**National University of Computer & Emerging Sciences, Peshawar **

**Computer Science Department**

**Spring 2025, Lab Manual - 04**

|  |  |
| --- | --- |
| **Course Code: CL-2005** | **Course : Database Systems Lab** |
| **Instructor Name :** | **Yasir Arfat** |

**Contents:**

* Groups of Data (Group by, Having)
* Sub Queries (Single Row, Multiple and correlated)
* Sub Queries and DML
* Tasks

**Group by Statement:**

The GROUP BY statement group’s rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

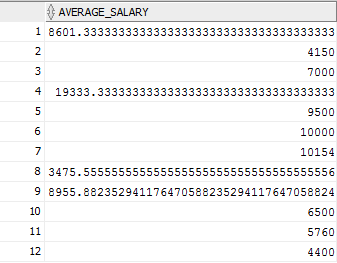
**Group by Syntax**

|  |
| --- |
| SELECT *column\_name(s)* FROM *table\_name* GROUP BY *column\_name(s)* |

Group by:

|  |
| --- |
| SELECT      AVG(salary) as “average\_salary”  FROM      employees  GROUP BY Department\_id |

Sample Output:



**Group by (Having)**

HAVING Clause is used with GROUP BY Clause to restrict the groups of returned rows where condition is TRUE.

Syntax:

|  |
| --- |
| SELECT expression1, expression2, ... expression\_n,   aggregate\_function (aggregate\_expression)  FROM [table](https://www.javatpoint.com/oracle-having-clause)s  WHERE conditions  GROUP BY expression1, expression2, ... expression\_n  HAVING having\_condition; |

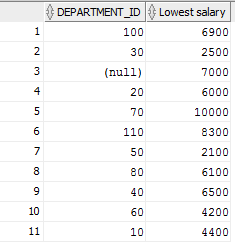
**HAVING Example: (with GROUP BY SUM function)**

|  |
| --- |
| SELECT department\_id, SUM(salary) AS "Total Salary"  FROM employees  GROUP BY department\_id  HAVING SUM(salary) < 100000; |

**HAVING Example: (with GROUP BY MIN function)**

|  |
| --- |
| SELECT Department\_ID,  MIN(salary) AS "Lowest salary"  FROM employees  GROUP BY Department\_ID  HAVING MIN(salary) < 15000; |

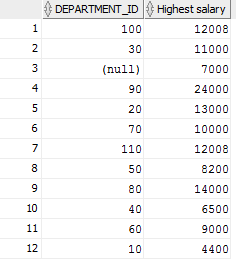
Sample Output:



**HAVING Example: (with GROUP BY MAX function)**

|  |
| --- |
| SELECT Department\_ID,  MAX(salary) AS "Highest salary"  FROM employees  GROUP BY Department\_ID  HAVING MAX(salary) > 3000; |

Sample Output:



**Sub Queries:**

A Subquery is a query within another SQL query and embedded within the WHERE clause.

Important Rule:

* A subquery can be placed in a number of SQL clauses like WHERE clause, FROM clause, HAVING clause.
* You can use Subquery with SELECT, UPDATE, INSERT, DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
* A subquery is a query within another query. The outer query is known as the main query, and the inner query is known as a subquery.
* Subqueries are on the right side of the comparison operator.
* A subquery is enclosed in parentheses.
* In the Subquery, ORDER BY command cannot be used. But GROUP BY command can be used to perform the same function as ORDER BY command.

**NOTE**:

Subqueries are useful when a query is based on unknown values.

**Sub Queries with SELECT Statement:**

Syntax:

|  |
| --- |
| SELECT column\_name  FROM table\_name  WHERE column\_name expression operator  ( SELECT column\_name  from table\_name WHERE ... ); |

**Types of Subqueries**:

1. **Single Row Sub Query**: Sub query which returns single row output. They mark the usage of single row comparison operators, when used in WHERE conditions.
2. **Multiple row sub query**: Sub query returning multiple row output. They make use of multiple row comparison operators like IN, ANY, ALL. There can be sub queries returning multiple columns also.
3. **Correlated Sub Query**: Correlated subqueries depend on data provided by the outer query. This type of subquery also includes subqueries that use the EXISTS operator to test the existence of data rows satisfying specified criteria.

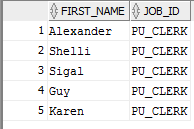
**Single Row Sub Queries:**

* Return only one row
* Use single-row comparison operators

|  |  |
| --- | --- |
| Operator | Meaning |
| = | Equal to |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |
| <> , =! | Not equal to |

|  |
| --- |
| SELECT First\_Name, Job\_ID FROM Employees WHERE job = ( SELECT job\_ID FROM Employees WHERE empno=7369 ) |

Sample Output:



**Single Row Functions:**

Finds the employees who have the highest salary:

|  |
| --- |
| SELECT      employee\_id, first\_name, last\_name, salary  FROM      employees  WHERE      salary = (SELECT              MAX(salary)          FROM              employees) |

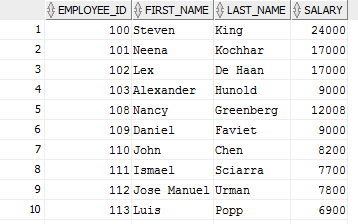
Sample Output:



**Finds all employees who salaries are greater than the average salary of all employees:**

|  |
| --- |
| SELECT      employee\_id, first\_name, last\_name, salary  FROM      employees  WHERE      salary > (SELECT              AVG(salary)          FROM              employees) |

Sample Output:



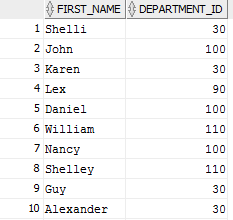
**Multiple row sub query:**

* Return more than one row
* Use multiple-row comparison operators
* [> ALL] More than the highest value returned by the subquery
* [< ALL] Less than the lowest value returned by the subquery
* [< ANY] Less than the highest value returned by the subquery
* [> ANY] More than the lowest value returned by the subquery
* [= A NY] Equal to any value returned by the subquery (same as IN)

**IN:**

|  |
| --- |
| SELECT first\_name, department\_id  FROM employees  WHERE department\_id IN (SELECT Department\_id  FROM departments  WHERE LOCATION\_ID = 100) |

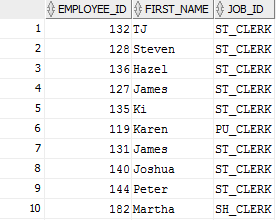
**Sample Output:**



**ANY:**

|  |
| --- |
| SELECT employee\_ID, First\_Name, job\_ID FROM EMPLOYEES WHERE SALARY < ANY  ( SELECT salary FROM EMPLOYEES WHERE JOB\_ID = 'PU\_CLERK' ); |

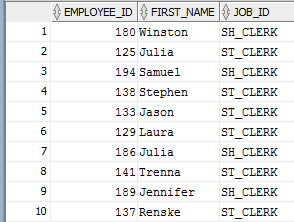
**Sample Output:**



**ALL:**

|  |
| --- |
| SELECT employee\_ID, First\_Name, job\_ID FROM EMPLOYEES WHERE SALARY >All  ( SELECT salary FROM HR.EMPLOYEES WHERE JOB\_ID = 'PU\_CLERK' ) AND job\_ID <> 'PU\_CLERK' ; |

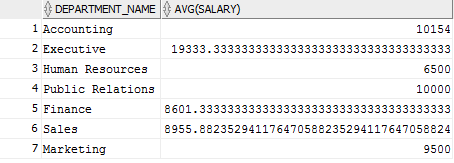
Sample Output:



**Group By and HAVING IN SUB QUERIES:**

|  |
| --- |
| SELECT department\_name, avg(salary)  FROM EMP\_DETAILS\_VIEW  GROUP BY department\_name  HAVING avg(salary) > (  SELECT avg(salary)  FROM EMPLOYEES  ); |

**Sample Output:**



**SUBQUERIES AND DML:**

Subqueries with the INSERT Statement

* SQL subquery can also be used with the Insert statement. In the insert statement, data returned from the subquery is used to insert into another table.
* In the subquery, the selected data can be modified with any of the character, date functions.

**Syntax:**

|  |
| --- |
| INSERT INTO table\_name (column1, column2, column3....)  SELECT \*  FROM table\_name  WHERE VALUE OPERATOR |

***You may login from a new user for DML sub Queries.***

Example: Let's assume we have an EMPLOYEE\_BKP table available which is backup of EMPLOYEE table having all the attributes of Employees table

|  |
| --- |
| INSERT INTO EMPLOYEE\_BKP  SELECT \* FROM EMPLOYEES  WHERE job\_ID IN (SELECT job\_id  FROM jobs WHERE job\_title='Accountant'); |

**Subqueries with the UPDATE Statement**

The subquery of SQL can be used in conjunction with the Update statement. When a subquery is used with the Update statement, then either single or multiple columns in a table can be updated.

**Syntax**

|  |
| --- |
| UPDATE table  SET column\_name = new\_value  WHERE VALUE OPERATOR     (SELECT COLUMN\_NAME     FROM TABLE\_NAME     WHERE condition); |

**Example:**

The given example updates the SALARY by 10 times in the EMPLOYEE table for all employee whose minimum salary is 3000.

|  |
| --- |
| Update employees  set salary= salary+(0.1\*salary)  WHERE job\_ID IN (SELECT job\_ID  FROM jobs WHERE min\_salary=3000); |

**Subqueries with the DELETE Statement**

The subquery of SQL can be used in conjunction with the Delete statement just like any other statements mentioned above.

**Syntax**

|  |
| --- |
| DELETE FROM TABLE\_NAME  WHERE VALUE OPERATOR     (SELECT COLUMN\_NAME     FROM TABLE\_NAME     WHERE condition); |

**Example:**

Let's assume we have an EMPLOYEE\_BKP table available which is backup of EMPLOYEE table. The given example deletes the records from the EMPLOYEE\_BKP table for all EMPLOYEE whose end date is ’31-DEC-06’.

|  |
| --- |
| Delete from employee\_BKP  WHERE job\_ID IN (SELECT job\_ID  FROM job\_History WHERE end\_Date='31-Dec-06'); |

|  |
| --- |
| SELECT  e.employee\_id,  e.first\_name,  e.last\_name,  (SELECT job\_title FROM jobs WHERE job\_id = e.job\_id) AS job\_title,  (SELECT department\_name FROM departments WHERE department\_id = e.department\_id) AS department\_name,  (SELECT city FROM locations WHERE location\_id = d.location\_id) AS department\_location,  (SELECT region\_name FROM regions WHERE region\_id = r.region\_id) AS region\_name  FROM  employees e,  departments d,  locations l,  regions r  WHERE  e.department\_id = d.department\_id  AND d.location\_id = l.location\_id; |