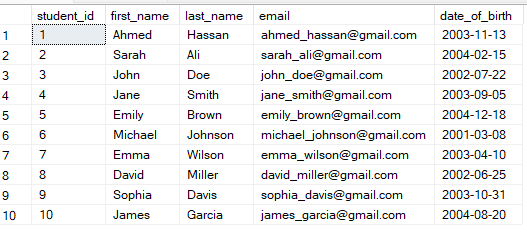
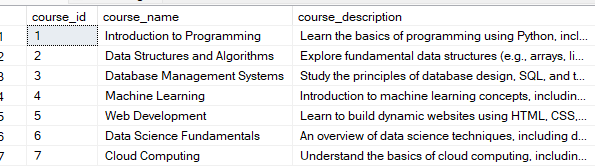
SQL Task

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Basic queries:

Select all students => (SELECT \* FROM Students ;)



Select all courses=> (SELECT \* FROM Courses ;)

Select all enrollments with student names and course names => (

SELECT e.\* , (s.first\_name+' '+s.last\_name)as full\_name ,c.course\_name from Enrollment as e

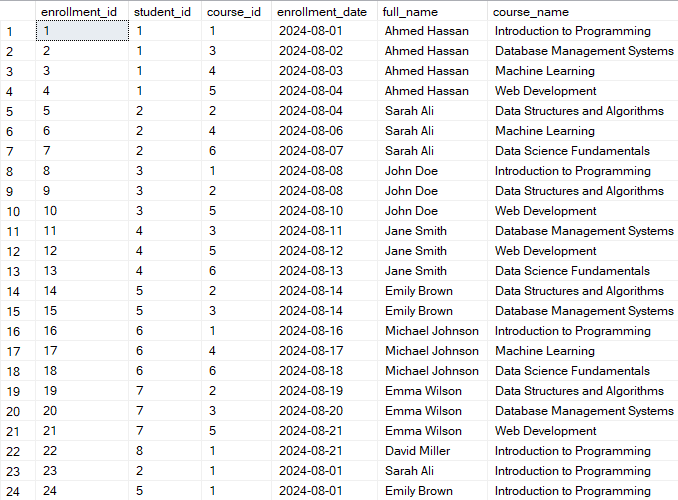
join Students as s

on e.student\_id = s.student\_id

join Courses as c

on e.course\_id = c.course\_id

)



Advanced queries:

Select students who enrolled in a specific course => (SELECT c.course\_name , (s.first\_name+' '+s.last\_name)as full\_name from Enrollment as e

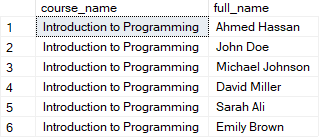
join Students as s

on e.student\_id = s.student\_id

join Courses as c

on e.course\_id = c.course\_id

where c.course\_id = 1;)



Select courses with more than 5 students => (SELECT c.course\_name , count(e.course\_id) as number\_of\_students from Enrollment as e

join Courses as c

on c.course\_id = e.course\_id

group by course\_name

having count(e.course\_id) > 5;)



Update a student's email = > (update Students

set email = 'ahmedhassanah027724@gmail.com'

where student\_id = 1;)

Delete a course that no students are enrolled in => (delete from Courses

where course\_id not in (

select distinct course\_id

from Enrollment

);)

Calculate the average age of students => ( SELECT AVG(DATEDIFF(year , date\_of\_birth , GETDATE())) as average\_age from Students ;)



Find the course with the maximum enrollments => (SELECT top 1 c.course\_name ,count(e.course\_id) as number\_of\_students from Enrollment as e

join Courses as c

on e.course\_id = c.course\_id

group by course\_name

order by count(e.course\_id) desc

)

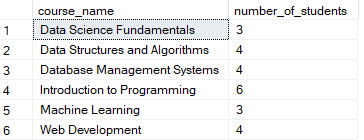


List courses along with the number of students enrolled (use GROUP BY) =>(SELECT c.course\_name ,count(e.course\_id) as number\_of\_students from Enrollment as e

join Courses as c

on e.course\_id = c.course\_id

group by course\_name)



Join queries :

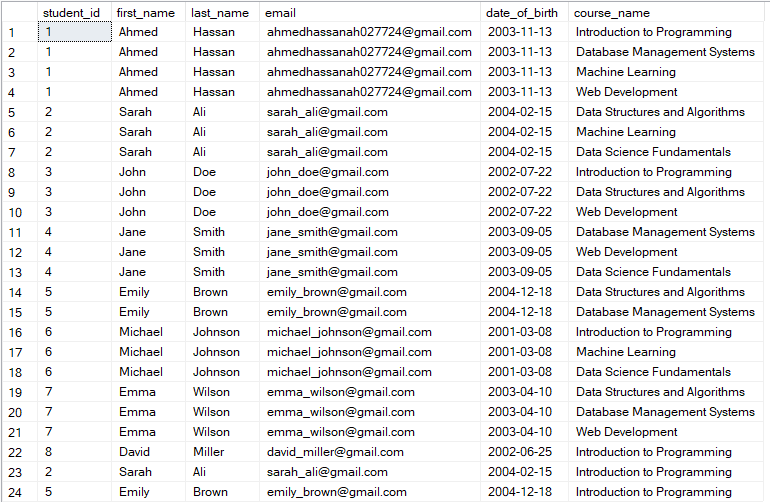
Select all students with their enrolled courses (use JOIN) => (SELECT s.\* , c.course\_name from Enrollment AS e

join Students AS s

ON e.student\_id = s.student\_id

join Courses AS c

ON e.course\_id = c.course\_id)



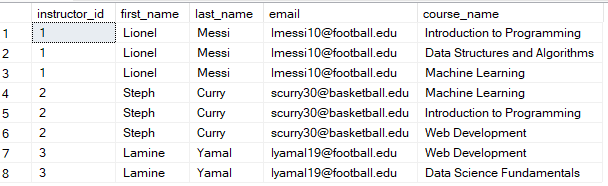
List all instructors and their courses => (SELECT i.\* , c.course\_name FROM Ins\_to\_course AS itc

join Instructors as i

on itc.instructor\_id = i.instructor\_id

join Courses as c

on itc.course\_id = c.course\_id)



Find students who are not enrolled in any course => (

SELECT (first\_name+' '+last\_name) as 'full name' FROM Students

WHERE student\_id NOT IN (SELECT DISTINCT student\_id FROM Enrollment))



Supqueries and Set Operations

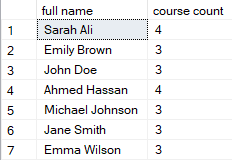
Select students enrolled in more than one course => (SELECT (s.first\_name+' '+s.last\_name) as 'full name' , count(e.student\_id) AS 'course count' FROM Enrollment AS e

join Students as s

on e.student\_id = s.student\_id

group by s.first\_name , s.last\_name

having COUNT(e.student\_id) >1)



Find courses taught by a specific instructor => (select c.course\_name , i.first\_name + ' ' + i.last\_name as 'instructor name' from Ins\_to\_course as itc

join Instructors as i

on itc.instructor\_id = i.instructor\_id

join Courses as c

on itc.course\_id = c.course\_id

where itc.course\_id not in (select course\_id from Ins\_to\_course

group by course\_id

having count(course\_id)>1))



Select the top 3 students with the most enrollments =>(select top 3 (s.first\_name +' '+s.last\_name) as 'student name' , count(e.student\_id) as 'number of courses' from Enrollment as e

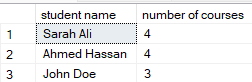
join Students as s

on e.student\_id = s.student\_id

group by s.first\_name , s.last\_name

order by count(e.student\_id) desc

)



Use UNION to combine results of two different SELECT queries =>(

SELECT

(first\_name +' '+last\_name) as 'full name'

FROM

Students

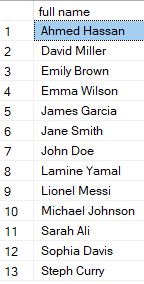
UNION

SELECT

(first\_name +' '+last\_name) as 'full name'

FROM

Instructors;)



Functions and Stored Procedures:

Create a stored procedure to add a new student =>(CREATE PROCEDURE AddNewStudent

@FirstName NVARCHAR(100),

@LastName NVARCHAR(100),

@Email NVARCHAR(100),

@BirthDate DATE

AS

BEGIN

INSERT INTO Students (first\_name, last\_name, email, date\_of\_birth)

VALUES (@FirstName, @LastName, @Email, @BirthDate);

END;

EXEC AddNewStudent 'omar' ,'nouh' , 'omar\_nouh@gmail.com' , '2004-3-1')

Create a function to calculate the age of a student based on their date of birth => (CREATE FUNCTION CalculateAge (@BirthDate DATE)

RETURNS INT

AS

BEGIN

RETURN DATEDIFF(YEAR, @BirthDate, GETDATE())

END;

SELECT

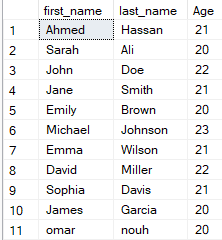
first\_name,

last\_name,

dbo.CalculateAge(date\_of\_birth) AS Age

FROM

Students;)



Aggregate Function and Grouping:

Calculate the total number of students =>(SELECT

COUNT(\*) AS TotalNumberOfStudents

FROM

Students;)



Calculate the average, minimum, and maximum number of enrollments per course => (SELECT

AVG(enrollment\_count) AS AverageEnrollments,

MIN(enrollment\_count) AS MinimumEnrollments,

MAX(enrollment\_count) AS MaximumEnrollments

FROM (

SELECT

course\_id,

COUNT(student\_id) AS enrollment\_count

FROM

Enrollment

GROUP BY

course\_id

) AS EnrollmentStats;)

