Savitribai Phule Pune University

F. Y. B. B. A. (C. A.) Semester-II

(CBCS 2019 Pattern)

RDBMS

CA-206: Lab Book

Student Name: _____

College Name: _____

Roll No.: _____ **Division:** _____ **Roll No.:** _____

Academic Year: _____

CERTIFICATE

This is to certify that Mr./Ms. _____
Seat Number _____ of F.Y.B.B.A. (C.A) Sem-II has successfully completed Laboratory course (RDBMS) in the year _____ .

She has scored _____mark out of 10 (For Lab Book).

Subject Teacher H.O.D./Coordinator

Internal Examiner **External**

Assignment No.1

Data Type, PLSQL Block and Control Structure.

SET A:

1. Write a PL/SQL block to accept a number and display multiplication table of the given number.

»
```
DECLARE
   num      NUMBER;
   i        NUMBER := 1;
BEGIN
   -- Accept input from the user
   num := &Enter_Number;

   DBMS_OUTPUT.PUT_LINE('Multiplication Table for ' || num || ':');
   WHILE i <= 10 LOOP
      DBMS_OUTPUT.PUT_LINE(num || ' x ' || i || ' = ' || (num * i));
      i := i + 1;
   END LOOP;
END;
/
```

2. Write a PL/SQL block which will accept student details, calculate the class using per value and insert the record into Student (rno, sname, class, per, class) table.

```
DECLARE
  v_rno    Student.rno%TYPE;
  v_sname  Student.sname%TYPE;
  v_per    Student.per%TYPE;
  v_class  Student.class%TYPE;
BEGIN
  v_rno := &Enter_Roll_Number;
  v_sname := '&Enter_Student_Name';
  v_per := &Enter_Percentage;
   IF v_per >= 75 THEN
     v_class := 'Distinction';
  ELSIF v_per >= 60 THEN
     v_class := 'First Class';
  ELSIF v_per >= 50 THEN
     v_class := 'Second Class';
  ELSIF v_per >= 35 THEN
     v_class := 'Pass';
  ELSE
     v_class := 'Fail';
  END IF;
  INSERT INTO Student (rno, sname, per, class)
  VALUES (v_rno, v_sname, v_per, v_class);
DBMS_OUTPUT.PUT_LINE('Student record inserted successfully.');
END;
/
```

SET B:

1. Write a PL/SQL block which will accept roll number of a student and display record of student from student table(use %ROWTYPE attribute).

```
DECLARE
   v_rno      Student.rno%TYPE;
   v_student  Student%ROWTYPE;
BEGIN
   -- Accept roll number input
   v_rno := &Enter_Roll_Number;
-- Fetch the record into the %ROWTYPE variable
   SELECT * INTO v_student
   FROM Student
   WHERE rno = v_rno;

   -- Display the student details
   DBMS_OUTPUT.PUT_LINE('Roll Number : ' || v_student.rno);
   DBMS_OUTPUT.PUT_LINE('Name        : ' || v_student.sname);
   DBMS_OUTPUT.PUT_LINE('Percentage  : ' || v_student.per);
   DBMS_OUTPUT.PUT_LINE('Class       : ' || v_student.class);

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No student found with roll number ' ||
v_rno);
END; /
```

2. Write a PL/SQL block which will accept roll number from student, select name and percentage of the student and calculate grade using percentage value. Display the record.(use %TYPE)

```
DECLARE
   v_rno     Student.rno%TYPE;
   v_name    Student.sname%TYPE;
   v_per     Student.per%TYPE;
   v_grade   VARCHAR2(20);
BEGIN
   -- Accept roll number
   v_rno := &Enter_Roll_Number;

   -- Fetch name and percentage
   SELECT sname, per INTO v_name, v_per
   FROM Student
   WHERE rno = v_rno;

   -- Calculate grade based on percentage
   IF v_per >= 75 THEN
      v_grade := 'A';
   ELSIF v_per >= 60 THEN
      v_grade := 'B';
   ELSIF v_per >= 50 THEN
      v_grade := 'C';
   ELSIF v_per >= 35 THEN
      v_grade := 'D';
   ELSE
      v_grade := 'F';
   END IF;

   -- Display the result
   DBMS_OUTPUT.PUT_LINE('Name       : ' || v_name);
   DBMS_OUTPUT.PUT_LINE('Percentage : ' || v_per);
   DBMS_OUTPUT.PUT_LINE('Grade      : ' || v_grade);

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No student found with Roll Number '
|| v_rno);
END; /
```

Assignment No. 2
Error and Exception Handling

SET A:

Consider the following entities and their relationships.

Wholesaler (w_no, w_name, address, city)

Product (product_no, product_name, rate)

Relation between Wholesaler and Product is Many to Many with quantity as descriptive attribute.

Constraint: Primary key, rate should be > 0.

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a function to accept quantity from user. Quantity must be within range 50-200. If user enters the quantity out of range then raise an user defined exception “quantity_out_of _range” otherwise enter the record in table.

» **Step 1: Create Tables in 3NF**

```
CREATE TABLE Wholesaler (
 w_no     NUMBER PRIMARY KEY,
w_name   VARCHAR2(50),
address  VARCHAR2(100),
city     VARCHAR2(50)
);

-- Product Table
CREATE TABLE Product (
   product_no    NUMBER PRIMARY KEY,
   product_name  VARCHAR2(50),
   rate          NUMBER CHECK (rate > 0)
);

-- Wholesaler_Product (Associative table for Many-to-Many with quantity)
```

```
CREATE TABLE Wholesaler_Product (
   w_no        NUMBER,
   product_no  NUMBER,
   quantity    NUMBER,
   PRIMARY KEY (w_no, product_no),
   FOREIGN KEY (w_no) REFERENCES Wholesaler(w_no),
   FOREIGN KEY (product_no) REFERENCES Product(product_no)
);
```

Step 2: **Function with User-Defined Exception**

```
          CREATE OR REPLACE FUNCTION insert_quantity(
   p_wno        IN Wholesaler.w_no%TYPE,
   p_pno        IN Product.product_no%TYPE,
   p_quantity   IN NUMBER
) RETURN VARCHAR2
IS
   -- User-defined exception
   quantity_out_of_range EXCEPTION;

BEGIN
   -- Check quantity range
   IF p_quantity < 50 OR p_quantity > 200 THEN
      RAISE quantity_out_of_range;
   END IF;

   -- Insert into Wholesaler_Product table
   INSERT INTO Wholesaler_Product (w_no, product_no, quantity)
   VALUES (p_wno, p_pno, p_quantity);

   RETURN 'Record inserted successfully.';

-- Exception handler
EXCEPTION
   WHEN quantity_out_of_range THEN
      RETURN 'Error: Quantity must be between 50 and 200.';
   WHEN OTHERS THEN
      RETURN 'Error: ' || SQLERRM;
END;
/
```

2. Write a PL/SQL block which accept rate from user. If user enters rate less than or equal to zero then raise an user defined exception "Invalid_Rate_Value" otherwise display message "Correct Input".

```
DECLARE
   v_rate NUMBER;

   -- User-defined exception
   Invalid_Rate_Value EXCEPTION;
BEGIN
   -- Accept input from user
   v_rate := &Enter_Rate;

   -- Check for valid rate
   IF v_rate <= 0 THEN
      RAISE Invalid_Rate_Value;
   ELSE
      DBMS_OUTPUT.PUT_LINE('Correct Input');
   END IF;

-- Exception handler
EXCEPTION
   WHEN Invalid_Rate_Value THEN
      DBMS_OUTPUT.PUT_LINE('Error: Invalid Rate Value');
END;
/
```

SET B:

Consider the following entities and their relationships.

Student (rollno, sname, class, timetable, mobileno)

Lab (LabNo, LabName, capacity, equipment)

Relation between Student and Lab is Many to One.

Constraint: Primary Key, capacity should not be null.

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a function to accept lab number from user as parameter. ” if user enters invalid lab number then raise an user defined exception “Invalid_Lab_No” otherwise display the student details of the same lab.

➢ **Step 1: Table Structure in 3NF**

```
CREATE TABLE Lab (
    LabNo       NUMBER PRIMARY KEY,
    LabName     VARCHAR2(50),
    Capacity    NUMBER NOT NULL,
    Equipment   VARCHAR2(100)
);
-- Student Table (Many to One relationship with Lab)
CREATE TABLE Student (
    RollNo      NUMBER PRIMARY KEY,
    SName       VARCHAR2(50),
    Class       VARCHAR2(20),
    Timetable   VARCHAR2(100),
    MobileNo    VARCHAR2(15),
    LabNo       NUMBER,
    FOREIGN KEY (LabNo) REFERENCES Lab(LabNo)
```

);

Step 2: Function to Display Student Details by LabNo

```
 CREATE OR REPLACE FUNCTION get_students_by_lab(p_labno IN
Lab.LabNo%TYPE)
RETURN VARCHAR2
IS
   -- Declare user-defined exception
   Invalid_Lab_No EXCEPTION;

   -- Check variable
   v_count NUMBER:= 0;

BEGIN
   -- Check if lab number exists
   SELECT COUNT(*) INTO v_count
   FROM Lab
   WHERE LabNo = p_labno;

   IF v_count = 0 THEN
      RAISE Invalid_Lab_No;
   END IF;

   -- Display students in that lab
   FOR s IN (
      SELECT RollNo, SName, Class, Timetable, MobileNo
      FROM Student
      WHERE LabNo = p_labno
```

```
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('Roll No   : ' || s.RollNo);
        DBMS_OUTPUT.PUT_LINE('Name      : ' || s.SName);
        DBMS_OUTPUT.PUT_LINE('Class     : ' || s.Class);
        DBMS_OUTPUT.PUT_LINE('Timetable : ' || s.Timetable);
        DBMS_OUTPUT.PUT_LINE('Mobile No : ' || s.MobileNo);
        DBMS_OUTPUT.PUT_LINE('--------------------------');
    END LOOP;

    RETURN 'Student details displayed successfully.';

-- Exception handler
EXCEPTION
    WHEN Invalid_Lab_No THEN
        RETURN 'Error: Invalid Lab Number';
    WHEN OTHERS THEN
        RETURN 'Unexpected Error: ' || SQLERRM;
END;
/
```

Assignment No. 3
Procedure

SET A:

Consider the following entities and their relationship.
Newspaper (name,language , publisher , cost)
Cities (pincode , city, state)
Relationship between Newspaper and Cities is many-to-many with descriptive attribute daily required
Constraints: name and pincode primary key

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a procedure to calculate city wise total cost of each newspaper.

» **Step 1**: **Create Relational Database in 3NF**

```
CREATE TABLE Newspaper (
   name       VARCHAR2(50) PRIMARY KEY,
   language   VARCHAR2(30),
   publisher  VARCHAR2(50),
   cost       NUMBER CHECK (cost > 0)
);

-- Cities Table
CREATE TABLE Cities (
   pincode    NUMBER PRIMARY KEY,
   city       VARCHAR2(50),
   state      VARCHAR2(50)
);

-- Associative Table: Many-to-Many with descriptive attribute
CREATE TABLE Newspaper_City (
```

```
    name          VARCHAR2(50),
    pincode       NUMBER,
    daily_required NUMBER,
    PRIMARY KEY (name, pincode),
    FOREIGN KEY (name) REFERENCES Newspaper(name),
    FOREIGN KEY (pincode) REFERENCES Cities(pincode)
);
```

Step 2: **Procedure to Calculate City-wise Total Cost**

```
        CREATE OR REPLACE PROCEDURE citywise_total_cost
IS
BEGIN
    FOR rec IN (
        SELECT
            c.city,
            nc.name AS newspaper_name,
            SUM(n.cost * nc.daily_required) AS total_cost
        FROM
            Newspaper n
            JOIN Newspaper_City nc ON n.name = nc.name
            JOIN Cities c ON nc.pincode = c.pincode
        GROUP BY
            c.city, nc.name
        ORDER BY
            c.city, nc.name
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('City       : ' || rec.city);
        DBMS_OUTPUT.PUT_LINE('Newspaper   : ' ||
rec.newspaper_name);
        DBMS_OUTPUT.PUT_LINE('Total Cost  : Rs. ' || rec.total_cost);
        DBMS_OUTPUT.PUT_LINE('------------------------------');
    END LOOP;
END;
/
```

2. Write a procedure which display details of news papers having cost greater than 2 Rs.

Step 1: Table Reminder

```
        CREATE TABLE Newspaper (
   name       VARCHAR2(50) PRIMARY KEY,
   language   VARCHAR2(30),
   publisher  VARCHAR2(50),
   cost       NUMBER CHECK (cost > 0)
);
```

Step 2: Procedure to Display Newspapers with Cost > ₹2

```
        CREATE OR REPLACE PROCEDUR
           show_expensive_newspapers
IS
BEGIN
   FOR rec IN (
      SELECT *
      FROM Newspaper
      WHERE cost > 2
   )
   LOOP
      DBMS_OUTPUT.PUT_LINE('Name      : ' || rec.name);
      DBMS_OUTPUT.PUT_LINE('Language  : ' || rec.language);
      DBMS_OUTPUT.PUT_LINE('Publisher : ' || rec.publisher);
      DBMS_OUTPUT.PUT_LINE('Cost      : Rs. ' || rec.cost);
      DBMS_OUTPUT.PUT_LINE('------------------------------');
   END LOOP;
   END;
    /
```

SET B:

Consider the following entities and their relationships.

Library(Lno, Lname, Location, Librarian, no_of_books)

Book(Bid, Bname, Author_Name, Price, publication)

Relation between Library and Book is one to many.

Constraint: Primary key, Price should not be null.

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a procedure which will accept publication name from user and display details of books published by it.

```
CREATE TABLE Library (
    Lno NUMBER PRIMARY KEY,
    Lname VARCHAR2(100),
    Location VARCHAR2(100),
    Librarian VARCHAR2(100),
    no_of_books NUMBER
);
CREATE TABLE Book (
    Bid NUMBER PRIMARY KEY,
    Bname VARCHAR2(100),
    Author_Name VARCHAR2(100),
    Price NUMBER NOT NULL,
    publication VARCHAR2(100),
    Lno NUMBER,
    FOREIGN KEY (Lno) REFERENCES Library(Lno)
);
CREATE OR REPLACE PROCEDURE
display_books_by_publication(pub_name IN VARCHAR2)
```

```
IS
BEGIN
   FOR rec IN (
      SELECT Bid, Bname, Author_Name, Price, publication
      FROM Book
      WHERE publication = pub_name
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Book ID: ' || rec.Bid);
      DBMS_OUTPUT.PUT_LINE('Book Name: ' || rec.Bname);
      DBMS_OUTPUT.PUT_LINE('Author Name: ' || rec.Author_Name);
      DBMS_OUTPUT.PUT_LINE('Price: ' || rec.Price);
      DBMS_OUTPUT.PUT_LINE('Publication: ' || rec.publication);
      DBMS_OUTPUT.PUT_LINE('-----------------------------');
   END LOOP;
END;
/
```

2. Write a procedure which will accept Library number from user and display Book name and their price.

```
DECLARE
   -- Input variable to hold Library number
   lib_no NUMBER := &Enter_Library_Number;
BEGIN
   -- Loop through books of the given library
   FOR rec IN (
      SELECT Bname, Price
      FROM Book
      WHERE Lno = lib_no
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Book Name: ' || rec.Bname || ', Price: ' || rec.Price);
   END LOOP;
END;
/
```

Assignment No.4

Function

SET A:

Consider the following entities and their relationships.

Client (client_no, client_name, address, birthdate)

Policy_info (policy_no, desc, maturity_amt, prem_amt, date)

Relation between Client and Policy_info is Many to Many Constraint: Primary key, prem_amt and maturity_amt should be > 0

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a function which will return total maturity amount of policies of a particular client.

» CREATE TABLE Client (

client_no NUMBER PRIMARY KEY,

client_name VARCHAR2(100),

address VARCHAR2(200),

birthdate DATE

);

CREATE TABLE Policy_info (

policy_no NUMBER PRIMARY KEY,

description VARCHAR2(200),

maturity_amt NUMBER CHECK (maturity_amt > 0),

prem_amt NUMBER CHECK (prem_amt > 0),

policy_date DATE

);

```
CREATE TABLE Client_Policy (
    client_no   NUMBER,
    policy_no   NUMBER,
    PRIMARY KEY (client_no, policy_no),
    FOREIGN KEY (client_no) REFERENCES Client(client_no),
    FOREIGN KEY (policy_no) REFERENCES Policy_info(policy_no)
);
CREATE OR REPLACE FUNCTION get_total_maturity_amt(p_client_no IN
NUMBER)
RETURN NUMBER
IS total_maturity NUMBER := 0;
BEGIN
    SELECT NVL(SUM(p.maturity_amt), 0)
    INTO total_maturity
    FROM Policy_info p
    JOIN Client_Policy cp ON p.policy_no = cp.policy_no
    WHERE cp.client_no = p_client_no;
 RETURN total_maturity;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;
    WHEN OTHERS THEN
        RAISE;
END;/
```

2. Write a function which will return minimum maturity amount of all policies.

```
>> CREATE TABLE Client (
    client_no     NUMBER PRIMARY KEY,
    client_name   VARCHAR2(100),
    address       VARCHAR2(200),
    birthdate     DATE
);
-- POLICY_INFO TABLE
CREATE TABLE Policy_info (
    policy_no      NUMBER PRIMARY KEY,
    description    VARCHAR2(200),
    maturity_amt   NUMBER CHECK (maturity_amt > 0),
    prem_amt       NUMBER CHECK (prem_amt > 0),
    policy_date    DATE
);

-- CLIENT_POLICY: junction table for many-to-many relation

CREATE TABLE Client_Policy (
    client_no   NUMBER,
    policy_no   NUMBER,
    PRIMARY KEY (client_no, policy_no),
    FOREIGN KEY (client_no) REFERENCES Client(client_no),
    FOREIGN KEY (policy_no) REFERENCES Policy_info(policy_no)
);
CREATE OR REPLACE FUNCTION get_min_maturity_amt
RETURN NUMBER
IS min_maturity NUMBER;
BEGIN
    SELECT MIN(maturity_amt)
    INTO min_maturity
    FROM Policy_info;

RETURN min_maturity;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
       RETURN NULL;
    WHEN OTHERS THEN
       RAISE;
END; /
```

SET B:

Consider the following Item_Supplier database

Item (itemno, itemname)

Supplier (supplier_No , supplier_name, address, city)

Relationship between Item and Supplier is many-to-many with descriptive attribute rate and quantity

Constraints: itemno ,supplier_No primary key

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write function to print the total number of suppliers who supplies "Keyboard"

```
» CREATE TABLE Item (
    itemno     NUMBER PRIMARY KEY,
    itemname   VARCHAR2(100));
  CREATE TABLE Supplier (
    supplier_no    NUMBER PRIMARY KEY,
    supplier_name  VARCHAR2(100),
    address        VARCHAR2(200),
    city           VARCHAR2(100));
  CREATE TABLE Item_Supplier (
    itemno       NUMBER,
    supplier_no  NUMBER,
    rate         NUMBER,
    quantity     NUMBER,
    PRIMARY KEY (itemno, supplier_no),
    FOREIGN KEY (itemno) REFERENCES Item(itemno),
    FOREIGN KEY (supplier_no) REFERENCES Supplier(supplier_no));
  CREATE OR REPLACE FUNCTION count_keyboard_suppliers
  RETURN NUMBER
  IS total_count NUMBER := 0;
  BEGIN
    SELECT COUNT(DISTINCT s.supplier_no)
    INTO total_count
    FROM Item i
    JOIN Item_Supplier isup ON i.itemno = isup.itemno
    JOIN Supplier s ON s.supplier_no = isup.supplier_no
    END; /
```

2. Write function which will return rate of “Harddisk” supplied by “Mr. Patil”.

```
CREATE TABLE Item (
   itemno      NUMBER PRIMARY KEY,
   itemname    VARCHAR2(100));

CREATE TABLE Supplier (

   supplier_no     NUMBER PRIMARY KEY,
   supplier_name   VARCHAR2(100),
   address         VARCHAR2(200),
   city            VARCHAR2(100));
CREATE TABLE Item_Supplier (
   itemno        NUMBER,
   supplier_no   NUMBER,
   rate          NUMBER,
   quantity      NUMBER,
   PRIMARY KEY (itemno, supplier_no),
   FOREIGN KEY (itemno) REFERENCES Item(itemno),
   FOREIGN KEY (supplier_no) REFERENCES Supplier(supplier_no));
CREATE OR REPLACE FUNCTION get_rate_harddisk_by_patil
RETURN NUMBER
IS
   v_rate NUMBER;
BEGIN
   SELECT isup.rate
   INTO v_rate
   FROM Item i
   JOIN Item_Supplier isup ON i.itemno = isup.itemno
   JOIN Supplier s ON s.supplier_no = isup.supplier_no
   WHERE LOWER(i.itemname) = 'harddisk'
     AND LOWER(s.supplier_name) = 'mr. patil';
RETURN v_rate;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN NULL;
   WHEN TOO_MANY_ROWS THEN
      RAISE_APPLICATION_ERROR(-20001, 'Multiple rates found.
Data inconsistency.');
   WHEN OTHERS THEN
      RAISE; END;/
```

Assignment No.5

Cursors

SET A:

Consider the following entities and their relationships.
Project (pno, pname, start_date, budget, status)
Department (dno, dname, HOD, loc)
The relationship between Project and Department is Many to One.
Constraint: Primary key. Project Status
Constraints: C – Completed, -Progressive, I –Incomplete
Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a cursor which will display list of projects started in month of "January" 2020**.**

```
CREATE TABLE Department (
    dno     NUMBER PRIMARY KEY,
    dname   VARCHAR2(100),
    HOD     VARCHAR2(100),
    loc     VARCHAR2(100)
);

-- PROJECT TABLE
CREATE TABLE Project (
    pno         NUMBER PRIMARY KEY,
    pname       VARCHAR2(100),
    start_date  DATE,
    budget      NUMBER,
    status      CHAR(1) CHECK (status IN ('C', 'P', 'I')),
    dno         NUMBER,
    FOREIGN KEY (dno) REFERENCES Department(dno)
);
DECLARE
    -- Define a cursor for projects started in January 2020
    CURSOR jan_projects_cursor IS
        SELECT pno, pname, start_date, budget, status
        FROM Project
```

```
    WHERE start_date BETWEEN TO_DATE('01-JAN-2020', 'DD-
MON-YYYY')
                    AND TO_DATE('31-JAN-2020', 'DD-MON-YYYY');

  -- Variables to hold fetched data
  v_pno        Project.pno%TYPE;
  v_pname      Project.pname%TYPE;
  v_start_date Project.start_date%TYPE;
  v_budget     Project.budget%TYPE;
  v_status     Project.status%TYPE;
BEGIN
  OPEN jan_projects_cursor;

  LOOP
    FETCH jan_projects_cursor INTO v_pno, v_pname, v_start_date,
v_budget, v_status;
    EXIT WHEN jan_projects_cursor%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('Project No: ' || v_pno || ', Name: ' ||
v_pname ||
                         ', Start Date: ' || TO_CHAR(v_start_date, 'DD-MON-
YYYY') ||
                         ', Budget: ' || v_budget || ', Status: ' || v_status);
  END LOOP;

  CLOSE jan_projects_cursor;
END;
/
```

2. Write a cursor which will display status wise project details of each department.

```
» CREATE TABLE Department (
    dno      NUMBER PRIMARY KEY,
    dname    VARCHAR2(100),
    HOD      VARCHAR2(100),
    loc      VARCHAR2(100)
);
-- PROJECT TABLE
CREATE TABLE Project (
    pno         NUMBER PRIMARY KEY,
    pname       VARCHAR2(100),
    start_date  DATE,
    budget      NUMBER,
    status      CHAR(1) CHECK (status IN ('C', 'P', 'I')), -- C=Completed, P=Progressive, I=Incomplete
    dno         NUMBER,
    FOREIGN KEY (dno) REFERENCES Department(dno)
);
DECLARE
    -- Cursor to get project details grouped by status and department
    CURSOR project_status_cursor IS
        SELECT d.dname, d.HOD, p.pname, p.start_date, p.budget, p.status
        FROM Department d
        JOIN Project p ON d.dno = p.dno
        ORDER BY d.dname, p.status;

    -- Variables to hold fetched data
```

```
    v_dname      Department.dname%TYPE;
    v_HOD        Department.HOD%TYPE;
    v_pname      Project.pname%TYPE;
    v_start_date Project.start_date%TYPE;
    v_budget     Project.budget%TYPE;
    v_status     Project.status%TYPE;

    v_last_dname Department.dname%TYPE := NULL;
    v_last_status Project.status%TYPE := NULL;
BEGIN
    OPEN project_status_cursor;

    LOOP
        FETCH project_status_cursor INTO v_dname, v_HOD, v_pname,
v_start_date, v_budget, v_status;
        EXIT WHEN project_status_cursor%NOTFOUND;

        -- Print department name and status heading when it changes
        IF v_dname != v_last_dname OR v_status != v_last_status THEN
            DBMS_OUTPUT.PUT_LINE('----------------------------------------');
            DBMS_OUTPUT.PUT_LINE('Department: ' || v_dname || ' | Status: '
||
                                 CASE v_status
                                    WHEN 'C' THEN 'Completed'
                                    WHEN 'P' THEN 'Progressive'
                                    WHEN 'I' THEN 'Incomplete'
                                    ELSE 'Unknown'
                                 END);
```

SET B:

Consider the following entities and their relationships.

Gym (Name, city, charges, scheme)

Member (ID, Name, PhoneNo, address)

Relation between Gym and member is one to many.

Constraint: Primary Key, charges must be greater than 0.

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a cursor which will display Gym details having charges more than 5000 from 'Pune' city.

```
» DECLARE
CURSOR gym_cursor IS
    SELECT Name, city, charges, scheme
    FROM Gym
    WHERE charges > 5000
     AND LOWER(city) = 'pune';
 v_name   Gym.Name%TYPE;
  v_city   Gym.city%TYPE;
  v_charges Gym.charges%TYPE;
  v_scheme Gym.scheme%TYPE;
BEGIN
  OPEN gym_cursor;
LOOP
    FETCH gym_cursor INTO v_name, v_city, v_charges, v_scheme;
    EXIT WHEN gym_cursor%NOTFOUND;
 DBMS_OUTPUT.PUT_LINE('Gym Name: ' || v_name ||
                      ', City: ' || v_city ||
                      ', Charges: ' || v_charges ||
  END LOOP;
CLOSE gym_cursor;
END;/
```

2. Write a cursor which will display city wise Gym details.

```
≫ DECLARE
 CURSOR gym_city_cursor IS
    SELECT city, Name, charges, scheme
    FROM Gym
    ORDER BY city, Name;
  v_city    Gym.city%TYPE;
  v_name    Gym.Name%TYPE;
  v_charges Gym.charges%TYPE;
  v_scheme  Gym.scheme%TYPE;
 v_last_city Gym.city%TYPE := NULL;
BEGIN
 OPEN gym_city_cursor;
   LOOP   FETCH gym_city_cursor INTO v_city, v_name, v_charges,
v_scheme;
      EXIT WHEN gym_city_cursor%NOTFOUND;
-- Print city heading only when city changes
      IF v_city != v_last_city THEN
        DBMS_OUTPUT.PUT_LINE('----------------------------');
        DBMS_OUTPUT.PUT_LINE('City: ' || v_city);
        DBMS_OUTPUT.PUT_LINE('----------------------------');
        v_last_city := v_city;
      END IF;
DBMS_OUTPUT.PUT_LINE('Gym Name: ' || v_name ||
                     ', Charges: ' || v_charges ||
                     ', Scheme: ' || v_scheme);
  END LOOP; CLOSE gym_city_cursor;
END; /
```

Assignment No.6

Triggers

SET A:

Consider the following entities and their relationships.

Employee (emp_id, emp_name, address)

Investment (inv_no, inv_name, inv_date, inv_amount)

Relation between Employee and Investment is One to Many.

Constraint: Primary key, inv_amount should be > 0.

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a trigger which will fire before insert or update on Investment having investment amount less than 10000. (Raise user defined exception and give appropriate message).

```
CREATE TABLE Employee (
   emp_id     NUMBER PRIMARY KEY,
   emp_name   VARCHAR2(100),
   address    VARCHAR2(200));
CREATE TABLE Investment (
   inv_no      NUMBER PRIMARY KEY,
   inv_name    VARCHAR2(100),
   inv_date    DATE,
   inv_amount  NUMBER CHECK (inv_amount > 0),
   emp_id      NUMBER,
   FOREIGN KEY (emp_id) REFERENCES Employee(emp_id));
CREATE OR REPLACE TRIGGER trg_check_investment_amount
BEFORE INSERT OR UPDATE ON Investment
FOR EACH ROW
DECLARE
   EXCEPTION;
   PRAGMA EXCEPTION_INIT(ex_low_amount, -20001);
BEGINIF :NEW.inv_amount < 10000 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Investment amount must
be at least 10,000.');
   END IF; END;/
```

2. Write a trigger which will fire before insert or update on Employee having Emp id less than equal to zero (Raise user defined exception and give appropriate message).

≫

```
CREATE TABLE Employee (
   emp_id     NUMBER PRIMARY KEY,
   emp_name   VARCHAR2(100),
   address    VARCHAR2(200)
);

-- Investment Table
CREATE TABLE Investment (
   inv_no      NUMBER PRIMARY KEY,
   inv_name    VARCHAR2(100),
   inv_date    DATE,
   inv_amount  NUMBER CHECK (inv_amount > 0),
   emp_id      NUMBER,
   FOREIGN KEY (emp_id) REFERENCES Employee(emp_id)
);

CREATE OR REPLACE TRIGGER trg_check_emp_id
BEFORE INSERT OR UPDATE ON Employee
FOR EACH ROW

DECLARE
   ex_invalid_emp_id EXCEPTION;
   PRAGMA EXCEPTION_INIT(ex_invalid_emp_id, -20002);

BEGIN
   IF :NEW.emp_id <= 0 THEN
      RAISE_APPLICATION_ERROR(-20002, 'Invalid Employee ID:
emp_id must be greater than 0.');

  END IF;
END;
/
```

SET B:

Consider the following entities and their relationships.

Bill (billno, day, tableno, total)

Menu (dish_no, dish_desc, price)

The relationship between Bill and Menu is Many to Many with quantity as descriptive attribute.

Constraint: Primary key, price should be > 0

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a trigger which will fire before insert or update on Menu having price less than or equal to zero. (Raise user defined exception and give appropriate message).

➢ CREATE TABLE Menu (

dish_no NUMBER PRIMARY KEY,

dish_desc VARCHAR2(100),

price NUMBER CHECK (price > 0)

);

-- BILL table

CREATE TABLE Bill (

billno NUMBER PRIMARY KEY,

day DATE,

tableno NUMBER,

total NUMBER

);

```
-- BILL_MENU table (junction table with quantity)
CREATE TABLE Bill_Menu (
    billno      NUMBER,
    dish_no     NUMBER,
    quantity    NUMBER,
    PRIMARY KEY (billno, dish_no),
    FOREIGN KEY (billno) REFERENCES Bill(billno),
    FOREIGN KEY (dish_no) REFERENCES Menu(dish_no)
);
CREATE OR REPLACE TRIGGER trg_check_menu_price
BEFORE INSERT OR UPDATE ON Menu
FOR EACH ROW
DECLARE
    ex_invalid_price EXCEPTION;
    PRAGMA EXCEPTION_INIT(ex_invalid_price, -20001);
BEGIN
    IF :NEW.price <= 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Invalid Price: Price
must be greater than 0.');
    END IF;
END;
/
```

2. Write a trigger which will fire before insert or update on Bill having day other than seven week days. (Raise user defined exception and give appropriate message)

```
CREATE TABLE Menu (
    dish_no     NUMBER PRIMARY KEY,
    dish_desc   VARCHAR2(100),
    price       NUMBER CHECK (price > 0)
);

-- BILL Table
CREATE TABLE Bill (
    billno      NUMBER PRIMARY KEY,
    day         VARCHAR2(10),
    tableno     NUMBER,
    total       NUMBER
);

-- BILL_MENU Table (junction table for many-to-many relationship)
CREATE TABLE Bill_Menu (
    billno    NUMBER,
    dish_no   NUMBER,
    quantity  NUMBER,
    PRIMARY KEY (billno, dish_no),
    FOREIGN KEY (billno) REFERENCES Bill(billno),
    FOREIGN KEY (dish_no) REFERENCES Menu(dish_no)
);
CREATE OR REPLACE TRIGGER trg_check_bill_day
BEFORE INSERT OR UPDATE ON Bill
FOR EACH ROW
DECLARE
    ex_invalid_day EXCEPTION;
    PRAGMA EXCEPTION_INIT(ex_invalid_day, -20002);
BEGIN
    IF UPPER(:NEW.day) NOT IN ('MONDAY', 'TUESDAY',
'WEDNESDAY', 'THURSDAY', 'FRIDAY', 'SATURDAY', 'SUNDAY')
THEN
        RAISE_APPLICATION_ERROR(-20002, 'Invalid day: Must be a
valid weekday (Monday to Sunday).');
    END IF;
END; /
```

Assignment No.7

Package

SET A:

Consider the following entities and their relationships. College (code, college_name, address)

Teacher (teacher_id, teacher_name, Qualification, specialization, salary, Desg)

Relation between Teacher and College is Many to One. Constraint: Primary Key, qualification should not be null.

Create a RDB in 3NF and write PL/SQL blocks in Oracle for the following:

1. Write a package, which consists of one procedure and one function. Pass college code as a parameter to procedure and display details of college. Write a function which will return teacher name having maximum salary.

➢ CREATE OR REPLACE PACKAGE BODY college_pkg AS

```
-- Procedure to display college details
  PROCEDURE show_college_details(p_code IN College.code%TYPE) IS
     v_name    College.college_name%TYPE;
     v_address College.address%TYPE;
  BEGIN
     SELECT college_name, address
     INTO v_name, v_address
     FROM College
     WHERE code = p_code;

     DBMS_OUTPUT.PUT_LINE('College Code : ' || p_code);
     DBMS_OUTPUT.PUT_LINE('College Name : ' || v_name);
     DBMS_OUTPUT.PUT_LINE('Address      : ' || v_address);
```

```
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No college found with code ' || p_code);
    END show_college_details;

    -- Function to return teacher name with maximum salary
    FUNCTION get_top_paid_teacher RETURN VARCHAR2 IS
        v_teacher_name Teacher.teacher_name%TYPE;
    BEGIN
        SELECT teacher_name
        INTO v_teacher_name
        FROM Teacher
        WHERE salary = (SELECT MAX(salary) FROM Teacher);

        RETURN v_teacher_name;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RETURN 'No teacher found.';
    END get_top_paid_teacher;

END college_pkg;
/
```