Create Database <Database Name>

Use <database Name>

Create table <Table Name> (<Column Name Data type >)

Select\*from <Table Name>

Insert into <Table Name> values(xxxxxx)

Insert into <Table Name> (Column Name) values(xxxx)

Alter Table <Table Name> add <Column Name Data type>

SP\_rename ‘<Table Name.Old Column Name>’,’<New Column Name>’

Update <Table Name> set <Column Name=Data> where <column Name=xxxx>

Delete from <Table Name> where < Column Name=xxx>

Delete from <Table Name> / Truncate Table <Table Name>

Drop table <Table Name>

Select Distinct <Column Name> from <Table Name>

Select Top <4\*> from <Table Name>

Select Top <50\*> Percent from <Table Name>

Select \* from < Table Name> order by <Column Name> Desc

Select \* from < Table Name> where <Column Name> Like ‘x%’

Select \* from < Table Name> where <Column Name> Like ‘%x’

Select \* from < Table Name> where <Column Name> Like ‘%x%’

Select \* from < Table Name> where <Column Name> Like ‘\_\_\_x%’

**Use of Sum ,Min , Max ,Avg ,Count**

Select Sum(<Column Name>) as xxxx from <Table Name>

Select Max(<Column Name>) as xxxx ,<Column Name1> from <Table Name > Where <Column Name3=xxx> Group by <Column Name1> Having Max(<Column Name>)=xxxx

Begin Transaction

Save Transaction xxx

< Update <Table Name> set <Column Name=Data> where <column Name=xxxx>

Rollback /Commit

Create Table <Table Name> (<Column Name Data type Primary Key>)

Create Table <Table Name> (<Column Name Data type Foreign key references Table Name(Column Name) on update cascade on delete cascade>)

**If You use foreign key**

Select A.id,B.name from xxx A,yyyy B where Column Name =xyxy

Select A.id,B.name from xxx A join yyyy B on A.id=B.id

Create Clustered Index <Name for Cluster > on <Table Name (column Name order )>

Create NonClustered Index <Name for Cluster > on <Table Name (column Name order )>

Drop Index table\_name.Index.name

**Case**

SELECT OrderID, Quantity,  
CASE  
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'  
    WHEN Quantity = 30 THEN 'The quantity is 30'  
    ELSE 'The quantity is under 30'  
END AS QuantityText  
FROM OrderDetails;

**Create Procedure**

CREATE PROCEDURE SelectAllCustomers @City varchar(30)  
AS  
SELECT \* FROM Customers WHERE City = @City;

EXEC SelectAllCustomers @City = 'London';

Begin try

Query

End try

begin catch

print error\_message()

end catch

**Right** - RIGHT(*string*, *number\_of\_chars*) or LEFT

**Systemdate&time** - SELECT SYSDATETIME() AS SysDateTime;

SELECT GETDATE();

**Month Name** - SELECT DATENAME(month, '2017/08/25') AS DatePartString;

**Month Number** - SELECT DATEPART(month, '2017/08/25') AS DatePartInt;

**DateAdd** - SELECT DATEADD(day, 54561, '2017/08/25') AS DateAdd;

**DateDiff** - SELECT DATEDIFF(year, '2017/08/25', '2011/08/25') AS DateDiff;

**Charindex** - SELECT CHARINDEX('t', 'Customer') AS MatchPosition;

The CHARINDEX() function searches for a substring in a string, and returns the position.

If the substring is not found, this function returns 0.

**Note:** This function performs a case-insensitive search.

**Substring** - SELECT SUBSTRING('SQL Tutorial', 5, 8) AS ExtractString;

The SUBSTRING() function extracts some characters from a string

**Len** - The LEN() function returns the length of a string.

SELECT LEN(' W3Schools.com ')-4;

Return the length of an expression (in bytes):

SELECT DATALENGTH('W3Schools.com');

**Replicate** - SELECT REPLICATE('SQL Tutorial', 5);

**Stuff** - STUFF(string, start, length, new\_string)

* SELECT STUFF('SQL Tutorial', 1, 3, 'HTML');

***select email,stuff (email,2,len(email)-2,replicate('\*',len(email)-2)) from cust\_info***

**View** - CREATE VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

SQL JOIN

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

## Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

* (INNER) JOIN: Returns records that have matching values in both tables
* LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

      

Replace – select Replace (‘Happy’,’p’,’m’)

Concat – select Concat (‘Jay’,’ Masurkar’)

Concat\_WS - SELECT CONCAT\_WS('.', 'www', 'W3Schools', 'com');

Diffrence - SELECT DIFFERENCE('Juice', 'Jucy');

The DIFFERENCE() function compares two SOUNDEX values, and returns an integer. The integer value indicates the match for the two SOUNDEX values, from 0 to 4

Lower - SELECT LOWER('SQL Tutorial is FUN!');

The LOWER() function converts a string to lower-case

SELECT LTRIM('     SQL Tutorial') AS LeftTrimmedString

The LTRIM() function removes leading spaces from a string

SELECT RTRIM('SQL Tutorial     ') AS RightTrimmedString;

The RTRIM() function removes trailing spaces from a string.

Trim - SELECT TRIM('     SQL Tutorial!     ') AS TrimmedString;

Remove characters and spaces from a string:

SELECT TRIM('#! ' FROM '    #SQL Tutorial!    ') AS TrimmedString;

### **INSERT INTO SELECT Syntax**

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

The INSERT INTO SELECT statement requires that the data types in source and target tables match.

INSERT INTO table2  
SELECT \* FROM table1WHERE condition;

**DATEFROMPARTS** - Return a date from it's parts:

SELECT DATEFROMPARTS(2018, 10, 31) AS DateFromParts;

**IIF** - Return "YES" if the condition is TRUE, or "NO" if the condition is FALSE:

SELECT IIF(500<1000, 'YES', 'NO');

**Set Operaters**

**Union**:- To combine the result of two or more than two select statements as a single unit of values without duplicate values.

Select \* from emp1

Union

Select \* from emp2

**Union All** :- To combine the result of two or more than two select statements as a single unit of values with duplicate values.

**Intersect** :- To return common value from the table

**Except** :- To return all values from left hand side table which are not found in the right hand side table

**Windows Functions –**