Order-to-Cash (O2C) Business Analytics Platform

Cloud Data Engineering + Data Warehouse + BI Reporting

Executive Summary

Many companies generate revenue but still struggle with cash flow.
The real problem is not sales. The real problem is **how efficiently orders convert into collected payments**.

This project builds an end-to-end Order-to-Cash analytics platform that tracks the full financial lifecycle:

Customer → Order → Shipment → Invoice → Payment

The system ingests raw operational data, validates it, models it into an analytics warehouse, and exposes business insights through an interactive dashboard.

The goal is to help management answer:

• Are we growing?
• Are customers actually paying?
• Where is revenue getting stuck?
• Which customers are valuable vs risky?

Business Problem

In many organizations, sales, logistics, billing, and finance teams operate in separate systems. Because of this:

• Finance cannot track unpaid invoices quickly
• Delayed deliveries go unnoticed
• High-value customers are not identified
• Management sees revenue but not collection efficiency

The business requires a single analytics system that monitors the **entire revenue cycle**, not just sales.

This project simulates a company's operational data and builds a reporting platform to monitor financial performance and cash collection behavior.

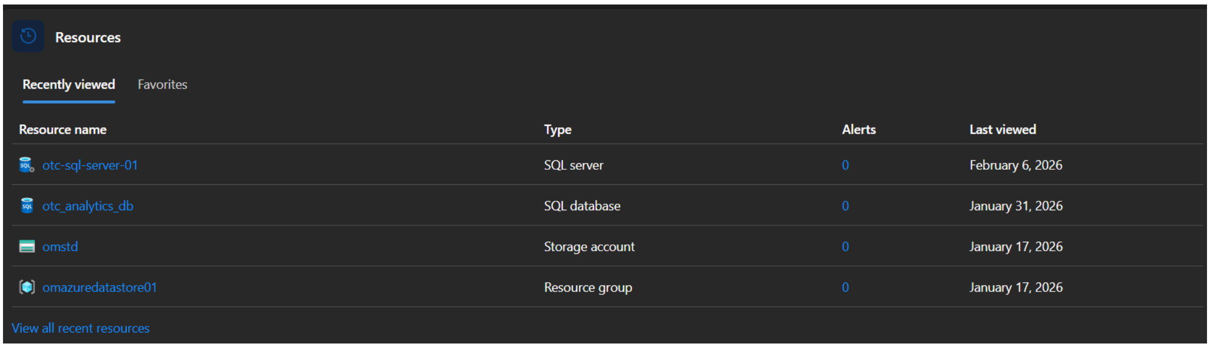
---

2. Solution Overview

The project implements a cloud-based data pipeline and reporting system.

Raw CSV Files
→ Azure Blob Storage
→ Python ETL (Cleaning + Validation)
→ Azure SQL Database (Analytics Warehouse)
→ Power BI Dashboard

In a production environment, the pipeline would be scheduled and automated using **Azure Data Factory**. Azure Data Factory would orchestrate ingestion, trigger validation, and refresh the warehouse automatically. The current project runs manually to demonstrate logic and data quality controls, but the architecture is ready for automation.



Technology Stack

Layer	Technology Used
Data Source	Raw CSV Operational Files
Cloud Storage	Azure Blob Storage
Data Processing	Python (Pandas ETL + Validation)
Data Warehouse	Azure SQL Database
Data Modeling	Star Schema (Facts & Dimensions)
Visualization	Power BI Dashboard

---

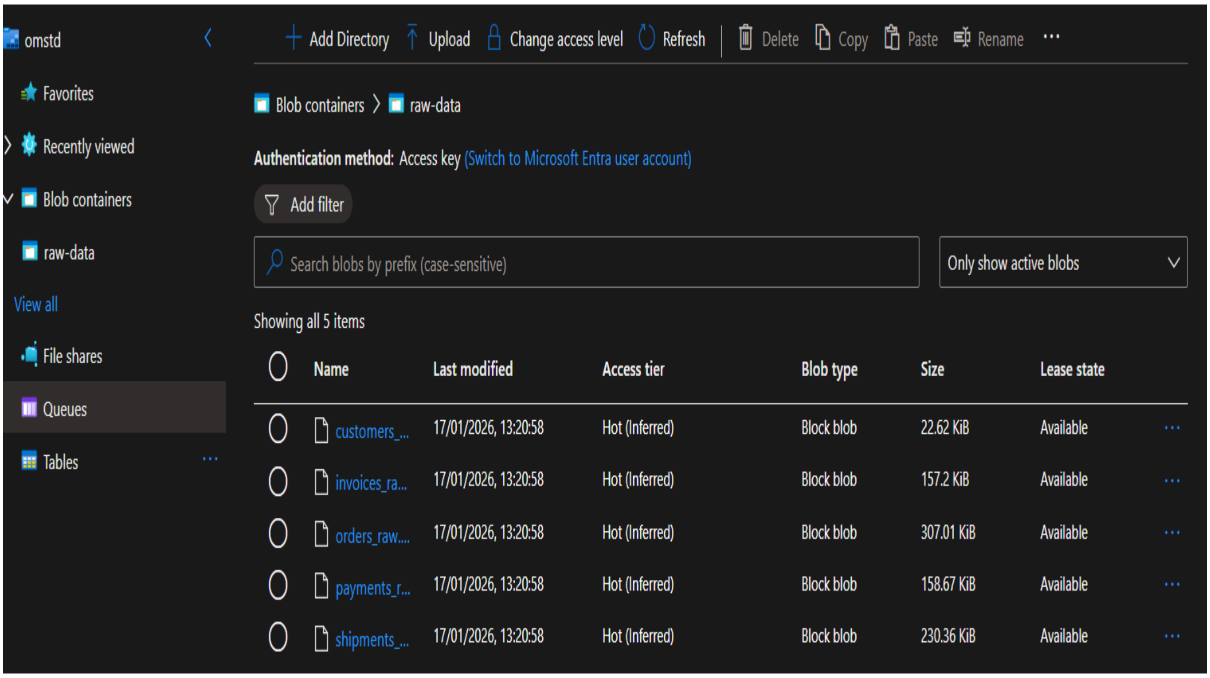
4. Data Ingestion (Cloud Storage)

All operational CSV files are stored in Azure Blob Storage as the raw data layer.

Tables ingested:

• Customers
• Orders
• Shipments
• Invoices
• Payments

Blob Storage acts as a landing zone similar to how companies receive ERP exports.



---

5. ETL Pipeline & Data Validation

A Python ETL script was developed to extract data from Blob Storage and load it into Azure SQL.

The ETL process performs:

Data Cleaning

• Date conversion
• Null handling
• Data type correction
• Status normalization

Data Validation Rules

Customers
• No duplicate customer IDs
• Valid regions only

Orders

- Valid customer reference (FK validation)
- Valid order status
- Positive order values

Shipments

- Linked to valid order
- Delivery date cannot be before ship date
- Delivered shipments must have delivery date

Invoices

- Linked to valid order
- Invoice amount > 0
- Valid invoice status

Payments

- Linked to valid invoice
- Payment amount > 0
- Valid payment status

The ETL is **idempotent**.
Every run clears and reloads tables safely in dependency order.

Load Order:
Customers → Orders → Shipments → Invoices → Payments

In a production environment, Azure Data Factory would schedule and trigger this ETL automatically.

etl_pipeline_execution.png

```
C:\Users\Om\azure_otc_project>python -u etl\otc_etl_with_validation.py
Clearing tables in FK-safe order
All tables cleared
Reading customers_raw.csv from Blob Storage
Validating customers data
Clearing existing customers table
Loading customers into Azure SQL
Customers load completed successfully
Reading orders_raw.csv from Blob Storage
Unique order_status values:
['Completed' 'Cancelled']
Rows with invalid promised_ship_date:
      order_id  customer_id                 order_date  order_value order_status promised_ship_date
2            3          170 2025-09-03 06:24:32.323090     40389.30    Cancelled                NaN
11          12           46 2024-09-08 06:24:32.323113     41491.78    Cancelled                NaN
25          26          107 2024-08-02 06:24:32.323143     39827.04    Cancelled                NaN
31          32          292 2024-12-20 06:24:32.323156     15149.17    Cancelled                NaN
32          33          257 2025-05-12 06:24:32.323159     23229.87    Cancelled                NaN
...        ...          ...                        ...          ...          ...                ...
3949      3950          137 2024-08-18 06:24:32.369225     24195.74    Cancelled                NaN
3954      3955          258 2025-06-05 06:24:32.369236     26593.79    Cancelled                NaN
3955      3956           58 2024-10-31 06:24:32.369238     38316.18    Cancelled                NaN
3980      3981          363 2025-07-06 06:24:32.369290     14742.06    Cancelled                NaN
3983      3984          324 2024-12-26 06:24:32.369296     43294.80    Cancelled                NaN

[396 rows x 6 columns]
Reading customers table for FK validation
Validating orders data
Loading orders into Azure SQL
Orders load completed successfully
Reading shipments_raw.csv from Blob Storage
Unique shipment_status values:
['Delivered' 'Delayed']
Reading orders table for FK validation
Validating shipments data
Loading shipments into Azure SQL
Shipments load completed successfully
Reading invoices_raw.csv from Blob Storage
Unique invoice_status values:
['Issued' 'Pending']
Reading orders table for FK validation
Validating invoices data
Loading invoices into Azure SQL
Invoices load completed successfully
Reading payments_raw.csv from Blob Storage
Unique payment_status values:
['Paid' 'Partial']
Reading invoices table for FK validation
Validating payments data
Loading payments into Azure SQL
Payments load completed successfully

C:\Users\Om\azure_otc_project>
```

6. Data Warehouse Design (Azure SQL Analytics Layer)

After the ETL process, the cleaned data is stored in an **Azure SQL analytics warehouse**.
Power BI does NOT connect to raw CSV files or staging tables. It connects only to this analytics layer.

A **Star Schema** model was implemented to support reliable reporting and fast aggregations.

Why this was necessary:

Operational data contains multiple transactions (orders, shipments, invoices, payments).
If reporting is built directly on transactional tables:

- filters propagate incorrectly
- DAX measures calculate wrong totals
- performance becomes slow

The star schema fixes this by separating descriptive data from transactional events.

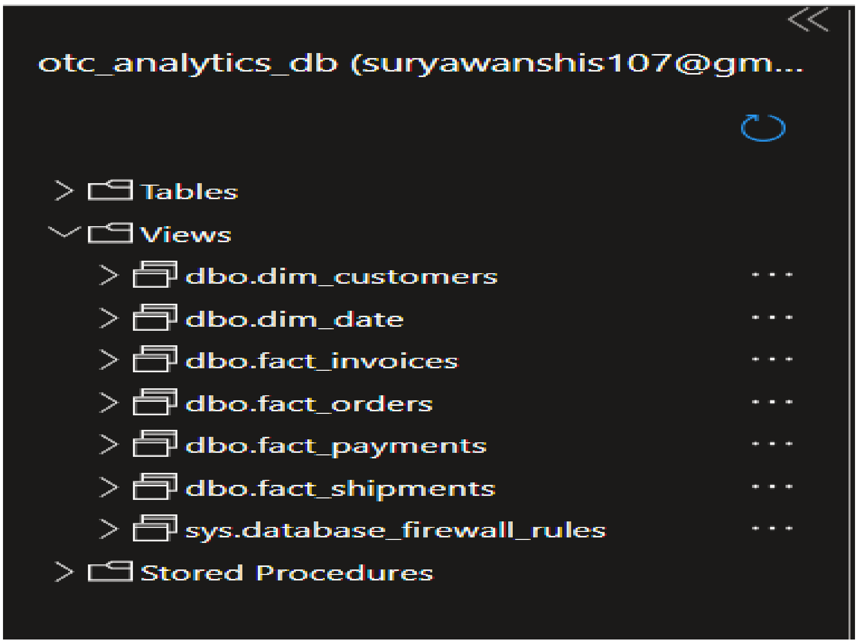
Schema Structure

Dimension Tables (used for filtering)

- **dim_customers** → customer and region information
- **dim_date** → calendar hierarchy (year, month, weekday)

Fact Tables (business events)

- **fact_orders** → order placed and order value
- **fact_shipments** → delivery execution
- **fact_invoices** → billing generation
- **fact_payments** → cash received



Business Process Relationship

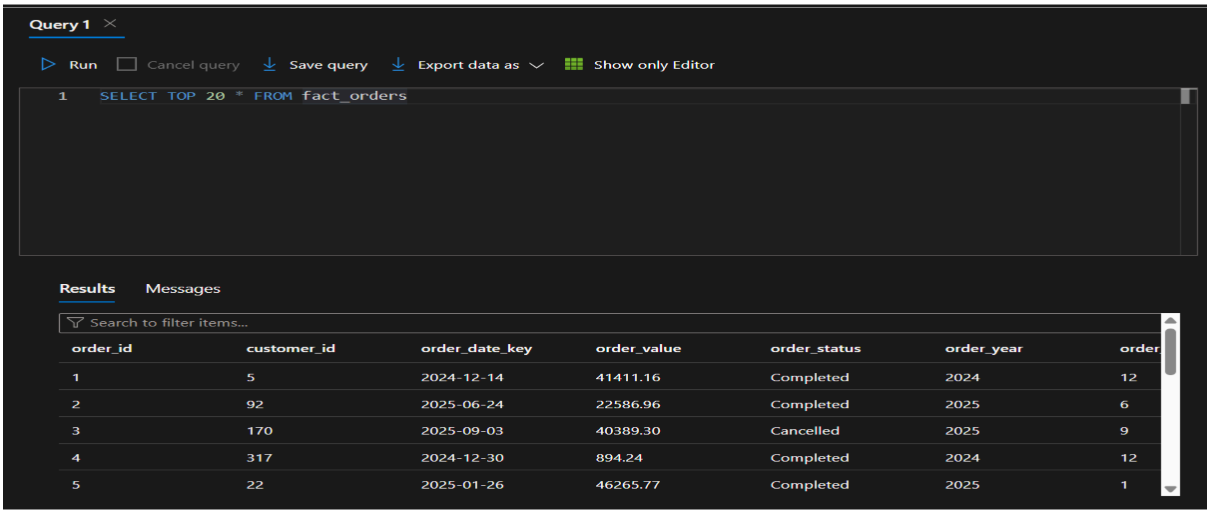
The warehouse models the real Order-to-Cash cycle:

Customer → Order → Shipment → Invoice → Payment

Relationships:

- fact_orders.customer_id → dim_customers.customer_id
- fact_orders.order_date_key → dim_date.date_key
- fact_shipments.order_id → fact_orders.order_id
- fact_invoices.order_id → fact_orders.order_id
- fact_payments.invoice_id → fact_invoices.invoice_id

This allows Power BI to trace revenue from customer acquisition to cash collection.



order_id	customer_id	order_date_key	order_value	order_status	order_year	order
1	5	2024-12-14	41411.16	Completed	2024	12
2	92	2025-06-24	22586.96	Completed	2025	6
3	170	2025-09-03	40389.30	Cancelled	2025	9
4	317	2024-12-30	894.24	Completed	2024	12
5	22	2025-01-26	46265.77	Completed	2025	1

Table Grain (Important for BI Calculations)

Each row represents a real business event:

Table	Represents
fact_orders	One order
fact_shipments	One shipment
fact_invoices	One invoice
fact_payments	One payment

Correct grain is required for accurate metrics like AOV, Collection Efficiency and CLV.

Why This Matters for Power BI

Power BI uses:

Dimensions → filtering
Facts → aggregation

Because filters flow one direction (dimension → fact):

• measures calculate correctly
• no ambiguous relationships
• dashboard performance improves

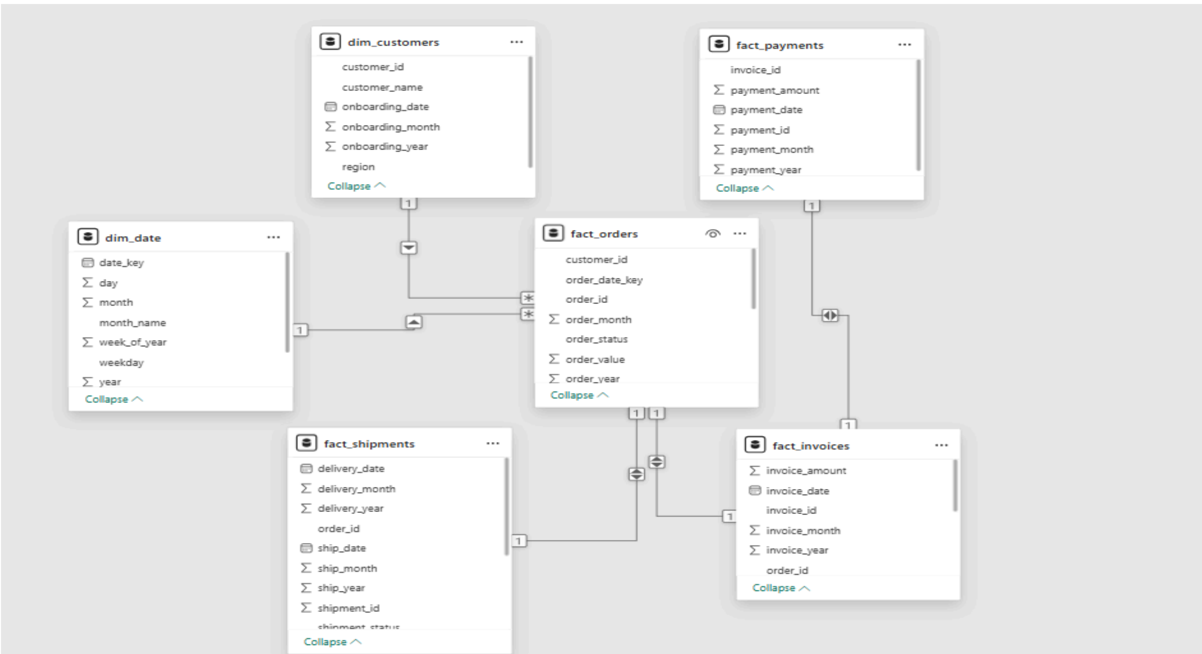
This warehouse layer enables analysis of revenue growth, order conversion, and payment collection efficiency.

Power BI Data Model

The warehouse is connected to Power BI and relationships are created to model the Order-to-Cash lifecycle.

Customer → Orders → Shipments → Invoices → Payments

The model allows tracking revenue from sale creation to final cash collection.

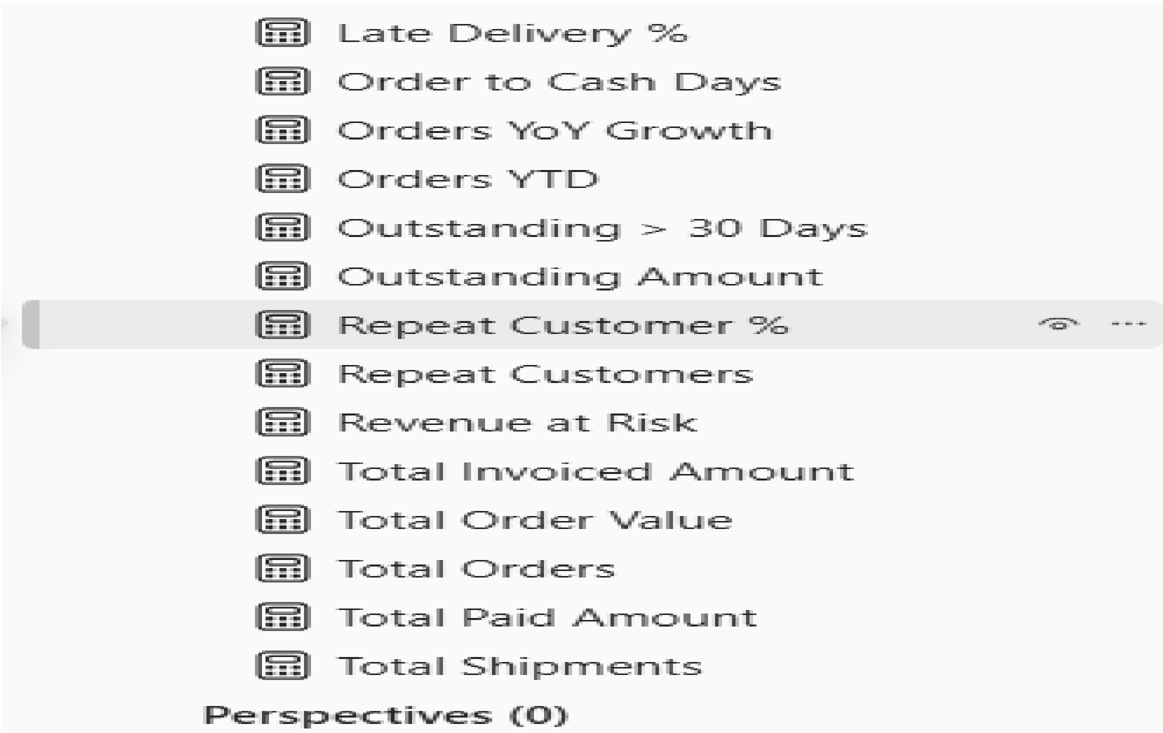


Business Metrics (KPIs)

The dashboard calculates business-level financial metrics:

• Total Revenue
• Collection Efficiency %
• Outstanding Amount
• Average Order Value (AOV)
• Repeat Customer Rate
• Customer Lifetime Value (CLV)
• Revenue at Risk
• Late Delivery Rate
• Aging Buckets (0-30, 31-60, 61-90, 90+ Days)

These metrics evaluate both **sales performance and financial health**.

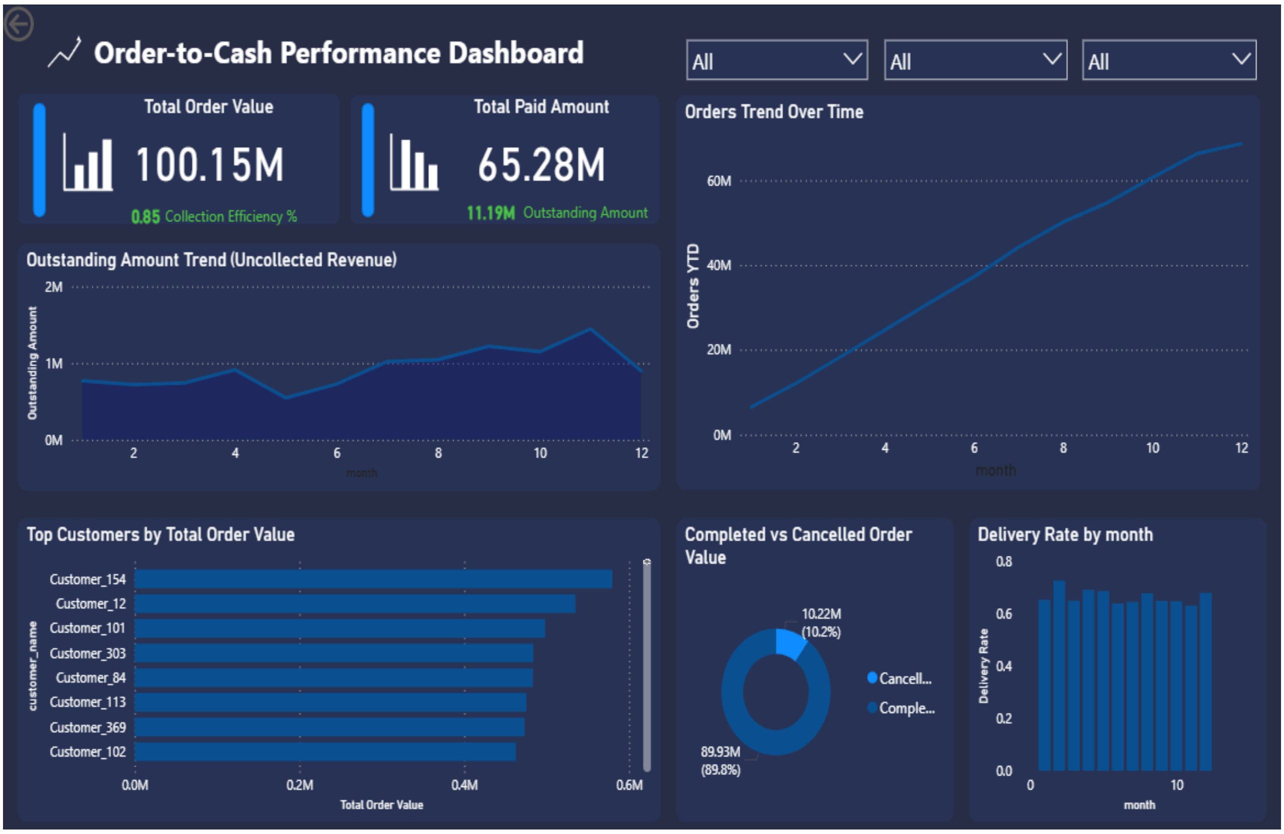


Interactive Dashboard

An executive dashboard was built to monitor operational and financial performance.

It enables management to:

- Track revenue trends over time
- Monitor order growth
- Identify top customers
- Detect payment delays
- Evaluate collection efficiency
- Monitor overdue invoices



Key Insights Delivered

The dashboard helps stakeholders identify:

- Customers who generate high revenue but delay payments
- Revenue stuck in overdue invoices
- Delivery performance impact on payments
- Collection efficiency trends month-over-month

The system converts operational data into actionable financial intelligence.

Conclusion

This project demonstrates a complete data analytics workflow, not just visualization.

It covers:

Data ingestion → Data validation → Data warehousing → Data modeling → Business intelligence reporting

The platform provides a unified view of the company's revenue cycle and enables better financial decision-making by tracking how effectively revenue converts into cash.