Operator = and !=

```
SELECT f_name,l_name
from employee_data
where title="Programmer";

SELECT f_name, l_name
from employee_data
where age = 32;
```

- For **data type integer *NO NEED*** to put ‘’ **or** “”
- **!=** means **not equal to**.

Operator < and >

- To list employee whose **age more than 32 years**.

```
SELECT f_name, l_name
from employee_data
where age > 32;
```

Operator < and >

```
SELECT f_name, l_name
from employee_data
where salary > 120000;
```

```
SELECT f_name, l_name from
employee_data where yos < 3;
```

Operator <= and >=

- To list name, age and salary of employees whose age more than and equal to 33 years.

```
SELECT f_name, l_name, age, salary
FROM employee_data
WHERE age >= 33;
```

Operator <= and >=

```
SELECT f_name, l_name
FROM employee_data
WHERE yos <= 2;
```

Logical Operator

- SQL also consist of Boolean operator:
  1) AND
  2) OR
  3) NOT

- For example, if we put range for this condition:

```
SELECT f_name, l_name
from employee_data
where salary > 70000 AND
salary < 90000;
```

Logical Operator

- To display employees who work more than 3 years and age more than 30 years old.

```
SELECT f_name, l_name
 FROM employee_data
 WHERE yos>3 AND age>30;
```

PATTERN MATCH : LIKE Operator

- To display last name which start with S or A

```
SELECT l_name
from employee_data
where l_name like 'S%' OR
l_name like 'A%';
```

LIKE Operator

```
SELECT f_name,l_name,age
FROM employee_data
WHERE (l_name like 'S%` OR l_name like 'A%')
      AND age < 30;
```

NOT Operator

- **NOT** Operator will display a list of employees who are not programmer (including Senior programmers, Multimedia Programmers and Programmers).

```
SELECT f_name, l_name, title
FROM employee_data
WHERE title NOT LIKE "%programmer%";
```

IN and BETWEEN Operator

- To display employees who work as Web Designers dan System Administrators:

```
SELECT f_name, l_name, title
from employee_data
where title = 'Web Designer'
OR    title = 'System
Administrator';
```

IN and NOT IN Operator

```
SELECT f_name, l_name, title
FROM employee_data
WHERE title IN ('Web Designer','System
Administrator');
```

- To display vice versa use NOT IN operator.

```
SELECT f_name, l_name, title
FROM employee_data
WHERE title NOT IN
('Programmer','Marketing Executive');
```

BETWEEN Operator

- BETWEEN operator can be used to display a range according to condition.

```
SELECT f_name, l_name, age
FROM employee_data
WHERE age BETWEEN 32 AND 40;
```

NOT BETWEEN Operator

- To display employees and salary which is less than RM90000 and more than RM150000.

```
SELECT f_name, l_name, salary
FROM employee_data
WHERE salary
NOT BETWEEN 90000 AND 150000;
```

SEE U NEXT CLASS !!!

ORDER BY CLAUSE

- The results of a query are sorted using the ORDER BY clause in the SELECT statement.
- The ORDER BY consists of a list of column identifiers that the result is to be sorted on, separated by commas.

```
SELECT l_name, f_name from
employee_data
ORDER BY l_name;
```

ASC and DESC

```
SELECT f_name, l_name, age
from employee_data
ORDER BY age;
```

- By default, the data will be arranged ascendingly. Data will be arranged descendingly by request.

```
SELECT f_name from employee_data
ORDER by f_name DESC;
```

LIMIT Clause

- Data that will display can be restricted using **LIMIT** clause.
- To list the first of five employees :

```
SELECT f_name, l_name
from employee_data LIMIT 5;
```

LIMIT Clause

```
SELECT f_name, l_name, age
from employee_data
ORDER BY age DESC LIMIT 4;

SELECT f_name, l_name, age
from employee_data
ORDER BY age LIMIT 2;
```

LIMIT with condition…

- To put condition, for example to begin data from certain row.

```
SELECT <column_name>
from <table>
LIMIT <row begin with, number>;

SELECT f_name, l_name
from employee_data
LIMIT 6,3;
```

DISTINCT

- To display unrepeatable data using DISTINCT .
- To display title in the company:

```
select title
from employee_data;

select DISTINCT title
from employee_data;
```

DISTINCT

```
select DISTINCT age
from employee_data
ORDER BY age;
```

- **DISTINCT** normally used with **COUNT**

SQL AGGREGATE FUNCTIONS

- The SQL standard defines 5 aggregate functions:
  1) MIN : returns the smallest value in the specified column
  2) MAX : returns the largest value in the specified column
  3) SUM : returns the sum of values in the specified column
  4) AVG : returns the average of values in the specified column
  5) COUNT: returns the number of values in the specified column

- **Minimum value**

```
select MIN(salary)
from employee_data;
```

MAX and SUM

- **Maximum value**

```
select MAX(salary)
from employee_data;
```

- **Sum value**

```
select SUM(salary)
from employee_data;
select SUM(perks)
from employee_data;
```

SUM

- Display all of the salary and perks for the employees.

```
select sum(salary) +
sum(perks) from
employee_data;
```

AVG

```
select avg(salary) from employee_data;
```

AS

- To rename a new column using **AS** clause.

```
select avg(salary)
AS 'Average Salary'
from employee_data;

select (SUM(perks)/SUM(salary)
* 100) AS 'Perk Percentage'
from employee_data;
```

COUNT

- To count and display the number of records.

```
select COUNT(*)
from employee_data;

select COUNT(*)
from employee_data
where title = 'Programmer';
```

GROUP BY clause

- To group the same data.

```
select title
from employee_data
GROUP BY title;
```

- Equivalent to the function of DISTINCT.

GROUP BY clause

- To display the number of employees according to title.

```
select title, count(*)
from employee_data
GROUP BY title;
```

SORTING DATA

- To display the number of employees according to title and sort the data.

```
select title,
count(*)AS Number
from employee_data
GROUP BY title ORDER BY Number;
```

Example:

- To display the average salary of employees from various title:

```
select title, AVG(salary)
from employee_data
GROUP BY title;
```

Example:

- To display title which average salary more than $100000, **HAVING** clause is used.

```
select title, AVG(salary)
from employee_data
GROUP BY title
HAVING AVG(salary) > 100000;
```

CONCAT function:

- Value can be combined using **CONCAT**

```
SELECT CONCAT(f_name, " ", l_name)
from employee_data
where title = 'Programmer';
```

CONCAT function:

- To rename a column using AS :

```
SELECT CONCAT(f_name, " ",
l_name)AS NAME
from employee_data
where title = 'Programmer';
```

UPDATE

- To modify data using **UPDATE** clause :

```
UPDATE <table_name>
SET <column1_name> = <value1>,
< column2_name > = <value2>,
< column3_name > = <value3> ...
[WHERE <condition>];
```

- UPDATE clause will update all records in a column.

UPDATE

```
UPDATE employee_data
SET salary=220000, perks=55000
WHERE title='CEO';
```

UPDATE

```
UPDATE employee_data
SET salary = salary +
20000,perks = perks + 5000
WHERE title='CEO';

update employee_data
SET title = 'Web Developer'
WHERE title = 'Web Designer';
```

DELETE

- SQL DELETE statement will request for table name and condition.

```
DELETE from table_name
[WHERE condition];
```

- All data will be deleted from table if no condition applied.

```
DELETE from employee_data
WHERE emp_id = 10;
```

DATE operation

- **Date type column allow operations like sort, condition, etc.**

- Using **= dan != operators**

```
select p_email, phone
from employee_per
where birth_date = '1969-12-
31';
```

DATE operation

```
select e_id, birth_date
from employee_per
where birth_date >= '1970-01-
01';
```

DATE operation

- **To put a range**

```
select e_id, birth_date
from employee_per
where birth_date BETWEEN '1969-
01-01' AND ‘1974-01-01';

select e_id, birth_date
from employee_per where
birth_date >= '1969-01-01'AND
birth_date <= '1974-01-01';
```

DATE operation

- Using DATE to sort data:

```
select e_id, birth_date
from employee_per
ORDER BY birth_date;
```

DATE operation

- **To display the data of employees who born in March.**

```
select e_id, birth_date
from employee_per
where MONTH(birth_date) = 3;


select e_id, birth_date
from employee_per
where MONTHNAME(birth_date) = 'January';
```

DATE operation

- The name of month is case sensitive
  (for example : *January* not *JANUARY*)

```
select e_id, birth_date
from employee_per
where year(birth_date) = 1972;

select e_id, birth_date
from employee_per
where DAYOFMONTH(birth_date) = 20;
```

DATE operation

```
select e_id, birth_date
from employee_per
where MONTH(birth_date) =
MONTH(CURRENT_DATE);
```

NULL

- To make a column NULL , leave the column name when using INSERT statement.
- Column is NULL by default except declare as NOT NULL.

```
select e_id, children
from employee_per
where children IS NOT NULL;
```