THE NEOTIA UNIVERSITY

DATABASE MANAGEMENT SYSTEM LAB

WORK INSTRUCTION

ASSIGNMENT NO:01

EXPERIMENT NAME:

For the given ODI database perform the following manipulations:

MATCH (match_id, team1, team2, ground, mdate, winner) {primary key match_id}
PLAYER (p_id, lname, fname, country, yborn, bplace, ftest) {primary key p_id}
BATTING (match_id, p_id, mts, order, out_type, fow, nruns, nballs, fours, sixes) {primary key match_id, p_id}
BOWLING (match_id, p_id, novers, maidens, nruns, nwickets) {primary key match_id, p_id}

a) Write appropriate DDL statements to create the above database with integrity constraints.
b) Create a table COUNTRY with c_id, c_name.
c) Add c_id as PRIMARY KEY for COUNTRY table.
d) Drop the table COUNTRY.
e) Write appropriate DML statements to insert values in all the tables under the above database.
f) Add a new column to BATTING table.
g) Delete a column from BATTING table.
h) Change any column name of a table.
i) Get the match id which was played on 27th August, 2008 at Colombo.
j) Get all match records as the column 'mdate' will be shown as "match_date".
k) Find the player id, first name and last name of all players.
l) Insert a new player Michael Clarke with player id 200311 in PLAYER table.

OBJECTIVE:

Creating, altering and dropping tables with integrity constraints.

PRINCIPLE:

a) CREATE TABLE match (match_id NUMBER(10) PRIMARY KEY, team1 VARCHAR(15), team2 VARCHAR(15), ground VARCHAR(15), mdate DATE, winner VARCHAR(15));
CREATE TABLE player (p_id NUMBER(10) PRIMARY KEY, lname VARCHAR(15), fname VARCHAR(15), country VARCHAR(15), yborn NUMBER(4), bplace VARCHAR(15), ftest NUMBER(3));
CREATE TABLE batting (match_id NUMBER(10), p_id NUMBER(10), mts NUMBER(3), order NUMBER(2), out_type CHAR(3), fow NUMBER(4), nruns NUMBER(3), nballs

NUMBER(3), fours NUMBER(2), sixes NUMBER(2), FOREIGN KEY(match_id) REFERENCES match(match_id), FOREIGN KEY(p_id) REFERENCES player(p_id));
CREATE TABLE bowling (match_id NUMBER(10), p_id NUMBER(10), novers NUMBER(2), maidens NUMBER(2), nruns NUMBER(3), nwickets NUMBER(2), FOREIGN KEY(match_id) REFERENCES match(match_id), FOREIGN KEY(p_id) REFERENCES player(p_id));

b) CREATE TABLE country (c_id NUMBER(10), c_name VARCHAR(20));

c) ALTER TABLE country ADD PRIMARY KEY(c_id);

d) DROP TABLE country;

e) INSERT INTO match VALUES (2324, 'Pakistan', 'India', 'Peshawar', 06-02-2006, 'Pakistan');

f) ALTER TABLE batting ADD COLUMN not_out INTEGER;

g) ALTER TABLE batting DROP COLUMN not_out;

h) SELECT team1 AS hometown FROM match;

i) SELECT match_id FROM match WHERE mdate = 27-08-2008 AND ground = 'Colombo';

j) SELECT match_id, team1, team2, ground, mdate AS match_date, winner FROM match;

k) SELECT p_id, fname, lname FROM player;

l) INSERT INTO player (p_id, lname, fname, country, yborn, bplace, ftest) VALUES (200311, 'Clarke', 'Michael', NULL, NULL, NULL, NULL);
INSERT INTO player VALUES (200311, 'Clarke', 'Michael', NULL, NULL, NULL, NULL);

THE NEOTIA UNIVERSITY

ASSIGNMENT NO: 02

EXPERIMENT NAME:

For the given ODI database perform the following manipulations:

MATCH (match_id, team1, team2, ground, mdate, winner) {primary key match_id}
PLAYER (p_id, lname, fname, country, yborn, bplace, ftest) {primary key p_id}
BATTING (match_id, p_id, mts, order, out_type, fow, nruns, nballs, fours, sixes) {primary key match_id, p_id}
BOWLING (match_id, p_id, novers, maidens, nruns, nwickets) {primary key match_id, p_id}

a) Find all the information about players who are from 'INDIA' and was born after 1980.
b) Find details of matches that have been played in 'AUSTRALIA'.
c) List the matches played in which INDIA or ENGLAND was team1.
d) Find ids of all players those have bowled in an ODI match.
e) Find names of teams and grounds where India has played an ODI match outside India.
f) Find ids of all players that batted in match no 2755.
g) Find the player ids of players who have made a century in each of the ODI matches 2755 and 2689.
h) Find the player ids of those players whose date of first test match (FTEST) is not given in the database.
i) Modify the bplace and ftest of player id 200311 to Sydney and 28 respectively.
j) Delete match id 2689 from MATCH table.
k) Delete bowling records of Brian Lara.

OBJECTIVE:

Retrieving and modifying data from a database.

PRINCIPLE:

a) SELECT * FROM player WHERE country = 'INDIA' AND yborn> 1980;
b) SELECT * FROM match WHERE team1 = 'AUSTRALIA';
c) SELECT * FROM match WHERE team1 = 'INDIA' OR team1 = 'ENGLAND';
d) SELECT DISTINCT (p_id) FROM bowling;
e) SELECT DISTINCT (team1, ground) FROM match WHERE team2 = 'INDIA';
f) SELECT p_id FROM batting WHERE match_id = '2755';
g) SELECT p_id FROM batting b1, batting b2 WHERE b1.match_id = '2755' AND b2.match_id = '2689' AND b1.nruns > 99 AND b2.nruns > 99;
h) SELECT p_id FROM player WHERE ftest IS NULL;
i) UPDATE player SET bplace = 'Sydney', ftest = '28' WHERE p_id = 200311;
j) DELETE FROM match WHERE match_id = '2689';
k) DELETE FROM bowling WHERE p_id = (SELECT p_id FROM player WHERE lname = 'Lara' AND fname = 'Brian');

ASSIGNMENT NO: 03

EXPERIMENT NAME:

For the given ODI database perform the following manipulations:

MATCH (match_id, team1, team2, ground, mdate, winner) {primary key match_id}
PLAYER (p_id, lname, fname, country, yborn, bplace, ftest) {primary key p_id}
BATTING (match_id, p_id, mts, order, out_type, fow, nruns, nballs, fours, sixes) {primary key match_id, p_id}
BOWLING (match_id, p_id, novers, maidens, nruns, nwickets) {primary key match_id, p_id}

a) Display the sorted list of ground names in ascending & descending order where AUSTRALIA has played as team1.
b) Find the name of all players whose last name starts with 's', third letter is 'n'.
c) Find match ids of those matches in which 'Tendulkar' batted.
d) Find the match details of those matches in which 'Dhoni' has batted.
e) Find the match ids of matches in which Sachin Tendulkar has played.
f) Find ids and scores of players who scored less than 75 but more than 50 in Colombo.

OBJECTIVE:

Retrieving data from database using IN, BETWEEN, LIKE, ORDER BY, GROUP BY and HAVING clause.

PRINCIPLE:

a) SELECT ground FROM match WHERE team1 = 'AUSTRALIA' ORDER BY ground;
SELECT ground FROM match WHERE team1 = 'AUSTRALIA' ORDER BY ground DESC;
b) SELECT fname, lname FROM player WHERE lname LIKE "s_n%";
c) SELECT match_id FROM batting WHERE p_id IN (SELECT p_id FROM player WHERE fname = 'Tendulkar');
SELECT match_id FROM batting b, player p WHERE b.p_id = p.p_id AND fname = 'Tendulkar';
d) SELECT * FROM match WHERE match_id IN (SELECT match_id FROM batting WHERE p_id IN (SELECT p_id FROM player WHERE lname = 'Dhoni'));
SELECT * FROM match m, batting b, player p WHERE m.match_id = b.match_id AND p.p_id = b.p_id AND lname = 'Dhoni';
e) SELECT match_id FROM batting b, (SELECT p_id FROM player WHERE fname like 'Sachin' AND lname like 'Tendulkar') st WHERE b.p_id = st.p_id;
f) SELECT p_id, nruns FROM batting WHERE nruns BETWEEN 51 AND 74 AND match_id IN (SELECT match_id FROM match WHERE ground = 'Colombo');

ASSIGNMENT NO: 04

EXPERIMENT NAME:

For the given ODI database perform the following manipulations:

MATCH (match_id, team1, team2, ground, mdate, winner) {primary key match_id}
PLAYER (p_id, lname, fname, country, yborn, bplace, ftest) {primary key p_id}
BATTING (match_id, p_id, mts, order, out_type, fow, nruns, nballs, fours, sixes) {primary key match_id, p_id}
BOWLING (match_id, p_id, novers, maidens, nruns, nwickets) {primary key match_id, p_id}

a) Find the number of players that bowled in the ODI match 2689.
b) Find the average batting score of all the players that batted in the ODI match 2689.
c) Find the youngest player in ODI database.
d) Find the number of players in the ODI database from each country.
e) Find the batting average of each player.
f) Find the batting average of each player from India along with their first name.
g) Find the average scores for each player when playing in Australia.

OBJECTIVE:

Use of scalar and aggregate functions.

PRINCIPLE:

a) SELECT COUNT(*) AS nbowlers FROM bowling WHERE match_id = '2689';
b) SELECT AVG(nruns) AS avgruns_2689 FROM batting WHERE match_id = '2689';
c) SELECT lname AS young_player FROM player WHERE yborn = (SELECT MAX(yborn) FROM player);
d) SELECT country, COUNT(*) AS nplayers FROM player GROUP BY country;
e) SELECT p_id, AVG(nruns) AS average FROM batting GROUP BY p_id;
f) SELECT fname, AVG(nruns) AS average FROM player p, batting b WHERE p.p_id = b.p_id AND country = 'India' GROUP BY fname;
g) SELECT p_id AS playerid, AVG(nruns) AS average FROM batting WHERE match_id IN (SELECT match_id FROM match WHERE team1 = 'Australia') GROUP BY p_id;

ASSIGNMENT NO: 05

EXPERIMENT NAME:

For the given BOOK CLUB database perform the following manipulations:

AUTHOR (**a_id**, name, city, country) {primary key a_id}
CATALOG (**b_id**, title, a_id, p_id, c_id, year, price) {primary key b_id}
PUBLISHER (**p_id**, name, city, country) {primary key p_id}
MEMBER (**m_id**, name, address, city, state, pin, phone, email) {primary key m_id}
ORDER_SUMMARY (**o_id**, m_id, order_date, amount, order_status) {primary key o_id}
CATEGORY (**c_id**, description) {primary key c_id}
ORDER_DETAILS (o_id, b_id, quantity) {foreign key o_id, b_id}

a) Get the title and publisher names of all books. (EQUI JOIN)
b) Get the title and publisher names of all books that are priced above 1000. (NON-EQUI JOIN / GREATER-THAN JOIN)
c) Get the title, author name, country and price of all the books with India-based authors and price less than 500. (NON_EQUI JOIN / LESS_THAN JOIN)
d) Get the details of all the books with their categorical descriptions that are priced above 1000. (NATURAL JOIN)
e) Find out the titles that have the same price. (SELF JOIN)
f) Get the title, author name, publisher name and category name of all books that are published after 2008.
g) Get the details of order which was ordered by a member. (INNER JOIN)
h) By a SQL statement, show how LEFT OUTER JOIN works.
i) By a SQL statement, show how RIGHT OUTER JOIN works.
j) By a SQL statement, show how FULL OUTER JOIN works.
k) By a SQL statement, show how CARTESIAN PRODUCT works.
l) Get the details of all authors and publishers in India ordered by name.

OBJECTIVE:

Retrieving data from a database using Equi , Non Equi , Outer and Self Join.

PRINCIPLE:

a) SELECT c.title, p.name FROM catalog c, publisher p WHERE c.p_id = p.p_id;
b) SELECT c.title, p.name FROM catalog c, publisher p WHERE c.p_id = p.p_id AND c.price> 1000;
c) SELECT c.title, a.name, a.country, c.price FROM author a, catalog c WHERE c.a_id = a.a_id AND a.country = 'India' AND c.price< 500;
d) SELECT cl.b_id, cl.title, cl.a_id, cl.p_id, cl.c_id, cl.year, cl.price, cg.description FROM catalog cl, category cg WHERE cl.c_id = cg.c_id AND cl.price> 1000;

e) SELECT DISTINCT c1.title, c1.price FROM catalog c1, catalog c2 WHERE c1.price = c2.price AND c1.b_id <> c2.b_id ORDER BY c1.price;
f) SELECT cl.title, a.name, p.name, cg.description FROM catalog cl, author a, publisher p, category cg WHERE cl.a_id = a.a_id AND cl.p_id = p.p_id AND cl.c_id = cg.c_id AND cl.year>2008;
g) SELECT m.name, os.o_id, os.order_date, os.amount, os.order_status FROM member m, order_summaryos WHERE m.m_id = os.m_id;
h) SELECT m.name, os.o_id, os.amount FROM member LEFT OUTER JOIN order_summary ON m.m_id = os.m_id;
i) SELECT m.name, os.o_id, os.amount FROM member RIGHT OUTER JOIN order_summary ON m.m_id = os.m_id;
j) SELECT m.name, os.o_id, os.amount FROM member FULL OUTER JOIN order_summary ON m.m_id = os.m_id;
k) SELECT catalog.*, author.* FROM catalog, author;
l) SELECT name, city, country FROM author WHERE country = 'India' UNION SELECT name, city, country FROM publisher WHERE country = 'India' ORDER BY 1;

THE NEOTIA UNIVERSITY

ASSIGNMENT NO: 06

EXPERIMENT NAME:

For the given ODI database perform the following manipulations:

MATCH (match_id, team1, team2, ground, mdate, winner) {primary key match_id}
PLAYER (p_id, lname, fname, country, yborn, bplace, ftest) {primary key p_id}
BATTING (match_id, p_id, mts, order, out_type, fow, nruns, nballs, fours, sixes) {primary key match_id, p_id}
BOWLING (match_id, p_id, novers, maidens, nruns, nwickets) {primary key match_id, p_id}

a) Find player ids of players who have scored more than 30 in every ODI match that they have batted.
b) List the attributes of BATTING separated by comma.
c) Find match ids of those matches in which 'Lee' bowled.
d) Find ids of players that have both bowled and batted in the ODI match 2689.
e) Find ids of players that have either bowled or batted in the ODI match 2689.
f) Find ids of players that have batted in match 2689 but have not bowled.
g) Find the senior player in ODI database.

OBJECTIVE:

Using sub queries, rowid and rownum for retrieving data.

PRINCIPLE:

a) SELECT p_id AS playerid FROM batting b1 WHERE NOT EXISTS (SELECT * FROM batting b2 WHERE b1.p_id = b2.p_id AND nruns<30);
b) SELECT match_id ||','|| p_id ||','|| mts ||','|| order||','|| out_type ||','|| fow ||','|| nruns ||','|| nballs ||','|| fours ||','|| sixes FROM batting;
c) SELECT b.match_id FROM player p, bowling b WHEREb.p_id = p.p_id AND p.lname = 'Lee';
d) SELECT p_idFROM batting WHERE match_id = '2689' AND p_id IN (SELECT p_id FROM bowling WHERE match_id = '2689');
   SELECT p_id FROM batting WHERE match_id = '2689' INTERSECT (SELECT p_id FROM bowling WHERE match_id = '2689');
e) SELECT p_id FROM batting WHERE match_id = '2689' UNION (SELECT p_id FROM bowling WHERE match_id = '2689');
f) SELECT p_id FROM batting WHERE match_id = '2689' AND p_id NOT IN (SELECT p_id FROM bowling WHERE match_id = '2689');
   SELECT p_id FROM batting WHERE match_id = '2689' EXCEPT (SELECT p_id FROM bowling WHERE match_id = '2689');
g) SELECT lname AS senior_player FROM player WHERE yborn = (SELECT MIN(yborn) FROM player);

ASSIGNMENT NO: 07

EXPERIMENT NAME:

For the given ODI database perform the following manipulations:

MATCH (match_id, team1, team2, ground, mdate, winner) {primary key match_id}
PLAYER (p_id, lname, fname, country, yborn, bplace, ftest) {primary key p_id}
BATTING (match_id, p_id, mts, order, out_type, fow, nruns, nballs, fours, sixes) {primary key match_id, p_id}
BOWLING (match_id, p_id, novers, maidens, nruns, nwickets) {primary key match_id, p_id}

a) Create a VIEW having p_id, fname as firstname, lname as lastname, country, novers, nwickets where match_id = 2755.
b) Find the name of players that have taken 2wickets or more in ODI match 2755.
c) Create a VIEW having match_id, team1, team2, ground and mdate.
d) Insert data "India" as team1, "Auatralia" as team2, "Mohali" as ground with match_id 290782 into this view.
e) Update "290782(match is)'s" ground to "Eden Gardens" in the created view.
f) Delete the record of 290782 from the view.
g) Drop the created views.
h) Create an index on table PLAYER by taking p_id, lname, fname, country (in descending order), yborn.
i) Create an index on table MATCH by taking match_id, team1, team2, mdate (in descending order), ground.
j) Delete one of the created indexes.

OBJECTIVE:

Use of views, indexes and sequences.

PRINCIPLE:

a) CREATE VIEW M2755 AS SELECT p_id, fname 'firstname', lname 'lastname', country, novers, nwickets FROM player p, bowling b WHERE p.p_id = b.p_id AND b.match_id = 2755;
b) SELECT fname, lname FROM M2755 WHERE nwickets ≥ 2;
c) CREATE VIEW AB23 AS SELECT match_id, team1, team2, ground, mdate FROM match;
d) INSERT INTO AB23 VALUES ('290782', 'India', 'Australia', 'Mohali', NULL);
e) UPDATE AB23 SET ground = 'Eden Gardens' WHERE match_id = 290782;
f) DELETE FROM M2689 WHERE p_id = '290782';
g) DROP VIEW M2689;
h) CREATE UNIQUE INDEX xyz ON player (p_id, lname, fname, country DESC, yborn);
i) CREATE UNIQUE INDEX abc ON match (match_id, team1, team2, mdate DESC, ground);
j) DROP INDEX abc;

SSIGNMENT NO: 08

EXPERIMENT NAME:

For the given ODI database perform the following manipulations:

MATCH (match_id, team1, team2, ground, mdate, winner) {primary key match_id}
PLAYER (p_id, lname, fname, country, yborn, bplace, ftest) {primary key p_id}
BATTING (match_id, p_id, mts, order, out_type, fow, nruns, nballs, fours, sixes) {primary key match_id, p_id}

k) Grant the SELECT authority on the MATCH table to all users.
l) Grant the SELECT, DELETE & UPDATE authority on PLAYER table to user 'ALEX'.
m) Grant the SELECT, DELETE & UPDATE authority with the capability to grant those privileges to other users on PLAYER table to user 'ALEX'.
n) Grant ALL privileges on MATCH table to user 'ROY'.
o) Give the system privileges for creating tables and views to 'ROY'.
p) Grant the UPDATE authority on the FOURS column of the BATTING to user 'ROY'.
q) Revoke the system privileges for creating tables from 'ROY'.
r) Revoke the SELECT privilege on PLAYER table from 'ALEX'.
s) Revoke the UPDATE privilege on PLAYER table from all users.
t) Remove ALL privileges on MATCH table from user 'ROY'.
u) Remove DELETE and UPDATE authority on the COUNTRY & YBORN columns of the PLAYER table from user 'ALEX'.

OBJECTIVE:

Giving privileges to database users using GRANT & REVOKE commands.

PRINCIPLE:

```
k) GRANT SELECT ON match TO public;
l) GRANT SELECT, DELETE, UPDATE ON player TO alex;
m) GRANT SELECT, DELETE, UPDATE ON player TO alex WITH GRANT OPTION;
n) GRANT ALL ON match TO roy;
o) GRANT CREATE TABLE, CREATE VIEW TO roy;
p) GRANT UPDATE (fours) ON batting TO roy;
q) REVOKE CREATE TABLE FROM roy;
r) REVOKE SELECT ON player FROM alex;
s) REVOKE UPDATE ON player FROM public;
t) REVOKE ALL ON match FROM roy;
u) REVOKE DELETE, UPDATE (country, yborn) ON player FROM alex;
```

ASSIGNMENT NO: 09

EXPERIMENT NAME:

1) Write a PL / SQL code block to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in an empty table named AREAS with radius and area as attributes.
2) Write PL / SQL code to calculate sum of digits of a number.
3) Write PL / SQL code to calculate sum of natural series.
4) Write PL / SQL code for inverting a number 8975 to 5798.

OBJECTIVE:

Perform coding in PL/SQL, using output from server.

PRINCIPLE:

1)
```
CREATE TABLE areas (radius NUMBER(5), area NUMBER(14, 2));
SET SERVEROUTPUT ON;
DECLARE
        piconstant number(4, 2) := 3.14 ;
        radius number(5);
        area number(14, 2);

BEGIN
        radius:= .3;
                WHILE RADIUS ≤ 7
                LOOP
                        area := pi * power(radius, 2);
                        INSERT INTO areas VALUES (radius, area);
                        radius := radius + l;
                END LOOP;
END;
```

2)
```
SET SERVEROUTPUT ON;
 DECLARE
        given_number number(8);
        sum_of_digit number(8):=0;
        rem number(8);
BEGIN
        given_number:=given_number || &given_number;
        whilegiven_number>0
        LOOP
                rem: =mod (given_number,10);
                sum of digit:=sum of digit+rem;
```

```
            given_number:=given_number/ 10;
        END LOOP;
        dbms_output.put_line('The sum of digit is: '||sum_of_digit);
    END;
3)  SET SERVEROUTPUT ON;
    DECLARE
        term number (10);
        sum1 number ( 9) ;
    BEGIN
        Sum1:=0;
        term:=&term;
        fori in 1.. term
        LOOP
            sum1 :=sum1+i;
        END LOOP;
        dbms_output.put_line  (' sum = '||sum1);
    END;
4)  SET SERVEROUTPUT ON;
    DECLARE
        given_number varchar(5) := '5639';
        str_length number(2);
        inverted_number varchar(5);
    BEGIN
        str_length := length(given_number);
        FOR cntrIN REVERSE l..str_length
        LOOP
            inverted_number := inverted_number || substr(given_number, cntr, l);
        END LOOP;
        dbms_output.put_line ('The Given number is' || given_number);
        dbms_output.put_line ('The Invertednumberis' || inverted_number);
    END;
```

ASSIGNMENT NO: 10

EXPERIMENT NAME:

For a table Employee (Empnovarchar(8), Salary number (10)), Write a PL/SQL code to create a Procedure that will find salary of a specific employee & write a Function for the same purpose.

OBJECTIVE:

Data manipulation using stored procedures & functions in PL/ SQL.

PRINCIPLE:

```
i) create or replace procedure findsalary(empnol in varchar2,salaryl out number) is salary2
number(10);
        begin
                select salary into salary2 from employee where empno = empnol;
                salaryl:=salary2;
        end;

        setserveroutput on;
        declare
                empnol varchar2(8);
                salaryl varchar2(10);
        begin
                empnol:=&empnol;
                findsalary(empnol,salaryl);
                dbms_output.put_line(' The salary of the employee is '||salaryl);
        end;
  ii)  create or replace function findsalaryf(empnol varchar2)return number as salary1
number (10);
        begin
                select salary into salaryfromemployeewhereempno=empno1;
                return (salaryl);
        end;

        setserveroutput on;
        declare
                empnol varchar2(8);
                salaryl varchar2(10);
        begin
                empnol:=&empnol;
                salaryl:=findsalaryf(empnol);
                dbms_output.put_line(' the salary of the employee is' ||salaryl);
        end;
```

ASSIGNMENT NO: 11

EXPERIMENT NAME:

Write a PL/SQL block of code that first withdraws an amount of Rs.1,000. Then deposits an amount of Rs.1,40,000. Update the current balance. Then check to see that the current balance of all the accounts in the bank does not exceed Rs.2,00,000. If the balance exceeds Rs.2,00,000 then undo the deposit just made.

OBJECTIVE:

Perform Oracle defined and User defined Exception handling in PL/SQL.

PRINCIPLE:

```
SET SERVEROUTPUT ON;

DECLARE
        mBAL number(8,2);
BEGIN
        INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR,
    AMT, BALANCE)
                VALUES ('TIOO', 'CAIO', '04-JUL-2004', 'C', 'Telephone Bill', 'W', 1000, 31000);
        UPDATE ACCT_MSTR SET CURBAL = CURBAL – 1000 WHERE ACCT_NO = 'CA10';
        SAVEPOINT no_update;
        INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR,
AMT, BALANCE)
                VALUES ('TIOI', 'CAIO', '04-JUL-2004', 'C', 'Deposit', 'D', 140000, 171000);
        UPDATE  ACCT_MSTR SET CURBAL = CURBAL + 140000 WHERE ACCT_NO = 'CA10';
        SELECT SUM(CURBAL) INTO mBAL FROM ACCT_MSTR;
        IF mBAL> 200000 THEN
                ROLLBACK To SAVEPOINT no_update;
        END IF;
        COMMIT;
END;
```

ASSIGNMENT NO: 12

EXPERIMENT NAME:

For the following table *Employee* solve the problem:

Column name	Data type	Size	Attributes
Emp_code	Varchar2	6	Primary Key
Emp_name	Varchar2	25	
Job	Varchar2	25	
Salary	Number	8,2	

The HRD manager has decided to raise the salary of employees by 0.15. Write a PL/SQL block to accept the employee number and update the salary of that employee. Display appropriate message based on successful modification of the record in the *Employee* table.

OBJECTIVE:

Data handling by implicit & explicit cursors in PL/SQL.

PRINCIPLE:

```
        SET SERVEROUTPUT ON;

        EXEC SQL
                DECLARE sal_update CURSOR FOR
                        UPDATE employee SET salary = salary * 0.15 WHERE emp_code=
    &emp_code;

                        IF SQL%FOUND THEN
                                dbms_output.put_line('Employee Record Modified Successfully');
                        ELSE
                        dbms_output.put_line('Employee No. Does not Exist');
                        END IF;

        END EXEC;
```

ASSIGNMENT NO: 13

EXPERIMENT NAME:

Create a transparent audit system for a table client- master. The system keeps track of the records that are being deleted or updated. The functionality being when a record is deleted or modified the original record details and the date ofoperation is stored in the audit-client table, and then delete or update is allowed to go through. Write a trigger for the above problem.

Client-master (client_no, name, address, city, bal_due)

Audit-client (c1ient_no, name, bal_due, operation, user_id, o_date)

Operation: Operation performed on the client-master table.

o_date : The date when the operation was performed.

user_id : The name *of* the user performing the operation.

OBJECTIVE:

Data manipulation Using trigger in PL/SQL.

PRINCIPLE:

This trigger is fired when an update or delete is fired on the table client_master. The trigger first checks for the operation being performed on the table. Then depending on the operation being performed, a variable is assigned the value 'update' or 'delete'. Previous values of the modified record of the table client master are stored into appropriate variables declared. The contents of these variables are then inserted into the audit table auditclient.

```
 CREATE TRIGGER audit_trail
AFTER UPDATE OR DELETE ON client_master

FOR EACH ROW

 DECLARE

           /* The value in the oper  variable will be inserted into the operation field in
   the auditclient table */
     oper varchar2(8);

           /* These variables will hold the previous values of client_no, name and
       bal_due*/
```

```
            client_no, varchar2(6);
            name varchar2(20);
            bal_due number(l0, 2);

BEGIN                                              .
                               /* if the records are updated in client_master table then operis
                               set to 'update'. */ ,
            IF updating THEN
                        oper := 'update';
            END IF;

                            /* if the records are deleted in client _master table then operis set to
                               'delete' */
            IF deleting THEN
                        oper := 'delete';
            END IF;

                            /* Store :old.client_no, :old.name, and :old.bal_due into client_no,
                               name and bal_due. These variables can then be used to insert
                               data into the auditclient table */

                  client_no := :old.client_no ;
                  name := :old.name ;
                  bat_due := :old.bal_due ;
                  INSERT INTO auditclientVALUES  (client_no, name, bal_due, oper, user,
                                    sysdate);

      END;
```