

Neka je data tabela **Radnik(ID, ime, plata, datum, grad, oblast)** kreirana sledećim SQL skriptom:

```
create table Radnik(  
ID int,  
ime nvarchar (10),  
plata int,  
datum datetime,  
grad nvarchar (10),  
oblast char (1));
```

```
insert into radnik (ID, ime, plata, datum, grad, oblast)  
values (1, 'Janko', 40420, '02/01/94', 'Beograd', 'C');
```

```
insert into radnik (ID, ime, plata, datum, grad, oblast)  
values (2, 'Boban', 14420, '01/02/95', 'Subotica', 'S');
```

```
insert into radnik (ID, ime, plata, datum, grad, oblast)  
values (3, 'Sandra', 24020, '12/03/96', 'Nis', 'J');
```

```
insert into radnik (ID, ime, plata, datum, grad, oblast)  
values (4, 'Lidija', 40620, '11/04/97', 'Beograd', 'C');
```

```
insert into radnik (ID, ime, plata, datum, grad, oblast)  
values (5, 'Darko', 80026, '10/05/98', 'Smederevo', 'C');
```

```
insert into radnik (ID, ime, plata, datum, grad, oblast)  
values (6, 'Dejan', 70060, '09/06/99', 'Nis', 'J');
```

```
insert into radnik (ID, ime, plata, datum, grad, oblast)  
values (7, 'Ana', 90620, '08/07/00', 'Beograd', 'C');
```

```
insert into radnik (ID, ime, plata, datum, grad, oblast)  
values (8, 'Sasa', 26020, '07/08/01', 'Kraljevo', 'Z');
```

```
insert into radnik (ID, ime, plata, datum, grad, oblast)  
values (9, 'Marija', 60020, '06/09/02', 'Beograd', 'C');
```

Primer 1:

- a. Kreirati T-SQL proceduru **DodajZapis** kojom se dodaje jedan red u prethodno kreiranu tabelu **Radnik**. Kao argument se procedure prosleđuju ID i ime zaposlenog.
- b. Demonstrirati upotrebu kreirane procedure.

```
-- procedura
CREATE PROCEDURE DodajZapis
    @ID int,
    @Ime varchar(50)
AS
INSERT Radnik (ID, Ime) VALUES (@ID, @Ime)

-- upotreba
EXEC DodajZapis 20, 'DBA'
```

Primer 2:

- a. Kreirati T-SQL proceduru **UbacilzmeniRadnika** koja za zadato ime zaposlenog i zadatu platu najpre proverava da li u tabeli **Radnik** postoji zaposleni sa tim imenom. Ukoliko takav zaposleni postoji, onda se iznos njegove plate postavlja na novu vrednost, a u suprotnom se dodaje novi zaposleni sa imenom i platom koji su dati kao argument procedure. U slučaju bilo kakve greške, treba je prijaviti i napustiti proceduru.
- b. Demonstrirati upotrebu kreirane procedure.
- c. Ukloniti definiciju procedure **UbacilzmeniRadnika**.

```
CREATE PROCEDURE UbacilzmeniRadnika
    @Ime nvarchar(50),
    @Plata int
AS
IF EXISTS (SELECT * FROM Radnik WHERE ime = @Ime)
    UPDATE Radnik
    SET Plata = @Plata
    WHERE Ime = @Ime
ELSE
    INSERT INTO Radnik (ime, plata)
    SELECT @Ime, @Plata
IF @@Error <> 0
    RAISERROR 50001 'UbacilzmeniRadnika problem'

EXEC UbacilzmeniRadnika 'Marija', 999999

DROP PROCEDURE UbacilzmeniRadnika
```

Primer 3:

Kreirati T-SQL skript koji najpre uklanja proceduru **dohvatiPodatke** ukoliko ona postoji, a potom je ponovo kreira tako da se ponaša kao „parametrizovani pogled“ nad tabelom **Radnik**.

```
IF EXISTS(SELECT * FROM sys.objects WHERE type = 'P' AND ime = 'dohvatiPodatke')
    DROP PROCEDURE dohvatiPodatke
```

```
CREATE PROCEDURE dohvatiPodatke
AS
SELECT * FROM Radnik
```

Primer 4:

- a. Kreirati T-SQL proceduru **dodajNovog** kojom se dodaje jedan red u prethodno kreiranu tabelu **Radnik**. Kao argument procedure se prosleđuju ID, grad i oblast zaposlenog. Procedura treba da doda novog zaposlenog samo ako za dati oblast u tabeli sa zaposlenima već postoji bar jedan zaposleni iz tog oblata.
- b. Demonstrirati upotrebu kreirane procedure, najpre na primeru validnih argumenata, a potom na primeru nevalidnih argumenata.

```
DROP PROCEDURE dodajNovog
```

```
CREATE PROCEDURE dodajNovog @ID nvarchar(20), @Grad nchar(50), @OblastID nchar(1)
AS
BEGIN
    DECLARE @Broj int

    SELECT @Broj = COUNT(*)
    FROM Radnik
    WHERE oblast = @OblastID

    IF @Broj < 1
        RAISERROR ('OblastID nije validan. Proverite uneti argument i ponovite zahtev.', 11, 1)
    ELSE
        INSERT INTO Radnik (ID, grad, oblast)
        VALUES (@ID, @Grad, @OblastID)
END
```

```
EXEC dodajNovog @ID=99, @Grad = 'DDD', @OblastID='N'
```

```
EXEC dodajNovog @ID=99, @Grad = 'DDD', @OblastID='A'
```

[SQL] #50000: OblastID nije validan. Proverite uneti argument i ponovite zahtev.

Primer 5:

- a. Kreirati T-SQL funkciju **brojRadnika**, koja nema argumente, a koja vraća broj redova u tabeli **Radnik**.
- b. Demonstrirati upotrebu funkcije **brojRadnika**.

```
CREATE FUNCTION brojRadnika ()  
RETURNS int  
AS  
BEGIN  
    RETURN (SELECT COUNT(*) FROM Radnik)  
END
```

-- da bi saznali naziv seme NNN

```
SELECT table_schema  
FROM information_schema.tables
```

-- iskoristiti naziv seme NNN pri pozivu funkcije (T-SQL to zahteva pri pozivu skalarnih funkcija)

```
SELECT NNN. brojRadnika () AS '# Radnika'
```

Primer 6:

- a. Kreirati T-SQL funkciju **tabelaPlata**, koja kao argument prihvata celobrojnu vrednost za iznos plate, a kao rezultat vraća tabelu sa informacijama o onim zaposlenima iz tabele **Radnik** čija je plata veća od zadate vrednost.
- b. Demonstrirati upotrebu funkcije **tabelaPlata** za slučaj plate 2500.

DROP function tabelaPlata

```
CREATE FUNCTION tabelaPlata (@iznosPlate int)  
RETURNS TABLE  
AS  
RETURN (SELECT * FROM Radnik where plata > @iznosPlate)
```

```
SELECT * FROM tabelaPlata(2500)
```

Primer 7:

- a. Kreirati T-SQL funkciju **listaRadnika**, koja kao argument prihvata celobrojnu vrednost koja predstavlja ID zaposlenog, a kao rezultat vraća tabelu **Lista**(ID, ime, plata) sa informacijama o zaposlenom iz tabele **Radnik** sa zadatom vrednošću ID. Ukoliko se za ID pri pozivu prosledi NULL, funkcija treba da vrati tabelu **Lista** sa informacijama o svim zaposlenima iz tabele **Radnik**.
- b. Demonstrirati upotrebu funkcije **listaRadnika**.

```
CREATE FUNCTION listaRadnika (@ID Int)
RETURNS @Lista Table
( ID int,
  Ime nvarchar(50),
  Plata int
)
AS
BEGIN
  IF @ID IS NULL
  BEGIN
    INSERT INTO @Lista (ID, Ime, Plata)
    SELECT ID, ime, plata
    FROM Radnik
  END
  ELSE
  BEGIN
    INSERT INTO @Lista (ID, Ime, Plata)
    SELECT ID, ime, plata
    FROM Radnik
    WHERE ID = @ID
  END
  RETURN
END
```

```
SELECT * FROM listaRadnika(1)
```

Primer 8:

- a. Kreirati T-SQL okidač **testOkidac** koji nakon bilo koje od operacija INSERT, UPDATE, DELETE nad tabelom **Radnik** prikazuje novi i stari sadržaj onih redova nad kojima je izvršena promena.
- b. Demonstrirati upotrebu kreiranog trigeru na primerima odgovarajućih INSERT, UPDATE, DELETE naredbi.

```
CREATE TRIGGER testOkidac
ON radnik
FOR UPDATE, INSERT, DELETE
AS
SELECT * FROM inserted
SELECT * FROM deleted
```

```
INSERT Radnik (ID, ime) VALUES (31, 'Ranko')
```

```
UPDATE Radnik
SET ime = 'Branko'
WHERE ID = 3
```

```
DELETE FROM Radnik WHERE ID = 3
```

Primer 9:

- a. Kreirati T-SQL funkciju **stampajPodatke**, koja nema argumente, koja koristeći kursor, kao rezultat vraća tabelu **PodaciRadnika**(RedniBr, ID) koja je popunjena sa informacijama za prvih 5 zaposlenih iz tabele **Radnik** (popunjava se redni broj i njihov ID).
- b. Funkciju opisanu u prethodnoj tački implementirati bez upotrebe kursora.

```
CREATE FUNCTION stampajPodatke()
RETURNS @PodaciRadnika Table
( RedniBr int,
  ID int
)
AS
BEGIN
DECLARE @IdRad int, @RedBr int
DECLARE ListaRad CURSOR FOR SELECT TOP 5 ID FROM Radnik
OPEN ListaRad
FETCH NEXT FROM ListaRad INTO @IdRad
SET @RedBr = 0
WHILE @@FETCH_STATUS = 0
BEGIN
  SET @RedBr = @RedBr + 1
  INSERT INTO @PodaciRadnika (RedniBr, ID) VALUES (@RedBr, @IdRad)
  FETCH NEXT FROM ListaRad INTO @IdRad
END
CLOSE ListaRad
DEALLOCATE ListaRad
RETURN
END
```

```
CREATE FUNCTION stampajPodatke()
RETURNS @PodaciRadnika Table
( RedniBr int,
  ID int
)
AS
BEGIN
DECLARE @IdRad int, @RedBr int
SELECT TOP 1 @IdRad=ID FROM Radnik
SET @RedBr = 0
WHILE @RedBr < 5
BEGIN
  SET @RedBr = @RedBr + 1
  INSERT INTO @PodaciRadnika (RedniBr, ID) VALUES (@RedBr, @IdRad)
  SELECT TOP 1 @IdRad=ID FROM Radnik WHERE ID > @IdRad
END
RETURN
END
```