Trzeci sprawdzian z SQL 1.04.2020

Oczekujemy rozwiązania w postaci pliku zawierającego TREŚCI poleceń SQL, a nie znalezionej odpowiedzi. Nie będą sprawdzane jakiekolwiek zapytania niepoprawne składniowo, sprawdź swoje rozwiązanie używając `\i plik.sql` ! Plik możesz wysyłać wielokrotnie, sprawdzana będzie wyłącznie najnowsza wersja.

Wczytaj do swojej bazy danych plik `hsm.dump`. Jest to dump bazy `https://hsm.stackexchange.com/`poświęconej dyskusjom na tematy związane z historią matematyki i nauki.

Zachęcam do korzystania z dokumentacji PostgreSQL.

Format nazwy pliku z rozwiązaniem: `grupa-imie-nazwisko.sql`, gdzie `grupa` to inicjały prowadzącego Twoją grupę: (`pwi/plg/mpy/rfe/pga`), np. `pwi-Jan-Kowalski.sql`. Wymagany format pliku z rozwiązaniem (tu też podaj swoje imię, nazwisko i grupę):

```
-- Imię Nazwisko, grupa np. Jan Kowalski, pwi
-- Zadanie 1
<zapytanie>            -- zastąp napis `<zapytanie>` swoim zapytaniem :)

-- Zadanie 2
<zapytanie>
...
```

Zadanie 1 (2 pkt.) Uznajemy, że dany post `A` jest *wyczerpującą odpowiedzią* na inny post `B`, jeśli `B.AcceptedAnswerID=A.id`. Utwórz ranking najbardziej pomocnych użytkowników. W tym celu zdefiniuj perspektywę `ranking` z polami: `displayname`, `liczba_odpowiedzi` taką, że każdy użytkownik posiada w niej krotkę ze swoim `displayname` i liczbą wyczerpujących odpowiedzi, których udzielił. Perspektywa powinna być posortowana według `liczba_odpowiedzi` malejąco, a w drugiej kolejności wg `displayname` alfabetycznie.

Oryginalnie: `A.AcceptedAnswerID=B.id`

Oryginalnie: *którzy wyczerpująco odpowiedzieli na najwięcej pytań*

Ze względu na usterki w oryginalnym sformułowaniu zadania będzie ono sprawdzane z tolerancją dla ew. pomyłek, które mogły być tymi usterkami spowodowane. Ponadto (również w przypadku wszystkich pozostałych zadań) można sobie wyobrazić wiele różnych poprawnych rozwiązań, prezentowana wersja nie jest jedyną poprawną.

Oczywiście sortowanie ma większy sens przy pisaniu zapytania korzystającego z perspektywy, a nie ją definiującego.

```
CREATE VIEW ranking(DisplayName, liczba_odpowiedzi) AS
SELECT DisplayName, count(DISTINCT answers.Id) AS liczba_odpowiedzi
  FROM posts AS answers
    JOIN posts AS questions
              ON questions.AcceptedAnswerId = answers.Id
    RIGHT JOIN users
              ON answers.OwnerUserId=users.id
  GROUP BY users.Id, DisplayName
  ORDER BY 2 DESC, 1
```

```
;

-- sprawdzenie
SELECT * FROM ranking ORDER BY 2 DESC, 1;
```

Zadanie 2 (3 pkt.) Wyświetl `id`, `displayname` i `reputation` użytkowników, którzy

- nie dostali nigdy odznaki *Enlightened*
- ale mają więcej `upvotes` niż średnia liczba `upvotes` użytkowników z tą odznaką (uwaga: weź pod uwagę, że jeden użytkownik może dostać jedną odznakę wielokrotnie)
- oraz napisali więcej niż jeden komentarz pod postami stworzonymi w 2020 r.

Wynik uporządkuj rosnąco względem daty założenia konta użytkownika.

```
WITH enlightened AS
(SELECT DISTINCT u.id, u.upvotes
 FROM users u
 JOIN badges ON u.id = userid
WHERE badges.name = 'Enlightened')
SELECT u.id,u.displayname,u.reputation
 FROM users u
 JOIN comments c ON c.userid = u.id
 JOIN posts p ON p.id = c.postid
WHERE extract(year from p.creationdate) = 2020
 AND u.upvotes > (SELECT avg(upvotes) FROM enlightened)
 AND u.id NOT IN (SELECT id FROM enlightened)
GROUP BY u.id, u.displayname, u.reputation, u.creationdate
HAVING count(c.id)>1
ORDER BY u.creationdate;
```

Zadanie 3 (3 pkt.) Znajdź użytkowników (`id`, `displayname`), którzy mają pośredni kontakt z rekurencją. Mówimy, że użytkownik *ma pośredni kontakt z rekurencją*, jeśli:

- w `body` któregoś posta użył słowa *recurrence* lub
- napisał komentarz do posta osoby, która ma pośredni kontakt z rekurencją.

```
WITH RECURSIVE foo AS (
  SELECT users.Id, DisplayName
    FROM users JOIN posts ON users.Id = OwnerUserId
    WHERE body LIKE '%recurrence%'
  UNION
  SELECT users.Id, users.DisplayName
```

```
    FROM users
      JOIN comments ON users.Id = UserId
      JOIN posts ON PostId = posts.Id
      JOIN foo ON OwnerUserId = foo.Id
)
SELECT * FROM foo;
```