1. **What is SQL?**

SQL (Structured Query Language) is a standard language used to interact with databases. It allows users to create, modify, and manage databases, tables, and the data within them. SQL is an ANSI (American National Standards Institute) standard, and its main operations include querying data, inserting records, updating records, and deleting data.

2. **What is a database?**

A database is a structured collection of data that is stored and accessed electronically. It consists of tables, queries, views, and schemas that help organize and manage data. Databases are managed by a Database Management System (DBMS), which allows users to interact with the data.

3. **Does SQL support programming language features?**

No, SQL is not a programming language but rather a command-based language used for data manipulation. SQL lacks typical programming constructs such as loops and conditionals. Instead, it provides commands to query, update, insert, or delete data in a database.

4. **What is the difference between CHAR and VARCHAR2 datatype in SQL?**

- **CHAR** is used for fixed-length strings. If you define a column as `CHAR(5)`, any data inserted will be padded to the fixed length of 5 characters.

- **VARCHAR2** is used for variable-length strings. In `VARCHAR2(5)`, the string can be of any length up to 5 characters

without padding.

5. **What do you mean by Data Definition Language (DDL)?**

DDL refers to SQL commands used to define or modify the structure of the database, such as `CREATE`, `ALTER`, and `DROP` commands. It deals with the schema definition and table manipulation.

6. **What do you mean by Data Manipulation Language (DML)?**

DML refers to SQL commands that deal with data manipulation within a table. Examples of DML commands are `INSERT`, `UPDATE`, `DELETE`, and `SELECT`, which are used to modify or retrieve data from a database.

7. **What is a view in SQL?**

A view is a virtual table that represents the result of a SQL query. It allows users to create a structured result set from one or more tables. Views can simplify complex queries and can be used to restrict access to specific rows or columns of data.

**Syntax to create a view:**

```sql
CREATE VIEW view_name AS
SELECT column1, column2
FROM table_name
WHERE condition;
```

```

8. **What do you mean by foreign key?**

A foreign key is a field in a table that refers to the primary key of another table. It creates a relationship between the two tables, enforcing referential integrity. For example, the `C_ID` field in the `Orders` table is a foreign key referring to the `C_ID` primary key in the `Customers` table.

**Syntax:**

```sql
CREATE TABLE Orders (
    O_ID int NOT NULL,
    ORDER_NO int NOT NULL,
    C_ID int,
    PRIMARY KEY (O_ID),
    FOREIGN KEY (C_ID) REFERENCES Customers(C_ID)
);
```

9. **What are table and field?**

- **Table**: A structured set of data made up of rows (records) and columns (fields).

- **Field**: A single piece of information in a record, corresponding
```

to a column in a table.

10. **What is a primary key?**

A primary key is a unique identifier for each record in a table. It ensures that each row is uniquely identified, and there can only be one primary key per table.

11. **What is a Default constraint?**

The `DEFAULT` constraint sets a default value for a column when no value is provided during an insert. This ensures that all records have a value, even if one is not specified.

12. **What is normalization?**

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves decomposing tables based on functional dependencies to achieve certain properties like minimizing insertion, deletion, and update anomalies. The main objective is to create smaller, well-structured tables.

**13. What is Denormalization?**

Denormalization is a database optimization technique where redundant data is added to improve query performance by avoiding costly joins. It's applied after normalization to enhance database efficiency.

**14. What is a query?**

An SQL query retrieves specific data from a database. Efficient queries ensure optimal database performance by reducing resource consumption and enhancing speed.

**15. What is a subquery?**

A subquery is a query nested within another SQL query. It's often used within the WHERE clause to filter data based on the results of the subquery.

**16. What are the different operators available in SQL?**

SQL provides three types of operators:

- Arithmetic Operators: `+`, `-`, `*`, `/`

- Logical Operators: `AND`, `OR`, `NOT`

- Comparison Operators: `=`, `<`, `>`, `<=`, `>=`, `<>`

**17. What is a Constraint?**

Constraints are rules applied to table columns to enforce data integrity. Examples include `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`, and `DEFAULT`.

**18. What is Data Integrity?**

Data integrity ensures that the data in a database is accurate and consistent. It enforces rules like uniqueness, foreign key relationships, and constraints to maintain correct and reliable data.

**19. What is Auto Increment?**

Auto Increment generates unique numeric values automatically for each new record in a table, usually used for primary key columns.

**20. What is MySQL collation?**

MySQL collation defines rules for comparing and sorting characters in a particular character set. Each character set can have multiple collations, but each collation is unique to a character set.

**21. What are user-defined functions?**

User-defined functions (UDFs) are custom functions created to perform specific tasks not provided by built-in SQL functions. They can be used in SELECT statements, WHERE clauses, or INSERT and UPDATE queries.

**22. What are all types of user-defined functions?**

There are three types of user-defined functions in SQL:

1. **Scalar UDF**: Returns a single scalar value.

2. **Inline Table-Value UDF**: Returns a table, often used as an alternative to views.

3. **Multi-statement Table-Value UDF**: Returns a table, supporting multiple T-SQL statements.

**23. What is a stored procedure?**

A stored procedure is a group of SQL statements that perform one or more operations on the database. It can accept parameters, execute

queries, and return values, helping automate repetitive tasks.

**24. What are aggregate and scalar functions?**

SQL functions are divided into two main categories:

- **Aggregate Functions**: Perform operations on multiple values from a column and return a single result (e.g., `SUM()`, `COUNT()`, `AVG()`, `MIN()`, `MAX()`).

- **Scalar Functions**: Operate on individual inputs and return a single value (e.g., `UCASE()`, `LCASE()`, `LEN()`, `ROUND()`).

**25. What is an ALIAS command?**

The ALIAS command gives temporary names to columns or tables in SQL queries for better readability. Aliases are used within the query but do not change the original column or table names in the database.

Example:

```sql

SELECT CustomerName AS Name FROM Customers;

```

**26. What are Union, Minus, and Intersect commands?**

These are **set operations** used in SQL:

- **UNION**: Combines rows from two or more queries, eliminating duplicates. Use `UNION ALL` to retain duplicates.

- **INTERSECT**: Returns rows common to both queries. Use `INTERSECT ALL` to retain duplicates.

- **EXCEPT (Minus)**: Returns rows from the first query that are not present in the second. Use `EXCEPT ALL` to retain duplicates.

**27. What is T-SQL?**

**T-SQL (Transact-SQL)** is an extension of SQL developed by Microsoft, used specifically with SQL Server. It adds procedural programming capabilities, such as variables, control-of-flow statements, and error handling, to standard SQL.

**28. What is ETL in SQL?**

**ETL** stands for Extract, Transform, and Load. It is a data warehousing process where:

- **Extract**: Data is extracted from source systems.

- **Transform**: Data is cleansed and formatted in a staging area.

- **Load**: Data is loaded into a data warehouse or target database.

**29. How to copy tables in SQL?**

In SQL, you can create a copy of a table using the following syntax:

```sql

CREATE TABLE NewTable LIKE OriginalTable;

```

This creates an empty copy of the original table, preserving its structure.

**30. What is SQL injection?**

**SQL Injection** is a technique used by attackers to insert malicious SQL statements into an input field to manipulate or gain unauthorized access to a database. It's a common web hacking technique that exploits security vulnerabilities in an application's software.

**31. Can we disable a trigger? If yes, how?**

Yes, we can disable a trigger in PL/SQL using the `ALTER TRIGGER` statement with the `DISABLE` option. To disable all triggers associated with a table, use the `ALTER TABLE` statement with the `DISABLE ALL TRIGGERS` option. Disabling triggers can be useful when loading large amounts of data without firing triggers or when an object the trigger references is unavailable.

**32. What are the differences between SQL and PL/SQL?**

- **SQL** is a query execution language, while **PL/SQL** is a complete programming language.

- **SQL** is data-oriented and declarative, used for manipulating data, while **PL/SQL** is procedural, used for creating applications.

- SQL executes one statement at a time, while PL/SQL can execute blocks of statements.

- SQL tells the database **what** to do, whereas PL/SQL tells it **how** to do it.

- You can embed SQL in PL/SQL, but not PL/SQL in SQL.

**33. What is the difference between BETWEEN and IN operators in SQL?**

- **BETWEEN**: Retrieves rows based on a range of values.

  Example: `SELECT * FROM Students WHERE ROLL_NO BETWEEN 20 AND 30;`

- **IN**: Retrieves rows based on specific sets of values.

  Example: `SELECT * FROM Students WHERE ROLL_NO IN (20, 21, 23);`

**34. Write an SQL query to find the names of employees starting with ‘A’.**

```sql
SELECT * FROM Employees WHERE EmpName LIKE 'A%';
```

This query uses the `LIKE` operator to filter employees whose names start with the letter 'A'.

**35. What is the difference between primary key and unique constraints?**

- A **Primary Key** cannot contain `NULL` values, while **Unique Constraints** can.

- A table can have only one **Primary Key**, but multiple **Unique Constraints**.

- **Primary Key** automatically creates a clustered index, whereas **Unique Constraint** does not.

**36. What is a join in SQL? What are the types of joins?**

A **JOIN** is used to combine data from two or more tables based on a common field.

- **INNER JOIN**: Returns matching rows from both tables.

- **LEFT JOIN**: Returns all rows from the left table and matching rows from the right.

- **RIGHT JOIN**: Returns all rows from the right table and matching rows from the left.

- **FULL JOIN**: Combines `LEFT` and `RIGHT JOIN`, returning all rows from both tables, with `NULL` for non-matching rows.

**37. What is an index?**

An **index** is a data structure that improves the speed of data retrieval operations on a database table, but requires additional space and can slow down write operations. It allows for faster searches by storing copies of table data in a specific order.

**38. What is the On Delete Cascade constraint?**

An **ON DELETE CASCADE** constraint automatically deletes rows in the child table when the corresponding rows in the parent table are deleted.

**39. Explain WITH clause in SQL.**

The **WITH** clause defines a temporary table that can be referenced within a query. It allows you to write more readable and modular queries, especially when dealing with complex subqueries.

**40. What are all the different attributes of indexes?**

- **Access Types**: The type of data access (e.g., value-based search, range access).

- **Access Time**: The time required to retrieve data.

- **Insertion Time**: Time needed to insert new data and update the index.

- **Deletion Time**: Time needed to delete an item and update the index.

- **Space Overhead**: Additional space required by the index.

**41. What is a Cursor?**

A **Cursor** is a temporary work area allocated by the database server when performing operations like fetching multiple rows from a table. Cursors allow row-by-row processing of the result set.

**42. Write down various types of relationships in SQL.**

- **One-to-One Relationship**

- **One-to-Many Relationship**

- **Many-to-One Relationship**

- **Self-Referencing Relationship**

**43. What is a trigger?**

A **Trigger** is a set of actions executed automatically when a specific event occurs in the database, such as inserting, updating, or deleting data. Triggers enforce complex constraints or perform tasks that are not possible with standard SQL constraints.

### 44. Difference Between SQL DELETE and SQL TRUNCATE Commands

| Aspect | SQL DELETE | SQL TRUNCATE |
|---|---|---|
| **Operation** | Removes rows one at a time, logging each row separately. | Removes data by deallocating data pages, logging page deallocations only. |
| **Speed** | Slower because it logs each row deletion. | Faster since it deals with pages rather than individual rows. |
| **Permissions Required** | Requires `DELETE` permission on the table. | Requires `ALTER` permission on the table. |
| **Identity Column** | Retains identity values. | Resets the identity column to its seed value. |
| **Indexed Views** | Can be used with indexed views. | Cannot be used with indexed views. |

|

### 45. Difference Between Clustered and Non-Clustered Index

| Aspect | Clustered Index | Non-Clustered Index |
|---|---|---|
| **Speed** | Faster. | Slower compared to clustered. |
| **Memory Usage** | Requires less memory. | Requires more memory. |
| **Data Storage** | Actual data stored at the index. | Stores data copies and points to actual data. |
| **Index Count** | Only one per table. | Multiple non-clustered indexes per table. |
| **Data Organization** | Rearranges the physical order of data. | Does not affect physical order of data. |
| **Disk Storage** | Has inherent ability to store data. | Relies on additional pointers for storage. |

### 46. What is a Live Lock?

A **Live Lock** occurs when two or more processes continuously respond to each other without making progress. Unlike a **Deadlock**, where processes wait indefinitely, live locks remain active but do no useful work.

### 47. What is CASE WHEN in SQL?

The **CASE** statement allows for **conditional logic** in SQL, similar to **IF/THEN** logic in programming languages. It is useful for filtering or query optimization.

- **Syntax 1**: `CASE value WHEN condition THEN result [ELSE result] END`

- **Syntax 2**: `CASE WHEN condition THEN result [ELSE result] END`

### 48. Case Manipulation Functions in SQL

There are three case manipulation functions in SQL:

1. **LOWER()**: Converts all characters to lowercase.
   - Example: `LOWER('HELLO')` returns `'hello'`.

2. **UPPER()**: Converts all characters to uppercase.
   - Example: `UPPER('hello')` returns `'HELLO'`.

3. **INITCAP()**: Converts the first character of each word to uppercase.
   - Example: `INITCAP('hello world')` returns `'Hello World'`.

### 49. Difference Between Local and Global Variables

| Aspect | Local Variable | Global Variable |
|---|---|---|
| Scope | Only available within the function that defines it. | Available across all functions after being defined. |
| Usage | Specific to individual function logic. | Can be accessed and modified by any function. |

### 50. Function to Remove Spaces at the End of a String

In SQL, the `TRIM()` function removes spaces from a string, including those at the end.

- **Syntax**: `TRIM('  your_string  ')` returns `'your_string'`.

### 51. Difference Between SQL TRUNCATE and DROP Commands

| Aspect | SQL DROP | SQL TRUNCATE |
|---|---|---|
| **Operation** | Removes the table definition and data. | Removes all rows from the table but keeps the structure. |
| **Freeing Space** | Frees the table space from memory. | Does not free space but marks the pages for reuse. |
| **Type of Command** | DDL (Data Definition Language). | DDL (Data Definition Language). |
| **Constraints** | Removes integrity constraints with the table. | Retains constraints on the table. |
| **Undo Space** | Does not use undo space. | Uses minimal undo space, less than DELETE. |
| **Speed** | Quick but can cause complications. | Faster than DROP and suitable for clearing data efficiently. |

**52. LIKE Operator for Pattern Matching:**

The `LIKE` operator is used to search for a specified pattern in a column. It is typically used in a `WHERE` clause. For example:

```sql
SELECT * FROM Employees WHERE Name LIKE 'A%';
```

**53. ORDER BY Statement:**

`ORDER BY` is used to sort the result-set in ascending (`ASC`) or descending (`DESC`) order by one or more columns. For instance:

```sql
```

```
SELECT * FROM Employees ORDER BY Salary DESC;
```

**54. HAVING Statement:**

`HAVING` is used to filter records after the `GROUP BY` clause, typically with aggregate functions. Example:

```sql
SELECT Department, COUNT(*)
FROM Employees
GROUP BY Department
HAVING COUNT(*) > 10;
```

**55. SQL AND/OR Statements:**

- `AND`: Returns records where **both** conditions are true.
- `OR`: Returns records where **either** condition is true.

Example:

```sql
SELECT * FROM Employees WHERE Age > 30 AND Department = 'HR';
```

**56. BETWEEN Statement:**

Used to filter records within a range of values, including both endpoints. Example:

```sql

SELECT * FROM Orders WHERE OrderDate BETWEEN '2023-01-01' AND '2023-12-31';

```

**57. COMMIT and ROLLBACK Commands:**

- `COMMIT`: Saves the transaction permanently.

- `ROLLBACK`: Undoes changes made by the current transaction.

Example:

```sql

COMMIT; -- to finalize

ROLLBACK; -- to revert

```

**58. ACID Properties:**

- **Atomicity**: Ensures all operations are completed successfully or none at all.

- **Consistency**: Ensures the database transitions from one valid state to another.

- **Isolation**: Ensures transactions are executed independently of one another.

- **Durability**: Ensures that changes made by a transaction are

permanent.

**59. NULL Values vs Zero or Blank:**

`NULL` represents missing or undefined data, while zero (`0`) and blank (`''`) represent specific values. Two `NULL` values are not equal in SQL.

**60. Group Functions in SQL:**

Aggregate functions such as `COUNT()`, `SUM()`, `AVG()`, `MIN()`, and `MAX()` operate on sets of rows to return a single result.

**61. MERGE Statement:**

Combines `INSERT`, `UPDATE`, and `DELETE` into one operation. Example:

```sql
MERGE INTO TargetTable AS T
USING SourceTable AS S
ON T.ID = S.ID
WHEN MATCHED THEN UPDATE SET T.Name = S.Name
WHEN NOT MATCHED THEN INSERT (ID, Name) VALUES (S.ID, S.Name);
```

**62. Fetching Common Records from Two Tables (JOIN):**

You can use joins to fetch common records:

```sql
SELECT A.column1, B.column2
FROM TableA A
JOIN TableB B ON A.ID = B.ID;
```

Here are the explanations for SQL questions 63 to 70:

**63. Advantages of PL/SQL Functions:**

- **Single Database Call**: Reduces the number of database calls by bundling multiple SQL statements into a single block.
- **Modularity**: Code can be divided into smaller, manageable modules.
- **Reusability**: Functions can be reused in multiple applications.
- **Security**: The code is stored in the database, protecting internal details from users.

**64. SQL Query to Display the Current Date:**

The `CURRENT_DATE` function returns the current date.

```sql
SELECT CURRENT_DATE;
```

**65. Nested Triggers:**

A trigger can modify data that causes another trigger to fire, creating a chain of trigger events. This is known as a **nested trigger**.

**66. Finding Constraint Information in a Table:**

You can find constraint information using SQL Server's data dictionary tables such as `INFORMATION_SCHEMA.TABLE_CONSTRAINTS` and `INFORMATION_SCHEMA.KEY_COLUMN_USAGE`. Example:

```sql
SELECT *
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 'your_table_name';
```

**67. Avoiding Duplicates Without Using `DISTINCT`:**

There are several ways to avoid duplicates:

- **Using `ROW_NUMBER()`**: Assigns a unique row number to records.
- **Using Self Join**: Joining the table to itself to identify unique rows.
- **Using `GROUP BY`**: Aggregates rows based on column values.

**68. Difference Between `NVL` and `NVL2` Functions:**

- **`NVL(expr1, expr2)`**: Replaces `NULL` in `expr1` with `expr2`.

```sql
```

```
SELECT NVL(salary, 0) FROM Employees;
```

- **`NVL2(expr1, expr2, expr3)`**: Returns `expr2` if `expr1` is not null; otherwise, returns `expr3`.

```sql
SELECT NVL2(salary, bonus, 0) FROM Employees;
```

**69. Difference Between `COALESCE()` and `ISNULL()`:**

- **`COALESCE()`**: Returns the first non-NULL expression among its arguments.

```sql
SELECT COALESCE(address, city, 'Unknown') FROM Employees;
```

- **`ISNULL()`**: Replaces NULL with a specified value (SQL Server specific).

```sql
SELECT ISNULL(salary, 0) FROM Employees;
```

**70. Operator Used for Appending Strings:**

The **concatenation operator** (`||`) is used to append two strings in SQL.

```sql
```

```
SELECT 'FirstName' || ' ' || 'LastName' AS FullName FROM Employees;
```