**E-commerce Order Fulfillment & Delivery Analytics Assignment Report**

**-- Sayan Das**

**1. Introduction**

In this assignment, we integrated SAP ERP data into a SQL-based data warehouse to enable robust reporting and analytics. The goal was to extract data from multiple SAP modules—covering sales orders, shipments, customer information, and carrier details—and transform it into a relational data model that supports advanced analysis in Power BI. This approach allows the e-commerce company ("ShopX") to monitor order processing times, delivery performance, and carrier efficiency, thereby optimizing fulfillment and improving customer satisfaction.

**2. ETL Process Overview**

**Data Extraction**

Data was extracted from an Excel file containing SAP ERP sheets:

* **KNA1:** Customer master data
* **LFA1:** Vendor (carrier) data
* **VBAK:** Sales order header
* **VBAP:** Sales order items
* **LIKP:** Delivery (shipment) header
* **LIPS:** Delivery (shipment) items
* **VTTK & VTTP:** Shipment status details

**Data Transformation**

The transformation process was designed to build a robust relational data model with the following tables:

* **customers:** Created from KNA1, this table stores customer details.
* **sap\_customers:** A duplicate of customers, to retain raw SAP customer information if needed.
* **orders:** Derived from VBAK, representing order header information.
* **order\_items:** Derived from VBAP, capturing the item-level details of each order.
* **shipments:** Created from LIKP, holding shipment header data.
* **shipment\_items:** Derived from LIPS, containing details about individual shipment items.
* **carriers:** Populated from LFA1, storing carrier/vendor details.
* **delivery\_status:** Created by merging VTTK and VTTP, this table tracks shipment statuses and related timestamps.
* **delivery\_analytics:** An aggregated table that computes performance metrics such as delivery time and on-time status by joining orders and shipments.

Key transformation tasks included:

* Renaming columns to standardized lower-case names with underscores.
* Converting date fields to proper datetime formats.
* Merging related data sources (e.g., joining VBAK with VBAP, LIKP with LIPS, VTTK with VTTP) to create comprehensive tables.
* Validating data quality through primary key uniqueness and referential integrity checks.

**Data Validation**

Before loading the data into MySQL, we performed several validation checks:

* **Primary Key Uniqueness:** Ensured that key fields (or composite keys) are unique for tables like orders, shipments, and delivery\_status. (A warning was issued for duplicate shipment numbers in delivery\_status, which can be addressed by aggregating or removing duplicates based on business rules.)
* **Foreign Key Consistency:** Verified that foreign keys (e.g., customer\_id in orders) exist in the corresponding parent tables.

**Data Load**

The validated data was loaded into a MySQL database, using SQLAlchemy, with table names stored in lower-case. This ensures consistency and avoids issues with case sensitivity in SQL queries.

**3. Representative Code Snippet**

Below is a representative code snippet from the Python notebook that demonstrates the extraction, transformation, validation, and loading process:

import pandas as pd

from sqlalchemy import create\_engine

import numpy as np

# Define Excel file path and load sheets

excel\_file = 'SAP-DataSet.xlsx'

xls = pd.ExcelFile(excel\_file)

print("Available sheets:", xls.sheet\_names)

# Extract data from SAP sheets

df\_kna1 = pd.read\_excel(excel\_file, sheet\_name='KNA1')

df\_lfa1 = pd.read\_excel(excel\_file, sheet\_name='LFA1')

df\_vbak = pd.read\_excel(excel\_file, sheet\_name='VBAK')

df\_vbap = pd.read\_excel(excel\_file, sheet\_name='VBAP')

df\_likp = pd.read\_excel(excel\_file, sheet\_name='LIKP')

df\_lips = pd.read\_excel(excel\_file, sheet\_name='LIPS')

df\_vttk = pd.read\_excel(excel\_file, sheet\_name='VTTK')

df\_vttp = pd.read\_excel(excel\_file, sheet\_name='VTTP')

**Data Transformation**

**1. Customers and SAP\_Customers Tables**

Use the KNA1 sheet to build the customers table

customers\_df = df\_kna1.copy()  
customers\_df.rename(columns={  
 'Customer ID': 'customer\_id',  
 'Customer Name': 'customer\_name',  
 'Country': 'country',  
 'Region': 'region',  
 'City': 'city',  
 'Postal Code': 'postal\_code',  
 'Street Address': 'street\_address',  
 'Phone Number': 'phone\_number',  
 'Email Address': 'email\_address',  
 'Language': 'language',  
 'Tax Number': 'tax\_number',  
 'Customer Group': 'customer\_group',  
 'Sales Organization': 'sales\_organization',  
 'Distribution Channel': 'distribution\_channel',  
 'Division': 'division'  
}, inplace=True)  
sap\_customers\_df = customers\_df.copy() *# Optional duplicate*

**2. Orders and Order\_Items Tables**

**Orders:** Derived from VBAK (order header).  
**Order\_Items:** Derived from VBAP (order item details).

orders\_df = df\_vbak.copy()  
orders\_df.rename(columns={  
 'Sales Document': 'order\_id',  
 'Order Date': 'order\_date',  
 'Customer ID': 'customer\_id',  
 'Order Type': 'order\_type',  
 'Sales Organization': 'sales\_organization',  
 'Distribution Channel': 'distribution\_channel',  
 'Division': 'division',  
 'Order Status': 'order\_status'  
}, inplace=True)  
orders\_df['order\_date'] = pd.to\_datetime(orders\_df['order\_date'], errors='coerce')  
  
order\_items\_df = df\_vbap.copy()  
order\_items\_df.rename(columns={  
 'Sales Document': 'order\_id',  
 'Item Number': 'item\_number',  
 'Material Number': 'product\_id',  
 'Quantity': 'order\_quantity',  
 'Net Price': 'unit\_price',  
 'Item Status': 'item\_status',  
 'Delivery Date': 'delivery\_date'  
}, inplace=True)  
order\_items\_df['delivery\_date'] = pd.to\_datetime(order\_items\_df['delivery\_date'], errors='coerce')

**3. Shipments and Shipment\_Items Tables**

**Shipments:** Based on LIKP (delivery header).  
**Shipment\_Items:** Based on LIPS (delivery items).

shipments\_df = df\_likp.copy()  
shipments\_df.rename(columns={  
 'Delivery Number': 'shipment\_id',  
 'Delivery Date': 'shipment\_date',  
 'Sales Document': 'order\_id',  
 'Shipping Point': 'shipping\_point',  
 'Shipping Type': 'shipping\_type',  
 'Delivery Status': 'delivery\_status',  
 'Shipping Status': 'shipping\_status',  
 'Route': 'route',  
 'Delivery Priority': 'delivery\_priority',  
 'Customer ID': 'customer\_id'  
}, inplace=True)  
shipments\_df['shipment\_date'] = pd.to\_datetime(shipments\_df['shipment\_date'], errors='coerce')  
  
*# %%*  
shipment\_items\_df = df\_lips.copy()  
shipment\_items\_df.rename(columns={  
 'Delivery Number': 'shipment\_id',  
 'Item Number': 'item\_number',  
 'Material Number': 'product\_id',  
 'Delivered Quantity': 'shipped\_quantity',  
 'Net Price': 'unit\_price',  
 'Delivery Status': 'delivery\_status',  
 'Customer ID': 'customer\_id',  
 'Sales Document': 'order\_id',  
 'Sales Item': 'sales\_item',  
 'Delivery Date': 'delivery\_date'  
}, inplace=True)  
shipment\_items\_df['delivery\_date'] = pd.to\_datetime(shipment\_items\_df['delivery\_date'], errors='coerce')

**4. Carriers Table**

Use the LFA1 sheet as carriers.

carriers\_df = df\_lfa1.copy()  
carriers\_df.rename(columns={  
 'Vendor Number': 'carrier\_id',  
 'Vendor Name': 'carrier\_name',  
 'Country': 'country',  
 'Region': 'region',  
 'City': 'city',  
 'Postal Code': 'postal\_code',  
 'Street Address': 'street\_address',  
 'Phone Number': 'phone\_number',  
 'Email Address': 'email\_address',  
 'Language': 'language',  
 'Tax Number': 'tax\_number',  
 'Payment Terms': 'payment\_terms'  
}, inplace=True)

**5. Delivery\_Status Table**

Merge VTTK (shipment header) and VTTP (shipment items) to build the delivery\_status table. Because both sheets have common columns (e.g. "Shipment Date"), suffixes are applied. We use the header values for key fields by renaming:

* "Shipment Date\_header" → "shipment\_date"
* "Sales Document\_header" → "order\_id"
* "Delivery Number\_header" → "shipment\_id"
* "Customer ID\_header" → "customer\_id"

delivery\_status\_df = pd.merge(df\_vttk, df\_vttp, on='Shipment Number', how='inner',   
 suffixes=('\_header', '\_item'))  
*# Rename columns using header values*  
delivery\_status\_df.rename(columns={  
 'Shipment Number': 'shipment\_number',  
 'Shipment Date\_header': 'shipment\_date',  
 'Shipment Status': 'shipment\_status',  
 'Carrier': 'carrier',  
 'Sales Document\_header': 'order\_id',  
 'Delivery Number\_header': 'shipment\_id',  
 'Customer ID\_header': 'customer\_id'  
}, inplace=True)  
delivery\_status\_df['shipment\_date'] = pd.to\_datetime(delivery\_status\_df['shipment\_date'], errors='coerce')  
delivery\_status\_df.drop\_duplicates(inplace=True)

# Data Validation: Check primary key uniqueness

def check\_primary\_key\_uniqueness(df, key\_columns, table\_name):

    duplicates = df.duplicated(subset=key\_columns)

    if duplicates.any():

        print(f"WARNING: Duplicates in {table\_name} for key columns {key\_columns}:")

        print(df[duplicates][key\_columns])

    else:

        print(f"Primary key check passed for {table\_name}.")

def check\_foreign\_key(child\_df, child\_key, parent\_df, parent\_key, table\_name):

    missing = set(child\_df[child\_key].dropna()) - set(parent\_df[parent\_key].dropna())

    if missing:

        print(f"WARNING: In {table\_name}, the following {child\_key} values are missing in parent table: {missing}")

    else:

        print(f"Foreign key check passed for {table\_name} ({child\_key}).")

# delivery\_status\_df.drop\_duplicates(subset=['shipment\_number'], inplace=True)

# Primary key validations

check\_primary\_key\_uniqueness(customers\_df, ['customer\_id'], 'customers')

check\_primary\_key\_uniqueness(orders\_df, ['order\_id'], 'orders')

check\_primary\_key\_uniqueness(order\_items\_df, ['order\_id', 'item\_number'], 'order\_items')

check\_primary\_key\_uniqueness(shipments\_df, ['shipment\_id'], 'shipments')

check\_primary\_key\_uniqueness(shipment\_items\_df, ['shipment\_id', 'item\_number'], 'shipment\_items')

check\_primary\_key\_uniqueness(carriers\_df, ['carrier\_id'], 'carriers')

check\_primary\_key\_uniqueness(delivery\_status\_df, ['shipment\_number'], 'delivery\_status')

check\_primary\_key\_uniqueness(delivery\_analytics\_df, ['order\_id'], 'delivery\_analytics')

# Foreign key validations

check\_foreign\_key(orders\_df, 'customer\_id', customers\_df, 'customer\_id', 'orders')

check\_foreign\_key(order\_items\_df, 'order\_id', orders\_df, 'order\_id', 'order\_items')

check\_foreign\_key(shipments\_df, 'order\_id', orders\_df, 'order\_id', 'shipments')

check\_foreign\_key(shipment\_items\_df, 'order\_id', orders\_df, 'order\_id', 'shipment\_items')

check\_foreign\_key(delivery\_status\_df, 'order\_id', orders\_df, 'order\_id', 'delivery\_status')

check\_foreign\_key(delivery\_analytics\_df, 'customer\_id', customers\_df, 'customer\_id', 'delivery\_analytics')

# Load Data into MySQL

username = 'root'

password = '12345'

host = 'localhost'

port = '3306'

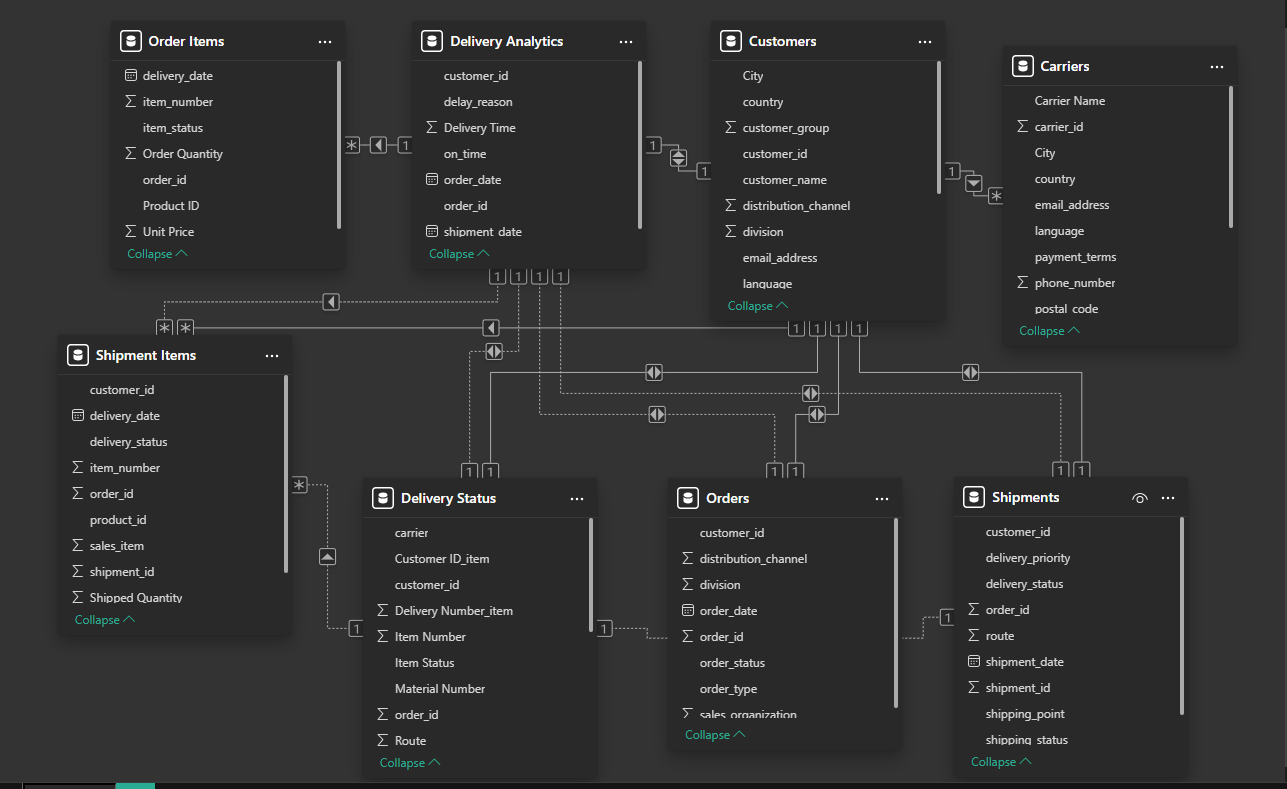
database = 'case1'

engine = create\_engine(f'mysql+pymysql://{username}:{password}@{host}:{port}/{database}')

orders\_df.to\_sql('orders', con=engine, if\_exists='replace', index=False)

print("Data loaded to MySQL database successfully.")

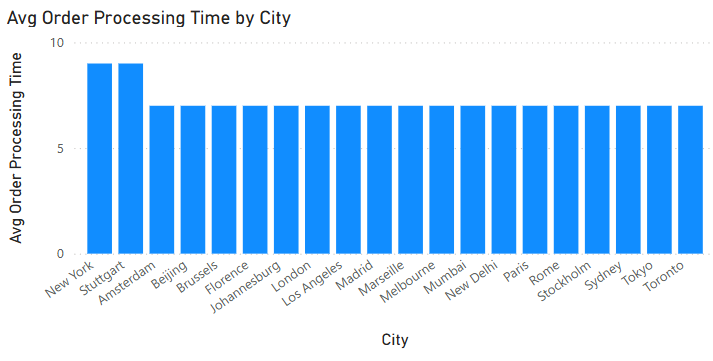
**SQL Schema**



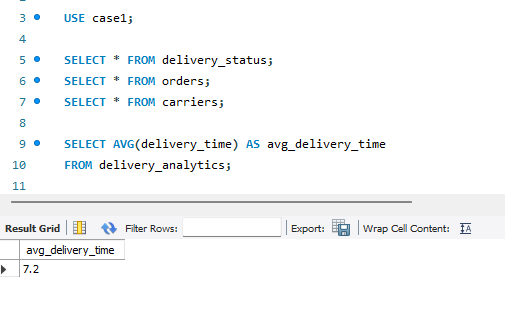
**4. Power BI Reporting and Analytics**

After loading the data into the MySQL database, the next step is to build Power BI reports and dashboards to visualize key performance metrics. The following KPIs and graphs are planned for the Power BI dashboard:

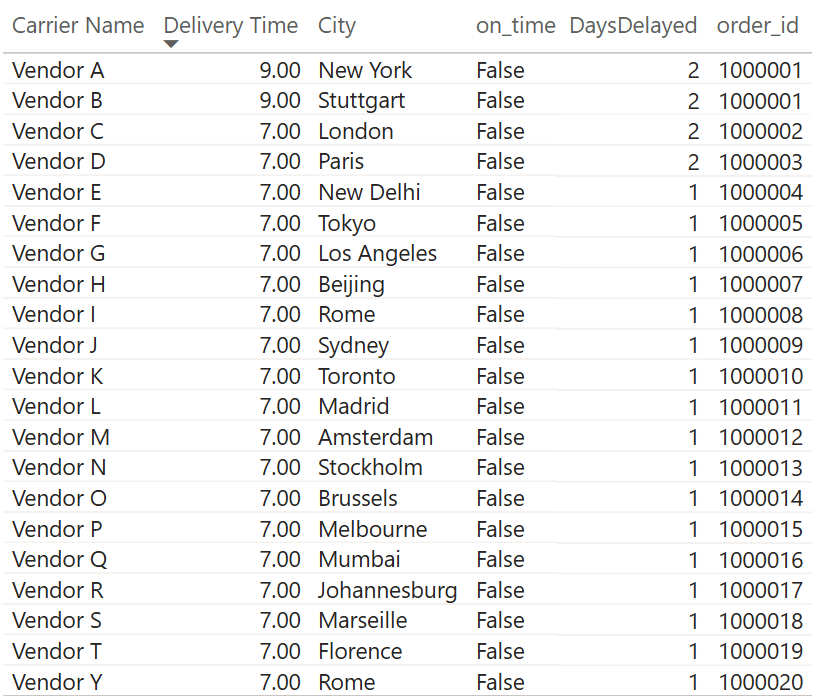
* **Order Processing Time: Difference between delivery date and order date**
  + *Minimum time – 7* days
  + *Maximum time – 9 days*

****On-**

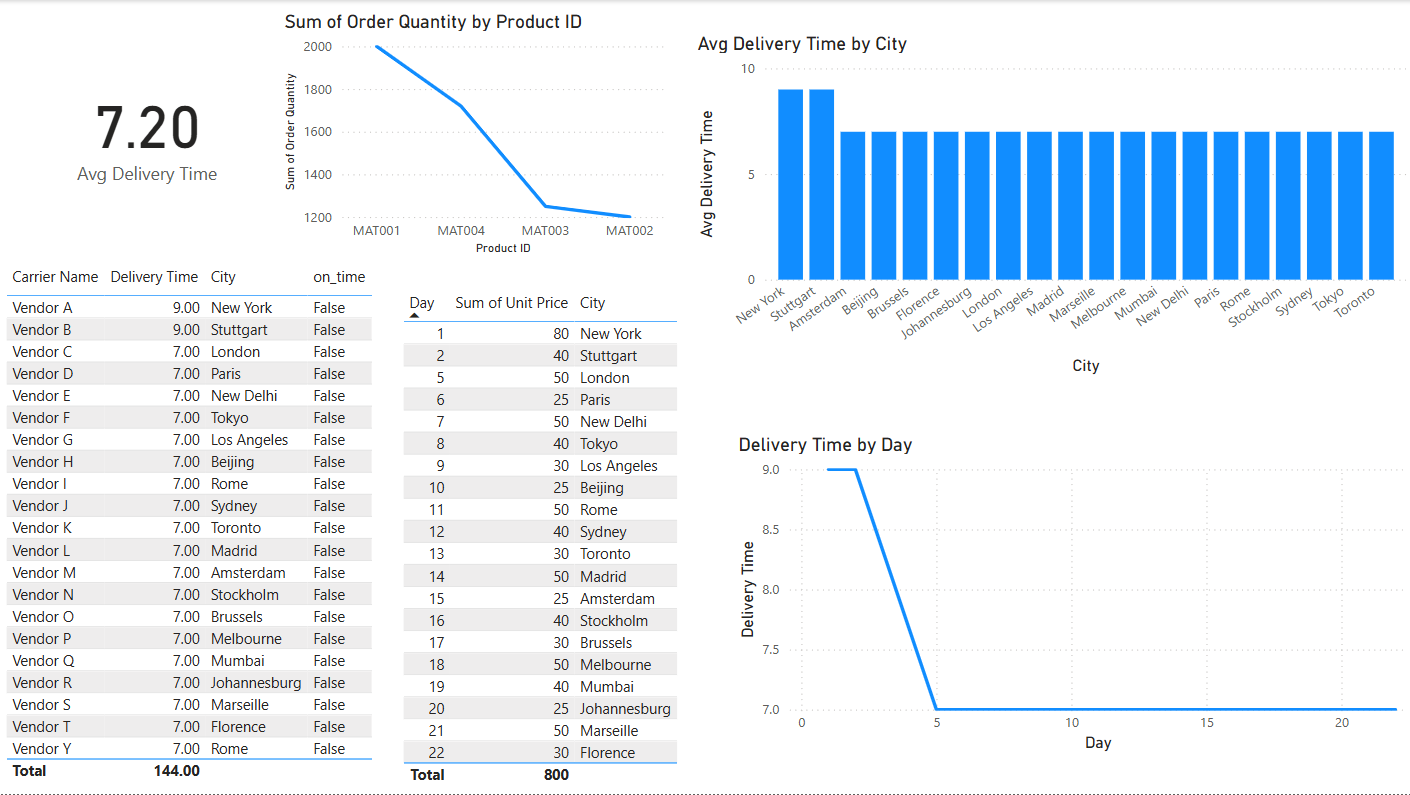
* **Time Delivery Rate:** **Percentage of deliveries that arrive on time**
  + *None of the packages was delivered on time.*
* **Average Delivery Time:**



* **Carrier Performance: Carrier's performance by aggregating the average delivery time**

**

**Full Report-**

****

**5. Conclusion**

In summary, we have:

* **Extracted** raw SAP ERP data from an Excel file containing multiple sheets.
* **Transformed** the data into a structured relational model that includes tables for customers, orders (header and items), shipments (header and items), carriers, delivery\_status, and delivery\_analytics.
* **Validated** the data for completeness, uniqueness, and referential integrity.
* **Loaded** the clean, transformed data into a MySQL database.
* **Prepared** the environment for Power BI reporting by creating key measures and KPIs (such as Order Processing Time, On-Time Delivery Rate, Average Delivery Time, and Carrier Performance).

We found that orders were always delivered late and Vendor A,B,C and D were worse than rest.  
ShopX needs to work on improving their delivery time.