☐ rksuthar19 / dbbook Public

Aয় 1 contributor

```
2973 lines (1379 sloc) | 51 KB
       # Chapter 2
  1
  2
  3
      -----PL*SQL Assignments-----
  4
  5
      #Exercise 1
  6
  7
      1. Write a program that computes the perimeter and the area of a rectangle. Define your own values
  8
  9
       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.
 10
 11
 12
 13
       Declare
 14
 15
       Length number:=&length;
 16
 17
      Width number:=&width;
 18
 19
      Area number;
 20
 21
      Parimeter number;
 22
      Begin
 23
 24
 25
      Area:=length*width;
 26
 27
      Parimeter:= 2*(length+width);
 28
 29
       dbms_output.put_line('AREA OF RECTANGLE IS:'||Area);
```

```
30
31
     dbms_output.put_line('PARIMETER OF RECTANGLE IS:'||PARIMETER);
32
33
     End;
34
35
     /
36
     2. Write a program that declares an integer variable called num, assigns a value to it, and comput
37
38
     inserts into the tempp table the value of the variable itself, its square, and its cube.
39
40
41
42
43
     SQL> create table tempp
44
45
      (
46
47
      item number,
48
49
      square number,
50
51
      cube number
52
53
      );
54
55
56
57
     Table created.
58
59
60
61
     SQL> DECLARE
62
63
      num number:=#
64
65
      begin
66
67
      insert into tempp values(num, num*num, num*num*num);
68
69
      end;
70
71
     SQL> /
72
     Enter value for num: 5
73
74
75
     old 2: num number:=#
76
77
     new 2: num number:=5;
```

```
79
 80
 81
      PL/SQL procedure successfully completed.
 82
 83
 84
 85
      SQL> select * from tempp;
 86
 87
 88
 89
      ITEM
                      SQUARE
                                      CUBE
 90
 91
      <<<<<<<<<<<<
 92
 93
      5
                              25
                                                      125
 94
 95
 96
 97
      3. Convert a temperature in Fahrenheit (F) to its equivalent in Celsius (C) and vice versa. The re
 98
      formulae are:-
99
100
101
      C = (F-32)*5/9
102
103
      F = 9/5*C + 32
104
105
      Display the output on the screen using dbms_output.put_line. Data has to be input by the user.
106
107
108
109
      Declare
110
111
       F number:=&Fahrenheit;
112
113
       C number:=&Celsius;
114
115
       result number;
116
117
      Begin
118
119
      result:=(F-32)*5/9;
120
121
      dbms_output.put_line('fahrenheit value entered '||F||' equal to celsius :'||' '||result);
122
123
      result:=9/5*C+32;
124
125
      dbms_output.put_line('fahrenheit value entered '||F||' equal to celsius :'||' '||result);
126
127
      end;
```

```
128
129
      /
130
131
132
133
      4. Convert a number of inches into yards, feet, and inches. For example, 124 inches equals 3 yards
134
      and 4 inches. Display the output on the screen using dbms_output.put_line. Data has to be input by
135
      user.
136
137
138
139
140
      declare
141
142
      inch int:=&howmanyinch;
143
144
      yard int;
145
146
      foot int;
147
148
      begin
149
150
      foot:=inch/12;
151
152
      yard:=foot / 3;
153
154
      foot:=foot mod 3;
155
156
      inch:=inch mod 12;
157
      dbms_output.put_line(yard||' '||foot||' '||inch);
158
159
160
      end;
161
162
163
164
165
166
      5. Write a program that enables a user to input an integer. The program should then state whether
167
      integer is evenly divisible by 5. (Use decode instead of IF statement where required). Display the
168
      output on the screen using dbms output.put line. Data has to be input by the user.
169
170
171
172
173
      declare
174
175
      num number:=&enter value;
176
```

```
177
      begin
178
179
      if (num mod 5=0) then
180
181
      dbms_output.put_line('GIVEN NO. IS DIVISIBLE BY 5');
182
183
      else
184
185
      dbms_output.put_line('not divisible by 5');
186
187
      end if;
188
189
190
      end;
191
192
193
194
195
196
197
198
      begin
199
200
      select decode(num mod 5, 0, 'divisible', 'nod devisible') into result from dual;
201
202
      dbms output.put line(result);
203
204
      end;
205
206
207
      6. Your block should read in two real numbers and tell whether the product of the two numbers is e
208
209
      or greater than 100. Display the output on the screen using dbms_output.put_line. (Use decode inst
210
      of IF statement where required). Data has to be input by the user.
211
212
213
214
215
      declare
216
217
      a number:=&firstno;
218
219
      b number:=&secondno;
220
221
      result varchar2(50);
222
223
      begin
224
225
      select decode(trunc(a*b/100),0,'LESS\ THEN\ 100','GREATER\ THEN\ OR\ EQUAL\ TO\ 100') into result
```

```
from dual;
226
227
228
      dbms output.put line(result);
229
230
      end;
231
232
233
234
235
236
     PL*SQL
237
238
      #Exercise 2
239
240
241
242
      1. In a PL*SQL block, create a datatype by the name of addr_type. It should contain the following
243
      components:-
244
245
246
      name varchar2 (20)
247
248
      street varchar2 (30)
249
250
      city varchar2 (20)
251
252
      state varchar2 (15)
253
254
      Your block should accept the names and addresses of 4 employees in 4 different variables of dataty
255
      addr type. Output the names and addresses of the 4 employees on the screen in the form of Labels a
256
      shown below:-
257
      ********************
258
259
260
      * Name:- Jack ** Name:- Scott *
261
262
      * Street:- M.G. Road ** Street:- Bhosale Marg *
263
264
      * City:- Mumbai ** City:- Chennai *
265
266
      * State:- Maharashtra ** State:- Tamil Nadu *
267
268
269
      *******************
270
271
272
      * Name:- King ** Name:- Adams *
273
274
      * Street:- Lane No:-2 ** Street:- P. M. Road *
```

```
275
276
     * City:- Nagpur ** City:- Bangalore *
277
278
     * State:- Maharashtra ** State:- Karnataka *
279
280
281
282
283
284
     SQL> create type addr_type as object
285
286
      (
287
288
      name varchar2(20), street varchar2(30), city varchar2(20), state varchar2(5
289
290
      0));
291
292
293
294
295
296
297
298
     declare
299
      temp1 addr_type := addr_type('jack','mg road','mumbai','maharashtra');
300
301
302
      temp2 addr_type := addr_type('scott','bhosale marg','chennai','tamil nadu');
303
      temp3 addr_type := addr_type('king','lan no:-2','nagpur','maharashtra');
304
305
306
      temp4 addr_type := addr_type('adams','pm road','bangalore','karnataka');
307
308
     begin
309
      310
311
312
      dbms_output.put_line('* Name:- '||temp1.name||' ** Name:- '||temp2.name||' *');
313
314
      dbms_output.put_line('* Street:- '||temp1.street||' ** Street:- '||temp2.street||'*');
315
316
      dbms_output.put_line('* Street:- '||temp1.city||' ** Street:- '||temp2.city||' *');
317
318
      dbms_output.put_line('* Street:- '||temp1.state||' ** Street:- '||temp2.state||' *');
319
320
      321
322
      323
```

```
324
325
      dbms_output.put_line('* Name:- '||temp3.name||' ** Name:- '||temp4.name||' *');
326
      dbms output.put line('* Street:- '||temp3.street||' ** Street:- '||temp4.street||' *');
327
328
329
      dbms_output.put_line('* Street:- '||temp3.city||' ** Street:- '||temp4.city||' *');
330
      dbms output.put line('* Street:- '||temp4.state||' ** Street:- '||temp4.state||' *');
331
332
      333
334
335
      ---dbms_output.put_line(temp1.street);
336
337
      end;
338
339
340
341
342
343
344
     PL*SQL
345
346
     #Exercise 3
347
348
349
350
     1. Input a number and determine whether it is within a given range (for example, between 1 and 10)
351
     low and high values of the range may be input by the user rather than be fixed by the program. Dis
352
      the output on the screen using dbms output.put line.
353
354
      ------ANONYMS-PROGRAM-----------
355
356
357
     declare
358
359
      lb number:=&lower_bound;
360
361
      ub number:=&upper bound;
362
363
      data number:=#
364
365
     begin
366
367
      if data>lb then
368
369
      if data<ub then
370
371
      dbms output.put line('number is between'||lb||' and '||ub);
372
```

```
373
       end if;
374
375
       else
376
       dbms_output.put_line('number is not between'||lb||' and '||ub);
377
378
379
       end if;
380
381
      end;
382
383
384
385
      2. Input three positive integers representing the sides of a triangle, and determine whether they
386
      triangle. Hint: In a triangle, the sum of any two sides must always be greater than the third side
387
      the output on the screen using dbms_output.put_line.
388
389
390
      ------ANONYMS-PROGRAM-----------
391
392
      declare
393
394
       a number:=&first side;
395
396
       b number:=&second_side;
397
398
       c number:=&third side;
399
400
      begin
401
402
       if a+b>c then
403
       if b+c>a then
404
405
406
       if c+a>b then
407
408
       dbms_output.put_line('valid triangle');
409
410
       end if;
411
412
       end if;
413
414
       else
415
416
       dbms output.put line('invalid triangle');
417
418
       end if;
419
420
      end;
421
```

```
422
423
424
425
426
427
428
429
430
      3. Check if a given a year is a leap year. The condition is:-
431
432
433
      year should be (divisible by 4 and not divisible by 100) or (divisible by 4 and divisible by 400.)
434
      output on the screen using dbms_output.put_line. The year should be input by the user.
435
436
437
438
      ------ANONYMS-PROGRAM------
439
440
      declare
441
442
      year int:=&enter_yera;
443
444
       result1 int;
445
446
       result2 int;
447
448
      result3 int;
449
      begin
450
451
452
       result1:=mod(year,100);
453
454
       result2:=mod(year,400);
455
456
       result3:=mod(year,4);
457
458
       if (result3=0) and not(result1=0) or (result3=0) and (result2=0) then
459
460
       dbms_output.put_line('leap year');
461
462
       else
463
464
       dbms_output.put_line('not a leap year');
465
466
       end if;
467
468
      end;
469
470
```

```
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
     -----ANONYMS-PROGRAM------
480
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
     end;
502
503
504
505
     5. Program should accept the age of the user. Depending upon the following conditions it should ou
506
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
```

```
520
       age number:=&user_age;
521
522
       user varchar(20);
523
524
      begin
525
526
      if (age<18) then
527
528
       user:='child';
529
530
531
       elsif (age>18) and (age<21) then
532
533
       user:='major';
534
535
       else
536
537
       user:='adult';
538
539
       end if;
540
541
       dbms output.put line('user is '||user);
542
543
      end;
544
545
      /
546
547
      6. Write a program that asks the user to input two character strings. Your program should then det
548
      character string exists inside another character string. Display the above on the screen using
549
      dbms output.put line.
550
551
       -----ANONYMS-PROGRAM------
552
553
      declare
554
555
       str1 varchar(30):='&first_string';
556
557
      str2 varchar(30):='&second_string';
558
559
      begin
560
561
       if instr(str1,str2)>0 or instr(str2,str1)>0 then
562
563
       dbms output.put line('one string contained in another');
564
565
       else
566
567
       dbms output.put line('both are different string');
568
```

```
end if;
569
570
571
     end;
572
573
     /
574
575
     7. Suppose the grade obtained by a student depends upon his scores and the grading rule is as foll
576
577
     Scores Grades
578
579
      95-100 A
580
581
      85-94 B
582
583
      70-84 C
584
585
      60-69 D
586
587
      0-59 E
588
589
     Write a block to accept a student s marks and accordingly output his grade. Display the output on
590
      screen using dbms_output.put_line.
591
592
      ------ANONYMS-PROGRAM------
593
594
     declare
595
596
      score number:='&score';
597
      grade varchar(1);
598
599
600
     begin
601
602
      if score<=59 then
603
604
      grade:='E';
605
606
      elsif score<=69 then
607
608
      grade:='D';
609
610
      elsif score<=84 then
611
612
      grade:='C';
613
614
      elsif score<=94 then
615
616
617
      grade:='B';
```

```
618
619
       elsif score<=100 then
620
       grade:='A';
621
622
623
       end if;
624
625
       dbms_output.put_line('your marks are'||score||' your grade is --->'||grade);
626
627
      end; /
628
629
      8. A company manufactures three products:- computer stationery, fixed disks and computers. The fol
630
      codes are used to indicate them:-
631
632
      Product Code
633
634
      Computer Stationery 1
635
636
      Fixed Disks 2
637
638
      Computers 3
639
640
      The company has a discount policy as follows:-
641
642
      Product Order amount Discount rate
643
644
      Computer stationery Rs. 5000 or more 12%
645
646
      Computer stationery Rs. 3000 or more 8%
647
648
      Computer stationery Below Rs. 3000 2%
649
      Fixed disks Rs. 20000 or more 10%
650
651
652
      Fixed disks Rs. 15000 or more 5%
653
654
      Computers Rs. 50000 or more 10%
655
656
      Computers Rs. 25000 or more 5%
657
658
      Write a program to accept the order details i.e. product code and order amounts for the products,
659
      the discount amounts as per this policy and output the net order amount. Display the output on the
      using dbms_output.put_line.
660
661
      662
663
664
      declare
665
       code number:='&item_code';--if it is a procedure item_code will be permanently saved in user_sour
666
```

```
667
668
       amt number:='&item_amount';
669
670
       disc number;
671
672
       prod varchar2(30);
673
674
      begin
675
676
       if code=1 then
677
678
       prod:='Computer Stationary';
679
680
       if amt>=5000 then
681
682
       disc:=.12;
683
684
       elsif amt>=3000 then
685
686
       disc:=.08;
687
688
       else
689
690
       disc:=.02;
691
692
       end if;
693
694
       elsif code=2 then
695
696
       prod:='Fixed Disks';
697
698
       if amt>=20000 then
699
700
       disc:=.12;
701
702
       elsif amt>=15000 then
703
704
       disc:=.08;
705
706
       end if;
707
708
709
       elsif code=3 then
710
711
       prod:='Computers';
712
713
       if amt>=50000 then
714
715
       disc:=.12;
```

```
716
717
       elsif amt>=25000 then
718
719
       disc:=.08;
720
721
       end if;
722
723
       end if;
724
       dbms_output.put_line('_____');
725
726
       dbms_output.put_line('PRODUCT '||prod);
727
728
729
       dbms_output.put_line('ORDER AMOUNT '||amt);
730
731
       dbms_output.put_line('DISCOUNT IS '||disc);
732
733
       dbms output.put line('AFTER DISCOUNT '||((1-disc)*amt));
734
735
       dbms_output.put_line('_____');
736
737
      end;
738
739
      /
740
741
742
743
744
      PL*SQL
745
746
      #Exercise 4
747
748
749
750
      1. Write a program containing a loop that iterates from 1 to 1000 using a variable I, which is
751
      incremented each time around the loop. The program should output the value of I every hundred
752
      iterations (i.e., the output should be 100, 200, etc). Display the output on the screen using
753
      dbms_output.put_line.
754
755
756
757
758
      declare
759
760
      i int:=1;
761
762
      begin
763
764
       loop
```

```
765
766
       dbms_output.put_line(i);
767
       dbms_output.put_line('____');
768
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
      the cubes of the digits equal the number itself. Display the output on the screen using
783
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
```

```
814
       dbms_output.put_line('____');
815
816
       end if;
817
818
       exit when i=1000;
819
820
       i:=i+1;
821
822
       end loop;
823
824
      end;
825
826
827
828
      3. Write a PL*SQL block that reads in a minimum and maximum value for a radius, along with an
829
      increment factor, and generates a series of radii by repeatedly adding the increment to the minimu
830
      until the maximum is reached. For each value of the radius, compute and display the circumference,
831
      area, and volume of the sphere. (Be sure to include both the maximum and the minimum values.).
832
      Validate each of the input values to be sure they are positive. If the minimum is typed in place o
833
      maximum, swap the values within the program, and continue execution. Display the results on the
834
      screen using dbms_output.put_line.
835
836
837
       declare
838
839
       num int;
840
841
       j int:=1;
842
843
       x int;
844
845
       y int;
846
847
       z int;
848
       begin
849
850
851
       loop
852
853
       num:=j;
854
855
       x:=mod(num,10);
856
857
       num:=trunc(num/10);
858
859
       y:=mod(num,10);
860
861
       num:=trunc(num/10);
862
```

```
863
       z:=mod(num,10);
864
865
       if (x*x*x+y*y*y+z*z*z)=j then
866
867
       dbms_output.put_line(j);
868
869
870
       dbms_output.put_line('____');
871
872
       end if;
873
874
       exit when j=999;
875
876
       j:=j+1;
877
878
       end loop;
879
880
      end;
881
882
883
884
      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.
885
886
887
       declare
888
889
       num int:='&number';
890
891
       counter int:=1;
892
893
       begin
894
895
       loop
896
897
       counter:=counter+1;
898
899
       num:=2*num;
900
901
       exit when num>1000000;
902
903
       end loop;
904
905
       dbms_output.put_line('number'||num||' needs '||counter||' times
906
      multiplication to reach till 1 million');
907
908
       end;
909
910
       /
911
```

```
912
      4. A palindrome is a word that is spelled the same forward and backward, such as level, radar, etc
913
      program to read in a five letter word from the user and determine whether it is a palindrome. Disp
914
      the results on the screen using dbms_output.put_line.
915
916
917
918
919
      declare
920
       str varchar2(50):='&string';
921
922
923
       counter int:=length(str);
924
925
      begin
926
927
       dbms_output.put_line(counter);
928
929
       loop
930
931
       exit when counter=0;
932
933
       exit when not(substr(str,counter,1)=substr(str,((length(str)+1)-counter),1));
934
935
       counter:=counter-1;
936
937
       end loop;
938
939
       if counter=0 then
940
941
       dbms output.put line(str||'is palindrom');
942
       else
943
944
945
       dbms_output.put_line(str||'is not palindrom');
946
947
       end if;
948
949
      end;
950
951
      /
952
953
954
955
      5. Modify the above program to accept a variable length word. This requires determining how many
956
      characters are read in.
957
958
959
       declare
960
```

```
961
        str varchar2(50):='&string';
962
963
        counter int:=length(str);
964
965
966
       begin
967
968
        dbms output.put line(counter);
969
970
        loop
971
972
        exit when counter=0;
973
974
        exit when not(substr(str,counter,1)=substr(str,((length(str)+1)-counter),1));
975
976
        counter:=counter-1;
977
978
        end loop;
979
980
        if counter=0 then
981
982
        dbms output.put line(str||'is palindrom');
983
984
        else
985
986
        dbms output.put line(str||'is not palindrom');
987
988
        end if;
989
990
       end;
991
992
993
994
995
996
       6. Write a program to read in a number and print it out digit by digit, as a series of words. For
997
       number 523 would be printed as "five two three". Use decode function within a for loop. Display th
998
       results on the screen using dbms_output.put_line.
999
1000
1001
1002
1003
       declare
1004
1005
        num varchar(10):='&number';
1006
1007
        i varchar(1);
1008
1009
        c int:=length(num);
```

```
1010
1011
        result varchar(10);
1012
1013
       begin
1014
1015
        dbms_output.put_line('ENTERED NO. IS');
1016
1017
        loop
1018
1019
        i:=substr(num,1,1);
1020
1021
        num:=substr(num,2);
1022
1023
        select decode(i, 1, 'one', 2, 'two', 3, 'three', 4, 'four', 5, 'five', 6, 'six', 7, 'seven', 8, '
1024
       ,'nine','zero') into result from dual;
1025
1026
        dbms_output.put_line(result);
1027
1028
        exit when c=1;
1029
1030
        c:=c-1;
1031
1032
        end loop;
1033
1034
       end;/
1035
1036
1037
1038
1039
1040
       PL*SQL
1041
       #Exercise 5
1042
1043
1044
1045
1046
       1. Create a table SCHOOL which has the following structure:-
1047
1048
1049
       Roll _no Number 4
1050
1051
       Name Varchar2 20
1052
1053
       Section Number 4
1054
1055
       Class Character 7
1056
1057
       Oracle Number 3
1058
```

```
1059
       Dev_2000 Number 3
1060
1061
        Fill in the following sample data:-
1062
1063
       Roll no. Name Section Class Oracle Dev_2000
1064
1065
       1 Mukesh Khanna 9012 Working 55 80
1066
1067
       2 Rajiv Chawala 9025 Student 75 85
1068
1069
       3 Pramila Bordes 9025 Working 45 45
1070
1071
       4 Nitish Bharadwaj 9025 Working 67 75
1072
1073
       5 Anita Sood 9012 Student 86 72
1074
1075
       6 Kalyani Deshmukh 9012 Working 55 65
1076
1077
       7 Rakesh Surana 9025 Working 95 95
1078
1079
       8 Alok Kumar Nath 9025 Working 25 40
1080
1081
       9 Sushmita Bannerjee 9025 Student 73 83
1082
1083
       10 Pranay Aiyyer 9012 Student 62 85
1084
1085
       11 Shalini Patel 9012 Student 35 00
1086
1087
       12 Ketan Tendulkar 9012 Working 83 98
1088
1089
       13 Arun Trivedi 9012 Working 67 53
1090
1091
       14 Victor D souza 9025 Working 59 63
1092
1093
       15 Sarah Ahmed 9025 Student 65 73
1094
1095
        Create another table with the following structure:-
1096
1097
       Roll_no Number 4
1098
1099
       Total Number 3
1100
1101
       Percent Number 5,2
1102
1103
       Grade Varchar2 10
1104
1105
       Insert into this table the total marks, percentage and grades of the respective students. The rule
1106
       grades are as follows:-
1107
```

```
1108
      For working persons
1109
1110
      Percentage Grade
1111
1112
      < 50 % FAIL
1113
1114
      >= 50 % PASS
1115
1116
1117
1118
      For students
1119
1120
      Percentage Grade
1121
1122
      < 40% FAIL
1123
1124
      40 - 49.99% C
1125
1126
      50 59.99% B
1127
1128
      60 79.99% A
1129
1130
1131
      >= 80% HONOURS
1132
1133
      -----tables-----
1134
      create table SCHOOL
1135
1136
      (
1137
1138
      Roll_no Number(4),
1139
1140
      Name Varchar2(20),
1141
1142
      Section Number(4),
1143
1144
      Class Character(7),
1145
1146
      Oracle Number(3),
1147
1148
      Dev 2000 Number(3)
1149
1150
      )
1151
1152
1153
1154
      insert all
1155
1156
      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
        into school values(6, 'Kalyani Deshmukh', 9012, 'Working', 55, 65)
1166
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
```

```
1206
       create table Dev_2000_result
1207
1208
1209
1210
       Roll_no Number (4),
1211
1212
       Total Number (3),
1213
1214
       Percent Number (5,2),
1215
1216
       Grade Varchar2 (10)
1217
1218
       )
1219
1220
       /
1221
1222
       -----anonyms procedure for Dev_2000_result result-----
1223
       declare
1224
1225
1226
       v_student school%rowtype;
1227
1228
       v_result oracle_result%rowtype;
1229
1230
       grade varchar2(10);
1231
1232
       cursor c1 is select * from SCHOOL;
1233
1234
       begin
1235
1236
       for v_student in c1
1237
1238
       loop
1239
1240
       if v_student.class='Working' then
1241
1242
       if v_student.Dev_2000 <50 then
1243
1244
        grade:='FAIL';
1245
1246
        else
1247
1248
        grade:='PASS';
1249
1250
       end if;
1251
1252
       elsif v_student.class='Student' then
1253
1254
        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
```

```
1353
1354
1355
1356
       Meter Number Varchar2 4
1357
1358
       Meter Type Character 1
1359
1360
       Previous Reading Number 5
1361
1362
       Current Reading Number 5
1363
1364
       Customer Type Character 1
1365
1366
       Last Bill payment Character 1 (values could be Y or N )
1367
1368
1369
       There are two types of meters viz. 3- phase or 1-phase coded as T or S respectively. There are
1370
1371
       of customers viz. Agricultural Industrial, Commercial and Residential with coeds A , I , C an
1372
       respectively.
1373
1374
1375
1376
       Formulae:-
1377
1378
       Units used = Current Reading Previous Reading
1379
1380
       Rate =Rs.1/ 1.25/ 1.50/ 1.30 for A/I/C/R respectively.
1381
1382
       Amount = rate*units used
1383
1384
       Surcharge = 5% for single phase
1385
1386
       10% for 3 phase
1387
1388
       Excise = 30% of (amount +Surcharge)
1389
1390
       Net = Amount +Surcharge + Excise
1391
1392
1393
1394
       Write a block to calculate the bill for each customer. The program should insert the Meter no., Un
1395
       used, Rate, Amount, Surcharge, Excise duty and Net for each customer into some other suitable tabl
1396
       Also, at the end, it should insert the total Amount, Surcharge, Excise and Net into some other tab
1397
1398
       -----tables-----
1399
       create table CUSTOMER
1400
1401
```

```
1402
1403
       "Meter Number" Varchar2(4),
1404
       "Meter Type" Character(1),
1405
1406
1407
       "Previous Reading" Number(5),
1408
       "Current Reading" Number(5),
1409
1410
1411
       "Customer Type" Character(1),
1412
1413
       "Last Bill payment" Character(1) check("Last Bill payment"='Y' OR "Last Bill payment"='N')
1414
1415
      )/
1416
1417
      //insert dummy data into table customer
1418
1419
1420
      Mete M
                             Previous Reading
                                                          Current
                                                                                        Reading
1421
1422
      1423
1424
      1000
                             S
                                                                         3000
1425
1426
      1001
                             Т
                                                                         3000
1427
                             S
1428
      1002
                                                                         4000
1429
1430
1431
1432
1433
1434
1435
      create table bill
1436
1437
1438
1439
      "Meter Number" Varchar2(4) primary key,
1440
1441
      units number,
1442
1443
      rate number,
1444
1445
      amount number,
1446
1447
      surcharge number,
1448
1449
      Excise number,
1450
```

```
1451
      Net number
1452
1453
      )
1454
1455
      /
1456
1457
       -----procedure-----
1458
      create or replace procedure calculatebill
1459
1460
      as
1461
1462
       v_customer customer%rowtype;
1463
1464
       v_bill bill%rowtype;
1465
1466
        cursor c1 is select * from customer;
1467
1468
        rate number(3,2);
1469
1470
        units number;
1471
1472
        amount number;
1473
1474
        surcharge number;
1475
1476
       Excise number;
1477
1478
       Net number;
1479
1480
      begin
1481
1482
        delete from bill;
1483
1484
       for v_customer in c1
1485
1486
       loop
1487
1488
        select decode(v_customer."Customer Type",'A',1,'I',1.25,'C',1.50,'R',1.30) into rate
1489
       from dual;
1490
1491
        select decode(v_customer."Meter Type",'T',10,'S',5) into surcharge from dual;
1492
1493
        units:=v_customer."Current Reading"-v_customer."Previous Reading";
1494
1495
        amount:=rate*units;
1496
1497
        surcharge:=surcharge*amount;
1498
1499
        Excise:=(amount +Surcharge)*30/100;
```

```
1500
1501
       Net:= Amount +Surcharge + Excise;
1502
1503
       Insert into bill values(v customer."Meter Number"
1504
       ,units,rate,amount,surcharge,Excise,Net);
1505
1506
       end loop;
1507
1508
      end;
1509
1510
1511
1512
      -----output------
1513
1514
1515
      Compile:
1516
1517
1518
1519
      SQL> alter procedure calculatebill compile;
1520
1521
1522
1523
      Call:
1524
1525
      SQL> exec calculatebill
1526
1527
1528
1529
1530
1531
      3. A table consists of the following fields:-
1532
1533
1534
1535
1536
      Invoice Number Varchar2 4
1537
1538
      Invoice Date Date
1539
1540
      Customer Code Number 1
1541
1542
      Product Code Number 1
1543
1544
      Quantity Sold Number 3
1545
1546
1547
1548
```

```
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------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
1581
1582
      rj24
                          31-MAR-12
                                                                                         1
1583
1584
      rj24
                                                                                         2
                           31-MAR-12
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
       cursor c4(v\_pcode number) is select * from corder where "Product Code "=v\_pcode;
1624
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
```

```
1647
1648
       for j in c2(i."Customer Code ")
1649
1650
       loop
1651
1652
       select decode(j."Product Code ",0,15,1,35,2,42,3,51,4,60,0) into rs from dual;
1653
1654
       rs:=rs*j."Quantity Sold ";
1655
1656
       TOTAL PURCHASE:=TOTAL PURCHASE+rs;
1657
1658
       end loop;
1659
1660
       insert into totalpurchase values(i."Customer Code ",TOTAL_PURCHASE);
1661
1662
       end loop;
1663
1664
1665
1666
       rs:=0;
1667
1668
1669
1670
       for i in c3
1671
1672
       loop
1673
1674
       for j in c4(i."Product Code ")
1675
1676
       loop
1677
1678
       select decode(j."Product Code ",0,15,1,35,2,42,3,51,4,60,0) into rs from dual;
1679
1680
       rs:=rs*j."Quantity Sold ";
1681
1682
       TOTAL_SALE:=TOTAL_SALE+rs;
1683
1684
       end loop;
1685
1686
       insert into totalsale values(i."Product Code ",TOTAL_SALE);
1687
1688
       end loop;
1689
1690
      end;
1691
1692
1693
1694
       ------output-----
1695
      SQL>exec find_sale_purchase;
```

```
1696
1697
      SQL> select * from totalsale;
1698
1699
      Product
                                     Code Total Sale
1700
1701
      1702
1703
      1
                                            105
1704
1705
      2
                                            231
1706
1707
                                            276
1708
1709
1710
1711
1712
      SQL> select * from totalpurchase;
1713
1714
1715
1716
      Customer
                                     Code Total Purchase
1717
1718
      1719
1720
                                            276
1721
1722
1723
1724
1725
      4. Create a table EMPLOYEE with the following columns:-
1726
1727
1728
1729
1730
      Employee No. Varchar2 4
1731
1732
      Employee Name Varchar2 30
1733
1734
      Designation Varchar2 10
1735
1736
      Category Character 1
1737
1738
      Basic Salary Number 4
1739
1740
1741
1742
1743
1744
      Category may be \ensuremath{\mathsf{J}} , \ensuremath{\mathsf{S}} , or \ensuremath{\mathsf{W}} for \ensuremath{\mathsf{Jr}} . officers or Worker category.
```

```
1745
1746
1747
1748
      Formulae:-
1749
1750
      DA = 35% of Basic Salary correct up to paise.
1751
1752
      HRA = 15% of Basic Salary subject to a maximum of Rs. 250/1000/30000 for categories W/J/S
1753
      respectively.
1754
1755
      Gross = Basic Salary +DA +HRA
1756
1757
1758
1759
       Output the Employee Number and the Gross for each employee in a separate table.
1760
1761
       -----tables-----
1762
      create table EMPLOYEE
1763
1764
1765
1766
      "Employee No " Varchar2 (4),
1767
1768
      "Employee Name" Varchar2 (30),
1769
1770
      Designation Varchar2 (10),
1771
1772
      Category Character (1),
1773
      "Basic Salary " Number (4)
1774
1775
1776
      )
1777
1778
      /
1779
1780
      Insert dummy data:-
1781
1782
      Empl
                   Employee Name DESIGNATIO
                                                                          Basic Salary
1783
1784
      1785
1786
      1000
                   rakesh
                                               Sr.off
                                                                                 3000
1787
1788
      1001
                   peeyoosh
                                               Sr.off
                                                                                 4000
1789
1790
      1002
                   malik
                                               Jr.off
                                                                                 5000
1791
1792
1793
```

```
1794
       create table employee_gross
1795
1796
       create table gross
1797
1798
       (
1799
1800
       "Employee No " Varchar2 (4),
1801
1802
1803
       "Gross Salary " Number (4)
1804
1805
       )
1806
1807
       /
1808
1809
       -----procedure-----
1810
       create or replace procedure gross
1811
1812
       as
1813
1814
       cursor c1 is select * from employee;
1815
1816
       emp_record employee%rowtype;
1817
1818
       da number(20,2);
1819
1820
       hra number(20,2);
1821
1822
       gross number(20,2);
1823
1824
       begin
1825
1826
       delete from employee gross;
1827
1828
       for emp_record in c1
1829
1830
       loop
1831
1832
       da:=emp_record."Basic Salary "*35/100;
1833
1834
       hra:=emp record."Basic Salary "*15/100;
1835
1836
       if emp_record.Category='j' and hra>250 then
1837
1838
       hra:=250;
1839
1840
       elsif emp_record.Category='s' and hra>1000 then
1841
1842
       hra:=1000;
```

```
1843
1844
       elsif emp_record.Category='w' and hra>30000 then
1845
1846
       hra:=30000;
1847
1848
       else
1849
1850
       hra:=0;
1851
1852
       end if;
1853
1854
       gross:=emp_record."Basic Salary "+da+hra;
1855
1856
       insert into employee_gross values(emp_record."Employee No ",gross);
1857
1858
       end loop;
1859
1860
       end;
1861
1862
1863
1864
       ------output------
1865
1866
      SQL>exec gross;
1867
1868
      Empl Gross
                                   Salary
1869
1870
      1871
1872
      1000
                                   4050
1873
1874
      1001
                                   5400
1875
1876
      1002
                                   6750
1877
1878
1879
1880
1881
1882
      PL*SQL
1883
1884
      #Exercise 6
1885
1886
1887
1888
      1. The median of an array of numbers is the element m of the array such that half the remaining nu
1889
      or equal to m and half are less than or equal to m, if the number of elements in the array is odd.
1890
      the median is the average of the two elements m1 and m2 such that half the remaining elements are
1891
      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  ${
m i}$ 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:-

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.

123451250.75

```
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*SOL
1985
1986
       #Exercise 7
1987
1988
1989
```

```
1990
      1. Write a PL*SQL block that prompts the user to enter the salary of an employee. Your program sho
1991
      display the name of the employee (from the EMP table) who s getting that salary. If more than 1
1992
      employee is receiving that salary, or if no employees exist getting that salary, your program should
1993
      display appropriate messages. Use too many rows and no data found exceptions to achieve this.
1994
      Display the results on the screen using dbms_output.put_line.
1995
1996
1997
       -----TABLE------TABLE------
1998
1999
      create table emp
2000
2001
      (
2002
2003
      enpno varchar(4),
2004
2005
      empname varchar(30),
2006
2007
      designation varchar2(10),
2008
2009
      category char(1),
2010
2011
      basicsalary number(4),
2012
2013
      joined date
2014
2015
      )
2016
2017
2018
2019
       -----EXCEPTION CODE-----
2020
2021
      DECLARE
2022
2023
      mysal number;
2024
2025
      sal NUMBER:=&salary;
2026
2027
      BEGIN
2028
2029
      select basicsalary into mysal from emp where basicsalary =sal;
2030
2031
      exception
2032
2033
      when too many rows then
2034
2035
      dbms_output.put_line('too many data found in result can''t handel ');
2036
2037
      when no data found then
2038
```

```
2039
       dbms_output.put_line('no data found for your query');
2040
2041
       END;
2042
2043
2044
2045
2046
2047
       2. Write a PL*SQL block to check if any employee from EMP table is receiving a salary greater than
2048
       9999.99. Make the use of value_error exception to achieve this. Display the results on the screen
2049
       dbms output.put line.
2050
2051
2052
        -----EXCEPTION CODE-----
2053
2054
       DECLARE
2055
2056
       mysal number;
2057
2058
       sal NUMBER:=&salary;
2059
2060
       BEGIN
2061
2062
       insert into emp values('1003','kumar','sr.off','s',sal,sysdate);
2063
2064
       exception
2065
2066
       when value_error then
2067
2068
       dbms output.put line('salary is limited to 4 digits only');
2069
2070
       when others then
2071
2072
       dbms_output.put_line('un identified error occured');
2073
2074
2075
       END;
2076
2077
2078
2079
2080
2081
2082
2083
       3. Create a user-defined exception by the name of exp_check. Select the ename and hiredate of all
2084
       employees into a cursor. Your program should calculate the experience of all the employees in year
2085
       and insert the ename and experience of each employee into tempp table. If any employee has experie
2086
       less than 2 years, the program should be aborted with a suitable message. Raise the user-defined
2087
       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</pre>
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
```

```
2137
      -----EXCEPTION CODE------
2138
2139
      create or replace function triangle(a number, b number, c number)
2140
2141
      return boolean
2142
2143
      as
2144
2145
      invalid_triangle exception;
2146
2147
      begin
2148
2149
      if not (a+b)=c and b+c>=a and c+a>=b) then
2150
2151
      raise invalid_triangle;
2152
2153
      else
2154
2155
      return true;
2156
2157
      end if;
2158
2159
      exception
2160
2161
      when invalid_triangle then
2162
      dbms_output.put_line('xxxxxx- invalid triangle -xxxxxxxxxx');
2163
2164
2165
      return false;
2166
2167
2168
      when others then
2169
2170
      dbms_output.put_line('un identified error occured');
2171
2172
      END;
2173
2174
2175
2176
2177
2178
       2179
2180
      declare
2181
2182
      a number:=&side1;
2183
2184
      b number:=&side2;
2185
```

```
2186
       c number:=&side3;
2187
2188
       x boolean;
2189
2190
       begin
2191
2192
        x:=triangle(a,b,c);
2193
2194
       end;
2195
2196
2197
2198
       SQL> /
2199
2200
       Enter value for side1: 2
2201
2202
       old 2: a number:=&side1;
2203
2204
       new 2: a number:=2;
2205
2206
       Enter value for side2: 3
2207
2208
       old 3: b number:=&side2;
2209
2210
       new 3: b number:=3;
2211
2212
       Enter value for side3: 5
2213
2214
       old 4: c number:=&side3;
2215
2216
       new 4: c number:=5;
2217
2218
2219
2220
       5. Write a function that generates a random number between 1 and 10. Use any logic of your choice
2221
       achieve this. Display the results on the screen using dbms_output.put_line.
2222
2223
2224
       -----RANDOM METHOD-----
2225
2226
       declare
2227
2228
       x number;
2229
2230
       begin
2231
2232
       select trunc(dbms_random.value(1,10)) into x from dual;
2233
2234
       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
       ------ANONYMS-PROGRAM------
2297
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
```

```
2333
2334
2335
      7. Create a function that accepts a string of n characters and exchanges the first character with
2336
      second with the next to last, and so forth until n exchanges have been made. What will the fin
2337
      string look like? Write the function to verify your conclusion. Display the results on the screen
2338
      dbms_output.put_line.
2339
2340
2341
      -----FUNCTION------
2342
2343
      create or replace function revstr(st in out varchar2)
2344
2345
       return boolean
2346
2347
       as
2348
2349
       len number:=length(st);
2350
2351
       begin
2352
2353
       for j in 1..len
2354
2355
2356
       loop
2357
2358
       st:=st||substr(st,len-j,1);
2359
2360
       end loop;
2361
2362
       st:=substr(st,len);
2363
2364
       len:=length(st);
2365
2366
       st:=substr(st,1,len-1);
2367
2368
       dbms_output.put_line('reverse string is '||st);
2369
2370
       return true;
2371
2372
       end;
2373
2374
2375
2376
2377
2378
2379
2380
          -----CALLING-CODE------
2381
```

```
2382
      declare
2383
2384
      k boolean;
2385
2386
      val varchar2(50):='rakesh kumar';
2387
2388
      begin
2389
2390
      k:=revstr(val);
2391
2392
      end;
2393
2394
2395
2396
2397
2398
2399
2400
2401
      PL*SQL
2402
2403
      #Exercise 8
2404
2405
2406
2407
      1. Write a stored procedure by the name of Comp intr to calculate the amount of interest on a bank
2408
      account that compounds interest yearly. The formula is:-
2409
2410
2411
       I = p (1 + r/100) y p
2412
2413
      where:-
2414
2415
       I is the total interest earned.
2416
2417
       p is the principal.
2418
2419
       r is the rate of interest as a decimal less than 1, and
2420
2421
       y is the number of years the money is earning interest.
2422
2423
2424
2425
      Your stored procedure should accept the values of p, r and y as parameters and insert the Interest
2426
      Total amount into tempp table.
2427
2428
       -----PROCEDURE-----
2429
2430
      CREATE OR REPLACE PROCEDURE COMP_INTR (P IN NUMBER, R IN NUMBER, Y IN
```

```
2431
      NUMBER)
2432
2433
      AS
2434
2435
      I NUMBER(6,2);
2436
2437
      BEGIN
2438
2439
      I:=P*POWER(1+R,Y)-P;
2440
2441
      DBMS OUTPUT.PUT LINE(I);
2442
2443
      END;
2444
2445
2446
2447
      ------CALLING -PROCEDURE------
2448
2449
      SQL> begin
2450
2451
       2 comp_intr(1600,9,4);
2452
2453
       3 end;
2454
2455
       4 /
2456
2457
      658.530576
2458
2459
2460
2461
      0r
2462
2463
2464
2465
      SQL> exec comp_intr(1600,9,4);
2466
2467
2468
2469
      2. Create a stored function by the name of Age_calc. Your stored function should accept the date o
2470
      of a person as a parameter. The stored function should calculate the age of the person in years, m
2471
      and days e.g. 35 years, 3 months, 17 days. The stored function should return the age in years dire
2472
      (with the help of Return statement). The months and days are to be returned indirectly in the form
2473
      OUT parameters. Write a PL*SQL block to accept the date of birth of an employee from the user, cal
2474
      the stored function, and display the age of the employee on the screen. Display the above results
2475
      screen using dbms_output.put_line.
2476
2477
2478
        ------FUNCTION------
2479
```

```
2480
2481
        create or replace function age_calc(dat in date,d out number,m out number)
2482
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
       DBMS OUTPUT.PUT LINE('DAYS: '||P2||' MONTHS: '||P3||' YEARS: '||P1);
2527
2528
```

```
2529
       END;
2530
2531
2532
2533
2534
2535
       3. Create a package by the name of Payroll calc. The package should contain separate procedures fo
2536
       HRA, Gross, Tax and Net calculation.
2537
2538
2539
       Formulae:-
2540
2541
       DA = 10% of Sal for Managers and 5% of Sal for others.
2542
2543
       HRA = 20% of Sal for employees of department 10 and 7% of Sal for employees of other departments.
2544
2545
       Gross = Sal + DA + HRA.
2546
2547
       If Gross exceeds 4000, Tax is to be deducted at 5% of the amount exceeding 4000. If Gross exceeds
2548
       5000, Tax is to be deducted at 5% of the amount exceeding 4000 and an additional of 2% of the amou
2549
       exceeding 5000.
2550
2551
       Net = Gross
                    Tax.
2552
2553
       Write a PL*SQL block that calls the procedures from the above package. The PL*SQL block should
2554
       print the Pay slips for all the employees. The format of the Pay slip should be as follows:-
2555
2556
2557
2558
2559
2560
        Name: - KING Designation: - PRESIDENT Dept: - ACCOUNTING
2561
2562
        Sal:- xxx DA:- xxx HRA:- xxx Gross:- xxx Tax:- xxx Net:- xxx
2563
        ************************
2564
2565
2566
2567
2568
2569
2570
       PL*SQL
2571
2572
       #Exercise 9
2573
2574
2575
2576
       Create the following 3 tables and insert sample data as shown:-
2577
```

 122, 12.00	1 141	abbookonapt	orz.txt at master i modula	110/GDDOOK CILITAD			
2578	Ord_mst						
2579							
2580	Ord_no	Cust_cd	Status				
2581							
2582	······································						
2583	4	C1					
2584	1	C1		Р			
2585 2586							
2587							
2588							
2589							
2590	Ord_dtl						
2591	01 4_4 21						
2592	Ord_no	Prod_cd	Qty				
2593	01 <u>4_</u> 110		ę.,				
2594	<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<						
2595							
2596	1	P1		100			
2597							
2598	1	P2		200			
2599							
2600							
2601							
2602							
2603							
2604	Prod_mst						
2605							
2606	Prod_cd	Prod_name	Qty_i	.n_stock	Booked_qty		
2607							
2608	<<<<<<	······	<<<<<<<	<<<			
2609							
2610	P1	Floppies		10000			
2611							
2612	P2	Printers		5000			
2613	D2	Madawa		2000			
2614 2615	Р3	Modems		3000			
2616							
2617							
2618	1 Write a Refore Insert trigger on Ord dtl Anytime a now is inserted in Ord dtl the Poeked aty						
2619	1. Write a Before Insert trigger on Ord_dtl. Anytime a row is inserted in Ord_dtl, the Booked_qty should be increased accordingly.						
2620	22.2.00 20 21101						
2621							
2622			TABLES				
2623							
2624	create table	Ord_mst					
2625							
2626	(						

```
2627
2628
       Ord_no number,
2629
       Cust_cd varchar(2),
2630
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
        -----TRIGGER------TRIGGER-----
2675
```

```
2676
2677
      create or replace trigger Ord_dtl_1 before insert on Ord_dtl for each row
2678
2679
      begin
2680
2681
      update Prod_mst set Booked_qty=Booked_qty-:new.Qty where Prod_cd=:new.Prod_cd;
2682
2683
      dbms_output.put_line();
2684
2685
      end;
2686
2687
      /
2688
2689
      -----OUTPUT------
2690
2691
      SQL> insert into Ord_dtl values(1,'P1',50);
2692
2693
      SQL> select * from ord_dtl;
2694
2695
2696
2697
      ORD NO
                   PR
                                       QTY
2698
2699
      2700
2701
      1
                          Ρ1
                                              100
2702
2703
      1
                          P2
                                              200
2704
2705
      1
                          Ρ1
                                              50
2706
2707
      SQL> select * from prod mst;
2708
2709
2710
2711
     PR
                   PROD NAME
                                              QTY_IN_STOCK
                                                                        BOOKED_QTY
2712
2713
      2714
2715
      Ρ1
                   Floppies
                                              10000
                                                                               1050
2716
2717
     P2
                   Printers
                                              5000
                                                                               600
2718
2719
      Р3
                   Modems
                                              3000
                                                                               200
2720
2721
2722
2723
      2. Write a Before Delete trigger on Ord dtl. Anytime a row is deleted from Ord dtl, the Booked qty
2724
      Prod_mst should be decreased accordingly.
```

2.39 1	T IVI	dbbook/chapterz.txt a	i master - rksuthar 19/upbook -	Oill lub			
25							
26 27	TRIGGER CREATE or replace TRIGGER ORD_DTL_2 BEFORE DELETE ON ORD_DTL						
28							
29 30							
1	FOR EACH ROW						
2							
33	BEGIN						
34	<pre>update Prod_mst set Booked_qty=Booked_qty-:old.Qty where Prod_cd=:old.Prod_cd;</pre>						
5							
2736							
dbms_output.put_line('data deleted and updated in prod_mst table'); 2738							
9	END;						
0 1			OUTDUT				
			001201				
3	SQL> delete from ord_dtl	whoma atv-E0:					
)  -	SQL> defece from ord_dcf	where qty=50,					
5							
5							
,	PR	PROD_NAME	QTY_IN_STOCK	BOOKED_QTY			
	FIX	PROD_NAME	Q11_11_310CK	DOOKED_QTT			
	<<<<<<<<<						
	***************************************						
	P1	Floppies	10000	1000			
	12	1 1000103	10000	1000			
	P2	Printers	5000	600			
	P3	Modems	3000	200			
	3. Write a Before Update o	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 sho	uld be increased/decre	ased accordingly.				
L							
			TRIGGER				
	<pre>create trigger Prod_cd_3 b</pre>	efore update on ord_dt	1				
	for each row						
)	begin						
.							
2	update prod_mst set booked	_qty=booked_qty-:old.q	ty+:new.qty, prod_cd=:	new.prod_cd			
3							

```
2774
     where prod_cd=:old.prod_cd;
2775
2776
     end;
2777
2778
     -----FIRST UPDATE-----
2779
2780
     SQL> select * from ord dtl;
2781
2782
2783
2784
     ORD_NO
                        PR
                                                 QTY
2785
2786
     2787
2788
     1
                              Ρ1
                                                       100
2789
2790
     1
                              P2
                                                       200
2791
2792
2793
     SQL> select * from prod_mst;
2794
2795
2796
2797
2798
     PR
                        PROD NAME
                                                 QTY_IN_STOCK
                                                                   BOOKED QTY
2799
     2800
2801
2802
     Ρ1
                        Floppies
                                                 10000
                                                                          1000
2803
2804
     P2
                        Printers
                                                 5000
                                                                          600
2805
2806
     Р3
                        Modems
                                                 3000
                                                                          200
2807
2808
2809
2810
     SQL> update ord_dtl set qty=300 where prod_cd='P1';
2811
2812
2813
2814
     1 row updated.
2815
2816
2817
2818
     SQL> select * from ord_dtl;
2819
2820
2821
                                                       QTY
2822
     ORD_NO
                        PR
```

-012	.2, 12.00 1	1 141	abbook on aptor 2.txt at madici	TROUBLE TO/GDDOOK CITIED				
	2823							
	2824	······································						
	2825							
	2826	1	P1	300				
	2827							
	2828	1	P2	200				
	2829							
	2830							
	2831							
	2832	SQL> select * from prod_mst;						
	2833							
	2834							
	2835							
	2836	PR	PROD_NAME	QTY_IN_STOCK	BOOKED_QTY			
	2837							
	2838	<<<<<<<<<<						
	2839							
	2840	P1	Floppies	10000	12			
	2841							
	2842	P2	Printers	5000	60			
	2843							
	2844	Р3	Modems	3000	20			
	2845							
	2846							
	2847							
	2848	SECOND UPDATE						
	2849	SECOND OI DATE						
	2850	SQL> update ord_dtl set prod_cd='P3',qty=300 where prod_cd='P1';						
	2851	SQLV aparte of a_act set proa_ca- rs ,qcy=see where proa_ca= r1 ;						
	2852							
	2853							
	2854	1 row updated.						
	2855	- · · · · · · · · · · · · · · · · · · ·						
	2856							
	2857							
	2858	SQL> select * from pro	d mst:					
	2859	SQLY SELECT . II OIII bi or iiist?						
	2860							
	2861							
	2862							
	2863	PR	PROD_NAME	QTY_IN_STOCK	BOOKED_QTY			
	2864	1 11	. NOD_NAME	811 <sup>-11</sup> 310CV	POOKED_611			
	2865							
	2866	· · · · · · · · · · · · · · · · · · ·						
		D1	Flonnics	10000	1200			
	2867	P1	Floppies	10000	1200			
	2868 2869	בח	Dointons	F000	600			
	2009	P2	Printers	5000	600			
	2870 2871	P3	Modems	3000	200			

```
2872
2873
2874
2875
2876
2877
2878
2879
       4. Write a Before Update of Status trigger on Ord mst. If the Status is updated from P (Pending) t
2880
       (Delivered), the Booked_qty and Qty_in_stock from Prod_mst should be decreased accordingly. If the
2881
       Status is updated from P (Pending) to C (Cancelled), the details of the order should be deleted fr
2882
       and corresponding Booked qty from Prod mst should be decreased accordingly. (The Before delete tri
2883
       on Ord_dtl would automatically decrease the Booked_qty from Prod_mst).
2884
2885
2886
        -----TRIGGER------
2887
2888
       CREATE OR REPLACE TRIGGER ORD_MST_1 BEFORE UPDATE ON ORD_MST
2889
2890
       FOR EACH ROW
2891
2892
       Declare -- DECLARE IS NECESSARY
2893
2894
       cursor c1 is select * from Ord_dtl where Ord_no=:old.Ord_no;
2895
2896
       BEGIN
2897
2898
       IF :NEW.STATUS='c' or :NEW.STATUS='C' THEN
2899
2900
       DELETE FROM ORD DTL WHERE ORD NO=:OLD.ORD NO;
2901
2902
       END IF;
2903
2904
       IF (:NEW.STATUS='d') or (:NEW.STATUS='D') THEN
2905
2906
       for i in c1
2907
2908
       loop
2909
2910
       UPDATE PROD MST SET QTY IN STOCK=(QTY IN STOCK-
2911
       i.QTY),booked_QTY=(booked_QTY-i.QTY) where Prod_cd=i.Prod_cd;
2912
2913
       dbms_output.put_line('updated as cancel in prod_mst');
2914
2915
       end loop;
2916
2917
       END IF;
2918
2919
       END;
2920
```

```
2921
2922
2923
      -----OUTPUT------
2924
2925
     SQL> select * from ord_dtl;
2926
2927
      ORD NO PR QTY
2928
2929
2930
2931
      1 P1 100
2932
2933
      1 P2 200
2934
2935
     SQL> select * from ord_mst;
2936
2937
2938
2939
     ORD NO
                 CU
2940
2941
     <<<<<<<<<<<<
2942
2943
     1
                         C1
2944
2945
2946
2947
     SQL> select * from prod_mst;
2948
2949
     PR
                  PROD NAME
                                     QTY IN STOCK BOOKED QTY
2950
2951
     2952
2953
     P1
                  Floppies
                                      10000
                                                                1000
2954
2955
     P2
                  Printers
                                      5000
                                                                600
2956
2957
     Р3
                  Modems
                                      3000
                                                                200
2958
2959
2960
2961
     SQL> update ord mst set status='c';
2962
2963
     data deleted and updated in prod_mst table //shows two rows deleted of ord_mst
2964
2965
     data deleted and updated in prod_mst table
2966
2967
2968
2969
     and if
```

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