SQL Interview Questions

Share with:

SQL Interv...

Are you preparing for your SQL developer interview?

Then you have come to the right place.

This guide will help you to brush up on your SQL skills, regain your confidence and be job-ready!

Here, you will find a collection of real-world Interview questions asked in companies like Google, Oracle, Amazon, and Microsoft, etc. Each question comes with a perfectly written answer inline, saving your interview preparation time.
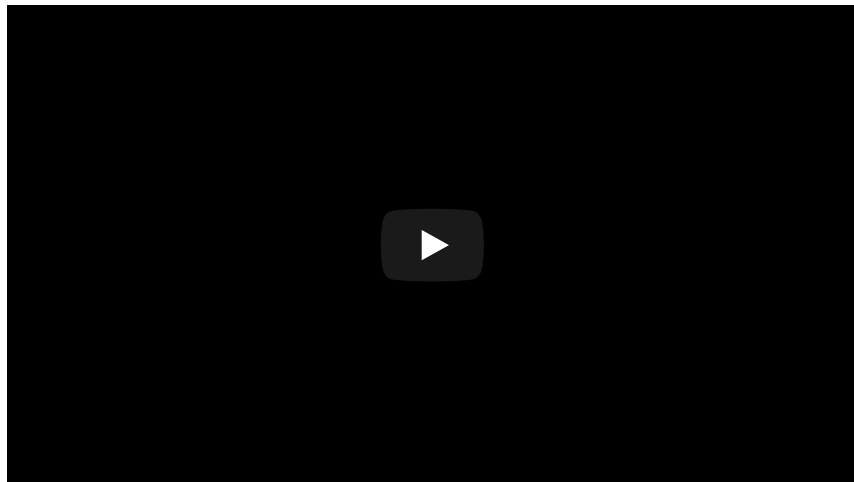
It also covers practice problems to help you understand the basic concepts of SQL.

We've divided this article into the following sections:

SQL Interview Questions

PostgreSQL Interview Questions

In the end, multiple-choice questions are provided to test your understanding.

SQL Interview Questions

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.

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:

```sql
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.

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

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

Some common SQL clauses used in conjuction 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**).

```sql
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.

```sql
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

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)

- **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)

Books Table (2nd Normal Form)

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

```
WX -> Y    [W and X together determine Y]
XY -> Z    [X and 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)

Books Table (3rd Normal Form)

Salutations Table (3rd Normal Form)

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 –> 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.

```sql
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.

```sql
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.

```sql
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 Ouput 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 processer, 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 tries 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:

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.

Conclusion:

SQL is a language for the database. It has a vast scope and robust capability of creating and manipulating a variety of database objects using commands like CREATE, ALTER, DROP, etc, and also in loading the database objects using commands like INSERT. It also provides options for Data Manipulation using commands like DELETE, TRUNCATE and also does effective retrieval of data using cursor commands like FETCH, SELECT, etc. There are many such commands which provide a large amount of control to the programmer to interact with the database in an efficient way without wasting many resources. The popularity of SQL has grown so much that almost every programmer relies on this to implement their application's storage functionalities thereby making it an exciting language to learn. Learning this provides the developer a benefit of understanding the data structures used for storing the organization's data and giving an additional level of control and in-depth understanding of the application.

PostgreSQL being an open-source database system having extremely robust and sophisticated ACID, Indexing, and Transaction supports has found widespread popularity among the developer community.

References and Resources:

PostgreSQL Download

PostgreSQL Tutorial

Multiple Choice Questions

1. Which statement is true for a PRIMARY KEY constraint?

○   Primary key defines a realtionship between two tables.

○   A table in SQL must have a primary key associated with it to uniquely identify its records.

○   A table in SQL is indexed by default based on its primary key.

○   Primary key may or may not be unique but can be comprised of multiple fields.

2. Which statement is false for a FOREIGN KEY constraint?

○   Foreign key defines a relationship between two tables.

○   Foreign Key is automatically created when two tables are joined.

○   Foreign Key uniquely identifies all the records in the referenced table.

○   Foreign key may or may not be unique but can be comprised of multiple fields.

3. What is a Query?

○   A SELECT or UPDATE statement in SQL.

○   A request for data from a table in the database.

○   A request to input data from the user.

○   A request for data from single or multiple tables in the database.

4. What does SQL stand for?

○   Structured Question Language

○ Strong Query Language

○ Structured Query Language

○ Strong Question Language

5. Which statement is used to update data in the database?

○ MODIFY

○ UPDATE

○ ALTER TABLE

○ SAVE AS

6. Query to select all records with "bar" in their name?

○ SELECT * FROM people WHERE name = "%bar%";

○ SELECT * FROM people WHERE name LIKE "%bar%";

○ SELECT * FROM people WHERE name IN ("bar");

○ SELECT * FROM people WHERE name = "_bar_"

7. Which statement is false for the ORDER BY statement?

○ Requires a ASC or DESC keyword explicitly to sort the result set.

○ Sorts the result set in descending order using the DESC keyword.

○ Can sort based on multiple columns

○ None of the above.

8. SQL query used to fetch unique values from a field?

○ SELECT UNIQUE column_name FROM table_name;

○ SELECT DISTINCT column_name FROM table_name;

○ SELECT column_name FROM table_name WHERE COUNT(column_name) = 1;

○ SELECT UNIQUE column_name FROM table_name WHERE COUNT(column_name) = 1;

9. What is the main advantage of a clustered index over a non-clustered index?

○ It is easier to create and manipulate.

○ It requires extra memory but allows for speedy retrieval of records.

○ It does not require additional memory and allows for speedy retrieval of records.

○ None of the above.

10. Normalization which has neither composite values nor partial dependencies?

○ Second Normal Formal

○ Third Normal Form

○ Boyce-Codd Normal Form

○ All of the above

11. An SQL query to delete a table from the database and memory while keeping the structure of the table intact?

○ DROP TABLE table_name;

○ DROP FROM TABLE table_name;

○ DELETE FROM TABLE table_name;

○ TRUNCATE TABLE table_name;

12. Which of the following is known as a virtual table in SQL?

○ SELF JOIN

○ INNER JOIN

○ VIEW

○ NONE

13. What is the name of the component that requests data to the PostgreSQL server?

○ Client

○ Thin Client

○ Workstation

○ Interface

14. What statement is used for adding data to PostgreSQL?

○ UPDATE

○ ADD

○ APPEND

○ INSERT

15. What is the order of results shown by default if the ASC or DESC parameter is not specified with the ORDER BY command?

○ Results are shown in descending order

○ Results are shown in ascending order

○ Results display is random

○ Results are shown in ascending and descending order alternately.

16. Which command is used to tell PostgreSQL to make all changes made to the database permanent?

○ Submit

○ Execute

○ Apply

○ Commit

17. What is a pre-requisite for creating a database in PostgreSQL?To create a database in PostgreSQL, you must have the special CREATEDB privilege or

○ Super user privilege or CREATEDB privilege

○ Admin privilege

○ CREATEDBL privilege and Super user privilege

○ Just run the script

18. What command is used for restoring the backup of PostgreSQL which was created using pg_dump?

○ psql -R db_dump.psql db_name

○ psql -r db_dump.psql db_name

○ psql -f db_dump.psql db_name

○ psql -F db_dump.psql db_name

19. What allows us to define how various tables are related to each other formally in a database?

○ Views

○ Foreign Key Constraints

○ Primary Key Constraints

○ Database manager

20. What languages are supported by PostgreSQL?

○ PL/pgSQL, PL/Tcl, PL/Perl and PL/Python

○ PL/pgSQL, PL/Pcl, PL/Ruby and PL/Java

○ PL/Perl, PL/Dcl, PL/Dtl and PL/Dml

○ Only SQL

Practice Questions

Programming

Scripting

System Design

Databases

Puzzle

Fast Track Courses

Python

Java

C++

Javascript

Online Interviewbit Compilers

Online C Compiler   Online C++ Compiler

Online Java Compiler   Online Javascript Compiler

Online Python Compiler

Interview Preparation

System Design Questions   Google Interview Questions

Facebook Interview Questions   Amazon Interview Questions

Microsoft Interview Questions   Javascript Interview Questions

jQuery Interview Questions   Angular Interview Questions

Data Structure Interview Questions   Directi Interview Questions

Yahoo Interview Questions   LinkedIn Interview Questions

eBay Interview Questions   Flipkart Interview Questions

Oracle Interview Questions   Adobe Interview Questions

C Interview Questions   Php Interview Questions

C Sharp Interview Questions   Web Api Interview Questions

Hibernate Interview Questions   Node Js Interview Questions

Cpp Interview Questions   Oops Interview Questions

Devops Interview Questions   Machine Learning Interview Questions

Docker Interview Questions   Mysql Interview Questions

Css Interview Questions   Laravel Interview Questions

Asp Net Interview Questions   Django Interview Questions

Dot Net Interview Questions   Kubernetes Interview Questions

Operating System Interview Questions   React Native Interview Questions

Aws Interview Questions   Git Interview Questions

Java 8 Interview Questions   Mongodb Interview Questions

Dbms Interview Questions   Spring Boot Interview Questions

Power Bi Interview Questions   Pl Sql Interview Questions

Tableau Interview Questions   Linux Interview Questions

Ansible Interview Questions   Java Interview Questions

Jenkins Interview Questions   Agile Interview Questions

Networking Interview Questions   Azure Interview Questions

Software Testing Interview Questions   Automation Testing Interview Questions

Android Interview Questions   Hr Interview Questions

Rest Api Interview Questions   Unix Interview Questions

Multithreading Interview Questions   Typescript Interview Questions

Angular 8 Interview Questions   Salesforce Interview Questions

Spark Interview Questions   Bootstrap Interview Questions

Web Services Interview Questions   Jsp Interview Questions

Or use email

1 Million +

Strong Tech Community

500 +

Questions to Practice

100 +

Expert Interview Guides