Section A: Supplier Table (Oracle Version)

```
CREATE TABLE Supplier (
   Sup_No VARCHAR2(5) PRIMARY KEY,
   Sup_Name VARCHAR2(50),
   Item_Supplied VARCHAR2(50),
   Item_Price NUMBER,
   City VARCHAR2(50)
);

INSERT ALL
   INTO Supplier VALUES ('S1', 'Suresh', 'Keyboard', 400, 'Hyderabad')
   INTO Supplier VALUES ('S2', 'Kiran', 'Processor', 8000, 'Delhi')
   INTO Supplier VALUES ('S3', 'Mohan', 'Mouse', 350, 'Delhi')
   INTO Supplier VALUES ('S4', 'Ramesh', 'Processor', 9000, 'Bangalore')
   INTO Supplier VALUES ('S5', 'Manish', 'Printer', 6000, 'Mumbai')
   INTO Supplier VALUES ('S6', 'Srikanth', 'Processor', 8500, 'Chennai')
SELECT * FROM dual;


-- Q1
SELECT Sup_No, Sup_Name
FROM Supplier
WHERE Sup_Name LIKE 'R%';

-- Q2
SELECT Sup_Name
FROM Supplier
WHERE Item_Supplied = 'Processor' AND City = 'Delhi';

-- Q3
SELECT Sup_Name
FROM Supplier
WHERE Item_Supplied = (
   SELECT Item_Supplied
   FROM Supplier
   WHERE Sup_Name = 'Ramesh'
) AND Sup_Name <> 'Ramesh';

-- Q4
UPDATE Supplier
SET Item_Price = Item_Price + 200
WHERE Item_Supplied = 'Keyboard';

-- Q5
SELECT Sup_No, Sup_Name, Item_Price
FROM Supplier
WHERE City = 'Delhi'
```

```
ORDER BY Item_Price;

-- Q6
ALTER TABLE Supplier
ADD ContactNo VARCHAR2(15);

-- Q7
DELETE FROM Supplier
WHERE Item_Price = (
    SELECT MIN(Item_Price) FROM Supplier
);

-- Q8
CREATE OR REPLACE VIEW SupplierView AS
SELECT Sup_No, Sup_Name FROM Supplier;

-- Q9
SELECT * FROM Supplier
ORDER BY Item_Supplied, Item_Price DESC;

-- Q10
SELECT * FROM Supplier
WHERE Item_Supplied NOT IN ('Processor', 'Keyboard');
```

Section B : EmpDetails Table

```
-- Create EmpDetails table
CREATE TABLE EmpDetails (
    Eid VARCHAR2(10) PRIMARY KEY,
    Ename VARCHAR2(50),
    DOB DATE,
    Designation VARCHAR2(30),
    Salary NUMBER,
    DOJ DATE
);

-- Insert dummy data into EmpDetails table
INSERT INTO EmpDetails VALUES ('E001', 'Alice', TO_DATE('1985-05-15',
'YYYY-MM-DD'), 'Programmer', 60000, TO_DATE('2012-01-15', 'YYYY-MM-DD'));
INSERT INTO EmpDetails VALUES ('E002', 'Bob', TO_DATE('1990-03-10', 'YYYY-MM-DD'),
'DBA', 70000, TO_DATE('2015-06-23', 'YYYY-MM-DD'));
INSERT INTO EmpDetails VALUES ('E003', 'Charlie', TO_DATE('1988-07-25',
'YYYY-MM-DD'), 'Programmer', 55000, TO_DATE('2014-04-30', 'YYYY-MM-DD'));
INSERT INTO EmpDetails VALUES ('E004', 'David', TO_DATE('1992-12-30',
'YYYY-MM-DD'), 'Analyst', 48000, TO_DATE('2018-02-20', 'YYYY-MM-DD'));
```

```
INSERT INTO EmpDetails VALUES ('E005', 'Eva', TO_DATE('1991-09-05', 'YYYY-MM-DD'),
'Programmer', 65000, TO_DATE('2013-08-01', 'YYYY-MM-DD'));
INSERT INTO EmpDetails VALUES ('E006', 'Fiona', TO_DATE('1987-11-10',
'YYYY-MM-DD'), 'DBA', 75000, TO_DATE('2017-04-18', 'YYYY-MM-DD'));
INSERT INTO EmpDetails VALUES ('E007', 'George', TO_DATE('1993-02-14',
'YYYY-MM-DD'), 'Manager', 90000, TO_DATE('2019-09-23', 'YYYY-MM-DD'));
INSERT INTO EmpDetails VALUES ('E008', 'Hannah', TO_DATE('1989-06-22',
'YYYY-MM-DD'), 'Clerk', 30000, TO_DATE('2016-11-05', 'YYYY-MM-DD'));

-- Q11
SELECT * FROM EmpDetails
WHERE Designation = 'Programmer';

-- Q12
SELECT * FROM EmpDetails
WHERE DOJ > TO_DATE('31-12-2014', 'DD-MM-YYYY');

-- Q13
SELECT * FROM EmpDetails
WHERE Ename LIKE '%a';

-- Q14
SELECT SUM(Salary) AS Total_Programmer_Salary
FROM EmpDetails
WHERE Designation = 'Programmer';

-- Q15
SELECT UPPER(Ename) AS Name_In_UpperCase
FROM EmpDetails;

-- Q16
SELECT * FROM EmpDetails
WHERE DOJ = (SELECT MIN(DOJ) FROM EmpDetails);

-- Q17
SELECT * FROM EmpDetails
WHERE Ename LIKE '%ee%';

-- Q18
UPDATE EmpDetails
SET Salary = Salary + 5000
WHERE Designation = 'DBA';

-- Q19
SELECT * FROM EmpDetails
WHERE Salary > (SELECT AVG(Salary) FROM EmpDetails);

-- Q20
```

```
SELECT Ename || ' is working as ' || Designation || ' with a Salary of Rs. ' || Salary AS Info
FROM EmpDetails;
```

Section C: Employee and Department Tables

```
-- Create Department table
CREATE TABLE Department (
    DeptId VARCHAR2(5) PRIMARY KEY,
    Dname VARCHAR2(50) NOT NULL
);

-- Create Employee table
CREATE TABLE Employee (
    Eid NUMBER PRIMARY KEY,
    Ename VARCHAR2(50),
    DeptId VARCHAR2(5),
    Designation VARCHAR2(30),
    Salary NUMBER CHECK (Salary > 10000),
    DOJ DATE,
    FOREIGN KEY (DeptId) REFERENCES Department(DeptId)
);

-- Insert data into Department
INSERT INTO Department VALUES ('D1', 'HR');
INSERT INTO Department VALUES ('D2', 'IT');
INSERT INTO Department VALUES ('D3', 'Finance');

-- Insert data into Employee
INSERT INTO Employee VALUES (101, 'Alice', 'D1', 'Manager', 55000,
TO_DATE('2010-06-15', 'YYYY-MM-DD'));
INSERT INTO Employee VALUES (102, 'Bob', 'D2', 'Clerk', 25000, TO_DATE('2011-02-20',
'YYYY-MM-DD'));
INSERT INTO Employee VALUES (103, 'Charlie', 'D3', 'Analyst', 35000,
TO_DATE('2015-08-01', 'YYYY-MM-DD'));
INSERT INTO Employee VALUES (104, 'David', 'D2', 'Programmer', 40000,
TO_DATE('2012-02-11', 'YYYY-MM-DD'));
INSERT INTO Employee VALUES (105, 'Eva', 'D1', 'Clerk', 30000, TO_DATE('2018-09-10',
'YYYY-MM-DD'));
INSERT INTO Employee VALUES (106, 'Fiona', 'D3', 'Manager', 60000,
TO_DATE('2011-11-30', 'YYYY-MM-DD'));
INSERT INTO Employee VALUES (107, 'George', 'D2', 'DBA', 45000,
TO_DATE('2020-03-05', 'YYYY-MM-DD'));

-- Oracle Queries
```

```
-- Q21
SELECT * FROM Employee
WHERE Salary > (SELECT AVG(Salary) FROM Employee);

-- Q22
SELECT E.Eid, E.Ename, D.Dname
FROM Employee E
JOIN Department D ON E.DeptId = D.DeptId;

-- Q23
SELECT * FROM Employee
ORDER BY Salary DESC;

-- Q24
SELECT DISTINCT Designation FROM Employee;

-- Q25
SELECT * FROM Employee
ORDER BY DeptId, Salary ASC;

-- Q26
SELECT * FROM Employee
WHERE Designation = 'Clerk' AND DeptId = 'D2';

-- Q27
SELECT * FROM Employee
WHERE EXTRACT(YEAR FROM DOJ) = 2011;

-- Q28
SELECT * FROM Employee
WHERE EXTRACT(MONTH FROM DOJ) = 2;

-- Q29
SELECT * FROM Employee
WHERE Salary BETWEEN 30000 AND 45000;

-- Q30
SELECT *, FLOOR(MONTHS_BETWEEN(SYSDATE, DOJ)/12) AS Experience_Years
FROM Employee;
```

Section D: Student Table

```
-- Create Student table
CREATE TABLE Student (
    Sid NUMBER PRIMARY KEY,
```

```
    Sname VARCHAR2(50),
    DOB DATE,
    State VARCHAR2(50),
    Gender CHAR(1),
    Category VARCHAR2(30),
    Course VARCHAR2(30)
);

-- Insert dummy data into Student table
INSERT INTO Student VALUES (1, 'Alice', TO_DATE('2000-05-15', 'YYYY-MM-DD'),
'Telangana', 'F', 'OBC', 'Comp');
INSERT INTO Student VALUES (2, 'Bob', TO_DATE('2001-08-20', 'YYYY-MM-DD'),
'AndhraPradesh', 'M', 'General', 'Electronics');
INSERT INTO Student VALUES (3, 'Charlie', TO_DATE('1999-03-25', 'YYYY-MM-DD'), 'Tamil
Nadu', 'M', 'SC', 'Electrical');
INSERT INTO Student VALUES (4, 'David', TO_DATE('2000-11-10', 'YYYY-MM-DD'),
'Karnataka', 'M', 'General', 'Mechanical');
INSERT INTO Student VALUES (5, 'Eva', TO_DATE('2002-07-05', 'YYYY-MM-DD'), 'Kerala',
'F', 'OBC', 'Comp');
INSERT INTO Student VALUES (6, 'Frank', TO_DATE('1998-12-15', 'YYYY-MM-DD'),
'Telangana', 'M', 'OBC', 'Civil');
INSERT INTO Student VALUES (7, 'Grace', TO_DATE('2000-06-30', 'YYYY-MM-DD'),
'Maharashtra', 'F', 'General', 'Biotechnology');
INSERT INTO Student VALUES (8, 'Hannah', TO_DATE('2001-04-11', 'YYYY-MM-DD'),
'Tamil Nadu', 'F', 'ST', 'Chemical');

-- Q31
SELECT * FROM Student
WHERE State NOT IN ('Telangana', 'AndhraPradesh');

-- Q32
CREATE OR REPLACE VIEW Telangana_Students AS
SELECT Sid, Sname FROM Student
WHERE State = 'Telangana';

-- Q33
CREATE INDEX idx_sname ON Student (Sname);

-- Q34
SELECT * FROM Student
WHERE Gender = 'F' AND Course = 'Comp' AND Category = 'OBC';

-- Q35
SELECT Sid, Sname, FLOOR(MONTHS_BETWEEN(SYSDATE, DOB)/12) AS Age
FROM Student;

-- Q36
SELECT * FROM Student
```

```
ORDER BY Course, Sname;

-- Q37
DELETE FROM Student
WHERE Course = 'Comp' AND DOB > TO_DATE('31-12-2002', 'DD-MM-YYYY');

-- Q38
ALTER TABLE Student
ADD (ContactNo VARCHAR2(15), Email VARCHAR2(50));

-- Q39
SELECT
   CASE
      WHEN Gender = 'F' THEN 'Ms. ' || Sname
      ELSE 'Mr. ' || Sname
   END AS Prefixed_Name
FROM Student;

-- Q40
SELECT * FROM Student
WHERE LENGTH(Sname) = 5;
```

Section E: Library Table

```
-- Create Library table
CREATE TABLE Library (
   BookId VARCHAR2(10) PRIMARY KEY,
   BookName VARCHAR2(100) NOT NULL,
   Author VARCHAR2(50),
   DatePurchased DATE,
   Publisher VARCHAR2(50),
   Price NUMBER
);

-- Insert dummy data into Library table
INSERT INTO Library VALUES ('B001', 'The Art of Programming', 'John Sharma',
TO_DATE('2010-05-20', 'YYYY-MM-DD'), 'Himalaya', 600);
INSERT INTO Library VALUES ('B002', 'Database Design Basics', 'Anil Kumar',
TO_DATE('2012-08-14', 'YYYY-MM-DD'), 'Kalyani', 750);
INSERT INTO Library VALUES ('B003', 'Advanced Java', 'Rajesh Sharma',
TO_DATE('2015-10-11', 'YYYY-MM-DD'), 'Himalaya', 650);
INSERT INTO Library VALUES ('B004', 'Data Structures and Algorithms', 'Sunil Sharma',
TO_DATE('2018-02-19', 'YYYY-MM-DD'), 'Pearson', 900);
INSERT INTO Library VALUES ('B005', 'Web Development with React', 'Ravi Shankar',
TO_DATE('2019-06-25', 'YYYY-MM-DD'), 'McGraw Hill', 1200);
```

```
INSERT INTO Library VALUES ('B006', 'Operating Systems Concepts', 'Deepak Sharma',
TO_DATE('2016-11-13', 'YYYY-MM-DD'), 'Kalyani', 500);
INSERT INTO Library VALUES ('B007', 'Machine Learning Fundamentals', 'Suresh Kumar',
TO_DATE('2021-03-07', 'YYYY-MM-DD'), 'McGraw Hill', 950);
INSERT INTO Library VALUES ('B008', 'Artificial Intelligence', 'Vikram Sharma',
TO_DATE('2017-09-22', 'YYYY-MM-DD'), 'Himalaya', 550);

-- Q41
SELECT Author FROM Library
WHERE Publisher = 'Himalaya';

-- Q42
SELECT Publisher, SUM(Price) AS Total_Cost
FROM Library
GROUP BY Publisher;

-- Q43
SELECT COUNT(*) AS Total_Kalyani_Books
FROM Library
WHERE Publisher = 'Kalyani';

-- Q44
ALTER TABLE Library
RENAME COLUMN Publisher TO Publications;

-- Q45
SELECT * FROM Library
ORDER BY DatePurchased;

-- Q46
CREATE INDEX idx_book_author
ON Library (BookName, Author);

-- Q47
SELECT * FROM Library
WHERE Price BETWEEN 500 AND 700;

-- Q48
UPDATE Library
SET Price = Price + 200
WHERE Publications NOT IN ('Himalaya', 'Kalyani');

-- Q49
SELECT * FROM Library
WHERE Author LIKE '%Sharma%';

-- Q50
CREATE OR REPLACE VIEW HimalayaBooks AS
```

```
SELECT BookId, BookName
FROM Library
WHERE Publications = 'Himalaya';
```

```
-- Create table
CREATE TABLE Student1 (
    StudentNO VARCHAR2(10) PRIMARY KEY,
    StudentName VARCHAR2(50),
    DBMS_Marks NUMBER(3),
    ECommerce_Marks NUMBER(3),
    FIT_Marks NUMBER(3),
    WebProgramming_Marks NUMBER(3)
);

-- (a) Insert Five Records
INSERT INTO Student1 (StudentNO, StudentName, DBMS_Marks, ECommerce_Marks,
FIT_Marks, WebProgramming_Marks) VALUES
('S001', 'Amit Kumar', 75, 80, 65, 70);
INSERT INTO Student1 (StudentNO, StudentName, DBMS_Marks, ECommerce_Marks,
FIT_Marks, WebProgramming_Marks) VALUES
('S002', 'Priya Sharma', 85, 90, 75, 88);
INSERT INTO Student1 (StudentNO, StudentName, DBMS_Marks, ECommerce_Marks,
FIT_Marks, WebProgramming_Marks) VALUES
('S003', 'Rahul Verma', 60, 55, 50, 62);
INSERT INTO Student1 (StudentNO, StudentName, DBMS_Marks, ECommerce_Marks,
FIT_Marks, WebProgramming_Marks) VALUES
('S004', 'Sneha Reddy', 92, 88, 95, 90);
INSERT INTO Student1 (StudentNO, StudentName, DBMS_Marks, ECommerce_Marks,
FIT_Marks, WebProgramming_Marks) VALUES
('S005', 'Vikram Singh', 70, 72, 68, 75);

-- (b) calculate Total for Marks field
-- Add a column for Total_Marks
ALTER TABLE Student1 ADD (Total_Marks NUMBER(4));

-- Calculate and update Total_Marks
UPDATE Student1
SET Total_Marks = DBMS_Marks + ECommerce_Marks + FIT_Marks +
WebProgramming_Marks;

-- To verify:
SELECT * FROM Student1;
```

```
-- Create table
CREATE TABLE Student2 (
    StudentNO VARCHAR2(10) PRIMARY KEY,
    StudentName VARCHAR2(50),
    BLaw_Marks NUMBER(3),
    CostAccts_Marks NUMBER(3),
    CorpAccts_Marks NUMBER(3),
    WebProgramming_Marks NUMBER(3)
);

-- (a) Insert Five Records
INSERT INTO Student2 (StudentNO, StudentName, BLaw_Marks, CostAccts_Marks,
CorpAccts_Marks, WebProgramming_Marks) VALUES
('S101', 'Anjali Mehta', 65, 70, 75, 80);
INSERT INTO Student2 (StudentNO, StudentName, BLaw_Marks, CostAccts_Marks,
CorpAccts_Marks, WebProgramming_Marks) VALUES
('S102', 'Rohan Desai', 80, 85, 90, 78);
INSERT INTO Student2 (StudentNO, StudentName, BLaw_Marks, CostAccts_Marks,
CorpAccts_Marks, WebProgramming_Marks) VALUES
('S103', 'Kavita Nair', 55, 60, 45, 50);
INSERT INTO Student2 (StudentNO, StudentName, BLaw_Marks, CostAccts_Marks,
CorpAccts_Marks, WebProgramming_Marks) VALUES
('S104', 'Suresh Iyer', 90, 92, 88, 95);
INSERT INTO Student2 (StudentNO, StudentName, BLaw_Marks, CostAccts_Marks,
CorpAccts_Marks, WebProgramming_Marks) VALUES
('S105', 'Deepa Rao', 72, 68, 70, 75);

-- (b) calculate Total for Marks field
ALTER TABLE Student2 ADD (Total_Marks NUMBER(4));
UPDATE Student2
SET Total_Marks = BLaw_Marks + CostAccts_Marks + CorpAccts_Marks +
WebProgramming_Marks;

-- (c) Calculate Average of Marks
ALTER TABLE Student2 ADD (Average_Marks NUMBER(5,2));
UPDATE Student2
SET Average_Marks = (BLaw_Marks + CostAccts_Marks + CorpAccts_Marks +
WebProgramming_Marks) / 4.0;
-- Or using the Total_Marks column:
-- UPDATE Student2 SET Average_Marks = Total_Marks / 4.0;

-- To verify:
SELECT * FROM Student2;
-- Create table
CREATE TABLE Student3 (
    StudentNO VARCHAR2(10) PRIMARY KEY,
    StudentName VARCHAR2(50),
    StudentCollege VARCHAR2(100),
```

```
    University VARCHAR2(100),
    Address VARCHAR2(255),
    FirstYear_Marks NUMBER(4),  -- Assuming marks per year could be up to 9999
    SecondYear_Marks NUMBER(4),
    FinalYear_Marks NUMBER(4)
);

-- (a) Insert Five Records
INSERT INTO Student3 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S201', 'Raj Patel', 'City College', 'Osmania University', '123 Abids, Hyd', 650, 700, 750);
INSERT INTO Student3 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S202', 'Meena Kumari', 'Commerce College', 'Osmania University', '456 Koti, Hyd', 750,
800, 850);
INSERT INTO Student3 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S203', 'Arjun Reddy', 'Deccan College', 'Osmania University', '789 Begumpet, Hyd', 550,
600, 580);
INSERT INTO Student3 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S204', 'Lakshmi Devi', 'City College', 'Osmania University', '321 Secunderabad', 880, 900,
920);
INSERT INTO Student3 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S205', 'Imran Khan', 'Commerce College', 'Osmania University', '654 Charminar', 600, 650,
620);

-- (b) calculate Percentage for Marks field
-- Assuming maximum marks per year is 1000, so total maximum marks is 3000.
-- If total max marks are different, adjust the divisor.
ALTER TABLE Student3 ADD (Total_Academic_Marks NUMBER(5));
ALTER TABLE Student3 ADD (Percentage_Marks NUMBER(5,2));

-- Calculate Total_Academic_Marks
UPDATE Student3
SET Total_Academic_Marks = FirstYear_Marks + SecondYear_Marks + FinalYear_Marks;

-- Calculate Percentage_Marks
-- Assuming max marks for each year is 1000, total 3000
UPDATE Student3
SET Percentage_Marks = (Total_Academic_Marks / 3000.0) * 100;
-- Or, if each year's marks are already percentages (e.g. out of 100):
-- SET Percentage_Marks = (FirstYear_Marks + SecondYear_Marks + FinalYear_Marks) /
3.0;
-- The question is a bit ambiguous. I'll assume the marks are raw scores and need
conversion to percentage based on a total.
```

```
-- To verify:
SELECT * FROM Student3;
-- Create table (Similar to Student3, let's call it Student4)
CREATE TABLE Student4 (
    StudentNO VARCHAR2(10) PRIMARY KEY,
    StudentName VARCHAR2(50),
    StudentCollege VARCHAR2(100),
    University VARCHAR2(100),
    Address VARCHAR2(255),
    FirstYear_Marks NUMBER(4),
    SecondYear_Marks NUMBER(4),
    FinalYear_Marks NUMBER(4)
);

-- (a) Insert Five Records
INSERT INTO Student4 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S301', 'Ajay Ghose', 'Aurora College', 'Osmania University', '11 Ameerpet, Hyd', 450, 480,
400); -- Low marks
INSERT INTO Student4 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S302', 'Bhavna Rao', 'Nizam College', 'Osmania University', '22 Banjara Hills, Hyd', 920,
950, 930); -- High marks
INSERT INTO Student4 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S303', 'Chetan Kumar', 'Badruka College', 'Osmania University', '33 Dilsukhnagar, Hyd',
600, 650, 700); -- Mid marks
INSERT INTO Student4 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S304', 'Divya Sri', 'St. Anns College', 'Osmania University', '44 Mehdipatnam, Hyd', 350,
400, 380); -- Low marks
INSERT INTO Student4 (StudentNO, StudentName, StudentCollege, University, Address,
FirstYear_Marks, SecondYear_Marks, FinalYear_Marks) VALUES
('S305', 'Farhan Ali', 'Loyola Academy', 'Osmania University', '55 Alwal, Hyd', 880, 910, 940);
-- High marks

-- (b) calculate Percentage for Marks field
ALTER TABLE Student4 ADD (Total_Academic_Marks NUMBER(5));
ALTER TABLE Student4 ADD (Percentage_Marks NUMBER(5,2));

UPDATE Student4
SET Total_Academic_Marks = FirstYear_Marks + SecondYear_Marks + FinalYear_Marks;

-- Assuming max marks for each year is 1000, total 3000
UPDATE Student4
SET Percentage_Marks = (Total_Academic_Marks / 3000.0) * 100;

-- To verify the calculations so far:
```

```
SELECT * FROM Student4;

-- (c) List all the students who got less than 50% Marks.
SELECT StudentNO, StudentName, Percentage_Marks
FROM Student4
WHERE Percentage_Marks < 50;

-- (d) List all the students who got more than 90% Marks .
SELECT StudentNO, StudentName, Percentage_Marks
FROM Student4
WHERE Percentage_Marks > 90;

-- Create table
CREATE TABLE Student5 (
   StudentNO VARCHAR2(10) PRIMARY KEY,
   StudentName VARCHAR2(50),
   DBMS_Marks NUMBER(3),
   ECommerce_Marks NUMBER(3),
   FIT_Marks NUMBER(3),
   CLanguage_Marks NUMBER(3),
   WebProgramming_Marks NUMBER(3)
);

-- (a) Insert Five Records
INSERT INTO Student5 (StudentNO, StudentName, DBMS_Marks, ECommerce_Marks,
FIT_Marks, CLanguage_Marks, WebProgramming_Marks) VALUES
('S401', 'Gita Reddy', 70, 75, 60, 65, 72),
('S402', 'Hari Prasad', 40, 35, 45, 30, 42), -- Low average
('S403', 'Ishaan Khan', 80, 85, 90, 78, 88),
('S404', 'Jaya Singh', 30, 40, 35, 45, 38), -- Low average
('S405', 'Kiran Kumar', 65, 60, 70, 55, 68);

-- (b) calculate Total for Marks field
ALTER TABLE Student5 ADD (Total_Marks NUMBER(4));
UPDATE Student5
SET Total_Marks = DBMS_Marks + ECommerce_Marks + FIT_Marks + CLanguage_Marks
+ WebProgramming_Marks;

-- (c) Calculate Average of Marks
ALTER TABLE Student5 ADD (Average_Marks NUMBER(5,2));
UPDATE Student5
SET Average_Marks = (DBMS_Marks + ECommerce_Marks + FIT_Marks +
CLanguage_Marks + WebProgramming_Marks) / 5.0;
-- Or: UPDATE Student5 SET Average_Marks = Total_Marks / 5.0;

-- To verify before update:
SELECT StudentNO, StudentName, DBMS_Marks, ECommerce_Marks, FIT_Marks,
CLanguage_Marks, WebProgramming_Marks, Average_Marks FROM Student5;
```

```
-- (d) Increase 10 marks for the students where average is < 50 .
-- This step is tricky. If you increase individual subject marks, the average and total need recalculation.
-- If the question means to increase the *existing* average by 10, it's simpler but might not be the intent.
-- Usually, it means increasing subject marks, implying a re-evaluation.
-- Let's assume it means increase each subject mark by 10 for those students.
-- However, the question only says "Increase 10 marks", not "to each subject".
-- A common interpretation in such RDBMS questions is to add to the Total_Marks or Average_Marks directly for simplicity if not specified otherwise.
-- But if the intent is to affect the source marks, it's more complex.

-- Option 1: Add 10 to the Average_Marks directly (less realistic but simple)
-- UPDATE Student5
-- SET Average_Marks = Average_Marks + 10
-- WHERE Average_Marks < 50;

-- Option 2: Add 10 to the Total_Marks (and then average would need recalc)
-- UPDATE Student5
-- SET Total_Marks = Total_Marks + 10
-- WHERE Average_Marks < 50;
-- Then, update average:
-- UPDATE Student5
-- SET Average_Marks = Total_Marks / 5.0
-- WHERE Average_Marks < 50; -- (or just update all averages again)

-- Option 3: Add 10 to *one* of the subjects (e.g., WebProgramming) for simplicity if subject not specified.
-- This is the most common interpretation if it doesn't say "each subject".
-- Let's assume we add 10 marks to *one* subject, e.g., WebProgramming, for students with avg < 50.
-- Or it could mean add 10 to the *overall total*. Given it affects the *average*, adding to the total and recalculating average is more logical.

-- Let's go with increasing Total_Marks by 10 and recalculating Average.
UPDATE Student5
SET Total_Marks = Total_Marks + 10
WHERE Average_Marks < 50;

-- Now, recalculate the average for those students (or all students for simplicity)
UPDATE Student5
SET Average_Marks = Total_Marks / 5.0;

-- To verify:
SELECT StudentNO, StudentName, DBMS_Marks, ECommerce_Marks, FIT_Marks, CLanguage_Marks, WebProgramming_Marks, Total_Marks, Average_Marks FROM Student5;
```

-- Note: Students S402 and S404 should show an increase in Total_Marks by 10 and their Average_Marks recalculated.
-- E.g., S402: Old Total (40+35+45+30+42 = 192), Old Avg (192/5 = 38.4).
-- New Total = 192 + 10 = 202. New Avg = 202/5 = 40.4.

**********Prime_Number(ERROR)**********

```
SET SERVEROUTPUT ON;
DECLARE
 num NUMBER:= 2;
 i NUMBER;
 limit_num NUMBER;
 is_prime BOOLEAN;

BEGIN
 limit_num := &Enter_limit;
 DBMS_OUTPUT.PUT_LINE('Prime Number upto ' || limit_num || ':');
 WHILE num <= limit_num LOOP
  is_prime := TRUE;
  FOR i IN 2 .. TRUNC(SQRT(num)) LOOP
   IF MOD(num,i) = 0 THEN
    is_prime := FALSE;
    EXIT;
   END IF;
  END LOOP;
  IF is_prime THEN
   DBMS_OUTPUT.PUT_LINE(num);
  END IF;
  num := num +1;
 END LOOP;
END;
/
```

*********Prime_Number(ERROR)*********

```
SET SERVEROUTPUT ON;
DECLARE
        i NUMBER;
        j NUMBER;
        limit NUMBER;
BEGIN
        limit := &Enter_last_value;
        i:=2;
        LOOP
                j:=2;

                LOOP
                        EXIT WHEN((MOD(i,j)=0) OR (i=j));
                        j := j+1;
                END LOOP;

                IF(i=j) THEN
                        DBMS_OUTPUT.PUT_LINE('The prime is '|| i);
                        END IF;
                i := i+i;
                EXIT WHEN i = limit;
        END LOOP;
END;
/
```

*********Fibonacci_Series**********

```
SET SERVEROUTPUT ON;
DECLARE
        n NUMBER := &n;
        a NUMBER := 0;
        b NUMBER := 1;
        c NUMBER;
        i NUMBER;
BEGIN
        DBMS_OUTPUT.PUT_LINE(a);
        DBMS_OUTPUT.PUT_LINE(b);
```

```
        FOR i IN 3..n LOOP
                c:=a+b;
                DBMS_OUTPUT.PUT_LINE(c);
                a := b;
                b :=c;
        END LOOP;
END;
/
```

**********Procedures**********

******1******

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE Procedure disp
AS
        b VARCHAR := 'Hello World';
BEGIN
        DBMS_OUTPUT.PUT_LINE('My Procedure is '|| b);
END disp;
/
```

*******2*****

```
SET SERVEROUTPUT ON;
DECLARE
        a NUMBER := &a;
        PROCEDURE SQ_NUM(n NUMBER)
        IS
                c NUMBER(3);
        BEGIN
                c := b*b;
                DBMS_OUTPUT.PUT_LINE('Square of '||b||'is : '||c);
        END SQ_NUM;
BEGIN
        SQ_NUM(a);
END;
/
```

********3******

```
SET SERVEROUTPUT ON;
DECLARE
        d NUMBER(3);
        a NUMBER := &a;
        PROCEDURE SQ_NUM(b IN NUMBER, c OUT NUMBER)
        IS

        BEGIN
                c := b*b;
        END SQ_NUM;
BEGIN
        SQ_NUM(a,d);
        DBMS_OUTPUT.PUT_LINE('Square of '||a||' is '||d);
END;
/
```

```
********4******
SET SERVEROUTPUT ON;
DECLARE
        d NUMBER(3);
        a NUMBER := &a;
        PROCEDURE SQ_NUM(b IN OUT NUMBER)
        IS

        BEGIN
                b := b*b;
        END SQ_NUM;
BEGIN
        d :=a;
        SQ_NUM(a);
        DBMS_OUTPUT.PUT_LINE('Square of '||d||' is '||a);
END;
/
```

```
********5(Prime_Number(Procedure))********


SET SERVEROUTPUT ON;
CREATE OR REPLACE Procedure prim(a NUMBER,b NUMBER)
AS
```

```
        i NUMBER;
        j NUMBER;
        is_prime BOOLEAN;
BEGIN
        FOR i IN a..b LOOP
                is_prime := TRUE;
                FOR j IN 2..TRUNC(SQRT(i)) LOOP
                        IF MOD(i,j)=0 THEN
                                is_prime := FALSE;
                                EXIT;
                        END IF;
                END LOOP;
                IF is_prime THEN
                        DBMS_OUTPUT.PUT_LINE(i);
                END IF;
        END LOOP;
END prim;
/
```

********6(Max_Of_Three(Procedure))********

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE Procedure max_three(a NUMBER,b NUMBER,c NUMBER)
AS
        max_val NUMBER;
BEGIN
        max_val := GREATEST(a,b,c);
        DBMS_OUTPUT.PUT_LINE(max_val);
END max_three;
/
```

********7(Cube(Function))********

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE FUNCTION cube_num(a NUMBER)
RETURN NUMBER
```

```
AS
	c NUMBER(3);
BEGIN
	c := a*a*a;
	RETURN c;
	DBMS_OUTPUT.PUT_LINE(c);
END cube_num;
/
```

********8(Cube(Function_Call))********

```
DECLARE
	a NUMBER := &a;
	b NUMBER;
BEGIN
	b:=cube_num(a);
	DBMS_OUTPUT.PUT_LINE(b);
END;
/
```

********9(Prime_Number(Function))********

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE FUNCTION prime_num(a NUMBER, b NUMBER)
RETURN NUMBER
AS
	i NUMBER;
	j NUMBER;
	is_prime BOOLEAN;
BEGIN
	FOR i IN a..b LOOP
		is_prime := TRUE;
		FOR j IN 2..TRUNC(SQRT(i)) LOOP
			IF MOD(i,j)=0 THEN
				is_prime := FALSE;
				EXIT;
			END IF;
		END LOOP;
		IF is_prime THEN
			DBMS_OUTPUT.PUT_LINE(i);
```

```
                END IF;
        END LOOP;
END prime_num;
/
```

********10(Prime_Number(Function_Call))********

```
DECLARE
        a NUMBER := &a;
        b NUMBER := &b;
        c NUMBER;
BEGIN
        c:=prime_num(a,b);
        DBMS_OUTPUT.PUT_LINE(c);
END;
/
```

********11(Fibonacci_Series(Function))********

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE FUNCTION fibo(n NUMBER)
RETURN NUMBER
AS
        a NUMBER := 0;
        b NUMBER := 1;
        c NUMBER;
        i NUMBER;
BEGIN
        DBMS_OUTPUT.PUT_LINE(a);
        DBMS_OUTPUT.PUT_LINE(b);

        FOR i IN 3..n LOOP
                c:=a+b;
                DBMS_OUTPUT.PUT_LINE(c);
                a := b;
                b :=c;
        END LOOP;
END fibo;
```

```
/
```

********12(Fibonacci_Series(Function_Call))********

```
DECLARE
        n NUMBER := &n;
        a NUMBER := 0;
        b NUMBER := 1;
        c NUMBER;
        i NUMBER;
BEGIN
        c:=fibo(n);
        DBMS_OUTPUT.PUT_LINE(c);
END;
/
```

**********13(Greatest_of_three(JoyDeep sir))**********

```
SET SERVEROUTPUT ON;
declare
   d number(5);
   a number:=&a;
   b number:=&b;
   c number:=&c;

   function max_num(x number,y number,z number)
   return number
   is
      max_val number(5);
   begin
      if x>=y and x>=z then
            max_val:=x;
      elsif y>=x and y>=z then
            max_val:=y;
      else
         max_val:=z;
      end if;
      return max_val;
   end max_num;
```

```
begin
   d:=max_num(a,b,c);
   dbms_output.put_line(d);
end;
/
```

********14(Factorial(Function))********

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE FUNCTION fact(n NUMBER) RETURN NUMBER
IS
BEGIN

        IF n = 0 THEN
                RETURN 1;
        ELSIF n<0 THEN
                RETURN 404;
        ELSE
                RETURN n * fact(n-1);
        END IF;
END fact;
/
```

********15(Factorial(Function_Call))********

```
DECLARE
        n NUMBER := &n;
        a NUMBER;

BEGIN
        a := fact(n);
        DBMS_OUTPUT.PUT_LINE(a);
END;
/
```

IMPLICIT CURSOR

QUE : using cursor, display information of an employee from emp table where e_id is provided by user ?

```
     E_ID ENAME                 AGE     SALARY
---------- -------------------- ---------- ----------
         1 Akansha               21      20000
         2 Pratiksha             21      20000
         3 Prakash               23      10000
```

```
set serveroutput on;
DECLARE
          a emp.e_id%type := &a;
          b emp.ename%type;
          c emp.age%type;
          d emp.salary%type;
BEGIN
          select e_id, ename, age, salary into a,b,c,d from emp where e_id = a;
          dbms_output.put_line(a || ' ' || b ||' ' || c || ' '|| d );
END;
/
```

QUE : update the salary of those employees whose salary is greater than 15000 using cursor ?

```
     E_ID ENAME                 AGE     SALARY
---------- -------------------- ---------- ----------
         1 Akansha               21      20000
         2 Pratiksha             21      20000
         3 Prakash               23      10000
```

```
set serveroutput on;
DECLARE
          a number;
BEGIN
          update emp set salary = salary + salary * 0.1 where salary > 15000;
          a := sql%rowcount;
          dbms_output.put_line(a || ' rows affected.');
END;
/
```

```
     E_ID ENAME                 AGE     SALARY
---------- -------------------- ---------- ----------
         1 Akansha               21      22000
         2 Pratiksha             21      22000
```

3 Prakash 23 10000

EXPLICIT CURSOR

QUE : DispLay the employees whose salry is gerater than 15000 using cursor (Explicit cursor will be used as multiple rows to be displayed) ?

```
DECLARE
        a emp.e_id%type;
        b emp.ename%type;
        c emp.age%type;
        d emp.salary%type;
        cursor cur_emp is
                select e_id, ename, age, salary from emp where salary > 15000;
BEGIN
        open cur_emp;
        loop
                fetch cur_emp into a,b,c,d;
                exit when cur_emp%notfound;
                dbms_output.put_line('ID ' ||a||' Name ' || b || ' Age ' || c || ' Salary ' || d);
        END loop;
        close cur_emp;
END;
/
```

QUE : Using cursor calculate and display tax(30 %) of each employee and also display the employe information ?

```
DECLARE
        a emp.e_id%type;
        b emp.ename%type;
        c emp.age%type;
        d emp.salary%type;
        t number;
        cursor cur_emp is
                select e_id, ename, age, salary from emp;
BEGIN
        open cur_emp;
        loop
                fetch cur_emp into a,b,c,d;
                exit when cur_emp%notfound;
                t := d * 0.3;
```

```
                dbms_output.put_line('ID ' ||a||' Name ' || b || ' Age ' || c || ' Salary ' || d || ' Tax =
' || t);
        END loop;
        close cur_emp;
END;
/
```

QUE : Using cursor calculate tax(30 %) and dispaly the salary of each employee after tax deduction and also display the employe information ?

```
DECLARE
        a emp.e_id%type;
        b emp.ename%type;
        c emp.age%type;
        d emp.salary%type;
        t number;
        cursor cur_emp is
                select e_id, ename, age, salary from emp for update;
BEGIN
        open cur_emp;
        loop
                fetch cur_emp into a,b,c,d;
                exit when cur_emp%notfound;
                dbms_output.put_line('ID ' ||a||' Name ' || b || ' Age ' || c || ' Salary ' || d );
                t := d * 0.3;
                d := d - t;
                update emp set salary = d where current of cur_emp;
                dbms_output.put_line('ID ' ||a||' Name ' || b || ' Age ' || c || ' New Salary ' || d || '
Tax = ' || t);
        END loop;
        close cur_emp;
END;
/
```

QUE : Insert into a table newemp , the records of all employees. Also display the employee information on the screen using cursor ?

(to create a carbon copy of a table, if you do not want the data inside table then use where 1 = 2)

```
create table newemp as select * from emp where 1=2
```

```
DECLARE
        a emp.e_id%type;
        b emp.ename%type;
```

```
        c emp.age%type;
        d emp.salary%type;
        cursor cur_emp is
                select e_id, ename, age, salary from emp for update;
BEGIN
        open cur_emp;
        loop
                fetch cur_emp into a,b,c,d;
                exit when cur_emp%notfound;
                insert into newemp values(a,b,c,d);
                dbms_output.put_line('ID ' ||a||' Name ' || b || ' Age ' || c || ' Salary ' || d );
        END loop;
        close cur_emp;
END;
/
```

QUE : update the emp table by increasing the salary by 10 % for the employees where salary is less than average salary ?

```
DECLARE
        avg_sal number;
BEGIN
        select avg(salary) into avg_sal from emp;
        update emp set salary = salary + salary * 0.1 where salary < avg_sal;
        dbms_output.put_line(sql%rowcount ||' rows affected');
END;
/
```

QUE :

create table supplier (sid number primary key,

```
DECLARE
        cursor cur_col is select p.color, c.cost from part p, catalog c where p.pid = c.pid for
update of p.color;
BEGIN
        open cur_col;
        loop
                fetch cur.col into a,b;
                exit when cur_col%notfound
                if b<2000
                update parts set color='Green' where current of cur_col;
                end if;
```

```
        END loop;
        close cur_col
END;
/
```

QUE : Cursor, for loop

```
FOR record_name in cursor.name
LOOP
        statement;
END LOOP;
```

QUE : ?

```
     E_ID ENAME                    AGE     SALARY
---------- -------------------- ---------- ----------
         1 Akansha                    21      20000
         2 Pratiksha                  21      20000
         3 Prakash                    23      10000
```

```
set serveroutput on;
DECLARE
        a emp%rowtype;
        b emp%type := &b;
BEGIN
        select * into a where e_id = b;
```

```
        dbms_output.put_line('ID '|| a.e_id);
END;
/
```

QUE : loop is only required for explicit cursors

```
DECLARE
        cursor c u
                select e_name, salary from emp;
BEGIN
        FOR a in cursor
        LOOP
                dbms_output.put_line('Namw ' || a.name || ' Salary ' || a.salary);
        END LOOP;
END;
/
```

QUE :

. In order to select or Update an explicit cursor, we have to declare the cursor as for update.

```
.eg             cursor c is
                        select * from emp for update [of salary];
```

. Then to delete or update, we need to use WHERE CURRENT OF cursor_name;

```
.       DELETE from emp WHERE CURRENT OF c;
        Update em set salary = salary+salary*0.1 WHERE CURRENT OF c;
```

QUE : Exception

```
DECLARE
        a emp%rowtype;
        b emp%type := &b;
BEGIN
        select * into a from emp where id = b;
        dbms_output.put_line('ID' || a.id ||'Name' || a.e_name ||'SALARY' || a.salary);
EXCEPTION
        when NO_DATA_FOUND THEN
                dbms_output.put_line('No such Employee');
```

```
        WHEN OTHERS Then
                dbms_output.put_line('ERROR');
END;
/
```

QUE : Exception (SYSTEM DEFINES)

NO DATA_FOUND
        It is raised when select into statement return no rows
INVALID_CURSOR
        It is raised when attemps are made to make a cursor operation that is not allowed, Such as closing an un equal cursor.
ROWTYPE_MISMATCH
        It is raisedwhen a cursor fetches value in a variable having incompatible datatyoe.
TOO_MANY_ROWS
        It is raised when select into statemeny retrurns more than one rows.

QUE : Exception

```
DECLARE
        a emp%rowtype;
        b emp%type := &b;
        e Exception
        if b <= 0 Then raise e;
        else
                select * into a from emp where id = b;
                dbms_output.put_line('ID' || a.id ||'Name' || a.e_name ||'SALARY' || a.salary);
EXCEPTION
        when NO_DATA_FOUND THEN
                dbms_output.put_line('No such Employee');
        WHEN OTHERS Then
                dbms_output.put_line('ERROR');
                WHEN e THEN
                        dbms_output.put_line('Id cannot be zero or less');
END;
/
```

QUE : Consider the following relations

Student (snum, sname, major, level, age)

Class (cname, meets_at:date, room)

Enrolled(anum, cname)

a) Write pl/ssql code using cursor to update major to "Adv. DB" for those who are enrolled in the "RDBS" course.

```
CREATE TABLE Student (
   snum NUMBER PRIMARY KEY,
   sname VARCHAR2(50),
   major VARCHAR2(50),
   level VARCHAR2(10),
   age NUMBER
);

CREATE TABLE Class (
   cname VARCHAR2(50) PRIMARY KEY,
   meets_at DATE,
   room VARCHAR2(20)
);

CREATE TABLE Enrolled (
   anum NUMBER,          -- assuming this refers to Student.snum
   cname VARCHAR2(50),   -- referring to Class.cname
   CONSTRAINT fk_student FOREIGN KEY (anum) REFERENCES Student(snum),
   CONSTRAINT fk_class FOREIGN KEY (cname) REFERENCES Class(cname)
);

-- Insert Students
INSERT INTO Student (snum, sname, major, level, age) VALUES (101, 'Alice', 'CS', 'Junior',
20);
INSERT INTO Student (snum, sname, major, level, age) VALUES (102, 'Bob', 'Math',
'Senior', 22);
INSERT INTO Student (snum, sname, major, level, age) VALUES (103, 'Charlie', 'Physics',
'Sophomore', 19);
INSERT INTO Student (snum, sname, major, level, age) VALUES (104, 'Diana', 'CS',
'Senior', 23);

-- Insert Classes
INSERT INTO Class (cname, meets_at, room) VALUES ('RDBS', TO_DATE('2025-06-01
10:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Room101');
INSERT INTO Class (cname, meets_at, room) VALUES ('Math101', TO_DATE('2025-06-02
09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Room102');
INSERT INTO Class (cname, meets_at, room) VALUES ('Physics202',
TO_DATE('2025-06-03 14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Room103');
```

```
-- Insert Enrollments
INSERT INTO Enrolled (anum, cname) VALUES (101, 'RDBS');
INSERT INTO Enrolled (anum, cname) VALUES (102, 'Math101');
INSERT INTO Enrolled (anum, cname) VALUES (103, 'Physics202');
INSERT INTO Enrolled (anum, cname) VALUES (104, 'RDBS');

DECLARE
  CURSOR c_students IS
    SELECT DISTINCT e.anum
    FROM Enrolled e
    WHERE e.cname = 'RDBS';

BEGIN
  FOR student_rec IN c_students LOOP
    UPDATE Student
    SET major = 'Adv. DB'
    WHERE snum = student_rec.anum;
  END LOOP;

  COMMIT;
END;
/
```

Q. student (snum, sname, major, level, age)
class (cname, meets_at : date, room)
enrolled (anum, cname)
write pl/sql code using cursor to uodate major to "adv db" for those who are enrolled in "rdbms" course

```
CREATE TABLE student (
    snum    NUMBER PRIMARY KEY,
    sname   VARCHAR2(50),
    major   VARCHAR2(50),
    level   VARCHAR2(10),
    age     NUMBER
);


CREATE TABLE class (
    cname     VARCHAR2(50) PRIMARY KEY,
    meets_at  DATE,
    room      VARCHAR2(20)
);


CREATE TABLE enrolled (
    anum   NUMBER,
```

```
    cname  VARCHAR2(50),
    FOREIGN KEY (anum) REFERENCES student(snum),
    FOREIGN KEY (cname) REFERENCES class(cname)
);


INSERT INTO student (snum, sname, major, level, age) VALUES (101, 'A', 'CS', 'JR', 20);
INSERT INTO student (snum, sname, major, level, age) VALUES (102, 'B', 'Math', 'SR', 22);
INSERT INTO student (snum, sname, major, level, age) VALUES (103, 'C', 'Physics', 'SR',
19);


INSERT INTO class (cname, meets_at, room) VALUES ('rdbms', TO_DATE('2025-06-20',
'YYYY-MM-DD'), 'R101');
INSERT INTO class (cname, meets_at, room) VALUES ('ai', TO_DATE('2025-06-21',
'YYYY-MM-DD'), 'R102');


INSERT INTO enrolled (anum, cname) VALUES (101, 'rdbms');
INSERT INTO enrolled (anum, cname) VALUES (102, 'ai');
INSERT INTO enrolled (anum, cname) VALUES (103, 'rdbms');

DECLARE

    CURSOR rdbms_students IS
        SELECT s.snum
        FROM student s
        JOIN enrolled e ON s.snum = e.anum
        WHERE e.cname = 'rdbms' for update of s.major;

BEGIN

    FOR stu_rec IN rdbms_students LOOP

        UPDATE student
        SET major = 'adv db1'
        WHERE  current of rdbms_students;
    END LOOP;


    COMMIT;
END;
```

Q. write a pl/sql code using cursor to update the level to sr for those who are entolled in RDBMS course

```
DECLARE
    CURSOR rdbms_students IS
```

```
        SELECT s.snum
        FROM student s
        JOIN enrolled e ON s.snum = e.anum
        WHERE e.came = 'rdbms' for update of s.level;

BEGIN
    FOR stu_rec IN rdbms_students LOOP
        UPDATE student
        SET level = 'SR'
        WHERE current of rdbms_students;

    END LOOP;


    COMMIT;



END;
/
```