Business School
UNIVERSITY OF COLORADO DENVER

Information Systems Program

Module 3
SQL Analytic Functions

Lesson 5b: PostgreSQL query patterns for RATIO_TO_REPORT

Lesson Objectives

- Explain at least one query pattern to formulate RATIO_TO_REPORT in PostgreSQL
- Write SELECT statements using the query formulation pattern
- Recognize lack of optimization for the database compiler

Ratio_To_Report Example (Oracle)

- Contribution ratio on sum of dollar sales by year and customer city
- Partition on year
- Order result by year and descending sum of sales
- PostgreSQL does not support RATIO_TO_REPORT

```
SELECT TimeYear, CustCity, SUM(SalesDollar) AS SumSales,
       RATIO_TO_REPORT(SUM(SalesDollar))
         OVER (PARTITION BY TimeYear) AS SumSalesRatio
 FROM SSCustomer, SSSales, SSTimeDim
 WHERE SSSales.CustID = SSCustomer.CustId
   AND SSSales.TimeNo = SSTimeDim.TimeNo
 GROUP BY TimeYear, CustCity
 ORDER BY TimeYear, SUM(SalesDollar) DESC;
```


Query Patterns for PostgreSQL

- Pattern 1 (SUM as analytic function and aggregate function)
  - Compute numerators using SUM aggregate function
  - Compute denominators using SUM analytic function
- Pattern 2 (SUM as aggregate function)
  - SELECT result columns, computed ratio column (Numerator in subquery 1/ Denominator in subquery 2)
  - WHERE condition to join partition columns from subqueries
  - Subquery 1 calculates numerators in ratios with grouping on all columns
  - Subquery 2 calculates denominators in ratios with grouping on partitioning columns

PostgreSQL Example 1 with Pattern 1

- SUM(SalesDollar): aggregate function
- Outer SUM in SUM(SUM(SalesDollar)) OVER(...): analytic function

```
SELECT TimeYear, CustCity, SUM(SalesDollar) AS SumSales,
       SUM(SalesDollar) / SUM ( SUM(SalesDollar) )
         OVER (PARTITION BY TimeYear) AS SumSalesRatio
 FROM SSCustomer, SSSales, SSTimeDim
 WHERE SSSales.CustID = SSCustomer.CustId
   AND SSSales.TimeNo = SSTimeDim.TimeNo
 GROUP BY TimeYear, CustCity
 ORDER BY TimeYear, SumSales  DESC;
```

PostgreSQL Example 1 (Pattern 2)

- Outer query with ratio calculations
- Subquery 1 with numerator calculations
- Subquery 2 with denominator calculations

```
SELECT X1.TimeYear, CustCity, SumSales, SumSales/SumYearSales AS SumSalesRatio
FROM
( SELECT SSTimeDim.TimeYear, CustCity, SUM(SalesDollar) AS SumSales
   FROM SSCustomer, SSSales, SSTimeDim
   WHERE SSSales.CustID = SSCustomer.CustId
     AND SSSales.TimeNo = SSTimeDim.TimeNo
   GROUP BY SSTimeDim.TimeYear, CustCity ) X1,
 ( SELECT TimeYear, SUM(SalesDollar) as SumYearSales
     FROM SSSales, SSTimeDim
     WHERE SSSales.TimeNo = SSTimeDim.TimeNo
     GROUP BY TimeYear ) X2
WHERE X1.TimeYear = X2.TimeYear
ORDER BY X1.TimeYear, SumSales DESC;
```


Additional Problems I

- Example 7
  - Contribution ratio on sum of 2021 units sold by month and item brand
  - Partition on month
  - Display month, item brand, sum of units sold, and contribution ratio
  - Order result by month and descending sum of units sold

Example 7 (Oracle)

```
SELECT TimeMonth, ItemBrand, SUM(SalesUnits) AS SumBrandUnits,
       RATIO_TO_REPORT(SUM(SalesUnits))
         OVER (PARTITION BY TimeMonth) AS SumUnitsRatio
 FROM SSItem, SSSales, SSTimeDim
 WHERE SSSales.ItemID = SSitem.ItemId
   AND SSSales.TimeNo = SSTimeDim.TimeNo
   AND TimeYear = 2021
 GROUP BY TimeMonth, ItemBrand
 ORDER BY TimeMonth, SUM(SalesUnits) DESC;
```

PostgreSQL Example 7 (Pattern 1)

- SUM(SalesDollar): aggregate function
- Outer SUM in SUM(SUM(SalesDollar)) OVER(...): analytic function

```
SELECT TimeMonth, ItemBrand, SUM(SalesUnits) AS SumUnits,
       SUM(SalesUnits) / SUM ( SUM(SalesUnits) )
         OVER (PARTITION BY TimeMonth) AS SumUnitsRatio
 FROM SSItem, SSSales, SSTimeDim
 WHERE SSSales.ItemID = SSitem.ItemId
   AND SSSales.TimeNo = SSTimeDim.TimeNo AND TimeYear = 2021
 GROUP BY TimeMonth, ItemBrand
 ORDER BY TimeMonth, SumUnits DESC;
```

PostgreSQL Example 7 (Pattern 2)

- Outer query with ratio calculations
- Subquery 1 with numerator calculations and CAST
- Subquery 2 with denominator calculations and CAST

```
SELECT X1.TimeMonth, ItemBrand, X1.SumUnits,
       SumUnits/SumMonthUnits AS SumUnitRatio
 FROM
 ( SELECT SSTimeDim.TimeMonth, ItemBrand,
          CAST(SUM(SalesUnits) AS NUMERIC) AS SumUnits
    FROM SSItem, SSSales, SSTimeDim
    WHERE SSSales.ItemId = SSitem.ItemId
      AND SSSales.TimeNo = SSTimeDim.TimeNo AND TimeYear = 2021
    GROUP BY SSTimeDim.TimeMonth, ItemBrand ) X1,
 ( SELECT SSTimeDim.TimeMonth,
          CAST(SUM(SalesUnits) AS NUMERIC) AS SumMonthUnits
     FROM SSSales, SSTimeDim
     WHERE SSSales.TimeNo = SSTimeDim.TimeNo AND TimeYear = 2021
     GROUP BY TimeMonth) X2
 WHERE X1.TimeMonth = X2.TimeMonth
 ORDER BY X1.TimeMonth, SumUnits DESC;
```


Additional Problems II

- Example 8
  - Contribution ratio on sum of 2021 units sold by item brand
  - No partitioning
  - Display item brand, sum of units, and contribution ratio
  - Order result by descending sum of units

```
SELECT ItemBrand, SUM(SalesUnits) AS SumBrandUnits,
       RATIO_TO_REPORT(SUM(SalesUnits)) OVER () AS SumBrandUnitRatio
 FROM SSItem, SSSales, SSTimeDim
 WHERE SSSales.ItemID = SSitem.ItemId
   AND SSSales.TimeNo = SSTimeDim.TimeNo AND TimeYear = 2021
 GROUP BY ItemBrand
 ORDER BY SUM(SalesUnits) DESC;
```

PostgreSQL Example 8 (Pattern 1)

- SUM(SalesDollar): aggregate function
- Outer SUM in SUM(SUM(SalesDollar)) OVER(...): analytic function

```
SELECT ItemBrand, SUM(SalesUnits) AS SumBrandUnits,
       SUM(SalesUnits) / SUM ( SUM(SalesUnits) )
         OVER () AS SumBrandUnitRatio
 FROM SSItem, SSSales, SSTimeDim
 WHERE SSSales.ItemID = SSitem.ItemId
   AND SSSales.TimeNo = SSTimeDim.TimeNo AND TimeYear = 2021
 GROUP BY ItemBrand
 ORDER BY SumBrandUnits DESC;
```

PostgreSQL Example 8 (Pattern 2)

- Outer query with ratio calculations
- Subquery 1 with numerator calculations and CAST
- Subquery 2 with denominator calculation and CAST
- No join condition in the outer query

```
SELECT ItemBrand, X1.SumBrandUnits,
       SumBrandUnits/SumTotUnits AS SumBrandUnitRatio
 FROM
 ( SELECT ItemBrand, CAST(SUM(SalesUnits) AS NUMERIC) AS SumBrandUnits
    FROM SSItem, SSSales, SSTimeDim
    WHERE SSSales.ItemId = SSitem.ItemId
      AND SSSales.TimeNo = SSTimeDim.TimeNo AND TimeYear = 2021
    GROUP BY ItemBrand ) X1,
 ( SELECT CAST(SUM(SalesUnits) AS NUMERIC) AS SumTotUnits
     FROM SSSales, SSTimeDim
     WHERE SSSales.TimeNo = SSTimeDim.TimeNo AND TimeYear = 2021 ) X2
 ORDER BY SumBrandUnits DESC;
```

Summary

- No RATIO_TO_REPORT function in PostgreSQL
- Alternative using two relatively simple query patterns
- Pattern 1 using SUM analytic and aggregate functions
- Pattern 2 using SUM aggregate functions and nested queries in the FROM clause
- Lack of optimization by database compiler