Introduction To Database

1. Database is an organized collection of Data. Here organized means, the Data should be arranged in a particular order. (i.e., It must be arranged).
2. If we have collection of Data, then only on the basis of collection of Data we can't say that is a database. We need to organize this Data in a particular order.
3. Dictionary, Telephone Directory, Attendance Register, etc. are the examples of Database in which all the Words/Numbers/Names are stored in alphabetic order.

Database Management System (DBMS)

1. It is a software to perform database activity. **OR**
2. It is an environment to perform database functionality. **OR**
3. DBMS is a way to achieve the database concept using various IDE (Integrated Development Environment).

Advantages of Database

1. Since Database is an organized collection of Data, Data retrieval (Searching Data), Updation, Deletion is very easier & Fast.
2. We can easily perform calculations on Data.
3. Data Security.

Types of Database

There are following types of Databases :

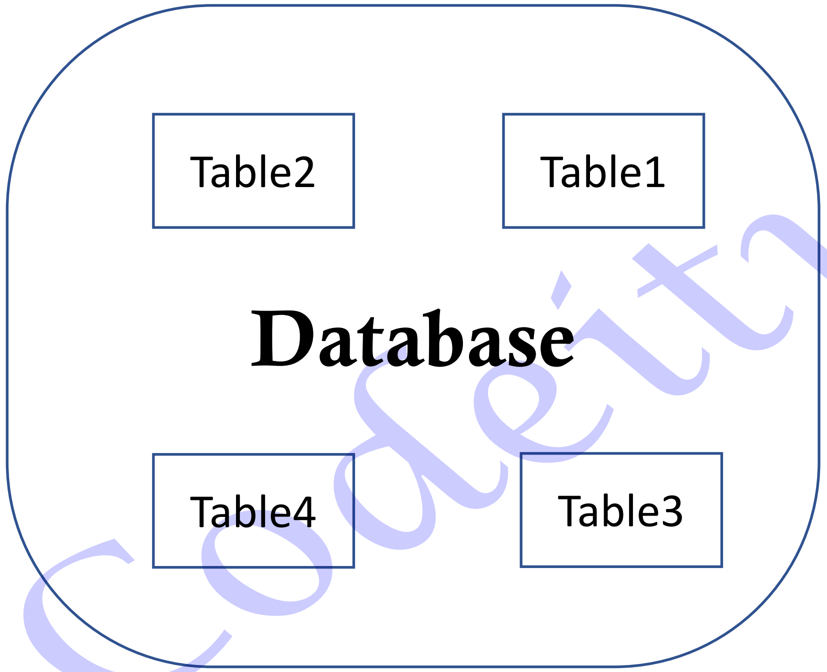
1. Relational Database Management System **(RDBMS).**
   - ➢ **SQL** is a type of RDBMS in which data is stored in rows and columns. i.e., in the form of table or tabular form.
2. Hierarchical Database Management System **(HDBMS).**
3. Network Database Management System **(NDBMS).**
4. Object Oriented Database Management System **(ODBMS).**

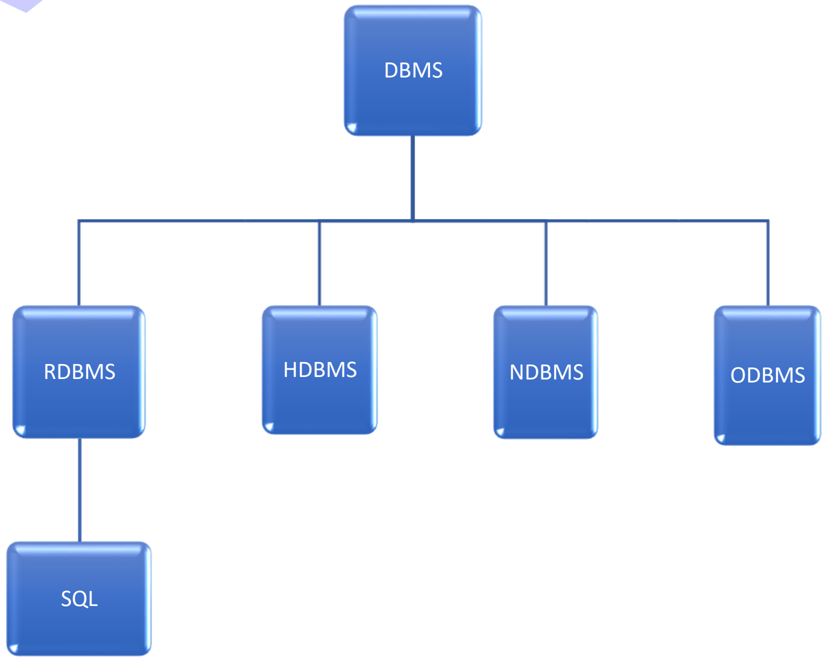
⇨ We will go for **SQL** which is a type of **RDBMS.**

To Create Database we have a lots of Database Management System **(DBMS).**

Note : Database is a collection of Tables

Examples : MySQL, SQL+, MS-Access, MongoDB, etc,.





SQL (Structured Query Language)

SQL Stands for **Structured Query Language**. It is used for creating Database. SQL adopts the concept of RDBMS (Relational Database Management System). Relational Database is a type of Database in which Data is stored in the form of Rows & Columns. i.e., in a table or tabular form.

Example :

Structured Query Language (SQL)

Database → DBMS → RDMS → SQL

Relational Database Management System :

In RDBMS, data is stored in rows & columns i.e., in a table.

Columns **OR** Field **OR** Attributes

Table Name : Students

Rows **OR** Tuples

Roll No.	Name	Gender	Address	Mobile No.
101	Amit	Male	Delhi	9897XXXXXX
102	Shivani	Female	Agra	8126XXXXXX
103	Sanjay	Male	Mathura	9639XXXXXX

Terms In RDBMS

1.	Table	➔	Relation.
2.	Field/Column	➔	Attribute.

3. Row/Tuple (Horizontal set of Information)
4. Columns/Field (Vertical set of Information)

5.	Cardinality	➔	No. of Rows
6.	Degree	➔	No. of Columns
7.	Domain	➔	Pool of Values

Example :

Cardinality	➔	3
Degree	➔	5

Roll_No.	**Name**	**Gender**	**Address**	**Mobile_No**
101	Amit	Male	Delhi	9897XXXXXX
102	Shivani	Female	Agra	8126XXXXXX
103	Sanjay	Male	Mathura	9639XXXXXX

SQL Commands

SQL is a collection of commands. For every task we have a defined command which we have to use.

All the commands are divided into 3 Parts

1. **DDL** Commands (Data Definition Language)
2. **DML** Commands (Data Manipulation Language)
3. **DCL/TCL** Commands (Data/Transaction Control Language)

DDL Commands (Data Definition Language)

DDL are the collection of all those commands which are used to create Database structure and Related activities.

Examples : CREATE TABLE, ALTER TABLE, etc.

DML Commands (Data manipulation Language)

DML are the collection of all those commands which are used to manipulate Data in the Database.

Examples : SELECT, INSERT, UPDATE, DELETE, etc.

TCL/DCL Commands (Transaction/Data Control Language)

TCL/DCL are the collection of those commands which are used to control the Database transaction.

Examples : GRANT, REVOKE, etc.

Data Types In SQL

1. **char**	➔	Store Character Value has fixed length
2. **varchar**	➔	Store Character Value has variable length
3. **int or integer**	➔	Stores Integer Value
4. **decimal**	➔	Stores Decimal Value.
5. **date**	➔	Stores Date.
6. **time**	➔	Stores Time.

Note : varchar is more suitable than char.

The Value of varchar & char Data Type are used in single quotes.

Difference Between char & varchar

char ➔ Fixed length

Example : Name char (15)

Fill with white spaces

R	A	M	_	_	_	_	_	_	_	_	_	_	_	_

varchar ➔ Variable length

Example : Name varchar (15)

Release (Not Occupy)

R	A	M												

Constraints In SQL

Constraints are set of rules that are applied to a Table/Attribute/Field.

There are following types of Constraints :

1. Primary Key
2. Unique Key
3. NOT NULL
4. Default
5. Check
6. Foreign Key

Note : Candidate Key & Alternate Key These are not Constraints.

Candidate Key : Those who might be considered as primary key.

Alternate Key : A candidate key that is not primary key is called alternate key.

1. <u>**Primary Key**</u>

Primary key is one which is unique and using this, the data is uniquely identified. Primary Key constraints implies that the attribute can not be duplicate value as well as it can not be left blank. There are be only one Primary Key in one table.

Composite Key : When one single column/field is unable to uniquely identify a record, then two or more columns/fields are joined together to form primary key but this time it is called Composite Key.

Primary Key = No duplicate values + It can’t be left blank.

OR

Primary Key = Unique Key + NOT NULL

Table Name :
Students

Primary Key

Can not be blank

Duplicate Value

Adm_No	Name	Gender	Address	Mobile_No
101	Amit	Male	Delhi	9897XXXXXX
102	Shivani	Female	Agra	8126XXXXXX
✘	Sanjay	Male	Mathura	9639XXXXXX
101 ✘	Anjali	Female	Lucknow	7017XXXXXX
103	Tarun	Male	Kanpur	8859XXXXXX

Here **Adm_No** is Primary Key and hence it can’t contain duplicate values as well as it can’t be left blank.

Table Name : Students

Primary Key

Adm_No	Name	Gender	Address	Mobile_No
101	Amit	Male	Delhi	9897XXXXXX
102	Shivani	Female	Agra	8126XXXXXX
103	Sanjay	Male	Mathura	9639XXXXXX
104	Anjali	Female	Lucknow	7017XXXXXX
105	Tarun	Male	Kanpur	8859XXXXXX

Here **Adm_No** is Primary Key and hence it can't contain duplicate values as well as it can't be left blank.

2. Unique Key

Unique Key implies that an attribute can’t contain duplicate values.

Primary Key

Roll_No	Name	Gender	Address	Mobile_No	
101	Amit	Male	Delhi	9897XXXXXX	
102	Shivani	Female	Agra	7017701770	
103	Sanjay	Male	Mathura		✓
104	Anjali	Female	Lucknow	7017701770	✘
105	Tarun	Male	Kanpur	8859XXXXXX	

Unique Key

Duplicate Value

Here **Mobile_No** is Unique Key and hence it can’t contain duplicate but you can leave this blank.

2. Unique Key

Unique Key implies that an attribute can't contain duplicate values.

Roll_No	Name	Gender	Address	Mobile_No
101	Amit	Male	Delhi	9897XXXXXX
102	Shivani	Female	Agra	7017701770
103	Sanjay	Male	Mathura	
104	Anjali	Female	Lucknow	8859885988
105	Tarun	Male	Kanpur	8859XXXXXX

Primary Key

Unique Key

✓

Left Blank

Here **Mobile_No** is Unique Key and hence it can't contain duplicate but you can leave this blank.

3. NOT NULL

This Implies that attribute can't be left blank.

Primary Key

NOT NULL

Unique Key

Roll_No	Name	Gender	Salary	Mobile_No
101	Amit	Male	50000	9897XXXXXX
102	Shivani	Female	60000	7017XXXXXX
103	Pawan	Male	55000	8888888888
104		Female	65000	7017701770

✘

Can not be blank

Here **Name** is NOT NULL and hence it can't be left blank. But it can contain duplicate values.

3. NOT NULL

This Implies that attribute can't be left blank.

Primary Key

NOT NULL

Unique Key

Roll_No	Name	Gender	Salary	Mobile_No
101	Amit	Male	50000	9897XXXXXX
102	Shivani	Female	60000	7017XXXXXX
103	Pawan	Male	55000	8888888888
104	Pawan	Female	65000	7017701770

✓

Duplicate Value

Here **Name** is NOT NULL and hence it can't be left blank. But it can contain duplicate values.

4. Default

This implies that when any attribute is left blank, then it cab be provided some default value which will be automatically inserted when no values will be passed for that column.

Roll_No	Name	Salary	Mobile_No	Gender
101	Amit	50000	9897XXXXXX	M
102	Shivani	60000	7017XXXXXX	F
103	Sanjay	55000	8879XXXXXX	
104	Anjali	52000	9368XXXXXX	F

Default 'M'

Taken Default Value 'M'

Here, Gender has a Default value 'M' then if no values will be passed, it will automatically be filled with 'M'.

5. Check

This constraints check for a particular condition and if that condition is TRUE, then only the value is inserted to the table otherwise discarded.

Roll No.	Name	Mobile No.	Gender	Salary
101	Amit	9897XXXXXX	M	50000
102	Shivani	7017XXXXXX	F	60000
103	Sanjay	8879XXXXXX	M	29000
104	Anjali	9368XXXXXX	F	52000

Check
[Salary > 45000]

Error

Here, If Salary has a check constraints and it is checking that the salary must not be less than 45,000, then if anyone will insert any value less than 45,000, it will not accept that value and an error will be displayed.

6. Foreign Key

When a filed in one table is playing the role of Primary Key & the same field is a non-primary key in the another table, then the non-primary key in the another table is called Foreign Key.

Minimum 2 Tables is required for Foreign Key (1st is Parent Table & 2nd is Child Table)

For this table **DepCode** is not primary key

Primary Key

AdmNo	Name	DepCode
101	Amit	1001
102	Ajay	1002
103	Aman	1003

For this table **DepCode** is primary key

DepCode	Name	HOD
1001	Admin	XYZ
1002	HR	MNO
1003	Finance	ABC

In this table, the primary key is Code in this table, DepCode is Primary Key. So, Here **DepCode** in Admission Table is called **Foreign Key.**

Keyword to use foreign key is references **OR** Foreign key create a term **Referential Integrity.**

MySQL Commands

1. Create Database
2. Show Database
3. Use
4. Show Tables
5. Create Table
6. Insert Into
7. Alter
8. Update
9. Desc
10. Select
11. Delete
12. Drop

1. CREATE DATABASE

This command is used to create a database.

Syntax :

```
MySQL> CREATE DATABASE DatabaseName;
```

Example :

```
MySQL> CREATE DATABASE students;
```

2. SHOW DATABASE

This command is used to display the list of all the database.

Syntax :

```
MySQL> SHOW DATABASES;
```

3. USE DATABASE

This command is used to open a database.

Syntax :

```
MySQL> USE DatabaseName;
```

Example :

```
MySQL> USE students;
```

4. SHOW TABLES

This command is used to show the table inside a database.

Syntax :

```
MySQL> SHOW TABLES;
```

Remember : Before Create Table you know about the table Structure.

i.e., what you want to type of data & size of field in table like.

TableName : students

Adm_No	Name	Mobile_No	Address	DOB	Fees	Gender
Integer	varchar(30)	char(10)	varchar(100)	date	decimal	char

Note : If you does not use size of particular column then it takes default range.

Note :

- ✓ Decimal & Integer can not write in single quotes.
- ✓ Char, Varchar & Date write in single quotes.
- ✓ Write Date always in the format of YYYY-MM-DD.

5. CREATE TABLE

This command is used to create a table.

Syntax :

```
MySQL> CREATE TABLE Table_Name (
    -> Col1        Datatype (Size)      constraint,
    -> Col2        Datatype (Size)      constraint,
    -> Col3        Datatype (Size)      constraint,
       _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
       _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
    -> );
```

Optional

Example :

```
MySQL> CREATE TABLE students (
        -> Adm_No        int                 primary key,
        -> Name          varchar (30)        NOT NULL,
        -> Mobile_No     char (10),
        -> Address       varchar (100)       NOT NULL,
        -> DOB           date,
        -> Fees          decimal             check (Fees > 500),
        -> Gender        char                default 'M'
        -> );
```

Default size is 1.

TableName : **students**

Adm_No	Name	Mobile_No	Address	DOB	Fees	Gender

6. INSERT INTO

This command and is used to insert values in the table. We can insert values in the table using 2 ways.

Syntax 1: For Every Column Value.

```
MySQL> INSERT INTO TableName values (
          -> Val1, Val2, Val3, ……, Valn );
```

Example :

```
MySQL> INSERT INTO students values (
          -> 101, 'Ram', 9897989798, 'Delhi', '2002-02-02', 600, 'M');
```

Adm_No	Name	Mobile_No	Address	DOB	Fees	Gender
101	Ram	9897989798	Delhi	2002-02-02	600	M

Syntax 2: For particular values of Column

```
MySQL> INSERT INTO TableName (Field1, Field2, …., Fieldn) values
              (Val1, Val2, ……, Valn);
```

Example :

```
MySQL> INSERT INTO students (Adm_No, Name, Adress, Fees, Gender)
            values (102, ‘Amit’, ‘Agra’, 900, ‘M’);
```

Adm_No	Name	Mobile_No	Address	DOB	Fees	Gender
101	Ram	9897989798	Delhi	2002-02-02	600	M
102	Amit		Agra		900	M

Example :

```
MySQL> INSERT INTO students values (
        -> 101, ‘Ram’, 9897989798, ‘Delhi’, ‘2002-02-02’, 600, ‘M’,
        -> 102, ‘Amit’, 8126681266, ‘Agra’, ‘2004-04-20’, 900, ‘M’,
        -> 103, ‘Sanju’, 8859885988, ‘Mathura’, ‘2003-07-12’, 550, ‘M’,
        -> 104, ‘Satyam’, 9368936893, ‘Delhi’, ‘2005-01-13’, 700, ‘M’,
        -> 105, ‘Pankaj’, 6395639563, ‘Agra’, ‘2007-06-15’, 850, ‘M’,
        -> 106, ‘Versha’, 7017701770, ‘Mathura’, ‘2001-05-29’, 530, ‘F’,
        -> 107, ‘Shivani’, 7907907907, ‘kanpur’, ‘2003-03-27’, 950, ‘F’,
        -> 108, ‘Anjali’, 9557955795, ‘Bihar’, ‘2009-08-11’, 580, ‘F’,
        -> );
```

TableName : **students**

Adm_No	Name	Mobile_No	Address	DOB	Fees	Gender
101	Ram	9897989798	Delhi	2002-02-02	600	M
102	Amit	8126681266	Agra	2004-04-20	900	M
103	Sanju	8859885988	Mathura	2003-07-12	550	M
104	Satyam	9368936893	Delhi	2005-01-13	700	M
105	Pankaj	6395639563	Agra	2007-06-15	850	M
106	Versha	7017701770	Mathura	2001-05-29	530	F
107	Shivani	7907907907	Kanpur	2003-03-27	950	F
108	Anjali	9557955795	Bihar	2009-08-11	580	F

7. Alter

This command is used to make change in table structure.

1. To add a column Using **ADD** Keyword.
2. To delete a column Using **DROP** Keyword.
3. To rename a column Using **RENAME** Keyword.
4. To Modifying The Size or Datatype Using **MODIFY** Keyword.

ADD Column

Syntax :

```
MySQL> ALTER TABLE Table_Name ADD FieldName datatype (size);
```

Example :

```
MySQL> ALTER TABLE students ADD Address varchar (50);
```

DROP Column

Syntax :

```
MySQL> ALTER TABLE Table_Name DROP FieldName;
```

Example :

```
MySQL> ALTER TABLE students DROP DOB;
```

RENAME Column

Syntax :

```
MySQL> ALTER TABLE TableName RENAME Column OldName to NewName;
```

Example :

```
MySQL> ALTER TABLE students RENAME Column DOB to Date_Of_Birth;
```

MODIFY Column

Syntax :

```
MySQL> ALTER TABLE TableName MODIFY FieldName Datatype (size);
```

Example :

```
MySQL> ALTER TABLE students MODIFY Address varchar (50);
```

8. Update

This command is used to update the pre-existing data in the table.

Important

Syntax 1 : For Single Field

```
MySQL> UPDATE TableName set FieldName = Value where condition;
```

Example :

```
MySQL> UPDATE students set Address = 'Gurgaon' where Name='Amit';
```

If you does not used where then all the field fill be that value

Use Primary Key Otherwise value fill with same type of values.

Adm_No	Name	Mobile_No	Address	DOB	Fees	Gender
101	Ram	9897989798	Delhi	2002-02-02	600	M
102	Amit		Gurgaon		900	M

Syntax 2 : For Multiple Field

MySQL> UPDATE TableName set Field1 = Value, field2 = Value, where condition;

Example :

SQL > Update students set Name='Rahul', Fees=1000 where Adm_No=102;

Adm_No	Name	Mobile_No	Address	DOB	Fees	Gender
101	Ram	9897989798	Delhi	2002-02-02	600	M
102	Rahul		Gurgaon		1000	M

9. <u>Desc</u>

This command is used to show the structure of the Table.

Syntax : MySQL> DESC Table_Name;

Example : MySQL> DESC students;

10. Select

This command is used to extract data from the table.

It shows all the data of the table.

Syntax : MySQL> Select $\circ$* from Table_Name;

Example : MySQL> Select * from students;

Select Particular Fields From The Table

Syntax :

```
MySQL> Select col1, col2, col3, … From Table_Name;
```

Example :

```
MySQL> Select Adm_No, Name, Gender from students;
```

Condition Based Searching

1. Where Clause : Where Clause is used to extract data from the table with a given condition.

Syntax : MySQL> Select * From TableName Where Condition;

Example : MySQL> Select * From students Where Fees=550;

2. AND / OR : There may be some conditions where we need to continue two or more conditions together. To combine the conditions we use either AND **or** OR.

AND : The conditions attached with AND will give the result TRUE only when all the conditions are satisfied.

OR : Says that any one condition should be true and the result will be TRUE.

3. Distinct Clause : This Clause removes duplicate values from the table.

Syntax : MySQL> Select Distinct (Field) From TableName where condition;

Example : MySQL> Select Distict (Address) From students;

Optional

4. Order By Clause : This clause is used to display the data in either ascending order or in descending order. By default is sorts the data in ascending order for sorting in descending order you may use **“Desc”** at the end of the statement.

Syntax : MySQL> Select * From TableName Order By FieldName;

Example : MySQL> Select * From students Order By Name;

Ascending Order

Syntax : MySQL> Select * From TableName Order By FieldName Desc;

Example : MySQL> Select * From students Order By Name Desc;

Descending Order

5. Between : This clause lets us the facility to give a range of values.

Here Including Both Values

Syntax :

```
MySQL> Select * From TableName Where Condition Between Value1 And Value2;
```

Example :

```
MySQL> Select * From students Where Fees Between 800 And 1000;
```

6. Membership Operator (In/Not In) : This clause lets us the facility to look for a set of values.

Syntax :

```
MySQL> Select * From TableName Where Column In (Value1, Value2, …);
```

Example :

```
MySQL> Select * From students Where Address In (‘Agra’, ‘Mathura’);
```

7. Pattern Matching (Like/Not Like) : Like clause is used for pattern matching. In addition % and _ and helps in SQL for pattern matching in SQL. Pattern matching refers to the extraction of data based on string matching concepts.

% is used for multiple characters.

_ (underscore sign is used for representing single character. Once underscore will represent one character, two underscore will represent two character and so on..

Example :

MySQL> Select * From students Where Name Like 'A%';

MySQL> Select * From students Where Name Like 'Am%';

MySQL> Select * From students Where Name Like 'A_ _ _';

MySQL> Select * From students Where Address Like '%Colony%';

11. Delete

This command is used to Delete the data from the Table.

Syntax : MySQL> Delete From Table_Name Where Condition;

Example : MySQL> Delete From students Where gender='F';

Syntax : MySQL> Delete From Table_Name; //Delete All Record Of table

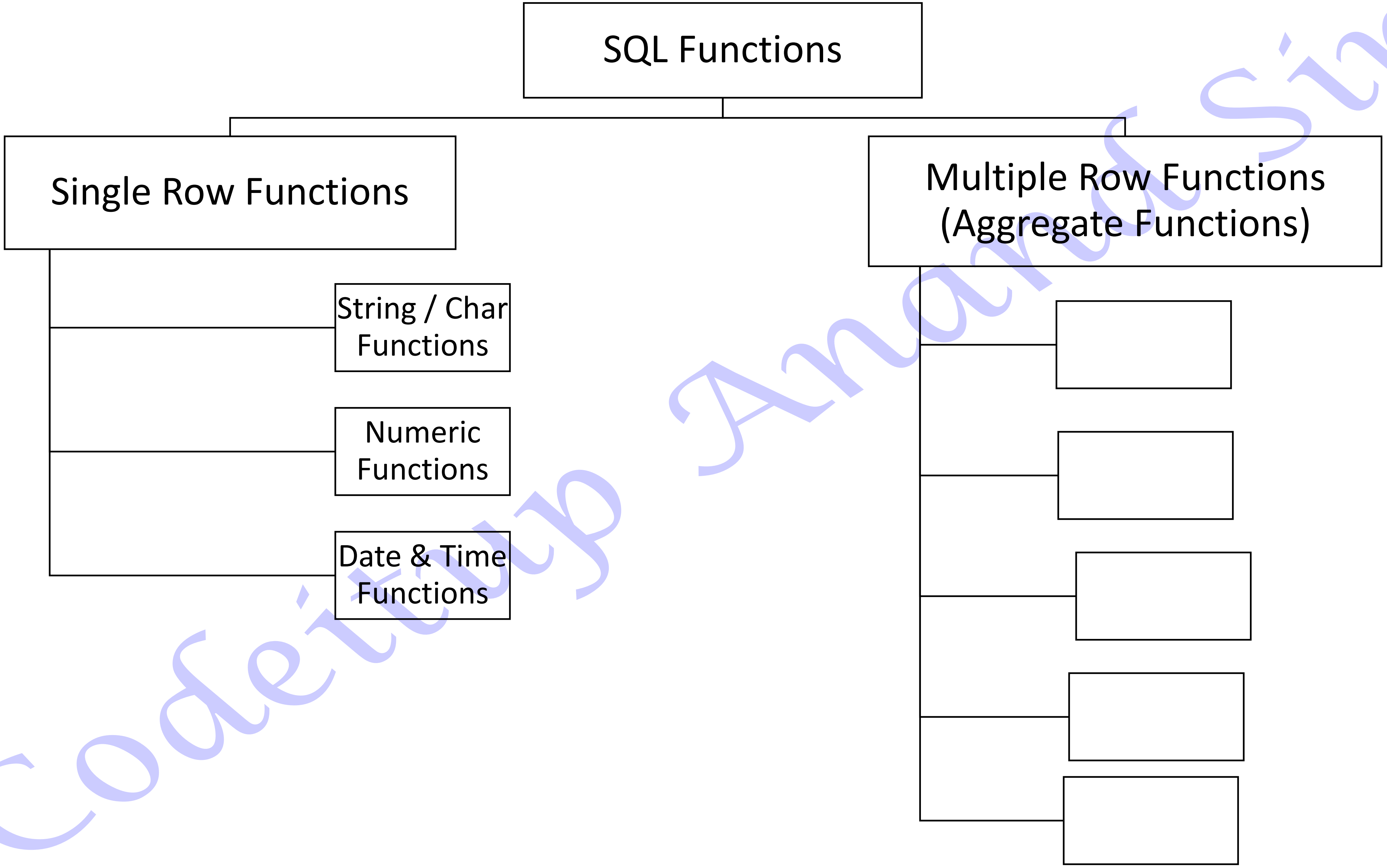
12. Drop

This command is used to Delete the data as well as the structure of the Table.

Syntax : MySQL> Drop Table_Name;

Example : MySQL> Drop students;

The main difference between delete and drop command is that drop command deletes the data as well as the structure of the table while delete command deletes the data only.


SQL Functions
Single Row Functions
Multiple Row Functions (Aggregate Functions)
String / Char Functions
Numeric Functions
Date & Time Functions

MySQL Functions

Function : A function is a perform a predefine task.

String / Char Functions

1. Char : Return character of given ASCII code.

Syntax : MySQL> Select char(Number);

Example : MySQL> Select char(65);

2. Concat : It concatenates two different strings.

Syntax : MySQL> Select concat(Val1, Val2, …);

Example : MySQL> Select concat('Name', 'Middle', 'Last');

ASCII : American Standard Code for Information Interchange.

A : 65, B : 66, C : 67, …   a : 98, b : 99, c : 100, …

3. Lower / lcase : This function converts its argument in its lower case.

Syntax : MySQL> Select lower('Value');

Example : MySQL> Select lower('RAM'); ➔ ram

4. Upper / ucase : This function converts its argument into upper case.

Syntax : MySQL> Select upper('Value');

Example : MySQL> Select lower('ram'); ➔ RAM

5. Substr :

Syntax : MySQL> Select substr('String', Position, No. of Characters);

Example 1: MySQL> Select substr('Jitendra', 5, 3); ➔ ndr

Example 2: MySQL> Select substr('Jitendra', -5, 3); ➔ end

6. LTRIM : This function remove whitespaces from the beginning.

Syntax : MySQL> Select LTRIM('Value');

Example : MySQL> Select LTRIM('   Jitendra   123'); ➔ Jitendra   123

7. RTRIM : This function remove whitespaces from the last.

Syntax : MySQL> Select RTRIM('Value');

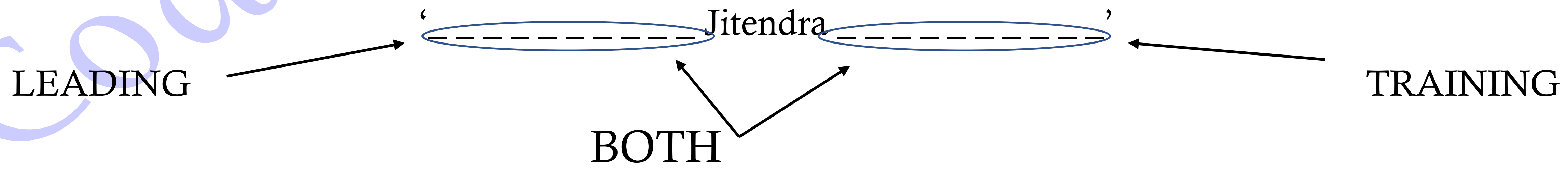
Example : MySQL> Select RTRIM('123   Jitendra   '); ➔ 123   Jitendra

8. TRIM : This function remove whitespaces from the beginning as well as from the end of the string.

Syntax : MySQL> Select RTRIM('Value');

Example : MySQL> Select RTRIM('   Jitendra   '); ➔ Jitendra

OR



LEADING Syntax : MySQL> Select TRIM(LEADING 'Val' from 'String');

Example : MySQL> Select TRIM(LEADING 'X' from 'XXXXJitendraX');

➔ JitendraX

TRAINING Syntax : MySQL> Select TRIM(TRAINING 'Val' from 'String');

Example : MySQL> Select TRIM(TRAINING 'X' from 'XJitendraXXXX');

➔ XJitendra

BOTH Syntax : MySQL> Select TRIM(BOTH 'Value' from 'String');

Example : MySQL> Select TRIM(BOTH 'X' from 'XXJitendraXXXX');

➔ Jitendra

9. INSTR : This function searches 2nd string in 1st string and returns its index value if 2nd string is not found than returns 0.

Syntax : MySQL> Select INSTR(1st–String, 2nd–String);

Example : MySQL> Select INSTR('Jitendra', 'dra'); ➔ 6

10. Length : This function returns the length of the given string.

Syntax : MySQL> Select Length('String');

Example : MySQL> Select Length('Jitendra'); ➔ 8

11. Left : This function return given no. of characters from left side of the string.

Syntax : MySQL> Select Left('String', No. of Characters);

Example : MySQL> Select Left('Jitendra', 4); ➔ Jeet

12. Right : This function return given no. of characters from right side of the string.

Syntax : MySQL> Select Right('String', No. of Characters);

Example : MySQL> Select Right('Jitendra Kumar', 9); ➔ dra Kumar

13. Mid : This function return given no. of characters from given position in the string.

Syntax : MySQL> Select Mid('String', Position, No. of Characters);

Example : MySQL> Select Mid('Jitendra Kumar', 5, 8); ➔ ndra Kum

Numeric Functions

1. MOD : This function divides 1st_No. with 2nd_No. and returns the remainder.

Syntax : MySQL> Select MOD(1st_No., 2nd_No.);

Example 1: MySQL> Select MOD(4, 2); ➔ 0

Example 2: MySQL> Select MOD(4, 0); ➔ 4

2. POWER / POW : This function return 2nd_No. raised to the power 1st_No.

Syntax : MySQL> Select Power(1st_No.,2nd_No.);

Example : MySQL> Select Power(2, 3); ➔ $2^3 = 8$

3. Round : This function rounds of a given no. up to a given no. of digits.

Syntax : MySQL> Select Round(No., No. of Digits);

Example : MySQL> Select Round(152.7932, 1); ➔ 152.8

Example 2 : MySQL> Select Round(157.8, -1); ➔ 160

-1	➔	Nearest Ten's	-2	➔	Nearest Hundred's
-3	➔	Nearest Thousand's	-4	➔	Nearest Ten Thousand's

4. Truncate : This function truncates the given no. for given no. of decimal places.

Syntax : MySQL> Select Truncate(Given_No., Given no. of decimal places);

Example 1 : MySQL> Select Truncate(157.29, 1); ➔ 157.2

Example 2 : MySQL> Select Round(157.29, 1); ➔ 157.3

Example 3 : MySQL> Select Truncate(14.28, -1); ➔ 10

-1	➔	Nearest Ten's	-2	➔	Nearest Hundred's
-3	➔	Nearest Thousand's	-4	➔	Nearest Ten Thousand's

See the difference between Round & Truncate

5. Sign : This function return the sign of given no.

1 → Positive, -1 → Negative, 0 → Zero

Syntax : MySQL> Select Sign(No.);

Example 1 : MySQL> Select Sign(-30); → 1

Example 2 : MySQL> Select Sign(10); → -1

Example 3 : MySQL> Select Sign(0); → 0

6. SQRT : This function returns the square root of given no.

Syntax : MySQL> Select SQRT(No.);

Example 1 : MySQL> Select SQRT(25); → 5

Example 2 : MySQL> Select SQRT(30); → 5.477225575…

Date & Time Functions

1. CURDATE / Current_Date : This function returns current system date.

Syntax : MySQL> Select Current_Date( ); ➔ YYYY-MM-DD

Que.) How to display the date after 10 days from current date?

Ans.) MySQL> Select CURDATE( )+10;

2. Date : This function return the date from its expression.

Syntax : MySQL> Select date(FieldName) from TableName;

OR MySQL> Select date('2020-11-26 01:02:03'); ➔ 2020-11-26

3. Month : This function return month from the given date.

Syntax : MySQL> Select Month(Date);

Example : MySQL> Select Month('2020-11-26 01:02:03'); ➔ 11

4. MonthName : This return name of the month from the given date.

Syntax : MySQL> Select MonthName(Date);

Example : MySQL> Select MonthName('2020-12-06 01:02:03'); ➔ December

5. Day : This function return day from the given date.

Syntax : MySQL> Select Day(Date);

Example : MySQL> Select Day('2020-12-06 01:02:03'); ➔ 06

Que.) How to display only day of the today's date.

Ans.) MySQL> Select Day(Curdate( )); ← Nested Function

6. DayName : This function return the day in text like Sunday, Monday, …etc.

Syntax : MySQL> Select Dayname(Date);

Example : MySQL> Select Dayname('2021-05-13'); ➔ Thursday

7. Year : This function return year from given date.

Syntax : MySQL> Select Year(Date);

Example : MySQL> Select Year('2021-05-13); ➔ 2021

8. Now : This function return current date & time in the format of

YYYY-MM-DD HH:MM:SS

Syntax : MySQL> Select Now( ); ➔ 2021-05-13 02:05:36

9. Sysdate : This function return the system date & time in the format of

YYYY-MM-DD HH:MM:SS

Syntax : MySQL> Select Sysdate( ); ➔ 2021-05-13 02:05:36

Que.) What is difference between Now( ) & Sysdate( ) ?

Que.) What is the difference between Day( ) & DayOfMonth( ) ?

10. DayOfYear : This function return Day of the given date.

Syntax : MySQL> Select DayOfYear(Date);

Example : MySQL> Select DayOfYear('2021-05-13 02:05:36); ➔ 133

11. DayOfMonth : This function return Day of the given date.

Syntax : MySQL> Select DayOfMonth(Date);

Example : MySQL> Select DayOfMonth('2021-05-13 02:05:36); ➔ 13

12. DayOfWeek : This function return week no. of given date.

Syntax : MySQL> Select DayOfWeek(Date);

Example : MySQL> Select DayOfWeek('2021-05-13 02:05:36); ➔ 5

1	➔	Monday	2	➔	Tuesday	3	➔	Wednesday
4	➔	Thursday	5	➔	Friday	6	➔	Saturday

Aggregate Functions MySQL

The function that operates on multiple rows at a time is called aggregate functions.

1. AVG : This function return the average of its argument.

Syntax : MySQL> Select Avg(Field) From TableName;

Example : MySQL> Select Avg(Field) From TableName;

2. SUM : This function return the sum of its argument.

Syntax : MySQL> Select Sum(Field) From TableName;

Example : MySQL> Select Avg(Field) From TableName;

3. MAX : This function return the maximum value of its argument.

Syntax : MySQL> Select Max(Field) From TableName;

Example : MySQL> Select Max(Fees) From students;

4. MIN : This function return the minimum value of its argument.

Syntax : MySQL> Select Min(Field) From TableName;

Example : MySQL> Select Min(Fees) From students;

5. COUNT : This function return the Total no. of rows **OR** Counting of values in given column (NOT NULL).

Syntax : MySQL> Select Count(*) From TableName;

Example : MySQL> Select Count(Name) From students;

Count the total number of values in particular column, If it is blank than it is not count.

Que.) Display the details of the student having maximum fees.

Ans.) MySQL> Select * From students where fees=(select max(fees) from students);

Foreign Key

It is a type of constraints (Set of Rules).

Table name : employee

Code	Name	DeptCode
101	Amit	1001
102	Ajay	1002
103	Aman	1003

Primary Key

For this table **DeptCode** is not primary key

Table Name : dept

DeptCode	Name	HOD
1001	Admin	XYZ
1002	HR	MNO
1003	Finance	ABC

For this table **DeptCode** is primary key

In this table, the primary key is Code in this table, DepCode is Primary Key. So, Here **DeptCode** in employee Table is called **Foreign Key**.

Keyword to use foreign key is references **OR** Foreign key create a term **Referential Integrity.**

Syntax 1 : Create New Table & Foreign Key

```
MySQL> Create Table dept (
        -> DeptCode      Integer          primary key,
        -> NameOfDept    varchar (50)     NOT NULL,
        -> HOD           varchar (50)     NOT NULL);
MySQL> Create Table employee (
        -> Code       Integer          primary key,
        -> Name       varchar (50)     NOT NULL,
        -> Salary     decimal,
        -> DeptCode   Integer references dept.DeptCode);
MySQL> Desc emp;
```

Step 1 : Create Table Which have Primary Key

Keyword for use Foreign Key

Syntax 2 : Add Foreign Key In Existing Table

```
MySQL> Alter Table employee ADD Foreign Key (DeptCode) refrences dept(DeptCode);
```

Join

Join is a query. Joins are used to extract data from two or more tables. There are different types of joins viz. Equi Join, Natural Join, Non-Equi Join, Cartesian product, Cross Join and many more. But exam point of view, Equi Join is most important.

Equi-join is a way to retrieve data from two or more table using (=) equality operator.

Note : When we want to extract data from 2 or more tables it is only possible if one column is common in two tables. If you don’t have proper column then you don’t extract data.

Example

Table name : employee

Code	Name	DeptCode
101	Amit	1001
102	Ajay	1002
103	Aman	1003

Table Name : dept

DeptCode	Name	HOD
1001	Admin	XYZ
1002	HR	MNO
1003	Finance	ABC

Here employee name alias as **e**

```
MySQL> Select code, e.Name, DeptCode from employee e, dept where e.DeptCode=dept.DeptCode;
```

```
MySQL> Select e.*, dept.* from employee e, dept where e.DeptCode=dept.DeptCode;
```

* Represents select all columns from table

Table name : employee

Code	Name	DeptCode
101	Amit	1001
102	Ajay	1002
103	Aman	1003

Here DeptCode column is common in both tables

Table Name : dept

DeptCode	Name	HOD
1001	Admin	XYZ
1002	HR	MNO
1003	Finance	ABC

Types of Joins

Unrestricted Join	Restricted Join
1. When we are joining 2 or more tables without any joining condition 2. Cartesian Product	1. When we are joining 2 or more tables with a join condition. 2. Proper Data is Retrieve.

Cartesian Product (Cross Join)

Table Name : a

Roll_No	Name
1	Ajay
2	Sanjay
3	Preeti

Table Name : b

Add	Contact_No.
Agra	123456
Mathura	789123
Delhi	456789

Cartesian Product of table a and b.

1	Ajay	Agra	123456
1	Ajay	Mathura	789123
1	Ajay	Delhi	456789
2	Sanjay	Agra	123456
2	Sanjay	Mathura	789123
2	Sanjay	Delhi	456789
3	Preeti	Agra	123456
3	Preeti	Mathura	789123
3	Preeti	Delhi	456789

Syntax : MySQL> Select * From a, b;

Equi Join (=) : When joining is done using the equality operator.

Table Name : emp

Code	Name	Salary	deptcode
1	Ajay	12000	101
2	Sanjay	15000	102
3	Preeti	11000	103
4	Anu	90000	102

Table Name : dept

deptcode	Add	Mobile
101	Agra	123456
102	Mathura	789123
103	Delhi	456789

```
MySQL> Select * From emp, dept where emp.deptcode=dept.deptcode;
```

Equi Join

Output :

Code	Name	Salary	deptcode	deptcode	Add	Mobile
1	Ajay	12000	101	101	Agra	123456
2	Sanjay	15000	102	102	Mathura	789123
3	Preeti	11000	103	103	Delhi	456789
4	Anu	90000	102	102	Mathura	789123

Redency of Data : One Data show multiple times

Output :

Code	Name	Salary	deptcode	deptcode	Add	Mobile
1	Ajay	12000	101	101	Agra	123456
2	Sanjay	15000	102	102	Mathura	789123
3	Preeti	11000	103	103	Delhi	456789
4	Anu	90000	102	102	Mathura	789123

Note : If you want fully qualified columns name i,.e, Do not show same column multiple times.

MySQL> Select emp.*, dept.Add, dept.Mobile From emp, dept where emp.deptcode=dept.deptcode

Output ➔

Code	Name	Salary	deptcode	Add	Mobile
1	Ajay	12000	101	Agra	123456
2	Sanjay	15000	102	Mathura	789123
3	Preeti	11000	103	Delhi	456789
4	Anu	90000	102	Mathura	789123

```
MySQL> Select emp.*, dept.Add, dept.Mobile From emp, dept where
emp.deptcode=dept.deptcode AND emp.Salary>11000;
```

OR (If Table Name is too Large)

```
MySQL> Select e.code, e.Name, e.Salary, d.Add from emp e, dept d where
e.deptcode=d.deptcode AND e.Salary>11000;
```

Alias Table Name

Table Name

Note : If a particular column name is same on both tables than in query we write which column name is selected from table i.e., emp.Code Otherwise if a particular column name is unique in both tables than use only its name.

MySQL> Select Code, e.Add, Salary, Mobile from emp e, dept d where e.deptcode=d.deptcode.

Equi Join (=)

MySQL> Select emp.*, dept.Mobile from emp, dept where emp.deptcode=dept.deptcode;

Non-Equi Join (≠)

MySQL> Select emp.*, dept.Mobile from emp, dept where salary Between 10000 AND 15000;

Cartesian Product

1. MySQL> Select * From emp JOIN dept;

   OR

   MySQL> Selct * From emp CROSS JOIN dept;

Equi Join

Condition

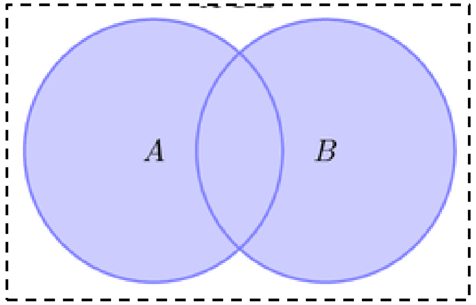
2. MYSQL> Select * From emp e JOIN dept d ON (e.deptcode=d.deptcode);
3. MySQL> Select * From emp NATURAL JOIN dept;
4. MySQL> Select e.Name, Salary From emp e where
   -> Salary Between 10000 AND 15000;

Common Columns does not repeat

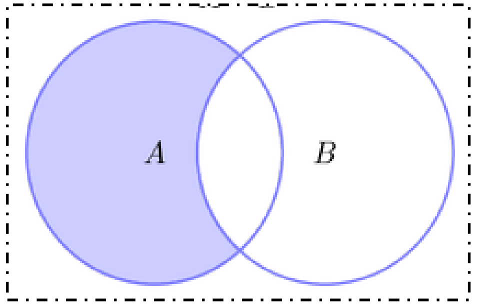
Non-Equi Join

Set Operation In MySQL

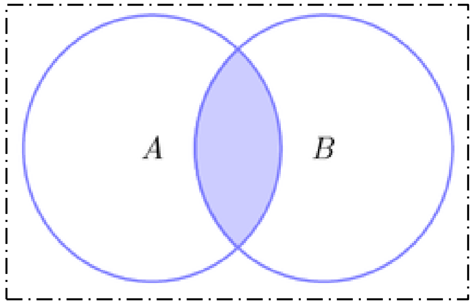
UNION



MINUS



INTERESECT



A

Roll_No	Name	Age
101	Aman	15
102	Chaman	17
103	Naman	21

B

Roll_No	Name	Age
501	Kajal	25
103	Payal	21
502	Ghayal	18

Set Operation In MySQL

1. UNION : Returns all the row from the tables and display the duplicate data just once.

Syntax : MySQL> Select * From A UNION Select * From B;

2. INTERSECT : Returns the common row from the tables.

Syntax : MySQL> Select * From A INTERSECT Select * From B;

3. MINUS : It gives all the data from A deducting the data from B.

Syntax : MySQL> Select * From A MINUS Select * From B;

UNION ALL : Show all rows of both tables either is same or not.

--------------------------------Thanks For Studying--------------------------------