1.Dla tabel z zadania 2 na liście 2 utwórz indeksy:

• dla tabeli filmy na tytule

• dla tabeli aktorzy na nazwisku i pierwszej literze imienia

• dla tabeli zagrali na aktorze

Wskaż, które z podanych indeksów już istniały.

CREATE INDEX filmy_inx ON filmy(tytul);

CREATE INDEX aktorzy_inx ON aktorzy(nazwisko,imie(1));

CREATE INDEX zagrali_inx ON zagrali(actor_id);

show index from (tabela)

2. Dla tabeli Kontrakty z listy 2 utwórz indeks na kolumnie koniec. Korzystając z indeksu wybierz wszystkich aktorów, którym kończy się kontrakt w ciągu następnego miesiąca (nie wpisuj daty na sztywno). Wybierz jeden z typów indeksów HASH lub BTREE, uzasadnij swój wybór.

CREATE INDEX kontrakty_inx ON kontrakty(koniec) using btree;

select aktor from kontrakty use index (kontrakty_inx) where koniec between current_date() and date_add(current_date(), interval 30 day);

B-Tree ponieważ pozwala stosować zakres przy wyszukiwaniu, gdzie hash stosuje się tylko dal porównań = lub <>

3. Napisz odpowiednie kwerendy, i uzasadnij w których przypadkach wykorzystane zostaną indeksy utworzone w zadaniu 1 i 2.

• Wypisz imiona aktorów rozpoczynające się na J.

• Wypisz nazwiska aktorów, którzy zagrali w przynajmniej 12 filmach.

• Wybierz tytuły wszystkich filmów, w których grał jakikolwiek aktor mający na koncie wspólny film z Zero Cage’em.

• Wybierz aktora, któremu do końca kontraktu pozostało najmniej czasu.

• Wybierz najpopularniejsze imię wśród aktorów.

select distinct imie from aktorzy where imie like 'j%';

Nie, ponieważ indeksy wielokolumnowe są konkatenacjami więc litera J znajduje się na końcu indeksu.

select distinct nazwisko from aktorzy where liczba_filmow>=12;

Nie, bo nie ma indeksu na liczbie filmów.

select tytul from (select distinct film_id from zagrali where actor_id in (select distinct actor_id from zagrali where film_id in (select film_id from zagrali where actor_id=(select id from aktorzy where nazwisko like 'Cage' and imie like 'Zero')))) a join filmy where film_id=id;

Tak, przy szukaniu id podanego aktora.

select aktor from (select aktor, datediff(koniec,current_date()) as dni from kontrakty where koniec>current_date()) a join (select min(dni) as min from (select datediff(koniec,current_date()) as dni from kontrakty where koniec>current_date()) b) c where dni=min;

Tak, ponieważ wybieramy koniec na którym jest index.

select imie from (select imie,count(*) as liczba from aktorzy group by imie) a join (select max(liczba) as maks from (select count(*) as liczba from aktorzy group by imie)b) c where liczba=maks;

Nie, indeks jest konkatenacją nazwiska i pierwszej litery imienia, a tutaj nie szukmy po nazwisku.

4. Utwórz nową bazę danych o dowolnej nazwie a następnie utwórz tabele:

• Ludzie(PESEL:char(11),imię:varchar(30),nazwisko:varchar(30),data urodzenia:date, wzrost: float, waga: float,rozmiar buta:int, ulubiony kolor: (’czarny’, ’czerwony’,’zielony’, ’niebieski’,’biały’))

• Pracownicy(PESEL: char(11), zawod:varchar(50), pensja: float).

Zadbaj o prawidłowy format kolumny PESEL. Dopilnuj by wartości liczbowe były nieujemne, a wszyscy pracownicy pełnoletni (Uwaga: nie wszyscy Ludzie muszą być pełnoletni). Dodaj do tabeli Ludzie informacje na temat 200 osób, a następnie przydziel zawody dla:

• 50 aktorów

• 33 agentów

• 13 informatyków (dodatkowo żaden z informatyków nie może zarabiać ponad 3x więcej od dowolnego innego informatyka)

• 2 reporterów

• 77 sprzedawców (dodatkowo żaden ze sprzedawców nie może być starszy niż 65 lat).

```
create database Pracownicy;

create table Ludzie (PESEL char(11) Primary Key,imię varchar(30),nazwisko
varchar(30),data_urodzenia date, wzrost float, waga float, rozmiar_buta int, ulubiony_kolor
enum('czarny', 'czerwony','zielony', 'niebieski','biały'));

create table Pracownicy (PESEL char(11) primary key, zawod varchar(50), pensja float, foreign key
(pesel) references ludzie(pesel));

        delimiter $$

        create trigger update(insert)_pesel_ludzie(pracownicy) before update on ludzie(pracownicy)

        for each row

        begin

                declare i int;

                set i=1;

                if char_length(pesel)<>11 then

                        SIGNAL SQLSTATE '12346'
```

```
                                SET MESSAGE_TEXT = 'Błędny pesel';
        end if;
        while i<=11 do
                if substr(pesel,i,1) NOT REGEXP '[0-9]' then
                        SIGNAL SQLSTATE '12346'
                                SET MESSAGE_TEXT = 'Błędny pesel';
                end if;
                set i=i+1;
        end while;
        end $$
delimiter ;

delimiter $$
create trigger update(insert)_numeric_ludzie before update(insert) on ludzie
for each row
begin
        if new.wzrost<0 or new.rozmiar_buta<0 or new.waga<0 then
                SIGNAL SQLSTATE '12345'
                                SET MESSAGE_TEXT = 'Wartość ujemna';
        end if;
        end $$
delimiter ;

delimiter $$
create trigger insert(update)_numeric_pracownicy before insert(update) on pracownicy
for each row
begin
        if new.pensja<0 then
                SIGNAL SQLSTATE '12345'
                                SET MESSAGE_TEXT = 'Wartość ujemna';
        end if;
```

```
		end $$
	delimiter ;


	delimiter $$
	create trigger update(insert)_wiek_pracownicy before update(insert) on pracownicy
	for each row
	begin
		if (select data_urodzenia from ludzie where pesel like
new.pesel)>date_sub(current_date(),interval 18 year) then
			SIGNAL SQLSTATE '12344'
				SET MESSAGE_TEXT = 'Pracownik nie ma 18 lat';
		end if;
		end $$
	delimiter ;


	delimiter $$
	create procedure insert_ludzie()
	begin
		declare numbers char(10);
		declare i,n int;
		declare spesel char(11);
		set numbers='0123456789';
		set n=1;
		while n<=200 do
			set spesel='';
			set i=1;
			while i<=11 do
				set spesel=concat(spesel,substr(numbers,floor(1+rand()*10),1));
				set i=i+1;
			end while;
		insert into ludzie (pesel,data_urodzenia,wzrost,waga,rozmiar_buta,ulubiony_kolor)
values (spesel,date_sub(current_date(),interval
```

```
                floor(16+rand()*50)
year),floor(140+rand()*60),floor(50+rand()*50),floor(32+rand()*14),floor(1+rand()*4));

                set @im=floor(1+rand()*199);

                update ludzie set imię=(select first_name from sakila.actor where actor_id=@im)
where pesel like spesel;

                set @na=floor(1+rand()*199);

                update ludzie set nazwisko=(select last_name from sakila.actor where actor_id=@na)
where pesel like spesel;

    set n=n+1;

                end while;

        end $$

        delimiter ;


insert into pracownicy (pesel) (select ludzie.pesel from ludzie left join (select pesel from pracownicy)
a on ludzie.pesel=a.pesel where a.pesel is null and date_add(data_urodzenia,interval 18
year)<current_date() order by rand() limit 50);

update pracownicy set zawod='aktor' where zawod is null;

update pracownicy set pensja=(floor(2000+rand()*10000)) where pensja is null;

insert into pracownicy (pesel) (select ludzie.pesel from ludzie left join (select pesel from pracownicy)
a on ludzie.pesel=a.pesel where a.pesel is null and date_add(data_urodzenia,interval 18
year)<current_date() order by rand() limit 33);

update pracownicy set zawod='agent' where zawod is null;

update pracownicy set pensja=(floor(2000+rand()*10000)) where pensja is null;

insert into pracownicy (pesel) (select ludzie.pesel from ludzie left join (select pesel from pracownicy)
a on ludzie.pesel=a.pesel where a.pesel is null and date_add(data_urodzenia,interval 18
year)<current_date() order by rand() limit 13);

update pracownicy set zawod='informatyk' where zawod is null;

update pracownicy set pensja=(floor(3000+rand()*6000)) where pensja is null;

insert into pracownicy (pesel) (select ludzie.pesel from ludzie left join (select pesel from pracownicy)
a on ludzie.pesel=a.pesel where a.pesel is null and date_add(data_urodzenia,interval 18
year)<current_date() order by rand() limit 2);

update pracownicy set zawod='reporter' where zawod is null;

update pracownicy set pensja=(floor(3000+rand()*6000)) where pensja is null;
```

insert into pracownicy (pesel) (select ludzie.pesel from ludzie left join (select pesel from pracownicy) a on ludzie.pesel=a.pesel

where a.pesel is null and date_add(data_urodzenia,interval 18 year)<current_date() and date_add(data_urodzenia,interval 65 year)>current_date() order by rand() limit 77);

update pracownicy set zawod='sprzedawca' where zawod is null;

update pracownicy set pensja=(floor(2000+rand()*10000)) where pensja is null;

5. Napisz procedurę, która przyjmując dwa parametry wejściowe: agg oraz kol wypisuje wynik o schemacie (kol, agg, X), gdzie X jest wynikiem zastosowania funkcji agregującej agg na kolumnie kol w tabeli Ludzie. Załóż obecność użytkownika, próbującego dokonać zmian w strukturze jak i zawartości bazy danych, próbującego wywołać błąd funkcji lub poznać strukturę tabel. Zadbaj o poprawność i bezpieczeństwo działania. Zastanów się jak będą działać dopuszczalne funkcje agregujące na kolumnie ulubiony kolor.

```
delimiter $$
create procedure agreguj (in agg ENUM('sum','avg','max','min','count','variance','std'),in kol varchar(15))
begin
    declare i int;
    declare ascii int;
    if kol IN (SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_SCHEMA = 'pracownicy' AND TABLE_NAME = 'ludzie') then
        set @zapytanie=concat('select ',agg,'(',kol,') from ludzie into @wynik;');
        prepare statement from @zapytanie;
        execute statement;
        deallocate prepare statement;
        select agg as 'f_agregująca', kol as 'kolumna', @wynik as 'wynik';
    else
        set i=1;
        set ascii=0;
        while i<=length(kol) do
            set ascii=ascii+ascii(substr(kol,i,1));
            set i=i+1;
        end while;
        select agg as 'f_agregująca', kol as 'kolumna', ascii as 'wynik';
```

```
        end if;
    end $$
    delimiter ;
```

6. Napisz procedurę, która przyjmując w parametrach wejściowych dostępny budżet oraz zawód, dla którego ma odbyć się wypłata, drukuje tabelę z informacją ('********abc', wypłacono), gdzie abc to trzy ostatnie cyfry PESELu. Wypłata ma miejsce tylko wtedy, gdy każdemu o danym zawodzie można wypłacić jego pensję. Procedura może mieć dostęp jedynie do jednego wiersza na raz, tzn. nie przegląda całej tabli wykonując operację dla jednego pracownika. Wykorzystaj transakcje.

```
    delimiter $$
    create procedure wyplata(in budzet int, in profesja varchar(50))
    begin
        DECLARE done INT DEFAULT 0;
        DECLARE vpesel char(11);
        DECLARE vplaca float;
        DECLARE kursor CURSOR FOR SELECT pesel, pensja FROM pracownicy where zawod
like profesja;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
        drop table if exists wyplaty;
        CREATE TEMPORARY TABLE wyplaty (pesel char(11), info char(9));
        OPEN kursor;
        start transaction;
            FETCH kursor INTO vpesel,vplaca;
            WHILE done=0 DO
                set budzet=budzet-vplaca;
                if budzet<0 then
                    set done=1;
                end if;
                insert into wyplaty values
(concat('********',substring(vpesel,9,3)),'wyplacono');
                FETCH kursor INTO vpesel,vplaca;
            END WHILE;
            CLOSE kursor;
            if budzet<0 then
```

```
                rollback;
            end if;
        select * from wyplaty;
    end $$
    delimiter ;
```

7. Wykorzystując konstrukcję PREPARE statement napisz procedurę która przyjmuje nazwę kolumny ze zbioru{wzrost,waga,pensja}oraz nazwę zawodu z zadania 4, a następnie zwraca sumę wartości odpowiedniej kolumny, dla pracowników wybranego zawodu zapewniając 0.05–prywatność różnicową.

```
    delimiter $$
    create procedure suma (in kol varchar(30),in zaw varchar(30))
    begin
        set @r=abs(rand()*20-10);
        set @zaw=zaw;
        set @zapytanie=concat('select sum(',kol,')+(0.05/(2*(select max(',kol,')-min(',kol,') from ludzie a join pracownicy b on a.pesel=b.pesel where zawod like ?)))*EXP(-@r*0.05/(select max(',kol,')-min(',kol,') from ludzie a join pracownicy b on a.pesel=b.pesel where zawod like ?)) into @suma from ludzie a join pracownicy b on a.pesel=b.pesel where zawod like ?;');
        prepare statement from @zapytanie;
        execute statement using @zaw,@zaw,@zaw;
        deallocate prepare statement;
        select @suma as wynik;
    end $$
    delimiter ;
```

8. Napisz trigger,który będzie prowadził logi wszystkich zmian pensji. Tabela logów powinna znajdować się w osobnej bazie danych i zawierać informacje o starych wartościach, nowych wartościach, czasie zmiany i użytkowniku, który ją wykonał.

```
    delimiter $$
    create trigger logi after update on pracownicy
    for each row
    begin
        if old.pensja<>new.pensja then
```

```
                    insert into logi.logi values
(old.pesel,old.zawod,old.pensja,new.pensja,concat(current_date(),' ',current_time()),current_user);
                end if;
        end $$
        delimiter ;
```