The project presents a set of SQL queries and analytical views that enable **comprehensive analysis of sales data** in a retail system. The scripts are based on tables containing information about orders, products, sales staff, and stores. The goal of the project is to create **clear and reusable views** that facilitate reporting and monitoring of sales performance.

Scope of the Project

1. **Discount Analysis** – calculating the average discount applied to orders.
2. **Best Selling Products Ranking** – a view presenting products with the highest sales, including total quantity sold and total sales value.
3. **Orders with Total Value** – a view combining order and order item data, allowing quick analysis of each order's total value.
4. **Sales Staff Performance Analysis** – a ranking of employees based on sales value, enabling assessment of individual performance.
5. **Store Performance Ranking –** comparison of all sales points in terms of generated revenue.

Goals and Use Cases

The project is designed to serve as a foundation for:

- business reports (e.g., BI dashboards),
- analysis of sales performance,
- identification of top-selling products and stores,
- supporting managerial decisions related to sales strategy.

Query Workflow and Descriptions

1. **View orders_with_totals (Orders with Total Value)**
   This view combines order data with the calculated total value for each order. For every order, all items from the order_items table are summed and joined with orders, taking into account discounts (actual sale price). The result is a table containing all orders with their total value.
2. **View best_selling_products (Top Selling Products)**
   Based on order item data, this view lists the best-selling products.
   For each product, it calculates:
   - total quantity sold,
   - total sales value after discounts. This allows easy comparison of products and identification of those generating the highest revenue.
3. **View top_sales_staff (Top Sales Staff)**
   This view uses orders_with_totals to sum the sales value assigned to each employee. Each staff member’s total handled order value is displayed, creating a ranking of sales staff by performance.

4. **View top_selling_stores (Top Selling Stores)**
   Similarly, this view is based on orders_with_totals. It sums all orders assigned to each store, creating a ranking of stores with the highest revenue.

5. **Query average_discount**
   This query does not create a view but analyzes order_items to calculate the average discount applied to orders. First, the discount value for each order is calculated, then the average across all orders is derived.

6. **Query finalized_orders_by_year_2016**
   This query uses orders_with_totals to count the number of orders completed in 2016.It filters data by the year in the required_date column and returns the total number of orders for that period. This provides a simple way to assess the scale of sales for a specific year.

Summary

The complete set of queries creates a **cohesive reporting structure**:

- orders_with_totals serves as the **foundation** for all analyses,
- best_selling_products, top_sales_staff, and top_selling_stores are **ranking views**,
- average_discount and finalized_orders_by_year_2016 are examples of **analytical queries** based on the data.

Best Practices Applied

- Use of **clear aliases** and consistent code structure for readability.
- Application of **aggregate functions** and rounding to improve report quality.
- **Modular design** – each view has a distinct purpose but can be combined for further analysis.
- **Prepared for scalability** – the code can be easily expanded or integrated with reporting systems.

Final Outcome

The project allows obtaining **key sales metrics** in a simple and clear way, without the need to write complex queries for every analysis. Thanks to the defined views, the data can be quickly utilized in reports and analytical dashboards.