

# MYSQL PRESENTATION

BY  
KRISHNAMOORTH  
K

# MY SQL DATABASE

- SQL = Structured Query Language Program
- SQL Programming language used to manage and manipulate relational databases.
- Data is organized into tables with rows and columns.
- It provides efficient storage and retrieval of structured data.
- It is commonly used in various applications and systems.

# MY SQL WORKBENCH

- My SQL Workbench is a visual tool for database developers
- It provides data modeling, SQL development, and comprehensive tools for server configuration, user administration, backup.

My SQL WORKBENCH 8.0 CE.....



# SQL SERVERS

- Microsoft SQL Server
- My SQL Server  
Workbench
- Navigation
- Oracle Database
- IBM Db2



# Difference Between SQL & MYSQL

## SQL

- It as a Structured Query Language
- Not a specific database system, but a language standard
- Widely used in various database systems and platforms
- Used for database management across different platforms and systems.

## MYSQL

- SQL is based on ANSI SQL Standard
- A specific relational database management systems (RDBMS)
- One of the most popular open-source RDBMS, especially for web applications.
- Primarily used as the backend database for web applications, especially those using PHP

# DATATYPE OF MYSQL

- String Data type
- Timestamp Data type
- Date Time Data type
- Integers Data type
- Numeric Data type

# KEYS IN DBMS

- Super Key
- Candidate Key
- Primary 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, mail\_id,)
- Duplicate can allow



# CANDIDATE KEY

- Candidate keys are a subset of super key
- No repeated attributes
- ( Ex: student\_id, roll\_no)

## ALTERNATE KEY OR SECONDARY KEY

- Primary Key – Candidate Key

# FOREIGN KEY

- A Foreign Key is a reference key.
- It used to linked two tables together.
- It maintain relationship between two tables.

# CONTENTS

- MY SQL General Commands
- MY SQL General Function
- MY SQL String Functions
- MY SQL Date Functions
- MY SQL Calculate Functions
- MY SQL Logical Functions
- MY SQL Joins
- MY Stored Procedure
- MYSQL Triggers

# MYSQL 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 TABALE** - creates a new table
- **ALTER DATABASE** - modifies a database
- **DROP TABLE** - deletes a table

# MYSQL GENERAL COMMANDS

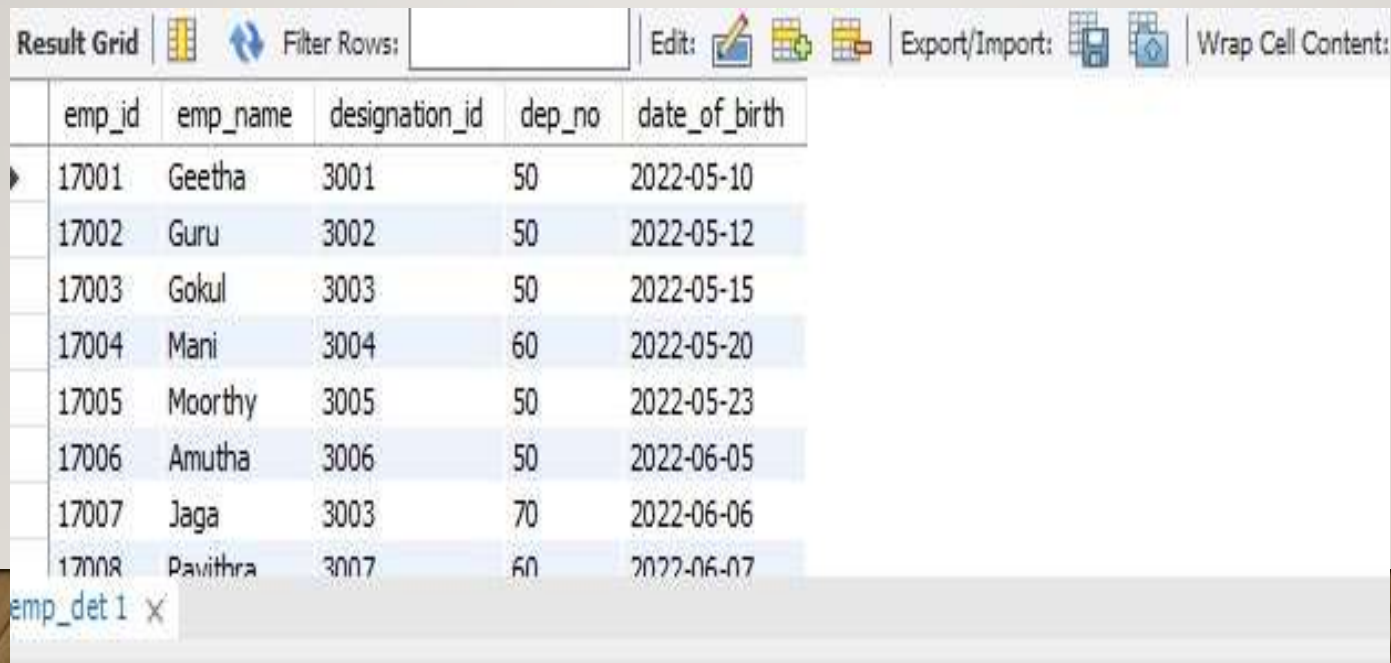
- **DDL** - DATA DEFINITION LANGUAGE
- **DML** - DATA MANIPULATION LANGUAGE
- **DQL** - DATA QUERY LANGUAGE
- **DCL** - DATA CONTROL LANGUAGE
- **TCL** - TRANSACTION CONTROL LANGUAGE

DDL	DML	DQL	DCL	TCL
<ul style="list-style-type: none"><li>▪ <b>Create</b></li><li>▪ <b>Alter</b></li><li>▪ <b>Drop</b></li><li>▪ <b>Truncate</b></li><li>▪ <b>Rename</b></li></ul>	<ul style="list-style-type: none"><li>▪ <b>Select</b></li><li>▪ <b>Insert</b></li><li>▪ <b>Update</b></li><li>▪ <b>Delete</b></li><li>▪ <b>Merge</b></li></ul>	<ul style="list-style-type: none"><li>▪ <b>Select</b></li><li>▪ <b>From</b></li></ul>	<ul style="list-style-type: none"><li>▪ <b>Grant</b></li><li>▪ <b>Revoke</b></li></ul>	<ul style="list-style-type: none"><li>▪ <b>Commit</b></li><li>▪ <b>Rollback</b></li><li>▪ <b>Save point</b></li></ul>

# TABLE CREATIONS

## TABLE I

- create table emp\_det (emp\_id int, emp\_name varchar(45), designation\_id int, dep\_no int, date\_of\_birth date, primary key(emp\_id));



The screenshot shows a database application interface. At the top, there is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. Below the toolbar is a table with the following data:

emp_id	emp_name	designation_id	dep_no	date_of_birth
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

At the bottom left, there is a tab labeled 'emp\_det 1' with a close button (X).



## TABLE 2

- **QUERY** : create table salary\_det(salary\_id int,emp\_date int,salary\_date date,branch\_id int,amount int, primary key (salary\_id));

**Output :**

emp_id	emp_name	designation_id	dep_no	date_of_birth
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

emp\_det 1 x



## TABLE 3

```
create table designation_det(designation_id int,designation varchar(45),primary
key(designation_id));
```

Result Grid						Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
emp_id	emp_name	designation_id	dep_no	date_of_birth					
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					

# TABLL 4

```
create table salary_det(salary_id int,emp_date int,salary_date date,branch_id
int,amount int, primary key (salary_id));
```

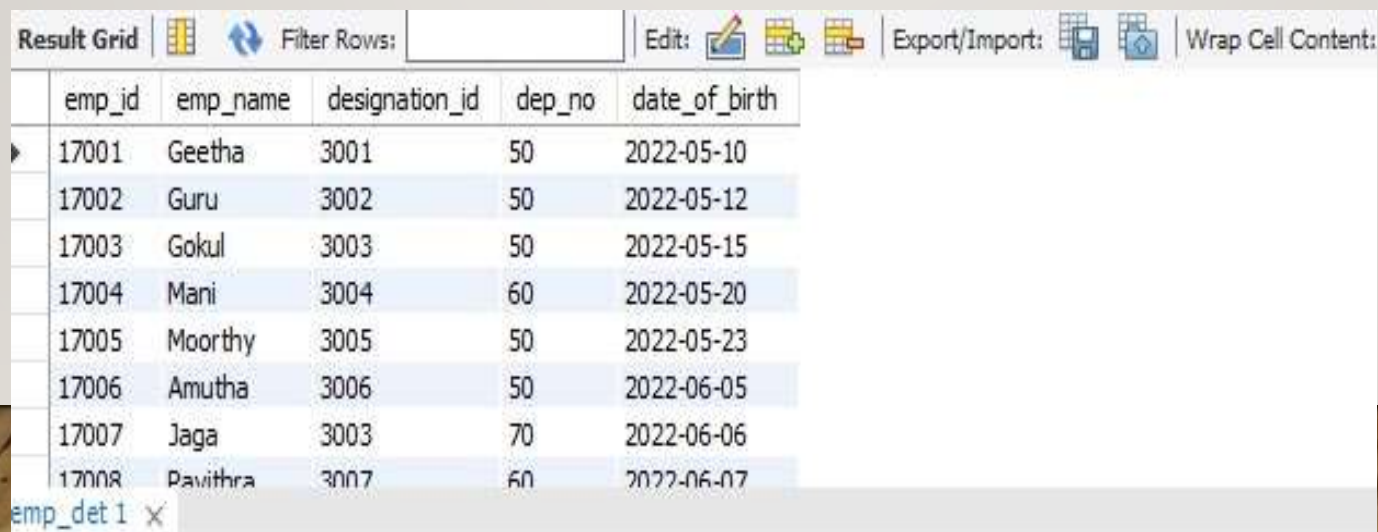
Result Grid					
Filter Rows:		Edit:		Export/Import:	
emp_id	emp_name	designation_id	dep_no	date_of_birth	
17001	Geetha	3001	50	2022-05-10	
17002	Guru	3002	50	2022-05-12	
17003	Gokul	3003	50	2022-05-15	
17004	Mani	3004	60	2022-05-20	
17005	Moorthy	3005	50	2022-05-23	
17006	Amutha	3006	50	2022-06-05	
17007	Jaga	3003	70	2022-06-06	
17008	Pavithra	3007	60	2022-06-07	

# MYSQL GENERAL FUNCTION

- where
- or
- and
- in
- not in
- >
- <
- >=
- <=
- <> (not in)
- !
- count
- Distinct
- count with discount
- order by Asc
- order by Desc
- 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 fulfil a specified condition.
- **QUERY:** select \* from emp\_det where designation\_id=3005;
- **OUTPUT**

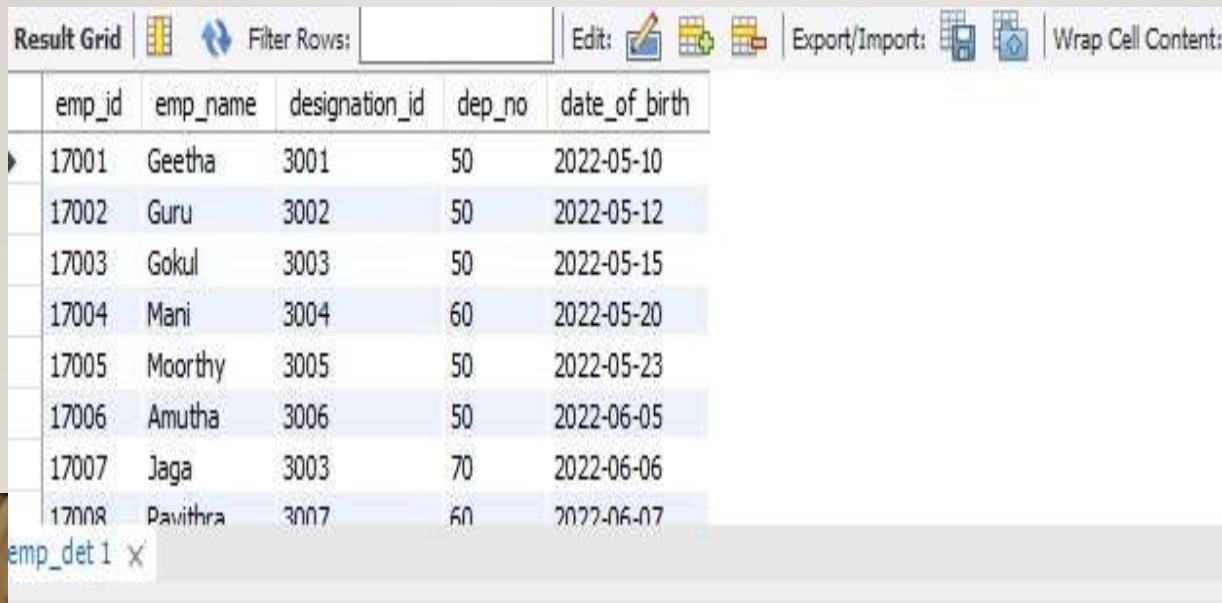


emp_id	emp_name	designation_id	dep_no	date_of_birth
17001	Geetha	3001	50	2022-05-10
17002	Guru	3002	50	2022-05-12
17003	Gokul	3003	50	2022-05-15
17004	Mani	3004	60	2022-05-20
17005	Moorthy	3005	50	2022-05-23
17006	Amutha	3006	50	2022-06-05
17007	Jaga	3003	70	2022-06-06
17008	Pavithra	3007	60	2022-06-07

emp\_det 1 x

# OR

- The OR operator displays a record if any of the conditions separated by OR is TRUE
- **QUERY:** select\* from emp\_det where dep\_no=50 or dep\_no=80;
- **OUTPUT**

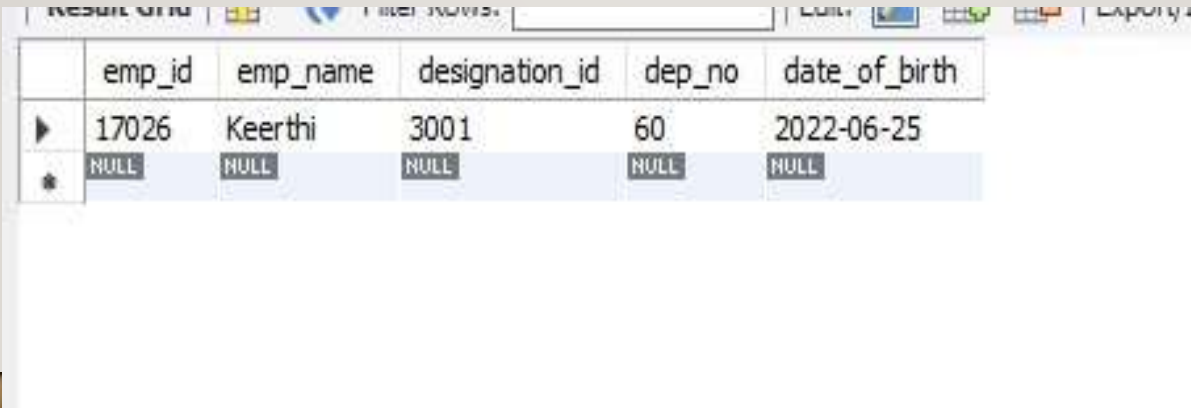


emp_id	emp_name	designation_id	dep_no	date_of_birth
17001	Geetha	3001	50	2022-05-10
17002	Guru	3002	50	2022-05-12
17003	Gokul	3003	50	2022-05-15
17004	Mani	3004	60	2022-05-20
17005	Moorthy	3005	50	2022-05-23
17006	Amutha	3006	50	2022-06-05
17007	Jaga	3003	70	2022-06-06
17008	Pavithra	3007	60	2022-06-07



# AND

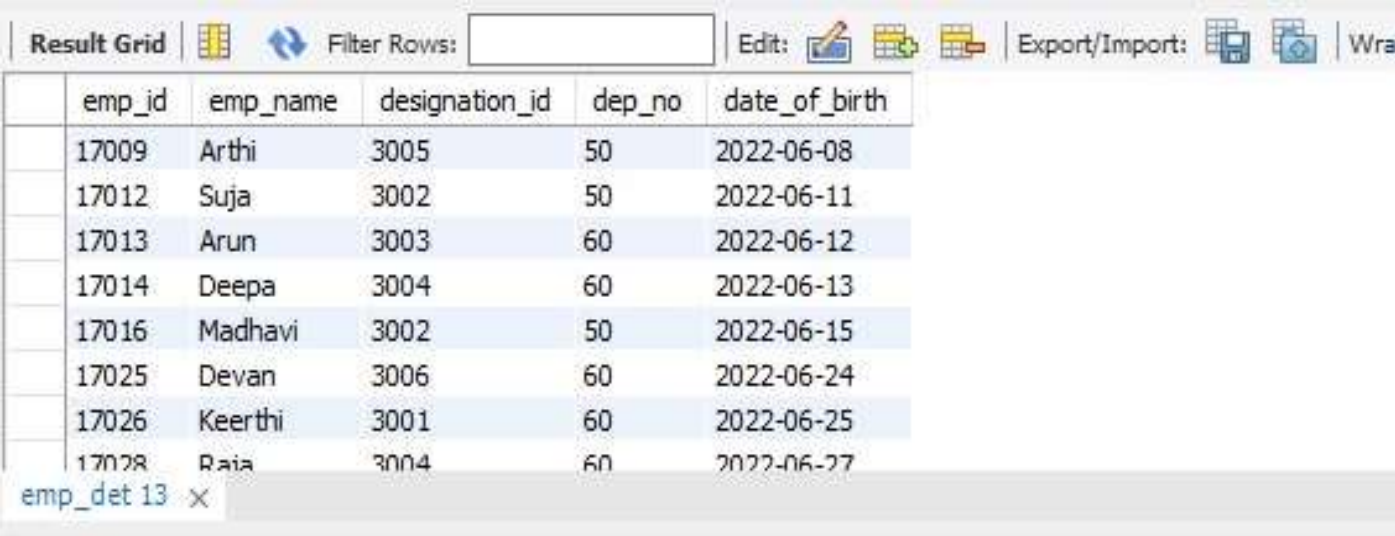
- The AND operator displays a record if all the conditions separated by AND are TRUE
- **QUERY** : `select * from emp_det where designation_id=3001 and dep_no=60;`
- **OUTPUT**



	emp_id	emp_name	designation_id	dep_no	date_of_birth
▶	17026	Keerthi	3001	60	2022-06-25
✱	NULL	NULL	NULL	NULL	NULL

# IN

- The IN operator allows you to specify multiple values in a WHERE clause.
- **QUERY** : select \*from emp\_det where dep\_no in (50,60);
- **OUTPUT**



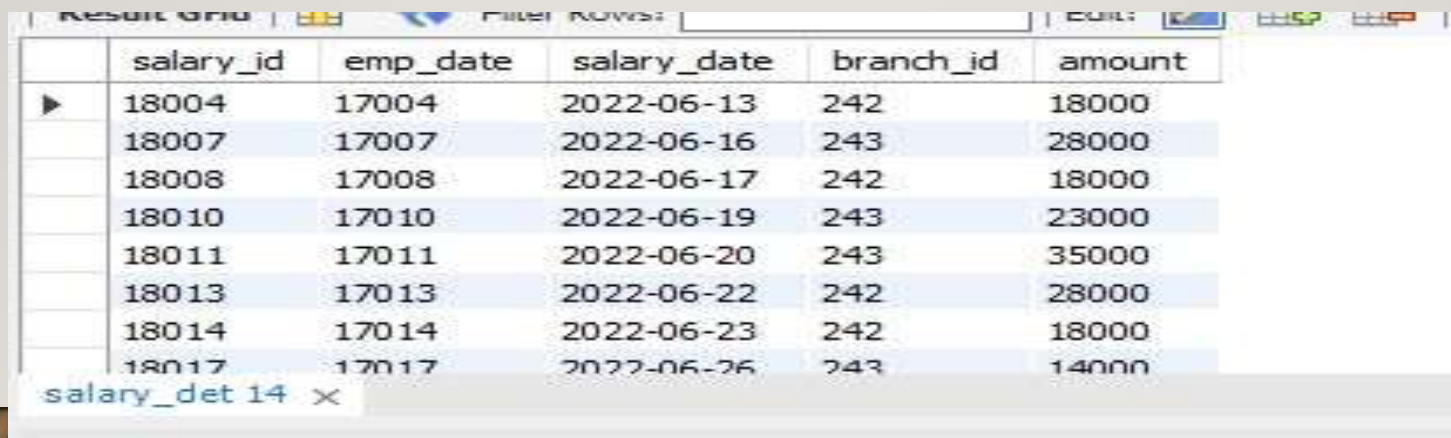
	emp_id	emp_name	designation_id	dep_no	date_of_birth
	17009	Arthi	3005	50	2022-06-08
	17012	Suja	3002	50	2022-06-11
	17013	Arun	3003	60	2022-06-12
	17014	Deepa	3004	60	2022-06-13
	17016	Madhavi	3002	50	2022-06-15
	17025	Devan	3006	60	2022-06-24
	17026	Keerthi	3001	60	2022-06-25
	17028	Raia	3004	60	2022-06-27

emp\_det 13 x



# NOT IN

- The NOT IN operators does not allows you to specify multiple values in a WHERE clause
- **QUERY** : `select* from salary_det where branch_id not in (241,244);`
- **OUTPUT**



The screenshot shows a database query result grid with the following columns: salary\_id, emp\_date, salary\_date, branch\_id, and amount. The results are filtered by branch\_id not in (241, 244). The table contains 9 rows of data.

	salary_id	emp_date	salary_date	branch_id	amount
▶	18004	17004	2022-06-13	242	18000
	18007	17007	2022-06-16	243	28000
	18008	17008	2022-06-17	242	18000
	18010	17010	2022-06-19	243	23000
	18011	17011	2022-06-20	243	35000
	18013	17013	2022-06-22	242	28000
	18014	17014	2022-06-23	242	18000
	18017	17017	2022-06-26	243	14000

salary\_det 14 x

# GREATER THAN

- The GREATER THAN operator is used to show the higher values
- **QUERY** : select\*from salary\_det where amount >25000;
- **OUTPUT**

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
salary_id	emp_date	salary_date	branch_id	amount
18001	17001	2022-06-10	241	35000
18003	17003	2022-06-12	241	28000
18005	17005	2022-06-14	241	30000
18007	17007	2022-06-16	243	28000
18009	17009	2022-06-18	241	30000
18011	17011	2022-06-20	243	35000
18013	17013	2022-06-22	242	28000
18015	17015	2022-06-24	244	30000

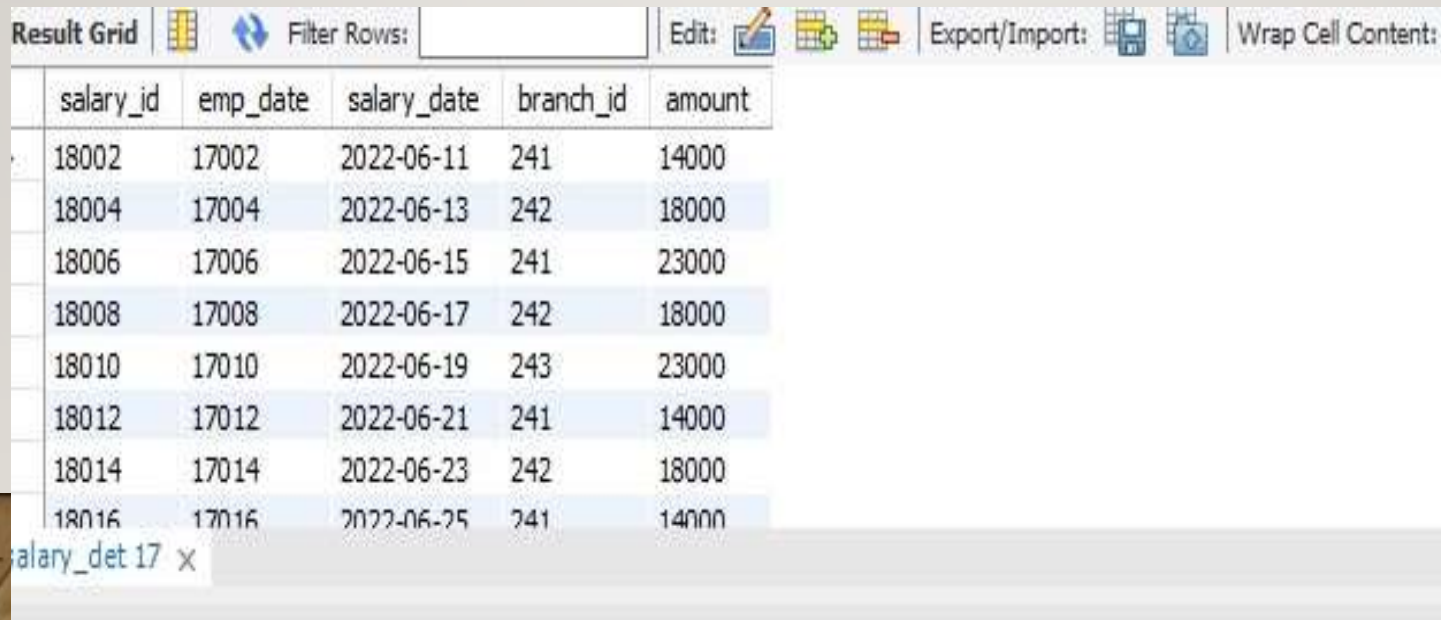
salary\_det 16 x

# LESSER THAN

The LESSER THAN operator is used to show the lower values.

- **QUERY** : `Select*from salary_det where amount <25000;`

- **OUTPUT**



The screenshot shows a database query result grid with a toolbar at the top. The toolbar includes icons for 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. The table below displays the results of the query, with columns for salary\_id, emp\_date, salary\_date, branch\_id, and amount. The data is filtered to show only records where the amount is less than 25000.

salary_id	emp_date	salary_date	branch_id	amount
18002	17002	2022-06-11	241	14000
18004	17004	2022-06-13	242	18000
18006	17006	2022-06-15	241	23000
18008	17008	2022-06-17	242	18000
18010	17010	2022-06-19	243	23000
18012	17012	2022-06-21	241	14000
18014	17014	2022-06-23	242	18000
18016	17016	2022-06-25	241	14000

salary\_det 17 x

# GREATER EQUAL

- The GREATER EQUAL is used to show the higher value and also the equal to values
- **QUERY** : `select*from salary_det where amount >=25000;`
- **OUTPUT**

	salary_id	emp_date	salary_date	branch_id	amount
▶	18001	17001	2022-06-10	241	35000
	18003	17003	2022-06-12	241	28000
	18005	17005	2022-06-14	241	30000
	18007	17007	2022-06-16	243	28000
	18009	17009	2022-06-18	241	30000
	18011	17011	2022-06-20	243	35000
	18013	17013	2022-06-22	242	28000
	18015	17015	2022-06-24	244	30000

salary\_det 18 x

# LESSER EQUAL

- 
- **QUERY** : select\*from salary\_det where amount <=25000;
- **OUTPUT**

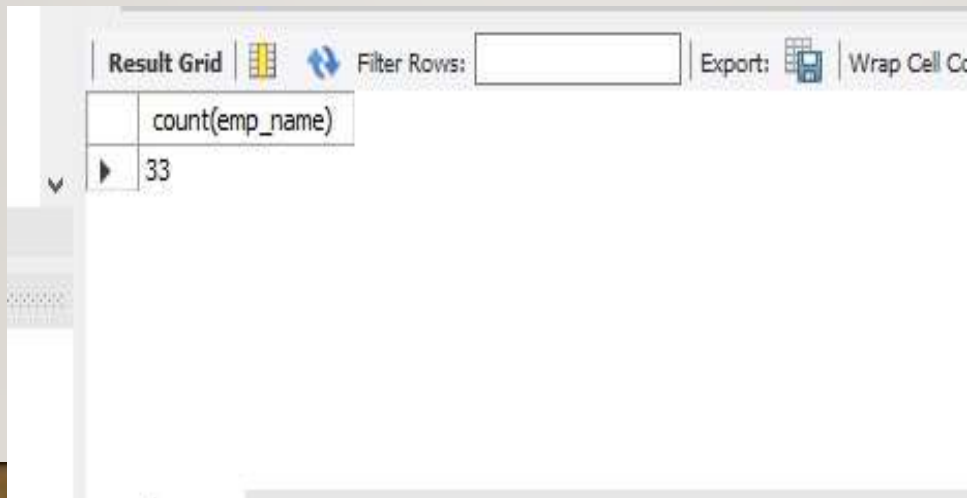
Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
salary_id	emp_date	salary_date	branch_id	amount
18002	17002	2022-06-11	241	14000
18004	17004	2022-06-13	242	18000
18006	17006	2022-06-15	241	23000
18008	17008	2022-06-17	242	18000
18010	17010	2022-06-19	243	23000
18012	17012	2022-06-21	241	14000
18014	17014	2022-06-23	242	18000
18016	17016	2022-06-25	241	14000

salary\_det 19 x



# COUNT

- The COUNT() functions returns the number of rows that matches a specified criterion
- **QUERY** : select count(emp\_name) from emp\_det;
- **OUTPUT**



The screenshot shows a database query result window. At the top, there is a toolbar with options like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar, a table displays the query results. The table has one column labeled 'count(emp\_name)' and one row with the value '33'.

count(emp_name)
33

# DISTINCT

- To SELECT DISTINCT statement is used to return only DISTINCT (different) values
- **QUERY** : select distinct emp\_name from emp\_det;
- **OUTPUT**




The screenshot shows a database query result grid. The grid has a header row with the column name 'emp\_name'. Below the header, there are ten rows of employee names: Geetha, Guru, Gokul, Mani, Moorthy, Amutha, Jaga, and Pavithra. The names are listed in descending order of length. The grid is titled 'Result Grid' and has a 'Filter Rows' button. The status bar at the bottom indicates 'emp\_det 21'.

emp_name
Geetha
Guru
Gokul
Mani
Moorthy
Amutha
Jaga
Pavithra



# COUNT WITH DISTINCT

- This is used to find the count values of DISTINCT values
- **Query** : `select count(distinct dep_name) from department_det7;`
- **OUTPUT**



The screenshot shows a database interface with a 'Result Grid' tab. The grid contains one column with the header 'count(distinct dep\_name)' and one row with the value '4'. Above the grid, there is a 'Filter Rows:' input field and an 'Export' button.

	count(distinct dep_name)
▶	4

# ORDER BY ASCENDING

- Select \* from salary\_details where amount between 10000 and 20000
- **QUERY** : select\* from salary\_det order by amount asc;
- **OUTPUT**



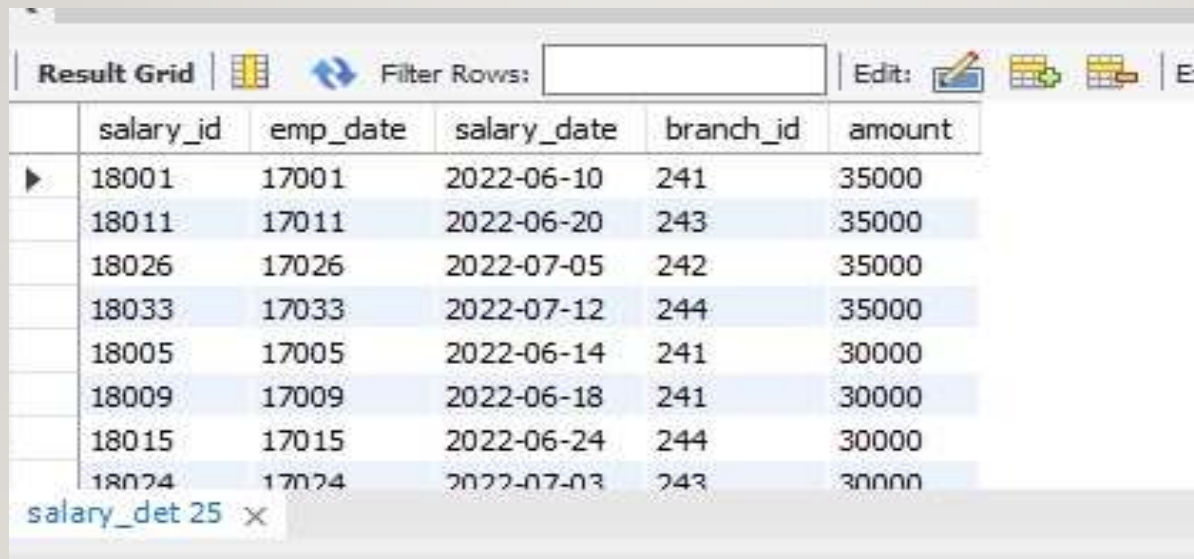
The screenshot shows a database query result grid. The grid has a toolbar at the top with icons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export'. The 'Filter Rows' field is empty. The table contains 9 rows of data, sorted by the 'amount' column in ascending order. The columns are 'salary\_id', 'emp\_date', 'salary\_date', 'branch\_id', and 'amount'.

	salary_id	emp_date	salary_date	branch_id	amount
	18021	17021	2022-06-30	244	14000
	18022	17022	2022-07-01	244	14000
	18023	17023	2022-07-02	244	14000
	18004	17004	2022-06-13	242	18000
	18008	17008	2022-06-17	242	18000
	18014	17014	2022-06-23	242	18000
	18028	17028	2022-07-07	242	18000
	18006	17006	2022-06-15	241	23000

salary\_det 24 x

# ORDER BY DESCENDING

- **QUERY** : `select* from salary_det order by amount desc;`
- **OUTPUT**



The screenshot shows a database application window titled "Result Grid". It contains a table with 6 columns: salary\_id, emp\_date, salary\_date, branch\_id, and amount. The data is sorted by the 'amount' column in descending order. The first four rows have an amount of 35000, and the next four rows have an amount of 30000. The table is displayed with alternating light blue and white row backgrounds. The window also includes a "Filter Rows" input field and an "Edit" button.

	salary_id	emp_date	salary_date	branch_id	amount
▶	18001	17001	2022-06-10	241	35000
	18011	17011	2022-06-20	243	35000
	18026	17026	2022-07-05	242	35000
	18033	17033	2022-07-12	244	35000
	18005	17005	2022-06-14	241	30000
	18009	17009	2022-06-18	241	30000
	18015	17015	2022-06-24	244	30000
	18024	17024	2022-07-03	243	30000

salary\_det 25 x

# 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 emp_name,count(emp_id)from emp_det group by emp_name;`

- **OUTPUT**

	emp_name	count(emp_id)
	Pandian	1
	Veera	1
	Devi	1
	Devan	1
	Keerthi	1
	Venkatesh	1
	Raja	1
	Priva	1





# DESC LIMIT

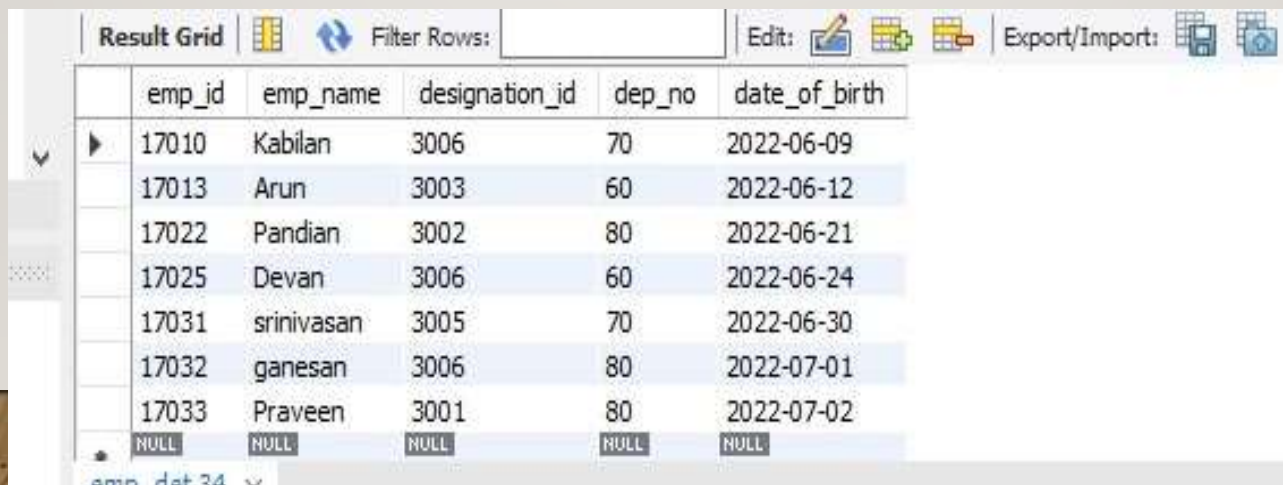
- **QUERY** : `select*from emp_det order by emp_id desc limit 20,5;`
- **OUTPUT**

	emp_id	emp_name	designation_id	dep_no	date_of_birth
▶	17013	Arun	3003	60	2022-06-12
	17012	Suja	3002	50	2022-06-11
	17011	Manasi	3001	70	2022-06-10
	17010	Kabilan	3006	70	2022-06-09
	17009	Arthi	3005	50	2022-06-08
★	NULL	NULL	NULL	NULL	NULL

emp\_det 31 x

# LIKE

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



The screenshot shows a database application interface with a 'Result Grid' tab. The grid displays a list of employees with columns for emp\_id, emp\_name, designation\_id, dep\_no, and date\_of\_birth. The data is as follows:

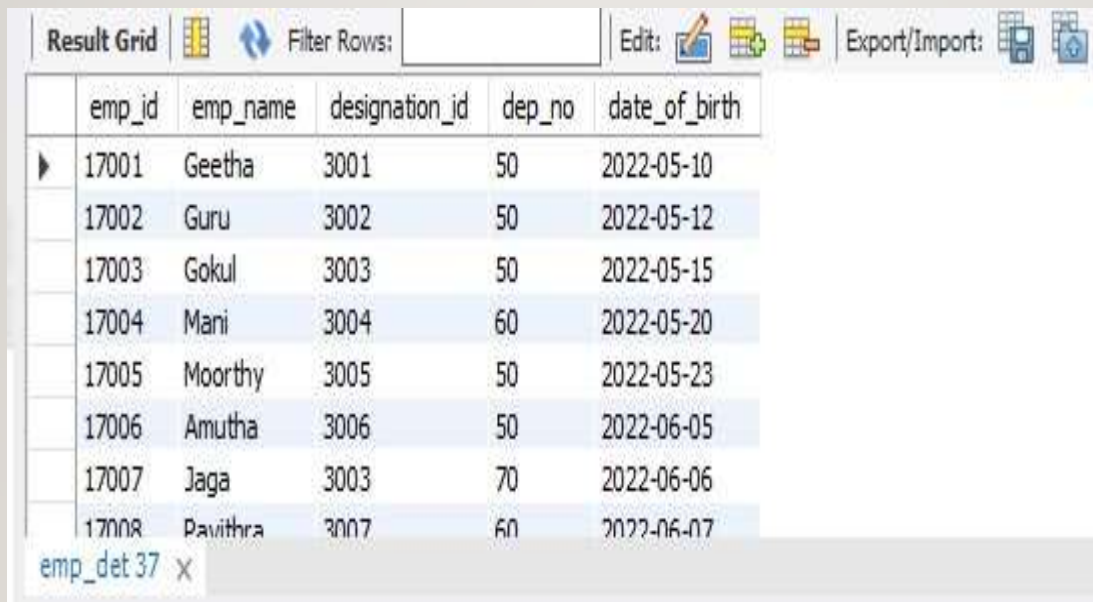
emp_id	emp_name	designation_id	dep_no	date_of_birth
17010	Kabilan	3006	70	2022-06-09
17013	Arun	3003	60	2022-06-12
17022	Pandian	3002	80	2022-06-21
17025	Devan	3006	60	2022-06-24
17031	srinivasan	3005	70	2022-06-30
17032	ganesan	3006	80	2022-07-01
17033	Praveen	3001	80	2022-07-02
NULL	NULL	NULL	NULL	NULL

The interface also includes a 'Filter Rows' field, an 'Edit' button, and an 'Export/Import' button. The table is titled 'emp\_det 34'.



# NOT LIKE

- **QUERY** : `select*from emp_det where emp_name not like'%n';`
- **OUTPUT**



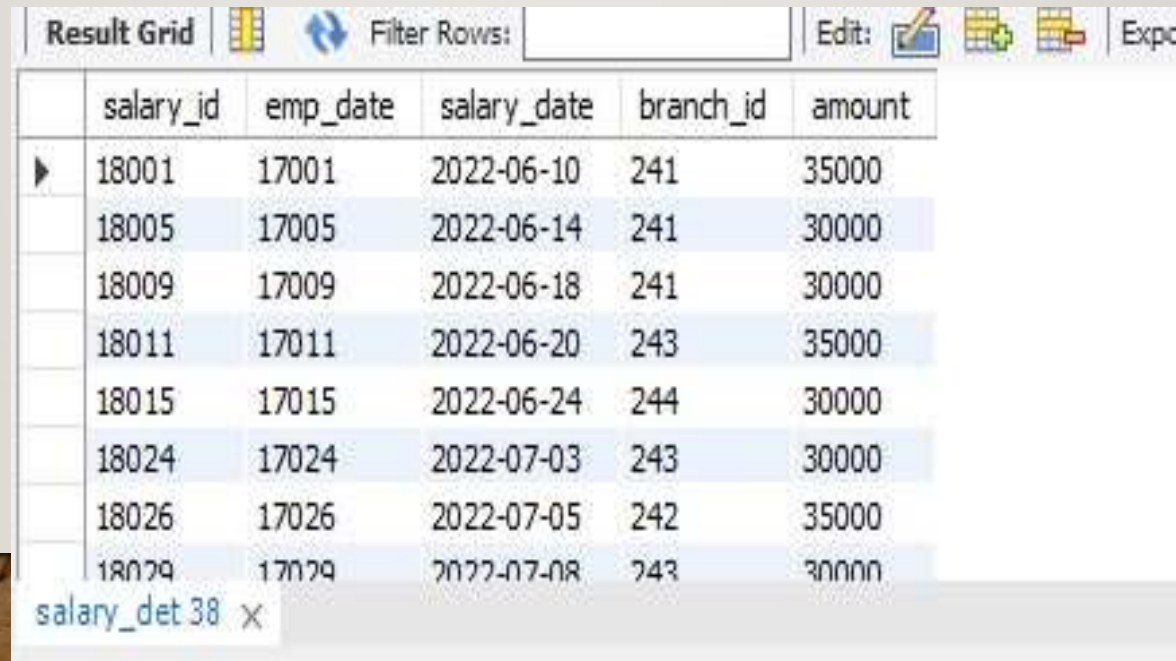
The screenshot shows a database query result grid with a toolbar at the top. The toolbar includes buttons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Import'. The grid displays a table with 6 columns: emp\_id, emp\_name, designation\_id, dep\_no, and date\_of\_birth. The data is as follows:

	emp_id	emp_name	designation_id	dep_no	date_of_birth
▶	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

At the bottom left of the grid, there is a tab labeled 'emp\_det 37' with a close button 'x'.

# BETWEEN AND

- The BETWEEN operator select values within a given range. The values can be numbers, text, or dates.
- **QUERY** : select\* from salary\_det where amount between 30000 and 40000;
- **OUTPUT**



The screenshot shows a database query result grid with the following columns: salary\_id, emp\_date, salary\_date, branch\_id, and amount. The results are filtered to show only records where the amount is between 30000 and 40000. The grid includes a toolbar with icons for filtering, editing, and exporting. A status bar at the bottom indicates that 38 rows are displayed.

	salary_id	emp_date	salary_date	branch_id	amount
▶	18001	17001	2022-06-10	241	35000
	18005	17005	2022-06-14	241	30000
	18009	17009	2022-06-18	241	30000
	18011	17011	2022-06-20	243	35000
	18015	17015	2022-06-24	244	30000
	18024	17024	2022-07-03	243	30000
	18026	17026	2022-07-05	242	35000
	18029	17029	2022-07-08	243	30000

salary\_det 38 x

# MYSQL STRING FUNCTION

➤ LCase

➤ UCase

➤ Left

➤ Right

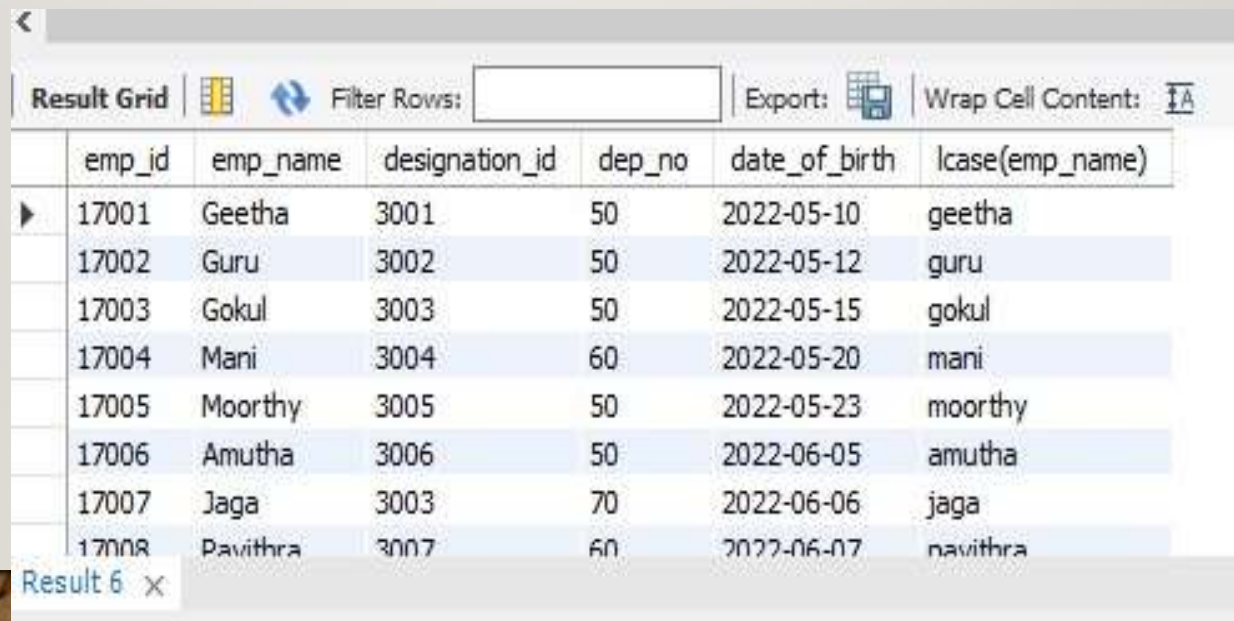
➤ Concat

➤ Trim

➤ Char\_Length

# LOWER CASE

- **QUERY** : `select*,lcase(emp_name)from emp_det;`
- **Output**



The screenshot shows a database query result grid. The grid has a toolbar at the top with icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar is a table with 7 columns: emp\_id, emp\_name, designation\_id, dep\_no, date\_of\_birth, and lcase(emp\_name). The table contains 8 rows of data. The first row is highlighted with a blue background. The second row is highlighted with a light blue background. The third row is highlighted with a light blue background. The fourth row is highlighted with a light blue background. The fifth row is highlighted with a light blue background. The sixth row is highlighted with a light blue background. The seventh row is highlighted with a light blue background. The eighth row is highlighted with a light blue background. The table is titled 'Result 6'.

	emp_id	emp_name	designation_id	dep_no	date_of_birth	lcase(emp_name)
▶	17001	Geetha	3001	50	2022-05-10	geetha
	17002	Guru	3002	50	2022-05-12	guru
	17003	Gokul	3003	50	2022-05-15	gokul
	17004	Mani	3004	60	2022-05-20	mani
	17005	Moorthy	3005	50	2022-05-23	moorthy
	17006	Amutha	3006	50	2022-06-05	amutha
	17007	Jaga	3003	70	2022-06-06	jaga
	17008	Pavithra	3007	60	2022-06-07	pavithra

# UPPER CASE

- **QUERY** : `select*,ucase(emp_name)from emp_det;`
- **Output**

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

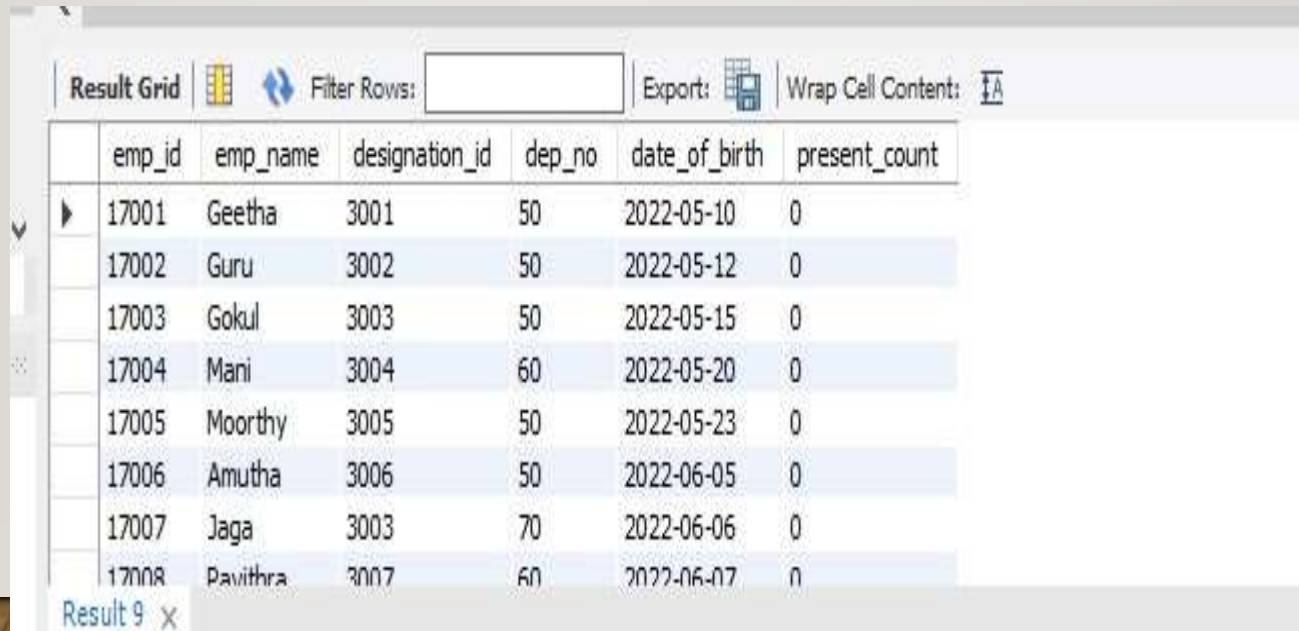
	emp_id	emp_name	designation_id	dep_no	date_of_birth	ucase(emp_name)
▶	17001	Geetha	3001	50	2022-05-10	GEETHA
	17002	Guru	3002	50	2022-05-12	GURU
	17003	Gokul	3003	50	2022-05-15	GOKUL
	17004	Mani	3004	60	2022-05-20	MANI
	17005	Moorthy	3005	50	2022-05-23	MOORTHY
	17006	Amutha	3006	50	2022-06-05	AMUTHA
	17007	Jaga	3003	70	2022-06-06	JAGA
	17008	Pavithra	3007	60	2022-06-07	PAVITHRA

Result 7 x



# TRIM

- **QUERY** : `select *,if (trim(emp_id) ='p','l','0') as present_count from emp_det;`
- **Output**



The screenshot shows a database query result grid. The grid has a toolbar at the top with options like 'Filter Rows', 'Export', and 'Wrap Cell Content'. The data is presented in a table with 7 columns: emp\_id, emp\_name, designation\_id, dep\_no, date\_of\_birth, and present\_count. There are 8 rows of data, each representing an employee. The 'present\_count' column contains values 0 for all employees. The table is titled 'Result 9' at the bottom.

	emp_id	emp_name	designation_id	dep_no	date_of_birth	present_count
▶	17001	Geetha	3001	50	2022-05-10	0
	17002	Guru	3002	50	2022-05-12	0
	17003	Gokul	3003	50	2022-05-15	0
	17004	Mani	3004	60	2022-05-20	0
	17005	Moorthy	3005	50	2022-05-23	0
	17006	Amutha	3006	50	2022-06-05	0
	17007	Jaga	3003	70	2022-06-06	0
	17008	Pavithra	3007	60	2022-06-07	0

Result 9 x

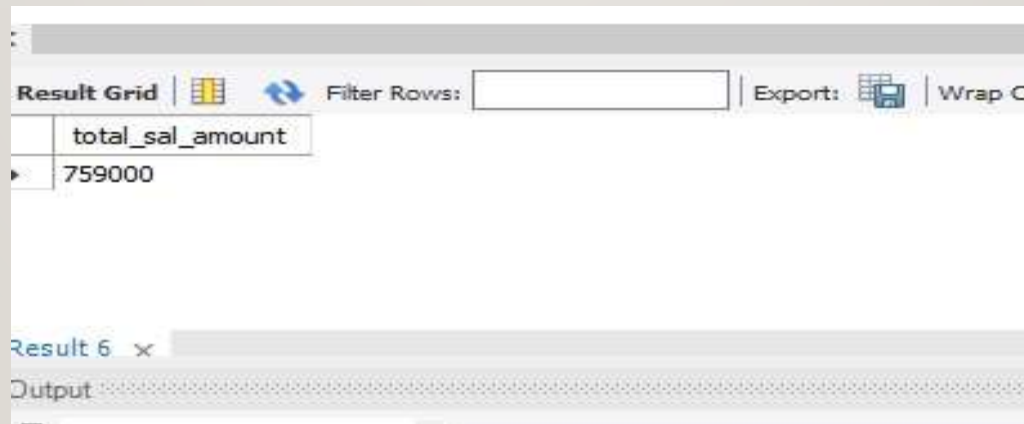


# MYSQL CALCULATE FUNCTION

- Sum
- Average
- Min
- Max
- Count

# SUM

- The SUM() function returns the total sum of a numeric column.
- **QUERY** : `select sum(amount) as total_sal_amount from salary_det;`
- **Output** :



The screenshot shows a database query result grid. The grid has two columns: 'total\_sal\_amount' and a value '759000'. The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, and an 'Export' button. Below the grid, there is a 'Result 6' tab and an 'Output' section.

total_sal_amount
759000

# AVERAGE

- The AVERAGE function returns the average value off a numeric column.
- **QUERY** : `select avg(amount)as total_sal_amount from salary_det;`
- **Output** :



The screenshot shows a database query result window. At the top, there is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar is a table with two columns: 'total\_sal\_amount' and a value '23000.0000'. The table is titled 'Result 16' and has a tab labeled 'Output'.

	total_sal_amount
▶	23000.0000

# MAX

- The MAX function returns the largest value of the selected column.
- **QUERY** : `select max(amount)as total_sal_amount from salary_det;`
- **Output :**



The screenshot shows a database query result window. At the top, there is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar, the query result is displayed in a table format. The table has one column named 'total\_sal\_amount' and one row with the value '35000'. The window title is 'Result 7'.

total_sal_amount
35000

# MIN

- The MIN function return the smallest value of the selected column.
- **QUERY** : select min(amount)as total\_sal\_amount from salary\_det;
- **Output :**



The screenshot shows a database query result window. At the top, there is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar, the query result is displayed in a table with one column named 'total\_sal\_amount' and one row containing the value '14000'. The window title bar at the bottom indicates 'Result 8'.

total_sal_amount
14000

# COUNT

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



The screenshot shows a database interface with a toolbar at the top containing icons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar is a table with one row and one column. The column header is 'count(emp\_id)' and the value in the row is '33'. At the bottom of the window, there is a tab labeled 'Result 9'.

count(emp_id)
33



# YEAR, MONTH, DATE

## ➤ YEAR

▪ **QUERY :** `select*,timestampdiff(year,date_of_birth,curdate())as date_of_birth  
from emp_det;`

▪ **Output :**

Result Grid						
		Filter Rows:			Export:	Wrap Cell Content: <a href="#">IA</a>
	emp_id	emp_name	designation_id	dep_no	date_of_birth	date_of_birth
▶	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
	17005	Moorthy	3005	50	2022-05-23	1
	17006	Amutha	3006	50	2022-06-05	1
	17007	Jaga	3003	70	2022-06-06	1
	17008	Pavithra	3007	60	2022-06-07	1

Result 33 x

# MONTH

➤ **QUERY** : `select*,timestampdiff(month,date_of_birth,curdate())as date_of_birth  
from emp_det;`

➤ **Output :**

	emp_id	emp_name	designation_id	dep_no	date_of_birth	date_of_birth
▶	17001	Geetha	3001	50	2022-05-10	22
	17002	Guru	3002	50	2022-05-12	22
	17003	Gokul	3003	50	2022-05-15	22
	17004	Mani	3004	60	2022-05-20	21
	17005	Moorthy	3005	50	2022-05-23	21
	17006	Amutha	3006	50	2022-06-05	21
	17007	Jaga	3003	70	2022-06-06	21
	17008	Pavithra	3007	60	2022-06-07	21

Result 32 ✕

# DAY

➤ **QUERY** : `select*,timestampdiff(day,date_of_birth,curdate())as date_of_birth from emp_det;`

➤ **Output :**

Result Grid						
		Filter Rows:		Export:	Wrap Cell Content:	
	emp_id	emp_name	designation_id	dep_no	date_of_birth	date_of_birth
▶	17001	Geetha	3001	50	2022-05-10	677
	17002	Guru	3002	50	2022-05-12	675
	17003	Gokul	3003	50	2022-05-15	672
	17004	Mani	3004	60	2022-05-20	667
	17005	Moorthy	3005	50	2022-05-23	664
	17006	Amutha	3006	50	2022-06-05	651
	17007	Jaga	3003	70	2022-06-06	650
	17008	Pavithra	3007	60	2022-06-07	649

Result 35 x

# MY SQL LOGICAL FUNCTION

- If
- Count If
- If With And Conditions
- If With Or Conditions

# IF CONDITIONS

- **QUERY** : `select*,if(dep_no<=60,'senior','jounior')as categroy from emp_det;`
- **Output** :

	emp_id	emp_name	designation_id	dep_no	date_of_birth	categroy
	17006	Amutha	3006	50	2022-06-05	senior
	17007	Jaga	3003	70	2022-06-06	jounior
	17008	Pavithra	3007	60	2022-06-07	senior
	17009	Arthi	3005	50	2022-06-08	senior
	17010	Kabilan	3006	70	2022-06-09	jounior

Result 15 x



# IF WITH AND CONDITIONS

➤ **QUERY** : `select*,if(dep_no<=60,'senior','jounior' and emp_id<=17010)as categroy`  
from emp\_det;

➤ **Output :**

emp_id	emp_name	designation_id	dep_no	date_of_birth	category
17016	Madhavi	3002	50	2022-06-15	senior
17017	Swetha	3002	70	2022-06-16	0
17018	Selvi	3002	70	2022-06-17	0
17019	Pooja	3002	70	2022-06-18	0
17020	Lakshmi	3002	70	2022-06-19	0



# MYSQL JOINS FUNCTIONS

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
- Types of Joins are :

INNER JOIN

LEFT JOIN

RIGHT JOIN



CROSS JOIN



# INNER JOIN

➤ **QUERY** : select \*from emp\_det inner join department\_det7 on emp\_det.dep\_no=department\_det7.dep\_no;

➤ **Output** :

Result Grid									
Filter Rows: <input type="text"/>									
Export:  Wrap Cell Content: 									
	emp_id	emp_name	designation_id	dep_no	date_of_birth	dep_no	dep_name	branch_id	branch_name
▶	17001	Geetha	3001	50	2022-05-10	50	Production Department	241	Annan Nagar
	17002	Guru	3002	50	2022-05-12	50	Production Department	241	Annan Nagar
	17003	Gokul	3003	50	2022-05-15	50	Production Department	241	Annan Nagar
	17004	Mani	3004	60	2022-05-20	60	HR Department	242	Velachery
	17005	Moorthy	3005	50	2022-05-23	50	Production Department	241	Annan Nagar
	17006	Amutha	3006	50	2022-06-05	50	Production Department	241	Annan Nagar
	17007	Jaga	3003	70	2022-06-06	70	Sales Department	243	Guindy
	17008	Pavithra	3007	60	2022-06-07	60	HR Department	242	Velachery


Result 25 x


# LEFT JOIN


➤ **QUERY** : select \*from emp\_det left join department\_det7 on emp\_det.dep\_no=department\_det7.dep\_no;

➤ **Output :**

Result Grid

 Filter Rows:

Export: 

Wrap Cell Content: 

	emp_id	emp_name	designation_id	dep_no	date_of_birth	dep_no	dep_name	branch_id	branch_name
▶	17001	Geetha	3001	50	2022-05-10	50	Production Department	241	Annan Nagar
	17002	Guru	3002	50	2022-05-12	50	Production Department	241	Annan Nagar
	17003	Gokul	3003	50	2022-05-15	50	Production Department	241	Annan Nagar
	17004	Mani	3004	60	2022-05-20	60	HR Department	242	Velachery
	17005	Moorthy	3005	50	2022-05-23	50	Production Department	241	Annan Nagar
	17006	Amutha	3006	50	2022-06-05	50	Production Department	241	Annan Nagar
	17007	Jaga	3003	70	2022-06-06	70	Sales Department	243	Guindy
	17008	Pavithra	3007	60	2022-06-07	60	HR Department	242	Velachery



Result 27 x


# RIGHT JOIN


➤ **QUERY** : select \*from department\_det7 right join emp\_det on department\_det7.dep\_no =emp\_det.dep\_no;

➤ **Output :**

Result Grid

  Filter Rows:

Export: 

Wrap Cell Content: 

	dep_no	dep_name	branch_id	branch_name	emp_id	emp_name	designation_id	dep_no	date_of_birth
▶	50	Production Department	241	Annan Nagar	17001	Geetha	3001	50	2022-05-10
	50	Production Department	241	Annan Nagar	17002	Guru	3002	50	2022-05-12
	50	Production Department	241	Annan Nagar	17003	Gokul	3003	50	2022-05-15
	60	HR Department	242	Velachery	17004	Mani	3004	60	2022-05-20
	50	Production Department	241	Annan Nagar	17005	Moorthy	3005	50	2022-05-23
	50	Production Department	241	Annan Nagar	17006	Amutha	3006	50	2022-06-05
	70	Sales Department	243	Guindy	17007	Jaga	3003	70	2022-06-06
	60	HR Department	242	Velachery	17008	Pavithra	3007	60	2022-06-07

Result 29 x



# UNION

- **QUERY :** (select \*from emp\_det left join department\_det7 on emp\_det.dep\_no=department\_det7.dep\_no)union(select \*from department\_det7 right join emp\_det on department\_det7.dep\_no =emp\_det.dep\_no);
- **Output :**

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

IA

	emp_id	emp_name	designation_id	dep_no	date_of_birth	dep_no	dep_name	branch_id	branch_name
▶	17001	Geetha	3001	50	2022-05-10	50	Production Department	241	Annan Nagar
	17002	Guru	3002	50	2022-05-12	50	Production Department	241	Annan Nagar
	17003	Gokul	3003	50	2022-05-15	50	Production Department	241	Annan Nagar
	17004	Mani	3004	60	2022-05-20	60	HR Department	242	Velachery
	17005	Moorthy	3005	50	2022-05-23	50	Production Department	241	Annan Nagar
	17006	Amutha	3006	50	2022-06-05	50	Production Department	241	Annan Nagar
	17007	Jaga	3003	70	2022-06-06	70	Sales Department	243	Guindy
	17008	Pavithra	3007	60	2022-06-07	60	HR Department	242	Velachery

Result 30

Output

# TRIGGERS IN SQL

- **Triggers creation** : A database trigger is a stored program which is automatically fired or executed when some events occur.

## TYPES OF TRIGGER

- **Row Level Trigger** :A event is triggered at low level for each row updated, inserted or deleted
- **Statement Level Trigger** :An event is triggered at table level for each SQL statement executed



# TRIGGERS TIMING:

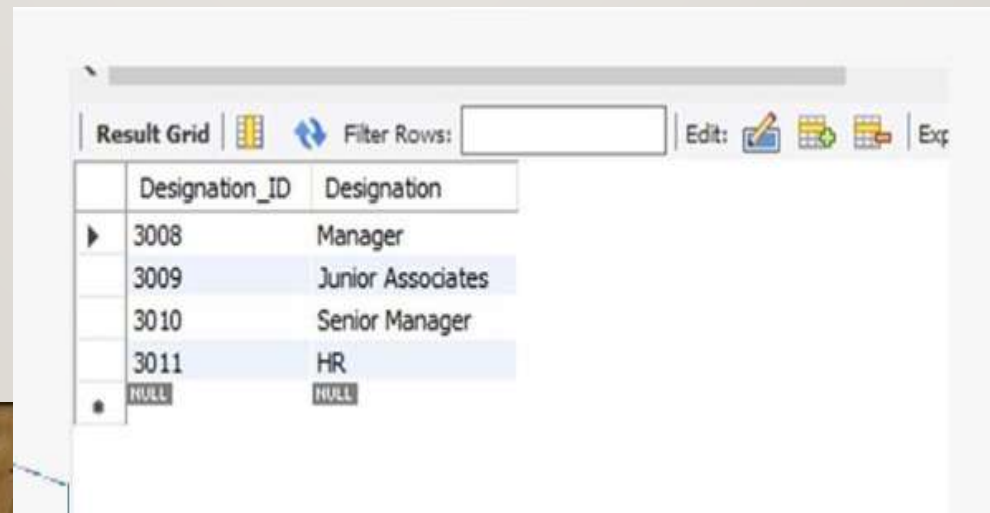
- Before Insert
- After Insert
- Before Update
- After Update
- Before delete
- After Delete

# AFTER INSERT

## ➤ QUERY:

```
delimiter //  
create trigger designation_update after insert on  
designation_detl for each row  
Begin  
insert into designation_backup(designation_id,designation) values  
(new.Designation_ID,new.Designation);  
end //  
delimiter ;
```

## ➤ Output :



The screenshot shows a database query result grid with the following data:

Designation_ID	Designation
3008	Manager
3009	Junior Associates
3010	Senior Manager
3011	HR
NULL	NULL

# BEFORE INSERT

## ➤ QUERY :

```
delimiter //  
create trigger dep_update before insert on dep_det for each row  
begin  
    if new.deplues (90,null ,242,'Tambaram'),(100,'Production  
Department',243,'Adaiyar') name is null then  
        set new.dep_name ="update your  dep_name";  
    end if ;  
end //  
Delimiter ;
```

## ➤ Output :

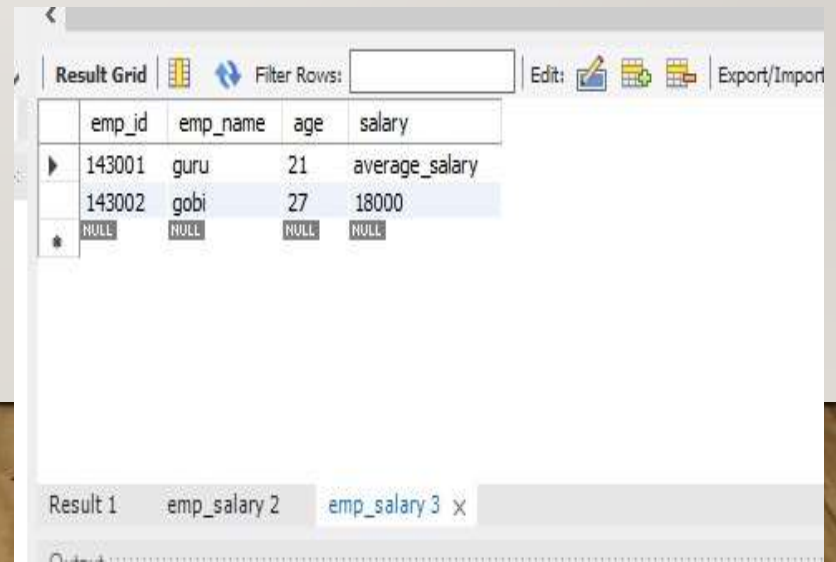
	Dep_NO	Dep_name	Branch_ID	Branch_Name
▶	50	Production Department	241	Annan Nagar
	60	HR Department	242	Velachery
	70	Sales Department	243	Guindy
	80	Finance Department	244	KMC
	90	update your dep_name	242	Tambaram
	100	Production Department	243	Adaiyar
*	NULL	NULL	NULL	NULL

dep\_det 43 x

# TRIGGER BEFORE UPDATE

## ➤ QUERY :

```
delimiter //  
create trigger salary_check before update on emp_salary for each row  
begin if new.salary>=40000 then  
set new.salary="high_salary";  
elseif new.salary>=35000 then  
set new.salary="good_salary";  
elseif new.salary>=15000 then  
set new.salary="average_salary";  
elseif new.salary>=0 then  
set new.salary="low_salary";  
  
end if;  
end //  
delimiter ;
```



emp_id	emp_name	age	salary
143001	guru	21	average_salary
143002	gobi	27	18000
NULL	NULL	NULL	NULL

*Thank  
you!*