krishnaik06 / The-Grand-Complete-Data-Science-Materials Public

Code Issues 1 Pull requests 43 Actions Projects Security Ins

main

The-Grand-Complete-Data-Science-Materials / Complete MySQL Interview Materials / Amazing MySQL Interview Preparation.md

krishnaik06 Update Amazing MySQL Interview Preparation.md 6 months ago

740 lines (547 loc) · 25.1 KB

Preview Code Blame Raw

Here are 100+ MySQL interview questions with their answers, ranging from basic to more intermediate topics:

- **How do you create a new database in MySQL?**
  - **Answer:**

```
CREATE DATABASE database_name;
```

- **How do you create a new table in MySQL?**
  - **Answer:**

```
CREATE TABLE table_name (
    column1 datatype1,
    column2 datatype2,
    ...
);
```

- **How do you insert values into a table?**
  - **Answer:**

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

- **How do you retrieve all the columns from a table?**
  - **Answer:**

```sql
SELECT * FROM table_name;
```

- **How can you retrieve specific columns from a table?**
  - **Answer:**

```sql
SELECT column1, column2
FROM table_name;
```

- **What is the use of the WHERE clause?**
  - **Answer:** To filter records based on specific conditions.
- **How would you fetch data from a table where the age is greater than 25?**
  - **Answer:**

```sql
SELECT * FROM table_name WHERE age > 25;
```

- **What are the different types of SQL JOINs?**
  - **Answer:** INNER JOIN, LEFT (or LEFT OUTER) JOIN, RIGHT (or RIGHT OUTER) JOIN, and FULL (or FULL OUTER) JOIN.
- **Write a SQL query to join two tables: `students` and `courses`, assuming each student is enrolled in a course and they share a common column `course_id`.**
  - **Answer:**

```sql
SELECT * FROM students
INNER JOIN courses
ON students.course_id = courses.course_id;
```

- **What is the difference between the `HAVING` clause and the `WHERE` clause?**
  - **Answer:** `WHERE` filters records before aggregating in `GROUP BY`, whereas `HAVING` filters after aggregation.
- **How would you list the number of students enrolled in each course, but only display courses with more than 5 students?**
  - **Answer:**

```sql
SELECT course_id, COUNT(student_id) as number_of_students
FROM enrollments
```

```sql
GROUP BY course_id
HAVING number_of_students > 5;
```

- **What is the `LIKE` operator used for?**
  - **Answer:** To search for a specified pattern in a column.
- **Write a SQL query to find all students whose names start with 'A'.**
  - **Answer:**

```sql
SELECT * FROM students WHERE name LIKE 'A%';
```

- **How would you update a record in a table?**
  - **Answer:**

```sql
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE some_column = some_value;
```

- **How can you delete records from a table?**
  - **Answer:**

```sql
DELETE FROM table_name WHERE condition;
```

- **How do you drop a table?**
  - **Answer:**

```sql
DROP TABLE table_name;
```

- **What is the purpose of the `ALTER` table command?**
  - **Answer:** To modify an existing table structure, such as adding, deleting, or modifying columns.
- **How would you add a new column `email` to the `students` table?**
  - **Answer:**

```sql
ALTER TABLE students ADD COLUMN email VARCHAR(255);
```

- **What does the `DISTINCT` keyword do in a SQL query?**
  - **Answer:** It removes duplicate rows from the result set.

- **Write a query to find the total number of distinct courses from the** `enrollments` **table.**
  - **Answer:**

```
SELECT COUNT(DISTINCT course_id) FROM enrollments;
```

- **What does the** `EXISTS` **operator do?**
  - **Answer:** It tests for the existence of any record in a subquery.
- **Write a SQL query to find students who have enrolled in a course.**
  - **Answer:**

```
SELECT student_id
FROM students
WHERE EXISTS (SELECT 1 FROM enrollments WHERE students.student_id = enroll
```

- **How can you concatenate columns in MySQL?**
  - **Answer:** Using the `CONCAT()` function.
- **Write a query to get the full name of a student, given** `first_name` **and** `last_name` **columns.**
  - **Answer:**

```
SELECT CONCAT(first_name, ' ', last_name) as full_name FROM students;
```

- **How do you find the total number of rows in a table?**
  - **Answer:**

```
SELECT COUNT(*) FROM table_name;
```

- **How can you fetch the first 5 records from a table?**
  - **Answer:**

```
SELECT * FROM table_name LIMIT 5;
```

- **What is the difference between** `CHAR` **and** `VARCHAR` **data types?**
  - **Answer:** `CHAR` is fixed-length while `VARCHAR` is variable-length.
- **How can you change the data type of a column?**
  - **Answer:**

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

- **Write a SQL query to find the 3rd highest salary from a `salaries` table.**
  - **Answer:**

```
SELECT DISTINCT salary
FROM salaries
ORDER BY salary DESC
LIMIT 1 OFFSET 2;
```

- **How do you create a primary key in a table?**
  - **Answer:**

```
ALTER TABLE table_name ADD PRIMARY KEY (column_name);
```

- **What is a foreign key constraint, and why is it used?**
  - **Answer:** A foreign key constraint establishes a link between two tables and ensures that records in one table correspond to records in another. It's used to maintain referential integrity in the database.
- **How can you add a foreign key constraint to an existing table?**
  - **Answer:**

```
ALTER TABLE table_name ADD FOREIGN KEY (column_name) REFERENCES other_tabl
```

- **How can you retrieve the unique values from a column?**
  - **Answer:**

```
SELECT DISTINCT column_name FROM table_name;
```

- **What is the difference between an `INNER JOIN` and a `LEFT JOIN`?**
  - **Answer:** An `INNER JOIN` returns rows when there is a match in both tables, while a `LEFT JOIN` returns all rows from the left table and the matched rows from the right table. If there's no match, the result is `NULL` on the right side.
- **What is normalization, and why is it important?**
  - **Answer:** Normalization is the process of organizing a database to reduce redundancy and ensure data integrity. It divides larger tables into smaller ones and establishes relationships between them using foreign keys.
- **Describe 1NF, 2NF, and 3NF in database normalization.**

- **Answer:**
    - **1NF (First Normal Form):** Each table has a primary key, and all attributes are atomic (no repeating groups or arrays).
    - **2NF (Second Normal Form):** All non-key attributes are fully functionally dependent on the primary key.
    - **3NF (Third Normal Form):** All attributes are functionally dependent only on the primary key.
- **What is a subquery, and how is it different from a JOIN?**
- **Answer:** A subquery is a query nested inside another query. A subquery can return data that will be used in the main query as a condition. A JOIN is used to combine rows from two or more tables based on a related column.

- **Write a query to find employees whose salary is above the average salary.**
    - **Answer:**

```sql
SELECT employee_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

- **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 invoked as needed.
- **How can you handle errors in stored procedures?**
    - **Answer:** In MySQL, you can use the `DECLARE` statement to define error handlers using `CONTINUE` or `EXIT` handlers.
- **How do you prevent SQL injection in your queries?**
    - **Answer:** Use parameterized queries or prepared statements, avoid constructing queries with string concatenation using external input, and always validate and sanitize user input.
- **What are `TRIGGERS` in MySQL?**
    - **Answer:** Triggers are automatic actions that the database can perform when a specified change occurs (like an `INSERT` , `UPDATE` , or `DELETE` operation).
- **Can you explain the difference between `CHAR_LENGTH` and `LENGTH` functions?**
    - **Answer:** `CHAR_LENGTH` returns the number of characters in a string, while `LENGTH` returns the number of bytes. For single-byte character sets, they return the same value.
- **What is the purpose of the `GROUP_CONCAT` function in MySQL?**
    - **Answer:** `GROUP_CONCAT` returns a concatenated string of aggregated data values for each group of rows in the result set.

- **Write a SQL query to concatenate all names from the `employees` table into a single string, separated by commas.**
    - **Answer:**

```
SELECT GROUP_CONCAT(employee_name) FROM employees;
```

- **How can you create an index in MySQL?**
    - **Answer:**

```
CREATE INDEX index_name ON table_name(column_name);
```

- **What is the difference between a clustered and a non-clustered index?**
    - **Answer:** A clustered index determines the physical order of data in a table. A table can have only one clustered index. Non-clustered indexes, on the other hand, do not determine the physical order and a table can have multiple non-clustered indexes.
- **What are views in MySQL, and why are they used?**
    - **Answer:** A view is a virtual table based on the result-set of an SQL statement. They allow encapsulating complex queries, providing a simplified representation or hiding certain data.
- **What are transactions in MySQL?**
    - **Answer:** Transactions are a sequence of one or more SQL operations executed as a single unit. They ensure data integrity, following the ACID properties (Atomicity, Consistency, Isolation, Durability).
- **How do you start and commit a transaction in MySQL?**
    - **Answer:**

```
START TRANSACTION;
-- SQL operations
COMMIT;
```

- **What is the difference between `UNION` and `UNION ALL`?**
    - **Answer:** `UNION` returns unique records from the combined dataset, while `UNION ALL` returns all records, including duplicates.
- **What are the advantages of using stored procedures?**
    - **Answer:** They provide better performance as they are precompiled, help in modular programming, offer a security mechanism, and reduce network traffic.
- **What is the difference between `DATEDIFF` and `TIMESTAMPDIFF` in MySQL?**
    - **Answer:** Both are used to find the difference between two dates, but `TIMESTAMPDIFF` allows for a more specific interval, like month or year, while

`DATEDIFF` returns the difference in days.

- **How do you clone a table in MySQL?**
  - **Answer:**

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

- **Write a SQL query to rank employees based on their salary in descending order.**
  - **Answer:**

```
SELECT employee_name, salary, RANK() OVER(ORDER BY salary DESC) AS rankinᵹ
FROM employees;
```

- **How do you remove duplicate rows in a table?**
  - **Answer:** One common way is to create a new table with the distinct rows and delete the original table:

```
CREATE TABLE new_table AS SELECT DISTINCT * FROM original_table;
DROP TABLE original_table;
RENAME TABLE new_table TO original_table;
```

- **What are the default storage engines in MySQL?**
  - **Answer:** The default storage engine was MyISAM up to MySQL 5.5, but InnoDB became the default from MySQL 5.5 onward.
- **What is a self-join, and why would you use it?**
  - **Answer:** A self-join is a join of a table to
- **What is the purpose of the `SET` data type in MySQL?**
  - **Answer:** The `SET` type is used to store a set of strings. You can store zero or more string values chosen from a list defined at table creation time.

```
CREATE TABLE t1 (colors SET('red', 'blue', 'green'));
INSERT INTO t1 (colors) VALUES ('red,blue');
```

- **How do you implement pagination in MySQL?**
  - **Answer:** Using `LIMIT` and `OFFSET`.

```
SELECT * FROM table_name LIMIT 10 OFFSET 20;  -- Skips the first 20 recorc
```

- **How can you retrieve the month part from a `DATE` field in MySQL?**
    - **Answer:** Using the `MONTH()` function.

```
SELECT MONTH(date_column) FROM table_name;
```

- **How do you convert a `DATETIME` field into a Unix timestamp?**
    - **Answer:** Using the `UNIX_TIMESTAMP()` function.

```
SELECT UNIX_TIMESTAMP(datetime_column) FROM table_name;
```

- **How can you perform a case-sensitive search in a column?**
    - **Answer:** Using the `BINARY` keyword.

```
SELECT * FROM table_name WHERE BINARY column_name = 'Value';
```

- **How can you transpose rows into columns, and vice versa, in a query result?**
    - **Answer:** This process is known as "Pivoting". To convert rows to columns, you use a combination of aggregate functions with `CASE` statements. For the reverse, known as "Unpivoting", you can use `UNION ALL`.

```
-- Pivoting:
SELECT
    SUM(CASE WHEN column = 'value1' THEN 1 ELSE 0 END) AS 'Value1',
    SUM(CASE WHEN column = 'value2' THEN 1 ELSE 0 END) AS 'Value2'
FROM table_name;

-- Unpivoting:
SELECT 'Value1' AS 'Column', Value1 AS 'Value' FROM table_name
UNION ALL
SELECT 'Value2' AS 'Column', Value2 AS 'Value' FROM table_name;
```

- **How can you get a list of all indexes in a database?**
    - **Answer:**

```
SHOW INDEXES FROM table_name IN database_name;
```

- **How can you optimize a MySQL query?**

    - **Answer:** Some methods include using `EXPLAIN` to analyze the query plan, indexing appropriate columns, avoiding the use of wildcard characters at the start of a `LIKE` query, and avoiding the use of `SELECT *`.
- **What is the difference between `MyISAM` and `InnoDB`?**
    - **Answer:** Major differences include:
        - `InnoDB` supports ACID-compliant transactions, whereas `MyISAM` does not.
        - `InnoDB` supports foreign key constraints, while `MyISAM` does not.
        - `MyISAM` typically offers better read performance, while `InnoDB` offers better write performance.
    - **How can you lock a table explicitly?**
    - **Answer:**

```
LOCK TABLES table_name READ|WRITE; --Lock for reading/writing
UNLOCK TABLES; --To release the lock
```

- **How do you get the second highest value from a table column?**
    - **Answer:**

```
SELECT MAX(column_name) FROM table_name WHERE column_name < (SELECT MAX(cc
```

- **What is a correlated subquery?**
    - **Answer:** A correlated subquery is a subquery that references columns from the outer query. It's executed once for each row processed by the outer query.

```
SELECT column_name
FROM table_name t1
WHERE some_value = (SELECT MAX(column_name) FROM table_name t2 WHERE t1.ic
```

- **How can you increase the performance of a MySQL database?**
    - **Answer:** Optimize queries using `EXPLAIN`, use indexes wisely, normalize database schema, consider hardware upgrades, and configure database parameters appropriately in `my.cnf` or `my.ini`.
- **How do you backup and restore a MySQL database?**
    - **Answer:**

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

To restore:

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

- **What are the different types of MySQL collations?**
  - **Answer:** Collations specify the rules for string comparison. There are various types like `utf8_general_ci`, `utf8mb4_unicode_ci`, and `latin1_general_ci`.
- **How do you find the total number of rows affected by a query?**
  - **Answer:**

```
SELECT ROW_COUNT();
```

- **Explain the difference between `CHAR` and `VARCHAR` data types.**
  - **Answer:** `CHAR` has a fixed length, while `VARCHAR` has a variable length. For `CHAR`, unused spaces are filled with blank spaces, whereas `VARCHAR` only uses the required storage plus one or two extra bytes for the length.
- **How can you change the data type of a column in MySQL?**
  - **Answer:**

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

- **How can you measure the size of a MySQL database?**
  - **Answer:**

```
SELECT table_schema AS "Database", ROUND(SUM(data_length + index_length) /
FROM information_schema.TABLES
GROUP BY table_schema;
```

- **How can you delete all records from a table without deleting the table?**
  - **Answer:**

```
TRUNCATE TABLE table_name;
```

- **How can you prevent a query from displaying duplicate rows?**
  - **Answer:**

```
SELECT DISTINCT column_name FROM table_name;
```

- **How do you combine results from multiple SQL queries and return a single table?**

    - **Answer:** You can use the `UNION` or `UNION ALL` operator, depending on whether or not you want duplicate records.
- **How can you convert a string to upper-case in MySQL?**
    - **Answer:**

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

- **How can you remove leading and trailing whitespace from a string in MySQL?**
    - **Answer:**

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

- **Explain the purpose of `information_schema` in MySQL.**
    - **Answer:** `information_schema` is a meta-database that provides detailed information about all other databases, tables, columns, indexes, constraints, etc. present in the MySQL server.
- **How can you ensure that a field value is unique across the table, other than using the `PRIMARY KEY` constraint?**
    - **Answer:** Use the `UNIQUE` constraint on the desired column.

```
ALTER TABLE table_name ADD UNIQUE (column_name);
```

- **How can you count the total number of tables in a database?**
    - **Answer:**

```
SELECT COUNT(*) FROM information_schema.tables WHERE table_schema = 'your_
```

- **How can you find all the tables that have a specific column name in a database?**
    - **Answer:**

```
SELECT table_name
FROM information_schema.columns
WHERE column_name = 'desired_column' AND table_schema = 'your_database_nan
```

- **How can you replace a specific string in a field?**
    - **Answer:**

```
UPDATE table_name SET column_name = REPLACE(column_name, 'old_string', 'ne
```

- **What is the difference between `NOW()` and `CURDATE()` functions in MySQL?**
  - **Answer:** `NOW()` returns the current date and time, while `CURDATE()` returns only the current date.

These questions cover a range of advanced topics and should help in assessing the depth of knowledge of individuals familiar with MySQL.

89. Explain the `WITH` clause and provide an example.

- **Answer:** The `WITH` clause, also known as Common Table Expressions (CTE), provides a temporary result set that you can reference within a `SELECT`, `INSERT`, `UPDATE`, or `DELETE` statement. It's useful for breaking down complex queries.

```
WITH CTE_Name AS (
    SELECT column1, column2
    FROM table_name
    WHERE condition
)
SELECT * FROM CTE_Name;
```

90. What is a self-join and why would you use it?

- **Answer:** A self-join is a join where a table is joined with itself. It's useful for finding relationships within the same table.

```
SELECT A.column_name, B.column_name
FROM table_name A, table_name B
WHERE A.column_name = B.column_name;
```

91. What are the different types of subqueries? Explain with examples.

- **Answer:** There are three types:
- Scalar subquery: Returns a single value.

```
SELECT column_name
FROM table_name
WHERE another_column = (SELECT MAX(column_name) FROM table_name);
```

- Row subquery: Returns a single row.

```
SELECT column1, column2
FROM table_name
WHERE (column1, column2) = (SELECT column1, column2 FROM another_table WHE
```

- Table subquery: Returns a table.

```
SELECT column_name
FROM (
  SELECT column_name FROM table_name WHERE condition
) AS subquery_name;
```

92. How can you update data in one table based on data in another table?

- **Answer:**

```
UPDATE table1
SET table1.column_name = table2.column_name
FROM table2
WHERE table1.another_column = table2.another_column;
```

93. How can you retrieve a random row from a table?

- **Answer:**

```
SELECT column_name FROM table_name ORDER BY RAND() LIMIT 1;
```

94. What's the difference between `INNER JOIN` and `OUTER JOIN` ?

- **Answer:** `INNER JOIN` returns rows when there's a match in both tables. `OUTER JOIN` returns all rows from one table and the matching rows from the other table, filling with NULL if no match is found.

95. How can you clone a table, including both data and schema?

- **Answer:**

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

96. How do you insert multiple rows in a single SQL query?

- **Answer:**

```
INSERT INTO table_name (column1, column2)
VALUES (value1a, value2a),
       (value1b, value2b),
       ...;
```

97. Explain partitions in MySQL. How do you create them?

- **Answer:** Partitioning divides a table into smaller, more manageable pieces, yet still being treated as a single table. It can improve performance and assist in organizing large datasets.

```
CREATE TABLE table_name (
   column_name1 INT,
   column_name2 DATE
)
PARTITION BY RANGE(YEAR(column_name2)) (
   PARTITION p0 VALUES LESS THAN (1991),
   PARTITION p1 VALUES LESS THAN (1995),
   PARTITION p2 VALUES LESS THAN (1999)
);
```

98. What is the `GROUP_CONCAT` function and provide an example.

- **Answer:** It's used to concatenate values from multiple rows into a single string.

```
SELECT group_column, GROUP_CONCAT(value_column)
FROM table_name
GROUP BY group_column;
```

99. How can you prevent SQL injection in your queries?

- **Answer:** Using parameterized queries or prepared statements. In PHP, for instance, you'd use PDO or MySQLi to bind parameters.

100. How do you show the current SQL mode, and how can you change it?

- **Answer:**

```
SELECT @@sql_mode;  -- To show
SET sql_mode = 'modes';  -- To change
```

101. What is a transaction and how would you use it in MySQL?

- **Answer:** Transactions group a set of tasks into a single execution unit. If one task fails, all fail. Useful for maintaining data integrity.

```
START TRANSACTION;
INSERT INTO table_name1 ...;
INSERT INTO table_name2 ...;
COMMIT;  -- Or ROLLBACK;
```

102. What are the differences between `VARCHAR` and `TEXT` data types?

- **Answer:** While both are used to store strings, `VARCHAR` can store up to 65,535 characters and you can specify its max length, while `TEXT` can store up to 65,535 characters without specifying max length. `VARCHAR` can have a default value, but `TEXT` cannot.

103. How do you find and fix broken foreign key constraints?

- **Answer:** Identify them using a `LEFT JOIN` to find orphaned records, and either delete these records or update them to restore referential integrity.

104. How do you use `FULLTEXT` indexing in MySQL?

- **Answer:** `FULLTEXT` indexes are used for full-text searches. You can create one with:

```
CREATE FULLTEXT INDEX index_name ON table_name(column_name);
```

Then you'd search with:

```
SELECT * FROM table_name WHERE MATCH(column_name) AGAINST('search term');
```

105. How can you check for index fragmentation on a table and defragment it?

- **Answer:** You can check fragmentation using `SHOW TABLE STATUS LIKE 'table_name';` and optimize (defragment) using `OPTIMIZE TABLE table_name;`.

106. How can you convert character sets in columns?

- **Answer:**

```
ALTER TABLE table_name MODIFY column_name COLUMN_TYPE CHARACTER SET charse
```

107. How do you schedule a recurring SQL script execution in MySQL?

- **Answer:** Using MySQL's Event Scheduler. First, ensure the scheduler is on with `SHOW VARIABLES LIKE 'event_scheduler';` , then create your scheduled event.

108. What are MySQL stored procedures and how do you use them?

- **Answer:** Stored procedures are SQL codes saved in the database to be reused. Created using `CREATE PROCEDURE` , and called via `CALL procedure_name()` .

109. How would you monitor the performance of your MySQL database in real-time?

- **Answer:** Tools like `SHOW PROCESSLIST` , Performance Schema, MySQL Enterprise Monitor, and third-party tools like Percona Monitoring and Management.

110. How can you run SQL script from the command line without entering the MySQL console?

- **Answer:** Use:

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

111. What is the `EXPLAIN` keyword in MySQL?

- **Answer:** `EXPLAIN` provides a query execution plan, showing how MySQL will execute the query, which can be vital for optimization.

112. How do you enforce a column to not accept NULL values?

- **Answer:** By adding the `NOT NULL` constraint during table creation or modification.

113. How do you rename a database in MySQL?

- **Answer:** MySQL does not have a straightforward command to rename a database. Instead, one common approach is to dump the database, create a new one with the desired name, and then restore the dumped database into the new one.

114. How can you reset the auto-increment value of a column?

- **Answer:**

```
ALTER TABLE table_name AUTO_INCREMENT = value;
```

115. How can you handle time zones in MySQL?

- **Answer:** MySQL provides the `CONVERT_TZ()` function to convert datetime values across time zones. Additionally, `SET time_zone = timezone;` sets the time zone for the current session.

116. How do you retrieve only a specified number of characters from a string column?

- **Answer:**

```
SELECT LEFT(column_name, number_of_chars) FROM table_name;
```

117. What are views in MySQL and why are they used?

- **Answer:** Views are virtual tables based on the result set of an SQL statement. They encapsulate the SQL statement and present data in a simplified manner, ensuring data abstraction, protection, and to represent a subset of the data.

118. How do you find the second highest value in a column?

- **Answer:**

```
SELECT MAX(column_name)
FROM table_name
WHERE column_name NOT IN (SELECT MAX(column_name) FROM table_name);
```

These questions should serve well for interviews at product-based companies that expect a deep understanding of MySQL.

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:

```sql
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.

```sql
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:

```sql
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!

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) A
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 CURDAT
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:

```sql
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:

```sql
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:

```sql
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.

Here are more complex interview questions involving nested subqueries:

53. Retrieve the departments where the total salary expenditure exceeds the average total salary expenditure across all departments.

Answer:

```
SELECT department_id
FROM employees
GROUP BY department_id
HAVING SUM(salary) > (
    SELECT AVG(total_salary)
    FROM (
        SELECT department_id, SUM(salary) as total_salary
        FROM employees
        GROUP BY department_id
    ) AS subquery
);
```

54. Find the employee with the third highest salary without using the LIMIT clause.

Answer:

```
SELECT name, salary
FROM employees e1
```

```
WHERE 2 = (
    SELECT COUNT(DISTINCT e2.salary)
    FROM employees e2
    WHERE e2.salary > e1.salary
);
```

55. Identify departments that have less than the company-wide median number of employees.

Answer:

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

56. Get the most common job title among employees who earn above the company average.

Answer:

```
SELECT job_title
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees)
GROUP BY job_title
ORDER BY COUNT(*) DESC
LIMIT 1;
```

57. Identify employees who earn more than the average salary in both their department and the company.

Answer:

```
SELECT id, name, salary
FROM employees e1
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e1.department_id
)
AND salary > (
    SELECT AVG(salary)
    FROM employees
);
```

58. Retrieve the month (in numbers) with the highest total sales from a table of daily sales.

Answer:

```
SELECT MONTH(date) as sales_month
FROM sales
GROUP BY MONTH(date)
ORDER BY SUM(amount) DESC
LIMIT 1;
```

59. Get the department that has the maximum difference between the highest and lowest salaries.

Answer:

```
SELECT department_id, (MAX(salary) - MIN(salary)) as salary_difference
FROM employees
GROUP BY department_id
HAVING salary_difference = (
    SELECT MAX(max_salary - min_salary)
    FROM (
        SELECT department_id, MAX(salary) as max_salary, MIN(salary) as m
        FROM employees
        GROUP BY department_id
    ) AS subquery
);
```

60. Find the employee who earns the median salary in each department.

Answer:

```
SELECT e1.department_id, e1.name, e1.salary
FROM employees e1
WHERE (
    SELECT COUNT(*)
    FROM employees e2
    WHERE e2.department_id = e1.department_id AND e2.salary <= e1.salary
) = (
    SELECT COUNT(*)
    FROM employees e3
    WHERE e3.department_id = e1.department_id AND e3.salary >= e1.salary
);
```

61. Retrieve employees who earn more than their respective department's median salary.

Answer:

```
SELECT e1.name, e1.salary, e1.department_id
FROM employees e1
WHERE e1.salary > (
    SELECT AVG(salary)
    FROM (
        SELECT salary
        FROM employees e2
        WHERE e2.department_id = e1.department_id
        ORDER BY salary
        LIMIT 2 - (SELECT COUNT(*) FROM employees e3 WHERE e3.department_
        OFFSET (SELECT (COUNT(*) - 1) / 2 FROM employees e4 WHERE e4.depa
    ) AS median_subquery
);
```

62. Identify the departments where the minimum salary is greater than the maximum salary of at least one other department.

Answer:

```
SELECT DISTINCT e1.department_id
FROM employees e1
WHERE e1.salary = (
    SELECT MIN(salary)
    FROM employees
    WHERE department_id = e1.department_id
)
AND e1.salary > ANY (
    SELECT MAX(salary)
    FROM employees
    GROUP BY department_id
);
```

63. Find employees whose salary ranks in the top 3 within their department.

Answer:

```
SELECT e1.name, e1.salary, e1.department_id
FROM employees e1
WHERE (
    SELECT COUNT(DISTINCT e2.salary)
    FROM employees e2
    WHERE e2.department_id = e1.department_id AND e2.salary > e1.salary
) < 3;
```

64. Identify the department with the most diverse salary distribution, i.e., the largest difference between the highest and lowest salaries.

Answer:

```
SELECT department_id
FROM employees
GROUP BY department_id
HAVING (MAX(salary) - MIN(salary)) = (
    SELECT MAX(salary_range)
    FROM (
        SELECT (MAX(salary) - MIN(salary)) as salary_range
        FROM employees
        GROUP BY department_id
    ) AS subquery
);
```

65. Retrieve the employees who do not have the lowest salary in their department but earn less than the department average.

Answer:

```
SELECT e1.name, e1.salary, e1.department_id
FROM employees e1
WHERE e1.salary NOT IN (
    SELECT MIN(e2.salary)
    FROM employees e2
    WHERE e2.department_id = e1.department_id
)
AND e1.salary < (
    SELECT AVG(e3.salary)
    FROM employees e3
```

```
        WHERE e3.department_id = e1.department_id
  );
```

66. Determine which departments have an average salary close to the company's median salary. Assume 'close' means a difference of less than 1000.

Answer:

```
  SELECT department_id
  FROM employees
  GROUP BY department_id
  HAVING ABS(AVG(salary) - (
      SELECT AVG(median_salary)
      FROM (
          SELECT salary AS median_salary
          FROM employees
          ORDER BY salary
          LIMIT 2 - (SELECT COUNT(*) FROM employees) MOD 2
          OFFSET (SELECT (COUNT(*) - 1) / 2 FROM employees)
      ) AS median_subquery
  )) < 1000;
```

67. Find the departments where the total number of employees is above the company's average.

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 avg_subquery
  );
```

68. Identify employees who earn more than the second highest earner in their respective department.

Answer:

```
SELECT e1.name, e1.salary, e1.department_id
FROM employees e1
WHERE e1.salary > (
    SELECT MAX(e2.salary)
    FROM employees e2
    WHERE e2.department_id = e1.department_id AND e2.salary < (
        SELECT MAX(e3.salary)
        FROM employees e3
        WHERE e3.department_id = e1.department_id
    )
);
```

69. Find the departments where the top earner makes at least twice as much as the second top earner.

Answer:

```
SELECT department_id
FROM employees
GROUP BY department_id
HAVING MAX(salary) >= 2 * (
    SELECT MAX(salary)
    FROM employees e2
    WHERE e2.department_id = employees.department_id AND salary < MAX(emp
);
```

70. Retrieve the employees who have been in the company for longer than the average tenure of their respective department managers.

Answer:

```
SELECT e1.name, e1.join_date
FROM employees e1
WHERE DATEDIFF(CURDATE(), e1.join_date) > (
    SELECT AVG(DATEDIFF(CURDATE(), e2.join_date))
    FROM employees e2
    WHERE e2.id IN (
        SELECT manager_id
        FROM employees
        WHERE department_id = e1.department_id
    )
);
```

71. Identify the department with the smallest gap between the lowest and average salary.

Answer:

~~~
SELECT department_id
FROM employees
GROUP BY department_id
HAVING (AVG(salary) - MIN(salary)) = (
    SELECT MIN(gap)
    FROM (
        SELECT (AVG(salary) - MIN(salary)) AS gap
        FROM employees
        GROUP BY department_id
    ) AS gap_subquery
);



**72. Identify the employees who earn below the average salary of their p

**Answer**:

```sql
SELECT e1.name, e1.salary, YEAR(e1.join_date) AS join_year
FROM employees e1
WHERE e1.salary < (
    SELECT AVG(e2.salary)
    FROM employees e2
    WHERE YEAR(e2.join_date) = YEAR(e1.join_date)
);
~~~

73. Retrieve the employee who has the closest salary to their department's median but isn't the median earner.

Answer:

```
SELECT e1.name, e1.salary
FROM employees e1
WHERE e1.department_id IN (
    SELECT department_id
    FROM employees
)
AND e1.salary <> (
    SELECT AVG(median_salary)
    FROM (
```

```
        SELECT salary AS median_salary
        FROM employees e2
        WHERE e2.department_id = e1.department_id
        ORDER BY salary
        LIMIT 2 - (SELECT COUNT(*) FROM employees e3 WHERE e3.department_
        OFFSET (SELECT (COUNT(*) - 1) / 2 FROM employees e4 WHERE e4.depa
    ) AS median_subquery
)
ORDER BY ABS(e1.salary - (
    SELECT AVG(median_salary)
    FROM (
        SELECT salary AS median_salary
        FROM employees e5
        WHERE e5.department_id = e1.department_id
        ORDER BY salary
        LIMIT 2 - (SELECT COUNT(*) FROM employees e6 WHERE e6.department_
        OFFSET (SELECT (COUNT(*) - 1) / 2 FROM employees e7 WHERE e7.depa
    ) AS median_subquery2
))
LIMIT 1;
```

74. Determine the departments whose average tenure (time since joining) is greater than the company average.

Answer:

```
SELECT department_id
FROM employees
GROUP BY department_id
HAVING AVG(DATEDIFF(CURDATE(), join_date)) > (
    SELECT AVG(DATEDIFF(CURDATE(), join_date))
    FROM employees
);
```

75. Identify departments where more than half of the employees earn above the company's median salary.

Answer:

```
SELECT e1.department_id
FROM employees e1
WHERE e1.salary > (
    SELECT AVG(median_salary)
    FROM (
        SELECT salary AS median_salary
        FROM employees
```

```
            ORDER BY salary
            LIMIT 2 - (SELECT COUNT(*) FROM employees) MOD 2
            OFFSET (SELECT (COUNT(*) - 1) / 2 FROM employees)
        ) AS median_subquery
    )
    GROUP BY e1.department_id
    HAVING COUNT(e1.id) > 0.5 * (
        SELECT COUNT(*)
        FROM employees e2
        WHERE e2.department_id = e1.department_id
    );
```

76. Find employees who earn a salary in the top 3 of their department but are not in the top 10 company-wide.

Answer:

```
SELECT e1.name, e1.salary, e1.department_id
FROM employees e1
WHERE (
    SELECT COUNT(DISTINCT e2.salary)
    FROM employees e2
    WHERE e2.department_id = e1.department_id AND e2.salary > e1.salary
) < 3
AND e1.salary NOT IN (
    SELECT DISTINCT salary
    FROM employees
    ORDER BY salary DESC
    LIMIT 10
);
```

77. Identify employees whose salary is above the average salary of the two departments with the highest average salaries.

Answer:

```
SELECT e1.name, e1.salary
FROM employees e1
WHERE e1.salary > (
    SELECT AVG(department_avg)
    FROM (
        SELECT department_id, AVG(salary) AS department_avg
        FROM employees
        GROUP BY department_id
        ORDER BY department_avg DESC
        LIMIT 2
    ) AS top_department_subquery
);
```

78. Find employees who have a manager earning less than the lowest salary in their department.

Answer:

```
SELECT e1.name, e1.salary
FROM employees e1
JOIN employees e2 ON e1.manager_id = e2.id
WHERE e2.salary < (
    SELECT MIN(e3.salary)
    FROM employees e3
    WHERE e3.department_id = e1.department_id
);
```

79. Identify the department with the least difference between the top earner and the average salary of the department.

Answer:

```
SELECT department_id
FROM employees
GROUP BY department_id
HAVING (MAX(salary) - AVG(salary)) = (
    SELECT MIN(top_minus_avg)
    FROM (
        SELECT (MAX(salary) - AVG(salary)) AS top_minus_avg
        FROM employees
        GROUP BY department_id
    ) AS difference_subquery
);
```

80. Retrieve the employees who have the same rank (in terms of salary) in their department as they do in the company overall.

Answer:

```
SELECT e1.name, e1.salary
FROM employees e1
WHERE (
    SELECT COUNT(DISTINCT e2.salary)
    FROM employees e2
    WHERE e2.department_id = e1.department_id AND e2.salary > e1.salary
) = (
    SELECT COUNT(DISTINCT e3.salary)
    FROM employees e3
    WHERE e3.salary > e1.salary
);
```

81. Determine the departments where the third-highest earner makes more than double the department's average salary.

Answer:

```
SELECT department_id
FROM employees e1
WHERE (
    SELECT DISTINCT salary
    FROM (
        SELECT salary
        FROM employees e2
        WHERE e2.department_id = e1.department_id
        ORDER BY e2.salary DESC
        LIMIT 3
    ) AS third_top_salary_subquery
    ORDER BY salary
    LIMIT 1 OFFSET 2
) > 2 * (
    SELECT AVG(e3.salary)
    FROM employees e3
    WHERE e3.department_id = e1.department_id
)
GROUP BY department_id;
```

82. Find employees who have more direct reports (subordinates) than their manager.

Answer:

```
SELECT e1.name
FROM employees e1
WHERE (
    SELECT COUNT(*)
    FROM employees e2
    WHERE e2.manager_id = e1.id
) > (
    SELECT COUNT(*)
    FROM employees e3
```

```
        WHERE e3.manager_id = e1.manager_id
    );
```