|  |  |
| --- | --- |
|  | **Database Management Systems**  **BSCS-4**  **Department of Computer Science**  **Bahria University, Lahore Campus** |

**Assignment: [1]**

Date: Week 4, 15th October 2023

Name: Muhammad Hammad

Roll No: 03-134221-024

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation of CLO** | **Question Number** | **Marks** | **Obtained Marks** |
| **CLO: Queries to extract information from database.** |  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Total Marks** | | **25** |  |

**Instructions for the Case Study:**

**Perform the following SQL queries based on the given database schema. Suitable tuples have to be entered so that queries may be executed correctly.**

**Scenario:**

A university database contains information about students, courses, and registrations. The database schema consists of the following tables:

**1. Students:**

- student\_id (Primary Key)

- student\_name

- student\_age

- student\_major

**2. Courses:**

- course\_id (Primary Key)

- course\_name

- course\_department

- course\_credits

**3. Registrations:**

- registration\_id (Primary Key)

- student\_id (Foreign Key referencing Students)

- course\_id (Foreign Key referencing Courses)

- registration\_date

**Assume suitable data exists in these tables.**

**Task 1:**

1. Write SQL queries to create the database named "UniversityDB" and create the three tables (Students, Courses, Registrations) with the appropriate attributes.

**Solution:**

create database UniversityDB

create table Students

(

student\_id int primary key,

student\_name varchar(25),

student\_age int,

student\_major varchar(25)

);

insert into Students values (001,'Usman Ali',21,'Computer Science');

insert into Students values (021,'Haider Khan',20,'Computer Science');

insert into Students values (011,'Ali Ahmed',21,'Language');

insert into Students values (034,'Zubair Ali',19,'Language');

insert into Students values (014,'Osama Khalid', 22 , Null);

create table Courses

(

course\_id int Primary Key,

course\_name varchar(25),

course\_department varchar(25),

course\_credits varchar(25)

);

insert into Courses values (101,'Bio','Science', 3);

insert into Courses values (211,'DLD','Science', 3);

insert into Courses values (055,'Urdu','Language', 2);

insert into Courses values (0,Null,Null,Null);

create table Registration

(

registration\_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),

registration\_date int

);

insert into Registration values (001,1, 101, 2023);

insert into Registration values (002,21, 211, 2023);

insert into Registration values (003,11, 55, 2021);

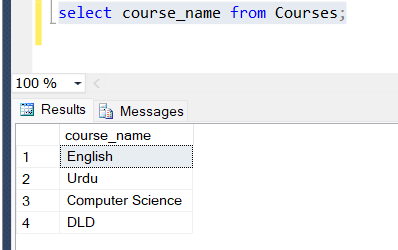
insert into Registration values (004,34, 55, 2023);

insert into Registration values (005,14, 0, 2022);

1. Display the names of all courses offered by the university.

**Solution:**

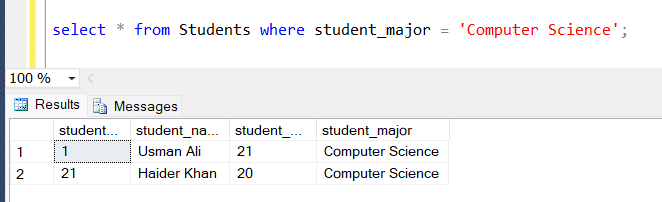
select course\_name from Courses;

****

1. List all students who are majoring in "Computer Science."

**Solution:**

select \* from Students where student\_major = 'Computer Science';



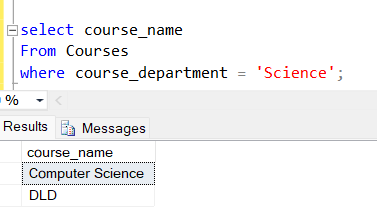
1. Show the course names that belong to the "Science" department.

**Solution:**

select course\_name

From Courses

where course\_department = 'Science'

****

1. Retrieve the names of students who have registered for the course with ID 101.

**Solution:**

**Method 1:**

SELECT s.student\_name

FROM Students as S,Registration as r

Where s.student\_id = r.student\_id and r.course\_id = 101;

**Method 2:**

SELECT student\_name

FROM Students

WHERE student\_id IN (

Select student\_id

FROM Registration

WHERE course\_id = 101

);

**A screenshot of a computer

Description automatically generated**

1. Display the total number of registrations in the database.

**Solution:**

select count(registration\_id) as Registered

From Registration

**A screenshot of a computer

Description automatically generated**

**Task 2:**

1. Write an SQL query to calculate the average age of all students.

**Solution:**

select avg(student\_age) as Avg\_Student\_Age

from Students;

**A screenshot of a computer

Description automatically generated**

1. List the course names that have more than 3 credits.

**Solution:**

select course\_name

From Courses

where course\_credits > 3;

**A screenshot of a computer message

Description automatically generated**

1. Show the names of students who are majoring in "Mathematics" and are older than 20 years.

**Solution:**

select student\_name from Students where student\_major = 'Mathematics' and student\_age>20;

A white rectangular object with black text

Description automatically generated

1. Display the student names who have not registered for any course.

**Solution:**

**Method 1:**

SELECT s.student\_name

FROM Students as S,Registration as r

Where s.student\_id = r.student\_id and r.course\_id = 0;

**Method 2:**

SELECT student\_name

FROM Students

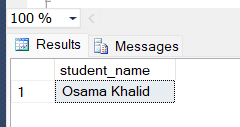
WHERE student\_id IN (

Select student\_id

FROM Registration

WHERE course\_id = 0

);

****

1. List the course names along with the count of students registered for each course.

**Solution:**

SELECT c.course\_name,

(SELECT COUNT(\*) FROM Registration r WHERE r.course\_id = c.course\_id) AS student\_count

FROM Courses c;

A screenshot of a computer

Description automatically generated

1. Retrieve the student names who registered for courses in the year 2022.

**Solution:**

**Method 1:**

SELECT S.student\_name

FROM Students AS S, Registration AS R

WHERE S.student\_id = R.student\_id AND R.registration\_date = 2022

**Method 2:**

SELECT student\_name

FROM Students

WHERE student\_id IN (

SELECT student\_id

FROM Registration

WHERE registration\_date = 2022

);

**A screenshot of a computer

Description automatically generated**

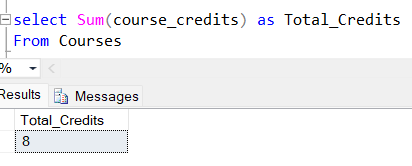
**Task 3:**

1. Calculate the sum of credits for all courses.

**Solution:**

select Sum(course\_credits) as Total\_Credits

From Courses



1. Show the student names along with their major and the department of the courses they have registered for.

**Solution:**

SELECT s.student\_name,s.student\_major,c.course\_department

FROM Students as s,Registration as r ,Courses as c

Where s.student\_id = r.student\_id and r.course\_id = c.course\_id;

**A screenshot of a computer

Description automatically generated**

1. Display the course names with the earliest registration date.

**Solution:**

SELECT c.course\_name, min(r.registration\_date) AS Earliest

FROM Courses as c,Registration as r

Where c.course\_id = r.course\_id and c.course\_name is not Null

GROUP BY c.course\_name order by Earliest;

**A screenshot of a computer

Description automatically generated**

1. List the student names who registered for the same course multiple times.

**Solution:**

SELECT s.student\_name,r.course\_id,COUNT(\*) AS registration\_count

FROM Students as s,Registration as r

Where s.student\_id = r.student\_id

GROUP BY s.student\_name, r.course\_id

HAVING COUNT(\*) > 1;

**A screenshot of a computer

Description automatically generated**

1. Retrieve the course names that have the highest number of registrations.

**Solution:**

With student\_count as

(SELECT c.course\_name, count(r.student\_id) as total\_count

FROM Courses as c, Registration as r

Where c.course\_id = r.course\_id

GROUP BY c.course\_name

)

select course\_name

from student\_count

WHERE total\_count = (SELECT MAX(total\_count) FROM student\_count);

**A screenshot of a computer

Description automatically generated**

1. Eliminate the "Registrations" table from the database.

**Solution:**

Drop table Registration;

A screenshot of a computer

Description automatically generated

**Please provide SQL queries with results for each task**