SQL Interview Q_A

1. **What is Database?**

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

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.

3. **Types of databases?**

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

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

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

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

5. **DDL and DML?**

   DDL- data definition language – deals with table structure.
   - **Create**
   - **Drop**
   - **Alter**
   - **truncate**

   DML- data manipulation language – deals with data directly.
   - **Insert**
   - **Update**
   - **Delete**
   - **Merge**

6. **Data Types?**

   There are three main types of SQL data types available in any RDBMS. They are listed below.

   1. **String**
      - **CHAR -**The maximum length of 8000 characters. (Fixed-Length non-Unicode Characters)
      - **VARCHAR -**The maximum length of 8000 characters. (Variable-Length non-Unicode Characters)
      - **VARCHAR(MAX) -**The maximum length of 231 characters (SQL Server 2005 only). (Variable Length non-Unicode data)
      - **TEXT -**The maximum length of 2,127,483,647 characters (Variable Length non-Unicode data)
   2. **Numeric**
      - **INT -** You can specify a width of up to 11 digits.
      - **BIGINT-** You can specify a width of up to 20 digits.
      - **FLOAT**
   3. **Date and Time**
      - **DATE -** A data type is used to store the data of date in a record.
      - **TIME -** A data type is used to store the data of time in a record.
      - **DATETIME-** A data type is used to store both the data, date, and time in the record.

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 deleted statements.

8. **What are Tables, rows/records, and Columns/ Fields/Attributes?**

   - A **table** is a physical table.
   - It consumes physical space in a database to store data.
   - **Table** is independent.
   - Tables hold the data in the form of rows and columns like excel spreadsheet.
   - **Columns/ Fields/Attributes** can be categorized as **vertical.**
   - **rows/records** as **horizontal**.

9. What are Temporary Table?

A Temporary Table is also a physical table but used to store data temporarily. Temporary tables are typically used to store intermediate results or to break down complex queries into simpler ones. They are often created and used within a single session and automatically dropped when the session ends.

10. What is a View?

- **View** is a virtual table.
- **View** does not store data in the database, it will store only the query expression.
- When view query executed it dynamically retrieves data based on a pre-defined "SELECT query" statement.
- It can fetch data from one or more tables.
- **View** is dependent on one or more tables

11. Materialized Views in SQL?

- Materialized views are also virtual table, but the result of the query is stored physically in the database.
- It can be accessed faster, and the performance of the materialized views is better than normal views. This is because the data is stored in the database.
- Sometimes, materialized views are also called "indexed views"
- Materialized Views are used when data is to be accessed frequently and data in table does not get updated on frequent basis.
- If the Data in tables are updated, we need to refresh Materialized Views.
- It consumes more space in database than normal view.

12. What are Constraints in SQL?

SQL constraints are rules applied to **columns to** limit the type of data that can be **inserted**, **updated**, or **deleted**. These rules ensure the data is valid, consistent, and as per the business logic.

1. **NOT NULL** - constraint ensures that a column cannot contain NULL values.
2. **CHECK** - constraint allows us to specify a condition that data must satisfy before it is inserted into the table..
3. **DEFAULT** - constraint provides a default value for a column when no value is specified during insertion.
4. **UNIQUE KEY**
5. **PRIMARY KEY**
6. **FOREIGN KEY**

13. What is a Primary Key?

- It uniquely identifies each row in a table.
- For primary key null and repeated values not allowed. This constraint is a combination of the NOT NULL and UNIQUE constraints.
- A table can have only one primary key.
- **Composite primary key** can have more than one or more column/field.
- Auto increment key is best for primary key.

14. What is a UNIQUE Key?

- constraint ensures that all values in a column are distinct across all rows in a table.
- 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.

15. Primary key Vs Unique key?

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

16. What is a Foreign Key?

- constraint links a column in one table to the primary key in another table.
- The foreign key constraint is used to prevent actions that would destroy links between two tables.
- The table with foreign key is called child table.
- The table with primary key is called parent/reference table.

17. What is the SELECT statement?

The SELECT statement is the most used command in Structured Query Language. It is used to access the records from one or more database tables and views.

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

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

The sql clauses can help filter out the data according to the users' needs. The main clauses are:
SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, INSERT, UPDATE, DELETE, and JOIN

19. What is a Query?

A query is a request for data/information from a table in a database. A database query can be either a select query or an action query.

/* select query */

```
SELECT FNAME, LNAME
FROM MYDB.STUDENTS
WHERE STUDENT_ID = 1;
```

/* action query */

```
UPDATE MYDB.STUDENTS
SET FNAME = 'CAPTAIN', LNAME = 'AMERICA'
WHERE STUDENT_ID = 1;
```

20. What is the difference between Drop, DELETE and TRUNCATE statements?

Drop:

- It's DDL command.
- 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.

  EX: DROP TABLE Candidates;

Delete:

- It's DML command.
- It's used to delete individual records and we can also delete all records.
- It's Drops the records one after another, so it takes more time than truncate, to drop all table records.

  EX: DELETE FROM Candidates WHERE CandidateId > 1000;

Truncate:

- It's DDL command.
- It also removes all records.
- truncate internally drop the table and recreates it.
- Truncate is more efficient than delete.

  EX: TRUNCATE TABLE Candidates;

21. What is a Join? List its different types.

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:
Only matching records from both left and right tables are considered. Nonmatching records are discarded.

Syntax:

```
SELECT *  FROM table1
INNER JOIN table2
ON table1.Column_Name = table2.Column_Name;
```

LEFT (OUTER) JOIN:
All matching records from the left and right table are considered.
And all non-matching records in the left table which do not have match in the right table will be padded with nulls.

Syntax:

```
SELECT *  FROM TABLE1
LEFT JOIN TABLE2
ON TABLE1.COLUMN_NAME = TABLE2.COLUMN_NAME;
```

RIGHT (OUTER) JOIN:
All matching records from the left and right table are considered.
And all non-matching records in right table which does not match the right table will be padded with the with nulls.

Syntax:

```
SELECT *  FROM TABLE1
RIGHT JOIN TABLE2
ON TABLE1.COLUMN_NAME = TABLE2.COLUMN_NAME;
```

FULL (OUTER) JOIN:
All matching and non-matching records from both left and right join are considered.

Syntax:

```
SELECT *  FROM TABLE1
FULL OUTER JOIN TABLE2
ON TABLE1.COLUMN_NAME = TABLE2.COLUMN_NAME;
```

22. What is a Self-Join?

A table is joined to itself based on some relation between its own columns.

23. What is a Cross-Join?

Cross join is a Cartesian product where number of rows in the first table multiplied by number of rows in the second table.

If a WHERE clause is used in cross join, then the query will work like an INNER JOIN.

24. Natural Join (EQUI JOIN) in SQL?

joins two tables based on the same column name and datatypes. The resulting table will contain all the columns of both tables but keep only one common column.

Syntax:

```
SELECT * FROM table1 NATURAL JOIN table2;
```

25. Difference between Where and Having Clause?

Having: -

- HAVING Clause is used to filter records from the group by.
- HAVING Clause is used after GROUP BY Clause.
- HAVING Clause cannot be used without GROUP BY Clause
- HAVING Clause can only be used with SELECT statement.
- HAVING Clause can contain aggregations like SUM, AVG, MIN and MAX, COUNT etc.

Where: -

- WHERE Clause is used to filter the records from the table.
- WHERE Clause is used before GROUP BY Clause.
- WHERE Clause can be used with SELECT, UPDATE, DELETE statement.
- WHERE Clause is used with single row function like UPPER, LOWER etc.

We can use where and having in same query. Where is more performant than having.

26. What is the difference between order by and group by?

ORDER BY

- The ORDER BY clause is used to sort the data in ascending or descending order.
- ASC denotes ascending order, while DESC denotes descending order.
- ORDER BY guarantees global ordering.

GROUP BY

- The GROUP BY clause is used to organize data that have the same column values.
- Group by clause in collaboration with aggregate functions like SUM, AVG, MIN, MAX, and COUNT.

27. Differentiate between Rank, Row Number and Dense Rank?

Also known as window functions. These are used to give numbering based on one or more condition.

If there are no duplicates then row number, rank, dense rank leads to similar results. The only difference comes if there are duplicates.

row number:

- It allows Numbering to duplicate records.
- When we use row number, we should use order by clause.
- We can also use partition by clause is optional –then row number starts from 1 for each partition.

rank:

- for duplicates same rank is assigned and for the next entry it skips the next rank. (There are holes between ranks).
- Order by clause is mandatory.
- partition by clause is optional.

dense rank:

- for duplicates same rank assigned and next entry starts from next rank (there are no holes between ranks).
- Order by clause is mandatory.
- partition by clause is optional.

Scenario: whenever you don't have duplicates use row number.

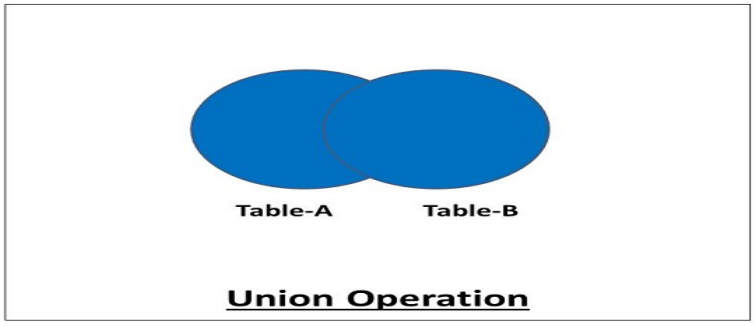
28. What is the difference between JOIN and UNION?

JOIN: Used to combine rows from multiple tables based on a related column between them. The resulting table will have more columns.

UNION: Used to combine two or more SELECT query results into single result. The resulting table will have more distinct rows.

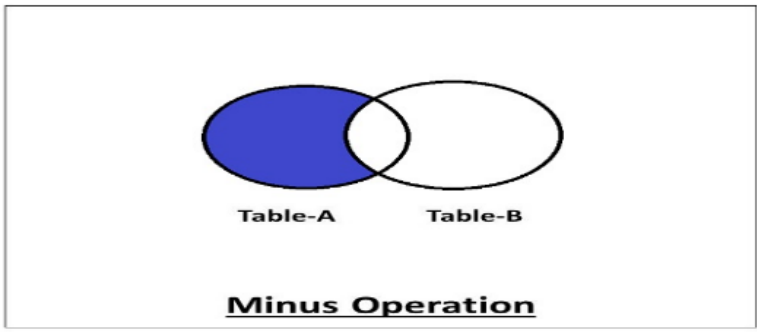
29. What is UNION, MINUS and INTERSECT commands?

1. **UNION:**
    - Used to combine the two or more select query results into single result.
    - It automatically removes duplicate rows from the results.

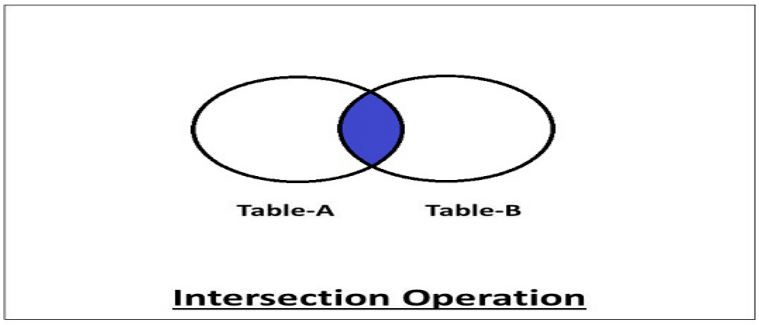


2. **UNION ALL:**
    - Used to combine the two or more select query results into single result.
    - UNION ALL will include duplicate records.

3. **MINUS:** returns the rows that are present only in the first select statement but not in the second select statement.



4. **INTERSECT:** returns only rows which are present in first select statement and second select statement.



30. What is a Slowly Changing Dimension?

A Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data over time in a data warehouse. It is considered and implemented as one of the most critical ETL tasks in tracking the history of dimension records.

- **SCD Type 0:** No changes; the dimensions are static.

- **SCD Type 1:**
  SCD type 1 methodology is used when there is no need to store historical data in the dimension table. This method overwrites the old data in the dimension table with the new data. It is used to correct data errors in the dimension. The advantage of type1 is ease of maintenance and less space occupied. The disadvantage is that there is no historical data kept in the data warehouse.

  - **Example,** i have the customer table with the below data.

    surrogate_key customer_id customer_name Location
    ---
    1 101 Marspton Illions

  - Here the customer's name is misspelt. It should be Marston instead of Marspton. If you use type1 method, it just simply overwrites the data. The data in the updated table will be.

    surrogate_key customer_id customer_name Location
    ---
    1 101 Marston Illions

- **SCD Type 2**:
  SCD type 2 stores the entire history the data in the dimension table. new and historical results captured in separate rows.With type 2 we can store unlimited

history in the dimension table. In type 2, you can store the data in three different ways. They are

- Versioning
- Flagging
- Effective Date and End date

- **SCD Type 2 Versioning**:
  In versioning method, a sequence number is used to represent the change. The latest sequence number always represents the current row and the previous sequence numbers represents the past data.
  - As an example, let's use the same example of customer who changes the location. Initially the customer is in Illions location and the data in dimension table will look as.

surrogate_key	customer_id	customer_name	Location	Version
1	101	Marston	Illions	1

  - The customer moves from Illions to Seattle and the version number will be incremented. The dimension table will look as

surrogate_key	customer_id	customer_name	Location	Version
1	101	Marston	Illions	1
2	101	Marston	Seattle	2

  - Now again if the customer is moved to another location, a new record will be inserted into the dimension table with the next version number.

- **SCD Type 2 Flagging**:
  In flagging method, a flag column is created in the dimension table. The current record will have the flag value as 1 and the previous records will have the flag as 0.
  - Now for the first time, the customer dimension will look as.

surrogate_key	customer_id	customer_name	Location	flag
1	101	Marston	Illions	1

  - Now when the customer moves to a new location, the old records will be updated with flag value as 0 and the latest record will have the flag value as

surrogate_key	customer_id	customer_name	Location	flag

1	101	Marston	Illions	0
2	101	Marston	Seattle	1

- **SCD Type 2 Effective Date and End date**:
  In Effective Date method, the period of the change is tracked using the start_date and end_date columns in the dimension table.
  - customer dimension will look as.

surrogate_key	customer_id	customer_name	Location	Start_date	End_date
1	101	Marston	Illions	01-Mar-2010	20-Fdb-2011
2	101	Marston	Seattle	21-Feb-2011	NULL

- The NULL in the End_Date indicates the current version of the data and the remaining records indicate the past data. Top of Form
  Bottom of Form

- **SCD Type 3**:
  In type 3 method, only the current status and previous status of the row is maintained in the table. To track these changes two separate columns are created in the table. The type 3 method will have limited history, and it depends on the number of columns you create.

  - **Example,** i have the customer table with the below data.

surrogate_key	customer_id	customer_name	Current_Location	previous_location
1	101	Marston	Illions	NULL

  - Let say, the customer moves from Illions to Seattle and the updated table will look as

surrogate_key	customer_id	customer_name	Current_Location	previous_location
1	101	Marston	Seattle	Illions

  - Now again if the customer moves from seattle to NewYork, then the updated table will be

surrogate_key	customer_id	customer_name	Current_Location	previous_location
1	101	Marston	NewYork	Seattle

- **SCD Type 4:** The historical data is maintained in a separate table. The current table maintains a single row and all history is kept in a separate table.

- **SCD Type 6:** Hybrid (Type 1 + Type 2 + Type 3). The table captures the historical rows and the current role maintains the current value and the previous value in two separate rows.

31. What are Dimension tables and fact tables?

Fact Tables:

- A fact table/ reality table is a primary table in a dimensional model.
- A Fact Table contains Measurements/facts and foreign key to dimension table.
- A fact is a piece of information with a specific numerical value.
- Facts are accompanied by dimensions.
- There are less attributes/columns than dimension table.
- There are more records than dimension table.
- Fact table forms a vertical table.
- It is used for analysis purpose and decision making.

Dimension tables:

- A dimension table contains dimensions of a fact.
- There are more attributes/columns than fact table.
- There are less records than the fact table.
- While dimension table forms a horizontal table.
- While the dimension table has a primary key, the table has a concatenated key.

32. What are Entities and Relationships?

Entity: An entity can be a real-world object. Each entity has some associated properties that provide it with an identity.

EX: college database, students, professors, workers, departments, and projects can be referred to as entities.

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

33. List the different types of relationships in SQL.

One-to-One – In this relationship 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 used relationship where a record in a table is associated with multiple records in the other table vice versa.

Many-to-Many - This is used in cases when multiple records 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.

34. What is Alias in SQL?

- It is a temporary name assigned to the table or table column for the purpose of a particular SQL query.
- In addition, is 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 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;
```

35. What is Pattern Matching in SQL?

Pattern matching is checking whether a specific sequence of characters or data exist within a dataset.

The LIKE operator is used with the WHERE clause. It enables users to use _ to match a single character and % to match an arbitrary number of characters.

1. Search a student in your database with first name beginning with the letter K:
   ```
   SELECT *
   FROM students
   WHERE first_name LIKE 'K%'
   ```

2. all students whose first name does not begin with K.
   ```
   SELECT *
   FROM students
   WHERE first_name NOT LIKE 'K%'
   ```

3. Search for a student in the database where he has a K in his first name.
   ```
   SELECT *
   FROM students
   WHERE first_name LIKE '%K%'
   ```

4. This query fetches all students with letter K at the third position in their first name.
   ```
   SELECT *
   FROM students
   WHERE first_name LIKE '__K%'
   ```

5. Matches first names with three or more letters.
   ```
   SELECT  *
   FROM students
   WHERE first_name LIKE '___%'
   ```

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

36. What are the differences between OLTP and OLAP?

OLTP:

- OLTP stands for Online Transaction Processing.
- 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.
- Several transactions per second act as an effective measure for such systems.
  EX: oracle databases

OLAP:

- OLAP stands for Online Analytical Processing.
- Queries are 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.
  EX: data warehouse

37. What are Aggregate and Scalar functions?

Aggregate functions are used with the GROUP BY and HAVING clauses of the SELECT statement. Following are the widely used SQL aggregate functions:

- AVG () - Calculates the meaning 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.

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).
- UPPER () - Converts a collection of string values to uppercase characters.
- LOWER () - Converts a collection of string values to lowercase characters.
- STR () - 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.
- COALESCE ()

38. 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.

39. What is an Index? Explain its different types.

- Index provides a quick lookup of data in a column or columns of a table.
- An Index is created on one or more columns of a table.
- This can significantly speed up the query execution time, especially for large tables.

- An index is a performance tuning method of allowing faster retrieval of records from the table.

There are three types of indexes:

- **Unique Index:** This indexing does not allow the field to have duplicate values if the column is unique indexed.Unique index can be applied automatically when primary key is defined.
- **Clustered Index:** This type of index reorders the physical order of the table and search based on the key values. Each table can have only one clustered index.
- **Non-Clustered Index:** Non-Clustered Index does not alter the physical order of the table and maintains logical order of data.

A table can have a single clustered index where it can have multiple non-clustered indexes.

```
CREATE INDEX index_name

ON table_name (column_1, column_2);

DROP INDEX index_name;
```

40. What is Data Integrity?

Data Integrity defines the accuracy and consistency of data stored in a database. It can also define integrity constraints to enforce business rules on the data when it is entered into the application or database.

41. 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. The outer query is called as main query, and inner query is called subquery. Subquery is always executed first, and the result of subquery is passed on to the main query.

There are two types of subqueries – Correlated and Non-Correlated.

- correlated subquery query is dependent query.
- non-Correlated sub query can be considered as independent query.

42. What are 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