CTU training solutions

0861 100 395 | www.ctutraining.co.za | enquiry@ctutraining.co.za

Gabriella Rakgotsoka

20232605

Table of Contents

No table of contents entries found.

Question 1

Problem Title: Data Transformation and Aggregation System for CTU Training Solutions

Problem Description:

Background:

CTU Training Solutions, a renowned educational institution, has approached you to design and develop a comprehensive Oracle database that addresses their specific needs for data manipulation, conversion, and reporting. The database will be used to manage student information, course details, and assessment scores.

Requirements:

Unit 5: Using Single-Row Functions to Customize Output: (20 marks)

CTU requires the ability to generate customized student reports. Each report should include the student's full name, age, email address, and a calculated field indicating their enrollment status (either 'Enrolled' or 'Not Enrolled' based on a flag in the database).

```
SELECT
   CONCAT(First_Name, ' ', Last_Name) AS Full_Name,
   Age,
   Email_Address,
   CASE
      WHEN Enrollment_Flag = 'Y' THEN 'Enrolled'
      ELSE 'Not Enrolled'
   END AS Enrollment_Status
FROM
   Students;
```

CONCAT(First_Name, ' ', Last_Name) AS Full_Name: Concatenates the First_Name and Last_Name columns to form the full name of the student.

Age: Retrieves the age of the student.

email_address: Retrieves the email address of the student.

CASE WHEN Enrollment_Flag = 'Y' THEN 'Enrolled' ELSE 'Not Enrolled' END AS Enrollment_Status: Uses a CASE statement to determine the enrollment status of

the student based on the value of the Enrollment_Flag column. If the flag is 'Y', it indicates the student is enrolled; otherwise, they are considered not enrolled.

Unit 6: Using Conversion Functions and Conditional Expressions: (25 marks)

CTU wants to ensure that the database stores student data consistently. Implement a mechanism to convert and store all email addresses in lowercase to prevent duplication.

To do this I could use a trigger to automatically convert and store email addresses in lowercase when inserting or updating student records. Such as in the following code:

```
CREATE OR REPLACE TRIGGER email_lowercase_trigger

BEFORE INSERT OR UPDATE ON Students

FOR EACH ROW

BEGIN

    :NEW.Email_Address := LOWER(:NEW.Email_Address);

END;
```

this trigger should fire before each insert or update operation on the students table and convert the email address to lowercase using the LOWER function.

Additionally, create a conditional expression to classify students into different categories (e.g., 'Beginner,' 'Intermediate,' 'Advanced') based on their assessment scores.

I can use a CASE statement to classify the students into different categories based on their assessment scores:

```
SELECT

    CONCAT(First_Name, ' ', Last_Name) AS Full_Name,

    Assessment_Score,

    CASE

        WHEN Assessment_Score >= 90 THEN 'Advanced'

        WHEN Assessment_Score >= 70 THEN 'Intermediate'

        ELSE 'Beginner'

    END AS category

FROM

    Students;
```

In the provided code, students with an assessment score of 90 or higher are classified as 'Advanced,' students with a score of 70 or higher but less than 90 are classified as 'Intermediate,' and all others are classified as 'Beginner.'

When these mechanisms are implemented in the Oracle database, it will likely ensure consistent storage of student data and provide a classification system for students based on their assessment scores. Data integrity will be maintained in this way and provide valuable insights into student performance.

Unit 7: Reporting Aggregated Data Using the Group Functions: (30 marks)

The institution needs the ability to generate reports on course performance. Develop a query that calculates and displays the average score for each course. The report should include the course name, the number of students enrolled in each course, and the average assessment score.

To generate a report on course performance, including the average score for each course along with the number of students enrolled in each course, you can use SQL aggregate functions along with JOINs to combine data from the Courses, Enrollments, and Students tables. Below is a query that would calculate and display the required information:

```
SELECT
    c.Course_Name,
    COUNT(e.Student_ID) AS Num_Students_Enrolled,
    AVG(s.Assessment_Score) AS Average_Assessment_Score
FROM
    Courses c
INNER JOIN
    Enrollments e ON c.Course_ID = e.Course_ID
INNER JOIN
    Students s ON e.Student_ID = s.Student_ID
GROUP BY
    c.Course_Name;
```

The Course_Name is selected from the Courses table to display the name of each course. Then we make use of the COUNT aggregate function along with the Student_ID from the Enrollments table to count the number of students enrolled in each course. Then, the AVG aggregate function would be used along with the Assessment_Score from the Students table to calculate the average assessment score for each course.

We then perform INNER JOINs between the Courses, Enrollments, and Students tables based on their corresponding keys (Course_ID and Student_ID) to retrieve the necessary data.

Lastly, we would group the results by Course_Name using the GROUP BY clause to aggregate the data by course.

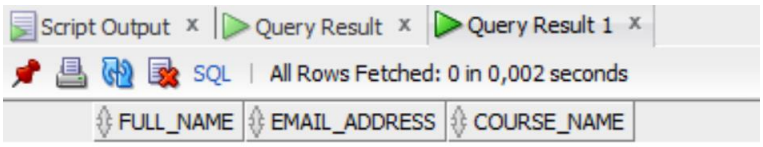
Executing this query would generate a report with three columns: Course_Name, Num_Students_Enrolled, and Average_Assessment_Score, manifesting insights into the performance from each course.

Unit 8: Displaying Data from Multiple Tables: (25 marks)

CTU requires a comprehensive report that combines student information and their enrolled courses. Create a query that retrieves each student's full name, email address, and the names of the courses they are enrolled in. The report should be sorted by student name and include only students who are currently enrolled.

To create a query that retrieves each student's full name, email address, and the names of the courses they are enrolled in, sorted by student name and including only currently enrolled students, JOINs between the Students, Enrollments, and Courses tables could be used. The following query should work:

```
SELECT
    CONCAT(s.First_Name, ' ', s.Last_Name) AS Full_Name,
    s.Email_Address,
    c.Course_Name
    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.Enrollment_Flag = 'Y'
ORDER BY
    Full_Name;
```



This will select the concatenated First_Name and Last_Name from the Students table to display the full name of each student.

The we select the Email_Address from the Students table to include the email address of each student. We would also select the Course_Name from the Courses table to display the names of the courses each student is enrolled in.

Afterwards we would perform INNER JOINs between the Students, Enrollments, and Courses tables based on their corresponding keys (Student_ID and Course_ID) to retrieve the necessary data.

A WHERE clause is added to filter out students who are not currently enrolled (Enrollment_Flag = 'Y'). Lastly, the ORDER BY clause would be used to sort the results by the full name of the students.

Completed Declaration of Authenticity

I Gabriella Rakgotsoka _ hereby
(FULL NAME)
declare that the contents of this assignment PRG522_FA2 is entirely my own work except for the following documents: (List the documents and page numbers of work in this portfolio that were generated in a group)

Activity	Date

Signature:  Date: 2024/04/29