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 task
2. A description of all data sources used
3. Documentation of any cleaning or manipulation of data
4. A summary of your analysis
5. 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.