Title: SQL Query, Database Security Assessment & Audit

Session Duration: 1 Hour

Information Systems Audit

Part 1: SQL Fundamentals (45 minutes)

1. Introduction to Databases & SQL

- **Database:** A structured collection of data that can be accessed, managed, and updated easily.
- **SQL (Structured Query Language):** A standardized language used to interact with relational databases.
- **RDBMS Examples:** MySQL, Oracle, SQL Server, PostgreSQL.

Example:

```
SELECT * FROM Customers;
```

This retrieves all records from the `Customers` table.

---

2. SQL Architecture

- **Database:** Container of data.
- **Schema:** Logical grouping of database objects.
- **Tables:** Store data in rows and columns.
- **Index:** Speeds up data retrieval.

Example:

```
CREATE TABLE Employees (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(50),
    Salary DECIMAL(10,2)
);
```

---

3. SQL Query Categories

- **DDL (Data Definition Language):** CREATE, ALTER, DROP
- **DML (Data Manipulation Language):** INSERT, UPDATE, DELETE
- **DQL (Data Query Language):** SELECT

- **DCL (Data Control Language):** GRANT, REVOKE
- **TCL (Transaction Control Language):** COMMIT, ROLLBACK

Example:

```
CREATE TABLE Students (StudentID INT, Name VARCHAR(50));
INSERT INTO Students VALUES (1, 'Ali');
SELECT * FROM Students;
```

---

4. CRUD Operations

- **Create:** Add data
- **Read:** Retrieve data
- **Update:** Modify data
- **Delete:** Remove data

Example:

```
INSERT INTO Students VALUES (2, 'Sara');
UPDATE Students SET Name='Sarah' WHERE StudentID=2;
DELETE FROM Students WHERE StudentID=2;
```

---

5. Filtering, Sorting & Aggregation

- **WHERE:** Filter rows
- **ORDER BY:** Sort results
- **GROUP BY:** Aggregate data

Example:

```
SELECT Grade, COUNT(*) FROM Students GROUP BY Grade;
SELECT * FROM Students WHERE Age > 12 ORDER BY Name ASC;
```

---

6. Joins & Subqueries

- **INNER JOIN:** Returns matching records
- **LEFT JOIN:** All from left + matches from right
- **Subquery:** Nested SELECT inside another query

Example:

```
SELECT s.Name, c.ClassName
FROM Students s
```

```
JOIN Classes c ON s.StudentID = c.StudentID;

SELECT Name FROM Students
WHERE Age > (SELECT AVG(Age) FROM Students);
```

---

7. Views & Functions

- **Views:** Virtual tables for security and simplicity
- **Functions:** Aggregation or string manipulation

Example:

```
CREATE VIEW HighPerformers AS
SELECT Name, Grade FROM Students WHERE Grade='A';

SELECT UPPER(Name), LENGTH(Name) FROM Students;
```

Practical Lab 1: SQL Query Practice

Tasks: 1. Create a table `Employees`. 2. Insert 5 records. 3. Retrieve names of employees earning more than 50,000. 4. Group employees by department. 5. Use a subquery to find employees earning above the average.

Part 2: Database Security Assessment (35 minutes)

1. Database Security Overview

- Databases store sensitive information (e.g., financial, personal data).
- A compromise can lead to financial loss, reputational damage, and regulatory violations.

Example Incident: A bank's database exposed 1 million customer records due to unpatched SQL Server.

---

2. Common Threats & Vulnerabilities

Threat	Description	Mitigation
SQL Injection	Malicious code injected into queries	Use parameterized queries
Weak Authentication	Default credentials used	Enforce strong password policy
Misconfiguration	Open ports, no encryption	Apply CIS benchmarks
Backup Leakage	Unencrypted backups stolen	Encrypt backups & limit access

SQL Injection Example:

```sql
SELECT * FROM Users WHERE Username='admin' AND Password='1234' OR '1'='1';
```

Mitigate with prepared statements.

---

3. Database Hardening Practices

1. Implement **Role-Based Access Control (RBAC)**.
2. Disable default accounts.
3. Enforce **TLS/SSL encryption** for connections.
4. Apply latest **patches and updates**.
5. Configure **audit trails** for all changes.

---

4. Audit Logs & Monitoring

- Enable query logging (SELECT, INSERT, UPDATE, DELETE).
- Monitor for failed login attempts.
- Store logs in secure, centralized location.

Example: Audit finding: Multiple failed login attempts from the same IP. Recommendation: Implement account lockout and review user activity.

Part 3: Database Audit Process (25 minutes)

1. Importance of Database Audit

- Ensures **integrity**, **confidentiality**, and **availability**.
- Identifies unauthorized changes and potential data breaches.

---

2. Audit Scope & Objectives

- Evaluate database configuration.
- Review user roles and privileges.
- Check for patch and version compliance.
- Validate logging and monitoring mechanisms.

---

3. Steps in Database Security Audit

1. **Planning:** Define audit scope.
2. **Information Gathering:** Review architecture & policies.
3. **Testing:** Run queries and assess configuration.
4. **Reporting:** Document findings and recommendations.

Example Finding: DBA has full OS-level access. **Recommendation:** Segregate duties between DBA and SysAdmin.

4. Regulatory Compliance References

- **ISO 27001:** A.12.4 – Logging and Monitoring.
- **PCI DSS:** Requirement 10 – Track and monitor access.
- **Bangladesh Bank Guidelines:** IT Governance Circulars.

---

5. Case Study: SQL Injection Exploitation & Mitigation

Scenario: Banking web app vulnerable to injection via login form. **Exploit Query:** `' OR 1=1--` **Mitigation:** Input validation, least privilege, and web application firewall (WAF).

Part 4: SQL Lab Exercises (15 minutes)

Lab 1: Create database `BankDB` and add `Accounts` & `Transactions` tables. **Lab 2:** Insert data, then retrieve all transactions > 100,000. **Lab 3:** Detect duplicate account numbers. **Lab 4:** Create audit view listing users with admin roles. **Lab 5:** Simulate SQL injection and demonstrate prevention.

• Continuous learning and testing improve both security posture and query optimization.

Instructor Note: Include live demo of 5 SQL queries and a short vulnerability test (SQL injection simulation) to engage students.
