MTN.BI.03

Author: Volha Kutsevol
Lead Software Engineer
Volha_Kutsevol@epam.com

SQL for Analysis and Reporting

Agenda

1. Overview of SQL for Analysis and Reporting
2. Rankings and percentiles
3. Reporting
4. Lag/lead analysis

RANK and DENSE_RANK Functions

The **RANK** and **DENSE_RANK** functions allow you to rank items in a group, for example, finding the top three products sold in California last year. There are two functions that perform ranking, as shown by the following syntax:

```
RANK ( ) OVER ( [query_partition_clause] order_by_clause )

DENSE_RANK ( ) OVER ( [query_partition_clause] order_by_clause )
```

The difference between **RANK** and **DENSE_RANK** is that DENSE_RANK leaves no gaps in ranking sequence when there are ties.

Ranking Order

```
select CHANNEL_DESC,
       TO_CHAR(SUM(AMOUNT_SOLD), '9,999,999,999') SALES$,
       RANK() over (order by SUM(AMOUNT_SOLD)) as DEFAULT_RANK,
       RANK() over (order by SUM(AMOUNT_SOLD) desc nulls last) as CUSTOM_RANK
from SALES, PRODUCTS, CUSTOMERS, TIMES, CHANNELS, COUNTRIES
where SALES.PROD_ID=PRODUCTS.PROD_ID and SALES.CUST_ID=CUSTOMERS.CUST_ID
      and CUSTOMERS.COUNTRY_ID = COUNTRIES.COUNTRY_ID and SALES.TIME_ID=TIMES.TIME_ID
      and SALES.CHANNEL_ID=CHANNELS.CHANNEL_ID
      and TIMES.CALENDAR_MONTH_DESC in ('2000-09', '2000-10')
     and COUNTRY_ISO_CODE='US'
group by CHANNEL_DESC
```

CHANNEL_DESC	SALES$	DEFAULT_RANK	CUSTOM_RANK
Direct Sales	1,320,497	3	1
Partners	800,871	2	2
Internet	261,278	1	3

Ranking on Multiple Expressions

```
select CHANNEL_DESC,
       CALENDAR_MONTH_DESC,
       TO_CHAR(TRUNC(SUM(AMOUNT_SOLD),-5), '9,999,999,999') SALES$,
       TO_CHAR(SUM(QUANTITY_SOLD), '9,999,999,999') SALES_COUNT,
       RANK() over (order by TRUNC(SUM(AMOUNT_SOLD)) desc, SUM(QUANTITY_SOLD) desc) as COL_RANK
from SALES, PRODUCTS, CUSTOMERS, TIMES, CHANNELS
where SALES.PROD_ID=PRODUCTS.PROD_ID and
      SALES.CUST_ID=CUSTOMERS.CUST_ID and
      SALES.TIME_ID=TIMES.TIME_ID and
      SALES.CHANNEL_ID=CHANNELS.CHANNEL_ID and
      TIMES.CALENDAR_MONTH_DESC in ('2000-09', '2000-10') and
     CHANNELS.CHANNEL_DESC <> 'Tele Sales'
group by CHANNEL_DESC, CALENDAR_MONTH_DESC
```

CHANNEL_DESC	CALENDAR_MONTH_DESC	SALES$	SALES_COUNT	COL_RANK
Direct Sales	2000-10	1,200,000	12,584	1
Direct Sales	2000-09	1,200,000	11,995	2
Partners	2000-10	600,000	7,508	3
Partners	2000-09	600,000	6,165	4
Internet	2000-10	200,000	1,450	5
Internet	2000-09	200,000	1,887	6

RANK and DENSE_RANK Difference

```
SELECT channel_desc, calendar_month_desc,
       to_char(trunc(sum(amount_sold),-5), '9,999,999,999') sales$,
       rank() OVER (ORDER BY trunc(sum(amount_sold),-5) DESC) AS rank,
       DENSE_RANK() OVER (ORDER BY trunc(sum(amount_sold),-5) DESC) AS DENSE_RANK
FROM sales, products, customers, times, channels
WHERE sales.prod_id=products.prod_id
      AND sales.cust_id=customers.cust_id
      AND sales.time_id=times.time_id AND sales.channel_id=channels.channel_id
      AND times.calendar_month_desc IN ('2000-09', '2000-10')
      AND channels.channel_desc<>'Tele Sales'GROUP BY channel_desc, calendar_month_desc
```

CHANNEL_DESC	CALENDAR_MONTH_DESC	SALES$	RANK	DENSE_RANK
Direct Sales	2000-10	1,200,000	1	1
Direct Sales	2000-09	1,200,000	1	1
Partners	2000-10	600,000	3	2
Partners	2000-09	600,000	3	2
Internet	2000-10	200,000	5	3
Internet	2000-09	200,000	5	3

Per Group Ranking

The RANK function can be made to operate within groups, that is, the rank gets reset whenever the group changes. This is accomplished with the PARTITION BY clause. The group expressions in the **PARTITION BY** subclause divide the data set into groups within which RANK operates.

```
select CHANNEL_DESC,
       CALENDAR_MONTH_DESC,
       TO_CHAR(SUM(AMOUNT_SOLD), '9,999,999,999') SALES$,
       RANK() over (partition by CHANNEL_DESC order by SUM(AMOUNT_SOLD) desc) as RANK_BY_CHANNEL
from SALES, PRODUCTS, CUSTOMERS, TIMES, CHANNELS
where SALES.PROD_ID=PRODUCTS.PROD_ID and SALES.CUST_ID=CUSTOMERS.CUST_ID
      and SALES.TIME_ID=TIMES.TIME_ID and SALES.CHANNEL_ID=CHANNELS.CHANNEL_ID
      and TIMES.CALENDAR_MONTH_DESC in ('2000-08', '2000-09', '2000-10', '2000-11')
      and CHANNELS.CHANNEL_DESC in ('Direct Sales', 'Internet')
group by CHANNEL_DESC, CALENDAR_MONTH_DESC
```

CHANNEL_DESC	CALENDAR_MONTH_DESC	SALES$	RANK_BY_CHANNEL
Direct Sales	2000-08	1,236,104	1
Direct Sales	2000-10	1,225,584	2
Direct Sales	2000-09	1,217,808	3
Direct Sales	2000-11	1,115,239	4
Internet	2000-11	284,742	1
Internet	2000-10	239,236	2
Internet	2000-09	228,241	3
Internet	2000-08	215,107	4

Per Group Ranking

```
SELECT channel_desc,
       CALENDAR_MONTH_DESC,
       TO_CHAR(SUM(AMOUNT_SOLD), '9,999,999,999') SALES$,
       RANK() OVER (PARTITION BY calendar_month_desc ORDER BY SUM(amount_sold) DESC) AS RANK_WITHIN_MONTH,
       RANK() OVER (PARTITION BY channel_desc ORDER BY SUM(amount_sold) DESC) AS RANK_WITHIN_CHANNEL
FROM sales, products, customers, times, channels, countries
where SALES.PROD_ID=PRODUCTS.PROD_ID and SALES.CUST_ID=CUSTOMERS.CUST_ID
      and CUSTOMERS.COUNTRY_ID = COUNTRIES.COUNTRY_ID and SALES.TIME_ID=TIMES.TIME_ID
      and SALES.CHANNEL_ID=CHANNELS.CHANNEL_ID
      and TIMES.CALENDAR_MONTH_DESC in ('2000-08', '2000-09', '2000-10', '2000-11')
      and CHANNELS.CHANNEL_DESC in ('Direct Sales', 'Internet')
group by CHANNEL_DESC, CALENDAR_MONTH_DESC
ORDER BY 1,4,5
```

CHANNEL_DESC	CALENDAR_MONTH_DESC	SALES$	RANK_WITHIN_MONTH	RANK_WITHIN_CHANNEL
Direct Sales	2000-08	1,236,104	1	1
Direct Sales	2000-10	1,225,584	1	2
Direct Sales	2000-09	1,217,808	1	3
Direct Sales	2000-11	1,115,239	1	4
Internet	2000-11	284,742	2	1
Internet	2000-10	239,236	2	2
Internet	2000-09	228,241	2	3
Internet	2000-08	215,107	2	4

Per Cube and Rollup Group Ranking

Analytic functions, RANK for example, can be reset based on the groupings provided by a **CUBE, ROLLUP**, or **GROUPING SETS** operator. It is useful to assign ranks to the groups created by CUBE, ROLLUP, and GROUPING SETS queries.

```
select CHANNEL_DESC,
       COUNTRY_ISO_CODE,
       TO_CHAR(SUM(AMOUNT_SOLD), '9,999,999,999') SALES$,
       GROUPING_ID(CHANNEL_DESC, COUNTRY_ISO_CODE) GR_CHANNEL,
       RANK() over (partition by GROUPING_ID(CHANNEL_DESC, COUNTRY_ISO_CODE) order by SUM(AMOUNT_SOLD) desc)
           as RANK_PER_GROUP,
       dense_rank() over (partition by GROUPING_ID(CHANNEL_DESC, COUNTRY_ISO_CODE) order by SUM(AMOUNT_SOLD) desc)
           as DENSE_RANK_PER_GROUP
from SALES, CUSTOMERS, TIMES, CHANNELS, COUNTRIES
where SALES.TIME_ID=TIMES.TIME_ID
      and SALES.CUST_ID=CUSTOMERS.CUST_ID
      and SALES.CHANNEL_ID = CHANNELS.CHANNEL_ID
      and CHANNELS.CHANNEL_DESC in ('Direct Sales', 'Internet')
      and TIMES.CALENDAR_MONTH_DESC='2000-09'
      and COUNTRY_ISO_CODE in ('GB', 'US', 'JP')
group by cube(CHANNEL_DESC, COUNTRY_ISO_CODE)
```

CHANNEL_DESC	COUNTRY_ISO_CODE	SALES$	GR_CHANNEL	RANK_PER_GROUP	DENSE_RANK_PER_GROUP
Direct Sales	JP	1,217,808	0	1	1
Direct Sales	GB	1,217,808	0	1	1
Direct Sales	US	1,217,808	0	1	1
Internet	GB	228,241	0	4	2
Internet	US	228,241	0	4	2
Internet	JP	228,241	0	4	2
Direct Sales		3,653,423	1	1	1
Internet		684,724	1	2	2
	GB	1,446,049	2	1	1
	JP	1,446,049	2	1	1
	US	1,446,049	2	1	1
		4,338,147	3	1	1

Treatment of NULLs

NULLs are treated like normal values. Also, for rank computation, a NULL value is assumed to be equal to another NULL value. Depending on the ASC | DESC options provided for measures and the **NULLS FIRST | NULLS LAST** clause, NULLs will either sort low or high and hence, are given ranks appropriately.

```
SELECT  times.time_id TIME,  sold,
  rank() OVER (ORDER BY (sold) DESC NULLS LAST) AS nlast_desc,
  rank() OVER (ORDER BY (sold) DESC NULLS FIRST) AS nfirst_desc,
  rank() OVER (ORDER BY (sold) ASC NULLS FIRST) AS nfirst,
  rank() OVER (ORDER BY (sold) ASC NULLS LAST) AS nlast
FROM
  (
   SELECT time_id,
          sum(sales.amount_sold)  sold
   FROM sales, products, customers, countries
   WHERE sales.prod_id=products.prod_id
         AND customers.country_id = countries.country_id
         AND sales.cust_id=customers.cust_id
         AND prod_name IN ('Envoy Ambassador', 'Mouse Pad')
   GROUP BY time_id) v, times
WHERE v.time_id (+) = times.time_id
      AND calendar_year=1999
      AND calendar_month_number=1
ORDER BY sold  DESC NULLS LAST
```

TIME	SOLD	NLAST_DESC	NFIRST_DESC	NFIRST	NLAST
25-Jan-99	3097.32	1	18	31	14
17-Jan-99	1791.77	2	19	30	13
30-Jan-99	127.69	3	20	29	12
28-Jan-99	120.34	4	21	28	11
23-Jan-99	86.12	5	22	27	10
20-Jan-99	79.07	6	23	26	9
13-Jan-99	56.1	7	24	25	8
7-Jan-99	42.97	8	25	24	7
8-Jan-99	33.81	9	26	23	6
2-Jan-99	22.76	10	27	21	4
10-Jan-99	22.76	10	27	21	4
26-Jan-99	19.84	12	29	20	3
16-Jan-99	11.27	13	30	19	2
14-Jan-99	9.52	14	31	18	1
9-Jan-99		15	1	1	15
11-Jan-99		15	1	1	15
12-Jan-99		15	1	1	15
3-Jan-99		15	1	1	15
15-Jan-99		15	1	1	15
31-Jan-99		15	1	1	15
4-Jan-99		15	1	1	15
19-Jan-99		15	1	1	15
5-Jan-99		15	1	1	15
29-Jan-99		15	1	1	15
27-Jan-99		15	1	1	15
1-Jan-99		15	1	1	15
21-Jan-99		15	1	1	15
18-Jan-99		15	1	1	15
24-Jan-99		15	1	1	15

CUME_DIST Function

The CUME_DIST function (defined as the inverse of percentile in some statistical books) computes the position of a specified value relative to a set of values. The order can be ascending or descending. Ascending is the default. The range of values for CUME_DIST is from greater than 0 to 1.

CUME_DIST(x) = number of values in S coming before and including x in the specified order/ N

```
CUME_DIST ( ) OVER ( [query_partition_clause] order_by_clause )
```

CUME_DIST Function

```
SELECT calendar_month_desc AS MONTH, channel_desc,
     TO_CHAR(SUM(amount_sold) , '9,999,999,999') SALES$,
     ROUND(CUME_DIST() OVER (PARTITION BY calendar_month_desc ORDER BY SUM(amount_sold) ), 2) AS CUME_DIST_BY_CHANNEL
FROM sales, products, customers, times, channels
WHERE sales.prod_id=products.prod_id
      AND sales.cust_id=customers.cust_id
      AND sales.time_id=times.time_id
      AND sales.channel_id=channels.channel_id
     AND times.calendar_month_desc IN ('2000-09', '2000-07','2000-08')
GROUP BY calendar_month_desc, channel_desc
```

MONTH	CHANNEL_DESC	SALES$	CUME_DIST_BY_CHANNEL
2000-07	Internet	140,423	0.33
2000-07	Partners	611,064	0.67
2000-07	Direct Sales	1,145,275	1
2000-08	Internet	215,107	0.33
2000-08	Partners	661,045	0.67
2000-08	Direct Sales	1,236,104	1
2000-09	Internet	228,241	0.33
2000-09	Partners	666,172	0.67
2000-09	Direct Sales	1,217,808	1

PERCENT_RANK Function

PERCENT_RANK is similar to CUME_DIST, but it uses rank values rather than row counts in its numerator. Therefore, it returns the percent rank of a value relative to a group of values. The function is available in many popular spreadsheets.

PERCENT_RANK =(rank of row in its partition - 1) /
(number of rows in the partition - 1)

```
PERCENT_RANK () OVER ([query_partition_clause] order_by_clause)
```

PERCENT_RANK Function

```
SELECT calendar_month_desc AS MONTH, channel_desc,
     TO_CHAR(SUM(amount_sold) , '9,999,999,999') SALES$,
     ROUND(PERCENT_RANK() OVER (PARTITION BY calendar_month_desc ORDER BY SUM(amount_sold) ), 2) AS CUME_DIST_BY_CHANNEL
FROM sales, products, customers, times, channels
WHERE sales.prod_id=products.prod_id
      AND sales.cust_id=customers.cust_id
      AND sales.time_id=times.time_id
      AND sales.channel_id=channels.channel_id
     AND times.calendar_month_desc IN ('2000-09', '2000-07','2000-08')
GROUP BY calendar_month_desc, channel_desc
```

MONTH	CHANNEL_DESC	SALES$	CUME_DIST_BY_CHANNEL
2000-07	Internet	140,423	0
2000-07	Partners	611,064	0.5
2000-07	Direct Sales	1,145,275	1
2000-08	Internet	215,107	0
2000-08	Partners	661,045	0.5
2000-08	Direct Sales	1,236,104	1
2000-09	Internet	228,241	0
2000-09	Partners	666,172	0.5
2000-09	Direct Sales	1,217,808	1

NTILE Function

NTILE allows easy calculation of tertiles, quartiles, deciles and other common summary statistics. This function divides an ordered partition into a specified number of groups called buckets and assigns a bucket number to each row in the partition. **NTILE** is a very useful calculation because it lets users divide a data set into fourths, thirds, and other groupings.

```
NTILE (expr) OVER ([query_partition_clause] order_by_clause)
```

NTILE Function

```
select CALENDAR_MONTH_DESC as month ,
       TO_CHAR(SUM(AMOUNT_SOLD), '9,999,999,999') SALES$,
       NTILE(4) over (order by SUM(AMOUNT_SOLD)) as TILE4
from SALES, PRODUCTS, CUSTOMERS, TIMES, CHANNELS
where SALES.PROD_ID=PRODUCTS.PROD_ID and SALES.CUST_ID=CUSTOMERS.CUST_ID
      and SALES.TIME_ID=TIMES.TIME_ID and SALES.CHANNEL_ID=CHANNELS.CHANNEL_ID
      and TIMES.CALENDAR_YEAR=2000 and PROD_CATEGORY= 'Electronics'
group by CALENDAR_MONTH_DESC
```

MONTH	SALES$	TILE4
2000-02	242,416	1
2000-01	257,286	1
2000-03	280,011	1
2000-06	315,951	2
2000-05	316,824	2
2000-04	318,106	2
2000-07	433,824	3
2000-08	477,833	3
2000-12	553,534	3
2000-10	652,225	4
2000-11	661,147	4
2000-09	691,449	4

Reporting

After a query has been processed, aggregate values like the number of resulting rows or an average value in a column can be easily computed within a partition and made available to other reporting functions. Reporting aggregate functions return the same aggregate value for every row in a partition. Their behavior with respect to NULLs is the same as the SQL aggregate functions.

```
{SUM | AVG | MAX | MIN | COUNT | STDDEV | VARIANCE  ... }
    ([ALL | DISTINCT] {value expression1 [,...] | *})
    OVER ([PARTITION BY value expression2[,...]])
```

Reporting

"For each product category, find the region in which it had maximum sales"

```
select SUBSTR(P.PROD_CATEGORY,1,8) as PROD_CATEGORY, CO.COUNTRY_REGION,
            SUM(AMOUNT_SOLD) as SALES,
             max(SUM(AMOUNT_SOLD)) over (partition by PROD_CATEGORY) as MAX_REG_SALES
      from SALES S, CUSTOMERS C, COUNTRIES CO, PRODUCTS P
      where S.CUST_ID=C.CUST_ID
            and C.COUNTRY_ID=CO.COUNTRY_ID
            and S.PROD_ID =P.PROD_ID and S.TIME_ID = TO_DATE('11-OCT-2001')
      group by P.PROD_CATEGORY, CO.COUNTRY_REGION
```

PROD_CATEGORY	COUNTRY_REGION	SALES	MAX_REG_SALES
Electron	Americas	581.92	581.92
Hardware	Americas	925.93	925.93
Peripher	Americas	3084.48	4290.38
Peripher	Asia	2616.51	4290.38
Peripher	Europe	4290.38	4290.38
Peripher	Oceania	940.43	4290.38
Software	Americas	4445.7	4445.7
Software	Asia	1408.19	4445.7
Software	Europe	3288.83	4445.7
Software	Oceania	890.25	4445.7

Reporting

```
select PROD_CATEGORY, COUNTRY_REGION, SALES
from
(
select SUBSTR(P.PROD_CATEGORY,1,8) as PROD_CATEGORY, CO.COUNTRY_REGION,
            SUM(AMOUNT_SOLD) as SALES,
             max(SUM(AMOUNT_SOLD)) over (partition by PROD_CATEGORY) as MAX_REG SALES
      from SALES S, CUSTOMERS C, COUNTRIES CO, PRODUCTS P
      where S.CUST_ID=C.CUST_ID
            and C.COUNTRY_ID=CO.COUNTRY_ID
            and S.PROD_ID =P.PROD_ID and S.TIME_ID = TO_DATE('11-OCT-2001')
      group by P.PROD_CATEGORY, CO.COUNTRY_REGION
)
where SALES = MAX_REG_SALES
```

PROD_CATEGORY	COUNTRY_REGION	SALES
Electron	Americas	581.92
Hardware	Americas	925.93
Peripher	Europe	4290.38
Software	Americas	4445.7

Reporting

The following is a query which finds the 5 top-selling products for each product subcategory that contributes more than 20% of the sales within its product category

```
select P.PROD_CATEGORY, P.PROD_SUBCATEGORY, P.PROD_ID,
       SUM(AMOUNT_SOLD) as SALES,
       SUM(SUM(AMOUNT_SOLD)) over (partition by P.PROD_CATEGORY) as CAT_SALES,
       SUM(SUM(AMOUNT_SOLD)) over (partition by P.PROD_SUBCATEGORY) as SUBCAT_SALES,
       RANK() over  (partition by P.PROD_SUBCATEGORY  order by SUM(AMOUNT_SOLD) desc ) as RANK_IN_LINE
     from SALES S, CUSTOMERS C, COUNTRIES CO, PRODUCTS P
     where S.CUST_ID=C.CUST_ID
           and C.COUNTRY_ID=CO.COUNTRY_ID and  S.PROD_ID=P.PROD_ID
           and S.TIME_ID=TO_DATE('11-OCT-2000')
     group by P.PROD_CATEGORY, P.PROD_SUBCATEGORY, P.PROD_ID
ORDER BY PROD_CATEGORY, PROD_SUBCATEGORY
```

PROD_CATEGORY	PROD_SUBCATEGORY	PROD_ID	SALES	CAT_SALES	SUBCAT_SALES	RANK_IN_LINE
Peripherals and Accessories	Printer Supplies	129	11606.65	15976.16	15976.16	1
Peripherals and Accessories	Printer Supplies	127	2264.61	15976.16	15976.16	2
Peripherals and Accessories	Printer Supplies	128	2104.9	15976.16	15976.16	3
Software/Other	Bulk Pack Diskettes	126	563.55	7207.29	955.44	1
Software/Other	Bulk Pack Diskettes	125	391.89	7207.29	955.44	2
Software/Other	Recordable CDs	114	487.9	7207.29	1814.07	1
Software/Other	Recordable CDs	116	358.96	7207.29	1814.07	2
Software/Other	Recordable CDs	115	221.27	7207.29	1814.07	6
Software/Other	Recordable CDs	117	260.28	7207.29	1814.07	3
Software/Other	Recordable CDs	119	246.54	7207.29	1814.07	4
Software/Other	Recordable CDs	118	239.12	7207.29	1814.07	5
Software/Other	Recordable DVD Discs	123	3032.13	7207.29	4437.78	1
Software/Other	Recordable DVD Discs	124	1405.65	7207.29	4437.78	2

Reporting

```
select SUBSTR(PROD_CATEGORY,1,8) as CATEG, PROD_SUBCATEGORY, PROD_ID, SALES
from
(
select P.PROD_CATEGORY, P.PROD_SUBCATEGORY, P.PROD_ID,
       SUM(AMOUNT_SOLD) as SALES,
       SUM(SUM(AMOUNT_SOLD)) over (partition by P.PROD_CATEGORY) as CAT_SALES,
       SUM(SUM(AMOUNT_SOLD)) over (partition by P.PROD_SUBCATEGORY) as SUBCAT_SALES,
       RANK() over  (partition by P.PROD_SUBCATEGORY  order by SUM(AMOUNT_SOLD)desc ) as RANK_IN_LINE
     from SALES S, CUSTOMERS C, COUNTRIES CO, PRODUCTS P
     where S.CUST_ID=C.CUST_ID
           and C.COUNTRY_ID=CO.COUNTRY_ID and  S.PROD_ID=P.PROD_ID
           and S.TIME_ID=TO_DATE('11-OCT-2000')
     group by P.PROD_CATEGORY, P.PROD_SUBCATEGORY, P.PROD_ID
order by PROD_CATEGORY, PROD_SUBCATEGORY
)
  where SUBCAT_SALES>0.2*CAT_SALES and RANK_IN_LINE<=5
```

CATEG	PROD_SUBCATEGORY	PROD_ID	SALES
Peripher	Printer Supplies	129	11606.65
Peripher	Printer Supplies	127	2264.61
Peripher	Printer Supplies	128	2104.9
Software	Recordable CDs	114	487.9
Software	Recordable CDs	116	358.96
Software	Recordable CDs	117	260.28
Software	Recordable CDs	119	246.54
Software	Recordable CDs	118	239.12
Software	Recordable DVD Discs	123	3032.13
Software	Recordable DVD Discs	124	1405.65

ROW_NUMBER Function

The **ROW_NUMBER** function assigns a unique number (sequentially, starting from 1, as defined by ORDER BY) to each row within the partition.

```
ROW_NUMBER ( ) OVER ( [query_partition_clause] order_by_clause )
```

ROW_NUMBER Function

```
select CHANNEL_DESC,
       CALENDAR_MONTH_DESC,
       TO_CHAR(TRUNC(SUM(AMOUNT_SOLD), -5), '9,999,999,999') SALES$,
       ROW_NUMBER() over (order by TRUNC(SUM(AMOUNT_SOLD), -6) desc) as ROW_NUMBER
from SALES, PRODUCTS, CUSTOMERS, TIMES, CHANNELS
where SALES.PROD_ID=PRODUCTS.PROD_ID and SALES.CUST_ID=CUSTOMERS.CUST_ID
  and SALES.TIME_ID=TIMES.TIME_ID and SALES.CHANNEL_ID=CHANNELS.CHANNEL_ID
  and TIMES.CALENDAR_MONTH_DESC in ('2001-09', '2001-10')
group by CHANNEL_DESC, CALENDAR_MONTH_DESC
```

CHANNEL_DESC	CALENDAR_MONTH_DESC	SALES$	ROW_NUMBER
Direct Sales	2001-10	1,000,000	1
Direct Sales	2001-09	1,100,000	2
Internet	2001-09	500,000	3
Partners	2001-09	600,000	4
Partners	2001-10	600,000	5
Internet	2001-10	700,000	6

RATIO_TO_REPORT Function

The **RATIO_TO_REPORT** function computes the ratio of a value to the sum of a set of values. If the expression value expression evaluates to NULL, RATIO_TO_REPORT also evaluates to NULL, but it is treated as zero for computing the sum of values for the denominator.

```
RATIO_TO_REPORT ( expr ) OVER ( [query_partition_clause] )
```

RATIO_TO_REPORT Function

```
select CH.CHANNEL_DESC, TO_CHAR(SUM(AMOUNT_SOLD),'9,999,999') as SALES,
       TO_CHAR(SUM(SUM(AMOUNT_SOLD)) over (), '9,999,999') as TOTAL_SALES,
       TO_CHAR(RATIO_TO_REPORT(SUM(AMOUNT_SOLD)) over (), '9.999') as RATIO_TO_REPORT
from SALES S, CHANNELS CH
where S.CHANNEL_ID=CH.CHANNEL_ID  and S.TIME_ID=TO_DATE('11-OCT-2000')
group by CH.CHANNEL_DESC
```

CHANNEL_DESC	SALES	TOTAL_SALES	RATIO_TO_REPORT
Partners	8,391	23,183	.362
Direct Sales	14,447	23,183	.623
Internet	345	23,183	.015

LAG/LEAD

The **LAG and LEAD** functions are useful for comparing values when the relative positions of rows can be known reliably. They work by specifying the count of rows which separate the target row from the current row. Because the functions provide access to more than one row of a table at the same time without a self-join, they can enhance processing speed. The LAG function provides access to a row at a given offset prior to the current position, and the LEAD function provides access to a row at a given offset after the current position.

```
{LAG | LEAD} ( value_expr [, offset] [, default] )
        [RESPECT NULLS|IGNORE NULLS]
        OVER ( [query_partition_clause] order_by_clause )
```

LAG/LEAD

```
select TIME_ID, TO_CHAR(SUM(AMOUNT_SOLD),'9,999,999') as SALES,
       TO_CHAR(LAG(SUM(AMOUNT_SOLD),1) over (order by TIME_ID),'9,999,999') as LAG1,
       TO_CHAR(LEAD(SUM(AMOUNT_SOLD),1) over (order by TIME_ID),'9,999,999') as LEAD1
from SALES
where TIME_ID>=TO_DATE('10-OCT-2000') and TIME_ID<=TO_DATE('14-OCT-2000')
group by TIME_ID
```

TIME_ID	SALES	LAG1	LEAD1
10-OCT-00	238,479	(null)	23,183
11-OCT-00	23,183	238,479	24,616
12-OCT-00	24,616	23,183	76,516
13-OCT-00	76,516	24,616	29,795
14-OCT-00	29,795	76,516	(null)

SQL for Analysis and Reporting

Q&A

MTN.BI.03

Oracle SQL for Aggregation in Data Warehouses

Author: Volha Kutsevol
Lead Software Engineer
Volha_Kutsevol@epam.com