## What are the two authentication modes in SQL Server?

There are two authentication modes –

* Windows Mode
* Mixed Mode

Modes can be changed by selecting the tools menu of SQL Server configuration properties and choose security page.

## What Is SQL Profiler?

SQL Profiler is a tool which allows system administrator to monitor events in the SQL server.  This is mainly used to capture and save data about each event of a file or a table for analysis.

## What is recursive stored procedure?

SQL Server supports recursive stored procedure which calls by itself. Recursive stored procedure can be defined as a method of problem solving wherein the solution is arrived repetitively. It can nest up to 32 levels.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | CREATE PROCEDURE [dbo].[Fact]  (  @Number Integer,  @RetVal Integer OUTPUT  )  AS  DECLARE @In Integer  DECLARE @Out Integer  IF @Number != 1  BEGIN  SELECT @In = @Number – 1  EXEC Fact @In, @Out OUTPUT - Same stored procedure has been called again(Recursively)  SELECT @RetVal = @Number \* @Out  END  ELSE  BEGIN  SELECT @RetVal = 1  END  RETURN  GO |

## What are the differences between local and global temporary tables?

There are two types of temporary tables: local and global. Local temporary tables are visible only to their creators during the same connection to an instance of SQL Server as when the tables were first created or referenced. Local temporary tables are deleted after the user disconnects from the instance of SQL Server. Global temporary tables are visible to any user and any connection after they are created, and are deleted when all users that are referencing the table disconnect from the instance of SQL Server.

Local Table:

|  |  |
| --- | --- |
| 1 | CREATE TABLE #<tablename> |

Global Table:

|  |  |
| --- | --- |
| 1 | CREATE TABLE ##<tablename> |

## [What is constraints and how many types of constraints in sql server](http://sqllearnergroups.blogspot.com/2013/02/what-is-constraints-and-how-many-types.html) Constraint are using to restrict the insertion of unwanted data in any columns. we can create constraints on single or multiple columns of any table. It maintain the data integrity of  the table. **There are 6 types of constraints in Sql Server**:-

1. Primary key constraint.
2. Foreign Key constraint.
3. Unique Key constraint.
4. Not Null constraint.
5. Check constraint
6. Default constraint

Description of  above constraints:-  
  
**1. Primary Key :-** Primary Key of a relational table uniquely identifies each record in the table. It can be either be a normal attribute that is guaranteed to be unique such as in a school name should be same of any student but roll number never be same of any student in a school.

**2. Foreign Key:-**One of the most important concept in database is creating relationships between  database tables. these relationships provide a mechanism for linking data stored in multiple tables and retriving it in an efficient manner. in order to create a link between two tables we must specify a foreign key in one table that references a column in another table**.**

**3. Unique Key:-**  Unique key constraint is use to make sure that there is no duplicate value in that column. **Both unique key and primary key both enforces the uniqueness of column but there is one difference between them unique key constraint allow one null value but primary key does not null value**.In a table we create one primary key but we can create more than one unique key in Sql Server.

**5. Check Constraint :-**  This constraint is using to check value at the time of insertion like as salary of any employee is always greater than zero. so we can create a check constraint on employee table which is greater than zero.

**6. Default Constraint:-** The Default constraint is using to set a specific value of column if we not passing the value at the time of insertion. Through this constraint we set the default value of column.

## Can SQL servers linked to other servers?

SQL server can be connected to any database which has OLE-DB provider to give a link. Example: Oracle has OLE-DB provider which has link to connect with the SQL server group.

## What is sub query and its properties?

A sub-query is a query which can be nested inside a main query like Select, Update, Insert or Delete statements. This can be used when expression is allowed. Properties of sub query can be defined as

* A sub query should not have order by clause
* A sub query should be placed in the right hand side of the comparison operator of the main query
* A sub query should be enclosed in parenthesis because it needs to be executed first before the main query
* More than one sub query can be included

## What are the types of sub query?

There are three types of sub query –

* Single row sub query which returns only one row
* Multiple row sub query which returns multiple rows
* Multiple column sub query which returns multiple columns to the main query. With that sub query result, Main query will be executed.

## What is Common Table Expressions

A common table expression (CTE) can be thought of as a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query. Unlike a derived table, a CTE can be self-referencing and can be referenced multiple times in the same query.

A CTE can be used to:

* Create a recursive query. For more information, see [Recursive Queries Using Common Table Expressions](https://technet.microsoft.com/en-us/library/ms186243(v=sql.105).aspx).
* Substitute for a view when the general use of a view is not required; that is, you do not have to store the definition in metadata.
* Enable grouping by a column that is derived from a scalar subselect, or a function that is either not deterministic or has external access.
* Reference the resulting table multiple times in the same statement.

Using a CTE offers the advantages of improved readability and ease in maintenance of complex queries. The query can be divided into separate, simple, logical building blocks. These simple blocks can then be used to build more complex, interim CTEs until the final result set is generated.

CTEs can be defined in user-defined routines, such as functions, stored procedures, triggers, or views.

It’s a headache for developers to write or read a complex SQL query using a number of Joins. Complex SQL statements can be made easier to understand and maintainable in the form of CTE or Common Table expressions. In this post, I will try to cover some of the benefits that we get when working with CTE.

When dealing with sub-queries, it is often required that you need to select a part of the data from a sub query or even join data from a query with some other tables. In that case, either you have an option to name your sub-queries with an alias or to use it directly. Gradually your requirement is getting more and more complex and your query would look unmaintainable at any time. CTE allows you to define the subquery at once, name it using an alias and later call the same data using the alias just like what you do with a normal table. CTE is standard ANSI SQL standard.

With T(Address, Name, Age) *--Column names for Temporary table*

AS

(

SELECT A.Address, E.Name, E.Age from Address A

INNER JOIN EMP E ON E.EID = A.EID

)

SELECT \* FROM T *--SELECT or USE CTE temporary Table*

WHERE T.Age > 50

ORDER BY T.NAME

## What are scheduled tasks(jobs) in SQL Server?

Scheduled tasks or jobs are used to automate processes that can be run on a scheduled time at a regular interval. This scheduling of tasks helps to reduce human intervention during night time and feed can be done at a particular time. User can also order the tasks in which it has to be generated.

(This question is refer to below question.)

## What is SQL server agent?

The SQL Server agent plays a vital role in day to day tasks of SQL server administrator(DBA). Server agent’s purpose is to implement the tasks easily with the scheduler engine which allows our jobs to run at scheduled date and time.

## What is COALESCE in SQL Server?

COALESCE is used to return first non-null expression within the arguments. This function is used to return a non-null from more than one column in the arguments.

Example –

|  |  |
| --- | --- |
| 1 | Select COALESCE(empno, empname, salary) from employee; |

## How exceptions can be handled in SQL Server Programming?

Exceptions are handled using TRY—-CATCH constructs and it is handles by writing scripts inside the TRY block and error handling in the CATCH block.

## What is the purpose of FLOOR function?

FLOOR function is used to round up a non-integer value to the previous least integer. Example is given

|  |  |
| --- | --- |
| 1 | FLOOR(6.7) |

Returns 6.

## Can we check locks in database? If so, how can we do this lock check?

Yes, we can check locks in the database. It can be achieved by using in-built stored procedure called sp\_lock.

## What is the use of SIGN function?

SIGN function is used to determine whether the number specified is Positive, Negative and Zero. This will return +1,-1 or 0.

Example –

MySQL

|  |  |
| --- | --- |
| 1 | SIGN(-35) returns -1 |

## What is a Trigger?

Triggers are used to execute a batch of SQL code when insert or update or delete commands are executed against a table. Triggers are automatically triggered or executed when the data is modified. It can be executed automatically on insert, delete and update operations.

Example

1. -- Create trigger on table Employee\_Demo for Insert statement
2. CREATE TRIGGER trgAfterInsert on Employee\_Demo
3. FOR INSERT
4. AS declare @empid int, @empname varchar(55), @empsal decimal(10,2), @audit\_action varchar(100);
5. select @empid=i.Emp\_ID from inserted i;
6. select @empname=i.Emp\_Name from inserted i;
7. select @empsal=i.Emp\_Sal from inserted i;
8. set @audit\_action='Inserted Record -- After Insert Trig
9. ger.'; insert into Employee\_Demo\_Audit(Emp\_ID,Emp\_Name,Emp\_Sal,Audit\_Action,Audit\_Timestamp)
10. values (@empid,@empname,@empsal,@audit\_action,getdate());
11. PRINT 'AFTER INSERT trigger fired.'
12. --Output will be

## What are the types of Triggers?

There are four types of triggers and they are:

* Insert
* Delete
* Update
* Instead of

## What is an IDENTITY column in insert statements?

IDENTITY column is used in table columns to make that column as Auto incremental number or a surrogate key.

## What is Bulkcopy in SQL?

Bulk copy is a tool used to copy large amount of data from Tables. This tool is used to load large amount of data in SQL Server.

## What will be query used to get the list of triggers in a database?

Query to get the list of triggers in database-

MySQL

|  |  |
| --- | --- |
| 1 | Select \* from sys.objects where type=’tr’ |

## What is the difference between UNION and UNION ALL?

## The SQL UNION Operator

The UNION operator is used to combine the result-set of two or more SELECT statements.

Notice that each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.

## SQL UNION Syntax

SELECT column\_name(s) FROM table1  
UNION  
SELECT column\_name(s) FROM table2;

**Note:** The UNION operator selects only distinct values by default. To allow duplicate values, use the ALL keyword with UNION.

SQL UNION ALL Syntax

SELECT column\_name(s) FROM table1  
UNION ALL  
SELECT column\_name(s) FROM table2;

**PS:** The column names in the result-set of a UNION are usually equal to the column names in the first SELECT statement in the UNION.

## Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

## 

## SQL UNION Example

The following SQL statement selects all the **different** cities (only distinct values) from the "Customers" and the "Suppliers" tables:

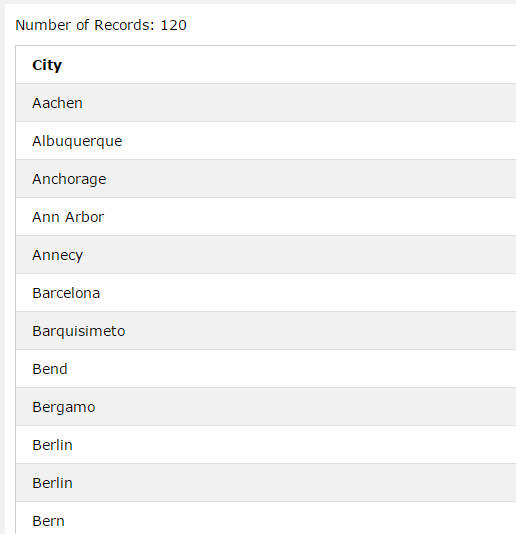
SELECT City FROM Customers  
UNION  
SELECT City FROM Suppliers  
ORDER BY City;



## SQL UNION ALL Example

The following SQL statement uses UNION ALL to select **all** (duplicate values also) cities from the "Customers" and "Suppliers" tables:

SELECT City FROM Customers  
UNION ALL  
SELECT City FROM Suppliers  
ORDER BY City;



## How Global temporary tables are represented and its scope?

Global temporary tables are represented with ## before the table name. Scope will be the outside the session whereas local temporary tables are inside the session. Session ID can be found using @@SPID.

## What are the differences between Stored Procedure and the dynamic SQL?

Stored Procedure is a set of statements which is stored in a compiled form. Dynamic SQL is a set of statements that dynamically constructed at runtime and it will not be stored in a Database and it simply execute during run time.

## What is Collation?

Collation is defined to specify the sort order in a table. There are three types of sort order –

1. Case sensitive
2. Case Insensitive
3. Binary

## How can we get count of the number of records in a table?

Following are the queries can be used to get the count of records in a table –

|  |  |
| --- | --- |
| 1 | Select \* from <tablename> Select count(\*) from <tablename> Select rows from sysindexes where id=OBJECT\_ID(tablename) and indid<2 |

## What is the command used to get the version of SQL Server?

|  |  |
| --- | --- |
| 1 | Select SERVERPROPERTY(‘productversion’) |

is used to get the version of SQL Server.

## What is UPDATE\_STATISTICS command?

UPDATE\_STATISTICS command is used to update the indexes on the tables when there is a large amount of deletions or modifications or bulk copy occurred in indexes.

## What is the use of SET NOCOUNT ON/OFF statement?

By default, NOCOUNT is set to OFF and it returns number of records got affected whenever the command is getting executed. If the user doesn’t want to display the number of records affected, it can be explicitly set to ON- (SET NOCOUNT ON).

## Which SQL server table is used to hold the stored procedure scripts?

Sys.SQL\_Modules is a SQL Server table used to store the script of stored procedure. Name of the stored procedure is saved in the table called Sys.Procedure.

## What are Magic Tables in SQL Server?

Magic tables are nothing but inserted and deleted which are temporary objects created by the server internally to hold recently inserted values in the case of insert and to hold recently deleted values in the case of delete, to hold before updating values or after updating values in the case of update.

## What is the difference between SUBSTR and INSTR in the SQL Server?

The SUBSTR function is used to return specific portion of string in a given string. But, INSTR function gives character position in a given specified string.

|  |  |
| --- | --- |
| 1 | SUBSTR(“Smiley”,3) |

Gives result as Smi

|  |  |
| --- | --- |
| 1 | INSTR(“Smiley”,’i’,1) |

Gives 3 as result as I appears in 3rd position of the string

## What is the use of =,==,=== operators?

= is used to assign one value or variable to another variable. == is used for comparing two strings or numbers. === is used to compare only string with the string and number with numbers.

## What is ISNULL() operator?

ISNULL function is used to check whether value given is NULL or not NULL in sql server. This function also provides to replace a value with the NULL.

## What is the use of FOR Clause?

FOR clause is mainly used for XML and browser options. This clause is mainly used to display the query results in XML format or in browser.

## What will be the maximum number of index per table?

For SQL Server 2008 1000 Index can be used as maximum number per table. 1 Clustered Index and 999 Non-clustered indexes per table can be used in SQL Server.

## What is the difference between COMMIT and ROLLBACK?

Every statement between BEGIN and COMMIT becomes persistent to database when the COMMIT is executed. Every statement between BEGIN and ROOLBACK are reverted to the state when the ROLLBACK was executed.

## What is the difference between varchar and nvarchar types?

Varchar and nvarchar are same but the only difference is that nvarhcar can be used to store Unicode characters for multiple languages and it also takes more space when compared with varchar.

## What is the use of @@SPID?

A @@SPID returns the session ID of the current user process.

## What is the command used to Recompile the stored procedure at run time?

Stored Procedure can be executed with the help of keyword called RECOMPILE.

|  |  |
| --- | --- |
| 1 | Exe <SPName>  WITH RECOMPILE |

Or we can include WITHRECOMPILE in the stored procedure itself.

## How to delete duplicate rows in SQL Server?

Duplicate rows can be deleted using CTE and ROW NUMER feature of SQL Server.

## Where are SQL Server user names and passwords stored in SQL Server?

User Names and Passwords are stored in sys.server\_principals and sys.sql\_logins. But passwords are not stored in normal text.

## What is the difference between GETDATE and SYSDATETIME?

Both are same but GETDATE can give time till milliseconds and SYSDATETIME can give precision till nanoseconds. SYSDATE TIME is more accurate than GETDATE.

## How data can be copied from one table to another table? (It is bulk copy)

INSERT INTO SELECT

This command is used to insert data into a table which is already created.

SELECT INTO

This command is used to create a new table and its structure and data can be copied from existing table.

## What is TABLESAMPLE?

TABLESAMPLE is used to extract sample of rows randomly that are all necessary for the application. The sample rows taken are based on the percentage of rows.

## Which command is used for user defined error messages?

RAISEERROR is the command used to generate and initiates error processing for a given session. Those user defined messages are stored in sys.messages table.

## What do mean by XML Datatype?

XML data type is used to store XML documents in the SQL Server database. Columns and variables are created and store XML instances in the database.

## What is CDC?

CDC is abbreviated as Change Data Capture which is used to capture the data that has been changed recently. This feature is present in SQL Server 2008.

## What is SQL injection?

SQL injection is an attack by malicious users in which malicious code can be inserted into strings that can be passed to an instance of SQL server for parsing and execution. All statements have to checked for vulnerabilities as it executes all syntactically valid queries that it receives.

Even parameters can be manipulated by the skilled and experienced attackers.

## What are the methods used to protect against SQL injection attack?

Following are the methods used to protect against SQL injection attack:

* Use Parameters for Stored Procedures
* Filtering input parameters
* Use Parameter collection with Dynamic SQL
* In like clause, user escape characters

## What is Filtered Index?

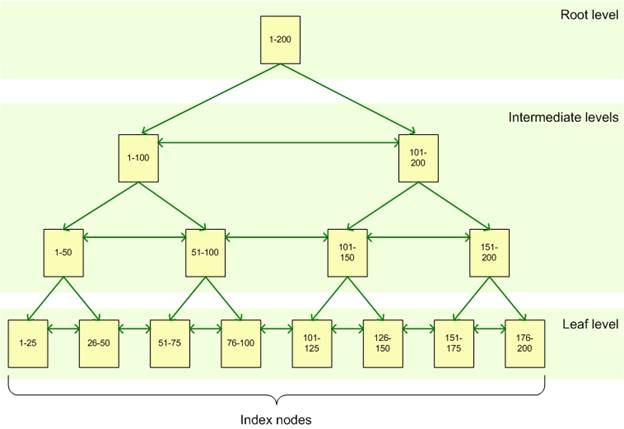
Filtered Index is used to filter some portion of rows in a table to improve query performance, index maintenance and reduces index storage costs. When the index is created with WHERE clause, then it is called Filtered Index

## Indexes in SQL Server

One of the most important routes to high performance in a SQL Server database is the index. Indexes speed up the querying process by providing swift access to rows in the data tables, similarly to the way a book’s index helps you find information quickly within that book.

Indexes are created on columns in tables or views. The index provides a fast way to look up data based on the values within those columns. For example, if you create an index on the primary key and then search for a row of data based on one of the primary key values, SQL Server first finds that value in the index, and then uses the index to quickly locate the entire row of data. Without the index, a table scan would have to be performed in order to locate the row, which can have a significant effect on performance.

An index is made up of a set of pages (index nodes) that are organized in a B-tree structure. This structure is hierarchical in nature, with the root node at the top of the hierarchy and the leaf nodes at the bottom, as shown in Figure



When a query is issued against an indexed column, the query engine starts at the root node and navigates down through the intermediate nodes, with each layer of the intermediate level more granular than the one above. The query engine continues down through the index nodes until it reaches the leaf node. For example, if you’re searching for the value 123 in an indexed column, the query engine would first look in the root level to determine which page to reference in the top intermediate level. In this example, the first page points the values 1-100, and the second page, the values 101-200, so the query engine would go to the second page on that level. The query engine would then determine that it must go to the third page at the next intermediate level. From there, the query engine would navigate to the leaf node for value 123. The leaf node will contain either the entire row of data or a pointer to that row, depending on whether the index is clustered or nonclustered.

* Clustered
  + Clustered indexes sort and store the data rows in the table or view based on their key values. These are the columns included in the index definition. There can be only one clustered index per table, because the data rows themselves can be sorted in only one order.
  + The only time the data rows in a table are stored in sorted order is when the table contains a clustered index. When a table has a clustered index, the table is called a clustered table. If a table has no clustered index, its data rows are stored in an unordered structure called a heap.
* Nonclustered
  + Nonclustered indexes have a structure separate from the data rows. A nonclustered index contains the nonclustered index key values and each key value entry has a pointer to the data row that contains the key value.
  + The pointer from an index row in a nonclustered index to a data row is called a row locator. The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table. For a heap, a row locator is a pointer to the row. For a clustered table, the row locator is the clustered index key.
  + You can add nonkey columns to the leaf level of the nonclustered index to by-pass existing index key limits, 900 bytes and 16 key columns, and execute fully covered, indexed, queries.

## Define Normalization and De- Normalization.

**Normalization**  
- It is the process of organizing data into related table.  
- To normalize database, we divide database into tables and establish relationships between the tables.  
- It reduces redundancy. It is done to improve performance of query.  
- Steps of normalization:  
  
**First Normal form**  
-Entities of the table must have unique identifier or entity key.   
**Second Normal Form**  
- All the attributes of the table must depend on the entity key for that entity.   
**Third Normal Form**  
-All attributes that are not part of the key must not depend on any other non-key attributes.  
  
**De-normalization**  
The process of adding redundant data to get rid of complex join, in order to optimize database performance. This is done to speed up database access by moving from higher to lower form of normalization.

## Removing Duplicates from a Table in SQL Server

Checking for Duplicates

On any version of SQL Server, you can identify duplicates using a simple query, with **GROUP BY** and **HAVING**, as follows:

|  |  |
| --- | --- |
|  | DECLARE @table TABLE (data VARCHAR(20))  INSERT INTO @table VALUES ('not duplicate row')  INSERT INTO @table VALUES ('duplicate row')  INSERT INTO @table VALUES ('duplicate row')    SELECT  data        , COUNT(data) nr  FROM    @table  GROUP BY data  HAVING  COUNT(data) > 1 |

The result indicates that there are two occurrences of the row containing the “duplicate row” text:

|  |  |
| --- | --- |
|  | data                 nr  -------------------- -----------  duplicate row        2 |

## Removing Duplicate Rows in SQL Server

SQL Server 2005 has introduced the **row\_number()** function, which provides an alternative means of identifying duplicates. Rewriting the first example, for tables with no primary key, we can now assign a row number to each row in a duplicate group, with a command such as:

|  |  |
| --- | --- |
|  | DECLARE  @duplicateTable4 TABLE (data VARCHAR(20))  INSERT INTO @duplicateTable4 VALUES ('not duplicate row')  INSERT INTO @duplicateTable4 VALUES ('duplicate row')  INSERT INTO @duplicateTable4 VALUES ('duplicate row')  INSERT INTO @duplicateTable4 VALUES ('second duplicate row')  INSERT INTO @duplicateTable4 VALUES ('second duplicate row')    SELECT  data        , row\_number() OVER ( PARTITION BY data ORDER BY data ) AS nr  FROM    @duplicateTable4 |

The result will show:

|  |  |
| --- | --- |
|  | data                 nr  -------------------- --------------------  duplicate row        1  duplicate row        2  not duplicate row    1  second duplicate row 1  second duplicate row 2 |

In the above example, we specify an ordering and partitioning for the **row\_number()** function. Note that the **row\_number**() is a ranking window function, therefore the **ORDER BY** and the **PARTITION BY**in the **OVER** clause are used only to determine the value for the **nr**column, and they do not affect the row order of the query. Also, while the above is similar to our previous **GROUP BY** clause, there is a big difference concerning the returned rows. With **GROUP BY** you must use an aggregate on the columns that are not listed after the **GROUP BY**. With the **OVER** clause there is no such restriction, and you can get access to the individual rows in the groups specified by the **PARTITION BY** clause. This gives us access to the individual duplicate rows, so we can get not only the number of occurrences, but also a sequence number for the individual duplicates. To filter out the duplicate rows only, we could just put the above query into a CTE or a subquery. The CTE approach is as follows:

|  |  |
| --- | --- |
|  | DECLARE  @duplicateTable4 TABLE (data VARCHAR(20))  INSERT INTO @duplicateTable4 VALUES ('not duplicate row')  INSERT INTO @duplicateTable4 VALUES ('duplicate row')  INSERT INTO @duplicateTable4 VALUES ('duplicate row')  INSERT INTO @duplicateTable4 VALUES ('second duplicate row')  INSERT INTO @duplicateTable4 VALUES ('second duplicate row')  ;  WITH    numbered            AS ( SELECT   data                        , row\_number() OVER ( PARTITION BY data ORDER BY data ) AS nr                 FROM     @duplicateTable4               )      SELECT  data      FROM    numbered      WHERE   nr > 1 |

## Difference between NULLIF and ISNULL

## NULLIF:

## NULLIF( expression , expression)

## **It returns a null value if both the expression is equal. NULLIF returns the first expression if both the expression is not equal.**

## ISNULL:

## ISNULL ( expression , replacement)

## **Replaces NULL with the specified replacement value. It returns the value of expression if it is not null.**

## 

## View in SQL Server

Creates a virtual table whose contents (columns and rows) are defined by a query. Use this statement to create a view of the data in one or more tables in the database. For example, a view can be used for the following purposes:

* To focus, simplify, and customize the perception each user has of the database.
* As a security mechanism by allowing users to access data through the view, without granting the users permissions to directly access the underlying base tables.
* To provide a backward compatible interface to emulate a table whose schema has changed.

## Find the nth highest salary using the TOP keyword in SQL Server

We can also use the TOP keyword (for databases that support the TOP keyword, like SQL Server) to find the nth highest salary. Here is some fairly simply SQL that would help us do that

SELECT TOP 1 SAL FROM (SELECT DISTINCT TOP 5 SAL FROM @TEMP ORDER BY SAL DESC) AS Emp

ORDER BY SAL

To understand the query above, first look at the subquery, which simply finds the N highest salaries in the Employee table and arranges them in descending order. Then, the outer query will actually rearrange those values in ascending order, which is what the very last line “ORDER BY Salary” does, because of the fact that the [ORDER BY Default](http://www.programmerinterview.com/index.php/database-sql/sql-order-by-default-sort-order/) is to sort values in ASCENDING order. Finally, that means the Nth highest salary will be at the top of the list of salaries, which means we just want the first row, which is exactly what “SELECT TOP 1 Salary” will do for us!

## Joins In SQL

* INNER JOIN (a.k.a. “simple join”): Returns all rows for which there is at least one match in BOTH tables. This is the default type of join if no specific*JOIN*type is specified.
* LEFT JOIN (or LEFT OUTER JOIN): Returns all rows from the left table, and the matched rows from the right table; i.e., the results will contain all records from the left table, even if the JOIN condition doesn’t find any matching records in the right table. This means that if the ON clause doesn’t match any records in the right table, the JOIN will still return a row in the result for that record in the left table, but with NULL in each column from the right table.
* RIGHT JOIN (or RIGHT OUTER JOIN): Returns all rows from the right table, and the matched rows from the left table. This is the exact opposite of a LEFT JOIN; i.e., the results will contain all records from the right table, even if the JOIN condition doesn’t find any matching records in the left table. This means that if the ON clause doesn’t match any records in the left table, the JOIN will still return a row in the result for that record in the right table, but with NULL in each column from the left table.
* FULL JOIN (or FULL OUTER JOIN): Returns all rows for which there is a match in EITHER of the tables. Conceptually, a FULL JOIN combines the effect of applying both a LEFT JOIN and a RIGHT JOIN; i.e., its result set is equivalent to performing a UNION of the results of left and right outer queries.
* CROSS JOIN: Returns all records where each row from the first table is combined with each row from the second table (i.e., returns the Cartesian product of the sets of rows from the joined tables). Note that a CROSS JOIN can either be specified using the CROSS JOIN syntax (“explicit join notation”) or (b) listing the tables in the FROM clause separated by commas without using a WHERE clause to supply join criteria (“implicit join notation”).

## COALESCE (संगठित होना) in sql

The **COALESCE** function in SQL returns the first non-NULL expression among its arguments. The syntax for **COALESCE** is as follows

For examples, say we have the following table,

Table ***Contact\_Info***

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Business\_Phone** | **Cell\_Phone** | **Home\_Phone** |
| Jeff | 531-2531 | 622-7813 | 565-9901 |
| Laura | NULL | 772-5588 | 312-4088 |
| Peter | NULL | NULL | 594-7477 |

and we want to find out the best way to contact each person according to the following rules:

1. If a person has a business phone, use the business phone number.

2. If a person does not have a business phone and has a cell phone, use the cell phone number.

3. If a person does not have a business phone, does not have a cell phone, and has a home phone, use the home phone number.

We can use the **COALESCE** function to achieve our goal:

**SELECT Name, COALESCE (Business\_Phone, Cell\_Phone, Home\_Phone) Contact\_Phone   
FROM Contact\_Info;**

Result:

|  |  |
| --- | --- |
| **Name** | **Contact\_Phone** |
| **Jeff** | **531-2531** |
| **Laura** | **772-5588** |
| **Peter** | **594-7477** |

## Difference between Stored Procedure and User Defined Function in SQL Server

**Stored Procedure**  
A Stored Procedure is nothing more than prepared SQL code that you save so you can reuse the code over and over again. So if you think about a query that you write over and over again, instead of having to write that query each time you would save it as a Stored Procedure and then just call the Stored Procedure to execute the SQL code that you saved as part of the Stored Procedure.  
  
In addition to running the same SQL code over and over again you also have the ability to pass parameters to the Stored Procedure, so depending on what the need is, the Stored Procedure can act accordingly based on the parameter values that were passed.  
  
Stored Procedures can also improve performance. Many tasks are implemented as a series of SQL statements. Conditional logic applied to the results of the first SQL statements determine which subsequent SQL statements are executed. If these SQL statements and conditional logic are written into a Stored Procedure, they become part of a single execution plan on the server. The results do not need to be returned to the client to have the conditional logic applied; all of the work is done on the server.   
  
**Benefits of Stored Procedures**

* **Precompiled execution**: SQL Server compiles each Stored Procedure once and then reutilizes the execution plan. This results in tremendous performance boosts when Stored Procedures are called repeatedly.
* **Reduced client/server traffic**: If network bandwidth is a concern in your environment then you'll be happy to learn that Stored Procedures can reduce long SQL queries to a single line that is transmitted over the wire.
* **Efficient reuse of code and programming abstraction**: Stored Procedures can be used by multiple users and client programs. If you utilize them in a planned manner then you'll find the development cycle requires less time.
* **Enhanced security controls**: You can grant users permission to execute a Stored Procedure independently of underlying table permissions.

**User Defined Functions**  
Like functions in programming languages, SQL Server User Defined Functions are routines that accept parameters, perform an action such as a complex calculation, and returns the result of that action as a value. The return value can either be a single scalar value or a result set.  
Functions in programming languages are subroutines used to encapsulate frequently performed logic. Any code that must perform the logic incorporated in a function can call the function rather than having to repeat all of the function logic.  
  
**SQL Server supports two types of functions**

* **Built-in functions**: Operate as defined in the Transact-SQL Reference and cannot be modified. The functions can be referenced only in Transact-SQL statements using the syntax defined in the Transact-SQL Reference.
* **User Defined Functions**: Allow you to define your own Transact-SQL functions using the CREATE FUNCTION statement. User Defined Functions use zero or more input parameters, and return a single value. Some User Defined Functions return a single, scalar data value, such as an int, char, or decimal value.

**Benefits of User Defined Functions**

* **They allow modular programming**: You can create the function once, store it in the database, and call it any number of times in your program. User Defined Functions can be modified independently of the program source code.
* **They allow faster execution:** Similar to Stored Procedures, Transact-SQL User Defined Functions reduce the compilation cost of Transact-SQL code by caching the plans and reusing them for repeated executions. This means the user-defined function does not need to be reparsed and reoptimized with each use resulting in much faster execution times. CLR functions offer significant performance advantage over Transact-SQL functions for computational tasks, string manipulation, and business logic. Transact-SQL functions are better suited for data-access intensive logic.
* **They can reduce network traffic**: An operation that filters data based on some complex constraint that cannot be expressed in a single scalar expression can be expressed as a function. The function can then invoked in the WHERE clause to reduce the number or rows sent to the client.

**Differences between Stored Procedure and User Defined Function in SQL Server**

**Differences between Stored Procedure and User Defined Function in SQL Server**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **User Defined Function** | **Stored Procedure** |
| 1 | Function must return a value. | Stored Procedure may or not return values. |
| 2 | Will allow only Select statements, it will not allow us to use DML statements. | Can have select statements as well as DML statements such as insert, update, delete and so on |
| 3 | It will allow only input parameters, doesn't support output parameters. | It can have both input and output parameters. |
| 4 | It will not allow us to use try-catch blocks. | For exception handling we can use try catch blocks. |
| 5 | Transactions are not allowed within functions. | Can use transactions within Stored Procedures. |
| 6 | We can use only table variables, it will not allow using temporary tables. | Can use both table variables as well as temporary table in it. |
| 7 | Stored Procedures can't be called from a function. | Stored Procedures can call functions. |
| 8 | Functions can be called from a select statement. | Procedures can't be called from Select/Where/Having and so on statements. Execute/Exec statement can be used to call/execute Stored Procedure. |
| 9 | A UDF can be used in join clause as a result set. | Procedures can't be used in Join clause |

## Transactions in SQL Server

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fails, the transaction fails. Therefore, a transaction has only two results: success or failure. Incomplete steps result in the failure of the transaction.

Users can group two or more Transact-SQL statements into a single transaction using the following statements:

* Begin Transaction
* Rollback Transaction
* Commit Transaction

If anything goes wrong with any of the grouped statements, all changes need to be aborted. The process of reversing changes is called **rollback** in SQL Server terminology. If everything is in order with all statements within a single transaction, all changes are recorded together in the database. In SQL Server terminology, we say that these changes are **committed** to the database.

Here is an example of a transaction :

USE pubs

DECLARE @intErrorCode INT

BEGIN TRAN

UPDATE Authors

SET Phone = '415 354-9866'

WHERE au\_id = '724-80-9391'

SELECT @intErrorCode = @@ERROR

IF (@intErrorCode <> 0) GOTO PROBLEM

UPDATE Publishers

SET city = 'Calcutta', country = 'India'

WHERE pub\_id = '9999'

SELECT @intErrorCode = @@ERROR

IF (@intErrorCode <> 0) GOTO PROBLEM

COMMIT TRAN

PROBLEM:

IF (@intErrorCode <> 0) BEGIN

PRINT 'Unexpected error occurred!'

ROLLBACK TRAN

END

Before the real processing starts, the BEGIN TRAN statement notifies SQL Server to treat all of the following actions as a single transaction. It is followed by two UPDATE statements. If no errors occur during the updates, all changes are committed to the database when SQL Server processes the COMMIT TRAN statement, and finally the stored procedure finishes. If an error occurs during the updates, it is detected by if statements and execution is continued from the PROBLEM label. After displaying a message to the user, SQL Server rolls back any changes that occurred during processing. Note: Be sure to match BEGIN TRAN with either COMMIT or ROLLBACK.