Lab – 06

**Aggregate Functions, Group By & Having Clause**

# Objective:

* Students will be able to implement function on group of records and implement multiple tasks using aggregate function.

# Aggregate Functions

An SQL group function or aggregate functions performs an operation on a set of rows and returns a single result. You may want to find group of item-prices and return total-price. This type of scenario is where you would use a group function.

**The following table is summary of some SQL group functions & query examples.**

|  |  |  |
| --- | --- | --- |
| **Function** | **Description** | **Query Example** |
| **AVG(fieldname)** | Returns average value of a column | SELECT avg(price)FROM inventory; |
| **COUNT(fieldname) COUNT(\*)** | Returns number of items in table or queried  items | SELECT count(product\_id)FROM product;  SELECT count(\*) FROM product; |
| **MAX(fieldname)** | Returns maximum value of a column | SELECT max(price)FROM inventory; |
| **MIN(fieldname)** | Returns minimum value of a column | SELECT min(price)FROM inventory; |
| **SUM(fieldname)** | Returns total value of a column | SELECT sum(price)FROM inventory; |
| **STDDEV** | Returns standard deviation | SELECT stddev(price)FROM inventory; |
| **VARIANCE** | Returns the variance | SELECT variance(price)FROM inventory; |

# Examples of Functions:

Find min, max, avg and total price from given table

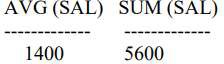
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product ID** | **Name** | **Description** | **Price** | **Cost** |
| 100000000 | Printer | Inkjet 300 color Printer | 120 | 80 |
| 100000001 | Printer | 1220CXI Inkjet Printer | 200 | 130 |
| 100000002 | Printer | Photo 890 Inkjet Printer | 250 | 200 |
| 100000003 | Printer | Photo 890 Inkjet Printer | 300 | 270 |

1. **AVG and SUM Function: (**used for numeric data type)

SELECT AVG (sal), SUM (sal)

FROM emp WHERE sal= 1250;

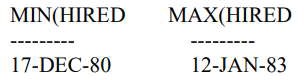
# Output:



1. **MIN and MAX Function:** (used for any data type)

SELECT MIN (hiredate), MAX (hiredate) FROM emp*;*

# Output:



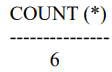
1. **COUNT Function:** (It returns the number of rows in a table)

SELECT COUNT (\*)

FROM emp

WHERE deptno = 30;

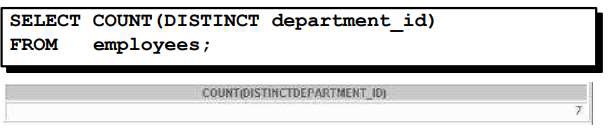
# Output:



The COUNT function has three formats:

* + COUNT(\*)
  + COUNT(expr)
  + COUNT(DISTINCT expr)

COUNT(DISTINCT expr) returns the number of distinct non null values of the expr. • Display the number of distinct department values in the EMPLOYEES table.



# Group By Clause

**Group By** is used to categorize the retrieved data. For example, you may want to list sales of each product identified by product id. To do this, the following is Group By example that lists.

# Example:

Consider the following database table called Employee Hours storing the daily hours for each employee of a factious company

|  |  |  |
| --- | --- | --- |
| **Employee** | **Date** | **Hours** |
| John Smith | 5/6/2004 | 8 |
| Allan Babel | 5/6/2004 | 8 |
| Tina Crown | 5/6/2004 | 8 |
| John Smith | 5/7/2004 | 9 |
| Allan Babel | 5/7/2004 | 8 |
| Tina Crown | 5/7/2004 | 10 |
| John Smith | 5/8/2004 | 8 |
| Allan Babel | 5/8/2004 | 8 |
| Tina Crown | 5/8/2004 | 9 |

If the manager of the company wants to get the simple sum of all hours worked by all employees, he needs to execute the following SQL statement:

SELECT SUM (Hours)

FROM EmployeeHours

But what if the manager wants to get the sum of all hours for each of his employees? Then use the **SQL GROUP BY** statement:

SELECT Employee, SUM (Hours) FROM EmployeeHours

GROUP BY Employee

# Output:

|  |  |
| --- | --- |
| **Employee** | **Hours** |
| John Smith | 25 |
| Allan Babel | 24 |
| Tina Crown | 27 |

As you can see we have only one entry for each employee, because we are grouping by the Employee column.

The **SQL GROUP BY** clause can be used with other SQL aggregate functions, for example SQL AVG:

SELECT Employee, AVG (Hours) FROM EmployeeHours GROUP BY Employee

# Output:

|  |  |
| --- | --- |
| **Employee** | **Hours** |
| John Smith | 8.33 |
| Allan Babel | 8 |
| Tina Crown | 9 |

In Employee table we can group by the date column too, to find out what is the total number of hours worked on each of the dates into the table:

SELECT Date, SUM (Hours) FROM EmployeeHours GROUP BY Date

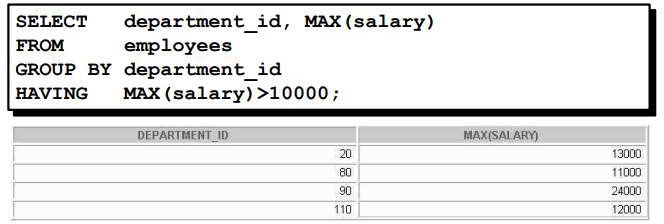
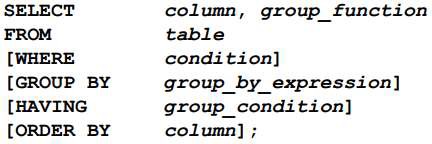
# Output:

|  |  |
| --- | --- |
| **Date** | **Hours** |
| 5/6/2004 | 24 |
| 5/7/2004 | 27 |
| 5/8/2004 | 25 |

**HAVING Clause**

Use the HAVING clause to restrict groups:

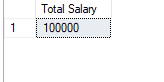
1. Rows are grouped.
2. The group function is applied.
3. Groups matching the HAVING clause are displayed.



# Lab Tasks:

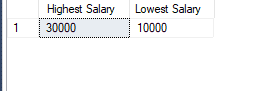
* 1. Display the total salary amount being dispersed every month of an organization assuming the records are placed in EMP table.

SELECT SUM(salary) AS 'Total Salary' FROM Employee;



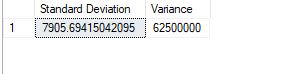
* 1. Show the lowest and highest salary of employees.

SELECT MAX(salary) AS ' Highest Salary',MIN(salary) As 'Lowest Salary' FROM Employee;



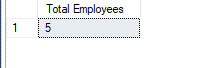
* 1. Display the variation using standard deviation and variance in salaries of employees.

SELECT STDEV(salary) AS 'Standard Deviation',Var(salary) As 'Variance' FROM Employee;



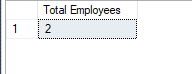
* 1. Show the total number of employees in an organization.

SELECT COUNT(\*) As ' Total Employees' FROM Employee;



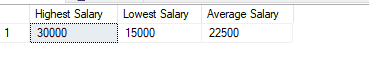
* 1. Write a query to display the total number of employees in dept no 2.

SELECT COUNT(\*) As 'Total Employees' FROM Employee WHERE dept\_no = 2;



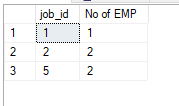
* 1. Display the highest, lowest & average salaries of employees who were hired in July and August month.

SELECT MAX(salary) AS 'Highest Salary',MIN(Salary) AS 'Lowest Salary',AVG(Salary) AS 'Average Salary' FROM Employee WHERE MONTH(hire\_date) = 06 OR MONTH(hire\_date) = 07 ;



* 1. Write a query to display the number of people with the same job.

SELECT job\_id,COUNT(empID) AS 'No of EMP'FROM Employee Group By(job\_id);



* 1. Determine the number of managers without listing them. Label the column Number of Managers. Hint: Use the MANAGER\_ID column to determine the number of managers.

SELECT COUNT(DISTINCT Manager\_ID) AS 'Number of Managers' FROM Employee;

A close-up of a number of marks

Description automatically generated

* 1. Write a query that displays the difference between the highest and lowest salaries. Label the column DIFFERENCE.

SELECT MAX(Salary)-MIN(Salary) AS 'DIFFERENCE' FROM Employee;

A close-up of a number

Description automatically generated

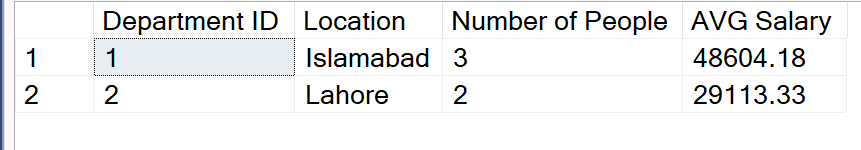
* 1. Display the manager number and the salary of the lowest paid employee for that manager. Exclude anyone whose manager is not known. Exclude any groups where the minimum salary is less than $8,000. Sort the output in descending order of salary.

SELECT Manager\_ID , MIN(salary) AS 'Lowest Paid' FROM Employee WHERE Manager\_ID IS NOT NULL GROUP BY Manager\_ID HAVING MIN(Salary)>8000 ORDER BY MIN(Salary) DESC;

A screenshot of a computer

Description automatically generated

* 1. Write a query to display each department’s name, location, number of employees, and the average salary for all employees in that department. Label the columns Name, Location, Number of People, and Salary, respectively. Round the average salary to two decimal places.

SELECT Department\_ID AS 'Department ID',Location,COUNT(emp\_ID) AS 'Number of People',ROUND(AVG(Salary),2) AS 'AVG Salary' FROM EMPLOYEE GROUP BY Department\_ID,Location;

* 1. Create a query that will display the total number of employees and, of that total, the number of employees hired in 2010, 2011, 2012, and 2013. Create appropriate column headings.
  2. SELECT Count(emp\_ID) AS 'TotalEmployee', YEAR(hire\_date) AS 'Years' FROM EMPLOYEE WHERE YEAR(hire\_date) BETWEEN 2010 AND 2014 GROUP BY YEAR(hire\_date);
  3. A screenshot of a data

     Description automatically generated