```
 CREATE DATABASE rent_a_bike;
DROP DATABASE rent_a_bike;
USE rent_a_bike;
CREATE TABLE korisnik (
 id INTEGER NOT NULL,
 ime VARCHAR(20) NOT NULL,
 prezime VARCHAR(20) NOT NULL,
 oib CHAR(11) NOT NULL,
 PRIMARY KEY (id)
);
CREATE TABLE zaposlenik (
 id INTEGER NOT NULL,
 ime VARCHAR(20) NOT NULL,
 prezime VARCHAR(20) NOT NULL,
 datum_zaposlenja DATETIME NOT NULL,
 PRIMARY KEY (id)
);
CREATE TABLE oprema (
 id INTEGER NOT NULL,
 naziv VARCHAR(20) NOT NULL,
 cijena NUMERIC(10,2) NOT NULL,
 PRIMARY KEY (id)
);
CREATE TABLE bicikl (
 id INTEGER NOT NULL,
 oznaka VARCHAR(10) NOT NULL,
 vrsta VARCHAR(20) NOT NULL,
 datum_kupnje DATETIME NOT NULL,
 cijena_najma NUMERIC(10,2) NOT NULL,
 PRIMARY KEY (id)
);
CREATE TABLE najam (
 id INTEGER NOT NULL,
 id_bicikl INTEGER NOT NULL,
 id_korisnik INTEGER NOT NULL,
 id_zaposlenik INTEGER NOT NULL,
 datum_pocetak DATETIME NOT NULL,
 datum_kraj DATETIME NOT NULL,
 PRIMARY KEY (id),
 FOREIGN KEY (id_bicikl) REFERENCES bicikl (id),
 FOREIGN KEY (id_korisnik) REFERENCES korisnik (id),
 FOREIGN KEY (id_zaposlenik) REFERENCES zaposlenik (id)
);
CREATE TABLE stavka_najam (
 id INTEGER NOT NULL,
 id_najam INTEGER NOT NULL,
 id_oprema INTEGER NOT NULL,
 kolicina INTEGER NOT NULL,
 PRIMARY KEY (id),
 FOREIGN KEY (id_najam) REFERENCES najam (id),
 FOREIGN KEY (id_oprema) REFERENCES oprema (id)
```

```
 FOREIGN KEY (id_oprema) REFERENCES oprema (id),
 UNIQUE (id_najam, id_oprema)
);


INSERT INTO korisnik VALUES (1, 'Marko', 'Marić', '12345678910'),
 (2, 'Tea', 'Bilić', '21234567891'),
 (3, 'Mirko', 'Marić', '32345678910');
INSERT INTO zaposlenik VALUES
 (11, 'Marina', 'Rović', STR_TO_DATE('10.11.2020.', '%d.%m.%Y.')),
 (12, 'Lea', 'Boban', STR_TO_DATE('11.11.2020.', '%d.%m.%Y.')),
 (13, 'Mauro', 'Matić', STR_TO_DATE('12.11.2020.', '%d.%m.%Y.'));

INSERT INTO oprema VALUES (21, 'Zaštitna kaciga', 30.0),
 (22, 'Štitnici za koljena', 20.0),
 (23, 'Rukavice', 20.0);
INSERT INTO bicikl VALUES
 (31, 'B1', 'road', STR_TO_DATE('10.10.2019.', '%d.%m.%Y.'), 100.00),
 (32, 'B2', 'mountain', STR_TO_DATE('20.10.2019.', '%d.%m.%Y.'), 120.0),
 (33, 'B3', 'road', STR_TO_DATE('10.11.2019.', '%d.%m.%Y.'), 90.00);
INSERT INTO najam VALUES
 (41, 31, 1, 11, STR_TO_DATE('15.11.2020.', '%d.%m.%Y.'),
STR_TO_DATE('16.11.2020.', '%d.%m.%Y.')),
 (42, 32, 2, 12, STR_TO_DATE('15.11.2020.', '%d.%m.%Y.'),
STR_TO_DATE('17.11.2020.', '%d.%m.%Y.')),
 (43, 32, 1, 12, STR_TO_DATE('18.11.2020.', '%d.%m.%Y.'),
STR_TO_DATE('20.11.2020.', '%d.%m.%Y.'));
INSERT INTO sta vka_najam VALUES (51, 41, 21, 1),
 (52, 41, 23, 2),
 (53, 42, 23, 2),
 (54, 43, 23, 2);

 #1.  Dodaj imenovano ograničenje koje će prilikom brisanja najma obrisati i povezane stavke najma.
ALTER TABLE najam
ADD CONSTRAINT najam_del FOREIGN KEY (id_bicikl) REFERENCES
bicikl (id) ON DELETE CASCADE,

ADD CONSTRAINT najam_del FOREIGN KEY (id_korisnik)
REFERENCES korisnik (id) ON DELETE CASCADE,

ADD CONSTRAINT najam_del FOREIGN KEY (id_zaposlenik)
REFERENCES zaposlenik (id) ON DELETE CASCADE;


 #2. Dodaj imenovano ograničenje koje će osigurati da vrsta bicikle bude: 'road', 'mountain' ili 'comf

 ALTER TABLE bicikl
    ADD CONSTRAINT bicikl_vrsta_chk CHECK(vrsta='road' OR
    vrsta='mountain' OR vrsta='comfort');

 #3. Dodaj imenovano ograničenje koje će osigurati da je cijena najma bicikla u rasponu <0, 1000].
ALTER TABLE bicikl
```

```
ALTER TABLE bicikl
ADD CONSTRAINT bicikl_check_cijena CHECK(cijena_najma>0 AND
cijena_najma<1000);

#4. Napiši funkciju koja će za bicikl (definiran parametrom p_id_bicikl) vratiti 'DA' ako je bicikl ba
 DELIMITER //
CREATE FUNCTION izdan_bar_jednom(p_id_bicikl INTEGER)
RETURNS VARCHAR(2)
DETERMINISTIC
BEGIN
DECLARE rezultat VARCHAR(2);
DECLARE var INTEGER;
SELECT COUNT(*) INTO var
FROM bicikl AS b
 INNER JOIN najam AS n ON n.id_bicikl = b.id
 WHERE b.id=p_id_bicikl;
IF var>0 THEN
SET rezultat='DA';
ELSE
SET rezultat='NE';
END IF;
RETURN rezultat;
END//
DELIMITER ;
SELECT izdan_bar_jednom(neki_bic_id);

#5. Napiši funkciju koja će za najam (definiran parametrom p_id_najam) vratiti 'DA' ako je najam napra
DELIMITER //
CREATE FUNCTION napravljen(p_id_najam INTEGER) RETURNS
VARCHAR(20)
DETERMINISTIC
BEGIN
DECLARE rezultat VARCHAR(2);
 DECLARE var INTEGER;

SELECT COUNT(*) INTO var
FROM bicikl AS b
 INNER JOIN najam AS n ON n.id_bicikl = b.id
 WHERE datum_pocetak = datum_pocetak -INTERVAL 1 MONTH
AND b.id=p_id_najam;
IF var>0 THEN
SET rezultat='DA';
ELSE
SET rezultat='NE';
END IF;
RETURN rezultat;
END//
DELIMITER ;
SELECT napravljen(31);

#6. Napiši funkciju koja će za bicikl (definiran parametrom p_id_bicikl) vratiti njegovu cijenu najma
DELIMITER //
```

```
DELIMITER //
CREATE FUNCTION cijena_bicikla(p_id_bicikl INTEGER, p_popust INTEGER) RETURNS INTEGER
DETERMINISTIC
BEGIN
DECLARE rezultat INTEGER;
DECLARE var INTEGER;
    SELECT cijena_najma INTO var
    FROM bicikl
    WHERE id=p_id_bicikl;

IF var < 100 THEN
SET rezultat =var*p_popust/100;
ELSE
SET rezultat = var;
END IF;
RETURN rezultat;
END//
DELIMITER ;
SELECT cijena_bicikla(33,10);

#7. Napiši funkciju koja će za zaposlenika (definiranog parametrom p_id_zaposlenik) vratiti broj najmo
#Zatim napiši upit koji će prikazati sve zaposlenike i broj najmova koje je pojedini zaposlenik naprav
DELIMITER //
CREATE FUNCTION br_najmova(p_id_zaposlenik INTEGER) RETURNS INTEGER
DETERMINISTIC
BEGIN
DECLARE rezultat INTEGER;
    SELECT COUNT(*) INTO rezultat
    FROM zaposlenik AS z
    INNER JOIN najam AS n ON z.id = n.id_zaposlenik
    WHERE z.id=p_id_zaposlenik;
RETURN rezultat;
END//
DELIMITER ;
SELECT br_najmova(12) FROM DUAL;

SELECT z.*, COUNT(n.id_zaposlenik) AS broj_najmova
FROM zaposlenik AS z
 LEFT JOIN najam AS n ON z.id = n.id_zaposlenik
 GROUP BY n.id_zaposlenik;

/*8. Npiši funkciju koja će za opremu (definiranu parametrom p_id_oprema) vratiti ukupnu
količinu u kojoj je ona izdana u najmovima (npr. oprema sa id-om 23 je ukupno izdana u
količini od 6). Zatim napiši upit koji će prikazati svu opremu i ukupnu količinu pojedine
opreme izdane u najmovima koristeći prethodno napisanu funkciju.
*/
DELIMITER //
CREATE FUNCTION ukupna_kolicina(p_id_oprema INTEGER) RETURNS INTEGER
DETERMINISTIC
BEGIN
DECLARE rezultat INTEGER;
SELECT SUM(kolicina) AS sveukupna INTO rezultat
```

```
FROM oprema AS o
 INNER JOIN stavka_najam AS sn ON o.id = sn.id_oprema
 WHERE o.id = p_id_oprema;
RETURN rezultat;
RETURN 1;
END//
DELIMITER ;
SELECT ukupna_kolicina(23) FROM DUAL;
DROP FUNCTION ukupna_kolicina;

SELECT o.*,SUM(kolicina) AS sveukupna_kolicina
FROM oprema AS o
 INNER JOIN stavka_najam AS sn ON o.id = sn.id_oprema
 GROUP BY o.id;


/*3. Napiši funkciju koja će za korisnika (definiranog parametrom p_id_korisnik) vratiti broj
najmova koje je on napravio (npr. korisnik sa id-om 1 je napravio dva najma). Zatim napiši
upit koji će prikazati sve korisnike i broj najmova koje je pojedini korisnik napravio koristeći
prethodno napisanu funkciju. */

DELIMITER //
CREATE FUNCTION broj_najmova(p_id_korisnik INTEGER)
RETURNS INTEGER
DETERMINISTIC
BEGIN
DECLARE rezultat INTEGER;
SELECT COUNT(n.id_korisnik) INTO rezultat
FROM zaposlenik AS z
 INNER JOIN najam AS n ON z.id=n.id_zaposlenik
 WHERE n.id_korisnik=p_id_korisnik;
RETURN rezultat;
END//
DELIMITER ;
SELECT broj_najmova(2) FROM DUAL;
DROP FUNCTION broj_najmova;
SELECT z.*,COUNT(n.id_korisnik) AS broj_najmova
FROM zaposlenik AS z
 LEFT JOIN najam AS n ON z.id=n.id_zaposlenik
 GROUP BY n.id_korisnik;


/*
Napiši proceduru koja će u izlaznu varijablu spremiti vrijednost 'DA' ako svi korisnici imaju
točno 11 znakova u oib-u, dok će u suprotnom (barem jedan korisnik nema oib dužine 11
znakova) u izlaznu varijablu spremiti vrijednost 'NE'.

*/
DELIMITER //
CREATE PROCEDURE izlaz_oib(OUT p_rezultat VARCHAR(2))
BEGIN
```

```
DECLARE rez INTEGER;
SELECT COUNT(*) into rez
FROM korisnik
    WHERE length(oib)=11;
    IF rez > 0 THEN
    SET p_rezultat='Da';
    ELSE
    SET p_rezultat='ne';
    END IF;
    END //
DELIMITER ;

DROP PROCEDURE izlaz_oib;

CALL izlaz_oib(@p_rezultat) ;
SELECT @p_rezultat;

/* Napiši proceduru koja će u izlaznu varijablu spremiti vrijednost 'DA' ako su svi zaposlenici
zaposleni u trenutnoj godini, dok će u suprotnom (barem jedan zaposlenik nije zaposlen u
trenutnoj godini) u izlaznu varijablu spremiti vrijednost 'NE'.*/


DELIMITER //
CREATE PROCEDURE zaposleni_now(OUT izlaz VARCHAR(2))
BEGIN
DECLARE var DATE;
SELECT datum_zaposlenja INTO var
FROM zaposlenik
 WHERE datum_zaposlenja > DATE(NOW()) - INTERVAL 1 YEAR;
IF datum_zaposlenja = var THEN
SET izlaz='DA';
ELSE
SET izlaz='NE';
END IF;
END //
DELIMITER ;
DROP PROCEDURE zaposleni_now;
CALL zaposleni_now(@izlaz);
SELECT @izlaz;



/*Napiši proceduru koja će u izlaznu varijablu spremiti broj korisnika koji imaju određeno
prezime (definirano ulaznim parametrom p_prezime), a ako takvo prezime nema niti jedan
korisnik će se u izlaznu varijablu spremiti vrijednost -1.
*/
DELIMITER //
CREATE PROCEDURE prezime_korisnika(p_prezime VARCHAR(20),OUT rezultat varchar(2))
BEGIN
DECLARE var VARCHAR(20);
    SELECT COUNT(prezime) INTO var
```

```
    FROM korisnik
    WHERE prezime =p_prezime;
IF p_prezime = var THEN
SET rezultat ='-1';
ELSE
SET rezultat=-'Da';
END IF;
END //
DELIMITER ;
DROP PROCEDURE prezime_korisnika;

CALL prezime_korisnika('Bilić',@rezultat);

SELECT @rezultat;
/*
Napiši okidač koji će osigurati da je završni datum (datum_kraj) najma veći ili jednak
početnom datumu (datum_pocetak) prilikom unosa najma. U slučaju da je završni datum
manji od početnog datuma će se završni datum postaviti na vrijednost početnog datuma + 1
dan.
*/
 DELIMITER //
CREATE TRIGGER bi_najam
BEFORE INSERT ON najam
FOR EACH ROW
BEGIN
DECLARE var VARCHAR(20);
SELECT datum_pocetak INTO var
FROM najam;

IF new.datum_kraj <= var THEN
SET new.datum_kraj = new.datum_pocetak + INTERVAL 1 DAY;
END IF;
END//
DELIMITER ;

INSERT INTO najam VALUES
(44, 32, 1, 12, STR_TO_DATE('18.11.2020.', '%d.%m.%Y.'), STR_TO_DATE('17.11.2020.', '%d.%m.%Y.'));

SELECT * FROM najam;

/*
Napiši okidač koji će osigurati da je cijena prilikom unosa opreme u rasponu <0, 500]. U
slučaju da je cijena izvan granica, potrebno ju je postaviti na najbližu graničnu vrijednost (ako
je cijena manja od 1 postaviti na 1, a ako je cijena veća od 500 postaviti na 500).
*/
DELIMITER //
CREATE TRIGGER bi_oprema
BEFORE INSERT ON oprema
FOR EACH ROW
BEGIN
IF new.cijena < 1 THEN
```

```
SET new.cijena =1;
ELSEIF new.cijena >500 THEN
SET new.cijena=500;
END IF;
END//
DELIMITER ;

/*
Napiši okidač koji će osigurati da je količina opreme prilikom izmjene stavaka najma u
rasponu <0, 10]. U slučaju da je količina izvan granica, potrebno ju je postaviti na najbližu
graničnu vrijednost (ako je količina manja od 1 postaviti na 1, a ako je količina veća od 10
postaviti na 10).*/
DELIMITER //
CREATE TRIGGER bi_stavka_najam
BEFORE INSERT ON stavka_najam
FOR EACH ROW
BEGIN
IF new.kolicina < 1 THEN
SET new.kolicina =1;
ELSEIF new.kolicina > 10 THEN
SET new.kolicina=10;
END IF;
END//
DELIMITER ;
```