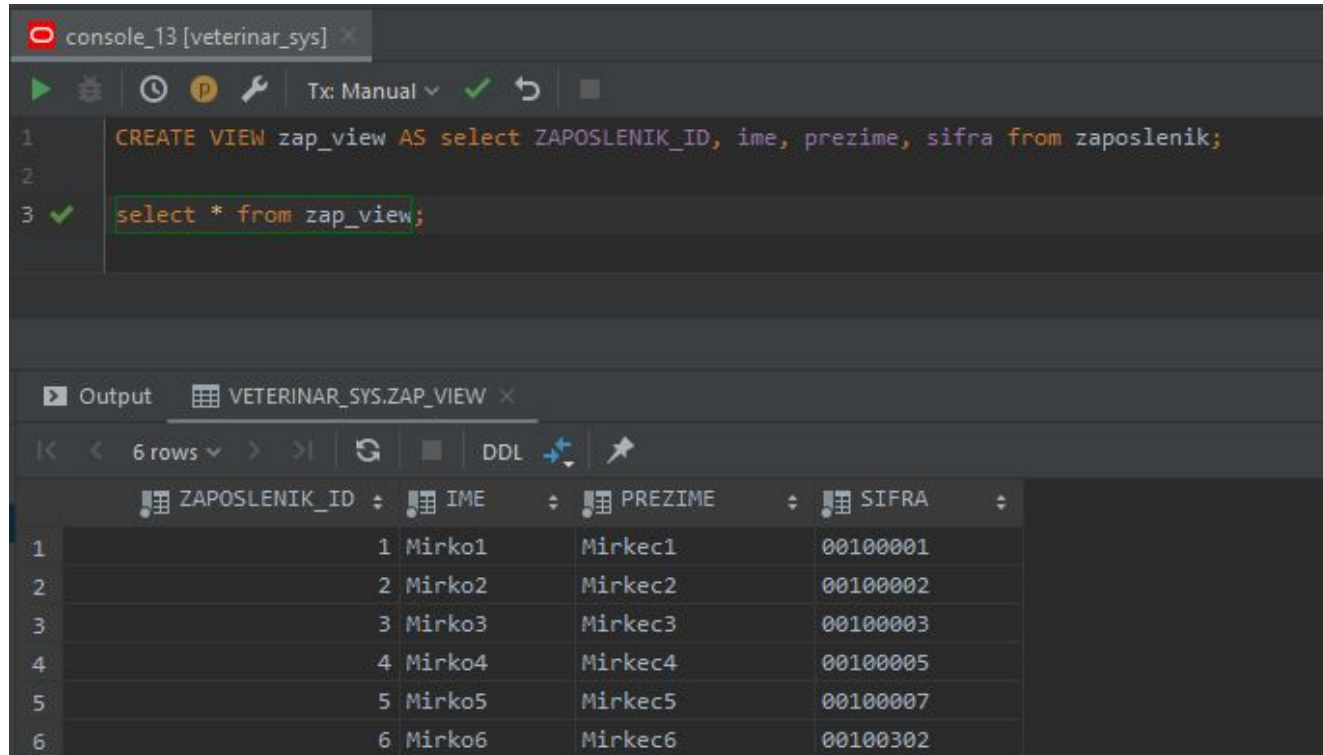


# Denis Kodrin, programer/komunikator

- Slanje mailova profesoru, organizacija sastanaka
- Podaci za tablice vezane za inspekcije,
- Funkcije, procedure, pogledi:
  - Voditelj odjela
  - Računovođa

Voditelj odjela - Upiti

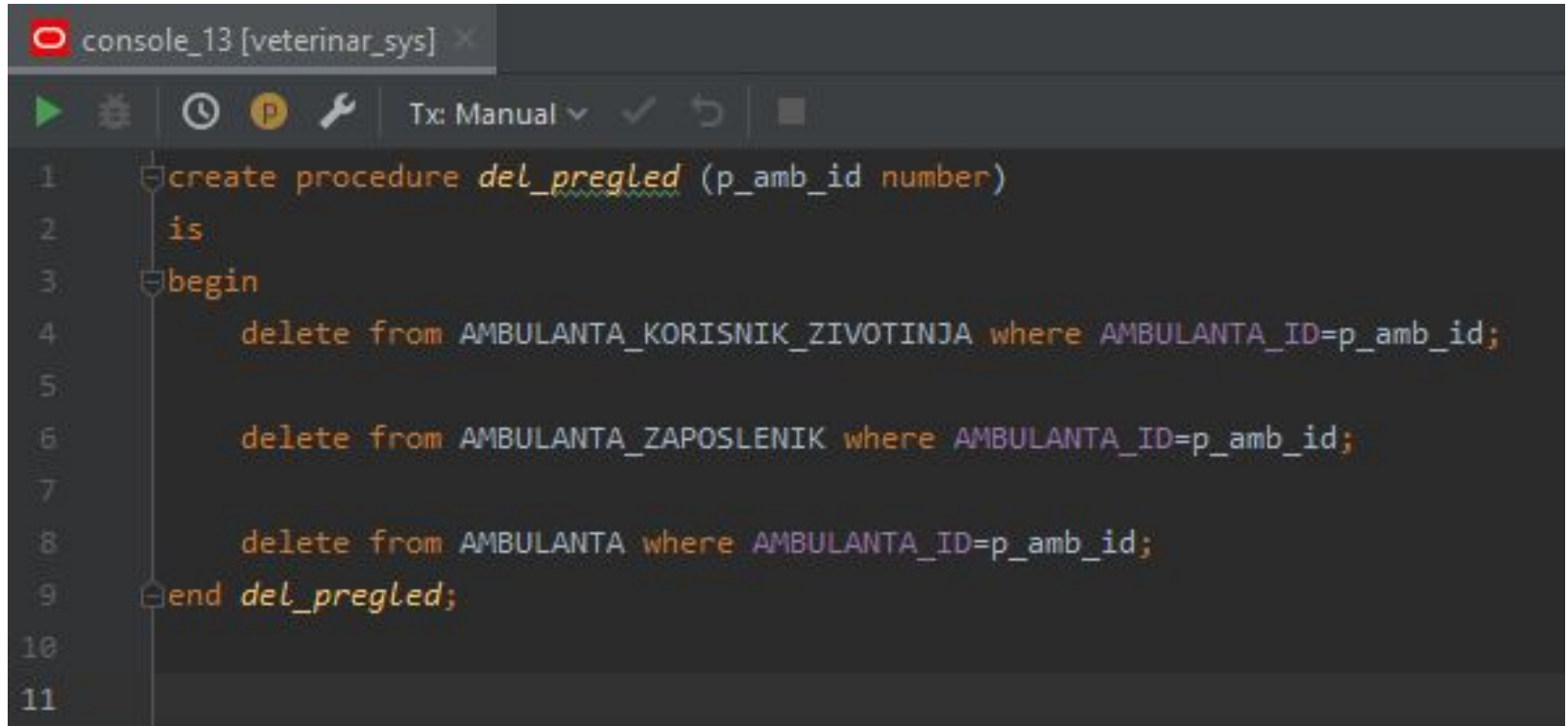
# 1. Prikaz zaposlenika (View)



The screenshot displays a database console window titled "console\_13 [veterinar\_sys]". The SQL editor contains two lines of code: `CREATE VIEW zap_view AS select ZAPOSLENIK_ID, ime, prezime, sifra from zaposlenik;` on line 1, and `select * from zap_view;` on line 3. The second query is highlighted with a green box. Below the editor, the "Output" tab is active, showing the results of the second query. The output is a table with 6 rows and 4 columns: `ZAPOSLENIK_ID`, `IME`, `PREZIME`, and `SIFRA`. The data rows are numbered 1 through 6 on the left.

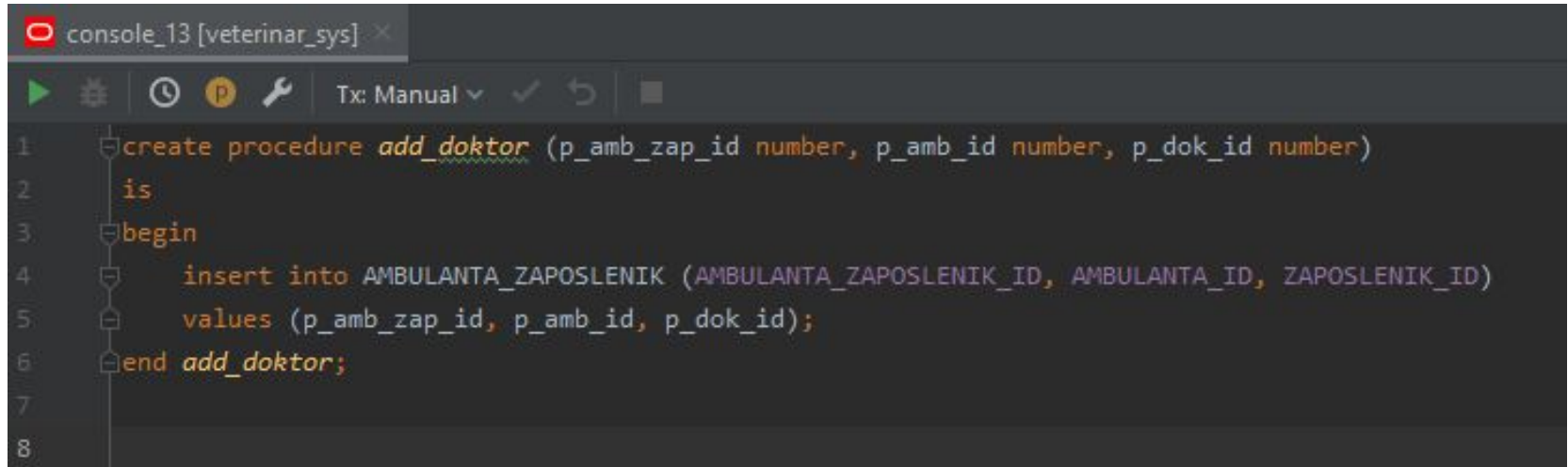
	ZAPOSLENIK_ID	IME	PREZIME	SIFRA
1	1	Mirko1	Mirkec1	00100001
2	2	Mirko2	Mirkec2	00100002
3	3	Mirko3	Mirkec3	00100003
4	4	Mirko4	Mirkec4	00100005
5	5	Mirko5	Mirkec5	00100007
6	6	Mirko6	Mirkec6	00100302

## 2. Izbriši pregled sa ID pregled (procedura)



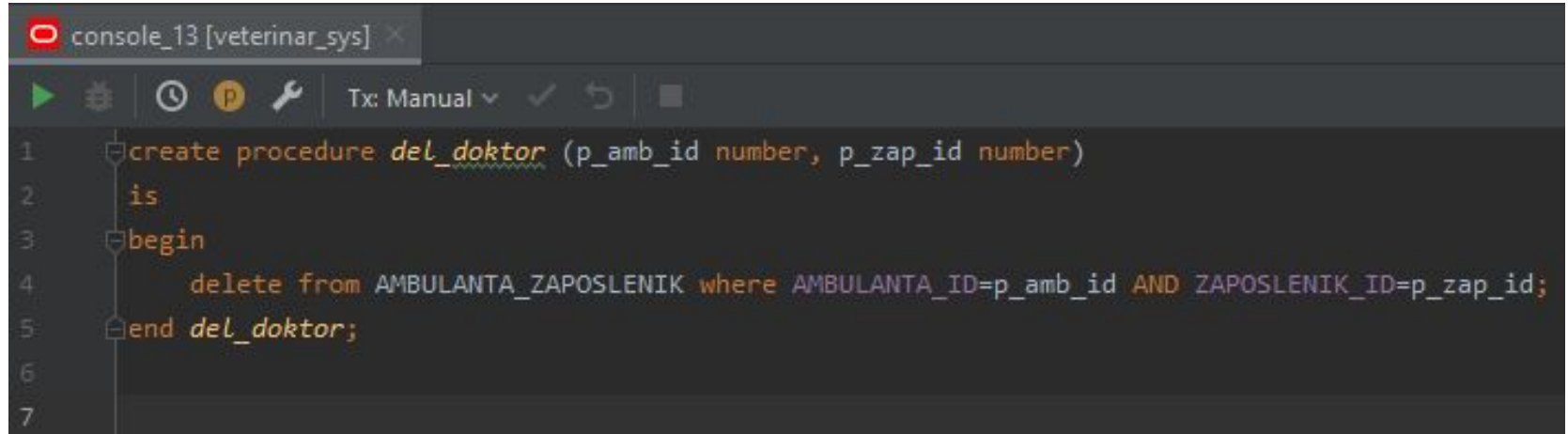
```
console_13 [veterinar_sys] x
▶ ⚙ ⌚ Ⓟ ⚙ Tx: Manual ✓ ↶ ■
1  create procedure del_pregled (p_amb_id number)
2      is
3      begin
4          delete from AMBULANTA_KORISNIK_ZIVOTINJA where AMBULANTA_ID=p_amb_id;
5
6          delete from AMBULANTA_ZAPOSLENIK where AMBULANTA_ID=p_amb_id;
7
8          delete from AMBULANTA where AMBULANTA_ID=p_amb_id;
9      end del_pregled;
10
11
```

### 3. Dodaj doktora na pregled (procedura)



```
console_13 [veterinar_sys] x
[Run] [Debug] [Clock] [P] [Wrench] Tx: Manual [Check] [Refresh] [Close]
1 create procedure add_doktor (p_amb_zap_id number, p_amb_id number, p_dok_id number)
2 is
3 begin
4     insert into AMBULANTA_ZAPOSLENIK (AMBULANTA_ZAPOSLENIK_ID, AMBULANTA_ID, ZAPOSLENIK_ID)
5     values (p_amb_zap_id, p_amb_id, p_dok_id);
6 end add_doktor;
7
8
```

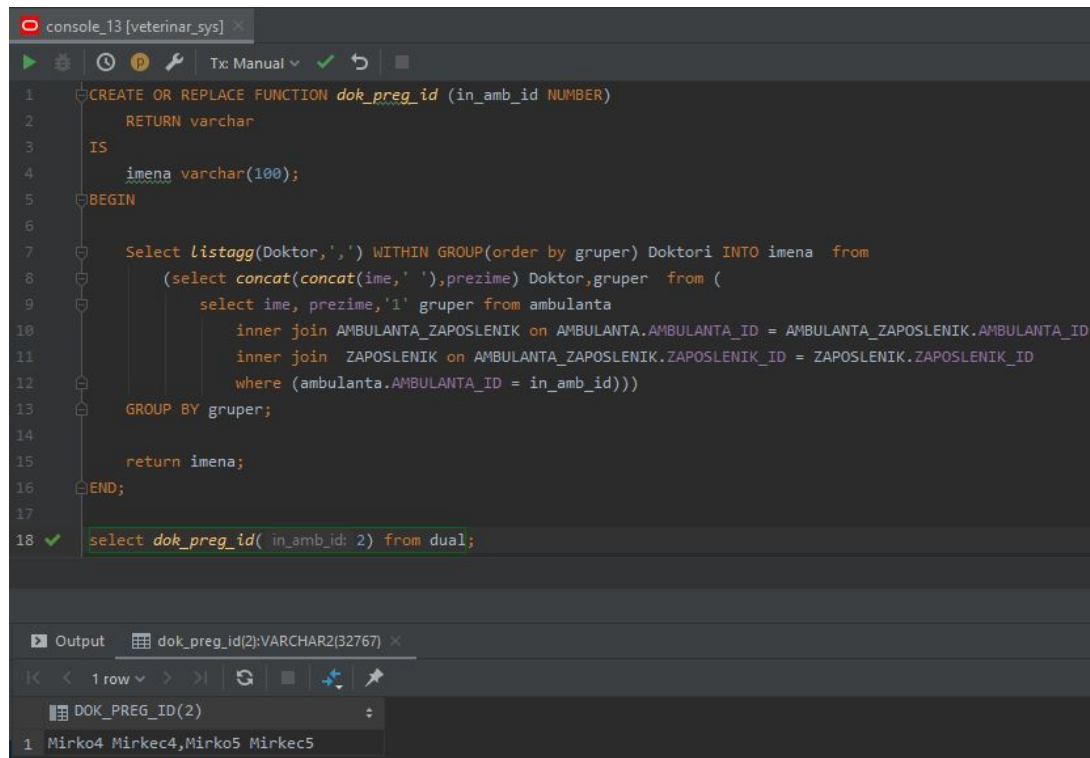
## 4. Izbriši doktora sa pregleda (procedura)



The screenshot shows a SQL console window titled "console\_13 [veterinar\_sys]". The window has a toolbar with icons for running, debugging, and other SQL-related actions. The SQL code is as follows:

```
1 create procedure del_doktor (p_amb_id number, p_zap_id number)
2 is
3 begin
4     delete from AMBULANTA_ZAPOSLENIK where AMBULANTA_ID=p_amb_id AND ZAPOSLENIK_ID=p_zap_id;
5 end del_doktor;
6
7
```

## 5. Prikaži doktora za pregled ID (funkcija)



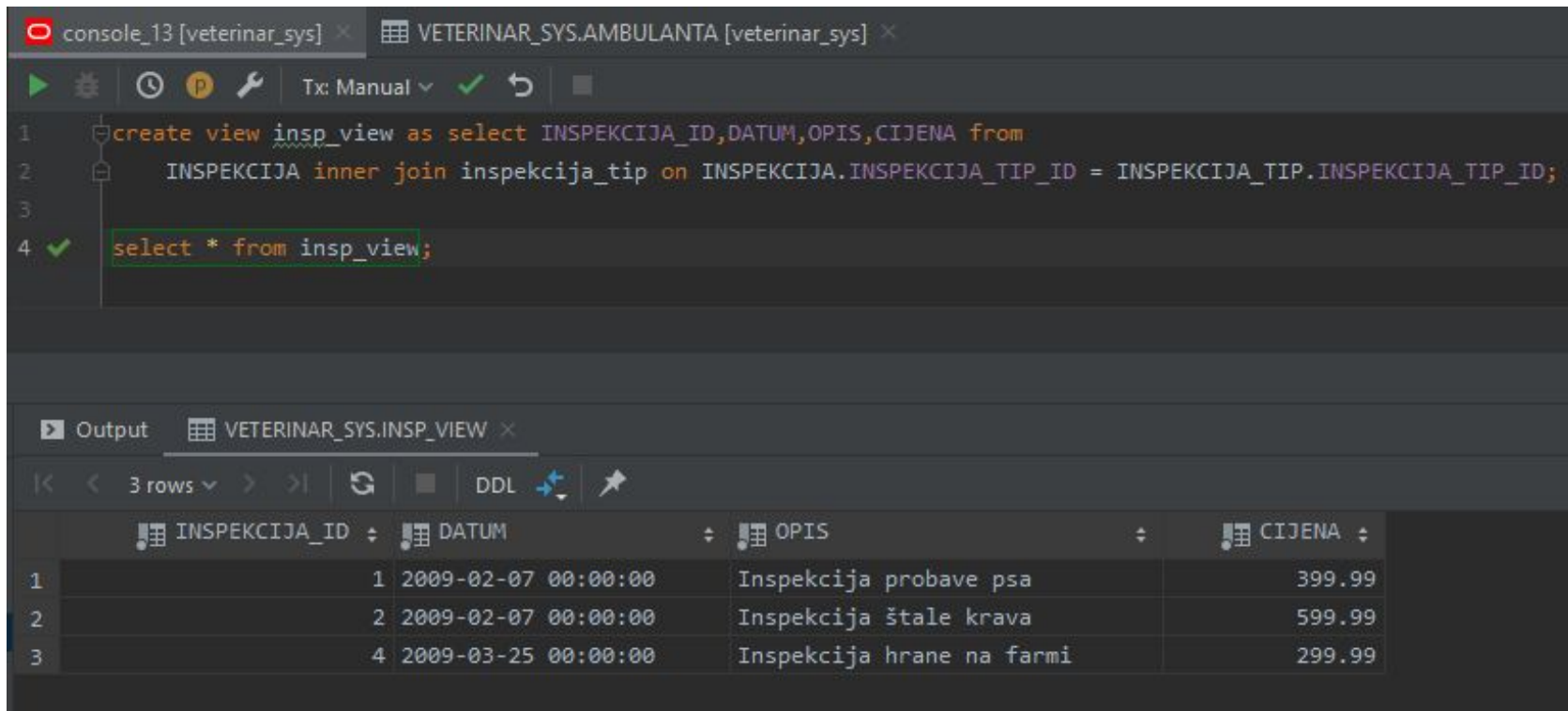
```
console_13 [veterinar_sys] x
Tx: Manual ✓ ↺
1 CREATE OR REPLACE FUNCTION dok_preg_id (in_amb_id NUMBER)
2   RETURN varchar
3   IS
4     imena varchar(100);
5   BEGIN
6
7     select listagg(Doktor,',') WITHIN GROUP(order by gruper) Doktori INTO imena from
8       (select concat(concat(ime,' '),prezime) Doktor,gruper from (
9         select ime, prezime,'1' gruper from ambulanta
10        inner join AMBULANTA_ZAPOSLENIK on AMBULANTA.AMBULANTA_ID = AMBULANTA_ZAPOSLENIK.AMBULANTA_ID
11        inner join ZAPOSLENIK on AMBULANTA_ZAPOSLENIK.ZAPOSLENIK_ID = ZAPOSLENIK.ZAPOSLENIK_ID
12        where (ambulanta.AMBULANTA_ID = in_amb_id)))
13    GROUP BY gruper;
14
15    return imena;
16  END;
17
18 ✓ select dok_preg_id( in_amb_id: 2) from dual;
```

Output dok\_preg\_id(2):VARCHAR2(32767) x

1 row

DOK_PREG_ID(2)
1 Mirk04 Mirkec4,Mirk05 Mirkec5

## 6. Prikaži inspekcije (view)



```
1 create view insp_view as select INSPEKCIJA_ID,DATUM,OPIS,CIJENA from
2   INSPEKCIJA inner join inspekcija_tip on INSPEKCIJA.INSPEKCIJA_TIP_ID = INSPEKCIJA_TIP.INSPEKCIJA_TIP_ID;
3
4 select * from insp_view;
```

Output VETERINAR\_SYS.INSP\_VIEW

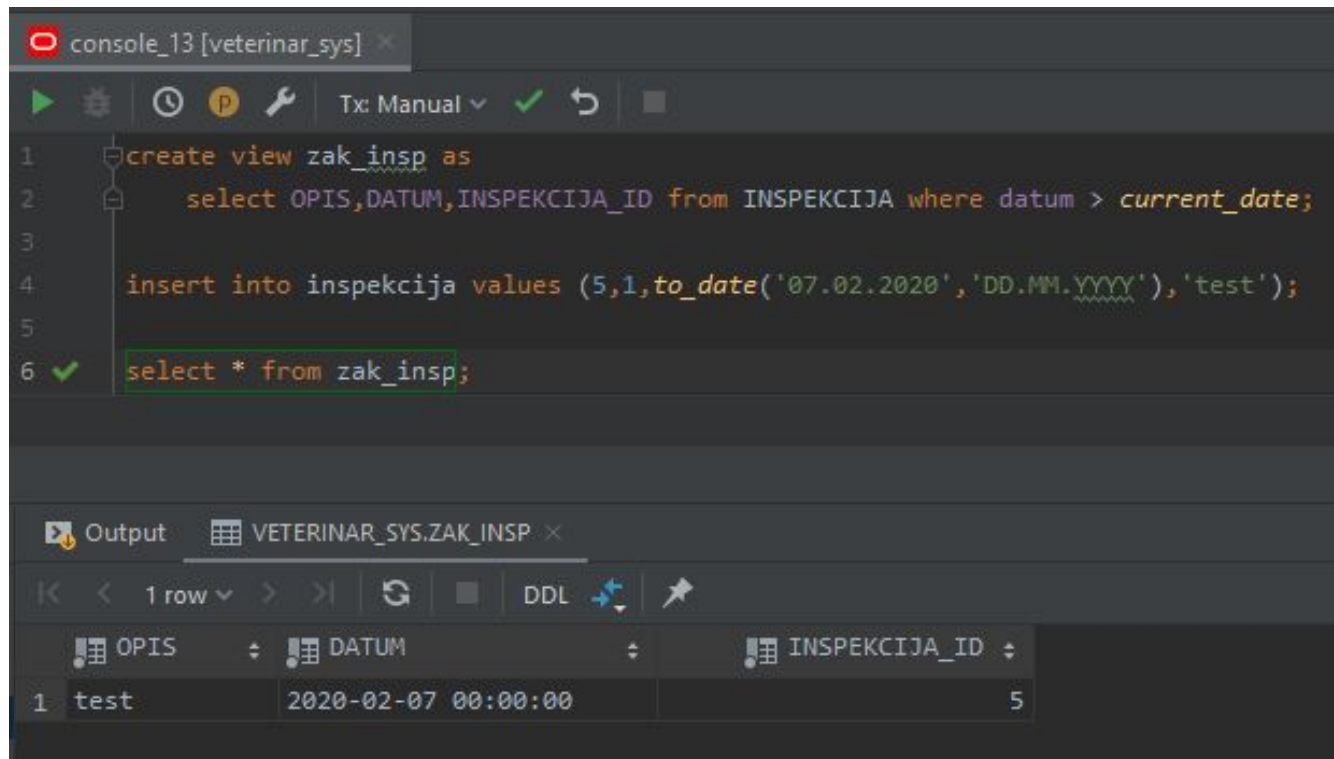
	INSPEKCIJA_ID	DATUM	OPIS	CIJENA
1	1	2009-02-07 00:00:00	Inspekcija probave psa	399.99
2	2	2009-02-07 00:00:00	Inspekcija štale krava	599.99
3	4	2009-03-25 00:00:00	Inspekcija hrane na farmi	299.99



## 7. Dodaj inspekciju i doktora (procedura)

```
console_13 [veterinar_sys] x
Tx: Manual ✓ ↺
1 create procedure add_insp_i (p_insp_id number, p_insp_tip number, p_insp_datum date, p_opis clob)
2 is
3 begin
4     insert into INSPEKCIJA (inspekcija_id, inspekcija_tip_id, datum, opis)
5     values (p_insp_id,p_insp_tip,p_insp_datum,p_opis);
6 end add_insp_i;
7
8 create procedure add_insp_d (p_dok_id number, p_insp_id number,)
9 is
10 begin
11     insert into INSPEKCIJA_ZAPOSLENIK (inspekcija_zaposlenik_id, inspekcija_id, zaposlenik_id)
12     values (p_dok_id,p_insp_id,p_dok_id);
13 end add_insp_d;
14
15
```

## 8. Prikaži zakazane inspekcije (view)



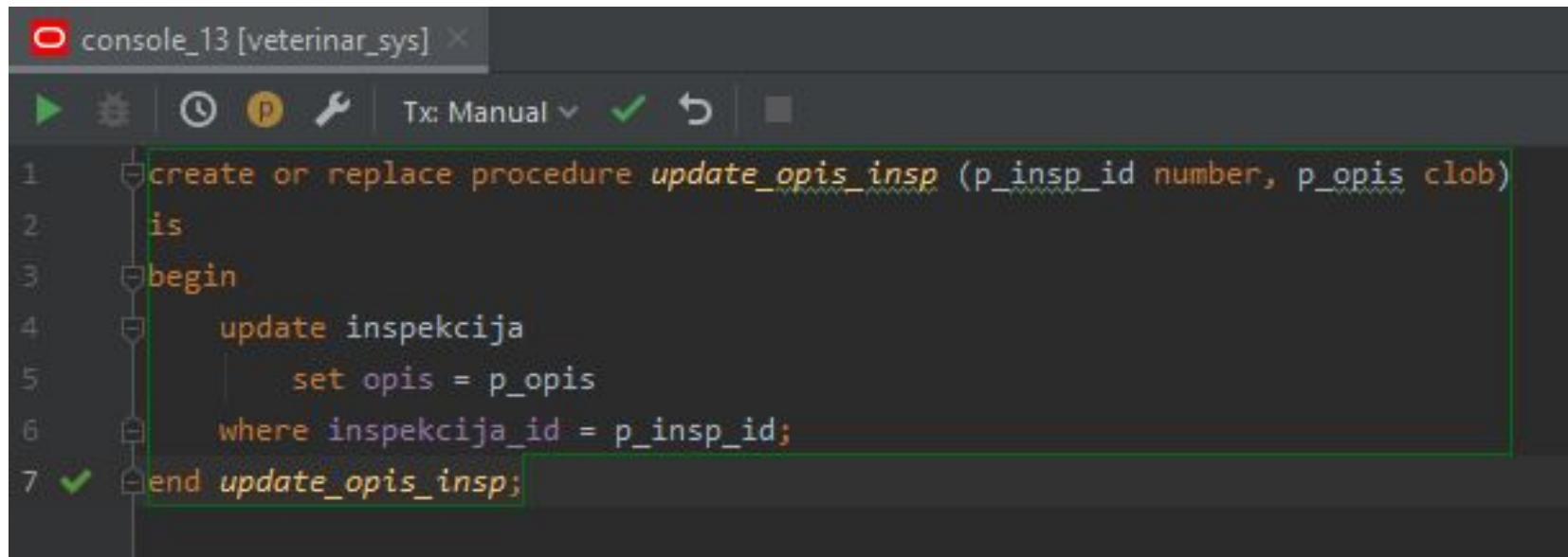
The screenshot shows a database console window titled 'console\_13 [veterinar\_sys]'. The SQL code is as follows:

```
1 create view zak_insp as
2     select OPIS,DATUM,INSPEKCIJA_ID from INSPEKCIJA where datum > current_date;
3
4     insert into inspekcija values (5,1,to_date('07.02.2020','DD.MM.YYYY'),'test');
5
6 select * from zak_insp;
```

The output section shows the result of the query 'VETERINAR\_SYS.ZAK\_INSP'. It displays 1 row with the following data:

	OPIS	DATUM	INSPEKCIJA_ID
1	test	2020-02-07 00:00:00	5

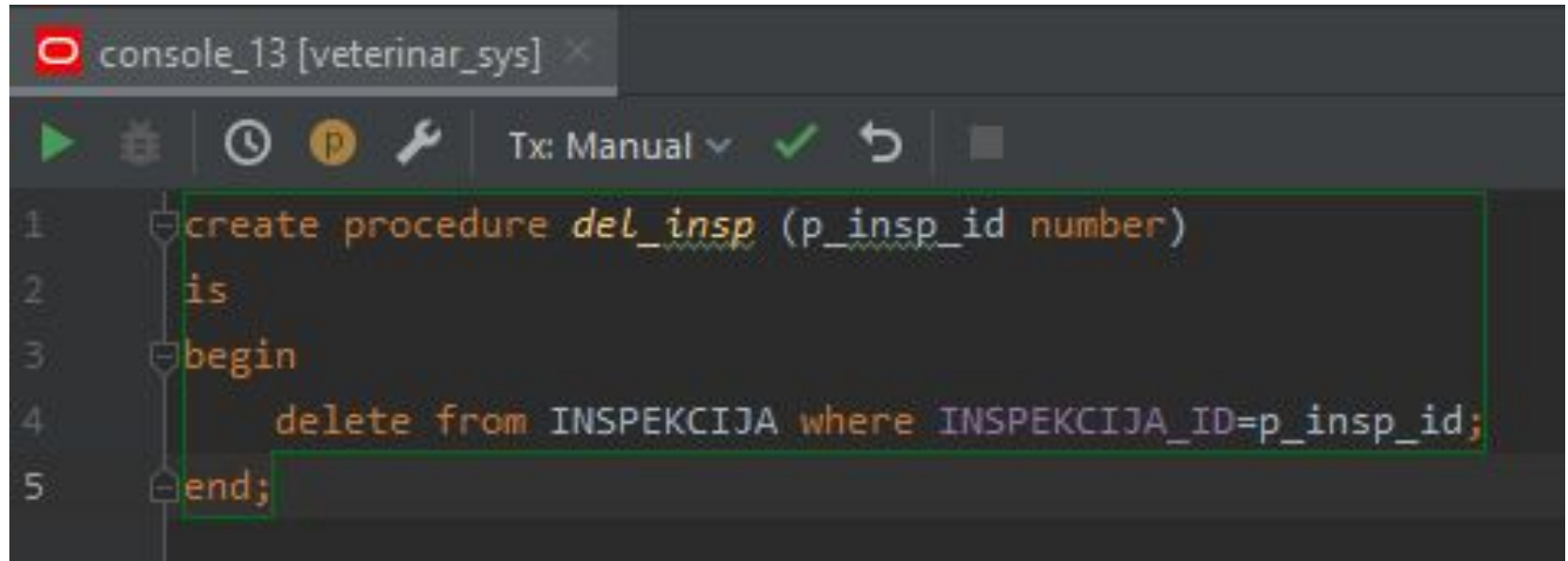
## 9. Promijeni opis inspekcije (procedura)



The screenshot shows a SQL IDE window titled 'console\_13 [veterinar\_sys]'. The code editor contains a PL/SQL procedure named 'update\_opis\_insp'. The procedure has two parameters: 'p\_insp\_id' of type 'number' and 'p\_opis' of type 'clob'. The procedure body starts with 'begin', followed by an 'update' statement on the 'inspekcija' table. The 'set' clause updates the 'opis' column to the value of 'p\_opis'. The 'where' clause filters for rows where 'inspekcija\_id' equals 'p\_insp\_id'. The procedure ends with 'end update\_opis\_insp;'. A green box highlights the entire procedure code. The line numbers 1 through 7 are visible on the left. The status bar at the bottom shows 'Tx: Manual' with a green checkmark and a refresh icon.

```
1 create or replace procedure update_opis_insp (p_insp_id number, p_opis clob)
2 is
3 begin
4     update inspekcija
5         set opis = p_opis
6     where inspekcija_id = p_insp_id;
7 end update_opis_insp;
```

## 10. Izbriši inspekciju (procedura)



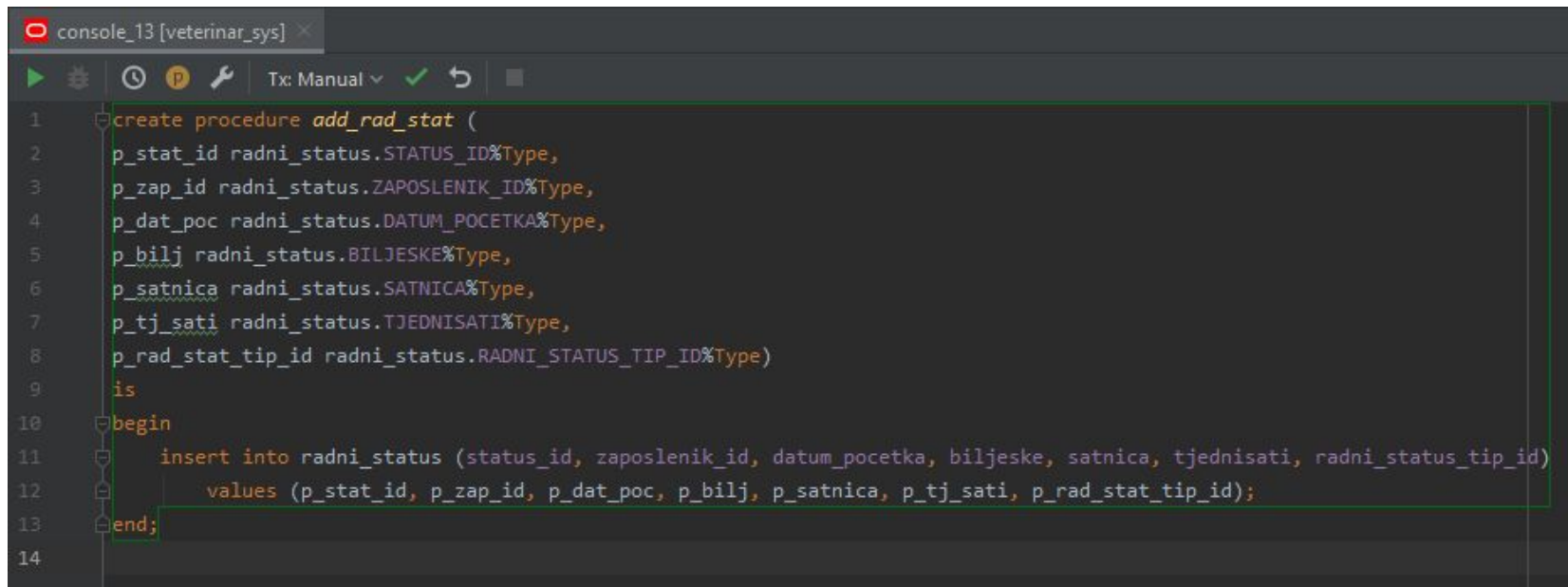
The screenshot shows a SQL IDE window titled 'console\_13 [veterinar\_sys]'. The interface includes a toolbar with icons for execution, debugging, and other functions. The main area displays a PL/SQL procedure named 'del\_insp' with the following code:

```
1 create procedure del_insp (p_insp_id number)
2 is
3 begin
4     delete from INSPEKCIJA where INSPEKCIJA_ID=p_insp_id;
5 end;
```

# Računovođa - Upiti

# 1. Unos novog zaposlenika

## 1.1. Unos novog radnog statusa (procedura)

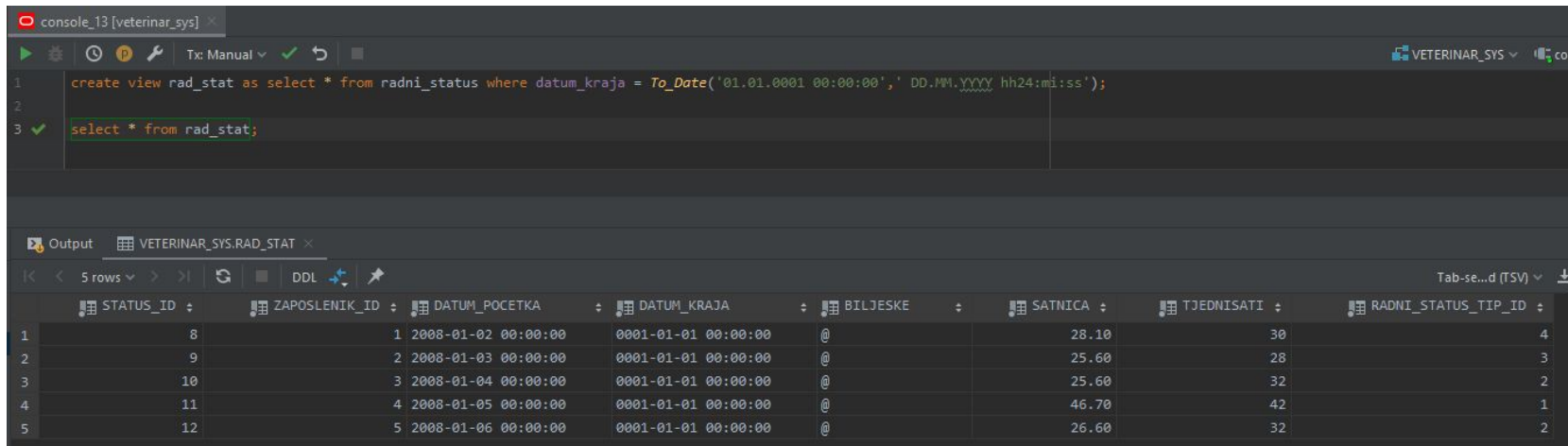


The screenshot shows a SQL IDE window titled "console\_13 [veterinar\_sys]". The code is a PL/SQL procedure named "add\_rad\_stat" that takes seven parameters: p\_stat\_id, p\_zap\_id, p\_dat\_poc, p\_bilj, p\_satnica, p\_tj\_sati, and p\_rad\_stat\_tip\_id. The procedure inserts a new record into the "radni\_status" table. The code is as follows:

```
1 create procedure add_rad_stat (  
2   p_stat_id radni_status.STATUS_ID%Type,  
3   p_zap_id radni_status.ZAPOSLENIK_ID%Type,  
4   p_dat_poc radni_status.DATUM_POCKETKA%Type,  
5   p_bilj radni_status.BILJESKE%Type,  
6   p_satnica radni_status.SATNICA%Type,  
7   p_tj_sati radni_status.TJEDNISATI%Type,  
8   p_rad_stat_tip_id radni_status.RADNI_STATUS_TIP_ID%Type)  
9   is  
10  begin  
11    insert into radni_status (status_id, zaposlenik_id, datum_pocetka, biljeske, satnica, tjednisati, radni_status_tip_id)  
12      values (p_stat_id, p_zap_id, p_dat_poc, p_bilj, p_satnica, p_tj_sati, p_rad_stat_tip_id);  
13  end;
```

# 1. Unos novog zaposlenika

## 1.2. Pogled za radne statute



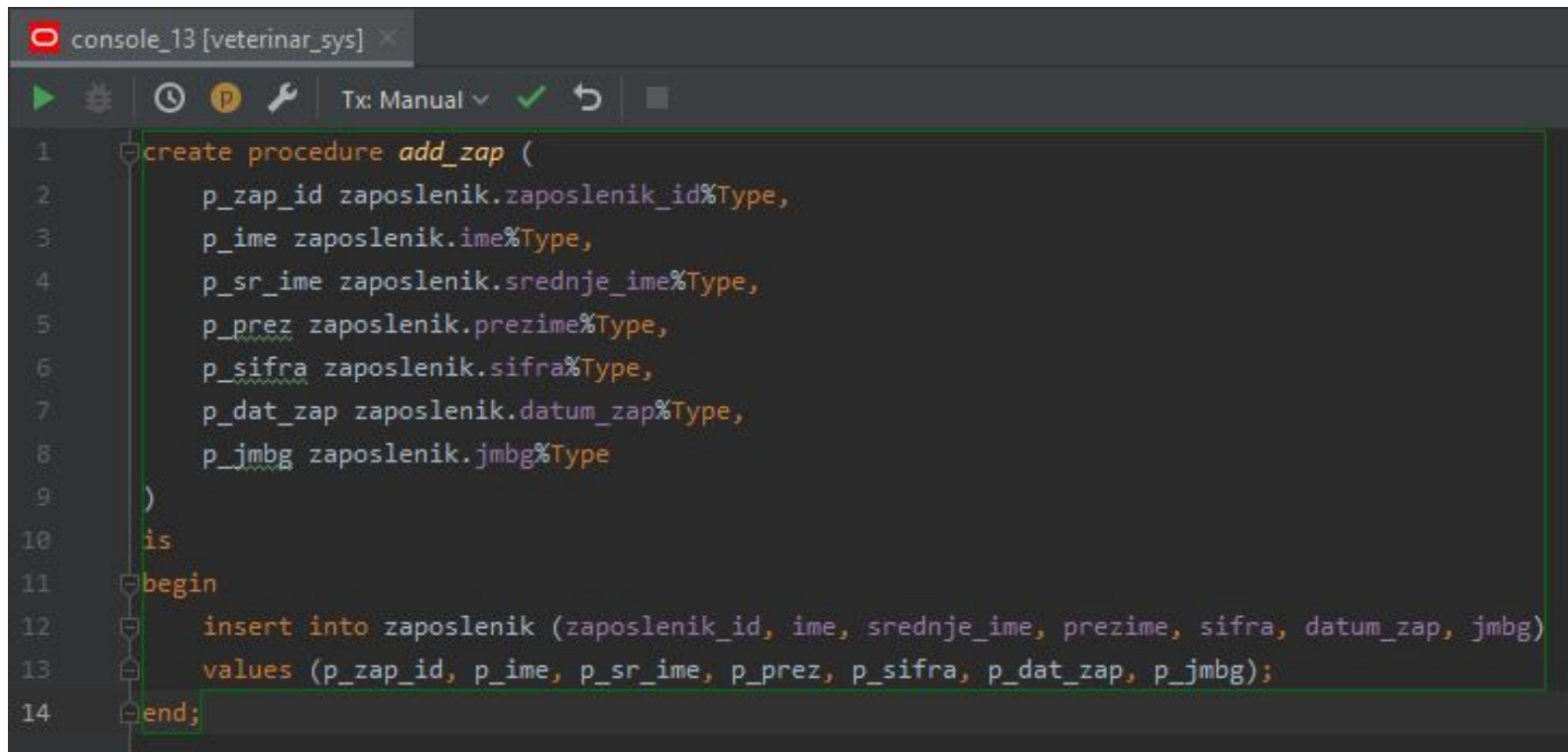
The screenshot shows a database console window titled 'console\_13 [veterinar\_sys]'. It contains two SQL statements: a 'create view' statement and a 'select' statement. Below the console, the 'Output' tab displays the results of the 'select' statement as a table with 8 columns and 5 rows.

```
1 create view rad_stat as select * from radni_status where datum_kraja = To_Date('01.01.0001 00:00:00',' DD.MM.YYYY hh24:mi:ss');
2
3 select * from rad_stat;
```

	STATUS_ID	ZAPOSLENIK_ID	DATUM_POCETKA	DATUM_KRAJA	BILJESKE	SATNICA	TJEDNISATI	RADNI_STATUS_TIP_ID
1	8	1	2008-01-02 00:00:00	0001-01-01 00:00:00	@	28.10	30	4
2	9	2	2008-01-03 00:00:00	0001-01-01 00:00:00	@	25.60	28	3
3	10	3	2008-01-04 00:00:00	0001-01-01 00:00:00	@	25.60	32	2
4	11	4	2008-01-05 00:00:00	0001-01-01 00:00:00	@	46.70	42	1
5	12	5	2008-01-06 00:00:00	0001-01-01 00:00:00	@	26.60	32	2

# 1. Unos novog zaposlenika

## 1.3. Unos zaposlenika (procedura)

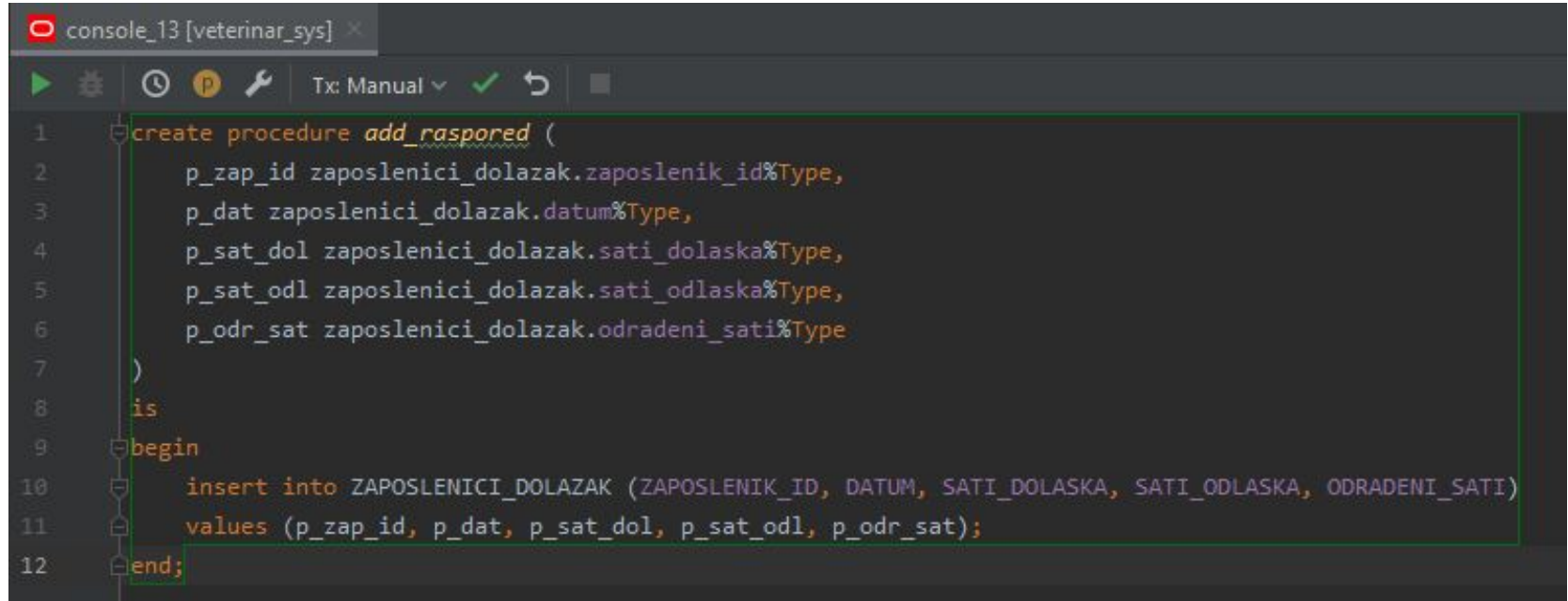


The screenshot shows a SQL console window titled "console\_13 [veterinar\_sys]". The window has a toolbar with icons for running, debugging, and other functions. The SQL code is as follows:

```
1 create procedure add_zap (  
2     p_zap_id zaposlenik.zaposlenik_id%Type,  
3     p_ime zaposlenik.ime%Type,  
4     p_sr_ime zaposlenik.srednje_ime%Type,  
5     p_prez zaposlenik.prezime%Type,  
6     p_sifra zaposlenik.sifra%Type,  
7     p_dat_zap zaposlenik.datum_zap%Type,  
8     p_jmbg zaposlenik.jmbg%Type  
9 )  
10 is  
11 begin  
12     insert into zaposlenik (zaposlenik_id, ime, srednje_ime, prezime, sifra, datum_zap, jmbg)  
13     values (p_zap_id, p_ime, p_sr_ime, p_prez, p_sifra, p_dat_zap, p_jmbg);  
14 end;
```

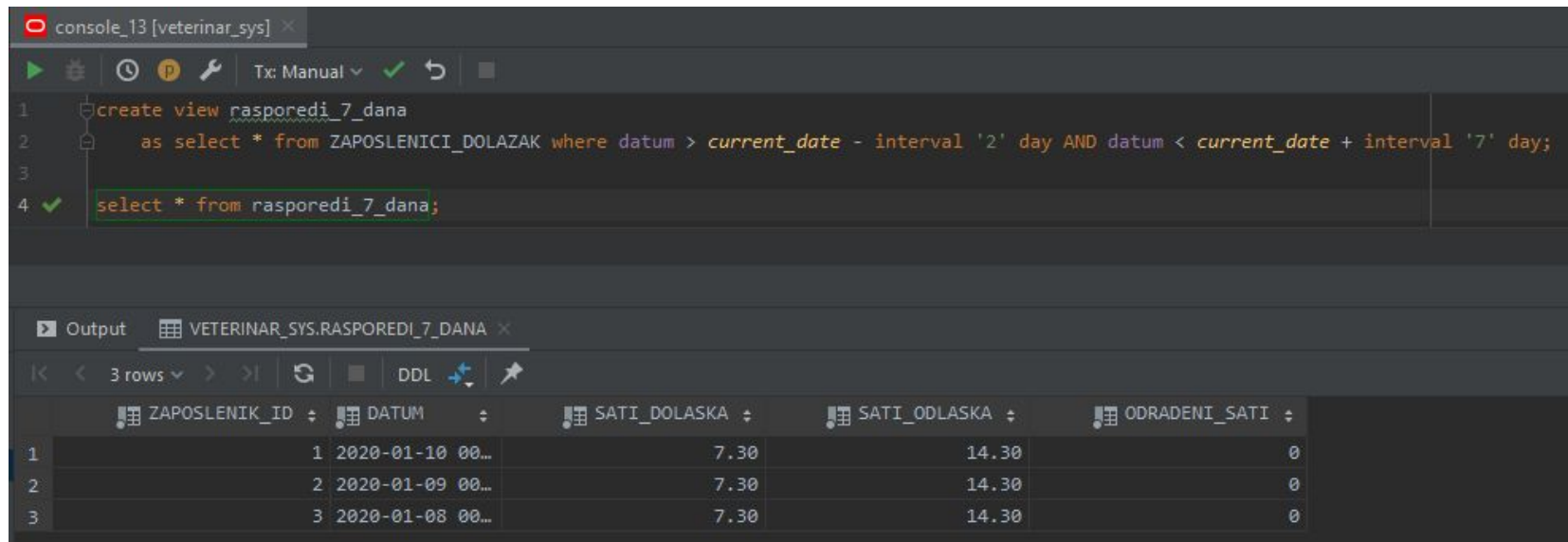


## 2. Unos rasporeda (procedura)



```
console_13 [veterinar_sys] x
Tx: Manual ✓ ↺
1 create procedure add_raspored (
2     p_zap_id zaposlenici_dolazak.zaposlenik_id%Type,
3     p_dat zaposlenici_dolazak.datum%Type,
4     p_sat_dol zaposlenici_dolazak.sati_dolaska%Type,
5     p_sat_odl zaposlenici_dolazak.sati_odlaska%Type,
6     p_odr_sat zaposlenici_dolazak.odradeni_sati%Type
7 )
8 is
9 begin
10     insert into ZAPOSLENICI_DOLAZAK (ZAPOSLENIK_ID, DATUM, SATI_DOLASKA, SATI_ODLASKA, ODRADENI_SATI)
11     values (p_zap_id, p_dat, p_sat_dol, p_sat_odl, p_odr_sat);
12 end;
```

### 3. Pregled rasporeda za sljedecih 7 dana (pogled)



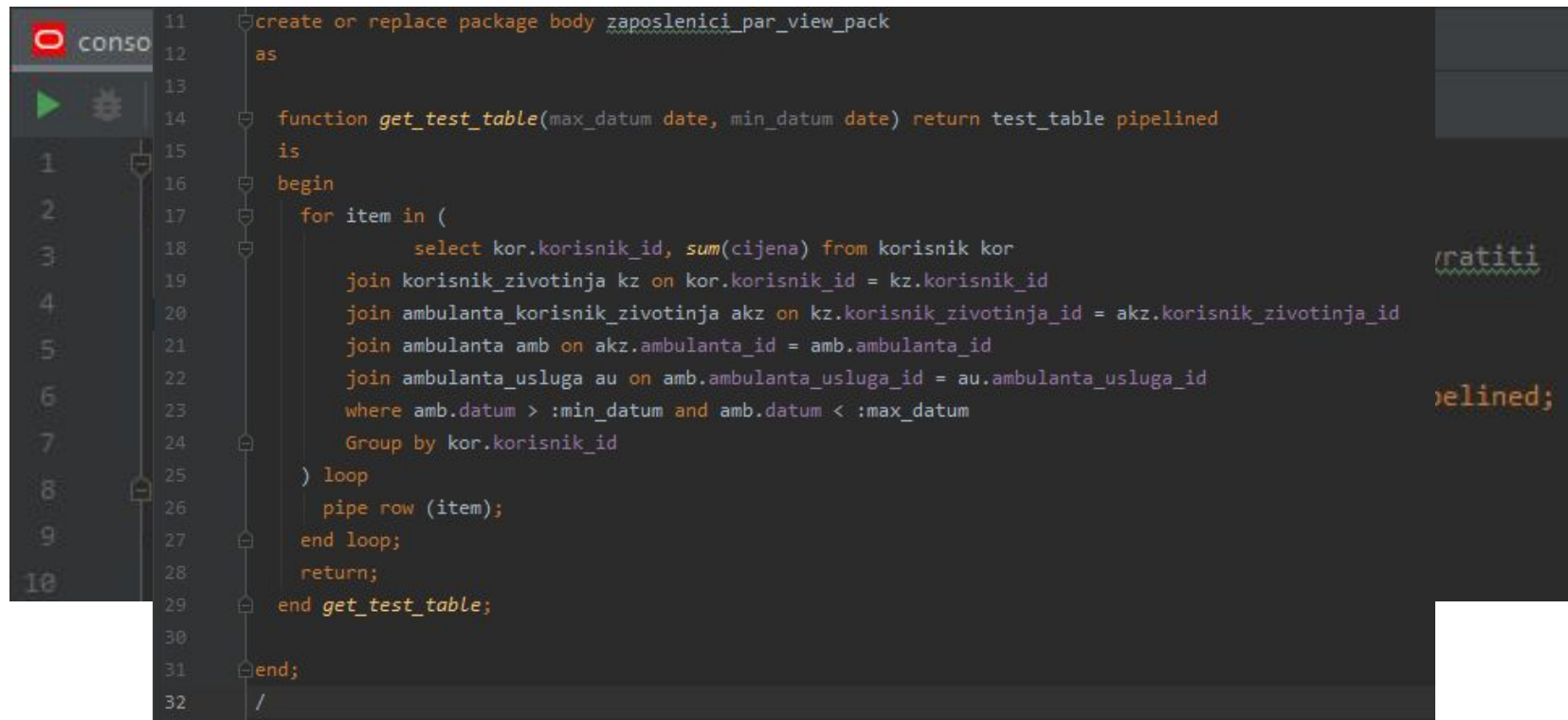
The screenshot shows a SQL console window titled 'console\_13 [veterinar\_sys]'. The SQL code is as follows:

```
1 create view rasporedi_7_dana
2   as select * from ZAPOSLENICI_DOLAZAK where datum > current_date - interval '2' day AND datum < current_date + interval '7' day;
3
4 select * from rasporedi_7_dana;
```

The output shows 3 rows of data from the view VETERINAR\_SYS.RASPOREDI\_7\_DANA:

	ZAPOSLENIK_ID	DATUM	SATI_DOLASKA	SATI_ODLASKA	ODRADENI_SATI
1	1	2020-01-10 00...	7.30	14.30	0
2	2	2020-01-09 00...	7.30	14.30	0
3	3	2020-01-08 00...	7.30	14.30	0

4. Za svakog korisnika vrati njegovu kupnu cijenu svih pregleda u ambulanti od jednog do drugog datuma (funkcija)



```
11 create or replace package body zaposlenici_par_view_pack
12 as
13
14 function get_test_table(max_datum date, min_datum date) return test_table pipelined
15 is
16 begin
17     for item in (
18         select kor.korisnik_id, sum(cijena) from korisnik kor
19         join korisnik_zivotinja kz on kor.korisnik_id = kz.korisnik_id
20         join ambulanta_korisnik_zivotinja akz on kz.korisnik_zivotinja_id = akz.korisnik_zivotinja_id
21         join ambulanta amb on akz.ambulanta_id = amb.ambulanta_id
22         join ambulanta_usluga au on amb.ambulanta_usluga_id = au.ambulanta_usluga_id
23         where amb.datum > :min_datum and amb.datum < :max_datum
24         Group by kor.korisnik_id
25     ) loop
26         pipe row (item);
27     end loop;
28     return;
29 end get_test_table;
30
31 end;
32 /
```

The screenshot shows a SQL IDE with a dark theme. On the left, there is a sidebar with a 'conso' icon and a list of line numbers from 1 to 10. The main editor area displays a PL/SQL function definition. The function is named 'get\_test\_table' and takes two date parameters: 'max\_datum' and 'min\_datum'. It returns a 'test\_table' and is marked as 'pipelined'. The function body starts with a 'begin' statement, followed by a 'for' loop. Inside the loop, a SQL query is executed. The query selects 'kor.korisnik\_id' and 'sum(cijena)' from the 'korisnik' table, joined with 'korisnik\_zivotinja', 'ambulanta\_korisnik\_zivotinja', 'ambulanta', and 'ambulanta\_usluga'. The query filters for records where 'amb.datum' is greater than ':min\_datum' and less than ':max\_datum', and groups the results by 'kor.korisnik\_id'. The loop body contains 'pipe row (item);'. After the loop, there is a 'return;' statement. The function ends with 'end get\_test\_table;'. The entire package body is enclosed in 'create or replace package body zaposlenici\_par\_view\_pack as' and 'end;' statements. The file ends with a forward slash '/'. There are some annotations on the right side of the code, such as 'ratiti' underlined and 'elined;'.

Hvala na pažnji :)