Questions

- **SQL: What is collation (词语定序)?**

  Collation refers to the set of rules that **determines how data is sorted and compared in a database**. Collation settings specify the character set and encoding used for data storage and retrieval, as well as the rules for comparing and sorting character data.

- **SQL: What are user defined functions?**

  A user-defined function (UDF) is a custom function that is created by a user to **perform a specific operation or set of operations**. User-defined functions can be used to simplify complex queries, improve code reusability, and perform custom calculations or manipulations on data.

- **SQL: What is the stored procedure?**

  A stored procedure is a named set of SQL statements that **have been previously compiled and stored in a database server for later use**. Stored procedures can be used to summarize complex database operations or queries, and to improve performance by reducing the amount of network traffic between the application and the database server.

- **SQL: What is Self-Join?**

  A self-join is a join operation in which **a table is joined with itself**. Self-joins are useful when a table contains hierarchical or recursive data, or when a table contains a relationship between rows within the same table.

- **SQL: What is Cross-Join?**

  A cross join is a join operation that **combines every row from one table with every row from another table**. The result of a cross join between two tables is a new table that contains all possible combinations of rows from the two input tables.

- **SQL: What is Auto Increment?**

  Auto-increment is a feature that **allows a numeric column in a table to automatically increment its value when a new row is inserted**. This is typically used to generate unique primary key values for each new row added to a table.

- **SQL: What is a constraint?**

  A constraint is a rule or restriction that is placed on a table to enforce data integrity and consistency. **Constraints are used to ensure that data values in a table meet specific criteria or conditions.**

  There are several types of constraints in SQL, including:

  1. Primary key constraint: A primary key constraint is used to uniquely identify each row in a table. It ensures that the **value of the primary key column is unique and cannot be null**.
  2. Foreign key constraint: A foreign key constraint is used to establish a relationship between two tables. It ensures that the **value in a column of one table matches a value in another table's primary key column**.
  3. Unique constraint: A unique constraint ensures that the **values in a column or a group of columns are unique and cannot be duplicated**.
  4. Not null constraint: A not null constraint ensures that **a column cannot contain null values**.

5. Check constraint: A check constraint is used to restrict the values that can be inserted into a column. It ensures that **only values that meet a specific condition or set of conditions are allowed**.

- **SQL: What is subquery?**

A subquery, also known as an inner query or nested query, is a query that is **embedded within another query**. The subquery is executed first and its result is used by the outer query as a condition or value for further processing.

- **SQL: What is a Cursor?**

A cursor is a database object that **enables crossing over the rows of a result set, one row at a time**. Cursors provide a way to iterate through a set of rows returned by a query and perform operations on each row individually.

Cursor is a **Temporary Memory or Temporary Work Station**. It is Allocated by Database Server at the Time of Performing DML(Data Manipulation Language) operations on Table by User. **Cursors are used to store Database Tables**. There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors. These are explained below.

- Implicit Cursors:
  - Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.
- Explicit Cursors :
  - Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manne

- **SQL: How does an index work?**

An index works under binary-search by creating a separate data structure that contains a copy of a portion of the table data in a specific order, allowing the database to locate the data more quickly. When a query is executed, the database system uses the index to find the data more efficiently by quickly locating the relevant rows and retrieving the data from those rows.

- **SQL: What are the types of join and explain each?**

In SQL, there are four types of join:

1. Inner join: Returns **only the rows where there is a match in both tables** being joined. It is the most commonly used join and is performed using the "JOIN" keyword.
2. Left join: Returns **all the rows from the left table and the matched rows from the right table**. If there are no matching rows in the right table, it returns NULL. It is performed using the "LEFT JOIN" or "LEFT OUTER JOIN" keywords.
3. Right join: Returns **all the rows from the right table and the matched rows from the left table**. If there are no matching rows in the left table, it returns NULL. It is performed using the "RIGHT JOIN" or "RIGHT OUTER JOIN" keywords.
4. Full outer join: **Returns all the rows from both tables, including the unmatched rows.** If there are no matching rows in either table, it returns NULL. It is performed using the "FULL OUTER JOIN" or "FULL JOIN" keywords.

- **SQL: What is a foreign key?**

A foreign key is a constraint that links two tables together based on a column in each table. It creates a relationship between the two tables, where one table (the "child" table) refers to the primary key of another table (the "parent" table). The foreign key constraint ensures that the values in the child table's foreign key column match the values in the parent table's primary key column, maintaining data integrity and consistency between the related tables.

- **SQL: What is a unique key?**

  In SQL, a unique key is a constraint that ensures that the values in a column (or set of columns) are unique and cannot be duplicated in a table. This constraint is similar to a primary key constraint, but it does not necessarily have to be the table's primary key.

  A unique key constraint can be applied to one or more columns in a table and ensures that each value in the specified column(s) is unique across all rows in the table. Attempting to insert a row with a duplicate value in the unique key column(s) will result in an error, preventing the duplication of data.

- **SQL: What is a primary key?**

  In SQL, a primary key is a constraint that uniquely identifies each row in a table. It is a column or set of columns that has a unique value for each row and cannot contain null values.

  A primary key is used to enforce data integrity by ensuring that there are no duplicate rows in the table, and it also provides a way to uniquely identify each row in the table. This is important for many database operations, such as updating, deleting, and joining data from multiple tables.

- **SQL: What is DBMS?**

  DBMS stands for Database Management System. It is a software system that enables users to define, create, maintain, and control access to a database. A DBMS provides tools and interfaces to create and manage databases, as well as the ability to define and enforce data integrity and security constraints.

- **SQL: What is RDBMS?**

  RDBMS stands for Relational Database Management System. It is a type of DBMS that uses a relational model for organizing and managing data. In an RDBMS, data is stored in tables with rows and columns, and relationships between tables are established using keys.

- **SQL: What are aggregate and scalar functions?**

  In SQL, **aggregate functions are functions that operate on a set of values and return a single value as the result. These functions typically perform calculations on a group of rows, such as calculating the sum, average, minimum, or maximum value of a column.** Examples of aggregate functions in SQL include SUM, AVG, COUNT, MIN, and MAX.

  Scalar functions, on the other hand, **are functions that operate on a single value and return a single value as the result.** These functions typically perform calculations on a single row or column, **such as converting data types, manipulating strings, or performing mathematical operations**. Examples of scalar functions in SQL include **UPPER, LOWER, TRIM, CAST, and ROUND**.

  In summary, aggregate functions in SQL operate on a set of values and return a single value as the result, while scalar functions operate on a single value and return a single value as the result.

- **SQL: How to fetch alternate records from a table?**

  SELECT * FROM mytable WHERE id % 2 = 0
- **SQL: What is the command used to fetch the first 5 characters of the string?**

  SELECT LEFT(mycolumn, 5) FROM mytable;
- **SQL: Which operator is used in a query for pattern matching?**

  SELECT column1, column2, ...

  FROM table_name

  WHERE column_name LIKE pattern
- **SQL: What is a relationship and what are they?**

  In the context of SQL, a relationship refers to the connection or association between two tables in a relational database. Relationships are established between tables based on their shared data elements or fields. There are three types of relationships in SQL:
  - One-to-One Relationship: In a one-to-one relationship, **each record in one table is associated with only one record in the other table**. This type of relationship is not very common in SQL databases.
  - One-to-Many Relationship: **In a one-to-many relationship, a record in one table can be associated with multiple records in the other table, but a record in the other table can only be associated with one record in the first table.**
  - **Many-to-Many Relationship: In a many-to-many relationship, each record in one table can be associated with multiple records in the other table, and vice versa.** This type of relationship requires a third table, known as a junction or link table, to establish the connections between the two tables.
- **SQL: What is a Database?**

  In SQL, a database is an **organized collection of data stored in a structured format**. It is a software system used to manage and store data in a computerized system. A database typically consists of one or more tables, with each table containing a set of related data organized into rows and columns. The columns represent the attributes or fields of the data, while the rows represent individual records or instances of the data.
- **SQL: What is a table?**

  In SQL, **a table is a collection of related data organized into rows and columns.** Tables are used to store data in a structured format that can be easily queried and manipulated. Each column in a table represents a specific type of data, and each row represents a single record or instance of that data.
- **SQL: What is SQL?**

  SQL (Structured Query Language) is a programming language used for managing and manipulating data in a relational database management system (RDBMS). SQL is used to create, modify, and query databases, as well as manipulate the data stored in them.
- **SQL: What is a field?**

  In SQL, **a field is a specific piece of data that is stored in a table. It is also referred to as a column, attribute, or variable.** Each field in a table represents a specific type of data, such as a text string, numeric value, or date/time stamp.

  Fields are used to organize and store data in a structured format that can be easily queried and manipulated. They define the characteristics of the data stored in a table, such as its data type, length, precision, and scale.

- **SQL: What is the main difference between "BETWEEN" and "IN" condition operators?**

  The main difference between the "BETWEEN" and "IN" condition operators in SQL is that **"BETWEEN" is used to select a range of values between two values, while "IN" is used to match any value in a list of values.**

- **SQL: List some case manipulation functions in SQL?**

  In SQL, there are several case manipulation functions that can be used to change the case of the text in a column. Some of the most commonly used case manipulation functions include:

  - UPPER: Converts all characters in a string to uppercase.
  - LOWER: Converts all characters in a string to lowercase.
  - INITCAP: Converts the first character of each word in a string to uppercase and all other characters to lowercase.

- **SQL: How can you select unique records from a table?**

  SELECT DISTINCT column_name FROM table_name;

- **SQL: What is the difference between "HAVING" and a "WHERE"?**

  In SQL, the HAVING and WHERE clauses are used to filter records in a query, but they operate at different stages of the query execution.

  The **WHERE clause is used to filter records before they are grouped and aggregated,** and it is used with the SELECT, UPDATE, DELETE, and INSERT INTO statements. It allows you to specify conditions that the records must meet in order to be included in the result set. The WHERE clause is applied to each row of the table individually, and only the rows that meet the specified conditions are included in the query result.

  The **HAVING clause, on the other hand, is used to filter records after they have been grouped and aggregated by the GROUP BY clause**. It is used only with the SELECT statement, and allows you to specify conditions that the groups must meet in order to be included in the result set. The HAVING clause is applied to the result of the GROUP BY clause, and only the groups that meet the specified conditions are included in the query result.

  In summary, the WHERE clause is used to filter rows based on individual conditions, whereas the HAVING clause is used to filter groups based on aggregate conditions.

- **SQL: What does the "USING" command do?**

  In SQL, the **"USING" command is used with the JOIN clause to specify the columns that are used for joining two or more tables.**

  When using the JOIN clause, you can specify the columns to join on in one of two ways: by using the "ON" clause or by using the "USING" clause.

  The "ON" clause is used to specify the join condition that links the tables. It allows you to join tables based on any columns, regardless of their names or data types.

  The "USING" clause, on the other hand, is a shorthand notation that allows you to join tables based on one or more columns that have the same name and data type in both tables. It eliminates the need to specify the table names and column names twice, and makes the SQL code more concise.

- **SQL: What is the difference between "DELETE" and "TRUNCATE" commands?**

  In SQL, both the "DELETE" and "TRUNCATE" commands are used to remove data from a table. However, there are some key differences between these two commands.

The main difference is that the **"DELETE" command removes specific rows from a table, while the "TRUNCATE" command removes all rows from a table.** Here are some additional differences:

- Syntax: The syntax for the two commands is different. The "DELETE" command uses a "WHERE" clause to specify the conditions for deleting rows, while the "TRUNCATE" command does not require a "WHERE" clause.
- Speed: **The "TRUNCATE" command is generally faster than the "DELETE" command, because it removes all rows at once. The "DELETE" command, on the other hand, removes rows one at a time, which can be slower.**
- Rollback: The **"DELETE" command can be rolled back, meaning that the deleted rows can be restored if necessary**. The "TRUNCATE" command cannot be rolled back, because it removes all rows at once.
- Logging: **The "DELETE" command is logged in the transaction log, which can make the log file larger. The "TRUNCATE" command is not logged in the same way, which can make it a more efficient option for removing large amounts of data.**

- **SQL: What is View in SQL?**

  In SQL, a view is a **virtual table that is based on the result of an SQL query**. It provides a way to simplify complex database operations and provide a more user-friendly interface. Views do not contain any data themselves, but instead retrieve data from one or more tables in the database. They can be used to simplify complex queries, provide a subset of data for users, restrict access to certain columns or rows, and provide a logical layer over the physical schema of the database. Views are created using the "CREATE VIEW" statement and can be queried like a regular table using the "SELECT" statement.

- **SQL: What is the difference between SQL and MySQL?**

  SQL provides a standard syntax for interacting with relational databases and is supported by many different database management systems, including MySQL. MySQL, **on the other hand, is a software system that implements the SQL language** and provides additional features and functionality specific to MySQL.

- **SQL: Why does an index make things faster?**

  When a query is executed on a table, an index can help the database engine to quickly locate the rows that match the query's criteria, rather than scanning the entire table. This can significantly improve the speed of the query and reduce the amount of time required to return results.

  Indexes work by maintaining a sorted data structure that contains a copy of the data in the indexed columns, along with a reference to the original row in the table. This allows the database engine to perform binary searches or other optimized lookup algorithms to find matching rows, rather than scanning the entire table.

- **P/J: What is a lambda function?**

  A lambda function in Python or Java is a small, anonymous function that can be defined inline without a name. **It is often used as a shortcut to define a simple function that is only needed in one place.**

  Lambda functions in Python are defined using the lambda keyword, followed by the function's arguments, separated by commas, and a colon : followed by the function's body. For example, the following lambda function takes a single argument x and returns the square of that number:

Python : lambda x: x2**

Lambda functions in Java are defined using the -> syntax, which separates the function's arguments from its body. For example, the following lambda function takes a single argument x and returns the square of that number:

Java : (x) -> x * x

Lambda functions are often used in functional programming and are useful for writing concise code that can be easily read and understood.

- **P/J: What is the difference between deep and shallow copy?**

  In Python and Java, copying an object involves creating a new object with the same contents as the original. However, there are two types of copying: deep copy and shallow copy.

  A shallow copy creates a new object and then fills it with references to the same objects as the original. In other words, the new object points to the same memory locations as the original. Any changes made to the original objects will be reflected in the shallow copy and vice versa. In Python, shallow copies can be made using the copy() method or the [:] slice operator. In Java, shallow copies can be made using the clone() method.

  A deep copy, on the other hand, creates a new object and then recursively fills it with copies of the objects in the original. In other words, the new object has its own separate memory locations, and any changes made to the original objects will not be reflected in the deep copy. In Python, deep copies can be made using the deepcopy() method from the copy module. In Java, deep copies can be made using serialization.

  The choice between a shallow copy and a deep copy depends on the specific use case. **Shallow copies are generally faster and more memory-efficient, but can lead to unexpected behavior if the original objects are modified. Deep copies are safer but can be slower and more memory-intensive.**

- **P/J: What is a hash table?**

  A hash table is a data structure that allows for efficient insertion, deletion, and retrieval of key-value pairs. **It uses a hash function to map keys to indexes in an array, which allows for constant-time access to elements.** Both Python and Java have built-in implementations of hash tables, with Python calling it a dictionary and Java calling it a HashMap.

- **P/J: What is a hash map?**

  A hash map is a data structure that allows for efficient storage, retrieval, and manipulation of key-value pairs. **It uses a hash function to map keys to indexes in an array**, which enables fast access to elements. In Python, a hash map is called a dictionary, and in Java, it is called a HashMap. Hash maps are widely used in programming for implementing caching, databases, and other applications where efficient access to data is required.

- **General: Name something you failed at.**
- **General: Do you have any personal projects?**

Code Questions

- **SQL Code: Write a SQL query to get the third highest salary of an employee from employee_table?**

```
SELECT employee_id, salary
FROM (SELECT employee_id, RANK() OVER (ORDER BY salary DESC) AS rank
FROM employee_table) AS tmp
WHERE rank = 3
```

- **SQL Code: Find which order has total sales greater than 1000 by using the HAVING.**

```
SELECT order
FROM orders
GROUP BY order
HAVING SUM(sales) > 1000
```

- **SQL Code: Find all orders that are in shipped status and have a total amount greater than 1500.**

```
SELECT *
FROM orders
WHERE status = 'shipped' AND total_amount > 1500
```

- **SQL Code: Create an order table...**

```
CREATE TABLE orders (
  order_id INT PRIMARY KEY,
  customer_id INT,
  order_date DATE,
  status VARCHAR(20),
  total_amount DECIMAL(10,2))
```

- **SQL Code: Create an index for a table.**

```
CREATE INDEX index_name ON table_name (column1, column2, ...)
```

- **P/J Code: Write a function that inverts a string(将字符串中的字符顺序颠倒过来).**

```
def invert_string(string):
    inverted_string = ""
    for char in string:
        inverted_string = char + inverted_string
    return inverted_string
```

- **P/J Code: Write a function that counts the appearance of a word in a text.**

```
def count_word_occurrences(text, word):
    words = text.split()
    count = 0
    for w in words:
        if w == word:
            count += 1
    return count
```

- **P/J Code: How can you randomize the items of a list (in place)?**

```
import random
my_list = [1, 2, 3, 4, 5]
shuffled_list = my_list.copy()
random.shuffle(my_list)
print(my_list)
```

- **P/J Code: Write a program to produce Star triangle.**

```
def print_star_triangle(n):
    for i in range(n):
        for j in range(i+1):
            print("* ", end="")
        print()

n = 5
print_star_triangle(n)
```

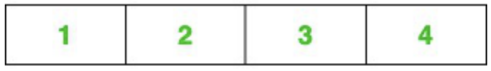
- **P/J Code: Count the number of capital letters in a file.**

```
def count_capital_letters(filename):
    count = 0
    with open(filename, "r") as f:
        for line in f:
            for c in line:
                if c.isupper():
                    count += 1
    return count
```

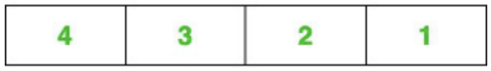
- **P/J Code: Describe an algorithm to sort a list (in place)?**

A Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of elements in the respective data structure.

Unsorted Array



Array sorted in ascending order



Array sorted in descending order