**DBMS**

**ONLINE GROCERY SHOP**

**AIM**

To create online grocery shop application using plsql.

**SCHEMA**

Customer(cid,cname,caddress,email);

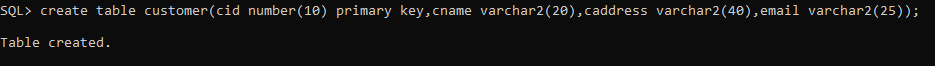
Product(pid,pname,price,discount);

Warehouse(wid,pid,wname,wlocation);

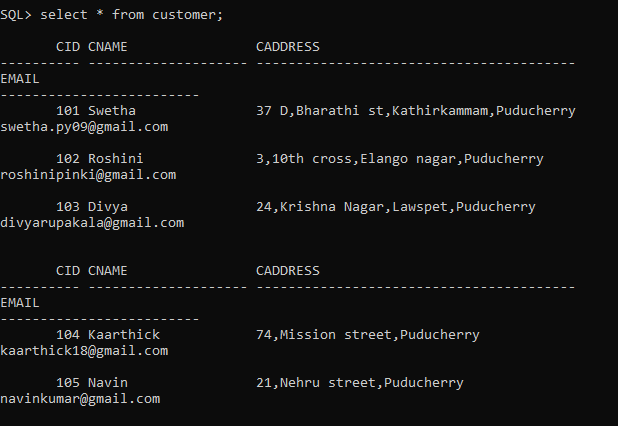
Inventories(pid,wid,stock);

Orders(oid,cid,pid,status,order\_date);

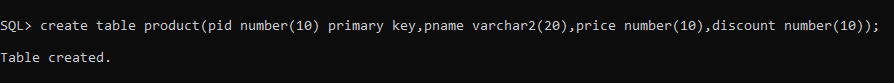
**CREATE TABLE CUSTOMER**



**DISPLAY TABLE CUSTOMER**



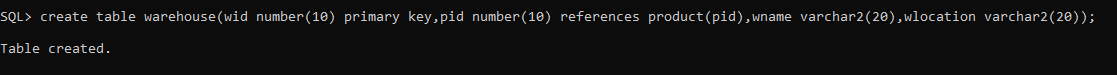
**CREATE TABLE PRODUCT**



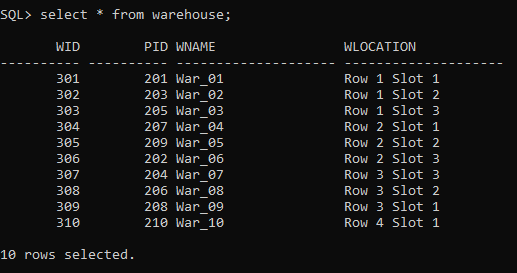
**DISPLAY TABLE PRODUCT**



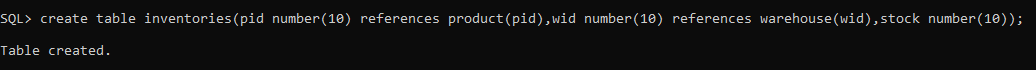
**CREATE TABLE WAREHOUSE**



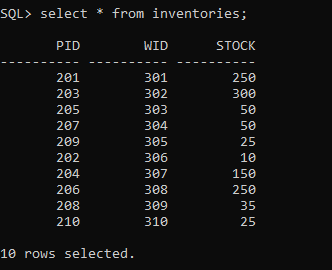
**DISPLAY TABLE WAREHOUSE**



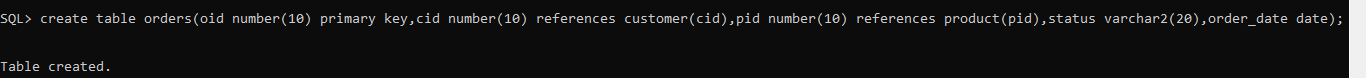
**CREATE TABLE INVENTORIES**



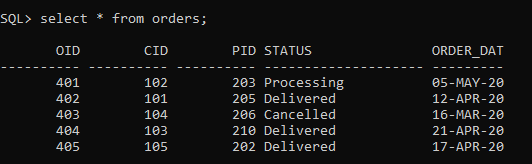
**DISPLAY TABLE INVENTORIES**



**CREATE TABLE ORDERS**



**DISPLAY TABLE ORDERS**



**PROCEDURE**

**1. Insert a record into the table customer**

CREATE OR REPLACE PROCEDURE "INSERTCUSTOMER"

(cid IN NUMBER,

cname IN VARCHAR2,

caddress IN VARCHAR2,

email IN VARCHAR2)

IS

BEGIN

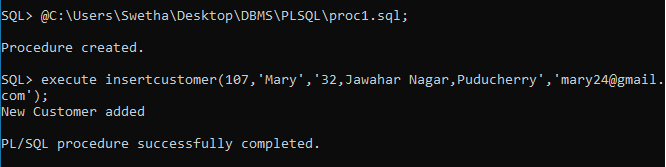
INSERT INTO customer values(cid,cname,caddress,email);

dbms\_output.put\_line('New Customer added');

END;

/

**EXECUTION AND OUTPUT**



**2. Update a record into the table customer**

CREATE OR REPLACE PROCEDURE "UPDATECUSTOMER"

IS

BEGIN

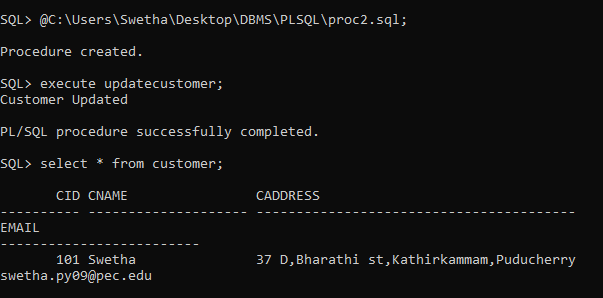
UPDATE customer SET email='swetha.py09@pec.edu' where cid=101;

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

END;

/

**EXECUTION AND OUTPUT**



**FUNCTION**

**1. To find the product whose cost is maximum**

CREATE OR REPLACE FUNCTION max\_cost

RETURN number IS

maxcost number(10):=0;

BEGIN

SELECT max(price) into maxcost FROM product;

RETURN maxcost;

END;

/

DECLARE

c number(10);

BEGIN

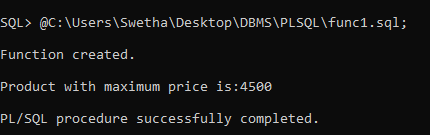
c:=max\_cost();

dbms\_output.put\_line('Product with maximum price is:' ||c);

END;

/

**EXECUTION AND OUTPUT**



**2. To find cost of the product after discount**

create or replace function calc(n1 in number, n2 in number)

return number

is

cost number(10);

begin

cost:= (n1-n2);

return cost;

end;

/

DECLARE

p number;

d number;

productprice number;

BEGIN

select price,discount into p,d

from product where pid=203;

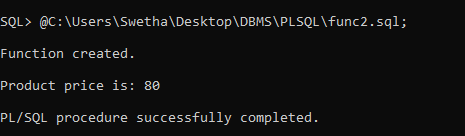
productprice:=calc(p,d);

dbms\_output.put\_line('Product price is: ' || productprice);

END;

/

**EXECUTION AND OUTPUT**



**CURSORS**

**1. To create an explicit cursor to display the product names with their cost in descending order.**

**Explicit cursor code**

DECLARE

CURSOR c IS SELECT pname,price from product order by price DESC;

BEGIN

FOR r in c

LOOP

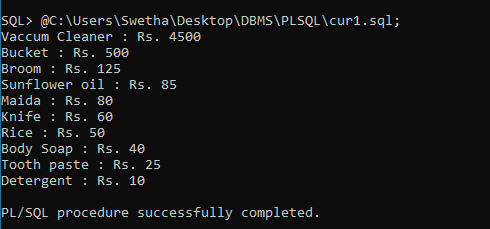
dbms\_output.put\_line(r.pname || ' : Rs. ' || r.price);

END LOOP;

END;

/

**EXECUTION AND OUTPUT**



**2. To create an implicit cursor to update all the product stock by 10.**

**Implicit Cursor Code**

DECLARE

total\_rows number(2);

BEGIN

UPDATE inventories

SET stock=stock+10;

IF sql%notfound THEN

dbms\_output.put\_line('No records updated');

ELSIF sql%found THEN

total\_rows:= sql%rowcount;

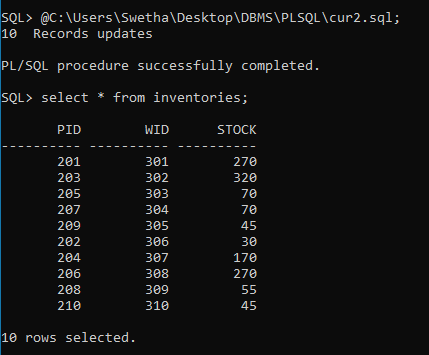
dbms\_output.put\_line(total\_rows || ' Records updates');

END IF;

END;

/

**EXECUTION AND OUTPUT**



**TRIGGERS**

**1. To create a trigger that lets us know that a record has been inserted after the insertion is successful.**

CREATE OR REPLACE TRIGGER insert\_order

AFTER INSERT ON orders

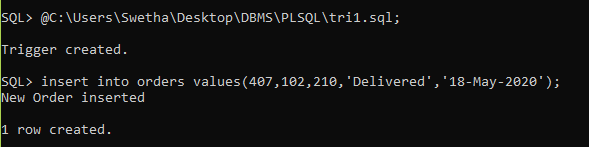
BEGIN

dbms\_output.put\_line('New Order inserted');

END;

/

**EXECUTION AND OUTPUT**



**2. To create a trigger that will display the stock difference of the products.**

CREATE OR REPLACE TRIGGER display\_stock\_changes

BEFORE DELETE OR INSERT OR UPDATE ON inventories

FOR EACH ROW

WHEN (NEW.pid > 0)

DECLARE

stock\_diff number;

BEGIN

stock\_diff := :NEW.stock - :OLD.stock;

dbms\_output.put\_line('Old Stock : ' || :OLD.stock);

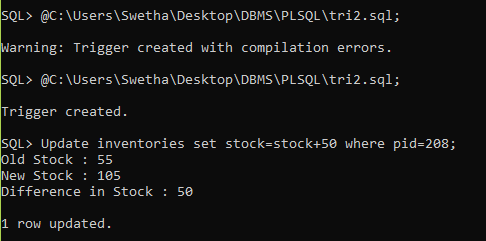
dbms\_output.put\_line('New Stock : ' || :NEW.stock);

dbms\_output.put\_line('Difference in Stock : ' || stock\_diff);

END;

/

**EXECUTION AND OUTPUT**



**PACKAGE**

**1. To create a package with two procedures updateRecord and delete Record.**

CREATE OR REPLACE PACKAGE pkg\_warehouse IS

PROCEDURE updateRecord;

PROCEDURE deleteRecord(o\_id orders.oid%type);

END pkg\_warehouse;

/

CREATE OR REPLACE PACKAGE BODY pkg\_warehouse IS

PROCEDURE updateRecord IS

BEGIN

UPDATE warehouse set wlocation='Row 5 Slot 6' where wid=302;

IF SQL%FOUND THEN

dbms\_output.put\_line('RECORD UPDATED');

ELSE

dbms\_output.put\_line('RECORD NOT UPDATED');

END IF;

END updateRecord;

PROCEDURE deleteRecord(o\_id orders.oid%type)

IS

BEGIN

DELETE from orders where oid=o\_id;

IF SQL%FOUND THEN

dbms\_output.put\_line('RECORD DELETED');

ELSE

dbms\_output.put\_line('RECORD NOT DELETED');

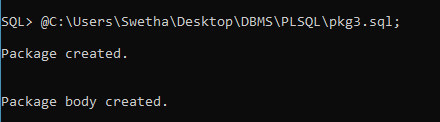
END IF;

END deleteRecord;

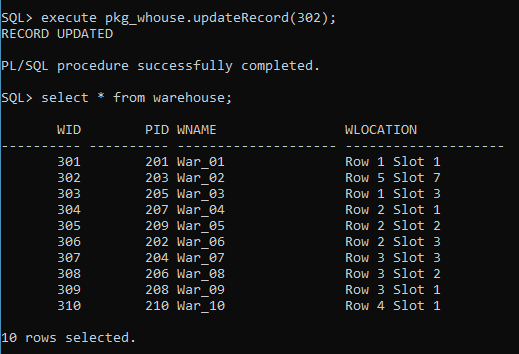
END pkg\_warehouse;

/

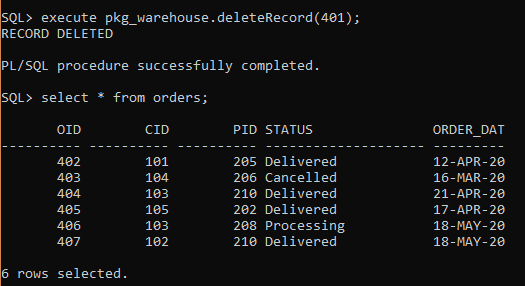
**EXECUTION AND OUTPUT**



**UPDATE RECORD**



**DELETE RECORD**



**HOSPITAL MANAGEMENT SYSTEM**

**AIM**

To create hospital management system using plsql.

**SCHEMA**

Patient(pid,pname,page,pgender,pheight,pweight);

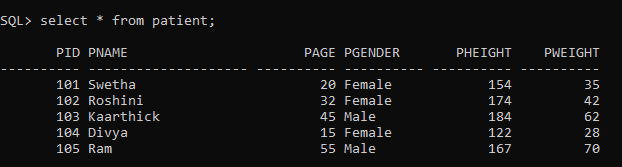
Doctor(did,dname,specialization,salary);

Treatment(did,pid,disease,joindate,discharge,fee);

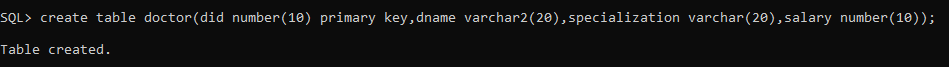
**CREATE TABLE PATIENT**

**C:\Users\Swetha\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (570).png**

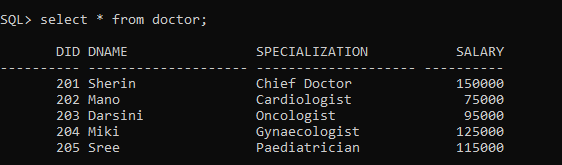
**DISPLAY TABLE PATIENT**

****

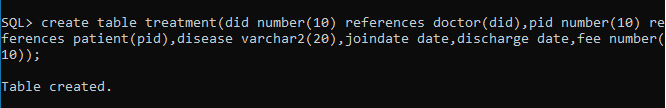
**CREATE TABLE DOCTOR**



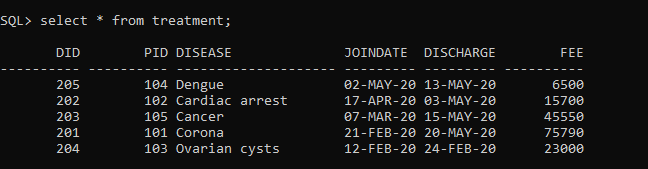
**DISPLAY TABLE DOCTOR**



**CREATE TABLE TREATMENT**

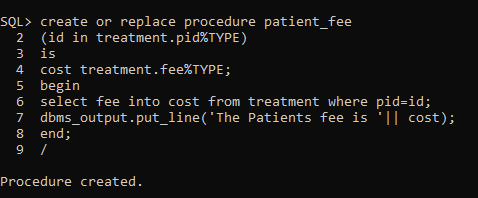
****

**DISPLAY TABLE TREATMENT**

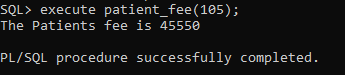


**PROCEDURE**

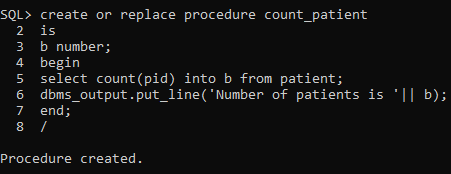
**1. To display the treatment fee of the patient**



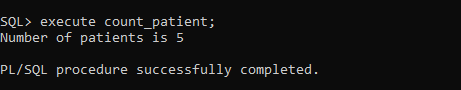
**EXECUTION AND OUTPUT**



**2. To find the number of patients in the hospital**



**EXECUTION AND OUTPUT**



**FUNCTION**

**1. To calculate the BMI of the patient**

create or replace function bmicalc(n1 in number, n2 in number)

return float

is

bmi float(8);

begin

bmi :=(n1/(n2\*n2))\*10000;

return bmi;

end;

/

DECLARE

h number;

W number;

patientbmi float;

BEGIN

select pheight,pweight into h,w

from patient where pid=104;

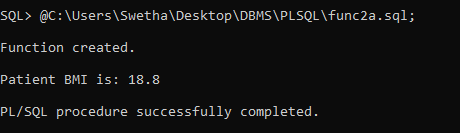
patientbmi:=bmicalc(w,h);

dbms\_output.put\_line('Patient BMI is: '||patientbmi);

END;

/

**EXECUTION AND OUTPUT**



**2. To increment the salary of the doctors by 10 percent**

create or replace function calc(n1 in number, n2 in number)

return float

is

inc float(8);

begin

inc :=n1\*(1+(n2/100));

return inc;

end;

/

DECLARE

sal number;

increment float;

BEGIN

select salary into sal

from doctor where did=201;

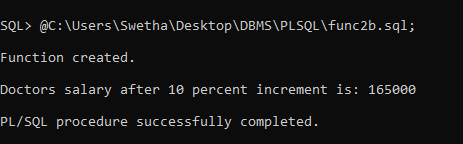
increment:=calc(sal,10);

dbms\_output.put\_line('Doctors salary after 10 percent increment is: '||increment);

END;

/

**EXECUTION AND OUTPUT**



**CURSORS**

**1. To create an explicit cursor to display the product names with their cost in descending order.**

**Explicit cursor code**

DECLARE

name patient.pname%type;

h patient.pheight%type;

w patient.pweight%type;

bmi float(10);

CURSOR pat is

SELECT pname,pheight,pweight FROM patient;

BEGIN

OPEN pat;

LOOP

FETCH pat into name,h,w;

EXIT WHEN pat%notfound;

bmi:=(w/(h\*h))\*10000;

IF bmi<18 THEN

dbms\_output.put\_line(name||':Underweight BMI:'||bmi);

ELSIF bmi<25 THEN

dbms\_output.put\_line(name||': Normal BMI:'||bmi);

ELSE

dbms\_output.put\_line(name||': Overweight BMI:'||bmi);

END IF;

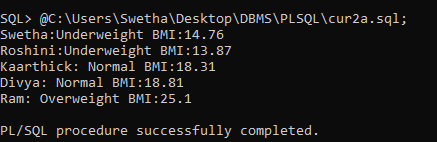
END LOOP;

CLOSE pat;

END;

/

**EXECUTION AND OUTPUT**



**2. To create an implicit cursor to update all the product stock by 10.**

**Implicit Cursor Code**

DECLARE

total\_rows number(2);

BEGIN

UPDATE treatment

SET fee=fee+500;

IF sql%notfound THEN

dbms\_output.put\_line('No records updated');

ELSIF sql%found THEN

total\_rows:= sql%rowcount;

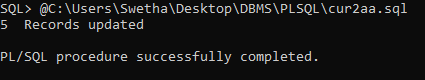
dbms\_output.put\_line(total\_rows || ' Records updated');

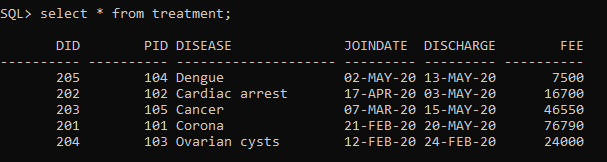
END IF;

END;

/

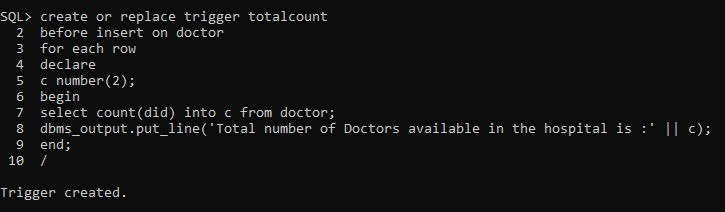
**EXECUTION AND OUTPUT**



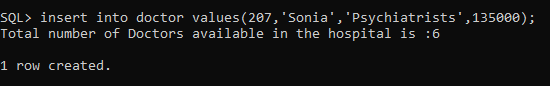


**TRIGGERS**

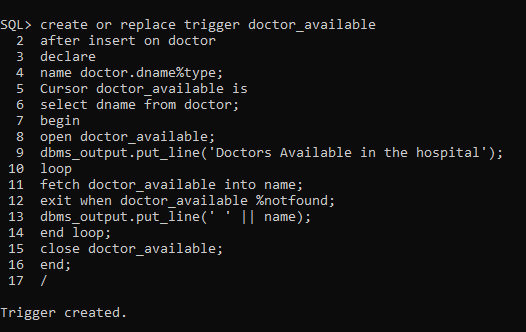
**1. To create a trigger that displays the total number of doctors available in the hospital before inserting a doctor**



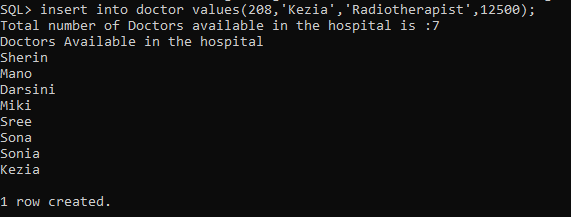
**EXECUTION AND OUTPUT**



**2. To create a trigger that displays names of doctors available in the hospital after inserting a doctor**



**EXECUTION AND OUTPUT**



**PACKAGE**

**1. To create a package with two procedures to fid count of patients and doctors in the hospital.**

create or replace package hospi

as

procedure patient\_count;

procedure doctor\_count;

end hospi;

/

create or replace package body hospi as

procedure patient\_count

as

b number;

begin

select count(pid) into b from patient;

dbms\_output.put\_line('Total number of patients in the hospital is ' || b);

end;

procedure doctor\_count

as

c number;

begin

select count(did) into c from doctor;

dbms\_output.put\_line('Total number of doctors in the hospital is ' || c);

end;

end hospi;

/

**EXECUTION AND OUTPUT**

