Chapter-4

What is Group Function?

Group functions operate on sets of rows to give one result per group. These sets may comprise the entire table or the table split into Groups. All group functions ignore null values in the column. The NVL function forces group functions to include null values (SELECT AVG(NVL(commission_pct, 0)) FROM employees;).

Example: SELECT avg(salary) from employees where job_id like '%RER%';

What are the type of Group Functions?

Types of Group Functions

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Guidelines for Using Group Functions:

- DISTINCT makes the function consider only nonduplicate 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 functions with an expr argument may be CHAR, VARCHAR2, NUMBER, or DATE.
- All group functions ignore null values. To substitute a value for null values, use the NVL, NVL2, or COALESCE functions.

How to use AVG and SUM?

Avg and Sum:
You can use AVG and SUM for numeric data.
Example: SELECT AVG(salary), MAX(salary), MIN(salary), SUM(salary) FROM employees WHERE job_id LIKE '%REP%';

How to use MIN and MAX?

Min and max: You can use MIN and MAX for numeric, character, and date data types.
Example: SELECT MIN(hire_date), MAX(hire_date) FROM employees;

Note: The AVG, SUM, VARIANCE, and STDDEV functions can be used only with numeric data types. MAX and MIN cannot be used with LOB or LONG data types.

What are uses of Count?

COUNT(*) returns the number of rows in a table. COUNT(expr) returns the number of rows with non-null values for expr.
Example: SELECT COUNT(*) FROM employees WHERE department_id = 50;

The COUNT function has three formats:

- COUNT(*)
- COUNT(expr)
- COUNT(DISTINCT expr)

Using the DISTINCT Keyword

- COUNT(DISTINCT expr) returns the number of distinct non-null values of expr.
- To display the number of distinct department values in the EMPLOYEES table:

What is Group by Clause?

GROUP BY Clause:

The Oracle GROUP BY clause is used in a SELECT statement to collect data across multiple records and group the results by one or more columns. We can use the GROUP BY clause to divide the rows in a table into groups. We can then use the group functions to return summary information for each group. The GROUP BY column does not have to be in the SELECT list.

Guidelines

• 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 receive an error message if you fail to include the column list in the GROUP BY clause.
• Using a WHERE clause, you can exclude rows before dividing them into groups.
• You must include the columns in the GROUP BY clause.
• You cannot use a column alias in the GROUP BY clause.

Illegal Queries Using Group Functions:

• You cannot use the WHERE clause to restrict groups.
• You use the HAVING clause to restrict groups.
• You cannot use group functions in the WHERE clause.

Uses of Having Clause

The Oracle server performs the following steps when you use the HAVING clause:
1. Rows are grouped.
2. The group function is applied to the group.
3. The groups that match the criteria in the HAVING clause are displayed.

N.B Place the HAVING and GROUP BY clauses after the WHERE clause in a statement. The order of the HAVING and GROUP clauses following the WHERE clause is not important. Place the ORDER BY clause last.

Chapter-5

Types of Joins

Joins that are compliant with the SQL:1999 standard include the following:
• Cross joins
• Natural joins
• USING clause
• Full (or two-sided) outer joins
• Arbitrary join conditions for outer joins

Creating Natural Joins:

• The NATURAL JOIN clause is based on all columns in the two tables that have the same name.
• It selects rows from the two tables that have equal values in all matched columns.
• If the columns having the same names have different data types, an error is returned.

Creating Joins with the USING Clause

• If several columns have the same names but the data types do not match, natural join can be applied by using the USING clause to specify the columns that should be used for an equijoin.
• Use the USING clause to match only one column when more than one column matches.
• Do not use a table name or alias in the referenced columns.
• The NATURAL JOIN and USING clauses are mutually exclusive.

Using Table Aliases:

• Use table aliases to simplify queries.
• Use table aliases to improve performance.

Guidelines

• Table aliases can be up to 30 characters in length, but shorter aliases are better than longer ones.
• If a table alias is used for a particular table name in the FROM clause, then that table alias must be substituted for the table name throughout the SELECT statement.
• Table aliases should be meaningful.
• The table alias is valid for only the current SELECT statement.

Creating Joins with the ON Clause

• The join condition for the natural join is basically an equijoin of all columns with the same name.
• Use the ON clause to specify arbitrary conditions or specify columns to join.
• The join condition is separated from other search conditions.
• The ON clause makes code easy to understand.

What is self-join?

A self-join is a join in which a table is joined with itself. To join a table itself means that each row of the table is combined with itself and with every other row of the table. The table appears twice in the FROM clause and is followed by table aliases that qualify column names in the join condition. The self-join can be viewed as a join of two copies of the same table. The table is not actually copied, but SQL performs the command as though it were. To perform a self-join, Oracle Database combines and returns rows of the table that satisfy the join condition

Nonequijoins

A nonequijoin is a join condition containing something other than an equality operator.
Example: SELECT e.last_name, e.salary, j.grade_level FROM employees e JOIN job_grades j ON e.salary BETWEEN j.lowest_sal AND j.highest_sal;

Outer Join:

An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other table satisfy the join condition.
There are three types of outer joins:
• LEFT OUTER
• RIGHT OUTER
• FULL OUTER

LEFT OUTER JOIN operation

A LEFT OUTER JOIN is one of the join operations that allow you to specify a join clause. It preserves the unmatched rows from the first (left) table, joining them with a NULL row in the shape of the second (right) table.
Example: SELECT e.last_name, e.department_id, d.department_name FROM employees e LEFT OUTER JOIN departments d ON (e.department_id = d.department_id) ;

RIGHT OUTER JOIN operation

A RIGHT OUTER JOIN is one of the join operations that allow you to specify a JOIN clause. It preserves the unmatched rows from the second (right) table, joining them with a NULL in the shape of the first (left) table. A LEFT OUTER JOIN B is equivalent to B RIGHT OUTER JOIN A, with the columns in a different order.
Example: SELECT e.last_name, e.department_id, d.department_name FROM employees e RIGHT OUTER JOIN departments d ON (e.department_id = d.department_id) ;

What is full outer join in Oracle?

A full outer join performs a join between two tables that returns the results of an INNER join as well as the results of a left and right outer join.
Example: SELECT e.last_name, d.department_id, d.department_name FROM employees e FULL OUTER JOIN departments d ON (e.department_id = d.department_id) ;

Cartesian Products

• A Cartesian product is formed when:
– A join condition is omitted
– A join condition is invalid
– All rows in the first table are joined to all rows in the second table
• To avoid a Cartesian product, always include a valid join condition.

CROSS JOIN operation

A CROSS JOIN is a JOIN operation that produces the Cartesian product of two tables. Unlike other JOIN operators, it does not let you specify a join clause. You may, however, specify a WHERE clause in the SELECT statement.
Example: SELECT last_name, department_name FROM employees CROSS JOIN departments ;

Creating Cross Joins

• The CROSS JOIN clause produces the cross-product of two tables.
• This is also called a Cartesian product between the two tables.