1. **a**. Consider the following database consisting of the following tables:

   Employee (ssn, first name, last name, gender, designation, doj, address)
   Employee-salary (ssn, basic pay, DA, TA, pay)
   Department (did, dname, mgrssn)
   Employee-department (ssn, deptid)

   (Create the tables with necessary primary and foreign key)
   (Enter at least five records for each relation)

   i. For each employee, retrieve the employee first name and last name
   ii. Display the distinct department count.
   iii. Retrieve all the information about employees working in Research department including the department information.
   iv. Retrieve the doj, address of employees who work for Research department using joins.
   v. Update the basic pay of each employee with 5%

```
1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));
2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);
3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));
4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));
```

```
1.  select firstname,lastnme from employee;
2.  select count(distinct dname) from department;
3.  select * from employee, department where ssn = ( select mgrssn from department where dname = 'res');
4.  select doj,  address from employee inner join department on employee.ssn=department.did;
5.  SQL> UPDATE Employee_salary SET basicpay = (basicpay*1.05);
```

b. Write a function to find the salary FOR the employee whose department name is passed as an argument (use existing table / create table with appropriate attributes).

```
create table employeepl (id int primary key , name varchar(10), salary int);

    DECLARE
     a varchar(20);
    BEGIN
      a:='&a';
      DBMS_OUTPUT.PUT_LINE(a);
      DECLARE
        cursor c_employee IS SELECT first_name,salary from employee
        WHERE designation=a;
        c_rtp c_employee%rowtype;

      BEGIN
```

```
        open c_employee;
        DBMS_OUTPUT.PUT_LINE('NAME  SALARY');
        LOOP
        FETCH c_employee INTO c_rtp;
        EXIT WHEN c_employee%notfound;
        DBMS_OUTPUT.PUT_LINE(c_rtp.first_name||' '||c_rtp.salary);
        END LOOP;
        Close c_employee;
    END;
END;
/
```

2. **a.** Consider the following database consisting of the following tables:

   Employee (ssn, first name, last name, gender, designation, doj, address)
   Employee-salary (ssn, basic pay, DA, TA, pay)
   Department (did, dname, mgrssn)
   Employee-department (ssn, deptid)

(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. Retrieve the employee name in the ascending order.
ii. Find the maximum salary in each department
iii. Retrieve all the information about employees working in Science department including the department information.
iv. Retrieve the doj, address of employees who work for Research department using joins.
v. Update the basic pay of each employee with 5%

```
1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));
2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);
3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));
4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));
```

```
1.  select firstname from employee order by(firstname);
2.  select max(mgrssn) from department;
3.  select * from department where dname='science dpt';
4.  select employee.doj,employee.address from employee inner join department on department.dname='rd';
5.  UPDATE Employee_salary SET basicpay = (basicpay*1.05) ;
```

b. Write a PL/SQL program to update the commission values for all employees with salary less than 2000 by adding Rs.1000 to existing employees. (Use existing table / create table with appropriate attributes).

```
create table employeepl (id int primary key , name varchar(10), salary int);
```

```
DECLARE
  no_count number;
BEGIN
  UPDATE employeepl set salary = salary+1000 where salary < 2000;
  if sql%notfound then
    dbms_output.put_line('No records found');
  ELSIF sql%found THEN
    no_count:=sql%rowcount;
    dbms_output.put_line(no_count|| ' records updated');
  end IF;
END;
/
```

3. **a.** Consider the following database consisting of the following tables:

   Employee (ssn, first name, last name, gender, designation, doj, address)
   Employee-salary (ssn, basic pay, DA, TA, pay)
   Department (did, dname, mgrssn)
   Employee-department (ssn, deptid)

   (Create the tables with necessary primary and foreign key)
   (Enter at least five records for each relation)

   i. For each employee, retrieve the employee ssn, first name and dname
   ii. Retrieve the doj, address of employees who work for Research department
   iii. Display the employee ssn who earning second lowest basic pay
   iv. Display the department having employee count > 5.
   v. Update the DA and TA to 3% if the basic pay is >4000.

```
CREATE TABLE Employee (ssn int , first_name varchar(6) , last_name varchar(6)
, gender char(7), designation varchar(10), doj varchar(8) , address varchar(10), PRIMARY KEY(ssn));
CREATE TABLE Employee_salary (ssn int , basic_pay int not null , DA int , TA int , Pay int,
FOREIGN KEY (ssn) REFERENCES Employee(ssn));
CREATE TABLE Department (did int primary key , dname varchar(10) not null, mgrssn int);
CREATE TABLE Employee_department ( ssn int REFERENCES Employee (ssn) , deptid int REFERENCES
Department (did) );
```

```
1. select employee.ssn , employee.firstname ,department.dname from employee inner join department on
employee.ssn=department.did;
2. select doj , address from Employee where ssn in (select ssn from Employee_department where deptid in (select
did from Department where dname = 'Research'));
3.SELECT ssn,basic_pay FROM employee_salary WHERE basic_pay in (SELECT MIN(basic_pay) FROM
employee_salary WHERE basic_pay <> (select min(basic_pay) From employee_salary));
4.select dname from Department where did in(select deptid from employee_department group by deptid having
count(*) >=5);
5.update employee_salary set DA=(DA*1.03) , TA=(TA*1.03) where basic_pay>4000;
```

b. Write a Pl/SQL program to raise the employee salary by 10%, for department id 30 and also maintain the raised details in the raise table. (use existing table / create table with appropriate attributes).

```
create table employeepl (id int primary key , name varchar(10), salary int);

DECLARE
  no_count number;
  did number;
BEGIN
  did:=&did;
  UPDATE employeepl set salary = (salary*1.1) where id = did;
  if sql%notfound then
    dbms_output.put_line('No records found');
  ELSIF sql%found THEN
    no_count:=sql%rowcount;
    dbms_output.put_line(no_count|| ' records updated');
  end IF;
END;
/
```

4. **a**. Consider the following database consisting of the following tables:
    Employee (employee name, street, city)
    Works (employee name, company name, salary, doj)
    Company (company name, city)
    Manager (employee name, manager name)

    (Create the tables with necessary primary and foreign key)
    (Enter at least five records for each relation)

    i. Find the names, streets and cities of residence of all employees who work for First Bank Corporation and earn more than 10,000.
    ii. Find those companies whose employees earn a higher salary on average than the average salary at First Bank Corporation.
    iii. Find the names of all employee who earn more than every employee of 'Samll Bank Corporation' (use natural join)
    iv. Find all employees in database who live in the city chennai and under the manager John.
    v. Find the employee name who earning the second highest salary.

```
create table Employee4 (employee_name varchar(15) primary key , street varchar(15) , doj varchar(8));

create table company (company_name varchar(10) primary key,city varchar(10));

create table works (employee_name varchar(15) references employee4(employee_name), company_name varchar(10) references company(company_name),salary int , doj varchar(10));


1.select * from employee4 where employee_name in(select employee_name from works where company_name='FBC' AND salary>10000);
```

2. select company_name,avg(salary) from works group by company_name having avg(salary) >(select avg(salary) from works where company_name='FBC');

3.select employee_name from employee4 natural join (select employee_name from works where salary > (select max(salary) from works where company_name='small bank'));

4.select employee_name from Manager where manager_name='john' AND employee_name in (select employee_name from employee4 where city='chennai');

5.select employee_name from works where salary = (select max(salary) from works where salary <> (select max(salary) from works));

b. Write a PL/SQL program to displaying top 10 employee details based on their salary using cursors(use existing table / create table with appropriate attributes).

create table employeepl (id int primary key , name varchar(10), salary int);

```
DECLARE
  no_count integer;
  cursor c_emp IS select id,name,salary from employeepl order by salary desc;
  c_rtp c_emp%rowtype;
BEGIN
  open c_emp;
  no_count:=&no_count;
  dbms_output.put_line('ID NAME SALARY');
  LOOP
  FETCH c_emp INTO c_rtp ;
  EXIT WHEN ((no_count=0) or (c_emp%notfound));
  dbms_output.put_line(c_rtp.id||' '||c_rtp.name||' '||c_rtp.salary);
  no_count:=no_count-1;
  END LOOP;
  close c_emp;
END;
/
```

5. **a**. Consider the following database consisting of the following tables:
   Employee (employee name, street, city)
   Works (employee name, company name, salary, doj)
   Company (company name, city)
   Manager (employee name, manager name)
   (Create the tables with necessary primary and foreign key)
   (Enter at least five records for each relation)
   i. Find the employees in the database who live in the same cities as the companies for which they work.
   ii. Find all employees who earn more than the average salary of all employees of that company.

iii. Write the query to find the employee name who works in the Compay 'XYX' using joins.
iv. Find the no. of employees in each company.
v. Update the salary of the employee with 7%.

```
create table Employee4 (employee_name varchar(15) primary key , street varchar(15) , doj varchar(8));

create table company (company_name varchar(10) primary key,city varchar(10));

create table works (employee_name varchar(15) references employee4(employee_name), company_name
varchar(10) references company(company_name),salary int , doj varchar(10));


1.select w.employee_name from Employee4 w where employee_name in (select employee_name from
works where company_name in (select company_name from company where city = 'chennai'));

2.select w.employee_name,w.company_name,w.salary from works w where salary > (select avg(salary)
from works where company_name = w.company_name);

3. select employee_name from Employee4 natural join (select employee_name from works where
company_name='FBC');

4.update works set salary=(salary*1.07);
```

b. Write a PL/SQL procedure raise_sal which increases the salary of an employee. It accepts an employee number and salary increase amount. It uses the employee number to find the current salary from the EMPLOYEE table and update the salary. (use existing table / create table with appropriate attributes).

```
create table employeepl (id int primary key , name varchar(10), salary int);


DECLARE
  Employee_id number;
  Amount number;
  no_count number;
BEGIN
  Employee_id:=&Employee_id;
  Amount:=&Amount;
  UPDATE employeepl set salary = (salary+Amount) where id = Employee_id;
  if sql%notfound then
    dbms_output.put_line('No records found');
  ELSIF sql%found THEN
    no_count:=sql%rowcount;
    dbms_output.put_line(no_count|| ' records updated successfully');
  end IF;
END;
/
```

6. **a**. Consider the following database consisting of the following tables:

   Department (dept id, dept name)

   Student (rollno, name, gender, mark1, mark2, mark3, dept id)

   Staff (staff id, name, designation, qualification, dept id)

   Tutor (rollno, staff id)

   (Create the tables with necessary primary and foreign key)

   (Enter at least five records for each relation)

   i. Display the student details who come under the tutor ship of the given staff name "X".
   ii. How many students are there in CSE department?
   iii. Display the staff details who work in CSE department.
   iv. Display the student details who got average >85.
   v. Display the student name who have secured second lowest total w.r.t each department.

```
create table Department6 (dept_id int primary key , dept_name varchar(10));

create table student6 (rollno int primary key , gender char(7) , mark1 int, mark2 int , mark3 int , dept_id
int REFERENCES DEPARTMENT6(dept_id));

create table staff (staff_id int primary key,name varchar(15) ,designation varchar(10) , qualification
varchar(5) , dept_id int REFERENCES DEPARTMENT6(dept_id));

create table tutor (rollno REFERENCES student6(rollno), staff_id REFERENCES staff(staff_id));

alter table student6 add name varchar(10);


1.select name from student6 where rollno in (select rollno from tutor where staff_id in (select staff_id from
staff where name='x'));

2.select count(*) as NO_STUDENTS_IN_CSE from student6 where dept_id in (select dept_id from
department6 where dept_name ='cse');

3.select * from staff where dept_id in (select dept_id from department6 where dept_name ='cse');

4.select * from student6 where ((mark1+mark2+mark3)/3)>85;

5.select * from student6 s where (mark1+mark2+mark3) in (select min((mark1+mark2+mark3)) from
student6 where (mark1+mark2+mark3) <> (select min((mark1+mark2+mark3)) from student6 where
dept_id=s.dept_id) AND dept_id=s.dept_id);
```

b. Write a PL/SQL program to calculate the grade for each student using case statement. Assume your own grade range. (use existing table / create table with appropriate attributes).

```
DECLARE

grade char(1);

BEGIN

grade := '&Enter_grade';
```

```
case grade

when 'A' then dbms_output.put_line('Excellent');

when 'B' then dbms_output.put_line('Very Good');

when 'C' then dbms_output.put_line('Good');

when 'D' then dbms_output.put_line('Average');

when 'F' then dbms_output.put_line('Fail');

end case;

END;

/
```

7. **a**. Consider the following database consisting of the following tables:
   Department (dept id, dept name)
   Student (rollno, name, gender, mark1, mark2, mar3, total, average, dept id)
   Staff (staff id, name, designation, qualification, dept id)
   Tutor (rollno, staff id)

   (Create the tables with necessary primary and foreign key)
   (Enter at least five records for each relation)

   i. Display the student details who got greater than overall average marks of their department.
   ii. Display the staff details who work in CSE department.
   iii. How many different designations and departments are there?
   iv. Display the no.of. Student under the department 'cse'.
   v. Update the qualification of staff from Assistant Professor to Professor.

```
create table Department6 (dept_id int primary key , dept_name varchar(10));

create table student6 (rollno int primary key , gender char(7) , mark1 int, mark2 int , mark3 int , dept_id int
REFERENCES DEPARTMENT6(dept_id));

create table staff (staff_id int primary key,name varchar(15) ,designation varchar(10) , qualification varchar(5) ,
dept_id int REFERENCES DEPARTMENT6(dept_id));

create table tutor (rollno REFERENCES student6(rollno), staff_id REFERENCES staff(staff_id));

alter table student6 add name varchar(10);


1.select * from student6 s where (mark1+mark2+mark3) >(select avg((mark1+mark2+mark3)) from student6
where dept_id = s.dept_id);

2.select * from staff where dept_id in (select dept_id from department6 where dept_name ='cse');

3.select count(distinct department6.dept_id) , count(distinct staff.designation) from Department6  , staff ;

4 update staff set designation = 'professor' where designation = 'Assistant Professor';
```

b. Write a PL/SQL program to display the list of marks for a student by explicitly specifying his roll_no. (use existing table / create table with appropriate attributes).

USE : Table student table from above

```
DECLARE
   no_count integer;
   cursor c_emp IS select rollno,name,mark1,mark2,mark3 from student6;
   c_rtp c_emp%rowtype;
BEGIN
   open c_emp;
   dbms_output.put_line('ID NAME Mark1 Mark2 Mark3');
   LOOP
   FETCH c_emp INTO c_rtp ;
   EXIT WHEN ((c_emp%notfound));
   dbms_output.put_line(c_rtp.rollno||' '||c_rtp.name||' '||c_rtp.mark1||' '||c_rtp.mark2||' '||c_rtp.mark3);
   END LOOP;
   close c_emp;
END;
/
```

8. **a.** Consider the following database consisting of the following tables:
   Branch (bname, bcity, assets)
   Account (ano, starting date, bname, balance)
   Customer (cusid, name, address)
   Deposit (ano, cusid, bname)
   Loan (lno, banme, amt)
   Borrower (cusid, lno)
   Transaction (ano, amount, mode, date of trans)

(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. Display the details of the loan whose branch is at 'Salem'
ii. Find the average account balance at each branch and display only if it is greater than 10000.
iii. Find the largest account balance in the bank.
iv. Find the names of all customers who have a loan with lno and amount(joins)
v. Update the loan amount to 5% if the loan amount is greater than 50000

```
CREATE TABLE BRANCH (
branch_name varchar(20),
branch_city varchar(20),
assets real,
constraint bpk PRIMARY KEY(branch_name));

CREATE TABLE CUSTOMER(
customer_name varchar(20),
customer_street varchar(20),
customer_city varchar(20),
constraint cpk PRIMARY KEY(customer_name));

CREATE TABLE ACCOUNT (
accno int,
branch_name varchar(20),
balance real,
constraint apk PRIMARY KEY(accno),
constraint afk FOREIGN KEY(branch_name) references BRANCH(branch_name));

CREATE TABLE DEPOSITOR (
customer_name varchar(20),
Accno int,
constraint dpk PRIMARY KEY(customer_name, accno),
constraint dfk1 FOREIGN KEY(customer_name) references CUSTOMER(customer_name),
constraint dfk2 FOREIGN KEY(accno) references ACCOUNT(accno) On Delete Cascade);

CREATE TABLE LOAN (
loan_number int,
branch_name varchar(20),
amount real,
constraint lpk PRIMARY KEY(loan_number),
constraint lfk FOREIGN KEY(branch_name) references BRANCH(branch_name));

CREATE TABLE BORROWER (
customer_name varchar(20),
Loan_number int,
constraint bpk PRIMARY KEY(customer_name, Loan_number), constraint bfk1 FOREIGN
KEY(customer_name) references CUSTOMER(customer_name),
constraint bfk2 FOREIGN KEY(Loan_number) references LOAN(Loan_number));

1.select * from loan where baname in (select bname from branch where bcity='salem');

2.select avg(balance) as AVERAGE BALANCE from Account group by bname having avg(balance) > 10000;

3.select max(balance) as largest balance from Account ;

4.select name from customer  natural join (select cusid from Borrower where lno in (select lno from loan));

5.update loan set amt=(amt*1.05) where amt>50000;
```

b. Write a PL/SQL code block that will accept an account number from the user and debit an amount of Rs. 2000 from the account, if the account has a minimum balance of 500 after the amount is debited. (use existing table / create table with appropriate attributes).

```
DECLARE
   Employee_id number;
   no_count number;
BEGIN
```

```
    Employee_id:=&Employee_id;
    UPDATE employeepl set salary = (salary-2000) where id = Employee_id AND salary = 2500;
    if sql%notfound then
      dbms_output.put_line('No records found');
    ELSIF sql%found THEN
      no_count:=sql%rowcount;
      dbms_output.put_line(no_count|| ' records updated successfully');
    end IF;
  END;
  /
```

9. **a.** Consider the insurance database given below. The primary keys are made bold and the data types are specified.
PERSON(**driver_id**:string , name:string , address:string)
CAR(**regno**:string , model:string , year:int)
ACCIDENT(**report_number**:int , accd_date:date , location:string)
OWNS(**driver_id**:string , **regno**:string)
PARTICIPATED(**driver_id**:string , **regno**:string , **report_number**:int , damage_amount:int)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)
    i. Find the person who owns more than two cars.
    ii. Find the total damage amount made by the driver name "XYZ"
    iii. Update the damage amount to 25000 for the car with a specific Regno in the ACCIDENT table with report number 12.
    iv. Add a new accident to the database.
    v. Find the total number of people who owned cars that were involved in accidents in the year 2012.

Date formate = '15-feb-2001'

```
create table person(driver_id varchar2(25) primary key, name varchar2(30), address varchar2(30));

create table car(regno varchar2(10) primary key, model varchar2(25), year varchar2(4));

create table accident( report_number int primary key, accd_date date, location varchar2(25));

create table owns(driver_id varchar2(25) references person(driver_id), regno varchar2(10) references car(regno));

create table participated(driver_id varchar2(25) references person(driver_id), regno varchar2(10) references car(regno), report_number int references accident(report_number), damage_amount int);


1.select name from person where driver_id in (select driver_id from owns group by driver_id having count(*)>2);

2.select sum(damage_amount) from participate where driver_id in (select driver_id from person where name = 'xyz');

3.update participate set damage_amount = 25000 where report_number = 12;

4.insert into accident values (16,'26-06-2022','sk-city');

5.select count(*) from partipated where report_number in (select report_number from account where accd_date like '%2012');
```

b. Consider the EMPLOYEE (ENO, SALARY, ENAME) TABLE. Write a procedure raise_sal which increases the salary of an employee. It accepts an employee number AND salary increase amount. It uses the employee number to find the current salary FROM the EMPLOYEE TABLE AND UPDATE the salary.

```
create table employeepl (id int primary key , name varchar(10), salary int);
```

```
DECLARE
   Employee_id number;
   Amount number;
   no_count number;
BEGIN
   Employee_id:=&Employee_id;
   Amount:=&Amount;
   UPDATE employeepl set salary = (salary+Amount) where id = Employee_id;
   if sql%notfound then
      dbms_output.put_line('No records found');
   ELSIF sql%found THEN
      no_count:=sql%rowcount;
      dbms_output.put_line(no_count|| ' records updated successfully');
   end IF;
END;
/
```

10. **a.** Consider the following database consisting of the following tables:

Employee (ssn, first name, last name, gender, designation, doj, address)
Employee-salary (ssn, basic pay, DA, TA, pay)
Department (did, dname, mgrssn)
Employee-department (ssn, deptid)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. For each employee, retrieve the employee first name and last name
ii. Display the distinct department count.
iii. Retrieve all the information about employees working in Research department including the department information.
iv. Retrieve the doj, address of employees who work for Research department using joins.
iv. Update the basic pay of each employee with 5%

```
1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));
2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);
3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));
4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));
```

```
1.select firstname,lastnme from employee;
```

2.select count(distinct dname) from department;

3.select * from Employee where ssn in (select ssn from employee_department where deptid in (select did from department where dname='resarch'));

4. select doj , address from employee natural join (select ssn from employee-department where deptid in (select did from department where dname='research'));

5.update Employee_salary set basic_pay = (basic_pay*1.05);

b. Write a PL/SQL program to check whether the given number is ODD or EVEN.

```
DECLARE
n number := &n;
a number;
BEGIN
   a := mod(n,2);
   if a=0
   then
   dbms_output.put_line('even');
   else
   dbms_output.put_line('odd');
   end if;
END;
/
PL/SQL procedure successfully completed.
```

11. **a.** Consider the following database consisting of the following tables:
Employee (ssn, first name, last name, gender, designation, doj, address)
Employee-salary (ssn, basic pay, DA, TA, pay)
Department (did, dname, mgrssn)
Employee-department (ssn, deptid)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. Retrieve the employee name in the ascending order.
ii. Find the maximum salary in each department
iii. Retrieve all the information about employees working in Science department including the department information.
iv. Retrieve the doj, address of employees who work for Research department using joins.
iv. Update the basic pay of each employee with 5%

1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));

2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);

3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));

4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));

1.select firstname from employee order by(firstname);

2.select max(pay) from employee_salary where ssn in (select mgrssn from department);

3.select * from Employee where ssn in (select ssn from employee_department where deptid in (select did from department where dname='science'));

4. select doj , address from employee natural join (select ssn from employee-department where deptid in (select did from department where dname='research'));

```
5.update Employee_salary set basic_pay = (basic_pay*1.05);
```

b. Write a PL/SQL program to find the given year is leap year or not.

```
SQL> set serveroutput on;
SQL> declare
2          year number := &year;
3          begin
4          if mod(year,4) = 0
5          then
6          dbms_output.put_line(year || 'is leap year' );   7  else
8          dbms_output.put_line(year || 'is not leap year');
9          end if;
10         end;
11         /
Enter value for year: 1600 old   2: year number := &year; new   2: year number := 1600;
1600is leap year

PL/SQL procedure successfully completed.
```

12. **a.** Consider the following database consisting of the following tables:
Employee (ssn, first name, last name, gender, designation, doj, address)
Employee-salary (ssn, basic pay, DA, TA, pay)
Department (did, dname, mgrssn)
Employee-department (ssn, deptid)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. For each employee, retrieve the employee ssn, first name and dname
ii. Retrieve the doj, address of employees who work for Research department
iii. Display the employee ssn who earning second lowest basic pay
iv. Display the department having employee count > 5.
v. Update the DA and TA to 3% if the basic pay is >4000.

```
1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));
2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);
3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));
4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));

1. select employee.ssn , employee.firstname ,department.dname from employee inner join department on employee.ssn=department.did;
2. select doj , address from Employee where ssn in (select ssn from Employee_department where deptid in (select did from Department where dname = 'Research'));
3.SELECT ssn,basic_pay FROM employee_salary WHERE basic_pay in (SELECT MIN(basic_pay) FROM employee_salary WHERE basic_pay <> (select min(basic_pay) From employee_salary));
4.select dname from Department where did in(select deptid from employee_department group by deptid having count(*) >=5);
5.update employee_salary set DA=(DA*1.03) , TA=(TA*1.03) where basic_pay>4000;
```

b. Write a PL/SQL program to find the factorial of a given number.

```
SQL> set serveroutput on ;
SQL> declare
2  fact number := 1;
3  n number := &n;
4  begin
5  while n>0
6  loop
7  fact := n* fact;
8  n := n-1;
9  end loop;
10 dbms_output.put_line(fact);
11 end;
12 /
Enter value for n: 5 old   3:
n number := &n; new   3: n
number := 5;
120

PL/SQL procedure successfully completed.
```

13. **a**. Consider the following database consisting of the following tables:
Employee (employee name, street, city)
Works (employee name, company name, salary, doj)
Company (company name, city)
Manager (employee name, manager name)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. Find the names, streets and cities of residence of all employees who work for First Bank Corporation and earn more than 10,000.
ii. Find those companies whose employees earn a higher salary on average than the average salary at First Bank Corporation.
iii. Find the names of all employee who earn more than every employee of 'Samll Bank Corporation' (use natural join)
iv. Find all employees in database who live in the city chennai and under the manager John.
iv. Find the employee name who earning the second highest salary.

```
create table Employee4 (employee_name varchar(15) primary key , street varchar(15) , doj varchar(8));

create table company (company_name varchar(10) primary key,city varchar(10));

create table works (employee_name varchar(15) references employee4(employee_name), company_name varchar(10)
references company(company_name),salary int , doj varchar(10));


1.select * from employee4 where employee_name in(select employee_name from works where company_name='FBC'
AND salary>10000);
2. select company_name,avg(salary) from works group by company_name having avg(salary) >(select avg(salary) from
works where company_name='FBC');
3.select employee_name from employee4 natural join (select employee_name from works where salary > (select
max(salary) from works where company_name='small bank'));
```

```
4.select employee_name from Manager where manager_name='john' AND employee_name in (select employee_name
from employee4 where city='chennai');
5.select employee_name from works where salary = (select max(salary) from works where salary <> (select max(salary)
from works));
```

b. Create a TRIGGER to ensure that department TABLE does NOT duplicate VALUES IN dept_no column

14. **a**. Consider the following database for a banking enterprise.

account(account_number, branch_name, balance)
branch (branch_name, branch_city, assets)
customer (customer_name customer_street, customer_city)
loan (loan_number, branch_name, amount)
depositor((customer_name, account_number)
borrower(customer_name, loan_number)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. List all branch names and their assests
ii. List all accounts of Brooklyn branch
iii. Change the size of the branch_city to varchar(20).
iv. For all customers who have a loan from the bank, find their names and loan numbers with the attribute loan_number replaced by loan_id.
v. Select the names of customers who have a loan at the bank, and whose names are neither Smith nor Jones

```
CREATE TABLE BRANCH (
branch_name varchar(20),
branch_city varchar(20),
assets real,
constraint bpk PRIMARY KEY(branch_name));

CREATE TABLE CUSTOMER(
customer_name varchar(20),
customer_street varchar(20),
customer_city varchar(20),
constraint cpk PRIMARY KEY(customer_name));

CREATE TABLE ACCOUNT (
accno int,
branch_name varchar(20),
balance real,
constraint apk PRIMARY KEY(accno),
constraint afk FOREIGN KEY(branch_name) references BRANCH(branch_name));

CREATE TABLE DEPOSITOR (
customer_name varchar(20),
Accno int,
constraint dpk PRIMARY KEY(customer_name, accno),
constraint dfk1 FOREIGN KEY(customer_name) references CUSTOMER(customer_name),
constraint dfk2 FOREIGN KEY(accno) references ACCOUNT(accno) On Delete Cascade);

CREATE TABLE LOAN (
loan_number int,
branch_name varchar(20),
amount real,
constraint lpk PRIMARY KEY(loan_number),
constraint lfk FOREIGN KEY(branch_name) references BRANCH(branch_name));

CREATE TABLE BORROWER (
customer_name varchar(20),
Loan_number int,
constraint bpk PRIMARY KEY(customer_name, Loan_number), constraint bfk1 FOREIGN
KEY(customer_name) references CUSTOMER(customer_name),
constraint bfk2 FOREIGN KEY(Loan_number) references LOAN(Loan_number));
```

```
1.select branch_name , asserts from branch;
2.select account_number from account where branch_name='brooklyn';
3.alter table branch add branch_city varchar(20);
4.alter table loan rename column loan_number to loan_id;
5.select customer_name from customer where name not in ('smith','jones');
```

b. Write a pl/sql program to find the total and average of 6 subjects then display the grade

```
SQL> set serveroutput on;
SQL> DECLARE
2  a number := &subject1;
3  b number := &subject2;
4  c number := &subject3;
5  d number := &subject4;
6  e number := &subject5;
7  f number := &subject6;
8  sumOf6 NUMBER;
9  avgOf6 NUMBER;
10
11  BEGIN
12  sumOf6 := a + b + c + d + e + f;
13  avgOf6 := sumOf6 / 6;
14  dbms_output.Put_line('Sum = '
15  ||sumOf6);
16  dbms_output.Put_line('Average = '
17  ||avgOf6);
18  END;
19  /
```

```
Enter value for subject1: 1 old   2: a
number := &subject1; new   2: a
number := 1; Enter value for
subject2: 2 old   3: b number :=
&subject2; new   3: b number := 2;
Enter value for subject3: 3 old   4: c
number := &subject3; new   4: c
number := 3; Enter value for
subject4: 4 old   5: d number :=
&subject4; new   5: d number := 4;
Enter value for subject5: 5 old   6: e
number := &subject5; new   6: e
number := 5; Enter value for
subject6: 6
old   7: f number := &subject6; new
7: f number := 6;
Sum = 21
Average = 3.5

PL/SQL procedure successfully completed.
```

15. **a**. Consider the following database consisting of the following tables:

Party (pid, pname, leader)
Constituency (cid, cname)
Contestant (ctid, ctname, ctaddr)
Election (ctid, number of votes, pname, cname)

(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. Display the contestant details if they secured greater than 10,000 votes.
ii. Find the number of contestants, constituency wise.
iii. Display the winner details in each constituency.
iv. Display the difference of votes in each constituency w.r.t first and second position.
v. Find the contestant who got the least number of votes

```
1.select * from contestant where ctid in (select ctid from election where no_votes >10000);
2.select cname,count(*) as NO_OF_CONTESTANT from election group by cname;
3. select * from contastant where ctid in (select ctid from election where NO_OF_VOTES in ( select max(no_of_votes) from election group by cname));
4.
SELECT constituency
, MAX(votes) AS votes1
, MIN(votes) AS votes2
, MAX(votes)-MIN(votes) AS votes_diff
FROM election e
WHERE EXISTS
(
  SELECT 1
  FROM election e2
  WHERE e2.constituency = e.constituency
    AND e2.votes > e.votes
```

```
                HAVING COUNT(*) < 2
               )
GROUP BY constituency
HAVING (MAX(votes) - MIN(votes)) < 50
ORDER BY votes_diff DESC
```

5.select * from contastant where ctid in (select ctid from election where NO_OF_VOTES in (select min(no_of_votes) from election));

b. Write a pl/sql code block to calculate the area of a circle. Store the radius AND the corresponding area IN an empty TABLE named area, consisting of two columns radius AND area.

Create table area_of_circle(radius number, area number);

```
DECLARE
   radius integer;
   area number;
BEGIN
   radius := &radius;
   area:=(radius*radius*3.14);
   insert into circle values(radius,area);
END;
/
```

16. **a**. Consider the insurance database given below. The primary keys are made bold and the data types are specified.

PERSON(driver_id:string , name:string , address:string)
CAR(regno:string , model:string , year:int)
ACCIDENT(report_number:int , accd_date:date , location:string)
OWNS(driver_id:string , regno:string)
PARTICIPATED(driver_id:string , regno:string , report_number:int , damage_amount:int)

i. Create the above tables by properly specifying the primary keys and foreign keys.
ii. Enter at least five tuples for each relation.
iii. Demonstrate how you a.Update the damage amount for the car with specific regno in the accident with report number 12 to 25000.
iv. Add a new accident to the database.
v. Find the total number of people who owned cars that were involved in accidents in the year 2008.

b. Write a function to find the salary for the employee whose department name is passed as an argument (use existing table / create table with appropriate attributes).

```
DECLARE
 a varchar(20);
BEGIN
   a:='&a';
   DBMS_OUTPUT.PUT_LINE(a);
```

```
DECLARE
   cursor c_employee IS SELECT first_name,salary from employee
   WHERE designation=a;
   c_rtp c_employee%rowtype;

BEGIN

   open c_employee;
   DBMS_OUTPUT.PUT_LINE('NAME  SALARY');
   LOOP
   FETCH c_employee INTO c_rtp;
   EXIT WHEN c_employee%notfound;
   DBMS_OUTPUT.PUT_LINE(c_rtp.first_name||' '||c_rtp.salary);
   END LOOP;
END;
END;
/
```

17. **a.** Consider the following relations for a order processing database application in a company.

CUSTOMER(custno:int , cname:string , city:string)
ORDER(orderno:int , odate:date , custno:int , ord_amt:int)
ORDER_ITEM(orderno:int , itemno:int , quantity:int)
ITEM(itemno:int , unitprice:int)
SHIPMENT(orderno:int , warehouseno:int , ship_date:date)
WAREHOUSE(warehouseno:int , city:string)

i. Create the above tables by properly specifying the primary keys and foreign keys.
ii. Enter at least five tuples for each relation.
iii. Produce a listing: custname , No_of_orders , Avg_order_amount , where the middle column is the total number of orders by the customer and the last column is the average order amount for that customer.
iv. List the orderno for orders that were shipped from all the warehouses that the company has in a specific city.
v. Demonstrate the deletion of an item from the ITEM table and demonstrate a method of handling the rows in the ORDER_ITEM table that contains this particular item.

b. Write a PL/SQL program to update the commission values for all employees with salary less than 2000 by adding Rs.1000 to existing employees. (Use existing table / create table with appropriate attributes).

```
DECLARE
   no_count number;
BEGIN
   UPDATE employeepl set salary = salary+1000 where salary < 2000;
   if sql%notfound then
      dbms_output.put_line('No records found');
   ELSIF sql%found THEN
      no_count:=sql%rowcount;
      dbms_output.put_line(no_count|| ' records updated');
```

```
  end IF;
END;
/
```

18.**a**. Consider the following database of student enrollment in courses and books adopted for that course.

STUDENT(regno:string , name:string , major:string , bdate:date)
COURSE (courseno:int , cname:string , dept:string)
ENROLL(regno:string , courseno:int , sem:int , marks:int)
BOOK_ADOPTION(courseno:int , sem:int , book_isbn:int)
TEXT(book_isbn:int , book_title:string , publisher:string , author:string)

i. Create the above tables by properly specifying the primary keys and foreign keys.
ii. Enter atleast five tuples for each relation.
iii. Demonstrate how you add a new text book to the database and make this book to be adopted by some department.
iv. Produce a list of text books (includes courseno , book_isbn , book_title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.
v. List any department that has all its books published by a specific publisher.

b. Write a Pl/SQL program to raise the employee salary by 10%, for department id 30 and also maintain the raised details in the raise table. (use existing table / create table with appropriate attributes).

```
create table employeepl (id int primary key , name varchar(10), salary int);

DECLARE
  no_count number;
  did number;
BEGIN
  did:=&did;
  UPDATE employeepl set salary = (salary*1.1) where id = did;
  if sql%notfound then
    dbms_output.put_line('No records found');
  ELSIF sql%found THEN
    no_count:=sql%rowcount;
    dbms_output.put_line(no_count|| ' records updated');
  end IF;
END;
/
```

19. **a.** Consider the following database consisting of the following tables:

Employee (employee name, street, city)
Works (employee name, company name, salary, doj)
Company (company name, city)
Manager (employee name, manager name)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

vi. Find the names, streets and cities of residence of all employees who work for First Bank Corporation and earn more than 10,000.

vii. Find those companies whose employees earn a higher salary on average than the average salary at First Bank Corporation.
viii. Find the names of all employee who earn more than every employee of ‘Samll Bank Corporation’ (use natural join)
ix. Find all employees in database who live in the city chennai and under the manager John.
x. Find the employee name who earning the second highest salary.

b. Write a PL/SQL program to displaying top 10 employee details based on their salary using cursors(use existing table / create table with appropriate attributes).

```
create table employeepl (id int primary key , name varchar(10), salary int);

        DECLARE
          no_count integer;
          cursor c_emp IS select id,name,salary from employeepl order by salary desc;
          c_rtp c_emp%rowtype;
        BEGIN
          open c_emp;
          no_count:=&no_count;
          dbms_output.put_line('ID NAME SALARY');
          LOOP
          FETCH c_emp INTO c_rtp ;
          EXIT WHEN ((no_count=0) or (c_emp%notfound));
          dbms_output.put_line(c_rtp.id||' '||c_rtp.name||' '||c_rtp.salary);
          no_count:=no_count-1;
          END LOOP;
          close c_emp;
        END;
        /
```

20. **a.** Consider the following database consisting of the following tables:
Employee (Name, SSN, Address, Sex, Salary, Dno)
Department (Dname, Dnumber, MGRSSN, MGRSTART Date)
Dept-Locations (Dnumber, Dlocations)
Project (Pname, Pnumber, Plocations, Dnum)
Works-On (ESSN, PNo, Hours)
Dependent (ESSN, Dependent-name, Sex, Bdate, Relationship)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)
Give the queries in SQL

i. Retrieve the names and address of employees who work for “Research”Department.
ii. List all the project names on which employee “Smith” is working.
iii. Retrieve all employees who either work in department 4 or make over 25000 per year or work in department 5 and make over 30,000.
iv. Retrieve the SSN of all employees who either work in department 5 or directly supervise an employee who works in department number
v. Retrieve names of each employee who have only daughter dependent.

b. Write a PL/SQL procedure raise_sal which increases the salary of an employee. It accepts an employee number and salary increase amount. It uses the employee number to find the current salary from the EMPLOYEE table and update the salary. (use existing table / create table with appropriate attributes).

```
create table employeepl (id int primary key , name varchar(10), salary int);
```

```
DECLARE
   Employee_id number;
   Amount number;
   no_count number;
BEGIN
   Employee_id:=&Employee_id;
   Amount:=&Amount;
   UPDATE employeepl set salary = (salary+Amount) where id = Employee_id;
   if sql%notfound then
      dbms_output.put_line('No records found');
   ELSIF sql%found THEN
      no_count:=sql%rowcount;
      dbms_output.put_line(no_count|| ' records updated successfully');
   end IF;
END;
/
```

21. **a**. Assume the three relations given below and write the queries
    STUDENT whose attributes are Stud No and StudName,
    ASSIGNED_TO whose attributes are Stud No and Project No
    PROJECT whose attributes are Project No and Project area.
   (Create the tables with necessary primary and foreign key)
   (Enter at least five records for each relation)
   i. Obtain Stud No and Stud Name of Name of all those students who are working on all those students who are working on database projects.
   ii. Obtain Stud No and Stud Name of all those students who are working on both the projects having project numbers.
   iii. Obtain Stud No and Stud Name of all those students who do not work on the project number
   iv. Obtain Stud No and Stud Name of all students other than the students with Stud No 54 who work on at least one project.
   v. Alter the table by adding column student dept in student table.

   b. Write a PL/SQL program to calculate the grade for each student using case statement. Assume your own grade range. (use existing table / create table with appropriate attributes).

```
 set serveroutput on;
declare
grade char(1);
begin
```

```
        grade := '&Enter_grade';
case grade
        when 'A' then dbms_output.put_line('Excellent');
        when 'B' then dbms_output.put_line('Very Good');
        when 'C' then dbms_output.put_line('Good');
        when 'D' then dbms_output.put_line('Average');
        when 'F' then dbms_output.put_line('Fail');
end case;
end;
/
```

22. **a.** Consider the following schema and write the queries for the g given below:

    SAILORS(Sid, Sname, rating, age)
    BOATS(bid, bname, color)
    RESERVES(sid, bid,day)
    (Create the tables with necessary primary and foreign key)
    (Enter at least five records for each relation)

    i. Find names of sailors who reserved green boat
    ii. Find the colors of boats reserved by "Ramesh"
    iii. Find names of sailors who have reserved a red or a green boat.
    iv. Find the "sids" of sailors with age over 20 who have not registered a red boat.
    v. Find the maximum number or registrations for the boat which is red boat or green boat

    b. Write a PL/SQL program to display the list of marks for a student by explicitly specifying his roll_no. (use existing table / create table with appropriate attributes).

    USE : Table student table from above

```
DECLARE
  no_count integer;
  cursor c_emp IS select rollno,name,mark1,mark2,mark3 from student6;
  c_rtp c_emp%rowtype;
BEGIN
  open c_emp;
  dbms_output.put_line('ID NAME Mark1 Mark2 Mark3');
  LOOP
  FETCH c_emp INTO c_rtp ;
  EXIT WHEN ((c_emp%notfound));
  dbms_output.put_line(c_rtp.rollno||' '||c_rtp.name||' '||c_rtp.mark1||' '||c_rtp.mark2||' '||c_rtp.mark3);
  END LOOP;
  close c_emp;
END;
/
```

23. **a.** Consider the following schema

    Emp (eid: integer, ename: string, age: integer, sal: real)
    Works (eid: integer, pid: integer, no-of-hrs: integer, did: integer)
    Dept (did: integer, dname:string, mgrid: integer)

Project (pid: integer, Pname: string)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)
Write SQL statement to

i. Give every employee of did = ‘6’ and 10% raise in salary.
ii. Add ‘John’ as an employee with eid = ‘99’, age = ‘30’, and salary =’15,000’.
iii. Delete the ‘Research’ department and explain what happens when this statement is executed.
iv. Write the query to find the employee name who works in the project ‘XYX’ using joins.
v. Display all data of employee who work for projects in more than one departments

b. Write a PL/SQL code block that will accept an account number from the user and debit an amount of Rs. 2000 from the account, if the account has a minimum balance of 500 after the amount is debited. (use existing table / create table with appropriate attributes).

```
DECLARE
  Employee_id number;
  no_count number;
BEGIN
  Employee_id:=&Employee_id;
  UPDATE employeepl set salary = (salary-2000) where id = Employee_id AND salary = 2500;
  if sql%notfound then
    dbms_output.put_line('No records found');
  ELSIF sql%found THEN
    no_count:=sql%rowcount;
    dbms_output.put_line(no_count|| ' records updated successfully');
  end IF;
END;
/
```

24. **a.** The following are maintained by abook dealer.
AUTHOR(author_id:int , name:string , city:string , country:string)
PUBLISHER(publisher_id:int , name:string , city:string , country:string)
CATALOG(book_id:int , title:string , author_id:int , publisher_id:int , category_id:int , year:int , price:int)
CATEGORY(category_id:int , description:string)
ORDER_DETAILS(order_no:int , book_id:int , quantity:int)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)
Write SQL statement to

i. Give the details of the authors who have 2 or more books in the catalog and the price of the books is greater than the average price of the books in the catalog and the year of publication is after 2000.
ii. Find the author of the book that has maximum sales.
iii. Demonstrate how you increase the price of books published by a specific publisher by 10%.

iv. Find the author who published in the country “India” and publisher “ Tata MachrwHill” in the category “FICTION”.
v. Find the number of quantities for the book id “5555”.

b. Consider the EMPLOYEE (ENO, SALARY, ENAME) TABLE. Write a procedure raise_sal which increases the salary of an employee. It accepts an employee number AND salary increase amount. It uses teh employee number to find the current salary FROM the EMPLOYEE TABLE AND UPDATE the salary.

25. **a**. Consider the following database for a banking enterprise.
account(account_number, branch_name, balance)
branch (branch_name, branch_city, assets)
customer (customer_name customer_street, customer_city)
loan (loan_number, branch_name, amount)
depositor((customer_name, account_number)
borrower(customer_name, loan_number)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)
i. List all loans with amount > 1000.
ii. List all accounts of Perryridge branch with balance < 1000.
iii. Add a column phoneNo to customer table.
iv. Find all customers who have an account but no loan at the bank.
v. Find those branches where the average accounts balance is more than Rs. 1200.

b. Write a PL/SQL code block to display the customer details having both an account and a loan in the bank (use existing table / create table with appropriate attributes).

USE : Above tables for this pl/sql

```
DECLARE
   no_count integer;
   cursor c_customer IS select customer_number,customer_name,customer_city from customer
   where customer_name in (select customer_name from depositor) and customer_name in (select
customer_name from borrower);
   c_rtp c_emp%rowtype;
BEGIN
   open c_emp;
   dbms_output.put_line('ID NAME city');
   LOOP
   FETCH c_emp INTO c_rtp ;
   EXIT WHEN ((c_emp%notfound));
   dbms_output.put_line(customer_number||' '||c_rtp.customer_name||' '||c_rtp.customer_city);
   END LOOP;
   close c_emp;
END;
/
```

26. **a.** Consider the following database for a banking enterprise.
account(account_number, branch_name, balance)
branch (branch_name, branch_city, assets)
customer (customer_name customer_street, customer_city)

loan (loan_number, branch_name, amount)
depositor((customer_name, account_number)
borrower(customer_name, loan_number)

(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. List all branch names and their assests
ii. List all accounts of Brooklyn branch
iii. Change the size of the branch_city to varchar(20).
iv. For all customers who have a loan from the bank, find their names and loan numbers with the attribute loan_number replaced by loan_id.
v. Select the names of customers who have a loan at the bank, and whose names are neither Smith nor Jones

b. Write a PL/SQL code block that will accept an account number from the user and debit an amount of Rs. 2000 from the account, if the account has a minimum balance of 500 after the amount is debited. (use existing table / create table with appropriate attributes).

```
DECLARE
   Employee_id number;
   no_count number;
BEGIN
   Employee_id:=&Employee_id;
   UPDATE employeepl set salary = (salary-2000) where id = Employee_id AND salary = 2500;
   if sql%notfound then
      dbms_output.put_line('No records found');
   ELSIF sql%found THEN
      no_count:=sql%rowcount;
      dbms_output.put_line(no_count|| ' records updated successfully');
   end IF;
END;
/
```

27. **a**. Consider the following database consisting of the following tables:
Party (pid, pname, leader)
Constituency (cid, cname)
Contestant (ctid, ctname, ctaddr)
Election (ctid, number of votes, pname, cname)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. Display the contestant details if they secured greater than 10,000 votes.
ii. Find the number of contestants, constituency wise.
iii. Display the winner details in each constituency.
iv. Display the difference of votes in each constituency w.r.t first and second position.
v. Find the contestant who got the least number of votes

b. Create a TRIGGER that raises an user defined error message and does not allow update and insertion. (use existing table / create table with appropriate attributes).

28. **a.** Consider the following database consisting of the following tables:
Employee (ssn, first name, last name, gender, designation, doj, address)

Employee-salary (ssn, basic pay, DA, TA, pay)
Department (did, dname, mgrssn)
Employee-department (ssn, deptid)

(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. For each employee, retrieve the employee ssn, first name and dname
ii. Retrieve the doj, address of employees who work for Research department

1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));
2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);
3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));
4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));

i. **Select employee.ssn, employee.first_name, department.dname from employee, department where employee.ssn = department.mgrssn;**
ii. **select employee.doj, employee.address from employee where ssn = (select mgrssn from department where dname = 'res');**

b. Create emp_table, and backup_tbl with the following attributes
emp_table(empid, empname, salary)
backup_tbl(empid, empname, salary, operation)
Write a trigger function which will be invoked before the insert, delete or update operation. It does the following:

i. Before delete operation, it inserts the old data into backup_tbl.
ii. Before update operation, it inserts the old data into backup_tbl.

29**. a**. Consider the following database consisting of the following tables:
Department (dept id, dept name)
Student (rollno, name, gender, mark1, mark2, mark3, dept id)
Staff (staff id, name, designation, qualification, dept id)
Tutor (rollno, staff id)

(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. Display the student details who come under the tutor ship of the given staff name "X".
ii. How many students are there in CSE department?

create table Department6 (dept_id int primary key , dept_name varchar(10));

create table student6 (rollno int primary key , gender char(7) , mark1 int, mark2 int , mark3 int , dept_id int REFERENCES DEPARTMENT6(dept_id));

create table staff (staff_id int primary key,name varchar(15) ,designation varchar(10) , qualification varchar(5) , dept_id int REFERENCES DEPARTMENT6(dept_id));

create table tutor (rollno REFERENCES student6(rollno), staff_id REFERENCES staff(staff_id));

alter table student6 add name varchar(10);

i. **select * from student6 where rollno = (select rollno from tutor where staff_id = (select staff_id from staff where sname = 'bala'));   // check whether it is right or wrong**

ii. **select count(rollno) from student6 where dept_id = (select dept_id from department6 where dept_name = 'cse');**

b. Create emp_table, and backup_tbl with the following attributes

emp_table(empid, empname, salary)

backup_tbl(empid, empname, salary, operation)

Write a trigger function which will be invoked before the insert, delete or update operation. It does the following:

i. Before delete operation, it inserts the old data into backup_tbl.

ii. Before insert operation, it inserts the new data into backup_tbl.

30. **a.** Consider the following database consisting of the following tables:

Employee (ssn, first name, last name, gender, designation, doj, address)

Employee-salary (ssn, basic pay, DA, TA, pay)

Department (did, dname, mgrssn)

Employee-department (ssn, deptid)

(Create the tables with necessary primary and foreign key)

(Enter at least five records for each relation)

i. For each employee, retrieve the employee first name and last name

ii. Display the distinct department count.

iii. Retrieve all the information about employees working in Research department including the department information.

iv. Retrieve the doj, address of employees who work for Research department using joins.

v. Update the basic pay of each employee with 5%

1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));

2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);

3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));

4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));

1. **select firstname,lastnme from employee;**
2. **select count(distinct dname) from department;**
3. **select * from employee, department where ssn = (select mgrssn from department where dname = 'res');**
4. **select doj, address from employee inner join department on employee.ssn=department.did;**
5. **SQL> UPDATE Employee_salary SET basicpay = (basicpay*1.05);**

b. Write a PL/SQL program to check whether the given employee number is odd or even.

```
SQL> set serveroutput on;
SQL> declare
2  n number := &n;
3  a number;
4  begin
```

```
5  a := mod(n,2);
6  if a=0
7  then
8  dbms_output.put_line('even');
9  else
10 dbms_output.put_line('odd');
11 end if;
12 end;
13 /
Enter value for n: 10 old   2:
n number := &n; new   2: n
number := 10;
even

PL/SQL procedure successfully completed.
```

31. **a.** Consider the following database consisting of the following tables:

Employee (ssn, first name, last name, gender, designation, doj, address)
Employee-salary (ssn, basic pay, DA, TA, pay)
Department (did, dname, mgrssn)
Employee-department (ssn, deptid)

(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. For each employee, retrieve the employee ssn, first name and dname
ii. Retrieve the doj, address of employees who work for Research department
iii. Display the employee ssn who earning second lowest basic pay
iv. Display the department having employee count > 5.
v. Update the DA and TA to 3% if the basic pay is >4000.

```
1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));
2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);
3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));
4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));
```

```
1. select employee.ssn , employee.firstname ,department.dname from employee inner join department on employee.ssn=department.did;
2. select doj , address from Employee where ssn in (select ssn from Employee_department where deptid in (select did from Department where dname = 'Research'));
3.SELECT ssn,basic_pay FROM employee_salary WHERE basic_pay in (SELECT MIN(basic_pay) FROM employee_salary WHERE basic_pay <> (select min(basic_pay) From employee_salary));
4.select dname from Department where did in(select deptid from employee_department group by deptid having count(*) >=5);
5.update employee_salary set DA=(DA*1.03) , TA=(TA*1.03) where basic_pay>4000;
```

b. Write a Pl/SQL program to raise the employee salary by 10%, for department id 30 and also maintain the raised details in the raise table. (use existing table / create table with appropriate attributes).

```
create table employeepl (id int primary key , name varchar(10), salary int);
```

```
DECLARE
   no_count number;
   did number;
BEGIN
   did:=&did;
   UPDATE employeepl set salary = (salary*1.1) where id = did;
   if sql%notfound then
      dbms_output.put_line('No records found');
   ELSIF sql%found THEN
      no_count:=sql%rowcount;
      dbms_output.put_line(no_count|| ' records updated');
   end IF;
END;
/
```

32. **a.** Consider the following database consisting of the following tables:

Employee (ssn, first name, last name, gender, designation, doj, address)
Employee-salary (ssn, basic pay, DA, TA, pay)
Department (did, dname, mgrssn)
Employee-department (ssn, deptid)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

v. For each employee, retrieve the employee first name and last name
vi. Display the distinct department count.
vii. Retrieve all the information about employees working in Research department including the department information.
iv. Retrieve the doj, address of employees who work for Research department using joins.
v. Update the basic pay of each employee with 5%

```
1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));
2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);
3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));
4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));
```

```
1. select firstname,lastnme from employee;
2. select count(distinct dname) from department;
3. select * from employee, department where ssn = ( select mgrssn from department where dname = 'res');
4. select doj,  address from employee inner join department on employee.ssn=department.did;
5. SQL> UPDATE Employee_salary SET basicpay = (basicpay*1.05);
```

b. Write a PL/SQL program to update the commission values for all employees with salary less than 2000 by adding Rs.1000 to existing employees. (Use existing table / create table with appropriate attributes).

```
create table employeepl (id int primary key , name varchar(10), salary int);
```

```
DECLARE
   no_count number;
BEGIN
   UPDATE employeepl set salary = salary+1000 where salary < 2000;
   if sql%notfound then
      dbms_output.put_line('No records found');
   ELSIF sql%found THEN
      no_count:=sql%rowcount;
      dbms_output.put_line(no_count|| ' records updated');
   end IF;
END;
/
```

33. **a**. Consider the following database for a banking enterprise.
account(account_number, branch_name, balance)
branch (branch_name, branch_city, assets)
customer (customer_name customer_street, customer_city)
loan (loan_number, branch_name, amount)
depositor((customer_name, account_number)
borrower(customer_name, loan_number)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

iv. List all branch names and their assests
v. List all accounts of Brooklyn branch
vi. Change the size of the branch_city to varchar(20).
iv. For all customers who have a loan from the bank, find their names and loan numbers with the attribute loan_number replaced by loan_id.
vi. Select the names of customers who have a loan at the bank, and whose names are neither Smith nor Jones

```
CREATE TABLE BRANCH (
branch_name varchar(20),
branch_city varchar(20),
assets real,
constraint bpk PRIMARY KEY(branch_name));

CREATE TABLE CUSTOMER(
customer_name varchar(20),
customer_street varchar(20),
customer_city varchar(20),
constraint cpk PRIMARY KEY(customer_name));
```

```
CREATE TABLE ACCOUNT (
accno int,
branch_name varchar(20),
balance real,
constraint apk PRIMARY KEY(accno),
constraint afk FOREIGN KEY(branch_name) references BRANCH(branch_name));

CREATE TABLE DEPOSITOR (
customer_name varchar(20),
Accno int,
constraint dpk PRIMARY KEY(customer_name, accno),
constraint dfk1 FOREIGN KEY(customer_name) references CUSTOMER(customer_name),
constraint dfk2 FOREIGN KEY(accno) references ACCOUNT(accno) On Delete Cascade);

CREATE TABLE LOAN (
loan_number int,
branch_name varchar(20),
amount real,
constraint lpk PRIMARY KEY(loan_number),
constraint lfk FOREIGN KEY(branch_name) references BRANCH(branch_name));

CREATE TABLE BORROWER (
customer_name varchar(20),
Loan_number int,
constraint bpk PRIMARY KEY(customer_name, Loan_number), constraint bfk1 FOREIGN
KEY(customer_name) references CUSTOMER(customer_name),
constraint bfk2 FOREIGN KEY(Loan_number) references LOAN(Loan_number));
```

```
1.select branch_name , asserts from branch;
2.select account_number from account where branch_name='brooklyn';
3.alter table branch add branch_city varchar(20);
4.alter table loan rename column loan_number to loan_id;
5.select customer_name from customer where name not in ('smith','jones');
```

b. Write a PL/SQL program to find the given year is leap year or not.

```
SQL> set serveroutput on ;
SQL> declare
2   fact number := 1;
3   n number := &n;
4   begin
5   while n>0
6   loop
7   fact := n* fact;
```

```
8  n := n-1;
9  end loop;
10 dbms_output.put_line(fact);
11 end;
12 /
Enter value for n: 5 old   3:
n number := &n; new   3: n
number := 5;
120

PL/SQL procedure successfully completed.
```

34. **a**. Consider the following database of student enrollment in courses and books adopted for that course.

STUDENT(regno:string , name:string , major:string , bdate:date)
COURSE (courseno:int , cname:string , dept:string)
ENROLL(regno:string , courseno:int , sem:int , marks:int)
BOOK_ADOPTION(courseno:int , sem:int , book_isbn:int)
TEXT(book_isbn:int , book_title:string , publisher:string , author:string)

i. Create the above tables by properly specifying the primary keys and foreign keys.
ii. Enter atleast five tuples for each relation.
iii. Demonstrate how you add a new text book to the database and make this book to be adopted by some department.
iv. Produce a list of text books (includes courseno , book_isbn , book_title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.
v. List any department that has all its books published by a specific publisher.

i.

```
1. create table student(regno char(10) primary key, name varchar2(30), major varchar2(10), bdate date);
2. create table course(courseno int primary key, cname varchar2(25), dept varchar2(10));
3. create table enroll(regno char(10) references student(regno), courseno int references course(courseno), sem int primary key, marks int);
4. create table book_adoption(courseno int references course(courseno), sem int references enroll(sem), book_isbn int primary key);
5. create table text(book_isbn int references book_adoption(book_isbn), book_title varchar2(25), publisher varchar2(25), author varchar2(25));
```

ii.

iii.
```
insert into table text values( 1, ‘Operating System’, ‘Pearson Education’, ‘Harvey M. Deitel’);
insert into table book_adoption values(1, 4, 1);
```

iv.
```
select book_adoption.courseno, book_adoption.isbn, text.title from book_adoption  inner join text on orderby asec book_adoption.courseno in (select courseno from course where dept = ‘CS’);
```

v.
```
select dept from department where courseno in (select courseno from book_adoption where isbn in (select isbn from text where publisher = ‘’));
```

b. Write a pl/sql code block to calculate the area of a circle. Store the radius AND the corresponding area IN an empty TABLE named area, consisting of two columns radius AND area

```
SQL> set serveroutput on;
SQL> declare
2    area number;
3    perimeter number;
4    radius number := &radius;
5    pi constant number := 3.14;
6    begin
7    area := pi*radius*radius;
8    perimeter := 2*pi*radius;
9    dbms_output.put_line('Area= '||area);
10   dbms_output.put_line('Perimeter '|| perimeter);
11   end;
12   /
Enter value for radius: 3 old   4: radius
number := &radius; new   4: radius
number := 3;
Area= 28.26
Perimeter 18.84

PL/SQL procedure successfully completed.
```

35. **a.** Consider the following database consisting of the following tables:
Employee (ssn, first name, last name, gender, designation, doj, address)
Employee-salary (ssn, basic pay, DA, TA, pay)
Department (did, dname, mgrssn)
Employee-department (ssn, deptid)
(Create the tables with necessary primary and foreign key)
(Enter at least five records for each relation)

i. For each employee, retrieve the employee ssn, first name and dname
ii. Retrieve the doj, address of employees who work for Research department
iii. Display the employee ssn who earning second lowest basic pay
iv. Display the department having employee count > 5.
v. Update the DA and TA to 3% if the basic pay is >4000.

```
1. create table employee(ssn int primary key, first_name varchar2(25), last_name varchar2(25), gender char(1), desingnation varchar2(50), doj varchar(10), address varchar2(50), check(gender in('M','F')));
2. create table emp_sal(ssn int references employee(ssn),basic_pay int, DA int, TA int, pay int);
3. create table department(did int primary key, dname varchar2(3), mgrssn int references employee(ssn));
4. create table emp_dep(ssn int references employee(ssn), deptid references department(did));
```

```
1. select employee.ssn , employee.firstname ,department.dname from employee inner join department on employee.ssn=department.did;
2. select doj , address from Employee where ssn in (select ssn from Employee_department where deptid in (select did from Department where dname = 'Research'));
```

3.SELECT ssn,basic_pay FROM employee_salary WHERE basic_pay in (SELECT MIN(basic_pay) FROM employee_salary WHERE basic_pay <> (select min(basic_pay) From employee_salary));

4.select dname from Department where did in(select deptid from employee_department group by deptid having count(*) >=5);

5.update employee_salary set DA=(DA*1.03) , TA=(TA*1.03) where basic_pay>4000;

b. Consider the EMPLOYEE (ENO, SALARY, ENAME) TABLE. Write a procedure raise_sal which increases the salary of an employee. It accepts an employee number AND salary increase amount. It uses the employee number to find the current salary FROM the EMPLOYEE TABLE AND UPDATE

```
create table employeepl (id int primary key , name varchar(10), salary int);
```

```
DECLARE
   Employee_id number;
   Amount number;
   no_count number;
BEGIN
   Employee_id:=&Employee_id;
   Amount:=&Amount;
   UPDATE employeepl set salary = (salary+Amount) where id = Employee_id;
   if sql%notfound then
      dbms_output.put_line('No records found');
   ELSIF sql%found THEN
      no_count:=sql%rowcount;
      dbms_output.put_line(no_count|| ' records updated successfully');
   end IF;
END;
/
```