**1. What is SQL?**

It stands for [**Structured Query Language,**](https://www.datacamp.com/blog/all-about-sql-the-essential-language-for-database-management) and it's a programming language used for interaction with relational database management systems (RDBMS). This includes fetching, updating, inserting, and removing data from tables.

**2. What are SQL dialects? Give some examples.**

The various versions of SQL, both free and paid, are also called SQL dialects. All the flavours of SQL have a very similar syntax and vary insignificantly only in additional functionality. Some examples are Microsoft SQL Server, PostgreSQL, MySQL, SQLite, T-SQL, Oracle, and MongoDB.

**3. Give some examples of common SQL commands of each type.**

* **DDL(Definition):** CREATE, ALTER TABLE, DROP, TRUNCATE, and ADD COLUMN
* **DML(Manipulation):** UPDATE, DELETE, and INSERT
* **DCL(Control):** GRANT and REVOKE
* **TCL(Transmission control):** COMMIT, SET TRANSACTION, ROLLBACK, and SAVEPOINT
* **DQL(Query):** – SELECT

**4. What is DBMS, and what types of DBMS do you know?**

It stands for Database Management System, a software package used to perform various operations on the data stored in a database, such as accessing, updating, wrangling, inserting, and removing data. There are various types of DBMS, such as relational, hierarchical, network, graph, or object-oriented. These types are based on the way the data is organized, structured, and stored in the system.

**5. What is RDBMS? Give some examples of RDBMS.**

It stands for Relational Database Management System. It's the most common type of DBMS used for working with data stored in multiple tables related to each other by means of shared keys. The SQL programming language is designed to interact with RDBMS. Some examples of RDBMS are MySQL, PostgreSQL, Oracle, MariaDB, etc.

**6. What is a subquery?**

Also called an inner query, a query placed inside another query, or an outer query. A subquery may occur in the clauses such as SELECT, FROM, WHERE, UPDATE, etc. It's also possible to have a subquery inside another subquery. The innermost subquery is run first, and its result is passed to the containing query (or subquery).

**7. What types of SQL subqueries do you know?**

* **Single-row** – returns at most one row.
* **Multi-row** – returns at least two rows.
* **Multi-column** – returns at least two columns.
* **Correlated** – a subquery related to the information from the outer query.
* **Nested** – a subquery inside another subquery.

**8. What is a constraint, and why use constraints?**

A set of conditions defining the type of data that can be input into each column of a table. Constraints ensure data integrity in a table and block undesired actions.

**9. What SQL constraints do you know?**

* DEFAULT – provides a default value for a column.
* UNIQUE – allows only unique values.
* NOT NULL – allows only non-null values.
* PRIMARY KEY – allows only unique and strictly non-null values (NOT NULL and UNIQUE).
* FOREIGN KEY – provides shared keys between two or more tables.

**10. What is a join?**

A clause used to combine and retrieve records from two or multiple tables. SQL tables can be joined based on the relationship between the columns of those tables. Check out our [SQL joins**tutorial**](https://www.datacamp.com/tutorial/introduction-to-sql-joins) for more context, plus our dedicated guide to [**SQL joins interview questions**](https://www.datacamp.com/blog/top-sql-joins-interview-questions).

**11. What types of joins do you know?**

* (INNER) JOIN – returns only those records that satisfy a defined join condition in both (or all) tables. It's a default SQL join.
* LEFT (OUTER) JOIN – returns all records from the left table and those records from the right table that satisfy a defined join condition.
* RIGHT (OUTER) JOIN – returns all records from the right table and those records from the left table that satisfy a defined join condition.
* FULL (OUTER) JOIN – returns all records from both (or all) tables. It can be considered as a combination of left and right joins.

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

A column (or multiple columns) of a table to which the PRIMARY KEY constraint was imposed to ensure unique and non-null values in that column. In other words, a primary key is a combination of the NOT NULL and UNIQUE constraints. The primary key uniquely identifies each record of the table. Each table should contain a primary key and can't contain more than one primary key.

**13. What is a unique key?**

A column (or multiple columns) of a table to which the UNIQUE constraint was imposed to ensure unique values in that column, including a possible NULL value (the only one).

**14. What is a foreign key?**

A column (or multiple columns) of a table to which the FOREIGN KEY constraint was imposed to link this column to the primary key in another table (or several tables). The purpose of foreign keys is to keep connected various tables of a database.

**15. What is an index?**

A special data structure related to a database table and used for storing its important parts and enabling faster data search and retrieval. Indexes are especially efficient for large databases, where they significantly enhance query performance.

**16. What types of indexes do you know?**

* **Unique index** – doesn't allow duplicates in a table column and hence helps maintain data integrity.
* **Clustered index** – defines the physical order of records of a database table and performs data searching based on the key values. A table can have only one clustered index.
* **Non-clustered index** – keeps the order of the table records that don't match the physical order of the actual data on the disk. It means that the data is stored in one place and a non-clustered index – in another one. A table can have multiple non-clustered indexes.

**17. What is a schema?**

A collection of database structural elements such as tables, stored procedures, indexes, functions, and triggers. It shows the overall database architecture, specifies the relationships between various objects of a database, and defines different access permissions for them. Read our [**database schema guide**](https://www.datacamp.com/tutorial/database-schema) for a deeper understanding.

**18. What types of SQL operators do you know?**

* **Arithmetic** (+, -, \*, /, etc.)
* **Comparison** (>, <, =, >=, etc.)
* **Compound** (+=, -=, \*=, /=, etc.)
* **Logical** (AND, OR, NOT, BETWEEN, etc.)
* **String** (%, \_, +, ^, etc.)
* **Set** (UNION, UNION ALL, INTERSECT, and MINUS (or EXCEPT))

**19. How to select common records from two tables?**

Using the INTERSECT statement:

SELECT \* FROM table\_1

INTERSECT

SELECT \* FROM table\_1;

**20. What is a NULL value? How is it different from zero or a blank space?**

A NULL value indicates the absence of data for a certain cell of a table. Instead, zero is a valid numeric value, and an empty string is a legal string of zero length.

**21. What is the difference between SQL and NoSQL?**

SQL databases are relational, structured, and use tables with predefined schemas, while NoSQL databases are non-relational, schema-less, and designed to handle unstructured or semi-structured data.

**22. What are some common challenges when working with SQL databases?**

Challenges include performance tuning for large datasets, managing indexing strategies, ensuring data integrity with constraints, handling concurrent transactions, and optimizing query execution.

**23. What aggregate functions do you know?**

* AVG() – returns the average value
* SUM() – returns the sum of values
* MIN() – returns the minimum value
* MAX() – returns the maximum value
* COUNT() – returns the number of rows, including those with null values
* FIRST() – returns the first value from a column
* LAST()– returns the last value from a column

**24. What scalar functions do you know?**

* LEN() (in other SQL flavors – LENGTH()) – returns the length of a string, including the blank spaces
* UCASE() (in other SQL flavors – UPPER()) – returns a string converted to the upper case
* LCASE() (in other SQL flavors – LOWER()) – returns a string converted to the lower case
* INITCAP() – returns a string converted to the title case (i.e., each word of the string starts from a capital letter)
* MID() (in other SQL flavors – SUBSTR()) – extracts a substring from a string
* ROUND() – returns the numerical value rounded to a specified number of decimals
* NOW() – returns the current date and time

**25. What are case manipulation functions? Give some examples.**

Case manipulation functions represent a subset of character functions, and they're used to change the case of the text data. With these functions, we can convert the data into the upper, lower, or title case.

* UCASE() (in other SQL flavors – UPPER()) – returns a string converted to the upper case
* LCASE() (in other SQL flavors – LOWER()) – returns a string converted to the lower case
* INITCAP() – returns a string converted to the title case (i.e., each word of the string starts from a capital letter)

**26. What are character manipulation functions? Give some examples.**

Character manipulation functions represent a subset of character functions, and they're used to modify the text data.

* CONCAT() – joins two or more string values appending the second string to the end of the first one
* SUBSTR() – returns a part of a string satisfying the provided start and end points
* LENGTH() (in other SQL flavors – LEN()) – returns the length of a string, including the blank spaces
* REPLACE() – replaces all occurrences of a defined substring in a provided string with another substring
* INSTR() – returns the numeric position of a defined substring in a provided string
* LPAD() and RPAD() – return the padding of the left-side/right-side character for right-justified/left-justified value
* TRIM() – removes all the defined characters, as well as white spaces, from the left, right, or both ends of a provided string

**27. What is the difference between local and global variables?**

Local variables can be accessed only inside the function in which they were declared. Instead, global variables, being declared outside any function, are stored in fixed memory structures and can be used throughout the entire program.

**28. What is the difference between LEFT JOIN and LEFT OUTER JOIN?**

There is no [**difference between LEFT JOIN and LEFT OUTER JOIN**](https://www.datacamp.com/tutorial/left-join-vs-left-outer-join). They are interchangeable. SQL allows the OUTER keyword to be optional, so LEFT JOIN is simply a shorthand for LEFT OUTER JOIN. Both return all records from the left table and the matching records from the right table.

**29. What is indexing in SQL, and how does it improve performance?**

Indexing creates a special data structure that speeds up data retrieval by allowing the database to find rows more efficiently. It works like an optimized lookup table, reducing the need for full table scans. However, excessive indexing can slow down insert, update, and delete operations due to the need for index maintenance.

**30. What is a stored procedure, and how is it different from a function?**

A [**stored procedure**](https://www.datacamp.com/tutorial/sql-stored-procedure) is a precompiled set of SQL statements that can be executed as a single unit to perform a specific task. Unlike functions, stored procedures can return multiple result sets and modify database objects, while functions typically return a single value and cannot modify data.

**31. What set operators do you know?**

* UNION – returns the records obtained by at least one of two queries (excluding duplicates)
* UNION ALL – returns the records obtained by at least one of two queries (including duplicates)
* INTERSECT – returns the records obtained by both queries
* EXCEPT (called MINUS in MySQL and Oracle) – returns only the records obtained by the first query but not the second one

**32. What is the difference between a primary key and a unique key?**

While both types of keys ensure unique values in a column of a table, the first one uniquely identifies each record of the table, and the second one prevents duplicates in that column.

**33. What is a composite primary key?**

The primary key of a table, based on multiple columns.

**34. What is a view, and why use it?**

A virtual table containing a subset of data retrieved from one or more database tables (or other views). Views take very little space, simplify complex queries, limit access to the data for security reasons, enable data independence, and summarize data from multiple tables.

**35. Can we create a view based on another view?**

Yes. This is also known as nested views. However, we should avoid nesting multiple views since the code becomes difficult to read and debug.

**36. Can we still use a view if the original table is deleted?**

No. Any views based on that table will become invalid after deleting the base table. If we try to use such a view anyway, we'll receive an error message.

**37. What is normalization in SQL, and why use it?**

Normalization is a process of database design that includes organizing and restructuring data in a way to reduce data redundancy, dependency, duplication, and inconsistency. This leads to enhanced data integrity, more tables within the database, more efficient data access and security control, and greater query flexibility.

**38. What is denormalization in SQL, and why use it?**

Denormalization is the process opposite of normalization: it introduces data redundancy and combines data from multiple tables. Denormalization optimizes the performance of the database infrastructure in situations when read operations are more important than write operations since it helps avoid complex joins and reduces the time of query running.

**39. What is the difference between nested and correlated subqueries?**

A correlated subquery is an inner query nested in a bigger (outer) query that refers to the values from the outer query for its execution, meaning that a correlated subquery depends on its outer query. Instead, a non-correlated subquery doesn't rely on the data from the outer query and can be run independently of it.

**40. What is the difference between clustered and non-clustered indexes?**

While a clustered index **defines the physical order of records** of a table and performs data searching based on the key values, a non-clustered index **keeps the order of records that do not match the physical order of the actual data** on the disk. A table can have only one clustered index but many non-clustered ones.

**41. What is the CASE() function?**

The way to implement the *if-then-else* logic in SQL. This function sequentially checks the provided conditions in the WHEN clauses and returns the value from the corresponding THEN clause when the first condition is satisfied. If none of the conditions is satisfied, the function returns the value from the ELSE clause in case it's provided, otherwise, it returns NULL. The syntax is:

CASE

WHEN condition\_1 THEN value\_1

WHEN condition\_2 THEN value\_2

WHEN condition\_3 THEN value\_3

...

ELSE value

END;

**42. What is the difference between the DELETE and TRUNCATE statements?**

DELETE is a reversible DML (Data Manipulation Language) command used to delete one or more rows from a table based on the conditions specified in the WHERE clause. Instead, TRUNCATE is an irreversible DDL (Data Definition Language) command used to delete all rows from a table. DELETE works slower than TRUNCATE. Also, we can't use the TRUNCATE statement for a table containing a foreign key.

**43. What is the difference between the DROP and TRUNCATE statements?**

DROP deletes a table from the database completely, including the table structure and all the associated constraints, relationships with other tables, and access privileges. TRUNCATE deletes all rows from a table without affecting the table structure and constraints. DROP works slower than TRUNCATE. Both are irreversible DDL (Data Definition Language) commands.

**44. What is the difference between the HAVING and WHERE statements?**

The first one works on aggregated data after they are grouped, while the second one checks each row individually. If both statements are present in a query, they appear in the following order: WHERE – GROUP BY – HAVING. The SQL engine interprets them also in the same order.

**45. How do you select all even or all odd records in a table?**

By checking the remainder of the division by 2. In some SQL versions (e.g., PostgreSQL and My SQL), we use the MOD function, in the others (Microsoft SQL Server and SQLite) – the modulo operator (%). To select all even records using MOD:

SELECT \* FROM table\_name

WHERE MOD(ID\_column, 2) = 0;

To select all even records using %:

SELECT \* FROM table\_name

WHERE ID\_column % 2 = 0;

To select all odd records, the syntax is identical in both cases, only that we would use the inequality operator <> instead of =.

**46. How do you find the nth highest value in a column of a table?**

Using the OFFSET clause. For example, to find the 6th highest value from a column, we would use the following syntax:

SELECT \* FROM table\_name

ORDER BY column\_name DESC

LIMIT 1

OFFSET 5;

**47. How do you find the last id in a table?**

Using the MAX() function. Otherwise, in many SQL versions, we can use the following syntax:

SELECT id

FROM table\_name

ORDER BY id DESC

LIMIT 1;

or in Microsoft SQL Server:

SELECT TOP 1 id

FROM table\_name

ORDER BY id DESC

**48. How to select random rows from a table?**

Using the RAND() function in combination with ORDER BY and LIMIT. In some SQL flavors, such as PostgreSQL, it's called RANDOM(). For example, the following code will return five random rows from a table in MySQL:

SELECT \* FROM table\_name

ORDER BY RAND()

LIMIT 5;