Logistic & Delivery Time Optimization

BY HITESH ANANTH

Data Preprocessing

Data preprocessing is a crucial step in the data analysis process that involves cleaning, transforming, and organizing raw data into a structured and usable format.

In real-world logistics data, raw datasets often contain:

- Missing values
- Incorrect data types
- Duplicate records
- Inconsistent date formats
- Invalid delivery timelines

To ensure accurate analysis and meaningful insights, these issues must be addressed before visualization and modeling.

For this project, **Python** was used as the primary tool for data preprocessing due to its efficiency, flexibility, and strong data-handling libraries.

```python
# STEP 1: Import Required Libraries

import pandas as pd
import numpy as np

# STEP 2: Load the Raw Dataset
df = pd.read_csv("logistics_delivery_raw_data.csv")

print("Dataset Loaded Successfully")
print(df.head())
```

- In this step, the required Python libraries **Pandas** and **NumPy** are imported to handle data manipulation and numerical operations.
- The raw logistics dataset is then loaded using the `read_csv()` function. After loading, the first few rows of the dataset are displayed using `head()` to verify that the data has been imported correctly.
- This step ensures the dataset is successfully loaded and ready for further preprocessing.

```
# STEP 3: Basic Dataset Information
print("\nDataset Info:")
print(df.info())

print("\nDataset Shape (Rows, Columns):")
print(df.shape)

# STEP 4: Convert Date Columns to Date Format
date_columns = ["Order_Date", "Dispatch_Date", "Delivery_Date"]

for col in date_columns:
    df[col] = pd.to_datetime(df[col], errors="coerce")

print("\nDate Conversion Completed")
```

- In this step, the overall structure of the dataset is examined using df.info() to understand column names, data types, and missing values.
  The dataset size is also checked using df.shape to identify the total number of rows and columns.
- Next, all date-related columns (Order_Date, Dispatch_Date, and Delivery_Date) are converted into a proper date format using pd.to_datetime().
  The errors='coerce' parameter ensures that invalid date values are automatically handled as missing values.
- This step is important to enable accurate time-based analysis in later stages.

```
# STEP 5: Check Missing Values
print("\nMissing Values Count:")
print(df.isnull().sum())

# STEP 6: Handle Missing Values
# Fill missing delay reasons with 'None'
df["Delay_Reason"].fillna("None", inplace=True)

# Fill missing customer ratings with median
df["Customer_Rating"].fillna(df["Customer_Rating"].median(), inplace=True)

# Drop rows where important values are missing
df.dropna(subset=["Order_Date", "Dispatch_Date", "Delivery_Date"], inplace=True)

print("\nMissing Values After Handling:")
print(df.isnull().sum())
```

- In this step, missing values in the dataset are identified using isnull().sum() to understand which columns require cleaning.
- Missing values in the **Delay_Reason** column are filled with "None" to indicate that no delay occurred.
  Missing values in **Customer_Rating** are replaced with the median value to avoid skewing the data.
- Rows with missing critical date information (Order_Date, Dispatch_Date, Delivery_Date) are removed, as these records cannot be used for delivery time analysis.
- Finally, missing values are checked again to confirm successful data cleaning.

```
# STEP 7: Remove Duplicate Records
duplicates_before = df.duplicated().sum()
df.drop_duplicates(inplace=True)
duplicates_after = df.duplicated().sum()

print(f"\nDuplicates before removal: {duplicates_before}")
print(f"Duplicates after removal: {duplicates_after}")

# STEP 8: Validate Date Logic
# Delivery_Date >= Dispatch_Date >= Order_Date
df = df[
    (df["Delivery_Date"] >= df["Dispatch_Date"]) &
    (df["Dispatch_Date"] >= df["Order_Date"])
]

print("\nInvalid date records removed")
```

- In this step, duplicate records in the dataset are identified and removed to ensure that each order is counted only once. This prevents data duplication from affecting delivery performance metrics.
- Next, date logic is validated by ensuring that the **Delivery Date is greater than or equal to the Dispatch Date**, and the **Dispatch Date is greater than or equal to the Order Date**.
  Records that do not follow this logical sequence are removed, as they represent invalid delivery timelines.
- This step improves the reliability and accuracy of time-based analysis.

```python
# STEP 9: Feature Engineering - Delivery Time
df["Delivery_Time_Days"] = (
    df["Delivery_Date"] - df["Dispatch_Date"]
).dt.days

print("\nDelivery Time Column Added")

# STEP 10: Create Delay Flag Column
df["Is_Delayed"] = df["Delivery_Status"].apply(
    lambda x: 1 if x == "Delayed" else 0
)

print("\nDelay Flag Column Added")
```

- In this step, new meaningful features are created to support logistics analysis.
- A new column **Delivery_Time_Days** is calculated by finding the difference between the delivery date and dispatch date. This represents the actual time taken to deliver each order and is a key metric for delivery performance evaluation.
- Additionally, a **Delay Flag (Is_Delayed)** column is created, where delayed deliveries are marked as 1 and non-delayed deliveries as 0. This binary indicator helps in identifying and analyzing delayed orders easily.
- Feature engineering enhances the dataset by converting raw data into actionable insights.

```
# STEP 11: Outlier Inspection (Summary Stats)
print("\nSummary Statistics:")
print(df[["Distance_km", "Shipping_Cost", "Shipment_Weight_kg"]].describe())

# STEP 12: Final Cleaned Dataset Preview
print("\nCleaned Dataset Preview:")
print(df.head())

print("\nFinal Dataset Shape (Rows, Columns):")
print(df.shape)

# STEP 13: Save Cleaned Dataset
df.to_csv("logistics_delivery_cleaned_data.csv", index=False)

print("\nCleaned Dataset Saved as 'logistics_delivery_cleaned_data.csv'")
```

- In this step, summary statistics are generated for key numerical columns such as **Distance_km**, **Shipping_Cost**, and **Shipment_Weight_kg** using the describe() function. This helps in identifying potential outliers and understanding the overall distribution of the data.
- Next, a preview of the fully cleaned dataset is displayed to verify that all preprocessing steps have been applied successfully. The final dataset shape is also checked to confirm the number of rows and columns after cleaning.
- Finally, the cleaned dataset is saved as a new CSV file, which is later used for visualization and analysis in Power BI.

Cleaned Dataset

Order_ID	Customer_	Warehous	Order_Date	Dispatch_Date	Delivery_Date	Origin_City	Destination_City	Distance_km	Transport_Mode	Shipment_Weight_kg	Shipping_Cost	Delivery_Status	Delay_Reason	Customer_Rating	Delivery_Time_Days	I
100001	4507	WH_A	11-03-2024	11-03-2024	18-06-2024	Bangalore	Ahmedabad	1480	Rail	248	32016	Delayed	Weather	3	99	
100006	4772	WH_D	05-02-2024	11-04-2024	21-06-2024	Mumbai	Jaipur	1812	Road	115	39087	On Time	Traffic	4	71	
100017	3919	WH_B	01-01-2024	16-06-2024	01-07-2024	Delhi	Jaipur	616	Sea	465	40897	Early	None	3	15	
100019	1130	WH_C	04-02-2024	29-05-2024	16-06-2024	Bangalore	Pune	1043	Sea	71	25747	Early	Traffic	3	18	
100021	4380	WH_B	23-01-2024	30-01-2024	18-07-2024	Chennai	Pune	1678	Sea	492	49762	Delayed	Weather	5	170	
100026	3853	WH_B	04-01-2024	29-06-2024	18-07-2024	Mumbai	Coimbatore	1262	Air	175	15872	Early	None	1	19	
100028	2215	WH_B	11-04-2024	21-06-2024	25-06-2024	Bangalore	Kolkata	1396	Road	343	1972	Delayed	Traffic	1	4	
100031	2184	WH_D	12-03-2024	15-05-2024	23-05-2024	Mumbai	Ahmedabad	1859	Sea	398	31302	Delayed	Mechanical Issue	4	8	
100032	1459	WH_A	03-01-2024	20-06-2024	02-07-2024	Mumbai	Kolkata	1254	Sea	224	5372	Delayed	Weather	4	12	
100033	4385	WH_D	21-05-2024	15-06-2024	19-07-2024	Bangalore	Kolkata	1008	Road	499	28047	On Time	Mechanical Issue	2	34	
100041	3558	WH_A	16-01-2024	03-05-2024	23-05-2024	Mumbai	Coimbatore	311	Rail	314	40084	Delayed	Mechanical Issue	1	20	
100042	4753	WH_C	06-03-2024	23-04-2024	14-06-2024	Hyderabad	Ahmedabad	1596	Sea	309	30941	On Time	Traffic	5	52	
100046	1975	WH_B	10-06-2024	21-06-2024	11-07-2024	Chennai	Jaipur	774	Road	57	10135	On Time	Traffic	5	20	
100049	4005	WH_D	11-04-2024	19-06-2024	07-07-2024	Bangalore	Jaipur	70	Air	272	35669	Early	Traffic	2	18	
100051	4005	WH_D	13-01-2024	26-03-2024	27-04-2024	Hyderabad	Jaipur	309	Rail	495	44631	On Time	None	2	32	
100054	4638	WH_B	03-02-2024	05-04-2024	27-05-2024	Delhi	Kolkata	1189	Air	80	43246	Early	Weather	3	52	
100067	1600	WH_C	13-01-2024	15-03-2024	10-04-2024	Hyderabad	Ahmedabad	1330	Air	18	49484	Delayed	Warehouse Delay	5	26	
100068	3363	WH_C	21-01-2024	09-02-2024	11-07-2024	Delhi	Pune	1241	Road	478	3060	Early	Mechanical Issue	1	153	
100071	3041	WH_D	19-01-2024	06-06-2024	27-06-2024	Bangalore	Ahmedabad	1671	Sea	471	24590	Early	Warehouse Delay	1	21	
100074	3612	WH_C	31-01-2024	11-05-2024	28-05-2024	Chennai	Kolkata	1048	Road	340	3281	Early	Weather	4	17	
100075	3945	WH_B	21-03-2024	24-03-2024	21-05-2024	Bangalore	Ahmedabad	353	Rail	19	17830	On Time	None	2	58	
100077	3139	WH_D	01-02-2024	02-04-2024	09-06-2024	Delhi	Kolkata	465	Air	116	35456	Early	Warehouse Delay	1	68	
100079	4003	WH_D	10-01-2024	05-02-2024	12-04-2024	Hyderabad	Kolkata	245	Air	74	16115	Delayed	Warehouse Delay	2	67	

Key Performance Indicators (KPIs)

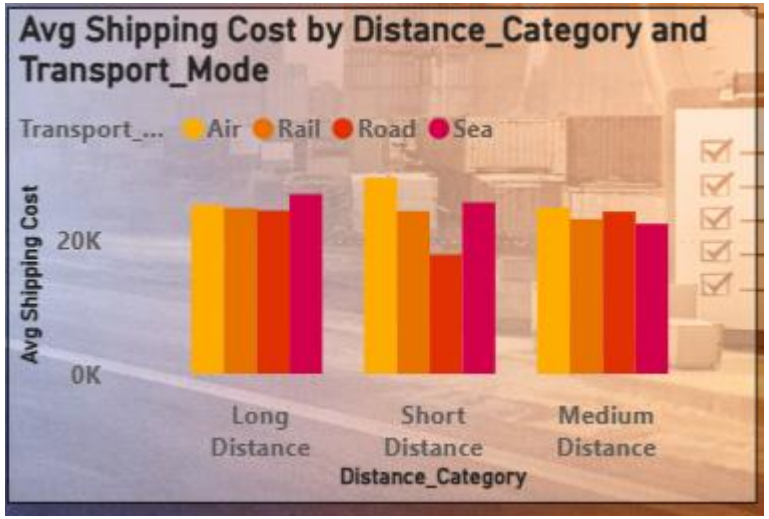
- This section displays the **key metrics** that provide a high-level overview of logistics performance.
- **Average Delivery Time** shows the average number of days taken to deliver orders, helping assess overall delivery efficiency.
- **Total Orders** represents the total number of shipments processed in the dataset, indicating business volume.
- **Delay Percentage** indicates the proportion of delayed deliveries compared to total orders, highlighting service reliability.
- These KPIs allow stakeholders to quickly understand operational performance before exploring detailed visual analysis.

Average Delivery Time by Destination City

- This bar chart shows the **average delivery time** for orders delivered to different destination cities. It helps compare delivery efficiency across locations.
- From the analysis:
- **Kolkata** and **Coimbatore** have the highest average delivery times, indicating possible logistical challenges.
- **Ahmedabad** shows the lowest average delivery time, suggesting more efficient delivery operations.
- Cities like **Pune** and **Jaipur** fall in the mid-range.
- This visualization helps identify cities where delivery optimization efforts should be prioritized.

Average Shipping Cost by Distance Category and Transport Mode

- This clustered column chart compares the **average shipping cost** across different **distance categories** (Short, Medium, Long) and **transport modes** (Air, Rail, Road, Sea).
- Key observations:
- **Air transport** has the highest shipping cost across all distance categories, especially for long distances.
- **Road and Rail** are more cost-effective for short and medium distances.
- **Sea transport** becomes comparatively economical for long-distance shipments.
- This visualization helps in identifying the most **cost-efficient transport mode** based on shipment distance, supporting logistics cost optimization.

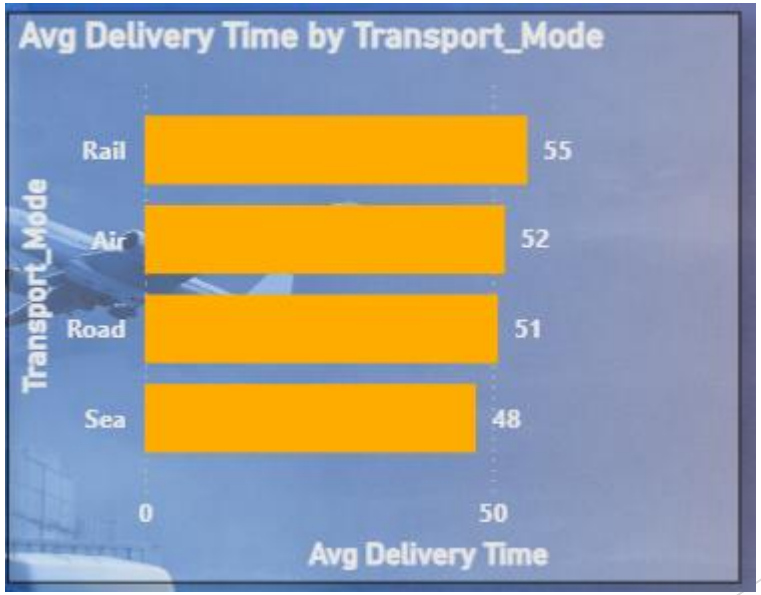


Number of Orders by Delay Reason

- This donut chart represents the **distribution of delayed orders based on different delay reasons**. It helps identify the most common causes of delivery delays.
- Key insights from the chart:
- **Mechanical issues** and **warehouse delays** contribute significantly to delayed orders.
- **Traffic** and **weather conditions** also have a noticeable impact on delivery delays.
- Orders marked as **“None”** indicate shipments that were not delayed.
- This analysis helps logistics teams focus on operational areas that require improvement to reduce delivery delays.

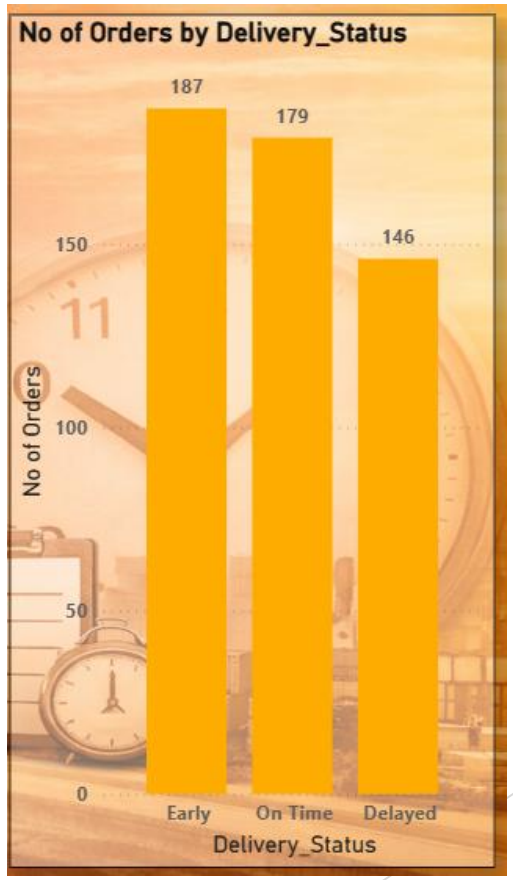
Average Delivery Time by Transport Mode

- This bar chart compares the **average delivery time** across different **transport modes** used for shipments.
- Key insights:
- **Rail transport** has the highest average delivery time, indicating slower transit or handling processes.
- **Air transport** performs faster than rail but is still not the quickest option.
- **Road transport** shows relatively better delivery speed.
- **Sea transport** has the lowest average delivery time among the modes shown.
- This visualization helps identify the most time-efficient transport modes for logistics planning.



Number of Orders by Delivery Status

- This column chart shows the **distribution of orders based on delivery status**, categorized as **Early**, **On Time**, and **Delayed**.
- Key observations:
- A large number of orders are delivered **Early** and **On Time**, indicating generally good logistics performance.
- **Delayed deliveries** are comparatively lower but still significant and require attention.
- The chart helps assess overall service reliability and punctuality.
- This visualization provides a clear understanding of how well deliveries are meeting expected timelines.



Interactive Filters (Slicers)

- This section includes **interactive slicers** that allow users to dynamically filter the dashboard and analyze logistics performance from different perspectives.
- **Warehouse ID slicer** helps compare performance across different warehouses.
- **Delivery Status slicer** allows analysis of early, on-time, or delayed orders.
- **Transport Mode slicer** enables comparison between different shipment methods such as air, road, rail, and sea.
- These slicers improve interactivity and allow stakeholders to explore insights based on specific operational conditions.

Conclusion

•Identified delivery delays caused by **inefficient transport modes, warehouse delays, traffic, and mechanical issues**.

•Observed **variation in delivery time across cities and warehouses**, indicating inconsistent logistics performance.

•Found that **shipping cost and delivery time increase with distance**, especially for air transport.

•Recommended **transport mode optimization**, **warehouse process improvements**, and **proactive delay management**.

•Enabled **data-driven decision-making** by converting raw logistics data into actionable insights using Python and Power BI.