**SQL**

**What is Database:** Database is a collection of interrelated data.

**What is DBMS:** DBMS (*Database Management System*) is software used to create, manage, and organize databases.

**What is RDBMS?**

* RDBMS (Relational *Database Management System)* - is a DBMS based on the concept of tables (also called relations).
* Data is organized into tables (also known as relations) with rows (records) and columns (attributes).
* Eg - MySQL, PostgreSQL, Oracle etc.

**What is SQL?**

SQL is *Structured Query Language* - used to store, manipulate and retrieve data from

RDBMS. (It is not a database, it is a language used to interact with database)

We use SQL for *CRUD* Operations:

* **CREATE** - To create databases, tables, insert tuples in tables et
* **READ** - To read data present in the database.
* **UPDATE** - Modify already inserted data.
* **DELETE** - Delete database, table or specific data point/tuple/row or multiple rows.

**Note** - SQL keywords are NOT case sensitive. Eg: select is the same as SELECT in SQL.

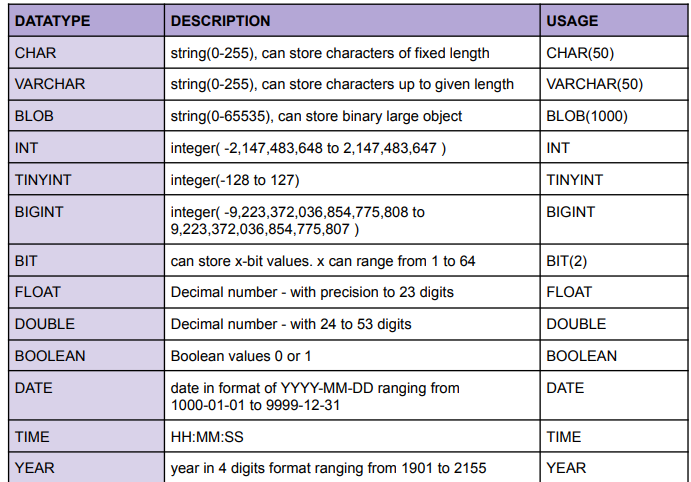
**SQL v/s MySQL**

SQL is a language used to perform CRUD operations in Relational DB, while MySQL is a RDBMS that uses SQL

**Note**: A database can consist of multiple tables.

**SQL Data Types**

In SQL, data types define the kind of data that can be stored in a column or variable. Here are the frequently used SQL data types:

****

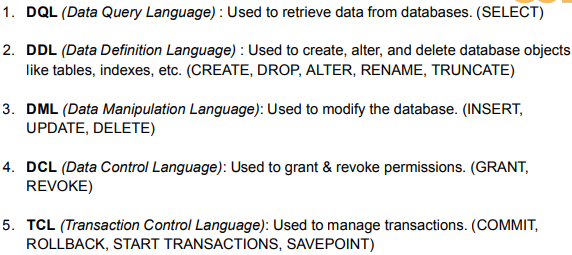
**Difference between CHAR and VARCHAR:**  CHAR occupies all the alloted memory while VARCHAR occupies only the required memory from the alloted memory.

Ex, in char(50) and varchar(50) we store “Chintu” char will occupy all the 50 bytes while varchar will only occupy 6 bytes and remaining bytes can be used to store something else.

**Signed Data type:**  Datatype can store positive and negative values. Ex, **TINYINT** (Range -128 to 127)

**Unsigned Data type:** Datatype can only store positive values.Ex, **UNSIGNED** **TINYINT** (Range 0 to 255) Dekh yaha saara negative ka range positive ko shift kr diya h

**Types of SQL Commands**

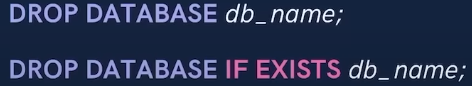


**Creating and Deleting a Database**

**Syntax:** There are 2 ways to create a database

1.

2. here **IF NOT EXISTS** says if the database named **“db\_name”** doesn’t exist than create a database named that. Otherwise, don’t.



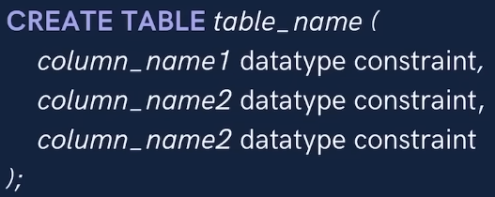
**Syntax:** There are 2 ways of deleting a database

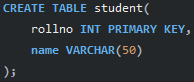
**Use Keyword**

****It says from now on do all the tasks in the specified database.

**Syntax:**

**Eg,**

**Creating a Table**

****Syntax: Ex,

(It’s a table with 3 columns) ( It’s a table with 2 columns. Col 1 and 2 will store rollno and name respectively)

This may seem superficial for now, sb aage clear hoga.

**Viewing a Database and Table**

****

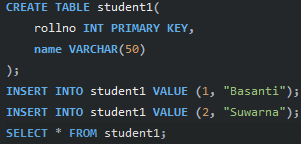
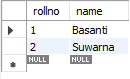
****

Here **\*** means all (more about select in pg 6).

**Inserting into a table**

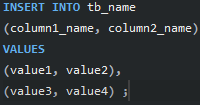
There are 2 ways to Insert in a table

1. **Syntax:** (it’s a syntax of 2 column but we can can create tables with n number of columns.)

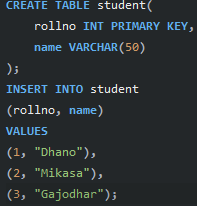
****Value1 and value2 are the value assigned to the columns .

Eg, Output:

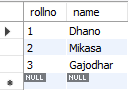
1. **Syntax:**

****Here, it’s a syntax of 2 column but we can can create tables with n number of columns.

Here col1 will store value1 and col2 will store value2 and than again col1 will store value3 and col2 will store value4 and so on.

****

**Eg,**

**** Here a table named student has **2 columns** named **rollno** and **name** which will store below mentioned values.

**Output:**

**So better prefer syntax 2 to store multiple values and syntax 1 for single values.**

**Keys**

**Primary Key:**

For eg, roll\_number in a student table, emp\_id in a employee table, etc.

**Foreign key: It is a column or set of columns in a table which refers to the primary key another table. There can be multiple foreign keys in a table. It can have null values as well as duplicate values. It establishes the link between 2 tables.**

Eg, In table 1 **cityid** is a **foreign key** because it is using (referring) primary key (here id) of another table (here table2).

It might seem superficial for now aage sb clear hoga while coding.

**Note**: Primary key waala table is known as Parent Table. Foreign key waala table => Child Table (as it is using primary key from another table)

**Constraints**

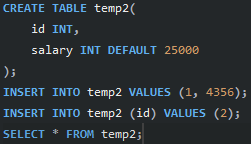
It is used to specify a rule or condition for the table.

Few constraints are mentioned below

**1**

**2**

**3** Mtlb if we keep the value empty, by default the specified value will be assigned to the column

Eg, Output:

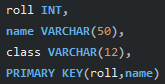
**4 PRIMARY KEY :** makes a column unique & not null but can be used only once in a table.

There are 2 ways of declaring a column as Primary key.

1. (here id the is the name of the column).

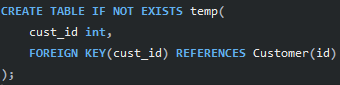


2, (here we have set the contraint to not null we could have kept as empty as well)

We can also set 2 columns are primary key,

Eg, **Here only the combination of roll and name must be unique**. Therefore, we can have same roll but than the name has to different and vice- versa.



**5**

Eg, here, we are saying bhai **cust\_id** from temp table please become a Foreign Key and refer from column **id** of another table (as the definition states).

This way we have linked the temp and Customer table too.

**6**

There are 2 ways of using CHECK constraint

1. Eg Now, if we while inserting we must enter age greater than 17 & branch must be equal to INFT otherwise we’ll face error.



1. Eg,

There are many Constraints which we will see ahead.

**Select**

It is used to select and display any specified data or all data from the table

**Basic Syntax**: Here we have selected 2 columns we , can select more or less as required.

**To select and display all columns:**

**To select and display on distinct (unique) values from a Table:**

Here only unique values will be displayed.

**CLAUSE**

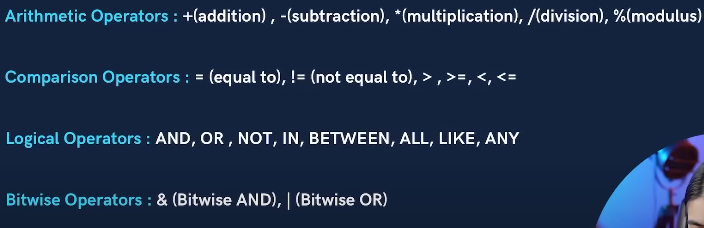
**WHERE Clause**

While retrivieing and displaying data we can put some condition and if that condition is true that row of data is retrived and displayed.

Syntax: In this we have selected col1 and col2 columns we , could have used \* instead.

Eg, Here we are displaying rows where city is , Ranchi

 Here only if marks > 80 from student table.

**Using Operators in Clause**

Sabke baare me pata hi h but logical me some operators are new.

Let’s see those.

**BETWEEN : Selects for a given range.**

**Eg,** displays the row if marks is from 80 to 90 (Note: 80 and 90 are covered here too).

**IN: Matches any value in the list.**

**Eg,** Displays the row if city is Delhi or Mumbai or any described value in the list

**NOT: It negates the given condition specified by logical Operators.**

**Eg,** If city i is , not same as specified in the list than display that row.

**Eg,** But better use **!=** operator.

**LIMIT Clause**

**sets the limit ki itna hi rows(tuples) retrive and display krna.**



**Syntax,** Eg,



We can even combine Limit clause with other condition

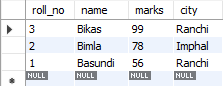
**Order By Clause**

**Arranges the rows in ascending or decending order according to the specified data.**

**Syntax:** Here arranged in descending order of the specified column (here col\_name). Use **ASC** to arrange in ascending order.

If not specified ASC or DESC by default ASC me arrange ho jayega.

Eg, here the rows will be ranged in descending order of the marks

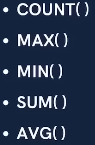
Output : (Assume we have student table with these values in ,it).



Eg, will print top 3 ranked rows.

**Aggregate Function**

Aggregate function takes a set of values and returns a single value. Basically, programming language ke hisab se inbuilt function maan le.

Some Aggregate functions,

****

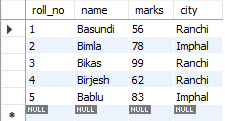
**= > Syntax** Same Syntax for other functions.

As their name suggest they perform that.

**Group By Clause**





Syntax,

Eg, Consider this table,

Isme Group by use krte

h.

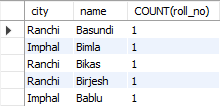


Output => Here grouped city in one column



Output =>

Here grouped city with total number of student in it



Output =>

Here city and name dono common/duplicate hua to we group them into 1 tuple. Assume we have city as Ranchi and name Basundi twice to COUNT(roll\_no) = 2 hota instead of 1. (Basically, read it as Ranchi me basundi naam ki ek ladki h Imphal me bimla naam ki 1 h and so on).

**Having Clause**

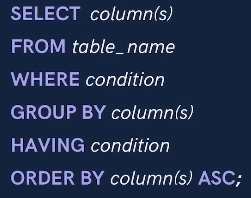


Basically, **Where** clause is used to apply condition on single row(tuple) but **having** is used after we group the tuple.

Eg,

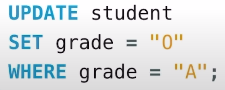
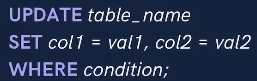
Output => Yaha pahle group bana Ranchi and Imphal ka and then by using , HAVING clause hamne bola group me kisi ek ka bhi marks 90 se upar to hi us group ka data retrive and print krna.

**General Order of writing**

****

**Table Related Quries**

1. **Update (it updates the existing rows in the table)**

****

**Syntax,** Eg,

**Note: Where clause** is optional.

Here we updated the grade of students to “O” if they have grade = “A”

**Note:** In mySQL safe mode is by default on. It doesn’t allow us to change the data from a table. We first must turn it off to perform such quries.

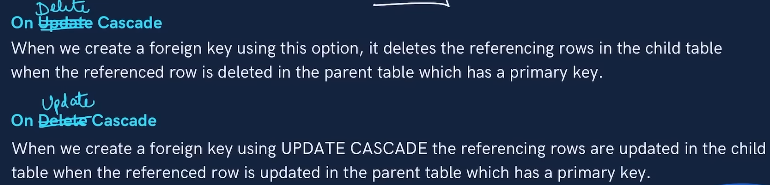
So, write this first. Here 0 indicates safe mode is off while 1 indicates safe mode is on.

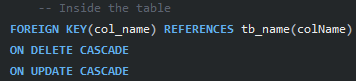
1. **Delete (it deletes the existing rows in the table)**

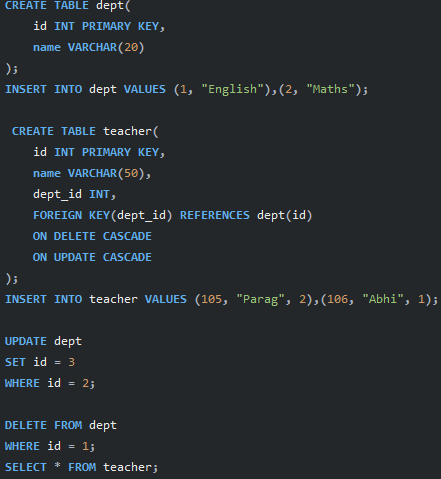
****

**Syntax,** Eg, Delete tuple , jiska marks < 33

**Cascading for Foreign Key**



Syntax,



Eg, Output =>

Basically, all the changes will be reflected in the child table if cascaded.

1. **Alter (changes the schema (basic design) of a table)**

* **Add Column into a table**

Syntax:

* **Drop Column**

Syntax:

* **Rename Table**

Syntax:

* **Change Column**

Syntax:

* **Modify Column’s Datatype and Constraint**

Syntax:

* **Truncate**(Delete all rows in the table)

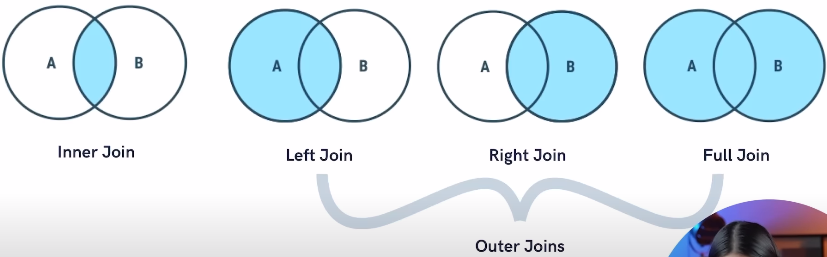
Syntax:

**Note: If we drop a table the complete table is deleted but truncate only deletes all the rows but the table still exists.**

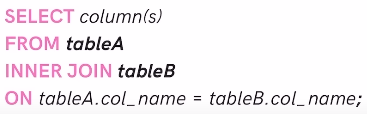
**Join**

****

**Types of Joins**

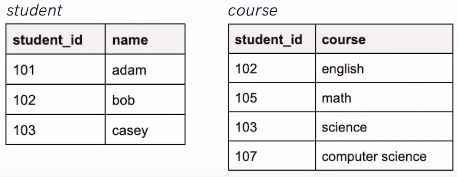
****

1. **Inner Join: Returns records/rows that have matching values.**

** Syntax =>**

Here Instead of column(s) we could have written **\*** as well.

**Note:** It’snot neccesary to have same column name as specified in the syntax.

 Eg: Consider 2 tables

If applied Inner Join =>

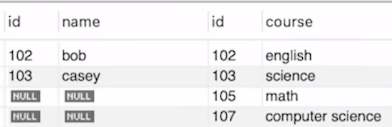
 Output => only 102 and 103 will be printed since they are common.

1. **Left Join: returns all records from the left table and the matched records from both the table.**

**Syntax: Bss remove INNER and write LEFT.**

 Output =>

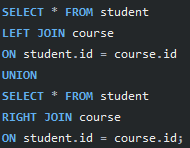
1. **Right Join: returns all records from the right table and the matched records from both the table.**

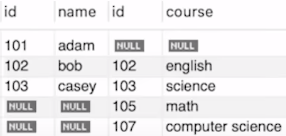
 **Syntax: Bss remove INNER and write RIGHT.**

Output =>

1. **Full Join: returns all the records from the left ,right table and make commons as one record.**

**Note:** MySQL doesn’t have any syntax for Full Join but we make use of Left Join, Right Join and Union (we’ll study about it later).

Here we are saying student table pura and course table pura union (jod do)

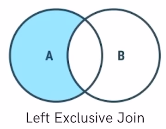
****

Output =>

**Alias**

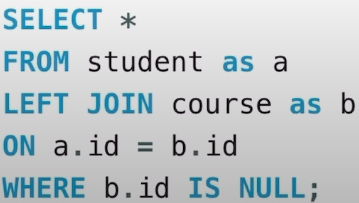
Temporary/alterante names given to tables or columns for the purpose of a particular SQL query.

For eg, we could have written Inner Join’s ex as:

****There are more types of Joins such as **Left Exclusive Join** and **Right Exclusive Joing.**

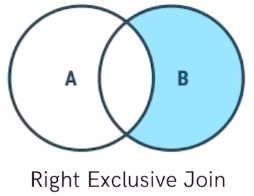
**Left Exclusive Join: return only the uncommon records of left table.**

If we consider upar ke 2 tables student and course as eg,

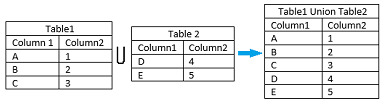
****The code would be

To Understand this code, refer Left Join ka Output

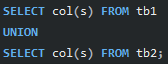
 Output =>

**Right Exlcusive Join: returns only the uncommon records of right table.**

Code khud se kr hi sakta h.

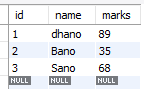
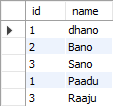
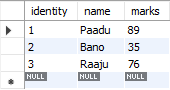
**Union**

It is used to combine columns of two or more Select statements. It is done between 2 or table or can be done In a single table**. It basically combines records and returns unique records only.**

**Syntax:**

To use it :

* Every SELECT should have same no of columns (as mentioned in syntax)
* Columns must have data type.
* Columns in every SELECT should be in same order.

Consider **it** and **extc** table

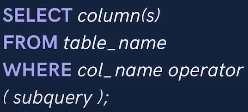
Eg, Output =>

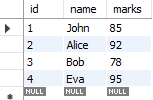
**Note: If we write UNION ALL instead of UNION, we can retrieve duplicate values too.**

**SQL Sub Queries**

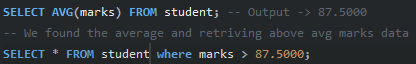
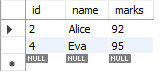
A subquery/ nested query/inner query is a query within a query. It basically involves 2 SELECT statements.

Sub queries can be written inside **SELECT, FROM** and **WHERE** (mostily used in where).

**Syntax**:



consider a student table

If we needed to retrieve data of student who scored above avg marks only what we normally could have done is Output =>

Instead of this, we could have used the concept of Subqueries.

Eg,

It is dynamic since if the table in the future is updated, this code will still work propery but not our previous code.

Eg (with **FROM)**

 Here a table with only citi = Mumbai is created and from there we are retriving max marks.

**NOTE:** In **FROM**, if using subqueireswe must rename the newly created table using Alias.

**MySQL Views**

In Database if we talk about table, it is real, the data stored in it is real. If we perform anything using it the changes are reflected in the database.

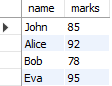
But

View is a virtual table created using a real table which can consisits consist of either all the attributes from a table or the desired attributes from a table. The changes done in Views is not reflected in the database.

**Why use Views?**

Suppose a teacher is checking marks of students. He didn’t need to know age,city of the students. We can create Views here.

**Syntax**: “AS” tells which attributes will be stored in the view.

Eg, Consider student table from above page

Ouput =>

**The End**(5/9/23)