SQL Orders Database Project

Executive Summary

This project presents an exploration of a relational **Sales Orders database** using SQL within a notebook environment. By querying and analyzing realistic sales, customer, and order records, the work demonstrates foundational data analyst skills: database navigation, table relationships, filtering, joining, and aggregation. The analysis extracts actionable insights for business context (e.g., operations, sales, resource needs) and illustrates technical fluency in SQL.

1. Business Context

Organizations track orders, customers, and operations in relational databases. Understanding these databases—and writing powerful SQL queries—is fundamental for data analysts supporting business decision-making, sales reporting, and customer analytics.

Objective:

- Practice SQL skills on real order and customer data.
- Solve practical business questions (e.g., sales trends, fulfillment status, customer habits).

2. Data Overview

A simplified sales database consisting primarily of:

Table	Description
`sales.orders`	Each row is an order (order ID, customer, status, staff, store, and date fields)
Other Tables (not shown but typical)	`customers`, `products`, `order_items`, `stores`, `staff` (assumed for richer joins)

Key Features in `sales.orders`:

- `order_id`, `customer_id`: Primary/foreign keys
- `order_status`: Numeric code (e.g., shipped, fulfilled, canceled, pending)
- `order_date`, `required_date`, `shipped_date`: Operational timing fields
- `store_id`, `staff_id`: For operational segmentation

3. Methodology

- **Exploratory Querying:**
  Used `SELECT * FROM sales.orders` to view and understand all columns and schema.
- **Focused Analysis (suggested steps):**
  - Filtered orders by `order_status` (e.g., unshipped, late)
  - Aggregated by `store_id`, `staff_id`, or by month for trend identification
  - (Optional) Joined to customer or staff tables for deeper demographic or performance analysis
- **Interpretation:**
  All queries are performed and results interpreted in clear text cells for stakeholder readability.

4. Key Findings and Business Insights

Sample findings based on common use-cases (using available schema):

- **Order Fulfillment:**
  Identified pending and overdue orders easily by filtering `order_status` and comparing `order_date` vs. `shipped_date`.
- **Peak Activity:**
  Aggregated order counts by `order_date` to suggest business seasonality or staffing needs.
- **Staff/store performance:**
  Segmented orders by `store_id` and `staff_id` to inform operations about productivity or bottlenecks.

5. Business & Technical Impact

Business Impact:

- Empowers operations and management with fast, accurate operational reporting.
- Highlights "red flag" periods (e.g., shipping delays) to prioritize resource allocation.
- Lays groundwork for churn, loyalty, or performance studies as database complexity grows.

Technical Impact:

- Demonstrates proficiency running SQL in production-style notebooks.
- Shows ability to read, understand, and query across real-world schemas.
- Lays a foundation for advanced analytics: multi-table joins, aggregations, and reporting automation.

6. Recommendations

- Join orders to customers and staff for richer business/customer profiling.
- Visualize key analysis (e.g., orders per month, percent fulfillment) for easier business consumption.
- Expand project with optimization (indexing, query tuning) as volume grows.
- Automate reporting for ongoing monitoring and alerting on order delays.

7. Conclusion

This SQL notebook project proves essential data analyst capabilities: rapidly extracting insights from complex databases, communicating results clearly, and connecting data logic to real business needs. The skills shown here translate directly to technical interviews, dashboard building, and real-world analytical support for diverse industries.