



Module 21: GROUP BY

1. GROUP BY

✓ Definition:

GROUP BY is used to **group rows** that have the same values in specified columns and perform **aggregate functions** on each group.

✓ Syntax:

```
SELECT column, AGG_FUNC(column2)
FROM table
GROUP BY column;
```

🎯 Common Use:

- Used with aggregate functions like `SUM()`, `AVG()`, `COUNT()`, `MAX()`, `MIN()`

Reference Table Used in This Module

The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are three tabs: 'SQLQuery11.sql - U...(USER\jbcom (59))*', 'SQLQuery10.sql - U...(USER\jbcom (58))*', and 'SQLQuery9.sql -'. The active tab is 'SQLQuery11.sql'. The query window contains the following SQL code:

```
32 SELECT * FROM SALES ;
33
34
35
36
37
38
39
40
41
42
43
44
```

Below the query window, there is a 'Results' pane showing a grid of data. The grid has 9 columns: 'OrderDate', 'Region', 'Manager', 'SalesMan', 'Item', 'Units', 'Unit_price', and 'Sale_amt'. The data is as follows:

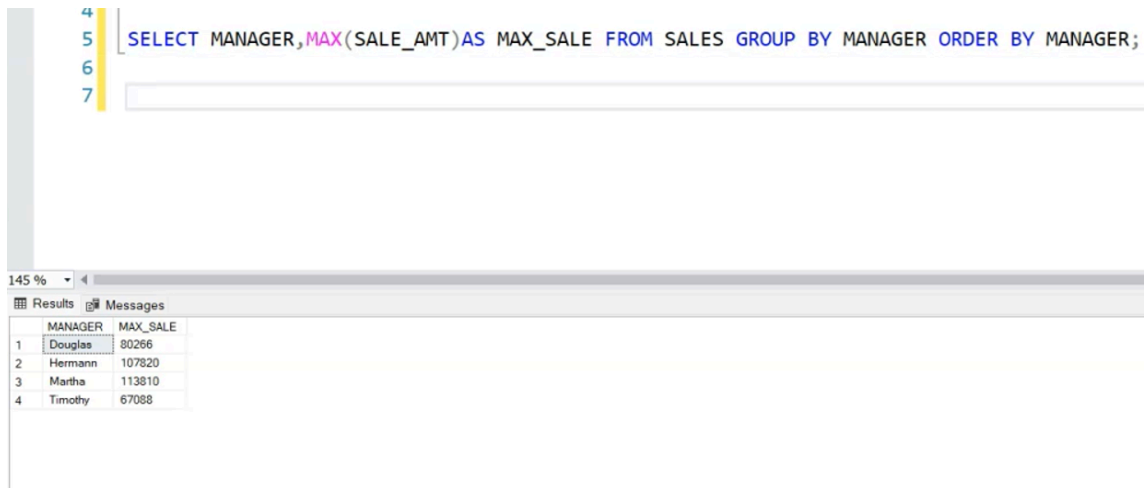
	OrderDate	Region	Manager	SalesMan	Item	Units	Unit_price	Sale_amt
1	2018-01-06	East	Martha	Alexander	Television	95	1198	113810
2	2018-01-23	Central	Hermann	Shelli	Home Theater	50	500	25000
3	2018-02-09	Central	Hermann	Luis	Television	36	1198	43128
4	2018-02-26	Central	Timothy	David	Cell Phone	27	225	6075
5	2018-03-15	West	Timothy	Stephen	Television	56	1198	67088
6	2018-04-01	East	Martha	Alexander	Home Theater	60	500	30000
7	2018-04-18	Central	Martha	Steven	Television	75	1198	89850
8	2018-05-05	Central	Hermann	Luis	Television	90	1198	107820
9	2018-05-22	West	Douglas	Michael	Television	32	1198	38336
10	2018-06-08	East	Martha	Alexander	Home Theater	60	500	30000
11	2018-06-25	Central	Hermann	Sigal	Television	90	1198	107820
12	2018-07-12	East	Martha	Diana	Home Theater	29	500	14500
13	2018-07-29	East	Douglas	Karen	Home Theater	81	500	40500
14	2018-08-15	East	Martha	Alexander	Television	35	1198	41930
15	2018-09-01	Central	Douglas	John	Desk	2	125	250
16	2018-09-18	East	Martha	Alexander	Video Games	16	58.5	936
17	2018-10-05	Central	Hermann	Sigal	Home Theater	28	500	14000

At the bottom of the screenshot, there is a status bar that says 'Query executed successfully.'

Examples:

```
SELECT Manager, MAX(Sale_amt) AS Max_Sale
FROM Sales
GROUP BY Manager
ORDER BY Manager;
```

✓ Groups records by Manager and shows each Manager's highest sale amount.



The screenshot shows a SQL query editor with the following query:

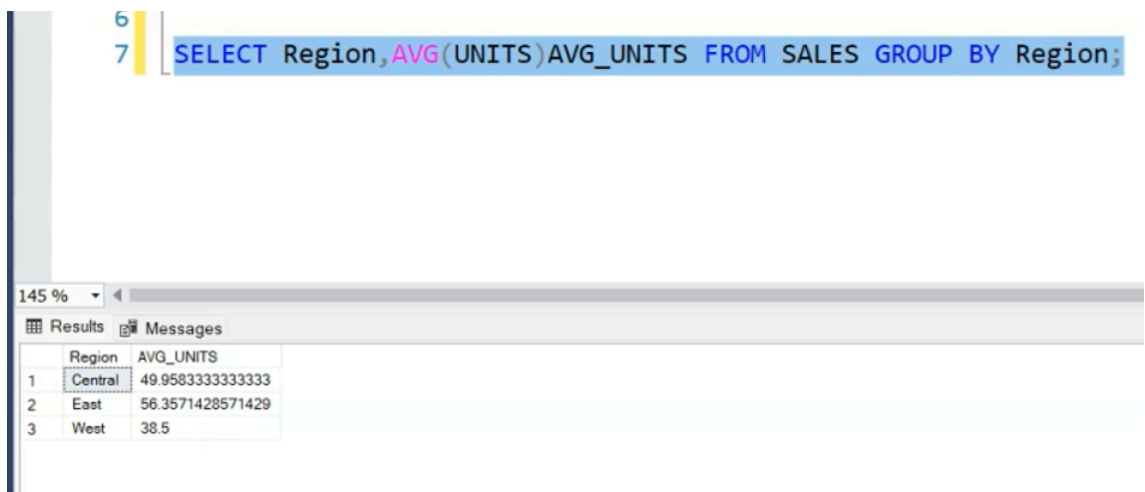
```
SELECT MANAGER, MAX(SALE_AMT) AS MAX_SALE FROM SALES GROUP BY MANAGER ORDER BY MANAGER;
```

Below the query editor, the results are displayed in a table with two columns: MANAGER and MAX_SALE. The results are ordered by MANAGER.

	MANAGER	MAX_SALE
1	Douglas	80266
2	Hermann	107820
3	Martha	113810
4	Timothy	67088

```
SELECT Region, AVG(Units) AS Avg_units
FROM Sales
GROUP BY Region;
```

✓ Groups rows by Region and shows average units sold in each.



The screenshot shows a SQL query editor with the following query:

```
SELECT Region, AVG(UNITS) AS AVG_UNITS FROM SALES GROUP BY Region;
```

Below the query editor, the results are displayed in a table with two columns: Region and AVG_UNITS. The results are ordered by Region.

	Region	AVG_UNITS
1	Central	49.95833333333333
2	East	56.3571428571429
3	West	38.5

◆ 2. HAVING

✓ Definition:

HAVING is used to **filter grouped results** based on conditions on aggregate functions.

✓ Syntax:

```
SELECT column, AGG_FUNC(column2)
FROM table
WHERE condition
GROUP BY column
HAVING aggregate_condition;
```

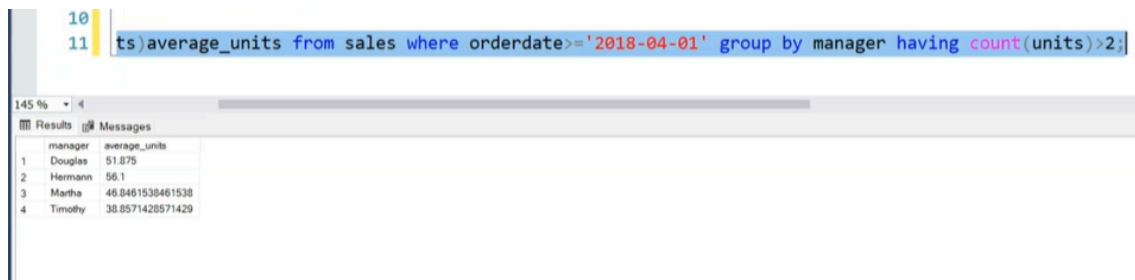
🎯 Common Use:

- **WHERE** filters **rows before** grouping
- **HAVING** filters **groups after** aggregation

📌 Example:

```
SELECT Manager, AVG(Units) AS Average_Units
FROM Sales
WHERE OrderDate >= '2018-04-01'
GROUP BY Manager
HAVING COUNT(Units) > 2;
```

✓ Shows only those Managers with **more than 2 records** after April 1, 2018.



	manager	average_units
1	Douglas	51.875
2	Hermann	56.1
3	Martha	46.8461538461538
4	Timothy	38.8571428571429

What Are Aggregate Functions?

Aggregate functions perform **calculations across multiple rows**:

Function	Use
<code>SUM()</code>	Adds values
<code>AVG()</code>	Finds average
<code>COUNT()</code>	Counts rows
<code>MAX()</code>	Highest value
<code>MIN()</code>	Lowest value

Why Not Use `WHERE` with Aggregates?

- `WHERE` works **before grouping**, and cannot handle aggregate functions like `AVG()`, `COUNT()`, etc.
- Use `HAVING` to filter based on **grouped/aggregated values**

Key Points to Remember

- ✓ Use `GROUP BY` with **columns you want to group by**
- ✓ Use `HAVING` to **filter aggregated groups**
- ✓ `WHERE` filters **rows before grouping**
- ✓ You **must include** the grouped column(s) in the `SELECT` statement
- ✓ Aggregate functions like `SUM`, `AVG`, `COUNT` work best with `GROUP BY`