

Ben Forta

# SQL

## W MGNIENIU OKA

OPANUJ JĘZYK ZAPYTAŃ W **10** MINUT DZIENNE



Wydanie V

Helion 

Tytuł oryginału: SQL in 10 Minutes a Day, Sams Teach Yourself (5th Edition)

Tłumaczenie: Tomasz Walczak z wykorzystaniem fragmentów książki „SQL w mgnieniu oka. Opanuj język zapytań w 10 minut dziennie. Wydanie IV” w tłumaczeniu Rafała Jończy

ISBN: 978-83-283-9490-2

Projekt okładki: Studio Gravite / Olsztyn

Obarek, Pokoński, Pazdrijowski, Zapucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock Images LLC.

Authorized translation from the English language edition, entitled SQL IN 10 MINUTES A DAY, SAMS TEACH YOURSELF, 5th Edition by FORTA, BEN, published by Pearson Education, Inc, publishing as Sams Publishing, Copyright © 2020 by Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc. POLISH language edition published by Helion S.A., Copyright © 2021, 2022.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<https://ftp.helion.pl/przyklady/sqlo5v.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/sqlo5v>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

# Spis treści

O autorze.....	9
Podziękowania.....	10
Wprowadzenie.....	11
<b>Rozdział 1. Podstawy języka SQL .....</b>	<b>15</b>
Podstawy baz danych .....	15
Bazy danych .....	16
Tabele .....	16
Kolumny i typy danych .....	17
Wiersze .....	18
Klucze główne .....	19
Język SQL .....	20
Ćwicz .....	20
Podsumowanie .....	22
<b>Rozdział 2. Pobieranie danych .....</b>	<b>23</b>
Instrukcja SELECT .....	23
Pobieranie pojedynczych kolumn .....	24
Pobieranie wielu kolumn .....	25
Pobieranie wszystkich kolumn .....	27
Pobieranie jedynie unikatowych wierszy .....	28
Ograniczenie liczby zwracanych wyników .....	29
Komentarze .....	32
Podsumowanie .....	33
Zadania .....	34
<b>Rozdział 3. Sortowanie pobranych danych .....</b>	<b>35</b>
Sortowanie danych .....	35
Sortowanie na podstawie wielu kolumn .....	37
Sortowanie na podstawie położenia kolumny .....	38

	Określenie kierunku sortowania .....	39
	Podsumowanie .....	41
	Zadania .....	41
<b>Rozdział 4.</b>	<b>Filtrowanie danych .....</b>	<b>43</b>
	Stosowanie klauzuli WHERE .....	43
	Operatory klauzuli WHERE .....	44
	Sprawdzanie pod kątem jednej wartości .....	45
	Pobieranie niepasujących danych .....	46
	Sprawdzanie zakresu wartości .....	47
	Sprawdzanie braku wartości .....	47
	Podsumowanie .....	49
	Zadania .....	49
<b>Rozdział 5.</b>	<b>Zaawansowane filtrowanie danych .....</b>	<b>51</b>
	Łączenie klauzul WHERE .....	51
	Używanie operatora AND .....	51
	Używanie operatora OR .....	52
	Kolejność wykonywania działań .....	53
	Operator IN .....	55
	Operator NOT .....	57
	Podsumowanie .....	58
	Zadania .....	58
<b>Rozdział 6.</b>	<b>Filtrowanie za pomocą znaków wieloznacznych .....</b>	<b>61</b>
	Korzystanie z operatora LIKE .....	61
	Znak procentu (%) .....	62
	Znak podkreślenia ( _ ) .....	64
	Znaki nawiasów kwadratowych .....	65
	Wskazówki dotyczące używania znaków wieloznacznych .....	67
	Podsumowanie .....	67
	Zadania .....	67
<b>Rozdział 7.</b>	<b>Tworzenie pól obliczanych .....</b>	<b>69</b>
	Pola obliczane .....	69
	Konkatenacja pól .....	70
	Stosowanie aliasów .....	73
	Przeprowadzanie obliczeń matematycznych .....	75
	Podsumowanie .....	76
	Zadania .....	77

<b>Rozdział 8. Modyfikacja danych za pomocą funkcji</b> .....	79
Czym są funkcje .....	79
Problem z funkcjami .....	79
Stosowanie funkcji .....	80
Funkcje tekstowe .....	81
Funkcje daty i czasu .....	83
Funkcje numeryczne .....	86
Podsumowanie .....	87
Zadania .....	87
<b>Rozdział 9. Funkcje agregujące</b> .....	89
Funkcje agregujące .....	89
Funkcja AVG() .....	90
Funkcja COUNT() .....	91
Funkcja MAX() .....	92
Funkcja MIN() .....	93
Funkcja SUM() .....	94
Agregacja tylko unikatowych wartości .....	95
Łączenie funkcji agregujących .....	97
Podsumowanie .....	97
Zadania .....	97
<b>Rozdział 10. Grupowanie danych</b> .....	99
Omówienie grupowania danych .....	99
Tworzenie grup .....	100
Filtrowanie grup .....	101
Grupowanie i sortowanie .....	104
Kolejność klauzul instrukcji SELECT .....	106
Podsumowanie .....	106
Zadania .....	107
<b>Rozdział 11. Zapytania zagnieżdżone</b> .....	109
Zapytania zagnieżdżone .....	109
Filtrowanie na podstawie zapytań zagnieżdżonych .....	109
Zapytania zagnieżdżone jako pola obliczane .....	113
Podsumowanie .....	116
Zadania .....	116

<b>Rozdział 12. Złączanie tabel</b> .....	117
Czym są złączenia? .....	117
Relacyjne bazy danych .....	117
Po co używać złączeń? .....	119
Tworzenie złączeń .....	119
Znaczenie klauzuli WHERE .....	121
Złączenia wewnętrzne .....	123
Złączanie wielu tabel .....	124
Podsumowanie .....	126
Zadania .....	126
<b>Rozdział 13. Tworzenie zaawansowanych złączeń</b> .....	129
Stosowanie aliasów tabel .....	129
Używanie innych typów złączeń .....	130
Tworzenie złączeń własnych .....	130
Złączenia naturalne .....	132
Złączenia zewnętrzne .....	133
Złączenia i funkcje agregujące .....	135
Złączenia i ich warunki .....	137
Podsumowanie .....	137
Zadania .....	137
<b>Rozdział 14. Łączenie zapytań</b> .....	139
Łączenie zapytań .....	139
Tworzenie unii .....	139
Stosowanie operatora UNION .....	140
Zasady stosowania unii .....	142
Dołączanie lub eliminowanie zduplikowanych wierszy .....	143
Sortowanie zwróconych połączonych wyników .....	144
Podsumowanie .....	145
Zadania .....	145
<b>Rozdział 15. Wstawianie danych</b> .....	147
Wstawianie danych .....	147
Wstawianie całych wierszy .....	147
Wstawianie niepełnych wierszy .....	150
Wstawianie pobranych danych .....	151
Kopiowanie z jednej tabeli do innej .....	153
Podsumowanie .....	154
Zadania .....	155

<b>Rozdział 16. Aktualizacja i usuwanie danych</b> .....	<b>157</b>
Aktualizacja danych .....	157
Usuwanie danych .....	159
Wskazówki związane z aktualizacją lub usuwaniem danych .....	161
Podsumowanie .....	161
Zadania .....	162
<b>Rozdział 17. Tworzenie i modyfikacja tabel</b> .....	<b>163</b>
Tworzenie tabel .....	163
Tworzenie prostej tabeli .....	164
Wartości NULL .....	165
Podawanie wartości domyślnych .....	167
Aktualizacja tabel .....	168
Usuwanie tabel .....	170
Zmiana nazwy tabeli .....	171
Podsumowanie .....	171
Zadania .....	171
<b>Rozdział 18. Stosowanie perspektyw</b> .....	<b>173</b>
Perspektywy .....	173
Dlaczego warto używać perspektyw .....	174
Zasady tworzenia perspektyw i ich ograniczenia .....	175
Tworzenie perspektyw .....	176
Wykorzystanie perspektyw do upraszczania złożonych złączeń .....	176
Formatowanie zwracanych danych za pomocą perspektyw .....	177
Użycie perspektyw do filtrowania niechcianych danych .....	180
Perspektywy z polami obliczanymi .....	181
Podsumowanie .....	182
Zadania .....	182
<b>Rozdział 19. Korzystanie z procedur składowanych</b> .....	<b>183</b>
Procedury składowane .....	183
Dlaczego warto używać procedur składowanych? .....	184
Wykonywanie procedur składowanych .....	185
Tworzenie procedur składowanych .....	187
Podsumowanie .....	190

Rozdział 20. Zarządzanie transakcjami .....	191
Przetwarzanie transakcji .....	191
Sterowanie transakcjami .....	193
Polecenie ROLLBACK .....	194
Polecenie COMMIT .....	195
Stosowanie punktów kontrolnych .....	196
Podsumowanie .....	198
Rozdział 21. Kursory .....	199
Działanie kursorów .....	199
Praca z kursorami .....	200
Tworzenie kursorów .....	200
Korzystanie z kursora .....	201
Zamykanie kursora .....	203
Podsumowanie .....	204
Rozdział 22. Zaawansowane funkcje języka SQL .....	205
Ograniczenia .....	205
Klucze główne .....	206
Klucze obce .....	207
Zapewnienie unikatowości .....	209
Sprawdzanie ograniczeń .....	210
Omówienie indeksów .....	211
Wyzwalacze .....	213
Bezpieczeństwo baz danych .....	215
Podsumowanie .....	215
Dodatek A Skrypty przykładowych tabel .....	217
Dodatek B Składnia instrukcji w SQL-u .....	223
Dodatek C Typy danych języka SQL .....	229
Dodatek D Słowa kluczowe języka SQL .....	235
Rozwiązania.....	243



# Rozdział 10

## Grupowanie danych

---

*W tym rozdziale opisana jest funkcja grupowania danych, która umożliwia podsumowywanie podzbiorów tabeli. Wprowadzone są też dwie nowe klauzule instrukcji SELECT: GROUP BY i HAVING.*

### Omówienie grupowania danych

W poprzednim rozdziale opisywałem funkcje agregujące języka SQL używane do tworzenia podsumowań danych. Umożliwiały one zliczanie wierszy, obliczanie sumy i średniej, a także znajdowanie wartości największej i najmniejszej. Odbywało się bez potrzeby pobierania wszystkich danych.

Do tej pory obliczenia przeprowadzane były na wszystkich danych w tabeli, lub na danych spełniających warunek określony w klauzuli WHERE. Oto krótkie przypomnienie — poniższy przykład zwraca liczbę wszystkich produktów oferowanych przez dostawcę DLL01:

#### Wejście ▼

---

```
SELECT COUNT(*) AS liczba_prod
FROM Produkty
WHERE dost_id = 'DLL01';
```

---

#### Wyjście ▼

---

```
liczba_prod
-----
4
```

W jaki sposób pobrać liczbę produktów oferowanych przez poszczególnych producentów? Jak uzyskać dane dotyczące tylko tych producentów, którzy oferują jeden produkt lub oferują powyżej 10 produktów?

W takiej sytuacji niezastąpione jest grupowanie. Umożliwia ono podzielenie danych na logiczne zbiory i uruchomienie funkcji agregujących dla każdej z grup osobno.

## Tworzenie grup

Grupy tworzy się za pomocą klauzuli `GROUP BY` w instrukcji `SELECT`. Najłatwiej zrozumieć to na przykładzie:

### Wejście ▼

---

```
SELECT dost_id, COUNT(*) AS liczba_prod
FROM Produkty
GROUP BY dost_id;
```

---

### Wyjście ▼

---

dost_id	liczba_prod
-----	-----
BRS01	3
DLL01	4
FNG01	2

### Analiza ▼

---

Powyzsza instrukcja `SELECT` określa dwie kolumny, `dost_id` (która zawiera identyfikator dostawcy) i `liczba_prod` (która jest polem obliczanym za pomocą funkcji agregującej `COUNT(*)`). Klauzula `GROUP BY` sprawia, że SZBD sortuje wyniki i grupuje je na podstawie `dost_id`. Powoduje to obliczanie `liczba_prod` dla każdego unikatowego `dost_id`, zamiast zbiorowo dla całej tabeli. W ten sposób można się dowiedzieć, iż dostawca BRS01 oferuje 3 produkty, dostawca DLL01 4 produkty, a dostawca FNG01 2 produkty.

Ponieważ pojawiła się klauzula `GROUP BY`, nie trzeba było określać każdej grupy, by poznać jej wartość. Wszystko zostało wykonane automatycznie. Klauzula `GROUP BY` powoduje, że system najpierw grupuje dane, a następnie uruchamia funkcję agregującą osobno dla każdej grupy zamiast dla wszystkich wyników.

Zanim jednak rozpocznie się stosowanie klauzuli `GROUP BY`, warto dokładniej poznać jej działanie:

- ▶ Klauzula `GROUP BY` może zawierać dowolną liczbę kolumn. Umożliwia to tworzenie zagnieżdżonych grup, a tym samym bardziej precyzyjne grupowanie danych.
- ▶ W przypadku używania zagnieżdżonych grup w klauzuli `GROUP BY` dane podsumowywane są dla ostatniej określonej kolumny (dla jej grup). Oznacza to, że wszystkie podane kolumny są przetwarzane razem. Nie jest możliwe uzyskanie danych z poziomu poszczególnych wymienionych kolumn.

- ▶ Wszystkie kolumny wymienione w klauzuli GROUP BY muszą być kolumnami pobieranymi z bazy lub poprawnymi wyrażeniami (ale nie funkcjami agregującymi). Jeśli wyrażenie występuje po SELECT, w takiej samej postaci musi się znaleźć w klauzuli GROUP BY. Nie można tu stosować aliasów.
- ▶ Większość implementacji SQL-a nie dopuszcza, aby w klauzuli GROUP BY znajdowały się kolumny typów danych o zmiennej długości (na przykład pola tekstowe lub memo).
- ▶ Wszystkie kolumny występujące w instrukcji SELECT muszą się także znaleźć w klauzuli GROUP BY (nie dotyczy to funkcji agregujących).
- ▶ Jeśli grupowana kolumna zawiera wiersz z wartością NULL, powstanie osobna grupa o nazwie NULL. Jeśli istnieje kilka wierszy o wartości NULL, zostaną one scalone w jedną grupę.
- ▶ W przypadku występowania dodatkowych klauzuli, funkcja GROUP BY musi się pojawić po klauzuli WHERE, ale przed ORDER BY.

### Klauzula ALL

Pewne implementacje SQL-a (na przykład Microsoft SQL Server) obsługują opcjonalną klauzulę ALL dla GROUP BY. Klauzula ta może posłużyć do zwrócenia wszystkich grup, nawet tych, dla których agregacja spowodowałaby zwrócenie wartości NULL. Szczegółów należy szukać w dokumentacji SZBD.

**Wskazówka**  
Wskazówka

### Określanie kolumn na podstawie ich pozycji

Niektóre implementacje SQL-a umożliwiają podanie w klauzuli GROUP BY pozycji kolumn z listy SELECT. Można wtedy na przykład napisać GROUP BY 2, 1, aby *grupowanie najpierw odbyło się na podstawie drugiej kolumny, a następnie pierwszej*. Choć ten skrótowy zapis jest bardzo kuszący, nie wszystkie implementacje go obsługują, a dodatkowo niesie on ze sobą ryzyko pojawienia się błędów po zmianie kolejności kolumn.

**Ostrzeżenie**  
Ostrzeżenie

## Filtrowanie grup

Poza samą możliwością grupowania danych język SQL oferuje także filtrowanie grup (aby jedne z nich wyświetlać, a inne ignorować). Na przykład można wyświetlić wszystkich klientów, którzy dokonali przynajmniej dwóch zamówień. Takie filtrowanie musi się odbywać na podstawie kompletnych grup, a nie poszczególnych wierszy.

Nie można posłużyć się tutaj klauzulą WHERE opisaną w rozdziale 4., „Filtrowanie danych”, gdyż powoduje ona filtrowanie wierszy. Klauzula WHERE nie wie, czym jest grupowanie.

Jaka jest więc alternatywa dla WHERE? Język SQL udostępnia dodatkową klauzulę HAVING. Jest ona bardzo podobna do WHERE. W zasadzie wszystkie opisane do tej pory techniki filtrowania związane z WHERE mogą zostać także użyte w klauzuli HAVING. Jedyna różnica polega na tym, iż WHERE filtruje wiersze, a HAVING grupy.

### Wskazówka

#### **Klauzula HAVING obsługuje wszystkie operatory klauzuli WHERE**

W rozdziałach 4. i 5. („Zaawansowane filtrowanie danych”) opisałem warunki klauzuli WHERE (z uwzględnieniem znaków wieloznacznych i stosowania wielu operatorów). Wszystkie techniki i opcje, które poznałeś w kontekście klauzuli WHERE, można stosować także w klauzuli HAVING. Składnia jest identyczna, zmienia się tylko słowo kluczowe.

W jaki sposób filtrować grupy? Oto przykład:

#### **Wejście ▼**

```
SELECT kl_id, COUNT(*) AS zamowienia
FROM Zamowienia
GROUP BY kl_id
HAVING COUNT(*) >= 2;
```

#### **Wyjście ▼**

kl_id	zamowienia
1000000001	2

#### **Analiza ▼**

Pierwsze trzy wiersze tej instrukcji SELECT są bardzo podobne do wcześniejszego przykładu z tego rozdziału. W ostatnim wierszu dodana została klauzula HAVING, która przepuszcza tylko te grupy, które posiadają minimum dwa zamówienia — `COUNT(*) >= 2`.

Można się przekonać, iż klauzula WHERE nie przeprowadziłaby poprawnego filtrowania, ponieważ musi ono bazować na wartości z agregacji grupy, a nie na wartościach znajdujących się w poszczególnych wierszach.

**Uwaga**  
Uwaga

### Różnica między HAVING i WHERE

Można na to zagadnienie spojrzeć inaczej. Klauzula WHERE filtruje dane przed grupowaniem, natomiast HAVING po. Jest to ważna różnica — wiersze wyeliminowane przez klauzulę WHERE w ogóle nie zostaną wzięte pod uwagę przy tworzeniu grup. Powoduje to zmianę wartości pól obliczanych, a tym samym zmianę wyświetlanych grup, gdy przeprowadzane jest filtrowanie z użyciem klauzuli HAVING.

Czy więc zachodzi potrzeba jednoczesnego stosowania klauzul WHERE i HAVING w jednym zapytaniu? W pewnych sytuacjach jest to nawet konieczne. Przyjrzyjmy się na przykład, iż poprzednie zapytanie powinno zwrócić dowolnych klientów z więcej niż jednym zamówieniem, ale pod uwagę należy brać tylko ostatnich rok. W takiej sytuacji klauzula WHERE spowoduje przeanalizowanie zamówień tylko z ostatnich 12 miesięcy, a klauzula HAVING wskaże tylko klientów z co najmniej dwoma zamówieniami.

Aby lepiej to zrozumieć, zapoznaj się z następującym przykładem. Zapytanie powoduje wyświetlenie listy dostawców z więcej niż jednym produktem o cenie powyżej 10 złotych.

### Wejście ▼

```
SELECT dost_id, COUNT(*) AS liczba_prod
FROM Produkty
WHERE prod_cena >= 10
GROUP BY dost_id
HAVING COUNT(*) >= 2;
```

### Wyjście ▼

dost_id	liczba_prod
-----	-----
BRS01	3
FNG01	2

### Analiza ▼

Zapytanie to wymaga krótkiego wyjaśnienia. Pierwszy wiersz to prosta instrukcja SELECT z funkcją agregującą — podobnie jak w poprzednich przykładach. Klauzula WHERE filtruje wszystkie wiersze, których prod\_cena jest mniejsza od 10 złotych. Następnie dane są grupowane pod kątem dost\_id, a klauzula HAVING przepuszcza tylko grupy zawierające więcej niż jedno zamówienie. Bez klauzuli WHERE zostałyby pobrane dodatkowy wiersz, ponieważ dostawca DLL01 sprzedaje wszystkie cztery produkty po cenie poniżej 10 złotych.

**Wejście ▼**

```
SELECT dost_id, COUNT(*) AS liczba_prod
FROM Produkty
GROUP BY dost_id
HAVING COUNT(*) >= 2;
```

**Wyjście ▼**

```
dost_id  liczba_prod
-----  -
```

BRS01	3
DLL01	4
FNG01	2

**Uwaga**  
Uwaga**Stosowanie HAVING i WHERE**

Klauzula HAVING jest tak podobna do klauzuli WHERE, że większość SZBD traktuje je identycznie, jeśli nie określono klauzuli GROUP BY. Mimo to warto samemu jednoznacznie rozdzielać obie klauzule, czyli HAVING stosować tylko z GROUP BY, a WHERE używać do standardowego filtrowania na poziomie wierszy.

## Grupowanie i sortowanie

Trzeba zdać sobie sprawę z tego, iż klauzule GROUP BY i ORDER BY są bardzo różne, choć często wykonują podobne zadanie. Tabela 10.1 zawiera różnice występujące między tymi klauzulami.

Tabela 10.1. Klauzule ORDER BY i GROUP BY

ORDER BY	GROUP BY
Sortuje wygenerowane dane wyjściowe.	Grupuje wiersze. Dane wyjściowe nie muszą być jednak posortowane na podstawie grup.
Można stosować dla dowolnych kolumn (także tych, które nie są zwracane).	Można zastosować tylko do zwracanych kolumn lub wyrażeń. Wystąpić muszą wszystkie zwracane kolumny lub wyrażenia.
Stosowanie klauzuli nie jest wymagane.	Stosowanie klauzuli jest konieczne, jeśli używa się kolumn (lub wyrażeń) z funkcjami agregującymi.

Pierwsza różnica wymieniona w tabeli 10.1 jest niezmiernie ważna. Choć nie jest to wymagane w specyfikacji SQL-a, to grupy zazwyczaj są wyświetlane

w sposób posortowany. Co więcej, choć dany system może zawsze sortować grupy według klauzuli GROUP BY, konieczny może okazać się inny sposób sortowania. Choć grupowanie odbywa się w taki, a nie inny sposób (w celu uzyskania zagregowanych wartości dla grup), nie oznacza to jednocześnie, że sortowanie także musi odbywać się w ten sam sposób. Zawsze warto zastosować opcjonalną klauzulę ORDER BY, aby wymusić odpowiednie sortowanie pogrupowanych danych, nawet jeśli sortowanie ma odbywać się identycznie jak na podstawie klauzuli GROUP BY.

### Nie zapominaj o klauzuli ORDER BY

Ogólnie rzecz ujmując, za każdym razem, gdy stosuje się klauzulę GROUP BY, powinno się także stosować klauzulę ORDER BY, gdyż jest to jedyny sposób zapewnienia odpowiedniego sortowania danych. Nigdy nie należy polegać na sortowaniu na podstawie klauzuli GROUP BY.

**Wskazówka**  
Wskazówka

Oto krótki przykład, który demonstruje używanie klauzul GROUP BY i ORDER BY. Przedstawione zapytanie SELECT jest bardzo podobne do poprzednich. Pobiera numery zamówień i liczbę zamawianych elementów dla wszystkich zamówień zawierających minimum trzy elementy.

### Wejście ▼

```
SELECT zam_numer, COUNT(*) AS elementy
FROM ElementyZamowienia
GROUP BY zam_numer
HAVING COUNT(*) >=3;
```

### Wyjście ▼

zam_numer	elementy
-----	-----
20006	3
20007	5
20008	5
20009	3

Aby posortować dane wyjściowe na podstawie liczby zamówionych elementów, wystarczy dodać odpowiednią klauzulę ORDER BY.

### Wejście ▼

```
SELECT zam_numer, COUNT(*) AS elementy
FROM ElementyZamowienia
GROUP BY zam_numer
HAVING COUNT(*) >=3
ORDER BY elementy, zam_numer;
```

**Wyjście ▼**

zam_numer	elementy
20006	3
20009	3
20007	5
20008	5

**Analiza ▼**

W tym przykładzie klauzula `GROUP BY` służy do grupowania danych na podstawie numeru zamówienia (kolumna `zam_numer`), więc funkcja `COUNT(*)` zwraca liczbę elementów w poszczególnych zamówieniach. Klauzula `HAVING` filtruje dane, więc zwracane są tylko zamówienia z więcej niż dwoma elementami. Na końcu wyniki są sortowane za pomocą klauzuli `ORDER BY`.

**Kolejność klauzul instrukcji SELECT**

Nadszedł chyba najlepszy czas na omówienie kolejności występowania klauzul w instrukcji `SELECT`. Tabela 10.2 zawiera poprawną kolejność wszystkich omówionych do tej pory klauzul.

Tabela 10.2. Klauzule instrukcji `SELECT` i ich kolejność

Klauzula	Opis	Wymagane
<code>SELECT</code>	zwracane kolumny lub wyrażenia	tak
<code>FROM</code>	tabele, z których pobierane są dane	tylko wtedy, gdy wymagane są dane z tabel
<code>WHERE</code>	filtrowanie wierszy	nie
<code>GROUP BY</code>	tworzenie grup	tylko do obliczania funkcji agregujących dla grup
<code>HAVING</code>	filtrowanie grup	nie
<code>ORDER BY</code>	sortowanie danych wyjściowych	nie

**Podsumowanie**

W rozdziale 9. „Funkcje agregujące” zapoznałeś się z używaniem funkcji agregujących do wykonywania obliczeń podsumowujących na danych. W tym rozdziale wykorzystasteś klauzulę `GROUP BY` do wykonywania takich obliczeń na grupach



elementów i zwracania wyników dla poszczególnych grup. Klauzula HAVING służy do filtrowania grup. Dodatkowo w rozdziale pojawiło się wyjaśnienie różnic między klauzulami GROUP BY i ORDER BY oraz między WHERE i HAVING.

## Zadania

1. Tabela `ElementyZamowienia` zawiera pojedyncze pozycje z każdego zamówienia. Napisz w SQL-u instrukcję, która zwraca liczbę pozycji (jako pole `zam_pozycje`) dla każdego numeru zamówienia (`zam_numer`). Posortuj wyniki według pola `zam_pozycje`.
2. Napisz w SQL-u instrukcję, która zwraca pole o nazwie `najtanszy_prod`, zawierające najtańszy produkt od każdego producenta (użyj pola `prod_cena` z tabeli `Produkty`). Posortuj wyniki od najtańszego produktu do najdroższego.
3. Ważne jest identyfikowanie najlepszych klientów, dlatego napisz w SQL-u instrukcję, która zwraca numery (`zam_numer` z tabeli `ElementyZamowienia`) wszystkich zamówień obejmujących przynajmniej 100 elementów.
4. Inny sposób ustalania najlepszych klientów polega na uwzględnieniu ich wydatków. Napisz w SQL-u instrukcję zwracającą numery (`zam_numer` z tabeli `ElementyZamowienia`) wszystkich zamówień, w których łączna cena wyniosła przynajmniej 1000. Wskazówka: w tym zadaniu musisz obliczyć sumaryczną cenę produktów (`cena_elem` mnożone przez `ilosc`). Posortuj wyniki według numerów zamówień.
5. Jaki błąd znajduje się w poniższej instrukcji w SQL-u? Spróbuj to stwierdzić bez uruchamiania kodu.

```
SELECT zam_numer, COUNT(*) AS elementy
FROM ElementyZamowienia
GROUP BY elementy
HAVING COUNT(*) >= 3
ORDER BY elementy, zam_numer;
```



# PROGRAM PARTNERSKI

— GRUPY HELION —

- 
1. ZAREJESTRUJ SIĘ
  2. PREZENTUJ KSIĄŻKI
  3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA  
**Helion**

## SQL jest najważniejszym narzędziem pracy na bazach danych.

Profesjonalny twórca aplikacji bazodanowych czy administrator baz danych nie mógłby wykonywać swoich obowiązków bez sumiennego zgłębienia niuansów tego języka. Zdobyć takiej wiedzy kosztuje sporo wysiłku. Niemniej już podstawowa znajomość SQL przynosi wiele korzyści każdemu programiście, twórcy aplikacji internetowych i mobilnych, a nawet nieco bardziej zaawansowanemu użytkownikowi pakietów biurowych. Szczęśliwie się składa, że solidne opanowanie podstaw SQL nie wymaga wielkiego trudu — wystarczy spędzić z tą książką 10 minut dziennie!

**Oto piąte, zaktualizowane wydanie świetnego podręcznika**, dzięki któremu niemal bez wysiłku nauczysz się podstaw języka SQL. Książka została podzielona na 22 rozdziały — lektura jednego z nich nie powinna Ci zająć więcej niż 10 minut. Najpierw pokazano, w jaki sposób dokonać prostego pobierania danych, a później przedstawiono nieco bardziej zaawansowane zagadnienia, takie jak złączenia, zapytania zagnieżdżone, tworzenie procedur składowanych, wykorzystanie kursorów, wyzwalaczy i ograniczeń tabel. Układ treści ułatwia systematyczne zapoznawanie się z materiałem, przy czym stopień trudności stopniowo wzrasta. Zdobytą wiedzę można przetestować w zadaniach, które znajdują się w rozdziałach 2 – 18.

### Dzięki tej książce nauczysz się:

- stosować najważniejsze instrukcje SQL
- tworzyć złożone zapytania SQL z użyciem wielu klauzul i operatorów
- pobierać, sortować, filtrować i formatować zawartość baz danych
- używać funkcji agregujących i złączać powiązane tabele
- wstawiać, modyfikować i usuwać dane
- korzystać z perspektyw, procedur składowanych itd.

## 10 minut z SQL. Najbardziej pożyteczne 10 minut w ciągu dnia!

**Ben Forta** jest dyrektorem do spraw inicjatyw edukacyjnych w Adobe Systems oraz autorem ponad 40 popularnych książek dotyczących między innymi języka SQL i wyrażeń regularnych, w tym kilku bestsellerów. Od ponad 30 lat zajmuje się przemysłem komputerowym, rozwojem, marketingiem i szkoleniami. Mieszka z rodziną w Oak Park w stanie Michigan.

	<i>Sprawdź nasze szkolenia!</i>	<b>KOD KORZYŚCI</b> Sięgnij po więcej! 	
 <a href="http://helion.pl">helion.pl</a>		ISBN 978-83-283-9490-2	
 <b>HELION SA</b> ul. Kościuszki 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	<b>AKADEMIA IT &amp; BUSINESS</b>		
<b>HELIONSZKOLENIA.PL</b>		9 788328 394902	
<b>INFORMATYKA W NAJLEPSZYM WYDANIU</b>		Cena: 59,00 zł	