rksuthar19 / dbbook Public

Code Issues Pull requests Actions Projects Wiki Security Insig

master

dbbook / manuscript / chapter2.txt

rksuthar19 finalizing content History

1 contributor

2973 lines (1379 sloc) | 51 KB

```
# Chapter 2

---------PL*SQL Assignments--------

#Exercise 1


1. Write a program that computes the perimeter and the area of a rectangle. Define your own values
length and width. (Assuming that L and W are the length and width of the rectangle, Perimeter =
2*(L+W) and Area = L*W. Display the output on the screen using dbms_output.put_line.


Declare

Length number:=&length;

Width number:=&width;

Area number;

Parimeter number;

Begin

Area:=length*width;

Parimeter:= 2*(length+width);

dbms_output.put_line('AREA OF RECTANGLE IS:'||Area);
```

```
dbms_output.put_line('PARIMETER OF RECTANGLE IS:'||PARIMETER);

End;

/

2. Write a program that declares an integer variable called num, assigns a value to it, and comput
inserts into the tempp table the value of the variable itself, its square, and its cube.



SQL> create table tempp

 (

 item number,

 square number,

 cube number

 );


Table created.


SQL> DECLARE

 num number:=&num;

 begin

 insert into tempp values(num, num*num, num*num*num);

 end;

SQL> /

Enter value for num: 5

old 2: num number:=&num;

new 2: num number:=5;

```

```
PL/SQL procedure successfully completed.


SQL> select * from tempp;


ITEM            SQUARE          CUBE

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

5                       25                      125


3. Convert a temperature in Fahrenheit (F) to its equivalent in Celsius (C) and vice versa. The re
formulae are:-

C= (F-32)*5/9

F= 9/5*C + 32

Display the output on the screen using dbms_output.put_line. Data has to be input by the user.


Declare

 F number:=&Fahrenheit;

 C number:=&Celsius;

 result number;

Begin

result:=(F-32)*5/9;

dbms_output.put_line('fahrenheit value entered '||F||' equal to celsius :'||' '||result);

result:=9/5*C+32;

dbms_output.put_line('fahrenheit value entered '||F||' equal to celsius :'||' '||result);

end;
```

```
/


4. Convert a number of inches into yards, feet, and inches. For example, 124 inches equals 3 yards
and 4 inches. Display the output on the screen using dbms_output.put_line. Data has to be input by
user.



declare

inch int:=&howmanyinch;

yard int;

foot int;

begin

foot:=inch/12;

yard:=foot / 3;

foot:=foot mod 3;

inch:=inch mod 12;

dbms_output.put_line(yard||' '||foot||' '||inch);

end;

/


5. Write a program that enables a user to input an integer. The program should then state whether
integer is evenly divisible by 5. (Use decode instead of IF statement where required). Display the
output on the screen using dbms_output.put_line. Data has to be input by the user.



declare

num number:=&enter_value;

```

```
begin

if (num mod 5=0) then

dbms_output.put_line('GIVEN NO. IS DIVISIBLE BY 5');

else

dbms_output.put_line('not divisible by 5');

end if;


end;

/




begin

select decode(num mod 5, 0, 'divisible', 'nod devisible') into result from dual;

dbms_output.put_line(result);

end;


6. Your block should read in two real numbers and tell whether the product of the two numbers is e
or greater than 100. Display the output on the screen using dbms_output.put_line. (Use decode inst
of IF statement where required). Data has to be input by the user.



declare

a number:=&firstno;

b number:=&secondno;

result varchar2(50);

begin

select decode(trunc(a*b/100),0,'LESS THEN 100','GREATER THEN OR EQUAL TO 100') into result
```

```
from dual;

dbms_output.put_line(result);

end;

/



PL*SQL

#Exercise 2



1. In a PL*SQL block, create a datatype by the name of addr_type. It should contain the following
components:-

name varchar2 (20)

street varchar2 (30)

city varchar2 (20)

state varchar2 (15)

Your block should accept the names and addresses of 4 employees in 4 different variables of dataty
addr_type. Output the names and addresses of the 4 employees on the screen in the form of Labels a
shown below:-

************************************************************

* Name:- Jack ** Name:- Scott *

* Street:- M.G. Road ** Street:- Bhosale Marg *

* City:- Mumbai ** City:- Chennai *

* State:- Maharashtra ** State:- Tamil Nadu *

************************************************************

************************************************************

* Name:- King ** Name:- Adams *

* Street:- Lane No:-2 ** Street:- P. M. Road *
```

```
* City:- Nagpur ** City:- Bangalore *

* State:- Maharashtra ** State:- Karnataka *




SQL> create type addr_type as object

 (

 name varchar2(20), street varchar2(30), city varchar2(20), state varchar2(5

 0));







declare

 temp1 addr_type := addr_type('jack','mg road','mumbai','maharashtra');

 temp2 addr_type := addr_type('scott','bhosale marg','chennai','tamil nadu');

 temp3 addr_type := addr_type('king','lan no:-2','nagpur','maharashtra');

 temp4 addr_type := addr_type('adams','pm road','bangalore','karnataka');

begin

 dbms_output.put_line('**************************************************');

 dbms_output.put_line('* Name:- '||temp1.name||' ** Name:- '||temp2.name||' *');

 dbms_output.put_line('* Street:- '||temp1.street||' ** Street:- '||temp2.street||'*');

 dbms_output.put_line('* Street:- '||temp1.city||' ** Street:- '||temp2.city||' *');

 dbms_output.put_line('* Street:- '||temp1.state||' ** Street:- '||temp2.state||' *');

 dbms_output.put_line('**************************************************');


 dbms_output.put_line('**************************************************');
```

```
 dbms_output.put_line('* Name:- '||temp3.name||' ** Name:- '||temp4.name||' *');

 dbms_output.put_line('* Street:- '||temp3.street||' ** Street:- '||temp4.street||' *');

 dbms_output.put_line('* Street:- '||temp3.city||' ** Street:- '||temp4.city||' *');

 dbms_output.put_line('* Street:- '||temp4.state||' ** Street:- '||temp4.state||' *');

 dbms_output.put_line('**************************************************');

---dbms_output.put_line(temp1.street);

 end;

/



PL*SQL

#Exercise 3



1. Input a number and determine whether it is within a given range (for example, between 1 and 10)
low and high values of the range may be input by the user rather than be fixed by the program. Dis
the output on the screen using dbms_output.put_line.


----------------------------------------------ANONYMS-PROGRAM-----------------------------------

declare

 lb number:=&lower_bound;

 ub number:=&upper_bound;

 data number:=&num;

begin

 if data>lb then

 if data<ub then

 dbms_output.put_line('number is between'||lb||' and '||ub);

```

```
 end if;

 else

 dbms_output.put_line('number is not between'||lb||' and '||ub);

 end if;

end;

/

2. Input three positive integers representing the sides of a triangle, and determine whether they
triangle. Hint: In a triangle, the sum of any two sides must always be greater than the third side
the output on the screen using dbms_output.put_line.


----------------------------------------------ANONYMS-PROGRAM-----------------------------------

declare

 a number:=&first_side;

 b number:=&second_side;

 c number:=&third_side;

begin

 if a+b>c then

 if b+c>a then

 if c+a>b then

 dbms_output.put_line('valid triangle');

 end if;

 end if;

 else

 dbms_output.put_line('invalid triangle');

 end if;

end;

```

```
/
```

3. Check if a given a year is a leap year. The condition is:-

year should be (divisible by 4 and not divisible by 100) or (divisible by 4 and divisible by 400.)
output on the screen using dbms_output.put_line. The year should be input by the user.

```
----------------------------------------------ANONYMS-PROGRAM-----------------------------------
declare
 year int:=&enter_yera;
 result1 int;
 result2 int;
 result3 int;
begin
 result1:=mod(year,100);
 result2:=mod(year,400);
 result3:=mod(year,4);
 if (result3=0) and not(result1=0) or (result3=0) and (result2=0) then
 dbms_output.put_line('leap year');
 else
 dbms_output.put_line('not a leap year');
 end if;
end;
/
```

```
471
472 4. Ask the user to enter the weight of an apple box. If the
473
474 weight is >= 10 kg, rate =Rs. 5/kg
475
476 weight is < 10 kg, rate = Rs. 7/kg
477
478 Calculate the cost of the apple box. Display the output on the screen using dbms_output.put_line.
479
480 ----------------------------------------------ANONYMS-PROGRAM-----------------------------------
481
482 declare
483
484  weight number:=&apple_weight;
485
486  rate number;
487
488 begin
489
490  if (weight>=10) then
491
492  rate:=5;
493
494  else
495
496  rate:=7;
497
498  end if;
499
500  dbms_output.put_line('TOTAL WEIGHT'||weight||'TOTAL COST '||(weight*rate));
501
502 end;
503
504 /
505
506 5. Program should accept the age of the user. Depending upon the following conditions it should ou
507
508 age <18 years,  child
509
510 age >= 18 years and <21 years,  major
511
512 age>= 21years  adult
513
514  Display the output on the screen using dbms_output.put_line.
515
516  ----------------------------------------------ANONYMS-PROGRAM-----------------------------------
517
518 declare
519
```

```
 age number:=&user_age;

 user varchar(20);

begin

 if (age<18) then

 user:='child';


 elsif (age>18) and (age<21) then

 user:='major';

 else

 user:='adult';

 end if;

 dbms_output.put_line('user is '||user);

end;

/

6. Write a program that asks the user to input two character strings. Your program should then det
character string exists inside another character string. Display the above on the screen using
dbms_output.put_line.

 ------------------------------------------------ANONYMS-PROGRAM-----------------------------------

declare

 str1 varchar(30):='&first_string';

 str2 varchar(30):='&second_string';

begin

 if instr(str1,str2)>0 or instr(str2,str1)>0 then

 dbms_output.put_line('one string contained in another');

 else

 dbms_output.put_line('both are different string');

```

```
 end if;

end;

/

7. Suppose the grade obtained by a student depends upon his scores and the grading rule is as foll

Scores Grades

 95-100 A

 85-94 B

 70-84 C

 60-69 D

 0-59 E

Write a block to accept a student s marks and accordingly output his grade. Display the output on
screen using dbms_output.put_line.

-----------------------------------------------ANONYMS-PROGRAM-----------------------------------

declare

 score number:='&score';

 grade varchar(1);

begin

 if score<=59 then

 grade:='E';

 elsif score<=69 then

 grade:='D';

 elsif score<=84 then

 grade:='C';

 elsif score<=94 then


 grade:='B';
```

```
 elsif score<=100 then

 grade:='A';

 end if;

 dbms_output.put_line('your marks are'||score||' your grade is --->'||grade);

end; /
```

8. A company manufactures three products:- computer stationery, fixed disks and computers. The fol
codes are used to indicate them:-

Product Code

Computer Stationery 1

Fixed Disks 2

Computers 3

The company has a discount policy as follows:-

Product Order amount Discount rate

Computer stationery Rs. 5000 or more 12%

Computer stationery Rs. 3000 or more 8%

Computer stationery Below Rs. 3000 2%

Fixed disks Rs. 20000 or more 10%

Fixed disks Rs. 15000 or more 5%

Computers Rs. 50000 or more 10%

Computers Rs. 25000 or more 5%

Write a program to accept the order details i.e. product code and order amounts for the products,
the discount amounts as per this policy and output the net order amount. Display the output on the
using dbms_output.put_line.

```
----------------------------------------------ANONYMS-PROGRAM-----------------------------------

declare

 code number:='&item_code';--if it is a procedure item_code will be permanently saved in user_sour
```

```
 amt number:='&item_amount';

 disc number;

 prod varchar2(30);

begin

 if code=1 then

 prod:='Computer Stationary';

 if amt>=5000 then

 disc:=.12;

 elsif amt>=3000 then

 disc:=.08;

 else

 disc:=.02;

 end if;

 elsif code=2 then

 prod:='Fixed Disks';

 if amt>=20000 then

 disc:=.12;

 elsif amt>=15000 then

 disc:=.08;

 end if;


 elsif code=3 then

 prod:='Computers';

 if amt>=50000 then

 disc:=.12;
```

```
 elsif amt>=25000 then

 disc:=.08;

 end if;

 end if;

 dbms_output.put_line('_____________________________________');

 dbms_output.put_line('PRODUCT '||prod);

 dbms_output.put_line('ORDER AMOUNT '||amt);

 dbms_output.put_line('DISCOUNT IS '||disc);

 dbms_output.put_line('AFTER DISCOUNT '||((1-disc)*amt));

 dbms_output.put_line('_____________________________________');

end;

/
```

PL*SQL

#Exercise 4

1. Write a program containing a loop that iterates from 1 to 1000 using a variable I, which is incremented each time around the loop. The program should output the value of I every hundred iterations (i.e., the output should be 100, 200, etc). Display the output on the screen using dbms_output.put_line.

```
declare

 i int:=1;

begin

 loop
```

```
765
766   dbms_output.put_line(i);
767
768   dbms_output.put_line('____');
769
770   exit when i>1000;
771
772   i:=i+100;
773
774   end loop;
775
776 end;
777
778 /
779
780
781
782 2. Write a program that examines all the numbers from 1 to 999, displaying all those for which the
783 the cubes of the digits equal the number itself. Display the output on the screen using
784 dbms_output.put_line.
785
786
787
788
789 declare
790
791   i int:=1;
792
793   x int;
794
795   y int;
796
797   z int;
798
799 begin
800
801   loop
802
803   x:=mod(i,10);
804
805
806   y:=mod(i,10);
807
808   z:=mod(i,10);
809
810   if (x*x*x+y*y*y+z*z*z)=i then
811
812   dbms_output.put_line(i);
813
```

```
 dbms_output.put_line('____');

 end if;

 exit when i=1000;

 i:=i+1;

 end loop;

end;

/
```

3. Write a PL*SQL block that reads in a minimum and maximum value for a radius, along with an increment factor, and generates a series of radii by repeatedly adding the increment to the minimu until the maximum is reached. For each value of the radius, compute and display the circumference, area, and volume of the sphere. (Be sure to include both the maximum and the minimum values.). Validate each of the input values to be sure they are positive. If the minimum is typed in place o maximum, swap the values within the program, and continue execution. Display the results on the screen using dbms_output.put_line.

```
 declare

 num int;

 j int:=1;

 x int;

 y int;

 z int;

 begin

 loop

 num:=j;

 x:=mod(num,10);

 num:=trunc(num/10);

 y:=mod(num,10);

 num:=trunc(num/10);

```

```
 z:=mod(num,10);

 if (x*x*x+y*y*y+z*z*z)=j then

 dbms_output.put_line(j);


 dbms_output.put_line('____');

 end if;

 exit when j=999;

 j:=j+1;

 end loop;

end;

/

Allow any positive integer to be typed in. The program should count how many times the number has
doubled before it reaches 1 million. Display the results on the screen using dbms_output.put_line.

 declare

 num int:='&number';

 counter int:=1;

 begin

 loop

 counter:=counter+1;

 num:=2*num;

 exit when num>1000000;

 end loop;

 dbms_output.put_line('number'||num||' needs '||counter||' times
multiplication to reach till 1 million');

 end;

 /

```

```
4. A palindrome is a word that is spelled the same forward and backward, such as level, radar, etc
program to read in a five letter word from the user and determine whether it is a palindrome. Disp
the results on the screen using dbms_output.put_line.



declare

 str varchar2(50):='&string';

 counter int:=length(str);

begin

 dbms_output.put_line(counter);

 loop

 exit when counter=0;

 exit when not(substr(str,counter,1)=substr(str,((length(str)+1)-counter),1));

 counter:=counter-1;

 end loop;

 if counter=0 then

 dbms_output.put_line(str||'is palindrom');

 else

 dbms_output.put_line(str||'is not palindrom');

 end if;

end;

/



5. Modify the above program to accept a variable length word. This requires determining how many
characters are read in.


 declare

```

```
 str varchar2(50):='&string';

 counter int:=length(str);


begin

 dbms_output.put_line(counter);

 loop

 exit when counter=0;

 exit when not(substr(str,counter,1)=substr(str,((length(str)+1)-counter),1));

 counter:=counter-1;

 end loop;

 if counter=0 then

 dbms_output.put_line(str||'is palindrom');

 else

 dbms_output.put_line(str||'is not palindrom');

 end if;

end;

/



6. Write a program to read in a number and print it out digit by digit, as a series of words. For
number 523 would be printed as "five two three". Use decode function within a for loop. Display th
results on the screen using dbms_output.put_line.




declare

 num varchar(10):='&number';

 i varchar(1);

 c int:=length(num);
```

```
 result varchar(10);

begin

 dbms_output.put_line('ENTERED NO. IS');

 loop

 i:=substr(num,1,1);

 num:=substr(num,2);

 select decode(i, 1, 'one', 2, 'two', 3, 'three', 4, 'four', 5, 'five', 6, 'six', 7, 'seven', 8, '
,'nine','zero') into result from dual;

 dbms_output.put_line(result);

 exit when c=1;

 c:=c-1;

 end loop;

end;/




PL*SQL

#Exercise 5



1. Create a table SCHOOL which has the following structure:-


Roll _no Number 4

Name Varchar2 20

Section Number 4

Class Character 7

Oracle Number 3

```

```
Dev_2000 Number 3

 Fill in the following sample data:-

Roll no. Name Section Class Oracle Dev_2000

1 Mukesh Khanna 9012 Working 55 80

2 Rajiv Chawala 9025 Student 75 85

3 Pramila Bordes 9025 Working 45 45

4 Nitish Bharadwaj 9025 Working 67 75

5 Anita Sood 9012 Student 86 72

6 Kalyani Deshmukh 9012 Working 55 65

7 Rakesh Surana 9025 Working 95 95

8 Alok Kumar Nath 9025 Working 25 40

9 Sushmita Bannerjee 9025 Student 73 83

10 Pranay Aiyyer 9012 Student 62 85

11 Shalini Patel 9012 Student 35 00

12 Ketan Tendulkar 9012 Working 83 98

13 Arun Trivedi 9012 Working 67 53

14 Victor D souza 9025 Working 59 63

15 Sarah Ahmed 9025 Student 65 73

 Create another table with the following structure:-

Roll_no Number 4

Total Number 3

Percent Number 5,2

Grade Varchar2 10

Insert into this table the total marks, percentage and grades of the respective students. The rule
grades are as follows:-

```

```
For working persons

Percentage Grade

< 50 % FAIL

>= 50 % PASS



For students

Percentage Grade

< 40% FAIL

40 - 49.99% C

50   59.99% B

60   79.99% A


>= 80% HONOURS

----------------------------------------------------------tables----------------------------------------
create table SCHOOL

(

Roll_no Number(4),

Name Varchar2(20),

Section Number(4),

Class Character(7),

Oracle Number(3),

Dev_2000 Number(3)

)

/

insert all

into school values(1,'Mukesh Khanna',9012,'Working',55,80)
```

```
1157
1158  into school values(2,'Rajiv Chawala',9025,'Student',75,85)
1159
1160  into school values(3,'Pramila Bordes',9025,'Working',45,45)
1161
1162  into school values(4,'Nitish Bharadwaj',9025,'Working',67,75)
1163
1164  into school values(5,'Anita Sood',9012,'Student',86,72)
1165
1166  into school values(6,'Kalyani Deshmukh',9012,'Working',55,65)
1167
1168  into school values(7,'Rakesh Surana',9025,'Working',95,95)
1169
1170  into school values(8,'Alok Kumar Nath',9025,'Working',25,40)
1171
1172  into school values(9,'Sushmita Bannerjee',9025,'Student',73,83)
1173
1174  into school values(10,'Pranay Aiyyer',9012,'Student',62,85)
1175
1176  into school values(11,'Shalini Patel',9012,'Student',35,00)
1177
1178  into school values(12,'Ketan Tendulkar',9012,'Working',83,98)
1179
1180  into school values(13,'Arun Trivedi',9012,'Working',67,53)
1181
1182  into school values(14,'Victor DÆsouza',9025,'Working',59,63)
1183
1184  into school values(15,'Sarah Ahmed',9025,'Student',65,73)
1185
1186  select * from dual
1187
1188  /
1189
1190  ------------------------------------------------------------------------------------------------------
1191  create table oracle_result
1192
1193  (
1194
1195  Roll_no Number (4),
1196
1197  Total Number (3),
1198
1199  Percent Number (5,2),
1200
1201  Grade Varchar2 (10)
1202
1203  )/
1204
1205  ------------------------------------------------------------------------------------------------------
```

```
create table Dev_2000_result

(

Roll_no Number (4),

Total Number (3),

Percent Number (5,2),

Grade Varchar2 (10)

)

/

-----------------------------------------------anonyms procedure for Dev_2000_result result-------
declare


v_student school%rowtype;

v_result oracle_result%rowtype;

grade varchar2(10);

cursor c1 is select * from SCHOOL;

begin

for v_student in c1

loop

if v_student.class='Working' then

 if v_student.Dev_2000 <50 then

 grade:='FAIL';

 else

 grade:='PASS';

 end if;

elsif v_student.class='Student' then

 if v_student.Dev_2000 >=80 then
```

```
1255
1256    grade:='HONOURS';
1257
1258    elsif v_student.Dev_2000 >=60 then
1259
1260    grade:='A';
1261
1262    elsif v_student.Dev_2000 >=50 then
1263
1264    grade:='B';
1265
1266    elsif v_student.Dev_2000 >=40 then
1267
1268    grade:='C';
1269
1270    else
1271
1272    grade:='B';
1273
1274    end if;
1275
1276   end if;
1277
1278   insert into Dev_2000_result
1279   values(v_student.Roll_no,v_student.Oracle,v_student.Dev_2000,grade);
1280
1281   end loop;
1282
1283   end;
1284
1285   /
1286
1287   ------------------------------------------------anonyms procedure for oracle result----------------
1288   declare
1289
1290   v_student school%rowtype;
1291
1292   v_result oracle_result%rowtype;
1293
1294   grade varchar2(10);
1295
1296   cursor c1 is select * from SCHOOL;
1297
1298   begin
1299
1300   for v_student in c1
1301
1302   loop
1303
```

```
1304 if v_student.class='Working' then
1305
1306  if v_student.Oracle<50 then
1307
1308  grade:='FAIL';
1309
1310  else
1311
1312  grade:='PASS';
1313
1314  end if;
1315
1316 elsif v_student.class='Student' then
1317
1318  if v_student.Oracle>=80 then
1319
1320  grade:='HONOURS';
1321
1322
1323  elsif v_student.Oracle>=60 then
1324
1325  grade:='A';
1326
1327  elsif v_student.Oracle>=50 then
1328
1329  grade:='B';
1330
1331  elsif v_student.Oracle>=40 then
1332
1333  grade:='C';
1334
1335  else
1336
1337  grade:='B';
1338
1339  end if;
1340
1341 end if;
1342
1343 insert into oracle_result values(v_student.Roll_no,v_student.Oracle,v_student.Oracle,grade);
1344
1345 end loop;
1346
1347 end;
1348
1349 /
1350
1351 2. The CUSTOMER table of a state electricity board consists of the following fields:-
1352
```

Meter Number Varchar2 4

Meter Type Character 1

Previous Reading Number 5

Current Reading Number 5

Customer Type Character 1

Last Bill payment Character 1 (values could be Y or N)

There are two types of meters viz. 3- phase or 1-phase coded as T or S respectively. There are
of customers viz. Agricultural Industrial, Commercial and Residential with coeds A , I , C an
respectively.

Formulae:-

Units used = Current Reading   Previous Reading

Rate =Rs.1/ 1.25/ 1.50/ 1.30 for A/I/C/R respectively.

Amount = rate*units used

Surcharge = 5% for single phase

10% for 3 phase

Excise = 30% of (amount +Surcharge)

Net = Amount +Surcharge + Excise

Write a block to calculate the bill for each customer. The program should insert the Meter no., Un
used, Rate, Amount, Surcharge, Excise duty and Net for each customer into some other suitable tabl
Also, at the end, it should insert the total Amount, Surcharge, Excise and Net into some other tab

----------tables----------
create table CUSTOMER

(

```
"Meter Number" Varchar2(4),

"Meter Type" Character(1),

"Previous Reading" Number(5),

"Current Reading" Number(5),

"Customer Type" Character(1),

"Last Bill payment" Character(1) check("Last Bill payment"='Y' OR "Last Bill payment"='N')

)/

//insert dummy data into table customer


Mete M                  Previous Reading                Current                         Reading

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

1000                    S                                                       3000

1001                    T                                                       3000

1002                    S                                                       4000




---------------------------------------------------------------------------------------------------
create table bill

(

"Meter Number" Varchar2(4) primary key,

units number,

rate number,

amount number,

surcharge number,

Excise number,

```

```
Net number

)

/

-----------------------------------------------procedure-----------------------------------------
create or replace procedure calculatebill

as

 v_customer customer%rowtype;

 v_bill bill%rowtype;

 cursor c1 is select * from customer;

 rate number(3,2);

 units number;

 amount number;

 surcharge number;

 Excise number;

 Net number;

begin

 delete from bill;

 for v_customer in c1

 loop

 select decode(v_customer."Customer Type",'A',1,'I',1.25,'C',1.50,'R',1.30) into rate
from dual;

 select decode(v_customer."Meter Type",'T',10,'S',5) into surcharge from dual;

 units:=v_customer."Current Reading"-v_customer."Previous Reading";

 amount:=rate*units;

 surcharge:=surcharge*amount;

 Excise:=(amount +Surcharge)*30/100;
```

```
 Net:= Amount +Surcharge + Excise;

 Insert into bill values(v_customer."Meter Number"
,units,rate,amount,surcharge,Excise,Net);

 end loop;

end;

/

------------------------------------------------output-------------------------------------------------


Compile:



SQL> alter procedure calculatebill compile;



Call:

SQL> exec calculatebill




3. A table consists of the following fields:-




Invoice Number Varchar2 4

Invoice Date Date

Customer Code Number 1

Product Code Number 1

Quantity Sold Number 3



```

```
1549
1550  There are ten customers with codes 0 to 9 and five products with codes 0 to 4. The rates of produc
1551  Rs. 15, 35, 42, 51 and 60 respectively. Write a program to find the total purchase in Rs. of each
1552  customer and total sale of each product using this table and insert these values in two other tabl
1553
1554  -------------------------------------------------tables-----------------------------------------
1555
1556  create table corder
1557
1558  (
1559
1560  "Invoice Number " Varchar2(4),
1561
1562  "Invoice Date " Date,
1563
1564  "Customer Code " Number(1),
1565
1566  "Product Code " Number(1),
1567
1568  "Quantity Sold " Number(3)
1569
1570  )
1571
1572  /
1573
1574  Insert dummy data:-
1575
1576  Invo                    Invoice                          Customer Code          Product Code
1577
1578  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
1579
1580  rj24                    31-MAR-12                     2                                      0
1581
1582  rj24                    31-MAR-12                     2                                      1
1583
1584  rj24                    31-MAR-12                     2                                      2
1585
1586  ----------------------------------------------------------------------------------------------------
1587
1588  create table totalpurchase
1589
1590  (
1591
1592  "Customer Code " Number(1),
1593
1594   Total Purchase  Number
1595
1596  ) /
1597
```

```
1598 ----------------------------------------------------------------------------------------------------
1599
1600 create table totalsale
1601
1602 (
1603
1604 "Product Code " Number(1),
1605
1606  Total Sale  Number
1607
1608 ) /
1609
1610 ------------------------------------------------procedure-----------------------------------------
1611 create or replace procedure find_sale_purchase
1612
1613
1614 as
1615
1616  cursor c1 is select distinct("Customer Code ") from corder;
1617
1618  cursor c2(v_ccode number) is select * from corder where "Customer Code "=v_ccode;
1619
1620
1621
1622  cursor c3 is select distinct("Product Code ") from corder;
1623
1624  cursor c4(v_pcode number) is select * from corder where "Product Code "=v_pcode;
1625
1626
1627
1628  rs number;
1629
1630
1631
1632  TOTAL_PURCHASE number:=0;
1633
1634  TOTAL_SALE number:=0;
1635
1636 begin
1637
1638  delete from totalpurchase;
1639
1640  delete from totalsale;
1641
1642
1643
1644  for i in c1
1645
1646  loop
```

```
 for j in c2(i."Customer Code ")

 loop

 select decode(j."Product Code ",0,15,1,35,2,42,3,51,4,60,0) into rs from dual;

 rs:=rs*j."Quantity Sold ";

 TOTAL_PURCHASE:=TOTAL_PURCHASE+rs;

 end loop;

 insert into totalpurchase values(i."Customer Code ",TOTAL_PURCHASE);

 end loop;



 rs:=0;



 for i in c3

 loop

 for j in c4(i."Product Code ")

 loop

 select decode(j."Product Code ",0,15,1,35,2,42,3,51,4,60,0) into rs from dual;

 rs:=rs*j."Quantity Sold ";

 TOTAL_SALE:=TOTAL_SALE+rs;

 end loop;

 insert into totalsale values(i."Product Code ",TOTAL_SALE);

 end loop;

end;

/

-------------------------------------------------output-----------------------------------------
SQL>exec find_sale_purchase;
```

```
SQL> select * from totalsale;

Product                         Code Total Sale

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

1                                       105

2                                       231

0                                       276



SQL> select * from totalpurchase;


Customer                        Code Total Purchase

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

2                                       276



4. Create a table EMPLOYEE with the following columns:-



Employee No. Varchar2 4

Employee Name Varchar2 30

Designation Varchar2 10

Category Character 1

Basic Salary Number 4



Category may be  J ,  S , or  W  for Jr. officers, Sr. officers or Worker category.
```

Formulae:-

DA = 35% of Basic Salary correct up to paise.

HRA = 15% of Basic Salary subject to a maximum of Rs. 250/1000/30000 for categories W/J/S respectively.

Gross = Basic Salary +DA +HRA

Output the Employee Number and the Gross for each employee in a separate table.

----------tables----------

```
create table EMPLOYEE

(

"Employee No " Varchar2 (4),

"Employee Name" Varchar2 (30),

Designation Varchar2 (10),

Category Character (1),

"Basic Salary " Number (4)

)

/
```

Insert dummy data:-

Empl	Employee Name	DESIGNATIO	C	Basic Salary
1000	rakesh	Sr.off	s	3000
1001	peeyoosh	Sr.off	s	4000
1002	malik	Jr.off	s	5000

```
 create table employee_gross

 create table gross

 (


 "Employee No " Varchar2 (4),

 "Gross Salary " Number (4)

 )

 /

-----------------------------------------------procedure-----------------------------------------
create or replace procedure gross

 as

 cursor c1 is select * from employee;

 emp_record employee%rowtype;

 da number(20,2);

 hra number(20,2);

 gross number(20,2);

 begin

 delete from employee_gross;

 for emp_record in c1

 loop

 da:=emp_record."Basic Salary "*35/100;

 hra:=emp_record."Basic Salary "*15/100;

 if emp_record.Category='j' and hra>250 then

 hra:=250;

 elsif emp_record.Category='s' and hra>1000 then

 hra:=1000;
```

```
 elsif emp_record.Category='w' and hra>30000 then

 hra:=30000;

 else

 hra:=0;

 end if;

 gross:=emp_record."Basic Salary "+da+hra;

 insert into employee_gross values(emp_record."Employee No ",gross);

 end loop;

 end;

/

-----------------------------------------------output-----------------------------------------------

SQL>exec gross;

Empl Gross                      Salary

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

1000                            4050

1001                            5400

1002                            6750




PL*SQL

#Exercise 6



1. The median of an array of numbers is the element m of the array such that half the remaining nu
or equal to m and half are less than or equal to m, if the number of elements in the array is odd.
the median is the average of the two elements m1 and m2 such that half the remaining elements are
and m2, and half the elements are less than or equal to m1 and m2. Write a PL*SQL block that allow
```

elements in a number array and outputs the median of the numbers in the array. Write another PL*SQ
to enter 11 elements in a number array and outputs the median of the numbers in the array. Display
using dbms_output.put_line.

2. The mode of an array of numbers is the number m in the array that is repeated most frequently.
with equal maximum frequencies, there is no mode. Write a PL*SQL block that allows the user to ent
array and outputs the mode or indication that the mode does not exist. Display the above output on
dbms_output.put_line.

3. Write a PL*SQL program to do the following:-

Read a group of 10 temperature readings into two number arrays. A reading consists of two numbers
90, representing the latitude at which the reading was taken, and the observed temperature at that
on screen in tabular format) consisting of each latitude and the average temperature at that latit
readings at a particular latitude, print NO DATA instead of an average. Then print the average t
southern hemispheres (the northern consists of latitudes 1 through 90 and the southern consists of
average temperature should be computed as the average of the averages, not the average of the orig
which hemisphere is warmer. In making the determination, take the average temperatures in all lati
which there are data for both that latitude and the corresponding latitude in the other hemisphere
for latitude 57 but not for latitude 57, then the average temperature for latitude 57 should be i
hemisphere is warmer). Display the above output on the screen using dbms_output.put_line.

4. Write a PL*SQL block to accept a number from the user. With the help of PL*SQL arrays, write a
conversion up to 99 crores. The program should cater to Rs. and paise also.

For example, if the user enters:-

123451250.75

The output of your program should be:-

Rs. Twelve crores, Thirty Four lakhs, Fifty One thousand, Two hundred and Fifty

and Seventy five paise only.

```
1941
1942
1943    If the user enters:-
1944
1945    9728
1946
1947    The output of your program should be:-
1948
1949    Rs. Nine thousand, Seven hundred and Twenty Eight only.
1950
1951
1952
1953  5. Write a PL*SQL block to accept a character string from the user. The user should enter a number
1954  PL*SQL arrays, write a program for Word to number conversion up to 99 crores. The program should c
1955
1956
1957
1958  For example, if the user enters:-
1959
1960  Rs. Twelve crores, Thirty Four lakhs, Fifty One thousand, Two hundred and Fifty
1961
1962  and Seventy five paise only.
1963
1964    The output of your program should be:-
1965
1966    123451250.75
1967
1968
1969
1970
1971    If the user enters:-
1972
1973    Rs. Nine thousand, Seven hundred and Twenty Eight only.
1974
1975    The output of your program should be:-
1976
1977    9728
1978  \
1979
1980
1981
1982
1983
1984  PL*SQL
1985
1986  #Exercise 7
1987
1988
1989
```

```
1. Write a PL*SQL block that prompts the user to enter the salary of an employee. Your program sho
display the name of the employee (from the EMP table) who s getting that salary. If more than 1
employee is receiving that salary, or if no employees exist getting that salary, your program shou
display appropriate messages. Use too_many_rows and no_data_found exceptions to achieve this.
Display the results on the screen using dbms_output.put_line.


 ----------------------------------------------TABLE-----------------------------------------------

create table emp

(

enpno varchar(4),

empname varchar(30),

designation varchar2(10),

category char(1),

basicsalary number(4),

joined date

)

/

-----------------------------------------------EXCEPTION CODE---------------------------------------

DECLARE

mysal number;

sal NUMBER:=&salary;

BEGIN

select basicsalary into mysal from emp where basicsalary =sal;

exception

when too_many_rows then

dbms_output.put_line('too many data found in result can''t handel ');

when no_data_found then

```

```
dbms_output.put_line('no data found for your query');

END;

/



2. Write a PL*SQL block to check if any employee from EMP table is receiving a salary greater than
9999.99. Make the use of value_error exception to achieve this. Display the results on the screen
dbms_output.put_line.


 ----------------------------------------------EXCEPTION CODE-------------------------------------

DECLARE

mysal number;

sal NUMBER:=&salary;

BEGIN

insert into emp values('1003','kumar','sr.off','s',sal,sysdate);

exception

when value_error then

dbms_output.put_line('salary is limited to 4 digits only');

when others then

dbms_output.put_line('un identified error occured');


END;

/




3. Create a user-defined exception by the name of exp_check. Select the ename and hiredate of all
employees into a cursor. Your program should calculate the experience of all the employees in year
and insert the ename and experience of each employee into tempp table. If any employee has experie
less than 2 years, the program should be aborted with a suitable message. Raise the user-defined
exception exp_check to achieve this. Display the results on the screen using dbms_output.put_line.
```

```
2088
2089
2090  ----------------------------------------------EXCEPTION CODE-----------------------------------------
2091
2092  declare
2093
2094  exp_check exception;
2095
2096  cursor c1 is select empname,joined from emp;
2097
2098  BEGIN
2099
2100  for i in c1
2101
2102  loop
2103
2104   if(trunc(months_between(sysdate,i.joined)/12)<2) then
2105
2106   raise exp_check;
2107
2108   else
2109
2110   insert into tempp2 values(i.empname,trunc(months_between(sysdate,i.joined)/12));
2111
2112   end if;
2113
2114  end loop;
2115
2116  exception
2117
2118  when exp_check then
2119
2120  dbms_output.put_line('experiance is less then 2 years not allowed');
2121
2122  when others then
2123
2124  dbms_output.put_line('un identified error occured');
2125
2126  END;
2127
2128  /
2129
2130  4. Write a PL*SQL function to take three parameters, the sides of a triangle. The sides of the tri
2131  should be accepted from the user. The function should return a Boolean value:- true if the triangl
2132  valid, false otherwise. A triangle is valid if the length of each side is less than the sum of the
2133  the other two sides. Check if the dimensions entered by the user can form a valid triangle. Displa
2134  results on the screen using dbms_output.put_line.
2135
2136
```

```
----------------------------------------------EXCEPTION CODE--------------------------------------

create or replace function triangle(a number,b number,c number)

return boolean

as

invalid_triangle exception;

begin

 if not (a+b>=c and b+c>=a and c+a>=b) then

 raise invalid_triangle;

 else

 return true;

 end if;

exception

when invalid_triangle then

dbms_output.put_line('xxxxxx- invalid triangle -xxxxxxxxxx');

return false;


when others then

dbms_output.put_line('un identified error occured');

END;

/



----------------------------------------------------------------CALLING-CODE----------------------

declare

a number:=&side1;

b number:=&side2;

```

```
c number:=&side3;

x boolean;

begin

 x:=triangle(a,b,c);

end;



SQL> /

Enter value for side1: 2

old 2: a number:=&side1;

new 2: a number:=2;

Enter value for side2: 3

old 3: b number:=&side2;

new 3: b number:=3;

Enter value for side3: 5

old 4: c number:=&side3;

new 4: c number:=5;



5. Write a function that generates a random number between 1 and 10. Use any logic of your choice
achieve this. Display the results on the screen using dbms_output.put_line.


-------RANDOM METHOD--------

declare

x number;

begin

select trunc(dbms_random.value(1,10)) into x from dual;

dbms_output.put_line(x);
```

```
2235
2236  end;
2237
2238  /
2239
2240  --------CREATING-RANDOM -NO--------
2241
2242  declare
2243
2244  seed number:=&number;
2245
2246  begin
2247
2248  IF seed=0 THEN
2249
2250   seed := EXP(TO_NUMBER(TO_CHAR(SYSDATE,'ss'))/59);
2251
2252  END IF;
2253
2254  seed := 1/(seed - TRUNC(seed));
2255
2256  seed := seed - TRUNC(seed);
2257
2258  dbms_output.put_line(seed);
2259
2260  end;
2261
2262
2263  /
2264
2265  ------------CALLING-CODE------
2266
2267  declare
2268
2269  seed number;
2270
2271  n number:=&number;
2272
2273  begin
2274
2275   seed := EXP(TO_NUMBER(TO_CHAR(SYSDATE,'ss'))/59)-1;
2276
2277   seed := 1/(seed - TRUNC(seed));
2278
2279  seed := seed - TRUNC(seed);
2280
2281  n:=trunc(n/10)-1;
2282
2283  dbms_output.put_line(trunc(seed,n)*power(10,n));
```

```
2284
2285  end;
2286
2287  /
2288
2289
2290
2291  6. Design a structure to store length in yards, feet, and inches (for example, 7 yards, 2 feet, 3
2292  program should accept 2 length measurements from the user. Write a PL*SQL procedure to find the
2293  difference between two measurements as represented by these structures. Display the results on the
2294  screen using dbms_output.put_line.
2295
2296
2297  ----------------------------------------------ANONYMS-PROGRAM-------------------------------------
2298
2299  declare
2300
2301   y number:=&yard;
2302
2303   f number:=&feet;
2304
2305   i number:=&inch;
2306
2307   y2 number:=&yard2;
2308
2309   f2 number:=&feet2;
2310
2311   i2 number:=&inch2;
2312
2313  begin
2314
2315   i:=y*3*12+f*12+i;
2316
2317   i2:=y2*3*12+f2*12+i2;
2318
2319   i:=i-i2;
2320
2321   y:=trunc(i/(3*12));
2322
2323   i:=mod(i,(3*12));
2324
2325   f:=trunc(i/(12));
2326
2327   i:=mod(i,12);
2328
2329   dbms_output.put_line('yard: '||y||' foot: '||f||' inch: '||i);
2330
2331  end;
2332
```

```
/

7. Create a function that accepts a string of n characters and exchanges the first character with
second with the next   to   last, and so forth until n exchanges have been made. What will the fin
string look like? Write the function to verify your conclusion. Display the results on the screen
dbms_output.put_line.


----------------------------------------------FUNCTION-----------------------------------------------

create or replace function revstr(st in out varchar2)

 return boolean

 as

 len number:=length(st);

 begin

 for j in 1..len


 loop

 st:=st||substr(st,len-j,1);

 end loop;

 st:=substr(st,len);

 len:=length(st);

 st:=substr(st,1,len-1);

 dbms_output.put_line('reverse string is '||st);

 return true;

 end;

/




----------------------------------------------CALLING-CODE------------------------------------------

```

```
declare

k boolean;

val varchar2(50):='rakesh kumar';

begin

k:=revstr(val);

end;

/









PL*SQL

#Exercise 8



1. Write a stored procedure by the name of Comp_intr to calculate the amount of interest on a bank
account that compounds interest yearly. The formula is:-

 I = p (1+ r/100) y   p

where:-

 I is the total interest earned.

 p is the principal.

 r is the rate of interest as a decimal less than 1, and

 y is the number of years the money is earning interest.



Your stored procedure should accept the values of p, r and y as parameters and insert the Interest
Total amount into tempp table.

-----------------------------------------------------PROCEDURE-------------------------------------

CREATE OR REPLACE PROCEDURE COMP_INTR (P IN NUMBER,R IN NUMBER,Y IN
```

```
NUMBER)

AS

I NUMBER(6,2);

BEGIN

I:=P*POWER(1+R,Y)-P;

DBMS_OUTPUT.PUT_LINE(I);

END;

/

--------------------------------------------------CALLING -PROCEDURE-----------------------------

SQL> begin

 2 comp_intr(1600,9,4);

 3 end;

 4 /

658.530576



Or



SQL> exec comp_intr(1600,9,4);



2. Create a stored function by the name of Age_calc. Your stored function should accept the date o
of a person as a parameter. The stored function should calculate the age of the person in years, m
and days e.g. 35 years, 3 months, 17 days. The stored function should return the age in years dire
(with the help of Return statement). The months and days are to be returned indirectly in the form
OUT parameters. Write a PL*SQL block to accept the date of birth of an employee from the user, cal
the stored function, and display the age of the employee on the screen. Display the above results
screen using dbms_output.put_line.


--------------------------------------------------FUNCTION--------------------------------------

```

```
2480
2481
2482    create or replace function age_calc(dat in date,d out number,m out number)
2483
2484    return number as
2485
2486    y number;
2487
2488
2489    begin
2490
2491    d:=sysdate-dat;
2492
2493    y:=d/365;
2494
2495    y:=trunc(y);
2496
2497    m:=(d-y*365)/30;
2498
2499    M:=trunc(m);
2500
2501    d:=trunc(d-y*365-m*30);
2502
2503    return y;
2504
2505    end;
2506
2507   /
2508
2509   ----------------------------------------------------CALLING -PROCEDURE-----------------------------
2510
2511   DECLARE
2512
2513   D varchar2(20):='r';
2514
2515   P1 NUMBER:=&day_of_birts;
2516
2517   P2 NUMBER:=&month_of_birth;
2518
2519   P3 NUMBER:=&year_of_birth;
2520
2521   BEGIN
2522
2523   D:=to_char(p1)||'-'||to_char(p2)||'-'||to_char(p3);
2524
2525   P1:=AGE_CALC(to_date(d,'dd-mm-yyyy'),P2,P3);
2526
2527   DBMS_OUTPUT.PUT_LINE('DAYS: '||P2||' MONTHS: '||P3||' YEARS: '||P1);
2528
```

```
END;

/



3. Create a package by the name of Payroll_calc. The package should contain separate procedures fo
HRA, Gross, Tax and Net calculation.


Formulae:-

DA = 10% of Sal for Managers and 5% of Sal for others.

HRA = 20% of Sal for employees of department 10 and 7% of Sal for employees of other departments.

Gross = Sal + DA + HRA.

If Gross exceeds 4000, Tax is to be deducted at 5% of the amount exceeding 4000. If Gross exceeds
5000, Tax is to be deducted at 5% of the amount exceeding 4000 and an additional of 2% of the amou
exceeding 5000.

Net = Gross   Tax.

Write a PL*SQL block that calls the procedures from the above package. The PL*SQL block should
print the Pay slips for all the employees. The format of the Pay slip should be as follows:-



 ****************************************************************

 Name:- KING Designation:- PRESIDENT Dept:- ACCOUNTING

 Sal:- xxx DA:- xxx HRA:- xxx Gross:- xxx Tax:- xxx Net:- xxx

 ****************************************************************




PL*SQL

#Exercise 9



Create the following 3 tables and insert sample data as shown:-

```

```
 Ord_mst

Ord_no                  Cust_cd                 Status

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

1                               C1                              P









Ord_dtl

Ord_no                  Prod_cd                 Qty

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

1                               P1                              100

1                               P2                              200









Prod_mst

Prod_cd                 Prod_name                       Qty_in_stock                    Booked_qty

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

P1                              Floppies                        10000

P2                              Printers                        5000

P3                              Modems                          3000



1. Write a Before Insert trigger on Ord_dtl. Anytime a row is inserted in Ord_dtl, the Booked_qty
should be increased accordingly.


-----------------------------------------TABLES-------------------------------------------------

 create table Ord_mst

 (
```

```
2627
2628    Ord_no number,
2629
2630    Cust_cd varchar(2),
2631
2632    Status varchar(1)
2633
2634    )
2635
2636    /
2637
2638    create table Ord_dtl
2639
2640    (
2641
2642    Ord_no number,
2643
2644    Prod_cd varchar(2),
2645
2646    Qty number(3)
2647
2648    )
2649
2650    /
2651
2652    create table Prod_mst
2653
2654    (
2655
2656    Prod_cd varchar(2),
2657
2658    Prod_name varchar(20),
2659
2660    Qty_in_stock number,
2661
2662    Booked_qty number
2663
2664
2665    )
2666
2667    /
2668
2669
2670
2671
2672
2673
2674
2675   ----------------------------------------------TRIGGER----------------------------------------------
```

```
 create or replace trigger Ord_dtl_1 before insert on Ord_dtl for each row

 begin

 update Prod_mst set Booked_qty=Booked_qty-:new.Qty where Prod_cd=:new.Prod_cd;

 dbms_output.put_line();

 end;

 /

-----------------------------------------------------OUTPUT---------------------------------------

 SQL> insert into Ord_dtl values(1,'P1',50);

 SQL> select * from ord_dtl;


ORD_NO          PR                      QTY

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

1                       P1                      100

1                       P2                      200

1                       P1                      50

 SQL> select * from prod_mst;


PR              PROD_NAME                       QTY_IN_STOCK                    BOOKED_QTY

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

P1              Floppies                        10000                                   1050

P2              Printers                        5000                                    600

P3              Modems                          3000                                    200


2. Write a Before Delete trigger on Ord_dtl. Anytime a row is deleted from Ord_dtl, the Booked_qty
Prod_mst should be decreased accordingly.
```

```
-------------------------------------------TRIGGER---------------------------------------------

 CREATE or replace TRIGGER ORD_DTL_2 BEFORE DELETE ON ORD_DTL

 FOR EACH ROW

 BEGIN

 update Prod_mst set Booked_qty=Booked_qty-:old.Qty where Prod_cd=:old.Prod_cd;

 dbms_output.put_line('data deleted and updated in prod_mst table');

 END;

--------------------------------------------------OUTPUT---------------------------------------

 SQL> delete from ord_dtl where qty=50;



PR                              PROD_NAME              QTY_IN_STOCK    BOOKED_QTY

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

P1                              Floppies               10000                  1000

P2                              Printers               5000                   600

P3                              Modems                 3000                   200
```

3. Write a Before Update of Prod_cd, Qty trigger on Ord_dtl. Anytime the Prod_cd or Qty is updated
Booked_qty in Prod_mst should be increased/decreased accordingly.

```
---------------------------------------------------TRIGGER-------------------------------------

create trigger Prod_cd_3 before update on ord_dtl

for each row

begin

update prod_mst set booked_qty=booked_qty-:old.qty+:new.qty, prod_cd=:new.prod_cd
```

```
where prod_cd=:old.prod_cd;

end;

-----------------------------------------------------FIRST UPDATE---------------------------------

SQL> select * from ord_dtl;



ORD_NO                  PR                              QTY

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

1                               P1                              100

1                               P2                              200



SQL> select * from prod_mst;



PR                      PROD_NAME                       QTY_IN_STOCK            BOOKED_QTY

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

P1                      Floppies                        10000                           1000

P2                      Printers                        5000                            600

P3                      Modems                          3000                            200



SQL> update ord_dtl set qty=300 where prod_cd='P1';



1 row updated.



SQL> select * from ord_dtl;



ORD_NO                  PR                                      QTY
```

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

1                               P1                                      300

1                               P2                                      200


SQL> select * from prod_mst;


PR                      PROD_NAME                       QTY_IN_STOCK                    BOOKED_QTY

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

P1                      Floppies                        10000                                   12

P2                      Printers                        5000                                    60

P3                      Modems                          3000                                    20


----------------------------------------SECOND UPDATE-----------------------------------------

SQL> update ord_dtl set prod_cd='P3',qty=300 where prod_cd='P1';


1 row updated.


SQL> select * from prod_mst;


PR                      PROD_NAME                       QTY_IN_STOCK            BOOKED_QTY

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

P1                      Floppies                        10000                           1200

P2                      Printers                        5000                            600

P3                      Modems                          3000                            200
```

4. Write a Before Update of Status trigger on Ord_mst. If the Status is updated from P (Pending) t
(Delivered), the Booked_qty and Qty_in_stock from Prod_mst should be decreased accordingly. If the
Status is updated from P (Pending) to C (Cancelled), the details of the order should be deleted fr
and corresponding Booked_qty from Prod_mst should be decreased accordingly. (The Before delete tri
on Ord_dtl would automatically decrease the Booked_qty from Prod_mst).

```
----------------------------------------------------TRIGGER----------------------------------------

CREATE OR REPLACE TRIGGER ORD_MST_1 BEFORE UPDATE ON ORD_MST

FOR EACH ROW

Declare --DECLARE IS NECESSARY

cursor c1 is select * from Ord_dtl where Ord_no=:old.Ord_no;

BEGIN

 IF :NEW.STATUS='c' or :NEW.STATUS='C' THEN

 DELETE FROM ORD_DTL WHERE ORD_NO=:OLD.ORD_NO;

 END IF;

 IF (:NEW.STATUS='d') or (:NEW.STATUS='D') THEN

 for i in c1

 loop

 UPDATE PROD_MST SET QTY_IN_STOCK=(QTY_IN_STOCK-
i.QTY),booked_QTY=(booked_QTY-i.QTY) where Prod_cd=i.Prod_cd;

dbms_output.put_line('updated as cancel in prod_mst');

 end loop;

 END IF;

END;
```

```
/

-----------------------------------------------------OUTPUT-----------------------------------------

SQL> select * from ord_dtl;

 ORD_NO PR QTY

---------- -- ----------

 1 P1 100

 1 P2 200

SQL> select * from ord_mst;



ORD_NO          CU              S

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

1                       C1              P



SQL> select * from prod_mst;

PR              PROD_NAME               QTY_IN_STOCK            BOOKED_QTY

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

P1              Floppies                10000                           1000

P2              Printers                5000                            600

P3              Modems                  3000                            200



SQL> update ord_mst set status='c';

data deleted and updated in prod_mst table //shows two rows deleted of ord_mst

data deleted and updated in prod_mst table



and if
```

```
2970
2971  SQL> update ord_mst set status='d';
2972
2973  updated as cancel in prod_mst
```