**Databases Notes – mySql**

**In this notes we will learn about all the things of mysql**

**So Lets get started :-**

**Query For Databases**

Question 1) **Query to show database ?**

* Show databases;

**Question 2)Query to create database?**

* CREATE DATABASE <database\_name>

**Question 3) Query to drop database ?**

* DROP DATABASE <database\_name>

**Question 4) Query to Use database ?**

* USE DATABASE <database\_name>

**TABLES AND DATA TYPES**

**Question 1) Query to create a table?**

**🡪create** **table** <table\_name>

(

<column\_name> <column\_DataType>,

<column\_name> <column\_DataType>

)

**Question 2 ) Query to show all tables ?**

**🡪show** tables

**Question 3) Query to Describe tables?**

* DESC <table\_name>

**Question 4)Dropping Tables**

* **drop** <table\_name>

**INSERTING THE DATA INTO TABLES**

**Question 5) Inserting the data into tables**

**🡪insert** **into** dogs(name,age)

**values** (**'roxy'**,10)

**Question 6)Inserting multiple data in single insert call ?(We can insert multiple values in the single insert statement)?**

**🡪insert** **into** dogs (name,age)

**values** (**'bruno'**,10),(**'marsh'**,20),(**'shadow'**,30)

**Question 7) What is not null and null values ?**

* By default when we create the columns their **NULL** properties is automatically set to **yes which means that while inserting the new values we can add the empty field for that column .**
* **On** the other hand when we need that for every entry particular columns needs to be fulfilled then we need to add **not Null clause while defining the columns**

**Below is the query to create columns whose values is not null**

**create** **table** people2(

first\_name **VARCHAR**(128) **not** **null**,

last\_name **VARCHAR**(128) **not** **null**,

age **int** **not** **null**

)

We need to all **not null clause**

**Question 8) What is the Default Value?**

**🡪**Sometimes there are condition arises where we the column values are mandatory , but their may be cases are their where the values for that particular column may would not be their in the dataset due to polluted data , so in that **case we use default values**

**Where the sql automatically fills with the default value in that column if in the insert statements the value for that particular column is not present.**

* Default values may be **standalone values or it could be a functions also .**

**Question 9)Query to insert the default values ?**

**🡪**-- Using of the default values

**create** **table** people3 (

first\_name **varchar**(128) **not** **null** **default** **'unknown'**,

last\_name **varchar**(128) **not** **null** **default** **'unknown'**,

age **int** **not** **null** **default** 0

)

**Question 10) What are the primary key constraint?**

* The primary keys are the one the most important things in the sql tables it created to make each row of data as single **unique entity or indentifier**  which depicts one data .
* **The Primary key should be distinguishable and hence its unique in the table ,**
* **Only single column can be selected as the primary keys .**
* **Primary Keys cannot be null.**

**Question 11) What is unique key constraint ?**

* As the primary keys are the sole indentifier of the data for the table . the unique keys are applied to non primary column where we need **constraint that the column should not have a duplicate values , like the order number and cart number can be unique values , addhar can be values**
* **A table can have multiple unique keys or columns .**

**Question 12) Query to create the Primary Key ?**

**🡪 create** **table** people4 (

people\_id **int** **primary** **key** **auto\_increment**,

first\_name **varchar**(128) **not** **null** **default** **'unknown'**,

last\_name **varchar** (128) **not** **null** **default** **'unknown'**,

age **int** **not** **null** **default** 0

)

**Question 13) What is the AUTO\_INCREMENT Constraints ?**

* When we define the column name with primary key constraints , so with every insert statement we need to add the primary key .
* If the column is just set to create the unique indentity and that column is not present in our data . We can put the constraint to **column with AUTO\_INCREMENT Constraint ,**  what this will do is that it will automatically puts the incremented id for the primary key column .

**CRUD BASICS**

In the below sections we will learn about the crud basics operations and advances to further ladders

**Question 14) Query to select all the entries from the tables ?**

**🡪 select** \* **from** <table\_name>

**Question 15) Query to select only the selected columns only from the table?**

**🡪 select** <column\_name>,<column\_name> **from** <table\_name>.

**We** can also select the specific column from the table rather than selecting the whole all columns

**Question 14) What is WHERE Clause and how to make selection using the where clause ?**

* Where clause works like the filter to filter out the specific data where the condition mets .
  + **select** <column\_name>,<column\_name> **from** <table\_name> **where** <column\_name>=<**any** value>

**Question 15) What are aliases and how to use the aliases ?**

* Aliasing is an exercise where while selecting the data we want to get the data and columns name to changed to some meaningful or required representative names . For that scenario we use the aliasing.
* **Aliasing Just means to provide the temporary pet name to the columns.**
* To alias the column while selecting we use **, the AS keyword.**

**Question 16) Query syntax for the aliasing concepts ?**

-- Aliasing Query syntax

**select** <actual\_column\_name> **as** <aliased\_column\_name> **from** <table\_name>

**Question 17 ) Query syntax for the UPDATE Statements?**

* Update statement is used to update the data pre – present in the table , its also widely used scenario in the databases.
* -- Update Query Syntax
* **update** <table\_name> **set** <column\_name>=<updated\_value> **where** <any\_syntax>

**UPDATE AND SET are two keywords which are required to update the data present in the table.**

**Question 18 ) Query Syntax for the Delete statements ?**

* Delete statements is used to delete the rows from the tables and its important to use **where conditions if we want to delete only certain amount of rows.**

**Below are the syntax to use the delete query**

-- delete query syntax to delete particular rows

**delete** **from** <table\_name> **where** <column\_name>=<some\_value>

-- query syntax to delete all rows (never use this )

**delete** **from** <table\_name>

**Refining Selections - TBD**

**Question 19) Distinct Keyword in sql ?**

* The distinct keyword is used to get the unique items from the columns .

Below is the syntax for the distinct keyword.

-- syntax for the distinct key

**select** **distinct** <column\_name> **from** <table\_name>;

**MOST MOST IMPORTANT ( JOINS and Relationships IN SQL )**

**Question 1) What are the relationships in sql ?**

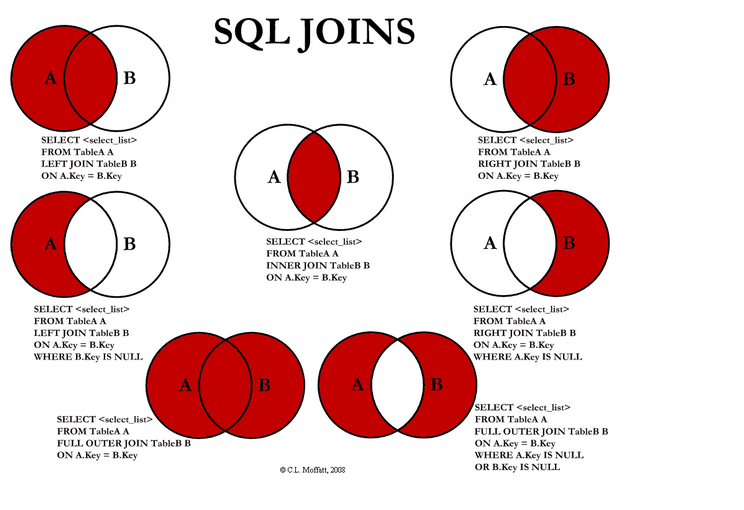
* The relationships are essential part of any sql as In sql we cant create everything in the single table , actually it defeats the rule of the single responsibility principle. And may even may the querying the database very heavy and latency may be increased.
* So we create the relationships between the tables . so each table stores the unit sensible data

**Question 2) What are the kinds of the relationships in sql ?**

* **We are the types of relationships between the tables :-**
  + **One to One Relationship –** This type of relationship exist one table has only one entry in other tables like **data customer relation with aadhar card data ( not have wide use cases)**
  + **One to many Relationship -**  This type of relationship exits one table has many entries in the second table means **one entity could have many entities in other tables (Most Use cases Occurs in this type of the relationships)**
    - **Like :-** orders data , cart data , purchase history , transactions data and many more usecases
  + **Many to many Relationship –** This type of the relationships where many enitities have many relations into other tables (**Second mose usecases occurs in this type of relationships)**
    - **Like one authors have many books and one books can have many authors (So In relation to both tables have many to many relations)**

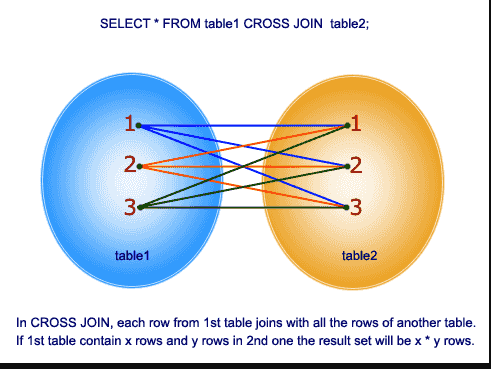
**Question 3) What are the joins ?**

* Now as we have learned that there are relations between the tables , so get the relavent and connected data from the tables in the single shot .
* We have joins which helps to access the data in the single shot .
* Below are the joins we will learn in this topic
  + **Cross Join**
  + **Left Join**
  + **Right Join**
  + **OuterJoin**

****

**Question4) What is cross join ?**

* The cross join is the join where it returns the cartesian products of the rows from the tables .
* Means each rows from table 1 will be joined with the each rows of the table 2
* Its is very less used join in the sql because , it just did not provide us with some analytical insights of joins.

****

**Question 6) Syntax of the Cross Joins ?**

**🡪** -- CrossJoins query syntax

**select** \* **from** customers,orders;

**Question 7) What is the Inner joins ?**

* The inner join is the join where we get the intersection of the two tables , which means the common values which are present in both the tables .
* The inner joins are essential to get the data where we need only those data entitites which are present in both tables .
* For **example provide the data for only those customers which are have orders in the orders tables .**

**Question 8) Provide the Query Syntax for the Inner Join ? (Very Very Important )**

**🡪** -- Inner Joins Query Syntax

-- Inner Join query to get all the customers and orders tables data

**select** \* **from** customers

**inner** **join** orders **on** orders.customer\_id = customers.customer\_id

-- Inner join query syntax to get specified customers and orders tables data

**select** first\_name , last\_name ,email,order\_date,amount **from** customers

**inner** **join** orders **on** orders.customer\_id = customers.customer\_id

Main Key Syntax Points we need to learn

* JOIN – This keyword is required to tell which type of the join we need to make on which tables
* ON – this keyword is to determine on which columns is used as based for the join note columns should be present in both the columns.

**Questions9 ) Provide the query syntax of the inner Join with the Group by syntax?**

* Consider a scenario where we need to get all the total order amount of only those customers from customer table which have maded the order.
* So in these case we can **GROUP BY Clause.**
* -- Question I want total order amount of the customers
* **select** first\_name , last\_name , **SUM**(orders.amount) **as** *order\_total* **from** customers
* **inner** **join** orders **on** orders.customer\_id = customers.customer\_id
* **group** **by** first\_name ,last\_name
* **order** **by** *order\_total* **desc** ;

**Understand the query**

* **We are taking all the required the parameters from both customers and orders tables selective search**
* **We maded the inner join because we wanted only the intersection values between the two tables**
* **Then maded grouping using the firstName and lastname by which we can use the Aggregate sum function**
* **Then we maded sorted in the ascending order.**

**Question 10 ) What is left Join ?**

* The left join is also the special kind of join where we query the two databases based on the relations between them .
* The left join work in a way that , the left join will take all the data from the left table and takes the intersection data from the left table .

**Unlike the inner join where we got only the common data from the two tables , in the left join we will get all the data from the left table and its corresponding data from the right tables**

**Let us suppose the :-**

* In the left table we have 5 customers only
* In the right table we have 10 rows of data , then on the left join all the 5 customers data from right will be taken , rest 5 rows which are in the right table will not be taken

Below is the query for left join

-- Left join query syntax for customers and order data

**select** customers.first\_name ,customers.last\_name ,orders.amount ,orders.order\_date **from** customers

**left** **join** orders **on** customers.customer\_id = orders.customer\_id ;

// things to note on the above syntax

LEFT JOIN – this keyword is used to make the left join

ON – For key relation on which join has to be maded on both the tables

Left Table will always takes the priority.

**Question 11) Query of the Left Join will the group by ?**

* Scenario we need to find the order total of the customers present in the customers table and if the customers doesn’t have any orders then we need to set order total to 0 rather than null . Provide the Query syntax for the same.
* -- left join query syntax for customers and order data with group by
* **select** customers.first\_name ,customers.last\_name,**ifnull**(**SUM**(orders.amount),0) **as** *order\_total* **from** customers
* **left** **join** orders **on** customers.customer\_id=orders.customer\_id
* **group** **by** customers.first\_name ,customers.last\_name
* **order** **by** *order\_total* **ASC**;

**Question 12 ) What is the right join ?**

* Right Join is very similar to left join ,just is that here the all the data is captured from the right table and all the corresponding relative data of right table is taken from the left tables

**Let us suppose the :-**

* In the left table we have 10 customers only
* In the right table we have 5 rows of data , then on the right join all the 5 orders data from right will be taken , rest 5 rows which are in the left table will not be considered

Below is the query for right join :-

-- right join query syntax for customers and order data with group by

**select** customers.first\_name , customers.last\_name , orders.amount ,orders.order\_date **from** customers

**right** **join** orders **on** orders.customer\_id = customers.customer\_id ;

**Question 13) What is the delete cascade ? (Very Very Important) Interview Questions**

**🡪**Now lets us suppose there is the relations between the two tables ie customers and orders , and we want to delete one customer permanently from the customers tables . Can we do it ?

Now if we delete the customer which has relations of orders , the **SQL will give use the error that we cant be able to delete the customer as it has the constraints of foreign key relation on the orders tables .**

**Want :-**

* **We want that if the customers is deleted all its orders should also be deleted automatically .so for this we use (ON DELETE CASCADE) clause while declaring the foreign key .**

**Below** is the query syntax for the same

-- Creating the orders tables

**create** **table** orders (

order\_id **INT** **primary** **key** **auto\_increment**,

order\_date **DATE**,

amount **DECIMAL**(8,2),

customer\_id **INT**,

**foreign** **KEY**(customer\_id) **REFERENCES** customers(customer\_id) **on** **delete** **cascade**

)

**MANY TO MANY RELATIONS**

**Question 14) We will learn about many to many relations ?**

* Apart from one to one relation and one to many relations we also have , many to many relations
* In many to many relations , one entity in table 1 can have many entries in table 2 and one entity in table1 can have many entities in the table 2 .

**Question 15) (Practice Question ) Aggregate functions with joins ?**

-- Find the Count , Min , Max , Avg, Status for each of the users rating

**SELECT** REVIEWERS.FIRST\_NAME ,REVIEWERS.LAST\_NAME,

**COUNT**(REVIEWS.RATING) **AS** **COUNT**,

**IFNULL**(**MIN**(REVIEWS.RATING),0) **AS** **MIN**,

**IFNULL**(**MAX**(REVIEWS.RATING),0) **AS** **MAX**,

**IFNULL** (**AVG**(REVIEWS.RATING),0) **AS** **AVG**,

**CASE**

**WHEN** **COUNT**(REVIEWS.RATING) >= 10 **THEN** **'POWERUSER'**

**WHEN** **COUNT**(REVIEWS.RATING) > 0 **THEN** **'ACTIVE'**

**ELSE** **'INACTIVE'**

**END** **AS** *STATUS*

**FROM** REVIEWERS

**LEFT** **JOIN** REVIEWS **ON** reviews.reviewer\_id = reviewers.reviewer\_id

**GROUP** **BY** REVIEWERS.REVIEWER\_ID ;

**Question 16) (Practice Question) MultiJoins ?(Interview Questions)**

-- Multi Join (Super Important)

**SELECT** series.TITLE ,reviews.rating , **CONCAT**(reviewers.FIRST\_NAME,**' '**,reviewers.LAST\_NAME) **AS** *reviewer* **FROM** SERIES

**INNER** **JOIN** REVIEWS

**ON** series.SERIES\_ID = reviews.SERIES\_ID

**INNER** **JOIN** REVIEWERS

**ON** reviews.REVIEWER\_ID = reviewers.REVIEWER\_ID ;

**VIEWS IN SQL**

**Question 17) What are views in sql ?**

* Views in sql is used to create thevirtual tables from the given set of queries ,the purpose of the views is to provide , pre-calculated query result in a virtual which can used in a different places and user can query on top of it
* The views in sql are auto-updated , if any data is deleted or added in the sql the views get automatically updated

**Question 18) Can You Provide me a syntax to how to create a views ?**

* **Below is the syntax to create the views:-**
* -- Syntax to create a views
* **CREATE** **VIEW** series\_reviewers\_rating **AS**
* -- query by which view will be created
* **SELECT** series.TITLE ,reviews.rating , **CONCAT**(reviewers.FIRST\_NAME,**' '**,reviewers.LAST\_NAME) **AS** reviewer **FROM** SERIES
* **INNER** **JOIN** REVIEWS
* **ON** series.SERIES\_ID = reviews.SERIES\_ID
* **INNER** **JOIN** REVIEWERS
* **ON** reviews.REVIEWER\_ID = reviewers.REVIEWER\_ID;

This is super important to know these keys in the create views section

* **CREATE VIEW <view\_name>** –Is the important keyword which is required to create the view
* **AS**-- keyword is used to set which query is need to create the views

**Question 19 ) Are views table are updateable ?**

* The views tables are generally used for read operations
* **As per best practices we should never alter or modify directly the view table as it is a virtual table**.
* When certain conditions are present like query has **group by , aggregate functions , distinct clause , join in that case we cannot update the view table and many more clauses are their see the sql docs.**
* But we could also update the virtual view tables . If view doesn’t have above clauses defined but still we don’t need to update the views.

**Question 20) How to alter,update or drop views query by which view is created ?**

* Sometimes we have the conditions like where we need to update the query by which the views are created.
* So we could **ALTER the view queries , REPLACE THE VIEWS AND DROP the VIEWS**

-- Updating the query using the create/replace methods

**CREATE** **OR** **REPLACE** **VIEW** ordered\_series **AS**

**SELECT** \* **FROM** series **ORDER** **BY** released\_year **DESC**;

**SELECT** \* **FROM** ***ordered\_series***;

-- ALtering the query using ALTER (most preferred)

**ALTER** **VIEW** ordered\_series **AS**

**SELECT** \* **FROM** series **ORDER** **BY** released\_year **ASC** ;

**Question 20) What is the having clause ?**

* Having clause works same like the where clause , but it only used with aggregrate function and with the group by clause as the agg functions can only be used with group by .
* Below is the syntax example query for the having clause :-
* -- Having clause only works with Group by clause , where doesnt work here
* **SELECT** series.TITLE , **AVG**(reviews.RATING) **AS** *avg\_rating* **FROM** SERIES
* **JOIN** REVIEWS
* **ON** series.SERIES\_ID = reviews.REVIEW\_ID
* **GROUP** **BY** series.TITLE
* **HAVING** *avg\_rating* >8;

**AGGREGATE FUNCTIONS**

**Question 1) Count Aggregate functions ?**

* The count functions returns the total number of **non null rows or entries are present in the data set**
* **The null values for rows or cell will be ignored**

**Below is the query to get all the rows present in the table**

-- Get all the data rows from the database

**SELECT** **COUNT**(\*) **FROM** books;

**Question 2) Count with Distinct Rows functions?**

**🡪**-- Get count of the distinct books title from database

**SELECT** **count**(**DISTINCT**(title)) **FROM** books;

**Question 3) What is Group By (clause) ?--> One of the important thing in the sql**

* Group by clause is used to aggregate/combine rows based on the field we have used the group by clause
* **The most of aggregration function can only be used with Group by clause**

**Example Using count functions using the group by clause.**

-- GRoup by clause

**SELECT** AUTHOR\_LNAME ,**COUNT**(\*) **FROM** books

**GROUP** **BY** AUTHOR\_LNAME ;

Count Aggregration only works with group by clause;

**Question 4) MIN And Max functions ?**

* The min and max functions as name suggest returns the minimum and maximum values from the columns .
* The min and max works for column wise if only it is being queried alone
* The min and max works with **Group BY Clause when grouping the rows .**

**Below is the query syntax of min and max functions**

-- Min and Max functions query syntax example

**SELECT** **MIN**(stock\_quantity) **FROM** books;

**SELECT** **MAX**(stock\_quantity) **FROM** books;

-- Min and Max Syntax with query with group by syntax

**SELECT** **CONCAT**(author\_fname,**' '**,author\_lname),**MIN**(STOCK\_QUANTITY) **AS** *minimum\_stock* , **MAX**(STOCK\_QUANTITY) **AS** *maximum\_stock*

**FROM** books

**GROUP** **BY** AUTHOR\_LNAME , AUTHOR\_FNAME ; 🡪 This is called grouping multiple columns

**Question 5) What are subqueries ?**

**🡪**Subqueries is a shorthand way of writing queries where we can write one query into another mainquery , mostly the subqueries are used in **where , having clauses**

* + **Subquery is always executed first in order.( Important thing to be noted)**
  + **Subquery** should always be written inside the paranthesis.

**Below is the syntax of the subqueries:-**

-- subqueries to select the row which has minimum pages books and maximum pages books

**SELECT** \* **FROM** books **WHERE** pages = (**SELECT** **min**(pages) **FROM** books);

**SELECT** \* **FROM** books **WHERE** pages = (**SELECT** **max**(pages) **FROM** books);

**Question 6 ) Sum and Average aggregate functions ?**

* There are many aggregate functions which are defined in sql docs , But the sum and average are one of the most used aggregate functions .
* Both sum and average aggregate function are used with **group by clause**
* **And** can be used standalone if only used as single query item in the query ,
* -- Sum query syntax
* **SELECT** **sum**(stock\_quantity) **FROM** books;
* **SELECT** **concat**(author\_fname,**' '**,author\_lname) **AS** *author\_name*,**sum**(pages) **FROM** BOOKS
* **GROUP** **BY** *author\_name*;

-- Average query syntax

**SELECT** **avg**(stock\_quantity) **FROM** books;

**SELECT** **CONCAT**(author\_fname,**' '**,author\_lname) **AS** *author\_name* , **avg**(pages) **FROM** BOOKS

**GROUP** **BY** *author\_name*;

**Data Types in Sql**

The data types are the bases of the sql ;

Till now we had learned about Varchar and Int but lets learn about the different , data types present in the sql .

**Different Types of Data Type :-**

* **Char -**  The char data type is used **to store the string** , The char datatype is used to store the fixed length of character .
  + **For** Example if the column is Declared char(4) , so always the data will be stored is of 4 character length , if store less character value for example less than 4 , the char will automatically add empty values to it and then store.(**Very Very Important)**
  + **If** character greater than specified length Is stored inside the column , then the error will be thrown.
* **Varchar(Most Most Used Data Type) -**  The varchar data type is used **to store the string** , the varchar datatype is used to store variable length strings .
  + **For** Example if the column is declared with VARCHAR(10) , then for of whatsoever length we store the data into the column within that column then it will only store the memory on the variable string length (**Important)**
  + **If** string greater than specified length is stored inside the column , then error will be thrown .
* **TinyInt ,Small,MediumInt , Int and Bigint(Important and Most Used Data type) –** These types are used to store integer values in the column , the only difference between these integers are range of value they can store inside the column , and higher the range of values , it can store higher the meomery usage.(**Important)**
* **Decimal –** When we used to store the floating point numbers we can use **decimal data type** to store the values .
  + **The** Decimal data types while declaring two values

abc Decimal(a,b)

where **a = Equal to total digits of decimal which can be stored included the decimal digits**

**b –** The decimal precision digits .

**Below is the example**

Price Decimal(5,2)

The values which can stored in the decimal digits are

112.34 where a = 5 and b = 2 (34)

* If the decimal value (floating values ) exceeds the length then it will rounded to nearest decimal value.
  + Example **(123.456)** then it will be rounded to **123.46**
* But if the value of non decimal digits exceeds the total digit defined while declaration then ??

**Float and Double Types(Important and Most Used) :-**As the decimal is used to store precise values for which we need to define the total digits and total decimal precision points .

We can store the floating point numbers using another data type which is the **float and double.**

* Both Double and Float are used to store the decimal or floating point numbers .
* The only difference between the double and float are the range of value they can store , the float occupies 4 bytes of space whereas the double is used to 8 bytes of the space .

**Date and Time DataTypes:-**

* **Date** is data type where is used to store only date typeof data in the column with format **YYYY-MM-DD**
* **Time** is the data type which is used to store only time typeof data in the column with format **HH-MM-SS.**
* **DateTime** is the data type which is used to store both DateAndTime typeof data in the column with format **YYYY-MM-DD HH-MM-SS**

In the above dataType if the wrong value is inserted **the error will be thrown.(Important)**

**Date and Time Functions:-**

The date and time functions are used to manipulate the date values from different columns and they are used heavily in the sql .

There are dozens of date and Time functions in the sql but we will seeing few which are very necessary .

* **CurrDate()-**It Returns the Current Date in **yyyy-mm-dd** format
* **CurrTime()-** **It** Returns the Current Time in **hh-mm-ss** format
* **Now() –**It Returns the CurrentDateTime in ‘**yyyy-mm-dd hh-mm-ss’** format.

**Date functions**

* **Day (date) –** It Day value from the provided date
* **Year(Date) –** It Returns the Year Value from the Provided Date
* **MonthName(Date) –** It Returns the monthName from the Provided Date like jan , feb , march
* **DayOfWeek (date) –** It returns the week number from the date we have provided .

**Like 0-** Sunday , 1-Monday ……….7-Saturday.

* **DayOfYear(date)** – It returns the day in number out of 365 which are total days in a year .

**Time Functions**

* **Hour(Time) –** It Returns the Hour Section from the Time Argument
* **Minute(Time) –** It Returns the Minute Section from the Time Argument
* **Seconds(Time) –** It Returns the Seconds Section from the Time Argument

Below is the Syntax

For all the functions

-- Using the CurrentDate , Current Time and Now Function to insert the data into the table

**INSERT** **INTO** ***people***(***name***,***birthDate***,***birthTime***,***birthdt***) **VALUES** (**'Rishabh'**,**CURDATE**(),**CURTIME**(),**NOW**());

-- Using the Common date Functions

**SELECT**

**Day**(***birthDate***) **AS** *birth\_day*,

**MONTH**(***birthDate***) **AS** *birth\_month*,

**Year**(***birthDate***) **AS** *birth\_year*,

**DAYNAME**(***birthDate***) **AS** *birth\_dayname*,

**DAYOFYEAR**(***birthDate***) **AS** *birth\_dayofYear*

**FROM** ***people***;

-- Using Common Time Functions

**SELECT**

**HOUR**(***birthTime***) **AS** *birth\_hour*,

**MINUTE**(***birthTime***) **AS** *birth\_minute*,

**SECOND** (***birthTime***) **AS** *Birth\_second*,

**HOUR**(***birthdt***) **AS** *birth\_hour*,

**MINUTE**(***birthdt***) **AS** *birth\_minute*,

**SECOND** (***birthdt***) **AS** *Birth\_second*

**FROM** ***people***;

**More Data types to be Added soon.**

**Comparison and Logical Operators**

Like other languages in sql also we have a comparison operators :-

* Lets learn about them most of the comparison and logical operators will be used with where clauses .

1) **Not Clause –** It will give you result not presenting to value , syntax is **!=**

**2) NOT LIKE –** It will give you results where the data is not Like a sql regex , **syntax is NOT LIKE ‘C%’**

**3) Greater Than –** It is comparison operator which will give all data greater than a value syntax is **where a > b**

**4) Less Than –** It is comparison operator which will give all the data lesser than a value syntax is **where a<b**

**5) Greater Than or Equal to - >**=

**6) Less than or Equal to 🡪 <=**

**Logical Operators**

1) **AND Operator –** It will be used when two or more conditions need to be true , syntax is

**Condition1 AND Condition2**

**2) OR Operator –** It will be used when amongst two or more conditions any one of needs to be true , syntax is **Condition1 OR Condition 2**

**3) Between –** It will be generally used when we to have find whether the data lies between the two range value , Below is Syntax

**Between <value1> AND <value2>**

**4)IN –** This is used a checking operator which will check whether the values lies in set of values just like the python in , Below is the syntax

**Value IN (‘a’,’b’,’c’)**

**Conditional Statements**

**1) Case Statements :-** The case statements is used for conditional Statements same as If else Statements .

**Below Is Syntax**

The Case Statements Starts with keyword CASE and to close the Case Statement we will use the END Key word ,

**When , then and Else is used to handle**

**Case**

When <column\_name> = “some condition1” then “store\_this”

When <column\_name> = “some condition1” then “store\_this”

When <column\_name> = “some condition1” then “store\_this”

When <column\_name> = “some condition1” then “store\_this”

Else ‘ default\_value’

**End as <column\_name>**

2) **IS NULL :-**Selects all the rows or columns where data is null .

**Select \* from books where author\_lname IS NULL;**