1. Utwórz bazę danych Laboratorium-Filmoteka. Utwórz użytkownika ’idx’@’localhost’ (lub ’idx’@’%’), gdzie idx jest twoim numerem indeksu. Ustaw dla tego użytkownika hasło będące konkatenacją twojego imienia i trzech ostatnich cyfr idx. Nadaj utworzonemu użytkownikowi uprawnienia do Selectowania, wstawiania i zmieniania danych w tabeli, jednak nie do tworzenia usuwania i modyfikowania tabel.

CREATE DATABASE `Laboratorium-Filmoteka`;

CREATE USER '111111'@'localhost';

SET PASSWORD FOR '111111'@'localhost' = 'piotr111';

GRANT SELECT, INSERT, UPDATE ON `Laboratorium-Filmoteka`.* TO '111111'@'localhost';

FLUSH PRIVILEGES;

2. Utwórz tabele aktorzy oraz filmy oraz zagrali zawierające informacje na temat odpowiednio:

• imion i nazwisk aktorów;

• tytułów, gatunków, czasu trwania oraz kategorii wiekowej filmów;

• aktorów, którzy zagrali w danym filmie;

Możesz używać dodatkowych kolumn zapewniających unikalność danych. Utworzone tabele uzupełnij danymi z bazy sakila, pomiń aktorów i tytuły filmu, dla których potrzebne są znaki nie występujące w języku polskim (np. x, v).

CREATE TABLE aktorzy (id int PRIMARY KEY, imie varchar(20), nazwisko varchar(20));

CREATE TABLE filmy (id int PRIMARY KEY, tytul varchar(20), gatunek varchar(15), czas_trwania int, kategoria_wiekowa varchar(10));

CREATE TABLE zagrali (actor_id int not null, film_id int not null);

INSERT INTO `Laboratorium-Filmoteka`.aktorzy (id,imie,nazwisko) (SELECT actor_id,first_name,last_name FROM actor WHERE first_name not like '%Q%' and first_name not like '%X%' and first_name not like '%V%' and last_name not like '%Q%' and last_name not like '%X%' and last_name not like '%V%')

insert into `Laboratorium-Filmoteka`.filmy (id,tytul,gatunek,czas_trwania,kategoria_wiekowa) (select c.film_id,title,name,length,rating from (select a.film_id,title,length,rating,category_id from film a join film_category b on a.film_id=b.film_id) c join category d on c.category_id=d.category_id where title not like '%Q%' and title not like '%X%' and title not like '%V%');

insert into `Laboratorium-Filmoteka`.zagrali (actor_id,film_id) (select actor_id,film_id from film_actor where actor_id in (select id from `Laboratorium-Filmoteka`.aktorzy) and film_id in (select id from `Laboratorium-Filmoteka`.filmy));

3. Zmodyfikuj tabelę aktorzy, dodając dla każdego z aktorów liczbę filmów, w których zagrali oraz kolumnę zawierającą listę tytułów filmów, dla aktorów, którzy zagrali w mniej niż 4 produkcjach.

alter table aktorzy add liczba_filmow int after nazwisko;

update aktorzy set liczba_filmow=(select liczba from (select actor_id, count(*) as liczba from (select distinct * from zagrali) a group by actor_id) b where id=actor_id);

alter table aktorzy add tytuly varchar(100) after liczba_filmow;

update aktorzy set tytuly=(select tytuly from (select actor_id, group_concat(tytul) as tytuly from (select * from zagrali where actor_id in (select id from aktorzy where liczba_filmow<4)) a join filmy on film_id=id group by actor_id) b where b.actor_id=id);

4. Utwórz tabele:

(a) Agenci(licencja:varchar(30), nazwa: varchar(90), wiek: int, typ: (’osoba indywidualna’, ’agencja’,’inny’))

(b) Kontrakty(ID: int, agent:varchar(30), aktor: int, początek: date, koniec: date, gaża: int)

Zadbaj o to, by podkreślone atrybuty były kluczami głównymi, tam gdzie to możliwe zastosuj automatyczną inkrementację. Zadbaj by agent zawsze miał przynajmniej 21 lat. W tabeli Kontrakty atrybuty agent oraz aktor powinny być kluczami obcymi. Ponadto, kontrakt może się kończyć najwcześniej dzień po dacie rozpoczęcia, a gaża nie może być ujemna.

create table agenci (licencja varchar(30) PRIMARY KEY, nazwa varchar(30), wiek int, typ enum('osoba indywidualna', 'agencja','inny');

create table kontrakty (ID int auto_increment, agent varchar(30), aktor int, początek date, koniec date, gaża int, PRIMARY KEY (ID), FOREIGN KEY (agent) REFERENCES agenci(licencja), FOREIGN KEY (aktor) REFERENCES aktorzy(id));

```
DELIMITER $$

CREATE TRIGGER update(insert)_test_wieku BEFORE UPDATE(insert) ON agenci

FOR EACH ROW

BEGIN

  IF new.wiek < 21 THEN

    SIGNAL SQLSTATE '12345'

      SET MESSAGE_TEXT = 'Błąd. Wiek poniżej 21 lat';

  END IF;

        END$$

        DELIMITER;


DELIMITER $$

        CREATE TRIGGER update(insert)_kontrakt_test BEFORE UPDATE(insert) ON kontrakty

        FOR EACH ROW

        BEGIN

        IF new.gaża<0 or (new.początek+interval 1 DAY)>new.koniec THEN

                SIGNAL SQLSTATE '12345'
```

```
SET MESSAGE_TEXT = 'Błędne dane';
END IF;
END;$$
DELIMITER ;
```

5. Napisz procedurę tworzącą 1000 agentów i dodającą ich do tabeli Agenci. Uzupełnij tabelę kontrakty, tak by każdy aktor miał aktualnie dokładnie jednego agenta, decyzja o przydziale agentów powinna być losowa.

```
DELIMITER $$;
create procedure stworz_agentow()
begin
    declare i int;
    declare tekst varchar(24);
    set tekst = "ABCDEFGHIJKLMNOPRSTUWXYZ";
    set i=1;
    while i<=1000 do
        insert into agenci values(

concat(substr(tekst,1+floor(rand()*24),1),substr(tekst,1+floor(rand()*24),1),substr(tekst,1+floor(rand()*24),1),i),

concat(substr(tekst,1+floor(rand()*24),1),substr(tekst,1+floor(rand()*24),1),substr(tekst,1+floor(rand()*24),1)),
            floor(21+rand()*40),
            floor(1+rand()*3));
            set i=i+1;
    end while;
END$$;
DELIMITER ;

insert into kontrakty (aktor) select id from aktorzy;
update kontrakty set początek=(date_sub(curdate(), interval 1+floor(rand()*30) day));
update kontrakty set koniec=(date_add(curdate(), interval 1+floor(rand()*30) day));
update kontrakty set gaża=1000+floor(rand()*10000);
```

update kontrakty set agent=(select licencja from agenci order by rand() limit 1);

6. Połącz się z bazą przy pomocy konta użytkownika utworzonego w zadaniu 1. Dodaj do tabeli Kontrakty dane historyczne, tzn. wprowadź 30 nowych rekordów mówiących o zakończonych kontraktach, zadbaj by w każdym dniu, każdy aktor miał co najwyżej jeden kontrakt. Spróbuj wprowadzić rekordy naruszające ograniczenia zdefiniowane przy tworzeniu tabel – jeśli ci się udało, uaktualnij schemat tabel tak, by nie pozwalał na wprowadzanie błędnych danych. Do kolumny gaża dodaj komentarz mówiący o walucie oraz jednostce czasu na jaką przypada wypłata z tabeli.

insert into kontrakty (aktor) (Select id from aktorzy order by rand() limit 30);

update kontrakty set początek=(date_sub(date_sub(curdate(), interval 90 day),interval 1+floor(rand()*30) day)) where początek IS NULL;

update kontrakty set koniec=(date_add(date_sub(curdate(), interval 90 day),interval 1+floor(rand()*30) day)) where koniec IS NULL;

update kontrakty set gaża=(date_sub(date_sub(curdate(), interval 90 day),interval 1+floor(rand()*30) day)) where gaża IS NULL;

update kontrakty set agent=(select licencja from agenci order by rand() limit 1) where agent IS NULL;

alter table kontrakty modify gaża int comment 'PLN za miesiąc';

7. Napisz procedurę lub funkcję, która po podaniu imienia i nazwiska aktora wypisuje aktualnego agenta i liczbę dni do końca kontraktu.

```
DELIMITER $$

create procedure dni_do_konca (fname varchar(30),lname varchar(30))

begin

        declare agent_aktora varchar(30);

        declare dni_do_końca int;

        select nazwa, okres into agent_aktora,dni_do_końca from (Select
nazwa,DATEDIFF(koniec,current_date()) as okres from (select agent, koniec from aktorzy a join
kontrakty b on a.id=b.aktor where imie like fname and nazwisko like lname and
koniec>current_date()) c join agenci d on c.agent=d.licencja) t;

        if agent_aktora is null then

                select 'Obecnie brak kontraktu' as Kontrakt;

        else

                select agent_aktora,dni_do_końca;

        end if;

end $$

DELIMITER ;
```

8. Napisz procedurę lub funkcję, która po podaniu numeru licencji wypisuje średnią wartość aktualnego kontraktu. Zadbaj by przy podaniu błędnego numeru licencji nie były wyświetlane żadne dane.

```
delimiter $$
create procedure srednia_wartosc_licencji(arglicencja varchar(10))
begin
        declare srednia double;
    set srednia=0;
        if arglicencja in (select licencja from agenci) then
            select avg(gaża) into srednia from kontrakty where agent like arglicencja and current_date() between początek and koniec;
            select arglicencja as licencja ,srednia;
        end if;
end $$
delimiter ;
```

9. Za pomocą konstrukcji PREPARE statement przygotuj zapytanie zwracające liczbę unikalnych klientów, których przez całą swoją działalność miał agent o podanej przy EXECUTE nazwie.

```
set @str ='select count(*) as liczbaklientow from (select distinct aktor from kontrakty a join agenci b on a.agent=b.licencja where nazwa like ?) c';
prepare stm from @str;
set @i='nazwa';
execute stm using @i;
deallocate prepare stm;
```

10. Napisz procedurę nie przyjmującą żadnego parametru wejściowego, która do wskazanych zmiennych przypisuje informacje o agencie, który aktualnie najdłużej nieprzerwanie współpracuje z jednym ze swych klientów (współpraca może obejmować wiele kontraktów, jeśli nie było dni przerwy).

```
Delimiter $$
create procedure najdlusza_wspolpraca ()
begin
        declare i,idmax,maxdni int;
        declare date1 date;
        set i=1;
```

```
set maxdni=0;
while i <= (select max(id) from kontrakty) do
if (select koniec from kontrakty where id=i)>current_date() then
set date1=(select początek from kontrakty where id=i);
while exists (select początek from kontrakty where koniec=date1 and aktor=(select aktor from kontrakty where id=i) and agent=(select agent from kontrakty where id=i)) do
set date1=(select początek from kontrakty where koniec=date1 and aktor=(select aktor from kontrakty where id=i) and agent=(select agent from kontrakty where id=i));
end while;
if (select DATEDIFF(current_date(),date1)>maxdni) then
set maxdni=DATEDIFF(current_date(),date1);
set idmax=i;
end if;
end if;
set i=i+1;
end while;
select licencja,nazwa,wiek,typ from agenci a join kontrakty b on a.licencja=b.agent where id=idmax;
end $$
Delimiter ;
```

11. Napisz triggery, które przy uzupełnianiu, aktualizowaniu lub usuwaniu rekordów z tabeli zagrali uaktualniają odpowiednie kolumny w tabeli aktorzy.

```
DELIMITER $$
create trigger insert_zagrali after insert on zagrali
FOR EACH ROW
BEGIN
update aktorzy set liczba_filmow=(select count(*) as liczba from (select distinct * from zagrali where actor_id=new.actor_id) b) where id=new.actor_id;
if (select liczba_filmow from aktorzy where id=new.actor_id)<4 then
update aktorzy set tytuly=(select group_concat(tytul) from (select distinct film_id from zagrali where actor_id=new.actor_id) a
join filmy on a.film_id=id) where id=new.actor_id;
```

```
        else
                update aktorzy set tytuly=null where id=new.actor_id;
        end if;
    END $$
    DELIMITER ;

    DELIMITER $$
    create trigger update_zagrali after update on zagrali
    FOR EACH ROW
    BEGIN
        update aktorzy set liczba_filmow=(select count(*) as liczba from (select distinct *
from zagrali where actor_id=new.actor_id) b) where id=new.actor_id;
        if( new.actor_id<>old.actor_id ) then
            update aktorzy set liczba_filmow=(select count(*) as liczba from (select
distinct * from zagrali where actor_id=old.actor_id) b) where id=old.actor_id;
        end if;
  if (select liczba_filmow from aktorzy where id=old.actor_id)<4 then
            update aktorzy set tytuly=(select group_concat(tytul) from (select distinct
film_id from zagrali where actor_id=old.actor_id) a
    join filmy on a.film_id=id) where id=old.actor_id;
        else
            update aktorzy set tytuly=null where id=old.actor_id;
        end if;
  if new.actor_id<>old.actor_id && (select liczba_filmow from aktorzy where id=new.actor_id)<4
then
            update aktorzy set tytuly=(select group_concat(tytul) from (select distinct
film_id from zagrali where actor_id=new.actor_id) a
    join filmy on a.film_id=id) where id=new.actor_id;
        else
            update aktorzy set tytuly=null where id=new.actor_id;
        end if;
    END $$
    DELIMITER ;
```

```
DELIMITER $$
create trigger delete_zagrali after delete on zagrali
FOR EACH ROW
BEGIN
        update aktorzy set liczba_filmow=(select count(*) as liczba from (select distinct *
from zagrali where actor_id=old.actor_id) b) where id=old.actor_id;
    if (select liczba_filmow from aktorzy where id=old.actor_id)<4 then
                update aktorzy set tytuly=(select group_concat(tytul) from (select distinct
film_id from zagrali where actor_id=old.actor_id) a
      join filmy on a.film_id=id) where id=old.actor_id;
        end if;
END $$
DELIMITER ;
```

12. Napisz trigger, który pozwoli dodać kontrakt pomiędzy istniejącym aktorem a dowolnym agentem. Jeśli agent do tej pory nie istniał, należy dopisać go do odpowiedniej tabeli. Jeśli aktor miał obowiązujący kontrakt, to należy go zerwać.

```
create trigger insert_kontrakt before insert on kontrakty
FOR EACH ROW
BEGIN
        if new.aktor not in (select id from aktorzy) then
                SIGNAL SQLSTATE '12345'
                        SET MESSAGE_TEXT = 'Nie ma takiego aktora';
        end if;
        if new.agent not in (select licencja from agenci) then
                insert into agenci values
(new.agent,'nazwa',21+floor(rand()*40),1+floor(rand()*3));
        end if;
        if exists (select * from kontrakty where kontrakty.aktor=new.aktor and new.początek
between kontrakty.początek and kontrakty.koniec) then
                SIGNAL SQLSTATE '12345'
                        SET MESSAGE_TEXT = 'Aktor ma już kontrakt';
```

```
        end if;
END $$
DELIMITER ;
```

13. Napisz trigger, który przy usuwaniu filmu z tabeli filmy usuwa odpowiednie rekordy z tabeli zagrali. Jak zachowa się w tej sytuacji tabela aktorzy?

```
delimiter $$
create trigger delete_film after delete on filmy
FOR EACH ROW
begin
        delete from zagrali where film_id=old.id;
end $$
delimiter ;
```

14. Napisz widok zawierający informacje jedynie o imieniu i nazwisku aktora, nazwie jego agenta oraz liczbie dni do końca kontraktu. Czy widok może być utworzony przez użytkownika z zadania 1? Czy użytkownik ma do niego dostęp?

create view info as select imie,nazwisko,nazwa as agent,DATEDIFF(koniec,current_date()) as dni_do_konca from aktorzy a join kontrakty b on a.id=b.aktor and koniec>current_date() join agenci on agent=licencja;

Urzytkownik 2 nie może utworzyć widoków, ale może je selektować

15. Utwórz widoki zawierające publiczne informacje o agentach, aktorach i filmach, nie uwzględniaj w nich danych, które mogą być wrażliwe lub nie są używane we wcześniejszych zadaniach. Utwórz nowego użytkownika, który nie ma dostępu do tabel, jednak może pobierać informacje z widoków. Które z zaimplementowanych procedur, funkcji i triggerów może wykonywać? Jakie uprawnienia o tym mówią?

```
create view agent_info as (select licencja, nazwa, wiek from agenci);
create view aktor_info as (select id,imie,nazwisko,liczba_filmow,tytuly from aktorzy);
create view film_info as (select id,tytul from filmy);


CREATE USER 'newuser'@'localhost';
GRANT select ON `laboratorium-filmoteka`.agent_info TO 'newuser'@'localhost';
GRANT select ON `laboratorium-filmoteka`.aktor_info TO 'newuser'@'localhost';
GRANT select ON `laboratorium-filmoteka`.film_info TO 'newuser'@'localhost';
FLUSH PRIVILEGES;
```