Amazon Logistics Optimization SQL Project

BY : ASWANI KASIREDDY

Project Introduction:

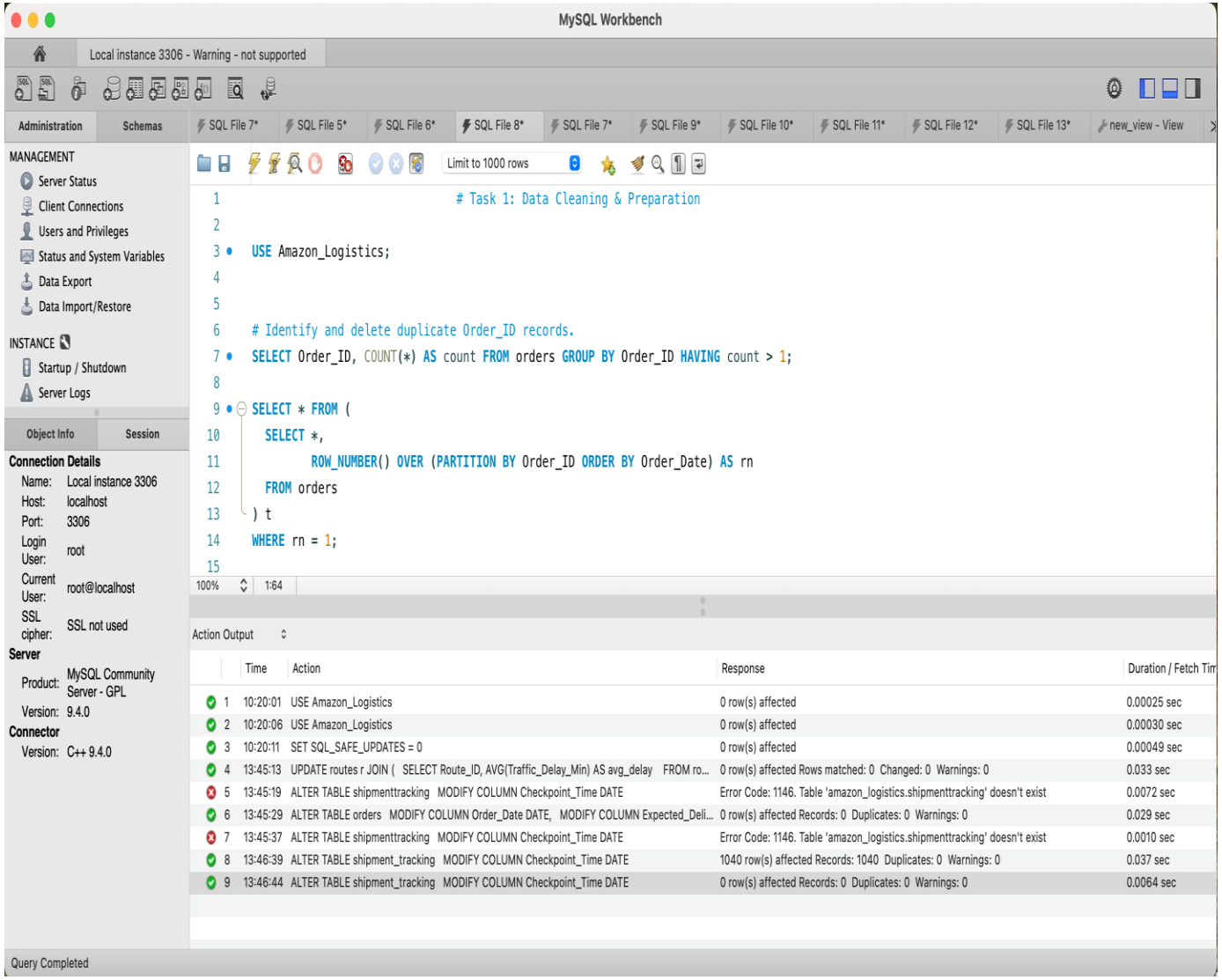
Amazon, a global e-commerce leader, handles millions of daily shipments across various regions. The logistics network is vast, comprising warehouses, fulfillment centers, and last-mile delivery partners. With increasing order volumes, delays and inefficiencies in delivery routes can significantly affect customer satisfaction and operational costs.

PROJECT OBJECTIVE:

Build a SQL-driven Logistics analytics system to analyze delays, optimize routes, and enhance shipment efficiency by leveraging queries, aggregations. The project aims to answer key business questions, uncover inefficiencies, and recommend actionable improvements based on data analysis.

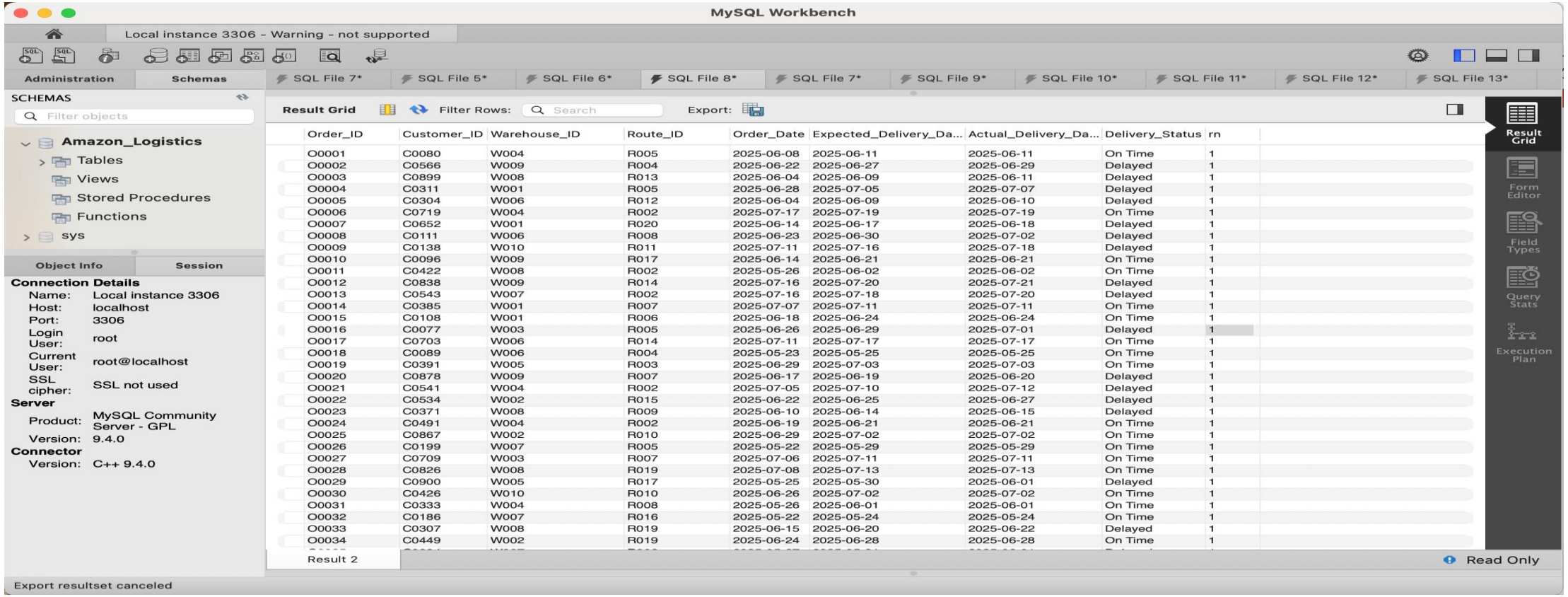
Task 1: Data Cleaning & Preparation

```
USE Amazon_Logistics;
# 1. Identify and delete duplicate Order_ID records.
SELECT Order_ID, COUNT(*) AS count FROM orders GROUP BY Order_ID HAVING count > 1;

SELECT * FROM (
 SELECT *,
      ROW_NUMBER() OVER (PARTITION BY Order_ID ORDER BY Order_Date) AS rn
 FROM orders
) t
WHERE rn = 1;

DELETE FROM orders WHERE Order_ID IN (
 SELECT Order_ID FROM (
  SELECT Order_ID, ROW_NUMBER() OVER (PARTITION BY Order_ID ORDER BY Order_ID) AS row_num
  FROM orders
 ) t WHERE row_num > 1
);

# 2. Replace null Traffic_Delay_Min with the average delay for that route.
SELECT * FROM routes WHERE Traffic_Delay_Min IS NULL;

UPDATE routes r
JOIN (
 SELECT Route_ID, AVG(Traffic_Delay_Min) AS avg_delay
 FROM routes
 WHERE Traffic_Delay_Min IS NOT NULL
 GROUP BY Route_ID
) t ON r.Route_ID = t.Route_ID
SET r.Traffic_Delay_Min = t.avg_delay
WHERE r.Traffic_Delay_Min IS NULL;

# 3. Convert all date columns into YYYY-MM-DD format using SQL functions.
ALTER TABLE orders
 MODIFY COLUMN Order_Date DATE,
 MODIFY COLUMN Expected_Delivery_Date DATE,
 MODIFY COLUMN Actual_Delivery_Date DATE;

ALTER TABLE shipmenttracking
 MODIFY COLUMN Checkpoint_Time DATE;

# 4. Ensure that no Actual_Delivery_Date is before Order_Date (flag such records).
SELECT COUNT(*) AS invalid_delivery_count FROM orders WHERE Actual_Delivery_Date < Order_Date;

SELECT Order_ID, Order_Date, Actual_Delivery_Date,
CASE WHEN Actual_Delivery_Date < Order_Date THEN 'Invalid' ELSE 'Valid' END AS status
FROM orders WHERE Actual_Delivery_Date < Order_Date;
```



Insights:

- No duplicate Order_ID records were found, ensuring the dataset maintains data integrity.
- All missing values in Traffic_Delay_Min were successfully replaced with the average delay calculated per route, improving completeness.
- All date columns were standardized into YYYY-MM-DD format, ensuring uniformity across records for time-based analysis.
- Validation confirmed that no Actual_Delivery_Date occurs before Order_Date, indicating no timeline inconsistencies.
- The dataset is now fully cleaned and ready for accurate analysis in subsequent tasks.

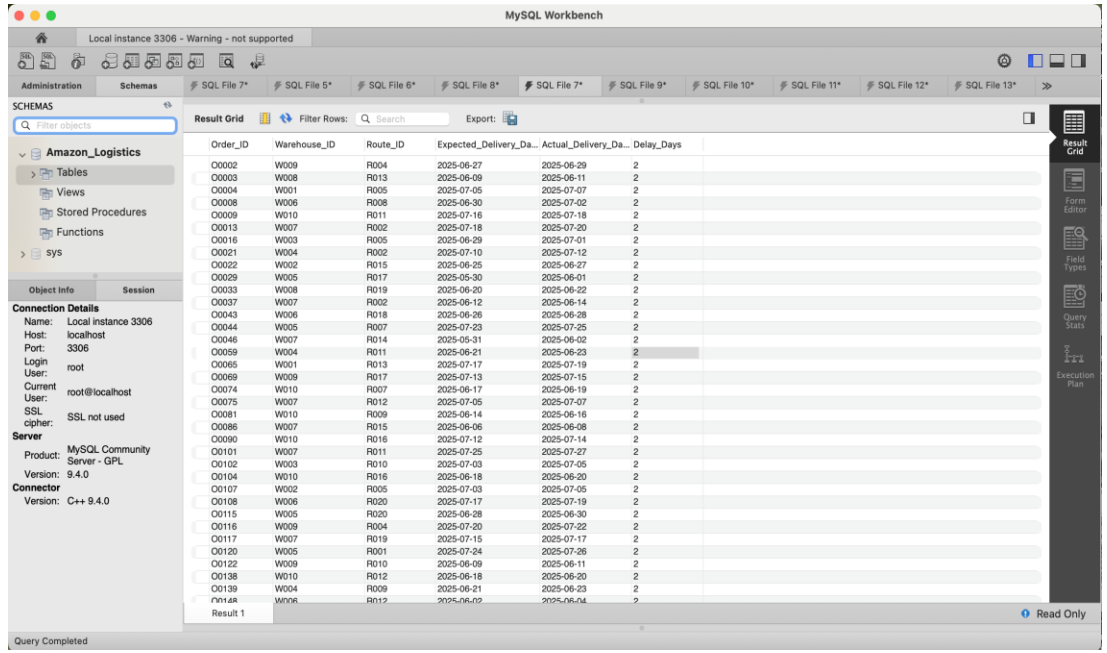


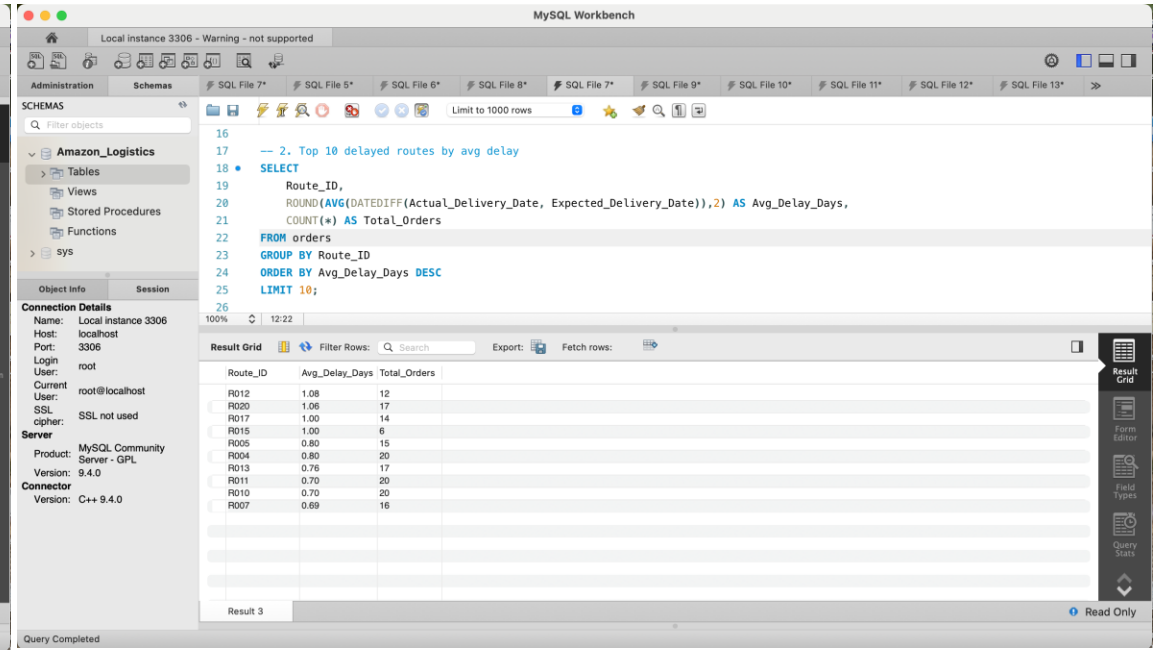
Task 2: Delivery Delay Analysis

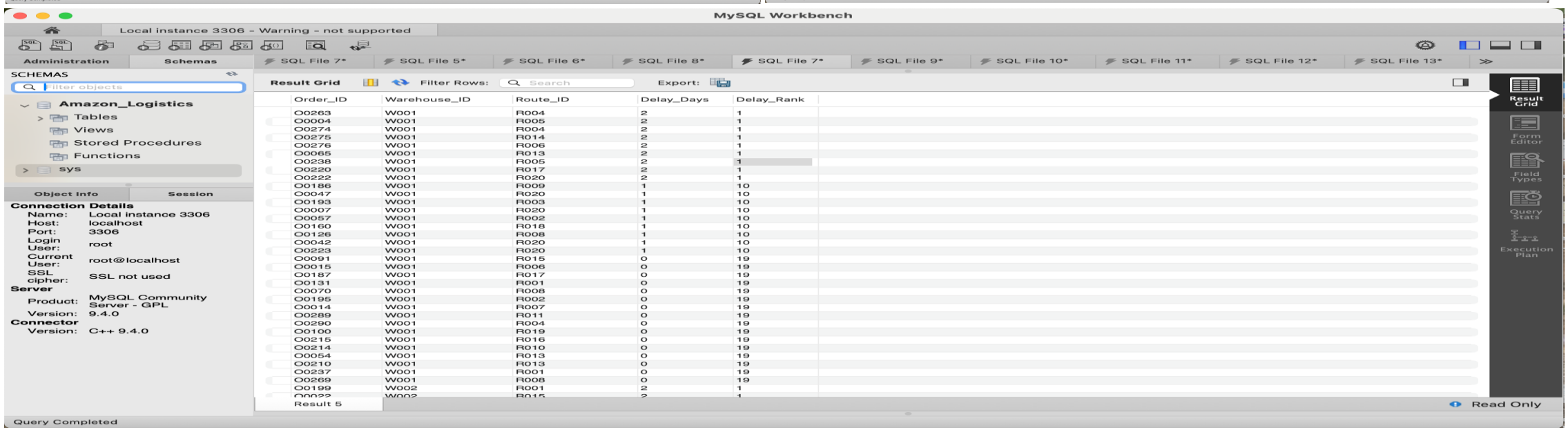
```
-- 1.Calculate delivery delay for each order
SELECT
   Order_ID,
   Warehouse_ID,
   Route_ID,
   Expected_Delivery_Date,
   Actual_Delivery_Date,
   DATEDIFF(Actual_Delivery_Date, Expected_Delivery_Date) AS Delay_Days
FROM orders
ORDER BY Delay_Days DESC;

-- 2. Top 10 delayed routes by avg delay
SELECT
   Route_ID,
   ROUND(AVG(DATEDIFF(Actual_Delivery_Date, Expected_Delivery_Date)),2) AS Avg_Delay_Days,
   COUNT(*) AS Total_Orders
FROM orders
GROUP BY Route_ID
ORDER BY Avg_Delay_Days DESC
LIMIT 10;


- 3. Rank orders within each warehouse by delay
SELECT
   Order_ID,
   Warehouse_ID,
   Route_ID,
   DATEDIFF(Actual_Delivery_Date, Expected_Delivery_Date) AS Delay_Days,
   RANK() OVER (PARTITION BY Warehouse_ID ORDER BY DATEDIFF(Actual_Delivery_Date, Expected_Delivery_Date) DESC) AS Delay_Rank
FROM orders
ORDER BY Warehouse_ID, Delay_Rank;
```

Insights:

- Delivery delays mostly ranged between **1–2 days** across orders.
- **Top delayed routes** were identified as **R012, R020, and R017** with the highest average delays.
- Several other routes (**R015, R011, R010, R007**) also showed recurring delays.
- **Warehouse W001** recorded the **highest number of 2-day delays**, marking it as a critical bottleneck.
- **Warehouse W010** showed consistent delays, affecting overall delivery timelines.
- Delay ranking placed **W001, W002, and W003** among the top warehouses with severe delays.
- Insights that indicated the need to optimize **R012, R020, R017 routes** and improve **W001 & W010 warehouse operations**.


MySQL Workbench
Local instance 3306 - Warning - not supported
SCHEMAS
Amazon_Logistics
Tables
Views
Stored Procedures
Functions
sys
Result Grid
Filter Rows:
Export:
Order_ID
Warehouse_ID
Route_ID
Expected_Delivery_Da...
Actual_Delivery_Da...
Delay_Days
Result 1
Read Only
Query Completed


MySQL Workbench
Local instance 3306 - Warning - not supported
-- 2. Top 10 delayed routes by avg delay
SELECT
Route_ID,
ROUND(AVG(DATEDIFF(Actual_Delivery_Date, Expected_Delivery_Date)),2) AS Avg_Delay_Days,
COUNT(*) AS Total_Orders
FROM orders
GROUP BY Route_ID
ORDER BY Avg_Delay_Days DESC
LIMIT 10;
Result Grid
Route_ID
Avg_Delay_Days
Total_Orders
R012 1.08 12
R020 1.06 17
R017 1.00 14
R015 1.00 6
R005 0.80 15
R004 0.80 20
R013 0.76 17
R011 0.70 20
R010 0.70 20
R007 0.69 16
Result 3
Read Only
Query Completed


MySQL Workbench
Local instance 3306 - Warning - not supported
SCHEMAS
Amazon_Logistics
Tables
Views
Stored Procedures
Functions
sys
Result Grid
Filter Rows:
Export:
Order_ID
Warehouse_ID
Route_ID
Delay_Days
Delay_Rank
Result 5
Read Only
Query Completed

Task 3: Route Optimization Insights

```
-- 1.For each route, calculate:
# Average delivery time (in days)
SELECT Route_ID, AVG(DATEDIFF(Actual_Delivery_Date, Order_Date)) AS avg_delivery_days
FROM orders
GROUP BY Route_ID;
# Average traffic delay
SELECT Route_ID, ROUND(AVG(Traffic_Delay_Min),2) AS avg_traffic_delay
FROM routes
GROUP BY Route_ID;
 # Distance-to-time efficiency ratio: Distance_KM / Average_Travel_Time_Min.
SELECT Route_ID, Distance_KM / Average_Travel_Time_Min AS efficiency_ratio
FROM routes;
-- 2.Identify 3 routes with the worst efficiency ratio.
# (Assuming worst = lowest efficiency ratio)
SELECT Route_ID, Distance_KM / Average_Travel_Time_Min AS efficiency_ratio
FROM routes
ORDER BY efficiency_ratio ASC LIMIT 3;
-- 3. Find routes with >20% delayed shipments.
SELECT
   Route_ID,
   ROUND(SUM(CASE WHEN Delivery_Status = 'Delayed' THEN 1 ELSE 0 END) * 100.0 / COUNT(*),2) AS delayed_percent
FROM orders
GROUP BY Route_ID
HAVING delayed_percent > 20;
```

-- 4. Recommend potential routes for optimization.

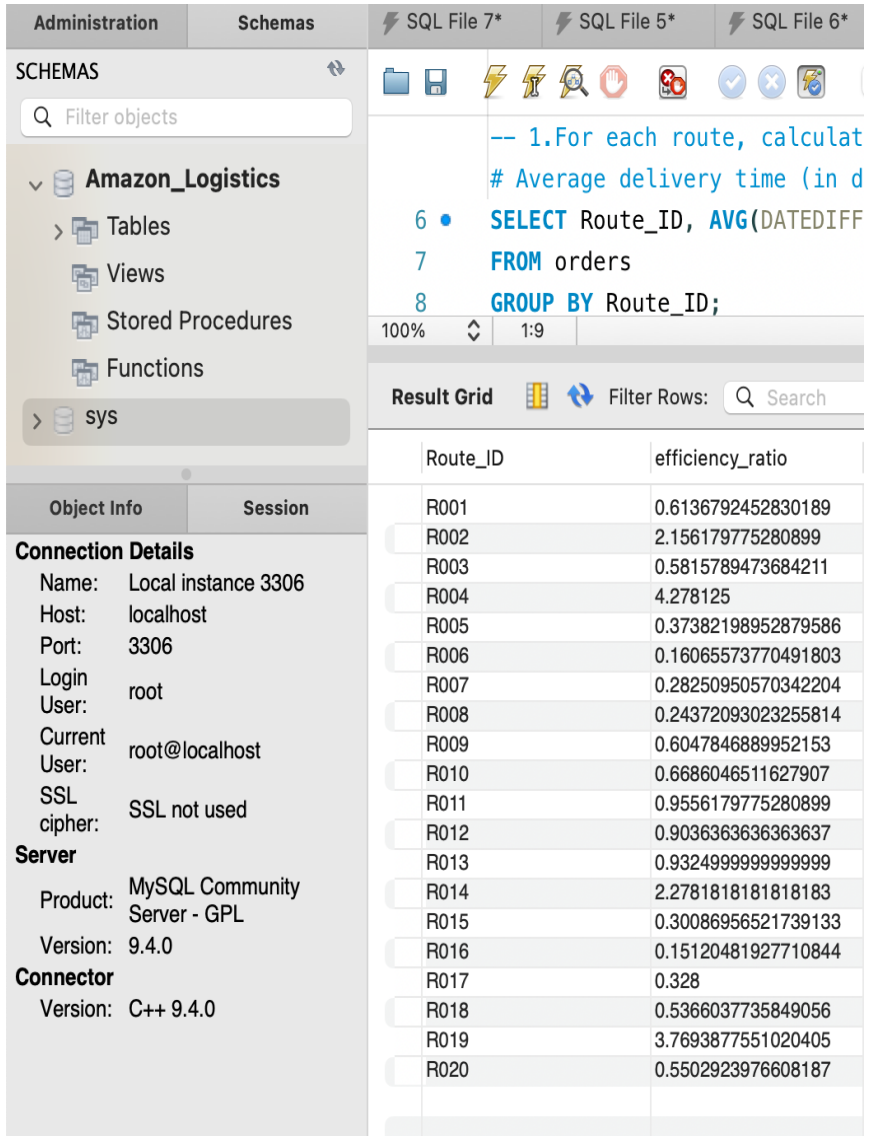
Optimize R020 (Juliestad to South Jonathanshire) with dynamic rerouting and top-performing agents (>90% on-time) due to 70.59% delayed shipments and 5.88-day average delivery.

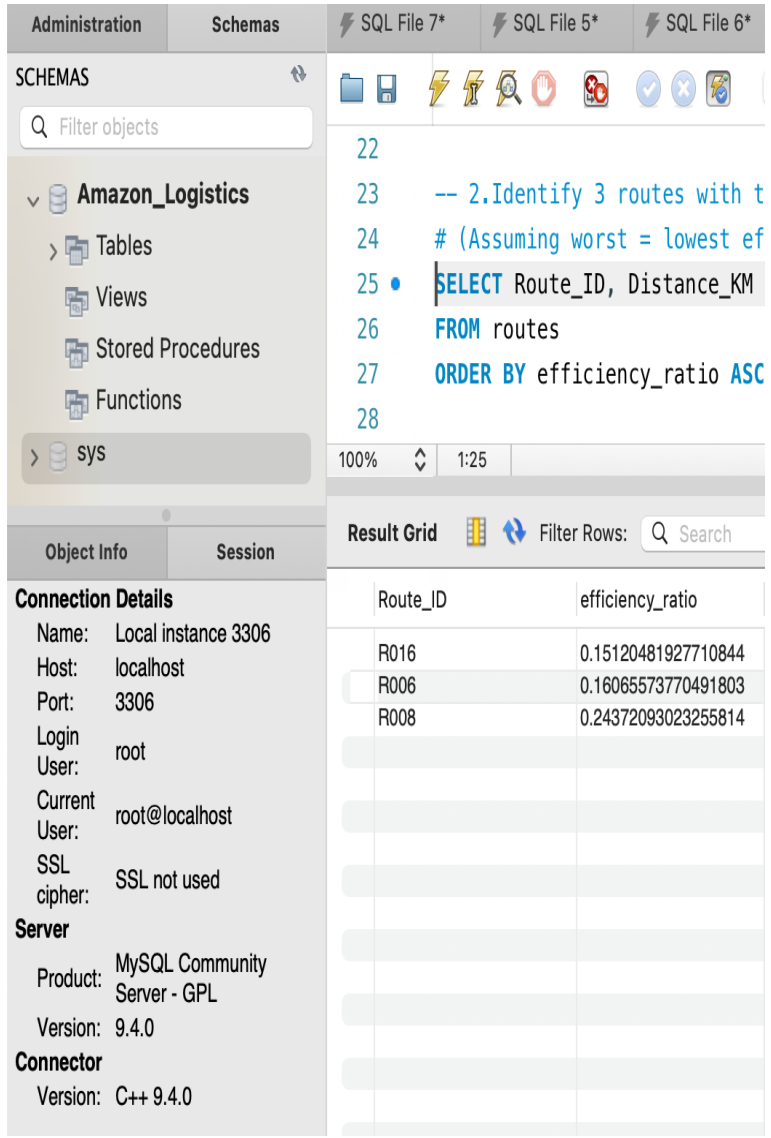
Shorten R017 (Port Tammybury to Abbottside) route or automate warehouse sorting for 57.14% delays and 6.29-day average delivery.

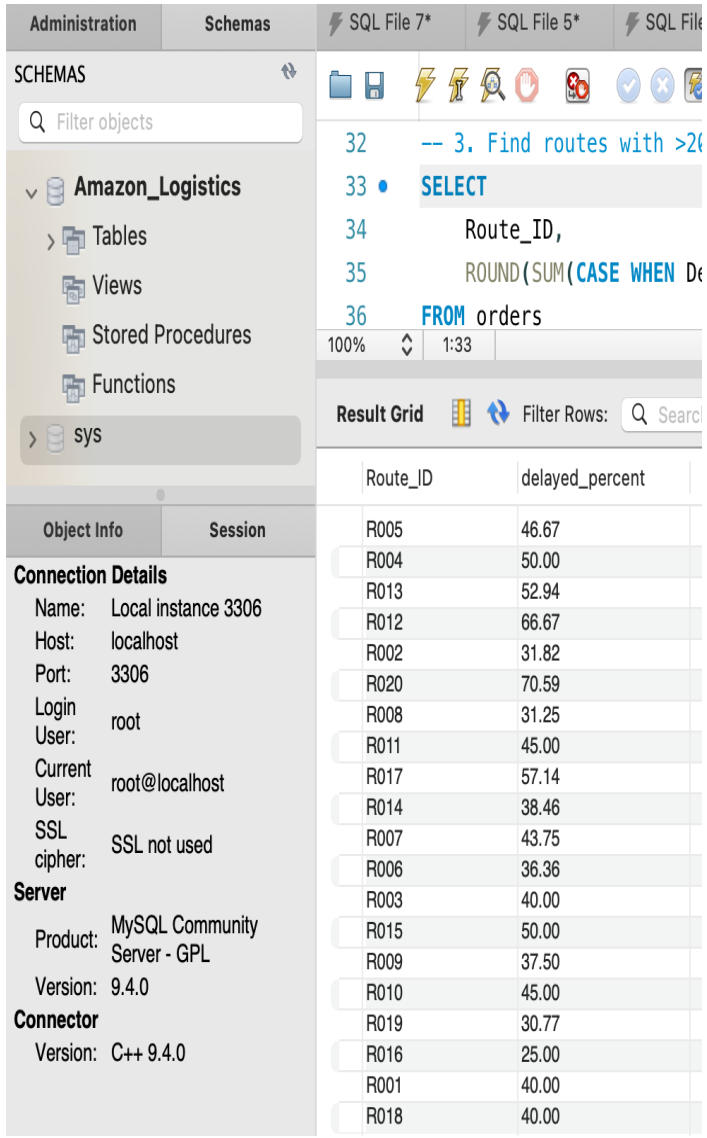
Schedule R007 (Aaronhaven to Jeanneberg) deliveries during off-peak hours to reduce 51-minute traffic delay and 48.75% delays.

Reroute R009 (Bellfurt to Lake Marvin) via less congested paths to address 46-minute traffic delay and 50.00% delays.

Assign faster vehicles to R006 (Lake Robinhaven to East Christopherville) for its 0.1607 efficiency ratio and 40.00% delays.


Administration
Schemas
SQL File 7*
SQL File 5*
SQL File 6*
SCHEMAS
Filter objects
Amazon_Logistics
Tables
Views
Stored Procedures
Functions
sys
-- 1.For each route, calculat
Average delivery time (in d
SELECT Route_ID, AVG(DATEDIFF
FROM orders
GROUP BY Route_ID;
100%
1:9
Result Grid
Filter Rows:
Search
Route_ID
efficiency_ratio
R001 0.6136792452830189
R002 2.156179775280899
R003 0.5815789473684211
R004 4.278125
R005 0.37382198952879586
R006 0.16065573770491803
R007 0.28250950570342204
R008 0.24372093023255814
R009 0.6047846889952153
R010 0.6686046511627907
R011 0.9556179775280899
R012 0.9036363636363637
R013 0.9324999999999999
R014 2.2781818181818183
R015 0.30086956521739133
R016 0.15120481927710844
R017 0.328
R018 0.5366037735849056
R019 3.7693877551020405
R020 0.5502923976608187
Object Info
Session
Connection Details
Name: Local instance 3306
Host: localhost
Port: 3306
Login User: root
Current User: root@localhost
SSL cipher: SSL not used
Server
Product: MySQL Community Server - GPL
Version: 9.4.0
Connector
Version: C++ 9.4.0


Administration
Schemas
SQL File 7*
SQL File 5*
SQL File 6*
SCHEMAS
Filter objects
Amazon_Logistics
Tables
Views
Stored Procedures
Functions
sys
-- 2.Identify 3 routes with t
(Assuming worst = lowest ef
SELECT Route_ID, Distance_KM
FROM routes
ORDER BY efficiency_ratio ASC
100%
1:25
Result Grid
Filter Rows:
Search
Route_ID
efficiency_ratio
R016 0.15120481927710844
R006 0.16065573770491803
R008 0.24372093023255814
Object Info
Session
Connection Details
Name: Local instance 3306
Host: localhost
Port: 3306
Login User: root
Current User: root@localhost
SSL cipher: SSL not used
Server
Product: MySQL Community Server - GPL
Version: 9.4.0
Connector
Version: C++ 9.4.0


Administration
Schemas
SQL File 7*
SQL File 5*
SQL File
SCHEMAS
Filter objects
Amazon_Logistics
Tables
Views
Stored Procedures
Functions
sys
-- 3. Find routes with >2
SELECT
Route_ID,
ROUND(SUM(CASE WHEN De
FROM orders
100%
1:33
Result Grid
Filter Rows:
Route_ID
delayed_percent
R005 46.67
R004 50.00
R013 52.94
R012 66.67
R002 31.82
R020 70.59
R008 31.25
R011 45.00
R017 57.14
R014 38.46
R007 43.75
R006 36.36
R003 40.00
R015 50.00
R009 37.50
R010 45.00
R019 30.77
R016 25.00
R001 40.00
R018 40.00
Object Info
Session
Connection Details
Name: Local instance 3306
Host: localhost
Port: 3306
Login User: root
Current User: root@localhost
SSL cipher: SSL not used
Server
Product: MySQL Community Server - GPL
Version: 9.4.0
Connector
Version: C++ 9.4.0

Task 4: Warehouse Performance

```
-- 1. Find the top 3 warehouses with the highest average processing time.
SELECT Warehouse_ID, ROUND(AVG(Processing_Time_Min),2) AS avg_processing_time
FROM warehouses
GROUP BY Warehouse_ID
ORDER BY avg_processing_time DESC
LIMIT 3;
-- 2. Calculate total vs. delayed shipments for each warehouse.
SELECT Warehouse_ID, COUNT(*) AS total_shipments, SUM(CASE WHEN Delivery_Status = 'Delayed' THEN 1 ELSE 0 END) AS delayed_shipments
FROM orders
GROUP BY Warehouse_ID;
-- 3. Use CTEs to find bottleneck warehouses where processing time > global average.
WITH global_avg AS (SELECT AVG(Processing_Time_Min) AS avg_proc FROM warehouses)
SELECT w.Warehouse_ID, w.Processing_Time_Min
FROM warehouses w, global_avg g
WHERE w.Processing_Time_Min > g.avg_proc;
-- 4. Rank warehouses based on on-time delivery percentage.
SELECT Warehouse_ID,
    ROUND(((COUNT(*) - SUM(CASE WHEN Delivery_Status = 'Delayed' THEN 1 ELSE 0 END)) * 100.0 / COUNT(*)),2) AS on_time_percent,
    RANK() OVER (ORDER BY ((COUNT(*) - SUM(CASE WHEN Delivery_Status = 'Delayed' THEN 1 ELSE 0 END)) * 100.0 / COUNT(*)) DESC) AS on_time_percent_rank
FROM orders
GROUP BY Warehouse_ID;
```

Insights:

- **W010, W009, and W007** recorded the **highest processing times** (171, 170, 166 mins).
- **W010 (67% delayed)** and **W001 (53% delayed)** are the **worst-performing warehouses**.
- **W003 (57% delayed)** and **W005 (56% delayed)** also showed severe inefficiencies.
- **Bottlenecks:** W001, W007, W008, W009, W010 all exceed global average processing times.
- **Best performers: W008 (76% on-time)** and **W006 (68% on-time)** lead in reliability.
- **Worst performer: W010 (33% on-time)** ranks last in overall delivery performance.

SQL File 7* | SQL File 5* | SQL File 6* | SQL File 8*

Limit to 1000 rows

```
1
2
3   -- 1. Find the top 3 warehouses with the high
4   SELECT Warehouse_ID, ROUND(AVG(Processing_Tim
5   FROM warehouses
6   GROUP BY Warehouse_ID
```

100% 31:2

Result Grid | Filter Rows: Search | Export:

Warehouse_ID	avg_processing_ti...
W010	171.00
W009	170.00
W007	166.00

SQL File 7* | SQL File 5* | SQL File 6* | SQL File 8*

Limit to 1000 rows

```
13
14  -- 2. Calculate total vs. delayed shipments f
15  SELECT Warehouse_ID, COUNT(*) AS total_shipme
16  FROM orders
17  GROUP BY Warehouse_ID;
```

100% 1:13

Result Grid | Filter Rows: Search | Export:

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

SQL File 7* | SQL File 5* | SQL File 6* | SQL File 8*

Limit to 1000 rows

```
19
20  -- 3. Use CTEs to find bottleneck warehouses
21  WITH global_avg AS (SELECT AVG(Processing_Tim
22  SELECT w.Warehouse_ID, w.Processing_Time_Min
23  FROM warehouses w, global_avg g
24  WHERE w.Processing_Time_Min > g.avg_proc;
25
```

100% 23:17

Result Grid | Filter Rows: Search | Export:

Warehouse_ID	Processing_Time_Min
W001	142
W007	166
W008	161
W009	170
W010	171

SQL File 7* | SQL File 5* | SQL File 6* | SQL File 8*

Limit to 1000 rows

```
    -- 4. Rank warehouses based on on-time delivery
29  SELECT Warehouse_ID,
30         ROUND(((COUNT(*) - SUM(CASE WHEN Deliver
31         RANK() OVER (ORDER BY ((COUNT(*) - SUM(C
32  FROM orders
33  GROUP BY Warehouse_ID;
34
```

100% 22:29

Result Grid | Filter Rows: Search | Export:

Warehouse_ID	on_time_percent	on_time_percent_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

```
-- 1. Rank agents (per route) by on-time delivery percentage.
SELECT Agent_ID, Route_ID, On_Time_Percentage,
RANK() OVER (PARTITION BY Route_ID
ORDER BY On_Time_Percentage DESC) AS On_Time_Percentage_rank
FROM deliveryagents;

-- 2. Find agents with on-time % < 80%.
SELECT Agent_ID, Route_ID, On_Time_Percentage
FROM deliveryagents
WHERE On_Time_Percentage < 80;

-- 3. Compare average speed of top 5 vs bottom 5 agents using subqueries.
SELECT
  AVG(top5.Avg_Speed_KM_HR) AS top5_avg_speed,
  AVG(bottom5.Avg_Speed_KM_HR) AS bottom5_avg_speed
FROM (
  SELECT Avg_Speed_KM_HR
  FROM deliveryagents
  ORDER BY On_Time_Percentage DESC
  LIMIT 5
) top5,
(
  SELECT Avg_Speed_KM_HR
  FROM deliveryagents
  ORDER BY On_Time_Percentage ASC
  LIMIT 5
) bottom5;
```

Insights:

- **A014 (99.9%), A007 (99.11%), and A004 (98.25%)** are the **best-performing agents** across routes.
- **R002, R012, R017, and R018** have multiple agents consistently delivering **>90% on-time**.
- **A010 (70.45%), A038 (71.8%), A011 (71.5%)** and others fall **below 80%**, marking them as underperformers.
- Nearly **40% of agents have <80% on-time rates**, showing widespread performance gaps.
- **Top 5 agents average 60 km/hr speed**, compared to **56.8 km/hr** for bottom 5, linking speed with punctuality.
- Training, route optimization, and performance monitoring are needed to improve **low-performing agents**.

Result Grid | Filter Rows: Search | Export:

Agent_ID	Route_ID	On_Time_Percentage	On_Time_Percentage_rank
A006	R001	81.56	1
A037	R001	78.94	2
A033	R002	98.01	1
A024	R002	97.16	2
A027	R002	73.8	3
A050	R002	72.75	4
A011	R002	71.5	5
A015	R003	83.31	1
A029	R005	81.71	1
A035	R005	74.5	2
A041	R006	91.47	1
A047	R006	84.62	2
A020	R006	77.65	3
A014	R007	99.9	1
A049	R007	99.82	2
A040	R007	77.83	3
A028	R007	74.47	4
A018	R007	73.21	5
A032	R008	99.32	1
A022	R008	87.72	2
A034	R008	81.67	3
A045	R009	95.08	1
A042	R010	93.37	1
A012	R010	80.85	2
A009	R012	92.27	1

Result Grid | Filter Rows: Search

Agent_ID	Route_ID	On_Time_Percentage
A003	R014	74.15
A008	R018	72.39
A010	R019	70.45
A011	R002	71.5
A016	R012	78.39
A018	R007	73.21
A020	R006	77.65
A023	R017	74.21
A027	R002	73.8
A028	R007	74.47
A030	R014	76.6
A035	R005	74.5
A037	R001	78.94
A038	R018	71.8
A039	R016	72.6
A040	R007	77.83
A043	R016	79.51
A046	R020	75.46
A048	R012	77.92
A050	R002	72.75

100% | 11:36

Result Grid | Filter Rows: Search | Export:

top5_avg_speed	bottom5_avg_speed
60.0000	56.8000

Task 6: Shipment Tracking Analytics

```
-- 1. For each order, list the last checkpoint and time.
SELECT s.Order_ID, s.Checkpoint, s.Checkpoint_Time
FROM shipment_tracking s
WHERE s.Checkpoint_Time = (SELECT MAX(Checkpoint_Time)
FROM shipment_tracking
WHERE Order_ID = s.Order_ID);

-- 2. Find the most common delay reasons (excluding None).
SELECT Delay_Reason, COUNT(*) AS count
FROM shipment_tracking
WHERE Delay_Reason != 'None'
GROUP BY Delay_Reason
ORDER BY count DESC;

-- 3. Identify orders with >2 delayed checkpoints.
SELECT Order_ID, COUNT(*) AS delayed_checkpoints
FROM shipment_tracking
WHERE Delay_Reason != 'None'
GROUP BY Order_ID
HAVING delayed_checkpoints > 2;
```

Insights:

- **Sorting Delay (272 cases)** is the **biggest cause of shipment delays**, followed by **Weather (263)** and **Traffic (247)**.
- Over **100+ orders had >2 delayed checkpoints**, showing repeated disruptions in delivery flow.
- Orders like **O0002, O0003, O0004, O0005** faced **3–5 delays each**, making them critical problem cases.
- **Final checkpoint visibility is intact** — most shipments are tracked up to their last checkpoint.
- **Recurring delays** highlight bottlenecks in **sorting centers and route congestion**, needing urgent optimization.

100% 1:2

Result Grid Filter Rows: Search Export:

Order_ID	Checkpoint	Checkpoint_Time
O0001	Checkpoint 1	2025-06-09
O0001	Checkpoint 2	2025-06-09
O0002	Checkpoint 3	2025-06-23
O0002	Checkpoint 4	2025-06-23
O0003	Checkpoint 2	2025-06-05
O0003	Checkpoint 3	2025-06-05
O0003	Checkpoint 4	2025-06-05
O0004	Checkpoint 2	2025-06-29
O0004	Checkpoint 3	2025-06-29
O0004	Checkpoint 4	2025-06-29
O0004	Checkpoint 5	2025-06-29
O0005	Checkpoint 1	2025-06-04
O0005	Checkpoint 2	2025-06-04
O0005	Checkpoint 3	2025-06-04
O0005	Checkpoint 4	2025-06-04
O0005	Checkpoint 5	2025-06-04
O0006	Checkpoint 1	2025-07-18
O0006	Checkpoint 2	2025-07-18
O0007	Checkpoint 1	2025-06-15
O0007	Checkpoint 2	2025-06-15
O0007	Checkpoint 3	2025-06-15
O0008	Checkpoint 2	2025-06-24
O0009	Checkpoint 1	2025-07-11
O0009	Checkpoint 2	2025-07-11
O0009	Checkpoint 3	2025-07-11

100% 51:15

Result Grid Filter Rows: Search

Order_ID	delayed_checkpoi...
O0002	3
O0003	4
O0004	4
O0005	4
O0007	3
O0015	4
O0016	5
O0019	5
O0021	4
O0024	4
O0025	3
O0027	3
O0030	3
O0033	3
O0038	5
O0041	4
O0046	3
O0048	4
O0052	3
O0053	4
O0055	4
O0057	3
O0058	3
O0060	3
O0061	3
O0064	4

Result 11

100% 1:13

Result Grid Filter Rows: Search Export:

Delay_Reason	count
Sorting Delay	272
Weather	263
Traffic	247

Task 7: Advanced KPI Reporting

```
-- 1. Average Delivery Delay per Region (Start_Location).
SELECT r.Start_Location, AVG(DATEDIFF(o.Actual_Delivery_Date, o.Expected_Delivery_Date)) AS avg_delay
FROM orders o
LEFT JOIN routes r ON o.Route_ID = r.Route_ID
GROUP BY r.Start_Location;

-- 2. On-Time Delivery % = (Total On-Time Deliveries / Total Deliveries) * 100.
SELECT
  (SUM(CASE WHEN Delivery_Status = 'On Time' THEN 1 ELSE 0 END) / COUNT(*) * 100)
  AS on_time_percent
FROM orders;

-- 3. Average Traffic Delay per Route.
SELECT Route_ID,
     AVG(Traffic_Delay_Min) AS avg_traffic_delay
FROM routes
GROUP BY Route_ID;
```

Insights:

- Regions like **South Davidborough (0.44 avg delay)** and **Lake Kathrynchester (0.42 avg delay)** show the **least delivery delays**, while **West Calvinview (1.08)** and **Juliestad (1.05)** face the **highest delays**, making them critical focus areas.
- The overall **On-Time Delivery % is 56%**, highlighting that **nearly half of deliveries are delayed**, signaling operational inefficiency.
- **Route R014 (58 min)** and **R007 (51 min)** have the **highest average traffic delays**, indicating frequent congestion issues.
- In contrast, **Routes R004 (3 min)** and **R018/R003 (11 min each)** show **minimal traffic delays**, serving as benchmarks for efficiency.
- Traffic bottlenecks in **R007, R014, and R009** should be prioritized for rerouting or scheduling changes to improve delivery efficiency.

100% 15:6

Result Grid Filter Rows: Search

Start_Location	avg_delay
New Jessica	0.8000
Ramosside	0.8000
South Johnville	0.7647
West Calvinview	1.0833
South Brianchester	0.5000
Juliestad	1.0588
South Davidborough	0.4375
Lowemouth	0.7000
Port Tammybury	1.0000
North Edward	0.6154
Aaronhaven	0.6875
Lake Robinhaven	0.5455
East Amy	0.5000
South Jenniferfurt	1.0000
Bellfurt	0.5000
North Aliceside	0.7000
Lake Oscarfurt	0.6154
Lake Kathrynchester	0.4167
New Aimeechester	0.6667
Lake Jessicastad	0.4667

100% 1:27

Result Grid Filter Rows: Search

Route_ID	avg_traffic_del...
R001	36.0000
R002	22.0000
R003	11.0000
R004	3.0000
R005	26.0000
R006	42.0000
R007	51.0000
R008	34.0000
R009	46.0000
R010	40.0000
R011	25.0000
R012	29.0000
R013	38.0000
R014	58.0000
R015	29.0000
R016	8.0000
R017	8.0000
R018	11.0000
R019	37.0000
R020	23.0000

100% 1:9

Result Grid Filter Rows: Search Export:

on_time_percent
56.0000

Task 8: PPT Presentation and Video Submission

Video Link :

https://drive.google.com/file/d/1dG63Qymg4oibHRyr844_87RILQJwdDUP/view?usp=sharing

Thank You