Chapter 16:

Advanced MySQL-
Grouping Records and Joining Tables

Informatics Practices

Class XII

By- Rajesh Kumar Mishra

PGT (Comp.Sc.)

KV No.1, AFS, Suratgarh (Raj.)

e-mail : rkmalld@gmail.com

Grouping Records in a Query

- Some time it is required to apply a query in a group of records instead of whole table.
- You can group records by using GROUP BY <Column Name> clause with Select command. A group column is chosen which have non-distinct values like City, Job etc.
- Generally, Aggregate Functions [MIN(), MAX(), SUM(), AVG(), COUNT()] etc. are applied on groups.

Aggregate or Group Functions

Name	Purpose	Example
SUM()	Returns the sum of given column.	Select SUM(Pay) from Emp; Select Sum(Pay), Sum(Net) from Emp;
MIN()	Returns the minimum value in the given column.	Select MIN(Pay) from Emp;
MAX()	Returns the maximum value in the given column.	Select MAX(Pay) from Emp;
AVG()	Returns the Average value of the given column.	Select AVG(Pay) from Emp;
COUNT()	Returns the total number of values/ records as per given column.	Select COUNT(Name) from Emp; Select COUNT(*) from Emp;

A query using Aggregate Function with GROUP BY clause works on the group instead of whole records in the table.

Aggregate or Group Functions

An Aggregate function may applied on a column with DISTINCT or ALL keyword. If nothing is given ALL is assumed.

- **SUM ([DISTINCT|ALL] <Column>)**

  This function returns the sum of values in given column or expression.

```
mysql> Select Sum(Sal) from EMP;
mysql> Select Sum(DISTINCT Sal) from EMP;
mysql> Select Sum (Sal) from EMP where City='Kanpur';
mysql> Select Sum (Sal) from EMP Group By City;
mysql> Select Job, Sum(Sal) from EMP Group By Job;
```

- **MIN ([DISTINCT | ALL] <column>)**

This functions returns the Minimum value in the given column.

```
mysql> Select Min(Sal) from EMP;
mysql> Select Min(Sal) from EMP Group By City;
mysql> Select Job, Min(Sal) from EMP Group By Job;
```

Aggregate or Group Functions

- **MAX ([DISTINCT|ALL] <Column>)**

  This function returns the Maximum value in given column.

  mysql> Select Max(Sal) from EMP;

  mysql> Select Max(Sal) from EMP where City='Kanpur';

  mysql> Select Max(Sal) from EMP Group By City;

- **AVG ([DISTINCT | ALL] <column>)**

This functions returns the Average value in the given column.

mysql> Select AVG(Sal) from EMP;

mysql> Select AVG(Sal) from EMP Group By City;

- **COUNT ([DISTINCT | ALL] <*|column>)**

This functions returns the number of rows in the given column.

mysql> Select Count (*) from EMP;

mysql> Select Count(Sal) from EMP Group By City;

mysql> Select Count(*), Sum(Sal) from EMP Group By Job;

Using Group By clause in *Select Query*

- **Grouping Results**

The Group By clause along with Aggregate functions may cause of grouping.

```
mysql> Select Max(Sal) from EMP Group By City;
```

- **Grouping on Multiple columns- Nested Groups**

You may define multiple columns with GROUP BY to make a nested groups.

```
mysql> Select City, Job, Count (*) from EMP Group By City, Job;
```

- **Condition with Group**

You may use any condition on group, if required. HAVING <condition> clause is used to apply a condition.

```
mysql> Select Job, Count (*) from EMP
        Group By Job HAVING Count(*)<=3;
```

Where clause is not used with aggregate functions.
HAVING can't be used without GROUP By clause.

Joining tables - Join Query

Some times it is required to access information from two or more tables , which requires the Joining of two or more tables. Such query is called Join Query.

MySQL facilitates you to handle Join Queries. The major types of Join is as follows-

- **Cross Join (Cartesian Product)**
- **Equi Join**
- **Non-Equi Join**
- **Natural Join**
- **Outer Join**
  - **Left Outer Join**
  - **Right Outer Join**
  - **Full Outer Join**

Cross Join – Mathematical Principle

Consider the two set **A= {a,b}** and **B={1,2}**

The Cartesian Product i.e. **AxB = {(a,1) (a,2) (b,1) (b,2)}**

Similarly, we may compute Cross Join of two tables by joining each Record of first table to each record of second table.

R

A	B	C
p	q	s
m	n	t
o	p	s
l	m	u

X

S

C	X	Y
s	q	r
t	n	m
o	p	s

R x S

A	B	C	C	X	Y
p	q	s	s	q	r
p	q	s	t	n	m
p	q	s	o	p	s
m	n	t	s	q	r
m	n	t	t	n	m
m	n	t	o	p	s
...	...	...	...	..	..
l	m	u	o	p	s

The table will contain (4x3=12) rows and 6 columns.

Equvi Join – Mathematical Principle

In Equvi Join, records are joined on the equality condition of Joining Column. Generally, the Join column is column which is common in both tables.

Consider the following table **R** and **S** having **C** as Join column.

R

A	B	C
p	q	s
m	n	t
o	p	s
l	m	u

S

C	X	Y
s	q	r
t	n	m
o	p	s

T (Equi Join)

A	B	C	C	X	Y
p	q	s	s	q	r
m	n	t	t	n	m
o	p	s	s	p	r

The result table will contain 6 columns but records are selected those are having Equal value for C column in both table.

Non-Equvi Join – Mathematical Principle

In Non-Equvi Join, records are joined on the condition other than Equal operator (>,<,<>,>=,<=) for Joining Column (common column).

Consider the following table **R** and **S** having **C** as Join column and **<>** (not equal) operator is applied in join condition.

R

A	B	C
p	q	s
m	n	t
o	p	s
l	m	u

<>

S

C	X	Y
s	q	r
t	n	m
o	p	s

T (Non-Equi Join)

A	B	C	C	X	Y
p	q	s	t	n	m
p	q	s	o	p	s
m	n	t	s	p	r
m	n	t	o	p	s
o	p	s	t	n	m
o	p	s	o	p	s
l	m	u	s	q	R
l	m	u	t	n	M
l	m	u	o	p	s

The result table will contain 6 columns but records are selected those are having not- equal value for C column in both table.

Natural Join – Mathematical Principle

The Natural Join is much similar to Equi Join i.e. records are joined on the equality condition of Joining Column except that the common column appears one time.

Consider the following table **R** and **S** having **C** as Join column.

R

A	B	C
p	q	s
m	n	t
o	p	s
l	m	u

S

C	X	Y
s	q	r
t	n	m
o	p	s

T (Natural Join)

A	B	C	X	Y
p	q	s	q	r
m	n	t	n	m
o	p	s	p	r

The result table will contain **5** columns (common column is eliminated) but records are selected those are having Equal value for C column in both table.

Outer Join (Left, Right and Full)

The Outer join is much similar to Natural Join but it also contains left (ignored) record during Join.

Left outer Join = Natural Join + Record ignored from Left Table.

Right outer Join = Natural Join + Record ignored from Right Table.

Full outer Join = Natural Join + Record ignored from Left & Right Table.

R

A	B	C
p	q	s
m	n	t
o	p	s
l	m	u

S

C	X	Y
s	q	r
t	n	m
o	p	s

T (Full Outer Join)

A	B	C	X	Y
p	q	s	q	r
m	n	t	n	m
o	p	s	p	R
l	m	u	-	-
-	-	o	p	s

Left outer Join result contains ignored record from Left table only.
Right Outer Join contains ignored record from Right table only.
The Full outer Join result contains ignored records from both table.

Implementing Join Operation in MySQL

Suppose two table EMP and DEPT are given -

EMP

EmpID	EName	City	Job	Pay	DeptNo
E1	Amitabh	Mumbai	Manager	50000	D1
E2	Sharukh	Delhi	Manager	40000	D2
E3	Amir	Mumbai	Engineer	30000	D1
E4	Kimmi	Kanpur	Operator	10000	D2
E4	Puneet	Chennai	Executive	18000	D3
E5	Anupam	Kolkatta	Manager	35000	D3
E6	Syna	Banglore	Secretary	15000	D1
...	....	....	....	...	...

DEPT

DeptNo	DName	Location
D1	Production	Mumbai
D2	Sales	Delhi
D3	Admn	Mumbai
D4	Research	Chennai

Suppose we want complete details of employees with their Deptt. Name and Location..... this query requires the join of both tables

How to Join ?

MySQL offers different ways by which you may join two or more tables.

- **Method 1 : Using Multiple table with FROM clause**

  The simplest way to implement JOIN operation , is the use of multiple table with FROM clause followed with Joining condition in WHERE clause.

```
Select * From EMP, DEPT
Where Emp.DeptNo=Dept.DeptNo ;
```

To avoid ambiguity you should use Qualified name i.e. <Table>.<column>

```
Select E.*, D.* From EMP E, DEPT D
Where  E.DeptNo=D.DeptNo
```

You may Alias the tables for easy referencing

- **Method 2: Using JOIN keyword**

  MySQL offers JOIN keyword, which can be used to implement all type of Join operation.

```
Select * From EMP JOIN DEPT ON Emp.DeptNo=Dept.DeptNo ;
```

Using Multiple Table with FROM clause

The General Syntax of Joining table is-

SELECT < List of Columns> FROM <Table1, Table 2, …>
WHERE <Joining Condition> [Order By ..] [Group By ..]

- You may add more condition with Joining Condition using AND/OR NOT operators, if required.
- All types of Join (Equi, No-Equi, Natural etc. are implemented by changing the Operators in Joining Condition and selection of columns with SELECT clause.

Ex. Find out the name of Employees working in Production Deptt.

```
Select  Ename From EMP, DEPT
Where Emp.DeptNo=Dept.DeptNo AND Dname='Production';
```

Ex. Find out the name of Employees working in same city from where they belongs (hometown).

```
Select  Ename From EMP, DEPT
Where Emp.DeptNo=Dept.DeptNo And City=Location;
```

Using JOIN keyword with FROM clause

MySQL 's JOIN Keyword may be used with From clause.

SELECT < List of Columns>
FROM <Table1> [CROSS][NATURAL][LEFT][RIGHT]
JOIN <Table2> [ON <Joining Condition>] [USING <column>]
[WHERE <Condition>] [Order By ..] [Group By ..]

Ex. Find out the name of Employees working in Production Deptt.

```
Select  Ename From EMP JOIN DEPT ON Emp.DeptNo=Dept.DeptNo
Where Dname='Production';
```

Or

```
Select  Ename From EMP JOIN DEPT USING DeptNo
Where Dname='Production';
```

Ex. Find out the name of Employees working in same city from where they belongs (hometown) .

```
Select  Ename
From EMP NATURAL JOIN DEPT ON  Emp.DeptNo=Dept.DeptNo
WHERE City=Location;
```

Nested Query (A query within another query)

Sometimes it is required to join two sub-queries to solve a problem related to the single or multiple table. Nested query contains multiple query in which inner query evaluated first.

The general form to write Nested query is-

Select …. From <Table>

Where <Column1> <Operator>

(Select Column1 From <Table> [Where <Condition>])

Ex. Find out the name of Employees working in Production Deptt.

```
Select  Ename From EMP
Where DeptNo =
(Select DeptNo From DEPT Where DName='Production');
```

Ex. Find out the name of Employees who are getting more pay than 'Ankit'.

```
Select  Ename From EMP
Where Pay >= (Select Pay From EMP WHERE Ename='Ankit' );
```