SQL Journey with Full Examples

This extended guide covers **all SQL topics** from the provided table, using the **Student-Teacher-Course story** as context. Each concept comes with an example query.

Data Types

SQL Data Types

- INT: `StudentID INT`
- VARCHAR: `FirstName VARCHAR(50)`
- DATE: `BirthDate DATE`
- FLOAT: `GPA FLOAT`
- BOOLEAN: `IsActive BIT`

Database Management

SQL Database

A database is a container for tables.

```sql
CREATE DATABASE SchoolDB;
USE SchoolDB;
```

SQL Create DB

```sql
CREATE DATABASE UniversityDB;
```

SQL Drop DB

```sql
DROP DATABASE UniversityDB;
```

SQL Backup DB

```sql
BACKUP DATABASE SchoolDB
TO DISK = 'C:\backups\SchoolDB.bak';
```

Table Operations

SQL Create Table

```
CREATE TABLE Students (
  StudentID INT IDENTITY(1,1) PRIMARY KEY,
  FirstName VARCHAR(50) NOT NULL,
  LastName VARCHAR(50) NOT NULL,
  Age INT,
  Department VARCHAR(50)
);
```

SQL Drop Table

```
DROP TABLE Students;
```

SQL Alter Table

```
ALTER TABLE Students ADD Email VARCHAR(100);
ALTER TABLE Students DROP COLUMN Email;
```

Constraints

SQL Constraints

Constraints maintain data integrity.

SQL Not Null

```
CREATE TABLE Teachers (
  TeacherID INT NOT NULL,
  Name VARCHAR(50) NOT NULL
);
```

SQL Unique

```
ALTER TABLE Students ADD CONSTRAINT UC_Email UNIQUE (Email);
```

SQL Primary Key

```
CREATE TABLE Courses (
  CourseID INT PRIMARY KEY,
  CourseName VARCHAR(100)
);
```

SQL Foreign Key

```
CREATE TABLE Enrollments (
  EnrollmentID INT PRIMARY KEY,
  StudentID INT,
  CourseID INT,
  FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
  FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);
```

SQL Check

```
ALTER TABLE Students ADD CONSTRAINT CHK_Age CHECK (Age >= 16);
```

SQL Default

```
ALTER TABLE Students ADD CONSTRAINT DF_Department DEFAULT 'General' FOR
Department;
```

SQL Index

```
CREATE INDEX idx_lastname ON Students(LastName);
```

SQL Auto Increment

```
CREATE TABLE Students (
  StudentID INT IDENTITY(1,1) PRIMARY KEY,
  Name VARCHAR(50)
);
```

Data Handling

SQL Insert Into

```
INSERT INTO Students (FirstName, LastName, Age, Department)
VALUES ('Alice', 'Brown', 20, 'Computer Science');
```

SQL Select

```sql
SELECT * FROM Students;
```

SQL Select Distinct

```sql
SELECT DISTINCT Department FROM Students;
```

SQL Where

```sql
SELECT * FROM Students WHERE Age > 20;
```

SQL Order By

```sql
SELECT * FROM Students ORDER BY LastName ASC;
```

SQL And, Or, Not

```sql
SELECT * FROM Students WHERE Age > 18 AND Department = 'Math';
SELECT * FROM Students WHERE Age < 20 OR Department = 'History';
SELECT * FROM Students WHERE NOT Department = 'Physics';
```

SQL Null Values

```sql
SELECT * FROM Students WHERE Email IS NULL;
```

SQL Update

```sql
UPDATE Students SET Age = 21 WHERE FirstName = 'Alice';
```

SQL Delete

```sql
DELETE FROM Students WHERE LastName = 'Smith';
```

SQL Select Top

```sql
SELECT TOP 3 * FROM Students;
```

Aggregate Functions

SQL Min and Max

```
SELECT MIN(Age) AS Youngest, MAX(Age) AS Oldest FROM Students;
```

SQL Count

```
SELECT COUNT(*) FROM Students;
```

SQL Sum

```
SELECT SUM(Credits) FROM Courses;
```

SQL Avg

```
SELECT AVG(Age) FROM Students;
```

Pattern Matching

SQL Like

```
SELECT * FROM Students WHERE FirstName LIKE 'A%';
```

SQL Wildcards

```
SELECT * FROM Students WHERE LastName LIKE '_ohn';
```

Filtering

SQL In

```
SELECT * FROM Students WHERE Department IN ('Math', 'Physics');
```

SQL Between

```
SELECT * FROM Students WHERE Age BETWEEN 18 AND 22;
```

Aliases

SQL Aliases

```
SELECT FirstName AS Name, Department AS Dept FROM Students;
```

Joins

SQL Inner Join

```
SELECT s.FirstName, c.CourseName
FROM Students s
INNER JOIN Enrollments e ON s.StudentID = e.StudentID
INNER JOIN Courses c ON e.CourseID = c.CourseID;
```

SQL Left Join

```
SELECT s.FirstName, c.CourseName
FROM Students s
LEFT JOIN Enrollments e ON s.StudentID = e.StudentID
LEFT JOIN Courses c ON e.CourseID = c.CourseID;
```

SQL Right Join

```
SELECT t.FirstName, c.CourseName
FROM Courses c
RIGHT JOIN Enrollments e ON c.CourseID = e.CourseID
RIGHT JOIN Students t ON e.StudentID = t.StudentID;
```

SQL Full Join

```
SELECT s.FirstName, c.CourseName
FROM Students s
FULL JOIN Enrollments e ON s.StudentID = e.StudentID
FULL JOIN Courses c ON e.CourseID = c.CourseID;
```

SQL Self Join

```
SELECT A.FirstName AS Student1, B.FirstName AS Student2, A.Department
FROM Students A, Students B
WHERE A.Department = B.Department AND A.StudentID <> B.StudentID;
```

Set Operations

SQL Union

```
SELECT FirstName FROM Students
UNION
SELECT FirstName FROM Teachers;
```

SQL Union All

```
SELECT FirstName FROM Students
UNION ALL
SELECT FirstName FROM Teachers;
```

Grouping

SQL Group By

```
SELECT Department, COUNT(*) FROM Students GROUP BY Department;
```

SQL Having

```
SELECT Department, COUNT(*)
FROM Students
GROUP BY Department
HAVING COUNT(*) > 2;
```

Subqueries

SQL Exists

```
SELECT * FROM Students s
WHERE EXISTS (SELECT * FROM Enrollments e WHERE e.StudentID = s.StudentID);
```

SQL Any, All

```
SELECT * FROM Students
WHERE Age > ANY (SELECT Age FROM Students WHERE Department = 'Math');
```

```sql
SELECT * FROM Students
WHERE Age > ALL (SELECT Age FROM Students WHERE Department = 'Physics');
```

Table Creation from Queries

SQL Select Into

```sql
SELECT * INTO NewStudents FROM Students;
```

SQL Insert Into Select

```sql
INSERT INTO Alumni (FirstName, LastName)
SELECT FirstName, LastName FROM Students WHERE Department = 'Physics';
```

Logic

SQL Case

```sql
SELECT FirstName,
  CASE
    WHEN Age < 20 THEN 'Teen'
    WHEN Age BETWEEN 20 AND 22 THEN 'Young Adult'
    ELSE 'Adult'
  END AS AgeGroup
FROM Students;
```

Handling Nulls

SQL Null Functions

```sql
SELECT ISNULL(Email, 'No Email Provided') FROM Students;
```

Stored Procedures

SQL Stored Procedures

```sql
CREATE PROCEDURE GetAllStudents
AS
SELECT * FROM Students;
```

Comments

SQL Comments

```sql
-- This is a single-line comment
/* This is a
   multi-line comment */
```

Operators

SQL Operators

```sql
SELECT * FROM Students WHERE Age >= 18 AND Age <= 25;
```

Dates

SQL Dates

```sql
SELECT * FROM Students WHERE BirthDate > '2000-01-01';
```

Views

SQL Views

```sql
CREATE VIEW StudentDetails AS
SELECT FirstName, LastName, Department FROM Students;
```

Security

SQL Injection

Bad:

```
SELECT * FROM Users WHERE Name = '" + userInput + "';
```

Safe:

```
SELECT * FROM Users WHERE Name = @Name;
```