



DML

(Data Manipulation Language)

Podstawowe zapytania

SELECT

dimon.work/kurs.html

Polecenia i przykłady będziemy ćwiczyć na przykładzie bazy sakila

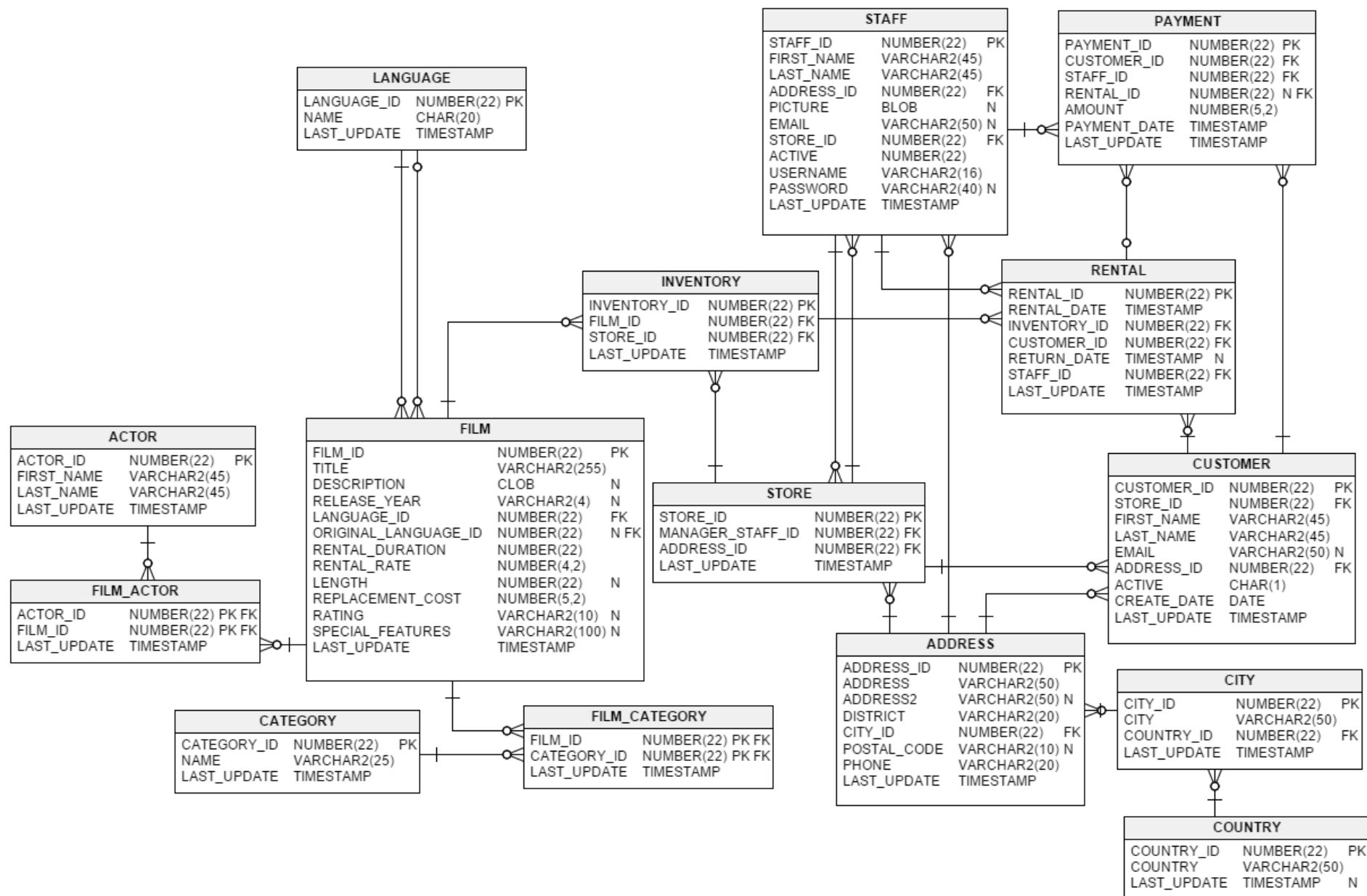
Jeżeli nie pamiętasz jak dodać bazę danych w phpmyadmin

<https://dimon.work/kurs/phpmyadmin.pdf>

Baza danych Sakila to ładnie znormalizowany schemat modelujący wypożyczalnię DVD, zawierający takie elementy jak filmy, aktorzy, relacje film-aktor oraz centralną tabelę zapasów (inventory), która łączy filmy, sklepy i wypożyczalnie.

Sakila-Schema

dimon.work



Pełna selekcja

dimon.work

```
SELECT * FROM `film`
```

Gwiazdka – operator który wybiera wszystkie kolumny z tabeli

```
SELECT title, rating FROM `film`
```

Wskazując nazwę kolumn, zamiast gwiazdki, wyświetlą się nam tylko konkretne kolumny.

The screenshot shows a database management interface with a toolbar at the top containing buttons for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, and Triggers. The main area displays the results of a query: 'SELECT * FROM `film`'. A status bar indicates 'Showing rows 0 - 24 (1000 total, Query took 0.0003 seconds.)'. Below the query, there are links for Profiling, Edit inline, Edit, Explain SQL, Create PHP code, and Refresh. A control bar shows 'Number of rows: 25', 'Filter rows: Search this table', and 'Sort by key: None'. An 'Extra options' button is also present. The results are shown in a table with columns: film_id, title, description, release_year, language_id, original_language_id, rental_duration, and rental_rate. The first three rows are visible, each with edit, copy, and delete icons.

| | film_id | title | description | release_year | language_id | original_language_id | rental_duration | rental_rate |
|---|---------|------------------|---|--------------|-------------|----------------------|-----------------|-------------|
| <input type="checkbox"/> Edit Copy Delete | 1 | ACADEMY DINOSAUR | A Epic Drama of a Feminist And a Mad Scientist who... | 2006 | 1 | NULL | 6 | 0.99 |
| <input type="checkbox"/> Edit Copy Delete | 2 | ACE GOLDFINGER | A Astounding Epistle of a Database Administrator A... | 2006 | 1 | NULL | 3 | 4.99 |
| <input type="checkbox"/> Edit Copy Delete | 3 | ADAPTATION HOLES | A Astounding Reflection of a Lumberjack And a Car ... | 2006 | 1 | NULL | 7 | 2.99 |

Selekcja i prosta arytmetyka

dimon.work

```
SELECT title,  
rental_rate*rental_duration  
FROM `film`;
```

W SQL można korzystać ze wszystkich podstawowych operacji matematycznych, wykonywać obliczenia i stosować funkcje

The screenshot shows a database management interface with the following components:

- Header:** Server: 127.0.0.1 » Database: sakila » Table: film
- Navigation Bar:** Browse, Structure, SQL, Search, Insert, Export, Import
- Status Bar:** Showing rows 0 - 24 (1000 total, Query took 0.0003 seconds.)
- SQL Editor:** `SELECT title, rental_rate*rental_duration FROM `film`;`
- Options:** Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
- Row Controls:** 1 > >> | Number of rows: 25 | Filter rows: Search this table
- Table View:** A table with columns 'title' and 'rental_rate*rental_duration'.

| | title | rental_rate*rental_duration |
|--------------------------|------------------|-----------------------------|
| <input type="checkbox"/> | ACADEMY DINOSAUR | 5.94 |
| <input type="checkbox"/> | ACE GOLDFINGER | 14.97 |
| <input type="checkbox"/> | ADAPTATION HOLES | 20.93 |
| <input type="checkbox"/> | AFFAIR PREJUDICE | 14.95 |
| <input type="checkbox"/> | AFRICAN EGG | 17.94 |
| <input type="checkbox"/> | AGENT TRUMAN | 8.97 |
| <input type="checkbox"/> | AIRPLANE SIERRA | 29.94 |
| <input type="checkbox"/> | AIRPORT POLLOCK | 29.94 |

SELECT DISTINCT

dimon.work

```
SELECT DISTINCT district, city_id  
FROM address;
```

to polecenie w SQL, które służy do wybierania **unikalnych** wartości z określonej kolumny (lub wielu kolumn) w tabeli bazy danych. Usuwa zduplikowane wiersze z wyniku zapytania.

Porównaj wynik zapytania wyżej z zapytaniem bez **DISTINCT**

Server: 127.0.0.1 » Database: sakila » Table: address

Browse Structure SQL Search Insert

Show query box

✓ Showing rows 0 - 24 (601 total, Query took 0.0003 seconds.)

```
SELECT DISTINCT district, city_id FROM address;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

1 > >> | Number of rows: 25 Filter rows: Search

Extra options

| | | | | district | city_id |
|--------------------------|------|------|--------|------------|---------|
| <input type="checkbox"/> | Edit | Copy | Delete | Alberta | 300 |
| <input type="checkbox"/> | Edit | Copy | Delete | QLD | 576 |
| <input type="checkbox"/> | Edit | Copy | Delete | Nagasaki | 463 |
| <input type="checkbox"/> | Edit | Copy | Delete | California | 449 |
| <input type="checkbox"/> | Edit | Copy | Delete | Attika | 38 |

SELECT COUNT()

dimon.work

```
SELECT COUNT(DISTINCT district)
FROM address;
```

To zapytanie policzy liczbę unikalnych wartości w kolumnie dzielnicy z tabeli adresów.

Wyświetl rezultat:

```
SELECT COUNT(district)
FROM address
WHERE district = 'California';
```

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM address;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

Extra options

COUNT(*)

603

Your SQL query has been executed successfully.

```
SELECT COUNT(district) FROM address;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

Extra options

COUNT(district)

603

Your SQL query has been executed successfully.

```
SELECT COUNT(DISTINCT district) FROM address;
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

Extra options

COUNT(DISTINCT district)

378

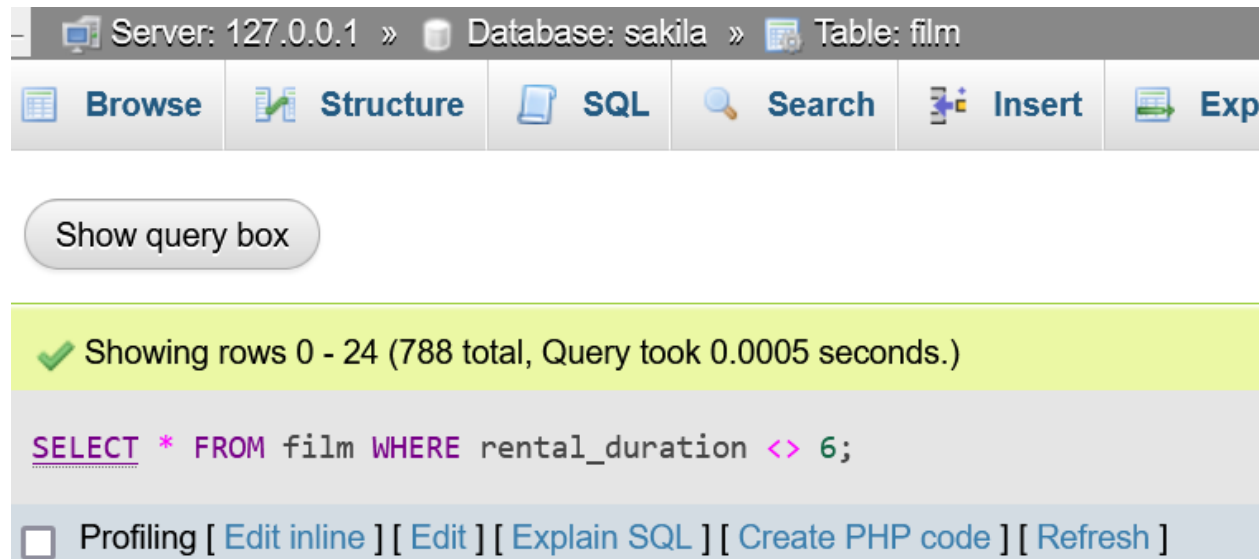
Warunkiem WHERE są logiczne operatory porównania:

- $a = b$
- $a > b$
- $a \geq b$
- $a < b$
- $a \leq b$
- $a <> b$ lub $a \neq b$

Wszystkie te wyrażenia zwracają boolean value (True/False)

```
SELECT COUNT(district)
FROM address
WHERE district = 'California';
```


1. Wyświetl wszystkie filmy, które mają tytuł równy „Avatar”
2. Wyświetl filmy, które mają więcej niż 120 minut długości (length)
3. Wyświetl filmy, które mają długość mniejszą niż 90 minut
4. Wyświetl filmy, które mają last_update później ,2006-02-15’
5. Wyświetl filmy, które mają rental_duration mniejszy lub równy 3
6. Wyświetl filmy, które mają rental_duration różny od 6



The screenshot shows a database management tool interface. At the top, a breadcrumb trail indicates the path: Server: 127.0.0.1 » Database: sakila » Table: film. Below this is a toolbar with icons for Browse, Structure, SQL, Search, Insert, and Export. A button labeled 'Show query box' is positioned below the toolbar. The main area displays a green status bar indicating 'Showing rows 0 - 24 (788 total, Query took 0.0005 seconds.)'. Below this, the SQL query is shown: `SELECT * FROM film WHERE rental_duration <> 6;`. At the bottom, there is a checkbox for 'Profiling' and several links: [Edit inline], [Edit], [Explain SQL], [Create PHP code], and [Refresh].

Server: 127.0.0.1 » Database: sakila » Table: film

Browse Structure SQL Search Insert Export

Show query box

✓ Showing rows 0 - 24 (788 total, Query took 0.0005 seconds.)

```
SELECT * FROM film WHERE rental_duration <> 6;
```

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Kombinowanie warunków

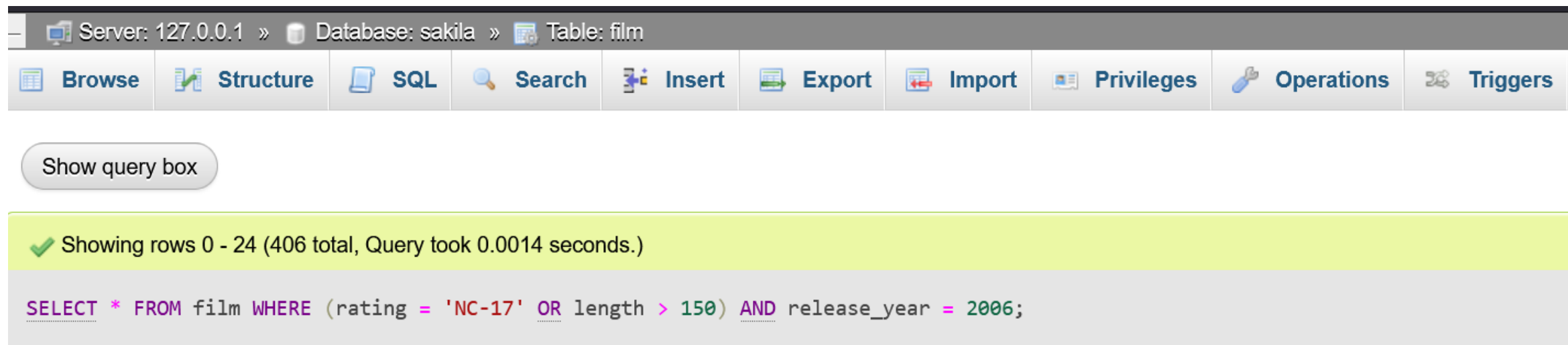
dimon.work

Logiczne „**i**” – **oba** warunki muszą być True

Logiczne „**albo**” – przynajmni **jeden** warunek musi być True

```
SELECT *  
FROM tabel_name  
WHERE condition1 AND condition2;
```

```
SELECT *  
FROM tabel_name  
WHERE condition1 OR condition2;
```



The screenshot shows a database management interface with a top navigation bar containing icons and labels for 'Server: 127.0.0.1', 'Database: sakila', and 'Table: film'. Below this is a row of buttons: 'Browse', 'Structure', 'SQL', 'Search', 'Insert', 'Export', 'Import', 'Privileges', 'Operations', and 'Triggers'. A 'Show query box' button is located below the navigation bar. The main area displays a green status bar indicating 'Showing rows 0 - 24 (406 total, Query took 0.0014 seconds.)'. Below the status bar, the executed SQL query is shown: `SELECT * FROM film WHERE (rating = 'NC-17' OR length > 150) AND release_year = 2006;`

1. Wyświetl filmy, które mają rating równy PG lub PG-13
2. Wyświetl filmy, które mają długość większą niż 100 minut i rating R:
3. Wyświetl filmy, które zostały wydane w 2006 roku i mają długość mniejszą niż 90 minut
4. Wyświetl filmy, które mają rating G lub długość większą niż 130 minut
5. Wyświetl filmy, które mają rating różny od NC-17 i zostały wydane w 2006 roku
6. Wyświetl filmy, które mają długość mniejszą niż 100 minut lub rating R oraz zostały wydane po 2006 roku
7. Wyświetl filmy, które mają rating NC-17 lub długość większą niż 150 minut oraz zostały wydane w 2006 roku

Operator przedziałów – pomiędzy

```
SELECT *  
FROM film  
WHERE length BETWEEN 91 AND 119;
```

-- to samo, ale z użyciem AND

```
SELECT *  
FROM film  
WHERE length >= 90 AND length <= 120;
```

```
SELECT *  
FROM film  
WHERE rating IN ('PG', 'R');
```

-- to samo, ale z użyciem OR

```
SELECT *  
FROM film  
WHERE rating = 'PG' OR rating = 'R'
```

Operator **ORDER BY** w SQL służy do sortowania wyników zapytania. Można sortować według jednej lub więcej kolumn i określić rosnąco lub malejąco - **ASC**ending / **DES**cending.

1. Sortowanie według jednej kolumny (kolejność rosnąca): Aby posortować filmy według roku premiery (kolejność rosnąca):

```
SELECT * FROM film  
ORDER BY release_year;
```

2. Sortowanie według wielu kolumn z różnymi kolejnościami

```
SELECT * FROM film  
ORDER BY release_year ASC, length DESC;
```

1. MIN - uzyskanie minimalnej wartości.

Znaleźć minimalną długość filmów:

```
SELECT MIN(length) FROM film;
```

MIN(length)

46

2. MAX - uzyskanie maksymalnej wartości.

Znaleźć maksymalną długość filmów:

```
SELECT MAX(length) FROM film;
```

MAX(length)

185

3. AVG - uzyskanie średniej.

Obliczyć średnią długość filmów:

```
SELECT AVG(length) FROM film;
```

AVG(length)

115.2720

Można również użyć tych funkcji z operatorem **GROUP BY**, aby uzyskać agregację według grupy. Na przykład, jeśli chcesz uzyskać średnią długość filmów dla każdej oceny:

```
SELECT rating,  
AVG(length) FROM film  
GROUP BY rating;
```

| rating | AVG(length) |
|--------|-------------|
| G | 111.0506 |
| PG | 112.0052 |
| PG-13 | 120.4439 |
| R | 118.6615 |
| NC-17 | 113.2286 |

EX (SELECT)

Znajdź wszystkie filmy z rankingiem **R** ,
które są krótsze niż średnia długość filmu,
wypisz tylko **title**, **special_features** i **length**
w rezultacie

| title | special_features | length |
|----------------------|---|--------|
| AIRPORT POLLOCK | Trailers | 54 |
| ALONE TRIP | Trailers,Behind the Scenes | 82 |
| AMELIE HELLFIGHTERS | Commentaries,Deleted Scenes,Behind the Scenes | 79 |
| ANACONDA CONFESSIONS | Trailers,Deleted Scenes | 92 |
| ANYTHING SAVANNAH | Trailers,Deleted Scenes,Behind the Scenes | 82 |
| BANGER PINOCCHIO | Trailers,Commentaries,Deleted Scenes | 113 |
| BEAST HUNCHBACK | Deleted Scenes,Behind the Scenes | 89 |

Pattern Matching – sprawdzenie odpowiedności określonego szablonowi (w naszym przypadku str)

LIKE – wykorzystuje się po to aby szukać *string`i* odpowiadające określonym szablonowi

Placeholder:

% (*percent*) – oznacza 0, 1 lub więcej znaków

_ (*underscore*) – dokładnie 1 znak

Placeholder:

% (*percent*) – oznacza 0, 1 lub więcej znaków

_ (*underscore*) – dokładnie 1 znak

- LIKE 'U%' - ciągi rozpoczynające się od U
- LIKE '%a' - ciągi znaków kończące się na a
- LIKE '%John%' - ciągi zawierające John
- LIKE 'J%n' - ciągi rozpoczynające się od J i kończące się na n
- LIKE '_oh_' - ciągi, w których 2 i 3 znak to oh, a pierwszy (1) i ostatni (4) znak są dowolne.
- LIKE '_oh%' - ciągi, w których 2, 3 znak to oh, pierwszy - dowolny i na końcu 0, 1 lub więcej dowolnych znaków

1. Znajdź wszystkich aktorów, których imiona zaczynają się od litery "P":

```
SELECT first_name, last_name  
FROM actor  
WHERE first_name LIKE 'P%';
```

2. Znajdź wszystkich aktorów, których nazwiska kończą się na "n,,
3. Znajdź wszystkich aktorów, których imiona zawierają "EN,,
4. Znajdź wszystkich aktorów, których imiona zaczynają się na "J" i kończą na "n,,
5. Znajdź wszystkich aktorów, których imiona mają cztery znaki, a drugim i trzecim znakiem jest "OH,,
6. Znajdź wszystkich aktorów, których imiona mają dowolny pierwszy znak, drugi i trzeci to "OH", a po nich może następować dowolna liczba znaków:

W SQL **LIMIT** służy do ograniczenia liczby zwracanych wierszy w wyniku zapytania. Jest szczególnie przydatny, gdy chcesz zwrócić tylko określoną liczbę rekordów, np. podczas przeglądania próbek danych.

```
SELECT title, rental_rate  
FROM film  
LIMIT 10;
```

IS NULL, IS NOT NULL

W SQL funkcje **IS NULL** oraz **IS NOT NULL** służą do sprawdzania, czy dana kolumna ma wartość NULL lub czy jest ona różna od NULL.

```
SELECT address_id, address, address2  
FROM address  
WHERE address2 IS NOT NULL;
```

\\.

Funkcja **GROUP BY** w SQL służy do grupowania wyników zapytania na podstawie wartości w jednej lub kilku kolumnach. Zazwyczaj jest używana w połączeniu z funkcjami agregującymi, takimi jak **COUNT()**, **SUM()**, **AVG()**, **MIN()**, **MAX()**.

1. Policz, ile różnych numerów telefonów znajduje się w każdej dzielnicy (district):

```
SELECT district, COUNT(DISTINCT phone) FROM address  
GROUP BY district;
```

1. Znajdź maksymalną i minimalną długość numeru telefonu w każdej dzielnicy (district):
2. Średnia liczba znaków w adresie (address) w każdej dzielnicy:

HAVING służy do filtrowania wyników na podstawie wartości uzyskanych przez funkcje agregujące.

1. Znajdź dzielnice (district), które mają więcej niż 1 adres:

```
SELECT district, COUNT(address_id) FROM address  
GROUP BY district HAVING COUNT(address_id) > 1;
```

2. Średnia długość adresu w każdej dzielnicy, ale pokaż tylko dzielnice, gdzie średnia długość adresu jest większa niż 10 znaków:

```
SELECT district, AVG(LENGTH(address)) FROM address  
GROUP BY district HAVING AVG(LENGTH(address)) > 10;
```

Operacje **UNION** w SQL są to operatory setowe (zbiorów), które pozwalają na wykonywanie operacji na wynikach dwóch lub więcej zapytań. Używając tych operatorów, możemy manipulować zestawami danych na podstawie podobieństw lub różnic między nimi.

Operator **UNION** łączy wyniki dwóch lub więcej zapytań, usuwając duplikaty. Łączy wiersze z obu zapytań w jedną tabelę wyników.

```
SELECT title FROM films WHERE rating = 'PG'  
UNION  
SELECT title FROM films WHERE release_year = 2006;
```

- Którzy aktorzy mają na imię „Scarlett”?
- Którzy aktorzy mają na nazwisko „Johansson”?
- Ile jest różnych nazwisk aktorów?
- Które nazwiska się nie powtarzają?
- Które nazwiska pojawiają się więcej niż raz?
- Który aktor wystąpił w największej liczbie filmów?
- Czy film „Academy Dinosaur” jest dostępny do wypożyczenia w Sklepie 1?
- Wstaw rekord reprezentujący Mary Smith wypożyczającą dziś film „Academy Dinosaur” od Mike'a Hillyera w Store 1.
- Kiedy będzie można obejrzeć film „Academy Dinosaur”?
- Jaki jest średni czas trwania wszystkich filmów w sakila DB?
- Jaki jest średni czas trwania filmów według kategorii?
- Dlaczego to zapytanie zwraca pusty zbiór?