

Database Lab: Data Aggregation and Analysis

SQL Lab Exercise

Objective

Master the use of aggregate functions (`COUNT()`, `SUM()`, `AVG()`, `MAX()`, `MIN()`), the `GROUP BY` clause, and the `HAVING` clause to perform complex data analysis and filtering.

Scenario

A retail company uses a database to track product information and monthly sales figures. You need to write queries to summarize sales performance and identify key trends.

Database Schema (Conceptual)

Table 1: Conceptual Schema for the Retail Database

Table Name	Key Column	Description
Products Sales	product_id (PK)	Stores product name and category.
	sale_id (PK)	Records individual sales transactions.
	product_id (FK)	Links to the Products table.
	quantity_sold	The number of units sold in that transaction.
	unit_price	The price per unit at the time of sale.

Sample Query

Calculate the total revenue ($\text{quantity_sold} \times \text{unit_price}$) and the number of distinct sales transactions for each product category, but only for categories that have generated more than \$5,000 in total revenue.

```
1 SELECT
2     P.category,
3     COUNT(S.sale_id) AS Total_Transactions,
4     SUM(S.quantity_sold * S.unit_price) AS Total_Revenue
5 FROM
6     Products AS P
7 JOIN
8     Sales AS S ON P.product_id = S.product_id
9 GROUP BY
10    P.category
11 HAVING
12    SUM(S.quantity_sold * S.unit_price) > 5000
13 ORDER BY
```

```
14 Total_Revenue DESC;
```

Listing 1: SQL Query for Aggregation and Filtering

Reflection

The major learning point in this lab was the distinction between `WHERE` and `HAVING`. I learned that `WHERE` filters individual rows *before* aggregation (e.g., filtering sales before calculating the total), while `HAVING` filters the results of the grouped data *after* aggregation (e.g., filtering out product categories whose total revenue is too low). Understanding that `HAVING` is essentially the `WHERE` clause for groups is key to summarizing large datasets effectively.