SQL Analítico - Funcións Analíticas

Laboratorio de Bases de Datos

Funcións Analíticas

Nesta seccións mostrarse un conxunto de exercicios onde se deberán utilizar **funcións analíticas** para obter o resultado desexado.

1. PARTITION BY e ORDER BY

A continuación, será necesario utilizar o almacén de datos que garda as vendas de produtos (*dwsales*, *dwtimes*, etc.) para resolver os seguintes exercicios.

1. Mostra, para cada venda, a súa data, o identificador do cliente, o identificador do produto e o número de vendas que se fixeron o mesmo día que ela. Ordenar de tal forma que as máis antigas aparecerán primeiro.

   Solución:

```
01 | SELECT time_id, cust_id, prod_id,
02 |        COUNT(*) OVER (PARTITION BY time_id) n_vendas_dia
03 | FROM   dwsales s
04 | ORDER BY time_id
```

2. Modifica a consulta 1 para que mostre tamén o número de vendas feitas o mesmo mes (do mesmo ano).

   Solución:

```
01 | SELECT s.time_id, cust_id, prod_id,
02 |        COUNT(*) OVER (PARTITION BY s.time_id) n_dia,
03 |        COUNT(*) OVER (PARTITION BY
04 |                  calendar_month_number, calendar_year) n_mes
05 | FROM   dwsales s JOIN dwtimes t ON s.time_id=t.time_id
06 | ORDER BY s.time_id
```

3. Para cada venda feita durante o ano 1998, mostra a súa data, o identificador do cliente, o identificador do produto, os ingresos desa venda, os ingresos medios de tódalas vendas nese mes e a diferenza entre os ingresos da venda coa media do mes.

   Solución:

```
01 | SELECT s.time_id, cust_id, prod_id, amount_sold,
02 |        AVG(amount_sold) OVER (PARTITION BY
03 |                          calendar_month_number) avg_total,
04 |        amount_sold - AVG(amount_sold) OVER (PARTITION BY
05 |                          calendar_month_number) diff_avg
06 | FROM   dwsales s JOIN dwtimes t ON s.time_id=t.time_id
07 | WHERE  calendar_year = 1998
08 | ORDER BY s.time_id
```

Nota: Non é necesario engadir o campo *calendar_year* no *PARTITION BY* porque tódalas vendas (e meses) son do mesmo ano.

4. Para cada venda, mostra o identificador do cliente e o identificador do produto. Obtén tamén a cantidade total do produto comprado polo cliente até ese día, a cantidade total do produto comprado polo cliente (en toda a historia) e a cantidade total de tódolos produtos comprados polo cliente.

   Solución:

```
01 | SELECT cust_id, prod_id, time_id, quantity_sold AS Cantidade,
02 |        SUM(quantity_sold) OVER (PARTITION BY cust_id, prod_id
03 |                          ORDER BY time_id) AS a_pro,
04 |        SUM(quantity_sold) OVER (PARTITION BY
05 |                                    cust_id, prod_id) AS t_pro,
06 |        SUM(quantity_sold) OVER (PARTITION BY cust_id) AS total
07 | FROM   dwsales
08 | ORDER BY cust_id, prod_id, time_id
```

5. Para cada venda, mostra a data, ingresos e cantidade de cada una, os ingresos acumulados ata ese día e os ingresos acumulados ata ese día dentro do seu mes (só se teñen en conta as vendas que se fixeron o mesmo mes). Obtén tamén o cantidade máis grande e pequena que se produciu nunha venda no seu ano.

   Solución:

```
01 | SELECT s.time_id, amount_sold AS Impo, quantity_sold AS Cant,
02 |        SUM(amount_sold) OVER (ORDER BY s.time_id) acum,
03 |        SUM(amount_sold) OVER (PARTITION BY
04 |                          calendar_year, calendar_month_number
05 |                          ORDER BY s.time_id) acum_mes,
06 |        MAX(quantity_sold) OVER (PARTITION BY
07 |                          calendar_year) max_can,
08 |        MIN(quantity_sold) OVER (PARTITION BY
09 |                          calendar_year) min_can
10 | FROM   dwsales s JOIN dwtimes t ON s.time_id=t.time_id
11 | ORDER BY s.time_ida;
```

6. Obtén a data de cada venda, o seu trimestre, o ingreso, a media de ingresos no seu trimestre, e a evolución do media dos ingresos dentro do seu trimestre (*quarter*) ata o día da venda actual (e dicir, como varía o valor medio cada día). Mostra tamén a diferenza entre o ingreso da venda e a media do seu trimestre. Aplicar só para as vendas producidas no ano 1998.

   Solución:

```
SELECT s.time_id, calendar_quarter_number AS TR,
       amount_sold AS Imp,
       AVG(amount_sold) OVER (PARTITION BY
                         calendar_quarter_number) avg_ing_trm,
       AVG(amount_sold) OVER (PARTITION BY
                         calendar_quarter_number
                              ORDER BY  calendar_month_number,
                         day_number_in_month) ac_avg_ing_trm,
       amount_sold - AVG(amount_sold) OVER (PARTITION BY
                         calendar_quarter_number) diff_ing_trm
FROM  dwsales s JOIN dwtimes t ON s.time_id = t.time_id
WHERE calendar_year = 1998;
```

2. Funcións analíticas sobre agrupamentos

7. Mostra o importe ingresado cada día e o importe acumulado até ese día.

   Solución:

```
SELECT s.time_id,
       SUM (amount_sold) Ingresado,
       SUM(SUM (amount_sold)) OVER (
              ORDER BY s.time_id) AS acum_ing
FROM  dwsales s
GROUP BY s.time_id
ORDER BY s.time_id
```

8. Para cada mes de cada ano, mostra:

   - O número de vendas nese mes.
   - O número de venda feitas até ese mes (inclusive).
   - O número de vendas nese trimestre.
   - O número de vendas feitas até ese trimestre (inclusive).
   - O número de vendas nese ano.
   - O número de vendas feitas até ese ano (inclusive).

   Solución:

```
SELECT calendar_month_number AS mes, calendar_year AS ano,
       COUNT(*) as n_mes,
       SUM(COUNT(*)) OVER (ORDER BY calendar_year,
            calendar_month_number) as a_mes,
       SUM(COUNT(*)) OVER (PARTITION BY
            calendar_quarter_number, calendar_year) as n_tri,
       SUM(COUNT(*)) OVER (ORDER BY calendar_year,
            calendar_quarter_number) as a_tri,
       SUM(COUNT(*)) OVER (PARTITION BY
            calendar_year) as n_ano,
       SUM(COUNT(*)) OVER (ORDER BY calendar_year) as a_ano
FROM  dwsales s JOIN dwtimes t ON t.time_id=s.time_id
GROUP BY calendar_year, calendar_quarter_number,
         calendar_month_number
ORDER BY calendar_year, calendar_month_number
```

3. Funcións analíticas LAG e LEAD

9. Mostra o mes, o ano e o importe total que se fixo nas vendas dese mes. Obtén tamén a diferenza do importe co mes anterior e seguinte

   Solución:

```
SELECT calendar_month_number AS mes, calendar_year AS ano,
       SUM(amount_sold) AS Cant,
       SUM(amount_sold) - LAG(SUM(amount_sold)) OVER (
        ORDER BY calendar_year, calendar_month_number) diff_p,
       SUM(amount_sold) - LEAD(SUM(amount_sold)) OVER (
        ORDER BY calendar_year, calendar_month_number) diff_n
FROM  dwsales s JOIN dwtimes t ON t.time_id=s.time_id
GROUP BY calendar_year, calendar_month_number
ORDER BY calendar_year, calendar_month_number
```

10. Modifica a consulta anterior para que mostre a cantidade vendida nese mes nas columnas coa diferenza en vez do valor *NULL*.

    Solución:

```
SELECT calendar_month_number AS mes, calendar_year AS ano,
       SUM(quantity_sold) AS Cant,
       SUM(quantity_sold) - LAG(SUM(quantity_sold),1,0) OVER (
        ORDER BY calendar_year, calendar_month_number) diff_p,
      SUM(quantity_sold) - LEAD(SUM(quantity_sold),1,0) OVER (
        ORDER BY calendar_year, calendar_month_number) diff_n
FROM  dwsales s JOIN dwtimes t ON t.time_id=s.time_id
GROUP BY calendar_year, calendar_month_number
ORDER BY calendar_year, calendar_month_number

```

11. Repite a consulta anterior onde as diferenzas sexa cos dous meses anteriores e os dous posteriores.

    Solución:

```
SELECT calendar_month_number AS mes, calendar_year AS ano,
       SUM(quantity_sold) AS Cant,
       SUM(quantity_sold) - LAG(SUM(quantity_sold,2,0)) OVER (
        ORDER BY calendar_year, calendar_month_number) diff_p,
      SUM(quantity_sold) - LEAD(SUM(quantity_sold),2,0) OVER (
        ORDER BY calendar_year, calendar_month_number) diff_n
FROM  dwsales s JOIN dwtimes t ON t.time_id=s.time_id
GROUP BY calendar_year, calendar_month_number
ORDER BY calendar_year, calendar_month_number

```

12. Mostra por cada día, os ingresos obtidos e a diferenza cos ingresos do do mesmo día pero do ano anterior.

    Solución:

```
SELECT day_number_in_month AS dia,
       calendar_month_number AS mes,
       calendar_year AS ano,
```

```
04 |        SUM(amount_sold) AS ingresos,
05 |        SUM(amount_sold) - LAG(SUM(amount_sold))
06 |              OVER (PARTITION BY calendar_month_number,
07 |                                  day_number_in_month
08 |              ORDER BY calendar_year) AS diff_p
09 |  FROM  dwsales s JOIN dwtimes t ON s.time_id=t.time_id
10 |  GROUP BY calendar_year, calendar_month_number,
11 |           day_number_in_month
12 |  ORDER BY calendar_year, calendar_month_number,
13 |           day_number_in_month
```

4. RANK e DENSE RANK

13. Mostra o número de vendas feitas cada mes (de cada ano), e o media de vendas mensuais por ano. Obtén tamén a clasificación de cada mes dentro do seu ano segundo o número de vendas (1 significa o mes máis vendido).

    Solución:

```
01 |  SELECT calendar_month_number AS mes, calendar_year AS ano,
02 |        COUNT(*) AS vendas,
03 |        AVG(COUNT(*)) OVER (
04 |                       PARTITION BY calendar_year) AS media_ano,
05 |        RANK() OVER (PARTITION BY calendar_year
06 |                     ORDER BY COUNT(*)) AS ranking
07 |  FROM     dwsales s JOIN dwtimes t ON s.time_id=t.time_id
08 |  GROUP BY calendar_year, calendar_month_number
09 |  ORDER BY calendar_year, calendar_month_number
```

14. Por cada subcategoría, mostra a categoría á que pertence, o número de vendas dos seus produtos e a cantidade vendida. Mostra tamén a clasificación dentro da sua categoría (o 1 significa a máis vendida) e a clasificación global entre tódalas subcategorías (o 1 significa a máis vendida). Ordena o resultado pola cantidade vendida e as mesmas subcategorías dunha mesma categoría deben aparecer xuntas.

    Solución:

```
01 |  SELECT prod_category, prod_subcategory,
02 |        COUNT(*) AS vendas, SUM(quantity_sold) AS cantidade,
03 |        RANK() OVER (PARTITION BY prod_category
04 |                     ORDER BY SUM(quantity_sold) DESC) ranking,
05 |        RANK() OVER (ORDER BY SUM(quantity_sold) DESC) Global
06 |  FROM   dwsales s JOIN dwproducts p ON s.prod_id=p.prod_id
07 |  GROUP BY prod_category, prod_subcategory
08 |  ORDER BY prod_category, ranking
```

15. Deseña unha consulta onde mostre para cada produto cun prezo mínimo maior de 200 (prod_min_price), o número de vendas, a cantidade de produtos vendidos e a súa clasificación dos produtos máis vendidos (o 1 significa que foi o máis vendido) para cada mes de 1998.

    Solución:

```
SELECT p.prod_id, p.prod_name,
       calendar_month_number AS mes,
       COUNT(*) AS vendas,
       SUM(quantity_sold) AS cantidade,
       RANK() OVER (PARTITION BY calendar_month_number
              ORDER BY SUM(quantity_sold) DESC) AS rank
FROM  dwsales s JOIN dwproducts p ON s.prod_id=p.prod_id
JOIN dwtimes t    ON s.time_id=t.time_id
WHERE calendar_year=1998 AND prod_min_price > 200
GROUP BY p.prod_id, p.prod_name, calendar_month_number
```

5. Cláusula marco

16. Tendo en conta só as vendas feitas por clientes que naceron no ano 1980. Para cada mes de cada ano mostra o seguinte:

    - O número de vendas realizas ese mes.
    - O número de vendas acumuladas ata ese mes.
    - A media do número de vendas mensuais tendo en conta só os 2 meses anteriores e o actual.
    - A media do número de vendas mensuais tendo en conta unicamente o mes anterior, o actual e o seguinte.

    Solución:

```
SELECT calendar_year AS Ano, calendar_month_number AS Mes,
       COUNT(*) AS vendas,
       SUM(COUNT(*)) OVER (PARTITION BY calendar_year
                           ORDER BY calendar_month_number) AS acum,
       AVG(COUNT(*)) OVER (PARTITION BY calendar_year
                           ORDER BY calendar_month_number
         ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS prev_3m,
       AVG(COUNT(*)) OVER (PARTITION BY calendar_year
                          ORDER BY calendar_month_number
         ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS p1_f1_m
FROM  dwsales s JOIN dwcustomers c ON s.cust_id=c.cust_id
                JOIN dwtimes t     ON s.time_id=t.time_id
WHERE c.cust_year_of_birth = 1980
GROUP BY calendar_year, calendar_month_number
```

6. Deseño paso a paso dunha consulta

17. Mostra por cada mes de cada ano os ingresos feitos ese mes (a suma do importe) e a suma de ingresos entre os 3 meses anteriores e o mes actual.. Ordena o resultado por orden cronolóxico, as máis antigas primeiro.

    Solución:

```
SELECT calendar_month_number AS mes, calendar_year AS ano,
       SUM(amount_sold) Ingresos,
       SUM(SUM(amount_sold)) OVER (
        ORDER BY calendar_year, calendar_month_number
        ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS sum_3
FROM     dwsales s JOIN dwtimes t ON s.time_id = t.time_id
GROUP BY calendar_year, calendar_month_number
ORDER BY calendar_year, calendar_month_number
```

18. Modifica a consulta anterior para que mostre para cada mes a diferenza cos ingresos do mes pasado.

    Solución:

```
SELECT calendar_month_number AS mes, calendar_year AS ano,
       SUM(amount_sold) Ingresos,
       SUM(SUM(amount_sold)) OVER (
        ORDER BY calendar_year, calendar_month_number
        ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS sum_3,
       SUM(amount_sold) - LAG(SUM(amount_sold)) OVER (
ORDER BY calendar_year, calendar_month_number) AS dif
FROM     dwsales s JOIN dwtimes t ON s.time_id = t.time_id
GROUP BY calendar_year, calendar_month_number
ORDER BY calendar_year, calendar_month_number
```

19. Modifica a consulta anterior para que mostre a diferenza con mes do ano anterior. Pódese supoñer que houbo vendas tódolos meses do ano.

    Solución:

```
SELECT calendar_month_number AS mes, calendar_year AS ano,
       SUM(amount_sold) Ingresos,
       SUM(SUM(amount_sold)) OVER (
        ORDER BY calendar_year, calendar_month_number
        ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS sum_3,
       SUM(amount_sold) - LAG(SUM(amount_sold)) OVER (
        ORDER BY calendar_year, calendar_month_number) AS dif,
       SUM(amount_sold) - LAG(SUM(amount_sold)) OVER (
        PARTITION BY calendar_month_number
        ORDER BY calendar_year) AS dif_ano
FROM     dwsales s JOIN dwtimes t ON s.time_id = t.time_id
GROUP BY calendar_year, calendar_month_number
ORDER BY calendar_year, calendar_month_number
```

Alternativa:

```
SELECT calendar_month_number AS mes, calendar_year AS ano,
       SUM(amount_sold) Ingresos,
       SUM(SUM(amount_sold)) OVER (
        ORDER BY calendar_year, calendar_month_number
        ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS sum_3,
       SUM(amount_sold) - LAG(SUM(amount_sold)) OVER (
        ORDER BY calendar_year, calendar_month_number) AS dif,
       SUM(amount_sold) - LAG(SUM(amount_sold), 12) OVER (
        ORDER BY calendar_year, calendar_month_number) AS
    dif_ano
FROM     dwsales s JOIN dwtimes t ON s.time_id = t.time_id
```

```
11 | GROUP BY calendar_year, calendar_month_number
12 | ORDER BY calendar_year, calendar_month_number
```

20. Modifica a consulta anterior para que mostre as seguintes clasificacións (rankings) segundo os seus ingresos (primeiro os de maior ingresos):

    - Clasificación de cada mes de cada no durante toda a historia.
    - Clasificación de cada mes de cada ano dentro do seu ano.
    - Clasificación de cada mes de cada ano entre o mesmo mes (xaneiro, febreiro, etc.)

    Ordenar o resultado pola clasificación de toda a historia.

    Solución:

```
01 | SELECT calendar_month_number AS mes, calendar_year AS ano,
02 |        SUM(amount_sold) Ingresos,
03 |        SUM(SUM(amount_sold)) OVER (
04 |         ORDER BY calendar_year, calendar_month_number
05 |         ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) AS sum_3,
06 |        SUM(amount_sold) - LAG(SUM(amount_sold)) OVER (
07 |         ORDER BY calendar_year, calendar_month_number) AS dif,
08 |        SUM(amount_sold) - LAG(SUM(amount_sold)) OVER (
09 |         PARTITION BY calendar_month_number
10 |         ORDER BY calendar_year) AS dif_ano,
11 |        RANK() OVER (ORDER BY SUM(amount_sold) DESC) AS rank_a,
12 |        RANK() OVER (PARTITION BY calendar_year
13 |         ORDER BY SUM(amount_sold) DESC) AS rank_ano,
14 |        RANK() OVER (PARTITION BY calendar_month_number
15 |         ORDER BY SUM(amount_sold) DESC) AS rank_mes
16 | FROM     dwsales s JOIN dwtimes t ON s.time_id = t.time_id
17 | GROUP BY calendar_year, calendar_month_number
18 | ORDER BY rank_a
```

21. Mostra por cada mes de cada ano os ingresos feitos ese mes (a suma do importe) e tamén:

    - O importe máximo obtido por un mes (en toda a historia).
    - O importe máximo obtido por un mes até o seu mes.
    - O importe máximo obtido por un mes do seu ano.
    - O importe máximo obtido por un mes do seu ano ata o seu mes.
    - O importe máximo obtido por un mes entre os 6 meses anteriores e o actual.
    - O importe máximo obtido por un mes entre os 6 meses anteriores e o os 6 posteriores.
    - O importe máximo obtido por un mes entre o mesmo mes de calquera ano (o importe máximo no mes xaneiro, en febreiro, etc. de tódolos anos).

- O importe máximo obtido por un mes entre o mesmo mes do mesmo ano ou de anteriores (o importe máximo no mes xaneiro, en febreiro, etc. pero só tendo en conta ese ano e os anteriores).

Ordena o resultado por orden cronolóxico, as máis antigas primeiro.

Solución:

```
01 | SELECT calendar_month_number AS mes, calendar_year AS ano,
02 |        SUM(amount_sold) Ingresos,
03 |        MAX(SUM(amount_sold)) OVER() max_ingresos,
04 |        MAX(SUM(amount_sold)) OVER(
05 |         ORDER BY calendar_year, calendar_month_number)
       max_actual,
06 |        MAX(SUM(amount_sold)) OVER(PARTITION BY calendar_year)
       max_ano,
07 |        MAX(SUM(amount_sold)) OVER(PARTITION BY calendar_year
08 |         ORDER BY calendar_month_number) max_ano_actual,
09 |        MAX(SUM(amount_sold)) OVER(
10 |         ORDER BY calendar_year, calendar_month_number
11 |         ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) max_6_pre,
12 |        MAX(SUM(amount_sold)) OVER(
13 |         ORDER BY calendar_year, calendar_month_number
14 |         ROWS BETWEEN 6 PRECEDING AND 6 FOLLOWING)
       max_6_pre_fol,
15 |        MAX(SUM(amount_sold)) OVER(PARTITION BY
       calendar_month_number) max_mes,
16 |        MAX(SUM(amount_sold)) OVER(PARTITION BY
       calendar_month_number
17 |         ORDER BY calendar_year) max_mes_actual
18 | FROM     dwsales s JOIN dwtimes t ON s.time_id = t.time_id
19 | GROUP BY calendar_year, calendar_month_number
20 | ORDER BY calendar_year, calendar_month_number
```

22. Modifica a consulta anterior para que mostre o nome do mes (January, February, etc.) e non o número de mes (1, 2, etc.). Débese manter o orde cronolóxica. **Solución:**

```
01 | SELECT calendar_month_name AS mes, calendar_year AS ano,
02 |        SUM(amount_sold) Ingresos,
03 |        MAX(SUM(amount_sold)) OVER() max_ingresos,
04 |        MAX(SUM(amount_sold)) OVER(
05 |         ORDER BY calendar_year, calendar_month_number)
       max_actual,
06 |        MAX(SUM(amount_sold)) OVER(PARTITION BY calendar_year )
        max_ano,
07 |        MAX(SUM(amount_sold)) OVER(PARTITION BY calendar_year
08 |         ORDER BY calendar_month_number) max_ano_actual,
09 |        MAX(SUM(amount_sold)) OVER(
10 |         ORDER BY calendar_year, calendar_month_number
11 |         ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) max_6_pre,
12 |        MAX(SUM(amount_sold)) OVER(
13 |         ORDER BY calendar_year, calendar_month_number
14 |         ROWS BETWEEN 6 PRECEDING AND 6 FOLLOWING)
       max_6_pre_fol,
15 |        MAX(SUM(amount_sold)) OVER(PARTITION BY
       calendar_month_number) max_mes,
```

```
16 |         MAX(SUM(amount_sold)) OVER(PARTITION BY
       calendar_month_number
17 |          ORDER BY calendar_year) max_mes_actual
18 | FROM      dwsales s JOIN dwtimes t ON s.time_id = t.time_id
19 | GROUP BY calendar_year, calendar_month_number,
       calendar_month_name
20 | ORDER BY calendar_year, calendar_month_number
```

7. Consultas avanzadas

Nas seguintes consultas é posible que sexa necesario usar GROUPING SETS/ROLLUP/CUBE ou PIVOT.

23. Deseña unha consulta onde mostre para cada produto a súa clasificación dos produtos máis vendidos (o 1 significa que foi o máis vendido) para cada mes de 1998. Os meses deben aparecer como columnas. Para o produto quérese o seu nome e identificador, e ordenar por este último.

    Solución:

```
01 | SELECT *
02 | FROM (
03 |   SELECT p.prod_id, p.prod_name,
04 |          calendar_month_number AS mes,
05 |          RANK() OVER (PARTITION BY calendar_month_number
06 |                       ORDER BY SUM(quantity_sold) DESC) AS
       r_global
07 |   FROM dwsales s JOIN dwproducts p ON s.prod_id=p.prod_id
08 |                  JOIN dwtimes t    ON s.time_id=t.time_id
09 |   WHERE calendar_year=1998
10 |   GROUP BY p.prod_id, p.prod_name, calendar_month_number
11 | )
12 | PIVOT (SUM(r_global) FOR mes IN (1, 2, 3, 4, 5,
13 |                                  6, 7, 8, 9, 10, 11, 12))
14 | ORDER BY prod_id
```

24. Deseña un consulta que obteña por cada cidade e tendo en conta só o primeiro trimestre de 1998:

    - O número de vendas feitas en cada mes.
    - O número de vendas totais no primeiro trimestre de 1998.
    - A diferenza entre o número de vendas feitas en cada mes nesa cidade e o número máximo de vendas feitas nese mes nunha cidade do mesmo país
    - A diferenza entre o número de vendas feitas no primeiro trimestre nesa cidade e o número máximo de vendas feitas nese trimestre nunha cidade do mesmo país

As filas das cidades do mesmo país deben aparecer xuntas. Ordena de forma cronolóxica.

Solución:

```
SELECT CUST_CITY AS cidade, calendar_month_number AS mes,
       COUNT(*) AS vendas,
       COUNT(*) - MAX(COUNT(*)) OVER (PARTITION BY
                country_id, calendar_month_number) AS max_pais
FROM  dwsales s JOIN dwcustomers c ON s.cust_id=c.cust_id
                JOIN dwtimes t     ON s.time_id=t.time_id
WHERE calendar_year = 1998 AND calendar_quarter_number = 1
GROUP BY cust_city, country_id, ROLLUP(calendar_month_number)
ORDER BY country_id, CUST_CITY, mes
```

25. Modifica a consulta anterior para que mostre os datos do primeiro trimestre de cada ano. Non é necesario mostrar o número de vendas.

Solución:

```
SELECT CUST_CITY AS Cidade, calendar_month_number AS mes,
       calendar_year AS ano, COUNT(*) AS vendas,
       COUNT(*) - MAX(COUNT(*)) OVER (PARTITION BY country_id,
        calendar_month_number, calendar_year) AS max_pais
FROM  dwsales s JOIN dwcustomers c ON s.cust_id=c.cust_id
                JOIN dwtimes t     ON s.time_id=t.time_id
WHERE calendar_quarter_number = 1
GROUP BY cust_city, country_id,GROUPING SETS((calendar_year,
    calendar_month_number), calendar_year )
ORDER BY country_id, CUST_CITY, ano, mes
```

26. Modifica a consulta anterior para que, para cada ano e mes (xaneiro, febreiro e marzo) mostre a media das diferenzas. Por exemplo, para xaneiro será a media das diferenzas de xaneiro de 1998, xaneiro de 1999, etc. Os meses deberán aparecer como columnas cos nomes **xaneiro**, **febreiro** e **marzo** respectivamente. Opcional: mostra tamén unha columna coa media das diferencias do primeiro trimestre.

Solución:

```
SELECT *
FROM (
  SELECT CUST_CITY AS Cidade,
         COALESCE(calendar_month_number, 13) AS mes,
         COUNT(*) - MAX(COUNT(*)) OVER (
          PARTITION BY country_id, calendar_month_number,
                                   calendar_year) AS max_pais
  FROM  dwsales s JOIN dwcustomers c ON s.cust_id=c.cust_id
                  JOIN dwtimes t     ON s.time_id=t.time_id
  WHERE calendar_quarter_number = 1
  GROUP BY cust_city, country_id,
       GROUPING SETS((calendar_year, calendar_month_number),
                     calendar_year ))
PIVOT (AVG(max_pais) FOR mes IN (1 AS "Xaneiro",
                2 AS "Febreiro", 3 AS "Marzo", 13 AS "Ano"))
ORDER BY cidade
```