**Comprehensive Theory of MySQL Commands**

This document provides detailed theoretical explanations of MySQL commands ranging from basic to advanced, including database management, table creation, data manipulation, and advanced queries. Use this guide to prepare for placements and solidify your understanding of SQL.

**1. Database Management**

**Create Database**

* Command: CREATE DATABASE database\_name;
* **Purpose:** Creates a new database to store and manage related data.
* **Example:** CREATE DATABASE SchoolDB;

**Use Database**

* Command: USE database\_name;
* **Purpose:** Sets the current database for all subsequent operations.
* **Example:** USE SchoolDB;

**Show Databases**

* Command: SHOW DATABASES;
* **Purpose:** Lists all available databases.

**Drop Database**

* Command: DROP DATABASE database\_name;
* **Purpose:** Deletes an entire database and all its data permanently.
* **Example:** DROP DATABASE SchoolDB;

**2. Table Management**

**Create Table**

* Command: CREATE TABLE table\_name (column\_name datatype, ...);
* **Purpose:** Creates a table with specified columns and their data types.
* **Example:**
* CREATE TABLE Students (
* StudentID INT AUTO\_INCREMENT PRIMARY KEY,
* FirstName VARCHAR(50),
* LastName VARCHAR(50),
* Age INT
* );

**Alter Table**

* Add Column: ALTER TABLE table\_name ADD column\_name datatype;
* Modify Column: ALTER TABLE table\_name MODIFY COLUMN column\_name datatype;
* Drop Column: ALTER TABLE table\_name DROP COLUMN column\_name;
* **Purpose:** Allows modification of table structure after creation.

**Drop Table**

* Command: DROP TABLE table\_name;
* **Purpose:** Deletes the table and all its data permanently.
* **Example:** DROP TABLE Students;

**3. Data Manipulation**

**Insert Data**

* Command: INSERT INTO table\_name (column1, column2, ...) VALUES (value1, value2, ...);
* **Purpose:** Adds new rows of data to a table.
* **Example:**
* INSERT INTO Students (FirstName, LastName, Age)
* VALUES ('Alice', 'Smith', 20);

**Update Data**

* Command: UPDATE table\_name SET column\_name = value WHERE condition;
* **Purpose:** Modifies existing data in a table.
* **Example:** UPDATE Students SET Age = 21 WHERE StudentID = 1;

**Delete Data**

* Command: DELETE FROM table\_name WHERE condition;
* **Purpose:** Removes rows matching the condition.
* **Example:** DELETE FROM Students WHERE Age < 18;

**4. Data Retrieval**

**Basic Select**

* Command: SELECT column1, column2 FROM table\_name;
* **Purpose:** Retrieves specific columns of data.
* **Example:** SELECT FirstName, LastName FROM Students;

**Filtering with WHERE**

* Command: SELECT \* FROM table\_name WHERE condition;
* **Purpose:** Retrieves rows matching a condition.
* **Example:** SELECT \* FROM Students WHERE Age > 20;

**Sorting with ORDER BY**

* Command: SELECT \* FROM table\_name ORDER BY column\_name ASC|DESC;
* **Purpose:** Sorts query results in ascending or descending order.
* **Example:** SELECT \* FROM Students ORDER BY LastName ASC;

**Unique Values with DISTINCT**

* Command: SELECT DISTINCT column\_name FROM table\_name;
* **Purpose:** Retrieves unique values of a column.
* **Example:** SELECT DISTINCT Gender FROM Students;

**5. Joins**

**Inner Join**

* Command:
* SELECT columns
* FROM table1
* INNER JOIN table2 ON table1.column = table2.column;
* **Purpose:** Retrieves rows with matching data in both tables.
* **Example:**
* SELECT Students.FirstName, Courses.CourseName
* FROM Students
* INNER JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
* INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID;

**Left Join**

* Command: SELECT columns FROM table1 LEFT JOIN table2 ON condition;
* **Purpose:** Retrieves all rows from the left table and matching rows from the right table.
* **Example:**
* SELECT Students.FirstName, Courses.CourseName
* FROM Students
* LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;

**Right Join**

* Command: SELECT columns FROM table1 RIGHT JOIN table2 ON condition;
* **Purpose:** Retrieves all rows from the right table and matching rows from the left table.

**Full Outer Join (Simulated)**

* Command: Combines LEFT JOIN and RIGHT JOIN using UNION.
* Example:
* SELECT Students.FirstName, Courses.CourseName
* FROM Students
* LEFT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
* UNION
* SELECT Students.FirstName, Courses.CourseName
* FROM Students
* RIGHT JOIN Enrollments ON Students.StudentID = Enrollments.StudentID;

**6. Subqueries**

**Single-Row Subquery**

* Command: SELECT \* FROM table\_name WHERE column = (subquery);
* **Purpose:** Retrieves rows matching a single value from a subquery.
* Example:
* SELECT \* FROM Students
* WHERE Age = (SELECT MAX(Age) FROM Students);

**Multi-Row Subquery**

* Command: SELECT \* FROM table\_name WHERE column IN (subquery);
* **Purpose:** Matches rows to multiple values returned by a subquery.

**7. Aggregate Functions**

**Common Functions**

* COUNT(): Counts rows.
* AVG(): Calculates average.
* SUM(): Sums up values.
* MIN() and MAX(): Find minimum and maximum values.
* Example:
* SELECT COUNT(\*) AS TotalStudents, AVG(Age) AS AverageAge FROM Students;

**8. Grouping and Filtering**

**GROUP BY**

* Command: SELECT column, aggregate\_function(column) FROM table GROUP BY column;
* **Purpose:** Groups rows by a column and applies aggregate functions.

**HAVING Clause**

* Command: SELECT column, aggregate\_function(column) FROM table GROUP BY column HAVING condition;
* **Purpose:** Filters grouped data.
* Example:
* SELECT Gender, AVG(Age) AS AverageAge
* FROM Students
* GROUP BY Gender
* HAVING AVG(Age) > 20;

**9. Transactions**

**Commands**

* START TRANSACTION;: Begins a transaction.
* COMMIT;: Saves changes made during the transaction.
* ROLLBACK;: Reverts changes made during the transaction.
* **Example:**
* START TRANSACTION;
* UPDATE Students SET Age = Age + 1 WHERE StudentID = 2;
* ROLLBACK; -- Cancels changes

**10. Views**

**Create View**

* Command: CREATE VIEW view\_name AS SELECT query;
* **Purpose:** Creates a virtual table based on a query.
* **Example:**
* CREATE VIEW StudentCourses AS
* SELECT Students.FirstName, Students.LastName, Courses.CourseName
* FROM Students
* INNER JOIN Enrollments ON Students.StudentID = Enrollments.StudentID
* INNER JOIN Courses ON Enrollments.CourseID = Courses.CourseID;

**11. Indexes**

**Create Index**

* Command: CREATE INDEX index\_name ON table\_name(column\_name);
* **Purpose:** Improves query performance by indexing columns.
* **Example:** CREATE INDEX idx\_age ON Students(Age);

**12. Backup and Restore**

**Backup Database**

* Command (CLI):
* mysqldump -u username -p database\_name > backup\_file.sql

**Restore Database**

* Command (CLI):
* mysql -u username -p database\_name < backup\_file.sql

**13. User and Permissions**

**Create User**

* Command: CREATE USER 'username'@'hostname' IDENTIFIED BY 'password';

**Grant Permissions**

* Command: GRANT privileges ON database\_name.\* TO 'username'@'hostname';

**Revoke Permissions**

* Command: REVOKE privileges ON database\_name.\* FROM 'username'@'hostname';

This guide covers theoretical explanations for each command in MySQL, from basic to advanced operations. Practice these commands to master SQL for placements and real-world applications.