Short NoteBook Explanation (Author : Joel Siby)

- Socials :~
- Linkedin : https://www.linkedin.com/in/joelsiby/
- Github : https://github.com/joelsiby02
- Portfolio : https://joelsiby.vercel.app

1) Core Business Problem (and why it matters)

A growing hotel chain operates multiple bars across different locations. They are currently facing two major inventory challenges:

- **Stockouts of high-demand items (fast movers)**
  When popular brands run out, the bar cannot serve guests, which directly reduces revenue and negatively impacts guest satisfaction.

- **Overstocking of slow-moving items (slow movers)**
  When low-demand items are overstocked, it increases holding costs, ties up capital, and increases the risk of waste.

Business goal

Build a system that helps managers decide:

How much of each brand should each bar keep, and when should they reorder?

2. Assumptions Made (and Why)?

1) **Daily aggregation of demand**
The raw data is transaction-level (timestamp included). I aggregated consumption into daily demand per **Bar × Alcohol Type × Brand** because forecasting and inventory planning require consistent daily time steps.

2) **Missing days treated as zero demand**
If a bar-brand combination had no record on a day, I assumed demand was **0** for that day. This is necessary to model intermittent demand correctly and avoid biasing forecasts.

3) **Lead time assumed as 2 days**
The dataset does not provide supplier delivery lead time. I assumed a realistic **2-day lead time** to simulate ordering and replenishment delays.

4) **Service level fixed at 95% (z = 1.65)**
I assumed a **95% service level** to balance stockout risk and holding cost. This reflects a common business tradeoff for hospitality inventory.

5) **Bottle ordering constraint (750 ml)**
Since alcohol is typically replenished in bottles, I assumed orders must be placed in **750ml multiples**. This makes the simulation more realistic and explains why inventory sometimes exceeds the Par level.

6) **Stable operating conditions**
The model assumes normal bar operations and does not explicitly account for rare events like closures, strikes, or one-time large events. These would require external signals such as hotel occupancy or event calendars.

3) Model Used (and why)

Forecasting goal

Forecast daily item-level demand (**Consumed (ml)**) for each **Bar** × **Alcohol Type** × **Brand** time series.

Models tested (baseline forecasting)

I tested two simple and explainable baseline forecasting approaches:

1) **MA7 (Rolling 7-day Moving Average)**
Forecast = average demand of the last 7 days

2) **Seasonal Naive (7-day lag)**
Forecast = demand from the same day last week

Why MA7 was chosen

Using a time-based split and MAE evaluation across all 96 bar-brand series:

- **MA7 performed better in 69 series (~72%)**
- **Seasonal naive performed better in 27 series (~28%)**

This indicates demand is noisy and intermittent, so MA7 is a stronger default baseline.

Why not more complex models?

I did not use ARIMA/Prophet/XGBoost/Deep Learning because:

- the dataset is relatively small (~6.5k rows)
- demand is intermittent with many zero-demand days
- the goal is a practical, explainable system rather than heavy model complexity

A simple baseline model is more suitable for this real-world simulation.

4) System Performance (and what I would improve)

Forecasting performance

To validate forecasting quality, I compared MA7 vs Seasonal Naive using a time-based split and **MAE**.

Across 96 time series (6 bars × 16 brands):

- **MA7 performed better in 69 series (~72%)**
- Seasonal Naive performed better in 27 series (~28%)

This suggests demand is more noisy/intermittent than strongly weekly-seasonal.

Inventory + simulation performance

Forecasting was converted into inventory decisions using:

- **ROP (Reorder Point)** = when to reorder
- **Par level** = target inventory after replenishment
- Lead time = 2 days
- Service level = 95%
- Orders rounded to 750ml bottles

A simulation was run to mimic real operations:

- inventory decreases daily due to demand
- orders trigger when inventory falls below ROP
- stock arrives after lead time
- stockout days and order counts are tracked

Example outcome (Brown's Bar – Yellow Tail):

- **Orders placed:** 39
- **Stockout days:** 5
- Overstock peaks occurred mainly due to bottle-size rounding.

Improvements (next steps)

If more business data/time were available, I would like to improve:

1) **Use different service levels by item type**
Higher service level for fast movers, lower for slow movers.

2) **Include external demand drivers**
Weekday, holidays, hotel occupancy, events, seasonality.

3) **Better ordering optimization**
Reduce overstock caused by bottle rounding by adding caps or smarter order sizing.

4) **Use real lead times**
Learn lead time per supplier/location instead of assuming a constant valu

5) How this solution would work in a real hotel

This system can be used as a daily decision-support tool for bar managers.

Step-by-step workflow

1) **Daily update** Each bar records daily consumption and inventory balances (opening, purchases, closing).

2) **Demand forecasting** The system forecasts next-day demand for each **Bar** × **Brand** using the MA7 method.

3) **Inventory recommendation** For each bar-brand, the system calculates:

- **ROP (Reorder Point)** → when to reorder
- **Par level** → how much inventory to maintain after replenishment

4) **Order suggestion** If current inventory falls below ROP, the system recommends an order quantity. Orders are rounded to 750ml bottles to reflect real purchasing constraints.

5) **Operational simulation / monitoring** The system tracks:

- stockout days (unmet demand)
- order frequency
- overstock risk (inventory above par)

What managers get

For each bar location, managers receive:

- forecasted demand per brand
- recommended ROP and Par levels
- suggested reorder quantities
- simple KPIs (stockout risk, overstock risk)

This directly helps reduce both stockouts and overstock across all bar locations.

6. What would break at scale? What would I track in production?

What could break at scale

If this system is deployed across many hotels, bars, and items, the main challenges would be:

- **Too many item-location combinations**
  More bars and brands means many more time series to forecast and simulate.
- **Changing demand patterns** Demand can shift due to seasonality, new menus, promotions, or guest behavior.
- **Unstable lead times** Supplier delays or partial deliveries can break the lead-time assumption.
- **External events** Holidays, events or occupancy spikes can cause demand spikes that MA7 cannot predict without extra signals.
- **Data quality issues** Missing inventory logs, incorrect consumption entries, or delayed updates will reduce reliability.

What I would track in production

To ensure the system is working correctly, I would monitor:

- **Stockout rate** (% of days with stockout)
- **Total unmet demand (ml)**
- **Average inventory level** (proxy for holding cost / overstock)
- **Inventory turnover**
- **Forecast error (MAE / bias)**
- **Order frequency and order quantity**
- **Supplier fill rate and actual lead time**