SQL Interview Question with Answers

1. What is the SQL server query execution sequence?

○ FROM -> goes to Secondary files via primary file

○ WHERE -> applies filter condition (non-aggregate column) ○ SELECT -> dumps data in tempDB system database ○ GROUP BY -> groups data according to grouping predicate ○ HAVING -> applies filter condition (aggregate function) ○ ORDER BY -> sorts data ascending/descending

2. What is Normalization?

Step by step process to reduce the degree of data redundancy.

Breaking down one big flat table into multiple table based on normalization rules. Optimizing the memory but not in term of performance.

Normalization will get rid of insert, update and delete anomalies.

Normalization will improve the performance of the delta operation (aka. DML operation); UPDATE, INSERT, DELETE

Normalization will reduce the performance of the read operation; SELECT

3. What are the three degrees of normalization and how is normalization done in each degree?

1NF:

A table is in 1NF when: All the attributes are single-valued.

With no repeating columns (in other words, there cannot be two different columns with the same information).

With no repeating rows (in other words, the table must have a primary key).

All the composite attributes are broken down into its minimal component.

There should be SOME (full, partial, or transitive) kind of functional dependencies between non-key and key attributes.

99% of times, it’s usually 1NF.

2NF:

A table is in 2NF when: ● It is in 1NF.

● There should not be any partial dependencies so they must be removed if they exist.

3NF:

A table is in 3NF when: ● It is in 2NF.

● There should not be any transitive dependencies so they must be removed if they exist.

BCNF:

■ A stronger form of 3NF so it is also known as 3.5NF

■ We do not need to know much about it. Just know that here you compare between a prime attribute and a prime attribute and a non-key attribute and a non-key attribute.

4. What are the different database objects ?

There are total seven database objects (6 permanent database object + 1 temporary database object)

Permanent DB objects

- Table
- Views
- Stored procedures
- User-defined Functions
- Triggers
- Indexes

Temporary DB object

- Cursors

5. What is collation?

Bigdata Hadoop: SQL Interview Question with Answers

Collation is defined as set of rules that determine how character data can be sorted and compared. This can be used to compare A and, other language characters and also depends on the width of the characters.

ASCII value can be used to compare these character data.

6. What is a constraint and what are the seven constraints?

Constraint: something that limits the flow in a database.

○ 1. Primary key

○ 2. Foreign key

○ 3. Check

■ Ex: check if the salary of employees is over 40,000

○ 4. Default

■ Ex: If the salary of an employee is missing, place it with the default value.

○ 5. Nullability

■ NULL or NOT NULL

○ 6. Unique Key

○ 7. Surrogate Key

■ mainly used in data warehouse

7. What is a Surrogate Key ?

'Surrogate' means 'Substitute'.

Surrogate key is always implemented with a help of an identity column.

Identity column is a column in which the value are automatically generated by a SQL Server based on the seed value and incremental value.

Identity columns are ALWAYS INT, which means surrogate keys must be INT. Identity columns cannot have any NULL and cannot have repeated values. Surrogate key is a logical key.

8. What is a derived column , hows does it work , how it affects the performance of a database and how can it be improved?

The Derived Column a new column that is generated on the fly by applying expressions to transformation input columns.

Ex: FirstName + ' ' + LastName AS 'Full name'

Derived column affect the performances of the data base due to the creation of a temporary new column.

Execution plan can save the new column to have better performance next time.

9. What is a Transaction?

○ It is a set of TSQL statement that must be executed together as a single logical unit. ○ Has ACID properties:

Atomicity: Transactions on the DB should be all or nothing. So transactions make sure that any operations in the transaction happen or none of them do.

Consistency: Values inside the DB should be consistent with the constraints and integrity of the DB before and after a transaction has completed or failed.

Isolation: Ensures that each transaction is separated from any other transaction occurring on the system.

Durability: After successfully being committed to the RDMBS system the transaction will not be lost in the event of a system failure or error.

○ Actions performed on explicit transaction:

BEGIN TRANSACTION: marks the starting point of an explicit transaction for a connection.

COMMIT TRANSACTION (transaction ends): used to end an transaction successfully if no errors were encountered. All DML changes made in the transaction become permanent.

ROLLBACK TRANSACTION (transaction ends): used to erase a transaction which errors are encountered. All DML changes made in the transaction are undone.

SAVE TRANSACTION (transaction is still active): sets a savepoint in a transaction. If we roll back, we can only rollback to the most recent savepoint. Only one save point is possible per transaction. However, if you nest Transactions within a Master Trans, you may put Save points in each nested Tran. That is how you create more than one Save point in a Master Transaction.

10. What are the differences between OLTP and OLAP?

OLTP stands for Online Transactional Processing

OLAP stands for Online Analytical Processing

OLTP:

Normalization Level: highly normalized

Data Usage : Current Data (Database)

Processing : fast for delta operations (DML)

Operation : Delta operation (update, insert, delete) aka DML Terms Used : table, columns and relationships

OLAP:

Normalization Level: highly denormalized

Data Usage : historical Data (Data warehouse)

Processing : fast for read operations

Operation : read operation (select)

Terms Used : dimension table, fact table

11. How do you copy just the structure of a table?

SELECT * INTO NewDB.TBL_Structure

FROM OldDB.TBL_Structure

WHERE 1=0 -- Put any condition that does not make any sense.

12.What are the different types of Joins?

○ INNER JOIN: Gets all the matching records from both the left and right tables based on joining columns.

○ LEFT OUTER JOIN: Gets all non-matching records from left table & AND one copy of matching records from both the tables based on the joining columns.

○ RIGHT OUTER JOIN: Gets all non-matching records from right table & AND one copy of matching records from both the tables based on the joining columns.

○ FULL OUTER JOIN: Gets all non-matching records from left table & all non-matching records from right table & one copy of matching records from both the tables.

○ CROSS JOIN: returns the Cartesian product.

13. What are the different types of Restricted Joins?

○ SELF JOIN: joining a table to itself

○ RESTRICTED LEFT OUTER JOIN: gets all non-matching records from left

side

○ RESTRICTED RIGHT OUTER JOIN - gets all non-matching records from right

side

○ RESTRICTED FULL OUTER JOIN - gets all non-matching records from left table & gets all nonmatching records from right table.

14. What is a sub-query?

○ It is a query within a query

○ Syntax:

```
SELECT <column_name> FROM <table_name>

WHERE <column_name> IN/NOT IN

(

<another SELECT statement>

)
```

○ Everything that we can do using sub queries can be done using Joins, but anything that we can do using Joins may/may not be done using Subquery.

○ Sub-Query consists of an inner query and outer query. Inner query is a SELECT statement the result of which is passed to the outer query. The outer query can be SELECT, UPDATE, DELETE. The result of the inner query is generally used to filter what we select from the outer query.

○ We can also have a subquery inside of another subquery and so on. This is called a nested Subquery. Maximum one can have is 32 levels of nested Sub-Queries.

15. What are the SET Operators?

○ SQL set operators allows you to combine results from two or more SELECT statements.

○ Syntax:

SELECT Col1, Col2, Col3 FROM T1 <SET OPERATOR>

SELECT Col1, Col2, Col3 FROM T2

○ Rule 1: The number of columns in first SELECT statement must be same as the number of columns in the second SELECT statement.

○ Rule 2: The metadata of all the columns in first SELECT statement MUST be exactly same as the metadata of all the columns in second SELECT statement accordingly.

○ Rule 3: ORDER BY clause do not work with first SELECT statement. ○ UNION, UNION ALL, INTERSECT, EXCEPT

16. What is a derived table?

○ SELECT statement that is given an alias name and can now be treated as a virtual table and operations like joins, aggregations, etc. can be performed on it like on an actual table. ○ Scope is query bound, that is a derived table exists only in the query in which it was defined. SELECT temp1.SalesOrderID, temp1.TotalDue FROM

(SELECT TOP 3 SalesOrderID, TotalDue FROM Sales.SalesOrderHeader ORDER BY TotalDue DESC) AS temp1 LEFT OUTER JOIN

(SELECT TOP 2 SalesOrderID, TotalDue FROM Sales.SalesOrderHeader ORDER BY TotalDue DESC) AS temp2 ON temp1.SalesOrderID = temp2.SalesOrderID WHERE temp2.SalesOrderID IS NULL

17. What is a View?

○ Views are database objects which are virtual tables whose structure is defined by underlying SELECT statement and is mainly used to implement security at rows and columns levels on the base table.

○ One can create a view on top of other views.

○ View just needs a result set (SELECT statement).

○ We use views just like regular tables when it comes to query writing. (joins, subqueries, grouping)

○ We can perform DML operations (INSERT, DELETE, UPDATE) on a view. It actually affects the underlying tables only those columns can be affected which are visible in the view.

18. What are the types of views?

1. Regular View:

It is a type of view in which you are free to make any DDL changes on the underlying table.

-- create a regular view

CREATE VIEW v_regular AS SELECT * FROM T1

2. Schemabinding View:

It is a type of view in which the schema of the view (column) are physically bound to the schema of the underlying table. We are not allowed to perform any DDL changes to the underlying table for the columns that are referred by the schemabinding view structure.

■ All objects in the SELECT query of the view must be specified in two part naming conventions (schema_name.tablename).

■ You cannot use * operator in the SELECT query inside the view (individually name the columns)

■ All rules that apply for regular view.

CREATE VIEW v_schemabound WITH SCHEMABINDING AS SELECT ID, Name FROM

dbo.T2 -- remember to use two part naming convention

3. Indexed View:

19. What is an Indexed View?

○ It is technically one of the types of View, not Index.

○ Using Indexed Views, you can have more than one clustered index on the same table if needed. ○ All the indexes created on a View and underlying table are shared by Query Optimizer to select the best way to execute the query.

○ Both the Indexed View and Base Table are always in sync at any given point.

○ Indexed Views cannot have NCI-H, always NCI-CI, therefore a duplicate set of the data will be created.

20. What does WITH CHECK do?

○ WITH CHECK is used with a VIEW.

○ It is used to restrict DML operations on the view according to search predicate (WHERE clause) specified creating a view.

○ Users cannot perform any DML operations that do not satisfy the conditions in WHERE clause while creating a view.

○ WITH CHECK OPTION has to have a WHERE clause.

21. What is a RANKING function and what are the four RANKING functions?

Ranking functions are used to give some ranking numbers to each row in a dataset based on some ranking functionality.

Every ranking function creates a derived column which has integer value.

Different types of RANKING function:

ROW_NUMBER(): assigns an unique number based on the ordering starting with 1. Ties will be given different ranking positions.

RANK(): assigns an unique rank based on value. When the set of ties ends, the next ranking position will consider how many tied values exist and then assign the next value a new ranking with consideration the number of those previous ties. This will make the ranking position skip placement. position numbers based on how many of the same values occurred (ranking not sequential).
DENSE_RANK(): same as rank, however it will maintain its consecutive order nature regardless of ties in values; meaning if five records have a tie in the values, the next ranking will begin with the next ranking position.
Syntax:

<Ranking Function>() OVER(condition for ordering) -- always have to have an OVER clause

Ex:

SELECT SalesOrderID, SalesPersonID,

TotalDue,

ROW_NUMBER() OVER(ORDER BY TotalDue), RANK() OVER(ORDER BY TotalDue),

DENSE_RANK() OVER(ORDER BY TotalDue) FROM Sales.SalesOrderHeader

■ NTILE(n): Distributes the rows in an ordered partition into a specified number of groups.

22. What is PARTITION BY?

○ Creates partitions within the same result set and each partition gets its own ranking. That is, the rank starts from 1 for each partition.

○ Ex:

SELECT *, DENSE_RANK() OVER(PARTITION BY Country ORDER BY Sales DESC) AS DenseRank FROM SalesInfo

23. What is Temporary Table and what are the two types of it? ○ They are tables just like regular tables but the main difference is its scope.

○ The scope of temp tables is temporary whereas regular tables permanently reside. ○ Temporary table are stored in tempDB.

○ We can do all kinds of SQL operations with temporary tables just like regular tables like JOINs, GROUPING, ADDING CONSTRAINTS, etc.

○ Two types of Temporary Table

■ Local

#LocalTempTableName -- single pound sign

Only visible in the session in which they are created. It is session-bound.

■ Global

##GlobalTempTableName -- double pound sign

Global temporary tables are visible to all sessions after they are created, and are deleted when the session in which they were created in is disconnected.

It is last logged-on user bound. In other words, a global temporary table will disappear when the last user on the session logs off.

24. Explain Variables ?

○ Variable is a memory space (place holder) that contains a scalar value EXCEPT table variables, which is 2D data.

○ Variable in SQL Server are created using DECLARE Statement. ○ Variables are BATCH-BOUND.

○ Variables that start with @ are user-defined variables.

25. Explain Dynamic SQL (DSQL). ?

Dynamic SQL refers to code/script which can be used to operate on different data-sets based on some dynamic values supplied by front-end applications. It can be used to run a template SQL query against different tables/columns/conditions.

Declare variables: which makes SQL code dynamic.

Main disadvantage of D-SQL is that we are opening SQL Tool for SQL Injection attacks. You should build the SQL script by concatenating strings and variable.

26. What is SQL Injection Attack?

○ Moderator's definition: when someone is able to write a code at the front end using DSQL, he/she could use malicious code to drop, delete, or manipulate the database. There is no perfect protection from it but we can check if there is certain commands such as 'DROP' or 'DELETE' are included in the command line.

○ SQL Injection is a technique used to attack websites by inserting SQL code in web entry fields.

27. What is SELF JOIN?

○ JOINing a table to itself ○ When it comes to SELF JOIN, the foreign key of a table points to its primary key. ○ Ex: Employee(Eid, Name, Title, Mid)

○ Know how to implement it!!!

28. What is Correlated Subquery?

○ It is a type of subquery in which the inner query depends on the outer query. This means that that the subquery is executed repeatedly, once for each row of the outer query.

○ In a regular subquery, inner query generates a result set that is independent of the outer query.

○ Ex:

```
SELECT *

FROM HumanResources.Employee E

WHERE 5000 IN (SELECT S.Bonus

FROM Sales.SalesPerson S

WHERE S.SalesPersonID = E.EmployeeID)
```

○ The performance of Correlated Subquery is very slow because its inner query depends on the outer query. So the inner subquery goes through every single row of the result of the outer subquery.

29. What is the difference between Regular Subquery and Correlated Subquery?

○ Based on the above explanation, an inner subquery is independent from its outer subquery in Regular Subquery. On the other hand, an inner subquery depends on its outer subquery in Correlated Subquery.

30. What are the differences between DELETE and TRUNCATE .?

Delete:

DML statement that deletes rows from a table and can also specify rows using a WHERE clause. Logs every row deleted in the log file.

Slower since DELETE records every row that is deleted.

DELETE continues using the earlier max value of the identity column. Can have triggers on

DELETE.

Truncate:

DDL statement that wipes out the entire table and you cannot delete specific rows.

Does minimal logging, minimal as not logging everything. TRUNCATE will remove the pointers that point to their pages, which are deallocated.

Faster since TRUNCATE does not record into the log file. TRUNCATE resets the identity column.

Cannot have triggers on TRUNCATE.

31. What are the three different types of Control Flow statements?

1. WHILE
2. IF-ELSE
3. CASE

32. What is Table Variable? Explain its advantages and disadvantages.?

○ If we want to store tabular data in the form of rows and columns into a variable then we use a table variable. ○ It is able to store and display 2D data (rows and columns).

○ We cannot perform DDL (CREATE, ALTER, DROP).

Advantages:

■ Table variables can be faster than permanent tables.

■ Table variables need less locking and logging resources.

Disadvantages:

■ Scope of Table variables is batch bound.

■ Table variables cannot have constraints.

■ Table variables cannot have indexes.

■ Table variables do not generate statistics.

■ Cannot ALTER once declared (Again, no DDL statements).

33. What are the differences between Temporary Table and Table Variable?

Temporary Table:

It can perform both DML and DDL Statement. Session bound Scope

Syntax CREATE TABLE #temp

Have indexes

Table Variable:

Can perform only DML, but not DDL Batch bound scope

DECLARE @var TABLE(...)

Cannot have indexes

34. What is Stored Procedure (SP)?

It is one of the permanent DB objects that is precompiled set of TSQL statements that can accept and return multiple variables.

It is used to implement the complex business process/logic. In other words, it encapsulates your entire business process.

Compiler breaks query into Tokens. And passed on to query optimizer. Where execution plan is generated the very 1st time when we execute a stored procedure after creating/altering it and same execution plan is utilized for subsequent executions.

Database engine runs the machine language query and execute the code in 0's and 1's.

When a SP is created all Tsql statements that are the part of SP are pre-compiled and execution plan is stored in DB which is referred for following executions.

Explicit DDL requires recompilation of SP's.

35. What are the four types of SP?

System Stored Procedures (SP_****): built-in stored procedures that were created by Microsoft.

User Defined Stored Procedures: stored procedures that are created by users. Common naming convention (usp_****)

CLR (Common Language Runtime): stored procedures that are implemented as public static methods on a class in a Microsoft .NET Framework assembly.

Extended Stored Procedures (XP_****): stored procedures that can be used in other platforms such as Java or C++.

36. Explain the Types of SP..? ○ SP with no parameters:

○ SP with a single input parameter:

○ SP with multiple parameters:

○ SP with output parameters:

Extracting data from a stored procedure based on an input parameter and outputting them using output

variables.

○ SP with RETURN statement (the return value is always single and integer value)

37. What are the characteristics of SP? ○ SP can have any kind of DML and DDL statements. ○ SP can have error handling (TRY ... CATCH).

○ SP can use all types of table.

○ SP can output multiple integer values using OUT parameters, but can return only one scalar INT value. ○ SP can take any input except a table variable.

○ SP can set default inputs.

○ SP can use DSQL.

○ SP can have nested SPs.

○ SP cannot output 2D data (cannot return and output table variables).

○ SP cannot be called from a SELECT statement. It can be executed using only a EXEC/EXECUTE statement.

38. What are the advantages of SP?

○ Precompiled code hence faster.

○ They allow modular programming, which means it allows you to break down a big chunk of code into smaller pieces of codes. This way the code will be more readable and more easier to manage.

○ Reusability.

○ Can enhance security of your application. Users can be granted permission to execute SP without having to have direct permissions on the objects referenced in the procedure.

○ Can reduce network traffic. An operation of hundreds of lines of code can be performed through single statement that executes the code in procedure rather than by sending hundreds of lines of code over the network.

○ SPs are pre-compiled, which means it has to have an Execution Plan so every time it gets executed after creating a new Execution Plan, it will save up to 70% of execution time. Without it, the SPs are just like any regular TSQL statements.

39. What is User Defined Functions (UDF)?

○ UDFs are a database object and a precompiled set of TSQL statements that can accept parameters, perform complex business calculation, and return of the action as a value.

○ The return value can either be single scalar value or result set-2D data. ○ UDFs are also precompiled and their execution plan is saved.

○ PASSING INPUT PARAMETER(S) IS/ARE OPTIONAL, BUT MUST HAVE A RETURN STATEMENT.

40. What is the difference between Stored Procedure and UDF?

Stored Procedure:

may or may not return any value. When it does, it must be scalar INT. Can create temporary tables.

Can have robust error handling in SP (TRY/CATCH, transactions). Can include any DDL and DML statements.

UDF: must return something, which can be either scalar/table valued. Cannot access to temporary

tables.

No robust error handling available in UDF like TRY/ CATCH and transactions. Cannot have any DDL and can do DML only with table variables.

41. What are the types of UDF?

1. Scalar

Deterministic UDF: UDF in which particular input results in particular output. In other words, the output depends on the input.

Non-deterministic UDF: UDF in which the output does not directly depend on the input.

2. In-line UDF:

UDFs that do not have any function body(BEGIN...END) and has only a RETURN statement. In-line UDF must return 2D data.

3. Multi-line or Table Valued Functions:

It is an UDF that has its own function body (BEGIN ... END) and can have multiple SQL statements

that return a single output. Also must return 2D data in the form of table variable.

42. What is the difference between a nested UDF and recursive UDF?

○ Nested UDF: calling an UDF within an UDF

○ Recursive UDF: calling an UDF within itself

43. What is a Trigger?

○ It is a precompiled set of TSQL statements that are automatically executed on a particular DDL, DML or log-on

event.

○ Triggers do not have any parameters or return statement.

○ Triggers are the only way to access to the INSERTED and DELETED tables (aka. Magic Tables). ○ You can DISABLE/ENABLE Triggers instead of DROPPING them:

DISABLE TRIGGER <name> ON <table/view name>/DATABASE/ALL SERVER

ENABLE TRIGGER <name> ON <table/view name>/DATABASE/ALL SERVER

44.What are the types of Triggers?

1. DML Trigger

DML Triggers are invoked when a DML statement such as INSERT, UPDATE, or DELETE occur which modify data in a specified TABLE or VIEW.

A DML trigger can query other tables and can include complex TSQL statements. They can cascade changes through related tables in the database.

They provide security against malicious or incorrect DML operations and enforce restrictions that are more complex than those defined with constraints.

2. DDL Trigger

Pretty much the same as DML Triggers but DDL Triggers are for DDL operations. DDL Triggers are at the database or server level (or scope).

DDL Trigger only has AFTER. It does not have INSTEAD OF.

3. Logon Trigger

Logon triggers fire in response to a logon event.

This event is raised when a user session is established with an instance of SQL server. Logon TRIGGER has server scope.

45. What are 'inserted' and 'deleted' tables (aka. magic tables)?

○ They are tables that you can communicate with between the external code and trigger body.

○ The structure of inserted and deleted magic tables depends upon the structure of the table in a DML statement. ○ UPDATE is a combination of INSERT and DELETE, so its old record will be in the deleted table and its new record will be stored in the inserted table.

46. What are some String functions to remember? LEN(string): returns the length of string.

UPPER(string) & LOWER(string): returns its upper/lower string

LTRIM(string) & RTRIM(string): remove empty string on either ends of the string LEFT(string): extracts a certain number of characters from left side of the string RIGHT(string): extracts a certain number of characters from right side of the string SUBSTRING(string, starting_position, length): returns the sub string of the string REVERSE(string): returns the reverse string of the string

Concatenation: Just use + sign for it

REPLACE(string, string_replaced, string_replace_with)

47. What are the three different types of Error Handling?

1. TRY CATCH

The first error encountered in a TRY block will direct you to its CATCH block ignoring the rest of the code in the TRY block will generate an error or not.

2. @@error

stores the error code for the last executed SQL statement. If there is no error, then it is equal to 0.

If there is an error, then it has another number (error code).

3. RAISERROR() function

A system defined function that is used to return messages back to applications using the same format which SQL uses for errors or warning message.

48. Explain about Cursors ..?

○ Cursors are a temporary database object which are used to loop through a table on row-by-row basis. There are five types of cursors:

■ 1. Static: shows a static view of the data with only the changes done by session which opened the cursor.

■ 2. Dynamic: shows data in its current state as the cursor moves from record-to-record.

■ 3. Forward Only: move only record-by-record ■

4. Scrolling: moves anywhere.

■ 5. Read Only: prevents data manipulation to cursor data set.

49. What is the difference between Table scan and seek ? ○ Scan: going through from the first page to the last page of an offset by offset or row by row. ○ Seek: going to the specific node and fetching the information needed.

○ 'Seek' is the fastest way to find and fetch the data. So if you see your Execution Plan and if all of them is a seek, that means it's optimized.

50. Why are the DML operations are slower on Indexes?

○ It is because the sorting of indexes and the order of sorting has to be always maintained.

○ When inserting or deleting a value that is in the middle of the range of the index, everything has to be rearranged again. It cannot just insert a new value at the end of the index.

51. What is a heap (table on a heap)?

○ When there is a table that does not have a clustered index, that means the table is on a heap. ○ Ex: Following table 'Emp' is a table on a heap.

SELECT * FROM Emp WHERE ID BETWEEN 2 AND 4 -- This will do scanning.

52. What is the architecture in terms of a hard disk, extents and pages?

○ A hard disk is divided into Extents.

○ Every extent has eight pages.

○ Every page is 8KBs (8060 bytes).

53. What are the nine different types of Indexes?

○ 1. Clustered

○ 2. Non-clustered

○ 3. Covering

○ 4. Full Text Index

○ 5. Spatial

○ 6. Unique

○ 7. Filtered

○ 8. XML

○ 9. Index View

54. What is a Clustering Key?

○ It is a column on which I create any type of index is called a Clustering Key for that particular index.

55. Explain about a Clustered Index.?

○ Unique Clustered Indexes are automatically created when a PK is created on a table.

○ But that does not mean that a column is a PK only because it has a Clustered Index.

○ Clustered Indexes store data in a contiguous manner. In other words, they cluster the data into a certain spot on a hard disk continuously.

○ The clustered data is ordered physically.

○ You can only have one CI on a table.

56. What happens when Clustered Index is created?

○ First, a B-Tree of a CI will be created in the background.

○ Then it will physically pull the data from the heap memory and physically sort the data based on the clustering

key.

○ Then it will store the data in the leaf nodes.

○ Now the data is stored in your hard disk in a continuous manner.

57. What are the four different types of searching information in a table?

○ 1. Table Scan -> the worst way

○ 2. Table Seek -> only theoretical, not possible ○ 3. Index Scan -> scanning leaf nodes ○

4. Index Seek -> getting to the node needed, the best way

58. What is Fragmentation .?

○ Fragmentation is a phenomenon in which storage space is used inefficiently.

○ In SQL Server, Fragmentation occurs in case of DML statements on a table that has an index.

○ When any record is deleted from the table which has any index, it creates a memory bubble which causes fragmentation.

○ Fragmentation can also be caused due to page split, which is the way of building B-Tree dynamically according to the new records coming into the table.

○ Taking care of fragmentation levels and maintaining them is the major problem for Indexes. ○ Since Indexes slow down DML operations, we do not have a lot of indexes on OLTP, but it is recommended to have many different indexes in OLAP.

59. What are the two types of fragmentation?

1. Internal Fragmentation

It is the fragmentation in which leaf nodes of a B-Tree is not filled to its fullest capacity and contains memory bubbles.

2. External Fragmentation

It is fragmentation in which the logical ordering of the pages does not match the physical ordering of the pages on the secondary storage device.

60. What are Statistics?

○ Statistics allow the Query Optimizer to choose the optimal path in getting the data from the underlying table. ○ Statistics are histograms of max 200 sampled values from columns separated by intervals.

○ Every statistic holds the following info:

■ 1. The number of rows and pages occupied by a table's data

■ 2. The time that statistics was last updated

■ 3. The average length of keys in a column

■ 4. Histogram showing the distribution of data in column

61. What are some optimization techniques in SQL?

1. Build indexes. Using indexes on a table, It will dramatically increase the performance of your read operation because it will allow you to perform index scan or index seek depending on your search predicates and select predicates instead of table scan.

Building non-clustered indexes, you could also increase the performance further.

2. You could also use an appropriate filtered index for your non clustered index because it could avoid performing a key lookup.
3. You could also use a filtered index for your non-clustered index since it allows you to create an index on a particular part of a table that is accessed more frequently than other parts.
4. You could also use an indexed view, which is a way to create one or more clustered indexes on the same table.

In that way, the query optimizer will consider even the clustered keys on the indexed views so there might be a possible faster option to execute your query.

5. Do table partitioning. When a particular table as a billion of records, it would be practical to partition a table so that it can increase the read operation performance. Every partitioned table will be considered as physical smaller tables internally.
6. Update statistics for TSQL so that the query optimizer will choose the most optimal path in getting the data

from the underlying table. Statistics are histograms of maximum 200 sample values from columns separated by intervals.

7. Use stored procedures because when you first execute a stored procedure, its execution plan is stored and the

same execution plan will be used for the subsequent executions rather than generating an execution plan every time.

8. Use the 3 or 4 naming conventions. If you use the 2 naming convention, table name and column name, the SQL engine will take some time to find its schema. By specifying the schema name or even server name, you will be able to save some time for the SQL server.
9. Avoid using SELECT *. Because you are selecting everything, it will decrease the performance. Try to select columns you need.
10. Avoid using CURSOR because it is an object that goes over a table on a row-by-row basis, which is similar to the table scan. It is not really an effective way.
11. Avoid using unnecessary TRIGGER. If you have unnecessary triggers, they will be triggered needlessly. Not only slowing the performance down, it might mess up your whole program as well.
12. Manage Indexes using RECOMPILE or REBUILD.

The internal fragmentation happens when there are a lot of data bubbles on the leaf nodes of the b-tree and the leaf nodes are not used to its fullest capacity. By recompiling, you can push the actual data on the b-tree to the left side of the leaf level and push the memory bubble to the right side. But it is still a temporary solution because the memory bubbles will still exist and won't be still accessed much.

The external fragmentation occurs when the logical ordering of the b-tree pages does not match the physical ordering on the hard disk. By rebuilding, you can cluster them all together, which will solve not only the internal but also the external fragmentation issues. You can check the status of the fragmentation by using Data Management Function, sys.dm_db_index_physical_stats(db_id, table_id, index_id, partition_num, flag), and looking at the columns, avg_page_space_used_in_percent for the internal fragmentation and avg_fragmentation_in_percent for the external fragmentation.

13. Try to use JOIN instead of SET operators or SUB-QUERIES because set operators and subqueries are slower than joins and you can implement the features of sets and sub-queries using joins.
14. Avoid using LIKE operators, which is a string matching operator but it is mighty slow.
15. Avoid using blocking operations such as order by or derived columns.
16. For the last resort, use the SQL Server Profiler. It generates a trace file, which is a really detailed version of execution plan. Then DTA (Database Engine Tuning Advisor) will take a trace file as its input and analyzes it and gives you the recommendation on how to improve your query further.

62. How do you present the following tree in a form of a table?

```
A

/ \

B C

/ \ / \

D E F G
```

```
CREATE TABLE tree ( node CHAR(1), parent Node CHAR(1), [level] INT) INSERT INTO tree
VALUES ('A', null, 1),
('B', 'A', 2),
('C', 'A', 2),
('D', 'B', 3),
('E', 'B', 3),
('F', 'C', 3),
('G', 'C', 3)

SELECT * FROM tree
```

Result:

```
A NULL 1
B A 2
C A 2
D B 3
E B 3
F C 3
G C 3
```

63. How do you reverse a string without using REVERSE (‘string’) ?

```
CREATE PROC rev (@string VARCHAR(50)) AS
BEGIN
DECLARE @new_string VARCHAR(50) = ''
DECLARE @len INT = LEN(@string)
WHILE (@len <> 0)
BEGIN
DECLARE @char CHAR(1) = SUBSTRING(@string, @len, 1) SET @new_string = @new_string +
@char
SET @len = @len - 1
END
PRINT @new_string
END
EXEC rev 'dinesh'
```

64. What is Deadlock?

○ Deadlock is a situation where, say there are two transactions, the two transactions are waiting for each other to release their locks.

○ The SQL automatically picks which transaction should be killed, which becomes a deadlock victim, and roll back the change for it and throws an error message for it.

65. What is a Fact Table?

The primary table in a dimensional model where the numerical performance measurements (or facts) of the

business are stored so they can be summarized to provide information about the history of the operation of an organization.

We use the term fact to represent a business measure. The level of granularity defines the grain of the fact table.

66. What is a Dimension Table?

Dimension tables are highly denormalized tables that contain the textual descriptions of the business and facts in their fact table.

Since it is not uncommon for a dimension table to have 50 to 100 attributes and dimension tables tend to be relatively shallow in terms of the number of rows, they are also called a wide table.

A dimension table has to have a surrogate key as its primary key and has to have a business/alternate key to link between the OLTP and OLAP.

67. What are the types of Measures?

○ Additive: measures that can be added across all dimensions (cost, sales).

○ Semi-Additive: measures that can be added across few dimensions and not with others.

○ Non-Additive: measures that cannot be added across all dimensions (stock rates).

68. What is a Star Schema?

○ It is a data warehouse design where all the dimensions tables in the warehouse are directly connected to the fact table.

○ The number of foreign keys in the fact table is equal to the number of dimensions. ○ It is a simple design and hence faster query.

69. What is a Snowflake Schema?

○ It is a data warehouse design where at least one or more multiple dimensions are further normalized. ○ Number of dimensions > number of fact table foreign keys

○ Normalization reduces redundancy so storage wise it is better but querying can be affected due to the excessive joins that need to be performed.

70. What is granularity?

○ The lowest level of information that is stored in the fact table. ○ Usually determined by the time dimension table.

○ The best granularity level would be per transaction but it would require a lot of memory.

71. What is a Surrogate Key?

○ It is a system generated key that is an identity column with the initial value and incremental value and ensures the uniqueness of the data in the dimension table.

○ Every dimension table must have a surrogate key to identify each record!!!

72. What are some advantages of using the Surrogate Key in a Data Warehouse?

○ 1. Using a SK, you can separate the Data Warehouse and the OLTP: to integrate data coming from heterogeneous sources, we need to differentiate between similar business keys from the OLTP. The keys in OLTP are the alternate key (business key).

○ 2. Performance: The fact table will have a composite key. If surrogate keys are used, then in the fact table, we will have integers for its foreign keys.

■ This requires less storage than VARCHAR.

■ The queries will run faster when you join on integers rather than VARCHAR.

■ The partitioning done on SK will be faster as these are in sequence.

○ 3. Historical Preservation: A data warehouse acts as a repository of historical data so there will be various versions of the same record and in order to differentiate between them, we need a SK then we can keep the history of data.

○ 4. Special Situations (Late Arriving Dimension): Fact table has a record that doesn't have a match yet in the dimension table. Surrogate key usage enables the use of such a 'not found' record as a SK is not dependent on the

ETL process.

73. What is the datatype difference between a fact and dimension tables?

○ 1. Fact Tables

They hold numeric data.

They contain measures. They

are deep.

○ 2. Dimensional Tables They

hold textual data.

They contain attributes of their fact tables.

They are wide.

74. What are the types of dimension tables?

○ 1. Conformed Dimensions ■ when a particular dimension is connected to one or more fact

tables. ex) time dimension ○ 2.

Parent-child Dimensions

■ A parent-child dimension is distinguished by the fact that it contains a hierarchy based on a

recursive relationship. ■ when a particular dimension points to its own surrogate key to show an

unary relationship. ○ 3.

Role Playing Dimensions

■ when a particular dimension plays different roles in the same fact table. ex) dim_time and orderDateKey, shippedDateKey...usually a time dimension table.

■ Role-playing dimensions conserve storage space, save processing time, and improve database manageability .

○ 4. Slowly Changing Dimensions: A dimension table that have data that changes slowly that occur by inserting and updating of records.

■ 1. Type 0: columns where changes are not allowed - no change ex) DOB, SSNm

■ 2. Type 1: columns where its values can be replaced without adding its new row - replacement

■ 3. Type 2: for any change for the value in a column, a new record it will be added - historical data. Previous

values are saved in records marked as outdated. For even a single type 2 column, startDate, EndDate, and status are needed.

■ 4. Type 3: advanced version of type 2 where you can set up the upper limit of history which drops the oldest record when the limit has been reached with the help of outside SQL implementation.

■ Type 0 ~ 2 are implemented on the column level.

○ 5. Degenerated Dimensions: a particular dimension that has an one-to-one relationship between

itself and the fact table. ■ When a particular Dimension table grows at the same rate as a fact table,

the actual dimension can be removed and the dimensions from the dimension table can be inserted

into the actual fact table.

■ You can see this mostly when the granularity level of the the facts are per transaction.

■ E.g. The dimension salesorderdate (or other dimensions in DimSalesOrder would grow everytime a sale is made therefore the dimension (attributes) would be moved into the fact table.

○ 6. Junk Dimensions: holds all miscellaneous attributes that may or may not necessarily belong to any other dimensions. It could be yes/no, flags, or long open-ended text data.

75. What is your strategy for the incremental load?

The combination of different techniques for the incremental load in my previous projects; time stamps, CDC (Change Data Capture), MERGE statement and CHECKSUM() in TSQL, LEFT OUTER JOIN, TRIGGER, the Lookup Transformation in SSIS.

76. What is CDC?

CDC (Change Data Capture) is a method to capture data changes, such as INSERT, UPDATE and DELETE,

happening in a source table by reading transaction log files. Using CDC in the process of an incremental load, you

are going to be able to store the changes in a SQL table, enabling us to apply the changes to a target table incrementally.

In data warehousing, CDC is used for propagating changes in the source system into your data warehouse,

updating dimensions in a data mart, propagating standing data changes into your data warehouse and such.

The advantages of CDC are:

- It is almost real time ETL.

- It can handle small volume of data.

- It can be more efficient than replication.

- It can be auditable.

- It can be used to configurable clean up.

Disadvantages of CDC are:

- Lots of change tables and functions

- Bad for big changes e.g. truncate & reload Optimization of CDC:

- Stop the capture job during load

- When applying changes to target, it is ideal to use merge.

77. What is the difference between a connection and session ?

○ Connection: It is the number of instance connected to the database. An instance is modelized soon as the application is open again.

○ Session: A session run queries.In one connection, it allowed multiple sessions for one connection.

78. What are all different types of collation sensitivity?

Following are different types of collation sensitivity - Case

Sensitivity - A and a and B and b.

Accent Sensitivity.

Kana Sensitivity - Japanese Kana characters.

Width Sensitivity - Single byte character and double byte character.

79. What is CLAUSE?

SQL clause is defined to limit the result set by providing condition to the query. This usually filters some rows from the whole set of records.

Example - Query that has WHERE condition Query that has HAVING condition.

80. What is Union, minus and Interact commands?

UNION operator is used to combine the results of two tables, and it eliminates duplicate rows from the tables.

MINUS operator is used to return rows from the first query but not from the second query. Matching records of first and second query and other rows from the first query will be displayed as a result set.

INTERSECT operator is used to return rows returned by both the queries.

81.How to fetch common records from two tables?

Common records result set can be achieved by -.

Select studentID from student. <strong>INTERSECT </strong> Select StudentID from Exam

82.How to fetch alternate records from a table?

Records can be fetched for both Odd and Even row numbers -.

To display even numbers-.

Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=0 To display odd numbers-.

Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1 from (Select rowno, studentId from student) where mod(rowno,2)=1.[/sql]

83. How to select unique records from a table?

Select unique records from a table by using DISTINCT keyword.

Select DISTINCT StudentID, StudentName from Student.

84.How to remove duplicate rows from table?

Step 1: Selecting Duplicate rows from table

Select rollno FROM Student WHERE ROWID <>

(Select max (rowid) from Student b where rollno=b.rollno);

Step 2: Delete duplicate rows

Delete FROM Student WHERE ROWID <>

(Select max (rowid) from Student b where rollno=b.rollno);

85.What is ROWID and ROWNUM in SQL?

RowID

1.ROWID is nothing but Physical memory allocation

2.ROWID is permanant to that row which identifies the address of that row.

3.ROWID is 16 digit Hexadecimal number which is uniquely identifies the rows.

4.ROWID returns PHYSICAL ADDRESS of that row.

5. ROWID is automatically generated unique id of a row and it is generated at the time of insertion of row.
6. ROWID is the fastest means of accessing data.

ROWNUM:

1. ROWNUM is nothing but the sequence which is allocated to that data retreival bunch.
2. ROWNUM is tempararily allocated sequence to the rows.

3.ROWNUM is numeric sequence number allocated to that row temporarily.

4.ROWNUM returns the sequence number to that row.

5. ROWNUM is an dynamic value automatically retrieved along with select statement output.

6.ROWNUM is not related to access of data.

86. How to find count of duplicate rows?

Select rollno, count (rollno) from Student

Group by rollno Having count (rollno)>1 Order by count (rollno) desc;

87.How to find Third highest salary in Employee table using self-join?

Select * from Employee a Where 3 = (Select Count (distinct Salary) from Employee where a.salary<=b.salary;

88. How to display following using query?

*

**

***

We cannot use dual table to display output given above. To display output use any table. I am using Student table.

SELECT lpad (‘*’, ROWNUM,’*’) FROM Student WHERE ROWNUM <4;

89. How to display Date in DD-MON-YYYY table?

Select to_date (Hire_date,’DD-MON-YYYY’) Date_Format from Employee;

90. If marks column contain the comma separated values from Student table. How to calculate the count of that comma separated values?

Student Name Marks

Dinesh 30,130,20,4

Kumar 100,20,30

Sonali 140,10

Select Student_name, regexp_count (marks,’,’) + As “Marks Count” from Student;

91. What is query to fetch last day of previous month in oracle?

Select LAST_DAY (ADD_MONTHS (SYSDATE,-1)) from dual;

92. How to display the String vertically in Oracle?

SELECT SUBSTR (‘AMIET’, LEVEL, 1) FROM dual Connect by level <= length (‘AMIET’);

93. How to display departmentwise and monthwise maximum salary?

Select Department_no, TO_CHAR (Hire_date,’Mon’) as Month from Employee group by Department_no, TO_CHAR (Hire_date,’mon’);

94. How to calculate number of rows in table without using count function?

Select table_name, num_rows from user_tables where table_name=’Employee’;

Tip: User needs to use the system tables for the same. So using user_tables user will get the number of rows in the table

95. How to fetch common records from two different tables which has not any joining condition ?

Select * from Table1

Intersect

Select * from Table2;

96. Explain Execution Plan.?

Query optimizer is a part of SQL server that models the way in which the relational DB engine works and comes up with the most optimal way to execute a query. Query Optimizer takes into account amount of resources used, I/O and CPU processing time etc. to generate a plan that will allow query to execute in most efficient and faster manner. This is known as EXECUTION PLAN.

Optimizer evaluates a number of plans available before choosing the best and faster on available. Every query has an execution plan.

Definition by the mod: Execution Plan is a plan to execute a query with the most optimal way which is generated by Query Optimizer. Query Optimizer analyzes statistics, resources used, I/O and CPU processing time and etc. and comes up with a number of plans. Then it evaluates those plans and the most optimized plan out of the plans is Execution Plan. It is shown to users as a graphical flow chart that should be read from right to left and top to bottom.

97. Types of SQL Commands
SQL commands are categorized into **five types**:

1️⃣ DDL – Data Definition Language
Defines the structure of the database.
Commands:
- CREATE – creates database objects
- ALTER – modifies objects
- DROP – deletes objects
- TRUNCATE – removes all rows (auto-commit)

Important Points:
- DDL commands are **auto-commit**.
- Cannot be rolled back.

2️⃣ DML – Data Manipulation Language
Manages data stored in database tables.
Commands:
- INSERT
- UPDATE
- DELETE

Important Points:
- Not auto-commit (can be rolled back).
- Works on table data.

3️⃣ DCL – Data Control Language
Controls access/permissions.
Commands:
- GRANT
- REVOKE

4️⃣ TCL – Transaction Control Language
Manages transactions in SQL.
Commands:
- COMMIT
- ROLLBACK
- SAVEPOINT

5️⃣ DQL – Data Query Language
Used for data retrieval.
Command:
- SELECT

98. Primary Key
A *Primary Key* uniquely identifies each row in a table.
Key Features:
- Must contain **unique** values
- Cannot contain **NULL**
- Only **one** primary key per table
- Automatically creates a **unique index**

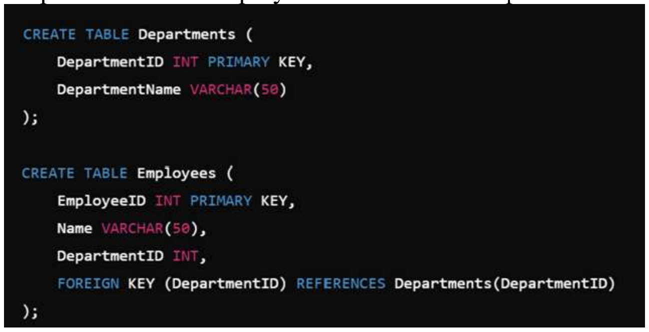
Example:
EmployeeID is primary key in Employees table.

99. Foreign Key
A *Foreign Key* references a primary key in another table and creates relationships.

Key Features:

- Enforces **referential integrity**
- Can accept NULL
- Can contain duplicates
- Uses cascading rules
    - ON DELETE CASCADE
    - ON UPDATE CASCADE
    - SET NULL, SET DEFAULT, NO ACTION

Example:
DepartmentID in Employees → references Departments table.



```
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50)
);

CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(50),
    DepartmentID INT,
    FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
```

100. UNIQUE Key
Ensures all values in a column are distinct.
Key Features:

- Prevents duplicate values
- Allows **one NULL** (MySQL/SQL Server)
- Many databases allow multiple NULLs (PostgreSQL)
- Multiple UNIQUE constraints allowed
- Creates a **non-clustered index**

101. NOT NULL Constraint
Prevents NULL insertion into a column.
Key Features:

- Ensures data completeness
- Often paired with CHECK or DEFAULT
- Can be applied to multiple columns

102. DEFAULT Constraint
Provides a default value if no value is specified.
Key Features:

- Automatically assigns value
- Does NOT apply if NULL is explicitly passed
- Can use functions (e.g., CURRENT_TIMESTAMP)

103. Denormalization
Process of adding redundancy to improve read performance.
Key Features:

- Used in OLAP, data warehousing
- Reduces number of JOINS
- Supports star and snowflake schema
- Sacrifices storage to gain speed

104. Auto Increment
Automatically generates sequential numbers (usually for primary key).
DB Specific:

- MySQL → AUTO_INCREMENT
- SQL Server → IDENTITY
- PostgreSQL → SERIAL or IDENTITY

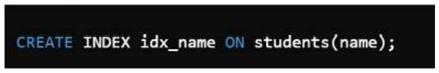
Important:

- Does not reset after deletion
- Ensures unique record identifiers

105. Clustered Index
Stores table data **physically** in sorted order.
Key Points:

- Only **one** clustered index per table
- Very fast for **range queries**
- Automatically created on primary key (in many DBMS)
- Re arranges table rows physically

106. Non-Clustered Index
Stores pointers to data (separate structure).
Key Points:

- Multiple allowed
- Improves search performance
- Does not change physical order
- Good for frequently searched columns
- Supports "include columns" (SQL Server)
- 

```
CREATE INDEX idx_name ON students(name);
```

107. Handling NULL in SQL
NULL = unknown/missing value.
Methods:

- IS NULL — check NULL
- IS NOT NULL — check non-NULL
- COALESCE(a, b) — returns first non-NULL
- IFNULL(a, b) — MySQL
- NULLIF(a, b) — returns NULL if equal

Important:

- NULL ≠ ‘’
- NULL ≠ 0
- NULL ≠ NULL (use IS NULL)

108. Dynamic SQL
SQL generated and executed at runtime.
Key Uses:

- When table names/columns vary
- Used in stored procedures
- Helps create flexible logic

Risk:

- SQL injection — avoid by using **prepared statements**.

109. SQL Query Optimization
Improves speed and efficiency.

Best Practices

1. Create **indexes** on columns used in WHERE, JOIN, ORDER BY.
2. Avoid SELECT * — select only needed columns.
3. Use **LIMIT/TOP** for reducing data load.
4. Prefer **JOINs** over subqueries.
5. Use **EXISTS** instead of IN for large sets.
6. Avoid functions in WHERE clause (break index).
7. Rewrite OR conditions using UNION.
8. Partition large tables.
9. Use temporary tables / CTEs.
10. Use execution plan for analysis.
11. Use covering indexes for faster lookup.
12. Avoid indexing low-cardinality columns.

110. Query Execution Plan

Shows how SQL executes a query.

Why use it?

- Identify table scans
- Check index usage
- Understand join type (Hash, Merge, Nested Loop)
- Detect performance bottlenecks

Commands:

- MySQL → EXPLAIN
- Postgres → EXPLAIN ANALYZE
- SQL Server → Execution Plan GUI

111. Indexing

Speeds up SELECT queries by creating a lookup structure.

Types:

- Primary Index
- Unique Index
- Clustered Index
- Non-clustered Index
- Composite Index

Important Points:

- Indexes slow down INSERT/UPDATE/DELETE
- Left-most prefix rule for composite index
- Avoid indexing boolean-like fields (low cardinality)

112. Table Partitioning

Splits a large table into smaller logical parts.

Types:

- Range
- List
- Hash
- Composite

Benefits:

- Faster queries
- Supports archival
- Reduces locking
- Allows partition pruning

113. Avoiding Deadlocks

- Access tables in same order

- Keep transactions short
- Avoid user input inside transactions
- Use lower isolation levels if possible
- Use proper indexing
- Use NOLOCK for read-only queries (SQL Server)

114. EXISTS in SQL
Used to check if a subquery returns rows.
Returns:

- TRUE → at least one row
- FALSE → no rows

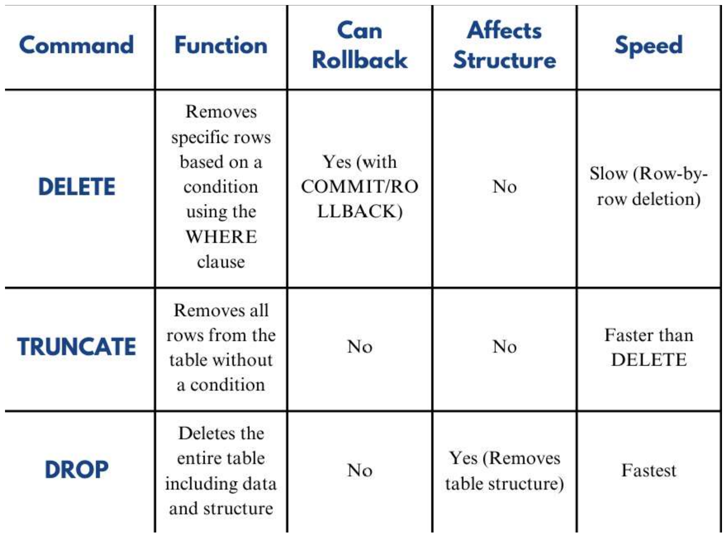
Performance:

- Faster than IN for large datasets
- NOT EXISTS is faster than NOT IN

115. Stored Procedure vs Function

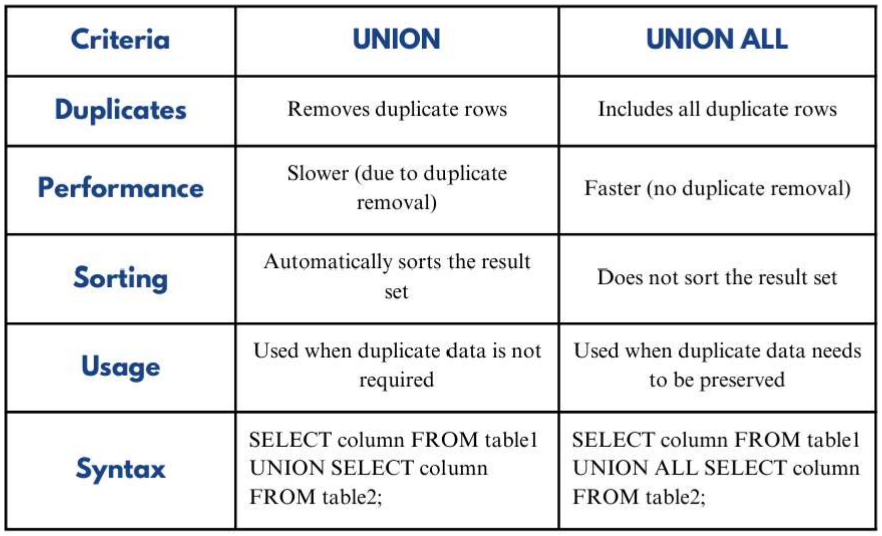
Feature	Stored Procedure	Function
Return Type	Multiple values	Single value/Table
DML (Insert/Update/Delete)	Allowed	Not allowed
Use in SELECT	✗ No	✓ Yes
Parameters	IN, OUT	Only IN
Error Handling	Supported	Not supported
Purpose	Business logic	Calculations

116. Delete vs truncate vs drop



Command	Function	Can Rollback	Affects Structure	Speed
DELETE	Removes specific rows based on a condition using the WHERE clause	Yes (with COMMIT/ROLLBACK)	No	Slow (Row-by-row deletion)
TRUNCATE	Removes all rows from the table without a condition	No	No	Faster than DELETE
DROP	Deletes the entire table including data and structure	No	Yes (Removes table structure)	Fastest

117. Union vs Union All

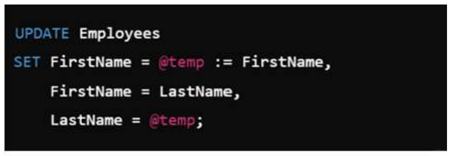


Criteria	UNION	UNION ALL
Duplicates	Removes duplicate rows	Includes all duplicate rows
Performance	Slower (due to duplicate removal)	Faster (no duplicate removal)
Sorting	Automatically sorts the result set	Does not sort the result set
Usage	Used when duplicate data is not required	Used when duplicate data needs to be preserved
Syntax	SELECT column FROM table1 UNION SELECT column FROM table2;	SELECT column FROM table1 UNION ALL SELECT column FROM table2;

118. How to swap two columns in SQL?

You can swap the values of two columns using the UPDATE statement with the TEMPORARY variable technique.

Syntax: Correct Way Using Temporary Variable



```
UPDATE Employees
SET FirstName = @temp := FirstName,
    FirstName = LastName,
    LastName = @temp;
```

119. What is Table Partitioning?

Table Partitioning is a technique used to divide large tables into smaller, more manageable pieces without changing the table structure.

Key Points:

Improves query performance on large datasets.

Simplifies data management.

Helps in faster data retrieval.

Each partition is stored separately.

Data can be partitioned by range, list, hash, or composite methods.

Types of Partitioning:

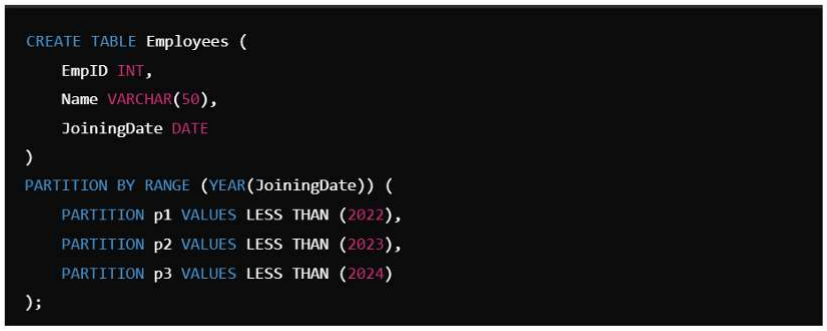
Range Partitioning – Divides data based on value ranges.

1. List Partitioning – Divides data based on specific column values.

2. Hash Partitioning – Distributes data evenly using a hash function.

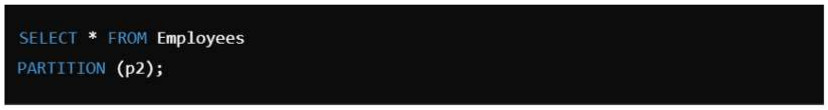
3. Composite Partitioning – Combination of Range and Hash partitioning

Syntax (MySQL Range Partitioning):



```
CREATE TABLE Employees (
    EmpID INT,
    Name VARCHAR(50),
    JoiningDate DATE
)
PARTITION BY RANGE (YEAR(JoiningDate)) (
    PARTITION p1 VALUES LESS THAN (2022),
    PARTITION p2 VALUES LESS THAN (2023),
    PARTITION p3 VALUES LESS THAN (2024)
);
```

xample Query:



```
SELECT * FROM Employees
PARTITION (p2);
```

120. What is a Recursive Common Table Expression (CTE) and how is it used?

A **Recursive CTE** is a SQL construct that allows a query to reference itself. It is used to traverse **hierarchical data** such as organization charts, folder structures, or parent–child relationships.

Structure

```
WITH RECURSIVE cte_name (column_list) AS (
   -- Anchor member: base query
   SELECT ...

   UNION ALL

   -- Recursive member: refers back to the CTE
   SELECT ...
   FROM cte_name
   JOIN ...
)
SELECT * FROM cte_name;
```

Use Case Example — Fetch all subordinates of a manager

```
WITH RECURSIVE EmployeeHierarchy AS (
   SELECT employee_id, manager_id, name
   FROM Employees
   WHERE manager_id IS NULL     -- Top-level manager

   UNION ALL

   SELECT e.employee_id, e.manager_id, e.name
   FROM Employees e
   INNER JOIN EmployeeHierarchy eh
      ON e.manager_id = eh.employee_id
)
```

```
SELECT * FROM EmployeeHierarchy;
```

This starts from the top-level manager and recursively retrieves all employees in the hierarchy.

121. How do you calculate the median in SQL?
SQL has **no built-in MEDIAN() function**, so we use window functions.
Example:

```
SELECT AVG(salary) AS median_salary
FROM (
   SELECT salary,
         ROW_NUMBER() OVER (ORDER BY salary) AS row_num,
         COUNT(*) OVER () AS total_rows
   FROM Employees
) AS ordered_salaries
WHERE row_num IN ((total_rows + 1) / 2,
                 (total_rows + 2) / 2);
```

This handles both even and odd row counts.

122. How can you identify and remove duplicate records in SQL?
Identify duplicates

```
SELECT column1, column2, COUNT(*)
FROM TableName
GROUP BY column1, column2
HAVING COUNT(*) > 1;
```

Remove duplicates (PostgreSQL example)
Keep only one row (using ctid):

```
DELETE FROM TableName
WHERE ctid NOT IN (
   SELECT MIN(ctid)
   FROM TableName
   GROUP BY column1, column2
);
```

Note: Use primary key instead of ctid in MySQL or SQL Server.

123. What is the MERGE (UPSERT) statement in SQL?
MERGE performs **Insert + Update** in one atomic operation.
Syntax

```
MERGE INTO target_table AS target
USING source_table AS source
ON target.id = source.id
WHEN MATCHED THEN
   UPDATE SET target.column = source.column
WHEN NOT MATCHED THEN
   INSERT (id, column)
   VALUES (source.id, source.column);
```

Useful for data warehousing, CDC (Change Data Capture), and sync jobs.

124 How do you implement auditing in SQL?
Use **triggers** to capture inserts, updates, deletes.
Example

```
CREATE TRIGGER AuditEmployeeChanges
AFTER INSERT, UPDATE, DELETE ON Employees
```

```
FOR EACH ROW
BEGIN
   INSERT INTO AuditLog (change_type, changed_by, change_time)
   VALUES ('UPDATE', CURRENT_USER, NOW());
END;
```

Tracks who changed what and when.

125. What is the purpose of GROUPING SETS in SQL?
GROUPING SETS allows multiple groupings in a single GROUP BY.
Example

```
SELECT department, product, SUM(sales)
FROM Sales
GROUP BY GROUPING SETS (
   (department),
   (product),
   (department, product)
);
```

Equivalent to running three separate queries, but more efficient.

126. How do you implement row-level security in SQL?
Row-level security restricts which **rows** a user can see.
SQL Server Example

```
CREATE FUNCTION fn_securitypredicate(@UserID INT)
RETURNS TABLE WITH SCHEMABINDING AS
RETURN SELECT 1 AS result WHERE @UserID = USER_ID();


CREATE SECURITY POLICY EmployeeSecurity
ADD FILTER PREDICATE fn_securitypredicate(UserID)
ON Employees
WITH (STATE = ON);
```

Each user only sees rows matching their ID.

127. How can you schedule SQL tasks (like backups or refresh jobs)?
Options

- **SQL Server:** SQL Server Agent
- **PostgreSQL/MySQL:** CRON + Shell Scripts
- **Oracle:** DBMS_SCHEDULER

SQL Server Example

```
EXEC sp_add_job ...
EXEC sp_add_jobstep ...
EXEC sp_add_schedule ...
EXEC sp_attach_schedule ...
```

Used for:

- Refreshing materialized views
- Nightly ETL
- Automating reports

128. What are stored functions and how are they different from stored procedures?

Feature	Stored Functions	Stored Procedures
Return Value	Must return a single value	No mandatory return value

Callable in SELECT	Yes	No
Usage	Calculations, formatting, reusable logic	Workflows, batch operations

Example (Function)

```
CREATE FUNCTION GetEmployeeAge(@birthdate DATE)
RETURNS INT AS
BEGIN
  RETURN DATEDIFF(YEAR, @birthdate, GETDATE());
END;
```

129. How does ON DELETE CASCADE differ from ON DELETE SET NULL?

Constraint	Behavior
ON DELETE CASCADE	Deleting parent row deletes child rows
ON DELETE SET NULL	Child row remains; FK becomes NULL

When to choose which?

- **CASCADE** → If child rows are meaningless without parent (invoice → invoice_lines)
- **SET NULL** → If orphaned child rows should remain but be disassociated