**Case Study 1: Employee Management System**

**Problem:** Retrieve a list of employees who have been working for more than five years.  
**Solution:**

SELECT \* FROM employees

WHERE DATEDIFF(CURDATE(), hire\_date) > (5 \* 365);

**Case Study 2: E-Commerce Sales Analysis**

**Problem:** Find the top 5 best-selling products.  
**Solution:**

SELECT product\_id, COUNT(\*) AS total\_sales

FROM orders

GROUP BY product\_id

ORDER BY total\_sales DESC

LIMIT 5;

**Case Study 3: Customer Retention in an Online Store**

**Problem:** Find customers who have made at least three purchases in the last six months.  
**Solution:**

SELECT customer\_id, COUNT(\*) AS purchase\_count

FROM orders

WHERE order\_date >= DATE\_SUB(CURDATE(), INTERVAL 6 MONTH)

GROUP BY customer\_id

HAVING purchase\_count >= 3;

**Case Study 4: Library Management System**

**Problem:** List books that are currently overdue.  
**Solution:**

SELECT book\_id, user\_id, due\_date

FROM borrowed\_books

WHERE due\_date < CURDATE();

**Case Study 5: Student Grade Analytics**

**Problem:** Find students who scored below the passing mark (50) in any subject.  
**Solution:**

SELECT student\_id, subject\_id, marks

FROM grades

WHERE marks < 50;

**Case Study 6: Bank Transactions Fraud Detection**

**Problem:** Identify accounts with transactions greater than $10,000 in a single day.  
**Solution:**

SELECT account\_id, transaction\_date, SUM(amount) AS total\_amount

FROM transactions

GROUP BY account\_id, transaction\_date

HAVING total\_amount > 10000;

**Case Study 7: Hotel Booking System**

**Problem:** Find rooms that have not been booked in the last three months.  
**Solution:**

SELECT room\_id

FROM rooms

WHERE room\_id NOT IN (

SELECT DISTINCT room\_id

FROM bookings

WHERE booking\_date >= DATE\_SUB(CURDATE(), INTERVAL 3 MONTH)

);

**Case Study 8: Inventory Management**

**Problem:** Identify products that are out of stock.  
**Solution:**

SELECT product\_id, product\_name

FROM products

WHERE stock\_quantity = 0;

**Case Study 9: Healthcare System - Patient Visits**

**Problem:** Find patients who have visited the hospital more than five times in the last year.  
**Solution:**

SELECT patient\_id, COUNT(\*) AS visit\_count

FROM visits

WHERE visit\_date >= DATE\_SUB(CURDATE(), INTERVAL 1 YEAR)

GROUP BY patient\_id

HAVING visit\_count > 5;

**Case Study 10: Social Media User Engagement**

**Problem:** Identify users who have posted more than ten times in the last month.  
**Solution:**

SELECT user\_id, COUNT(\*) AS post\_count

FROM posts

WHERE post\_date >= DATE\_SUB(CURDATE(), INTERVAL 1 MONTH)

GROUP BY user\_id

HAVING post\_count > 10;

**Case Study 11: Car Rental System**

**Problem:** Find cars that have been rented more than 20 times in the last year.  
**Solution:**

SELECT car\_id, COUNT(\*) AS rental\_count

FROM rentals

WHERE rental\_date >= DATE\_SUB(CURDATE(), INTERVAL 1 YEAR)

GROUP BY car\_id

HAVING rental\_count > 20;

**Case Study 12: Online Learning Platform**

**Problem:** Find students who have completed at least 3 courses.  
**Solution:**

SELECT student\_id, COUNT(\*) AS completed\_courses

FROM course\_enrollments

WHERE status = 'Completed'

GROUP BY student\_id

HAVING completed\_courses >= 3;

**Case Study 13: Taxi Ride Analysis**

**Problem:** Identify drivers with an average rating below 3.5.  
**Solution:**

SELECT driver\_id, AVG(rating) AS avg\_rating

FROM rides

GROUP BY driver\_id

HAVING avg\_rating < 3.5;

**Case Study 14: Grocery Store Analytics**

**Problem:** Find the most purchased item in the last month.  
**Solution:**

SELECT product\_id, COUNT(\*) AS purchase\_count

FROM sales

WHERE sale\_date >= DATE\_SUB(CURDATE(), INTERVAL 1 MONTH)

GROUP BY product\_id

ORDER BY purchase\_count DESC

LIMIT 1;

**Case Study 15: Cybersecurity - Failed Login Attempts**

**Problem:** Find users with more than 5 failed login attempts in a day.  
**Solution:**

SELECT user\_id, login\_date, COUNT(\*) AS failed\_attempts

FROM login\_attempts

WHERE status = 'Failed'

GROUP BY user\_id, login\_date

HAVING failed\_attempts > 5;

**Case Study 16: Movie Ticket Booking**

**Problem:** Find the top 3 most watched movies.  
**Solution:**

SELECT movie\_id, COUNT(\*) AS bookings

FROM tickets

GROUP BY movie\_id

ORDER BY bookings DESC

LIMIT 3;

**Case Study 17: Real Estate Management**

**Problem:** Find properties that haven't been rented in the last 6 months.  
**Solution:**

SELECT property\_id

FROM properties

WHERE property\_id NOT IN (

SELECT DISTINCT property\_id

FROM rentals

WHERE rental\_date >= DATE\_SUB(CURDATE(), INTERVAL 6 MONTH)

);

**Case Study 18: Music Streaming Platform**

**Problem:** Identify the most played song in the last week.  
**Solution:**

SELECT song\_id, COUNT(\*) AS play\_count

FROM plays

WHERE play\_date >= DATE\_SUB(CURDATE(), INTERVAL 1 WEEK)

GROUP BY song\_id

ORDER BY play\_count DESC

LIMIT 1;

**Case Study 19: Manufacturing Quality Control**

**Problem:** Identify defective products where the defect rate is above 5%.  
**Solution:**

SELECT product\_id, (COUNT(defect\_id) / COUNT(\*)) \* 100 AS defect\_rate

FROM product\_quality

GROUP BY product\_id

HAVING defect\_rate > 5;

**Case Study 20: Airline Ticketing System**

**Problem:** Identify flights that have been booked to full capacity.  
**Solution:**

SELECT flight\_id

FROM bookings

GROUP BY flight\_id

HAVING COUNT(passenger\_id) = (SELECT seat\_capacity FROM flights WHERE flights.flight\_id = bookings.flight\_id);

-- 1. Employee Management System

CREATE TABLE employees (

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

hire\_date DATE NOT NULL,

department VARCHAR(50) NOT NULL,

salary DECIMAL(10,2) CHECK (salary > 0)

);

INSERT INTO employees (name, hire\_date, department, salary) VALUES

('Alice Johnson', '2015-06-15', 'HR', 55000),

('Bob Smith', '2017-04-10', 'IT', 75000),

('Charlie Brown', '2012-09-20', 'Finance', 80000),

('David Wilson', '2019-03-25', 'Marketing', 60000),

('Emily White', '2020-08-30', 'IT', 70000);

-- 2. E-Commerce Sales Analysis

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_id INT NOT NULL,

customer\_id INT NOT NULL,

order\_date DATE NOT NULL,

quantity INT CHECK (quantity > 0)

);

INSERT INTO orders (product\_id, customer\_id, order\_date, quantity) VALUES

(1, 101, '2023-01-10', 2),

(2, 102, '2023-02-15', 5),

(3, 103, '2023-03-20', 3),

(1, 104, '2023-04-05', 4),

(2, 105, '2023-05-10', 1);

-- 3. Customer Retention

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL

);

INSERT INTO customers (name, email) VALUES

('John Doe', 'john.doe@email.com'),

('Jane Smith', 'jane.smith@email.com'),

('Michael Brown', 'michael.brown@email.com'),

('Emily Davis', 'emily.davis@email.com'),

('Chris Wilson', 'chris.wilson@email.com');

-- 4. Library Management System

CREATE TABLE borrowed\_books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

due\_date DATE NOT NULL

);

INSERT INTO borrowed\_books (user\_id, due\_date) VALUES

(201, '2023-12-10'),

(202, '2024-01-05'),

(203, '2024-01-12'),

(204, '2024-02-01'),

(205, '2023-11-15');

-- 5. Student Grade Analytics

CREATE TABLE grades (

student\_id INT NOT NULL,

subject\_id INT NOT NULL,

marks INT CHECK (marks BETWEEN 0 AND 100),

PRIMARY KEY (student\_id, subject\_id)

);

INSERT INTO grades (student\_id, subject\_id, marks) VALUES

(301, 101, 85),

(302, 102, 78),

(303, 103, 90),

(304, 104, 45),

(305, 105, 55);

-- 6. Bank Transactions

CREATE TABLE transactions (

transaction\_id INT AUTO\_INCREMENT PRIMARY KEY,

account\_id INT NOT NULL,

transaction\_date DATE NOT NULL,

amount DECIMAL(10,2) CHECK (amount > 0)

);

INSERT INTO transactions (account\_id, transaction\_date, amount) VALUES

(401, '2023-01-10', 5000),

(402, '2023-02-20', 12000),

(403, '2023-03-25', 8000),

(404, '2023-04-30', 2000),

(405, '2023-05-05', 15000);

-- 7. Hotel Booking System

CREATE TABLE bookings (

booking\_id INT AUTO\_INCREMENT PRIMARY KEY,

room\_id INT NOT NULL,

booking\_date DATE NOT NULL

);

INSERT INTO bookings (room\_id, booking\_date) VALUES

(501, '2023-06-01'),

(502, '2023-07-15'),

(503, '2023-08-20'),

(504, '2023-09-10'),

(505, '2023-10-05');

-- 8. Inventory Management

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100) NOT NULL,

stock\_quantity INT CHECK (stock\_quantity >= 0)

);

INSERT INTO products (product\_name, stock\_quantity) VALUES

('Laptop', 10),

('Phone', 0),

('Tablet', 5),

('Monitor', 7),

('Keyboard', 0);

-- 9. Healthcare System

CREATE TABLE visits (

visit\_id INT AUTO\_INCREMENT PRIMARY KEY,

patient\_id INT NOT NULL,

visit\_date DATE NOT NULL

);

INSERT INTO visits (patient\_id, visit\_date) VALUES

(601, '2023-01-10'),

(602, '2023-02-20'),

(603, '2023-03-15'),

(604, '2023-04-05'),

(605, '2023-05-25');

-- 10. Social Media Engagement

CREATE TABLE posts (

post\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

post\_date DATE NOT NULL

);

INSERT INTO posts (user\_id, post\_date) VALUES

(701, '2023-01-01'),

(702, '2023-02-14'),

(703, '2023-03-18'),

(704, '2023-04-21'),

(705, '2023-05-30');

-- 11. Car Rental System

CREATE TABLE rentals (

rental\_id INT AUTO\_INCREMENT PRIMARY KEY,

car\_id INT NOT NULL,

rental\_date DATE NOT NULL

);

INSERT INTO rentals (car\_id, rental\_date) VALUES

(801, '2023-01-05'),

(802, '2023-02-10'),

(803, '2023-03-15'),

(804, '2023-04-20'),

(805, '2023-05-25');

-- 12. Online Learning Platform

CREATE TABLE course\_enrollments (

enrollment\_id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id INT NOT NULL,

course\_id INT NOT NULL,

status ENUM('Enrolled', 'Completed') NOT NULL

);

INSERT INTO course\_enrollments (student\_id, course\_id, status) VALUES

(901, 201, 'Completed'),

(902, 202, 'Completed'),

(903, 203, 'Enrolled'),

(904, 204, 'Completed'),

(905, 205, 'Enrolled');

-- 13. Taxi Ride Analysis

CREATE TABLE rides (

ride\_id INT AUTO\_INCREMENT PRIMARY KEY,

driver\_id INT NOT NULL,

rating DECIMAL(2,1) CHECK (rating BETWEEN 1 AND 5)

);

INSERT INTO rides (driver\_id, rating) VALUES

(1001, 4.5),

(1002, 3.2),

(1003, 2.8),

(1004, 4.0),

(1005, 3.7);

-- 14. Grocery Store

CREATE TABLE sales (

sale\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_id INT NOT NULL,

sale\_date DATE NOT NULL

);

INSERT INTO sales (product\_id, sale\_date) VALUES

(1, '2023-01-15'),

(2, '2023-02-10'),

(3, '2023-03-05'),

(4, '2023-04-20'),

(5, '2023-05-30');