**Group By, Having Clause, CASE Statement, Exist Operator, Insert into select , CTAS**

---------------------------------------------------------------------------------------------

/\*

The SQL GROUP BY Statement

The GROUP BY statement groups 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

WHERE condition

GROUP BY column\_name(s)

ORDER BY column\_name(s);

Demo Database

Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID CustomerName ContactName Address City PostalCode Country

1

Alfreds Futterkiste Maria Anders Obere Str. 57 Berlin 12209 Germany

2 Ana Trujillo Emparedados y helados Ana Trujillo Avda. de la Constitución 2222 México D.F. 05021 Mexico

3 Antonio Moreno Taquería Antonio Moreno Mataderos 2312 México D.F. 05023 Mexico

4

Around the Horn Thomas Hardy 120 Hanover Sq. London WA1 1DP UK

5 Berglunds snabbköp Christina Berglund Berguvsvägen 8 Luleå S-958 22 Sweden

\*/

-- The following SQL statement lists the number of customers in each country:

SELECT COUNT(CustomerID), Country

FROM Customers

GROUP BY Country;

SELECT CustomerID, Country

FROM Customers order by Country;

-- The following SQL statement lists the number of customers in each country, sorted high to low:

SELECT COUNT(CustomerID), Country

FROM Customers

GROUP BY Country

ORDER BY COUNT(CustomerID) DESC;

-- GROUP BY With JOIN

-- The following SQL statement lists the number of orders sent by each shipper:

SELECT COUNT(Orders.OrderID) AS NumberOfOrders

FROM Orders

LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID

GROUP BY ShipperName;

---------------------------------------------------------------------------------------------

-- Having Clause

---------------------------------------------------------------------------------------------

/\*

The SQL HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

HAVING Syntax

SELECT column\_name(s)

FROM table\_name

WHERE condition

GROUP BY column\_name(s)

HAVING condition

ORDER BY column\_name(s);

Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID CustomerName ContactName Address City PostalCode Country

1

Alfreds Futterkiste Maria Anders Obere Str. 57 Berlin 12209 Germany

2 Ana Trujillo Emparedados y helados Ana Trujillo Avda. de la Constitución 2222 México D.F. 05021 Mexico

3 Antonio Moreno Taquería Antonio Moreno Mataderos 2312 México D.F. 05023 Mexico

4

Around the Horn Thomas Hardy 120 Hanover Sq. London WA1 1DP UK

5 Berglunds snabbköp Christina Berglund Berguvsvägen 8 Luleå S-958 22 Sweden

\*/

-- The following SQL statement lists the number of customers in each country. Only include countries with more than 5 customers:

SELECT COUNT(CustomerID) as customers\_count, Country

FROM Customers

GROUP BY Country

HAVING customers\_count > 5;

-- The following SQL statement lists the number of customers in each country, sorted high to low (Only include countries with more than 5 customers):

SELECT COUNT(CustomerID), Country

FROM Customers

GROUP BY Country

HAVING COUNT(CustomerID) > 5

ORDER BY COUNT(CustomerID) DESC;

-- The following SQL statement lists the employees that have registered more than 10 orders:

SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders

FROM (Orders

INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID)

GROUP BY LastName

HAVING NumberOfOrders > 10;

-- The following SQL statement lists if the employees "Davolio" or "Fuller" have registered more than 25 orders:

SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders

FROM Orders

INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID

WHERE LastName = 'Davolio' OR LastName = 'Fuller'

GROUP BY LastName

HAVING COUNT(Orders.OrderID)>20

ORDER BY COUNT(Orders.OrderID) DESC;

---------------------------------------------------------------------------------------------

-- CASE Statement

---------------------------------------------------------------------------------------------

/\*

The SQL CASE Expression

The CASE expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

If there is no ELSE part and no conditions are true, it returns NULL.

CASE Syntax

CASE

WHEN condition1 THEN result1

WHEN condition2 THEN result2

WHEN conditionN THEN resultN

ELSE result

END;

Below is a selection from the "OrderDetails" table in the Northwind sample database:

OrderDetailID OrderID ProductID Quantity

1 10248 11 12

2 10248 42 10

3 10248 72 5

4 10249 14 9

5 10249 51 40

\*/

-- The following SQL goes through conditions and returns a value when the first condition is met:

SELECT OrderID, Quantity,

CASE

WHEN Quantity > 30 THEN 'The quantity is greater than 30'

WHEN Quantity = 30 THEN 'The quantity is 30'

ELSE 'The quantity is under 30'

END AS QuantityText,

OrderDetailID

FROM Order\_Details;

-- The following SQL will order the customers by City. However, if City is NULL, then order by Country:

SELECT CustomerName, POSTALCODE, Country,CIty

FROM Customers

ORDER BY

(CASE

WHEN POSTALCODE IS NULL THEN Country

ELSE City

END);

select \* from Customers;

---------------------------------------------------------------------------------------------

-- EXISTS Operator

---------------------------------------------------------------------------------------------

/\*

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns TRUE if the subquery returns one or more records.

EXISTS Syntax

SELECT column\_name(s)

FROM table\_name

WHERE EXISTS

(SELECT column\_name FROM table\_name WHERE condition);

\*/

-- The following SQL statement returns TRUE and lists the suppliers with a product price less than 20:

SELECT SupplierName

FROM Suppliers

WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price < 20);

-- The following SQL statement returns TRUE and lists the suppliers with a product price equal to 22:

SELECT SupplierName

FROM Suppliers

WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price = 22);

CREATE TABLE IF NOT EXISTS EMPLOYEES(id int); -- > drop the employees(data will be lost) table and then it will create a new table

CREATE OR REPLACE TABLE EMPLOYEES(id int);

select \* from employees;

---------------------------------------------------------------------------------------------

-- INSERT INTO SELECT Statement

---------------------------------------------------------------------------------------------

/\*

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

The INSERT INTO SELECT statement requires that the data types in source and target tables match.

Note: The existing records in the target table are unaffected.

INSERT INTO SELECT Syntax

Copy all columns from one table to another table:

INSERT INTO table2

SELECT \* FROM table1

WHERE condition;

Copy only some columns from one table into another table:

INSERT INTO table2 (column1, column2, column3, ...)

SELECT column1, column2, column3, ...

FROM table1

WHERE condition;

\*/

-- Copy "Suppliers" into "Customers" (the columns that are not filled with data, will contain NULL):

INSERT INTO Customers (CustomerName, City, Country)

SELECT SupplierName, City, Country FROM Suppliers;

select \* from customers;

-- Copy "Suppliers" into "Customers" (fill all columns):

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)

SELECT SupplierName, ContactName, Address, City, PostalCode, Country FROM Suppliers;

-- Copy only the German suppliers into "Customers":

INSERT INTO Customers (CustomerName, City, Country)

SELECT SupplierName, City, Country FROM Suppliers

WHERE Country='Germany';

---------------------------------------------------------------------------------------------

-- Create Table AS Select statement(CTAS)

---------------------------------------------------------------------------------------------

create table customer\_copy as select \* from customers;

select \* from customer\_copy;

create table customer\_subset as select customerid,CustomerName, City, Country from customers;

select \* from customer\_subset;