**Case Study 2:** Aggregate Functions and Group By

Scenario: You are a data analyst at a retail company. The company wants to analyze the sales data to determine the total revenue generated by each product category and find the average order value for each category.

1. Write a SQL query to calculate the total revenue generated by each product category.

2. Write a SQL query to calculate the average order value for each product category.

**Sol.**

1. Calculate the total revenue generated by each product category:

```sql

SELECT category, SUM(quantity \* price) AS total\_revenue

FROM Sales

GROUP BY category;

```

In this query, the `Sales` table is assumed to have columns such as `category`, `quantity`, and `price`. The query uses the `SUM` function to calculate the total revenue for each product category by multiplying the `quantity` and `price` columns. The `GROUP BY` clause groups the results by the `category` column.

2. Calculate the average order value for each product category:

```sql

SELECT category, AVG(order\_value) AS average\_order\_value

FROM (

SELECT order\_id, category, SUM(quantity \* price) AS order\_value

FROM Sales

GROUP BY order\_id, category

) AS subquery

GROUP BY category;

```

This query first calculates the order value for each order by multiplying the `quantity` and `price` columns, grouped by `order\_id` and `category`. Then, it uses the subquery `subquery` to aggregate the order values and calculate the average order value for each product category. The results are grouped by the `category` column.

Note: In both queries, you may need to adjust the table name (`Sales`) and column names (`category`, `quantity`, `price`, `order\_id`) to match the actual schema of your database.