MySQL -DQL

www.hexaware.com | © Hexaware Technologies. All rights reserved.

Course Objective

- To retrieve data from MySQL database
- To implement conditions while retrieving the data
- To implement basic functions and explore advance function in MySQL

Session Objective

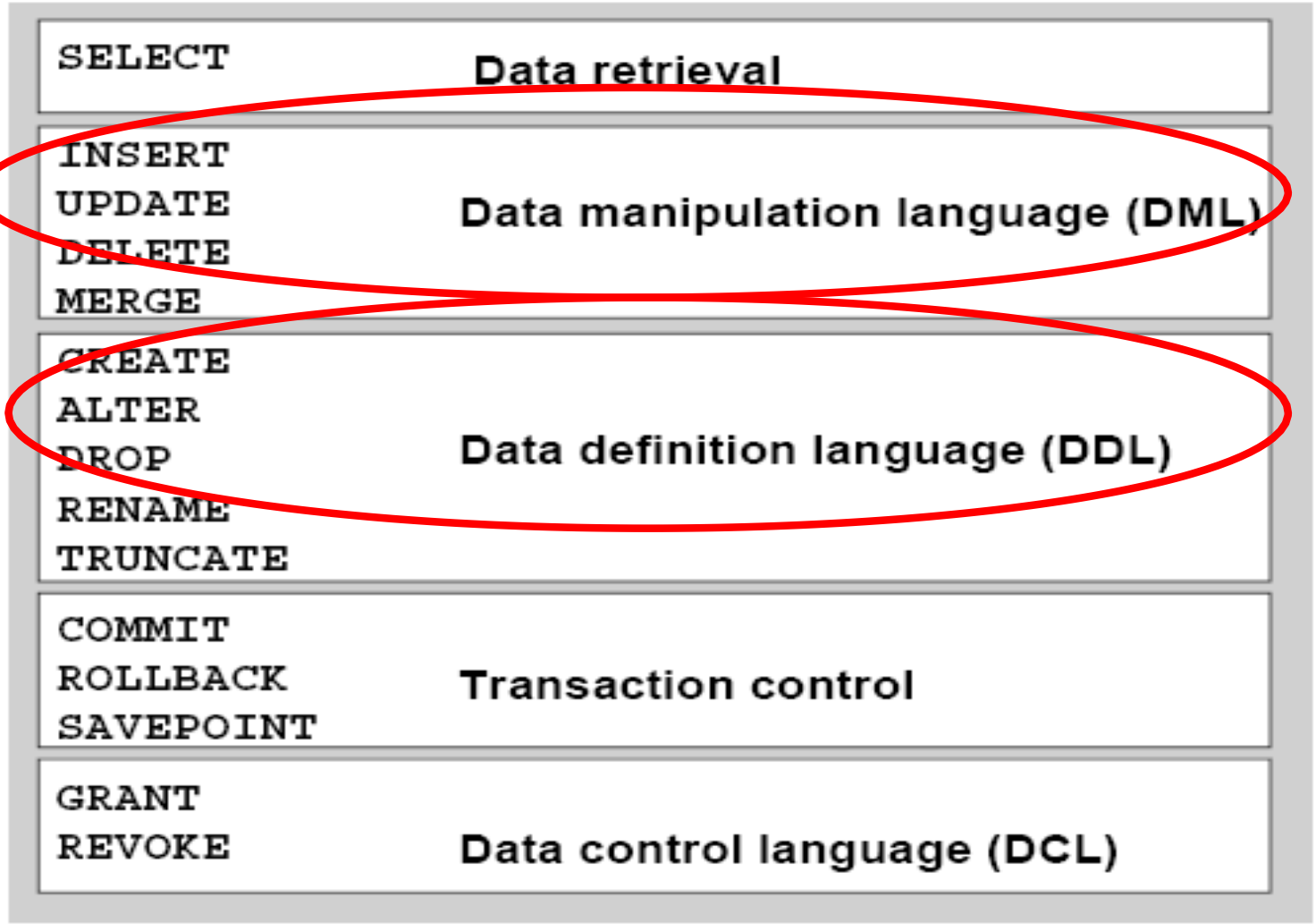
- DQL –Select
- Arithmetic operators
- Comparison conditions
- Order by clause
- Functions – Group functions
- Group by clause
- Having clause

SQL

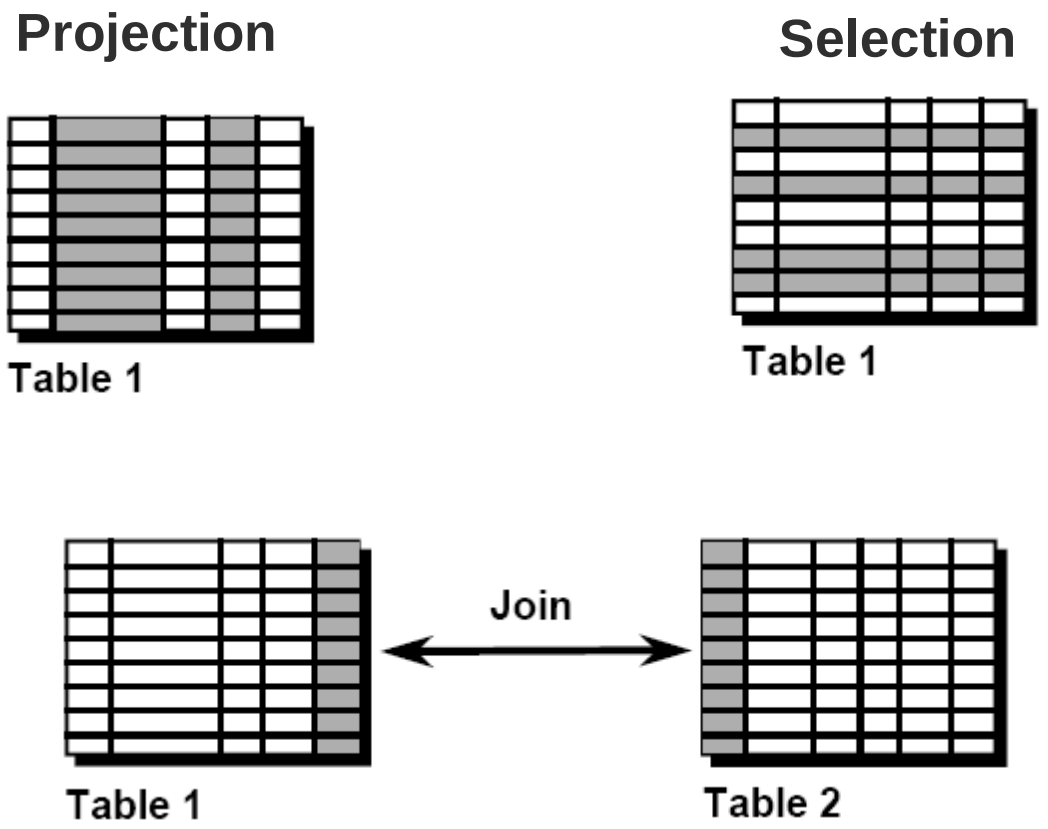
SQL

- SQL stands for Structured Query Language SQL allows you to access a database.
- SQL is an ANSI standard computer language SQL can execute queries against a database SQL can retrieve data from a database.
- SQL can insert new records in a database SQL can delete records from a database SQL can update records in a database

SQL Statements



Capabilities Of SQL Select



Writing SQL Statements

- SQL statements are NOT case sensitive.
- SQL statements can be on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.

SQL Select

Syntax

```
SELECT [DISTINCT|ALL ]{*|[columnExpression[AS
newName]][,…]}
FROM TableName[Aliase][,…]
[WHERE condition]
GROUP BY columnList][HAVING condition]
[ORDER BY columnList]
```

Projection Capability

Projection Capability:

- Used to choose the columns in a table that you want returned by your query.
- Can be used to choose as few or as many columns of the table as you require.

Examples:

```
SELECT *  FROM Dept;
```

```
SELECT distinct(deptid)
FROM emp;
```

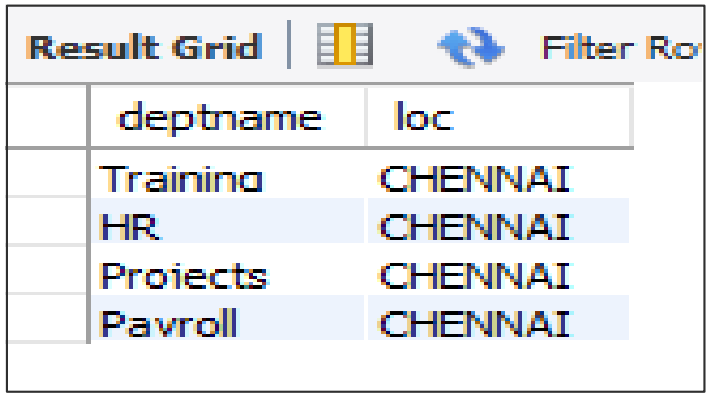
Result Grid | Filter F

deptid	deptname
101	Training
102	HR
103	Projects
104	Payroll
NULL	NULL

Result Grid

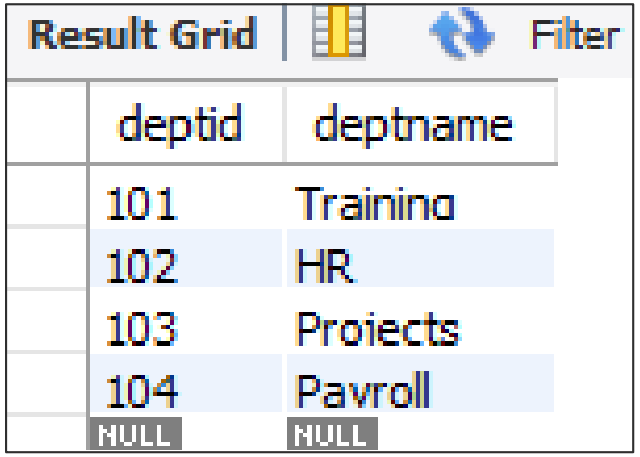
deptid
101
102
103
104

```
SELECT  deptname, loc
FROM Dept;
```



deptname	loc
Training	CHENNAI
HR	CHENNAI
Projects	CHENNAI
Payroll	CHENNAI

```
SELECT deptid,deptname
FROM Dept;
```



deptid	deptname
101	Training
102	HR
103	Projects
104	Payroll
NULL	NULL

Column Alias Name

Renames a column heading by using the alias name through your query.

Examples:

```
SELECT Deptid AS "Dept No", deptname AS "Dept Name", loc AS Location
 FROM Dept;
```

```
SELECT Deptid "Dept No", deptname "Dept  Name", loc Location
FROM Dept;
```

```
SELECT Deptno Dept_no, dname Dept_Name, loc Location
FROM Dept;
```

Result Grid | Filter Rows:

Dept No	Dept Name	Location
101	Training	CHENNAI
102	HR	CHENNAI
103	Projects	CHENNAI
104	Payroll	CHENNAI

Arithmetic Operators

We can use arithmetic operators in any clause of a SQL statement except in the FROM clause.

Operators:

Example:

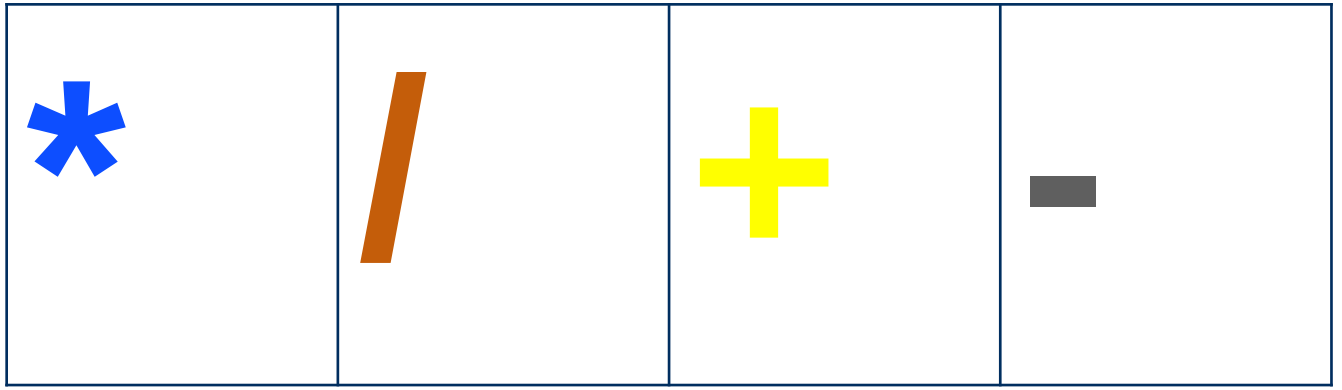
```
SELECT ename, esalary, esalary *12
FROM emp;
```

Result Grid | Filter Rows:

ename	esalary	esalary *12
Ashok	10000	120000
Ashoka	10000	120000
oopi	4000	48000
Hari	10000	120000
Kevin	7000	84000
Narmadha	5000	60000
Sridhar	4000	48000
Sriram	12000	144000

Arithmetic Operators

Operator Precedence :



Selection Capability

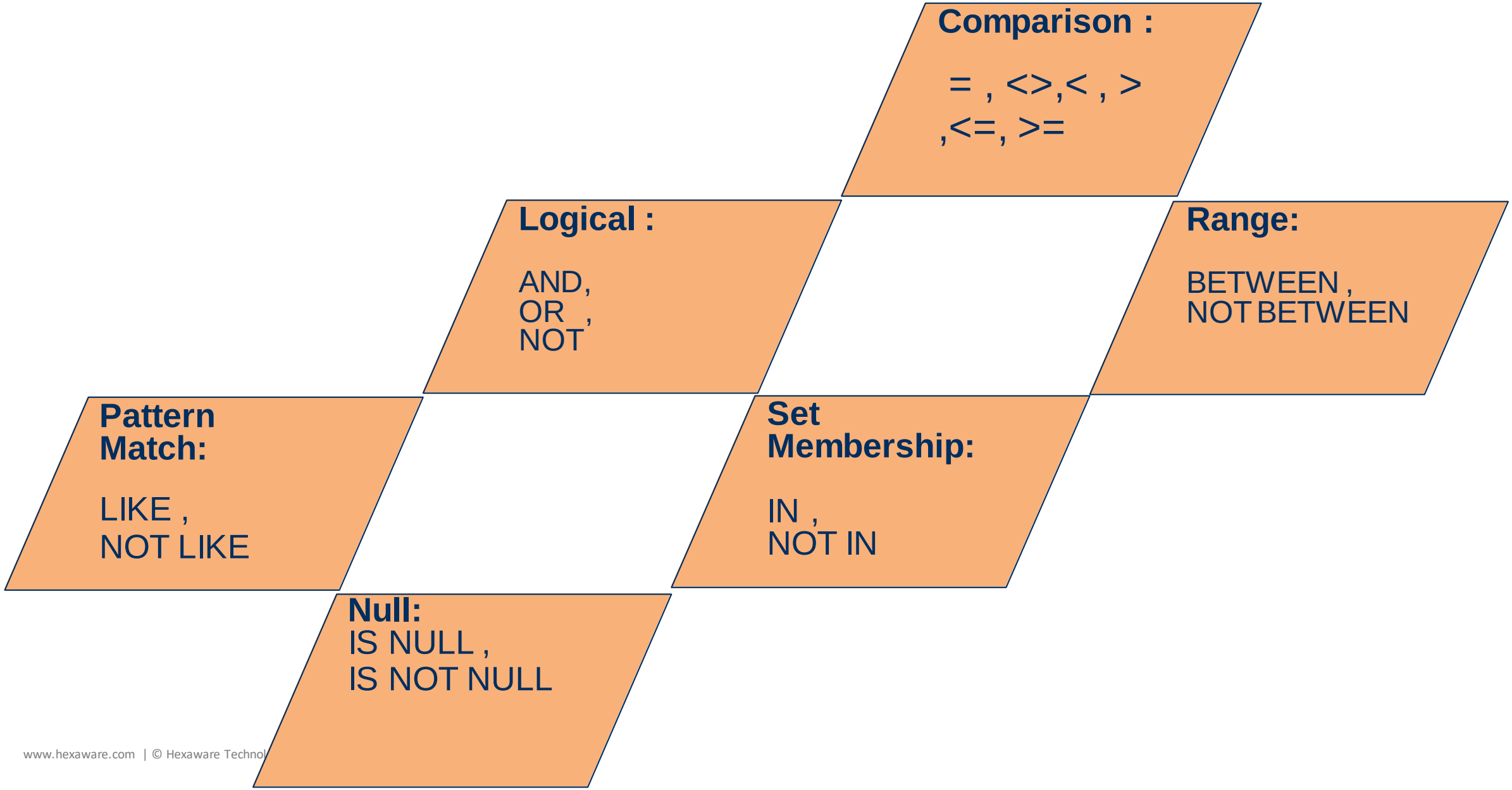
Selection Capability:

- Used to choose the rows in a table that you want returned by a query.
- Various criteria can be used to restrict the rows that you see.
- Restrict the rows returned, by using the WHERE clause.

Syntax:

```
SELECT *|{[DISTINCT] column|expression
[alias],...}
FROM table
[WHERE condition(s)];
```

Operators Used IN Where Clause



Comparison Search Condition

Comparison Conditions :

Conditions that compare one expression to another value or expression.

Examples:

```
SELECT * FROM emp
 WHERE esalary> 3000 AND deptid=101;
```

Result Grid | Filter Rows:

ename	eid	esalary	deptid
Ashok	1004	10000	101
Narmadha	1005	5000	101
Sridhar	1006	4000	101
NULL	NULL	NULL	NULL

```
SELECT * FROM emp
 WHERE deptid = 102 OR Deptid=103;
```

Result Grid | Filter Rows:

ename	eid	esalary	deptid
Hari	1009	10000	102
Kevin	1008	7000	103
Sriram	1007	12000	103
NULL	NULL	NULL	NULL

```
SELECT * FROM emp
WHERE deptid=103;
```

Result Grid | Filter Rows:

ename	eid	esalary	deptid
Kevin	1008	7000	103
Sriram	1007	12000	103
NULL	NULL	NULL	NULL

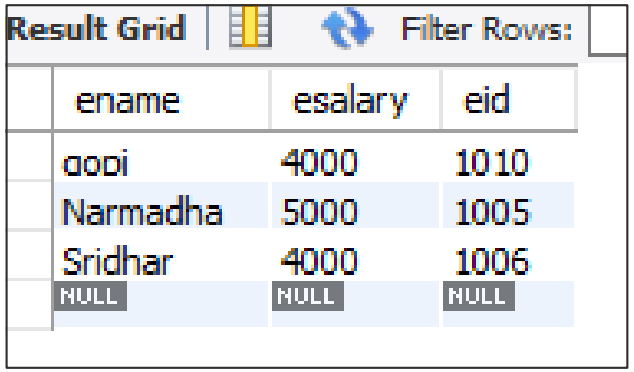
Range Search Condition

Range Condition:

You can display rows based on a range of values using the BETWEEN range condition. The range that you specify contains a lower limit and an upper limit

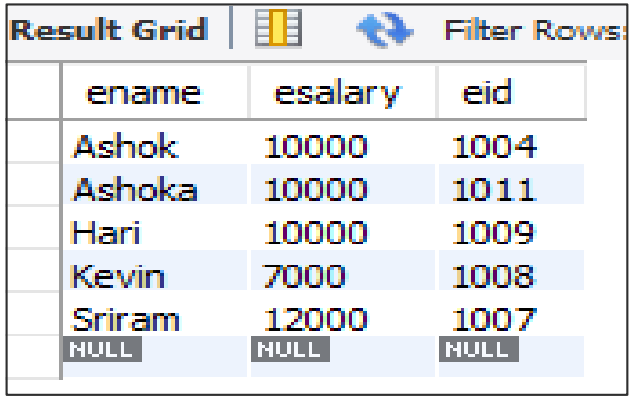
Examples:

```
SELECT ename,esalary,eid FROM emp
 WHERE esalary BETWEEN 3000 AND 5000;
```


Result Grid | Filter Rows:

ename	esalary	eid
oopi	4000	1010
Narmadha	5000	1005
Sridhar	4000	1006
NULL	NULL	NULL

```
SELECT ename,salary,eid FROM emp
WHERE esalary NOT BETWEEN 3000 AND 5000;
```


Result Grid | Filter Rows:

ename	esalary	eid
Ashok	10000	1004
Ashoka	10000	1011
Hari	10000	1009
Kevin	7000	1008
Sriram	12000	1007
NULL	NULL	NULL

Set Membership search Conditions
Set Membership

- Used to test for values in a specified set of values.
- Uses the keyword:
  - IN
  - NOT IN
- The *membership* condition is also known as IN *condition.*

Examples:

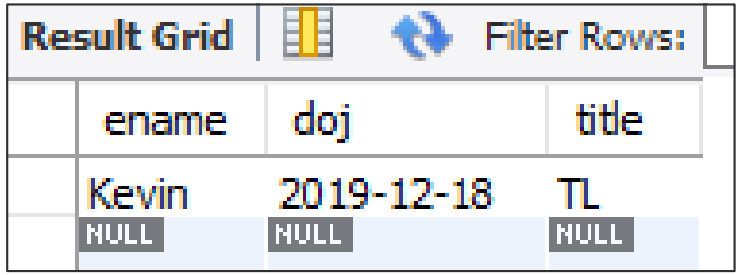
```
SELECT ename, doj,title
FROM emp
WHERE title IN ('SSE', 'ASE', 'Manager');
```

Result Grid | Filter Rows:

ename	doj	title
Ashok	2019-12-18	SSE
Ashoka	2019-12-18	SSE
oopi	2019-12-12	ASE
Hari	2019-12-18	SSE
Narmadha	2019-12-12	ASE
Sridhar	2019-12-12	ASE
Sriram	2012-12-12	Manaoer
NULL	NULL	NULL

Contd…

```
SELECT ename, doj,title
FROM emp
WHERE title NOT IN ('SSE', 'ASE', 'Manager');
```


Result Grid | Filter Rows:

ename	doj	title
Kevin	2019-12-18	TL
NULL	NULL	NULL

Pattern Match Search Condition

The pattern-matching operation is referred to as a *wildcard* search.
Two symbols can be used to construct the search string.

SQL has two special Pattern Matching symbols (wildcard)

% - represents any sequence of zero or more characters

_ - represents any single character

Examples

```
SELECT ename,title,doj
FROM emp
 WHERE title LIKE  '_S%';
```

Result Grid | Filter Rows:

ename	title	doj
Ashok	SSE	2019-12-18
Ashoka	SSE	2019-12-18
oopi	ASE	2019-12-12
Hari	SSE	2019-12-18
Narmadha	ASE	2019-12-12
Sridhar	ASE	2019-12-12
NULL	NULL	NULL

```
SELECT ename,title,doj
FROM emp
 WHERE ename LIKE 'A%';
```

Result Grid | Filter Rows:

ename	title	doj
Ashok	SSE	2019-12-18
Ashoka	SSE	2019-12-18
NULL	NULL	NULL

NULL Search Condition

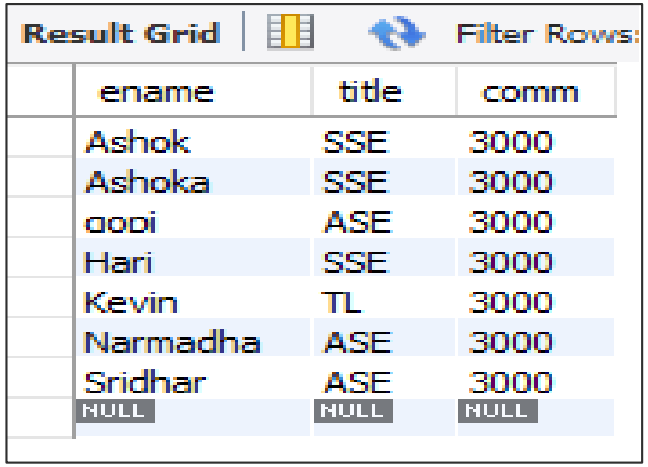
NULL

- Means the value is Unavailable, unassigned ,unknown, or inapplicable.
- Cannot be tested with = because a null cannot be equal or unequal to any value.
- Include the IS NULL condition and the IS NOT NULL condition.
- IS NULL condition tests for nulls.

Examples:

```
SELECT ename, title,comm
 FROM emp
 WHERE comm IS NULL;
```


```
SELECT ename, title,comm
FROM emp
WHERE comm IS NOT NULL;
```



ORDER BY Clause

Sort rows with the ORDER BY clause

- ASC: ascending order, default
- DESC: descending order

The ORDER BY clause comes last in the SELECT statement.

Single column Ordering:

Examples:

```
SELECT ename, title, esalary
FROM emp
ORDER BY salary;
```

Result Grid | Filter Rows:

ename	title	esalary
oopi	ASE	4000
Sridhar	ASE	4000
Narmadha	ASE	5000
Kevin	TL	7000
Ashok	SSE	10000
Ashoka	SSE	10000
Hari	SSE	10000
Sriram	Manager	12000

```
SELECT ename, title, esalary,doj
FROM emp
ORDER BY doj DESC;
```

Result Grid | Filter Rows:

ename	title	esalary	doj
Ashok	SSE	10000	2019-12-18
Ashoka	SSE	10000	2019-12-18
Hari	SSE	10000	2019-12-18
Kevin	TL	7000	2019-12-18
oopi	ASE	4000	2019-12-12
Narmadha	ASE	5000	2019-12-12
Sridhar	ASE	4000	2019-12-12
Sriram	Manager	12000	2012-12-12
NULL	NULL	NULL	NULL

Contd…

- **Multiple column Ordering:**

```
SELECT ename, title, deptid, esalary,doj
FROM emp
ORDER BY deptid DESC, esalary Asc;
```

Result Grid | Filter Rows: | Edit:

ename	title	deptid	esalary	doj
oopi	ASE	104	4000	2019-12-12
Ashoka	SSE	104	10000	2019-12-18
Kevin	TL	103	7000	2019-12-18
Sriram	Manaoer	103	12000	2012-12-12
Hari	SSE	102	10000	2019-12-18
Sridhar	ASE	101	4000	2019-12-12
Narmadha	ASE	101	5000	2019-12-12
Ashok	SSE	101	10000	2019-12-18
NULL	NULL	NULL	NULL	NULL

Functions in MySQL

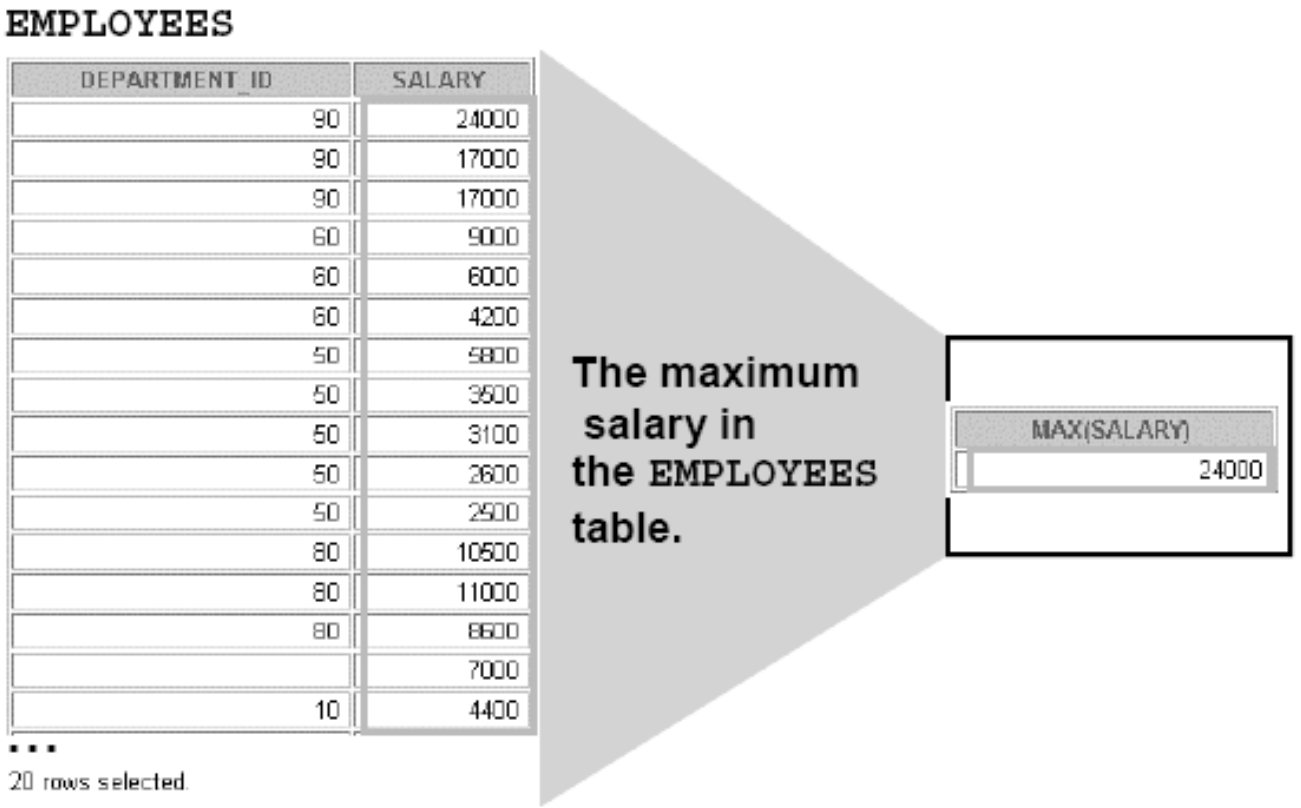
Function -MySQL

MySQL functions including aggregate functions, string functions, date time functions, control flow functions, etc.

Aggregate
String
Control Flow
Date and Time
Comparison
Numeric

Group Functions

Group functions operate on sets of rows to give one result per group.



Group Functions

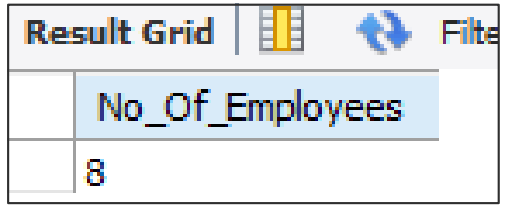
Function Name	Example
Sum	SELECT SUM(salary) AS TotalSalary FROM employee;
Avg	Select Avg(salary) as AVGSalary from employee
Count	Select count(salary) NoOfEmployee from employees
Max	Select max(salary) as MaxSalary from employees
Min	Select min(salary)as MinSalary from employees

Group Functions

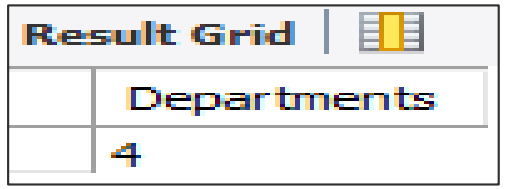
COUNT

Examples:

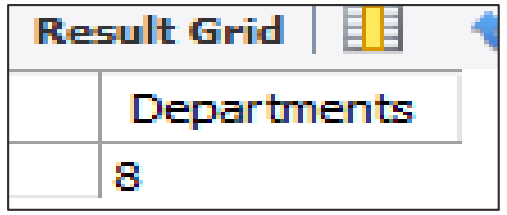
```
SELECT COUNT(*) AS
 No_Of_Employees
 FROM emp;
```




```
SELECT COUNT( distinct deptid) AS Departments
FROM emp;
```



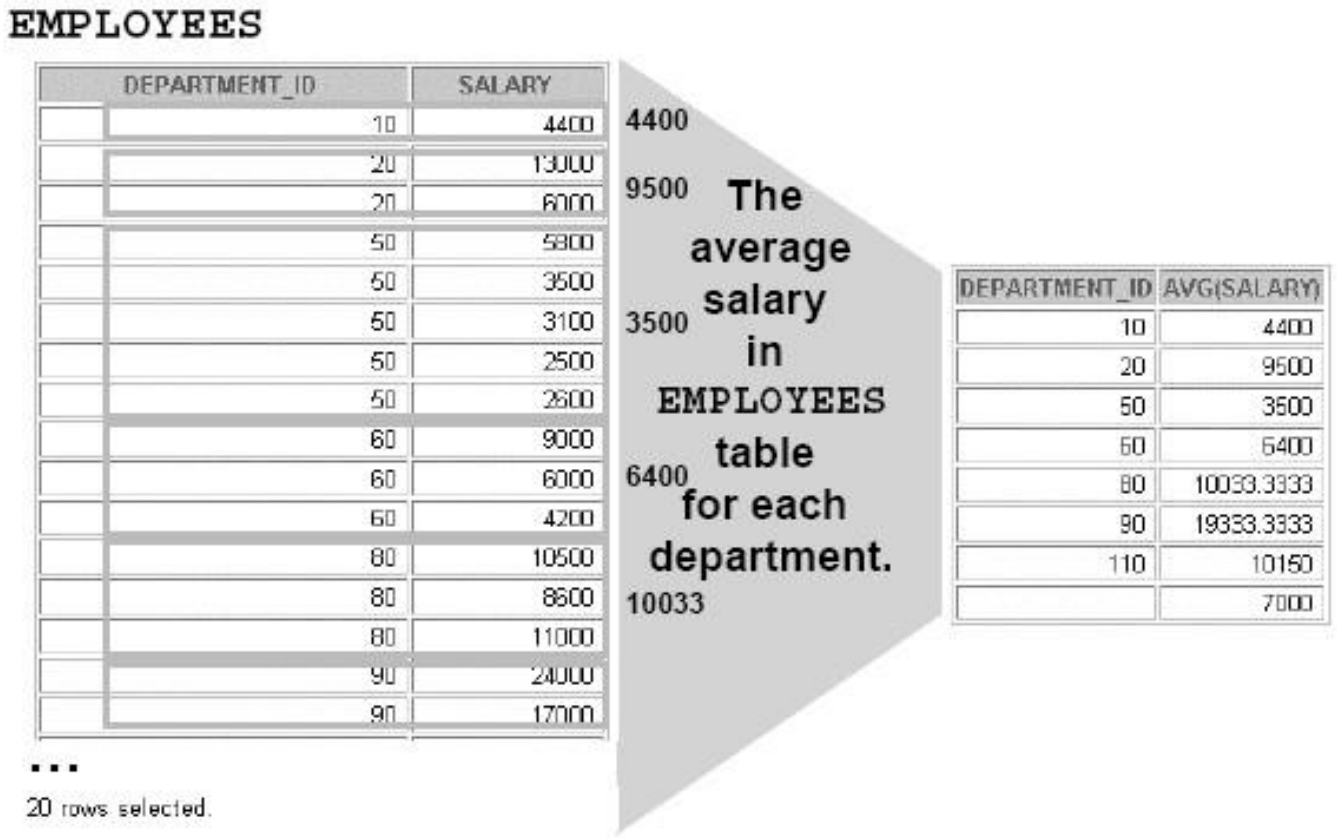

```
SELECT COUNT(deptno) as Departments FROM emp;
```



COUNT(*) - Counts all rows of a table, regardless of whether nulls or duplicate values occur

The GROUP BY Clause

Divide rows in a table into smaller groups



The GROUP BY Clause

- Aggregate functions are normally used in conjunction with a GROUP BY clause.
- The GROUP BY clause enables the aggregate functions to answer more complex managerial Queries

Guidelines for Group by Clause

- All columns in the SELECT list that are not in group functions must be in the GROUP BY clause.
- GROUP BY clause does not support the use of column alias, but the actual names.
- GROUP BY clause can only be used with aggregate functions like SUM, AVG, COUNT, MAX, and MIN.
- Aggregate functions cannot be used in a GROUP BY clause.

The GROUP BY Clause

Syntax

```
SELECT [column,] group_function(column), ...
FROM  table  [WHERE
condition]  [GROUP BY
column] [ORDER BY
column];
```

Examples:

```
SELECT COUNT(ename),deptid
FROM emp
GROUP BY deptid;
```

Result Grid | Filter Rc

count(ename)	deptid
3	101
1	102
2	103
2	104

```
SELECT deptid,  COUNT(ename) AS EmployeeCount,SUM(esalary) AS TotalSalary
 FROM emp
    GROUP BY deptid;
```

Result Grid | Filter Rows:

deptid	EmployeeCount	Total_Salary
101	3	19000
102	1	10000
103	2	19000
104	2	14000

Contd..

Grouping more than one column:

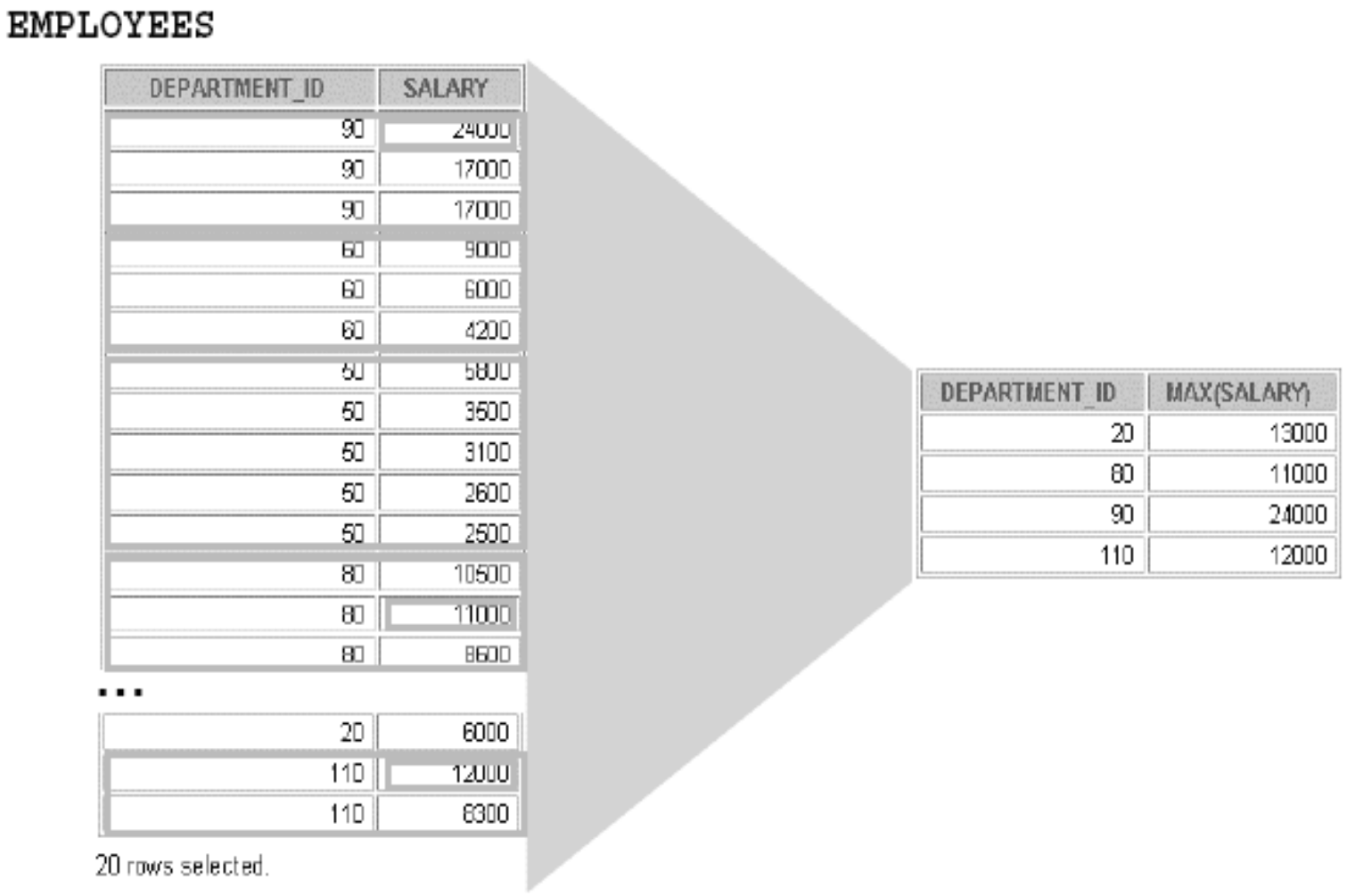
Examples:

```
SELECT title,deptid,SUM(esalary) AS Total_Salary
FROM emp
GROUP BY title,deptid;
```

Result Grid | Filter Rows:

title	deptid	Total_Salary
ASE	101	9000
ASE	104	4000
Manaoer	103	12000
SSE	101	10000
SSE	102	10000
SSE	104	10000
TL	103	7000

Restricting Groupings – Having Clause

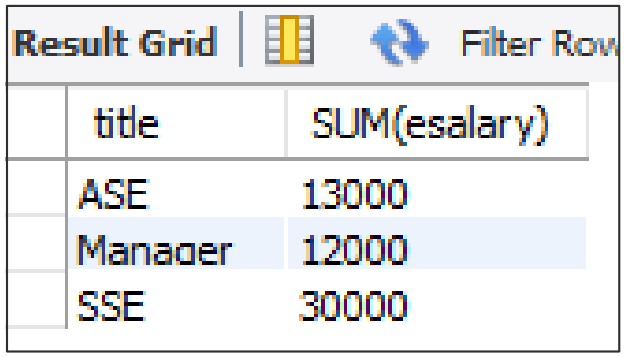


Restricting Groupings – Having Clause

The HAVING clause is used for aggregate functions in the same way that a WHERE clause is used for column names and expressions.

Example:

```
SELECT title,SUM(esalary)
FROM emp
GROUP BY title
HAVING SUM(esalary) > 10000;
```



title	SUM(esalary)
ASE	13000
Manager	12000
SSE	30000

Having Clause with Where clause

In the same way that you use the WHERE clause to restrict the rows that you select, you use the HAVING clause to restrict Groups

- **Syntax:**

  SELECT column, group_function

  FROM table

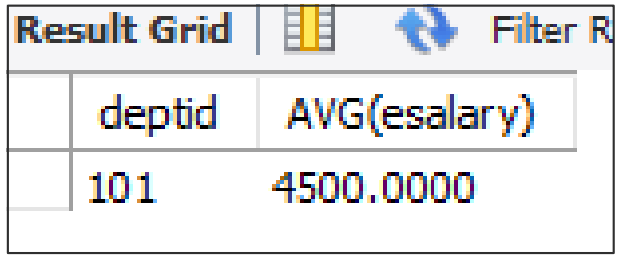
  [WHERE condition]

  [GROUP BY group_by_expression] [HAVING group_condition] [ORDER BY column];

Having Clause with Where clause cont…

- Example:

```
SELECT deptid, AVG(esalary)  FROM emp
WHERE esalary < 7000
GROUP BY deptid
HAVING AVG(esalary) > 4200;
```



- **Using the WHERE, GROUP BY, and HAVING Clauses Together**
  - The WHERE clause first filters the rows,
  - And the remaining rows are grouped into blocks by using GROUP BY clause,
  - Finally the row groups are filtered by the HAVING clause.

Assignment

4. Basic SELECT

5. Restricting & Sorting Data

6. Group Clause

Readings reference:

- https://www.w3schools.com/sql/sql_ref_mysql.asp
- https://www.techonthenet.com/mysql/functions/
- https://www.w3resource.com/mysql/mysql-functions-and-operators.php
- https://www.tutorialspoint.com/mysql/mysql-useful-functions.htm
- http://www.mysqltutorial.org/mysql-functions.aspx

thank you

www.hexaware.com