

Materiały do przedmiotu:

Bazy danych

Laboratorium nr 4:

Wybrane polecenia języka SQL: Filtry, unie, funkcje agregujące, łączenie tabel, operacje na danych

Autor:

Alicja Żmudzińska



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Spis treści

Wprowadzenie, tematyka zajęć	3
Cel ćwiczenia/laboratorium	3
Instrukcja laboratoryjna/ćwiczeniowa	3
Ćwiczenia samodzielne	14
Pytania pomocnicze	16
Podsumowanie	16
Literatura	16



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Wprowadzenie, tematyka zajęć

SQL to znacznie więcej niż `SELECT * FROM ...`; — to język, który pozwala wydobyć z bazy dokładnie te informacje, które są potrzebne, połączyć je z różnych tabel i przygotować tak, by nadawały się do raportu lub analizy. Na tych zajęciach przejdziemy od prostych filtrów (klauszula `WHERE`, operatory porównania i logiczne) przez zaawansowane techniki filtrowania (`LIKE`, `IN`, `BETWEEN`, `IS NULL`, `EXISTS`, `REGEXP`) aż po łączenia tabel (`JOIN`), operacje na datach i czasie oraz agregacje z `GROUP BY` i `HAVING`. Będziemy korzystać z bazy Sakila, aby ćwiczyć rzeczywiste scenariusze: wyszukiwanie rekordów, łączenie relacji, obsługę braków danych oraz tworzenie podsumowań i rankingów.

Cel ćwiczenia/laboratorium

Celem zajęć jest nauczenie pisania czytelnego, poprawnego i efektywnego zapytania, takiego, które wybiera tylko potrzebne kolumny, filtrowanych przy użyciu właściwych warunków, łączących właściwe tabele i zwracających wynik gotowy do dalszej analizy.

Instrukcja laboratoryjna/ćwiczeniowa

Pisząc zapytania SQL, zwykle nie ograniczamy się tylko do prostego `SELECT * FROM ...`. Najczęściej chodzi o coś więcej: o wyciągnięcie tylko potrzebnych danych, połączenie ich z różnych tabel i odpowiednie przekształcenie, żeby dało się z tego zrobić raport albo analizę. Do tego właśnie służą różne konstrukcje języka SQL.

Filtry pozwalają ograniczyć liczbę zwracanych wierszy poprzez stosowanie warunków. Najczęściej używa się do tego klauzuli `WHERE`, w której można korzystać zarówno z operatorów porównania (`=`, `<`, `>`, `<>`), jak i logicznych (`AND`, `OR`, `NOT`). Dzięki nim wybieramy tylko te rekordy, które naprawdę nas interesują.

Przykład 1: Wyświetl wszystkich aktorów o nazwisku „Davis”

```
SELECT *  
FROM sakila.actor  
WHERE last_name = 'Davis';
```

Wyjaśnienie krok po kroku:

1. `SELECT first_name AS imie` – pobiera kolumnę `first_name` i nadaje jej alias `imie`.
2. `last_name AS nazwisko` – pobiera `last_name` i zmienia nagłówek na `nazwisko`.
3. `FROM sakila.actor` – wskazuje tabelę `actor` w bazie `sakila`.
4. `WHERE last_name = Davis` – filtruje wiersze do tych z nazwiskiem `Davis`.

Co zwróci polecenie: listę aktorów o nazwisku `Davis` (kolumny: `imie`, `nazwisko`).



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



✓ Pokazano wiersze 0 - 2 (3 total, Wykonanie zapytania trwało 0,0008 sekund(y).)

```
SELECT * FROM sakila.actor WHERE last_name = 'Davis';
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Liczba wierszy: 25 | Filter rows: Sortuj w

Extra options

	actor_id	first_name	last_name	last_update
☐ Edit Copy Delete	4	JENNIFER	DAVIS	2006-02-15 04:34:33
☐ Edit Copy Delete	101	SUSAN	DAVIS	2006-02-15 04:34:33
☐ Edit Copy Delete	110	SUSAN	DAVIS	2006-02-15 04:34:33

Dodatkowe operatory filtrowania:

LIKE / NOT LIKE – dopasowanie wzorca:

% – dopasowuje **dowolny ciąg znaków (także pusty)**

- 'A%' → wszystko, co zaczyna się na „A” (np. AL, ANNA, A)
- '%son' → wszystko, co kończy się na „son” (np. Jackson, Harrison)
- '%ann%' → wszystko, co **zawiera** „ann” gdziekolwiek).

_ (podkreślnik) – dopasowuje dokładnie jeden dowolny znak

- '_a%' → drugi znak to „a” (np. Jack, Max)
- 'J_n' → Jan, Jon, Jin itd.

Przykład: Wyświetl aktorów z nazwiskiem zaczynającym się na „A”

```
SELECT first_name, last_name
FROM sakila.actor
WHERE last_name LIKE 'A%';
```

Wyjaśnienie krok po kroku:

1. SELECT first_name, last_name – wybiera imię i nazwisko aktora.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



2. FROM sakila.actor – z tabeli aktorów.
3. WHERE last_name LIKE 'A%' – filtr: nazwisko zaczyna się od „A”.

Co zwróci polecenie: listę aktorów, których nazwisko zaczyna się od litery „A”.

✓ Pokazano wiersze 0 - 6 (7 total, Wykonanie zapytania trwało 0,0008 sekund(y).)

```
SELECT first_name, last_name FROM sakila.actor WHERE last_name LIKE 'A%';
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Liczba wierszy: 25 | Filter rows:

Extra options

	first_name	last_name
☐ Edit Copy Delete	CHRISTIAN	AKROYD
☐ Edit Copy Delete	KIRSTEN	AKROYD
☐ Edit Copy Delete	DEBBIE	AKROYD
☐ Edit Copy Delete	CUBA	ALLEN
☐ Edit Copy Delete	KIM	ALLEN
☐ Edit Copy Delete	MERYL	ALLEN
☐ Edit Copy Delete	ANGELINA	ASTAIRE

IN/NOT IN – filtr do listy wartości lub podzapytania.

Przykład: klienci z wybranych miast.

```
SELECT c.first_name, c.last_name
FROM sakila.customer c
JOIN sakila.address a USING(address_id)
JOIN sakila.city ci USING(city_id)
WHERE ci.city IN ('London', 'Paris', 'Rome');
```

Wyjaśnienie krok po kroku:

1. JOIN łączy klientów z ich adresem i miastem.
2. WHERE ci.city IN (...) – przepuszcza tylko wiersze z listy miast.
3. SELECT zwraca imię i nazwisko klientów spełniających warunek.

Co zwróci polecenie: klientów mieszkających w Londynie, Paryżu lub Rzymie



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



✓ Pokazano wiersze 0 - 1 (2 total, Wykonanie zapytania trwało 0,0048 sekund(y).)

```
SELECT c.first_name, c.last_name FROM sakila.customer c JOIN sakila.address a USING(address_id) JOIN sakila.city ci USING(city_id) WHERE ci.city IN ('London', 'Paris', 'Rome');
```

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

☐ Show all | Liczba wierszy: 25 | Filter rows: Przeszukaj tę tabelę | Sortuj wg klucza: Żaden

Extra options

first_name	last_name
MATTIE	HOFFMAN
CECIL	VINES

BETWEEN / NOT BETWEEN – zakres (włącznie z granicami).

Przykład: filmy o stawce wypożyczenia w przedziale 0.99–2.99.

```
SELECT title, rental_rate
FROM sakila.film
WHERE rental_rate BETWEEN 0.99 AND 2.99;
```

Wyjaśnienie krok po kroku:

1. SELECT title, rental_rate – pobiera tytuł i stawkę.
2. WHERE rental_rate BETWEEN 0.99 AND 2.99 – wybiera wiersze w podanym zakresie (łącznie z 0.99 i 2.99).

Co zwróci polecenie: filmy, których rental_rate mieści się w tym przedziale.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



✓ Pokazano wiersze 0 - 24 (664 total, Wykonanie zapytania trwało 0,0004 sekund(y).)

```
SELECT title, rental_rate FROM sakila.film WHERE rental_rate BETWEEN 0.99 AND 2.99;
```

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Liczba wierszy: 25 | Filter rows: Przeszukaj tę tabelę S

Extra options

	title	rental_rate
☐ Edit Copy Delete	ACADEMY DINOSAUR	0.99
☐ Edit Copy Delete	ADAPTATION HOLES	2.99
☐ Edit Copy Delete	AFFAIR PREJUDICE	2.99
☐ Edit Copy Delete	AFRICAN EGG	2.99
☐ Edit Copy Delete	AGENT TRUMAN	2.99
☐ Edit Copy Delete	ALABAMA DEVIL	2.99
☐ Edit Copy Delete	ALAMO VIDEOTAPE	0.99
☐ Edit Copy Delete	ALASKA PHANTOM	0.99
☐ Edit Copy Delete	ALICE FANTASIA	0.99
☐ Edit Copy Delete	ALIEN CENTER	2.99

IS NULL / IS NOT NULL – sprawdzanie braków danych.

Przykład: wypożyczenia bez daty zwrotu.

```
SELECT rental_id, customer_id
FROM sakila.rental
WHERE return_date IS NULL;
```

Wyjaśnienie krok po kroku:

1. SELECT rental_id, customer_id – wybiera identyfikator wypożyczenia i klienta.
2. FROM sakila.rental – z tabeli wypożyczeń.
3. WHERE return_date IS NULL – filtr: brak daty zwrotu.

Co zwróci polecenie: wypożyczenia, które nie zostały jeszcze zwrócone.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



✓ Pokazano wiersze 0 - 24 (183 total, Wykonanie zapytania trwało 0,0090 sekund(y).)

```
SELECT rental_id, customer_id FROM sakila.rental WHERE return_date IS NULL;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

1 > >> ☐ Show all | Liczba wierszy: 25 Filter rows: Prze:

Extra options

	rental_id	customer_id
☐ Edit Copy Delete	11496	155
☐ Edit Copy Delete	11541	335
☐ Edit Copy Delete	11563	83
☐ Edit Copy Delete	11577	219
☐ Edit Copy Delete	11593	99
☐ Edit Copy Delete	11611	192
☐ Edit Copy Delete	11646	11
☐ Edit Copy Delete	11652	597
☐ Edit Copy Delete	11657	53
☐ Edit Copy Delete	11672	521

EXISTS / NOT EXISTS – sprawdzenie istnienia powiązanych wierszy (wydajne przy podzapytaniach).

Przykład: klienci, którzy mają jakiegokolwiek płatności.

```
SELECT c.customer_id, c.first_name, c.last_name
FROM sakila.customer c
WHERE EXISTS (
  SELECT 1 FROM sakila.payment p
  WHERE p.customer_id = c.customer_id);
```

Wyjaśnienie krok po kroku:

1. Zewnętrzne SELECT – kandydaci: wszyscy klienci.
2. EXISTS (podzapytanie) – sprawdza, czy istnieje **co najmniej jeden** wiersz w payment dla danego klienta.
3. Jeśli tak – klient przechodzi filtr; jeśli nie – jest odrzucony.

Co zwróci polecenie: klientów, którzy mają przynajmniej jedną płatność.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



✓ Pokazano wiersze 0 - 24 (599 total, Wykonanie zapytania trwało 0,0065 sekund(y).)

```
SELECT c.customer_id, c.first_name, c.last_name FROM sakila.customer c WHERE EXISTS ( SELECT 1 FROM sakila.payment p WHERE p.customer_id = c.customer_id );
```

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Liczba wierszy: 25 | Filter rows: Przeszukaj tę tabelę | Sortuj wg klucza: Żaden

Extra options

	customer_id	first_name	last_name
☐ Edit Copy Delete	1	MARY	SMITH
☐ Edit Copy Delete	2	PATRICIA	JOHNSON
☐ Edit Copy Delete	3	LINDA	WILLIAMS
☐ Edit Copy Delete	4	BARBARA	JONES
☐ Edit Copy Delete	5	ELIZABETH	BROWN
☐ Edit Copy Delete	6	JENNIFER	DAVIS
☐ Edit Copy Delete	7	MARIA	MILLER
☐ Edit Copy Delete	8	SUSAN	WILSON
☐ Edit Copy Delete	9	MARGARET	MOORE

REGEXP – dopasowanie wyrażeń regularnych (precyzyjniejsze niż LIKE).

Przykład: nazwiska zaczynające się od „Mc” lub „Mac”.

```
SELECT first_name, last_name
FROM sakila.actor
WHERE last_name REGEXP '^(Mc|Mac)';
```

Wyjaśnienie krok po kroku:

1. FROM sakila.actor – wskazuje tabelę z danymi aktorów, z której będziemy pobierać wiersze.
2. WHERE last_name REGEXP '^(Mc|Mac)' – filtruje wiersze za pomocą wyrażenia regularnego:
 - REGEXP – operator dopasowania wzorca (regex) w MySQL.
 - ^ – kotwica początku tekstu (wzorec musi pasować od pierwszego znaku nazwiska).
 - (Mc|Mac) – alternatywa: pasuje „Mc” lub „Mac”.

Co zwróci polecenie: Listę aktorów, których nazwisko zaczyna się na „Mc” lub „Mac”



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



✓ Pokazano wiersze 0 - 6 (7 total, Wykonanie zapytania trwało 0,0020 sekund(y).)

```
SELECT first_name, last_name FROM sakila.actor WHERE last_name REGEXP '^(Mc|Mac)';
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Liczba wierszy: 25 | Filter rows: Sortuj w

Extra options

	first_name	last_name
☐ Edit Copy Delete	JULIA	MCQUEEN
☐ Edit Copy Delete	TOM	MCKELLEN
☐ Edit Copy Delete	MICHELLE	MCCONAUGHEY
☐ Edit Copy Delete	CARY	MCCONAUGHEY
☐ Edit Copy Delete	MORGAN	MCDORMAND
☐ Edit Copy Delete	CATE	MCQUEEN
☐ Edit Copy Delete	GENE	MCKELLEN

W relacyjnych bazach danych informacje są podzielone na różne, powiązane ze sobą tabele. Aby uzyskać pełny obraz, musimy je połączyć, a służy do tego polecenie **JOIN**. Podstawą takich połączeń są klucze: **PRIMARY KEY**, czyli unikalny identyfikator w tabeli, oraz **FOREIGN KEY**, wskazujący na rekord w innej tabeli. W bazie Sakila przykładem jest tabela inventory, w której kolumna film_id jest kluczem obcym odnoszącym się do tabeli film. Najczęściej spotykane rodzaje złączeń to: **INNER JOIN**, zwracający tylko pasujące rekordy z obu tabel; **LEFT JOIN**, który zwraca wszystkie rekordy z tabeli po lewej oraz pasujące z prawej, a brakujące uzupełnia wartością NULL; **RIGHT JOIN**, czyli odwrotność **LEFT JOIN**; **FULL OUTER JOIN**, obejmujący wszystkie rekordy z obu tabel (w MySQL trzeba go symulować za pomocą **UNION**); **CROSS JOIN**, który tworzy iloczyn kartezjański obu tabel; oraz **SELF JOIN**, w którym tabela łączy się sama ze sobą, np. w celu porównania rekordów.

Przykład

```
SELECT f.title AS tytuł, a.last_name AS nazwisko
FROM sakila.film AS f
INNER JOIN sakila.film_actor AS fa ON f.film_id = fa.film_id
INNER JOIN sakila.actor AS a ON fa.actor_id = a.actor_id;
```

Wyjaśnienie krok po kroku:

1. sakila.film AS f – tabela filmów (alias f).
2. sakila.film_actor AS fa – tabela wiążąca filmy z aktorami.
3. sakila.actor AS a – dane aktorów.
4. INNER JOIN ... ON f.film_id = fa.film_id – związanie filmów z relacjami.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



5. INNER JOIN ... ON fa.actor_id = a.actor_id – dociągnięcie aktorów.

6. SELECT f.title AS tytuł, a.last_name AS nazwisko – wybór i aliasy.

Co zwróci polecenie: listę par „tytuł filmu – nazwisko aktora”.

✓ Pokazano wiersze 0 - 24 (5462 total, Wykonanie zapytania trwało 0,0014 sekund(y).)

```
SELECT f.title AS tytuł, a.last_name AS nazwisko FROM sakila.film AS f INNER JOIN sakila.film_actor AS fa ON f.film_id = fa.film_id INNER JOIN sakila.actor AS a ON fa.actor_id = a.actor_id;
```

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> Liczba wierszy: 25 Filter rows: Przeszukaj tę tabelę Sortuj wg klucza: Żaden

Extra options

tytuł	nazwisko
BACKLASH UNDEFEATED	AKROYD
BETRAYED REAR	AKROYD
CAPER MOTIONS	AKROYD
CATCH AMISTAD	AKROYD
CHANCE RESURRECTION	AKROYD
CONFUSED CANDLES	AKROYD
CUPBOARD SINNERS	AKROYD
DIVIDE MONSTER	AKROYD
DOOM DANCING	AKROYD

Na koniec warto wspomnieć o pracy z datami i czasem, bo to bardzo częsty element analiz. W bazie Sakila występują na przykład kolumny rental_date, return_date czy payment_date. SQL obsługuje różne typy czasowe: DATE, zawierający wyłącznie datę (rok, miesiąc, dzień), DATETIME i TIMESTAMP, które przechowują zarówno datę, jak i godzinę, oraz TIME, przechowujący sam czas. Przykładowo w tabeli rental kolumna rental_date zapisuje dokładny moment wypożyczenia filmu. Do pracy z datami przydają się również funkcje systemowe, takie jak NOW(), zwracająca bieżącą datę i godzinę, CURRENT_DATE, która podaje dzisiejszą datę, czy CURRENT_TIMESTAMP, będąca odpowiednikiem funkcji NOW().

Przykład

```
SELECT rental_id AS id, rental_date AS data_wypozyczenia
FROM sakila.rental
WHERE rental_date <= CURRENT_DATE;
```

Wyjaśnienie krok po kroku:

1. sakila.rental – historia wypożyczeń.
2. rental_id AS id – identyfikator z aliasem id.
3. rental_date AS data_wypozyczenia – data/godzina wypożyczenia.
4. WHERE rental_date <= CURRENT_DATE – zwraca rekordy do początku dzisiejszego dnia (00:00:00) włącznie.

Co zwróci polecenie:



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



✓ Pokazano wiersze 0 - 24 (16044 total, Wykonanie zapytania trwało 0,0003 sekund(y).)

```
SELECT rental_id AS id, rental_date AS data_wypozyczenia FROM sakila.rental WHERE rental_date <= CURRENT_DATE;
```

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Liczba wierszy: 25 | Filter rows: Przeszukaj tę tabelę | Sortuj wg klucza: Żaden

Extra options

		id	data_wypozyczenia
☐	Edit	1	2005-05-24 22:53:30
☐	Edit	2	2005-05-24 22:54:33
☐	Edit	3	2005-05-24 23:03:39
☐	Edit	4	2005-05-24 23:04:41
☐	Edit	5	2005-05-24 23:05:21
☐	Edit	6	2005-05-24 23:08:07
☐	Edit	7	2005-05-24 23:11:53
☐	Edit	8	2005-05-24 23:31:46
☐	Edit	9	2005-05-25 00:00:40
☐	Edit	10	2005-05-25 00:02:21

Kiedy dane pochodzą z kilku zapytań i chcemy je połączyć w jedno zestawienie, korzystamy z unii. Operator UNION łączy dane i usuwa duplikaty, dzięki czemu każda kombinacja wartości pojawia się tylko raz. Z kolei UNION ALL działa szybciej, ponieważ łączy wszystkie wyniki, pozostawiając duplikaty i pozwalając zobaczyć faktyczną liczbę danych. Trzeba jednak pamiętać, że każde z łączonych zapytań musi zwracać tę samą liczbę kolumn o zgodnych typach danych.

Przykład 2: Wyświetl listę imion aktorów oraz imion klientów

```
SELECT first_name FROM sakila.actor
UNION
SELECT first_name FROM sakila.customer;
```

Wyjaśnienie krok po kroku:

1. Pierwsze zapytanie wybiera first_name z tabeli actor i podaje je jako imię.
2. Drugie zapytanie wybiera first_name z customer, również jako imię.



Fundusze Europejskie
dla Rozwoju Społecznego



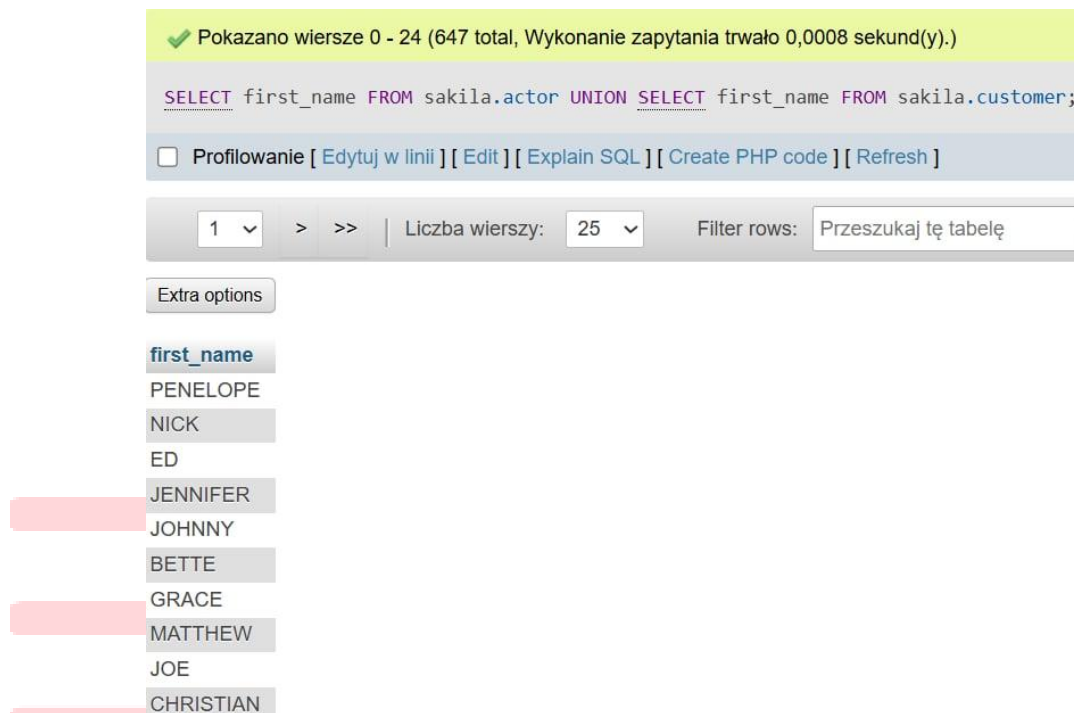
Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



3. UNION – łączy oba wyniki i usuwa duplikaty (wymaga takiej samej liczby i typów kolumn po obu stronach).

Co zwróci polecenie: unikalną listę imion z obu tabel.



✓ Pokazano wiersze 0 - 24 (647 total, Wykonanie zapytania trwało 0,0008 sekund(y).)

```
SELECT first_name FROM sakila.actor UNION SELECT first_name FROM sakila.customer;
```

☐ Profilowanie [Edytuj w linii] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Liczba wierszy: 25 | Filter rows: Przeszukaj tę tabelę

Extra options

first_name

PENELOPE

NICK

ED

JENNIFER

JOHNNY

BETTE

GRACE

MATTHEW

JOE

CHRISTIAN

SQL umożliwia także wykonywanie obliczeń na całych zbiorach rekordów, a nie tylko na pojedynczych wierszach. Służą do tego funkcje agregujące: COUNT() do zliczania wierszy, SUM() do sumowania wartości, AVG() do obliczania średniej, a także MIN() i MAX() do znajdowania najmniejszej lub największej wartości. Żeby miały sens, często stosuje się je razem z klauzulą GROUP BY, która grupuje dane według wybranej kolumny. Jeśli dodatkowo chcemy odfiltrować tylko niektóre grupy, używamy klauzuli HAVING, która działa już po agregacji, w przeciwieństwie do WHERE.

```
SELECT city_id, COUNT(*) AS ile
FROM sakila.address
GROUP BY city_id
ORDER BY ile DESC
LIMIT 10;
```

Wyjaśnienie krok po kroku:

1. city_id – kolumna zwracana i kolumna grupująca (identyfikator miasta).
2. COUNT(*) AS ile – zlicza wiersze (adresy) w każdej grupie miasta; alias wyniku to ile.
3. FROM sakila.address – źródło danych: tabela adresów.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



4. GROUP BY city_id – grupuje rekordy według miasta, aby COUNT(*) liczył per miasto.
5. ORDER BY ile DESC – sortuje miasta malejąco według liczby adresów.
6. LIMIT 10 – zwraca Top 10 miast o największej liczbie adresów.

Co zwróci polecenie: dziesięć city_id z największą liczbą wierszy w sakila.address wraz z tą liczbą

✓ Pokazano wiersze 0 - 9 (10 total, Wykonanie zapytania trwało 0,0004 sekund(y).)

```
SELECT city_id, COUNT(*) AS ile FROM sakila.address GROUP BY city_id ORDER BY ile DESC LIMIT 10;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

Extra options

city_id	ile
576	2
312	2
42	2
300	2
13	1
29	1
45	1
61	1
77	1
93	1

Ćwiczenia samodzielne

1. Wyświetl wszystkich aktorów o nazwisku „SMITH”.
2. Wyświetl klientów z miasta „London”.
3. Wyświetl tytuły filmów dłuższych niż 120 minut.
4. Wyświetl filmy, których koszt wypożyczenia mieści się w przedziale od 0.99 do 2.99.
5. Wyświetl aktorów, których nazwisko zaczyna się na literę „A”.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



6. Wyświetl płatności większe niż 10.
7. Wyświetl wszystkie wypożyczenia dokonane przez klienta o nazwisku „Brown”.
8. Wyświetl listę imion aktorów i klientów (bez duplikatów).
9. Wyświetl listę imion aktorów i klientów (z duplikatami).
10. Wyświetl wspólną listę nazwisk aktorów i pracowników.
11. Wyświetl liczbę aktorów dla każdej litery alfabetu (pierwsza litera nazwiska).
12. Wyświetl średnią długość filmów w kategoriach „Action” i „Comedy”.
13. Wyświetl liczbę filmów w każdej kategorii.
14. Wyświetl 10 miast z największą liczbą klientów.
15. Wyświetl klientów, którzy wypożyczyli więcej niż 20 filmów.
16. Wyświetl sumę płatności dla każdego klienta.
17. Wyświetl minimalną i maksymalną opłatę za wypożyczenie filmu.
18. Wyświetl tytuły filmów wraz z językiem, w którym zostały nagrane.
19. Wyświetl listę klientów wraz z nazwą miasta, w którym mieszkają.
20. Wyświetl wszystkich aktorów grających w filmie „ACADEMY DINOSAUR”.
21. Wyświetl tytuły filmów należących do kategorii „Action”.
22. Wyświetl wszystkich klientów i sumę ich płatności (uwzględnij także tych, którzy nie mają żadnych płatności).
23. Wyświetl pary aktorów mających to samo nazwisko.
24. Wyświetl listę klientów i ich płatności używając symulacji pełnego złączenia (FULL OUTER JOIN).
25. Wyświetl listę filmów z liczbą aktorów występujących w każdym z nich.
26. Wyświetl wszystkie wypożyczenia z ostatnich 7 dni.
27. Wyświetl łączną kwotę płatności dokonanych w bieżącym miesiącu.
28. Wyświetl średnią kwotę płatności w podziale na dni.
29. Wyświetl klientów, którzy nie wypożyczali filmów od 60 dni.
30. Wyświetl listę wypożyczeń wraz z czasem (w dniach) pomiędzy wypożyczeniem a zwrotem filmu.



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



Pytania pomocnicze

1. Po co w ogóle stosujemy klauzulę WHERE w zapytaniach SQL?
2. Dlaczego LIKE 'A%' i LIKE '%A' zwrócą różne wyniki?
3. Czym różni się INNER JOIN od LEFT JOIN?
4. Jaka jest różnica między UNION a UNION ALL?
5. Dlaczego oba zapytania łączone przez UNION muszą zwracać tę samą liczbę kolumn?
6. Kiedy stosujemy klauzulę GROUP BY?
7. Do czego służy znak %, a do czego _ w operatorze LIKE?

Podsumowanie

SQL to narzędzie, które pozwala nie tylko pobierać dane, ale też je filtrować, łączyć i analizować. Dzięki klauzuli WHERE możemy ograniczyć liczbę wyników do tych, które nas interesują. Operatory takie jak LIKE, IN, BETWEEN, IS NULL czy EXISTS pozwalają formułować bardziej precyzyjne warunki. Łączenie tabel (JOIN) umożliwia uzyskanie pełnego obrazu danych z relacyjnej bazy, a praca z datami i czasem jest kluczowa w analizach biznesowych. Z kolei UNION pozwala łączyć wyniki z wielu zapytań, a funkcje agregujące (COUNT, SUM, AVG, MIN, MAX) wraz z GROUP BY i HAVING dają możliwość tworzenia zestawień i raportów.

Literatura

Karwin, Bill, i in. Antywzorce języka SQL : jak uniknąć pułapek podczas programowania baz danych. Wyd. Helion, 2012.

<https://www.w3schools.com/sql/>



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską

