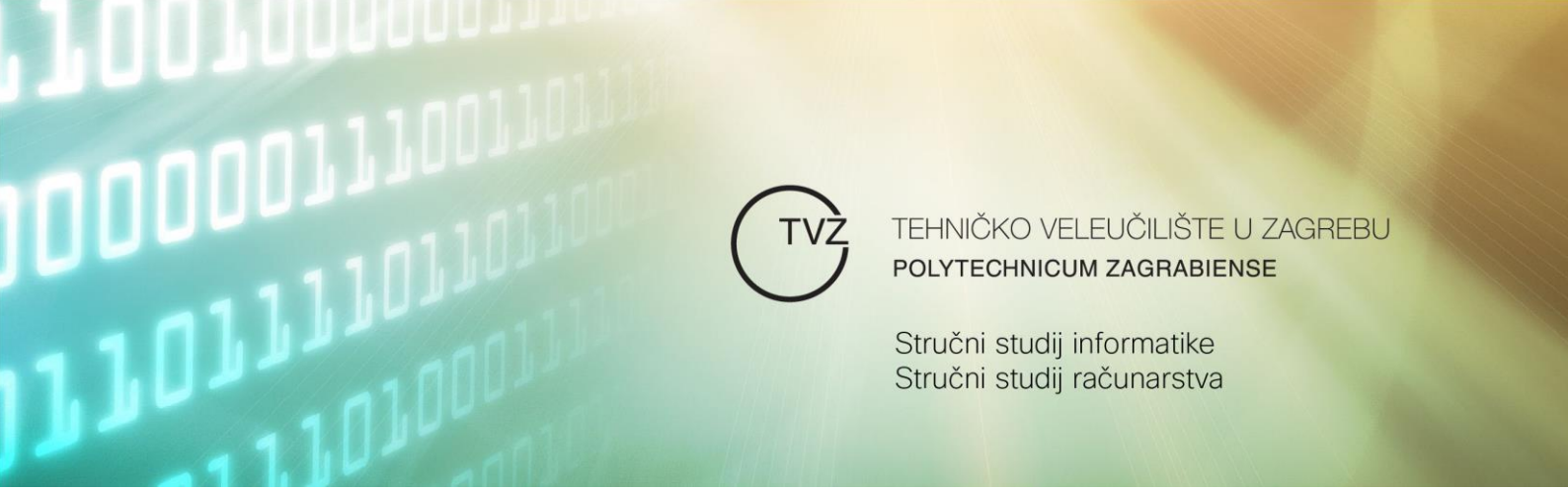

TEHNIČKO VELEUČILIŠTE U ZAGREBU
POLYTECHNICUM ZAGRABIENSE

Stručni studij informatike
Stručni studij računarstva

Napredne baze podataka

Zadaci za vježbu

Priprema za prvi međuispit

Akademska godina: 2013/14

1. Napisati transakciju koja će kvaru s nazivom kvara *Uštimavanje pokretnog krova* promijeniti vrijednost *satiKvara* na 1. Potrebno je opozvati promjene koje je transakcija učinila.

```
SET AUTOCOMMIT=0;
BEGIN WORK;
     UPDATE kvar SET satiKvar=1
          WHERE nazivKvar='Uštimavanje pokretnog krova';
ROLLBACK WORK;
SET AUTOCOMMIT=1;
```

2. Napisati transakciju koja će kvaru s nazivom kvara 'Uštimavanje pokretnog krova' promijeniti vrijednost satiKvara na 3, te nakon toga obrisati sve kvarove čija je vrijednost atributa *satiKvar* veća od 3. Potrebno je opozvati naredbu brisanja.

```
SET AUTOCOMMIT=0;
BEGIN WORK;
     UPDATE kvar SET satiKvar=3
     WHERE nazivKvar='Uštimavanje pokretnog krova';
     SAVEPOINT prijeBrisanja;
     DELETE FROM kvar WHERE satiKvar>3;
     ROLLBACK TO SAVEPOINT prijeBrisanja;
COMMIT;
SET AUTOCOMMIT=1;
```

3. Napisati transakciju koja će unijeti novi zapis u relaciju i promijeniti podatke: u tablicu *nalog* unijeti podatke 1152, 31, 334, 2011-11-1, 2, 7 te tome nalogu naknadno promijeniti *prioritetNalog* na 1. Potrebno je potvrditi sve promjene.

```
SET autocommit=0;
BEGIN WORK;
     INSERT INTO nalog VALUES(1147,31,334,20111101,2,7);
     UPDATE nalog SET prioritetNalog=1
          WHERE (sifKlijent=1147 AND sifKvar=31);
COMMIT WORK;
SET autocommit=1
```

4. Napisati transakciju koja će radniku sa šifrom 277 smanjiti koeficijent plaće za 0.5, a radniku sa šifrom 313 povećati za isti iznos. Potvrditi da se promjena stvarno nije dogodila.

```
SET autocommit=0;
BEGIN WORK;
     UPDATE radnik SET koefPlaca=koefPlaca-0.5 WHERE sifRadnik=277;
```

```
     UPDATE radnik SET koefPlaca=koefPlaca+0.5 WHERE sifRadnik=313;
ROLLBACK;
SET autocommit=1;
```

5. Napisati proceduru koja će za kvar sa zadanom šifrom, preko parametara vratiti naziv kvara. Potrebno je napisati primjer poziva procedure te ispisa rezultata.

```
DROP PROCEDURE IF EXISTS vrati_naziv;
DELIMITER //
CREATE PROCEDURE vrati_naziv(IN sif INT, OUT naziv VARCHAR(50))
BEGIN
     SELECT nazivKvar INTO naziv FROM kvar WHERE sifKvar=sif;
END
//
DELIMITER ;

CALL vrati_naziv(14, @a);
SELECT @a;
```

6. Napisati proceduru koja za zadani naziv županije, preko parametra vraća iznos najmanje i najveće plaće radnika koji živi u toj županiji. Potrebno je napisati primjer poziva procedure te ispisa rezultata.

```
DELIMITER ??
DROP PROCEDURE IF EXISTS iznosiPlace ??
CREATE PROCEDURE iznosiPlace(IN zup VARCHAR(50),
     OUT minimum INT, OUT maksimum INT)
BEGIN
     SELECT MAX(koefplaca*iznosOsnovice),
     MIN(koefplaca*iznosOsnovice) INTO maksimum , minimum
          FROM radnik JOIN mjesto ON pbrStan=pbrMjesto
          NATURAL JOIN zupanija
          WHERE nazivZupanija=zup;
END ??
DELIMITER ;

CALL iznosiPlace('Grad Zagreb',@a,@b);
SELECT @a AS najmanja, @b AS najveca;
```

7. Napisati proceduru koja zadanom kvaru povećava vrijednost atributa *satiKvar* za zadanu vrijednost. Procedura prima dva ulazna parametra: naziv kvara i broj sati (za koji će se uvećati vrijednost atributa *satiKvar*). Potrebno je napisati primjer poziva procedure.

```
DELIMITER //
CREATE PROCEDURE uvecajSate (IN odj VARCHAR(50), IN sati INT)
BEGIN
     UPDATE kvar SET satikvar=satikvar+sati WHERE nazivkvar=odj;
END //
DELIMITER ;

CALL uvecajSate ('Popravak felgi', 2);
```

8. Napisati proceduru koja će primiti šifru kvara te preko iste varijable za zadani kvar vratiti vrijednost atributa *satiKvar*. Potrebno je napisati primjer poziva procedure te ispisa rezultata.

```
DROP PROCEDURE dohvatiSatiKvar;
DELIMITER //
CREATE PROCEDURE dohvatiSatiKvar (INOUT k INT)
BEGIN
     SELECT satiKvar INTO k FROM kvar WHERE kvar.sifkvar=k;
END //
DELIMITER ;

SET @kv=7;
CALL dohvatiSatiKvar (@kv);
SELECT @kv;
```

9. Napisati proceduru koja će ispisati sve kvarove čija je vrijednost atributa *satiKvar* veća od prosječne vrijednosti *satServisa* iz tablice rezervacija na zadani dan u tjednu. Procedura preko parametra prima oznaku dana u tjednu. Potrebno je napisati primjer poziva procedure.

```
DROP PROCEDURE IF EXISTS kvarUDanu;
DELIMITER //
CREATE PROCEDURE kvarUDanu (IN dan VARCHAR(50))
BEGIN
     SELECT * FROM kvar WHERE satiKvar>
          (SELECT AVG(satServis) FROM rezervacija
               WHERE datVrstaDan=dan);
END //
DELIMITER ;

CALL kvarUDanu('PE');
```

10. Napisati proceduru koja će obrisati sve radnike u zadanom odjelu. Odjel je potrebno zadati preko njegovog imena (*odjel.nazivOdjel*). Potrebno je napisati primjer poziva procedure te ispisa rezultata.

```
DROP PROCEDURE brisiRadnikeUOdjelu;
DELIMITER //
CREATE PROCEDURE brisiRadnikeUOdjelu (IN odj VARCHAR(50))
BEGIN
      DELETE FROM radnik WHERE radnik.sifodjel=
            (SELECT sifOdjel  FROM odjel WHERE nazivodjel=odj);
END //
DELIMITER ;

CALL brisiRadnikeUOdjelu('Limarija');
```

11. Modificirajte tablicu *zupanija* tako da dodate novi atribut *brojMjesta* tipa INT. Napišite proceduru koja prima šifru županije te popunjava atribut *brojMjesta* podatkom koliko se mjesta nalazi u toj županiji. Potrebno je napisati primjer poziva procedure.

```
ALTER TABLE zupanija ADD brojMjesta INT(11);
DROP PROCEDURE brojiMjesta;
DELIMITER //
CREATE PROCEDURE brojiMjesta (IN zup INT)
BEGIN
      DECLARE br INT DEFAULT NULL;
      SELECT COUNT(*) INTO br FROM mjesto WHERE sifzupanija=zup;
      UPDATE zupanija SET brojMjesta=br WHERE sifZupanija=zup;

END //
DELIMITER ;

CALL brojiMjesta (1);
```

12. Modificirajte tablicu *nalog* tako da dodate novi atribut *razlikaNalog* tipa INT. Napisati proceduru koja prima šifru klijenta, šifru kvara i datum primitka naloga te u atribut *razlikaNalog* upisuje kolika je razlika između predviđenog trajanja za popravak kvara (*satiKvar* iz tablice *kvar*) i ostvarenih sati rada za popravak kvara po tom nalogu (*ostvareniSatiRada*). Potrebno je napisati primjer poziva procedure te ispisa rezultata.

```
ALTER TABLE nalog ADD razlikaNalog INT(11);
DROP PROCEDURE IF EXISTS racunajRazliku;
DELIMITER //
CREATE PROCEDURE racunajRazliku (IN zadaniKlijent INT,
     IN zadaniKvar INT, IN zadaniDatum DATE)
BEGIN
     DECLARE predvidenoTrajanje, stvarniSati, razlika INT;
     SELECT satiKvar INTO predvidenoTrajanje FROM kvar
          WHERE sifKvar=zadaniKvar;
     SELECT OstvareniSatiRada INTO stvarniSati FROM nalog
          WHERE sifKlijent=zadaniKlijent AND sifKvar=zadaniKvar
          AND datPrimitkaNalog=zadaniDatum;
     SET razlika=predvidenoTrajanje-stvarniSati;
     UPDATE nalog SET razlikaNalog=razlika
          WHERE sifKlijent=zadaniKlijent AND sifKvar=zadaniKvar
          AND datPrimitkaNalog=zadaniDatum;
END //
DELIMITER ;

CALL racunajRazliku (1167,13,20050415);
CALL racunajRazliku (1464,2,20050809);
```

13. Napisati funkciju koja vraća broj klijenata na čijim je nalozima radio radnik (šifra radnika je ulazni parametar). Potrebno je napisati primjer poziva funkcije.

```
DROP FUNCTION IF EXISTS brojiKlijente;
DELIMITER //
CREATE FUNCTION brojiKlijente(zadaniRadnik INT) RETURNS INT
DETERMINISTIC
BEGIN
     DECLARE br INT;
     SELECT COUNT(DISTINCT(sifKlijent)) INTO br FROM nalog
          WHERE sifRadnik=zadaniRadnik;
     RETURN br;
END
//
DELIMITER ;

SELECT brojiKlijente(122);
```

14. Napisati funkciju koja za zadanog radnika vraća koliko je klijenata registriralo vozilo u županiji u kojoj radi zadani radnik. Šifru radnika funkcija mora primiti preko globalne varijable. Potrebno je napisati primjer poziva funkcije.

```
DROP FUNCTION IF EXISTS brojiReg;
DELIMITER //
CREATE FUNCTION brojiReg() RETURNS INT
DETERMINISTIC
BEGIN
      DECLARE vrati, zadanaZupanija INT;
      SELECT sifZupanija INTO zadanaZupanija
            FROM zupanija NATURAL JOIN
            mjesto JOIN
            radnik ON pbrStan=pbrMjesto
            WHERE sifRadnik=@zadaniradnik;
      SELECT COUNT(*) INTO vrati
            FROM klijent JOIN
            mjesto ON pbrReg=pbrMjesto NATURAL JOIN zupanija
            WHERE sifZupanija=zadanaZupanija;
      RETURN vrati;
END;
//
DELIMITER ;

SET @zadaniradnik=122;
SELECT brojiReg();
```

15. Napisati funkciju koja generira jedinstveni identifikator klijenta. Funkcija prima šifru klijenta i vraća identifikator u obliku *'iprezime123'* pri čemu je:

    i – prvo slovo imena

    prezime – prezime klijenta

    123 – posljednje tri brojke iz godine rođenja klijenta

```
DROP FUNCTION IF EXISTS napraviIdentifikator;
DELIMITER //
CREATE FUNCTION napraviIdentifikator(sif INT) RETURNS VARCHAR(50)
DETERMINISTIC
BEGIN
      DECLARE id VARCHAR(50) DEFAULT NULL;
      SELECT CONCAT(LCASE(LEFT(imeklijent,1)), LCASE(prezimeklijent),
      SUBSTRING(jmbgKlijent,5,3)) INTO id
            FROM klijent WHERE sif=sifklijent;
      RETURN id;
END;
//
DELIMITER ;

SELECT napraviIdentifikator(1147);
```

16. Napisati funkciju koja broji koliko je naloga zaprimljeno u posljednjih mjesec dana. Ako nije zaprimljen ni jedan nalog, funkcija mora vratiti -1 (inače vraća broj naloga).

```
DELIMITER //
DROP FUNCTION IF EXISTS brojiNaloge //
CREATE FUNCTION brojiNaloge() RETURNS INT
DETERMINISTIC
BEGIN
      DECLARE vrati INT DEFAULT NULL;
      SELECT COUNT(*) INTO vrati FROM nalog
            WHERE (MONTH(datPrimitkaNalog)=MONTH(CURDATE())-1 AND
                  DAY(datPrimitkaNalog)>=DAY(CURDATE()) AND
                  YEAR(datPrimitkaNalog)=YEAR(CURDATE()))
            OR (MONTH(datPrimitkaNalog)=MONTH(CURDATE()) AND
                  DAY(datPrimitkaNalog)<=DAY(CURDATE()) AND
                  YEAR(datPrimitkaNalog)=YEAR(CURDATE()));
      IF vrati=0 THEN SET vrati=-1;
      END IF;
      RETURN vrati;
END;
//
DELIMITER ;
SELECT brojiNaloge();
```

17. Napisati funkciju za unos novoga kvara u tablicu *kvar*. Ako se unosi kvar sa postojećom šifrom, potrebno je novom kvaru pridijeliti novu šifru i ispisati odgovarajuću poruku: *Unijeli ste podatke o novom kvaru. Automatski mu je pridijeljena nova šifra novasifra*. Ako se unosi kvar s korektnom šifrom, potrebno je vratiti poruku: *Unijeli ste podatke o kvaru i dodijelili mu šifru novasifra*. Potrebno je napisati primjer poziva funkcije i ispisa rezultata.

```
DELIMITER //
DROP FUNCTION IF EXISTS unesiKvar //
CREATE FUNCTION unesiKvar(sif INT, naz VARCHAR(50), odj INT, brRad INT,
sati INT) RETURNS TEXT
DETERMINISTIC
BEGIN
      DECLARE vrati TEXT DEFAULT NULL;
      DECLARE br, maks INT DEFAULT NULL;
      SELECT COUNT(*) INTO br FROM kvar
            WHERE sifKvar=sif;
      IF br=0 THEN
            INSERT INTO kvar VALUES (sif, naz, odj, brRad, sati);
            SET vrati=CONCAT('Unijeli ste podatke o kvaru i dodijelili
                  mu šifru ',sif,'.');
      ELSE
            SELECT MAX(sifKvar) INTO maks FROM kvar;
            INSERT INTO kvar VALUES ((maks+1), naz, odj, brRad, sati);
            SET vrati=CONCAT('Unijeli ste podatke o novom kvaru. Automatski
                  mu je pridijeljena nova šifra ',maks+1,'.');
```

```
	END IF;
	RETURN vrati;
END;
//
DELIMITER ;
SELECT unesiKvar(38334,'Popravak auto-alarma',9,1,3 );
```

18. Napisati funkciju koja će kapitalizirati prvo slovo svake riječi u danom tekstu. Pretpostavite da riječ počinje nakon nekog od sljedećih znakova: ' ', '&', '"', '_', '?', ';', ':', '!', ',', '-', '/', '(', '.' Potrebno je napisati primjer poziva funkcije i ispisa rezultata.
(vidi http://dev.mysql.com/doc/refman/5.0/en/string-functions.html)

```
DELIMITER $$
DROP FUNCTION IF EXISTS kapitaliziraj $$
CREATE FUNCTION kapitaliziraj(tekst VARCHAR(1024))
	RETURNS VARCHAR(1024)
	DETERMINISTIC
BEGIN
	DECLARE i INT DEFAULT 1;
	DECLARE trenutni, prethodni CHAR(1);
	DECLARE izlaz VARCHAR(1024) DEFAULT tekst;
	WHILE i <= CHAR_LENGTH(tekst) DO
		SET trenutni = SUBSTRING(tekst, i, 1);
		CASE i
			WHEN 1 THEN SET prethodni=' ';
			ELSE SET prethodni=SUBSTRING(tekst, i - 1, 1) ;
		END CASE;
		/*ili SET prethodni = CASE
			WHEN i = 1 THEN ' '
			ELSE SUBSTRING(tekst, i - 1, 1)
			END; */
		IF prethodni IN (' ', '&', '''', '_', '?', ';', ':', '!', ',',
'-', '/', '(', '.')
			THEN SET izlaz = INSERT(izlaz, i, 1, UPPER(trenutni));
		END IF;
		SET i = i + 1;
	END WHILE;
	SET tekst = izlaz;
	RETURN izlaz;
END$$
DELIMITER ;

SET @ulaz = 'Ovo je zadatak za kapitaliziranje riječi';
CALL kapitaliziraj(@ulaz);
SELECT @ulaz;


SELECT kapitaliziraj('Ovo je neki tekst koji treba obraditi');
```

19. Napisati funkciju koja za ulazni parametar prima ulaznu varijablu *pbr* tipa integer. Funkcija nad tablicom *radnik* umanjuje vrijednost atributa *KoefPlaca* za 1 za sve zapise kod kojih je *pbrStan* jednak ulaznoj varijabli (*pbr*). Funkcija vraća broj 1. Zadatak je potrebno riješiti pomoću kursora.

```
DELIMITER //
DROP FUNCTION IF EXISTS smanjiKoef //
CREATE FUNCTION smanjiKoef(pbr INT) RETURNS INT
DETERMINISTIC
BEGIN
	DECLARE sif INT DEFAULT NULL;
	DECLARE koef DECIMAL(6,2) DEFAULT NULL;
	DECLARE kraj INT DEFAULT 0;
	DECLARE kursor CURSOR FOR
		SELECT koefPlaca, sifRadnik FROM radnik
		WHERE pbrStan=pbr;
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=1;
	OPEN kursor;
	petlja:LOOP FETCH kursor INTO koef, sif;
		IF kraj=1 THEN LEAVE petlja;
		END IF;
		SET koef=koef-1;
		UPDATE radnik SET koefPlaca=koef WHERE sifRadnik=sif;
	END LOOP;
	CLOSE kursor;
	RETURN 1;
END; //
DELIMITER ;

SELECT smanjiKoef(21000);
```

20. Napisati proceduru koja će sve radnike promaknuti u viši odjel (odjel veći za 1 od trenutnog odjela). Procedura vraća broj promaknutih radnika. Obavezno je koristiti kursore.

```
DELIMITER //
DROP PROCEDURE IF EXISTS promakni1 //
CREATE PROCEDURE promakni1()
	BEGIN
		DECLARE n INT DEFAULT 0;
		DECLARE kraj BOOL DEFAULT FALSE;
		DECLARE sif, odj INT DEFAULT NULL;
		DECLARE kursor CURSOR FOR
			SELECT sifradnik, sifOdjel FROM radnik;
		DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=TRUE;

		OPEN kursor;
		vrti:LOOP
			FETCH kursor INTO sif, odj;
			IF kraj=TRUE THEN LEAVE vrti;
```

```
                  END IF;
                  UPDATE radnik SET sifOdjel=odj+1
                        WHERE radnik.sifradnik=sif;
                  SET n =n +1;
            END LOOP;
            CLOSE kursor;
            SELECT n ;
      END;
      //
DELIMITER ;

CALL promakni1();
```

21. Modificirati proceduru iz prethodnog zadatka na način da promakne samo one radnike koji su iz zadanog mjesta (procedura prima naziv mjesta).

```
DELIMITER //
DROP PROCEDURE IF EXISTS promakni2 //
CREATE PROCEDURE promakni2(IN zadanoMjesto VARCHAR(50))
      BEGIN
            DECLARE n INT DEFAULT 0;
            DECLARE kraj BOOL DEFAULT FALSE;
            DECLARE sif, odj INT DEFAULT NULL;
            DECLARE kursor CURSOR FOR
                  SELECT sifradnik, sifOdjel
                  FROM radnik JOIN mjesto ON pbrStan=pbrMjesto
                        WHERE nazivMjesto=zadanoMjesto;
            DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=TRUE;

            OPEN kursor;
            vrti:LOOP
                  FETCH kursor INTO sif, odj;
                  IF kraj=TRUE THEN LEAVE vrti;
                  END IF;
                  UPDATE radnik SET sifOdjel=odj+1
                        WHERE radnik.sifradnik=sif;
                  SET n=n+1;
            END LOOP;
            CLOSE kursor;
            SELECT n ;
      END;
      //
DELIMITER ;

CALL promakni2('zagreb');
```

22. Modificirati proceduru iz prethodnog zadatka na način da ako se radnik promiče u nepostojeći odjel (šifra odjela veća od maksimalne moguće), onda je potrebno opozvati promaknuće (rollback) a inače potvrditi promaknuće(commit).

```
DELIMITER //
DROP PROCEDURE IF EXISTS promakni3 //
CREATE PROCEDURE promakni3(IN zadanoMjesto VARCHAR(50))
      BEGIN
            DECLARE n, maxOdjel INT DEFAULT 0;
            DECLARE kraj BOOL DEFAULT FALSE;
            DECLARE sif, odj INT DEFAULT NULL;
            DECLARE kursor CURSOR FOR
                  SELECT sifradnik, sifOdjel FROM radnik
                        JOIN mjesto ON pbrStan=pbrMjesto
                        WHERE nazivMjesto=zadanoMjesto;
            DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=TRUE;
            SELECT MAX(sifOdjel) INTO maxOdjel FROM odjel;
            OPEN kursor;
            vrti:LOOP
                  FETCH kursor INTO sif, odj;
                  IF kraj=TRUE THEN LEAVE vrti;
                  END IF;
                  SET autocommit=0;
                  START TRANSACTION;
                        SET odj=odj+1;
                        UPDATE radnik SET sifOdjel=odj
                              WHERE radnik.sifradnik=sif;
                        IF odj>maxOdjel THEN ROLLBACK;
                              ELSE COMMIT;SET n=n+1;
                        END IF;
                  SET autocommit=0;
            END LOOP;
            CLOSE kursor;
            SELECT n ;
      END;
      //
DELIMITER ;
CALL promakni3('dubrovnik');
```

23. Nadograditi proceduru iz prethodnog zadatka tako da vraća podatke o promaknutim radnicima (šifru radnika) te odjel kojem radnik nakon promaknuća pripada.

```
DELIMITER //
DROP PROCEDURE IF EXISTS promakni4 //
CREATE PROCEDURE promakni4(IN zadanoMjesto VARCHAR(50))
      BEGIN
            DECLARE n, maxOdjel INT DEFAULT 0;
            DECLARE kraj BOOL DEFAULT FALSE;
            DECLARE sif, odj INT DEFAULT NULL;
```

```
            DECLARE kursor CURSOR FOR
                  SELECT sifradnik, sifOdjel FROM radnik
                        JOIN mjesto ON pbrStan=pbrMjesto
                        WHERE nazivMjesto=zadanoMjesto;
            DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=TRUE;
            DROP TEMPORARY TABLE IF EXISTS tmpRadnik;
            CREATE TEMPORARY TABLE
                  tmpRadnik(sifRad INT, noviOdjel INT);
            SELECT MAX(sifOdjel) INTO maxOdjel FROM odjel;
            OPEN kursor;
            vrti:LOOP
                  FETCH kursor INTO sif, odj;
                  IF kraj=TRUE THEN LEAVE vrti;
                  END IF;
                  SET autocommit=0;
                  START TRANSACTION;
                        SET odj=odj+1;
                        UPDATE radnik SET sifOdjel=odj
                              WHERE radnik.sifradnik=sif;
                        INSERT INTO tmpRadnik VALUES(sif,odj);
                        IF (odj>maxOdjel) THEN ROLLBACK;
                              ELSE COMMIT; SET n=n+1;
                        END IF;
                  SET autocommit=0;
            END LOOP;
            CLOSE kursor;
            SELECT * FROM tmpRadnik ;
      END;
      //
DELIMITER ;
CALL promakni4('dubrovnik');
```

24. Tablici radnik dodati novi atribut *brNaloga* inicijalno postavljen na -1. Napisati proceduru koja će u novostvoreni atribut upisati koliko je naloga radnik obradio. Potrebno je obraditi sve radnike.

```
ALTER TABLE radnik ADD brNaloga INT DEFAULT -1;
DROP PROCEDURE IF EXISTS brojiNaloge1;
DELIMITER //
CREATE PROCEDURE brojiNaloge1()
BEGIN
      DECLARE kraj BOOL DEFAULT FALSE;
      DECLARE sifra, n INT DEFAULT NULL;
      DECLARE dohvaceno, obradeno INT DEFAULT 0;
      DECLARE k CURSOR FOR SELECT sifRadnik FROM radnik;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=TRUE;

      OPEN k;
      SELECT FOUND_ROWS() INTO dohvaceno;
      vrti:LOOP
```

```
            FETCH k INTO sifra;
            IF kraj=TRUE THEN LEAVE vrti;
            END IF;
            SELECT COUNT(*) INTO n FROM nalog NATURAL JOIN radnik
                  WHERE sifRadnik=sifra;
            UPDATE radnik SET brNaloga=n WHERE sifRadnik=sifra;
            SET obradeno=obradeno+1;
      END LOOP;
      CLOSE k;
      SELECT obradeno AS obradeno_zapisa;
END
//
DELIMITER ;
CALL brojiNaloge1();
```

25. Modificirati proceduru iz prethodnog zadatka da obradi samo one radnike koji su radili na zadanom kvaru (ulazni parametar u proceduru neka bude naziv kvara).

```
DROP PROCEDURE IF EXISTS brojiNaloge2;
DELIMITER //
CREATE PROCEDURE brojiNaloge2(IN kv VARCHAR(50))
BEGIN
      DECLARE kraj BOOL DEFAULT FALSE;
      DECLARE sifra, n INT DEFAULT NULL;
      DECLARE dohvaceno, obradeno INT DEFAULT 0;
      DECLARE k CURSOR FOR
            SELECT radnik.sifRadnik FROM radnik
            JOIN nalog ON radnik.sifRadnik=nalog.sifRadnik
            JOIN kvar ON nalog.sifKvar=kvar.sifKvar WHERE nazivKvar=kv;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=TRUE;

      OPEN k;
      SELECT FOUND_ROWS() INTO dohvaceno;
      vrti:LOOP
            FETCH k INTO sifra;
            IF kraj=TRUE THEN LEAVE vrti;
            END IF;
            SELECT COUNT(*) INTO n FROM nalog NATURAL JOIN radnik
                  WHERE sifRadnik=sifra;
            UPDATE radnik SET brNaloga=n WHERE sifRadnik=sifra;
            SET obradeno=obradeno+1;
      END LOOP;
      CLOSE k;
      SELECT dohvaceno,obradeno;
END
//
DELIMITER ;
CALL brojiNaloge2('Zamjena blatobrana');
```

26. Modificirati proceduru iz prethodnog zadatka na način da se upiše vrijednost brNaloga samo ako je veća od 10.

```
DROP PROCEDURE IF EXISTS brojiNaloge3;
DELIMITER //
CREATE PROCEDURE brojiNaloge3(IN kv VARCHAR(50))
BEGIN
	DECLARE kraj BOOL DEFAULT FALSE;
	DECLARE sifra, n INT DEFAULT NULL;
	DECLARE k CURSOR FOR SELECT radnik.sifRadnik FROM radnik
		JOIN nalog ON radnik.sifRadnik=nalog.sifRadnik
		JOIN kvar ON nalog.sifKvar=kvar.sifKvar WHERE nazivKvar=kv;
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=TRUE;

	OPEN k;
	vrti:LOOP
		FETCH k INTO sifra;
		IF kraj=TRUE THEN LEAVE vrti;
		END IF;
		SET autocommit=0;
		START TRANSACTION;
			SELECT COUNT(*) INTO n FROM nalog NATURAL JOIN radnik
				WHERE sifRadnik=sifra;
			UPDATE radnik SET brNaloga=n WHERE sifRadnik=sifra;
			IF n<10 THEN ROLLBACK;
				ELSE COMMIT;
			END IF;
			SET autocommit=0;
		END LOOP;
	CLOSE k;
END
//
DELIMITER ;

CALL brojiNaloge3('Zamjena blatobrana');
```

27. Modificirati proceduru iz prethodnog zadatka na način da se ispisuju i podaci o radnicima kojima je promijenjena vrijednost atributa *brNaloga*. Neka se ispišu atributi *sifRadnik* i *brNaloga.*

```
UPDATE radnik SET brNaloga=-1;
DROP PROCEDURE IF EXISTS brojiNaloge4;
DELIMITER //
CREATE PROCEDURE brojiNaloge4(IN kv VARCHAR(50))
BEGIN
	DECLARE kraj BOOL DEFAULT FALSE;
	DECLARE sifra, n INT DEFAULT NULL;
	DECLARE dohvaceno, obradeno INT DEFAULT 0;
	DECLARE k CURSOR FOR
```

```
            SELECT radnik.sifRadnik FROM radnik
            JOIN nalog ON radnik.sifRadnik=nalog.sifRadnik
            JOIN kvar ON nalog.sifKvar=kvar.sifKvar WHERE nazivKvar=kv;
      DECLARE CONTINUE HANDLER FOR NOT FOUND SET kraj=TRUE;
      DROP TEMPORARY TABLE IF EXISTS tmpRadnik;
      CREATE TEMPORARY TABLE tmpRadnik (sRad INT, brNal INT);
      OPEN k;
      SELECT FOUND_ROWS() INTO dohvaceno;
      vrti:LOOP
            FETCH k INTO sifra;
            IF kraj=TRUE THEN LEAVE vrti;
            END IF;
            SET autocommit=0;
            START TRANSACTION;
                  SELECT COUNT(*) INTO n FROM nalog NATURAL JOIN radnik
                         WHERE sifRadnik=sifra;
                  UPDATE radnik SET brNaloga=n WHERE sifRadnik=sifra;
                  INSERT INTO tmpRadnik  VALUES(sifra,n);
                  IF n<10 THEN ROLLBACK;
                        ELSE COMMIT;
                  END IF;
                  SET autocommit=0;
      END LOOP;
      CLOSE k;

      SELECT * FROM tmpRadnik ;
END
//
DELIMITER ;
CALL brojiNaloge4('Zamjena blatobrana');
```