|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Faculty of Information Technology** | | | | | | | | | |
| I declare that I am familiar with, and will abide to the Examination rules of CTU              **Signature** | **SUBJECT NAME: Business Programming**  **SUBJECT CODE:** PRG522 | | | | | | | | |
| **Formative Assessment 2 Duration**: 36 Days  **Date**:  22-09-2023  **Total Marks**: 100  **Total pages**: - | | | | **Examiner**:  Mr Junior Manganyi  **Moderator:** | | | | |
| **Student number** | | | | | | | | |
| 2 | 0 | 2 | 3 | 1 | 4 | 0 | 5 |  |
| **Surname**:  Moshoeshoe | | | | **Initials**: Leduma Tlotliso | | | / | % |
|  |  |  |  |  |  |  |  |  |  |

**Question 1:**

**Code:**

-- Create the CTU\_DB database

CREATE DATABASE IF NOT EXISTS CTU\_DB;

USE CTU\_DB;

-- Create the Students table

CREATE TABLE Students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(255),

last\_name VARCHAR(255),

age INT,

email\_address VARCHAR(255),

enrolled\_flag INT

);

-- Create the Courses table

CREATE TABLE Courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(255)

);

-- Create the Enrollments table

CREATE TABLE Enrollments (

enrollment\_id INT PRIMARY KEY,

student\_id INT,

course\_id INT,

FOREIGN KEY (student\_id) REFERENCES Students(student\_id),

FOREIGN KEY (course\_id) REFERENCES Courses(course\_id)

);

-- Insert dummy data into Students

INSERT INTO Students (student\_id, first\_name, last\_name, age, email\_address, enrolled\_flag)

VALUES

(1, 'John', 'Doe', 22, 'john@doe.com', 1),

(2, 'Mac', 'Tastic', 23, 'Mac@tastic.com', 1),

(3, 'Johnson', 'Nate', 23, 'johnson@nate.com', 1),

(4, 'Michael', 'Phelps', 25, 'michael@phelps.com', 1),

(5, 'Lauriston', 'Barends', 26, 'lauriston@barends.com', 1),

(6, 'Pieter', 'Cordier', 26, 'pieter@cordier.com', 1),

(7, 'Orefemetse', 'Botlhoko', 27, 'orefemetse@botlhoko.com', 1),

(8, 'Bokang', 'Monye', 28, 'bokang@monye.com', 1),

(9, 'Melvin', 'Abrahams', 29, 'Melvin@abrahams.com', 1),

(10, 'Lebogang', 'Kalamore', 22, 'lebogang@kalamore.com', 1);

-- Insert dummy data into Courses

INSERT INTO Courses (course\_id, course\_name)

VALUES

(101, 'Database Fundamentals'),

(102, 'SQL Mastery'),

(103, 'Computer Architecture'),

(104, 'Digital Literacy Proficiency'),

(105, 'Principles of Program and Design'),

(106, 'Network Architecture'),

(107, 'Programming with Python'),

(108, 'Robotics Development'),

(109, 'Cloud Fundamentals'),

(110, 'Core Web Development');

-- Insert dummy data into Enrollments

INSERT INTO Enrollments (enrollment\_id, student\_id, course\_id)

VALUES

(1001, 1, 101),

(1002, 2, 102),

(1003, 3, 103),

(1004, 4, 104),

(1005, 5, 105),

(1006, 6, 106),

(1007, 7, 107),

(1008, 8, 108),

(1009, 9, 109),

(1010, 10, 110);

-- Query to generate customized student reports

SELECT

CONCAT(first\_name, ' ', last\_name) AS full\_name,

age,

email\_address,

CASE

WHEN enrolled\_flag = 1 THEN 'Enrolled'

ELSE 'Not Enrolled'

END AS enrollment\_status

FROM Students;

ALTER TABLE Students ADD COLUMN assessment\_score INT;

-- Query to convert email addresses to lowercase and classify students

SELECT

CONCAT(first\_name, ' ', last\_name) AS full\_name,

age,

LOWER(email\_address) AS lowercase\_email,

CASE

WHEN assessment\_score >= 90 THEN 'Advanced'

WHEN assessment\_score >= 70 THEN 'Intermediate'

ELSE 'Beginner'

END AS student\_category

FROM Students;

-- Note: You need to add the assessment\_score column to the Students table to run this query.

-- Query to calculate and display the average score for each course

SELECT

C.course\_name,

COUNT(E.student\_id) AS num\_students\_enrolled,

AVG(S.assessment\_score) AS average\_score

FROM Courses C

LEFT JOIN Enrollments E ON C.course\_id = E.course\_id

LEFT JOIN Students S ON E.student\_id = S.student\_id

GROUP BY C.course\_name;

-- Note: You need to add the assessment\_score column to the Students table to run this query.

-- Query to retrieve student information and their enrolled courses

SELECT

CONCAT(S.first\_name, ' ', S.last\_name) AS student\_name,

S.email\_address,

C.course\_name

FROM Students S

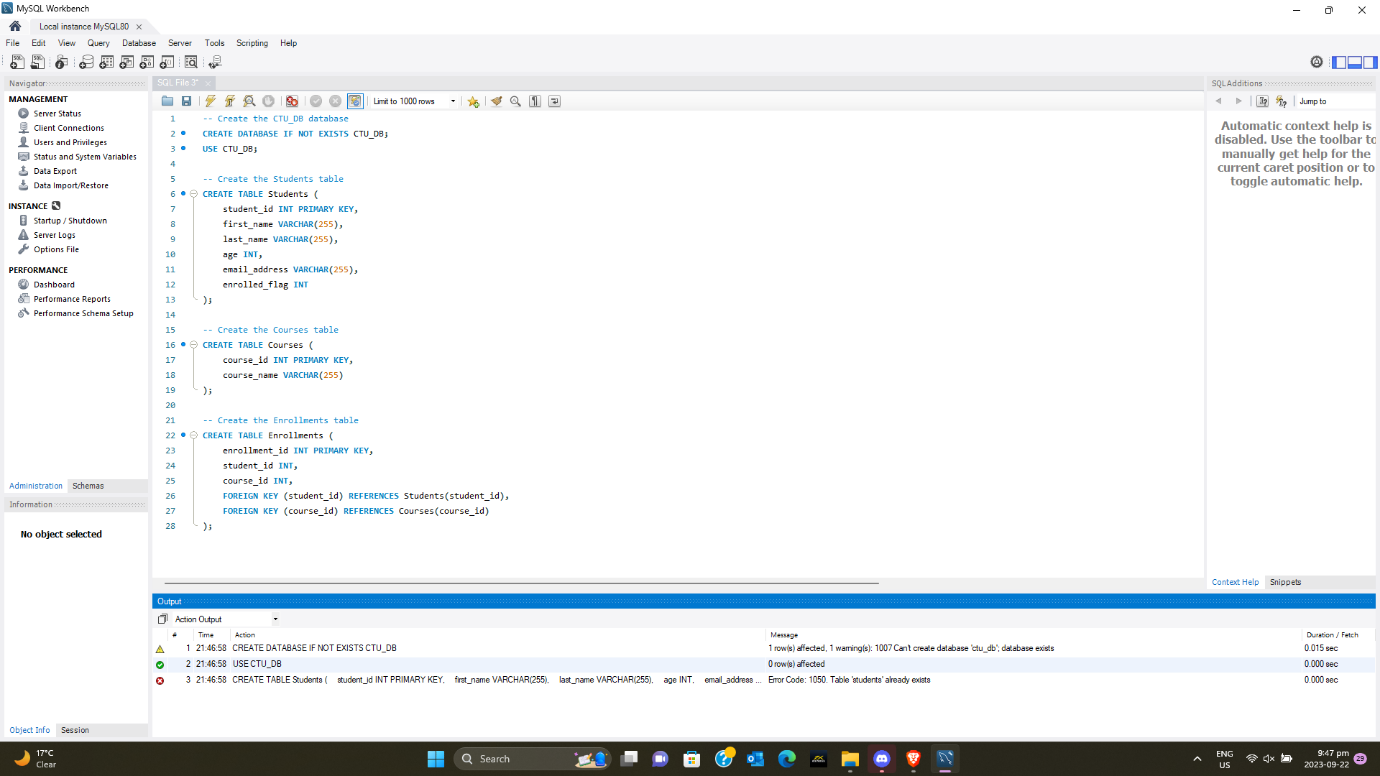
INNER JOIN Enrollments E ON S.student\_id = E.student\_id

INNER JOIN Courses C ON E.course\_id = C.course\_id

WHERE S.enrolled\_flag = 1

ORDER BY student\_name;

**Output:**

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

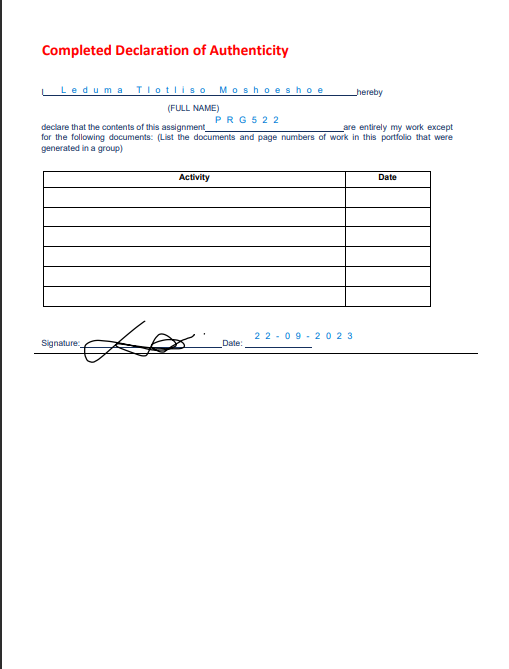
Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

****