```
CREATE TABLE t_sales
(
    country         text,
    product_name    text,
    year              int,
    amount_sold     numeric
);

INSERT INTO t_sales VALUES
    ('Argentina', 'Shoes', 2020, 12),
    ('Argentina', 'Shoes', 2021, 14),
    ('Argentina', 'Hats', 2020, 54),
    ('Argentina', 'Hats', 2021, 57),
    ('Germany', 'Shoes', 2020, 34),
    ('Germany', 'Shoes', 2021, 29),
    ('Germany', 'Hats', 2020, 19),
    ('Germany', 'Hats', 2021, 22),
    ('USA', 'Shoes', 2020, 99),
    ('USA', 'Shoes', 2021, 103),
    ('USA', 'Hats', 2020, 81),
    ('USA', 'Hats', 2021, 90)
;
```

SELECT *FROM t_sales

simple aggregation:

```
SELECT   country, sum(amount_sold)
       FROM     t_sales
       GROUP BY 1;


SELECT   country, product_name, sum(amount_sold)
       FROM     t_sales
       GROUP BY 1, 2
       ORDER BY 1, 2;


SELECT CASE WHEN country = 'USA'
                THEN 'USA'
                ELSE 'non-US'
              END,
              sum(amount_sold)
       FROM t_sales
       GROUP BY 1;
```

GROUPING SETS: The basic building blocks

```
SELECT  country, product_name, sum(amount_sold)
       FROM     t_sales
       GROUP BY GROUPING SETS ((1), (2))
       ORDER BY 1, 2;
```

Or

```
SELECT  NULL AS country , product_name, sum(amount_sold)
       FROM     t_sales
       GROUP BY  1, 2
       UNION ALL
       SELECT  country, NULL, sum(amount_sold)
       FROM     t_sales
       GROUP BY  1, 2
       ORDER BY 1, 2;
```

However, the GROUPING SETS version is ways more efficient because it only has to read the data once.

ROLLUP: Adding the “bottom line”

```
      SELECT  country, product_name, sum(amount_sold)
       FROM     t_sales
       GROUP BY ROLLUP (1, 2)
       ORDER BY 1, 2;
```

```
*change NULL with TOTAL*
```

```
SELECT   CASE WHEN country IS NULL
                  THEN 'TOTAL' ELSE country END,
              CASE WHEN product_name IS NULL
                  THEN 'TOTAL' ELSE product_name END,
              sum
       FROM (SELECT    country, product_name, sum(amount_sold)
             FROM       t_sales
             GROUP BY ROLLUP (1, 2)
             ORDER BY 1, 2
           ) AS x;
```

CUBE: Creating data cubes in PostgreSQL efficiently

ROLLUP is useful if you want to add the “bottom line”. However, you often want to see all combinations of countries and products. GROUP BY CUBE will do exactly that:

```
SELECT  country, product_name, sum(amount_sold)
       FROM     t_sales
       GROUP BY CUBE (1, 2)
       ORDER BY 1, 2;
```

Technically, it’s the same as: GROUP BY country + GROUP BY product_name + GROUP BY country_product_name + GROUP BY ()

Grouping sets: Execution plans

```
explain SELECT country, product_name, sum(amount_sold)
           FROM t_sales
           GROUP BY CUBE (1, 2)
           ORDER BY 1, 2;
```

Looking at the MixedAggregate also reveals which aggregations are performed as part of the grouping set.

Query for other operations

Drill down

SELECT … GROUP BY ROLLDOWN(columns);

Example query:

SELECT Time, Location, product ,sum(revenue) AS Profit FROM sales GROUP BY ROLLDOWN(Time, Location, product);

Slicing

Selection conditions on some attributes using <WHERE clause> <Group by> and   aggregation on some attribute

Example query:

Select products, sum(revenue) from sales where Products= ‘OPV’ GROUP BY Products ;

Dicing

Selection conditions on some attributes using <WHERE clause> Group by and aggregation on some attribute

Example query:

Select products, sum(revenue) from sales where Products= ‘EL’ and Location=’Europe’ group by Products;