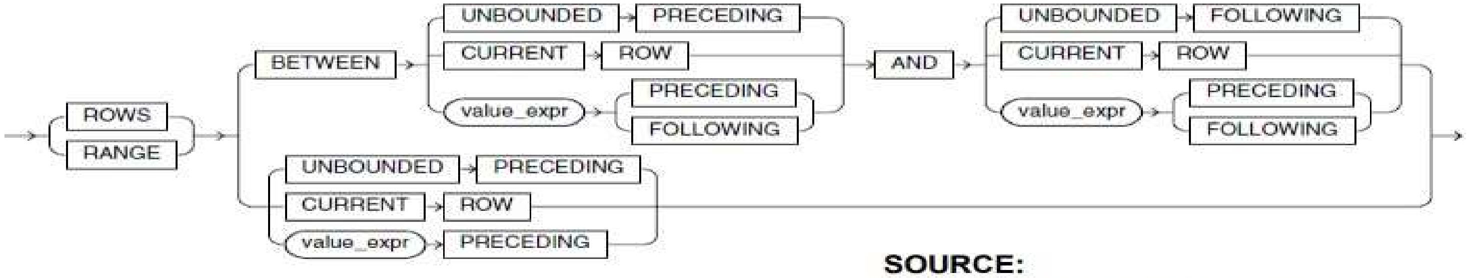
UNBOUNDED FOLLOWING:
the frame ends at the final row of the partition.

M FOLLOWING:
the frame ends at the Mth row after the current row.

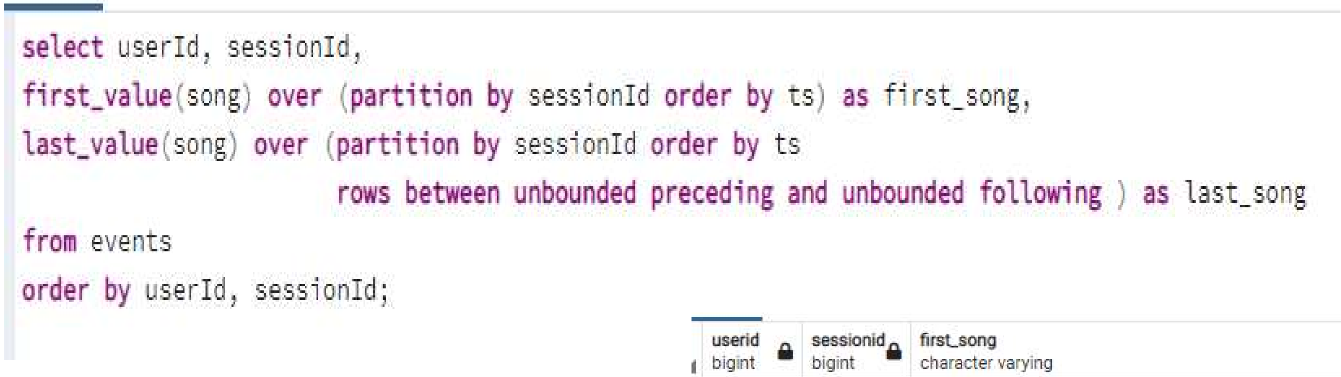
By default, window functions use “**RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW**”

Windowing

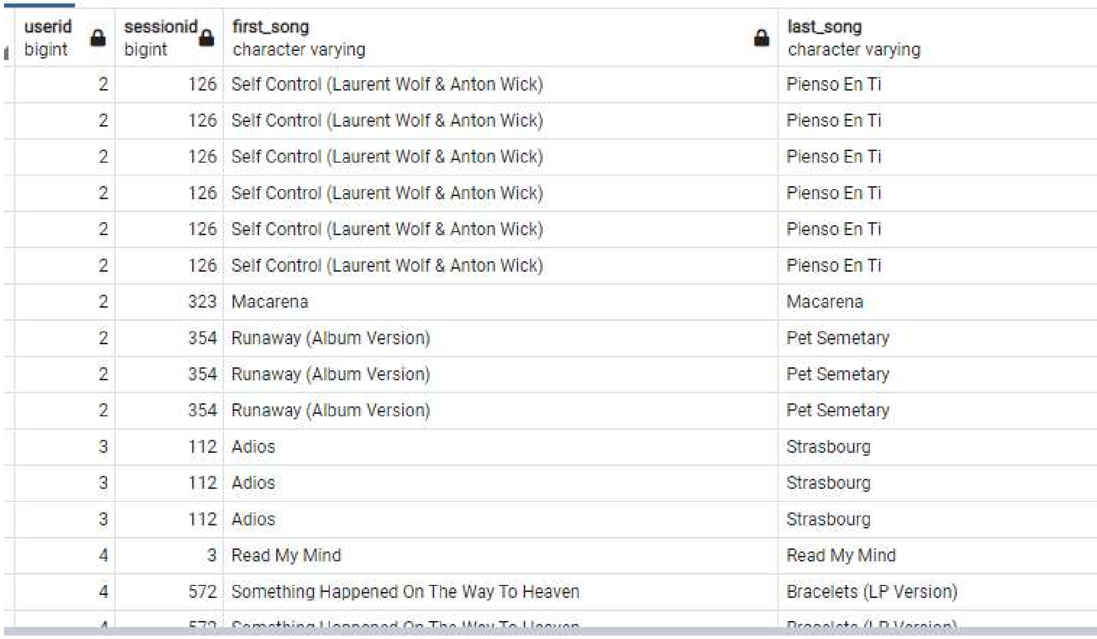
- Selects a smaller subset than the partition based on a number of records before/after or a time period before/after.
- Syntax Diagram:



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;
```



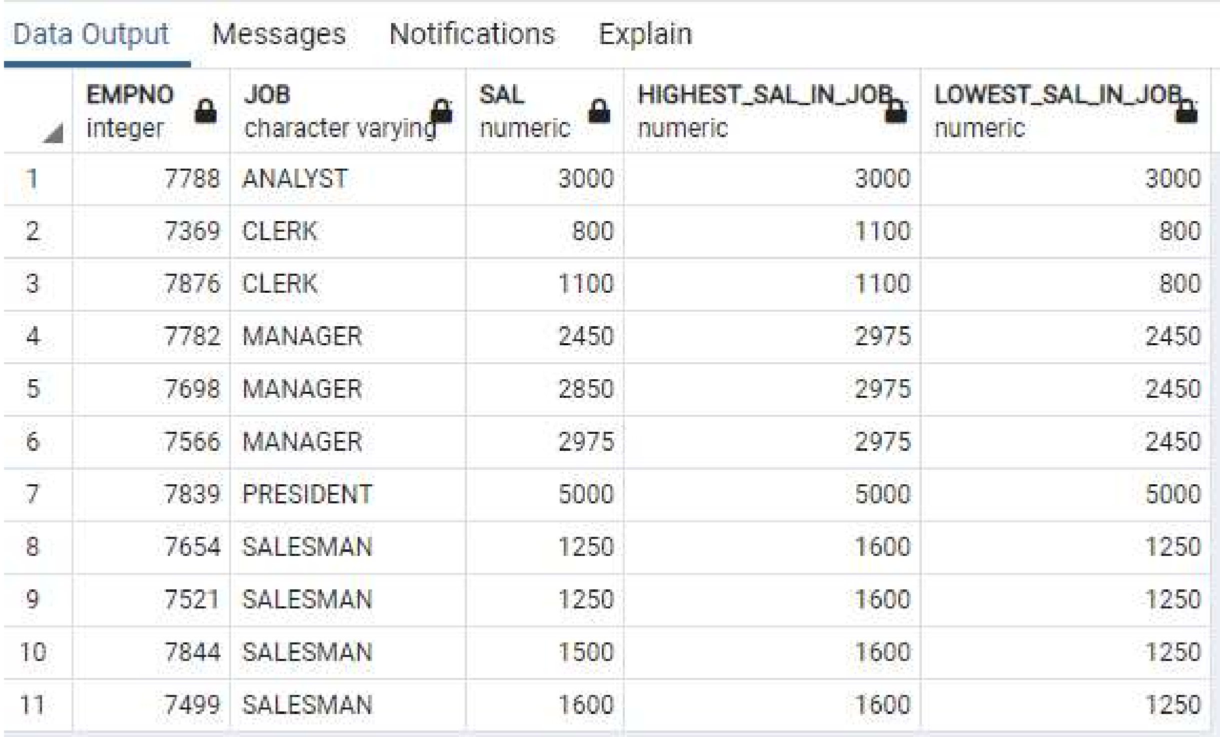
userid bigint	sessionid bigint	first_song character varying	last_song character varying
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & Anton Wick)	Pienso En Ti
2	126	Self Control (Laurent Wolf & 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

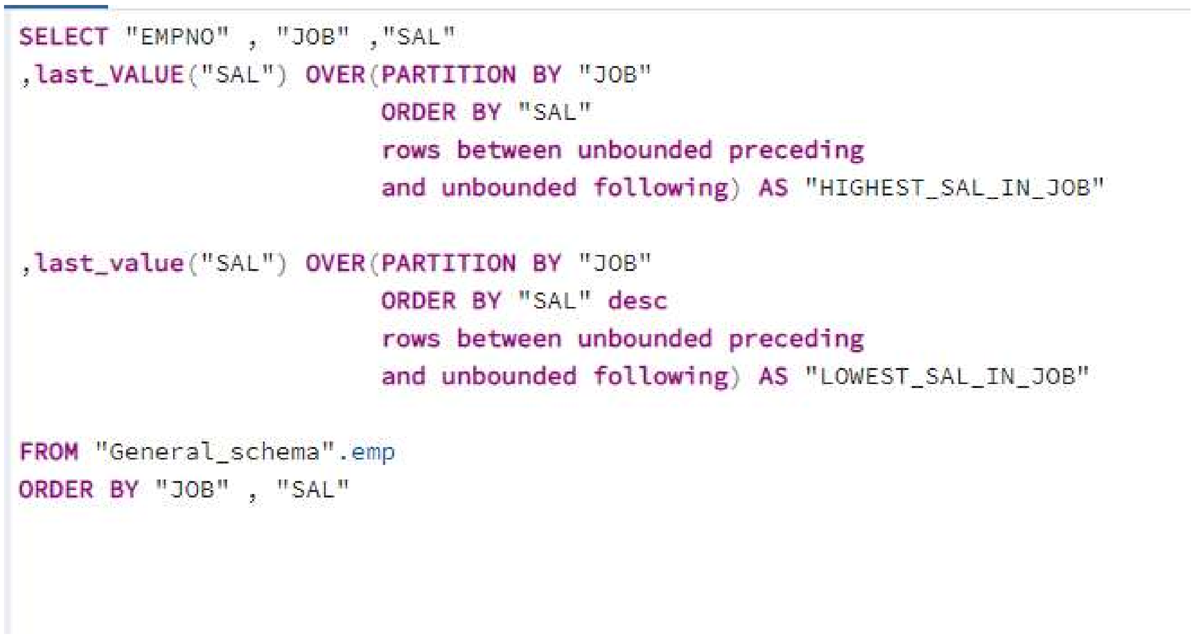
Find for each employee his salary ,Highest salary in his job and Lowest salary in his job using LAST_VALUE function



	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

HANDS ON

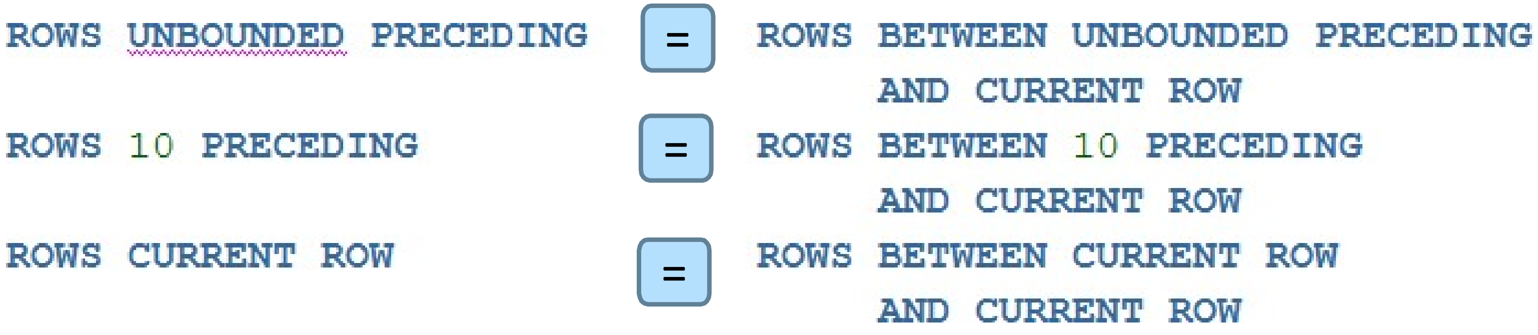
Find for each employee his salary ,Highest salary in his job and Lowest salary in his job using LAST_VALUE function



```
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 <= 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
```

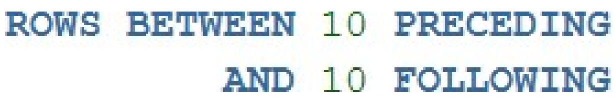
```
SELECT deptno, ename, sal
, SUM ( sal ) OVER ( ORDER BY ename
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING ) sum1
, SUM ( sal ) OVER ( PARTITION BY deptno ORDER BY ename
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING ) sum2
FROM emp;
```

```
    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

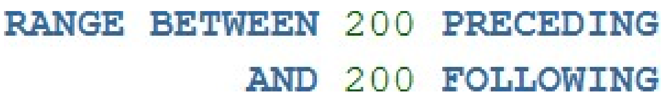


```
ROWS BETWEEN 10 PRECEDING
         AND 10 FOLLOWING
```

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

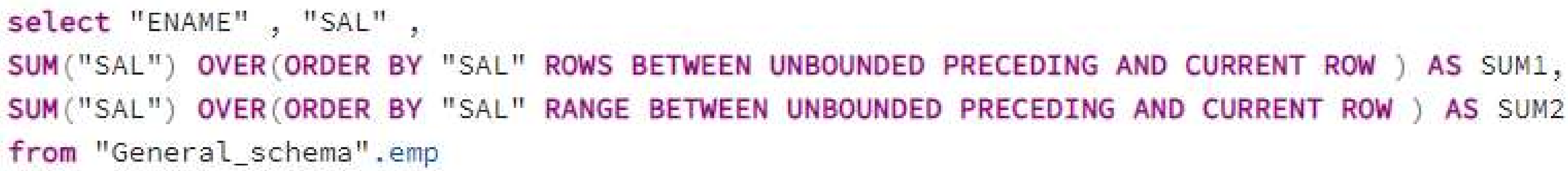
- Range (Oracle SQL)

Restricts window by a value references field used in ORDER BY

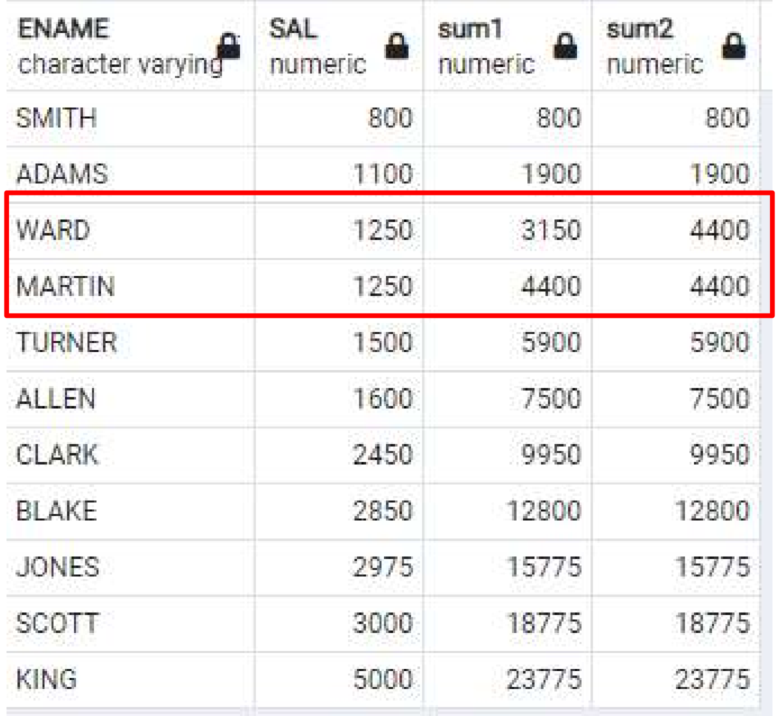


```
RANGE BETWEEN 200 PRECEDING
          AND 200 FOLLOWING
```

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



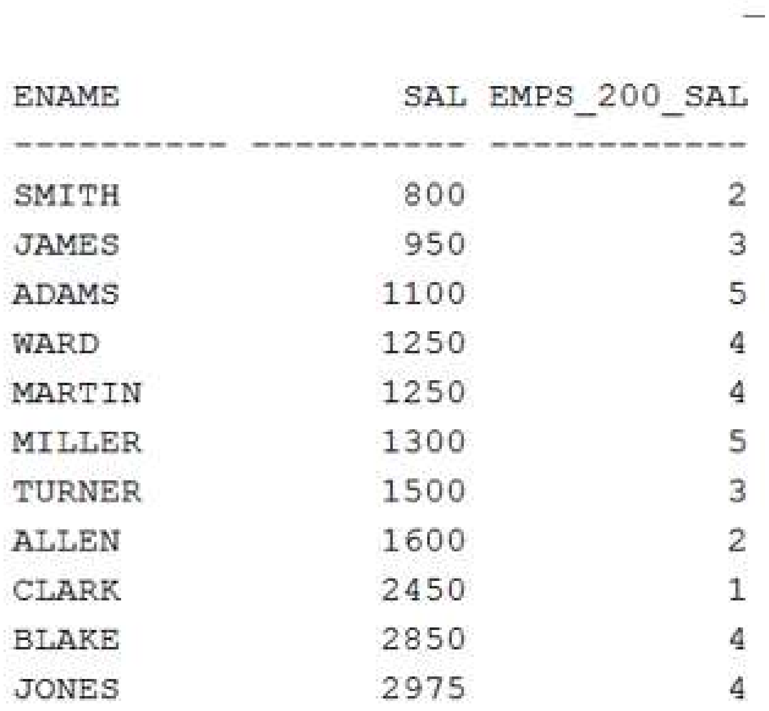
```
select "ENAME" , "SAL" ,
SUM("SAL") OVER(ORDER BY "SAL" ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW ) AS SUM1,
SUM("SAL") OVER(ORDER BY "SAL" RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW ) AS SUM2
from "General_schema".emp
```



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

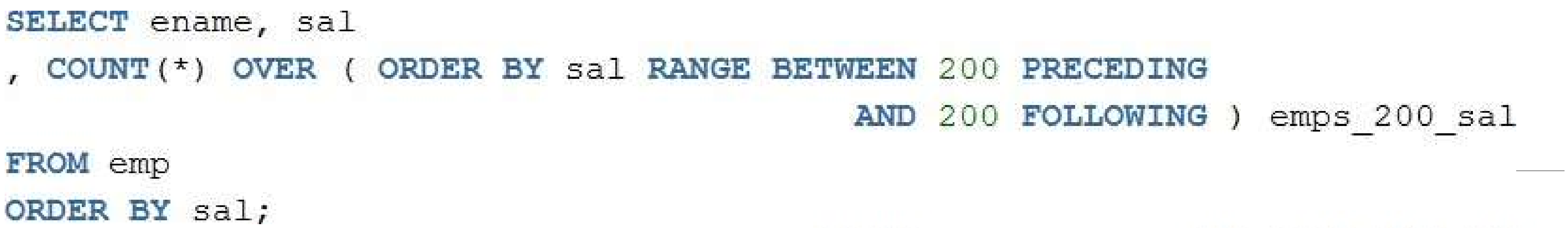
HANDS ON

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
```

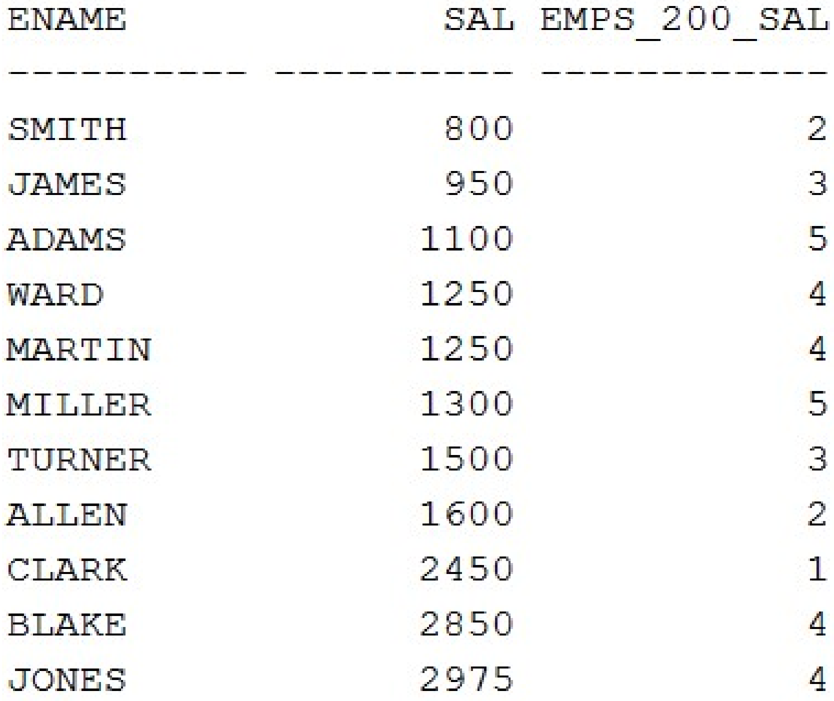
HANDS ON



```
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
```

Range of Intervals

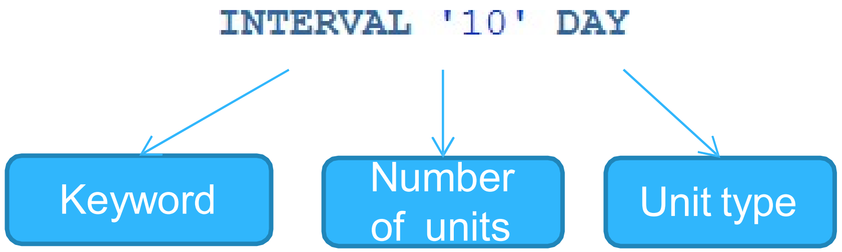
- How many people were hired within six months of this person?
- How many people were hired six months after this person?

```
SELECT empno, ename, hiredate
, COUNT(*) OVER ( ORDER BY hiredate
            RANGE BETWEEN INTERVAL '6' MONTH PRECEDING
            AND INTERVAL '6' MONTH FOLLOWING ) AS six_mo
, COUNT(*) OVER ( ORDER BY hiredate
            RANGE BETWEEN CURRENT ROW
            AND INTERVAL '6' MONTH FOLLOWING ) AS six_mo_after
FROM emp
ORDER BY hiredate;
```

```
     EMPNO ENAME      HIREDATE      SIX_MO SIX_MO_AFTER
---------- ---------- --------- ---------- ------------
      7369 SMITH      17-DEC-80          6            6
      7499 ALLEN      20-FEB-81          6            5
      7521 WARD       22-FEB-81          6            4
      7566 JONES      02-APR-81          8            5
      7698 BLAKE      01-MAY-81          8            4
      7782 CLARK      09-JUN-81         11            6
      7844 TURNER     08-SEP-81          9            6
      7654 MARTIN     28-SEP-81          9            5
      7839 KING       17-NOV-81          7            4
      7900 JAMES      03-DEC-81          7            3
      7902 FORD       03-DEC-81          7            3
```

INTERVAL Keyword

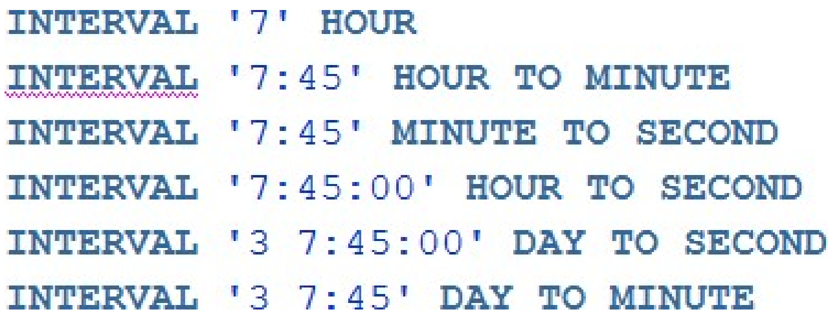
- An Interval is a period of time between two dates or two timestamps



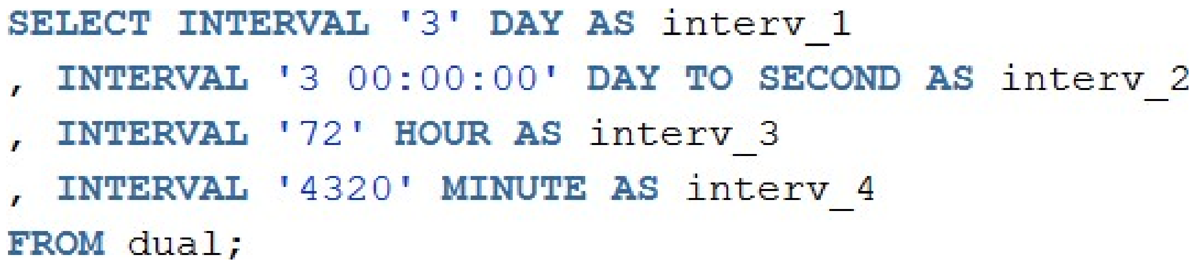
- The valid ranges for interval units are:

YEAR >> MONTH

DAY >> SECOND

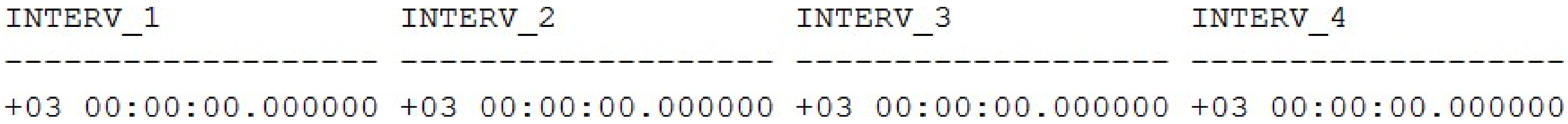


```
INTERVAL '7' HOUR
INTERVAL '7:45' HOUR TO MINUTE
INTERVAL '7:45' MINUTE TO SECOND
INTERVAL '7:45:00' HOUR TO SECOND
INTERVAL '3 7:45:00' DAY TO SECOND
INTERVAL '3 7:45' DAY TO MINUTE
```



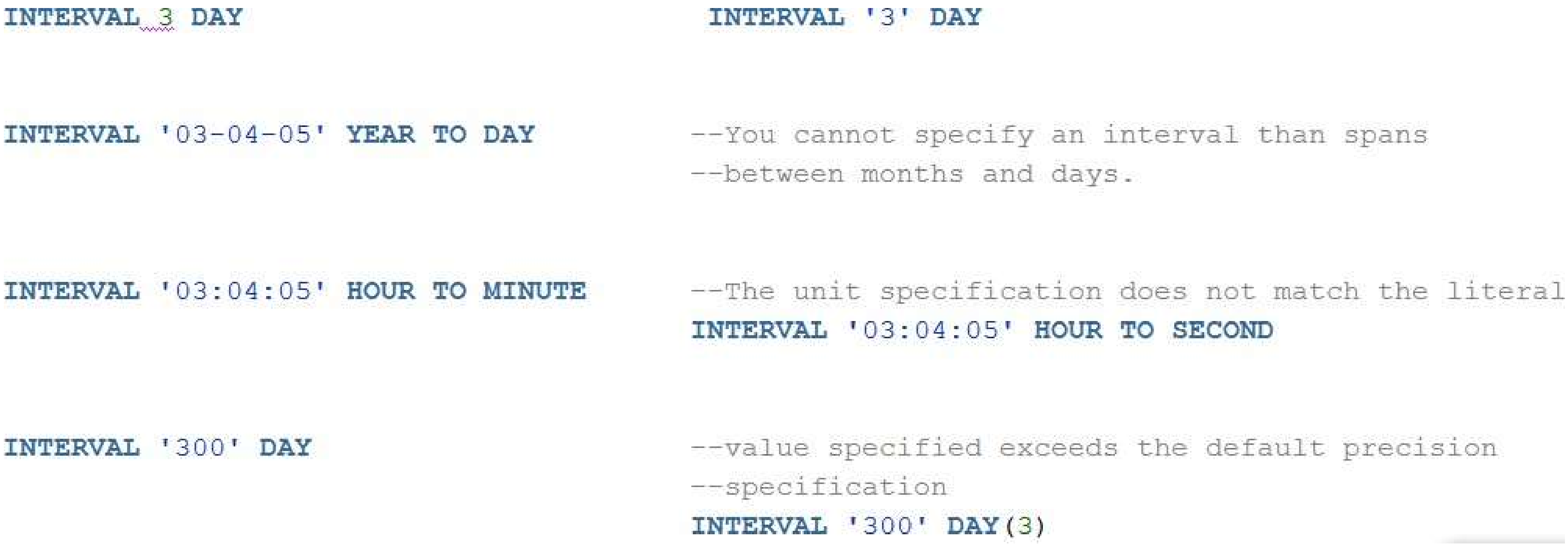
```
SELECT INTERVAL '3' DAY AS interv_1
, INTERVAL '3 00:00:00' DAY TO SECOND AS interv_2
, INTERVAL '72' HOUR AS interv_3
, INTERVAL '4320' MINUTE AS interv_4
FROM dual;
```

All of these express the interval three days

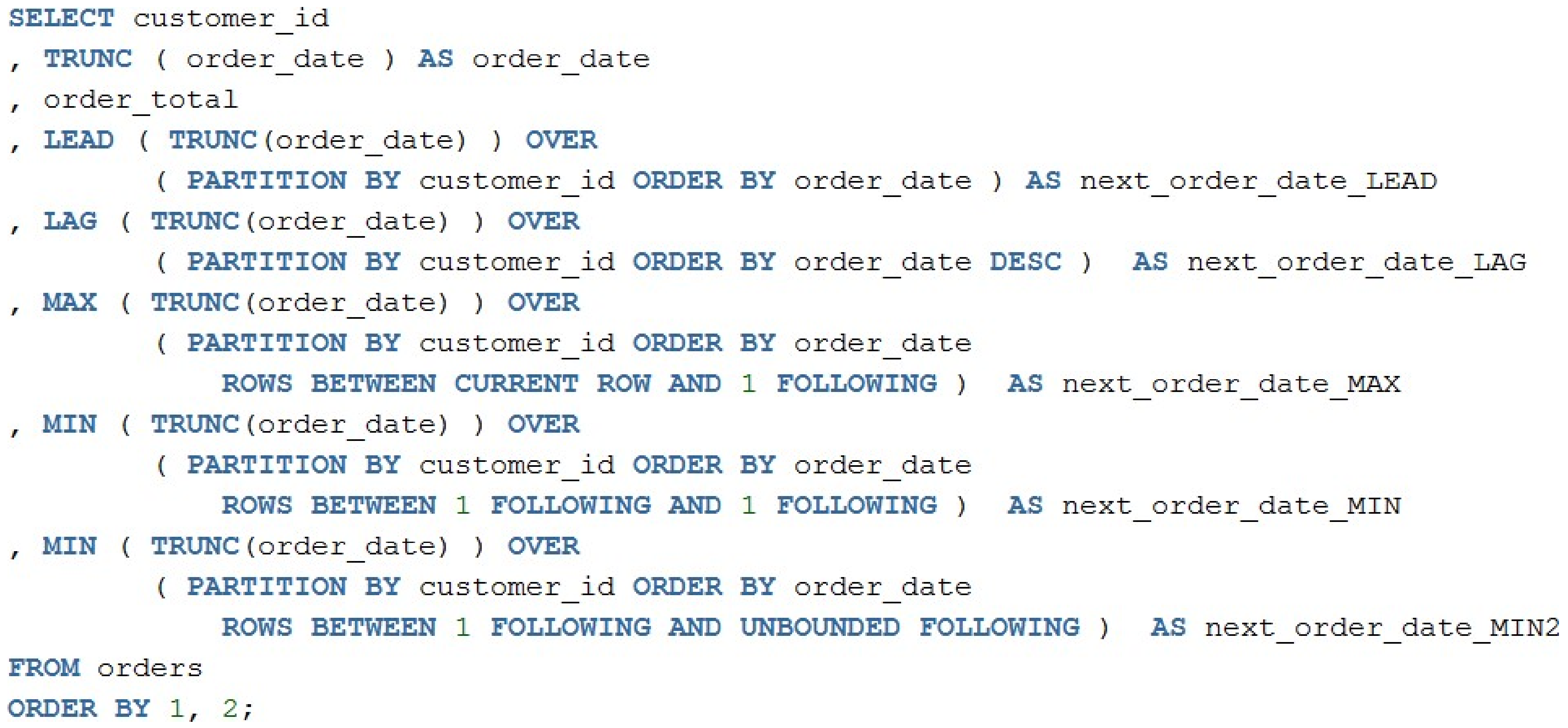


```
INTERV_1            INTERV_2            INTERV_3            INTERV_4
------------------- ------------------- ------------------- -------------------
+03 00:00:00.000000 +03 00:00:00.000000 +03 00:00:00.000000 +03 00:00:00.000000
```

INTERVAL Errors



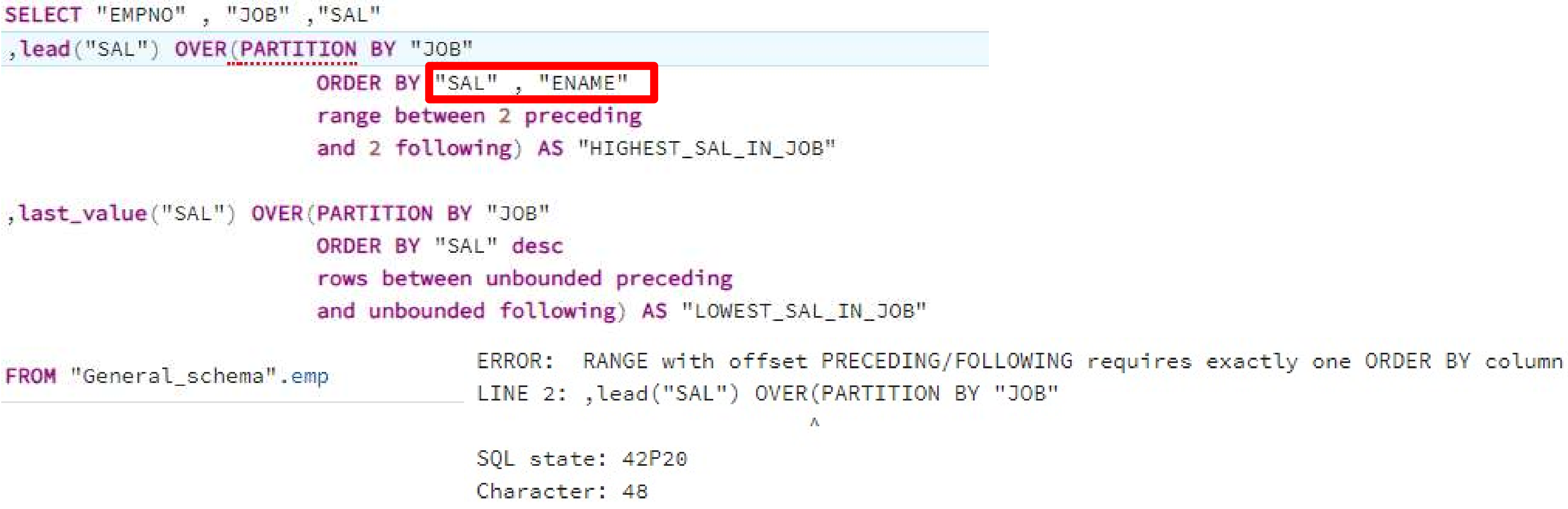
```
INTERVAL 3 DAY                              INTERVAL '3' DAY


INTERVAL '03-04-05' YEAR TO DAY            --You cannot specify an interval than spans
                                           --between months and days.


INTERVAL '03:04:05' HOUR TO MINUTE         --The unit specification does not match the literal
                                           INTERVAL '03:04:05' HOUR TO SECOND


INTERVAL '300' DAY                         --value specified exceeds the default precision
                                           --specification
                                           INTERVAL '300' DAY(3)
```



```sql
SELECT customer_id
, TRUNC ( order_date ) AS order_date
, order_total
, LEAD ( TRUNC(order_date) ) OVER
        ( PARTITION BY customer_id ORDER BY order_date ) AS next_order_date_LEAD
, LAG ( TRUNC(order_date) ) OVER
        ( PARTITION BY customer_id ORDER BY order_date DESC )  AS next_order_date_LAG
, MAX ( TRUNC(order_date) ) OVER
        ( PARTITION BY customer_id ORDER BY order_date
            ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING )  AS next_order_date_MAX
, MIN ( TRUNC(order_date) ) OVER
        ( PARTITION BY customer_id ORDER BY order_date
            ROWS BETWEEN 1 FOLLOWING AND 1 FOLLOWING )  AS next_order_date_MIN
, MIN ( TRUNC(order_date) ) OVER
        ( PARTITION BY customer_id ORDER BY order_date
            ROWS BETWEEN 1 FOLLOWING AND UNBOUNDED FOLLOWING )  AS next_order_date_MIN2
FROM orders
ORDER BY 1, 2;
```

Notes

- Only one sort key allowed when windowing with RANGE because range depends on the ORDER BY to derive the field

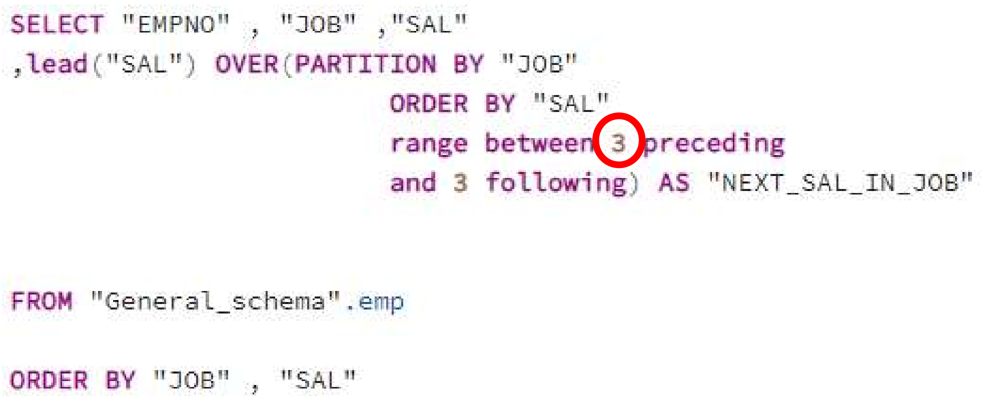


```
SELECT "EMPNO" , "JOB" ,"SAL"
,lead("SAL") OVER(PARTITION BY "JOB"
                        ORDER BY "SAL" , "ENAME"
                        range between 2 preceding
                        and 2 following) AS "HIGHEST_SAL_IN_JOB"

,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
```

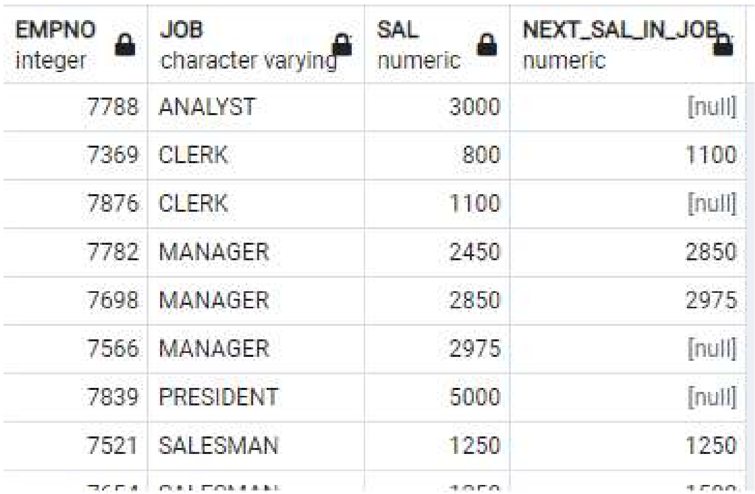
```
ERROR:  RANGE with offset PRECEDING/FOLLOWING requires exactly one ORDER BY column
LINE 2: ,lead("SAL") OVER(PARTITION BY "JOB"
                         ^
SQL state: 42P20
Character: 48
```

Notes

- You cannot use LAG or LEAD with a RANGE/ROWS window => the result will still be the same



```
SELECT "EMPNO" , "JOB" ,"SAL"
,lead("SAL") OVER(PARTITION BY "JOB"
                        ORDER BY "SAL"
                        range between 3 preceding
                        and 3 following) AS "NEXT_SAL_IN_JOB"


FROM "General_schema".emp

ORDER BY "JOB" , "SAL"
```



EMPNO integer	JOB character varying	SAL numeric	NEXT_SAL_IN_JOB numeric
7788	ANALYST	3000	[null]
7369	CLERK	800	1100
7876	CLERK	1100	[null]
7782	MANAGER	2450	2850
7698	MANAGER	2850	2975
7566	MANAGER	2975	[null]
7839	PRESIDENT	5000	[null]
7521	SALESMAN	1250	1250
7654	SALESMAN	1250	1500

Aggregate Analytical Functions

- MAX , MIN , AVG & SUM
- COUNTIF
- ANY_VALUE
- ARRAY_AGG
- STRING_AGG

More on aggregate analytical Functions

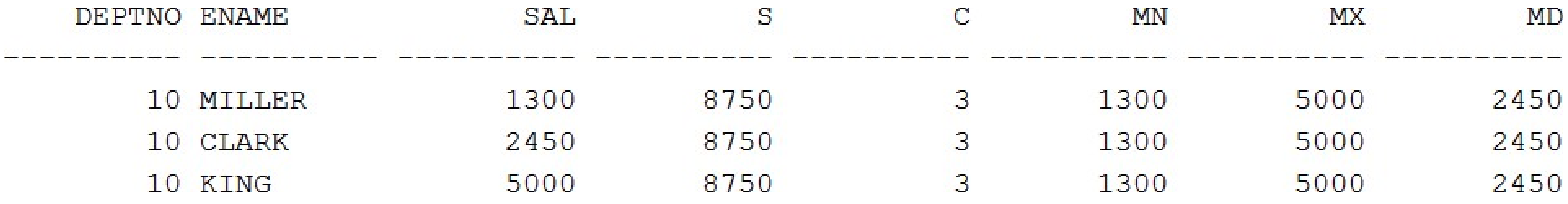
ANSI-SQL:

- AVG
- COUNT
- SUM
- MAX
- MIN
- MEDIAN

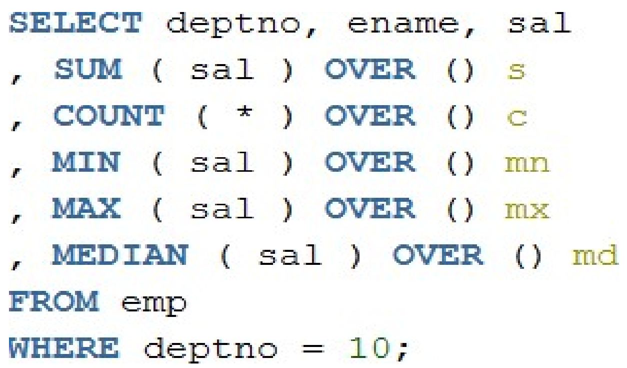
Non-ANSI-SQL:

- COUNTIF
- ANY_VALUE
- ARRAY_AGG
- STRING_AGG

Calculate the SALARY SUM , COUNT , MIN , MAX & MEDIAN for department “10”



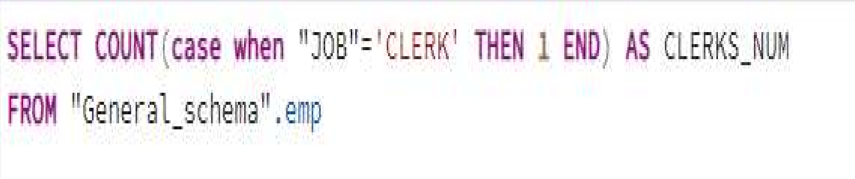
```
    DEPTNO ENAME             SAL          S          C         MN         MX         MD
---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
        10 MILLER           1300       8750          3       1300       5000       2450
        10 CLARK            2450       8750          3       1300       5000       2450
        10 KING             5000       8750          3       1300       5000       2450
```



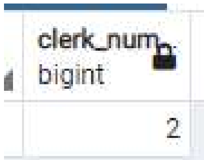
```
SELECT deptno, ename, sal
, SUM ( sal ) OVER () s
, COUNT ( * ) OVER () c
, MIN ( sal ) OVER () mn
, MAX ( sal ) OVER () mx
, MEDIAN ( sal ) OVER () md
FROM emp
WHERE deptno = 10;
```

COUNTIF()

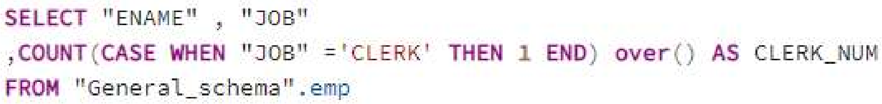
- COUNTIF() is an extension of COUNT where it returns the number of rows that satisfy the condition.
- COUNTIF() is not ANSI instead you can use COUNT(CASE WHEN <condition> THEN 1 END)



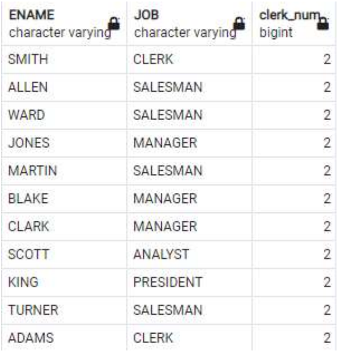
```
SELECT COUNT(case when "JOB"='CLERK' THEN 1 END) AS CLERKS_NUM
FROM "General_schema".emp
```



clerk_num bigint
2



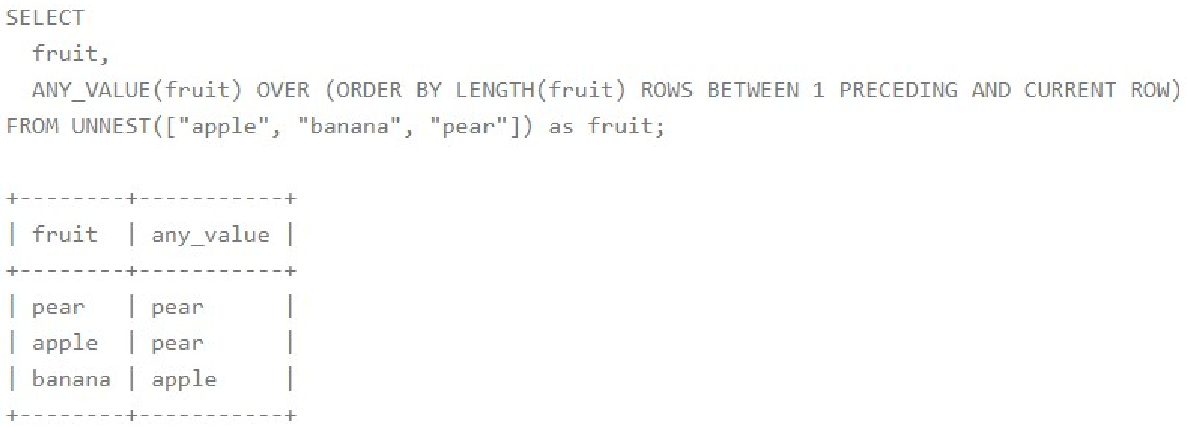
```
SELECT "ENAME" , "JOB"
,COUNT(CASE WHEN "JOB" ='CLERK' THEN 1 END) over() AS CLERK_NUM
FROM "General_schema".emp
```



ENAME character varying	JOB character varying	clerk_num bigint
SMITH	CLERK	2
ALLEN	SALESMAN	2
WARD	SALESMAN	2
JONES	MANAGER	2
MARTIN	SALESMAN	2
BLAKE	MANAGER	2
CLARK	MANAGER	2
SCOTT	ANALYST	2
KING	PRESIDENT	2
TURNER	SALESMAN	2
ADAMS	CLERK	2

ANY_VALUE()

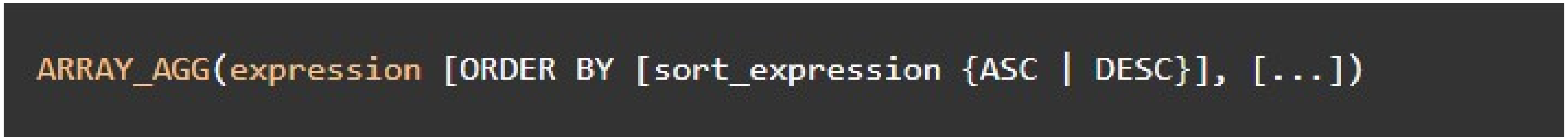
- Returns any value from the input or NULL if there are zero input rows. The value returned is non-deterministic, which means you might receive a different result each time you use this function.
- You can use it if you want any value from each partition



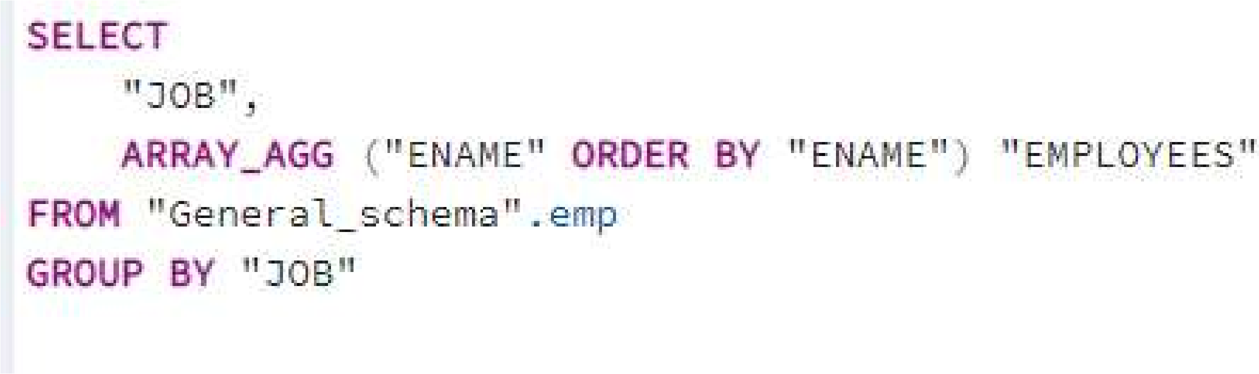
```
SELECT
  fruit,
  ANY_VALUE(fruit) OVER (ORDER BY LENGTH(fruit) ROWS BETWEEN 1 PRECEDING AND CURRENT ROW)
FROM UNNEST(["apple", "banana", "pear"]) as fruit;

+--------+-----------+
| fruit  | any_value |
+--------+-----------+
| pear   | pear      |
| apple  | pear      |
| banana | apple     |
+--------+-----------+
```

ARRAY_AGG()

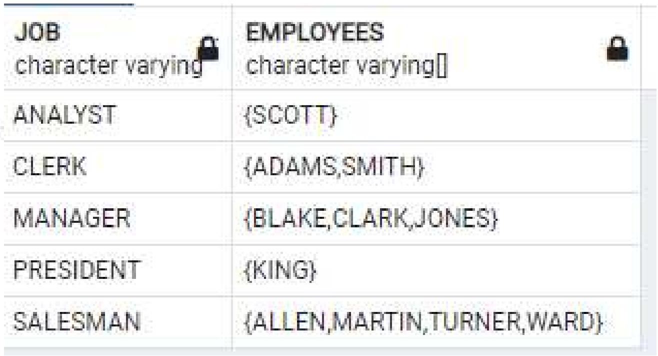
- Function that accepts a set of values and returns an **array** in which each value in the set is assigned to an element of the array.



```
ARRAY_AGG(expression [ORDER BY [sort_expression {ASC | DESC}], [...])
```



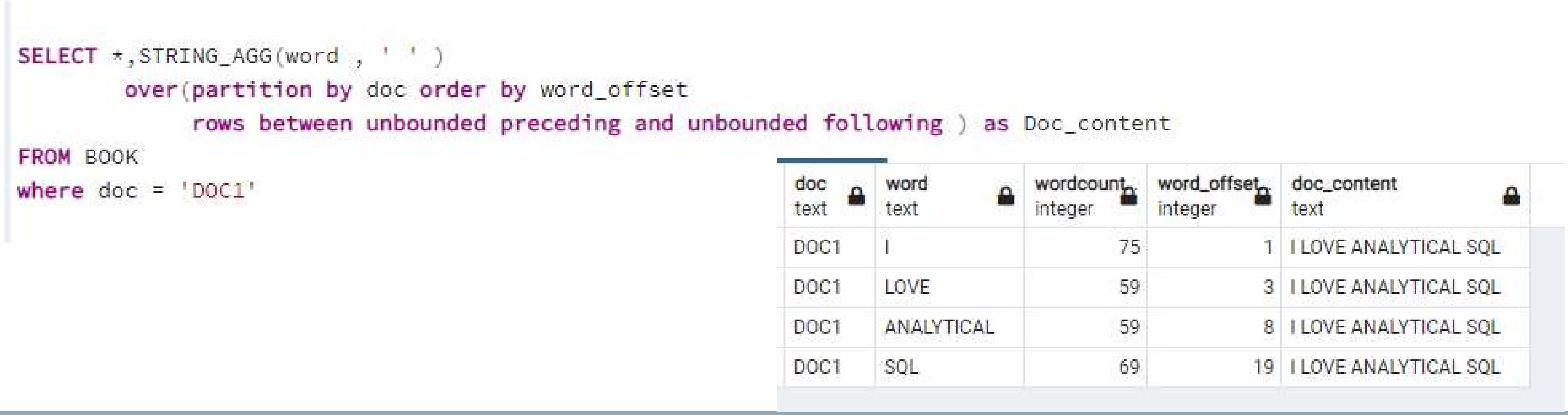
```
SELECT
    "JOB",
    ARRAY_AGG ("ENAME" ORDER BY "ENAME") "EMPLOYEES"
FROM "General_schema".emp
GROUP BY "JOB"
```



JOB character varying	EMPLOYEES character varying[]
ANALYST	{SCOTT}
CLERK	{ADAMS,SMITH}
MANAGER	{BLAKE,CLARK,JONES}
PRESIDENT	{KING}
SALESMAN	{ALLEN,MARTIN,TURNER,WARD}

STRING_AGG()

- STRING_AGG is an aggregate function that takes all expressions from rows and concatenates them into a single string
- STRING_AGG (expression, separator) [<order_clause>]
- The separator is not added at the end of string.

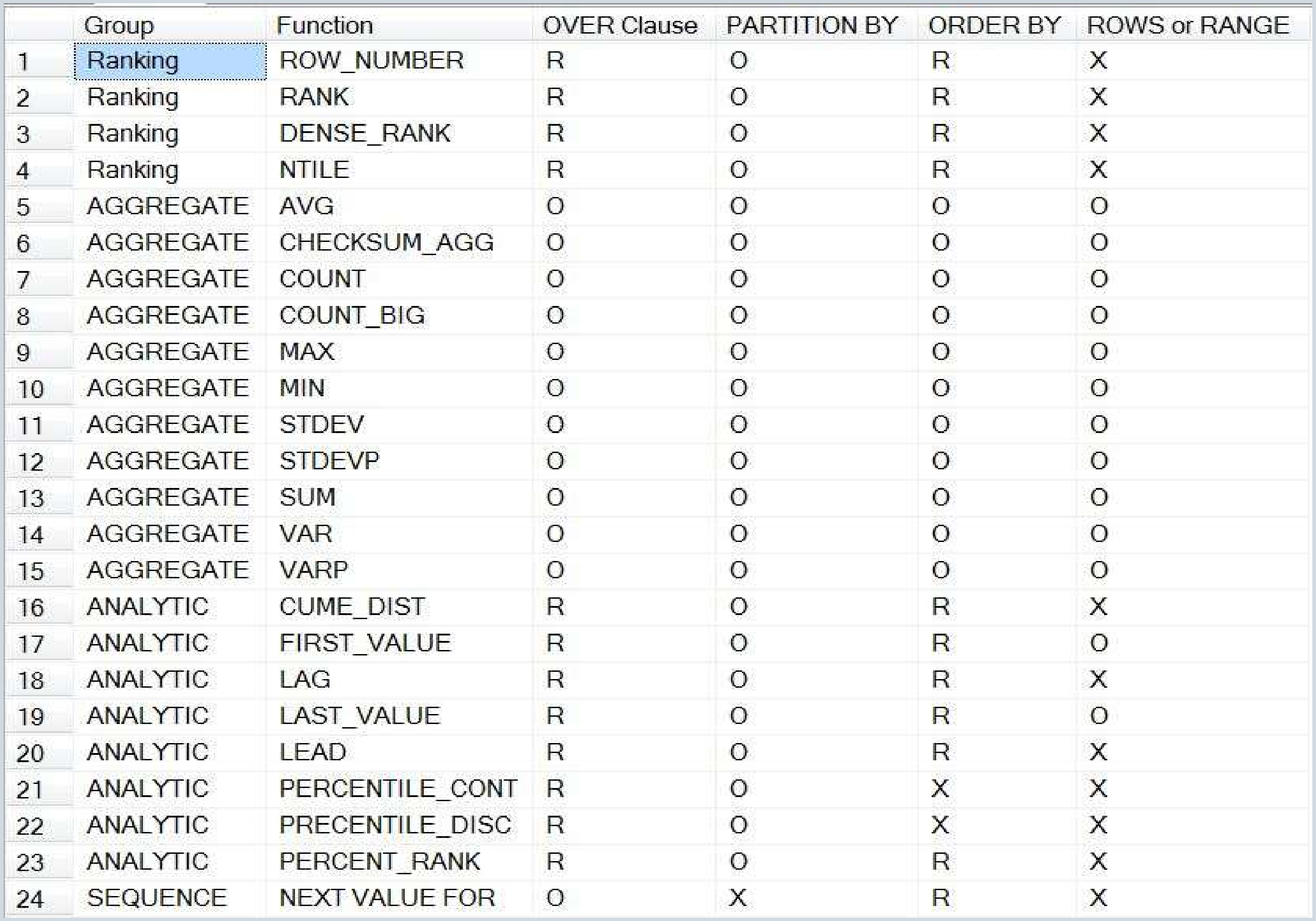


```
SELECT *,STRING_AGG(word , ' ' )
        over(partition by doc order by word_offset
             rows between unbounded preceding and unbounded following ) as Doc_content
FROM BOOK
where doc = 'DOC1'
```

doc text	word text	wordcount integer	word_offset integer	doc_content text
DOC1	I	75	1	I LOVE ANALYTICAL SQL
DOC1	LOVE	59	3	I LOVE ANALYTICAL SQL
DOC1	ANALYTICAL	59	8	I LOVE ANALYTICAL SQL
DOC1	SQL	69	19	I LOVE ANALYTICAL SQL

chart of the various functions that can use the OVER clause, as well as which portions of the clause are **(allowed / required / optional)**

- **R-Required**
- **O-Optional**
- **X-Not Allowed**



	Group	Function	OVER Clause	PARTITION BY	ORDER BY	ROWS or RANGE
1	Ranking	ROW_NUMBER	R	O	R	X
2	Ranking	RANK	R	O	R	X
3	Ranking	DENSE_RANK	R	O	R	X
4	Ranking	NTILE	R	O	R	X
5	AGGREGATE	AVG	O	O	O	O
6	AGGREGATE	CHECKSUM_AGG	O	O	O	O
7	AGGREGATE	COUNT	O	O	O	O
8	AGGREGATE	COUNT_BIG	O	O	O	O
9	AGGREGATE	MAX	O	O	O	O
10	AGGREGATE	MIN	O	O	O	O
11	AGGREGATE	STDEV	O	O	O	O
12	AGGREGATE	STDEVP	O	O	O	O
13	AGGREGATE	SUM	O	O	O	O
14	AGGREGATE	VAR	O	O	O	O
15	AGGREGATE	VARP	O	O	O	O
16	ANALYTIC	CUME_DIST	R	O	R	X
17	ANALYTIC	FIRST_VALUE	R	O	R	O
18	ANALYTIC	LAG	R	O	R	X
19	ANALYTIC	LAST_VALUE	R	O	R	O
20	ANALYTIC	LEAD	R	O	R	X
21	ANALYTIC	PERCENTILE_CONT	R	O	X	X
22	ANALYTIC	PRECENTILE_DISC	R	O	X	X
23	ANALYTIC	PERCENT_RANK	R	O	R	X
24	SEQUENCE	NEXT VALUE FOR	O	X	R	X