Study Material: Querying Using SQL
Subject: Informatics Practices
Class XII
Course Instructor: **Prabhat Das (PGP-IP)**

Contents:

1. Introduction to Querying Using SQL
2. Ordering Records in Results – ORDER BY
3. Types of SQL Functions
4. Grouping Result – GROUP BY
5. Nested Groups – Grouping on Multiple Columns
6. Placing Conditions on Groups – HAVING Clause
7. Non-Group Expressions with GROUP BY

1. Introduction:

This chapter will talk about some SQL commands in details. After this chapter, you shall be able to display groups of records, summary of groups of records and display selected groups of records.

2. Ordering Records in Results – ORDER BY

The ORDER BY clause is used to sort out the query result in either Ascending or Descending order. Keywords ASC and DESC denote the order- ASC stands for ascending, and the DESC stands for descending. If not specified, the default ORDER BY clause sorts the result set into ascending order.

You can use the ORDER BY clause of SQL SELECT statement as per following format:

SELECT <comma separated select list> FROM <table> [WHERE <condition>] ORDER BY <fieldname> [ASC|DESC] [, <fieldname> [ASC|DESC], …];

3. Types of SQL Functions:

All the functions can be generally categorized into following two types:

1) Single Row (or Scalar) functions:
   Scalar functions work with a single row at a time. A single row function returns result for every row of a queried table.

2) Multiple Row (or Group or Aggregate) functions:
   Group functions work with data of multiple rows at a time and return aggregated value.

   Many types of Multiple Row functions are:

1. AVG:
   This function computes the average of given data.
   Syntax:
   AVG([DISTINCT | ALL] n)

2. COUNT:
   This function counts the number of rows in a given column or expression.

   Syntax:
   COUNT({*[DISTINCT | ALL] expr})

3. MAX:
   This function returns the maximum value from a given column or expression.

   Syntax:
   MAX([DISTINCT | ALL] expr)

4. MIN:
   This function returns the minimum value from a given column or expression.

   Syntax:
   MIN([DISTINCT | ALL] expr)

5. SUM:
   This function returns the sum of value in a given column or expression.

   Syntax:
   SUM([DISTINCT | ALL] n)

4. Grouping Result – GROUP BY:

The GROUP BY clause is used in SELECT statements to divide the table into groups. Grouping can be done by a column name, or with aggregate functions in which case the aggregate produces a value for each group.

For example, To calculate the *number of employees* in each grade, you use the command

SELECT job, COUNT(*) FROM empl GROUP BY job;

Job	COUNT(*)
ANALYST	2
CLERK	4
MANAGER	3
PRESIDENT	1
SALESMAN	4

5. Nested Groups – Grouping on Multiple Columns:

With GROUP BY clause, you can create groups within groups. Such type of grouping is called ***Nested grouping.***

This can be done by specifying in GROUP BY expression, where the *first field* determines the *highest group level*, the *second field* determines the *second group level*, so on. The *last field* determines the *lowest level of grouping.*

For example, To group records *job wise* within *Deptno* wise, you need to issue a query statement like:

```
SELECT Deptno, Job, COUNT(empno)
FROM empl
GROUP BY Deptno, Job;
```

Deptno	Job	Count(empno)
10	CLERK	1
10	MANAGER	1
10	PRESIDENT	1
20	ANALYST	2
20	CLERK	2
20	MANAGER	1
30	CLERK	1
30	MANAGER	1
30	SALESMAN	4

6. Placing Conditions on Groups – HAVING Clause:

The HAVING clause places conditions on groups in contrast to WHERE clause that places conditions on individual rows. While WHERE conditions include aggregate functions, HAVING conditions can do so.

For example, to calculate the average gross and total gross for employees belonging to 'E4' grade, the command would be:

```
SELECT AVG(gross), SUM(gross)
FROM employee
GROUP BY grade
HAVING grade = 'E4';
```

This condition would be applicable on group and not on individual rows

7. Non-Group Expressions with GROUP BY

If you include a non-group expression in the select-list of a query with GROUP BY, MySQL will not produce any error. Rather it will pick value of the specified non-group field from the first row of the group. But we do not recommend this practice because it will produce ambiguous results. For instance, consider the following query and its output.

```
SELECT ename, sum(sal)
FROM empl
GROUP BY deptno;
```

This will produce the following output:

Ename	Sum(sal)
AMIR	8750.00
SMITH	10885.00
ANYA	9400.00

See, isn't it conveying hat AMIR's salary-sum is 8750.00, SMITH's 10885.00 and ANYA's 9400.00 ?

Thus, it is recommend not to use non-group expressions in GROUP BY query unless otherwise necessary.