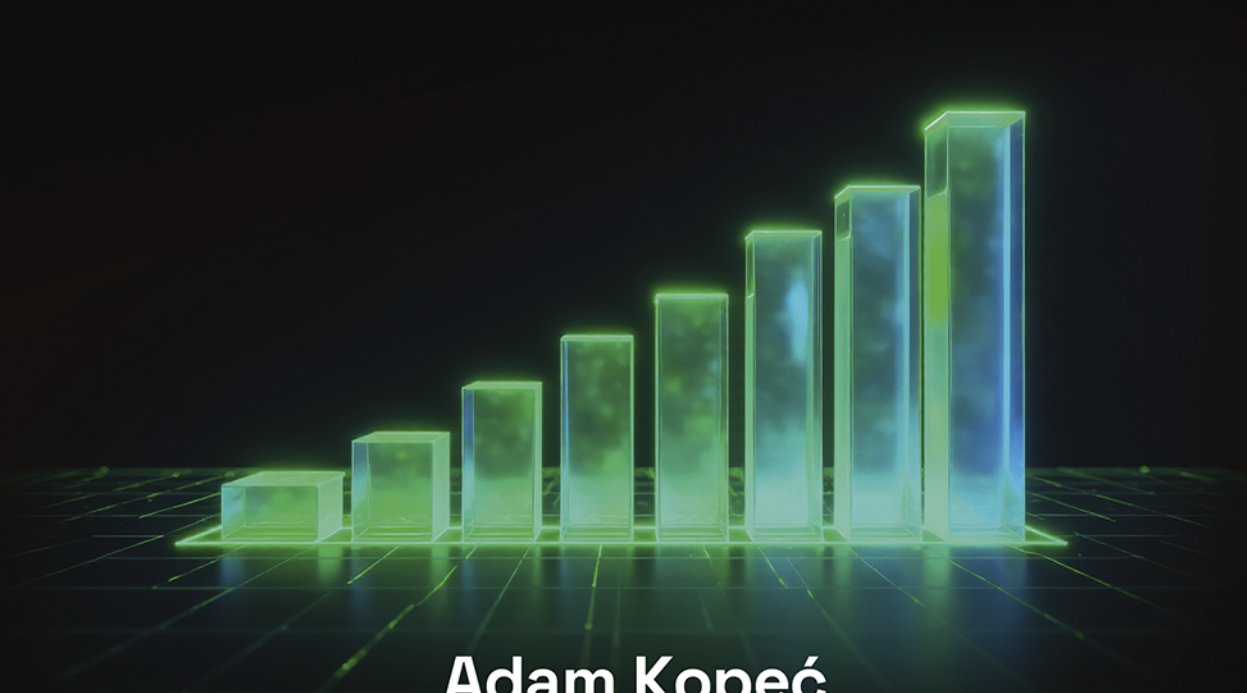


Adam Kopeć

Power Query w Excelu

Analizuj dane jak profesjonalista

Helion

Redaktor prowadzący: Natalia Hermansa

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą AdobeStock.com.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/powqex
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Kody źródłowe wybranych przykładów dostępne są pod adresami:
https://ftp.helion.pl/przyklady/powqex.zip
https://github.com/ExceliAdam/Power-Query-Helion

ISBN: 978-83-289-3618-8

Printed in Poland.

Spis treści

Od chaosu danych do precyzyjnej analizy

W dzisiejszym świecie najcenniejszym zasobem jest czas. Codziennie mierzymy się z ogromną ilością informacji płynących z różnych kierunków. Dane te często wymagają żmudnego przygotowania, zanim zostaną przekształcone w użyteczne odpowiedzi. Książka, którą trzymasz w ręku, to przewodnik po Power Query, czyli narzędziu, które przyspieszy Twoją analizę danych, a tym samym pozwoli Ci wcześniej uzyskać odpowiedzi na pytania istotne dla Ciebie lub Twojej firmy.

Power Query to silnik typu ETL (ang. *Extract, Transform, Load*), którego głównym zadaniem jest pobieranie, przekształcanie i ładowanie danych w sposób zautomatyzowany. Książka poprowadzi Cię przez proces zamiany „surowych" danych w proste do analizy tabele. Zamiast tracić godziny na ręczne korygowanie informacji, stworzysz automatyczne procesy analizujące dane.

Każdy kolejny rozdział wprowadzi Cię w inne tajniki Power Query, od interfejsu graficznego po zaawansowany kod M. Dowiesz się, jak importować dane z niemal dowolnego źródła: od plików tekstowych *.csv* i arkuszy Excela przez bazy danych i strony internetowe aż po PDF-y i obrazy. Poznasz techniki oczyszczania tekstów, pracy z datami oraz zaawansowanego scalania i grupowania informacji. Odkryjesz, jak tworzyć niestandardowe funkcje i optymalizować wydajność zapytań, aby Twoje raporty działały szybciej i sprawniej.

Każdy rozdział zawiera praktyczne przykłady, które pozwolą Ci natychmiast wdrożyć zdobytą wiedzę w swojej pracy.

Niezależnie od tego, czy dopiero zaczynasz pracę z analizą danych w Excelu, czy zdobyłeś już pierwsze doświadczenia z Power Query, ta książka dostarczy Ci narzędzi do automatyzacji powtarzalnych zadań.

Zacznij swoją podróż po świecie Power Query, który przyspieszy analizę danych i pozwoli Ci skoncentrować się na wyciąganiu wniosków i podejmowaniu trafnych decyzji biznesowych.

Dodatkowe materiały – pliki początkowe, pliki końcowe oraz rysunki w kolorze – dostępne są pod poniższymi adresami:

https://ftp.helion.pl/przyklady/powqex.zip
https://github.com/ExceliAdam/Power-Query-Helion

ROZDZIAŁ 1.

Wprowadzenie do Power Query

Czym jest Power Query?

W tej książce omówimy Power Query, popularny dodatek do Excela, a także integralną część Power BI Desktop. Narzędzie to służy przede wszystkim do zarządzania danymi. Można o nim myśleć jak o magicznym procesie, który pozwala najpierw pobrać dane z różnych miejsc, uporządkować je i wyczyścić, a następnie załadować tam, gdzie są potrzebne do dalszej analizy i/lub wizualizacji.

W terminologii IT ten proces nazywa się ETL, co po angielsku oznacza *Extract, Transform, Load*, czyli pobierz, przekształć i załaduj.

Zadania Power Query

Power Query umożliwia automatyzowanie żmudnych i powtarzalnych czynności związanych z przygotowywaniem danych do analizy. Pozwala to oszczędzić czas i eliminuje błędy, które mogły się pojawić przy ręcznym kopiowaniu i modyfikowaniu informacji. Dzięki temu, nawet jeśli pobieramy dane z różnych źródeł:

- plików Excela,
- baz danych,
- plików tekstowych, *.csv*, *.pdf*, a nawet obrazów,
- internetu,

możemy je łatwo połączyć, uporządkować i przygotować do dalszej analizy i/lub wizualizacji.

Przykładowe zastosowania

Power Query możemy wykorzystać np. do:

- łączenia wielu plików z danymi sprzedaży w jedną tabelę,
- usuwania/odfiltrowywania pustych czy błędnych wierszy i kolumn,
- tworzenia nowych kolumn z obliczeniami,
- grupowania danych,
- scalania danych,
- oczyszczania danych,
- dzielenia danych według ograniczników lub liczby znaków.

Wszystko to bez konieczności pisania skomplikowanego kodu, bo odbywa się to w większości w prostym interfejsie graficznym. Każdy wykonany przez nas krok jest zapisywany i możesz go w każdej chwili zmienić lub usunąć (jeśli już dobrze rozumiesz zależności między krokami), co daje Ci dużą elastyczność.

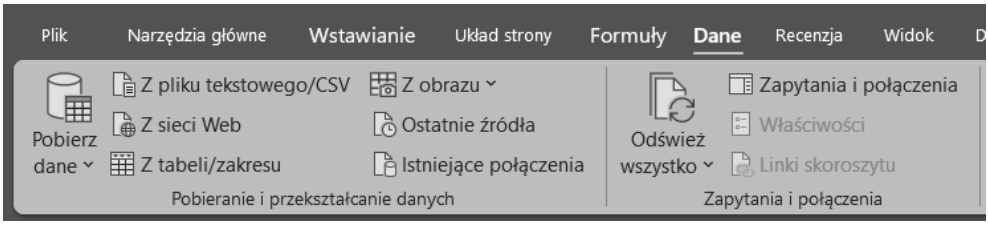
Historia

Power Query zaczynał jako osobny dodatek do Excela, który trzeba było samodzielnie zainstalować. Od Excela 2016 narzędzie to jest już integralną częścią programu. Ułatwia to korzystanie z niego szerokiemu gronu użytkowników. Microsoft stale rozwija Power Query — rozszerza jego możliwości i integruje je głębiej zarówno z Excelem, jak i Power BI, w którym stanowi fundament procesu przygotowywania i oczyszczania danych.

W skrócie: Power Query to narzędzie pomagające w prosty sposób zamienić surowe **dane** w użyteczne informacje, które można łatwo analizować i/lub wizualizować. Pozwala ograniczyć żmudne ręczne przetwarzanie danych dzięki powtarzalności zastosowanych przekształceń i łatwości ich implementacji.

My będziemy pracować na wersji Power Query z końca 2025 roku, powiązanej z Excelem 365.

Polecenia, które nas interesują, są dostępne na karcie *Dane*. To grupa poleceń *Pobieranie i przekształcanie danych*, ale także grupa poleceń *Zapytania i połączenia* (rysunek 1.1).



RYSUNEK 1.1. Polecenia Power Query na karcie Dane

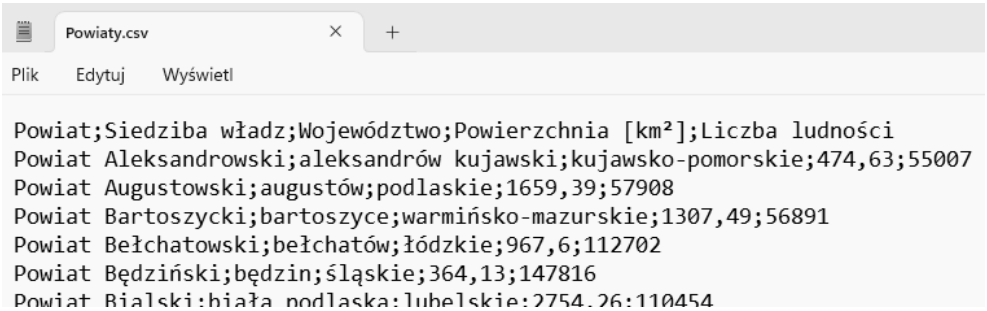
Omówimy teraz interfejs edytora Power Query, korzystając z przykładu pobierania danych z pliku *.csv*, czyli prostego pliku tekstowego.

Pierwszy przykład — pobieranie danych z pliku .csv

Rozszerzenie tego pliku oznacza *comma-separated values*, czyli 'wartości rozdzielone przecinkiem'. Jednak nie zawsze symbolem dzielącym dane na kolumny jest przecinek. Przykładowo w Polsce, Hiszpanii, Czechach czy Finlandii dane rozdziela się średnikiem. Jest to związane z zapisem liczb, w którym część całkowitą od dziesiętnej rozdziela się przecinkiem, a nie kropką, jak w Stanach Zjednoczonych, Wielkiej Brytanii czy Japonii.

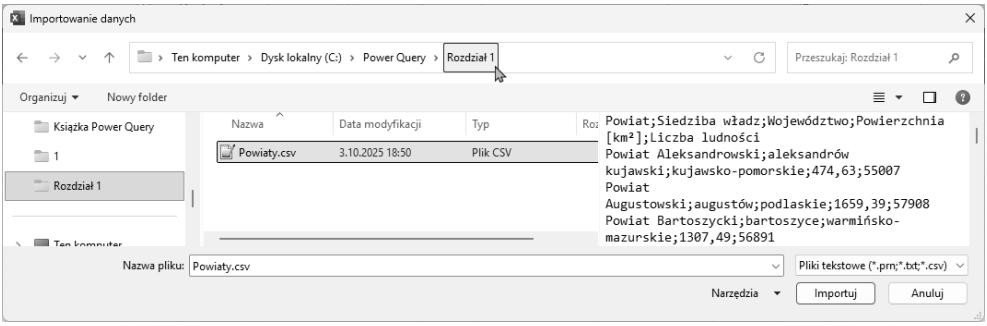
Źródło danych

Dane będziemy ładować do pustego pliku Excela z pliku *Powiaty.csv*. Plik ten ma zakodowanych pięć kolumn: `Powiat`, `Siedziba władz`, `Województwo`, `Powierzchnia [km²]` oraz `Liczba ludności` (rysunek 1.2).



RYSUNEK 1.2. Dane w pliku Powiaty.csv

Żeby pobrać dane z tego pliku, w Excelu przechodzimy na kartę *Dane* i klikamy polecenie *Z pliku tekstowego/CSV* (patrz rysunek 1.1). Spowoduje to otworzenie okna Eksploratora Windows, w którym musimy zlokalizować plik *.csv* lub inny format pliku tekstowego (rysunek 1.3).



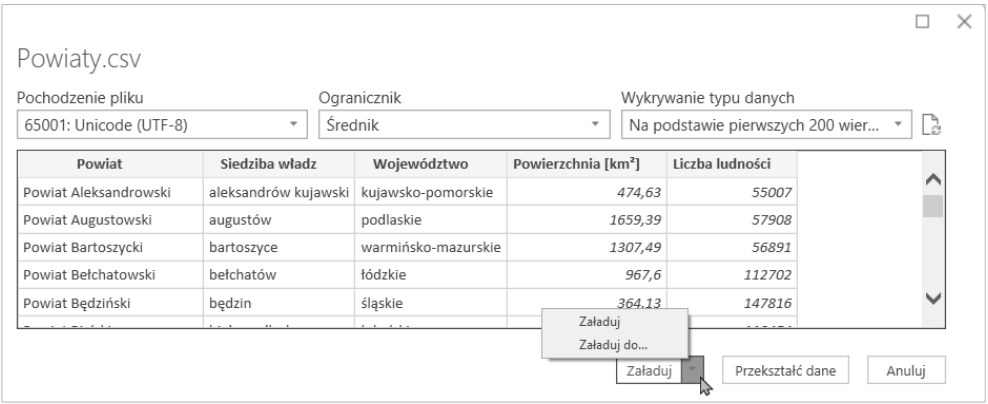
RYSUNEK 1.3. Okno Importowanie danych

Właściwości importu

Żeby zatwierdzić import konkretnego pliku, albo klikamy go dwukrotnie, albo klikamy tylko raz (zaznaczamy), a następnie klikamy przycisk *Importuj*. Spowoduje to otworzenie okna Nawigatora Power Query (rysunek 1.4) służącego do ustalenia ustawień pobierania.

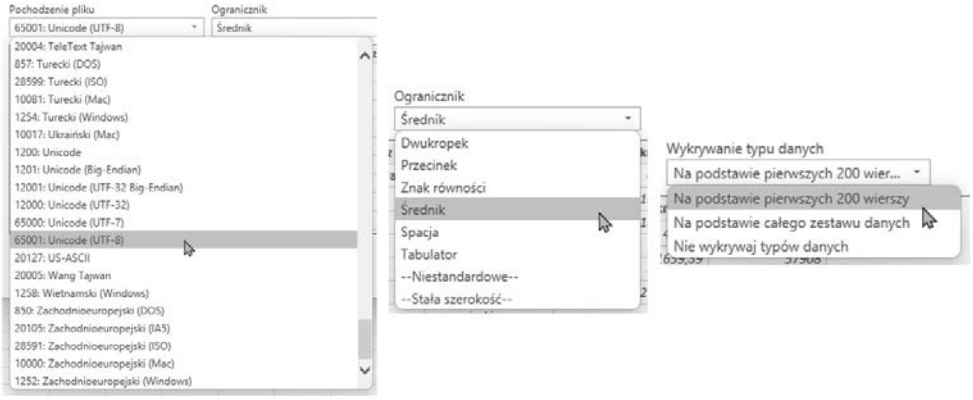
W większości okien interfejsu użytkownika Power Query można zwiększyć czcionkę za pomocą skrótu klawiszowego *Ctrl+Shift+=*.

Do zmniejszenia czcionki należy użyć skrótu klawiszowego *Ctrl+Shift+–*.



RYSUNEK 1.4. Okno Nawigatora pobierania Power Query

W tym oknie mamy trzy listy rozwijane (*Pochodzenie pliku*, *Ogranicznik*, *Wykrywanie typu danych*), z których Power Query powinien automatycznie wybrać odpowiednie opcje. Jeśli Power Query nie uda się poprawnie zidentyfikować danych, zawsze możemy zmienić wybrane wartości na listach. Wszystkie rozwinięte listy są pokazane na rysunku 1.5.



RYSUNEK 1.5. Rozwinięte listy dla opcji importowania pliku .csv

W tym przykładzie Power Query poprawnie wykrył wszystkie ustawienia, więc nie musimy nic zmieniać.

Warto zauważyć, że pierwszy wiersz danych jest specjalnie wyróżniony, co oznacza, że Power Query rozpoznał go jako nagłówek kolumn.

W przypadku Polski najbardziej popularne systemy kodowania plików (*Pochodzenie pliku*) to UTF-8, zachodnioeuropejski i środkowoeuropejski.

Gdy wszystkie ustawienia wyglądają poprawnie, możemy załadować dane do Power Query za pomocą przycisku *Przekształć dane* (patrz rysunek 1.4). Przycisk *Załaduj* i jego rozwinięta opcja *Załaduj do* (patrz rysunek 1.4) pozwalają na bezpośrednie wstawienie danych do Excela. Opcja *Załaduj* ładuje dane do nowego arkusza, a po wybraniu *Załaduj do*

możemy precyzyjniej określić sposób załadowania danych do Excela (patrz rysunek 1.17). Jednak prawie nigdy nie chcemy korzystać z opcji bezpośredniego wstawiania danych do Excela, ponieważ powinniśmy się upewnić, że Power Query poprawnie wykrył dane, i przeważnie też chcemy dokonać jakiegoś przekształcenia, zanim dane trafią do Excela. Dlatego teraz klikamy przycisk *Przekształć dane*. Spowoduje to otworzenie edytora Power Query z załadowanymi danymi (rysunek 1.6).

RYSUNEK 1.6. Dane załadowane do edytora Power Query

Podstawy edytora Power Query

Jeśli nad danymi nie widać paska formuły, musimy przejść na kartę *Widok* i zaznaczyć tam checkbox *Pasek formuły* (rysunek 1.7). Od razu możemy zaznaczyć też checkbox *Zawsze zezwalaj* w grupie poleceń *Parametry*, na potrzeby późniejszych przykładów (patrz rozdział 12.).

RYSUNEK 1.7. Karta Widok

W edytorze Power Query (patrz rysunek 1.6) można wyróżnić kilka najważniejszych części:

1. Karty menu (analogiczne do tych w Excelu), znajdujące się na górze okna programu. Zawierają przede wszystkim polecenia, które ułatwiają przekształcenia danych.
2. Zakładkę zapytań, znajdującą się po lewej stronie. Power Query powinien ją automatycznie otworzyć. Jeśli zakładka nie jest widoczna, można ją rozwinąć przez kliknięcie symbolu „większe niż" (>), wyświetlany po lewej stronie na wysokości paska formuły. Analogicznie zwija się tę zakładkę: kliknięciem symbolu „mniejsze niż" (<) (patrz rysunek 1.6).
3. Pasek formuły, umieszczony nad danymi z aktywnego kroku. W nowszych wersjach Excela/ Power Query kluczowe elementy kodu są odpowiednio kolorowane. W pasku formuły widoczna jest formuła języka M dla aktywnego/zaznaczonego kroku.

4. Centralnie umiejscowiony wynik aktywnego/zaznaczonego kroku. Najczęstszym wynikiem przekształceń jest tabela. Każda kolumna w tabeli ma swoją nazwę oraz przypisany typ danych. Typ danych można rozróżnić po ikonie znajdującej się po lewej stronie nagłówka kolumny.
5. Zakładka *Ustawienia zapytania*, znajdująca się po prawej stronie. Na górze widać nazwę zapytania, utworzoną na podstawie nazwy źródła (nazwy pliku, folderu czy tabeli, z których są pobierane dane). Poniżej nazwy znajduje się lista zastosowanych kroków, czyli lista wykonanych po kolei przekształceń danych.

Najważniejsze w zapytaniu są oczywiście dane, a w drugiej kolejności — wykonane kroki, czyli przekształcenia danych mające na celu uzyskanie danych czystych i łatwych do dalszej analizy. W tym przykładzie powinniśmy zobaczyć trzy kroki (przekształcenia). Żeby zobaczyć wygląd danych po każdym kroku/przekształceniu, wystarczy kliknąć nazwę danego kroku po prawej stronie edytora Power Query.

Zawsze w pierwszym kroku, nazywanym *Źródło*, pobierane są dane. W tym przykładzie dane pobierane są z pliku *.csv*, ale to również mogą być dane z internetu, z tabeli Excela, baz danych i wielu innych miejsc.

Drugi krok w tym zapytaniu nazwany jest *Nagłówki o podwyższonym poziomie*. Wykorzystuje on dane z pierwszego wiersza jako dane nagłówkowe, czyli jako nazwy kolumn.

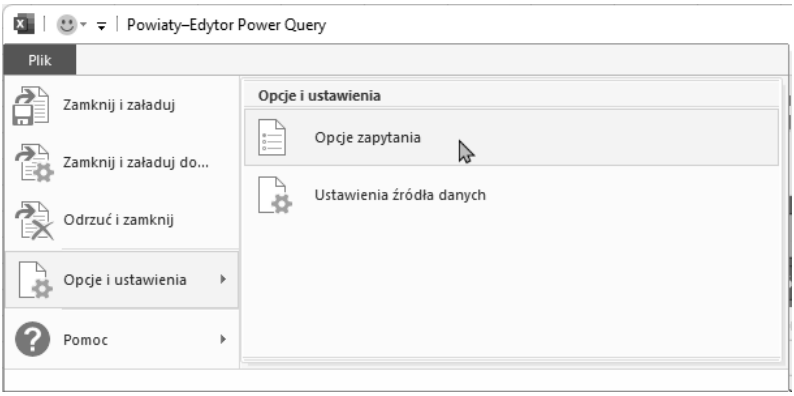
Trzeci krok (*Zmieniono typ*) automatycznie wykrywa typy danych dla poszczególnych kolumn. Ponieważ pobieramy dane z pliku tekstowego, domyślnie każda kolumna ma przypisany typ danych *Tekst*, ale nie jest to poprawny typ danych dla kolumn `Powierzchnia [km²]` oraz `Liczba ludności`. Właściwe typy danych dla tych kolumn to, odpowiednio, *Liczba dziesiętna* i *Liczba całkowita*. Zostały one poprawnie wykryte przez Power Query w trzecim kroku.

Podobnie jak w Excelu wartości tekstowe w kolumnach są wyrównane do lewej, a liczbowe do prawej.

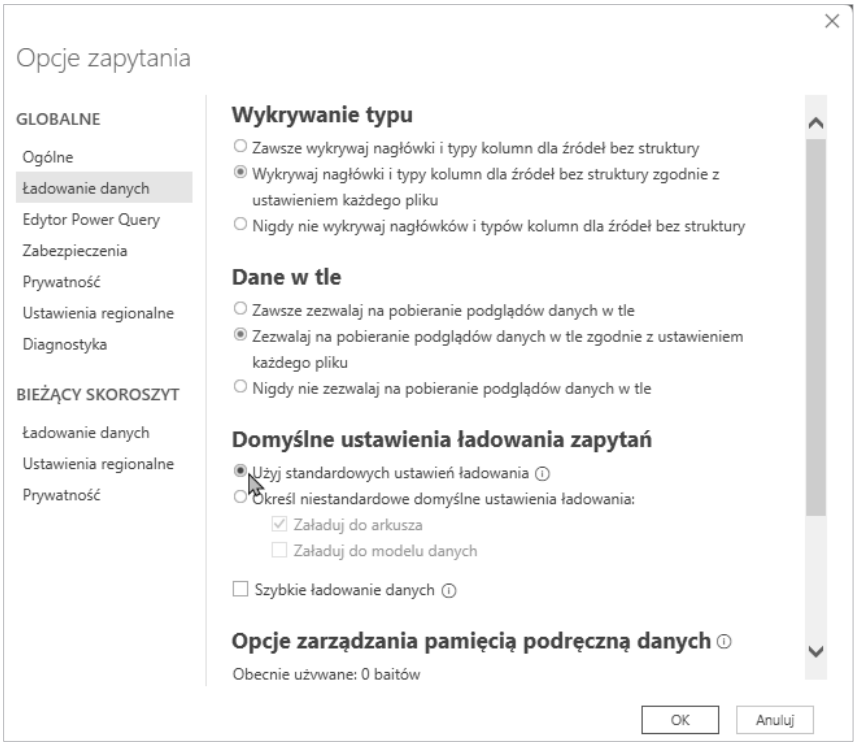
Opcje Power Query

Przy domyślnych ustawieniach Power Query ten krok powinien się dodać automatycznie. Jeśli się nie doda, będziemy musieli odpowiednio zmienić ustawienia edytora: rozwijamy menu *Plik*, a następnie przechodzimy do *Opcje i ustawienia* i klikamy *Opcje zapytania* (rysunek 1.8).

Spowoduje to otworzenie okna *Opcje zapytania*, gdzie interesują nas dwie zakładki. W pierwszej kolejności w sekcji *Globalne* chcemy przejść na zakładkę *Ładowanie danych* (rysunek 1.9), w której zaznaczamy pierwsze lub drugie ustawienie opcji *Wykrywanie typu*. Dodatkowo zakładamy, że chcemy używać standardowych ustawień ładowania, dlatego w sekcji *Domyślne ustawienia ładowania zapytań* wybieramy opcję *Użyj standardowych ustawień ładowania*. Będzie to dla nas miało znaczenie przy ładowaniu danych do Excela (patrz rysunek 1.16).

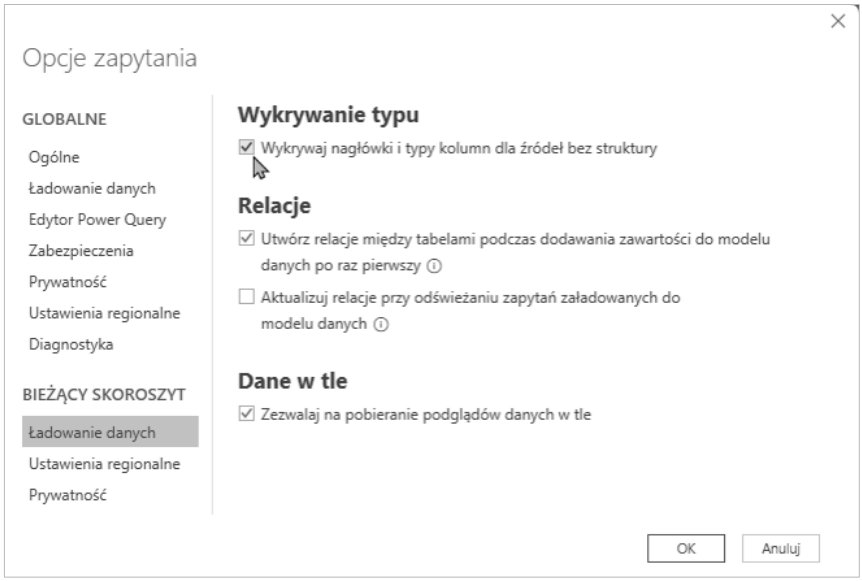


RYSUNEK 1.8. Opcje zapytania



RYSUNEK 1.9. Globalne opcje ładowania danych

Teraz przechodzimy do sekcji *Bieżący skoroszyt* i ponownie przechodzimy do zakładki o nazwie *Ładowanie danych* (rysunek 1.10), gdzie upewniamy się, że zaznaczony jest checkbox *Wykrywaj nagłówki i typy kolumn dla źródeł bez struktury zgodnie z ustawieniami każdego pliku* (domyślnie powinien być zaznaczony dla każdego nowego pliku Excela).

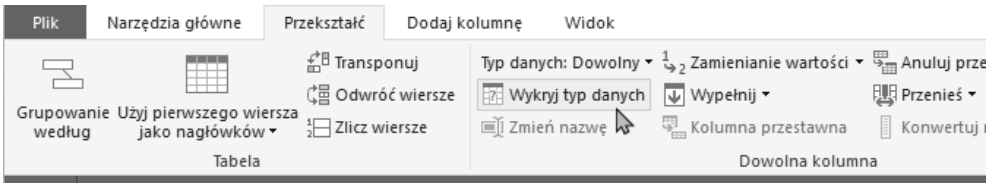


RYSUNEK 1.10. Opcje ładowania danych dla bieżącego skoroszytu

Zmienienie opcji zapytań nie wpłynie na bieżące kroki. Dopiero przy kolejnych przekształceniach bądź zapytaniach będziemy mogli zaobserwować ewentualne różnice w zachowaniu edytora.

Szybkie przypisanie typów danych

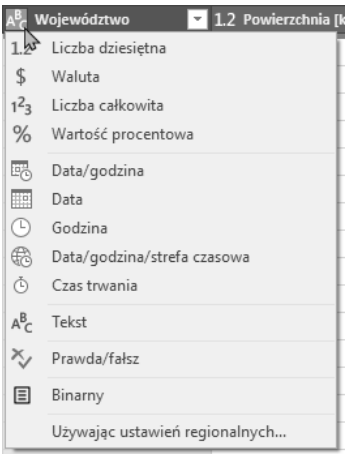
Jak w takim razie możemy szybko wykryć typy danych dla wszystkich kolumn, jeśli odpowiedni krok nie dodał się automatycznie? Wystarczy, że zaznaczymy wszystkie kolumny (np. za pomocą skrótu *Ctrl+A*, gdy zaznaczone są jakieś dane), a następnie na karcie *Przekształć* klikniemy polecenie *Wykryj typ danych* (rysunek 1.11).



RYSUNEK 1.11. Karta Przekształć

Czasami Power Query niepoprawnie wykrywa typ danych. W takiej sytuacji zawsze możemy zmienić typ danych samodzielnie. Jeśli chcemy zmienić typ danych tylko dla pojedynczej kolumny, możemy kliknąć ikonę bieżącego typu danych w nagłówku kolumny. Spowoduje to utworzenie listy rozwijanej, na której widać typy danych wraz z przypisanymi im ikonami (rysunek 1.12).

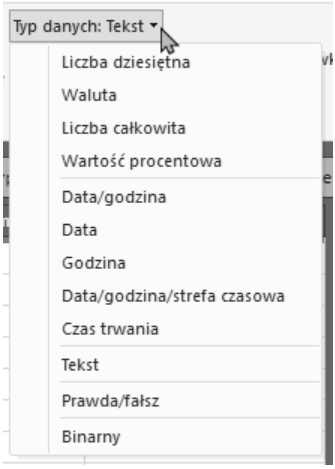
RYSUNEK 1.12.
Podręczne menu typu danych



Jeśli musimy kilku kolumnom nadać ten sam typ danych, możemy je zaznaczyć, a następnie na karcie *Narzędzia główne* rozwinąć polecenie *Typ danych* (rysunek 1.13). To polecenie przy okazji pokazuje typ danych dla aktywnej kolumny, ale nie pokazuje ikon powiązanych z poszczególnymi typami danych.

W edytorze możemy zaznaczać wiele kolumn w ten sam sposób co w Excelu, czyli przytrzymując klawisz *Ctrl*, możemy zaznaczać dowolne kolumny i usuwać ich zaznaczenie, a przytrzymując klawisz *Shift*, zaznaczamy wszystkie kolumny od aktywnej do tej, którą kliknęliśmy.

RYSUNEK 1.13.
Polecenie Typ danych na karcie Narzędzia główne

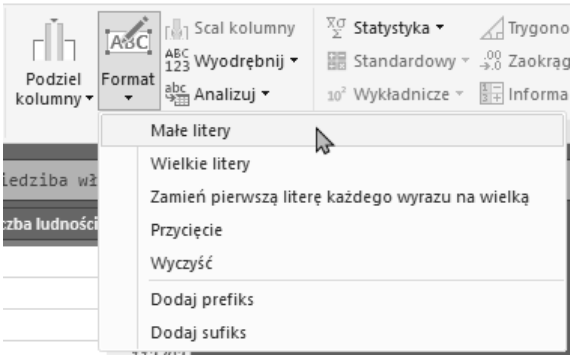


W tym przykładzie Power Query poprawnie wykrył wszystkie typy danych (patrz rysunek 1.6), ale zanim załadujemy dane do Excela, chcemy jeszcze poprawić pisownię w kolumnach tekstowych.

Podstawowe przekształcenia

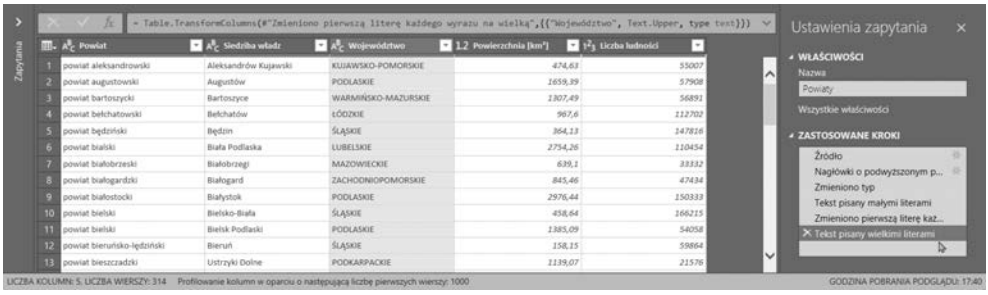
Zgodnie z polską pisownią wszystkie wyrazy w kolumnie `Powiat` powinny być pisane małymi literami. Nazwy miast z kolumny `Siedziba` władz powinny rozpoczynać się wielką literą. Natomiast w nazwach województw wszystkie litery chcemy zamienić na wielkie. Odpowiednie przekształcenia znajdziemy po rozwinięciu polecenia *Format* na karcie *Przekształć* (rysunek 1.14).

RYSUNEK 1.14. Rozwinięte polecenie Format na karcie Przekształć



Polecenie *Format*, jak i wiele innych poleceń z karty *Przekształć*, można też znaleźć na karcie *Dodaj kolumnę*, jednak jest pomiędzy nimi kluczowa różnica. Polecenia z karty *Przekształć* zmieniają istniejące kolumny, natomiast polecenia z karty *Dodaj kolumnę* pozostawiają oryginalną kolumnę oraz dodają kolumnę po zastosowaniu transformacji.

Chcemy teraz po kolei dodać kroki przekształcające wspomniane kolumny. Oznacza to, że musimy się upewnić, że aktywny/zaznaczony jest ostatni krok zapytania. Następnie zaznaczamy pojedynczą kolumnę i dokonujemy na niej wybranego przekształcenia. Przy okazji obserwujemy kroki dodane do zapytania oraz pasek formuły, żeby przyzwyczaić się do struktury kodu M. Po wykonaniu wszystkich zmian na liście zastosowanych kroków powinniśmy zobaczyć sześć kroków, a wynikowa tabela powinna wyglądać tak jak na rysunku 1.15.



RYSUNEK 1.15. Dane po przekształceniach

To wszystkie przekształcenia, jakie chcemy wykonać dla tego zapytania, więc możemy już załadować dane do Excela, ale jest jedna operacja, o której powinniśmy pamiętać w przypadku każdego zapytania. Jest to sprawdzenie nazwy zapytania: czy faktycznie odpowiada

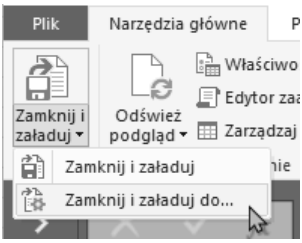
danym wynikowym oraz czy jest wystarczająco jasna i prosta, ponieważ służy ona również do tworzenia nazw wynikowych tabel Excela. W tym przykładzie domyślna nazwa pasuje do uzyskanego wyniku, więc nie musimy jej zmieniać.

Nazwę zapytania zawsze powinno się zmieniać na początku przekształceń, żeby o tym nie zapomnieć przed załadowaniem wyniku zapytania do Excela. Nazwę zapytania można zmienić w sekcji *Ustawienia zapytania* po prawej stronie edytora albo po dwukrotnym kliknięciu nazwy zapytania na liście zapytań (lewa strona edytora).

Ładowanie zapytania do Excela

Teraz możemy załadować wynik zapytania do Excela. Wystarczy, że na karcie *Narzędzia główne* rozwiniemy polecenie *Zamknij i załaduj* i wybierzemy opcję *Zamknij i załaduj do...* (rysunek 1.16).

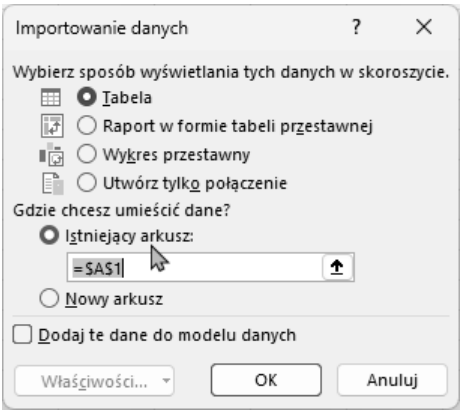
RYSUNEK 1.16.
Polecenie Zamknij i załaduj do



Wybieramy tę opcję, zamiast po prostu kliknąć polecenie *Zamknij i załaduj*, ponieważ często dane musimy załadować w konkretny sposób, więc nie zawsze chcemy polegać na domyślnych ustawieniach ładowania (patrz rysunek 1.9), zgodnie z którymi program tworzy nowy arkusz (o nazwie takiej samej jak nazwa zapytania) i do niego wgrywa dane jako tabelę Excela.

Zastosowanie wybranej przez nas opcji spowoduje zamknięcie edytora Power Query i otworzenie okna *Importowanie danych* (rysunek 1.17).

RYSUNEK 1.17.
Okno Importowanie danych



W tym oknie pozostawiamy zaznaczoną tabelę jako miejsce docelowe załadowania danych. Zmieniamy jednak miejsce umieszczenia danych z nowego arkusza na istniejący arkusz. Domyślnie powinna być zaznaczona na nim komórka `A1`, ale możemy to zmienić. Ponieważ pracujemy z pustym plikiem Excela, nie musimy zmieniać komórki docelowej. Ponowne kliknięcie komórki `A1` bądź innej dołoży do odwołania nazwę arkusza, np. `=Arkusz1!$A$1`.

Ponieważ jest to nasze pierwsze ładowanie zapytania do Excela, omówimy dokładnie możliwości, jakie mamy przy importowaniu danych. Zaczniemy od sposobu wyświetlania danych:

1. **Tabela**. Standardowa tabela Excela, do której zostanie załadowany wynik z ostatniego kroku zapytania.
2. **Tabela przestawna**. Wynik zapytania zostaje w pamięci Power Query (nie jest ładowany do komórek Excela), ale tworzona jest pusta tabela przestawna, która powiązana jest z tym wynikiem. Kolumny z zapytania są ładowane jako pola tabeli przestawnych, które można wykorzystać do tworzenia raportów.
3. **Wykres przestawny**. Podobnie jak w poprzednim punkcie dane są ładowane do tabeli przestawnej, ale dodatkowo jest też tworzony wykres przestawny.
4. **Utwórz tylko połączenie**. Dane pozostają jedynie w pamięci Power Query (za kulisami Excela) i mogą być wykorzystywane w innych zapytaniach, ale nie bezpośrednio w Excelu.

Pierwsze trzy warianty pozwalają na załadowanie wyniku do istniejącego arkusza Excela albo do nowego arkusza. Nowy arkusz dostanie nazwę zgodną z nazwą zapytania. W ostatnim wariancie opcje umieszczania danych przestają być aktywne (stają się wyszarzone), ponieważ dane pozostają tylko w pamięci Power Query.

Checkbox *Dodaj te dane do modelu danych* jest powiązany z dodatkiem Power Pivot. Ten temat nie jest omawiany w książce.

Przy pierwszym importowaniu danych do Excela przycisk *Właściwości* jest nieaktywny (wyszarzony). Właściwościami zapytań zajmiemy się w późniejszym przykładzie (patrz rysunek 17.1).

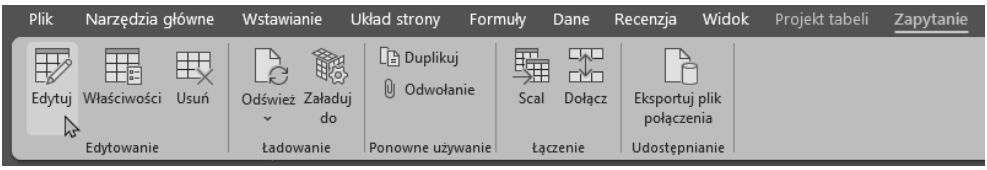
Możemy teraz zaakceptować sposób ładowania wyniku zapytania do Excela przyciskiem *OK*. Jeśli naciśniemy przycisk *Anuluj*, dane zostaną załadowane tylko jako połączenie.

Karta Zapytanie

Gdy zaznaczymy komórkę w tabeli, którą przed chwilą załadowaliśmy do Excela, pokażą się dwie dodatkowe karty. Pierwsza to standardowa karta związana z tabelą (*Projekt tabeli*), a druga to *Zapytanie*, która zawiera polecenia związane z obsługą zapytania od strony Excela (rysunek 1.18).

Poszczególne polecenia pozwalają na:

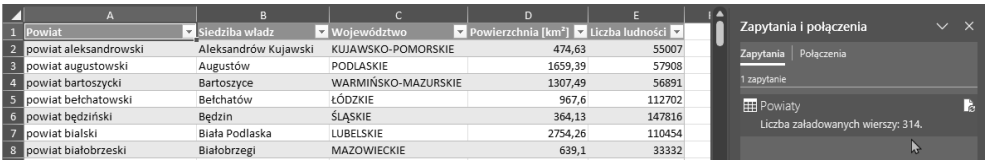
- *Edytuj* — przejście do edycji zapytania w Power Query.
- *Właściwości* — otworzenie okna właściwości zapytania.



RYSUNEK 1.18. Karta Zapytanie

- *Usuń* — usunięcie zapytania z pamięci Power Query. Jeśli zapytanie zostanie załadowane do Excela jako tabela, to ta tabela pozostanie w Excelu.
- *Odśwież* — pozwala odświeżyć zapytanie, czyli ponownie odwołać się do źródła i wykonać wszystkie wcześniej zapisane kroki.
- *Załaduj do* — otwiera ponownie okno *Importowanie danych* (patrz rysunek 1.17) z opcją zmiany sposobu ładowania zapytania do Excela.
- *Duplikuj* i *Odwołanie* — pierwsze tworzy nowe zapytanie, które jest duplikatem wcześniejszego, a drugie odwołuje się do ostatniego kroku tego zapytania.
- *Scal* i *Dołącz* — opcje omawiane w rozdziale 6.
- *Eksportuj plik połączenia* — eksportuje zapytanie jako plik *.odc*.

Po załadowaniu pierwszego zapytania do Excela po prawej stronie powinno się automatycznie otworzyć okno *Zapytania i połączenia* (rysunek 1.19).



	A	B	C	D	E
1	Powiat	Siedziba władz	Województwo	Powierzchnia [km²]	Liczba ludności
2	powiat aleksandrowski	Aleksandrów Kujawski	KUJAWSKO-POMORSKIE	474,63	55007
3	powiat augustowski	Augustów	PODLASKIE	1659,39	57908
4	powiat bartoszycki	Bartoszyce	WARMIŃSKO-MAZURSKIE	1307,49	56891
5	powiat bełchatowski	Bełchatów	ŁÓDZKIE	967,6	112702
6	powiat będziński	Będzin	ŚLĄSKIE	364,13	147816
7	powiat bialski	Biała Podlaska	LUBELSKIE	2754,26	110454
8	powiat białobrzeski	Białobrzegi	MAZOWIECKIE	639,1	33332

RYSUNEK 1.19. Załadowany wynik zapytania i okno Zapytania i połączenia

Jeśli to okno nie otworzy się automatycznie, możemy je otworzyć ręcznie przez kliknięcie polecenia *Zapytania i połączenia* na karcie *Dane* (patrz rysunek 1.1). Możemy zobaczyć w nim podsumowanie zapytania: zostało załadowanych 314 wierszy (nie wliczając w to nagłówka tabeli). W tym oknie, pod nazwą zapytania, mogą się też pojawić informacje na temat błędów w zapytaniu, ale żeby były widoczne, szerokość okna musi być większa niż szerokość komunikatu o liczbie załadowanych wierszy.

Zaimportowaliśmy dane do Excela z pliku *.csv* za pomocą naszego pierwszego zapytania. W następnym rozdziale poznasz więcej metod importowania danych z prostych źródeł, jak również więcej sposobów na przekształcanie danych.

ROZDZIAŁ 16.

Przykłady użycia Power Query

W tym rozdziale omówimy kilka przykładów użycia Power Query, które wymagają wiedzy na temat różnych przekształceń.

Czy klient powracający

W pierwszym przykładzie chcemy sprawdzić, czy zakup klienta traktujemy jako zakup powracający. Żeby można było zaliczyć zakup klienta jako zakup powracający, musi to być oczywiście zakup dokonany przez tego samego klienta, a dodatkowo nie mógł wystąpić później niż w ciągu 14 dni po wcześniejszym zakupie.

Pracujemy z danymi w tabeli `Transakcje` znajdującej się na arkuszu *Powracający*. Dane w tabeli specjalnie posortowano według kwoty zakupu (rysunek 16.1), by podkreślić, że dane źródłowe nie muszą być posortowane po potrzebnej nam kolumnie.

RYSUNEK 16.1.
Dane źródłowe
w tabeli Transakcje

	A	B	C
1	Id Klienta	Data Zakupu	Kwota Zakupu
2	5	2026-01-06	249,00 zł
3	23	2026-03-08	220,00 zł
4	13	2026-02-10	199,99 zł
5	7	2026-02-25	149,00 zł
6	2	2026-01-03	129,99 zł
7	3	2026-02-02	120,00 zł
8	5	2026-03-18	110,00 zł
9	19	2026-03-01	99,99 zł

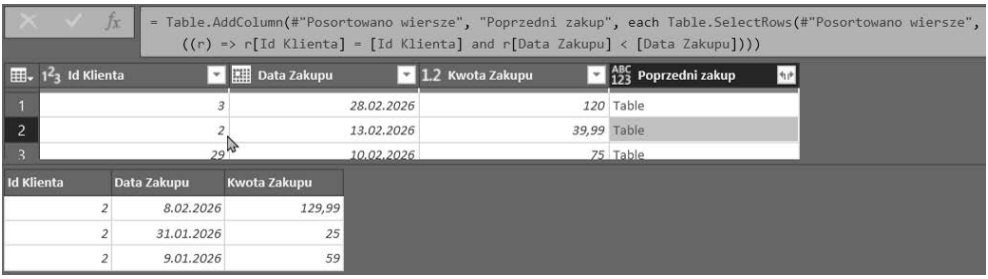
Standardowo ładujemy tabelę `Transakcje` do Power Query, pamiętając o nadaniu właściwych typów danych. Szczególnie chodzi tu o kolumnę `Data Zakupu`, w której dane powinny być zapisane tylko jako data, a nie data z czasem. Następnie sortujemy dane po kolumnie `Data Zakupu` *od Z do A*.

Własna funkcja filtrująca

Jako kolejne przekształcenie dodamy kolumnę, w której umieścimy wszystkie wiersze tabeli z tym samym ID klienta oraz datą wcześniejszą niż `Data Zakupu` z danego wiersza. Rozpoczniemy od kliknięcia na karcie *Dodaj kolumnę* polecenia *Kolumna niestandardowa* (rysunek 4.19). Następnie w oknie *Kolumna niestandardowa* zmieniamy nazwę kolumny na `Poprzedni zakup`, a jako formułę wpisujemy następujący kod:

```
Table.SelectRows(#"Posortowano wiersze", ((r) => r[Id Klienta] = [Id Klienta]
↪and r[Data Zakupu] < [Data Zakupu]))
```

Kod ten trochę przypomina kod z poprzedniego rozdziału, w którym korzystaliśmy z funkcji `List.Accumulate`, ponieważ tu również wywołujemy stworzoną przez siebie funkcję. Różnica polega na tym, że argument, który nazwaliśmy `r` (ang. *row*, wiersz), służy nam do przekazania wiersza z kroku/tabeli *#"Posortowano wiersze"* (powinna to być nazwa poprzedniego kroku) do bieżącego wiersza z bieżącego kroku. Dlatego przed pierwszymi odwołaniami do kolumn znajduje się nazwa argumentu, a przed drugimi — nie. Słowo kluczowe `each` przekazuje bowiem odwołanie do bieżącego wiersza. Pełny wynik tej formuły jest pokazany na rysunku 16.2.



RYSUNEK 16.2. Przefiltrowane tabele zagnieżdżone w komórkach

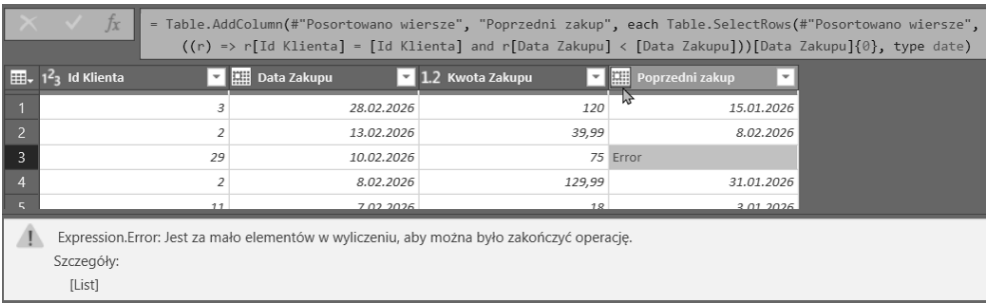
Nazwa argumentu przekazującego wiersz/rekord (rysunek 16.2) może być dowolnym wyrażeniem bez spacji i znaków specjalnych.

Obsługa błędów

Nam nie jest potrzebna cała tabela, lecz jedynie pierwsza data zakupu. Dlatego do formuły w kolumnie niestandardowej dodajemy jeszcze wyszczególnienie pierwszej komórki z kolumny `Data Zakupu`. Formuła po modyfikacji powinna wyglądać następująco:

```
Table.SelectRows(#"Posortowano wiersze", ((r) => r[Id Klienta] = [Id Klienta]
↪and r[Data Zakupu] < [Data Zakupu]))[Data Zakupu]{0}
```

Taka modyfikacja spowoduje, że w niektórych wierszach zobaczymy błąd (rysunek 16.3), ponieważ próbujemy wyodrębnić pierwszy element z tabel/list, które nie miały żadnych wierszy/elementów.

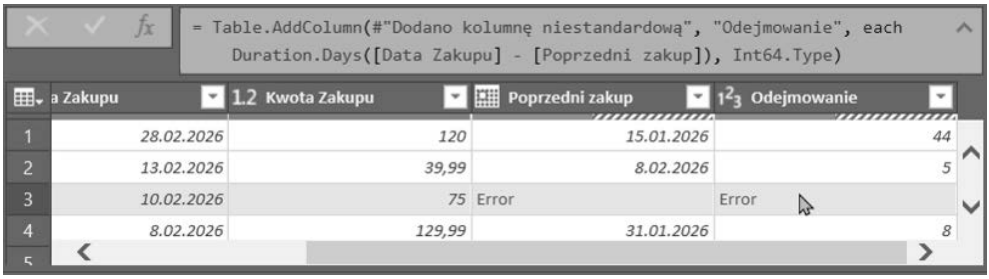


RYSUNEK 16.3. Formuła zwracająca datę poprzedniego zakupu

W Power Query mogą istnieć puste tabele i listy.

Błąd pojawia się w wierszach, w przypadku których był to pierwszy zakup danego klienta.

Obsługą błędów zajmiemy się w późniejszym kroku. Teraz zaznaczamy najpierw kolumnę `Data Zakupu`, potem `Poprzedni zakup`, a następnie na karcie *Dodaj kolumnę* rozwijamy polecenie *Data*, by wybrać opcję *Odejmij dni* (rysunek 5.18). Możemy nie zmieniać dla niej domyślnej nazwy (rysunek 16.4).

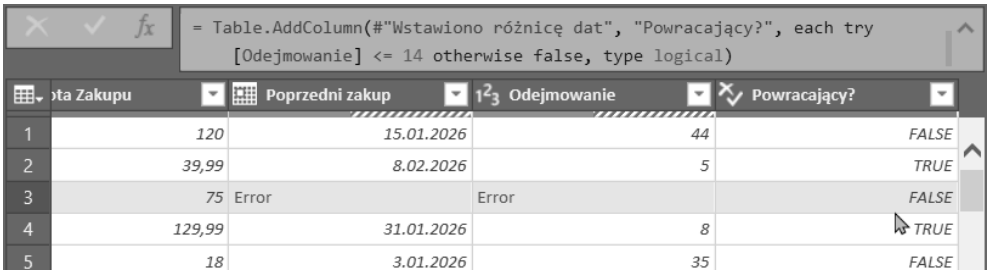


RYSUNEK 16.4. Liczba dni od poprzedniego zakupu

Wystarczy nam informacja, czy klient jest powracający, w postaci wartości logicznej prawda/fałsz (`true/false`). Dlatego możemy dodać kolumnę niestandardową nazwaną `Powracający?` z krótkim kodem sprawdzającym:

```
try [Odejmowanie] <= 14 otherwise false
```

W tym właśnie kroku w miejsce błędów wstawimy wartość `false`, czyli informację na temat tego, że nie był to zakup klienta powracającego. Możemy też określić w pasku formuły, że kolumna powinna przyjmować logiczny typ danych (rysunek 16.5).



RYSUNEK 16.5. Informacja logiczna o tym, czy klient jest powracający

Pozostaje nam jedynie usunąć kolumny `Poprzedni zakup` i `Odejmowanie` oraz zmienić nazwę zapytania na *KlienciPowracający*. Następnie możemy załadować wynik do Excela na arkusz *Powracający*.

Usuwanie pustych kolumn

W tym przykładzie chcemy usunąć wszystkie puste kolumny danych. Będziemy pobierać informacje z arkusza *PusteKolumny*, gdzie specjalnie w niektórych komórkach wstawiono formuły zwracające puste ciągi tekstowe, czyli dwa podwójne cudzysłowy (rysunek 16.6).

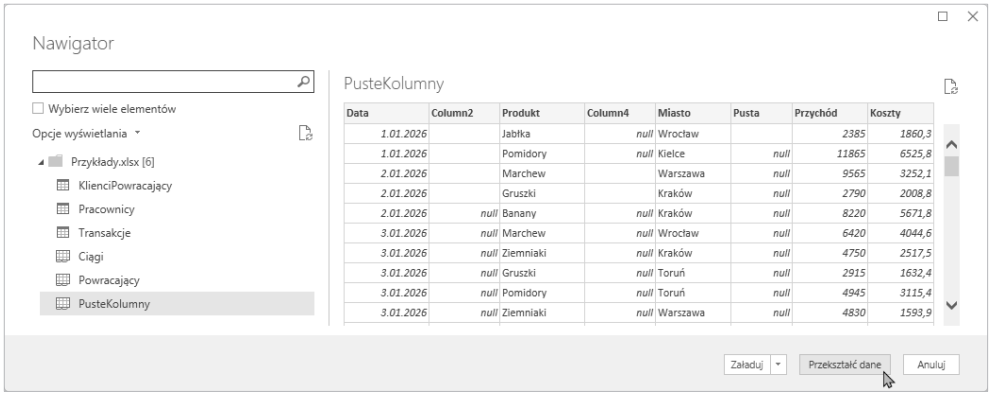
	A	B	C	D	E	F	G	H
1	Data		Produkt		Miasto	Pusta	Przychód	Koszty
2	2026-01-01	=""	Jabłka		Wrocław		2 385,00 zł	1 860,30 zł
3	2026-01-01		Pomidory		Kielce		11 865,00 zł	6 525,80 zł
4	2026-01-02		Marchew		Warszawa		9 565,00 zł	3 252,10 zł
5	2026-01-02		Gruszki		Kraków		2 790,00 zł	2 008,80 zł
6	2026-01-02		Banany		Kraków		8 220,00 zł	5 671,80 zł
7	2026-01-03		Marchew		Wrocław		6 420,00 zł	4 044,60 zł
8	2026-01-03		Ziemniaki		Kraków		4 750,00 zł	2 517,50 zł

RYSUNEK 16.6. Dane źródłowe na arkuszu PusteKolumny

Jednej z pustych kolumn przypisaliśmy nazwę (tekst w pierwszym wierszu), ale zakładamy, że chcemy usunąć kolumny, które nie mają treści, a nie kolumny, które nie są nazwane.

Zaczynamy więc od pobrania danych z arkusza, czyli najpierw na karcie *Dane* Excela rozwijamy polecenie *Pobierz dane*, przechodzimy do elementu *Z pliku* i wybieramy opcję *Ze Skoroszytu* (rysunek 3.2). Następnie odnajdujemy nasz plik Excela (*Przykłady.xlsx)* i po jego zaznaczeniu klikamy przycisk *Importuj* (podobnie jak na rysunku 1.3).

W nowo otwartym oknie *Nawigator* zaznaczamy arkusz *PusteKolumny* i klikamy przycisk *Przekształć dane* (rysunek 16.7).



RYSUNEK 16.7. Pobieranie danych z arkusza PusteKolumny

W tym kroku (rysunek 16.7) widać, że kolumnom bez nazw nagłówków zostały przypisane domyślne nazwy kolumn (`Column2` itd.) i że kilka komórek w tych kolumnach jest pustych, pozostałe zaś zawierają wartość `null` (rozróżnienie pomiędzy komórkami z formułami zwracającymi pusty ciąg tekstowy a faktycznie pustymi komórkami Excela).

Po załadowaniu danych z arkusza do Power Query domyślnie powinny pojawić się cztery kroki:

1. *Źródło* — ten krok zawiera lokalizację (ścieżkę dostępu) pliku.
2. *Nawigacja* — ten krok wybiera arkusz, z którego pobieramy dane.
3. *Nagłówki o podwyższonym poziomie* — ten krok używa wartości z pierwszego wiersza jako nazw kolumn. Jeśli komórka w pierwszym wierszu zawiera wartość `null`, to domyślna nazwa kolumny się nie zmieni, ale jeśli zawiera pusty ciąg tekstowy, to nazwa zostanie podmieniona. Takie sytuacje należy skorygować w źródle danych.
4. *Zmieniono typ* — ten krok przypisuje typy danych poszczególnym kolumnom. Nie jest on niezbędny dla tego przykładu, więc jeśli nie dodał się automatycznie, wciąż możemy wykonywać dalsze przekształcenia, trzeba tylko pamiętać odwoływać się do odpowiedniego kroku (nazwy kroku).

Nazwa kroku *Nawigacja*, wyodrębniającego informacje z arkusza wewnątrz edytora zaawansowanego, jest zgodna z wyodrębnianym elementem. W tym przykładzie jest to *PusteKolumny_Sheet*. Następny krok zapytania korzysta z nazwy zapisanej w edytorze zaawansowanym, a nie na liście kroków.

Budowanie formuły podzielone na etapy

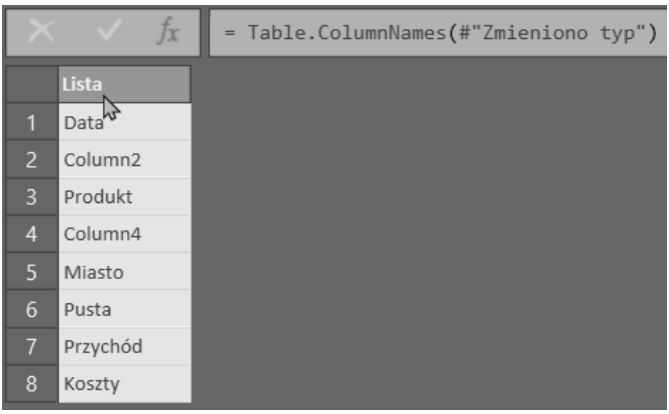
Przekształcenie usuwające puste kolumny docelowo zapiszemy w pojedynczym kroku, ale żeby go lepiej zrozumieć, będziemy dobudowywać kolejne etapy przekształcenia i pokazywać ich wyniki.

Kod M możemy pisać bezpośrednio w pasku formuły albo w edytorze zaawansowanym. Dla edytora Power Query nie ma to znaczenia. Wybór zależy od naszej wygody.

Zakładając, że wykorzystamy pasek formuły, musimy najpierw kliknąć ikonę *fx* obok niego. Powinno wstawić się odwołanie do wcześniejszego kroku, do którego musimy dołożyć pierwszą funkcję. Cała formuła w pierwszym etapie powinna wyglądać następująco (rysunek 16.8):

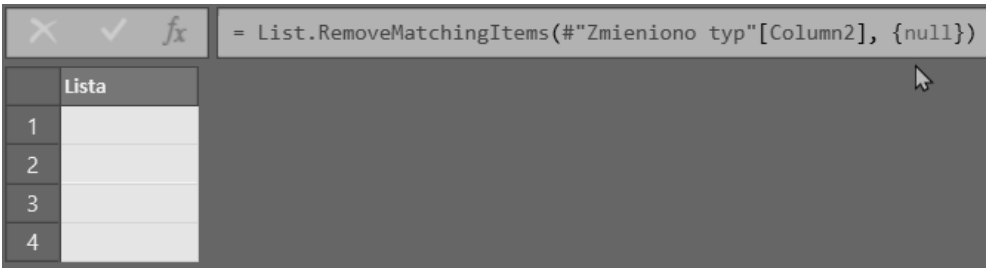
```
= Table.ColumnNames(#"Zmieniono typ")
```

RYSUNEK 16.8.
Lista nazw kolumn



W tym etapie wyodrębniamy listę nazw wszystkich kolumn z poprzedniego kroku *(#"Zmieniono typ")*. Ta funkcja będzie nam potrzebna później. Teraz zamieniamy formułę na (rysunek 16.9):

```
= List.RemoveMatchingItems(#"Zmieniono typ"[Column2], {null})
```



RYSUNEK 16.9. Kolumna Column2 po usunięciu z niej wartości null

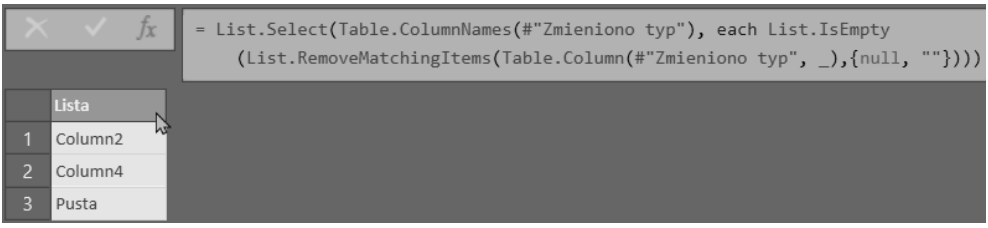
Dzięki funkcji `List.RemoveMatchingItems` usuwamy z kolumny wszystkie komórki zawierające wartości z podanej listy. Na razie usunęliśmy wartości `null`, ale chcemy też usunąć puste komórki (puste ciągi tekstowe). Jeśli kolumna po usunięciu wskazanych przez nas wartości jest pusta, to chcemy ją usunąć. Czyli dodajemy do formuły wszystkie wartości, jakie chcemy usunąć z kolumny, oraz dodajemy funkcję `List.IsEmpty`:

```
= List.IsEmpty(List.RemoveMatchingItems(#"Zmieniono typ"[Column2],{null, ""}))
```

Ta formuła zwróci wartość `TRUE` dla kolumn, które chcemy usunąć.

W następnym etapie dodamy wcześniej użytą funkcję wyodrębniającą wszystkie nazwy kolumn (rysunek 16.8), żeby sprawdzić, które kolumny chcemy usunąć. Żeby odwoływać się poprawnie do kolumn w Power Query, język M musi wiedzieć, z jakiej tabeli wyodrębnić tę kolumnę. Przy usuwaniu wartości z kolumny (rysunek 16.9) zapisaliśmy na stałe nazwę tabeli/kroku (*#"Zmieniono typ"*) oraz nazwę kolumny (`Column2`). W kolejnym etapie musimy zmieniać nazwę kolumny dynamicznie na podstawie wartości z wyodrębnionej wcześniej listy (rysunek 16.8). Niestety nie wystarczy za słowem kluczowym `each` wstawić w nawiasy kwadratowe znaku podkreślenia `[_]`, ponieważ wtedy Power Query będzie szukał kolumny nazywającej się `znak podkreślenia`. Dlatego musimy dołożyć funkcję `Table.Column`, żeby poprawnie odwoływać się dynamicznie do kolejnych kolumn. Oznacza to, że formuła w tym etapie powinna wyglądać następująco (rysunek 16.10):

```
= List.Select(Table.ColumnNames(#"Zmieniono typ"), each List.IsEmpty(List.RemoveMatchingItems
↪(Table.Column(#"Zmieniono typ", _),{null, ""})))
```

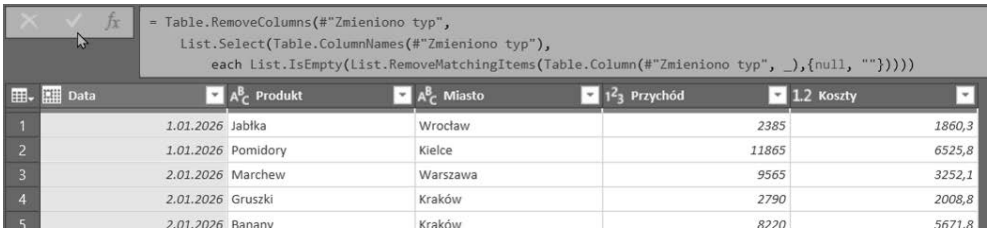


RYSUNEK 16.10. Lista kolumn do usunięcia

W powyższej formule za pomocą funkcji `List.Select` przechodzimy po wszystkich nazwach kolumn zwracanych przez funkcję `Table.ColumnNames` (rysunek 16.8). Te nazwy kolumn przekazujemy do funkcji `Table.Column` z użyciem słowa kluczowego `each`. Wtedy poprawne odwołanie do kolumny trafia do funkcji `List.RemoveMatchingItems`, która usuwa puste komórki i komórki z wartościami `null`. Następnie funkcja `List.IsEmpty` sprawdza, czy kolumna (lista) jest teraz pusta. Jeśli tak, funkcja zwraca wartość logiczną `TRUE`, w przeciwnym razie funkcja zwraca wartość logiczną `FALSE`. Te wartości przekazywane są do funkcji `List.Select`, a funkcja na ich podstawie decyduje, które elementy z listy wszystkich nazw kolumn zostają na liście.

W ten sposób utworzyliśmy listę kolumn do usunięcia. Pozostaje więc nam dodać do formuły funkcję `Table.RemoveColumns`, żeby usunąć obliczone przez siebie kolumny (rysunek 16.11):

```
= Table.RemoveColumns(#"Zmieniono typ", List.Select(Table.ColumnNames(#"Zmieniono typ"),
↪each List.IsEmpty(List.RemoveMatchingItems(Table.Column(#"Zmieniono typ", _),{null, ""}))))
```



```
= Table.RemoveColumns(#"Zmieniono typ",
    List.Select(Table.ColumnNames(#"Zmieniono typ"),
        each List.IsEmpty(List.RemoveMatchingItems(Table.Column(#"Zmieniono typ", _),{null, ""}))))
```

	Data	Produkt	Miasto	Przychód	Koszty
1	1.01.2026	Jabłka	Wrocław	2385	1860,3
2	1.01.2026	Pomidory	Kielce	11865	6525,8
3	2.01.2026	Marchew	Warszawa	9565	3252,1
4	2.01.2026	Gruszki	Kraków	2790	2008,8
5	2.01.2026	Banany	Kraków	8220	5671,8

RYSUNEK 16.11. Dane po usunięciu pustych kolumn

Teraz musimy już tylko zamienić nazwę zapytania np. na *Oczyszczone* i załadować je do Excela na nowy arkusz.

Najdłuższe ciągi

W tym przykładzie chcemy znaleźć najdłuższe ciągi wystąpień wartości `x` z uwzględnieniem remisów, a dodatkowo wypisać nagłówki kolumn, w których te iksy wystąpiły. `X` może oznaczać dzień bez wypadku, wartość sprzedaży powyżej lub poniżej ustalonego progu lub dowolny inny test logiczny/filtr.

Dane znajdują się w tabeli `Pracownicy` na arkuszu *Ciągi* (rysunek 16.12).

	A	B	C	D	E	F	G	H	I
1	Imię	1	2	3	4	5	6	ABC	123
2	Tom	Magazyn	x	x	Magazyn	Sortownia	Sortownia	x	x
3	Jerry	x			x	x	x	Magazyn	Magazyn
4	Eve	Recepcja	x	x	x	x	Sortownia	Sortownia	x
5	Walle	Sprzątanie	Recepcja	Recepcja	Transport	Transport	Sprzątanie	Sprzątanie	Sprzątanie
6	Bob								

RYSUNEK 16.12. Dane źródłowe w tabeli Pracownicy

Dla wykonywanych przez nas przekształceń istotne jest, że Jerry ma dwie puste komórki pomiędzy iksami, Tom ma dwa ciągi takiej samej długości, w przypadku Wallego nie ma żadnego iksa, a cały wiersz dotyczący Boba jest pusty.

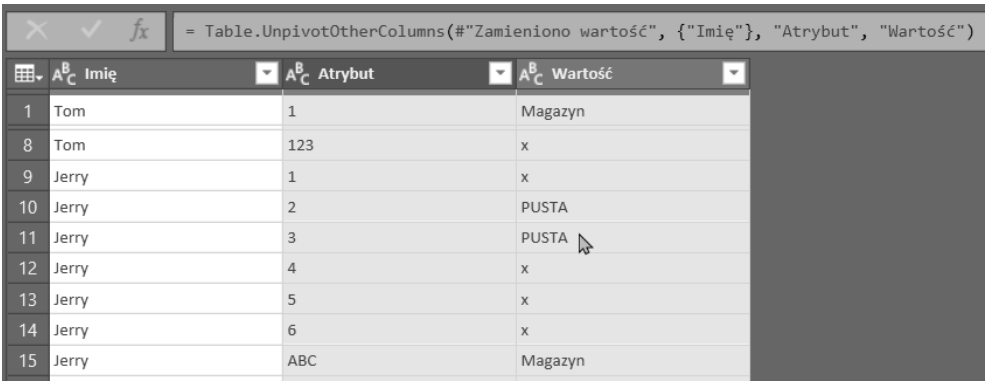
W nagłówkach kolumn może znaleźć się zarówno tekst, jak i liczby.

Wstępne oczyszczanie danych

Pierwszą operacją, jaką wykonamy po załadowaniu danych do Power Query, jest zduplikowanie zapytania (rysunek 7.24) oraz zmiana nazwy nowego zapytania na *Ciągi*. W tym zapytaniu chcemy usunąć inne kolumny poza kolumną `Imię`, więc przykładowo klikamy ją prawym przyciskiem myszy i w podręcznym menu wybieramy opcję *Usuń inne kolumny* (rysunek 13.15).

Po tej zmianie wracamy do oryginalnego zapytania i zmieniamy jego nazwę na *Przekształcenia*. W nim musimy zamienić wartości `null` na jakikolwiek tekst (najlepiej wartość inną od już istniejących w danych). Załóżmy, że zamienimy je na wartość `PUSTA`. Wystarczy, że zaznaczymy wszystkie kolumny (ewentualnie bez kolumny `Imię`) i na karcie *Narzędzia główne* klikniemy polecenie *Zamienianie wartości* (rysunek 2.7). Następnie w oknie *Zamienianie wartości* w polu *Wartość do znalezienia* wpiszemy wartość **`null`**, a w polu *Zamień na* wpiszemy ustalony tekst **`PUSTA`** (podobnie jak na rysunku 2.8).

Dopiero teraz możemy kliknąć prawym przyciskiem myszy kolumnę `Imię` i z podręcznego menu wybrać opcję *Anuluj przestawienie innych kolumn* (rysunek 11.4). Istotne było, żebyśmy przed tym przekształceniem zamienili wartości `null` na inne wartości, ponieważ przy anulowaniu przestawienia kolumn (i w wielu innych operacjach) komórki z wartościami `null` są pomijane. W tym przykładzie sprawiłoby to, że dwa ciągi iksów u Jerry'ego złączyłyby się w jeden ciąg, a tak są rozdzielone (rysunek 16.13)

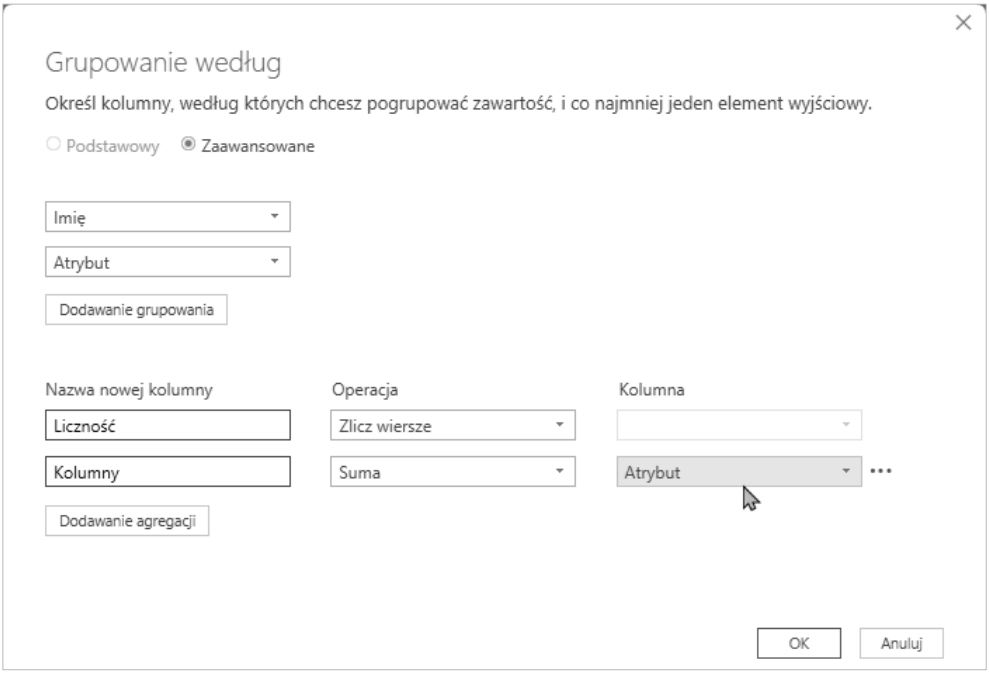


	Imię	Atrybut	Wartość
1	Tom	1	Magazyn
8	Tom	123	x
9	Jerry	1	x
10	Jerry	2	PUSTA
11	Jerry	3	PUSTA
12	Jerry	4	x
13	Jerry	5	x
14	Jerry	6	x
15	Jerry	ABC	Magazyn

RYSUNEK 16.13. Dane po anulowaniu przestawienia kolumn

Grupowanie danych

Teraz możemy zaznaczyć kolumnę `Imię` i `Wartość` oraz kliknąć polecenie *Grupowanie według* na karcie *Narzędzia główne* (rysunek 7.4). Następnie w oknie *Grupowanie według* ustalamy dwie agregacje. Pierwszą pozostaje domyślna operacja *Zlicz wiersze*, a druga może być dowolną możliwą operacją dla kolumny `Atrybut`. To w tej agregacji chcemy połączyć nazwy kolumn, dlatego nazywamy ją `Kolumny` (rysunek 16.14).



RYSUNEK 16.14. Operacje dotyczące grupowania

Teraz musimy zmienić wybraną operację `Suma` (użytą funkcję `List.Sum`) na operację łączącą tekst (funkcję `Text.Combine`), podobnie jak to robiliśmy w rozdziale 7. (patrz np. rysunek 7.10). Musimy zmienić też typ grupowania na grupowanie lokalne (podobnie jak na rysunku 7.23). Dlatego domyślnie utworzoną formułę:

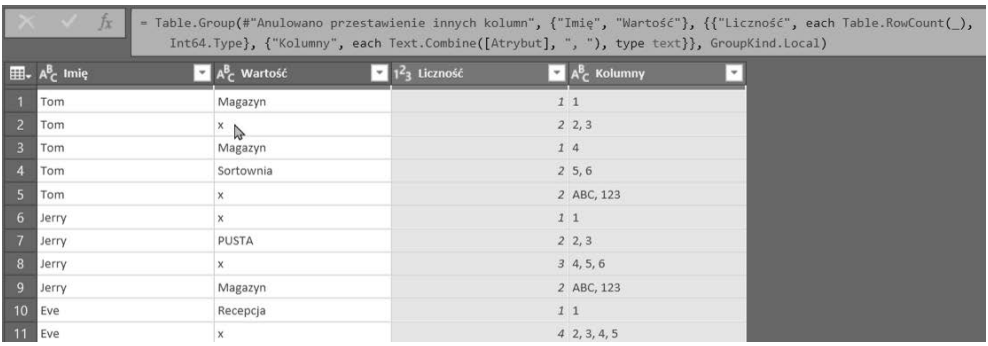
```
= Table.Group(#"Anulowano przestawienie innych kolumn", {"Imię", "Wartość"}, {{"Liczność",
↪each Table.RowCount(_), Int64.Type}, {"Kolumny", each List.Sum([Atrybut]), type text}})
```

zmieniamy na:

```
= Table.Group(#"Anulowano przestawienie innych kolumn", {"Imię", "Wartość"}, {{"Liczność",
↪each Table.RowCount(_), Int64.Type}, {"Kolumny", each Text.Combine([Atrybut], ", "), type
↪text}}, GroupKind.Local)
```

Ta formuła obliczy długość poszczególnych ciągów i przy okazji zapisze, w jakich kolumnach występowały (rysunek 16.15).

Teraz możemy odfiltrować ciągi, które nas nie interesują. W tym przykładzie wystarczy, że dla kolumny `Wartość` wybierzemy `x`.



	Imię	Wartość	Liczność	Kolumny
1	Tom	Magazyn	1	1
2	Tom	x	2	2, 3
3	Tom	Magazyn	1	4
4	Tom	Sortownia	2	5, 6
5	Tom	x	2	ABC, 123
6	Jerry	x	1	1
7	Jerry	PUSTA	2	2, 3
8	Jerry	x	3	4, 5, 6
9	Jerry	Magazyn	2	ABC, 123
10	Eve	Recepcja	1	1
11	Eve	x	4	2, 3, 4, 5

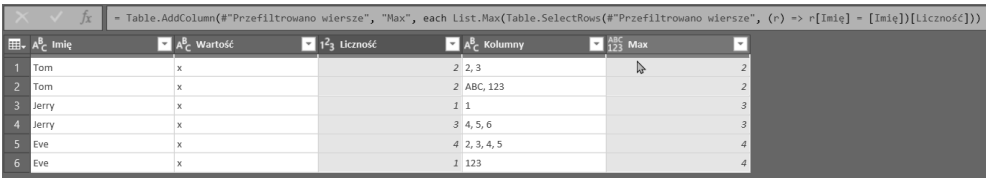
RYSUNEK 16.15. Pogrupowane dane

Własna funkcja obliczająca długość najdłuższego ciągu

Po odfiltrowaniu ciągów, które nie są w obszarze naszego zainteresowania, musimy w jakiś sposób znaleźć dla każdego z pracowników długość najdłuższego ciągu. Stworzymy własną funkcję wewnątrz istniejącej funkcji, podobnie jak w pierwszym przykładzie z tego rozdziału.

Zaczniemy od dodania kolumny niestandardowej (rysunek 4.19). W oknie *Kolumna niestandardowa* nadajemy nowej kolumnie nazwę `Max`, a jako formułę wpisujemy filtrowanie (funkcja `Table.SelectRows`), które będzie szukało wierszy dla danego pracownika, a następnie wyodrębniało maksymalną wartość (funkcja `List.Max`) dla kolumny `Liczność` (rysunek 16.16):

```
List.Max(Table.SelectRows(#"Przefiltrowano wiersze", (r) => r[Imię] = [Imię])[Liczność])
```

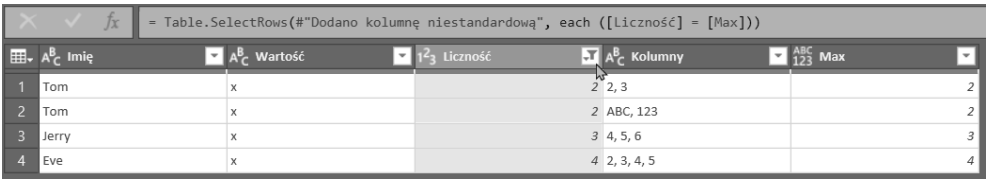


	Imię	Wartość	Liczność	Kolumny	Max
1	Tom	x	2	2, 3	2
2	Tom	x	2	ABC, 123	2
3	Jerry	x	1	1	3
4	Jerry	x	3	4, 5, 6	3
5	Eve	x	4	2, 3, 4, 5	4
6	Eve	x	1	123	4

RYSUNEK 16.16. Obliczenie najdłuższego ciągu dla poszczególnych pracowników

Teraz wystarczy nałożyć filtr na kolumnę `Liczność`, wybierając dowolną liczbę, a następnie zmodyfikować ten filtr, by wartość była równa kolumnie `Max` (rysunek 16.17):

```
= Table.SelectRows(#"Dodano kolumnę niestandardową", each ([Liczność] = [Max]))
```



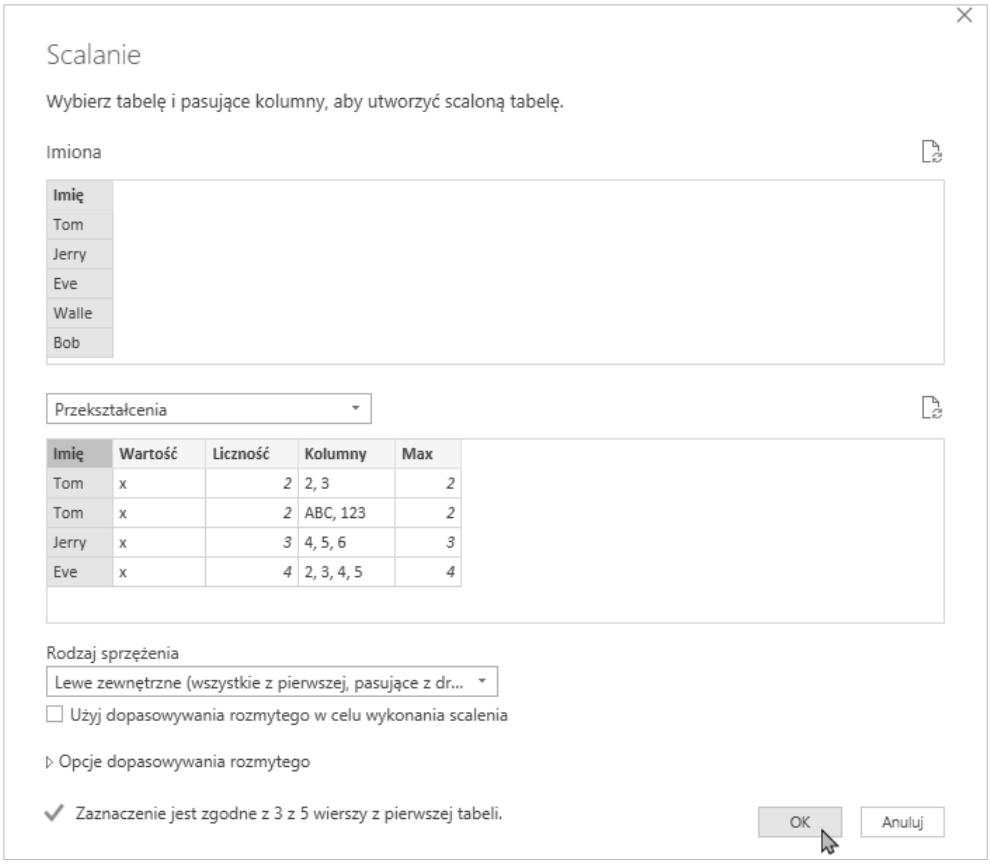
	Imię	Wartość	Liczność	Kolumny	Max
1	Tom	x	2	2, 3	2
2	Tom	x	2	ABC, 123	2
3	Jerry	x	3	4, 5, 6	3
4	Eve	x	4	2, 3, 4, 5	4

RYSUNEK 16.17. Lista najdłuższych ciągów

Scalanie danych

Znaleźliśmy najdłuższe ciągi, ale na liście nie widzimy pracowników Wallego ani Boba. Wynika to z tego, że w dotyczących ich wierszach nie był wpisany żaden iks (nie było w nich wartości spełniających nasz szukany warunek). Dlatego na początku utworzyliśmy duplikat zapytania z wszystkimi pracownikami. Przechodzimy do niego, a następnie na karcie *Narzędzia główne* klikamy polecenie *Scal zapytania* (rysunek 6.7).

W nowo otwartym oknie *Scalanie* wybieramy na dole zapytanie *Przekształcenia* i łączymy oba zapytania po kolumnie `Imię` (rysunek 16.18).



RYSUNEK 16.18. Scalanie zapytań

Po scaleniu pozostaje nam rozwinąć kolumnę `Przekształcenia`, wyodrębniając z niej kolumny nazwane `Kolumny` i `Max`.

Na koniec możemy jeszcze zamienić wartości `null` w kolumnie `Max` na zera, podobnie jak robiliśmy to na początku tego przykładu. Teraz możemy już załadować stworzone przez nas zapytania do Excela.

Ponieważ stworzyliśmy na jeden raz dwa zapytania, możemy je załadować jedynie na nowych arkuszach. Arkusz *Przekształcenia* jest nam niepotrzebny, a tabelę z arkusza *Imiona* wycinamy (*Ctrl+X*) i wklejamy na arkusz *Ciągi*. Po tym możemy usunąć arkusz *Imiona*.

W tym rozdziale omówiliśmy trzy przykłady, których głównym zadaniem było wykorzystanie różnych przekształceń poznanych do tej pory, czyli m.in. grupowaliśmy i scalaliśmy dane, anulowaliśmy przestawienie kolumn, używaliśmy własnych funkcji wewnątrz istniejących funkcji Power Query, filtrowaliśmy i sortowaliśmy dane oraz wykonywaliśmy wiele innych przekształceń, sprawdzając, czy klient jest powracający, usuwając puste kolumny oraz wyznaczając najdłuższe ciągi.

W następnym rozdziale usystematyzujemy wiedzę na temat optymalizacji kodu M i dobrych praktyk podczas jego pisania.

Power Query w Excelu

Automatyzacja, o jakiej marzysz

Dane są dziś powszechnie dostępne, ale rzadko w takiej formie, w jakiej ich potrzebujemy. Na szczęście z pomocą przychodzi Power Query. To nowoczesne narzędzie idealne do analityki w Excelu i Power BI, które zamienia żmudne, wielogodzinne „czyszczenie" danych w automatyczny, błyskawiczny proces.

Ta książka stanowi kompleksowy przewodnik po wszystkich aspektach pracy z Power Query, od podstaw po zaawansowane techniki optymalizacji.

- **Opanuj podstawy pracy z danymi: sprawnie pobieraj informacje z różnych źródeł, od formatów takich jak .txt i .csv, przez proste bazy danych, aż po pliki .xml czy całe foldery**
- **Zapoznaj się z zaawansowanymi przekształceniami, które czynią Power Query potężnym: między innymi scalaniem i dołączaniem zapytań, grupowaniem danych, odpiwotowaniem (Unpivot) kolumn**
- **Naucz się języka M i funkcji niestandardowych: pisz własne formuły i twórz parametry, dzięki którym Twoje raporty staną się elastyczne i bardziej odporne na zmiany w źródłach**
- **Poznaj proces optymalizacji i dobre praktyki: mechanizmy diagnostyczne i techniki przyspieszania zapytań, aby nawet przy setkach tysięcy wierszy Twój arkusz pracował błyskawicznie**

Zapomnij o ręcznym kopiowaniu danych i błędach wynikających z rutyny. Ten podręcznik będzie idealnym wyborem nie tylko dla analityków, ale dla każdego użytkownika Excela ceniącego swój czas.

Przestań walczyć z danymi. Zacznij je analizować!

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 978-83-289-3618-8
9 788328 936188
Cena: 89,00 zł