60+ MySQL Interview Questions and Answers [2025 Updated]

Description

Preparing for a MySQL interview requires a solid grasp of both fundamental and advanced concepts. Key areas to focus on include understanding MySQL's architecture, proficiency in writing and optimizing SQL queries, and familiarity with various storage engines like InnoDB and MyISAM.

It's also essential to comprehend indexing strategies, transaction management, and the implementation of triggers and stored procedures. Additionally, being adept at database backup and recovery processes, as well as understanding the differences between primary and foreign keys, will demonstrate a comprehensive knowledge of MySQL.

To aid in your preparation, consider reviewing a comprehensive list of MySQL interview questions, which includes 25 basic, 23 intermediate, and 16 advanced questions.

Basic MySQL Interview Questions and Answers

Q1. What is MySQL and how does it differ from SQL?

- **MySQL:** A relational database management system (RDBMS) that uses SQL for managing databases. It is open-source and supports multiple storage engines.
- **SQL:** A standard programming language used for managing and querying relational databases. It is not specific to MySQL and works with other RDBMS like PostgreSQL, SQLite, etc.

Q2. How do you create a database in MySQL?

Use the `CREATE DATABASE` statement.

Example:

```
CREATE DATABASE mydatabase;
```

Q3. What is the default port of MySQL?

The default port for MySQL is **3306**.

Q4. What are the differences between CHAR and VARCHAR?

- **CHAR:** Fixed-length; padded with spaces to meet the defined length.
- **VARCHAR:** Variable-length; only uses the storage required for the string.

Q5. What are some common MySQL data types?

- **String types:** CHAR, VARCHAR, TEXT, BLOB
- **Numeric types:** INT, FLOAT, DOUBLE
- **Date/Time types:** DATE, TIME, DATETIME, TIMESTAMP

Q6. How do you create a table in MySQL?

Use the CREATE TABLE statement.

Example:

```
CREATE TABLE Employee (
    Employee_ID INT,
    Employee_Name VARCHAR(255),
    Salary DECIMAL(10,2)
);
```

Q7. How do you insert data into a MySQL table?

Use the INSERT INTO statement.

Example:

```
INSERT INTO Employee (Employee_ID, Employee_Name, Salary)
VALUES (1, 'John Doe', 50000.00);
```

Q8. How do you retrieve the top 10 rows from a table?

Use the LIMIT clause.

Example:

```
SELECT * FROM Employee LIMIT 10;
```

Q9. What is the difference between NOW() and CURRENT_DATE()?

- **NOW()**: Returns the current date and time.

- **CURRENT_DATE()**: Returns only the current date.

Q10. How do you delete a column in a MySQL table?

Use the ALTER TABLE statement.

Example:

```
ALTER TABLE Employee DROP COLUMN Salary;
```

Q11. What is the difference between HAVING and WHERE clauses?

- **WHERE**: Filters rows before grouping.

- **HAVING**: Filters groups after applying GROUP BY.

Q12. What are the different types of tables in MySQL?

- Heap table
- Merge table
- MyISAM table
- InnoDB table
- ISAM table

Q13. What is a BLOB in MySQL?

A BLOB (Binary Large Object) stores large binary data such as images or videos. Types include TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB.

Q14. How do you add a column to an existing table?

Use the ALTER TABLE statement.

Example:

```
ALTER TABLE Employee ADD COLUMN Department VARCHAR(255);
```

Q15. What is the use of the DISTINCT keyword?

Removes duplicate rows from the result set.

Example:

```
SELECT DISTINCT Department FROM Employee;
```

Q16. Explain the difference between `CHAR_LENGTH` and `LENGTH`.

- **CHAR_LENGTH**: Counts the number of characters.
- **LENGTH**: Counts the number of bytes.

Q17. How many indexes can be created on a MySQL table?

You can create up to **16 indexed columns** in a table.

Q18. How do you delete a MySQL table?

Use the `DROP TABLE` statement.

Example:

```
DROP TABLE Employee;
```

Q19. What are `%` and `_` in the `LIKE` statement?

- `%`: Matches zero or more characters.
- `_`: Matches exactly one character.

Example:

```
SELECT * FROM Employee WHERE Employee_Name LIKE 'J%';
```

Q20. What is the difference between `mysql_fetch_array()` and `mysql_fetch_object()`?

- **mysql_fetch_array()**: Retrieves result rows as an associative or indexed array.
- **mysql_fetch_object()**: Retrieves result rows as an object.

Q21. What is a primary key in MySQL?

A primary key is a unique identifier for table records. It ensures that no duplicate or `NULL` values exist in the column(s).

Q22. How do you add a user in MySQL?

Use the `CREATE USER` statement.

Example:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
```

Q23. What is the use of the `REGEXP` operator?

It performs pattern matching using regular expressions.

Example:

```
SELECT * FROM Employee WHERE Employee_Name REGEXP '^J';
```

Q24. What are storage engines in MySQL?

Storage engines are mechanisms for storing data. Common storage engines include:

- InnoDB (default, supports ACID compliance)
- MyISAM (faster, lacks transaction support)

Q25. How do you remove duplicate records from a MySQL table?

You can use the `DISTINCT` keyword.

Example:

```
SELECT DISTINCT * FROM Employee;
```

Or, use a `DELETE` query with a `GROUP BY` clause.

Intermediate MySQL Questions and Answers

Q26. What is a Foreign Key in MySQL?

A foreign key is a column (or group of columns) in one table that provides a link between data in two tables. It enforces referential integrity by ensuring that a value in one table matches a value in another table.

Q27. How does MySQL handle NULL values?

MySQL considers NULL as an unknown or missing value. NULL is not equal to 0, an empty string, or any value. To test for NULL, you use the `IS NULL` or `IS NOT NULL` condition.

Q28. What are MySQL indexes, and why are they used?

Indexes are data structures that improve the speed of data retrieval operations on a table at the cost of additional storage. They are used to quickly locate data without having to search every row in a database table.

Q29. How can you find duplicate records in a MySQL table?

You can use the `GROUP BY` and `HAVING` clauses to identify duplicates.

Example:

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name
HAVING COUNT(*) > 1;
```

Q30. Explain the purpose of the FOREIGN KEY constraint in MySQL.

A FOREIGN KEY is used to maintain referential integrity between two tables by linking a column in one table to the primary key of another table.

Q31. How do you create a stored procedure in MySQL?

Use the CREATE PROCEDURE statement.

Example:

```
DELIMITER //
CREATE PROCEDURE GetEmployeeDetails()
BEGIN
    SELECT * FROM Employee;
END //
DELIMITER ;
```

Q32. What is the difference between UNION and UNION ALL?

- UNION: Combines results from two queries and removes duplicates.
- UNION ALL: Combines results but keeps duplicates.

Q33. How do you create a MySQL trigger?

Use the CREATE TRIGGER statement.

Example:

```
CREATE TRIGGER after_insert_employee
AFTER INSERT ON Employee
FOR EACH ROW
INSERT INTO LogTable (LogMessage) VALUES ('Employee added');
```

Q34. What are the differences between INNER JOIN and OUTER JOIN?

- INNER JOIN: Returns only matching rows from both tables.
- OUTER JOIN: Includes matching rows plus unmatched rows (LEFT, RIGHT, FULL).

Q35. How can you optimize a MySQL query?

- Use proper indexing.
- Avoid SELECT *; specify columns.
- Use query execution plans (EXPLAIN).

- Minimize subqueries; use joins when possible.

Q36. What are MySQL Views, and how do you create one?

Views are virtual tables based on the result of a query.

Example:

```
CREATE VIEW EmployeeView AS
SELECT Employee_ID, Employee_Name FROM Employee;
```

Q37. How can you copy data from one table to another in MySQL?

Use the `INSERT INTO ... SELECT` statement.

Example:

```
INSERT INTO NewTable (column1, column2)
SELECT column1, column2 FROM OldTable;
```

Q38. What is the difference between `DELETE`, `TRUNCATE`, and `DROP`?

- `DELETE`: Removes rows and logs each deletion; can use `WHERE`.
- `TRUNCATE`: Removes all rows; faster but cannot use `WHERE`.
- `DROP`: Deletes the table structure and data.

Q39. How do you perform a case-insensitive search in MySQL?

Use the `LOWER()` or `UPPER()` functions.

Example:

```
SELECT * FROM Employee WHERE LOWER(Employee_Name) = 'john';
```

Q40. What is MySQL replication, and how does it work?

Replication involves copying data from one database server (master) to another (slave) for redundancy or load balancing.

Q41. How do you change the data type of a column in MySQL?

Use the `ALTER TABLE` statement.

Example:

```
ALTER TABLE Employee MODIFY COLUMN Salary BIGINT;
```

Q42. What is a `COMMIT` in MySQL?

A `COMMIT` saves all changes made during a transaction to the database.

Q43. How do you rollback a transaction in MySQL?

Use the `ROLLBACK` statement to undo changes made in a transaction.

Example:

```
START TRANSACTION;
-- Your queries here
ROLLBACK;
```

Q44. What are MySQL partitions, and why are they used?

Partitions divide a table into smaller, manageable pieces based on column values, improving query performance and manageability.

Q45. How can you update multiple rows in a MySQL table?

Use the `UPDATE` statement with conditions.

Example:

```
UPDATE Employee
SET Salary = Salary * 1.1
WHERE Department = 'Sales';
```

Q46. What is the purpose of the `IFNULL` function?

`IFNULL` replaces `NULL` with a specified value.

Example:

```
SELECT IFNULL(Salary, 0) FROM Employee;
```

Q47. How do you find the nth highest salary in a table?

Use the `LIMIT` and `ORDER BY` clauses.

Example:

```
SELECT DISTINCT Salary
FROM Employee
ORDER BY Salary DESC
LIMIT n-1, 1;
```

Q48. How do you monitor query performance in MySQL?

- Use the `EXPLAIN` statement to analyze queries.
- Enable the slow query log to identify inefficient queries.

Advanced MySQL Interview Questions and Answers

Q49. Explain the concept of Query Optimization in MySQL. How does the optimizer work, and what techniques can be used to analyze query performance?

Query optimization refers to the process of improving the efficiency of SQL queries, minimizing resource usage like CPU, memory, and disk I/O. MySQL's query optimizer evaluates multiple execution plans for a query and selects the one with the least cost.

Techniques for analyzing and optimizing queries include:

- **EXPLAIN**: Shows the query execution plan.
- **Indexes**: Using appropriate indexes speeds up searches.
- ****Avoiding SELECT *****: Select only the necessary columns.
- **Joins Optimization**: Use INNER JOIN over OUTER JOIN if possible.
- **Query Caching**: Cache frequent query results.
- **Normalization/Denormalization**: Balance between data redundancy and read efficiency.

Q50. What are MySQL events, and how can they be used for scheduling tasks?

MySQL events are scheduled tasks that run automatically at specified intervals or times. They are similar to cron jobs but are managed within MySQL itself. You can enable the event scheduler with the `SET GLOBAL event_scheduler = ON;`.

Example:

```
CREATE EVENT my_event
ON SCHEDULE EVERY 1 HOUR
DO
  UPDATE my_table SET status = 'inactive' WHERE status = 'active';
```

Q51. How does MySQL handle full-text search, and what are the advantages of using FULLTEXT indexes?

Full-text search in MySQL uses FULLTEXT indexes, which are optimized for searching large text columns. These indexes store a list of words, making searches faster for specific text queries.

Advantages:

- Speeds up searches for words in large text fields.
- Supports Boolean operators like +, -, and * for complex searches.
- Works well for searching documents and articles.

Q52. What are Materialized Views in MySQL? How do they differ from regular views, and what are the best practices for using them?

MySQL does not natively support Materialized Views. However, a materialized view is a database object that contains the results of a query and is refreshed periodically. The key difference from regular

views is that a materialized view stores the result data, whereas a regular view only stores the query.

Best practices:

- Use triggers or events to refresh materialized views.
- Use them for reporting and aggregating data, where real-time data isn't critical.

Q53. What is MySQL Sharding, and how does it help in scaling a database?

Sharding is the process of splitting a large database into smaller, more manageable pieces called "shards," each of which is stored on a separate database server. This approach helps in horizontal scaling by distributing data across multiple servers.

Benefits:

- Better performance due to parallel processing.
- Improved availability, as different shards can reside on different servers.

Q54. What are the types of joins available in MySQL? Explain with examples.

MySQL supports several types of joins:

- **INNER JOIN**: Returns records that have matching values in both tables.

Example:

```
SELECT * FROM employees INNER JOIN departments ON employees.department_id = de
```

- **LEFT JOIN**: Returns all records from the left table and matched records from the right table.

Example:

```
SELECT * FROM employees LEFT JOIN departments ON employees.department_id = dep
```

- **RIGHT JOIN**: Returns all records from the right table and matched records from the left table.

Example:

```
SELECT * FROM employees RIGHT JOIN departments ON employees.department_id = de
```

- **FULL JOIN**: MySQL does not support FULL OUTER JOIN natively, but it can be simulated using UNION.

- **CROSS JOIN**: Returns the Cartesian product of both tables.

Example:

```
SELECT * FROM employees CROSS JOIN departments;
```

Q55. What are Transactions in MySQL, and how do they ensure data consistency?

A transaction is a sequence of SQL operations that are executed as a single unit. It ensures the ACID (Atomicity, Consistency, Isolation, Durability) properties, guaranteeing data integrity and consistency.

Example:

```
START TRANSACTION;
UPDATE account SET balance = balance - 100 WHERE account_id = 1;
UPDATE account SET balance = balance + 100 WHERE account_id = 2;
COMMIT;
```

Q56. What is the difference between a clustered index and a non-clustered index in MySQL?

- **Clustered Index**: The table's rows are stored in the same order as the index. MySQL supports only one clustered index per table (often on the primary key).
- **Non-Clustered Index**: The index is stored separately from the table data. Multiple non-clustered indexes can be created for a table.

Q57. What is the purpose of the GROUP BY clause in SQL, and how does it work?

The GROUP BY clause groups rows that have the same values into summary rows, often used with aggregate functions like COUNT(), SUM(), AVG(), etc.

Example:

```
SELECT department, AVG(salary) FROM employees GROUP BY department;
```

Q58. What are the different types of locks in MySQL?

- **Table Locks**: Locks the entire table, preventing other transactions from reading or writing until the lock is released.
- **Row Locks**: Lock only the rows being modified, allowing other transactions to access different rows.
- **Intent Locks**: Used to indicate the type of lock a transaction intends to acquire on a row or table.

Q59. What is Normalization and list the different types of normalization?

Normalization is used to avoid duplication and redundancy. it is a process of organizing data. There are many normal forms of normalization. which are also called successive levels. The first three regular forms are sufficient.

- **First Normal Form (1NF):** There are no repeating groups within rows.
- **Second Normal form(2NF):** Value of every supporting column depending on the whole primary key.
- **Third Normal Form(3NF):** It depends only on the primary key and no other value of non-key column.

Q60. What is the difference between NOW() and CURRENT_TIMESTAMP() in MySQL?

Both `NOW()` and `CURRENT_TIMESTAMP()` return the current date and time, but:

- `NOW()` is a function, while `CURRENT_TIMESTAMP` is a keyword (they are functionally equivalent).
- `CURRENT_TIMESTAMP` can also be used in table definitions as the default value for a `DATETIME` column.

Q61. How can you prevent SQL injection attacks in MySQL?

- Use **Prepared Statements**: With bound parameters, this prevents malicious input from altering SQL logic.
- Use **Stored Procedures**: Encapsulate SQL statements in stored procedures.
- Validate and sanitize user inputs: Ensure only expected data types and formats are passed.

Q62. How do you manage user permissions in MySQL?

MySQL uses **GRANT** and **REVOKE** statements to manage user permissions. Permissions can be granted for specific databases, tables, and columns.

Q63. What is the purpose of the `EXPLAIN` keyword in MySQL?

The `EXPLAIN` keyword provides information about how MySQL executes a query, including the query execution plan. It helps in optimizing queries by showing indexes used, the order of table reads, and potential bottlenecks.

Q64. What is `AUTO_INCREMENT` in MySQL, and how does it work?

`AUTO_INCREMENT` is used to automatically generate unique integer values for a column, typically the primary key. It increments automatically with each insertion.