[**Interview Questions SQL Server**](http://www.aspdotnet-suresh.com/2010/05/interview-questions-in-aspnetcnetsql.html)

**Difference:**

**=> Unioun and Unioun All**

UNION ALL keeps all of the records from each of the original data sets, UNION removes any duplicate records. UNION first performs a sorting operation and eliminates of the records that are duplicated across all columns before finally returning the combined data set.

UNION removes duplicate records (where all columns in the results are the same), UNION ALL does not.

There is a performance hit when using UNION instead of UNION ALL, since the database server must do additional work to remove the duplicate rows, but usually you do not want the duplicates (especially when developing reports).

Both UNION and UNION ALL concatenate the result of two different SQLs. They differ in the way they handle duplicates.

* UNION performs a DISTINCT on the result set, eliminating any duplicate rows.
* UNION ALL does not remove duplicates, and it therefore faster than UNION.

**=>Stored Procedure VS Function**

1) Procedure can return zero or n values whereas function can return one value which is mandatory.  
2) Procedures can have input, output parameters for it whereas functions can have only input parameters.  
3) Procedure allows select as well as DML statement in it whereas function allows only select statement in it.  
4) Functions can be called from procedure whereas procedures cannot be called from function.  
5) Exception can be handled by try-catch block in a procedure whereas try-catch block cannot be used in a function.  
6) We can go for transaction management in procedure whereas we can't go in function.  
7) Procedures cannot be utilized in a select statement whereas function can be embedded in a select statement.

**=>nvarchar VS varchar**

**varchar**: Variable-length, non-Unicode character data. The database collation determines which code page the data is stored using.

**nvarchar**: Variable-length Unicode character data. Dependent on the database collation for comparisons.

**=>Truncate, Delete and Drop**

1) Delete keep the lock over each row where Truncate keeps the lock on table not on all the row.  
2) Counter of the Identity column is reset in Truncate where it is not reset in Delete.   
3) Trigger is not fired in Truncate where as trigger is fired in Delete.

4) In TRUNCATE we cannot rollback.

5) In DELETE we can rollback

**=>Primary key VS Unique key**

Primary key does not allow the null values but unique key allows one null value.

Primary key will create clustered index on column but unique key will create non-clustered index by default.

| **Primary Key** | **Unique Key** |
| --- | --- |
| Unique identifier for rows of a table | Unique identifier for rows of a table when primary key is not present |
| Cannot be NULL | Can be NULL |
| Only one primary key can be present in a table | Multiple Unique Keys can be present in a table |
| present in a table | present in a table |
| Selection using primary key creates clustered index | Selection using unique key creates non-clustered index |

**=>temp table VS temp variable VS Gloable table**

* Temporary Tables are real tables so you can do things like CREATE INDEXes, etc. If you have large amounts of data for which accessing by index will be faster then temporary tables are a good option.
* Table variables can have indexes by using PRIMARY KEY or UNIQUE constraints. (If you want a non-unique index just include the primary key column as the last column in the unique constraint. If you don't have a unique column, you can use an identity column.) [SQL 2014 has non-unique indexes too](https://stackoverflow.com/questions/886050/sql-server-creating-an-index-on-a-table-variable/17385085#17385085).
* Table variables don't participate in transactions and SELECTs are implicitly with NOLOCK. The transaction behaviour can be very helpful, for instance if you want to ROLLBACK midway through a procedure then table variables populated during that transaction will still be populated!
* Temp tables might result in stored procedures being recompiled, perhaps often. Table variables will not.
* You can create a temp table using SELECT INTO, which can be quicker to write (good for ad-hoc querying) and may allow you to deal with changing datatypes over time, since you don't need to define your temp table structure upfront.
* You can pass table variables back from functions, enabling you to encapsulate and reuse logic much easier (eg make a function to split a string into a table of values on some arbitrary delimiter).
* Using Table Variables within user-defined functions enables those functions to be used more widely (see CREATE FUNCTION documentation for details). If you're writing a function you should use table variables over temp tables unless there's a compelling need otherwise.
* Both table variables and temp tables are stored in tempdb. But table variables (since 2005) default to the collation of the current database versus temp tables which take the default collation of tempdb ([ref](https://docs.microsoft.com/sql/t-sql/language-elements/declare-local-variable-transact-sql)). This means you should be aware of collation issues if using temp tables and your db collation is different to tempdb's, causing problems if you want to compare data in the temp table with data in your database.
* Global Temp Tables (##tmp) are another type of temp table available to all sessions and users

**=>RANK() VS DENSE\_RANK()**

RANK gives you the ranking within your ordered partition. Ties are assigned the same rank, with the next ranking(s) skipped. So, if you have 3 items at rank 2, the next rank listed would be ranked 5.

DENSE\_RANK again gives you the ranking within your ordered partition, but the ranks are consecutive. No ranks are skipped if there are ranks with multiple items.

As for nulls, it depends on the ORDER BY clause. Here is a simple test script you can play with to see what happens

**=>WHERE VS HAVING**

**WHERE Clause:**   
WHERE Clause is used to filter the records from the table or used while joining more than one table.Only those records will be extracted who are satisfying the specified condition in WHERE clause. It can be used with SELECT, UPDATE, DELETE statements

**HAVING Clause:** HAVING Clause is used to filter the records from the groups based on the given condition in the HAVING Clause. Those groups who will satisfy the

**Difference between Where and Having Clause in SQL :**

| SR.NO. | WHERE Clause | HAVING Clause |
| --- | --- | --- |
| 1. | WHERE Clause is used to filter the records from the table based on the specified condition. | HAVING Clause is used to filter record from the groups based on the specified condition. |
| 2. | WHERE Clause can be used without GROUP BY Clause | HAVING Clause cannot be used without GROUP BY Clause |
| 3. | WHERE Clause implements in row operations | HAVING Clause implements in column operation |
| 4. | WHERE Clause cannot contain aggregate function | HAVING Clause can contain aggregate function |
| 5. | WHERE Clause can be used with SELECT, UPDATE, DELETE statement. | HAVING Clause can only be used with SELECT statement. |
| 6. | WHERE Clause is used before GROUP BY Clause | HAVING Clause is used after GROUP BY Clause |
| 7. | WHERE Clause is used with single row function like UPPER, LOWER etc. | HAVING Clause is used with multiple row function like SUM, COUNT etc. |

**=>Index types in SQL Server or Different between cluster index and non clusete index?**

**Clustered Index**

Only 1 allowed per table physically rearranges the data in the table to confirm to the index constraints for use on columns that are frequently searched for ranges of data for use on columns with low selectivity.

**Non-Clustered Index**

Up to 249 allowed per table creates a separate list of key values with pointers to the location of the data in the data pages For use on columns that are searched for single values

A clustered index is a special type of index that reorders the way records in the table are physically stored. Therefore table can have only one clustered index. The leaf nodes of a clustered index contain the data pages. A non-clustered index is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk. The leaf node of a non-clustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.

**Included Column Index (New in SQL Server 2005)**

In SQL Server 2005, the functionality of non-clustered indexes is extended by adding non-key columns to the leaf level of the non-clustered index. Non-key columns can help to create cover indexes. By including non-key columns, you can create non-clustered indexes that cover more queries. The Database Engine does not consider non-key columns when calculating the number of index key columns or index key size. Non-key columns can be included in non-clustered index to avoid exceeding the current index size limitations of a maximum of 16 key columns and a maximum index key size of 900 bytes. Another advantage is that using non-key column in index we can have index data types not allowed as index key columns generally.

In following example column Filename is varchar(400), which will increase the size of the index key bigger than it is allowed. If we still want to include in our cover index to gain performance we can do it by using the Keyword INCLUDE.

USE AdventureWorks  
GO  
CREATE INDEX IX\_Document\_Title  
ON Production.Document (Title, Revision)  
INCLUDE (FileName)

Non-key columns can be included only in non-clustered indexes. Columns can’t be defined in both the key column and they INCLUDE list. Column names can’t be repeated in the INCLUDE list. Non-key columns can be dropped from a table only after the non-key index is dropped first. For Included Column Index to exist there must be at least one key column defined with a maximum of 16 key columns and 1023 included columns.

Avoid adding unnecessary columns. Adding too many index columns, key or non-key as they will affect negatively on performance. Fewer index rows will fit on a page. This could create I/O increases and reduced cache efficiency. More disk space will be required to store the index. Index maintenance may increase the time that it takes to perform modifications, inserts, updates, or deletes, to the underlying table or indexed view.

**Another example to test:**

Create following Index on Database AdventureWorks in SQL SERVER 2005  
  
USE AdventureWorks  
GO  
CREATE NONCLUSTERED INDEX IX\_Address\_PostalCode  
ON Person.Address (PostalCode)  
INCLUDE (AddressLine1, AddressLine2, City, StateProvinceID)  
GO   
  
Test the performance of following query before and after creating Index. The performance improvement is significant.

SELECT AddressLine1, AddressLine2, City, StateProvinceID, PostalCode  
FROM Person.Address  
WHERE PostalCode BETWEEN '98000'  
AND '99999';  
GO

**=>What is trigger and different types of Triggers?**

https://www.c-sharpcorner.com/article/what-is-trigger-what-are-the-types-of-triggers-part-1/

**=>What is Cursor?**

https://www.geeksforgeeks.org/what-is-cursor-in-sql//

**=>What are Advantages of Stored Procedures?**

https

**=>What is a join?types?**

https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/

A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are:

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN

**=>What is Self-Join and Cross-Join?**

<https://www.geeksforgeeks.org/self-join-and-cross-join-in-ms-sql-server/>

**=>What are Constraints in SQL?**

<https://www.geeksforgeeks.org/sql-constraints/>

**=>What are types functions?**

<https://www.dotnettricks.com/learn/sqlserver/different-types-of-sql-server-functions>

**=>What is CTE?**

https://www.geeksforgeeks.org/cte-in-sql/

**=>What is Primary Key,Foreign Key,Union Key**

<https://www.geeksforgeeks.org/difference-between-primary-key-and-unique-key/>

**=>What are aggregate and scalar functions?**

<https://www.geeksforgeeks.org/sql-functions-aggregate-scalar-functions/>

**=>What are SQL Injections? And How to Prevent SQL Injection Attacks?**

https

**=>What are the type of locks ?**

<https://www.sqlshack.com/locking-sql-server/>

**=>What is ACID?**

<https://www.geeksforgeeks.org/acid-properties-in-dbms/>

**=>Offset fetch**

<https://www.geeksforgeeks.org/sql-offset-fetch-clause/>

**=>What is set Nocount ON?**

<https://www.sqlshack.com/set-nocount-on-statement-usage-and-performance-benefits-in-sql-server/>

**=>SQL query execution order**

https

**=>Improve performance of SQL**

https

**=>To display data what will you do to bring milliouncs of data in SQL database table**

https

**=>Execution plan in SQL**

https

**=>Find 2nd or 4th Highest salary**