Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Informatyka algorytmiczna

Marek Świergoń (261750)

26 października 2021 r.

Bazy Danych i Systemy Informacyjne Laboratorium: sprawozdanie z listy 1

1. Wprowadzenie

1.1. Cel zadań

Polecenia z listy 1. sprawdzają i pozwalają na zapoznanie się z podstawami składni języka zapytań SQL (ang, *Structured Query Language*), w szczególności z zakresu formułowania zapytań (DQL – ang, *Data Query Language*). Część listy wprowadza podstawy modyfikacji zawartości baz danych (usuwanie kolumn, aktualizacja ich zawartości zależna od warunków).

1.2. Środowisko SQL wykorzystane w zadaniach

W rozwiązywaniu zadań wykorzystano darmowe zintegrowane środowisko programistyczne MySQL, w którego skład wchodzą:

- MySQL Workbench 8.0 – wizualne, programistyczne narzędzie do projektowania baz danych zawierające pole do wprowadzania zapytań w składni MySQL (wszystkie zadanie zostały rozwiązane z pomocą kwerend pisanych w języku SQL)
- MySQL Server – oprogramowanie służące za serwer bazodanowy, umożliwia połączenie programu MySQL Workbench z bazami danych

2. Opis rozwiązań poleceń

Uwaga – w rezultatach często są pokazane tylko fragmenty wypisanych rekordów ze względu na ich dużą liczbę.

2.1. Zadanie 1

2.1.1. Polecenie

Wypisz wszystkie znajdujące się w bazie tabele.

2.1.2. Kwerendy niezbędne do realizacji

```
1 •  SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_SCHEMA = 'sakila';
```

2.1.3. Rezultat

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE_ROWS	AVG_ROW_LENGTH	DATA_LENGTH	MAX_DATA_LENGTH	IND
def	sakila	actor	BASE TABLE	InnoDB	10	Dynamic	200	81	16384	0	163
def	sakila	address	BASE TABLE	InnoDB	10	Dynamic	603	163	98304	0	163
def	sakila	category	BASE TABLE	InnoDB	10	Dynamic	16	1024	16384	0	0
def	sakila	city	BASE TABLE	InnoDB	10	Dynamic	600	81	49152	0	163
def	sakila	country	BASE TABLE	InnoDB	10	Dynamic	109	150	16384	0	0
def	sakila	customer	BASE TABLE	InnoDB	10	Dynamic	599	136	81920	0	491
def	sakila	film	BASE TABLE	InnoDB	10	Dynamic	1000	180	180224	0	819
def	sakila	film_actor	BASE TABLE	InnoDB	10	Dynamic	5462	35	196608	0	819
def	sakila	film_category	BASE TABLE	InnoDB	10	Dynamic	1000	65	65536	0	163
def	sakila	film_text	BASE TABLE	InnoDB	10	Dynamic	1000	180	180224	0	163
def	sakila	inventory	BASE TABLE	InnoDB	10	Dynamic	4581	39	180224	0	196
def	sakila	language	BASE TABLE	InnoDB	10	Dynamic	6	2730	16384	0	0
def	sakila	payment	BASE TABLE	InnoDB	10	Dynamic	16086	98	1589248	0	638
def	sakila	rental	BASE TABLE	InnoDB	10	Dynamic	16008	99	1589248	0	119
def	sakila	staff	BASE TABLE	InnoDB	10	Dynamic	2	32768	65536	0	327
def	sakila	store	BASE TABLE	InnoDB	10	Dynamic	2	8192	16384	0	327

2.1.4. Opis rozwiązania

W kwerendzie wykorzystałem klasyczną składnię zapytań. Po SELECT wprowadzamy * aby uzyskać wszystkie kolumny (jeśli w poleceniu byłoby doprecyzowanie, że należy wypisać tylko nazwy, to wtedy zamiast gwiazdki pojawiłoby się TABLE_NAME jako nazwa kolumny z nazwami). Po słowie kluczowym FROM zaznaczamy, że chcemy pobrać rekordy z tabeli, która przechowuje dane dotyczące tabel z baz danych (INFORMATION_SCHEMA.TABLES). Ponieważ chcemy pobrać informacje dotyczące tylko tabel (bez np. widoków), po słowie kluczowym WHERE wprowadzam warunek: typ tabeli musi być BASE_TABLE. Pobieram informacje o tabelach tylko z bazy danych sakila, zatem konieczny jest jeszcze w MySQL warunek TABLE_SCHEMA = ‘sakila’.

2.2. Zadanie 2

2.2.1. Polecenie

Wypisz tytuły filmów o długości większej niż 120 minut.

2.2.2. Kwerendy niezbędne do realizacji

```
1 • SELECT title from sakila.film WHERE length > 120;
```

2.2.3. Rezultat

title
AFRICAN EGG
AGENT TRUMAN
ALAMO VIDEOTAPE
ALASKA PHANTOM
ALI FOREVER
ALLEY EVOLUTION
AMERICAN CIRCUS
ANALYZE HOOSIERS
ANONYMOUS HUMAN
ANTITRUST TOMATOES
APOLLO TEEN
ARACHNOPHOBIA ROL...
ARGONAUTS TOWN
ARIZONA BANG
ARMY FLINTSTONES
ARSENIC INDEPENDENCE
ARTIST COLDBLOODED
ATLANTIS CAUSE

film 1 ×

2.2.4. Opis rozwiązania

Chcemy wypisać tylko tytuły filmów, zatem po SELECT pojawia się kolumna title pobrana z (FROM) tabeli film bazy danych sakila. Aby wyświetlić filmy o długości większej niż 120 minut wprowadzam warunek do kolumny length (zawierającej długość filmu) po słowie kluczowym WHERE

2.3. Zadanie 3

2.3.1. Polecenie

Wypisz tytuły filmów oraz ich język, dla wszystkich filmów, w których opisie występuje słowo *Documentary*.

2.3.2. Kwerendy niezbędne do realizacji

```
1 •  SELECT film.title, language.name FROM sakila.film INNER JOIN sakila.language ON film.language_id = language.language_id
2    WHERE film.description LIKE '%Documentary%';
```

2.3.3. Rezultat

title	name
AFFAIR PREJUDICE	English
AFRICAN EGG	English
ARSENIC INDEPENDENCE	English
BALLROOM MOCKINGBIRD	English
BEVERLY OUTLAW	English
BIRD INDEPENDENCE	English
BLANKET BEVERLY	English
BLUES INSTINCT	English
CASPER DRAGONFLY	English
CELEBRITY HORN	English

Result 5

2.3.4. Opis rozwiązania

Aby poznać język filmu (nie tylko id języka) muszę połączyć tabele sakila.film i sakila.language frazą kluczową INNER JOIN. Złączenie następuje przez przyrównanie language_id z obu tabel (ON …) i, ponieważ użyty jest INNER JOIN, wyświetlane są tylko rekordy mające pasujące do siebie pola language_id. Słowo Documentary wykrywam, korzystając ze wzorca po słowie kluczowym LIKE. Określam, że opis filmu może zawierać dowolny ciąg znaków przed i po słowie Documentary, poprzez znak specjalny % (innym znakiem specjalnym jest _; określa on, iż przed danym zapisem może znaleźć się jeden dowolny znak).

2.4. Zadanie 4

2.4.1. Polecenie

Wypisz tytuły filmów z kategorii *Documentary*, które w swoim opisie nie zawierają słowa *Documentary*.

2.4.2. Kwerendy niezbędne do realizacji

```
1 •  SELECT film.title FROM sakila.film
2    INNER JOIN sakila.film_category ON film_category.film_id=film.film_id
3    INNER JOIN sakila.category ON film_category.category_id=category.category_id
4    WHERE category.name = 'Documentary' AND film.description NOT LIKE '%Documentary%';
```

2.4.3. Rezultat

title
ACADEMY DINOSAUR
ADAPTATION HOLES
ARMY FLINTSTONES
BEACH HEARTBREAKERS
BED HIGHBALL
BILL OTHERS
BONNIE HOLOCAUST
BROTHERHOOD BLANKET
CAUSE DATE
CHICKEN HELLFIGHTERS
CIDER DESIRE
CLERKS ANGELS
CUPBOARD SINNERS
DANCING FEVER
DEEP CRUSADE
DELIVERANCE MULHOLL...
DOZEN LION
DUFFEL APOCALYPSE

Result 2

2.4.4. Opis rozwiązania

W tym zadaniu wykorzystuję dwukrotne złączenie tabel INNER JOIN (najpierw sakila.film i sakila.film_category poprzez przyrównanie pól film_id, następnie powstałe połączenie łączę z tabelą category poprzez przyrównanie klucza category_id). Pierwsze połączenie jest potrzebne, żeby dowiedzieć się jakie category_id ma przypisany każdy film; drugie połączenie sprawdza, jaką kategorię przedstawia dane category_id. W warunkach określamy, że nazwa kategorii ma być *Documentary* oraz w opisie filmu nie może występować słowo *Documentary*. Robię to poprzez dopisanie słowa kluczowego NOT przed LIKE (które określa ten sam wzorzec co w zadaniu 3.). Dzięki temu, zamiast szukać wystąpień danego wzorca w rekordach, szukamy tych rekordów, które nie spełniają go.

2.5. Zadanie 5

2.5.1. Polecenie

Wypisz imiona i nazwiska wszystkich aktorów, którzy wystąpili w filmach zawierających usunięte sceny.

2.5.2. Kwerendy niezbędne do realizacji

```
1 •  SELECT imie, nazwisko
2    FROM (SELECT DISTINCT actor.actor_id, actor.first_name AS imie, actor.last_name AS nazwisko FROM sakila.actor
3    INNER JOIN sakila.film_actor ON film_actor.actor_id = actor.actor_id
4    INNER JOIN sakila.film ON film_actor.film_id = film.film_id
5    WHERE film.special_features LIKE '%Deleted%') sub ORDER BY nazwisko, imie;
```

2.5.3. Rezultat

imie	nazwisko
CHRISTIAN	AKROYD
DEBBIE	AKROYD
KIRSTEN	AKROYD
CUBA	ALLEN
KIM	ALLEN
MERYL	ALLEN
ANGELINA	ASTAIRE
RUSSELL	BACALL
AUDREY	BAILEY
JESSICA	BAILEY
HARRISON	BALE
RENEE	BALL
JULIA	BARRY...
VIVIEN	BASINGER
MICHAEL	BENING
SCARLETT	BENING
VIVIEN	BERGEN
LIZA	BERGMAN

Result 2 ×

2.5.4. Opis rozwiązania

Na początku (wewnątrz) tworzę tabelę pomocniczą o nazwie sub, w której chcę wyświelić aktorów występujących w filmach zawierających usunięte sceny, bez powtórzeń w przypadku, gdy aktor wystąpił w kilku takich filmach. W tym celu łączę ze sobą trzy tabele (INNER JOIN): sakila.actor z sakila.film_actor poprzez klucz aktor_id (aby dowiedzieć się jakie są id filmów, w których aktorzy występowali) oraz sakila.film_actor z sakila.film poprzez klucz film_id (aby dowiedzieć się jakie filmy są przypisane do danych film_id i móc sprawdzić czy zawierają usunięte sceny). Sprawdzam, czy dany film zawiera usunięte sceny korzystając ze wzorca (LIKE) '%Deleted%', ponieważ zauważam, że taki występuje, gdy film zawiera usunięte sceny ('Deleted Scenes').

Tabela pomocnicza jest po to, żeby wyświetlać danego aktora tylko raz, ale nie pomijać aktorów o tych samych imionach i nazwiskach. Stąd słowo kluczowe DISTINCT do tego służące znajduje się przed kolumną actor_id. Zabieg ten daje pozytywne skutki – w tabeli występuje dwa razy SUSAN DAVIS, ponieważ są w tabeli actor dwie różne osoby z takim imieniem i nazwiskiem.

W tabeli głównej biorę imiona i nazwiska z tabeli sub i sortuję je najpierw po nazwisku a potem po imieniu słowem kluczowym ORDER BY (dla czytelności wyników).

2.6. Zadanie 6

2.6.1. Polecenie

Dla każdej kategorii wiekowej filmów (*G, PG-13, PG, NC-17, R*) wypisz liczbę filmów do niej należących.

2.6.2. Kwerendy niezbędne do realizacji

```
1 •   SELECT rating, COUNT(rating) FROM sakila.film GROUP BY rating;
```

2.6.3. Rezultat

	rating	COUNT(rating)
▶	PG	194
	G	178
	NC-17	210
	PG-13	223
	R	195

2.6.4. Opis rozwiązania

W tym zadaniu po raz pierwszy stosuję grupowanie i funkcję agregującą. Wyświetlam kategorię wiekową i wynik zliczania występowań danej kategorii (funkcja COUNT()) z tabeli sakila.film. W tym celu grupuję rekordy względem kategorii wiekowych (GROUP BY) i stosuję funkcję zliczającą COUNT().

2.7. Zadanie 7

2.7.1. Polecenie

Wypisz tytuły filmów wypożyczonych pomiędzy 25 a 30 maja 2005. Wyniki posortuj alfabetycznie.

2.7.2. Kwerendy niezbędne do realizacji

```
1 •  SELECT DISTINCT film.title FROM sakila.film
2    INNER JOIN sakila.inventory ON inventory.film_id=film.film_id
3    INNER JOIN sakila.rental ON rental.inventory_id=inventory.inventory_id
4    WHERE rental.rental_date BETWEEN '2005-05-25 00:00:00' AND '2005-05-30 23:59:59'
5    ORDER BY film.title;
```

2.7.3. Rezultat

title
ACADEMY DINOSAUR
AFFAIR PREJUDICE
AFRICAN EGG
AGENT TRUMAN
AIRPORT POLLOCK
ALAMO VIDEOTAPE
ALASKA PHANTOM
ALIEN CENTER
ALLEY EVOLUTION
ALONE TRIP
ALTER VICTORY
AMADEUS HOLY
AMERICAN CIRCUS
ANACONDA CONFES...
ANGELS LIFE
ANNIE IDENTITY
ANTHEM LUKE

Result 1

2.7.4. Opis rozwiązania

Łączę ze sobą trzy tabele: sakila.film z sakila.inventory poprzez klucz film_id oraz sakila.inventory z sakila.rental poprzez klucz inventory_id. Dzięki temu mogę pozyskać kolumnę rental_date kluczową dla tego zadania. W warunku korzystam ze słów kluczowych BETWEEN … AND … i zamieszczam przedział, w którym ma się znajdować data wypożyczenia filmu. Ponieważ dany tytuł mógł być wypożyczany kilka razy w tym okresie, korzystam ze słowa DISTINCT, aby uniknąć powtórzeń. Wyniki sortuję po tytule rosnąco.

2.8. Zadanie 8

2.8.1. Polecenie

Wypisz tytuły 5 najdłuższych filmów o kategorii R.

2.8.2. Kwerendy niezbędne do realizacji

```
1 •   SELECT title FROM sakila.film
2     WHERE rating='R' ORDER BY length DESC LIMIT 5;
```

2.8.3. Rezultat

	title
▶	SWEET BROTHERHOOD
	SOLDIERS EVOLUTION
	HOME PITY
	SMOOCHY CONTROL
	SATURN NAME

2.8.4. Opis rozwiązania

Wybieram kolumnę title z tabeli sakila.film i daję warunek, że film należy do kategorii R. Wyniki sortuję malejąco po długości filmów (słowo kluczowe DESC) i ograniczam liczbę wyświetlanych rekordów do 5 (LIMIT 5).

2.9. Zadanie 9

2.9.1. Polecenie

Nie używając grupowania wypisz imiona oraz nazwiska wszystkich klientów, którzy byli obsłużeni przez 2 różnych pracowników.

2.9.2. Kwerendy niezbędne do realizacji

```
1 •  SELECT DISTINCT pomb.customer_id, customer.first_name, customer.last_name FROM
2    (SELECT *, (COUNT(staff_id) OVER (PARTITION BY customer_id)) AS rozni FROM
3    (SELECT DISTINCT staff_id, customer_id FROM rental ORDER BY customer_id) poma) pomb
4    INNER JOIN customer ON customer.customer_id = pomb.customer_id WHERE pomb.rozni = 2;
```

2.9.3. Rezultat

customer_id	first_name	last_name
1	MARY	SMITH
2	PATRICIA	JOHNSON
3	LINDA	WILLIAMS
4	BARBARA	JONES
5	ELIZABETH	BROWN
6	JENNIFER	DAVIS
7	MARIA	MILLER
8	SUSAN	WILSON
9	MARGARET	MOORE
10	DOROTHY	TAYLOR
11	LISA	ANDERSON
12	NANCY	THOMAS
13	KAREN	JACKSON
14	BETTY	WHITE
15	HELEN	HARRIS
16	SANDRA	MARTIN
17	DONNA	THOMPSON
18	CAROL	GARCIA

Result 1 ×

2.9.4. Opis rozwiązania

W tym zadaniu nie mogę skorzystać z grupowania, zatem skorzystam z funkcji analitycznej PARTITION BY, aby podzielić zbiór rekordów na rozłączne podzbiory, po których będę zliczać.

Zaczynając od wewnątrz, tworzę tablicę pomocniczą poma, w której wyświetlam po jednym rekordzie opisującym fakt obsłużenia danego klienta przez konkretnego pracownika (korzystając ze słowa DISTINCT). Tabelę sortuję po id klienta.

Następna tabela, pomb dzieli zawartość tabeli poma na podzbiory: każdy zbiór zawiera tylko jednego klienta i wszystkich różnych pracowników, którzy go obsługiwali (PARTITION BY customer_id). Liczbę rekordów w danym podzbiorze zliczam, korzystając ze składni COUNT(staff_id) OVER (…) – nadaję rezultatowi alias rozni, używając słowa kluczowego AS.

Główna tabela łączy tabelę pomocniczą pomb z tabelą customer używając klucza customer_id, aby dowiedzieć się imion i nazwisk klientów stojących za danymi id. Kluczowy jest warunek, że w tabeli pomb zostały zliczone dwa rekordy dla danego klienta – oznacza to, że został on obsłużony przez dwóch różnych pracowników. Nie chcemy, aby dany klient się powtarzał, zatem korzystam ze słowa DISTINCT.

2.10. Zadanie 10

2.10.1. Polecenie

Wybierz wszystkich klientów, którzy wypożyczyli więcej filmów niż klient o emailu PETER.MENARD@sakilacustomer.org.

2.10.2. Kwerendy niezbędne do realizacji

```
1 •  SELECT customer.*, COUNT(*) AS liczba FROM sakila.customer
2    INNER JOIN sakila.rental ON customer.customer_id=rental.customer_id
3    GROUP BY customer_id
4    HAVING liczba > (SELECT COUNT(*) FROM sakila.customer
5        INNER JOIN sakila.rental ON customer.customer_id=rental.customer_id
6        WHERE customer.email = 'PETER.MENARD@sakilacustomer.org') ;
```

2.10.3. Rezultat

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update	liczba
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-15 04:57:20	32
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20	27
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20	26
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-15 04:57:20	38
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:36	2006-02-15 04:57:20	28
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36	2006-02-15 04:57:20	33
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:36	2006-02-15 04:57:20	24
10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1	2006-02-14 22:04:36	2006-02-15 04:57:20	25
11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:36	2006-02-15 04:57:20	24
12	1	NANCY	THOMAS	NANCY.THOMAS@sakilacustomer.org	16	1	2006-02-14 22:04:36	2006-02-15 04:57:20	28
13	2	KAREN	JACKSON	KAREN.JACKSON@sakilacustomer.org	17	1	2006-02-14 22:04:36	2006-02-15 04:57:20	27
14	2	BETTY	WHITE	BETTY.WHITE@sakilacustomer.org	18	1	2006-02-14 22:04:36	2006-02-15 04:57:20	28
15	1	HELEN	HARRIS	HELEN.HARRIS@sakilacustomer.org	19	1	2006-02-14 22:04:36	2006-02-15 04:57:20	32
16	2	SANDRA	MARTIN	SANDRA.MARTIN@sakilacustomer.org	20	0	2006-02-14 22:04:36	2006-02-15 04:57:20	28
19	1	RUTH	MARTINEZ	RUTH.MARTINEZ@sakilacustomer.org	23	1	2006-02-14 22:04:36	2006-02-15 04:57:20	24
20	2	SHARON	ROBINSON	SHARON.ROBINSON@sakilacustomer.org	24	1	2006-02-14 22:04:36	2006-02-15 04:57:20	30
21	1	MICHELLE	CLARK	MICHELLE.CLARK@sakilacustomer.org	25	1	2006-02-14 22:04:36	2006-02-15 04:57:20	35
23	2	SARAH	LEWIS	SARAH.LEWIS@sakilacustomer.org	27	1	2006-02-14 22:04:36	2006-02-15 04:57:20	30

Result 3 ×

2.10.4. Opis rozwiązania

Łączę tablicę sakila.customer z sakila.rental poprzez klucz customer_id, dzięki temu mam dostęp do danych z obu tablic. Grupuję rekordy względem customer_id aby zliczać to, ile razy dany klient wypożyczał film. Po raz pierwszy w tym zadaniu stosuję słowo kluczowe HAVING, istotne przy dawaniu warunków do funkcji agregujących po słowie GROUP BY. Jako warunek daję liczbę wypożyczeń większą niż klienta o podanym w treści zadania mailu.

Aby pozyskać informację o tym, ile wypożyczeń na koncie ma ten klient, korzystam z tablicy pomocniczej, gdzie przechowam tę liczbę. Tablica ta korzysta z tego samego złączenia tablic customer i rental oraz posiada warunek nakazujący wyświetlanie rekordów o danym adresie email. Dzięki temu mogę skorzystać z funkcji COUNT bez konieczności grupowania i w tabeli pomocniczej zapisany został tylko jeden rekord – jest nim właśnie szukana liczba wypożyczeń.

Jako rezultat, wyświetlam dane klienta z tabeli customer oraz liczbę wypożyczeń pozyskaną z funkcji COUNT().

2.11. Zadanie 11

2.11.1. Polecenie

Wypisz wszystkie pary aktorów, którzy wystąpili razem w więcej niż jednym filmie. Każda para powinna występować co najwyżej raz. Jeśli występuje para (X, Y), to nie wypisuj pary (Y, X).

2.11.2. Kwerendy niezbędne do realizacji

```
SELECT imie_jeden, nazwisko_jeden, actor.first_name AS imie_dwa, actor.last_name AS nazwisko_dwa
FROM sakila.actor
    INNER JOIN
    (SELECT actor.first_name AS imie_jeden, actor.last_name AS nazwisko_jeden, drugi_aktor
    FROM sakila.actor
    INNER JOIN (SELECT film_actor.actor_id AS pierwszy_aktor, druga.actor_id AS drugi_aktor, COUNT(*)
    FROM
        sakila.film_actor
    INNER JOIN sakila.film_actor AS druga ON film_actor.film_id = druga.film_id
    WHERE
        film_actor.actor_id < druga.actor_id
    GROUP BY pierwszy_aktor , drugi_aktor
    HAVING COUNT(*) > 1
    ORDER BY pierwszy_aktor , drugi_aktor) AS tab_pom ON actor.actor_id = pierwszy_aktor) AS tab_pom2 ON actor.actor_id = drugi_aktor
ORDER BY nazwisko_jeden, imie_jeden, nazwisko_dwa, imie_dwa;
```

2.11.3. Rezultat

imie_jeden	nazwisko_jeden	imie_dwa	nazwisko_dwa
CHRISTIAN	AKROYD	KIM	ALLEN
CHRISTIAN	AKROYD	AUDREY	BAILEY
CHRISTIAN	AKROYD	JESSICA	BAILEY
CHRISTIAN	AKROYD	SCARLETT	BENING
CHRISTIAN	AKROYD	HENRY	BERRY
CHRISTIAN	AKROYD	LAURA	BRODY
CHRISTIAN	AKROYD	LAURENCE	BULLOCK
CHRISTIAN	AKROYD	JON	CHASE
CHRISTIAN	AKROYD	RALPH	CRUZ
CHRISTIAN	AKROYD	RIVER	DEAN
CHRISTIAN	AKROYD	NICK	DEGENERES
CHRISTIAN	AKROYD	CHARLIZE	DENCH
CHRISTIAN	AKROYD	CHRIS	DEPP
CHRISTIAN	AKROYD	SYLVESTER	DERN
CHRISTIAN	AKROYD	KEVIN	GARLAND
CHRISTIAN	AKROYD	MERYL	GIBSON
CHRISTIAN	AKROYD	GREGORY	GOODING
CHRISTIAN	AKROYD	MEG	HAWKE
CHRISTIAN	AKROYD	WOODY	JOLIE
CHRISTIAN	AKROYD	MARY	KEITEL

Result 2

2.11.4. Opis rozwiązania

Najpierw łączę tabelę film_actor samą ze sobą poprzez film_id i daję warunek, że id aktora z jednej tabeli film_actor ma być mniejsze. Grupuję ze sobą wystąpienia par aktorów i zliczam je, aby zostawić te pary, które wystąpiły razem w więcej niż jednym filmie. Następnie wykonuję dwa łączenia tej tabeli pomocniczej z tabelą actor, żeby najpierw wyświetlić imię i nazwisko pierwszej osoby przy pierwszym połączeniu (i przekazać dalej id drugiego aktora) i imię i nazwisko drugiego aktora przy drugim łączeniu. Wyniki sortuję względem najpierw pierwszego aktora a potem drugiego

2.12. Zadanie 12

2.12.1. Polecenie

Wypisz nazwiska aktorów, którzy nie wystąpili w żadnym filmie, którego tytuł zaczyna się literą B.

2.12.2. Kwerendy niezbędne do realizacji

```
SELECT actor.last_name FROM actor INNER JOIN
    (SELECT DISTINCT film_actor.actor_id AS aktor_id FROM film_actor WHERE film_actor.actor_id NOT IN
        (SELECT DISTINCT film_actor.actor_id FROM film_actor INNER JOIN film ON film.film_id = film_actor.film_id
        WHERE film.title LIKE 'B%')
    ) AS tab_pom ON actor.actor_id=aktor_id;
```

2.12.3. Rezultat

last_name
AKROYD
BACALL
BERGEN
BERGMAN
BOLGER
CAGE
CHAPLIN
CRONYN
CROWE
DEPP
DEPP
FAWCETT
GABLE
GIBSON
HAWKE
HESTON
HOFFMAN
HOPKINS
HOPKINS
JOHANSS...

Result 1

2.12.4. Opis rozwiązania

Na początek w zapytaniu pomocniczym wyznaczam wszystkich różnych aktorów, którzy wystąpili w filmie, którego tytuł zaczyna się na literę B. W tym celu łączę tablice film_actor i film kluczem film_id i korzystam ze słów kluczowych WHERE … LIKE wzorzec.

W kolejnym zapytaniu pomocniczym tab_pom wybieram tych aktorów, którzy w poprzednim zapytaniu nie wystąpili – to oznacza, że nie wystąpili w takim filmie.

Ostatecznie łączę tablicę actor z tab_pom po actor_id i dzięki temu wyświetlam nazwiska aktorów.

2.13. Zadanie 13

2.13.1. Polecenie

Wypisz nazwiska aktorów, którzy nie wystąpili w żadnym filmie, którego tytuł zaczyna się literą B.

2.13.2. Kwerendy niezbędne do realizacji

```
SELECT  last_name, actor.actor_id FROM actor INNER JOIN
    (
    (SELECT actor_id, COUNT(*) AS no_horror, no_akcja FROM film_actor
    INNER JOIN film_category ON film_actor.film_id=film_category.film_id
    INNER JOIN category ON film_category.category_id=category.category_id
    LEFT JOIN
        (SELECT actor_id AS id_akcja, COUNT(*) AS no_akcja FROM film_actor
        INNER JOIN film_category ON film_actor.film_id=film_category.film_id
        INNER JOIN category ON film_category.category_id=category.category_id
        WHERE category.name = 'Action'
        GROUP BY actor_id) AS tab_pom
    ON film_actor.actor_id=tab_pom.id_akcja
    WHERE category.name = 'Horror'
    GROUP BY actor_id)

    UNION


    (SELECT actor_id, no_horror, COUNT(*) AS no_akcja FROM film_actor
    INNER JOIN film_category ON film_actor.film_id=film_category.film_id
    INNER JOIN category ON film_category.category_id=category.category_id
    LEFT JOIN
        (SELECT actor_id AS id_horror, COUNT(*) AS no_horror FROM film_actor
        INNER JOIN film_category ON film_actor.film_id=film_category.film_id
        INNER JOIN category ON film_category.category_id=category.category_id
        WHERE category.name = 'Horror'
        GROUP BY actor_id) AS tab_pom
    ON film_actor.actor_id=tab_pom.id_horror
    WHERE category.name = 'Action'
    GROUP BY actor_id)
    ) AS tab_agreg
  ON actor.actor_id = tab_agreg.actor_id
  WHERE IFNULL(tab_agreg.no_horror, 0) > IFNULL(tab_agreg.no_akcja,0);
```

2.13.3. Rezultat

last_name
AKROYD
ALLEN
ALLEN
ALLEN
ASTAIRE
BACALL
BALE
BARRYMORE
BASINGER
BERGEN
BERGMAN
BERRY
BERRY

Result 2

2.13.4. Opis rozwiązania

Poważnym mankamentem w MySQL przy rozwiązywaniu tego zadania jest brak połączenia FULL OUTER JOIN. Zamiast niego zmuszony jestem wykonać dwa razy LEFT JOIN i rezultaty zsumować sumą w rozumieniu teoriomnogościowym (UNION). W pierwszym połączeniu lewostronnym zliczam wystąpienia każdego aktora w horrorach z uwzględnieniem sytuacji, gdzie dany aktor nie grał w żadnym horrorze (pojawia się wtedy NULL) oraz zliczam wystąpienia w filmach akcji ale niestety bez uwzględnienia przypadku zerowego. Następnie postępuję analogicznie z drugim złączeniem (tym razem uwzględniam braki wystąpienia w filmach akcji).

Ostatecznie wyniki dwóch złączeń sumuję ze sobą i muszę jeszcze poradzić sobie z wartościami NULL. Robię to już w samym sprawdzaniu czy dany aktor wystąpił więcej razy w horrorach niż w filmach akcji z pomocą funkcji IFNULL(nazwa_kolumny_do _sprawdzenia, wartość_gdy_jest_null) (gdy nie ma wartości NULL to pozostaje wartość pierwotna).

2.14. Zadanie 14

2.14.1. Polecenie

Wypisz wszystkich klientów, których średnia opłata za wypożyczony film jest wyższa niż średnia opłata dokonana 7 lipca 2005.

2.14.2. Kwerendy niezbędne do realizacji

```
1 •  SELECT customer.* FROM customer INNER JOIN
2        (SELECT customer_id, srednia_klienta FROM
3            (SELECT customer_id, AVG(amount) AS srednia_klienta FROM payment
4            GROUP BY customer_id) AS tab_pom
5        WHERE srednia_klienta > (SELECT avg(amount) FROM payment
6            WHERE payment_date BETWEEN '2005-07-07 00:00:00' AND '2005-07-07 23:59:59')) AS tab_pom2
7        ON customer.customer_id = tab_pom2.customer_id;
```

2.14.3. Rezultat

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36	2006-02-15 04:57:20
11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:36	2006-02-15 04:57:20
13	2	KAREN	JACKSON	KAREN.JACKSON@sakilacustomer.org	17	1	2006-02-14 22:04:36	2006-02-15 04:57:20
17	1	DONNA	THOMPSON	DONNA.THOMPSON@sakilacustomer.org	21	1	2006-02-14 22:04:36	2006-02-15 04:57:20
19	1	RUTH	MARTINEZ	RUTH.MARTINEZ@sakilacustomer.org	23	1	2006-02-14 22:04:36	2006-02-15 04:57:20
21	1	MICHELLE	CLARK	MICHELLE.CLARK@sakilacustomer.org	25	1	2006-02-14 22:04:36	2006-02-15 04:57:20
22	1	LAURA	RODRIGUEZ	LAURA.RODRIGUEZ@sakilacustomer.org	26	1	2006-02-14 22:04:36	2006-02-15 04:57:20
26	2	JESSICA	HALL	JESSICA.HALL@sakilacustomer.org	30	1	2006-02-14 22:04:36	2006-02-15 04:57:20
32	1	AMY	LOPEZ	AMY.LOPEZ@sakilacustomer.org	36	1	2006-02-14 22:04:36	2006-02-15 04:57:20
33	2	ANNA	HILL	ANNA.HILL@sakilacustomer.org	37	1	2006-02-14 22:04:36	2006-02-15 04:57:20
39	1	DEBRA	NELSON	DEBRA.NELSON@sakilacustomer.org	43	1	2006-02-14 22:04:36	2006-02-15 04:57:20
41	1	STEPHANIE	MITCHELL	STEPHANIE.MITCHELL@sakilacustomer.org	45	1	2006-02-14 22:04:36	2006-02-15 04:57:20
44	1	MARIE	TURNER	MARIE.TURNER@sakilacustomer.org	48	1	2006-02-14 22:04:36	2006-02-15 04:57:20
45	1	JANET	PHILLIPS	JANET.PHILLIPS@sakilacustomer.org	49	1	2006-02-14 22:04:36	2006-02-15 04:57:20
47	1	FRANCES	PARKER	FRANCES.PARKER@sakilacustomer.org	51	1	2006-02-14 22:04:36	2006-02-15 04:57:20
48	1	ANN	EVANS	ANN.EVANS@sakilacustomer.org	52	1	2006-02-14 22:04:36	2006-02-15 04:57:20

Result 1 ×

2.14.4. Opis rozwiązania

W tabeli pomocniczej tab_pom wyznaczam średnią opłatę za wyznaczony film dla każdego z klientów, korzystając z grupowania po klientach i funkcji AVG(amount).

W tabeli pomocniczej tab_pom2 pobieram dane z tab_pom i zapisuję tylko tych klientów których średnia jest wyższa niż średnia opłata dokonana 7 lipca 2005 (aby wyznaczyć tę średnią korzystam z jeszcze jednego zapytania pomocniczego z funkcją AVG(amount) i warunkiem daty między 7 lipca godz. 00:00 a godz. 23:59 - BETWEEN)

W zapytaniu głównym łączę tabelę customer z tabelą pomocniczą tab_pom2 po customer_id i wyświetlam wszystkie informacje z tabeli customer.

2.15. Zadanie 15

2.15.1. Polecenie

Do tabeli language dodaj kolumnę films_no i uzupełnij ją liczbą filmów w danym języku.

2.15.2. Kwerendy niezbędne do realizacji

```
1 •  ALTER TABLE sakila.language
2    ADD COLUMN film_no int NOT NULL DEFAULT 0;
```

```
1 •  SET SQL_SAFE_UPDATES = 0;
```

```
1 •  UPDATE language
2    INNER JOIN
3    (SELECT language.*, COUNT(*) AS liczba FROM language INNER JOIN film ON language.language_id = film.language_id GROUP BY film.language_id) T_POM
4    ON language.language_id = T_POM.language_id
5    SET language.film_no = T_POM.liczba;
```

2.15.3. Rezultat

W tabeli language powstała nowa kolumna films_no z liczbami filmów w danym języku (początkowo wszystkie 1000 filmów miało język angielski).

2.15.4. Opis rozwiązania

W pierwszej kwerendzie modyfikuję tabelę language korzystając ze słów kluczowych ALTER TABLE. Deklaruję utworzenie nowej kolumny film_no. Będzie to kolumna typu int, nie będzie przyjmować wartości NULL; wartością domyślną jest 0.

W drugiej kwerendzie muszę wyłączyć zabezpieczenia obecne w MySQL Server, aby móc zmieniać dane nowoutworzonej kolumny.

Trzecia kwerenda łączy tabelę language z tabelą pomocniczą. W tabeli pomocniczej zliczam liczbę filmów występujących w danym języku, do tego potrzebuję połączyć tabelę language z tabelą film oraz skorzystać z grupowania po language_id i użyć funkcji zliczającej COUNT. Następnie w tej kwerendzie wysyłane jest polecenie ustawienia wartości kolumny flim_no na tą jaka została wyliczona w tabeli pomocniczej (czyli de facto liczbę filmów w tym języku).

2.16. Zadanie 16

2.16.1. Polecenie

Zmień język filmu WON DARES na mandaryński oraz wszystkich filmów, w których występuje NICKWAHLBERG na niemiecki. Zaktualizuj dane w tabeli language.

2.16.2. Kwerendy niezbędne do realizacji

```
1 • UPDATE film
2   SET language_id = 4
3   WHERE title = 'WON DARES';
```

```
1 • UPDATE film INNER JOIN film_actor ON film.film_id = film_actor.film_id INNER JOIN actor ON film_actor.actor_id = actor.actor_id
2   SET language_id = 6 WHERE actor.first_name = 'NICK' AND actor.last_name = 'WAHLBERG';
```

```
1 • UPDATE language
2   INNER JOIN
3   (SELECT language.*, COUNT(*) AS liczba FROM language INNER JOIN film ON language.language_id = film.language_id GROUP BY film.language_id) T_POM
4   ON language.language_id = T_POM.language_id
5   SET language.film_no = T_POM.liczba;
```

2.16.3. Rezultat

Zmianie uległy tabele film i language (nowa tabela language poniżej)

language_id	name	last_update	film_no
1	English	2021-10-24 15:44:10	974
2	Italian	2006-02-15 05:02:19	0
3	Japanese	2006-02-15 05:02:19	0
4	Mandarin	2021-10-24 15:44:10	1
5	French	2006-02-15 05:02:19	0
6	German	2021-10-24 15:44:10	25
NULL	NULL	NULL	NULL

2.16.4. Opis rozwiązania

We wszystkich kwerendach korzystam ze składni UPDATE nazwa_tabeli SET rzeczy_do_zmiany WHERE warunek. W pierwszej kwerendzie ustawiam language_id na 4 (jest to id języka mandaryńskiego) w rekordach o tytule 'WON DARES' (jest taki jeden). W drugiej kwerendzie łączę tablicę film z film_actor po kluczu film_id po to, aby znaleźć filmy, w których występuje NICK WAHLBERG (dwuwarunkowe WHERE … AND …). Tylko dla tych rekordów następuje zmiana language_id na 6 (jest to id języka niemieckiego). Ostatnia kwerenda jest taka sama jak w zadaniu 15., aktualizuję tabelę language.

2.17. Zadanie 17

2.17.1. Polecenie

Usuń kolumnę release_year z tabeli film.

2.17.2. Kwerendy niezbędne do realizacji

```
1 • ALTER TABLE film
2   DROP COLUMN release_year;
```

2.17.3. Rezultat

Usunięto kolumnę release_year z tabeli film

2.17.4. Opis rozwiązania

Korzystam ze słów kluczowych ALTER TABLE aby zmienić strukturę tablicy film, następnie precyzuję, że chcę usunąć kolumnę release_year frazą DROP COLUMN.