Department of Computer Applications

CA702

DBMS LAB

205119037

Harshita Jha

1. Data Definition Language (DDL) Commands

1.1 Create table EMP(EMPNO number(4), ENAME varchar2(20),
JOB varchar2(10), MGR number(4), DEPTNO number(3),
SAL number(7,2));

1.2 Alter table EMP add COMM number(5);

1.3 Alter table EMP modify JOB varchar2(20);

1.4 Create table DEPT(DEPTNO number(2), DNAME varchar2(10),
DLOC varchar2(20));

1.5 Alter table EMP modify EMPNO NOT NULL PRIMARY KEY;
Alter table DEPT add FOREIGN KEY(DEPTNO) REFERENCES
DEPT(DEPTNO);

1.6 Alter table EMP add CHECK(EMPNO>100);

1.7 Alter table EMP modify SAL DEFAULT 5000 NOT NULL;

1.8 Alter table EMP add DOB date;

2. Data Manipulation Language (DDL) Commands

2.1 Insert all

```
    into DEPT (DNAME, DEPTNO, DLOC) values
        (10, ’MANAGEMENT’, ’MAIN BLOCK’)
    into DEPT (DNAME, DEPTNO, DLOC) values
        (20, ’DEVELOPMENT’,’MANUFACTURING UNIT’)
    into DEPT (DNAME, DEPTNO, DLOC) values
        (30, ’MAINTAINANCE’, ’MAIN BLOCK’)
    into DEPT (DNAME, DEPTNO, DLOC) values
        (40, ’TRANSPORT’, ’ADMIN BLOCK’)
    into DEPT (DNAME, DEPTNO, DLOC) values
        (50, ’SALES’, ’HEAD OFFICE’)
   Select * from DUAL;
```

2.2 insert into EMP values(7369, 'smith', 'clerk', 7566, 20, 800, 0, '17-dec-1980');

```
   insert into EMP values(7399, 'asant', 'salesman', 7566, 20, 1600, 300, '20-feb-1981');
  insert into EMP values(7499, 'allen', 'salesman', 7698, 30, 1600, 300, '20-feb-1981');
  insert into EMP values(7521, 'ward', 'salesman', 7698, 30, 1250, 500, '22-feb-1982');
  insert into EMP values(7566, 'jones', 'manager', 7839, 20, 5975, 500, '02-apr-1981');
  insert into EMP values(7698, 'blake', 'manager', 7839, 30, 9850, 1400,'01-may-1979');
  insert into EMP values(7611, 'scott', 'hod', 7839, 10, 3000, NULL,'12-jun-1976');
  insert into EMP values(7839, 'clark', 'ceo', NULL, 10, 9900, NULL, '16-mar-1972');
```

insert into EMP values(7368, 'ford', 'supervisor', 7366, 20, 800, 0, '17-dec-1980');

insert into EMP values(7599, 'alley', 'salesman', 7698, 30, 1600, 300, '20-feb-1981');

insert into EMP values(7421, 'drank', 'clerck', 7698, 30, 1250, 500, '22-jan-1982');

2.3 Update EMP set COMM=1000 where JOB='manager';

2.4 Create table EMPLOYEE(EMPNO number(6), ENAME varchar2(20),

JOB varchar2(10), MGR number(4), DEPTNO number(3),

SAL number(7,2),COMM number(5), DOB date);

INSERT INTO EMPLOYEE SELECT * FROM EMP;

2.5 Delete from EMPLOYEE where JOB='supervisor';

2.6 Delete from EMPLOYEE where EMPNO=7599;

2.7 select * from EMP ORDER BY SAL;

2.8 select * from EMP ORDER BY SAL DESC;

2.9 select * from EMP where DEPTNO=30;

2.10 select DISTINCT DEPTNO from EMP;

2.11 select * from EMP ORDER BY ENAME;

2.12 create table MANAGER as select * from EMP where JOB='manager';

2.13 select * from EMP where COMM=NULL;

2.14 select ENAME, DNAME from EMP, DEPT where EMP.DEPTNO=DEPT.DEPTNO;

3. In-Built Functions

3.1 select * from EMP where DEPTNO IN (20,10);

3.2 select * from EMP where ENAME LIKE 'S%';

3.3 select * from EMP where ENAME NOT LIKE ‘S%’;

3.4 select * from EMP where EMPNO BETWEEN 7500 AND 7600;

3.5 select * from EMP where EMPNO NOT BETWEEN 7500 AND 7600;

3.6 select SQRT(SAL) from EMP;

3.7 select COUNT(*) from EMP;

3.8 select SUM(SAL), AVG(SAL) from EMP;

3.9 select MIN(SAL) as MIN_SAL, MAX(SAL) as MAX_SAL from EMP;

3.10 select SUM(SAL) from EMP;

3.11 select JOB, SUM(SAL) from EMP GROUP BY JOB;

3.12 select TO_CHAR(TO_DATE(’14-jul-09’),’month’) from DUAL;

3.13 select TO_DATE(DOB,’DD-MM-YY’) from EMP;

3.14 select ADD_MONTHS(DOB,2) from EMP;

3.15 select LAST_DAY(’05-oct-09’) from DUAL

3.16 select ROUND(TO_DATE(DOB),'month') from EMP;

select ROUND(TO_DATE(DOB),'year') from EMP;

select ROUND(TO_DATE(DOB),'day') from EMP;

3.17 select (SYSDATE-60) from DUAL;

3.18 select ENAME , SAL, SAL+0.15*SAL from EMP;

3.19 select ENAME from EMP where ENAME LIKE ‘B%’ or ENAME LIKE ‘C%’;

3.20 select ENAME, SAL, MGR, from EMP where SAL in (select MIN(SAL) from EMP GROUP BY MGR);

3.21 select DNAME, COUNT(ENAME) from EMP, DEPT where EMP.DEPTNO=DEPT.DEPTNO GROUP BY DNAME;

3.22 select ENAME from EMP where length(ENAME)<=5;

3.23 select ENAME from EMP where MGR IN(7399,7698,7566);

3.24 select COUNT(DISTINCT JOB) from EMP;

3.25 select MAX(SAL) - MIN(SAL) from EMP;

3.26 select COUNT(DISTINCT DEPTNO) from EMP;

3.27 select ENAME, DOB from EMP where TO_CHAR(DOB,'MON')='feb';

3.28 select ENAME from EMP where TO_CHAR(DOB,'MON') LIKE TO_CHAR(SYSDATE,'MON');

3.29 select ENAME from EMP where ENAME LIKE 'S%H';

3.30 select ENAME from EMP where SAL>5000;

4. Nested Queries and Joins

4.1 select * from EMP, DEPT where EMP.DEPTNO=DEPT.DEPTNO and (DEPT.DNAME=’MAINTAINANCE’ or DEPT.DNAME=’DEVELOPMENT’)

4.2 select ENAME,SAL from EMP where SAL>(select MIN(SAL) from EMP) and JOB LIKE ‘M%’;

4.3 select * from EMP where JOB=(select JOB from EMP where ENAME=’jones’);

4.4 select * from EMP where SAL>(select MAX(SAL) from EMP where DEPTNO=30);

4.5 select * from EMP where JOB=(select JOB from EMP where ENAME=’jones’) and SAL>=(select SAL from EMP where ENAME=’ford’);

4.6 select ENAME, JOB from EMP where DEPTNO=20 and JOB=ANY(select JOB from EMP E, DEPT D where E.DEPTNO=D.DEPTNO and D.DNAME=’MANAGEMENT’);

4.7 select * from EMP OUT where SAL>(select AVG(SAL) from EMP where DEPTNO=OUT.DEPTNO);

4.8 select ENAME, JOB, DNAME from EMP E, DEPT D where E.DEPTNO=D.DEPTNO;

4.9 select * from EMP where JOB=ANY(select E.JOB from DEPT D,EMP E where D.DEPTNO=E.DEPTNO and DLOC='MAIN BLOCK') and DEPTNO!=(select DEPTNO from DEPT where DLOC='MAIN BLOCK');

4.10 select * from EMP where DEPTNO=10 and JOB=ANY(select JOB from EMP, DEPT where DEPT.DEPTNO=EMP.DEPTNO and DEPT.DNAME=’DEVELOPMENT’);

4.11 select * from EMP where JOB=(select JOB from MEP where ENAME=’ford’) and SAL=(select SAL from EMP where ENAME=’ford’);

4.12 select DNAME from DEPT where DEPTNO= ANY(select DEPTNO from (select COUNT(JOB) as NO, DEPTNO from EMP where JOB=’salesman’ GROUP BY DEPTNO) where NO>=2);

4.13 select * from EMP where DEPTNO=20 and JOB=ANY(select JOB from EMP where DEPTNO=30);

4.14 select * from EMP where SAL> ANY(select MAX(SAL) from EMP where DEPTNO=20 or DEPTNO=30 GROUP BY DEPTNO);

4.15 select MAX(SAL) from EMP GROUP BY DEPTNO HAVING MAX(SAL) >9000;

4.16 select MAX(SAL) from EMP GROUP BY ENAME HAVING MIN(SAL)>1000 and MIN(SAL)<5000;

JOINS

✓ Table Creation

Create table AccDept(DEPTNO number(2) PRIMARY KEY , DNAME varchar2(20), DLOC varchar2(20));

✓ Inserting values in AccDept Table

insert into AccDept values(10, 'MANAGEMENT', 'MAIN BLOCK');

insert into AccDept values(20,'DEVELOPMENT','MANUFACTURINGUNIT');

insert into AccDept values(30, 'MAINTAINANCE', 'MAIN BLOCK');

4.17 select A.DNAME from DEPT D, ACCDEPT A where D.DEPTNO=A.DEPTNO;

4.18 select ENAME from EMP where DEPTNO!=ANY (select DEPTNO from AccDept);

4.19 select * from EMP LEFT JOIN DEPT on DEPT.DEPTNO=EMP.DEPTNO;

4.20 select * from EMP RIGHT JOIN DEPT on DEPT.DEPTNO=EMP.DEPTNO;

4.21 select * from EMP FULL JOIN DEPT on DEPT.DEPTNO=EMP.DEPTNO;

4.22 select E.ENAME, E!.ENAME from EMP E, EMP E1 where E.MGR=E1.EMPNO;

4.23 select E.ENAME, E1.SAL from EMP E, EMP E1 where E.MGR=E1.EMPNO;

4.24 select E.ENAME , E.JOB, E.EMPNO, D.DNAME, D.DLOC from EMP E, DEPT D where E.DEPTNO=E.DEPTNO and D.DEPTNO=E.DEPTNO;

4.25 select E.EMPNO, E.ENAME, E.JOB, E1.ENAME from EMP E, EMP E1 where E.MGR=E1.EMPNO;

4.26 select E.ENAME , E1.ENAME from EMP E, EMP E1 where E.SAL=E1.SAL and E.ENAME!=E1.ENAME;

5. SET Operators and Views

5.1 select DEPTNO from DEPT UNION select DEPTNO from AccDept;

5.2 select DEPTNO from DEPT UNION all select DEPTNO from AccDept;

5.3 select DEPTNO from DEPT INTERSECT select DEPTNO from AccDept;

5.4 select DEPTNO from DEPT MINUS select DEPTNO from AccDept;

5.5 create view manager1 as select * from EMP where JOB='manager';

5.6 create view general as select EMPNO,ENAME, EMP. DEPTNO from EMP, DEPT where EMP. DEPTNO = DEPT. DEPTNO;

5.7 create view all1 as select EMPNO,ENAME, EMP. DEPTNO, DNAME from EMP, DEPT where EMP. DEPTNO = DEPT. DEPTNO;

5.8 select view_name from user_views;

5.9 insert into manager values(7201,'CLAVE',20,'Computer');

5.10 drop VIEW all1;

6. Control Structures

6.1 Write a PL/SQL program to swap two numbers without taking third variable

```
 declare
  a number(10);
  b number(10);
  begin
      a:=&a;
      b:=&b;
     dbms_output.put_line('THE PREV VALUES OF A AND B WERE');
     dbms_output.put_line(a);
     dbms_output.put_line(b);
      a:=a+b;
      b:=a-b;
      a:=a-b;
      dbms_output.put_line('THE VALUES OF A AND B ARE');
      dbms_output.put_line(a);
      dbms_output.put_line(b);
  end;
```

<u>OUTPUT</u>

```
Enter value for a: 5
Enter value for b: 3
THE PREV VALUES OF A AND B WERE
 5
 3
THE VALUES OF A AND B ARE
 3
 5
```

6.2 Write a PL/SQL program to swap two numbers by taking third variable

```
declare
a number(10);
b number(10);
c number(10);
begin
    dbms_output.put_line('THE PREV VALUES OF A AND B WERE');
    dbms_output.put_line(a);
    dbms_output.put_line(b);
    a:=&a;
    b:=&b;
    c:=a;
    a:=b;
    b:=c;
    dbms_output.put_line('THE VALUES OF A AND B ARE');
    dbms_output.put_line(a);
    dbms_output.put_line(b);
 end;
```

<u>OUTPUT</u>

```
Enter value for a: 5
Enter value for b: 3
THE PREVIOUS VALUES OF A AND B WERE
 15
 10
THE VALUES OF A AND B ARE
 10
 15
```

6.3 Write a PL/SQL program to find largest of two numbers

```
declare
a number;
b number;
begin
   a:=&a;
   b:=&b;
    if a=b then
       dbms_output.put_line('BOTH ARE EQUAL');
    elsif a>b then
       dbms_output.put_line('A IS GREATER');
    else
       dbms_output.put_line('B IS GREATER');
    end if;
 end;
```

<u>OUTPUT</u>

```
Enter value for a: 5
Enter value for b: 2
A IS GREATER
```

6.4 Write a PL/SQL program to find total and average of 6 subjects and display the grade

```
declare
daa number(10);
dbms number(10);
os number(10);
rmt number(10);
python number(10);
c number(10);
total number(10);
per number(10);
begin
    dbms_output.put_line('ENTER THE MARKS');
    daa:=&daa;
    dbms:=&dbms;
    os:=&os;
    rmt:=&rmt;
    python:=&python;
    c:=&c;
    total:=(daa+dbms+os+rmt+python+c);
    per:=(total/600)*100;
    if daa<40 or dbms<40 or os<40 or rmt<40 or python<40 or c<40 then
        dbms_output.put_line('FAIL');
    if per>75 then
         dbms_output.put_line('GRADE A');
    elsif per>65 and per<75 then
```

```
                dbms_output.put_line('GRADE B');
        elsif per>55 and per<65 then
                dbms_output.put_line('GRADE C');
        else
               dbms_output.put_line('INVALID INPUT');
        end if;
     dbms_output.put_line('PERCENTAGE IS '||per);
    dbms_output.put_line('TOTAL IS '||total);
end;
```

OUTPUT

```
Enter value for daa: 75
Enter value for dbms: 89
Enter value for os: 91
Enter value for rmt: 65
Enter value for python: 70
Enter value for c: 61
GRADE A
PERCENTAGE IS 75
TOTAL IS 451
```

6.5 Write a PL/SQL program to find the sum of digits in a given number

```
declare
a number;
d number:=0;
sum1 number:=0;
begin
   a:=&a;
   while a>0
    loop
      d:=mod(a,10);
      sum1:=sum1+d;
      a:=trunc(a/10);
    end loop;
   dbms_output.put_line('sum is'|| sum1);
 end;
```

<u>OUTPUT</u>

```
Enter value for a: 121
sum is:- 4
```

6.6 Write a PL/SQL program to display the number in reverse order

```
declare
a number;
rev number;
d number;
begin
    a:=&a;
    rev:=0;
    while a>0
     loop
      d:=mod(a,10);
      rev:=(rev*10)+d;
      a:=trunc(a/10);
    end loop;
  dbms_output.put_line('No is:- '|| rev);
end;
```

<u>OUTPUT</u>

```
Enter value for a: 121
No is:- 635
```

6.7 Write a PL/SQL program to check whether the given number is prime or not

```
declare
a number;
c number:=0;
i number;
begin
    a:=&a;
    for i in 1..a
     loop
        if mod(a,i)=0 then
          c:=c+1;
        end if;
     end loop;
        if c=2 then
            dbms_output.put_line(a ||'is a prime number');
        else
            dbms_output.put_line(a ||'is not a prime number');
        end if;
   end;
```

<u>OUTPUT</u>

```
Enter value for a: 11
11 is a prime number
```

6.8 Write a PL/SQL program to find factorial of given number

```
declare
n number;
f number:=1;
begin
     n:=&n;
     for i in 1..n
     loop
         f:=f*i;
     end loop;
     dbms_output.put_line('The factorial is'|| f);
 end;
```

<u>OUTPUT</u>

```
Enter value for n: 6
The factorial is 720
```

6.9 Write a PL/SQL program to calculate the area of circle for a value of radius varying from 3 to 7

```
 create table areas(radius number(10),area number(6,2));
```

PROGRAM

```
declare
pi constant number(4,2):=3.14;
radius number(5):=3;
area number(6,2);
begin
    while radius<7 loop
    area:=pi*power(radius,2);
    insert into areas values(radius,area);
    radius:=radius+1;
    end loop;
 end;
```

OUTPUT

```
SELECT * FROM AREAS;
RADIUS AREA
---------- ----------
    3        28.26
    4        50.24
    5        78.5
    6        113.04
```

6.10 Write a PL/SQL program that will accept an account number from the user,check if the users balance is less than minimum balance,only then deduct rs.100/- from the balance.this process is fired on the acct table.

- ✓ create table acct(name varchar2(10),cur_bal number(10),
  acctno number(6,2));

- ✓ insert into stud values('&sname',&rollno,&marks);
- ✓ select * from acct;

```
ACCTNO  NAME  CUR_BAL

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

    777      sirius     10000

    765       john       1000

    855        sam        500

    353       peter       800
```

PROGRAM

```
declare

mano number(5);

mcb number(6,2);

minibal constant number(7,2):=1000.00;

fine number(6,2):=100.00;

begin

       mano:=&mano;

       select cur_bal into mcb from acct where acctno=mano;

        if mcb<minibal then

              update acct set cur_bal=cur_bal-fine where acctno=mano;

        end if;

end;
```

7. Procedures and Functions

7.1
```
create or replace procedure salary(deptid number) as
begin
    update emp set sal=sal+1000 where sal>5000 AND deptno=deptid;
end;
```

7.2
```
create or replace procedure salary1(empid number) as
begin
    update emp set sal=sal+sal*(0.1) where empno=empid;
end;
```

7.3
```
create or replace procedure get_sal(dept number) as
begin
     for s in (select * from emp where deptno = dept)
     loop
        dbms_output.put_line(s.sal);
     end loop;
 end;
```

7.4
```
create or replace procedure get_nature(dept number) as
begin
   for s in (select * from emp where deptno = dept)
   loop
      dbms_output.put_line(s.job);
    end loop;
 end;
```

7.5
```
create or replace procedure dep_name(deptid number)  as
begin
   select dept.dname from dept,emp where emp.deptno=dept.deptno AND
deptno=deptid;   end;
```

8. Triggers

8.1 Write a trigger to ensure that DEPT TABLE does not contain duplicate of null values in DEPTNO column

```
CREATE OR REPLACE TRIGGER first
BEFORE INSERT on DEPT
for each row
DECLARE
    a number;
 BEGIN
     if (:new.deptno is NULL)  then
         raise_application_error(-20001,'error::deptno cannot be NULL');
     else
         select count(*) into a from DEPT where deptno=:new.deptno;
          if(a=1)  then
             raise_application_error(-20002,'error::cannot have duplicate value);
          end if;
      end if;
   END;
```

8.2 Write a trigger to carry out the following action: on deleting a deptno from **DEPT** table, all the records with that **DEPTNO** has to be deleted from the **EMP** table

```
CREATE OR REPLACE TRIGGER trgr_to_delete
BEFORE DELETE ON DEPT FOR EACH ROW
DECLARE
        CURSOR get_emp( p_deptno NUMBER ) IS
         select EMPNO, ENAME , JOB, MGR, SAL, COMM, DOB
         from EMP
        WHERE deptno=p_deptno;
BEGIN
              dbms_output.put_line( 'Delete dept = ' || :old.deptno );
              dbms_output.put_line( '- dept name = ' || :old.dname );
              dbms_output.put_line( '- dept loc  = ' || :old.loc );
     FOR get_emp_rec IN get_emp( :old.deptno ) LOOP
              dbms_output.put( '- emp ( ' || get_emp_rec.empno );
              dbms_output.put( ', ' || get_emp_rec.ename );
              dbms_output.put( ', ' || get_emp_rec.job );

              dbms_output.put( ', ' || get_emp_rec.mgr );
              dbms_output.put( ', ' || get_emp_rec.sal );
              dbms_output.put( ', ' || get_emp_rec.comm );

              dbms_output.put( ', ' || get_emp_rec.job );
             dbms_output.put_line( ' )' );
    END LOOP;
END;
```

8.3 Write a Trigger to carry out the following action: on deleting any records from the EMP table,the same values must be inserted into the LOG table.

```
 CREATE OR REPLACE TRIGGER EMP_AFTER_DELETE

 AFTER DELETE ON EMP FOR EACH ROW

 DECLARE

          CURSOR GET_EMP( P_EMPNO NUMBER ) IS

          SELECT EMPNO, ENAME , JOB, MGR, DEPTNO,SAL,
COMMISSION, DOB

           from EMP

            where EMPNO=P_EMPNO;

            BEGIN

                  Insert into LOG (DELETED_EMPNO,    DELETED_EMPNAME,
DELETED_JOB, DELETED_MGR, DELETED_DEPTNO,
DELETED_SAL, DELETED_COMMI, DELETED_DOB, DATE1)

                 values(:OLD.EMPNO, :OLD.ENAME, :OLD.JOB, :OLD.MGR,
:OLD.DEPTNO, :OLD.SAL, :OLD.COMMISSION, :OLD.DOB, SYSDATE());

                  dbms_output.put(Record of employee with employee id' ||
:OLD.DEPTNO || 'has been deleted from EMP and has been inserted to LOG '
);

             END;
```

Thank You!!