1. GROUP BY:

The `GROUP BY` clause is used to group rows that have the same values in specified columns into summary rows, like finding the total sales per category or the average salary per department.

Example 1: Grouping by a Single Column

```sql
-- Create a table
CREATE TABLE sales (
    product_id INT,
    category VARCHAR(50),
    amount DECIMAL(10, 2)
);

-- Insert sample data
INSERT INTO sales VALUES (1, 'Electronics', 1000);
INSERT INTO sales VALUES (2, 'Clothing', 500);
INSERT INTO sales VALUES (3, 'Electronics', 800);
INSERT INTO sales VALUES (4, 'Clothing', 1200);
INSERT INTO sales VALUES (5, 'Electronics', 1500);

-- Query to get the total amount per category
SELECT category, SUM(amount) AS total_amount
FROM sales
GROUP BY category;
```

This query groups the sales table by the 'category' column, calculating the total amount for each category.

Example 2: Grouping by Multiple Columns

```sql
-- Query to get the total amount per category and product_id
SELECT category, product_id, SUM(amount) AS total_amount
FROM sales
GROUP BY category, product_id;
```

This query groups the sales table by both 'category' and 'product_id,' providing a more detailed breakdown of the total amount.

2. HAVING:

The `HAVING` clause is used in combination with the `GROUP BY` clause to filter the results of a grouped query based on specified conditions.

Example 1: Filtering by Aggregate Function Result

```sql
-- Query to get categories with a total amount greater than 1000
SELECT category, SUM(amount) AS total_amount
FROM sales
GROUP BY category
HAVING SUM(amount) > 1000;
```

This query retrieves categories with a total amount greater than 1000 by using the `HAVING` clause.

Example 2: Filtering by Count

```sql
-- Query to get categories with more than 1 product sold
SELECT category, COUNT(product_id) AS product_count
FROM sales
GROUP BY category
HAVING COUNT(product_id) > 1;
```

This query filters categories with more than one product sold, using the `HAVING` clause in combination with the `COUNT` aggregate function.

Certainly! Let's continue with more examples for both the `GROUP BY` and `HAVING` clauses in SQL:

3. GROUP BY with Aggregate Functions:

```sql
-- Query to find the average amount and total sales count per category
SELECT category, AVG(amount) AS avg_amount, COUNT(*) AS total_sales
FROM sales
GROUP BY category;
```

This example demonstrates the use of aggregate functions (`AVG` and `COUNT`) in conjunction with the `GROUP BY` clause. It calculates the average amount and total sales count for each category.

4. GROUP BY with JOIN:

```sql
-- Create a second table for demonstration
CREATE TABLE products (
    product_id INT,
    product_name VARCHAR(50)
);

-- Insert sample data into the products table
INSERT INTO products VALUES (1, 'Laptop');
INSERT INTO products VALUES (2, 'T-shirt');
INSERT INTO products VALUES (3, 'Smartphone');

-- Query to find the total sales per product name
SELECT p.product_name, SUM(s.amount) AS total_sales
FROM sales s
JOIN products p ON s.product_id = p.product_id
GROUP BY p.product_name;
```

This example involves a join operation along with the `GROUP BY` clause to find the total sales per product name by combining data from the 'sales' and 'products' tables.

5. HAVING with Multiple Conditions:

```sql
-- Query to find categories with total sales between 800 and 1500
SELECT category, SUM(amount) AS total_amount
FROM sales
GROUP BY category
HAVING SUM(amount) BETWEEN 800 AND 1500;
```

This example uses the `HAVING` clause with a range condition (`BETWEEN`) to filter categories with total sales between 800 and 1500.

6. HAVING with Logical Operators:

```sql
-- Query to find categories with an average amount greater than 500 and total sales count
greater than 1
SELECT category, AVG(amount) AS avg_amount, COUNT(*) AS total_sales
FROM sales
GROUP BY category
HAVING AVG(amount) > 500 AND COUNT(*) > 1;
```

Here, the `HAVING` clause is used with logical operators (`AND`) to filter categories with an average amount greater than 500 and a total sales count greater than 1.

These additional examples showcase more advanced scenarios where the `GROUP BY` and `HAVING` clauses can be effectively utilized in SQL queries for data analysis and reporting.