SQL - Ponavljanje

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
);

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

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

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
);

CREATE TABLE stavka_racun (
  id INTEGER NOT NULL,
  id_racun INTEGER NOT NULL,
  id_artikl INTEGER NOT NULL,
  kolicina INTEGER NOT NULL
);
```

Unos podataka

```
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);
```

Upit sa uvjetom: Prikaži sve kupce sa imenom *'Lea'* ili *'David'*

```
SELECT *
  FROM kupac
  WHERE ime = 'Lea' OR ime = 'David';
```

Generalizirana projekcija: Prikaži *id*-eve i puno ime (spojeno ime i prezime) svih kupaca

```
SELECT id, CONCAT(ime, " ", prezime) AS puno_ime
  FROM kupac;
```

Uvjet & sortiranje: Prikaži sve artikle koji imaju cijenu između 6 i 100 (tj. **[6, 100]**) i sortiraj rezultat silazno prema cijeni

```
SELECT *
  FROM artikl
  WHERE cijena BETWEEN 6 AND 100
  ORDER BY cijena DESC;
```

Operacije nad stringovima: Prikaži imena sa velikim slovima i dužinu imena (broj slova u imenu) svih kupaca čije prezime završava na *'ić'*

```
SELECT UPPER(ime), LENGTH(ime)
  FROM kupac
  WHERE prezime LIKE '%ić';
```

Distinct: Prikaži sva jedinstvena prezimena zaposlenika

```
SELECT DISTINCT prezime
  FROM zaposlenik;
```

Unija: Prikaži sva imena zaposlenika i kupaca u jednom stupcu

```
SELECT ime
  FROM kupac
UNION # dodati ALL ako ne želimo da se duplikati brišu
SELECT ime
  FROM zaposlenik;
```

Sortiranje: Prikaži prezimena i imena svih zaposlenika tako da su sortirani silazno prvo po prezimenu pa po imenu

```sql
SELECT prezime, ime
  FROM zaposlenik
  ORDER BY prezime DESC, ime DESC;
```

Join: Prilaži sve kupce i njihove pripadne račune

```sql
SELECT *
  FROM kupac k, racun r
  WHERE r.id_kupac = k.id;

-- ili
SELECT *
  FROM kupac k
  INNER JOIN racun r ON r.id_kupac = k.id;
```

Left outer join: Prikaži sve kupce i njihove pripadne račune, a pritom prikazati i kupce koji nemaju niti jedan račun

```sql
SELECT *
  FROM kupac k
  LEFT OUTER JOIN racun r ON r.id_kupac = k.id;
```

Full outer join: Prikaži sve kupce i zaposlenike koji su "surađivali" (navedeni na istim računima), pritom prikazati kupce i zaposlenike koji nisu navedeni na niti jednom računu

```sql
SELECT k.*, z.*
  FROM racun r
  INNER JOIN kupac k ON r.id_kupac = k.id
  RIGHT OUTER JOIN zaposlenik z ON z.id = r.id_zaposlenik
UNION
SELECT k.*, z.*
  FROM racun r
  INNER JOIN zaposlenik z ON z.id = r.id_zaposlenik
  RIGHT OUTER JOIN kupac k ON r.id_kupac = k.id;
```

Podupit: Prikaži sve zaposlenike koji se zovu *'Lea'*, *'David'* ili *'Tea'*

```sql
-- uvodni primjer (nije konačno rješenje zadatka)
SELECT *
  FROM zaposlenik
  WHERE ime IN ('Lea', 'David', 'Tea');

-- konačno rješenje
SELECT *
  FROM zaposlenik
  WHERE ime IN (SELECT DISTINCT ime FROM kupac);
```

Agregacija: Prikaži najveću, najmanju i srednju cijenu artikala

```sql
SELECT MAX(cijena), MIN(cijena), AVG(cijena)
  FROM artikl;
```

Podupit & Limit: Prikaži artikl sa najvećom cijenom

```sql
SELECT *
  FROM artikl
  WHERE cijena = (SELECT MAX(cijena) FROM artikl);

-- ili
SELECT *
  FROM artikl
  ORDER BY cijena DESC
  LIMIT 1;
```

Grupiranje i agregacija: Prikaži sve kupce i broj njihovih računa

```sql
SELECT k.*, COUNT(r.id) AS broj_racuna
  FROM kupac k
  INNER JOIN racun r ON k.id = r.id_kupac
  GROUP BY k.id;
```

Grupiranje i agregacija & Filtriranje grupa: Prikaži sve kupce i broj njihovih računa, pritom prikazati samo one kupce koji imaju barem dva računa

```sql
SELECT k.*, COUNT(r.id) AS broj_racuna
  FROM kupac k
  INNER JOIN racun r ON k.id = r.id_kupac
  GROUP BY k.id
  HAVING COUNT(r.id) > 1;
```

Grupiranje i agregacija: Prikaži sve račune i njihov ukupan iznos

```sql
SELECT r.*, SUM(cijena * kolicina)
  FROM racun r
  INNER JOIN stavka_racun s ON s.id_racun = r.id
  INNER JOIN artikl a ON s.id_artikl = a.id
  GROUP BY r.id;
```

Podupit & Korelirani podupit: Prikaži sve artikle koji su izdani u ukupnoj količini većoj od 5

```sql
SELECT a.*
  FROM artikl a
  WHERE a.id IN (SELECT id_artikl
                   FROM stavka_racun
                   GROUP BY id_artikl
                   HAVING SUM(kolicina) > 5);

-- primjer pomoću koreliranog upita (napomena: značajno sporije nego običan
podupit)
SELECT a.*
  FROM artikl a
    WHERE (
      SELECT SUM(kolicina)
      FROM stavka_racun
      WHERE id_artikl = a.id
    ) > 5;
```

Pogled (View): Napravi pogled koji prikazuje sve račune i njihov ukupan iznos

```sql
# DROP VIEW racun_sa_iznosom;
CREATE VIEW racun_sa_iznosom AS
SELECT r.*,
       SUM(cijena * kolicina) AS iznos
  FROM racun r
  INNER JOIN stavka_racun s ON s.id_racun = r.id
  INNER JOIN artikl a ON s.id_artikl = a.id
```

```
  GROUP BY r.id;

-- primjeri korištenja pogleda
SELECT *
  FROM racun_sa_iznosom;

SELECT *
  FROM racun_sa_iznosom
  WHERE broj = '00001';
```

Pogled (View) & Unos podataka: Napravi pogled koji prikazuje skupe artikle (cijena > 10), pritom se kroz pogled mogu unositi podaci

```
CREATE VIEW skup_artikl AS
SELECT *
  FROM artikl
  WHERE cijena > 10;

-- dohvaćanje podataka pogleda
SELECT *
  FROM skup_artikl;

-- unos kroz pogled, podatak nije vidljiv kroz pogled zbog uvjeta
INSERT INTO skup_artikl VALUES (24, 'Brašno', 5);

-- unos kroz pogled, podatak je vidljiv kroz pogled
INSERT INTO skup_artikl VALUES (25, 'Merci', 40);
```

Pogled (View) WITH CHECK OPTION: Napravi pogled koji prikazuje skupe artikle (cijena > 10), pritom se kroz pogled mogu unositi samo podaci koji zadovoljavaju uvjet (cijena > 10)

```
DROP VIEW skup_artikl;
CREATE VIEW skup_artikl AS
SELECT *
  FROM artikl
  WHERE cijena > 10
WITH CHECK OPTION;

-- unos kroz pogled, podatak nećemo moći unjeti kroz pogled zbog uvjeta i
ograničenja
INSERT INTO skup_artikl VALUES (26, 'Bonboni', 5);
-- unos podatka normalno prolazi jer je zadovoljen uvijet
INSERT INTO skup_artikl VALUES (27, 'Teletina', 32);
```

Zadaća

1. Prikaži sve zaposlenike čije ime sadrži barem 4 slova
2. Prikaži sva imena zaposlenika koja se pojavljuju kao imena kupaca
3. Ažuriraj artikle tako da im se cijena spusti za 10%
4. Prikaži artikle koji imaju iznadprosječnu cijenu
5. Prikaži sve artikle i račune na kojima su izdani, pritom prikazati i artikle koji nisu niti jednom kupljeni (niti jednom dodani na stavke računa)
6. Prikaži najučestalijeg kupca (kupac koji ima najviše računa)