SAMSUNG INNOVATION CAMPUS

BIG DATA

Capstone Project-1

Logistics & Delivery Performance Analysis

TEAM MEMBERS

BD702 - Group 8

Menna Mahmoud

Malak Osama

Menna Tarek

Table of Contents

Introduction

Today, companies depend a lot on data to improve how they work, especially in logistics and delivery. With the huge increase in online shopping and global supply chains, problems like late deliveries, high shipping costs and warehouse inefficiencies are becoming very common. Solving these problems need more than just traditional methods. They need the power of big data.

This project, “Logistics & Delivery Performance Analysis”, is about using big data tools to study logistics operations and discover useful insights. The dataset we used comes from the Smart Logistics Supply Chain dataset on Kaggle, which includes information about orders, warehouses, shipping costs and delivery times. By analyzing this data, our goal is to understand delivery performance better, detect late deliveries, see how warehouses are performing and spot where most of the costs are coming from.

The work was done step by step:
Data Extraction – bringing the raw data into the Hadoop ecosystem.
Data Preprocessing and Transformation – cleaning and transforming the data with PySpark, including calculating delivery time differences.
Data Loading and Integration – storing the cleaned data in Hive tables and connecting them with Power BI through ODBC.
Data Visualization – building an interactive Power BI dashboard that shows KPIs and deeper warehouse-level insights.

For the tools, we used HDFS for storage, Apache Spark for processing and query, Power BI for creating the dashboard. Together, these tools made it possible to handle large data, process it efficiently and then turn it into visual insights that are easy to understand.

The final result is a dashboard that not only shows the big picture (like total orders, average delivery time and average shipping cost) but also digs deeper into warehouse performance, late deliveries and shipping cost distribution. This way, the project doesn’t just stop at handling data—it turns it into something useful that can actually guide decisions and improve operations.

Data Source and structure

Logistics Supply chain real world data

The primary dataset used for this project originates from real-world logistics and supply chain operations. For the purpose of analysis, the dataset has been divided into two main components:

1. Logistics Information: Contains details related to shipments, delivery tracking, warehouse handling, and transit times.
2. Order Information: Contains details related to individual orders, including order IDs, order_date, category_name, ,department_name, order_item quantity, expected_delivery_date.

There was not a Warehouse_ID in the data so we assigned each country a warehouse using “string_indexer” and dropped the columns not related to the project goal

```
logistics_df.show()

+---------+-------------+-----------+--------------+-----------+----------------+--------------+---------------+-------------+--------------------+
| order_id|shipping_cost|category_id| shipping_mode|customer_id|customer_segment|customer_state|   order_status|order_country|actual_delivery_date|
+---------+-------------+-----------+--------------+-----------+----------------+--------------+---------------+-------------+--------------------+
|15081.289|     84.99157|        9.0|Standard Class|  12097.683|        Consumer|            PR|       COMPLETE|      Austria|          2015-08-17|
|56444.684|       181.99|       48.0|Standard Class|  5108.1045|        Consumer|            CA|        PENDING|    Argentina|          2017-02-15|
|7508.5713|     93.81015|       46.0|  Second Class|  4293.4478|        Consumer|            PR|       COMPLETE|       France|          2015-01-05|
|56196.926|      99.8906|       17.0|  Second Class|   546.5306|        Consumer|            PR|     PROCESSING|  El Salvador|          2017-06-05|
|5565.5796|    171.07587|       48.0|Standard Class|   1546.398|        Consumer|            CA|       COMPLETE|       Mexico|          2015-04-01|
|32955.824|    145.46329|       17.0|Standard Class|  5048.3975|        Consumer|            PR|         CLOSED|United States|          2016-06-11|
|35385.855|       167.99|       46.0|Standard Class|  7413.2383|       Corporate|            PR|       COMPLETE|United States|          2016-05-21|
|  36338.4|       116.99|       18.0|Standard Class|  6775.2695|     Home Office|            PR|     PROCESSING|United States|          2016-06-16|
|27692.854|    113.15623|       18.0|Standard Class|  4784.4346|        Consumer|            KY|        ON_HOLD|     Thailand|          2016-06-10|
|56918.418|       127.39|       29.0|   First Class|  11880.099|       Corporate|            PR|PENDING_PAYMENT|       Mexico|          2017-09-03|
| 67071.39|   111.575935|       18.0|Standard Class|  3338.5322|       Corporate|            PR|        ON_HOLD|       France|          2017-09-03|
| 58629.56|       299.98|       45.0|  Second Class|  1701.7117|        Consumer|            NJ|PENDING_PAYMENT|       Brazil|          2017-04-21|
|15836.077|    290.95166|       17.0|  Second Class|  1256.4532|     Home Office|            AZ|       COMPLETE|      Germany|          2015-09-16|
|45317.816|        189.0|       48.0|Standard Class|   8478.369|       Corporate|            PR|         CLOSED|      Nigeria|          2016-10-04|
```

```
order_df.show(10)

+---------+--------------------+--------------------+-------------+---------------+-------------------+------------+----------------------+
| order_id|          order_date|       category_name|order_country|department_name|order_item_quantity|warehouse_id|expected_delivery_date|
+---------+--------------------+--------------------+-------------+---------------+-------------------+------------+----------------------+
|15081.289|2015-08-12 00:00:...|    Cardio Equipment|      Austria|       Footwear|                1.0|        29.0|            2015-08-17|
|56444.684|2017-02-10 00:00:...|        Water Sports|    Argentina|       Fan Shop|                1.0|        25.0|            2017-02-15|
|7508.5713|2015-01-01 00:00:...|Indoor/Outdoor Games|       France|       Fan Shop|                2.0|         2.0|            2015-01-06|
|56196.926|2017-05-31 00:00:...|              Cleats|  El Salvador|        Apparel|                2.0|        14.0|            2017-06-05|
|5565.5796|2015-03-28 00:00:...|        Water Sports|       Mexico|       Fan Shop|                1.0|         1.0|            2015-04-02|
|32955.824|2016-06-06 00:00:...|         Electronics|United States|       Footwear|                4.0|         0.0|            2016-06-11|
|35385.855|2016-05-17 00:00:...|Indoor/Outdoor Games|United States|       Fan Shop|                4.0|         0.0|            2016-05-22|
|  36338.4|2016-06-09 00:00:...|       Men's Footwear|United States|        Apparel|                1.0|         0.0|            2016-06-14|
|27692.854|2016-06-06 00:00:...|       Men's Footwear|     Thailand|        Apparel|                1.0|        21.0|            2016-06-11|
|56918.418|2017-08-29 00:00:...|        Shop By Sport|       Mexico|           Golf|                4.0|         1.0|            2017-09-03|
+---------+--------------------+--------------------+-------------+---------------+-------------------+------------+----------------------+
only showing top 10 rows
```

Data Extraction

To simulate a real-world big data project, we first placed our dataset on HDFS (Hadoop Distributed File System), which is commonly used for storing and accessing large volumes of data.

Step 1: Uploading Data to HDFS

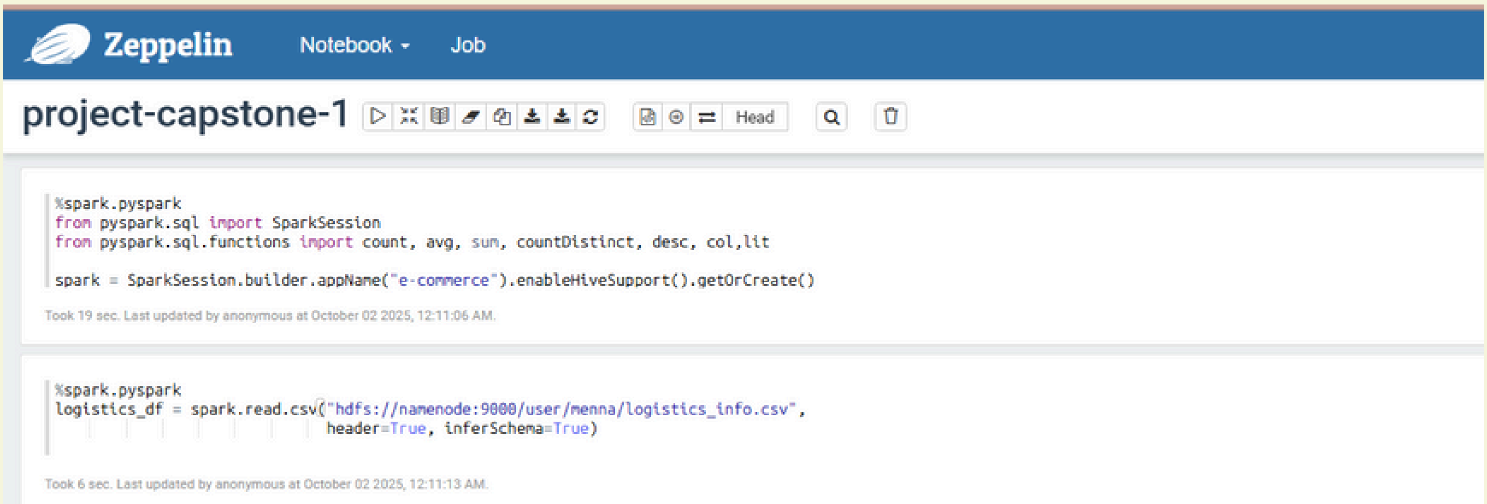
We created a dedicated directory in HDFS for our user and uploaded the logistics dataset:

hdfs dfs -mkdir -p /user/menna/

hdfs dfs -put /home/menna/projects/Big-Data-Cluster/logistics_info.csv /user/menna/

Step 2: Reading Data from HDFS in PySpark

After uploading, the dataset was read directly from HDFS into a PySpark DataFrame for further processing:



Data Transformation

After loading the datasets into PySpark, several transformations were applied to prepare the data for analysis and KPI computation:

1. **Joining Datasets**
   - The Logistics Information and Order Information datasets were joined on the order_id column.
   - This join allows integration of shipment and order-level details, making it possible to calculate metrics such as delivery performance at the order level

2. Calculating Delivery Time Delta

- A new column, delivery_time_delta_days, was derived using the Spark datediff function:
- This represents the number of days by which an order was delivered early or late.

3. Deriving Key Performance Indicator (KPI)

- A Logistics KPI, is_late_delivery, was created as a performance flag to indicate whether an order was delivered late:
- is_late_delivery = delivery_time_delta_days > 0
- If delivery_time_delta_days is greater than 0, the delivery is considered late; otherwise, it is on time or early.

```
# Calculate delivery delta and late delivery
joined_df = joined_df.withColumn(
    "delivery_time_delta_days",
    F.datediff(F.col("actual_delivery_date"), F.col("expected_delivery_date"))
).withColumn(
    "is_late_delivery",
    F.when(F.col("delivery_time_delta_days") > 0, True).otherwise(False)
)
```

Took 0 sec. Last updated by anonymous at October 02 2025, 12:11:49 AM.

```
%spark.pyspark
# Select the columns that exist in joined_df
Staging_Logistics_Fact = joined_df.select(
    "order_id",
    "delivery_time_delta_days",
    "is_late_delivery",
    "shipping_cost",
    "warehouse_id"
)

Staging_Logistics_Fact.show()
+---------+------------------------+----------------+-------------+------------+
| order_id|delivery_time_delta_days|is_late_delivery|shipping_cost|warehouse_id|
+---------+------------------------+----------------+-------------+------------+
|15081.289|                       0|           false|     84.99157|        29.0|
|56444.684|                       0|           false|       181.99|        25.0|
|7508.5713|                      -1|           false|     93.81015|         2.0|
|56196.926|                       0|           false|      99.8906|        14.0|
|5565.5796|                      -1|           false|    171.07587|         1.0|
|32955.824|                       0|           false|    145.46329|         0.0|
|35385.855|                      -1|           false|       167.99|         0.0|
|  36338.4|                       2|            true|       116.99|         0.0|
|27692.854|                      -1|           false|    113.15623|        21.0|
|56918.418|                       0|           false|       127.39|         1.0|
| 67071.39|                       0|           false|   111.575935|         2.0|
| 58629.56|                      -1|           false|       299.98|         5.0|
|15836.077|                       2|            true|    290.95166|         4.0|
|45317.816|                      -1|           false|        189.0|        20.0|
|39779.484|                      -1|           false|       123.49|         0.0|
```

Took 1 sec. Last updated by anonymous at October 02 2025, 12:11:52 AM.

4. Creating the Final Output Table: Staging_Logistics_Fact

After performing the initial joins and calculating the delivery metrics, we prepared the final output table Staging_Logistics_Fact to consolidate the key attributes needed for analysis and reporting.

We selected only the relevant columns from the joined DataFrame to create a concise table.
The table captures

- Operational metrics (delivery time delta, late delivery flag) a
- Financial/logistics metrics (shipping cost, warehouse_id) for downstream analysis.

```
%spark.pyspark
# Select the columns that exist in joined_df
Staging_Logistics_Fact = joined_df.select(
    "order_id",
    "delivery_time_delta_days",
    "is_late_delivery",
    "shipping_cost",
    "warehouse_id"
)

Staging_Logistics_Fact.show()
```

order_id	delivery_time_delta_days	is_late_delivery	shipping_cost	warehouse_id
15081.289	0	false	84.99157	29.0
56444.684	0	false	181.99	25.0
7508.5713	-1	false	93.81015	2.0
56196.926	0	false	99.8906	14.0
5565.5796	-1	false	171.07587	1.0
32955.824	0	false	145.46329	0.0
35385.855	-1	false	167.99	0.0
36338.4	2	true	116.99	0.0
27692.854	-1	false	113.15623	21.0
56918.418	0	false	127.39	1.0
67071.39	0	false	111.575935	2.0
58629.56	-1	false	299.98	5.0
15836.077	2	true	290.95166	4.0
45317.816	-1	false	189.0	20.0

Took 1 sec. Last updated by anonymous at October 02 2025, 12:11:52 AM.

5. Calculating Average Delivery Time Delta per Warehouse

To evaluate the performance of each warehouse, we calculated the average delivery time delta using Spark SQL on the Staging_Logistics_Fact table. This metric provides insight into how early or late orders are delivered on average for each warehouse.
Steps Performed:

1. Registering the Fact Table as a Temp View
   - The Staging_Logistics_Fact DataFrame was registered as a temporary SQL view to enable Spark SQL queries:
2. Computing the Average Delivery Time Delta
   - Using Spark SQL, we calculated the average of delivery_time_delta_days grouped by warehouse_id:

```
%spark.pyspark
Staging_Logistics_Fact.createOrReplaceTempView("staging_logistics_fact")

avgDeliveryTimeDelta = spark.sql("""
    SELECT
        warehouse_id,
        AVG(delivery_time_delta_days) AS avg_delivery_time_delta
    FROM staging_logistics_fact
    GROUP BY warehouse_id
""")
avgDeliveryTimeDelta.show()
```

warehouse_id	avg_delivery_time_delta
147.0	-1.0
8.0	0.06310679611650485

warehouse_id	avg_delivery_time_delta
147.0	-1.0
8.0	0.06310679611650485
70.0	0.058823529411764705
67.0	0.21052631578947367
0.0	0.07115198451113262
69.0	-0.16666666666666666
7.0	0.09925093632958802
142.0	2.0
112.0	-0.3333333333333333
124.0	1.0
128.0	-0.5
108.0	0.0
133.0	1.0
88.0	0.1111111111111111

Data Load and Integration

After completing all transformations, the processed data was loaded into HDFS and then integrated into Hive, simulating a real-world big data pipeline.

Step 1: Create HDFS Directory

A dedicated directory was created in HDFS to store the processed datasets

```
menna@DESKTOP-3PNO10O:~/projects/Big-Data-Cluster$ docker exec -it namenode bash
root@126a36e89378:/# hdfs dfs -ls /user/bigdata
Found 2 items
drwxr-xr-x   - root supergroup          0 2025-10-01 21:17 /user/bigdata/avgDeliveryTimeDelta
drwxr-xr-x   - root supergroup          0 2025-10-01 21:12 /user/bigdata/staging_logistics_fact
```

Step 2: Write Processed Data to HDFS

The final tables, Staging_Logistics_Fact_fixed and avgDeliveryTimeDelta, were saved in Parquet format:

```
%spark.pyspark
from pyspark.sql.functions import col

Staging_Logistics_Fact_fixed = Staging_Logistics_Fact.select(
    col("order_id").cast("int"),
    col("delivery_time_delta_days").cast("int"),
    col("is_late_delivery").cast("boolean"),
    col("shipping_cost").cast("float"),
    col("warehouse_id").cast("int")
)

Staging_Logistics_Fact_fixed.write.mode("overwrite").parquet("hdfs://namenode:9000/user/bigdata/staging_logistics_fact")
```

Took 1 sec. Last updated by anonymous at October 02 2025, 12:12:09 AM.

```
%spark.pyspark
avgDeliveryTimeDelta.write.mode("overwrite").parquet("hdfs://namenode:9000/user/bigdata/avgDeliveryTimeDelta")
```

Took 0 sec. Last updated by anonymous at October 02 2025, 12:17:01 AM.

Hadoop | Overview | Datanodes | Datanode Volume Failures | Snapshot | Startup Progress | Utilities

Browse Directory

//user/bigdata/staging_logistics_fact Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	0 B	Oct 01 20:40	3	128 MB	_SUCCESS
-rw-r--r--	root	supergroup	223.35 KB	Oct 01 20:40	3	128 MB	part-00000-c0018834-cd9e-4fe5-a06a-18b5793b3e1c-c000.snappy.parquet

Showing 1 to 2 of 2 entries

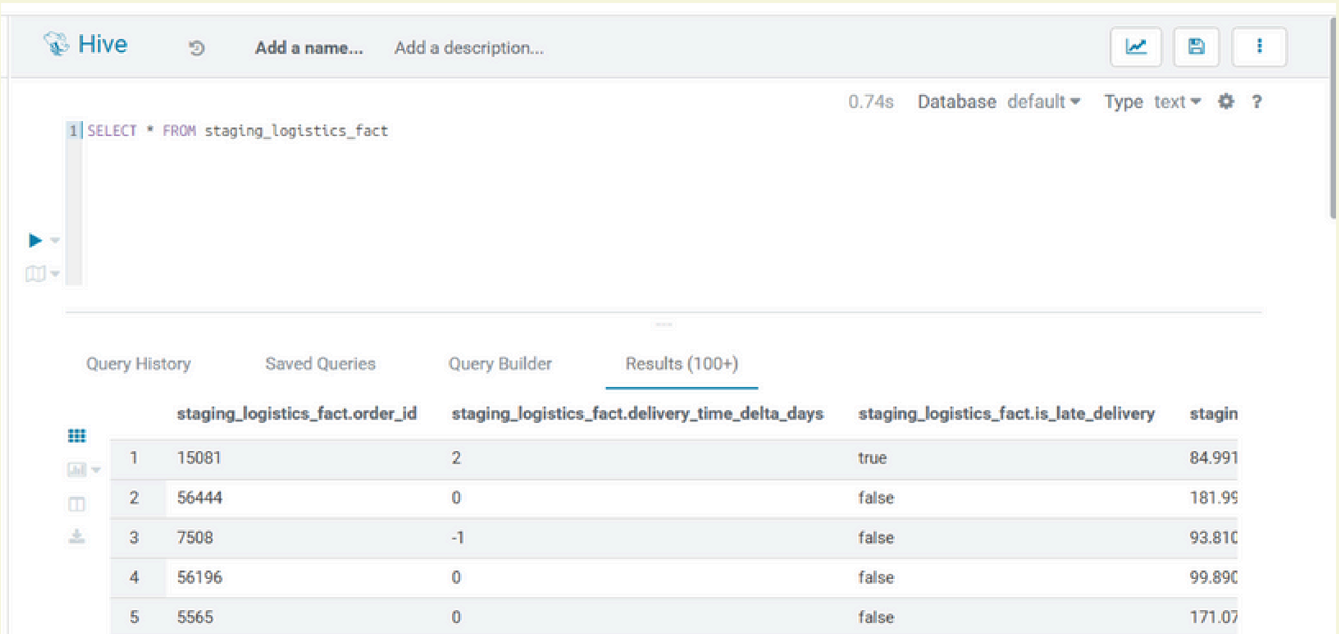
Previous 1 Next

Step 3: Create Hive Table Schema

Before loading the data into Hive, a table schema is defined to specify the structure of the table and data types

```
menna@DESKTOP-3PNO10O:~/projects/Big-Data-Cluster$ docker exec -it hive-server bash
root@b35050373558:/# beeline -u "jdbc:hive2://localhost:10000/default"
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://localhost:10000/default
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.2 by Apache Hive
0: jdbc:hive2://localhost:10000/default>
0: jdbc:hive2://localhost:10000/default> CREATE EXTERNAL TABLE staging_logistics_fact (
. . . . . . . . . . . . . . . . . . . .>   order_id STRING,
. . . . . . . . . . . . . . . . . . . .>   delivery_time_delta_days INT,
. . . . . . . . . . . . . . . . . . . .>   is_late_delivery BOOLEAN,
. . . . . . . . . . . . . . . . . . . .>   shipping_cost DOUBLE,
. . . . . . . . . . . . . . . . . . . .>   warehouse_id STRING
. . . . . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . . . . .> STORED AS PARQUET
. . . . . . . . . . . . . . . . . . . .> LOCATION 'hdfs:///user/bigdata/staging_logistics_fact';
No rows affected (1.449 seconds)
0: jdbc:hive2://localhost:10000/default> SHOW TABLES;
+-------------------------+
|        tab_name         |
+-------------------------+
| staging_logistics_fact  |
```

```
0: jdbc:hive2://localhost:10000/default> CREATE TABLE avgDeliveryTimeDelta(
. . . . . . . . . . . . . . . . . . . .>   warehouse_id INT,
. . . . . . . . . . . . . . . . . . . .>   avg_delivery_time_delta INT
. . . . . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . . . . .> STORED AS PARQUET
. . . . . . . . . . . . . . . . . . . .> LOCATION 'hdfs://namenode:9000/user/bigdata/avgDeliveryTimeDelta';
No rows affected (1.472 seconds)
```



After schema creation, the Hive table points to the Parquet files in HDFS, making the data queryable via SQL.
This step integrates the processed datasets into the Hive ecosystem for analysis, reporting, and visualization.

Overall, this workflow demonstrates a complete ETL process for big data

Data Visualization

To analyze and interpret the logistics dataset effectively, an interactive dashboard was designed.
The dashboard consists of three rows, each providing insights into different aspects of orders, delivery performance, warehouses, and shipping costs.

First Row: Key Performance Indicators (KPIs):
This row contains three cards that summarize the overall dataset with essential metrics:
Total Orders: Displays the total number of orders processed in the dataset.
Average Delivery Time Delta (Days): Represents the mean delivery time difference between scheduled and actual delivery, expressed in days.
Average Shipping Cost: Shows the average shipping cost across all orders.
This row provides a quick overview of the dataset's scale, efficiency, and cost profile.

Second Row: Delivery Performance by Warehouse:
This row highlights warehouse-level performance, particularly focusing on delivery timeliness.

Left – Clustered Column Chart (Top 10 Warehouses by Late Deliveries)

X-axis: Warehouse

Y-axis: Count of late deliveries

Filter: Top 10 warehouses

Insight: Identifies which warehouses are most responsible for late deliveries.

Middle – Clustered Column Chart (Orders by Warehouse and Late Delivery Status)

X-axis: Warehouse

Y-axis: Percentage of orders

Legend: Late Delivery (True/False)

Filter: Top 10 warehouses

Insight: Compares the proportion of on-time vs. late orders per warehouse.

Right – Doughnut Chart (Orders by Late Delivery Status and Warehouse)

Legend: Late Delivery (True/False)

Values: Count of orders

Filter: Top 10 warehouses

Insight: Shows the overall split between late and on-time deliveries by warehouse.

Third Row: Shipping Costs and Geographic Distribution

This row focuses on shipping costs and the geographic spread of warehouse activity.

- **Left – Stacked Column Chart (Top 10 Orders by Shipping Cost)**
  X-axis: Order ID (Top 10 by cost)
  Y-axis: Sum of shipping cost
  Legend: Order components (stacked)
  Insight: Highlights the orders contributing the most to overall shipping expenses.

- **Middle – Map Visualization (Top 10 Warehouses by Order Count)**
  Location: Warehouse coordinates
  Values: Number of orders
  Filter: Top 10 warehouses
  Insight: Provides a spatial perspective of warehouse activity distribution.

- **Right – Pie Chart (Top 10 Warehouses by Shipping Cost)**
  Legend: Warehouse
  Values: Sum of shipping cost
  Filter: Top 10 warehouses
  Insight: Displays which warehouses incur the highest total shipping expenses.

Tools Used

Tool	Usage in Project
Hadoop (HDFS)	Used to store raw and processed datasets; enabled scalable, fault-tolerant data storage.
Apache Spark	Used for data extraction, transformation, and aggregation; supports PySpark and Spark SQL for analytics.
Hive	Used to Store the parqut files to be connected to the analysis tool.
ODBC (Open Database Connectivity)	Provided connectivity between Hive and Power BI.
Power BI	Used to visualize and analyze the processed data; the dashboard is created through it