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.

```
sales
+------------+-------------+--------------+
| SalesID    | SalesPersonID | ProductName  | Amount   |
+------------+-------------+--------------+
| 1          | 101         | Laptop       | 1000.00  |
| 2          | 102         | Smartphone   | 800.00   |
| 3          | 101         | Mouse        | 20.00    |
| 4          | 103         | Laptop       | 1200.00  |
| 5          | 102         | Headphones   | 50.00    |
| 6          | 103         | Keyboard     | 30.00    |
+------------+-------------+--------------+
```

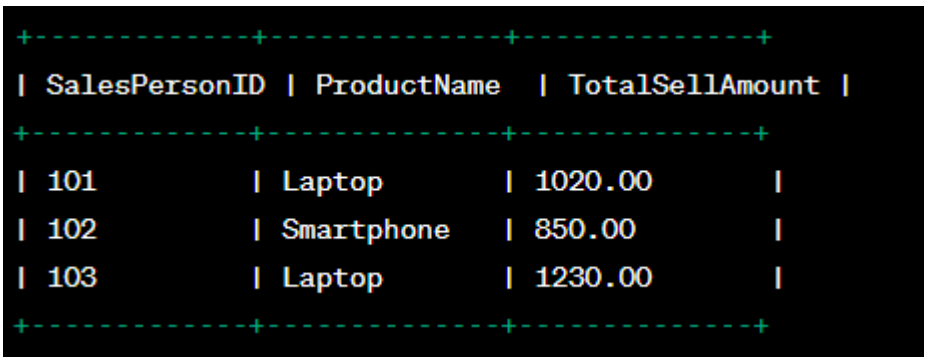
If we run a query like below, with `SalesPersonID` in the `GROUP BY` clause and include `ProductName` in the SELECT clause without using any aggregate function on it:

```
SELECT SalesPersonID, ProductName, SUM(Amount) AS TotalSellAmount
FROM sales
GROUP BY SalesPersonID;
```

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:



SalesPersonID	ProductName	TotalSellAmount
101	Laptop	1020.00
102	Smartphone	850.00
103	Laptop	1230.00

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.

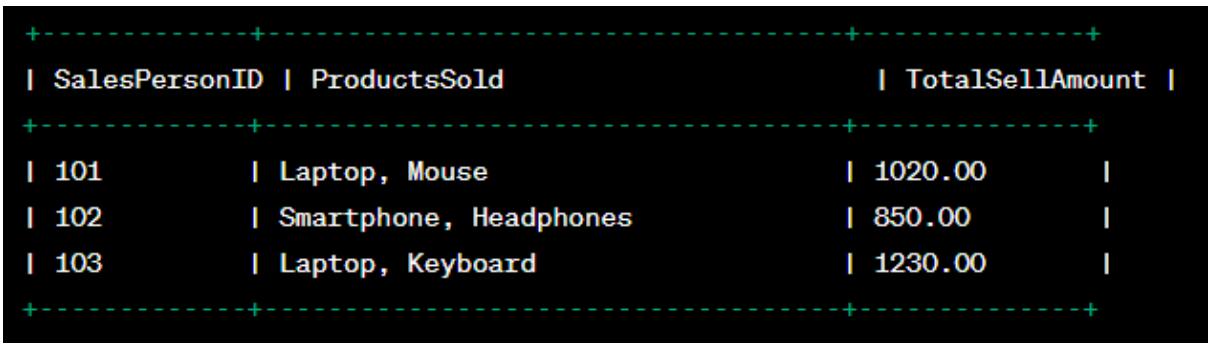
To get the correct and unambiguous result, you should include `ProductName` in the `GROUP BY` clause or use an appropriate aggregate function like `GROUP_CONCAT()` to concatenate the product names for each salesperson.

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:



SalesPersonID	ProductsSold	TotalSellAmount
101	Laptop, Mouse	1020.00
102	Smartphone, Headphones	850.00
103	Laptop, Keyboard	1230.00

Example 2:

```
SET sql_mode =
'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISIO
N_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;
```

```sql
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;
```

Run (Ctrl-Enter)

Output Input Comments 0

```
city     gender  max_age
Lisbon   M       27
Chicago  F       27
New York         F       20
```

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