# SQL DDL - Funkcije

### Evidencija račun

Potrebno je evidentirati račune izdane u trgovini. Za svaki račun potrebno je pratiti *broj_računa*, *datum_izdavanja* i *zaposlenika* koji je račun izdao, te kupca koji je platio račun. Za zaposlenika se prati: *ime*, *prezime*, *oib*, *datum_zaposlenja*, dok se za kupca prati *ime* i *prezime*. Na svakom računu se nalazi stavke koje u sebi sadrže *artikl* i *količinu*. Artikl se sastoji od *naziva* i *cijene*.

```
kupac(id, ime, prezime)
zaposlenik(id, ime, prezime, oib, datum_zaposlenja)
artikl(id, naziv, cijena)
racun(id, id_zaposlenik, id_kupac, broj, datum_izdavanja)
stavka_racun(id, id_racun, id_artikl, kolicina)
```

### Baza podataka & tablice

```
CREATE DATABASE trgovina;
USE trgovina;

CREATE TABLE kupac (
  id INTEGER NOT NULL,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE zaposlenik (
  id INTEGER NOT NULL,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL,
  oib CHAR(11) NOT NULL,
  datum_zaposlenja DATETIME NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE artikl (
  id INTEGER NOT NULL,
  naziv VARCHAR(20) NOT NULL,
  cijena NUMERIC(10,2) NOT NULL,
  PRIMARY KEY (id)
);
```

```sql
CREATE TABLE racun (
  id INTEGER NOT NULL,
  id_zaposlenik INTEGER NOT NULL,
  id_kupac INTEGER NOT NULL,
  broj VARCHAR(100) NOT NULL,
  datum_izdavanja DATETIME NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (id_zaposlenik) REFERENCES zaposlenik (id),
  FOREIGN KEY (id_kupac) REFERENCES kupac (id)
);

CREATE TABLE stavka_racun (
  id INTEGER NOT NULL,
  id_racun INTEGER NOT NULL,
  id_artikl INTEGER NOT NULL,
  kolicina INTEGER NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (id_racun) REFERENCES racun (id) ON DELETE CASCADE,
  FOREIGN KEY (id_artikl) REFERENCES artikl (id),
  UNIQUE (id_racun, id_artikl)
);
```

### Unos podataka

```sql
INSERT INTO kupac VALUES (1, 'Lea', 'Fabris'),
                         (2, 'David', 'Sirotić'),
                         (3, 'Tea', 'Bibić');

INSERT INTO zaposlenik VALUES
  (11, 'Marko', 'Marić', '123451', STR_TO_DATE('01.10.2020.', '%d.%m.%Y.')),
  (12, 'Toni', 'Milovan', '123452', STR_TO_DATE('02.10.2020.', '%d.%m.%Y.')),
  (13, 'Tea', 'Marić', '123453', STR_TO_DATE('02.10.2020.', '%d.%m.%Y.'));

INSERT INTO artikl VALUES (21, 'Puding', 5.99),
                          (22, 'Milka čokolada', 30.00),
                          (23, 'Čips', 9);

INSERT INTO racun VALUES
  (31, 11, 1, '00001', STR_TO_DATE('05.10.2020.', '%d.%m.%Y.')),
  (32, 12, 2, '00002', STR_TO_DATE('06.10.2020.', '%d.%m.%Y.')),
  (33, 12, 1, '00003', STR_TO_DATE('06.10.2020.', '%d.%m.%Y.'));

INSERT INTO stavka_racun VALUES (41, 31, 21, 2),
                                (42, 31, 22, 5),
                                (43, 32, 22, 1),
                                (44, 32, 23, 1);
```

**Definiranje funkcije:** Primjer funkcije koja uvijek vraća izraz *It works*:

```
# Definiranje funkcije
DELIMITER //
CREATE FUNCTION demo() RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
  RETURN "It works";
END//
DELIMITER ;

# Korištenje funkcije
SELECT demo() FROM DUAL;
```

**Funkcija koja zbraja dva broja:** Napiši funkciju koja će za dva broja (definirane parametrima *a* i *b*) vratiti njihov zbroj:

```
# Definiranje funkcije
DELIMITER //
CREATE FUNCTION zbroji(a INTEGER, b INTEGER) RETURNS INTEGER
DETERMINISTIC
BEGIN
  RETURN a + b;
END//
DELIMITER ;

# Primjer korištenja funkcije
SELECT zbroji(1, 3) FROM DUAL;
```

**Definiranje varijabli:** Napiši funkciju koja će za dva broja (definirane parametrima *a* i *b*) vratiti njihov zbroj:

```
DELIMITER //
CREATE FUNCTION zbroji_2(a INTEGER, b INTEGER) RETURNS INTEGER
DETERMINISTIC
BEGIN
  DECLARE rezultat INTEGER DEFAULT 0;

  SET rezultat = a + b;

  RETURN rezultat;
END//
DELIMITER ;

SELECT zbroji_2(1, 3) FROM DUAL;
```

**Spremanje rezultata izvođenja upita u varijablu:** Napiši funkciju koja će vratiti broj artikala zapisanih u bazu podataka:

```
DELIMITER //
CREATE FUNCTION broj_artikala() RETURNS INTEGER
DETERMINISTIC
BEGIN
  DECLARE broj INTEGER DEFAULT 0;

  SELECT COUNT(*) INTO broj
    FROM artikl;

  RETURN broj;
END//
DELIMITER ;

SELECT broj_artikala() FROM DUAL;
```

**Više varijabli:** Napiši funkciju koja će vratiti izraz sa informacijama o cijeni artikala (npr. *'AVG: 15, MAX: 30, MIN: 6'*):

```
DELIMITER //
CREATE FUNCTION info_o_cijeni_artikala() RETURNS VARCHAR(50)
DETERMINISTIC
BEGIN
  DECLARE avg_cijena, max_cijena, min_cijena INTEGER DEFAULT 0;
  DECLARE info VARCHAR(50);

  SELECT AVG(cijena), MAX(cijena), MIN(cijena)
         INTO avg_cijena, max_cijena, min_cijena
    FROM artikl;

  SET info = CONCAT("AVG:  ", avg_cijena, ", MAX: ", max_cijena, ", MIN: ",
min_cijena);

  RETURN info;
END//
DELIMITER ;
```

**Zadatak:** Napiši funkciju koja će za određeni artikl (definiran sa parametrom *p_id_artikl*) izračunati ukupnu prodanu količinu. Zatim napiši upit koji će prikazati sve artikle i njihovu prodanu količinu koristeći prethodno napisanu funkciju:

```
DELIMITER //
CREATE FUNCTION prodana_kolicina(p_id_artikl INTEGER) RETURNS INTEGER
DETERMINISTIC
BEGIN
  DECLARE kol INTEGER;

  SELECT SUM(kolicina) INTO kol
    FROM stavka_racun
    WHERE id_artikl = p_id_artikl;

  RETURN kol;
END//
DELIMITER ;

SELECT *, prodana_kolicina(id) AS prodana_kolicina
  FROM artikl;
```

**Zadatak:** Napiši funkciju koja će za određeni račun (definiran sa parametrom *p_id_racun*) izračunati ukupan iznos. Zatim napiši upit koji će prikazati sve račune i njihov iznos koristeći prethodno napisanu funkciju:

```
DELIMITER //
CREATE FUNCTION iznos_racuna(p_id_racun INTEGER) RETURNS DECIMAL(10, 2)
DETERMINISTIC
BEGIN
  DECLARE iznos DECIMAL(10, 2);

  SELECT SUM(kolicina * cijena) INTO iznos
    FROM stavka_racun s
    INNER JOIN artikl a ON s.id_artikl = a.id
    WHERE s.id_racun = p_id_racun;

  RETURN iznos;
END//
DELIMITER ;

SELECT *, iznos_racuna(id) AS iznos
  FROM racun;
```

**Grananja:** Napiši funkciju koja će za određeni cijenu (definiran sa parametrom *p_cijena*) vratiti vrijednost *"Jeftino"* ako je cijena manja od 10, u suprotnom će vratiti vrijednost *"Skupo"*. U slučaju da je *p_cijena* manja ili jednaka 0, vraća se vrijednost *"Došlo je do greške"*:

```
DELIMITER //
CREATE FUNCTION opis_artikla(p_cijena INTEGER) RETURNS VARCHAR(50)
DETERMINISTIC
BEGIN
  DECLARE rez VARCHAR(50);

  IF p_cijena <= 0 THEN
    SET rez = "Došlo je do greške";
  ELSEIF p_cijena < 10 THEN
    SET rez = "Jeftino";
  ELSE
    SET rez = "Skupo";
  END IF;

  RETURN rez;
END//
DELIMITER ;

SELECT *, opis_artikla(cijena)
  FROM artikl;
```

**Zadatak:** Napiši funkciju koja će za određeni artikl (definiran sa parametrom *p_id_artikl*) odrediti njegovu popularnost, odnosno nije nikada prodan će se vratiti vrijednost *"Artikl nije nikada izdan"*, ako je prodan u količini manjoj od 5 će vratiti vrijednost *"Nije popularan"*, inače će vratiti vrijednost *"Popularan"*:

```
CREATE FUNCTION popularnost_artikla(p_id_artikl INTEGER) RETURNS VARCHAR(50)
DETERMINISTIC
BEGIN
  DECLARE kol INTEGER;
  DECLARE rez VARCHAR(50);

  SELECT SUM(kolicina) INTO kol
    FROM stavka_racun
    WHERE id_artikl = p_id_artikl;

  IF kol = 0 THEN
    SET rez = "Artikl nije nikada izdan";
  ELSEIF kol < 5 THEN
    SET rez = "Nije popularan";
  ELSE
    SET rez = "Popularan";
  END IF;

  RETURN rez;
END//
DELIMITER ;

SELECT *, popularnost_artikla(id)
  FROM artikl;
```

**While petlja:** Napiši funkciju koja će za broj (definiran parametrom *p_x*) izračunati njegov faktorijel:

```
DELIMITER //
CREATE FUNCTION faktorijel(p_x INTEGER) RETURNS INTEGER
DETERMINISTIC
BEGIN
  DECLARE rez INTEGER DEFAULT 1;

  WHILE p_x > 0 do
    SET rez = rez * p_x;
    SET p_x = p_x - 1;
  END WHILE;

  RETURN rez;
END//
DELIMITER ;

SELECT faktorijel(4) FROM DUAL;
```

**Repeat petlja:** Napiši funkciju koja će za broj (definiran parametrom *p_x*) izračunati njegov faktorijel:

```
DELIMITER //
CREATE FUNCTION faktorijel_2(p_x INTEGER) RETURNS INTEGER
DETERMINISTIC
BEGIN
  DECLARE rez INTEGER DEFAULT 1;

  REPEAT
    SET rez = rez * p_x;
    SET ps_x = p_x - 1;
  UNTIL p_x = 0
  END REPEAT;

  RETURN rez;
END//
DELIMITER ;

SELECT faktorijel_2(4) FROM DUAL;s
```

**Pozivanje funkcije unutar funkcije:** Različiti načini kako da spremimo rezultat funkcije koju pozivamo unutar funkcije:

```
DELIMITER //
CREATE FUNCTION demo_faktorijeli(p_x INTEGER) RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
  DECLARE f1, f2 INTEGER DEFAULT 1;

  SET f1 = faktorijel(p_x);

  SELECT faktorijel(p_x) INTO f2;

  RETURN CONCAT("F1: ", f1, ", F2: ", f2);
END//
DELIMITER ;

SELECT demo_faktorijeli(4) FROM DUAL;
```

**Zadaci:**

1. Napiši funkciju koja će vratiti trenutni datum u obliku sljedećeg primjera **31.10.20.**
2. Napiši funkciju koja će za dva broja (definirane sa parametrima *a* i *b*) izračunati i vratiti njihov umnožak
3. Napiši funkciju koja će za određeni broj (definiranu sa parametrom *x*) vratiti 'DA' ako je broj paran, inače će vratiti vrijednost 'NE'
4. Napiši funkciju koja će za određeni račun (definiran sa parametrom *p_id_racun*) vratiti 'DA' ako račun ima barem jednu stavku, u suprotnom će vratiti vrijednost 'NE'. Isprobajte funkciju na nekom računu
5. Napiši funkciju koja će za određeno ime (definiran sa parametrom *p_ime*) prebrojati koliko puta se ono pojavljuje u tablicama zaposlenika i kupca (npr. ime 'Tea' se pojavljuje dva puta). Zatim napiši upit koji će prikazati sve kupce zajedno sa brojem pojavljivanja imena pojedinog kupca koristeći prethodno napisanu funkciju.