

# Fundamental Concepts of Database



# Contents(1)

- What is Data?
  - a collection of facts about something.
- Let's take an example. For the DVD collection, what information would we hold about each DVD?
  - Title, Year, Director, Runtime, Principal actors etc.
- In electronic machine, data is represented in the form of text , numbers or media files.

# Contents(2)

## ■ What is Database?

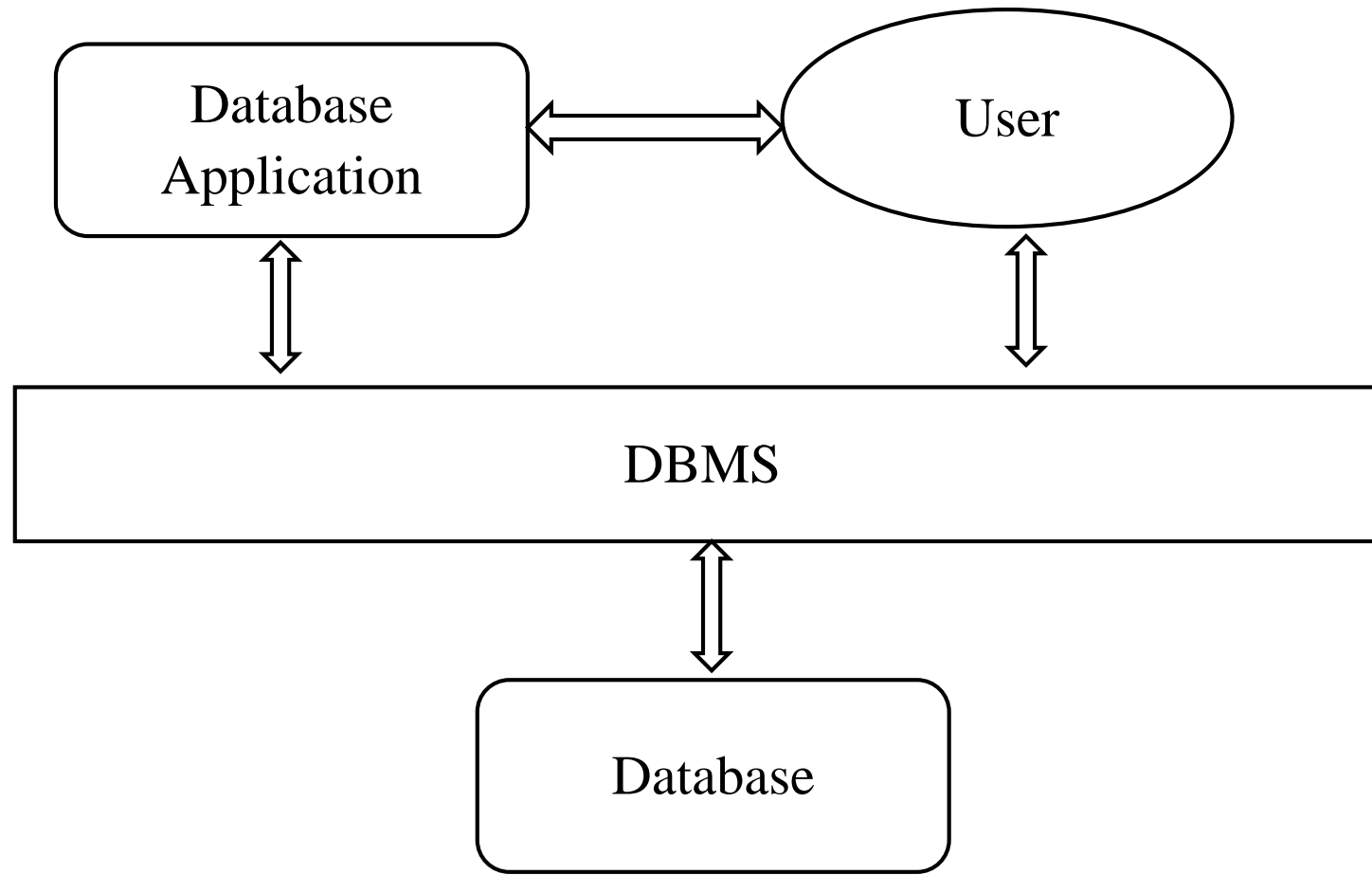
- A collection of interrelated data organized to meet users' needs.
- Is also a repository of data that is defined once and then is accessed by various users.
- Contains information about a particular enterprise.

## ■ Database Examples

- Sales – customers, products, sale-records
- Banking – transactions, account
- Hotel reservation – room, customers, services



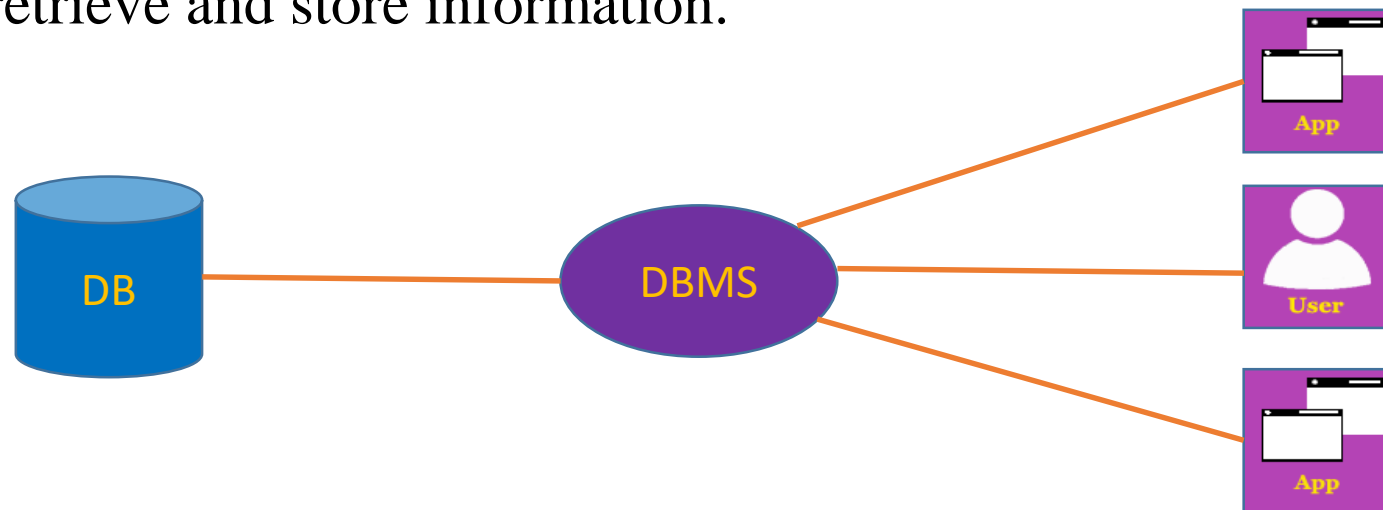
# Components of Database



## Contents(3)

### ■ Database Management System(DBMS)

- Is a group of programs that manipulate the database and provide an interface between the database and the user of the database or other application programs.
- An environment that is both convenient and efficient for users to retrieve and store information.



# Database Keys



# Types of Database Key

- Primary Key
- Secondary Key
- Foreign Key
- Simple Key
- Compound Key

# 1. Primary Key

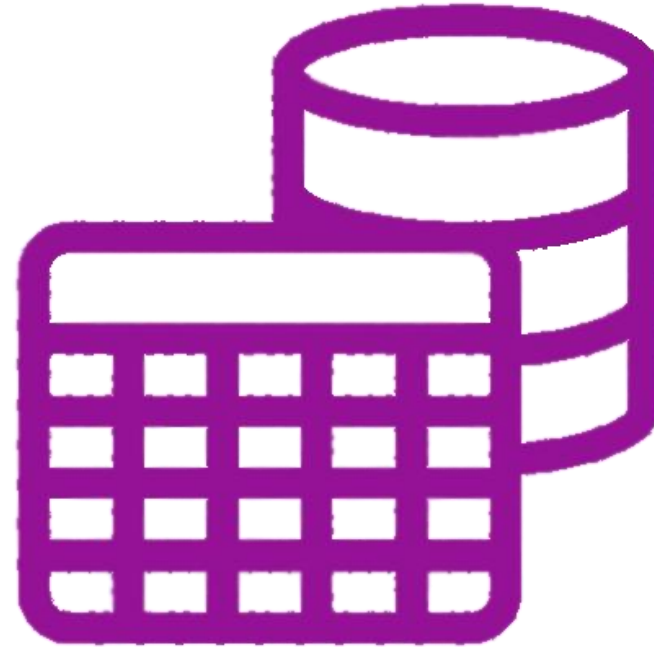
- A column or group of columns uniquely identifies every record in the table.
- A primary key is mandatory and it cannot be null.
- For example, student ID is a primary key as this uniquely identifies within the student records system.



## 2. Secondary Key

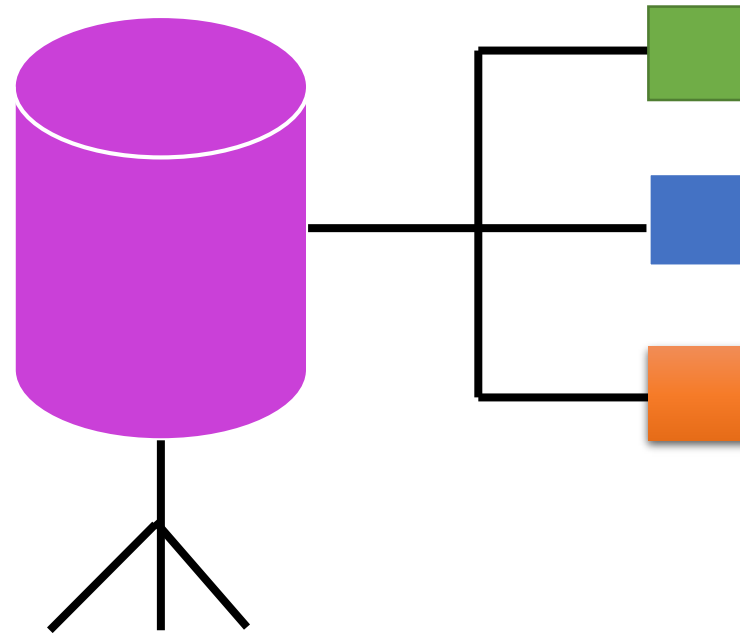
- An entity may have one or more choices for the primary key. These are also known as candidate keys.
- One is selected as the primary key. Those not selected are known as secondary keys.
- For example Stud ID, Roll No, and email are candidate keys which help to uniquely identify the student record.

# Relational Data Model



# Contents(1)

- What is Relational Data Model (RDM)
  - It represents the database as a collection of relations, i.e., tables.
  - Eg. Oracle, MSSQL Server, MySQL etc.
- Components of RDM
  1. Table
  2. Columns
  3. Row
  4. Primary key
  5. Relationship



## Contents(2)

- What is MySQL?
- MySQL Data Type
  1. Numeric
  2. Text
  3. Date/time etc.



# String Data Type

Data Type Syntax	Maximum Size	Description
CHAR(size)	255 characters	<b>size</b> is the number of characters. Fixed-length strings.
VARCHAR(size)	255 characters	<b>size</b> is the number of characters. Variable-length strings.
TEXT(size)	65,535 characters	
BINARY(size)	255 bytes	<b>size</b> is the number of binary characters. Fixed-length strings.
VARBINARY(size)	255 bytes	<b>size</b> is the number of binary characters. Variable-length strings.
BLOB	65,535 bytes	

# Integer Data Type

Data Type Syntax	Description
SMALLINT	<ul style="list-style-type: none"><li>▪ Signed values range is from -32768 to 32767.</li><li>▪ Unsigned values range is from 0 to 65535.</li></ul>
INT	<ul style="list-style-type: none"><li>▪ Standard integer value.</li><li>▪ Signed values range from -2147483648 to 2147483647.</li><li>▪ Unsigned values range from 0 to 4294967295.</li></ul>
BOOL	<ul style="list-style-type: none"><li>▪ Treated as a Boolean data type</li><li>▪ Zero is considered as false, nonzero values are considered as true.</li></ul>

# Floating Data Type

Data Type Syntax	Description
FLOAT(m,d)	<p><b>m</b> is total digits</p> <p><b>d</b> is number of digit after the decimal</p> <p>FLOAT is accurate to approximately 7 decimal places</p>
DOUBLE(m,d)	<p>Double is accurate to approximately 14 decimal places</p>
DECIMAL	<p>DECIMAL can store 30 digits after decimal point.</p> <p>For Business Oriented Math, always use Decimal.</p>

# Date Data Type

<b>Data Type Syntax</b>	<b>Format</b>
DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD hh:mm:ss
TIME	hh:mm:ss
YEAR	YYYY



# Structure Query Language (SQL)



# SQL

- What is SQL?
- Every software developer should have a really good grasp of SQL knowledge(DDL and DML).
- **DDL**: It covers items such as CREATE TABLE and ALTER TABLE.
- **DML**: It covers items such as Select, Update, Insert and Delete.
- You should also understand all the major clauses such as **WHERE, GROUP BY, HAVING, and ORDER BY.**
- In addition you should be comfortable with sub queries and joins.

# Data Definition Language - DDL

- What is DDL?
- Four main DDL commands
  1. **CREATE** – to create objects in the database.
  2. **DROP** – to remove objects from the database.
  3. **ALTER** – to modify the structure of database objects.
  4. **RENAME** – to change database object names.



# DDL – CREATE Command

- “CREATE” command is used to create new objects such as database, table etc.
- Creating Database

## *Syntax:*

```
CREATE DATABASE [IF NOT EXIST] db_name;
```

- Creating Table

## *General Syntax:*

```
CREATE TABLE [IF NOT EXISTS] tblname (col-name dataType);
```

# Database Constraints

- Before studying DDL & DML statements, you should learn database keys and constraints.
- Database Constraints are rules and restrictions applied on the columns of the table to meet data integrity.
- Types of Constraints:

1. Data Type
2. Nullability
3. Unique Key
4. Primary Key

5. Foreign Key
6. Default
7. Checked

# 1. Nullability Constraint

- It defines the column value accepts empty value or not.

## *General Syntax:*

```
CREATE TABLE table-name  
(  
  Column-name data-type NOT NULL,  
  Column-name datatype NULL,  
  ...  
);
```

## 2. UNIQUE Constraint

- It ensures the column will have unique value for each row.

### *General Syntax -1:*

```
CREATE TABLE table_name  
(  
    col-name data_type UNIQUE,  
);
```

### *General Syntax - 2:*

```
CREATE TABLE table_name(  
    col-name data_type,  
    ...  
    UNIQUE(column_name)  
);
```

### 3. PRIMARY KEY Constraint

- It forces the column to have unique value.
- This constraint is another type of UNIQUE constraint.

#### *General Syntax -1:*

```
CREATE TABLE table_name  
(  
col-name data_type PRIMARY KEY,  
);
```

#### *General Syntax - 2:*

```
CREATE TABLE table_name(  
col-name data_type,  
...  
PRIMARY KEY(column_name)  
);
```



# DML Contents

- DML
- Storing New Data
- Modifying Existing Data
- Removing Unused Data
- Data Retrieval



# Data Manipulation Language - DML

- What is DML?
- Four main DML commands
  1. **INSERT** to insert data into a table
  2. **UPDATE** to update existing data within a table
  3. **DELETE** to remove existing data from a table
  4. **SELECT** to retrieve data from a table

# Storing New Data

- **INSERT INTO** command is used to insert new records into the table.
- **Syntax 1:**

```
INSERT INTO table_name VALUES (value1,value2,value3,..., value_n);
```

- **Syntax 2:**

```
INSERT INTO table_name (col1,col2,col3,...) VALUES (val1, val2, val3,  
...);
```

# Example

User table				
id	name	email	township	city
1	Mg Mg	<a href="mailto:mgmng@gmail.com">mgmng@gmail.com</a>	Bahan	Yangon
2	Aung Aung	<a href="mailto:aung@gmail.com">aung@gmail.com</a>	Hlaing	Yangon
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay

```
INSERT INTO users (id,name,email,township,city)
```

```
VALUES (4, 'Yuri', 'yuri@gmail.com', 'Chan Aye Thar San', 'Mandalay')
```

Result				
id	name	email	township	city
1	Mg Mg	<a href="mailto:mgmng@gmail.com">mgmng@gmail.com</a>	Bahan	Yangon
2	Aung Aung	<a href="mailto:aung@gmail.com">aung@gmail.com</a>	Hlaing	Yangon
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay
4	Yuri	<a href="mailto:yuri@gmail.com">yuri@gmail.com</a>	Chan Aye Thar San	Mandalay

# Modifying Existing Data

- **UPDATE** command is used to update any record of data in a table.
- One or more fields can be updated together.
- **General Syntax:**

```
UPDATE table_name SET field1=new-value1, field2=new-value2  
[WHERE Clause]
```

## Example - 1

employee table			
id	name	age	salary
1	Yuki	22	3000
2	Mr. Jeon	23	5000
3	Kyaw Kyaw	20	2600

```
UPDATE employee SET salary = salary + (salary * 0.2) ;
```

Result			
id	name	age	salary
1	Yuki	22	3600
2	Mr. Jeon	23	6000
3	Kyaw Kyaw	20	3120

## Example - 2

employee table			
id	name	age	salary
1	Yuki	22	3000
2	Mr. Jeon	23	5000
3	Kyaw Kyaw	20	2600

```
UPDATE employee SET name = 'JK' WHERE id= 3 ;
```

Result			
id	name	age	salary
1	Yuki	22	3000
2	Mr. Jeon	23	5000
3	JK	20	2600

## Example - 3

employee table			
id	name	age	salary
1	Yuki	22	3000
2	Mr. Jeon	23	5000
3	Kyaw Kyaw	20	2600

```
UPDATE employee SET name = 'JK', salary = 6000 WHERE id= 3 ;
```

Result			
id	name	age	salary
1	Yuki	22	3000
2	Mr. Jeon	23	5000
3	JK	20	6000



# Removing Unused Data

- **DELETE** command is used to delete data from a table.
- **Syntax:**

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

- **Drop** command is used to permanently delete database objects. It is rarely used.
- **Syntax:**

```
DROP TABLE table_name;
```

# Example

User table				
id	name	email	township	city
1	Mg Mg	<a href="mailto:mgmng@gmail.com">mgmng@gmail.com</a>	Bahan	Yangon
2	Aung Aung	<a href="mailto:aung@gmail.com">aung@gmail.com</a>	Hlaing	Yangon
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay
4	Yuri	<a href="mailto:yuri@gmail.com">yuri@gmail.com</a>	Chan Aye Thar San	Mandalay

```
DELETE FROM users WHERE city = 'Yangon';
```

Result				
id	name	email	township	city
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay
4	Yuri	<a href="mailto:yuri@gmail.com">yuri@gmail.com</a>	Chan Aye Thar San	Mandalay

# Data Retrieval

- **SELECT** command is used to retrieve data from database.
- *Syntax*

```
SELECT column1, column2,... FROM table_name;
```

```
SELECT * FROM table_name;
```

```
SELECT * FROM table_name LIMIT number;
```

## Example - 1

User table				
id	name	email	township	city
1	Mg Mg	<a href="mailto:mgmng@gmail.com">mgmng@gmail.com</a>	Bahan	Yangon
2	Aung Aung	<a href="mailto:aung@gmail.com">aung@gmail.com</a>	Hlaing	Yangon
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay
4	Yuri	<a href="mailto:yuri@gmail.com">yuri@gmail.com</a>	Chan Aye Thar San	Mandalay
5	Jeon	<a href="mailto:jeon@gmail.com">jeon@gmail.com</a>	Pale	Monywa

```
SELECT name, city FROM users;
```

Result	
name	city
Mg Mg	Yangon
Aung Aung	Yangon
Kyaw Kyaw	Mandalay
Yuri	Mandalay
Jeon	Monywa

## Example - 2

User table				
id	name	email	township	city
1	Mg Mg	<a href="mailto:mgmng@gmail.com">mgmng@gmail.com</a>	Bahan	Yangon
2	Aung Aung	<a href="mailto:aung@gmail.com">aung@gmail.com</a>	Hlaing	Yangon
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay
4	Yuri	<a href="mailto:yuri@gmail.com">yuri@gmail.com</a>	Chan Aye Thar San	Mandalay
5	Jeon	<a href="mailto:jeon@gmail.com">jeon@gmail.com</a>	Pale	Monywa

```
SELECT * FROM users LIMIT 3;
```

Result				
id	name	email	township	city
1	Mg Mg	<a href="mailto:mgmng@gmail.com">mgmng@gmail.com</a>	Bahan	Yangon
2	Aung Aung	<a href="mailto:aung@gmail.com">aung@gmail.com</a>	Hlaing	Yangon
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay

# SELECT DISTINCT statement

- **SELECT DISTINCT** command is used to retrieve unique values from table.
- *Syntax*

```
SELECT DISTINCT column-name FROM table_name;
```

# Example

User table				
id	name	email	township	city
1	Mg Mg	<a href="mailto:mgmng@gmail.com">mgmng@gmail.com</a>	Bahan	Yangon
2	Aung Aung	<a href="mailto:aung@gmail.com">aung@gmail.com</a>	Hlaing	Yangon
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay
4	Yuri	<a href="mailto:yuri@gmail.com">yuri@gmail.com</a>	Chan Aye Thar San	Mandalay
5	Jeon	<a href="mailto:jeon@gmail.com">jeon@gmail.com</a>	Pale	Monywa

```
SELECT DISTINCT city FROM users;
```

Result
city
Yangon
Mandalay
Monywa

# WHERE Clause

- **WHERE** clause is used to retrieve only user wanted data from database.
- *Syntax*

```
SELECT column(s)  
FROM table-name  
WHERE col-name operator value;
```



# Example

User table				
id	name	email	township	city
1	Mg Mg	<a href="mailto:mgmng@gmail.com">mgmng@gmail.com</a>	Bahan	Yangon
2	Aung Aung	<a href="mailto:aung@gmail.com">aung@gmail.com</a>	Hlaing	Yangon
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay
4	Yuri	<a href="mailto:yuri@gmail.com">yuri@gmail.com</a>	Chan Aye Thar San	Mandalay
5	Jeon	<a href="mailto:jeon@gmail.com">jeon@gmail.com</a>	Pale	Monywa

```
SELECT * FROM users WHERE city = 'Mandalay';
```

Result				
id	name	email	township	city
3	Kyaw Kyaw	<a href="mailto:kyaw@gmail.com">kyaw@gmail.com</a>	Mahar Myaing	Mandalay
4	Yuri	<a href="mailto:yuri@gmail.com">yuri@gmail.com</a>	Chan Aye Thar San	Mandalay

# List of Operators

Operator	Meaning
=	Equal
!=	Not equal
<>	
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
BETWEEN val1 IN val2	BETWEEN inclusive range
LIKE	Search for pattern matching
IN	To specify multiple possible values for a column

## Example - 1

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT * FROM employee WHERE salary <= 6000;
```

Result			
id	name	city	salary
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

## Example - 2

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT * FROM employee WHERE salary BETWEEN 9000 AND 6000;
```

Result			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000

## Example - 3

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT * FROM employee WHERE salary IN (1000,4000,7800);
```

Result			
id	name	city	salary
2	Aung Aung	Yangon	7800
5	Jeon	Monywa	4000

# LIKE Operator

- **LIKE** operator is used to perform pattern matching to find the correct result.
- *Syntax :*

*--Starting pattern—*

```
SELECT|UPDATE|DELETE  
statements...  
WHERE fieldname LIKE 'xx%';
```

*--Ending pattern—*

```
SELECT|UPDATE|DELETE statements...  
WHERE fieldname LIKE '%xx ';
```

*--Containing pattern--*

```
SELECT|UPDATE|DELETE statements...  
WHERE fieldname LIKE '%xx %';
```

## Example - 1

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT * FROM employee WHERE name LIKE 'a%';
```

Result			
id	name	city	salary
2	Aung Aung	Yangon	7800

## Example - 2

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT * FROM employee WHERE name LIKE '%g';
```

Result			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800



## Example - 3

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT * FROM employee WHERE name LIKE '%n%';
```

Result			
id	name	city	salary
2	Aung Aung	Yangon	7800
5	Jeon	Monywa	4000

# Logical Operator

- Logical operators are used with SELECT, INSERT, UPDATE or DELETE statements to test two or more conditions in an individual query.
- Three types of logical operators : **AND, OR, NOT**
- **NOT** operator can be used with a comparison operator to negate the result of the comparison.
  - NOT BETWEEN...AND...
  - NOT IN (value1, value2, value3 ...)
  - NOT LIKE

# AND Operator

- **AND Operator** displays a record if all specified conditions are true.
- *Syntax*

```
SELECT column(s)
```

```
FROM table-name
```

```
WHERE condition1 AND condition2 ..... AND condition-n;
```

## Example

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT * FROM employee WHERE city = 'Mandalay' AND salary > 4000
```

Result			
id	name	city	salary
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200

# OR Operator

- **OR Operator** displays a record if at least one condition is true.
- *Syntax*

```
SELECT column(s)
```

```
FROM table-name
```

```
WHERE condition1 OR condition2 ..... OR condition-n;
```

## Example

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT * FROM employee WHERE city = 'Mandalay' OR salary > 4000
```

Result			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200

# Sorting Data

- **ORDER BY** clause is used to sort the query result sets by a specified column in descending or ascending order (default).
- *Syntax*

```
SELECT column(s)  
FROM table-name  
[WHERE conditions]  
ORDER BY column-name [ASC|DESC];
```

# Example

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT * FROM employee ORDER BY salary ASC
```

id	name	city	salary
5	Jeon	Monywa	4000
4	Yuri	Mandalay	4200
3	Kyaw Kyaw	Mandalay	6000
2	Aung Aung	Yangon	7800
1	Mg Mg	Yangon	9000



# Built-in Functions

- MySQL provides many built-in functions to perform operations on data.
- **Aggregate** : return a single value after performing calculations on a group of values. E.g. AVG, COUNT, MAX, MIN, SUM etc.
- **Scalar** : returns a single value from an input value. E.g. UCASE, LCASE, ROUND etc.

# COUNT() Function

- **COUNT** function returns the number of records ( NULL value will not be counted ) of the specified columns.
- *Syntax*

```
SELECT COUNT(columns)  
FROM table-name  
[WHERE conditions];
```

# Example

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT COUNT(*) 'Total' FROM employee WHERE city = 'Yangon';
```

Result
--------

Total
-------

2
---

```
SELECT COUNT(*) 'Total' FROM employee
```

Result
--------

Total
-------

5
---

# MAX(), MIN() Function

- **MAX** function returns the largest value of the selected column.
- *Syntax*

```
SELECT MAX(column-name)  
FROM table-name [WHERE conditions];
```

- MIN function returns the smallest value of the selected column.
- *Syntax*

```
SELECT MIN(column-name)  
FROM table-name [WHERE conditions];
```

# Example

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT MIN(salary) 'Min Salary' FROM employee;
```

```
SELECT MAX(salary) 'Max Salary' FROM employee;
```

```
SELECT MAX(salary) 'Max Salary' FROM employee  
WHERE city = 'Mandalay';
```

Result	Result
Min Salary	Max Salary
4000	9000

Result
Max Salary
6000

# SUM(), AVG() Function

- **SUM** function returns the total sum of selected columns in numeric values.

- *Syntax*

```
SELECT SUM(column-name)  
FROM table-name [WHERE conditions];
```

- **AVG** function returns average value selected columns in numeric values.

- *Syntax*

```
SELECT AVG(column-name)  
FROM table-name [WHERE conditions];
```

# Example

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT SUM(salary) 'Total Salary' FROM employee;
```

Result
--------

Total Salary
--------------

31000
-------

```
SELECT AVG(salary) 'Average Salary' FROM employee;
```

Result
--------

Average Salary
----------------

6200
------

# UCASE(), LCASE() Function

- **UCASE** function used to convert value of string column to uppercase characters.

- *Syntax*

```
SELECT UCASE(column-name)  
FROM table-name [WHERE conditions];
```

- **LCASE** function used to convert value of string column to lowercase characters.

- *Syntax*

```
SELECT LCASE(column-name)  
FROM table-name [WHERE conditions];
```



# Example

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT id, UCASE(name), LCASE(city) FROM employee;
```

id	name	city
1	MG MG	yangon
2	AUNG AUNG	yangon
3	KYAW KYAW	mandalay
4	YURI	mandalay
5	JEON	monywa

# GROUP BY Clause

- **GROUP BY** clause is used to group the results of a SELECT query.
- It returns one row for each group.
- It is used with aggregate functions such as SUM, AVG, MAX, MIN, and COUNT.
- *Syntax*

```
SELECT col-name, aggregate_function(ci)
FROM table-names
[WHERE clause]
GROUP BY col-name;
```

## Example - 1

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT city, MAX(salary) 'Maximum Salary' FROM employee GROUP BY city;
```

Result	
city	Maximum Salary
Yangon	9000
Mandalay	6000
Monywa	4000

## Example - 2

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT city, COUNT(*) 'Total Employee' FROM employee GROUP BY city;
```

Result	
city	Total Employee
Yangon	2
Mandalay	2
Monywa	1

# HAVING Clause

- **HAVING** clause is used to restrict the results returned by the GROUP BY clause.
- It is used together with aggregate functions.
- HAVING clause must be placed immediately after GROUP BY clause.
- *Syntax*

```
SELECT column-name, aggregate_function (ci)
FROM table-name
[WHERE conditions]
GROUP BY column-name
HAVING aggregate_function (ci) operator value;
```

## Example

employee table			
id	name	city	salary
1	Mg Mg	Yangon	9000
2	Aung Aung	Yangon	7800
3	Kyaw Kyaw	Mandalay	6000
4	Yuri	Mandalay	4200
5	Jeon	Monywa	4000

```
SELECT city, COUNT(*) 'Total Employee' FROM employee  
GROUP BY city HAVING COUNT(*) > 1;
```

Result	
city	Total Employee
Yangon	2
Mandalay	2



# Q & A