# MySQL

Certainly! Here’s a structured 1-week plan for a JavaScript developer to learn MySQL through an ebook. This plan assumes a basic understanding of programming and databases.

### Day 1: Introduction to MySQL and Setup

**Topics to Cover:**

* What is MySQL?
* MySQL vs other databases
* Installing MySQL
* Basic MySQL commands (start, stop, status)

**Reading:**

* Introduction chapters of the ebook
* Installation and setup chapters

**Practice:**

* Install MySQL on your machine
* Set up a simple database and run basic commands

### Day 2: Basic SQL Queries

**Topics to Cover:**

* SELECT, INSERT, UPDATE, DELETE statements
* Basic syntax and usage
* Filtering with WHERE clause
* Sorting results with ORDER BY

**Reading:**

* Chapters on basic SQL queries

**Practice:**

* Create a simple table
* Perform basic CRUD (Create, Read, Update, Delete) operations

### Day 3: Advanced SQL Queries

**Topics to Cover:**

* Joins (INNER, LEFT, RIGHT, FULL)
* Subqueries
* Aggregation (COUNT, SUM, AVG, MIN, MAX)
* Grouping results with GROUP BY

**Reading:**

* Chapters on advanced queries

**Practice:**

* Create related tables
* Perform JOIN operations
* Use aggregate functions

### Day 4: Database Design and Normalization

**Topics to Cover:**

* Database design principles
* Normalization (1NF, 2NF, 3NF)
* Primary keys, foreign keys, and indexes

**Reading:**

* Chapters on database design and normalization

**Practice:**

* Design a small database schema
* Normalize the schema

### Day 5: MySQL and JavaScript Integration

**Topics to Cover:**

* Connecting to MySQL from JavaScript (Node.js)
* Executing SQL queries from JavaScript
* Handling query results

**Reading:**

* Chapters on MySQL and JavaScript integration

**Practice:**

* Set up a Node.js project
* Connect to MySQL and perform queries

### Day 6: Advanced MySQL Features

**Topics to Cover:**

* Transactions
* Stored procedures
* Triggers
* Views

**Reading:**

* Chapters on advanced features

**Practice:**

* Implement transactions
* Create and use stored procedures and triggers

### Day 7: Performance Tuning and Security

**Topics to Cover:**

* Indexing for performance
* Query optimization techniques
* MySQL security best practices
* Backup and restore

**Reading:**

* Chapters on performance tuning and security

**Practice:**

* Optimize queries
* Implement security measures
* Perform a backup and restore

### Suggested Ebook

**"Learning MySQL" by Seyed M.M. and Steve Sucher**

* Covers MySQL from basics to advanced topics
* Practical examples and exercises

### Day 1: Introduction to MySQL and Setup

**Topics to Cover:**

* What is MySQL?
* MySQL vs other databases
* Installing MySQL
* Basic MySQL commands (start, stop, status)

**Reading:**

* Introduction chapters of the ebook
* Installation and setup chapters

**Explanation and Examples:**

**What is MySQL?**  
MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL). It is widely used for web applications.

**Installing MySQL:**

1. **Windows:**
   * Download MySQL Installer from the official MySQL website.
   * Run the installer and follow the setup instructions.
2. **macOS:**
   * Use Homebrew: brew install mysql
3. **Linux:**
   * Use package managers: sudo apt-get install mysql-server (Debian/Ubuntu)

**Starting and Stopping MySQL:**

* **Windows:**
* net start MySQL  
  net stop MySQL
* **macOS/Linux:**
* sudo service mysql start  
  sudo service mysql stop

**Basic MySQL Commands:**

* Open MySQL command line:
* mysql -u root -p
* Check MySQL status:
* SHOW STATUS;

### Day 2: Basic SQL Queries

**Topics to Cover:**

* SELECT, INSERT, UPDATE, DELETE statements
* Basic syntax and usage
* Filtering with WHERE clause
* Sorting results with ORDER BY

**Reading:**

* Chapters on basic SQL queries

**Explanation and Examples:**

**Creating a Table:**

CREATE TABLE users (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(100),  
 email VARCHAR(100)  
);

**Inserting Data:**

INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com');

**Selecting Data:**

SELECT \* FROM users;

**Updating Data:**

UPDATE users SET email = 'john.doe@example.com' WHERE name = 'John Doe';

**Deleting Data:**

DELETE FROM users WHERE name = 'John Doe';

**Filtering Data with WHERE:**

SELECT \* FROM users WHERE email LIKE '%example.com';

**Sorting Data with ORDER BY:**

SELECT \* FROM users ORDER BY name ASC;

### Day 3: Advanced SQL Queries

**Topics to Cover:**

* Joins (INNER, LEFT, RIGHT, FULL)
* Subqueries
* Aggregation (COUNT, SUM, AVG, MIN, MAX)
* Grouping results with GROUP BY

**Reading:**

* Chapters on advanced queries

**Explanation and Examples:**

**Joining Tables:**

SELECT users.name, orders.amount  
FROM users  
INNER JOIN orders ON users.id = orders.user\_id;

**Subqueries:**

SELECT name FROM users  
WHERE id IN (SELECT user\_id FROM orders WHERE amount > 100);

**Aggregation:**

SELECT COUNT(\*) AS user\_count FROM users;  
SELECT SUM(amount) AS total\_amount FROM orders;

**Grouping Results:**

SELECT user\_id, COUNT(\*) AS order\_count  
FROM orders  
GROUP BY user\_id;

### Day 4: Database Design and Normalization

**Topics to Cover:**

* Database design principles
* Normalization (1NF, 2NF, 3NF)
* Primary keys, foreign keys, and indexes

**Reading:**

* Chapters on database design and normalization

**Explanation and Examples:**

**Designing a Schema:**

CREATE TABLE customers (  
 customer\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 name VARCHAR(100),  
 email VARCHAR(100)  
);  
  
CREATE TABLE orders (  
 order\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 customer\_id INT,  
 amount DECIMAL(10, 2),  
 FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)  
);

**Normalization:**

* **1NF:** Ensure each column has atomic values.
* **2NF:** Ensure no partial dependency on a composite primary key.
* **3NF:** Ensure no transitive dependency.

**Creating Indexes:**

CREATE INDEX idx\_customer\_name ON customers(name);

### Day 5: MySQL and JavaScript Integration

**Topics to Cover:**

* Connecting to MySQL from JavaScript (Node.js)
* Executing SQL queries from JavaScript
* Handling query results

**Reading:**

* Chapters on MySQL and JavaScript integration

**Explanation and Examples:**

**Connecting to MySQL from Node.js:**

1. **Install MySQL package:**

* npm install mysql

1. **Connecting and Querying:**

* const mysql = require("mysql");  
  const connection = mysql.createConnection({  
   host: "localhost",  
   user: "root",  
   password: "password",  
   database: "test\_db",  
  });  
    
  connection.connect();  
    
  connection.query("SELECT \* FROM users", (error, results, fields) => {  
   if (error) throw error;  
   console.log(results);  
  });  
    
  connection.end();

### Day 6: Advanced MySQL Features

**Topics to Cover:**

* Transactions
* Stored procedures
* Triggers
* Views

**Reading:**

* Chapters on advanced features

**Explanation and Examples:**

**Transactions:**

START TRANSACTION;  
  
UPDATE accounts SET balance = balance - 100 WHERE user\_id = 1;  
UPDATE accounts SET balance = balance + 100 WHERE user\_id = 2;  
  
COMMIT;

**Stored Procedures:**

DELIMITER //  
  
CREATE PROCEDURE GetUser(IN userId INT)  
BEGIN  
 SELECT \* FROM users WHERE id = userId;  
END //  
  
DELIMITER ;

**Triggers:**

CREATE TRIGGER before\_insert\_user  
BEFORE INSERT ON users  
FOR EACH ROW  
BEGIN  
 SET NEW.created\_at = NOW();  
END;

**Views:**

CREATE VIEW user\_orders AS  
SELECT users.name, orders.amount  
FROM users  
JOIN orders ON users.id = orders.user\_id;

### Day 7: Performance Tuning and Security

**Topics to Cover:**

* Indexing for performance
* Query optimization techniques
* MySQL security best practices
* Backup and restore

**Reading:**

* Chapters on performance tuning and security

**Explanation and Examples:**

**Indexing for Performance:**

CREATE INDEX idx\_users\_email ON users(email);

**Query Optimization:**

* Use EXPLAIN to analyze query performance:
* EXPLAIN SELECT \* FROM users WHERE email = 'john@example.com';

**Security Best Practices:**

* Use strong passwords for MySQL users.
* Restrict user privileges to only what's necessary.

**Backup and Restore:**

* **Backup:**
* mysqldump -u root -p test\_db > backup.sql
* **Restore:**
* mysql -u root -p test\_db < backup.sql

By following this plan, you'll build a solid foundation in MySQL and be able to integrate it seamlessly with your JavaScript projects.

### Day 8: User Management and Security

**Topics to Cover:**

* Creating and managing user accounts
* Granting and revoking privileges
* Understanding and using roles
* Securing MySQL connections

**Reading:**

* Chapters on user management and security

**Explanation and Examples:**

**Creating and Managing User Accounts:**

CREATE USER 'new\_user'@'localhost' IDENTIFIED BY 'password';

**Granting Privileges:**

GRANT SELECT, INSERT, UPDATE ON test\_db.\* TO 'new\_user'@'localhost';

**Revoking Privileges:**

REVOKE INSERT ON test\_db.\* FROM 'new\_user'@'localhost';

**Creating and Using Roles:**

CREATE ROLE 'read\_only';  
GRANT SELECT ON test\_db.\* TO 'read\_only';  
GRANT 'read\_only' TO 'new\_user'@'localhost';

**Securing MySQL Connections:**

* Ensure MySQL is configured to use SSL for client connections.
* Update MySQL configuration (my.cnf or my.ini) to enable SSL.

### Day 9: Data Import and Export

**Topics to Cover:**

* Importing data from CSV files
* Exporting data to CSV files
* Using data import/export tools

**Reading:**

* Chapters on data import/export

**Explanation and Examples:**

**Importing Data from CSV:**

LOAD DATA INFILE '/path/to/data.csv'  
INTO TABLE users  
FIELDS TERMINATED BY ','  
ENCLOSED BY '"'  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;

**Exporting Data to CSV:**

SELECT \* FROM users  
INTO OUTFILE '/path/to/export.csv'  
FIELDS TERMINATED BY ','  
ENCLOSED BY '"'  
LINES TERMINATED BY '\n';

**Using Data Import/Export Tools:**

* Use MySQL Workbench for an easy GUI-based import/export process.

### Day 10: Handling Large Datasets

**Topics to Cover:**

* Partitioning tables
* Using LIMIT and OFFSET for pagination
* Optimizing queries for large datasets

**Reading:**

* Chapters on handling large datasets

**Explanation and Examples:**

**Partitioning Tables:**

CREATE TABLE orders (  
 order\_id INT,  
 order\_date DATE,  
 customer\_id INT,  
 amount DECIMAL(10,2)  
)  
PARTITION BY RANGE (YEAR(order\_date)) (  
 PARTITION p0 VALUES LESS THAN (2021),  
 PARTITION p1 VALUES LESS THAN (2022),  
 PARTITION p2 VALUES LESS THAN (2023)  
);

**Using LIMIT and OFFSET for Pagination:**

SELECT \* FROM users LIMIT 10 OFFSET 20;

**Optimizing Queries:**

* Ensure proper indexing.
* Avoid using SELECT \* in production queries.

### Day 11: Data Backup and Recovery

**Topics to Cover:**

* Full and incremental backups
* Restoring from backups
* Using MySQL Enterprise Backup

**Reading:**

* Chapters on data backup and recovery

**Explanation and Examples:**

**Full Backup:**

mysqldump -u root -p --all-databases > full\_backup.sql

**Incremental Backup:**

* MySQL doesn't natively support incremental backups, but you can use tools like Percona XtraBackup.

**Restoring from Backup:**

mysql -u root -p < full\_backup.sql

### Day 12: MySQL Performance Schema

**Topics to Cover:**

* Introduction to Performance Schema
* Monitoring query performance
* Identifying slow queries

**Reading:**

* Chapters on MySQL Performance Schema

**Explanation and Examples:**

**Enabling Performance Schema:**

* Ensure performance\_schema is enabled in your MySQL configuration.

**Querying Performance Data:**

SELECT \* FROM performance\_schema.events\_statements\_summary\_by\_digest  
ORDER BY SUM\_TIMER\_WAIT DESC  
LIMIT 10;

**Identifying Slow Queries:**

SHOW VARIABLES LIKE 'slow\_query\_log%';  
SET GLOBAL slow\_query\_log = 'ON';

### Day 13: Advanced Data Types and Functions

**Topics to Cover:**

* Using JSON data types
* Spatial data types
* Advanced string and date functions

**Reading:**

* Chapters on advanced data types and functions

**Explanation and Examples:**

**Using JSON Data Types:**

CREATE TABLE user\_data (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 data JSON  
);  
  
INSERT INTO user\_data (data) VALUES ('{"name": "John", "age": 30}');  
SELECT JSON\_EXTRACT(data, '$.name') FROM user\_data;

**Spatial Data Types:**

CREATE TABLE locations (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 location POINT  
);  
  
INSERT INTO locations (location) VALUES (ST\_GeomFromText('POINT(1 1)'));  
SELECT ST\_AsText(location) FROM locations;

**Advanced Functions:**

SELECT CONCAT(first\_name, ' ', last\_name) AS full\_name FROM users;  
SELECT DATE\_ADD(NOW(), INTERVAL 1 DAY) AS tomorrow;

### Day 14: Review and Project

**Topics to Cover:**

* Review all topics covered
* Build a small project integrating MySQL with JavaScript

**Reading:**

* Review chapters as needed

**Explanation and Examples:**

**Project: Build a Simple Blog**

1. **Create Database Schema:**

* CREATE TABLE posts (  
  id INT AUTO\_INCREMENT PRIMARY KEY,  
  title VARCHAR(255),  
  content TEXT,  
  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
  );  
    
  CREATE TABLE comments (  
  id INT AUTO\_INCREMENT PRIMARY KEY,  
  post\_id INT,  
  comment TEXT,  
  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
  FOREIGN KEY (post\_id) REFERENCES posts(id)  
  );

1. **Set Up Node.js Project:**

* mkdir my\_blog  
  cd my\_blog  
  npm init -y  
  npm install express mysql body-parser

1. **Create Server File (index.js):**

* const express = require("express");  
  const mysql = require("mysql");  
  const bodyParser = require("body-parser");  
    
  const app = express();  
  app.use(bodyParser.json());  
    
  const db = mysql.createConnection({  
   host: "localhost",  
   user: "root",  
   password: "password",  
   database: "blog\_db",  
  });  
    
  db.connect((err) => {  
   if (err) throw err;  
   console.log("MySQL connected");  
  });  
    
  app.get("/posts", (req, res) => {  
   db.query("SELECT \* FROM posts", (err, results) => {  
   if (err) throw err;  
   res.json(results);  
   });  
  });  
    
  app.post("/posts", (req, res) => {  
   const post = req.body;  
   db.query("INSERT INTO posts SET ?", post, (err, result) => {  
   if (err) throw err;  
   res.json({ id: result.insertId });  
   });  
  });  
    
  app.listen(3000, () => {  
   console.log("Server running on port 3000");  
  });

By following this extended plan, you will have a thorough understanding of MySQL and its integration with JavaScript. The final project will also give you practical experience in applying what you've learned.

Certainly! Let's continue with the plan for learning MySQL for a JavaScript developer, covering the remaining topics and providing examples where necessary.

### Day 15: MySQL and Node.js Authentication

**Topics to Cover:**

* Implementing user authentication with MySQL and Node.js
* Using bcrypt for password hashing
* Session management

**Reading:**

* Chapters on authentication with MySQL and Node.js

**Explanation and Examples:**

**Installing bcrypt:**

npm install bcrypt

**Hashing Passwords:**

const bcrypt = require("bcrypt");  
const saltRounds = 10;  
  
const password = "myPassword";  
  
bcrypt.hash(password, saltRounds, (err, hash) => {  
 if (err) throw err;  
 console.log("Hashed password:", hash);  
});

**Verifying Passwords:**

const hashedPassword =  
 "$2b$10$Cz3HECUn2J.yft8lP0.t.O0SZ5/Wn2O8ktA6CV5OXsy5lTdbd8lXu";  
  
bcrypt.compare(password, hashedPassword, (err, result) => {  
 if (err) throw err;  
 console.log("Password match:", result); // true or false  
});

**Session Management:**

* Use packages like express-session for managing sessions in Node.js.

### Day 16: MySQL and ORM (Object-Relational Mapping)

**Topics to Cover:**

* Introduction to ORM (e.g., Sequelize)
* Setting up Sequelize with MySQL
* Defining models and associations

**Reading:**

* Chapters on ORM with MySQL

**Explanation and Examples:**

**Installing Sequelize:**

npm install sequelize mysql2

**Setting up Sequelize:**

const { Sequelize } = require("sequelize");  
  
const sequelize = new Sequelize("database", "username", "password", {  
 host: "localhost",  
 dialect: "mysql",  
});  
  
sequelize  
 .authenticate()  
 .then(() => {  
 console.log("Connection has been established successfully.");  
 })  
 .catch((err) => {  
 console.error("Unable to connect to the database:", err);  
 });

**Defining Models:**

const { Model, DataTypes } = require("sequelize");  
  
class User extends Model {}  
User.init(  
 {  
 username: DataTypes.STRING,  
 password: DataTypes.STRING,  
 },  
 { sequelize, modelName: "user" }  
);  
  
(async () => {  
 await sequelize.sync();  
})();

### Day 17: MySQL and GraphQL

**Topics to Cover:**

* Introduction to GraphQL
* Setting up GraphQL with MySQL (e.g., Apollo Server)
* Writing GraphQL queries for MySQL

**Reading:**

* Chapters on GraphQL with MySQL

**Explanation and Examples:**

**Setting up Apollo Server with Express:**

npm install apollo-server-express graphql

**Creating a GraphQL Schema:**

const { ApolloServer, gql } = require("apollo-server-express");  
  
const typeDefs = gql`  
 type Query {  
 users: [User]  
 }  
  
 type User {  
 id: Int  
 username: String  
 email: String  
 }  
`;  
  
const resolvers = {  
 Query: {  
 users: async () => {  
 return await db.query("SELECT \* FROM users");  
 },  
 },  
};  
  
const server = new ApolloServer({ typeDefs, resolvers });  
  
server.applyMiddleware({ app });  
  
app.listen({ port: 3000 }, () =>  
 console.log(`Server ready at http://localhost:3000${server.graphqlPath}`)  
);

### Day 18: MySQL and Docker

**Topics to Cover:**

* Using Docker for MySQL development and deployment
* Docker Compose for MySQL and Node.js applications

**Reading:**

* Chapters on Docker with MySQL

**Explanation and Examples:**

**Running MySQL in Docker:**

# docker-compose.yml  
version: "3.8"  
  
services:  
 mysql:  
 image: mysql:5.7  
 ports:  
 - "3306:3306"  
 environment:  
 MYSQL\_ROOT\_PASSWORD: example  
 MYSQL\_DATABASE: my\_app\_db  
 volumes:  
 - ./data:/var/lib/mysql

**Connecting Node.js to Dockerized MySQL:**

const mysql = require("mysql");  
  
const connection = mysql.createConnection({  
 host: "localhost",  
 user: "root",  
 password: "example",  
 database: "my\_app\_db",  
});  
  
connection.connect();  
  
connection.query("SELECT \* FROM users", (error, results, fields) => {  
 if (error) throw error;  
 console.log(results);  
});  
  
connection.end();

### Day 19: MySQL and Performance Optimization

**Topics to Cover:**

* Query optimization techniques
* Indexing strategies
* Using EXPLAIN to analyze queries

**Reading:**

* Chapters on MySQL performance optimization

**Explanation and Examples:**

**Using EXPLAIN to Analyze Queries:**

EXPLAIN SELECT \* FROM users WHERE id = 1;

**Indexing Strategies:**

CREATE INDEX idx\_email ON users(email);

**Query Optimization:**

* Avoid SELECT \* if only specific columns are needed.
* Use WHERE, ORDER BY, and GROUP BY clauses efficiently.

### Day 20: Final Project and Deployment

**Topics to Cover:**

* Building a complete application using MySQL and JavaScript
* Deployment strategies (e.g., AWS, Heroku)

**Reading:**

* Review chapters as needed

**Explanation and Examples:**

**Final Project: Build a Todo List Application**

1. **Database Schema:**

* CREATE TABLE todos (  
  id INT AUTO\_INCREMENT PRIMARY KEY,  
  title VARCHAR(255) NOT NULL,  
  completed BOOLEAN DEFAULT false,  
  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
  );

1. **Node.js Backend:**
   * Use Express.js for routing.
   * Sequelize or plain MySQL queries for database operations.
2. **Frontend:**
   * Use React, Vue.js, or plain HTML/CSS/JS.
   * Axios or Fetch API for communicating with the backend.
3. **Deployment:**
   * Deploy backend (Node.js + MySQL) to AWS EC2 or Heroku.
   * Deploy frontend (if separate) to AWS S3, Netlify, or Vercel.

By following this comprehensive plan, you will gain a thorough understanding of MySQL and its integration with JavaScript, enabling you to build and deploy robust applications effectively.