1. **What is SQL and short-cut for?**

**Answer:**

* SQL stands for Structured Query Language.
* SQL is used to communicate with a database.
* SQL lets you access and manipulate databases.

1. **What SQl can do?**

**Answer:**

* SQL can retrieve data from a database
* SQL can insert records in a database
* SQL can update records in a database
* SQL can delete records from a database
* SQL can create new databases
* SQL can create new tables in a database
* SQL can create stored procedures in a database
* SQL can create views in a database
* SQL can set permissions on tables, procedures, and views
* SQL can execute queries against a database

1. **What SQL data type?**

**Answer:**

* SQL Data Type is an attribute that specifies the type of data of any object.
* SQL Server offers six categories of data types.

## 1) Exact Numeric Data Types:

* Such as: bigint,int,smallint,tinyint,bit,decimal,numeric,money,smallmoney.

## 2) Approximate Numeric Data Types:

## - such as : Float ,real.

## 3) Date and Time Data Types:

## - Such as :dataTime,smalldatatime,date,time.

## 4) Date and Time Data Types:

## -Such as: char,varchar,varchar(max),text.

## 5) Binary Data Types:

## -Such as: binary ,var binary, var binary(max),image.

## 6) Misc Data Types:

## - such as:sql-variant , timestamp , uniqueidentifier ,xml ,cursor,table.

## Create database?

**Answer:**

* statement is used to create a new SQL database.
* **Syntax:** create database databaseName ;

1. **Create tables?**

**Answer:**

* statement is used to create a new table.
* **Syntax:** create table table\_name(

Column1 datatype,

Column2 datatype,

Column3 datatype

);

1. **Drop table?**

* statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.
* **Syntax:** drop table table\_name;

1. **Drop columns?**

* Statement is used to delete a column in an existing table.
* **Syntax:**

alter table table\_name drop column column\_name;

1. **Select?**

* statement is used to select data from a database.
* The data returned is stored in a result table, called the result-set.
* **Syntax:**

select column1, column2  
from table\_name;

1. **Insert?**

* The SQL INSERT INTO Statement is used to add new rows of data to a table in the database.
* **Syntax:**

There are two basic syntax.

1) INSERT INTO TABLE\_NAME (column1, column2, column3)

VALUES (value1, value2, value3);

2) INSERT INTO TABLE\_NAME VALUES (value1,value2,value3);

1. **Delete?**

* is used to delete the existing records from a table.
* **Syntax:**

DELETE FROM table\_name WHERE [condition];

1. **Where clause ?**

* The SQL WHERE clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table.
* The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement.
* **Syntax:**

SELECT column1, column2, columnN

FROM table\_name

WHERE [condition]

1. **Update?**

* The SQL UPDATE Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.
* **Syntax:**

UPDATE table\_name

SET column1 = value1, column2 = value2...., columnN = valueN

WHERE [condition];

1. **Aliases?**

* You can rename a table or a column temporarily by giving another name known as **Alias**. The use of table aliases is to rename a table in a specific SQL statement. The renaming is a temporary change and the actual table name does not change in the database. The column aliases are used to rename a table's columns for the purpose of a particular SQL query.
* **Syntax:**

The basic syntax of a **table** alias is:

SELECT column1, column2

FROM table\_name AS alias\_name

WHERE [condition];

The basic syntax of a **column** alias is:

SELECT column\_name AS alias\_name

FROM table\_name

WHERE [condition];

1. **Joins?**

* The SQL **Joins** clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

1. **Types of join?**

* There are different types of joins available in SQL:
* [INNER JOIN](https://www.tutorialspoint.com/sql/sql-inner-joins.htm) : returns rows when there is a match in both tables.
* [LEFT JOIN](https://www.tutorialspoint.com/sql/sql-left-joins.htm) : returns all rows from the left table, even if there are no matches in the right table.
* [RIGHT JOIN](https://www.tutorialspoint.com/sql/sql-right-joins.htm) : returns all rows from the right table, even if there are no matches in the left table.
* [FULL JOIN](https://www.tutorialspoint.com/sql/sql-full-joins.htm) : returns rows when there is a match in one of the tables.
* [SELF JOIN](https://www.tutorialspoint.com/sql/sql-self-joins.htm) : is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.
* [CARTESIAN JOIN](https://www.tutorialspoint.com/sql/sql-cartesian-joins.htm) : returns the Cartesian product of the sets of records from the two or more joined tables.

1. **sql functions?**

* SQL has many built-in functions for performing calculations on data
* **SQL Aggregate functions:**

SQL aggregate functions return a single value, calculated from values in a column.

* AVG() - Returns the average value
* COUNT() - Returns the number of rows
* FIRST() - Returns the first value
* LAST() - Returns the last value
* MAX() - Returns the largest value
* MIN() - Returns the smallest value
* SUM() - Returns the sum
* **SQL Scalar functions**:

SQL scalar functions return a single value, based on the input value.

* UCASE() - Converts a field to upper case
* LCASE() - Converts a field to lower case
* MID() - Extract characters from a text field
* LEN() - Returns the length of a text field
* ROUND() - Rounds a numeric field to the number of decimals specified
* NOW() - Returns the current system date and time
* FORMAT() - Formats how a field is to be displayed

1. **stored procedures?**

* A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
* So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.
* You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.
* **Syntax:**

CREATE PROCEDURE procedure\_name  
AS  
sql\_statement  
GO;

* EXEC procedure\_name;

1. **Primary key?**

* The PRIMARY KEY constraint uniquely identifies each record in a table.
* Primary keys must contain UNIQUE values, and cannot contain NULL values.
* A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

1. **foreign key?**

* A FOREIGN KEY is a key used to link two tables together.
* A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
* The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

1. **Types of data types in SQL?**
2. **String data type:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data type** | **Description** | **Max size** | **Storage** |
| char(n) | Fixed width character string | 8,000 characters | Defined width |
| varchar(n) | Variable width character string | 8,000 characters | 2 bytes + number of chars |
| varchar(max) | Variable width character string | 1,073,741,824 characters | 2 bytes + number of chars |
| text | Variable width character string | 2GB of text data | 4 bytes + number of chars |
| nchar | Fixed width Unicode string | 4,000 characters | Defined width x 2 |
| nvarchar | Variable width Unicode string | 4,000 characters |  |
| nvarchar(max) | Variable width Unicode string | 536,870,912 characters |  |
| ntext | Variable width Unicode string | 2GB of text data |  |
| binary(n) | Fixed width binary string | 8,000 bytes |  |
| varbinary | Variable width binary string | 8,000 bytes |  |
| varbinary(max) | Variable width binary string | 2GB |  |
| image | Variable width binary string | 2GB |  |

### **Numeric data types:**

|  |  |  |
| --- | --- | --- |
| **Data type** | **Description** | **Storage** |
| bit | Integer that can be 0, 1, or NULL | 1 byte |
| tinyint | Allows whole numbers from 0 to 255 | 2 bytes |
| smallint | Allows whole numbers between -32,768 and 32,767 | 4 bytes |
| int | Allows whole numbers between -2,147,483,648 and 2,147,483,647 | 8 bytes |
| bigint | Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807 | 5-17 bytes |
| decimal(p,s) | Fixed precision and scale numbers.  Allows numbers from -10^38 +1 to 10^38 –1.  The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18.The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0 | 5-17 bytes |
| numeric(p,s) | Fixed precision and scale numbers.  Allows numbers from -10^38 +1 to 10^38 –1.  The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18.  The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0 | 5-17 bytes |
| smallmoney | Monetary data from -214,748.3648 to 214,748.3647 | 4 bytes |
| money | Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807 | 8 bytes |
| float(n) | Floating precision number data from -1.79E + 308 to 1.79E + 308.  The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53. | 4 or 8 bytes |
| real | Floating precision number data from -3.40E + 38 to 3.40E + 38 | 4 bytes |

### **Date and Time data types:**

|  |  |  |
| --- | --- | --- |
| **Data type** | **Description** | **Storage** |
| datetime | From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds | 8 bytes |
| datetime2 | From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds | 6-8 bytes |
| smalldatetime | From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute | 4 bytes |
| date | Store a date only. From January 1, 0001 to December 31, 9999 | 3 bytes |
| time | Store a time only to an accuracy of 100 nanoseconds | 3-5 bytes |
| datetimeoffset | The same as datetime2 with the addition of a time zone offset | 8-10 bytes |
| timestamp | Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable |  |

### **Other data types:**

|  |  |
| --- | --- |
| **Data type** | **Description** |
| sql\_variant | Stores up to 8,000 bytes of data of various data types, except text, ntext, and timestamp |
| uniqueidentifier | Stores a globally unique identifier (GUID) |
| xml | Stores XML formatted data. Maximum 2GB |
| cursor | Stores a reference to a cursor used for database operations |
| table | Stores a result-set for later processing |

1. **Like?**

* The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
* There are two wildcards often used in conjunction with the LIKE operator:
* % - The percent sign represents zero, one, or multiple characters
* \_ - The underscore represents a single character
* **Syntax:**

SELECT column1, column2  
 FROM table\_name  
 WHERE columnN LIKE pattern;

# **Wildcards?**

* A wildcard character is used to substitute one or more characters in a string.
* Wildcard characters are used with the [SQL LIKE](https://www.w3schools.com/sql/sql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Description** | **Example** |
| % | Represents zero or more characters | bl% finds bl, black, blue, and blob |
| \_ | Represents a single character | h\_t finds hot, hat, and hit |
| [] | Represents any single character within the brackets | h[oa]t finds hot and hat, but not hit |
| ^ | Represents any character not in the brackets | h[^oa]t finds hit, but not hot and hat |
| - | Represents a range of characters | c[a-b]t finds cat and cbt |

1. **IN ?**

* The IN operator allows you to specify multiple values in a WHERE clause.
* The IN operator is a shorthand for multiple OR conditions.
* **Syntax:**

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

1. **Between?**

* The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
* The BETWEEN operator is inclusive: begin and end values are included.
* **Syntax:**

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name BETWEEN value1 AND value2;

1. **Types of Relations?**

* one-to-one
* one-to-many, and
* many-to-many