Solving Inventory Inefficiencies Using SQL

Ayush Kumar Mandal (IITR)
Sahil (IITR)
Visheshta (IITR)

```sql
CREATE DATABASE IF NOT EXISTS Inventory;
USE Inventory;
CREATE TABLE stores (
    store_id VARCHAR(10) PRIMARY KEY,
    region VARCHAR(50)
);
CREATE TABLE products (
    product_id VARCHAR(10) PRIMARY KEY,
    category VARCHAR(50)
);
CREATE TABLE inventory_transactions (
    date DATE,
    store_id VARCHAR(10),
    product_id VARCHAR(10),
    inventory_level INT,
    units_sold INT,
    units_ordered INT,
    demand_forecast FLOAT,
    price FLOAT,
    discount INT,
    weather_condition VARCHAR(20),
    holiday_promotion BOOLEAN,
    competitor_pricing FLOAT,
    seasonality VARCHAR(20),
    PRIMARY KEY (date, store_id, product_id),
    FOREIGN KEY (store_id) REFERENCES stores(store_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

Database: Inventory

This database is designed to manage and analyze inventory transactions across multiple stores and product categories.

Table: stores

Purpose: Contains metadata about each retail store.

Column Name	Data Type	Description
store_id	VARCHAR(10)	Unique identifier for each store (Primary Key)
region	VARCHAR(50)	Geographical location of the store (e.g., North, South)

Table: products

Purpose: Stores product information.

Column Name	Data Type	Description
product_id	VARCHAR(10)	Unique identifier for each product (Primary Key)
category	VARCHAR(50)	Product category (e.g., Electronics, Grocery)

Table: inventory_transactions

Purpose: Tracks daily inventory activities and factors influencing sales.

Column Name	Data Type	Description
date	DATE	The date of the transaction
store_id	VARCHAR(10)	Store where transaction took place (Foreign Key)
product_id	VARCHAR(10)	Product involved in the transaction (Foreign Key)
inventory_level	INT	Number of units available in inventory
units_sold	INT	Number of units sold on that day
units_ordered	INT	Number of units ordered for restocking
demand_forecast	FLOAT	Forecasted customer demand for the product

price	FLOAT	Price at which the product was sold
discount	INT	Discount applied (as a percentage)
weather_condition	VARCHAR(20)	Weather on that day (e.g., Sunny, Rainy)
holiday_promotion	BOOLEAN	Whether a holiday promotion was active (TRUE/FALSE)
competitor_pricing	FLOAT	Pricing of the same product by competitors
seasonality	VARCHAR(20)	Season or period affecting demand (e.g., Winter, Summer)

Primary Key: Combination of (date, store_id, product_id) ensures each record is unique per store-product-day.

Foreign Keys:

- store_id → References stores(store_id)
- product_id → References products(product_id)

Observations:

- The schema allows robust analysis of demand patterns, pricing impact, promotions, and weather/seasonal effects on sales.
- Can be used for building predictive models and inventory optimization strategies.
- Normalization is maintained by separating product and store metadata.

```
30      -- current stock level calculation
31  •   SELECT
32          store_id,
33          product_id,
34          MAX(date) AS latest_date,
35          SUBSTRING_INDEX(GROUP_CONCAT(inventory_level ORDER BY date DESC), ',', 1) AS inventory_level
36      FROM inventory_transactions
37      GROUP BY store_id, product_id
38      ORDER BY store_id, product_id;
39      -- Low Inventory Detection
```

This SQL query calculates the **most recent stock level** for each product at each store.

Objective: To determine the latest inventory level of each product in every store based on the most recent date available in the inventory_transactions table.

Clause	Function
MAX(date)	Finds the most recent date for each store-product pair.
GROUP_CONCAT (inventory_level ORDER BY date DESC)	Concatenates inventory levels ordered from newest to oldest.
SUBSTRING_INDEX(..., ',', 1)	Extracts only the first (latest) inventory level from the concatenated list.
GROUP BY store_id, product_id	Ensures one row per store-product pair.
ORDER BY store_id, product_id	Sorts the result for easier reading.

MySQL doesn't allow direct access to non-aggregated columns in GROUP BY, so this trick ensures you're getting the inventory level that corresponds to the latest date without needing a subquery or window function.