CAT – II Lab
Database Technologies – ITA5008
SLOT- B2+TB2

Name: Kamran Ansari

Reg No: 22MCA0223

Question 2

2.1 Write a PL / SQL program to check whether the given number is prime or not.

```
CREATE OR REPLACE PROCEDURE ISPRIME (
    NUM NUMBER
) AS
    I NUMBER;
BEGIN
    I := 2;
    LOOP
        EXIT WHEN I >= NUM;
        IF REMAINDER(NUM, I) = 0 THEN
            DBMS_OUTPUT.PUT_LINE(NUM
                || ' IS NOT PRIME');
            RETURN;
        END IF;
        I := I + 1;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(NUM
        || ' IS PRIME');
END;
/

EXECUTE ISPRIME(&NUMBER);
```

```
SQL> EXECUTE ISPRIME(&NUMBER);
Enter value for number: 32

PL/SQL procedure successfully completed.

SQL> set serveroutput on;
SQL> EXECUTE ISPRIME(&NUMBER);
Enter value for number: 33
33 IS NOT PRIME

PL/SQL procedure successfully completed.

SQL> EXECUTE ISPRIME(&NUMBER);
Enter value for number: 44
44 IS NOT PRIME

PL/SQL procedure successfully completed.

SQL> EXECUTE ISPRIME(&NUMBER);
Enter value for number: 17
17 IS PRIME

PL/SQL procedure successfully completed.
```

```
SQL> EXECUTE ISPRIME(&NUMBER);
Enter value for number: 71
71 IS PRIME

PL/SQL procedure successfully completed.

SQL> EXECUTE ISPRIME(&NUMBER);
Enter value for number: 2
2 IS PRIME

PL/SQL procedure successfully completed.
```

2.2 Create tables identifying the primary keys and foreign keys
Insert necessary tuples into the tables
STUDENT (StudentID, FirstName, LastName, Age, Branch, Email id)

```
CREATE TABLE STUDENT(
    STUDENTID VARCHAR(20),
    FIRSTNAME VARCHAR(50),
    LASTNAME VARCHAR(50),
    AGE NUMBER,
    BRANCH VARCHAR(20),
    EMAILID VARCHAR(50),
    CONSTRAINT STUDENT_STUDENTID_PK PRIMARY KEY(STUDENTID)
);
```

```
SQL> desc student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENTID                                 NOT NULL VARCHAR2(20)
 FIRSTNAME                                          VARCHAR2(50)
 LASTNAME                                           VARCHAR2(50)
 AGE                                                NUMBER
 BRANCH                                             VARCHAR2(20)
 EMAILID                                            VARCHAR2(50)
```

```
SET LINESIZE 1000

COLUMN STUDENTID FORMAT A10;

COLUMN FIRSTNAME FORMAT A10;

COLUMN LASTNAME FORMAT A10;

COLUMN AGE FORMAT 99;
```

```
COLUMN BRANCH FORMAT A6;

COLUMN EMAILID FORMAT A15;

INSERT INTO STUDENT VALUES(
    '22001',
    'fn1',
    'ln1',
    21,
    'MCA',
    'fn1@gmail.com'
);

INSERT INTO STUDENT VALUES(
    '22002',
    'fn2',
    'ln2',
    20,
    'CSE',
    'fn2@gmail.com'
);

INSERT INTO STUDENT VALUES(
    '22003',
    'fn3',
    'ln3',
    18,
    'BCA',
    'fn3@gmail.com'
```

```
);

INSERT INTO STUDENT VALUES(
    '22004',
    'fn4',
    'ln4',
    22,
    'MCA',
    'fn4@gmail.com'
);

INSERT INTO STUDENT VALUES(
    '22005',
    'fn5',
    'ln5',
    18,
    'CSE',
    'fn5@gmail.com'
);

SELECT
    *
FROM
    STUDENT;
```

```
SQL> SELECT
  2      *
  3  FROM
  4      STUDENT;

STUDENTID  FIRSTNAME  LASTNAME    AGE BRANCH EMAILID
---------- ---------- ---------- --- ------ ---------------
22001      fn1        ln1         21 MCA    fn1@gmail.com
22002      fn2        ln2         20 CSE    fn2@gmail.com
22003      fn3        ln3         18 BCA    fn3@gmail.com
22004      fn4        ln4         22 MCA    fn4@gmail.com
22005      fn5        ln5         18 CSE    fn5@gmail.com
```

(A) Convert STUDENT FirstName into uppercase whenever an STUDENT record is inserted or updated. Trigger to fire before the insert or update.

```
CREATE OR REPLACE TRIGGER CONVERT_FIRST_NAME_TO_UPPERCASE BEFORE
    INSERT OR UPDATE ON STUDENT FOR EACH ROW
DECLARE
    ROWCOUNT INT;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Trigger Fired!');
    DBMS_OUTPUT.PUT_LINE('Converting '
        || :NEW.FIRSTNAME
        || ' to '
        || UPPER( :NEW.FIRSTNAME ));
    :NEW.FIRSTNAME := UPPER( :NEW.FIRSTNAME );
END;
/
INSERT INTO STUDENT VALUES(
    '22006',
    'lower',
    'lastname',
    25,
    'BCA',
    'lower@gmail.com'
);

SELECT
    *
FROM
    STUDENT;
```

```
SQL> INSERT INTO STUDENT VALUES(
  2       '22006',
  3       'lower',
  4       'lastname',
  5       25,
  6       'BCA',
  7       'lower@gmail.com'
  8  );
Trigger Fired!
Converting lower to LOWER

1 row created.

SQL> SELECT *FROM STUDENT;

STUDENTID  FIRSTNAME  LASTNAME    AGE BRANCH EMAILID
---------- ---------- ---------- --- ------ ---------------
22001      fn1        ln1         21 MCA    fn1@gmail.com
22002      fn2        ln2         20 CSE    fn2@gmail.com
22003      fn3        ln3         18 BCA    fn3@gmail.com
22004      fn4        ln4         22 MCA    fn4@gmail.com
22005      fn5        ln5         18 CSE    fn5@gmail.com
22006      LOWER      lastname    25 BCA    lower@gmail.com

6 rows selected.
```

(B) To write a Cursor to display the list of STUDENTS who are enrolled in CSE and MCA branch.

```
SELECT
    *
FROM
    STUDENT;


DECLARE
    CURSOR MCA_CSE_BRANCH_STUDENT IS
        SELECT
            *
        FROM
            STUDENT
        WHERE
            BRANCH IN ('MCA',
            'CSE');
    I NUMBER;
BEGIN
    I := 1;
    FOR STUDENT IN MCA_CSE_BRANCH_STUDENT LOOP
        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT_LINE('Student '
            || I
            || '-');
        DBMS_OUTPUT.PUT_LINE('ID: '
            || STUDENT.STUDENTID);
        DBMS_OUTPUT.PUT_LINE('First Name: '
            || STUDENT.FIRSTNAME);
        DBMS_OUTPUT.PUT_LINE('Last Name: '
```

```
            || STUDENT.LASTNAME);
        DBMS_OUTPUT.PUT_LINE('Age: '
            || STUDENT.AGE);
        DBMS_OUTPUT.PUT_LINE('Branch: '
            || STUDENT.BRANCH);
        DBMS_OUTPUT.PUT_LINE('Email Id: '
            || STUDENT.EMAILID);
        I := I + 1;
    END LOOP;
END;
/
```

```
SQL> SELECT
  2      *
  3  FROM
  4      STUDENT;

STUDENTID  FIRSTNAME  LASTNAME    AGE BRANCH EMAILID
---------- ---------- ---------- --- ------ ---------------
22001      fn1        ln1         21 MCA    fn1@gmail.com
22002      fn2        ln2         20 CSE    fn2@gmail.com
22003      fn3        ln3         18 BCA    fn3@gmail.com
22004      fn4        ln4         22 MCA    fn4@gmail.com
22005      fn5        ln5         18 CSE    fn5@gmail.com
22006      LOWER      lastname    25 BCA    lower@gmail.com

6 rows selected.
```

```
 30           I := I + 1;
 31       END LOOP;
 32  END;
 33  /
Student 1-
ID: 22001
First Name: fn1
Last Name: ln1
Age: 21
Branch: MCA
Email Id: fn1@gmail.com
Student 2-
ID: 22002
First Name: fn2
Last Name: ln2
Age: 20
Branch: CSE
Email Id: fn2@gmail.com
```

```
Student 3-
ID: 22004
First Name: fn4
Last Name: ln4
Age: 22
Branch: MCA
Email Id: fn4@gmail.com
Student 4-
ID: 22005
First Name: fn5
Last Name: ln5
Age: 18
Branch: CSE
Email Id: fn5@gmail.com
```

2.3 Write the PL/SQL programs to create the procedure to find Fibonacci series.

```
CREATE OR REPLACE PROCEDURE FIBONACCI (
    NUM_OF_TERMS NUMBER
) AS
    COUNTER NUMBER;
    A       NUMBER;
    B       NUMBER;
    TEMP    NUMBER;
BEGIN
    A := 0;
    B := 1;
    DBMS_OUTPUT.PUT_LINE('Fibonacci Sequence of '
        || NUM_OF_TERMS
        || ' terms is');
    DBMS_OUTPUT.PUT_LINE(A);
    DBMS_OUTPUT.PUT_LINE(B);
    COUNTER := 2;
    LOOP
        TEMP := B;
        B := A + B;
        A := TEMP;
        DBMS_OUTPUT.PUT_LINE(B);
        COUNTER := COUNTER + 1;
        EXIT WHEN COUNTER >= NUM_OF_TERMS;
    END LOOP;
END;
/
```

```
EXECUTE FIBONACCI(&NUMBER_OF_TERMS);
```

```
SQL> EXECUTE FIBONACCI(&NUMBER_OF_TERMS);
Enter value for number_of_terms: 4
Fibonacci Sequence of 4 terms is
0
1
1
2

PL/SQL procedure successfully completed.
```

```
SQL> EXECUTE FIBONACCI(&NUMBER_OF_TERMS);
Enter value for number_of_terms: 10
Fibonacci Sequence of 10 terms is
0
1
1
2
3
5
8
13
21
34

PL/SQL procedure successfully completed.
```

```
SQL> EXECUTE FIBONACCI(&NUMBER_OF_TERMS);
Enter value for number_of_terms: 15
Fibonacci Sequence of 15 terms is
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377

PL/SQL procedure successfully completed.
```