**GROUP BY 句と集約関数の関係：切っても切れない仲**

GROUP BY 句と集約関数は、SQLにおいてデータを集計・分析する上で、まさに「切っても切れない仲」です。

* **GROUP BY 句**: データを特定の属性（列）の値に基づいてグループ化します。
* **集約関数**: グループ化されたデータに対して、何らかの計算（合計、平均、最大、最小、件数など）を行います。

**GROUP BY 句でグループ分けし、集約関数でそれぞれのグループの特性値を求める、という一連の流れが基本です。**GROUP BY 句 *なし* で集約関数を使うと、テーブル全体のデータに対して集計が行われますが、GROUP BY 句と組み合わせることで、より詳細な分析が可能になります。

**SQL事例集：従業員、部署、商品など日本語で**

以下に、様々なシナリオを想定したSQL事例を、GROUP BY 句と集約関数の関係に注目しながらご紹介します。

**1. 従業員テーブル (従業員ID, 氏名, 部署ID, 給与, 入社年月日)**

**1.1 部署ごとの平均給与**

SELECT

　　部署ID,

　　AVG(給与) AS 平均給与

FROM

　　従業員

GROUP BY

　　部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: 部署ID でグループ化し、各部署の 給与 の平均を AVG() 関数で計算しています。

**1.2 部署ごとの従業員数**

SELECT

　　部署ID,

　　COUNT(\*) AS 従業員数

FROM

　　従業員

GROUP BY

　　部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: 部署ID でグループ化し、各部署の従業員数を COUNT(\*) 関数で数えています。COUNT(\*) は、各グループの行数をカウントします。

**1.3 部署ごとの最高給与と最低給与**

SELECT

　　部署ID,

　　MAX(給与) AS 最高給与,

　　MIN(給与) AS 最低給与

FROM

　　従業員

GROUP BY

　　部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: 部署ID でグループ化し、各部署の最高給与を MAX()、最低給与を MIN() 関数で取得しています。

**1.4 部署ごとの合計給与と平均年齢(年齢は入社年月日からの算出)**

SELECT

　　部署ID,

　　SUM(給与) AS 合計給与,

　　AVG(

　　　　CAST(strftime('%Y.%m%d', 'now') - strftime('%Y.%m%d', 入社年月日) AS INT)

　　) AS 平均年齢 --簡易的な計算です。

FROM

　　従業員

GROUP BY

　　部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: 部署ごとにグループ化し、SUM(給与)で合計給与、入社年月日と現在の日付の差から平均年齢を算出しています。(注意: これは簡易的な年齢計算です。より正確な計算には、データベースシステム固有の日付/時刻関数を使用する必要があります。)

**2. 部署テーブル (部署ID, 部署名)**

**2.1 部署ごとの従業員数と部署名**

SELECT

　　部署.部署名,

　　COUNT(従業員.従業員ID) AS 従業員数

FROM

　　従業員

INNER JOIN

　　部署 ON 従業員.部署ID = 部署.部署ID

GROUP BY

　　部署.部署ID, 部署.部署名;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: 部署テーブルと従業員テーブルを結合し、部署ごとに従業員数をカウント。GROUP BY には、SELECT で選択する非集計列（この場合は部署.部署名）をすべて含める必要があります。

**3. 商品テーブル (商品ID, 商品名, カテゴリID, 価格, 販売日)**

**3.1 カテゴリごとの売上合計**

SELECT

　　カテゴリID,

　　SUM(価格) AS 売上合計

FROM

　　商品

GROUP BY

　　カテゴリID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: カテゴリID でグループ化し、各カテゴリの売上合計を SUM(価格) で計算します。

**3.2 カテゴリごとの商品数**

SELECT

　　カテゴリID,

　　COUNT(\*) AS 商品数

FROM

　　商品

GROUP BY

　　カテゴリID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: カテゴリID でグループ化し、各カテゴリの商品数を COUNT(\*) で数えます。

**3.3 月別の売上合計**

SELECT

　　strftime('%Y-%m', 販売日) AS 販売年月,

　　SUM(価格) AS 売上合計

FROM

　　商品

GROUP BY

　　販売年月;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: 販売日 を strftime('%Y-%m', 販売日) で年月の形式に変換し、その年月でグループ化。各月の売上合計を計算します。

**3.4 カテゴリ別、月別の売上合計**

SELECT

　　カテゴリID,

　　strftime('%Y-%m', 販売日) AS 販売年月,

　　SUM(価格) AS 売上合計

FROM

　　商品

GROUP BY

　　カテゴリID, 販売年月;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: カテゴリIDと販売年月の二つの軸でグループ化。それぞれの組み合わせに対して売上合計を算出

**4. HAVING 句との組み合わせ**

HAVING 句は、GROUP BY でグループ化した結果に対して、さらに条件を絞り込むために使用します。集約関数を使った条件を指定できるのが特徴です。

**4.1 平均給与が500万円以上の部署**

SELECT

　　部署ID,

　　AVG(給与) AS 平均給与

FROM

　　従業員

GROUP BY

　　部署ID

HAVING

　　AVG(給与) >= 5000000;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: 部署ID でグループ化し、AVG(給与) で平均給与を計算。HAVING 句で、平均給与が500万円以上の部署のみに絞り込んでいます。

**4.2 従業員数が10人を超える部署**

SELECT

　　 部署ID, COUNT(\*) AS 従業員数

FROM 従業員

GROUP BY 部署ID

HAVING COUNT(\*) > 10;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* **解説**: 部署IDでグループ化し、COUNT(\*)で各部署の従業員数を算出。HAVING句を使って、従業員数が10人を超える部署に限定。

**まとめ：GROUP BY と集約関数は最強コンビ**

GROUP BY 句と集約関数は、SQLでデータ分析を行う上で、まさに最強のコンビです。様々な組み合わせと、HAVING 句などの追加要素を使いこなすことで、データの奥深くに隠された洞察を引き出すことができます。

SQLの神様として、あなたのデータ分析の旅が実り多いものになるよう、心から応援しています！

SELECT内に集約関数を使わない属性があった場合、その属性は必ずGROUPBYで処理する必要はありますか。

edit

more\_vert

はい、その通りです！ SQLの神様、再びお答えします。🧙‍♂️✨

SELECT句内に集約関数を使わない属性（列）がある場合、原則として、その属性は必ずGROUP BY句で指定する必要があります。 これを守らないと、ほとんどのデータベースシステムでエラーが発生します。

**理由：集計の整合性を保つため**

GROUP BY句は、指定された属性の値が同じ行を1つのグループにまとめます。集約関数（SUM、AVG、COUNTなど）は、このグループごとに値を計算します。

もしSELECT句に集約関数を使わない属性があり、それをGROUP BY句で指定しないと、データベースシステムはどの行の値を表示すればよいか判断できません。

**例で確認**

次の「従業員」テーブルを例に考えてみましょう。

|  |  |  |  |
| --- | --- | --- | --- |
| **従業員ID** | **氏名** | **部署ID** | **給与** |
| 1 | 山田 | 10 | 5000000 |
| 2 | 田中 | 10 | 6000000 |
| 3 | 佐藤 | 20 | 4500000 |
| 4 | 鈴木 | 20 | 5500000 |

**1. 間違った例 (エラーになる)**

-- エラーになる例

SELECT

部署ID,

氏名, -- 集約関数を使っていない

AVG(給与) AS 平均給与

FROM

従業員

GROUP BY

部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

このクエリはエラーになります。なぜなら、GROUP BY句で部署IDを指定してグループ化しているのに、SELECT句には氏名（集約関数を使っていない）が含まれているからです。部署IDが同じグループ内に複数の氏名が存在する場合、データベースはどの氏名を表示すればいいのか分かりません。

**2. 正しい例**

SELECT

部署ID,

AVG(給与) AS 平均給与

FROM

従業員

GROUP BY

部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

これは正しいクエリです。SELECT 句にあるのは部署ID（グループ化のキー）とAVG(給与)（集約関数の結果）だけなので、問題ありません。

**3. 氏名も表示したい場合 (正しい例)**

もし氏名も表示したい場合は、氏名もGROUP BY句に含める必要があります。ただし、この場合は部署IDと氏名の組み合わせでグループ化されるため、結果が変わることに注意してください。

SELECT

部署ID,

氏名,

AVG(給与) AS 平均給与

FROM

従業員

GROUP BY

部署ID,

氏名;

[with caution](https://support.google.com/legal/answer/13505487).SQL

このクエリはエラーにはなりませんが、同じ部署IDでも氏名が異なれば別のグループとして集計されます。  
もし、部署ごとの平均給与と、その部署の*誰か一人*の氏名を表示したい、という場合は、少し工夫が必要になります。これはデータベースシステムによって異なるので、具体的な方法は後述します。

**例外：一部のデータベースシステム**

ごく一部のデータベースシステム（古いバージョンのMySQLなど）では、GROUP BY句で指定していない属性をSELECT句に含めてもエラーにならない場合があります。しかし、これはSQLの標準的な動作ではなく、結果が予測不可能になる可能性があるため、避けるべきです。　（MySQLでも、ONLY\_FULL\_GROUP\_BYというSQLモードが有効になっていると、標準的な動作になりエラーになります。）

**補足：どうしてもGROUP BYに入れたくない属性と集約関数を同時に使いたい場合**

先ほどの例で、「部署ごとの平均給与」と「その部署の*誰か一人*の氏名」を表示したい場合、以下のような方法が考えられます。

1. **サブクエリを使う:**
2. SELECT
3. e.部署ID,
4. (SELECT 氏名 FROM 従業員 WHERE 部署ID = e.部署ID LIMIT 1) AS 代表氏名, -- 誰か一人を選ぶ
5. AVG(e.給与) AS 平均給与
6. FROM
7. 従業員 e
8. GROUP BY

e.部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* + 外側のクエリで部署ごとにグループ化し、平均給与を計算。
  + 内側のサブクエリで、各部署から氏名を1つだけ取得（LIMIT 1）。

1. **ウィンドウ関数を使う (より高度):**  
   ウィンドウ関数が利用できるデータベース(PostgreSQL, MySQL 8.0+, Oracle, SQL Serverなど)では、より洗練された方法があります。
2. SELECT DISTINCT
3. 部署ID,
4. FIRST\_VALUE(氏名) OVER (PARTITION BY 部署ID) AS 代表氏名,
5. AVG(給与) OVER (PARTITION BY 部署ID) AS 平均給与
6. FROM

従業員;

[with caution](https://support.google.com/legal/answer/13505487).SQL

* + FIRST\_VALUE(氏名) OVER (PARTITION BY 部署ID): 部署IDでパーティション（グループ化）し、各パーティション内の最初の氏名を取得。
  + AVG(給与) OVER (PARTITION BY 部署ID): 部署IDでパーティションし、各パーティション内の平均給与を計算。
  + DISTINCT: FIRST\_VALUEとAVGは各行に対して計算されるため、DISTINCTで重複を取り除き、部署ごとの結果を1行で表示。  
    ウィンドウ関数は、GROUP BYを使わずに、グループ化に似た操作を柔軟に行える強力な機能です。

**結論**

* 原則: SELECT句に集約関数を使わない属性がある場合、その属性は必ずGROUP BY句に含める。
* 例外: 一部の古いDBMSではエラーにならない場合があるが、避けるべき。
* 応用: サブクエリやウィンドウ関数を使うことで、より柔軟な集計処理が可能。

SQLの理解が深まったでしょうか？ どんな些細な疑問でも、遠慮なく聞いてくださいね！ ✨

同様な注意点があったら是非教えてください。神様

edit

more\_vert

もちろんです、SQLの神様、再び降臨です！🧙‍♂️✨ GROUP BYと集約関数に関連する、あるいはSQL全般における、同様の注意点をいくつかご紹介します。落とし穴にはまらないように、しっかりチェックしていきましょう！

**1. GROUP BY と ORDER BY は別物**

* **GROUP BY**: データをグループ化する。
* **ORDER BY**: 結果セットを並べ替える。

GROUP BY しただけでは、結果が特定の順序で並ぶとは限りません。結果を並べ替えたい場合は、必ず ORDER BY 句を明示的に使用してください。

-- 部署ごとの平均給与を計算し、平均給与の降順で並べ替え

SELECT

部署ID,

AVG(給与) AS 平均給与

FROM

従業員

GROUP BY

部署ID

ORDER BY

平均給与 DESC;

[with caution](https://support.google.com/legal/answer/13505487).SQL

**2. WHERE と HAVING の使い分け**

* **WHERE**: GROUP BY *前* の個々の行に対するフィルタリング。
* **HAVING**: GROUP BY *後* のグループに対するフィルタリング（集約関数の結果に対する条件など）。

-- 給与が400万円以上の従業員に限定し、部署ごとの平均給与を計算。

-- さらに、平均給与が550万円以上の部署のみ表示。

SELECT

部署ID,

AVG(給与) AS 平均給与

FROM

従業員

WHERE

給与 >= 4000000 -- 個々の行に対する条件

GROUP BY

部署ID

HAVING

AVG(給与) >= 5500000; -- グループに対する条件

[with caution](https://support.google.com/legal/answer/13505487).SQL

**3. 集約関数と NULL 値**

集約関数は、通常、NULL 値を無視します（COUNT(\*) は例外で、NULL を含むすべての行をカウントします）。この挙動を理解していないと、意図しない結果になることがあります。

-- 例: 商品テーブル (商品ID, 商品名, 価格)

-- 価格がNULLの商品がある場合

-- NULLを除く価格の平均を計算

SELECT AVG(価格) FROM 商品;

-- NULLを含むすべての行数をカウント

SELECT COUNT(\*) FROM 商品;

-- 価格がNULLでない行数

SELECT COUNT(価格) FROM 商品;

[with caution](https://support.google.com/legal/answer/13505487).SQL

**4. 集約関数の中の計算**

集約関数の中で計算を行うことも可能です。

-- 部署ごとの給与の合計を、1000で割って表示

SELECT

部署ID,

SUM(給与) / 1000 AS 給与合計\_千単位

FROM

従業員

GROUP BY

部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

**5. DISTINCT と GROUP BY**

DISTINCT は、SELECT 句で指定した列の重複を取り除きます。GROUP BY と似た結果を得られる場合もありますが、意味合いが異なります。

* **DISTINCT**: 重複行を排除した結果を返す。
* **GROUP BY**: 指定した列でグループ化し、各グループに対して集約関数を適用できる。

-- 部署IDの重複なしリスト

SELECT DISTINCT 部署ID FROM 従業員;

-- 部署IDごとの従業員数

SELECT 部署ID, COUNT(\*) FROM 従業員 GROUP BY 部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

DISTINCTは集約を行わないため、単純にユニークな値のリストが欲しい場合に、GROUP BYよりも高速に動作することがあります。

**6. データ型**

集約関数を使用する際は、対象となる列のデータ型に注意してください。例えば、文字列型の列に対して SUM や AVG を使用するとエラーになります。

**7. パフォーマンス**

大規模なデータに対して GROUP BY と集約関数を使用する場合、パフォーマンスに注意が必要です。適切なインデックスが設定されていないと、クエリが非常に遅くなる可能性があります。

**8. 複雑な条件と CASE 式**  
集計時に複雑な条件分岐が必要な場合、CASE式を集約関数と組み合わせて使用すると便利です。

-- 部署ごとに、給与が500万以上の従業員と、それ未満の従業員の数をそれぞれカウント

SELECT

部署ID,

COUNT(CASE WHEN 給与 >= 5000000 THEN 1 END) AS 高給与従業員数,

COUNT(CASE WHEN 給与 < 5000000 THEN 1 END) AS 低給与従業員数

FROM

従業員

GROUP BY

部署ID;

[with caution](https://support.google.com/legal/answer/13505487).SQL

**9. サブクエリ、CTE(共通テーブル式)、ウィンドウ関数の活用**

より複雑な集計処理を行う場合、サブクエリ、CTE（共通テーブル式）、ウィンドウ関数などを活用すると、クエリをより分かりやすく、効率的に記述できます。

これらの注意点を意識することで、SQLの集計処理をより深く理解し、効果的に活用できるようになるでしょう。SQLの世界は奥深いですが、一歩ずつ着実にスキルアップしていきましょう！いつでも神様に質問してくださいね！✨

thumb\_upthumb\_down16.1s