St. Paul's College of Management & IT
(Affiliated to Osmania University & Approved by AICTE)
SyNo: 603, 604 & 605. Turkayamjal (vi.), Hayathnagar (M), R.R.Dist.

CERTIFICATE

This is to be certify that Mr. /Ms……..
Of M.B.A III Semester with H.NO……
Has Submitted RDBMS-LAB record during the Academic year
20 -20 .

Internal Examiner: External Examiner:

Head of the Department

Date:

INDEX

1. CREATE A TABLE WITH VALIDATIONS:

Parent Table:

```
SQL>  create table dept_tab(
      deptno number(3) primary key,
      dname varchar2(15),
      loc varchar2(15),
      constraint dname_ukey unique(dname,loc),
      constraint loc_check1
      check(loc in('newyork','boston','chicago')));
```

Table created.

Child table :

.

```
SQL>  create Table emp_tab(
      empno number(5) primary key,
      ename varchar2(15) not null,
      job varchar2(10),
     mgr number(5)constraint mgr_fkey references emp_tab,
     hiredate date,
     sal number(7,2),
    comm number(5,2),
    deptno number(3) not null constraint dept_fkey
    references dept_tab on delete cascade);
```

Table created.

OUTPUT:

Parent table:

SQL> desc dept_tab;

Name	*Null?*	*Type*
DEPTNO	*NOT NULL*	*NUMBER(3)*
DNAME		*VARCHAR2(15)*
LOC		*VARCHAR2(15)*

Child table:

SQL> desc emp_tab;

Name	*Null?*	*Type*
EMPNO	*NOT NULL*	*NUMBER(5)*
ENAME	*NOT NULL*	*VARCHAR2(15)*
JOB		*VARCHAR2(10)*
MGR		*NUMBER(5)*
HIREDATE		*DATE*
SAL		*NUMBER(7,2)*
COMM		*NUMBER(5,2)*
DEPTNO	*NOT NULL*	*NUMBER(3)*

2. LIST THE VARIOUS CONSTRAINT MODIFICATIONS:

Enabling Constraints:

SQL>create table emp_tab(empno number(5) primary key);

SQL>alter table emp_tab add primary key(empno);

Disabling constraints:

SQL>create table emp_tab(empno number(5) primary key disable);

SQL>alter table emp_tab add primary key (empno) disable;

***Enabling Disabling Constriants**:*

SQL>alter table dept_tab enable primary key enable unique(dname) enable unique(loc);

***Disabling Enabling Constraints**:*

SQL>alter table dept_tab
disable constraint dname_ukey;

SQL>alter table dept_tab
disable primary key
disable unique(dname)
disable unique(loc);

Dropping Integrity Constraints:

SQL>alter table dept_tab
drop unique(dname);

SQL>alter table dept_tab
drop unique(loc);

OUTPUT:

In the above queries we can ADD,ENABLE,DISABLE and DROP a constraint
with the ALTER command as shown above.

We get confirmation of the modified table with the message as
TABLE ALTERED

SQL>alter table empoyee enable primary key;
Table altered.

desc employee;

Name	*Null?*	*Type*
ENMAE		*VARCHAR2(20)*
EMPNO	*NOT NULL*	*NUMBER(5)*

SQL>alter table employee drop primary key;
table altered.

desc employee;

Name	*Null?*	*Type*
ENAME		*VARCHAR2(20)*
EMPNO		*NUMBER(5)*

3. *LIST THE VARIOUS TABLE MANIPULATIONS:*

Inserting values in a table:

SQL>insert into dept values(deptno,'&dname,'&loc');

SQL>insert into dept values(10,'sales','hyderbad');

Altering a table:

SQL>alter table dept modify(dname varchar2(15)not null);

SQL>alter table dept add(dot date);

SQL>alter table products add(constraints fk_products-02
foreign key (color) references(avail_colors.colors));

SQL>alter table emp drop constriant emp_pri;

Deleting a row:

SQL>delete from emp where empno=120;

Droping tables:

SQL>drop table products;

SQL>drop table products cascade constraints;

OUTPUT:

SQL>insert into dept values(10,'salesman','Hyderabad');

1 row created.

*SQL>select * from dept;*

DEPTNO DNAME LOC
-------- ---------- ---------
10 salesman Hyderabad

4. **PL/SQL Programs :**

1) SQL> begin
dbms_output.put_line('hello world');
end;

PL/SQL procedure successfully completed.

OUTPUT:

SQL> set serveroutput on;
SQL> /
hello world

2) SQL> declare
order_no number(4);
begin
Order no:=&order no;
dbms_output.put_line('order number is '||order no);
End;

OUTPUT :

Enter value for order_no: 12
old 4: order_no:=&order_no;
new 4: order_no:=12;
order_number is 12

PL/SQL procedure successfully completed.

EXAMPLE FOR STRUCTURE DECLARATION :

.

```
SQL> declare
        type student is record(sno number,sname varchar2(10));
        a student;
        begin
       a.sno:=20;a.sname:='raju';
       dbms_output.put_line(a.sno||' '||a.sname);
      end;
```

PL/SQL procedure successfully completed.

OUTPUT:

```
SQL> set serveroutput on;
SQL> /
20 raju

PL/SQL procedure successfully completed.
```

5. ***DECLARATION USING ATTRIBUTES:***

DECLARATION USING % TYPE :

```
SQL> declare
     dno department.deptno%type;
     dname department.dname%type;
     begin
     select deptno,dname into dno,dnam from department
    deptno=&deptno;
    dbms_output.put_line('deptno='||dno);
    dbms_output.put_line('dname='||dnam);
    end;
```

PL/SQL procedure successfully completed.

OUTPUT :

```
Enter value for deptno: 1
old   5:   select deptno,dname into dno,dnam from department where
deptno=&deptno;
new   5:   select deptno,dname into dno,dnam from department where
deptno=1;
deptno=1
dname=suman

PL/SQL procedure successfully completed.
```

DECLARATION USING % ROW TYPE ATTRIBUTE :

```
SQL>  declare
        d department%rowtype;
        begin
        select deptno,dname into d.deptno,d.dname from department
where
       deptno=&deptno;
       dbms_output.put_line('deptno='||d.deptno);
       dbms_output.put_line('dname='||d.dname);
       end;


PL/SQL procedure successfully completed.
```

OUTPUT:

```
Enter value for deptno: 1
old   4:  select deptno,dname into d.deptno,d.dname from department
where deptno=&deptno;
new   4:  select deptno,dname into d.deptno,d.dname from department
where deptno=1;
deptno=1
dname=suman

PL/SQL procedure successfully completed.
```

6. ***FACTORIAL OF A GIVEN NUMBER***

```
SQL>  declare
        a number:=&a;
        fact number:=1;
        begin
        for i in 2..a loop
        fact:=fact*i;
       end loop;
       dbms_output.put_line('Factorial of a = '||fact);
       end;
```

PL/SQL procedure successfully completed.

OUTPUT :

```
Enter value for a: 3
old   2: a number:=&a;
new   2: a number:=3;
Factorial of a = 6

PL/SQL procedure successfully completed.
```

7. CHECKING PRIME NUMBER OR NOT :

```
SQL> declare
       n number:=&n;
       f number:=0;
       begin
       for i in 2..n-1 loop
      if mod(n,i)=0 then
      f:=f+1;
      end if;
      end loop;
      if f=0 then
     dbms_output.put_line('prime');
     else
    dbms_output.put_line('not prime');
    end if;
   end;
```

PL/SQL procedure successfully completed.

OUTPUT :

```
Enter value for n: 3
old   2: n number:=&n;
new   2: n number:=3;
prime

PL/SQL procedure successfully completed.
```

8. ***PROGRAM USING CONDITIONAL CONTROL STRUCTURES:***

PROGRAM USING IF...ELSE...END IF :

```
SQL>  declare
       x number:=&x;
       begin
       if mod(x,2)=0 then
      dbms_output.put_line('It Is Even number');
      else
      dbms_output.put_line('It Is Odd number');
      end if;
      end;
```

PL/SQL procedure successfully completed.

OUTPUT :

```
Enter value for x: 3
old   2: x number:=&x;
new   2: x number:=3;
It Is Odd number

PL/SQL procedure successfully completed.
```

Program using if...elsif...Else...Endif

```
SQL>  declare
        x integer:=&number;
        y integer:=&number;
        z integer:=&number;
        begin
        if x>y and x>z then
       dbms_output.put_line(x||'is big');
       elsif y>z then
       dbms_output.put_line(y||'is big');
       else
       dbms_output.put_line(z||'is big');
       end if;
       end;
```

PL/SQL procedure successfully completed.

OUTPUT :

```
Enter value for number: 12
old   2: x integer:=&number;
new   2: x integer:=12;
Enter value for number: 14
old   3: y integer:=&number;
new   3: y integer:=14;
Enter value for number: 16
old   4: z integer:=&number;
new   4: z integer:=16;
16is big
```

PL/SQL procedure successfully completed.

9. ***PROGRAM USING ITERATIVE CONTROL STRUCTURES:***

PROGRAM USING LOOP...END LOOP:

```
SQL> declare
     i number;
     begin
     i:=0;
     loop
     i:=i+1;
     dbms_output.put_line('Value of is '||i);
     exit when i=10;
     end loop;
     end;
```

PL/SQL procedure successfully completed.

OUTPUT :

```
Value of is 1
Value of is 2
Value of is 3
Value of is 4
Value of is 5
Value of is 6
Value of is 7
Value of is 8
Value of is 9
Value of is 10
```

PL/SQL procedure successfully completed.

PROGRAM USING FOR J IN...LOOP...END LOOP :

```
SQL>  declare
        j number;
        begin
       for j in 1..10 loop
       dbms_output.put_line('Value of j is '||j);
       end loop;
       end;
```

PL/SQL procedure successfully completed.

OUTPUT :

Value of j is 1
Value of j is 2
Value of j is 3
Value of j is 4
Value of j is 5
Value of j is 6
Value of j is 7
Value of j is 8
Value of j is 9
Value of j is 10

PL/SQL procedure successfully completed.

PROGRAM WITH WHILE...END LOOP :

```
SQL>  declare
        k number;
        begin
        k:=10;
       while k>0 loop
       dbms_output.put_line('Value of k is '||k);
       k:=k-1;
       end loop;
       end;
```

PL/SQL procedure successfully completed.

OUTPUT :

Value of k is 10
Value of k is 9
Value of k is 8
Value of k is 7
Value of k is 6
Value of k is 5
Value of k is 4
Value of k is 3
Value of k is 2
Value of k is 1

PL/SQL procedure successfully completed.

10. USING CURSORS TO EXTRACT ROWS FROM TABLE:

```
SQL>  declare
        cursor c1 is select ename from emp;
        name emp.ename%type;
        begin
        open c1;
        dbms_output.put_line('Employee Nmaes');
        dbms_output.put_line('***************');
        loop
       fetch c1 into name;
       exit when c1%notfound;
       dbms_output.put_line(name);
      end loop;
      close c1;
      end;
```

PL/SQL procedure successfully completed.

OUTPUT :

```
SQL> set serveroutput on;
SQL> /
Employee Nmaes
***************
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
```

PL/SQL procedure successfully completed.

11. RETRIEVING THE DATA THROUGH CURSORS:

```
SQL> declare
     cursor my_cur is select *from studs;
     msno number;
     msname varchar2(10);
     mm1 number;
     mm2 number;
     mm3 number;
     mtot number;
     mavg number;
     begin
     open my_cur;
     dbms_output.put_line('sno sname   m1  m2  m3  tot   avg');
     loop
     fetch my_cur into msno,msname,mm1,mm2,mm3;
     exit when my_cur%notfound;
     mtot:=mm1+mm2+mm3;
     mavg:=mtot/3;
     dbms_output.put_line(msno||' '||msname||' '||mm1||'
'||mm2||''||mm3||'
    '||mtot||' '||mavg);
     end loop;
     close my_cur;
     end;
```

OUTPUT:

```
sno sname   m1  m2  m3  tot   avg

1   ganesh  45  65  76  186   62

PL/SQL procedure successfully completed.
```

12.PL/SQL WITH EXCEPTION :

1).SQL> declare
my_empid emp.empno%type;
begin
my_empid:=5969;
select empno into my_empid from emp where empno=my_empid;
exception
when no_data_found then
dbms_output.put_line('No data found');
end;

2).SQL> begin
insert into dept(deptno,dname)values(&dno,'&dname');
commit;
exception
when dup_val_on_index then
dbms_output.put_line('Violates unique constraint');
end;

OUTPUT :

1. No data found

PL/SQL procedure successfully completed.

2. Enter value for dno: 10
Enter value for dname: games
old 2: insert into dept(deptno,dname)values(&dno,'&dname');
new 2: insert into dept(deptno,dname)values(10,'games');
Violates unique constraint

PL/SQL procedure successfully completed.

13. HANDLING MULTIPLE EXCEPTIONS :

```
SQL> declare
     v_ename emp.ename%type;
     v_job emp.job%type;
     begin
     select ename,job into v_ename,v_job from emp where sal between
     10000 and 20000;
     exception
     when no_data_found then
     dbms_output.put_line('Data not from table');
     when too_many_rows then
    dbms_output.put_line('More than one row is available');
    end;
```

PL/SQL procedure successfully completed.

OUTPUT :

```
SQL> set serveroutput on;
     Data not from table

PL/SQL procedure successfully completed.

SQL> select sal from emp;

     SAL
---------
     800              1100
    1600              950
    1250              3000
    2975              1300
    1250
    2850
    2450          14 rows selected.
    3000
    5000
    1500
```

14. EXAMPLE FOR ZERO DIVIDE EXCEPTION:

```
SQL>  declare
        a number;
        x number:=&x;
       begin
       a:=100/x;
      dbms_output.put_line('a value is '||FLOOR(a));
      exception
      when zero_divide then
      dbms_output.put_line('You should give x value other than
zero');
     end;
```

OUTPUT:

```
Enter value for x: 5
old   3: x number:=&x;
new   3: x number:=5;
a value is 20


PL/SQL procedure successfully completed.


SQL> /
Enter value for x: 0
old   3: x number:=&x;
new   3: x number:=0;
You should give x value other than zero


PL/SQL procedure successfully completed.
```

15. EXAMPLE FOR USER DEFINE EXCEPTION:

```
SQL>  declare
        x number:=&x;
       Zing_zang exception;
        begin
        if x=0 then
            raise zing_zang;
       else
            dbms_output.put_line('Given Number is '||x);
     end if;
     exception
    when zing_zang then
  dbms_output.put_line('zing_zang exception is raised');
  end;
```

OUTPUT:

```
Enter value for x: 56
old   2: x number:=&x;
new   2: x number:=56;
Given Number is 56

PL/SQL procedure successfully completed.

SQL> /
Enter value for x: 0
old   2: x number:=&x;
new   2: x number:=0;
zing_zang exception is raised

PL/SQL procedure successfully completed.
```

16. IMPLEMENTATION OF TRIGGERS:

```
SQL>  create or replace trigger print_salary_changes
        before insert or update on emp
        for each row
       when(new.empno>0)
       declare
      sal_diff number;
      begin
     sal_diff := :new.sal-:old.sal;
     dbms_output.put('Name :'||:new.ename);
     dbms_output.put(' Old salary: '||:old.sal);
    dbms_output.put(' New salary:'||:new.sal);
   dbms_output.put_line(' The difference '|| sal_diff);
   end;
```

OUTPUT:

Trigger created.

```
SQL> update emp set sal=sal+500 where deptno=10;
Name :CLARK Old salary: 2450 New salary:2950 The difference
500
Name :KING Old salary: 5000 New salary:5500 The difference 500
Name :MILLER Old salary: 1300 New salary:1800 The difference
500
```

3 rows updated.

17. EXAMPLE FOR ARRAY DECLARATION:

```
SQL>  declare
        type my_array is table of number index by binary_integer;
       a my_array;
       begin
       for i in 1..4 loop
      a(i):=i*2;
     end loop;
     for i in 1..4 loop
    dbms_output.put_line(a(i));
   end loop;
   end;
```

OUTPUT:

```
2
4
6
8

PL/SQL procedure successfully completed.
```

18. VARIOUS TYPES OF INDEXES:

Unique indexes:
SQL>Create unique index employee_lastname_indx_01
on employee(lastname);

Nonunique indexex:
SQL>Create index employee_last_indx_01
on employee(lastname);

Composite indexes:
SQL>Create unique index employee_last_first_indx_01
on employee(lastname,firstname);

Bitmap indexes:
SQL>Create bitmap index employee_status_indx_01
on employee(empl_status);

OUTPUT:

A Unoique index is created on the table 'employee' with respect to the column 'lastname'.

Index created.

ANon-unique index is created on the table 'employee' with respect to the column 'lastname'.

Index created.

A Composite unique index is created on the table 'employee' with respect to the column
'lastname and firstname'.

Index created

19. QUERIES:

SQL> select empno,ename from emp where deptno not in(10,20);

```
   EMPNO ENAME
--------- ----------
    7499 ALLEN
    7521 WARD
    7654 MARTIN
    7698 BLAKE
    7844 TURNER
    7900 JAMES
```

SQL> select sum(sal),deptno from emp group by deptno having sum(sal)>1000;

```
 SUM(SAL)    DEPTNO
--------- ---------
    8750         10
   10875          20
    9400         30
```

SQL> select sum(sal),min(sal),max(sal),deptno from emp group by deptno having count(*)>3;

```
 SUM(SAL)  MIN(SAL)  MAX(SAL)    DEPTNO
--------- --------- --------- ---------
   10875       800      3000         20
    9400       950      2850         30
```

SQL> select ename from emp where sal=(select min(sal) from emp);

```
ENAME
----------
SMITH
```

SQL> select ename from emp where (sal,deptno) in (select max(sal),deptno from emp group by deptno);

ENAME

BLAKE
SCOTT
FORD
KING

SQL> select ename from emp e where sal>(select avg(sal) from emp where deptno=e.deptno);

ENAME

ALLEN
JONES
BLAKE
SCOTT
KING
FORD

6 rows selected.

*SQL> select * from dept d where not exists(select * from emp where deptno=d.deptno);*

DEPTNO DNAME LOC
--------- ------------- -------------
40 OPERATIONS BOSTON

SQL> select ename ,job,sal,grade from emp,salgrade where sal between losal and hisal and grade=1;

ENAME JOB SAL GRADE
---------- --------- --------- ---------
SMITH CLERK 800 1
ADAMS CLERK 1100 1
JAMES CLERK 950 1

SQL> select e.ename,e.sal,m.ename,m.sal from emp e,emp m where e.mgr=m.empno and e.sal>m.sal;

```
ENAME          SAL ENAME          SAL
---------- --------- ---------- ---------
SCOTT         3000 JONES         2975
FORD          3000 JONES         2975
```

SQL> select e.ename,e.hiredate,m.ename,m.hiredate from emp e,emp m where e.mgr=m.empno and e.hiredat e>m.hiredate;

```
ENAME      HIREDATE  ENAME      HIREDATE
---------- --------- ---------- ---------
MARTIN     28-SEP-81 BLAKE      01-MAY-81
SCOTT      19-APR-87 JONES      02-APR-81
TURNER     08-SEP-81 BLAKE      01-MAY-81
ADAMS      23-MAY-87 SCOTT      19-APR-87
JAMES      03-DEC-81 BLAKE      01-MAY-81
FORD       03-DEC-81 JONES      02-APR-81
MILLER     23-JAN-82 CLARK      09-JUN-81
```

SQL> select ename,sal from emp e where sal in(select sal from emp where sal=e.sal and rowid<>e.rowid);

```
ENAME          SAL
---------- ---------
WARD          1250
MARTIN         1250
SCOTT         3000
FORD          3000
```

SQL> select ename,sal from emp e where 2>(select count(*) from emp where e.sal<sal)order by sal desc;

```
ENAME          SAL
---------- ---------
KING          5000
SCOTT         3000
FORD          3000
```

GRANT COMMAND:

GRANT privileges ON object TO user[WITH GRANT OPTION];

GRANT select ,update,insert ON employee TO Howlett WITH GARNT OPTION;(Executed By MARY)

GRANT select ON employee TO Athena WITH GRANT option;
(Executed By HOWLETT)
GRANT select ON employee.empid TO Susan; (Executed by ATHENA)

REVOKE COMMAND:

REVOKE[GRANT OPTION FOR] privileges ON object FROM users {RESTRICT/CASCADE};

REVOKE select ,update,insert ON employee FROM Howlett CASCADE;
(Executed By MARY)
(OR)
REVOKE GRANT OPTION FOR select ON employee FROM Athena CASCADE;(Executed By HOWLETT)

SYNONYM

CREATE SYNONYM e FOR emp;
CREATE SYSNONYM objects FOR products;
CREATE PUBLIC SYNONYM objects FOR products;

SEQUENCE:

CREATE SEQUENCE DEPT_SEQ START WITH 10
INCREMENT BY 10
MAXVALUE 100
Minvalue 10;