Aggregate Function and Group by Clause

Learning Objective

After completing this lab the student should be able to:

- Identify the available group functions(Aggregate functions)
- Use of Group functions.
- Group data by using group by clause
- Having clause restriction on group rows
- Difference between Having clause and Where clause.

Tools and Technologies

- Oracle Database 11g Express Edition/Enterprise Edition.

Oracle Credentials for Lab

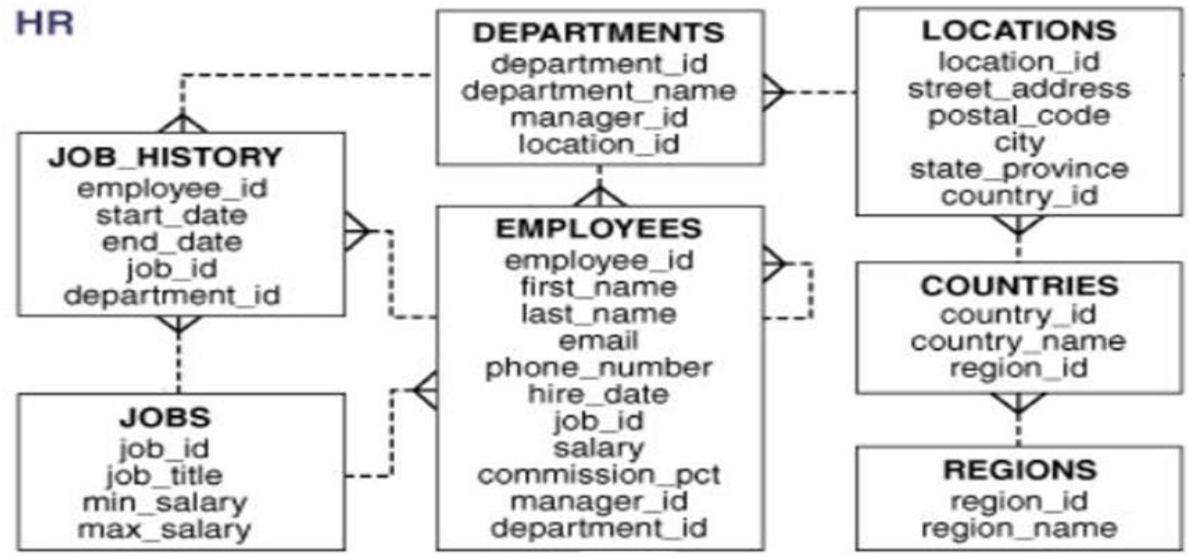
Enter the Url in your browser **http://172.17.10.114:8080/apex/**

Username **hr**

Password **hr**

HR Schema

.



HR Table Descriptions

Table **COUNTRIES**

Name	Null?	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

Table **DEPARTMENTS**

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

Table **EMPLOYEES**

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE		NOT NULL DATE
JOB_ID		NOT NULL VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

Table JOBS

Name	Null?	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

Table JOB_HISTORY

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

Table LOCATIONS

Name	Null?	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

Table REGIONS

Name	Null?	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

Group function

So far we have studied single-row functions, which accept one or more arguments and return one value for each row returned by the query. In group functions we will operate on sets of rows to give one result per group. These sets may be the whole table or the table split into groups.

Types of group functions

- AVG
- COUNT
- MAX
- MIN

- SUM

Each of the functions accepts an argument. The following table identifies the options that you can use in the syntax.

Function	Description
AVG ([DISTINCT|ALL] n)	Average value of n, ignoring null values
COUNT ({*[DISTINCT|ALL] *expr*})	Number of rows, where *expr* evaluates to something other than null (Count all selected rows using * including duplicates rows with nulls)
MAX ([DISTINCT|ALL] *expr*)	Maximum value of *expr*, ignoring null values
MIN ([DISTINCT|ALL] *expr*)	Minimum value of *expr*, ignoring null values
SUM ([DISTINCT|ALL] n)	Sum values of n, ignoring null values

Guidelines for Using Group Functions

- DISTINCT makes the function consider only non-duplicate values.
- ALL makes it consider every value including duplicates. The default is ALL and therefore does not need to be specified.
- The data types for the arguments may be CHAR, VARCHAR2, NUMBER, or DATE where expr is listed.
- All group functions except COUNT (*) ignore null values. To substitute a value for null values use the NVL function.

Example

```
SELECT    AVG (salary), MAX(salary), MIN(salary), SUM(salary)
FROM      employees
WHERE     job_id LIKE '%CLERK';
```

Aggregate Function Apply on numeric data types.

You can use MAX and MIN functions for any datatypes.

Example

```
SELECT MIN (hire_date), MAX(hire_date)
FROM        employees;
```

Note: AVG, SUM functions can be used only with numeric data types.

Count Function

The COUNT function has two functions:

- COUNT (*) returns the number of rows in a table, including duplicate rows and rows containing null values in any of the columns. If a WHERE clause is included in the SELECT statement, COUNT (*) returns the number of rows that satisfies the condition in the WHERE clause.
- In contrast, COUNT (expr) returns the number of non null rows in the column identified by expr.

Example

Display the number of employees in department 30.

```
SELECT COUNT(*)
FROM        employees
WHERE department_id=30;
```

To display the number of department in the EMPLOYEES table.

```
SELECT  COUNT(department_id)
FROM        employees;
```

To display the number of distinct departments in the EMPLOYEES table.

```
SELECT COUNT (DISTINCT (department_id))
FROM        employees;
```

Group By Clause

We can use the GROUP BY clause to divide the rows in a table into groups. You can then use the group functions to return summary information for each group. The SQL GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.

Syntax:
SELECT column, group_function(column)
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[ORDER BY column];

Note:

- If you include a group function in a SELECT clause, you cannot select individual results as well unless the individual column appears in the GROUP BY clause. You will receive an error message if you fail to include the column list.
- Using a WHERE clause, you can pre exclude rows before dividing them into groups.
- You must include the columns in the GROUP BY clause.
- You cannot use the column alias in the GROUP BY clause.
- By default rows are sorted by ascending order of the columns included in the GROUP BY list. You can override this by using the ORDER BY clause.

Using the group by clause

When using the GROUP BY clause, make sure that all columns in the SELECT list that are not in the group functions are included in the GROUP BY clause.

Example

```
SELECT department_id, AVG(salary)
FROM  employees
GROUP BY   department_id;
```

The example above displays the department number and the average salary for each department. Here is how this SELECT statement, containing a GROUP BY clause, is evaluated:

- The SELECT clause specifies the columns to be retrieved;
    - Department number column in the EMPLOYEES table

  - The average of all the salaries in the group you specified in the GROUP BY clause

- The FROM clause specifies the table that the database must access: the EMPLOYEES table.
- The WHERE clause specifies the rows to retrieved. Since there is no WHERE clause, by default all rows are retrieved.

The GROUP BY clause specifies how the rows should be grouped. The rows are being grouped by department number, so the AVG function that is being applied to the salary column will calculate the average salary for each department.

The GROUP BY column does not have to be in the SELECT clause.

```
SELECT AVG (salary)
FROM        employees
GROUP  BY   department_id;
```

Grouping by more than one column

You can return summary results for groups and subgroups by listing more than one GROUP BY column. You can determine the default sort order of the results by the order of the columns in the GROUP BY clause.

Example

```
SELECT department_id, job_id, sum (salary)
FROM employees
GROUP BY department_id, job_id;
```

Explanation

- The SELECT clause specifies the column to retrieved:
  - Department number in the EMPLOYEES table
  - Job title in the EMPLOYEES table
  - The sum of all the salaries in the group that you specified in the GROUP BY clause

- The FROM clause specifies the tables that the database must access: the EMPLOYEES table
- The GROUP BY clause specifies how you must group the rows:
  - First, the rows are grouped by department number.
  - Second, within the department number groups, the rows are grouped by job title.

So the SUM function is being applied to the salary column for all job titles within each department number group.

Having clause

As we use the WHERE clause to restrict the rows that we select, we use the HAVING clause to restrict groups.

You use the HAVING clause to specify which groups are to be displayed. Therefore, you further restrict the groups on the basis of aggregate information.

Syntax

```
SELECT      column, group_function
FROM        table;

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

The Oracle Server performs the following steps when you use the HAVING clause:

- Rows are grouped.
- The group function is applied to the group.
- The groups that match the criteria in the HAVING clause are displayed.

The HAVING clause can precede the GROUP BY clause, but it is recommended that you place the GROUP BY clause first because it is more logical. Groups are formed and group functions are calculated before the HAVING clause is applied to the groups in the SELECT list.

Example

```
SELECT department_id, max (salary)
FROM employees
GROUP BY department_id
HAVING max (salary)>8000;
```

This example displays department numbers and maximum salary for those department whose maximum salary is greater than 2900.

You can use the GROUP BY clause without using a group function in the SELECT list. If you restrict rows based on the result of a group function, you must have a GROUP BY clause as well as the HAVING clause.

Example

```
SELECT job_id, SUM (salary) PAYROLL
FROM employees
WHERE job  id NOT LIKE '% ‘CLERK'
GROUP BY job_id
HAVING SUM(salary)>5000
ORDER BY SUM(salary);
```

The above example displays the job title and total monthly salary for each job title with a total payroll exceeding 5000. It also exclude salespeople and sorts the list by the total monthly salary.

Nesting group functions

Group functions can be nested to a depth of two.

Example

```
SELECT max(avg (salary))
FROM employees
GROUP BY department_id;
```

Order of evaluation of the clauses

1. WHERE clause
2. GROUP BY clause
3. HAVING clause

Lab Exercise

1. You are required to read the lab manual and implement all the queries mentioned in the manual. (4 Marks)

2. Show the average salary of all those employees whose first_name contains 'a' and 'e';
   (3 Marks)

3. Display the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results up to the two decimal position.
   (3 Marks)

4. Modify the above questions to display the minimum, maximum, sum,

and average salary for each job type. (3 marks)

5. Write a query to display the number of people with the same job. (2 Marks).

Lab Instructor:

Qazi Shuja ud Din (Riphah International University)
Email: qazi.shujauddin@riphah.edu.pk