Admissions & Support:
support@mockattempt.com
Phone / WhatsApp: +91 98765 43210
Course Code: MYSQ • **Updated:** 03 Nov 2026
https://mockattempt.com/courses/mysql-training

MYSQ • RELATIONAL DATABASES & SQL ENGINEERING **BEGINNER TO PRODUCTION MYSQL & INNODB ARCHITECT**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

MYSQL Training

Master relational SQL & MySQL from 3NF schema design, constraints, JOINs, GROUP BY/HAVING & Recursive CTEs to Window Functions, Clustered B+Tree EXPLAIN tuning, Triggers, JSON & MVCC ACID Transactions — with runnable SQL scripts, 386 Topic Mock Tests (17,860 MCQs), and Spoken Voice + MockAttempt AI Interviews at ₹1,499 (Incl. 18% GST).

Duration: 15–17 Weeks • Live + Hands-On Projects (212 Hrs) • **Modules:** 16 • **Live Topics:** 53 • **Subtopics:** 424+ • **Labs & Tasks:** 106 • **Mocks / AI Vivas:** 386 / 386 • **Projects:** 16 Module + 1 Live Client Prod

All-Inclusive Direct Fee (Incl. 18% GST • Zero Hidden Charges): Total Payable: ₹1,499 (Net Tuition ₹1,270 + Included 18% GST ₹229) • **Split Option:** 100% Credit to Job Track • **Effective ROI:** ~₹10/Guided Hr (~₹28/Live Topic)

Faculty: 100% Active Working IT Professionals • **Live Batch Tracks (7:00 AM – 8:00 PM IST Flexible Slots):** Weekday Live Online Batch (Mon – Fri • 1 Hr/Day • 7:00 AM – 8:00 PM) (Monday to Friday (1 Hour Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 1-Hour Slot Based on Trainer Availability) • 18–20 Weeks Fast-Track) | Weekend Live Online Batch (Sat – Sun • 2 Hrs/Day • 7:00 AM – 8:00 PM) (Saturday & Sunday (2 Hours Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 2-Hour Slot Based on Trainer Availability) • 20–22 Weeks Weekend Track)

✓ **Trainer LeetCode & HackerRank Mentorship (SQL & Stored Procedures (LeetCode SQL 50 & HackerRank SQL)): Live trainer support to crack LeetCode & HackerRank problems, optimize complexity, and handle edge cases.**

Core Technologies & Engineering Stack: MySQL 8.0+ • InnoDB & MVCC • Complex JOINs & CTEs • Window Analytics • B+Tree Indexing & EXPLAIN • Procedures & Triggers • ACID & Backups

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Design normalized 3NF relational schemas using utf8mb4, exact DECIMAL financial types, Foreign Keys, and CHECK constraints
2. Write complex analytical SQL using INNER/LEFT/SELF JOINs, GROUP BY WITH ROLLUP, Recursive CTEs, and Window Ranking Functions
3. Eliminate slow full-table scans (type: ALL) using Composite Covering B+Tree Indexes, SARGable predicates, and EXPLAIN ANALYZE
4. Clear 386 Timed Mock Exams (17,500 MySQL MCQs) and 350 Spoken Voice + MockAttempt AI SQL & Database Technical Interviews
5. Normalized 3NF transactional schema design balanced with strategic denormalization for high-read reporting.
6. Crack LeetCode and HackerRank problems in SQL & Stored Procedures (LeetCode SQL 50 & HackerRank SQL) with live trainer mentorship on algorithmic/query patterns, edge cases, and optimal complexity.
7. InnoDB clustered primary keys, covering composite B-Tree indexes, and `EXPLAIN ANALYZE` execution plan mastery.
8. Advanced analytical SQL: CTEs, Window Functions (`ROW_NUMBER`, `LAG`, `SUM OVER`), JSON columns, and partitioning.

Eligibility & Prerequisites: No prior SQL or database experience required — starts from relational tables & installation from scratch • Any laptop/PC running MySQL 8 / XAMPP MariaDB / MySQL Workbench • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews. • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards.

2. DETAILED ZERO-TO-EXPERT MODULE, TOPIC & SUBTOPIC SYLLABUS (16 MODULES • 53 LIVE TOPICS • 424+ SUBTOPICS)

Progressive Zero-to-Expert Mastery Stages: Modules 1–4: Stage 1: Day-1 Foundations & Core Mental Models (16 Topics / 128 Subtopics) • Modules 5–8: Stage 2: Applied Engineering, Data & API Workflows (13 Topics / 104 Subtopics) • Modules 9–12: Stage 3: Internals, Concurrency, Security & Performance (12 Topics / 96 Subtopics) • Modules 13–16: Stage 4: Production Architecture, Testing, CI/CD & Expert Mastery (12 Topics / 96 Subtopics)

MODULE 1 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: MySQL 8 Architecture, Storage Engines (`InnoDB`), Data Types & `utf8mb4` Databases

Understand MySQL's client/optimizer/storage-engine architecture, why InnoDB dominates OLTP over MyISAM, choose exact financial & temporal data types, and provision full-Unicode `utf8mb4` databases.

Objectives: ✓ MySQL Architecture — Relational Model, Cost-Based Optimizer & InnoDB Buffer Pool Lifecycle — Relational tuples/attributes/keys, SQL sub-languages (`DDL`, `DML`, `DQL`, `DCL`, `TCL`), MySQL 8's removal of the legacy Query Cache, and how the InnoDB Buffer Pool serves hot 16 KB pages from RAM. • ✓ MySQL Storage Engines — `InnoDB` (ACID, Row Locks, FK) vs `MyISAM`, `MEMORY` & `CSV` — Pluggable storage engine architecture (`SHOW ENGINES`), why `InnoDB` is the default transactional engine (row-level locking, MVCC, clustered PK, Foreign Keys, crash recovery via redo log), and when `MEMORY` / `ARCHIVE` appear. • ✓ MySQL Data Types — `BIGINT`, Exact `DECIMAL` vs `FLOAT`, `VARCHAR` vs `CHAR`, `DATETIME` vs `TIMESTAMP` — Choosing the smallest accurate type: integer widths (`TINYINT` to `BIGINT UNSIGNED`), why `DECIMAL(12,2)` prevents binary floating-point rounding bugs (`0.1 + 0.2 != 0.3`), `VARCHAR(N)` character length, and `TIMESTAMP` (UTC-converted, 2038 limit) vs `DATETIME` (timezone-naive, year 9999). • ✓ Database DDL & Character Sets — `CREATE DATABASE IF NOT EXISTS`, `utf8mb4` vs `utf8mb3` & Collations — Creating/selecting/dropping schemas safely (`CREATE DATABASE IF NOT EXISTS ... CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci`), why historical MySQL `utf8` truncated 4-byte emojis (`Incorrect string value`), and case-insensitive `_ci` vs case-sensitive `_cs`/`_bin` collations. • ✓ Build, verify, and ship Module Capstone: Multi-Currency EdTech Database Blueprint (`utf8mb4` + Precision `DECIMAL` + `InnoDB`).

1.1 MySQL Architecture — Relational Model, Cost-Based Optimizer & InnoDB Buffer Pool Lifecycle

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: DDL / DML / DQL / DCL / TCL & Cost-Based Query Optimizer • Core Syntax, CLI & Production APIs: SELECT VERSION(), SHOW VARIABLES LIKE "innodb_buffer_pool_size", DDL / DML / DQL / DCL / TCL • Deep-Dive Mechanics: InnoDB 16 KB pages & Buffer Pool & MySQL 8 removed Query Cache • Advanced Patterns: Client-Server 3306 protocol & RDBMS vs NoSQL • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Inspecting MySQL Version, Current Database & Session SQL Mode • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for MySQL Architecture

Purpose: To understand and apply MySQL Architecture — Relational Model, Cost-Based Optimizer & InnoDB Buffer Pool Lifecycle in real MYSQL systems — specifically: Relational tuples...

What Topic Is Next → Next Topic 1.2: MySQL Storage Engines — `InnoDB` (ACID, Row Locks, FK) vs `MyISAM`, `MEMORY` & `CSV`

1.2 MySQL Storage Engines — `InnoDB` (ACID, Row Locks, FK) vs `MyISAM`, `MEMORY` & `CSV`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: InnoDB row-level locking & MyISAM table-lock bottleneck • Core Syntax, CLI & Production APIs: CREATE TABLE ... ENGINE=InnoDB; SHOW ENGINES; ALTER TABLE t ENGINE=InnoDB; • Deep-Dive Mechanics: ACID redo & undo logs & Foreign Key support • Advanced Patterns: SHOW ENGINES inspection & ALTER TABLE ENGINE=InnoDB • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Creating an Explicit `InnoDB` Table & Verifying Engine Metadata • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for MySQL Storage Engines

Purpose: To understand and apply MySQL Storage Engines — `InnoDB` (ACID, Row Locks, FK) vs `MyISAM`, `MEMORY` & `CSV` in real MYSQL systems — specifically: Pluggable storage engin...

What Topic Is Next → Next Topic 1.3: MySQL Data Types — `BIGINT`, Exact `DECIMAL` vs `FLOAT`, `VARCHAR` vs `CHAR`, `DATETIME` vs `TIMESTAMP`

1.3 MySQL Data Types — `BIGINT`, Exact `DECIMAL` vs `FLOAT`, `VARCHAR` vs `CHAR`, `DATETIME` vs `TIMESTAMP`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: DECIMAL(p,s) vs FLOAT rounding & TIMESTAMP UTC vs DATETIME • Core Syntax, CLI & Production APIs: BIGINT UNSIGNED, DECIMAL(12, 2), VARCHAR(191), DATETIME(3), TIMESTAMP, TINYINT(1) • Deep-Dive Mechanics: VARCHAR variable bytes vs CHAR & UNSIGNED integer range doubling • Advanced Patterns: VARCHAR(N) counts characters & STRICT_TRANS_TABLES overflow guard • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Demonstrating Exact `DECIMAL(10,2)` Financial Arithmetic Precision • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for MySQL Data Types

Purpose: To understand and apply MySQL Data Types — `BIGINT`, Exact `DECIMAL` vs `FLOAT`, `VARCHAR` vs `CHAR`, `DATETIME` vs `TIMESTAMP` in real MYSQL systems — specifically: Choo...

What Topic Is Next → Next Topic 1.4: Database DDL & Character Sets — `CREATE DATABASE IF NOT EXISTS`, `utf8mb4` vs `utf8mb3` & Collations

1.4 Database DDL & Character Sets — `CREATE DATABASE IF NOT EXISTS`, `utf8mb4` vs `utf8mb3` & Collations

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Utf8mb4 4-byte full Unicode & Legacy utf8mb3 emoji truncation • Core Syntax, CLI & Production APIs: CREATE DATABASE IF NOT EXISTS app_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci; USE a... • Deep-Dive Mechanics: Utf8mb4_0900_ai_ci vs _bin & CREATE DATABASE IF NOT EXISTS • Advanced Patterns: Matching JOIN column collations & SHOW CREATE DATABASE • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Provisioning a Full-Unicode `utf8mb4` Database Safely • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Database DDL

Purpose: To understand and apply Database DDL & Character Sets — `CREATE DATABASE IF NOT EXISTS`, `utf8mb4` vs `utf8mb3` & Collations in real MYSQL systems — specifically: Creatin...

What Topic Is Next → Module 1 Capstone (Multi-Currency EdTech Database Blueprint (`utf8mb4` + Precision `DECIMAL` + `InnoDB`)) → Next Topic 2.1: `CREATE TABLE`, `AUTO_INCREMENT`, `TEMPORARY` Tables & 1NF–3NF Relational Normalization

Module 1 Hands-On Capstone Project (100 Marks): Multi-Currency EdTech Database Blueprint (`utf8mb4` + Precision `DECIMAL` + `InnoDB`) — Provision a production-grade `mockattempt_edtech` MySQL 8 database using `CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci` with `courses`, `coupons`, and `payments` tables using exact `DECI...

MODULE 2 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: Table Schema Design, Online `ALTER TABLE` & Relational Integrity Constraints

Architect normalized 3NF tables with `CREATE TABLE`, evolve live production schemas safely with `ALTER TABLE ALGORITHM=INSTANT`, and enforce ironclad data quality via `UNIQUE`, `CHECK`, and `FOREIGN KEY` constraints.

Objectives: ✓ `CREATE TABLE`, `AUTO_INCREMENT`, `TEMPORARY` Tables & 1NF–3NF Relational Normalization — Authoring self-documenting `CREATE TABLE` DDL (`PRIMARY KEY`, `DEFAULT CURRENT_TIMESTAMP`, `ON UPDATE CURRENT_TIMESTAMP`, `COMMENT`), session-scoped `CREATE TEMPORARY TABLE`, and eliminating update anomalies via 1NF, 2NF, and 3NF. • ✓ Online DDL Evolution — `ALTER TABLE`, `ALGORITHM=INSTANT/INPLACE` & `TRUNCATE` vs `DELETE` vs `DROP` — Adding/renaming/modifying columns on live multi-gigabyte tables without downtime (`ALGORITHM=INSTANT` in MySQL 8.0.12+), why `ALGORITHM=COPY` locks writes and doubles disk space, and comparing `TRUNCATE TABLE` vs `DELETE FROM`. • ✓ Relational Integrity Constraints — `NOT NULL`, `UNIQUE` (Multiple NULLs!), `CHECK` & `DEFAULT` — Declarative business invariants inside the database engine: named `CONSTRAINT chk_fee CHECK (discount_price <= price)` (enforced since MySQL 8.0.16!), composite `UNIQUE (group_id, user_id)`, and how `UNIQUE` treats `NULL`. • ✓ Primary Keys, Composite Keys & Foreign Keys — `ON DELETE CASCADE`, `RESTRICT` & `SET NULL` — Referential integrity between parent and child InnoDB tables, matching exact data types & `UNSIGNED` attributes (`ERROR 3780: Referencing column and referenced column... are incompatible`), and choosing `RESTRICT` vs `CASCADE` vs `SET NULL`. • ✓ Build, verify, and ship Module Capstone: 3NF University & Cohort Enrollment Schema with Enforced `CHECK` & `FOREIGN KEY` Cascades.

2.1 `CREATE TABLE`, `AUTO_INCREMENT`, `TEMPORARY` Tables & 1NF–3NF Relational Normalization

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: 1NF atomic columns (no CSV lists) & 2NF & 3NF anomaly elimination • Core Syntax, CLI & Production APIs: CREATE TABLE courses (id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY, ...); DESCRIBE courses; • Deep-Dive Mechanics: AUTO_INCREMENT persistence in 8.0 & DEFAULT & ON UPDATE CURRENT_TIMESTAMP • Advanced Patterns: SHOW CREATE TABLE & Session TEMPORARY tables • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Normalized `courses` Table DDL with Audit Timestamps • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for `CREATE TABLE`

Purpose: To understand and apply `CREATE TABLE`, `AUTO_INCREMENT`, `TEMPORARY` Tables & 1NF–3NF Relational Normalization in real MYSQL systems — specifically: Authoring self-docum...

What Topic Is Next → Next Topic 2.2: Online DDL Evolution — `ALTER TABLE`, `ALGORITHM=INSTANT/INPLACE` & `TRUNCATE` vs `DELETE` vs `DROP`

2.2 Online DDL Evolution — `ALTER TABLE`, `ALGORITHM=INSTANT/INPLACE` & `TRUNCATE` vs `DELETE` vs `DROP`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ALGORITHM=INSTANT metadata-only DDL & ALGORITHM=INPLACE vs COPY rebuild • Core Syntax, CLI & Production APIs: ALTER TABLE users ADD COLUMN bio VARCHAR(255) NULL, ALGORITHM=INSTANT; TRUNCATE TABLE logs; • Deep-Dive Mechanics: TRUNCATE (DDL) vs DELETE (DML) & TRUNCATE implicit COMMIT • Advanced Patterns: RENAME COLUMN in MySQL 8 & Online schema migration tools • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Adding a Column Online & Comparing `TRUNCATE` Auto-Increment Reset • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Online DDL Evolution

Purpose: To understand and apply Online DDL Evolution — `ALTER TABLE`, `ALGORITHM=INSTANT/INPLACE` & `TRUNCATE` vs `DELETE` vs `DROP` in real MYSQL systems — specifically: Adding/...

What Topic Is Next → Next Topic 2.3: Relational Integrity Constraints — `NOT NULL`, `UNIQUE` (Multiple NULLs!), `CHECK` & `DEFAULT`

2.3 Relational Integrity Constraints — `NOT NULL`, `UNIQUE` (Multiple NULLs!), `CHECK` & `DEFAULT`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: UNIQUE permits multiple NULLs & MySQL 8.0.16+ enforced CHECK • Core Syntax, CLI & Production APIs: CONSTRAINT uq_batch_student UNIQUE (group_id, user_id), CONSTRAINT chk_score CHECK (score BE... • Deep-Dive Mechanics: Composite UNIQUE (col_a, col_b) & NOT NULL storage & optimizer win • Advanced Patterns: Named CONSTRAINT diagnostics & Application check TOCTOU race • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Enforced Named `CHECK` & Composite `UNIQUE` Constraints in MySQL 8 • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Relational Integrity Constraints

Purpose: To understand and apply Relational Integrity Constraints — `NOT NULL`, `UNIQUE` (Multiple NULLs!), `CHECK` & `DEFAULT` in real MYSQL systems — specifically: Declarative b...

What Topic Is Next → Next Topic 2.4: Primary Keys, Composite Keys & Foreign Keys — `ON DELETE CASCADE`, `RESTRICT` & `SET NULL`

2.4 Primary Keys, Composite Keys & Foreign Keys — `ON DELETE CASCADE`, `RESTRICT` & `SET NULL`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Exact type & UNSIGNED match rule & ON DELETE RESTRICT vs CASCADE • Core Syntax, CLI & Production APIs: CONSTRAINT fk_mod_course FOREIGN KEY (course_id) REFERENCES courses(id) ON DELETE CASCADE • Deep-Dive Mechanics: ON DELETE SET NULL & Composite Primary Keys • Advanced Patterns: Orphan row prevention & Supporting FK B+Tree index • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Parent-Child Referential Integrity with `ON DELETE CASCADE` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Primary Keys

Purpose: To understand and apply Primary Keys, Composite Keys & Foreign Keys — `ON DELETE CASCADE`, `RESTRICT` & `SET NULL` in real MYSQL systems — specifically: Referential integ...

What Topic Is Next → Module 2 Capstone (3NF University & Cohort Enrollment Schema with Enforced `CHECK` & `FOREIGN KEY` Cascades) → Next Topic 3.1: Bulk `INSERT`, `INSERT ... SELECT` & Atomic Upserts (`ON DUPLICATE KEY UPDATE` vs `REPLACE`)

Module 2 Hands-On Capstone Project (100 Marks): 3NF University & Cohort Enrollment Schema with Enforced `CHECK` & `FOREIGN KEY` Cascades — Design a 5-table 3NF schema (`departments`, `courses`, `batches`, `students`, `batch_enrollments`) enforcing composite `UNIQUE (batch_id, student_id)`, `CHECK (discount_price <= price)`, `ON...

MODULE 3 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: DML Mutations, Atomic Upserts, `SELECT` Filtering & Three-Valued `NULL` Logic

Master high-throughput multi-row `INSERT`s, atomic `ON DUPLICATE KEY UPDATE` upserts, safe `UPDATE`/`DELETE` guards, `WHERE` operator precedence, and SQL's three-valued `TRUE / FALSE / UNKNOWN` `NULL` logic.

Objectives: ✓ Bulk `INSERT`, `INSERT ... SELECT` & Atomic Upserts (`ON DUPLICATE KEY UPDATE` vs `REPLACE`) — Multi-row `INSERT` batching (10x–30x faster than single-row loops), atomic `INSERT ... ON DUPLICATE KEY UPDATE` (MySQL 8 `AS alias` syntax), why `REPLACE INTO` is dangerous (it `DELETE`s + re-`INSERT`s, changing PK & firing delete cascades!), and `LAST_INSERT_ID()`. • ✓ Safe `UPDATE` & `DELETE` Mutations — `sql_safe_updates`, `ORDER BY ... LIMIT` Batching & Multi-Table Joins — Targeted `UPDATE`/`DELETE`, enabling `SET sql_safe_updates = 1` so keyless updates error out harmlessly, chunking massive deletes (`DELETE FROM logs WHERE created_at < ? ORDER BY id LIMIT 5000`) to avoid giant undo locks, and `UPDATE ... JOIN`. • ✓ `SELECT` Filtering — `WHERE`, `AND` vs `OR` Precedence, `BETWEEN`, `IN`, `LIKE` Wildcards & `LIMIT` — Boolean operator precedence (`NOT` > `AND` > `OR`!), inclusive `BETWEEN a AND b`, `IN (...)` list lookups, prefix `LIKE "mysq%"` (uses B+Tree index!) vs leading-wildcard `LIKE "%sql%"` (full scan!), `DISTINCT`, and `ORDER BY ... LIMIT`. • ✓ Three-Valued `NULL` Logic — `IS NULL`, `IS NOT NULL`, Null-Safe `<=>` Equal & `COALESCE()` — Why `NULL` means "unknown/missing" rather than zero or empty string `""`, why `col = NULL` and `col != NULL` ALWAYS evaluate to `UNKNOWN` (returning 0 rows!), MySQL's null-safe equality operator `<=>`, and `COALESCE(a, b, 0)`. • ✓ Build, verify, and ship Module Capstone: Idempotent Daily Attendance & Leaderboard Upsert Pipeline.

3.1 Bulk `INSERT`, `INSERT ... SELECT` & Atomic Upserts (`ON DUPLICATE KEY UPDATE` vs `REPLACE`)

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Multi-row INSERT batching & ON DUPLICATE KEY UPDATE atomic upsert • Core Syntax, CLI & Production APIs: INSERT INTO stats (uid, hits) VALUES (1, 1) AS incoming ON DUPLICATE KEY UPDATE hits = stats... • Deep-Dive Mechanics: MySQL 8 AS row_alias syntax & REPLACE INTO delete+insert hazard • Advanced Patterns: Connection-safe LAST_INSERT_ID() & INSERT INTO ... SELECT • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Atomic Score High-Water-Mark Upsert Using `ON DUPLICATE KEY UPDATE` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Bulk `INSERT`

Purpose: To understand and apply Bulk `INSERT`, `INSERT ... SELECT` & Atomic Upserts (`ON DUPLICATE KEY UPDATE` vs `REPLACE`) in real MYSQL systems — specifically: Multi-row `INSE...

What Topic Is Next → Next Topic 3.2: Safe `UPDATE` & `DELETE` Mutations — `sql_safe_updates`, `ORDER BY ... LIMIT` Batching & Multi-Table Joins

3.2 Safe `UPDATE` & `DELETE` Mutations — `sql_safe_updates`, `ORDER BY ... LIMIT` Batching & Multi-Table Joins

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Sql_safe_updates disaster guard & Chunked DELETE ... LIMIT 5000 • Core Syntax, CLI & Production APIs: UPDATE courses SET discount_price = 999 WHERE slug = ?; DELETE FROM logs WHERE id < ? LIMIT... • Deep-Dive Mechanics: UPDATE ... INNER JOIN syntax & ROW_COUNT() affected rows • Advanced Patterns: Wrapping manual DML in START TRANSACTION & Avoiding unindexed WHERE locks • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Targeted Conditional `UPDATE` & Checking Affected `ROW_COUNT()` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Safe `UPDATE`

Purpose: To understand and apply Safe `UPDATE` & `DELETE` Mutations — `sql_safe_updates`, `ORDER BY ... LIMIT` Batching & Multi-Table Joins in real MYSQL systems — specifically: T...

What Topic Is Next → Next Topic 3.3: `SELECT` Filtering — `WHERE`, `AND` vs `OR` Precedence, `BETWEEN`, `IN`, `LIKE` Wildcards & `LIMIT`

3.3 `SELECT` Filtering — `WHERE`, `AND` vs `OR` Precedence, `BETWEEN`, `IN`, `LIKE` Wildcards & `LIMIT`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: AND binds tighter than OR & Inclusive BETWEEN vs half-open dates • Core Syntax, CLI & Production APIs: SELECT id, title FROM courses WHERE is_published = 1 AND (category = "SQL" OR discount_price... • Deep-Dive Mechanics: Prefix LIKE "abc%" index range & Leading "%abc" full scan • Advanced Patterns: ORDER BY + LIMIT deterministic sort & SELECT explicit columns over * • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Parenthesized `AND`/`OR` Precedence & Prefix `LIKE` Filtering • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for `SELECT` Filtering

Purpose: To understand and apply `SELECT` Filtering — `WHERE`, `AND` vs `OR` Precedence, `BETWEEN`, `IN`, `LIKE` Wildcards & `LIMIT` in real MYSQL systems — specifically: Boolean...

What Topic Is Next → Next Topic 3.4: Three-Valued `NULL` Logic — `IS NULL`, `IS NOT NULL`, Null-Safe `<=>` Equal & `COALESCE()`

3.4 Three-Valued `NULL` Logic — `IS NULL`, `IS NOT NULL`, Null-Safe `<=>` Equal & `COALESCE()`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: TRUE / FALSE / UNKNOWN 3-valued logic & Col = NULL is always UNKNOWN • Core Syntax, CLI & Production APIs: WHERE deleted_at IS NULL, a <=> b, COALESCE(discount_price, price), NULLIF(divisor, 0) • Deep-Dive Mechanics: Col != X drops NULL rows trap & Null-safe <=> operator • Advanced Patterns: COALESCE first non-null & NULLIF(x, 0) divide-by-zero guard • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Null-Safe `<=>` Comparison, `COALESCE` Pricing & `NULLIF` Division Safety • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Three-Valued `NULL` Logic

Purpose: To understand and apply Three-Valued `NULL` Logic — `IS NULL`, `IS NOT NULL`, Null-Safe `<=>` Equal & `COALESCE()` in real MYSQL systems — specifically: Why `NULL` means...

What Topic Is Next → Module 3 Capstone (Idempotent Daily Attendance & Leaderboard Upsert Pipeline) → Next Topic 4.1: MySQL Aggregate Functions — `COUNT(*)` vs `COUNT(col)`, `COUNT(DISTINCT)`, `SUM`, `AVG` & `GROUP_CONCAT`

Module 3 Hands-On Capstone Project (100 Marks): Idempotent Daily Attendance & Leaderboard Upsert Pipeline — Write a batch DML ingestion script that bulk-upserts daily student mock scores using MySQL 8 `INSERT INTO ... VALUES ... AS new_row ON DUPLICATE KEY UPDATE best_score = GREATEST(best_score,...

MODULE 4 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Aggregations, `GROUP BY`, `ONLY_FULL_GROUP_BY`, `HAVING` & Conditional `CASE` Pivots

Build executive KPI reports using `COUNT`, `SUM`, `AVG`, `GROUP_CONCAT`, master MySQL 8 `ONLY_FULL_GROUP_BY` compliance and `WITH ROLLUP` subtotals, separate `WHERE` from `HAVING`, and pivot rows into columns with `CASE WHEN`.

Objectives: ✓ MySQL Aggregate Functions — `COUNT(*)` vs `COUNT(col)`, `COUNT(DISTINCT)`, `SUM`, `AVG` & `GROUP_CONCAT` — How SQL aggregates collapse row sets while ignoring `NULL`s (`AVG([10, 20, NULL])` divides `30 / 2 = 15`, NOT `30 / 3 = 10`!), `COALESCE(SUM(x), 0)` on empty sets, and

`GROUP_CONCAT(DISTINCT tag ORDER BY tag SEPARATOR ", ")`. • ✓ `GROUP BY` Mechanics, `ONLY_FULL_GROUP_BY` Mode, Functional Dependency & `WITH ROLLUP` — Grouping rows into summary buckets, why MySQL 8 enables `ONLY_FULL_GROUP_BY` by default (rejecting ambiguous non-aggregated `SELECT` columns), functional dependency when grouping by a `PRIMARY KEY`, and super-aggregate subtotals via `GROUP BY ... WITH ROLLUP` + `GROUPING()`. • ✓ `HAVING` vs `WHERE` — Pre-Grouping Row Filtering vs Post-Grouping Aggregate Filtering — Why `WHERE` cannot reference aggregate functions (`ERROR 1111: Invalid use of group function`), why `HAVING` filters grouped buckets AFTER `GROUP BY`, and why putting non-aggregate column filters into `HAVING` wastes CPU & skips index pruning. • ✓ Conditional SQL — `CASE WHEN ... THEN ... ELSE ... END`, `IF()`, `IFNULL()` & Conditional Pivot Aggregation — Searched `CASE WHEN` expressions (short-circuit top-to-bottom evaluation!), pivoting status rows into side-by-side summary columns in a single table scan (`SUM(CASE WHEN ...)`), and safe grade bucketing. • ✓ Build, verify, and ship Module Capstone: Single-Pass Cohort Revenue, Pass-Rate & Monthly Pivot Analytics Dashboard SQL.

4.1 MySQL Aggregate Functions — `COUNT(*)` vs `COUNT(col)`, `COUNT(DISTINCT)`, `SUM`, `AVG` & `GROUP_CONCAT` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: COUNT(*) vs COUNT(col) NULL rule & AVG(col) ignores NULL denominator • Core Syntax, CLI & Production APIs: COUNT(*), COUNT(DISTINCT user_id), COALESCE(SUM(fee), 0), AVG(score), GROUP_CONCAT(code ORDE... • Deep-Dive Mechanics: COALESCE(SUM(x), 0) empty set & COUNT(DISTINCT col) • Advanced Patterns: GROUP_CONCAT max_len limit & InnoDB MVCC COUNT(*) scan • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Comparing `AVG(score)` vs `AVG(COALESCE(score, 0))` & `GROUP_CONCAT` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for MySQL Aggregate Functions

Purpose: To understand and apply MySQL Aggregate Functions — `COUNT(*)` vs `COUNT(col)`, `COUNT(DISTINCT)`, `SUM`, `AVG` & `GROUP_CONCAT` in real MYSQL systems — specifically: How...

What Topic Is Next → Next Topic 4.2: `GROUP BY` Mechanics, `ONLY_FULL_GROUP_BY` Mode, Functional Dependency & `WITH ROLLUP`

4.2 `GROUP BY` Mechanics, `ONLY_FULL_GROUP_BY` Mode, Functional Dependency & `WITH ROLLUP` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ONLY_FULL_GROUP_BY determinism & ERROR 1055 fix patterns • Core Syntax, CLI & Production APIs: SELECT course_id, COUNT(*) FROM groups GROUP BY course_id WITH ROLLUP; GROUPING(course_id) • Deep-Dive Mechanics: PRIMARY KEY functional dependency & GROUP BY ... WITH ROLLUP • Advanced Patterns: GROUPING(col) subtotal label & Index-backed GROUP BY scan • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Category Revenue Subtotals & Grand Total Using `GROUP BY ... WITH ROLLUP` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for `GROUP BY` Mechanics

Purpose: To understand and apply `GROUP BY` Mechanics, `ONLY_FULL_GROUP_BY` Mode, Functional Dependency & `WITH ROLLUP` in real MYSQL systems — specifically: Grouping rows into su...

What Topic Is Next → Next Topic 4.3: `HAVING` vs `WHERE` — Pre-Grouping Row Filtering vs Post-Grouping Aggregate Filtering

4.3 `HAVING` vs `WHERE` — Pre-Grouping Row Filtering vs Post-Grouping Aggregate Filtering [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: WHERE before GROUP BY & HAVING after GROUP BY • Core Syntax, CLI & Production APIs: WHERE status = "completed" GROUP BY user_id HAVING COUNT(*) >= 3 AND AVG(score) >= 60 • Deep-Dive Mechanics: ERROR 1111 aggregate in WHERE & Finding duplicates HAVING COUNT(*) > 1 • Advanced Patterns: Early WHERE index pruning & Combining WHERE and HAVING • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Detecting Duplicate Emails & High-Scoring Students with `WHERE` + `HAVING` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for `HAVING` vs `WHERE`

Purpose: To understand and apply `HAVING` vs `WHERE` — Pre-Grouping Row Filtering vs Post-Grouping Aggregate Filtering in real MYSQL systems — specifically: Why `WHERE` cannot ref...

What Topic Is Next → Next Topic 4.4: Conditional SQL — `CASE WHEN ... THEN ... ELSE ... END`, `IF()`, `IFNULL()` & Conditional Pivot Aggregation

4.4 Conditional SQL — `CASE WHEN ... THEN ... ELSE ... END`, `IF()`, `IFNULL()` & Conditional Pivot Aggregation [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Searched CASE WHEN order & Single-pass pivot SUM(CASE WHEN) • Core Syntax, CLI & Production APIs: CASE WHEN score >= 85 THEN "Distinction" WHEN score >= 60 THEN "Pass" ELSE "Retry" END • Deep-Dive Mechanics: Always include ELSE 0 in SUM & ANSI CASE vs MySQL IF() • Advanced Patterns: IFNULL vs standard COALESCE & Conditional UPDATE SET CASE • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Single-Scan Status Pivot Report Using `SUM(CASE WHEN ...)` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Conditional SQL

Purpose: To understand and apply Conditional SQL — `CASE WHEN ... THEN ... ELSE ... END`, `IF()`, `IFNULL()` & Conditional Pivot Aggregation in real MYSQL systems — specifically:...

What Topic Is Next → Module 4 Capstone (Single-Pass Cohort Revenue, Pass-Rate & Monthly Pivot Analytics Dashboard SQL) → Next Topic 5.1: `INNER JOIN` Mechanics — Explicit `ON` Predicates, Driving Table Choice & MySQL 8 Hash Joins

Module 4 Hands-On Capstone Project (100 Marks): Single-Pass Cohort Revenue, Pass-Rate & Monthly Pivot Analytics Dashboard SQL — Write a set of single-scan analytical SQL queries computing per-course enrollment counts, paid vs pending revenue via `SUM(CASE WHEN ...)`, pass-rate percentages protected by `NULLIF`, `GROU...

MODULE 5 OF 16 • STAGE 2 • 14 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Relational `JOIN` Mastery (`INNER`, `LEFT`, `SELF`, `CROSS`) & `UNION` vs `UNION ALL`

Combine normalized multi-table data accurately using `INNER JOIN`, `LEFT JOIN` anti-joins, hierarchical `SELF JOIN`s, calendar `CROSS JOIN`s, and high-speed `UNION ALL` set operations.

Objectives: ✓ `INNER JOIN` Mechanics — Explicit `ON` Predicates, Driving Table Choice & MySQL 8 Hash Joins — Matching rows across normalized tables via `INNER JOIN ... ON`, why ANSI explicit `JOIN` beats legacy comma-joins (`FROM a, b`), how MySQL's optimizer picks the smallest filtered table as the driving table, and Block Nested-Loop replacement by **Hash Join** in MySQL 8.0.18+. • ✓ `LEFT JOIN` — Preserving Unmatched Rows, `ON` vs `WHERE` Placement Trap & Anti-Joins (`IS NULL`) — Preserving every row of the left table while padding unmatched right columns with `NULL`, finding missing child records via `LEFT JOIN ... WHERE r.id IS NULL` (Anti-Join), and why putting `WHERE r.status = "passed"` destroys outer-join `NULL` rows!. • ✓ `SELF JOIN` Hierarchies,

`RIGHT JOIN` Rewrites & Intentional `CROSS JOIN` Matrix Generation — Joining a table to itself using mandatory distinct aliases (`FROM employees e LEFT JOIN employees mgr ON e.manager_id = mgr.id`), why teams standardize on `LEFT JOIN` over `RIGHT JOIN` for readability, and generating date×course grids via `CROSS JOIN`. • ✓ Set Operations — `UNION ALL` (Zero-Sort Streaming) vs `UNION` (Deduplication) & MySQL 8 `INTERSECT`/`EXCEPT` — Stacking compatible result sets vertically (`same column count & compatible types`), why `UNION` incurs a costly temporary-table deduplication pass while `UNION ALL` streams rows directly, plus MySQL 8.0.31+ `INTERSECT` and `EXCEPT`. • ✓ Build, verify, and ship Module Capstone: Multi-Table Student Progress, Unattempted Gap Finder (`LEFT JOIN` Anti-Join) & Org Hierarchy SQL.

5.1 `INNER JOIN` Mechanics — Explicit `ON` Predicates, Driving Table Choice & MySQL 8 Hash Joins [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ANSI INNER JOIN ... ON & Driving table selection • Core Syntax, CLI & Production APIs: SELECT c.title, m.title FROM courses c INNER JOIN modules m ON m.course_id = c.id WHERE c.is... • Deep-Dive Mechanics: Indexed FK ref / eq_ref join & Matching column types & collations • Advanced Patterns: MySQL 8 Hash Join & Avoiding accidental Cartesian join • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Multi-Table `INNER JOIN` Across Courses & Published Modules • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for `INNER JOIN` Mechanics

Purpose: To understand and apply `INNER JOIN` Mechanics — Explicit `ON` Predicates, Driving Table Choice & MySQL 8 Hash Joins in real MYSQL systems — specifically: Matching rows a...

What Topic Is Next → Next Topic 5.2: `LEFT JOIN` — Preserving Unmatched Rows, `ON` vs `WHERE` Placement Trap & Anti-Joins (`IS NULL`)

5.2 `LEFT JOIN` — Preserving Unmatched Rows, `ON` vs `WHERE` Placement Trap & Anti-Joins (`IS NULL`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ON vs WHERE filter trap in LEFT JOIN & COUNT(right.id) vs COUNT(*) = 1 bug • Core Syntax, CLI & Production APIs: SELECT u.id FROM users u LEFT JOIN attempts a ON a.user_id = u.id AND a.is_passed = 1 WHERE... • Deep-Dive Mechanics: Anti-Join WHERE right.id IS NULL & Preserving zero-match left rows • Advanced Patterns: COALESCE(SUM(right.val), 0) & LEFT OUTER JOIN equivalence • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Zero-Attempt Student Finder Using `LEFT JOIN` Anti-Join & `COUNT(a.id)` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for `LEFT JOIN`

Purpose: To understand and apply `LEFT JOIN` — Preserving Unmatched Rows, `ON` vs `WHERE` Placement Trap & Anti-Joins (`IS NULL`) in real MYSQL systems — specifically: Preserving...

What Topic Is Next → Next Topic 5.3: `SELF JOIN` Hierarchies, `RIGHT JOIN` Rewrites & Intentional `CROSS JOIN` Matrix Generation

5.3 `SELF JOIN` Hierarchies, `RIGHT JOIN` Rewrites & Intentional `CROSS JOIN` Matrix Generation [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: SELF JOIN mandatory aliases & LEFT SELF JOIN keeps root CEO row • Core Syntax, CLI & Production APIs: SELECT e.name, COALESCE(m.name, "CEO") AS manager FROM staff e LEFT JOIN staff m ON e.manage... • Deep-Dive Mechanics: Pairwise comparison e1.id < e2.id & CROSS JOIN M*N calendar grids • Advanced Patterns: Preferring LEFT JOIN over RIGHT JOIN & Avoiding NATURAL JOIN implicit bugs • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Mentor-Student Hierarchy Using `LEFT` Self-Join • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for `SELF JOIN` Hierarchies

Purpose: To understand and apply `SELF JOIN` Hierarchies, `RIGHT JOIN` Rewrites & Intentional `CROSS JOIN` Matrix Generation in real MYSQL systems — specifically: Joining a table...

What Topic Is Next → Next Topic 5.4: Set Operations — `UNION ALL` (Zero-Sort Streaming) vs `UNION` (Deduplication) & MySQL 8 `INTERSECT`/`EXCEPT`

5.4 Set Operations — `UNION ALL` (Zero-Sort Streaming) vs `UNION` (Deduplication) & MySQL 8 `INTERSECT`/`EXCEPT` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: UNION ALL fast streaming & UNION costly distinct temp table • Core Syntax, CLI & Production APIs: (SELECT "mock" AS kind, created_at FROM mock_attempts WHERE user_id = 1) UNION ALL (SELECT "... • Deep-Dive Mechanics: Matching column count & types & Per-branch ORDER BY LIMIT + outer LIMIT • Advanced Patterns: MySQL 8.0.31 INTERSECT & EXCEPT & UNION vs JOIN (vertical vs horizontal) • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Comparing Row Counts Between `UNION` and `UNION ALL` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Set Operations

Purpose: To understand and apply Set Operations — `UNION ALL` (Zero-Sort Streaming) vs `UNION` (Deduplication) & MySQL 8 `INTERSECT`/`EXCEPT` in real MYSQL systems — specifically:...

What Topic Is Next → Module 5 Capstone (Multi-Table Student Progress, Unattempted Gap Finder (`LEFT JOIN` Anti-Join) & Org Hierarchy SQL) → Next Topic 6.1: Subqueries in MySQL — Scalar, Derived Tables, `EXISTS` Semi-Joins & The Deadly `NOT IN (NULL)` Trap

Module 5 Hands-On Capstone Project (100 Marks): Multi-Table Student Progress, Unattempted Gap Finder (`LEFT JOIN` Anti-Join) & Org Hierarchy SQL — Write normalized multi-table reporting queries joining `users`, `groups`, `modules`, `topics`, and `mock_attempts` — including a `LEFT JOIN ... WHERE a.id IS NULL` anti-join finding students...

MODULE 6 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Subqueries, `EXISTS` vs `NOT IN`, Recursive CTEs (`WITH`) & Updatable Views

Write clean modular SQL with correlated `EXISTS` semi-joins, avoid the deadly `NOT IN (NULL)` trap, traverse arbitrary-depth trees with MySQL 8 `WITH RECURSIVE` CTEs, and encapsulate security views.

Objectives: ✓ Subqueries in MySQL — Scalar, Derived Tables, `EXISTS` Semi-Joins & The Deadly `NOT IN (NULL)` Trap — Uncorrelated vs correlated subqueries, derived tables (`FROM (SELECT ...) AS dt` requiring an alias), MySQL 8's semi-join optimizer, and why `WHERE id NOT IN (SELECT parent_id FROM t)` silently returns **0 rows** whenever ANY `parent_id` is `NULL`!. • ✓ Common Table Expressions (`WITH` & `WITH RECURSIVE`) — Multi-Step Pipelines & Hierarchical Trees — Introduced in MySQL 8.0: named temporary result sets (`WITH cohort_stats AS (...) SELECT ...`), referencing a CTE multiple times cleanly, and `WITH RECURSIVE` (`Anchor UNION ALL Recursive Member`) for category trees, org charts, and missing date-series calendars. • ✓ MySQL Views (`CREATE OR REPLACE VIEW`) — `ALGORITHM=MERGE` vs `TEMPTABLE`, Security & `WITH CHECK OPTION` — Virtual stored queries hiding sensitive PII columns (`password`, `phone`)

from BI analysts, when MySQL rewrites a view inline (`ALGORITHM=MERGE` — keeps underlying indexes!) vs materializing (`ALGORITHM=TEMPTABLE` on `GROUP BY`/`DISTINCT`), and `WITH CHECK OPTION`. • ✓ Build, verify, and ship Module Capstone: Recursive Syllabus Prerequisite Tree (`WITH RECURSIVE`) & Executive Security Views.

6.1 Subqueries in MySQL — Scalar, Derived Tables, `EXISTS` Semi-Joins & The Deadly `NOT IN (NULL)` Trap [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: NOT IN (NULL) returns 0 rows bug & NOT EXISTS NULL-safe anti-join • Core Syntax, CLI & Production APIs: WHERE EXISTS (SELECT 1 FROM enrollments e WHERE e.user_id = u.id) / WHERE NOT EXISTS (...) • Deep-Dive Mechanics: EXISTS short-circuit semi-join & Derived table mandatory alias • Advanced Patterns: ERROR 1093 self-subquery update & Scalar vs row (a,b) subqueries • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Proving `NOT EXISTS` Succeeds Where `NOT IN (NULL)` Returns Zero Rows • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Subqueries in MySQL

Purpose: To understand and apply Subqueries in MySQL — Scalar, Derived Tables, `EXISTS` Semi-Joins & The Deadly `NOT IN (NULL)` Trap in real MYSQL systems — specifically: Uncorrel...

What Topic Is Next → Next Topic 6.2: Common Table Expressions (`WITH` & `WITH RECURSIVE`) — Multi-Step Pipelines & Hierarchical Trees

6.2 Common Table Expressions (`WITH` & `WITH RECURSIVE`) — Multi-Step Pipelines & Hierarchical Trees [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: MySQL 8.0+ WITH clause & WITH RECURSIVE anchor + member • Core Syntax, CLI & Production APIs: WITH RECURSIVE seq AS (SELECT 1 AS n UNION ALL SELECT n + 1 FROM seq WHERE n < 7) SELECT * F... • Deep-Dive Mechanics: Cte_max_recursion_depth guard & Date-gap series generation • Advanced Patterns: Category & org-tree traversal & CTE readability vs nested subqueries • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Generating a 5-Day Study Schedule Series with `WITH RECURSIVE` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Common Table Expressions (`WITH` & `WITH RECURSIVE`)

Purpose: To understand and apply Common Table Expressions (`WITH` & `WITH RECURSIVE`) — Multi-Step Pipelines & Hierarchical Trees in real MYSQL systems — specifically: Introduced...

What Topic Is Next → Next Topic 6.3: MySQL Views (`CREATE OR REPLACE VIEW`) — `ALGORITHM=MERGE` vs `TEMPTABLE`, Security & `WITH CHECK OPTION`

6.3 MySQL Views (`CREATE OR REPLACE VIEW`) — `ALGORITHM=MERGE` vs `TEMPTABLE`, Security & `WITH CHECK OPTION` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Virtual query (not materialized) & ALGORITHM=MERGE index pushdown • Core Syntax, CLI & Production APIs: CREATE OR REPLACE ALGORITHM=MERGE VIEW public_courses_v AS SELECT id, title, discount_price... • Deep-Dive Mechanics: ALGORITHM=TEMPTABLE materialization & Column-level PII redaction • Advanced Patterns: WITH CHECK OPTION guard & SHOW FULL TABLES WHERE Table_type = "VIEW" • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Inspecting View Metadata & Virtual Projection Behavior • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for MySQL Views (`CREATE OR REPLACE VIEW`)

Purpose: To understand and apply MySQL Views (`CREATE OR REPLACE VIEW`) — `ALGORITHM=MERGE` vs `TEMPTABLE`, Security & `WITH CHECK OPTION` in real MYSQL systems — specifically: Vi...

What Topic Is Next → Module 6 Capstone (Recursive Syllabus Prerequisite Tree (`WITH RECURSIVE`) & Executive Security Views) → Next Topic 7.1: Window Functions (`OVER`, `PARTITION BY`, `ORDER BY`) & Frame Clauses (`ROWS` vs `RANGE`)

Module 6 Hands-On Capstone Project (100 Marks): Recursive Syllabus Prerequisite Tree (`WITH RECURSIVE`) & Executive Security Views — Model an arbitrary-depth topic prerequisite graph traversed via `WITH RECURSIVE` (with cycle/depth guard `cte_max_recursion_depth`), identify uncompleted dependencies using `NOT EXISTS`, and...

MODULE 7 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Analytical Window Functions (`OVER`, `PARTITION BY`, Ranking & `LEAD`/`LAG`)

Perform advanced OLAP analytics directly in MySQL 8 without collapsing detail rows: running totals (`SUM() OVER`), Top-N per group (`ROW_NUMBER`, `DENSE_RANK`), and retention deltas (`LAG`/`LEAD`).

Objectives: ✓ Window Functions (`OVER`, `PARTITION BY`, `ORDER BY`) & Frame Clauses (`ROWS` vs `RANGE`) — How MySQL 8 evaluates window functions AFTER `WHERE`, `GROUP BY`, and `HAVING` (right inside `SELECT`/`ORDER BY`), named `WINDOW w AS (...)`, and why default `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW` peers tied `ORDER BY` values together unless you specify `ROWS BETWEEN`!. • ✓ MySQL Ranking Functions — `ROW_NUMBER()`, `RANK()`, `DENSE_RANK()`, `NTILE()` & Nth Highest Record — Exact tie semantics (`100, 100, 90`: `ROW_NUMBER` -> `1,2,3`, `RANK` -> `1,1,3`, `DENSE_RANK` -> `1,1,2`), solving classic product-company interviews ("Find 3rd Highest Distinct Salary" with `DENSE_RANK() = 3`, "Deduplicate Keep Latest Row" with `ROW_NUMBER() = 1`), and quartiles via `NTILE(4)`. • ✓ Offset Window Functions — `LAG()`, `LEAD()`, `FIRST_VALUE()` & `LAST_VALUE()` Frame Gotcha — Peeking backward (`LAG(col, offset, default)`) or forward (`LEAD(col, offset, default)`) within a partition without self-joins, computing attempt-to-attempt score improvement, and why `LAST_VALUE()` requires `ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING`!. • ✓ Build, verify, and ship Module Capstone: Cohort Leaderboard, Percentile Rank & Attempt-over-Attempt Improvement Engine.

7.1 Window Functions (`OVER`, `PARTITION BY`, `ORDER BY`) & Frame Clauses (`ROWS` vs `RANGE`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Preserves detail rows vs GROUP BY & Evaluated after WHERE/HAVING • Core Syntax, CLI & Production APIs: SUM(amount) OVER (PARTITION BY user_id ORDER BY id ROWS BETWEEN UNBOUNDED PRECEDING AND CURR... • Deep-Dive Mechanics: ROWS vs RANGE frame subtle tie rule & Named WINDOW w AS (...) • Advanced Patterns: Moving 7-day average frame & Partition + order composite index • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Row-by-Row Running Revenue Total Per Course Using `SUM() OVER` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Window Functions (`OVER`, `PARTITION BY`, `ORDER BY`...

Purpose: To understand and apply Window Functions (`OVER`, `PARTITION BY`, `ORDER BY`) & Frame Clauses (`ROWS` vs `RANGE`) in real MYSQL systems — specifically: How MySQL 8 evalua...

What Topic Is Next → Next Topic 7.2: MySQL Ranking Functions — `ROW_NUMBER()`, `RANK()`, `DENSE_RANK()`, `NTILE()` & Nth Highest Record

7.2 MySQL Ranking Functions — `ROW_NUMBER()`, `RANK()`, `DENSE_RANK()`, `NTILE()` & Nth Highest Record

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ROW_NUMBER vs RANK vs DENSE_RANK & Top-N per group via CTE • Core Syntax, CLI & Production APIs: WITH ranked AS (SELECT *, DENSE_RANK() OVER (PARTITION BY dept_id ORDER BY salary DESC) AS d... • Deep-Dive Mechanics: Nth highest distinct salary & Deduplication keep rn = 1 • Advanced Patterns: NTILE(4) quartile bucketing & PERCENT_RANK & CUME_DIST • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Finding Top-2 Tied Scores Per Track Using `DENSE_RANK()` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for MySQL Ranking Functions

Purpose: To understand and apply MySQL Ranking Functions — `ROW_NUMBER()`, `RANK()`, `DENSE_RANK()`, `NTILE()` & Nth Highest Record in real MYSQL systems — specifically: Exact tie...

What Topic Is Next → Next Topic 7.3: Offset Window Functions — `LAG()`, `LEAD()`, `FIRST_VALUE()` & `LAST_VALUE()` Frame Gotcha

7.3 Offset Window Functions — `LAG()`, `LEAD()`, `FIRST_VALUE()` & `LAST_VALUE()` Frame Gotcha

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: LAG(col, offset, default) previous row & LEAD(col, offset, default) next row • Core Syntax, CLI & Production APIs: score - LAG(score, 1, score) OVER (PARTITION BY user_id ORDER BY attempt_no) AS delta • Deep-Dive Mechanics: LAST_VALUE frame trap (UNBOUNDED FOLLOWING) & FIRST_VALUE baseline delta • Advanced Patterns: Churn & gap-between-events analysis & Zero self-join overhead • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Computing Student Attempt-over-Attempt Score Delta with `LAG()` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Offset Window Functions

Purpose: To understand and apply Offset Window Functions — `LAG()`, `LEAD()`, `FIRST_VALUE()` & `LAST_VALUE()` Frame Gotcha in real MYSQL systems — specifically: Peeking backward...

What Topic Is Next → Module 7 Capstone (Cohort Leaderboard, Percentile Rank & Attempt-over-Attempt Improvement Engine) → Next Topic 8.1: Creating B+Tree Indexes — Composite Leftmost-Prefix Rule, SARGable Queries & Covering Indexes

Module 7 Hands-On Capstone Project (100 Marks): Cohort Leaderboard, Percentile Rank & Attempt-over-Attempt Improvement Engine — Write MySQL 8 Window Function analytics computing each student's tie-safe rank per batch (`DENSE_RANK()`), Top-3 podium winners per module via CTE + `ROW_NUMBER() <= 3`, cumulative XP runnin...

MODULE 8 OF 16 • STAGE 2 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: B+Tree Indexing Internals, InnoDB Clustered Keys & `EXPLAIN` Performance Tuning

Slash slow 5-second queries down to 0.4 milliseconds by mastering InnoDB Clustered vs Secondary B+Trees, Composite Leftmost-Prefix design, SARGable predicates, Covering Indexes, and `EXPLAIN ANALYZE`.

Objectives: ✓ Creating B+Tree Indexes — Composite Leftmost-Prefix Rule, SARGable Queries & Covering Indexes — How B+Tree leaf nodes stay doubly-linked in sorted order, the **Leftmost-Prefix Rule** on `INDEX idx_status_course_date (status, course_id, created_at)`, why wrapping an indexed column in a function (`WHERE DATE(created_at) = CURDATE()`) blinds the index, and **Covering Indexes** (`Extra: Using index`). • ✓ InnoDB Clustered Index vs Secondary (Non-Clustered) Indexes — Bookmark Lookups & UUIDv4 Page Splits — In InnoDB, the table **IS** the `PRIMARY KEY` Clustered B+Tree (leaf nodes store the entire row!), while every Secondary Index stores `(secondary_key, primary_key)` — requiring a double-lookup ("bookmark lookup") unless covered, and why random `UUIDv4` `CHAR(36)` PKs cause severe page splits. • ✓ Mastering `EXPLAIN` & `EXPLAIN ANALYZE` — Access Types (`const` > `eq_ref` > `ref` > `range` > `ALL`) — Decoding `type`, `possible_keys`, `key`, `key_len`, `rows`, `filtered`, and `Extra` (`Using index`, `Using index condition` ICP, `Using temporary`, `Using filesort`), plus MySQL 8.0.18+ `EXPLAIN ANALYZE` actual millisecond timing & loop counts. • ✓ Build, verify, and ship Module Capstone: Slow-Query Forensics & Sub-Millisecond Composite Covering Index Redesign.

8.1 Creating B+Tree Indexes — Composite Leftmost-Prefix Rule, SARGable Queries & Covering Indexes

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: B+Tree O(log N) sorted leaf chain & Composite Leftmost-Prefix rule • Core Syntax, CLI & Production APIs: CREATE INDEX idx_grp_status_created ON enrollments (group_id, enrollment_status, enrolled_at... • Deep-Dive Mechanics: SARGable range vs function-on-column & Covering Index (Using index) • Advanced Patterns: Equality -> Range/Sort column order & Write amplification of over-indexing • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Creating a Composite Index & Inspecting Cardinality via `SHOW INDEX` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Creating B+Tree Indexes

Purpose: To understand and apply Creating B+Tree Indexes — Composite Leftmost-Prefix Rule, SARGable Queries & Covering Indexes in real MYSQL systems — specifically: How B+Tree lea...

What Topic Is Next → Next Topic 8.2: InnoDB Clustered Index vs Secondary (Non-Clustered) Indexes — Bookmark Lookups & UUIDv4 Page Splits

8.2 InnoDB Clustered Index vs Secondary (Non-Clustered) Indexes — Bookmark Lookups & UUIDv4 Page Splits

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Table IS the Clustered PK B+Tree & Secondary Index stores PK pointer • Core Syntax, CLI & Production APIs: id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY (or UUID_TO_BIN(uuid, 1) sequential BINARY(16)... • Deep-Dive Mechanics: Bookmark Lookup vs Covering Index & UUIDv4 page-split fragmentation • Advanced Patterns: Narrow 8-byte BIGINT PK win & Hidden 6-byte DB_ROW_ID fallback • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Comparing Compact Binary UUID Storage (`UUID_TO_BIN` 16B vs `CHAR(36)`) • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for InnoDB Clustered Index vs Secondary (Non-Clustered)...

Purpose: To understand and apply InnoDB Clustered Index vs Secondary (Non-Clustered) Indexes — Bookmark Lookups & UUIDv4 Page Splits in real MYSQL systems — specifically: In InnoD...

What Topic Is Next → Next Topic 8.3: Mastering `EXPLAIN` & `EXPLAIN ANALYZE` — Access Types (`const` > `eq_ref` > `ref` > `range` > `ALL`)

8.3 Mastering `EXPLAIN` & `EXPLAIN ANALYZE` — Access Types (`const` > `eq_ref` > `ref` > `range` > `ALL`) [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Const > eq_ref > ref > range > index > ALL & Key_len verifies composite columns used • Core Syntax, CLI & Production APIs: EXPLAIN SELECT ...; EXPLAIN FORMAT=JSON SELECT ...; EXPLAIN ANALYZE SELECT ...; • Deep-Dive Mechanics: Extra: Using index vs Using filesort & Index Condition Pushdown (Using index condition) • Advanced Patterns: EXPLAIN ANALYZE actual runtime & ANALYZE TABLE stale stats refresh • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Verifying `type: const` Primary Key Lookup via `EXPLAIN` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Mastering `EXPLAIN`

Purpose: To understand and apply Mastering `EXPLAIN` & `EXPLAIN ANALYZE` — Access Types (`const` > `eq_ref` > `ref` > `range` > `ALL`) in real MYSQL systems — specifically: Decodi...

What Topic Is Next → Module 8 Capstone (Slow-Query Forensics & Sub-Millisecond Composite Covering Index Redesign) → Next Topic 9.1: Stored Procedures & Functions — `DELIMITER`, `IN`/`OUT` Params, `SIGNAL SQLSTATE` & Prepared SQL

Module 8 Hands-On Capstone Project (100 Marks): Slow-Query Forensics & Sub-Millisecond Composite Covering Index Redesign — Take 5 deliberately unoptimized OLTP queries (non-SARGable `WHERE DATE(created_at) = ...`, wrong composite column order, `OFFSET 100000` deep pagination, `SELECT *` bookmark-lookup storm) an...

MODULE 9 OF 16 • STAGE 3 • 12 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Stored Procedures, Triggers & Indexed Native Binary `JSON` Columns

Encapsulate database routines safely with Stored Procedures (`IN`/`OUT`, `SIGNAL SQLSTATE`, Prepared Statements), automate row-level audit trails with Triggers (`NEW`/`OLD`), and index semi-structured MySQL 8 `JSON` fields.

Objectives: ✓ Stored Procedures & Functions — `DELIMITER`, `IN`/`OUT` Params, `SIGNAL SQLSTATE` & Prepared SQL — Creating routines with `CREATE PROCEDURE`, `DECLARE EXIT HANDLER FOR SQLEXCEPTION`, raising custom business errors via `SIGNAL SQLSTATE "45000"`, dynamic SQL safety with `PREPARE ... EXECUTE ... USING @param`, and `PROCEDURE` vs `FUNCTION` differences. • ✓ MySQL Triggers (`BEFORE` / `AFTER` `INSERT`, `UPDATE`, `DELETE`) — `NEW` vs `OLD` & Audit Logs — Row-level `FOR EACH ROW` triggers, normalizing/validating incoming fields in `BEFORE INSERT`/`BEFORE UPDATE` (`SET NEW.email = LOWER(TRIM(NEW.email))`), recording tamper-evident history in `AFTER UPDATE`/`AFTER DELETE` using `OLD` and `NEW`, and trigger pitfalls. • ✓ Native `JSON` in MySQL 8 — Binary DOM Storage, `->` vs `->>`, `JSON_TABLE` & Virtual Functional Indexes — How MySQL stores `JSON` in an optimized binary DOM (validating syntax on write + O(1) key lookup without reparsing text!), `col->"$.tier"` (quoted JSON) vs `col->>"$.tier"` (`JSON_UNQUOTE` plain text!), multi-valued indexes (`CAST(... AS ... ARRAY)`), and Generated Virtual Columns. • ✓ Build, verify, and ship Module Capstone: Immutable Financial Audit Trigger Trail & Indexed JSON Metadata Search Engine.

9.1 Stored Procedures & Functions — `DELIMITER`, `IN`/`OUT` Params, `SIGNAL SQLSTATE` & Prepared SQL [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: PROCEDURE (CALL + TCL) vs FUNCTION (SELECT) & IN / OUT / INOUT parameters • Core Syntax, CLI & Production APIs: CREATE PROCEDURE enroll_student(IN p_group INT, IN p_user INT, OUT p_ok TINYINT) ...; CALL e... • Deep-Dive Mechanics: SIGNAL SQLSTATE "45000" errors & DECLARE EXIT HANDLER rollback • Advanced Patterns: PREPARE ... EXECUTE ... USING & SQL SECURITY INVOKER vs DEFINER • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Server-Side Parameterized `PREPARE` / `EXECUTE` / `DEALLOCATE` Pattern • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Stored Procedures

Purpose: To understand and apply Stored Procedures & Functions — `DELIMITER`, `IN`/`OUT` Params, `SIGNAL SQLSTATE` & Prepared SQL in real MYSQL systems — specifically: Creating ro...

What Topic Is Next → Next Topic 9.2: MySQL Triggers (`BEFORE` / `AFTER` `INSERT`, `UPDATE`, `DELETE`) — `NEW` vs `OLD` & Audit Logs

9.2 MySQL Triggers (`BEFORE` / `AFTER` `INSERT`, `UPDATE`, `DELETE`) — `NEW` vs `OLD` & Audit Logs [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: BEFORE (mutate NEW) vs AFTER (audit) & OLD vs NEW pseudo-records • Core Syntax, CLI & Production APIs: CREATE TRIGGER trg_course_fee_audit AFTER UPDATE ON courses FOR EACH ROW ... • Deep-Dive Mechanics: TRUNCATE & FK CASCADE skip triggers & FOR EACH ROW bulk-write cost • Advanced Patterns: Atomic rollback on trigger error & SHOW TRIGGERS visibility • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Trigger Pseudo-Record Availability Matrix (`NEW` vs `OLD`) • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for MySQL Triggers (`BEFORE` / `AFTER` `INSERT`, `UPDATE...

Purpose: To understand and apply MySQL Triggers (`BEFORE` / `AFTER` `INSERT`, `UPDATE`, `DELETE`) — `NEW` vs `OLD` & Audit Logs in real MYSQL systems — specifically: Row-level `FO...

What Topic Is Next → Next Topic 9.3: Native `JSON` in MySQL 8 — Binary DOM Storage, `->` vs `->>`, `JSON_TABLE` & Virtual Functional Indexes

9.3 Native `JSON` in MySQL 8 — Binary DOM Storage, `->` vs `->>`, `JSON_TABLE` & Virtual Functional Indexes [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ->> unquoted vs -> JSON-quoted & Binary JSON validation & partial update • Core Syntax, CLI & Production APIs: meta->>"$.level", JSON_SET(meta, "$.featured", true), GENERATED ALWAYS AS (meta->>"$.level")... • Deep-Dive Mechanics: Generated VIRTUAL column B+Tree index & MySQL 8.0.17+ Multi-Valued Array Index • Advanced Patterns: JSON_SET vs JSON_INSERT vs JSON_REMOVE & JSON_TABLE relational projection • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Extracting Unquoted Scalar Values & Updating Keys with `JSON_SET` • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Native `JSON` in MySQL 8

Purpose: To understand and apply Native `JSON` in MySQL 8 — Binary DOM Storage, `->` vs `->>`, `JSON_TABLE` & Virtual Functional Indexes in real MYSQL systems — specifically: How...

What Topic Is Next → Module 9 Capstone (Immutable Financial Audit Trigger Trail & Indexed JSON Metadata Search Engine) → Next Topic 10.1: ACID Transactions, InnoDB MVCC (`REPEATABLE READ`), `SELECT ... FOR UPDATE` & Deadlock Prevention

Module 9 Hands-On Capstone Project (100 Marks): Immutable Financial Audit Trigger Trail & Indexed JSON Metadata Search Engine — Build an automated `AFTER UPDATE` fee-change audit trigger logging `OLD.discount_price` -> `NEW.discount_price` into `price_audit_logs`, a transactional `sp_issue_certificate` Stored Procedu...

MODULE 10 OF 16 • STAGE 3 • 16 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: ACID Transactions, InnoDB MVCC Isolation/Locking, DCL Security & Online Backups

Guarantee zero lost money or corrupted seats under high concurrency using ACID Transactions, InnoDB MVCC (`REPEATABLE READ`), `SELECT ... FOR UPDATE`, Least-Privilege `GRANT` roles, and zero-lock `mysqldump --single-transaction` backups.

Objectives: ✓ ACID Transactions, InnoDB MVCC (`REPEATABLE READ`), `SELECT ... FOR UPDATE` & Deadlock Prevention — Atomicity (undo log), Consistency, Isolation (MVCC ReadView snapshots + Next-Key Gap Locks), Durability (`innodb_flush_log_at_trx_commit = 1` redo log), `SELECT ... FOR UPDATE` (`SKIP LOCKED` job queues in MySQL 8!), and preventing AB-BA deadlocks. • ✓ Database Security (`DCL`) — `CREATE USER`, `caching_sha2_password`, Least-Privilege `GRANT`/`REVOKE` & Roles — Creating dedicated application users (`CREATE USER "app_api"@"10.0.%" IDENTIFIED BY ...`), MySQL 8 `caching_sha2_password` auth plugin, granting ONLY schema-scoped DML (`GRANT SELECT, INSERT, UPDATE, DELETE ON app_db.*`), MySQL 8 `CREATE ROLE`, and `REQUIRE SSL`. • ✓ Production Backups, Point-in-Time Recovery (PITR) & Replication — `mysqldump --single-transaction` & Binlogs — Taking zero-downtime consistent InnoDB backups via `mysqldump --single-transaction --routines --triggers --events` (uses MVCC snapshot without locking tables!), replaying Binary Logs (`mysqlbinlog`) for Point-in-Time Recovery right before an accidental `DROP`, GTID async read-replicas, and `SHOW PROCESSLIST`. • ✓ Build, verify, and ship Module Capstone: Concurrency-Safe Wallet & Seat-Booking Transaction Engine + Hot Backup Runbook.

10.1 ACID Transactions, InnoDB MVCC (`REPEATABLE READ`), `SELECT ... FOR UPDATE` & Deadlock Prevention [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: ACID Undo (Atomicity) & Redo (Durability) & Default REPEATABLE READ & MVCC • Core Syntax, CLI & Production APIs: START TRANSACTION; SELECT seats FROM groups WHERE id = 7 FOR UPDATE; UPDATE ...; COMMIT; / R... • Deep-Dive Mechanics: Next-Key Gap Locks vs Phantom Reads & SELECT ... FOR UPDATE SKIP LOCKED • Advanced Patterns: DDL causes implicit COMMIT trap & Lock rows in ascending PK order • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Atomic Wallet Transfer with `SAVEPOINT` Rollback Verification • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for ACID Transactions

Purpose: To understand and apply ACID Transactions, InnoDB MVCC (`REPEATABLE READ`), `SELECT ... FOR UPDATE` & Deadlock Prevention in real MYSQL systems — specifically: Atomicity...

What Topic Is Next → Next Topic 10.2: Database Security (`DCL`) — `CREATE USER`, `caching_sha2_password`, Least-Privilege `GRANT`/`REVOKE` & Roles

10.2 Database Security (`DCL`) — `CREATE USER`, `caching_sha2_password`, Least-Privilege `GRANT`/`REVOKE` & Roles [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: User@host identity pair & Principle of Least Privilege • Core Syntax, CLI & Production APIs: CREATE USER "maa_app"@"localhost" IDENTIFIED BY "..."; GRANT SELECT, INSERT, UPDATE, DELETE... • Deep-Dive Mechanics: Never run web apps as root & Caching_sha2_password in MySQL 8 • Advanced Patterns: MySQL 8 CREATE ROLE & SHOW GRANTS FOR CURRENT_USER() • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Inspecting Active Session User Identity & Privilege Grants • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Database Security (`DCL`)

Purpose: To understand and apply Database Security (`DCL`) — `CREATE USER`, `caching_sha2_password`, Least-Privilege `GRANT`/`REVOKE` & Roles in real MYSQL systems — specifically:...

What Topic Is Next → Next Topic 10.3: Production Backups, Point-in-Time Recovery (PITR) & Replication — `mysqldump --single-transaction` & Binlogs

10.3 Production Backups, Point-in-Time Recovery (PITR) & Replication — `mysqldump --single-transaction` & Binlogs [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Mysqldump --single-transaction MVCC & Binary Log PITR (mysqlbinlog) • Core Syntax, CLI & Production APIs: mysqldump --single-transaction --routines --triggers -u backup_usr -p app_db > backup.sql • Deep-Dive Mechanics: --routines --triggers flags & GTID primary-replica read scaling • Advanced Patterns: SHOW PROCESSLIST & slow_query_log & Regular restore drills • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Checking Binary Log Format & Slow-Query Observability Variables • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Production Backups

Purpose: To understand and apply Production Backups, Point-in-Time Recovery (PITR) & Replication — `mysqldump --single-transaction` & Binlogs in real MYSQL systems — specifically:...

What Topic Is Next → Module 10 Capstone (Concurrency-Safe Wallet & Seat-Booking Transaction Engine + Hot Backup Runbook) → Next Topic 11.1: InnoDB Clustered Primary Keys, Secondary Indexes & Covering Scans

Module 10 Hands-On Capstone Project (100 Marks): Concurrency-Safe Wallet & Seat-Booking Transaction Engine + Hot Backup Runbook — Implement a deadlock-free seat booking & wallet debit transaction acquiring `SELECT seats_left FROM batches WHERE id = ? FOR UPDATE` in deterministic ascending ID order with `SAVEPOINT` part...

MODULE 11 OF 16 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

InnoDB Storage Engine Internals, B+Tree Clustering & EXPLAIN Tuning

Production-grade MYSQL Training engineering module covering InnoDB Storage Engine Internals, B+Tree Clustering & EXPLAIN Tuning with hands-on labs and capstone build.

Objectives: ✓ InnoDB Clustered Primary Keys, Secondary Indexes & Covering Scans — Design compact BIGINT/UUIDv7 keys and composite indexes that avoid random disk seeks. • ✓ Reading EXPLAIN ANALYZE (type, key, rows, Extra: Using filesort) — Eliminate full table scans and temporary disk sorts on slow reporting endpoints. • ✓ Sargable Predicates, Prefix Indexes & Functional Indexes in MySQL 8 — Rewrite non-sargable YEAR(created_at) filters so MySQL uses B+Tree

indexes at full speed. • ✓ Build, verify, and ship Module Capstone: Sub-10ms Query Optimization Audit on a 5M-Row E-Commerce Schema.

11.1 InnoDB Clustered Primary Keys, Secondary Indexes & Covering Scans [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Secondary Indexes • Integration with Reading EXPLAIN ANALYZE (type, key, rows, Extra • Live Debugging, Telemetry & Technical Interview Trade • What Is InnoDB Clustered Primary Keys, Secondary Indexes & Covering Scans & Why Industry Teams Use It • In MYSQL Training, InnoDB Clustered Primary Keys, Secondary Indexes & Covering Scans exists for one clear real • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply InnoDB Clustered Primary Keys, Secondary Indexes & Covering Scans in real MYSQL systems — specifically: Design compact BIGINT/UUIDv7 keys and comp...

What Topic Is Next → Next Topic 11.2: Reading EXPLAIN ANALYZE (type, key, rows, Extra: Using filesort)

11.2 Reading EXPLAIN ANALYZE (type, key, rows, Extra: Using filesort) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of key • Integration with Sargable Predicates, Prefix Indexes & Functional Indexes in MySQL 8 • Live Debugging, Telemetry & Technical Interview Trade • What Is Reading EXPLAIN ANALYZE (type, key, rows, Extra • In MYSQL Training, Reading EXPLAIN ANALYZE (type, key, rows, Extra • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply Reading EXPLAIN ANALYZE (type, key, rows, Extra: Using filesort) in real MYSQL systems — specifically: Eliminate full table scans and temporary di...

What Topic Is Next → Next Topic 11.3: Sargable Predicates, Prefix Indexes & Functional Indexes in MySQL 8

11.3 Sargable Predicates, Prefix Indexes & Functional Indexes in MySQL 8 [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Prefix Indexes • Integration with InnoDB Clustered Primary Keys, Secondary Indexes & Covering Scans • Live Debugging, Telemetry & Technical Interview Trade • What Is Sargable Predicates, Prefix Indexes & Functional Indexes in MySQL 8 & Why Industry Teams Use It • In MYSQL Training, Sargable Predicates, Prefix Indexes & Functional Indexes in MySQL 8 exists for one clear real • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply Sargable Predicates, Prefix Indexes & Functional Indexes in MySQL 8 in real MYSQL systems — specifically: Rewrite non-sargable YEAR(created_at) fi...

What Topic Is Next → Module 11 Capstone (Sub-10ms Query Optimization Audit on a 5M-Row E-Commerce Schema) → Next Topic 12.1: InnoDB MVCC Undo Logs & Repeatable Read vs Read Committed

Module 11 Hands-On Capstone Project (100 Marks): Sub-10ms Query Optimization Audit on a 5M-Row E-Commerce Schema — Design, implement, test, and document 'Sub-10ms Query Optimization Audit on a 5M-Row E-Commerce Schema' applying every production concept from Module 11 (InnoDB Storage Engine Internals, B+T...

MODULE 12 OF 16 • STAGE 3 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

ACID Transactions, MVCC Isolation Levels, Gap Locks & Deadlocks

Production-grade MYSQL Training engineering module covering ACID Transactions, MVCC Isolation Levels, Gap Locks & Deadlocks with hands-on labs and capstone build.

Objectives: ✓ InnoDB MVCC Undo Logs & Repeatable Read vs Read Committed — Understand phantom prevention, consistent snapshots, and locking read behavior. • ✓ Record Locks, Gap Locks, Next-Key Locks & SELECT ... FOR UPDATE — Prevent overselling stock and diagnose InnoDB deadlock graphs (SHOW ENGINE INNODB STATUS). • ✓ Idempotent Upserts (INSERT ... ON DUPLICATE KEY UPDATE) & Batching — Ingest high-velocity telemetry and webhook syncs atomically without race errors. • ✓ Build, verify, and ship Module Capstone: Race-Free Wallet Ledger & Flash-Sale Inventory Transaction Engine.

12.1 InnoDB MVCC Undo Logs & Repeatable Read vs Read Committed [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Repeatable Read vs Read Committed • Integration with Record Locks, Gap Locks, Next • Live Debugging, Telemetry & Technical Interview Trade • What Is InnoDB MVCC Undo Logs & Repeatable Read vs Read Committed & Why Industry Teams Use It • In MYSQL Training, InnoDB MVCC Undo Logs & Repeatable Read vs Read Committed exists for one clear real • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply InnoDB MVCC Undo Logs & Repeatable Read vs Read Committed in real MYSQL systems — specifically: Understand phantom prevention, consistent snapshot...

What Topic Is Next → Next Topic 12.2: Record Locks, Gap Locks, Next-Key Locks & SELECT ... FOR UPDATE

12.2 Record Locks, Gap Locks, Next-Key Locks & SELECT ... FOR UPDATE [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Gap Locks • Integration with Idempotent Upserts (INSERT ... ON DUPLICATE KEY UPDATE) & Batching • Live Debugging, Telemetry & Technical Interview Trade • What Is Record Locks, Gap Locks, Next • In MYSQL Training, Record Locks, Gap Locks, Next • FOR UPDATE exists for one clear real • Runnable Code Implementation (SQL)

Purpose: To understand and apply Record Locks, Gap Locks, Next-Key Locks & SELECT ... FOR UPDATE in real MYSQL systems — specifically: Prevent overselling stock and diagnose InnoD...

What Topic Is Next → Next Topic 12.3: Idempotent Upserts (INSERT ... ON DUPLICATE KEY UPDATE) & Batching

12.3 Idempotent Upserts (INSERT ... ON DUPLICATE KEY UPDATE) & Batching [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Batching • Integration with InnoDB MVCC Undo Logs & Repeatable Read vs Read Committed • Live Debugging, Telemetry & Technical Interview Trade • What Is Idempotent Upserts (INSERT ... ON DUPLICATE KEY UPDATE) & Batching & Why Industry Teams Use It • In

MYSQL Training, Idempotent Upserts (INSERT ... • ON DUPLICATE KEY UPDATE) & Batching exists for one clear real • Runnable Code Implementation (SQL)

Purpose: To understand and apply Idempotent Upserts (INSERT ... ON DUPLICATE KEY UPDATE) & Batching in real MYSQL systems — specifically: Ingest high-velocity telemetry and webhoo...

What Topic Is Next → Module 12 Capstone (Race-Free Wallet Ledger & Flash-Sale Inventory Transaction Engine) → Next Topic 13.1: RANK, DENSE_RANK, LAG, LEAD & Moving-Window Aggregations

Module 12 Hands-On Capstone Project (100 Marks): Race-Free Wallet Ledger & Flash-Sale Inventory Transaction Engine — Design, implement, test, and document 'Race-Free Wallet Ledger & Flash-Sale Inventory Transaction Engine' applying every production concept from Module 12 (ACID Transactions, MVCC Isolation...

MODULE 13 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

MySQL 8 Window Functions, Recursive CTEs, JSON & Table Partitioning

Production-grade MYSQL Training engineering module covering MySQL 8 Window Functions, Recursive CTEs, JSON & Table Partitioning with hands-on labs and capstone build.

Objectives: ✓ RANK, DENSE_RANK, LAG, LEAD & Moving-Window Aggregations — Compute running balances, top-N per category, and MoM growth in pure MySQL 8 SQL. • ✓ Recursive CTE Hierarchies & Generated Indexed JSON Columns — Query org/category trees and index semi-structured JSON attributes cleanly. • ✓ RANGE / LIST Table Partitioning & Instant Partition Drop Archival — Purge expired 90-day logs in milliseconds without locking active InnoDB tables. • ✓ Build, verify, and ship Module Capstone: Executive Cohort Analytics & Range-Partitioned Audit Warehouse.

13.1 RANK, DENSE_RANK, LAG, LEAD & Moving-Window Aggregations [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of DENSE_RANK • Integration with Recursive CTE Hierarchies & Generated Indexed JSON Columns • Live Debugging, Telemetry & Technical Interview Trade • What Is RANK, DENSE_RANK, LAG, LEAD & Moving • In MYSQL Training, RANK, DENSE_RANK, LAG, LEAD & Moving • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply RANK, DENSE_RANK, LAG, LEAD & Moving-Window Aggregations in real MYSQL systems — specifically: Compute running balances, top-N per category, and M...

What Topic Is Next → Next Topic 13.2: Recursive CTE Hierarchies & Generated Indexed JSON Columns

13.2 Recursive CTE Hierarchies & Generated Indexed JSON Columns [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Generated Indexed JSON Columns • Integration with RANGE / LIST Table Partitioning & Instant Partition Drop Archival • Live Debugging, Telemetry & Technical Interview Trade • What Is Recursive CTE Hierarchies & Generated Indexed JSON Columns & Why Industry Teams Use It • In MYSQL Training, Recursive CTE Hierarchies & Generated Indexed JSON Columns exists for one clear real • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply Recursive CTE Hierarchies & Generated Indexed JSON Columns in real MYSQL systems — specifically: Query org/category trees and index semi-structure...

What Topic Is Next → Next Topic 13.3: RANGE / LIST Table Partitioning & Instant Partition Drop Archival

13.3 RANGE / LIST Table Partitioning & Instant Partition Drop Archival [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Instant Partition Drop Archival • Integration with RANK, DENSE_RANK, LAG, LEAD & Moving • Live Debugging, Telemetry & Technical Interview Trade • What Is RANGE / LIST Table Partitioning & Instant Partition Drop Archival & Why Industry Teams Use It • In MYSQL Training, RANGE / LIST Table Partitioning & Instant Partition Drop Archival exists for one clear real • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply RANGE / LIST Table Partitioning & Instant Partition Drop Archival in real MYSQL systems — specifically: Purge expired 90-day logs in milliseconds...

What Topic Is Next → Module 13 Capstone (Executive Cohort Analytics & Range-Partitioned Audit Warehouse) → Next Topic 14.1: GTID Binary Log Replication & Read/Write Workload Splitting

Module 13 Hands-On Capstone Project (100 Marks): Executive Cohort Analytics & Range-Partitioned Audit Warehouse — Design, implement, test, and document 'Executive Cohort Analytics & Range-Partitioned Audit Warehouse' applying every production concept from Module 13 (MySQL 8 Window Functions, Recursive C...

MODULE 14 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Replication, Point-in-Time Recovery (PITR), Security & Zero-Downtime DDL

Production-grade MYSQL Training engineering module covering Replication, Point-in-Time Recovery (PITR), Security & Zero-Downtime DDL with hands-on labs and capstone build.

Objectives: ✓ GTID Binary Log Replication & Read/Write Workload Splitting — Scale read-heavy traffic across replicas and promote standbys safely during maintenance. • ✓ Logical/Physical Backups + Binlog Point-in-Time Recovery (PITR) — Restore accidentally deleted rows to the exact second before human error occurred. • ✓ Least-Privilege MySQL Roles, Slow Query Log & ALGORITHM=INSTANT DDL — Alter multi-million-row production tables online without blocking live customer writes. • ✓ Build, verify, and ship Module Capstone: High-Availability Primary–Replica & PITR Disaster Recovery Setup.

14.1 GTID Binary Log Replication & Read/Write Workload Splitting [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Read/Write Workload Splitting • Integration with Logical/Physical Backups + Binlog Point • Live Debugging, Telemetry & Technical Interview Trade • What Is GTID Binary Log Replication & Read/Write Workload Splitting & Why Industry Teams Use It • In MYSQL Training, GTID Binary Log Replication & Read/Write Workload Splitting exists for one clear real • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply GTID Binary Log Replication & Read/Write Workload Splitting in real MYSQL systems — specifically: Scale read-heavy traffic across replicas and pro...

What Topic Is Next → Next Topic 14.2: Logical/Physical Backups + Binlog Point-in-Time Recovery (PITR)

14.2 Logical/Physical Backups + Binlog Point-in-Time Recovery (PITR) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with Least • Live Debugging, Telemetry & Technical Interview Trade • What Is Logical/Physical Backups + Binlog Point • In MYSQL Training, Logical/Physical Backups + Binlog Point • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply Logical/Physical Backups + Binlog Point-in-Time Recovery (PITR) in real MYSQL systems — specifically: Restore accidentally deleted rows to the exa...

What Topic Is Next → Next Topic 14.3: Least-Privilege MySQL Roles, Slow Query Log & ALGORITHM=INSTANT DDL

14.3 Least-Privilege MySQL Roles, Slow Query Log & ALGORITHM=INSTANT DDL [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Slow Query Log • Integration with GTID Binary Log Replication & Read/Write Workload Splitting • Live Debugging, Telemetry & Technical Interview Trade • What Is Least • In MYSQL Training, Least • Even if you come from a non • Runnable Code Implementation (SQL)

Purpose: To understand and apply Least-Privilege MySQL Roles, Slow Query Log & ALGORITHM=INSTANT DDL in real MYSQL systems — specifically: Alter multi-million-row production table...

What Topic Is Next → Module 14 Capstone (High-Availability Primary–Replica & PITR Disaster Recovery Setup) → Next Topic 15.1: InnoDB Row Locks, Gap Locks, Next-Key Locks & Deadlock Log Diagnosis

Module 14 Hands-On Capstone Project (100 Marks): High-Availability Primary–Replica & PITR Disaster Recovery Setup — Design, implement, test, and document 'High-Availability Primary–Replica & PITR Disaster Recovery Setup' applying every production concept from Module 14 (Replication, Point-in-Time Recovery...

MODULE 15 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

High-Concurrency InnoDB Locking, Deadlock Diagnosis & Partitioning

Self-explanatory list-and-bullet production mastery module covering High-Concurrency InnoDB Locking, Deadlock Diagnosis & Partitioning in MYSQL Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ InnoDB Row Locks, Gap Locks, Next-Key Locks & Deadlock Log Diagnosis — Purpose: Understand why concurrent transactions block or deadlock and order SQL updates to prevent them. • ✓ Zero-Downtime Online DDL Schema Migrations & Safe Column Evolution — Purpose: Add indexes and columns to multi-million-row production MySQL tables without locking live traffic. • ✓ Range/List Table Partitioning, Archival Strategies & Slow Query Log Tuning — Purpose: Keep hot operational tables compact and sub-10ms fast as historical data grows. • ✓ Build, verify, and document 'Deadlock-Free Wallet Ledger & Range-Partitioned Audit Schema in MySQL 8' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

15.1 InnoDB Row Locks, Gap Locks, Next-Key Locks & Deadlock Log Diagnosis [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for InnoDB Row Locks, Gap Locks, Next • Performance Optimization, Complexity & Latency Impact of InnoDB Row Locks, Gap Locks, Next • Live Debugging, Stack • Runnable Code Implementation (SQL) • Hands • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for InnoDB Row Locks • Production Hardening & Security: least-privilege GRANT roles, parameterized queries & row-locking order

Purpose: To understand and apply InnoDB Row Locks, Gap Locks, Next-Key Locks & Deadlock Log Diagnosis in real MYSQL systems — specifically: Understand why concurrent transactions...

What Topic Is Next → Next Topic 15.2: Zero-Downtime Online DDL Schema Migrations & Safe Column Evolution

15.2 Zero-Downtime Online DDL Schema Migrations & Safe Column Evolution [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Zero • Performance Optimization, Complexity & Latency Impact of Zero • Live Debugging, Stack • Runnable Code Implementation (SQL) • Hands • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Zero-Downtime Online DDL Schema Migrations • Production Hardening & Security: least-privilege GRANT roles, parameterized queries & row-locking order

Purpose: To understand and apply Zero-Downtime Online DDL Schema Migrations & Safe Column Evolution in real MYSQL systems — specifically: Add indexes and columns to multi-million-...

What Topic Is Next → Next Topic 15.3: Range/List Table Partitioning, Archival Strategies & Slow Query Log Tuning

15.3 Range/List Table Partitioning, Archival Strategies & Slow Query Log Tuning [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Range/List Table Partitioning, Archival Strategies & Slow Query Log Tuning • Performance Optimization, Complexity & Latency Impact of Range/List Table Partitioning, Archival Strategies & Slow.... • Live Debugging, Stack • Runnable Code Implementation (SQL) • Hands • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Range/List Table Partitioning • Production Hardening & Security: least-privilege GRANT roles, parameterized queries & row-locking order

Purpose: To understand and apply Range/List Table Partitioning, Archival Strategies & Slow Query Log Tuning in real MYSQL systems — specifically: Keep hot operational tables compa...

What Topic Is Next → Module 15 Capstone (Deadlock-Free Wallet Ledger & Range-Partitioned Audit Schema in MySQL 8) → Next Topic 16.1: GTID Replication, Read/Write Splitting & Replication Lag Safeguards

Module 15 Hands-On Capstone Project (100 Marks): Deadlock-Free Wallet Ledger & Range-Partitioned Audit Schema in MySQL 8 — Architect, implement, test, and document 'Deadlock-Free Wallet Ledger & Range-Partitioned Audit Schema in MySQL 8' applying every bullet-wise engineering concept from Module 15 (High-Concurr...

MODULE 16 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

MySQL Replication, Point-in-Time Recovery (PITR) & SQL Interview Mastery

Self-explanatory list-and-bullet production mastery module covering MySQL Replication, Point-in-Time Recovery (PITR) & SQL Interview Mastery in MYSQL Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ GTID Replication, Read/Write Splitting & Replication Lag Safeguards — Purpose: Scale read-heavy workloads across replicas while handling read-your-own-writes consistency. • ✓ Logical/Physical Backups, Binary Log Point-in-Time Recovery & Least-Privilege Grants — Purpose: Restore accidental table drops to the exact second before failure and enforce strict DB privileges. • ✓ Complex Analytical SQL (Window Functions, Recursive CTEs & Top-N Per Group) — Purpose: Write high-scoring SQL queries for data engineering and backend product-company interviews. • ✓ Build, verify, and document 'Read-Replica Routing Blueprint, Binary-Log PITR Drill & Analytical SQL Suite' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

16.1 GTID Replication, Read/Write Splitting & Replication Lag Safeguards [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for GTID Replication, Read/Write Splitting & Replication Lag Safeguards • Performance Optimization, Complexity & Latency Impact of GTID Replication, Read/Write Splitting & Replication Lag S... • Live Debugging, Stack • Runnable Code Implementation (SQL) • Hands • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for GTID Replication • Production Hardening & Security: least-privilege GRANT roles, parameterized queries & row-locking order

Purpose: To understand and apply GTID Replication, Read/Write Splitting & Replication Lag Safeguards in real MYSQL systems — specifically: Scale read-heavy workloads across replic...

What Topic Is Next → Next Topic 16.2: Logical/Physical Backups, Binary Log Point-in-Time Recovery & Least-Privilege Grants

16.2 Logical/Physical Backups, Binary Log Point-in-Time Recovery & Least-Privilege Grants [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Logical/Physical Backups, Binary Log Point • Performance Optimization, Complexity & Latency Impact of Logical/Physical Backups, Binary Log Point • Live Debugging, Stack • Runnable Code Implementation (SQL) • Hands • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Logical/Physical Backups • Production Hardening & Security: least-privilege GRANT roles, parameterized queries & row-locking order

Purpose: To understand and apply Logical/Physical Backups, Binary Log Point-in-Time Recovery & Least-Privilege Grants in real MYSQL systems — specifically: Restore accidental tabl...

What Topic Is Next → Next Topic 16.3: Complex Analytical SQL (Window Functions, Recursive CTEs & Top-N Per Group)

16.3 Complex Analytical SQL (Window Functions, Recursive CTEs & Top-N Per Group) [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Complex Analytical SQL (Window Functions, Recursive CTEs & Top • Performance Optimization, Complexity & Latency Impact of Complex Analytical SQL (Window Functions, Recursive CTEs &... • Live Debugging, Stack • Runnable Code Implementation (SQL) • Hands • Idiomatic MYSQ Implementation: 3NF relational schemas, covering composite indexes, CTEs & Window Functions • MySQL 8.0 InnoDB Buffer Pool, B+Tree Clustered Pages & MVCC Redo/Undo Logs Execution & Memory Lifecycle for Complex Analytical SQL (Window Functions, Recursive... • Production Hardening & Security: least-privilege GRANT roles, parameterized queries & row-locking order

Purpose: To understand and apply Complex Analytical SQL (Window Functions, Recursive CTEs & Top-N Per Group) in real MYSQL systems — specifically: Write high-scoring SQL queries f...

What Topic Is Next → Module 16 Capstone → End-of-Course Live Client Project: OrderVault High-Concurrency E-Commerce Relational Schema & Query Tuning Suite

Module 16 Hands-On Capstone Project (100 Marks): Read-Replica Routing Blueprint, Binary-Log PITR Drill & Analytical SQL Suite — Architect, implement, test, and document 'Read-Replica Routing Blueprint, Binary-Log PITR Drill & Analytical SQL Suite' applying every bullet-wise engineering concept from Module 16 (MySQL R...

3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: OrderVault High-Concurrency E-Commerce Relational Schema & Query Tuning Suite • **Confirmed Role:** MySQL 8 Database Design & Performance Tuning Intern

Domain & Production Infra: Multi-Vendor Retail & Inventory Analytics Client (MySQL 8.0 InnoDB + Read Replicas + Composite Indexing + Automated Backup & Slow-Log Audit)

- Normalized 3NF transactional e-commerce schemas with foreign-key constraints, ACID InnoDB transactions, and deadlock guards.

- Eliminated full-table scans using EXPLAIN ANALYZE, covering composite B-Tree indexes, and window-function analytics.
- Authored auditable stored procedures, triggers, views, and role-based database privilege policies.
- Executed zero-data-loss schema migrations and cut p95 checkout query latency by 78% in live client production.

Target Job Roles: SQL & Database Developer, Junior MySQL DBA / Data Engineer, Backend Database Specialist, BI & SQL Analytics Associate

Realistic Annual CTC Scope: **Fresher / Entry-Level (0–1 Yr):** ₹4.2 LPA – ₹7.8 LPA **Mid-Level Database / Data Engineer (1–3 Yrs):** ₹8.5 LPA – ₹16.5 LPA **Senior Data Architect / Backend Lead (3–6+ Yrs):** ₹17.0 LPA – ₹32.0+ LPA

MockAttempt Academy Pvt. Ltd. • Email: support@mockattempt.com • Phone/WhatsApp: +91 98765 43210 • Web: https://mockattempt.com/courses/mysql-training
Try 2 Free Topic-Wise Timed Topic Mock Tests & Spoken MockAttempt AI Interviews daily on every topic above before enrolling.