SQL Case Study: Trainings Database

This case study is designed for learners to practice SQL using a sample Trainings table. It covers CRUD operations, aggregations, built-in functions, window functions, subqueries, views, and triggers. Each section provides problem statements and example solutions.

# 1. Table Schema & Data

We will use the following Trainings table:

CREATE TABLE Trainings (  
 TrainingID INT PRIMARY KEY AUTO\_INCREMENT,  
 Title VARCHAR(100),  
 Trainer VARCHAR(50),  
 StartDate DATE,  
 EndDate DATE,  
 DurationDays INT,  
 Location VARCHAR(50),  
 Fees DECIMAL(10,2),  
 IsOnline BOOLEAN  
);

# 2. CRUD Operations

* Insert a new training record

INSERT INTO Trainings (Title, Trainer, StartDate, EndDate, DurationDays, Location, Fees, IsOnline)  
VALUES ('Data Visualization', 'Ritika Sen', '2025-07-01', '2025-07-05', 5, 'Kolkata', 9000, 1);

* Read: Select trainings from Delhi

SELECT \* FROM Trainings WHERE Location = 'Delhi';

* Update: Increase fees by 15% for trainings in Mumbai

UPDATE Trainings SET Fees = Fees \* 1.15 WHERE Location = 'Mumbai';

* Delete: Remove training with ID 5

DELETE FROM Trainings WHERE TrainingID = 5;

# 3. Aggregation Queries

* Count total trainings

SELECT COUNT(\*) AS TotalTrainings FROM Trainings;

* Find average fee

SELECT AVG(Fees) AS AvgFee FROM Trainings;

* Trainings grouped by location

SELECT Location, COUNT(\*) AS TrainingCount FROM Trainings GROUP BY Location;

* Maximum duration training

SELECT MAX(DurationDays) AS MaxDuration FROM Trainings;

# 4. Built-in SQL Functions

* Convert title to uppercase

SELECT UPPER(Title) AS UpperTitle FROM Trainings;

* Extract year from StartDate

SELECT Title, YEAR(StartDate) AS StartYear FROM Trainings;

* Days between start and end

SELECT Title, DATEDIFF(EndDate, StartDate) AS Duration FROM Trainings;

* Concatenate Title and Trainer

SELECT CONCAT(Title, ' by ', Trainer) AS TrainingDetails FROM Trainings;

# 5. Window Functions

* Rank trainings by fees

SELECT Title, Fees, RANK() OVER (ORDER BY Fees DESC) AS FeeRank FROM Trainings;

* Running total of fees by location

SELECT Location, Title, Fees,  
SUM(Fees) OVER (PARTITION BY Location ORDER BY StartDate) AS RunningTotal  
FROM Trainings;

* Compare training fee with location average

SELECT Title, Location, Fees,  
AVG(Fees) OVER (PARTITION BY Location) AS AvgLocFee  
FROM Trainings;

# 6. Subqueries

* Trainings above average fee

SELECT \* FROM Trainings WHERE Fees > (SELECT AVG(Fees) FROM Trainings);

* Trainer with most expensive training

SELECT Trainer FROM Trainings WHERE Fees = (SELECT MAX(Fees) FROM Trainings);

# 7. Views

* View for expensive trainings

CREATE VIEW ExpensiveTrainings AS SELECT Title, Fees FROM Trainings WHERE Fees > 10000;

# 8. Triggers

* Prevent very low fees (<3000)

DELIMITER $$  
CREATE TRIGGER trg\_MinFees  
BEFORE INSERT ON Trainings  
FOR EACH ROW  
BEGIN  
 IF NEW.Fees < 3000 THEN  
 SIGNAL SQLSTATE '45000'  
 SET MESSAGE\_TEXT = 'Fees cannot be less than 3000';  
 END IF;  
END$$  
DELIMITER ;

# 9. Advanced Case-Based Questions for Learners

1. Increase the fees of online trainings by 10%.
2. Find the top 3 highest fee trainings with trainer names.
3. Identify trainers who conducted more than 1 training.
4. Show trainings where duration > average duration.
5. Rank all trainings within each location by duration.
6. Create a view to show only online trainings in 2025.
7. Delete all trainings with duration < 3 days.
8. Find the total revenue generated per location.
9. Using a window function, find the cumulative revenue per trainer.
10. Write a trigger that automatically updates DurationDays if incorrect.