**PROJEKTNI ZADATAK**

Pojednostavljeni modul Bankarskog sistema

*TEHNIČKA DOKUMENTACIJA*

*Melisa Čehajić SARAJEVO, 2020.god.*

SADRŽAJ

[OPIS PROBLEMA 2](#_Toc33478942)

[ZADATAK 1: 2](#_Toc33478943)

[ZADATAK 2: 4](#_Toc33478944)

[IZRADA PROJEKTNOG ZADATKA 5](#_Toc33478945)

[ZADATAK 1: 5](#_Toc33478946)

[A) Komitenti (Tabela koja sadrži podatke o klijentima banke): 6](#_Toc33478947)

[B) Konta (Šifarnik kontnog plana banke): 6](#_Toc33478948)

[C) Kursevi (Tabela sa kursevima koja se puni svaki dan i samo jednom dnevno): 6](#_Toc33478949)

[D) Nalozi 7](#_Toc33478950)

[E) Stavke (Tabela koja sadrži sva knjigovodstevna knjiženja u Banci): 7](#_Toc33478951)

[ZADATAK 2: 9](#_Toc33478952)

[1. Sekvenca za punjenje polja „sifra“ u tabeli Komitenti 9](#_Toc33478953)

[3. Sekvenca za punjenje polja „nalog“ u tabeli Nalozi 10](#_Toc33478954)

[4. Kontrola za tabelu Stavke 10](#_Toc33478955)

[7. Skripte za popunjavanje tabela pod tačkama a), b) i c) i testiranje zadatka 2 11](#_Toc33478956)

[8. Funkcija koja kreirati proknjižen nalog u tabeli nalozi 14](#_Toc33478957)

[9. Procedura koja u tabeli Stavke kreira nalog sa dvije stavke 15](#_Toc33478958)

[10. Upiti (*Queries*) 16](#_Toc33478959)

# OPIS PROBLEMA

Kao primjer pojednostavljenog modula Bankarskog sistema, potrebno je dizajnirati sistem koji će omogućiti evidenciju klijenata i njegovih knjigovodstvenih naloga, kao i za preglede evidentiranih podataka.

U tu svrhu, potrebno je realizirati sljedeće:

## ZADATAK 1:

Kreirati sljedeće tabele i relacije (veze) između njih. Posebnu požnju obratiti na primarne i strane ključeve.

1. **Komitenti (Tabela koja sadrži podatke o klijentima banke):**

Komitent (Jedinstve šifra komitenta max 6 karaktera)

Naziv\_komitenta (Naziv komitenta)

Vrsta\_komitenta (Dozvojene vrijdnosti su F – Fizičko lice i P – Pravno lice)

JMBG (Jedinstveni matični broj fizičkog lica)

Identifikacioni\_broj (Identifikacioni broj pravnog lica)

Adresa (Adresa komitenta)

Mjesto (Mjesto komitenta)

Drzava (Država komitenta)

1. **Konta (Šifarnik kontnog plana banke):**

Konto (Jedinstvena šifra konta max. 10 cifara)

Naziv\_konta (Naziv konta)

1. **Kursevi (Tabela sa kursevima koja se puni svaki dan i samo jednom dnevno):**

Valuta (Jedisntvena oznaka valute)

Datum kursa (Datum za koji vrijedi kursna lista)

Kurs\_prodajni (Iznos kursa prodajni)

Kurs\_kupovni (Iznos kursa kupovni)

Kurs\_srednji (Iznos srednjeg kursa)

1. **Nalozi (Tabela naloga koji sadrže stavke iz tabele stavki):**

- Nalog (Jedinstvena oznaka naloga)

- Datum\_naloga (datum naloga -mora biti današnji datum)

- Proknjizen (Ozaka da li je nalog validan – Dozvoljene vrijednosti: NULL ili N ili D)

1. **Stavke (Tabela koja sadrži sva knjigovodstevna knjiženja u Banci):**

- Nalog ( broj naloga za tu stavku)

- Stavka ( redni broj stavke naloga)

- Datum\_naloga (datum naloga -mora biti današnji datum)

- Komitent (šifra komitenta iz sifarnika Komitenata)

- Konto (šifra konta iz kontnog plana)

- Valuta (oznaka valute)

- Datum\_kursa (Datum za koji vrijedi kursna lista)

- Kurs\_srednji (Iznos srednjeg kursa)

- Iznos\_duguje\_KM ( Protuvrijednost dugovnog iznos u KM valuti)

- Iznos\_potrazuje\_KM ( Protuvrijednost potražnog iznos u KM valuti)

- Iznos\_duguje\_DEV ( Dugovni iznos u valuti)

- Iznos\_potrazuje\_DEV ( Potražni iznos u valuti)

- Proknjizena (Oznaka da li je nalog validan – Dozvoljene vrijednosti: NULL ili N ili D)

## ZADATAK 2:

1. Kreirati sekvencu kojom će se puniti polje sifra u tabeli komitenti.
2. Napraviti da se prilikom inserta novog sloga u tabelu komitenti automatski puni kolona sifra na osnovu sekvence iz tačke 1.
3. Kreirati sekvencu za polje Nalog u tabeli nalozi.
4. Napraviti kontrolu da se u tabelu stavke ne može insertovati slog kod koga je i potrazni i dugovni iznos jednak nuli.
5. Kreirati log tabelu log u kojoj će se smještati sve izmjene (Insert, Update, Delete) po tabelama iz zadatka 1. Osmisliti kako da se sve logira u jednu tabelu. Kreirati index kako bi pretraga bila brza. Log tabela mora da ima podatak o tome koji korisniku je uradio izmjenu (Insert, Update, Delete).
6. Nad tabelama iz zadatka 1 dodati da se izmjene logiraju u tabelu log.
7. Pripremiti skripte za popunjavanje tabela pod tačkama a, b i c iz zadatka 1 i za testiranje zadataka 2.
8. Napraviti funkciju koja će kreirati proknjižen nalog u tabeli nalozi i vratiti kao vrijednost šifru naloga.
9. Napraviti proceduru koja će u tabeli stavke kreirati nalog sa dvije stavke. Za broj naloga u tabeli stavke koristiti funkciju iz tačke 8.
10. Pripremiti upite za sljedeće izvještaje:
11. Prikazati na dnevnom nivou ukupan dugovni i potražni promet (Iznos\_duguje\_DEV i Iznos\_potrazuje\_DEV) u zadnjih mjesec dana.
12. Prikazati ukupan broj fizičkih i pravnih lica iz tabele komitenata.
13. Prikazati datum kursa za valutu USD (Američki dolar) kada je ta valuta imala najveći iznos srednjeg kursa u tabeli kursevi.
14. Prikazati broj komitenata po mjestu stanovanja i poredati prikaz od najvećeg broja ka najmanjem.

Rješenje zadanog problema realizovano je u Microsoft SQL Server Management Studio 18 alatu.

# IZRADA PROJEKTNOG ZADATKA

## ZADATAK 1:

U nastavku će biti prikazani kodni isječci pomoću kojih su kreirane zadane tabele, kao i njihov ER dijagram.

*(Na ilustracijama kolona tabela crvenom bojom obojena su polja koja predstavljaju primarni ključ, dok su zelenom bojom obojena polja koja predstavljaju strane ključeve.)*

### Komitenti (Tabela koja sadrži podatke o klijentima banke):

CREATE TABLE Komitenti (

Komitent INT IDENTITY(1,2) PRIMARY KEY CHECK(LEN(Komitent)<=6),

Naziv\_komitenta VARCHAR (20) NOT NULL,

Vrsta\_Komitenta char(1) CHECK(Vrsta\_Komitenta='F' OR Vrsta\_Komitenta='P'),

JMBG nchar(13) CHECK(LEN(JMBG)=13) NOT NULL,

Identifikacioni\_broj int NOT NULL,

Adresa varchar(30) NOT NULL,

Mjesto varchar(30) NOT NULL,

Drzava varchar(30) NOT NULL

);

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Komitent | Naziv\_komitenta | Vrsta\_komitenta | JBMG | Identif\_broj | Adresa | Mjesto | Drzava |

### Konta (Šifarnik kontnog plana banke):

CREATE TABLE Konta(

Konto int PRIMARY KEY CHECK(LEN(Konto)<=10),

Naziv\_konta varchar(20)

);

|  |  |
| --- | --- |
| Konto | Naziv\_konta |

### Kursevi (Tabela sa kursevima koja se puni svaki dan i samo jednom dnevno):

CREATE TABLE Kursevi(

Valuta nchar(3) CHECK(LEN(Valuta)=3),

Datum\_kursa DATETIME DEFAULT GETDATE(),

Kurs\_prodajni DECIMAL(8,6),

Kurs\_kupovni DECIMAL(8,6),

Kurs\_srednji DECIMAL(8,6)

PRIMARY KEY (Valuta, Datum\_kursa)

);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Valuta | Datum\_kursa | Kurs\_prodajni | Kurs\_kupovni | Kurs\_srednji |

### Nalozi

CREATE TABLE Nalozi(

Nalog INT IDENTITY(1,2) PRIMARY KEY,

Datum\_Naloga DATETIME DEFAULT GETDATE() NOT NULL,

Proknjizen char(1) CHECK(Proknjizen='N' OR Proknjizen='D')

);

|  |  |  |
| --- | --- | --- |
| Nalog | Datum\_naloga | Proknjizen |

### Stavke (Tabela koja sadrži sva knjigovodstevna knjiženja u Banci):

CREATE TABLE Stavke(

Nalog INT,

Stavka INT,

Datum\_naloga DATETIME DEFAULT GETDATE(),

Komitent INT ,

Konto int CHECK(LEN(Konto)<=10),

Valuta nchar(3) CHECK(LEN(Valuta)=3),

Datum\_kursa DATETIME DEFAULT GETDATE(),

Kurs\_srednji DECIMAL(8,6),

Iznos\_duguje\_KM DECIMAL (10,2),

Iznos\_potrazuje\_KM DECIMAL (10,2),

PRIMARY KEY (Nalog, Stavka);

CONSTRAINT DUG\_POTR\_KM CHECK (Iznos\_duguje\_KM!=0 OR Iznos\_potrazuje\_KM!=0),

Iznos\_duguje\_DEV DECIMAL (10,2),

Iznos\_potrazuje\_DEV DECIMAL (10,2),

CONSTRAINT DUG\_POTR\_DEV CHECK (Iznos\_duguje\_DEV!=0 OR Iznos\_potrazuje\_DEV!=0),

Proknjizena char(1) CHECK(Proknjizena='N' OR Proknjizena='D'),

FOREIGN KEY(Nalog) REFERENCES Nalozi(Nalog),

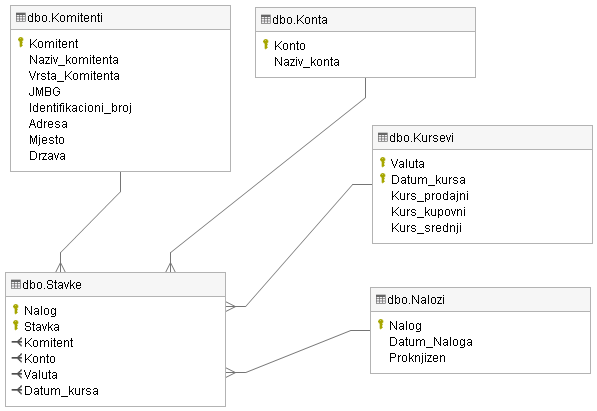
FOREIGN KEY(Komitent) REFERENCES Komitenti(Komitent) ON DELETE SET NULL,

FOREIGN KEY(Konto) REFERENCES Konta(Konto) ON DELETE SET NULL,

FOREIGN KEY(Valuta, Datum\_kursa) REFERENCES Kursevi(Valuta, Datum\_kursa) ON DELETE SET NULL,

);

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nalog | Stavka | Datum\_naloga | Komitent | Konto | Valuta | Datum kursa | Kurs srednji | Duguje KM | Potrazuje KM | Duguje  DEV | Potrazuje DEV | Proknjizena |



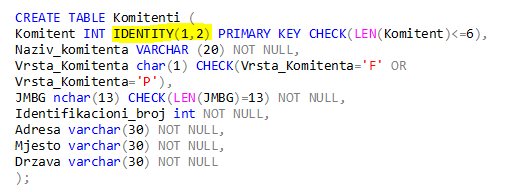
ER (*Entity Relationship*) dijagram kreiranih tabela

## ZADATAK 2:

### Sekvenca za punjenje polja „sifra“ u tabeli Komitenti

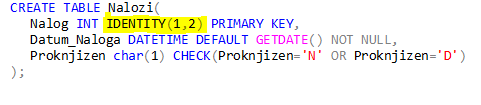
Budući da korištena verzija alata Microsoft SQL Server Management Studio-a ne podržava **SEQUENCE** kao ključnu riječ za kreiranje sekvenci, to jeste, kreiranje sekvenci na taj način nije moguće, u ovom projektnom zadatku, korištena je ključna riječ **IDENTITY**, unutar same definicije tabele, kojom je specificirano kako će se puniti polje nad kojim se koristi ta opcija.

IDENTITY *(a,b)* predstavlja način za automatsko generisanje vrijednosti polja na koje se primjenjuje, gdje *a* predstavlja prvu unesenu vrijednost, dok *b* predstavlja inkrementalnu vrijednost u odnosu na prethodnu vrijednost)



### Sekvenca za punjenje polja „nalog“ u tabeli Nalozi

Realizacija navedenog postignuta je na isti način kao i u zadatku 1.



### Kontrola za tabelu Stavke

Napraviti kontrolu da se u tabelu stavke ne može insertovati slog kod koga je i potražni i dugovni iznos jednak nuli.

Ova kontrola postignuta je na način da se u definiciji tabele Stavke dodalo ograničenje (*Constraint*) koje je prikazano ispod:



### Skripte za popunjavanje tabela pod tačkama a), b) i c) i testiranje zadatka 2

1. **Skripta za popunjavanje tabele „*Komitenti“:***

DECLARE @count INT

SET @count = 1

DECLARE @count1 INT

SET @count1 = 0

DECLARE @tip1 CHAR(1)

SET @tip1 = 'F'

DECLARE @tip2 CHAR(1)

SET @tip2 = 'P'

DECLARE @jmbg nchar(13)

SET @jmbg = '2807995195001';

DECLARE @idbroj INT

DECLARE @idbrojmin INT

DECLARE @idbrojmax INT

SET @idbrojmin = 1

SET @idbrojmax = 10000

WHILE @count <= 1000

BEGIN

SELECT @idbroj = Round(((@idbrojmax - @idbrojmin) \* RAND ()) + @idbrojmin, 0)

INSERT INTO Komitenti VALUES ( 'Komitent - ' + CAST(@count AS NVARCHAR(10)), @tip1, @jmbg, @idbroj, 'Adresa - ' + CAST(@count AS NVARCHAR(10)), 'Mjesto - ' + CAST(@count AS NVARCHAR(10)), 'Drzava - ' + CAST(@count AS NVARCHAR(10)));

INSERT INTO Komitenti VALUES ( 'Komitent - ' + CAST(@count1 AS NVARCHAR(10)), @tip2, @jmbg, @idbroj, 'Adresa - ' + CAST(@count1 AS NVARCHAR(10)), 'Mjesto - ' + CAST(@count1 AS NVARCHAR(10)), 'Drzava - ' + CAST(@count1 AS NVARCHAR(10)));

PRINT @count

SET @count = @count + 1

SET @count1 = @count1 + 1000

END

1. **Skripta za popunjavanje tabele „*Konta“:***

DECLARE @Konto INT

DECLARE @Konto\_min INT

DECLARE @Konto\_max INT

SET @Konto\_min = 1

SET @Konto\_max = 10000

DECLARE @count INT

SET @count=1

WHILE @count <= 1000

BEGIN

SELECT @Konto = Round(((@Konto\_max - @Konto\_min) \* RAND ()) + @Konto\_min, 0)

INSERT INTO Konta VALUES (@Konto, 'Konto - ' + CAST(@count AS NVARCHAR(10)))

PRINT @count

SET @count = @count + 1

END

SELECT \* FROM Konta;

CREATE TABLE Komitenti (

Komitent INT IDENTITY(1,2) PRIMARY KEY CHECK(LEN(Komitent)<=6),

Naziv\_komitenta VARCHAR (20) NOT NULL,

Vrsta\_Komitenta char(1) CHECK(Vrsta\_Komitenta='F' OR Vrsta\_Komitenta='P'),

JMBG nchar(13) CHECK(LEN(JMBG)=13) NOT NULL,

Identifikacioni\_broj int NOT NULL,

Adresa varchar(30) NOT NULL,

Mjesto varchar(30) NOT NULL,

Drzava varchar(30) NOT NULL

);

1. **Skripta za popunjavanje tabele „*Kursevi“:***

DELETE FROM Kursevi;

SELECT \* FROM Kursevi;

--SELECT RAND()\*(b-a)+a;

INSERT INTO Kursevi VALUES ('EUR', DEFAULT, RAND()\*(1.96-1.95)+1.95,RAND()\*(1.96-1.95)+1.95,RAND()\*(1.96-1.95)+1.95);

INSERT INTO Kursevi VALUES ('AUD', DEFAULT, RAND()\*(1.22-1.2)+1.2,RAND()\*(1.22-1.2)+1.2,RAND()\*(1.22-1.2)+1.2);

INSERT INTO Kursevi VALUES ('CAD', DEFAULT, RAND()\*(1.37-1.36)+1.36,RAND()\*(1.37-1.36)+1.36,RAND()\*(1.37-1.36)+1.36);

INSERT INTO Kursevi VALUES ('HRK', DEFAULT, RAND()\*(26.3-26.2)+26.2,RAND()\*(26.3-26.2)+26.2,RAND()\*(26.3-26.2)+26.2);

INSERT INTO Kursevi VALUES ('CZK', DEFAULT, RAND()\*(0.08-0.07)+0.07,RAND()\*(0.08-0.07)+0.07,RAND()\*(0.08-0.07)+0.07);

INSERT INTO Kursevi VALUES ('DKK', DEFAULT, RAND()\*(0.265-0.262)+0.262,RAND()\*(0.265-0.262)+0.262,RAND()\*(0.265-0.262)+0.262);

INSERT INTO Kursevi VALUES ('HUF', DEFAULT, RAND()\*(0.583-0.58)+0.58,RAND()\*(0.583-0.58)+0.58,RAND()\*(0.583-0.58)+0.58);

INSERT INTO Kursevi VALUES ('JPY', DEFAULT, RAND()\*(1.65-1.63)+1.63,RAND()\*(1.65-1.63)+1.63,RAND()\*(1.65-1.63)+1.63);

INSERT INTO Kursevi VALUES ('NOK', DEFAULT, RAND()\*(0.196-0.194)+0.194,RAND()\*(0.196-0.194)+0.194,RAND()\*(0.196-0.194)+0.194);

INSERT INTO Kursevi VALUES ('SEK', DEFAULT, RAND()\*(0.186-0.184)+0.184,RAND()\*(0.186-0.184)+0.184,RAND()\*(0.186-0.184)+0.184);

INSERT INTO Kursevi VALUES ('CHF', DEFAULT, RAND()\*(1.85-1.83)+1.83,RAND()\*(1.85-1.83)+1.83,RAND()\*(1.85-1.83)+1.83);

INSERT INTO Kursevi VALUES ('TRY', DEFAULT, RAND()\*(0.3-0.29)+0.29,RAND()\*(0.3-0.29)+0.29,RAND()\*(0.3-0.29)+0.29);

INSERT INTO Kursevi VALUES ('GBP', DEFAULT, RAND()\*(2.36-2.34)+2.34,RAND()\*(2.36-2.34)+2.34,RAND()\*(2.36-2.34)+2.34);

INSERT INTO Kursevi VALUES ('USD', DEFAULT, RAND()\*(1.83-1.8)+1.8,RAND()\*(1.83-1.8)+1.8,RAND()\*(1.83-1.8)+1.8);

INSERT INTO Kursevi VALUES ('RUB', DEFAULT, RAND()\*(0.029-0.028)+0.028,RAND()\*(0.029-0.028)+0.028,RAND()\*(0.029-0.028)+0.028);

INSERT INTO Kursevi VALUES ('CNY', DEFAULT, RAND()\*(0.26-0.25)+0.25,RAND()\*(0.26-0.25)+0.25,RAND()\*(0.26-0.25)+0.25);

INSERT INTO Kursevi VALUES ('RSD', DEFAULT, RAND()\*(1.67-1.65)+1.65,RAND()\*(1.67-1.65)+1.65,RAND()\*(1.67-1.65)+1.65);

### Funkcija koja kreirati proknjižen nalog u tabeli nalozi

Budući da realizacija tražene funkcije nije uspjela do kraja (nejasno je kako kreirati neki slog u tabeli, kada SQL User-Defined funkcije ne dozvoljavaju naredbe kao što su UPDATE ili INSERT, dakle dozvoljena je samo SELECT naredba, i u skladu s tim napravljena je funkcija koja djelimično obavlja ono što se tražilo.

Funkcija čiji je kod naveden u nastavku, prima broj naloga, provjerava da li je vrijednost polja Proknjizen = 'D' (iz tabele Nalozi), ukoliko jest, vraća kao vrijednost šifru naloga, ukoliko to nije slučaj, ispisuje grešku da traženi nalog nije proknjižen.

CREATE FUNCTION fnProknjizen(@Nalog INT)

RETURNS INT

AS

BEGIN

IF (( SELECT Proknjizen FROM Nalozi WHERE Nalog=@Nalog) = 'D')

BEGIN

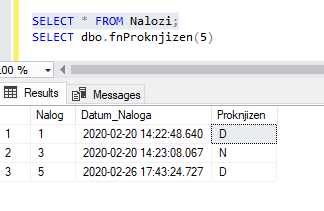
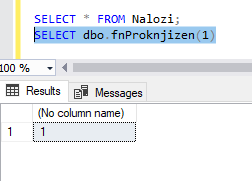
RETURN (SELECT Nalog FROM Nalozi WHERE Nalog=@Nalog)

END

RETURN 'Ovaj nalog nije proknjizen!';

END

Funkcija se poziva naredbom SELECT dbo.fnProknjizen(broj\_naloga).

### Procedura koja u tabeli Stavke kreira nalog sa dvije stavke

Procedura (*Stored Procedure)* kreirana je koristeći se kodom prikazanim u nastavku.

CREATE PROCEDURE stavke\_naloga

@broj\_naloga INT

AS

BEGIN

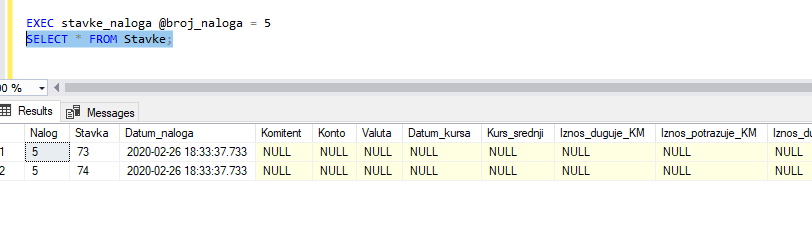
DECLARE @br\_naloga INT;

SET @br\_naloga = dbo.fnProknjizen (@broj\_naloga);

INSERT INTO Stavke VALUES (@broj\_naloga,DEFAULT,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);

INSERT INTO Stavke VALUES (@broj\_naloga,DEFAULT,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL);

END



### Upiti (*Queries*)

1. **Ukupan broj fizičkih i pravnih lica iz tabele komitenata**

Ovaj upit može se realizovati na način da se jednom naredbom vraćaju i ukupan broj fizičkih i ukupan broj pravnih lica:

SELECT COUNT(Vrsta\_komitenta) AS 'Ukupan broj fizickih i pravnih lica', Vrsta\_komitenta

FROM Komitenti

GROUP BY Vrsta\_komitenta;

Ili, kao pojedinačni upiti:

SELECT COUNT(Vrsta\_komitenta) AS 'Broj fizickih lica'

FROM Komitenti

WHERE Vrsta\_komitenta = 'F';

SELECT COUNT(Vrsta\_komitenta) AS 'Broj pravnih lica'

FROM Komitenti

WHERE Vrsta\_komitenta = 'P';

1. **Ukupni dugovni i potražni promet u zadnjih mjesec dana, na dnevnom nivou**

SELECT TOP 30 Datum\_naloga AS DatumNaloga, SUM(Iznos\_duguje\_DEV) AS UkupanDugovniPromet, SUM(Iznos\_potrazuje\_DEV) AS UkupanPotrazniPromet

FROM Stavke

GROUP BY Datum\_naloga

*Napomena:* Kako SQL Server Management Studio 18 ne podržava DATE kao tip podatka, u ovom projektnom zadatku korišten je tip podatka DATETIME, koji vraća datum i tačno vrijeme. Ovaj upit korišten u navedenoj verziji programa neće vratiti ono što se očekuje, jer će iste datume sa različitim vremenima posmatrati kao dva odvojena datuma, međutim, ukoliko bi se Datum\_naloga deklarisao kao tip DATE (i koristio u nekoj drugoj verziji programa koji podržava DATE), tad bi ovaj upit vratio ono što se od njega očekuje.

1. **Datum kursa za valutu USD kada je ta valuta imala najveći iznos srednjeg kursa u tabeli *Kursevi***

SELECT Datum\_kursa FROM Kursevi WHERE Kurs\_srednji=(SELECT MAX(Kurs\_srednji) FROM Kursevi WHERE Valuta='USD');

1. **Broj komitenata po mjestu stanovanja, poredati prikaz od najvećeg ka najmanjem broju**

SELECT COUNT(Mjesto) AS 'Broj komitenata po mjestu stanovanja', Mjesto

FROM Komitenti

GROUP BY Mjesto

Order by Mjesto;