

SELECT

   film\_id,

   title,

   release\_year

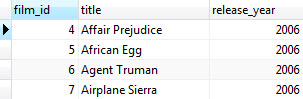
FROM

   film

ORDER BY

   film\_id

LIMIT 4 OFFSET 3;



SELECT

    film\_id,

    title

FROM

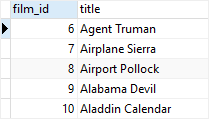
    film

ORDER BY

    title

OFFSET 5 ROWS

FETCH FIRST 5 ROW ONLY;



GROPING SETS:

CREATE TABLE sales (

    brand VARCHAR NOT NULL,

    segment VARCHAR NOT NULL,

    quantity INT NOT NULL,

    PRIMARY KEY (brand, segment)

);

INSERT INTO sales (brand, segment, quantity)

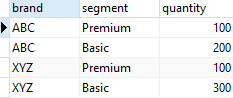
VALUES

    ('ABC', 'Premium', 100),

    ('ABC', 'Basic', 200),

    ('XYZ', 'Premium', 100),

    ('XYZ', 'Basic', 300);



the following query defines a grouping set of the brand and segment. It returns the number of products sold by brand and segment.

SELECT

    brand,

    segment,

    SUM (quantity)

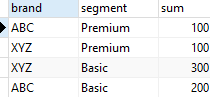
FROM

    sales

GROUP BY

    brand,

    segment;



The following query finds the number of product sold by brand. It defines a grouping set of the brand:

SELECT

    brand,

    SUM (quantity)

FROM

    sales

GROUP BY

    brand;

PostgreSQL GROUPING SETS - one grouping set brand

The following query finds the number of products sold by segment. It defines a grouping set of the segment:

SELECT

    segment,

    SUM (quantity)

FROM

    sales

GROUP BY

    segment;

PostgreSQL GROUPING SETS - one grouping set segment

The following query finds the number of products sold for all brands and segments. It defines an empty grouping set.

SELECT

    SUM (quantity)

FROM

    sales;

PostgreSQL GROUPING SETS - empty grouping set

SELECT

    brand,

    segment,

    SUM (quantity)

FROM

    sales

GROUP BY

    GROUPING SETS (

        (brand, segment),

        (brand),

        (segment),

        ()

    );



**PostgreSQL INSERT examples**

CREATE TABLE link (

   ID serial PRIMARY KEY,

   url VARCHAR (255) NOT NULL,

   name VARCHAR (255) NOT NULL,

   description VARCHAR (255),

   rel VARCHAR (50)

);

INSERT INTO link (url, name) VALUES  ('http://www.postgresqltutorial.com','PostgreSQL Tutorial');

INSERT INTO link (url, name) VALUES  ('http://www.oreilly.com','O''Reilly Media');

INSERT INTO link (url, name) VALUES ('http://www.google.com','Google'), ('http://www.yahoo.com','Yahoo'), ('http://www.bing.com','Bing');

ALTER TABLE link ADD COLUMN last\_update DATE;

ALTER TABLE link ALTER COLUMN last\_update

SET DEFAULT CURRENT\_DATE;

INSERT INTO link (url, name, last\_update) VALUES ('http://www.facebook.com','Facebook','2013-06-01');

INSERT INTO link (url, name, last\_update) VALUES

   ('https://www.tumblr.com/','Tumblr',DEFAULT);

CREATE TABLE link\_tmp (LIKE link); //same structure

Second, insert rows from the link table whose values of the date column are not NULL:

INSERT INTO link\_tmp

SELECT

   \*

FROM

   link

WHERE

   last\_update IS NOT NULL;

The following statement inserts a new row into the linktable and returns the last insert id:

INSERT INTO link (url, NAME, last\_update)

VALUES('http://www.postgresql.org','PostgreSQL',DEFAULT)

RETURNING id;

PostgreSQL last insert id

