Ciekawostki

Rzutowanie na int

Zadanie 1. Napisz funkcję *cast_ int(varchar)*, która rzutuje podany argument na int lub zwraca NULL jeśli rzutowanie nie jest możliwe. Nie powinien pojawiać się błąd.

Przykład

```
select cast_int('14');
```

zwraca

```
 cast_int
----------
       14
(1 row)
```

Przykład

```
select cast_int('x');
```

zwraca

```
 cast_int
----------

(1 row)
```

Zamiana krotki na tablicę

Zadanie 2. Napisz zapytanie, które z każdej krotki tabeli *tab* utworzy tablicę z tekstową reprezentacją krotki. Rozwiązanie powinno zwracać jedną kolumnę i być niezależne od atrybutów tabeli *tab*.

Przykład

Dla tabeli *tab* z pliku init02.sql wynik zapytania powinien być następujący:

```
          wynik
-------------------------
 {1,Anna,Kowalska}
 {2,Jan,Nowak}
 {3,Andrzej,Chmielnicki}
(3 rows)
```

Operacje na nullach

Zadanie 3. Napisz funkcję *nulls* przyjmującą dowolną liczbę argumentów tego samego typu i zwracającą liczbę argumentów o wartości NULL.

Przykład

Poniższa tabela akceptuje tylko krotki z co najwyżej jedną wartością NULL.

```
CREATE TABLE test (
  a int, b int, c int,
  CHECK (nulls(a,b,c) <= 1)
);
```

Powtarzające się krotki

Zadanie 4. Napisz funkcję *remove_duplicates(text)*, która dla zadanej na wejściu tabeli usuwa z niej powtarzające się krotki. W wyniku działania funkcji w tabeli powinien zostać dokładnie jeden reprezentant każdej krotki. Zabronione jest usuwanie tabel.

Przykład

Dla tabeli z pliku init04.sql zapytanie

```
SELECT remove_duplicates('tab');
SELECT * FROM tab;
```

powinno zwrócić wynik

```
 a | b
---+---
 1 | 2
 2 | 3
 1 | 4
 3 | 3
 2 | 2
 2 | 4
(6 rows)
```

Przecięcie tablic

Zadanie 5. Napisz funkcję *array_intersect*, która otrzymuje na wejściu dwie dowolne tablice tego samego typu i zwraca ich przecięcie.

Przykład 1

```
SELECT array_intersect(array[1,2,3,4],array[2,3,1,1,5,5]);
```

zwraca

```
 array_intersect
-----------------
 {1,2,3}
(1 row)
```

Przykład 2

```
SELECT array_intersect(array['a','b','a','e','d'],array['a','c','d']);
```

zwraca

```
 array_intersect
-----------------
 {a,d}
(1 row)
```

Uniwersalne sortowanie

Zadanie 6. Napisz funkcję *array_sort*, która na wejściu otrzymuje dowolną tablicę, którą sortuje i zwraca.

Przykład 1

```
SELECT array_sort(array[1,2,4,3]);
```

zwraca

```
 array_sort
------------
 {1,2,3,4}
(1 row)
```

Przykład 2

```
SELECT array_sort(array['a','v','d']);
```

zwraca

```
 array_sort
------------
 {a,d,v}
(1 row)
```

Automatyczne numerowanie krotek

Zadanie 7. Wykorzystując ctid napisz zapytanie, które dla zadanej tabeli *tab* wypisze wszystkie dane z co trzeciej krotki (poczynając od pierwszej) oraz jej numer.

Przykład

Dla tabeli *tab* z pliku init07.sql wynik zapytania powinien być następujący:

```
 id |   imie   |  nazwisko   | numer
----+----------+-------------+-------
  5 | Anna     | Kowalska    |     1
 20 | Wojciech | Szczepański |     4
 35 | Piotr    | Ptak        |     7
(3 rows)
```

Sortowanie raz jeszcze

Zadanie 8. W pliku init08.sql dana jest tabela *zwierzeta(nazwa varchar(30), rodzaj varchar(20))*. Wypisz wszystkie krotki tej tabeli tak, aby najpierw pojawiły się wszystkie psy, potem wszystkie koty, a następnie wszystkie pozostałe zwierzęta. Drugim warunkiem sortowania jest kolumna *nazwa*. Rozwiązanie powinno składać się z jednego selecta.

Zmiana typu kolumny

Zadanie 9. W pliku init09.sql dana jest tabela *tab(a int, b char(3))*, która w kolumnie *b* dopuszcza tylko wartości 'tak' i 'nie'. Zmień typ tej kolumny na *boolean* tak, aby wartość 'tak' zinterpretować jako True, a wartość 'nie' jako False. Zabronione jest tworzenie pomocniczych tabel lub kolumn.

Meta informacje o tabelach

Zadanie 10. Napisz zapytanie wypisujące nazwy wszystkich tabel ze schematu public.

Zadanie 11. Napisz funkcję *remove_all* usuwającą wszystkie tabele ze schematu public, a następnie wykonaj ją.

Zadanie 12. Napisz zapytanie, które dla każdej tabeli ze schematu public znajdzie jej rozmiar. Rozwiązanie powinno zwracać kolumny: nazwę tabeli i oraz jej rozmiar wyrażony w bajtach.