INDEX

S.No	Name of the Experiment	Date	Remarks
1.	Create tables department and employee with required constraints.	21-01-2020	
2.	Alter the table		
3.	Add constraint to the table.		
4.	Calculate hra,da,gross and net by using PL/SQL program	28-01-2020	
5.	Salary is updated and its value becomes less than 5000 a trigger has to be raised preventing the operation.		
6.	The assertions are: hra should not be less than 10% of basic and da should not be less than 50% of basic.		
7.	When the da becomes more than 100%, a message has to be generated and with user permission da has to be merged with basic	4-2-2020	
8.	Empno should be unique and has to be generated automatically.		
9.	If the employee is going to retire in a particular month, automatically a message has to be generated.		
10	Display the default value of date of birth.	11-2-2020	
11	Display the description of the fields		
12	Using the average , maximum commands		
13	Using Rollback and Commit.	25-2-2020	
14	Use substitution variables to insert values repeatedly.		
15	Update the wrong information.		
16	Using between and like commands	3-2-2020	
17	Using the delete command.		

18	Create an alias for columns.		
19	Using ascending order commands.	29-12-2020	
20	Use _&&‘ wherever necessary		
21	PL/SQL program		
22	PL/SQL program	30-12-2020	
23	Using the select command on Date functions.		
24	Apply conditions on retrieving the data.		
25	Create a view.	1-1-2021	
26	PL/SQL procedures.		

DBMS LAB

A college consists of number of employees working in different departments. In this context, create two tables employee and department. Employee consists of columns empno, empname, basic, hra, da, deductions, gross, net, date-of-birth. The calculation of hra,da are as per the rules of the college. Initially only empno, empname, basic have valid values. Other values are to be computed and updated later. Department contains deptno, deptname, and description columns. Deptno is the primary key in department table and referential integrity constraint exists between employee and department tables.

Perform the following operations on the database:

1. Create tables department and employee with required constraints.

Query for Employee Table:

```
CREATE TABLE employee (empno INT NOT NULL PRIMARY KEY , empname VARCHAR(20) NOT
NULL , basic INT  ,dob DATE, INT , hra INT , da INT , deductions INT , gross INT, net INT);
```

Output:

Table created.

Query for Department Table:

```
CREATE TABLE department (deptno int NOT NULL PRIMARY KEY , deptname VARCHAR(20) NOT
NULL , description VARCHAR2(100));
```

Output:

Table Created

2. Initially only the few columns (essential) are to be added. Add the remaining columns separately by using appropriate SQL command.

Query for adding the columns:

```
alter table employee add deptno int ;
```

Output:

Table altered.

Query for adding the Foreign key :

```
ALTER   TABLE  employee ADD FOREIGN KEY(deptno) REFERENCES department(deptno);
```

Adding the foreign key “deptno” column in employee table which references to “deptno” in department table.

Output:

Table altered.

3. Basic column should not be null.

Query for adding the NOT NULL Constraint:

```
ALTER TABLE   employee
MODIFY  basic  INT  NOT NULL;
```

Output:

Table altered.

4. Add constraint that basic should not be less than 5000.

Query for adding the NOT NULL Constraint:

```
ALTER TABLE employee
ADD CONSTRAINT basic_check CHECK (basic >= 5000);
```

Output:

Table altered.

5. Calculate hra , da , gross , deductions and net by using PL/SQL program.

```
DECLARE
        e_empno employee.empno%type;
        e_basic employee.basic%type;
        e_hra employee.hra%type;
        e_da employee.da%type;
        e_deductions employee.deductions%type;
        e_gross employee.gross%type;
        e_net employee.net%type;
        CURSOR e_employees is
        select empno , basic from employee;
BEGIN
        OPEN e_employees ;
        LOOP
                FETCH e_employees into e_empno , e_basic ;
                EXIT WHEN e_employees%notfound;
                dbms_output.put_line(e_empno|| ' employee ' || e_basic);

                  IF e_basic > 15000 THEN
                    e_hra := e_basic * 0.12;
                    e_da := e_basic * 0.08;
                    e_deductions := e_basic * 0.22;
                  ELSIF e_basic > 12000 THEN
                    e_hra := e_basic * 0.1;
                    e_da := e_basic * 0.06;
                    e_deductions := e_basic * 0.2;
```

```
          ELSIF e_basic > 12000 THEN
              e_hra := e_basic * 0.1;
              e_da := e_basic * 0.06;
              e_deductions := e_basic * 0.2;
          ELSIF e_basic> 9000 THEN
              e_hra := e_basic * 0.07;
              e_da := e_basic * .04;
              e_deductions := e_basic * 0.1;
          ELSE
              e_hra := e_basic * .05;
              e_da := e_basic * 200;
              e_deductions := e_basic * 0.01;
            END IF;
            e_net := e_basic + e_hra + e_da - e_deductions;
            e_gross := e_basic + e_hra + e_da;
            DBMS_OUTPUT.PUT_LINE ('BASIC: ' || e_basic);
            DBMS_OUTPUT.PUT_LINE ('HRA: ' || e_hra);
            DBMS_OUTPUT.PUT_LINE ('DA: '|| e_da);
            DBMS_OUTPUT.PUT_LINE ('DEDUCTIONS:  '||e_deductions);
            DBMS_OUTPUT.PUT_LINE ('GROSS: ' || e_gross);
            DBMS_OUTPUT.PUT_LINE ('NET: ' || e_net);
            UPDATE employee
            SET hra = e_hra , da = e_da , deductions = e_deductions , gross = e_gross ,
                   net = e_net        WHERE empno = e_empno;

          END LOOP;
          CLOSE e_employees;


END;
```

Output:

```
106employee 23000
BASIC: 23000
HRA: 2760
DA: 1840
DEDUCTIONS:  5060
GROSS: 27600
NET: 22540
```

1 rows updated.

6. Whenever salary is updated and its value becomes less than 5000 a trigger has to be raised preventing the operation.

```
CREATE OR REPLACE TRIGGER salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON employee
FOR EACH ROW
WHEN (NEW.empno > 0 and NEW.basic<5000)

BEGIN
dbms_output.put_line('changes are not allowed ');

RAISE_APPLICATION_ERROR(-20101, 'the salary updation is not accepted');

END;
```

Output:

Trigger is created.

7. The assertions are: hra should not be less than 10% of basic and da should not be less than 50% of basic.

```
Alter table employee add check (hra>sal*0.10);
Alter table employee add check (da>sal*0.45);
```

Output:

1 rows altered.

8. When the da becomes more than 100%, a message has to be generated and with user permission da has to be merged with basic.

```
Alter table employee add check (hra>sal*0.10);
Alter table employee add check (da>sal*0.45);
```

Output:

1 rows altered.
The DA is merged with Basic (On adding a basic greater than da).

9. Empno should be unique and has to be generated automatically.

Query :

```
create sequence seq_empno
start with 1
increment by 1
cache 10 ;
```

```
insert into employee (empno , empname , basic , deptno)  values (seq_emp.nextval, 'emp1' , 23000 ,
1);
```

Output:

1 row’s inserted.

10. If the employee is going to retire in a particular month, automatically a message has to be generated.

```
DECLARE
e_empno employee.empno%type;
e_dob employee.dob%type;
no_months NUMBER;

CURSOR e_employees is
SELECT empno , ADD_MONTHS(dob , 60*12) FROM employee;


BEGIN
OPEN e_employees ;
LOOP
FETCH e_employees into e_empno , e_dob ;
EXIT WHEN e_employees%notfound;
no_months :=MONTHS_BETWEEN (e_dob ,SYSDATE);
IF (no_months<=1 and no_months>=0) THEN
   dbms_output.put_line(e_empno|| '  Employee is going to retire on   ' ||e_dob );
END IF;
```

Output:

```
102 Employee is going to retire on   12-JAN-21
```

11. The default value for date-of-birth is 1 jan, 1970.

Query for adding the Default Constraint:

```
ALTER  TABLE   employee
MODIFY  dob   DEFAULT   '1-jan-1970' ;
```

Output:

Table is altered.

12. Display the information of the employees and departments with description of the fields.

Query for describing the employee and department tables:

```
DESCRIBE   employee  ;
DESCRIBE   department ;
```

Output:

Employee Table

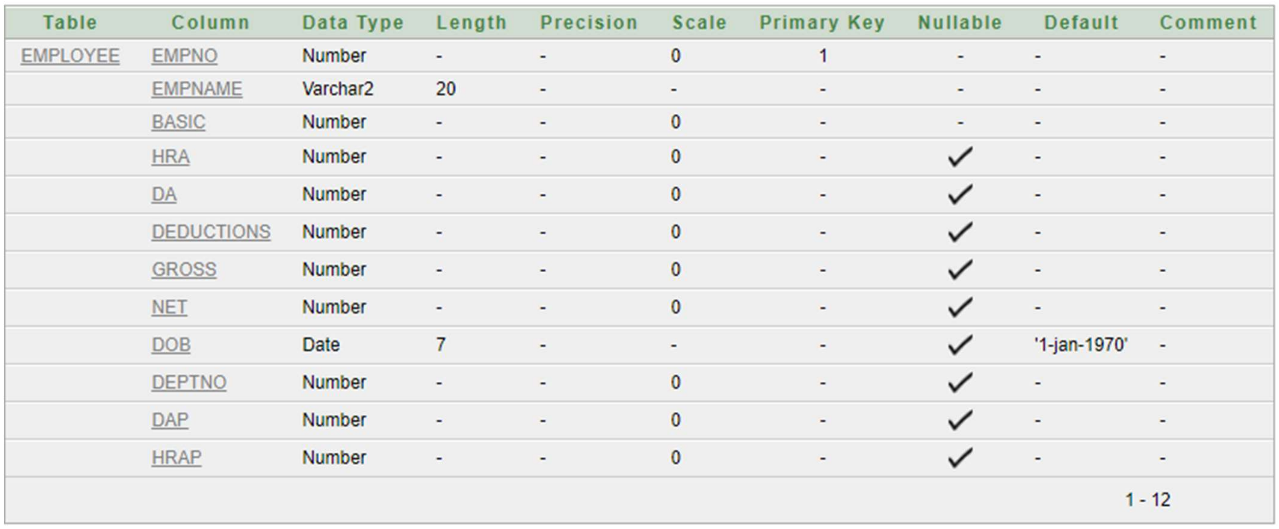


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPLOYEE	EMPNO	Number	-	-	0	1	-	-	-
	EMPNAME	Varchar2	20	-	-	-	-	-	-
	BASIC	Number	-	-	0	-	-	-	-
	HRA	Number	-	-	0	-	✓	-	-
	DA	Number	-	-	0	-	✓	-	-
	DEDUCTIONS	Number	-	-	0	-	✓	-	-
	GROSS	Number	-	-	0	-	✓	-	-
	NET	Number	-	-	0	-	✓	-	-
	DOB	Date	7	-	-	-	✓	'1-jan-1970'	-
	DEPTNO	Number	-	-	0	-	✓	-	-
	DAP	Number	-	-	0	-	✓	-	-
	HRAP	Number	-	-	0	-	✓	-	-
									1 - 12

Department Table

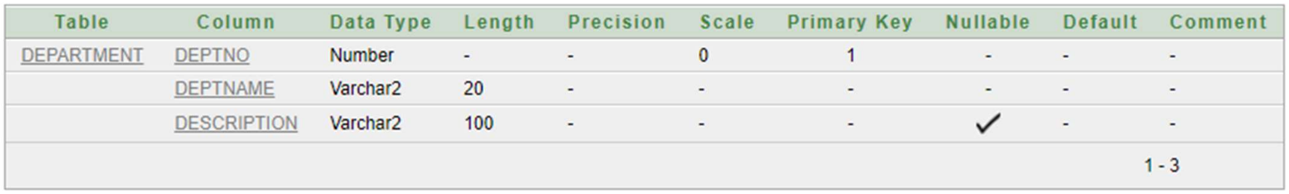


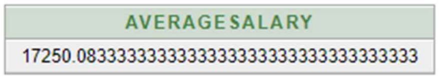
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DEPARTMENT	DEPTNO	Number	-	-	0	1	-	-	-
	DEPTNAME	Varchar2	20	-	-	-	-	-	-
	DESCRIPTION	Varchar2	100	-	-	-	✓	-	-
									1 - 3

13. Display the average salary of all the departments.

Query for finding average salary:

```
SELECT    avg(sal)    AverageSalary     FROM
( SELECT avg(basic) as sal  FROM  employee    GROUP BY   deptno  );
```

Output:



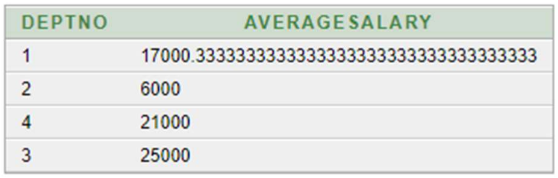
AVERAGESALARY
17250.0833333333333333333333333333333

14. Display the average salary department wise.

Query for finding average salary:

```
SELECT deptno, avg(basic)   AverageSalary  FROM   employee    GROUP BY   deptno  ;
```

Output:



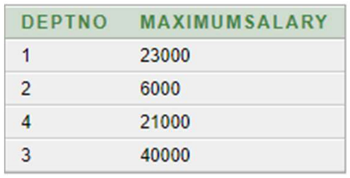
DEPTNO	AVERAGESALARY
1	17000.33333333333333333333333333333333333
2	6000
4	21000
3	25000

15. Display the maximum salary of each department and also all departments put together.

Query for finding maximum salary:

```
SELECT  deptno , max(basic)  MaximumSalary  FROM  employee   GROUP BY   deptno  ;
```

Output:



DEPTNO	MAXIMUMSALARY
1	23000
2	6000
4	21000
3	40000

16. Commit the changes whenever required and rollback if necessary.

Query for finding maximum salary:

```
DELETE FROM emplAoyee WHERE empno = 101;
ROLLBACK;
```

Output:

Rollback Complete.

Query for finding maximum salary:

```
DELETE FROM employee WHERE empno = 101;
COMMIT;
```

Output:

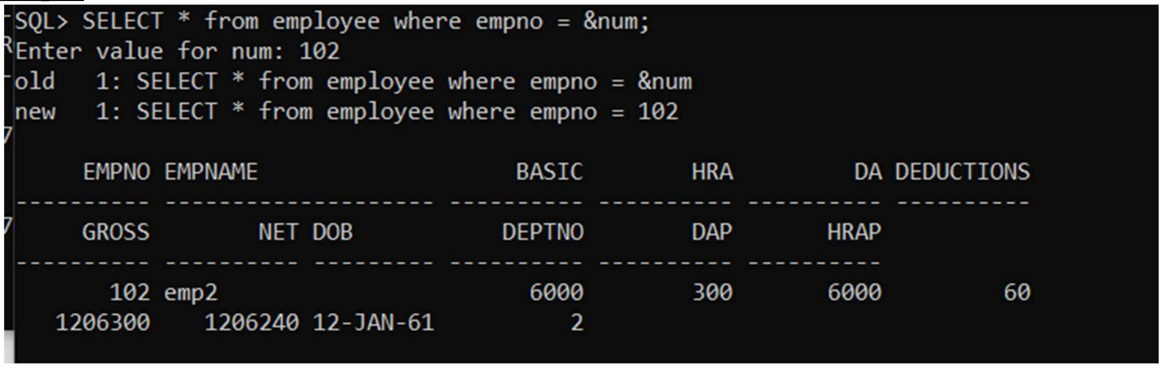
Succesfully Committed.

17. Use substitution variables to insert values repeatedly.

Query :

```
SELECT * from employee where empno = &num;
```

Output:



```
SQL> SELECT * from employee where empno = &num;
Enter value for num: 102
old   1: SELECT * from employee where empno = &num
new   1: SELECT * from employee where empno = 102

     EMPNO EMPNAME                   BASIC        HRA         DA DEDUCTIONS
---------- ------------------- ---------- ---------- ---------- ----------
     GROSS        NET DOB           DEPTNO        DAP       HRAP
---------- ---------- --------- ---------- ---------- ----------
       102 emp2                       6000        300       6000         60
   1206300    1206240 12-JAN-61          2
```

18. Assume some of the employees have given wrong information about date-of-birth. Update the corresponding tables to change the value.

Query :

```
UPDATE   employee
       SET  dob ='04-jan-14'   Where   empno =101;
```

Output:

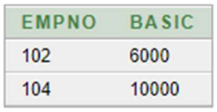
1 rows updated.

19. Find the employees whose salary is between 5000 and 10000 but not exactly 7500

Query :

```
SELECT empno , basic  FROM employee
            WHERE  basic  BETWEEN 5000 AND 10000 AND  NOT  basic=7500;
```

Output:



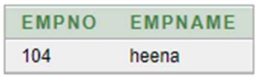
EMPNO	BASIC
102	6000
104	10000

20. Find the employees whose name contains 'en'.

Query :

```
SELECT empno , ename  FROM  employee
       WHERE  empname  LIKE   '%en%' ;
```

Output:



EMPNO	EMPNAME
104	heena

21. Try to delete a particular deptno. What happens if there are employees in it and if there are no employees

Query :

```
DELETE FROM department WHERE deptno = 3;
```

Output:

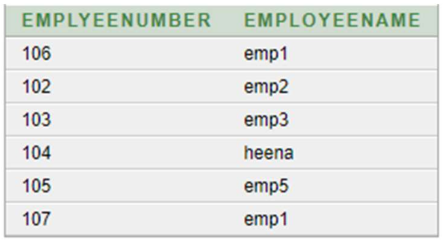
If employees are there in that Department that we are going to delete then the Referential integrity violates and we cannot drop it without dropping corresponding employee entries

22. Create alias for columns and use them in queries.

Query :

```
SELECT  empno  AS   emplyeenumber , empname    AS   employeename FROM  employee;
```

Output:



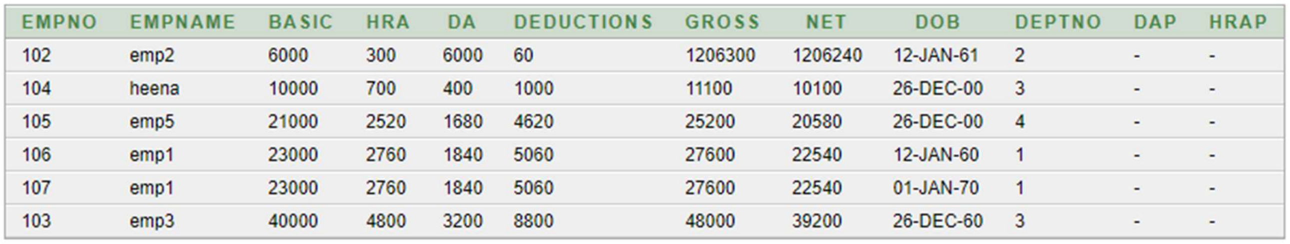
EMPLYEENUMBER	EMPLOYEENAME
106	emp1
102	emp2
103	emp3
104	heena
105	emp5
107	emp1

23. List the employees according to ascending order of salary.

Query :

```
SELECT  * FROM  employee
 ORDER BY  basic asc ;
```

Output:



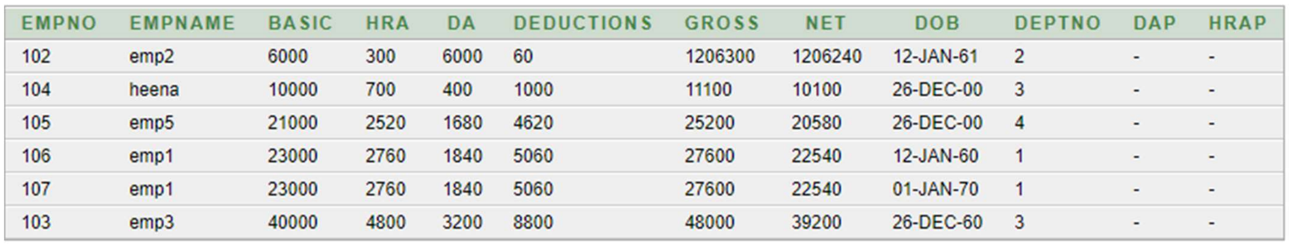
EMPNO	EMPNAME	BASIC	HRA	DA	DEDUCTIONS	GROSS	NET	DOB	DEPTNO	DAP	HRAP
102	emp2	6000	300	6000	60	1206300	1206240	12-JAN-61	2	-	-
104	heena	10000	700	400	1000	11100	10100	26-DEC-00	3	-	-
105	emp5	21000	2520	1680	4620	25200	20580	26-DEC-00	4	-	-
106	emp1	23000	2760	1840	5060	27600	22540	12-JAN-60	1	-	-
107	emp1	23000	2760	1840	5060	27600	22540	01-JAN-70	1	-	-
103	emp3	40000	4800	3200	8800	48000	39200	26-DEC-60	3	-	-

24. List the employees according to ascending order of salary in each department.

Query :

```
SELECT  * FROM  employee
 ORDER BY  basic  asc ,deptno  ;
```

Output:

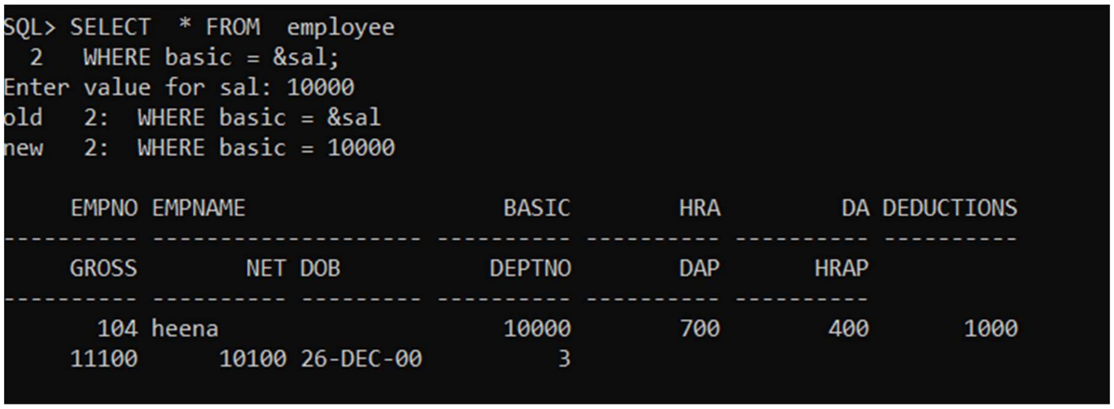


EMPNO	EMPNAME	BASIC	HRA	DA	DEDUCTIONS	GROSS	NET	DOB	DEPTNO	DAP	HRAP
102	emp2	6000	300	6000	60	1206300	1206240	12-JAN-61	2	-	-
104	heena	10000	700	400	1000	11100	10100	26-DEC-00	3	-	-
105	emp5	21000	2520	1680	4620	25200	20580	26-DEC-00	4	-	-
106	emp1	23000	2760	1840	5060	27600	22540	12-JAN-60	1	-	-
107	emp1	23000	2760	1840	5060	27600	22540	01-JAN-70	1	-	-
103	emp3	40000	4800	3200	8800	48000	39200	26-DEC-60	3	-	-

25. Use '&&' wherever necessary

```
SELECT  * FROM  employee
 WHERE basic = &&al;
```

Output:



```
SQL> SELECT  * FROM  employee
  2  WHERE basic = &sal;
Enter value for sal: 10000
old   2:  WHERE basic = &sal
new   2:  WHERE basic = 10000

     EMPNO EMPNAME                   BASIC        HRA         DA DEDUCTIONS
---------- -------------------- ---------- ---------- ---------- ----------
     GROSS        NET DOB           DEPTNO        DAP       HRAP
---------- ---------- --------- ---------- ---------- ----------
       104 heena                     10000        700        400       1000
     11100      10100 26-DEC-00          3
```

26. Amount 6000 has to be deducted as CM relief fund in a particular month which has to be accepted as input from the user. Whenever the salary becomes negative it has to be maintained as 1000 and the deduction amount for those employees is reduced appropriately.

```
DECLARE
        e_empno employee.empno%type;
        e_basic employee.basic%type;
        fund NUMBER;
        remaining_amount NUMBER;
```

```
          CURSOR e_employees is
          select empno , basic from employee;

BEGIN
          OPEN e_employees ;
          LOOP
                    FETCH e_employees into e_empno , e_basic ;
                    EXIT WHEN e_employees%notfound;
                    fund := 6000;
                    dbms_output.put_line(e_empno|| '  Employee  ' || e_basic);
                    remaining_amount := e_basic - fund;
                     IF  remaining_amount < 1000 THEN
                         fund := e_basic -1000;
                         remaining_amount := 1000;
                      END IF;
                      DBMS_OUTPUT.PUT_LINE ('The amount deducted for CM relief fund
                    is : ' || fund);
                      DBMS_OUTPUT.PUT_LINE ('The amount remaining in salary : ' ||
                    remaining_amount);
          END LOOP;
          CLOSE e_employees;
END;
```

Output:

```
106 Employee 23000
The amount deducted for CM relief fund is: 6000
The amount remaining in salary: 17000
102 Employee 6000
The amount deducted for CM relief fund is: 5000
The amount remaining in salary: 1000
103 Employee 40000
```

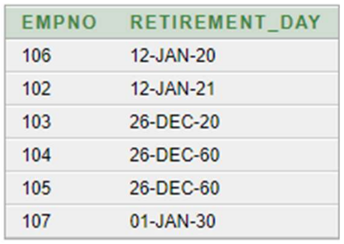
Statement processed.

27. The retirement age is 60 years. Display the retirement day of all the employees.

Query :

```
SELECT  ADD_MONTHS(dob , 60*12)  FROM  employee;
```

Output:



EMPNO	RETIREMENT_DAY
106	12-JAN-20
102	12-JAN-21
103	26-DEC-20
104	26-DEC-60
105	26-DEC-60
107	01-JAN-30

28. Find the employees who are born in leap year.

Query :

```
DECLARE
        e_empno employee.empno%type;
        e_dob employee.dob%type;
        year NUMBER ;
        CURSOR e_employees is
        select empno , dob from employee;
BEGIN
       OPEN e_employees ;
       dbms_output.put_line(' The employees born in leap year : ');
       LOOP
        FETCH e_employees into e_empno , e_dob;
        EXIT WHEN e_employees%notfound;
        year := EXTRACT(YEAR FROM e_dob);
        IF MOD(year, 4)=0  AND   MOD(year, 100)!=0   OR MOD(year, 400)=0  THEN
              dbms_output.put_line(e_empno|| '  Employee  ' || year);
        END IF;
      END LOOP;
      CLOSE e_employees;
END;
```

Output:

```
The employees born in leap year :
106  Employee  2016
103  Employee  1960
104  Employee  2012
105  Employee  2000
```

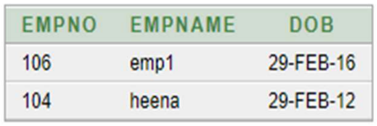
Statement processed.

29. Find the employees who are born on feb 29.

Query :

```
SELECT  *  FROM  employee WHERE  dob   LIKE   '%29-FEB%';
```

Output:



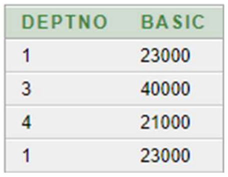
EMPNO	EMPNAME	DOB
106	emp1	29-FEB-16
104	heena	29-FEB-12

30. Find the departments where the salary of atleast one employee is more than 20000.

Query :

```
SELECT deptno , basic FROM  employee WHERE basic>20000;
```

Output:



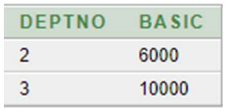
DEPTNO	BASIC
1	23000
3	40000
4	21000
1	23000

31. Find the departments where the salary of all the employees is less than 20000.

Query :

```
SELECT deptno , basic FROM  employee WHERE basic<20000;
```

Output:



DEPTNO	BASIC
2	6000
3	10000

32. As a designer identify the views that may have to be supported and create views.

Query :

```
CREATE VIEW Employee_Details AS
SELECT empno, empname , basic , dob , deptno
FROM employee;
```

Output:

View created.

33. As a designer identify the PL/SQL procedures necessary and create them using cursors.

Query :

```
DECLARE
  a number;
  b number;
  c number;
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
  IF x < y THEN
    z:= x;
  ELSE
    z:= y;
  END IF;
END;

BEGIN
  a:= 23;
  b:= 45;
  findMin(a, b, c);
  dbms_output.put_line(' Minimum of (23, 45) : ' || c);
END
```

Output:

```
Minimum of (23, 45) : 23
```

Statement processed.