# Report

**Airlift Technologies**

**BizOps Analyst-- Case Study**

**Name:** Syed Arham Ali Rizvi

**Date:** 10/09/2021

# Table Of Contents

**Specific Instructions for the Case Study:**

**Context --** It is extremely important for Airlift Express to have an efficient supply chain. As a retailing company, Airlift makes money on the products it sells, customers come to Airlift Express to order products on-demand that have to be delivered to them within 45 minutes.

At Airlift Express our biggest challenge is to manage the replenishments of our 4000+ products at scale across a network of multiple warehouses. The main objectives are:

1. We never want to be Out of Stock for any items in our inventory since that leads to bad customer experience and lost revenue.

2. We do not want to overstock and have too much of our financial capital tied up in stored inventory.

Airlift Express's overall strategy is to rapidly grow and win market share in the ecommerce market whilst being capital efficient and having positive gross margins comparable to the retail sector.

Airlift is currently following a very manual replenishment process for ordering products from suppliers. We feel that there's a lot of room for data-driven improvement in the process.

**Problem**:

1) The Sales and Purchase data for 2 products is provided in the link.

( BizOps Analyst - Supply Chain Case Study )

2) Please analyse this data and identify if there are any problems in the replenishment process of these 2 products.

3) What learnings can you derive from the data of these 2 SKUs that can be used to create a replenishment formula that can apply to all products?

4) What would that replenishment formula look like?

5) What would be a short-term approach to solving the root problem and what would the long-term roadmap look like?

**All the analysis was done using Jupyter Notebook, Python, Pandas and Matplotlib.**

**The notebook can be found in the same folder as this report.*

# Analysis & Insights:

**Note: The Analysis will help in the following section of '**Insights & Replenishment Formula**'*

After reading the Sales dataset:

```
df_sales.describe()
```

| | Price | Sales | Check-In |
|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 22.000000 |
| **mean** | 34.346667 | 64.813333 | 536.545455 |
| **std** | 10.766582 | 37.627704 | 504.872613 |
| **min** | 24.000000 | 0.000000 | 25.000000 |
| **25%** | 24.000000 | 45.000000 | 112.500000 |
| **50%** | 29.000000 | 61.000000 | 427.000000 |
| **75%** | 47.000000 | 94.000000 | 788.000000 |
| **max** | 51.000000 | 148.000000 | 1767.000000 |

From the table above, we can pick a few points (These values will come in handy for our analysis later on in the case study):

1. The average **Price** of the sales dataset is 34.346667, while the minimum is: 24 and maximum is: 51
2. The average **Sales** of the sales dataset is 64.813333, while the minimum is: 0 and maximum is: 148
3. The average **Check-in** qty of the sales dataset is 536.545455, while the minimum is: 25 and maximum is: 1767

Next, I created a column to retrieve the number of months the dataset is valuable for:

```
df_sales['month'] = df_sales['Date'].dt.month
df_sales.head(5)
```

]:

| | Product | Date | Price | Sales | Check-In | month |
|---|---|---|---|---|---|---|
| **0** | Apples | 2020-01-01 | 45.0 | 42.0 | 396.0 | 1.0 |
| **1** | Apples | 2020-01-02 | 45.0 | 37.0 | NaN | 1.0 |
| **2** | Apples | 2020-01-03 | 45.0 | 51.0 | NaN | 1.0 |
| **3** | Apples | 2020-01-04 | 45.0 | 49.0 | 40.0 | 1.0 |
| **4** | Apples | 2020-01-05 | 45.0 | 49.0 | NaN | 1.0 |

```
df_sales.month.unique()
```

]: array([ 1., 2., 3., nan, 5.])

**We find out that this Dataset is based on 3 months (Jan-Mar 2020).**

**Note**: There was an anomaly in the dataset where the value '2020-05-18' might have been accidentally entered. The value will note be considered in the analysis.

Next, the Dataset was divided into 2 ('Apples' and 'Milk') to get a better understanding of the data.

```
df_sales_apples.describe()
```

| | Price | Sales | Check-In | month |
|---|---|---|---|---|
| **count** | 75.000000 | 75.000000 | 14.000000 | 75.000000 |
| **mean** | 43.146667 | 41.026667 | 248.071429 | 1.786667 |
| **std** | 8.519760 | 32.987292 | 246.181442 | 0.758614 |
| **min** | 30.000000 | 0.000000 | 25.000000 | 1.000000 |
| **25%** | 30.000000 | 8.500000 | 46.750000 | 1.000000 |
| **50%** | 47.000000 | 45.000000 | 195.000000 | 2.000000 |
| **75%** | 51.000000 | 54.000000 | 363.750000 | 2.000000 |
| **max** | 51.000000 | 148.000000 | 847.000000 | 3.000000 |

```
df_sales_milk.describe()
```

| | Price | Sales | Check-In | month |
|---|---|---|---|---|
| **count** | 75.000000 | 75.000000 | 8.000000 | 75.000000 |
| **mean** | 25.546667 | 88.600000 | 1041.375000 | 1.786667 |
| **std** | 1.961062 | 24.814447 | 440.955112 | 0.758614 |
| **min** | 24.000000 | 46.000000 | 540.000000 | 1.000000 |
| **25%** | 24.000000 | 69.500000 | 735.500000 | 1.000000 |
| **50%** | 24.000000 | 91.000000 | 865.000000 | 2.000000 |
| **75%** | 28.000000 | 103.500000 | 1407.000000 | 2.000000 |
| **max** | 28.000000 | 136.000000 | 1767.000000 | 3.000000 |

As we can see from the figures above (Apples and Milk):

1. Both the products have an equal count therefore comparisons can be made more fairly.
2. The average Price of the apples is 43.146667, while the minimum is: 30 and maximum is: 51
3. The average Price of the milk is 25.546667, while the minimum is: 24 and maximum is: 28
4. The average Sales of the apples is 41.026667, while the minimum is: 0 and maximum is: 148
5. The average Sales of the milk is 88.6, while the minimum is: 46 and maximum is: 136
6. The average qty of check-ins of the apples is 248.071429, while the minimum is: 25 and maximum is: 847
7. The average qty of check-ins of the milk is 1041.375, while the minimum is: 540 and maximum is: 1767

While the above points show a significant difference between the 2 products, the **visualisations** below tell a better story.

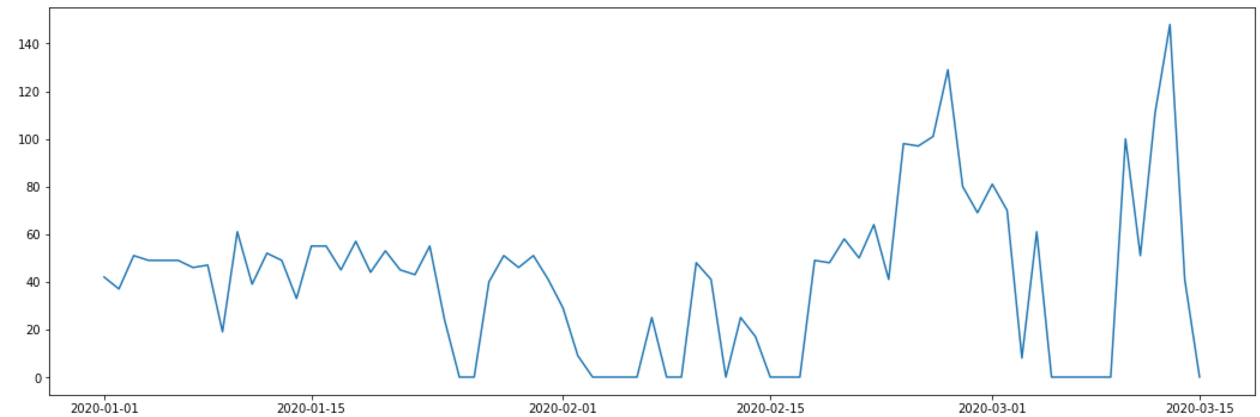



Figure: Apple Sales Time-series (Y-axis: Sales, X-axis: Date)

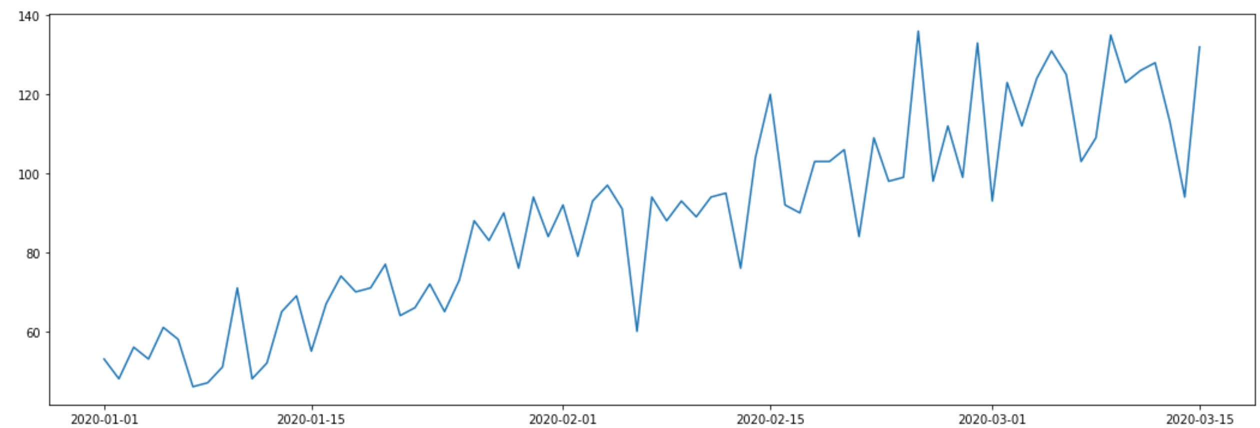



Figure: Milk Sales Time-series (Y-axis: Sales, X-axis: Date)

From the above figures, it's safe to say that on an average **Milk has had better sales as compared to Apples**. The company can also make '**Sales Forecasts**' for its 4000 products by automating and analysing the products by taking into consideration past data, consumer trends, and economic factors, businesses should predict the consumer demand they can expect to see in the future. This will also help them to realize how much inventory should be ordered to maximize profits while limiting losses.

We repeat the same process for the Price of both the products.

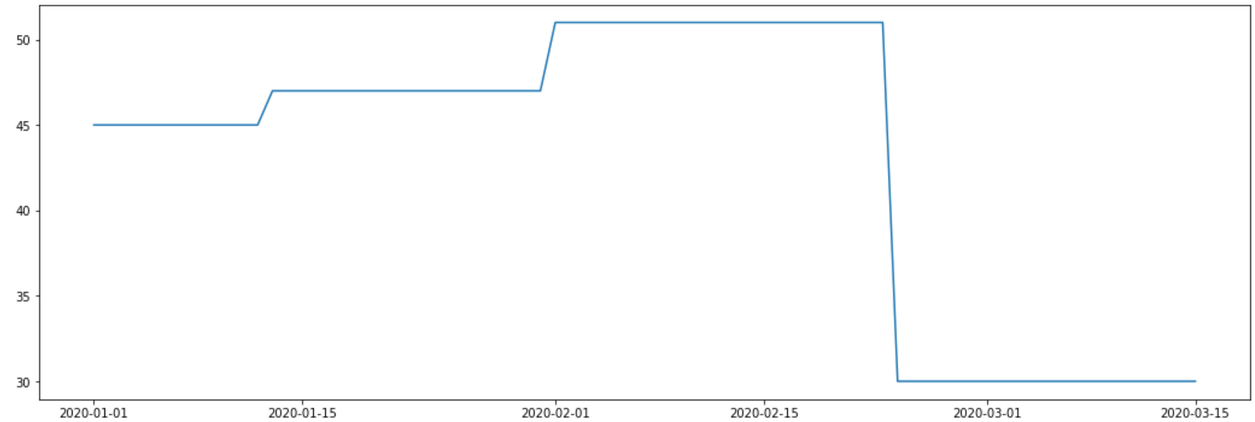



Figure: Apple Price Time-series (Y-axis: Price, X-axis: Date)

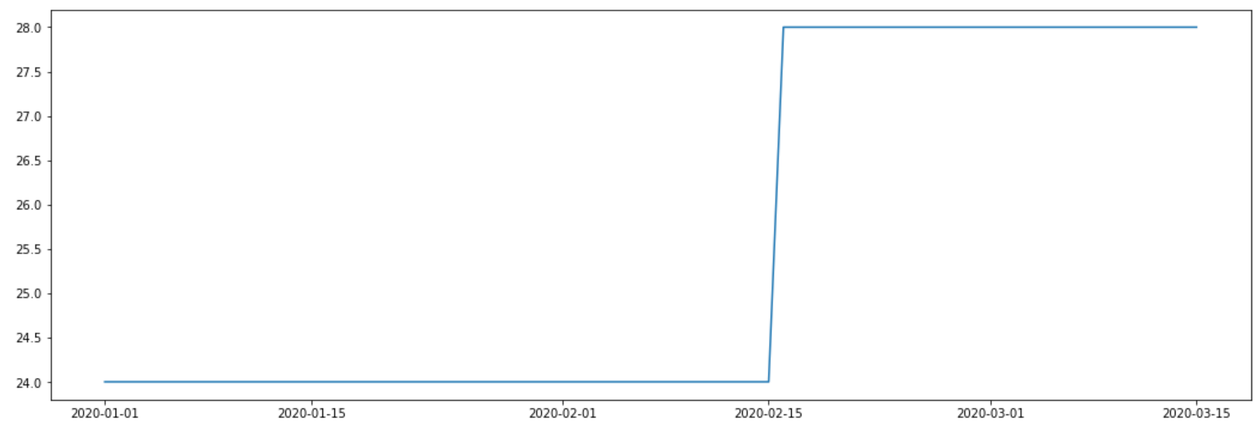



Figure: Milk Price Time-series (Y-axis: Price, X-axis: Date)

From the above 2 time series for Price, we can see that although apples have had several fluctuations in their price, they have enjoyed a higher price and their lowest price is still higher than the maximum price for milk.

Milk on the other hand had a steady price until half of 2nd month after which the price shot up eventually coming to a (higher) steady price again.

The company can make more informed decisions on when to increase or decrease the price which can lead to an advantage to their competitor.

Lastly, we move onto the amount of check-ins made per product for the 3 months.

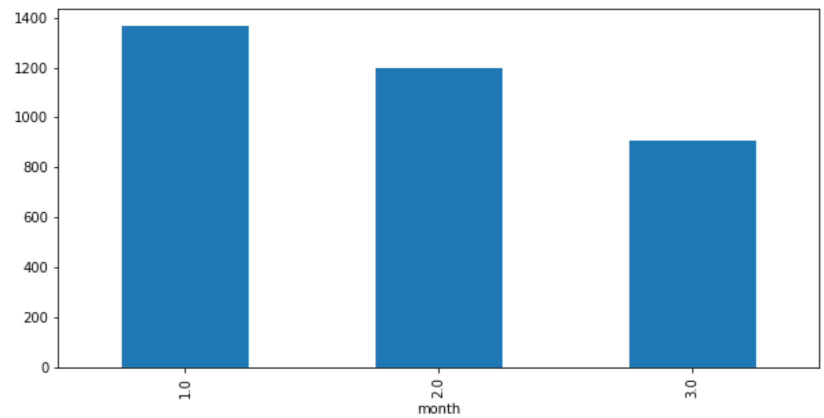



Figure: Apple Check In Bar Plot (Y-axis: Total Check Ins, X-axis: Months)

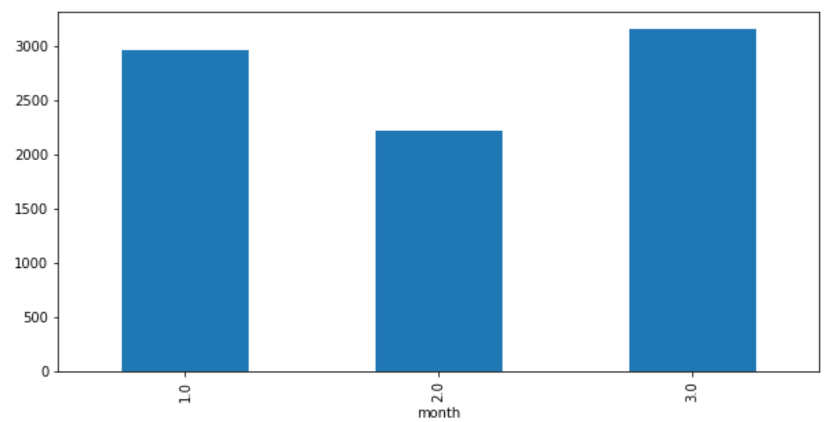



Figure: Milk Check In Bar Plot (Y-axis: Total Check Ins, X-axis: Months)

From the above 2 barplots, we can deduce that:

1. Milk had far greater check-ins than Apples
2. The check-ins for Apples have a steady decline across the months
3. The check-ins for Milk take a dip in the 2nd month but increase again in the 3rd month
4. This compliments the point for sales as the amount of sales for milk is higher compared to that of apples so naturally the amount of check ins for Milk would also be higher

It is important to analyse the check-ins made as we need to ensure the optimal amount of products to keep.

Next we move on to the **Purchase Orders** dataset:

We derive the Time Taken from the Date when the order was placed to the date when it was delivered to us. **In other words, The Lead Time.**

```
df_purchase['Time Taken'] = df_purchase['Delivery Date'] - df_purchase['Order Placement Date']
```

```
df_purchase.head()
```

]:

| | Order Placement Date | Supplier | Product | Ordered Quantity | Delivery Date | Time Taken |
|---|---|---|---|---|---|---|
| **0** | 2019-12-25 | AppleFarm | Apples | 550 | 2020-01-01 | 7 days |
| **1** | 2019-12-27 | AppleFarm | Apples | 50 | 2020-01-04 | 8 days |
| **2** | 2020-01-02 | AppleFarm | Apples | 40 | 2020-01-07 | 5 days |
| **3** | 2020-01-02 | AppleFarm | Apples | 300 | 2020-01-10 | 8 days |
| **4** | 2020-01-10 | AppleFarm | Apples | 300 | 2020-01-14 | 4 days |

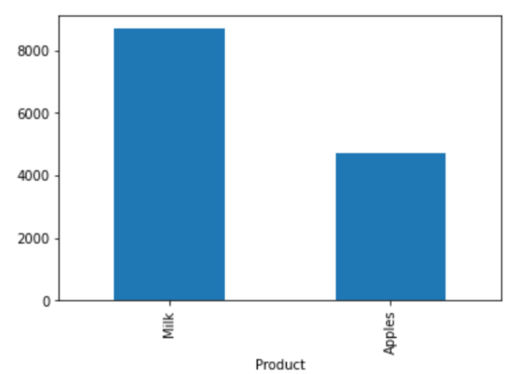
Next, we get the total ordered quantity per product:

```
df_purchase_product_ord_qty = df_purchase.groupby(['Product'])['Ordered Quantity'].sum()
df_purchase_product_ord_qty.nlargest()
```

```
9]: Product
Milk      8700
Apples    4730
Name: Ordered Quantity, dtype: int64
```

```
df_purchase_product_ord_qty.nlargest().plot(kind='bar')
```

```
0]: <AxesSubplot:xlabel='Product'>
```




From the visualisation above we can see that:

1. Milk was ordered in a far greater quantity than apples (8700:4730).
2. This compliments the point of milk being in more demand and making more sales and in turn being ordered more.

Again, the Dataset was divided into 2 ('Apples' and 'Milk') to get a better understanding of the data.

```
In [68]: df_purchase_apples.describe()
```

Out[68]:

| | Ordered Quantity | Time Taken |
|---|---|---|
| **count** | 14.000000 | 14 |
| **mean** | 337.857143 | 5 days 22:17:08.571428571 |
| **std** | 331.388189 | 3 days 07:01:13.352621380 |
| **min** | 40.000000 | 1 days 00:00:00 |
| **25%** | 62.500000 | 4 days 00:00:00 |
| **50%** | 300.000000 | 5 days 00:00:00 |
| **75%** | 487.500000 | 7 days 18:00:00 |
| **max** | 1100.000000 | 13 days 00:00:00 |

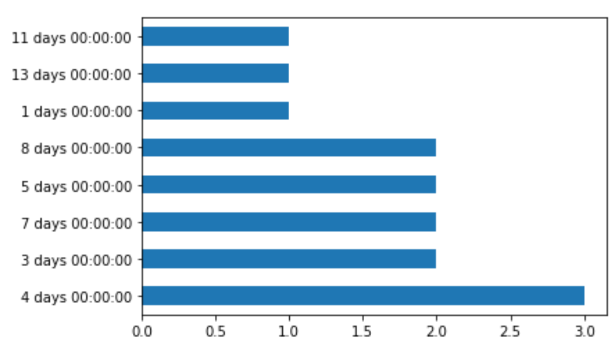
```
df_purchase_milk.describe()
```

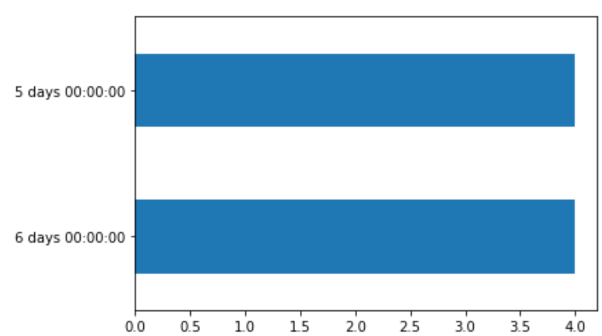
| | Ordered Quantity | Time Taken |
|---|---|---|
| **count** | 8.00000 | 8 |
| **mean** | 1087.50000 | 5 days 12:00:00 |
| **std** | 461.17087 | 0 days 12:49:42.742602466 |
| **min** | 600.00000 | 5 days 00:00:00 |
| **25%** | 775.00000 | 5 days 00:00:00 |
| **50%** | 900.00000 | 5 days 12:00:00 |
| **75%** | 1425.00000 | 6 days 00:00:00 |
| **max** | 1900.00000 | 6 days 00:00:00 |

From the above figures, we can see that:

1. The average Ordered Quantity for Apples is 337.857143, while the minimum is: 40 and maximum is: 1100 and count is: 14
2. The average Time Taken for Apples is 5 days, while the minimum is: 1 day and maximum is: 13 days
3. The average Ordered Quantity for Milk is 1087.5, while the minimum is: 600 and maximum is: 1900 and count is: 8
4. The average Time Taken for Milk is 5 days, while the minimum is: 5 day and maximum is: 6 days

Next, we check the frequency of the time taken for the products to be delivered (for both Apple & Milk)




Apple (Y-axis: days, X-axis: frequency of delivery)     Milk (Y-axis: days, X-axis: frequency of delivery)

From the above bar plots we can deduce that:

1. The average time taken for milk to be delivered was 5 to 6 days.
2. The apples were mostly delivered in 4 days.

Lead times can have a significant impact especially when the inventory is running low.

# Insights & Replenishment Formula:

Problem:
Inventory Management can be quite complex especially if you have a large number of SKUs. Manually keeping track of the stock levels and without proper inventory management, you're at risk of losing money and having excess inventory (that could potentially expire or become obsolete before you sell it), or running out of stock and losing customers.

Solution:
Having the optimal amount of product is key. Many businesses rely on what is called a **reorder quantity formula** which is similar to an economic order quantity (EOQ), you are trying to find the optimal order quantity to minimize logistics costs, warehousing space, stockouts, and overstock costs.

Let's further break it down:

**What is reorder quantity?**
Reorder quantity is the total number of product units you request from a manufacturer or supplier on an inventory replenishment purchase order. The exact amount should not be so high that you have too much capital tied up in inventory and subsequent warehousing costs, but not so low that there's not enough safety stock and you risk selling out before you can get the next batch of inventory.

**What is the reorder quantity formula?**
The reorder quantity calculation is done by using a formula that **multiplies average daily units sold by the average lead time**:

**Optimal Reorder Quantity for a SKU = Avg. Daily Units Sold  x Avg. Lead Time**

Lead time = how long it takes from when you place a purchase order with your manufacturer to when you receive the product in your warehouse

For example, in our case:

For **Apples**:

```
In [34]: df_sales_apples.describe()
Out[34]:
```

| | Price | Sales | Check-In | month |
|---|---|---|---|---|
| count | 75.000000 | 75.000000 | 14.000000 | 75.000000 |
| mean | 43.146667 | 41.026667 | 248.071429 | 1.786667 |
| std | 8.519760 | 32.987292 | 246.181442 | 0.758614 |
| min | 30.000000 | 0.000000 | 25.000000 | 1.000000 |
| 25% | 30.000000 | 8.500000 | 46.750000 | 1.000000 |
| 50% | 47.000000 | 45.000000 | 195.000000 | 2.000000 |
| 75% | 51.000000 | 54.000000 | 363.750000 | 2.000000 |
| max | 51.000000 | 148.000000 | 847.000000 | 3.000000 |

```
In [68]: df_purchase_apples.describe()
Out[68]:
```

| | Ordered Quantity | Time Taken |
|---|---|---|
| count | 14.000000 | 14 |
| mean | 337.857143 | 5 days 22:17:08.571428571 |
| std | 331.388189 | 3 days 07:01:13.352621380 |
| min | 40.000000 | 1 days 00:00:00 |
| 25% | 62.500000 | 4 days 00:00:00 |
| 50% | 300.000000 | 5 days 00:00:00 |
| 75% | 487.500000 | 7 days 18:00:00 |
| max | 1100.000000 | 13 days 00:00:00 |

Avg_daily_sales_apples ~ 41
Avg_lead_time_apples = 5

Then our **reorder quantity** for **Apples** is 205 units.

Similarly, for **Milk**:

```
df_sales_milk.describe()
```

| | Price | Sales | Check-In | month |
|---|---|---|---|---|
| **count** | 75.000000 | 75.000000 | 8.000000 | 75.000000 |
| **mean** | 25.546667 | 88.600000 | 1041.375000 | 1.786667 |
| **std** | 1.961062 | 24.814447 | 440.955112 | 0.758614 |
| **min** | 24.000000 | 46.000000 | 540.000000 | 1.000000 |
| **25%** | 24.000000 | 69.500000 | 735.500000 | 1.000000 |
| **50%** | 24.000000 | 91.000000 | 865.000000 | 2.000000 |
| **75%** | 28.000000 | 103.500000 | 1407.000000 | 2.000000 |
| **max** | 28.000000 | 136.000000 | 1767.000000 | 3.000000 |

```
df_purchase_milk.describe()
```

| | Ordered Quantity | Time Taken |
|---|---|---|
| **count** | 8.00000 | 8 |
| **mean** | 1087.50000 | 5 days 12:00:00 |
| **std** | 461.17087 | 0 days 12:49:42.742602466 |
| **min** | 600.00000 | 5 days 00:00:00 |
| **25%** | 775.00000 | 5 days 00:00:00 |
| **50%** | 900.00000 | 5 days 12:00:00 |
| **75%** | 1425.00000 | 6 days 00:00:00 |
| **max** | 1900.00000 | 6 days 00:00:00 |

Avg_daily_sales_milk ~ 88.6
Avg_lead_time_milk = 5

Then our **reorder quantity** for **Milk** is 443 units.

However, there may be exceptions to this formula. For example, if you are planning for any big promotions, thinking through holiday inventory preparations, or working on demand forecasting efforts, and you believe that your inventory velocity will increase, you can increase your reorder quantity so that you don’t run out of stock prematurely.

If your inventory days on hand metric is low, meaning you sell through inventory quickly so it’s not sitting around your warehouse for long, then your reorder quantity and inventory turnover rate will be high.

**Reorder quantity vs. reorder points**

**Reorder quantity:** is the number of products you order from a supplier.
**Reorder point:** is the predetermined number of units you have left on hand at which you need to reorder more inventory from your supplier or you will run out.

In other words, the reorder point is the quantity that will last you long enough until the reordered quantity arrives and becomes available in your warehouse.

The **reorder point formula** is:

**The sum of the total units needed to get you through the manufacturing lead time wait**
**+**
**some safety stock**

If you foresee either delays with your manufacturer or your sales increasing, you should increase your reorder point so that you reorder inventory sooner.

## Approaches To Solve The Problems:

1. In order to effectively create an effective replenishment plan, an **inventory management software** (IMS) is necessary to accurately track stock volumes. These automated systems can also integrate with POS (Point Of Sale) systems to provide the most up to date information regarding product quantities as well as tracking tools to notify users when they reach their reorder point. This will create a harmonious supply chain and allow the business to achieve service level targets, ultimately driving profits and enhancing customer satisfaction. However, Inventory management softwares comes with a hefty price tag and may add complexity which may result in the staff to undergo training to grasp how the system works. In such a scenario, an alternate method/plan may be required to solve the problem.

2. **ABC analysis**: If the levels of inventory are not reported accurately, this puts the whole reordering and replenishment process in jeopardy. The ability to determine which products to

prioritize will assist businesses in making sure they are stocking each item at their most optimal volume in a way that will also maximize profits.

There is a simple model available when assessing inventory prioritization and determining the right stocking levels for each item. This is called the ABC analysis model.

Items are categorized and ordered by-

A- Well-selling or high-profit margin items.
B- Mid-range valued items.
C- Low-profit margin items.

For example, A and B items could be prioritized for replenishment, whereas C items could be re-ordered based on current demand.

3. **Determine accurate lead times**: This is about knowing when to order inventory before an item becomes sold out. The lead time refers to how quickly the order can be fulfilled by the supplier. Having an accurate picture of vendor lead times helps businesses understand the right frequency of replenishing to prevent product shortages. When businesses wait too long to reorder popular products without taking into consideration how long it will take to receive these items, there will inevitably be a period of stockout.

4. **Utilize sales forecasting**: Past performance is one of the best factors to consider when deciding on the minimum and maximum quantities to order. This is when sales forecasting comes in. By taking into consideration past data, consumer trends, and economic factors, businesses should predict the consumer demand they can expect to see in the future. This will also help them to realize how much inventory should be ordered to maximize profits while limiting losses.

5. **Replenish from within the business**: For businesses that have multiple warehouses or store locations, first, check whether it's possible to replenish and redistribute inventory from other locations before reordering. This frees up working capital and allows a business to fulfill service levels without adding excess inventory purchasing costs.