Here is a list of 100+ advanced MySQL interview questions and providing detailed answers

1. How can you find the Nth highest salary from a table?

Answer:

```
SELECT DISTINCT salary
FROM employee e1
WHERE N-1 = (SELECT COUNT(DISTINCT salary)
             FROM employee e2
             WHERE e2.salary > e1.salary);
```

2. Explain `JOIN` and different types of `JOIN` in MySQL.

Answer: JOIN is used to combine rows from two or more tables based on a related column between them. Types of joins:

- INNER JOIN: Returns records that have matching values in both tables.
- LEFT JOIN (or LEFT OUTER JOIN): Returns all records from the left table, and the matched records from the right table.
- RIGHT JOIN (or RIGHT OUTER JOIN): Returns all records from the right table, and the matched records from the left table.
- FULL JOIN (or FULL OUTER JOIN): Returns all records when there's a match in one of the tables.

3. How can you optimize a MySQL query?

Answer: Some of the ways include:

- Using indexes effectively.
- Avoiding `SELECT *`.
- Limiting the result set using `LIMIT`.
- Using `EXPLAIN` to understand the query execution plan.
- Avoiding heavy operations like subqueries or joins if not necessary.

4. Explain the difference between `CHAR` and `VARCHAR` data types.

Answer: CHAR has a fixed length whereas VARCHAR has a variable length. CHAR always uses the same amount of storage space per entry, while VARCHAR uses only the space required plus a small overhead.

5. Write a query to retrieve duplicate records from a table without using the `DISTINCT` keyword.

Answer:

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

6. What are the differences between `UNION` and `UNION ALL`?

Answer: UNION combines the result sets of two or more queries and removes duplicates. UNION ALL combines result sets but does not remove duplicates.

7. How can you fetch alternate records from a table?

Answer: For odd rows:

```
SELECT * FROM table_name WHERE MOD(id,2) = 1;
```

For even rows:

```
SELECT * FROM table_name WHERE MOD(id,2) = 0;
```

8. What is a `stored procedure` in MySQL?

Answer: A stored procedure is a precompiled group of SQL statements stored in the database. It can be executed multiple times whenever required.

9. How can you prevent SQL injection in MySQL?

Answer: Use prepared statements with parameterized queries, escape user inputs, and avoid using raw SQL queries with user input.

10. Write a query to find the second highest salary from a table.

Answer:

```
SELECT MAX(salary)
FROM employee
WHERE salary NOT IN (SELECT MAX(salary) FROM employee);
```

11. How do you index a column in a table?

Answer:

```
ALTER TABLE table_name ADD INDEX(index_name, column_name);
```

12. Explain the `ACID` properties in a database.

Answer: ACID stands for Atomicity, Consistency, Isolation, and Durability. It ensures that database transactions are processed reliably.

13. How can you improve the performance of a MySQL database?

Answer: Some methods include:

- Normalizing the database.
- Using appropriate indexes.
- Using the latest versions of MySQL.
- Using caching mechanisms.
- Optimizing server settings.

14. Write a query to find all employees who started after Jan 1, 2020, but before Jan 1, 2023.

Answer:

```
SELECT * FROM employees
WHERE start_date BETWEEN '2020-01-01' AND '2022-12-31';
```

15. What is a `trigger` in MySQL?

Answer: A trigger is a set of instructions that are automatically executed (or fired) in response to a specific event, such as inserting, updating, or deleting records in a table.

If you need more questions or further elaboration on any of the given questions, please let me know!

16. What is a view in MySQL?

Answer: A view is a virtual table based on the result set of an SQL statement. It contains rows and columns from one or more tables. Views do not store data physically, but rather, they provide a way to look at data in different ways without changing the underlying schema.

17. How can you implement pagination in MySQL?

Answer: Pagination can be implemented using the `LIMIT` and `OFFSET` clauses.

```
SELECT * FROM table_name
LIMIT 10 OFFSET 20;
```

This would retrieve records 21 through 30.

18. Explain the difference between `MyISAM` and `InnoDB`.

Answer: MyISAM and InnoDB are storage engines for MySQL.

- MyISAM: Table-level locking, no foreign key constraints, no transaction support.
- InnoDB: Row-level locking, supports foreign key constraints, ACID-compliant with transaction support.

19. How can you find all tables that have specific column names in a database?

Answer:

```
SELECT table_name
FROM INFORMATION_SCHEMA.COLUMNS
WHERE COLUMN_NAME = 'your_column_name'
AND TABLE_SCHEMA = 'your_database_name';
```

20. How can you backup and restore a MySQL database?

Answer: To backup:

```
mysqldump -u username -p database_name > backup.sql
```

To restore:

```
mysql -u username -p database_name < backup.sql
```

21. How do you concatenate strings in MySQL?

Answer: You can use the `CONCAT` function or the `||` operator (if the `PIPES_AS_CONCAT` SQL mode is enabled).

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM employees;
```

22. How can you retrieve unique values from a column without using the `DISTINCT` keyword?

Answer:

```
SELECT column_name
FROM table_name
GROUP BY column_name;
```

23. Explain the difference between a `PRIMARY KEY` and a `UNIQUE` constraint.

Answer: Both enforce uniqueness for the values in a column, but a table can have only one primary key, whereas it can have multiple unique constraints. Additionally, primary keys automatically create a clustered index on the column, whereas unique constraints create a non-clustered index by default.

24. How can you create a copy of a table, including both structure and data, without using any backup utilities?

Answer:

```
CREATE TABLE new_table AS SELECT * FROM old_table;
```

25. How can you convert a UNIX timestamp into a readable date format in MySQL?

Answer:

```
SELECT FROM_UNIXTIME(your_unix_timestamp_column)
FROM your_table;
```

26. What's the difference between `NOW()` and `CURRENT_DATE()` in MySQL?

Answer: `NOW()` returns the current date and time, while `CURRENT_DATE()` returns only the current date.

27. Write a query to get the length of the string in a column.

Answer:

```
SELECT LENGTH(column_name)
FROM table_name;
```

28. How do you delete all records from a table without deleting the table itself?

Answer:

```
TRUNCATE TABLE table_name;
```

29. What is the purpose of the `GROUP_CONCAT` function in MySQL?

Answer: `GROUP_CONCAT` function is used to concatenate values from multiple rows into a single string. It's especially useful when used with `GROUP BY`.

30. How do you convert a data type of a column in a table?

Answer:

```
ALTER TABLE table_name
MODIFY column_name NEW_DATA_TYPE;
```

I hope these questions help. If you need more questions or any further clarifications, let me know!

You:

provide more further questions on very complex queries

ChatGPT:

Of course! Here are some more questions based on complex SQL queries:

31. How would you retrieve the total count of rows, but only count each distinct value in a column once?

Answer:

```
SELECT COUNT(DISTINCT column_name)
FROM table_name;
```

32. How would you find the three most frequent values in a column along with their counts?

Answer:

```
SELECT column_name, COUNT(column_name)
FROM table_name
GROUP BY column_name
ORDER BY COUNT(column_name) DESC
LIMIT 3;
```

33. Write a query to get the monthly sales amount for the last 12 months.

Answer:

```
SELECT MONTH(sale_date) AS month, YEAR(sale_date) AS year, SUM(amount) AS monthly_sales
FROM sales
WHERE sale_date BETWEEN DATE_SUB(NOW(), INTERVAL 12 MONTH) AND NOW()
GROUP BY YEAR(sale_date), MONTH(sale_date)
ORDER BY YEAR(sale_date) DESC, MONTH(sale_date) DESC;
```

34. Write a query to find employees who have managers with a salary greater than $100,000.

Answer:

```
SELECT e1.*
FROM employees e1
INNER JOIN employees e2 ON e1.manager_id = e2.id
WHERE e2.salary > 100000;
```

35. How would you get the rank of students based on their scores in descending order?

Answer:

```
SELECT student_name, score,
       DENSE_RANK() OVER(ORDER BY score DESC) as rank
FROM students;
```

36. Find the employees who earn more than the average salary in their respective departments.

Answer:

```
SELECT e1.id, e1.name, e1.salary, e1.department_id
FROM employees e1
JOIN (SELECT department_id, AVG(salary) AS avg_salary
      FROM employees
      GROUP BY department_id) e2
ON e1.department_id = e2.department_id
WHERE e1.salary > e2.avg_salary;
```

37. Retrieve all pairs of students who have the same scores.

Answer:

```
SELECT a.student_name, b.student_name, a.score
FROM students a, students b
WHERE a.score = b.score
AND a.student_name != b.student_name;
```

38. Write a query to retrieve the last 7 days' records, excluding weekends.

Answer:

```
SELECT *
FROM table_name
WHERE date_column BETWEEN DATE_SUB(CURDATE(), INTERVAL 7 DAY) AND CURDATE()
AND DAYOFWEEK(date_column) NOT IN (1,7);
```

39. Find the employees who have the same job roles in different departments.

Answer:

```
SELECT a.name, a.job_role, a.department_id, b.department_id
FROM employees a, employees b
WHERE a.job_role = b.job_role
AND a.department_id != b.department_id;
```

40. Retrieve the total sales amount, but replace null values with zeros.

Answer:

```
SELECT COALESCE(SUM(sales_amount), 0)
FROM sales;
```

These questions test the applicant's ability to write complex SQL queries, understand advanced SQL functions, and combine multiple techniques into a single query. If you need more questions or further details, feel free to ask!

41. How would you retrieve the name and salary of the top 3 earning employees?

Answer:

```
SELECT name, salary
FROM (
    SELECT name, salary, DENSE_RANK() OVER (ORDER BY salary DESC) AS rnk
    FROM employees
) AS subquery
WHERE rnk <= 3;
```

42. Find employees who earn above the average salary of their department and their department's average salary is above the company's average.

Answer:

```
SELECT e.name, e.salary
FROM employees e
WHERE e.salary > (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e.department_id
)
AND (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e.department_id
) > (
    SELECT AVG(salary)
    FROM employees
);
```

43. Retrieve departments that have more employees than the average number of employees across all departments.

Answer:

```
SELECT department_id
FROM employees
GROUP BY department_id
HAVING COUNT(id) > (
    SELECT AVG(employee_count)
    FROM (
        SELECT COUNT(id) as employee_count
        FROM employees
        GROUP BY department_id
    ) AS subquery
);
```

44. Find the second highest departmental average salary.

Answer:

```
SELECT MAX(avg_salary)
FROM (
    SELECT department_id, AVG(salary) as avg_salary
    FROM employees
    GROUP BY department_id
) AS subquery
WHERE avg_salary < (
    SELECT MAX(avg_salary)
    FROM (
        SELECT department_id, AVG(salary) as avg_salary
        FROM employees
        GROUP BY department_id
    ) AS subquery2
);
```

45. Retrieve the highest earning employee from each department.

Answer:

```
SELECT e.department_id, e.name, e.salary
FROM employees e
INNER JOIN (
    SELECT department_id, MAX(salary) as max_salary
    FROM employees
    GROUP BY department_id
) AS subquery
ON e.department_id = subquery.department_id
AND e.salary = subquery.max_salary;
```

46. Which departments have the same average salary?

Answer:

```
SELECT a.department_id AS dept1, b.department_id AS dept2, a.avg_salary
FROM (
    SELECT department_id, AVG(salary) as avg_salary
    FROM employees
    GROUP BY department_id
) AS a
JOIN (
    SELECT department_id, AVG(salary) as avg_salary
    FROM employees
    GROUP BY department_id
) AS b
ON a.avg_salary = b.avg_salary
AND a.department_id < b.department_id;
```

47. Find employees whose salary is above the median salary of the company.

Answer:

```
SELECT name, salary
FROM employees
WHERE salary > (
    SELECT AVG(salary)
    FROM (
        SELECT salary
        FROM employees
        ORDER BY salary
        LIMIT 2 - (SELECT COUNT(*) FROM employees) MOD 2
        OFFSET (SELECT (COUNT(*) - 1) / 2 FROM employees)
    ) AS subquery
);
```

These questions test an individual's proficiency with nested subqueries, understanding their execution order, and the ability to write efficient SQL statements. They are also indicative of real-world problems a developer might face, where breaking down problems is essential.

48. Retrieve the department names which have employees with salaries in the top 10% of all salaries.

Answer:

```
SELECT DISTINCT d.department_name
FROM departments d
JOIN employees e ON d.department_id = e.department_id
WHERE e.salary > (
    SELECT MIN(top_salary)
    FROM (
        SELECT salary as top_salary
        FROM employees
        ORDER BY salary DESC
        LIMIT (SELECT ROUND(COUNT(*) * 0.1) FROM employees)
    ) AS inner_subquery
);
```

49. Find the average salary of the departments which have more than five employees earning above the overall average salary.

Answer:

```
SELECT department_id, AVG(salary)
FROM employees
WHERE department_id IN (
    SELECT department_id
    FROM employees
    WHERE salary > (SELECT AVG(salary) FROM employees)
    GROUP BY department_id
    HAVING COUNT(id) > 5
)
GROUP BY department_id;
```

50. Retrieve employees who have the same name as their manager.

Answer:

```
SELECT e1.name
FROM employees e1
WHERE e1.manager_id IS NOT NULL
AND e1.name = (
    SELECT e2.name
    FROM employees e2
    WHERE e2.id = e1.manager_id
);
```

51. Determine if any department's average salary is higher than the maximum salary in another department.

Answer:

```
SELECT d1.department_id
FROM employees e1
JOIN departments d1 ON e1.department_id = d1.department_id
WHERE (
    SELECT AVG(e2.salary)
    FROM employees e2
    WHERE e2.department_id = d1.department_id
) > (
    SELECT MAX(e3.salary)
    FROM employees e3
    WHERE e3.department_id != d1.department_id
)
LIMIT 1;
```

52. Find the employee who has the closest salary to the company's median salary but doesn't earn the median salary.

Answer:

```
SELECT id, name, salary
FROM employees
WHERE salary <> (
    SELECT AVG(salary)
    FROM (
        SELECT salary
        FROM employees
        ORDER BY salary
        LIMIT 2 - (SELECT COUNT(*) FROM employees) MOD 2
        OFFSET (SELECT (COUNT(*) - 1) / 2 FROM employees)
    ) AS median_subquery
)
ORDER BY ABS(salary - (
    SELECT AVG(salary)
    FROM (
        SELECT salary
        FROM employees
        ORDER BY salary
        LIMIT 2 - (SELECT COUNT(*) FROM employees) MOD 2
        OFFSET (SELECT (COUNT(*) - 1) / 2 FROM employees)
    ) AS median_subquery2
))
LIMIT 1;
```

These deeply nested subqueries showcase the power of SQL when dissecting complex requirements. They can often be found in analytical or reporting applications where data is summarized or transformed in multi-step processes.