

SQL RDBMS Best Practices and Industry Standards for Developers

1) Create Database

```
create database PracticeSQL;  
use PracticeSQL;
```

2) Create Tables (DDL)

```
CREATE TABLE Students (  
    StudentID INT PRIMARY KEY,  
    Name VARCHAR(30) NOT NULL,  
    Age INT CHECK (Age > 0)  
);  
  
CREATE TABLE Enrollments (  
    EnrollID INT PRIMARY KEY,  
    StudentID INT,  
    Course VARCHAR(30),  
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID)  
);
```

3) Insert Data (DML)

```
INSERT INTO Students VALUES  
(1, 'Arjun', 20),  
(2, 'Priya', 22),  
(3, 'Ravi', 21);  
  
INSERT INTO Enrollments VALUES  
(1, 1, 'SQL'),  
(2, 1, 'C#'),  
(3, 2, 'Java');
```

4) Select (DQL)

```
SELECT * FROM Students;
```

	StudentID	Name	Age
1	1	Arjun	20
2	2	Priya	22
3	3	Ravi	21

5) INNER JOIN

```
SELECT S.Name, E.Course
FROM Students S
INNER JOIN Enrollments E ON S.StudentID = E.StudentID;
```

	Name	Course
1	Arjun	SQL
2	Arjun	C#
3	Priya	Java

6) LEFT JOIN (shows students with no course)

```
SELECT S.Name, E.Course
FROM Students S
LEFT JOIN Enrollments E ON S.StudentID = E.StudentID;
```

	Name	Course
1	Arjun	SQL
2	Arjun	C#
3	Priya	Java
4	Ravi	NULL

7) Subquery (IN)

```
SELECT Name
FROM Students
WHERE StudentID IN (
    SELECT StudentID
    FROM Enrollments
    WHERE Course = 'SQL'
);
```

The screenshot shows a 'Results' tab in SSMS. A single row is displayed in a table with two columns: 'Name' and an index '1'. The value 'Arjun' is shown in the 'Name' column.

	Name
1	Arjun

8) CTE Example

```
WITH CourseCount AS (
    SELECT StudentID, COUNT(*) AS TotalCourses
    FROM Enrollments
    GROUP BY StudentID
)
SELECT S.Name, C.TotalCourses
FROM Students S
JOIN CourseCount C ON S.StudentID = C.StudentID;
```

The screenshot shows a 'Results' tab in SSMS. Two rows are displayed in a table with three columns: 'Name', 'TotalCourses', and an index '1'. The first row has 'Arjun' in 'Name' and '2' in 'TotalCourses'. The second row has 'Priya' in 'Name' and '1' in 'TotalCourses'.

	Name	TotalCourses
1	Arjun	2
2	Priya	1

9) Window Function (ROW_NUMBER)

```
SELECT
    StudentID,
    Course,
    ROW_NUMBER() OVER (PARTITION BY StudentID ORDER BY Course) AS RowNum
FROM Enrollments;
```

The screenshot shows a 'Results' tab in SSMS. Three rows are displayed in a table with four columns: 'StudentID', 'Course', 'RowNum', and an index '1'. The first row has '1' in 'StudentID' and 'C#' in 'Course', with 'RowNum' 1. The second row has '1' in 'StudentID' and 'SQL' in 'Course', with 'RowNum' 2. The third row has '2' in 'StudentID' and 'Java' in 'Course', with 'RowNum' 1.

	StudentID	Course	RowNum
1	1	C#	1
2	1	SQL	2
3	2	Java	1

10) Transaction (COMMIT / ROLLBACK)

```
BEGIN TRANSACTION;
UPDATE Students SET Age = 25 WHERE StudentID = 2;

ROLLBACK;
SELECT * FROM Students WHERE StudentID = 2;
```

Results			
	StudentID	Name	Age
1	2	Priya	22

11) Index Practice

```
CREATE INDEX IX_StudentName ON Students(Name);
SELECT * FROM Students WHERE Name = 'Arjun';
```

Results			
	StudentID	Name	Age
1	1	Arjun	20

12) Update + Delete

```
UPDATE Students SET Age = 23 WHERE StudentID = 1;
SELECT * FROM Students;
```

Results			
	StudentID	Name	Age
1	1	Arjun	23
2	2	Priya	22
3	3	Ravi	21

```
DELETE FROM Enrollments WHERE EnrollID = 3;
SELECT * FROM Enrollments;
```

Results			
	EnrollID	StudentID	Course
1	1	1	SQL
2	2	1	C#

13) PRIMARY KEY And FOREIGN KEY Test

```
INSERT INTO Students VALUES (1, 'Test', 18);

INSERT INTO Enrollments VALUES (4, 99, 'Python');
```

Messages

Msg 208, Level 16, State 1, Line 81
Invalid object name 'Enrollments'.

Completion time: 2026-02-07T11:10:57.1671206+05:30

14) NOT NULL Test

```
INSERT INTO Students VALUES (4, NULL, 20);
```

Messages

Msg 208, Level 16, State 1, Line 83
Invalid object name 'Students'.

Completion time: 2026-02-07T11:18:06.2293750+05:30

15) CHECK Constraint Test

```
INSERT INTO Students VALUES (4, 'Aman', -5);
```

Messages

Msg 208, Level 16, State 1, Line 85
Invalid object name 'Students'.

Completion time: 2026-02-07T11:20:36.0649342+05:30

16) Stored Procedure

```
CREATE PROCEDURE GetStudentByID
@ID INT
AS
BEGIN
    SELECT * FROM Students WHERE StudentID = @ID;
END;

EXEC GetStudentByID 2;
```

The screenshot shows the 'Results' tab of a SQL query window. It displays a table with four columns: StudentID, Name, and Age. There is one row with values 1, Priya, and 22.

	StudentID	Name	Age
1	2	Priya	22

17) Scalar Function

```
CREATE FUNCTION dbo.GetAgeAfter5Years (@Age INT)
RETURNS INT
AS
BEGIN
    RETURN @Age + 5;
END;

SELECT Name, dbo.GetAgeAfter5Years(Age) AS AgeAfter5Years
FROM Students;
```

The screenshot shows the 'Results' tab of a SQL query window. It displays a table with three columns: Name and AgeAfter5Years. There are three rows with values Arjun (28), Priya (27), and Ravi (26).

	Name	AgeAfter5Years
1	Arjun	28
2	Priya	27
3	Ravi	26

18) Cursor Practice

```
DECLARE @StudentName VARCHAR(30);

DECLARE StudentCursor CURSOR FOR
SELECT Name FROM Students;

OPEN StudentCursor;

FETCH NEXT FROM StudentCursor INTO @StudentName;

WHILE @@FETCH_STATUS = 0
BEGIN
    PRINT 'Student: ' + @StudentName;
    FETCH NEXT FROM StudentCursor INTO @StudentName;
END;

CLOSE StudentCursor;
DEALLOCATE StudentCursor;
```

Messages

Student: Arjun
Student: Priya
Student: Ravi

Completion time: 2026-02-07T11:33:36.6236923+05:30