## **Project Focus: Sales, Demand, and Pricing Optimization for E-commerce**

This project will demonstrate a comprehensive analytical approach to sales and inventory management, culminating in data-driven pricing insights.

### **1. Sales Trend Forecasting**

**Objective:** Predict overall future sales to help with high-level business planning and resource allocation.

* **Data Preparation (Key Steps):**
  + **Calculate Total Sales per Transaction:** Create a LineTotal column: Quantity \* UnitPrice.
  + **Aggregate Daily Sales:** Group your data by InvoiceDate (or just the date part of InvoiceDate) and sum the LineTotal to get daily total sales.
  + **Handle Missing Dates:** Ensure your daily sales data is continuous. If there are days with no sales, fill them with zeros to maintain the time series integrity.
* **Modeling (Free Tools/Libraries):**
  + **Python with Libraries:**
    - **Pandas:** For data aggregation and manipulation.
    - **Matplotlib/Seaborn:** For visualizing sales trends and forecasts.
    - **Statsmodels:** For classical time series models like ARIMA/SARIMA.
    - **Prophet (from Meta):** An excellent choice for forecasting, especially good with seasonality and holidays (though you'd need to add external holiday data if desired).
* **Insights You'll Gain:**
  + Overall daily/weekly/monthly sales predictions.
  + Identification of peak sales periods (e.g., holidays, specific months).
  + Understanding of general growth or decline trends.
  + Basis for high-level budget planning and staffing.

### **2. Demand Forecasting (Product Level)**

**Objective:** Predict future sales for individual products to optimize inventory and supply chain.

* **Data Preparation (Key Steps):**
  + **Filter for Key Products:** Start with a few high-value or high-volume StockCodes/Descriptions. Forecasting every single product might be too computationally intensive initially.
  + **Aggregate Product Sales:** For each selected product, group data by InvoiceDate (date part) and sum the Quantity sold to get daily product demand.
  + **Handle Missing Dates/Zeros:** Similar to overall sales, ensure continuous daily data for each product, filling missing dates with zero quantity.
* **Modeling (Free Tools/Libraries):**
  + **Python with Libraries:**
    - **Pandas:** For data manipulation.
    - **Matplotlib/Seaborn:** For visualizations.
    - **Statsmodels (ARIMA/SARIMA):** Effective for individual product time series.
    - **Prophet:** Also suitable here, especially if products have distinct seasonal patterns.
    - *Consider also:* Simpler models like Exponential Smoothing (from statsmodels or sklearn if you prepare the data as a sequence).
* **Insights You'll Gain:**
  + Specific demand predictions for individual products.
  + Identification of product-specific seasonality and trends.
  + Direct input for inventory reorder points and quantities.
  + Insights to reduce stockouts or overstocking.

### **3. Price Elasticity/Optimization**

**Objective:** Understand how price changes affect demand and recommend optimal prices for specific products.

* **Data Preparation (Key Steps):**
  + **Focus on Products with Price Variation:** This is the most crucial step. Identify StockCodes that have been sold at *different UnitPrices* over time in your dataset. If all instances of a StockCode have the same UnitPrice, you won't be able to calculate elasticity from this dataset alone. You might need to:
    - **Explore if UnitPrice truly varies for the same StockCode:** Sometimes prices change over time even in static datasets.
    - **If not, consider making assumptions/simulations (for demonstration):** For a project, you could *simulate* price variations for a few products (e.g., assume a 10% price reduction for a week and observe hypothetical demand changes based on common elasticity values found in literature). *However, to be truly data-driven, you need actual historical variations.*
  + **Feature Engineering:** For each unique StockCode with price variation, create features like Avg\_Quantity\_at\_Price\_X, Num\_Sales\_at\_Price\_X, etc. You'll likely want to aggregate sales for each specific UnitPrice for a given product.
* **Modeling (Free Tools/Libraries):**
  + **Python with Libraries:**
    - **Pandas:** Data preparation.
    - **Scikit-learn:** For Linear Regression or other regression models.
    - **Statsmodels:** For Ordinary Least Squares (OLS) regression, which provides statistical summaries including coefficient p-values.
* **Insights You'll Gain:**
  + **Price Elasticity Coefficient:** Quantify how much demand changes for a 1% change in price for a specific product.
  + **Optimal Pricing Recommendations:** Based on the elasticity, suggest a price that theoretically maximizes revenue (or profit if you incorporate cost data, which isn't in your current dataset).
  + Understanding which products are price-sensitive versus price-insensitive.