

# 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:

- To execute queries against a database
- To retrieve data from a database
- To inserts records in a database
- To updates records in a database
- To delete records from a database
- To create new databases
- To create new tables in a database
- To create views in a database
- 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:

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

To read more information, [click here](#).

---

## 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,...)

To read more information, [click here](#).

---

## 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. );

To read more information, [click here](#).

---

### 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.

To read more information, [click here](#).

---

## 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 Form, and they are:

- First normal form(1NF)



- Second normal form(2NF)
- Third normal form(3NF)
- 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:

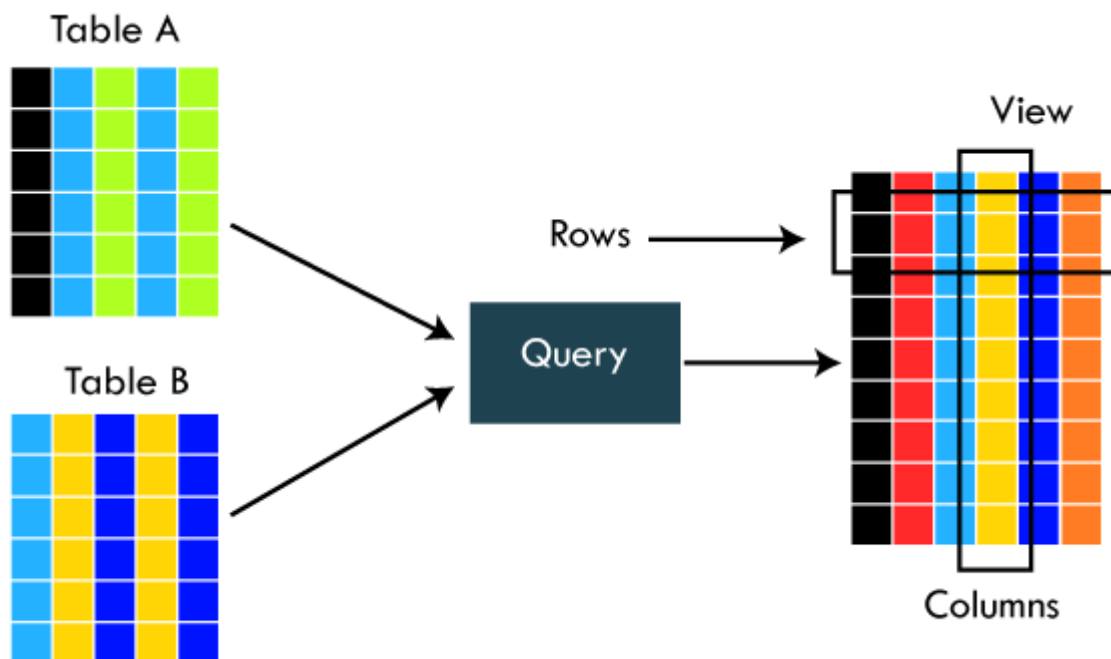
- **Arithmetic operators:** These operators are used to perform mathematical operations on numerical data. The categories of this operators are addition (+), subtraction (-), multiplication (\*), division (/), remainder/modulus (%), etc.
- **Logical operators:** These operators evaluate the expressions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

- Unique Index
  - Clustered Index
  - Non-Clustered Index
  - Bit-Map Index
  - Normal Index
  - Composite Index
  - B-Tree Index
  - 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);

To read more information, [click here](#).

---

## 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.

To read more information, [click here](#).

---

## 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.

To read more information, [click here](#).

### 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:



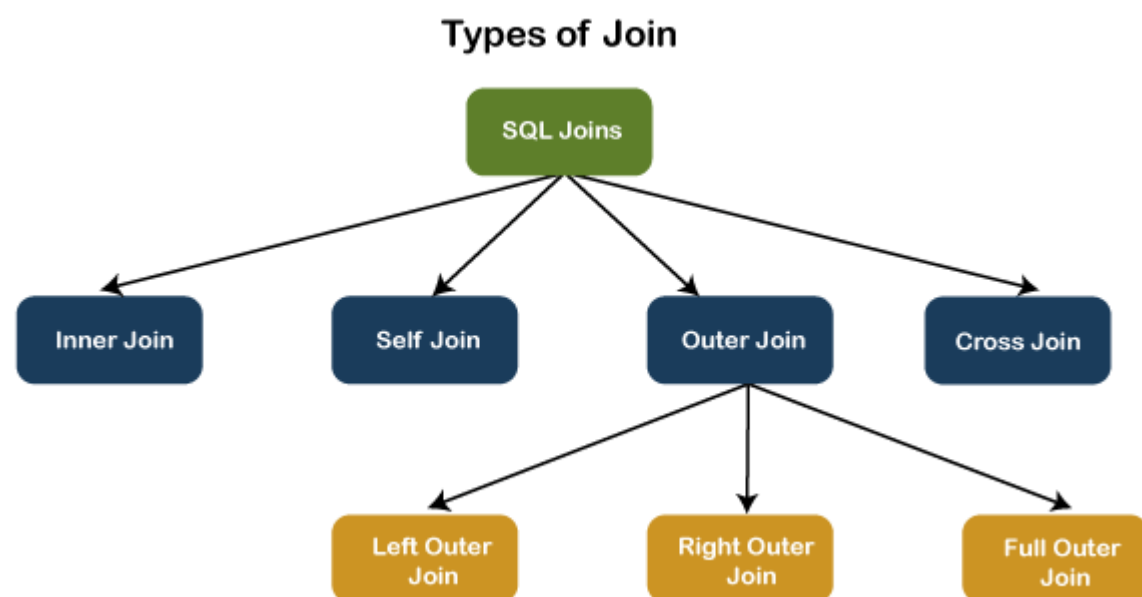
- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN

---

### 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:

- INNER JOIN
- SELF JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN
- CROSS JOIN



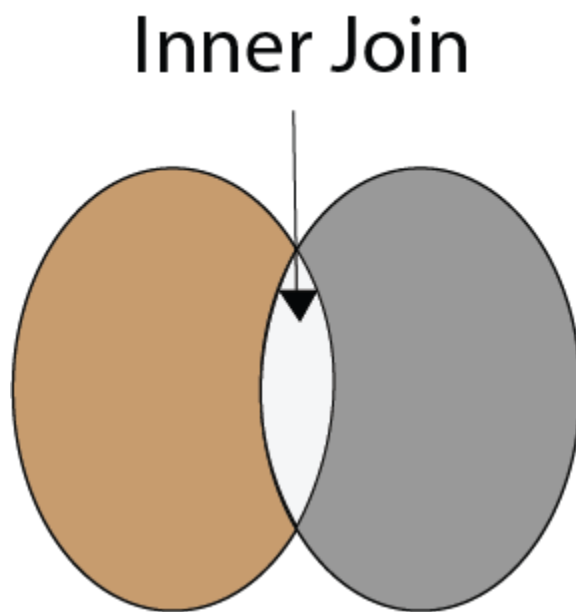
To read more information, [click here](#).

---

### 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:



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. ...;

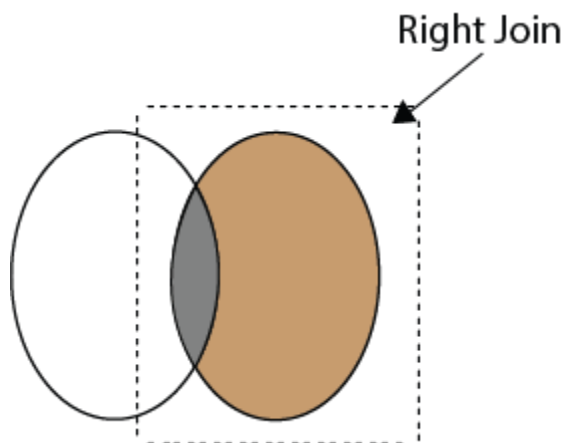
To read more information, [click here](#).

---

## 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;

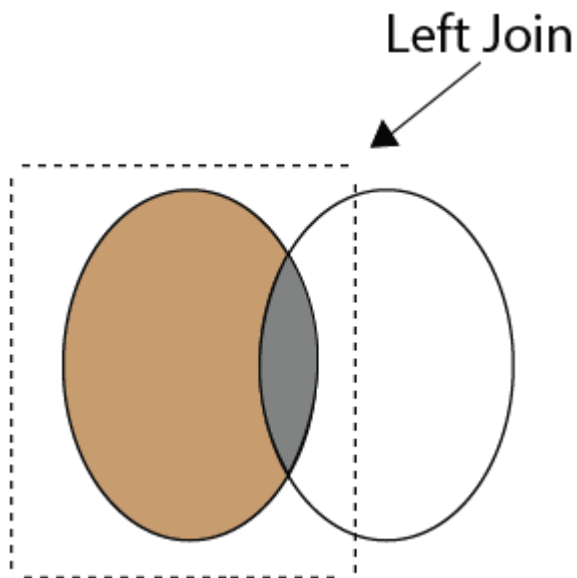
To read more information, [click here](#).

---

### 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:



The following syntax illustrates the RIGHT JOIN:

1. SELECT colum\_lists
2. FROM table1
3. LEFT JOIN table2
4. ON join\_condition;

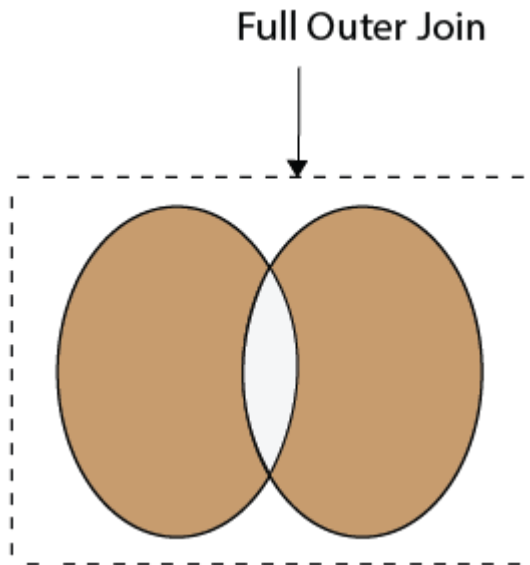
To read more information, [click here](#).

---

## 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:



The following syntax illustrates the FULL JOIN:

1. `SELECT * FROM table1`
2. `FULL OUTER JOIN table2`
3. `ON join_condition;`

To read more information, [click here](#).

---

## 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;

To read more information, [click here](#).

---

## 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

To read more information, [click here](#).

---

### 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 categorized 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;

To read more information, [click here](#).

---

## 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 select 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.

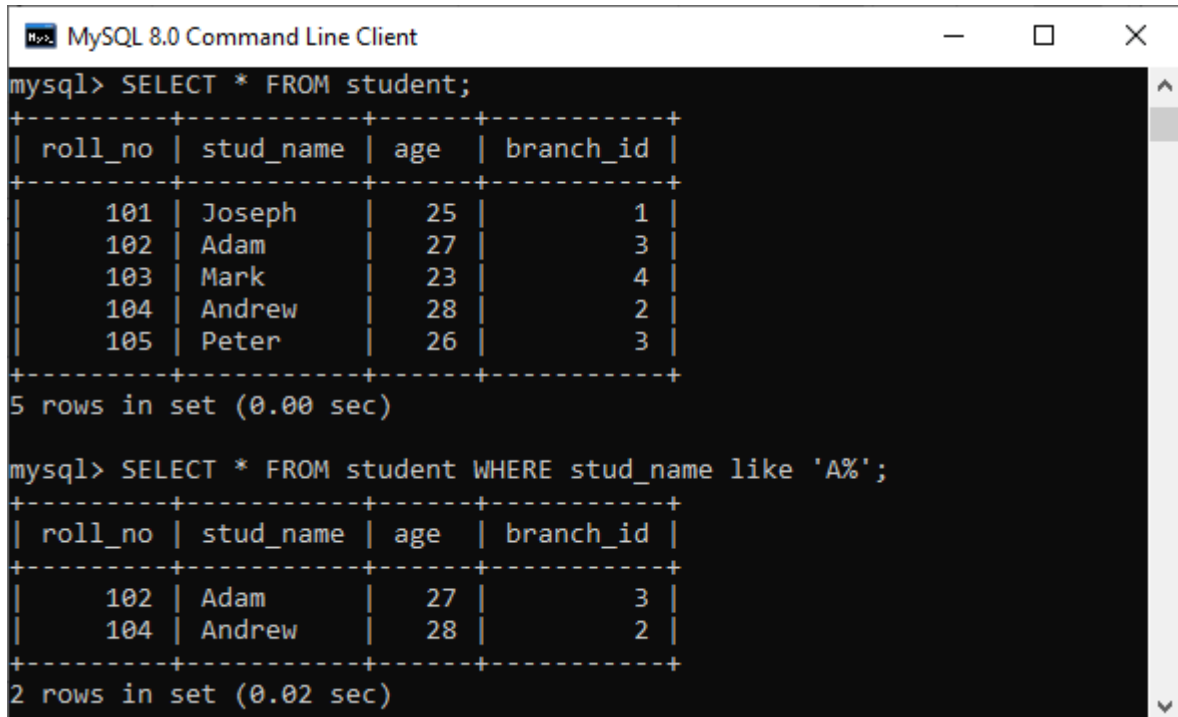
To read more information, [click here](#).

## 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> 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 <b>DML command</b> .	TRUNCATE is a <b>DML command</b> .
3)	We <b>can use the WHERE</b> clause in the DELETE command.	We <b>cannot use the WHERE</b> clause with TRUNCATE.
4)	DELETE statement is used <b>to delete a row</b> from a table.	TRUNCATE statement is used <b>to remove all the rows</b> from a table.
5)	DELETE is <b>slower</b> because it maintained the log.	TRUNCATE statement is <b>faster</b> than DELETE statement as it deletes entire data at a time without maintaining transaction logs.
6)	You <b>can roll back</b> data after using the DELETE statement.	It is <b>not possible to roll back</b> after using the TRUNCATE statement.
7)	DELETE query <b>takes more space</b> .	TRUNCATE query <b>occupies less space</b> .

To read more information, [click here](#).

---

## 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.

To read more information, [click here](#).

---

## 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 convert 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 convert 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 convert 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 Javatpoint') 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 Javatpoint', 'Javatpoint');

**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 'ABCD CBA');

**Output:** BCD C B

**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', 'Javatpoint', NULL) AS Result;

This statement will return the following output:

```
MySQL 8.0 Command Line Client

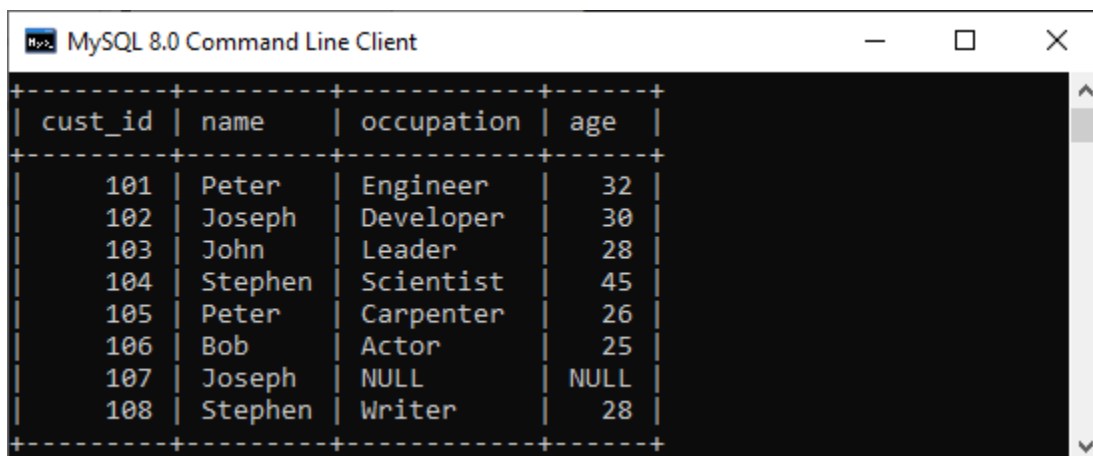
mysql> SELECT COALESCE(NULL, 'Hello', 'Javatpoint', NULL) AS Result;
+-----+
| Result |
+-----+
| Hello  |
+-----+
1 row in set (0.00 sec)
```

## 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:

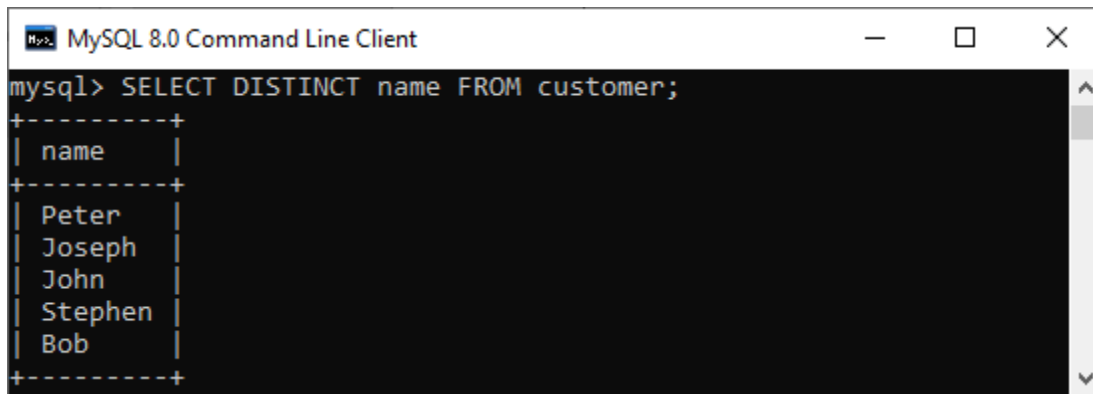
1. SELECT DISTINCT column\_lists FROM table\_name WHERE [condition];

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



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.

A screenshot of the MySQL 8.0 Command Line Client window. The title bar reads "MySQL 8.0 Command Line Client". The command prompt shows the query: `mysql> SELECT DISTINCT name FROM customer;`. The output is displayed in a table format with a header row and five data rows. The header row has a column named "name". The data rows contain the names: Peter, Joseph, John, Stephen, and Bob. The table is enclosed in a box with dashed lines.

name
Peter
Joseph
John
Stephen
Bob

## 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.

```
mysql> SELECT * FROM customer ORDER BY name;
+-----+-----+-----+-----+
| cust_id | name   | occupation | age |
+-----+-----+-----+-----+
| 106     | Bob    | Actor      | 25  |
| 103     | John   | Leader     | 28  |
| 102     | Joseph | Developer  | 30  |
| 107     | Joseph | NULL       | NULL|
| 101     | Peter  | Engineer   | 32  |
| 105     | Peter  | Carpenter  | 26  |
| 104     | Stephen| Scientist  | 45  |
| 108     | Stephen| Writer     | 28  |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> SELECT * FROM customer ORDER BY name DESC;
+-----+-----+-----+-----+
| cust_id | name   | occupation | age |
+-----+-----+-----+-----+
| 104     | Stephen| Scientist  | 45  |
| 108     | Stephen| Writer     | 28  |
| 101     | Peter  | Engineer   | 32  |
| 105     | Peter  | Carpenter  | 26  |
| 102     | Joseph | Developer  | 30  |
| 107     | Joseph | NULL       | NULL|
| 103     | John   | Leader     | 28  |
| 106     | Bob    | Actor      | 25  |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

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 group 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.

To know more differences, [click here](#).

---

## 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:**

- **AVG():** This function is used to returns the average value from specified columns.
  - **COUNT():** This function is used to returns the number of table rows, including rows with null values.
  - **MAX():** This function is used to returns the largest value among the group.
  - **MIN():** This function is used to returns the smallest value among the group.
  - **SUM():** This function is used to returns the total summed values(non-null) of the specified column.
  - **FIRST():** This function is used to returns the first value of an expression.
  - **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:

- Accessing confidential data to modify an SQL query to get desired results.
  - UNION attacks to steal data from different database tables.
  - 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.

- **Single Line Comments:** It starts with two consecutive hyphens (--).
- **Multi-line Comments:** It starts with /\* and ends with \*/.

### 1. What is Database?

A database is an organized collection of data, stored and retrieved digitally from a remote or local computer system. Databases can be vast and complex, and such databases are developed using fixed design and modeling approaches.

### 2. What is DBMS?

DBMS stands for Database Management System. DBMS is a system software responsible for the creation, retrieval, updation, and management of the database. It ensures that our data is consistent, organized, and is easily accessible by serving as an interface between the database and its end-users or application software.

### 3. What is RDBMS? How is it different from DBMS?

RDBMS stands for Relational Database Management System. The key difference [here](#), compared to DBMS, is that RDBMS stores data in the form of a collection of tables, and relations can be defined between the common fields of these tables. Most modern database management systems like MySQL, Microsoft SQL Server, Oracle, IBM DB2, and Amazon Redshift are based on RDBMS.



You can download a PDF version of Sql Interview Questions.

[Download PDF](#)

---

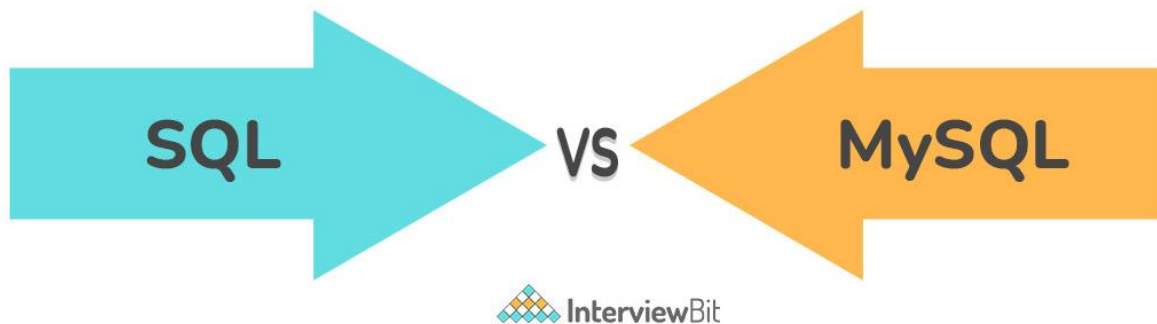
### 4. What is SQL?

SQL stands for Structured Query Language. It is the standard language for relational database management systems. It is especially useful in handling organized data comprised of entities (variables) and relations between different entities of the data.

### 5. What is the difference between SQL and MySQL?

SQL is a standard language for retrieving and manipulating structured databases. On the contrary, MySQL is a relational database management system, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.





## 6. What are Tables and Fields?

A table is an organized collection of data stored in the form of rows and columns. Columns can be categorized as vertical and rows as horizontal. The columns in a table are called fields while the rows can be referred to as records.

## 7. What are Constraints in SQL?

Constraints are used to specify the rules concerning data in the table. It can be applied for single or multiple fields in an SQL table during the creation of the table or after creating using the ALTER TABLE command. The constraints are:

- **NOT NULL** - Restricts NULL value from being inserted into a column.
- **CHECK** - Verifies that all values in a field satisfy a condition.
- **DEFAULT** - Automatically assigns a default value if no value has been specified for the field.
- **UNIQUE** - Ensures unique values to be inserted into the field.
- **INDEX** - Indexes a field providing faster retrieval of records.
- **PRIMARY KEY** - Uniquely identifies each record in a table.
- **FOREIGN KEY** - Ensures referential integrity for a record in another table.

## 8. What is a Primary Key?

The PRIMARY KEY constraint uniquely identifies each row in a table. It must contain UNIQUE values and has an implicit NOT NULL constraint.

A table in SQL is strictly restricted to have one and only one primary key, which is comprised of single or multiple fields (columns).

```
CREATE TABLE Students ( /* Create table with a single field as primary
key */
    ID INT NOT NULL
    Name VARCHAR(255)
    PRIMARY KEY (ID)
);
```

```
CREATE TABLE Students ( /* Create table with multiple fields as primary
key */
```

```

ID INT NOT NULL
LastName VARCHAR(255)
FirstName VARCHAR(255) NOT NULL,
CONSTRAINT PK_Student
PRIMARY KEY (ID, FirstName)
);

ALTER TABLE Students /* Set a column as primary key */
ADD PRIMARY KEY (ID);
ALTER TABLE Students /* Set multiple columns as primary key */
ADD CONSTRAINT PK_Student /*Naming a Primary Key*/
PRIMARY KEY (ID, FirstName);

```

write a sql statement to add primary key 't\_id' to the table 'teachers'.

Write a SQL statement to add primary key constraint 'pk\_a' for table 'table\_a' and fields 'col\_b, col\_c'.

## 9. What is a UNIQUE constraint?

A UNIQUE constraint ensures that all values in a column are different. This provides uniqueness for the column(s) and helps identify each row uniquely. Unlike primary key, there can be multiple unique constraints defined per table. The code syntax for UNIQUE is quite similar to that of PRIMARY KEY and can be used interchangeably.

```

CREATE TABLE Students ( /* Create table with a single field as unique
*/
    ID INT NOT NULL UNIQUE
    Name VARCHAR(255)
);

CREATE TABLE Students ( /* Create table with multiple fields as unique
*/
    ID INT NOT NULL
    LastName VARCHAR(255)
    FirstName VARCHAR(255) NOT NULL
    CONSTRAINT PK_Student
    UNIQUE (ID, FirstName)
);

ALTER TABLE Students /* Set a column as unique */
ADD UNIQUE (ID);
ALTER TABLE Students /* Set multiple columns as unique */
ADD CONSTRAINT PK_Student /* Naming a unique constraint */
UNIQUE (ID, FirstName);

```

## 10. What is a Foreign Key?

A FOREIGN KEY comprises of single or collection of fields in a table that essentially refers to the PRIMARY KEY in another table. Foreign key constraint ensures referential integrity in the relation between two tables.

The table with the foreign key constraint is labeled as the child table, and the table containing the candidate key is labeled as the referenced or parent table.

```
CREATE TABLE Students ( /* Create table with foreign key - Way 1 */
    ID INT NOT NULL
    Name VARCHAR(255)
    LibraryID INT
    PRIMARY KEY (ID)
    FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

CREATE TABLE Students ( /* Create table with foreign key - Way 2 */
    ID INT NOT NULL PRIMARY KEY
    Name VARCHAR(255)
    LibraryID INT FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

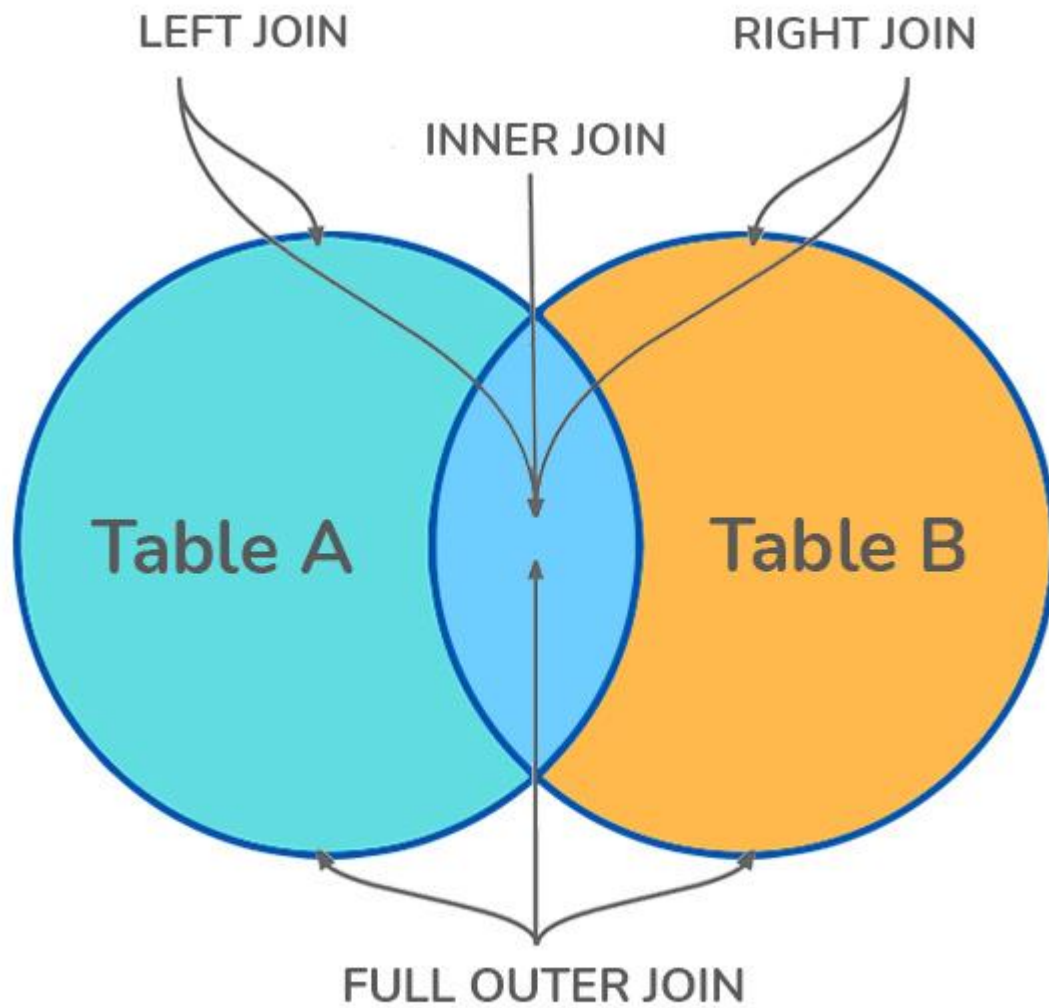
ALTER TABLE Students /* Add a new foreign key */
ADD FOREIGN KEY (LibraryID)
REFERENCES Library (LibraryID);
```

What type of integrity constraint does the foreign key ensure?

Write a SQL statement to add a FOREIGN KEY 'col\_fk' in 'table\_y' that references 'col\_pk' in 'table\_x'.

## 11. What is a Join? List its different types.

The **SQL Join** clause is used to combine records (rows) from two or more tables in a SQL database based on a related column between the two.



There are four different types of JOINS in SQL:

- **(INNER) JOIN:** Retrieves records that have matching values in both tables involved in the join. This is the widely used join for queries.

```
SELECT *  
FROM Table_A  
JOIN Table_B;  
SELECT *  
FROM Table_A  
INNER JOIN Table_B;
```

- **LEFT (OUTER) JOIN:** Retrieves all the records/rows from the left and the matched records/rows from the right table.

```
SELECT *  
FROM Table_A A  
LEFT JOIN Table_B B
```

```
ON A.col = B.col;
```

- **RIGHT (OUTER) JOIN:** Retrieves all the records/rows from the right and the matched records/rows from the left table.

```
SELECT *  
FROM Table_A A  
RIGHT JOIN Table_B B  
ON A.col = B.col;
```

- **FULL (OUTER) JOIN:** Retrieves all the records where there is a match in either the left or right table.

```
SELECT *  
FROM Table_A A  
FULL JOIN Table_B B  
ON A.col = B.col;
```

## 12. What is a Self-Join?

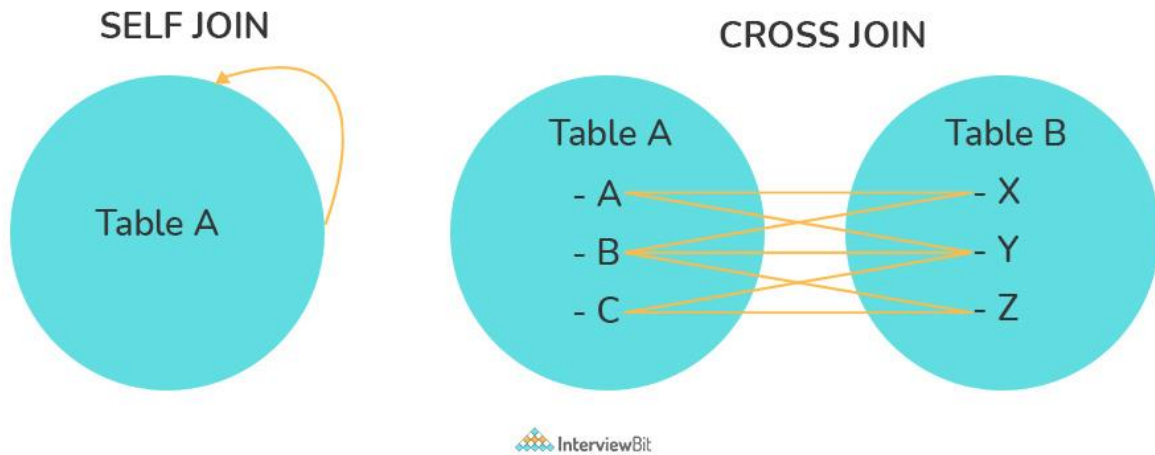
A **self JOIN** is a case of regular join where a table is joined to itself based on some relation between its own column(s). Self-join uses the INNER JOIN or LEFT JOIN clause and a table alias is used to assign different names to the table within the query.

```
SELECT A.emp_id AS "Emp_ID", A.emp_name AS "Employee",  
B.emp_id AS "Sup_ID", B.emp_name AS "Supervisor"  
FROM employee A, employee B  
WHERE A.emp_sup = B.emp_id;
```

## 13. What is a Cross-Join?

Cross join can be defined as a cartesian product of the two tables included in the join. The table after join contains the same number of rows as in the cross-product of the number of rows in the two tables. If a WHERE clause is used in cross join then the query will work like an INNER JOIN.

```
SELECT stu.name, sub.subject  
FROM students AS stu  
CROSS JOIN subjects AS sub;
```



Write a SQL statement to CROSS JOIN 'table\_1' with 'table\_2' and fetch 'col\_1' from table\_1 & 'col\_2' from table\_2 respectively. Do not use alias.

Write a SQL statement to perform SELF JOIN for 'Table\_X' with alias 'Table\_1' and 'Table\_2', on columns 'Col\_1' and 'Col\_2' respectively.

#### 14. What is an Index? Explain its different types.

A database index is a data structure that provides a quick lookup of data in a column or columns of a table. It enhances the speed of operations accessing data from a database table at the cost of additional writes and memory to maintain the index data structure.

```
CREATE INDEX index_name /* Create Index */
ON table_name (column_1, column_2);
DROP INDEX index_name; /* Drop Index */
```

There are different types of indexes that can be created for different purposes:

- **Unique and Non-Unique Index:**

Unique indexes are indexes that help maintain data integrity by ensuring that no two rows of data in a table have identical key values. Once a unique index has been defined for a table, uniqueness is enforced whenever keys are added or changed within the index.

```
CREATE UNIQUE INDEX myIndex
ON students (enroll_no);
```

Non-unique indexes, on the other hand, are not used to enforce constraints on the tables with which they are associated. Instead, non-unique indexes are used solely to improve query performance by maintaining a sorted order of data values that are used frequently.

- **Clustered and Non-Clustered Index:**

Clustered indexes are indexes whose order of the rows in the database corresponds to the order of the rows in the index. This is why only one clustered index can exist in a given table, whereas, multiple non-clustered indexes can exist in the table.

The only difference between clustered and non-clustered indexes is that the database manager attempts to keep the data in the database in the same order as the corresponding keys appear in the clustered index.

Clustering indexes can improve the performance of most query operations because they provide a linear-access path to data stored in the database.

Write a SQL statement to create a UNIQUE INDEX "my\_index" on "my\_table" for fields "column\_1" & "column\_2".

## **15. What is the difference between Clustered and Non-clustered index?**

As explained above, the differences can be broken down into three small factors -

- Clustered index modifies the way records are stored in a database based on the indexed column. A non-clustered index creates a separate entity within the table which references the original table.
- Clustered index is used for easy and speedy retrieval of data from the database, whereas, fetching records from the non-clustered index is relatively slower.
- In SQL, a table can have a single clustered index whereas it can have multiple non-clustered indexes.

## **16. What is Data Integrity?**

Data Integrity is the assurance of accuracy and consistency of data over its entire life-cycle and is a critical aspect of the design, implementation, and usage of any system which stores, processes, or retrieves data. It also defines integrity constraints to enforce business rules on the data when it is entered into an application or a database.

## 17. What is a Query?

A query is a request for data or information from a database table or combination of tables. A database query can be either a select query or an action query.

```
SELECT fname, lname      /* select query */
FROM myDb.students
WHERE student_id = 1;
UPDATE myDB.students     /* action query */
SET fname = 'Captain', lname = 'America'
WHERE student_id = 1;
```

## 18. What is a Subquery? What are its types?

A subquery is a query within another query, also known as a **nested query** or **inner query**. It is used to restrict or enhance the data to be queried by the main query, thus restricting or enhancing the output of the main query respectively. For example, here we fetch the contact information for students who have enrolled for the maths subject:

```
SELECT name, email, mob, address
FROM myDb.contacts
WHERE roll_no IN (
    SELECT roll_no
    FROM myDb.students
    WHERE subject = 'Maths');
```

There are two types of subquery - **Correlated** and **Non-Correlated**.

- A **correlated** subquery cannot be considered as an independent query, but it can refer to the column in a table listed in the FROM of the main query.
- A **non-correlated** subquery can be considered as an independent query and the output of the subquery is substituted in the main query.

Write a SQL query to update the field "status" in table "applications" from 0 to 1.

Write a SQL query to select the field "app\_id" in table "applications" where "app\_id" less than 1000.

Write a SQL query to fetch the field "app\_name" from "apps" where "apps.id" is equal to the above collection of "app\_id".



## 19. What is the SELECT statement?

SELECT operator in SQL is used to select data from a database. The data returned is stored in a result table, called the result-set.

```
SELECT * FROM myDB.students;
```

## 20. What are some common clauses used with SELECT query in SQL?

Some common SQL clauses used in conjunction with a SELECT query are as follows:

- **WHERE** clause in SQL is used to filter records that are necessary, based on specific conditions.
- **ORDER BY** clause in SQL is used to sort the records based on some field(s) in ascending (**ASC**) or descending order (**DESC**).

```
SELECT *  
FROM myDB.students  
WHERE graduation_year = 2019  
ORDER BY studentID DESC;
```

- **GROUP BY** clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.
- **HAVING** clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records.

```
SELECT COUNT(studentId), country  
FROM myDB.students  
WHERE country != "INDIA"  
GROUP BY country  
HAVING COUNT(studentID) > 5;
```

## 21. What are UNION, MINUS and INTERSECT commands?

The **UNION** operator combines and returns the result-set retrieved by two or more SELECT statements.

The **MINUS** operator in SQL is used to remove duplicates from the result-set obtained by the second SELECT query from the result-set obtained by the first SELECT query and then return the filtered results from the first.

The **INTERSECT** clause in SQL combines the result-set fetched by the two SELECT statements where records from one match the other and then returns this intersection of result-sets.

Certain conditions need to be met before executing either of the above statements in SQL -

- Each SELECT statement within the clause must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement should necessarily have the same order

```
SELECT name FROM Students /* Fetch the union of queries */
UNION
SELECT name FROM Contacts;
SELECT name FROM Students /* Fetch the union of queries with
duplicates*/
UNION ALL
SELECT name FROM Contacts;
SELECT name FROM Students /* Fetch names from students */
MINUS /* that aren't present in contacts */
SELECT name FROM Contacts;
SELECT name FROM Students /* Fetch names from students */
INTERSECT /* that are present in contacts as well */
SELECT name FROM Contacts;
```

Write a SQL query to fetch "names" that are present in either table "accounts" or in table "registry".

Write a SQL query to fetch "names" that are present in "accounts" but not in table "registry".

Write a SQL query to fetch "names" from table "contacts" that are neither present in "accounts.name" nor in "registry.name".

## 22. What is Cursor? How to use a Cursor?

A database cursor is a control structure that allows for the traversal of records in a database. Cursors, in addition, facilitates processing after traversal, such as retrieval, addition, and deletion of database records. They can be viewed as a pointer to one row in a set of rows.

### Working with SQL Cursor:

1. **DECLARE** a cursor after any variable declaration. The cursor declaration must always be associated with a SELECT Statement.
2. Open cursor to initialize the result set. The **OPEN** statement must be called before fetching rows from the result set.

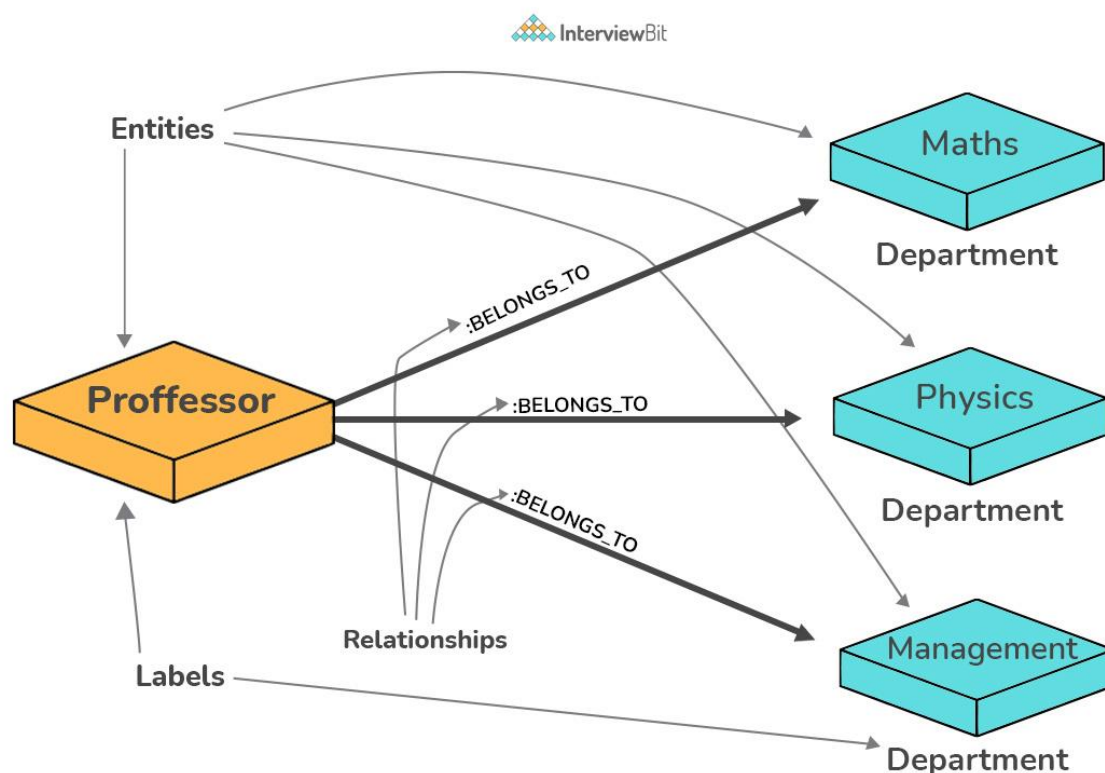
3. **FETCH** statement to retrieve and move to the next row in the result set.
4. Call the **CLOSE** statement to deactivate the cursor.
5. Finally use the **DEALLOCATE** statement to delete the cursor definition and release the associated resources.

```
DECLARE @name VARCHAR(50) /* Declare All Required Variables */
DECLARE db_cursor CURSOR FOR /* Declare Cursor Name*/
SELECT name
FROM myDB.students
WHERE parent_name IN ('Sara', 'Ansh')
OPEN db_cursor /* Open cursor and Fetch data into @name */
FETCH next
FROM db_cursor
INTO @name
CLOSE db_cursor /* Close the cursor and deallocate the resources */
DEALLOCATE db_cursor
```

## 23. What are Entities and Relationships?

**Entity:** An entity can be a real-world object, either tangible or intangible, that can be easily identifiable. For example, in a college database, students, professors, workers, departments, and projects can be referred to as entities. Each entity has some associated properties that provide it an identity.

**Relationships:** Relations or links between entities that have something to do with each other. For example - The employee's table in a company's database can be associated with the salary table in the same database.



## 24. List the different types of relationships in SQL.

- **One-to-One** - This can be defined as the relationship between two tables where each record in one table is associated with the maximum of one record in the other table.
- **One-to-Many & Many-to-One** - This is the most commonly used relationship where a record in a table is associated with multiple records in the other table.
- **Many-to-Many** - This is used in cases when multiple instances on both sides are needed for defining a relationship.
- **Self-Referencing Relationships** - This is used when a table needs to define a relationship with itself.

## 25. What is an Alias in SQL?

An alias is a feature of SQL that is supported by most, if not all, RDBMSs. It is a temporary name assigned to the table or table column for the purpose of a particular SQL query. In addition, aliasing can be employed as an obfuscation technique to secure the real names of database fields. A table alias is also called a correlation name.

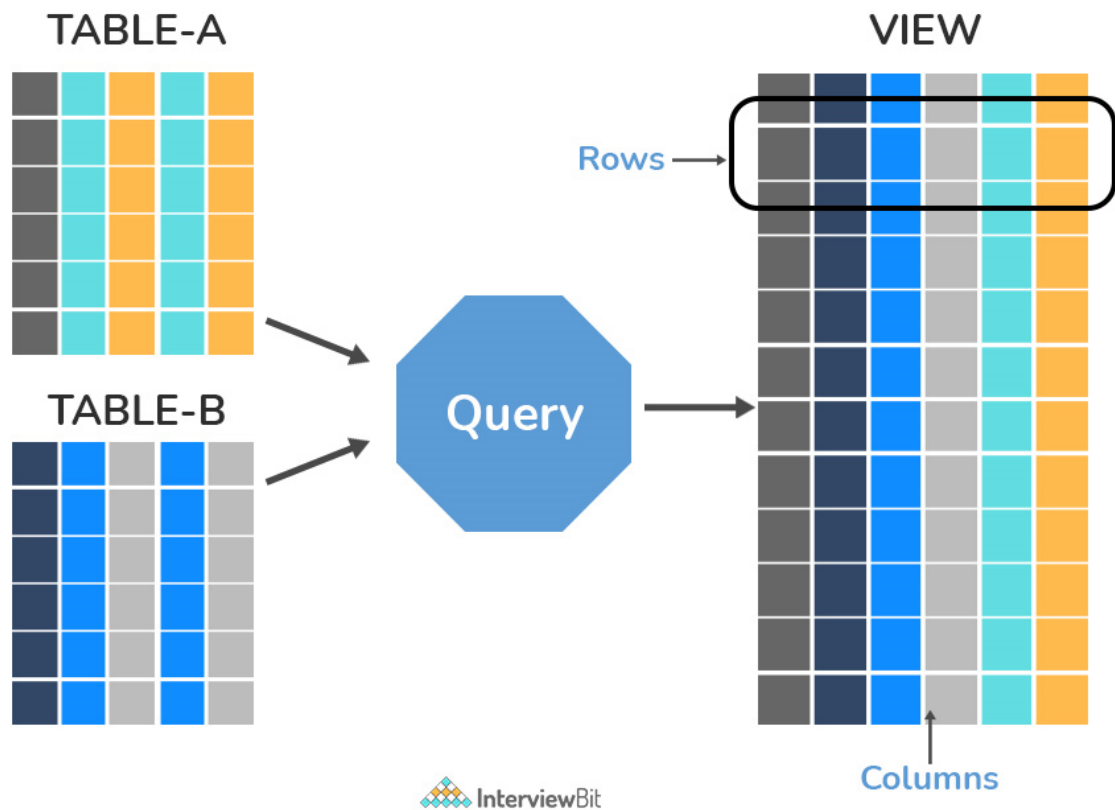
An alias is represented explicitly by the AS keyword but in some cases, the same can be performed without it as well. Nevertheless, using the AS keyword is always a good practice.

```
SELECT A.emp_name AS "Employee" /* Alias using AS keyword */  
B.emp_name AS "Supervisor"  
FROM employee A, employee B /* Alias without AS keyword */  
WHERE A.emp_sup = B.emp_id;
```

Write an SQL statement to select all from table "Limited" with alias "Ltd".

## 26. What is a View?

A view in SQL is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.



## 27. What is Normalization?

Normalization represents the way of organizing structured data in the database efficiently. It includes the creation of tables, establishing relationships between them, and defining rules for those relationships. Inconsistency and redundancy can be kept in check based on these rules, hence, adding flexibility to the database.

## 28. What is Denormalization?

Denormalization is the inverse process of normalization, where the normalized schema is converted into a schema that has redundant information. The performance is improved by using redundancy and keeping the redundant data consistent. The reason for performing denormalization is the overheads produced in the query processor by an over-normalized structure.

## 29. What are the various forms of Normalization?

Normal Forms are used to eliminate or reduce redundancy in database tables. The different forms are as follows:

- **First Normal Form:**  
A relation is in first normal form if every attribute in that relation is a **single-valued attribute**. If a relation contains a composite or multi-valued

attribute, it violates the first normal form. Let's consider the following **students** table. Each student in the table, has a name, his/her address, and the books they issued from the public library -

**Students Table**

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter), Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho), Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward), Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

As we can observe, the Books Issued field has more than one value per record, and to convert it into 1NF, this has to be resolved into separate individual records for each book issued. Check the following table in 1NF form -

**Students Table (1st Normal Form)**

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter)	Ms.
Sara	Amanora Park Town 94	Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho)	Mr.
Ansh	62nd Sector A-10	Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward)	Mrs.
Sara	24th Street Park Avenue	Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

- **Second Normal Form:**

A relation is in second normal form if it satisfies the conditions for the first normal form and does not contain any partial dependency. A relation in 2NF has **no partial dependency**, i.e., it has no non-prime attribute that depends on any proper subset of any candidate key of the table. Often, specifying a single column Primary Key is the solution to the problem. Examples -

**Example 1** - Consider the above example. As we can observe, the Students Table in the 1NF form has a candidate key in the form of [Student, Address] that can uniquely identify all records in the table. The field Books Issued (non-prime

attribute) depends partially on the Student field. Hence, the table is not in 2NF. To convert it into the 2nd Normal Form, we will partition the tables into two while specifying a new **Primary Key** attribute to identify the individual records in the Students table. The **Foreign Key** constraint will be set on the other table to ensure referential integrity.

**Students Table (2nd Normal Form)**

Student_ID	Student	Address	Salutation
1	Sara	Amanora Park Town 94	Ms.
2	Ansh	62nd Sector A-10	Mr.
3	Sara	24th Street Park Avenue	Mrs.
4	Ansh	Windsor Street 777	Mr.

**Books Table (2nd Normal Form)**

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

**Example 2** - Consider the following dependencies in relation to R(W,X,Y,Z)

WX → Y	[W <b>and</b> X together determine Y]
XY → Z	[X <b>and</b> Y together determine Z]

Here, WX is the only candidate key and there is no partial dependency, i.e., any proper subset of WX doesn't determine any non-prime attribute in the relation.

- **Third Normal Form**

A relation is said to be in the third normal form, if it satisfies the conditions for the second normal form and there is **no transitive dependency** between the non-prime attributes, i.e., all non-prime attributes are determined only by the candidate keys of the relation and not by any other non-prime attribute.

**Example 1** - Consider the Students Table in the above example. As we can observe, the Students Table in the 2NF form has a single candidate key Student\_ID (primary

key) that can uniquely identify all records in the table. The field Salutation (non-prime attribute), however, depends on the Student Field rather than the candidate key. Hence, the table is not in 3NF. To convert it into the 3rd Normal Form, we will once again partition the tables into two while specifying a new **Foreign Key** constraint to identify the salutations for individual records in the Students table. The **Primary Key** constraint for the same will be set on the Salutations table to identify each record uniquely.

**Students Table (3rd Normal Form)**

Student_ID	Student	Address	Salutation_ID
1	Sara	Amanora Park Town 94	1
2	Ansh	62nd Sector A-10	2
3	Sara	24th Street Park Avenue	3
4	Ansh	Windsor Street 777	1

**Books Table (3rd Normal Form)**

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

**Salutations Table (3rd Normal Form)**

Salutation_ID	Salutation
1	Ms.
2	Mr.
3	Mrs.

**Example 2** - Consider the following dependencies in relation to R(P,Q,R,S,T)

```

P -> QR      [P together determine C]
RS -> T      [B and C together determine D]
Q -> S
T -> P

```



For the above relation to exist in 3NF, all possible candidate keys in the above relation should be {P, RS, QR, T}.

- **Boyce-Codd Normal Form**

A relation is in Boyce-Codd Normal Form if satisfies the conditions for third normal form and for every functional dependency, Left-Hand-Side is super key. In other words, a relation in BCNF has non-trivial functional dependencies in form  $X \rightarrow Y$ , such that X is always a super key. For example - In the above example, Student\_ID serves as the sole unique identifier for the Students Table and Salutation\_ID for the Salutations Table, thus these tables exist in BCNF. The same cannot be said for the Books Table and there can be several books with common Book Names and the same Student\_ID.

### 30. What are the TRUNCATE, DELETE and DROP statements?

**DELETE** statement is used to delete rows from a table.

```
DELETE FROM Candidates
WHERE CandidateId > 1000;
```

**TRUNCATE** command is used to delete all the rows from the table and free the space containing the table.

```
TRUNCATE TABLE Candidates;
```

**DROP** command is used to remove an object from the database. If you drop a table, all the rows in the table are deleted and the table structure is removed from the database.

```
DROP TABLE Candidates;
```

Write a SQL statement to wipe a table 'Temporary' from memory.

Write a SQL query to remove first 1000 records from table 'Temporary' based on 'id'.

Write a SQL statement to delete the table 'Temporary' while keeping its relations intact.

### 31. What is the difference between DROP and TRUNCATE statements?

If a table is dropped, all things associated with the tables are dropped as well. This includes - the relationships defined on the table with other tables, the integrity checks and constraints, access privileges and other grants that the table has. To create and use the table again in its original form, all these relations, checks, constraints, privileges and relationships need to be redefined. However, if a table is truncated, none of the above problems exist and the table retains its original structure.

### 32. What is the difference between DELETE and TRUNCATE statements?

The **TRUNCATE** command is used to delete all the rows from the table and free the space containing the table.

The **DELETE** command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.

### 33. What are Aggregate and Scalar functions?

An aggregate function performs operations on a collection of values to return a single scalar value. Aggregate functions are often used with the GROUP BY and HAVING clauses of the SELECT statement. Following are the widely used SQL aggregate functions:

- **AVG()** - Calculates the mean of a collection of values.
- **COUNT()** - Counts the total number of records in a specific table or view.
- **MIN()** - Calculates the minimum of a collection of values.
- **MAX()** - Calculates the maximum of a collection of values.
- **SUM()** - Calculates the sum of a collection of values.
- **FIRST()** - Fetches the first element in a collection of values.
- **LAST()** - Fetches the last element in a collection of values.

**Note:** All aggregate functions described above ignore NULL values except for the COUNT function.

A scalar function returns a single value based on the input value. Following are the widely used SQL scalar functions:

- **LEN()** - Calculates the total length of the given field (column).
- **UCASE()** - Converts a collection of string values to uppercase characters.
- **LCASE()** - Converts a collection of string values to lowercase characters.
- **MID()** - Extracts substrings from a collection of string values in a table.

- **CONCAT()** - Concatenates two or more strings.
- **RAND()** - Generates a random collection of numbers of a given length.
- **ROUND()** - Calculates the round-off integer value for a numeric field (or decimal point values).
- **NOW()** - Returns the current date & time.
- **FORMAT()** - Sets the format to display a collection of values.

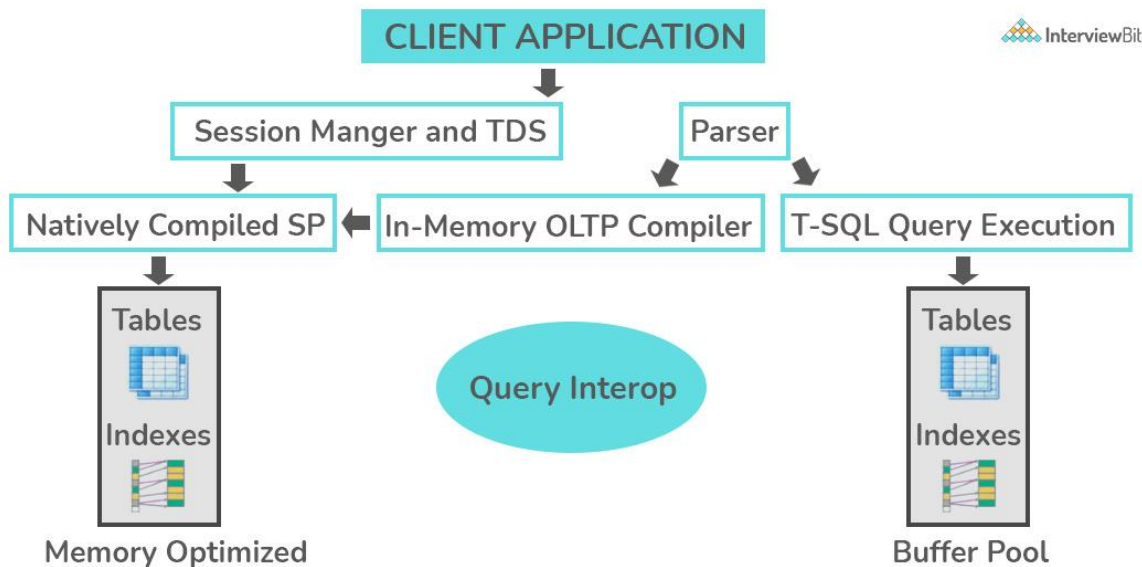
### 34. What is User-defined function? What are its various types?

The user-defined functions in SQL are like functions in any other programming language that accept parameters, perform complex calculations, and return a value. They are written to use the logic repetitively whenever required. There are two types of SQL user-defined functions:

- **Scalar Function:** As explained earlier, user-defined scalar functions return a single scalar value.
- **Table-Valued Functions:** User-defined table-valued functions return a table as output.
  - **Inline:** returns a table data type based on a single SELECT statement.
  - **Multi-statement:** returns a tabular result-set but, unlike inline, multiple SELECT statements can be used inside the function body.

### 35. What is OLTP?

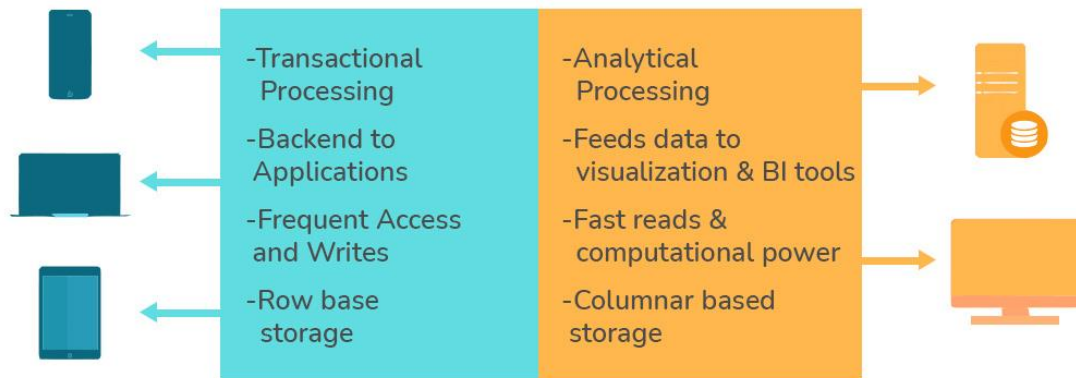
**OLTP** stands for Online Transaction Processing, is a class of software applications capable of supporting transaction-oriented programs. An essential attribute of an OLTP system is its ability to maintain concurrency. To avoid single points of failure, OLTP systems are often decentralized. These systems are usually designed for a large number of users who conduct short transactions. Database queries are usually simple, require sub-second response times, and return relatively few records. Here is an insight into the working of an OLTP system [ *Note - The figure is not important for interviews* ] -



### 36. What are the differences between OLTP and OLAP?

**OLTP** stands for **Online Transaction Processing**, is a class of software applications capable of supporting transaction-oriented programs. An important attribute of an OLTP system is its ability to maintain concurrency. OLTP systems often follow a decentralized architecture to avoid single points of failure. These systems are generally designed for a large audience of end-users who conduct short transactions. Queries involved in such databases are generally simple, need fast response times, and return relatively few records. A number of transactions per second acts as an effective measure for such systems.

**OLAP** stands for **Online Analytical Processing**, a class of software programs that are characterized by the relatively low frequency of online transactions. Queries are often too complex and involve a bunch of aggregations. For OLAP systems, the effectiveness measure relies highly on response time. Such systems are widely used for data mining or maintaining aggregated, historical data, usually in multi-dimensional schemas.



### 37. What is Collation? What are the different types of Collation Sensitivity?

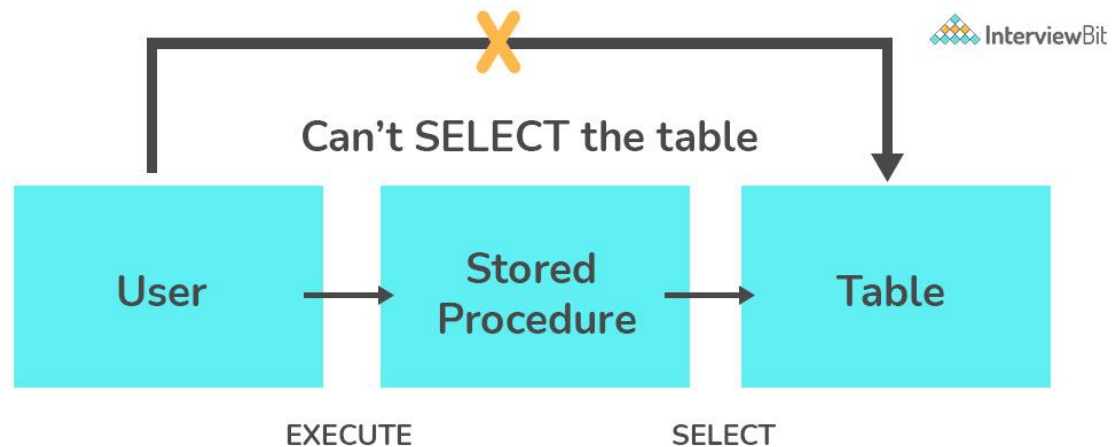
Collation refers to a set of rules that determine how data is sorted and compared. Rules defining the correct character sequence are used to sort the character data. It incorporates options for specifying case sensitivity, accent marks, kana character types, and character width. Below are the different types of collation sensitivity:

- **Case sensitivity:** **A** and **a** are treated differently.
- **Accent sensitivity:** **a** and **á** are treated differently.
- **Kana sensitivity:** Japanese kana characters Hiragana and Katakana are treated differently.
- **Width sensitivity:** Same character represented in single-byte (half-width) and double-byte (full-width) are treated differently.

### 38. What is a Stored Procedure?

A stored procedure is a subroutine available to applications that access a relational database management system (RDBMS). Such procedures are stored in the database data dictionary. The sole disadvantage of stored procedure is that it can be executed nowhere except in the database and occupies more memory in the database server. It also provides a sense of security and functionality as users who can't access the data directly can be granted access via stored procedures.

```
DELIMITER $$
CREATE PROCEDURE FetchAllStudents()
BEGIN
SELECT * FROM myDB.students;
END $$
DELIMITER ;
```



### 39. What is a Recursive Stored Procedure?

A stored procedure that calls itself until a boundary condition is reached, is called a recursive stored procedure. This recursive function helps the programmers to deploy the same set of code several times as and when required. Some SQL programming languages limit the recursion depth to prevent an infinite loop of procedure calls from causing a stack overflow, which slows down the system and may lead to system crashes.

```
DELIMITER $$      /* Set a new delimiter => $$ */
CREATE PROCEDURE calctotal( /* Create the procedure */
    IN number INT,      /* Set Input and Output variables */
    OUT total INT
) BEGIN
DECLARE score INT DEFAULT NULL; /* Set the default value => "score" */
SELECT awards FROM achievements /* Update "score" via SELECT query */
WHERE id = number INTO score;
IF score IS NULL THEN SET total = 0; /* Termination condition */
ELSE
CALL calctotal(number+1); /* Recursive call */
SET total = total + score; /* Action after recursion */
END IF;
END $$            /* End of procedure */
DELIMITER ;      /* Reset the delimiter */
```

### 40. How to create empty tables with the same structure as another table?

Creating empty tables with the same structure can be done smartly by fetching the records of one table into a new table using the INTO operator while fixing a WHERE clause to be false for all records. Hence, SQL prepares the new table with a duplicate structure to accept the fetched records but since no records get fetched due to the WHERE clause in action, nothing is inserted into the new table.

```
SELECT * INTO Students_copy
FROM Students WHERE 1 = 2;
```

## 41. What is Pattern Matching in SQL?

SQL pattern matching provides for pattern search in data if you have no clue as to what that word should be. This kind of SQL query uses wildcards to match a string pattern, rather than writing the exact word. The LIKE operator is used in conjunction with **SQL Wildcards** to fetch the required information.

- **Using the % wildcard to perform a simple search**

The % wildcard matches zero or more characters of any type and can be used to define wildcards both before and after the pattern. Search a student in your database with first name beginning with the letter K:

```
SELECT *  
FROM students  
WHERE first_name LIKE 'K%'
```

- **Omitting the patterns using the NOT keyword**

Use the NOT keyword to select records that don't match the pattern. This query returns all students whose first name does not begin with K.

```
SELECT *  
FROM students  
WHERE first_name NOT LIKE 'K%'
```

- **Matching a pattern anywhere using the % wildcard twice**

Search for a student in the database where he/she has a K in his/her first name.

```
SELECT *  
FROM students  
WHERE first_name LIKE '%Q%'
```

- **Using the \_ wildcard to match pattern at a specific position**

The \_ wildcard matches exactly one character of any type. It can be used in conjunction with % wildcard. This query fetches all students with letter K at the third position in their first name.

```
SELECT *  
FROM students  
WHERE first_name LIKE '__K%'
```

- **Matching patterns for a specific length**

The \_ wildcard plays an important role as a limitation when it matches exactly one character. It limits the length and position of the matched results. For example -

```

SELECT * /* Matches first names with three or more letters */
FROM students
WHERE first_name LIKE '____%'

SELECT * /* Matches first names with exactly four characters */
FROM students
WHERE first_name LIKE '____'

```

## PostgreSQL Interview Questions

### 42. What is PostgreSQL?

[PostgreSQL](#) was first called Postgres and was developed by a team led by Computer Science Professor Michael Stonebraker in 1986. It was developed to help developers build enterprise-level applications by upholding data integrity by making systems fault-tolerant. PostgreSQL is therefore an enterprise-level, flexible, robust, open-source, and object-relational DBMS that supports flexible workloads along with handling concurrent users. It has been consistently supported by the global developer community. Due to its fault-tolerant nature, PostgreSQL has gained widespread popularity among developers.

### 43. How do you define Indexes in PostgreSQL?

Indexes are the inbuilt functions in PostgreSQL which are used by the queries to perform search more efficiently on a table in the database. Consider that you have a table with thousands of records and you have the below query that only a few records can satisfy the condition, then it will take a lot of time to search and return those rows that abide by this condition as the engine has to perform the search operation on every single to check this condition. This is undoubtedly inefficient for a system dealing with huge data. Now if this system had an index on the column where we are applying search, it can use an efficient method for identifying matching rows by walking through only a few levels. This is called indexing.

```
Select * from some_table where table_col=120
```

### 44. How will you change the datatype of a column?

This can be done by using the ALTER TABLE statement as shown below:

#### Syntax:

```

ALTER TABLE tname
ALTER COLUMN col_name [SET DATA] TYPE new_data_type;

```

### 45. What is the command used for creating a database in PostgreSQL?



The first step of using PostgreSQL is to create a database. This is done by using the `createdb` command as shown below: `createdb db_name`  
After running the above command, if the database creation was successful, then the below message is shown:

```
CREATE DATABASE
```

## 46. How can we start, restart and stop the PostgreSQL server?

- To **start** the PostgreSQL server, we run:

```
service postgresql start
```

- Once the **server** is successfully started, we get the below message:

```
Starting PostgreSQL: ok
```

- To **restart** the PostgreSQL server, we run:

```
service postgresql restart
```

Once the server is successfully restarted, we get the message:

```
Restarting PostgreSQL: server stopped  
ok
```

- To **stop** the server, we run the command:

```
service postgresql stop
```

Once stopped successfully, we get the message:

```
Stopping PostgreSQL: server stopped  
ok
```

## 47. What are partitioned tables called in PostgreSQL?

Partitioned tables are logical structures that are used for dividing large tables into smaller structures that are called partitions. This approach is used for effectively increasing the query performance while dealing with large database tables. To create a partition, a key called partition key which is usually a table column or an expression, and a partitioning method needs to be defined. There are three types of inbuilt partitioning methods provided by Postgres:

- **Range Partitioning:** This method is done by partitioning based on a range of values. This method is most commonly used upon date fields to get monthly, weekly or yearly data. In the case of corner cases like value belonging to the end of the range, for example: if the range of partition 1 is

10-20 and the range of partition 2 is 20-30, and the given value is 10, then 10 belongs to the second partition and not the first.

- **List Partitioning:** This method is used to partition based on a list of known values. Most commonly used when we have a key with a categorical value. For example, getting sales data based on regions divided as countries, cities, or states.
- **Hash Partitioning:** This method utilizes a hash function upon the partition key. This is done when there are no specific requirements for data division and is used to access data individually. For example, you want to access data based on a specific product, then using hash partition would result in the dataset that we require.

The type of partition key and the type of method used for partitioning determines how positive the performance and the level of manageability of the partitioned table are.

## 48. Define tokens in PostgreSQL?

A token in PostgreSQL is either a keyword, identifier, literal, constant, quotes identifier, or any symbol that has a distinctive personality. They may or may not be separated using a space, newline or a tab. If the tokens are keywords, they are usually commands with useful meanings. Tokens are known as building blocks of any PostgreSQL code.

## 49. What is the importance of the TRUNCATE statement?

`TRUNCATE TABLE name_of_table` statement removes the data efficiently and quickly from the table.

The truncate statement can also be used to reset values of the identity columns along with data cleanup as shown below:

```
TRUNCATE TABLE name_of_table  
RESTART IDENTITY;
```

We can also use the statement for removing data from multiple tables all at once by mentioning the table names separated by comma as shown below:

```
TRUNCATE TABLE  
table_1,  
table_2,  
table_3;
```

## 50. What is the capacity of a table in PostgreSQL?

The maximum size of PostgreSQL is 32TB.

## 51. Define sequence.

A sequence is a schema-bound, user-defined object which aids to generate a sequence of integers. This is most commonly used to generate values to identity columns in a table. We can create a sequence by using the `CREATE SEQUENCE` statement as shown below:

```
CREATE SEQUENCE serial_num START 100;
```

To get the next number 101 from the sequence, we use the `nextval()` method as shown below:

```
SELECT nextval('serial_num');
```

We can also use this sequence while inserting new records using the `INSERT` command:

```
INSERT INTO ib_table_name VALUES (nextval('serial_num'), 'interviewbit');
```

## 52. What are string constants in PostgreSQL?

They are character sequences bound within single quotes. These are using during data insertion or updation to characters in the database.

There are special string constants that are quoted in dollars.

Syntax: `$tag$<string_constant>$tag$` The tag in the constant is optional and when we are not specifying the tag, the constant is called a double-dollar string literal.

## 53. How can you get a list of all databases in PostgreSQL?

This can be done by using the command `\l` -> backslash followed by the lower-case letter L.

## 54. How can you delete a database in PostgreSQL?

This can be done by using the `DROP DATABASE` command as shown in the syntax below:

```
DROP DATABASE database_name;
```

If the database has been deleted successfully, then the following message would be shown:

```
DROP DATABASE
```

## 55. What are ACID properties? Is PostgreSQL compliant with ACID?

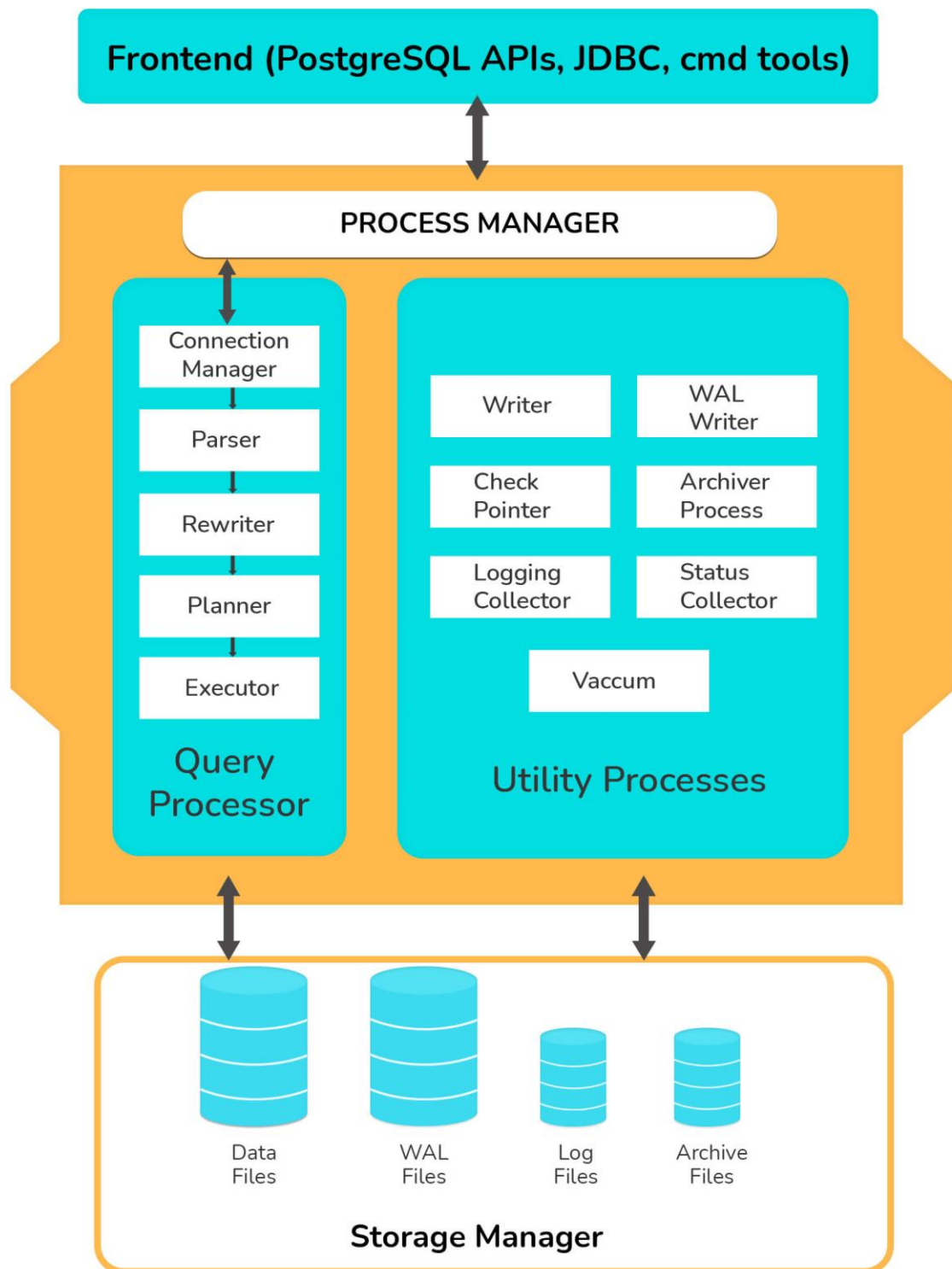
ACID stands for Atomicity, Consistency, Isolation, Durability. They are database transaction properties which are used for guaranteeing data validity in case of errors and failures.

- **Atomicity:** This property ensures that the transaction is completed in all-or-nothing way.
- **Consistency:** This ensures that updates made to the database is valid and follows rules and restrictions.
- **Isolation:** This property ensures integrity of transaction that are visible to all other transactions.
- **Durability:** This property ensures that the committed transactions are stored permanently in the database.

PostgreSQL is compliant with ACID properties.

## 56. Can you explain the architecture of PostgreSQL?

- The architecture of PostgreSQL follows the client-server model.
- The server side comprises of background process manager, query processor, utilities and shared memory space which work together to build PostgreSQL's instance that has access to the data. The client application does the task of connecting to this instance and requests data processing to the services. The client can either be GUI (Graphical User Interface) or a web application. The most commonly used client for PostgreSQL is pgAdmin.



57. What do you understand by multi-version concurrency control?

MVCC or Multi-version concurrency control is used for avoiding unnecessary database locks when 2 or more requests try to access or modify the data at the same time. This ensures that the time lag for a user to log in to the database is avoided. The transactions are recorded when anyone tries to access the content.

For more information regarding this, you can refer [here](#).

## 58. What do you understand by command enable-debug?

The command enable-debug is used for enabling the compilation of all libraries and applications. When this is enabled, the system processes get hindered and generally also increases the size of the binary file. Hence, it is not recommended to switch this on in the production environment. This is most commonly used by developers to debug the bugs in their scripts and help them spot the issues. For more information regarding how to debug, you can refer [here](#).

## 59. How do you check the rows affected as part of previous transactions?

SQL standards state that the following three phenomena should be prevented whilst concurrent transactions. SQL standards define 4 levels of transaction isolations to deal with these phenomena.

- **Dirty reads:** If a transaction reads data that is written due to concurrent uncommitted transaction, these reads are called dirty reads.
- **Phantom reads:** This occurs when two same queries when executed separately return different rows. For example, if transaction A retrieves some set of rows matching search criteria. Assume another transaction B retrieves new rows in addition to the rows obtained earlier for the same search criteria. The results are different.
- **Non-repeatable reads:** This occurs when a transaction tries to read the same row multiple times and gets different values each time due to concurrency. This happens when another transaction updates that data and our current transaction fetches that updated data, resulting in different values.

To tackle these, there are 4 standard isolation levels defined by SQL standards. They are as follows:

- **Read Uncommitted** – The lowest level of the isolations. Here, the transactions are not isolated and can read data that are not committed by other transactions resulting in dirty reads.
- **Read Committed** – This level ensures that the data read is committed at any instant of read time. Hence, dirty reads are avoided here. This level makes

use of read/write lock on the current rows which prevents read/write/update/delete of that row when the current transaction is being operated on.

- **Repeatable Read** – The most restrictive level of isolation. This holds read and write locks for all rows it operates on. Due to this, non-repeatable reads are avoided as other transactions cannot read, write, update or delete the rows.
- **Serializable** – The highest of all isolation levels. This guarantees that the execution is serializable where execution of any concurrent operations are guaranteed to be appeared as executing serially.

The following table clearly explains which type of unwanted reads the levels avoid:

Isolation levels	Dirty Reads	Phantom Reads	Non-repeatable reads
Read Uncommitted	Might occur	Might occur	Might occur
Read Committed	Won't occur	Might occur	Might occur
Repeatable Read	Won't occur	Might occur	Won't occur
Serializable	Won't occur	Won't occur	Won't occur

## 60. What can you tell about WAL (Write Ahead Logging)?

Write Ahead Logging is a feature that increases the database reliability by logging changes **before** any changes are done to the database. This ensures that we have enough information when a database crash occurs by helping to pinpoint to what point the work has been complete and gives a starting point from the point where it was discontinued.

For more information, you can refer [here](#).

## 61. What is the main disadvantage of deleting data from an existing table using the DROP TABLE command?

`DROP TABLE` command deletes complete data from the table along with removing the complete table structure too. In case our requirement entails just remove the data, then we would need to recreate the table to store data in it. In such cases, it is advised to use the `TRUNCATE` command.

## 62. How do you perform case-insensitive searches using regular expressions in PostgreSQL?

To perform case insensitive matches using a regular expression, we can use POSIX `(~*)` expression from pattern matching operators. For example:

```
'interviewbit' ~* '.*INTervIewBit.*'
```

### 63. How will you take backup of the database in PostgreSQL?

We can achieve this by using the `pg_dump` tool for dumping all object contents in the database into a single file. The steps are as follows:

**Step 1:** Navigate to the bin folder of the PostgreSQL installation path.

```
C:\>cd C:\Program Files\PostgreSQL\10.0\bin
```

**Step 2:** Execute `pg_dump` program to take the dump of data to a .tar folder as shown below:

```
pg_dump -U postgres -W -F t sample_data >  
C:\Users\admin\pgbackup\sample_data.tar
```

The database dump will be stored in the `sample_data.tar` file on the location specified.

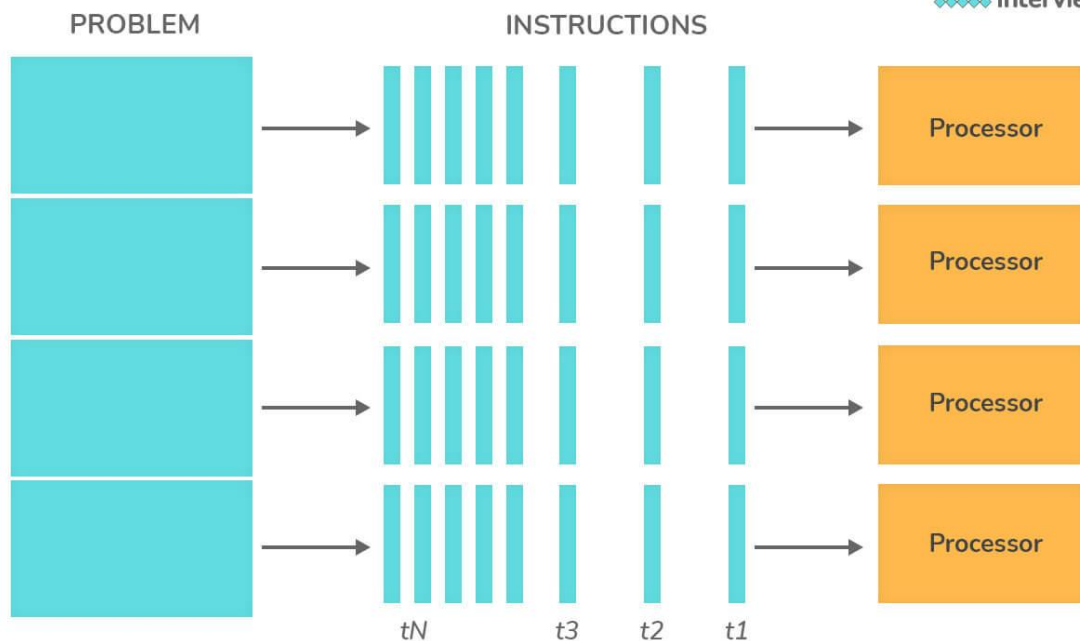
### 64. Does PostgreSQL support full text search?

Full-Text Search is the method of searching single or collection of documents stored on a computer in a full-text based database. This is mostly supported in advanced database systems like SOLR or ElasticSearch. However, the feature is present but is pretty basic in PostgreSQL.

### 65. What are parallel queries in PostgreSQL?

Parallel Queries support is a feature provided in PostgreSQL for devising query plans capable of exploiting multiple CPU processors to execute the queries faster.





## 66. Differentiate between commit and checkpoint.

The commit action ensures that the data consistency of the transaction is maintained and it ends the current transaction in the section. Commit adds a new record in the log that describes the COMMIT to the memory. Whereas, a checkpoint is used for writing all changes that were committed to disk up to SCN which would be kept in datafile headers and control files.