SQL Interview Questions and Answers

The following are the most popular and useful SQL interview questions and answers for fresher and experienced candidates. These questions are created specifically to familiarise you with the types of questions you might encounter during your SQL interview. According to our experiences, good interviewers rarely plan to ask any specific topic during the interview. Instead, questioning usually begins with a basic understanding of the subject, and based on your responses, further discussion happened.

1) What is SQL?

SQL stands for the Structured Query Language. It is the standard language used to maintain the relational database and perform many different data manipulation operations on the data. SQL was initially invented in 1970. It is a database language used for database creation, deletion, fetching and modifying rows, etc. sometimes, it is pronounced as 'sequel.' We can also use it to handle organized data comprised of entities (variables) and relations between different entities of the data.

---

2) When SQL appeared?

SQL first appeared in 1974. It is one of the most used languages for maintaining the relational database. In 1986, SQL became the standard of the American National Standards Institute (ANSI) and ISO (International Organization for Standardization) in 1987.

---

3) What are the usages of SQL?

SQL is responsible for maintaining the relational data and the data structures present in the database. Some of the common usages are given below:

- o To execute queries against a database

- o To retrieve data from a database

- o To inserts records in a database

- o To updates records in a database

- o To delete records from a database

- o To create new databases

- o To create new tables in a database

- o To create views in a database

- o To perform complex operations on the database.

---

4) Does SQL support programming language features?

SQL refers to the Standard Query Language. Therefore, it is true that SQL is a language but does not actually support the programming language. It is a common language that doesn't have a loop, conditional statements, and logical operations. It cannot be used for anything other than data manipulation. It is a command language to perform database operations. The primary purpose of SQL is to retrieve, manipulate, update, delete, and perform complex operations like joins on the data present in the database.

---

5) What are the subsets of SQL?

The following are the four significant subsets of the SQL:

- o **Data definition language (DDL):** It defines the data structure that consists of commands like CREATE, ALTER, DROP, etc.

- o **Data manipulation language (DML):** It is used to manipulate existing data in the database. The commands in this category are SELECT, UPDATE, INSERT, etc.

- Data control language (DCL): It controls access to the data stored in the database. The commands in this category include GRANT and REVOKE.

- Transaction Control Language (TCL): It is used to deal with the transaction operations in the database. The commands in this category are COMMIT, ROLLBACK, SET TRANSACTION, SAVEPOINT, etc.

---

6) What is the purpose of DDL Language?

DDL stands for Data definition language. It is the subset of a database that defines the data structure of the database when the database is created. **For example,** we can use the DDL commands to add, remove, or modify tables. It consists of the following commands: CREATE, ALTER and DELETE database objects such as schema, tables, indexes, view, sequence, etc.

Example

1. CREATE TABLE Students

2. (

3. Roll_no INT,

4. Name VARCHAR(45),

5. Branch VARCHAR(30),

6.);

---

7) What is the purpose of DML Language?

Data manipulation language makes the user able to retrieve and manipulate data in a relational database. The DML commands can only perform read-only operations on data. We can perform the following operations using DDL language:

- Insert data into the database through the INSERT command.

- Retrieve data from the database through the SELECT command.

- Update data in the database through the UPDATE command.

- Delete data from the database through the DELETE command.

Example

1. INSERT INTO Student VALUES (111, 'George', 'Computer Science')

---

8) What is the purpose of DCL Language?

Data control language allows users to control access and permission management to the database. It is the subset of a database, which decides that what part of the database should be accessed by which user at what point of time. It includes two commands, GRANT and REVOKE.

GRANT: It enables system administrators to assign privileges and roles to the specific user accounts to perform specific tasks on the database.

REVOKE: It enables system administrators to revoke privileges and roles from the user accounts so that they cannot use the previously assigned permission on the database.

Example

1. GRANT * ON mydb.Student TO Tpoint Tech@localhsot;

---

9) What are tables and fields in the database?

A table is a set of organized data in the form of rows and columns. It enables users to store and display records in the structure format. It is similar to worksheets in the spreadsheet application. Here rows refer to the tuples, representing the simple data item, and columns are the attribute of the data items present in a particular row. Columns can categorize as vertical, and Rows are horizontal.

Fields are the components to provide the structure for the table. It stores the same category of data in the same data type. A table contains a fixed number of columns but can have any number of rows known as the record. It is also called a column in the table of the database. It represents the attribute or characteristics of the entity in the record.

Example

Table: Student

Field: Stud_rollno, Stud_name, Date of Birth, Branch, etc.

---

10) 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. **For example,** the ROLL Number can be treated as the primary key for a student in the university or college.

roll_number	name
101	Peter
102	John
103	Andrew
104	Stevan
105	David

We can define a primary key into a student table as follows:

1. CREATE TABLE Student (

2.   roll_number INT PRIMARY KEY,

3.   name VARCHAR(45),

4.);

---

11) 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.

We can define a foreign key into a table as follows:

1. CONSTRAINT constraint_name]

2.   FOREIGN KEY [foreign_key_name] (col_name, ...)

3.   REFERENCES parent_tbl_name (col_name,...)

---

12) 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. **For example,** the email addresses and roll numbers of student's tables should be unique. It can accept a null value but only one null value per column. It ensures the integrity of the column or group of columns to store different values into a table.

We can define a foreign key into a table as follows:

1. CREATE TABLE table_name(

2.   col1 datatype,

3.   col2 datatype UNIQUE,

4.   ...

5.);

13) 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. The following comparison chart explains it more clearly:

Primary Key	Unique Key
The primary key act as a unique identifier for each record in the table.	The unique key is also a unique identifier for records when the primary key is not present in the table.
We cannot store NULL values in the primary key column.	We can store NULL value in the unique key column, but only one NULL is allowed.

We cannot change or delete the primary key column values.	We can modify the unique key column values.

14) What is a Database?

A database is an organized collection of data that is structured into tables, rows, columns, and indexes. It helps the user to find the relevant information frequently. It is an electronic system that makes data access, data manipulation, data retrieval, data storing, and data management very easy. Almost every organization uses the database for storing the data due to its easily accessible and high operational ease. The database provides perfect access to data and lets us perform required tasks.

The following are the common features of a database:

- Manages large amounts of data
- Accurate
- Easy to update
- Security
- Data integrity
- Easy to research data

15) What is meant by DBMS?

DBMS stands for Database Management System. It is a software program that primarily functions as an interface between the database and the end-user. It provides us the power such as managing the data, the database engine, and the database schema to facilitate the organization and manipulation of data using a simple query in almost no time. It is like a File Manager that manages data in a database rather than saving it in file systems. Without the database management system, it would be far more difficult for the user to access the database's data.

The following are the components of a DBMS:

- Software
- Data

- Procedures

- Database Languages

- Query Processor

- Database Manager

- Database Engine

- Reporting

---

16) What are the different types of database management systems?

The database management systems can be categorized into several types. Some of the important lists are given below:

- Hierarchical databases (DBMS)

- Network databases (IDMS)

- Relational databases (RDBMS

- Object-oriented databases

- Document databases (Document DB)

- Graph databases

- ER model databases

- NoSQL databases

---

17) What is RDBMS?

RDBMS stands for Relational Database Management System. It is a database management system based on a relational model. It facilitates you to manipulate the data stored in the tables by using relational operators. RDBMS stores the data into the collection of tables and links those tables using the relational operators easily whenever required. Examples of relational database management systems are Microsoft Access, MySQL, SQL Server, Oracle database, etc.

---

18) What is Normalization in a Database?

Normalization is used to minimize redundancy and dependency by organizing fields and table of a database.

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

- o First normal form(1NF)

- o Second normal form(2NF)

- o Third normal form(3NF)

- o Boyce-Codd normal form(BCNF)

Using these steps, the redundancy, anomalies, inconsistency of the data in the database can be removed.

---

19) What is the primary use of Normalization?

Normalization is mainly used to add, delete or modify a field that can be made in a single table. The primary use of Normalization is to remove redundancy and remove the insert, delete and update distractions. Normalization breaks the table into small partitions and then links them using different relationships to avoid the chances of redundancy.

---

20) 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.

---

21) What is an inconsistent dependency?

An Inconsistent dependency refers to the difficulty of getting relevant data due to a missing or broken path to the data. It leads users to search the data in the wrong table, resulting in an error as an output.

---

22) 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.

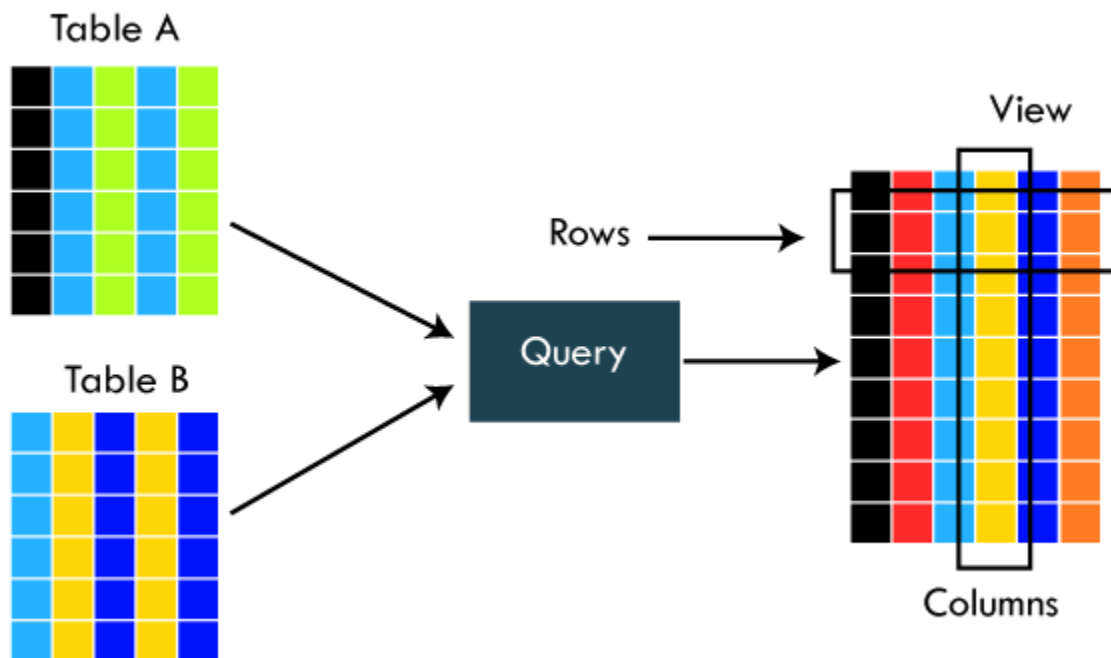---

23) What are the different types of SQL operators?

Operators are the special keywords or special characters reserved for performing particular operations. They are also used in SQL queries. We can primarily use these operators within the WHERE clause of SQL commands. It's a part of the command to filters data based on the specified condition. The SQL operators can be categorized into the following types:

- **Arithmetic operators:** These operators are used to perform mathematical operations on numerical data. The categories of this operators are addition (+), subtraction (-), multiplication (*), division (/), remainder/modulus (%), etc.

- **Logical operators:** These operators evaluate the expressions and return their results in True or False. This operator includes ALL, AND, ANY, ISNULL, EXISTS, BETWEEN, IN, LIKE, NOT, OR, UNIQUE.

- **Comparison operators:** These operators are used to perform comparisons of two values and check whether they are the same or not. It includes equal to (=), not equal to (!= or <>), less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=), not less than (!<), not greater than (!>), etc.

- **Bitwise operators:** It is used to do bit manipulations between two expressions of integer type. It first performs conversion of integers into binary bits and then applied operators such as AND (& symbol), OR (|, ^), NOT (~), etc.

- **Compound operators:** These operators perform operations on a variable before setting the variable's result to the operation's result. It includes Add equals (+=), subtract equals (-=), multiply equals (*=), divide equals (/=), modulo equals (%=), etc.

- **String operators:** These operators are primarily used to perform concatenation and pattern matching of strings. It includes + (String concatenation), += (String concatenation assignment), % (Wildcard), [] (Character(s) matches), [^] (Character(s) not to match), _ (Wildcard match one character), etc.

---

24) 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.

We can create a view by using the following syntax:

1. CREATE VIEW view_name AS

2. SELECT column_lists FROM table_name

3. WHERE condition;

---

25) 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.

For example: Suppose we have a book which carries the details of the countries. If you want to find out information about India, why will you go through every page of that book? You could directly go to the index. Then from the index, you can go to that particular page where all the information about India is given.

---

26) What are the different types of indexes in SQL?

SQL indexes are nothing more than a technique of minimizing the query's cost. The higher the query's cost, the worse the query's performance. The following are the different types of Indexes supported in SQL:

- o   Unique Index

- o   Clustered Index

- o   Non-Clustered Index

- o   Bit-Map Index

- o   Normal Index

- o   Composite Index

- o   B-Tree Index

- o   Function-Based Index

27) What is the 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.

We can create it by using the following syntax:

1. CREATE UNIQUE INDEX index_name

2. ON table_name (index_column1, index_column2,...);

Example

1. CREATE TABLE Employee(

2.   ID int AUTO_INCREMENT PRIMARY KEY,

3.   Name varchar(45),

4.   Phone varchar(15),

5.   City varchar(25),

6.);

Suppose we want to make a Phone column as a unique index. We can do this like below:

1. CREATE UNIQUE INDEX index_name_phone ON Employee (Phone);

28) What is clustered index in SQL?

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.

29) What is the non-clustered index in SQL?

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 purpose of creating a non-clustered index is for searching the data. Its best example is a book where the content is written in one place, and the index is at a different place. We can create 0 to 249 non-clustered indexes in each table. The non-clustered indexing improves the performance of the queries which use keys without assigning the primary key.

---

30) What are the differences between SQL, MySQL, and SQL Server?

The following comparison chart explains their main differences:

SQL	MySQL	SQL Server
SQL or Structured Query Language is useful for managing our relational databases. It is used to query and operate the database.	MySQL is the popular database management system used for managing the relational database. It is a fast, scalable, and easy-to-use database.	SQL Server is an RDBMS database system mainly developed for the Windows system to store, retrieve, and access data requested by the developer.
SQL first appeared in 1974.	MySQL first appeared on May 23, 1995.	SQL Server first appeared on April 24, 1989.
SQL was developed by IBM Corporation.	MySQL was developed by Oracle Corporation.	SQL Server was developed by Microsoft Company.
SQL is a query language for managing databases.	MySQL is database software that uses SQL language to conduct with the database.	SQL Server is also a software that uses SQL language to conduct with the database.

SQL has no variables.	MySQL can use variables constraints and data types.	SQL Server can use variables constraints and data types.
SQL is a programming language, so that it does not get any updates. Its commands are always fixed and remain the same.	MySQL is software, so it gets frequent updation.	SQL Server is also software, so it gets frequent updation.

31) What is the difference between SQL and PL/SQL?

The following comparison chart explains their main differences:

SQL	PL/SQL
SQL is a database structured query language used to communicate with relational databases. It was developed by IBM Corporations and first appeared in 1974.	PL/SQL or Procedural Language/Structured Query Language is a dialect of SQL used to enhance the capabilities of SQL. Oracle Corporation developed it in the early 90's. It uses SQL as its database language.
SQL is a declarative and data-oriented language.	PL/SQL is a procedural and application-oriented language.
SQL has no variables.	PL/SQL can use variables constraints and data types.
SQL can execute only a single query at a time.	PL/SQL can execute a whole block of code at once.

SQL query can be embedded in PL/SQL.	PL/SQL cannot be embedded in SQL as SQL does not support any programming language and keywords.
SQL can directly interact with the database server.	PL/SQL cannot directly interact with the database server.
SQL is like the source of data that we need to display.	PL/SQL provides a platform where SQL data will be shown.

32) Is it possible to sort a column using a column alias?

Yes. We can use the alias method in the ORDER BY instead of the WHERE clause for sorting a column.

33) What is the difference between clustered and non-clustered indexes in SQL?

Indexing is a method to get the requested data very fast. There are mainly two types of indexes in SQL, clustered index and non-clustered index. The differences between these two indexes are very important from an SQL performance perspective. The following comparison chart explains their main differences:

Clustered Index	Non-Clustered Index
A clustered index is a table or view where the data for the rows are stored. In a relational database, if the table column contains a primary key, MySQL automatically creates a clustered index named PRIMARY.	The indexes other than PRIMARY indexes (clustered indexes) are called non-clustered indexes. It has a structure separate from the data row. The non-clustered indexes are also known as secondary indexes.

Clustered indexes store the data information and the data itself.	Non-clustered indexes stores only the information, and then it will refer you to the data stored in clustered data.
There can only be one clustered index per table.	There can be one or more non-clustered indexes in a table.
A clustered index determines how data is stored physically in the table. Therefore, reading from a clustered index is faster.	It creates a logical ordering of data rows and uses pointers for accessing the physical data files. Therefore, reading from a clustered index is slower.
A clustered index always contains an index id of 0.	A non-clustered index always contains an index id>0.

34) What is the SQL query to display the current date?

There is a built-in function in SQL called GetDate(), which is used to return the current timestamp.

35) Which are joins in SQL? Name the most commonly used SQL joins?

SQL joins are used to retrieve data from multiple tables into a meaningful result set. It is performed whenever you need to fetch records from two or more tables. They are used with SELECT statement and join conditions.
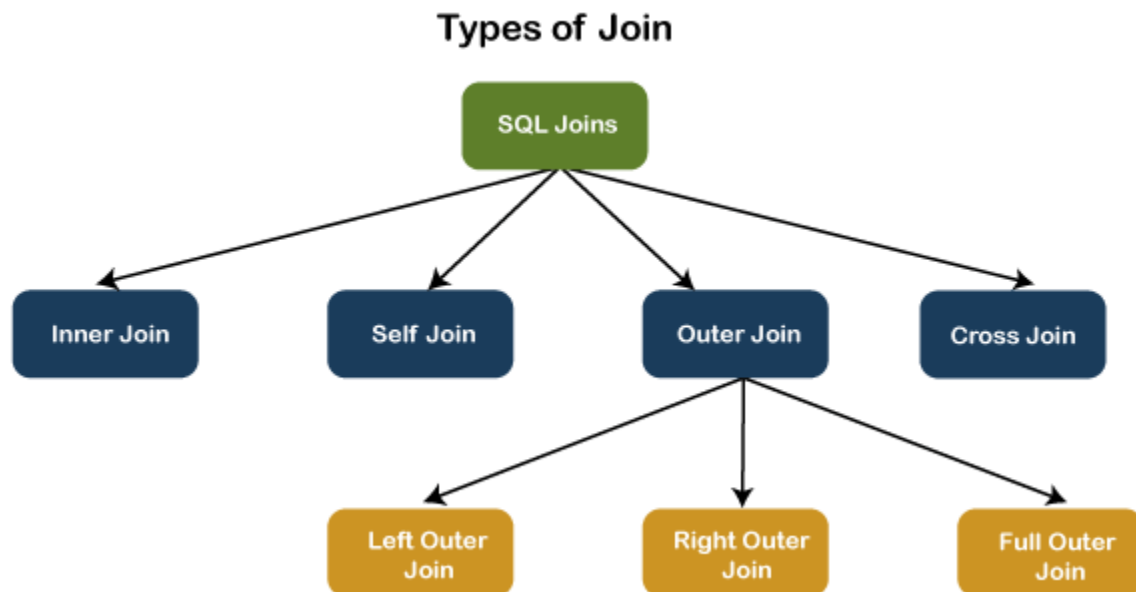
The following are the most commonly used joins in SQL:

- o INNER JOIN
- o LEFT OUTER JOIN
- o RIGHT OUTER JOIN

36) What are the different types of joins in SQL?

Joins are used to merge two tables or retrieve data from tables. It depends on the relationship between tables. According to the ANSI standard, the following are the different types of joins used in SQL:

- o INNER JOIN

- o SELF JOIN

- o LEFT OUTER JOIN

- o RIGHT OUTER JOIN
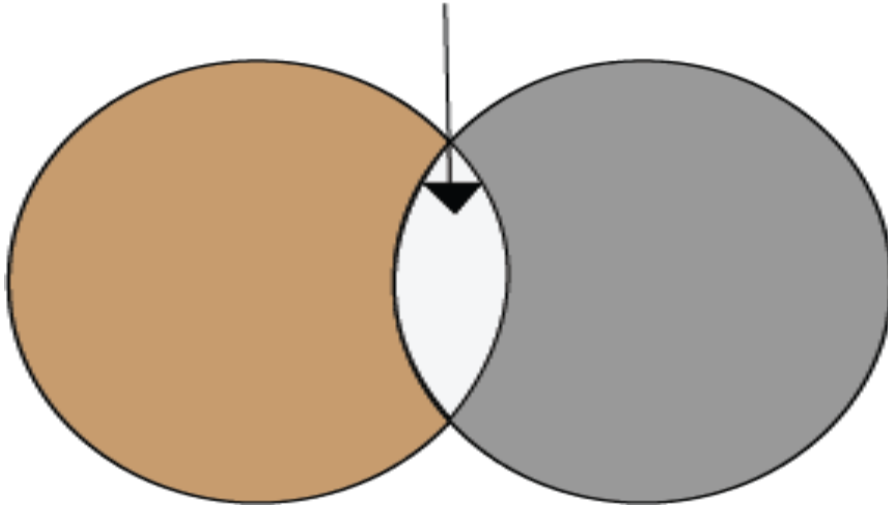
- o FULL OUTER JOIN

- o CROSS JOIN

Types of Join

37) What is INNER JOIN in SQL?

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. INNER JOIN keyword joins the matching records from two tables. It is assumed as a default join, so it is optional to use the INNER keyword with the query.

The below visual representation explain this join more clearly:
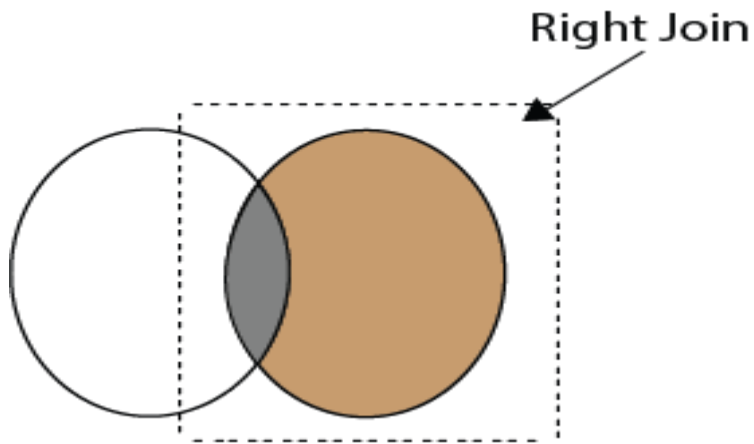
Inner Join

The following syntax illustrates the INNER JOIN:

1. SELECT column_lists

2. FROM table1

3. INNER JOIN table2 ON join_condition1

4. INNER JOIN table3 ON join_condition2

5. ...;

---

38) What is the Right JOIN in SQL?

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. It returns all the rows from the right-hand side table even though there are no matches in the left-hand side table. If it finds unmatched records from the left side table, it returns a Null value. This join is also known as Right Outer Join.

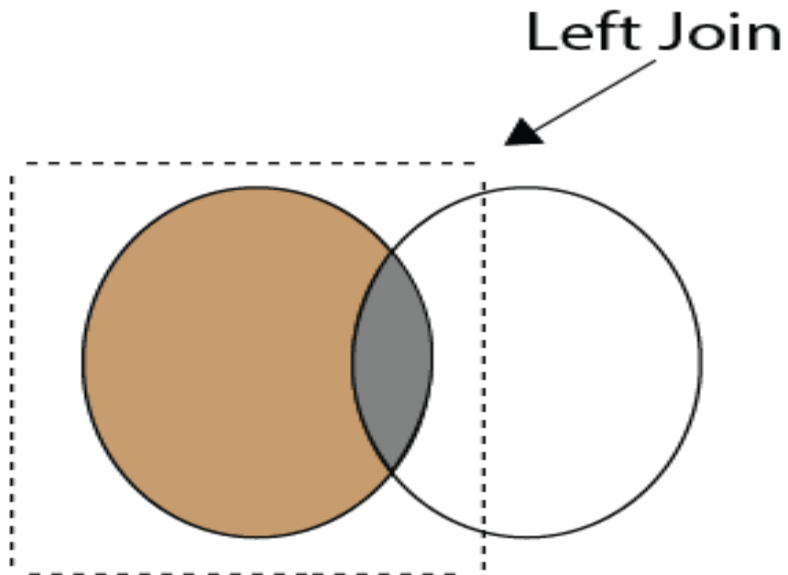The below visual representation explain this join more clearly:

The following syntax illustrates the RIGHT JOIN:

1. SELECT colum_lists

2. FROM table1

3. RIGHT JOIN table2

4. ON join_condition;

---

39) What is Left Join in SQL?

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. If it will not find any matching record from the right side table, then it returns null. This join can also be called a Left Outer Join.

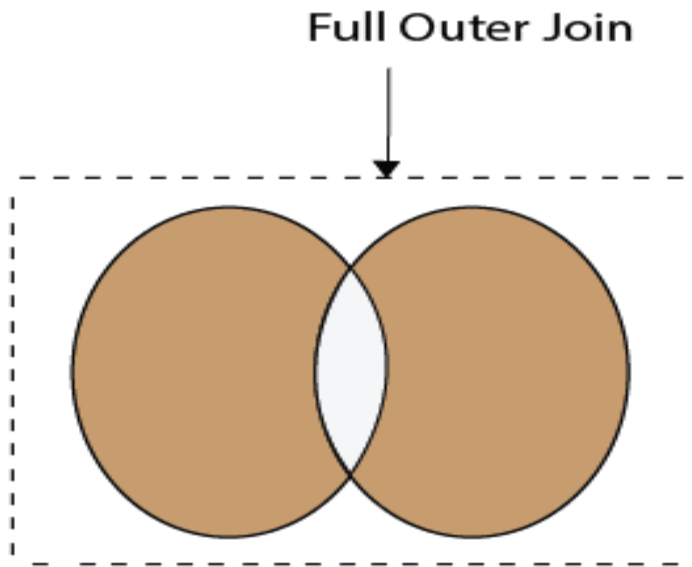The following visual representation explains it more clearly:

Left Join

The following syntax illustrates the RIGHT JOIN:

1. SELECT colum_lists

2. FROM table1

3. LEFT JOIN table2

4. ON join_condition;

---

40) What is Full Join in SQL?

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.

The following visual representation explains it more clearly:

Full Outer Join

The following syntax illustrates the FULL JOIN:

1. SELECT * FROM table1

2. FULL OUTER JOIN table2

3. ON join_condition;

---

41) 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.

We use the CREATE TRIGGER statement for creating a trigger in SQL. Here is the syntax:

1. CREATE TRIGGER trigger_name

2.   (AFTER | BEFORE) (INSERT | UPDATE | DELETE)

3.   ON table_name FOR EACH ROW

4.   BEGIN

5.   --variable declarations

6.   --trigger code

7.   END;

---

42) What is self-join and what is the requirement of 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. It will throw an error if we use the same table name more than once in a single query without using table aliases.

A SELF JOIN is required when we want to combine data with other data in the same table itself. It is often very useful to convert a hierarchical structure to a flat structure.

The following syntax illustrates the SELF JOIN:

1. SELECT column_lists

2. FROM table1 AS T1, table1 AS T2

3. WHERE join_conditions;

Example

Suppose we have a table 'Student' having the following data:

student_id	name	course_id	duration
1	Adam	1	3
2	Peter	2	4
1	Adam	2	4
3	Brian	3	2
2	Shane	3	5

If we want to get retrieve the student_id and name from the table where student_id is equal, and course_id is not equal, it can be done by using the self-join:

1. SELECT s1.student_id, s1.name

2. FROM student AS s1, student s2

3. WHERE s1.student_id=s2.student_id

4. AND s1.course_id<>s2.course_id;

Here is the result:

student_id	name
1	Adam
2	Shane
1	Adam
2	Peter

---

43) 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:

A. UNION: It combines two or more results from multiple SELECT queries into a single result set. It has a default feature to remove the duplicate rows from the tables. The following syntax illustrates the Union operator:

1. SELECT columns FROM table1

2. UNION

3. SELECT columns FROM table2;

B. 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. The following syntax illustrates the UNION ALL operator:

1. SELECT columns FROM table1

2. UNION ALL

3. SELECT columns FROM table2;

C. 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. The following syntax illustrates the INTERSECT operator:

1. SELECT columns FROM table1

2. INTERSECT

3. SELECT columns FROM table2;

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

1. SELECT columns FROM table1

2. MINUS

3. SELECT columns FROM table2;

---

44) What is the difference between IN and BETWEEN operators?

The following comparison chart explains their main differences:

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

It returns records whose column value lies in between the defined range.	It compares the specified column's value and returns the records when the match exists in the set of values.
The following syntax illustrates this operator: SELECT * FROM table_name WHERE column_name BETWEEN 'value1' AND 'value2';	The following syntax illustrates this operator: SELECT * FROM table_name WHERE column_name IN ('value1','value 2');

45) What is a constraint? Tell me about its various levels.

The constraint is used to specify the rule and regulations that allows or restricts what values/data will be stored in the table. It ensures data accuracy and integrity inside the table. It enforces us to store valid data and prevents us from storing irrelevant data. If any interruption occurs between the constraint and data action, the action is failed. Some of the most commonly used constraints are NOT NULL, PRIMARY KEY, FOREIGN KEY, AUTO_INCREMENT, UNIQUE KEY, etc.

The following syntax illustrates us to create a constraint for a table:

1. CREATE TABLE table_name (

2.   column1 datatype constraint,

3.   column2 datatype constraint,

4.   ....

5.);

SQL categories the constraints into two levels:

Column Level Constraints: These constraints are only applied to a single column and limit the type of data that can be stored in that column.
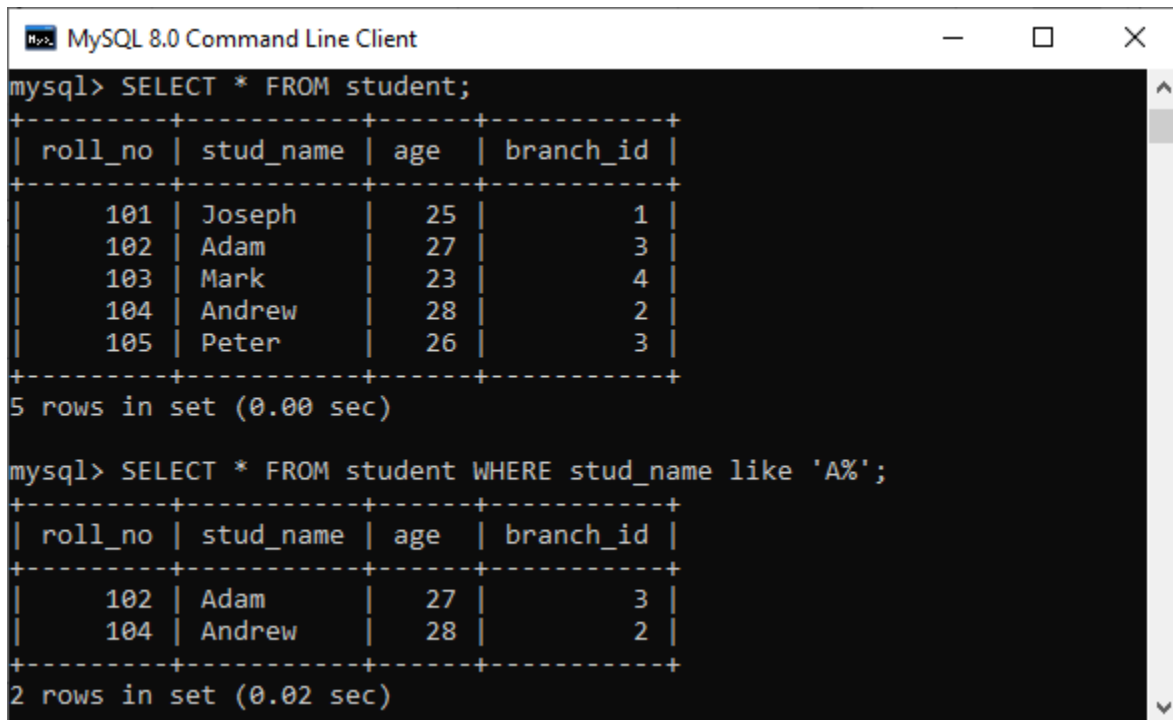
Table Level Constraints: These constraints are applied to the entire table and limit the type of data that can be entered.

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

We can write the following query to get the student details whose name starts with A:

1. SELECT * FROM student WHERE stud_name like 'A%';

Here is the demo example where we have a table named student that contains two names starting with the 'A' character.

```
MySQL 8.0 Command Line Client                                    —    □    ×

mysql> SELECT * FROM student;
+---------+------------+------+-----------+
| roll_no | stud_name  | age  | branch_id |
+---------+------------+------+-----------+
|     101 | Joseph     |   25 |         1 |
|     102 | Adam       |   27 |         3 |
|     103 | Mark       |   23 |         4 |
|     104 | Andrew     |   28 |         2 |
|     105 | Peter      |   26 |         3 |
+---------+------------+------+-----------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM student WHERE stud_name like 'A%';
+---------+------------+------+-----------+
| roll_no | stud_name  | age  | branch_id |
+---------+------------+------+-----------+
|     102 | Adam       |   27 |         3 |
|     104 | Andrew     |   28 |         2 |
+---------+------------+------+-----------+
2 rows in set (0.02 sec)
```

---

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

The following query is the simplest way to get the third maximum salary of an employee:

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

Here is the demo example that shows how to get the third maximum salary of an employee.

```
MySQL 8.0 Command Line Client                                        —   □   ×

mysql> SELECT * FROM employees;
+----+---------------+-------------+--------+
| id | name          | performance | salary |
+----+---------------+-------------+--------+
|  1 | Mary          |         101 |  55000 |
|  2 | John          |         103 |  66950 |
|  3 | Suzi          |         104 |  89250 |
|  4 | Gracia        |         105 | 118800 |
|  5 | Nancy Johnson |         103 |  97850 |
|  6 | Joseph        |         102 |  45450 |
|  7 | Donald        |         103 |  51500 |
|  8 | William       |        NULL |  74825 |
|  9 | Rayan         |        NULL |  94300 |
+----+---------------+-------------+--------+
9 rows in set (0.00 sec)

mysql> SELECT * FROM `employees` ORDER BY `salary` DESC LIMIT 1 OFFSET 2;
+----+-------+-------------+--------+
| id | name  | performance | salary |
+----+-------+-------------+--------+
|  9 | Rayan |        NULL |  94300 |
+----+-------+-------------+--------+
1 row in set (0.00 sec)
```

The following are the alternative way to get the third-highest salary of an employee:

A. Using LIMIT Keyword

1. SELECT salary FROM employees

2. ORDER BY salary DESC

3. LIMIT 2, 1;

B. Using Subquery

1. SELECT salary

2. FROM

3.   (SELECT salary

4.   FROM employees

5.   ORDER BY salary DESC

6.   LIMIT 3) AS Temp

7. ORDER BY salary LIMIT 1;

C. Using TOP Keyword

1. SELECT TOP 1 salary

2. FROM

3. (SELECT DISTINCT TOP 3 salary

4. FROM employees

5. ORDER BY salary DESC) AS Temp

6. ORDER BY salary ASC;

---

48) What is the difference between DELETE and TRUNCATE statements in SQL?

The main difference between them is that the delete statement deletes data without resetting a table's identity, whereas the truncate command resets a particular table's identity. The following comparison chart explains it more clearly:

No.	DELETE	TRUNCATE
1)	The delete statement removes single or multiple rows from an existing table depending on the specified condition.	The truncate command deletes the whole contents of an existing table without the table itself. It preserves the table structure or schema.
2)	DELETE is a **DML command.**	TRUNCATE is a **DML command.**
3)	We **can use the WHERE** clause in the DELETE command.	We **cannot use the WHERE** clause with TRUNCATE.

4)	DELETE statement is used **to delete a row** from a table.	TRUNCATE statement is used **to remove all the rows** from a table.
5)	DELETE is **slower** because it maintained the log.	TRUNCATE statement is **faster** than DELETE statement as it deletes entire data at a time without maintaining transaction logs.
6)	You **can roll back** data after using the DELETE statement.	It is **not possible to roll back** after using the TRUNCATE statement.
7)	DELETE query **takes more space.**	TRUNCATE query **occupies less space.**

---

49) 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.

---

50) Is a blank space or zero the same as a NULL value?

No. The NULL value is not the same as zero or a blank space. The following points explain their main differences:

- A NULL value is a value, which is 'unavailable, unassigned, unknown or not applicable.' It would be used in the absence of any value. We can perform arithmetic operations on it. On the other hand, zero is a number, and a blank space is treated as a character.

- The NULL value can be treated as an unknown and missing value, but zero and blank spaces differ from the NULL value.

- We can compare a blank space or a zero to another blank space or a zero. On the other hand, one NULL may not be the same as another NULL. NULL indicates that no data has been provided or that no data exists.

---

51) What are functions and their usage in SQL?

SQL functions are simple code snippets that are frequently used and re-used in database systems for data processing and manipulation. Functions are the measured values. It always performs a specific task. The following rules should be remembered while creating functions:

- A function should have a name, and the name cannot begin with a special character such as @, $, #, or other similar characters.

- Functions can only work with the SELECT statements.

- Every time a function is called, it compiles.

- Functions must return value or result.

- Functions are always used with input parameters.

SQL categories the functions into two types:

- **User-Defined Function:** Functions created by a user based on their needs are termed user-defined functions.

- **System Defined Function:** Functions whose definition is defined by the system are termed system-defined functions. They are built-in database functions.

SQL functions are used for the following purposes:

- To perform calculations on data

- To modify individual data items

- To manipulate the output

- To format dates and numbers

- To convert data types

---

52) What is meant by case manipulation functions? Explains its different types in SQL.

Case manipulation functions are part of the character functions. It converts the data from the state in which it is already stored in the table to upper, lower, or mixed case. The conversion performed by this function can be used to format the output. We can use it in almost every part of the SQL statement. Case manipulation functions are mostly used when you need to search for data, and you don't have any idea that the data you are looking for is in lower case or upper case.

There are three case manipulation functions in SQL:

LOWER: This function is used to converts a given character into lowercase. The following example will return the 'STEPHEN' as 'stephen':

1. SELECT LOWER ('STEPHEN') AS Case_Reault FROM dual;

NOTE: Here, 'dual' is a dummy table.

UPPER: This function is used to converts a given character into uppercase. The following example will return the 'stephen' as 'STEPHEN':

1. SELECT UPPER ('stephen') AS Case_Reault FROM dual;

INITCAP: This function is used to converts given character values to uppercase for the initials of each word. It means every first letter of the word is converted into uppercase, and the rest is in lower case. The following example will return the 'hello stephen' as 'Hello Stephen':

1. SELECT INITCAP ('hello stephen') AS Case_Reault FROM dual;

---

53) Explain character-manipulation functions? Explains its different types in SQL.

Character-manipulation functions are used to change, extract, and alter the character string. When one or more characters and words are passed into the function, the function will perform its operation on those input strings and return the result.

The following are the character manipulation functions in SQL:

A) CONCAT: This function is used to join two or more values together. It always appends the second string into the end of the first string. For example:

Input: SELECT CONCAT ('Information-', 'technology') FROM DUAL;

Output: Information-technology

B) SUBSTR: It is used to return the portion of the string from a specified start point to an endpoint. For example:

Input: SELECT SUBSTR ('Database Management System', 9, 11) FROM DUAL;

Output: Management

C) LENGTH: This function returns the string's length in numerical value, including the blank spaces. For example:

Input: SELECT LENGTH ('Hello Tpoint Tech') FROM DUAL;

Output: 16

D) INSTR: This function finds the exact numeric position of a specified character or word in a given string. For example:

Input: SELECT INSTR ('Hello Tpoint Tech', 'Tpoint Tech');

Output: 7

E) LPAD: It returns the padding of the left-side character value for right-justified value. For example:

Input: SELECT LPAD ('200', 6,'*');

Output: ***200

F) RPAD: It returns the padding of the right-side character value for left-justified value. For example:

Input: SELECT RPAD ('200', 6,'*');

Output: 200***

G) TRIM: This function is used to remove all the defined characters from the beginning, end, or both. It also trimmed extra spaces. For example:

Input: SELECT TRIM ('A' FROM 'ABCDCBA');

Output: BCDCB

H) REPLACE: This function is used to replace all occurrences of a word or portion of the string (substring) with the other specified string value. For example:

Input: SELECT REPLACE ('It is the best coffee at the famous coffee shop.', 'coffee', 'tea');

Output: It is the best tea at the famous tea shop.

---

54) What is the usage of the NVL() function?

The NVL() function is used to convert the NULL value to the other value. The function returns the value of the second parameter if the first parameter is NULL. If the first parameter is anything other than NULL, it is left unchanged. This function is used in Oracle, not in SQL and MySQL. Instead of NVL() function, MySQL have IFNULL() and SQL Server have ISNULL() function.

---

55) Which function is used to return remainder in a division operator in SQL?

The MOD function returns the remainder in a division operation.

---

56) What are the syntax and use of the COALESCE function?

The COALESCE() function evaluates the arguments in sequence and returns the first NON-NULL value in a specified number of expressions. If it evaluates arguments as NULL or not found any NON-NULL value, it returns the NULL result.
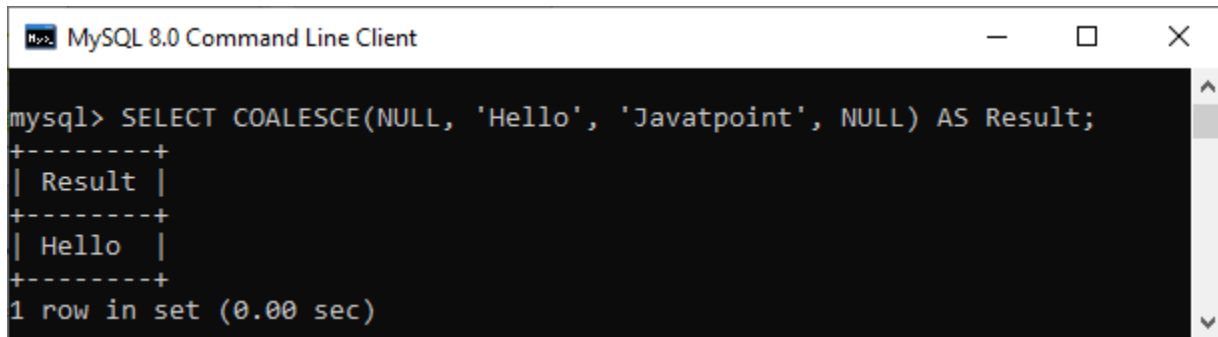
The syntax of COALESCE function is given below:

1. COALESCE (exp1, exp2, ... expn)

Example:

1. SELECT COALESCE(NULL, 'Hello', 'Tpoint Tech', NULL) AS Result;

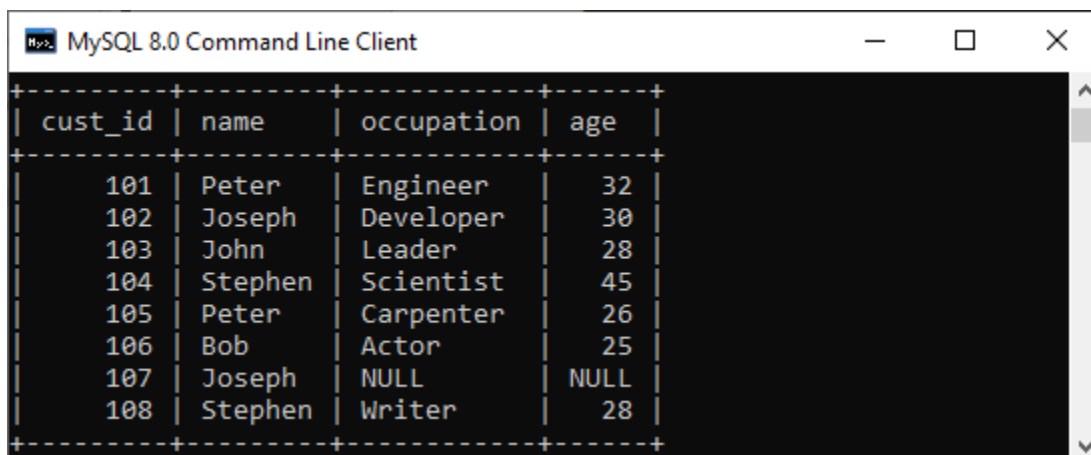This statement will return the following output:

```
MySQL 8.0 Command Line Client                          —    □    ×

mysql> SELECT COALESCE(NULL, 'Hello', 'Javatpoint', NULL) AS Result;
+--------+
| Result |
+--------+
| Hello  |
+--------+
1 row in set (0.00 sec)
```

57) 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. We can use it with the help of the following syntax:

1. SELECT DISTINCT column_lists FROM table_name WHERE [condition];

Suppose we have a table 'customer' containing eight records in which the name column has some duplicate values.

```
MySQL 8.0 Command Line Client                          —    □    ×

+---------+---------+-----------+------+
| cust_id | name    | occupation | age |
+---------+---------+-----------+------+
|     101 | Peter   | Engineer   |  32 |
|     102 | Joseph  | Developer  |  30 |
|     103 | John    | Leader     |  28 |
|     104 | Stephen | Scientist  |  45 |
|     105 | Peter   | Carpenter  |  26 |
|     106 | Bob     | Actor      |  25 |
|     107 | Joseph  | NULL       | NULL |
|     108 | Stephen | Writer     |  28 |
+---------+---------+-----------+------+
```

If we want to get the name column without any duplicate values, the DISTINCT keyword is required. Executing the below command will return a name column with unique values.

---

58) 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.

The syntax to do this is given below:

1. SELECT expressions FROM tables

2. WHERE conditions

3. ORDER BY expression [ASC | DESC];

We have taken a customer table in the previous example. Now, we will demonstrate the ORDER BY clause on them as well.

In the below output, we can see that the first query will sort the table data in ascending order based on the name column. However, if we run the second query by specifying the DESC keyword, the table's order is changed in descending order.

---

59) Is the following query returns the output?

1. SELECT subject_code, AVG (marks)

2. FROM Students

3. WHERE AVG(marks) > 70

4. GROUP BY subject_code;

Answer: No. The above query does not return the output because we cannot use the WHERE clause to restrict the groups. We need to use the HAVING clause instead of the WHERE clause to get the correct output.

---

60) 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. The below comparison chart explains the most common differences:

WHERE	HAVING
This clause is implemented in row operations.	This clause is implemented in column operations.
It does not allow to work with aggregate functions.	It can work with aggregate functions.
This clause can be used with the SELECT, UPDATE, and DELETE statements.	This clause can only be used with the SELECT statement.

61) How many Aggregate functions are available in SQL?

The aggregate function is used to determine and calculate several values in a table and return the result as a single number. For example, the average of all values, the sum of all values, and the maximum and minimum value among particular groupings of values.

The following syntax illustrates how to use aggregate functions:

1. function_name (DISTINCT | ALL expression)

SQL provides seven (7) aggregate functions, which are given below:

- o **AVG():** This function is used to returns the average value from specified columns.

- o **COUNT():** This function is used to returns the number of table rows, including rows with null values.

- o **MAX():** This function is used to returns the largest value among the group.

- o **MIN():** This function is used to returns the smallest value among the group.

o   **SUM():** This function is used to returns the total summed values(non-null) of the specified column.

o   **FIRST()**: This function is used to returns the first value of an expression.

o   **LAST()**: This function is used to returns the last value of an expression.

---

62) What is SQL Injection?

SQL injection is a type of vulnerability in website and web app code that allows attackers to control back-end operations and access, retrieve, and destroy sensitive data from databases. In this technique, malicious SQL statements are inserted into a database entry field, and once they are performed, the database becomes vulnerable to an attacker. This technique is commonly used to access sensitive data and perform administrative activities on databases by exploiting data-driven applications. It is also known as **SQLi attack.**

Some common examples of SQL injection are:

o   Accessing confidential data to modify an SQL query to get desired results.

o   UNION attacks to steal data from different database tables.

o   Examine the database to extract information regarding the version and structure of the database.

---

63) What is the difference between the RANK() and DENSE_RANK() functions?

The **RANK function** determines the rank for each row within your ordered partition in the result set. If the two rows are assigned the same rank, then the next number in the ranking will be its previous rank plus a number of duplicate numbers. For example, if we have three records at rank 4, the next rank listed would be ranked 7.

The **DENSE_RANK** function assigns a unique rank for each row within a partition as per the specified column value without any gaps. It always specifies ranking in consecutive order. If the two rows are assigned the same rank, this function will assign it with the same rank, and the next rank being the next sequential number. For example, if we have 3 records at rank 4, the next rank listed would be ranked 5.

---

64) Is it possible to implicitly insert a row for the identity column?

Yes. We can implicitly insert a row for the identity column. Here is an example of doing this:

1. SET IDENTITY_INSERT TABLE1 ON

2. INSERT INTO demo_table1 (id, name, branch)

3. SELECT id, name, branch FROM demo_table2

4. SET IDENTITY_INSERT OFF

---

65) What are SQL comments?

Comments are explanations or annotations in SQL queries that are readable by programmers. It's used to make SQL statements easier to understand for humans. During the parsing of SQL code, it will be ignored. Comments can be written on a single line or across several lines.

o   **Single Line Comments:** It starts with two consecutive hyphens (--).

o   **Multi-line Comments:** It starts with /* and ends with */.