ASSIGNMENT NO 7

```
SET SERVEROUTPUT ON;

CREATE TABLE Accounts (
        Acc_No NUMBER PRIMARY KEY,
        Acc_Holder VARCHAR2(50),
        Last_Transaction DATE,
        Status VARCHAR2(10) CHECK (Status IN ('Active', 'Inactive'))
);

INSERT INTO Accounts VALUES (101, 'Piya Mali', TO_DATE('2022-01-10', 'YYYY-MM-DD'), 'Inactive');
INSERT INTO Accounts VALUES (102, 'Parth Kathane', TO_DATE('2023-03-15', 'YYYY-MM-DD'),
'Inactive');
INSERT INTO Accounts VALUES (103, 'Omkar Dhanawade', TO_DATE('2021-12-05', 'YYYY-MM-DD'),
'Inactive');
INSERT INTO Accounts VALUES (104, 'John Doe', TO_DATE('2023-07-20', 'YYYY-MM-DD'), 'Active');
INSERT INTO Accounts VALUES (105, 'Jane Smith', TO_DATE('2022-05-10', 'YYYY-MM-DD'), 'Inactive');

DECLARE
        v_count NUMBER;
BEGIN
        UPDATE Accounts
        SET Status = 'Active'
        WHERE Status = 'Inactive'
        AND Last_Transaction <= ADD_MONTHS(SYSDATE, -12);

        v_count := SQL%ROWCOUNT;

        IF SQL%FOUND THEN
        DBMS_OUTPUT.PUT_LINE(v_count || ' accounts have been activated.');
        ELSIF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('No inactive accounts found for activation.');
        END IF;
END;
/

select * from accounts;
```

OUTPUT:

```
  ACC_NO ACC_HOLDER                                          LAST_TRAN
---------- -------------------------------------------------- ---------
STATUS
----------
       101 Piya Mali                                          10-JAN-22
Active

       102 Parth Kathane                                      15-MAR-23
```

Active

103 Omkar Dhanawade 05-DEC-21
Active

2]

```
SET SERVEROUTPUT ON;

create table EMP(E_no NUMBER primary key,Salary NUMBER(10,2));

INSERT INTO EMP (E_no, Salary) VALUES (101, 50000);
INSERT INTO EMP (E_no, Salary) VALUES (102, 40000);
INSERT INTO EMP (E_no, Salary) VALUES (103, 60000);
INSERT INTO EMP (E_no, Salary) VALUES (104, 35000);
INSERT INTO EMP (E_no, Salary) VALUES (105, 70000);

create table increment_salary(E_no NUMBER primary key,Salary NUMBER(10,2));

declare
        cursor cur_salary is
        select E_no,Salary from EMP
        where Salary<(select avg(Salary) from EMP);

        v_E_no EMP.E_no%TYPE;
        v_Salary EMP.Salary%type;
        V_new_salary EMP.Salary%type;

begin
        open cur_salary;
        loop
        fetch cur_salary into v_E_no,v_Salary;
        exit when cur_salary%notfound;
        V_new_salary:=v_Salary*1.1;
        update EMP set Salary=V_new_salary where E_no=v_E_no;
        insert into increment_salary values (v_E_no,V_new_salary);
        end loop;
        close cur_salary;
end;
/
select * from increment_salary;
```

OUTPUT:

E_NO	SALARY
101	55000
102	44000
104	38500

4]

```
SET SERVEROUTPUT ON;
CREATE TABLE N_RollCall (
        Roll_No NUMBER PRIMARY KEY,
        Name VARCHAR2(50),
        Class VARCHAR2(20),
        Section VARCHAR2(5)
);

CREATE TABLE O_RollCall (
        Roll_No NUMBER PRIMARY KEY,
        Name VARCHAR2(50),
        Class VARCHAR2(20),
        Section VARCHAR2(5)
);
INSERT INTO N_RollCall VALUES (1, 'Piya Mali', '10th', 'A');
INSERT INTO N_RollCall VALUES (2, 'Parth Kathane', '10th', 'B');
INSERT INTO N_RollCall VALUES (3, 'Omkar Dhanawade', '9th', 'A');
INSERT INTO N_RollCall VALUES (4, 'Amit Sharma', '8th', 'C');


INSERT INTO O_RollCall VALUES (2, 'Parth Kathane', '10th', 'B');
INSERT INTO O_RollCall VALUES (3, 'Omkar Dhanawade', '9th', 'A');
INSERT INTO O_RollCall VALUES (5, 'Riya Desai', '11th', 'D');
INSERT INTO O_RollCall VALUES (6, 'Vikram Singh', '8th', 'C');
declare
cursor cur_N_RollCall (p_roll_no NUMBER) is
select Roll_No,Name,Class,Section from N_RollCall where Roll_No=p_roll_no;

v_roll_no NUMBER;
v_name VARCHAR2(50);
v_class VARCHAR2(20);
v_section VARCHAR2(5);
v_count NUMBER;

begin
        for rec in (select * from N_RollCall) loop

        select count(*) into v_count from O_RollCall where Roll_No=rec.Roll_No;
        if v_count=0 then
        open cur_N_RollCall(rec.Roll_No);
        fetch cur_N_RollCall into v_roll_no,v_name,v_class,v_section;
        close cur_N_RollCall;
        INSERT INTO O_RollCall VALUES(v_roll_no,v_name,v_class,v_section);
        end if;
        end loop;
        DBMS_OUTPUT.PUT_LINE('Data merged successfully!');
end;
```

```
/
select * from O_RollCall ;
```

5]

```
SET SERVEROUTPUT ON;
CREATE TABLE EMP (
        e_no NUMBER PRIMARY KEY,
        d_no NUMBER,
        Salary NUMBER
);

INSERT INTO EMP VALUES (101, 1, 50000);
INSERT INTO EMP VALUES (102, 1, 55000);
INSERT INTO EMP VALUES (103, 2, 60000);
INSERT INTO EMP VALUES (104, 2, 62000);
INSERT INTO EMP VALUES (105, 3, 70000);

CREATE TABLE dept_salary (
        d_no NUMBER PRIMARY KEY,
        Avg_salary NUMBER
);

DECLARE
        CURSOR cur_avg_salary (p_d_no NUMBER) IS
        SELECT AVG(Salary) FROM EMP WHERE d_no = p_d_no;
        v_avg_salary NUMBER;

BEGIN

        FOR rec IN (SELECT DISTINCT d_no FROM EMP) LOOP

        OPEN cur_avg_salary(rec.d_no);
        FETCH cur_avg_salary INTO v_avg_salary;
        CLOSE cur_avg_salary;


        INSERT INTO dept_salary VALUES (rec.d_no, v_avg_salary);

        END LOOP;

        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Department-wise average salary inserted successfully!');

END;
/

select * from dept_salary;
```

OUTPUT:

```
D_NO AVG_SALARY
---------- ----------
         1      52500
         2      61000
         3      70000
```

6]

```
SET SERVEROUTPUT ON;

CREATE TABLE stud21 (
        roll NUMBER(4) PRIMARY KEY,
        att NUMBER(4),
        status VARCHAR(1)
);

INSERT INTO stud21 VALUES (101, 80, NULL);
INSERT INTO stud21 VALUES (102, 70, NULL);
INSERT INTO stud21 VALUES (103, 60, NULL);
INSERT INTO stud21 VALUES (104, 90, NULL);
INSERT INTO stud21 VALUES (105, 65, NULL);

CREATE TABLE d_stud (
        roll NUMBER(4) PRIMARY KEY,
        att NUMBER(4)
);

DECLARE
        cursor atttendance is
        select roll,att from stud21 where att<75;
begin
        for student_rec in atttendance loop
        update stud21 set status='D' where roll=student_rec.roll;

        insert into d_stud(roll,att) VALUES (student_rec.roll,student_rec.att);
        end loop;
end;
/
select * from d_stud;
```

OUTPUT:

```
    ROLL        ATT
----------  ----------
     102          70
     103          60
     105          65
```

ASSIGNMENT NO 6

1]

```
SET SERVEROUTPUT ON;
DECLARE
        v_num NUMBER := &input_number;
        v_factorial NUMBER := 1;
begin
        IF v_num < 0 THEN
        DBMS_OUTPUT.PUT_LINE('Factorial is not defined for negative numbers.');
        ELSE
        FOR i IN 1..v_num LOOP
        v_factorial := v_factorial * i;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('Factorial of ' || v_num || ' is: ' || v_factorial);
        END IF;
END;
/
```

OUTPUT:

```
Enter value for input_number: 6
Factorial of 6 is: 720
```

2]

```
SET SERVEROUTPUT ON;
create function primecheck(num in NUMBER)
return number
is
        cnt number:=0;
begin
        for i in 1..num loop
        if(num mod i=0) then
        cnt:=cnt+1;
        end if;
        end loop;
        return cnt;
end;
/

DECLARE
        v_num NUMBER := &input_number;
        v_count number;

begin
        v_count:=primecheck(v_num);
        if v_num=1 then
        dbms_output.put_line(v_num || ' is neither prime nor composite.');
```

```
        elsif v_count=2 then
        dbms_output.put_line(v_num||' is a prime number.');
        else
        dbms_output.put_line(v_num||' is a composite number.');
        end if;
END;
/
```

OUTPUT:

```
Enter value for input_number: 45
45 is a composite number.
```

3]

```
SET SERVEROUTPUT ON;
CREATE TABLE departments (
        department_id NUMBER PRIMARY KEY,
        department_name VARCHAR2(50) NOT NULL
);

INSERT INTO departments VALUES (1, 'HR');
INSERT INTO departments VALUES (2, 'IT');
INSERT INTO departments VALUES (3, 'Sales');
INSERT INTO departments VALUES (4, 'Finance');
INSERT INTO departments VALUES (5, 'Marketing');

select * from departments;

CREATE TABLE employees (
        employee_id NUMBER PRIMARY KEY,
        employee_name VARCHAR2(50) NOT NULL,
        department_id NUMBER,
        FOREIGN KEY (department_id) REFERENCES departments(department_id)
);

INSERT INTO employees VALUES (101, 'Piya Mali', 1);
INSERT INTO employees VALUES (102, 'Parth Kathane', 1);
INSERT INTO employees VALUES (103, 'Omkar Dhanawade', 2);
INSERT INTO employees VALUES (104, 'Sandesh Phad', 2);
INSERT INTO employees VALUES (105, 'Ayush Deshmukh', 2);
INSERT INTO employees VALUES (106, 'Sushant Mali', 3);
INSERT INTO employees VALUES (107, 'Mangesh Mali', 3);
INSERT INTO employees VALUES (108, 'Mahesh Mali', 3);
INSERT INTO employees VALUES (109, 'Ruturaj Thalkar', 4);
INSERT INTO employees VALUES (110, 'Sarang Jadhav', 4);

select * from employees;

SET SERVEROUTPUT ON;
DECLARE
```

```
        v_department_name departments.department_name%TYPE;
        v_employee_count NUMBER;

        CURSOR dept_cursor IS
        SELECT d.department_name, COUNT(e.employee_id) AS emp_count
        FROM departments d
        LEFT JOIN employees e ON d.department_id = e.department_id
        GROUP BY d.department_name;
BEGIN
        OPEN dept_cursor;
        LOOP
        FETCH dept_cursor INTO v_department_name, v_employee_count;
        EXIT WHEN dept_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Department: ' || v_department_name || ', Employees: ' ||
v_employee_count);
        END LOOP;
        CLOSE dept_cursor;
END;
/
```

OUTPUT:

```
Department: HR, Employees: 2
Department: IT, Employees: 3
Department: Sales, Employees: 3
Department: Finance, Employees: 2
Department: Marketing, Employees: 0
```

4]

```
SET SERVEROUTPUT ON;
CREATE TABLE students (
        Roll_no NUMBER PRIMARY KEY,
        Name VARCHAR2(30),
        Marks NUMBER
);
INSERT INTO students (Roll_no, Name, Marks)
VALUES (1, 'Piya Mali', 85);

INSERT INTO students (Roll_no, Name, Marks)
VALUES (2, 'Parth Kathane', 90);

INSERT INTO students (Roll_no, Name, Marks)
VALUES (3, 'Omkar Dhanawade', 78);
select * from students;

begin
        begin
        EXECUTE IMMEDIATE 'alter TABLE students add Grade char(1)';
        exception
        when others then
```

```
        null;
        end;
end;
/

create procedure calculategrade(marks in number,grade out char)
is
begin
        if marks>=90 then
        grade:='A';
        elsif marks>=75 and marks<90 then
        grade:='B';
        elsif marks>=60 and marks<75 then
        grade:='C';
        else
        grade:='D';
        end if;
end;
/

begin

        DECLARE
        n number:=0;
        v_marks number;
        v_grade char(1);
        begin
        select count(*) into n from students;
        for i in 1..n loop
        select Marks into v_marks from students where Roll_no=i;
        calculategrade(v_marks,v_grade);
        update students set Grade=v_grade where Roll_no=i;
        end loop;
        end;
END;
/
select * from students;
```

OUTPUT:

```
ROLL_NO NAME                           MARKS
---------- ------------------------------ ----------
         1 Piya Mali                              85
         2 Parth Kathane                          90
         3 Omkar Dhanawade                        78


PL/SQL procedure successfully completed.
Procedure created.
PL/SQL procedure successfully completed.
```

```
  ROLL_NO NAME                           MARKS G
---------- ------------------------------ ---------- -
         1 Piya Mali                              85 B
         2 Parth Kathane                          90 A
         3 Omkar Dhanawade                        78 B
```

5]

```
SET SERVEROUTPUT ON;
CREATE TABLE accounts (
        account_no NUMBER PRIMARY KEY,
        account_name VARCHAR2(50),
        balance NUMBER
);

INSERT INTO accounts VALUES (101, 'Piya Mali', 5000);
INSERT INTO accounts VALUES (102, 'Parth Kathane', 3000);
INSERT INTO accounts VALUES (103, 'Omkar Dhanawade', 7000);

COMMIT;

DECLARE
        v_account_no NUMBER;
        v_withdraw_amount NUMBER;
        v_balance NUMBER;
        v_name VARCHAR2(50);

        INSUFFICIENT_BALANCE EXCEPTION;
BEGIN
        DBMS_OUTPUT.PUT_LINE('Enter Account Number:');
        v_account_no := &account_no;

        DBMS_OUTPUT.PUT_LINE('Enter Withdrawal Amount:');
        v_withdraw_amount := &withdraw_amount;

        SELECT balance, account_name INTO v_balance, v_name FROM accounts WHERE
account_no = v_account_no;

        IF v_withdraw_amount > v_balance THEN
        RAISE INSUFFICIENT_BALANCE;
        ELSE
        UPDATE accounts
        SET balance = balance - v_withdraw_amount
        WHERE account_no = v_account_no;

        DBMS_OUTPUT.PUT_LINE('Transaction successful!');
        DBMS_OUTPUT.PUT_LINE('Account Holder: ' || v_name);
        DBMS_OUTPUT.PUT_LINE('Remaining Balance: ' || (v_balance - v_withdraw_amount));
```

```
        END IF;

EXCEPTION

        WHEN INSUFFICIENT_BALANCE THEN
        DBMS_OUTPUT.PUT_LINE('Error: Insufficient balance for this transaction.');

        WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Account number not found.');

        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

OUTPUT:

```
Enter value for account_no: 101
Enter value for withdraw_amount: 400
Transaction successful!
Account Holder: Piya Mali
Remaining Balance: 4300
```

ASSIGNMENT NO 5

```
use assignment;
drop table add_dets;
CREATE TABLE cust_mstr (
        cust_no INT PRIMARY KEY,
        fname VARCHAR(50),
        lname VARCHAR(50)
);

CREATE TABLE add_dets (
        code_no INT PRIMARY KEY,
        cust_no INT,  -- Foreign key to link with cust_mstr
        add1 VARCHAR(100),
        add2 VARCHAR(100),
        state VARCHAR(50),
        city VARCHAR(50),
        pincode VARCHAR(10),
        FOREIGN KEY (cust_no) REFERENCES cust_mstr(cust_no)
);
```

```
INSERT INTO cust_mstr (cust_no, fname, lname) VALUES
(101, 'Amit', 'Sharma'),
(102, 'Priya', 'Verma'),
(103, 'Rajesh', 'Gupta'),
(104, 'Neha', 'Patel'),
(105, 'Suresh', 'Yadav');

INSERT INTO add_dets (code_no, cust_no, add1, add2, state, city, pincode) VALUES
(201, 101, '123, MG Road', 'Near City Mall', 'Maharashtra', 'Mumbai', '400001'),
(202, 102, '45, Nehru Nagar', 'Opp. Park', 'Karnataka', 'Bangalore', '560001'),
(203, 103, '78, Gandhi Marg', 'Behind Hospital', 'Delhi', 'New Delhi', '110001'),
(204, 104, '89, Residency Road', 'Near Bus Stand', 'Gujarat', 'Ahmedabad', '380001'),
(205, 105, '150, Anna Salai', 'Close to Metro', 'Tamil Nadu', 'Chennai', '600001');

-- Query: Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'
SELECT a.add1, a.add2, a.state, a.city, a.pincode
FROM cust_mstr c
JOIN add_dets a ON c.cust_no = a.cust_no
WHERE c.fname = 'Suresh' AND c.lname = 'Yadav';

-- Customer-FD Relationship Table
CREATE TABLE acc_fd_cust_dets (
        codeno INT,
        acc_fd_no INT PRIMARY KEY
);

-- FD Details Table
CREATE TABLE fd_dets (
        fd_sr_no INT PRIMARY KEY,
        acc_fd_no INT,
        amt DECIMAL(12,2),
        FOREIGN KEY (acc_fd_no) REFERENCES acc_fd_cust_dets(acc_fd_no)
);

INSERT INTO acc_fd_cust_dets (codeno, acc_fd_no) VALUES
(201, 5001),
(202, 5002),
(203, 5003),
(204, 5004),
(205, 5005);

INSERT INTO fd_dets (fd_sr_no, acc_fd_no, amt) VALUES
(1, 5001, 5000.00),
(2, 5002, 120000.50),
```

```
(3, 5003, 98000.25),
(4, 5004, 200000.00),
(5, 5005, 150000.75);

-- Query: List the customer holding fixed deposit of amount more than 5000
select c.cust_no,c.fname,c.lname from cust_mstr c
join add_dets a on a.cust_no=c.cust_no
join acc_fd_cust_dets ac on a.code_no=ac.codeno
join fd_dets f on f.acc_fd_no=ac.acc_fd_no
where f.amt>5000;


CREATE TABLE branch_mstr (
        b_no INT PRIMARY KEY,
        name VARCHAR(100)
);


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) ON DELETE
CASCADE
);

INSERT INTO branch_mstr (b_no, name) VALUES
(201, 'Shivaji Nagar Branch'),
(202, 'Kothrud Branch'),
(203, 'Hinjawadi IT Park Branch'),
(204, 'Viman Nagar Branch'),
(205, 'Hadapsar Branch');

INSERT INTO emp_mstr (e_mpno, f_name, l_name, m_name, dept, desg, branch_no) VALUES
(1, 'Amit', 'Sharma', 'Kumar', 'IT', 'Software Engineer', 201),
(2, 'Priya', 'Verma', 'Rani', 'HR', 'HR Manager', 202),
(3, 'Rajesh', 'Gupta', 'Kumar', 'Finance', 'Accountant', 203),
(4, 'Neha', 'Patel', 'Devi', 'Marketing', 'Marketing Head', 204),
(5, 'Suresh', 'Yadav', 'Singh', 'Operations', 'Operations Manager', 205);
```

```
SELECT e.e_mpno, e.f_name, e.l_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;

CREATE TABLE emp_mstr2 (
        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,
        emp_no INT,
        cntc_type VARCHAR(50),
        cntc_data VARCHAR(100),
        FOREIGN KEY (emp_no) REFERENCES emp_mstr2(emp_no) ON DELETE CASCADE
ON UPDATE CASCADE
);


INSERT INTO emp_mstr2 (emp_no, f_name, l_name, m_name, dept) VALUES
(1, 'Amit', 'Sharma', 'Kumar', 'IT'),
(2, 'Priya', 'Verma', 'Rani', 'HR'),
(3, 'Rajesh', 'Gupta', 'Kumar', 'Finance'),
(4, 'Neha', 'Patel', 'Devi', 'Marketing'),
(5, 'Suresh', 'Yadav', 'Singh', 'Operations');

INSERT INTO cntc_dets (code_no, emp_no, cntc_type, cntc_data) VALUES
(101, 1, 'Phone', '9876543210'),
(102, 1, 'Email', 'amit.sharma@example.com'),
(103, 2, 'Phone', '9123456789'),
(104, 3, 'Email', 'rajesh.gupta@example.com'),
(105, 4, 'Phone', '9765432109'),
(106, 5, 'Email', 'suresh.yadav@example.com');

-- Query: List the employee details along with contact details using left outer join & right join
SELECT e.emp_no, e.f_name, e.l_name, e.dept, c.cntc_type, c.cntc_data
FROM emp_mstr2 e
LEFT OUTER JOIN cntc_dets c ON e.emp_no = c.emp_no;

-- Customer Master Table
CREATE TABLE cust_mstr3 (
```

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


CREATE TABLE add_dets3 (
	code_no INT PRIMARY KEY,
	cust_no INT,
	pincode VARCHAR(10),
	FOREIGN KEY (cust_no) REFERENCES cust_mstr3(cust_no) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE bank_branch (
	branch_id INT PRIMARY KEY,
	branch_name VARCHAR(100),
	pincode VARCHAR(10)
);


INSERT INTO cust_mstr3 (cust_no, fname, lname) VALUES
(1, 'Amit', 'Sharma'),
(2, 'Priya', 'Verma'),
(3, 'Rajesh', 'Gupta'),
(4, 'Neha', 'Patel'),
(5, 'Suresh', 'Yadav');

INSERT INTO add_dets3 (code_no, cust_no, pincode) VALUES
(101, 1, '411001'),
(102, 2, '411002'),
(103, 3, '411003'),
(104, 4, '411004'),
(105, 5, '411005');


INSERT INTO bank_branch (branch_id, branch_name, pincode) VALUES
(201, 'SBI Shivaji Nagar', '411001'),
(202, 'HDFC Kothrud', '411002'),
(203, 'ICICI Hinjawadi', '411003');

-- Query: List the customer who do not have bank branches in their vicinity.
SELECT c.cust_no, c.fname, c.lname, a.pincode
FROM cust_mstr3 c
```

```
JOIN add_dets3 a ON c.cust_no = a.cust_no
LEFT JOIN bank_branch b ON a.pincode = b.pincode
WHERE b.pincode IS NULL;

-- 6. Queries on Views

CREATE TABLE borrower2 (
        borrower_id INT PRIMARY KEY,
        name VARCHAR(50),
        loan_amount DECIMAL(10,2)
);


CREATE TABLE depositor2 (
        depositor_id INT PRIMARY KEY,
        name VARCHAR(50),
        deposit_amount DECIMAL(10,2)
);

-- a) Create View on borrower table by selecting any two columns and perform insert update
delete operations Borrower Table
CREATE VIEW borrower_view AS SELECT borrower_id, name FROM borrower2;

INSERT INTO borrower_view (borrower_id, name) VALUES (101, 'Amit Sharma');
UPDATE borrower_view SET name = 'Amit Kumar' WHERE borrower_id = 101;
DELETE FROM borrower_view WHERE borrower_id = 101;

-- b) Create view on borrower and depositor table by selecting any one column from each table
perform insert update delete operations
CREATE VIEW borrower_depositor_view AS
SELECT b.name AS borrower_name, d.name AS depositor_name
FROM borrower2 b
JOIN depositor2 d ON b.borrower_id = d.depositor_id;
INSERT INTO borrower2 (borrower_id, name, loan_amount) VALUES (102, 'Rajesh Gupta',
50000);
INSERT INTO depositor2 (depositor_id, name, deposit_amount) VALUES (102, 'Rajesh Gupta',
70000);

-- c) create updateable view on borrower table by selecting any two columns and perform insert
update delete operations.
CREATE VIEW borrower_update_view AS SELECT borrower_id, loan_amount FROM
borrower2 WITH CHECK OPTION;

INSERT INTO borrower_update_view (borrower_id, loan_amount) VALUES (103, 75000);
```

```
UPDATE borrower_update_view SET loan_amount = 80000 WHERE borrower_id = 103;
DELETE FROM borrower_update_view WHERE borrower_id = 103;
```

ASSIGNMENT NO 4

```
use assignment;
drop table Books;
drop table Members;
drop table IssuedBooks;
CREATE TABLE Books (
        BookID INT PRIMARY KEY,
        Title VARCHAR(255) NOT NULL,
        Author VARCHAR(255) NOT NULL,
        Genre VARCHAR(100),
        Price DECIMAL(10,2),
        PublishedYear INT,
        Quantity INT CHECK (Quantity >= 0)
);

CREATE TABLE Members (
        MemberID INT PRIMARY KEY,
        Name VARCHAR(255) NOT NULL,
        MembershipType VARCHAR(50),
        JoinDate DATE NOT NULL
);

CREATE TABLE IssuedBooks (
        IssueID INT PRIMARY KEY,
        BookID INT,
        MemberID INT,
        IssueDate DATE NOT NULL,
        ReturnDate DATE,
        FOREIGN KEY (BookID) REFERENCES Books(BookID),
        FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
);

INSERT INTO Books (BookID, Title, Author, Genre, Price, PublishedYear, Quantity) VALUES
(1, 'Mrutyunjay', 'Shivaji Sawant', 'Historical Fiction', 18.99, 1989, 6),
(2, 'Yayati', 'V.S. Khandekar', 'Mythological Fiction', 14.50, 1959, 5),
(3, 'Chhava', 'Shivaji Sawant', 'Historical Fiction', 16.75, 1980, 7),
(4, 'Shriman Yogi', 'Ranjit Desai', 'Historical', 20.00, 1967, 4),
```

```
(5, 'Batatyachi Chaal', 'P.L. Deshpande', 'Humor', 12.99, 1960, 8),
(6, 'Vyakti Ani Valli', 'P.L. Deshpande', 'Humor', 11.50, 1974, 6),
(7, 'Swami', 'Ranjit Desai', 'Historical Fiction', 15.00, 1970, 5),
(8, 'Panipat', 'Vishwas Patil', 'Historical', 17.25, 1990, 6);

INSERT INTO Members (MemberID, Name, MembershipType, JoinDate) VALUES
(101, 'Suresh Patil', 'Premium', '2023-01-15'),
(102, 'Amruta Kulkarni', 'Regular', '2023-03-10'),
(103, 'Vijay Deshmukh', 'Premium', '2023-05-22'),
(104, 'Neha Jadhav', 'Regular', '2023-06-18'),
(105, 'Akshay Shinde', 'Premium', '2023-07-30');

INSERT INTO IssuedBooks (IssueID, BookID, MemberID, IssueDate, ReturnDate) VALUES
(201, 1, 101, '2024-01-05', '2024-01-20'),
(202, 5, 102, '2024-01-10', '2024-01-25'),
(203, 3, 103, '2024-01-15', NULL),
(204, 7, 104, '2024-01-18', NULL),
(205, 6, 105, '2024-01-20', '2024-02-01'),
(206, 2, 102, '2024-01-20', '2024-02-01');


-- 2. Write a query to display all book titles along with their authors and genres.
select Title,Author,Genre from Books;

-- 3. Write a query to list the names of members who joined in the last year.
select Name from Members where year(JoinDate)=year(curdate())-1;

-- 4. Write a query to update the quantity of a book by reducing it when it is issued.
update Books set Quantity=Quantity-1 where BookID in (SELECT BookID FROM IssuedBooks);

-- 5. Write a query to delete records of books that were published before the year 2000.
SET SQL_SAFE_UPDATES = 0;
delete from Books where PublishedYear<2000;
SET SQL_SAFE_UPDATES = 1;
select * from books;

-- 6. Write a query to find the average price of books for each genre.
select Genre,avg(price) from Books group by Genre;

-- 7. Write a query to list all the members who have issued more than 2 books.
select Members.Name from Members join IssuedBooks on
Members.MemberID=IssuedBooks.MemberID group by IssuedBooks.MemberID having
count(IssuedBooks.MemberID)>1;
```

-- 8. Write a query to display all books that belong to either the 'Fiction' or 'Science'genre.
select Title from Books where Genre in ('Historical Fiction','Historical');

-- 9. Write a query to count the total number of books issued in the current month.
select count(*) from IssuedBooks where month(IssueDate)=month(curdate()) and
year(IssueDate)=year(curdate());

-- 10. Write a query to display all overdue books, i.e., books that have not been returned and whose ReturnDate is earlier than today.
select Name from Members join IssuedBooks on Members.MemberID=IssuedBooks.MemberID
where ReturnDate<curdate();
select * from IssuedBooks where ReturnDate<curdate();
—----------

ASSIGNMENT NO-3

```
use assignment;
drop table Departments;
drop table Employees;
drop table Projects;
CREATE TABLE Departments (
DepartmentID INT PRIMARY KEY,
DepartmentName VARCHAR(100));

CREATE TABLE Employees (
EmployeeID INT PRIMARY KEY,
EmployeeName VARCHAR(100),
Age INT,
Salary DECIMAL(10, 2),
DepartmentID INT,
FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID));

CREATE TABLE Projects (
ProjectID INT PRIMARY KEY,
ProjectName VARCHAR(100),
EmployeeID INT,
FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID));

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES
(1, 'Software Development'),
(2, 'Human Resources'),
(3, 'Finance'),
(4, 'Marketing'),
(5, 'Operations');
```

```
select * from Departments;

INSERT INTO Employees (EmployeeID, EmployeeName, Age, Salary, DepartmentID) VALUES
(101, 'Amit Sharma', 28, 75000.00, 1),
(102, 'Neha Verma', 32, 82000.00, 2),
(103, 'Rohit Mehta', 29, 90000.00, 3),
(104, 'Priya Iyer', 26, 72000.00, 4),
(105, 'Vikram Singh', 35, 95000.00, 5),
(106, 'Kiran Desai', 30, 77000.00, 1),
(107, 'Sanjay Gupta', 40, 88000.00, 3),
(108, 'Pooja Reddy', 27, 73000.00, 2),
(109, 'Ankit Joshi', 31, 86000.00, 5),
(110, 'Meera Nair', 33, 91000.00, 4);

select * from Employees;

INSERT INTO Projects (ProjectID, ProjectName, EmployeeID) VALUES
(201, 'E-commerce Platform', 101),
(202, 'HR Management System', 102),
(203, 'Financial Analysis Tool', 103),
(204, 'Social Media Marketing', 104),
(205, 'Logistics Optimization', 105),
(206, 'AI Chatbot Development', 106),
(207, 'Taxation Software', 107),
(208, 'Recruitment Portal', 108),
(209, 'Supply Chain Dashboard', 109),
(210, 'Advertising Campaign Tracker', 110);

--------------------------------------------------------------------------
-- 1. Find all employees older than 30
select * from Employees where age>30;

-- 2. List all departments with more than 5 employees
select Departments.DepartmentName,count(Employees.EmployeeID) AS TotalEmployees from
Departments join Employees on Departments.DepartmentID=Employees.DepartmentID group
by Departments.DepartmentName having count(Employees.EmployeeID)>1;

-- 3. Display employees' names and their departments.
select Employees.EmployeeName,Departments.DepartmentName from Employees join
Departments on Departments.DepartmentID=Employees.DepartmentID;

-- 4. Find the highest salary in each department.
SELECT DepartmentID, MAX(Salary) AS HighestSalary
FROM Employees
```

```
GROUP BY DepartmentID;

-- 5. Find the average salary of employees.
select avg(salary) from Employees;

-- 6. List employees working on multiple projects.
update Projects set EmployeeID=101 where ProjectID=207;
select * from projects;
select Employees.EmployeeName from Employees join Projects on
Employees.EmployeeID=Projects.EmployeeID GROUP BY Employees.EmployeeID having
count(Projects.ProjectID)>1;

-- 7. Retrieve employees sorted by salary in descending order.
select EmployeeName,Salary from Employees order by Salary desc;

-- 9. Find the youngest employee in each department.
select EmployeeName from Employees where Age=(select min(age) from Employees);

-- 10. Find employees whose name starts with A.
select EmployeeName from Employees where EmployeeName like 'A%';

-- 11. List employees and their project names (if any).
select Employees.EmployeeName,Projects.ProjectName from Employees join Projects on
Employees.EmployeeID=Projects.EmployeeID;

-- 12. Count the total number of employees in each department.
select Departments.DepartmentName,count(Employees.EmployeeID) AS TotalEmployees from
Departments join Employees on Departments.DepartmentID=Employees.DepartmentID group
by Departments.DepartmentName;

-- 13. Find the department with the highest average salary.
select Departments.DepartmentName from Departments join Employees on
Departments.DepartmentID=Employees.DepartmentID group by
Departments.DepartmentName having max(avg(Employees.Salary));

-- 14. Display employees not assigned to any project.
select * from Employees left join Projects on Employees.EmployeeID=Projects.EmployeeID;
select Employees.EmployeeName from Employees left join Projects on
Employees.EmployeeID=Projects.EmployeeID where Projects.EmployeeID is null;

-- 15. Find employees aged between 25 and 40.
select EmployeeName from Employees where age between 25 and 40;

-- 16. List projects with more than 3 employees assigned.
```

```
select ProjectName from Projects group by projectName having count(EmployeeID)>1;
SELECT Projects.ProjectName
FROM Projects
GROUP BY Projects.ProjectID, Projects.ProjectName
HAVING COUNT(Projects.EmployeeID) > 1;

-- 17. Find the employee with the longest name.
select EmployeeName from Employees order by length(EmployeeName) desc limit 1;

-- 18. Calculate the total number of projects handled by each department.
select Departments.DepartmentName,count(Projects.ProjectName) from Departments
join Employees on Employees.DepartmentID=Departments.DepartmentID
join Projects on Employees.EmployeeID=Projects.EmployeeID group by
Departments.DepartmentName;

-- 19. Find all employees earning less than the department average salary.
select EmployeeName from Employees E where salary < (select avg(salary) from Employees
where DepartmentID = E.DepartmentID);

-- 20. List the department with the maximum number of employees.
SELECT DepartmentName
FROM Departments
WHERE DepartmentID = (
        SELECT DepartmentID
        FROM Employees
        GROUP BY DepartmentID
        ORDER BY COUNT(EmployeeID) DESC
        LIMIT 1);
```

ASSIGNMENT NO 2

```
use assignment;
drop table stu_info;
CREATE TABLE stu_info (
        prn INT PRIMARY KEY,
        name VARCHAR(50),
        college VARCHAR(50),
        department VARCHAR(50),
        year INT,
        address VARCHAR(50) DEFAULT 'pune',
        age INT CHECK(age > 17)
);

INSERT INTO stu_info (prn, name, college, department, year, address, age) VALUES
```

```
(1, 'Sneha', 'pccoe', 'aiml', 1, 'pune', 18),
(2, 'Piya', 'pict', 'cse', 2, 'pune', 19),
(3, 'Mangesh', 'dyp', 'mech', 3, 'pune', 20),
(6, 'Parth', 'pccoe', 'it', 4, 'pune', 21);

SELECT * FROM stu_info;

-- 1. Update the department for a specific student (PRN 3)
UPDATE stu_info SET department = 'cse' WHERE prn = 3;

-- 2. Retrieve students from 'PCCOE' only
SELECT * FROM stu_info WHERE college = 'pccoe';

-- 3.. Find students who are in 2nd year
SELECT * FROM stu_info WHERE year = 2;

-- 4.. Count the number of students from each college
SELECT college, COUNT(*) AS total_students FROM stu_info GROUP BY college;

-- 5.Find the student with the highest age
SELECT * FROM stu_info ORDER BY age DESC LIMIT 1;

-- 6.Retrieve students sorted by department in ascending order
SELECT * FROM stu_info ORDER BY department ASC;

-- 7.Retrieve the names of students who are from 'PCCOE' and are in their final year (4th year)
SELECT name FROM stu_info WHERE college = 'pccoe' AND year = 4;

-- 8.Select students whose names start with 'P' and are from 'pict' college
SELECT * FROM stu_info WHERE name LIKE 'P%' AND college = 'pict';

-- 9.Find students who are not in AIML or CSE department
SELECT * FROM stu_info WHERE department NOT IN ('aiml', 'cse');

-- 10.Retrieve the department with the highest number of students
SELECT department FROM stu_info
GROUP BY department
ORDER BY COUNT(*) DESC
LIMIT 1;
```

ASSIGNMENT NO 1

```
use assignment;

drop table borrower;
drop table depositor;
drop table loan;
drop table account;
drop table branch;
drop table customer;

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

create table branch (branch_name varchar(20) primary key,branch_city varchar(20),assets
decimal (15,2) check(assets>0));

create table account (Acc_no int primary key,branch_name varchar(20),balance decimal (15,2)
check(balance>0),
FOREIGN KEY (branch_name) REFERENCES branch(branch_name) ON DELETE
CASCADE);

create table depositor(cust_name varchar(20),Acc_no int,
foreign key (cust_name) references customer(cust_name) on delete cascade,
foreign key (Acc_no) references account(Acc_no) on delete cascade);

create table loan(loan_no int primary key,branch_name varchar(20),amount decimal (15,2)
check(amount>0),
FOREIGN KEY (branch_name) REFERENCES branch(branch_name) ON DELETE
CASCADE);

create table borrower(cust_name varchar(20),loan_no int,
foreign key (cust_name) references customer(cust_name) on delete cascade,
foreign key (loan_no) references loan(loan_no) on delete cascade);

INSERT INTO customer (cust_name, cust_street, cust_city) VALUES
('Rahul Sharma', 'Sector 27', 'Pimpri-Chinchwad'),
('Priya Patil', 'Sector 10', 'Solapur'),
('Amit Joshi', 'Sector 5', 'Nagpur'),
('Sneha Kulkarni', 'Sector 15', 'Ichalkaranji'),
('Rohan Desai', 'Sector 20', 'Kolhapur'),
('Sagar Mehta', 'Karve Road', 'Pune'),
('Neha Jadhav', 'Panchavati', 'Nashik');

INSERT INTO account (Acc_no, branch_name, balance) VALUES
(101, 'Akurdi', 5000.00),
```

```
(102, 'Nigdi', 15000.00),
(103, 'Chinchwad', 8000.00),
(104, 'Bhosari', 12000.00),
(105, 'Akurdi', 350.00),
(106, 'Kothrud', 10000.00),
(107, 'Nashik Road', 7000.00);

INSERT INTO branch (branch_name, branch_city, assets) VALUES
('Akurdi', 'Pimpri-Chinchwad', 3000000.00),
('Nigdi', 'Pimpri-Chinchwad', 2500000.00),
('Chinchwad', 'Pimpri-Chinchwad', 4000000.00),
('Bhosari', 'Pimpri-Chinchwad', 2000000.00),
('Kothrud', 'Pune', 3500000.00),
('Nashik Road', 'Nashik', 2800000.00);

INSERT INTO Depositor (cust_name, acc_no) VALUES
('Rahul Sharma', 101),
('Priya Patil', 102),
('Amit Joshi', 103),
('Sneha Kulkarni', 104),
('Rohan Desai', 105),
('Sagar Mehta', 106),
('Neha Jadhav', 107);

INSERT INTO Loan (loan_no, branch_name, amount) VALUES
(201, 'Akurdi', 15000.00),
(202, 'Nigdi', 1300.00),
(203, 'Chinchwad', 1350.00),
(204, 'Bhosari', 12000.00),
(205, 'Akurdi', 5000.00),
(206, 'Kothrud', 22000.00),
(207, 'Nashik Road', 9000.00);

INSERT INTO Borrower (cust_name, loan_no) VALUES
('Rahul Sharma', 201),
('Priya Patil', 202),
('Amit Joshi', 203),
('Sneha Kulkarni', 204),
('Sagar Mehta', 206),
('Neha Jadhav', 207);

-- 1. Find the names of all branches in loan relation.
select distinct branch_name from loan;
```

```
-- 2. Find all loan numbers for loans made at Akurdi Branch with loan amount >12000.
select loan_no from loan where branch_name='Akurdi' and amount>12000;

-- 3. Find no. of depositors at each branch.
select account.branch_name,count(Depositor.cust_name) from account join Depositor on
account.Acc_no=Depositor.Acc_no group by account.branch_name;

-- 4. Delete all loans with loan amount between 1300 and 1500.
SET SQL_SAFE_UPDATES = 0;
delete from loan where amount between 1300 and 1500;

-- 5. Delete all tuples at every branch located in Nigdi.
select * from loan;
delete from branch where branch_name='Nigdi';

-- 8. Find the names of all customers who have taken loans.
select * from branch;
select distinct cust_name from Borrower;

-- 9. Find the names of all customers who have not taken loans.
select cust_name from customer where cust_name not in (select cust_name from Borrower);

-- 10. Find the name, account number, and balance of all customers who have an account with
account balance of 400 or less.
select Depositor.cust_name,account.Acc_no,account.balance from Depositor join account on
account.Acc_no=Depositor.Acc_no where account.balance<=400;

-- 11. Find the name, account number, and balance of all customers who have an account.
select Depositor.cust_name,account.Acc_no,account.balance from Depositor join account on
account.Acc_no=Depositor.Acc_no;

-- 12. Find the name of all branches with assets between one and four million.
select branch_name from branch where assets between 1000000 and 4000000;

-- 13. Alter table customer by adding Contact_details column.
alter table customer add column contact_details int;

-- 14. Alter table customer by removing Contact_details column.
alter table customer drop column contact_details;

-- 15. Drop table Depositor.
drop table depositor;

-- 16. Truncate table Borrower.
```

```
TRUNCATE TABLE Borrower;

--------------------------------------------------------------------------
-- Q.2 Create table college (college_id primary key, college_code, college-name)

create table college (college_id int primary key,college_code varchar(10),college_name
varchar(20));

INSERT INTO college (college_id, college_code, college_name) VALUES
(1, 'PCCOE123', 'Pimpri Chinchwad College of Engineering'),
(2, 'COEP456', 'College of Engineering Pune'),
(3, 'PICT789', 'Pune Institute of Computer Technology');
alter table college modify column college_name varchar(40);
select * from college;

-- 1. Create Index College_Index using using any column.
create index college_index on college(college_name);
show indexes from college;

-- 2. Create unique index for unique values.
create unique index unique_college_code on college(college_code);
show indexes from college;

-- 3. Remove index from tables.
drop index college_index on college;
drop index unique_college_code on college;
show indexes from college;

--------------------------------------------------------------------------
-- Q.3 Create synonym for customer table as cust.
CREATE VIEW cust AS SELECT * FROM customer;
select* from cust;
--------------------------------------------------------------------------
-- Q.4 Create sequence roll_seq and use in student table for roll_no column.
CREATE TABLE student (
        roll_no INT PRIMARY KEY auto_increment,
        student_name VARCHAR(100));
insert into student values (null,'piya'),(null,'parth');
select * from student;
insert into student values (null,'mangesh');
insert into student (student_name) values ('sneha');
select * from student order by student_name;
--------------------------------------------------------------------------
```