Amazon Logistics Optimization

Amazon SQL-Driven Analytics Project

Presented by: Mahesh Katula

PROJECT OVERVIEW

Amazon Logistics

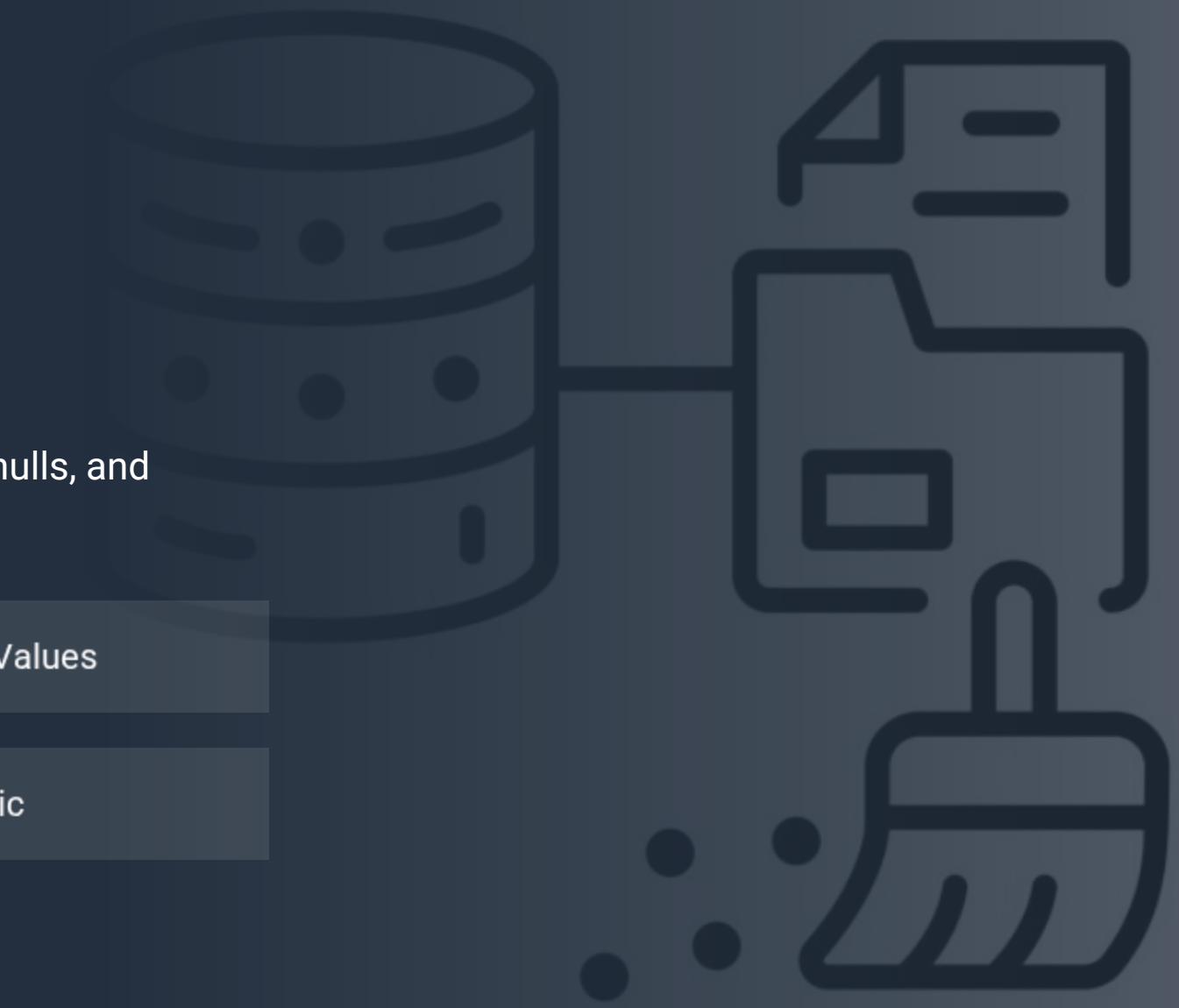
Amazon handles millions of daily shipments. However, increasing order volumes have introduced **delays and route inefficiencies** that impact customer satisfaction.

Key Objectives:

- Identify root causes of delays (Weather, Traffic, Warehouse bottlenecks).
- Optimize delivery routes using efficiency ratios.
- Evaluate the performance of delivery agents and warehouses.

TASK 1

DATA CLEANING & PREPARATION



Objective: Ensure data integrity by handling duplicates, nulls, and format inconsistencies.

- ✓ 1.1 Identify Duplicates
- ✓ 1.2 Handle Null Values
- ✓ 1.3 Standardize Dates
- ✓ 1.4 Validate Logic

IDENTIFY & DELETE DUPLICATES

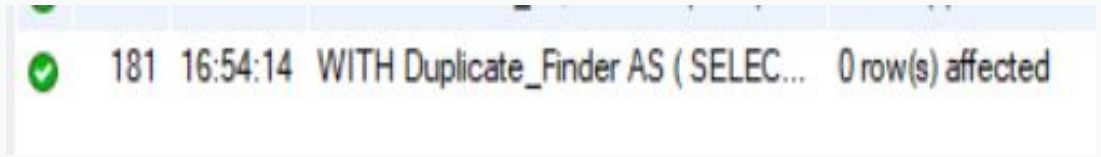
Task 1.1

Created Common Table Expression (CTE) named Duplicate_Finder to find duplicate Order_ID then executed a DELETE statement to remove any rows where row_num > 1.

SQL Query

```sql
-- Identify and delete duplicate Order_ID records.
WITH Duplicate_Finder AS (
SELECT Order_ID, ROW_NUMBER() OVER (
    PARTITION BY Order_ID
    ORDER BY Order_Date -- Keeps the earliest entry if dates differ
) AS row_num FROM Orders
)
-- Deleting rows where row_num is greater than 1
DELETE FROM Orders
WHERE Order_ID IN (
    SELECT Order_ID
    FROM Duplicate_Finder
    WHERE row_num > 1
);
```

Result



0 rows effected means no duplicate values.

HANDLE NULL VALUES

Task 1.2

We found null values in **Traffic_Delay_Min** and replaced them with the average delay for that specific route to maintain data continuity.

SQL Query

```
4     -- Replace null Traffic_Delay_Min with the average delay for that route
5     -- find null values
6     SELECT
7         COUNT(*) AS Null_Count FROM
8         -- Routes -- There is no null values in routes table
9         Warehouses -- Null values present in warehouses table
10    WHERE Traffic_Delay_Min IS NULL;
11
12    SET SQL_SAFE_UPDATES = 0; -- Turn off safe mode
13    Update warehouses as w set Traffic_Delay_Min = (
14        select Avg(r.Traffic_Delay_Min) from Routes as r join Orders as o
15        on o.Route_ID=r.Route_ID where w.Warehouse_ID=o.Warehouse_ID
16    ) where Traffic_Delay_Min is null;
17    SET SQL_SAFE_UPDATES = 1; -- Turn safe mode back on (Good practice)
```

Result

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Warehouse_ID	Location	Processing_Time_Min	Dispatch_Time	Traffic_Delay_Min	Delivery_Status
W001	Zacharyland	142	00:30:25	27	NULL
W002	Smithport	73	10:19:56	34	NULL
W003	Lake Cody	29	06:20:18	24	NULL
W004	Barnettville	84	02:08:31	27	NULL
W005	Lake Patriciaside	78	06:14:33	34	NULL
W006	Petershaven	27	05:24:56	31	NULL
W007	Port Juliachester	166	12:38:28	26	NULL
W008	Andersontown	161	11:25:53	30	NULL
W009	New Crystalborough	170	06:24:50	29	NULL
W010	Davidshire	171	18:57:42	27	NULL

I performed a multi-table imputation. I mapped each warehouse to its corresponding routes via the Orders table and replaced missing Traffic_Delay_Min values with the average traffic delay of those specific routes.

STANDARDIZE DATES

Task 1.3

We converted all date columns (Order Date, Expected Date, Actual Date) into the standard **YYYY-MM-DD** format for consistent SQL date operations.

SQL Query

```
15 •  UPDATE Orders
16    SET
17        Order_Date = DATE_FORMAT(Order_Date, '%Y-%m-%d'),
18        Expected_Delivery_Date = DATE_FORMAT(Expected_Delivery_Date, '%Y-%m-%d'),
19        Actual_Delivery_Date = DATE_FORMAT(Actual_Delivery_Date, '%Y-%m-%d');
```

Result

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Order_ID	Customer_ID	Warehouse_ID	Route_ID	Order_Date	Expected_Delivery_Date	Actual_Delivery_Date	Delivery_Status
O0001	C0080	W004	R005	2025-06-08	2025-06-11	2025-06-11	On Time
O0002	C0566	W009	R004	2025-06-22	2025-06-27	2025-06-29	Delayed
O0003	C0899	W008	R013	2025-06-04	2025-06-09	2025-06-11	Delayed
O0004	C0311	W001	R005	2025-06-28	2025-07-05	2025-07-07	Delayed
O0005	C0304	W006	R012	2025-06-04	2025-06-09	2025-06-10	Delayed
O0006	C0719	W004	R002	2025-07-17	2025-07-19	2025-07-19	On Time
O0007	C0652	W001	R020	2025-06-14	2025-06-17	2025-06-18	Delayed
O0008	C0111	W006	R008	2025-06-23	2025-06-30	2025-07-02	Delayed
O0009	C0138	W010	R011	2025-07-11	2025-07-16	2025-07-18	Delayed
O0010	C0096	W009	R017	2025-06-14	2025-06-21	2025-06-21	On Time
O0011	C0422	W008	R002	2025-05-26	2025-06-02	2025-06-02	On Time
O0012	C0838	W009	R014	2025-07-16	2025-07-20	2025-07-21	Delayed
O0013	C0543	W007	R002	2025-07-16	2025-07-18	2025-07-20	Delayed
O0014	C0385	W001	R007	2025-07-07	2025-07-11	2025-07-11	On Time

orders 12 x

LOGIC VALIDATION

Task 1.4

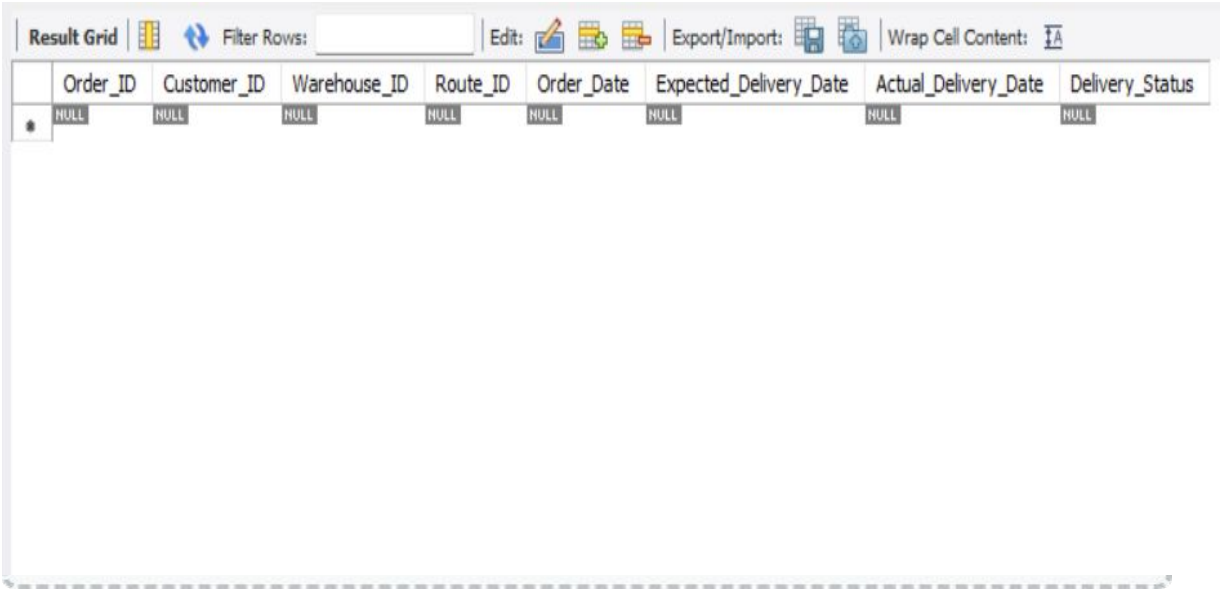
We ran a validation check to ensure no **Actual_Delivery_Date** was recorded before the **Order_Date**, flagging erroneous records for review.

</> SQL Query

PASTE SQL QUERY HERE

```
23 •  select * from orders WHERE Actual_Delivery_Date < Order_Date;
```

Result



TASK 2

DELIVERY DELAY ANALYSIS

Objective: Calculate specific delay metrics and rank orders to find localized issues.

- 2.1 Calculate Delays
- 2.2 Top 10 Delayed Routes
- 2.3 Rank Orders by Delay

CALCULATE DELIVERY DELAY

Task 2.1

We created a calculated field (`Actual - Expected`) to determine the exact delay in days for every single order.

SQL Query

```
24 •  SELECT
25        Order_ID,
26        Actual_Delivery_Date,
27        Expected_Delivery_Date,
28        DATEDIFF(Actual_Delivery_Date, Expected_Delivery_Date) AS Delay_Days
29    FROM Orders;
```

Result

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Order_ID	Actual_Delivery_Date	Expected_Delivery_Date	Delay_Days
O0004	2025-07-07	2025-07-05	2
O0005	2025-06-10	2025-06-09	1
O0006	2025-07-19	2025-07-19	0
O0007	2025-06-18	2025-06-17	1
O0008	2025-07-02	2025-06-30	2
O0009	2025-07-18	2025-07-16	2
O0010	2025-06-21	2025-06-21	0
O0011	2025-06-02	2025-06-02	0
O0012	2025-07-21	2025-07-20	1
O0013	2025-07-20	2025-07-18	2
O0014	2025-07-11	2025-07-11	0
O0015	2025-06-24	2025-06-24	0
O0016	2025-07-01	2025-06-29	2
O0017	2025-07-17	2025-07-17	0

Result 16

While we could have created a separate permanent column, it is a best practice to use a calculated field because it ensures the data is always dynamically accurate based on the latest date entries. This avoids data redundancy and keeps the database "lean" by calculating values only when needed for specific analysis

TOP 10 DELAYED ROUTES

Task 2.2

We aggregated the average delay by Route ID and sorted them to identify the **Top 10 worst-performing routes** requiring immediate attention.

SQL Query

```sql
-- Find Top 10 delayed routes based on average delay days.
SELECT r.Route_ID, r.Start_Location, r.End_Location,
    AVG(DATEDIFF(o.Actual_Delivery_Date, o.Expected_Delivery_Date))
    AS Avg_Delay_Days
FROM Orders o JOIN Routes r ON o.Route_ID = r.Route_ID
WHERE o.Delivery_Status = 'Delayed'
GROUP BY r.Route_ID, r.Start_Location, r.End_Location
ORDER BY Avg_Delay_Days DESC
LIMIT 10;
```

Chart

Result Grid | Filter Rows: | Export: | Wrap

Route_ID	Start_Location	End_Location	Avg_Delay_Days
R019	Lake Oscarfurt	New Ivanshire	2.0000
R015	South Jenniferfurt	Lindsayton	2.0000
R017	Port Tammybury	Abbottside	1.7500
R005	New Jessica	Davismouth	1.7143
R016	Lake Kathrynchester	Port Michael	1.6667
R001	New Aimeechester	Sabrinaland	1.6667
R012	West Calvinview	Port Audreyberg	1.6250
R004	Ramosside	Stevenview	1.6000
R014	North Edward	Matthewport	1.6000
R002	South Brianchester	Huffmanmouth	1.5714

Again I used a calculated field to create that column which calculates delay days for every row on the fly because it ensures the data is always dynamically accurate based on the latest date entries. This avoids data redundancy and keeps the database "lean" by calculating values only when needed for specific analysis

RANK ORDERS BY DELAY

Task 2.3

> Using window functions (DENSE_RANK), we ranked shipments by delay magnitude within each warehouse to isolate specific fulfillment bottlenecks.

</> SQL Query

```
49    -- Use window functions to rank all orders by delay within each warehouse.
50    SELECT Warehouse_ID, Order_ID, Expected_Delivery_Date, Actual_Delivery_Date,
51        DATEDIFF(Actual_Delivery_Date, Expected_Delivery_Date) AS Delay_Days,
52        -- DENSE_RANK ensures no numbers are skipped in the ranking sequence
53        DENSE_RANK() OVER (
54            PARTITION BY Warehouse_ID
55            ORDER BY DATEDIFF(Actual_Delivery_Date, Expected_Delivery_Date) DESC
56        ) AS Delay_Rank
57    FROM Orders
58    WHERE Delivery_Status = 'Delayed';
```

Result

Warehouse_ID	Order_ID	Expected_Delivery_Date	Actual_Delivery_Date	Delay_Days	Delay_Rank
W001	O0263	2025-06-03	2025-06-05	2	1
W001	O0007	2025-06-17	2025-06-18	1	2
W001	O0126	2025-07-05	2025-07-06	1	2
W001	O0223	2025-07-08	2025-07-09	1	2
W001	O0193	2025-06-14	2025-06-15	1	2
W001	O0047	2025-06-28	2025-06-29	1	2
W001	O0057	2025-06-15	2025-06-16	1	2
W001	O0186	2025-07-24	2025-07-25	1	2
W001	O0042	2025-07-20	2025-07-21	1	2
W001	O0160	2025-07-01	2025-07-02	1	2
W002	O0171	2025-06-14	2025-06-16	2	1
W002	O0199	2025-06-01	2025-06-03	2	1

TASK 3

ROUTE OPTIMIZATION INSIGHTS

Objective: Compute efficiency ratios to find suboptimal routes.

- 3.1 Route Statistics
- 3.2 Efficiency Ratio
- 3.3 High Delay Routes
- 3.4 Recommendations

ROUTE STATISTICS

Task 3.1

We calculated the Average Delivery Time (Days) and Average Traffic Delay (Minutes) for every route to establish a performance baseline.

SQL Query

```
-- Calculated Avg delivery time, Avg traffic delay, Distance-to-time efficiency ratio for each route
SELECT r.Route_ID, r.Start_Location, r.End_Location,
    -- 1. Average delivery time (Order_Date to Actual_Delivery_Date)
    round(AVG(DATEDIFF(o.Actual_Delivery_Date, o.Order_Date)), 1) AS Avg_Delivery_Time_Days,
    -- 2. Average traffic delay (from the Routes table)
    round((r.Traffic_Delay_Min)) AS Avg_Traffic_Delay_Min,
    -- 3. Distance-to-time efficiency ratio
    round((r.Distance_KM / r.Average_Travel_Time_Min), 2) AS Efficiency_Ratio
FROM Routes r JOIN Orders o ON r.Route_ID = o.Route_ID
GROUP BY r.Route_ID, r.Start_Location, r.End_Location, r.Distance_KM, r.Average_Travel_Time_Min
ORDER BY r.Route_ID;
```

Result

Route_ID	Start_Location	End_Location	Avg_Delivery_Time_Days	Avg_Traffic_Delay_Min	Efficiency_Ratio
R001	New Aimeechester	Sabrinaland	5.0	36	0.61
R002	South Brianchester	Huffmanmouth	4.7	22	2.16
R003	East Amy	South Katherinemouth	4.5	11	0.58
R004	Ramosside	Stevenview	5.1	3	4.28
R005	New Jessica	Davismouth	5.0	26	0.37
R006	Lake Robinhaven	East Christopherville	4.8	42	0.16
R007	Aaronhaven	Jeanneberg	5.8	51	0.28
R008	South Davidborough	Carsonberg	5.4	34	0.24
R009	Bellfurt	Lake Marvin	4.9	46	0.60
R010	North Aliceside	Lake Jennifer	5.3	40	0.67
R011	Lowemouth	Samanthaborough	5.6	25	0.96
R012	West Calvinview	Port Audreyberg	4.8	29	0.90
R013	South Johnville	Port Scott	4.4	38	0.93
R014	North Edward	Matthewport	5.6	58	2.28

EFFICIENCY RATIO ANALYSIS

Task 3.2

We calculated the **Distance-to-Time Efficiency Ratio** (Distance / Avg Time). We identified the 3 routes with the lowest ratios, indicating poor travel efficiency.

</> SQL Query

```sql
-- Identified Top 3 Routes with the Worst Efficiency Ratio
SELECT Route_ID, Start_Location, End_Location,
round((Distance_KM / Average_Travel_Time_Min), 2)
AS Efficiency_Ratio FROM Routes
ORDER BY Efficiency_Ratio ASC
LIMIT 3;
```

Result

Route_ID	Start_Location	End_Location	Efficiency_Ratio
R016	Lake Kathrynchester	Port Michael	0.15
R006	Lake Robinhaven	East Christopherville	0.16
R008	South Davidborough	Carsonberg	0.24

HIGH DELAY ROUTES (>20%)

Task 3.3

We filtered for routes where **more than 20%** of total shipments missed their expected delivery date.

SQL Query

```sql
-- Routes with >20% delayed shipments
SELECT Route_ID,
-- Step 1: Count orders that were actually delayed
COUNT(CASE WHEN Delivery_Status = 'Delayed' THEN 1 END) AS Delayed_Orders,
-- Step 2: Count the total number of orders for that route
COUNT(*) AS Total_Orders,
-- Step 3: Calculate the percentage and round it
ROUND((COUNT(CASE WHEN Delivery_Status = 'Delayed' THEN 1 END) / COUNT(*)) * 100, 2)
AS Delay_Percentage FROM Orders
GROUP BY Route_ID HAVING Delay_Percentage > 20
ORDER BY Delay_Percentage DESC;
```

Result

	Route_ID	Delayed_Orders	Total_Orders	Delay_Percentage
	R013	9	17	52.94
	R004	10	20	50.00
	R015	3	6	50.00
	R005	7	15	46.67
	R011	9	20	45.00
	R010	9	20	45.00
	R007	7	16	43.75
	R003	4	10	40.00
	R001	6	15	40.00
	R018	6	15	40.00
	R014	5	13	38.46
	R009	6	16	37.50

Result 35

RECOMMENDATIONS

Task 3.4

Based on the data, we identified specific routes that require re-planning or mode switching to improve speed.

Finding potential routes for immediate optimization :

```sql
SELECT r.Route_ID, r.Start_Location, r.End_Location,
ROUND((r.Distance_KM / r.Average_Travel_Time_Min), 2) AS Efficiency_Ratio,
r.Traffic_Delay_Min, delay_stats.Delay_Percentage
FROM Routes r JOIN (
    SELECT
    Route_ID,
    ROUND((COUNT(CASE WHEN Delivery_Status = 'Delayed' THEN 1 END) / COUNT(*)) * 100, 2) AS Delay_Percentage
    FROM Orders
    GROUP BY Route_ID
) AS delay_stats ON r.Route_ID = delay_stats.Route_ID
WHERE
    (r.Distance_KM / r.Average_Travel_Time_Min) < 0.6 -- Slow routes
    AND delay_stats.Delay_Percentage > 20            -- High failure routes
ORDER BY delay_stats.Delay_Percentage DESC;
```

Route_ID	Start_Location	End_Location	Efficiency_Ratio	Traffic_Delay_Min	Delay_Percentage
R020	Juliestad	South Jonathanshire	0.55	23	70.59
R017	Port Tammybury	Abbottside	0.33	8	57.14
R015	South Jenniferfurt	Lindsayton	0.30	29	50.00
R005	New Jessica	Davismouth	0.37	26	46.67
R007	Aaronhaven	Jeanneberg	0.28	51	43.75
R003	East Amy	South Katherinemouth	0.58	11	40.00
R018	Lake Jessicastad	Samuelport	0.54	11	40.00
R006	Lake Robinhaven	East Christopherville	0.16	42	36.36
R008	South Davidborough	Carsonberg	0.24	34	31.25
R016	Lake Kathrynchester	Port Michael	0.15	8	25.00

ROUTE OPTIMIZATION RECOMMENDATIONS

Objective: Analyze processing times and throughput to rank facilities. This. of over 40 minutes), I recommend rescheduling deliveries to off-peak hours or utilizing.

Primary Routes to Optimize (Ordered by Route ID)

- **R003 & R005:** High delay rates (40%+) despite moderate traffic.
- **R006, R007 & R008:** Critical bottlenecks with very low efficiency ratios (< 0.30) and high traffic congestion.
- **R015, R016, R017 & R018:** Frequent failures where more than 1 in 4 shipments are late.
- **R020:** The most unreliable route in the network, with a 70.59% delay rate.

Key Recommendations for Improvement

- **Reduce Traffic Impact:** For routes like R007 and R006 (which have the highest traffic delays of over 40 minutes), I recommend rescheduling deliveries to off-peak hours or utilizing alternative smaller roads to bypass main highway congestion.
- **Improve Last-Mile Speed:** Routes R016 and R006 have the lowest physical efficiency (0.15 and 0.16 KM/Min). We should consider replacing heavy delivery trucks with smaller, more agile vehicles (like electric vans) to navigate these locations faster.
- **Investigate Processing Delays:** Route R020 has a massive 70% delay rate but decent road speed. This suggests the delay isn't happening on the road, but during the loading or unloading process. A warehouse audit is recommended for the start and end points of this route.

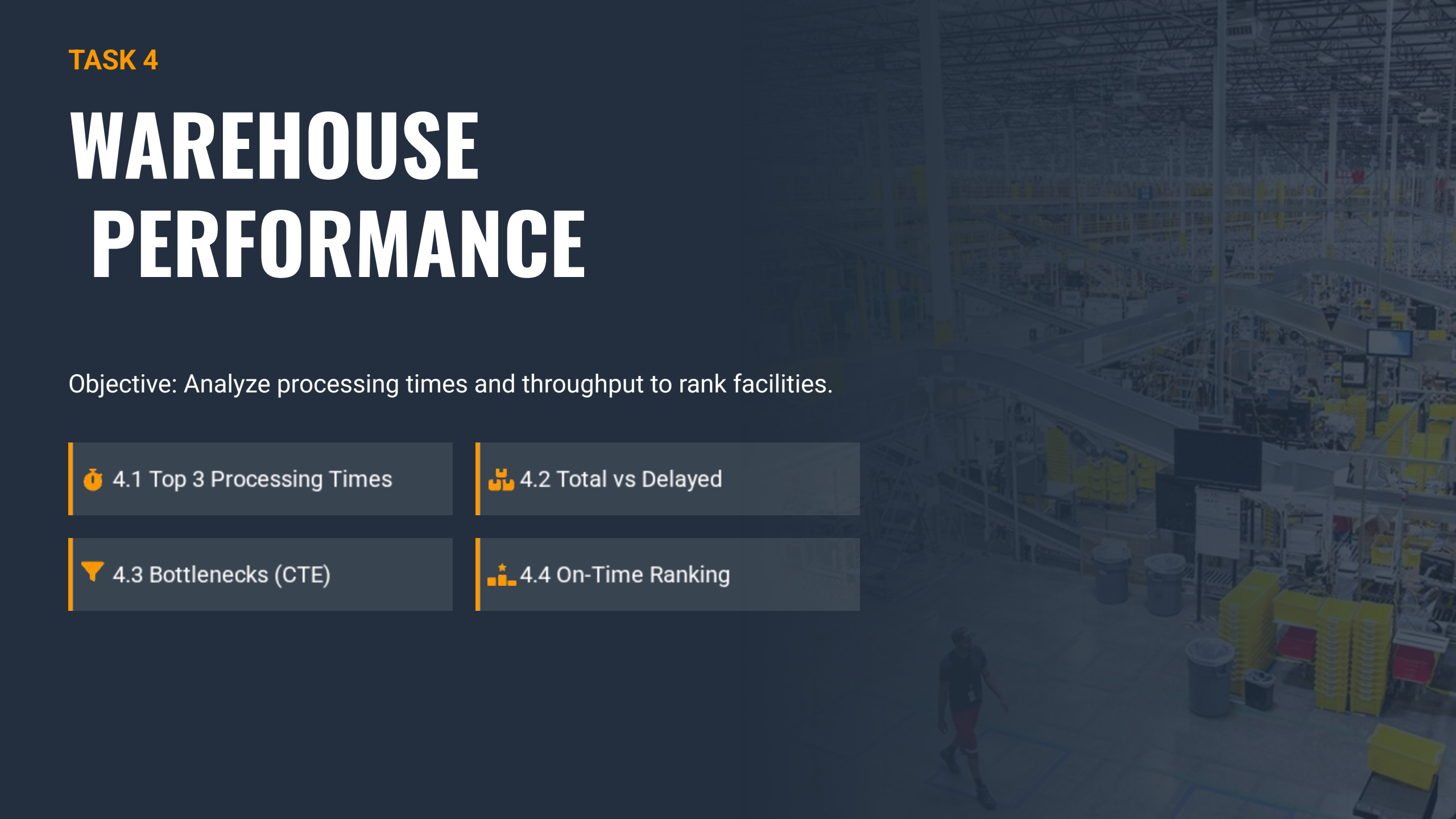
Summary: Analyze the top processing processing times through our now efficiency ratios. (< 0.30): tnv/raffic congess, and okay Recommendations, utilizing awehouse audit is recommend to one main highway congestion.

TASK 4

WAREHOUSE PERFORMANCE

Objective: Analyze processing times and throughput to rank facilities.

- 4.1 Top 3 Processing Times
- 4.2 Total vs Delayed
- 4.3 Bottlenecks (CTE)
- 4.4 On-Time Ranking



TOP 3 HIGH PROCESSING TIMES

Task 4.1

We sorted warehouses by their average processing time to identify the top 3 facilities that are slowest to dispatch goods.

SQL Query

```sql
-- Top 3 warehouses with the highest average processing time.
SELECT Warehouse_ID, Location, Processing_Time_Min FROM warehouses
ORDER BY Processing_Time_Min DESC LIMIT 3;
```

Result

Warehouse_ID	Location	Processing_Time_Min
W010	Davidshire	171
W009	New Crystalborough	170
W007	Port Juliachester	166
NULL	NULL	NULL

TOTAL VS DELAYED SHIPMENTS

Task 4.2

We calculated the volume of total shipments versus delayed shipments for each warehouse to understand their workload and failure rates.

SQL Query

```sql
-- Calculate total vs. delayed shipments for each warehouse
SELECT Warehouse_ID, COUNT(Order_ID) as Total_Shipments,
SUM(CASE WHEN Delivery_Status = 'Delayed' THEN 1 ELSE 0 END) as Delayed_Shipments
FROM orders GROUP BY Warehouse_ID -- ORDER BY Total_Shipments DESC;
ORDER BY Warehouse_ID;
```

Result

Warehouse_ID	Total_Shipments	Delayed_Shipments
W001	34	18
W002	28	10
W003	23	13
W004	31	12
W005	25	14
W006	37	12
W007	38	15
W008	21	5
W009	33	13
W010	30	20

BOTTLENECK IDENTIFICATION

Task 4.3

Using **CTEs**, we compared individual warehouse processing times against the Global Average to pinpoint specific bottlenecks.

SQL Query

```sql
-- Using CTE to find bottleneck warehouses where processing time > global average.
WITH Global_Average_Table AS ( -- CTE 1: Calculate the benchmark (Global Average)
    SELECT AVG(Processing_Time_Min) AS Global_Avg
    FROM Warehouses
), -- We use a comma here to define the next CTE, NOT the 'WITH' keyword
-- CTE 2: Identify only the warehouses that exceed that benchmark
Bottleneck_List AS (
    SELECT w.Warehouse_ID, w.Location, w.Processing_Time_Min, g.Global_Avg
    FROM Warehouses w CROSS JOIN Global_Average_Table g
    WHERE w.Processing_Time_Min > g.Global_Avg
)
SELECT * FROM Bottleneck_List
ORDER BY Processing_Time_Min DESC;
```

Result

Warehouse_ID	Location	Processing_Time_Min	Global_Avg
W010	Davidshire	171	110.1000
W009	New Crystalborough	170	110.1000
W007	Port Juliachester	166	110.1000
W008	Andersontown	161	110.1000
W001	Zacharyland	142	110.1000

ON-TIME RANKING

Task 4.4

We ranked all warehouses based on their **On-Time Delivery Percentage** to create a performance leaderboard.

SQL Query

```sql
-- Rank warehouses based on on-time delivery percentage.
WITH OnTime_Percentage_CTE AS (
    SELECT Warehouse_ID,
    ROUND((SUM(CASE WHEN Delivery_Status = 'On Time' THEN 1 ELSE 0 END) / COUNT(*)) * 100, 2)
    AS On_Time_Percentage FROM Orders GROUP BY Warehouse_ID
)
SELECT Warehouse_ID, On_Time_Percentage,
    DENSE_RANK() OVER (ORDER BY On_Time_Percentage DESC) AS Warehouse_Rank
FROM OnTime_Percentage_CTE;
```

Chart

Warehouse_ID	On_Time_Percentage	Warehouse_Rank
W008	76.19	1
W006	67.57	2
W002	64.29	3
W004	61.29	4
W009	60.61	5
W007	60.53	6
W001	47.06	7
W005	44.00	8
W003	43.48	9
W010	33.33	10

TASK 5

DELIVERY AGENT PERFORMANCE

Objective: Profile agents by reliability and speed to identify training needs.

- 5.1 Rank Agents
- 5.2 Underperformers (<80%)
- 5.3 Speed Comparison

RANK AGENTS

Task 5.1

We ranked delivery agents for each route based on their **On-Time Delivery Percentage**.

SQL Query

```
-- Rank agents (per route) by on-time delivery percentage
SELECT Agent_ID, Route_ID, On_Time_Percentage,
DENSE_RANK() OVER (
        PARTITION BY Route_ID
        ORDER BY On_Time_Percentage DESC
    ) AS Agent_Route_Rank
FROM Delivery_Agents;
```

Result

Agent_ID	Route_ID	On_Time_Percentage	Agent_Route_Rank
A006	R001	81.56	1
A037	R001	78.94	2
A033	R002	98.01	1
A024	R002	97.16	2
A027	R002	73.80	3
A050	R002	72.75	4
A011	R002	71.50	5
A015	R003	83.31	1
A029	R005	81.71	1
A035	R005	74.50	2

IDENTIFY UNDERPERFORMERS

Task 5.2

We filtered for agents with an On-Time Percentage **below 80%** to identify candidates for retraining.

SQL Query

```sql
-- Find agents with on-time % < 80%
SELECT Agent_ID, Route_ID, On_Time_Percentage
FROM Delivery_Agents
WHERE On_Time_Percentage < 80
ORDER BY On_Time_Percentage ASC;
```

Result

Agent_ID	Route_ID	On_Time_Percentage
A010	R019	70.45
A011	R002	71.50
A038	R018	71.80
A008	R018	72.39
A039	R016	72.60
A050	R002	72.75
A018	R007	73.21
A027	R002	73.80
A003	R014	74.15
A023	R017	74.21
A028	R007	74.47

SPEED VS RELIABILITY

Task 5.3

We compared the Average Speed of the **Top 5 Agents vs. the Bottom 5 Agents** to see if higher speeds correlate with better on-time performance.

SQL Query

```sql
-- Compare average speed of top 5 vs bottom 5 agents using subqueries.
SELECT
    (SELECT AVG(Avg_Speed_KM_HR) FROM (
        SELECT Avg_Speed_KM_HR FROM Delivery_Agents
        ORDER BY On_Time_Percentage DESC LIMIT 5
    ) AS Top_5) AS Top5_Agents_Avg_Speed,
    (SELECT AVG(Avg_Speed_KM_HR) FROM (
        SELECT Avg_Speed_KM_HR FROM Delivery_Agents
        ORDER BY On_Time_Percentage ASC LIMIT 5
    ) AS Bottom_5) AS Bottom5_Agents_Avg_Speed,
    -- Calculate the difference to show the gap
    (SELECT AVG(Avg_Speed_KM_HR) FROM (
        SELECT Avg_Speed_KM_HR FROM Delivery_Agents
        ORDER BY On_Time_Percentage DESC LIMIT 5
    ) AS Top_5) -
    (SELECT AVG(Avg_Speed_KM_HR) FROM (
        SELECT Avg_Speed_KM_HR FROM Delivery_Agents
        ORDER BY On_Time_Percentage ASC LIMIT 5
    ) AS Bottom_5) AS Speed_Gap;
```

Result

Top5_Agents_Avg_Speed	Bottom5_Agents_Avg_Speed	Speed_Gap
60.000000	56.800000	3.200000

TASK 6

SHIPMENT TRACKING ANALYTICS

Objective: Analyze checkpoints to find delay reasons and tracking failures.

- 6.1 Last Checkpoint Status
- 6.2 Common Delay Reasons
- 6.3 High Risk Orders

LAST CHECKPOINT STATUS

Task 6.1

> For every order, we retrieved the last recorded checkpoint and its timestamp to visualize the current status of shipments.

SQL Query

```
--   For each order, list the last checkpoint and time.
WITH Ranked_Checkpoints AS (
    SELECT Order_ID, Checkpoint, Checkpoint_Time,
        -- Rank by time first, then by the Checkpoint name to break ties
        ROW_NUMBER() OVER (
            PARTITION BY Order_ID
            ORDER BY Checkpoint_Time DESC, Checkpoint DESC
        ) AS Recent_Rank
    FROM Shipment_Tracking
)
SELECT Order_ID, Checkpoint AS Last_Checkpoint, Checkpoint_Time AS Last_Time
FROM Ranked_Checkpoints WHERE Recent_Rank = 1
ORDER BY Order_ID;
```

Result

Order_ID	Last_Checkpoint	Last_Time
O0001	Checkpoint 2	2025-06-09 00:00:00
O0002	Checkpoint 4	2025-06-23 00:00:00
O0003	Checkpoint 4	2025-06-05 00:00:00
O0004	Checkpoint 5	2025-06-29 00:00:00
O0005	Checkpoint 5	2025-06-04 00:00:00
O0006	Checkpoint 2	2025-07-18 00:00:00
O0007	Checkpoint 3	2025-06-15 00:00:00
O0008	Checkpoint 2	2025-06-24 00:00:00
O0009	Checkpoint 3	2025-07-11 00:00:00
O0010	Checkpoint 4	2025-06-15 00:00:00
O0011	Checkpoint 5	2025-05-27 00:00:00
O0012	Checkpoint 3	2025-07-17 00:00:00

COMMON DELAY REASONS

Task 6.2

We aggregated delay reasons (excluding 'None') to identify the most frequent causes of stoppages (e.g., Weather, Traffic).

</> SQL Query

```sql
-- Most common delay reasons (excluding None).
SELECT Delay_Reason, COUNT(*) AS Occurrence_Count
FROM Shipment_Tracking
WHERE Delay_Reason IS NOT NULL AND Delay_Reason != 'None'
GROUP BY Delay_Reason
ORDER BY Occurrence_Count DESC;
```

Chart

Delay_Reason	Occurrence_Count
Sorting Delay	272
Weather	263
Traffic	247

HIGH RISK ORDERS

Task 6.3

We identified specific orders that were flagged as 'Delayed' at **more than two separate checkpoints** during their journey.

SQL Query

```sql
-- Identify orders with >2 delayed checkpoints
SELECT Order_ID, COUNT(*) AS Delayed_Checkpoints_Count
FROM Shipment_Tracking
WHERE Delay_Reason IS NOT NULL AND Delay_Reason != 'None'
GROUP BY Order_ID HAVING COUNT(*) > 2;
```

Result

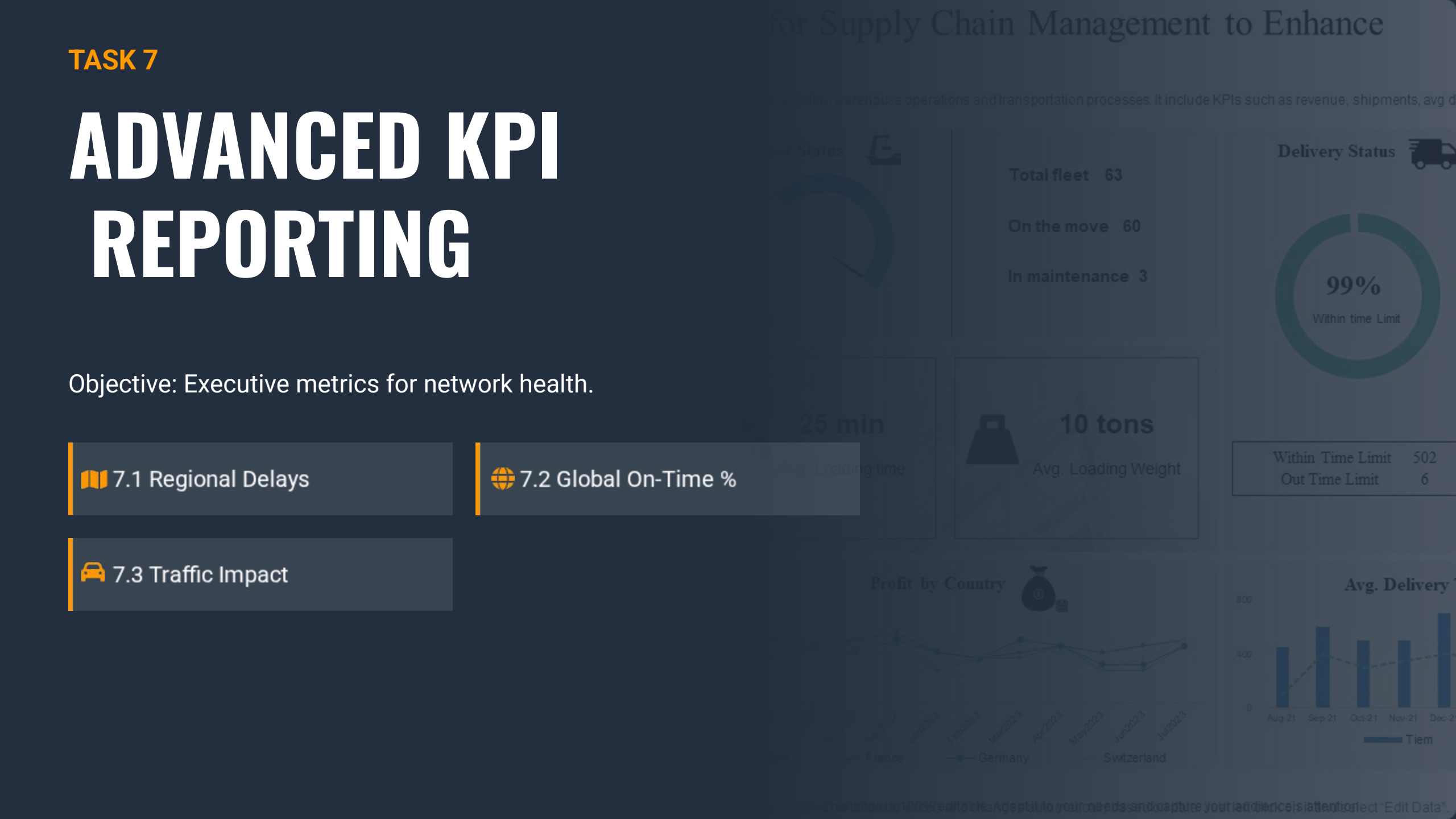
Order_ID	Delayed_Checkpoints_Count
O0002	3
O0003	4
O0004	4
O0005	4
O0007	3
O0015	4
O0016	5
O0019	5
O0021	4
O0024	4
O0025	3
O0027	3

TASK 7

ADVANCED KPI REPORTING

Objective: Executive metrics for network health.

- 7.1 Regional Delays
- 7.2 Global On-Time %
- 7.3 Traffic Impact



AVG DELIVERY DELAY PER REGION

Task 7.1

We calculated the average delay days grouped by the Start Location (Region) to find geographic pain points.

SQL Query

```sql
-- Average Delivery Delay per Region (Start_Location).
SELECT r.Start_Location AS Region,
ROUND(AVG(DATEDIFF(o.Actual_Delivery_Date, o.Expected_Delivery_Date)), 2)
AS Avg_Delivery_Delay_Days FROM Orders o
JOIN Routes r ON o.Route_ID = r.Route_ID
GROUP BY r.Start_Location
ORDER BY Avg_Delivery_Delay_Days DESC;
```

Result

Region	Avg_Delivery_Delay_Days
West Calvinview	1.08
Juliestad	1.06
Port Tammybury	1.00
South Jenniferfurt	1.00
New Jessica	0.80
Ramosside	0.80
South Johnville	0.76
Lowemouth	0.70
North Aliceside	0.70
Aaronhaven	0.69
New Aimeechester	0.67
North Edward	0.62

GLOBAL ON-TIME DELIVERY %

Task 7.2

We calculated the overall success rate: (Total On-Time Deliveries / Total Deliveries) * 100.

</> SQL Query

```sql
-- On-Time Delivery %
SELECT COUNT(*) AS Total_Deliveries,
SUM(CASE WHEN Delivery_Status = 'On Time' THEN 1 ELSE 0 END)
AS Total_On_Time_Deliveries,
ROUND(
(SUM(CASE WHEN Delivery_Status = 'On Time' THEN 1 ELSE 0 END) / COUNT(*)) * 100,
2
) AS On_Time_Percentage
FROM Orders;
```

Result

Total_Deliveries	Total_On_Time_Deliveries	On_Time_Percentage
300	168	56.00

AVG TRAFFIC DELAY PER ROUTE

Task 7.3

We quantified the impact of external factors by calculating the average traffic delay minutes per route.

SQL Query

```sql
-- Average Traffic Delay per Route
SELECT Route_ID,
ROUND(AVG(Traffic_Delay_Min), 2) AS Avg_Traffic_Delay_Min
FROM Routes GROUP BY Route_ID ORDER BY Route_ID;
-- ORDER BY Avg_Traffic_Delay_Min DESC;
```

Result

Route_ID	Avg_Traffic_Delay_Min
R001	36.00
R002	22.00
R003	11.00
R004	3.00
R005	26.00
R006	42.00
R007	51.00
R008	34.00
R009	46.00
R010	40.00
R011	25.00
R012	29.00

Conclusion & Video

The analysis pinpoints specific low-efficiency routes and bottleneck warehouses.

Addressing these will improve the Global On-Time Percentage.

https://drive.google.com/file/d/1Zj5GxmIIO6Nl9sMSSg4seTVlkkUWqwNG/view?usp=sharing