# Lab 1

## Rafał Włodarczyk

2024-10-06

Download the sources via:

- index
- db.tar.gz

```
# Maintenance

## Up DB
docker compose up -d
## Down DB
docker compose down && docker volume rm lista1_db_data

# Execution

## Root user
docker exec -it mariadb-container mariadb -D db2024 -u root -prootpassword
## Normal user
docker exec -it mariadb-container mariadb -D db2024 -u rafisto -
prafistopassword
```

```
-- inside mariadb
source /docker-entrypoint-initdb.d/001-init-schema.sql
source /docker-entrypoint-initdb.d/002-init-data.sql
```

# Solves

1. Wypisz wszystkie znajdujące się w bazie tabele.

```
SHOW TABLES;
```

2. Wypisz tytuły filmów o długości większej niż 2 godziny.

Zakładamy większe-równe dwie godziny.

```
SELECT title
FROM film
```

```
WHERE length > 120;
```

3. Wypisz tytuły 4 najkrótszych filmów o kategorii wiekowej PG-13.

```
SELECT title, length
FROM film
WHERE rating='PG-13'
ORDER BY length ASC
LIMIT 4;
```

4. Wypisz tytuły filmów oraz ich język, dla wszystkich filmów, w których opisie występuje słowo Drama.

```
SELECT film.title, language.name
FROM film
INNER JOIN language ON film.language_id=language.language_id
WHERE description LIKE '%Drama%';
```

5. Wypisz tytuły filmów z kategorii Family, które w swoim opisie zawierają słowo Documentary.

```
SELECT film.title
FROM film
INNER JOIN film_category ON film.film_id=film_category.film_id
INNER JOIN category ON film_category.category_id=category.category_id
WHERE category.name="Family" AND film.description LIKE "%Documentary%";
```

6. Wypisz tytuły filmów z kategorii Children, które nie należą do kategorii wiekowej PG-13.

```
SELECT film.title
FROM film
INNER JOIN film_category ON film.film_id=film_category.film_id
INNER JOIN category ON film_category.category_id=category.category_id
WHERE category.name="Children" AND film.rating!="PG-13";
```

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

```
SELECT rating, COUNT(film_id)
FROM film
GROUP BY rating;
```

8. Wypisz tytuły filmów wypożyczonych pomiędzy 31 maja a 30 czerwca 2005. Wyniki posortuj w odwrotnej kolejności alfabetycznej.

Wypożyczonych, ale niekoniecznie zwróconych

```
SELECT film.title, rental.rental_date
FROM film
INNER JOIN inventory ON film.film_id=inventory.film_id
INNER JOIN rental ON inventory.inventory_id=rental.rental_id
WHERE rental.rental_date > "2005-05-31" AND rental.rental_date < "2005-06-
30";
```

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

```
SELECT actor.first_name, actor.last_name, film.title
FROM film
INNER JOIN film_actor ON film.film_id=film_actor.film_id
INNER JOIN actor ON film_actor.actor_id=actor.actor_id
WHERE film.special_features='Deleted Scenes';
```

10. Wypisz imiona oraz nazwiska wszystkich klientów, których wypożyczenie i odpowiadająca mu płatność były obsłużone przez 2 różnych pracowników.

```
SELECT DISTINCT customer.first_name, customer.last_name
FROM customer
INNER JOIN rental ON customer.customer_id = rental.customer_id
INNER JOIN payment ON rental.rental_id = payment.rental_id
WHERE rental.staff_id != payment.staff_id;
```

11. Wypisz imiona i nazwiska wszystkich klientów, którzy wypożyczyli więcej filmów niż klient o e-mailu `MARY.SMITH@sakilacustomer.org`.

```
SELECT customer.first_name, customer.last_name, COUNT(rental.rental_id) AS
rental_count
FROM customer
INNER JOIN rental ON customer.customer_id = rental.customer_id
GROUP BY customer.first_name, customer.last_name
HAVING COUNT(rental.rental_id) > (
    SELECT COUNT(rental.rental_id)
    FROM customer
    INNER JOIN rental ON customer.customer_id = rental.customer_id
    WHERE customer.email = 'MARY.SMITH@sakilacustomer.org'
)
ORDER BY rental_count DESC;
```

12. 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)

Zliczmy ile wspólnych, ale z joinem tak, że pierwszy index aktora jest zawsze mniejszy od drugiego (bez powtórzeń par XY YX)

```
SELECT act1.first_name,
       act1.last_name,
       act2.first_name,
       act2.last_name,
       COUNT(*) AS common_films
FROM film_actor a1
INNER JOIN film_actor a2 ON a1.film_id = a2.film_id AND a1.actor_id <
a2.actor_id
INNER JOIN actor act1 ON a1.actor_id = act1.actor_id
INNER JOIN actor act2 ON a2.actor_id = act2.actor_id
GROUP BY a1.actor_id, a2.actor_id
HAVING common_films > 1;
```

13. Wypisz nazwiska aktorów, którzy nie wystąpili w żadnym filmie, którego tytuł zaczyna się na literę C.

Użyję klauzuli `NOT IN`. Weźmy tych którzy wystąpili a potem odejmijmy od całości:

```
SELECT actor.first_name, actor.last_name
FROM actor
WHERE actor.actor_id NOT IN (
SELECT actor.actor_id
FROM film
INNER JOIN film_actor ON film.film_id=film_actor.film_id
INNER JOIN actor ON film_actor.actor_id=actor.actor_id
WHERE title LIKE "C%"
);
```

14. Wypisz nazwiska aktorów, którzy zagrali w większej liczbie horrorów niż filmów akcji

Zliczmy horrory i akcje następnie having.

```
SELECT actor.first_name, actor.last_name,
       SUM(CASE WHEN category.name = 'Horror' THEN 1 ELSE 0 END) AS
horror_films,
       SUM(CASE WHEN category.name = 'Action' THEN 1 ELSE 0 END) AS
action_films
FROM actor
INNER JOIN film_actor ON actor.actor_id = film_actor.actor_id
INNER JOIN film ON film.film_id = film_actor.film_id
INNER JOIN film_category ON film.film_id = film_category.film_id
```

```
INNER JOIN category ON film_category.category_id = category.category_id
GROUP BY actor.actor_id
HAVING SUM(CASE WHEN category.name = 'Horror' THEN 1 ELSE 0 END) >
       SUM(CASE WHEN category.name = 'Action' THEN 1 ELSE 0 END);
```

15. Wypisz wszystkich klientów, których średnia opłata za wypożyczony film jest niższa niż średnia opłata dokonana `30 lipca 2005`

Policzmy średnią 30 lipca 2005, następnie średnią dla każdego klienta.

```
-- srednia z 30 lipca
SELECT AVG(amount)
FROM payment
WHERE payment.payment_date > "2005-07-30"
AND payment.payment_date < "2005-07-31";

-- fianl query
SELECT customer.first_name, customer.last_name, AVG(payment.amount) AS
avg_payment
FROM customer
INNER JOIN payment ON customer.customer_id = payment.customer_id
GROUP BY customer.customer_id
HAVING AVG(payment.amount) < (
    SELECT AVG(amount)
    FROM payment
    WHERE payment.payment_date > '2005-07-30'
    AND payment.payment_date < '2005-07-31'
)
ORDER BY avg_payment DESC;
```

16. Zmień język filmu `YOUNG LANGUAGE` na włoski.

```
-- italian language_id
SELECT language_id FROM language WHERE name = 'Italian';

-- final query
UPDATE film
SET language_id = (SELECT language_id FROM language WHERE name = 'Italian')
WHERE film.title="YOUNG LANGUAGE";

-- check
SELECT film.title, language.name
FROM film
INNER JOIN language ON film.language_id=language.language_id
WHERE film.title="YOUNG LANGUAGE";
```

17. Dodaj do tabeli language język hiszpański i zmień język wszystkich filmów, w których występuje `ED CHASE` na hiszpański.

```
-- add spanish
INSERT INTO language (name, last_update)
VALUES ('Spanish', current_timestamp());

-- update all movies with ED CHASE
UPDATE film
SET language_id = (SELECT language_id FROM language WHERE name = 'Spanish')
WHERE film_id IN (
    SELECT film.film_id
    FROM film
    JOIN film_actor ON film.film_id = film_actor.film_id
    JOIN actor ON film_actor.actor_id = actor.actor_id
    WHERE actor.first_name = 'ED' AND actor.last_name = 'CHASE'
);

-- changed 22 (backlog)
```

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

Unsigned gwarantuje liczbę większą-równą 0.

```
ALTER TABLE language
ADD COLUMN films_no INT UNSIGNED;

-- check
DESCRIBE language;

-- count films
UPDATE language
SET language.films_no = (
    SELECT COUNT(*)
    FROM film
    WHERE film.language_id = language.language_id
);
```

19. Usuń kolumnę `release_year` z tabeli film

```
ALTER TABLE film
DROP COLUMN release_year;

-- check
DESCRIBE film;
```