DOKUMENTACIJA ZA PROJEKTNI ZADATAK „MESNICA“

TIM 24

Sveučilište Jurja Dobrile u Puli

Ivan Šimić

Diego Šćulac

Dario Žepić

Mauro Aničić

Patrik Kovačević

Smjer : Računarstvo

Kolegij : Baze podataka I

Sadržaj

1. UVOD

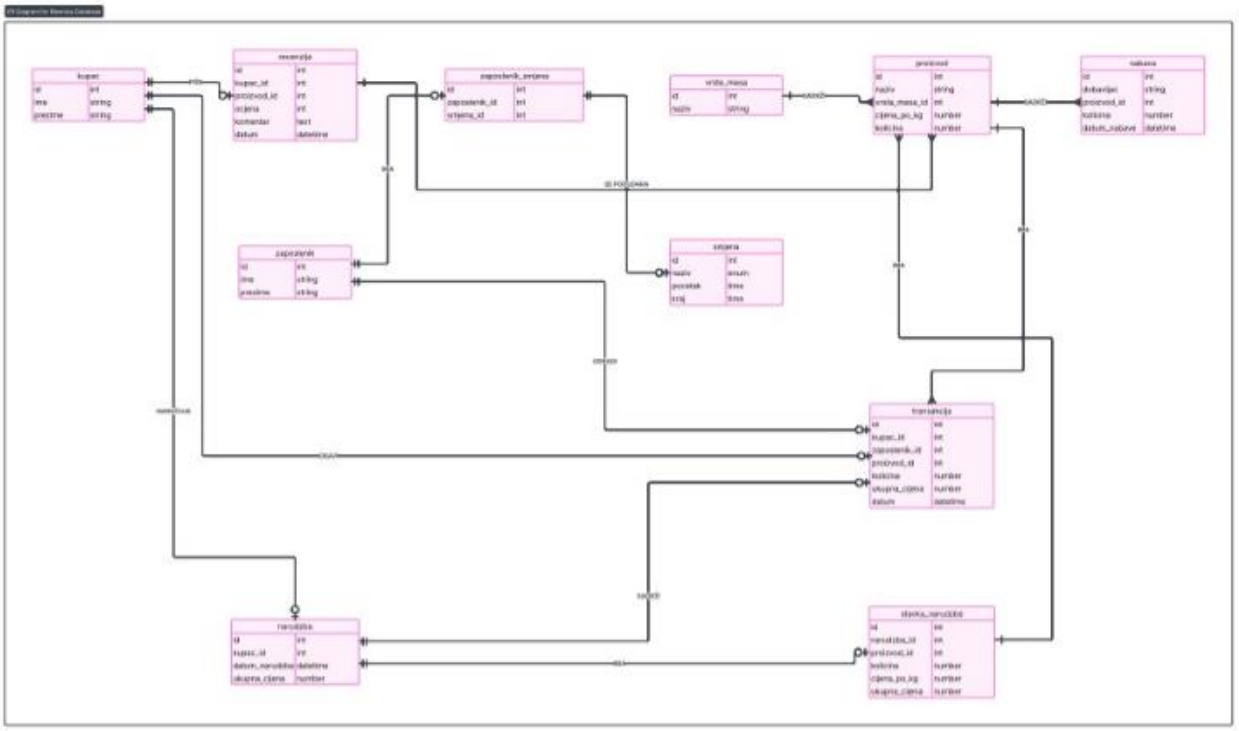
Ovaj projekt se od svoje početne verzije dosta promijenio, naravno na bolje. Kako su nam se znanje i vještine nadograđivale tokom semestra uvijek je postojala preinaka koja je poboljšala samu strukturu i funkciju pojedinih elemenata na izvedbu i rad same baze podataka. Sam proces izrade baze podataka je obavljen preko GitHuba zbog lakše komunikacije i nemogućnosti zajedničkog nalaženja. Izgradili smo bazu podataka imena ''Mesnica'' te smo nastojali što detaljnije opisati sve poslovne procese i konceptualni model koji je prikazan preko ER dijagrama. Naravno, naša baza je samo djelić one prave, koju smo morali reducirati i izričito specificirati opis poslovanja za potrebe našeg projekta. Krajnji cilj je opisati što se događa kada kupac dođe u mesnicu, od početnih koraka, narudžbe i kupovine, do evidencije proizvoda. Tu su također dodatni skupovi entiteta koji su detaljno opisani u konceptualnom modelu. EER dijagram, odnosno logička shema baze podataka je generirana u MySQL Workbench-u preko Reverse Engineering opcije. Sva imena, prezimena i ostali podaci su generirani nasumično pomoću online alata, dok su nazivi proizvoda i vrsta preneseni iz stvarne mesnice.

2. OPIS POSLOVNOG PROCESA

U mesnici se prati prodaja proizvoda. Prodavač obavlja unos narudžbi kupaca. Za svakog kupca se prati id, ime i prezime, Za svaku narudžbu prati se id, datum_narudžbe, kupac i ukupna_cijena. Proizvodi u mesnici imaju id, naziv, vrstu, cijenu i kolicinu. Prodavači rade na različitim smjenama, a za njih se prati id, ime i prezime. Kupac može napraviti više narudžbi, a svaka narudžba može sadržavati više proizvoda. Evidentiraju se i plaćanja narudžbi s datumom, kolicinom i iznosom.

3. ENTITY RELATIONSHIP (ER) DIJAGRAM

ER dijagram u nastavku jasno i detaljno prikazuje sve entitetske skupove s njihovim atributima, kao i skupove veza među njima. Strelice koje označavaju kardinalnost mapiranja pokazuju koliko entiteta može biti povezano s nekim entitetom putem određenog skupa veza.



3.1. OPIS ER DIJAGRAMA

Kupac → Narudžba

Jedan **kupac** može napraviti više **narudžbi**, dok svaka narudžba pripada samo jednom kupcu. Ova veza omogućava praćenje svih kupovina pojedinog kupca kroz vrijeme, što je korisno za analizu ponašanja korisnika, lojalnosti i statistiku potrošnje.

Narudžba → Stavka_narudžbe

Svaka **narudžba** se sastoji od jedne ili više **stavki**, pri čemu svaka stavka predstavlja određeni **proizvod** i njegovu naručenu količinu. Ova struktura omogućava fleksibilno upravljanje narudžbama koje uključuju više proizvoda te izračun ukupne cijene narudžbe.

Proizvod → Stavka_narudžbe

Jedan **proizvod** može biti dio mnogih stavki narudžbe, u različitim narudžbama i u različitim količinama. Ovo omogućava praćenje potražnje za svakim proizvodom i identifikaciju najprodavanijih artikala.

Zaposlenik → Transakcija

Jedan **zaposlenik** može obraditi više **transakcija** (npr. kao blagajnik, prodajni savjetnik ili administrator). Time se može analizirati učinak zaposlenika, njihov doprinos prodaji i količina obrađenih transakcija po zaposleniku.

Kupac → Transakcija

Jedan **kupac** može izvršiti više **transakcija**. Svaka transakcija predstavlja kupovinu koja može, ali ne mora, biti povezana s narudžbom. Ovo omogućava praćenje povijesti kupovina po korisniku, čak i kada narudžbe nisu prethodno formalno unesene.

Zaposlenik → Zaposlenik_smjena

Jedan **zaposlenik** može raditi u više **smjena**, dok istovremeno jedna **smjena** može uključivati više zaposlenika. Ova **veza** se realizira kroz pomoćnu tablicu **zaposlenik_smjena** koja omogućava evidentiranje rasporeda i radnog vremena zaposlenih.

Smjena → Zaposlenik_smjena

Kao nastavak prethodnog odnosa, kroz ovu vezu se može planirati i evidentirati kada i koji su zaposlenici radili. Može služiti i za obračun radnih sati, raspoređivanje osoblja i vođenje evidencije prisutnosti.

Proizvod → Nabava (

Jedan **proizvod** može biti predmet više **nabava**, koje dolaze od različitih dobavljača ili u različitim količinama i terminima. Ova veza omogućava praćenje zaliha i opskrbe, analizu dobavljača te planiranje budućih narudžbi prema povijesnim podacima.

Račun → Kupac i Zaposlenik

Entitet **račun** ima veze prema **kupcu** i **zaposleniku**. To znači da svaki račun prikazuje:

- **tko** je izvršio kupovinu (kupac),
- **tko** je obradio ili izdao račun (zaposlenik).

Račun vjerojatno predstavlja finalni dokument nakon obrade narudžbe i plaćanja, koji uključuje podatke o proizvodima, cijeni, datumu i izvršiteljima. Može se koristiti za fiskalne svrhe, analizu prodaje i korisničku podršku.

4. RELACIJSKI MODEL

vrste_mesa (id, naziv)

proizvod (id, naziv, vrsta_mesa_id, cijena_po_kg, kolicina)

kupac (id, ime, prezime)

zaposlenik (id, ime, prezime)

radno_vrijeme_mesnice (id, dan_u_tjednu, otvaranje, zatvaranje)

smjena (id, naziv, pocetak, kraj)

zaposlenik_smjena (id, zaposlenik_id, smjena_id)

nabava (id, dobavljac, proizvod_id, kolicina, datum_nabave)

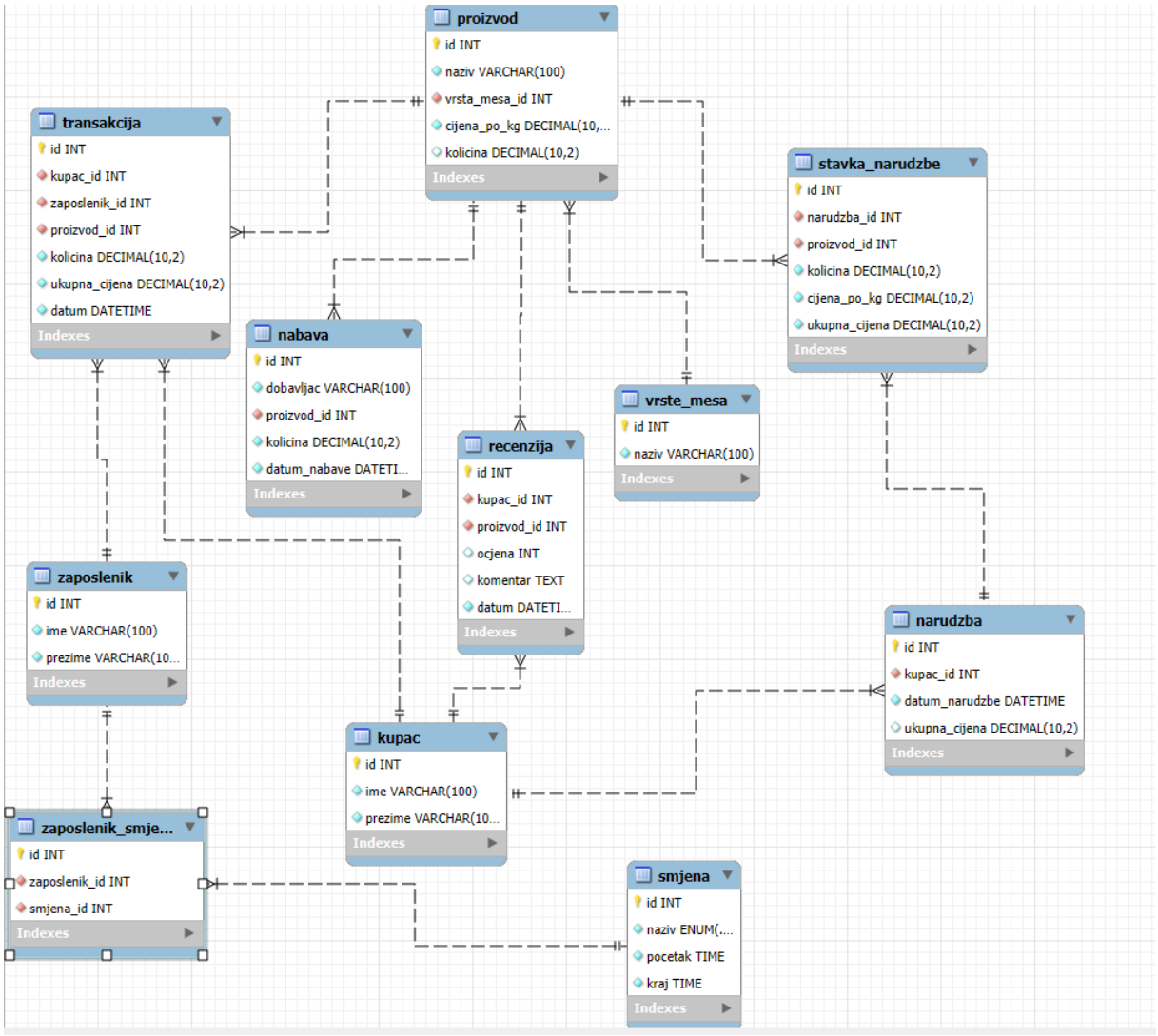
narudzba (id, kupac_id, datum_narudzbe, ukupna_cijena)

stavka_narudzbe (id, narudzba_id, proizvod_id, kolicina, cijena_po_kg, ukupna_cijena)

transakcija (id, kupac_id, zaposlenik_id, proizvod_id, kolicina, ukupna_cijena, datum)

recenzija (id, kupac_id, proizvod_id, ocjena, komentar, datum)

5. EER DIJAGRAM (MYSQL)



6. TABLICE

6.1. TABLICA VRSTE_MESA

Tablica vrste_mesa sadrži podatke o vrstama mesa dostupnim u mesnici. Svaki zapis identificiran je jedinstvenim identifikatorom id koji je cijeli broj i automatski se povećava za svaki novi unos. Atribut naziv predstavlja naziv vrste mesa i ne može biti prazan (NOT NULL), a tip je tekstualni niz do 100 znakova.

SQL :

```
CREATE TABLE vrste_mesa (
    id INT PRIMARY KEY AUTO_INCREMENT,
    naziv VARCHAR(100) NOT NULL
);
```

6.2. TABLICA PROIZVOD

Tablica proizvod sadrži informacije o pojedinačnim proizvodima mesnice. Svaki proizvod ima jedinstveni identifikator id koji je cijeli broj i automatski se povećava. Atribut naziv je naziv proizvoda, obavezno polje s maksimalno 100 znakova. vrsta_mesa_id označava na koju vrstu mesa se proizvod odnosi i predstavlja vanjski ključ prema tablici vrste_mesa. Cijena proizvoda po kilogramu pohranjena je u atributu cijena_po_kg kao decimalni broj s dva decimalna mjesta, što je također obavezno. Atribut kolicina označava količinu proizvoda, može biti decimalni broj s dva decimalna mjesta, a ako nije specificirana, vrijednost mu je zadana na 1.

SQL

```
CREATE TABLE proizvod (
    id INT PRIMARY KEY AUTO_INCREMENT,
    naziv VARCHAR(100) NOT NULL,
    vrsta_mesa_id INT NOT NULL,
    cijena_po_kg DECIMAL(10,2) NOT NULL,
    kolicina DECIMAL(10,2) DEFAULT 1,
    CONSTRAINT fk_proizvod_vrsta FOREIGN KEY (vrsta_mesa_id) REFERENCES
vrste_mesa(id)
);
```

6.3. TABLICA KUPAC

Tablica kupac evidentira kupce mesnice. Svaki kupac ima jedinstveni identifikator id koji automatski raste. Polja ime i prezime su obavezna i predstavljaju ime i prezime kupca, oba su tekstualni nizovi do 100 znakova.

SQL

```
CREATE TABLE kupac (
    id INT PRIMARY KEY AUTO_INCREMENT,
    ime VARCHAR(100) NOT NULL,
    prezime VARCHAR(100) NOT NULL
);
```

6.4 TABLICA ZAPOSLENIK

Tablica zaposlenik sadrži podatke o zaposlenicima mesnice. Svaki zaposlenik ima jedinstveni identifikator id koji je cijeli broj i automatski se povećava. Polja ime i prezime su obavezna i pohranjuju ime i prezime zaposlenika, oba su tekstualni nizovi do 100 znakova.

SLQ

```
CREATE TABLE zaposlenik (
    id INT PRIMARY KEY AUTO_INCREMENT,
    ime VARCHAR(100) NOT NULL,
    prezime VARCHAR(100) NOT NULL
);
```

6.5. TABLICA RADNO_VRIJEME_MESNICE

Tablica radno_vrijeme_mesnice bilježi radno vrijeme mesnice za svaki dan u tjednu. Svaki zapis ima jedinstveni id koji se automatski povećava. dan_u_tjednu je tekstualni niz do 20 znakova koji označava dan u tjednu. Polja otvaranje i zatvaranje pohranjuju vrijeme otvaranja i zatvaranja mesnice tog dana, oba su tipa TIME i obavezna su.

SQL

```
CREATE TABLE radno_vrijeme_mesnice (
    id INT PRIMARY KEY AUTO_INCREMENT,
    dan_u_tjednu VARCHAR(20) NOT NULL,
    otvaranje TIME NOT NULL,
    zatvaranje TIME NOT NULL
);
```

6.6. TABLICA SMJENA

Tablica smjena evidentira smjene u mesnici. Svaki zapis ima jedinstveni id koji se automatski povećava. Polje naziv je tip ENUM koji može biti samo vrijednost 'Jutarnja' ili 'Popodnevna'. Polja pocetak i kraj su vremena početka i kraja smjene, oba su tipa TIME i obavezna.

SQL

```
CREATE TABLE smjena (
    id INT PRIMARY KEY AUTO_INCREMENT,
    naziv ENUM('Jutarnja', 'Popodnevna') NOT NULL,
    pocetak TIME NOT NULL,
    kraj TIME NOT NULL
);
```

6.7. ZAPOSLENIK SMJENA

Tablica zaposlenik_smjena povezuje zaposlenike sa smjenama u kojima rade. Svaki zapis ima jedinstveni id koji se automatski povećava. Atribut zaposlenik_id je strani ključ prema tablici zaposlenik, dok je smjena_id strani ključ prema tablici smjena. Ova tablica omogućava dodjelu jednog zaposlenika jednoj ili više smjena.

SQL

```
CREATE TABLE zaposlenik_smjena (
    id INT PRIMARY KEY AUTO_INCREMENT,
    zaposlenik_id INT NOT NULL,
    smjena_id INT NOT NULL,
    FOREIGN KEY (zaposlenik_id) REFERENCES zaposlenik(id),
    FOREIGN KEY (smjena_id) REFERENCES smjena(id)
);
```

6.8. TABLICA NABAVA

Tablica nabava bilježi nabavke proizvoda od dobavljača. Svaki zapis ima jedinstveni id koji se automatski povećava. Polje dobavljac je tekstualni niz do 100 znakova i označava ime dobavljača, obavezno je. proizvod_id je strani ključ koji povezuje nabavku s proizvodom iz tablice proizvod. Polje kolicina označava količinu nabavljenog proizvoda i predstavlja decimalni broj s dva decimalna mjesta. Atribut datum_nabave bilježi datum i vrijeme nabave (tip DATETIME), i također je obavezan.

SQL

```
CREATE TABLE nabava (
    id INT PRIMARY KEY AUTO_INCREMENT,
    dobavljac VARCHAR(100) NOT NULL,
    proizvod_id INT NOT NULL,
    kolicina DECIMAL(10,2) NOT NULL,
    datum_nabave DATETIME NOT NULL,
    FOREIGN KEY (proizvod_id) REFERENCES proizvod(id)
);
```

6.9. TABLICA NARUDZBA

Tablica narudzba evidentira narudžbe kupaca. Svaki zapis ima jedinstveni id koji se automatski povećava. kupac_id je strani ključ koji povezuje narudžbu s kupcem iz tablice kupac. Polje datum_narudzbe je tipa DATETIME i označava kada je narudžba izvršena. ukupna_cijena predstavlja ukupnu cijenu narudžbe, decimalnog je tipa s dva decimalna mjesta, ali nije obavezno polje.

SQL

```
CREATE TABLE narudzba (
    id INT PRIMARY KEY AUTO_INCREMENT,
    kupac_id INT NOT NULL,
    datum_narudzbe DATETIME NOT NULL,
    ukupna_cijena DECIMAL(10,2),
    FOREIGN KEY (kupac_id) REFERENCES kupac(id)
);
```

6.10. TABLICA STAVKA NARUDZBE

Tablica stavka_narudzbe pohranjuje stavke pojedinačnih narudžbi. Svaka stavka ima jedinstveni id koji se automatski povećava. narudzba_id je strani ključ koji povezuje stavku s narudžbom, dok je proizvod_id strani ključ koji povezuje stavku s proizvodom. Polje kolicina predstavlja količinu proizvoda u narudžbi (decimalni broj). cijena_po_kg je cijena proizvoda po kilogramu u trenutku narudžbe (decimalni broj). ukupna_cijena predstavlja ukupnu cijenu te stavke, također decimalnog tipa i obavezno je.

SQL

```
CREATE TABLE stavka_narudzbe (
    id INT PRIMARY KEY AUTO_INCREMENT,
    narudzba_id INT NOT NULL,
    proizvod_id INT NOT NULL,
    kolicina DECIMAL(10,2) NOT NULL,
    cijena_po_kg DECIMAL(10,2) NOT NULL,
    ukupna_cijena DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (narudzba_id) REFERENCES narudzba(id),
    FOREIGN KEY (proizvod_id) REFERENCES proizvod(id)
);
```

6.11. TABLICA TRANSAKCIJA

Tablica transakcija bilježi detalje o transakcijama koje uključuju kupce, zaposlenike i proizvode. Svaki zapis ima jedinstveni id koji se automatski povećava. Polja kupac_id, zaposlenik_id i proizvod_id su strani ključevi povezani s tablicama kupac, zaposlenik i proizvod. kolicina označava količinu proizvoda u transakciji, a ukupna_cijena predstavlja ukupnu cijenu te transakcije. Oba su decimalni brojevi s dva decimalna mjesta i obavezna su. datum bilježi datum i vrijeme transakcije.

SQL

```
CREATE TABLE transakcija (
    id INT PRIMARY KEY AUTO_INCREMENT,
    kupac_id INT NOT NULL,
    zaposlenik_id INT NOT NULL,
    proizvod_id INT NOT NULL,
    kolicina DECIMAL(10,2) NOT NULL,
    ukupna_cijena DECIMAL(10,2) NOT NULL,
    datum DATETIME NOT NULL,
    FOREIGN KEY (kupac_id) REFERENCES kupac(id),
    FOREIGN KEY (zaposlenik_id) REFERENCES zaposlenik(id),
    FOREIGN KEY (proizvod_id) REFERENCES proizvod(id)
);
```

6.12. TABLICA RECENZIJA

Tablica recenzija pohranjuje ocjene i komentare koje kupci ostavljaju za proizvode. Svaki zapis ima jedinstveni id koji se automatski povećava. Polja kupac_id i proizvod_id su strani ključevi koji povezuju recenziju s kupcem i proizvodom iz njihovih tablica. Atribut ocjena je cijeli broj koji predstavlja ocjenu proizvoda, a polje komentar je tekstualni niz do 500 znakova u kojem kupac može ostaviti svoj komentar; oba su polja obavezna.

SQL

```
CREATE TABLE recenzija (
    id INT PRIMARY KEY AUTO_INCREMENT,
    kupac_id INT NOT NULL,
    proizvod_id INT NOT NULL,
    ocjena INT NOT NULL,
    komentar VARCHAR(500) NOT NULL,
    FOREIGN KEY (kupac_id) REFERENCES kupac(id),
    FOREIGN KEY (proizvod_id) REFERENCES proizvod(id)
);
```

7.UPITI

7.1 – 7.6 Dario Žepić

7.7 – 7.16 Patrik Kovačević

7.1. UPIT 1

```
CREATE VIEW popust_na_proizvode AS

SELECT
    p.naziv AS naziv_proizvoda,
    vm.naziv AS vrste_mesa,
    p.cijena_po_kg AS originalna_cijena,
    ROUND(
        CASE
            WHEN vm.naziv = 'Svinjsko meso' THEN p.cijena_po_kg *
0.90
            WHEN vm.naziv = 'Teleće meso' THEN p.cijena_po_kg * 0.85
            ELSE p.cijena_po_kg
        END, 2
    ) AS snizena_cijena,
    CASE
        WHEN vm.naziv = 'Svinjsko meso' THEN '10% popusta'
        WHEN vm.naziv = 'Teleće meso' THEN '15% popusta'
        ELSE 'Nema popusta'
    END AS status_popusta
FROM proizvod p
JOIN vrste_mesa vm ON p.vrsta_mesa_id = vm.id;

SELECT * FROM popust_na_proizvode;
```

OPIS :

Za rješavanje ovoga upita korištene su tablice proizvod i vrste_mesa, koje se povezuju putem stranog ključa vrsta_mesa_id iz tablice proizvod. Cilj upita je prikazati sve proizvode s izračunatom sniženom cijenom, ovisno o vrsti mesa.

Prvi korak bio je spajanje tablica proizvod i vrste_mesa pomoću naredbe JOIN, na temelju podudarnosti vrsta_mesa_id i id iz tablice vrste_mesa.

U SELECT dijelu upita odabrani su:

naziv proizvoda (iz tablice proizvod),

naziv vrste mesa (iz tablice vrste_mesa),

originalna cijena proizvoda po kilogramu.

Pomoću naredbe CASE određena je nova vrijednost – snižena cijena, ovisno o vrsti mesa: za Svinjsko meso izračunava se 10% popusta, za Teleće meso 15% popusta, ostale vrste mesa ostaju bez popusta.

Izračunata cijena se dodatno zaokružuje na dvije decimale funkcijom ROUND.

Korištenjem dodatnog CASE izraza, u stupcu status_popusta prikazuje se tekstualni opis visine popusta (ili oznaka "Nema popusta").

Na kraju je stvoren pogled pod nazivom popust_na_proizvode pomoću naredbe CREATE VIEW, a svim podacima iz pogleda pristupa se naredbom SELECT *.

7.2 UPIT 2

```
SELECT DISTINCT
  z.ime AS ime_zaposlenika,
  z.prezime AS prezime_zaposlenika,
  s.pocetak,
  s.kraj,
  k.ime AS ime_kupca,
  k.prezime AS prezime_kupca
FROM transakcija t
JOIN zaposlenik z ON t.zaposlenik_id = z.id
JOIN zaposlenik_smjena zs ON z.id = zs.zaposlenik_id
JOIN smjena s ON zs.smjena_id = s.id
JOIN kupac k ON t.kupac_id = k.id
WHERE k.prezime = 'Horvat'
  AND TIME(t.datum) BETWEEN s.pocetak AND s.kraj;

SELECT * FROM zaposlenici_s_horvatom;
```

OPIS :

Ovaj upit prikazuje imena i prezimena zaposlenika koji su radili u određenoj smjeni te su zaprimili barem jednu narudžbu od kupca prezimena "Horvat".
Za to se koristi više tablica: zaposlenik, smjena, zaposlenik_smjena, transakcija i kupac.

Unutar CREATE VIEW naredbe definira se pogled zaposlenici_s_horvatom, u kojem se:

spajaju tablice preko zajedničkih ID-eva pomoću JOIN,

filtriraju samo one transakcije gdje je kupac "Horvat" (k.prezime = 'Horvat'),

dodatno se provjerava je li vrijeme transakcije unutar radnog vremena smjene (TIME(t.datum) BETWEEN s.pocetak AND s.kraj).

Naredba SELECT DISTINCT osigurava da se isti zaposlenik ne prikaže više puta ako je zaprimio više narudžbi. Na kraju se iz pogleda dohvaćaju svi rezultati s SELECT * FROM zaposlenici_s_horvatom;.

7.3. UPIT 3

```
CREATE VIEW proizvodi_prodani_u_svibnju AS


SELECT
    vm.naziv AS vrste_mesa,
    p.naziv AS proizvod,
    p.cijena_po_kg,
    COALESCE(SUM(t.kolicina), 0) AS ukupno_prodano
FROM proizvod p
JOIN vrste_mesa vm ON p.vrsta_mesa_id = vm.id
LEFT JOIN transakcija t ON p.id = t.proizvod_id
WHERE t.datum BETWEEN '2025-05-01' AND '2025-05-31'
GROUP BY vm.naziv, p.naziv, p.cijena_po_kg
HAVING SUM(t.kolicina) > 0
ORDER BY ukupno_prodano DESC;


SELECT * FROM proizvodi_prodani_u_svibnju;
```

OPIS :

Ovaj upit prikazuje proizvode (naziv i vrstu mesa), njihovu cijenu po kilogramu i ukupnu količinu prodanu u svibnju 2025..
Korištene su tablice proizvod, vrste_mesa i transakcija, gdje se spajaju podaci o proizvodima i vrstama mesa, a transakcije se ograničavaju na datume između 1.5. i 31.5.2025.

Pomoću SUM(t.kolicina) računa se ukupno prodana količina, a uvjet HAVING SUM(...) > 0 isključuje proizvode koji nisu prodani.
Rezultati su sortirani silazno prema količini prodaje (ORDER BY DESC), a COALESCE osigurava da se količine pravilno prikažu.

7.4. UPIT 4

```
SELECT * FROM proizvodi_prodani_u_svibnju;


CREATE VIEW provjera_recenzija AS


SELECT
    k.ime AS ime_kupca,
    k.prezime AS prezime_kupca,
    r.ocjena,
    r.komentar,
    r.datum AS datum_recenzije,
    t.datum AS datum_transakcije,
    p.naziv AS proizvod,
    t.kolicina,
    t.ukupna_cijena,
    z.ime AS ime_zaposlenika,
    z.prezime AS prezime_zaposlenika
FROM recenzija r
JOIN kupac k ON r.kupac_id = k.id
JOIN transakcija t ON k.id = t.kupac_id
JOIN proizvod p ON t.proizvod_id = p.id
JOIN zaposlenik z ON t.zaposlenik_id = z.id
WHERE r.ocjena = 5
ORDER BY k.prezime, t.datum;


SELECT * FROM provjera_recenzija;
```

OPIS :

Ovaj upit prikazuje sve recenzije s najvišom ocjenom (ocjena = 5), zajedno s transakcijama koje su ti kupci obavili, uključujući detalje o kupljenom proizvodu, količini, cijeni, te imenu i prezimenu zaposlenika koji je obavio prodaju.

Korištene su tablice: recenzija, kupac, transakcija, proizvod, zaposlenik.
Pomoću JOIN povezuju se svi relevantni podaci, a ORDER BY sortira rezultate po prezimenu kupca i datumu transakcije.

Cilj je uvidjeti s kojim vrstama mesa i zaposlenicima su kupci najzadovoljniji.

7.5. UPIT 5

```
SELECT * FROM provjera_recenzija;
CREATE VIEW jednostavna_ili_slozena AS
-- Jednostavne narudžbe (samo 1 ili 2 proizvoda)
SELECT
    n.id AS narudzba_id,
    k.ime AS ime_kupca,
    k.prezime AS prezime_kupca,
    COUNT(sn.id) AS broj_proizvoda,
    'Jednostavna narudžba' AS tip_narudzbe
FROM narudzba n
JOIN kupac k ON n.kupac_id = k.id
JOIN stavka_narudzbe sn ON n.id = sn.narudzba_id
GROUP BY n.id
HAVING COUNT(sn.id) <3
UNION
-- Složene narudžbe (3 ili više proizvoda)
SELECT
    n.id AS narudzba_id,
    k.ime AS ime_kupca,
    k.prezime AS prezime_kupca,
    COUNT(sn.id) AS broj_proizvoda,
    'Složena narudžba' AS tip_narudzbe
FROM narudzba n
JOIN kupac k ON n.kupac_id = k.id
JOIN stavka_narudzbe sn ON n.id = sn.narudzba_id
GROUP BY n.id
HAVING COUNT(sn.id) >= 3;

SELECT * FROM jednostavna_ili_slozena;
```

OPIS :

Ovim upitom određuje se tip narudžbe na temelju broja proizvoda u njoj:

Jednostavne narudžbe imaju 1 ili 2 stavke,

Složene narudžbe imaju 3 ili više stavki.

Korištene su tablice: narudzba, kupac, stavka_narudzbe.
GROUP BY grupira prema narudžbi, a COUNT(sn.id) broji stavke.
UNION spaja rezultate obje vrste narudžbi, a svaka vrsta ima svoj opis (AS tip_narudzbe).

7.6 UPIT 6

```
CREATE VIEW nabava_proizvoda AS
SELECT
    p.id AS proizvod_id,
    p.naziv AS naziv_proizvoda,
    COALESCE(n.dobavljac, 'Nema nabave') AS dobavljac,
    n.kolicina,
    n.datum_nabave
FROM proizvod p
LEFT JOIN nabava n ON p.id = n.proizvod_id
ORDER BY p.naziv, n.datum_nabave;

SELECT * FROM nabava_proizvoda;
```

OPIS :

Ovaj upit prikazuje sve proizvode, bez obzira na to imaju li evidentiranu nabavu.
Korišten je LEFT JOIN između tablica proizvod i nabava kako bi se prikazali i oni proizvodi koji nemaju vezanu nabavu.

Ako nema podataka o dobavljaču, koristi se COALESCE za prikaz poruke "Nema nabave".
Rezultat je sortiran po nazivu proizvoda i datumu nabave.

7.7. UPIT 7

```
SELECT p.naziv AS proizvod, vm.naziv AS vrsta_mesa, p.cijena_po_kg
FROM proizvod p
JOIN vrste_mesa vm ON p.vrsta_mesa_id = vm.id
ORDER BY vm.naziv, p.naziv;
```

OPIS :

Ovaj upit prikazuje sve proizvode dostupne u tablici proizvod, uz pripadajuću vrstu mesa i cijenu po kilogramu.

Pomoću JOIN spajaju se tablice proizvod i vrste_mesa, preko stranog ključa vrsta_mesa_id.

Atribut p.naziv prikazuje naziv proizvoda, vm.naziv naziv vrste mesa, a p.cijena_po_kg prikazuje cijenu.

Rezultat se sortira prema vrsti mesa, a zatim nazivu proizvoda (ORDER BY vm.naziv, p.naziv), radi preglednosti.

Ovaj upit je koristan za prikaz asortimana proizvoda u mesnici po vrstama mesa.

7.8. UPIT 8

```
SELECT dobavljac, SUM(kolicina) AS ukupna_kolicina
FROM nabava
GROUP BY dobavljac
ORDER BY ukupna_kolicina DESC;
```

OPIS :

Ovaj upit prikazuje ukupnu količinu mesa koju je svaki dobavljač isporučio, koristeći podatke iz tablice nabava.

Funkcija SUM(kolicina) zbraja ukupne količine mesa po svakom dobavljaču.

GROUP BY dobavljac grupira rezultate po imenu dobavljača.

ORDER BY ukupna_kolicina DESC sortira rezultate silazno, pa se na vrhu nalaze dobavljači koji su isporučili najviše mesa.

Ovaj upit omogućuje uvid u to koji su dobavljači najaktivniji ili najvažniji za poslovanje mesnice.

7.9. UPIT 9

TRAŽENO RJEŠENJE:

Prikaz proizvoda koji je najviše puta naručen po količini iz tablice stavka_narudzbe.

KOD:

```
SELECT p.naziv, SUM(sn.kolicina) AS ukupno_prodano
FROM stavka_narudzbe sn
JOIN proizvod p ON sn.proizvod_id = p.id
GROUP BY sn.proizvod_id
ORDER BY ukupno_prodano DESC
LIMIT 1;
```

OPIS :

Ovaj upit koristi se za dohvat proizvoda koji je u ukupnom zbroju naručen u najvećoj količini.
Iz tablice `stavka_narudzbe` zbrajaju se količine svih narudžbi po proizvodu korištenjem agregatne funkcije `SUM`.
Zatim se ti rezultati povezuju s tablicom `proizvod` pomoću `JOIN` naredbe kako bi se dobilo ime proizvoda.
Nakon što se svi proizvodi grupiraju (`GROUP BY`) prema svojem identifikatoru, upit ih sortira silazno (`ORDER BY ... DESC`) po ukupno naručenoj količini.
Primjenom `LIMIT 1` vraća se samo jedan proizvod koji je najviše naručen u ukupnoj količini, čime se jednostavno identificira najtraženiji artikl u sustavu.
Ovakva analiza korisna je za poslovne odluke, kao što su dodatne promocije najprodavanijih artikala ili optimizacija zaliha.

7.10 UPIT 10

TRAŽENO RJEŠENJE:

Izračun ukupne zarade za svaki proizvod temeljem ukupne cijene u transakcijama.

KOD:

```
SELECT p.naziv, SUM(t.ukupna_cijena) AS ukupna_zarada
FROM transakcija t
JOIN proizvod p ON t.proizvod_id = p.id
GROUP BY t.proizvod_id
ORDER BY ukupna_zarada DESC;
```

OPIS :

Ovaj upit prikazuje koliko je svaki proizvod ukupno zaradio kroz sve transakcije evidentirane u bazi podataka.
Za izračun koristi se agregatna funkcija `SUM` nad stupcem `ukupna_cijena`, koji prikazuje iznos ostvaren svakom pojedinom transakcijom.
Korištenjem `JOIN` naredbe povezuje se tablica `transakcija` s tablicom `proizvod`, kako bi se za svaku zaradu mogao dohvatiti pripadajući naziv proizvoda.
Rezultati se grupiraju po proizvodima (`GROUP BY`) prema njihovom ID-u, a zatim sortiraju silazno (`ORDER BY ... DESC`) po ukupnoj zaradi, omogućujući uvid u proizvode koji ostvaruju najveći prihod.
Ovaj podatak može biti od velike koristi menadžmentu za kreiranje strategije prodaje i analizu profitabilnosti pojedinih artikala.

7.11. UPIT 11

TRAŽENO RJEŠENJE:

Prikaz broja narudžbi za svaki dan u tablici narudzba.

KOD:

```
SELECT DATE(datum_narudzbe) AS dan, COUNT(*) AS broj_narudzbi
FROM narudzba
GROUP BY dan
ORDER BY dan;
```

OPIS :

Cilj ovog upita je izračunati i prikazati broj narudžbi izvršenih svakog dana.
Pomoću funkcije `DATE()` izdvajamo samo datum iz vremenskog zapisa `datum_narudzbe` (ignorira se vrijeme), čime se sve narudžbe unutar istog dana svode pod istu vrijednost.
Korištenjem agregatne funkcije `COUNT(*)` brojimo koliko se narudžbi dogodilo za svaki pojedini dan.
Nakon grupiranja rezultata prema datumu (`GROUP BY`), koristi se `ORDER BY` za sortiranje datuma u uzlaznom redoslijedu.
Ovaj upit je iznimno koristan za analizu frekvencije narudžbi tijekom vremena te za prepoznavanje dana kada postoji veća potražnja.
Menadžeri mogu koristiti ove informacije za planiranje radne snage, optimizaciju zaliha i provođenje ciljanih marketinških kampanja.

7.12 . UPIT 12

TRAŽENO RJEŠENJE:

Prikaz prosječne ocjene za svaki proizvod koji ima barem jednu recenziju.

KOD:

```
SELECT p.naziv, AVG(r.ocjena) AS prosjecna_ocjena
FROM recenzija r
JOIN proizvod p ON r.proizvod_id = p.id
GROUP BY p.id
HAVING COUNT(r.id) >= 1
ORDER BY prosjecna_ocjena DESC;
```

OPIS :

Ovim upitom dobivamo prosječnu ocjenu koju je svaki proizvod dobio od korisnika koji su ostavili recenziju.
Podaci se dohvaćaju iz tablica `recenzija` i `proizvod` koje se spajaju prema ID-u proizvoda pomoću `JOIN` naredbe.
Za svaku grupu recenzija koje pripadaju istom proizvodu izračunava se srednja vrijednost korištenjem funkcije `AVG`.
Uvjet `HAVING COUNT(r.id) >= 1` osigurava da se u rezultate uključe samo proizvodi koji imaju barem jednu recenziju.
Na kraju, rezultati se sortiraju silazno (`ORDER BY`) po prosječnoj ocjeni, čime se na vrhu prikazuju najkvalitetniji proizvodi prema ocjenama korisnika.
Takvi uvidi omogućuju trgovcima da istaknu proizvode visoke kvalitete i poboljšaju one s lošijim ocjenama.

7.13 UPIT 13

TRAŽENO RJEŠENJE:

Prikaz svih zaposlenika sa smjenama, uključujući ime, prezime, naziv smjene te početak i kraj.

KOD:

```
SELECT z.ime, z.prezime, s.naziv AS smjena, s.pocetak, s.kraj
FROM zaposlenik z
JOIN zaposlenik_smjena zs ON z.id = zs.zaposlenik_id
JOIN smjena s ON zs.smjena_id = s.id
ORDER BY z.prezime, z.ime;
```

OPIS :

Ovaj upit omogućuje detaljan pregled rasporeda rada svakog zaposlenika.
Povezuje tri tablice: `zaposlenik`, `zaposlenik_smjena` i `smjena` kako bi za svakog zaposlenika dobili sve informacije o njegovoj dodijeljenoj smjeni.
Korištenjem `JOIN` naredbi ostvaruje se poveznica između zaposlenika i njihovih smjena preko međutablice `zaposlenik_smjena`.
Iz rezultata se prikazuje ime i prezime zaposlenika te naziv smjene s pripadajućim vremenima početka i kraja.
Na kraju se rezultati sortiraju abecedno po prezimenu i imenu, kako bi se olakšala preglednost i identifikacija pojedinih zaposlenika.
Takva struktura je važna za organizaciju ljudskih resursa, praćenje radnog vremena i optimizaciju rasporeda.

7.14 UPIT 14

TRAŽENO RJEŠENJE:

Prikaz ukupne potrošnje za svakog kupca na temelju njihovih narudžbi.

KOD:

```
SELECT k.ime, k.prezime, SUM(n.ukupna_cijena) AS ukupno_potroseno
FROM narudzba n
JOIN kupac k ON n.kupac_id = k.id
GROUP BY k.id
ORDER BY ukupno_potroseno DESC;
```

OPIS :

Ovaj upit izračunava koliko je svaki kupac ukupno potrošio kroz svoje narudžbe.
Korištenjem `JOIN` naredbe povezuje se tablica `narudzba` s tablicom `kupac`, čime se omogućuje prikaz imena i prezimena kupca.
Zatim se pomoću agregatne funkcije `SUM` zbrajaju vrijednosti iz stupca `ukupna_cijena` za sve narudžbe pojedinog kupca.
Rezultati se grupiraju prema kupčevom ID-u kako bi se dobio po jedan redak za svakog kupca, a zatim sortiraju po ukupnoj potrošnji u silaznom redoslijedu.
Ovaj upit je posebno koristan za prepoznavanje najvjernijih ili najizdašnijih kupaca, a može poslužiti za ciljanje lojalnih korisnika posebnim ponudama ili popustima.

7.15 UPIT 15

TRAŽENO RJEŠENJE:

Prikaz svih proizvoda čija je cijena po kilogramu veća od prosjeka svih proizvoda.

KOD:

```
SELECT naziv, cijena_po_kg
FROM proizvod
WHERE cijena_po_kg > (SELECT AVG(cijena_po_kg) FROM proizvod)
ORDER BY cijena_po_kg DESC;
```

OPIS :

Ovaj upit identificira proizvode čija je cijena po kilogramu viša od prosjeka svih cijena u tablici `proizvod`.
Korištenjem podupita `(SELECT AVG(...))` računa se prosječna cijena, a zatim se koristi u `WHERE` uvjetu za filtriranje svih skupljih proizvoda.
Rezultati se prikazuju s nazivom proizvoda i cijenom te se sortiraju u silaznom redoslijedu, počevši od najskupljeg.
Ovakvi podaci su korisni za analizu strukture cijena, postavljanje cjenovne politike ili isticanje premium proizvoda u ponudi.

7.16. UPIT 16

TRAŽENO RJEŠENJE:

Prikaz da li je mesnica otvorena za određeni dan i vrijeme (primjer: ponedjeljak u 14:00).

KOD:

```
SELECT dan_u_tjednu, otvaranje, zatvaranje,
       '14:00:00' BETWEEN otvaranje AND zatvaranje AS otvorena
FROM radno_vrijeme_mesnice
WHERE dan_u_tjednu = 'Ponedjeljak';
```

OPIS :

Ovaj upit provjerava da li mesnica radi u konkretno navedenom vremenu.
U ovom primjeru koristi se vrijeme `'14:00:00'`, koje se uspoređuje s vremenima otvaranja i zatvaranja u tablici `radno_vrijeme_mesnice`.
Koristi se logički izraz `'vrijeme' BETWEEN otvaranje AND zatvaranje` kako bi se dobila informacija u obliku 1 (otvoreno) ili 0 (zatvoreno).
Filtriranjem po `dan_u_tjednu = 'Ponedjeljak'` provjerava se status rada samo za određeni dan.
Ovaj upit je posebno praktičan za implementaciju korisničkih funkcionalnosti kao što je prikaz radnog vremena trgovine u aplikacijama ili na web stranicama

8. ZAKLJUČAK

Smatramo da naša baza podataka dobro funkcionira u ovakvom obliku, ali isto tako znamo da postoji još mnogo prostora za unaprjeđenje. Sama mesnica nije pretjerano kompleksna, ali obuhvaća niz važnih elemenata koje je potrebno precizno definirati i povezati. Koristili smo osnovne tablice i podatke kako bismo obuhvatili ključne poslovne procese. Unatoč tome, smatramo da smo izradili kvalitetnu bazu podataka za jednu mesnicu te uspješno ostvarili sve ciljeve koji su bili postavljeni u ovom projektnom zadatku.

Za izradu ovog projekta koristili smo:

- za komunikaciju: Whatsapp

- za pisanje SQL skripte: MySQL Workbench

- za kreiranje ER dijagrama: Lucidchart

- za izradu dokumentacije: GitHub