## **Problems Without Normalization**

If a table is not properly normalized and have data redundancy then it will not only eat up extra memory space but will also make it difficult to handle and update the database, without facing data loss. Insertion, Updation and Deletion Anomalies are very frequent if database is not normalized. To understand these anomalies let us take an example of a **Student** table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **rollno** | **name** | **branch** | **hod** | **office\_tel** |
| 401 | Akon | CSE | Mr. X | 53337 |
| 402 | Bkon | CSE | Mr. X | 53337 |
| 403 | Ckon | CSE | Mr. X | 53337 |
| 404 | Dkon | CSE | Mr. X | 53337 |

In the table above, we have data of 4 Computer Sci. students. As we can see, data for the fields branch, hod(Head of Department) and office\_tel is repeated for the students who are in the same branch in the college, this is **Data Redundancy**.

#### Insertion Anomaly

Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as **NULL**.

Also, if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students.

These scenarios are nothing but **Insertion anomalies**.

#### Updation Anomaly

What if Mr. X leaves the college? or is no longer the HOD of computer science department? In that case all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency. This is Updation anomaly.

#### Deletion Anomaly

In our **Student** table, two different informations are kept together, Student information and Branch information. Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information. This is Deletion anomaly.

## **Normalization Rule**

Normalization rules are divided into the following normal forms:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF
5. Fourth Normal Form

### First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single(atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

In the next tutorial, we will discuss about the [**First Normal Form**](https://www.studytonight.com/dbms/first-normal-form.php) in details.

### Second Normal Form (2NF)

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.

To understand what is Partial Dependency and how to normalize a table to 2nd normal for, jump to the [**Second Normal Form**](https://www.studytonight.com/dbms/second-normal-form.php) tutorial.

### Third Normal Form (3NF)

A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

Here is the [**Third Normal Form**](https://www.studytonight.com/dbms/third-normal-form.php) tutorial. But we suggest you to first study about the second normal form and then head over to the third normal form.

### Boyce and Codd Normal Form (BCNF)

**Boyce and Codd Normal Form** is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

* R must be in 3rd Normal Form
* and, for each functional dependency ( X → Y ), X should be a super Key.

To learn about BCNF in detail with a very easy to understand example, head to [**Boye-Codd Normal Form**](https://www.studytonight.com/dbms/boyce-codd-normal-form.php) tutorial.

### Fourth Normal Form (4NF)

A table is said to be in the Fourth Normal Form when,

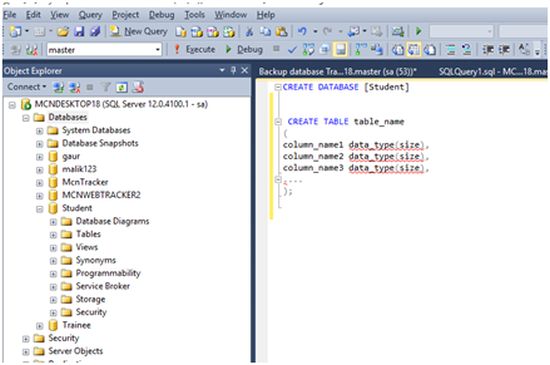
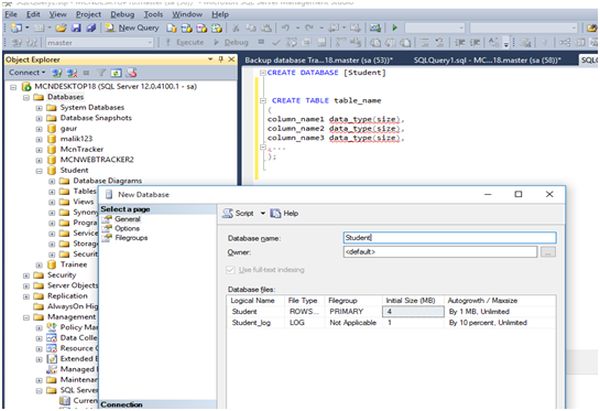
1. It is in the Boyce-Codd Normal Form.
2. And, it doesn't have Multi-Valued Dependency.

Here is the [**Fourth Normal Form**](https://www.studytonight.com/dbms/fourth-normal-form.php) tutorial. But we suggest you to understand other normal forms before you head over to the fourth normal form.

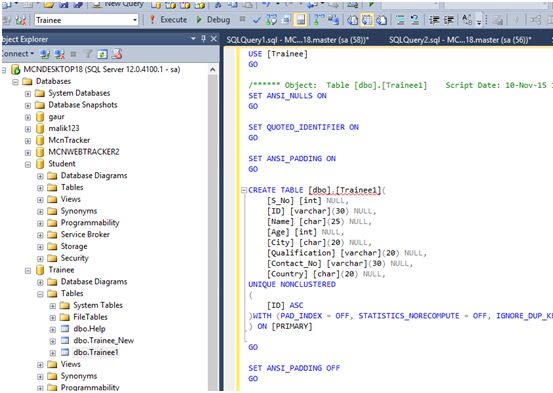
## **Question 1: How to create a database in SQL?**

**Answer 1:**A database is an organized file of data. It is a collection of schemas, tables, procedures, code functions, and other objects. Various query languages are used to access and manipulate data. In SQL Server, a table is the object that stores data in a tabular (columns and rows) form.

You can create a new database using CREATE DATABASE SQL command.

**Syntax:**CREATE DATABASE DatabaseName  
**Example:**CREATE DATABASE Student  
  
  
  
Or you can create a database using the SQL Server Management Studio. Right click on Databases and select New Database and follow the wizard steps.   
  


## **Question 2: What is SQL?**

**Answer 2:** Structured Query Language (SQL) is a programming language for accessing and manipulating Relational Database Management Systems (RDBMSs). SQL is widely used in popular RDBMSs such as SQL Server, Oracle, and MySQL. The smallest unit of execusion in SQL is a query. A SQL query is used to select, update, and delete data.  
  
In RDBMSs, all the data is stored in tables with each table consisting of rows and columns. The following is an example of a SQL query, CREATE DATABASE.  
  
  
Here is an example of a SQL query and the result in an Oracle database.  
  
  
  
Here we can see our database is created.  
  


Learn more here: [What is SQL](https://www.c-sharpcorner.com/article/what-is-sql/)

## **Question 3: What is PL/SQL?**

**Answer 3:**PL/SQL (Procedural Language for SQL) is a procedural language developed by Oracle to work with Oracle database using procedures in SQL. PL/SQL program units are compiled by the Oracle Database server and stored inside the database. And at run-time, both PL/SQL and SQL run within the same server process, bringing optimal efficiency. PL/SQL automatically inherits the robustness, security, and portability of the Oracle Database. PL/SQL syntaxes includes declarations for variables, constants, procedures, functions, conditions and loops.

Control statements in PL/SQL:

* Control statements are very important in PL/SQL.
* Control Statements are elements in a program that control the flow of program execution.
* The syntax of control statements are similar to regular English and are very similar to choices that we make every day.
* Branching statements are as follows:  
  + If statement
  + If - THEN - ELSE
  + Nested IF
  + Branching with logical connectivity
  + While
  + For Loop

Here is a more detailed article on PL/SQL: [PL/SQL Control Statements in Oracle](http://www.c-sharpcorner.com/UploadFile/a8024d/plsql-control-statements-in-oracle/).

## **Question 4: What is the difference between SQL and PL/SQL?**

**Answer:** SQL is standard query language for adding, accessing, and manipulating data in RDBMSs. SQL is a

* Only simple IF / Else statements.
* Through SQL you can interact with database through ADO.NET
* In SQL you can execute a line of code
* It can run only on windows  
  PL/SQL: It is referred as Procedure Language / Structure Query Language:
* In PL/SQL you can execute a block of code not a single line of code.
* Deep control statements
* It can run in UNIX also.
* PL/SQL language includes object oriented programming techniques such as encapsulation, function overloading, and information hiding (all but inheritance).

Click on the following link to read further: [SQL Vs PL / SQL](http://www.c-sharpcorner.com/Blogs/1422/sql-vs-pl-sql.aspx)

## **Question 5: What is RDBMS?**

**Answer:**RDBMS: It is referred as Relation Database Management Systems (RDBMS).  
  
RDBMS possesses a set of the below given characteristics:

* Write-intensive operations: The RDBMS is frequently written to and is often used in transaction-oriented applications.
* Data in flux or historical data: The RDBMS is designed to handle frequently changing data. Alternatively, RDBMS can also store vast amounts of historical data, which can later be analyzed or "mined".
* Application-specific schema: The RDBMS is configured on a per-application basis and a unique schema exists to support each application.
* Complex data models. The relational nature of the RDBMS makes it suitable for handling sophisticated, complex data models that require many tables, foreign key values, complex join operations, and so on.
* Data integrity: The RDBMS features many components designed to ensure data integrity. This includes rollback operations, referential integrity, and transaction-oriented operations.

Click on the following link to read further: [Directory Services Vs RDBMS](http://www.c-sharpcorner.com/UploadFile/anandnarayanswamy/DSvsRDBMS08232005071026AM/DSvsRDBMS.aspx)

## **Question 6: What is a database table?**

**Answer:**Database table: Table contains records in the form of rows and columns. A permanent table is created in the database you specify and remains in the database permanently, until you delete it.  
 **Syntax:**

1. **Create** **table** TableName (ID **INT**, **NAME** **VARCHAR**(30) )
2. **Drop** syntax: **drop** **table** TableName
3. **Select** Syntax: **Select** \* **from** TableName

Click on the following link to read further: [Temporary Tables in SQL](http://www.c-sharpcorner.com/UploadFile/d35d6a/temporary-table-in-sql/)

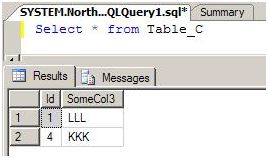
## **Question 7: How to create a table in SQL?**

**Answer:**SQL provides an organized way for table creation:  
  
**Syntax**

1. **Create** **table** TableName (columnName1 datatype, columnName2 datatype )

The following is an example of creating a simple table.

1. **create** **table** Info
2. (
3. **Name** **varchar**(20),
4. BirthDate **date**,
5. Phone nvarchar(12),
6. City **varchar**(20)
7. )

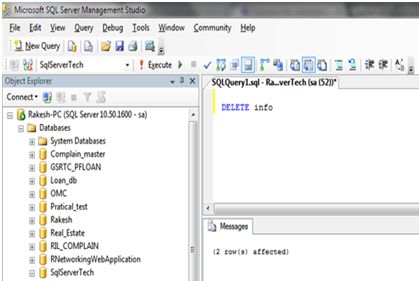


Click on the following link to read further: [SQL Server Database Basics Query Command](http://www.c-sharpcorner.com/UploadFile/b926a6/learning-sql-server-database-basics/)

## **Question 8: How to delete a table in SQL Server?**

**Answer:** Delete Data Record from Database Table and deleting an existing table by the following method:  
  
**Syntax:** To delete all table records of a table:

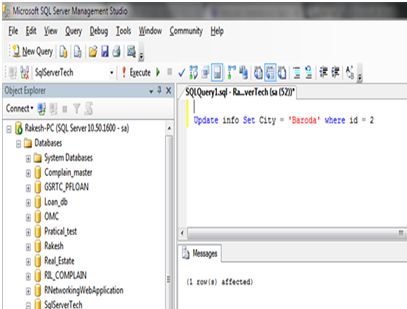
1. Delete TableName
2. DELETE info

  
  
Click on the following link to read further: [SQL Server Database Basics Query Command](http://www.c-sharpcorner.com/UploadFile/b926a6/learning-sql-server-database-basics/)

## **Question 9: How to update a database table using SQL?**

**Answer:** To update an existing Table we use SQL Command UPDATE: It will update the records as per user defined query/need.  
  
**Syntax:**

1. **Update** TableName **SET** ColumnName = NewData **where** Condition
2. **Update** info **Set** City = 'Baroda' **where** id = 2



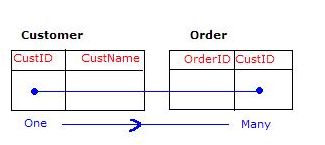
Click on the following link to read further: [SQL Server Database Basics Query Command](http://www.c-sharpcorner.com/UploadFile/b926a6/learning-sql-server-database-basics/)

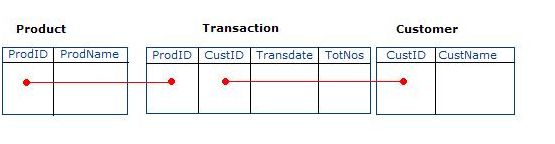
## **Question 10: What is a database relationship?**

**Answer:** Relationships are created by linking the column in one table with the column in another table. There are four different types of relationship that can be created.  
  
The relationships are listed below:

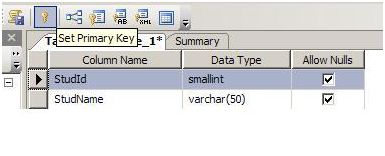
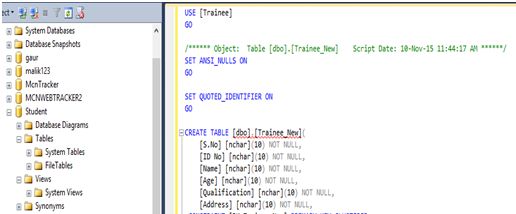
1. One to One Relationship
2. Many to One relationship
3. Many to Many relationship
4. One to One relationship

**One to Many & Many to One Relationship**

* For a One to many relationship, a single column value in one table has one or more dependent column values in another table. Look at the following diagram:  
  

**Many to Many Relationship**  
  
The third table acts as a bridge between the tables that want to establish a Many to Many relationship. The bridge table stores the common information between Many to Many relationship tables. Have a look at the following diagram:  
****  
Click on the following link to read further: [Database Table Relationship](http://www.c-sharpcorner.com/uploadfile/6897bc/database-table-relationship/)

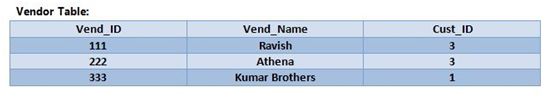
## **Question 11: What is a primary key of a database?**

**Answer:  
  
Primary key:**  
  
A table column with this constraint is called the key column for the table. This constraint helps the table to make sure that the value is not repeated and also that there are no null entries.  
  
  
Now this column does not allow null values and duplicate values. You can try inserting values to violate these conditions and see what happens. A table can have only one Primary key. Multiple columns can participate on the primary key.  
  
**Example:**  
  
  
  
Click on the following link to read further: [Table Constraints in SQL 2005](http://www.c-sharpcorner.com/uploadfile/6897bc/table-constraints-in-sql-2005/)

## **Question 12: What is a foreign key of a database?**

**Answer:** To define the relationship between two tables (one is called parent and the other one is the child table) connected by columns, a foreign key constraint is used. In this constraint the values of the child table must appear in the parent table, which means that for a foreign key, one table should point to a Primary Key in another table. A table can have multiple foreign keys and each foreign key can have a different referenced table.  
  
**Example:** To understand the foreign key clearly let's assume the following two tables:

1. CUSTOMER {Cust\_ID, Cust\_Name, Age, ContactNo, Gender, Address}
2. VENDOR {Vend\_ID, Vend\_Name, Cust\_ID}

  
  
  
  
**Example:**  
  
Foreign Key Constraint while using CREATE TABLE statement.  
  
**Syntax**

1. **CREATE** **TABLE** table\_name
2. (
3. Col1 datatype NOT NULL,
4. Col2 datatype NOT NULL,
5. Col3 datatype NOT NULL,
6. **CONSTRAINT** FK\_Column **FOREIGN** **KEY**(Col1, Col2, Col3) **REFERENCES** parent\_table(Col1, Col2, Col3)
7. );

AT SINGLE COLUMN LEVEL  
  
**Method 1:** With In-Line Specification/Column Level,

Click on the following link to read further: [Constraints in Oracle : Part 2](http://www.c-sharpcorner.com/UploadFile/nipuntomar/constraints-in-oracle-part-23/)

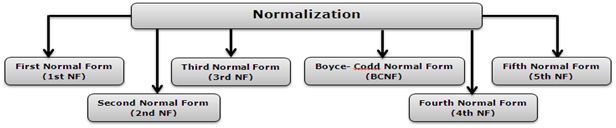
## **Question 13: What is database normalization?**

**Answer:** Database normalization is the process of organizing the fields and tables of a relational database to minimize redundancy and dependency. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships among them. Normalization is a bottom-up technique for database design.  
  
The evolution of Normalization theories is illustrated below:  


* First Normal Form (1NF)
* Second Normal Form (2NF)
* Third Normal Form (3NF)
* Boyce-Codd Normal Form (BCNF)
* 4th Normal Form
* 5th Normal Form
* 6th Normal Form

Click on the following link to read further: [Database Normalization](http://www.c-sharpcorner.com/UploadFile/0146e3/database-normalization/)

## **Question 14: What are database normalization forms?**

**Answer:** Normalization is the process of organizing data into a related table. it also eliminates redundancy and increases the integrity which improves performance of the query. To normalize a database, we divide the database into tables and establish relationships between the tables.  
  


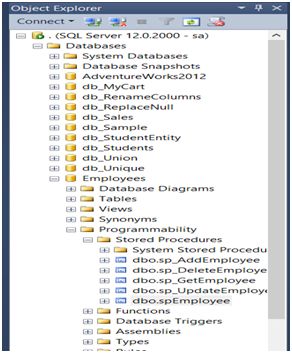
* First Normal Form (1st NF)
* Second Normal Form (2nd NF)
* Third Normal Form (3rd NF)
* Boyce-Codd Normal Form (BCNF)
* Fourth Normal Form (4th NF)
* Fifth Normal Form (5th NF)

Click on the following link to read further: [Normalization and its Types](http://www.c-sharpcorner.com/uploadfile/nipuntomar/normalization-and-its-types/)

## **Question 15: What is a stored procedure?**

**Answer:** A Stored Procedure is a collection or a group of T-SQL statements. Stored Procedures are a precompiled set of one or more statements that are stored together in the database. They reduce the network load because of the precompilation. We can create a Stored Procedure using the "Create proc" statement.  
  
**Why we use Stored Procedure**  
  
There are several reasons to use a Stored Procedure. They are a network load reducer and decrease execution time because they are precompiled. The most important use of a Stored Procedure is for security purposes. They can restrict SQL Injection. We can avoid SQL injection by use of a Stored Procedure.  
  
**How to create a Stored Procedure**

1. **CREATE** **PROCEDURE** spEmployee
2. **AS**
3. **BEGIN**
5. **SELECT** EmployeeId, **Name**, Gender, DepartmentName
6. **FROM** tblEmployees
7. **INNER** JOIN tblDepartments
8. **ON** tblEmployees.EmployeeDepartmentId = tblDepartments.DepartmentId
9. **END**

  
  
**Advantages of using a Stored Procedure in SQL Server**

* It is very easy to maintain a Stored Procedure and they are re-usable.
* The Stored Procedure retains the state of the execution plans.
* Stored Procedures can be encrypted and that also prevents SQL Injection Attacks

Click on the following link to read further: [Stored Procedures in SQL Server](http://www.c-sharpcorner.com/UploadFile/219d4d/stored-procedures-in-sql-server/)

## **Question 16: What is a function in SQL Server?**

**Answer:**A function is a sequence of statements that accepts input, processes them to perform a specific task and provides the output. Functions must have a name but the function name can never start with a special character such as @, $, #, and so on.  
  
**Types of function**

* Pre-Defined Function
* User-Defined Function

**User-defined Function:**  
  
In a user-defined function we write our logic according to our needs. The main advantage of a user-defined function is that we are not just limited to pre-defined functions. We can write our own functions for our specific needs or to simplify complex SQL code. The return type of a SQL function is either a scalar value or a table.  
  
**Creation of a function**

1. **Create** **function** ss(@id **int**)
2. **returns** **table**
3. **as**
4. **return** **select** \* **from** item **where** itemId = @id

**Execution of a Function**

1. **select** \* **from** ss(1)

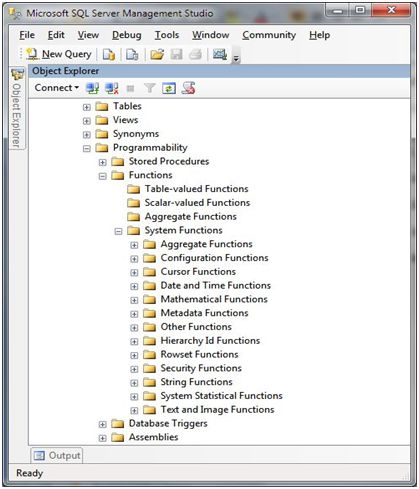
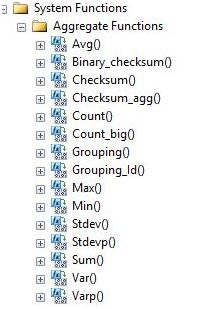
**Output**  
  


Click on the following link to read further: [Function in SQL Server 2012](http://www.c-sharpcorner.com/UploadFile/63f5c2/3/)

## **Question 17: What are the different types of functions in SQL Server?**

**Answer:** A function must return a result. So that is also called a function that returns a result or a value. When we create it a function must specify a value type that will return a value.

* Functions only work with select statements.
* Functions can be used anywhere in SQL, such as AVG, COUNT, SUM, MIN, DATE and so on with select statements.
* Functions compile every time.
* Functions must return a value or result.
* Functions only work with input parameters.
* Try and catch statements are not used in functions.

**Function Types**  
  
The following is the function list in SQL Server databases.  
  
  
  
SQL Server contains the following aggregates functions:  
  
  
  
Click on the following link to read further: [Functions in SQL Server Databases](http://www.c-sharpcorner.com/UploadFile/b926a6/function-operation-in-sql-database/)

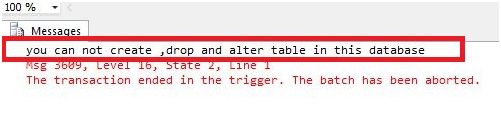
## **Question 18: What is a trigger in SQL Server?**

**Answer:**"A Trigger is a Database object just like a stored procedure or we can say it is a special kind of Stored Procedure which fires when an event occurs in a database.".  
  
It is a database object that is bound to a table and is executed automatically. We cannot explicitly call any trigger. Triggers provide data integrity and used to access and check data before and after modification using DDL or DML query.  
  
**Type of Triggers**  
  
There are two types of Triggers:

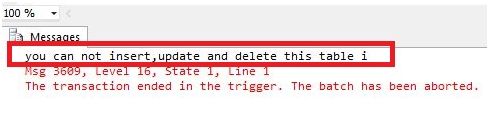
1. DDL Trigger
2. DML trigger

**DDL Triggers**  
  
They fire in response to DDL (Data Definition Language) command events that start with Create, Alter and Drop like Create\_table, Create\_view, drop\_table, Drop\_view and Alter\_table.  
  
**Code of DDL Triggers**

1. **create** **trigger** saftey
2. **on** **database**
3. **for**
4. create\_table, alter\_table, drop\_table
5. **as**
6. print 'you can not create ,drop and alter table in this database'
7. **rollback**;

**Output**  
  
  
  
**DML Triggers:**  
  
They fire in response to DML (Data Manipulation Language) command events that start with Insert, Update and Delete like insert\_table, Update\_view and Delete\_table.  
 **Code of DML Trigger:**

1. **create** **trigger** deep
2. **on** emp
3. **for**
4. **insert**, **update**, **delete**
5. **as**
6. print 'you can notinsert,update and delete this table i'
7. **rollback**;

**Output:**  
  
When we insert, update or delete in a table in a database then the following message appears:  
  


Click on the following link to read further: [Triggers in Sql Server](http://www.c-sharpcorner.com/UploadFile/63f5c2/triggers-in-sql-server/)

## **Question 19: Why and when to use triggers?**

**Answer:**We use a trigger when we want some event to happen automatically on certain desirable scenarios.  
  
You have a table that changes frequently, now you want to know how many times and when these changes take place. In that case you can create a trigger that will insert the desired data into another table whenever any change in the main table occurs.  
  
In SQL Server we can create the following 3 types of triggers:

* Data Definition Language (DDL) triggers
* Data Manipulation Language (DML) triggers
* Logon triggers

**Example**

1. **CREATE** **TRIGGER** trgAfterInsert **ON**[dbo].[Employee\_Test]
2. **FOR** **INSERT**
3. **AS**
4. **declare**@ empid **int**;
5. **declare**@ empname **varchar**(100);
6. **declare**@ empsal **decimal**(10, 2);
7. **declare**@ audit\_action **varchar**(100);
8. **select**@ empid = i.Emp\_ID **from** inserted i;
9. **select**@ empname = i.Emp\_Name **from** inserted i;
10. **select**@ empsal = i.Emp\_Sal **from** inserted i;
11. **set**@ audit\_action = 'Inserted Record -- After Insert Trigger.';
13. **insert** **into** Employee\_Test\_Audit
14. (Emp\_ID, Emp\_Name, Emp\_Sal, Audit\_Action, Audit\_Timestamp)
15. **values**(@empid, @empname, @empsal, @audit\_action, getdate());
17. PRINT 'AFTER INSERT trigger fired.'
18. GO

Click on the following link to read further: [Triggers in SQL Server](http://www.c-sharpcorner.com/UploadFile/a53f1a/triggers-in-sql-server/)

## **Question 20: What are the different types of triggers?**

**Answer:** Triggers are a special type of stored procedure which are executed automatically based on the occurrence of a database event. These events can be categorized as:

* Data Manipulation Language (DML) and
* Data Definition Language (DDL) events.

The benefits derived from triggers is based in their events driven nature. Once created, the trigger automatically fires without user intervention based on an event in the database.

1. **Using DML Triggers:**  
     
   DML triggers are invoked when any DML command such as INSERT, DELETE, and UPDATE occurs on the data of a table and/or view.  
     
   * DML triggers are powerful objects for maintaining database integrity and consistency.
   * DML triggers evaluate data before it has been committed to the database.
   * During this evaluation the following actions are performed.

We cannot use the following commands in DML trigger:

* + ALTER DATABASE
  + CREATE DATABASE
  + DISK DATABASE
  + LOAD DATABASE
  + RESTORE DATABASE

1. **Using DDL Triggers:**  
     
   * These triggers focus on changes to the definition of database objects as opposed to changes to the actual data.
   * This type of trigger is useful for controlling development and production database environments.

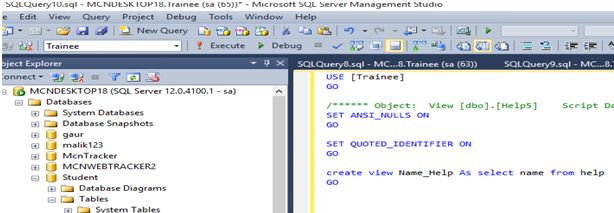
Let us create DDL trigger now;  
  
The following is the syntax.

1. **CREATE** **TRIGGER** trigger\_name
2. **ON**
3. {
4. ALL SERVER | **DATABASE**
5. }
6. [**WITH** < ddl\_trigger\_option > [, ...n]]
7. {
8. **FOR** | **AFTER**
9. }
10. {
11. event\_type | event\_group
12. }[, ...n]
13. **AS**
14. {
15. sql\_statement[;][...n] | EXTERNAL **NAME** < method specifier > [;]
16. }
17. **CREATE** **TRIGGER** tr\_TableAudit
18. **ON** **DATABASE**
19. **FOR** CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE
20. **AS**
21. PRINT 'You must disable the TableAudit **trigger** in **order**
22. **to** change any **table** in this **database** '
23. **ROLLBACK**
24. GO

Click on the following link to read further: [Creating and managing Triggers in SQL Server 2005/2008](http://www.c-sharpcorner.com/UploadFile/37db1d/creating-and-managing-triggers-in-sql-server-20052008/)

## **Question 21: What is a view in the database?**

**Answer:** A View is nothing but a select query with a name given to it or we can simply say a view is a Named Query. Ok! Why do we need a view? There can be many answers for this. Some of the important stuff I feel is:

1. A view can combine data from multiple tables using adequate joins and while bringing it may require complex filters and calculated data to form the required result set. From a user's point of view, all these complexities are hidden data queried from a single table.
2. Sometimes for security purposes, access to the table, table structures and table relationships are not given to the database user. All they have is access to a view not knowing what tables actually exist in the database.
3. Using the view, you can restrict the user update only portions of the records.  
     
   

The following are the key points to be noted about views:

1. Multiple views can be created on one table.
2. Views can be defined as read-only or updatable.
3. Views can be indexed for better performance.
4. Insert, update, delete can be done on an updatable view.

Click on the following link to read further: [Creating and Using SQL Server View](http://www.c-sharpcorner.com/UploadFile/6897bc/creating-and-using-sql-server-view/)

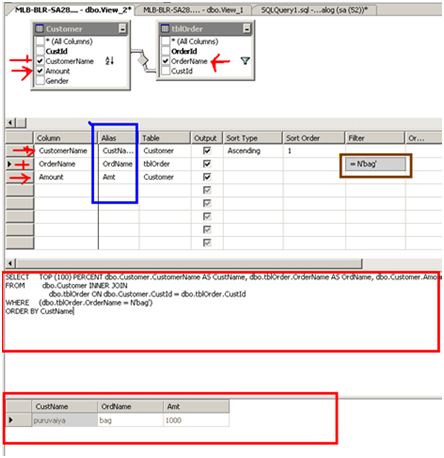
## **Question 22: What do I need views in a database?**

**Answer:** There are a number of scenarios where we have to look for a view as a solution.

* To hide the complexity of the underlying database schema, or customize the data and schema for a set of users.
* To control access to rows and columns of data.
* To aggregate data for performance.

Views are used for security purposes because they provide encapsulation of the name of the table. Data is in the virtual table, not stored permanently. Views display only selected data.  
  
**Syntax of a View:**

1. **CREATE** **VIEW** view\_name **AS**
2. **SELECT** column\_name(s)
3. **FROM** table\_name
4. **WHERE** condition

  
  
There are two types of views.

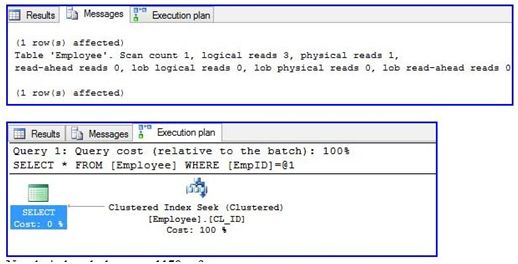
* Simple View
* Complex View

Click on the following link to read further: [Concept of View in SQL Server](http://www.c-sharpcorner.com/UploadFile/6cde20/concept-of-view-in-sql-server/)

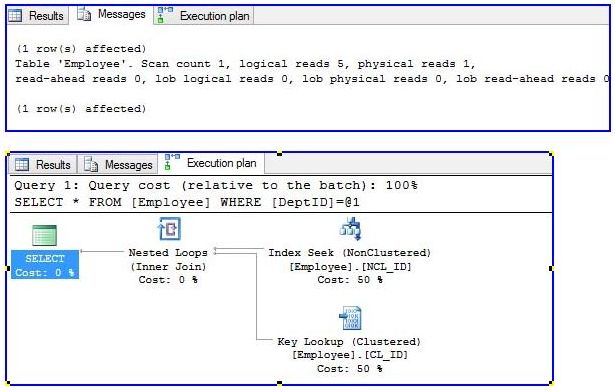
## **Question 23: What is an index?**

**Answer:**An Index is one of the most powerful techniques to work with this enormous information. Database tables are not enough for getting the data efficiently in case of a huge amount of data. In order to get the data quickly we need to index the column in a table.  
  
An index is a database object that is created and maintained by the DBMS. Indexed columns are ordered or sorted so that data searching is extremely fast. An index can be applied on a column or a view. A table can have more than one index.  
  
**Types of Index:**  
  
Microsoft SQL Server has two types of indexes. These are:  
  
**Clustered Index:**  
  
A Clustered Index sorts and stores the data in the table based on keys. A Clustered Index can be defined only once per table in the SQL Server Database, because the data rows can be sorted in only one order. Text, nText and Image data are not allowed as a Clustered index.

1. **SET** **STATISTICS** IO **ON**
2. **SELECT** \* **FROM** Employee **WHERE** EmpID = 20001

**EmpID         EmpName                 Cell                    Dept  
20001         Black Smith       12345678901             1**  
  
  
  
**Non-Clustered Index:**  
  
Non Clustered Indexes or simply indexes are created outside of the table. SQL Server supports 999 Non-Clustered per table and each Non-Clustered can have up to 1023 columns. A Non-Clustered Index does not support the Text, nText and Image data types.

1. **CREATE** NONCLUSTERED **INDEX** NCL\_ID **ON** Employee(DeptID)
2. **SET** **STATISTICS** IO **ON**
3. **SELECT** \* **FROM** Employee **WHERE** DeptID = 20001

**EmpID             EmpName                    Cell                       Dept  
40001              Black Smith           12345678901          20001**  
  


Click on the following link to read further: [SQL Server Index](http://www.c-sharpcorner.com/UploadFile/97ec13/sql-server-index/)

## **Question 24: Why do I need an index in a database?**

**Answer:** An Index is a database object that can be created on one or more columns (16 max column combinations). When creating the index it will read the column(s) and forms a relevant data structure to minimize the number of data comparisons. The index will improve the performance of data retrieval and add some overhead to the data modification such as create, delete and modify. So it depends on how much data retrieval can be done on table versus how much of DML (Insert, Delete and Update) operations.  
  
Need of Index in Database: An index is basically used for fast data retrieval from the database.  
  
**Syntax:**

1. **CREATE** **INDEX** index\_name **ON** table\_name;
2. or
3. **DROP** **INDEX** index\_name;

**Type of Index:**

In a SQL Server database there are mainly the following two types of indexes:

1. Clustered index and
2. Non Clustered index

Click on the following link to read further:  [Index in SQL SERVER](http://www.c-sharpcorner.com/UploadFile/a53f1a/index-in-sql-server/)

**Question 25: What is a query in a database?  
  
Answer:** SQL is a complete data manipulation language that is used not only for database queries, but also to modify and update data in the database. Compared to the complexity of the SELECT statement, which supports SQL queries, the SQL statements that modify and create database contents are somewhat simple.  
  
However, database updates pose some challenges for a DBMS beyond those presented by database queries. The DBMS must protect the integrity of the stored data during changes, ensuring that only valid data is introduced into the database. The DBMS must also coordinate simultaneous updates by multiple users, ensuring that the users and their changes do not interfere with one another.  
  
INSERT statement adds new rows of data to a table.  
  
**Syntax:**

1. **Insert** **into** table\_Name(**column** names) **values**(**Values** **for** **column**).
2. **INSERT** **INTO** employee(ID, SURNAME, FIRSTNAME, EMAIL, COUNTRY, PHONE)
3. **VALUES**(111, 'vithal', 'wadje', 'vithal.wadje@yahoo.com', 'India', '+914545455454')

Click on the following link to read further: [Introduction to Insert Statement in SQL Server](http://www.c-sharpcorner.com/UploadFile/0c1bb2/introduction-to-insert-statement-in-sql-server/)

## **Question 26: What are query types in a database?**

**Answer:** Types of Commands in SQL ServerThese commands are categorized into:

* DDL
* DCL
* DML
* TCL

Let's see these categories one-by-one.  
  
**DDL**  
  
Data Definition Language (DDL) commands are the category responsible for dealing with the structure of objects. I mean that with these commands we can modify our object/entity structure. For example if there's one table and you want to modify the structure of that table, you can use DDL commands.  
  
The following are the commands in this category:

|  |  |
| --- | --- |
| **Command** | **Description** |
| Create | Used to create objects. |
| Alter | Used to modify created object. |
| Drop | Used to delete object. |

Using these commands you can create any objects like tables, views, databases, triggers, and so on.  
  
**For example:**

1. **CREATE** **DATABASE** DB2
2. GO
3. **CREATE** **TABLE** tblDemo
4. (
5. Id **int** **primary** **key**,
6. **Name** **char**(20)
7. )
8. GO
9. **DROP** **DATABASE** DB2

**DML**  
  
Data Manipulation Language (DML) commands manipulate data stored in objects like tables, views and so on. With the help these commands you can easily modify, insert and delete your data.  
  
The following are the commands in this category:

|  |  |
| --- | --- |
| **Command** | **Description** |
| Insert | Insert data into table. |
| Delete | Delete data from table. |
| Update | Update data into table. |
| Insert Into | Insert bulk data into table. |

Using these commands you can manipulate any kind of data stored in entities.  
  
**For example:**

1. **INSERT** **INTO** tblDemo **VALUES**(1, 'Abhishek')
2. GO
3. **DELETE** **FROM** tblDemo **WHERE** Id = 4
4. GO
5. **UPDATE** tblDemo
6. **SET** **Name** = 'Sunny'
7. **WHERE** Id = 6
8. GO

**DCL**  
  
Data Control Language (DCL) commands are for security purposes. These commands are used to provide roles, permissions, access and so on.  
  
The following are the commands in this category:

|  |  |
| --- | --- |
| **Command** | **Description** |
| Grant | Provide user access to Database or any other object. |
| Revoke | Take back the access from user. |

For example: we have the following data.  
  
**Database:**CSharpCornerDB  
**Table:**  
**User:** CSharpCorner  
  
Currently we didn't provide any permission to this user.  
  
  
Now we'll create a table in the CSharpCornerDB database.

1. **CREATE** **table** tblArticles
2. (
3. ArticleId **int** **primary** **key** identity,
4. ArticleName **varchar**(10),
5. Category **varchar**(10)
6. )

If we execute this command, we'll get an error message.  
  
Msg 262, Level 14, State 1, Line 1CREATE TABLE permission denied in database 'CSharpCornerDB'.  
  
This is because this user doesn't have permission to create anything right now. We'll learn how to grant or revoke permission on objects in our next article.  
  
**TCL**  
  
Transaction Control Language (TCL) commands are for managing transactions in SQL Server. The following are the commands in this category:

|  |  |
| --- | --- |
| **Command** | **Description** |
| Commit | Used to save any transaction permanently. |
| Rollback | This command Is used to restore database to its last committed state. |
| Save Tran | This command is used to save the transaction so that we can rollback that transaction to the point whenever necessary. |

For example, we have a table named "tblStudent" with 3 records as shown below:  
  
  
Now we'll begin our transaction and add another record and commit that transaction.

1. **Begin** Tran
2. **Insert** **INTO** tblStudents **VALUES**('Sumit')
3. **COMMIT**

Now we have 4 Records.  
  
  
Now, we'll add three records, one by one with save point, but we don't commit our transaction.

1. **Begin** Tran
2. **Insert** **INTO** tblStudents **VALUES**('Kajal')
3. SAVE Tran A;
4. **Insert** **INTO** tblStudents **VALUES**('Rahul')
5. SAVE Tran B;
6. **Insert** **INTO** tblStudents **VALUES**('Ram')
7. SAVE Tran C;
9. **SELECT** \* **from** tblStudents

Now we have the following records in the table, from which the last three records are not yet committed.  
  
  
  
Now we have 3 savepoints, in other words A, B and C. Since our transaction is not yet committed, we can roll it back to any savepoint. We'll roll it back to point B, in other words at "Rahul".

1. ROLLBACK TRAN B
2. COMMIT Now when you fire the select query, you'll get records up to ID 6.

So this was the article based on types of commands in SQL Server with which you can play with data stored in SQL Server.  
  
In this article, we have seen the types of commands in SQL Server and done some overview of that. We have also seen how to commit transactions and how to roll back any transaction to any saved point.  
  
In my next article, we'll be explaining how to deal with the GRANT and REVOKE commands. Until then keep learning and keep sharing.

For further information click on the link : [Types of Commands in SQL Server](http://www.c-sharpcorner.com/UploadFile/ff0d0f/types-of-command-in-sql-server/)

## **Question 27: What is a join in SQL Server?**

**Answer:** If You want to retrieve data from multiple tables then you need to use joins in SQL Server. Joins are used to get data from two or more tables based on the relationships among some of the columns in the tables.  
  
**Syntax**  
  
The Inner join syntax is as follows:

1. **SELECT** < **column** list >
2. **FROM** < left joined **table** > [**INNER**] JOIN < right joined **table** >
3. **ON** < join condition >

The example is developed in SQL Server 2012 using the SQL Server Management Studio.  
  
**Creating Table in SQL Server**  
  
Now to create 3 tables in the Master database named Table1, Table2 and Table3.  
  
**Table1**

1. **CREATE** **TABLE** Table1
2. (
3. ID **INT**, **Name** **VARCHAR**(20)
4. )

**Table2**

1. **CREATE** **TABLE** Table2
2. (
3. ID **INT**, **Name** **VARCHAR**(30)
4. )

**Table3**

1. **CREATE** **TABLE** Table3
2. (
3. ID **INT**, **Name** **VARCHAR**(40)
4. )

Click on the following link to read further: [Joining Three or More Tables in SQL Server 2012](http://www.c-sharpcorner.com/UploadFile/rohatash/joining-three-or-more-tables-in-sql-server-2012/)

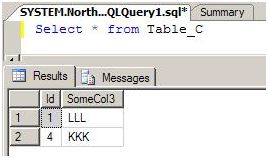
## **Question 28: What are different types of joins in SQL Server?**

**Answer:** Joins are useful for bringing data together from different tables based on their database relations. First we will see how the join operates between tables. Then we will explore the Order of Execution when both a join and a where condition exist. Finally we will move our exploration to the importance of the Join order.  
  
A Join condition defines a way two tables are related in a query by :

* Specifying the column to be used for the Join from each table. In joining foreign keys in a table and its associated key in the other table.
* To use the logical operator in comparing values from the columns.

There are three type of joins available based on the way we join columns of two different tables.

1. Full Join
2. Inner Join
3. Left outer Join
4. Right outer Join

**Full Join**  
  
A full join is somewhat different from the Cartesian product. A Cartesian product will get all the possible row combinations of the two joining tables. A Full Join takes the matching columns plus all table rows from the left table that does not match the right and all table rows in the right that does not match the left. It applies null for unmatched rows on the other end when doing so. The following example shows the full join between Table\_A and Table\_C  
 ****  
  
Click on the following link to read further: [Table Join in SQL 2005](http://www.c-sharpcorner.com/uploadfile/6897bc/table-join-in-sql-2005/)

## **Question 29: What is an inner join in SQL?**

**Answer:**  
  
**Inner or Self Join:**This Join returns a row when there is at least one match in both tables.  
  
**let's see an example:**

1. **Select** \* **From** Table1
2. **Inner** Join Table2
3. **on** table1.ID = table2.ID

The following query displays the Employee Name and the corresponding Manager Name within the employee table.

1. **SELECT** e1.Employee\_Name EmployeeName, e2.Employee\_Name ManagerName
2. **FROM** employee e1(nolock), employee e2(nolock)
3. **WHERE** e1.EmployeeID = e2.ManagerID

**Output**  
  
  
  
An inner join (sometimes called a "simple join") is a join of two or more tables that returns only those rows that satisfy the join condition.

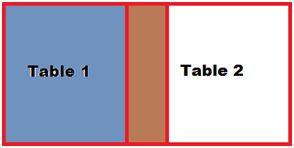
Click on the following link to read further: [Explain SQL Joins](http://www.c-sharpcorner.com/UploadFile/47fc0a/explain-sql-joins/)

## **Question 30: What is an outer join in SQL?**

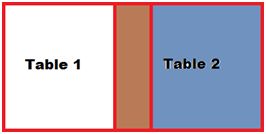
**Answer:**There are three different types of outer joins; let's see 1 by 1.

* Left Outer Join
* Right Outer Join
* Full Outer Join

**Left Outer Join**

A LEFT OUTER JOIN is one of the JOIN operations that allows you to specify a join clause. It preserves the unmatched rows from the first (left) table, joining them with a NULL row in the shape of the second (right) table.  
  


1. **Select** \* **From** Table1
2. Left Outer Join
3. **on** table1.ID = table2.ID

**Right Outer Join**  
  
A RIGHT OUTER JOIN is one of the JOIN operations that allows you to specify a JOIN clause. It preserves the unmatched rows from the Table2 (right) table, joining them with a NULL in the shape of the Table1 (left) table. A LEFT OUTER JOIN B is equivalent to B RIGHT OUTER JOIN A, with the columns in a different order.  
  


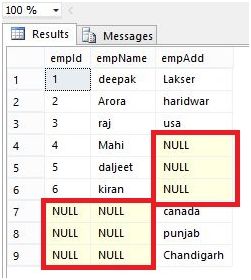
1. **Select** \* **From** Table1
2. Right Outer Join
3. **on** table1.ID = table2.ID

Click on the following link to read further: [Introduction to Joins in Sql Server](http://www.c-sharpcorner.com/uploadfile/c8b468/introduction-to-joins-in-sql-server/)

## **Question 31: What is full join in SQL?**

**Answer:** A Full Outer Join fetches all records of both tables; where the record does not match, it returns Null.

1. **select** e.empId, e.empName, e1.empAdd **from** emp e **full** outer join emp\_add e1 **on** e.empId = e1.empId

**Output:**  
  
  
  
Or  
  
**Full Outer Join**  
  
FULL OUTER JOIN: This JOIN is a combination of both. All records from both Left\_Table and Right\_Table are in the result set and matched when they can be on the Join\_Condition; when no record is found in the opposite table, NULL values are used for the columns.

1. **Select** \* **From** Table1
2. **Full** Outer Join
3. **on** table1.ID = table2.ID

Click on the following link to read further: [Joins in SQL Server 2012](http://www.c-sharpcorner.com/UploadFile/63f5c2/joins-in-sql-server/)

## **Question 32: What is left join in SQL Server?**

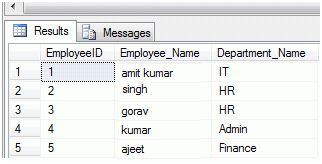
**Answer:**Left Join: A LEFT OUTER JOIN is one of the JOIN operations that allows you to specify a join clause.

It preserves the unmatched rows from the first (left) table, joining them with a NULL row in the shape of the second (right) table.

1. **Select** \* **From** Table1
2. Left Outer Join
3. **on** table1.ID=table2.ID

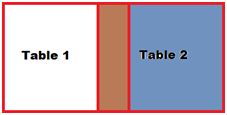
A left outer join displays all the rows from the first table and the matched rows from the second table.  
  
**Example**  
  
The following query retrieves the employee name and the corresponding department he belongs to, whereas all the departments are displayed even if the employee is not assigned to any department.

1. **SELECT** e.EmployeeID, e.Employee\_Name, d.Department\_Name
2. **FROM** employee e(nolock) LEFT JOIN department d(nolock)
3. **ON** e.DepartmentID = d.DepartmentID

**Output**  
  


Click on the following link to read further:  [Explain SQL Joins](http://www.c-sharpcorner.com/UploadFile/47fc0a/explain-sql-joins/)

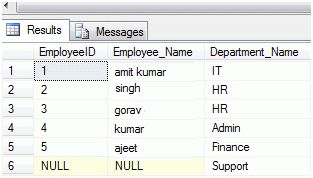
## **Question 33: What is a right join in SQL Server?**

**Answer:**  
  
**Right Join:**A RIGHT OUTER JOIN is one of the JOIN operations that allows you to specify a JOIN clause. It preserves the unmatched rows from the Table2 (right) table, joining them with a NULL in the shape of the Table1 (left) table. A LEFT OUTER JOIN B is equivalent to B RIGHT OUTER JOIN A, with the columns in a different order.  
  


1. **Select** \* **From** Table1
2. Right Outer Join
3. **on** table1.ID = table2.ID

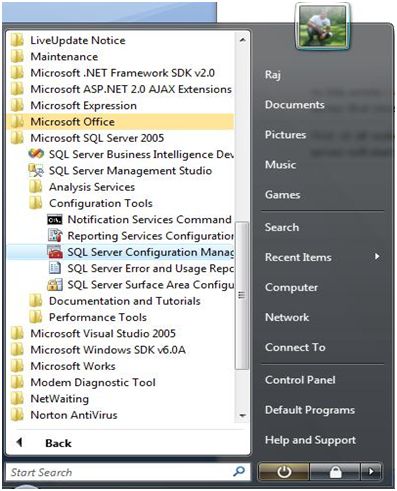
The right outer join displays all the rows from the second table and matched rows from the first table.  
  
**Example**

1. **SELECT** e.EmployeeID, e.Employee\_Name, d.Department\_Name
2. **FROM** employee e(nolock) RIGHT JOIN department d(nolock)
3. **ON** e.DepartmentID = d.DepartmentID

**Output**  
  


Click on the following link to read further:  [Explain SQL Joins](http://www.c-sharpcorner.com/UploadFile/47fc0a/explain-sql-joins/)

## **Question 34: What is database engine in SQL Server?**

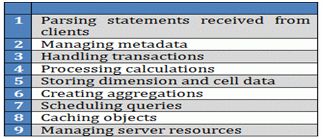
**Answer:**The SQL Server Database Engine, SQL Server Agent, and several other SQL Server components run as services. These services typically are started when the operating system starts. This depends on what is specified during setup; some services are not started by default.  
  
A service is a type of application (executable) that runs in the system background. Services usually provide core operating system features, such as Web serving, event logging, or file serving. Services can run without showing a user interface on the computer desktop. The SQL Server Database Engine, SQL Server Agent, and several other SQL Server components run as services. These services typically are started when the operating system starts. This depends on what is specified during setup; some services are not started by default.  
  
This article describes the management of the various SQL Server services on your machine. Before you log in to an instance of SQL Server, you need to know how to start, stop, pause, resume, and restart an instance of SQL Server. After you are logged in, you can perform tasks such as administering the server or querying a database.  
  
Let's start now, select start/All Programs/Microsoft SQL Server2005/Configuration Tools/SQL Server Configuration Manager. It should look like Figure 1.  
  


Click on the following link to read further: [How to Start and Stop Database Engine in SQL 2005](http://www.c-sharpcorner.com/UploadFile/raj1979/how-to-start-and-stop-database-engine-in-sql-2005/)

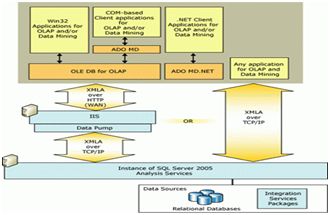
## **Question 35: What are the Analysis Services in SQL Server?**

**Answer:** The purpose of analysis services is to turn data into information and to provide quick and easy access to that information for decision makers. SSAS provides OLAP by letting you design, create and manage multidimensional structures that contain data aggregated from other data sources, such as relational databases and provides many data mining algorithms for mining data from data sources. So for delivering OLAP and data mining it uses client and server technologies.  
  
The main idea of SSAS is to provide fast results from data sources when we apply a query because in order to make a decision we need data of various dimensions.

Components of the Architecture in detail. 

**Server Architecture:**This runs as a Windows service. The Msmdsrv.exe application is a server component. This application consists of security, XMLA listener, query processor and other components that perform the following tasks:  
  
****

**Client Architecture:** SSAS has a thin client Component Architecture. All queries and calculations are resolved by the server only. So for each request a server to client connection is required. There are several providers with SSAS to support various programming languages. These providers communicate using SOAP packets. You can better understand this by the following diagram:

****

Click on the following link to read further: [SQL Server Analysis Services (SSAS) : Part 1](http://www.c-sharpcorner.com/UploadFile/abhijmk/sql-server-analysis-service-ssas-part-1/)

## **Question 36: What are the integration services in SQL Server?**

**Answer:** Integration Services is a platform for building high performance data integration and workflow solutions, including extraction, transformation and loading (ETL) operations for data warehousing.  
  
This includes graphical tools and wizards for building and debugging packages.  
  
**Uses of Integration Services:**  
  
One use of Integration Services is to merge data from multiple data stores and update the data to data warehouses and/or data marts. Create the Data Transformation process logic and automate the data loading process.  
  
**Architecture of Integration Services**  
  
Architecture of Integration Services  
  
Some important components to using Integration Services:

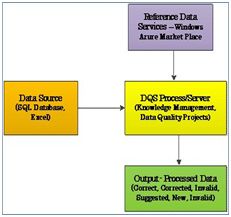
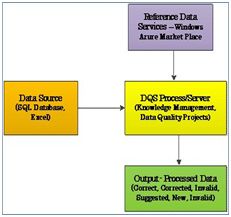
* SSIS Designer
* Runtime engine
* Tasks and other executables
* Data Flow engine and Data Flow components
* API or object model
* Integration Services Service
* SQL Server Import and Export Wizard
* Other tools, wizards and command prompt utilities

Click on the following link to read further: [An Introduction to SQL Server Integration Service (SSIS)](http://www.c-sharpcorner.com/UploadFile/a9d961/an-introduction-to-sql-server-integration-service-ssis/)

## **Question 37: What are the data quality services in SQL Server?**

**Answer:**  
  
**SQL Server Data Quality Services:**  
  
SQL Server 2012 Data Quality Services (DQS) is the data quality product from Microsoft SQL Server 2012. DQS enables you to perform a variety of critical data quality tasks, including correction, enrichment, standardization and de-duplication of your data.  
  
DQS provides the following features to resolve data quality issues.

* Data Cleansing
* Matching
* Reference Data Services
* Profiling and Monitoring
* Knowledge Base

  
  
DQS enables you to perform data cleansing using cloud-based reference data services provided by reference data providers. DQS also provides profiling that is integrated into its data-quality tasks, enabling to analyze the integrity of the data.  
 **Data Quality Server**  
  
Data Quality Server is implemented as three SQL Server catalogs DQS\_MAIN, DQS\_PROJECTS, and DQS\_STAGING\_DATA.  
  
DQS\_MAIN includes DQS Stored Procedures, DQS engine, and published Knowledge Bases.  
  
DQS\_PROJECTS includes data that is required for Knowledge Base management and DQS project activities.  
  
DQS\_STAGING\_DATA provides an intermediate staging database where you can copy your source data to perform DQS operations, and then export your processed data.  
  
****  
Click on the following link to read further: [SQL Server 2012 Data Quality Services (DQS)](http://www.c-sharpcorner.com/UploadFile/db7c6f/sql-server-2012-data-quality-services-dqs/)

## **Question 38: What are the reporting services in SQL Server?**

**Answer:** SQL Server Reporting Services is a comprehensive reporting platform that includes processing components. Processing components are the basis for the multilayered architecture of SQL Server Reporting Services. Processing components interact with each other to retrieve data and deliver a report.  
  
SQL Server Reporting Services has the following two basic components.

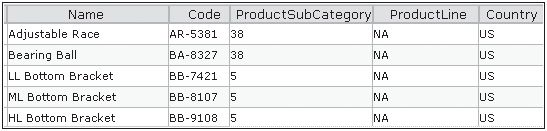
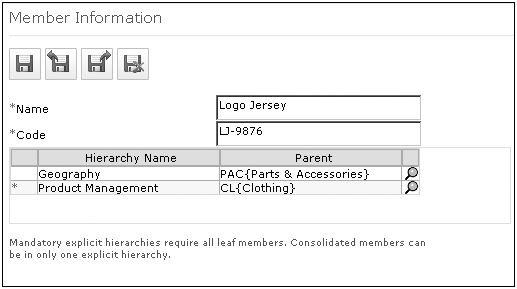
* Processors
* Extensions

**Architecture of SQL Server Reporting Services**  
The following diagram shows the architecture of SQL Server Reporting Services.  
  
Architecture of SQL Server Reporting Services  
  
**Tools and Components of SQL Server Reporting Services architecture**  
  
This architecture consists mainly of the following types of components and tools.

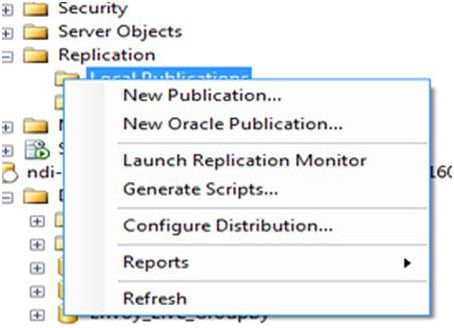
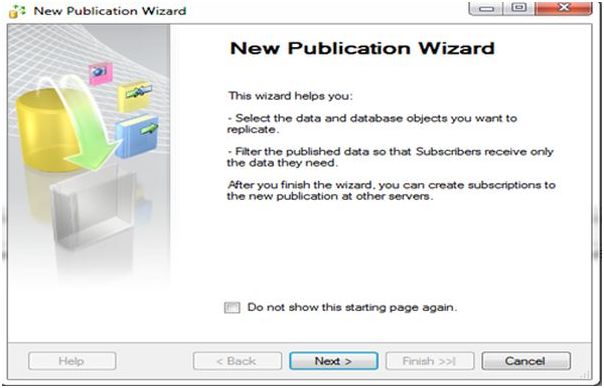
* Report Builder
* Report Designer
* Report Manager
* Report Server
* Report server database
* Data sources

Click on the following link to read further: [Architecture of SQL Server Reporting Services (SSRS)](http://www.c-sharpcorner.com/UploadFile/35fef0/architecture-of-sql-server-reporting-servicesssrs/)

## **Question 39: What are the master data services in SQL Server?**

**Answer:** The goal of MDS is to address the challenges of both operational and analytical master data management by providing a master data hub to centrally organize, maintain, and manage your master data. This master data hub supports these capabilities with a scalable and extensible infrastructure built on SQL Server and the Windows Communication Foundation (WCF) APIs.  
  
**Master Data Services Components**  
  
The wizard installs Master Data Services Configuration Manager, installs the files necessary to run the Master Data Services Web service, and registers assemblies. After installation, you use the Master Data Services Configuration Manager to create and configure a Master Data Services database in a SQL Server instance that you specify, create the Master Data Services Web application, and enable the Web service.  
  
**Data Stewardship**  
  
Master Data Manager is the data stewardship portal in which authorized business users can perform all activities related to master data management. At minimum, a user can use this Web application to review the data in a master data model. Users with higher permissions can make changes to the master data and its structure, define business rules, review changes to master data, and reverse changes.  
  
**Model Objects**  
  
Most activities in MDS revolve around models and the objects they contain. A model is a container for all objects that define the structure of the master data. A model contains at least one entity, which is analogous to a table in a relational database. An entity contains members, which are like the rows in a table, as shown in Figure 7-1. Members (also known as leaf members) are the master data that you are managing in MDS. Each leaf member of the entity has multiple attributes, which correspond to table columns in the analogy.  
  
  
  
**Master Data Maintenance**  
  
Master Data Manager is more than a place to define model objects. It also allows you to create, edit, and update leaf members and consolidated members. When you add a leaf member, you initially provide values for only the Name and Code attributes, as shown in Figure 7-4. You can also use a search button to locate and select the parent consolidated member in each hierarchy.  
  
  
  
Click on the following link to read further: [Chapter 7: Master Data Services](http://www.c-sharpcorner.com/uploadfile/mspress/chapter-7-master-data-services/)

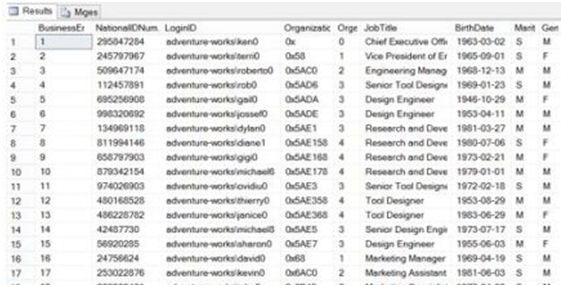
## **Question 40: What is replication in SQL Server?**

**Answer:** Replication is a process or method to synchronize the data across multiple servers. Replication is done by a replica set. A replication maintain the same data set. Replica sets provide redundancy and high availability with multiple copies of data on different database servers.  
  
Replication removes dependencies from a single server so replication protects a database from the loss of a single server. Replication provides a mechanism to recover from hardware failure and service interruptions.  
  
Replication is also used to increase the read capacity.  
  
Replication provides choices for the client to select a different server for read and write operations. Replication maintains copies in different data centers to increase the locality and availability of data for distributed applications.  
  
**Example:** Snapshot Replication  
  
**Step 1**  
  
Open the replication node in your database and choose the option Local Publications.  
 ****  
**Step 2**  
  
Right-click on Local Publications and click on New publication.  
  
  
  
**Step 3**  
  
After clicking on the new publication tab the following window will appear and click on the “Next” button.  
  
  
  
Click on the following link to read further: [Snapshot Replication (Setting up Publisher) in SQL Server 2008: Part 2](http://www.c-sharpcorner.com/UploadFile/97fc7a/snapshot-replication-setting-up-publisher-in-sql-server-20/)

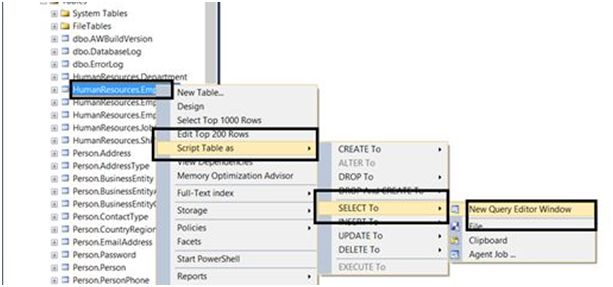
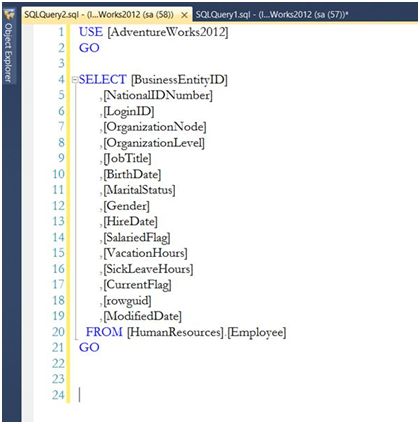
## **Question 43: How to I select data from an SQL Server table?**

**Answer:** How to select specific rows or all columns, selecting distinct rows, filtering with where clause, sorting rows using orderby and so on.  
  
We will be using the AdventureWorks2012 database for this demo.

1. To select all the rows and columns from a table, we use the following query:
   1. **SELECT** \* **FROM** HumanResources.Employee

Execute the query by pressing F5 or via the execute button.  
  
**Output**  
  
  
  
There is another way to select all the columns from a table. Instead of using \* we can specify the column names.

1. **SELECT** BusinessEntityID, NationalIDNumber, LoginID, OrganizationNode, OrganizationLevel, JobTitle, BirthDate, MaritalStatus, Gender, HireDate, SalariedFlag, VacationHours, SickLeaveHours, CurrentFlag, rowguid, ModifiedDate **FROM** HumanResources.Employee

The output will be the same.  
  
If you feel lazy in writing this long query given above then what you can do is go to the Object Explorer window, then expand adventureWorks2012 then select HumanResourcesEmployee table and right-click on it. After that "select script table as" then select "To", then you will see a New query editor window.  
  
  
  
SQL Server will generate the SELECT query for us.  
  


Click on the following link to read further:  [SELECT Statement in SQL Server](http://www.c-sharpcorner.com/UploadFile/219d4d/select-statement-in-sql-server/)

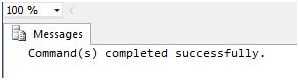
## **Question 41: What is a check in SQL?**

**Answer:** A Check Constraint is a rule that identifies valid values for columns of data. A Check Constraint helps to enforce Domain Integrity. If the condition in a Check Constraint is not satisfied then it prevents the value from entering into the database.  
  
**Syntax:**

1. **Create** **table** tableName(Column1 dataType **Check**(expression), Column2, columnN)

**Example:**

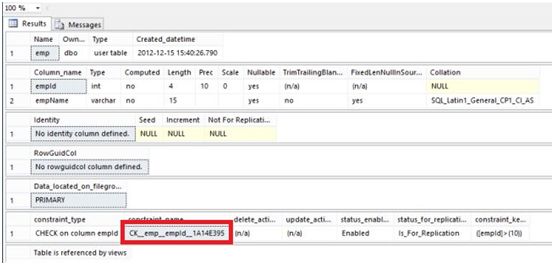
1. **create** **table** emp(empId **int** **check**(empId >10),empName **varchar**(15))

**Output:**  
  


1. **insert** **into** emp **values**(8,'d')

**Output:**  
  
  
  
**Dropping the Check Constraint:**  
  
Firstly, we can determine the name of the constraint using the following command:

1. **exec** sp\_help emp

**Output:**  
  


Click on the following link to read further:  [Check Constraint in SQL Server 2012](http://www.c-sharpcorner.com/UploadFile/63f5c2/check-constraint-in-sql-server-20121/)

## **Question 42: What is a default in SQL?**

**Answer:** Constraints are rules that decide what kind of data can enter into the database tables. SQL server has six types of constraints and we will explore all these constraints here with suitable examples. The constraints that we are going to explore are listed below:

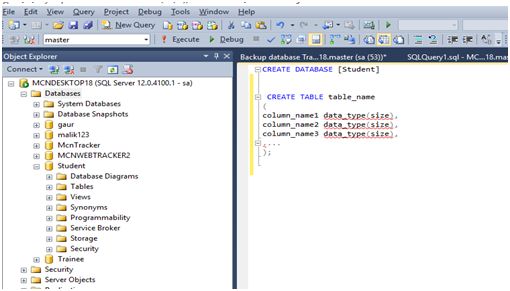
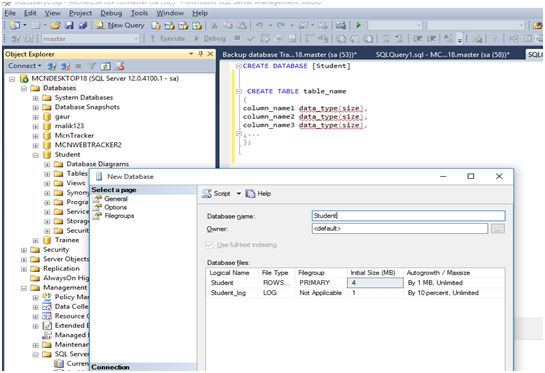
1. Primary Key Constraint
2. Foreign Key Constraint
3. Not Null Constraint
4. Unique constraint
5. Default Constraint
6. Check Constraint

**Default Constraint**  
  
Default constraint allows you set a default value for the column. That means when a row is created for the first time, and there is no entry specified for the column that has a default constraint on it, then the default value is stored in the column.  
  
Note that this not a Not Null constraint and do not confuse the default value constraint with disallowing the Null entries. Default value for the column is set only when the row is created for the first time and column value is ignored on the Insert. Modification to the column with NULL value or even the Insert operation specifying the Null value for the column is allowed.  
  
Let us set the Default value of 1 for the Class. Here are the steps:

* Bring up the table designer
* Select the Class Row as you already did.
* At the bottom of the layout, you will see a Column properties as shown in the below picture. Set the default as shown below:

column properties  
  
For further information click on the link: [Table Constraints in SQL 2005](http://www.c-sharpcorner.com/uploadfile/6897bc/table-constraints-in-sql-2005/)

## **Question 43: How to create a database using SQL?**

**Answer:**A database is described as an organized way of collection of DATA. It is the collection of schemes, tables, queries, reports, views and other objects.  
  
**Syntax:** CREATEDATABASEDatabaseName  
**Example:**CREATEDATABASE Student  
  
  
  
Or you can create Database through Design/ Wizard form by right clicking on DATABASE option - New Database  
  


## **Question 44: What is a constraint in SQL?**

**Answer:**Constraints are the rules that decide what kind of data can enter into the database tables. SQL server has six types of constraints and we will explore all these constraints here with suitable examples. The constraints that we are going to explore are listed below:

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

First Create two tables  
  
To explain these constraints we need two tables. Firstly, let us create these tables. Run the scripts shown below to create the tables. Copy and paste the code into the new Query Editor window, then execute it.

1. **CREATE** **TABLE** Student(StudId **smallint**, StudName **varchar**(50), Class tinyint);
2. **CREATE** **TABLE** TotalMarks(StudentId **smallint**, TotalMarks **smallint**);
3. Go

Note that there are no constraints at present on these tables. We will add the constraints one by one.  
  
**Primary Key Constraint**  
  
A table column with this constraint is called the key column for the table. This constraint helps the table to make sure that the value is not repeated and also that there are no null entries. We will mark the StudId column of the Student table as the primary key. Follow these steps:

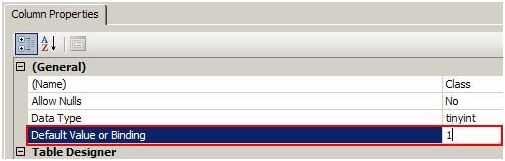
* Right click the student table and click on the modify button
* From the displayed layout select the StudId row by clicking the Small Square like button on the left side of the row.
* Click on the Set Primary Key toolbar button to set the StudId column as primary key column.

Now this column does not allow null values and duplicate values. You can try inserting values to violate these conditions and see what happens. A table can have only one Primary key. Multiple columns can participate on the primary key column. Then the uniqueness is considered among all the participant columns by combining their values.  
  
**Not Null Constraint**  
  
This constraint is useful to stop storing the null entries in the specified columns. We will mark student name column as not null column. This allows us to always having some entries in the student name column of the student table without having NULL. Follow the steps below:

* As you did previously, bring up the table design view by clicking the modify context menu for the table.
* Remove the check mark as shown in the picture below. This action will enable the Not Null constraint for the StudName column.

**Default Constraint**  
  
Default constraint allows you set a default value for the column. That means when a row is created for the first time, and there is no entry specified for the column that has a default constraint on it, then the default value is stored in the column. Note that this not a Not Null constraint and do not confuse the default value constraint with disallowing the Null entries. Default value for the column is set only when the row is created for the first time and column value is ignored on the Insert. Modification to the column with NULL value or even the Insert operation specifying the Null value for the column is allowed.  
  
Let us set the Default value of 1 for the Class. Here are the steps:

* Bring up the table designer
* Select the Class Row as you already did.
* At the bottom of the layout, you will see a Column properties as shown in the below picture. Set the default as shown below:

  
  
For further information, click on the following link: [Table Constraints in SQL 2005](http://www.c-sharpcorner.com/uploadfile/6897bc/table-constraints-in-sql-2005/)

## **Question 45: How do I define constraints in SQL?**

**Answer: Constraints**  
  
Constraints are rules and restrictions applied on a column or a table such that unwanted data can't be inserted into tables. This ensures the accuracy and reliability of the data in the database. We can create constraints on single or multiple columns of any table. Constraints maintain the data integrity and accuracy in the table.  
  
Constraints can be classified into the following two types.  
  
**Column Types Constraints**  
  
Definitions of these types of constraints is given when the table is created.

1. **Create** **Table** My\_Constraint
2. (
3. IID **int** NOT NULL,
4. Salary **int** **CHECK**(Salary > 5000)
5. )

**Table Types Constraints**  
  
Definitions of these types of constraints is given after the creation of the table using the Alter Command.

* 1. **Alter** **Table** My\_Cosntraint
  2. **Add** **constraint** Check\_Constraint **Check**(Age>50)

SQL Server contains the following six types of constraints:

* Not Null Constraint
* Check Constraint
* Default Constraint
* Unique Constraint
* Primary Constraint
* Foreign Constraint

Let us understand each constraint briefly.  
 **Not Null Constraint**  
  
A Not Null constraint restrict the insertion of null values into a column. If we are using a Not Null Constraint for a column then we cannot ignore the value of this column during an insertion of data into the table.  
 **Column Level**  
  
**Syntax**

1. **CREATE** **TABLE** Table\_Name
2. (
3. Column\_Name Datatype **CONSTRAINT** Constraint\_Name NOT NULL,
4. );

**Example**

1. **Create** **Table** My\_Constraint
2. (
3. IID **int** NOT NULL,
4. **Name** nvarchar(50) **CONSTRAINT** Cons\_NotNull not null,
5. Age **int** Not Null,
6. )

**Table Level**  
  
**Syntax**

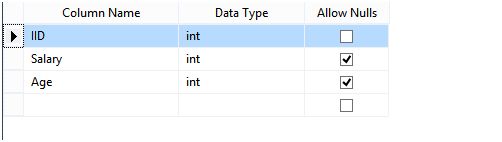
1. **ALTER** **TABLE** Table\_Name
2. **ALTER** **COLUMN** Column\_Name Datatype NOT NULL

**Example**

1. **Alter** **Table** My\_Constraint
2. **Alter** **Column** IId **int** Not Null

**Without SQL Command**

We can also create a Not Null constraint in Microsoft SQL Server without execution of a SQL query.

First right-click on the table and select and click on the design option. Now check all the columns in the “Allow Nulls” option that should have a Null Value.  
  
  
  
**Check Constraint**  
  
A Check constraint checks for a specific condition before inserting data into a table. If the data passes all the Check constraints then the data will be inserted into the table otherwise the data for insertion will be discarded. The CHECK constraint ensures that all values in a column satisfies certain conditions.

For further information click on the link: [Constraints in SQL Server](http://www.c-sharpcorner.com/UploadFile/f0b2ed/constraints-in-sql-server/)

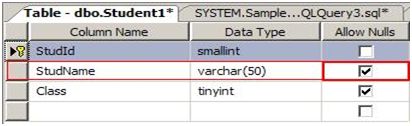
## **Question 46: What is the meaning of Not Null in SQL?**

**Answer:**Constraints are rules that decide what kind of data can enter into the database tables. SQL server has six types of constraints and we will explore all these constraints here with suitable examples. The constraints that we are going to explore are listed below:

* Primary Key Constraint
* Foreign Key Constraint
* Not Null Constraint
* Unique constraint
* Default Constraint
* Check Constraint

This constraint is useful to stop storing the null entries in the specified columns. We will mark student name column as not null column. This allows us to always have some entries in the student name column of the student table without having NULL. Here are the steps:

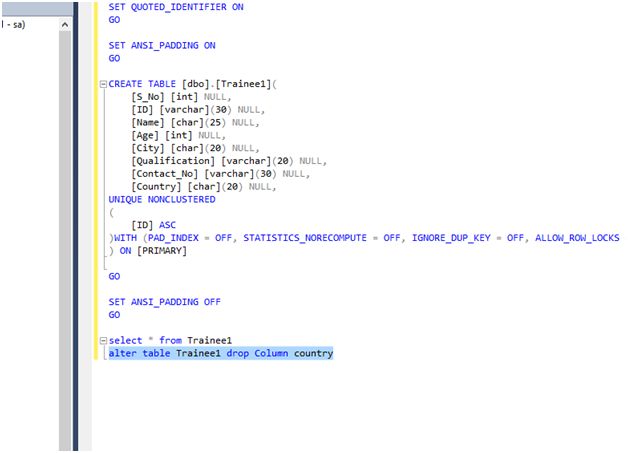
1. As you did previously, bring up the table design view by clicking the modify context menu for the table.
2. Remove the check mark as shown in the picture below. This action will enable the Not Null constraint for the StudName column.

**Example:**  
  
  
  
Click on the following link to read further: [Table Constraints in SQL 2005](http://www.c-sharpcorner.com/uploadfile/6897bc/table-constraints-in-sql-2005/)

## **Question 47: How to alter a table schema in SQL Server?**

**Answer:**  
  
Altering Tables: It is used to modify an existing table.

1. **CREATE** **TABLE** Stock
2. (
3. ID **SMALLINT**
4. );
5. mysql > **ALTER** **TABLE** Stock - > **ADD** **COLUMN** Quantity **SMALLINT** UNSIGNED NOT NULL, - > **MODIFY** ID **SMALLINT** UNSIGNED NOT NULL, - > **ADD** **PRIMARY** **KEY**(ID);
6. mysql > Describe Stock;
7. mysql > **ALTER** **TABLE** Stock;

**Example in Sql :**  
  


Click on the following link to read further: [DROP in MySQL](http://www.c-sharpcorner.com/UploadFile/65fc13/dropping-in-mysql-server/)