Próbny sprawdzian z SQL 18.03.2020

Dla każdego z poniższych zadań napisz odpowiednie polecenia SQL w zadaniu na SKOSIE. 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` znajdujący się na SKOSIE: `https://skos.ii.uni.wroc.pl/mod/resource/view.php?id=15325`. Jest to dump bazy `stackexchange.org` poświęconej dyskusjom na tematy związane ze historią matematyki i nauki.

Zachęcam do korzystania przede wszystkim z dokumentacji PostgreSQL: `https://www.postgresql.org/docs/11/index.html`.

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>

-- Zadanie 2
<zapytanie>
```

Do zdobycia są 2 punkty, po 0,5 punkta za zadanie.

Zadanie 1 Wypisz wszystkie daty utworzenia postów, których treść (*body*) zawiera słowo *Turing*. Wyniki posortuj wg dat od najnowszych.

```
SELECT creationdate::date FROM posts
WHERE body LIKE '%Turing%'
ORDER BY creationdate DESC;
```

Zadanie 2 Wypisz `id` oraz `title` postów, które

- zostały napisane po 10 października 2018,
- w miesiącach od września do grudnia włącznie.
- ich `title` nie jest nullem,
- ich `score` jest nie niższy niż 9.

Wyniki posortuj alfabetycznie wg tytułów.

```
SELECT id, title FROM posts
WHERE
 creationdate > '2018-10-10' AND
 EXTRACT (MONTH FROM creationdate) BETWEEN 9 AND 12 AND
```

```
 title IS NOT NULL AND
 score >= 9
ORDER by title;
```

Zadanie 3 Wypisz `displayname` oraz `reputation` użytkowników, którzy są właścicielami posta spełniającego oba poniższe warunki:

- w treści posta występuje fragment `deterministic`,
- do posta napisano komentarz, w którego tekście występuje fragment `deterministic`.

Zadbaj, aby wyniki się nie powtarzały oraz były posortowane malejąco wg reputacji.

```
SELECT distinct u.displayname, u.reputation FROM
         users u JOIN
         posts p ON (u.id=p.owneruserid) JOIN
         comments c ON (p.id = c.postid)
WHERE p.body LIKE '%deterministic%' AND
      c.text LIKE '%deterministic%'
ORDER BY reputation DESC;
```

Zadanie 4 Wypisz `displayname` osób, które nigdy nie napisały żadnego komentarza ale napisały jakiś post. Zadbaj, aby wyniki się nie powtarzały oraz były posortowane alfabetycznie. Wypisz tylko pierwsze 10 krotek.

```
SELECT DISTINCT u.displayname FROM
      users u LEFT JOIN
      comments c on (u.id=c.userid) JOIN
      posts p ON (p.owneruserid=u.id)
WHERE c.id IS NULL
ORDER BY u.displayname
LIMIT 10;
```

Wśród osób, które postanowiły użyć `EXCEPT` popularnym błędem było liczenie różnicy względem atrybutu `users.displayname`. Niestety ten atrybut nie posiada cech klucza–w szczególności nasza baza zawiera przykłady różnych użytkowników z taką samą wartością tego atrybutu. Operator rożnicy może więc niepoprawnie usunąć z wyniku `displayname` użytkownika jeśli tylko istnieje jakikolwiek inny użytkownik o tym samym `displayname`, który napisał komentarz. Istotne więc było aby różnicę obliczać względem atrybutu `users.id`. Przykład poprawnego rozwiązania z wykorzystaniem różnicy zbiorów:

```
SELECT DISTINCT t.displayname FROM                  -- DISTINCT konieczne!
        ((SELECT u.id, u.displayname FROM           -- u.id konieczne!
           users u JOIN posts p ON u.id = p.owneruserid)
```

```
        EXCEPT
        (SELECT u.id, u.displayname FROM            -- u.id konieczne!
         users u JOIN comments c ON u.id = c.userid))
        AS t
ORDER BY t.displayname
LIMIT 10;
```

Przy okazji zauważcie, że `DISTINCT` jest konieczny ponieważ choć to prawda, że `EXCEPT` usuwa duplikaty to jednak przetwarza pary `id, displayname`.