Dane – informacja o rzeczywistości, która nadaje się do umieszczenia w systemie komputerowym – ma postać cyfrową.

Baza danych – uporządkowany zbiór danych na temat danego fragmentu rzeczywistości.

System zarządzania bazą danych (SZBD) – (DataBase Management System – DBMS) – program, narzędzie, aplikacja (lub ich zbiór) pozwalających i wspomagających tworzenie i zarządzanie bazami danych (np. pakiet XAMPP, MS Access itd.).

System bazy danych – system zarządzania bazą danych (konkretna aplikacja) obejmujący i obsługujący konkretną bazę danych.

Podstawowe operacje na danych w bazie danych:

- Wprowadzanie
- Zapamiętywanie i przechowywanie
- Wyszukiwanie i analizowanie
- Dodawanie i usuwanie
- Aktualizowanie

SZBD mogą być lokalne, serwerowe albo chmurowe. Jednym z takich serwerowych SZBD jest Microsoft SQL Server, MySQL lub XAMPP. Lokalnym SZBD jest np. MS Access – można też lokalnie używać funkcji XAMPPa.

Cechy systemu baz danych:

- Baza danych musi mieć jasno **określoną strukturę** – jak dane są w niej poukładane, co gdzie jest i co się z czym łączy
- Dane w bazie danych muszą być **trwałe**, czyli musimy mieć możliwość przechowywania ich w niej przez długi czas, jak w magazynie
- Musimy mieć możliwość realizowania **operacji** na danych – tworzenia, wyszukiwania, aktualizowania i usuwania
- Dane muszą być **niezależne** od bazy danych – informacje powinny dać się przechować i zapisać w różnych systemach bazodanowych
- Dane w bazie muszą być **bezpieczne** – osoby niepowołane nie powinny mieć do nich dostępu
- Dane w bazie muszą być **integralne** – zgodne z rzeczywistością; kiedy rzeczywistość się zmienia, powinny zmienić się także wpisy w bazie
- Dane w bazie muszą być **spójne** – zapewniamy poprawność danych i odporność na anomalie (np. w polu na numer telefonu nie powinny pojawiać się litery, tylko same cyfry)
- Baza musi być **skalowalna** – powinniśmy mieć możliwość rozbudowy i powiększania bazy

Modele baz danych:

- Jednorodny, jednotabelowy – charakteryzują się nadmiarowością danych i utrudnionym wyszukiwaniem informacji
- Hierarchiczny, drzewiasty – mamy elementy nadrzędne i podrzędne
- Sieciowy, rozproszony – elementy łączą się ze sobą na wielu poziomach w sieć, skomplikowany, ale łatwo w nim wyszukiwać informacje
- **Relacyjny** – dane w bazie są zgrupowane w relacje, między którymi zachodzą pewne zależności

Relacyjny model baz danych opiera się na tabelach (encjach, relacjach), które są zbudowane z kolumn (atrybutów) i wierszy (rekordów, krotek). Pomiędzy tabelami zachodzą powiązania (relacje).

Tabela – podstawowy zbiór danych w relacyjnej bazie danych, reprezentujący wybrany fragment rzeczywistości i zbierający obiekty o podobnym typie (np. tabela z wszystkimi filmami w filmotece). Każda tabela w bazie danych musi mieć jednoznaczną i unikalną nazwę. Dane w tabeli nie powinny się powtarzać.

Atrybut – kolumna w tabeli, ma określony typ danych (np. tekstowy, liczbowy, data), który w niej można umieścić. Atrybut wskazuje na pewną cechę opisywanego obiektu (np. tytuł filmu, jego gatunek, czas trwania, rok produkcji itd.). Również musi mieć jednoznaczną nazwę, która jest unikalna w obrębie jednej tabeli (tzn. możemy mieć dwie różne tabele, w których pojawią się kolumny o tych samych nazwach, ale w jednej tabeli już tak nie może być).

Rekord – wiersz w tabeli (krotka), pojedynczy wpis w tabeli, rozciągający się na wszystkie jej kolumny (np. konkretny film: „Fantastyczny Pan Lis", „familijny", „92 minuty", „2009 rok").

Pole – najmniejszy element tabeli, pojedyncza jej komórka. Pole w tabeli powinno zawierać wartość atomową, tzn. niepodzielną (np. zamiast „Wes Anderson" w jednym polu powinniśmy zapisać „Wes" w jednym, a „Anderson" w drugim).

Klucz główny – taka kolumna, dzięki której możemy jednoznacznie zidentyfikować każdy konkretny wiersz w tabeli (np. ID filmu – taki „stworzony" klucz nazywamy kluczem sztucznym. W przypadku np. ludzi można zastosować ich PESEL – taki klucz jest nazywany naturalnym). Jeżeli klucz główny składa się z kilku kolumn to jest nazywany złożonym.

Klucz obcy – taka kolumna w tabeli, która ma być odwołaniem do innej tabeli. Przykład:

Tabela - film

ID filmu	***Tytuł***	***Reżyser***
1	Fantastyczny Pan Lis	3

ID filmu i ID reżys. są kluczami głównymi. Kolumna „Reżyser" w tabeli „film" jest kluczem obcym. Pomiędzy tymi tabelami może zajść pewna relacja.

Tabela - reżyser

ID reżys.	***Imię***	***Nazwisko***
3	Wes	Anderson

Relacja – logiczny związek między tabelami, posiada liczność (krotność), może być obowiązkowa lub opcjonalna. Typy relacji:

- Jeden do jednego, 1-1 – jednemu elementowi z tabeli A przypada tylko jeden element z tabeli B (np. jeden reżyser może mieć tylko jeden numer PESEL (albo jego zagraniczny odpowiednik).
- Jeden do wielu, 1-N – najczęstszy typ relacji, jednemu elementowi z tabeli A przypada jeden lub więcej elementów z tabeli B (np. jeden reżyser mógł nakręcić wiele filmów, ale jeden film ma jednego reżysera (w uproszczeniu)).
- Wiele do wielu, N-M – wielu elementom z tabeli A może przypadać wiele elementów z tabeli B (np. wielu reżyserów mogło brać udział w jakiejś gali rozdania nagród, a wiele gali rozdania

nagród mogło gościć wielu reżyserów). Raczej unika się takiego typu relacji, rozbijając relację N-M na dwie (lub więcej) relacji 1-N.

Normalizacja bazy danych to proces zamiany i przekształcania bazy tak, aby z jednego, dużego zbioru danych, w którym występuje nadmiarowość atrybutów i danych, stał się zbiorem większej ilości pomniejszych tabelek. Ma wiele poziomów, ale najczęściej kończy się na trzecim. Przykład:

Tytuł filmu	Reżyser	Czas trwania	Ocena	Recenzent	Gatunek	Kraj pochodzenia
„Fantastyczny Pan Lis” „Wyspa Psów”	Wes Anderson	91 minut 102 minuty	5/5 4/5	Adam Kwiatkowski	Familijny Familijny	USA USA

Ta tabela nie jest znormalizowana. W jednej tabeli są informacje o wielu „rodzajach” obiektów, o filmie, o reżyserze i o recenzji. Ponadto, w jednym polu pojawiają się informacje o wielu obiektach (np. wiele tytułów filmów). Najpierw rozdzielamy dane na poszczególne rekordy:

ID	Tytuł filmu	Reżyser	Czas trwania	Ocena	Recenzent	Gatunek	Kraj pochodzenia
1	„Fantastyczny Pan Lis”	Wes Anderson	91 minut	5/5	Adam Kwiatkowski	Familijny	USA
2	„Wyspa Psów”	Wes Anderson	102 minuty	4/5	Adam Kwiatkowski	Familijny	USA

Teraz należy „zatomizować” dane, czyli rozdzielić pola złożone z kilku informacji na mniejsze – dopiero teraz jest to **pierwsza postać normalna (1NF)**

ID	Tytuł filmu	Imię reż.	Nazw. reż.	Czas trwania	Ocena	Imię rec.	Nazw. rec.	Gatunek	Kraj pochodzenia
1	„Fantastyczny Pan Lis”	Wes	Anderson	91 minut	5/5	Adam	Kwiatkowski	Familijny	USA
2	„Wyspa Psów”	Wes	Anderson	102 minuty	4/5	Adam	Kwiatkowski	Familijny	USA

Następnie musimy podzielić tabelę na kilka mniejszych – każda tabela ma przechowywać informacje tylko o określonym typie obiektów (np. tabela o filmie, tabela o reżyserze, tabela o recenzencie). To będzie **druga postać normalna (2NF)**

Tabela film

ID filmu	ID reżysera	ID recenzji	Tytuł	Czas trwania	Gatunek	Kraj pochodzenia
1	1	1	„Fantastyczny Pan Lis”	91 minut	Familijny	USA
2	1	2	„Wyspa Psów”	102 minuty	Familijny	USA

Tabela reżyser

ID reżysera	Imię	Nazwisko
1	Wes	Anderson

Tabela recenzja

ID recenzji	Imię	Nazwisko	Ocena
1	Adam	Kwiatkowski	5/5
2	Adam	Kwiatkowski	4/5

Trzecia postać normalna (3NF) mówi o tym, że każda kolumna, która nie jest kluczem głównym, nie powinna „zależeć" od innej kolumny. Innymi słowy, jeżeli możemy coś rozdzielić na mniejsze tabele, powinniśmy to zrobić, aby uniknąć **redundancji** (czyli powtarzalności danych). W powyższym przykładzie powinniśmy rozdzielić np. tabelę recenzji tak, aby zamienić imię i nazwisko recenzenta na ID recenzenta, a potem przygotować dodatkową tabelę o recenzentach.

Przykładowe (ważne!) typy danych:

- Integer – całkowitoliczbowy (1, 2, 3)
- Float – zmiennoprzecinkowy (3.14, 21.37)
- Char – tekstowy o stałej długości znaków (do 255 liter)
- Varchar – tekstowy o zmiennej długości znaków (do 255 liter)
- Date – data (rrrr-mm-dd, np. 1999-11-06)

Pozostałe typy danych:

- Decimal – zmiennoprzecinkowy o określonej precyzji (tj. ilości miejsc po przecinku)
- Time – czas
- Datetime – data i godzina
- Enum – wyliczanie po kolei
- Text – duży blok tekstu
- Tinyint – liczba od -128 do 127 lub od 0 do 255
- Bit – wartość 0, 1 lub NULL

Wartość NULL – wartość mówiąca „nic tu nie ma". Wypełniając bazę danych musimy podać wszystkie wartości we wszystkich polach. Jeżeli nie mamy danych, które możemy wstawić w dane pole (np. data śmierci reżysera, który jeszcze żyje), wstawimy tam NULL. Wartość NULL i wartość „ " (spacja, pusty znak) to nie to samo.

Modelowanie bazy danych obejmuje przygotowanie projektu bazy danych, przygotowanie modelu w jakimś środowisku, następnie zbudowanie działającej bazy w wybranym systemie. Najczęściej do tworzenia modeli baz danych używa się tzw. **notacji ER** (Entity Relationship, związki encji) aby stworzyć diagram **ERD**. Modelowanie możemy prowadzić od ogółu do szczegółu (top-down), od szczegółu do ogółu (bottom-up) lub od elementów najważniejszych do tych mniej istotnych (inside-out).

Kolejne kroki modelowania bazy danych:

1. Zdefiniowanie przeznaczenia bazy (po co, dla kogo)
2. Zebranie informacji o rzeczywistości (jakie dane będziemy umieszczać w bazie)
3. Zdefiniowanie encji i nadanie im nazw, określenie ich atrybutów
4. Sprawdzenie poprawności pod względem redundancji danych (czy nie mamy niepotrzebnie powtarzających się informacji)
5. Określenie typów danych dla atrybutów encji
6. Określenie kluczy podstawowych dla każdej encji
7. Opisanie związków między encjami
8. Wykonanie diagramu ERD
9. Sprawdzenie poprawności, wykonanie poprawek
10. Implementacja modelu w systemie bazodanowym

Diagram ERD składa się z następujących symboli:

W diagramie **Chena** elementy diagramu łączy się za pomocą linii prostych. Przy połączeniach encji ze związkami określa się ich liczność.

W notacji **Martina** encje łączy się liniami prostymi z symboliką określającą liczność związku oraz jego opcjonalność. W tabeli 3.2 zamieszczono wykaz oznaczeń powiązań.

Tabela 3.2. Rodzaje oznaczeń powiązań w notacji Chena

powiązanie określające liczność relacji 1 do 1	
powiązanie określające liczność relacji 1 do N	
powiązanie określające liczność relacji N do M	

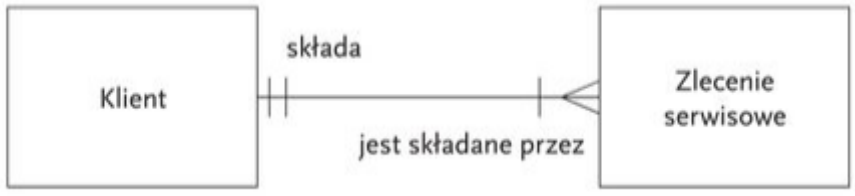
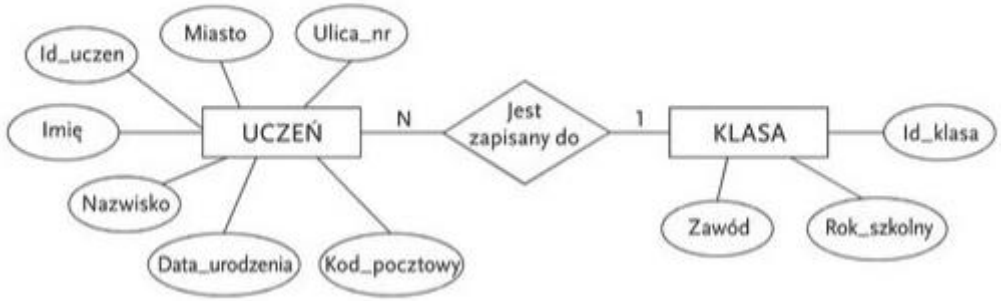
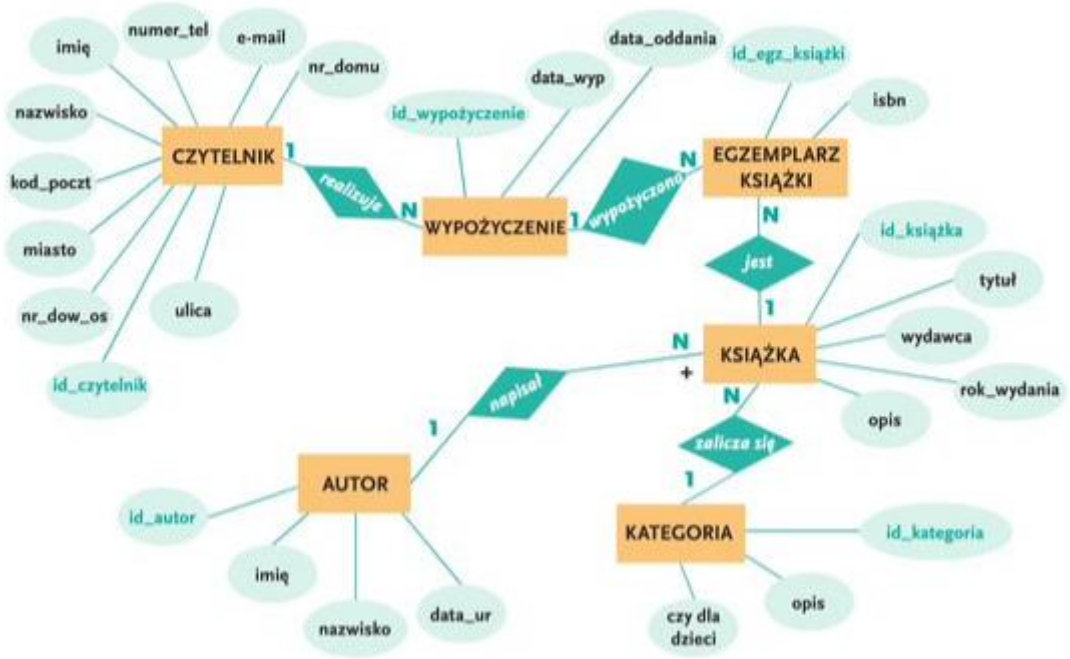


Diagram ER opisujący encje Uczeń i Klasa wraz z opisującymi je atrybutami zapisany w notacji Chena pokazano na rys. 3.7.





Rys. 4.1. Przykład diagramu ER w notacji Chena

SQL (Structured Query Language) – ustrukturalizowany język zapytań, służy do pisania poleceń do bazy danych, celem wykonania różnych operacji. Polecenia z tego języka można podzielić na cztery rodziny/grupy:

- **DDL** – Data Definition Language – język definiowania danych – służy do tworzenia, modyfikowania i usuwania całych obiektów (np. baz danych lub ich tabel). Polecenia:
  - **CREATE** – stwórz obiekt
  - **ALTER** – zmień obiekt
  - **DROP** – usuń obiekt
- **DML** – Data Manipulation Language – język manipulowania danymi – służy do wprowadzania, aktualizowania i kasowania danych w tabelach. Polecenia:
  - **INSERT** – wstaw dane do tabeli
  - **UPDATE** – zaktualizuj dane w tabeli
  - **DELETE** – usuń dane z tabeli
- **DCL –** Data Control Language – język kontroli (dostępu do) danych – służy do określania, kto ma mieć jakie uprawnienia przy jakich obiektach bazy danych

- **DQL** – Data Query Language – język zapytań do danych – służy do tworzenia poleceń wykonujących tzw. **selekcję** (wybieranie pionowe – określenie, które kolumny z tabeli mają zostać wyświetlone i zwrócone jako wynik zapytań) oraz **projekcję** (wybieranie poziome – określenie, które wiersze z tabeli mają zostać wyświetlone i zwrócone). Polecenie **SELECT**. W ramach polecenia SELECT istnieje wiele warunków, klauzuli i funkcji, które to polecenie wzbogacają, przykładowo:
    - **Klauzula WHERE** – określa warunki, które wiersz tabeli musi spełnić, by zostać zwrócony jako wynik polecenia SELECT, filtruje tabelę źródłową
    - **Operatory logiczne** – AND (wszystkie warunki muszą być spełnione), OR (chociaż jeden warunek musi być spełniony), NOT (musi być spełnione przeciwieństwo warunku)
    - **Operatory arytmetyczne** - = (równy), > (większy od), < (mniejszy od), >= (większy-równy od), <= (mniejszy-równy od), <> (różny od)
    - **Złączenia JOIN** – istnieje kilka wariantów funkcji JOIN – służy do powiązywania ze sobą kilku tabel, dzięki wykorzystaniu kluczy obcych
    - **Funkcje agregujące** – funkcje kolumnowe, zwracające wynik jakiegoś obliczenia, przeprowadzonego na zadanej kolumnie
        - **COUNT** – zliczenie wystąpień
        - **SUM** – sumowanie wartości
        - **AVG** – średnia z wartości
        - **MIN** – minimalna wartość
        - **MAX** – maksymalna wartość
    - **Grupowanie** – klauzula **GROUP BY** pozwala na zgrupowanie wyników, według powtarzających się wartości w którejś z kolumn
    - **Klauzula HAVING** – drugi poziom filtrowania, który może nastąpić wyłącznie po zgrupowaniu wyników

Przykładowe polecenia SQL:

Polecenie	Wynik
SELECT * FROM film	Z tabeli film zostaną zwrócone wszystkie kolumny i wszystkie wiersze – pokazana zostanie cała tabela
SELECT tytuł, ID_reżyser FROM film	Z tabeli film zostaną zwrócone tylko kolumny tytuł i ID_reżyser oraz wszystkie wiersze
SELECT tytuł, ID_reżyser FROM film WHERE ID_reżyser = 2	Z tabeli film zostaną zwrócone tylko kolumny tytuł i ID_reżyser oraz tylko te wiersze, w których wartość kolumny ID_reżyser jest równa 2
SELECT tytuł, ID_reżyser FROM film WHERE tytuł LIKE “A%”	Z tabeli film zostaną zwrócone tylko kolumny tytuł i ID_reżyser oraz tylko te wiesze, w których tytuł filmu zaczyna się od litery A (wielkość liter ma znaczenie). % oznacza dowolny ciąg znaków
SELECT tytuł, ID_reżyser FROM film WHERE tytuł NOT LIKE “%s” AND ID_reżyser <> 3	Z tabeli film zostaną zwrócone tylko kolumny tytuł i ID_reżyser oraz tylko te wiersze, w których tytuł filmu NIE kończy się małą literą s ORAZ ID reżysera jest różne od 3
SELECT tytuł, czas_trwania FROM film WHERE tytuł LIKE “_ _ _ _”	Z tabeli film zostaną zwrócone tylko kolumny tytuł i czas_trwania oraz tylko te wiersze, w których tytuł składa się z czterech liter (znak „_”

OR czas_trwania BETWEEN 90 AND 140	oznacza dokładnie jedną literę[1]) LUB czas trwania filmu (w minutach) jest w przedziale od 90 do 140
SELECT tytuł FROM film ORDER BY tytuł DESC	Z tabeli film zostanie zwrócona tylko kolumna tytuł, posortowana malejąco, w kolejności odwrotnej do alfabetycznej
SELECT tytuł FROM film LIMIT 5	Z tabeli film zostanie zwrócona tylko kolumna tytuł, ale tylko pięć pierwszych wierszy tej tabeli
SELECT CONCAT (reżyser.imię, " ", reżyser.nazwisko) AS "Twórca" FROM reżyser	Z tabeli reżyser zostaną pobrane kolumny z imieniem i nazwiskiem reżysera, ale zostaną one wyświetlone jako jedna, duża kolumna, w której imię i nazwisko zostaną oddzielone spacją (stąd zapis „ „) oraz nazwa tej kolumny zostanie zmieniona aliasem AS na „Twórca"
SELECT film.tytuł, reżyser.imię, reżyser.nazwisko FROM film JOIN reżyser ON film.ID_reżyser = reżyser.ID_reżyser	Z tabeli film zostanie pobrana kolumna z tytułem filmu. Z tabeli reżyser zostaną pobrane kolumny o imieniu i nazwisku reżysera. Dzięki zastosowaniu JOIN ON można powiązać ze sobą dwie tabele tak, aby znaleźć w jednej z nich wartości klucza obcego, które odpowiadają wartościom klucza głównego tej drugiej – w tym przykładzie wiążemy ID reżysera z tabeli film z ID reżysera z tabeli reżyser i na podstawie tego powiązania możemy skojarzyć tytuł filmu (bo w tabeli film znamy ID reżysera, który go zrobił) z imieniem i nazwiskiem tegoż reżysera (bo wiemy pod jakim ID występują w tabeli reżyser)
SELECT film.tytuł, reżyser.imię, reżyser.nazwisko FROM film WHERE (reżyser.nazwisko LIKE "%son" AND reżyser.imię NOT LIKE "_ _ _") OR film.czas_trwania >= 90 JOIN reżyser ON film.ID_reżyser = reżyser.ID_reżyser	Z tabeli film zostanie pobrana kolumna z tytułem filmu. Z tabeli reżyser zostaną pobrane kolumny o imieniu i nazwisku reżysera. Zwrócone zostaną te wiersze, gdzie: • ALBO nazwisko reżysera kończy się na „son" ORAZ imię reżysera NIE składa się z trzech liter • ALBO czas trwania filmu jest większy bądź równy 90 minut
SELECT COUNT(ID_filmu) FROM film	Zwrócona zostanie wartość liczbowa równa ilości filmów w tabeli film
SELECT SUM/AVG/MIN/MAX(czas_trwania) FROM film WHERE ID_reżyser = 3	Zwrócona zostanie sumaryczna/średnia/minimalna/maksymalna długość wszystkich filmów reżysera o ID równym 3
SELECT COUNT(film.tytuł) AS „Liczba filmów", reżyser.nazwisko FROM film	Zwrócona zostanie liczba filmów poszczególnych reżyserów – grupowanie następuje po nazwiskach reżyserów[2] (dzięki zastosowaniu AS

[1] Spacje są dodane dla przejrzystości, nie powinny pojawić się w prawdziwym poleceniu

[2] Problem – jeżeli w naszej tabeli byłoby dwóch (lub więcej) reżyserów o tym samym nazwisku, ale różnych imionach, to zostaliby oni zgrupowani razem (np. mamy reżysera Jana Kowalskiego i Karola Kowalskiego – każdy z nich zrobił po jednym filmie, ale to polecenie zwróciłoby wynik „2 Kowalski".

JOIN reżyser ON film.ID_reżyser = reżyser.ID_reżyser GROUP BY reżyser.nazwisko	(aliasu) kolumnie zostanie zmieniona nazwa na „Liczba filmów”[3])
SELECT COUNT(filmy.ID_film), reżyser.kraj_pochodzenia FROM filmy WHERE reżyser.data_smierci IS NOT NULL JOIN reżyser ON filmy.ID_reżyser = reżyser.ID_reżyser GROUP BY reżyser.kraj_pochodzenia HAVING reżyser.kraj_pochodzenia NOT LIKE “Korea Północna”	Policz ile filmów zrobili reżyserowie (ale tylko ci, którzy już nie żyją) pochodzący z różnych krajów, zgrupuj wyniki właśnie według ich krajów, wypisz tę wartość obok kraju pochodzenia, ale nie wyświetlaj wartości tych wyliczeń, które dotyczą reżyserów z Korei Północnej

Więcej przykładów i wyjaśnień możecie znaleźć w podręcznikach oraz na w3schools.com

[3] Jeżeli nie zastosowalibyśmy aliasu, nazwa kolumny, która korzysta z jakiejś funkcji, zostanie ustalona na treść tej funkcji właśnie (zamiast napisu „Liczba filmów” widzielibyśmy „COUNT(film.tytuł)”)