**Manav Rachna International Institute of Research and Studies**

**ORACLE LAB**

**Aryan paou 22/FCA/BCA(CS)/007**

# 1. Step 1: Defining tables Students Table

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Age INT,

Major VARCHAR(50)

**Courses table**

);

CREATE TABLE Courses (

CourseID INT PRIMARY KEY,

CourseName VARCHAR(100),

Credits INT

);

**Enrollments Table (for many-to-many relationship)**

CREATE TABLE Enrollments (

EnrollmentID INT PRIMARY KEY,

StudentID INT,

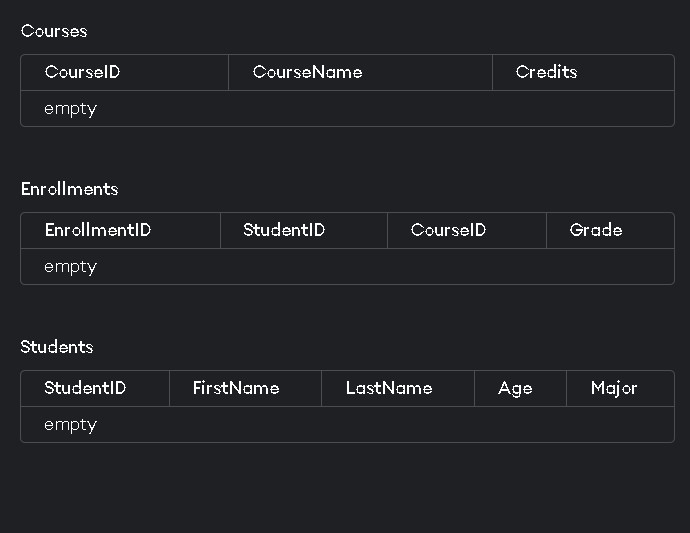
CourseID INT,

Grade CHAR(1),

FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)

);



# 2. Step 2: Inserting Sample Data

**Insert Data into Students Table**

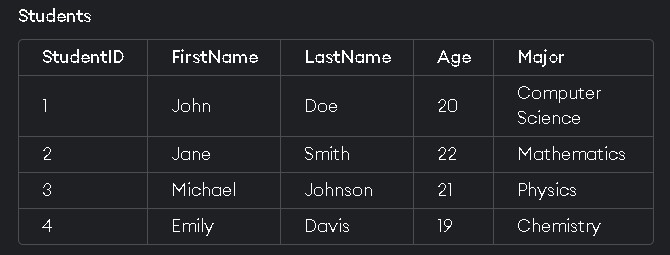
**INSERT INTO Students (StudentID, FirstName, LastName, Age, Major) VALUES**

**(1, 'John', 'Doe', 20, 'Computer Science'),**

**(2, 'Jane', 'Smith', 22, 'Mathematics'),**

**(3, 'Michael', 'Johnson', 21, 'Physics'),**

**(4, 'Emily', 'Davis', 19, 'Chemistry');**



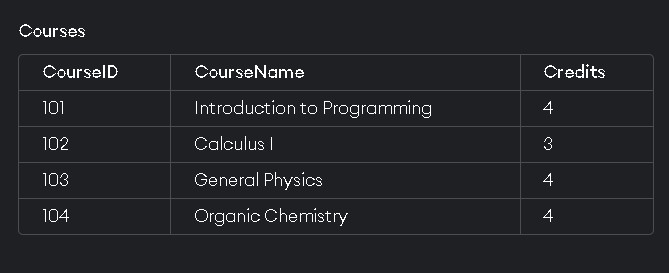
**INSERT INTO Courses (CourseID, CourseName, Credits) VALUES**

**(101, 'Introduction to Programming', 4),**

**(102, 'Calculus I', 3),**

**(103, 'General Physics', 4),**

**(104, 'Organic Chemistry', 4);**



**INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES**

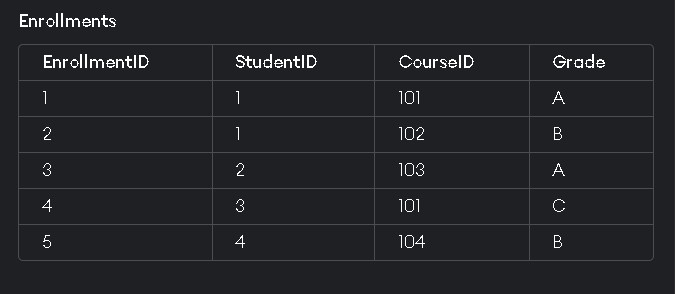
**(1, 1, 101, 'A'),**

**(2, 1, 102, 'B'),**

**(3, 2, 103, 'A'),**

**(4, 3, 101, 'C'),**

**(5, 4, 104, 'B');**



# 3. Step 3: Queries

1. **DDL (Data Definition Language) Queries 1. Create Table:**

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

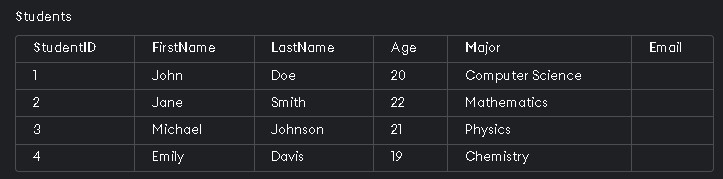
DepartmentName VARCHAR(100)

);



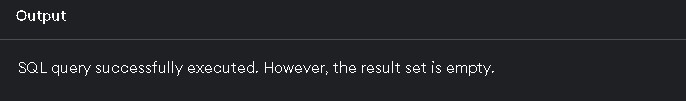
1. **Alter Table:**

ALTER TABLE Students ADD Email VARCHAR(100);



**3.Drop Table**

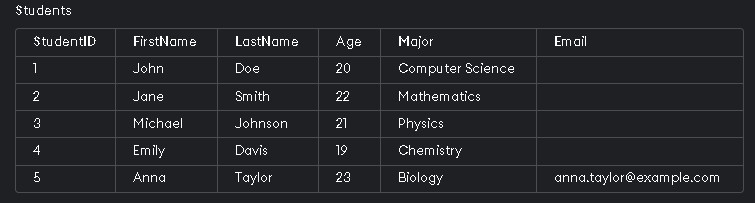
DROP TABLE Departments;



## 2. DML (Data Manipulation Language) Queries 1. Insert Data

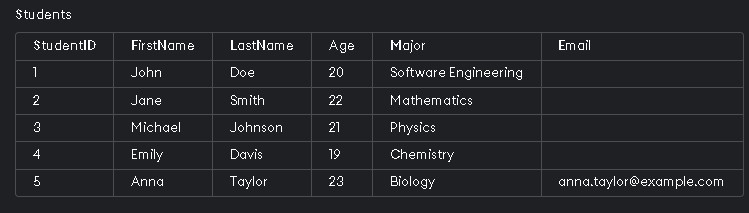
INSERT INTO Students (StudentID, FirstName, LastName, Age, Major, Email) VALUES (5,

'Anna', 'Taylor', 23, 'Biology', 'anna.taylor@example.com');



## 2. Update Data

UPDATE Students SET Major = 'Software Engineering' WHERE StudentID = 1;



## 3.Delete Data

Delete FROM Students WHERE StudentID = 4;

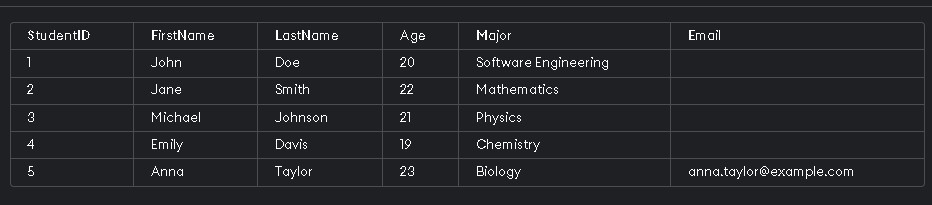
-- Deletion anomaly –

Output: Deletes Emily's record from the Students table.

## 3.DQL (Data Query Language) Queries

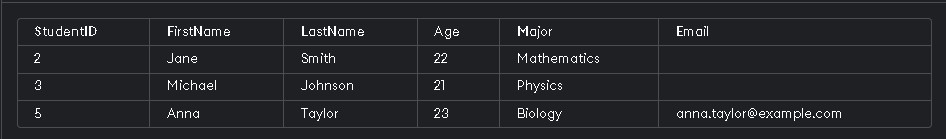
1. Select Data

SELECT \* FROM Students;



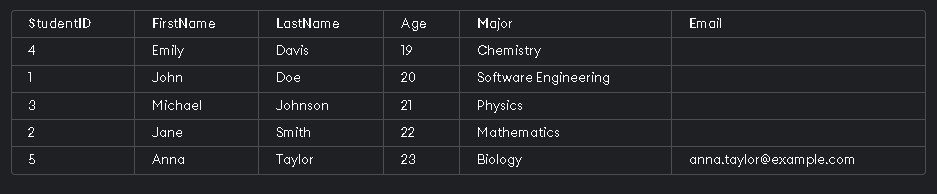
1. Where Clause

SELECT \* FROM Students WHERE Age > 20;



1. Order By Clause

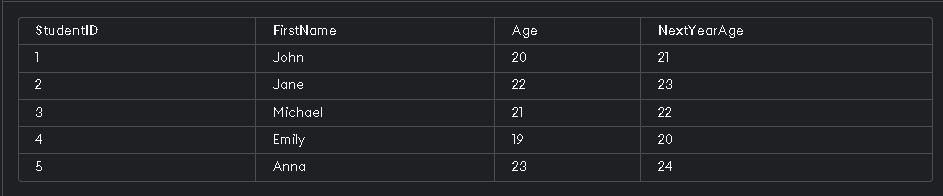
SELECT \* FROM Students ORDER BY LastName;



**4.Arithmetic Operations**

1. **Select with Arithmetic Operations**

SELECT StudentID, FirstName, Age, Age + 1 AS NextYearAge FROM Students;



**5.Primary and Foreign Key Relationships**

**1. Create Table with Foreign Key** CREATE TABLE Advisors (

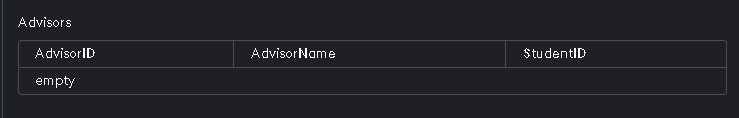
AdvisorID INT PRIMARY KEY,

AdvisorName VARCHAR(100),

StudentID INT,

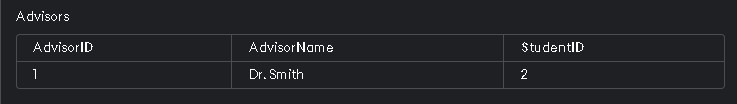
FOREIGN KEY (StudentID) REFERENCES Students(StudentID)

);



1. **Insert Data into Table with Foreign Key**

INSERT INTO Advisors (AdvisorID, AdvisorName, StudentID) VALUES (1, 'Dr. Smith', 2);



**6. Join Operations**

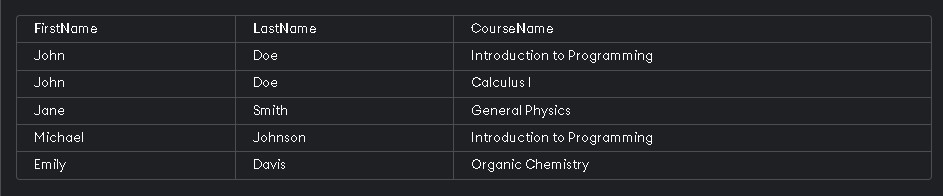
1. **Inner Join**

SELECT Students.FirstName, Students.LastName, Courses.CourseName

FROM Students

JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

JOIN Courses ON Enrollments.CourseID = Courses.CourseID;

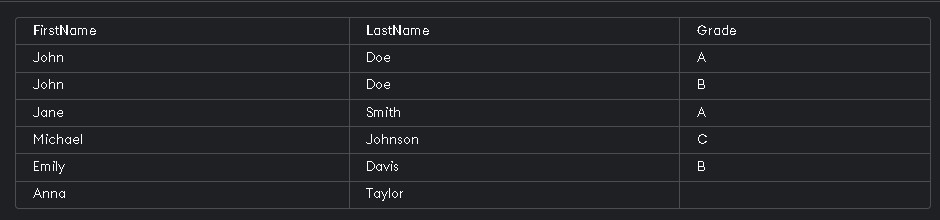


1. **Left Join**

SELECT Students.FirstName, Students.LastName, Enrollments.Grade

FROM Students

LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;



1. **Right Join**

SELECT Students.FirstName, Students.LastName, Enrollments.Grade

FROM Students

RIGHT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;

***Output: Displays all enrollments and the respective student details, if available.***

1. **Step 4: All Queries**

### 1. Create Students Table

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

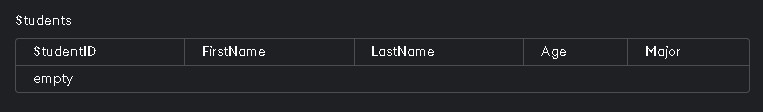
FirstName VARCHAR(50),

LastName VARCHAR(50),

Age INT,

Major VARCHAR(50)

);



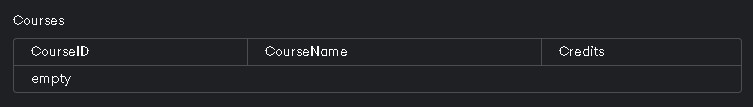
1. **Create Courses Table** CREATE TABLE Courses (

CourseID INT PRIMARY KEY,

CourseName VARCHAR(100),

Credits INT

);



1. **Create Enrollments Table**

CREATE TABLE Enrollments (

EnrollmentID INT PRIMARY KEY,

StudentID INT,

CourseID INT,

Grade CHAR(1),

FOREIGN KEY (StudentID) REFERENCES Students(StudentID),

FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)

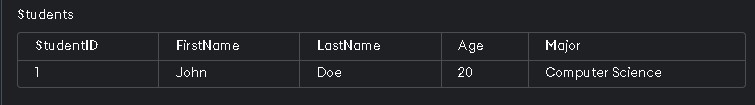
);



1. **Insert Data into Students Table**

INSERT INTO Students (StudentID, FirstName, LastName, Age, Major) VALUES (1, 'John', 'Doe', 20,

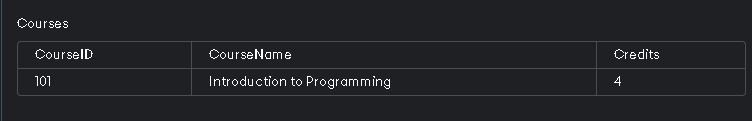
'Computer Science');



1. Insert Data into Courses Table

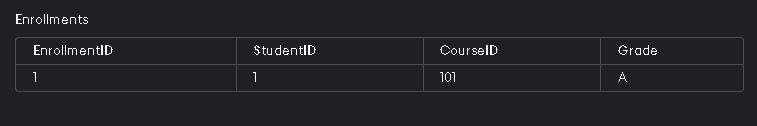
INSERT INTO Courses (CourseID, CourseName, Credits) VALUES (101, 'Introduction to

Programming', 4);



1. **Insert Data into Enrollments Table**

INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade) VALUES (1, 1, 101, 'A');



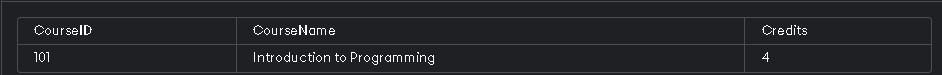
1. **Select All Students**

SELECT \* FROM Students;



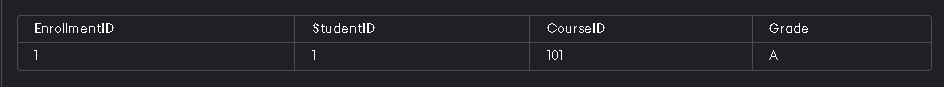
1. **Select all Courses**

SELECT \* FROM Courses;



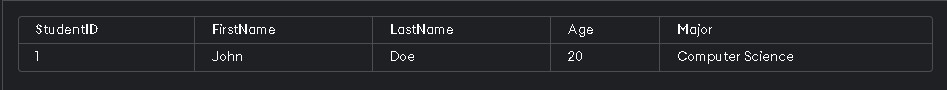
1. **Select All Enrollments**

SELECT \* FROM Enrollments;



1. **Select Students Older than 19**

SELECT \* FROM Students WHERE Age > 19;



1. **Update Student Major**

UPDATE Students SET Major = 'Software Engineering' WHERE StudentID = 1;



1. Delete a Student Record

DELETE FROM Students WHERE StudentID = 4; *Output: Emily Davis' record deleted.*

1. Create Departments Table

CREATE TABLE Departments ( DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);



1. Add Column to Students Table

ALTER TABLE Students ADD Email VARCHAR(100);



1. Drop Departments Table

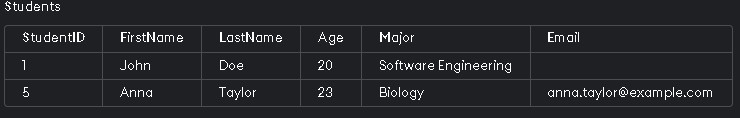
DROP TABLE Departments;

*Output: Departments table dropped.*

1. Insert New Student with Email

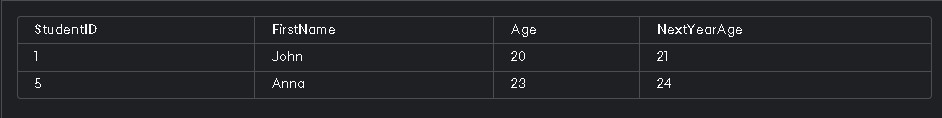
INSERT INTO Students (StudentID, FirstName, LastName, Age, Major, Email) VALUES (5, 'Anna',

'Taylor', 23, 'Biology', 'anna.taylor@example.com');



1. **Select Students with Age Arithmetic Operation**

SELECT StudentID, FirstName, Age, Age + 1 AS NextYearAge FROM Students;



### 18. Create Advisors Table with Foreign Key

CREATE TABLE Advisors (

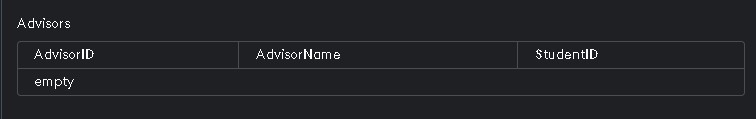
AdvisorID INT PRIMARY KEY,

AdvisorName VARCHAR(100),

StudentID INT,

FOREIGN KEY (StudentID) REFERENCES Students(StudentID)

);



19. **Insert Data into Advisors Table**

INSERT INTO Advisors (AdvisorID, AdvisorName, StudentID) VALUES (1, 'Dr. Smith', 2);

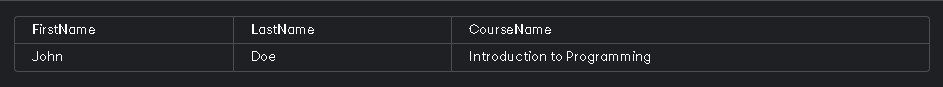
#### 20. Inner Join Students and Enrollments

SELECT Students.FirstName, Students.LastName, Courses.CourseName

FROM Students

JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

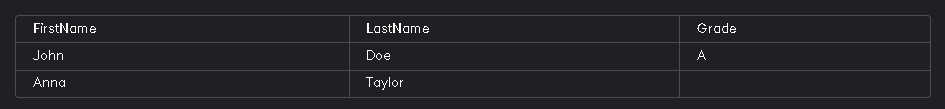
JOIN Courses ON Enrollments.CourseID = Courses.CourseID;



1. **Left Join Students and Enrollments**

SELECT Students.FirstName, Students.LastName, Enrollments.Grade FROM Students

LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;



1. **Right Join Students and Enrollments**

SELECT Students.FirstName, Students.LastName, Enrollments.Grade

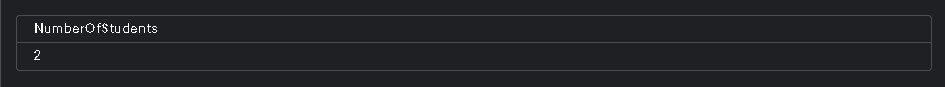
FROM Students

RIGHT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;

*Output: Displays all enrollments and the respective student details, if available.*

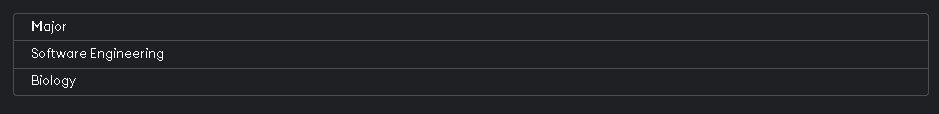
1. Count Number of Students

SELECT COUNT(\*) AS NumberOfStudents FROM Students;



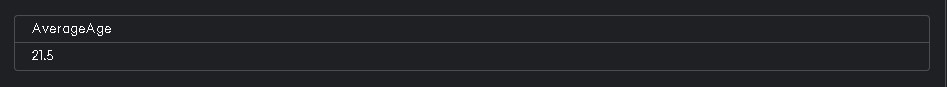
1. Select Distinct Majors

SELECT AVG(Age) AS AverageAge FROM Students;



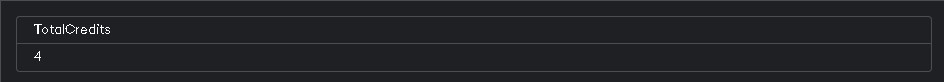
1. **Select Average Age of Students**

SELECT AVG(Age) AS AverageAge FROM Students;



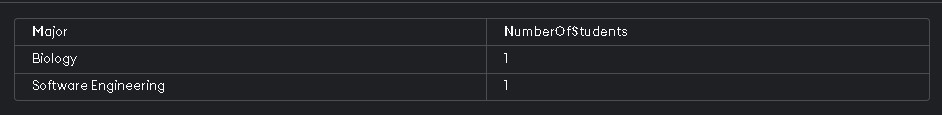
1. **Select Sum of Credits**

SELECT SUM(Credits) AS TotalCredits FROM Courses;



1. **Select Students Grouped by Major**

SELECT Major, COUNT(\*) AS NumberOfStudents FROM Students GROUP BY Major;



1. **Select Students with a Specific Major**

SELECT \* FROM Students WHERE Major = ‘Biology’;



1. **Select Students with Age Between 20 and 22**

SELECT \* FROM Students WHERE Age BETWEEN 20 AND 22;



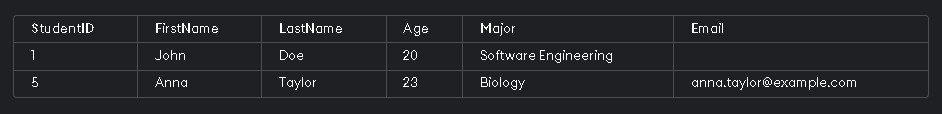
1. **Select Students with Names Starting with 'J'**

SELECT \* FROM Students WHERE FirstName LIKE 'J%';



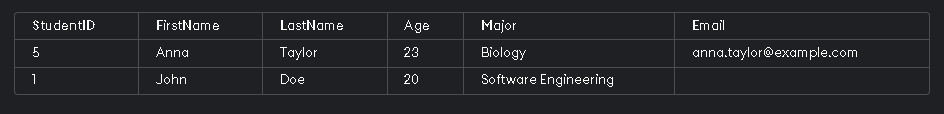
1. **Select Students in Ascending Order of Age**

SELECT \* FROM Students ORDER BY Age ASC;



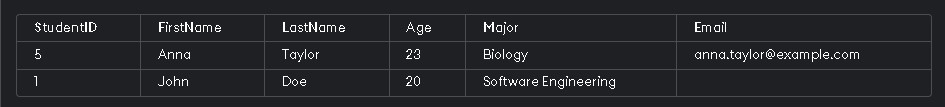
1. **Select Students in Descending Order of Last Name**

SELECT \* FROM Students ORDER BY LastName DESC;



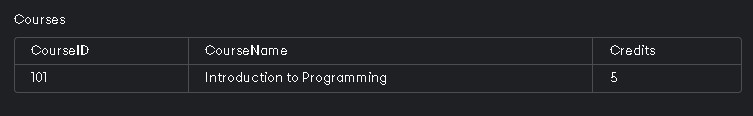
1. **Select Top 3 Oldest Students**

SELECT \* FROM Students ORDER BY Age DESC LIMIT 3;



1. **Update Course Credits**

UPDATE Courses SET Credits = 5 WHERE CourseID = 101;



1. **Delete a Course Record**

DELETE FROM Courses WHERE CourseID = 104;

*Output: Deletes course with ID 104.*

1. **Select Students Enrolled in a Specific Course** SELECT Students.FirstName, Students.LastName

FROM Students

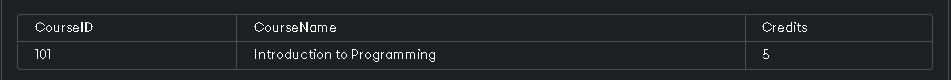
JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

WHERE Enrollments.CourseID = 101;



1. **Select Courses with More Than 3 Credits**

SELECT \* FROM Courses WHERE Credits > 3;



1. **Select Students with Null Email**

SELECT \* FROM Students WHERE Email IS NULL;



1. **Select Students with Non-Null Email**

SELECT \* FROM Students WHERE Email IS NOT NULL;



1. **Select Students Who Have Taken Multiple Courses**

SELECT Students.FirstName, Students.LastName

FROM Students

JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

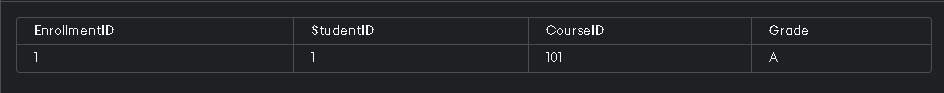
GROUP BY Students.StudentID, Students.FirstName, Students.LastName

HAVING COUNT(Enrollments.CourseID) > 1;



1. **Select Enrollments with Grades A or B**

SELECT \* FROM Enrollments WHERE Grade IN ('A', 'B');



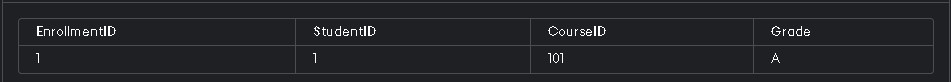
1. **Select Students with Age Not Between 18 and 22**

SELECT \* FROM Students WHERE Age NOT BETWEEN 18 AND 22;



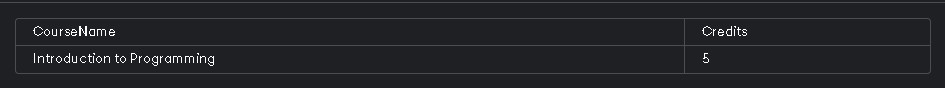
1. **Select Enrollments in Ascending Order of Grade**

SELECT \* FROM Enrollments ORDER BY Grade ASC;



1. **Select Course Names and Credits**

SELECT CourseName, Credits FROM Courses;



1. **Select Students Enrolled in Specific Course with Grade A**

SELECT Students.FirstName, Students.LastName

FROM Students

JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

WHERE Enrollments.CourseID = 101 AND Enrollments.Grade = 'A';



1. **Select Students Grouped by Major and Age** SELECT Major, Age, COUNT(\*) AS NumberOfStudents

FROM Students

GROUP BY Major, Age;

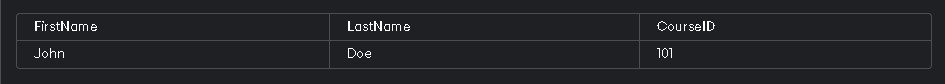


1. **Select Students and Their Enrollments**

SELECT Students.FirstName, Students.LastName, Enrollments.CourseID

FROM Students

JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;



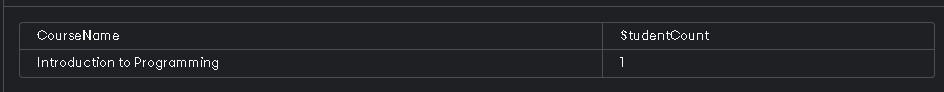
1. **Select Course Names with Student Count**

SELECT Courses.CourseName, COUNT(Enrollments.StudentID) AS StudentCount

FROM Courses

JOIN Enrollments ON Courses.CourseID = Enrollments.CourseID

GROUP BY Courses.CourseName;



1. **Select Advisors and Their Students**

SELECT Advisors.AdvisorName, Students.FirstName, Students.LastName

FROM Advisors

JOIN Students ON Advisors.StudentID = Students.StudentID;



1. **Select Students Who Haven't Taken Any Courses**

SELECT Students.FirstName, Students.LastName

FROM Students

LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID

WHERE Enrollments.StudentID IS NULL;

