Grouping Sets in SQL

The GROUPING SETS clause is used in conjunction with the GROUP BY clause to allow you to easily summarize data by aggregating a fact over as many dimensions as you like.

SQL GROUP BY clause

Recall that the SQL GROUP BY clause allows you to summarize an aggregation such as SUM or AVG over the distinct members, or groups, of a categorical variable or dimension.

You can extend the functionality of the GROUP BY clause using SQL clauses such as CUBE and ROLLUP to select multiple dimensions and create multi-dimensional summaries. These two clauses also generate grand totals, like a report you might see in a spreadsheet application or an accounting style sheet. Just like CUBE and ROLLUP, the SQL GROUPING SETS clause allows you to aggregate data over multiple dimensions but does not generate grand totals.

Examples

Let's start with an example of a regular GROUP BY aggregation and then compare the result to that of using the GROUPING SETS clause. We'll use data from a fictional company called Shiny Auto Sales. The schema for the company's warehouse is displayed in the entity-relationship diagram in Figure 1.

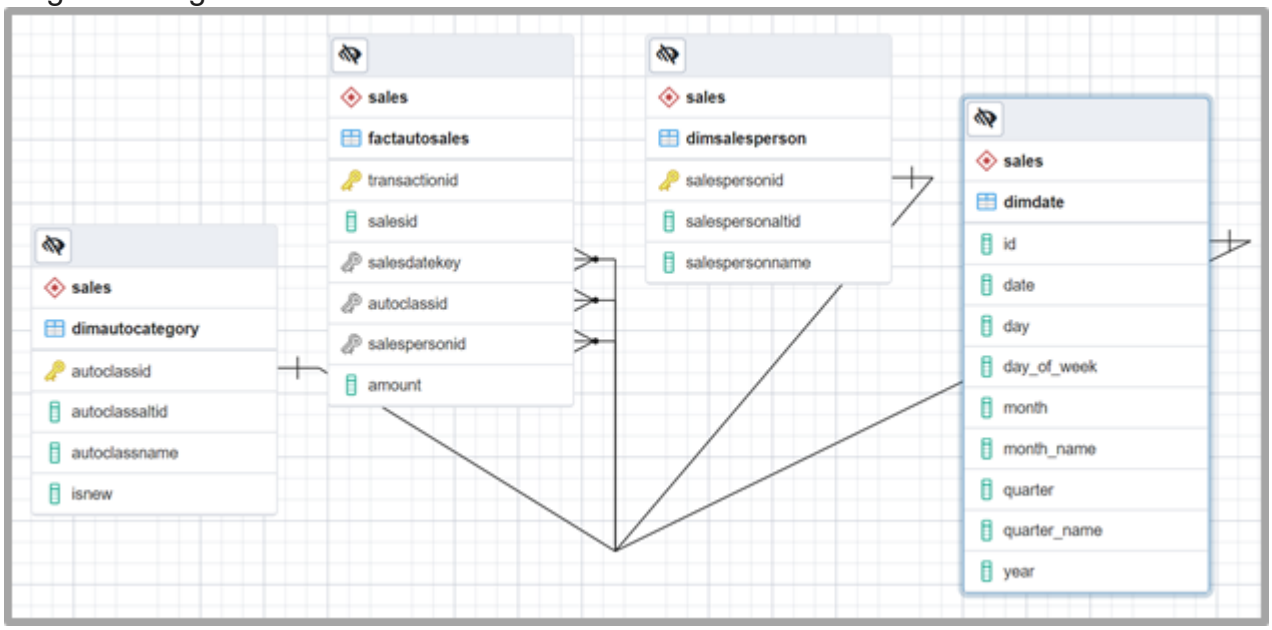

Fig. 1. Entity-relationship diagram for a "sales" star schema based on the fictional "Shiny Auto Sales" company.

We'll work with a convenient materialized view of a completely denormalized fact table from the sales star schema, called DNsales, which looks like the following:

```
SELECT * FROM DNsales LIMIT 5;

    date    | autoclassname | isnew |   salespersonname    |   amount
------------+---------------+-------+----------------------+------------
 2021-01-04 | 4 Door Sedan  | t     | Jake Salesbouroughs | $42,000.00
 2021-01-04 | Truck         | t     | Gocart Joe           | $17,680.00
 2021-01-05 | Truck         | t     | Jake Salesbouroughs | $37,100.00
 2021-01-05 | Midsize SUV   | t     | Cadillac Jack        | $26,500.00
 2021-01-05 | Truck         | f     | Jane Honda           |  $8,200.00
(5 rows)
```

This DNsales table was created by joining all the dimension tables to the central fact table and selecting only the columns which are displayed. Each record in DNsales contains details for an

individual sales transaction.

Example 1

Consider the following SQL code which invokes GROUP BY on the auto class dimension to summarize total sales of new autos by auto class.

```
SELECT
    autoclassname,
    SUM(amount)
FROM
    DNsales
WHERE
    isNew=True
GROUP BY
    autoclassname
```

The result looks like this:

```
 autoclassname |     sum
---------------+-------------
 Compact SUV   | $168,598.00
 Truck         |  $96,879.00
 Midsize SUV   |  $58,599.00
 4 Door Sedan  |  $42,000.00
```

Example 2

Now suppose you want to generate a similar view, but you also want to include the total sales by salesperson. You can use the GROUPING SETS clause to access both the auto class and salesperson dimensions in the same query. Here is the SQL code you can use to summarize total sales of new autos, both by auto class and by salesperson, all in one expression:

```
SELECT
    autoclassname,
    salespersonname,
    SUM(amount)
FROM
    DNsales
WHERE
    isNew=True
GROUP BY
GROUPING SETS(autoclassname,
              salespersonname)
```

Here is the query result. Notice that the first four rows are identical to the result of Example 1, while the next 5 rows are what you would get by substituting salespersonname for autoclassname in Example 1.

```
 autoclassname |   salespersonname   |     sum
---------------+---------------------+-------------
 Compact SUV   |                     | $168,598.00
 Truck         |                     |  $96,879.00
 Midsize SUV   |                     |  $58,599.00
 4 Door Sedan  |                     |  $42,000.00
               | Happy Dollarmaker   |  $74,500.00
               | Jake Salesbouroughs | $121,199.00
               | Cadillac Jack       |  $26,500.00
               | William Jeepman     |  $51,999.00
               | Gocart Joe          |  $91,878.00
```

Essentially, applying GROUPING SETS to the two dimensions, salespersonname and autoclassname, provides the same result that you would get by appending the two individual results of applying GROUP BY to each dimension separately as in Example 1.