# SRI RAMAKRISHNA DEGREE (AUTONOMOUS) COLLEGE :: NANDYAL

## 3RD YEAR B.SC. & B.COM (CA) ORACLE LAB MANUAL

## SQL PROGRAMS

**Program1. Create a table books with attributes with appropriate data type**

| Bookno | Title | Publication | Author | Price | Quantity | edition |
|---|---|---|---|---|---|---|
| Number | Varchar2 | Varchar2 | Varchar2 | Number | Number | Number |
| 3 | 25 | 25 | 25 | 6,2 | 3 | 2 |

**Solution:**
1.Using create command create the table name as books
2.Display the structure of the table using desc command
3. insert any five records into the above created table

```
BOOKNO     TITLE              PUBLICATION    AUTHOR          PRICE  QUANTITY    EDITION
---------- ------------------ -------------- --------------- ------ ----------- -------
1          database concepts  tata           balagurusamy    850    10          4
2          database system    tata           silberschatz    700    17          3
3          database oracle    s.chand & co   s.s.khandars    500    12          2
4          mastering of database bpb publications ivan bayross  900    18          1
5          database system    prentice hall  jeffrey d.ullman 550   20          2
```

**Q2:** Show the list of titles with their authors.
**SQL>** select title,author from books;
**Output:**

```
TITLE                AUTHOR
-------------------- ------------------------
database concepts    balagurusamy
database system      silberschatz
database oracle      s.s.khandars
mastering of database ivan bayross
database system      jeffrey d.ullman
```

**Q3:** List various authors for the book title 'database system'
**Query:**
**SQL>** select * from books where title='database system';
**Output:**

```
BOOKNO     TITLE              PUBLICATION    AUTHOR          PRICE  QUANTITY    EDITION
---------- ------------------ -------------- --------------- ------ ----------- -------
2          database system tata           silberschatz    700    17          3
5          database system prentice hall  jeffrey d.ullman 550    20          2
```

**Q4:** Show the authors details for the table books
**Query:**
**SQL>** select author from books;
**Output:**

```
AUTHOR
------------------------
balagurusamy
silberschatz
s.s.khandars
ivan bayross
jeffrey d.ullman
```

**Q5: Select the list of book details whose price is greater than 800.**
**Query:**
**SQL>** select * from books where price>800.00;
**Output:**

| BOOKNO | TITLE | PUBLICATION | AUTHOR | PRICE | QUANTITY | EDITION |
|---|---|---|---|---|---|---|
| 1 | database concepts | tata | balagurusamy | 850 | 10 | 4 |
| 4 | mastering of database | bpb publications | ivan bayross | 900 | 18 | 1 |

**Q6: List the details of books which have more than 15 copies in the order of price.**
**Query:**
**SQL>** select * from books where quantity>15;
**Output:**

| BOOKNO | TITLE | PUBLICATION | AUTHOR | PRICE | QUANTITY | EDITION |
|---|---|---|---|---|---|---|
| 2 | database system | tata | silberschatz | 700 | 17 | 3 |
| 4 | mastering of database | bpb publications | ivan bayross | 900 | 18 | 1 |
| 5 | database system | prentice hall | jeffrey d.ullman | 550 | 20 | 2 |

**Q7: Display the list of books published by the author 'balagurusamy' with the publication 'TATA'**
**Query:**
**SQL>** select * from books where author='balagurusamy' and publication='tata';
**Output:**

| BOOKNO | TITLE | PUBLICATION | AUTHOR | PRICE | QUANTITY | EDITION |
|---|---|---|---|---|---|---|
| 1 | database concepts | tata | balagurusamy | 850 | 10 | 4 |

**Program 2. Create a table dept with attributes with appropriate data type**

| Dno | Dname | Loc |
|---|---|---|
| Number | Varchar2 | Varchar2 |
| 3 | 20 | 20 |

**Create a table employee with attributes with appropriate data type**

| ssn | name | bdate | salary | mgrssn | Dno |
|---|---|---|---|---|---|
| Number | Varchar2 | Date | Number | Number | Number |
| 6 | 30 | | 10,2 | 6 | 3 |

**Solution:**
**SQL>** create table dept
(dno number(3) primary key,
dname varchar2(20),
loc varchar2(20));

**SQL>** table created

**SQL>** create tabale employee
(ssn number(6), name varchar2(30), bdate date,
salary number(10,2), mgrssn number(6),
dno number(3)
foreign key (dno) references (dept(dno)));
**SQL>** table created
2.Display the structure of the table using desc command
3. insert any records into the above created table

**Dept table:**

| DNO | DNAME | LOC |
|---|---|---|
| 1 | admin | madurai |
| 2 | research | chennai |
| 3 | accounts | bangalore |

**Employee table:**

| SSN | NAME | BDATE | SALARY | MGRSSN | DNO |
|---|---|---|---|---|---|
| 1111 | sathya | 17-DEC-88 | 50000 | 4323 | 3 |
| 1112 | vinotha | 02-AUG-90 | 32000 | 5462 | 1 |
| 1113 | nandu | 07-OCT-93 | 30000 | 6452 | 2 |
| 1114 | rajesh | 05-APR-85 | 40000 | 8264 | 2 |
| 1115 | nive | 06-OCT-92 | 45000 | 7241 | 1 |

**Q2:** Display the names of the employees working for Accounts department.
**Query:**
**SQL>** select name from employee where dno=(select dno from department where dname='accounts');
**Output:**
NAME
---
Sathya

**Q3:** Display names of employees whose salary is greater than the employee SSN=1234
**Query:**
**SQL>** select name from employee where salary>(select salary from employee where ssn=1234);
**Output:**
NAME
---
sathya
vinotha
rajesh
nive

**Q4:** Display all the employees drawing more than or equal to the average salary of department number 3.
**Query:**
**SQL>** select name from employee where salary>=(select avg(salary) from employee group by dno having dno=3);
**Output:**
NAME
---
sathya

**Q5:** Display the name of the highest paid employee.
**Query:**
**SQL>** select name from employee where salary=(select max(salary) from employee);
**Output:**
NAME
---
Sathya

**Q6:** Find the Name and Salary of people who draw in the range Rs. 20,000 to Rs. 40,000.
**Query:**
**SQL>** select name, salary from employee where salary in(select salary from employee where salary between 20000 and 40000);
**Output:**

| NAME | SALARY |
|---|---|
| vinotha | 32000 |
| nandu | 30000 |
| rajesh | 40000 |

**Q7:** Update the salary by 0.25 times for all employees who works in research department.
**Query:**
**SQL>** update employee set salary=(salary*0.25)where dno=(select dno from department where dname='research');

**Output:**
2 rows updated.
**SQL>** select * from employee;

| SSN | NAME | BDATE | SALARY | MGRSSN | DNO |
|---|---|---|---|---|---|
| 1111 | sathya | 17-DEC-88 | 50000 | 4323 | 3 |
| 1112 | vinotha | 02-AUG-90 | 32000 | 5462 | 1 |
| 1113 | nandu | 07-OCT-93 | 37500 | 6452 | 2 |
| 1114 | rajesh | 05-APR-85 | 50000 | 8264 | 2 |
| 1115 | nive | 06-OCT-92 | 45000 | 7241 | 1 |

**Q8:** Delete all the employee details from admin department.
**Query:**
**SQL>** delete from employee where dno=(select dno from department where dname='admin');
**Output:**
2 rows deleted.
**SQL>** select * from employee;

| SSN | NAME | BDATE | SALARY | MGRSSN | DNO |
|---|---|---|---|---|---|
| 1111 | sathya | 17-DEC-88 | 50000 | 4323 | 3 |
| 1113 | nandu | 07-OCT-93 | 30000 | 6452 | 2 |
| 1114 | rajesh | 05-APR-85 | 40000 | 8264 | 2 |

**Q9:** Display the department name in which employee that has highest salary.
**Query:**
**SQL>** select dname from department where dno=(select dno from employee where salary=(select max(salary)from employee);
**Output:**

| DNAME |
|---|
| accounts |

**Q10:** Display the employee details of all employees who earn more than that of 'nandu' and in the same department as 'sathya'
**Query:**
**SQL>** select name from employee where salary>(select salary from employee where name= 'nandu')and dno=(select dno from employee where name= 'sathya');
**Output:**

| SSN | NAME | BDATE | SALARY | MGRSSN | DNO |
|---|---|---|---|---|---|
| 1111 | sathya | 17-DEC-88 | 50000 | 4323 | 3 |
| 1112 | vinotha | 02-AUG-90 | 32000 | 5462 | 1 |
| 1113 | nandu | 07-OCT-93 | 37500 | 6452 | 2 |
| 1114 | rajesh | 05-APR-85 | 50000 | 8264 | 2 |
| 1115 | nive | 06-OCT-92 | 45000 | 7241 | 1 |

**Program 3.Write a PL/SQL code to create an employee database with the tables and fields specified as below.**

a) Employee

| Emp_no | Employee_name | Street | city |
|---|---|---|---|
| Number | Varchar2 | Varchar2 | Varchar2 |
| 10 | 20 | 20 | 20 |

b) Works

| Empno_no | Company_name | Joining_date | Designation | salary |
|---|---|---|---|---|
| Number | Varchar2 | date | Varchar2 | Number |
| 10 | 20 | | 20 | 10,2 |

c) Company

| Empno_no | city |
|---|---|

| Number | Varchar2 |
|---|---|
| 10 | 20 |

d) Manages

| Empno_no | Manager_name | Mang_no |
|---|---|---|
| Number | Varchar2 | Number |
| 10 | 20 | 10 |

**SOLUTION:**

```
SQL> create table employee (emp_no number(10) primary key,
employee_name varchar2(20),street varchar2(20),city varchar2(20));
Table created.
SQL> create table works (emp_no number(10) references employee,
company_name varchar2(20), joining_date date, designation
varchar2(20), salary number(10,2));
Table created.
SQL> create table company (emp_no number(10) references employee,
city varchar2(20));
Table created.
SQL> create table manages(emp_no number(10)references
employee,manager_name varchar2(20),mang_no number(20));
Table created.
```

**SQL> desc employee;**

```
Name Null? Type
----------------------------------------- -------- -----------------
EMP_NO NOT NULL NUMBER(10)
EMPLOYEE_NAME VARCHAR2(20)
STREET VARCHAR2(20)
CITY VARCHAR2(20)
Emp_no Employee_name Street City
Emp_no Company_name Joining_date Designation Salary
Emp_no City
Emp_no Manager_name Mang_no
```

**SQL> desc works;**

```
Name Null? Type
----------------------------------------- -------- ----------------------------
EMP_NO NUMBER(10)
COMPANY_NAME VARCHAR2(20)
JOININD_DATE DATE
DESIGNATION VARCHAR2(20)
SALARY NUMBER(10,2)
```

**SQL> desc manages;**

```
Name Null? Type
----------------------- --------------- --------
EMP_NO NUMBER(10)
MANAGER_NAME VARCHAR2(20)
MANG_NO NUMBER(20)
```

**SQL> desc company;**

```
Name Null? Type
--------------------------- --------------- --------
EMP_NO NUMBER(10)
CITY VARCHAR2(20)
```

```
SQL> create sequence emp_seq;
Sequence created.

SQL> insert into employee values(emp_seq.nextval,'rajesh','firstcross','gulbarga');
```

```
1 row created.
SQL> insert into employee values(emp_seq.nextval,'paramesh','secondcross','bidar');
1 row created.
SQL> insert into employee values(emp_seq.nextval,'pushpa','ghandhiroad','banglore');
1 row created.
SQL> insert into employee values(emp_seq.nextval,'vijaya','shivajinagar','manglore');
1 row created.
SQL> insert into employee values(emp_seq.nextval,'keerthi','anandsagar street','bijapur');
1 row created.

SQL> select * from employee;
EMP_NO EMPLOYEE_NAME STREET CITY
-------------- ------------------------- -------------------- ------------
2 paramesh second cross bidar
3 pushpa ghandhi road banglore
4 vijaya shivaji nagar manglore
5 keerthi anand sagar street bijapur

SQL> insert into works values(1,'abc','23-nov-2000','projectlead',40000);
1 row created.
SQL> insert into works values(2,'abc','25-dec-2010','softwareengg',20000);
1 row created.
SQL> insert into works values(3,'abc','15-jan-2011','softwareengg',19000);
1 row created.
SQL> insert into works values(4,'abc','19-jan-2011','softwareengg',19000);
1 row created.
SQL> insert into works values(5,'abc','06-feb-2011','softwareengg',18000);
1 row created.

SQL> select * from works;
EMP_NO COMPANY_NAME JOININD_D DESIGNATION SALARY
---------- -------------------- --------- ------------------- ------
1 abc 23-NOV-00 project lead 40000
2 abc 25-DEC-10 software engg 20000
3 abc 15-JAN-11 software engg 19000
4 abc 19-JAN-11 software engg 19000
5 abc 06-FEB-11 software engg 18000

SQL> insert into company values(1,'gulbarga');
1 row created.
SQL> insert into company values(2,'bidar');
1 row created.
SQL> insert into company values(3,'banglore');
1 row created.
SQL> insert into company values(4,'manglore');
1 row created.
SQL> insert into company values(5,'bijapur');
1 row created.

SQL> select * from company;
EMP_NO CITY
---------- --------------------
1 gulbarga
2 bidar
3 banglore
4 manglore
5 bijapur
SQL> insert into manages values(2,'rajesh',1);
1 row created.
SQL> insert into manages values(3,'rajesh',1);
1 row created.
SQL> insert into manages values(4,'rajesh',1);
```

```
1 row created.
SQL> insert into manages values(5,'rajesh',1);
1 row created.

SQL> select * from company;
EMP_NO CITY
---------- --------------------
1 gulbarga
2 bidar
3 banglore
4 manglore
5 bijapur

SQL> select * from manages;
EMP_NO MANAGER_NAME MANG_NO
---------- -------------------- ----------
2 rajesh 1
3 rajesh 1
4 rajesh 1
5 rajesh 1
```

**Program 4: create a table to store data with the following structure and constraints**

| Regno | Sname | Degree | Medium | Year |
|---|---|---|---|---|
| Number | Varchar2 | Varchar2 | `char | Number |
| 4 | 20 | 10 | 2 | 2 |

**Constraints:**

a. Register number column does not allow duplicates and null values
b. Student name shouldn't be blank
c. Degree should be 'bsc', 'bcom', 'ba'
d. Medium should be either 'e' or 't'
e. Year should be either 1,2, or 3

**Queries:**

1. Describe the structure of the table created above
2. Insert any 5 records through user interaction
3. List all the records
4. Display only student name and their degree
5. Display student details who are studying 'bsc'
6. Display student details who are studying 'bcom' with English medium
7. Display the student whose name begin with 's'
8.Displayname,year,degree who are studying in second year ba or bsc
9. Display the records of student who are studying in second and final year with 'ba' course
10. Give the student details of a given register number

```
SQL> create table std(regno number(5) primary key,
  2      name varchar2(20) not null,
  3      degree varchar2(10) check(degree in('bsc','bcom','ba')),
  4      medium char(2) check(medium in('e','t')),
  5     year number(1) check(year between 1 and 3));
Table created.
```

**1. Describe the structure of the table created above**

```
SQL>desc std
 Name                           Null?            Type
 ------------------------------ --------          ----
REGNO                           NOT NULL         NUMBER(5)
```

```
NAME                                      NOT NULL     VARCHAR2(20)
DEGREE                                                 VARCHAR2(10)
MEDIUM                                                 CHAR(2)
YEAR                                                   NUMBER(1)
```

**2. Insert any 5 records through user interaction**

```
SQL>insert into std values(100,'sahiti','bcom','e',3);
SQL>insert into std values(101,'supriya','bcom','e',3);
SQL>insert into std values(102,'kiranmayi','bsc','e',3);
SQL>insert into std values(103,'rohini','bsc','e',2);
SQL>insert into std values(104,'lohita','ba','t',3);
SQL>insert into std values(105,'koumudi','ba','t',2);
```

**3. List all the records**

```
SQL> select *from std;
REGNO NAME              DEGREE               ME      YEAR
--------- -------------------- ----------     --      ---------
    100 sahiti                  bcom          e       3
    101 supriya                 bcom          e       3
    102 kiranmayi               bsc           e       3
    103 rohini                  bsc           e       2
    104 lohita                  ba            t       3
    105 koumudi                 ba            t       2
6 rows selected.
```

**4. Display only student name and their degree**

```
SQL> select name,degree from std;
NAME              DEGREE
-------------------- ----------
sahiti               bcom
supriya              bcom
kiranmayi            bsc
rohini               bsc
lohita               ba
koumudi              ba
6 rows selected.
```

**5. Display student details who are studying 'bsc'**

```
SQL> select *from std where degree='bsc';
REGNO NAME              DEGREE     ME     YEAR
--------- -------------------- ----------   --       ---------
    102 kiranmayi               bsc     e          3
    103 rohini                  bsc     e          2
```

**6. Display student details who are studying 'bcom' with English medium**

```
SQL> select *from std where degree='bcom' and medium='e';
REGNO NAME              DEGREE     ME     YEAR
--------- -------------------- ----------    --       ---------
    100 sahiti                  bcom      e        3
    101 supriya                 bcom      e        3
```

**7. Display the student whose name begin with 's'**

```
SQL> select *from std where name like 's%';
REGNO NAME              DEGREE     ME     YEAR
--------- -------------------- ----------    --       ---------
    100 sahiti                  bcom    e        3
    101 supriya                 bcom    e        3
```

**8. Display name,year,degree who are studying in second year ba or bsc**

```
SQL> select name,year,degree from std where degree='bsc' or degree='ba' and year=2 order by name;
NAME                 YEAR DEGREE
```

```
-------------------- --------- ----------
kiranmayi            3         bsc
koumudi              2         ba
rohini               2         bsc
```

**9. Display the records of student who are studying in second and final year with 'ba' course**

```
SQL> select *from std where year in(2,3) and degree='ba';
REGNO NAME              DEGREE     ME      YEAR
--------- -------------------- ----------    --       ---------
    104 lohita                  ba       t        3
    105 koumudi                 ba       t        2
```

**10. Give the student details of a given register number**

```
SQL> select *from std where regno=&regno;
Enter value for regno: 100
old   1: select *from std where regno=&regno
new   1: select *from std where regno=100
REGNO NAME              DEGREE     ME      YEAR
--------- -------------------- ----------    --       ---------
    100 sahiti                  bcom      e        3
SQL> /
Enter value for regno: 105
old   1: select *from std where regno=&regno
new   1: select *from std where regno=105
REGNO NAME              DEGREE     ME      YEAR
--------- -------------------- ----------    --       ---------
    105 koumudi                 ba       t        2
```

**Program 5: create a table to store data with the following structure and constraints**

| rollno | Name | S1 | S2 | S3 | S4 | S5 | Tot | Avg | Res | Grade |
|---|---|---|---|---|---|---|---|---|---|---|
| Number | Varchar2 | number | Number | number | number | number | number | number | Char | Varchar2 |
| 5 | 20 | 3 | 3 | 3 | 3 | 3 | 5 | 5,2 | 10 | 20 |

**Constraints:**

a. Register number column does not allow duplicates and null values

b. Student name shouldn't be blank

**Queries:**

1.Insert 10 records in the above created table only in rollno,name,s1,s2,s3,s4,s5 columns

2. Calculate marks total and update in tot column

3. Calculate average marks and update in avg column

4. Update res column as 'pass' if the students has score>=35 in all subjects, other wise 'fail'

5.Update the grade column as per the following conditions

| | | |
|---|---|---|
| Distinction | - | should be pass and avg>=75 |
| First Class | - | should be pass and avg>=60 and avg<75 |
| Second Class | - | should be pass and avg>=50 and avg<60 |
| Third Class | - | should be pass and avg>=35 and avg<50 |
| Fail | - | if the res is fail |

6. Display all the records of marks table

7. Remove the record of a student whose rollno is 45

8. Make a duplication of table as stumarks

9. Make empty grade column in stumarks table

10. Change the table name stumarks as marks_data

```
SQL> create table std_marks(rollno number(5) primary key,
```

```
2  name varchar2(20) not null,
3  s1 number(3),
4  s2 number(3),
5  s3 number(3),
6  s4 number(3),
7  s5 number(3),
8  tot number(5),
9  avg number(5,2),
10  res char(10),
11  grade varchar2(20));
 Table created.
```

**1.Insert 10 records in the above created table only in rollno,name,s1,s2,s3,s4,s5 columns**

```
SQL>insert into std_marks(rollno,name,s1,s2,s3,s4,s5) values(100,'shyam',99,85,87,66,32);
1 row created.
SQL> insert into std_marks(rollno,name,s1,s2,s3,s4,s5) values(101,'sahiti',88,78,99,85,100)
1 row created.
SQL> insert into std_marks(rollno,name,s1,s2,s3,s4,s5) values(102,'sahitya',66,69,65,62,63)
1 row created.
SQL> insert into std_marks(rollno,name,s1,s2,s3,s4,s5) values(103,'hemanth',56,58,59,57,53)
1 row created.
SQL> insert into std_marks(rollno,name,s1,s2,s3,s4,s5) values(104,'taruni',48,49,35,49,47)
1 row created.
```

**2. Display all the records of marks table**

```
SQL> select *from std_marks;
ROLLNO NAME                 S1   S2   S3   S4   S5  TOTAVG RES       GRADE
------ -------------------- ---- ---- ---- ---- ----  ---- ---- ---------- --------------------
   100 shyam                99   85   87   66   32
   101 sahiti               88   78   99   85  100
   102 sahitya              66   69   65   62   63
   103 hemanth              56   58   59   57   53
   104 taruni               48   49   35   49   47
5 rows selected.
```

**3. Calculate marks total and update in tot column**

```
SQL> update std_marks set tot=s1+s2+s3+s3+s5;
5 rows updated.
SQL> select *from std_marks;
ROLLNO NAME                 S1   S2   S3   S4   S5  TOTAVG RES       GRADE
------ -------------------- ---- ---- ---- ---- ----  ---- ---- ---------- --------------------
   100 shyam                99   85   87   66   32   390
   101 sahiti               88   78   99   85  100   464
   102 sahitya              66   69   65   62   63   328
   103 hemanth              56   58   59   57   53   285
   104 taruni               48   49   35   49   47   214
5 rows selected.
```

**4. Calculate average marks and update in avg column**

```
SQL> update std_marks set avg=tot/5;
5 rows updated.
SQL> select *from std_marks;
ROLLNO NAME                 S1   S2   S3   S4   S5  TOT      AVG RES       GRADE
------ -------------------- ---- ---- ---- ---- ---- ----      ---- ---------- --------------------
   100 shyam                99   85   87   66   32  390      78
   101 sahiti               88   78   99   85  100  464      92.8
   102 sahitya              66   69   65   62   63  328      65.6
   103 hemanth              56   58   59   57   53  285      57
```

```
  104 taruni                    48   49   35   49   47  214      42.8
5 rows selected.
```

**5. Update res column as 'pass' if the students has scor>=35 in all subjects other wise 'fail'**

```
SQL> update std_marks set res='pass' where s1>=35 and s2>=35 and s3>=35 and s4>=35 and s5>=35;
4 rows updated.
SQL> select *from std_marks;
ROLLNO NAME                   S1   S2   S3   S4   S5  TOT      AVG    RES       GRADE
------ --------------------   ---- ---- ---- ---- ---- ----     ----   ---------- --------------------
  100 shyam                   99   85   87   66   32  390      78
  101 sahiti                  88   78   99   85  100  464      92.8    pass
  102 sahitya                 66   69   65   62   63  328      65.6   pass
  103 hemanth                 56   58   59   57   53  285      57     pass
  104 taruni                  48   49   35   49   47  214      42.8   pass
5 rows selected.
SQL> update std_marks set res='fail' where s1<35 or s2<35 or s3<35 or s4<35 or s5<35;
1 row updated.
SQL> select *from std_marks ;
ROLLNO NAME                   S1   S2   S3   S4   S5  TOT      AVG    RES       GRADE
------ --------------------   ---- ---- ---- ---- ---- ----     ----   ---------- --------------------
  100 shyam                   99   85   87   66   32  390      78     fail
  101 sahiti                  88   78   99   85  100  464      92.8   pass
  102 sahitya                 66   69   65   62   63  328      65.6   pass
  103 hemanth                 56   58   59   57   53  285       57    pass
  104 taruni                  48   49   35   49   47  214      42.8   pass
5 rows selected.
```

**5.Update the grade column as per the following conditions**

Distinction-should be pass and avg>=75, First Class-should be pass and avg>=60 and avg<75,Second Class-should be pass and avg>=50 and avg<60,Third Class-should be pass and avg>=35 and avg<50,Fail -if the res is fail

```
SQL> update std_marks set grade='distinctin' where res='pass' and avg>=75;
1 row updated.
SQL> select *from std_marks;
ROLLNO NAME                   S1   S2   S3   S4   S5  TOT      AVG    RES       GRADE
------ --------------------   ---- ---- ---- ---- ---- ----     ----   ---------- --------------------
  100 shyam                   99   85   87   66   32  390      78     fail
  101 sahiti                  88   78   99   85  100  464      92.8   pass      distinction
  102 sahitya                 66   69   65   62   63  328      65.6   pass
  103 hemanth                 56   58   59   57   53  285      57     pass
  104 taruni                  48   49   35   49   47  214      42.8   pass
SQL> update std_marks set grade='firstclass' where res='pass' and avg>=60 and avg<75;
1 row updated.
SQL> select *from std_marks;
ROLLNO NAME                   S1   S2   S3   S4   S5  TOT      AVG    RES       GRADE
------ --------------------   ---- ---- ---- ---- ---- ----     ----   ---------- --------------------
  100 shyam                   99   85   87   66   32  390      78     fail
  101 sahiti                  88   78   99   85  100  464      92.8   pass      distinction
  102 sahitya                 66   69   65   62   63  328      65.6   pass      firstclass
  103 hemanth                 56   58   59   57   53  285      57     pass
  104 taruni                  48   49   35   49   47  214      42.8   pass
SQL> update std_marks set grade='secondclass' where res='pass' and avg>=50 and avg<60;
1 row updated.
SQL> select *from std_marks;
ROLLNO NAME                   S1   S2   S3   S4   S5  TOT      AVG    RES       GRADE
------ --------------------   ---- ---- ---- ---- ---- ----     ----   ---------- --------------------
```

```
  100 shyam                 99  85  87  66  32  390     78     fail
  101 sahiti                88  78  99  85  100 464     92.8   pass     distinctin
  102 sahitya               66  69  65  62  63  328     65.6   pass     firstclass
  103 hemanth               56  58  59  57  53  285     57     pass     secondclass
  104 taruni                48  49  35  49  47  214     42.8   pass
SQL> update std_marks set grade='thirdclass' where res='pass' and avg>=35 and avg<50;
1 row updated.
SQL> select *from std_marks;
ROLLNO NAME                S1  S2  S3  S4  S5  TOT     AVG    RES      GRADE
------ ------------------- ---- ---- ---- ---- ---- ----    ----   ---------- --------------------
  100 shyam                 99  85  87  66  32  390     78     fail
  101 sahiti                88  78  99  85  100 464     92.8   pass     distinction
  102 sahitya               66  69  65  62  63  328     65.6   pass     firstclass
  103 hemanth               56  58  59  57  53  285     57     pass     secondclass
  104 taruni                48  49  35  49  47  214     42.8   pass     thirdclass
SQL> update std_marks set grade='fail' where res='fail';
1 row updated.
SQL> select *from std_marks;
ROLLNO NAME                S1  S2  S3  S4  S5  TOT     AVG    RES            GRADE
------ ------------------- ---- ---- ---- ---- ---- ----    ---   ---------- --------------------
  100 shyam                 99  85  87  66  32  390     78     fail           fail
  101 sahiti                88  78  99  85  100 464     92.8   pass           distinction
  102 sahitya               66  69  65  62  63  328     65.6   pass           firstclass
  103 hemanth               56  58  59  57  53  285     57 p   ass            secondclass
  104 taruni                48  49  35  49  47  214     42.8   pass           thirdclass
```

**6. Display all the records of std_marks table**

```
SQL> select *from std_marks;
ROLLNO NAME                S1  S2  S3  S4  S5  TOT     AVG     RES            GRADE
------ ------------------- ---- ---- ---- ---- ---- ------    ----   ---------- --------------------
  100 shyam                 99  85  87  66  32  390     78     fail           fail
  101 sahiti                88  78  99  85  100 464     92.8   pass           distinction
  102 sahitya               66  69  65  62  63  328     65.6   pass           firstclass
  103 hemanth               56  58  59  57  53  285     57     pass           secondclass
  104 taruni                48  49  35  49  47  214     42.8   pass           thirdclass
```

**7. Remove the record of a student whose rollno is 45**

```
SQL> delete from std_marks where rollno=45;
0 rows deleted.
```

**8. Make a duplication of table as stumarks**

```
SQL> create table stu_marks as select *from std_marks;
Table created.
```

**9. Make empty grade column in stu_marks table**

```
SQL> update stu_marks set grade=null;
5 rows updated.
SQL> select *from stu_marks;
ROLLNO NAME                S1  S2  S3  S4  S5  TOT     AVG     RES       GRADE
------ ------------------- ---- ---- ---- ---- -------- -- -----    --------------------
  100 shyam                 99  85  87  66  32   390     78     fail
  101 sahiti                88  78  99  85  100  464     92.8   pass
  102 sahitya               66  69  65  62  63   328     65.6   pass
  103 hemanth               56  58  59  57  53   285     57     pass
  104 taruni                48  49  35  49  47   214     42.8   pass
```

**10.Change the table name stumarks as marks_data**

```
SQL> rename std_marks to marks_data;
Table renamed.
```

SQL> select *from marks_data;

| ROLLNO | NAME | S1 | S2 | S3 | S4 | S5 | TOT | AVGRES | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 100 | shyam | 99 | 85 | 87 | 66 | 32 | 390 | 78 | fail | fail |
| 101 | sahiti | 88 | 78 | 99 | 85 | 100 | 464 | 92.8 | pass | distinctin |
| 102 | sahitya | 66 | 69 | 65 | 62 | 63 | 328 | 65.6 | pass | firstclass |
| 103 | hemanth | 56 | 58 | 59 | 57 | 53 | 285 | 57 | pass | secondclass |
| 104 | taruni | 48 | 49 | 35 | 49 | 47 | 214 | 42.8 | pass | thirdclass |

**Program 6: create a table to store data with the following structure and constraints**

Customer

| Cust_id | Name | age | Address | Salary |
|---|---|---|---|---|
| Number | Varchar2 | Number | Varchar2 | Number |
| 5 | 30 | 3 | 30 | 10,2 |

**SOLUTION:**

```
SQL> Create table customer
(cust_id number(5),
Name varchar2(30),
Age number(3),
Address varchar2(20),
Salary nuber(10,2));
SQL> table created
```

**Query:** insert any five records into above created value

```
SQL> insert into customer values(1,'Akshay',25,'Delhi',30000);
insert into customer values(2,'Manish',27,'Mumbai',35000);
insert into customer values(3,'Kushagra',26,'Kolkata',30000);
insert into customer values(4,'Mukesh',31,'Hyderabad',32000);
insert into customer values(5,'Himanshu',29,'Chennai',40000)
insert into customer values(6,'Neeraj',30,'Noida',36000);
insert into customer values(7,'Nishant',32,'Delhi',30000);
```

**Query: display all records from customer table**

SQL> select * from customer;

| ID | NAME | AGE | ADDRESS | SALARY |
|---|---|---|---|---|
| 1 | Akshay | 25 | Delhi | 30000 |
| 2 | Manish | 27 | Mumbai | 35000 |
| 3 | Kushagra | 26 | Kolkata | 30000 |
| 4 | Mukesh | 31 | Hyderabad | 32000 |
| 5 | Himanshu | 29 | Chennai | 40000 |
| 6 | Neeraj | 30 | Noida | 36000 |
| 7 | Nishant | 32 | Delhi | 30000 |

**Q1. Write a query to find the salary of a person where age is <= 26 and salary >= 25000 from customer table.**

SQL>SELECT * FROM CUSTOMERS WHERE AGE <= 26 AND SALARY >= 25000;

**Output:**

| ID | NAME | AGE | ADDRESS | SALARY |
|---|---|---|---|---|
| 1 | Akshay | 25 | Delhi | 30000 |
| 2 | Kushagra | 26 | Kolkata | 30000 |

**Q2. Write a query to find the salary of a person where age is <= 26 or salary > =33000 from customer table.**

SQL>SELECT * FROM CUSTOMERS WHERE AGE <= 26 or SALARY > =33000;

**Output:**

Q3.Write a query to find the name of customer whose name is like “Ku%”.
SQL>SELECT * FROM CUSTOMERS WHERE NAME LIKE 'Ku%';
**Output:**

Q4. Write a query to find the customer details using “IN” and “Between” operator where age can be 25 or 27.
SQL>SELECT * FROM CUSTOMERS WHERE AGE IN ( 25, 27 ); (or)
SQL>SELECT * FROM CUSTOMERS WHERE AGE BETWEEN 25 AND 27;
**Output:**

**Program 7: create a table to store data with the following structure and constraints**

| Product ID | Name | Description | Price | colour |
|---|---|---|---|---|
| Number | Varchar2 | Varchar2 | Number | Number |
| 5 | 30 | 30 | 5,2 | 3 |

Solution:
SQL> create table product
(product_id number(3),
Name varchar2(30),
Description varchar2(30),
Price number(5,2),
Colour number(3));

**Query: insert any five records into customer table**
SQL> insert into product values(10,' Printer',' Inkjet 300 colour Printer', 120, 80);
insert into product values(11,' Printer ','1220XI Inkjet Printer', 200,130);
insert into product values(12, 'Printer',' Photo 890 Inkjet Printer', 250 ,200);
insert into product values(13 ,'Printer', 'Photo 890 Inkjet Printer', 300, 270);
Query: display all records from product table
SQL> select *from product;

| product_id | name | description | price | colour |
|---|---|---|---|---|
| 10 | Printer | Inkjet 300 colour Printer | 120 | 80 |
| 11 | Printer | 1220XI Inkjet Printer | 200 | 130 |
| 12 | Printer | Photo 890 Inkjet Printer | 250 | 200 |
| 13 | Printer | Photo 890 Inkjet Printer | 300 | 270 |

**Query1. Write a query find the total price of the product.**
SQL>SELECT sum(price) FROM product;
**Output:**

**Query2. Write a query find the average price of the product.**
SQL>SELECT avg(price) FROM product;
**Output:**

**Query3. Write a query find the max price of the product.**
SELECT max(price) FROM product;
Output:

**Query4. Write a query find the min price of the product.**
SELECT min(price) FROM product;
Output:

**Query5: Write a query to find the total number of products**
Select count(product_id) from product;
**Output:**

**Program 8. Create the tables by using following constraints for a banking enterprise**

BRANCH (branch_name: string, branch_city: string, assets: real)
ACCOUNT (accno: int, branch_name: string, balance: real)
CUSTOMER (customer_name: string, customer_street: string, city:string)
DEPOSITOR (customer_name: string, accno: int)
LOAN (loan_number: int, branch_name: string, amount: real)
BORROWER (customer_name: string, loan_number: int)

i)Create the above tables by properly specifying the primary keys and the foreign keys.
ii)Enter atleast five tuples for each relation.
iii)Find all the customers who atleast two accounts at the MAIN branch.
iv)Find all the customers who have an account at all branches located in a specific city.
v)Demonstrate how you delete all account tuples at every branch located in a specific city.
vi)Generation of suitable reports.
vii)Create suitable front end for querying and displaying the results.

**Solution:..**

i)Create the above tables by properly specifying the primary keys and the foreign keys.

```
create table branch(br_name varchar2(30) primary key,
br_city varchar2(30),
assets number(10,2));

create table account(acc_no number(4) primary key,
br_name references branch,
balance number(10,2));

create table customers(c_name varchar2(30) primary key,
c_street varchar2(30),
c_city varchar(30));

create table depositor(c_name references customers,
acc_no references account,
qty number);

create table loan(loan_no number(4) primary key,
br_name references branch,
amt number(10,2));

create table borrower(c_name references customers,
loan_no references loan);
```

ii) Enter atleast five tuples for each relation.

```
insert into branch values('rajaji nagar','bangalore',1000000);
insert into branch values('jayanagar','bangalore',50000);
insert into branch values('mvit','bangalore',10000);
insert into branch values('jawahar nagar','bangalore',100000);
insert into branch values('rajbhavan','bangalore',23566);

insert into account values(1000,'rajaji nagar',2500);
insert into account values(2000,'rajaji nagar',8996);
insert into account values(3000,'rajaji nagar',7415);
insert into account values(4000,'jayanagar',2121);
insert into account values(5000,'mvit',8596);
insert into account values(6000,'jawahar nagar',9999);
insert into account values(7000,'rajbhavan',235);
```

```
insert into customers values('prasanna','patel road','raichur');
insert into customers values('harish','indiranagar','bangalore');
insert into customers values('sunil','ring road','bangalore');
insert into customers values('srinivas','woc road','bangalore');
insert into customers values('rudre','maruti galli','belgaum');

insert into depositor values('prasanna',1000,2000);
insert into depositor values('prasanna',2000,3000);
insert into depositor values('harish',3000,5000);
insert into depositor values('sunil',4000,1520);
insert into depositor values('srinivas',5000,1120);
insert into depositor values('rudre',6000,1250);
insert into depositor values('prasanna',7000,1250);

insert into loan values(100,'rajaji nagar',5000);
insert into loan values(200,'rajaji nagar',4000);
insert into loan values(300,'jayanagar',6323);
insert into loan values(400,'mvit',4512);
insert into loan values(500,'jawahar nagar',1235);
insert into loan values(600,'rajbhavan',9632);
insert into loan values(700,'rajbhavan',3456);

insert into borrower values('prasanna',100);
insert into borrower values('harish',200);
insert into borrower values('sunil',300);
insert into borrower values('srinivas',400);
insert into borrower values('rudre',500);
```

iii)Find all the customers who atleast two accounts at the MAIN branch.

```
SQL>Select c_name from depositor d,account a where
        a.acc_no = d.acc_no and
        a.br_name = 'rajaji nagar'
        group by c_name
        having count(*) > 1;
```

iv)Find all the customers who have an account at all branches located in a specific city.

```
SQL> select c_name from customers c where not exists
        (select br_name from branch where br_city='bangalore' minus select br_name from
        depositor d,acc
        where d.acc_no=a.acc_no and d.c_name=c.c_name)
        and exists
        (select br_name from branch where br_city='bangalore');
```

v)Demonstrate how you delete all account tuples at every branch located in a specific city.

```
SQL> Delete from depositor where acc_no in
        (select acc_no from account where br_name in
        (select br_name from branch where br_city = 'bangalore));
SQL> Delete from account where br_name in
        (select br_name from branch where br_city = 'bangalore);
```

**Program 9. Create the table as per following specifications.**

| Deptno | Dname | Loc |
|---|---|---|
| Number | Varchar2 | Varchar2 |
| 3 | 15 | 15 |

Create table deptt
(deptno number(3) primary key,
Dname varchar2(15),
Loc varchar2(15));

| Empno | Ename | Job | Sal | Mgr | Comm | Hiredate | Deptno |
|---|---|---|---|---|---|---|---|
| Number | Varchar2 | Varchar2 | Number | Number | number | Date | Number |
| 4 | 15 | 15 | 10,2 | 5 | 5 | | 3 |

Create table empp
(empno number(4) primary key,
Ename varchar2(15),
Job varchar2(15),
Sal number(10,2),
Mgr number(5),
Comm number(5),
Hiredate date,
Deptno number(3),
Foreign key(deptno) references dept(deptno));

**Query: insert the data into the above created tables**

Insert into deptt values(10,'accounting','delhi');
Insert into deptt values(20,'research','hyderabad');
Insert into deptt values(30,'sales','bangalore');
Insert into deptt values(40,'operations','chennai');

Insert into empp values(1,'ramesh','clerk',5000,2,100,'20-mar-2018',10);
Insert into empp values(2,'priya','analyst',15000,18,200,'12-dec-2016',20);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(3,'mohini','clerk',6000,15,'06-jun-2019',30);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(4,'ramya','saleman',6500,6,'06-jun-2017',40);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(5,'raja','saleman',8500,12,'06-aug-2017',20);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(6,'amani','manager',10500,7,'06-nov-2018',20);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(7,'sowmya','president',19500,10,'26-sep-2014',20);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(8,'kavya','salesman',9500,4,'13-feb-2018',40);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(9,'kowsalya','clerk',7500,16,'18-may-2018',40);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(10,'rajan','clerk',6500,5,'12-oct-2016',30);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(11'rajani','salesman',9500,5,'26-may-2017',40);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(12,'nalini','manager',22500,10,'05-sep-2014',20);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(13,'kesav','analyst',13500,9,'15-sep-2014',30);
Insert into empp(empno,ename,job,sal,mgr,hiredate,deptno) values(14,'naveena','clerk',9500,1,'16-mar-2014',20);

**Query: display all the information from empp table**
Select *from empp;

**Query: display all the information from deptt table**
Select *from deptt;

**Query:Display unique jobs from empp table**
Select distinct job from empp;

**Query: List the details of employees in the ascending order of their salaries**
Select *from empp order by sal asc;

**Query:List the detils of the employees in ascending order of the department numbers and descending order of jobs**
Select *from empp order by deptno asc, job desc;

**Query:Display all unique job groups in the descending order**
Select distinct job from empp order by job desc;

**Query: List the details of the employee who joined before 2016**
Select *from empp where hiredate < (01-jan-2016');

**Query: display all the details of all 'mgrs'**
Select *from empp where empno in(select mgr from empp);

**Query: display the empno,ename,job,hiredate,experience of all mgrs.**
Select empno,ename,job,hiredate,months_between(sysdate,hiredate) as "experience" from empp where empno in(select mgr from empp);

**Query: Display all the details of the employees who are not getting any commission**
Select *from empp where comm is null;

**Query: display all the details of the employees whose commission is more than their salary**
Select *from empp where comm > sal;

**Query: display the details of employee whose daily salary is more than Rs.100**
Select *from empp where (sal/30) > 100;

**Query: List the details of the employee who are either 'clerk' or 'analyst' in the descending order**
Select *from empp where job='clerk' or job='analyst' order by job desc;

**Query: List the details of the employee who joined on 20-mar-2018,12-dec-2016,06-aug-2017,26-sep-2014 in ascending order of seniority**
Select *from empp where hirdate in ('20-mar-2018','12-dec-2016','06-aug-2017','26-sep-2014') order by hiredate asc;

**Query: list the details of the employee who are working for the department number 10 or 20**
Select *from empp where deptno=10 or deptno=20;

## **Program10.** *Display the data of empp and deptt tables using queries*

**Query: list the details of the employee who are joined in the year 2016**
Select *from empp where hiredae between '01-jan-16' and '31-dec'16';

**Query: List the details of the employees who are joined in the month of February in 2016**
Select * from empp where hiredate between '01-feb'16' and '28-feb-16';

**Query: List the details of the employee whose annual salary ranging from 22000 from 45000**
Select * from empp where 12*sal between 22000 and 45000;

**Query: List the employee names having five characters in their name**
Select ename from empp where length(ename)=5;

**Query: List the employee names starting with 's' and with five characters**
Select ename from empp where ename like 's%' and length(ename)=5;

**Query:list the employee names having four characters and third character must be 'r'**
Select *from empp where length(ename)=4 and ename like '_ _ _r%';

**Query:List the five character name starting with 's' and ending with 'h'**
Select *from emp where length(ename)=5 and ename like 's%h';

**Query: list the details of the employee who joined in January**
Select *from empp where to_char(hiredate,'mon')='jan';

**Query:list the details of the employees who joined in the month of which second character is 'a'**
Select *from emp where to_char(hiredate,'mon') like '_ a%';

**Query: list the details of the employees whose salary is four digit numbers and ending with zero**
Select *from empp where length(sal)=4 and sal like '%0';

**Query: list the details of the employees whose names having a character set 'LL' together**
Select *from empp where ename like 'll%';

**Query:list the details of the employees that do not belong to department number 20**
Select *from empp where deptno not in (20);

**Query:list the details of all employees except 'president' & 'manager' in ascending order of salaries**
Select *from empp where job not in ('president','manager') order by sal desc;

**Query:list the details of the employees who joined before or after 2015**
Select *from empp where to_char(hiredate,'yyyy') not in (2015);

**Query:list the details of the employee whose employee number start with 1**
Select *from empp where empno not like '1%';

## **Program** *11. display the data from empp, deptt tables by using queries*

**Query: display all designations in department 20 and 30 of empp table without remaining any duplicate values**
Select job from empp where deptno=20
Union all
Select job from empp where deptno=30;

**Query:list the jobs common to department 20 and 30**
Select job from empp where deptno=20
Intersect
Select job from empp where deptno=30;

**Query:list jobs of department number 10 those are not found in department number 20**
Select job from empp where deptno=10
Minus
Select job from empp where deptno=20;

**Query:list the maximum salary of empp table**
Select max(sal) from empp;

**Query: list the minimum salary of empp table**
Select min(sal) from empp;

**Query: list the total salary of empp table**
Select sum(sal) from empp;

**Query: find the details of highest paid employees**
Select *from empp where sal in (select max(sal) from empp);

**Query:find the details of employee who is getting second highest salaty**

Select *from empp where sal=(select max(sal) from empp where sal<(select max(sal) from empp);

**Query:find the highest paid employee of sales department**
Select *from empp where sal aain(select max(sal) from empp where deptno in(select deptno from deptt where dname='sales'));
**Query: find the details of lowest paid employee**
Select *from empp where sal in(select min(sal) from empp);

**Query:find the details of employee who is getting second lowest salary**
Select *from empp where sal=(select min(sa) from empp where sal>(select min(sal) from empp));

**Query:list the employees who are senior to most recently hired employee working under kavya**
Select *from empp where hiredate < (select max(hiredate) from empp where mgr in (select empno from empp where ename='kavya');

**Query:List the managers of department 10 or 20**
Select *from empp where job='manager' and (deptno=10 or deptno=20);

**Query:List the details of employees working at Hyderabad**
Select *from empp where deptno in(select deptno from deptt where loc='hyderabad');

**Query:list the alternate records (even numbered) from empp table**
Select *from empp where rowed in (select decode(mod(rownum,2),0,rowid,null) from empp);

**Query:list the employee names along with their salaries from empp table whose salary is greatesr than the highest salary of an employee belonging to department number 20**
Select ename,sal from empp where
Sal>all(select sal from empp where deptno=20);

**Query:list the employee details whose jobs are same as kavya or salary is more than naveena**
Select *from empp where job=(select job from empp where ename='kavya') or
sal > (select sal from empp where ename='naveena');

**Query:list the details of employees who are working at Chennai**
Select *from emp where deptno in(select deptno from deptt where loc='chennai');

**Query: list the employees details joined January with salary ranging from 4000 to 8000**
Select *from emp where to_char(hiredate,'mon')='jan' and sal between 4000 and 8000;

**Query:list the name of the employees who are getting the highest salary department wise**
Select ename,sal,deptno from empp where sal in(select max(sal) from empp group by deptno);

**Program 12.Design a table cricket as per the given structure to store test match scores of Indian tem Australia tour**
batsman varchar2(20)
testno number(3)
innings_no number(1)
score number(3)
fours number(2)
sixes number(2)
**Constraints:**
a.Batsman and testno should not be empty
b. Innings_no default value as 1
c. Fours default value as zero
**Queries:**
1.Insert any 5 records into the above created table
2. Display test wise total scores

3. Display total scores testwise and innings_no wise
4. Display batsman,avgscore by individual
5. Display total score, highest score and lowest score of sachin in the series
6.Display total score of batsman who scored more than 200 runs in the series
7. Find the batsman with highest score of the series

```
SQL> create table cricket(batsman varchar2(20) not null,
2  testno number(3)not null,
3  innings_no number(1)default 1,
4  score number(3),
5  fours number(2)default 0,
6  sixes number(2));
 Table created.
SQL>desc cricket;
 Name                           Null?           Type
 ------------------------------ --------        ----
 BATSMAN                         NOT NULL       VARCHAR2(20)
 TESTNO                         NOT NULL        NUMBER(3)
 INNINGS_NO                                     NUMBER(1)
 SCORE                                          NUMBER(3)
 FOURS                                          NUMBER(2)
 SIXES                                          NUMBER(2)
SQL> insert into cricket values('sachin',100,1,250,10,3)
1 row created.
SQL> insert into cricket values('sachin',100,2,100,3,2)
1 row created.
SQL>insert into cricket values('kohli',100,1,75,4,0)
1 row created.
SQL>insert into cricket values('kohli',100,2,55,1,0)
1 row created.
SQL>insert into cricket values('dhoni',101,1,65,5,2)
1 row created.
SQL>insert into cricket values('dhoni',101,2,85,2,0)
1 row created.
SQL>insert into cricket values('dhawan',101,1,125,5,3)
1 row created.
SQL>insert into cricket values('dhawan',101,2,45,2,1)
1 row created.
SQL> select *from cricket;
BATSMAN              TESTNOINNINGS_NO SCORE FOURS SIXES
-------------------- ------ ---------- ----- ----- -----
Sachin               100             1   250    10     3
Sachin               100             2   100     3     2
Kohli                100             1    75     4     0
Kohli                100             2    55     1     0
Dhoni                101             1    65     5     2
Dhoni                101             2    85     2     0
Dhawan               101             1   125     5     3
Dhawan               101             2    45     2     1
8 rows selected.
```

**2. Display test wise total scores**

```
SQL> select testno,sum(score) from cricket group by testno;
TESTNOSUM(SCORE)
------ ----------
```

```
  100       480
  101       320
```

**3. Display total scores testwise and innings_no wise**

```
SQL> select testno,innings_no,sum(score) from cricket group by testno,innings_no;
TESTNOINNINGS_NOSUM(SCORE)
------ ----------          ----------
  100         1             325
  100         2             155
  101         1             190
  101         2             130
```

**4. Display batsman,avgscore by individual**

```
SQL> select batsman,avg(score) from cricket group by batsman;

BATSMAN            AVG(SCORE)
------------------- ----------
dhawan                    85
dhoni                     75
kohli                     65
sachin                    175
```

**5. Display total score, highest score and lowest score of sachin in the series**

```
SQL> select batsman,sum(score) "totalscore",max(score) "highestscore",min(score) "lowestscore"
2  from cricket group by batsman having batsman='sachin';
BATSMAN            totalscore   highestscore    lowestscore
------------------- ----------     ------------      -----------
sachin                  350         250             100
```

**6.Display total score of batsman who scored more than 200 runs in the series**

```
  SQL>selectbatsman,sum(score) from cricket group by batsman having sum(score)>200;
BATSMAN            SUM(SCORE)
------------------- ----------
sachin                  350
```

**7. Find the batsman with highest score of the series**

```
SQL> select batsman,score from cricket where score=(select max(score) from cricket);
BATSMAN                SCORE
--------------------       -----
sachin                    250
```



## PL/SQL PROGRAMS

**Program 1. Write a PL/SQL program to read two numbers and display maximum value**

**SOLUTION:**

```
declare
b number;
c number;
begin
b:=10;
c:=20;
if(c>b) then
dbms_output.put_line('C is maximum');
end if;
end;
/
```

**OUTPUT:**
C is maximum
PL/SQL procedure successfully completed.

**Program 2.Write a PL/SQL program read three numbers and display maximum value**

**SOLUTION:**

```
SQL> declare
 a number;
b number;
c number;
d number;
begin
a:=&a;
b:=&b;
 c:=&b;
if(a>b)and(a>c) then
dbms_output.put_line('A is maximum');
 elsif(b>a)and(b>c)then
dbms_output.put_line('B is maximum');
else
dbms_output.put_line('C is maximum');
end if;
end;
 /

Enter value for a: 21
old 7: a:=&a;
new 7: a:=21;
Enter value for b: 12
old 8: b:=&b;
new 8: b:=12;
Enter value for b: 45
old 9: c:=&b;
new 9: c:=45;
```

**OUTPUT:**
C is maximum
PL/SQL procedure successfully completed.

**Program 3. Write a PL/SQL program display sum of odd numbers**

**SOLUTION:**

```
SQL> declare
n number;
sum1 number default 0;
endvalue number;
begin
endvalue:=&endvalue;
n:=1;
for n in 1..endvalue
loop
 if mod(n,2)=1
then
sum1:=sum1+n;
end if;
```

```
 end loop;
dbms_output.put_line('sum ='||sum1);
end;
 /

Enter value for endvalue: 4
old 6: endvalue:=&endvalue;
new 6: endvalue:=4;
```

**OUTPUT:**

```
sum =4
PL/SQL procedure successfully completed.
```

**Program 4. Write a program to convert foreign heat temperature into centigrade temperature**

```
declare
f number (8,2):=&f;
c number (8,2);
zero_error exception;
begin
if f<=0 then
raise zero_error;
end if;
c:=(f-32)*5/9;
dbms_output.put_line('celcioustemparature is:'||c);
exception
when zero_error then
dbms_output.put_line('value must be greater than zero');
end;
```

**Output:**

```
Enter value for f: 104
old   2: f number(8,2):=&f;
new   2: f number(8,2):=104;
celcioustemparature is:40
```

**Program 5 .Create a pl/sql block that will accept an account number from user and debit an amount of Rs.2000/-from the account, if the account has a minimum balance of Rs.500/- after the amount is debited.**

**Table: accounts (accountno,name,balance)**

```
SQL> create table account(accno number(10),name varchar2(20),balance number(10,2));
Table created.
SQL> insert into account values(100,'kavitha',30000)
1 row created.
SQL> insert into account values(101,'raji',65000)
1 row created.
SQL> insert into account values(102,'kumar',200)
1 row created.
SQL> commit;
Commit complete.
SQL> select *from account;
ACCNO NAME            BALANCE
--------- -------------------- -----------------
 100 kavitha             30000
 101 raji                65000
 102 kumar               200
```

```
SQL> declare
2 v_accno number(10):=&v_accno;
3 v_balance number(10,2):=500.00;
4 v_debt constant number(10,2):=2000;
5 begin
6 select balance into v_balance from account where accno=v_accno;
7 if v_balance>500 then
8 update account set balance=balance-2000 where accno=v_accno;
9 else
10 dbms_output.put_line('insufficient balance');
11 end if;
12 exception
13when no_data_found then
14 dbms_output.put_line(‘there is no account for this number’);
15 end;
/
SQL> Enter value for v_accno: 102
old   2: v_accno number(10):=&v_accno;
new   2: v_accno number(10):=102;
insufficient balance
PL/SQL procedure successfully completed.
SQL> /
Enter value for v_accno: 100
old   2:  v_accno number(10):=&v_accno;
new   2:  v_accno number(10):=100;
PL/SQL procedure successfully completed.
SQL> select *from account;

ACCNO NAME                 BALANCE
--------- -------------------- ---------
     100 kavitha               26000
     101 raji                    63000
     102 kumar                 200
SQL> Enter value for v_accno: 300
old   2: v_accno number(10):=&v_accno;
new   2: v_accno number(10):=300;
there is no customer with this number
PL/SQL procedure successfully completed.
```

**Program 6.create a pl/sql block code that first insert a records in an employ table update the salaries of clerk and manager by 3000/- and 2000/ then check to see that the table salary does not exceed 20000/- if total salary is greater than 20000/- then undo the updates made to the salaries of clerk and manager**

```
SQL> create table employ(empno number(4) primary key,
2  name varchar2(20) not null,
3  sal number(10,2));
Table created.
SQL>desc employ
 Name                          Null?          Type
 ------------------------------ -------- ---------------------
EMPNO                          NOT NULL     NUMBER (4)
 NAME                          NOT NULL     VARCHAR2 (20)
 SAL                                        NUMBER(10,2)
SQL> insert into employ values(1,'blake',5000)
```

```
1 row created.
SQL> insert into employ values(2,'clark',6000)
1 row created.
SQL> commit;
Commit complete.
SQL> select *from employ;
EMPNO NAME                 SAL
--------- --------------------      ---------
      1 blake                  5000
      2 clark                  6000
SQL> declare
2  v_sal number(10,2);
3  begin
4  savepoint a;
5  update employ set sal=sal+2000 where name='blake';
6  update employ set sal=sal+1500 where name='clark';
7  select sum(sal) into v_sal from employ;
8  if v_sal>20000 then
9  rollback to savepoint a;
10  end if;
11  commit;
12  end;
13  /
PL/SQL procedure successfully completed.
SQL> SELECT *FROM EMPLOY;
EMPNO NAME            SAL
--------- -------------------- ---------
      1 blake                 7000
      2 clark                7500
```

**Program 7. Write a pl/sql code to evaluate the da,hra,pf,it,gross salary and net salary of the employ for the following conditions. If basic>8000 da is 56% else 44.4%, hra is 12.5% of basic, pf is 10% of basic.**

```
SQL> create table employ(empno number(4) primary key,
  name varchar2(20) not null,
  sal number(10,2));
Table created.
SQL> insert into employ values(1,'blake',5000)
1 row created.
SQL> insert into employ values(2,'clark',6000)
1 row created.
SQL> declare
  cursor emp_cur is select *from employ;
  v_empno number(5);
  v_name varchar2(30);
  v_sal number(10,2);
  da number(10,2);
  hra number(10,2);
  pf number(10,2);
  it number(10,2);
  gs number(10,2);
  ns number(10,2);
  begin
  open emp_cur;
```

```
 dbms_output.put_line('empno    name    grosssalarynetsalary');
 dbms_output.put_line('**********************************');
 loop
 fetch  emp_cur into v_empno,v_name,v_sal;
 exit when emp_cur%notfound;
 ifv_sal>5000 then
 da:=v_sal*56/100;
 else
 da:=v_sal*44/100;
 end if;
 hra:=v_sal*12.5/100;
 pf:=v_sal*10/100;
 it:=v_sal*5/100;
 gs:=v_sal+da+hra;
 ns:=gs-(pf+it);
dbms_output.put_line(v_empno||'    '||v_name||'   '||gs||'       '||ns);
end loop;
dbms_output.put_line('**********************************');
close emp_cur;
end;
SQL>/
empno name grosssalarynetsalary
**********************************
1       blake       11795           10745
2       clark       12637.5         11512.5
**********************************
PL/SQL procedure successfully completed.
```

**Program 8.Write a program read two numbers and display the value using elseif statement**

```
DECLARE
a number (2) := 21;
b number (2) := 10;
BEGIN
IF (a = b) then
dbms_output.put_line('Line 1 - a is equal to b');
ELSE
dbms_output.put_line('Line 1 - a is not equal to b');
END IF;
IF (a < b) then
dbms_output.put_line('Line 2 - a is less than b');
ELSE
dbms_output.put_line('Line 2 - a is not less than b');
END IF;
IF ( a> b ) THEN
dbms_output.put_line('Line 3 - a is greater than b');
DEV BHOOMI INSTITUTE OF TECHNOLOGY
LAB MANUAL
Course Name:DBMS Lab.
Experiment No. 6
Course Code : PCS-404
Faculty :Mr.Saurabh Singh
Branch: CSE Semester:IV
ELSE
dbms_output.put_line('Line 3 - a is not greater than b');
END IF;
END;
/
```

**Output:**
Line 1 - a is not equal to b
Line 2 - a is not less than b
Line 3 - a is greater than b

**Program 9. Display sal for given emp, if sal>3000 then raise exception and display message using exception.**

```
Create table emp(empno number(4), name varchar2(20), sal number(10,2), job varchar2(20));
Insert into emp values(1,'ramani',15000,'analyst');
Insert into emp values(2,'ramana',25000,'manager');
Insert into emp values(3,'kailash','2000,'clerk');
Insert into emp values(4,'lohitha',3100);

declare
v_empno emp.empno%type:=&v_empno;
v_ename emp.ename%type;
v_sal emp.sal%type;
emp_exec exception;
begin
select ename,sal into v_ename,v_sal from emp where empno=v_empno;
if v_sal>3000 then
raise emp_exec;
else
dbms_output.put_line(v_sal);
end if;
exception
when emp_exec then
dbms_output.put_line('salary greater than 3000');
end;
```

**Output:**

**Program 10.Accept employee number and salary from a user and update employee salary as per following rules if employee exist other wise display appropriate message using exception handling**
**a) If job is Manager then Salary not more then 7000**
**b) If job is Clerk then salary not more then 4000**
**c) If job is Analyst then salary not more then 24000**

```
Declare
v_sal emp.sal%type:=&v_sal;
v_eno emp.empno%type:=&v_eno;
v_job emp.job%type;
upd_error exception;
Begin
select job into v_job from remp where empno = v_eno;
if upper(v_job) = 'MANAGER' AND v_sal > 3000 then
raise upd_error;
elsif upper(v_job) = 'CLERK' AND v_sal > 500 then
raise upd_error;
elsif upper(v_job) = 'ANALYST' AND v_sal > 1000 then
raise upd_error;
else
update remp set sal = v_sal where empno=v_eno;
dbms_output.put_line('salary of employee '|| v_eno ||'is Updated to'|| v_sal);
end if;
Exception
when no_data_found then
dbms_output.put_line('Employee numbe not exist');
when upd_error then
dbms_output.put_line('Salary is not been changed as per the Rule');
end;
```

**Output:**

**Program 11.Accept employee number from a user and increase its salary depends on the current salary as follows. If sal>= 5000 12.5%; <5000 11%**

```
declare
 v_sal number(9,2);
 v_empno emp.empno%type:=&v_empno;
 Begin
 Select sal into v_sal from emp where empno=v_empno;
 If v_sal >= 5000 then
 update emp set sal = sal+(sal*0.125)where empno=v_empno;
 dbms_output.put_line('sal is:'||v_sal);
 else
 update emp set sal = sal + (sal*0.11) where empno=v_empno;
 dbms_output.put_line('sal is:'||v_sal);
 end if;
 end;
```

**Output:**

**Program 12. Write a PL/SQL Procedure to get the Employee Id from the table salary as input and Basic as IN OUT Parameter and calculate the bonus based on their Basic as per the following condition.**
**If the Basic below 10000 then increase the Basic to 8%**
**If the Basic between 10000 and 20000 then increase the Basic to 12%**
**If the Basic between 20000 and 30000 then increase the Basic to 15%**
**If the Basic above 30000 then increase the Basic to 20%**

**Query:**

```
SQL> CREATE OR REPLACE PROCEDURE emp_Bonus ( id IN salary.empid%type , Bas IN OUT
Salary.Basic%type)
IS
tmp_sal salary.Basic%type;
BEGIN
tmp_sal:=Bas;
IF tmp_sal < 10000 THEN
Bas := tmp_sal +(tmp_sal * .08);
ELSIF tmp_sal between 10000 and 20000 THEN
Bas := tmp_sal +(tmp_sal * .12);
ELSIF tmp_sal between 20000 and 30000 THEN
Bas := tmp_sal +(tmp_sal * .15);
ELSIF tmp_sal > 30000 THEN
Bas := tmp_sal +(tmp_sal * .20);
END IF;
END;
/
Procedure created.
```

**Output:**

```
SQL > DECLARE
CURSOR updated_sal is
SELECT empid, Basic FROM Salary;
pre_sal salary.Basic%type;
BEGIN
FOR emp_rec IN updated_sal
LOOP
pre_sal := emp_rec.Basic;
emp_Bonus(emp_rec.empID, emp_rec.Basic);
```

```
dbms_output.put_line('The Bonus of ' || emp_rec.empID || ' increased from '|| pre_sal || ' to
'||emp_rec.Basic);
END LOOP;
END;
/
```

**Output:**

```
The Bonus of 10 increased from 23000 to 26450
The Bonus of 11 increased from 33000 to 39600
The Bonus of 12 increased from 43000 to 51600
PL/SQL procedure successfully completed.
```

**Program 13. Create a pl/sql function to display the details of a person from phonebook**

```
SQL> create table phonebook (phone_no number (6) primary key,
Username varchar2(30),doorno varchar2(10),
street varchar2(30),place varchar2(30),pincode char(6));

SQL>Table created.

SQL> insert into phonebook values(20312,'vijay','120/5D', kota street ', Nandyal ','518501');
1 row created.
SQL> insert into phonebook values(29467,'vasanth','39D4', Gandhi road ', Kurnool ','518001');
1 row created.

SQL> select * from phonebook;
PHONE_NO USERNAME DOORNO STREET                PLACE   PINCODE
--------------------------------------------------------------------------------
20312  vijay            120/5D          kota street              Nandyal          518501
29467  vasanth          39D4    Gandhi road     Kurnool           518001

SQL> create or replace function findAddress(phone in number) return varchar2 as
address varchar2(100);

begin
select username||','||doorno ||','||street ||','||place||','||pincode into address from phonebook
where phone_no=phone;
return address;
exception
when no_data_found then return 'address not found';
end;
/
Function created.
```

**OUTPUT 1:**

```
SQL>declare
address varchar2(100);
begin
address:=findaddress(20312);
dbms_output.put_line(address);
6 end;
7 /

Vijay,120/5D, kota street , Nandyal,518501
PL/SQL procedure successfully completed.
```

**OUTPUT 2:**

```
SQL> declare
address varchar2(100);
begin
address:=findaddress(23556);
dbms_output.put_line(address);
end;
```

```
/

Address not found
PL/SQL procedure successfully completed.
```

**Program 14. Write a PL/SQL code to display employee number, name and basic of 5 highest paid employees.**

SOLUTION:

```
SQL> select * from employee;
EMP_NO EMPLOYEE_NAME          STREET              CITY
---------- -------------------- -------------------- ------------------
1          rajesh               10 ganpath          new delhi
2          paramesh             second cross        noida
3          pushpa               ghandhi road        banglore
4          vijaya               t nagar             chennai
5          keerthi              s r  nagar          hyderabad

SQL> select * from employee_salary;
EMP_NO     BASIC     HRA  DA   TOTAL_DEDUCTION NET_SALARY GROSS_SALARY
---------- ---------- -------- ----- --------------- ---------- ----------
2          15000   4000 1000     5000                15000      20000
1          31000   8000 1000     5000                35000      40000
3          14000   4000 1000     5000                15000      19000
4          14000   4000 1000     5000                15000      19000
5          13000   4000 1000     5000                15000      18000

SQL>
declare
 i number:=0;
cursor ec is select employee.emp_no,employee_name,basic from
employee, employee_salary where
employee.emp_no=employee_salary.emp_no order by gross_salary desc;
 r ec%rowtype;
 begin
 open ec;
 loop
 exit when i=5;
 fetch ec into r;
 dbms_output.put_line(r.emp_no||' '||r.employee_name||' '||r.basic);
 i:=i+1;
end loop;
 close ec;
end;
 /
1 rajesh 31000
2 paramesh 15000
3 pushpa 14000
4 vijaya 14000
5 keerthi 13000
PL/SQL procedure successfully completed.
```

**Program 15. Write a PL/SQL code to update the salary of employees who earn less than the average salary.**

**SOLUTION:**

```
SQL> select * from employee_salary;
EMP_NO BASIC             HRA  DA   TOTAL_DEDUCTION NET_SALARY GROSS_SALARY
---------- ---------- ------------ --------------- ---------- ----------
2          15000   4000 1000     5000                15000      20000
1          31000   8000 1000     5000                35000      40000
```

```
3               14000   4000 1000      5000                  15000          19000
4               14000   4000 1000      5000                  15000          19000
5               13000   4000 1000      5000                  15000          18000
SQL>
declare
average number;
bs number;
gs number;
diff number;
cursor ec is select * from employee_salary;
rw ec%rowtype;
begin
select avg(basic) into average from employee_salary;
dbms_output.put_line('the average salary is '||average);
open ec;
loop
fetch ec into rw;
exit when ec%notfound;
 if(rw.basic<=average) then
diff:=rw.basic-average;
update employee_salary set basic=average, gross_salary =
gross_salary + diff where emp_no=rw.emp_no;
select basic,gross_salary into bs,gs from employee_salary where
emp_no = rw.emp_no;
 dbms_output.put_line('the emploee number is '||rw.emp_no);
 dbms_output.put_line('old basic ='||rw.basic||'old gross_salary ='
|| rw.gross_salary);
dbms_output.put_line('updated new basic ='||bs||' new gross salary
is ='||gs);
end if;
end loop;
end;
 /
```

**OUTPUT:**

```
the average salary is 17400
the emploee number is 2
old basic =15000 old gross_salary=20000
updated new basic =17400 new gross salary is =17600
the emploee number is 3
old basic =14000 old gross_salary=19000
updated new basic =17400 new gross salary is =15600
the emploee number is 4
old basic =14000 old gross_salary=19000
updated new basic =17400 new gross salary is =15600
the emploee number is 5
old basic =13000 old gross_salary=18000
updated new basic =17400 new gross salary is =13600
PL/SQL procedure successfully completed.
```

**Program 16. Write a trigger on the employee table which shows the old values and new values of Ename after any updations on ename on Employee table.**

**SOLUTION:**

```
SQL> select * from employee;
EMP_NO EMPLOYEE_NAME               STREET                CITY
---------- -------------------- -------------------- ------------------
1               rajesh                    10 ganpath      new delhi
2               paramesh                        second cross    noida
3               pushpa                    ghandhi road    banglore
4               vijaya                    t nagar         chennai
5               keerthi                   s r  nagar              hyderabad
```

```
SQL>
create or replace trigger show
before update on employee
for each row
begin
dbms_output.put_line('the old name was :');
dbms_output.put_line(:old.employee_name);
dbms_output.put_line('the updated new name is :');
dbms_output.put_line(:new.employee_name);
end;
 /
Trigger created.
```

**OUTPUT:**

```
SQL> update employee set employee_name='kiran' where emp_no=1;
the old name was :
rajesh
the updated new name is :
kiran
1 row updated.

SQL> select * from employee;
EMP_NO EMPLOYEE_NAME               STREET                    CITY
---------- -------------------- -------------------- ------------------
1                kiran                          10 ganpath      new delhi
2                paramesh                       second cross    noida
3                pushpa                ghandhi road     banglore
4                vijaya                t nagar          chennai
5                keerthi               s r  nagar               hyderabad
```

**Program 17. Writ a PL/SQL procedure to find the number of students ranging from 100-70%, 69-60%, 59-50% & below 49% in each course from the student_course table given by the procedure as parameter.**

**SOLUTION:**

```
SQL> select * from student_enrollment;
ROLL_NO COURSE COURSE_COD SEM TOTAL_MARKS  PERCENTAGE
---------- ---------- ---------- ---------- ----------              ----------
111              BSc              1001             1        300                     50
112              BSc              1001             1        400                     66
113              BCom             1002             1        465                     77
114              BCom             1002             1        585                     97
SQL>
create or replace procedure rank(crc varchar) is
dis number:=0;
first number:=0;
sec number:=0;
pass number:=0;
cursor st is select * from student_enrollment;
r st%rowtype;
begin
open st;
loop
fetch st into r;
exit when st%notfound;
if(r.course=crc) then
if(r.percentage>=70 and r.percentage<=100)then
dis:=dis+1;
end if;
if(r.percentage>=60 and r.percentage<70) then
first:=first+1;
```

```
end if;
if(r.percentage>=50 and r.percentage<60) then
sec:=sec+1;
end if;
if(r.percentage>=35 and r.percentage<50)then
pass:=pass+1;
end if;
end if;
end loop;
close st;
dbms_output.put_line('distinction is '||dis);
dbms_output.put_line('first class is '||first);
dbms_output.put_line('second class is '||sec);
dbms_output.put_line('just pass is '||pass);
end;
/
Procedure created.
```

**OUTPUT:**

```
SQL> exec rank('cs');
distinction is 0
first class is 1
second class is 1
just pass is 0
PL/SQL procedure successfully completed.

SQL> exec rank('is');
distinction is 2
first class is 0
second class is 0
just pass is 0
PL/SQL procedure successfully completed.
```

**Program 18. Create a store function that accepts 2 numbers and returns the addition of passed values. Also write the code to call your function.**

**SOLUTION:**

```
SQL>
create or replace function addition(a number,b number)return number is
begin
dbms_output.put('the sum of '||a||' and '||b||' is :');
return (a+b);
end;
 /
Function created.
```

**OUTPUT:**

```
SQL> begin
dbms_output.put_line(addition(6,78));
end;
 /
the sum of 6 and 78 is: 84
PL/SQL procedure successfully completed.
```

**Program 19. Write a PL/SQL function that accepts department number and returns the total salary of the department. Also write a function to call the function.**

**SOLUTION:**

```
SQL> create table works
(emp_no number(3),
```

```
company_name varchar2(20),
joining_date date,
designation varchar2(20),
salary number(10,2),
deptno number(2));

SQL> table created.
insert into works
SQL> select * from works;
EMP_NO COMPANY_NAME JOINING_D  DESIGNATION  SALARY         DEPTNO
---------- -------------------- ---------   -------------------- ---------      ------
1               abc                      23-NOV-00 project lead        40000  1
2               abc                      25-DEC-10 software engg       20000  2
3               abc                      15-JAN-11 software engg       19000  1
4               abc                      19-JAN-11 software engg       19000  2
5               abc                      06-FEB-11 software engg       18000  1
SQL>
create or replace function tot_sal_of_dept(dno number)return number is
tot_sal number:=0;
begin
select sum(salary) into tot_sal from works where deptno=dno;
return tot_sal;
end;
/
Function created.
```

**OUTPUT:**

```
SQL> begin
dbms_output.put_line('Total salary of DeptNo 1 is :' ||tot_sal_of_dept(1));
end;
/

Total salary of DeptNo 1 is :77000
PL/SQL procedure successfully completed.

SQL> begin
dbms_output.put_line('total salary of dept 2 is :'||tot_sal_of_dept(2));
end;
 /
Total salary of DeptNo 2 is :39000
PL/SQL procedure successfully completed.
```

**Program 20. Write a PL/SQL code to create,**
**a) Package specification**
**b) Package body.**
**For the insert, retrieve, update and delete operations on a student table.**

**SOLUTION:**

```
SQL>
create or replace package alloperation is
procedure forinsert(rno number,sname varchar,crc varchar,gen varchar);
procedure forretrive(rno number);
procedure forupdate(rno number,sname varchar);
procedure fordelete(rno number);
end alloperation;
SQL> /
Package created.

SQL>
create or replace package body alloperation is
procedure forinsert(rno number,sname varchar,crc varchar,gen varchar) is
```

```
begin
insert into student values(rno,sname,crc,gen);
end forinsert;
procedure forretrive(rno number) is
sname student.student_name%type;
crc student.course%type;
gen student.gender%type;
begin
select student_name,course,gender into sname,crc,gen
from student where roll_no=rno;
dbms_output.put_line(sname||' '||crc||' '||gen);
end forretrive;
procedure forupdate(rno number,sname varchar) is
begin
update student set student_name=sname where roll_no=rno;
end forupdate;
procedure fordelete(rno number) is
begin
delete student where roll_no=rno;
end fordelete;
end alloperation;
/
Package body created.

SQL> select * from student;
ROLL_NO STUDENT_NAME COURS GENDER
---------- -------------------- ----- ----------
111             ravi                 cs    male
112             praveen              cs    male
113             bhuvana              is    female
114             apparna              is    female
```

**OUTPUT:**

```
SQL> begin
alloperation.forinsert(444,'vivekananda','ec','male');
alloperation.forretrive(444);
alloperation.forupdate(111,'swamy');
end;
 /

vivekananda ec male
PL/SQL procedure successfully completed.

SQL> select * from student;
ROLL_NO STUDENT_NAME COURS GENDER
---------- -------------------- ----- ----------
111 swamy cs male
112 praveen cs male
113 bhuvana is female
114 apparna is female
444 vivekananda ec male

SQL> begin
alloperation.fordelete(444);
end;
 /
PL/SQL procedure successfully completed.

SQL> select * from student;
ROLL_NO STUDENT_NAME COURS GENDER
---------- -------------------- ----- ----------
111 swamy cs male
```

112 praveen cs male
113 bhuvana is female
114 apparna is female