### Whether to use Group By by disabling **only\_full\_group\_by**?

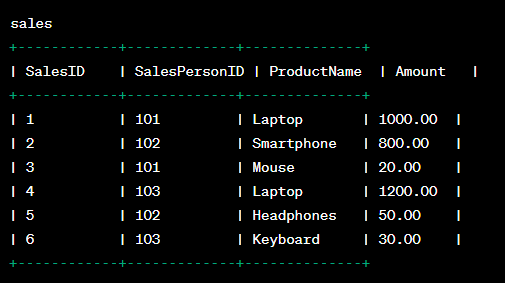
### What are the industry standards?

-> That depends on industry to industry. But it is preferable that you don't select columns that are not in the GROUP BY without aggregating.

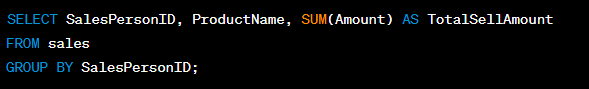
Enabling **only\_full\_group\_by** can be seen as a best practice as it encourages writing SQL queries that are less error-prone and easier to understand. By adhering to this strict mode, you explicitly state which columns you want to group the data by and which columns you want to aggregate.

**Let's see the below scenario:**

Let's consider a table named `sales` that contains information about sales transactions, including the salesperson ID, product name, and the total sale amount. Each salesperson can sell multiple products.



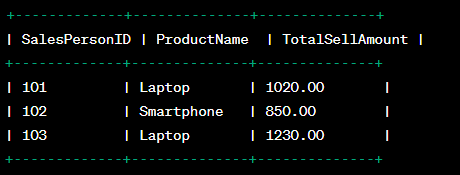
If we run a query like below, with `SalesPersonID` in the `GROUP BY` clause and include `ProductName` in the SELECT clause without using any aggregate function on it:



With `only\_full\_group\_by` disabled, this query would run without any error, but the result will be ambiguous and might not produce the expected outcome.

The output will show the `SalesPersonID`, `ProductName`, and the sum of the `Amount` for each salesperson, but the specific `ProductName` displayed for each salesperson will be non-deterministic. It means the database engine will choose any random `ProductName` associated with a particular `SalesPersonID`, and the actual product name displayed might vary with each execution of the query.

Example of a possible output:



In this example, for `SalesPersonID` 101, it displays `Laptop`, but it could have shown `Mouse` as well since both products are associated with the same salesperson.

To get the correct and unambiguous result, you should include `ProductName` in the `GROUP BY` clause or use an appropriate aggregate function like `GROUP\_CONCAT()` to concatenate the product names for each salesperson.

Corrected query:

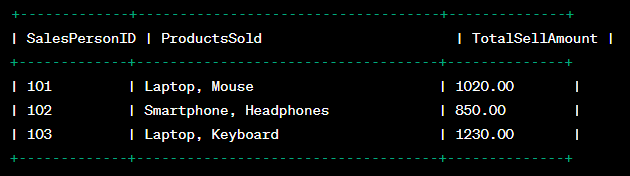
SELECT SalesPersonID, GROUP\_CONCAT(ProductName) AS ProductsSold, SUM(Amount) AS TotalSellAmount

FROM sales

GROUP BY SalesPersonID;

With this corrected query, you will get a result that concatenates all the products sold by each salesperson, ensuring that you see a comprehensive list of products associated with each `SalesPersonID`.

Example of the corrected output:



### **Example 2:**

SET sql\_mode = 'STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE TABLE elite\_agent (

id INT,

city VARCHAR(50),

gender CHAR(1),

age INT

);

INSERT INTO elite\_agent (id, city, gender, age) VALUES

(1, 'Lisbon', 'M', 21),

(2, 'Chicago', 'F', 20),

(3, 'New York', 'F', 20),

(4, 'Chicago', 'M', 27),

(5, 'Lisbon', 'F', 27),

(6, 'Lisbon', 'M', 19),

(7, 'Lisbon', 'F', 23),

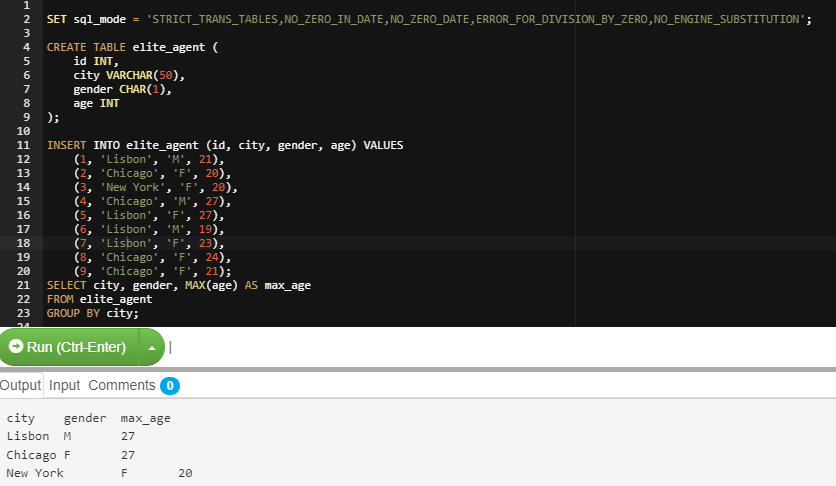
(8, 'Chicago', 'F', 24),

(9, 'Chicago', 'F', 21);

SELECT city, gender, MAX(age) AS max\_age

FROM elite\_agent

GROUP BY city;



Ambiguity in this result: Why gender is F in the case of city=Chicago? The max age in the case of City Chicago is 27 of id=4, whose gender is **M,** still getting the result as **F.**

These are the ambiguities with using Group By on disabling only\_full\_group\_by