Data Analysis with SQL Window Functions

Product – Orderline – Orders

```
product (
    productid int,
    productgroupname varchar(50),
    …
);
```

```
orderline (
    productid int,
    orderid int,
    totalprice real,
    …
);
```

```
orders (
    orderid int,
    orderdate date,
    …
);
```

Schema from: Gordon S. Linoff, **Data Analysis Using SQL,** Wiley; 2 edition (Dec. 14, 2015)

orders per month per category

```
CREATE TABLE T AS
SELECT
        TO_CHAR(orderdate, 'YYYY') AS year,
        TO_CHAR(orderdate, 'MM') AS month,
        productgroupname AS cat,
        COUNT(orderid) AS countorders,
        SUM(orderline.totalprice) AS revenue
FROM orders JOIN
           orderline USING(orderid) JOIN
           products USING(productid)
GROUP BY
      TO_CHAR(orderdate, 'YYYY'),
      TO_CHAR(orderdate, 'MM'),
      productgroupname
ORDER BY 1,2;
```

year	month	cat	countorders	revenue
2009	10	ARTWORK	1782	45416
2009	10	BOOK	731	15299
2009	10	OCCASION	169	3476
2009	11	ARTWORK	2138	79390
2009	11	BOOK	2353	45808
2009	11	OCCASION	485	10041
2009	12	APPAREL	17	719
…	…	…	…	…

WINDOW FUNCTIONS

For each category, which months were revenues below the average of the current year?

- **First**: Mix **detail** (individual tuples) and **aggregate** information over a **window** of tuples.
- **Second**: Extract what you want with an enclosing query.

First: Mix **detail** and aggregate information over **window**

```
SELECT cat, year, month, revenue,
      AVG(revenue) OVER (PARTITION BY cat, year) AS avgrev
FROM T;
```

Detail

Window
All tuples of **T** with same **cat** and **year** as in the detail part.

cat	year	month	revenue	avgrev
APPAREL	2009	12	719.1	719.1
ARTWORK	2009	12	32924.25	52576.42
ARTWORK	2009	10	45415.5	52576.42
ARTWORK	2009	11	79389.5	52576.42
BOOK	2009	11	45808.31	26678
BOOK	2009	12	18926.84	26678
…				

Second: Extract what you want with enclosing query

```
SELECT cat, year, month
FROM
   (SELECT cat, year, month, revenue,
          AVG(revenue) OVER (PARTITION BY cat, year) AS avgrev
    FROM T) X
WHERE revenue < avgrev
ORDER BY cat, year, month;
```

cat	year	month	revenue	avgrev
APPAREL	2009	12	719.1	719.1
ARTWORK	2009	12	32924.25	52576.42
ARTWORK	2009	10	45415.5	52576.42
ARTWORK	2009	11	79389.5	52576.42
BOOK	2009	11	45808.31	26678
BOOK	2009	12	18926.84	26678
…				

Several levels of aggregations at once

```
SELECT
    year, month, cat, revenue,
    AVG(revenue) OVER (PARTITION BY year, cat) AS avg_y_c,
    AVG(revenue) OVER (PARTITION BY year) AS avg_y,
    AVG(revenue) OVER () AS avg
FROM T;
```

year	month	cat	revenue	avg_y_c	avg_y	avg
2009	12	APPAREL	719.1001	719.1001	23875.16	24436.53
2009	12	ARTWORK	32924.25	52576.42	23875.16	24436.53
2009	10	ARTWORK	45415.5	52576.42	23875.16	24436.53
2009	11	ARTWORK	79389.5	52576.42	23875.16	24436.53
2009	10	BOOK	15298.94	26677.74	23875.16	24436.53
2009	11	BOOK	45807.43	26677.74	23875.16	24436.53
2009	12	BOOK	18926.86	26677.74	23875.16	24436.53
…						

…without window functions

```
SELECT T.year, T.month, T.cat, X.avgr_y_c, Y.avg_y, Z.avg
FROM T,

   (SELECT year, cat, AVG(revenue) AS avgr_y_c
    FROM T
    GROUP BY year, cat) X,

   (SELECT year, AVG(revenue) AS avg_y
    FROM T
    GROUP BY year) Y,

   (SELECT AVG(revenue) AS avg
    FROM T) Z

WHERE T.year=X.year AND T.cat=X.cat AND T.year=Y.year;
```

Which months did the revenues from a product category drop below those of the same month of the previous year?

```
SELECT *
FROM (
  SELECT year, month, cat, revenue,
        LAG(revenue,12) OVER (PARTITION BY cat ORDER BY year, month)
            AS prev_year_rev
  FROM T ) X
WHERE revenue < prev_year_rev
ORDER BY year, month;
```

year	month	cat	revenue	prev_year_rev
2010	10	ARTWORK	24186.65	45415.5
2010	11	OCCASION	2983.7	10040.52
2010	12	OCCASION	2930.38	9729.37
2011	1	OCCASION	5646.9	8491.27
2011	2	CALENDAR	297	494.65
…				

Which months did revenues from a category drop below those of the same month one year ago without increasing again the next year?

```
SELECT *
FROM (
  SELECT year, month, cat, revenue,
       LAG(revenue,12) OVER (PARTITION BY cat ORDER BY year, month)
                                                   AS prev_year_rev,
       LEAD(revenue,12) OVER (PARTITION BY cat ORDER BY year, month)
                                                   AS next_year_rev
  FROM T) X
WHERE revenue < prev_year_rev AND next_year_rev <= revenue
ORDER BY year, month;
```

year	month	cat	revenue	prev_year_rev	next_year_rev
2011	5	ARTWORK	64538.66	64620.75	32268.12
2011	7	ARTWORK	49170.82	61181.51	42955.31
2011	9	OTHER	887.8	1939.79	364.47
2011	10	OTHER	1000.21	3154.7	297.26
2011	11	BOOK	56548.76	62757.74	31030.95
…					

Which are the top 10 months in terms of revenue for each category?

ROW_NUMBER is a binary operator; it takes a tuple and an ordered set and returns the rank of the tuple in the set. It is always for sure that the tuple is member of the set.

```
SELECT *
FROM (
  SELECT cat, year, month, round(revenue,-3),
     ROW_NUMBER() OVER (PARTITION BY cat ORDER BY round(revenue,-3) DESC) AS rank
  FROM T) X
WHERE rank<=10
ORDER BY cat, rank;
```

- The “window” is the subset of tuples with same **cat** as the detail (first part of SELECT).
- The window is ordered by revenue (rounded to the nearest thousand). Ties are broken arbitrarily.
- ROW_NUMBER() returns the rank of the detail in the ordered window.

Results (ROW_NUMBER)

cat	year	month	round(revenue,-3)	rank
APPAREL	2014	8	19000	1
APPAREL	2014	7	14000	2
APPAREL	2015	3	13000	3
APPAREL	2014	12	10000	4
APPAREL	2014	11	9000	5
APPAREL	2013	12	**8000**	**6**
APPAREL	2014	10	**8000**	**7**
APPAREL	2012	12	**7000**	**8**
APPAREL	2015	12	**7000**	**9**
APPAREL	2014	5	6000	10
ARTWORK	2014	12	400000	1
ARTWORK	2013	12	389000	2
…	…	…	…	…

Ties are broken arbitrarily.

Which are the top 10 months in terms of revenue for each category?

```
SELECT *
FROM (
  SELECT cat, year, month, round(revenue,-3),
    RANK() OVER (PARTITION BY cat ORDER BY round(revenue,-3) DESC) AS rank
  FROM T) X
WHERE rank<=10
ORDER BY cat, rank;
```

Similar to ROW_NUMBER, but ties are not broken.
See next slides for results.

Results (RANK)

cat	year	month	round(revenue,-3)	rank
APPAREL	2014	8	19000	1
APPAREL	2014	7	14000	2
APPAREL	2015	3	13000	3
APPAREL	2014	12	10000	4
APPAREL	2014	11	9000	5
APPAREL	2013	12	8000	6
APPAREL	2014	10	8000	6
APPAREL	2012	12	7000	8
APPAREL	2015	12	7000	8
APPAREL	2014	5	6000	10
APPAREL	2015	2	6000	10
APPAREL	2014	6	6000	10
APPAREL	2014	9	6000	10
ARTWORK	2014	12	400000	1
ARTWORK	2013	12	389000	2
…	…	…	…	…

Ties are not broken. However, ranks produced have gaps.

Which are the top 10 months in terms of revenue for each category?

```
SELECT *
FROM (
  SELECT cat, year, month, round(revenue,-3),
    DENSE_RANK() OVER (PARTITION BY cat ORDER BY round(revenue,-3) DESC)
AS rank
  FROM T) X
WHERE rank<=10
ORDER BY cat, rank;
```

Results (DENSE_RANK)

cat	year	month	round(revenue,-3)	rank
APPAREL	2014	8	19000	1
APPAREL	2014	7	14000	2
APPAREL	2015	3	13000	3
APPAREL	2014	12	10000	4
APPAREL	2014	11	9000	5
APPAREL	2013	12	**8000**	**6**
APPAREL	2014	10	**8000**	**6**
APPAREL	2012	12	**7000**	**7**
APPAREL	2015	12	**7000**	**7**
APPAREL	2014	5	**6000**	**8**
APPAREL	2015	2	**6000**	**8**
APPAREL	2014	9	**6000**	**8**
APPAREL	2014	6	**6000**	**8**
APPAREL	2015	11	**5000**	**9**
APPAREL	2016	6	**5000**	**9**
APPAREL	2011	12	**4000**	**10**
APPAREL	2015	6	**4000**	**10**
APPAREL	2015	5	**4000**	**10**
APPAREL	2015	4	**4000**	**10**
APPAREL	2012	11	**4000**	**10**
ARTWORK	2014	12	400000	1
ARTWORK	2013	12	389000	2
…	…	…	…	…

Ties are not broken.
Ranks produced don't have gaps. They a dense.

If there were no ties,
ROW_NUMBER, RANK and
DENSE_RANK would be the same.