Executive Summary

This executive summary presents the analytical outputs derived from SQL-driven inventory monitoring for Urban Retail Co. The objective was to improve inventory efficiency, reduce costs, and enable data-backed decision-making. Insights were extracted from historical sales, supplier logs, and inventory data using advanced SQL queries and are summarized below.

1. Fast-Selling vs Slow-Moving Products
Fast-selling products like toys and clothing consistently showed high sales and low stock buffer, indicating robust customer demand. Conversely, items like furniture and certain home essentials demonstrated low movement, occupying valuable shelf space.

Recommendation: Prioritize restocking fast movers and reduce procurement for underperforming SKUs. Utilize markdowns and bundling for stagnant inventory.

2. Stock Adjustment Recommendations
Overstocking of low-demand items resulted in inventory bloating and increased warehousing costs. In some cases, average stock exceeded expected demand by 40% for such products.

Recommendation: Adopt dynamic replenishment policies, incorporate Just-in-Time (JIT) models, and cut future orders for slow-moving SKUs.

3. Supplier Performance Inconsistencies
Several suppliers, especially in the electronics segment, failed to meet lead time expectations, resulting in stockouts. Supplier ratings were not always reflective of delivery consistency.

Recommendation: Introduce supplier scorecards including metrics like delivery punctuality, cost accuracy, and return volume. Adjust procurement partnerships based on actual performance.

4. Seasonal & Demand Forecasting Trends
Products in categories like toys and apparel showed seasonality-driven sales peaks during holidays and promotions, while essentials remained consistent. Electronics showed moderate cyclicality driven by promotions.

Recommendation: Align reorder points with seasonal patterns and initiate procurement in advance for high-demand cycles. Leverage these insights to optimize both stock availability and marketing efforts.

Business Impact:
These findings support data-driven inventory decisions, enabling Urban Retail Co. to reduce stockouts, optimize warehouse utilization, and improve procurement agility.

Inventory Monitoring and Optimization

Introduction:

Effective inventory management is essential for achieving operational excellence and cost efficiency. This project focuses on developing a comprehensive Inventory Monitoring System to optimize stock management, streamline workflows, and enhance decision-making using advanced SQL-based solutions.

Objectives:

1. Accurate Stock Level Calculations: Determine stock availability across stores, warehouses, regions, and categories for better resource allocation.

2. Low Inventory Alerts: Identify stock shortages using dynamic reorder point analysis.

3. Reorder Point Optimization: Utilize historical sales trends to estimate optimal reorder points.

4. Turnover and KPI Analysis: Analyze inventory turnover and key metrics such as stockout rates, inventory age, and average stock levels to improve operational performance.

Core Features:

- Dynamic Inventory Reports: Real-time insights into stock levels by integrating diverse data attributes like region, category, and seasonality.

- Low Stock Detection: Proactive identification of inventory needs with automated alerts for timely action.

- Predictive Analytics: Advanced queries for estimating reorder points and ensuring balanced stock levels.

- Operational Insights: Evaluate inventory turnover and track KPIs to drive strategic improvements.

Schema Design

```sql
CREATE TABLE inventory_forecasting (
  Date DATE,
  Store_ID INT,
  Product_ID INT,
  Category VARCHAR(255),
  Region VARCHAR(255),
  Inventory_Level INT,
  Units_Sold INT,
  Units_Ordered INT,
  Demand_Forecast INT,
  Price DECIMAL(10, 2),
  Discount DECIMAL(5, 2),
  Weather_Condition VARCHAR(255),
  Holiday_Promotion BOOLEAN,
  Competitor_Pricing DECIMAL(10, 2),
  Seasonality VARCHAR(255),
  PRIMARY KEY (Date, Store_ID, Product_ID)
);

CREATE TABLE stores (
  Store_ID INT PRIMARY KEY,
  Store_Name VARCHAR(255),
  Region VARCHAR(255),
  Address VARCHAR(255),
  Store_Type VARCHAR(50), -- e.g., Retail, Wholesale
  Manager_Name VARCHAR(255),
  Contact_Number VARCHAR(15)
);


CREATE TABLE products (
  Product_ID INT PRIMARY KEY,
  Product_Name VARCHAR(255),
  Category VARCHAR(255),
  Supplier_ID INT,
  Cost_Price DECIMAL(10, 2),
  Retail_Price DECIMAL(10, 2),
  Is_Active BOOLEAN DEFAULT TRUE,
  FOREIGN KEY (Supplier_ID) REFERENCES suppliers(Supplier_ID)
);
```

```
CREATE TABLE suppliers (
  Supplier_ID INT PRIMARY KEY,
  Supplier_Name VARCHAR(255),
  Contact_Person VARCHAR(255),
  Phone_Number VARCHAR(15),
  Email VARCHAR(255),
  Address VARCHAR(255),
  Performance_Rating DECIMAL(3, 2) -- e.g., 4.5 out of 5
);

CREATE TABLE sales_transactions (
  Transaction_ID INT PRIMARY KEY AUTO_INCREMENT,
  Date DATE,
  Store_ID INT,
  Product_ID INT,
  Quantity_Sold INT,
  Sale_Amount DECIMAL(10, 2),
  Discount_Applied DECIMAL(5, 2),
  FOREIGN KEY (Store_ID) REFERENCES stores(Store_ID),
  FOREIGN KEY (Product_ID) REFERENCES products(Product_ID)
);

CREATE TABLE purchase_orders (
  Purchase_Order_ID INT PRIMARY KEY AUTO_INCREMENT,
  Supplier_ID INT,
  Product_ID INT,
  Store_ID INT,
  Order_Date DATE,
  Quantity_Ordered INT,
  Delivery_Date DATE,
  Total_Cost DECIMAL(10, 2),
  FOREIGN KEY (Supplier_ID) REFERENCES suppliers(Supplier_ID),
  FOREIGN KEY (Product_ID) REFERENCES products(Product_ID),
  FOREIGN KEY (Store_ID) REFERENCES stores(Store_ID)
);

CREATE TABLE seasonal_trends (
  Seasonality VARCHAR(255),
  Category VARCHAR(255),
  Average_Demand INT,
  Peak_Month VARCHAR(50)
);
```

```sql
CREATE TABLE discounts_promotions (
  Promotion_ID INT PRIMARY KEY AUTO_INCREMENT,
  Product_ID INT,
  Start_Date DATE,
  End_Date DATE,
  Discount_Percentage DECIMAL(5, 2),
  Is_Holiday_Promotion BOOLEAN,
  FOREIGN KEY (Product_ID) REFERENCES products(Product_ID)
);

CREATE TABLE weather_conditions (
  Date DATE PRIMARY KEY,
  Region VARCHAR(255),
  Weather_Condition VARCHAR(255)
);

CREATE TABLE returns (
  Return_ID INT PRIMARY KEY AUTO_INCREMENT,
  Store_ID INT,
  Product_ID INT,
  Return_Date DATE,
  Quantity_Returned INT,
  Reason VARCHAR(255),
  FOREIGN KEY (Store_ID) REFERENCES stores(Store_ID),
  FOREIGN KEY (Product_ID) REFERENCES products(Product_ID)
);

CREATE TABLE stock_adjustments (
  Adjustment_ID INT PRIMARY KEY AUTO_INCREMENT,
  Date DATE,
  Store_ID INT,
  Product_ID INT,
  Adjustment_Type VARCHAR(50), -- e.g., Shrinkage, Restocking
  Quantity_Adjusted INT,
  Reason VARCHAR(255),
  FOREIGN KEY (Store_ID) REFERENCES stores(Store_ID),
  FOREIGN KEY (Product_ID) REFERENCES products(Product_ID)
);
```

SQL Query Suite

Query 1: Calculate Stock Levels

```
SELECT
  Store_ID,
  SUM(Inventory_Level) as Total_Stock_Level
FROM
  inventory_forecasting
GROUP BY
  Store_ID
ORDER BY
  Total_Stock_Level DESC;

SELECT
  Region,
  SUM(Inventory_Level) AS Total_Stock_Level
FROM
  inventory_forecasting
GROUP BY
  Region
ORDER BY
  Total_Stock_Level DESC;

SELECT
  Category,
  SUM(Inventory_Level) AS Total_Stock_Level
FROM
  inventory_forecasting
GROUP BY
  Category
ORDER BY
  Total_Stock_Level DESC;
```

For Combine Stock Levels

```
SELECT
  Store_ID,
  Region,
  SUM(Inventory_Level) AS Total_Stock_Level
FROM
  inventory_forecasting
GROUP BY
  Store_ID, Region
ORDER BY
```

```
  Total_Stock_Level DESC;
```

Query 2: Detect Low Inventory

```
SELECT
  Store_ID,
  Product_ID,
  Category,
  Inventory_Level,
  Demand_Forecast,
  CASE
    WHEN Reorder_Point IS NULL THEN 50 -- Default reorder point if not available
    ELSE Reorder_Point
  END AS Reorder_Point,
  (CASE
    WHEN Inventory_Level < (CASE WHEN Reorder_Point IS NULL THEN 50 ELSE
Reorder_Point END)
    THEN 'Needs Reorder'
    ELSE 'Stock OK'
  END) AS Status
FROM
inventory_forecasting;
```

Query 3: Reorder Point Estimation

```
  SELECT
    Store_ID,
    Product_ID,
    AVG(Units_Sold) * 7 AS Base_Reorder,
    (AVG(Units_Sold) * 7 * 0.2) AS Safety_Stock,
    (AVG(Units_Sold) * 7) + (AVG(Units_Sold) * 7 * 0.2) AS Reorder_Point
  FROM inventory_forecasting
  GROUP BY Store_ID, Product_ID
)
SELECT
  f.Store_ID,
  f.Product_ID,
  f.Category,
  f.Inventory_Level,
  f.Demand_Forecast,
  COALESCE(r.Reorder_Point, 50) AS Reorder_Point,
  CASE
    WHEN f.Inventory_Level < COALESCE(r.Reorder_Point, 50) THEN 'Needs Reorder'
    ELSE 'Stock OK'
  END AS Status
FROM inventory_forecasting f
```

```
LEFT JOIN ReorderPoints r
ON f.Store_ID = r.Store_ID AND f.Product_ID = r.Product_ID;
```

Query 4: Inventory Turnover Analysis

```
SELECT
  Store_ID,
  Product_ID,
  SUM(Units_Sold) AS Total_Units_Sold,
  AVG(Inventory_Level) AS Avg_Inventory_Level,
  CASE
    WHEN AVG(Inventory_Level) = 0 THEN NULL
    ELSE SUM(Units_Sold) / AVG(Inventory_Level)
  END AS Inventory_Turnover
FROM
  inventory_forecasting
GROUP BY
  Store_ID, Product_ID
ORDER BY
  Inventory_Turnover DESC;
Explanation
SUM(Units_Sold):
```

Query 5: Summary Report with Key Inventory KPIs

```
SELECT
  Store_ID,
  Product_ID,

 -- Percentage of days with no inventory (stockout rate)
(SUM(CASE WHEN Inventory_Level = 0 THEN 1 ELSE 0 END) * 100.0) / COUNT(*) AS
Stockout_Rate_Percentage,

  -- Average number of days a product is available  in inventory for calculating age of inventory
AVG(DATEDIFF(NOW(), DATE)) AS Avg_Inventory_Age_Days,

  -- Average Stock Levels
  AVG(Inventory_Level) AS Avg_Stock_Level
FROM
  inventory_forecasting
GROUP BY
  Store_ID, Product_ID
ORDER BY
  Stockout_Rate_Percentage DESC;
```

Inventory KPI Report

This report highlights key inventory performance indicators for the given timeframe, focusing on stock levels, sales trends, and turnover efficiency.

Key Performance Indicators :

1. Total Stock Levels: Reflects current inventory availability across all categories.
2. Units Sold Daily: Tracks sales performance and identifies peak periods.
3. Turnover Rate: Measures efficiency in inventory movement and replenishment.
4. Low Stock Items: Identifies products nearing depletion requiring immediate action.
5. Category Performance: Highlights demand and sales patterns by product type.

Observations :

1. High demand for toys and clothing.
2. Electronics show potential for growth with promotions.
3. Balanced performance in furniture stock and sales.

Recommendations :

1. Restock High-Demand Items: Prioritize replenishment for fast-selling categories.
2. Enhance Electronics Demand: Launch targeted offers or promotions.
3. Maintain Inventory Flow: Monitor turnover and align restocking with demand.

Conclusion:

Optimizing stock replenishment and leveraging sales insights can improve efficiency and meet customer demand effectively. Strategic actions on highlighted recommendations will further enhance inventory management.