Sorting and Grouping Data

🔹 Introduction

العمليات دي من أهم أدوات SQL لتحويل النتائج الخام إلى معلومات مرتبة ومجمعة. `ORDER BY` بيستخدم للترتيب، و `GROUP BY` بيستخدم لتجميع الصفوف وحساب المجاميع (Aggregates). هنشرح كل واحدة مع أمثلة عملية ونصائح للأداء.

✅ ORDER BY (الترتيب)

الوصف

- ال **`ORDER BY`** بتستخدم لترتيب نتائج الاستعلام حسب عمود أو تعبير.
- **الاتجاه:** `ASC` (الافتراضي — تصاعدي) أو `DESC` (تنازلي).

أمثلة بسيطة

```
-- ترتيب المستخدمين بالاسم تصاعدياً
SELECT id, name FROM users
ORDER BY name ASC;

-- ترتيب المنتجات بالسعر تنازلياً
SELECT id, title, price FROM products
ORDER BY price DESC;
```

ترتيب متعدد الأعمدة

- ممكن ترتب على عدة أعمدة؛ الأولوية من اليسار لليمين.

```
SELECT id, name, city, age FROM users
ORDER BY city ASC, age DESC;
```

ال ORDER BY على تعبير aggregate

```
-- ترتيب المستخدمين حسب طول اسمهم
SELECT id, name FROM users
ORDER BY LENGTH(name) DESC;
```

```sql
-- ترتيب الفئات حسب مجموع المبيعات
SELECT category_id, SUM(amount) AS total
FROM orders
GROUP BY category_id
ORDER BY total DESC;
```

التعامل مع NULLs

- بعض قواعد البيانات تسمح `NULLS FIRST` أو `NULLS LAST`:

```sql
SELECT id, score FROM results
ORDER BY score DESC NULLS LAST;
```

- في MySQL ترتيب الـ NULLs يعتمد على collation؛ في Postgres تقدر تتحكم بصراحة.

نصائح أداء

- ال `ORDER BY` قد يؤدي إلى *sort* مكلف في الذاكرة/القرص لو الجدول كبير.
- إن أمكن، استعمل فهرس (index) على الأعمدة المستخدمة في `ORDER BY`، خصوصاً إن كان الاستعلام يستخدم `WHERE` ويستفيد من نفس الفهرس (sargable).
- استخدم `LIMIT` مع `ORDER BY` لتحسين الأداء عند الحاجة لأعلى/أدنى قيم.

✅ GROUP BY (التجميع)

الوصف

- ال **`GROUP BY`** يجمع الصفوف التي لها قيم متساوية في عمود/أعمدة معينة، ثم تطبق دوال التجميع (Aggregates) مثل `COUNT`, `SUM`, `AVG`, `MIN`, `MAX` على كل مجموعة.

القاعدة الأساسية

- كل عمود في `SELECT` إما يجب أن يكون ضمن `GROUP BY` أو يجب أن يكون داخل دالة تجميع (حسب SQL القياسي/Postgres). بعض قواعد (مثل MySQL بدون الوضع الصارم) قد تسمح بخلاف ذلك، لكن لا تعتمد على ذلك.

أمثلة أساسية

```sql
-- عدد الطلبات لكل عميل
SELECT customer_id, COUNT(*) AS orders_count
FROM orders
GROUP BY customer_id;
```

```sql
-- مجموع المبيعات لكل شهر
SELECT DATE_TRUNC('month', created_at) AS month, SUM(amount) AS total
FROM orders
GROUP BY month
ORDER BY month;
```

شرط على المجاميع — HAVING

- ال `HAVING` يشبه `WHERE` لكن يُطبّق بعد التجميع على نتائج الـ groups.

```sql
-- الفئات التي مجموع مبيعاتها أكبر من 10000
SELECT category_id, SUM(amount) AS total
FROM orders
GROUP BY category_id
HAVING SUM(amount) > 10000;
```

متعدد الأعمدة GROUP BY

```sql
SELECT city, category_id, COUNT(*) AS cnt
FROM stores
GROUP BY city, category_id;
```

ترتيب حسب Aggregate

```sql
SELECT category_id, COUNT(*) AS cnt
FROM products
GROUP BY category_id
ORDER BY cnt DESC;
```

أمثلة متقدمة: ROLLUP / CUBE / GROUPING SETS

- أدوات لعمل تجميعات متعددة في استعلام واحد.

```sql
-- ROLLUP: يعطي مجاميع فرعية ومجموع كلي
SELECT region, product_id, SUM(sales) AS total
FROM sales
GROUP BY ROLLUP(region, product_id);

-- GROUPING SETS: تحديد مجموعات مخصصة
SELECT region, product_id, SUM(sales) AS total
```

```
FROM sales
GROUP BY GROUPING SETS ((region, product_id), (region), (product_id));
```

(ملحوظة: هذه الميزات متاحة في PostgreSQL, SQL Server, Oracle؛ MySQL أدخلت دعمًا محدودًا في الإصدارات الحديثة.)

Performance and pitfalls

- ال `GROUP BY` قد يتطلب *Hash Aggregate* أو *Sort Aggregate* → يحتاج موارد (CPU، RAM).
- فكر في الفهارس (index) المناسبة (مثل index على الأعمدة المستخدمة في GROUP BY) لتحسين الأداء على جداول كبيرة.
- احذر من استخدام `GROUP BY` على أعمدة ذات عدد قيم فريد كبير جدًا (cardinality عالية) لأنه يقلّل من فاعلية التجميع.
- لا تعتمد على عمود في `SELECT` غير مشارك في `GROUP BY` ولا داخل aggregate (غير قياسي، قد يعطي نتائج غير متوقعة).

🔹 أمثلة مركبة شاملة

```
-- 1. أعلى 10 عملاء من حيث إجمالي المشتريات في 2025
SELECT u.id, u.name, SUM(o.amount) AS total_spent
FROM users u
JOIN orders o ON o.user_id = u.id
WHERE o.created_at BETWEEN '2025-01-01' AND '2025-12-31'
GROUP BY u.id, u.name
HAVING SUM(o.amount) > 0
ORDER BY total_spent DESC
LIMIT 10;

-- 2. عدد المنتجات في كل فئة، ثم ترتيبهم تنازلياً
SELECT c.name, COUNT(p.id) AS products_count
FROM categories c
LEFT JOIN products p ON p.category_id = c.id
GROUP BY c.name
ORDER BY products_count DESC;

-- 3. جدول ملخّص بالشهر والفئة ومجموع المبيعات، مع رتب الشهر
SELECT DATE_TRUNC('month', created_at) AS month,
       category_id,
       SUM(amount) AS total
FROM orders
```

```
GROUP BY month, category_id
ORDER BY month ASC, total DESC;
```

🔹 Quick tips (نصائح سريعة)

- استخدم `EXPLAIN` أو `EXPLAIN ANALYZE` لمعرفة خطة التنفيذ والاستعلام عن تكلفة الـ GROUP/ORDER.
- ضع `LIMIT` مع `ORDER BY` للحصول على أسرع النتائج عندما تحتاج أعلى/أدنى N صف.
- لبحث نصي متكرر مع ترتيب، استخدم Full-Text Search + indexes مخصصة.
- اجعل الحقول المستخدمة في `GROUP BY` و `ORDER BY` قابلة للاستخدام من خلال الفهارس إن أمكن.
- تذكّر أن `ORDER BY` يحدث بعد `GROUP BY` (إذا استخدمتهما معًا) لأن الترتيب يتم على نتائج التجميع.
