LOGISTICS PROJECT

TEAM NAMES

- NADA MAHMOUD
- MOHAMED HANY
- SAEED ELGHRABWY

PROJECT OVERVIEW

PROJECT: E-COMMERCE LOGISTICS & SALES

SCOPE: Analysis of a synthetic dataset containing 1,000+ simulated orders and their delivery logs.

OBJECTIVE: To demonstrate a scalable Big Data pipeline and the type of actionable business intelligence it can generate.

TECHNOLOGY: POSTGRESQL, Apache Sqoop, Apache Spark, HDFS

STRUCTURE OUR SOURCE DATA

```
saeed@DESKTOP-HG4UOKB:~$ cd "/mnt/d/cources/Samsung Big Data/projects/project_1"
saeed@DESKTOP-HG4UOKB:/mnt/d/cources/Samsung Big Data/projects/project_1$ docker cp Staging_OrderLines.csv external_postgres_db:/root/
saeed@DESKTOP-HG4UOKB:/mnt/d/cources/Samsung Big Data/projects/project_1$ docker cp Logistics.csv external_postgres_db:/root/
Successfully copied 40.4kB to external_postgres_db:/root/
saeed@DESKTOP-HG4UOKB:/mnt/d/cources/Samsung Big Data/projects/project_1$ docker cp Staging_OrderLines.csv external_postgres_db:/root/
Successfully copied 50.7kB to external_postgres_db:/root/
saeed@DESKTOP-HG4UOKB:/mnt/d/cources/Samsung Big Data/projects/project_1$
```

```
root@197ebbd66afe:/# hdfs dfs -mkdir -p /project1/input
root@197ebbd66afe:/# hdfs dfs -put /root/Staging_OrderLines.csv /project1/input/
hdfs dfs -put /root/Logistics.csv /project1/input/2025-09-30 15:52:32,025 INFO sasl.SaslDataTransferClient: SASL encryption trust check: lo

root@197ebbd66afe:/# hdfs dfs -put /root/Logistics.csv /project1/input/
2025-09-30 15:52:35,600 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
root@197ebbd66afe:/# hdfs dfs -ls /project1/input
Found 2 items
-rw-r--r--   3 root supergroup      38519 2025-09-30 15:52 /project1/input/Logistics.csv
-rw-r--r--   3 root supergroup      49022 2025-09-30 15:52 /project1/input/Staging_OrderLines.csv
root@197ebbd66afe:/#
```

COPYING DATA FROM LOCAL TO DOCKER CONTAINER

STRUCTURE OUR SOURCE DATA IN A POSTGRESQL DATABASE

```
nada_mahmoud467@DESKTOP-F1F883M:~/Big-Data-Cluster$ docker exec -it external_postgres_db bash
14094d109c13:/# psql -U external -d external
psql (15.1)
Type "help" for help.

external=#  CREATE TABLE logistics (
external(# _id VARCHAexternal(#   logistics_id VARCHAR(50),
external(# external(#   order_id VARCHAR(50),
external(# external(#   estimated_delivery_date DATE,
external(# external(#   actual_delivery_date DATE,
external(# external(#   shipping_cost NUMERIC,
al(#  external(# external(#   warehouse_id VARCHAR(50)
external(# external(# );
external(#
external(# CREATE TABLE staging_order_lines (
external(#   order_id VARCHAR(50),
external(#   product_id VARCHAR(50),
external(#   quantity INTEGER,
external(#   price NUMERIC
```

We created two tables to hold our raw data

defining schemas that match our CSV files.

STRUCTURE OUR SOURCE DATA IN A POSTGRESQL DATABASE

```
external=# \copy staging_orderlines FROM '/root/Staging_OrderLines.csv' CSV
COPY 2970
external=# \copy logistics FROM '/root/Logistics.csv' CSV HEADER;
COPY 1000
```

```
external=# select * from staging_orderlines limit 10;
 order_id | product_id | quantity | price
----------+------------+----------+--------
        1 |         28 |        8 | 196.06
        1 |         39 |        5 | 235.11
        1 |         35 |        9 | 489.75
        1 |         50 |        9 | 285.49
        2 |         26 |        7 |  20.21
        2 |         24 |        9 |  99.54
        2 |         20 |        8 | 132.81
        3 |         17 |        3 | 327.08
        3 |         10 |        9 | 131.71
        3 |         38 |        5 | 453.02
(10 rows)

external=# select * from logistics limit 10;
 logistics_id | order_id | estimated_delivery_date | actual_delivery_date | shipping_cost | warehouse_id
--------------+----------+-------------------------+----------------------+---------------+--------------
            1 |        1 | 2025-10-08              | 2025-10-12           |         29.69 | WH1
            2 |        2 | 2025-10-24              | 2025-10-23           |         22.50 | WH2
            3 |        3 | 2025-09-23              | 2025-09-26           |         31.22 | WH4
            4 |        4 | 2025-07-31              | 2025-07-31           |         33.17 | WH3
            5 |        5 | 2025-10-01              | 2025-10-06           |         23.84 | WH3
            6 |        6 | 2025-10-26              | 2025-10-26           |         48.00 | WH4
            7 |        7 | 2025-09-05              | 2025-09-05           |         48.16 | WH3
            8 |        8 | 2025-10-05              | 2025-10-06           |         50.82 | WH4
            9 |        9 | 2025-10-09              | 2025-10-10           |         13.13 | WH2
```

Check data is transformed into tables

DATA INGESTION WITH APACHE SQOOP

```
root@425f2f86a607:/# sqoop import \
>   --connect jdbc:postgresql://external_postgres_db:5432/external \
>   --username external \
>   --password external \
>   --table logistics \
>   --target-dir /staging_zone/logistics \
>   -m 1 \
```

```
root@425f2f86a607:/# sqoop import \
>   --connect jdbc:postgresql://external_postgres_db:5432/external \
>   --username external \
>   --password external \
 --table>   --table staging_order_lines \
>   --target-dir /staging_zone/staging_order_lines \
>   -m 1 \
```

Transfer our source data from the relational database into the Hadoop Distributed File System (HDFS)

TRANSFORMATION USING APACHE ZEPPELIN

```
%spark.pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.functions import col, to_date, datediff, when
from pyspark.sql.functions import sum as _sum, round, countDistinct

spark = SparkSession.builder.appName("ETL").enableHiveSupport().getOrCreate()
```

Took 0 sec. Last updated by anonymous at October 01 2025, 2:01:30 AM.

```
%spark.pyspark

# Load data and assign column names

logistics_df = spark.read.csv("/staging_zone/logistics", header=False, inferSchema=True) \
    .toDF("logistics_id", "order_id", "estimated_delivery_date", "actual_delivery_date", "shipping_cost", "warehouse_id")

order_lines = spark.read.csv("/staging_zone/staging_orderlines", header=False, inferSchema=True) \
    .toDF("order_id", "product_id", "quantity", "price")
```

Took 1 sec. Last updated by anonymous at October 01 2025, 2:01:31 AM. (outdated)

LOAD DATA AND ASSIGN COLUMN NAMES

TRANSFORMATION USING APACHE ZEPPELIN

```
%spark.pyspark

# Clean and transform logistics dates:

logistics_df = logistics_df.filter(col("estimated_delivery_date").isNotNull() & col("actual_delivery_date").isNotNull()) \
    .withColumn("estimated_delivery_date", to_date(col("estimated_delivery_date"), "yyyy-MM-dd")) \
    .withColumn("actual_delivery_date", to_date(col("actual_delivery_date"), "yyyy-MM-dd"))
```

Took 0 sec. Last updated by anonymous at October 01 2025, 2:01:31 AM. (outdated)

CLEAN AND TRANSFORM LOGISTICS DATA

TRANSFORMATION USING APACHE ZEPPELIN

```
%spark.pyspark

# Create logistics fact table:

logistics_fact = logistics_df \
    .withColumn("delivery_time_delta_days", datediff(col("actual_delivery_date") ,col("estimated_delivery_date"))) \
    .withColumn("is_late_delivery", when(col("delivery_time_delta_days") > 0, 1).otherwise(0)) \
    .select("order_id", "delivery_time_delta_days", "is_late_delivery", "shipping_cost", "warehouse_id")

logistics_fact.show(10)
```

```
+--------+-----------------------+----------------+-------------+------------+
|order_id|delivery_time_delta_days|is_late_delivery|shipping_cost|warehouse_id|
+--------+-----------------------+----------------+-------------+------------+
|       1|                      4|               1|        29.69|         WH1|
|       2|                     -1|               0|         22.5|         WH2|
|       3|                      3|               1|        31.22|         WH4|
|       4|                      0|               0|        33.17|         WH3|
|       5|                      5|               1|        23.84|         WH3|
|       6|                      0|               0|         48.0|         WH4|
|       7|                      0|               0|        48.16|         WH3|
|       8|                      1|               1|        50.82|         WH4|
|       9|                      1|               1|        13.13|         WH2|
|      10|                      2|               1|        20.46|         WH1|
+--------+-----------------------+----------------+-------------+------------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at October 01 2025, 2:01:32 AM. (outdated)

CREATE LOGISTICS FACT TABLE

TRANSFORMATION USING APACHE ZEPPELIN

```
%spark.pyspark

# Join logistics fact with order lines to get revenue

logistics_fact_joined = logistics_fact.join(order_lines, on="order_id", how="inner") \
    .withColumn("total_revenue",round(col("quantity") * col("price"), 2))

logistics_fact_joined.show(10)
```

```
+--------+------------------------+---------------+-------------+------------+----------+--------+------+-------------+
|order_id|delivery_time_delta_days|is_late_delivery|shipping_cost|warehouse_id|product_id|quantity| price|total_revenue|
+--------+------------------------+---------------+-------------+------------+----------+--------+------+-------------+
|       1|                       4|              1|        29.69|         WH1|        28|       8|196.06|      1568.48|
|       1|                       4|              1|        29.69|         WH1|        39|       5|235.11|      1175.55|
|       1|                       4|              1|        29.69|         WH1|        35|       9|489.75|      4407.75|
|       1|                       4|              1|        29.69|         WH1|        50|       9|285.49|      2569.41|
|       2|                      -1|              0|         22.5|         WH2|        26|       7| 20.21|       141.47|
|       2|                      -1|              0|         22.5|         WH2|        24|       9| 99.54|       895.86|
|       2|                      -1|              0|         22.5|         WH2|        20|       8|132.81|      1062.48|
|       3|                       3|              1|        31.22|         WH4|        17|       3|327.08|       981.24|
|       3|                       3|              1|        31.22|         WH4|        10|       9|131.71|      1185.39|
|       3|                       3|              1|        31.22|         WH4|        38|       5|453.02|       2265.1|
+--------+------------------------+---------------+-------------+------------+----------+--------+------+-------------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at October 01 2025, 2:01:33 AM.

JOIN LOGISTICS FACT WITH ORDER LINES TO GET REVENUE

TRANSFORMATION USING APACHE ZEPPELIN

```
%spark.pyspark

# Check if an order is served by multiple warehouses

order_warehouse_check = logistics_fact.groupBy("order_id") \
    .agg(countDistinct("warehouse_id").alias("warehouse_count")) \
    .withColumn("is_multi_warehouse", when(col("warehouse_count") > 1, 1).otherwise(0)) \
    .filter(col("warehouse_count") > 1)

order_warehouse_check.show()
```

```
+--------+---------------+------------------+
|order_id|warehouse_count|is_multi_warehouse|
+--------+---------------+------------------+
+--------+---------------+------------------+
```

Took 1 sec. Last updated by anonymous at October 01 2025, 2:24:31 AM. (outdated)

CHECK IF AN ORDER IS SERVED BY MULTIPLE WAREHOUSES

TRANSFORMATION USING APACHE ZEPPELIN

```
%spark.pyspark

# Aggregate order-level summary:

order_summary = logistics_fact_joined.groupBy("order_id", "warehouse_id", "is_late_delivery", "shipping_cost") \
    .agg(round(sum("total_revenue"), 2).alias("total_revenue_per_order")) \
    .orderBy("order_id")


order_summary.show(10)
```

```
+--------+------------+----------------+-------------+----------------------+
|order_id|warehouse_id|is_late_delivery|shipping_cost|total_revenue_per_order|
+--------+------------+----------------+-------------+----------------------+
|       1|         WH1|               1|        29.69|                9721.19|
|       2|         WH2|               0|         22.5|                2099.81|
|       3|         WH4|               1|        31.22|                4431.73|
|       4|         WH3|               0|        33.17|                1507.86|
|       5|         WH3|               1|        23.84|                5072.76|
|       6|         WH4|               0|         48.0|                 5398.6|
|       7|         WH3|               0|        48.16|                5417.74|
|       8|         WH4|               1|        50.82|                3702.07|
|       9|         WH2|               1|        13.13|                2143.59|
|      10|         WH1|               1|        20.46|                5970.36|
+--------+------------+----------------+-------------+----------------------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at October 01 2025, 2:01:34 AM. (outdated)

AGGREGATE ORDER LEVEL

TRANSFORMATION USING APACHE ZEPPELIN

```
%spark.pyspark

# Aggregate warehouse-level summary:

warehouse_summary = order_summary.groupBy("warehouse_id") \
    .agg(
        round(_sum("total_revenue_per_order"), 2).alias("total_revenue"),
        round(_sum("shipping_cost"), 2).alias("total_shipping_cost"),
        sum("is_late_delivery").alias("late_deliveries"),
        countDistinct("order_id").alias("total_orders")) \
    .withColumn("late_delivery_rate", round((col("late_deliveries") / col("total_orders")) * 100, 2)) \
    .withColumn("avg_shipping_cost_per_order", round(col("total_shipping_cost") / col("total_orders"), 2)) \
    .orderBy("warehouse_id")

warehouse_summary.show(10)
```

```
+------------+-------------+------------------+--------------+------------+------------------+--------------------------+
|warehouse_id|total_revenue|total_shipping_cost|late_deliveries|total_orders|late_delivery_rate|avg_shipping_cost_per_order|
+------------+-------------+------------------+--------------+------------+------------------+--------------------------+
|         WH1|   1094339.13|           7215.54|           193|         262|             73.66|                     27.54|
|         WH2|    923552.71|           6422.67|           160|         221|              72.4|                     29.06|
|         WH3|   1116060.54|           7068.98|           188|         262|             71.76|                     26.98|
|         WH4|    988992.51|           7216.58|           186|         255|             72.94|                      28.3|
+------------+-------------+------------------+--------------+------------+------------------+--------------------------+
```

Took 3 sec. Last updated by anonymous at October 01 2025, 2:03:35 AM. (outdated)

AGGREGATE WAREHOUSE-LEVEL

TRANSFORMATION USING APACHE ZEPPELIN

```
%spark.pyspark

# Warehouse efficiency analysis:calculate ratio of shipping cost to total revenue

warehouse_eff = warehouse_summary.withColumn("shipping_to_revenue_ratio",
            round(col("total_shipping_cost") / col("total_revenue"), 3)) \
            .select("warehouse_id", "shipping_to_revenue_ratio")

warehouse_eff.show()
```

```
+------------+-------------------------+
|warehouse_id|shipping_to_revenue_ratio|
+------------+-------------------------+
|         WH1|                    0.007|
|         WH2|                    0.007|
|         WH3|                    0.006|
|         WH4|                    0.007|
+------------+-------------------------+
```

Took 2 sec. Last updated by anonymous at October 01 2025, 2:09:34 AM. (outdated)

CALCULATE RATIO OF SHIPPING COST TO TOTAL REVENUE

TRANSFORMATION USING APACHE ZEPPELIN

```
%spark.pyspark

# Product-level analysis of late deliveries:

product_delay = logistics_fact_joined.groupBy("product_id") \
    .agg(sum("is_late_delivery").alias("late_deliveries"),
         countDistinct("order_id").alias("total_orders")) \
    .withColumn("late_rate", round(col("late_deliveries") / col("total_orders") * 100, 2)) \
    .orderBy(desc("late_rate"))

product_delay.show(10)
```

```
+----------+--------------+------------+---------+
|product_id|late_deliveries|total_orders|late_rate|
+----------+--------------+------------+---------+
|        41|            49|          55|    89.09|
|        24|            52|          62|    83.87|
|        17|            47|          57|    82.46|
|        29|            50|          61|    81.97|
|        30|            58|          71|    81.69|
|        32|            49|          60|    81.67|
|        13|            52|          64|    81.25|
|        19|            38|          47|    80.85|
|        37|            54|          67|     80.6|
|        39|            44|          55|     80.0|
+----------+--------------+------------+---------+
only showing top 10 rows
```

Took 1 sec. Last updated by anonymous at October 01 2025, 2:10:40 AM. (outdated)

PRODUCT-LEVEL ANALYSIS OF LATE DELIVERIES

INSIGHTS USING SPARK SQL

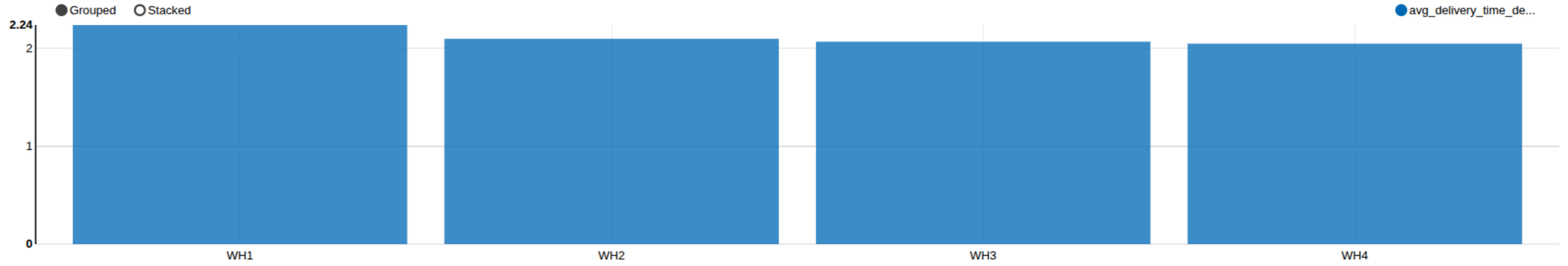
AVERAGE DELIVERY TIME DELTA PER WAREHOUSE

```
%spark.sql
-- Spark SQL Query: Calculate average delivery time delta (days) per warehouse

SELECT warehouse_id,
    ROUND(AVG(delivery_time_delta_days), 2) AS avg_delivery_time_delta
FROM logistics_fact
GROUP BY warehouse_id
ORDER BY warehouse_id;
```

SPARK JOB FINISHED

settings



Took 0 sec. Last updated by anonymous at October 01 2025, 10:40:16 PM. (outdated)

INSIGHTS USING SPARK SQL

TOP WAREHOUSES WITH HIGHEST LATE DELIVERY RATE

```
%spark.sql

-- Top warehouses with highest late delivery rate

SELECT warehouse_id, late_delivery_rate
FROM warehouse_summary
ORDER BY late_delivery_rate DESC;
```

SPARK JOB FINISHED

settings

warehouse_id	late_delivery_rate
WH1	73.66
WH4	72.94
WH2	72.4
WH3	71.76

Took 1 sec. Last updated by anonymous at October 01 2025, 10:41:27 PM. (outdated)

INSIGHTS USING SPARK SQL

TOP 10 PRODUCTS CAUSING DELAYS

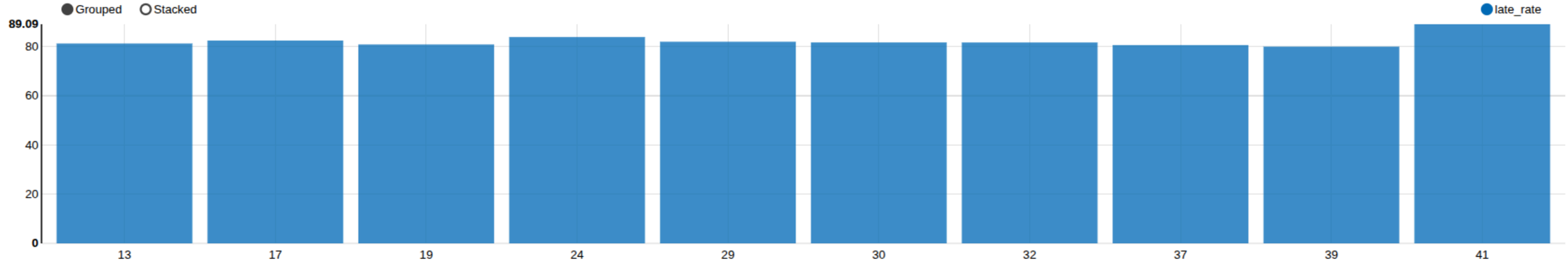
```
%spark.sql

-- Top 10 products causing delays

SELECT product_id, late_rate, late_deliveries, total_orders
FROM product_delay_summary
ORDER BY late_rate DESC
LIMIT 10;
```

SPARK JOB FINISHED

settings



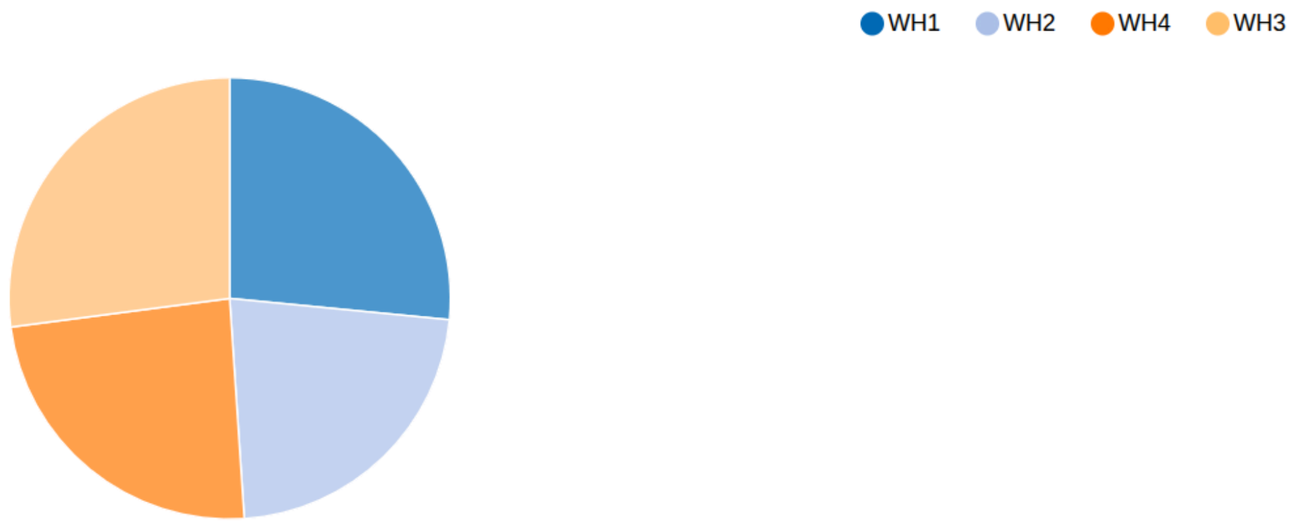
Took 1 sec. Last updated by anonymous at October 01 2025, 10:42:11 PM. (outdated)

INSIGHTS USING SPARK SQL

REVENUE VS SHIPPING COST RATIO PER WAREHOUSE

```
SELECT warehouse_id,
       total_revenue,
       total_shipping_cost,
       ROUND(total_shipping_cost/total_revenue,3) AS shipping_to_revenue_ratio
FROM warehouse_summary
ORDER BY shipping_to_revenue_ratio DESC;
```

settings



INSIGHTS USING SPARK SQL

WAREHOUSE EFFICIENCY VS LATE DELIVERIES

```
SELECT
    warehouse_id,
    COUNT(order_id) AS total_orders,
    SUM(is_late_delivery) AS late_deliveries,
    ROUND(SUM(total_revenue_per_order), 2) AS total_revenue,
    ROUND(SUM(shipping_cost), 2) AS total_shipping_cost,
    ROUND(SUM(total_revenue_per_order)/SUM(shipping_cost), 2) AS revenue_to_shipping_ratio,
    ROUND(SUM(is_late_delivery)/COUNT(order_id)*100, 2) AS late_delivery_rate_percentage
FROM order_summary
GROUP BY warehouse_id
ORDER BY revenue_to_shipping_ratio DESC, late_delivery_rate_percentage ASC;
```

settings

warehouse_id	total_orders	late_deliveries	total_revenue	total_shipping_cost	revenue_to_shipping_ratio	late_delivery_
WH3	262	188	1116060.54	7068.98	157.88	71.76
WH1	262	193	1094339.13	7215.54	151.66	73.66
WH2	221	160	923552.71	6422.67	143.8	72.4
WH4	255	186	988992.51	7216.58	137.04	72.94


THANK
YOU!