**CASE STUDY DOCUMENTATION**

TechElectro Inc. faces a series of intricate inventory management challenges that impede its operational efficiency and customer satisfaction: The company frequently finds itself burdened with excessive inventory of certain products, resulting in substantial capital tied up in unsold goods and limited storage capacity. Conversely, high-demand products regularly suffer from stockouts, leading to missed sales opportunities and irate customers unable to access their desired items. These inventory-related issues have a direct and detrimental effect on customer satisfaction and loyalty. Customers endure delays, frequent stockouts, and frustration when they cannot find the products they seek.

**Report Description for this Case Study**

1. A clear statement of the business task2. A description of all data sources used3. Documentation of any cleaning or manipulation of data4. A summary of your analysis5. Your top three recommendations based on your analysis

**ASK**

Three questions will guide the future marketing program:

1. How can TechElectro Inc. leverage MySQL-powered inventory optimization to reduce carrying costs and free up capital for strategic investments?2. In what ways does maintaining optimal inventory levels through MySQL optimization enhance customer satisfaction and foster loyalty at TechElectro Inc.?3. What are the specific competitive advantages TechElectro Inc. can gain by implementing streamlined inventory management and responding swiftly to market fluctuations using MySQL optimization?

**1. ANALYTICAL REPORT FOR THE FIRST QUESTION**

Step1: Ask

**Answers to Guiding Questions** – Using the Structured Query Language will greatly help in finding solutions that will optimize the inventory of goods using historical data. The deduced trends will be discovered at the end of the analysis. A recommendation will be stipulated out to drive efficiency and increase productivity.

**Business Task** – The primary objectives of this project are to implement a sophisticated inventory optimization system utilizing MySQL and address the identified business challenges effectively. The project aims to achieve the following goals:A. Optimal Inventory Levels: Utilize MySQL optimization techniques to determine the optimal stock levels for each product SKU, thereby minimizing overstock and understock situations.B. Data-Driven Decisions: Enable data-driven decision-making in inventory management by leveraging MySQL analytics to reduce costs and enhance customer satisfaction.

**Key Stakeholders** – (i) Executive Team

**Objective Statement:**

(i) Thoroughly read through the brief description of the company given in this Case Study. It was deduced from historical inventory data that there is unbalancing acts in stocks which may either be excessive or insufficient. Insufficiency in stock may lead to dissatisfied customers.

**The rationale of the project:**

* Cost of reduction
* Enhance customer satisfaction
* Competitive Advantage
* Profitability

Step 2: Prepare

**Answers to Guiding Questions** – The data was obtained from the Index of bucket "divvy-trip data" an HTML file. The data is organized into datasets designated for each month in a zip file. There are no issues with biasing or credibility because this data source is reliable, original, comprehensive, current and cited. Also, the licensing, privacy, security and accessibility of the data source were checked and will be adhered to strictly. I verified the data’s integrity by ensuring the reliability of the source (Motivate International Inc.) This provides the datasets on where I will be able to run my analysis. After careful consideration, there are no problems with this data.

**Key Tasks to be Done** - Download the data and store it appropriately. Identify how it’s organized. Sort and filter the data. Determine the credibility of the data.

**Deliverable:** All data collected are presented in CSV format.

Step 3: Process

**Answers to Guiding Questions** –

* The analytical tool I will use throughout is SQL (This is because it was the stipulated tool advised to be used to determine and achieve optimization of stock).
* I have ensured data integrity earlier in step 2. It is from a reliable source.
* I have ensured all rows of empty cells found in each table are cleared by filtering out empty cells in MySQL and deleting all. I also ensured there weren’t any form of duplicates in each table of data.
* To ensure my data is clean and ready for analysis I double-check all, my cleaning processes one step at a time by doing exact same cleaning steps.
* This is my documentation of my cleaning process done right here.

Step 4: Analyze

**Answers to Guiding Questions** –

-To organize my data before analysis, I ensured the values in each column were clean and consistent before importing the datasets in my MySQL server using xamp.

-I have ensured my data has been properly formatted using my simple query techniques.

**Method of Analysis**

1. **Applying SQL for Cleaning and Preparing Data**

(i) Data Cleanings

ALTER TABLE sales\_data\_\_1\_

ADD New\_Sales\_Date DATE;

SET SQL\_SAFE\_UPDAES = 0;

UPDATE sales\_data\_\_1\_

SET New\_Sales\_Date = STR\_TO\_Date(Sales Date, %d%m%Y);

ALTER TABLE sales\_data\_\_1\_

DROP Sales Date;

ALTER TABLE sales\_data\_\_1\_

CHANGE COLUMN New\_Sales\_Date Sales Date DATE;

(iv) Missing Data

-- Identify missing values using 'IS NULL' function

-- external factor

SELECT

SUM(CASE WHEN Sales\_Date is NULL THEN 1 ELSE 0 END) AS missing\_sales\_date,

SUM(CASE WHEN GDP is NULL THEN 1 ELSE 0 END) AS missing\_gdp,

SUM(CASE WHEN Inflation\_Rate is NULL THEN 1 ELSE 0 END) AS missing\_inflation\_rate,

SUM(CASE WHEN Seasonal\_Factor is NULL THEN 1 ELSE 0 END) AS missing\_seasonal\_factor

FROM external\_factors;

(v) Duplicates Cleanings

External\_factors

SELECT sales\_date, COUNT(\*) AS duplicate\_count

FROM external\_factors

GROUP BY sales\_date

HAVING COUNT(\*) > 1;

Product\_Information

SELECT Product\_ID, Product\_Category, COUNT(\*) AS duplicate\_count

FROM product\_information

GROUP BY Product\_ID, Product\_Category

HAVING COUNT(\*) > 1;

(vi) DATA INTEGRATION

-- sales\_data and product\_data first

CREATE VIEW sales\_product\_data AS

SELECT

s.Product\_ID,

s.Sales\_Date,

s.Inventory\_Quantity,

s.Product\_Cost,

p.Product\_Category,

p.Promotions

FROM sales\_data s

JOIN product\_information p ON s.Product\_ID = p.Product\_ID;

-- Sale\_product\_data and external\_factors

CREATE VIEW Inventory\_data AS

SELECT

sp.Product\_ID,

sp.Sales\_Date,

sp.Inventory\_Quantity,

sp.Product\_Cost,

sp.Product\_Category,

sp.Promotions,

e.GDP,

e.Inflation\_Rate,

e.Seasonal\_Factor

FROM sales\_product\_data sp

LEFT JOIN external\_factors e

ON sp.Sales\_Date = e.Sales\_Date;

**2. Descriptive Analysis**

(vii) Basic Statistics:

*-- Average Sales(Calculated as the product of "Inventory Quantity" and "Product\_Cost")*

*SELECT Product\_ID,*

*AVG(Inventory\_Quantity \* Product\_Cost) as avg\_sales*

*FROM Inventory\_data*

*GROUP BY Product\_ID*

*ORDER BY avg\_sales DESC;*

-- Median Stock Levels (i.e., "Inventory Quantity").

WITH RankedInventory AS (

SELECT

Product\_ID,

Inventory\_Quantity,

ROW\_NUMBER() OVER(PARTITION BY Product\_ID ORDER BY Inventory\_Quantity) AS row\_num\_asc,

ROW\_NUMBER() OVER(PARTITION BY Product\_ID ORDER BY Inventory\_Quantity DESC) AS row\_num\_desc,

COUNT(\*) OVER(PARTITION BY Product\_ID) AS count\_per\_product

FROM

inventory\_data

)

SELECT

Product\_ID,

AVG(Inventory\_Quantity) AS median\_stock

FROM

RankedInventory

WHERE

row\_num\_asc IN (

(count\_per\_product + 1) / 2,

(count\_per\_product + 2) / 2

)

GROUP BY

Product\_ID;

-- Product performance metrics (total sales per product).

SELECT Product\_ID,

ROUND(SUM(Inventory\_Quantity \* Product\_Cost)) as total\_sales

FROM inventory\_data

GROUP BY Product\_ID

ORDER BY total\_sales DESC;

-- Identify high-demand products based on average sales

WITH HighDemandProducts AS (

SELECT Product\_ID, AVG(Inventory\_Quantity) as avg\_sales

FROM inventory\_data

GROUP BY Product\_ID

HAVING avg\_sales > (

SELECT AVG(Inventory\_Quantity) \* 0.95 FROM sales\_data

)

)

-- Calculate stockout frequency for high-demand products

SELECT s.Product\_ID,

COUNT(\*) as stockout\_frequency

FROM inventory\_data s

WHERE s.Product\_ID IN (SELECT Product\_ID FROM HighDemandProducts)

AND s.Inventory\_Quantity = 0

GROUP BY s.Product\_ID;

Observation - Non of the high-demand product had any stockout.

GDP - It represents the overall economic health and growth of a nation. A higher GDP represents more customer spending leading to increased sales. A lower GDP represents an economic downfall potentially leading to decrease sales.

Inflation rate – It means how the general level of price of goods are increasing and purchasing power is falling.

(viii) INFLUENCE OF EXTERNAL FACTORS

-- GDP

SELECT Product\_ID,

AVG(CASE WHEN GDP > 0 THEN Inventory\_Quantity ELSE NULL END) AS avg\_sales\_positive\_gdp,

AVG(CASE WHEN GDP <= 0 THEN Inventory\_Quantity ELSE NULL END) AS avg\_sales\_negative\_gdp

FROM inventory\_data

GROUP BY Product\_ID

HAVING avg\_sales\_positive\_gdp IS NOT NULL;

-- INFLUENCE ON INFLATION RATE

SELECT Product\_ID,

AVG(CASE WHEN Inflation\_Rate > 0 THEN Inventory\_Quantity ELSE NULL END) AS avg\_sales\_positive\_inflation,

AVG(CASE WHEN Inflation\_Rate <= 0 THEN Inventory\_Quantity ELSE NULL END) AS avg\_sales\_negative\_inflation

FROM inventory\_data

GROUP BY Product\_ID

HAVING avg\_sales\_positive\_inflation IS NOT NULL;

Note: There are no form inflation rates for zero or negative average sales.

This shows how inflation rates influences the average sales of good.

-- OPTIMIZING INVENTORY

-- Determine the optimal reorder point for each product based on historical sales data and external factors.

-- Reorder Point= Lead Time Demand + Safety Stock

-- Leaad Time Demand = Rolling Average Sales x Lead Time

-- Safety Stock = Z x Lead Time root x Standard Deviation of Demand

-- Z=1.645

-- A constant lead time of 7 days for all productss.

-- We aim for a 95% service level.

WITH InventoryCalculation AS (

SELECT Product\_ID,

AVG(rolling\_avg\_sales) as avg\_rolling\_sales,

AVG(rolling\_variance) as avg\_rolling\_variance

FROM(

SELECT Product\_ID,

AVG(daily\_sales) OVER (PARTITION BY Product\_ID ORDER BY Sales\_Date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling\_avg\_sales,

AVG(squared\_diff) OVER (PARTITION BY Product\_ID ORDER BY Sales\_Date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling\_variance

FROM (

SELECT Product\_ID,

Sales\_Date, Inventory\_Quantity \* Product\_Cost as daily\_sales,

(Inventory\_Quantity \* Product\_Cost - AVG(Inventory\_Quantity \* Product\_Cost) OVER (PARTITION BY Product\_ID ORDER BY Sales\_Date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)) as squared

FROM inventory\_data

) subquery

) subquery2

GROUP BY Product\_ID

)

SELECT Product\_ID,

avg\_rolling\_sales \* 7 as Lead\_time\_demand,

1.645 \* (avg\_rolling\_variance \* 7) as safety\_stock,

(avg\_rolling\_sales \* 7) + (1.645 \* (average\_rolling\_variance \* 7 )) as reorder\_point

FROM InventoryCalculations;

-- Create the Inventory Optimization Table

CREATE TABLE inventory\_optimization (

Product\_ID INT, Reorder\_Point DOUBLE);

-- Create the Stored Procedure to Recalculate Reorder Point

DELIMITER //

CREATE PROCEDURE RecalculateRecorderPoint(productID INT)

BEGIN

DECLARE avgRollingSales DOUBLE;

DECLARE avgRollingVariance DOUBLE;

DECLARE leadTimeDemand DOUBLE;

DECLARE safetyStock DOUBLE;

DECLARE recorderPoint DOUBLE;

WITH InventoryCalculation AS (

SELECT Product\_ID,

AVG(rolling\_avg\_sales) as avg\_rolling\_sales,

AVG(rolling\_variance) as avg\_rolling\_variance

FROM(

SELECT Product\_ID,

AVG(daily\_sales) OVER (PARTITION BY Product\_ID ORDER BY Sales\_Date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling\_avg\_sales,

AVG(squared\_diff) OVER (PARTITION BY Product\_ID ORDER BY Sales\_Date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling\_variance

FROM (

SELECT Product\_ID,

Sales\_Date, Inventory\_Quantity \* Product\_Cost as daily\_sales,

(Inventory\_Quantity \* Product\_Cost - AVG(Inventory\_Quantity \* Product\_Cost) OVER (PARTITION BY Product\_ID ORDER BY Sales\_Date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)) as squared

FROM inventory\_data

) InnerDerived

) OuterDerived;

SET leadTimeDemand - avgRollingSales \* 7;

SET safetystock = 1.645 \* SQRT(avgRollingVariance \* 7);

SET recorderPoint = leadTimeDemand + safetyStock;

SELECT Product\_ID,

avg\_rolling\_sales \* 7 as Lead\_time\_demand,

1.645 \* (avg\_rolling\_variance \* 7) as safety\_stock,

(avg\_rolling\_sales \* 7) + (1.645 \* (average\_rolling\_variance \* 7 )) as reorder\_point

FROM InventoryCalculations;

INSERT INTO inventory\_optimization (Product\_ID, Reorder\_Point)

VALUES (productID, reorderPoint)

ON DUPLICATE KEY UPDATE Recorder\_Point = reorderPoint;

END //

DELIMITER;

-- Step 3 : make inventory\_data a permanent table

CREATE TABLE Inventory\_table AS SELECT + FROM Inventory\_data;

-- Step 4: Create the Triger

DELIMITER //

CREATE TRIGGER AfterInsertUnifiedTable

AFTER INSERT ON Inventory\_table

FOR EACH ROW

BEGIN

CALL RecalculateRecorderPoint(New.Product\_ID);

END //

DELIMITER;

**3. Overstocking and Understocking**

Overstocking means that inventory stock is more than the sales of goods. While Understocking means having zero stock with an increase demand for products.

(ix) OVERSTOCKING AND UNDERSTOCKING

WITH RollingSales AS (

SELECT Product\_ID,

Sales\_Date,

AVG(Inventory\_Quantity \* Product\_Cost) OVER (PARTITION BY Product\_ID ORDER BY Sales\_Date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as rolling\_avg\_sales

FROM inventory\_data

),

-- Calculate the number of days a product was out of stock

StockoutDays AS (

SELECT Product\_ID,

COUNT(\*) as stockout\_days

FROM inventory\_data

WHERE Inventory\_Quantity = 0

GROUP BY Product\_ID

)

-- Join the CTEs with the main table to get the results

SELECT f.Product\_ID,

AVG(f.Inventory\_Quantity \* f.Product\_Cost) as avg\_inventory\_value,

AVG(rs.rolling\_avg\_sales) as avg\_rolling\_sales,

COALESCE(sd.stockout\_days, 0) as stockout\_days

FROM inventory\_data f

JOIN RollingSales rs ON f.Product\_ID = rs.Product\_ID AND f.Sales\_Date = rs.Sales\_Date

LEFT JOIN StockoutDays sd ON f.Product\_ID = sd.Product\_ID

GROUP BY f.Product\_ID, sd.stockout\_days;

**Checking the results and they are no variance between the avg\_inventory\_value and avg\_rolling\_sales. This simply means that the products are not being sold or its constantly stocked. Which can result in overstocking.**

Step 5: Share

**--FEEDBACK LOOP**

**--Feedback Loop Establishment:**

-- Feedback Portal: Develop an online platform for stakeholders to easily submit feedback on inventory performance and challenges.

-- Review Meetings: Organize periodic sessions to discuss inventory system performance and gather direct insights.

-- System Monitoring; Use established SQL procedures to track system metrics, with deviations from expectations flagged for review.

**--Refinement Based on feedback;**

-- Feedback Analysis: Regularly compile and scrutinize feedback to identify recurring themes or pressing issues.

-- Action Implementation: Prioritize and act on the feedback to adjust reorder points, safety levels, or overall processes.

-- Change Communication: Inform stakeholders about changes, underscoring the value of their feedback and ensuring transparency.

**Deductions**

**-- General Insights:**

-- Inventory Discrepancies: The initial stages of the analysis revealed significant discrepancies in inventory levels, with instances of both overstocking and understocking

-- These inconsistencies were contributing to capital inefficiencies and customer dissatisfaction.

-- Sales Trends and External Factor Influences: The analysis indicated that sales trends were notably influenced by various external factors.

-- Recognizing these patterns provides an opportunity to forecast demand more accurately.

--Suboptimal Inventory Levels: The inventory optimization analysis showed that the existing inventory levels were not optimized for current sales trends.

-- Products that had either closed excess inventory was identified.

Step 6 Act

**-- Recommendation:**

1. Implement Dynamic Inventory Management: The company should transition from a static to a dynamic inventory management system, adjusting inventory levels based on real-time sales trends, seasonality, and external factors.

2. Optimize Reorder and safety stocks: Utilize the reorder points and safety stocks calculated during the analysis to minimize stockouts and reduce excess inventory. Regularly review these metrics to ensure they align with current market conditions.

3. Enhance Pricing Strategies: Conduct a thorough review of product pricing strategies, especially for products identified as unprofitable. Consider factors such as competitor pricing, market demand, and product acquisition costs.

4. Reduce Overstock: Identify products that are consistently overstocked and take steps to reduce their inventory levels. This could include promotional sales, discounts, or even discounting products with low sales performance.

5. Establish a feedback loop: Develop a systematic approach to collect and analyze feedback from various stakeholders. Use this feedback for continuous improvement and alignment with business objectives.

6. Regular Monitoring and Adjustments: Adopt a proactive approach to inventory management by regularly monitoring key metrics and making necessary adjustments to inventory levels, order quantities, and safety stocks.