Funkcje agregujące

Po wykonaniu grupowania rekordów, można przeprowadzić dodatkowo filtrowanie grup (klauzula HAVING) usuwając rekordy nie spełniające określonego warunku. Warunek przy tym musi posiadać wyrażenie agregujące lub grupujące.

Przykład: Podaj identyfikatory i średnie płace podstawowe w zespołach zatrudniających nie mniej niż 3 pracowników:

```
SELECT id_zesp, AVG(placa_pod)
FROM pracownicy
GROUP BY id_zesp
HAVING COUNT(*) >= 3;
```

Funkcje agregujące

UWAGA 1! Przy wykonywaniu zapytań, których wynikiem będzie jeden rekord w klauzuli SELECT nie wolno używać innych wyrażeń jak agregujących. Podobnie w przypadku wyników składających się z wielu rekordów w klauzuli SELECT nie wolno używać innych wyrażeń jak agregujących lub występujących po wyrażeniu GROUP BY:

```
SELECT etat, SUM(placa_pod)
FROM pracownicy WHERE etat = 'PROFESOR';
```

```
SELECT id_zesp, nazwisko, SUM(placa_pod)
FROM pracownicy GROUP BY id_zesp;
```

Funkcje agregujące

UWAGA 2! W klauzuli WHERE nie wolno umieszczać funkcji agregujących, natomiast w klauzuli HAVING można umieszczać tylko funkcje agregujące lub wyrażenia grupujące:

```
SELECT id_zesp FROM pracownicy
WHERE COUNT(*) > 3 GROUP BY id_zesp;
```

```
SELECT id_zesp, COUNT(placa_dod)
FROM pracownicy GROUP BY id_zesp
HAVING etat = 'PROFESOR';
```

Funkcje agregujące

UWAGA 3! Wyniki zapytań z wieloma grupami można porządkować jedynie za pomocą wyrażenia grupującego lub funkcji agregującej:

```
SELECT id_zesp, COUNT(placa_dod)
FROM pracownicy GROUP BY id_zesp
ORDER BY nazwisko;
```

Funkcje agregujące

Zadanie: Znajdź maksymalną sumę płac pracowników w poszczególnych zespołach.

```
SELECT MAX(SUM(placa_pod))
FROM pracownicy GROUP BY id_zesp;
```

Zadanie: Podaj wartość średniej płacy pracowników, zespołów 10, 20 i 30, ale tylko wówczas jeżeli liczba pracowników jest większa od 12.

```
SELECT AVG(placa_pod)
FROM pracownicy
WHERE id_zesp in (10,20,30)
HAVING COUNT(*) > 12;
```

Funkcje agregujące

Zadanie: Dla każdego zespołu, w którym średnia płaca podstawowa przekracza 1000 zł, podaj liczbę zatrudnionych pracowników, pomiń przy tym pracowników na etacie PROFESOR. Wyniki uporządkuj ze względu na sumę płac podstawowych.

```
SELECT id_zesp, COUNT(*)
FROM pracownicy
WHERE etat <> 'PROFESOR'
GROUP BY id_zesp
HAVING AVG(placa_pod) < 1000
ORDER BY SUM(placa_pod);
```

Funkcje agregujące

Zadanie: Wyświetl najniższą i najwyższą pensję oraz różnicę dzielącą najlepiej i najgorzej zarabiających pracowników.

```
SELECT MIN(placa_pod), MAX(placa_pod),
MAX(placa_pod)-MIN(placa_pod)
FROM pracownicy;
```

Zadanie: Wyświetl średnie pensje dla wszystkich etatów. Wyniki uporządkuj wg malejącej średniej pensji.

```
SELECT etat, AVG(placa_pod)
FROM pracownicy
GROUP BY etat
ORDER BY AVG(placa_pod) DESC
```

Funkcje agregujące

Zadanie: Wyświetl liczbę zatrudnionych PROFESORÓW.

```
SELECT COUNT(*)
FROM pracownicy
WHERE etat = 'PROFESOR';
```

Zadanie: Znajdź sumaryczne miesięczne płace dla każdego zespołu. Nie zapomnij o płacach dodatkowych. Wynik posortuj wg numeru zespołu.

```
SELECT id_zesp, SUM(placa_pod + nvl(placa_dod,0))
FROM pracownicy GROUP BY id_zesp
ORDER BY id_zesp;
```

Funkcje agregujące

Zadanie: Wyświetl numery zespołów, które zatrudniają więcej niż dwóch pracowników. Pomiń pracowników bez przydziału do zespołu. Wyniki uporządkuj wg malejącej liczby pracowników.

```
SELECT id_zesp, COUNT(*)
FROM pracownicy
WHERE id_zesp IS NOT NULL
GROUP BY id_zesp
HAVING COUNT(*) < 2
ORDER BY COUNT(*) desc;
```

Funkcje agregujące

Zadanie: Dla każdego pracownika wyświetl pensję najgorzej zarabiającego podwładnego. Wyniki uporządkuj wg malejącej pensji.

```
SELECT id_szefa, MIN(placa_pod)
FROM pracownicy
GROUP BY id_szefa
ORDER BY MIN(placa_pod) DESC;
```

Zadanie: Sprawdź, czy identyfikatory pracowników są unikalne.

```
SELECT id_prac FROM pracownicy
GROUP BY id_prac
HAVING COUNT(*) > 1;
```