OPERATIONAL BOTTLENECK ANALYSIS USING PYTHON FOR A MULTI-WAREHOUSE FULFILLMENT COMPANY IN THE U.S.

1. Background

A mid-sized U.S. fulfillment company operating five regional warehouses faced increasing order backlog, fulfillment delays, and customer SLA violations. Internal KPIs showed rising warehouse cycle times and inconsistent throughput across locations.

The client sought a Python-powered solution to analyze daily order flow, identify root causes of delays, and deliver actionable reporting. We were engaged to build a reproducible Jupyter-based workflow for warehouse analytics, performance tracking, and executive reporting.

2. Objective

- To conduct in-depth analysis of warehouse-level operational metrics and uncover the source of fulfillment delays
- To create a reusable Python script for daily performance monitoring
- To deliver clear visual reports highlighting bottlenecks by time of day, process step, or location
- To enable operational managers to track and respond to issues without additional tools

3. Data Used

Source: Internal WMS (Warehouse Management System) and shipment logs

Timeframe: 60 days of daily operational data across five warehouses

Dataset Size: 310,000 order-level records

Key Fields:

- Order_ID
- Warehouse_ID
- Order_Date
- Picked_Time, Packed_Time, Shipped_Time
- Order_Size (Small/Medium/Large)

- Delay_Reason (Late Pick, Missing Stock, System Downtime, etc.)
- Shipping_Method (Standard, Expedited, Same-Day)

4. Methodology

4.1 Data Preparation

- Calculated process lags:
    - Picking_Duration = Packed_Time – Picked_Time
    - Packing_Duration = Shipped_Time – Packed_Time
    - Total_Fulfillment_Time = Shipped_Time – Order_Date
- Flagged late orders based on method-specific SLA thresholds
- Merged and cleaned WMS and support ticket data to link delay reasons with order IDs
- Converted datetime fields to time blocks (e.g., Shift 1, Shift 2, Shift 3)

4.2 Analysis Performed

- Aggregated late order frequency by warehouse, shift, order type
- Used pivot tables to map time-of-day patterns across locations
- Applied correlation and group-based analysis on delay reasons
- Created performance scorecards by warehouse and staff shift

4.3 Tools Used

- Environment: Anaconda with Jupyter Notebook and VS Code
- Libraries: pandas, numpy, seaborn, matplotlib, datetime, plotly
- Reporting: Dynamic plots, daily summary exports to Excel, PDF visualization reports

5. Key Results

- **Late order rate**: 22.4% (before intervention)
- **Warehouse C** had the highest delay rate (34.1%), particularly during evening shifts
- **Top 3 delay reasons**:
    1. Inventory stockouts (41.5%)
    2. Picking delays due to floor congestion (26.8%)

3. Shift transition lag (12.3%)

- **Small orders using Same-Day shipping** had the highest SLA breach rate (38.7%)
- Median fulfillment time varied significantly across warehouses:
  - Warehouse A: 3.2 hrs
  - Warehouse C: 6.8 hrs

6. Insights Delivered

- Created visual dashboard showing hourly congestion and pick-pack lag patterns
- Identified that evening shift transitions lacked buffer staff, causing pick-time spikes
- Suggested warehouse-level SLA rebalancing (e.g., restrict same-day options post 3 PM)
- Provided analysis of stockout-prone SKUs causing delay in over 4,800 orders

7. Reporting Output

- **Python Script**:
  - Daily rerun script for tracking delay rates per warehouse and shift
  - Modular code with adjustable SLA rules and calendar filters
  - Built-in export to Excel, HTML, and PDF formats
- **PDF Report (Business Style)**:
  - Warehouse comparison charts
  - Sankey diagram of order flow and blockages
  - KPI table by date, warehouse, and delay cause
- **Excel Dashboard**:
  - Interactive pivot-based tracker for daily operations team
  - Highlighted “Red Zones” (dates with >25% delays)
  - Order type × delay reason matrix for shift supervisors

8. Business Impact

- Within two months, **average delay rate dropped to 18.3%** after staff realignment and process fixes
- The client restructured shift overlap periods and restocked high-delay SKUs more frequently
- Reassigned 12% of Same-Day orders to a better-performing warehouse
- Python workflow was adopted as a **weekly performance audit tool** by the operations team