Course code	: **CSE2007**
Course title	: **Database Management System**
Module	: **3**
Topic	: **4**

SQL - Clauses

Objectives

This session will give the knowledge about

- SQL CLAUSES
  - Where clause
  - Order by clause
  - Aggregate Functions
  - Group by clause
  - Having clause

SQL WHERE Clause

The SQL WHERE clause is used to filter the results and apply conditions in a select, insert, update or delete statement.

SYNTAX:

```
SELECT col_name,col_name FROM table_name WHERE condition;
```

EXAMPLE:

```
SELECT * FROM employee WHERE ename = ‘reddy';
SELECT * FROM employee WHERE (ename = ‘reddy‘ AND eid = ‘12b23') OR (salary >= 1000);
```

The following operators can be performed by using Where clause, they are:

=,<,<=,>,>=, Between, Like and IN.

SQL ORDER BY Clause

The ORDER BY clause is used in a SELECT statement to sort results either in ascending or descending order.

Oracle sorts query results in ascending order by default.

Syntax :

```
        SELECT column-list  FROM table_name [WHERE condition]
                [ORDER BY column1 [, column2, .. columnN] [DESC]];
```

Example: If we want to sort the employee table by salary of the employee:

```
        SELECT ename, salary FROM employee ORDER BY salary;

        SELECT name, salary FROM employee ORDER BY name, salary;
```

SQL ORDER BY Clause

The columns specified in ORDER BY clause should be one of the columns selected in the SELECT column list.

We can represent the columns in the ORDER BY clause by specifying the position of a column in the SELECT list, instead of writing the column name.

```
SELECT name, salary FROM employee ORDER BY 1, 2;
```

By default, the ORDER BY Clause sorts data in ascending order. If we want to sort the data in descending order, we must explicitly specify.

SQL ORDER BY Clause

SELECT name, salary FROM employee ORDER BY name, salary DESC;

The above query sorts only the column 'salary' in descending order and the column 'name' by ascending order.

If we want both columns in descending ,we need to specify :

ORDER BY name DESC, salary DESC;

SELECT name, salary FROM employee ORDER BY name DESC, salary ASC;

The SQL ORDER BY would return all records sorted by the name field in descending order, with a secondary sort by salary field in ascending order.

GROUP BY & HAVING Clause

Before we are going to Group by & Having clause ,we need to learn about Aggregate Function in SQL.

Aggregate Function : functions that take a collection of values as input and return a single value.

Some of the commonly used aggregate functions are :

- SUM
- COUNT
- AVG
- MIN
- MAX

COUNT() function

- The **COUNT** function is used to count rows or values of a column that do not contain a NULL value.
- The COUNT function returns a numeric value.
- **Syntax:**

  SELECT **COUNT [(*) | (DISTINCT | ALL)] (COLUMN NAME)** FROM TBLNAME;

- The **DISTINCT** command cannot be used with **COUNT(*)**, only with the **COUNT(column_name)**
- Because Count(*) will count the column with the duplicate values.
- Distinct wont allow duplicate values.

COUNT() function

- The DISTINCT command cannot be used with COUNT(*), only with the COUNT(column_name)

Examples:

- SELECT COUNT(eid) FROM employee;

  Counts all employee IDs
- SELECT COUNT(*) FROM employee;
- SELECT COUNT(DISTINCT(did)) FROM employee;

  Counts only the distinct rows

COUNT() function

- SELECT COUNT(ALL salary)FROM employee;

  Counts all rows for SALARY
- SELECT COUNT(*) FROM employee

  Counts all rows of the EMPLOYEE table;

The difference between ‘*’(asterisk) and ALL are, '*' counts the NULL value also but ALL counts only NON NULL value.

SUM() Function

The SUM function is used to return a total on the values of a column for a group of rows.

The SUM function can also be used in conjunction with DISTINCT.

When SUM is used with DISTINCT, only the distinct rows are total, but total will not accurate in this case ,because rows of data are omitted.

Syntax:

SUM ([DISTINCT] COLUMN NAME)

SUM() Function

Example:

```
SELECT SUM(salary) FROM employee;
```

Totals the salaries

```
SELECT SUM(DISTINCT salary) FROM employee;
```

Totals the distinct salaries

AVG() Function

The AVG function is used to find averages for a group of rows.

When used with the DISTINCT command, the AVG function returns the average of the distinct rows.

Syntax: AVG ([DISTINCT] COLUMN NAME)

Example:

SELECT AVG(salary) FROM employee - Returns the average SAL
SELECT AVG(DISTINCT salary) employee - Returns the distinct FROM average salary
SELECT AVG(exp), AVG(saalry) FROM employee;

MIN() Function

- The MIN function returns the minimum value of a column for a group of rows.
- NULL values are ignored when using the MIN function.

Syntax:

```
        SELECT MIN([ DISTINCT ] COLUMN NAME) from tablename;
```

Example:

```
        SELECT MIN(salary) FROM employee
                Returns the lowest salary
        SELECT MIN(DISTINCT salary) FROM employee
                Returns the lowest distinct salary
```

MAX() Function

- The MAX function returns the minimum value of a column for a group of rows.
- NULL values are ignored when using the MIN function.

Syntax:

SELECT MAX([DISTINCT] COLUMN NAME) from tablename;

Example:

SELECT MAX(salary) FROM employee

Returns the lowest salary

SELECT MAX(DISTINCT salary) FROM employee

Returns the lowest distinct salary

Aggregate Function

- Example for all the aggregate function in single query:
- SELECT COUNT(eid), SUM(salary), SUM(salary) / COUNT(eid) avg_sal FROM employee;

GROUP BY() Clause

Why Group data?

Grouping data is the process of combining columns with duplicate values in a logical order.

Grouping data is accomplished through the use of the GROUP BY clause of a SELECT statement (query).

The GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.

The GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

GROUP BY() Clause

The position of the GROUP BY clause in a query is as follows:

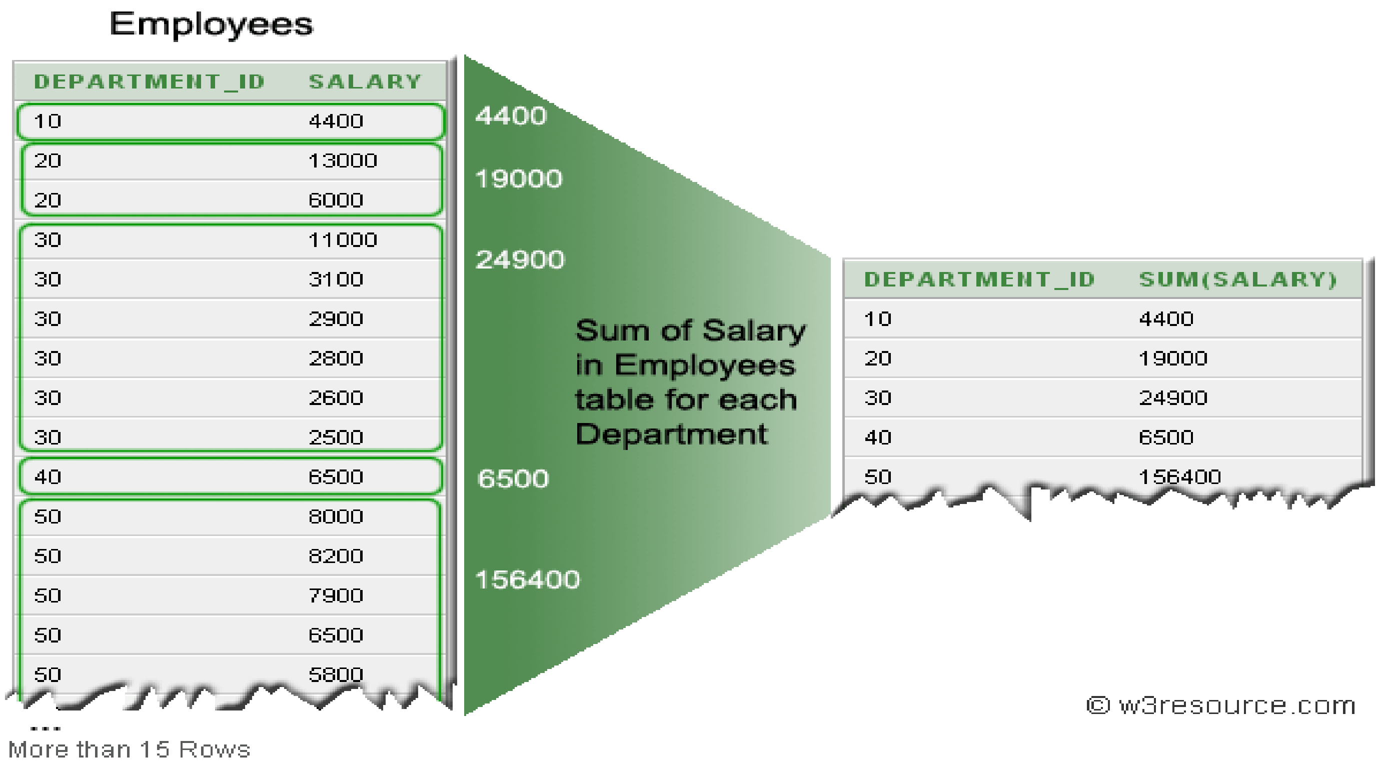
- SELECT
- FROM
- WHERE
- GROUP BY
- ORDER BY

Syntax:

SELECT COLUMN1, COLUMN2 FROM TABLE1, TABLE2 WHERE CONDITIONS GROUP BY COLUMN1, COLUMN2 ORDER BY COLUMN1, COLUMN2;

GROUP BY() Clause

SELECT did, SUM (salary) FROM employee GROUP BY did;



Employees

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
30	11000
30	3100
30	2900
30	2800
30	2600
30	2500
40	6500
50	8000
50	8200
50	7900
50	6500
50	5800

...
More than 15 Rows

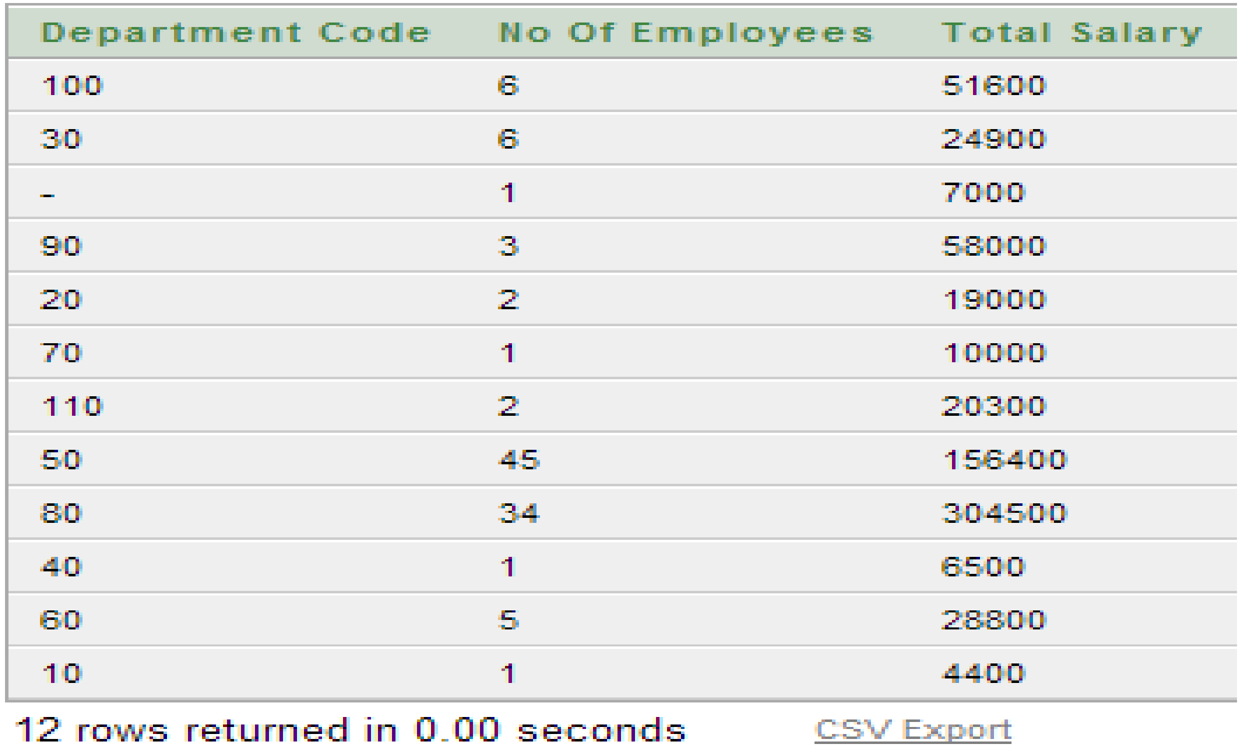
Sum of Salary in Employees table for each Department

DEPARTMENT_ID	SUM(SALARY)
10	4400
20	19000
30	24900
40	6500
50	156400

© w3resource.com

GROUP BY() Clause

SELECT did "Department Code", COUNT(*) "No of Employees", SUM(salary) "Total Salary" FROM employee GROUP BY did;



Department Code	No Of Employees	Total Salary
100	6	51600
30	6	24900
-	1	7000
90	3	58000
20	2	19000
70	1	10000
110	2	20300
50	45	156400
80	34	304500
40	1	6500
60	5	28800
10	1	4400

12 rows returned in 0.00 seconds

GROUP BY() Clause

```
SELECT did "Department Code", eid,  SUM(salary) "Total Salary“ FROM
employee GROUP BY  did,eid;
```

SQL GROUP BY with WHERE clause:

```
SELECT did "Department Code",  SUM(salary) "Total Salary"  FROM  employee  W
HERE eid = 101  GROUP BY  did;
```

HAVING() Clause

SQL HAVING clause specifies a search condition for a group or an aggregate.

HAVING is usually used in a GROUP BY clause, but even if we are not using GROUP BY clause, we can use HAVING to function like a WHERE clause.

We must use HAVING with SQL SELECT.

Syntax

SELECT <column_list> FROM < table name > WHERE <condition> GROUP BY <columns> [HAVING] <condition>;

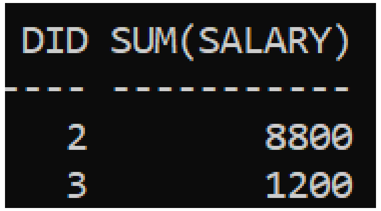
HAVING() Clause

How a HAVING clause works?

- The select clause specifies the columns.
- The from clause supplies a set of potential rows for the result.
- The where clause gives a filter for these potential rows.
- The group by clause divide the rows in a table into smaller groups.
- The having clause gives a filter for these group rows.

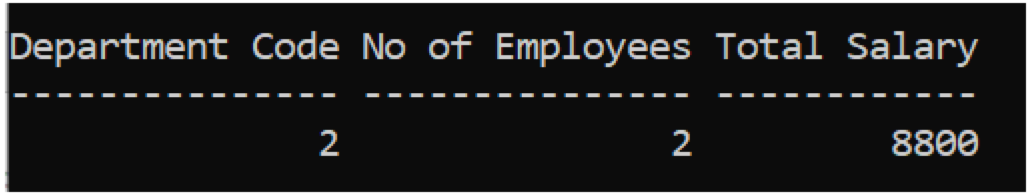
HAVING() Clause

SELECT did,SUM(salary) FROM employee GROUP BY did HAVING did>1;



```
DID SUM(SALARY)
---- -----------
  2        8800
  3        1200
```

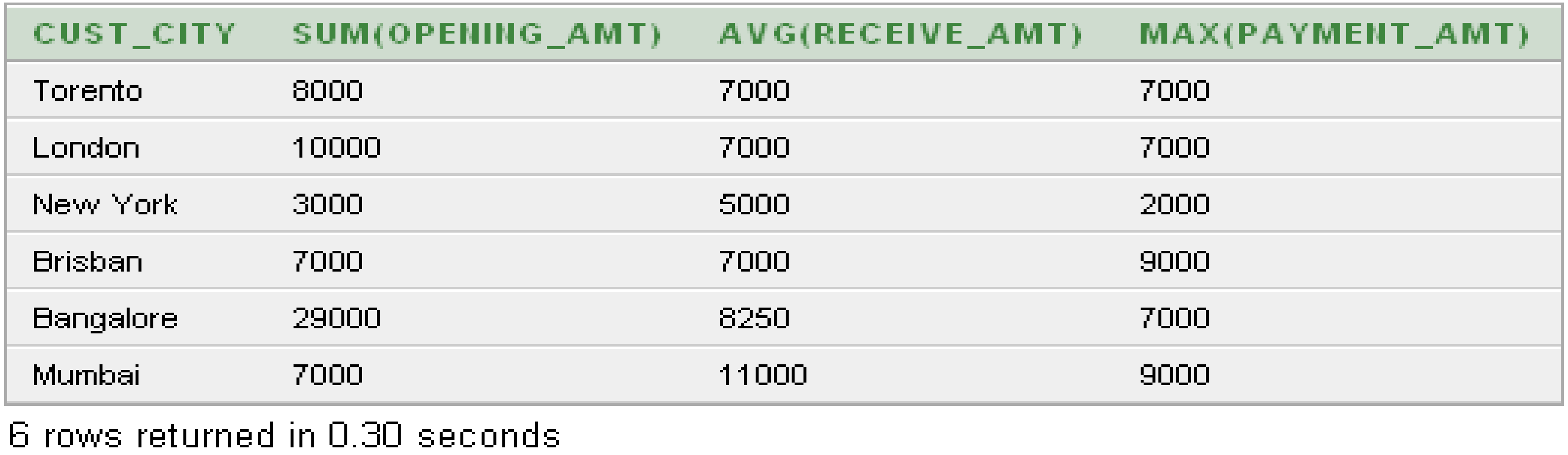
SELECT did "Department Code", COUNT(*) "No of Employees", SUM(salary) "Total Salary" FROM employee GROUP BY did HAVING count(*)>1;



```
Department Code No of Employees Total Salary
--------------- --------------- ------------
              2               2         8800
```

HAVING() Clause

SELECT cust_city,SUM(opening_amt), AVG(receive_amt), MAX(payment_amt) FROM customer WHERE grade=2 GROUP BY cust_city HAVING AVG(receive_amt)>500;



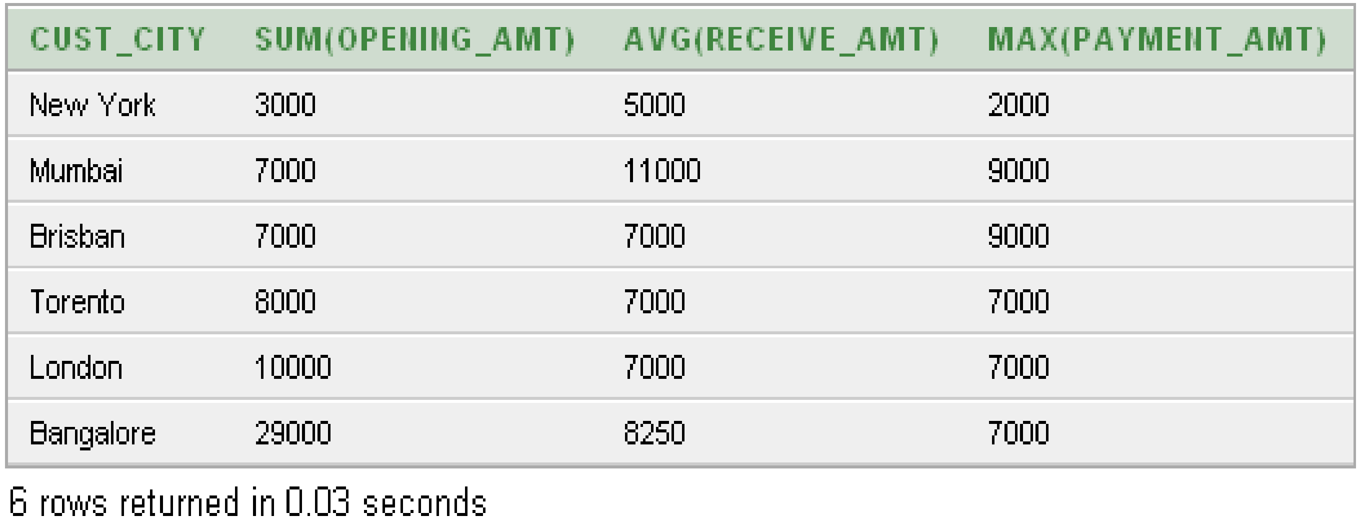
CUST_CITY	SUM(OPENING_AMT)	AVG(RECEIVE_AMT)	MAX(PAYMENT_AMT)
Torento	8000	7000	7000
London	10000	7000	7000
New York	3000	5000	2000
Brisban	7000	7000	9000
Bangalore	29000	8250	7000
Mumbai	7000	11000	9000

6 rows returned in 0.30 seconds

HAVING() Clause

SQL HAVING with order by:

```
SELECT cust_city,SUM(opening_amt),  AVG(receive_amt),
MAX(payment_amt)  FROM customer  WHERE grade=2
GROUP BY cust_city  HAVING AVG(receive_amt)>500  ORDER BY SUM(opening
_amt);
```



CUST_CITY	SUM(OPENING_AMT)	AVG(RECEIVE_AMT)	MAX(PAYMENT_AMT)
New York	3000	5000	2000
Mumbai	7000	11000	9000
Brisban	7000	7000	9000
Torento	8000	7000	7000
London	10000	7000	7000
Bangalore	29000	8250	7000

6 rows returned in 0.03 seconds

Difference between where & Having clause

- The **WHERE** clause specifies the criteria which individual records must meet to be selected by a query. It can be used without the GROUP BY clause.

- The **HAVING** clause cannot be used without the GROUP BY clause.

- The **WHERE** clause selects rows *before* grouping. The **HAVING** clause selects rows *after* grouping.

- The **WHERE** clause *cannot* contain aggregate functions.
  The **HAVING** clause *can* contain aggregate functions.

Summary

This session will give the knowledge about

- SQL CLAUSES
  - Where clause
  - Order by clause
  - Aggregate Functions
  - Group by clause
  - Having clause