

# Wall Street Analytics Challenge 2023

Team Name : shaikabdulhadhi

College : B.V Raju Institute Of Technology.

## Team Members

1. U.Rakshitha
2. Vijaya Rahul Sivapu
3. Kumara Swamy Inti

# Milestone 1

## Under Stock Estimation

### **Approach**

The calculation of understock situations in different warehouses is performed by Comparing the products with their safety stock levels

### **Strategy**

If the stock value is less than the the safety value we considered as a flagged situation

### **Observation**

The most of the flagged cases are belong to the the understock condition



# Over stock Estimation

## Approach

For consideration of over stock detection we considered the complete stock data and added the stock data on the each date of warehouse and compare with the the capacity of the warehouse .

## Strategy

If the stock value is more than the warehouse capacity we considered it as a overstock case.

## Observations

We observed the overstock condition is very less compared to other conditions



# Out Of Stock Estimation

**Approach:** We have considered a situation where a warehouse has 1000 stock. If there are two simultaneous transaction happening where A wants 900 and B wants 1000. But, during the transaction B's transaction fail and the ordered is placed for the A. The output for B would be out of stock, even though there are 100 left over.

**Strategy:** We have considered the second instances of a warehouse of a particular product and checked if the quantity ordered is less than the current stock.

**Observations:** There were total 13 instances of out of stock



# Ranking of Warehouses.

## Approach

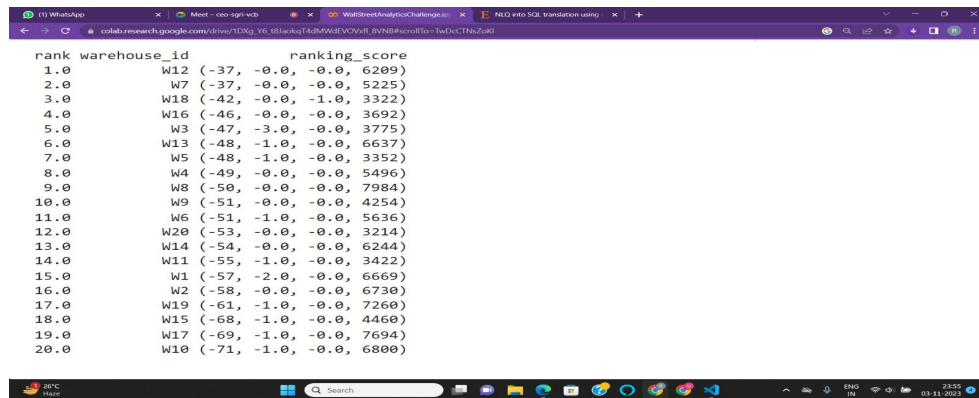
For ranking of warehouse we gone through all flagged cases consideration by adding the understock, overstock and out of stock situations.

The ranking of warehouses is done based on the correlation coefficients. To avoid conflicting ranks "warehouse capacity" is considered as the 4th parameter.

## Observation:

W12 is the best performing warehouse.

W10 is the worst performing warehouse.



rank	warehouse_id		ranking_score			
1.0	W12	(-37, -0.0, -0.0,	6289)			
2.0	W7	(-37, -0.0, -0.0,	5225)			
3.0	W18	(-42, -0.0, -1.0,	3322)			
4.0	W16	(-46, -0.0, -0.0,	3692)			
5.0	W3	(-47, -3.0, -0.0,	3775)			
6.0	W13	(-48, -1.0, -0.0,	6637)			
7.0	W5	(-48, -1.0, -0.0,	3352)			
8.0	W4	(-49, -0.0, -0.0,	5496)			
9.0	W8	(-50, -0.0, -0.0,	7984)			
10.0	W9	(-51, -0.0, -0.0,	4254)			
11.0	W6	(-51, -1.0, -0.0,	5636)			
12.0	W20	(-53, -0.0, -0.0,	3214)			
13.0	W14	(-54, -0.0, -0.0,	6244)			
14.0	W11	(-55, -1.0, -0.0,	3422)			
15.0	W1	(-57, -2.0, -0.0,	6669)			
16.0	W2	(-58, -0.0, -0.0,	6730)			
17.0	W19	(-61, -1.0, -0.0,	7260)			
18.0	W15	(-68, -1.0, -0.0,	4460)			
19.0	W17	(-69, -1.0, -0.0,	7694)			
20.0	W10	(-71, -1.0, -0.0,	6800)			

# Visualizations

We produce a line plot for each product in each warehouse showing it's stock levels and quantity ordered over a period of 2 years.

We highlighted 3 conditions.

They are under stock, out of stock and over stock.

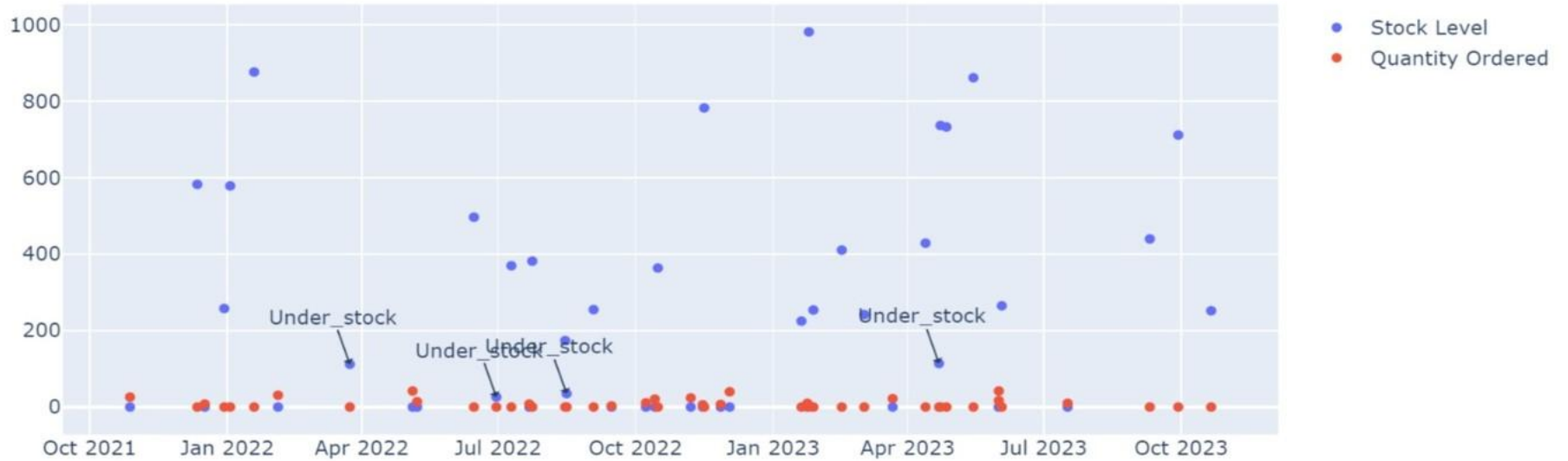
We used **plotly** library for visualizing the line plot.

We have created 2 dropdowns for the user to select a specific product in a specific warehouse so that, the specific product visualization is generated.



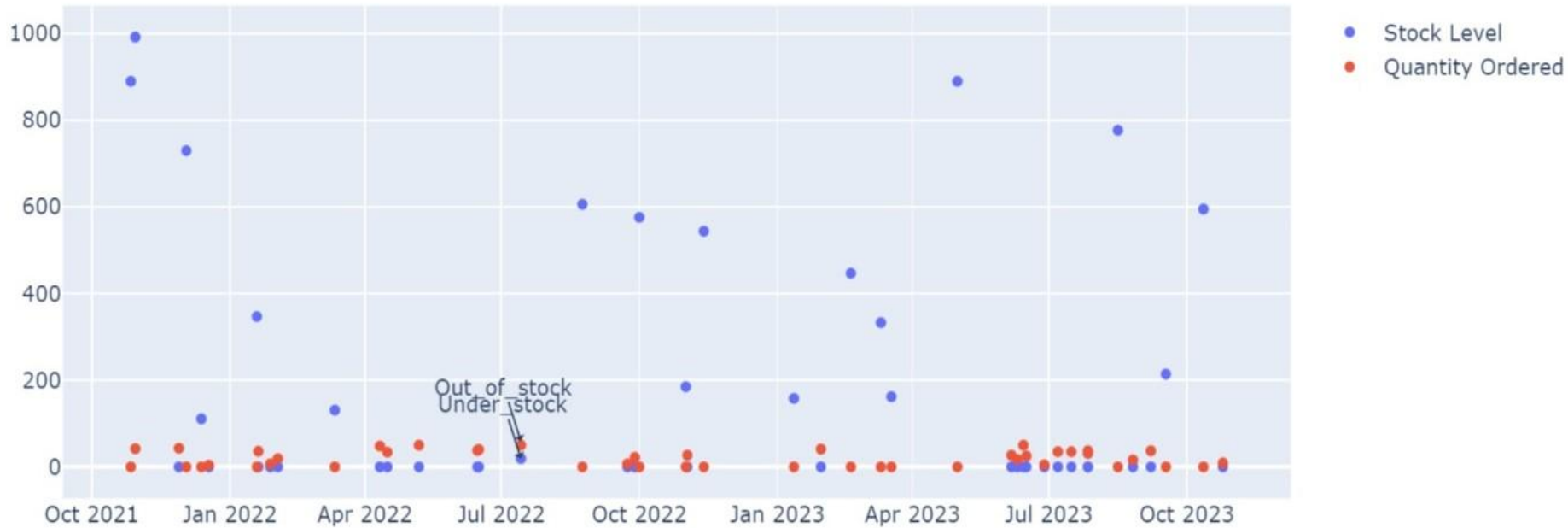
# Under\_Stock

Product P6 in Warehouse W13



# Out\_Of\_Stock

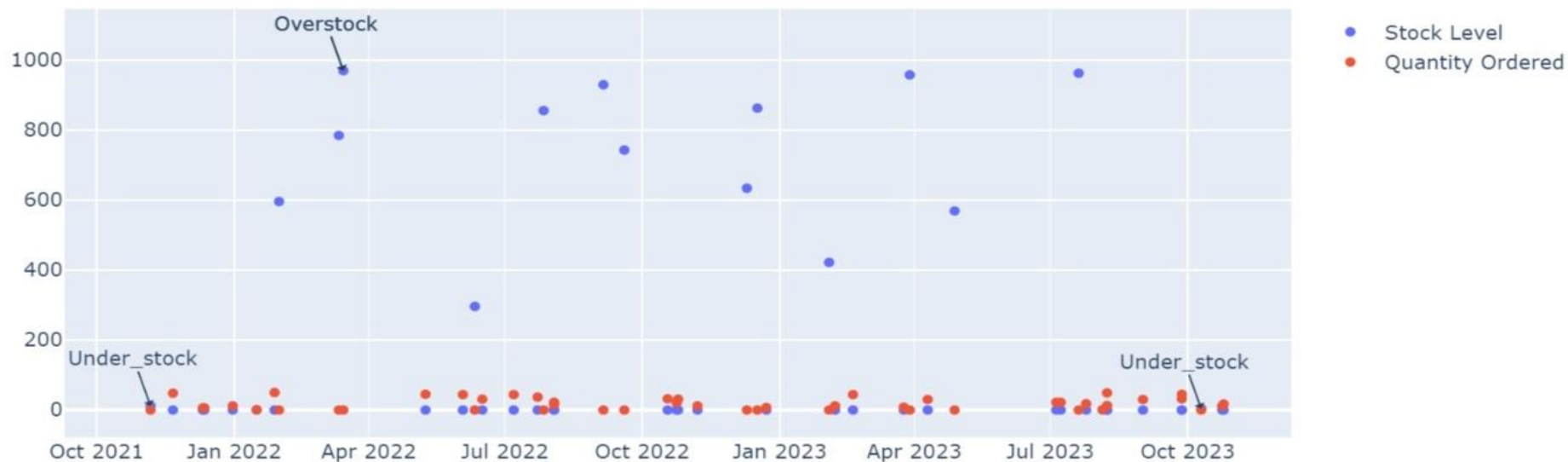
Product P10 in Warehouse W1





# Over\_Stock

Product P6 in Warehouse W18



# MileStone 2

In the first step where we have to convert the natural language query into a SQL query we have used the python library “spacy”.

We have used SingleStoreDB as our database.

In order to vectorize the dataset we have used Postman.

In the Postman, we have first set the the method as POST and then paste the url from the OpenAI.

The model we choose for the conversion is text-embedding-ada-002.

We then set it in JSON mode and give the model and input, and we send the request.

And based on the request we get the vector dataset.

For entering the queries we have used the streamlit library with the Langchain LLM.

For fine-tuning the model, where we can see a significant change in the model we need atleast 50-100 dataset.

The



# General setup of Postman

The screenshot displays the Postman application interface. At the top, there's a navigation bar with 'Home', 'Workspaces', 'API Network', and 'Explore'. A search bar labeled 'Search Postman' is also present. On the right side of the top bar, there are buttons for 'Invite', 'Upgrade', and a 'No Environment' dropdown.

The left sidebar is titled 'My Workspace' and contains a 'Collections' section. Under 'Collections', there's a 'Basic test syntax' folder containing 'GET pm.test' and 'GET pm.expect and pm.response'. Below this is an 'API tests' folder containing 'Authorization methods'. Under 'Authorization methods', there are several GET requests: 'Basic Auth', 'Google OAuth 2.0', 'Github Bearer token', 'Generate signed JWT', 'API Key', 'Hawk auth', 'Digest auth', and 'OAuth 1.0'. There's also a 'New Collection' section with a message 'This collection is empty' and a link 'Add a request to start working.'. Below that, there's a 'product information' folder containing a 'GET New Request' item. At the bottom, there's a 'warehouse\_name' folder with a similar message.

The main area shows a 'New Request' tab for a POST request to 'https://api.openai.com/v1/embeddings'. The request is titled 'product information / New Request'. The 'Body' tab is selected, showing a JSON payload:

```
1 {
2   ...
3   "model": "text-embedding-ada-002",
4   "input": "ToothPaste, ToothBrush, Shampoo, Conditioner, Soap, Towel, Toilet Paper, Sanitary Napkins, Deodorant, Hand Sanitizer, Face Wash, Face Cream, Face Mask, Hand Wash, Floor Cleaner, Dish Wash, Scrub, Brush, Comb, Perfume, Body Wash"
5   ...
6 }
```

The bottom status bar shows 'Body', 'Cookies', 'Headers (21)', and 'Test Results'. On the right, it displays '200 OK', '735 ms', '33.43 KB', and a 'Save as example' button.

# This is the vectorized data

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various collections and environments. The main workspace displays a 'POST' request to 'https://api.openai.com/v1/embeddings'. The response status is '200 OK' with a response time of '735 ms' and a size of '33.43 KB'. The response body is shown in the 'Preview' tab, displaying a JSON object with a list of 128 floating-point numbers, which are the vectorized data.

```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "index": 0,
      "embedding": [
        0.028696835, 0.019632576, 0.0055370517,
        -0.005024608, -0.023399368, 0.0038399987,
        -0.0081658205, -0.0019100166, 0.0039132047,
        -0.010481799, 0.029122762, 0.0031994444,
        -0.006665093, -0.00050578837, -0.009570049,
        0.0034839502, 0.033382032, -0.005849842,
        0.009237293, -0.020431189, -0.03213087,
        -0.0028117842, -0.009177397, 0.015439856,
        -0.0042825635, 0.014614622, 0.011207206,
        -0.023039993, -0.021349594, 0.0062258556,
        0.019605955, -0.021602489, 0.0049647125,
        -0.019765677, -0.010701417, 0.014761034,
        -0.01988547, -0.032317217, -0.029468829,
        -0.009530118, -0.015546338, 0.0153733045,
        0.0038932397, 0.01147341, -0.008458645,
        0.018594379, -0.011167275, 0.00042239155,
        -0.028057946, 0.000695459, 0.0088579515,
        0.004172754, -0.005287485, -0.0014366719,
        0.0037035688, 0.004621974, 0.01746301,
        0.0011879372, 0.009357085, 0.0041128583,
        0.0068281433, 0.008605057, -0.032876246,
        -0.016517984, -0.008511886, -0.01808859,
        -0.016145298, 0.008897883, -0.022600755,
        0.021549247, 0.026660372, 0.030081099,
        0.006994521, 0.007180864, 0.017289978,
        -0.01301074, 0.005650189, -0.0010664815,
        0.024464186, 0.003036394, 0.036416765,
        -0.011513341, -0.024890112, -0.013316876,
        0.027006438, 0.00074786803, 0.00013850948,
        0.04224664, 0.010348696, -0.010049216,
        0.0031229106, 0.017742524, 0.007966167,
        0.017263357, -0.024863493, 0.03921191,
        -0.028643595, 0.05185662, 0.0028300856,
        -0.029362347, 0.0070344517, 0.0103553515,
        -0.017662663, 0.00036894268, -0.015413236,
        0.0017868971, 0.014454899, -0.014627933,
        -0.0058764624, -0.015985575, -0.030507026,
        0.028440401, 0.017303286, -0.02045781,
        0.005370674, -0.0028999643, 0.009583359,
        -0.03024082, -0.01385594, -0.017649353,
        -0.0013634657, 0.01812852, 0.039451495,
        -0.012425091, 0.008318888, 0.0027951463,
        0.00039473124, 0.0031029452, -0.007946202,
        -0.01508048, 0.03508574, 0.00725407,
        0.023612332, -0.002079722, -0.00811258,
        0.003627035, -0.01746301, -0.00020058919,
        -0.008631678, -0.012105646, 0.0006621835,
        0.026766853, 0.0011272093, -0.0147077935,
        0.029468829, 0.019113477, 0.033435274,
        0.013543149, 0.01569275, 0.014375038,
        0.0146811735, 0.0012336911, -0.010302111,
        -0.017329907, 0.00903764, -0.003352512,
        0.012870983, 0.0054272423, -0.0070078312,
        -0.008006098, -0.0053240885, -0.028377391,
        0.02658051,
      ]
    }
  ]
}
```