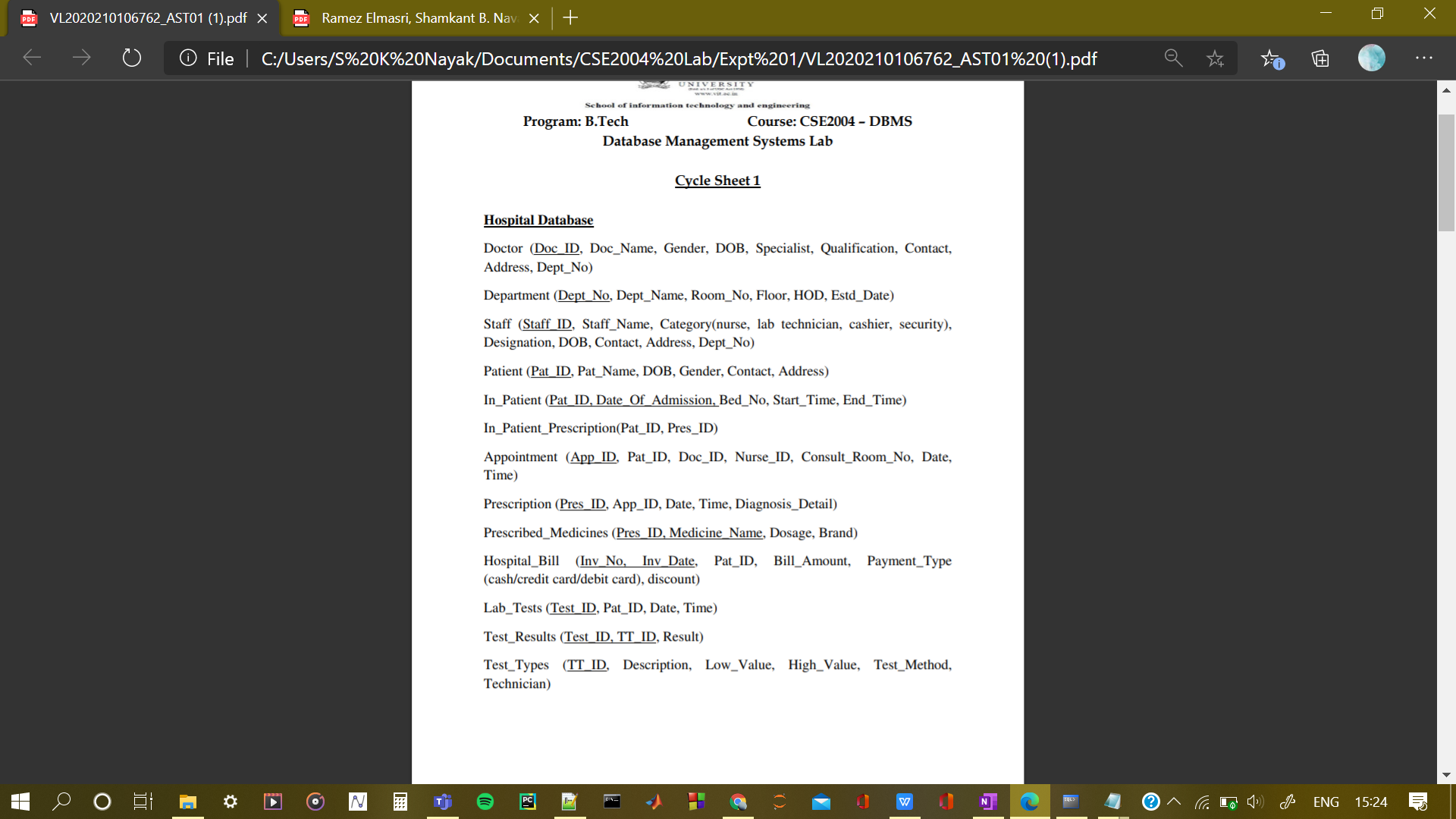
**Expt No.: 3 Name: Swaranjana Nayak**

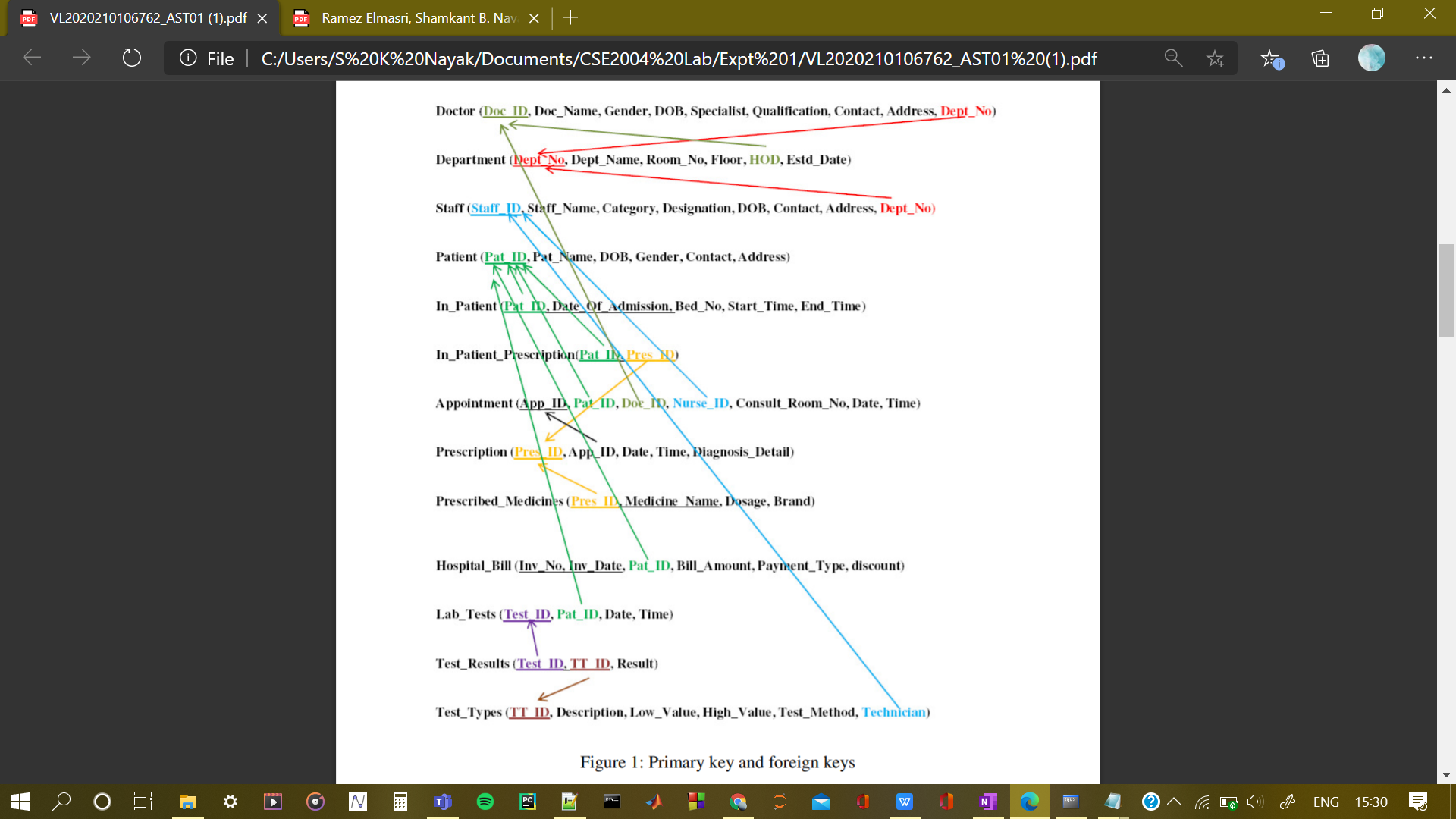
**Date: 17/10/2020 Reg. No.: 19BCE0977**

**CYCLE SHEET 3**

**Aim:**

To solve the given problems by implementing in SQL.





1. **Write a PL/SQL program to implement a simple calculator**

**Code:**

declare

num1 integer;

num2 integer;

resu integer;

op varchar2(5);

begin

num1 := &num1;

num2 := &num2;

op := '&op';

if op = '+' then

resu := num1 + num2;

else if op = '-' then

resu := num1 - num2;

else if op = '\*' then

resu := num1 \* num2;

else if op = '/' then

resu := num1 / num2;

end if;

end if;

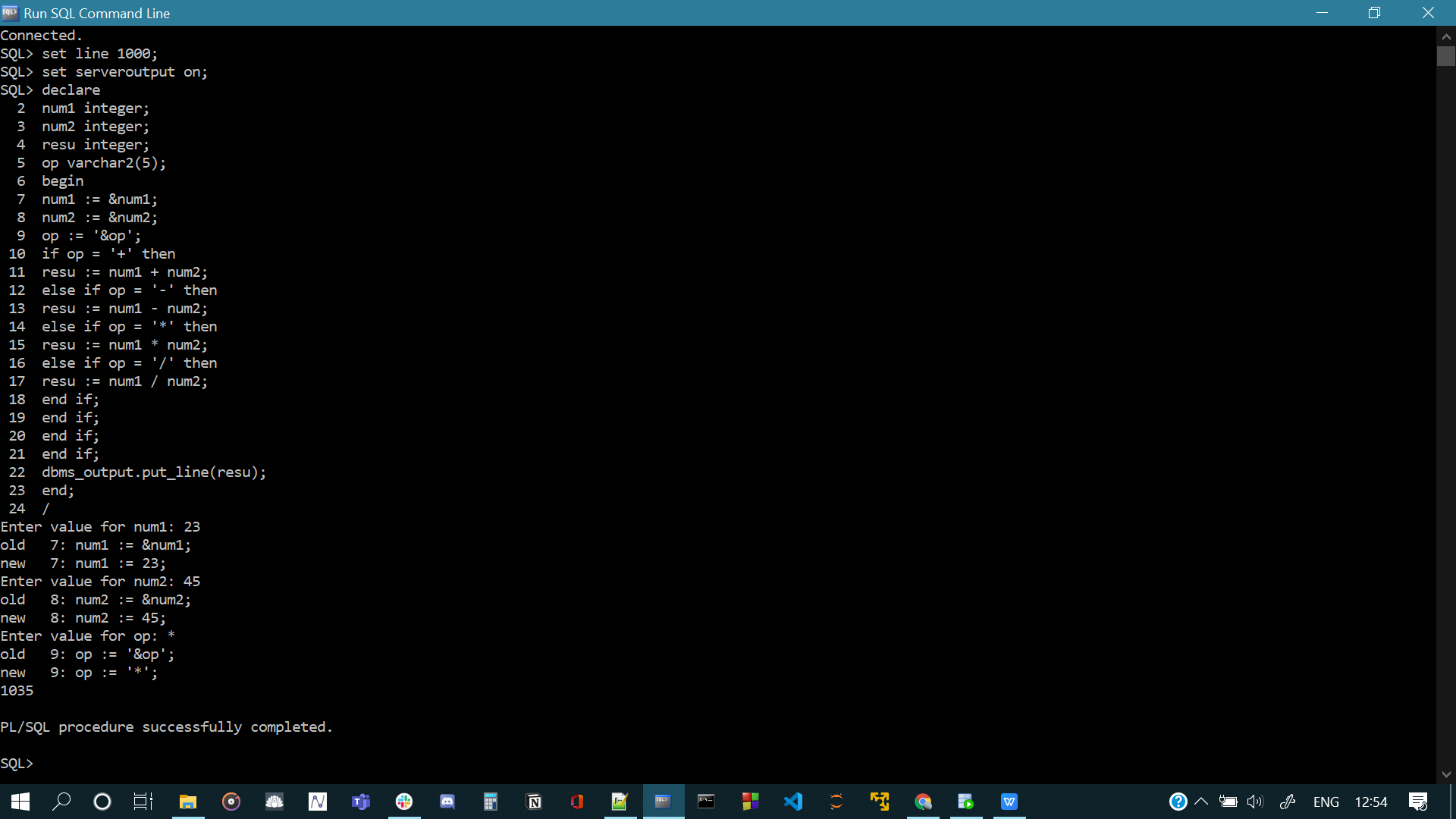
end if;

end if;

dbms\_output.put\_line(resu);

end;

**Output:**



1. **Write a PL/SQL program to practice reading the record from a table into local variables using different data types and %TYPE and display the same using locally declared variables.**

**Code:**

declare

did doctor.doc\_id%type;

dna doctor.doc\_name%type;

gen doctor.gender%type;

dob doctor.dob%type;

spe doctor.specialist%type;

con doctor.contact%type;

dep doctor.dept\_no%type;

begin

did := '&did';

select doc\_name into dna from doctor where doc\_id = did;

select gender into gen from doctor where doc\_id = did;

select dob into dob from doctor where doc\_id = did;

select specialist into spe from doctor where doc\_id = did;

select contact into con from doctor where doc\_id = did;

select dept\_no into dep from doctor where doc\_id = did;

dbms\_output.put\_line('Doctor ID:');

dbms\_output.put\_line(did);

dbms\_output.put\_line('Doctor Name: ');

dbms\_output.put\_line(dna);

dbms\_output.put\_line('Doctor Gender: ');

dbms\_output.put\_line(gen);

dbms\_output.put\_line('Specialization: ');

dbms\_output.put\_line(spe);

dbms\_output.put\_line('Contact: ');

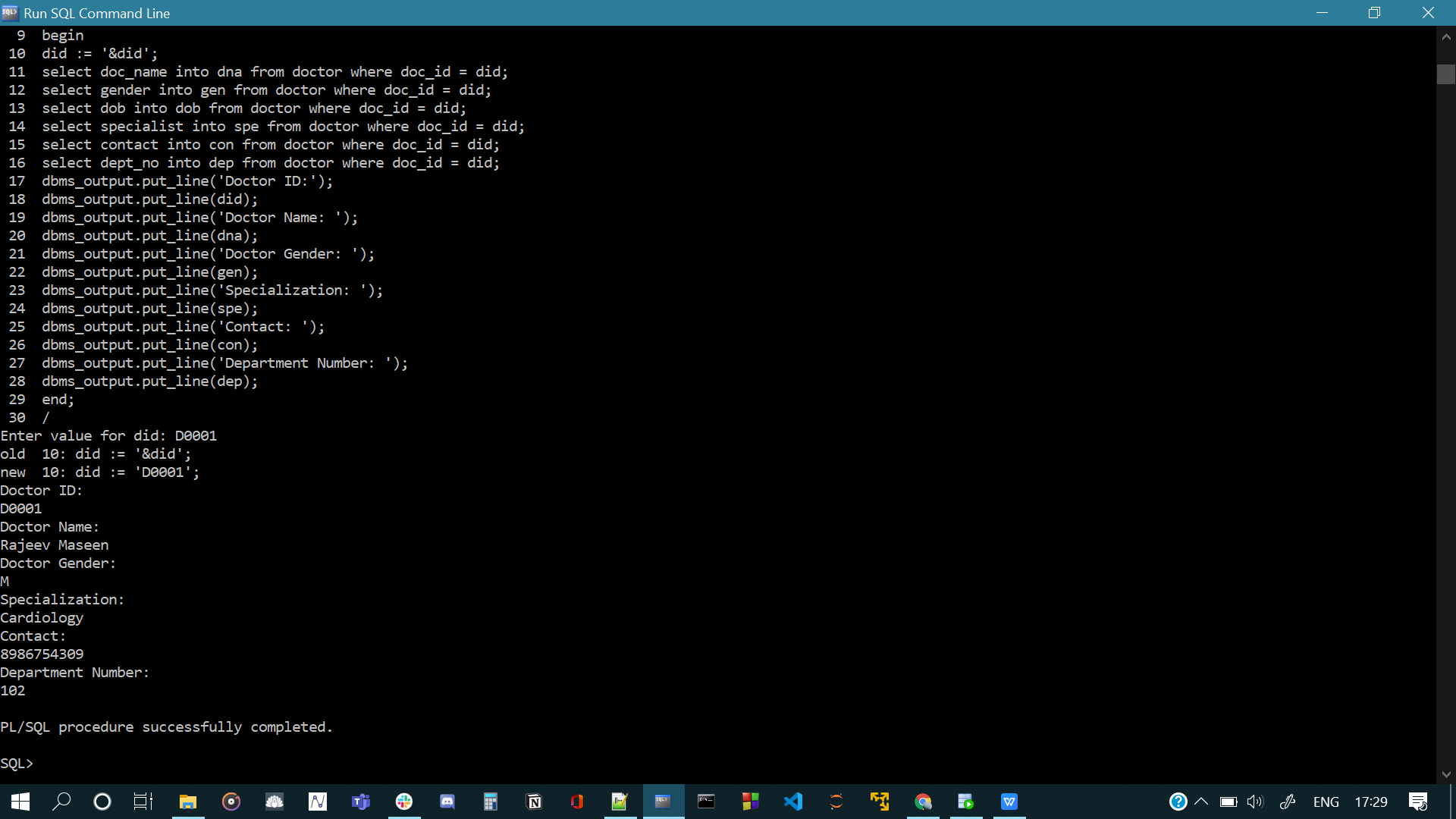
dbms\_output.put\_line(con);

dbms\_output.put\_line('Department Number: ');

dbms\_output.put\_line(dep);

end;

**Output:**



1. **Write a PL/SQL program to find the number of doctors in a given department with a given qualification (read values for department and qualification from user during runtime). If number is more than the number of doctors in that department with other qualifications then display ‘Well qualified’ else ‘Qualified’.**

**Code:**

declare

dep doctor.dept\_no%type;

qua doctor.qualification%TYPE;

dc number(3) := 0;

qc number(3) := 0;

begin

dep := &dep;

qua := '&qua';

select count(\*) into dc from doctor where dept\_no = dep;

select count(\*) into qc from doctor where dept\_no = dep and qualification = qua;

if qc > (dc - qc) then

dbms\_output.put\_line('Well Qualified');

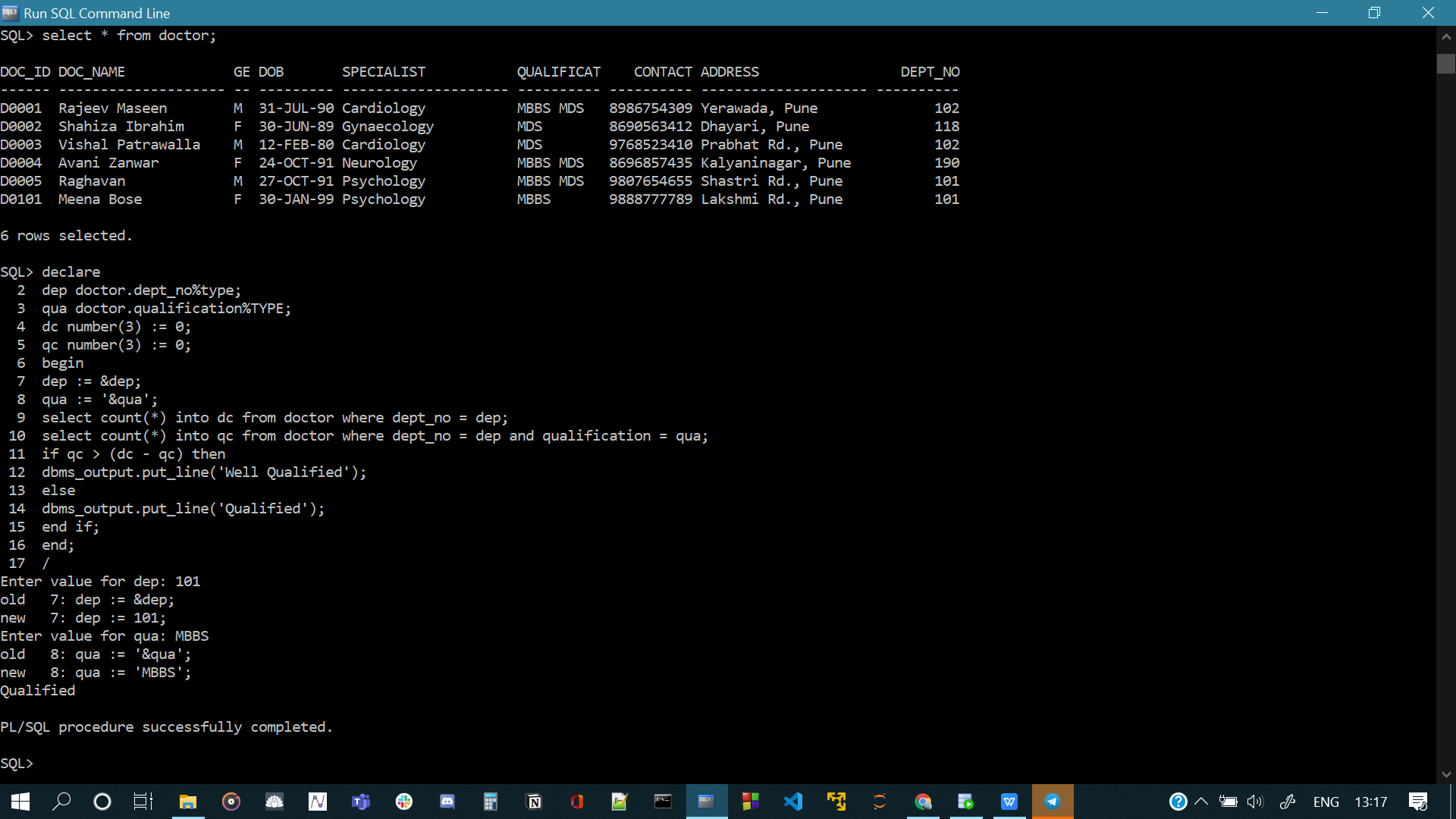
else

dbms\_output.put\_line('Qualified');

end if;

end;

**Output:**



1. **Write a PL/SQL program to insert records into any of the tables in your database.**

**Code:**

SQL> declare

2 mn medicines.med\_name%type;

3 br medicines.brand%type;

4 md medicines.manu\_date%type;

5 ed medicines.exp\_date%type;

6 begin

7 mn := '&mn';

8 br := '&br';

9 md := '&md';

10 ed := '&ed';

11 insert into medicines values(mn, br, md, ed);

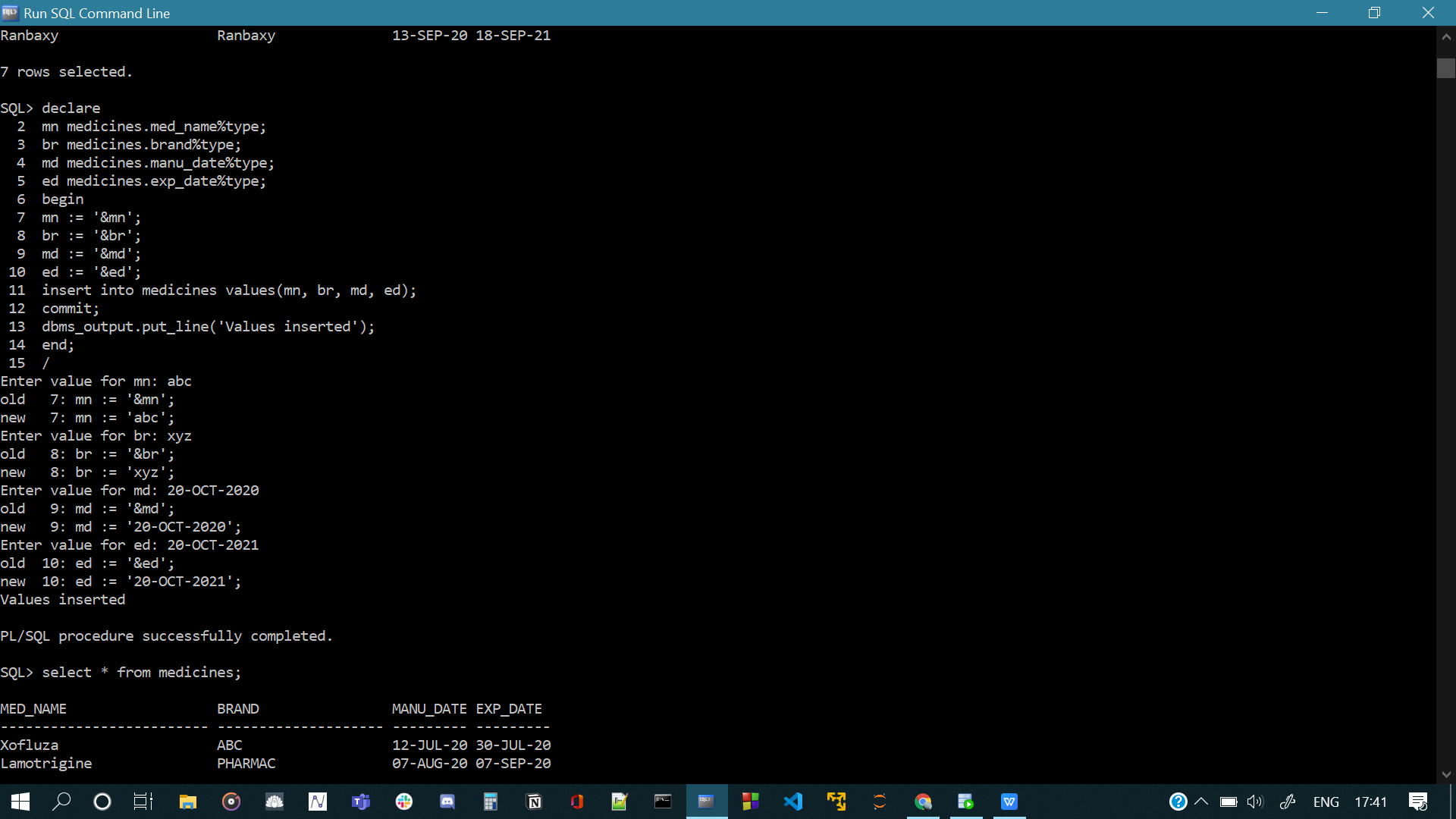
12 commit;

13 dbms\_output.put\_line('Values inserted');

14 end;

15 /

**Output:**



1. **Create a function to find the factorial of a given number.**

**Code:**

SQL> create or replace function factorial\_mine(num number)

2 return number is

3 i number;

4 f number;

5 begin

6 if (num = 0 or num = 1) then

7 return(1);

8 else

9 f := 1;

10 for i in 1..num

11 loop

12 f := f\*i;

13 end loop;

14 return f;

15 end if;

16 end;

17 /

SQL> declare

2 num number;

3 begin

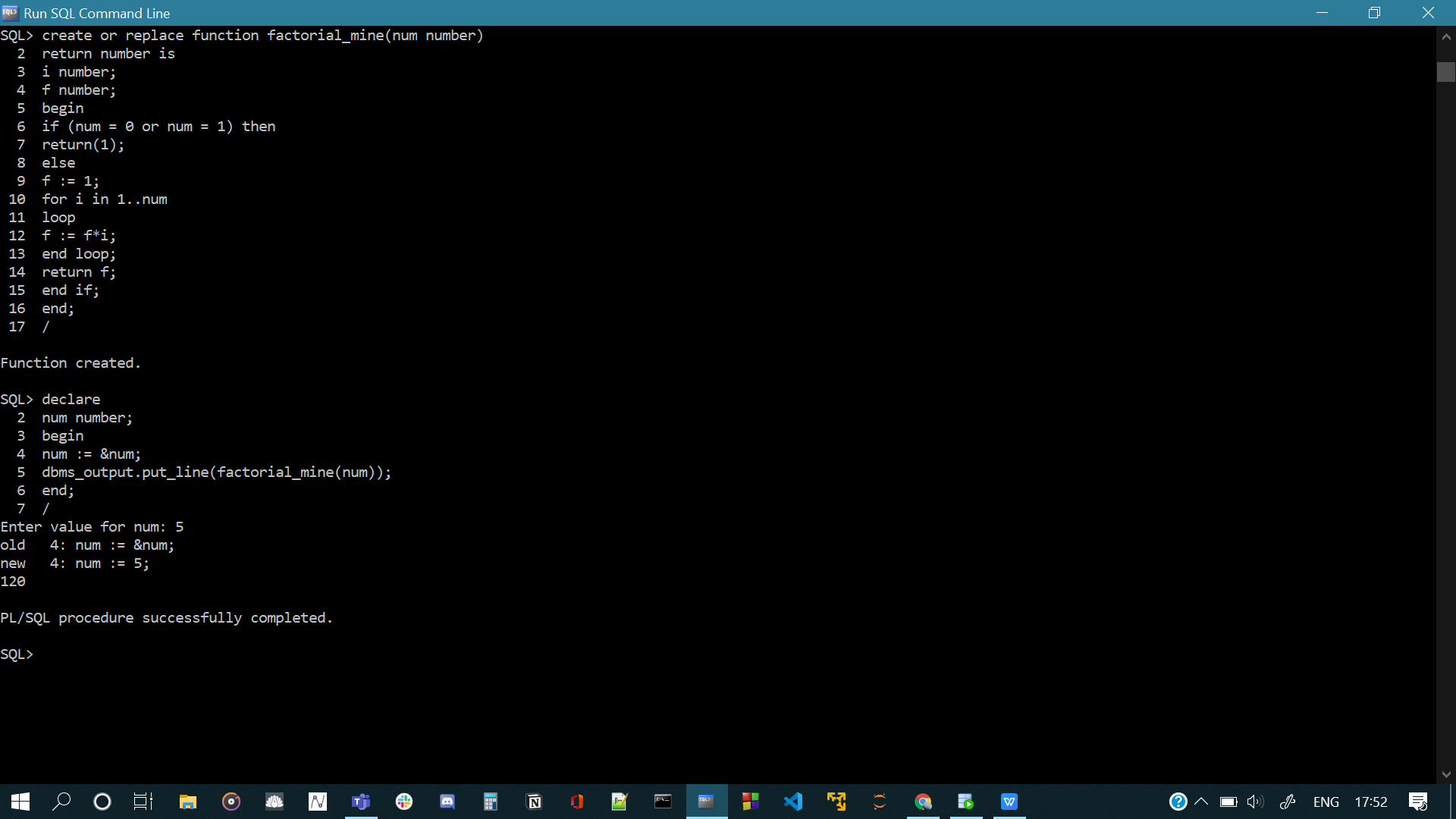
4 num := &num;

5 dbms\_output.put\_line(factorial\_mine(num));

6 end;

7 /

**Output:**



1. **Create a function DOC\_COUNT to find the number of doctors in the given department. Use the department name as the input parameter for the function.**

**Code:**

create or replace function doc\_count(dep doctor.dept\_no%type)

return number is

cou number;

begin

select count(\*) into cou from doctor where dept\_no = dep;

return cou;

end;

declare

dep doctor.dept\_no%type;

dc number(3) := 0;

begin

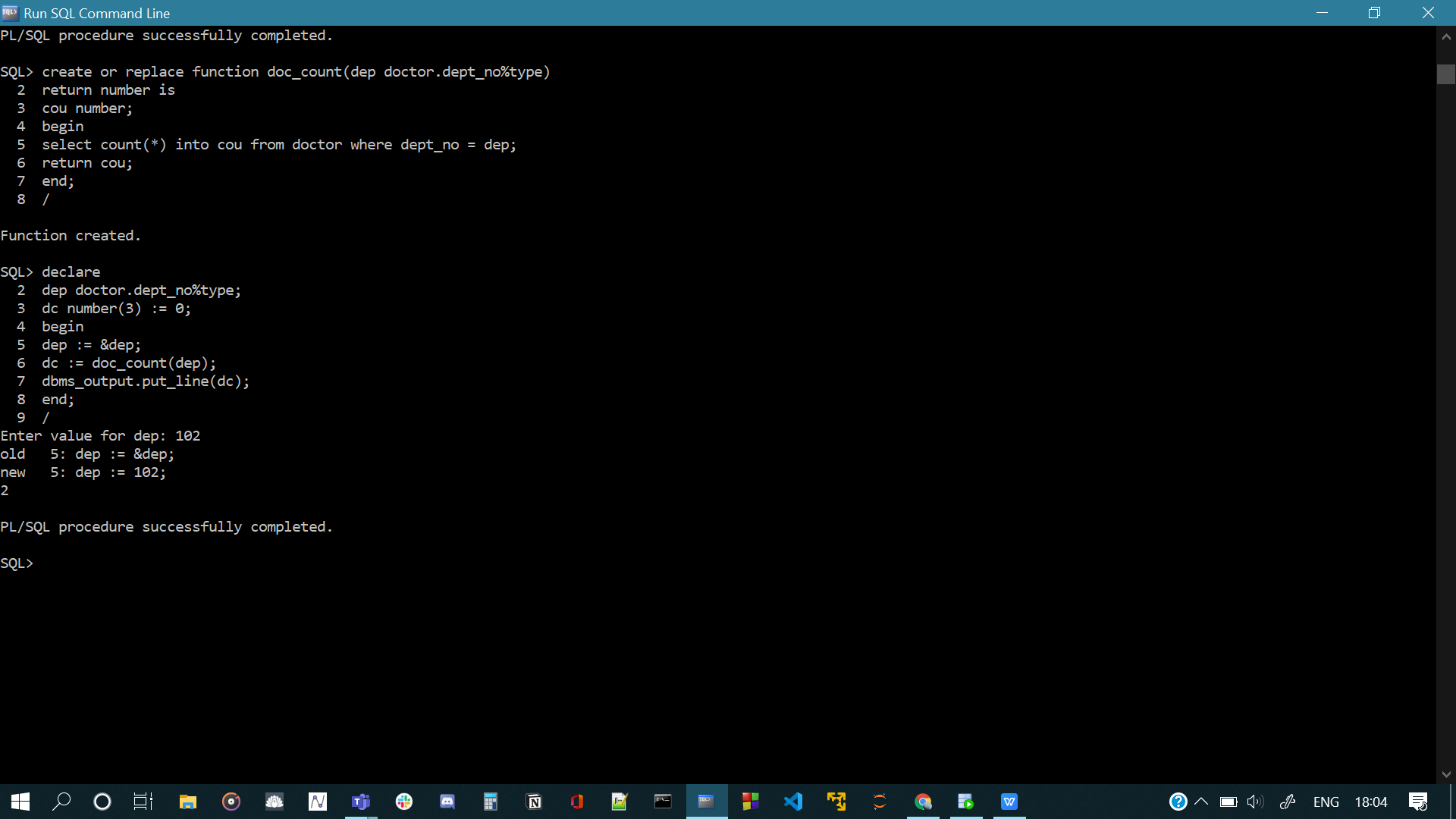
dep := &dep;

dc := doc\_count(dep);

dbms\_output.put\_line(dc);

end;

**Output:**



**Cursors:**

1. **Write a CURSOR to give 5% additional discount to all senior citizen patients.**

**Code:**

SQL> declare

2 cursor c is select pat\_id, age, discount from patient natural join hospital\_bill;

3 pid patient.pat\_id%type;

4 age patient.age%type;

5 dis hospital\_bill.discount%type;

6 begin

7 open c;

8 loop

9 fetch c into pid, age, dis;

10 exit when c%notfound;

11 if age > 45 then

12 dis := dis + 5;

13 dbms\_output.put\_line(dis);

14 end if;

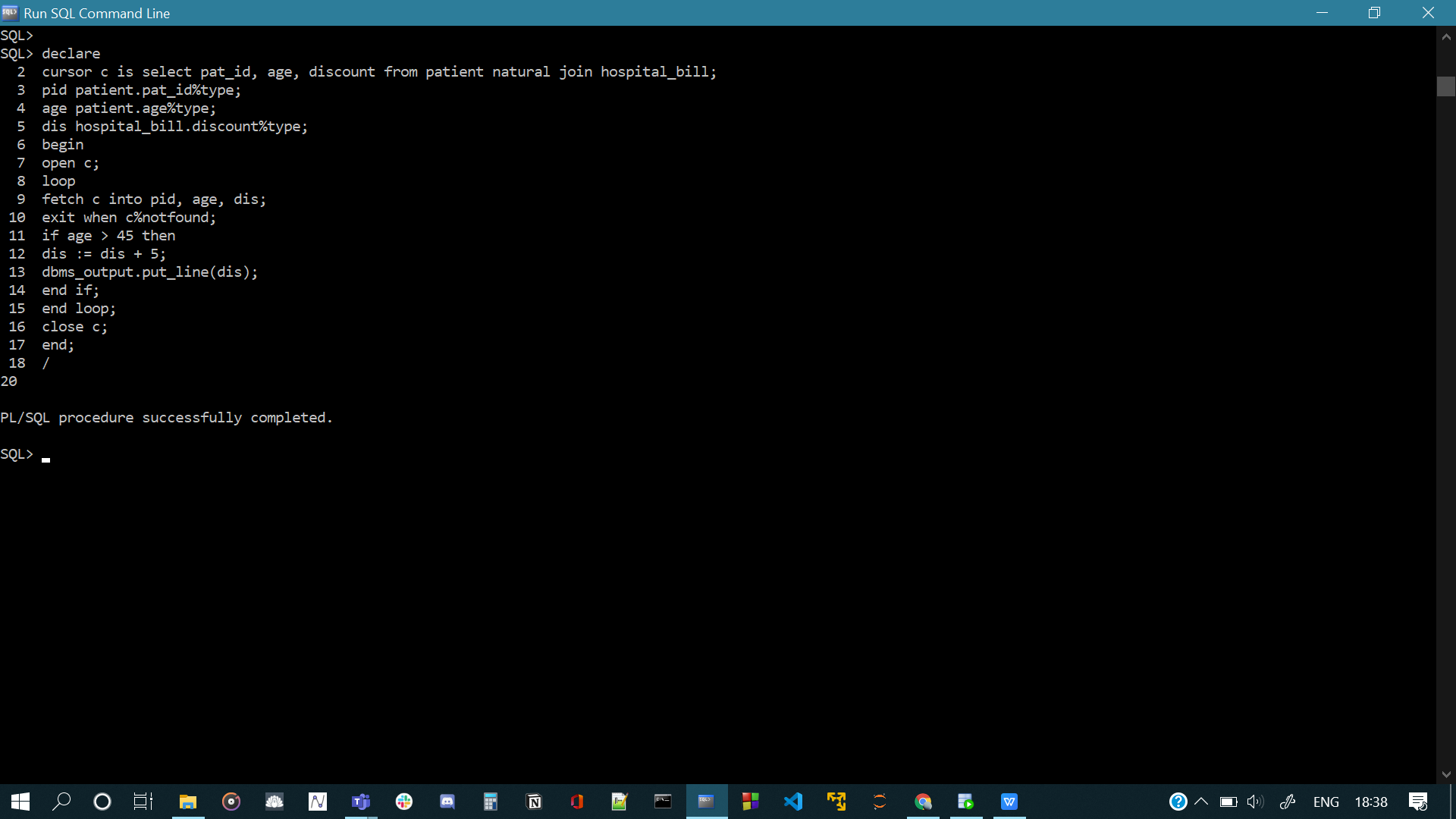
15 end loop;

16 close c;

17 end;

18 /

**Output:**



1. **Write a CURSOR to change the department number from 1 as 5 for all doctors with a qualification ‘MD’.**

**Code:**

declare

cursor c1 is select doc\_id, qualification, dept\_no from doctor;

did doctor.doc\_id%type;

qua doctor.qualification%type;

dep doctor.dept\_no%type;

begin

open c1;

loop

fetch c1 into did, qua, dep;

exit when c1%notfound;

if (qua = 'MD' and dep = 1) then

update doctor set dept\_no = 5 where doc\_id = did;

dbms\_output.put\_line('Updated');

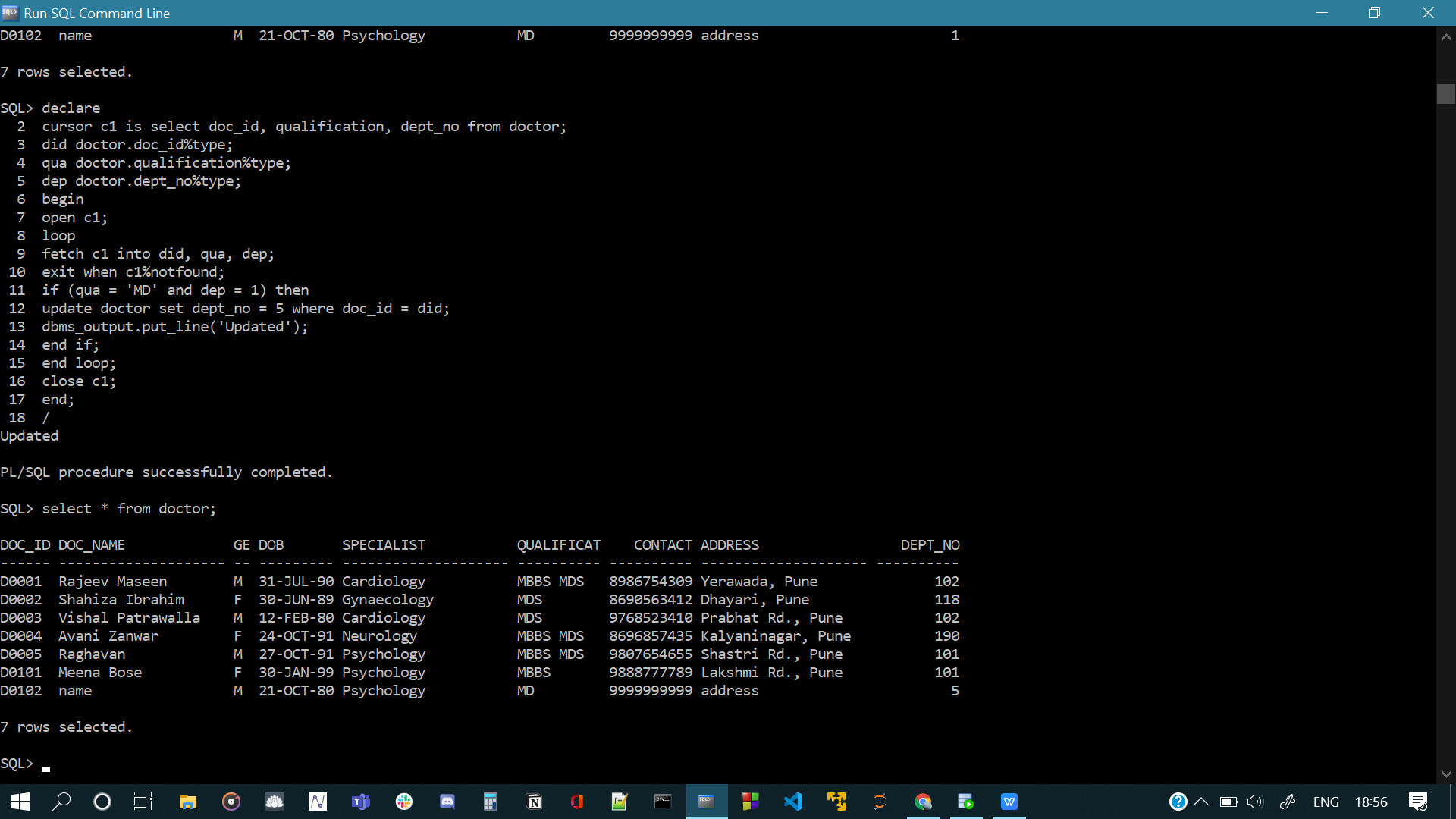
end if;

end loop;

close c1;

end;

**Output:**



**Functions and Procedures:**

1. **Write a PL/SQL stored function COUNT\_DOC to count the number of doctors who have treated at least 100 patients if given a doctor id as input parameter.**

**Code:**

create or replace function count\_doc(n number)

return number is

cursor c2 is select doc\_id, count(doc\_id) from appointment group by doc\_id;

cou number := 0;

did doctor.doc\_id%type;

dco number;

begin

open c2;

loop

fetch c2 into did, dco;

exit when c2%notfound;

if dco >= n then

cou := cou + 1;

end if;

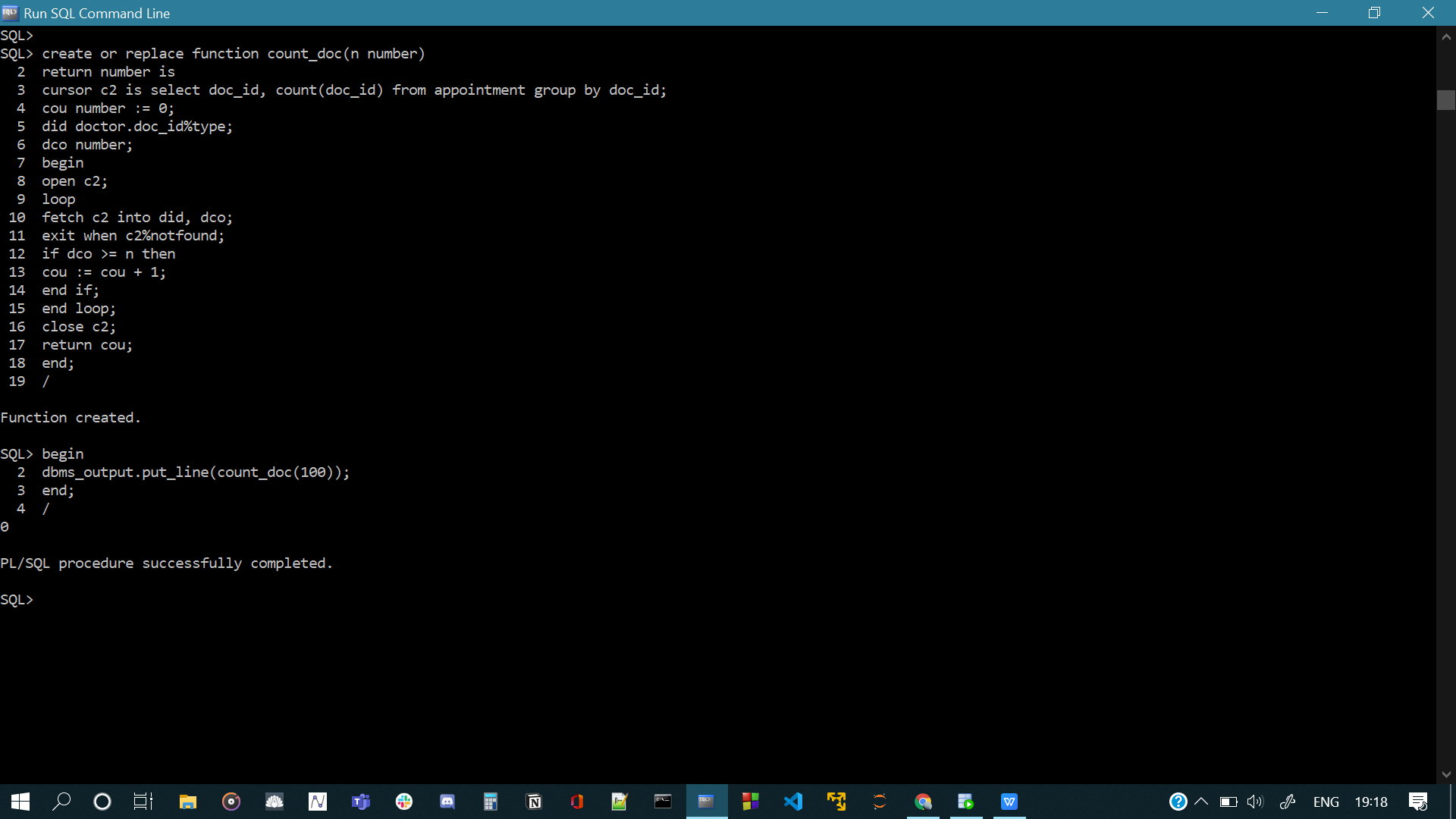
end loop;

close c2;

return cou;

end;

**Output:**



1. **Write a PL/SQL stored procedure to adjust the payment type of hospital bills to CASH if the patient id and amount details given as input.**

**Code:**

create or replace procedure cash(pid patient.pat\_id%type, amt hospital\_bill.bill\_amount%type)

is

begin

update hospital\_bill set payment\_type = 'CASH' where pat\_id = pid and bill\_amount = amt;

end;

declare

pid patient.pat\_id%type;

amt hospital\_bill.bill\_amount%type;

begin

pid := '&pid';

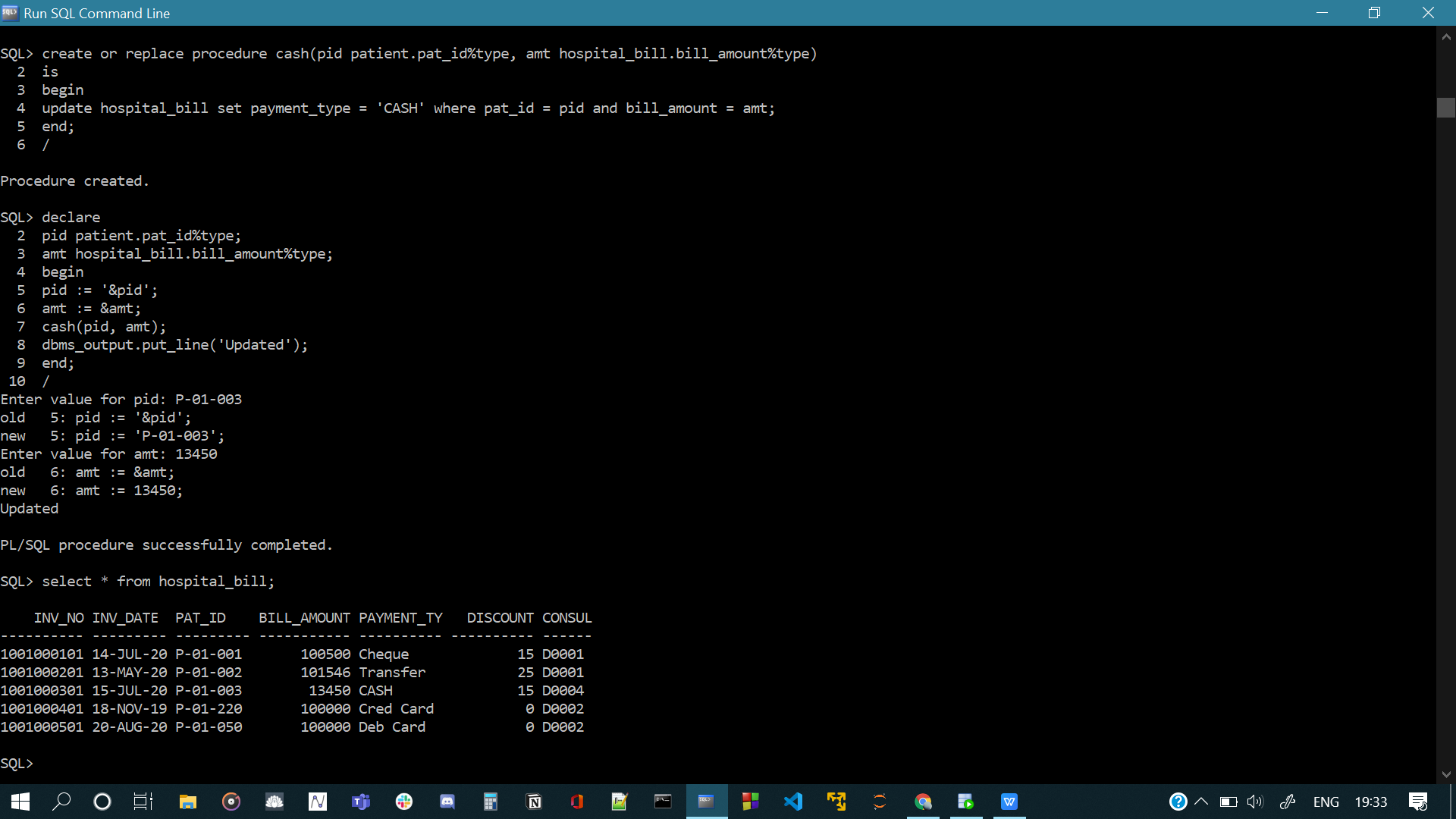
amt := &amt;

cash(pid, amt);

dbms\_output.put\_line('Updated');

end;

**Output:**



**Triggers:**

**Add an attribute with patients table to store the age of the patients. Then answer the following question;**

1. **Write a Trigger to find and fill the age of a patient whenever a patient record is inserted into patients table.**

**Code:**

create or replace trigger age\_patient

before insert on patient

for each row

declare

a patient.age%type;

pid patient.pat\_id%type;

begin

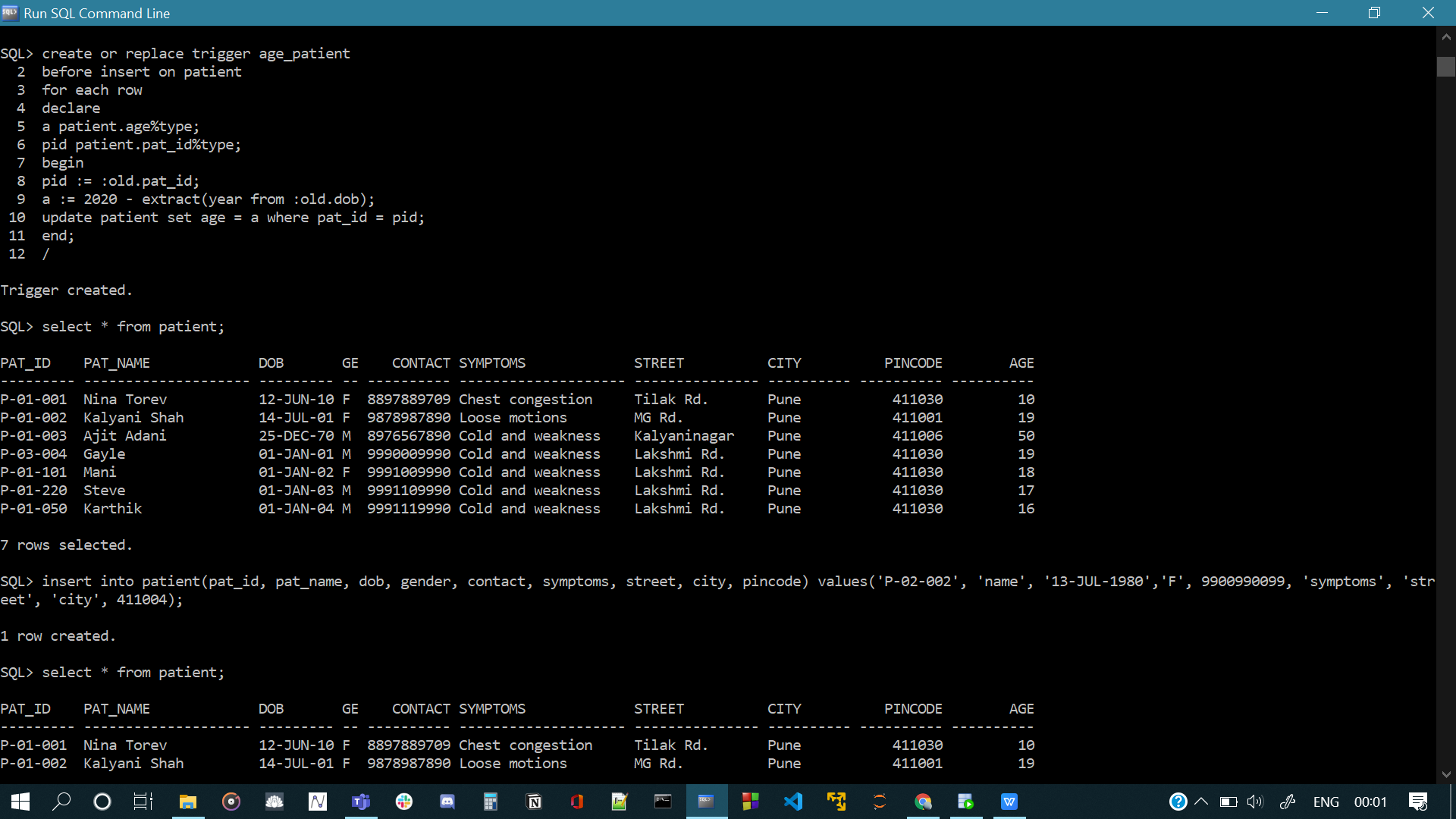
pid := :old.pat\_id;

a := 2020 - extract(year from :old.dob);

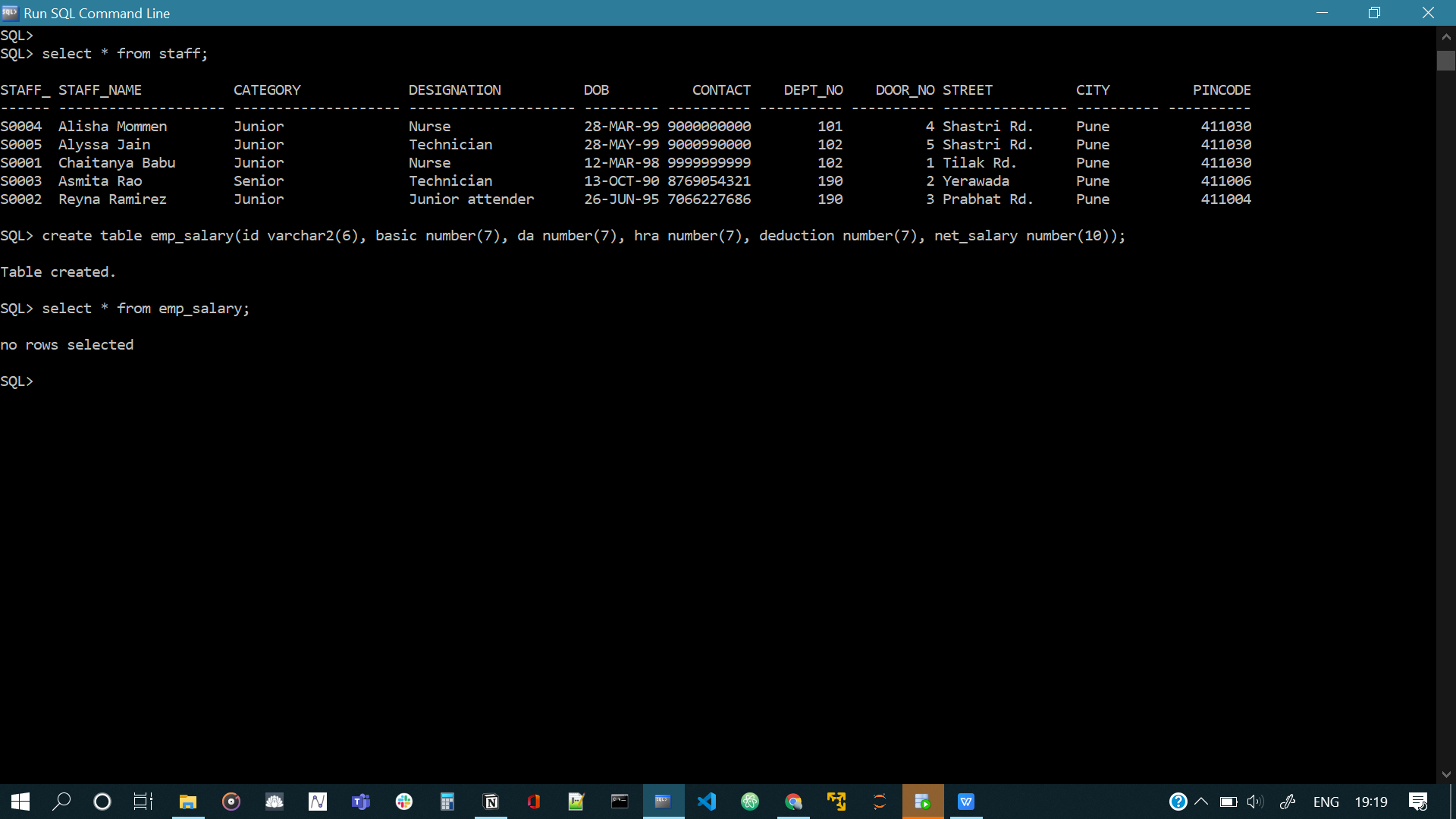
update patient set age = a where pat\_id = pid;

end;

**Output:**



**Create a table EMP\_SALARY with attributes ID, Basic, DA, HRA, Deduction, Net\_Salary. Here, ID refers the Staff\_ID of staff table. Treat ‘Net\_Salary’ as a derived attribute and don’t insert a value through insert operation. The value for Net Salary can be calculated as follows; Net\_Salary = Basic + DA + HRA – Deduction**



1. **Write a Trigger to perform the following; whenever new staff is recruited and a designation is assigned, insert an appropriate record into EMP\_SALARY table. Refer the following table for salary details.**

**Code:**

create or replace trigger insert\_emp\_salary

after insert on staff

for each row

enable

begin

if :new.Designation = 'Staff nurse' then

insert into emp\_salary values (:new.Staff\_Id, 6000, 6000, 2000, 2, (6000 + 6000 + 2000 - (0.02\*6000)));

ELSIF :new.Designation = 'Head nurse' then

insert into emp\_salary values (:new.Staff\_Id, 8000, 2500, 3000, 2, (8000 + 2500 + 3000 - (0.2\*8000)));

ELSIF :new.Designation = 'Technician' THEN

INSERT INTO emp\_salary VALUES (:new.Staff\_Id, 6000, 2000, 2000, 2, (6000 + 2000 + 2000 - (0.2\*6000)));

ELSIF :new.Designation = 'Senior technician' THEN

INSERT INTO emp\_salary VALUES (:new.Staff\_Id, 9000, 2500, 3500, 2, (9000 + 2500 + 3500 - (0.2\*9000)));

ELSIF :new.Designation = 'Junior attender' THEN

INSERT INTO emp\_salary VALUES (:new.Staff\_Id, 5000, 1500, 2000, 2, (5000 + 1500 + 2000 - (0.2\*5000)));

ELSIF :new.Designation = 'Senior attender' THEN

INSERT INTO emp\_salary VALUES (:new.Staff\_Id, 6500, 2000, 2000, 2, (6500 + 2000 + 2000 - (0.2\*6500)));

END IF;

END;

**Output:**

