Laboratorijska vežba br. 5.

MySQL Store Procedure

1. Cilj vežbe

- Uskladištene procedure (Stored Procedures): Naučiti kreiranje programabilnih objekata unutar baze podataka za enkapsulaciju složene poslovne logike i automatizaciju upita.
- Parametrizacija i kontrola toka: Savladati rad sa ulazno-izlaznim parametrima (IN, OUT, INOUT) i strukturama za donošenje odluka i ponavljanje radnji (IF, CASE, LOOP, REPEAT, WHILE).
- Efikasnost i sigurnost: Optimizacija performansi kroz smanjenje mrežnog saobraćaja između aplikacije i servera, kao i zaštita podataka putem kontrolisanog pristupa procedurama umesto direktno tabelama.

2. Neophodni preduslovi

- Baza: Instalirana Sakila baza podataka (tabele: film, customer, staff, inventory, rental).
- Sintaksa: Poznavanje naprednih SELECT upita, agregatnih funkcija (SUM, COUNT, MIN, MAX) i razumevanje koncepta promenljivih u SQL-u.
- Alati: Osnovno korišćenje MySQL klijenta koji podržava promenu delimitera (kao što su MySQL Workbench ili dBeaver).

3. Upiti

Napisati SQL naredbe za sledeće upite:

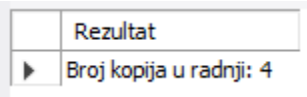
1. Kreirati uskladištenu proceduru pod nazivom GetBrojKopija koja omogućava menadžerima prodavnica brzu proveru dostupnosti određenog naslova.

   Tehnički zahtevi:

   a. Procedura treba da prima dva ulazna parametra: ID filma i ID prodavnice.
   b. Unutar procedure definisati lokalnu promenljivu koja će privremeno čuvati rezultat prebrojavanja.

c. Koristiti mehanizam direktnog dodeljivanja vrednosti iz SELECT upita u tu lokalnu promenljivu.
d. Kao krajnji rezultat, procedura treba da ispiše formatiranu poruku (npr. 'Broj kopija u radnji: X') koristeći funkciju za spajanje stringova.
e. Demonstrirati poziv procedure za film sa ID-jem 1 u prodavnici broj 1.

Očekivan rezultat – slika 1:



Slika 1.

Rešenje:

```
DELIMITER //
CREATE PROCEDURE GetBrojKopija(IN p_film_id INT, IN p_store_id INT)
BEGIN
   DECLARE v_ukupno INT;

   -- Koristimo INTO da smestimo rezultat prebrojavanja u lokalnu promenljivu
   SELECT COUNT(*) INTO v_ukupno
   FROM inventory
   WHERE film_id = p_film_id AND store_id = p_store_id;

   SELECT CONCAT('Broj kopija u radnji: ', v_ukupno) AS Rezultat;
END //
DELIMITER ;

-- POZIV:
CALL GetBrojKopija(1, 1); -- Za film 'ACADEMY DINOSAUR' u radnji 1
```

2. Potrebno je kreirati proceduru ProveriAktivnostKupca koja na osnovu ID-ja korisnika proverava da li je on trenutno aktivan u sistemu Sakila baze.

Tehnički zahtevi:

a. Procedura treba da ima jedan ulazni parametar (ID kupca) i jedan izlazni parametar (status) koji će vratiti tekstualnu vrednost aplikaciji.

b. Unutar procedure deklarisati lokalnu promenljivu koja će privremeno čuvati numeričku vrednost iz kolone active (0 ili 1).
c. Koristeći strukturu za grananje, ispitati vrednost te lokalne promenljive:
d. Ako je vrednost 1, u izlazni parametar upisati tekst 'AKTIVAN'.
e. U suprotnom, upisati tekst 'NEAKTIVAN'.
f. Demonstrirati poziv procedure koristeći korisničku promenljivu (sa @) za prihvat rezultata, a zatim tu promenljivu prikazati posebnim upitom.

Očekivan rezultat – slika 2:

	@trenutni_status
▶	AKTIVAN

Slika 2.

Rešenje:

```
DELIMITER //
CREATE PROCEDURE ProveriAktivnostKupca(IN p_customer_id INT, OUT
p_status VARCHAR(20))
BEGIN
   DECLARE v_aktivan TINYINT;

   SELECT active INTO v_aktivan FROM customer WHERE customer_id =
p_customer_id;

   IF v_aktivan = 1 THEN
      SET p_status = 'AKTIVAN';
   ELSE
      SET p_status = 'NEAKTIVAN';
   END IF;
END //
DELIMITER ;

-- POZIV:
CALL ProveriAktivnostKupca(1, @trenutni_status);
SELECT @trenutni_status;
```

3. Potrebno je kreirati proceduru PrimenPopust koja vrši modifikaciju cene proizvoda direktno u prosleđenoj promenljivoj, zavisno od kategorije kojoj proizvod pripada.

   Tehnički zahtevi:
   a. Procedura treba da koristi jedinstveni parametar koji će istovremeno služiti za prijem trenutne cene i za vraćanje izmenjene (umanjene) cene.
   b. Dodati drugi parametar koji prima naziv kategorije kao tekst.
   c. Koristeći CASE strukturu, ispitati vrednost kategorije:
   d. Ako je 'New', umanjiti cenu za 5%.
   e. Ako je 'Old', umanjiti cenu za 20%.
   f. Ako je 'Classic', umanjiti cenu za 50%.
   g. Za sve ostale kategorije, cenu ostaviti nepromenjenom.

   Demonstrirati poziv procedure:
   h. Prvo definisati sesijsku promenljivu sa početnom cenom od 4.99.
   i. Pozvati proceduru za kategoriju 'Old'.
   j. Prikazati izmenjenu vrednost te iste promenljive.

   Očekivan rezultat – slika 3:

	Cena sa popustom
▶	3.99

Slika 3.

Rešenje:

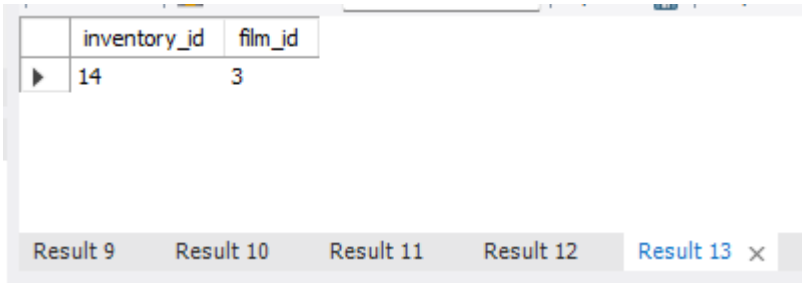
```
DELIMITER //
CREATE PROCEDURE PrimenPopust(INOUT p_cena DECIMAL(4,2), IN
p_kategorija VARCHAR(20))
BEGIN
   CASE p_kategorija
      WHEN 'New' THEN SET p_cena = p_cena * 0.95; -- 5% popusta
      WHEN 'Old' THEN SET p_cena = p_cena * 0.80; -- 20% popusta
      WHEN 'Classic' THEN SET p_cena = p_cena * 0.50; -- 50% popusta
      ELSE SET p_cena = p_cena;
   END CASE;
END //
DELIMITER ;
```

```
-- POZIV:
SET @cena_filma = 4.99;
CALL PrimenPopust(@cena_filma, 'Old');
SELECT @cena_filma AS 'Cena sa popustom';
```

4. Potrebno je kreirati proceduru ProveriOpsegInventara koja omogućava sekvencijalni pregled određenog broja stavki iz inventara, počevši od zadatog ID-ja.

   Tehnički zahtevi:
   a. Procedura treba da prima dva ulazna parametra: početni ID (odakle kreće provera) i broj stavki (koliko koraka petlja treba da izvrši).
   b. Unutar procedure deklarisati brojač (npr. v_counter) i postaviti mu početnu vrednost na 0.
   c. Koristiti WHILE petlju koja će se izvršavati sve dok je brojač manji od traženog broja provera.
   d. Unutar svakog kruga petlje:
   e. Prikazati podatke o filmu za trenutni ID.
   f. Inkrementirati (uvećati za 1) i trenutni ID i brojač, kako bi petlja prešla na sledeći korak.
   g. Pozvati proceduru za početni ID 10 i zatražiti ispis za narednih 5 stavki.

   Očekivan rezultat – slika 4:



Slika 4.

```
Rešenje:
DELIMITER //
CREATE PROCEDURE ProveriOpsegInventara(IN p_start_id INT, IN
p_broj_provera INT)
BEGIN
```

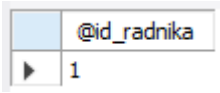
```
    DECLARE v_counter INT DEFAULT 0;
    DECLARE v_trenutni_id INT;

    SET v_trenutni_id = p_start_id;

    WHILE v_counter < p_broj_provera DO
        SELECT inventory_id, film_id FROM inventory WHERE inventory_id =
v_trenutni_id;
        SET v_trenutni_id = v_trenutni_id + 1;
        SET v_counter = v_counter + 1;
    END WHILE;
END //
DELIMITER ;

-- POZIV:
CALL ProveriOpsegInventara(10, 5); -- Prikazuje podatke za inventory_id od 10
do 14
```

5. Kreirati uskladištenu proceduru PronadjiSlobodnogRadnika koja pronalazi prvog dostupnog radnika u određenoj prodavnici čiji broj obrađenih iznajmljivanja ne prelazi zadatu granicu.

   Tehnički zahtevi:

   a. Ulaz/Izlaz: Procedura prima ID prodavnice (IN), a rezultat (ID radnika) vraća preko izlaznog parametra (OUT).
   b. Priprema podataka: Pomoću SELECT ... INTO odrediti raspon ID-jeva zaposlenih (minimalni i maksimalni) za tu prodavnicu.
   c. Imenovana petlja (Labela): Koristiti REPEAT petlju kojoj je dodeljena jedinstvena oznaka (labela) (npr. moj_ciklus:).
   d. Uslovni prekid (Rani izlaz): Unutar petlje proveriti broj iznajmljivanja za trenutnog radnika. Ako je broj manji od 10.000, upotrebiti naredbu LEAVE uz navođenje naziva labele kako bi se petlja odmah prekinula.
   e. Iteracija: Ukoliko uslov nije ispunjen, inkrementirati ID radnika i nastaviti pretragu dok se ne prođu svi zaposleni u prodavnici (UNTIL).
   f. Poziv: Testirati proceduru za prodavnicu 1 i prikazati dobijeni ID.

   Očekivan rezultat – slika 5:



Slika 5.

Rešenje:

```
DELIMITER //
CREATE PROCEDURE PronadjiSlobodnogRadnika(IN p_store_id INT, OUT
p_staff_id INT)
BEGIN
   DECLARE v_max_id INT;
   DECLARE v_min_id INT;
   DECLARE v_broj_iznajmljivanja INT;

   SELECT MIN(staff_id), MAX(staff_id) INTO v_min_id, v_max_id
   FROM staff WHERE store_id = p_store_id;

   SET p_staff_id = v_min_id;

   -- DODAJEMO LABELU 'moj_ciklus'
   moj_ciklus: REPEAT
      SELECT COUNT(*) INTO v_broj_iznajmljivanja
      FROM rental WHERE staff_id = p_staff_id;

      IF v_broj_iznajmljivanja < 10000 THEN
         LEAVE moj_ciklus; -- SADA MYSQL ZNA ŠTA DA NAPUSTI
      END IF;

      SET p_staff_id = p_staff_id + 1;
   UNTIL p_staff_id > v_max_id
   END REPEAT moj_ciklus; -- ZATVARAMO SA LABELOM
END //
DELIMITER ;

CALL PronadjiSlobodnogRadnika(1, @id_radnika);
SELECT @id_radnika;
```