

POSTGRESQL GROUPING SETS

Dashboard Properties SQL Statistics Dependencies Dependents dvdrental/postgres@PostgreSQL 13 *

Query Editor Query History Explain Notifications

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

Data Output

	title character varying	release_year smallint
1	12 Angry Men	1957

Messages

NOTICE: table "sales" does not exist, skipping
INSERT 0 4

Query returned successfully in 1 secs 724 msec.

Dashboard Properties SQL Statistics Dependencies Dependents dvdrental/postgres@PostgreSQL 13 *

Query Editor Query History Explain Notifications

```
1 --the following query uses the GROUP BY clause to return the number of products sold by brand and segment.
2 SELECT
3     brand,
4     segment,
5     SUM (quantity)
6 FROM
7     sales
8 GROUP BY
9     brand,
10    segment;
```

Data Output

	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

Messages

Successfully run. Total query runtime: 8 secs 358 msec.
4 rows affected.

✓ Successfully run. Total query runtime: 8 secs 358 msec. 4

Dashboard Properties SQL Statistics Dependencies Dependents dvdrental/postgres@PostgreSQL 13

Query Editor Query History Explain Notifications

```
1 --The following query finds the number of products sold by segment
2 SELECT
3     segment,
4     SUM (quantity)
5 FROM
6     sales
7 GROUP BY
8     segment;
```

Data Output

	segment character varying	sum bigint
1	Basic	500
2	Premium	200

Dashboard Properties SQL Statistics Dependencies Dependents dvdrental/postgres@PostgreSQL 13 *

Query Editor Query History Explain Notifications

```
1 -- query finds the number of products sold for all brands and segments. means an empty grouping
2 --set which is denoted by ()
3 SELECT SUM (quantity) FROM sales;
```

Data Output

	sum bigint
1	700

Dashboard Properties SQL Statistics Dependencies Dependents **dvdrental/postgres@PostgreSQL 13 ***

Query Editor Query History Explain Notifications Messages

```

1  --UNION ALL requires all result sets to have the same number of columns with compatible data types,
2  --you need to adjust the queries by adding NULL to the selection list
3
4  SELECT brand, segment, SUM (quantity) FROM sales GROUP BY brand, segment
5  UNION ALL SELECT brand, NULL, SUM (quantity) FROM sales GROUP BY brand
6  UNION ALL SELECT NULL, segment, SUM (quantity) FROM sales GROUP BY
7  segment UNION ALL SELECT NULL, NULL,
8  SUM (quantity) FROM sales;
9

```

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
6	XYZ	[null]	400
7	[null]	Basic	500
8	[null]	Premium	200
9	[null]	[null]	700

dvdrental/postgres@PostgreSQL 13

Query Editor Query History Explain Notifications Messages

```


1  --general syntax of the GROUPING SETS
2  SELECT c1, c2, aggregate_function(c3)
3  FROM table_name GROUP BY GROUPING SETS ( (c1, c2), (c1), (c2),() );

```

Data Output

dvdrental/postgres@PostgreSQL 13			
Query Editor Query History Explain Notifications Messages			
<pre> 1 -- use GROUPING SETS clause instead of the UNION ALL clause 2 SELECT 3 brand, 4 segment, 5 SUM (quantity) 6 FROM sales GROUP BY 7 GROUPING SETS ((brand, segment), (brand), (segment),()); </pre>			
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
8	[null]	Basic	500
9	[null]	Premium	200

dvdrental/postgres@PostgreSQL 13					
Query Editor Query History Explain Notifications Messages					
<pre> 1 --GROUPING() function returns bit 0 if the argument 2 --is a member of the current grouping set and 1 otherwise 3 SELECT 4 GROUPING(brand) grouping_brand,GROUPING(segment) grouping_segment, brand, segment,SUM (quantity) 5 FROM 6 sales 7 GROUP BY 8 GROUPING SETS ((brand),(segment),()) 9 ORDER BY brand, segment; </pre>					
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


dvdrental/postgres@PostgreSQL 13 ▾

Query Editor
Query History
Explain
Notifications
Messages

```

1  -- use the GROUPING() function in the HAVING clause to find the subtotal of each brand
2  SELECT
3      GROUPING(brand) grouping_brand,
4      GROUPING(segment) grouping_segment,
5      brand,
6      segment,
7      SUM (quantity)
8  FROM
9      sales
10 GROUP BY
11     GROUPING SETS (
12         (brand),
13         (segment),
14         ()
15     )
16 HAVING GROUPING(brand) = 0
17 ORDER BY
18     brand,
19     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	