Sveučilište Jurja Dobrile u Puli

Fakultet Informatike u Puli

Baze podataka II

Projekt

Sustav za upravljanje hotelom

Dokumentacija projekta

Studijski smjer : Informatika

Kolegij : Baze podataka 2

Mentor : doc.dr.sc. Goran Oreški

Studenti : Filip Bastijanić, Luka Blašković, Ivan Lorbek, Kristijan Žulić, Alesandro Žužić

Poveznica na video: Tim - 4 video

Pula, siječanj 2021.

Sadržaj

Za naš projekt smo odlučili napraviti bazu podatka za upravljanjem hotelom.

Pomoću ove baze korisnici mogu rezervirati i odabrati smještaj, izabrati asortiman i druge dodatne usluge koje im omogućuju iskoristiti najbolje od hotela što se pruža.

Shema baze podataka

Mjesto prebivališta

```
CREATE TABLE mjesto_prebivalista (
        id_mjesto_prebivalista SERIAL AUTO_INCREMENT,
        drzava VARCHAR(50) NOT NULL,
        grad VARCHAR(50) NOT NULL,
        postanski_broj VARCHAR(20) NOT NULL,
        adresa VARCHAR(50) NOT NULL,
        CONSTRAINT mjesto_prebivalista_id_mjesto_p_pk
        PRIMARY KEY (id_mjesto_prebivalista)
);
```

- Primarni ključ nam je 'id_mjesto_prebivališta' koji je ujedno i AUTO_INCREMENT kako bi nam se automatski upisivali redni brojevi unosa svaki puta kada bi se upisao novi podatak.
- Tablica *Mjesto prebivališta* će nam služiti za evidenciju svih ljudi koji su prijavljeni u sustav hotela bili kao zaposlenici ili kao gosti. Tablica će sadržavati podatke za upis države iz koje osoba dolazi, grad, poštanski broj i adresa stanovanja.

Gost

```
CREATE TABLE gost (
        id_gost SERIAL AUTO_INCREMENT,
        ime VARCHAR(20) NOT NULL,
        prezime VARCHAR(20) NOT NULL,
        OIB VARCHAR(30) NOT NULL,
        broj_osobne_iskaznice VARCHAR(30) NOT NULL,
        id_mjesto_prebivalista BIGINT UNSIGNED NOT NULL,
        datum_rodenja DATE NOT NULL,
        CONSTRAINT gost_id_gost_pk PRIMARY KEY (id_gost),
        CONSTRAINT gost_broj_oi_uq UNIQUE (broj_osobne_iskaznice),
        CONSTRAINT gost_oib_uq UNIQUE (OIB),
        CONSTRAINT gost_mp_fk FOREIGN KEY (id_mjesto_prebivalista)
        REFERENCES mjesto_prebivalista(id_mjesto_prebivalista)
);
```

- Tablica služi za evidenciju podataka gostiju.
- Tablicu *gost* ćemo popunjavati sa osobnim podacima u koju spadaju ime, prezime, mjesto prebivališta, datum rođenja te unikatni OIB i broj osobne iskaznice.

- Za datum rođenja koristimo funkciju 'DATE' u koju upisujemo datum i godinu rođenja.
- Podatke za mjesto prebivališta smo spojili sa prijašnjom tablicom *mjesto prebivališta* uz pomoć stranog ključa kako bi mogli pohraniti podatke za gosta.

Zvanje

```
CREATE TABLE zvanje (
        id_zvanje SERIAL AUTO_INCREMENT,
        naziv VARCHAR(50) NOT NULL,
        opis_posla TEXT NOT NULL,
        CONSTRAINT zvanje_id_zvanje_pk
        PRIMARY KEY (id_zvanje)
);
```

- Tablica *Zvanje* nam služi za unos podataka o zaposlenim osobama pomoću koje im dodjeljujemo opis posla i vrstu.
- id_zvanje se povećava automatski s naredbom *'AUTO_INCREMENT'*.

Djelatnik

```
CREATE TABLE djelatnik (
        id_djelatnik SERIAL AUTO_INCREMENT,
        ime VARCHAR(20) NOT NULL,
        prezime VARCHAR(20) NOT NULL,
        id_zvanje BIGINT UNSIGNED NOT NULL,
        datum_zaposljenja DATE NOT NULL,
        CONSTRAINT djelatnik_id_djelatnik_pk PRIMARY KEY (id_djelatnik),
        CONSTRAINT djelatnik_zvanje_fk FOREIGN KEY (id_zvanje)
        REFERENCES zvanje(id_zvanje)
);
```

- Tablica *Djelatnik* nam služi za upis podataka svake osobe koja je zaposlena u hotelu.
- ID djelatnika ćemo povećavati sa *'AUTO_INCREMENT'* naredbom. U tablicu idu ime i prezime djelatnika te i datum zaposlenja za koji koristimo naredbu 'DATE'.
- S ovom tablicom smo povezali i prijašnju tablicu *Zvanje* iz koje ćemo djelatniku dodijeliti sve podatke iz te tablice.

Soba

```
CREATE TABLE soba (
        id_soba SERIAL AUTO_INCREMENT,
        sifra VARCHAR(10) NOT NULL,
        kat VARCHAR(10) NOT NULL,
        standardna_cijena DECIMAL(7,2),
        vrsta VARCHAR(5) NOT NULL,
        stanje VARCHAR(15) NOT NULL DEFAULT 'slobodno',
        CONSTRAINT soba_id_soba_pk PRIMARY KEY (id_soba)
);
```

- Tablica *Soba* nam služi za uvrštavanje svih podataka o pojedinoj sobi.
- U stupac ***'vrsta sobe'*** dodjeljujemo tipove naziva svakoj sobi poput jednokrevetna, dvokrevetna itd.

- Dodajemo informacije o cijeni sobe sa stupcem ***'standarnda_cijena'***, a u stupac ***'kat'*** upisujemo informacije o lokaciji sobe.
- Stanje sobe po zadanome je 'slobodno'.
- Primarni ključ nam je ***'id_soba'***.

Sezona

```
CREATE TABLE sezona (
        id_sezona SERIAL AUTO_INCREMENT,
        kategorija CHAR NOT NULL,
        naziv VARCHAR(15) NOT NULL,
        multiplikator_cijene DECIMAL(4,2) NOT NULL,
        CONSTRAINT sezona_id_sezona_pk PRIMARY KEY (id_sezona)
);
```

- Tablica *Sezona* nam služi za određivanje godišnjih doba koje upisujemo u stupac *'naziv'* te pomoću stupca ***'multiplikator_cijene'*** povisujemo odnosno smanjujemo cijenu usluge ovisno o godišnjem dobu.
- Primarni ključ nam je ***'id_sezona'***.

Aranžman

```
CREATE TABLE aranzman (
        id_aranzman SERIAL AUTO_INCREMENT,
        naziv VARCHAR(20) NOT NULL,
        opis_aranzmana TEXT NOT NULL,
        cijena DECIMAL(7,2) NOT NULL,
        CONSTRAINT aranzman_id_aranzman_pk PRIMARY KEY (id_aranzman)
);
```

- Tablica *Aranžman* omogućava gostu mogućnost odabira vrste prehrane.
- Kao primarni ključ koristili smo ***'id_aranzman'*** kojemu smo dodijelili atribut ***'AUTO_INCREMENT'***.

Dodatne usluge

```
CREATE TABLE dodatne_usluge (
        id_dodatne_usluge SERIAL AUTO_INCREMENT,
        naziv VARCHAR(50) NOT NULL,
        cijena DECIMAL(7,2) NOT NULL,
        CONSTRAINT dodatne_usluge_id_dodatne_u_pk
        PRIMARY KEY (id_dodatne_usluge)
);
```

- Tablica *Dodatne usluge* nam dozvoljava da gostima ponudimo

Rezervacija

```
CREATE TABLE rezervacija (
        id_rezervacija SERIAL AUTO_INCREMENT,
        id_soba BIGINT UNSIGNED NOT NULL,
        id_gost BIGINT UNSIGNED NOT NULL,
```

```
        id_sezona BIGINT UNSIGNED NOT NULL,
        id_aranzman BIGINT UNSIGNED NOT NULL,
        pocetak_rezervacije DATE NOT NULL,
        kraj_rezervacije DATE NOT NULL,
        broj_osoba TINYINT NOT NULL DEFAULT 1,
        CONSTRAINT rezervacija_id_rezervacija_pk
        PRIMARY KEY (id_rezervacija),
        CONSTRAINT rezervacija_soba_fk FOREIGN KEY (id_soba)
        REFERENCES soba(id_soba),
        CONSTRAINT rezervacija_gost_fk FOREIGN KEY (id_gost)
        REFERENCES gost(id_gost),
        CONSTRAINT rezervacija_sezona_fk FOREIGN KEY (id_sezona)
        REFERENCES sezona(id_sezona),
        CONSTRAINT rezervacija_aranzman_fk FOREIGN KEY (id_aranzman)
        REFERENCES aranzman(id_aranzman),
        CONSTRAINT rezervacija_check_broj_osoba CHECK (broj_osoba>0)
);
```

- Tablica *Rezervacija* služi za rezervaciju smještaja.
- Primarni ključ nam je bio ***'id_rezervacija'***.
- Ovo nam je i najvažnija tablica preko koje se odvija skoro sve.
- S njom su povezane tablice soba, gost, sezona i aranžman.

Odabrani gosti

```
CREATE TABLE odabrani_gosti (
        id_odabrani_gost SERIAL AUTO_INCREMENT,
        id_gost BIGINT UNSIGNED NOT NULL,
        id_rezervacija BIGINT UNSIGNED NOT NULL,
        CONSTRAINT odabrani_gost_id_odabrani_g_pk
        PRIMARY KEY (id_odabrani_gost),
        CONSTRAINT odabrani_gost_gost_fk FOREIGN KEY (id_gost)
        REFERENCES gost(id_gost),
        CONSTRAINT odabrani_gost_rezervacija_fk
        FOREIGN KEY (id_rezervacija)
        REFERENCES rezervacija(id_rezervacija)
);
```

- Tablica *Odabrani gosti* za primarni ključ koristi ***'id_odabrani_gost'.***
- Povezana je s tablicama gost i rezervacija.

Odabrane usluge

```
CREATE TABLE odabrane_usluge (
        id_odabrane_usluge SERIAL AUTO_INCREMENT,
        id_dodatne_usluge BIGINT UNSIGNED NOT NULL,
        id_rezervacija BIGINT UNSIGNED NOT NULL,
        kolicina INTEGER NOT NULL,
        CONSTRAINT odabrane_usluge_id_odabrane_u_pk
        PRIMARY KEY (id_odabrane_usluge),
        CONSTRAINT odabrane_usluge_dodatne_u_fk
        FOREIGN KEY (id_dodatne_usluge)
        REFERENCES dodatne_usluge(id_dodatne_usluge),
        CONSTRAINT odabrane_usluge_rezervacija_fk
        FOREIGN KEY (id_rezervacija)
        REFERENCES rezervacija(id_rezervacija),
```

```
        CONSTRAINT odabrane_usluge_check_kolicina CHECK (kolicina>0)
);
```

- Tablica *Odabrane usluge* je povezana s tablicama dodatne usluge i rezervacija.
- Tablica nam služi da spremi sve usluge koje je gost izabrao.

Račun

```
CREATE TABLE racun (
        id_racun SERIAL AUTO_INCREMENT,
        sifra VARCHAR(10) NOT NULL,
        id_rezervacija BIGINT UNSIGNED NOT NULL,
        id_djelatnik BIGINT UNSIGNED NOT NULL,
        datum_i_vrijeme_izdavanja DATETIME NOT NULL,
        ukupna_cijena DECIMAL(10,2) DEFAULT 0.00,
        CONSTRAINT racun_id_racun_pk PRIMARY KEY (id_racun),
        CONSTRAINT racun_djelatnik_fk FOREIGN KEY (id_djelatnik)
        REFERENCES djelatnik(id_djelatnik),
        CONSTRAINT racun_rezervacija_fk FOREIGN KEY (id_rezervacija)
        REFERENCES rezervacija(id_rezervacija)
);
```

- Tablica *Račun* sprema sve podatke o troškovima koje je pojedini gost
- Povezana je s tablicama djelatnik i rezervacija
- Primarni ključ nam je ***'id_racun'***

Arhiva rezervacija

```
CREATE TABLE arhiva_rezervacija (
        id_rezervacija SERIAL AUTO_INCREMENT,
        id_soba BIGINT UNSIGNED NOT NULL,
        id_gost BIGINT UNSIGNED NOT NULL,
        id_sezona BIGINT UNSIGNED NOT NULL DEFAULT 0,
        id_aranzman BIGINT UNSIGNED NOT NULL,
        pocetak_rezervacije DATE NOT NULL,
        kraj_rezervacije DATE NOT NULL,
        broj_osoba TINYINT NOT NULL DEFAULT 1,
        vrijeme_brisanja DATETIME DEFAULT now()
);
```

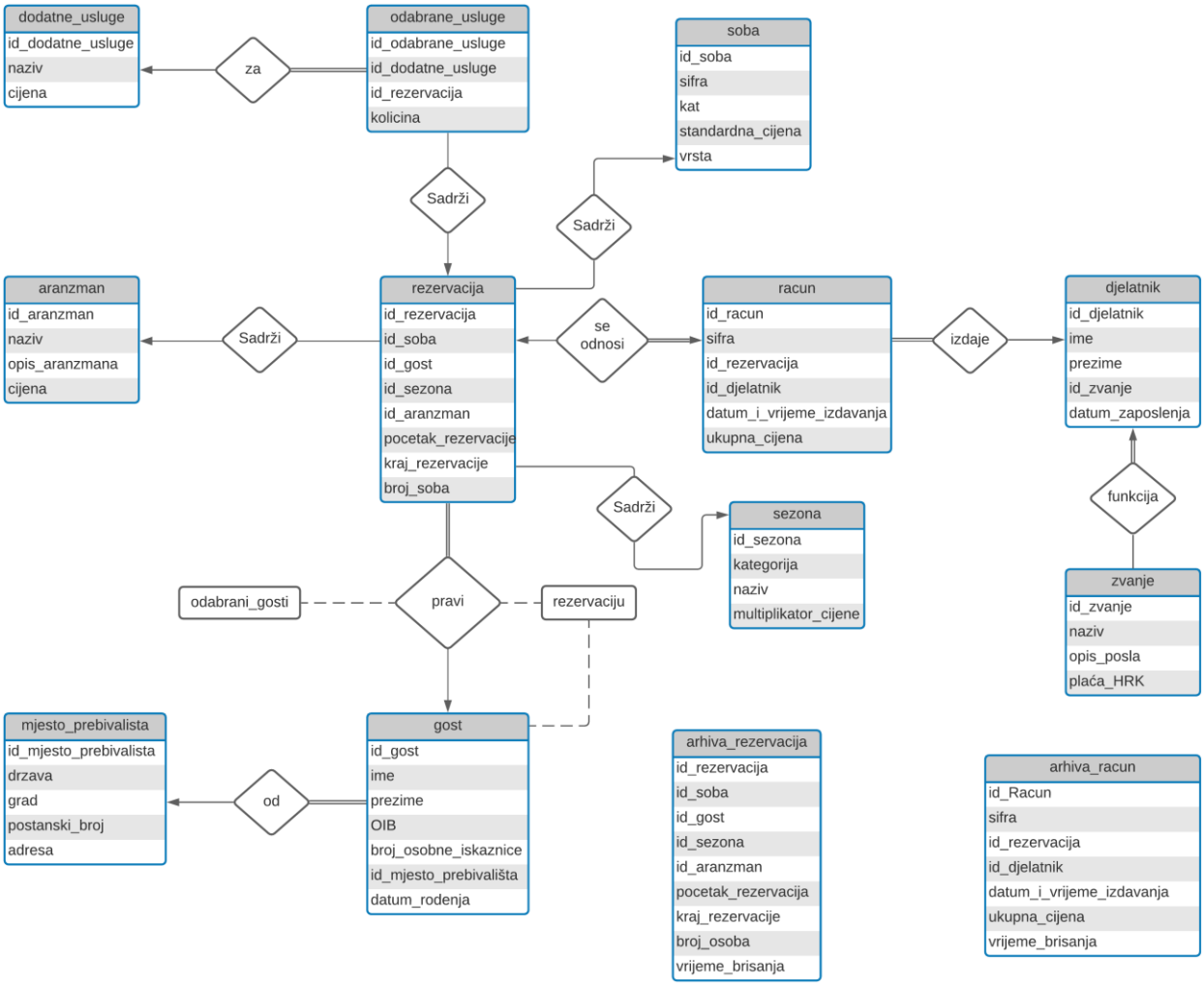
- Tablica *Arhiva rezervacija* nam služi tome da sve rezervacije starije od 10 godina možemo premjestiti u tablicu.

Arhiva račun

```
CREATE TABLE arhiva_racun (
        id_racun SERIAL AUTO_INCREMENT,
        sifra VARCHAR(12) NOT NULL DEFAULT '0-0-0',
        id_rezervacija BIGINT UNSIGNED NOT NULL,
        id_djelatnik BIGINT UNSIGNED NOT NULL,
        datum_i_vrijeme_izdavanja DATETIME NOT NULL,
        ukupna_cijena DECIMAL(10,2) DEFAULT 0.00,
        vrijeme_brisanja DATETIME DEFAULT now()
);
```

- Tablici *Arhiva račun* ista je svrha kao i tablici arhiva rezerv. samo što sprema račun

ER dijagram



Ovo je prikaz naše baze podataka pomoću ER dijagrama koji će nam omogućiti bolji pregled veza i odnosa između tablica. Dijagram je grafički uređen preko Lucidchart web stranice. ER dijagram nam prikazuje kako se samo jedna rezervacija odnosi na jedan račun dok je račun u potpunosti ovisan o djelatniku, što znači, da bez djelatnika nema ni računa. Na djelatnika se nadovezuje zvanje koje je u potpunosti ovisno za djelatnika (djelatnik bez zvanja ne postoji). Zatim imamo sezonu koja može biti na više rezervacija, ali samo jedna sezona može biti na jednoj rezervaciji. Mjesto prebivališta je u potpunosti ovisno o gostu koji je potpuno ovisan rezervaciji. Određena rezervacija može sadržavati samo jedan aranžman, dok se pojedini aranžman može nalaziti na više rezervacija. Odabrane usluge u potpunosti ovise o dodatnim uslugama. Rezervacija može sadržavati samo jednu sobu na popisu, dok se pojedina soba može pojaviti na više rezervacija.

FUNKCIONALNOST PROGRAMSKOG RJEŠENJA

Funkcije

Broj gostiju po rezervaciji

```
DELIMITER //
CREATE FUNCTION broj_gostiju_po_rezervaciji(id INT)
RETURNS INTEGER

DETERMINISTIC
BEGIN

        DECLARE rezultat INTEGER DEFAULT 0;
        SELECT COUNT(*) INTO rezultat FROM odabrani_gosti
        GROUP BY id_rezervacija HAVING id_rezervacija = id;
RETURN rezultat+1;

END//
DELIMITER ;
```

- Pomoću ove funkcije dobijemo broj gostiju iz tablice odabrani gosti za pojedinu rezervaciju tako što uzmemo ID rezervacije s kojom vratimo broj osoba za tu rezervaciju.

Ukupna cijena po rezervaciji

```
DELIMITER //
CREATE FUNCTION ukupna_cijena_po_rezervaciji(id INT) RETURNS DECIMAL(7,2)

DETERMINISTIC
BEGIN

        DECLARE cijene_odabranih_usluga DECIMAL(10,2) DEFAULT 0.00;
        DECLARE cijene_sobe_po_danima DECIMAL(10,2) DEFAULT 0.00;
        DECLARE cijena_aranzmana DECIMAL(10,2) DEFAULT 0.00;
        DECLARE ukupna_cijena DECIMAL(10,2) DEFAULT 0.00;
        DECLARE multiplikator DECIMAL(10,2) DEFAULT 0.00;

SELECT SUM(cijena*kolicina) INTO cijene_odabranih_usluga
        FROM odabrane_usluge NATURAL JOIN dodatne_usluge
        GROUP BY id_rezervacija HAVING id_rezervacija = id;
SELECT (standardna_cijena * DATEDIFF(kraj_rezervacije,
        pocetak_rezervacije)) INTO cijene_sobe_po_danima
        FROM rezervacija NATURAL JOIN soba NATURAL JOIN sezona
        WHERE id_rezervacija = id;
SELECT cijena INTO cijena_aranzmana FROM rezervacija
        NATURAL JOIN aranzman WHERE id_rezervacija = id;
SELECT multiplikator_cijene INTO multiplikator
        FROM rezervacija NATURAL JOIN soba NATURAL JOIN sezona
        WHERE id_rezervacija = id;

SET ukupna_cijena = cijene_odabranih_usluga +
        cijene_sobe_po_danima + cijena_aranzmana;
RETURN ukupna_cijena;
```

```
END//
DELIMITER ;
```

- S ovom funkcijom zbrajamo ukupnu cijenu po računu tako što uzmemo ID rezervacije i vratimo ukupnu cijenu u račun.
- Prvi upit (Odabrane usluge) sprema ukupnu cijenu odabranih usluga za određenu rezervaciju u varijablu ***'cijene_odabranih_usluga'***
- Drugi upit (Odabrana soba) sprema cijenu sobe pomnoženu s brojem dana u rezervaciji u varijablu ***'cijene_sobe_po_danima'***
- Treći upit (Odabran aranžman) sprema cijenu aranžmana za određenu rezervaciju u varijavlu ***'cijena_aranzmana'***
- Četvrti upit (Odabrana sezona) sprema ***'multiplikator_cijene'*** za određenu rezervaciju u varijablu ***'multiplikator'***
- Na kraju se sve zbraja i množi sa multiplikatorom

Generiranje šifre soba

```
DELIMITER //
CREATE FUNCTION gen_sifra_sobe(p_id_sobe VARCHAR(10), p_kat VARCHAR(10),
p_vrsta VARCHAR(10)) RETURNS VARCHAR(30)
DETERMINISTIC
BEGIN
    DECLARE prvi_dio VARCHAR(10); -- id_sobe
    DECLARE drugi_dio VARCHAR(10); -- kat
    DECLARE treci_dio VARCHAR(10); -- vrsta

    IF p_id_sobe != 0 THEN
                SET prvi_dio = p_id_sobe;
        ELSE
                SET prvi_dio = (SELECT id_soba FROM soba ORDER BY id_soba
DESC LIMIT 1)+1;
        END IF;
    IF p_kat = 'prvi' THEN
                SET drugi_dio = '1';
        ELSEIF p_kat = 'drugi' THEN
                SET drugi_dio = '2';
        ELSEIF p_kat = 'treći' THEN
                SET drugi_dio = '3';
        ELSEIF p_kat = 'četvrti' THEN
                SET drugi_dio = '4';
        ELSEIF p_kat = 'peti' THEN
                SET drugi_dio = '5';
        ELSEIF p_kat = 'šesti' THEN
                SET drugi_dio = '6';
        END IF;
    SET treci_dio = p_vrsta;

    RETURN CONCAT(prvi_dio,"-",drugi_dio,"-",treci_dio);
END //
DELIMITER ;
```

- **1.dio -** Uzima se ID od sobe i stavlja se u prvi dio varijable, ako je ID direktno upisan preko SQL-a. Ako je unesen pomoću stranice onda uzimamo zadnjeg ID- a iz sobe.

- **2.dio –** sprema se kat koji pomoću selekcije pretvaramo string u brojku
- **3.dio –** sprema se vrsta i vraćaju se sva tri dijela spojeno

Izračun sezone

Part 1

```
DELIMITER //
CREATE FUNCTION izracun_sezone(pocetni_datum DATE, krajnji_datum DATE)
RETURNS INTEGER

DETERMINISTIC
BEGIN

        DECLARE duljina_boravka INTEGER;
        DECLARE ljeto_pocetak DATE;
        DECLARE ljeto_kraj DATE;
        DECLARE jesen_pocetak DATE;
        DECLARE jesen_kraj DATE;
        DECLARE zima_pocetak DATE;
        DECLARE zima_kraj DATE;
        DECLARE proljece_pocetak DATE;
        DECLARE proljece_kraj DATE;

        SET ljeto_pocetak = CONCAT(EXTRACT(YEAR FROM pocetni_datum),
                        "-","06","-","21");
        SET ljeto_kraj = CONCAT(EXTRACT(YEAR FROM pocetni_datum),
                        "-","09","-","23");
        SET jesen_pocetak = CONCAT(EXTRACT(YEAR FROM pocetni_datum),
                        "-","09","-","23");
        SET jesen_kraj = CONCAT(EXTRACT(YEAR FROM pocetni_datum),
                        "-","12","-","21");
        IF MONTH(pocetni_datum)=1 OR MONTH(pocetni_datum)=2
        OR MONTH(pocetni_datum)=3 THEN
        SET zima_pocetak = CONCAT(EXTRACT(YEAR FROM pocetni_datum)-1,
                        "-","12","-","21");
        SET zima_kraj = CONCAT(EXTRACT(YEAR FROM pocetni_datum),
                        "-","03","-","21");
        ELSEIF MONTH(pocetni_datum)=12 THEN
        SET zima_pocetak = CONCAT(EXTRACT(YEAR FROM pocetni_datum),
                        "-","12","-","21");
        SET zima_kraj = CONCAT(EXTRACT(YEAR FROM pocetni_datum)+1,
                        "-","03","-","21");
    END IF;
        SET proljece_pocetak = CONCAT(EXTRACT(YEAR FROM pocetni_datum),
                        "-","03","-","21");
        SET proljece_kraj = CONCAT(EXTRACT(YEAR FROM pocetni_datum),
                        "-","06","-","21");
        SET duljina_boravka = -1;
        SELECT DATEDIFF(krajnji_datum,pocetni_datum) INTO duljina_boravka;
        IF duljina_boravka > 60 THEN RETURN 0;
END IF;
```

Part 2 - ljeto

```
CASE
WHEN pocetni_datum BETWEEN ljeto_pocetak AND ljeto_kraj
        THEN
        IF pocetni_datum=ljeto_kraj AND duljina_boravka>1
                THEN RETURN 12;
        ELSEIF pocetni_datum=ljeto_kraj AND duljina_boravka=1
                THEN RETURN 11;
        ELSEIF ljeto_kraj - pocetni_datum >  krajnji_datum - ljeto_kraj
                THEN RETURN 11;
        ELSE RETURN 12;
        END IF;
```

Part 3- jesen

```
WHEN pocetni_datum BETWEEN jesen_pocetak AND jesen_kraj
        THEN
        IF pocetni_datum=jesen_kraj AND duljina_boravka>1
                THEN RETURN 13;
        ELSEIF pocetni_datum=jesen_kraj AND duljina_boravka=1
                THEN RETURN 12;
        ELSEIF jesen_kraj - pocetni_datum >  krajnji_datum - jesen_kraj
                THEN RETURN 12;
        ELSE RETURN 13;
        END IF;
```

Part 4 – zima

```
WHEN pocetni_datum BETWEEN zima_pocetak AND zima_kraj
        THEN
        IF pocetni_datum=zima_kraj AND duljina_boravka>1
                THEN RETURN 14;
        ELSEIF pocetni_datum=zima_kraj AND duljina_boravka=1
                THEN RETURN 13;
        ELSEIF zima_kraj - pocetni_datum >  krajnji_datum - zima_kraj
                THEN RETURN 13;
        ELSE RETURN 14;
        END IF;
```

Part 5- proljece

```
WHEN pocetni_datum BETWEEN proljece_pocetak AND proljece_kraj
        THEN
        IF pocetni_datum=proljece_kraj AND duljina_boravka>1
                THEN RETURN 11;
        ELSEIF pocetni_datum=proljece_kraj AND duljina_boravka=1
                THEN RETURN 14;
        ELSEIF proljece_kraj - pocetni_datum >  krajnji_datum -
        proljece_kraj
                THEN RETURN 14;
        ELSE RETURN 11;
        END IF;
ELSE
        RETURN 0;
END CASE;

END //
DELIMITER ;
```

- Funkcija određuje pomoću datuma svako godišnje doba.

Generiranje šifra računa

```
DELIMITER //
CREATE FUNCTION gen_sifra_racuna(p_id_racun INTEGER, p_id_rezervacija
INTEGER, p_id_djelatnik INTEGER) RETURNS VARCHAR(30)
DETERMINISTIC
BEGIN
    DECLARE prvi_dio VARCHAR(10); -- zadnje 3 znamenke racuna
    DECLARE drugi_dio VARCHAR(10); -- id_rezervacija
    DECLARE treci_dio VARCHAR(10); -- id_djelatnik
    IF p_id_racun != 0 THEN
            SELECT SUBSTRING(p_id_racun, -3)
        INTO prvi_dio;
        ELSE
            SET prvi_dio = (SELECT SUBSTRING(id_racun, -3) FROM racun
ORDER BY id_racun DESC LIMIT 1)+1;
        END IF;

    SET drugi_dio = p_id_rezervacija;
    SET treci_dio = p_id_djelatnik;

    RETURN CONCAT(prvi_dio,"-",drugi_dio,"-",treci_dio);
END //
DELIMITER ;
```

- **1.dio –** zadnje 3 znamenje od ***'id_racun'***
- **2.dio –** ID rezervacije.
- **3.dio –** ID djelatnika.
- Na kraju izvođenja se sve spoji i vrati.

Broj noćenja soba

```
DELIMITER //
CREATE FUNCTION broj_nocenja_soba(p_id_soba INTEGER) RETURNS INTEGER

DETERMINISTIC
BEGIN

    DECLARE rezultat INTEGER DEFAULT 0;
    SELECT COUNT(*) AS broj_nocenja_sobe INTO rezultat
        FROM soba
        NATURAL JOIN rezervacija
        WHERE soba.id_soba = p_id_soba AND NOW()>kraj_rezervacije
        GROUP BY id_soba;
    RETURN rezultat;

END //
DELIMITER ;
```

- Vraća broj dana koliko se u pojedinoj sobi noćilo do današnjega dana.

PROCEDURE

Broj gostiju (ažuriranje)

```
DELIMITER //
CREATE PROCEDURE broj_gostiju_update(id INT)

BEGIN

        UPDATE rezervacija SET broj_osoba =
        broj_gostiju_po_rezervaciji(id)
        WHERE id_rezervacija = id;

END //
DELIMITER ;
```

- Sprema broj gostiju prilikom ažuriranja unosa.

Slobodne sobe, datum i vrsta

```
DELIMITER //
CREATE PROCEDURE slobodne_sobe_datum_i_vrsta(odabrani_pocetak DATE,
odabrani_kraj DATE, odabrana_vrsta VARCHAR(5))

BEGIN

        SELECT * FROM soba LEFT OUTER JOIN rezervacija
        ON rezervacija.id_soba = soba.id_soba
        WHERE vrsta LIKE odabrana_vrsta
        AND ((((pocetak_rezervacije > odabrani_pocetak)
        AND (pocetak_rezervacije > odabrani_kraj))
        OR ((kraj_rezervacije < odabrani_pocetak)
        AND (kraj_rezervacije < odabrani_kraj)))
        OR id_rezervacija IS NULL);

END //
DELIMITER ;
```

- Vraća sve slobodne sobe od odabranog perioda.

Broj zarade za određenu godinu po mjesecima

```
DELIMITER //
CREATE PROCEDURE
broj_zarade_za_odredenu_godinu_po_mjesecima(odabrana_godina YEAR)

BEGIN

        SELECT CONCAT('Godina: ', odabrana_godina, '.', '    Mjesec: ',
        mjeseci.mjesec, '.') AS DATUM, COALESCE(ZARADA, 0)
```

```
        FROM(SELECT DISTINCT MONTH(datum_i_vrijeme_izdavanja) AS mjesec
        FROM racun) AS mjeseci LEFT JOIN
        (SELECT MONTH(datum_i_vrijeme_izdavanja) AS mjesec,
        SUM(ukupna_cijena) AS zarada FROM racun
        WHERE YEAR(datum_i_vrijeme_izdavanja) = odabrana_godina
        GROUP BY MONTH(datum_i_vrijeme_izdavanja)) AS zarada_po_mjesecima
        ON mjeseci.mjesec=zarada_po_mjesecima.mjesec
        ORDER BY mjeseci.mjesec;

END //
DELIMITER ;
```

- prikazuje zaradu po mjesecima za odabranu godinu

Provjera OIB-a i broja osobne iskaznice za goste Republike Hrvatske

```
DELIMITER //
CREATE PROCEDURE provjera_hrv_podataka()

BEGIN

        DECLARE gost_temp INTEGER;
        DECLARE finish INTEGER DEFAULT 0;
        DECLARE gost_temp_ime VARCHAR(50);
        DECLARE gost_temp_prezime VARCHAR(50);
        DECLARE gost_temp_oib VARCHAR(50);
        DECLARE gost_temp_boi VARCHAR(50);

        DECLARE cur CURSOR FOR
            SELECT id_gost
            FROM gost
            NATURAL JOIN mjesto_prebivalista
            WHERE drzava = 'Hrvatska';

        DECLARE CONTINUE HANDLER
            FOR NOT FOUND SET finish = 1;
        OPEN cur;
        DROP TABLE IF EXISTS hrPodaci;
        CREATE TEMPORARY TABLE hrPodaci(
            id SERIAL,
            id_gost INTEGER,
            gost_ime VARCHAR(30) NOT NULL,
            gost_prezime VARCHAR(30) NOT NULL,
            oib_rezultat VARCHAR(100) NOT NULL,
            broj_osobne_iskaznice_rezultat VARCHAR(100) NOT NULL);

        iteriraj_goste: LOOP
        FETCH cur INTO gost_temp;

        IF finish = 1 THEN
            LEAVE iteriraj_goste;
        END IF;

SELECT ime INTO gost_temp_ime
            FROM gost
            WHERE id_gost = gost_temp;
```

```
        SELECT prezime INTO gost_temp_prezime
            FROM gost
            WHERE id_gost = gost_temp;
        SELECT broj_osobne_iskaznice INTO gost_temp_boi
            FROM gost
            WHERE id_gost = gost_temp;
        SELECT oib INTO gost_temp_oib
            FROM gost
            WHERE id_gost = gost_temp;

        IF LENGTH(gost_temp_oib) !=11 THEN
            IF LENGTH(gost_temp_boi)!=9 THEN
                INSERT INTO hrPodaci
        (id_gost,gost_ime,gost_prezime,oib_rezultat,broj_osobne_iskaznice_
        rezultat) VALUES(gost_temp,gost_temp_ime,gost_temp_prezime,'OIB -
        pogrešno upisan!', 'Br. osobne - pogrešno upisan');
ELSE
                INSERT INTO hrPodaci
(id_gost,gost_ime,gost_prezime,oib_rezultat,broj_osobne_iskaznice_rezulta
t) VALUES(gost_temp,gost_temp_ime,gost_temp_prezime,'OIB - pogrešno
upisan!', 'Br. osobne - OK!');
            END IF;
        ELSE
            IF LENGTH(gost_temp_boi)!=9 THEN
                INSERT INTO hrPodaci
(id_gost,gost_ime,gost_prezime,oib_rezultat,broj_osobne_iskaznice_rezulta
t) VALUES(gost_temp,gost_temp_ime,gost_temp_prezime,'OIB - OK!', 'Br.
osobne - pogrešno upisan');
            ELSE
                INSERT INTO hrPodaci
(id_gost,gost_ime,gost_prezime,oib_rezultat,broj_osobne_iskaznice_rezulta
t) VALUES(gost_temp,gost_temp_ime,gost_temp_prezime,'OIB - OK!', 'Br.
osobne - OK!');
            END IF;
        END IF;
        END LOOP iteriraj_goste;
        CLOSE cur;

        SELECT *
            FROM hrPodaci;

END //
DELIMITER ;
```

- kreira privremenu tablicu za hrvatske goste te zatim provjerava podatke jesu li hrvatski državljani.

Provjera stanja sobe

```
DELIMITER //
    CREATE PROCEDURE provjera_stanja_soba()
    BEGIN
    DECLARE soba_temp INTEGER;
    DECLARE finish INTEGER DEFAULT 0;
    DECLARE sifra_temp VARCHAR (50);
    DECLARE broj_nocenja_temp INTEGER DEFAULT 0;
    DECLARE kat_temp VARCHAR(10);
```

```
    DECLARE stanje_temp VARCHAR(100);
    DECLARE standardna_cijena_temp DECIMAL(7,2);
    DECLARE prosjecni_iznos_odrzavanja_temp DECIMAL(7,2);

    DECLARE cur CURSOR FOR
        SELECT id_soba
        FROM soba;
    DECLARE CONTINUE HANDLER
        FOR NOT FOUND SET finish = 1;
    OPEN cur;
    DROP TABLE IF EXISTS sobe_stanja;
        CREATE TEMPORARY TABLE sobe_stanja(
        id SERIAL,
        sifra VARCHAR(50),
        kat VARCHAR(10),
        standardna_cijena DECIMAL(7,2),
        broj_nocenja INTEGER NOT NULL,
        stanje VARCHAR(100),
        prosjecni_iznos_odrzavanja DECIMAL(7,2)
        );
    iteriraj_sobe: LOOP
    FETCH cur INTO soba_temp;
        IF finish = 1 THEN
            LEAVE iteriraj_sobe;
        END IF;

    SELECT sifra INTO sifra_temp
        FROM soba
        WHERE id_soba = soba_temp;
    SELECT kat INTO kat_temp
        FROM soba
        WHERE id_soba = soba_temp;
    SELECT standardna_cijena INTO standardna_cijena_temp
        FROM soba
        WHERE id_soba = soba_temp;
    SELECT broj_nocenja_soba(id_soba) INTO broj_nocenja_temp
        FROM soba
        WHERE id_soba = soba_temp;
    IF broj_nocenja_temp>0 THEN
        SET prosjecni_iznos_odrzavanja_temp =
broj_nocenja_temp*(0.1*standardna_cijena_temp);
    ELSE
        SET prosjecni_iznos_odrzavanja_temp =
0.05*standardna_cijena_temp;
    END IF;
    CASE
        WHEN broj_nocenja_temp = 0 THEN
            SET stanje_temp = "Stanje izvrsno. U sobi se nije noćilo.
Dodatnih troškova nema!";
        WHEN broj_nocenja_temp BETWEEN 1 AND 5 THEN
            SET stanje_temp = "Soba je malo korištena. Potrebno
održavanje svakih 10 dana!";
        WHEN broj_nocenja_temp BETWEEN 6 AND 15 THEN
            SET stanje_temp = "Soba je umjereno korištena. Potrebno
održavanje svakih 7 dana!";
        WHEN broj_nocenja_temp BETWEEN 16 AND 50 THEN
```

```
            SET stanje_temp = "Soba je dosta korištena. Potrebno
održavanje svaka 4 dana!";
        WHEN broj_nocenja_temp > 50 THEN
            SET stanje_temp = "Soba je jako puno korištena. Potrebno
održavanje svaki dan";
        END CASE;
        INSERT INTO sobe_stanja
(sifra,kat,standardna_cijena,broj_nocenja,stanje,prosjecni_iznos_odrzavan
ja)
            VALUES
(sifra_temp,kat_temp,standardna_cijena_temp,broj_nocenja_temp,stanje_temp
,prosjecni_iznos_odrzavanja_temp);
        END LOOP iteriraj_sobe;
        CLOSE cur;

        SELECT *
            FROM sobe_stanja;
    END //
    DELIMITER ;
```

- Koristi funkciju za broj noćenja pojedine sobe.
- Vraća u privremenoj tablici u kojoj su navedene sve sobe koje smo unijeli i broj noćenja.
- Provjerava broj dana korištenja i pomoću toga procjenjuje da li je potrebno često održavanje ili ne.

Prosjek potrošnje u proljeću

```
DELIMITER //
    CREATE PROCEDURE prosjek_potrosnje_proljece()
    BEGIN

        SELECT AVG(ukupna_cijena) FROM racun
        LEFT OUTER JOIN rezervacija ON racun.id_rezervacija =
                                       rezervacija.id_rezervacija
        WHERE rezervacija.id_sezona = '14';

    END //
    DELIMITER ;
```

- Izračunava prosjek cijena svih računa koji su izdati u proljeću

Smanjivanje cijena soba

```
DELIMITER //
CREATE PROCEDURE sm_cijena_soba()

BEGIN

        UPDATE soba
        SET standardna_cijena = standardna_cijena - (standardna_cijena *
                                                          (15/100))
            WHERE soba.id_soba IN
                (SELECT * FROM
                    (SELECT id_soba AS rez_sb FROM rezervacija
                        NATURAL JOIN soba
                        WHERE rezervacija.kraj_rezervacije < NOW() -
                                                    INTERVAL 1 MONTH
```

```
                        ) AS prv);
END //
DELIMITER ;
```

- Procedura smanjuje cijenu za onu sobu koja nije rezervirana duže od mjesec dana

Smanjivanje cijene za nepopularne usluge

```
DELIMITER //
CREATE PROCEDURE smanjivanje_cijene_nepopularne_usluge(postotak int)

BEGIN

        UPDATE dodatne_usluge
        SET cijena = cijena - cijena * postotak / 100
        WHERE id_dodatne_usluge IN
        (SELECT * FROM
                (SELECT id_dodatne_usluge FROM odabrane_usluge
                NATURAL JOIN dodatne_usluge
                GROUP BY id_dodatne_usluge
                ORDER BY COUNT(*)
                ASC
                LIMIT 3) AS p);

END //
DELIMITER ;
```

- Procedura smanjuje cijenu za nepopularne usluge tako što ih grupira i pomoću naredbe ASC uzlazno ih poreda.

Povećanje cijena za popularne usluge

```
DELIMITER //
CREATE PROCEDURE povecavanje_cijene_popularne_usluge(postotak int)

BEGIN

        UPDATE dodatne_usluge
        SET cijena = cijena + cijena * postotak / 100
        WHERE id_dodatne_usluge IN
        (SELECT * FROM
                (SELECT id_dodatne_usluge FROM odabrane_usluge
                NATURAL JOIN dodatne_usluge
                GROUP BY id_dodatne_usluge
                ORDER BY COUNT(*)
                DESC
                LIMIT 3) AS p);

END //
DELIMITER ;
```

- Procedura radi sve isto što i prijašnja procedura samo što povećava cijenu za popularne usluge poredavši ih silazno naredbom DESC

Arhiviranje

```
DELIMITER //
CREATE PROCEDURE arhiviranje()
BEGIN

        DELETE FROM rezervacija
                        WHERE kraj_rezervacije < NOW() - INTERVAL 10 YEAR;

END //
DELIMITER ;
```

- Pomoću ove procedure brišemo rezervacije i račune starije od 10 godina i premještamo ih u arhivu.

OKIDAČI

Broj osoba rezervacija

```
DELIMITER //
CREATE TRIGGER ai_broj_osoba_rezervacija
        AFTER INSERT ON odabrani_gosti
        FOR EACH ROW

BEGIN

        CALL broj_gostiju_update(new.id_rezervacija);

END//
DELIMITER ;
```

- Nakon poziva okidača ispiše nam se broj osoba koliko je upisano u rezervaciji

Ukupna cijena na računu

```
DELIMITER //
CREATE TRIGGER bi_ukupna_cijena_na_racunu
        BEFORE INSERT ON racun
        FOR EACH ROW

BEGIN

        SET new.ukupna_cijena =
        ukupna_cijena_po_rezervaciji(new.id_rezervacija);
```

```
END//
DELIMITER ;
```

Šifra sobe

```
DELIMITER //
CREATE TRIGGER bi_sifra_sobe
        BEFORE INSERT ON soba
        FOR EACH ROW

BEGIN

        SET new.sifra = gen_sifra_sobe(new.id_soba, new.kat, new.vrsta);

END //
DELIMITER ;
```

- Okidač pozove funkciju '***gen_sifra_sobe***' i unutar nje sprema novu šifru sobe

Rezervacija

```
DELIMITER //
CREATE TRIGGER bi_rezervacija
        BEFORE INSERT ON rezervacija
        FOR EACH ROW

BEGIN

        DECLARE tekst_greske TEXT;
        IF new.pocetak_rezervacije>new.kraj_rezervacije
        OR new.pocetak_rezervacije=new.kraj_rezervacije THEN
SET tekst_greske = CONCAT("Greška kod unosa rezervacije sa ID-em",
        new.id_rezervacija,". Molimo unesite ispravno datume početka i
        kraja rezervacije!");
        SIGNAL SQLSTATE '40000'
        SET MESSAGE_TEXT = tekst_greske;
END IF;
SET new.id_sezona = izracun_sezone(new.pocetak_rezervacije,
        new.kraj_rezervacije);
IF new.id_sezona = 0 THEN
SET tekst_greske = CONCAT("Greška kod unosa rezervacije sa ID-em ",
        new.id_rezervacija,". Maksimalan broj dana koji možete rezervirati
                                je 60. Molimo ispravite podatke!");
SIGNAL SQLSTATE '40000'
SET MESSAGE_TEXT = tekst_greske;
END IF;

END //
DELIMITER ;
```

- **1.dio** -Ukoliko se dogodi da kupac prilikom rezervacije stavi datum kraja rezervacije ranije nego datum početka rezervacije ili ukoliko je dan rezervacije i kraj rezervacije na isti dan okidač će ispisati grešku i tražiti ponovni unos.
- **2.dio** – pomoću funkcije izračun sezone postavljamo sezonu koja je trenutno

- **3.dio** – onemogućava rezervaciju smještaja ukoliko je raspon dana duži od 60 dana.

Šifra računa

```
DELIMITER //
CREATE TRIGGER bi_sifra_racuna
        BEFORE INSERT ON racun
        FOR EACH ROW

BEGIN

        SET new.sifra = gen_sifra_racuna(new.id_racun,
                new.id_rezervacija, new.id_djelatnik);

END //
DELIMITER ;
```

- Okidač pozove funkciju generiranje šifra računa i unutar nje sprema novu šifru računa.

Datum zaposlenja

```
DELIMITER //
CREATE TRIGGER bi_djelatnik_gd
        BEFORE INSERT ON djelatnik
        FOR EACH ROW

BEGIN

        IF new.datum_zaposljenja > NOW() THEN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Pogrešan datum unesen!';
        END IF;

END //
DELIMITER ;
```

- Okidač sprema datum zaposlenja i onemogućava unaprijed unošenje datuma.

Stvaranje sigurnosne kopije za tablicu podataka, ako je djelatnik uklonjen

```
DELIMITER //
CREATE TRIGGER ad_backup_zaposlenika
        AFTER DELETE ON djelatnik
        FOR EACH ROW

BEGIN

        CREATE TEMPORARY TABLE backup_djelatnik LIKE djelatnik;
        INSERT INTO backup_djelatnik(id_djelatnik, ime, prezime,
        id_zvanje, datum_zaposljenja)
        VALUES (OLD.id_djelatnik, OLD.ime, OLD.prezime, OLD.id_zvanje,
        OLD.datum_zaposljenja);

END //
DELIMITER ;
```

- Okidač sprema osobne podatke zaposlenika u slučaju da je potrebno ponovno doći do njih.

Provjerava da li je cijena preskupa za dodatne usluge (unos)

```
DELIMITER //
CREATE TRIGGER provjera_cijene_dodatne_usluge_i
    BEFORE INSERT ON dodatne_usluge
    FOR EACH ROW
BEGIN
    IF new.cijena > 2000 THEN
        SIGNAL SQLSTATE '40000'
        SET MESSAGE_TEXT = "Ne možete unijeti tako skupu uslugu";
    END IF;
END //
DELIMITER ;
```

- Okidač prilikom unosa podataka za dodatne usluge provjerava da li je cijena usluge veća od najveće koju je hotel odredio da će biti najveća za bilo koju dodatnu uslugu.

Provjerava da li je cijena preskupa za dodatne usluge (ažuriranje)

```
DELIMITER //
CREATE TRIGGER provjera_cijene_dodatne_usluge_u
        BEFORE UPDATE ON dodatne_usluge
        FOR EACH ROW

BEGIN

    IF new.cijena > 2000 THEN
        SIGNAL SQLSTATE '40000'
        SET MESSAGE_TEXT = "Ne možete unijeti tako skupu uslugu";
    END IF;

END //
DELIMITER ;
```

- Okidač radi isto što i prijašnji samo provjerava prilikom ažuriranja podataka

Provjerava da li je cijena preskupa za aranžman (ažuriranje)

```
DELIMITER //
CREATE TRIGGER provjera_aranzman_u
        BEFORE UPDATE ON aranzman
        FOR EACH ROW

BEGIN

    IF new.cijena > 2000 THEN
        SIGNAL SQLSTATE '40000'
        SET MESSAGE_TEXT = "Ne možete unijeti tako skupi aranžman";
    END IF;

END //
DELIMITER ;
```

- Okidač prilikom ažuriranja podataka za aranžman provjerava da li je cijena aranžmana veća od najveće koju je hotel odredio da će biti najveća za bilo koji aranžman.

Provjerava da li je cijena preskupa za aranžman (unos)

```
DELIMITER //
CREATE TRIGGER provjera_aranzman_i
        BEFORE INSERT ON aranzman
        FOR EACH ROW
BEGIN

    IF new.cijena > 2000 THEN
        SIGNAL SQLSTATE '40000'
        SET MESSAGE_TEXT = "Ne možete unijeti tako skupi aranžman";
    END IF;

END //
DELIMITER ;
```

- Okidač radi isto što i okidač *'provjera_aranzman_u'* samo provjerava prilikom unosa novih podataka.

Provjera godina rođenja

```
DELIMITER //
CREATE TRIGGER provjera_godina_rođenja_gost
 BEFORE INSERT ON gost
 FOR EACH ROW
BEGIN
 IF new.datum_rodenja > NOW() THEN
        SIGNAL SQLSTATE '40000'
        SET MESSAGE_TEXT = "Datum rođenja ne može biti veći od trenutnog
datuma!";
 END IF;
END//
DELIMITER ;
```

- Okidač onemogućava unos za datum rođenja gosta ukoliko je datum veći od trenutnoga.

Provjera unosa za ime i prezime kod kupaca

```
DELIMITER //
CREATE TRIGGER special_char_check
 BEFORE INSERT ON gost
 FOR EACH ROW
BEGIN

 DECLARE i INT DEFAULT 1 ;
 DECLARE j INT DEFAULT 1 ;
 DECLARE slovo CHAR;
 DECLARE greska TEXT;
      for_petlja: LOOP
                SET slovo = SUBSTRING(new.ime,i,1);
          IF (HEX(slovo)<HEX("'") || HEX(slovo)>HEX("'")) &&
                        (HEX(slovo)<HEX('A') || HEX(slovo)>HEX('Z')) &&
```

```
            (HEX(slovo)<HEX('a') || HEX(slovo)>HEX('z')) &&
                        (HEX(slovo)<HEX('À') || HEX(slovo)>HEX('Ö')) &&
                        (HEX(slovo)<HEX('Ø') || HEX(slovo)>HEX('ö')) &&
                        (HEX(slovo)<HEX('ø') || HEX(slovo)>HEX('ƿ')) &&
                        (HEX(slovo)<HEX('Ǆ') || HEX(slovo)>HEX('ɥ')) &&
                        (HEX(slovo)<HEX('Ɛ') || HEX(slovo)>HEX('Ɉ')) &&
                        (HEX(slovo)<HEX('Ɬ') || HEX(slovo)>HEX('ƀ')) &&
                        (HEX(slovo)<HEX('ʯ') || HEX(slovo)>HEX('ƿ')) &&
                        (HEX(slovo)<HEX('ˁ') || HEX(slovo)>HEX('□')) &&
                        (HEX(slovo)<HEX('') || HEX(slovo)>HEX(''))
         THEN
                        SET greska = CONCAT("Greška: Ne možete unijeti ime
", new.ime, " jer sadrži zabranjen znak ", slovo);
                        SIGNAL SQLSTATE '40000'
                        SET MESSAGE_TEXT = greska;
                END IF;
         IF i=LENGTH(new.ime) THEN
            LEAVE for_petlja;
         END IF;
         SET i = i + 1;
   END LOOP for_petlja;

   for_petlja2: LOOP
                SET slovo = SUBSTRING(new.prezime,j,1);
         IF (HEX(slovo)<HEX("'") || HEX(slovo)>HEX("'")) &&
            (HEX(slovo)<HEX('A') || HEX(slovo)>HEX('Z')) &&
            (HEX(slovo)<HEX('a') || HEX(slovo)>HEX('z')) &&
                        (HEX(slovo)<HEX('À') || HEX(slovo)>HEX('Ö')) &&
                        (HEX(slovo)<HEX('Ø') || HEX(slovo)>HEX('ö')) &&
                        (HEX(slovo)<HEX('ø') || HEX(slovo)>HEX('ƿ')) &&
                        (HEX(slovo)<HEX('Ǆ') || HEX(slovo)>HEX('ɥ')) &&
                        (HEX(slovo)<HEX('Ɛ') || HEX(slovo)>HEX('Ɉ')) &&
                        (HEX(slovo)<HEX('Ɬ') || HEX(slovo)>HEX('ƀ')) &&
                        (HEX(slovo)<HEX('ʯ') || HEX(slovo)>HEX('ƿ')) &&
                        (HEX(slovo)<HEX('ˁ') || HEX(slovo)>HEX('□')) &&
                        (HEX(slovo)<HEX('') || HEX(slovo)>HEX(''))
         THEN
                        SET greska = CONCAT("Greška: Ne možete unijeti
prezime ", new.prezime, " jer sadrži zabranjen znak ", slovo);
                        SIGNAL SQLSTATE '40000'
                        SET MESSAGE_TEXT = greska;
                END IF;
         IF j=LENGTH(new.prezime) THEN
            LEAVE for_petlja2;
         END IF;
         SET j = j + 1;
   END LOOP for_petlja2;

END//
DELIMITER ;
```

- Okidač onemogućava unošenje posebnih znakova i simbola prilikom unošenja podataka za ime i prezime gosta kao npr.
    - **Nedozvoljeno** - @°&%/) $“)#=
    - **Dozvoljeno** – John Malkovich

Zabrana za unos brojeva u adresu

```
    DELIMITER //
    CREATE TRIGGER bi_adresa_zabrana
        BEFORE INSERT ON mjesto_prebivalista
        FOR EACH ROW
    BEGIN
        IF NEW.adresa REGEXP '^[0-9]*$' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Adresa ne smije
sadržavati samo brojke!';
        END IF;
    END //
    DELIMITER ;
```

- Okidač onemogućava unošenje adrese, ako adresa sadržava samo brojeve kao npr.
    - **Nedozvoljeno** – 12345
    - **Dozvoljeno** – Kačićeva 114

bd_rezervacija

```
DELIMITER //
CREATE TRIGGER bd_rezervacija
        BEFORE DELETE ON rezervacija
        FOR EACH ROW
BEGIN
        DELETE FROM racun WHERE racun.id_rezervacija=old.id_rezervacija;
    IF old.kraj_rezervacije < NOW() - INTERVAL 10 YEAR THEN
                INSERT INTO arhiva_rezervacija (id_rezervacija,  id_soba,
id_gost,  id_sezona, id_aranzman, pocetak_rezervacije, kraj_rezervacije,
broj_osoba)
                        VALUES
                (old.id_rezervacija,
                old.id_soba,
                old.id_gost,
                old.id_sezona,
                old.id_aranzman,
                old.pocetak_rezervacije,
                old.kraj_rezervacije,
                old.broj_osoba);
    END IF;
END//
DELIMITER ;
```

- Prilikom brisanja rezervacije, okidač briše i račun vezan za tu rezervaciju.
- Ukoliko je rezervacija starija od 10 godina, okidač ju sprema u arhivu.

bd_racun

```
DELIMITER //
CREATE TRIGGER bd_racun
        BEFORE DELETE ON racun
        FOR EACH ROW
BEGIN
        DECLARE kraj_rezervacije_provjera DATE;

        SELECT kraj_rezervacije INTO kraj_rezervacije_provjera FROM
rezervacija WHERE old.id_rezervacija=rezervacija.id_rezervacija;
```

```
    IF kraj_rezervacije_provjera < NOW() - INTERVAL 10 YEAR THEN
                INSERT INTO arhiva_racun (id_racun, sifra, id_rezervacija,
id_djelatnik,  datum_i_vrijeme_izdavanja, ukupna_cijena)
                VALUES
                (old.id_racun,
                old.sifra,
                old.id_rezervacija,
                old.id_djelatnik,
                old.datum_i_vrijeme_izdavanja,
                old.ukupna_cijena);
        END IF;
END//
DELIMITER ;
```

- Ukoliko se račun briše provjerava se da li je stariji od 10 godina, ako je vrijednost točna sprema se u arhivu.

ai_cetiri_gosta

```
DELIMITER //
        CREATE TRIGGER ai_cetiri_gosta
         AFTER INSERT ON odabrani_gosti
         FOR EACH ROW
         FOLLOWS ai_broj_osoba_rezervacija
        BEGIN
         IF (SELECT rezervacija.broj_osoba FROM rezervacija WHERE
rezervacija.broj_osoba > 4) THEN
                SIGNAL SQLSTATE '40000'
                SET MESSAGE_TEXT = "Dozvoljeno je maksimalno četiri gosta
po sobi!";
         END IF;
END //
DELIMITER ;
```

- Okidač onemogućava unos za broj gostiju, ako isti ima više od četiri gosta na rezervaciji.

Provjera usluge

```
DELIMITER //
CREATE TRIGGER provjera_usluge
    BEFORE INSERT ON odabrane_usluge
    FOR EACH ROW
BEGIN
   DECLARE broj_dana INT;
   DECLARE ime_usluge VARCHAR(50);
   DECLARE greska TEXT;
   SELECT DATEDIFF(kraj_rezervacije, pocetak_rezervacije) INTO broj_dana
FROM rezervacija
           WHERE id_rezervacija = new.id_rezervacija;
   SELECT naziv INTO ime_usluge FROM dodatne_usluge
           WHERE id_dodatne_usluge = new.id_dodatne_usluge;

   IF ime_usluge LIKE '%/dan' AND broj_dana < new.kolicina THEN
                SET greska = CONCAT("Greska!", " Za odabranu uslugu id: ",
new.id_odabrane_usluge,". Unesite kolicinu usluge u trajanju dana,
premalo dana boravka previše dana usluge!");
```

```
        SIGNAL SQLSTATE '40000'
        SET MESSAGE_TEXT = greska;
    END IF;
    IF ime_usluge LIKE '%/sat' AND broj_dana < new.kolicina/24 THEN
                SET greska = CONCAT("Greska!", " Za odabranu uslugu id: ",
new.id_odabrane_usluge,". Unesite kolicinu usluge u trajanju dana,
premalo dana boravka previše sati usluge!");
        SIGNAL SQLSTATE '40000'
        SET MESSAGE_TEXT = greska;
    END IF;
END //
DELIMITER ;
```

- Provjerava može li gost potrošiti sve usluge koje će platiti, odnosno, ako će biti u hotelu 5 dana, ne smije rezervirati uslugu fitnesa na 6 dana, nego samo do 5 dana.
- Okidač obuhvaća usluge koje se rezerviraju i na sate.

Provjera rezervacije soba

```
DELIMITER //
CREATE TRIGGER provjera_rezervacija_soba
    AFTER INSERT ON odabrani_gosti
    FOR EACH ROW
    FOLLOWS ai_cetiri_gosta
BEGIN
    DECLARE soba_vrsta VARCHAR(10);
    DECLARE kapacitet INT;
    DECLARE id_sobe INT;
    DECLARE greska TEXT;
    SELECT id_soba INTO id_sobe FROM rezervacija NATURAL JOIN
odabrani_gosti WHERE id_rezervacija=new.id_rezervacija;
    SELECT vrsta INTO soba_vrsta FROM soba WHERE id_soba = id_sobe;
    IF soba_vrsta LIKE 'SGL' THEN SET kapacitet = 1;
    ELSEIF soba_vrsta LIKE 'DBL' THEN SET kapacitet = 2;
    ELSEIF soba_vrsta LIKE 'TRPL' THEN SET kapacitet = 3;
    ELSEIF soba_vrsta LIKE 'QDPL' THEN SET kapacitet = 4;
    END IF;
    IF kapacitet < (SELECT broj_osoba FROM rezervacija NATURAL JOIN
odabrani_gosti WHERE id_rezervacija=new.id_rezervacija) THEN
                SET greska = CONCAT("Greska! Za rezervaciju id: ",
new.id_rezervacija, ". Premala soba izaberi drugu sa više kapaciteta!");
                SIGNAL SQLSTATE '40000'
        SET MESSAGE_TEXT = greska;
    END IF;
END //
DELIMITER ;
```

- Okidač zabranjuje unos neadekvatne sobe za broj gostiju koji su rezervirali. Kao npr.

- **Nedozvoljeno** – Jednokrevetna soba / 4 osobe na rezervaciji
- **Dozvoljeno** – Jednokrevetna soba / 1 osoba na rezervaciji

Transakcije

Izrada rezervacije

```
DELIMITER //
CREATE PROCEDURE izradaRezervacije()
BEGIN
    DECLARE mp_id INTEGER DEFAULT 0;

    DECLARE booker_ime VARCHAR(50);
    DECLARE booker_prezime VARCHAR(50);
    DECLARE booker_oib VARCHAR(50);
    DECLARE booker_boi VARCHAR(50);
    DECLARE booker_dr DATE;
    DECLARE booker_id INTEGER;

    DECLARE od_soba_id INTEGER;
    DECLARE od_aranzman_id INTEGER;
    DECLARE od_pocetak_rezervacije DATE;
    DECLARE od_kraj_rezervacije DATE;
    DECLARE rezervacija_id INTEGER;

    DECLARE flag1 INTEGER DEFAULT 0;
    DECLARE flag2 INTEGER DEFAULT 0;
    DECLARE gosti_temp_br INTEGER DEFAULT 0;

    DECLARE id_gost_temp INTEGER;
    DECLARE ime_gost_temp VARCHAR(30);
    DECLARE prezime_gost_temp VARCHAR(50);
    DECLARE oib_gost_temp VARCHAR(50);
    DECLARE boi_gost_temp VARCHAR(50);
    DECLARE dr_gost_temp DATE;

    DECLARE odabrane_usluge_br INTEGER DEFAULT 0;
    DECLARE id_odabrane_usluge_temp INTEGER;
    DECLARE id_dodatne_usluge_temp INTEGER;
    DECLARE odabrane_usluge_kolicina_temp INTEGER;

    DECLARE cur1 CURSOR FOR
        SELECT id_gost
        FROM gosti_temp;
    DECLARE cur2 CURSOR FOR
        SELECT id_odabrane_usluge
        FROM temp_odabrane_usluge;
    DECLARE EXIT HANDLER FOR 1062
     BEGIN
        ROLLBACK;
        SELECT 'Došlo je do greške, duplicate entry';
     END;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
     BEGIN
        ROLLBACK;
        SELECT 'Došlo je do greške, izrada rezervacije je obustavljena!';
     END;
#_____________________________________________________________________
    SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
    START TRANSACTION;
        #Mjesto prebivalista insert
```

```
        INSERT INTO mjesto_prebivalista
(drzava,grad,postanski_broj,adresa)
            SELECT drzava,grad,postanski_broj,adresa FROM
odabrano_mjesto_prebivalista;
        #Fetch id_mjesto_prebivalista INTO mp_id
        SELECT id_mjesto_prebivalista INTO mp_id
            FROM mjesto_prebivalista
            ORDER BY id_mjesto_prebivalista DESC
            LIMIT 1;
        #Fetch ime INTO booker_ime
        SELECT ime INTO booker_ime
            FROM temp_booker;
        #Fetch prezime INTO booker_prezime
        SELECT prezime INTO booker_prezime
            FROM temp_booker;
        #Fetch oib INTO booker_oib
        SELECT oib INTO booker_oib
            FROM temp_booker;
        #Fetch broj_osobne_iskaznice INTO booker_boi
        SELECT broj_osobne_iskaznice INTO booker_boi
            FROM temp_booker;
        #Fetch datum_rodenja INTO booker_dr
        SELECT datum_rodenja INTO booker_dr
            FROM temp_booker;
        #Insert booker into GOST
        INSERT INTO gost
(ime,prezime,oib,broj_osobne_iskaznice,id_mjesto_prebivalista,datum_roden
ja)

VALUES(booker_ime,booker_prezime,booker_oib,booker_boi,mp_id,booker_dr);
        #Fetch booker ID INTO booker_id
        SELECT id_gost INTO booker_id
            FROM gost
            ORDER BY id_gost DESC
            LIMIT 1;
               #Fetch id_soba INTO od_soba
        SELECT id_soba INTO od_soba_id
            FROM temp_rezervacija;
        #Fetch id_aranzman INTO od_aranzman_id
        SELECT id_aranzman INTO od_aranzman_id
            FROM temp_rezervacija;
        #Fetch pocetak_rezervacije INTO od_pocetak_rezervacije
        SELECT pocetak_rezervacije INTO od_pocetak_rezervacije
            FROM temp_rezervacija;
        #Fetch kraj_rezervacije INTO od_kraj_rezervacije
        SELECT kraj_rezervacije INTO od_kraj_rezervacije
            FROM temp_rezervacija;
        #Insert rezervacija
            -- ID sezona 11 -> jer se racuna preko bi_rezervacija
triggera
            -- broj_osoba DEFAULT -> jer se racuna preko
ai_broj_osoba_rezervacija triggera
        INSERT INTO rezervacija
(id_soba,id_gost,id_sezona,id_aranzman,pocetak_rezervacije,kraj_rezervaci
je,broj_osoba)
```

```
VALUES(od_soba_id,booker_id,11,od_aranzman_id,od_pocetak_rezervacije,od_k
raj_rezervacije,DEFAULT);
        #Fetch id_rezervacija INTO rezervacija_id
        SELECT id_rezervacija INTO rezervacija_id
            FROM rezervacija
            ORDER BY id_rezervacija DESC
            LIMIT 1;
        #_____________________________________________
        #Spremanje broja dodatnih gostiju u gosti_temp_br
        BEGIN
        DECLARE CONTINUE HANDLER
            FOR NOT FOUND
            SET flag1 = 1;
        SELECT COUNT(*) INTO gosti_temp_br
            FROM gosti_temp;
        IF gosti_temp_br > 0 THEN
            OPEN cur1;
               iteriraj_gosti_temp: LOOP
            FETCH cur1 INTO id_gost_temp;
                IF flag1 = 1 THEN
                    LEAVE iteriraj_gosti_temp;
                END IF;
             #Fetch temp_gost ime
             SELECT ime INTO ime_gost_temp
                FROM gosti_temp
                WHERE id_gost = id_gost_temp;
            #Fetch temp_gost prezime
             SELECT prezime INTO prezime_gost_temp
                FROM gosti_temp
                WHERE id_gost = id_gost_temp;
             #Fetch temp_gost oib
             SELECT oib INTO oib_gost_temp
                FROM gosti_temp
                WHERE id_gost = id_gost_temp;
            #Fetch temp_gost broj_osobne_iskaznice
             SELECT broj_osobne_iskaznice INTO boi_gost_temp
                FROM gosti_temp
                WHERE id_gost = id_gost_temp;
            #Fetch temp_gost datum rodenja
             SELECT datum_rodenja INTO dr_gost_temp
                FROM gosti_temp
                WHERE id_gost = id_gost_temp;

                    INSERT INTO gost
(ime,prezime,oib,broj_osobne_iskaznice,id_mjesto_prebivalista,datum_roden
ja)
                        VALUES
(ime_gost_temp,prezime_gost_temp,oib_gost_temp,boi_gost_temp,mp_id,dr_gos
t_temp);
                    INSERT INTO odabrani_gosti (id_gost,id_rezervacija)
                        VALUES (LAST_INSERT_ID(),rezervacija_id);
            END LOOP iteriraj_gosti_temp;
            CLOSE cur1;
        END IF;
        END;
                #
```

```
        #Spremanje broja odabranih usluga u odabrane_usluge_br
        BEGIN
        DECLARE CONTINUE HANDLER
            FOR NOT FOUND
            SET flag2 = 1;
        SELECT COUNT(*) INTO odabrane_usluge_br
            FROM temp_odabrane_usluge;
        IF odabrane_usluge_br > 0 THEN
            OPEN cur2;
            iteriraj_temp_odabrane_usluge: LOOP
            FETCH cur2 INTO id_odabrane_usluge_temp;
                IF flag2 = 1 THEN
                    LEAVE iteriraj_temp_odabrane_usluge;
                END IF;
            #Fetch id_dodatne_usluge
             SELECT id_dodatne_usluge INTO id_dodatne_usluge_temp
                FROM temp_odabrane_usluge
                WHERE id_odabrane_usluge = id_odabrane_usluge_temp;
            #Fetch kolicina
             SELECT kolicina INTO odabrane_usluge_kolicina_temp
                FROM temp_odabrane_usluge
                WHERE id_odabrane_usluge = id_odabrane_usluge_temp;

                    INSERT INTO odabrane_usluge
(id_dodatne_usluge,id_rezervacija,kolicina)
                        VALUES
(id_dodatne_usluge_temp,rezervacija_id,odabrane_usluge_kolicina_temp);
        END LOOP iteriraj_temp_odabrane_usluge;
        CLOSE cur2;
        END IF;
        END;
            COMMIT;
            SELECT CONCAT('Transaction commited!');
END //
DELIMITER ;
```

- Poziva se preko internet stranice kada korisnik potvrdi sve podatke
- Provjerava jesu li svi podaci točni i ukoliko jesu rezervacija se izrađuje.

Izrada računa

```
DELIMITER //
CREATE PROCEDURE izradaRacuna(IN p_id_rezervacija INTEGER, IN
p_djelatnik_ime VARCHAR(30), IN p_djelatnik_prezime VARCHAR(50))
BEGIN
    DECLARE id_djelatnika INTEGER DEFAULT 0;
    SELECT id_djelatnik INTO id_djelatnika
        FROM djelatnik
        WHERE ime = p_djelatnik_ime AND prezime = p_djelatnik_prezime;
    BEGIN
    DECLARE EXIT HANDLER FOR 1216
        BEGIN
        ROLLBACK;
                CASE
                    WHEN (p_id_rezervacija NOT IN (SELECT id_rezervaciaj
FROM rezervacija))
```

```
                        THEN SELECT CONCAT('Unijeli ste ID za nepostojeću
rezervaciju!');
                    WHEN (id_djelatnika = 0)
                        THEN SELECT CONCAT('Djelatnik nije pronađen');
                END CASE;
        END;
        DECLARE EXIT HANDLER FOR SQLEXCEPTION
     BEGIN
        ROLLBACK;
        SELECT 'Došlo je do greške, izrada računa je obustavljena!';
     END;
        END;

    SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
    START TRANSACTION;
        INSERT INTO racun
(sifra,id_rezervacija,id_djelatnik,datum_i_vrijeme_izdavanja,ukupna_cijen
a)
            VALUES
(DEFAULT,p_id_rezervacija,id_djelatnika,NOW(),DEFAULT);
    COMMIT;
    SELECT CONCAT('Transaction commited!');
END //
DELIMITER ;
```

- Poziva se preko internet stranice kada korisnik potvrdi sve podatke
- Djelatnik je zadužen za izradu računa.

Pogledi

Prikaz godišnjih zarada

```
CREATE VIEW prikaz_zarade_za_sve_godine AS
SELECT YEAR(datum_i_vrijeme_izdavanja) AS GODINA, (SUM(ukupna_cijena)) AS
ZARADA FROM racun
                GROUP BY YEAR(datum_i_vrijeme_izdavanja)
         ORDER BY datum_i_vrijeme_izdavanja;
```

Prikaz rezervacije i odabranih usluga

```
CREATE VIEW prikaz_rezervacija_i_odabranih_usluga AS
        SELECT id_rezervacija,
                    DATEDIFF(kraj_rezervacije,pocetak_rezervacije) AS
broj_dana,
            id_odabrane_usluge,
            naziv,
            kolicina,
            cijena*kolicina AS cijena
        FROM rezervacija
    NATURAL JOIN odabrane_usluge
    NATURAL JOIN dodatne_usluge;
```

Pregled gostiju u pojedinoj rezervaciji

```
CREATE VIEW pregled_gostiju_u_pojedinoj_rezervaciji AS
SELECT CONCAT('Rezervacija:  ', id_rezervacija) AS id_rezervacija,
            CONCAT('Booker ime:  ', ime) AS ime, CONCAT('Booker prezime:
', prezime) AS prezime,
            CONCAT('Booker OIB:  ', OIB) AS OIB, CONCAT('Booker broj osobne
iskaznice:  ', broj_osobne_iskaznice) AS broj_osobne_iskaznice,
        CONCAT('Booker datum rođenja:  ', DATE_FORMAT(datum_rodenja,
'%d.%M.%Y')) AS datum_rodenja,
        CONCAT('Booker država:  ', drzava) AS drzava, CONCAT('Booker grad:
', grad) AS grad,
        CONCAT('Booker poštanski broj:  ', postanski_broj) AS
postanski_broj, CONCAT('Booker adresa:  ', adresa) AS adresa
         FROM rezervacija
    NATURAL JOIN gost
    NATURAL JOIN mjesto_prebivalista
UNION
SELECT CONCAT('Rezervacija:  ', r.id_rezervacija) AS id_rezervacija,
            CONCAT('Odabrani gost ime:  ', ime) AS ime, CONCAT('Odabrani
gost prezime:  ', prezime) AS prezime,
        CONCAT('Odabrani gost OIB:  ', OIB) AS OIB, CONCAT('Odabrani gost
broj osobne iskaznice:  ', broj_osobne_iskaznice) AS
broj_osobne_iskaznice,
        CONCAT('Odabrani gost datum rođenja:  ',
DATE_FORMAT(datum_rodenja, '%d.%M.%Y')) AS datum_rodenja,
        CONCAT('Odabrani gost država:  ', drzava) AS drzava,
CONCAT('Odabrani gost grad:  ', grad) AS grad,
        CONCAT('Odabrani gost poštanski broj:  ', postanski_broj) AS
postanski_broj, CONCAT('Odabrani gost adresa:  ', adresa) AS adresa
         FROM rezervacija AS r
```

```
        JOIN odabrani_gosti AS o_g ON r.id_rezervacija =
o_g.id_rezervacija
        JOIN gost ON gost.id_gost = o_g.id_gost
    NATURAL JOIN mjesto_prebivalista
ORDER BY id_rezervacija;
```

Prikaz gostiju – ukupni podaci

```
CREATE VIEW prikaz_gostiju_ukupni_podaci AS
SELECT gost.ime, gost.prezime, COUNT(*) AS broj_rezervacija,
SUM(ukupna_cijena) AS ukupno_potroseno
        FROM gost
        NATURAL JOIN rezervacija
        NATURAL JOIN racun
        GROUP BY gost.id_gost;
```

- Prikazuje ime i prezime gosta, broj rezervacije koju je gost napravio skupa sa ukupnim iznosom na računu

Prikaz računa

```
CREATE VIEW racun_prikaz AS
    SELECT racun.sifra,racun.datum_i_vrijeme_izdavanja,
           CONCAT(djelatnik.ime, ' ', djelatnik.prezime) AS racun_izdao,
           CONCAT (gost.ime, ' ', gost.prezime) AS
rezervirao,racun.ukupna_cijena
        FROM racun
        NATURAL JOIN rezervacija
        NATURAL JOIN gost
        INNER JOIN djelatnik
        ON djelatnik.id_djelatnik = racun.id_djelatnik;
```

Rezervacija bez računa

```
CREATE VIEW rezervacija_bez_racuna AS
    SELECT rezervacija.id_rezervacija AS rezervacija_id, CONCAT(gost.ime,
' ', gost.prezime) AS rezervirao,
           aranzman.naziv AS aranzman, soba.sifra AS sifra_sobe,
sezona.naziv AS sezona,
           rezervacija.pocetak_rezervacije, rezervacija.kraj_rezervacije,
rezervacija.broj_osoba
        FROM rezervacija
        LEFT JOIN racun
        ON rezervacija.id_rezervacija = racun.id_rezervacija
        NATURAL JOIN gost
        INNER JOIN  sezona
            ON sezona.id_sezona = rezervacija.id_sezona
        INNER JOIN aranzman
            ON aranzman.id_aranzman = rezervacija.id_aranzman
        INNER JOIN soba
            ON soba.id_soba = rezervacija.id_soba
        WHERE racun.id_rezervacija IS NULL;
```

- Prikazuje podatke o gostu koji je napravio rezervaciju, sezonu, sobu i aranžman
- Prikazuje samo rezervacije za koje još nije izdan račun.

Broj rezervacija po sezoni

```
CREATE VIEW broj_rezervacija_po_sezoni AS
SELECT
    SUM(IF(id_sezona = '11', 1, 0)) AS ljetne_rezervacije,
    SUM(IF(id_sezona = '12', 1, 0)) AS jesenske_rezervacije,
    SUM(IF(id_sezona = '13', 1, 0)) AS zimske_rezervacije,
    SUM(IF(id_sezona = '14', 1, 0)) AS proljetne_rezervacije
FROM rezervacija;
```

Zaposlenici zaposleni duže od 20 godina

```
CREATE VIEW zaposlenici_zaposleni_duze_od_20_godina AS
SELECT *
    FROM djelatnik
    WHERE datum_zaposljenja < NOW() - INTERVAL 20 YEAR;
```

Gosti s najdužim boravkom

```
CREATE VIEW goste_koji_imaju_najduzi_boravak AS
SELECT gost.ime, gost.prezime, DATEDIFF(rezervacija.kraj_rezervacije,
rezervacija.pocetak_rezervacije) AS duljina_boravka
    FROM gost, rezervacija
        WHERE gost.id_gost = rezervacija.id_gost
        ORDER BY duljina_boravka
        DESC LIMIT 10;
```

Najčešće države gostiju

```
CREATE VIEW najcesce_drzave_gostiju AS
SELECT drzava
    FROM mjesto_prebivalista
    GROUP BY drzava
    ORDER BY COUNT(drzava)
    DESC LIMIT 3;
```

Najpopularniji aranžmani

```
CREATE VIEW najpopularniji_aranzmani AS
        SELECT aranzman.naziv, COUNT(rezervacija.id_aranzman) AS
broj_aranzmana
            FROM aranzman
            LEFT JOIN rezervacija ON (aranzman.id_aranzman =
rezervacija.id_aranzman)
            GROUP BY aranzman.id_aranzman;
```

Najprofitabilniji zaposlenici

```
CREATE VIEW najprinosniji_zaposlenici AS
    SELECT djelatnik.ime AS Ime, djelatnik.prezime AS Prezime,
zvanje.naziv AS Radno_Mjesto, zvanje.plaća_HRK AS Plaća
        FROM djelatnik
        NATURAL JOIN zvanje
```

```
        WHERE (zvanje.plaća_HRK > 15000)
        GROUP BY id_djelatnik;
```

Djelatnik s najvećim brojem računa

```
CREATE VIEW djelatnik_sa_najvećim_brojem_računa AS
SELECT d.* FROM racun
        NATURAL JOIN djelatnik AS d
        GROUP BY id_djelatnik
        ORDER BY COUNT(*) DESC
        LIMIT 1;
```

Najpopularnija usluga

```
CREATE VIEW najpopularnija_usluga AS
SELECT d.* FROM odabrane_usluge
        NATURAL JOIN dodatne_usluge AS d
        GROUP BY id_dodatne_usluge
        ORDER BY COUNT(*) DESC
        LIMIT 1;
```

Najpopularnija soba

```
CREATE VIEW najpopularnija_soba AS
SELECT s.* FROM rezervacija
        NATURAL JOIN soba AS s
        WHERE kraj_rezervacije > NOW() - INTERVAL 2 MONTH
        GROUP BY id_soba
        ORDER BY COUNT(*) DESC
        LIMIT 1;
```

Trenutni gosti u hotelu

```
CREATE VIEW gosti_trenutno_u_hotelu AS
SELECT id_rezervacija, ime, prezime, sifra FROM rezervacija
        NATURAL JOIN gost
        NATURAL JOIN soba
        WHERE pocetak_rezervacije <= NOW() AND kraj_rezervacije > NOW();
```