LinkedIn

Optimizing Supply Chain Management

Start Now

By : Harsha_Kachavrapu

Executive Summary

Enhancing Supply Chain Efficiency with Data-Driven Insights

This project utilizes **SQL**, **Python**, and **Excel** to analyze supply chain operations, identify inefficiencies, and optimize logistics. The analysis covers **customer demand trends**, **inventory management**, **cost reduction**, and **performance evaluation**.

Key Phases of the Analysis

01

Orders & Customer analysis

Identify ordering trends and customer behavior for better demand forecasting.

02

Inventory & Demand Analysis

Monitor stock levels, detect shortages, and optimize replenishment strategies.

03

Freight & Cost Optimization

Analyze transportation costs and find cost-effective shipping routes.

04

Performance & Logistics

Evaluate supply chain efficiency by tracking delivery performance and bottlenecks.

1. Orders & Customers Analysis

Top Ordered Products

```sql
-- Identify the top 5 products with the highest order volume.

WITH cte AS (SELECT
                    product_id, COUNT(1) AS Total_orders_per_product, rank() over(order by count(1) desc) as rnk
             FROM
                 order_details
             GROUP BY product_id)

SELECT
    cte.rnk, cte.product_id, cte.Total_orders_per_product
FROM
    cte
WHERE
    rnk BETWEEN 1 AND 5;
```

Which products are ordered the most based on total quantity?

Takeaway: Identifying high-demand products ensures better stock management.

Result Grid | Filter Rows: | Expo

rnk	product_id	Total_orders_per_product
1	1689547	192
2	1677878	140
3	1689548	133
4	1689546	129
5	1688571	120

Average Order Size

```sql
-- Calculate the average order quantity per customer and
-- categorize them into small, medium, and large order sizes.
WITH cte AS (
    SELECT
        customer_id,
        ROUND(SUM(unit_quantity) / COUNT(order_id), 0) AS avg_order_quan_per_cx,
        NTILE(3) OVER (ORDER BY ROUND(SUM(unit_quantity) / COUNT(order_id), 0) ASC) AS size
    FROM order_details
    GROUP BY customer_id
)
SELECT
    customer_id,
    avg_order_quan_per_cx,
    CASE
        WHEN size = 1 THEN 'Small'
        WHEN size = 2 THEN 'Medium'
        WHEN size = 3 THEN 'Large'
    END AS category
FROM cte;
```

Result Grid | Filter Rows: | Export: | Wrap C

customer_id	avg_order_quan_per_cx	category
V555555_34	641	Small
V55_37	727	Small
V555555_24	769	Small
V5555555_22	837	Small
V555_15	897	Medium
V55555555555555_8	899	Medium
V5555555_12	945	Medium
V55555_4	1147	Medium
V555555555_27	1345	Medium
V55_47	1362	Medium
V55555555_7	1500	Medium
V555555_6	1602	Medium
V55555_53	1674	Medium
V555555555555555_44	1681	Medium
V5555_20	1718	Medium
V5555555_30	1839	Medium
V5555555_19	1876	Medium
V5555_25	1922	Medium
V55555_2	1995	Medium
V555555555555555555_17	2277	Large

What is the average order quantity per customer, and how do orders vary in size?

Takeaway: Segmenting customers by order size helps in demand forecasting.

```sql
-- Determine how each product's demand varies across different customers.

SELECT
    customer_id,
    product_id,
    SUM(unit_quantity) AS total_orders_per_product
FROM
    order_details
GROUP BY product_id , customer_id
ORDER BY total_orders_per_product DESC;
```

Order Trends by Product

How does product demand vary across different customers?

Takeaway: Understanding order patterns helps tailor inventory stocking.

Result Grid | Filter Rows: | Export:

customer_id	product_id	total_orders_per_product
V55555555_5	1684862	3216746
V55555555555_28	1676592	1119252
V55555_2	1664051	859885
V55555555555_28	1667927	814076
V55555555_5	1700569	791030
V55555_26	1683560	772283
V55555555555_28	1666524	670437
V55555555_5	1700130	639219
V55555555_5	1700143	634631

2. Inventory & Demand Analysis

Stock Availability Check

```sql
-- Stock Availability Check:
-- Compare ordered quantities with available stock per plant and
-- identify cases where demand exceeds supply.

SELECT
    od.plant_code, COUNT(1) AS num_of_orders, wh.order_capacity
FROM
    order_details od
        JOIN
    whcapacities wh ON od.plant_code = wh.plant_code
GROUP BY plant_code , Wh.order_capacity
HAVING num_of_orders > wh.order_capacity;
```

Are there cases where demand exceeds available stock?

Takeaway: Helps avoid stockouts and optimize replenishment strategies.

Result Grid | Filter Rows:

plant_code	num_of_orders	order_capacity
PLANT03	8541	1013
PLANT08	102	14
PLANT12	300	209
PLANT09	12	11

Product Demand per Plant

```sql
-- Product Demand per Plant: Identify the top 5 most ordered products from each plant.
WITH cte AS (SELECT
                od.product_id,
                COUNT(od.product_id) AS orders_per_product,
                pp.plant_code,
                RANK() OVER (PARTITION BY pp.plant_code ORDER BY COUNT(od.product_id) DESC) AS rnk
            FROM
                order_details od
                    JOIN
                products_per_plant pp ON od.product_id = pp.product_id
            GROUP BY pp.plant_code , od.product_id)

SELECT
    cte.plant_code,
    cte.product_id,
    cte.orders_per_product,
    cte.rnk
FROM
    cte
WHERE
    cte.rnk BETWEEN 1 AND 5;
```

Result Grid | Filter Rows: | Export:

plant_code	product_id	orders_per_product	rnk
PLANT01	1690144	8	1
PLANT01	1674378	1	2
PLANT01	1674380	1	2
PLANT02	1689547	192	1
PLANT02	1689548	133	2
PLANT02	1689546	129	3
PLANT02	1688571	120	4
PLANT02	1688629	119	5
PLANT03	1689547	192	1
PLANT03	1677878	140	2
PLANT03	1689548	133	3
PLANT03	1689546	129	4
PLANT03	1688571	120	5
PLANT04	1696074	49	1
PLANT04	1699948	46	2
PLANT04	1699952	42	3
PLANT04	1695348	34	4

Which products are most ordered from each plant?

Takeaway: Helps optimize production and distribution planning.

Least Ordered Products

```sql
-- Least Ordered Products: Identify the least ordered products per plant to detect slow-moving inventory.
WITH cte AS (SELECT
                od.product_id,
                COUNT(od.product_id) AS orders_per_product,
                pp.plant_code,
                RANK() OVER (PARTITION BY pp.plant_code ORDER BY COUNT(od.product_id) ASC) AS rnk
            FROM
                order_details od
                    JOIN
                products_per_plant pp ON od.product_id = pp.product_id
            GROUP BY pp.plant_code , od.product_id)

SELECT
    cte.plant_code,
    cte.product_id,
    cte.orders_per_product,
    cte.rnk
FROM
    cte
WHERE
    cte.rnk = 1;
```

Result Grid | Filter Rows:

plant_code	product_id	orders_per_product
PLANT01	1674378	1
PLANT01	1674380	1
PLANT02	1689783	1
PLANT02	1691546	1
PLANT02	1683401	1
PLANT02	1688621	1
PLANT03	1654251	1
PLANT03	1689783	1
PLANT03	1683401	1

Which products are slow-moving per plant?

Takeaway: Helps optimize storage and reduce waste.

```sql
-- Plant Demand Analysis:
-- Identify which plants fulfill the highest number of unique product orders and
-- analyze their contribution to overall supply chain efficiency.

WITH PlantProductOrders AS (
    SELECT
        plant_code,
        COUNT(DISTINCT product_id) AS unique_products_fulfilled,
        COUNT(order_id) AS total_orders
    FROM order_details
    GROUP BY plant_code
)
SELECT
    plant_code,
    total_orders,
    unique_products_fulfilled,
    ROUND((unique_products_fulfilled / SUM(unique_products_fulfilled) OVER()) *100, 2) AS contribution_percentage
FROM PlantProductOrders
ORDER BY unique_products_fulfilled DESC;
```

Plant Demand Analysis

Which plants fulfill the highest number of unique product orders?

Takeaway: Helps assess plant contribution to supply chain efficiency.

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

plant_code	total_orders	unique_products_fulfilled	contribution_percentage
PLANT03	8541	651	83.46
PLANT12	300	57	7.31
PLANT13	86	31	3.97
PLANT16	173	22	2.82
PLANT08	102	14	1.79
PLANT09	12	4	0.51
PLANT04	1	1	0.13

3.Freight & Cost Optimization

Shipping Cost Analysis

```sql
-- Shipping Cost Analysis: Calculate the average freight cost per order for each warehouse.
WITH OrderShipping AS (
    SELECT
        od.order_id,
        od.plant_code AS warehouse,
        pp.port_id AS origin_port,
        od.destination_port,
        od.weight,
        fr.min_cost,
        fr.cost_per_unit_weight,
        (fr.min_cost + (od.weight * fr.cost_per_unit_weight)) AS total_freight_cost
    FROM order_details od
    JOIN plant_ports pp
        ON od.plant_code = pp.plant_code
    JOIN freightrates fr
        ON pp.port_id = fr.origin_port
        AND od.destination_port = fr.destination_port
        AND od.weight BETWEEN fr.min_weight AND fr.max_weight
)
SELECT
    warehouse,
    ROUND(AVG(total_freight_cost), 2) AS avg_freight_cost_per_order
FROM OrderShipping
GROUP BY warehouse
ORDER BY avg_freight_cost_per_order DESC;
```

Result Grid | Filter Rows:

warehouse	avg_freight_cost_per_order
PLANT16	1592.12
PLANT09	24.49
PLANT13	22.02
PLANT08	17.12
PLANT12	17.08
PLANT03	15.76
PLANT04	6.49

What is the average freight cost per order for each warehouse?

Takeaway: Helps identify cost-efficient vs. high-cost shipping centers.

```sql
-- Most Expensive Routes: Find the top 3 most expensive freight routes based on historical data.
SELECT
    origin_port,
    destination_port,
    (ROUND(SUM(min_cost + (cost_per_unit_weight * min_weight)),2)) AS cost_per_route
FROM
    freightrates
GROUP BY origin_port , destination_port
ORDER BY cost_per_route DESC
LIMIT 3;
```

Most Expensive Routes

Which are the top 3 most expensive freight routes?

Takeaway: Helps optimize route selection for cost savings.

Result Grid | Filter Rows:

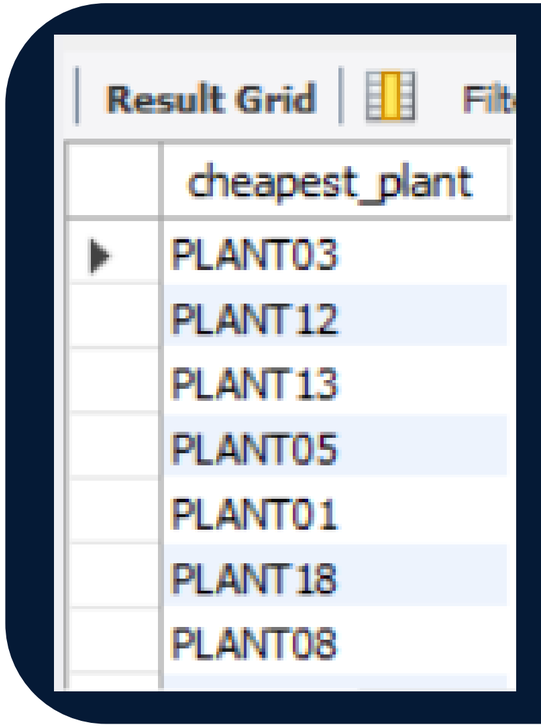
origin_port	destination_port	cost_per_route
PORT06	PORT09	63107.67
PORT10	PORT09	56961.29
PORT05	PORT09	35587.41

Optimal Plant Selection

```sql
-- Most Expensive Routes: Find the top 3 most expensive freight routes based on historical data.
SELECT
    origin_port,
    destination_port,
    (ROUND(SUM(min_cost + (cost_per_unit_weight * min_weight)),2)) AS cost_per_route
FROM
    freightrates
GROUP BY origin_port , destination_port
ORDER BY cost_per_route DESC
LIMIT 3;
```

Which plant can fulfill an order at the lowest freight cost?

Takeaway: Helps reduce overall supply chain expenses.



4. Performance & Logistics

```sql
-- Late Shipment Trends: Analyze which plants
-- or warehouses cause the most shipping delays.

SELECT
    plant_code AS Warehouse,
    COUNT(ship_late_day_count) AS Total_count_of_delays
FROM
    order_details
WHERE
    ship_late_day_count >= 1
GROUP BY plant_code
ORDER BY COUNT(ship_late_day_count) DESC;
```

Late Shipment Trends

Which plants or warehouses cause the most shipping delays?

Takeaway: Helps improve supply chain reliability.

Result Grid | Filter Rows:

Warehouse	Total_count_of_delays
PLANT03	174
PLANT08	18

Best Performing Plant

```sql
-- Best Performing Plant:
-- Identify the top 3 plants with the highest on-time delivery rate.

SELECT
    plant_code, COUNT(*) AS total_count
FROM
    order_details
WHERE
    ship_late_day_count = 0
GROUP BY plant_code
LIMIT 3;
```

Which plant has the highest on-time delivery rate?

Takeaway: Helps benchmark efficiency across plants.

Result Grid | Filter Rows:

plant_code	total_count
PLANT03	8367
PLANT16	173
PLANT12	300

Order Weight Distribution

```sql
-- Order Weight Distribution:
-- Analyze the distribution of order weights and
--  identify the most common weight range for shipments.

SELECT
    CASE
        WHEN weight*unit_quantity BETWEEN 0 AND 50 THEN '0-50 kg'
        WHEN weight*unit_quantity BETWEEN 51 AND 100 THEN '51-100 kg'
        WHEN weight*unit_quantity BETWEEN 101 AND 200 THEN '101-200 kg'
        WHEN weight*unit_quantity BETWEEN 201 AND 500 THEN '201-500 kg'
        ELSE '500+ kg'
    END AS weight_range,
    COUNT(*) AS order_count
FROM order_details
GROUP BY weight_range
ORDER BY order_count DESC;
```

Result Grid | Filter Rows:

weight_range	order_count
500+ kg	6959
201-500 kg	1201
0-50 kg	1055

What are the most common weight ranges for shipments?

Takeaway: Helps optimize packaging and freight cost estimation.

Key Takeaways

Data-Driven Supply Chain Optimization

Optimized Inventory Management

Balanced stock levels to prevent shortages and overstocking.

Reduced Freight Costs

Identified cost-effective routes, leading to significant savings.

Improved Decision-Making

Leveraged real-time data for actionable insights.

Enhanced Supply Chain Efficiency

Reduced delivery delays and optimized operations.

LinkedIn

Thank You For Attention

See You Next

By: Harsha_Kachavarapu