SQL Interview Q_A

1. **What is Database?**

   A collection of data and holds this data in the form of tables.the database provides us the capability to access and manipulate this data.

2. **What are Tables records and Fields?**

   Tables holds the data in form of rows and columns it is like excel spreadsheet.
   Columns/ fields can be categorized as **vertical** and **rows/records** as **horizontal**.

3. **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.

4. **Types of databases?**

   a.Relational database Management System (RDBMS): rows and columns and also have relation between them.
   Ex: MySQL, SQL Server, Postresql,sqlite,MariaDB,oracle, IBM DB2

   b.No-Sql database: key value ,document, graph
   Ex: Hbase,MangoDB,Cassandra

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

   SQL is a **structured query language** is used to query a **Relational database**.
   MySQL is a **Relational database management system**, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.

6. **Data Type?**

   Int – Numeric
   Varchar – Strings up to 255 characters.

7. **CRUD operations?**

   CREATE – creating records using insert statements.
   READ- reading records using select statements.

UPDATE- updating records using update statements.
DELETE- deleting records using delete statements.

8. **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 altered 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.

9. **What is a Primary Key?**

- Purpose of primary key is to make sure that uniquely identifies each row/record in a table.
- A table can have only one primary key.
- For primary key null values are not allowed and repeated values not allowed (only unique values allowed).
- **Composite primary key** can have more than one or more column/field.
- Auto increment key is best for primarykey

10. **What is a UNIQUE Key?**

- Purpose of unique key is to make sure that values in a column don't duplicate.
- We can have multiple unique keys in a table.
- Unique key can hold null.
- My sql db can hold any number of nulls, some db's can hold only one null.

11. **Primary key Vs Unique key?**

we can have only one Primary key and multiple unique keys in a table.

12. **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);
```

13. 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 le 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 le 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 le or right table.

```
SELECT *
```

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

14. **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;
    ```

15. **What is a Cross-Join?**

    Cross join can be defined as a cartesian product of the two tables included in the join. The table aer 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;
    ```

16. **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.

17. **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.

18. **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.

19. **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;
```

20. **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.

21. 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;
```

22. 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).

```
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;
```

23. 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

24. 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.

25. **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.

26. **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;
    ```

27. **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.

28. **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;
    ```

29. **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.

30. 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.

31. 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 oen 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.

32. 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.

33. What is OLTP?

OLTP stands for Online Transaction Processing, is a class of soware 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 oen 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

34. What are the differences between OLTP and OLAP?

OLTP stands for Online Transaction Processing, is a class of soware applications capable of supporting transaction-oriented programs. An important attribute of an OLTP system is its ability to maintain concurrency. OLTP systems oen 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 soware programs that are characterized by the relatively low frequency of online transactions. Queries are oen 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.

35. 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;
```

36. 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 aer 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%'
```

37. 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 '____'
```

38. What is the difference between NoSQL and RDBMS?

RDBMS:-

In RDBMS it is hard to handle different kind of data, if not possible to store in relational database. The development also makes tricky and takes more time.

It cannot scale as much as can globally, and it is difficult maintain large amount of data in a single system.

Large teams will tie up for the long period of time. This makes us the cost of the project will increase drastically.

NOSQL:-

It can easily handle the different kinds of data. With providing all the features to process that data.
It can easily scale up in globally according to your audience growth on particular area.
It can easily handle that large amount of data for long time.
The data is stored in a distributed environment with commodity hardware make your project maintenance low and accessing speed is more when compared to the traditional RDBMS.
The distribution system will be described by the CAP theorem, C-Consistency, A-Availability and P- Tolerance to network Partitioning.

39. **What is correlated and sub-queries in sql?**
In a situation where inner query and outer query want to execute on a common group by condition. Then we use correlated sub-queries

40. **How to get the second highest salary from the emp table ?**
Select * from employee
Where salary = (Select MAX(salary) from employee)
Where salary < (Select MAX(salary) from employee));
Or
Select * from employee order by sal limit 2 offset 1;

41. **How to write a query to find the duplicated records only ?**
select state from Area Group BY state Having count(*) > 1

42. **How to write a query on joining of two tables ?**
Select student.name , student_location.address from student
JOIN student_location ON student.name = student_address.name.

43. **Write a query to get the second topmost salary ?**
select max(salary) from salary where salary < (select max(salary) from salary);
SELECT name, salary FROM employees
WHERE salary = (SELECT MAX(salary) FROM employees WHERE salary < (SELECT MAX(salary) FROM employees));

44. **How do you get the best revenue in a particular location ?Without using primary key. How you find duplicate fields in SQL ?**
By using Unique Constraint

45. **How can u load Text file data into MySQL database? Tell the script query?**
LOAD DATA LOCAL INFILE '<PATH_OF_THE_FILE>' INTO TABLE <TABLE_NAME>;
Ex:- LOAD DATA LOCAL INFILE '/home/hadoop/mysql/inputs' INTO TABLE mytable;

46. **I am having two table while importing the data from source table to target table operation was cancelled write a query to know how many records are imported ?**
Ans:- mysql> **select * from student2;**

```
+----------+------+--------+------+
| name | id | course | year |
+----------+------+--------+------+
| Avinash | 14 | Hadoop | 2015 |
| Goutham | 22 | Spark | 2016 |
| Mahi | 16 | Hadoop | 2015 |
| Manoj | 24 | Spark | 2016 |
| Shiva | 23 | Spark | 2016 |

| Srinivas | 15 | Hadoop | 2015 |
+----------+------+--------+------+
6 rows in set (0.00 sec)
```

mysql> **select * from student21;**

```
+---------+------+--------+------+
| name | id | course | year |
+---------+------+--------+------+
| Goutham | 22 | Spark | 2016 |
| Manoj | 24 | Spark | 2016 |
| Shiva | 23 | Spark | 2016 |
+---------+------+--------+------+
3 rows in set (0.00 sec)
```

---> We can perform the count(*) operation but by that procedure we will get the number of the records but not the records information.

mysql> **select * from student2 where id NOT IN(select id from student21);** --> Rows that are not there in target table..

```
+----------+------+--------+------+
| name | id | course | year |
+----------+------+--------+------+
| Avinash | 14 | Hadoop | 2015 |
| Mahi | 16 | Hadoop | 2015 |
| Srinivas | 15 | Hadoop | 2015 |
+----------+------+--------+------+
3 rows in set (0.00 sec)
```

mysql**> select * from student2 where id IN(select id from student21);** --> Rows that are there in the target table..

```
+---------+------+--------+------+
| name | id | course | year |
+---------+------+--------+------+
| Goutham | 22 | Spark | 2016 |
| Manoj | 24 | Spark | 2016 |
| Shiva | 23 | Spark | 2016 |
+---------+------+--------+------+
3 rows in set (0.01 sec)
```

47. to get the last row of a SQL-Database use this sql string:

SELECT * FROM TableName WHERE id=(SELECT max(id) FROM TableName);

48. Selecting ODD or EVEN rows from a table

Even:

select * from emp where mod (rn, 2) = 0;

select empno, ename, sal, rownum as rn from emp order by empno where mod (rn, 2) = 0;

Odd:

select * from emp where mod (rn, 2) != 0;

select empno, ename, sal, rownum as rn from emp order by empno where mod (rn, 2) != 0;

49. **Finding duplicate values in a SQL table**

```
SELECT email, COUNT(email)
FROM users
GROUP BY email
HAVING COUNT(email) > 1
```

50. **Difference between Where and Having Clause ?**

Having:-

HAVING is used to check conditions after the aggregation takes place.

HAVING clause is the additional filter to the where clause..

Ex:- select City, CNT=Count(1) From Address Where State = 'MA' Group By City Having Count(1)>5;

Where:-

WHERE clause doesn't work with the aggregations like SUM,AVG,MIN and MAX etc.,

WHERE is used before the aggregation takes place.

Ex:- select City, CNT=Count(1) From Address Where State = 'MA' Group By City;

51. **What is the difference between UNION and UNION ALL?**

The difference between UNION and UNION ALL is that UNION will omit duplicate records whereas UNION ALL will include duplicate records