# AMAZONE

*STUDENT ID:11349153*

## *ABSTRACT*

E-E-Commerce, the short form of electronic commerce is quite a buzzword that needs no introduction.It has almost become a part of daily lifestyle for most people these days.This digital marketplace is a dynamic and rapidly growing sector that enables businesses to expand their services to a global level thus generating increased revenue and more customers.It covers a wide spectrum of marketplace starting from local retailers to online product repositories and keeps growing periodically.Its rise has forced all the e-commerce firms to bring innovations into their main picture.

## *INTRODUCTION*

In this coursework, we have tried to simulate something similar to an e-commerce platform involving some of the stages that an official e-commerce firm goes through.The background for our system mainly focused on this.

- *Amazon is partnering with UK grocery retailer Morrison to launch same-day and instant grocery delivery services in Manchester.*
- *The development team has been asked to incorporate the same into an existing business model.*
- *Fresh products are classified into bakery, drinks, and fruits and vegetables, while other products include books, CDs, mobile phones, and home appliances.*
- *Partners (delivery drivers) have attributes like location, activity status, and details for order assignment and payouts.*
- *Customers can provide ratings for products which can be crucial for predicting ratings and generating quick product recommendations upon customer login.*
- *Daily inventory levels maintained for warehouses.*

The final schema should be flexible and encompass all these points.

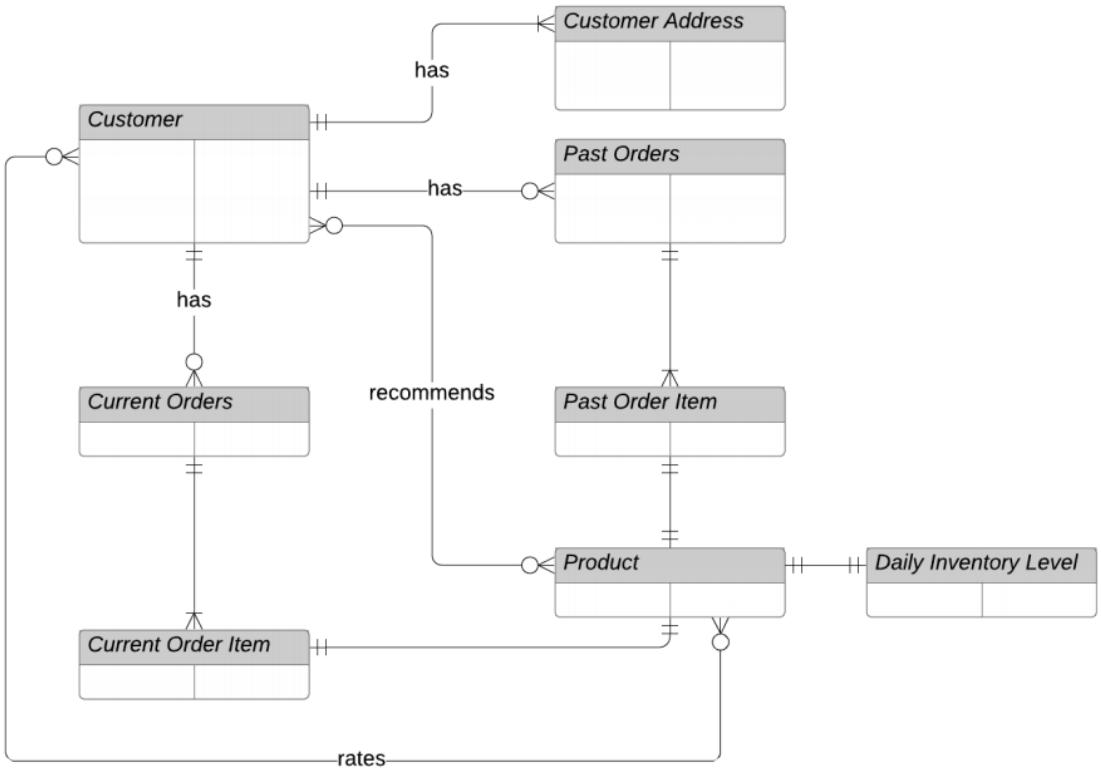
## *APPROACH*

Based *on the brief provided we decided to create below collections*

- *Customers- Customer details like orders,address,name,gender,age,payment details*
- *Products- Product specific attributes and ratings with supplier cost*
- *Partners-Details about delivery partners*
- *Stores- Stores name,address and products available in each store*
- *Daily_inventory_levels- 2 day records of inventory stocks in warehouses*
- *Financial Summary- A summarized report of how the company is performing financially*

## FINAL SCHEMA


## SCHEMA BEFORE



The workflow for our coursework followed below steps:-

1. Designing final schema diagram
2. Creation of “products” collection - Rakshit
3. Creation of “customers” collection - Dylan
4. Creation of “stores” collection - Joe
5. Creation of “Partners” collection - Pol
6. Creation of “daily_inventory_levels” - Abhinandan
7. Loading sample data into all collections for testing
8. Formulating queries as per brief
9. Creating indexes for quick access
10. Creating “financial summary” collection based on above
11. Testing with full data
12. Discussing and creating powerpoint presentation

## *MY ROLE*

In designing this e-commerce system “AMAZONE” my role was focused mainly on

I. Creating the collection for Daily Inventory Levels

II. Query to aid manager in checking the

- Sales
- Inventory Performance

III. Designing the final schema along with team members

IV. Creating the power-point presentation along with team members

Here I will explain in detail how I worked on each of the parts mentioned above

### Daily Inventory Levels

While creating the collection “daily_inventory_levels” there were certain assumptions that I had to make for our sample database.The first and the most important being taking two day time-span as a representative sample.In the brief, it was mentioned that AMAZONE had records for 5 years which is not possible to store in our demo database since it is built on a small scale.Hence the decision and other than that I decided to keep two Warehouses in two distinct geoJSON locations for simplicity as we aren't dealing with much data.The location points specified for each warehouse in turn, would also help the delivery partners in deciding the nearest warehouse for available products.They can decide among themselves who will take the responsibility on that basis.I have considered products only from “others” category for warehouses because they have longer shelf life compared to fresh products and those aren't a part of *same-day delivery* scheme of AMAZONE as well.Products have been embedded into this collection referenced by *product ids* from products collection displaying

the available quantity in each warehouse.Inventory needs to store records daily for monitoring stock levels or we can say that it involves regular write operations as a result I decided to keep this as a separate collection and not embed into *Stores* or *Products* collection.Main use of this collection most probably will be checking inventory performance at end of the day.So very few read operations compared to write, hence, embedding this won't be helpful and will make the other collection complex as well as increase retrieval time.Lastly, I decided to create an index explicitly on date because that's the most common and valid way of differentiating the records present in the collection and also it would make searching way quicker and easier for managers to check.

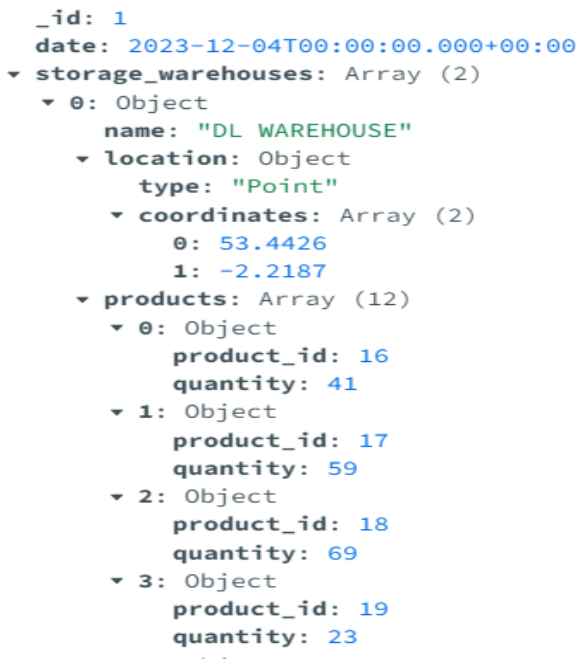



Fig.1 Daily_inventory_levels collection

*Challenges Faced*

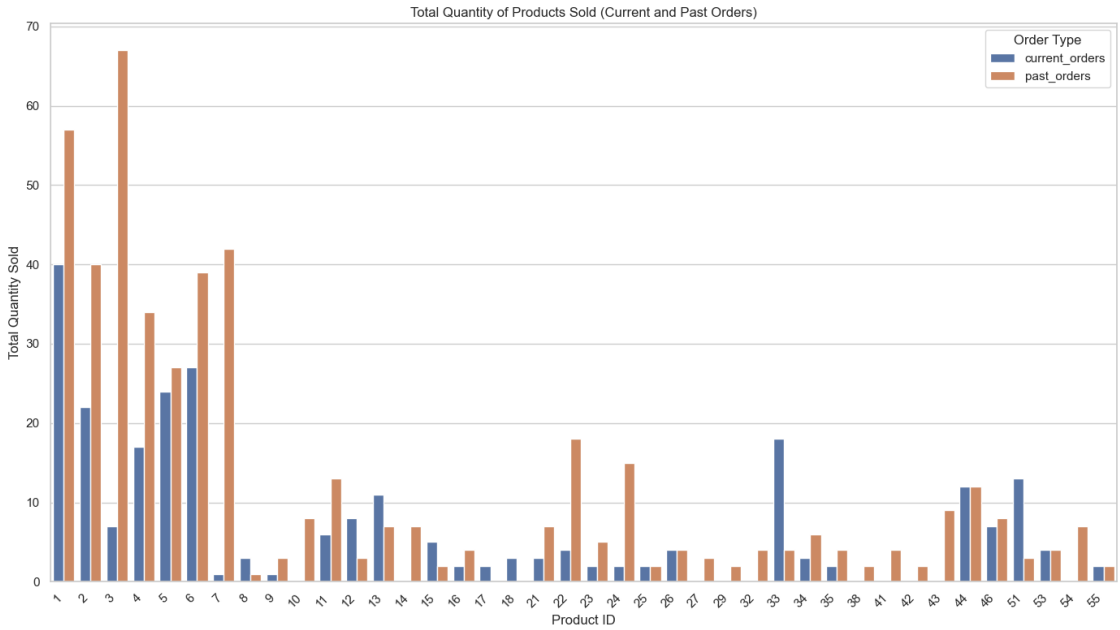
Creating this collection was not that tough.The confusing part was which products to consider and how many warehouses should we keep to make the records look realistic.Choosing the proper index for this collection was a tricky part because we could have indexed on the warehouses for searching as well but I did not do that since we are dealing with only 2 warehouses and those are getting updated daily so date made more sense.Other challenges faced were deciding the type of products to be kept in warehouses and for how many days since in real life such companies can have tons of records.

## Queries for Manager

### *Sales*

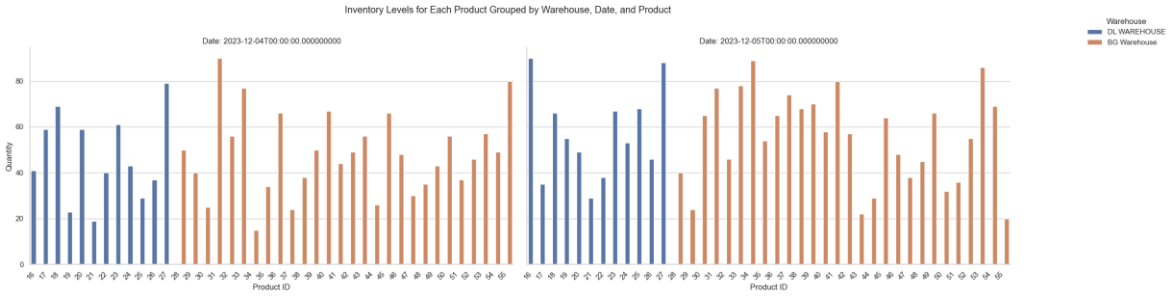
Here I had to make use of pymongo module to formulate this query since it involved visualization as well.So for that, after importing Pymongo first thing I had to do was connecting the python script to that db cluster.Coming to code I extracted order details for both past and current under customers collection and product details like id and quantity from products collection, followed by grouping of data by product id and order type.Then I

calculated the total quantity for each product and stored all these data in a pandas dataframe for plotting the graph.Used bar-plot feature of seaborn module for visualization.Finally,visualizations from this query can be utilized to optimize stocks for popular products and use marketing measures to promote the others.




*Inventory Performance*

Followed the same approach as above, difference being here I had to use the daily_inventory_levels collection created by me.I extracted daily inventory records for various products and storage warehouses like date,product id,quantity,warehouse name.I used catplot feature here from seaborn to plot the graph.The graph potrays quantity of products available in each warehouse on both the dates.I used a concise layout and rotated X labels for enhanced readability.Insights from this graph can be used to analyze temporal trends and tackle consumer demands by stocking products strategically in those warehouses depending on popular products in that area.




Challenges Faced

I didn't face any challenges as such in this except connecting database to python as I was not aware of that before.

## Schema & Power-point presentation

Regular meetings and discussions with team members were necessary to reach a consensus on the final design terms.

Some challenges faced in this process were deciding how to embed various features in collection so that database works perfectly and also which all features to be used for inter-collection referencing for ease of data retrieval.

## *LEARNINGS*

Doing all these tasks helped me get an idea of how an actual e-commerce system actually works in the back-end.Also learned how to tackle conflicts and ambiguities while designing schema for such applications as well as the assumptions that are required because client requirements are mostly ambiguous.

# The company has decided to expand its operations to some EU countries and has decided to set up an additional data center in Europe

*a) What type of replication algorithm/strategy will you suggest and why?*

In such cases, the best strategy to go with would be multi-Leader replication as it involves additional EU country.For that, maintaining multiple data centers is beneficial so that if one center stops working the replicated copies in other centers can makeup for it.Distributed data centers also reduces user latency and divides read and write requests for optimization.In case of conflict, we can decide by tracking request timings.

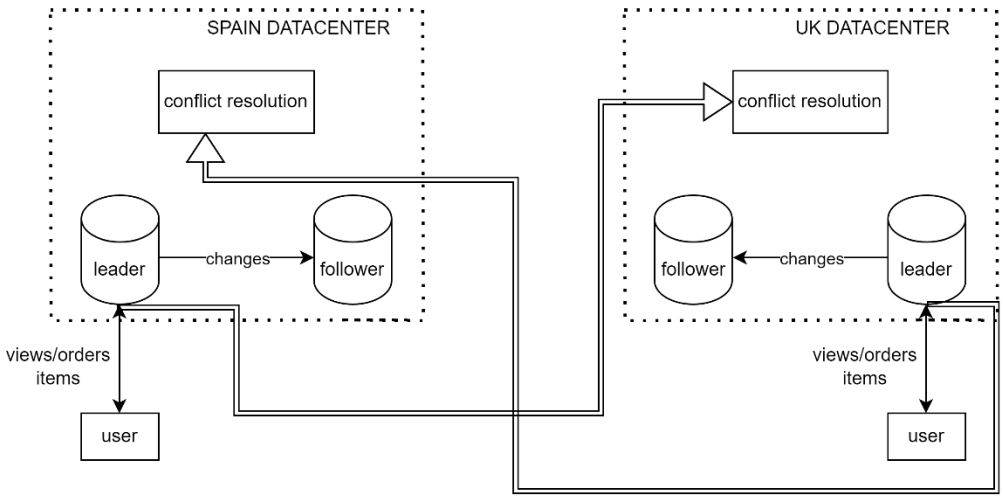



Fig.2 Multi-Leader replication in amazone

*b) What type of partition and allocation strategy will you suggest and why?*

I would suggest request routing strategy so that each data-center can handle all requests from that area.This would distribute requests evenly and prevent overload on one datacenter.This will also minimize user latency, improving performance and there will be a scope of future scaling.In that case data shifting wont be needed thus ensuring no downtime unlike rebalancing where data loading is required during failover.

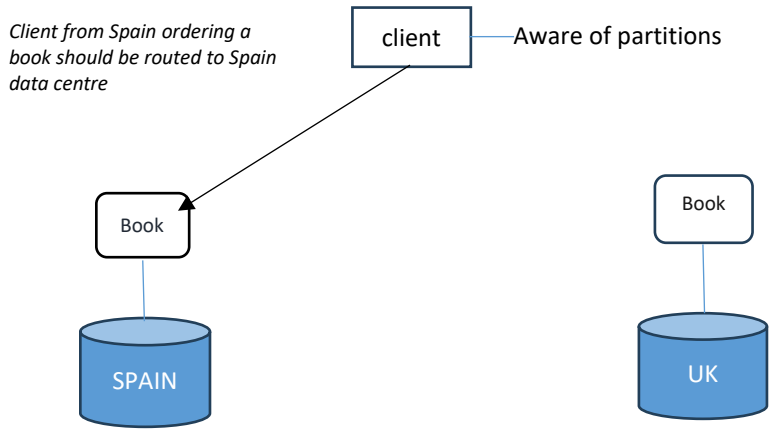



Fig.3 Request routing partitioning

*c) Will this operations expansion to Europe require a re-design of the database, and why?*

Yes, it would require database redesign since they are expanding their business to different countries which means increased request loads,country specific policies,varied consumer buying patterns.To accommodate all these we need to scale up infrastructure like adding datacenters,integrating systems with local API for loadbalancing also keeping in mind the country policies.To meet varying consumer patterns we need to study the product demand and supply pattern in that area to optimize inventory levels preventing downtime during peak season.This would enhance sales in that region.So, for redesigning the database we would also have to apply caching techniques for faster data retrieval and configure the indexing and querying strategy accordingly.

---