

MY SQL PROJECT

BY

Thulasi Dharan R

MY SQL DATABASE

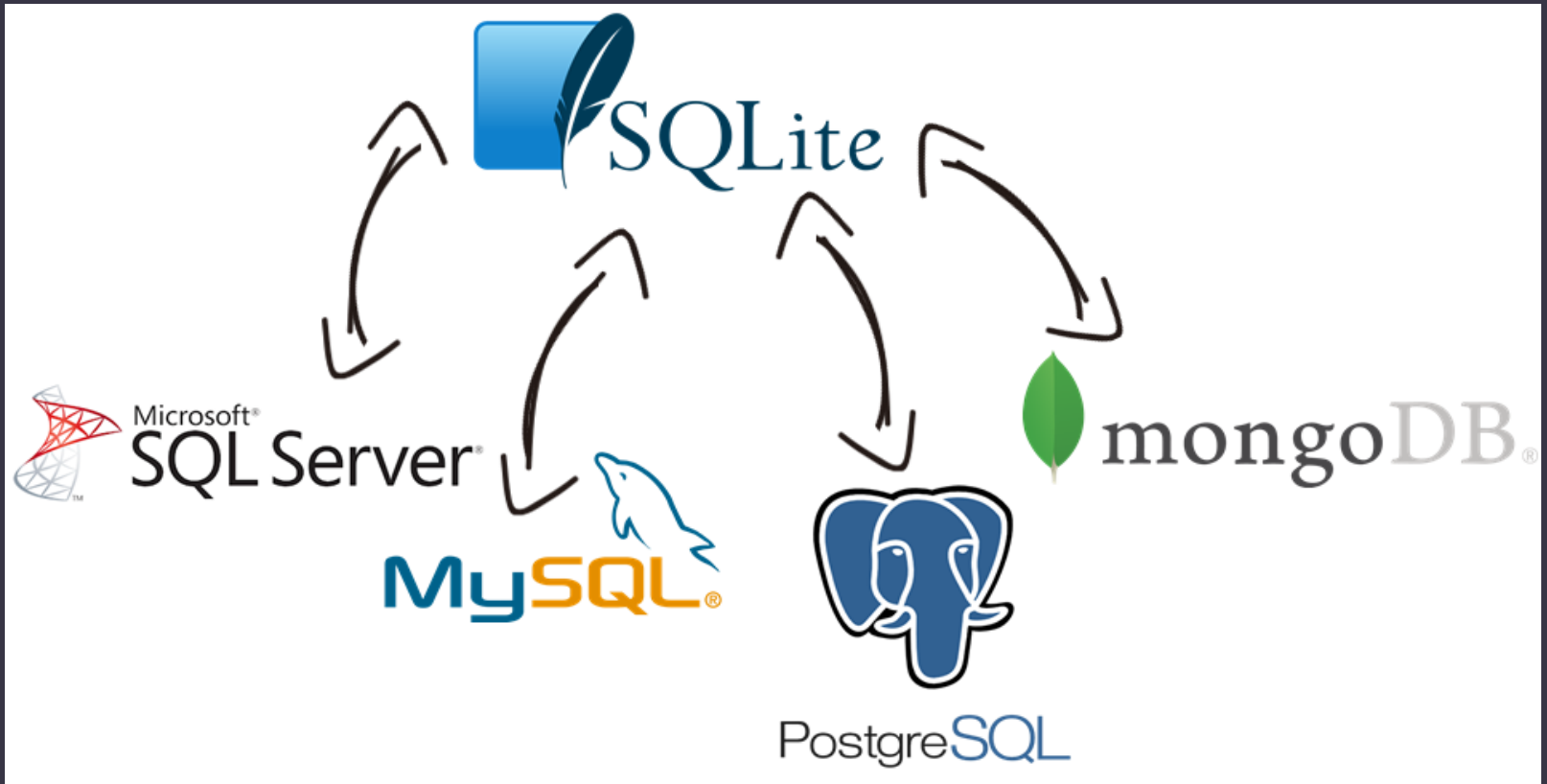
- MySQL is a Database Software.
- MySQL is a relational database management system.
- MySQL is open-source.
- MySQL is free to download.
- MySQL is ideal for both small and large applications.
- MySQL is very fast, reliable, scalable, and easy to use.
- MySQL is secured for storing data in database.

MY SQL WORKBENCH



- **MYSQL Workbench is the Official Graphical User Interface(GUI) Tool**
- **It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data.**
- **You can also migrate schemas and data from other database vendors to your MySQL database.**

MY SQL SERVERS



SQL VS MYSQL

SQL vs MySQL

	SQL	MySQL
Definition	SQL (Structured Query Language) is a programming language used to manage & manipulate databases	MySQL is a popular open-source relational database management system
Developed By	ANSI (American National Standards Institute)	MySQL AB (now owned by Oracle Corporation)
Purpose	SQL is used to define & manipulate the structure & data of database	MySQL is used to store, organize & retrieve data
Function Support	It support user-defined functions & XML	It does not support user-defined functions & XML
Syntax	SQL uses standardize syntax that is accepted by most database systems	MySQL uses a proprietary variant of SQL that includes additional features & capabilities
Servers	The servers & databases in SQL work independently	MySQL servers work independently
Eases Of Use	Complex	Easier

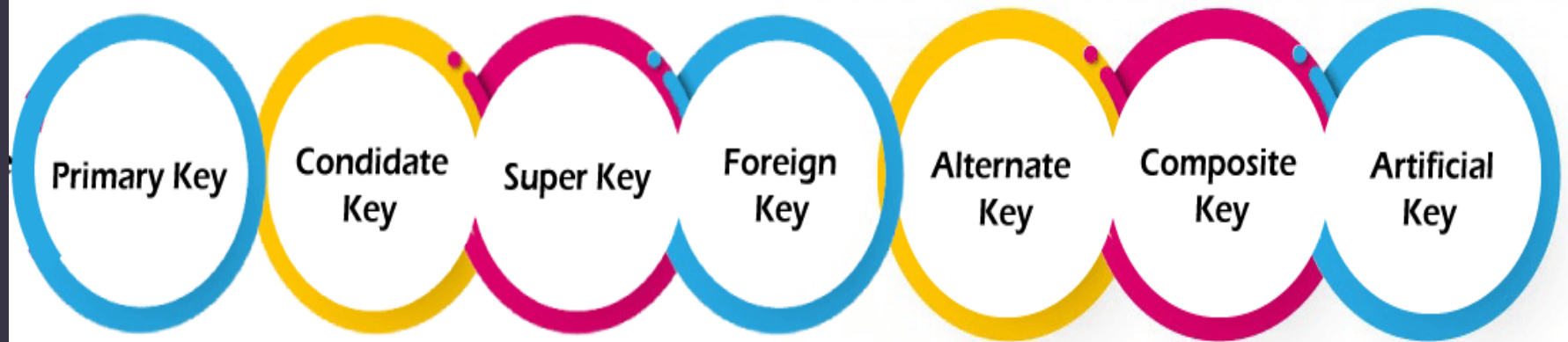
DBMS VS RDBMS

DBMS	RDBMS
DBMS stores data as file.	Stores data in tabular form.
Normalization is not present	Normalization is present
Does not support Distributed databases.	Support Distributed databases
It supports single user	It supports multiple user
It deals with small data	It handles large amount of data
Example of dbms are file systems like xml etc	Examples are MySQL, SQL Server, Oracle
Does not provide any security related data manipulation	It defines integrity constraint for the purpose of ACID properties

DATA TYPES

- **Numeric Datatype.**
- **String Datatype.**
- **DateTime Datatype.**
- **Large Object-(For file and Picture Saving).**

Keys



DBMS KEYS:

- **PRIMARY KEY**
- **SUPER KEY**
- **CANDIDATE KEY**
- **Alternate KEY / Secondary KEY**
- **Foreign KEY**

PRIMARY KEY:

- **A table/ Relation can have only one primary key allowed.**
- **No NULL Values.**
- **No Duplicate Values.**
- **Ex:emp_id.**

SUPER KEY:

- **Set of one or more attributes that allows identifying an entity uniquely.**
- **Ex: (Student_Id, Student_Name, Roll_no, Email_id).**

CANDIDATE KEY:

- **Candidate keys are a subset of super keys.**
- **Multiple candidate keys are allowed.**
- **No repeated Attributes .**
- **Ex: (Student_ID, Roll_no).**

SECONDARY KEY/ALTERNATE KEY:

- **A secondary key is an additional key, or alternate key, which can be use in addition to the primary key to locate specific data**
- **(Primary Key - Candidate key)**

FOREIGN KEY:

- **A foreign key is a reference key.**
- **Foreign key is a key used to link two tables together.**
- **Foreign key is used to maintain relationship between two tables.**
- **Ex. Foreign key(dept_no).**

CONTENTS

- **MySQL General Commands**
- **MySQL General Functions**
- **MySQL String Functions**
- **MySQL Date Functions**
- **MySQL Calculate Functions**
- **MySQL Logical Functions**
- **MySQL Joins**
- **MySQL Stored Procedure**
- **MySQL Triggers**

MY SQL GENERAL COMMANDS

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table

MY SQL GENERAL COMMANDS

- **DDL- DATA DEFINITION LANGUAGE**
- **DML-DATA MANIPULATION LANGUAGE**
- **DQL- DATA QUERY LANGUAGE**
- **DCL-DATA CONTROL LANGUAGE**
- **TCL-TRANSACTION CONTROL LANGUAGE**

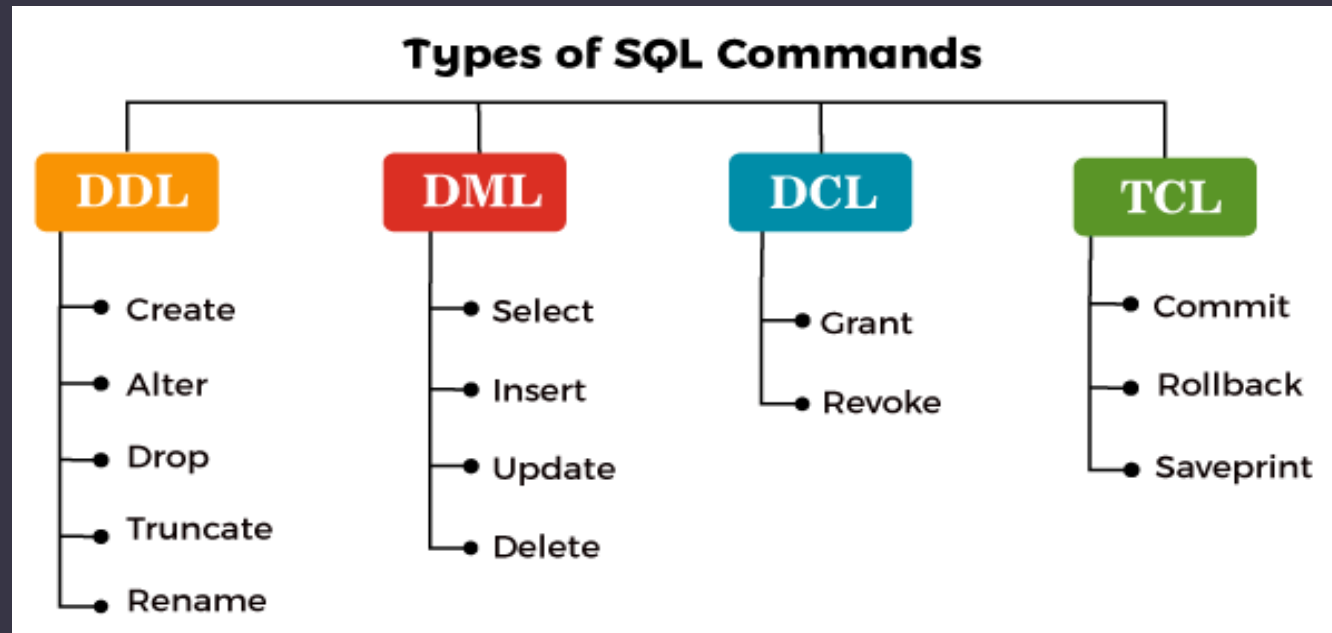


TABLE 1

- **create table EMP_details(EMP_ID int primary key,EMP_Name varchar(30),Designation_ID int,Dept_No int,Date_of_Join date);**
- **Insert into emp_details values;**
- **OUTPUT:**

	EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
▶	17001	Geetha	3001	50	2022-05-10
	17002	Guru	3002	50	2022-05-12
	17003	Gokul	3003	50	2022-05-15
	17004	Mani	3004	60	2022-05-20
	17005	Moorthy	3005	50	2022-05-23
	17006	Amutha	3006	50	2022-06-05
	17007	Jaga	3003	70	2022-06-06
	17008	Pavithra	3007	60	2022-06-07
	17009	Arthi	3005	50	2022-06-08
	17010	Kabilan	3006	70	2022-06-09
	17011	Manasi	3001	70	2022-06-10
	17012	Suja	3002	50	2022-06-11
	17013	Arun	3003	60	2022-06-12
	17014	Deepa	3004	60	2022-06-13
	17015	Sindhu	3005	80	2022-06-14
	17016	Madhavi	3002	50	2022-06-15
	17017	Swetha	3002	70	2022-06-16
	17018	Selvi	3002	70	2022-06-17
	17019	Pooja	3002	70	2022-06-18
	17020	Lakshmi	3007	70	2022-06-19

TABLE 2

- **create table Salary_Details(Salary_ID int primary key, EMP_ID int,Salary_Date date,Branch_ID int,Amount int);**
- **insert into Salary_Details values;**

➤ **OUTPUT:**

	Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
▶	18001	17001	2022-06-10	241	35000
	18002	17002	2022-06-12	241	14000
	18003	17003	2022-06-15	241	28000
	18004	17004	2022-06-20	242	18000
	18005	17005	2022-06-23	241	30000
	18006	17006	2022-07-06	241	23000
	18007	17007	2022-07-07	243	28000
	18008	17008	2022-07-08	242	18000
	18009	17009	2022-07-09	241	30000
	18010	17010	2022-07-10	243	23000
	18011	17011	2022-07-11	243	35000
	18012	17012	2022-07-12	241	14000
	18013	17013	2022-07-13	242	28000
	18014	17014	2022-07-14	242	18000
	18015	17015	2022-07-15	244	30000
	18016	17016	2022-07-16	241	14000
	18017	17017	2022-07-17	243	14000
	18018	17018	2022-07-18	243	14000
	18019	17019	2022-07-19	243	14000
	18020	17020	2022-07-20	243	14000
	18021	17021	2022-07-21	244	14000

TABLE 3

- **create table Designation_Det(Designation_ID int primary key, Designation varchar(40));**
- **insert into Designation_Det values;**

➤ **OUTPUT:**

	Designation_ID	Designation
▶	3001	Manager
	3002	Junior Associates
	3003	Senior Manager
	3004	HR
	3005	General Manager
	3006	Team Lead
	3007	Senior HR
●	NULL	NULL

TABLE 4

- **create table Department_Det(Dep_NO int,Dep_NAME varchar(30),Branch_ID int primary key,Branch_Name varchar(30));**
- **Insert into Department_Det values;**
- **OUTPUT:**

	Dep_NO	dep_name	Branch_ID	Branch_Name
▶	50	Production Department	241	Annan Nagar
	60	HR Department	242	Velachery
	70	Sales Department	243	Guindy
	80	Finance Department	244	KMC
*	NULL	NULL	NULL	NULL

My SQL GENERAL FUNCTIONS

- Where
- or
- and
- in
- Not in
- >
- <
- >=
- <=
- <> (Not equal to)
- !
- Distinct
- Count With Distinct
- Order by Asc
- Order By Desc
- Count
- Group By
- Limit
- Desc Limit
- Like (%)
- Not like
- Between

WHERE

- **The WHERE clause is used to filter records.**
- **It is used to extract only those records that fulfill a specified condition.**
- **QUERY: select * from emp_details where designation_id=3001;**

	EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
▶	17001	Geetha	3001	50	2022-05-10
	17011	Manasi	3001	70	2022-06-10
	17026	Keerthi	3001	60	2022-06-25
	17033	Praveen	3001	80	2022-07-02
•	NULL	NULL	NULL	NULL	NULL

OR

- **The OR operator displays a record if any of the conditions separated by OR is TRUE.**
- **QUERY: select * from emp_details where dept_no=50 or dept_no=60;**
- **OUTPUT:**

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
17001	Geetha	3001	50	2022-05-10
17002	Guru	3002	50	2022-05-12
17003	Gokul	3003	50	2022-05-15
17004	Mani	3004	60	2022-05-20
17005	Moorthy	3005	50	2022-05-23
17006	Amutha	3006	50	2022-06-05
17008	Pavithra	3007	60	2022-06-07
17009	Arthi	3005	50	2022-06-08
17012	Suja	3002	50	2022-06-11
17013	Arun	3003	60	2022-06-12
17014	Deepa	3004	60	2022-06-13
17016	Madhavi	3002	50	2022-06-15
17025	Devan	3006	60	2022-06-24
17026	Keerthi	3001	60	2022-06-25
17028	Raja	3004	60	2022-06-27
NULL	NULL	NULL	NULL	NULL

AND

- The **AND** operator displays a record if all the conditions separated by **AND** are **TRUE**.
- **QUERY : select * from emp_details where dept_no=50 and designation_id=3002;**
- **OUTPUT:**

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
17002	Guru	3002	50	2022-05-12
17012	Suja	3002	50	2022-06-11
17016	Madhavi	3002	50	2022-06-15
NULL	NULL	NULL	NULL	NULL

IN

- The IN operator allows you to specify multiple values in a WHERE clause.
- **QUERY : select * from emp_details where designation_id in (3005,3006);**

➤ **OUTPUT:**

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
17005	Moorthy	3005	50	2022-05-23
17006	Amutha	3006	50	2022-06-05
17009	Arthi	3005	50	2022-06-08
17010	Kabilan	3006	70	2022-06-09
17015	Sindhu	3005	80	2022-06-14
17024	Devi	3005	70	2022-06-23
17025	Devan	3006	60	2022-06-24
17029	Priya	3005	70	2022-06-28
17030	mariya	3006	80	2022-06-29
17031	srinivasan	3005	70	2022-06-30
17032	ganesan	3006	80	2022-07-01
NULL	NULL	NULL	NULL	NULL

NOT IN

- The **NOT IN** operator does not allows you to specify multiple values in a **WHERE** clause.
- **QUERY : select * from emp_details where designation_id not in (3001,3002);**

➤ **OUTPUT:**

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
17003	Gokul	3003	50	2022-05-15
17004	Mani	3004	60	2022-05-20
17005	Moorthy	3005	50	2022-05-23
17006	Amutha	3006	50	2022-06-05
17007	Jaga	3003	70	2022-06-06
17008	Pavithra	3007	60	2022-06-07
17009	Arthi	3005	50	2022-06-08
17010	Kabilan	3006	70	2022-06-09
17013	Arun	3003	60	2022-06-12
17014	Deepa	3004	60	2022-06-13
17015	Sindhu	3005	80	2022-06-14

GREATER THAN

- The greater than operator is used to show the higher values.
- **QUERY : select * from emp_details where designation_id > 3005;**
- **OUTPUT:**

	EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
▶	17006	Amutha	3006	50	2022-06-05
	17008	Pavithra	3007	60	2022-06-07
	17010	Kabilan	3006	70	2022-06-09
	17025	Devan	3006	60	2022-06-24
	17030	mariya	3006	80	2022-06-29
	17032	ganesan	3006	80	2022-07-01
•	NULL	NULL	NULL	NULL	NULL

GREATER THAN OR EQUAL TO

- The greater than or equal to is used to show the higher values and also the equal to values.
- **QUERY : select * from salary_details where salary_id >=18005;**

➤ **OUTPUT:**

Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
18005	17005	2022-06-23	241	30000
18006	17006	2022-07-06	241	23000
18007	17007	2022-07-07	243	28000
18008	17008	2022-07-08	242	18000
18009	17009	2022-07-09	241	30000
18010	17010	2022-07-10	243	23000
18011	17011	2022-07-11	243	35000
18012	17012	2022-07-12	241	14000
18013	17013	2022-07-13	242	28000
18014	17014	2022-07-14	242	18000
18015	17015	2022-07-15	244	30000
18016	17016	2022-07-16	241	14000
18017	17017	2022-07-17	243	14000

LESSER THAN

- The lesser than operator is used to show the lower values.
- **QUERY : select * from emp_details where dept_no < 60;**

➤ **OUTPUT:**

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
17001	Geetha	3001	50	2022-05-10
17002	Guru	3002	50	2022-05-12
17003	Gokul	3003	50	2022-05-15
17005	Moorthy	3005	50	2022-05-23
17006	Amutha	3006	50	2022-06-05
17009	Arthi	3005	50	2022-06-08
17012	Suja	3002	50	2022-06-11
17016	Madhavi	3002	50	2022-06-15
NULL	NULL	NULL	NULL	NULL

LIKE

- The **LIKE** operator is used in a **WHERE** clause to search for a specified pattern in a column.
- The percent sign **%** represents zero, one, or multiple character.
- The underscore sign **_** represents one, single character.
- **QUERY : select * from emp_details where emp_name like '_a%';**

➤ **OUTPUT:**

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
17004	Mani	3004	60	2022-05-20
17007	Jaga	3003	70	2022-06-06

- **QUERY : select * from emp_details where emp_name like 'd%';**

➤ **OUTPUT:**

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
17014	Deepa	3004	60	2022-06-13
17024	Devi	3005	70	2022-06-23

NOT LIKE

- **SQL not like statement syntax be like** **SELECT column FROM table_name WHERE column NOT LIKE pattern**
- **QUERY : select * from designation_det where designation not like 's%';**

➤ **OUTPUT:**

Designation_ID	Designation
3001	Manager
3002	Junior Associates
3004	HR
3005	General Manager
3006	Team Lead
NULL	NULL

COUNT

- The **COUNT()** function returns the number of rows that matches a specified criterion..
- **QUERY** : `select count(emp_id) as Total_employees from salary_details;`

	Total_employees
▶	30

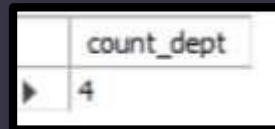
DISTINCT

- The **SELECT DISTINCT** statement is used to return only distinct (different) values.
- **QUERY** : `select distinct designation_id from emp_details;`

	designation_id
▶	3001
	3002
	3003
	3004
	3005
	3006
	3007

COUNT WITH DISTINCT

- This is used to find the count values of distinct values.
- Query: `select count(distinct dept_no)count_dept from emp_details;`



count_dept
4

BETWEEN AND

- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
- QUERY : `select * from salary_details where amount between 10000 and 20000;`

Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
18002	17002	2022-06-12	241	14000
18004	17004	2022-06-20	242	18000
18008	17008	2022-07-08	242	18000
18012	17012	2022-07-12	241	14000
18014	17014	2022-07-14	242	18000
18016	17016	2022-07-16	241	14000

ORDER BY ASCENDING

- **select * from salary_details where amount between 10000 and 20000;**
- **QUERY: select * from salary_details order by amount asc;**

Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
18002	17002	2022-06-12	241	14000
18012	17012	2022-07-12	241	14000
18016	17016	2022-07-16	241	14000
18017	17017	2022-07-17	243	14000
18018	17018	2022-07-18	243	14000
18019	17019	2022-07-19	243	14000
18022	17022	2022-07-22	243	14000

ORDER BY DESCENDING

- **QUERY : select * from salary_details order by amount desc;**

Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
18001	17001	2022-06-10	241	35000
18011	17011	2022-07-11	243	35000
18026	17026	2022-07-26	242	35000
18005	17005	2022-06-23	241	30000
18009	17009	2022-07-09	241	30000
18015	17015	2022-07-15	244	30000
18024	17024	2022-07-24	243	30000

LIMIT

- The limit is used to filter the specified range of values.
- **QUERY :** select * from emp_details limit 23,2;

➤ OUTPUT:

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
17024	Devi	3005	70	2022-06-23
17025	Devan	3006	60	2022-06-24
NULL	NULL	NULL	NULL	NULL

DESC LIMIT

- **QUERY :** select * from salary_details order by salary_id desc limit 29,4;

➤ OUTPUT:

Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
18004	17004	2022-06-20	242	18000
18003	17003	2022-06-15	241	28000
18002	17002	2022-06-12	241	14000
18001	17001	2022-06-10	241	35000
NULL	NULL	NULL	NULL	NULL

GROUP BY

- The **GROUP BY** statement groups rows that have the same values into summary rows, like "find the number of customers in each country".
- The **GROUP BY** statement is often used with aggregate functions (**COUNT()**, **MAX()**, **MIN()**, **SUM()**, **AVG()**) to group the result-set by one or more columns.

QUERY : **select designation_id,count(emp_id) from emp_details
group by designation_id;**

➤ **OUTPUT:**

designation_id	count(emp_id)
3001	4
3002	10
3003	4
3004	3
3005	6
3006	5
3007	1

My SQL String Functions

- L Case
- U Case
- Left
- Right
- Concat
- Trim
- Char_Length
- Mid
- Length

LOWER CASE

Query : select lcase(emp_name), emp_name from emp_details;

OUTPUT:

lcase(emp_name)	emp_name
geetha	Geetha
guru	Guru
gokul	Gokul
mani	Mani
moorthy	Moorthy
amutha	Amutha
jaga	Jaga
pavithra	Pavithra

UPPER CASE

Query : select ucase(emp_name),emp_name from emp_details;

OUTPUT:

ucase(emp_name)	emp_name
GEETHA	Geetha
GURU	Guru
GOKUL	Gokul
MANI	Mani
MOORTHY	Moorthy

LEFT

Query: select left(emp_id,2),emp_name,date_of_join from emp_details;

OUTPUT:

left(emp_id,2)	emp_name	date_of_join
17	Geetha	2022-05-10
17	Guru	2022-05-12
17	Gokul	2022-05-15

RIGHT

Query : select right(emp_id,3),emp_name,date_of_join from emp_details;

OUTPUT:

right(emp_id,3)	emp_name	date_of_join
001	Geetha	2022-05-10
002	Guru	2022-05-12
003	Gokul	2022-05-15

MID

**Query : select mid(emp_name,3,4),emp_name,date_of_join
from emp_details;**

OUTPUT:

mid(emp_name,3,4)	emp_name	date_of_join
etha	Geetha	2022-05-10
ru	Guru	2022-05-12
kul	Gokul	2022-05-15
ni	Mani	2022-05-20
orth	Moorthy	2022-05-23

CONCAT

The CONCAT function adds two or more strings together.

**QUERY:select concat(emp_id,".",ucase(emp_name))
Employee_det, designation_idfrom emp_details;**

OUTPUT:

Employee_det	designation_id
17001.GEETHA	3001
17002.GURU	3002

TRIM

The TRIM function removes leading and trailing spaces from a string.

QUERY : select trim(designation) as jobs from designation_det;

OUTPUT:

jobs
Manager
Junior Associates
Senior Manager

LENGTH

The LENGTH() function returns the number of bytes used to represent an expression

QUERY : select length(branch_name) from department_det;

OUTPUT:

length(branch_name)
11
9
6
3

MY SQL CALCULATE FUNCTIONS

- **Sum**
- **Average**
- **Min**
- **Max**
- **Count**

EXAMPLE:

```
SELECT SUM(sales), AVG(sales), MIN(sales), MAX(sales),  
COUNT(sales) FROM salary_details;
```


The SUM() function returns the total sum of a numeric column.

QUERY: select sum(amount) as total_sal_amount from salary_details;

SUM:

total_sal_amount
671000

AVERAGE:

The AVG function returns the average value of a numeric column.

QUERY : select round(avg(amount),0) as avg from salary_details;

avg
22367

MIN:

The MIN function returns the smallest value of the selected column.

QUERY : select min(amount) as min_sal from salary_details;

min_sal
14000

MAX:

The MAX function returns the largest value of the selected column.

QUERY : select max(amount) as max_sal from salary_details;

max_sal
35000

COUNT:

The COUNT function returns the number of rows that matches a specified criterion.

QUERY : select count(salary_id) from salary_details;

count(salary_id)
30

MY SQL DATE FUNCTIONS

DATE ADD:

QUERY :select *,date_add(date_of_join,interval 1 month)as Salary_date from emp_details;

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join	Salary_date
17001	Geetha	3001	50	2022-05-10	2022-06-10
17002	Guru	3002	50	2022-05-12	2022-06-12
17003	Gokul	3003	50	2022-05-15	2022-06-15

DATE DIFF :

QUERY:select*,round(datediff(curdate(),date_of_join)/365,1) as emp_experience_yrs from emp_details;

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join	emp_experience_yrs
17001	Geetha	3001	50	2022-05-10	1.6
17002	Guru	3002	50	2022-05-12	1.6
17003	Gokul	3003	50	2022-05-15	1.6

TIME STAMP DIFF :

QUERY : select *,timestampdiff(year,date_of_join,curdate()) as emp_exp
from emp_det;

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join	emp_exp
17001	Geetha	3001	50	2022-05-10	1
17002	Guru	3002	50	2022-05-12	1
17003	Gokul	3003	50	2022-05-15	1
17004	Mani	3004	60	2022-05-20	1

DATE FORMAT :

QUERY : select *, date_format(date_of_join,'%a')as days from
emp_details;

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join	days
17001	Geetha	3001	50	2022-05-10	Tue
17002	Guru	3002	50	2022-05-12	Thu
17003	Gokul	3003	50	2022-05-15	Sun
17004	Mani	3004	60	2022-05-20	Fri

YEAR :

**QUERY : select * from salary_details where
year(salary_date)='2022';**

	Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
*	18001	17001	2022-06-10	241	35000
	18002	17002	2022-06-12	241	14000
	18003	17003	2022-06-15	241	28000

MONTH :

**QUERY : select * from salary_details where
month(salary_date)='06';**

	Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
	18001	17001	2022-06-10	241	35000
	18002	17002	2022-06-12	241	14000
	18003	17003	2022-06-15	241	28000

DAY :

**QUERY : select * from salary_details where
day(salary_date)='12';**

	Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
	18002	17002	2022-06-12	241	14000
	18012	17012	2022-07-12	241	14000
	NULL	NULL	NULL	NULL	NULL

MY SQL LOGICAL FUNCTIONS

- **IF**
- **COUNT IF**
- **IF WITH AND CONDITIONS**
- **IF WITH OR CONDITIONS**
- **IF: Conditional statement returning one value if true, another if false.**
- **COUNT IF: Counts rows meeting specified condition.**
- **IF with AND conditions: Returns value if all conditions are true.**
- **IF with OR conditions: Returns value if any condition is true.**

IF :

QUERY : select *, if(dep_no<=60,'Production','HR') as job from department_det;

OUTPUT :

Dep_NO	dep_name	Branch_ID	Branch_Name	job
50	Production Department	241	Annan Nagar	Production
60	HR Department	242	Velachery	Production
70	Sales Department	243	Guindy	HR
80	Finance Department	244	KMC	HR

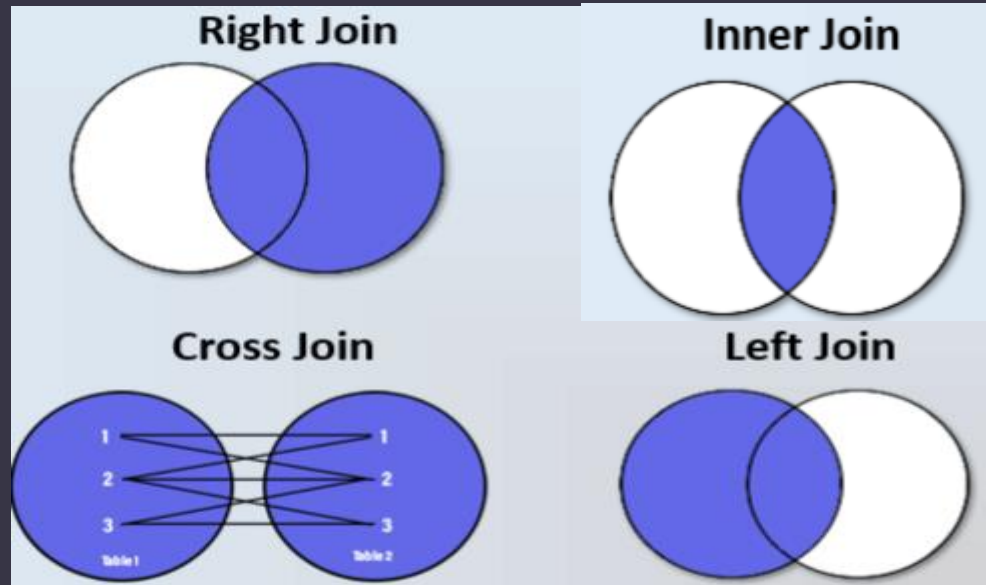
COUNT IF:

QUERY : select *, count(if(dep_no<=60,'Production','HR')) as job from department_det group by branch_id;

OUTPUT :

Dep_NO	dep_name	Branch_ID	Branch_Name	job
50	Production Department	241	Annan Nagar	1
60	HR Department	242	Velachery	1
70	Sales Department	243	Guindy	1
80	Finance Department	244	KMC	1

MY SQL JOINS



- **A JOIN clause is used to combine rows from two or more tables, based on a related column between them.**
- **JOINS TYPES:**
- **INNER JOIN**
- **LEFT JOIN**
- **RIGHT JOIN**
- **CROSS JOIN**

LEFT JOIN

The **LEFT JOIN** keyword returns all records from the left table (table1), and the matching records (if any) from the right table (table2).

QUERY:

```
select * from emp_det left join sal_info on  
emp_det.emp_id=sal_info.emp_id;
```

OUTPUT :

	emp_id	emp_name	Designation	date_of_join	sal_id	emp_id	sal_date	amount
►	1	Guru	Manager	2022-05-10	121	1	2022-06-10	10000
	3	Mani	Senior Manager	2022-05-15	134	3	2022-06-15	12000
	2	Gopi	Junior Accountant	2022-05-12	156	2	2022-06-12	18000
	4	Moorthy	HR	2022-05-20	167	4	2022-06-20	16000
	5	Muthu	General Manager	2022-05-23	178	5	2022-06-23	12000
	6	Abhi	Associate	2022-06-05	NULL	NULL	NULL	NULL

INNER JOIN

QUERY:

```
select* from emp_det inner join sal_info on emp_det .  
emp_id=sal_info.emp_id order by emp_det.emp_id asc;
```

OUTPUT :

	emp_id	emp_name	Designation	date_of_join	sal_id	emp_id	sal_date	amount
▶	1	Guru	Manager	2022-05-10	121	1	2022-06-10	10000
	2	Gopi	Junior Accountant	2022-05-12	156	2	2022-06-12	18000
	3	Mani	Senior Manager	2022-05-15	134	3	2022-06-15	12000
	4	Moorthy	HR	2022-05-20	167	4	2022-06-20	16000
	5	Muthu	General Manager	2022-05-23	178	5	2022-06-23	12000

RIGHT JOIN

The **RIGHT JOIN** keyword returns all records from the right table (table2), and the matching records (if any) from the left table (table1).

QUERY:

```
select * from emp_det right join sal_info on emp_det.emp_id=
sal_info.emp_id;
```

OUTPUT :

emp_id	emp_name	designation_id	designation
NULL	NULL	4	analyst
17001	Geetha	3001	Manager
17033	Praveen	3001	Manager
17011	Manasi	3001	Manager

CROSS JOIN

The **CROSS JOIN** keyword returns all records from both tables (table1 and table2).

QUERY : select * from designation_det cross join department_det;

OUTPUT :

	Designation_ID	Designation	Dep_NO	dep_name	Branch_ID	Branch_Name
•	4	analyst	50	Production Department	241	Annan Nagar
	4	analyst	60	HR Department	242	Velachery
	4	analyst	70	Sales Department	243	Guindy
	4	analyst	80	Finance Department	244	KMC
	3001	Manager	50	Production Department	241	Annan Nagar
	3001	Manager	60	HR Department	242	Velachery
	3001	Manager	70	Sales Department	243	Guindy
	3001	Manager	80	Finance Department	244	KMC
	3002	Junior Associates	50	Production Department	241	Annan Nagar
	3002	Junior Associates	60	HR Department	242	Velachery
	3002	Junior Associates	70	Sales Department	243	Guindy
	3002	Junior Associates	80	Finance Department	244	KMC

CASE END

select *,

Case

when (marks_info1.cost >= 35 and

marks_info1.corporate >= 35 and marks_info1.finance >= 35)

then "PASS"

else "FAIL"

end as results

from

marks_info1;

mark_id	stu_id	corporate	cost	finance	results
14001	1	75	76	65	PASS
14002	2	92	90	19	FAIL
14003	3	38	37	46	PASS
14004	4	39	90	58	PASS
14005	5	34	89	20	FAIL
14006	6	44	38	60	PASS

FOREIGN KEY

A foreign key is like a GPS coordinate in a database, guiding you from one table to another. It ensures data integrity by linking a column in one table to a primary key in another, creating a relationship between them.

QUERY :

TABLE 1 : create table dep_info (dep_no int,dep_name varchar(30),branch_id int,branch_name varchar(25),primary key(dep_no));

TABLE 2 : create table emp_info(emp_id int primary key,emp_name varchar(20),designation_id int,dep_no int,date_of_join date,constraint temp_foreign_key foreign key(dep_no)referencesdep_info(dep_no));

DROP FOREIGN KEY : alter table Emp_info drop foreign key temp_Foreign_Key;

MY SQL STORED PROCEDURE

**A stored procedure is a prepared sql code that you can save,
So the code can be reused over and over again.
So if you have an SQL query that you write over and over again
Save it as a stored procedure, and then just call it to execute it.**

➤ Advantages

Decrease the Network traffic

Centralize Business Logic For ex (If and or Elself)

Fully secured

➤ Disadvantages

Resource usage

Hard to maintain

delimiter //

create procedure multi_query()

Begin

select * from emp_details;

select length(emp_name) from emp_details;

**select * from salary_details where salary_id
in(18005,18012,18032,18023,18014);**

select count(emp_id) from emp_details;

end //

delimiter ;

call multi_query;

EMP_ID	EMP_Name	Designation_ID	Dept_No	Date_of_Join
17001	Geetha	3001	50	2022-05-10
17002	Guru	3002	50	2022-05-12

Salary_ID	EMP_ID	Salary_Date	Branch_ID	Amount
18005	17005	2022-06-23	241	30000
18012	17012	2022-07-12	241	14000
18014	17014	2022-07-14	242	18000
18023	17023	2022-07-23	244	14000

length(emp_name)
6
4
5
4

count(emp_id)
33

VARIABLES IN STORED PROCEDURE

delimiter //

create procedure store_variables()

Begin

declare sum_amount int;

select sum(amount) into sum_amount from salary_details;

select sum_amount;

end //

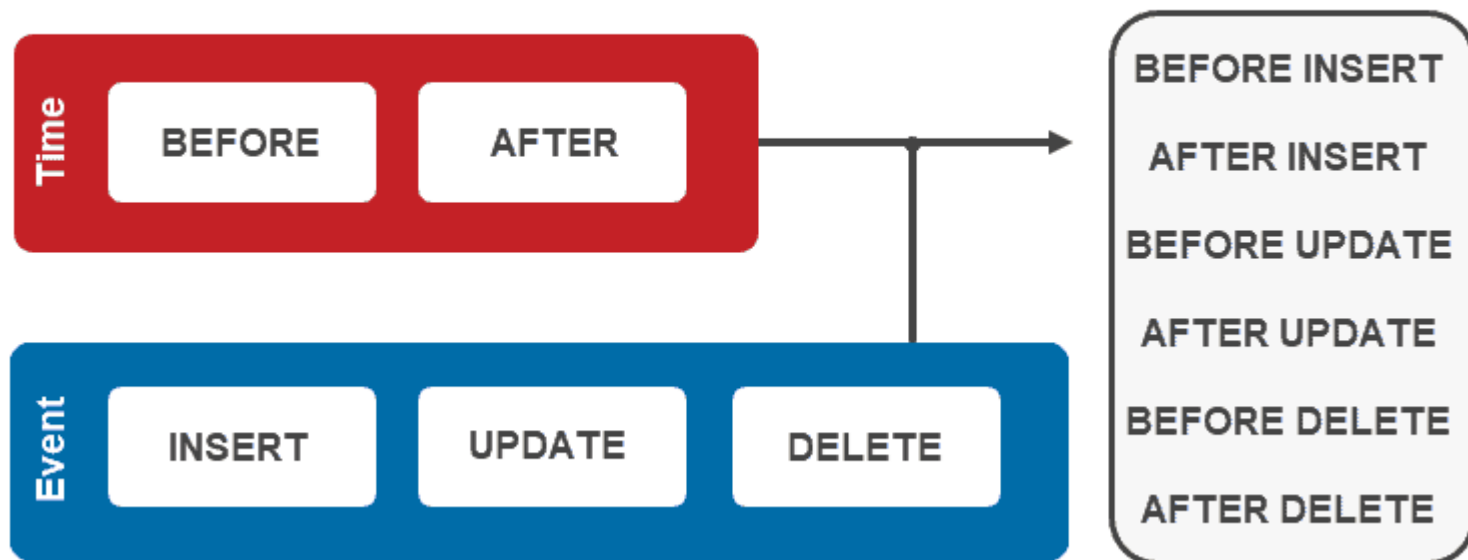
delimiter ;

call store_variables;

	sum_amount
▶	671000

TRIGGERS

MySQL Triggers



A database trigger is a stored program which is automatically fired or executed when some events occur.

Types of Trigger

- **Row Level Trigger**
- **Statement Level Trigger**

Row level Trigger -A event is triggered at row level for each row updated, inserted or deleted.

Statement Level trigger -An event is triggered at table Level for each sql statement executed.

THANK YOU