Problem statement 1.
Consider following Bank database schema and solve given queries:
Account(Acc_no, branch_name,balance)
branch(branch_name,branch_city, assets)
customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)

```
create table if not exists account (
            acc_no int primary key,
            branch_name varchar(20),
            balance int,
         foreign key(branch_name) References branch(branch_name)
            );

create table branch(
              branch_name varchar(20) primary key,
               branch_city varchar(20),
               assest int);

create table customer(
  customer_name varchar(20) primary key,
 cust_street varchar(20) ,
   cust_city varchar(20));


create table  depositor(
              cust_name  varchar(20),
              acc_no int ,
              foreign key (acc_no) References account(acc_no),
              foreign key (cust_name) References customer(customer_name));

create table loan (
        loan_no int primary key,
        branch_name varchar(20),
        amount int ,
        foreign key (branch_name) references branch(branch_name));

create table borrower(
```

```
    cust_name varchar(20),
    loan_no int,
    foreign key (cust_name) References customer(customer_name),
    foreign key(loan_no) References loan(loan_no));
```

2.

```
insert into account values(101,"akurdi",20000)
insert into account values(102,"nigdi",5000)


insert into branch values("akurdi" ,"pune" ,100000)
insert into branch values("nigdi" ,"pune" ,300000)

insert into customer values("jay","123abc","Hiwara")
insert into customer values("Avdhuth","123abc","akola")



insert into depositor values("jay",101)
insert into depositor values("Avdhuth",102)


insert into loan values(1001,"akurdi",100)

insert into borrower values ("Avdhuth",1001)
```

Q.2. Create synonym for customer table as cust.

```
create SYNONYM cust FOR customer


CREATE VIEW cust AS SELECT * FROM customer;
SELECT * FROM customer AS cust;
```

Q.3 Add customer phone number in Customer table.

```
alter table customer
```

add cust_mob int

Q.4 Delete phone number attribute from Customer table.

```
alter table customer
drop column cust_mob
```

Q.5. Find the names of all branches in loan relation.

```
select branch_name from loan
```

Q.6. Find all customers who have a loan from bank. Find their names,loan_no and loan amount.

```
select c.customer_name,l.loan_no ,l.amount from customer as c
inner join borrower as b
on c.customer_name =b.cust_name
inner join loan as l
on b.loan_no = l.loan_no
```

Q.7. List all customers in alphabetical order who have loan from Akurdi branch.

```
select b.cust_name from borrower as b
inner join loan as l
on b.loan_no=l.loan_no
where l.branch_name ="akurdi"
order by b.cust_name
```

Q.8. Find all customers who have an account or loan or both at bank

```
SELECT DISTINCT c.customer_name
FROM customer c
LEFT JOIN depositor d ON c.customer_name = d.cust_name
LEFT JOIN borrower b ON c.customer_name = b.cust_name
WHERE d.cust_name IS NOT NULL OR b.cust_name IS NOT NULL;
```

Q.9. Find average account balance at Akurdi branch.

```
select avg(balance) from account where branch_name="akurdi"
```

```
select avg(balance) from account
group by branch_name
having by
```

Q.10. Find no. of depositors at each branch.

```
SELECT a.branch_name, COUNT(d.cust_name) AS num_depositors
FROM Account a
LEFT JOIN Depositor d ON a.Acc_no = d.acc_no
GROUP BY a.branch_name;
```

11 Delete all tuples at every branch located in Nigdi:

```
DELETE FROM Account WHERE branch_name = 'Nigdi';
DELETE FROM Loan WHERE branch_name = 'Nigdi';
```

Problem statement 2.
a) Consider following database schema and solve given queries
cust_mstr(cust_no,fname,lname)
add_dets(code_no,add1,add2,state,city,pincode)

1. Create above Tables with suitable data
create table cust_mstr (cust_no int ,fname varchar(20), l_name varchar(20));
create table add_dets(code_no int,add1 varchar(20),add2 varchar(30),state varchar(40),city varchar(30),pincode int);

2. Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'
insert into cust_mstr values(101,"xyz","pqr")

```
select * from cust_mstr where fname='xyz' and l_name='pqr';
```

3. Create View on add_dets table by selecting any two columns and perform insert update delete
operations

```
insert into add_dets values(1,'xyz','pqr','dd','ff',11);
create view my_view as
select code_no, add1,add2 from add_dets;

select * from my_view

#insert

insert into my_view values(11,"xyz","pqr")

#update
update my_view
set add1="ttt" where code_no=11;

SET SQL_SAFE_UPDATES = 0;
UPDATE my_view SET add1 = "ttt" WHERE code_no = 11;

#delete
```

delete from my_view where add1="ttt"

b) CreatefollowingTables
emp_mstr(e_mpno,f_name,l_name,m_name,dept,desg,branch_no)
branch_mstr(name,b_no)
List the employee details along with branch names to which they belong

```
CREATE TABLE emp_mstr (
    e_mpno INT PRIMARY KEY,
    f_name VARCHAR(50),
    l_name VARCHAR(50),
    m_name VARCHAR(50),
    dept VARCHAR(50),
    desg VARCHAR(50),
    branch_no INT,
    FOREIGN KEY (branch_no) REFERENCES branch_mstr(b_no)
);

CREATE TABLE branch_mstr (
    b_no INT PRIMARY KEY,
    name VARCHAR(50)
);
-- Inserting data into branch_mstr
INSERT INTO branch_mstr (b_no, name) VALUES
(101, 'Branch A'),
(102, 'Branch B');

-- Inserting data into emp_mstr
INSERT INTO emp_mstr (e_mpno, f_name, l_name, m_name, dept, desg, branch_no) VALUES
(201, 'John', 'Doe', NULL, 'IT', 'Manager', 101),
(202, 'Jane', 'Smith', NULL, 'Finance', 'Analyst', 102);


SELECT e.e_mpno, e.f_name, e.l_name, e.m_name, e.dept, e.desg, b.name AS branch_name
FROM emp_mstr e
JOIN branch_mstr b ON e.branch_no = b.b_no;
```

Problem statement 3.
Consider following Bank database schema and solve given queries:
Account(Acc_no, branch_name,balance)
branch(branch_name,branch_city, assets)
customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no) Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)

Q.1 Create above tables with appropriate constraints like primary key, foreign key constraints, not null etc. with suitable data
Q.2. Modify “assets” attribute of branch table to “Property”
Q.3. Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000.

Q.4. Find all customers who have both an account and loan at the bank. Q.5. Find all customers who have an account but no loan at the bank. Q.6. Find the average account balance at each branch
Q.7. Find the branches where average account balance > 12000. Q.8. Find number of tuples in customer relation

Q.9. Calculate total loan amount given by bank.
Q.10. Delete all loans with loan amount between 1300 and 1500.
Q.11. Create sequence roll_seq and use in student table for roll_no column.

Q.1 Create above tables with appropriate constraints like primary key, foreign key

```
create table if not exists account (
            acc_no int primary key,
            branch_name varchar(20),
            balance int,
          foreign key(branch_name) References branch(branch_name)
            );

create table branch(
              branch_name varchar(20) primary key,
               branch_city varchar(20),
               assest int);
```

```
create table customer(
  customer_name varchar(20) primary key,
 cust_street varchar(20) ,
    cust_city varchar(20));


create table  depositor(
                cust_name  varchar(20),
                acc_no int ,
                foreign key (acc_no) References account(acc_no),
                foreign key (cust_name) References customer(customer_name));

create table loan (
          loan_no int primary key,
          branch_name varchar(20),
          amount int ,
          foreign key (branch_name) references branch(branch_name));

create table borrower(
        cust_name varchar(20),
        loan_no int,
        foreign key (cust_name) References customer(customer_name),
        foreign key(loan_no) References loan(loan_no));
```

2.

```
insert into account values(101,"akurdi",20000)
insert into account values(102,"nigdi",5000)


insert into branch values("akurdi" ,"pune" ,100000)
insert into branch values("nigdi" ,"pune" ,300000)

insert into customer values("jay","123abc","Hiwara")
insert into customer values("Avdhuth","123abc","akola")



insert into depositor values("jay",101)
insert into depositor values("Avdhuth",102)
```

```
insert into loan values(1001,"akurdi",100)

insert into borrower values ("Avdhuth",1001)
```

Q.2. Modify “assets” attribute of branch table to “Property”

```
ALTER TABLE branch
CHANGE COLUMN asset Property DECIMAL(10, 2);


select  assest from branch;
alter table branch
rename column assest to property
```

Q.3. Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000

```
select * from loan
select loan_no from loan where branch_name ="akurdi" and amount>12000
```

Q.4. Find all customers who have both account and loan at bank. #problem

```
SELECT DISTINCT c.customer_name
FROM customer c
JOIN Depositor d ON c.customer_name = d.cust_name
JOIN Borrower b ON c.customer_name = b.cust_name;
```

Q.5. Find all customer who have account but no loan at the bank.

```
SELECT DISTINCT c.customer_name
FROM customer c
LEFT JOIN Borrower b ON c.customer_name = b.cust_name
```

```
WHERE b.loan_no IS NULL;
```

Q.6. Find the average account balance at each branch

```
select avg(balance) from account
group by branch_name
```

Q.7. Find the branches where average account balance > 12000.

```
select branch_name, avg(balance) from account
group by branch_name

having avg(balance)>12000
```

Q.8. Find number of tuples in customer relation.

```
select count(customer_name) from customer
```

Q.9. Calculate total loan amount given by bank.

```
select sum(amount) from loan
```

Q.10. Delete all loans with loan amount between 1300 and 1500. # problem

```
SET SQL_SAFE_UPDATES = 0;
delete from loan
where amount between 1300 and 1500
```

.11. Create sequence roll_seq and use in student table for roll_no column.

```
create sequnces myseq
start with 1
increment by 1
minvalue 1
maxvalue 10
cycle
cache

CREATE SEQUENCE myseq
START WITH 1
INCREMENT BY 1
MINVALUE 1
```

```
MAXVALUE 10
CYCLE
CACHE 20;


CREATE TABLE your_table (
    id INT AUTO_INCREMENT PRIMARY KEY,
    -- Other columns...
);
```

Problem statement 4.
a) Create following Tables with suitable data and solve following query
cust_mstr(custno,fname,lname)
acc_fd_cust_dets(codeno,acc_fd_no)
fd_dets(fd_sr_no,amt)
1.1).List the customer holding fixed deposit of amount more than 5000
1.2).Create view on cust_mstr and acc_fd_cust_dets tables by selecting any one column from each table perform insert update delete operations

2.1)Create following Tables with suitable data and solve following query
emp_mstr(emp_no,f_name,l_name,m_name,dept)
cntc_dets(code_no,cntc_type,cntc_data)
List the employee details along with contact details using left outer join & right join

a) Create following Tables with suitable data and solve following query
cust_mstr(custno,fname,lname)
acc_fd_cust_dets(codeno,acc_fd_no)
fd_dets(fd_sr_no,amt)
1.1).List the customer holding fixed deposit of amount more than 5000
1.2).Create view on cust_mstr and acc_fd_cust_dets tables by selecting any one column from each table perform insert update delete operations

```
CREATE TABLE cust_mstrr (
    custno INT PRIMARY KEY,
    fname VARCHAR(50),
    lname VARCHAR(50)
);

CREATE TABLE acc_fd_cust_dets (
    codeno INT PRIMARY KEY,
    acc_fd_no INT,
    FOREIGN KEY (acc_fd_no) REFERENCES fd_dets(fd_sr_no)
);

CREATE TABLE fd_dets (
    fd_sr_no INT PRIMARY KEY,
```

```
    amt DECIMAL(10, 2)
);

-- Inserting sample data into cust_mstr
INSERT INTO cust_mstrr (custno, fname, lname) VALUES
(1, 'John', 'Doe'),
(2, 'Jane', 'Smith');

-- Inserting sample data into fd_dets
INSERT INTO fd_dets (fd_sr_no, amt) VALUES
(101, 5000.00),
(102, 7000.00),
(103, 3000.00);

-- Inserting sample data into acc_fd_cust_dets
INSERT INTO acc_fd_cust_dets (codeno, acc_fd_no) VALUES
(201, 101),
(202, 102),
(203, 103);
```

1.List the customer holding fixed deposit of amount more than 5000;# problem

```
SELECT cm.fname, cm.lname
FROM cust_mstrr cm
JOIN acc_fd_cust_dets afcd ON cm.custno = afcd.codeno
JOIN fd_dets fd ON afcd.acc_fd_no = fd.fd_sr_no
WHERE fd.amt > 5000;
```

2) Create view on cust_mstr and acc_fd_cust_dets tables by selecting any one column from each table perform insert update delete operations

```
create view myviews as
select codeno from acc_fd_cust_dets

create view myviewss as
select
```

2.1)Create following Tables with suitable data and solve following query
emp_mstr(emp_no,f_name,l_name,m_name,dept)
cntc_dets(code_no,cntc_type,cntc_data)
List the employee details along with contact details using left outer join & right join

```
CREATE TABLE emp_mstrrr (
    emp_no INT PRIMARY KEY,
    f_name VARCHAR(50),
    l_name VARCHAR(50),
    m_name VARCHAR(50),
    dept VARCHAR(50)
);

CREATE TABLE cntc_dets (
    code_no INT PRIMARY KEY,
    cntc_type VARCHAR(50),
    cntc_data VARCHAR(100)
);

-- Inserting sample data into emp_mstr
INSERT INTO emp_mstrrr (emp_no, f_name, l_name, m_name, dept) VALUES
(1, 'John', 'Doe', 'A', 'HR'),
(2, 'Jane', 'Smith', 'B', 'Finance');

-- Inserting sample data into cntc_dets
INSERT INTO cntc_dets (code_no, cntc_type, cntc_data) VALUES
(101, 'Email', 'john@example.com'),
(102, 'Phone', '123-456-7890'),
(103, 'Email', 'jane@example.com');
```

Left Outer Join: This will list all employee details along with any available contact details.

```
SELECT e.emp_no, e.f_name, e.l_name, e.m_name, e.dept, c.cntc_type, c.cntc_data
FROM emp_mstrrr e
LEFT OUTER JOIN cntc_dets c ON e.emp_no = c.code_no;
```

```
SELECT e.emp_no, e.f_name, e.l_name, e.m_name, e.dept, c.cntc_type, c.cntc_data
```

```
FROM emp_mstrrr e
RIGHT OUTER JOIN cntc_dets c ON e.emp_no = c.code_no;
```

Problem statement 5.
a) Consider following database schema and solve given queries
cust_mstr(cust_no,fname,lname)
add_dets(code_no,add1,add2,state,city,pincode)
1. Create above Tables with suitable data
2. Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'
3. Create View on add_dets table by selecting any two columns and perform insert update
delete operations b)Create following Tables
cust_mstr(cust_no,fname,lname)
add_dets(code_no,pincode)
List the customer who do not have bank branches in their vicinity.

1. Create above Tables with suitable data

done

2. Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'

```
select a.add1,a.add2 from add_dets as a
full outer join cust_dets as c
on c.cust_no=a.code_no where lname='pqr'

update cust_mstr
set fname='xyz' where cust_no=101

update add_dets
set add1="amravati" where add2="pqr"


SELECT a.add1, a.add2
FROM add_dets AS a
inner join cust_mstr AS c
ON c.cust_no = a.code_no
WHERE c.fname = 'xyz' and l_name='pqr';

select * from add_dets ;
select * from cust_mstr ;
delete from add_dets
```

where code_no=1

insert into add_dets values(101,"hiwara","korde",'Maharstra','akola',44106)

3. Create View on add_dets table by selecting any two columns and perform insert update
delete operations
create view myview1 as
select add1,add2 from add_dets

#add

#insert

#update

4. b)Create following Tables

cust_mstr(cust_no,fname,lname)
add_dets(code_no,pincode)
List the customer who do not have bank branches in their vicinity.

```
CREATE TABLE cust_mstrrrr (
   cust_no INT PRIMARY KEY,
   fname VARCHAR(50),
   lname VARCHAR(50)
);

CREATE TABLE add_detssss (
   code_no INT PRIMARY KEY,
   pincode VARCHAR(10)
);

-- Inserting sample data
INSERT INTO cust_mstrrrr (cust_no, fname, lname) VALUES
(1, 'John', 'Doe'),
(2, 'Jane', 'Smith'),
(3, 'Alice', 'Johnson');

INSERT INTO add_detssss (code_no, pincode) VALUES
(101, '90001'),
(102, '10001'),
(103, '77001');
```

```
SELECT c.fname, c.lname
FROM cust_mstrrrr c
LEFT JOIN add_detssss a ON c.cust_no = a.code_no
WHERE a.pincode IS NULL;
```

Problem statement 6.
Q 1.Consider table Stud(Roll, Att,Status)
Write a PL/SQL block for following requirement and handle the exceptions.
Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in Stud table. If attendance is less than 75% then display the message “Term not granted” and set the status in stud table as “D”. Otherwise display message “Term granted” and set the status in stud table as “ND”.

```
-- Create the Stud table
CREATE TABLE Stud (
  Roll NUMBER,
  Att NUMBER,
  Status VARCHAR2(2)
);

-- Insert sample data into the Stud table
INSERT INTO Stud (Roll, Att, Status)
VALUES (1, 80, 'ND');

INSERT INTO Stud (Roll, Att, Status)
VALUES (2, 70, 'ND');

INSERT INTO Stud (Roll, Att, Status)
VALUES (3, 60, 'ND');

INSERT INTO Stud (Roll, Att, Status)
VALUES (4, 85, 'ND');

INSERT INTO Stud (Roll, Att, Status)
VALUES (5, 50, 'ND');

COMMIT; -- Commit the changes
```

```
-- Declare the procedure
CREATE OR REPLACE PROCEDURE check_attendance(roll_no IN NUMBER)
IS
```

```
  v_attendance NUMBER;
BEGIN
  -- Retrieve attendance for the given roll number
  SELECT Att INTO v_attendance FROM Stud WHERE Roll = roll_no;

  -- Check if attendance is less than 75%
  IF v_attendance < 75 THEN
    -- If attendance is less than 75%, display message and update status as 'D'
    DBMS_OUTPUT.PUT_LINE('Term not granted');
    UPDATE Stud SET Status = 'D' WHERE Roll = roll_no;
  ELSE
    -- If attendance is 75% or more, display message and update status as 'ND'
    DBMS_OUTPUT.PUT_LINE('Term granted');
    UPDATE Stud SET Status = 'ND' WHERE Roll = roll_no;
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    -- Handle case when no data is found for the given roll number
    DBMS_OUTPUT.PUT_LINE('Roll number does not exist');
  WHEN OTHERS THEN
    -- Handle any other exceptions
    DBMS_OUTPUT.PUT_LINE('An error occurred');
END;
/

-- Enable server output
SET SERVEROUTPUT ON;

-- Generate a random roll number between 1 and 5
DECLARE
  v_roll NUMBER;
BEGIN
  v_roll := TRUNC(DBMS_RANDOM.VALUE(1, 5));
  DBMS_OUTPUT.PUT_LINE('Randomly generated roll number: ' || v_roll);
  -- Execute the procedure with the random roll number
  check_attendance(v_roll);
END;
/
```

2.The bank manager has decided to activate all those accounts#learn
which were previously marked as inactive for performing no
transaction in last 365 days. Write a PL/SQ block (using implicit cursor) to update the status of account,
display an approximate message based on the no. of rows affected by the update.
(Use of %FOUND, %NOTFOUND, %ROWCOUNT)

```
CREATE TABLE accounts (
    account_id NUMBER PRIMARY KEY,
    account_number VARCHAR2(20) UNIQUE,
    last_transaction_date DATE,
    status VARCHAR2(10)
);

INSERT INTO accounts  VALUES (1, 'ACC123456', TO_DATE('2023-04-19', 'YYYY-MM-DD'), 'Active');
INSERT INTO accounts  VALUES (2, 'ACC789012', TO_DATE('2022-12-15', 'YYYY-MM-DD'), 'Inactive');
INSERT INTO accounts  VALUES (3, 'ACC345678', TO_DATE('2023-01-20', 'YYYY-MM-DD'), 'Inactive');
INSERT INTO accounts VALUES (4, 'ACC901234', TO_DATE('2023-03-10', 'YYYY-MM-DD'), 'Active');


declare
 countRow number;
begin
    update accounts
    set status='active'
    where last_transaction_date < SYSDATE-365;
    countRow:=sql%rowcount;
  dbms_output.put_line(countRow);
exception
    when no_data_found then
    dbms_output.put_line('There is no data availabe in acount table');
end;
/
```

Problem statement 7.

Q 1. Write an SQL code block these raise a user defined exception where business rule is voilated. BR for client_master table specifies when the value of bal_due field is less than 0 handle the exception.

```
create table client_master ( id number, bal_due number);
```

```
DECLARE
   v_bal_due NUMBER;
BEGIN
   -- Get the balance due for a specific client (replace 'client_id' with the actual client ID)
   SELECT bal_due INTO v_bal_due
   FROM client_master
   WHERE client_id = 'your_client_id';

   -- Check if the balance due is less than 0
   IF v_bal_due < 0 THEN
      -- Raise a user-defined exception
      RAISE_APPLICATION_ERROR(-20001, 'Balance due cannot be negative.');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      -- Handle other exceptions here if needed
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

or

```
declare
   idv number ;
   bal_duev number;
   exe exception;
begin
   idv:=1;
```

```
    bal_duev:=&bal;
 if bal_duev<0 then
     raise exe;
else
    insert into client_master values(idv,bal_duev);
end if;
Exception
    when exe then
    dbms_output.put_line('Balace is less than 0 please enter the Greter than >0 balance ');
end;
/
```

Q 2. Organization has decided to increase the salary of employees by 10% of existing salary, who are having salary less than average salary of organization, Whenever such salary updates takes place, a record for the same is maintained in the increment_salary table.
EMP (E_no , Salary)
increment_salary(E_no , Salary)

```
create table EMP(E_no number , Salary number);
create table increment_salary(E_no number, salaray number);
insert into EMP values(1,30);
insert into EMP values(2,20);
insert into Emp values(3,50);


declare
    salavg number;
    E_idv number;
    Salaryv number;
    newsal number;
  cursor mycur is select E_no,Salary from Emp;
begin
    select avg(Salary) into salavg from Emp;
    open mycur;
    loop
    fetch mycur into E_idv, Salaryv ;


        if salavg>Salaryv then
         newsal:=Salaryv*1.1;

   insert into increment_salary values(E_idv ,newsal);
```

```
        end if;

    exit when mycur%notfound;
    end loop;
    close mycur;
end;
/
```

Problem statement 8.
Q 1.Borrower(Roll_no, Name, DateofIssue, NameofBook, Status)
Fine(Roll_no,Date,Amt)
1. Accept roll_no& name of book from user.
2. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.
3. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R
4. If condition of fine is true, then details will be stored into fine table.
5. Also handles the exception by named exception handler or user define exception handler.

```
CREATE TABLE Borrower (
  Roll_no NUMBER,
  Name VARCHAR2(100),
  DateofIssue DATE,
  NameofBook VARCHAR2(100),
  Status VARCHAR2(1)
);
CREATE TABLE Fine (
  Roll_no NUMBER,
  Date_of_Fine DATE,
  Amt NUMBER
);


-- Insert sample data into the Borrower table
INSERT INTO Borrower (Roll_no, Name, DateofIssue, NameofBook, Status)
VALUES (1, 'John', TO_DATE('2024-04-01', 'YYYY-MM-DD'), 'Book1', 'I');

INSERT INTO Borrower (Roll_no, Name, DateofIssue, NameofBook, Status)
VALUES (2, 'Alice', TO_DATE('2024-03-15', 'YYYY-MM-DD'), 'Book2', 'I');



DECLARE
  v_roll_no Borrower.Roll_no%TYPE;
  v_name_of_book Borrower.NameofBook%TYPE;
  v_date_of_issue Borrower.DateofIssue%TYPE;
  v_days NUMBER;
```

```
  v_fine_amt NUMBER := 0;



BEGIN
  -- Accept roll number and name of the book from the user
  v_roll_no := &roll_no;
  v_name_of_book := '&name_of_book';

  -- Retrieve the date of issue and status of the book
  SELECT DateofIssue INTO v_date_of_issue
  FROM Borrower
  WHERE Roll_no = v_roll_no AND NameofBook = v_name_of_book;

  -- Calculate the number of days from the date of issue
  v_days := TRUNC(SYSDATE) - v_date_of_issue;

  -- Calculate the fine amount based on the number of days
  IF v_days > 30 THEN
    v_fine_amt := (v_days - 30) * 50 + 15 * 5; -- Rs. 50 per day after 30 days, Rs. 5 per day for 15
days
  ELSIF v_days > 15 THEN
    v_fine_amt := (v_days - 15) * 5; -- Rs. 5 per day after 15 days
  END IF;

  -- Update the status of the book to 'R' (Returned)
  UPDATE Borrower SET Status = 'R' WHERE Roll_no = v_roll_no AND NameofBook =
v_name_of_book;

  -- Insert details into the Fine table if fine amount is greater than 0
  IF v_fine_amt > 0 THEN
    INSERT INTO Fine (Roll_no, Date_of_Fine, Amt) VALUES (v_roll_no, SYSDATE,
v_fine_amt);
  END IF;

  -- Display message
  DBMS_OUTPUT.PUT_LINE('Fine calculated and book status updated successfully.');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No record found for the provided roll number and name of the
book.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
```

```
END;
/
```

Problem statement 9.
Q 1. Write PL/SQL block using explicit cursor for following requirements:
College has decided to mark all those students detained (D) who are having attendance less than
75%.
Whenever such an update takes place, a record for the same is maintained in the D_Stud table.
create table stud21(roll number(4), att number(4), status varchar(1));
create table d_stud(roll number(4), att number(4));

```
CREATE TABLE stud21 (
  roll NUMBER(4),
  att NUMBER(4),
  status VARCHAR2(1)
);

CREATE TABLE d_stud (
  roll NUMBER(4),
  att NUMBER(4)
);
```

```
-- Insert sample data into stud21 table
INSERT INTO stud21 (roll, att, status)
VALUES (1, 70, NULL);

INSERT INTO stud21 (roll, att, status)
VALUES (2, 80, NULL);

INSERT INTO stud21 (roll, att, status)
VALUES (3, 60, NULL);
```

```
DECLARE
  CURSOR c_students IS
```

```
    SELECT roll, att
    FROM stud21
    WHERE att < 75;

  v_roll stud21.roll%TYPE;
  v_att stud21.att%TYPE;
BEGIN
  -- Open the cursor
  OPEN c_students;

  -- Fetch and process each row
  LOOP
    FETCH c_students INTO v_roll, v_att;
    EXIT WHEN c_students%NOTFOUND;

    -- Update the status of detained students
    UPDATE stud21
    SET status = 'D'
    WHERE roll = v_roll;

    -- Insert a record into d_stud table for the detained student
    INSERT INTO d_stud (roll, att) VALUES (v_roll, v_att);
  END LOOP;

  -- Close the cursor
  CLOSE c_students;

  -- Display message
  DBMS_OUTPUT.PUT_LINE('Detained students marked successfully.');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

Problem statement 10.

Q 1.Consider table Stud(Roll, Att,Status)
Write a PL/SQL block for following requirement and handle the exceptions.
Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in
Stud table. If attendance is less than 75% then display the message “Term not granted” and set the
status in stud table as “D”. Otherwise display message “Term granted” and set the status in stud
table as “ND”.

```
-- Create table Stud1
CREATE TABLE Stud1 (
  Roll NUMBER,
  Att NUMBER,
  Status VARCHAR2(2)
);

-- Sample data for testing
INSERT INTO Stud1 VALUES (1, 80, NULL);
INSERT INTO Stud1 VALUES (2, 70, NULL);
INSERT INTO Stud1 VALUES (3, 90, NULL);

-- PL/SQL block to handle the requirements
DECLARE
  v_roll_number Stud1.Roll%TYPE;
  v_attendance Stud1.Att%TYPE;
BEGIN
  -- Accept roll number from the user
  v_roll_number := &roll_number;

  -- Retrieve attendance from the Stud1 table for the given roll number
  SELECT Att INTO v_attendance
  FROM Stud1
  WHERE Roll = v_roll_number;

  -- Check if attendance is less than 75%
  IF v_attendance < 75 THEN
```

```
    -- Update status to 'D' (Term not granted)
    UPDATE Stud1 SET Status = 'D' WHERE Roll = v_roll_number;
    -- Display message
    DBMS_OUTPUT.PUT_LINE('Term not granted');
  ELSE
    -- Update status to 'ND' (Term granted)
    UPDATE Stud1 SET Status = 'ND' WHERE Roll = v_roll_number;
    -- Display message
    DBMS_OUTPUT.PUT_LINE('Term granted');
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Roll number not found in the database.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
```

Q 2.Write an update, delete trigger on client master table. The System should keep track of the records
that ARE BEING updated or deleted. The old value of updated or deleted records should be added
in the audit_trade table. (separate implementation using both row and statement triggers)

```
CREATE OR REPLACE TRIGGER update_audit_row
AFTER UPDATE ON clientmstr
FOR EACH ROW
BEGIN
    INSERT INTO audit_trade (action, client_id, old_value)
    VALUES ('UPDATE', :OLD.client_id, :OLD.client_details);
END;
/
```

```
CREATE OR REPLACE TRIGGER delete_audit_row
AFTER DELETE ON clientmstr
FOR EACH ROW
BEGIN
    INSERT INTO audit_trade (action, client_id, old_value)
    VALUES ('DELETE', :OLD.client_id, :OLD.client_details);
```

```
END;
/



CREATE OR REPLACE TRIGGER update_audit_stmt
AFTER UPDATE ON clientmstr
DECLARE
   CURSOR c_client_details IS
      SELECT client_id, client_details
      FROM clientmstr
      WHERE client_id IN (SELECT client_id FROM deleted_clients);
BEGIN
   FOR client_rec IN c_client_details LOOP
      INSERT INTO audit_trade (action, client_id, old_value)
      VALUES ('UPDATE', client_rec.client_id, client_rec.client_details);
   END LOOP;
END;
/



CREATE OR REPLACE TRIGGER delete_audit_stmt
AFTER DELETE ON clientmstr
BEGIN
   INSERT INTO audit_trade (action, client_id, old_value)
   SELECT 'DELETE', client_id, client_details
   FROM deleted_clients;
END;
/



or



INSERT INTO clientmstr (client_id, client_name, address) VALUES (122, 'jay Doe', '3 Main St');
```

1. Row-Level Trigger for Update:

```
CREATE OR REPLACE TRIGGER update_audit_trigger
AFTER UPDATE ON clientmstr
FOR EACH ROW
BEGIN
```

```
    IF :OLD.client_id IS NOT NULL THEN
        INSERT INTO audit_trade (action_type, client_id, old_value)
        VALUES ('UPDATE', :OLD.client_id, :OLD.bal_due);
    END IF;
END;
/
```

2. Row-Level Trigger for Delete:

```
CREATE OR REPLACE TRIGGER delete_audit_trigger
AFTER DELETE ON clientmstr
FOR EACH ROW
BEGIN
    INSERT INTO audit_trade (action_type, client_id, old_value)
    VALUES ('DELETE', :OLD.client_id, :OLD.bal_due);
END;
/
```

3. Statement-Level Trigger for Update:

```
CREATE OR REPLACE TRIGGER update_audit_statement_trigger
AFTER UPDATE ON clientmstr
BEGIN
    FOR rec IN (SELECT * FROM clientmstr WHERE client_id IN (SELECT client_id FROM
DELETED))
    LOOP
        INSERT INTO audit_trade (action_type, client_id, old_value)
        VALUES ('UPDATE', rec.client_id, rec.bal_due);
    END LOOP;
END;
/
```

4. Statement-Level Trigger for Delete:

```
CREATE OR REPLACE TRIGGER delete_audit_statement_trigger
AFTER DELETE ON clientmstr
BEGIN
    FOR rec IN (SELECT * FROM clientmstr WHERE client_id IN (SELECT client_id FROM
DELETED))
    LOOP
        INSERT INTO audit_trade (action_type, client_id, old_value)
        VALUES ('DELETE', rec.client_id, rec.bal_due);
    END LOOP;
END;
/
```

Problem statement 11.
Q 1. Write a stored function in PL/SQL for a given requirement and use the same in PL/SQL block.
Account no. and branch name will be accepted from the user. The same will be searched in table
acct_details. If status of account is active then display appropriate message and also store the
account details in active_acc_details table, otherwise display message on screen “account is
inactive”.

```
CREATE TABLE acct_details (
   account_no NUMBER,
   branch_name VARCHAR2(100),
   status VARCHAR2(20)
);


CREATE TABLE active_acc_details (
   account_no NUMBER,
   branch_name VARCHAR2(100)
);


-- Inserting sample data into acct_details table
INSERT INTO acct_details (account_no, branch_name, status) VALUES (1, 'Branch A', 'active');
INSERT INTO acct_details (account_no, branch_name, status) VALUES (2, 'Branch B',
'inactive');
INSERT INTO acct_details (account_no, branch_name, status) VALUES (3, 'Branch C', 'active');



CREATE OR REPLACE FUNCTION check_account_status(p_account_no IN NUMBER,
p_branch_name IN VARCHAR2)
RETURN VARCHAR2
IS
   v_status acct_details.status%TYPE;
BEGIN
   -- Check if the account exists
   SELECT status INTO v_status
   FROM acct_details
   WHERE account_no = p_account_no AND branch_name = p_branch_name;
```

```
    IF v_status = 'active' THEN
        -- Insert into active_acc_details table
        INSERT INTO active_acc_details (account_no, branch_name)
        VALUES (p_account_no, p_branch_name);

        -- Return message
        RETURN 'Account is active. Account details stored successfully.';
    ELSE
        -- Return message
        RETURN 'Account is inactive.';
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Account not found.';
END;
/



DECLARE
    v_account_no NUMBER := &account_no; -- Accept account number from user
    v_branch_name VARCHAR2(100) := '&branch_name'; -- Accept branch name from user
    v_message VARCHAR2(200);
BEGIN
    v_message := check_account_status(v_account_no, v_branch_name);
    DBMS_OUTPUT.PUT_LINE(v_message);
END;
/
```

Problem statement 12.
Q 1. Write an SQL code block that raises a user defined exception where business rule is violated.
BR for client_master table specifies when the value of bal_due field is less than 0 handle the exception.

Q 2.Write a before trigger for Insert, update event considering following requirement:

Emp(e_no, e_name, salary)
I) Trigger action should be initiated when salary is tried to be inserted is less than Rs. 50,000/-
II) Trigger action should be initiated when salary is tried to be updated for value less than Rs. 50,000/-
Action should be rejection of update or Insert operation by displaying appropriate error message. Also the new values expected to be inserted will be stored in new table.
Tracking(e_no, salary).

```
CREATE TABLE Emp (
    e_no INT,
    e_name VARCHAR(100),
    salary DECIMAL(10, 2)
);

CREATE TABLE Tracking (
    e_no INT,
    salary DECIMAL(10, 2)
);
```

```
CREATE OR REPLACE TRIGGER check_salary
BEFORE INSERT OR UPDATE ON Emp
FOR EACH ROW
DECLARE
   v_salary_limit DECIMAL(10, 2) := 50000.00;
BEGIN
   -- Check if the salary being inserted or updated is less than Rs. 50,000/-
   IF :NEW.salary < v_salary_limit THEN
      -- Store the rejected record in the Tracking table
      INSERT INTO Tracking (e_no, salary) VALUES (:NEW.e_no, :NEW.salary);

      -- Display appropriate error message and reject the operation
      IF INSERTING THEN
         RAISE_APPLICATION_ERROR(-20001, 'Error: Salary must be Rs. 50,000/- or more for
new records.');
      ELSE
         RAISE_APPLICATION_ERROR(-20002, 'Error: Salary must be Rs. 50,000/- or more for
updates.');
      END IF;
   END IF;
END;
/




-- Inserting a record with salary less than 50,000
INSERT INTO Emp (e_no, e_name, salary) VALUES (1, 'John Doe', 45000.00);


show error

-- Inserting a record with salary greater than or equal to 50,000
INSERT INTO Emp (e_no, e_name, salary) VALUES (2, 'Jane Smith', 60000.00);
execute succefully


-- Updating the salary of a record to less than 50,000
UPDATE Emp SET salary = 48000.00 WHERE e_no = 2;
show error
```

Problem statement 13.
Q 1. . Write a PL/SQL stored Procedure for following requirements and call the procedure in appropriate
PL/SQL block.
Borrower(Rollin, Name, DateofIssue, NameofBook, Status)
Fine(Roll_no,Date,Amt)
Accept roll_no& name of book from user.
1. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount
will be Rs 5per day.
2. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
3. After submitting the book, status will change from I to R.
4. If condition of fine is true, then details will be stored into fine table. ....execute above type ouput also

```
CREATE OR REPLACE PROCEDURE calculate_fine (
   p_roll_no IN NUMBER,
   p_book_name IN VARCHAR2
)
IS
   v_days_late NUMBER;
   v_fine_amt NUMBER;
BEGIN
   -- Calculate the number of days late from the DateofIssue
   SELECT TRUNC(SYSDATE - DateofIssue) INTO v_days_late
   FROM Borrower
   WHERE Rollin = p_roll_no AND NameofBook = p_book_name;

   -- Check the number of days late and calculate fine amount accordingly
   IF v_days_late BETWEEN 15 AND 30 THEN
      v_fine_amt := v_days_late * 5;
   ELSIF v_days_late > 30 THEN
      v_fine_amt := v_days_late * 50;
   ELSE
      v_fine_amt := 0;
   END IF;
```

```
    -- Update the status in Borrower table to 'R' (Returned)
    UPDATE Borrower
    SET Status = 'R'
    WHERE Rollin = p_roll_no AND NameofBook = p_book_name;

    -- If fine is applicable, store details in Fine table
    IF v_fine_amt > 0 THEN
        INSERT INTO Fine (Roll_no, Date, Amt)
        VALUES (p_roll_no, SYSDATE, v_fine_amt);
    END IF;

    -- Display fine amount
    DBMS_OUTPUT.PUT_LINE('Fine amount: Rs ' || v_fine_amt);
END;
/
```

```
DECLARE
    v_roll_no NUMBER := &roll_no; -- Accept roll number from user
    v_book_name VARCHAR2(100) := '&book_name'; -- Accept book name from user
BEGIN
    calculate_fine(v_roll_no, v_book_name);
END;
/
```

Problem statement 14.
Q 1. Write a Stored Procedure namely proc_Grade for the categorization of students. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class.
Write a PL/SQL block for using procedures created with the above requirement.
Stud_Marks(name, total_marks)
Result(Roll,Name, Class)

```
CREATE TABLE Stud_Marks (
    name VARCHAR2(100),
    total_marks NUMBER
);


CREATE TABLE Result (
    Roll NUMBER,
    Name VARCHAR2(100),
    Class VARCHAR2(100)
);
```

```
CREATE OR REPLACE PROCEDURE proc_Grade (
    p_name IN VARCHAR2,
    p_total_marks IN NUMBER,
    p_class OUT VARCHAR2
)
IS
BEGIN
    IF p_total_marks <= 1500 AND p_total_marks >= 990 THEN
        p_class := 'Distinction';
    ELSIF p_total_marks <= 989 AND p_total_marks >= 900 THEN
```

```
        p_class := 'First Class';
    ELSIF p_total_marks <= 899 AND p_total_marks >= 825 THEN
        p_class := 'Higher Second Class';
    ELSE
        p_class := 'Not Categorized';
    END IF;
END;
/
```

```
DECLARE
    v_name Stud_Marks.name%TYPE := '&student_name'; -- Accept student name from user
    v_total_marks Stud_Marks.total_marks%TYPE := &total_marks; -- Accept total marks from
user
    v_class Result.Class%TYPE; -- Declare variable to store class
BEGIN
    -- Call the proc_Grade procedure to categorize the student
    proc_Grade(v_name, v_total_marks, v_class);

    -- Insert the result into the Result table
    INSERT INTO Result (Roll, Name, Class)
    VALUES (1, v_name, v_class);

    -- Display the result
    DBMS_OUTPUT.PUT_LINE('Student ' || v_name || ' categorized as ' || v_class);
END;
/
```

Problem statement 15.
Create Database PCCOE
Create following Collections
Teachers(Tname,dno,dname,experience,salary,date_of_joining)
Students(Sname,roll_no,class)
Q1. Find the information about all teachers

```
use PCCOE

db.createCollection("Teachers")
db.createCollection("Students")
```

Q2. Find the information about all teachers of computer department

```
db.Teachers.find()
db.Teachers.find({ dname: "Computer" })
```

Q3. Find the information about all teachers of computer,IT,ande&TC department

```
db.Teachers.find({ dname: "Computer" })
```

Q4. Find the information about all teachers of computer,IT,and E&TC department having salary greater than or equl to 10000/-

```
db.Teachers.find({ dname: { $in: ["Computer", "IT", "E&TC"] }, salary: { $gte: 10000 } })
```

Q5. Find the student information having roll_no = 2 or Sname=xyz

```
db.Students.find({ $or: [ { roll_no: 2 }, { Sname: "xyz" } ] })
```

Q6. Update the experience of teacher-praveen to 10years, if the entry is not available in database
consider the entry as new entry.

```
db.Teachers.update(
   { Tname: "Praveen" },
   { $set: { experience: 10 } },
   { upsert: true }
)
```

Q7. Update the deparment of all the teachers working in IT deprtment to COMP

```
db.Teachers.updateMany(
   { dname: "IT" },
```

```
    { $set: { dname: "COMP" } }
)
```

Q8. Find the teachers name and their experience from teachers collection

```
db.Teachers.find({}, { _id: 0, Tname: 1, experience: 1 })
```

Q9. Using Save() method insert one entry in department collection

```
db.Teachers.save(
    { Tname: "New Teacher", dno: 1, dname: "Computer", experience: 5, salary: 15000,
date_of_joining: new Date() }
)
```

Q10. Using Save() method change the dept of teacher praveen to IT

```
db.Teachers.save(
    { Tname: "Praveen", dno: 1, dname: "IT", experience: 7, salary: 18000, date_of_joining: new
Date() }
)
```

Q11. Delete all the doccuments from teachers collection having IT dept.

```
db.Teachers.deleteMany({ dname: "IT" })
```

Q12. Display with pretty() method, the first 3 doccuments in teachers collection in ascending order

```
db.Teachers.find().limit(3).sort({ _id: 1 }).pretty()
```

Problem statements 16
Consider the relational database
Supplier(Sid,Sname,address)
Parts(Pid, Pname, Color)
Catalog(sid,pid,cost)
Q. Find the name of all parts whose color is green.

```
SELECT Pname
FROM Parts
WHERE Color = 'green';
```

Q. Find names of suppliers who supply some red parts.

```
SELECT DISTINCT Sname
FROM Supplier
JOIN Catalog ON Supplier.Sid = Catalog.Sid
JOIN Parts ON Catalog.Pid = Parts.Pid
WHERE Parts.Color = 'red';
```

Q. Find names of all parts whose cost is more than Rs25.

```
SELECT Pname
FROM Parts
JOIN Catalog ON Parts.Pid = Catalog.Pid
WHERE Catalog.cost > 25;
```

```
-- Create Supplier table
CREATE TABLE Supplier (
    Sid INT PRIMARY KEY,
    Sname VARCHAR(100),
    address VARCHAR(255)
);

-- Create Parts table
CREATE TABLE Parts (
    Pid INT PRIMARY KEY,
    Pname VARCHAR(100),
    Color VARCHAR(50)
);
```

```
-- Create Catalog table
CREATE TABLE Catalog (
    Sid INT,
    Pid INT,
    cost DECIMAL(10, 2),
    PRIMARY KEY (Sid, Pid),
    FOREIGN KEY (Sid) REFERENCES Supplier(Sid),
    FOREIGN KEY (Pid) REFERENCES Parts(Pid)
);



INSERT INTO Supplier (Sid, Sname, address) VALUES (1, 'Supplier1', 'Address1');
INSERT INTO Supplier (Sid, Sname, address) VALUES (2, 'Supplier2', 'Address2');
INSERT INTO Supplier (Sid, Sname, address) VALUES (3, 'Supplier3', 'Address3');

INSERT INTO Parts (Pid, Pname, Color) VALUES (101, 'Part1', 'green');
INSERT INTO Parts (Pid, Pname, Color) VALUES (102, 'Part2', 'red');
INSERT INTO Parts (Pid, Pname, Color) VALUES (103, 'Part3', 'blue');
INSERT INTO Parts (Pid, Pname, Color) VALUES (104, 'Part4', 'green');


INSERT INTO Catalog (Sid, Pid, cost) VALUES (1, 101, 20.00);
INSERT INTO Catalog (Sid, Pid, cost) VALUES (2, 102, 30.00);
INSERT INTO Catalog (Sid, Pid, cost) VALUES (3, 103, 25.00);
INSERT INTO Catalog (Sid, Pid, cost) VALUES (1, 104, 28.00);
INSERT INTO Catalog (Sid, Pid, cost) VALUES (2, 101, 22.00);
```

2.Consider the relational database
Person(pname,street city)
Company(cname,city)
Manages(pname,mname)

3. Find the street and city of all employees who work for “Idea”, live in Pune and earn more than
3000.

```
SELECT p.street, p.city
FROM Person p
JOIN Manages m ON p.pname = m.pname
JOIN Company c ON m.mname = c.cname
WHERE c.cname = 'Idea'
AND p.city = 'Pune'
AND p.earnings > 3000;
```

or

```
SELECT street, city
FROM Person
WHERE pname IN (SELECT pname FROM Manages WHERE mname = 'Idea')
  AND city = 'Pune'
  AND pname IN (SELECT pname FROM Manages WHERE mname = 'Idea')
  AND pname IN (SELECT pname FROM Manages WHERE mname IN (SELECT cname
FROM Company WHERE city = 'Pune'))
  AND pname IN (SELECT pname FROM Manages WHERE pname IN (SELECT pname FROM
Manages WHERE mname = 'Idea') AND city = 'Pune')
  AND pname IN (SELECT pname FROM Manages WHERE pname IN (SELECT pname FROM
Manages WHERE mname = 'Idea') AND city = 'Pune' AND street IN (SELECT street FROM
Person WHERE pname IN (SELECT pname FROM Manages WHERE mname = 'Idea') AND
city = 'Pune' AND pname IN (SELECT pname FROM Manages WHERE mname = 'Idea' AND
pname IN (SELECT pname FROM Person WHERE city = 'Pune'))));
```

4.Consider the relational database
Student(Rollno,name,address)
Subject(sub_code,sub_name)
Marks(Rollno,sub_code, marks)
Q. Find out average marks of each student along with the name of student.

```
SELECT s.name, AVG(m.marks) AS average_marks
FROM Student s
JOIN Marks m ON s.Rollno = m.Rollno
GROUP BY s.Rollno, s.name;
```

Q. Find how many students have failed in the subject “DBMS”

```
SELECT COUNT(*) AS num_failures
FROM Marks
WHERE sub_code = 'DBMS' AND marks < 40;
```

Problem statements 17

Write PI/SQL code block that will accept account number from user , check if the users balance is less than
the minimum balance , only deduct Rs.100/- from the balance .

```
-- Create the accounts table
CREATE TABLE accounts (
    account_number NUMBER PRIMARY KEY,
    balance NUMBER
);

-- Insert sample data
INSERT INTO accounts (account_number, balance) VALUES (1001, 950); -- Account with balance less than the minimum
INSERT INTO accounts (account_number, balance) VALUES (1002, 1500); -- Account with balance greater than the minimum


DECLARE
    v_account_number NUMBER;
    v_balance NUMBER;
    v_minimum_balance NUMBER := 1000; -- Assuming minimum balance is Rs. 1000
BEGIN
    -- Accept account number from the user
    v_account_number := &account_number;

    -- Retrieve the balance for the given account number
    SELECT balance INTO v_balance
    FROM accounts
    WHERE account_number = v_account_number;

    -- Check if the balance is less than the minimum balance
    IF v_balance < v_minimum_balance THEN
        -- Deduct Rs. 100 from the balance
        UPDATE accounts
        SET balance = balance - 100
        WHERE account_number = v_account_number;

        DBMS_OUTPUT.PUT_LINE('Balance is less than the minimum balance. Rs. 100 deducted from the account.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Balance is sufficient. No deduction required.');
```

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

Problem statements 18
Write PI/SQL code block for inverting number 1234 to 4321.

```
DECLARE
   v_number NUMBER := 1234; -- Change this to the desired number
   v_inverted_number NUMBER := 0;
   v_digit NUMBER;
BEGIN
   WHILE v_number > 0 LOOP
      -- Extract the last digit of the number
      v_digit := MOD(v_number, 10);

      -- Append the digit to the inverted number
      v_inverted_number := v_inverted_number * 10 + v_digit;

      -- Remove the last digit from the number
      v_number := TRUNC(v_number / 10); -- Use TRUNC to avoid getting decimal part
   END LOOP;

   -- Output the inverted number
   DBMS_OUTPUT.PUT_LINE('Inverted number: ' || v_inverted_number);
END;
/
```

Problem statements 19
The bank manager has decided to mark all those accounts as inactive (I) on which there are no
transactions performed in the last 365 days. Whenever any such update takes place a record for the
same is maintained in the INACT_MASTER_TABLE comprising of the account number, the
opening date and type of account. Write PL/SQL code block to do the same(cursor for loop)

```
-- Create the account table
CREATE TABLE account (
    account_number NUMBER PRIMARY KEY,
    opening_date DATE,
    account_type VARCHAR2(50),
    last_transaction_date DATE
);

-- Create the INACT_MASTER_TABLE
CREATE TABLE INACT_MASTER_TABLE (
    account_number NUMBER,
    opening_date DATE,
    account_type VARCHAR2(50)
);


-- Insert sample data into the account table
INSERT INTO account (account_number, opening_date, account_type, last_transaction_date)
VALUES (1001, TO_DATE('2023-01-01', 'YYYY-MM-DD'), 'Savings', SYSDATE - 400);
INSERT INTO account (account_number, opening_date, account_type, last_transaction_date)
VALUES (1002, TO_DATE('2022-11-15', 'YYYY-MM-DD'), 'Current', SYSDATE - 100);
INSERT INTO account (account_number, opening_date, account_type, last_transaction_date)
VALUES (1003, TO_DATE('2023-03-20', 'YYYY-MM-DD'), 'Savings', SYSDATE - 300);



DECLARE
    v_account_number NUMBER;
    v_opening_date DATE;
    v_account_type VARCHAR2(50);
```

```
BEGIN
   FOR acc IN (SELECT * FROM account) LOOP
      IF acc.last_transaction_date < SYSDATE - 365 THEN
         -- Mark account as inactive
         UPDATE account
         SET account_type = 'I'
         WHERE account_number = acc.account_number;

         -- Insert record into INACT_MASTER_TABLE
         INSERT INTO INACT_MASTER_TABLE (account_number, opening_date,
account_type)
         VALUES (acc.account_number, acc.opening_date, acc.account_type);

         DBMS_OUTPUT.PUT_LINE('Account ' || acc.account_number || ' marked as inactive.');
      END IF;
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('Inactive accounts marked successfully.');
END;
/
```

Problem statements 20
Write PL/SQL code block that will merge the data available in the newly created table NEW_BRANCHES with the data available in the table BRANCH_MASTER. If the data in the first table already exists in the second table then data should be skipped.(parameterized cursor)

```
-- Structure of the NEW_BRANCHES table
CREATE TABLE NEW_BRANCHES (
    branch_id NUMBER PRIMARY KEY,
    branch_name VARCHAR2(100),
    location VARCHAR2(100)
);

-- Structure of the BRANCH_MASTER table
CREATE TABLE BRANCH_MASTER (
    branch_id NUMBER PRIMARY KEY,
    branch_name VARCHAR2(100),
    location VARCHAR2(100)
);
```

```
-- Insert sample data into the NEW_BRANCHES table
INSERT INTO NEW_BRANCHES (branch_id, branch_name, location) VALUES (1, 'Branch A',
'Location A');
INSERT INTO NEW_BRANCHES (branch_id, branch_name, location) VALUES (2, 'Branch B',
'Location B');
INSERT INTO NEW_BRANCHES (branch_id, branch_name, location) VALUES (3, 'Branch C',
'Location C');
```

```
DECLARE
    v_branch_id NEW_BRANCHES.branch_id%TYPE;
    v_branch_name NEW_BRANCHES.branch_name%TYPE;
    v_location NEW_BRANCHES.location%TYPE;
BEGIN
    FOR new_branch IN (SELECT * FROM NEW_BRANCHES) LOOP
        -- Check if the branch_id already exists in BRANCH_MASTER
        SELECT COUNT(*) INTO v_branch_id
        FROM BRANCH_MASTER
        WHERE branch_id = new_branch.branch_id;
```

```
        -- If the branch_id doesn't exist, insert the record into BRANCH_MASTER
        IF v_branch_id = 0 THEN
            INSERT INTO BRANCH_MASTER (branch_id, branch_name, location)
            VALUES (new_branch.branch_id, new_branch.branch_name, new_branch.location);

            DBMS_OUTPUT.PUT_LINE('Record inserted for branch_id: ' || new_branch.branch_id);
        ELSE
            DBMS_OUTPUT.PUT_LINE('Record skipped for branch_id: ' || new_branch.branch_id ||
' as it already exists.');
        END IF;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Merge completed successfully.');
END;
/

-- Select all records from the BRANCH_MASTER table
SELECT *
FROM BRANCH_MASTER;
```

Problem statements 21

Write PL/SQL code block such that depending upon user supplied account number, the customer to
whom account belongs , the introducer of that account are inserted into ACCOUNT_MASTER_INFO table .If the user enters an account number that is not in the ACCOUNT_MASTER table, then the PL/SQL block must display appropriate error message(Exception Handling)

```
-- Structure of the ACCOUNT_MASTER table
CREATE TABLE ACCOUNT_MASTER (
    account_number NUMBER PRIMARY KEY,
    customer_name VARCHAR2(100),
    introducer_name VARCHAR2(100)
);

-- Structure of the ACCOUNT_MASTER_INFO table
CREATE TABLE ACCOUNT_MASTER_INFO (
    account_number NUMBER PRIMARY KEY,
    customer_name VARCHAR2(100),
    introducer_name VARCHAR2(100)
);


-- Insert sample data into the ACCOUNT_MASTER table
INSERT INTO ACCOUNT_MASTER (account_number, customer_name, introducer_name)
VALUES (1001, 'John Doe', 'Jane Smith');
INSERT INTO ACCOUNT_MASTER (account_number, customer_name, introducer_name)
VALUES (1002, 'Alice Johnson', 'Bob Miller');



DECLARE
    v_account_number NUMBER := &account_number; -- Accept account number from user
    v_customer_name VARCHAR2(100);
    v_introducer_name VARCHAR2(100);
BEGIN
    -- Retrieve customer and introducer information based on the supplied account number
    SELECT customer_name, introducer_name
    INTO v_customer_name, v_introducer_name
    FROM ACCOUNT_MASTER
    WHERE account_number = v_account_number;
```

```
    -- Insert the information into ACCOUNT_MASTER_INFO table
    INSERT INTO ACCOUNT_MASTER_INFO (account_number, customer_name,
introducer_name)
    VALUES (v_account_number, v_customer_name, v_introducer_name);

    DBMS_OUTPUT.PUT_LINE('Information inserted into ACCOUNT_MASTER_INFO table
successfully.');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Account number ' || v_account_number || ' does not
exist.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: An unexpected error occurred.');
END;
/

-- Select all records from the ACCOUNT_MASTER_INFO table
SELECT *
FROM ACCOUNT_MASTER_INFO;
```

Problem statements 22

A stored function is created to perform the ACCOUNT_NO check operation .F_checkAccNO() is the name of function which accept a variable ACCOUNT_NO and returns the value to host environment The value changes from 0(if ACCOUNT_NO does not exist) to 1(if ACCOUNT_NO exist) depending on the records retrieved.

```
CREATE TABLE ACCOUNT_MASTER (
    account_number NUMBER PRIMARY KEY,
    customer_name VARCHAR2(100),
    introducer_name VARCHAR2(100)
);


DECLARE
    v_account_number_to_check NUMBER := 1001; -- Account number to check
    v_result NUMBER;
BEGIN
    -- Call the function to check the account number
    v_result := F_checkAccNO(v_account_number_to_check);

    -- Display the result
    IF v_result = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Account number ' || v_account_number_to_check || ' exists.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Account number ' || v_account_number_to_check || ' does not
exist.');
    END IF;
END;
/



CREATE OR REPLACE FUNCTION F_checkAccNO(p_account_no IN NUMBER)
RETURN NUMBER
IS
    v_count NUMBER := 0;
BEGIN
    -- Check if the account number exists in the ACCOUNT_MASTER table
    SELECT COUNT(*)
    INTO v_count
```

```
    FROM ACCOUNT_MASTER
    WHERE account_number = p_account_no;

    -- Return 1 if account number exists, otherwise return 0
    IF v_count > 0 THEN
        RETURN 1;
    ELSE
        RETURN 0;
    END IF;
END;
/
```

Problem statements 23
create a row level trigger for the CUSTOMERS table that would fire for INSERT or UPDATE or DELETE
operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old
values and new values

```
CREATE TABLE CUSTOMERSs (
    customer_id NUMBER PRIMARY KEY,
    customer_name VARCHAR2(100),
    salary NUMBER
);


CREATE OR REPLACE TRIGGER salary_difference_trigge
AFTER INSERT OR UPDATE OR DELETE ON CUSTOMERSs
FOR EACH ROW
DECLARE
    v_old_salary NUMBER;
    v_new_salary NUMBER;
BEGIN
    IF INSERTING THEN
        DBMS_OUTPUT.PUT_LINE('New record inserted. Salary: ' || :NEW.salary);
    ELSIF UPDATING THEN
        v_old_salary := :OLD.salary;
        v_new_salary := :NEW.salary;
        DBMS_OUTPUT.PUT_LINE('Salary updated. Old Salary: ' || v_old_salary || ', New Salary: '
|| v_new_salary);
        DBMS_OUTPUT.PUT_LINE('Salary difference: ' || (v_new_salary - v_old_salary));
    ELSIF DELETING THEN
        DBMS_OUTPUT.PUT_LINE('Record deleted. Salary: ' || :OLD.salary);
    END IF;
END;
/




INSERT INTO CUSTOMERSs (customer_id, customer_name, salary) VALUES (1, 'John Doe',
50000);
INSERT INTO CUSTOMERSs(customer_id, customer_name, salary) VALUES (2, 'Alice
Johnson', 60000);




-- Update a customer's salary
```

```
UPDATE CUSTOMERSs SET salary = 55000 WHERE customer_id = 1;

-- Delete a customer
DELETE FROM CUSTOMERSs WHERE customer_id = 2;

-- Insert a new customer
INSERT INTO CUSTOMERSs (customer_id, customer_name, salary) VALUES (3, 'Bob Smith',
70000);


SET SERVEROUTPUT ON;
```

For example, if you executed an UPDATE operation on the CUSTOMERS table, you might see output like this:

yaml
Copy code
Salary updated. Old Salary: 50000, New Salary: 55000
Salary difference: 5000

Problem statements 24
Write PL/SQL block to update the Customer table and increase the salary of each customer by 500
and use the SQL%ROWCOUNTattribute to determine the number of rows affected.

```
CREATE TABLE Customer (
    customer_id NUMBER PRIMARY KEY,
    customer_name VARCHAR2(100),
    salary NUMBER
);


INSERT INTO Customer (customer_id, customer_name, salary) VALUES (1, 'John Doe',
50000);
INSERT INTO Customer (customer_id, customer_name, salary) VALUES (2, 'Alice Johnson',
60000);
INSERT INTO Customer (customer_id, customer_name, salary) VALUES (3, 'Bob Smith',
70000);


DECLARE
    v_rows_affected NUMBER;
BEGIN
    -- Update the salary of each customer by 500
    UPDATE Customer SET salary = salary + 500;

    -- Get the number of rows affected by the update statement
    v_rows_affected := SQL%ROWCOUNT;

    -- Display the number of rows affected
    DBMS_OUTPUT.PUT_LINE('Number of rows updated: ' || v_rows_affected);
END;
/
```