Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli



**TIM 4: Webshop informatičke opreme**

Loren Bažon

Morena Martan

Bruno Rebić

Fran Barba

Josip Milković

Leo Hrvojić

Smjer : Informatika

Kolegij : Baze podataka II

Mentor : doc. Dr. sc. Goran Oreški

Asistent : mag. inf. Romeo Šajina

Pula, studeni, 2024. godina

Sadržaj

[Opis poslovnog procesa 3](#_gjdgxs)

[ER dijagram 4](#_30j0zll)

[Opis ER dijagrama 5](#_1fob9te)

[Kombinacija shema 6](#_2et92p0)

[EER DIJAGRAM (MYSQL WORKBENCH) 7](#_3dy6vkm)

[Tablice 7](#_1t3h5sf)

[tablice Loren 7](#_4d34og8)

[tablice Morena 10](#_2s8eyo1)

[tablice Bruno 10](#_17dp8vu)

[tablice Fran 11](#_3rdcrjn)

[tablice Josip 11](#_26in1rg)

[tablice Leo 11](#_lnxbz9)

[Ograničenja tablice: 12](#_35nkun2)

[Ograničenja tablice: 13](#_1ksv4uv)

[Opis funkcionalnosti: 13](#_44sinio)

[Napomena: 14](#_2jxsxqh)

[Ograničenja tablice: 15](#_z337ya)

[Opis funkcionalnosti: 15](#_3j2qqm3)

[Primjeri upotrebe: 15](#_1y810tw)

[Napomena: 15](#_4i7ojhp)

[Upiti i pogledi 15](#_2xcytpi)

[Upiti i pogledi Loren 16](#_1ci93xb)

[Upiti i pogledi Morena 22](#_3whwml4)

[Upiti i pogledi Bruno 22](#_2bn6wsx)

[Upiti i pogledi Fran 22](#_qsh70q)

[Upiti i pogledi Josip 22](#_3as4poj)

[Upiti i pogledi Leo 22](#_1pxezwc)

[Pogled: Profil Korisnika 22](#_49x2ik5)

[Pogled: Preporuke drugih 23](#_2p2csry)

[Pogled: Popularni Proizvodi 24](#_23ckvvd)

[Pogled: Narudžbe Korisnika 25](#_ihv636)

[Funkcije 28](#_32hioqz)

[Funkcije Loren 28](#_1hmsyys)

[Funkcije Morena 31](#_41mghml)

[Funkcije Bruno 31](#_2grqrue)

[Funkcije Fran 31](#_vx1227)

[Funkcije Josip 31](#_3fwokq0)

[Funkcije Leo 31](#_1v1yuxt)

[Procedure 32](#_4f1mdlm)

[Procedure Loren 32](#_2u6wntf)

[Procedure Morena 43](#_19c6y18)

[Procedure Bruno 43](#_3tbugp1)

[Procedure Fran 43](#_28h4qwu)

[Procedure Josip 43](#_nmf14n)

[Procedure Leo 43](#_37m2jsg)

[Okidači 48](#_1mrcu09)

[Okidači Loren 48](#_46r0co2)

[Okidači Morena 51](#_2lwamvv)

[Okidači Bruno 51](#_3l18frh)

[Okidači Fran 51](#_206ipza)

[Okidači Josip 51](#_4k668n3)

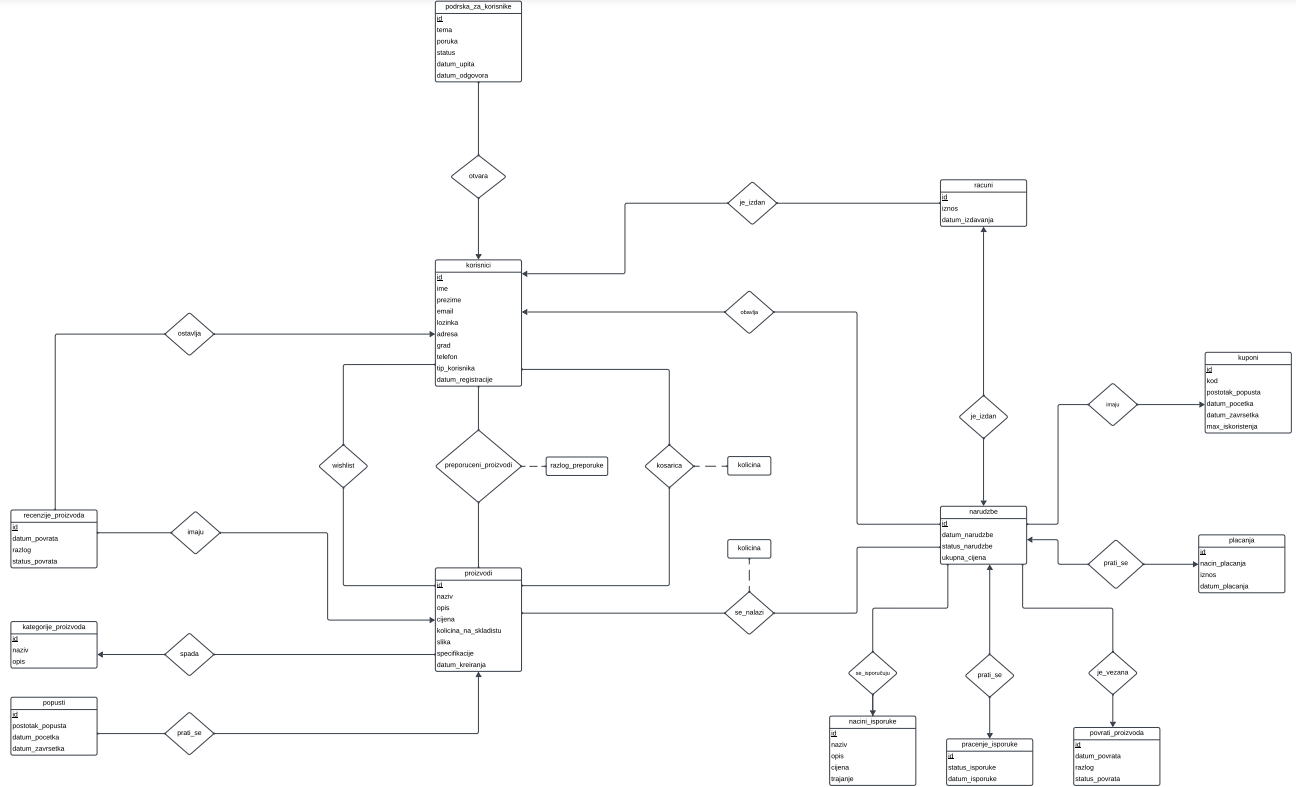
[Okidači Leo 51](#_2zbgiuw)

[Transakcije 52](#_1egqt2p)

# Opis poslovnog procesa

Naš webshop informatičke opreme omogućava korisnicima kupnju proizvoda te upravljanje narudžbama kroz jednostavan i intuitivan proces. Kada korisnik dođe na početnu stranicu webshopa, prikazuju mu se promotivni proizvodi, kategorije i preporuke. Proizvodi su rangirani prema popularnosti, popustima ili novitetima. Za *proizvode* se prati naziv, opis, cijena, kategorija\_id, kolicina\_na\_skladistu, slika, specifikacije, datum\_kreiranja. Prijavljeni korisnici vide preporučene proizvode (tablica *preporuceni\_proizvodi*) na temelju njihovih preferencija. Za prijavljene *korisnike* se prati ime, prezime, email, lozinka, adresa, grad, telefon, tip\_korisnika(admin ili kupac) te datum\_registracije. Korisnici mogu pregledavati proizvode kroz kategorije (*kategorije\_proizvoda*) za koje se prati naziv i opis. Klikom na određeni proizvod otvara se detaljna stranica sa već spomenutim informacijama o proizvodu, recenzijama, popustima te promotivnim kuponima. Za *recenzije* se prati ocjena, komentar i datum\_recenzije. Za *popuste* se prati postotak, datum početka i datum završetka te se za *kupone* prati kod, postotak popusta, datum početka, datum završetka te maksimalni broj korištenja. Sustav implementira listu želja, odnosno *wishlist*, koji omogućava korisniku da spremi omiljene proizvode koje planira kupiti. Kad korisnik želi kupiti određeni proizvod, klikom na „Dodaj u košaricu“ sustav provjerava dostupnost proizvoda (tablica *proizvodi,* atribut kolicina\_na\_skladistu). Ako je proizvod dostupan, dodaje se u *košaricu* za koju se prati količina. Ako je korisnik prijavljen, u košarici se veže njegov id sa proizvodom koji je dodao u košaricu. Ako korisnik nije prijavljen, kreira se privremeni *sesija\_id* koji se koristi za pohranu proizvoda u košaricu. Korisnik prije potvrde narudžbe može pregledati sve proizvode u košarici, promijeniti količine ili ukloniti stavke. Na kraju se prikazuje ukupna cijena (sa ili bez popusta) te troškovi dostave koji se računaju iz tablice *nacin\_isporuke*. Za način isporukese prati naziv, opis, cijena, trajanje. Da bi korisnik naručio proizvode, mora biti prijavljen. Ako nije prijavljen ili nema račun, mora se registrirati. Kada je to sve obavljeno, klikom na „Potvrdi narudžbu“ sustav kreira zapis u tablici *narudzbe* za koju se prati datum\_narudzbe, status\_narudzbe i ukupna cijena. Proizvodi iz košarice se prenose u tablicu *stavke\_narudzbe* za koju se prati količina. Korisnik bira način plaćanja koje sustav validira te se zapisuje u tablicu *placanja* za koju se prati iznos i datum plaćanja. Nakon uspješnog plaćanja, kreira se *račun* za koji se prati iznos i datum izdavanja. Korisnik može vidjeti status svoje narudžbe putem webshopa kroz tablicu *pracenje\_isporuke* za koju se prati status\_isporuke („priprema“, „poslano“ i „dostavljeno“) i datum\_isporuke. Ako nekim slučajem korisnik želi napraviti povrat narudžbe, podnosi zahtjev webshopu te se kreira zapis u tablici *povrati\_proizvoda* za koju se prati datum\_povrata, razlog i status\_povrata. Za bilo kakve nejasnoće ili probleme na koje korisnik naiđe za vrijeme korištenja nađeg webshopa, korisnik se obraća službi za korisnike. Povijest razgovora korisnika za službom se sprema u tablicu *podrska\_za\_korisnike* za koju se prati tema, poruka, status, datum\_upita i datum\_odgovora.

# ER dijagram



# Opis ER dijagrama

**Proizvod** spada pod **kategoriju** (Jedan proizvod može biti iz jedne kategorije, dok iz jedne kategorije može biti više proizvoda). Jedan na više.

**Korisnik** obavlja **narudžbe** (Jedan korisnik može obavljati više narudzbi, dok je jedna narudžba obavljena od strane samo jednog korisnika) jedan na više.

**Narudžbe** se isporučuju **načinom isporuke** (Jedna narudžba se može isporučiti samo jednim načinom isporuke, dok jednim načinom isporuke se može isporučiti više narudžbi) jedan na više.

**Narudžbe** imaju **kupone** (Jedna narudžba ima jedan kupon, dok jedan kupon može biti na više narudžbi) jedan na više.

**Proizvodi** se nalaze na **narudžbama** (jedan proizvod može biti na više narudžbi, dok na jednoj narudžbi može biti više proizvoda) više na više

Stvara se dodatni entitet **stavke\_narudzbe** gdje se prati količina proizvoda na narudžbama, a primarni ključevi entiteta proizvodi i narudzbe se referenciraju kao strani ključ u stavke\_narudzbe.

**Narudžbe** se prati **plaćanje** (Jednoj narudžbi se prati jedno plaćanje, dok jedno plaćanje je praćena za jednu narudžbu) jedan na jedan.

**Korisniku** je izdan **račun** (Jedan račun je izdan za jednog korisnika, dok jedan korisnik može imati više izdanih računa) jedan na više.

**Račun** je izdan za **narudžbu** (Za jednu narudžbu je izdan samo jedan račun, dok je jedan račun vezan samo za jednu narudžbu) jedan na jedan

**Popusti** se prati za **proizvode** (Jedan proizvod može imati više različitih popusta, dok jedan popust koji se prati je vezan samo za jedan proizvod) jedan na više.

**Narudžba** može imati **povrat** (Jedan povrat proizvoda može biti vezan za više narudžbi, dok jedna narudžba ima jedan povrat.) jedan na više

**Proizvod** ima **recenziju** (Jedna recenzija je vezana za jedan proizvod, dok jedan proizvod može imati više recenzija) jedan na više

**Korisnik** ostavlja **recenziju** (Jedan korisnik može ostaviti više recenzija, dok jedna recenzija je napisana od strane jednog korisnika) jedan na više

**Korisnik** sprema wishlistu **proizvoda** (Jedan korisnik može spremiti više proizvoda, dok jedan proizvod može biti spremljen kod više korisnika) više na više.

Stvara se dodatni entitet **wishlist** gdje primarni ključevi entiteta proizvodi i korisnici se referenciraju kao strani ključ u wishlist.

Za **narudžbu** se prati **isporuka** (Jedna narudzba ima jednu isporuku i jedna isporuka prati jednu narudžbu) jedan na jedan

**Korisniku** se preporučuje **proizvod** (Jedan korisnik može imati više preporuka, dok jedna preporuka može biti kod više korisnika) više na više.

Stvara se dodatni entitet **preporuceni\_proizvodi** gdje se prati razlog\_preporuke, a primarni ključevi entiteta proizvodi i korisnici se referenciraju kao strani ključ u preporuceni\_proizvodi.

**Korisnik** otvara poruku za **podršku** (Jedan korisnik može otvoriti više poruka za podršku, dok jedna poruka je otvorena od samo jednog korisnika) jedan na više

**Korisnik** sprema **proizvode** u košaricu (Jedan korisnik može spremiti više proizvoda u košaricu, dok jedan proizvod može biti spremljen kod više korisnika) više na više

Stvara se dodatni entitet **kosarica** gdje je prati količina, a primarni ključevi entiteta proizvodi i korisnici se referenciraju kao strani ključ u kosarica.

# Kombinacija shema

**kategorije\_proizvoda**(id, naziv, opis)

**proizvodi**(id, naziv, opis, cijena, kategorija\_id, kolicina\_na\_skladistu, slika, specifikacije, datum\_kreiranja)

**korisnici**(id, ime, prezime, email, lozinka, adresa, grad, telefon, tip\_korisnika, datum\_registracije)

**nacini\_isporuke**(id, naziv, opis, cijena, trajanje) PROMIJENITI TIP ZA trajanje u INTEGER

**kuponi**(id, kod, postotak\_popusta, datum\_pocetka, datum\_zavrsetka, max\_iskoristenja)

**narudzbe**(id, korisnik\_id, datum\_narudzbe, status\_narudzbe, ukupna\_cijena, nacin\_isporuke\_id, kupon\_id)

**stavke\_narudzbe**(id, narudzba\_id, proizvod\_id, kolicina)

**placanja**(id, narudzba\_id, nacin\_placanja, iznos, datum\_placanja)

**racuni**(id, korisnik\_id, narudzba\_id, iznos, datum\_izdavanja)

**povrati\_proizvoda**(id, narudzba\_id, datum\_povrata, razlog, status\_povrata)

**popusti**(id, proizvod\_id, postotak\_popusta, datum\_pocetka, datum\_zavrsetka)

**recenzije\_proizvoda**(id, proizvod\_id, korisnik\_id, ocjena, komentar, datum\_recenzije)

**wishlist**(id, korisnik\_id, proizvod\_id)

**pracenje\_isporuka**(id, narudzba\_id, status\_isporuke, datum\_isporuke)

**preporuceni\_proizvodi**(id, korisnik\_id, proizvod\_id, razlog\_preporuke)

**podrska\_za\_korisnike**(id, korisnik\_id, tema, poruka, status, datum\_upita, datum\_odgovora)

**kosarica**(id, korisnik\_id, proizvod\_id, kolicina)

# EER DIJAGRAM (MYSQL WORKBENCH)

# Tablice

## tablice Loren

**CREATE TABLE** kategorije\_proizvoda (

id **INT AUTO\_INCREMENT PRIMARY KEY,**

naziv **VARCHAR(255) NOT NULL UNIQUE,**

opis **TEXT NOT NULL**

);

Ova tablica se naziva „kategorije\_proizvoda“ i sadrži sljedeće atribute:

* „id“ je cjelobrojni atribut koji se automatski povećava svaki put kada se unese novi red u tablicu. Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svake kategorije.
* „naziv“ je tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova. Sadrži naziv kategorije i mora biti jedinstven kako bi se spriječilo dupliciranje naziva kategorija.
* „opis“ je tekstualni atribut (TEXT) koji omogućuje pohranu detaljnog opisa kategorije.

Tablica također ima sljedeća ograničenja:

* atribut „id“ postavljen je kao primarni ključ tablice.
* atribut „naziv“ mora sadržavati jedinstvene vrijednosti, čime se osigurava da svaka kategorija ima jedinstveno ime.

**CREATE TABLE** proizvodi (

id **INT AUTO\_INCREMENT PRIMARY KEY,**

naziv **VARCHAR(255) NOT NULL,**

opis **TEXT NOT NULL,**

cijena **DECIMAL(10, 2) NOT NULL,**

kategorija\_id **INT NOT NULL,**

kolicina\_na\_skladistu **INT NOT NULL,**

slika **VARCHAR(255),**

specifikacije **TEXT,**

datum\_kreiranja **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**CONSTRAINT** chk\_cijena\_proizvodi **CHECK (cijena > 0),**

**CONSTRAINT** chk\_kolicina\_proizvodi **CHECK (kolicina\_na\_skladistu >= 0),**

**CONSTRAINT** fk\_kategorija\_proizvoda\_proizvodi **FOREIGN KEY** (kategorija\_id) **REFERENCES** kategorije\_proizvoda(id)

);

Ova tablica se naziva „proizvodi“ i sadrži sljedeće atribute:

* „id“ je cjelobrojni atribut koji se automatski povećava svaki put kada se unese novi red u tablicu. Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svakog proizvoda.
* „naziv“ je tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova. Sadrži naziv proizvoda.
* „opis“ je tekstualni atribut (TEXT) koji sadrži detaljan opis proizvoda.
* „cijena“ je decimalni atribut (DECIMAL(10, 2)) koji bilježi cijenu proizvoda s do dvije decimalne znamenke.
* „kategorija\_id“ je cjelobrojni atribut koji se odnosi na primarni ključ u tablici kategorije\_proizvoda. Označava kojoj kategoriji proizvod pripada.
* „kolicina\_na\_skladistu“ je cjelobrojni atribut koji prikazuje broj dostupnih proizvoda na skladištu.
* „slika“ je tekstualni atribut (VARCHAR) koji sadrži URL ili putanju do slike proizvoda.
* „specifikacije“ je tekstualni atribut (TEXT) koji pohranjuje dodatne informacije o proizvodu, poput tehničkih specifikacija.
* „datum\_kreiranja“ je vrijeme kada je proizvod kreiran, s podrazumijevanom vrijednošću trenutnog vremena (CURRENT\_TIMESTAMP).

Tablica također ima sljedeća ograničenja:

* atribut „id“ postavljen je kao primarni ključ.
* atribut „kategorija\_id“ povezan je s tablicom „kategorije\_proizvoda“, čime se osigurava referencijalna integracija.
* atribut „cijena“ mora biti veća od 0, što je osigurano ograničenjem CHECK.
* atribut „kolicina\_na\_skladistu“ mora biti veća ili jednaka 0, što je osigurano ograničenjem CHECK.

**CREATE TABLE** recenzije\_proizvoda (

id **INT AUTO\_INCREMENT PRIMARY KEY,**

proizvod\_id **INT NOT NULL,**

korisnik\_id **INT NOT NULL,**

ocjena **INT NOT NULL,**

komentar **TEXT,**

datum\_recenzije **TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**,

**CONSTRAINT** chk\_ocjena\_recenzije\_proizvoda **CHECK** (ocjena **BETWEEN** 1 **AND** 5),

**CONSTRAINT** fk\_proizvod\_recenzija\_recenzije\_proizvoda **FOREIGN KEY** (proizvod\_id) **REFERENCES** proizvodi(id),

**CONSTRAINT** fk\_korisnik\_recenzija\_recenzije\_proizvoda **FOREIGN KEY** (korisnik\_id) **REFERENCES** korisnici(id) );

Ova tablica se naziva „recenzije\_proizvoda“ i sadrži sljedeće atribute:

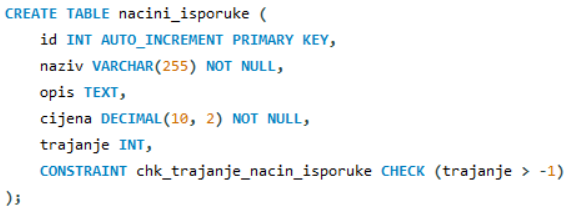
* „id“ je cjelobrojni atribut koji se automatski povećava svaki put kada se unese novi red u tablicu. Služi kao primarni ključ i osigurava jedinstvenu identifikaciju svake recenzije.
* „proizvod\_id“ je cjelobrojni atribut koji se odnosi na primarni ključ u tablici „proizvodi“. Označava proizvod na koji se odnosi recenzija.
* „korisnik\_id“ je cjelobrojni atribut koji se odnosi na primarni ključ u tablici „korisnici“. Označava korisnika koji je napisao recenziju.
* „ocjena“ je cjelobrojni atribut koji pohranjuje ocjenu proizvoda.
* „komentar“ je tekstualni atribut (TEXT) koji omogućuje korisnicima da napišu svoje mišljenje o proizvodu.
* „datum\_recenzije“ je vrijeme kada je recenzija napisana, s podrazumijevanom vrijednošću trenutnog vremena (CURRENT\_TIMESTAMP).

Tablica također ima sljedeća ograničenja:

* Atribut „id“ postavljen je kao primarni ključ.
* Vrijednost atributa „ocjena“ mora biti između 1 i 5, što je osigurano ograničenjem CHECK.
* Atribut „proizvod\_id“ povezan je s tablicom „proizvodi“, čime se osigurava referencijalna integracija za proizvode.
* Atribut „korisnik\_id“ povezan je s tablicom „korisnici“, čime se osigurava referencijalna integracija za korisnike.

## tablice Morena

## tablice Bruno

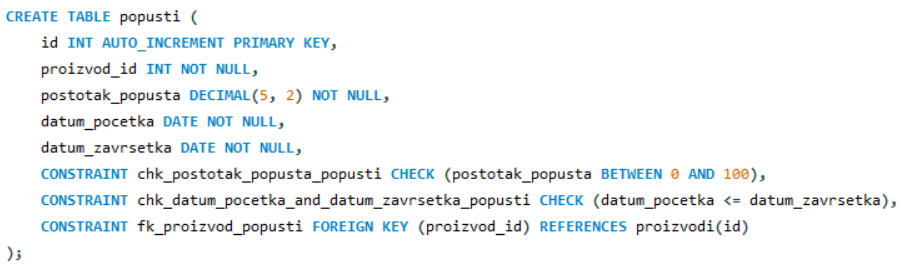


Ova tablica se naziva **nacini\_isporuke** i služi za pohranjivanje informacije o različitim načinima isporuke narudžbi. Sadrži sljedeće atribute:

* „id“ - cjelobrojni atribut koji se automatski povećava svaki put kad se napravi novi unos u tablicu. Osigurava da svaki način isporuke ima jedinstvenu identifikaciju.
* „naziv“ - tekstualni atribut (VARCHAR) s maksimalnom duljinom znakova od 255. Sadrži naziv načina isporuke.
* „opis“ - tekstualni atribut (TEXT) koji sadrži detaljniji opis načina isporuke.
* „cijena“ - decimalni atribut (DECIMAL(10,2)) koji sadržava cijenu načina isporuke na 2 decimale, do maksimalno 10 znamenki.
* „trajanje“ - cjelobrojni atribut koji sadržava maksimalno trajanje isporuke.

Također, tablica sadrži sljedeća ograničenja:

* NOT NULL – ograničenje za atribute „naziv“ i „cijena“, osigurava da atributi ne smiju biti prazni.
* CHECK – ograničenje za atribut „trajanje“, osigurava da trajanje isporuke ne smije biti negativno.



Ova tablica se naziva **popusti** i služi za pohranjivanje informacije o popustima za određene proizvode, kao i njihove datume dostupnosti.

Sadrži sljedeće atribute:

* „id“ – cjelobrojni atribut koji osigurava jedinstvenu identifikaciju popusta, te se automatski povećava svakim novim unosom u tablicu.
* „proizvod\_id“ – strani ključ koji referencira na koji proizvod se popust primjenjuje.
* „postotak\_popusta“ – decimalni atribut koji sadrži podatak o iznosu popusta u postotcima, izražava se na dvije decimale, a maksimalno može imati 5 znamenki.
* „datum\_pocetka“ – podatak tipa DATE, koji sadrži datum početka primjenjivanja popusta.
* „datum\_zavrsetka“ – podatak tipa DATE, koji sadrži datum završetka primjenjivanja popusta.

Također, tablica sadrži sljedeća ograničenja:

* CHECK – ograničenje za atribut „postotak\_popusta“ koje osigurava da je postotak između 0 i 100. Također, isto ograničenje za atribute „datum\_pocetka“ i „datum\_zavrsetka“ koje osigurava da je datum početka manji ili jednak datumu završetka.
* FOREIGN KEY – ograničenje pomoću kojeg atribut „proizvod\_id“ referenciramo na atribut „id“ iz tablice **proizvodi*.***

## tablice Fran

## tablice Josip

## tablice Leo

CREATE TABLE korisnici (

id INT AUTO\_INCREMENT PRIMARY KEY,

ime VARCHAR(255) NOT NULL,

prezime VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

lozinka VARCHAR(255) NOT NULL,

adresa TEXT,

grad VARCHAR(255),

telefon VARCHAR(20),

tip\_korisnika ENUM('kupac', 'admin') NOT NULL,

datum\_registracije DATE NOT NULL

);

Ova tablica se naziva **„korisnici“** i sadrži sljedeće atribute:

1. **„id“**
   * Cjelobrojni atribut koji se automatski povećava (AUTO\_INCREMENT) svaki put kada se unese novi red u tablicu.
   * Služi kao **primarni ključ** i osigurava jedinstvenu identifikaciju svakog korisnika.
2. **„ime“**
   * Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova.
   * Sadrži ime korisnika.
3. **„prezime“**
   * Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova.
   * Sadrži prezime korisnika.
4. **„email“**
   * Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova.
   * Mora biti jedinstven (**UNIQUE**) i služi za identifikaciju korisnika u sustavu.
5. **„lozinka“**
   * Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova.
   * Pohranjuje lozinku korisnika (obično hashiranu radi sigurnosti).
6. **„adresa“**
   * Tekstualni atribut (TEXT).
   * Sadrži fizičku adresu korisnika (ulica, broj i slično).
7. **„grad“**
   * Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 255 znakova.
   * Sadrži naziv grada korisnika.
8. **„telefon“**
   * Tekstualni atribut (VARCHAR) s maksimalnom duljinom od 20 znakova.
   * Pohranjuje telefonski broj korisnika.
9. **„tip\_korisnika“**
   * Enumerirani atribut (**ENUM**) s mogućim vrijednostima „kupac“ i „admin“.
   * Označava vrstu korisnika (npr. obični kupac ili administrator sustava).
10. **„datum\_registracije“**
    * Atribut tipa **DATE**.
    * Pohranjuje datum kada je korisnik kreirao račun u sustavu.

### Ograničenja tablice:

* **Primarni ključ:**Atribut **„id“** postavljen je kao primarni ključ tablice, osiguravajući jedinstvenost svakog korisnika.
* **Jedinstveni email:**Atribut **„email“** mora biti jedinstven kako bi se izbjeglo dupliciranje korisnika s istim mailom.
* **Obavezni atributi:**Atributi **„ime“**, **„prezime“**, **„email“**, **„lozinka“**, **„tip\_korisnika“**, i **„datum\_registracije“** moraju biti popunjeni prilikom unosa (NOT NULL).

CREATE TABLE wishlist (

id INT AUTO\_INCREMENT PRIMARY KEY,

korisnik\_id INT NOT NULL,

proizvod\_id INT NOT NULL,

FOREIGN KEY (korisnik\_id) REFERENCES korisnici(id),

FOREIGN KEY (proizvod\_id) REFERENCES proizvodi(id)

);

Ova tablica se naziva **„wishlist“** i sadrži sljedeće atribute:

1. **„id“**
   * Cjelobrojni atribut koji se automatski povećava (**AUTO\_INCREMENT**) svaki put kada se unese novi red u tablicu.
   * Služi kao **primarni ključ** i osigurava jedinstvenu identifikaciju svakog unosa u „wishlist“.
2. **„korisnik\_id“**
   * Cjelobrojni atribut koji se odnosi na primarni ključ u tablici **„korisnici“**.
   * Predstavlja referencu na korisnika koji je dodao proizvod u svoju „wishlist“.
3. **„proizvod\_id“**
   * Cjelobrojni atribut koji se odnosi na primarni ključ u tablici **„proizvodi“**.
   * Predstavlja referencu na proizvod koji je korisnik dodao u svoju „wishlist“.

### Ograničenja tablice:

* **Primarni ključ:**Atribut **„id“** postavljen je kao primarni ključ, osiguravajući jedinstvenost svakog unosa u „wishlist“.
* **Vanjski ključ „korisnik\_id“:**Atribut **„korisnik\_id“** povezan je s atributom **„id“** iz tablice **„korisnici“**, čime se osigurava referencijalna integracija između korisnika i njihove „wishlist“.
* **Vanjski ključ „proizvod\_id“:**Atribut **„proizvod\_id“** povezan je s atributom **„id“** iz tablice **„proizvodi“**, čime se osigurava referencijalna integracija između proizvoda i „wishlist“.

### Opis funkcionalnosti:

Tablica **„wishlist“** omogućava povezivanje korisnika i proizvoda koje su dodali u svoju „wishlist“. Svaki unos u tablicu predstavlja proizvod koji je određeni korisnik odabrao za „wishlist“.  
Na primjer:

* Korisnik s **„korisnik\_id“** može imati više proizvoda u svojoj „wishlist“, dok svaki proizvod može biti dodan u „wishlist“ više korisnika.
* Relacije između korisnika i proizvoda omogućene su pomoću vanjskih ključeva.

### Napomena:

* Ograničenja stranih ključeva osiguravaju referencijalnu integraciju. Na primjer, proizvod ne može biti dodan u „wishlist“ ako ne postoji u tablici **„proizvodi“**, i „wishlist“ unos se ne može kreirati za korisnika koji ne postoji u tablici **„korisnici“**.

CREATE TABLE preporuceni\_proizvodi (

id INT AUTO\_INCREMENT PRIMARY KEY,

korisnik\_id INT NOT NULL,

proizvod\_id INT NOT NULL,

razlog\_preporuke TEXT,

FOREIGN KEY (korisnik\_id) REFERENCES korisnici(id),

FOREIGN KEY (proizvod\_id) REFERENCES proizvodi(id)

);

Ova tablica se naziva **„preporuceni\_proizvodi“** i sadrži sljedeće atribute:

1. **„id“**
   * Cjelobrojni atribut koji se automatski povećava (**AUTO\_INCREMENT**) svaki put kada se unese novi red u tablicu.
   * Služi kao **primarni ključ** i osigurava jedinstvenu identifikaciju svake preporuke proizvoda.
2. **„korisnik\_id“**
   * Cjelobrojni atribut koji se odnosi na primarni ključ u tablici **„korisnici“**.
   * Predstavlja korisnika koji je preporučio određeni proizvod.
3. **„proizvod\_id“**
   * Cjelobrojni atribut koji se odnosi na primarni ključ u tablici **„proizvodi“**.
   * Predstavlja proizvod koji je preporučen.
4. **„razlog\_preporuke“**
   * Tekstualni atribut (**TEXT**) koji pohranjuje opis ili objašnjenje zašto je korisnik preporučio određeni proizvod.

### Ograničenja tablice:

* **Primarni ključ:**Atribut **„id“** postavljen je kao primarni ključ, osiguravajući jedinstvenost svake preporuke.
* **Vanjski ključ „korisnik\_id“:**Atribut **„korisnik\_id“** povezan je s atributom **„id“** iz tablice **„korisnici“**, čime se osigurava referencijalna integracija između korisnika i njihovih preporuka.
* **Vanjski ključ „proizvod\_id“:**Atribut **„proizvod\_id“** povezan je s atributom **„id“** iz tablice **„proizvodi“**, čime se osigurava referencijalna integracija između preporuka i proizvoda.

### Opis funkcionalnosti:

Tablica **„preporuceni\_proizvodi“** omogućava evidentiranje preporuka proizvoda koje su korisnici predložili, uključujući dodatne razloge zašto je određeni proizvod preporučen.  
Na primjer:

* Jedan korisnik može preporučiti više proizvoda.
* Više korisnika može preporučiti isti proizvod, svaki s različitim razlozima.

### Primjeri upotrebe:

1. **Dodavanje preporuke proizvoda:**
   * Kada korisnik preporuči proizvod, unosi se novi red s njegovim korisničkim ID-jem, ID-jem proizvoda i opisom razloga preporuke.
2. **Prikaz preporuka:**
   * Pomoću relacija između tablica moguće je prikazati popis preporučenih proizvoda za korisnika ili prikazati proizvode koje su korisnici najčešće preporučivali.

### Napomena:

* Ograničenja stranih ključeva osiguravaju referencijalnu integraciju. Na primjer, preporuka ne može biti kreirana za korisnika ili proizvod koji ne postoji u bazi podataka.
* Polje **„razlog\_preporuke“** omogućava fleksibilno objašnjenje razloga za preporuku. Ako nije potrebno, može ostati prazno.

# Upiti i pogledi

## Upiti i pogledi Loren

**Pregled svih proizvoda s kategorijama**

**CREATE VIEW** svi\_proizvodi\_s\_kategorijama **AS**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod,

p.cijena,

k.naziv **AS** kategorija,

p.kolicina\_na\_skladistu

**FROM** proizvodi p

**JOIN** kategorije\_proizvoda k **ON** p.kategorija\_id = k.id;

Pogled „svi\_proizvodi\_s\_kategorijama“ kombinira podatke iz tablica „proizvodi“ i „kategorije\_proizvoda“ na temelju zajedničkog atributa „kategorija\_id“. Ovaj pogled pruža način za pregled svih proizvoda zajedno s pripadajućim kategorijama. Stupci koji se prikazuju za ovaj pogled su „proizvod\_id“ koji prikazuje jedinstveni identifikator proizvoda iz tablice „proizvodi“, „proizvod“ koji sadrži naziv proizvoda, „cijena“ sa cijenom proizvoda, „kategorija“ sa nazivom kategorije kojoj proizvod pripada i „kolicina\_na\_skladistu“ koja ima trenutno dostupnu količinu proizvoda na skladištu. Pogled omogućava korisnicima pregled informacija o proizvodima i njihovim kategorijama kao da je riječ o jednoj tablici.

**Pregled proizvoda koji su trenutno na skladištu**

**CREATE VIEW** dostupni\_proizvodi **AS**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod,

p.kolicina\_na\_skladistu

**FROM** proizvodi p

**WHERE** p.kolicina\_na\_skladistu > 0;

Ovaj pogled pod imenom „dostupni\_proizvodi“ filtrira proizvode iz tablice „proizvodi“ koji imaju pozitivnu količinu na skladištu, odnosno proizvode koji su trenutno dostupni za prodaju. Atributi koji se prikazuju u ovom pogledu su „proizvod\_id“ koji sadrži jedinstveni identifikator proizvoda, „proizvod“ s nazivom proizvoda, „kolicina\_na\_skladistu“ s dostupnom količinom proizvoda na skladištu. Pogled se koristi za dohvat samo onih proizvoda koji su odmah dostupni za kupnju.

**Pregled svih recenzija s podacima o proizvodima i korisnicima**

**CREATE VIEW** recenzije\_s\_proizvodima\_i\_korisnicima **AS**

**SELECT**

r.id **AS** recenzija\_id,

p.naziv **AS** proizvod,

k.ime **AS** korisnik\_ime,

r.ocjena,

r.komentar,

r.datum\_recenzije

**FROM** recenzije\_proizvoda r

**JOIN** proizvodi p **ON** r.proizvod\_id = p.id

**JOIN** korisnici k **ON** r.korisnik\_id = k.id;

Pogled s nazivom „recenzije\_s\_proizvodima\_i\_korisnicima“ povezuje podatke iz tablica „recenzije\_proizvoda“, „proizvodi“ i „korisnici“ kako bi omogućio pregled recenzija zajedno s informacijama o proizvodima i korisnicima koji su ih napisali. Atributi koji se prikazuju u pogledu su „recenzija\_id“ s jedinstvenim identifikatorom recenzije, „proizvod“ sa nazivom proizvoda koji je recenziran, „ime“ s imenom korisnika koji je napisao recenziju, „ocjena“ s ocjenom proizvoda (vrijednost od 1 do 5), „komentar“ koji sadrži komentar korisnika o proizvodu i „datum\_recenzije“ koji prikazuje datum kada je recenzija napisana. Pogled se koristi za dohvat podataka o recenzijama proizvoda s detaljnim informacijama o proizvodima i korisnicima.

**Pregled najpopularnijih proizvoda (najveći broj recenzija)**

**CREATE VIEW** najpopularniji\_proizvodi **AS**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod,

**COUNT**(r.id) AS broj\_recenzija

**FROM** proizvodi p

**LEFT JOIN** recenzije\_proizvoda r **ON** p.id = r.proizvod\_id

**GROUP BY** p.id, p.naziv

**ORDER BY** broj\_recenzija **DESC**;

Pogled „najpopularniji\_proizvodi“ i prikazuje proizvode rangirane prema broju recenzija. Kombinira podatke iz tablica „proizvodi“ i „recenzije\_proizvoda“ koristeći LEFT JOIN kako bi uključio sve proizvode, čak i one bez recenzije. Stupci koji se prikazuju u pogledu su „proizvod\_id“ sa jedinstvenim identifikatorom proizvoda, „proizvod“ sa nazivom proizvoda, „broj\_recenzija“ koji sadži broj recenzija za svaki proizvod. Proizvodi su sortirani prema broju recenzija u opadajućem redoslijedu, tako da budu najpopularnijih proizvodi na samom vrhu pogleda.

**Pogled za proizvode koji su na popustu**

**CREATE VIEW** proizvodi\_na\_popustu **AS**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod\_naziv,

p.opis **AS** proizvod\_opis,

p.cijena **AS** originalna\_cijena,

(p.cijena - (p.cijena \* pop.postotak\_popusta / 100)) **AS** cijena\_sa\_popustom,

pop.postotak\_popusta,

pop.datum\_pocetka,

pop.datum\_zavrsetka

**FROM**

proizvodi p

**JOIN**

popusti pop ON p.id = pop.proizvod\_id

**WHERE**

**CURRENT\_DATE BETWEEN** pop.datum\_pocetka **AND** pop.datum\_zavrsetka

**ORDER BY**

pop.postotak\_popusta **DESC**;

Pogled „proizvodi\_na\_popustu“ prikazuje trenutačno aktivne popuste na proizvode, rangirane prema visini popusta. Pogled kombinira podatke iz tablica „proizvodi“ i „popusti“ koristeći „INNER JOIN“, čime uključuje samo one proizvode koji imaju aktivan popust u danom trenutku. Stupci koji se prikazuju u pogledu su „proizvod\_id“ sa jedinstvenim identifikatorom proizvoda, „proizvod“ sa nazivom proizvoda, „proizvod\_opis“ koji sadrži opis za svaki proizvod, „originalna\_cijena“ koja prikazuje cijenu bez popusta, „cijena\_sa\_popustom“ koja prikazuje cijenu s popustom, „postotak\_popusta“ s postotkom popusta i datumio za početak i završetak aktivnosti popusta. Filtrirano je tako da se prikazuju samo proizvodi koji su u datumu stvaranja između datuma početka i završetka popusta a zatim su sortirani u silaznom nizu po postotku.

**Upit proizvoda koji su dostupni na skladištu, imaju više od 10 recenzija i prosječnu ocjenu iznad 4**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod,

p.kolicina\_na\_skladistu,

**COUNT**(r.id) **AS** broj\_recenzija,

**AVG**(r.ocjena) **AS** prosjecna\_ocjena

**FROM** proizvodi p

**LEFT JOIN** recenzije\_proizvoda r **ON** p.id = r.proizvod\_id

**WHERE** p.kolicina\_na\_skladistu > 0

**GROUP BY** p.id, p.naziv, p.kolicina\_na\_skladistu

**HAVING** broj\_recenzija > 10 **AND** prosjecna\_ocjena > 4;

Ovaj upit dohvaća proizvode koji su dostupni na skladištu i koji imaju više od 10 recenzija s prosječnom ocjenom većom od 4. Podaci se kombiniraju iz tablica „proizvodi“ i „recenzije\_proizvoda“. Podaci se grupiraju prema svakom proizvodu, a rezultati su filtrirani s „HAVING“, gdje su uključeni samo proizvodi s više od 10 recenzija i prosječnom ocjenom većom od 4.

**Upit proizvoda s najvećom i najmanjom cijenom po kategoriji**

**SELECT**

k.naziv **AS** kategorija,

**MAX**(p.cijena) **AS** najskuplji\_proizvod,

**MIN**(p.cijena) **AS** najjeftiniji\_proizvod

**FROM** proizvodi p

**JOIN** kategorije\_proizvoda k **ON** p.kategorija\_id = k.id

**GROUP BY** k.naziv;

Ovaj upit pronalazi proizvode s najvećom i najmanjom cijenom unutar svake kategorije. Tablice „proizvodi“ i „kategorije\_proizvoda“ povezane su preko atributa „kategorija\_id“. Podaci se grupiraju po kategorijama, a funkcije agregacije „MAX“ i „MIN“ koriste se za izračunavanje maksimalnih i minimalnih cijena.

**Upit za korisnike koji su napisali najviše recenzija i njihove prosječne ocjene**

**SELECT**

k.id **AS** korisnik\_id,

k.ime,

k.prezime,

**COUNT**(r.id) **AS** broj\_recenzija,

**AVG**(r.ocjena) **AS** prosjecna\_ocjena

**FROM** korisnici k

**JOIN** recenzije\_proizvoda r **ON** k.id = r.korisnik\_id

**GROUP BY** k.id, k.ime, k.prezime

**ORDER BY** broj\_recenzija **DESC**

**LIMIT** 5;

Ovaj upit pronalazi korisnike koji su napisali najviše recenzija, zajedno s prosječnim ocjenama koje su dali. Podaci se kombiniraju iz tablica „korisnici“ i „recenzije\_proizvoda“. Podaci se grupiraju prema korisnicima, a rezultati su sortirani prema broju recenzija u silaznom redoslijedu. Prikazuju se podaci za najviše 5 korisnika pomoću ograničenja „LIMIT 5“.

**Upit za prosječne ocjene proizvoda na temelju recenzija**

**SELECT**

p.id **AS** proizvod\_id,

p.naziv **AS** proizvod\_naziv,

p.opis **AS** proizvod\_opis,

**AVG**(r.ocjena) **AS** prosjecna\_ocjena,

**COUNT**(r.id) **AS** broj\_recenzija

**FROM**

proizvodi p

**LEFT JOIN**

recenzije\_proizvoda r **ON** p.id = r.proizvod\_id

**GROUP BY**

p.id, p.naziv, p.opis

**ORDER BY**

prosjecna\_ocjena **DESC**;

Ovaj upit pronalazi prosječnu ocjenu za sve proizvode obzirom na recenzije koje su korisnici napisali. Podaci se kombiniraju iz tablica „korisnici“ i „recenzije\_proizvoda“. „LEFT JOIN“ osigurava da se u rezultat uključe svi proizvodi, čak i oni koji nemaju nijednu recenziju. Podaci se grupiraju prema „p.id“ , „p.naziv“, „p.opis“, kako bi se za svaki proizvod odvojeno izračunala prosječna ocjena i broj recenzija. Prikazuju se podaci koji su sortirani silazno tako da imamo prosječno najbolje ocjenjene proizvode prikazane prve.

## Upiti i pogledi Morena

## Upiti i pogledi Bruno

## Upiti i pogledi Fran

## Upiti i pogledi Josip

## Upiti i pogledi Leo

CREATE VIEW profil\_korisnika AS

SELECT

id AS korisnik\_id,

ime,

prezime,

email,

adresa,

grad,

telefon,

datum\_registracije

FROM korisnici;

### Pogled: Profil Korisnika

Pogled **„profil\_korisnika“** koristi se za dohvaćanje osnovnih informacija o korisnicima iz tablice **„korisnici“**. Služi za pojednostavljenje upita koji prikazuju korisničke profile, omogućavajući aplikaciji da lako prikaže informacije poput imena, prezimena, email adrese, adrese stanovanja, grada, telefona i datuma registracije.

* **Zašto se koristi ovaj pogled?**Umesto ponavljanja složenih upita u različitim delovima aplikacije, pogled pruža strukturiran i uvek ažuran prikaz korisničkih podataka. To čini aplikaciju efikasnijom i čitljivom.
* **Specifične prednosti:**Podaci se preuzimaju direktno u obliku koji je potreban aplikaciji, smanjujući potrebu za dodatnim manipulacijama na strani aplikacijskog koda.

-- Pogled: Preporuke drugih (Leo)

CREATE VIEW preporuke\_drugih AS

SELECT

k.id AS korisnik\_id,

k.ime,

k.prezime,

p.naziv AS proizvod\_naziv,

pp.razlog\_preporuke

FROM preporuceni\_proizvodi pp

JOIN korisnici k ON pp.korisnik\_id = k.id

JOIN proizvodi p ON pp.proizvod\_id = p.id;

### Pogled: Preporuke drugih

Pogled **„preporuke\_drugih“** koristi se za dohvaćanje informacija o proizvodima koje su preporučili drugi korisnici u aplikaciji. Pomoću ovog pogleda, aplikacija može jednostavno i efikasno prikazati proizvode koje su drugi korisnici preporučili, zajedno sa osnovnim informacijama o korisniku koji je dao preporuku, kao što su ime, prezime i razlog preporuke.

#### Zašto se koristi ovaj pogled?

Korištenje ovog pogleda omogućava jednostavno i brzo dohvaćanje svih potrebnih podataka o preporukama drugih korisnika bez potrebe za ponavljanjem složenih upita u različitim dijelovima aplikacije. Umjesto da svaki put radimo join između tablica korisnici, preporuceni\_proizvodi i proizvodi u aplikaciji, ovaj pogled centralizira upit i omogućava jednostavno preuzimanje relevantnih podataka u obliku koji je direktno potreban.

#### Specifične prednosti:

* **Pojednostavljenje upita**: Ovaj pogled omogućava aplikaciji da dobije sve potrebne informacije o preporukama drugih korisnika u samo jednom upitu, što poboljšava efikasnost i smanjuje dupliciranje koda.
* **Brzi pristup podacima**: Podaci o korisnicima i njihovim preporukama dostupni su odmah, što omogućava brže učitavanje stranica i bolji korisnički doživljaj.
* **Ažurnost podataka**: Pogled je uvijek ažuran jer se podaci iz baze povlače u trenutnom stanju, što znači da preporuke drugih korisnika neće biti zastarjele.

-- Pogled: Popularni proizvodi (najviše puta dodati u wishlist) (Leo)

CREATE OR REPLACE VIEW popularni\_proizvodi AS

SELECT

p.id AS proizvod\_id,

p.naziv,

COUNT(w.proizvod\_id) AS broj\_dodavanja

FROM proizvodi p

LEFT JOIN wishlist w ON p.id = w.proizvod\_id

GROUP BY p.id, p.naziv

ORDER BY broj\_dodavanja DESC;

### Pogled: Popularni Proizvodi

Pogled **„popularni\_proizvodi“** koristi se za identifikaciju proizvoda koji su najpopularniji među korisnicima prema broju dodavanja na **wishlist**. Kombinuje podatke iz tablica **„proizvodi“** i **„wishlist“** pomoću **LEFT JOIN**.

* **Zašto koristiti LEFT JOIN?  
  LEFT JOIN** osigurava da svi proizvodi iz tablice **„proizvodi“** budu uključeni u rezultat, čak i ako nijedan korisnik nije dodao taj proizvod na wishlist. Ovo je korisno za pružanje sveobuhvatne analize popularnosti proizvoda.
* **Šta omogućava ovaj pogled?**Funkcija **COUNT** broji koliko puta je svaki proizvod dodat na wishlist, dok **GROUP BY** grupiše rezultate po proizvodima. **ORDER BY** sortira proizvode po popularnosti u opadajućem redoslijedu, tako da se najpopularniji proizvodi nalaze na vrhu.
* **Prednosti:**Pogled omogućava brzi uvid u popularnost proizvoda i olakšava donošenje odluka, kao što su kreiranje marketinških kampanja ili ažuriranje ponude na osnovu popularnosti.

-- Pogled: Narudžbe Korisnika (Leo)

CREATE VIEW narudzbe\_korisnika AS

SELECT

n.id AS narudzba\_id,

n.datum\_narudzbe,

n.status\_narudzbe AS status,

s.proizvod\_id,

p.naziv AS proizvod\_naziv,

s.kolicina,

s.kolicina \* p.cijena AS ukupna\_cijena\_proizvoda,

n.korisnik\_id

FROM narudzbe n

JOIN stavke\_narudzbe s ON n.id = s.narudzba\_id

JOIN proizvodi p ON s.proizvod\_id = p.id;

### Pogled: Narudžbe Korisnika

Pogled **„narudzbe\_korisnika“** omogućava pregled narudžbi korisnika sa svim relevantnim detaljima, uključujući povezane proizvode i njihove troškove. Povezuje tablice **„narudzbe“**, **„stavke\_narudzbe“** i **„proizvodi“** pomoću **INNER JOIN**.

* **Zašto koristiti INNER JOIN?  
  INNER JOIN** osigurava da se u rezultate uključuju samo narudžbe koje imaju validne stavke i proizvode, čime se izbegava prikazivanje nepotpunih ili nevažećih podataka.
* **Kako funkcionira ovaj pogled?**
  + Informacije o narudžbama, kao što su datum, status i ukupni troškovi, dolaze iz tablice **„narudzbe“**.
  + Pojedinačne stavke narudžbi povezane su preko tablice **„stavke\_narudzbe“**, dok su podaci o proizvodima dohvaćeni iz tablice **„proizvodi“**.
  + Ukupna cena svake stavke računa se dinamički množenjem količine proizvoda sa njegovom cijenom.
* **Prednosti:**Pogled pruža sveobuhvatne informacije o narudžbama korisnika, što omogućava transparentan pregled narudžbi unutar korisničkog profila.

-- Upit: Provjera popularnosti proizvoda (koji proizvodi su najčešće dodati u wishlist) (Leo)

SELECT

p.id AS proizvod\_id,

p.naziv AS proizvod\_naziv,

COUNT(w.proizvod\_id) AS broj\_dodavanja

FROM proizvodi p

LEFT JOIN wishlist w ON p.id = w.proizvod\_id

GROUP BY p.id, p.naziv

ORDER BY broj\_dodavanja DESC

LIMIT 10; -- Dohvata prvih 10 proizvoda

**Upit: Provera popularnosti proizvoda (koji proizvodi su najčešće dodati u wishlist)**

**Opis:**Ovaj upit koristi se za identifikaciju proizvoda koji su najpopularniji među korisnicima na osnovu broja dodavanja u **wishlist**. Upit kombinuje podatke iz tablica **„proizvodi“** i **„wishlist“** pomoću **LEFT JOIN** kako bi uključio sve proizvode, čak i one koji nisu dodati na wishlist.

* **Detalji implementacije:**
  + **LEFT JOIN** omogućava da svi proizvodi budu uključeni u rezultat, čak i oni bez unosa u **wishlist**.
  + Funkcija **COUNT** računa broj puta koliko je svaki proizvod dodat u **wishlist**.
  + **GROUP BY** osigurava grupiranje rezultata prema jedinstvenim proizvodima.
  + **ORDER BY** sortira proizvode prema broju dodavanja u padajućem redoslijedu.
  + **LIMIT 10** ograničava rezultat na prvih 10 najpopularnijih proizvoda.

-- Upit: Provjera korisnika sa najviše narudžbi (Leo)

SELECT

k.id AS korisnik\_id,

k.ime,

k.prezime,

COUNT(n.id) AS broj\_narudzbi

FROM korisnici k

JOIN narudzbe n ON k.id = n.korisnik\_id

GROUP BY k.id, k.ime, k.prezime

ORDER BY broj\_narudzbi DESC

LIMIT 5; -- Prikazuje prvih 5 korisnika sa najviše narudžbi

**Upit: Provjera korisnika sa najviše narudžbi**

**Opis:**Ovaj upit omogućava identifikaciju korisnika koji su kreirali najviše narudžbi. Kombinuje podatke iz tablica **„korisnici“** i **„narudzbe“** pomoću **INNER JOIN** kako bi obezbedio da se uključe samo korisnici sa valjanim narudžbama.

* **Detalji implementacije:**
  + **INNER JOIN** osigurava povezivanje samo onih korisnika koji imaju najmanje jednu narudžbu.
  + Funkcija **COUNT** broji broj narudžbi za svakog korisnika.
  + **GROUP BY** grupiše rezultate prema jedinstvenim korisnicima.
  + **ORDER BY** sortira rezultate prema broju narudžbi u padajućem redoslijedu.
  + **LIMIT 5** ograničava prikaz na prvih 5 korisnika sa najviše narudžbi.

-- Upit: Ukupna zarada po korisnicima na osnovu narudžbi(Leo)

SELECT

k.id AS korisnik\_id,

k.ime,

k.prezime,

SUM(n.ukupna\_cijena) AS ukupna\_zarada

FROM korisnici k

JOIN narudzbe n ON k.id = n.korisnik\_id

GROUP BY k.id, k.ime, k.prezime

ORDER BY ukupna\_zarada DESC;

**Upit: Ukupna zarada po korisnicima na osnovu narudžbi**

**Opis:**Ovaj upit koristi se za izračunavanje ukupne zarade koju je svaki korisnik generisao na osnovu svojih narudžbi. Kombinuje podatke iz tablica **„korisnici“** i **„narudzbe“** pomoću **INNER JOIN**.

* **Detalji implementacije:**
  + **INNER JOIN** osigurava da se uzimaju u obzir samo korisnici koji imaju validne narudžbe.
  + Funkcija **SUM** računa ukupnu vrijednost svih narudžbi za svakog korisnika.
  + **GROUP BY** grupira rezultate prema jedinstvenim korisnicima.
  + **ORDER BY** sortira rezultate prema ukupnoj zaradi u padajućem redoslijedu

# Funkcije

## Funkcije Loren

**Funkcija koja vraća ukupnu vrijednost svih proizvoda na skladištu**

DELIMITER //

**CREATE FUNCTION** UkupnaVrijednostSkladista ()

**RETURNS DECIMAL**(15,2)

**DETERMINISTIC**

**BEGIN**

**DECLARE** ukupna\_vrijednost **DECIMAL**(15,2);

**SELECT** **SUM**(kolicina\_na\_skladistu \* cijena)

**INTO** ukupna\_vrijednost

**FROM** proizvodi;

**RETURN** ukupna\_vrijednost;

**END** //

DELIMITER ;

Funkcija „UkupnaVrijednostSkladista“ izračunava ukupnu vrijednost svih proizvoda na skladištu tako da množi cijenu svakog proizvoda s količinom na skladištu i zbroji sve rezultate. Prvo funkcija deklarira varijablu „ukupna\_vrijednost“ za spremanje ukupne vrijednosti. Zatim izračunava zbroj vrijednosti svih proizvoda koristeći „SUM“. I na kraju vraća rezultat tipa „DECIMAL“ s dvije decimale.

**Funkcija koja provjerava dostupnost proizvoda na skladištu**

DELIMITER //

**CREATE FUNCTION** provjeri\_dostupnost(

p\_proizvod\_id **INT**,

p\_kolicina **INT**

)

**RETURNS BOOLEAN**

**DETERMINISTIC**

**BEGIN**

**DECLARE** v\_kolicina\_na\_skladistu **INT**;

**SELECT** kolicina\_na\_skladistu

**INTO** v\_kolicina\_na\_skladistu

**FROM** proizvodi

**WHERE** id = p\_proizvod\_id;

**IF** v\_kolicina\_na\_skladistu **IS NULL THEN**

**RETURN FALSE**;

**END IF**;

**IF** p\_kolicina > v\_kolicina\_na\_skladistu **THEN**

**RETURN FALSE;**

**END IF**;

**RETURN TRUE**;

**END** //

DELIMITER ;

Funkcija „provjeri\_dostupnost“ služi za provjeru dostupnosti određenog proizvoda na skladištu. Prima dva parametra, „p\_proizvod\_id“, što je jedinstveni identifikator proizvoda, i „p\_kolicina“, što predstavlja željenu količinu tog proizvoda. Funkcija koristi tablicu **„**proizvodi“ kako bi dohvatila trenutnu količinu proizvoda na skladištu putem stupca „kolicina\_na\_skladistu“. Nakon što se podaci preuzmu, funkcija prvo provjerava postoji li proizvod s tim „id“. Ako proizvod ne postoji, funkcija vraća vrijednost „FALSE“, što označava da proizvod nije dostupan. Zatim funkcija uspoređuje traženu količinu s dostupnom količinom na skladištu. Ako tražena količina premašuje dostupnu količinu, funkcija također vraća „FALSE“, inače vraća „TRUE“, što znači da je tražena količina proizvoda dostupna na skladištu.

**Funkcija koja izračunava prosječne ocjene proizvoda na temelju recenzija.**

DELIMITER //

**CREATE FUNCTION** prosjecna\_ocjena\_proizvoda(

p\_proizvod\_id **INT**

)

**RETURNS DECIMAL**(3, 2)

**DETERMINISTIC**

**BEGIN**

**DECLARE** v\_prosjecna\_ocjena **DECIMAL**(3, 2);

**SELECT** **AVG**(ocjena)

**INTO** v\_prosjecna\_ocjena

**FROM** recenzije\_proizvoda

**WHERE** proizvod\_id = p\_proizvod\_id;

**RETURN** v\_prosjecna\_ocjena

**END** //

DELIMITER ;

Funkcija „prosjecna\_ocjena\_proizvoda“ izračunava prosječnu ocjenu proizvoda na temelju recenzija iz tablice „recenzije\_proizvoda“. Funkcija prima „p\_proizvod\_id“, koji predstavlja jedinstveni identifikator proizvoda čije recenzije se analiziraju. Unutar funkcije se deklarira varijabla „v\_prosjecna\_ocjena“ tipa „DECIMAL(3, 2)“, u koju se sprema rezultat izračuna prosječne ocjene, dobiven iz stupca „ocjena“ iz tablice „recenzije\_proizvoda“**.** Računa se prosjek svih ocjena za proizvod čiji identifikator odgovara „p\_proizvod\_id“. Nakon izvršenja upita, funkcija vraća izračunatu vrijednost, a ukoliko ne postoji nijedna recenzija vraća „NULL“.

## Funkcije Morena

## Funkcije Bruno

## Funkcije Fran

## Funkcije Josip

## Funkcije Leo

-- Funkcija: Formatiranje datuma (Leo)

DELIMITER //

CREATE FUNCTION formatiraj\_datum(input\_date DATE)

RETURNS VARCHAR(10)

DETERMINISTIC

BEGIN

RETURN DATE\_FORMAT(input\_date, '%d.%m.%Y');

END//

DELIMITER ;

Funkcija: Formatiranje datuma

Opis:  
Ova funkcija omogućava formatiranje datuma u željenom formatu „dan.mjesec.godina“. Namenjena je za upotrebu u raznim SQL upitima ili pogledima kako bi datumi bili prikazani u čitljivom formatu za korisnike.

* Detalji implementacije:
  + Funkcija koristi DATE\_FORMAT, ugrađenu MySQL funkciju, za prilagođavanje formata prikaza datuma.
  + Ulazni parametar input\_date očekuje vrijednost tipa DATE.
  + Vraća vrijednost tipa VARCHAR(10) sa formatom „dd.mm.yyyy“, gde:
    - dd predstavlja dan u mesecu,
    - mm predstavlja mesec u godini,
    - yyyy predstavlja četvorocifrenu godinu.
  + Funkcija je deklarisana kao DETERMINISTIC, što znači da će uvek vratiti isti izlaz za isti ulaz.

# Procedure

## Procedure Loren

**Procedura za dodavanje proizvoda s provjerom grešaka**

DELIMITER //

**CREATE PROCEDURE** DodajProizvod (

**IN** p\_naziv **VARCHAR**(255),

**IN** p\_opis **TEXT**,

**IN** p\_cijena **DECIMAL**(10, 2),

**IN** p\_kategorija\_id **INT**,

**IN** p\_kolicina **INT**,

**IN** p\_slika **VARCHAR**(255),

**IN** p\_specifikacije **TEXT**

)

**BEGIN**

**IF** p\_cijena <= 0 **THEN**

**SIGNAL SQLSTATE '45501'**

**SET MESSAGE\_TEXT** = 'Cijena mora biti veća od 0.';

**ELSEIF** p\_kolicina < 0 **THEN**

**SIGNAL SQLSTATE '45502'**

**SET MESSAGE\_TEXT** = 'Količina na skladištu ne može biti negativna.';

**ELSE**

**INSERT INTO** proizvodi (naziv, opis, cijena, kategorija\_id, kolicina\_na\_skladistu, slika, specifikacije)

**VALUES** (p\_naziv, p\_opis, p\_cijena, p\_kategorija\_id, p\_kolicina, p\_slika, p\_specifikacije);

**END IF**;

**END** //

DELIMITER ;

Procedura „DodajProizvod“ omogućava dodavanje proizvoda u tablicu „proizvodi“ uz provjeru valjanosti unosa. Prvo provjerava valjanost unosa tj. cijena mora biti veća od 0 i količina ne smije biti negativna. Ako je unos ispravan, proizvod se unosi u tablicu „proizvodi“. U slučaju neispravnog unosa, generira se greška pomoću „SIGNAL“.

**Procedura za brisanje proizvoda s provjerom povezanih podataka**

DELIMITER //

**CREATE PROCEDURE** ObrisiProizvod (

**IN** p\_proizvod\_id **INT**

)

**BEGIN**

**IF EXISTS** (

**SELECT** \*

**FROM** stavke\_narudzbe

**WHERE** proizvod\_id = p\_proizvod\_id

) **THEN**

**SIGNAL SQLSTATE '45503'**

**SET MESSAGE\_TEXT** = 'Proizvod ne može biti obrisan jer postoje povezane narudžbe.';

**ELSE**

**DELETE FROM** popusti

**WHERE** proizvod\_id = p\_proizvod\_id;

**DELETE FROM** recenzije\_proizvoda

**WHERE** proizvod\_id = p\_proizvod\_id;

**DELETE FROM** wishlist

**WHERE** proizvod\_id = p\_proizvod\_id;

**DELETE FROM** proizvodi

**WHERE** id = p\_proizvod\_id;

**END IF**;

**END** //

DELIMITER ;

Procedura „ObrisiProizvod“ briše proizvod i sve povezane zapise iz tablica „popusti“, „recenzije\_proizvoda“, i „wishlist“. Međutim, ne dopušta brisanje ako postoje zapisi u tablici „stavke\_narudzbe“. Najprije provjerava postoji li zapis u tablici „stavke\_narudzbe“ za neki proizvod i ako postoji, generira grešku. Ako nema povezanih podaka, briše sve povezane n-torke i sam proizvod iz tablice „proizvodi“.

**Procedura za ažuriranje svih podataka vezanih za neki proizvod**

DELIMITER //

**CREATE PROCEDURE** azuriraj\_proizvod(

**IN** p\_proizvod\_id **INT**,

**IN** p\_naziv **VARCHAR**(255),

**IN** p\_opis **TEXT**,

**IN** p\_cijena **DECIMAL**(10, 2),

**IN** p\_kategorija\_id **INT**,

**IN** p\_kolicina\_na\_skladistu **INT**,

**IN** p\_slika **VARCHAR**(255),

**IN** p\_specifikacije **TEXT**

)

**BEGIN**

**IF** p\_cijena <= 0 **THEN**

**SIGNAL SQLSTATE '45501'**

**SET MESSAGE\_TEXT** = 'Cijena mora biti veća od 0.';

**END IF**;

**IF** p\_kolicina\_na\_skladistu < 0 **THEN**

**SIGNAL SQLSTATE '45502'**

**SET MESSAGE\_TEXT** = 'Količina na skladištu ne može biti negativna.';

**END IF**;

**IF NOT EXISTS** (**SELECT** \* **FROM** kategorije\_proizvoda **WHERE** id = p\_kategorija\_id) **THEN**

**SIGNAL SQLSTATE '45509'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim ID ne postoji.';

**END IF**;

**IF NOT EXISTS** (**SELECT** \* **FROM** proizvodi **WHERE** id = p\_proizvod\_id) **THEN**

**SIGNAL SQLSTATE '45510'**

**SET MESSAGE\_TEXT** = 'Proizvod s unesenim ID ne postoji.';

**END IF**;

**UPDATE** proizvodi

**SET** naziv = p\_naziv,

opis = p\_opis,

cijena = p\_cijena,

kategorija\_id = p\_kategorija\_id,

kolicina\_na\_skladistu = p\_kolicina\_na\_skladistu,

slika = p\_slika,

specifikacije = p\_specifikacije

**WHERE** id = p\_proizvod\_id;

**END** //

DELIMITER ;

Procedura „azuriraj\_proizvod“ služi za ažuriranje podataka o proizvodu u tablici „proizvodi“. Procedura prima 8 ulaznih parametara: jedinstveni identifikator proizvoda, naziv, opis, cijenu, ID kategorije, količinu na skladištu, naziv slike i specifikacije. Prvo, procedura provodi nekoliko provjera kako bi osigurala ispravnost unesenih podataka. Provjerava se je li cijena veća od 0 te se u slučaju da nije, ispisuje grešku „Cijena mora biti veća od 0.“. Provjerava se je količina na skladištu pozitivna te se u slučaju da nije, ispisuje grešku „Količina na skladištu ne može biti negativna.“. Sljedeća provjera je da u tablici „kategorije\_proizvoda“ postoji zapis s unesenim „kategorija\_id“, ako nije ispisuje se „Kategorija s unesenim ID ne postoji.“. Na kraju provjerava ako postoji id koji je unesen u tablici „proizvodi“, ukoliko nije i za to ispisuje grešku „Kategorija s unesenim ID ne postoji.“. Ako nema nikakve greške procedura ažurira postojeći zapis iz tablice „proizvodi“ s novim vrijednostima za sve osim „id “ i „datum\_kreiranja“.

**Procedura za pretragu proizvode s ključnom riječi**

DELIMITER //

**CREATE PROCEDURE** PronadjiProizvode (

**IN** kljucna\_rijec **VARCHAR**(255)

)

**BEGIN**

**SELECT** \*

**FROM** proizvodi

**WHERE** naziv **LIKE** **CONCAT**('%', kljucna\_rijec, '%')

**OR** opis **LIKE CONCAT**('%', kljucna\_rijec, '%');

**END** //

DELIMITER ;

Procedura „PronadjiProizvode“ omogućava pretraživanje proizvoda na temelju ključne riječi u nazivima i opisima proizvoda. Koristi se „LIKE“ operator za pretraživanje proizvoda prema ključnim riječima u atributima „naziv“ i „opis“. Na kraju vraća rezultat kao tablicu sa svim n-torkama koje se podudaraju s ključnom riječi.

**Procedura za dodavanje kategorije proizvoda**

DELIMITER //

**CREATE PROCEDURE** dodaj\_kategoriju\_proizvoda(

**IN** p\_naziv VARCHAR(255),

**IN** p\_opis **TEXT**

)

**BEGIN**

**IF EXISTS** (**SELECT** \* **FROM** kategorije\_proizvoda **WHERE** naziv = p\_naziv) **THEN**

**SIGNAL SQLSTATE '45504'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim nazivom već postoji.';

**END IF**;

**INSERT INTO** kategorije\_proizvoda (naziv, opis)

**VALUES** (p\_naziv, p\_opis);

**END** //

DELIMITER ;

Procedura „dodaj\_kategoriju\_proizvoda“ se koristi za dodavanje nove kategorije u tablicu „kategorije\_proizvoda“. Prima „naziv“ i „opis“ za kategoriju koju dodajemo u tablicu „kategorije\_proizvoda“. Prvo se provjerava postoji li već kategorija s istim nazivom u tablici. Ukoliko postoji, ispisuje „Kategorija s unesenim nazivom već postoji.“. Ako kategorija ne postoji, procedura nastavlja izvršavanje i umetne novi zapis u tablicu „kategorije\_proizvoda“.

**Procedura za ažuriranje kategorije proizvoda**

DELIMITER //

**CREATE PROCEDURE** azuriraj\_kategoriju\_proizvoda(

**IN** p\_kategorija\_id **INT**,

**IN** p\_naziv **VARCHAR**(255),

**IN** p\_opis **TEXT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** kategorije\_proizvoda **WHERE** id = p\_kategorija\_id) **THEN**

**SIGNAL SQLSTATE '45509'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim ID ne postoji.';

**END IF;**

**IF EXISTS** (

**SELECT** \* **FROM** kategorije\_proizvoda

**WHERE** naziv = p\_naziv **AND** id != p\_kategorija\_id

) **THEN**

**SIGNAL SQLSTATE '45504'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim nazivom već postoji.';

**END IF;**

**UPDATE** kategorije\_proizvoda

**SET** naziv = p\_naziv,

opis = p\_opis

**WHERE** id = p\_kategorija\_id;

**END** //

DELIMITER ;

Procedura „azuriraj\_kategoriju\_proizvoda“ omogućava ažuriranje podataka o postojećoj kategoriji u tablici „kategorije\_proizvoda“. Prima 3 ulazna parametra, id kategorije koju mjenjamo, novi naziv kategorije i novi opis kategorije. Prvo provjerav postoji li kategorija s danim id u tablici. Ako ne postoji, ispisuje grešku „Kategorija s unesenim ID ne postoji.“. Nakon toga, procedura provjerava postoji li već neka druga kategorija s istim novim nazivom, kako bi se spriječilo dupliciranje naziva. Ako takva kategorija postoji, ispisuje se greška „Kategorija s unesenim nazivom već postoji.“. Kad obje provjere prođu bez problema, procedura nastavlja izvršavati naredbu „UPDATE“, koja u tablici „kategorije\_proizvoda“ postavlja novi „naziv“ i „opis“ za zapis čiji je „id“ jednak „p\_kategorija\_id“.

**Procedura za brisanje kategorije proizvoda**

DELIMITER //

**CREATE PROCEDURE** obrisi\_kategoriju\_proizvoda(

**IN** p\_kategorija\_id **INT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** kategorije\_proizvoda **WHERE** id = p\_kategorija\_id) **THEN**

**SIGNAL SQLSTATE '45509'**

**SET MESSAGE\_TEXT** = 'Kategorija s unesenim ID ne postoji.';

**END IF;**

**IF EXISTS** (**SELECT** \* **FROM** proizvodi **WHERE** kategorija\_id = p\_kategorija\_id) **THEN**

**SIGNAL SQLSTATE '45507'**

**SET MESSAGE\_TEXT** = 'Nije moguće obrisati kategoriju jer postoje proizvodi povezani s njom.';

**END IF**;

**DELETE FROM** kategorije\_proizvoda

**WHERE** id = p\_kategorija\_id;

**END** //

DELIMITER ;

Procedura „obrisi\_kategoriju\_proizvoda“ omogućava brisanje kategorije proizvoda iz tablice „kategorije\_proizvoda“. Procedura prima jedan ulazni parametar, „p\_kategorija\_id“, koji predstavlja jedinstveni identifikator kategorije koju želimo obrisati. Prvo se provjerava postoji li kategorija s tim id u tablici „kategorije\_proizvoda“, ako ne postoji ispisuje se greška „Kategorija s unesenim ID ne postoji.“. Zatim se provjerava postoji li u tablici „proizvodi“ bilo koji proizvod koji koristi ovu kategoriju, odnosno ima vrijednost „kategorija\_id“ jednaku id kojeg unesemo. Ako takav proizvod postoji, ispisuje se greška „'Nije moguće obrisati kategoriju jer postoje proizvodi povezani s njom.“. Ako obje provjere prođu procedura izvršava naredbu da s iz tablice „kategorije\_proizvoda“ izbriše unos s unesenim id i izvršava se brisanje n-torke.

**Procedura za unos recenzije proizvoda**

DELIMITER //

**CREATE PROCEDURE** dodaj\_recenziju\_proizvoda(

**IN** p\_proizvod\_id **INT**,

**IN** p\_korisnik\_id **INT**,

**IN** p\_ocjena **INT**,

**IN** p\_komentar **TEXT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** proizvodi **WHERE** id = p\_proizvod\_id) **THEN**

**SIGNAL SQLSTATE '45510'**

**SET MESSAGE\_TEXT** = 'Proizvod s unesenim ID ne postoji.';

**END IF**;

**IF NOT EXISTS** (**SELECT** \* **FROM** korisnici **WHERE** id = p\_korisnik\_id) **THEN**

**SIGNAL SQLSTATE '45508'**

**SET MESSAGE\_TEXT** = 'Korisnik s unesenim ID ne postoji.';

**END IF**;

**IF** p\_ocjena < 1 **OR** p\_ocjena > 5 **THEN**

**SIGNAL SQLSTATE '45511'**

**SET MESSAGE\_TEXT** = 'Ocjena mora biti između 1 i 5.';

**END IF**;

**IF EXISTS** (

**SELECT** \* **FROM** recenzije\_proizvoda

**WHERE** proizvod\_id = p\_proizvod\_id **AND** korisnik\_id = p\_korisnik\_id

) **THEN**

**SIGNAL SQLSTATE '45506'**

**SET MESSAGE\_TEXT** = 'Korisnik je već ostavio recenziju za ovaj proizvod.';

**END IF**;

**INSERT INTO** recenzije\_proizvoda (proizvod\_id, korisnik\_id, ocjena, komentar)

**VALUES** (p\_proizvod\_id, p\_korisnik\_id, p\_ocjena, p\_komentar);

**END** //

DELIMITER ;

Procedura „dodaj\_recenziju\_proizvoda“ omogućava za dodavanje nove recenzije proizvoda u tablicu recenzije\_proizvoda. Prima 4 parametra. Identifikator proizvoda ,identifikator korisnika ,ocjenu i tekstualni komentar. Prvo se provjerava ako proizvod s unesenim id postoji u tablici proizvodi. Ako ne postoji ispisuje se greška „Proizvod s unesenim ID ne postoji.“. Druga provjera je ako postoji korisnik s upisanim id u tablici „korisnik“. Ako ne ispisuje se greška „Korisnik s unesenim ID ne postoji.“. Zatim je provjera da unesena ocjena mora biti između 1 i 5 koja isto tako ukoliko ne ispoštujemo to ispisuje grešku „Ocjena mora biti između 1 i 5.“. I zadnja provjera je ta da se provjerava ako je korisnik već napisao recenziju, ukoliko nije procedura prolazi i izvršava unos novog zapisa u tablicu „recenzije\_proizvoda“ pomoću naredbe „INSERT“, čime se dodaje recenzija s navedenim podacima.

**Procedura za ažuriranje recenzije proizvoda**

DELIMITER //

**CREATE PROCEDURE** azuriraj\_recenziju\_proizvoda(

**IN** p\_recenzija\_id **INT**,

**IN** p\_ocjena **INT**,

**IN** p\_komentar **TEXT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** recenzije\_proizvoda **WHERE** id = p\_recenzija\_id) **THEN**

**SIGNAL SQLSTATE '45512'**

**SET MESSAGE\_TEXT** = 'Recenzija s unesenim ID ne postoji.';

**END IF**;

**IF** p\_ocjena < 1 **OR** p\_ocjena > 5 **THEN**

**SIGNAL SQLSTATE '45511'**

**SET MESSAGE\_TEXT** = 'Ocjena mora biti između 1 i 5.';

**END IF**;

**UPDATE** recenzije\_proizvoda

**SET** ocjena = p\_ocjena,

komentar = p\_komentar,

datum\_recenzije = **CURRENT\_TIMESTAMP**

**WHERE** id = p\_recenzija\_id;

**END** //

DELIMITER ;

Procedura „azuriraj\_recenziju\_proizvoda“ omogućava ažuriranje podataka o recenziji proizvoda u tablici „recenzije\_proizvoda“. Prima 3 parametra a to su, identifikator recenzije, novu ocjenu i novi komentar. Provjerava ako postoji recenzija u tablici „recenzije\_proizvoda“ koja se poklapa s id kojeg smo unijeli, ukoliko ne postoji ispisuje se greška „Recenzija s unesenim ID ne postoji.“. Nakon te provjere, druga provjera provjerava ako je nova ocjena koja je unesena između 1 i 5, ukoliko nije ispisuje se greška „Ocjena mora biti između 1 i 5.“. Kad prođu sve provjere ažuriraja se zapis putem naredbe „UPDATE“ i automatski, uz unesene podatke, postavlja se „datum\_recenzije“ na trenutni datum.

**Procedura za brisanje recenzije proizvoda**

DELIMITER //

**CREATE PROCEDURE** obrisi\_recenziju\_proizvoda(

**IN** p\_recenzija\_id **INT**

)

**BEGIN**

**IF NOT EXISTS** (**SELECT** \* **FROM** recenzije\_proizvoda **WHERE** id = p\_recenzija\_id) **THEN**

**SIGNAL SQLSTATE '45512'**

**SET MESSAGE\_TEXT** = 'Recenzija s unesenim ID ne postoji.';

**END IF**;

**DELETE FROM** recenzije\_proizvoda

**WHERE** id = p\_recenzija\_id;

**END** //

DELIMITER ;

Procedura „obrisi\_recenziju\_proizvoda“ omogućava brisanje recenzije proizvoda iz tablice „recenzije\_proizvoda“**.** Procedura prima jedan ulazni parametar, to je identifikator „recenzije\_proizvoda“ koji želimo obrisati. Prije nego što pokuša obrisati zapis, procedura provjerava postoji li recenzija s danim ID-jem u tablici. Ako recenzija ne postoji ispisuje se greška „Recenzija s unesenim ID ne postoji.“. Kad prođe provjeru procedura izvršava naredbu koja briše recenziju na temelju unesenog id.

## Procedure Morena

## Procedure Bruno

## Procedure Fran

## Procedure Josip

## Procedure Leo

-- Procedura: Dodavanje korisnika (Leo)

DELIMITER //

CREATE PROCEDURE dodaj\_korisnika(

IN p\_ime VARCHAR(255),

IN p\_prezime VARCHAR(255),

IN p\_email VARCHAR(255),

IN p\_lozinka VARCHAR(255),

IN p\_adresa TEXT,

IN p\_grad VARCHAR(255),

IN p\_telefon VARCHAR(20)

)

BEGIN

IF EXISTS (SELECT 1 FROM korisnici WHERE email = p\_email) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Korisnik sa ovim email-om već postoji!';

ELSE

INSERT INTO korisnici (ime, prezime, email, lozinka, adresa, grad, telefon, tip\_korisnika, datum\_registracije)

VALUES (p\_ime, p\_prezime, p\_email, p\_lozinka, p\_adresa, p\_grad, p\_telefon, 'kupac', CURDATE());

END IF;

END//

DELIMITER ;

Procedura dodaj\_korisnika omogućava dodavanje novog korisnika u bazu podataka. Pre unosa podataka, proverava da li već postoji korisnik sa istim mailom kako bi se osigurala jedinstvenost. Ako korisnik sa unetim email-om već postoji, procedura generira SQL grešku pomoću SIGNAL SQLSTATE. U slučaju da mail ne postoji, novi korisnik se unosi u tabelu korisnici zajedno sa svim podacima: ime, prezime, email, lozinka, adresa, grad, telefon, tip korisnika i datum registracije. Ova procedura osigurava referencijalni integritet i sprečava dupliciranje podataka u bazi.

CREATE PROCEDURE prikazi\_preporuke(korisnik\_id INT, page INT, per\_page INT)

BEGIN

DECLARE offset\_value INT;

-- Izračunavanje offset-a

SET offset\_value = (page - 1) \* per\_page;

SELECT

p.id AS proizvod\_id,

p.naziv,

p.opis,

p.cijena

FROM proizvodi p

WHERE p.kategorija\_id IN (

-- Preporučujemo proizvode iz kategorija koje je korisnik dodao u wishlist ili kupio

SELECT DISTINCT p2.kategorija\_id

FROM proizvodi p2

JOIN wishlist w ON p2.id = w.proizvod\_id

WHERE w.korisnik\_id = korisnik\_id

UNION

SELECT DISTINCT p3.kategorija\_id

FROM proizvodi p3

JOIN stavke\_narudzbe s ON p3.id = s.proizvod\_id

JOIN narudzbe n ON s.narudzba\_id = n.id

WHERE n.korisnik\_id = korisnik\_id

)

AND p.id NOT IN (

-- Iznimamo proizvode koje je korisnik već dodao u wishlist

SELECT proizvod\_id

FROM wishlist

WHERE korisnik\_id = korisnik\_id

)

ORDER BY p.cijena DESC

LIMIT per\_page OFFSET offset\_value; -- Koristimo izračunati offset

END//

DELIMITER ;

Procedura „prikazi\_preporuke“ koristi podatke o korisničkim aktivnostima, kao što su proizvodi koje je korisnik dodao u wishlist ili kupio, kako bi generirala personalizirane preporuke proizvoda. Na temelju tih podataka, aplikacija preporučuje proizvode iz kategorija koje su korisniku već bile zanimljive, čime se povećava relevantnost preporuka. Ujedno, izuzimaju se proizvodi koje je korisnik već dodao u wishlistu, čime se izbjegava ponavljanje. Ova procedura također implementira paginaciju, omogućujući korisnicima da pregledavaju proizvode stranica po stranica, čime se smanjuje opterećenje aplikacije i poboljšava korisničko iskustvo.

Kod je osmišljen tako da omogućava brzo i učinkovito dohvaćanje podataka o proizvodima koji pripadaju kategorijama koje je korisnik već pratio, bez potrebe za ponavljanjem složenih SQL upita u aplikaciji. Upit koristi UNION za kombiniranje kategorija proizvoda iz wishlist i narudžbi, dok ORDER BY sortira proizvode po cijeni. Paginacija se postiže korištenjem LIMIT i OFFSET, što omogućava prikazivanje samo određenog broja proizvoda po stranici. Na taj način, aplikacija ostaje efikasna i optimizirana, dok korisnici uživaju u personaliziranim i lako dostupnim preporukama.

-- Procedura: Brisanje korisnika (Leo)

DELIMITER //

CREATE PROCEDURE obrisi\_korisnika(

IN p\_korisnik\_id INT

)

BEGIN

DECLARE korisnik\_ima\_aktivne\_narudzbe BOOLEAN;

-- Provjera da li korisnik ima aktivne narudžbe koje nisu 'dostavljeno'

SELECT EXISTS (SELECT 1 FROM narudzbe WHERE korisnik\_id = p\_korisnik\_id AND status\_narudzbe != 'dostavljeno')

INTO korisnik\_ima\_aktivne\_narudzbe;

IF korisnik\_ima\_aktivne\_narudzbe THEN

SIGNAL SQLSTATE '45001'

SET MESSAGE\_TEXT = 'Korisnik ima aktivne narudžbe i ne može se obrisati!';

ELSE

-- Brisanje povezanih recenzija

DELETE FROM recenzije\_proizvoda WHERE korisnik\_id = p\_korisnik\_id;

-- Brisanje povezanih podataka

DELETE FROM placanja WHERE narudzba\_id IN (SELECT id FROM narudzbe WHERE korisnik\_id = p\_korisnik\_id);

DELETE FROM stavke\_narudzbe WHERE narudzba\_id IN (SELECT id FROM narudzbe WHERE korisnik\_id = p\_korisnik\_id);

DELETE FROM narudzbe WHERE korisnik\_id = p\_korisnik\_id;

DELETE FROM wishlist WHERE korisnik\_id = p\_korisnik\_id;

-- Na kraju, brisanje korisnika

DELETE FROM korisnici WHERE id = p\_korisnik\_id;

END IF;

END //

DELIMITER ;

Ova SQL procedura obrisi\_korisnika omogućava brisanje korisnika iz sistema, uz provjeru da li korisnik ima aktivne narudžbe koje nisu označene kao 'dostavljeno'. Procedura prima jedan parametar: p\_korisnik\_id, koji predstavlja identifikator korisnika koji treba biti obrisan. Prvo, procedura provodi provjeru da li korisnik ima aktivne narudžbe (koje nisu dostavljene). Ako korisnik ima aktivne narudžbe, procedura generira grešku s porukom "Korisnik ima aktivne narudžbe i ne može se obrisati!". Ako korisnik nema aktivne narudžbe, procedura nastavlja s brisanjem svih povezanih podataka: recenzija, stavki narudžbi, narudžbi, i wishlist-a. Na kraju, ako su svi povezani podaci izbrisani, korisnik se briše iz tabele korisnici. Ova procedura osigurava da se korisnici s aktivnim narudžbama ne mogu slučajno obrisati, čime se sprječavaju potencijalni gubici podataka i nepravilnosti u poslovnom procesu.

-- Procedura: Ažuriranje korsinika(Leo)

DELIMITER //

CREATE PROCEDURE azuriraj\_korisnika(

IN p\_korisnik\_id INT,

IN p\_ime VARCHAR(255),

IN p\_prezime VARCHAR(255),

IN p\_email VARCHAR(255),

IN p\_adresa TEXT,

IN p\_grad VARCHAR(255),

IN p\_telefon VARCHAR(20)

)

BEGIN

UPDATE korisnici

SET

ime = p\_ime,

prezime = p\_prezime,

email = p\_email,

adresa = p\_adresa,

grad = p\_grad,

telefon = p\_telefon

WHERE id = p\_korisnik\_id;

END//

DELIMITER ;

Procedura azuriraj\_korisnika omogućava ažuriranje podataka korisnika u tabeli korisnici. Procedura prihvata podatke kao što su ime, prezime, email, adresa, grad i telefon, zajedno sa jedinstvenim ID-jem korisnika. Koristeći SQL naredbu UPDATE, ažurira sve navedene podatke u tabeli korisnici na osnovu unijetog ID-ja. Ova procedura omogućava jednostavno i sigurno ažuriranje podataka korisnika u bazi, uz očuvanje postojećih veza i integriteta.

-- Procedura: Ažuriraj tip korisnika (Leo)

DELIMITER //

CREATE PROCEDURE azuriraj\_tip\_korisnika(IN p\_korisnik\_id INT, IN p\_tip\_korisnika ENUM('kupac', 'admin'))

BEGIN

UPDATE korisnici

SET tip\_korisnika = p\_tip\_korisnika

WHERE id = p\_korisnik\_id;

END//

DELIMITER ;

Ova SQL procedura azuriraj\_tip\_korisnika omogućava ažuriranje tipa korisnika u sistemu. Procedura prima dva parametra: p\_korisnik\_id, koji označava identifikator korisnika kojem treba biti promijenjen tip, i p\_tip\_korisnika, koji predstavlja novi tip korisnika koji može biti 'kupac' ili 'admin'. Procedura ažurira vrijednost u koloni tip\_korisnika u tabeli korisnici za korisnika sa specificiranim ID-em (p\_korisnik\_id). Ova procedura omogućava administratorima da lako mijenjaju status korisnika, na primjer, da promotivni korisnici postanu administratori ili obrnuto, bez potrebe za ručnim ažuriranjem podataka u bazi.

-- Procedura: Dodaj u wishlist (Leo)

DELIMITER //

CREATE PROCEDURE dodaj\_u\_wishlist(korisnik\_id INT, proizvod\_id INT, grupa VARCHAR(255))

BEGIN

-- Dodavanje proizvoda u wishlist (pretpostavljamo da je provjera duplikata već obavljena kroz triger)

INSERT INTO wishlist (korisnik\_id, proizvod\_id, grupa)

VALUES (korisnik\_id, proizvod\_id, grupa);

END//

DELIMITER ;

Ova SQL procedura dodaj\_u\_wishlist omogućava dodavanje proizvoda u wishlist korisnika. Procedura prima tri parametra: korisnik\_id (identifikator korisnika), proizvod\_id (identifikator proizvoda) i grupa (kategorija ili grupa u koju proizvod pripada). Unutar procedure, izvršava se SQL upit za unos novog proizvoda u tabelu wishlist s navedenim podacima. Pretpostavlja se da je provjera duplikata već obavljena kroz okidač (trigger) kako bi se osiguralo da isti proizvod ne može biti dodan više puta.

-- Procedura: Ukloni u wishlist (Leo)

DELIMITER //

CREATE PROCEDURE ukloni\_proizvod\_iz\_wishliste(korisnik\_id INT, proizvod\_id INT)

BEGIN

-- Brisanje proizvoda iz wishlist-a

DELETE FROM wishlist WHERE korisnik\_id = korisnik\_id AND proizvod\_id = proizvod\_id;

END//

DELIMITER ;

Ova SQL procedura ukloni\_proizvod\_iz\_wishliste omogućava uklanjanje proizvoda iz wishlist-a korisnika. Procedura prima dva parametra: korisnik\_id (identifikator korisnika) i proizvod\_id (identifikator proizvoda). Unutar procedure, izvršava se SQL upit za brisanje odgovarajućeg proizvoda iz tabele wishlist koji pripada korisniku s navedenim korisnik\_id i proizvod\_id. Ova funkcionalnost omogućava korisnicima da uklone proizvode koje su prethodno dodali u svoju wishlist-u.

# Okidači

## Okidači Loren

**Okidač za obavijest o niskoj zalihi**

**CREATE TEMPORARY TABLE** privremene\_obavijesti (

poruka **TEXT**,

vrijeme\_kreiranja **DATETIME**

);

DELIMITER //

**CREATE TRIGGER** au\_ObavijestNiskaZaliha

**AFTER UPDATE ON** proizvodi

**FOR EACH ROW**

**BEGIN**

**IF NEW**.kolicina\_na\_skladistu < 5 **THEN**

**INSERT INTO** privremene\_obavijesti (poruka, vrijeme\_kreiranja)

**VALUES** (

**CONCAT**('Upozorenje: Zaliha za proizvod "', **NEW**.naziv, '" je pala ispod 5.'),

**NOW**()

);

**END IF**;

**END** //

DELIMITER ;

Okidač „au\_ObavijestNiskaZaliha“ automatski generira obavijest kada zaliha proizvoda padne ispod 5. Obavijest se sprema u privremenu tablicu „privremene\_obavijesti“ koju kreiramo prije samog okidača. Okidač provjerava je li nova količina manja od 5. Ako je, u privremenu tablicu „privremene\_obavijesti“ unosi poruku „Zaliha za proizvod 'proizvod' je pala ispod 5.“ i trenutni datum i vrijeme.

**Okidač koji ne dopušta recenziju ako korisnik nije kupio proizvod ili ako je već napisao recenziju**

DELIMITER //

**CREATE TRIGGER** bi\_RestrikcijaRecenzije

**BEFORE INSERT ON** recenzije\_proizvoda

**FOR EACH ROW**

**BEGIN**

**IF NOT EXISTS** (

**SELECT** 1

**FROM** stavke\_narudzbe sn

**JOIN** narudzbe n **ON** sn.narudzba\_id = n.id

**WHERE** sn.proizvod\_id = **NEW**.proizvod\_id **AND** n.korisnik\_id = **NEW**.korisnik\_id

) **THEN**

**SIGNAL SQLSTATE '45505'**

**SET MESSAGE\_TEXT** = 'Korisnik nije kupio ovaj proizvod i ne može ostaviti recenziju.';

**END IF**;

**IF EXISTS** (

**SELECT** 1

**FROM** recenzije\_proizvoda

**WHERE** proizvod\_id = **NEW**.proizvod\_id **AND** korisnik\_id = **NEW**.korisnik\_id

) **THEN**

**SIGNAL SQLSTATE '45506'**

**SET MESSAGE\_TEXT** = 'Korisnik je već ostavio recenziju za ovaj proizvod.';

**END IF**;

**END** //

DELIMITER ;

Okidač „bi\_RestrikcijaRecenzije“ sprječava dodavanje recenzije za proizvod ako korisnik nije kupio taj proizvod. Prije dodavanja novog podatka u tablicu „recenzije\_proizvoda“, provjerava se postoji li odgovarajuća narudžba u tablici „stavke\_narudzbe“, gdje se podudaraju „proizvod\_id“ iz narudžbe i „korisnik\_id“ iz tablice „recenzije\_proizvoda“. Ako podudaranje ne postoji, generira se greška "Korisnik nije kupio ovaj proizvod i ne može ostaviti recenziju". Dodatno se provjerava da li već postoji recenzija za isti „proizvod\_id“ i „korisnik\_id“ u tablici „recenzije\_proizvoda“. Ako postoji, generira se greška "Korisnik je već ostavio recenziju za ovaj proizvod".

## Okidači Morena

## Okidači Bruno

## Okidači Fran

## Okidači Josip

## Okidači Leo

-- Okidač: Automatsko postavljanje datuma registracije korisnika (Leo)

DELIMITER //

CREATE TRIGGER postavi\_datum\_registracije

BEFORE INSERT ON korisnici

FOR EACH ROW

BEGIN

IF NEW.datum\_registracije IS NULL THEN

SET NEW.datum\_registracije = CURDATE();

END IF;

END//

DELIMITER ;

Ovaj okidač se izvršava prije svakog unosa (BEFORE INSERT) u tablicu korisnici. Njegova svrha je da automatski postavi vrijednost atributa datum\_registracije na trenutni datum (CURDATE()), ukoliko nije eksplicitno definirano prilikom unosa. Ovim se osigurava da svaki korisnik koji se unosi u bazu ima ispravno postavljen datum registracije. Okidač doprinosi dosljednosti i točnosti podataka u tablici korisnici, eliminisanjem mogućnosti unosa zapisa sa nedostajućim ili neispravnim datumom.

-- Okidač: spreči duplikate wishlist (Leo)

DELIMITER //

CREATE TRIGGER spreči\_duplikate\_wishlist

BEFORE INSERT ON wishlist

FOR EACH ROW

BEGIN

IF EXISTS (

SELECT 1

FROM wishlist

WHERE korisnik\_id = NEW.korisnik\_id AND proizvod\_id = NEW.proizvod\_id

) THEN

SIGNAL SQLSTATE '45002'

SET MESSAGE\_TEXT = 'Proizvod je već u wishlist-u ovog korisnika!';

END IF;

END//

DELIMITER ;

Ovaj okidač se izvršava pre svakog unosa (BEFORE INSERT) u tablicu wishlist. Njegova svrha je da spreči dodavanje duplikata u tabelu wishlist, tj. da isti proizvod ne može biti dodan u wishlist istog korisnika više puta. Provjerava postojanje odgovarajućeg zapisa koristeći EXISTS sa odgovarajućim kriterijima (korisnik\_id i proizvod\_id). Ukoliko zapis već postoji, okidač generira SQL grešku sa porukom „Proizvod je već u wishlist-u ovog korisnika!“ koristeći SIGNAL SQLSTATE. Ovaj okidač je ključan za održavanje integriteta podataka i sprečavanje redundantnih zapisa u tabeli wishlist.

# Transakcije