**[4]**Consider the following Relations. It defines Library Database Schema

BOOK (Book\_ISBN [PK], Title[Not Null], Publisher\_ Name, Price[Check Price>0], Date\_Of\_Publication(Default SYSDATE), Book\_Copy)

BOOK\_AUTHORS (Book\_ISBN ,Author\_Name [PK],Author\_City[Default

“Pune‟], Phone[UNIQUE])

Design and Develop SQL DML statements for above database using MySQL

1. Create tables for above database with all constraints and Insert at least five records in each table.

**-- Create BOOK table**

**CREATE TABLE BOOK (**

**Book\_ISBN INT PRIMARY KEY,**

**Title VARCHAR(100) NOT NULL,**

**Publisher\_Name VARCHAR(100),**

**Price DECIMAL(8, 2) CHECK (Price > 0),**

**Date\_Of\_Publication DATE DEFAULT CURRENT\_DATE,**

**Book\_Copy INT**

**);**

**-- Create BOOK\_AUTHORS table**

**CREATE TABLE BOOK\_AUTHORS (**

**Book\_ISBN INT,**

**Author\_Name VARCHAR(100),**

**Author\_City VARCHAR(100) DEFAULT 'Pune',**

**Phone VARCHAR(20),**

**UNIQUE (Phone),**

**PRIMARY KEY (Book\_ISBN, Author\_Name),**

**FOREIGN KEY (Book\_ISBN) REFERENCES BOOK(Book\_ISBN)**

**);**

**-- Insert records into BOOK table**

**INSERT INTO BOOK (Book\_ISBN, Title, Publisher\_Name, Price, Book\_Copy) VALUES**

**(1, 'Book 1', 'Publisher 1', 10.50, 5),**

**(2, 'Book 2', 'Publisher 2', 15.75, 3),**

**(3, 'Book 3', 'Publisher 3', 20.00, 2),**

**(4, 'Book 4', 'Publisher 4', 12.50, 6),**

**(5, 'Book 5', 'Publisher 5', 18.50, 4);**

**-- Insert records into BOOK\_AUTHORS table**

**INSERT INTO BOOK\_AUTHORS (Book\_ISBN, Author\_Name, Phone) VALUES**

**(1, 'Author 1', '1234567890'),**

**(1, 'Author 2', '2345678901'),**

**(2, 'Author 3', '3456789012'),**

**(2, 'Author 4', '4567890123'),**

**(3, 'Author 5', '5678901234');**

2. Add field Author\_email to BOOK\_AUTHORS table.

ALTER TABLE BOOK\_AUTHORS

ADD COLUMN Author\_email VARCHAR(100);

3.Create a view on Book table as “DBMS BOOKS” by selecting all books titled DBMS.

CREATE VIEW `DBMS BOOKS` AS

SELECT \*

FROM BOOK

WHERE Title = 'DBMS';

4.Drop the view “DBMS BOOKS” just created

DROP VIEW IF EXISTS `DBMS BOOKS`;

**[3]**Consider the following Relations. It defines Library Database Schema

BOOK (Book\_ISBN [PK], Title[Not Null], Publisher\_ Name, Price[Check Price>0], Date\_Of\_Publication(Default SYSDATE), Book\_Copy)

BOOK\_AUTHORS (Book\_ISBN ,Author\_Name [PK],Author\_City[Default

“Pune‟], Phone[UNIQUE])

Design and Develop SQL DML statements for above database using MySQL

1. Create tables for above database with all constraints and Insert at least five records in each table.

-- Create BOOK table

CREATE TABLE BOOK (

Book\_ISBN INT PRIMARY KEY,

Title VARCHAR(100) NOT NULL,

Publisher\_Name VARCHAR(100),

Price DECIMAL(8, 2) CHECK (Price > 0),

Date\_Of\_Publication DATE DEFAULT CURRENT\_DATE,

Book\_Copy INT

);

-- Create BOOK\_AUTHORS table

CREATE TABLE BOOK\_AUTHORS (

Book\_ISBN INT,

Author\_Name VARCHAR(100),

Author\_City VARCHAR(100) DEFAULT 'Pune',

Phone VARCHAR(20),

UNIQUE (Phone),

PRIMARY KEY (Book\_ISBN, Author\_Name),

FOREIGN KEY (Book\_ISBN) REFERENCES BOOK(Book\_ISBN)

);

-- Insert records into BOOK table

INSERT INTO BOOK (Book\_ISBN, Title, Publisher\_Name, Price, Book\_Copy) VALUES

(1, 'Book 1', 'Publisher 1', 10.50, 5),

(2, 'Book 2', 'Publisher 2', 15.75, 3),

(3, 'Book 3', 'Publisher 3', 20.00, 12),

(4, 'Book 4', 'Publisher 4', 12.50, 6),

(5, 'Book 5', 'Publisher 5', 18.50, 14);

-- Insert records into BOOK\_AUTHORS table

INSERT INTO BOOK\_AUTHORS (Book\_ISBN, Author\_Name, Phone) VALUES

(1, 'Author 1', '1234567890'),

(1, 'Author 2', '2345678901'),

(2, 'Author 3', '3456789012'),

(2, 'Author 4', '4567890123'),

(3, 'Author 5', '5678901234');

1. Select Book Names from table Book whose copies are in between 10 to 15.

SELECT Title

FROM BOOK

WHERE Book\_Copy BETWEEN 10 AND 15;

1. Update Book Copies as “10” whose Book Publisher is “Tata MacGraw Hill”.

UPDATE BOOK

SET Book\_Copy = 10

WHERE Publisher\_Name = 'Tata MacGraw Hill';

1. Select the Name of Publisher who supplied maximum books.

SELECT Publisher\_Name

FROM BOOK

GROUP BY Publisher\_Name

HAVING COUNT(\*) = (

SELECT MAX(Book\_Count)

FROM (

SELECT Publisher\_Name, COUNT(\*) AS Book\_Count

FROM BOOK

GROUP BY Publisher\_Name

) AS Counts

);

1. Display name of publishers as per no of books published by them in ascending order.

SELECT Publisher\_Name, COUNT(\*) AS Book\_Count

FROM BOOK

GROUP BY Publisher\_Name

ORDER BY Book\_Count ASC;

1. Select details of Books whose Author lives in “Pune”.

SELECT b.\*

FROM BOOK b

INNER JOIN BOOK\_AUTHORS ba ON b.Book\_ISBN = ba.Book\_ISBN

WHERE ba.Author\_City = 'Pune';

**[2]** Design and develop below database and execute following SQL DML statements using

MySQL .

emp (eno, ename, bdate, title, salary, dno)

proj (pno, pname, budget, dno)

dept (dno, dname, mgreno)

workson (eno, pno, hours)

-- Create emp table

CREATE TABLE emp (

eno INT PRIMARY KEY,

ename VARCHAR(100),

bdate DATE,

title VARCHAR(100),

salary DECIMAL(8, 2),

dno INT

);

-- Create proj table

CREATE TABLE proj (

pno INT PRIMARY KEY,

pname VARCHAR(100),

budget DECIMAL(10, 2),

dno INT

);

-- Create dept table

CREATE TABLE dept (

dno INT PRIMARY KEY,

dname VARCHAR(100),

mgreno INT

);

-- Create workson table

CREATE TABLE workson (

eno INT,

pno INT,

hours INT,

PRIMARY KEY (eno, pno),

FOREIGN KEY (eno) REFERENCES emp(eno),

FOREIGN KEY (pno) REFERENCES proj(pno)

);

-- Insert records into emp table

INSERT INTO emp (eno, ename, bdate, title, salary, dno) VALUES

(1, 'John Doe', '1990-01-01', 'EE', 40000.00, 1),

(2, 'Jane Smith', '1992-05-10', 'SA', 45000.00, 2),

(3, 'Robert Johnson', '1988-12-15', 'SE', 50000.00, 1),

(4, 'Alice Brown', '1995-07-20', 'EE', 38000.00, 3),

(5, 'Michael Davis', '1991-11-30', 'SA', 55000.00, 2);

-- Insert records into proj table

INSERT INTO proj (pno, pname, budget, dno) VALUES

(1, 'Project A', 150000.00, 1),

(2, 'Project B', 90000.00, 2),

(3, 'Project C', 200000.00, 3);

-- Insert records into dept table

INSERT INTO dept (dno, dname, mgreno) VALUES

(1, 'D1', 1),

(2, 'D2', 2),

(3, 'D3', 3);

-- Insert records into workson table

INSERT INTO workson (eno, pno, hours) VALUES

(1, 1, 40),

(2, 1, 35),

(3, 2, 45),

(4, 2, 30),

(5, 3, 50);

1) Write an SQL query that returns the project number and name for projects with a budget greater than $100,000.

SELECT pno, pname

FROM proj

WHERE budget > 100000.00;

2) Write an SQL query that returns the employees (number and name only) who have a title of 'EE' or 'SA' and salary more than $35,000.

SELECT eno, ename

FROM emp

WHERE title IN ('EE', 'SA') AND salary > 35000.00;

3) Write an SQL query that returns the employees (name only) in department 'D1' ordered by decreasing salary.

SELECT ename

FROM emp

WHERE dno = 1

ORDER BY salary DESC;

4) Write an SQL query that returns the departments (all fields) ordered by ascending department name.

SELECT \*

FROM dept

ORDER BY dname ASC;

5) Write an SQL query that returns the employee name, department name, and employee title.

SELECT e.ename, d.dname, e.title

FROM emp e

JOIN dept d ON e.dno = d.dno;

**[1]** Design and develop below database and execute following SQL DML statements using

MySQL .

emp (eno, ename, bdate, title, salary, dno)

proj (pno, pname, budget, dno)

dept (dno, dname, mgreno)

workson (eno, pno, hours)

-- Create emp table

CREATE TABLE emp (

eno INT PRIMARY KEY,

ename VARCHAR(100),

bdate DATE,

title VARCHAR(100),

salary DECIMAL(8, 2),

dno INT

);

-- Create proj table

CREATE TABLE proj (

pno INT PRIMARY KEY,

pname VARCHAR(100),

budget DECIMAL(10, 2),

dno INT

);

-- Create dept table

CREATE TABLE dept (

dno INT PRIMARY KEY,

dname VARCHAR(100),

mgreno INT

);

-- Create workson table

CREATE TABLE workson (

eno INT,

pno INT,

hours INT,

PRIMARY KEY (eno, pno),

FOREIGN KEY (eno) REFERENCES emp(eno),

FOREIGN KEY (pno) REFERENCES proj(pno)

);

-- Insert records into emp table

INSERT INTO emp (eno, ename, bdate, title, salary, dno) VALUES

(1, 'John Doe', '1990-01-01', 'EE', 40000.00, 1),

(2, 'Jane Smith', '1992-05-10', 'SA', 45000.00, 2),

(3, 'Robert Johnson', '1988-12-15', 'SE', 50000.00, 1),

(4, 'Alice Brown', '1995-07-20', 'EE', 38000.00, 3),

(5, 'Michael Davis', '1991-11-30', 'SA', 55000.00, 2);

-- Insert records into proj table

INSERT INTO proj (pno, pname, budget, dno) VALUES

(1, 'Project A', 150000.00, 1),

(2, 'Project B', 90000.00, 2),

(3, 'Project C', 200000.00, 3);

-- Insert records into dept table

INSERT INTO dept (dno, dname, mgreno) VALUES

(1, 'Consulting', 1),

(2, 'Development', 2),

(3, 'Testing', 3);

-- Insert records into workson table

INSERT INTO workson (eno, pno, hours) VALUES

(1, 1, 40),

(2, 1, 35),

(3, 2, 45),

(4, 2, 30),

(5, 3, 50);

1) Write an SQL query that returns the project name, hours worked, and project number for all works on records where hours > 10.

SELECT proj.pname, workson.hours, proj.pno

FROM workson

JOIN proj ON workson.pno = proj.pno

WHERE workson.hours > 10;

2) Write an SQL query that returns the project name, department name, and budget for all projects with a budget < $50,000.

SELECT proj.pname, dept.dname, proj.budget

FROM proj

JOIN dept ON proj.dno = dept.dno

WHERE proj.budget < 50000.00;

3) Write an SQL query that returns the employee numbers and salaries of all employees in the 'Consulting' department ordered by descending salary.

SELECT emp.eno, emp.salary

FROM emp

JOIN dept ON emp.dno = dept.dno

WHERE dept.dname = 'Consulting'

ORDER BY emp.salary DESC;

4) Write an SQL query that returns the employee name, project name, employee title, and hours for all works on records.

SELECT emp.ename, proj.pname, emp.title, workson.hours

FROM workson

JOIN emp ON workson.eno = emp.eno

JOIN proj ON workson.pno = proj.pno;

**[6]** Consider the following relation: Emp\_company (ename, cname, salary, joiningdate, and

city)

1. List the maximum salary in all companies.

SELECT MAX(salary) AS max\_salary

FROM Emp\_company;

2. Find maximum salary of employees of “TATA‟ Company.

SELECT MAX(salary) AS max\_salary

FROM Emp\_company

WHERE cname = 'TATA';

3. Count the number employees working in”ACC” company.

SELECT COUNT(\*) AS employee\_count

FROM Emp\_company

WHERE cname = 'ACC';

4. Display the names of companies and the maximum salary in that company.

SELECT cname, MAX(salary) AS max\_salary

FROM Emp\_company

GROUP BY cname;

5. Find the average salary of each company.

SELECT cname, AVG(salary) AS avg\_salary

FROM Emp\_company

GROUP BY cname;

6. Increase the salary of an employee by 15% who are living in city Mumbai.

UPDATE Emp\_company

SET salary = salary \* 1.15

WHERE city = 'Mumbai';

7. Delete rows of the table having salary less than 5000.

DELETE FROM Emp\_company

WHERE salary < 5000;

8. List the names of employees who joined in Jan-2000

SELECT ename

FROM Emp\_company

WHERE DATE\_FORMAT(joiningdate, '%Y-%m') = '2000-01';

**[5]** Consider following Computer training institute database schema with following information.

Course(coursecode, coursename, syllabus)

Batch(batchcode, coursecode, startingdate, duration, coursefees, netincome, expectedincome)

Enquiry(enquirynumber, fname, lname, coursecode, plotno, street, city, phone, enquirydate,

enquirystatus)

Enrollment(rollno, enquirynumber, batchcode, enrollmentdate) Write following Mysql queries:

1. Display the contents of batch for particular course having coursecode=10

SELECT \*

FROM Batch

WHERE coursecode = 10;

2. Display the batchcode of batch having expectedincome more than netincome by 2000

SELECT batchcode

FROM Batch

WHERE expectedincome > netincome + 2000;

3. Display the coursename, startingdate of all batches

SELECT c.coursename, b.startingdate

FROM Course c

JOIN Batch b ON c.coursecode = b.coursecode;

4. Find out coursename with coursefees greater than 50000

SELECT coursename

FROM Course

WHERE coursefees > 50000;

5. Display the details of batch having a specified coursename.

SELECT \*

FROM Batch

WHERE coursecode IN (SELECT coursecode FROM Course WHERE coursename = 'specified\_coursename');

6. Find out the number of persons whose name starts with „S‟.

SELECT COUNT(\*) AS person\_count

FROM Enquiry

WHERE fname LIKE 'S%';

7. Delete all rows from enquiry for a specified coursename.

DELETE FROM Enquiry

WHERE coursecode IN (SELECT coursecode FROM Course WHERE coursename = 'specified\_coursename');

**[7]** Consider the following Bank DB schema.

Deposit(accountno, cname, amount, acctdate,bname)

Branch(bname,city)

Customer(cname,city)

Borrow(loanno, cname, bname,amount)

Write the following Mysql queries.

1. List all data from borrow table.

SELECT \* FROM Borrow;

2. Get customer names who has deposit greater than 1000 and name like “A”.

SELECT cname

FROM Deposit

WHERE amount > 1000 AND cname LIKE 'A%';

3. Display name of customer with amount in descending order.

SELECT cname, amount

FROM Deposit

ORDER BY amount DESC;

4. List total loan.

SELECT SUM(amount) AS total\_loan

FROM Borrow;

5. List branchname and branch wise deposit.

SELECT bname, SUM(amount) AS total\_deposit

FROM Deposit

GROUP BY bname;

6. Select the branches having sum of deposit more than 4000.

SELECT bname

FROM Deposit

GROUP BY bname

HAVING SUM(amount) > 4000;

**[8]** Consider the following Bank DB schema.

Deposit(accountno, cname, amount, acctdate,bname)

Branch(bname,city)

Customer(cname,city)

Borrow(loanno, cname, bname,amount)

Write the following Mysql queries.

1. List all data from borrow table.

SELECT \* FROM Borrow;

2. Get customer names who has deposit greater than 1000 and name like “A”.

SELECT cname

FROM Deposit

WHERE amount > 1000 AND cname LIKE 'A%';

3. List the branches having sum of deposit more than 1000 and located in Mumbai.

SELECT bname

FROM Deposit

JOIN Branch ON Deposit.bname = Branch.bname

WHERE city = 'Mumbai'

GROUP BY bname

HAVING SUM(amount) > 1000;

4. List 10% interest to all depositors living in pune

UPDATE Deposit

SET amount = amount \* 1.10

WHERE cname IN (SELECT cname FROM Customer WHERE city = 'Pune');

5. Transfer Rs 500 from account of Anil to the account of Sunil.

UPDATE Deposit

SET amount = amount - 500

WHERE cname = 'Anil';

UPDATE Deposit

SET amount = amount + 500

WHERE cname = 'Sunil';

6. Delete depositors having deposit less than 500.

DELETE FROM Deposit

WHERE amount < 500;

**[9]**Design and Develop DB for “Customerorder” with all constraints(Not NULL, PrimaryKey, Foreign Key).

Customer (Cust\_no, name, Street, city, state)

Order (Order\_no,Cust\_no,Order\_date,Ship\_date,Tocity,ToState,ToZip)

Contains(Order\_no,Stock\_no,quantity,Discount)-

Stock(Stock\_no,price,tax)

1) Create View on columns Order\_no and Customer\_no with order table.

2) Display Name of customer, City, OrderNo and order date of customer\_no 101.

3)Display price and discount of stock\_no 102.

CREATE TABLE Customer (

Cust\_no INT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

Street VARCHAR(255),

city VARCHAR(255),

state VARCHAR(255)

);

CREATE TABLE Order (

Order\_no INT PRIMARY KEY,

Cust\_no INT,

Order\_date DATE,

Ship\_date DATE,

Tocity VARCHAR(255),

ToState VARCHAR(255),

ToZip INT,

FOREIGN KEY (Cust\_no) REFERENCES Customer(Cust\_no)

);

CREATE TABLE Contains (

Order\_no INT,

Stock\_no INT,

quantity INT,

Discount FLOAT,

PRIMARY KEY (Order\_no, Stock\_no),

FOREIGN KEY (Order\_no) REFERENCES Order(Order\_no),

FOREIGN KEY (Stock\_no) REFERENCES Stock(Stock\_no)

);

CREATE TABLE Stock (

Stock\_no INT PRIMARY KEY,

price DECIMAL(10, 2),

tax DECIMAL(10, 2)

);

-- 1) Create a view on columns Order\_no and Customer\_no with the Order table.

CREATE VIEW OrderView AS

SELECT Order\_no, Cust\_no

FROM Order;

-- 2) Display the Name of the customer, City, OrderNo, and OrderDate of customer\_no 101.

SELECT C.name, C.city, O.Order\_no, O.Order\_date

FROM Customer C

JOIN Order O ON C.Cust\_no = O.Cust\_no

WHERE C.Cust\_no = 101;

-- 3) Display the price and discount of stock\_no 102.

SELECT price, Discount

FROM Stock

JOIN Contains ON Stock.Stock\_no = Contains.Stock\_no

WHERE Stock.Stock\_no = 102;

**[12]**Design the DB for ”Pets” and perform following operations:

Pet(pet\_name,owner,sex,birth\_date,death\_date)

CREATE TABLE Pet (

pet\_name VARCHAR(255),

owner VARCHAR(255),

sex CHAR(1),

birth\_date DATE,

death\_date DATE

);

2) Display all the pets information.

SELECT \*

FROM Pet;

3)Display all pets in ascending and descending order according to birth date.

-- Ascending order

SELECT \*

FROM Pet

ORDER BY birth\_date ASC;

-- Descending order

SELECT \*

FROM Pet

ORDER BY birth\_date DESC;

4) Display the pet information of specific owner.

SELECT \*

FROM Pet

WHERE owner = 'specific\_owner';

5) count the pet which have same birth\_date

SELECT birth\_date, COUNT(\*) AS pet\_count

FROM Pet

GROUP BY birth\_date

HAVING COUNT(\*) > 1;

**[11]**Create a table „emp‟ with the following columns by assuming suitable data type and size with correct syntax in SQL.

Emp–id, Ename, City, State, Salary, Age, Hire\_ date.

CREATE TABLE emp (

Emp\_id INT,

Ename VARCHAR(255),

City VARCHAR(255),

State VARCHAR(255),

Salary DECIMAL(10, 2),

Age INT,

Hire\_date DATE

);

Give an expression in SQL to solve each of the following queries :

i) Find the names of all employees whose name starts with „Sa‟.

SELECT Ename

FROM emp

WHERE Ename LIKE 'Sa%';

ii) List all the employees name and salary whose age is less than 40 years.

SELECT Ename, Salary

FROM emp

WHERE Age < 40;

iii) Select the employees whose salary is between Rs. 20000 and Rs. 30000.

SELECT \*

FROM emp

WHERE Salary BETWEEN 20000 AND 30000;

**[10]** A database consists of following tables.

PROJECT(PNO, PNAME, CHIEF)

EMPLOYEE(EMPNO, EMPNAME)

ASSIGNED(PNO,EMPNO)

A. Get count of employees working on project.

SELECT COUNT(\*) AS employee\_count

FROM ASSIGNED;

B. Get details of employee working on project pr002.

SELECT EMPLOYEE.\*

FROM EMPLOYEE

INNER JOIN ASSIGNED ON EMPLOYEE.EMPNO = ASSIGNED.EMPNO

INNER JOIN PROJECT ON ASSIGNED.PNO = PROJECT.PNO

WHERE PROJECT.PNO = 'pr002';

C. Get details of employee working on project DBMS.

SELECT EMPLOYEE.\*

FROM EMPLOYEE

INNER JOIN ASSIGNED ON EMPLOYEE.EMPNO = ASSIGNED.EMPNO

INNER JOIN PROJECT ON ASSIGNED.PNO = PROJECT.PNO

WHERE PROJECT.PNAME = 'DBMS';

**[13]** Write PL/SQL block to implement Curser to calculate grade of minimum 10 students.

DECLARE

CURSOR student\_cursor IS

SELECT student\_id, student\_name, marks

FROM students

WHERE ROWNUM <= 10;

v\_grade VARCHAR2(10);

BEGIN

FOR student\_rec IN student\_cursor LOOP

IF student\_rec.marks >= 90 THEN

v\_grade := 'A';

ELSIF student\_rec.marks >= 80 THEN

v\_grade := 'B';

ELSIF student\_rec.marks >= 70 THEN

v\_grade := 'C';

ELSIF student\_rec.marks >= 60 THEN

v\_grade := 'D';

ELSE

v\_grade := 'F';

END IF;

DBMS\_OUTPUT.PUT\_LINE('Student ID: ' || student\_rec.student\_id || ', Name: ' || student\_rec.student\_name || ', Grade: ' || v\_grade);

END LOOP;

END;

/

**[14]** Write a database trigger on Library 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 Library\_Audit table.

CREATE OR REPLACE TRIGGER Library\_Trigger

AFTER UPDATE OR DELETE ON Library

FOR EACH ROW

DECLARE

v\_action VARCHAR2(10);

BEGIN

IF UPDATING THEN

v\_action := 'UPDATED';

ELSE

v\_action := 'DELETED';

END IF;

INSERT INTO Library\_Audit (Book\_ISBN, Title, Publisher\_Name, Price, Date\_Of\_Publication, Book\_Copy, Action)

VALUES (:OLD.Book\_ISBN, :OLD.Title, :OLD.Publisher\_Name, :OLD.Price, :OLD.Date\_Of\_Publication, :OLD.Book\_Copy, v\_action);

END;

/

**[15]** Write a Stored Procedure namely proc\_Grade for the categorization of student. 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 procedure created with above requirement.

Stud\_Marks(name, total\_marks)

Result(Roll,Name, Class)

CREATE OR REPLACE PROCEDURE proc\_Grade

IS

BEGIN

FOR student\_rec IN (SELECT name, total\_marks FROM Stud\_Marks) LOOP

DECLARE

v\_class VARCHAR2(20);

BEGIN

IF student\_rec.total\_marks >= 990 THEN

v\_class := 'Distinction';

ELSIF student\_rec.total\_marks >= 900 THEN

v\_class := 'First Class';

ELSIF student\_rec.total\_marks >= 825 THEN

v\_class := 'Higher Second Class';

ELSE

v\_class := 'Not Classified';

END IF;

INSERT INTO Result (Roll, Name, Class)

VALUES (student\_rec.roll, student\_rec.name, v\_class);

END;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Categorization completed successfully.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/