**SQL Interview Questions**

(Mohan\_oon)

What is a primary key?

What is a foreign key?

What is a unique key?

What is the difference between a primary key and a unique key?

What is Normalization in a Database?

What are the disadvantages of not performing database Normalization?

What is Denormalization in a Database?

What is a view in SQL?

What is an Index in SQL?

What are the different types of indexes in SQL?

What are the different types of joins in SQL?

What is a "TRIGGER" in SQL?

What are the set operators in SQL?

What is the difference between IN and BETWEEN operators?

How to write an SQL query to find students' names start with 'A'?

Write the SQL query to get the third maximum salary of an employee from a table named employees.

What is the ACID property in a database?

How do we use the DISTINCT statement? What is its use?

What is the default ordering of data using the ORDER BY clause? How could it be changed?

What is the difference between the WHERE and HAVING clauses?

How many Aggregate functions are available in SQL?

What is a primary key?

A primary key is a field or the combination of fields that uniquely identify each record in the table. It is one of a special kind of unique key. If the column contains a primary key, it cannot be null or empty. A table can have duplicate columns, but it cannot have more than one primary key. It always stores unique values into a column.

CREATE TABLE Student (

roll\_number INT PRIMARY KEY,

name VARCHAR(45)

);

**What is a foreign key?**

The foreign key is used to link one or more tables together. It is also known as the referencing key. A foreign key is specified as a key that is related to the primary key of another table. It means a foreign key field in one table refers to the primary key field of the other table. It identifies each row of another table uniquely that maintains the referential integrity. The primary key-foreign key relationship is a very crucial relationship as it maintains the ACID properties of the database sometimes. It also prevents actions that would destroy links between the child and parent tables.

CONSTRAINT constraint\_name]

FOREIGN KEY [foreign\_key\_name] (col\_name, ...)

REFERENCES parent\_tbl\_name (col\_name,...)

What is a unique key?

A unique key is a single or combination of fields that ensure all values stores in the column will be unique. It means a column cannot stores duplicate values. This key provides uniqueness for the column or set of columns.

CREATE TABLE table\_name(

col1 datatype,

col2 datatype UNIQUE

);

What is the difference between a primary key and a unique key?

The primary key and unique key both are essential constraints of the SQL. The main difference among them is that the primary key identifies each record in the table. In contrast, the unique key prevents duplicate entries in a column except for a NULL value.

**What is Normalization in a Database?**

There are some rules of database normalization, which is commonly known as Normal From, and they are:

First normal form (1NF)

Each table cell should contain a single value.

Each record needs to be unique.

Second normal form (2NF)

Rule 1 - Be in 1NF

Rule 2 - All non-key attributes are fully functional dependent on the primary key. (foreign key, uniq rows)

Third normal form (3NF)

Rule 1 - Be in 2NF

Rule 2 - Has no transitive functional dependencies This is Transitive Dependency. When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

What are the disadvantages of not performing database Normalization?

The major disadvantages are:

The occurrence of redundant terms in the database causes the waste of space in the disk. Due to redundant terms, inconsistency may also occur. If any change is made in the data of one table but not made in the same data of another table, then inconsistency will occur. This inconsistency will lead to the maintenance problem and effects the ACID properties as well.

**What is Denormalization in a Database?**

Denormalization is a technique used by database administrators to optimize the efficiency of their database infrastructure. The denormalization concept is based on Normalization, which is defined as arranging a database into tables correctly for a particular purpose. This method allows us to add redundant data into a normalized database to alleviate issues with database queries that merge data from several tables into a single table. It adds redundant terms into the tables to avoid complex joins and many other complex operations.

Denormalization doesn't mean that normalization will not be done. It is an optimization strategy that takes place after the normalization process.

**What is a view in SQL?**

A view is a database object that has no values. It is a virtual table that contains a subset of data within a table. It looks like an actual table containing rows and columns, but it takes less space because it is not present physically. It is operated similarly to the base table but does not contain any data of its own. Its name is always unique. A view can have data from one or more tables. If any changes occur in the underlying table, the same changes reflected in the views also.

The primary use of a view is to implement the security mechanism. It is the searchable object where we can use a query to search the view as we use for the table. It only shows the data returned by the query that was declared when the view was created.

CREATE VIEW view\_name AS

SELECT column\_lists FROM table\_name

WHERE condition;

What is an Index in SQL?

An index is a disc structure associated with a table or view that speeds up row retrieval. It reduces the cost of the query because the query's high cost will lead to a fall in its performance. It is used to increase the performance and allow faster retrieval of records from the table. Indexing reduces the number of data pages we need to visit to find a particular data page. It also has a unique value meaning that the index cannot be duplicated. An index creates an entry for each value which makes it faster to retrieve data.

**What are the different types of indexes in SQL?**

Unique Index. UNIQUE INDEX is used to enforce the uniqueness of values in single or multiple columns. We can create more than one unique index in a single table. For creating a unique index, the user has to check the data in the column because the unique indexes are used when any column of the table has unique values. This indexing does not allow the field to have duplicate values if the column is unique indexed. A unique index can be applied automatically when a primary key is defined.

CREATE UNIQUE INDEX index\_name

ON table\_name (index\_column1, index\_column2,...);

Clustered Index. A clustered index is actually a table where the data for the rows are stored. It determines the order of the table data based on the key values that can sort in only one direction. Each table can have only one clustered index. It is the only index, which has been automatically created when the primary key is generated. If many data modifications needed to be done in the table, then clustered indexes are preferred.

Non-Clustered Index. The indexes other than PRIMARY indexes (clustered indexes) are called non-clustered indexes. We know that clustered indexes are created automatically when primary keys are generated, and non-clustered indexes are created when multiple joins conditions and various filters are used in the query. The non-clustered index and table data are both stored in different places. It cannot be able to alter the physical order of the table and maintains the logical order of data. The non-clustered indexing improves the performance of the queries which use keys without assigning the primary key.

**What are the different types of joins in SQL?**

INNER JOIN Inner join returns only those records from the tables that match the specified condition and hides other rows and columns. In simple words, it fetches rows when there is at least one match of rows between the tables is found.

SELECT column\_lists

FROM table1

INNER JOIN table2 ON join\_condition1

INNER JOIN table3 ON join\_condition2

...;

SELF JOIN A SELF JOIN is used to join a table with itself. This join can be performed using table aliases, which allow us to avoid repeating the same table name in a single sentence. A SELF JOIN is required when we want to combine data with other data in the same table itself.

SELECT column\_lists

FROM table1 AS T1, table1 AS T2

WHERE join\_conditions;

OUTER JOIN

LEFT OUTER JOIN The Left Join is used to fetch all rows from the left-hand table and common records between the specified tables. It returns all the rows from the left-hand side table even though there are no matches on the right-hand side table.

RIGHT OUTER JOIN The Right join is used to retrieve all rows from the right-hand table and only those rows from the other table that fulfilled the join condition.

SELECT colum\_lists

FROM table1

RIGHT JOIN table2

ON join\_condition;

FULL OUTER JOIN The Full Join results from a combination of both left and right join that contains all the records from both tables. It fetches rows when there are matching rows in any one of the tables. This means it returns all the rows from the left-hand side table and all the rows from the right-hand side tables. If a match is not found, it puts NULL value. It is also known as FULL OUTER JOIN.

SELECT \* FROM table1

FULL OUTER JOIN table2

ON join\_condition;

CROSS JOIN

**What is a "TRIGGER" in SQL?**

A trigger is a set of SQL statements that reside in a system catalog. It is a special type of stored procedure that is invoked automatically in response to an event. It allows us to execute a batch of code when an insert, update or delete command is run against a specific table because the trigger is the set of activated actions whenever DML commands are given to the system.

SQL triggers have two main components one is action, and another is an event. When certain actions are taken, an event occurs as a result of those actions.

**What are the set operators in SQL?**

We use the set operators to merge data from one or more tables of the same kind. Although the set operators are like SQL joins, there is a significant distinction. SQL joins combine columns from separate tables, whereas SQL set operators combine rows from different queries. SQL queries that contain set operations are called compound queries. The set operators in SQL are categories into four different types:

UNION: It combines two or more results from multiple SELECT queries into a single result set.

SELECT columns FROM table1

UNION

SELECT columns FROM table2;

UNION ALL: This operator is similar to the Union operator, but it does not remove the duplicate rows from the output of the SELECT statements.

SELECT columns FROM table1

UNION ALL

SELECT columns FROM table2;

INTERSECT: This operator returns the common records from two or more SELECT statements. It always retrieves unique records and arranges them in ascending order by default. Here, the number of columns and data types should be the same.

SELECT columns FROM table1

INTERSECT

SELECT columns FROM table2;

MINUS: This operator returns the records from the first query, which is not found in the second query. It does not return duplicate values.

SELECT columns FROM table1

MINUS

SELECT columns FROM table2;

**What is the difference between IN and BETWEEN operators?**

BETWEEN This operator is used to selects the range of data between two values. The values can be numbers, text, and dates as well.

IN It is a logical operator to determine whether or not a specific value exists within a set of values. This operator reduces the use of multiple OR conditions with the query.

How to write an SQL query to find students' names start with 'A'?

SELECT \* FROM student WHERE stud\_name like 'A%';

Write the SQL query to get the third maximum salary of an employee from a table named employees.

SELECT \* FROM `employees` ORDER BY `salary` DESC LIMIT 1 OFFSET 2

What is the ACID property in a database?

The ACID properties are meant for the transaction that goes through a different group of tasks. A transaction is a single logical order of data. It provides properties to maintain consistency before and after the transaction in a database. It also ensures that the data transactions are processed reliably in a database system.

The ACID property is an acronym for Atomicity, Consistency, Isolation, and Durability.

Atomicity: It ensures that all statements or operations within the transaction unit must be executed successfully. If one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. Its main features are COMMIT, ROLLBACK, and AUTO-COMMIT.

Consistency: This property ensures that the data must meet all validation rules. In simple words, we can say that the database changes state only when a transaction will be committed successfully. It also protects data from crashes.

Isolation: This property guarantees that the concurrent property of execution in the transaction unit must be operated independently. It also ensures that statements are transparent to each other. The main goal of providing isolation is to control concurrency in a database.

Durability: This property guarantees that once a transaction has been committed, it persists permanently even if the system crashes, power loss, or failed.

**How do we use the DISTINCT statement? What is its use?**

The DISTINCT keyword is used to ensure that the fetched value always has unique values. It does not allow to have duplicate values. The DISTINCT keyword is used with the SELECT statement and retrieves different values from the table's column.

SELECT DISTINCT column\_lists FROM table\_name WHERE [condition];

What is the default ordering of data using the ORDER BY clause? How could it be changed?

The ORDER BY clause is used to sort the table data either in ascending or descending order. By default, it will sort the table in ascending order. If we want to change its default behavior, we need to use the DESC keyword after the column name in the ORDER BY clause.

SELECT expressions FROM tables

WHERE conditions

ORDER BY expression [ASC | DESC];

What is the difference between the WHERE and HAVING clauses?

The main difference is that the WHERE clause is used to filter records before any groupings are established, whereas the HAVING clause is used to filter values from a group

**What is a function in SQL, and why use functions?**

A database object representing a set of SQL statements frequently used for a certain task. A function takes in some input parameters, performs calculations or other manipulations on them, and returns the result. Functions help improve code readability and avoid repetition of the same code snippets.

**What types of SQL functions do you know?**

Aggregate functions – work on multiple, usually grouped records for the provided columns of a table, and return a single value (usually by group).

Scalar functions – work on each individual value and return a single value.

On the other hand, SQL functions can be built-in (defined by the system) or user-defined (created by the user for their specific needs).

**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

**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

**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)

**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

**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.

**What is the default data ordering with the ORDER BY statement, and how do you change it?**

By default, the order is ascending. To change it to descending, we need to add the DESC keyword as follows:

SELECT \* FROM table\_name

ORDER BY col\_1 DESC;

POWERED BY DATACAMP WORKSPACE

**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

**What operator is used in the query for pattern matching?**

The LIKE operator in combination with the % and \_ wildcards. The % wildcard represents any number of characters including zero, while \_ – strictly one character.

**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 identifies uniquely each record of the table, and the second one prevents duplicates in that column.

**What is a composite primary key?**

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

**What is the order of appearance of the common statements in the SELECT query?**

SELECT – FROM – JOIN – ON – WHERE – GROUP BY – HAVING – ORDER BY – LIMIT

**In which order the interpreter executes the common statements in the SELECT query?**

FROM – JOIN – ON – WHERE – GROUP BY – HAVING – SELECT – ORDER BY – LIMIT

**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.

**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.

**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.

**What types of SQL relationships do you know?**

One-to-one – each record in one table corresponds to only one record in another table

One-to-many – each record in one table corresponds to several records in another table

Many-to-many – each record in both tables corresponds to several records in another table

**What are the possible values of a BOOLEAN data field?**

In some SQL flavors, such as PostgreSQL, the BOOLEAN data type exists explicitly and takes values TRUE, FALSE, or NULL. In other flavors, such as Microsoft SQL Server, the BIT datatype is used to store Boolean values as integers 1 (true) or 0 (false).

**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.

**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.

**What is the difference between renaming a column and giving an alias to it?**

Renaming a column means permanently changing its actual name in the original table. Giving an alias to a column means giving it a temporary name while executing an SQL query, with the purpose to make the code more readable and compact.

**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.

**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 doesn't match the physical order of the actual data on the disk. A table can have only one clustered index but many non-clustered ones.

**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;

POWERED BY DATACAMP WORKSPACE

**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.

**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.

**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.

**How do you add a record to a table?**

Using the INSERT INTO statement in combination with VALUES. The syntax is:

INSERT INTO table\_name

VALUES (value\_1, value\_2, ...);

POWERED BY DATACAMP WORKSPACE

**How to delete a record from a table?**

Using the DELETE statement. The syntax is:

DELETE FROM table\_name

WHERE condition;

POWERED BY DATACAMP WORKSPACE

In this way, we can also delete multiple records if they satisfy the provided condition.

**How to add a column to a table?**

Using the ALTER TABLE statement in combination with ADD. The syntax is:

ALTER TABLE table\_name

ADD column\_name datatype;

POWERED BY DATACAMP WORKSPACE

**How to rename a column of a table?**

Using the ALTER TABLE statement in combination with RENAME COLUMN ... TO ... The syntax is:

ALTER TABLE table\_name

RENAME COLUMN old\_column\_name TO new\_column\_name;

POWERED BY DATACAMP WORKSPACE

**How to delete a column from a table?**

Using the ALTER TABLE statement in combination with DROP COLUMN. The syntax is:

ALTER TABLE table\_name

DROP COLUMN column\_name;

POWERED BY DATACAMP WORKSPACE

**How to 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;

POWERED BY DATACAMP WORKSPACE

To select all even records using %:

SELECT \* FROM table\_name

WHERE ID\_column % 2 = 0;

POWERED BY DATACAMP WORKSPACE

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

**How to prevent duplicate records when making a query?**

Using the DISTINCT statement in combination with SELECT or creating a unique key for that table.

**How to insert many rows in a table?**

Using the INSERT INTO statement in combination with VALUES. The syntax is:

INSERT INTO table\_name

VALUES (value\_1, value\_2, ...),

(value\_3, value\_4, ...),

(value\_5, value\_6, ...),

...;

POWERED BY DATACAMP WORKSPACE

**How to 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;

POWERED BY DATACAMP WORKSPACE

**How to find the values in a text column of a table that start with a certain letter?**

Using the LIKE operator in combination with the % and \_ wildcards. For example, we need to find all surnames in a table that start with "A". The query is:

SELECT \* FROM table\_name

WHERE surname LIKE 'A\_';

POWERED BY DATACAMP WORKSPACE

Here, we assume that a surname must contain at least two letters. Without this assumption (meaning that a surname can be just A), the query is as follows:

SELECT \* FROM table\_name

WHERE surname LIKE 'A%';

POWERED BY DATACAMP WORKSPACE

**How to 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;

POWERED BY DATACAMP WORKSPACE

or in Microsoft SQL Server:

SELECT TOP 1 id

FROM table\_name

ORDER BY id DESC

POWERED BY DATACAMP WORKSPACE

**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;