

Group BY
and
Aggregate Functions

SELECT - Syntax

```
SELECT [ ALL | DISTINCT ] [TOP expr [PERCENT] [ WITH TIES ] ]
    < select_list >
    [ FROM { <table_source> } [ ,...n ] ]
    [ WHERE <search_condition> ]
    [ GROUP BY ]
    [ HAVING <search_condition> ]
    [ ORDER BY ]
```

The GROUP BY Clause

- The GROUP BY clause groups a selected set of rows into a set of summary rows
- The GROUP BY clause is executed after the rows are filtered by the WHERE condition
- The grouping is done by using the values of one or more columns or expressions
- Only ONE ROW is returned for EACH GROUP
- Aggregate functions in the **select_list** are used to provide information about each group instead of individual rows

Group By Expression

- The **GROUP BY Expression** is an expression on which grouping is performed
  - It can be a column or an expression that references a column returned by the FROM clause
- Expressions in the GROUP BY clause can contain columns of the tables in the FROM clause
  - The columns are not required to appear in the <select> list
- Each column in any nonaggregate expression in the <select> list must be included in the GROUP BY list
- Rows that do not meet the conditions in the WHERE clause are removed before any grouping operation is performed.

Aggregate Functions

- Aggregate functions perform a calculation on a set of values and return a single (SCALAR) value
- Alias must be used for the aggregate function
  - Since no column name is supplied for the result set
- Aggregate functions can be used as expressions only in the following:
  - The select list of a SELECT statement
  - A HAVING clause
  - ORDER BY
- Aggregates cannot be used in the WHERE clause
  - WHERE clause filters rows before aggregate functions are used

Aggregate Functions (cont.)

- Categories of Aggregate functions
  - **Common** (SUM(), AVG(), MIN(), MAX(), COUNT(), COUNT(*))
  - **Statistical** (STDEV(), VAR(), …)
  - **Other** (CheckSum(),CheckSum_agg(), …)
- Aggregate functions and NULL
  - All Aggregate function ignore NULL except the COUNT(*)
- DISTINCT and Aggregate Functions
  - DISTINCT is used to summarize unique values only
  - DISTINCT forces only unique column values to be processed by an aggregate function

Grouping and Aggregation

- Steps in Grouping and Aggregation:
  - 1. Compute the FROM and WHERE clauses
  - 2. Group by the attributes in the GROUP BY clause
  - 3. Compute the SELECT clause
    - The NON-Aggregate columns
    - The grouped attributes and aggregates

Understanding GROUP BY

EmpID	FName	LName	DepName	Salary
1	Taye	Mekonnen	Administration	25,000.00
2	Sofia	Teferi	Research	18,000.00
3	Genet	Abebe	Administration	20,000.00
4	Daniel	Abera	Sales	12,000.00
5	Hanna	Samuel	Sales	15,000.00
6	Kaleb	Kassa	Sales	5,400.00
7	Solomon	Michael	Sales	5,000.00
8	Beza	Seifu	Administration	10,000.00
9	Michael	Yossef	Sales	4,500.00
10	Hanna	Natnael	Research	12,000.00

EmpID	FName	LName	DepName	Salary
3	Genet	Abebe	Administration	20,000.00
1	Taye	Mekonnen		25,000.00
8	Beza	Seifu		10,000.00
2	Sofia	Teferi	Research	18,000.00
10	Hanna	Natnael		12,000.00
9	Michael	Yossef	Sales	4,500.00
4	Daniel	Abera		12,000.00
5	Hanna	Samuel		15,000.00
6	Kaleb	Kassa		5,400.00
7	Solomon	Michael		5,000.00

1. GROUP BY DepName
2. Use Aggregate functions

```
SELECT  DepName , SUM(DepName) AS Total
FROM    Employee
GROUP BY  DepName
```

DepName	Total
Administration	55,000.00
Research	30,000.00
Sales	41,900.00

GROUP BY - Examples

```
SELECT    DepName , COUNT(DepName) AS NumberOfEmp
FROM      Employee
GROUP BY  DepName
```

```
SELECT    DepName , AVE(Salary) AS AvSalary
FROM      Employee
GROUP BY  DepName
```

```
SELECT FName , AVE(Salary) X
FROM   Employee
GROUP BY DepName ?
```

EmpID	FName	LName	DepName	Salary
3	Genet	Abebe	Administration	20,000.00
1	Taye	Mekonnen		25,000.00
8	Beza	Seifu		10,000.00
2	Sofia	Teferi	Research	18,000.00
10	Hanna	Natnael		12,000.00
9	Michael	Yossef	Sales	4,500.00
4	Daniel	Abera		12,000.00
5	Hanna	Samuel		15,000.00
6	Kaleb	Kassa		5,400.00
7	Solomon	Michael		5,900.00

Aggregate Function: Count

- COUNT - Applies to duplicates, unless otherwise stated
- Example

```
SELECT    COUNT (Category)
FROM      Product
WHERE     Year > 1900
```

- Example (Counting unique values ONLY)

```
SELECT    COUNT (DISTINCT Category)
FROM      Product
WHERE     Year > 1900
```

Aggregate Function: DISTINCT

- DISTINCT can be used with other aggregate functions too.
- Example

```
SELECT  Sum(Distinct Salary)
FROM      Employee

SELECT  Sum( Salary)
FROM      Employee
```

Aggregate Function: Example

- Examples

```
SELECT    Sum(price * quantity) AS Amount
FROM      Sales
WHERE     CustomerID = '12000911'
```

```
SELECT    ItemName, Sum(price * quantity) AS TotalSales
FROM      Sales
WHERE     date > '10/1/2005'
GROUP BY  ItemName
```

The HAVING Clause

- The HAVING clause is used with the GROUP BY clause to filter groups in the result set.
- Specifies a search condition for a group or an aggregate
- When GROUP BY is not used, HAVING behaves like a WHERE clause
  - HAVING is typically used with a GROUP BY clause
- Example

```
SELECT      DepName , COUNT(DepName)
FROM        Employee
GROUP BY    DepName
HAVING      COUNT(DepName) > 3
ORDER BY    COUNT(DepName) DESC
```

Examples 1

- Using aggregates without GROUP BY

```
SELECT    COUNT(EmployeeID) AS [Number of Employees]
FROM      Employee
```

- Using aggregates with GROUP BY

```
SELECT    Department
          ,COUNT(EmployeeID) AS [Number of Employees]
FROM      Employee
GROUP BY  Department
```

- Using aggregates with GROUP BY

```
SELECT    Department
          , COUNT(EmployeeID) AS [Number of Employees]
FROM      Employee
GROUP BY  Department
```

Examples 2

- Grouping can be done by any column referenced by the FROM clause

```
SELECT      DepartmentName, Gender, COUNT(Employee ID) AS ENo
FROM        Employee
GROUP BY    DepartmentName, Gender
```

- Grouping by an expression

```
SELECT      COUNT(ModifiedDate),  YEAR(ModifiedDate)
FROM        Customer
GROUP BY    YEAR(ModifiedDate)
```

Examples 3

- Using multiple aggregates

```
SELECT      DepName
            , MIN(Salary) AS MIN , MAX(Salary) AS MAX
FROM        Employee
GROUP BY    DepName
```

- Using aggregates ion ORDER BY

```
SELECT      DepName, SUM(Salary) AS TotalSalary
FROM        Employee
GROUP BY    DepName
ORDER BY    SUM(Salary)
```

Examples 4

- Using multiple aggregates

```
SELECT      JobTitle
            , SUM(Salary) AS TotalSalary
            , MIN(Salary)  AS MinSalary
            , MAX(Salary )   AS MaxSalary
            , AVE(Salary)    AS AverageSalary
FROM        Employee
GROUP BY    JobTitle
ORDER BY    TotalSalary
```

Examples 5

- Grouping by an expression

```
SELECT      DATEPART(yyyy, OrderDate) AS 'Year'
            , SUM(Amount) AS 'Total Order Amount'

FROM        Sales
WHERE       CustomerID = 10992
GROUP BY    DATEPART(yyyy, OrderDate)
HAVING      DATEPART(yyyy, OrderDate) >= '2003'
```

Examples 6

- Grouping and JOINs

```
SELECT      Name,  SUM(Quantity * UnitPrice) AS Amount
FROM        Sales JOIN Customer
            ON Sales.CustomerID = Customer.CustomerID
GROUP BY    Name
HAVING       SUM(Quantity * UnitPrice) > 20000
ORDER BY     SUM(Quantity * UnitPrice)
```