PostgreSQL GROUPING SETS

dvdrental/postgres@PostgreSQL

Query Editor | Query History | Explain | Messages

```
1  --Let's get started by creating a new table calledsales for the demonstration.
2
3  DROP TABLE IF EXISTS sales;
4  CREATE TABLE sales (
5      brand VARCHAR NOT NULL,
6      segment VARCHAR NOT NULL,
7      quantity INT NOT NULL,
8      PRIMARY KEY (brand, segment)
9  );
10
11 INSERT INTO sales (brand, segment, quantity)
12 VALUES
13     ('ABC', 'Premium', 100),
14     ('ABC', 'Basic', 200),
15     ('XYZ', 'Premium', 100),
16     ('XYZ', 'Basic', 300);
```

Data Output | Notifications

	title character varying	release_year smallint
1	12 Angry Men	1957
2	The Shawshank Redemption	1994

dvdrental/postgres@PostgreSQL

Query Editor | Query History | Explain | Messages

```
1  --the following query uses the GROUP BY clause to return the number of products sold by brand and segment
2  --it defines a grouping set of the brand and segment which is denoted by (brand, segement)
3  SELECT
4      brand,
5      segment,
6      SUM (quantity)
7  FROM
8      sales
9  GROUP BY
10     brand,
11     segment;
```

Data Output | Notifications

	brand [PK] character varying	segment [PK] character varying	sum bigint
1	XYZ	Basic	300
2	ABC	Premium	100
3	ABC	Basic	200
4	XYZ	Premium	100

dvdrental/postgres@PostgreSQL

Query Editor | Query History | Explain | Messages

```
1  --The following query finds the number of products sold by a brand. It defines a grouping set (brand):
2
3  SELECT
4      brand,
5      SUM (quantity)
6  FROM
7      sales
8  GROUP BY
9      brand;
```

Data Output | Notifications

	brand character varying	sum bigint
1	ABC	300
2	XYZ	400

dvdrental/postgres@PostgreSQL

Query Editor | Query History | Explain | Messages

```sql
--The following query finds the number of products sold by segment. It defines a grouping set (segment):

SELECT
    segment,
    SUM (quantity)
FROM
    sales
GROUP BY
    segment;
```

Data Output | Notifications

	segment character varying	sum bigint
1	Basic	500
2	Premium	200

Query Editor | Query History | Explain | Messages

```sql
--The following query finds the number of products sold for all brands and segments.
--It defines an empty grouping set which is denoted by ().

SELECT SUM (quantity) FROM sales;
```

Data Output | Notifications

	sum bigint
1	700

```sql
--UNION ALL requires all result sets to have the same number of columns with compatible data types,
--you need to adjust the queries by adding NULL to the selection list of each as shown below:
SELECT
    brand,
    segment,
    SUM (quantity)
FROM
    sales
GROUP BY
    brand,
    segment

UNION ALL
SELECT
    brand,
    NULL,
    SUM (quantity)
FROM
```

Data Output

	brand character varying	segment character varying	sum bigint
1	XYZ	Basic	300
2	ABC	Premium	100
3	ABC	Basic	200
4	XYZ	Premium	100
5	ABC	[null]	300

Query Editor Query History Explain Messages Notifications

```sql
--To apply this syntax to the above example, you can use GROUPING SETS clause instead of the UNION ALL clause like this:

SELECT
    brand,
    segment,
    SUM (quantity)
FROM
    sales
GROUP BY
    GROUPING SETS (
        (brand, segment),
        (brand),
        (segment),
        ()
    );
```

Data Output

	brand [PK] character varying	segment [PK] character varying	sum bigint
1	[null]	[null]	700
2	XYZ	Basic	300
3	ABC	Premium	100
4	ABC	Basic	200
5	XYZ	Premium	100
6	ABC	[null]	300
7	XYZ	[null]	400

dvdrental/postgres@PostgreSQL

Query Editor Query History Explain Messages Notifications

```sql
--The column_name or expression must match with the one specified in the GROUP BY clause
SELECT
    GROUPING(brand) grouping_brand,
    GROUPING(segment) grouping_segment,
    brand,
    segment,
    SUM (quantity)
FROM
    sales
GROUP BY
    GROUPING SETS (
        (brand),
        (segment),
        ()
    )
```

Data Output

	grouping_brand integer	grouping_segment integer	brand [PK] character varying	segment [PK] character varying	sum bigint
1	0	1	ABC	[null]	300
2	0	1	XYZ	[null]	400
3	1	0	[null]	Basic	500
4	1	0	[null]	Premium	200
5	1	1	[null]	[null]	700

Query Editor Query History Explain Messages Notifications

```sql
--You can use the GROUPING() function in the HAVING clause to find the subtotal of each brand like this:

SELECT
    GROUPING(brand) grouping_brand,
    GROUPING(segment) grouping_segment,
    brand,
    segment,
    SUM (quantity)
FROM
    sales
GROUP BY
    GROUPING SETS (
        (brand),
        (segment),
        ()
```

Data Output

	grouping_brand integer	grouping_segment integer	brand [PK] character varying	segment [PK] character varying	sum bigint
1	0	1	ABC	[null]	300
2	0	1	XYZ	[null]	400