Create a table copying the selected column from another table :-

Syntax :

```
Create table_Name (column1, column2..)
As (select * from oldtable_Name);
```

for eg :- Create a table student from an existing student department table.

```
create table student
As (select * from studentdepartment);
```

Operators And functions :-

1 Arithmetic Operators (+, −, ×, ÷) :-

Student data table

Name	Roll_No	Fees
Jashandeep	101	15000 /-
Chatan	102	12000 /-
Gagan	103	15000 /-
Rupinder	104	10000
Harmandeep	105	11000 /-
Sumandeep	106	12000 /-
Trishna	107	16000 /-
Pawandeep	108	14000 /-
Harmandeep	109	12000 /-

Pawandeep - Harmandeep
Harmandeep Sumandeep
Sumandeep Trishna
Trishna - Pawandeep

(i) Update the student fees by 500 Rupees where roll no is 104

```
Update Student data
Set fee = fee + 500
where Roll_No = 104;
```

(ii) Update the student fees of student whose roll no is 105 and 103 and fees is deducted (reduce) by 1000.

```
Update Student_data
set fee = fee - 1000
where Roll_no = 105 AND Roll_no = 103;
```

2 Relational or comparision operators :- These operators are used to compare column data with specific value. These operators are used to filters the data based on specific condition.

Its syntax (filter out) ↓ here these are called logical operators

operators are → <, >, <=, >=, =, != (also written as <>)

Query

for example :- List the students records whose fees is greater than or equal to 13000

```
Select * from Student_data
where fee >= 13000;
```

Output :-

Name	Roll_no	Fee
Tashardeep	101	15000
Gagan	103	15000
Tanisha.	107	16000
Pawandeep	108	14000

(iii) List the name of student and roll_no whose fees is not equal to 15000

Select * from Student_data
where fee != 15000 ;

Name	Roll_no	fee
Chetan	102	12000
Rupinder	104	10000
Harmandeep	105	11000
Sumandeep	106	12000
Tanisha	107	16000
Pawandeep	108	14000
Harmandeep	109	12000

3. Logical operator :-

(1) AND :- It is used when want to compair more than one condition.

Query → select name of student and roll_no from student_data table whose name is Harmandeep and fee greater than 1000

Select name, Roll_no from Student_data where name = 'Harmandeep' AND fee > 10000;

Name	Roll_no
Harmandeep	105
Harmandeep	109

(2) OR :- In 'Or' operator if there are more than one condition to get the result atleast one of the condition must be true.

Page No.
Date / 20

Query → Display the name and roll_no. of student whose name is Hasmandeep or fee > 10000

Select name, Roll_no from student_data where name = 'Hasmandeep' OR fee > 10000;

Name	Roll_no.
Jashandeep	101
Chatan	102
Gagan	103
Hasmandeep	105
Sumandeep	106
Trishna	107
Pawandeep	108
Hasmandeep	109

Query → Display the name and fee of student whose roll_no is 102 or fee > 14000

Select name, fee from student_data where roll_no = 102 OR fee > 14000

Name	fee
Chatan	12000
Jashandeep	15000
Gagan	15000
Trishna	16000

(2) NOT :- If user want to find the record that doesnot satisfie the condition then user can uses NOT operator.

Page No.
Date / 20

Query → Display name and roll_no of student ho whose fee != 15000.

Display name, Roll_no from Student_da where fee NOT (15000);

symbol != also ca

Name	Roll no.
Chetan	102
Rupinder	104
Harmandeep	105
Simandeep	106
Teishna	107
Pawandeep	108
Harmandeep	109

Concatenation operator : This operator is used to combine the string. It is represented by two vertical bar. There is another concatenation functio CONCAT used to combine the string.

for example - (i) list the name of student whos roll_no is 101

Select 'My name is' || name from student_d where Roll_no = 101;

Output → My name is Jashandeep.

(ii) display name and fee of student whos roll_no is 101

Page No.
Date / 20

Output → Jashandeep | fee is | 15000
↓ Name | ↓ string | ↓ fee

Select name || 'fee is' || fee from student data where Roll_no = 101;

Column aliasing :- This is used to provide an alternated name to the column.

For example :-

(i) Select name, fee, fee+1000 As Total_fee from student_data where Roll No=101;

Output →

Name	fee	Total_fee
Jashandeep	15000	16000

(ii) Select name, fee As amount from student_data where Roll_no=101;

Output →

Name	Amount
Jashandeep	15000

Special Other operators :-

1. LIKE → for patern matching
2. <NOT> NULL
3. BETWEEN → for range
4. IN/NOT IN

* Like operator :- The ~~ASI~~ SQL like operator is

used to check for matching the pattern. It is pattern matching operator used to search the value according to the pattern defined by user.

There are two pattern matching operator symbol available such as including

① % :- match any string of zero or more character

② _ (Underscore) :- matching exactly any single character.

Query - (i) Display the name of student whose name start with alphabet H

```
Select name from Student_data
where name LIKE 'H%';
```

Output →

Name
Harmandeep
Harmandeep

(ii) Select name, roll_no of a student whose second letter is small 'a'.

```
Select name, Roll_no from Student_data
where name LIKE '_a%';
```

Output →

Name	Roll_no
Jashandeep	101
Gagan	103
Harmandeep	105
Pawandeep	108
Harmandeep	109

(iii) Display the name of student whose name contain first letter 'H' and fourth letter 'm'.

```
Select name from Student_data
where name LIKE 'H__m%';
```

Output →

Name
Harmandeep
Harmandeep

* In / NOT IN :- The IN operator allow user to specified multiple value in a where clause the IN operator compare the value of an expression with value in a set

Query - (i) display the name and roll_no of Student whose name is Gagan, Chetan, ~~Krishna~~ Akash.

```
Select name, Roll_no from Student_data
name IN ('gagan', 'Chetan', 'Akash');
```

Name	Roll_no
Gagan	103
Chetan	102

SET operator :- There are situation where we need to combine the result from two or more table select command enable us to handle these requirement by using set operator.

→ Common points to remember :-

- While using set operator → the datatype and number of column set by query should be same. The name of corresponding column can be different.
- Null values using set operator are considered to be equal to each other.

→ type of sets :-

* Union :- • It combine the result of two sequence queries into single table of all matching rows. These two queries must have same number of column and compatable table column type.
- It eliminate duplicate values between two or more queries.

Name	Roll_no	Address
Amit	101	Ldh
Ajay	102	Ldh
Mohit	103	Fzd

BCA ←

Name	Roll_no	Address
Amar	201	Fzd
Ajay	202	Ldh
Mohit	203	Ldh.

BSc →

- Syntax :- <select statement 1>
  Union
  <select statement 2>
- For example :-

```
select name, address from BCA
UNION
select name, address from BSc;
```

Page No.
Date / 20

O/P →

Name	Address
Amit	Ldh
Ajay	Ldh
Mohit	Fzd
Amar	fzd
Mohit	Ldh

* UNION ALL :- In case of union all, it will display all the record including duplicate record but in case of UNION it eliminate the duplicate records.

- Syntax :- <select statement 1>
  UNION ALL
  <select statement 2>
- For eg :- select name, address from BCA
  UNION ALL
  select name, address from BSC;

O/P →

Name	Address
Amit	Ldh
Ajay	Ldh
Mohit	Fzd
Amar	Fzd
Ajay	Ldh
Mohit	Ldh

Name	Role no	Address
Amit	101	Ldh
Ajay	102	Ldh
Mohit	103	FZR

BCA table.

Name	Role no	Address
Amar	201	FZR
Ajay	202	Ldh
Mohit	203	Ldh

BSc table

* INTERSECT :- It is used to display the common record in two or more table

Syntax :-

```
Select Statement 1
INTERSECT
select Statement 2
```

eg :-

```
Select name, Address from BCA
INTERSECT
select name, Address from BSc;
```

Output →

Name	Address
Ajay	Ldh

* Minus operator :- the minus query will compare each record of table in statement 1 to the record in statement 2. The result will return ~~compare~~ record in select statement 1 and data are in select statement 2.

Syntax :-

```
Select Statement 1
MINUS
Select Statement 2.
```

~~select name, Address from B~~

eg :-

```
Select name, Address from BCA
MINUS
Select name, Address from BSc
```

Output →

Name	address
Amit	Ldh
Mohit	Ldh