2019320097_조이강

1. Use GROUPING SETS to count the number of films for each rental_rate and rating, respectively

```
dvdrental=# select rental_rate, rating, count(film_id) as film
dvdrental-# from film
dvdrental-# group by
dvdrental-# grouping sets (
dvdrental(# (rental_rate),
dvdrental(# (rating)
dvdrental(# );
 rental_rate | rating | film
-------------+--------+------
        2.99 |        |  323
        4.99 |        |  336
        0.99 |        |  341
             | NC-17  |  210
             | G      |  178
             | PG     |  194
             | PG-13  |  223
             | R      |  195
(8개 행)
```

2. For all combinations of actor_id and category name, print the information on the number of films in which the actors of actor_id 1 or 2 play (Note: It doesn’t mean that two actors star at the same time).

```
dvdrental=# select fa.actor_id, c.name, count(fa.film_id)
dvdrental-# from film_actor fa
dvdrental-# natural join film
dvdrental-# natural join film_category
dvdrental-# natural join category c
dvdrental-# where fa.actor_id = 1 or fa.actor_id = 2
dvdrental-# group by
dvdrental-# grouping sets (
dvdrental(# (fa.actor_id, c.name)
dvdrental(# )
dvdrental-# order by
dvdrental-# fa.actor_id;
 actor_id |    name     | count
----------+-------------+-------
        1 | Animation   |     1
        1 | Children    |     1
        1 | Classics    |     2
        1 | Comedy      |     1
        1 | Documentary |     1
        1 | Family      |     2
        1 | Foreign     |     1
        1 | Games       |     2
        1 | Horror      |     3
        1 | Music       |     1
        1 | New         |     2
        1 | Sci-Fi      |     1
        1 | Sports      |     1
        2 | Action      |     1
        2 | Animation   |     1
        2 | Children    |     1
        2 | Classics    |     2
        2 | Comedy      |     2
        2 | Documentary |     1
        2 | Drama       |     1
        2 | Family      |     4
        2 | Foreign     |     2
        2 | Games       |     1
        2 | Music       |     1
        2 | New         |     4
        2 | Sci-Fi      |     2
        2 | Travel      |     2
(27개 행)
```

3. Use ROLLUP to show total amount on rental_date by year, month, and day (Note: Use extract function).

```
dvdrental=# select
dvdrental-# extract (year from rental_date) y,
dvdrental-# extract (month from rental_date) m,
dvdrental-# extract (day from rental_date) d,
dvdrental-# count(rental_date)
dvdrental-# from rental
dvdrental-# group by rollup
dvdrental-# (extract (year from rental_date),
dvdrental(# extract (month from rental_date),
dvdrental(# extract (day from rental_date));
  y   | m | d  | count
------+---+----+-------
 2005 | 5 | 24 |     8
 2005 | 5 | 25 |   137
 2005 | 5 | 26 |   174
 2005 | 5 | 27 |   166
 2005 | 5 | 28 |   196
 2005 | 5 | 29 |   154
 2005 | 5 | 30 |   158
 2005 | 5 | 31 |   163
 2005 | 5 |    |  1156
 2005 | 6 | 14 |    16
 2005 | 6 | 15 |   348
 2005 | 6 | 16 |   324
 2005 | 6 | 17 |   325
 2005 | 6 | 18 |   344
 2005 | 6 | 19 |   348
 2005 | 6 | 20 |   331
 2005 | 6 | 21 |   275
 2005 | 6 |    |  2311
 2005 | 7 |  5 |    27
 2005 | 7 |  6 |   504
 2005 | 7 |  7 |   461
 2005 | 7 |  8 |   512
 2005 | 7 |  9 |   513
 2005 | 7 | 10 |   480
 2005 | 7 | 11 |   461
 2005 | 7 | 12 |   495
 2005 | 7 | 26 |    33
 2005 | 7 | 27 |   649
 2005 | 7 | 28 |   620
```

4. For each film category, find customer_id’s who is TOP-2 in order of the number of DVDs rented (if there are many ties, more than 2 customers can be printed)

```
dvdrental=# with cus_ranking as (
dvdrental(# select cat.name, cus.customer_id, count(r.rental_id),
dvdrental(# rank() over (partition by cat.name order by count(r.rental_id) desc) rank

dvdrental(# from film
dvdrental(# natural join film_category
dvdrental(# natural join category cat
dvdrental(# natural join inventory
dvdrental(# natural join rental r
dvdrental(# natural join customer cus
dvdrental(# group by
dvdrental(# grouping sets(
dvdrental(# (cat.name, cus.customer_id)
dvdrental(# )
dvdrental(# )
dvdrental-# select name, customer_id, rank
dvdrental-# from cus_ranking
dvdrental-# where rank <= 2;
    name     | customer_id | rank
-------------+-------------+------
 Action      |         209 |    1
 Action      |         410 |    1
 Action      |         147 |    1
 Animation   |         526 |    1
 Animation   |         445 |    2
 Animation   |         589 |    2
 Animation   |         302 |    2
 Animation   |          44 |    2
 Animation   |         406 |    2
 Animation   |         373 |    2
 Animation   |         380 |    2
 Children    |         574 |    1
 Children    |         467 |    1
 Children    |         209 |    1
 Classics    |         230 |    1
 Classics    |          75 |    1
 Classics    |         588 |    1
 Classics    |         480 |    1
 Classics    |           5 |    1
 Comedy      |         436 |    1
 Comedy      |         176 |    1
 Comedy      |         584 |    1
 Documentary |         208 |    1
```

5. For each customer, print her/his total amount, the total amount for her/his country, and dense ranking by the country’s total amount

```
dvdrental=# with amounts as (
dvdrental(# select distinct customer_id, country_id,
dvdrental(# sum(amount) over (partition by customer_id) cus_tot,
dvdrental(# sum(amount) over (partition by country_id) cnt_tot
dvdrental(# from customer
dvdrental(# natural join payment
dvdrental(# natural join address
dvdrental(# natural join city
dvdrental(# natural join country
dvdrental(# group by
dvdrental(# customer_id, country_id, amount)
dvdrental-# select customer_id, country_id, cus_tot, cnt_tot,
dvdrental-# dense_rank() over (order by cnt_tot desc) rank
dvdrental-# from amounts
dvdrental-# order by customer_id;
 customer_id | country_id | cus_tot | cnt_tot | rank
-------------+------------+---------+---------+------
           1 |         50 |   38.92 | 1185.61 |    4
           2 |        103 |   32.94 | 1344.31 |    3
           3 |         39 |   55.90 |   77.84 |   57
           4 |         64 |   36.92 |   73.85 |   59
           5 |         92 |   46.91 |  409.17 |   13
           6 |        103 |   33.93 | 1344.31 |    3
           7 |        108 |   44.91 |   73.85 |   59
           8 |         68 |   38.92 |   38.92 |   78
           9 |         71 |   24.94 |   42.89 |   75
          10 |         46 |   44.91 |  352.35 |   15
          11 |         50 |   41.92 | 1185.61 |    4
          12 |         44 |   29.94 | 2367.29 |    1
          13 |         97 |   53.92 |  581.86 |    9
          14 |        103 |   27.94 | 1344.31 |    3
          15 |         44 |   46.90 | 2367.29 |    1
          16 |        102 |   44.91 |  344.32 |   16
          17 |         80 |   29.93 | 1044.89 |    7
          18 |         69 |   29.93 |  493.00 |   11
          19 |         85 |   41.93 |  387.20 |   14
          20 |         72 |   35.92 |  164.65 |   30
          21 |         12 |   65.89 |  149.72 |   33
          22 |         62 |   39.93 |  121.77 |   41
          23 |         54 |   45.91 |  101.81 |   47
          24 |          6 |   33.93 |  488.97 |   12
          25 |         72 |   30.94 |  164.65 |   30
          26 |         41 |   54.90 |   54.90 |   68
          27 |         75 |   44.91 |  812.40 |    8
          28 |         44 |   35.93 | 2367.29 |    1
          29 |         50 |   46.91 | 1185.61 |    4
          30 |         92 |   41.92 |  409.17 |   13
          31 |         44 |   39.92 | 2367.29 |    1
          32 |         44 |   46.91 | 2367.29 |    1
          33 |         49 |   44.92 |  295.43 |   19
          34 |         50 |   26.94 | 1185.61 |    4
          35 |         34 |   31.93 |  126.72 |   40
          36 |         46 |   27.94 |  352.35 |   15
```