SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Ex.no: 15	Frame & execute the appropriate PL/SQL cursors & exceptional handling for student management.

Aim: To frame & execute the appropriate PL/SQL cursors & exceptional handling for student management system.

PL/SQL cursor: It is used to refer to a program to fetch & process the rows returned by SQL statement, one at a time

PL/SQL exceptional handling:

An error occurs during the program execution is called exception in SQL. PL/SQL facilitates programers to catch such conditions using exception block in the program & an appropriate action is taken against the error condition.

```
mysql> CREATE TABLE students (
    ->     student_id INTEGER PRIMARY KEY,
    ->     name VARCHAR(50) NOT NULL,
    ->     email VARCHAR(100) UNIQUE NOT NULL,
    ->     phone VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.13 sec)

mysql> INSERT INTO students (student_id, name, email, phone)
    -> VALUES (1, 'John Doe', 'jdoe@example.com', '555-1234');
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> INSERT INTO students (student_id, name, email, phone)
    -> VALUES (2, 'Jane Smith', 'jsmith@example.com', '555-5678');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> CREATE TABLE courses (
    ->     course_id INTEGER PRIMARY KEY,
    ->     name VARCHAR(50) NOT NULL,
    ->     description VARCHAR(200),
    ->     instructor VARCHAR(50) NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO courses (course_id, name, description, instructor)
    -> VALUES (101, 'Intro to Computer Science', 'An introduction to program
ming concepts and languages', 'Dr. A. Turing');
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> INSERT INTO courses (course_id, name, description, instructor)
    -> VALUES (102, 'Data Structures and Algorithms', 'A study of common dat
a structures and algorithms', 'Dr. G. Dijkstra');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> CREATE TABLE enrollments (
    ->     enrollment_id INTEGER PRIMARY KEY,
    ->     course_id INTEGER NOT NULL,
    ->     student_id INTEGER NOT NULL,
    ->     grade VARCHAR(2),
    ->     FOREIGN KEY (course_id) REFERENCES courses(course_id),
    ->     FOREIGN KEY (student_id) REFERENCES students(student_id)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Now, create a table : students

create table students (student_id integer primary key, name varchar (20) not null, email varchar (20) unique not null, phone varchar (20));

[Insert values into the table]

Now create a table : courses

create table courses (course_id integer primary key, name varchar (20) not null, description varchar (200), instructor (50) not null);

[Insert values into the table]

Now create a table : enrollments

create table enrollments (enrollment_id integer primary key, course_id integer not null, student_id integer not null, grade varchar (20), Foreign key (course_id) references courses (course_id), Foreign key (student_id) references students (student_id));

```
mysql> INSERT INTO enrollments (enrollment_id, course_id, student_id, grade)
    -> VALUES (1, 101, 1, 'A');
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> INSERT INTO enrollments (enrollment_id, course_id, student_id, grade)
    -> VALUES (2, 101, 2, 'B');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO enrollments (enrollment_id, course_id, student_id, grade)
    -> VALUES (3, 102, 1, 'B');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> INSERT INTO enrollments (enrollment_id, course_id, student_id, grade)
    -> VALUES (4, 102, 2, 'A');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from students;
+------------+------------+--------------------+----------+
| student_id | name       | email              | phone    |
+------------+------------+--------------------+----------+
|          1 | John Doe   | jdoe@example.com   | 555-1234 |
|          2 | Jane Smith | jsmith@example.com | 555-5678 |
+------------+------------+--------------------+----------+
2 rows in set (0.00 sec)

mysql> select * from courses;
+-----------+---------------------------+-----------------------------------------------------+--------------------+
| course_id | name                      | description                                         | instructor         |
+-----------+---------------------------+-----------------------------------------------------+--------------------+
|       101 | Intro to Computer Science | An introduction to programming concepts and languages | Dr. A. Turing      |
|       102 | Data Structures and Algorithms | A study of common data structures and algorithms | Dr. G. Dijkstra    |
+-----------+---------------------------+-----------------------------------------------------+--------------------+
2 rows in set (0.00 sec)

mysql> select * from enrollments;
+---------------+-----------+------------+-------+
| enrollment_id | course_id | student_id | grade |
+---------------+-----------+------------+-------+
|             1 |       101 |          1 | A     |
|             2 |       101 |          2 | B     |
|             3 |       102 |          1 | B     |
|             4 |       102 |          2 | A     |
+---------------+-----------+------------+-------+
4 rows in set (0.00 sec)
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

[Insert values into the table]

```
select * from students;

select * from courses;

select * from enrollments;
```

```
DECLARE
   course_id NUMBER := 101;
   total_enrollments NUMBER := 0;
   student_name VARCHAR2(50);
   student_email VARCHAR2(100);
   CURSOR c_enrolled_students IS
      SELECT s.name, s.email
      FROM students s, enrollments e
      WHERE s.student_id = e.student_id
      AND e.course_id = course_id;
BEGIN
   -- Count the total number of students enrolled in the course
   SELECT COUNT(*) INTO total_enrollments
   FROM enrollments
   WHERE course_id = course_id;

   -- Print the total number of enrollments to the console
   DBMS_OUTPUT.PUT_LINE('Total enrollments: ' || total_enrollments);
```

```
   -- Print the names and email addresses of all enrolled students to the co
   OPEN c_enrolled_students;
   LOOP
      FETCH c_enrolled_students INTO student_name, student_email;
      EXIT WHEN c_enrolled_students%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(student_name || ' (' || student_email || ')');
   END LOOP;
   CLOSE c_enrolled_students;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No data found');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END
```


Declare

course_id number := 101;

total_enrollments number := 0;

student_name varchar (20);

student_email varchar (100);

CURSOR C_enrolled_students IS

select S.name, S.email from students S, enrollments e where S.student_id = e.student_id and e.course_id = course_id;

Begin

select count (*) into total_enrollments from enrollments where course_id = course_id;

DBMS_output.put_line ("Total enrollments : ' || total_enrollments);

open C_enrolled_students;

LOOP

Fetch C_enrolled_students into student_name, student_email;

OUTPUT:

```
Total enrollments: 2
John Doe (jdoe@example.com)
Jane Smith (jsmith@example.com)
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

```
    Exit when c_enrolled_students%NOT FOUND;
    DBMS_output.Put_line (student_name || 'c' || student_
email || ')');
    END LOOP;
    close c_enrolled_students;
Exception
    when no_data_found then
        DBMS_output.put_line('no data found');
    when others then
        DBMS_output.put_line ('An error occured:' || SQLERRM);
END.
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Result: Hence appropriate PL/SQL cursors & exceptional handling for student management database has been framed & executed successfully.