Clauses :-

We can add these clauses to a query for providing the additional facilities like filtering rows, storting values, fetching a top most rows, grouping similar data based on columns, finding ~~Some~~ Sum, total and also grand total on the given values in table automatically.

SQL Server supports the following clauses are

(1) ~~Here~~ Where
(2) Order By
(3) Top n
(4) Group By
(5) Having
(6) Roll up
(7) Cube

(1) WHERE :- filtering rows before grouping data in table.

This clause can be used on select, update and delete commands only.

Syntax :-

Select
update → Where <filtering condition>
Delete

Query :-
Select * from Emp where Emp No = 7788

(2) ORDER BY :- Arranging values either ascending or descending manner (providing Sorting Mechanism) by default, 'order by' clause will arrange the values in Ascending manner. then use 'Desc' keyword for arranging the values in descending order.

Syntax :-
Select * from <Table Name> order By <Column Name 1> <ASC / Desc>, <Column Name 2> <ASC / Desc> - - - -

Ex1- Write a query to arrange employee salaries in descending manner
→ Select * from emp order By Salary Desc
(OR)
Select Salary from Emp order by salary Desc.

Ex:- To arrange employee names in ascending manner
⇒ select * from Emp order by EName

Ex:- Write a query to display employee who are working under department Number is 10. and arrange those emp salary in descending manner.

→ Select * from emp order by Salary Desc

Ex:- find the Second Highest Salary from employee table.

① max :-
Select max (Salary) from employee
< (Select max (Salary) from employee)

② Limit :-
Select Salary from (select Salary from employee order By Salary Desc Limit 2) As Emp order By Salary Limit 1.

③ Top :-
Select Top 1 Salary from (select Top 2 Salary from employee order By Salary DESC) AS emp1 order by Salary ASC;

* Group By :-

The group By statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The Group BY statement is often used with aggregate functions (Count(), Max(), MIN() SUM(), AVG()) to group the result-set by one or more columns.

Ex:- Write a query to display a Number of employee working in each job.

→ select Job, Count (*) AS Num-of-Employee from Emp Group By Job

Ex:- Write a query to display Number of employee working in each job along with depatment Number wise

→ Select Job, DeptNo, Count (*) AS Num-of-Employee from Emp Group By Job, DeptNo

Ex:- Write a query to display sum of Salary of DeptNo. wise

→ Select DeptNo, Sum (Salary) from Emp Group By DeptNo

Ex:- Write a query to find out average, minimum & maximum salaries from each department No. wise.

→ Select DeptNo, Avg(Salary) As Avgsal, Min(Salary) As Minsal, max(Salary) As Maxsal from Emp group By Dept No.

* Having clause :-

filtering rows after grouping data in table.

Syntax :-

select <col1>, <col2>, --- <aggregative function Name 1), --- from <Table Name> Group By <col1>, <col2>, --- Having <filtering Condition>;

Ex:- Write a query to display Number of employee of the job from the table in which job the Number of employee is more than 3.

→ select JOB, Count(*) from Emp Group By Job Having Count(*) > 3

Ex 8- Write a query to display sum of salary of the department No. from the table in which department No. the sum of salary is less than 20,0000 (2 Laks).

→ Select DeptNo, sum (Salary) from Emp Group by Dept.No Having sum (salary) < 20,0000

Using all clauses in a single Select Statement :-

Syntax :-

select [Top (n)] <col 1>, <col 2>, ----, <Aggregative function Name 1>, ---- from <Table Name> [Where <filtering Condition> Group By <col 1>, <col 2>, --- order By <Column Name 1> [ASC/DESC], ...];

Ex :- Select Top (1) DeptNo, Count(*) from Emp Where Salary > 15000 Group By Dept.No. Having Count(*) > 3 order By DeptNo DESC

Special Caluses :-

Rollup and cube :- These are special caluses in data base which are used to find subtotal & also grand total based on column's automatically.

These caluses are implimenting along with group By Caluses only.

Syntax for Rollup :-

Select <col1>, <col2>, --- <Aggrigative fimction Name 1>, --- from <Table Name> group By Rollup (<col1>, <col2>, ---);

↓ operational Columns

↓ Supporting Columns

Ex :- on Rollup with a Single Column :-

Ste select DeptNo, Count(*) from Emp Group By Rollup (DeptNo)

	DeptNO	(NO Column Name)	
1	10	4	Sub total
2	20	5	
3	30	6	
4	Null	15	← Grand total

* On Rollup with a Multiple Columns :-

→ Select DeptNo, JOB, Count(*) from emp Group By Rollup (DeptNo, JOB)

Note :- In the above example Rollup is finding Sub & grand total based on a single Column (Department No. Column).
- If we want to find Sub & grand total based on multiple Columns then we use Cube clause.

Syntax for Cube :-
Select <Col1>, <Col2>, --- <Aggregative function Name 1>, --- from <TN> Group By Cube (<Col1>, <Col2>, ---);

All Columns are operational Columns

* Example on Cube with a single Column :-
→ Select DeptNo, Count(*) from Emp GroupBy Cube (DeptNo)

* Cube with multiple Columns
→ Select DeptNo, JOB, Count(*) from Emp Group By Cube (DeptNo, JOB)

* How to Create a new table from the old table :-

Syntax 1 :- Select * into <New table Name> from <old TableName>

Ex:- Select * into NewDept from Dept

In this case we are Creating a new table with copy of all rows & Columns from the old table.

Syntax 2 :- select * Into <New Table Name> from <old tableName> Where <false Condition>;

Ex:- Select * Into Dummy from Dept Where 1=0

In this case we are Creating a New table without Copy of rows from the old table.

* How to copy data from one to another table :-

Before Copy data from one to another table we should follow the following things.

① The Number of Columns & orders of Columns should be same in both table's.

② Those Columns data type's must be match

Syntax :- Insert [Into] <Dest. Table Name> Select * from <Source Table Name>;

Ex :-

Select * from Dept --- Source table
Select * from Dummy Dept --- Dest. Table

Insert into Dummy Dept. Select * from Dept.

~~Roles~~ :- ~~Roles are used to archive data~~

Ex:- Select Count (EName) from Emp
Ans :- 5

iii) Count (Distint <Column Name>) :-
Counting unique values from a specific column. Here distinct keyword is used to avoid a counting of duplicate values

Ex:- Select Count (Distinct Ename) from Emp
Ans :- 3

Ans :- 90,000 → Highest Salary

(iv) Min() :- It returns MIN value
Ex :- Select MIN (salary) from Emp
Ans :- 8000 → Lowest Salary

(v) Count() :- This function again classified into 3 types:

(i) Count (*) :- Counting all value's including duplicate's and nulls.

Ex :- Select Count (*) from Emp

Emp table

EID	EName
101	AA
102	BB
103	NULL
104	CC
105	AA
106	CC

Ans :- All value's Counting → 6

(ii) Count (<Column Name>) :- Counting all value's including duplicate's but Not NULL'S

e) Dateadd() :- Adding the number of intervals to the given date expression.
Ex :- Select Dateadd (DD, 20, GetDate())
Ex :- Select Dateadd (MM, 10, GetDate ()):

f) DateDiff() :- It returns the number of intervals in between the given date expressions.
Ex :- Select datediff (DD, '2018-08-02', '2019-08-02')
Ans :- 365

g) Aggregative function :-

(i) Sum() :- It returns the sum of the given group of values of a Column.
Ex :- Select sum (salary) from Emp
Ans :- Total Sallary Ans ex :- 715,000

ii) AVG() :- It returns average of the given group of values of a Column.
Ex :- Select AVG (salary) from Emp
Ans :- 47666.6666

(iii) MAX() :- It returns MAX value.
Ex :- Select MAX (Salary) from Emp

(iii) Date and Time function :-

A) GetDate() :- It returns the current date & time information of the system
Ex:- Select GetDate()
Ans :- Current Date & Time

b) Get UTC Date() :- It returns the Current universal date & time information, here UTC stands for Co-ordinate universal Time.
Ex:- Select GETUTCDATE()

c) Datepart() :- It returns the specified interval from the given date expression.
Ex:- Select Datepart(DD, Getdate())
Ans:- 2 date
Ex :- Select Datepart(HH, GETdate())
Ans :- 20 hr's.

d) DateName() :- It returns the name of a specified interval from the given date expression.
Ex:- select Datename(DD, GETDATE())
Ans :- friday
Ex :- Select Datename(MM, GETDate())
Ans :- August

Ex:- Select Replicate ('SAI', 5) AS Result
Ans:- SAI, SAI, SAI, SAI, SAI

j) REPLACE() :- To replace an existing characters with new characters in the given string expression

Syntax:- REPLACE ('string', '<old chars>', '<new chars>')

Ex:- select replace ('Jack & Jue', 'J', 'BL') AS Result

Ans:- Black & Blue

k) CONCAT() :- Add two more than two expression

Ex:- Select Concat ('Good', 'Morning') AS Result Ans:- GoodMorning
No space

l) SUBSTRING() :- Substring ('string', <starting position of char>, <length of char>)

Ex:- Select Substring ('Welcome', 4, 2) AS Result

Ans:- CO

e) UPPER() :- It Converts Lower Case characters into upper case characters
Ex:- Select UPPER ('hello') AS Result
Ans :- HELLO

f) LTRIM() :- Trimming the left side space of the given string expression

Ex:- Select LTRIM (' SAI ') AS result
Ans :- only SAI

g) RTRIM() :- Trimming the ~~left~~ right side space of the given string expression

Ex:- Select RTRIM ('SAI ') AS Result
Ans:- only SAI

h) REVERSE() :- It reverse the character at the given string
Ex:- select Reverse ('SAI') AS Result
Ans :- IAS
Ex:- select Ename, Reverse (Ename) from Emp

i) REPLICATE() :- It repeat the given string of characters as per the specified number of types.

g) Log 10 ( ) :- It returns base 10 logarithmic value
Ex:- Select log 10 (10) AS result Ans = 1

ii) character or string function :-
a) LEN( )function :- It returns the length of given string.

Ex:- Select LEN ('HELLO') Ans :- 5
Ex:- select LEN ('WEL COME') Ans :- 8
space is also count
Ex:- Select Ename, LEN (ENome) from Emp
Ex:- Delete from Emp Where LEN (EName) > 3

b) ASCII ( ) :- It returns ascii number for a given character.
Ex:- Select ASCII ('Z') As Result Ans:- 90

c) CHAR ( ) :- It returns character of the given ASCIINO.
Ex:- Select CHAR (90) AS Result Ans:- Z

D) LOWER( ):- It Converts upper case characters into lower case characters.
Ex:- Update Emp set Ename = LOWER (Ename) Where Job = 'clerk'
Ex:- Select LOWER ('HELLO') As Result
Ans:- hello

Ex:- Select ABS(-12) AS Result
Ans :- 12

Ex:- Select EName, salary, Comm, ABS (Comm - Salary) As result from Emp
As result from Emp.

B) Ceiling :- It returns a value which is greater than the given expression
Ex:- Select Ceiling (9-3) Ans :- 10
Select Ceiling (-9.8) Ans :- -9

c) Floor :- It returns a value which is less than to a given expression
Ex:- Select floor (9.8) Ans :- 9
Select floor (-9.3) Ans :- -10

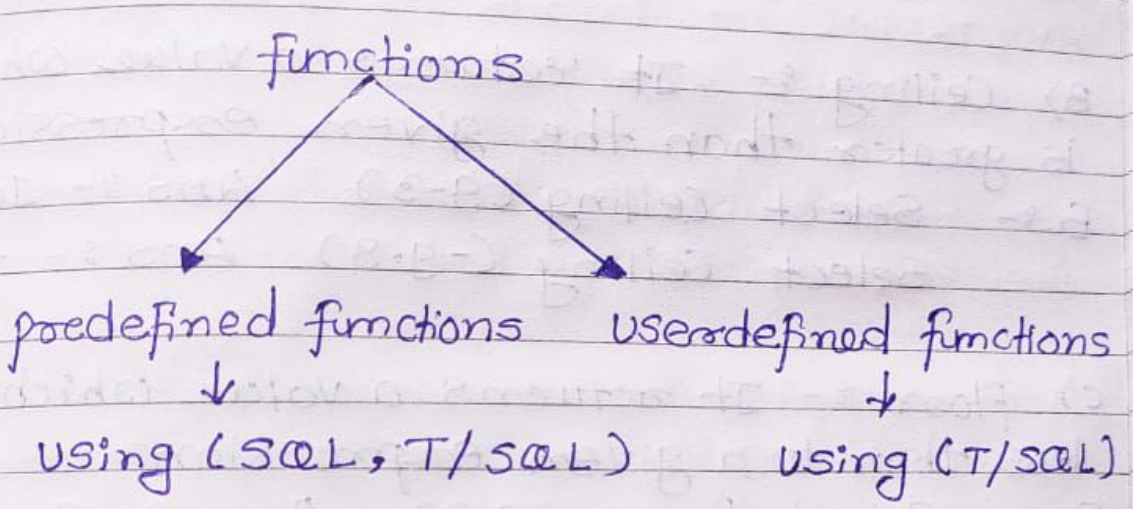
d) power :- It returns the power of the given expression
Ex:- select power (2,3) Ans :- 8

e) pi() :- It returns pi value
Ex:- select pi() Ans = 3.14

f) log() :- It returns the logarithamic value of given expression
Ex:- Select log(3) AS Result Ans :- 1.0986

* functions :- A function is nothing but to perform second task as per the given logic & it must returns a value.

- SQL server supports the following two types of functions there are :



Predefined functions :- SQL Server supports the following predefined function are

i) Number functions
ii) Character / String functions
iii) Date & Time functions
iv) Aggregative / Grouping functions

Syntax :- Select <Function Name> (value/ Expression)

i) Number function :-

A) ABS () :- It Converts negative sign value into to positive sign values

* Write a query to display employee who are working in both branches.

→ Select * from Emp Hyd
Intersect
Select * from Emp chennai

* Write a query to display employee who are working in Hydrabad but not in chennai branch.

→ Select * from Emp Hyd
Except
select * from Emp chennai

* Write a query to display employee who are working in chennai but not in hyd branch.

→ Select * from Emp chennai
Except
select * from Emp Hyd.

Ex:- Create table Emp_HyD (~~EID Int~~ (Eid int, Ename varchar(40), Salary ~~int~~ money)

Create table Emp_chennai (Eid int, Ename varchar(40), Salary Money)

① HyD

	Eid	Ename	Salary
1	101	SAI	85,000
2	102	ADAMS	38000
3	103	JAMES	48000

② Chennai

	Eid	Ename	Salary
1	101	SAI	85,000
2	102	MILLER	62,000
3	103	ALLEN	58,000

* Write a query to display all employee details who are working in the organization.

Ex:-

→ select * from Emp HyD
Union all
select * from Emp chennai

→ Select * from Emp HyD
Union
Select * from Emp chennai

① Union :-

Ex:- Select job from Emp Where DeptNo = 10
Union
Select job from Emp Where DeptNo = 20

DeptNo 10	DeptNo 20
Manager	clerk
president	Manager
clerk	Analyst
Manager	clerk
	clerk

② Union all :-

Ex:- Select Job from Emp Where DeptNo = 10
Union all
Select Job from Emp Where DeptNo = 20

③ Intersect :-

Ex:- Select Job from Emp Where DeptNo = 10
Intersect
Select Job from Emp Where DeptNo = 20

* Example on set operators with multiple tables

Syntax :-

Select * from <TableName1> [Where <condition>]
<set operator>
Select * from <TableName 2> [Where <condition>]

* Set Operators :-

Set operators are used to retrive the data from a single table or multiple tables in vertically.

These operators are :

1. Union
2. Union all
3. Intersect
4. Except

① Union :- It returns all values from all sets without duplicates.

② Union all :- It returns all values from all sets including duplicates.

③ Intersect :- It returns common values

④ Except :- It returns uncommon values from the left side set but not right side.

* Example on set operators with single table.

Syntax :-

Select * from <TableName> [Where <Condition>]
<set operator>
Select * from <Table Name> [Where <Condition>]

Note :- Generally When we use — % symbol's in Where Condition along with like operator. SQL Servers will treat as wildcard operator's, but not the Special characters. So to avoid this problem we should use the special keyword is " Escape '\' "

Ex :- select * from EName Where ENome like '%.\%.' Escape '\'

④ To display employee Who's Name is having % symbol

→ Select * from ~~EName~~ Emp Where ENome like '%.\%.%.' Escape '\'

⑤ To display employee Who's Name starts with A, C, N, W

→ Select * from EName Where Ename like '[A, C, N, W]%.'

Not like :-

Ex :- Write a query to display employee details Who's Name not starts with 'S' Character.

→ Select * from Ename Where Ename not like 'S%.'

Ex:- To display list of employee who are join in the month of feb.

→ Select * from Emp Where Hiredate Like '%-02-%'

* Like operator with special characters

① To Display employee Who's name is having @ Symbol

→ Select * from Emp Where Ename like '%.@%'

② To display employee Who's name is having # Symbol

→ Select * from Emp Where EName like '%#%'

③ To display employee Who's Name is having _ (underscore) Symbol

→ Select * from Emp Where EName Like '%_%' → Wrong Result

Ex:- To Display employee Who's Name starts with 'S' character.

→ Select * from Emp Where EName Like 'S%'

Ex:- To display employee who's employee Name is having a second char is 'O'

→ Select * from Emp Where EName Like '_O%'

Ex:- To display employee who's Name is having four chars.

→ select * from Emp Where EName like '____' (No space betn underscore)

Ex:- To display employee Who's Name Contains 'I' char.

→ Select * from Emp Where EName like '%I%'

Ex:- To display list of employee Who are join in the year 1981.

→ Select * from Emp Where Hiredate Like '1981%'

- select EName, Job, Salary, Comm, Salary + ~~Comm~~ ISNULL (Comm, 0) AS total from Emp Where EName = 'Smith'

	EName	Job	Salary	Comm	Total
1	Smith	clerk	8000	Null	8000

like :- To perform a database operation (select, update, delete) on specific character pattern.
- When we work with like operator we should use the following wildcard operator are.

① % ⇒ It represent the remaining group of characters after selected char in the expression.

② _ (underscore) → Counting a single char

③ [ ] → It represents set of char.

Syntax :-
Where <Column Name> Like '[<wildcard operator>]<special character>[<wildcard operator>]'

Note :- In the above query example the employee smith salary is 8000 & there is no Commission so that salary + Comm is 8000 only but it returns Null.

To overcome with above problem we should use a predefined function in SQL Server is ISNull function.

ISNull (exp1, exp2)

- It is a predefined function which is used to replace a user defined values in place of Null.
- This function is having the following two arguments are exp1, exp2
- If expression 1 is Null then it returns expression 2 value (user defined values)
- If exp1 is Not Null then it returns expression 1 value only.

Ex:- Select ISNull (Null, 0) AS Result --0
Select ISNull (Null, 100) AS Result --100
Select ISNull (0, 100) AS Result --0
Select ISNull (50, 0) AS Result ---50

→ select * from Emp Where COMM is not Null.

Working with Null :-

1) Null is a unknown or undefined value in database.
2) Null is not equal to zero & it not equal to space
3) If any airthmatic operator is performing the operation with Null then it again returns Null only.

i) a + Null → 1000 + Null → Null
ii) a - Null → 1000 - Null → Null
iii) a * Null → 1000 * Null → Null
iv) a / Null → 1000 / Null → Null

Ex:- Write a query to display employee Name, Job, Salary, COMM & also Salary + COMM from the table who's employee Name as Smith.

→ Select Ename, job, Salary, COMM, Salary + COMM AS total from Emp Where Ename = 'smith'

	Ename	Job	Salary	COMM	Total
1	smith	cleark	8000	Null	Null

④ NOT between :- It returns all values except the given range value

Syntax :- Where <Column Name> Not between <low value> AND <High Value>

Ex:- To Display list of employee who are not joined in the year of 1981

→ Select * from Emp Where Hiredate NOT between '1981-01-01' AND '1981-12-31'

⑤ Is Null :-

Comparing Nulls in a table

Syntax :- Where <Column Name> is Null

Ex:-

Write a query to display the list of employee who's Commission is ~~not~~ Null

→ Select * from Emp Where Comm is Null

⑥ IS Not Null :-

We are not Comparing Nulls in a table

Syntax :- Where <Column Name is not Null

Ex:- Write a query to display the list of employee who's Commission is not null

③ Between :- It will work on a particular range values.

Rules -

① It returns all values including source & destination values from the given range

② It can impliment along with AND operator.

③ It always use on low values to High values.

Syntax :-

Where <Column Name> Between <low value> AND <High value>

Ex:- To display employee who's employee salary between 10,000 & 47,000

→ select * from Emp ~~Salary~~ Where salary between 10,000 AND 47,000

(OR)

Select * from Emp Where (salary >= 10,000) AND (salary <= 47,000)

Between = (>= AND <=)

Ex:- To Display the list of employee who are join in the year of 1981

→ Select * from Emp Where HireDate between '1981-01-01' AND '1981-12-31'

Special operators :- sql server supports the following special operators are

① In operator :- Comparing the group of values based on a single condition in the query.

Syntax :-

Where <Column Name> In(<list of values>)

Ex :-

To display the list of employee who are working under the employee Numbers are 7369, 7566, 7788.

→ Select * from Emp Where EmpNO In (7369, 7566, 7788)

②NOT In :- It returns the list of values except the given conditional values

Syntax :-

Where <Column Name> NOT In <list of values>)

Ex :- To Delete list of employee from the table who are not working under the job is Salesman, Manager, president.

→ Delete from Emp Where JOB NOT In ('Salesman', 'Manager', 'president')

It returns a value if any one condition is true from the given group of condition

Syntax :-

Where <Condition 1> OR <Condition 2> OR <Condition 3> OR ----

Ex :-

Write a query to display a list of employee who are working under the employee Numbers are 7369, 7566, 7788

→ Select * from Emp Where EmpNo = 7369 OR EmpNO = 7566 OR EmpNo = 7788

Not :- It returns all values except the given Conditional values in the query

Syntax :-

Where NOT <Column Name> = <Value> AND NOT <ColumnName> = <Value> AND ---;

Ex :-

Write a query to delete the list of employee from the table Who are not working under the job is clerk & analyst.

→ Delete from Emp Where NOT JOB = 'Clerk' AND JOB = 'Analyst'

④ logical operators :- To cheak more than one Condition in the query. These operator's are AND, OR, NOT

AND Operator :-

Cond1	Cond2	Result
T	T	T
T	F	F
F	T	F
F	F	F

It returns a value when a given all Conditions are true in that query.

Syntax :- Where <Condition 1> AND <Condition 2> AND <Condition 3> ---

Ex :-

Write a query to display employee who are working in the job is cleark & Who's Name is 'Hard'

→ Select * from Emp Where Job = 'cleark' AND Ename = 'Hard'

OR :-

Cond 1	Cond 2	Result
T	T	T
T	F	T
F	T	T
F	F	F

Ex:- update Emp set Salary = Salary + Salary * 10/100 Where JOB = 'Analyst'

④ Write a query to update all employee salaries with a increment of 5%.

Ex:- update Emp$ set Salary = Salary + Salary * 0.5

Relational operators :-

Comparing a specific Column values with even user defined Conditions.

Syntax :- Where <Column Name> <Relational operator> <Value>

Ex:-

① Write a query to display list of employee who are joined before 1981?

→ Select * from emp where Hiredate < '1981-01-01'

② Write a query to insert studentid, Student Name and there subject marks find total, average & class of each student

→ perform by your own.

Ex:- Declare @ x int
Set @ x = 101

② Arithmatic operator :- To perform some mathematical calculations like addition, substraction, multiplication & division.

Syntax :- <Column Name> <Arithmatic operator> <value>

Ex:-

① Write a query to display employee salaries after adding 1000

→ Select Salary, Salary + 1000 AS Result from Emp

② Write a query to display employee Name, job, Salary & annual salary of the employee from the table.

Ex:- Select EName, Job, Salary, Salary * 12 AS Annual Salary from Emp.

③ Write a query to Update employee salary with a increment of 10% of working with job is analyst.

Operators :- To perform some operations on given operands values sql server supports the following operators are.

① Assignment operator → =
② Arithmatic operator → + - * /
③ Relational operator → <, >, <=, >=, !<, !>,
④ Logical operator → AND, OR, NOT
⑤ set operator's → Union, Union all, Intersect, Except
⑥ Special operator →

positive operator	Negative operator
In	Not in
between	Not between
is Null	is not null
Like	Not Like

① Assignment Operator :- To assign a value to a variable or to a attribute
In SQL :-
Syntax :- <Column Name> < Assignment operator> <value>

Ex :- Select * from Emp Where EmpNo = 7788
In T/SQL :- Syntax :-
Declare @ <variable Name> <DT> [size]
Set @ <variable Name> <Assignment operator> <value>

following syntax

Set identity_insert < Table Name > OFF/ ON

Here,

OFF :- It is a default Connection of identity, the user Can-not insert Value's to identity Column by explicitly.

ON :- User Can insert value's to on identity Column by explicitly.

Ex :- Set identity_insert Test ON
Insert Test (SNO, Name) Values (3, 'C')
--- Allowed.

Ex :- Set identity_insert Test OFF
Insert Test (SNO, Name) Values (4, 'D')
--- Error.

Insert Test1 (SNo, Name) values (2, 'B')
--- Error.

Insert Test1 (Name) values ('B') - Allowed

Table output :-

	SNo	Name
1	1	A
2	2	B

Example of identity with user defined value's

Create table Test2 (SNO Int. identity (100,5), Name Varchar (30))

Testing :-
Insert Test2 values ('A') -- Allowed
Insert Test 2 (Name) values ('B') - Allowed

Table output :-

	SNO.	Name
1	100	A
2	105	B

Note :- In the above example's user can not insert value's to and identity Column by explicitly.
- If we want to insert value's to an identity Column by explicitly then we follow the

Identity (seed, Increment) :-

It is a predefined method which is used to generate the identity values on a perticular column in the table automatically.

By using identity we will provide autoincrement value facility on table.

- A table should contain only one identity column.
- This method is having the following two arguments are :

i> Seed :- It represent starting value of identity default value is 1.

ii> Increment :- To represent incremental value in between ids, default value is 1.

Identity (seed, Increment) = identity (1, 1)

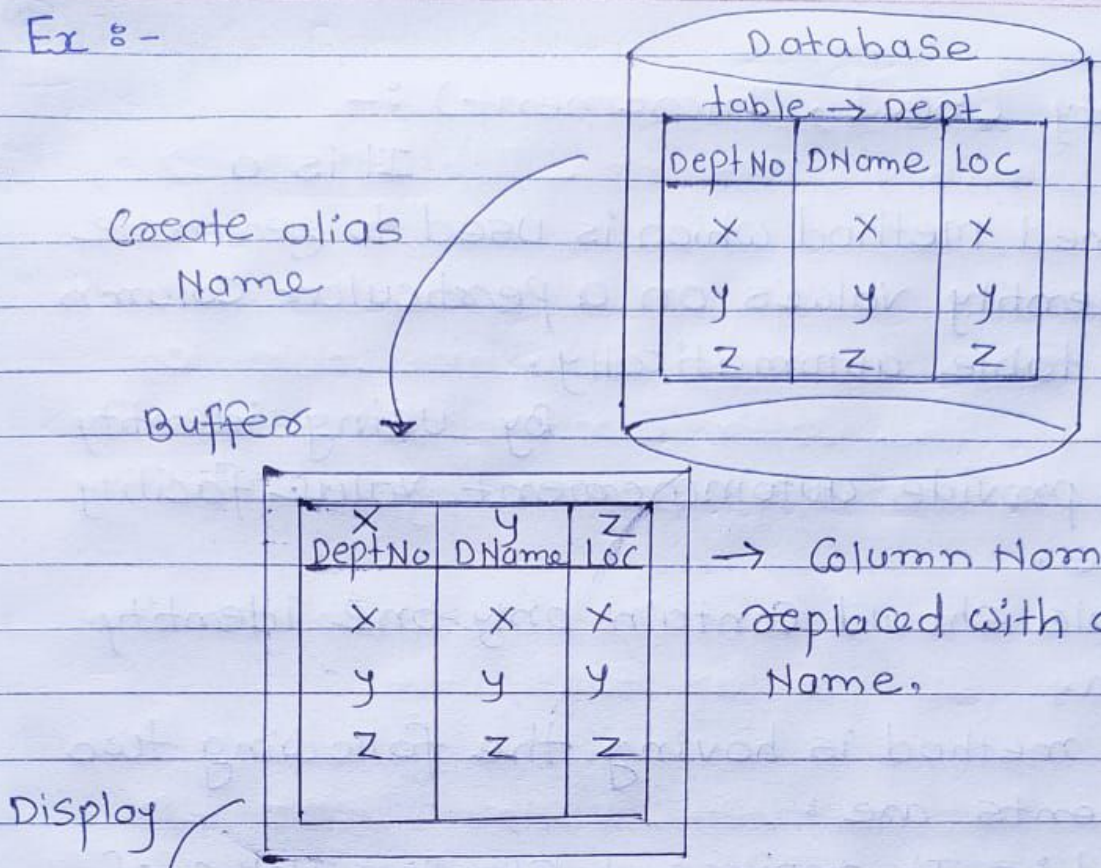
~~Eg~~ - Example of identity with default values (seed, increment)

```
Create table Test1 (SNo Int identity,
Name Varchar(30));
```

Testing :-

```
Insert Test1 values (1, 'A') -- Error
Insert Test1 values ('A') -- Allowed
```

Ex :-



Note :- Whenever we are Creating alias Name's on table or Column internally a database Server is Creating the virtual Copy on each alias Name & Storage in buffer memory

i) Column level alias name's :- In this level we are creating alias names for Columns in a table.

Syntax :- <ColumnName> [AS] <Column alias Name>

Ex :- DeptNo AS Dept (OR) DeptNo Dept

ii) Table level alias Names :- In this level we are creating alias names on table Name.

Syntax :- <TableName> [AS] <Table Alias Name>

Ex :- Dept AS D (OR) Dept D

Syntax to Combined Column + table level alias Name :-

Select <Column Name 1> [AS] <Column Alias Name 1>, <Column Name 2> [AS] <Column Alias Name 2>, ---:-- from <TableName> [AS] <Table Alias Name>;

Ex :- select Deptno as x, DName AS y, Loc AS z from Dept AS D

(OR)

Select DeptNo X, DName y, Loc Z from Dept D.

③ DQL :- Data Query Language

This Language is having only one command that is select.

i) select :- Retriving data from the table

Syntax :- Select *<list of Colums> from <TableName> [Where <condition>];

Here 'Star' is represent all Column's in the table.

Ex :- ① Write a query to display department details.

- Select * from Dept

(OR)

- Select DeptNo, DName, Loc from Dept

② Write a query to display employee who are working in the job is Manager.

- Select * from Emp Where job = "Manager"

Alia's Names :-

Alias is nothing but duplicate or temporary or alternative name for the original Column name or table name. We Can Create alias name's on two levels in database.

Ex:- Write a Query to delete employee from the table who are working in the job is cleark.

Delete from Emp Where job = 'cleark'

Ex:- Write a query to delete all emp details from the table

Delete from Emp

Difference between delete & Truncate

Delete	Truncate
① It is DML operation	① It is DDL operation
② It can delete a specific row from the table	② It is not possible
③ It support Where clause condition	③ It dosent support Where clause condition
④ It is the temporary data deletion	④ It is permant data deletion
⑤ We can restore deleted data by using rollback	⑤ We can not restore deleted data by using rollback
⑥ Execution speed is slow	⑥ Execution speed is fast.

Ex:- Insert student (STID) values (106), (107) (108)

② update :- Updating all rows data in a table at a time or a specific row data in a table by using 'Where' Condition.

Syntax :-

Update <TableName> SET <ColumnName1> = <value1>, <ColumnName2> = <value2> ---- [Where Condition];

Ex:- Write a query to update employee job as HR, Salary as 14,000 & who's employee Number is 7788

update emp set job = 'HR', Salary = 14000 where EmpNo = 7788

Ex:- Write a query to update all employee Commition as 500

update Emp set COMM = 500

③ Delete :- Deleting all rows from the table at a time or a specific row from the table by using where clause Condition

Syntax :-

Delete from <TableName> [Where<Condition>];

Ex:- Create table student (STID int, SName varchar(40), sfee decimal(6,2), Age tinyint)

Ex:-

Insert into student values (101, 'SAI', 2500, 21)

OR

Insert student values (102, 'JAMES', 4500, 23)

ii) Explicit Method :- Inserting value's for required column's only (with left any column in the table)

Syntax :-

Insert [INTO] <TableName> (Required Column Names) values (103 'ALLEN')

How to Insert Multiple rows into a table

Syntax for implicit :-

Insert [INTO] <TableName> values (Row1 values), (Row2 values) ---- ?

Ex:-

Insert into student values (104, 'Scott', 1800, 22), (105, 'Hard', 1000, 25)

Syntax for Explicit :-

Insert [INTO] <TableName> (Required Column Names) Values ((Row1 values), (Row2 values),

③ Truncate :- Deleting rows from the table but not structure of the table. by using truncate command we can not delete a specific row from the table because it dosent support 'where' clause condition.

Syntax :- Truncate table <TableName>;
Ex :- Truncate table Student

④ Drop :- Dropping a table from a database permantly
Syntax :- Drop table <TableName>;
Ex :- Drop table student;

② DML :- (Data Manupulation Language)
This Language Commands are used to change or manipulate data in database table
1> Insert :- Inserting a new row into a table. There are two methods to insert row's into a table
i> Implicit Method :- Inserting all value's for all column's into a table (without left any column)
Syntax :- Insert [into] <TableName> values (value1, value2, value3 ----);

Sp_HELP is predefined Stored ~~proctu~~ procedure

② ALTER :- To change or modify the structure of a table or a database.
by using the Alter Command we can perform a following four operations on existing table. To perform these operations we acquired subcommands of alter.

i) Alter - Alter Column
ii) Alter - Add
iii) Sp - Rename
iv) ALTER - Drop

i) Alter - Alter Column :- To change datatype & also size of the datatype of a particular column.
Syntax :-
Alter table <TN> Alter Column <Column Name> <New DT> [New Size];

Ex :- Alter table Student Alter Column SName Varchar (50);

ii) Alter - Add :- Adding a new column to add existing table.
Syntax :- Alter table <TN> ADD <New Column Name> <DT> [Size];
Ex :- Alter Table Student ADD SAddress Varchar (30)

① DDL (Data Defination Language)

This language commands are using to define, modify & Drop on object of database from SQL server.

① Create :- Creating a new database or new table in SQL Server

Step 1 :- Create a new database in SQL server

Syntax :- Create database <DB NAME>;

Ex :- Create database MYDB;

Step 2 :- Select The required database from SQL Server.

Syntax :- USE <DB Name>;

Ex :- USE MYDB;

Step 3 :- Create new table in database

syntax :- Create table <Table Name> (<Column Name1> <DT> [size], <Column Name 2> <DT> [size] -----); -- 1000 Columns

Ex :- Create Table Student (sid Int, Sname char(10), Sfee Decimal(6,2), AGE Tinyint);

Step 4 :- To view the Structure of table

Syntax :- Sp_HELP <Table Name>;

Ex :- Sp_HELP Student;

- Every SQL statement should ends with a semicolon but it is optional in SQL server.

Sublanguages of SQL

① Data Defination Language (DDL)
- Create
- Alter
- Sp_Rename
- Truncate
- Drop

② Data Manupulation Language (DML)
- Insert
- Update
- Delete

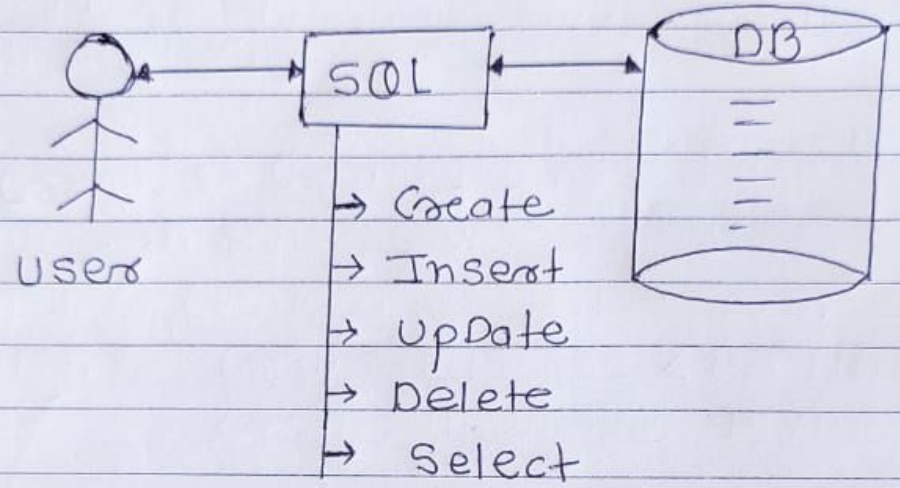
③ Data Query Language (DQL)
- Select

④ Transaction Control Language (TCL)
- Commit
- RollBack
- Savepoint

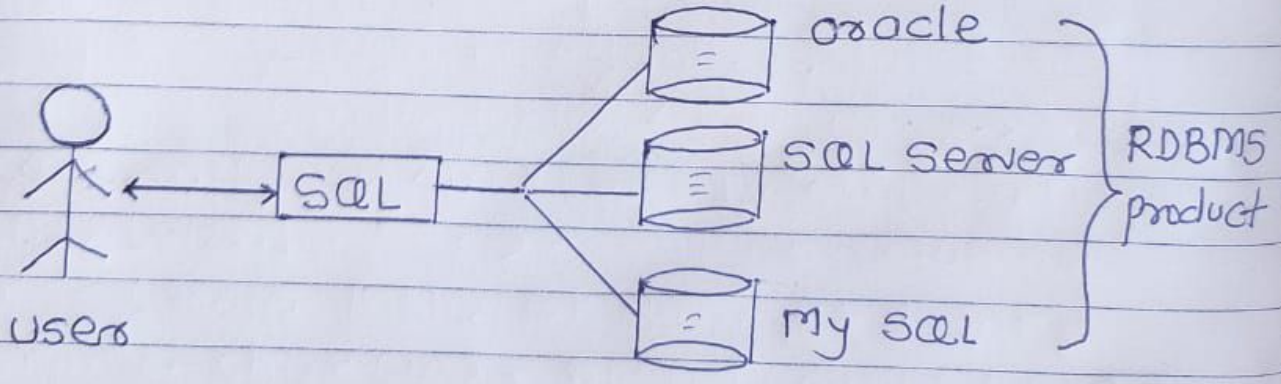
⑤ Data Control Language (DCL)
- Grant
- Revoke

SQL (structured Query Language)

SQL :- SQL is a non procedural Language which was introduced by the IBM in 1970's. Which is used to communicate with database



- SQL is also called as sequeal or CLI language (Common Language Interface). This is only the language which can use to communicate with any RDBMS product.



SQL is not Case Sensitive language that we can write SQL predefined queries or Syntaxes in any Case characters. (either upper or lower)