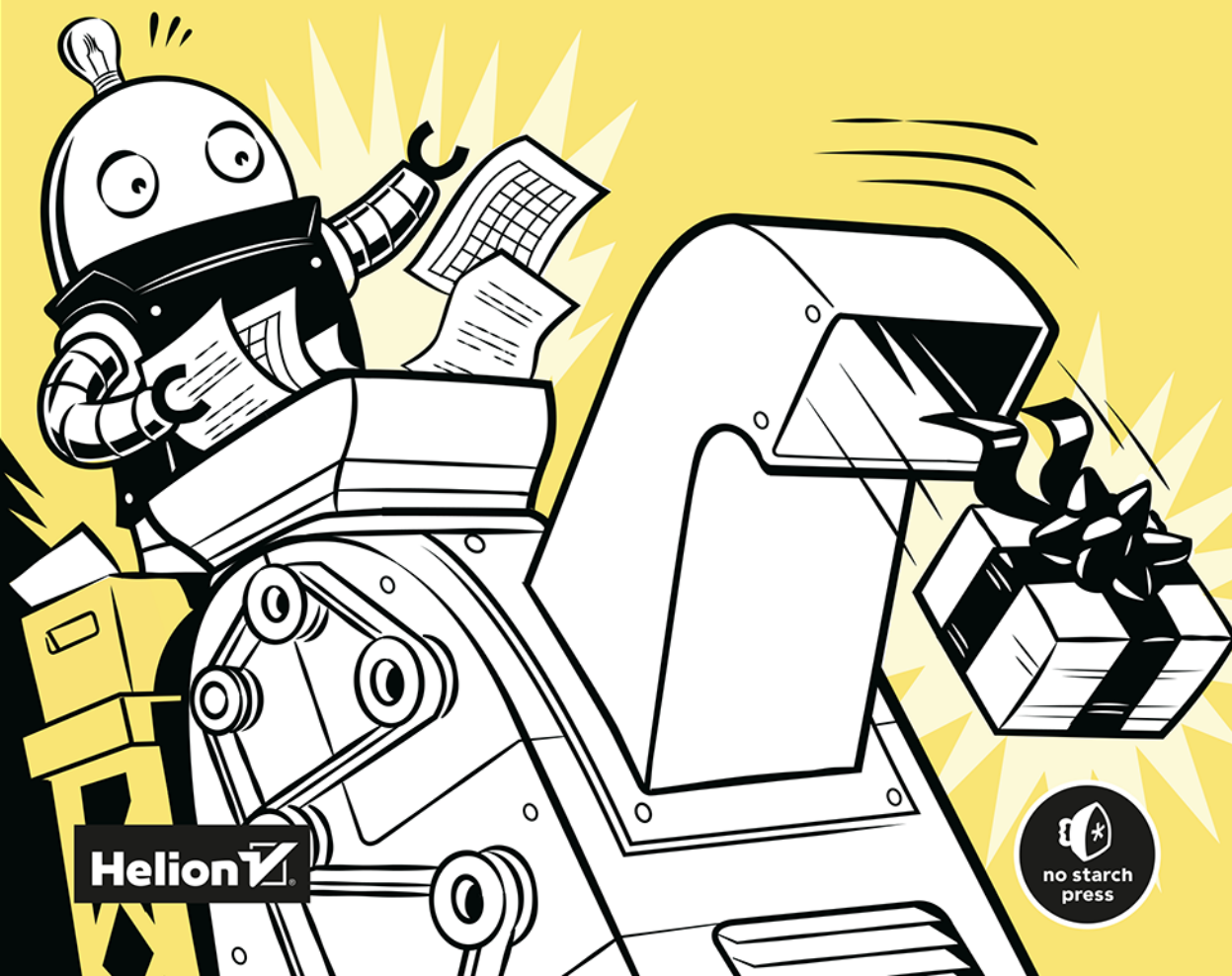


WYDANIE II

SQL W PRAKTYCE

JAK DZIĘKI DANYM
UZYSKIWAĆ CENNE INFORMACJE

ANTHONY DeBARROS



Helion



Tytuł oryginału: Practical SQL: A Beginner's Guide to Storytelling with Data, 2nd Edition

Tłumaczenie: Piotr Pilch

ISBN: 978-83-289-1018-8

Copyright © 2022 by Anthony DeBarros. Title of English-language original: *Practical SQL: A Beginner's Guide to Storytelling with Data, 2nd Edition* ISBN 9781718501065, published by No Starch Press Inc. 245 8th Street, San Francisco, California United States 94103.

The Polish-language 2nd edition Copyright © 2024 by Helion S.A. under license by No Starch Press Inc. All rights reserved.

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 the Publisher.

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.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/sqlpr2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<https://ftp.helion.pl/przyklady/sqlpr2.zip>

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: helion@helion.pl

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

Printed in Poland.

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

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

Spis treści

PRZEDMOWA DO WYDANIA DRUGIEGO	17
WPROWADZENIE	21
1	
KONFIGUROWANIE ŚRODOWISKA PROGRAMISTYCZNEGO	29
Instalowanie edytora tekstu	30
Pobieranie kodu i danych przykładów z serwisu GitHub	31
Instalowanie systemu bazy danych PostgreSQL i narzędzia pgAdmin	32
Instalacja w systemie Windows	32
Instalacja w systemie macOS	36
Instalacja w systemie Linux	38
Użycie narzędzia pgAdmin	40
Uruchamianie narzędzia pgAdmin i ustawianie hasła głównego	41
Łączenie się z domyślną bazą danych postgres	42
Zaznajomienie się z narzędziem Query Tool	44
Dostosowywanie narzędzia pgAdmin	45
Alternatywy dla narzędzia pgAdmin	46
Podsumowanie	46
2	
TWORZENIE PIERWSZEJ BAZY DANYCH I TABELI	47
Tabele	48
Tworzenie bazy danych	49
Uruchamianie kodu SQL w narzędziu pgAdmin	50
Łączenie się z bazą danych analysis	52
Tworzenie tabeli	53
Zastosowanie instrukcji CREATE TABLE	53
Tworzenie tabeli teachers	54
Wstawianie wierszy do tabeli	56
Użycie instrukcji INSERT	56
Wyświetlanie danych	57

Uzyskiwanie pomocy, gdy kod nie działa poprawnie	58
Formatowanie kodu SQL pod kątem czytelności	58
Podsumowanie	59

3

ROZPOCZĘCIE EKSPLOWANIA DANYCH ZA POMOCĄ INSTRUKCJI SELECT61

Podstawowa składnia instrukcji SELECT	62
Tworzenie zapytania dotyczącego podzbioru kolumn	63
Sortowanie danych za pomocą klauzuli ORDER BY	64
Zastosowanie słowa kluczowego DISTINCT do znajdowania unikalnych wartości	66
Filtrowanie wierszy za pomocą klauzuli WHERE	68
Zastosowanie operatorów LIKE i ILIKE z klauzulą WHERE	70
Łączenie operatorów za pomocą operatorów AND i OR	71
Połączenie wszystkiego ze sobą	73
Podsumowanie	73

4

TYPY DANYCH75

Typy znakowe	76
Typy liczbowe	79
Użycie liczb całkowitych	80
Liczby całkowite automatycznie zwiększane	80
Zastosowanie liczb dziesiętnych	83
Wybieranie swojego liczbowego typu danych	86
Daty i czas	86
Zastosowanie typu danych interval w obliczeniach	89
Formaty JSON i JSONB	90
Użycie różnych typów	91
Przekształcanie typu wartości na inny typ za pomocą funkcji CAST	91
Zastosowanie skróconej notacji funkcji CAST	92
Podsumowanie	93

5

IMPORTOWANIE I EKSPORTOWANIE DANYCH94

Praca z plikami tekstowymi z separatorami	95
Obsługa wierszy nagłówek	96
Ujęcie w cudzysłów wartości kolumny zawierających znaki separatora	96
Użycie instrukcji COPY do importowania danych	97
Importowanie danych spisu ludności opisujących hrabstwa	99
Tworzenie tabeli us_counties_pop_est_2019	100
Kolumny i typy danych zbioru spisu ludności	101
Przeprowadzanie importu zbioru danych spisu ludności za pomocą polecenia COPY	103
Sprawdzanie zaimportowanych danych	104

Importowanie podzbioru kolumn za pomocą polecenia COPY	106
Importowanie podzbioru wierszy za pomocą instrukcji COPY	107
Dodawanie wartości do kolumny w trakcie importowania	108
Zastosowanie instrukcji COPY do eksportowania danych	110
Eksportowanie wszystkich danych	110
Eksportowanie określonych kolumn	111
Eksportowanie wyników zapytania	111
Importowanie i eksportowanie z użyciem narzędzia pgAdmin	112
Podsumowanie	113

6

PODSTAWOWE OPERACJE MATEMATYCZNE I STATYSTYCZNE W JĘZYKU SQL	115
Operatory i funkcje matematyczne	116
Operacje matematyczne i typy danych	116
Dodawanie, odejmowanie i mnożenie	117
Użycie operatorów dzielenia i modulo	118
Zastosowanie wykładników potęgi, pierwiastków kwadratowych i silni	119
Zwracanie uwagi na kolejność operacji	120
Wykonywanie operacji matematycznych względem kolumn tabeli spisu ludności	120
Dodawanie i odejmowanie wartości kolumn	121
Określanie wartości procentowych całości	123
Monitorowanie zmiany procentowej	124
Użycie funkcji agregujących do obliczania średnich i sum	126
Określanie mediany	126
Wyznaczanie mediany za pomocą funkcji percentyla	128
Wyznaczanie mediany i percentyli dla danych spisu ludności	129
Wyznaczanie innych kwantyli przy użyciu funkcji percentyla	129
Znajdowanie mody	131
Podsumowanie	131

7

ŁĄCZENIE TABEL W RELACYJNEJ BAZIE DANYCH	133
Łączenie tabel za pomocą klauzuli JOIN	134
Powiązanie tabel z kolumnami kluczy	134
Uzyskiwanie danych z wielu tabel z użyciem klauzuli JOIN	138
Typy złączeń klauzuli JOIN	139
Złączenie JOIN	141
Złączenia LEFT JOIN i RIGHT JOIN	142
Złączenie FULL OUTER JOIN	143
Złączenie CROSS JOIN	144
Użycie wartości NULL do znajdowania wierszy z brakującymi wartościami	145

Trzy typy relacji między tabelami	147
Relacja jeden do jednego	147
Relacja jeden do wielu	147
Relacja wiele do wielu	147
Wybieranie w złączeniu konkretnych kolumn	148
Upraszczenie składni złączenia JOIN za pomocą aliasów tabel	149
Łączenie wielu tabel	150
Łączenie wyników zapytań za pomocą operatorów zbiorów	151
Operatory UNION i UNION ALL	152
Operatory INTERSECT i EXCEPT	154
Wykonywanie operacji matematycznych względem kolumn złączonych tabel	155
Podsumowanie	157

8

OPTYMALNY PROJEKT TABEL	159
Przestrzeganie konwencji nazewnictwa	160
Ujęcie identyfikatorów w cudzysłów umożliwia użycie różnych wielkości liter	160
Pułapki związane z ujmowaniem identyfikatorów w cudzysłów	161
Wytyczne dotyczące nadawania nazw identyfikatorom	162
Kontrolowanie wartości kolumn za pomocą ograniczeń	163
Klucze główne — naturalne albo zastępcze	163
Klucze obce	171
Metoda automatycznego usuwania powiązanych rekordów z użyciem opcji CASCADE	173
Ograniczenie CHECK	173
Ograniczenie UNIQUE	175
Ograniczenie NOT NULL	176
Metoda usuwania ograniczeń lub późniejszego ich dodawania	176
Przyspieszanie zapytań za pomocą indeksów	177
B-drzewo — domyślny indeks systemu PostgreSQL	178
Kwestie dotyczące stosowania indeksów	181
Podsumowanie	182

9

WYODRĘBNIANIE INFORMACJI PRZEZ GRUPOWANIE I PODSUMOWYWANIE	184
Tworzenie tabel badań dotyczących bibliotek	185
Tworzenie tabeli danych bibliotecznych z roku 2018	185
Tworzenie tabel z danymi dotyczącymi bibliotek z roku 2016 i 2017	187
Eksplorowanie danych dotyczących bibliotek za pomocą funkcji agregujących	188
Ustalanie liczby wierszy i wartości za pomocą funkcji count()	188
Znajdowanie maksymalnych i minimalnych wartości za pomocą funkcji max() i min()	191
Agregowanie danych za pomocą klauzuli GROUP BY	192
Podsumowanie	201

10

SPRAWDZANIE I MODYFIKOWANIE DANYCH 203

Importowanie danych dotyczących producentów mięsa, drobiu i jaj	204
Sprawdzanie zbioru danych	205
Sprawdzanie pod kątem brakujących wartości	207
Sprawdzanie pod kątem niespójnych wartości danych	209
Sprawdzanie pod kątem niepoprawnie sformatowanych wartości za pomocą funkcji length()	209
Modyfikowanie tabel, kolumn i danych	211
Modyfikowanie tabel za pomocą instrukcji ALTER TABLE	212
Modyfikowanie wartości przy użyciu instrukcji UPDATE	213
Wyświetlanie zmodyfikowanych danych za pomocą klauzuli RETURNING	214
Tworzenie tabel zapasowych	215
Przywracanie brakujących wartości kolumn	216
Aktualizowanie wartości w celu zapewnienia spójności	218
Poprawianie kodów pocztowych za pomocą operacji łączenia (konkatenacji)	220
Aktualizowanie wartości między tabelami	221
Usuwanie niepotrzebnych danych	223
Usuwanie wierszy z tabeli	224
Usuwanie kolumny z tabeli	225
Usuwanie tabeli z bazy danych	225
Użycie transakcji do zapisywania lub cofania zmian	226
Zwiększanie wydajności podczas aktualizowania dużych tabel	228
Podsumowanie	230

11

FUNKCJE STATYSTYCZNE JĘZYKA SQL 232

Tworzenie tabeli danych statystycznych spisu ludności	233
Pomiar korelacji za pomocą funkcji corr(Y, X)	235
Sprawdzanie dodatkowych korelacji	238
Predykcja wartości z wykorzystaniem analizy regresji	239
Identyfikowanie efektu zmiennej niezależnej za pomocą współczynnika determinacji (r-kwadrat)	241
Określanie zmienności i odchylenia standardowego	242
Tworzenie rankingów za pomocą języka SQL	243
Tworzenie rankingów przy użyciu funkcji rank() i dense_rank()	244
Tworzenie rankingów w obrębie podgrup za pomocą klauzuli PARTITION BY	245
Obliczanie wartości współczynników do dokonywania sensownych porównań	247
Ustalanie współczynników dla firm z branży turystycznej	247
Wygładzanie niejednorodnych danych	249
Podsumowanie	253

12

PRZETWARZANIE DAT I CZASU 255

Typy danych i funkcje obsługujące daty i godziny	256
Przetwarzanie dat i godzin	257
Wydobywanie składników wartości znacznika czasu	257
Tworzenie wartości dat i godzin za pomocą składników znacznika czasu	259
Uzyskiwanie aktualnej daty i godziny	260
Użycie stref czasowych	261
Znajdowanie ustawienia swojej strefy czasowej	262
Ustawianie strefy czasowej	263
Wykonywanie obliczeń względem dat i godzin	266
Znajdowanie wzorców w danych dotyczących taksówek w Nowym Jorku	266
Znajdowanie wzorców w danych operatora Amtrak	272
Podsumowanie	276

13

ZAAWANSOWANE TECHNIKI TWORZENIA ZAPYTAŃ 277

Zastosowanie podzapytań	278
Filtrowanie za pomocą podzapytań użytych w klauzuli WHERE	278
Tworzenie tabel pochodnych przy użyciu podzapytań	280
Złączanie tabel pochodnych	281
Generowanie kolumn za pomocą podzapytań	283
Wyrażenia podzapytań	284
Zastosowanie podzapytań ze słowem kluczowym LATERAL	286
Zastosowanie wyrażeń CTE	289
Zastosowanie tabel krzyżowych	292
Instalowanie funkcji crosstab()	292
Umieszczanie w tabeli wyników ankiety	293
Zastosowanie tabeli krzyżowej dla odczytów temperatury w mieście	295
Ponowne klasyfikowanie wartości za pomocą instrukcji CASE	297
Zastosowanie instrukcji CASE w wyrażeniu CTE	299
Podsumowanie	300

14

PRZESZUKIWANIE TEKSTU W CELU ZNALEZIENIA WARTOŚCIOWYCH DANYCH 302

Formatowanie tekstu za pomocą funkcji łańcuchowych	303
Formatowanie dotyczące wielkości znaków	303
Informacje o znakach	303
Usuwanie znaków	304
Wydobywanie i zastępowanie znaków	305
Dopasowywanie wzorców tekstowych za pomocą wyrażeń regularnych	305
Notacja wyrażeń regularnych	305
Użycie wyrażeń regularnych z klauzulą WHERE	308

Funkcje z wyrażeniami regularnymi zastępujące lub dzielące tekst	309
Przekształcanie tekstu w dane za pomocą funkcji wyrażen regularnych	311
Wyszukiwanie pełnotekstowe systemu PostgreSQL	324
Typy danych wyszukiwania tekstowego	325
Tworzenie tabeli do wyszukiwania pełnotekstowego	327
Przeszukiwanie tekstu przemówień	328
Tworzenie rankingu dopasowań zapytania według stopnia powiązania	332
Podsumowanie	334

15

ANALIZOWANIE DANYCH PRZESTRZENNYCH

ZA POMOCĄ ROZSZERZENIA POSTGIS 336

Aktywowanie rozszerzenia PostGIS i tworzenie bazy danych przestrzennych	337
Bloki konstrukcyjne danych przestrzennych	338
Dwuwymiarowe elementy geometryczne	338
Dane formatu WKT	340
Projekcje i układy współrzędnych	341
Identyfikator SRID	341
Typy danych rozszerzenia PostGIS	342
Tworzenie obiektów przestrzennych za pomocą funkcji rozszerzenia PostGIS	343
Tworzenie typu geometry za pomocą formatu WKT	343
Tworzenie typu geography za pomocą formatu WKT	344
Zastosowanie funkcji punktów	345
Użycie funkcji obiektu LineString	346
Zastosowanie funkcji wielokątów	346
Analizowanie danych dotyczących targów rolniczych	347
Tworzenie kolumny typu danych geography i wypełnianie jej	348
Dodawanie indeksu danych przestrzennych	349
Znajdowanie lokalizacji geograficznych w danej odległości	350
Określanie odległości między lokalizacjami geograficznymi	352
Znajdowanie najbliższych lokalizacji geograficznych	354
Praca z plikami shapefile spisu ludności	355
Zawartość plików formatu shapefile	355
Ładowanie plików shapefile	356
Eksplorowanie pliku shapefile z danymi spisu ludności hrabstw z roku 2019	359
Sprawdzanie danych demograficznych dla konkretnego dystansu	362
Tworzenie złączeń danych przestrzennych	364
Eksplorowanie danych dotyczących dróg gruntowych i wodnych	365
Złączanie tabel danych spisu ludności dotyczących dróg gruntowych i wodnych	365
Znajdowanie miejsca przecięcia obiektów	367
Podsumowanie	368

16

UŻYCIЕ DANYCH FORMATU JSON	370
Struktura formatu JSON	371
Kiedy warto zastosować format JSON razem z kodem języka SQL?	373
Zastosowanie typów danych json i jsonb	374
Importowanie i indeksowanie danych JSON	375
Użycie operatorów wyodrębniania typów danych json i jsonb	376
Wyodrębnianie wartości klucza	377
Wyodrębnianie elementów tablicy	378
Wyodrębnianie ze ścieżki	380
Ograniczanie i obecność	382
Analizowanie danych o trzęsieniach ziemi	385
Eksplorowanie i ładowanie danych o trzęsieniach ziemi	385
Przetwarzanie czasu wystąpienia trzęsień ziemi	386
Znajdowanie największych i najczęściej zgłaszanych trzęsień ziemi	388
Przekształcanie w dane przestrzenne danych JSON dotyczących trzęsień ziemi	390
Generowanie i przetwarzanie danych formatu JSON	395
Przekształcanie wyników zapytania w dane formatu JSON	395
Dodawanie, aktualizowanie oraz usuwanie kluczy i wartości	397
Zastosowanie funkcji przetwarzających dane JSON	400
Ustalanie długości tablicy	400
Zwracanie elementów tablicy jako wierszy	401
Podsumowanie	402

17

OSZCZĘDZANIE CZASU DZIĘKI WIDOKOM, FUNKCJOM I WYZWALACZOM	405
Zastosowanie widoków do uproszczenia zapytań	406
Tworzenie widoków i wykonywanie dotyczących ich zapytań	407
Tworzenie i odświeżanie widoku zmaterializowanego	410
Wstawianie, aktualizowanie i usuwanie danych za pomocą widoku	412
Tworzenie własnych funkcji i procedur	416
Tworzenie funkcji percent_change()	417
Zastosowanie funkcji percent_change()	418
Aktualizowanie danych za pomocą procedury	419
Zastosowanie w funkcji kodu języka Python	422
Automatyzowanie operacji w bazie danych za pomocą wyzwalaczy	424
Rejestrowanie w tabeli aktualizacji ocen	425
Automatyczne klasyfikowanie temperatur	429
Podsumowanie	431

18

PRACA Z SYSTEMEM POSTGRESQL W TRYBIE WIERWSZA POLECEŃ 433

Konfigurowanie trybu wiersza poleceń dla narzędzia psql	434
Konfiguracja narzędzia psql w systemie Windows	434
Konfiguracja narzędzia psql w systemie macOS	438
Konfiguracja narzędzia psql w systemie Linux	440
Korzystanie z narzędzia psql	441
Uruchamianie narzędzia psql i łączenie się z bazą danych	441
Wykonywanie zapytań języka SQL w oknie narzędzia psql	445
Nawigacja i formatowanie wyników	448
Metapolecenia uzyskujące informacje z bazy danych	451
Importowanie, eksportowanie i użycie plików	452
Dodatkowe narzędzia trybu wiersza poleceń usprawniające realizowanie zadań	456
Dodawanie bazy danych za pomocą narzędzia createdb	456
Ładowanie plików z kształtami za pomocą narzędzia shp2psql	457
Podsumowanie	458

19

UTRZYMANIE BAZY DANYCH 459

Odzyskiwanie niewykorzystywanej przestrzeni za pomocą polecenia VACUUM	460
Monitorowanie rozmiaru tabeli	461
Monitorowanie procesu automatycznego opróżniania	463
Ręczne wykonywanie polecenia VACUUM	464
Ograniczanie rozmiaru tabeli za pomocą polecenia VACUUM FULL	465
Modyfikowanie ustawień serwera	465
Lokalizowanie i edytowanie pliku postgresql.conf	466
Ponowne wczytywanie ustawień przy użyciu polecenia pg_ctl	468
Tworzenie kopii zapasowych i przywracanie bazy danych	469
Użycie narzędzia pg_dump do eksportowania bazy danych lub tabeli	469
Przywracanie danych wyeksportowanej bazy za pomocą narzędzia pg_restore	470
Sprawdzenie dodatkowych opcji tworzenia kopii zapasowych i przywracania	471
Podsumowanie	471

20

UZYSKIWANIE INFORMACJI NA PODSTAWIE DANYCH 473

Zacznij od pytania	474
Dokumentuj realizowany proces	474
Zgromadź swoje dane	475
Brak danych? Zbuduj własną bazę danych	475
Oceń pochodzenie danych	476
Sprawdź dane za pomocą zapytań	477
Skontaktuj się z właścicielem danych	477

Zidentyfikuj kluczowe wskaźniki i trendy pojawiające się z upływem czasu	478
Poproś o wyjaśnienie	480
Poinformuj o swoich obserwacjach	480
Podsumowanie	482

DODATEK

DODATKOWE ZASOBY ZWIĄZANE Z SYSTEMEM POSTGRESQL	483
Środowiska projektowe systemu PostgreSQL	483
Narzędzia i rozszerzenia systemu PostgreSQL	484
Społeczność i wiadomości związane z systemem PostgreSQL	485
Dokumentacja	486

9

Wyodrębnianie informacji przez grupowanie i podsumowywanie



Każdy zbiór danych niesie w sobie informacje tworzące pewną całość, a zadaniem analityka danych jest ich znalezienie. W rozdziale 3. dowiedziałeś się więcej na temat sprawdzania danych za pomocą instrukcji `SELECT`, które umożliwiają sortowanie kolumn, znajdowanie różnych wartości oraz filtrowanie wyników. Zaznajomiłeś się również z fundamentami operacji matematycznych języka SQL, typami danych, projektem tabel oraz złączaniem ich. Dysponując tymi narzędziami w swoim arsenale, możesz zacząć uzyskiwać dodatkowe wnioski przy użyciu *funkcji grupowania i agregujących*, które służą do podsumowywania Twoich danych.

Dzięki podsumowywaniu danych możesz zidentyfikować przydatne informacje, których nie mógłbyś dojrzeć w wyniku samego skanowania wierszy tabeli. W niniejszym rozdziale w ramach przykładu posłużymy się dobrze znanym obiektem, czyli lokalną biblioteką.

Biblioteki pozostają istotną częścią społeczności na całym świecie, ale internet i postępy w dziedzinie technologii związanych z bibliotekami zmieniły sposób, w jaki z nich korzystamy. Na przykład e-booki i dostęp sieciowy do materiałów cyfrowych na dobre zagościły obecnie w bibliotekach razem z książkami i periodykami.

W Stanach Zjednoczonych instytut IMLS (Institute of Museum and Library Services) w ramach corocznych badań Public Libraries Survey prowadzi pomiar aktywności związanej z bibliotekami. Podczas tych badań gromadzi się dane z około 9000 jednostek administrujących bibliotekami, które w badaniach określono jako agencje zapewniające usługi bibliotekarskie w konkretnej miejscowości. Niektóre agencje to systemy bibliotek hrabstw, inne są częścią okręgów szkół. Dane powiązane z każdą agencją obejmują liczbę oddziałów, pracowników, zbiór książek, godziny funkcjonowania bibliotek w skali roku itp. Instytut IMLS gromadzi dane każdego roku od 1988 r., a ponadto uwzględnia wszystkie publiczne agencje bibliotek zlokalizowane w 50 stanach, a także w Dystrykcie Kolumbia oraz na obszarze takich terytoriów Stanów Zjednoczonych jak Samoa Amerykańskie (więcej na temat programu możesz przeczytać pod adresem <https://www.imls.gov/research-evaluation/data-collection/public-libraries-survey/>).

Na potrzeby realizowanego tutaj ćwiczenia przyjmujemy, że zadaniem analityka, który właśnie otrzymał nową kopię zbioru danych bibliotek, jest utworzenie raportu opisującego trendy wynikające z danych. Zostaną utworzone trzy tabele przechowujące dane z badań przeprowadzonych w latach 2016, 2017 i 2018 (w dostrzeganiu trendów często pomocne jest ocenianie danych z wielu lat). Później w każdej tabeli zostaną podsumowane bardziej interesujące dane, po czym tabele zostaną złączone w celu pokazania, jak wyniki pomiarów zmieniały się z czasem.

Tworzenie tabel badań dotyczących bibliotek

Utwórzmy trzy tabele badań odnoszących się do bibliotek i zaimportujmy dane. Dla każdej kolumny zostaną zastosowane odpowiednie typy danych i ograniczenia, a ponadto tam, gdzie będzie to wskazane, zostaną dodane indeksy. Kod oraz trzy pliki CSV są dostępne w przykładach dołączonych do książki.

Tworzenie tabeli danych bibliotecznych z roku 2018

Zacniemy od utworzenia tabeli dla danych bibliotecznych z roku 2018. W kodzie z listingu 9.1 za pomocą instrukcji `CREATE TABLE` utworzono tabelę `pls_fy2018_libraries` przeznaczoną na zawartość pliku danych systemu bibliotek publicznych z roku budżetowego 2018 pochodzącego z badań Public Libraries Survey. W pliku tym podsumowano dane na poziomie agencji, ustalając liczbę działań podjętych we wszystkich ich punktach, które obejmują biblioteki centralne i oddziałowe oraz biblioteki mobilne. W ramach corocznych badań generowane są dwa dodatkowe pliki, z których nie będziemy korzystać: w pierwszym pliku dokonuje się podsumowania danych na poziomie stanów, a w drugim znajdują się dane dotyczące poszczególnych punktów. W przypadku omawianego ćwiczenia

pliki te są nadmiarowe, ale z informacjami opisującymi dane, jakie one zawierają, możesz zaznajomić się pod adresem https://www.ims.gov/sites/default/files/2018_pls_data_file_documentation.pdf.

Listing 9.1. Tworzenie i wypełnianie tabeli danymi z badań Public Libraries Survey z roku 2018

```
CREATE TABLE pls_fy2018_libraries (  
    stabr text NOT NULL,  
    fscskey text CONSTRAINT fscskey_2018_pkey PRIMARY KEY, ❶  
    libid text NOT NULL,  
    libname text NOT NULL,  
    address text NOT NULL,  
    city text NOT NULL,  
    zip text NOT NULL,  
    --cięcie--  
    longitude numeric(10,7) NOT NULL,  
    latitude numeric(10,7) NOT NULL  
);  
  
COPY pls_fy2018_libraries ❷  
FROM 'C:\TwojKatalog\pls_fy2018_libraries.csv'  
WITH (FORMAT CSV, HEADER);  
  
CREATE INDEX libname_2018_idx ON pls_fy2018_libraries (libname); ❸
```

Dla wygody utworzyłem dla tabel schemat nazewniczy: łańcuch `pls` odnosi się do tabeli badań, łańcuch `fy2018` identyfikuje rok budżetowy, z którego są dane, a łańcuch `libraries` to nazwa konkretnego pliku z badań. Dla uproszczenia wybrałem 47 spośród 166 bardziej odpowiednich kolumn zawartych w oryginalnym pliku z badań, aby wypełnić tabelę `pls_fy2018_libraries`, z wykluczeniem takich danych jak kody objaśniające źródło poszczególnych odpowiedzi. Gdy biblioteka nie przekazała danych, agencja uzyskiwała je za pomocą innych metod. Na potrzeby niniejszego ćwiczenia nie będą jednak potrzebne te informacje.

Dla ułatwienia kod z listingu 9.1 skrócono, na co wskazuje wiersz komentarza `--cięcie--` widoczny w kodzie. Pełna wersja tego kodu znajduje się jednak w przykładach dołączonych do książki.

Po znalezieniu kodu i pliku danych powiązanych z listingiem 9.1 połącz się za pomocą narzędzia pgAdmin ze swoją bazą danych `analysis` i uruchom kod. Pamiętaj o wstawieniu w miejsce łańcucha `C:\TwojKatalog` ścieżki identyfikującej miejsce, w którym zapisałeś plik `pls_fy2018_libraries.csv`.

Najpierw kod tworzy tabelę przy użyciu instrukcji `CREATE TABLE`. Dla kolumny o nazwie `fscskey` ❶ określono ograniczenie klucza głównego. Zgodnie ze słownikiem danych kolumna ta zawiera unikalny kod przypisywany każdej bibliotece. Ze względu na to, że wartość tej kolumny jest unikalna i obecna w każdym wierszu, a ponadto z małym prawdopodobieństwem zostanie zmieniona, może pełnić rolę naturalnego klucza głównego.

Definicja każdej kolumny uwzględnia odpowiedni typ danych oraz ograniczenia `NOT NULL`, w przypadku których w kolumnach nie ma brakujących wartości. Kolumny `startdate` i `enddate` zawierają daty, ale w kodzie jako ich typ danych określono typ `text`. W pliku CSV kolumny te uwzględniają wartości niebędące datami. Wykonywana przez nas operacja

importowania nie powiedzie się, jeśli podejmie się próbę użycia typu danych `date`. W rozdziale 10. dowiesz się, jak poradzić sobie z tego rodzaju sytuacjami. Na razie kolumny w tej postaci są zupełnie wystarczające.

Po utworzeniu tabeli instrukcja `COPY` ❷ importuje przy użyciu podanej przez Ciebie ścieżki dane z pliku CSV o nazwie `pls_fy2018_libraries.csv`. Do kolumny `libname` dodano indeks ❸, aby szybciej zapewnić wyniki dla operacji wyszukiwania konkretnej biblioteki.

Tworzenie tabel z danymi dotyczącymi bibliotek z roku 2016 i 2017

Tworzenie tabel dla danych badań z roku 2016 i 2017 dotyczących bibliotek wymaga wykonania podobnych kroków. Połączyłem kod, aby utworzyć i wypełnić obie tabele (listing 9.2). Zauważ, że ponownie kod w zaprezentowanym listingu został obcięty, ale pełny kod znajduje się w przykładach dołączonych do książki dostępnych do pobrania z serwera FTP o adresie <https://ftp.helion.pl/przyklady/sqlpr2.zip>

Listing 9.2. Tworzenie i wypełnianie tabel danymi z badań Public Libraries Survey z roku 2016 i 2017

```
CREATE TABLE pls_fy2017_libraries (  
    stabr text NOT NULL,  
    fscskey text CONSTRAINT fscskey_17_pkey PRIMARY KEY, ❶  
    libid text NOT NULL,  
    libname text NOT NULL,  
    address text NOT NULL,  
    city text NOT NULL,  
    zip text NOT NULL,  
    --cięcie--  
    longitude numeric(10,7) NOT NULL,  
    latitude numeric(10,7) NOT NULL  
);  
  
CREATE TABLE pls_fy2016_libraries (  
    stabr text NOT NULL,  
    fscskey text CONSTRAINT fscskey_16_pkey PRIMARY KEY,  
    libid text NOT NULL,  
    libname text NOT NULL,  
    address text NOT NULL,  
    city text NOT NULL,  
    zip text NOT NULL,  
    --cięcie--  
    longitude numeric(10,7) NOT NULL,  
    latitude numeric(10,7) NOT NULL  
);  
  
COPY pls_fy2017_libraries ❷  
FROM 'C:\TwojKatalog\pls_fy2017_libraries.csv'  
WITH (FORMAT CSV, HEADER);  
  
COPY pls_fy2016_libraries  
FROM 'C:\TwojKatalog\pls_fy2016_libraries.csv'  
WITH (FORMAT CSV, HEADER);
```

```
CREATE INDEX libname_2017_idx ON pls_fy2017_libraries (libname); ❸  
CREATE INDEX libname_2016_idx ON pls_fy2016_libraries (libname);
```

Dla obu operacji importowania zaktualizuj ścieżki do plików w instrukcjach COPY i uruchom kod.

Najpierw tworzone są dwie tabele, w ramach których ponownie użyto kolumny `fsc-key` ❶ jako klucza głównego. Dalej wykonano polecenia COPY ❷, aby do tabel zaimportować zawartość plików CSV. I wreszcie w obu tabelach utworzono indeks dla kolumny `libname` ❸.

Sprawdzając kod, zauważysz, że trzy tabele mają identyczną strukturę. Większość realizowanych badań będzie zawierać pewną ilość zmian pojawiających się w kolejnych latach, ponieważ twórcy badań przygotowują nowe pytania lub modyfikują już istniejące. Kolumny, które wybrałem na potrzeby tych trzech tabel, są jednak spójne. Dokumentacja z informacjami o kolejnych latach badań jest dostępna pod adresem <https://www.ims.gov/research-evaluation/data-collection/public-libraries-survey/>. Pora zagłębić się w te dane, aby stwierdzić, jakie informacje w sobie zawierają.

Eksplorowanie danych dotyczących bibliotek za pomocą funkcji agregujących

Funkcje agregujące łączą wartości z wielu wierszy, wykonują operację względem tych wartości i zwracają pojedynczy wynik. Jak się dowiedziałeś w rozdziale 6., możesz na przykład za pomocą funkcji agregującej `avg()` zwrócić średnią wartości. Niektóre funkcje agregujące są częścią standardu języka SQL, natomiast pozostałe są powiązane z systemem PostgreSQL oraz innymi systemami zarządzania bazami danych. Większość tych funkcji użytych w tym rozdziale wchodzi w skład standardu języka SQL (pełna lista funkcji agregujących systemu PostgreSQL znajduje się pod adresem <https://www.postgresql.org/docs/current/functions-aggregate.html>).

W niniejszym podrozdziale przeanalizujemy dane biblioteczne przy użyciu agregatów zastosowanych dla pojedynczych i wielu kolumn, a następnie dowiesz się, jak możesz rozszerzyć zakres wykorzystania agregatów przez grupowanie zwracanych przez nie wyników z wartościami z dodatkowych kolumn.

Ustalanie liczby wierszy i wartości za pomocą funkcji `count()`

Po zaimportowaniu zbioru danych pierwszym sensownym krokiem jest upewnienie się, że tabela zawiera oczekiwaną liczbę wierszy. W dokumentacji instytutu IMLS podano, że zaimportowany przez nas plik z danymi z roku 2018 liczy 9261 wierszy. Z kolei pliki z danymi z lat 2017 i 2016 zawierają odpowiednio 9245 i 9252 wiersze. Widoczna różnica prawdopodobnie odzwierciedla otwarcia, zamknięcia lub połączenia bibliotek. Po sprawdzeniu liczby wierszy w tych tabelach wyniki powinny być zgodne z tymi wartościami.

Funkcja agregująca `count()`, która jest częścią standardu ANSI SQL, ułatwia sprawdzenie liczby wierszy oraz umożliwia wykonanie innych operacji zliczających. Jeśli jako

dane wejściowe wprowadzisz znak gwiazdki, tak jak w przypadku wywołania funkcji `count(*)`, znak ten będzie pełnić rolę symbolu wieloznacznego, dlatego funkcja zwróci liczbę wierszy tabeli niezależnie od tego, czy zawierają one wartości NULL. Postąpiono tak w każdej z trzech instrukcji kodu z listingu 9.3.

Listing 9.3. Użycie funkcji `count()` do ustalenia liczby wierszy tabeli

```
SELECT count(*)
FROM pls_fy2018_libraries;

SELECT count(*)
FROM pls_fy2017_libraries;

SELECT count(*)
FROM pls_fy2016_libraries;
```

Wykonaj po jednym na raz każdą z instrukcji zawartych w listingu 9.3, aby ujrzeć liczbę wierszy tabeli. W przypadku tabeli `pls_fy2018_libraries` wynik powinien wyglądać następująco:

```
count
-----
9261
```

Dla tabeli `pls_fy2017_libraries` wynik prezentuje się następująco:

```
count
-----
9245
```

I wreszcie w przypadku tabeli `pls_fy2016_libraries` wynik powinien być taki:

```
count
-----
9252
```

Wszystkie trzy wyniki zawierają oczekiwaną liczbę wierszy. Jest to właściwy pierwszy krok, gdyż w ramach niego zostaniesz ostrzeżony o takich problemach jak brakujące wiersze lub sytuacja, w której mogłeś zaimportować zły plik.

Uwaga

Korzystając z interfejsu narzędzia `pgAdmin`, też możesz sprawdzić liczbę wierszy, ale jest to niewygodne. Kliknięcie prawym przyciskiem myszy nazwy tabeli w przeglądarce obiektów tego narzędzia i wybranie pozycji `All Rows` menu `View/Edit` spowoduje wykonanie zapytania SQL zwracającego wszystkie wiersze. W obrębie panelu wyników zostanie następnie wyświetlony komunikat z liczbą wierszy, który jednak znika po kilku sekundach.

Określanie liczby wartości obecnych w kolumnie

Jeżeli w funkcji `count()` podasz nazwę kolumny zamiast znaku gwiazdki, zwróci ona liczbę wierszy pozbawionych wartości `NULL`. Za pomocą tej funkcji możesz na przykład ustalić liczbę wartości innych niż `NULL` w kolumnie `phone` tabeli `p1s_fy2018_libraries` (listing 9.4).

Listing 9.4. Użycie funkcji `count()` do uzyskania liczby wartości w kolumnie

```
SELECT count(phone)
FROM p1s_fy2018_libraries;
```

Wynik potwierdza, że kolumna `phone` zawiera wartość w przypadku 9261 wierszy, co odpowiada łącznej liczbie wierszy, którą wcześniej podano.

```
count
-----
9261
```

Oznacza to, że w każdym wierszu kolumna `phone` przechowuje wartość. Być może już to podejrzewałeś, biorąc pod uwagę to, że w instrukcji `CREATE TABLE` dla kolumny tej określono ograniczenie `NOT NULL`. Warto jednak przeprowadzić to sprawdzenie, ponieważ brak wartości może mieć wpływ na Twoją decyzję dotyczącą tego, czy w ogóle kontynuować analizę. Aby w pełni zweryfikować dane, zwykle dobrym pomysłem będzie współpraca z ekspertami z danej dziedziny, a także dokładniejsze zagłębienie się w dane. Zalecam skorzystanie z rady ekspertów jako części obszerniejszej metodologii analizy (więcej na ten temat znajdziesz w rozdziale 20.).

Ustalanie liczby różnych wartości w kolumnie

W rozdziale 3. omówiono słowo kluczowe `DISTINCT` będące częścią standardu języka SQL, które w ramach instrukcji `SELECT` powoduje zwrócenie listy unikalnych wartości. Za pomocą tego słowa możesz uzyskać unikalne wartości w pojedynczej kolumnie lub wyświetlić niepowtarzalne kombinacje wartości z wielu kolumn. Słowo kluczowe `DISTINCT` może też zostać dodane do funkcji `count()` w celu zwrócenia przez nią liczby różnych wartości zawartych w kolumnie.

Kod z listingu 9.5 zawiera dwa zapytania. Pierwsze z nich zapewnia liczbę wszystkich wartości w kolumnie `libname` tabeli z danymi z roku 2018. Drugie zapytanie wykonuje to samo działanie, ale przed nazwą kolumny wstawiono w nim słowo kluczowe `DISTINCT`. Wykonaj oba zapytania po jednym na raz.

Listing 9.5. Zastosowanie funkcji `count()` do uzyskania liczby różnych wartości w kolumnie

```
SELECT count(libname)
FROM p1s_fy2018_libraries;

SELECT count(DISTINCT libname)
FROM p1s_fy2018_libraries;
```

Pierwsze zapytanie zwraca liczbę wierszy zgodną z liczbą wierszy w tabeli, którą uzyskano za pomocą kodu z listingu 9.3:

```
count
-----
9261
```

Znakomicie. Oczekuje się, że nazwa agencji bibliotek będzie podana w każdym wierszu. Drugie zapytanie zwraca jednak mniejszą liczbę:

```
count
-----
8478
```

Użycie słowa kluczowego DISTINCT do usunięcia duplikatów ogranicza liczbę unikalnych nazw bibliotek do 8478. Po bliższym przyjrzeniu się danym okazuje się, że w udostępnionych danych badań z 2018 r. 526 agencji bibliotek ma taką samą nazwę jak jedna lub większa liczba agencji. 10 agencji bibliotek nosi nazwę OXFORD PUBLIC LIBRARY, a każda z nich znajduje się w miejscowości o nazwie Oxford zlokalizowanej w różnych stanach, takich między innymi jak Alabama, Connecticut, Kansas i Pensylwania. W punkcie „Agregowanie danych za pomocą klauzuli GROUP BY” utworzymy zapytanie, które pozwoli ujrzeć kombinacje różnych wartości.

Znajdowanie maksymalnych i minimalnych wartości za pomocą funkcji max() i min()

Funkcje max() i min() zapewniają największe i najmniejsze wartości w kolumnie, a ponadto są przydatne z kilku powodów. Po pierwsze ułatwiają zorientowanie się w zakresie raportowanych wartości. Po drugie, jak się okaże, funkcje mogą ujawnić nieoczekiwane problemy z danymi.

Funkcje max() i min() działają w taki sam sposób z nazwą kolumny będącą ich danymi wejściowymi. W kodzie z listingu 9.6 użyto obu tych funkcji względem kolumny visits tabeli z danymi z roku 2018, w której jest rejestrowana liczba rocznych wizyt w agencji bibliotek oraz wszystkich jej oddziałów. Uruchom kod.

Listing 9.6. Znajdowanie największej i najmniejszej liczby wizyt przy użyciu funkcji max() i min()

```
SELECT max(visits), min(visits)
FROM pls_fy2018_libraries;
```

Zapytanie zwraca następujące wyniki:

max	min
-----	---
16686945	-3

Cóż, wyniki są interesujące. Wartość maksymalna wynosząca ponad 16,6 miliona jest sensowna w przypadku dużego systemu miejskich bibliotek, ale liczba -3 jako wartość minimalna? Początkowo można odnieść wrażenie, że taki wynik to pomyłka. Okazuje się jednak, że twórcy badań dotyczących bibliotek używają przy gromadzeniu danych powszechnej, lecz potencjalnie problematycznej konwencji polegającej na umieszczaniu w kolumnie liczby ujemnej lub jakiejś sztucznie zawyżonej wartości w celu wskazania pewnego warunku.

W tym przypadku wartości ujemne w kolumnach liczbowych wskazują następujące rzeczy:

Wartość -1 oznacza brak udzielonej odpowiedzi na pytanie.

Wartość -3 wskazuje status „nienadające się do zastosowania” i jest stosowana, gdy agencja bibliotek została tymczasowo lub trwale zamknięta.

W trakcie eksplorowania danych konieczne będzie uwzględnienie wartości ujemnych i wykluczenie ich, ponieważ sumowanie wartości kolumny wraz z wartościami ujemnymi spowoduje uzyskanie niepoprawnej sumy. Odfiltrowanie tych wartości umożliwi klauzula `WHERE`. Warto o tym pamiętać, aby zawsze zaznajomić się z dokumentacją danych w celu wyeliminowania problemu zawczasu, a nie być zmuszonym do cofania się po poświęceniu mnóstwa czasu na głębszą analizę!

Uwaga

Lepszą alternatywą dla takiego wariantu z wartością ujemną jest zastosowanie wartości `NULL` w kolumnie `visits` wierszy, w przypadku których nie ma danych odpowiedzi, a następnie utworzenie osobnej kolumny `visits_flag` do przechowywania kodów objaśniających powód braku danych.

Agregowanie danych za pomocą klauzuli `GROUP BY`

Gdy używasz klauzuli `GROUP BY` w połączeniu z funkcjami agregującymi, możesz grupować wyniki zgodnie z wartościami znajdującymi się w jednej lub większej liczbie kolumn. Pozwala to wykonywać operacje realizowane za pomocą funkcji `sum()` lub `count()` względem każdego stanu w tabeli lub poszczególnych typów agencji bibliotek.

Wyjaśnijmy, na czym polega stosowanie klauzuli `GROUP BY` z funkcjami agregującymi. Sama w sobie klauzula ta, która stanowi również część standardu ANSI SQL, eliminuje z wyników zduplikowane wartości (podobnie do klauzuli `DISTINCT`). W kodzie z listingu 9.7 zaprezentowano działanie klauzuli `GROUP BY`.

Listing 9.7. Użycie klauzuli `GROUP BY` względem kolumny `stabr`

```
SELECT stabr
FROM pls_fy2018_libraries
GROUP BY stabr ❶
ORDER BY stabr;
```

Klauzulę `GROUP BY` dodano po klauzuli `FROM` ❶ i dołączono nazwę kolumny, dla której ma zostać wykonane grupowanie. W tym przypadku wybrano kolumnę `stabr`, która zawiera skrót nazwy stanu, a ponadto dokonano grupowania według tej kolumny. Dalej zastosowano

też klauzulę ORDER BY stabr, aby wyniki zostały pogrupowane w kolejności alfabetycznej. Dzięki temu uzyska się wynik z unikalnymi skrótami nazw stanów z tabeli z danymi z roku 2018. Oto część wyników:

```
stabr
-----
AK
AL
AR
AS
AZ
CA
--ciącie--
WV
WY
```

Zauważ, że w 55 zwróconych wierszach nie ma żadnych duplikatów. Te standardowe, 2-literowe skrócone kody pocztowe dotyczą 50 stanów oraz miasta Waszyngton w Dystrykcie Kolumbii i kilku terytoriów Stanów Zjednoczonych, takich jak wyspa Guam i Wyspy Dziewicze.

Nie jesteś ograniczony do grupowania tylko jednej kolumny. W kodzie z listingu 9.8 użyto klauzuli GROUP BY względem danych z roku 2018, aby na potrzeby grupowania określić kolumny city i stabr.

Listing 9.8. Użycie klauzuli GROUP BY względem kolumn city i stabr

```
SELECT city, stabr
FROM pls_fy2018_libraries
GROUP BY city, stabr
ORDER BY city, stabr;
```

Wyniki są sortowane według miasta, a następnie z uwzględnieniem stanu. Następujące dane wyjściowe prezentują unikalne kombinacje w takiej kolejności:

```
city      stabr
-----
ABBEVILLE AL
ABBEVILLE LA
ABBEVILLE SC
ABBOTSFORD WI
ABERDEEN  ID
ABERDEEN  SD
ABERNATHY TX
--ciącie--
```

W ramach tego grupowania zwrócono 9013 wierszy, czyli o 248 wierszy mniej, niż wynosi łączna liczba wierszy tabeli. Wynik wskazuje na to, że plik zawiera wiele wierszy, w przypadku których dla konkretnej kombinacji miasta i stanu występuje więcej niż jedna agencja bibliotek.

Łączenie klauzuli GROUP BY z funkcją count()

Jeśli połączy się klauzulę GROUP BY z funkcją agregującą, taką jak count(), z naszych danych można wyodrębnić więcej opisowych informacji. Na przykład wiadomo, że w tabeli danych z roku 2018 figuruje 9261 agencji bibliotek. Możliwe jest uzyskanie liczby agencji według stanu i posortowanie ich w celu stwierdzenia, w których stanach jest ich najwięcej. Kod z listingu 9.9 demonstruje, jak to zrobić.

Listing 9.9. Zastosowanie klauzuli GROUP BY z funkcją count() względem kolumny stabr

```
SELECT stabr, count(*) ❶  
FROM pls_fy2018_libraries  
GROUP BY stabr ❷  
ORDER BY count(*) DESC; ❸
```

Tym razem zapytanie dotyczy wartości w kolumnie stabr oraz liczby wierszy zawierających daną wartość. W obrębie listy kolumn objętych zakresem zapytania ❶ określono kolumnę stabr wraz z funkcją count(), dla której jako dane wejściowe podano znak gwiazdki (spowoduje ona uwzględnienie przez tę funkcję wartości NULL). Ponadto, gdy w ramach funkcji agregującej wybiera się poszczególne kolumny, w klauzuli GROUP BY trzeba określić te same kolumny ❷. W przeciwnym razie baza danych zwróci komunikat o błędzie informujący o konieczności wykonania tego działania, gdyż nie możesz grupować wartości przez agregowanie oraz w tym samym zapytaniu używać niegrupowanych wartości kolumn.

W celu posortowania wyników tak, aby na samej górze został wyświetlony stan z największą liczbą agencji, można skorzystać z klauzuli ORDER BY ❸, która uwzględnia funkcję count() i słowo kluczowe DESC.

Uruchom kod z listingu 9.9. W wynikach widać, że w stanach Nowy Jork, Illinois i Teksas w 2018 r. była największa liczba agencji bibliotek:

stabr	count
NY	756
IL	623
TX	560
IA	544
PA	451
MI	398
WI	381
MA	369

--ciąćcie--

Pamiętaj o tym, że przykładowa tabela reprezentuje agencje bibliotek obsługujące okolice. Samo to, że w stanach Nowy Jork, Illinois i Teksas znajduje się najwięcej agencji bibliotek, nie oznacza, że dysponują one największą liczbą punktów, do których można się udać, aby przejrzeć zawartość półek. Agencja może mieć tylko jedną centralną bibliotekę albo może nie oferować żadnej takiej biblioteki, a jedynie 23 oddziały rozproszone po całym hrabstwie. Aby umożliwić policzenie punktów, każdy wiersz tabeli zawiera również wartości

w kolumnach `centlib` i `branlib`, które rejestrują odpowiednio liczbę bibliotek centralnych i oddziałowych. W celu określenia łącznych liczb należy względem obu kolumn zastosować funkcję agregującą `sum()`.

Zastosowanie klauzuli GROUP BY z funkcją count() względem wielu kolumn

Jeszcze więcej informacji można uzyskać z danych przez połączenie klauzuli GROUP BY z funkcją `count()` w przypadku wielu kolumn. Na przykład kolumna `stataddr` obecna we wszystkich trzech tabelach zawiera kod wskazujący, czy w ciągu ostatniego roku zmienił się adres agencji. Wartości w kolumnie `stataddr` są następujące:

- 00. Brak zmiany w ciągu ostatniego roku.
- 07. Przeniesiono do nowej lokalizacji.
- 15. Niewielka zmiana w adresie.

Kod z listingu 9.10 zawierający klauzulę GROUP BY z kolumnami `stabr` i `stataddr` oraz uwzględniający funkcję `count()` służy do ustalenia liczby agencji w każdym stanie, w przypadku których dokonano przeniesienia w nowe miejsce, wprowadzono drobną zmianę w adresie lub nie dokonano żadnej zmiany.

Listing 9.10. Użycie klauzuli GROUP BY z funkcją count() względem kolumn stabr i stataddr

```
SELECT stabr, stataddr, count(*) ❶  
FROM pls_fy2018_libraries  
GROUP BY stabr, stataddr ❷  
ORDER BY stabr, stataddr; ❸
```

Kluczowe sekcje zapytania to nazwy kolumn i funkcja `count()` umieszczone po instrukcji SELECT ❶. Ważne jest też określenie obu kolumn w klauzuli GROUP BY ❷, tak aby funkcja `count()` zapewniła liczbę unikalnych kombinacji wartości kolumn `stabr` i `stataddr`.

Aby ułatwić analizowanie danych wyjściowych, posortujmy je najpierw w kolejności rosnącej według kodów stanów i kodów statusu adresu ❸. Oto wyniki:

stabr	stataddr	count
AK	00	82
AL	00	220
AL	07	3
AL	15	1
AR	00	58
AR	07	1
AR	15	1
AS	00	1

--ciąćcie--

W kilku pierwszych wierszach widać, że kod 00 (brak zmiany w adresie) to najczęstsza wartość dla każdego stanu. Można było tego oczekiwać, ponieważ bardziej prawdopodobne jest to, że w przypadku większej liczby agencji bibliotek adres nie ulegnie zmianie, niż to, że się zmieni. Otrzymany wynik pomaga w upewnieniu się, że dane są analizowane w staranny sposób. Jeśli kod 07 (przeniesienie do nowej lokalizacji) występowałby najczęściej dla każdego stanu, przywodziłoby to na myśl pytanie o to, czy poprawnie utworzono zapytanie albo czy występuje problem z danymi.

Ponowne użycie funkcji `sum()` do sprawdzania aktywności związanej z bibliotekami

Rozszerzmy stosowane techniki, aby uwzględnić grupowanie i agregowanie dla złączonych tabel zawierających dane dotyczące bibliotek z lat 2016, 2017 i 2018. Naszym celem jest zidentyfikowanie trendów w zakresie odwiedzania bibliotek w okresie obejmującym trzy lata. Aby to zrealizować, niezbędne jest obliczenie całkowitych liczb za pomocą funkcji agregującej `sum()`.

Zanim zagłębimy się w te zapytania, zajmijmy się wartościami -3 i -1, które wskazują odpowiednio status „nienadające się do zastosowania” i brak udzielonej odpowiedzi. Aby uniknąć wpływu na analizę tych wartości ujemnych, zostaną one odfiltrowane za pomocą klauzuli `WHERE` ograniczającej wyniki zapytań do wierszy, w których wartości w kolumnie `visits` są większe lub równe 0.

Zacznijmy od obliczenia sumy uzyskanych z poszczególnych tabel liczb wizyt w bibliotekach z całego roku. Uruchom osobno każdą instrukcję `SELECT` zawartą w kodzie z listingu 9.11.

Listing 9.11. Zastosowanie funkcji agregującej `sum()` do zsumowania liczby wizyt w bibliotekach w latach 2016, 2017 i 2018

```
SELECT sum(visits) AS visits_2018
FROM pls_fy2018_libraries
WHERE visits >= 0;
```

```
SELECT sum(visits) AS visits_2017
FROM pls_fy2017_libraries
WHERE visits >= 0;
```

```
SELECT sum(visits) AS visits_2016
FROM pls_fy2016_libraries
WHERE visits >= 0;
```

W roku 2018 łączna liczba wizyt wyniosła w przybliżeniu 1,29 miliarda:

```
visits_2018
-----
1292348697
```

W roku 2017 całkowita liczba wizyt to w przybliżeniu 1,32 miliarda:

```
visits_2017
-----
1319803999
```

Z kolei w przypadku roku 2016 ustalono łączną liczbę wizyt wynoszącą w przybliżeniu 1,36 miliarda:

```
visits_2016
-----
1355648987
```

Coś tutaj stwierdziliśmy, ale może to nie być dobra wiadomość dla bibliotek. Wydaje się, że trend jest spadkowy z liczbą wizyt zmniejszającą się o około 5% w okresie od roku 2016 do roku 2018.

Udoskonalmy to rozwiązanie. Zapytania sumują wizyty zarejestrowane w każdej tabeli. Na podstawie liczb wierszy ustalonych wcześniej w rozdziale wiadomo, że każda tabela zawiera inną liczbę agencji bibliotek: 9261 agencji w 2018 r., 9245 agencji w 2017 r. oraz 9252 agencje w 2016 r. Różnice wynikają prawdopodobnie z faktu otwierania, zamykania lub łączenia agencji. Określmy zatem, jak suma wizyt będzie się różnić, jeśli analiza zostanie ograniczona do wierszy agencji bibliotek istniejących we wszystkich trzech tabelach i niezawierających w kolumnie `visits` wartości ujemnych. W tym celu należy złączyć tabele za pomocą kodu z listingu 9.12.

Listing 9.12. Użycie funkcji `sum()` do zsumowania liczby wizyt ze złączonych tabel z danymi z lat 2016, 2017 i 2018

```
SELECT sum(pls18.visits) AS visits_2018, ❶
       sum(pls17.visits) AS visits_2017,
       sum(pls16.visits) AS visits_2016
FROM pls_fy2018_libraries pls18 ❷
JOIN pls_fy2017_libraries pls17 ON pls18.fscskey = pls17.fscskey
JOIN pls_fy2016_libraries pls16 ON pls18.fscskey = pls16.fscskey
WHERE pls18.visits >= 0 ❸
      AND pls17.visits >= 0
      AND pls16.visits >= 0;
```

Zapytanie to łączy w sobie kilka zagadnień omówionych we wcześniejszych rozdziałach, w tym złączenia tabel. W kodzie na początku użyto funkcji agregującej `sum()` ❶ w celu zsumowania wartości w kolumnie `visits` każdej z trzech tabel. Podczas złączania tabel za pomocą ich kluczy głównych zadeklarowano aliasy tabel ❷ w sposób objaśniony w rozdziale 7. Pominęto tutaj przed każdym aliasem opcjonalne słowo kluczowe `AS`. Na przykład zadeklarowano nazwę `pls18` jako alias tabeli z danymi z roku 2018, aby uniknąć podawania w obrębie zapytania jej dłuższej, pełnej nazwy.

Zauważ, że użyto standardowego złączenia JOIN znanego również jako złączenie INNER JOIN. Oznacza to, że wyniki zapytania będą zawierać wyłącznie wiersze, w których wartości kolumny klucza głównego fscskey są takie same we wszystkich trzech tabelach.

Tak jak w kodzie z listingu 9.11, za pomocą klauzuli WHERE ❸ określono tutaj, że wynik powinien uwzględniać tylko te wiersze, w których wartość kolumny visits tabel jest większa lub równa 0. W ten sposób zapobiegnie się wpływaniu na sumy sztucznych wartości ujemnych.

Uruchom zapytanie. Wyniki powinny wyglądać następująco:

visits_2018	visits_2017	visits_2016
-----	-----	-----
1278148838	1319325387	1355078384

Wyniki są podobne do tego, co uzyskano po użyciu zapytań kierowanych osobno do każdej tabeli, choć podane sumy dla roku 2018 nie przekraczają 14 milionów. W dalszym ciągu trend spadkowy utrzymuje się.

Aby dysponować pełnym obrazem tego, jak zmienia się kwestia korzystania z bibliotek, wskazane będzie uruchomienie podobnego zapytania względem wszystkich kolumn zawierających wskaźniki wydajności mające na celu rejestrowanie trendu w każdej tabeli. Na przykład kolumna wifisess umożliwia stwierdzenie, ile razy użytkownicy połączyli się z bezprzewodową siecią internetową biblioteki. Jeśli w kodzie z listingu 9.11 użyje się tej kolumny zamiast kolumny visits, zostanie otrzymany następujący wynik:

wifi_2018	wifi_2017	wifi_2016
-----	-----	-----
349767271	311336231	234926102

Wprawdzie liczba wizyt spadła, ale pracownicy bibliotek zauważyli szybki wzrost stopnia korzystania z sieci bezprzewodowej. Fakt ten zapewnia znaczącą informację w odniesieniu do tego, jak zmienia się rola bibliotek.

Uwaga

Choć złączono tabele za pomocą kolumny fscskey, całkowicie możliwe jest to, że niektóre agencje bibliotek pojawiające się we wszystkich trzech tabelach w ciągu trzech lat zostały objęte procesem łączenia lub podziału. Dobrym pomysłem jest skontaktowanie się z instytutem IMLS w celu zapytania o zastrzeżenia towarzyszące analizowaniu tych danych.

Grupowanie sum wizyt według stanów

Gdy już wiadomo, że między rokiem 2016 i 2018 całościowo spadła liczba wizyt w bibliotekach na obszarze Stanów Zjednoczonych, możesz sobie zadać następujące pytanie: „Czy w każdej części hrabstwa zanotowano spadek, czy też stopień trendu spadkowego zmienia się zależnie od regionu?”. Odpowiedź na to pytanie można uzyskać przez zmodyfikowanie

poprzedniego zapytania pod kątem grupowania według kodu stanów. Zastosujemy również obliczenie zmiany procentowej w celu porównania trendu według stanów. Listing 9.13 zawiera pełny kod.

Listing 9.13. Użycie klauzuli *GROUP BY* do monitorowania zmiany procentowej liczby wizyt w bibliotekach poszczególnych stanów

```
SELECT pls18.stabr, ❶
    sum(pls18.visits) AS visits_2018,
    sum(pls17.visits) AS visits_2017,
    sum(pls16.visits) AS visits_2016,
    round( (sum(pls18.visits::numeric) - sum(pls17.visits)) /
        sum(pls17.visits) * 100, 1 ) AS chg_2018_17, ❷
    round( (sum(pls17.visits::numeric) - sum(pls16.visits)) /
        sum(pls16.visits) * 100, 1 ) AS chg_2017_16
FROM pls_fy2018_libraries pls18
    JOIN pls_fy2017_libraries pls17 ON pls18.fscskey = pls17.fscskey
    JOIN pls_fy2016_libraries pls16 ON pls18.fscskey = pls16.fscskey
WHERE pls18.visits >= 0
    AND pls17.visits >= 0
    AND pls16.visits >= 0
GROUP BY pls18.stabr ❸
ORDER BY chg_2018_17 DESC; ❹
```

Po słowie kluczowym *SELECT* podano nazwę kolumny *stabr* ❶ tabeli z danymi z roku 2018. Ta sama kolumna pojawia się w klauzuli *GROUP BY* ❸. Nie ma znaczenia to, z której tabeli zostanie użyta kolumna *stabr*, ponieważ zapytanie dotyczy wyłącznie agencji obecnych we wszystkich trzech tabelach. Po kolumnach *visits* dodano znajome już obliczenie zmiany procentowej, o którym była mowa w rozdziale 6. Obliczenia tego użyto dwa razy, stosując w celu zwiększenia przejrzystości aliasy *chg_2018_17* ❷ i *chg_2017_16*. Zapytanie zakończono klauzulą *ORDER BY* ❹ z sortowaniem według aliasu kolumny *chg_2018_17*.

Po uruchomieniu zapytania na początku wyników widocznych jest 10 stanów, dla których w okresie od roku 2017 do roku 2018 zwiększa się liczba wizyt. Reszta wyników prezentuje spadek wizyt. W przypadku Samoa Amerykańskiego znajdującego się na dole rankingu zanotowano spadek wynoszący 28%!

stabr	visits_2018	visits_2017	visits_2016	chg_2018_17	chg_2017_16
SD	3824804	3699212	3722376	3.4	-0.6
MT	4332900	4215484	4298268	2.8	-1.9
FL	68423689	66697122	70991029	2.6	-6.0
ND	2216377	2162189	2201730	2.5	-1.8
ID	8179077	8029503	8597955	1.9	-6.6
DC	3632539	3593201	3930763	1.1	-8.6
ME	6746380	6731768	6811441	0.2	-1.2
NH	7045010	7028800	7236567	0.2	-2.9
UT	15326963	15295494	16096911	0.2	-5.0
DE	4122181	4117904	4125899	0.1	-0.2
OK	13399265	13491194	13112511	-0.7	2.9
WY	3338772	3367413	3536788	-0.9	-4.8
MA	39926583	40453003	40427356	-1.3	0.1

WA	37338635	37916034	38634499	-1.5	-1.9
MN	22952388	23326303	24033731	-1.6	-2.9
<i>--ciąć--</i>					
GA	26835701	28816233	27987249	-6.9	3.0
AR	9551686	10358181	10596035	-7.8	-2.2
GU	75119	81572	71813	-7.9	13.6
MS	7602710	8581994	8915406	-11.4	-3.7
HI	3456131	4135229	4490320	-16.4	-7.9
AS	48828	67848	63166	-28.0	7.4

W kwestii zapewnienia kontekstu pomocne jest również sprawdzenie zmiany procentowej wartości kolumny `visits` tabel z danymi z okresu od roku 2016 do roku 2017. Dla wielu stanów, takich jak Minnesota, zanotowano ciągle spadki. W innych stanach, w tym w kilku z samej góry listy, stwierdzono wzrosty po znaczących spadkach w poprzednim roku.

Właśnie w takiej sytuacji dobrym pomysłem jest przeanalizowanie tego, co powoduje zmiany. Analiza danych może czasami skutkować pojawieniem się takiej samej liczby pytań i odpowiedzi, ale jest to częścią procesu. Zawsze warto zadzwonić do osoby zajmującej się bezpośrednio danymi, aby zaznajomiła się z Twoimi obserwacjami. Czasem taka osoba może zapewnić odpowiednie wyjaśnienie. Innym razem ekspert stwierdzi: „Nie wygląda to dobrze”. Taka odpowiedź może sprawić, że ponownie zgłosisz się do posiadacza danych lub dokumentacji w celu ustalenia, czy przeoczyłeś kod lub jakiś niuans powiązany z danymi.

Filtrowanie zapytania agregującego za pomocą klauzuli `HAVING`

Aby udoskonalić prowadzoną analizę, można sprawdzić podzbiór stanów i terytoriów mających podobne, wspólne cechy. W przypadku zmiany procentowej liczby wizyt sensowne jest oddzielenie dużych stanów od małych. W małym stanie, takim jak Rhode Island, jedna biblioteka zamknięta przez 6 miesięcy z powodu remontu mogłaby mieć znaczący wpływ na wynik. Zamknięcie jednej biblioteki w stanie Kalifornia może być ledwo zauważalne w łącznej liczbie dla całego stanu. W celu przyjrzenia się stanom z podobną liczbą wizyt wyniki można posortować przy użyciu jednej z kolumn `visits`. Dane wyjściowe byłyby jednak bardziej przejrzyste, gdyby uzyskano mniejszy zestaw wyników dzięki odfiltrowaniu danych zwracanych przez użyte zapytanie.

Aby wykonać operację filtrowania wyników funkcji agregujących, musisz skorzystać z klauzuli `HAVING` będącej częścią standardu ANSI SQL. Jesteś już zaznajomiony z użyciem klauzuli `WHERE` do filtrowania, ale funkcje agregujące, takie jak `sum()`, nie mogą być stosowane w obrębie klauzuli `WHERE`, ponieważ działają one na poziomie wierszy i między nimi. Klauzula `HAVING` określa warunki względem grup tworzonych w wyniku agregowania. W kodzie z listingu 9.14 zmodyfikowano zapytanie z listingu 9.13 przez wstawienie klauzuli `HAVING` po klauzuli `GROUP BY`.

Listing 9.14. Użycie klauzuli `HAVING` do filtrowania wyników zapytania agregującego

```
SELECT pls18.stabr,
       sum(pls18.visits) AS visits_2018,
       sum(pls17.visits) AS visits_2017,
       sum(pls16.visits) AS visits_2016,
```

```

round( (sum(pls18.visits::numeric) - sum(pls17.visits)) /
sum(pls17.visits) * 100, 1 ) AS chg_2018_17,
round( (sum(pls17.visits::numeric) - sum(pls16.visits)) /
sum(pls16.visits) * 100, 1 ) AS chg_2017_16
FROM pls_fy2018_libraries pls18
JOIN pls_fy2017_libraries pls17 ON pls18.fscskey = pls17.fscskey
JOIN pls_fy2016_libraries pls16 ON pls18.fscskey = pls16.fscskey
WHERE pls18.visits >= 0
AND pls17.visits >= 0
AND pls16.visits >= 0
GROUP BY pls18.stabr
HAVING sum(pls18.visits) > 50000000 ❶
ORDER BY chg_2018_17 DESC;

```

W tym przypadku tak określono wyniki zapytania, żeby zawierały wyłącznie wiersze z sumą wizyt w roku 2018 przekraczającą 50 milionów. Jest to wartość wybrana przeze mnie arbitralnie w celu pokazania tylko największych stanów. Dodanie klauzuli HAVING ❶ ogranicza liczbę wierszy w danych wyjściowych do zaledwie sześciu. W praktyce możesz poeksperymentować z różnymi wartościami. Oto wyniki:

stabr	visits_2018	visits_2017	visits_2016	chg_2018_17	chg_2017_16
FL	68423689	66697122	70991029	2.6	-6.0
NY	97921323	100012193	103081304	-2.1	-3.0
CA	146656984	151056672	155613529	-2.9	-2.9
IL	63466887	66166082	67336230	-4.1	-1.7
OH	68176967	71895854	74119719	-5.2	-3.0
TX	66168387	70514138	70975901	-6.2	-0.7

W pięciu z sześciu stanów zanotowano spadek liczby wizyt, ale zwróć uwagę na to, że skala zmiany procentowej nie jest tak szeroka jak w przypadku pełnego zestawu stanów i terytoriów. Zależnie od tego, czego się dowiemy od ekspertów zajmujących się bibliotekami, przyjrzenie się innym wariantom grupowania, jak i analizowanie jako grupy stanów z największą aktywnością może ułatwić opisanie trendów. Zastanów się nad następującym stwierdzeniem, jakie możesz sformułować: „Wśród stanów z największą liczbą wizyt Floryda była jedynym stanem, w którym zanotowano wzrost aktywności między rokiem 2017 i rokiem 2018. W reszcie stanów zauważono spadek liczby wizyt wynoszący od 2% do 6%”. Mógłbyś przygotować podobne wnioski odnoszące się do małych stanów i stanów średniej wielkości.

Podsumowanie

Jeżeli zainspirowałeś się już do odwiedzenia swojej lokalnej biblioteki i sprawdzenia kilku książek, zapytaj jej pracownika, czy w okresie trwającym dłużej niż kilka ostatnich lat ten oddział biblioteczny odnotował wzrost lub spadek liczby wizyt. Prawdopodobnie jesteś w stanie domyśleć się odpowiedzi. W tym rozdziale dowiedziałeś się, jak za pomocą

standardowych technik języka SQL podsumowywać dane w tabeli przez zastosowanie grupowania wartości i kilku funkcji agregujących. Dzięki złączeniu zbiorów danych byłeś w stanie zidentyfikować interesujące trendy.

Uświadomiono Ci też, że otrzymywane dane nie zawsze są idealnie przygotowane. Obecność w kolumnie wartości ujemnych, które pełnią bardziej rolę wskaźnika niż faktycznej wartości liczbowej, zmusiła nas do odfiltrowania takich wierszy. Niestety, tego rodzaju wyzwania są częścią codziennego świata analityka danych, dlatego w następnym rozdziale zostanie wyjaśnione, jak oczyścić zbiór danych z kilkoma problemami. W dalszej części książki zaznajomisz się też z kolejnymi funkcjami agregującymi, które ułatwią Ci zidentyfikowanie przydatnych informacji w posiadanych danych.

WYPRÓBUJ TO SAM

Wykorzystaj swoje umiejętności z zakresu grupowania i agregowania, aby sprawdzić się w pojedynku z następującymi wyzwaniami:

1. Stwierdzono, że w większości miejsc w ostatnim czasie zmniejszyła się liczba wizyt w bibliotekach. Jaki jest jednak schemat zatrudnienia w bibliotekach? Wszystkie trzy tabele badań dotyczących bibliotek zawierają kolumnę `totstaff`, w której przechowuje się liczbę pracowników zatrudnionych na pełnym etacie. Zmodyfikuj kod z listingów 9.13 i 9.14, aby obliczyć zmianę procentową w czasie w przypadku sumy wartości kolumny. Sprawdź wszystkie stany, a także stany z największą liczbą osób odwiedzających. Uważaj na wartości ujemne!
2. W tabelach badań dotyczących bibliotek jest kolumna o nazwie `obereg` zawierająca 2-cyfrowy kod urzędu BEA (Bureau of Economic Analysis), który klasyfikuje każdą agencję bibliotek zgodnie z regionem Stanów Zjednoczonych, takim jak Nowa Anglia, Góry Skaliste itp. Tak jak obliczono zmianę procentową dla liczby wizyt grupowanych według stanu, tak to samo działanie wykonaj w celu pogrupowania za pomocą kolumny `obereg` zmian procentowych liczby wizyt według regionów Stanów Zjednoczonych. Zajrzyj do dokumentacji badań, aby zaznajomić się ze znaczeniem kodu każdego regionu. W ramach dodatkowego wyzwania utwórz tabelę z kodem kolumny `obereg` pełniącym rolę klucza głównego oraz z nazwą regionu jako kolumną tekstową. Dołącz następnie tę tabelę do zapytania podsumowującego w celu pogrupowania wyników według nazwy regionu, a nie według kodu.
3. Wracając myślami do typów złączeń omówionych w rozdziale 7., zastanów się, jaki typ złączenia umożliwi pokazanie wszystkich wierszy każdej z trzech tabel z uwzględnieniem wierszy bez pasującej wartości? Utwórz takie zapytanie i dodaj w klauzuli `WHERE` filtr `IS NULL`, aby wyświetlić agencje, których nie ma w jednej lub większej liczbie tabel.

PROGRAM PARTNERSKI

— GRUPY HELION —

- 
- A top-down view of four hands holding four interlocking puzzle pieces. The top-left piece is red, the top-right is dark blue, the bottom-left is light blue, and the bottom-right is a slightly darker shade of light blue. The hands are positioned around the pieces, with fingers visible holding them. The background is a solid grey.
1. ZAREJESTRUJ SIĘ
 2. PREZENTUJ KSIĄŻKI
 3. ZBIERAJ PROWIZJĘ

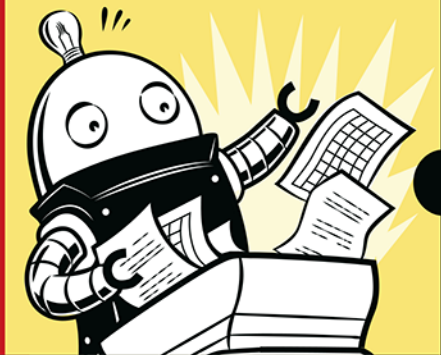
Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

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

GRUPA
Helion 

**PRZEJRZyste WPROWADZENIE
DO PROCESU EKSPLOWANIA
I ANALIZOWANIA DANYCH
ORAZ ZARZĄDZANIA NIMI**



Język SQL służy do definiowania, porządkowania i eksplorowania danych w relacyjnych bazach danych. Nieco bardziej złożone zapytania SQL pozwalają na efektywne wydobywanie wiedzy z danych. SQL jest dojrzałym językiem, używanym w wielu systemach bazodanowych. Jednym z nich jest PostgreSQL, darmowy i powszechnie znany, dostępny również dla środowisk chmurowych.

Książka jest przystępnym przewodnikiem po zastosowaniu języka SQL w procesie uzyskiwania informacji zawartych w danych. Zaczynasz od przyswojenia podstawowej wiedzy o bazach danych i SQL, a następnie przystąpisz do analizy prawdziwych zbiorów danych, takich jak demografia spisu ludności w Stanach Zjednoczonych, przejazdy taksówek w Nowym Jorku i szczegóły z krajowego katalogu targów rolniczych. Dzięki ćwiczeniom i przykładom zamieszczonym w każdym rozdziale szybko zaznajomisz się ze wszystkimi, również najnowszymi, narzędziami niezbędnymi do budowania zaawansowanych baz danych PostgreSQL. Zrozumiesz również, jak w szybki i efektywny sposób dane pozwalają zdobyć potrzebne informacje.

Dowiedz się, jak:

- tworzyć bazy PostgreSQL z użyciem własnych danych
- agregować, sortować i filtrować dane w celu zidentyfikowania wzorców
- tworzyć zapytania dla systemów GIS
- używać funkcji do wykonywania działań matematycznych i operacji statystycznych
- tworzyć złożone zapytania i automatyzować zadania

Anthony DeBarros jest redaktorem „Wall Street Journal”, gdzie zajmuje się gospodarką, demografią i finansami w polityce. Jest zwolennikiem tzw. dziennikarstwa danych i wyszukiwania w nich informacji. Wcześniej brał udział w przygotowywaniu wydań gazet „USA Today” i „Poughkeepsie Journal”, a także zajmował się opracowywaniem strategii dotyczącej treści.

Helion



helion.pl



HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶



ISBN 978-83-289-1018-8



Cena: 99,00 zł

