# ISSS621 Data Science for Business

## Project Castlery

## Maximising Profit

Group 9

Lim Ming Jie Anne  Chen Yiman  Gladwin Lam Yu Hay  Liu Zhengyao  Shi Chee Liang

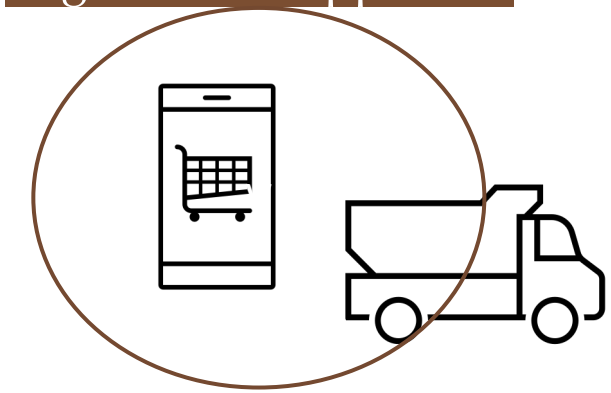
# Content



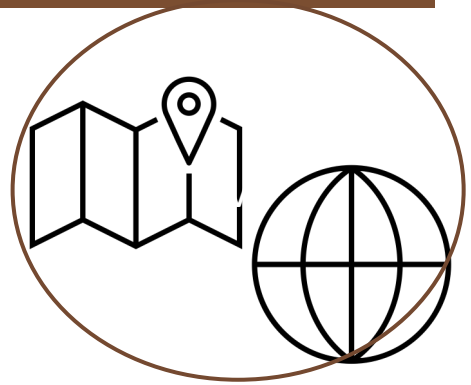
# All About CASTLERY:

A home-grown furniture retailer established in *2013*
Expanded to offering its products to several countries including *the United States, Canada, Australia, and several European countries.*

Digital- First approach



International Order



During Covid19

# Datasets

Raw data files:

## Fact_Saleorder.csv : 5000 rows

| order_id | user_id | order_date | sales |
|---|---|---|---|
| 100543 | 2644 | 5/26/2021 | 305.97 |
| 100559 | 4010 | 1/1/2021 | 2163 |
| 102578 | 4805 | 5/22/2021 | 29.97 |

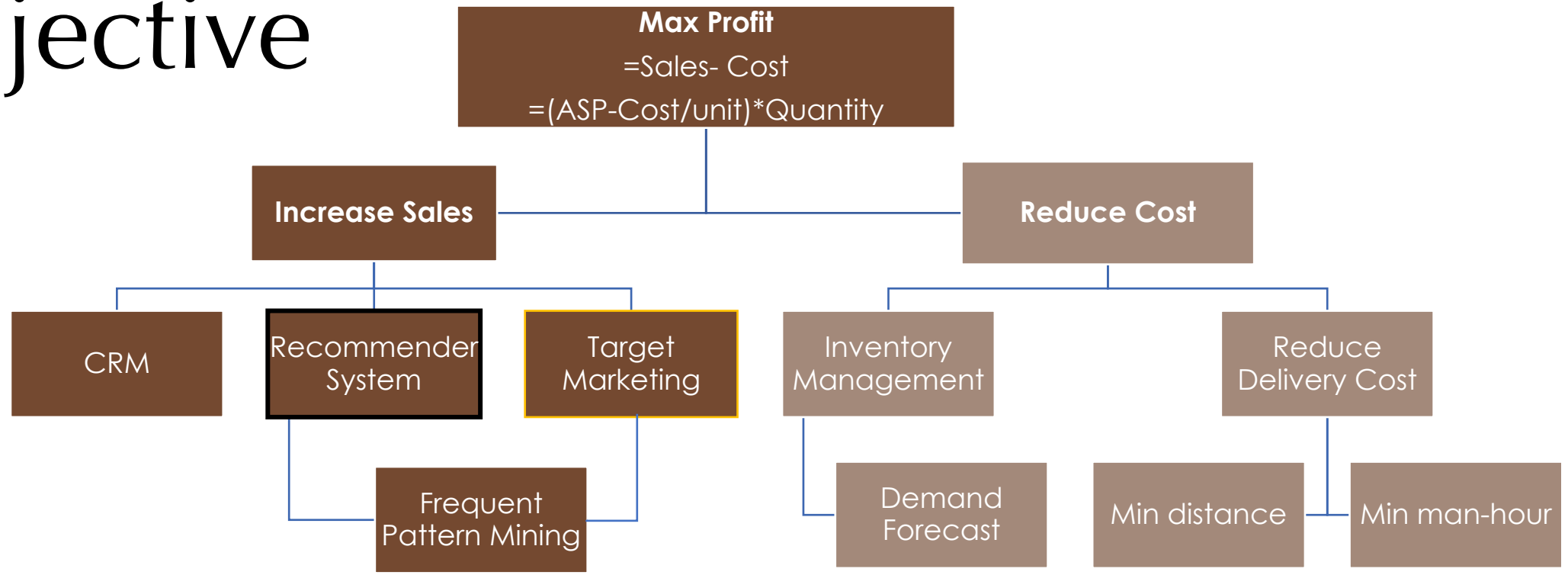
## Fact_Saleorderline.csv : 7898 rows

| saleline_id | order_id | product | category | sale_amount | quantity |
|---|---|---|---|---|---|
| 563832 | 323741 | SKU 1 | Cushions | 350 | 4 |
| 559257 | 320135 | SKU 1 | Cushions | 175 | 2 |
| 562777 | 305952 | SKU 1 | Cushions | 175 | 2 |

## La_Deliveries.csv : 1025 rows

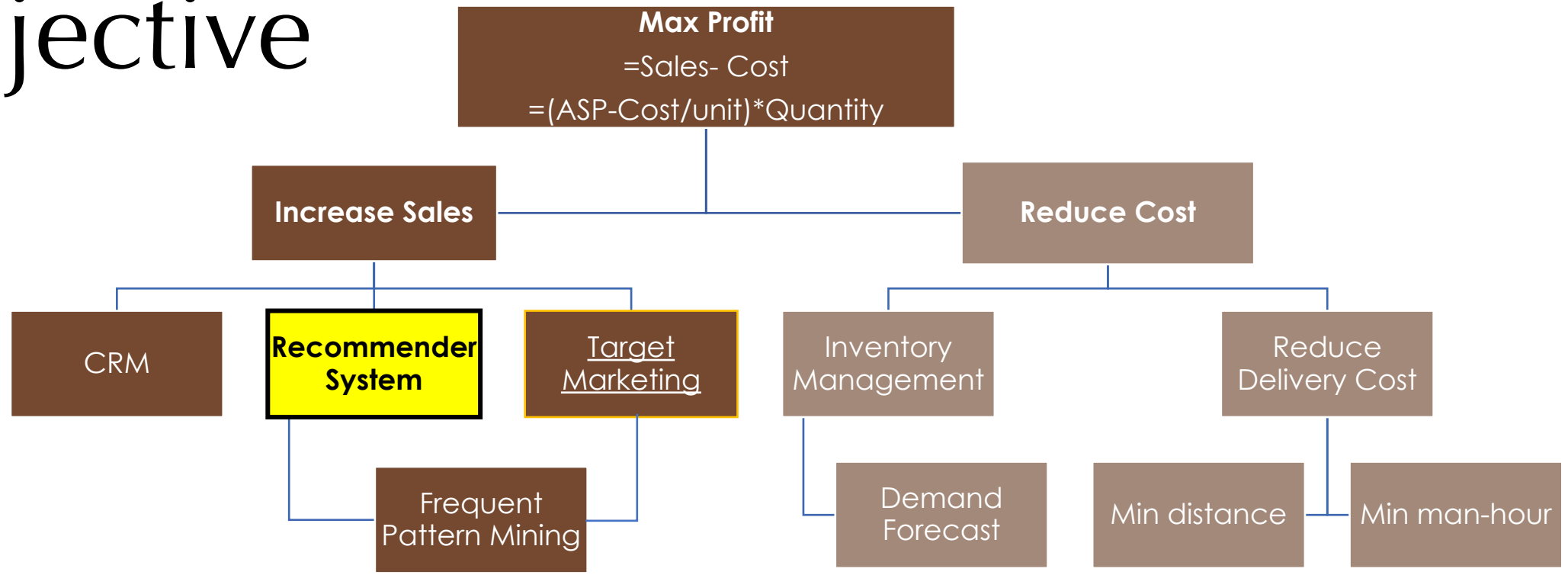
| delivery_id | total_cbm | delivery_date | cbsa_name | city | zip | population | lat | lng |
|---|---|---|---|---|---|---|---|---|
| 28466 | 0.78648 | 25/1/2022 | Los Angeles-Long Beach-Anaheim, CA | Los Angeles | 90001 | 59832 | 33.97398 | -118.24955 |
| 32246 | 0.493039 | 8/2/2022 | Los Angeles-Long Beach-Anaheim, CA | Los Angeles | 90005 | 39732 | 34.05912 | -118.30654 |
| 28028 | 0.248724 | 21/1/2022 | Los Angeles-Long Beach-Anaheim, CA | Los Angeles | 90005 | 39732 | 34.05912 | -118.30654 |

# Objective




1. How can Castlery use more targeted marketing to improve revenue?
2. How can Castlery manage their inventory to improve revenue and reduce cost?
3. How can Castlery manage their deliveries more efficiently to reduce cost?

# Objective

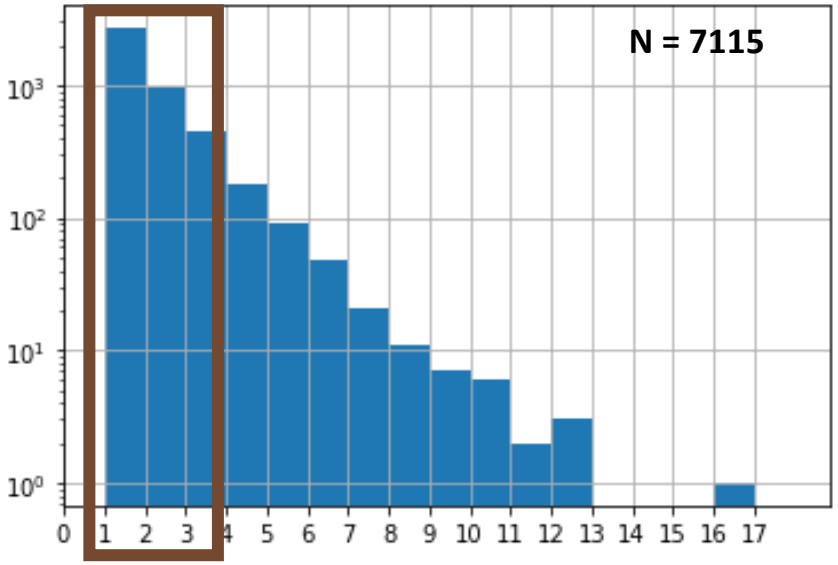



**1. How can Castlery use more targeted marketing to improve revenue?**

2. How can Castlery manage their inventory to improve revenue and reduce cost?

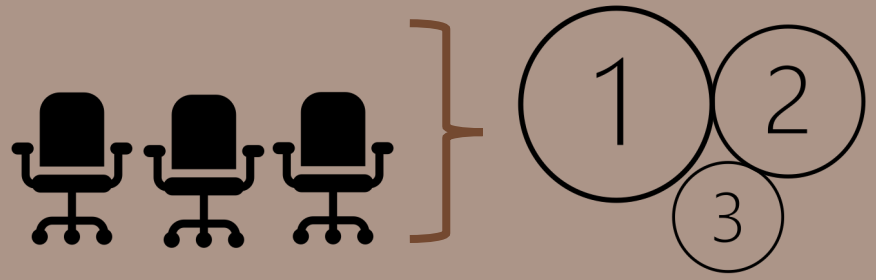
3. How can Castlery manage their deliveries more efficiently to reduce cost?

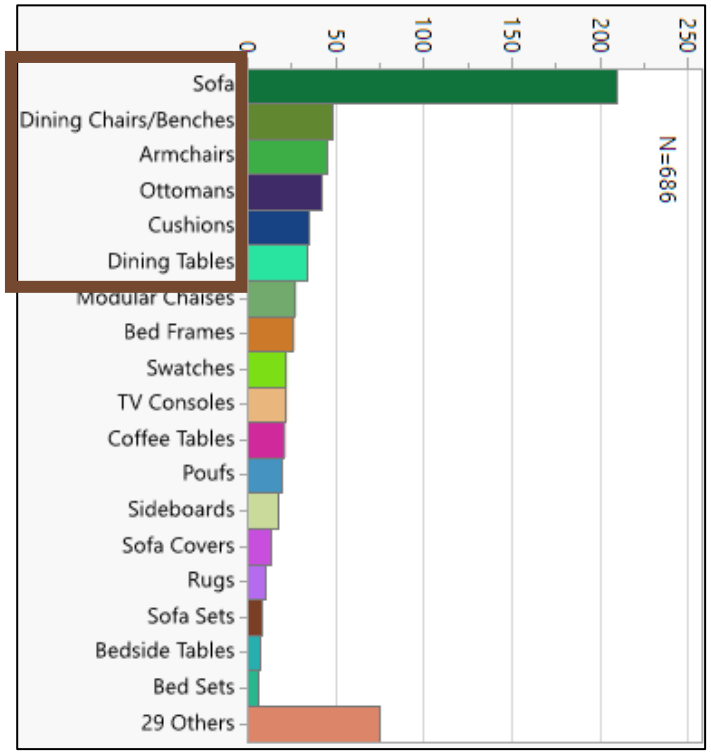
# What is in the order ?




Count of items per order

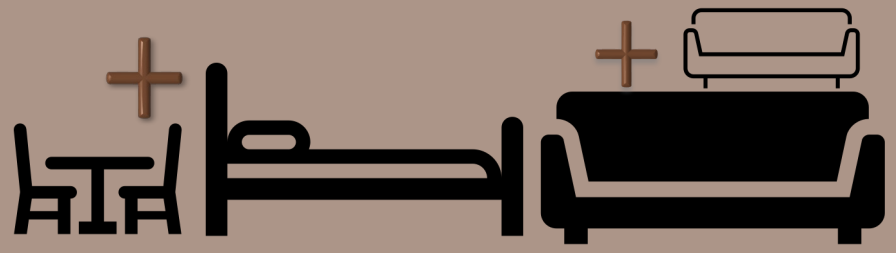
Majority of orders have:




Pareto of categories listed

Possible combinations:



# Apriori and Association Rule

**Apriori Algo**: Generate a k-itemset candidate by merging a pair of frequent (k-1) itemsets only if their first k-2 items are identical.
{659 & 353} + {659 & 89} → {659 & 89 & 353} where 0.34% > min_Sup

Min_sup = 0.3%

| Item | Category | Count | Support (%) |
|---|---|---|---|
| 659 | Table | 30 | 0.42% |
| 353 | Bench | 74 | 1.03% |
| 152 | Table | 32 | 0.45% |
| 614 | Chair | 293 | 4.10% |
| 681 | Table | 40 | 0.56% |
| 528 | Chair | 47 | 0.66% |
| 89 | Chair | 140 | 1.96% |
| … | … | … | … |

***count = # of orders this item appears in**

| Item | Count | Support (%) |
|---|---|---|
| 659 & 353 | 29 | 0.41% |
| 659 & 89 | 26 | 0.36% |
| 152 & 614 | 31 | 0.43% |
| 681 & 528 | 37 | 0.52% |
| … | … | … |

| Item | Count | Support (%) |
|---|---|---|
| 659 & 89 & 353 | 24 | 0.34% |
| … | … | … |

# Apriori and Association Rule

**Apriori Algo**: Generate a k-itemset candidate by merging a pair of frequent (k-1) itemsets only if their first k-2 items are identical.
{659 & 353} + {659 & 89} → {659 & 89 & 353} where 0.34% > min_Sup

## Min_sup = 0.3%

| Item | Category | Count | Support (%) |
|---|---|---|---|
| 659 | Table | 30 | 0.42% |
| 353 | Bench | 74 | 1.03% |
| 152 | Table | 32 | 0.45% |
| 614 | Chair | 293 | 4.10% |
| 681 | Table | 40 | 0.56% |
| 528 | Chair | 47 | 0.66% |
| 89 | Chair | 140 | 1.96% |
| … | … | … | … |

***count = # of orders this item appears in**

| Item | Count | Support (%) |
|---|---|---|
| 659 & 353 | 29 | 0.41% |
| 659 & 89 | 26 | 0.36% |
| 152 & 614 | 31 | 0.43% |
| 681 & 528 | 37 | 0.52% |
| … | … | … |

| Item | Count | Support (%) |
|---|---|---|
| 659 & 89 & 353 | 24 | 0.34% |
| … | … | … |

```
SKU_cat_dataset = sale_by_order_df['product_category'].to_list()

L, suppData = apriori(SKU_cat_dataset, 0.003) #trial and error to reach 0.003
print("Frequent itemsets:")
for i in L:
    for j in i:
        if len(j) != 1: #a modification to remove single SKU frequent sets
            print(j)
#print("Frequent itemsets:", L)

print()
print("Support of each itemset:")
for k in suppData.keys():
    if len(k) != 1:
        print(k, suppData[k] )
#print("Support:", suppData)
```

```
Frequent itemsets:
frozenset({'SKU 659_Rectangular Dining Tables', 'SKU 353_Dining Benches'})
frozenset({'SKU 152_Rectangular Dining Tables', 'SKU 614_Dining Chairs'})
frozenset({'SKU 528_Dining Chairs', 'SKU 681_Extendable Dining Tables'})
frozenset({'SKU 499_Dining Benches', 'SKU 681_Extendable Dining Tables'})
frozenset({'SKU 528_Dining Chairs', 'SKU 499_Dining Benches'})
frozenset({'SKU 614_Dining Chairs', 'SKU 687_Rectangular Dining Tables'})
```

: Looking for frequent itemsets

## Generate recommender rules from frequent itemsets :

```
dat = sale_by_order_df['product_category'].to_list()
L_b, suppData = apriori(dat, 0.003)                  #Apriori algo fine-tuned to min_sup of 0.3%
rules = generateRules(L_b, suppData, minConf=0.5)    #50% minConf would give a total of 20 rules
print()
```

```
frozenset({'SKU 659_Rectangular Dining Tables'}) --> frozenset({'SKU 353_Dining Benches'}) conf: 0.6666666666666666
frozenset({'SKU 152_Rectangular Dining Tables'}) --> frozenset({'SKU 614_Dining Chairs'}) conf: 0.5625
frozenset({'SKU 681_Extendable Dining Tables'}) --> frozenset({'SKU 528_Dining Chairs'}) conf: 0.675
frozenset({'SKU 528_Dining Chairs'}) --> frozenset({'SKU 681_Extendable Dining Tables'}) conf: 0.627906976744186
frozenset({'SKU 499_Dining Benches'}) --> frozenset({'SKU 528_Dining Chairs'}) conf: 0.5588235294117647
```

# Recommender System

There are way **more orders buying chair/bench SKUs** than **table SKUs**
Implying customers buying table would more likely pair with chairs
Whereas customers might just buy chair/bench without pairing a table?

| Item | Category | Count* | Support (%) |
|---|---|---|---|
| 659 | Table | 30 | 0.42% |
| 353 | Bench | 74 | 1.03% |
| 152 | Table | 32 | 0.45% |
| 614 | Chair | 293 | 4.10% |
| 681 | Table | 40 | 0.56% |
| 528 | Chair | 47 | 0.66% |
| 89 | Chair | 140 | 1.96% |
| … | … | … | … |

***count = # of orders this item appears in**

```python
def generate_recommendations(cart, rules, min_confidence=0.5):
    # Create set with items in the cart
    items = set(cart)

    # Initialize an empty list to store the recommended items
    recommended_items = []

    # Loop through the association rules and check if the antecedent is a subset of the purchased items
    for rule in rules:
        antecedent, consequent, confidence = rule
        if antecedent.issubset(items) and confidence >= min_confidence:
            # Add the consequent (recommended item) to the list of recommended items
            recommended_items.extend(list(consequent - items))

    # Return a list of unique recommended items
    return list(set(recommended_items)) #use set to remove duplicates
```

Let's see how the recommender rules tell us.

# Recommender System

| Item | Category | Count* | Support (%) |
|---|---|---|---|
| 659 | Table | 30 | 0.42% |
| 353 | Bench | 74 | 1.03% |
| 152 | Table | 32 | 0.45% |
| 614 | Chair | 293 | 4.10% |
| 681 | Table | 40 | 0.56% |
| 528 | Chair | 47 | 0.66% |
| 89 | Chair | 140 | 1.96% |
| … | … | … | … |

***count = # of orders this item appears in**

Q: Customers buying table would more likely pair with chairs, but not vice versa?

A: The recommender rules suggests Customers would buy this dining table SKU and pair with chairs and benches. But not vice versa.

```
# Assume the user has some items in the cart
user_cart = ['SKU 659_Rectangular Dining Tables']

# Generate recommendations using the association rules
recommended_items = generate_recommendations(user_cart, rules)

# Print the list of recommended items
print(recommended_items)
```

```
['SKU 89_Dining Chairs', 'SKU 353_Dining Benches']
```

```
# Assume the user has some items in the cart
user_cart = ['SKU 89_Dining Chairs', 'SKU 353_Dining Benches']

# Generate recommendations using the association rules
recommended_items = generate_recommendations(user_cart, rules)

# Print the list of recommended items
print(recommended_items)
```

```
[]
```

```
# Assume the user has some items in the cart
user_cart = ['SKU 349_3 Seater Sofas', 'SKU 155_3 Seater Sofas']

# Generate recommendations using the association rules
recommended_items = generate_recommendations(user_cart, rules)

# Print the list of recommended items
print(recommended_items)
```

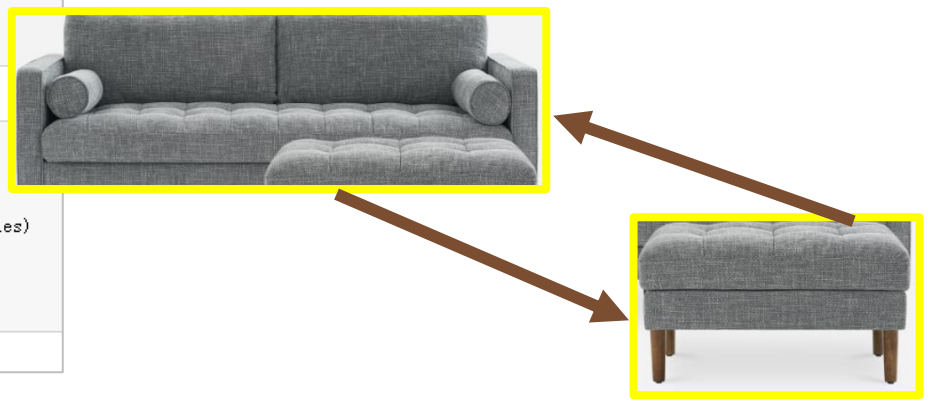
```
['SKU 316_Ottomans', 'SKU 529_Ottomans']
```

```
# Assume the user has some items in the cart
user_cart = ['SKU 316_Ottomans', 'SKU 529_Ottomans']

# Generate recommendations using the association rules
recommended_items = generate_recommendations(user_cart, rules)

# Print the list of recommended items
print(recommended_items)
```

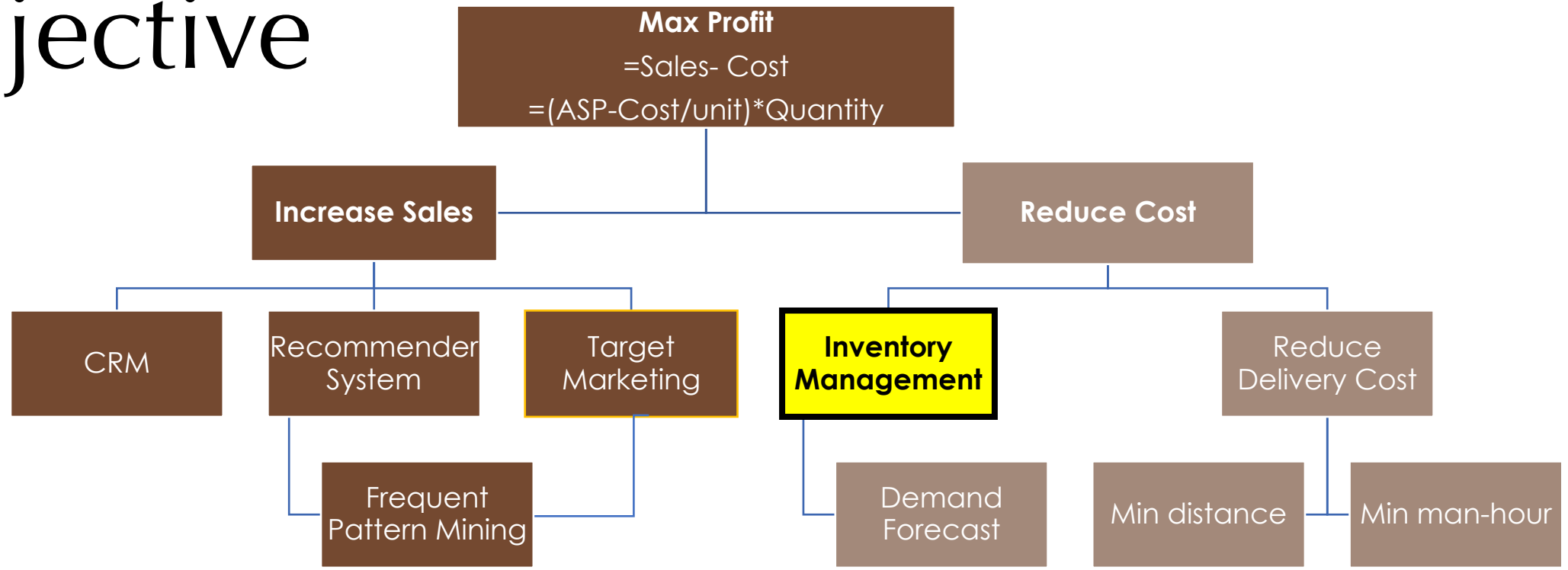
```
['SKU 349_3 Seater Sofas', 'SKU 155_3 Seater Sofas']
```

For the Dining table and Chairs and Benches:

For the Sofa and Ottoman pairs:



# Objective




1. How can Castlery use more targeted marketing to improve revenue?
2. **How can Castlery manage their inventory to improve revenue and reduce cost?**
3. How can Castlery manage their deliveries more efficiently to reduce cost?

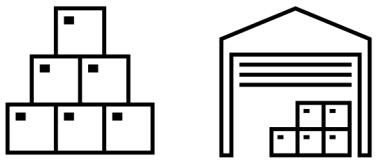
# Inventory Management & Forecasting

**Business Problem:**
How can Castlery manage their inventory to improve revenue and reduce cost?

Translate Business Problem into goals

TOO MUCH INVENTORY

- Expensive : furniture pieces take up large amount of warehouse space
- Sell slowly compared with other types of products like food



TOO LITTLE INVENTORY

- Cause a loss of sales
- Lead to additional costs in trying to speed up production processes or delivery lead times.

- **Goal 1 :Lower inventory** to minimise holding costs
- **Goal 2 :Maintain sufficient inventory levels** to avoid stockouts (i.e., lost sales)

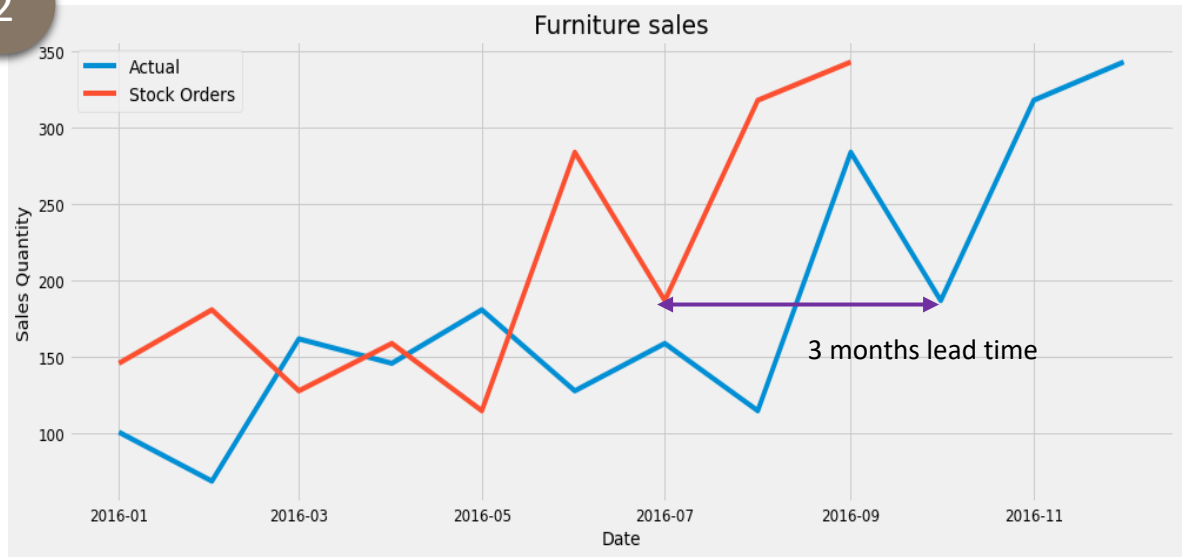
# Inventory Management & Forecasting

How much to order? And when to order?

1


**Naïve solution**: order as we sell
Problem: stocks take time to deliver

2

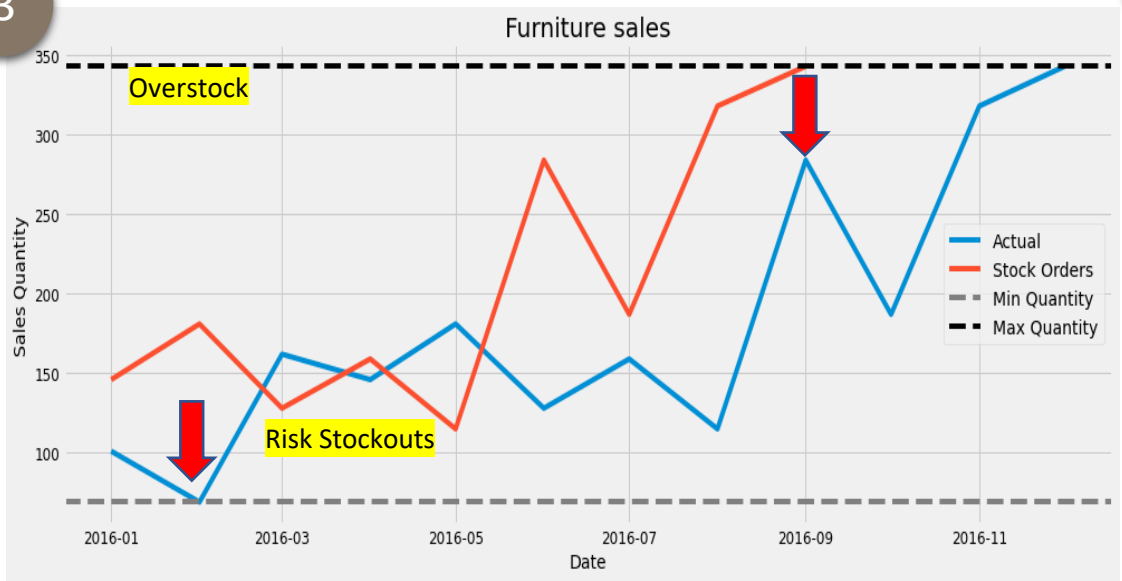



**Solution 1**: order in advance
Problem : How much to order?

# Inventory Management & Forecasting

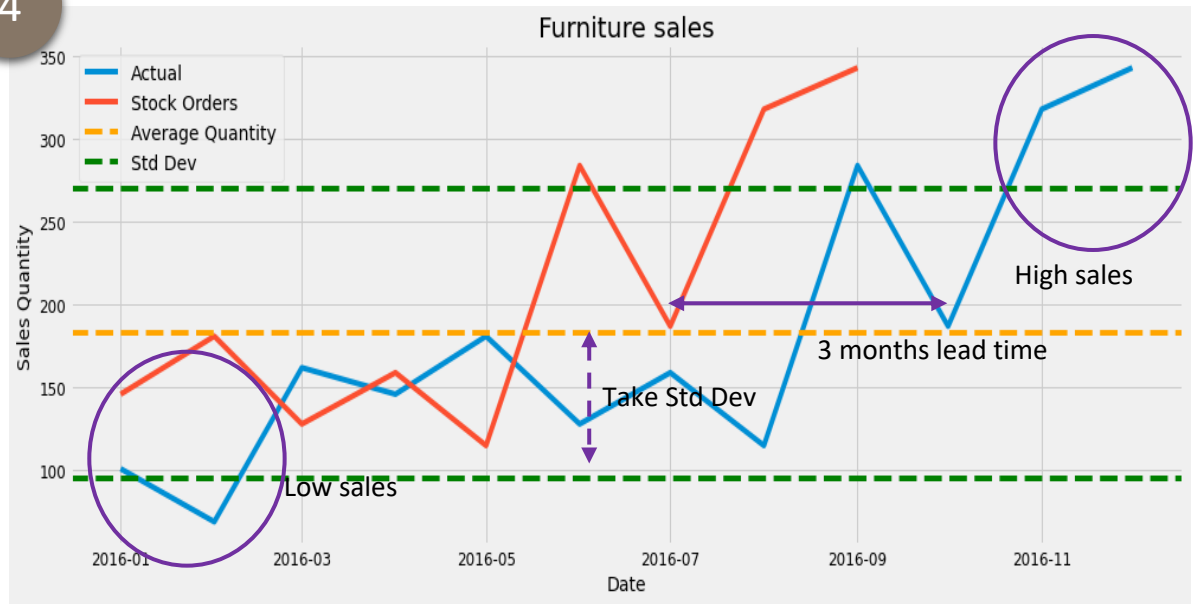
## How to handle variance

3




**Solution 2**: Order at Max / Min demand?
Problem: Costs from overstocking, stockout

4



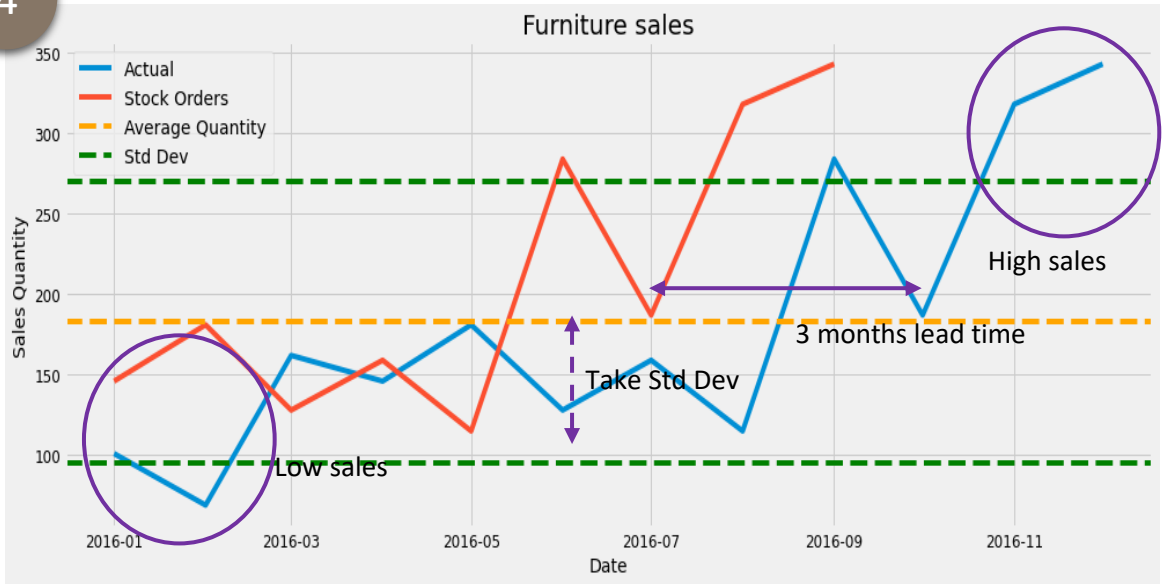

Solution 4: identify a representative line of demand

# Inventory Management & Forecasting

## Introduction to ROP, SS and EOQ concepts

4

### Solution 4: Average the demand and take SD




Solution is shaping up, but we still have some problems:

- Does not take trends into account
- Does not work for high variability in demand

How do we quantify it?
Identify the reorder point, safety stock and economic order quantity

Reorder Point (ROP): ROP is the inventory level at which a new order should be placed to avoid stockouts

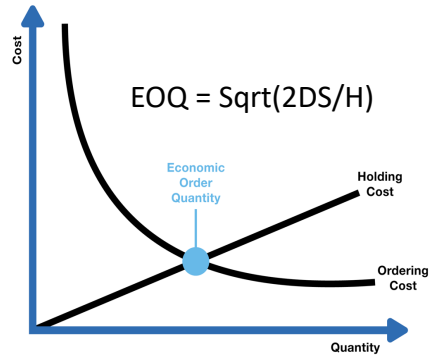
*ROP = (Demand x lead time) + Safety Stock*

Safety Stock (SS): SS is the extra inventory held to mitigate the risk of stockouts due to demand variability.

*SS = Z x σ x lead Time (Z is Z-core, σ is SD)*

Economic Order Quantity (EOQ): EOQ is the optimal order quantity that minimizes the total cost of inventory management

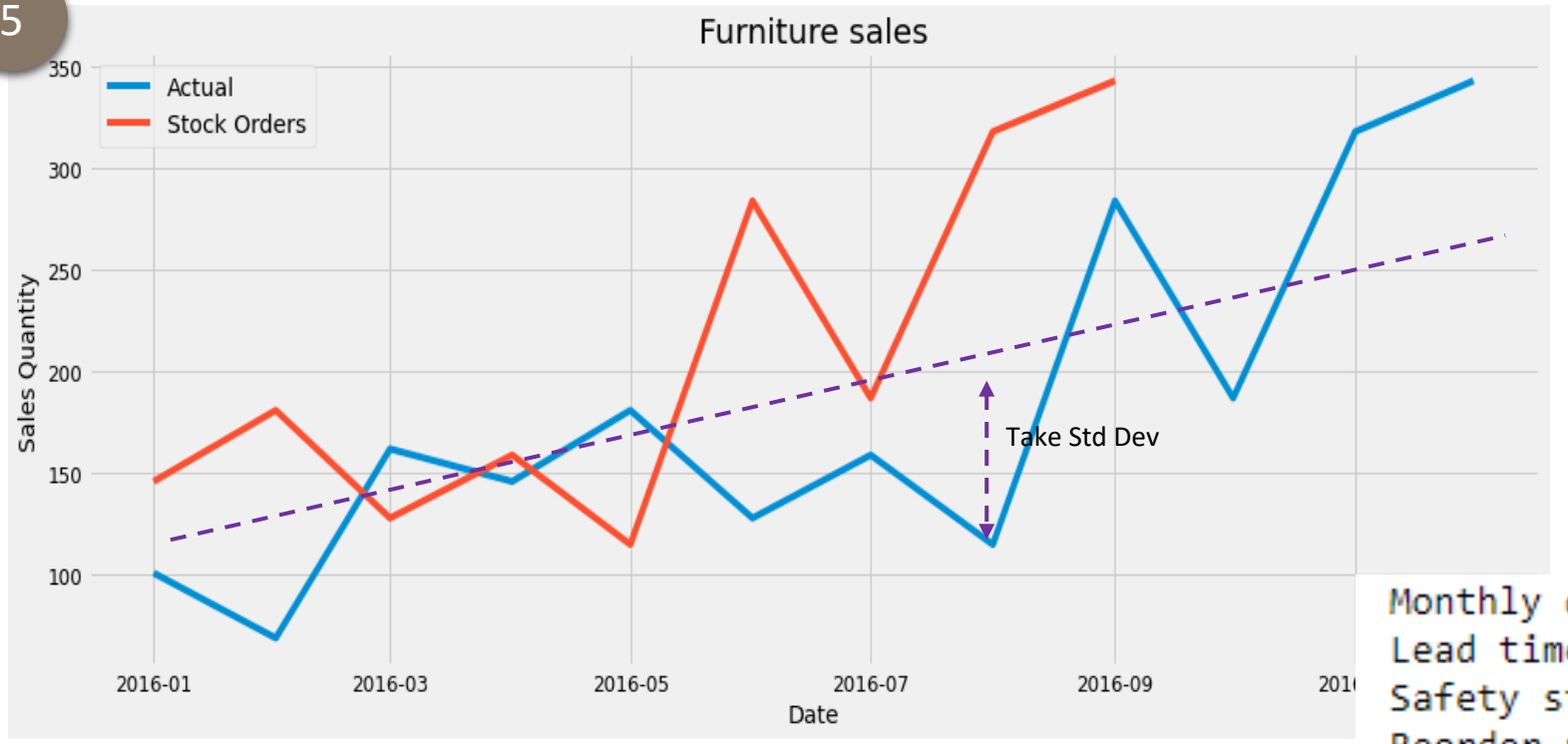
*EOQ = Sqrt((2 x demand x Ordering costs) /Holding Costs)*



# Inventory Management & Forecasting

Is there a better way to project demand?

5

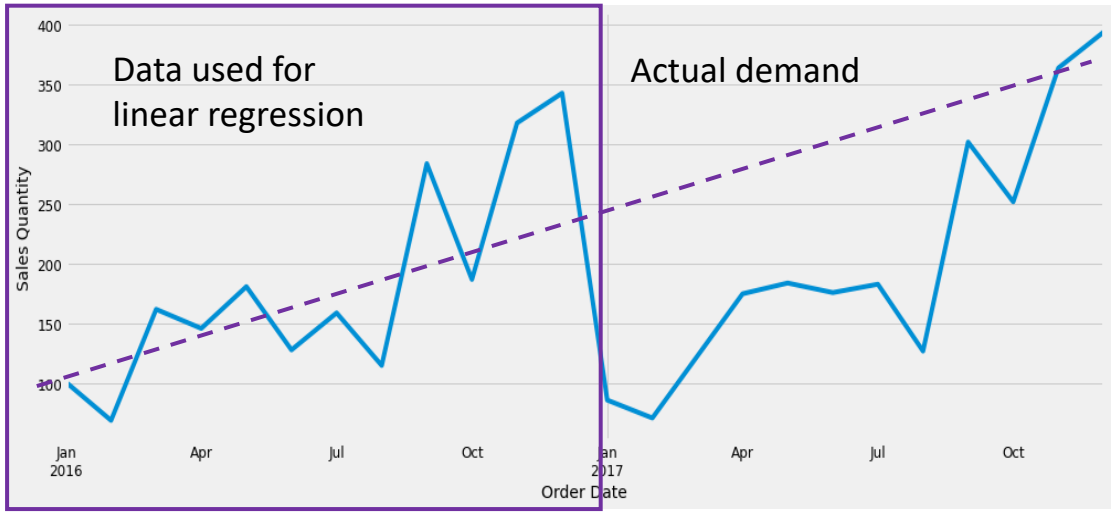


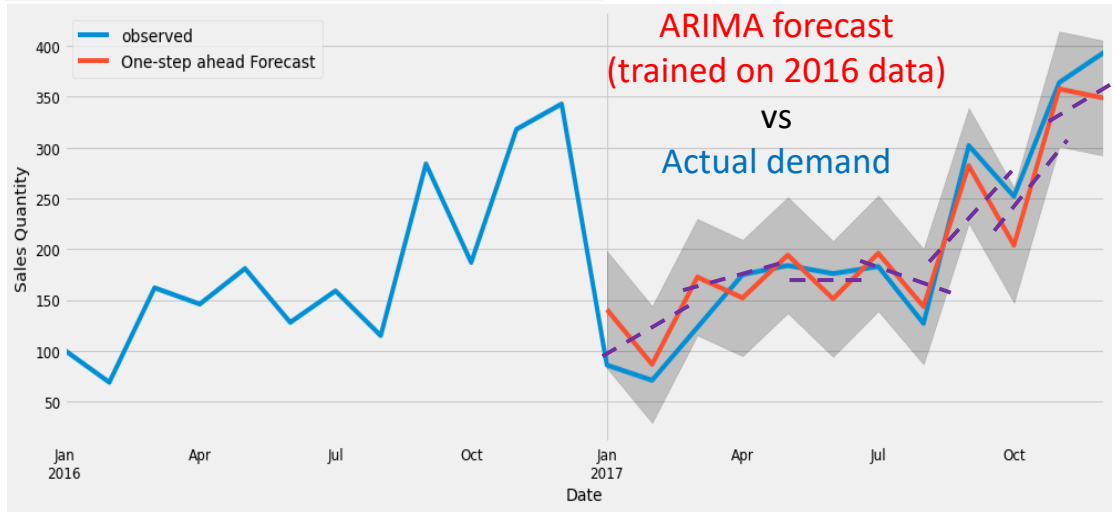

Solution 5: Derive linear regression of Annual demand

```
Monthly demand: 364.00
Lead time demand (3 month basis): 1092.00
Safety stock: 437.16
Reorder point: 1529.16
EOQ: 1427.89
Annual ordering cost: 1529.53
Annual holding cost: 1606.38
```

# Inventory Management & Forecasting

What if the demand looks like this?

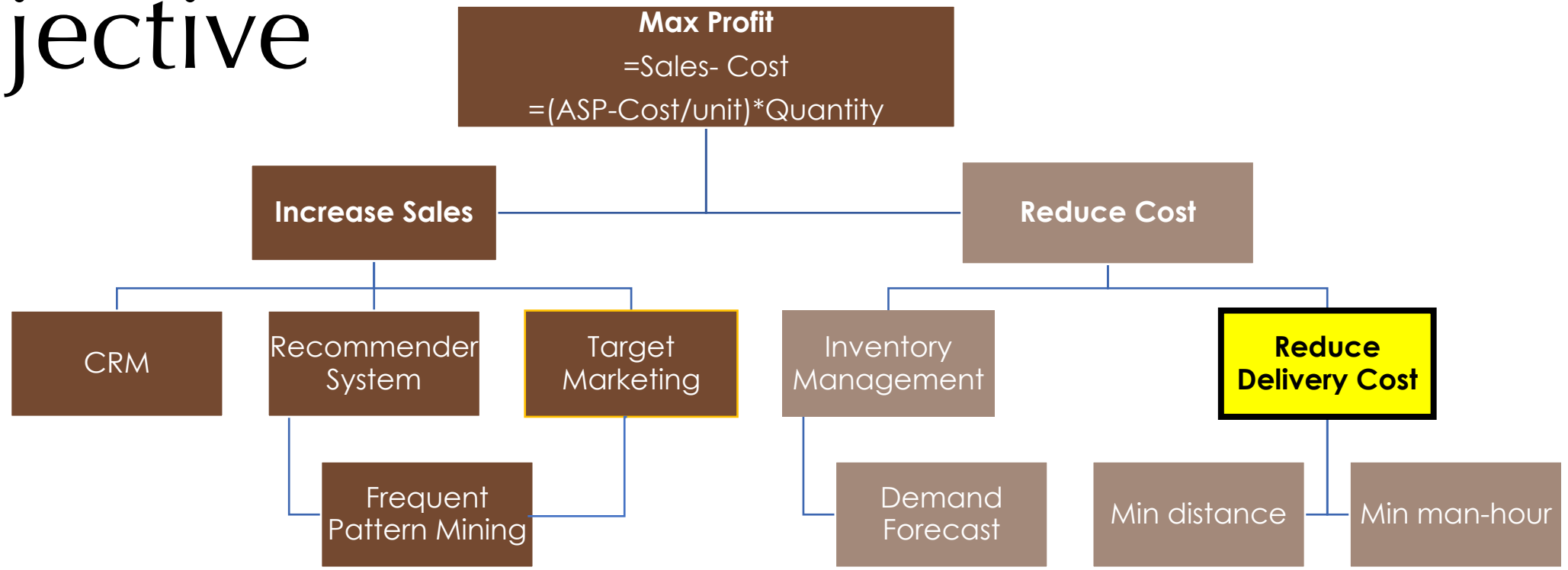



| | Monthly demand | Lead time demand | Safety stock | ROP | EOQ | Annual ordering cost | Annual holding cost |
|---|---|---|---|---|---|---|---|
| **ARIMA** | 260.84 | 939.63 | 252.39 | 1192.02 | 1260.70 | 884.61 | 663.45 |
| **EOQ** | 364.00 | 1092.00 | 437.16 | 1529.16 | 1427.89 | 3134.96 | 783.74 |

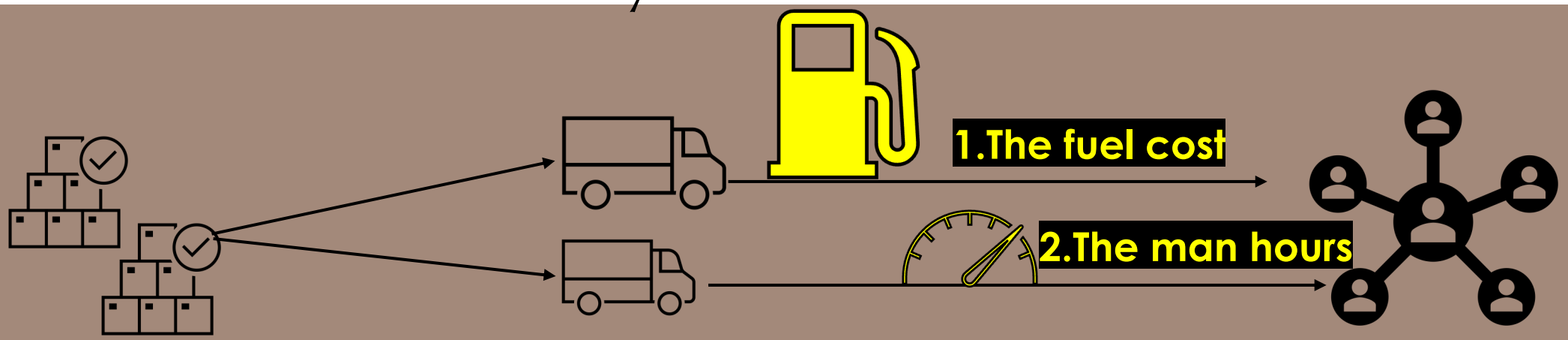
**Final solution:**
Apply a more appropriate forecasting technique like AutoRegression Integrated Moving Average (ARIMA) and compute EOQ, ROP and SS on a seasonally basis rather than yearly

# Objective




1. How can Castlery use more targeted marketing to improve revenue?
2. How can Castlery manage their inventory to improve revenue and reduce cost?
3. **How can Castlery manage their deliveries more efficiently to reduce cost?**

# The Last Mile Delivery Problem



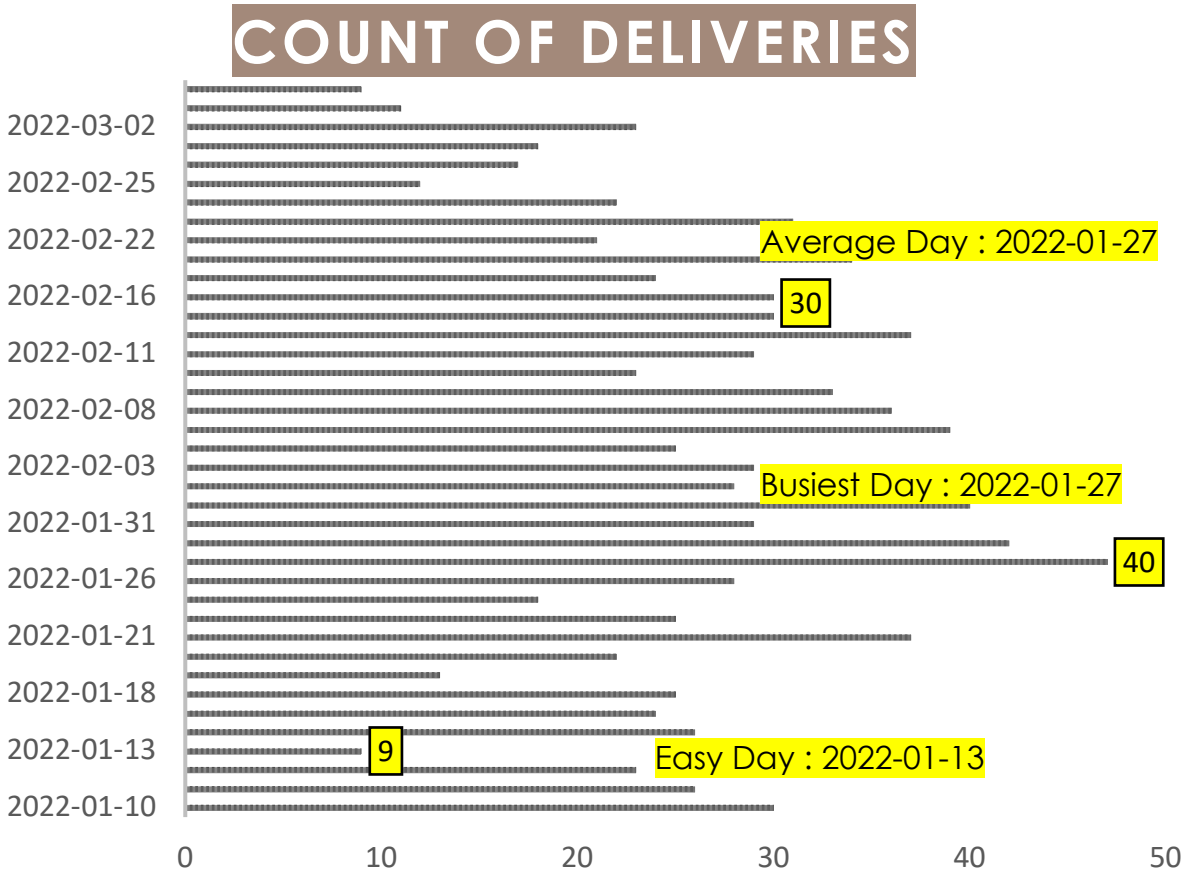

**Assumptions:**

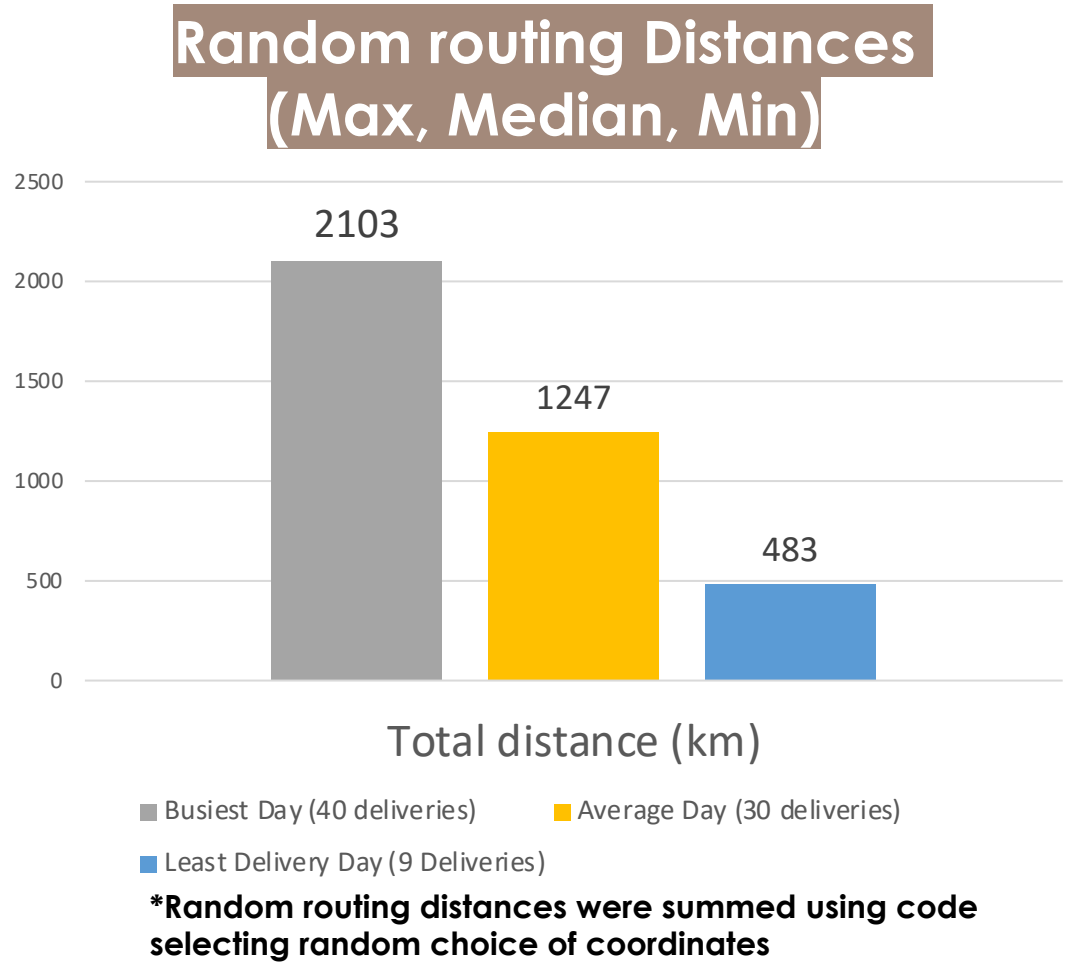
We assume that by reducing total distance travelled, it reduces the total number of hours needed to pay the drivers and reduces the fuel cost from travelling less which translates to cost savings

***How can we reduce the total distance travelled?***

# Current Delivery Situation

How does delivery look like on a busy day, an average day and an easy day?

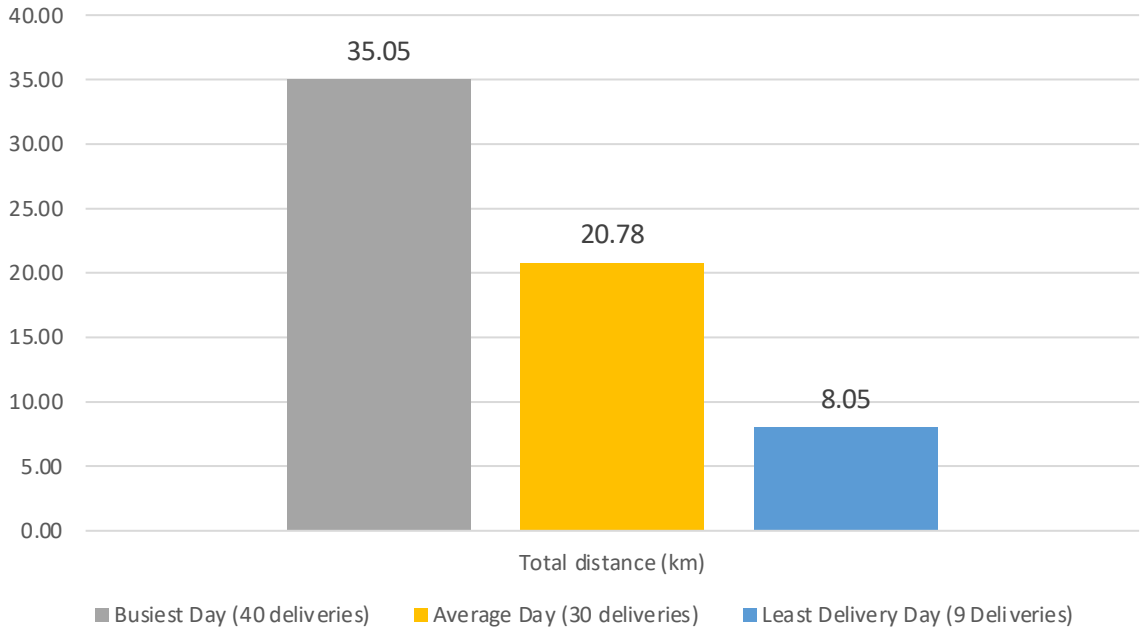


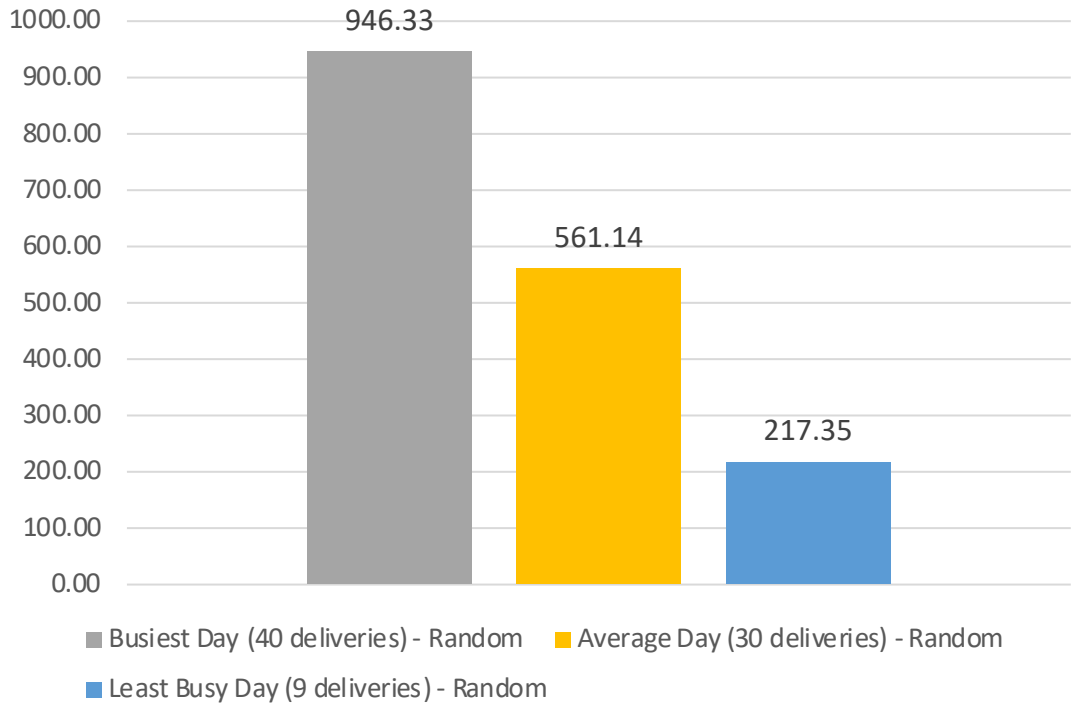

***Random routing distances were summed using code selecting random choice of coordinates**

# How does non-optimized routing translate to the number of man hours and diesel costs incurred?

## Random routing (Number of Man Hours)

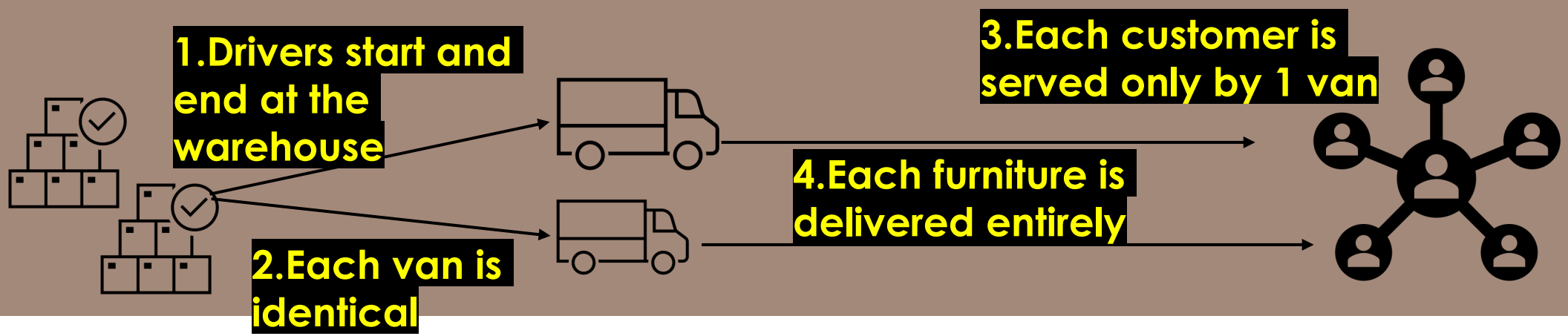



## Random routing (Diesel Cost)




Diesel cost: 0.45 USD/km

# Problem Modelling and Assumptions




| A Greedy Approach | Heuristic Search Method |
|---|---|
| • At each delivery decision point, choose the next location that is closest to current distance. | • Search for a route order that ensures cargo space is within each van's cargo limit while reducing total distance travelled by all vans |
| • Sum of all minimum distance travelled will lead to a minimum total distance | |

**5.Castlery has an existing fleet of delivery vans and can increase, decrease number of vans used without additional cost**

# Solution: Capacitated Vehicle Routing Problem

**Clarke & Wright Savings Algorithm:**

- Identifies a sequence of route for drivers to follow such that it minimizes the total distance travelled, all customers are served, and the delivery items does not exceed each van's capacity
- Calculates the distance savings if we are to combine delivery locations into 1 delivery route and plans optimal route

| Step | Category |
|---|---|
| 1 | • Calculate savings for each pair of distances *S(i,j)* when combined. Rank them in descending order |
| 2 | • Starting from highest savings<br>• Add sequence *i-> j* into delivery route |
| 3 | • Ensure neither *i* or *j* are already in the route<br>• otherwise skip to the pair with next best savings |
| | • If existing route ends with j, the next pair of sequence added will start with j |
| | • If vehicle capacity constraint is not exceeded |
| 4 | • Process step 2 to 3 until *S(i,j) list is exhausted and stop* |

## Eg: 2-delivery location example

***Total distance travelled without combining 2 delivery locations into 1 route***

$= 2 * \sum_{i=1}^{n} D(Depot, i)$

***Total distance travelled combining 2 delivery location into 1 route***

*= D(Depot , i) + D(i, j) + D(j, Depot)*

***Distance Savings between not combining and combining delivery locations into 1 route, S(i,j)***

*= [2* D (Depot, i) + 2* D(Depot, j)] – [D(Depot , i) + D(i, j) + D(j, Depot)*

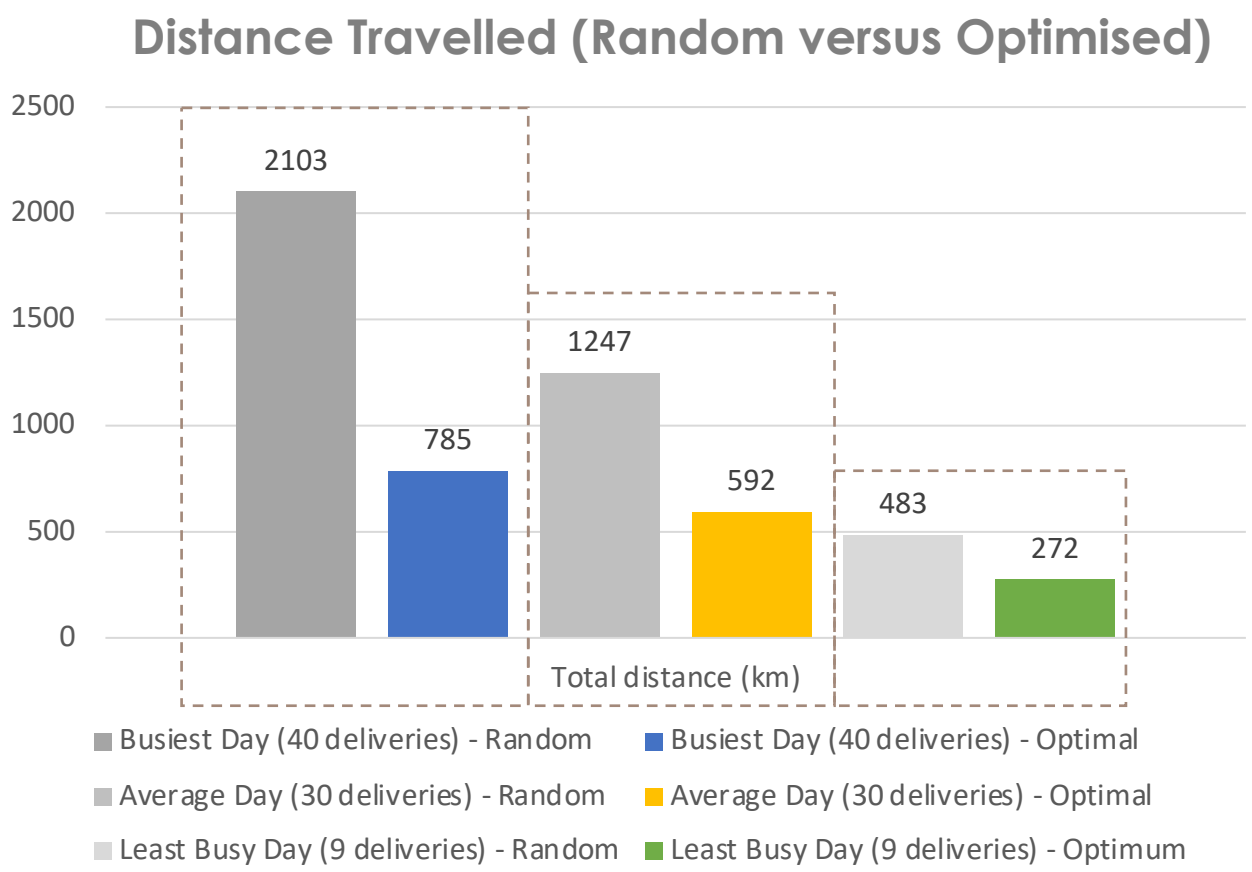
# Route Optimisation Results

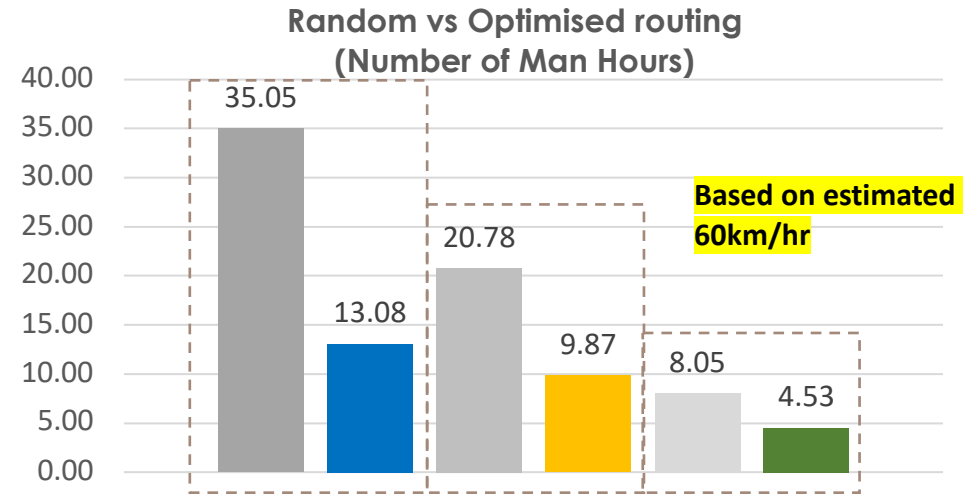
- Used VRPy library
- Distance is computed using GoogleMaps API to more accurately compute the driving distance between the delivery locations

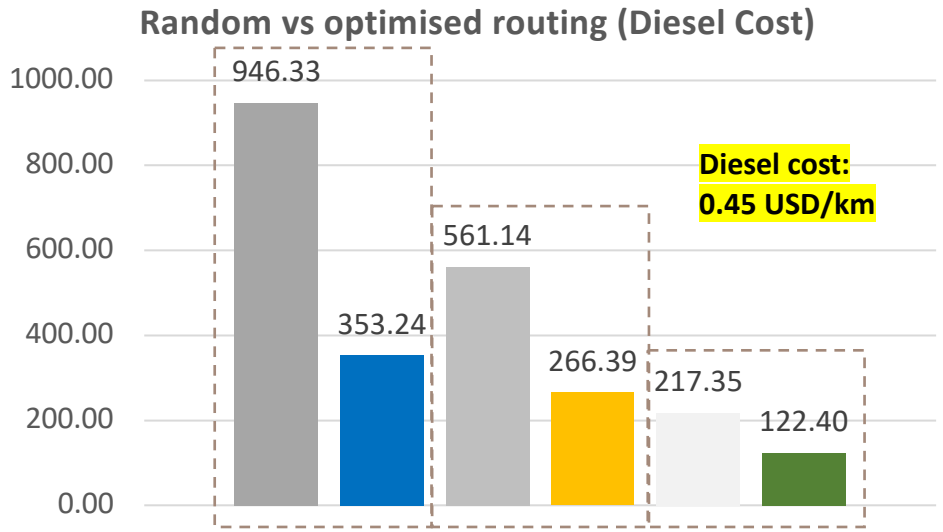
| | Minimimal Total Distance Travelled | Number of Vans Needed | Optimal Route | Total Distance Travelled By Each Van | Capacity Used for Each Van | Delivery Job Date |
|---|---|---|---|---|---|---|
| 0 | 752 | 3 | {1: ['Source', 20, 10, 21, 47, 46, 19, 45, 44, 9, 18, 13, 12, 11, 14, 6, 3, 23, 7, 'Sink'], 2: ['Source', 36, 38, 37, 35, 34, 33, 39, 42, 40, 8, 41, 26, 5, 27, 32, 29, 31, 16, 'Sink'], 3: ['Source', 2, 24, 4, 30, 1, 22, 25, 28, 17, 43, 15, 'Sink']} | {1: 307, 2: 210, 3: 235} | {1: 19.9906729, 2: 19.968116600000002, 3: 12.244240125} | 2022-01-27 |
| 1 | 644 | 3 | {1: ['Source', 8, 5, 4, 6, 30, 14, 'Sink'], 2: ['Source', 7, 12, 11, 28, 9, 27, 20, 13, 29, 'Sink'], 3: ['Source', 1, 16, 3, 25, 22, 26, 10, 2, 24, 21, 23, 19, 15, 18, 17, 'Sink']} | {1: 260, 2: 168, 3: 216} | {1: 8.889356900000001, 2: 8.717413899999999, 3: 13.160539025} | 2022-02-15 |
| 2 | 267 | 2 | {1: ['Source', 5, 6, 7, 1, 'Sink'], 2: ['Source', 2, 8, 9, 4, 3, 'Sink']} | {1: 133, 2: 134} | {1: 4.933046, 2: 4.3827484} | 2022-01-13 |

| For 27/1/22 orders: | |
|---|---|
| **Total distanced travelled by vans** | 752km |
| **Number of vans needed** | • 3 |
| **Van capacity** | • Van 1 capacity: 19.99 cbm used |
| | • Van 2 capacity: 19.97 cbm used |
| | • Van 3 capacity: 12.24 cbm used |
| • Van 1 will start from depot to order 20 then to 10 to 21 …… to 7 then back to depot | |

# Results – Random vs Optimised Routing



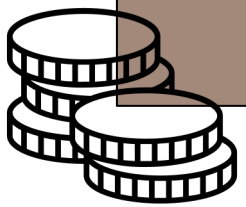



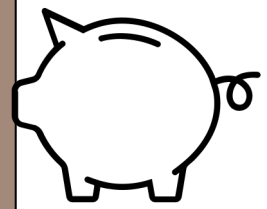
# Business Savings – Optimised vs Non-Optimised Routing

Cost Savings totals to

**$4004 USD** or **~57%**

of total cost!





| Factors | Optimised Routing | Non-optimised Routing |
|---|---|---|
| Total Distance Travelled (Km) | 1649 | 3833 |
| Manpower cost (23 USD/hr) | 632.12 | 1469.32 |
| Diesel cost (0.45 USD/km) | 742.03 | 1724.81 |
| Total Cost (USD) | 3023.15 | 7027.13 |
| Final Savings (USD) Total Cost (non-optimised) - Total Cost (optimised) | **4003.98** | |

# Last Mile Data Feedback Loop

- Collect data for different traffic conditions and for actual average speeds of different routes
- Fleet Management - Collect data for maximum and minimum number of vans needed
- Collect data on van maintenance costs which facilitates better vehicle fleet maintenance and planning
- Collect data on total delivery time for each driver - add in constraint to limit working hours
- Understanding the sales & delivery history and odd shaped cargo - to formulate the best method to fit in various furniture pieces that maximises the space

# Closed-Loop Data Eco-system

Sales History, Recommender System & Targeted Marketing

Intelligent Inventory Management & Forecasting

Delivery Optimisation, Routing Efficiency & On-Time Delivery

Customer Feedback & After-sale Service

# Data Governance

## Privacy, Ownership and Data Security Concerns

- Recommender systems rely largely on user data like purchase history. Customers may feel that they have a right to ownership and control over their own data
- Some customers may not even be aware that their data is being analysed, which can lead to trust issues with the organization later on.
- Risk that customer data could be misused, sold, or leaked, leading to privacy violations.

## Possible Strategies

- Transparency and consumer trust are essential in successful business practices
- For transparency: request customer consent to use their data and/or offering incentives to the customer
- Have a data management system that guards against data leakage, misuse. Secure user storage, clear data policies, anonymization of user data, staff training and awareness are some possible strategies

# Conclusion

- Employed a problem-solving approach in the areas of targeted marketing, intelligent inventory management and optimised vehicle routing solution for product deliveries.
- Combined with customer feedback, a close-looped data ecosystem should be utilised to enhance the process of continuous improvement and customer satisfaction.
- Thoughtful governance of user and operational data is essential for a successful business. By implementing clear data policies and a robust data management system, the organisation can gain consumer trust which leads to returning satisfied customers in the future.