



Database Management for Online and Retail Store

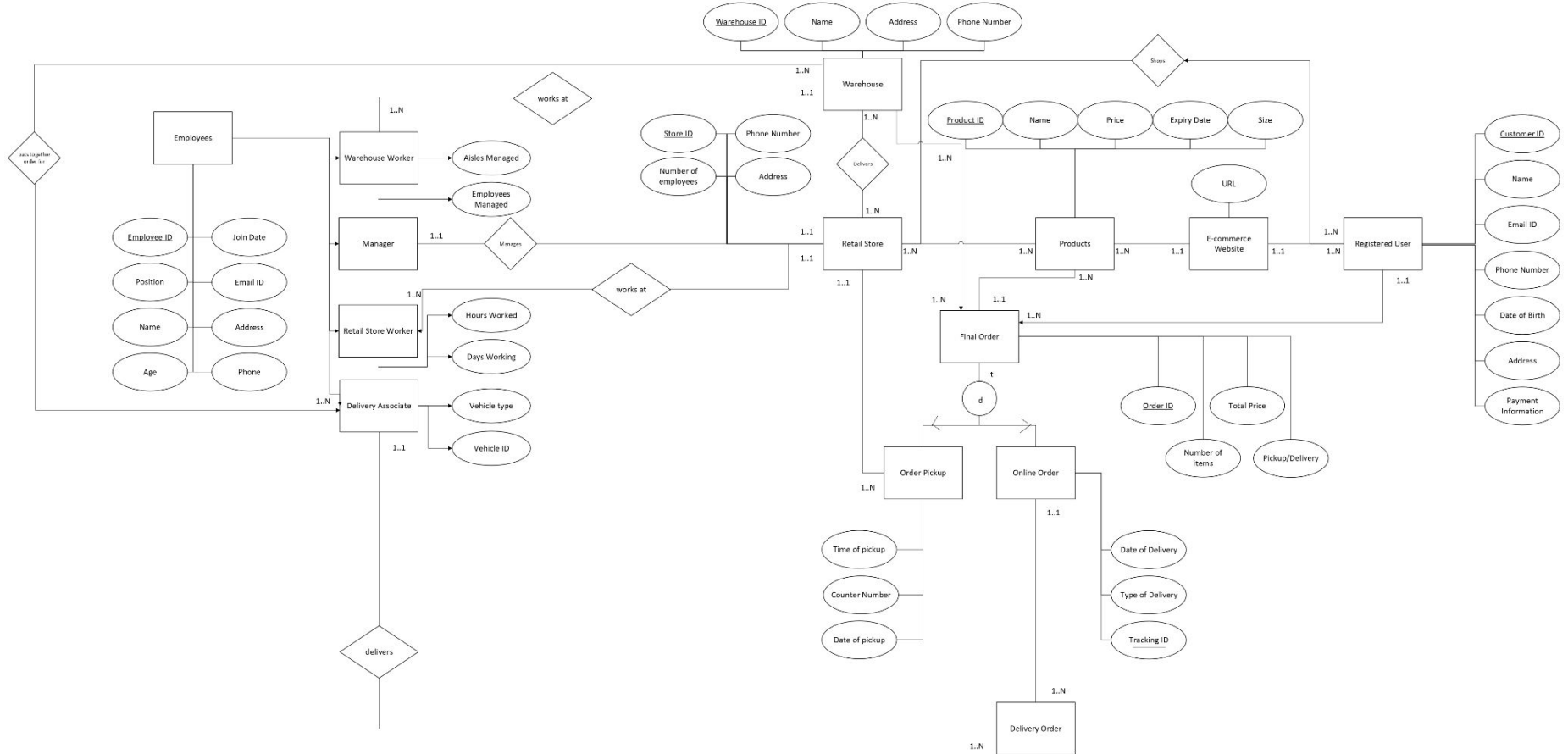
Group 28
Shrunali Salian
Siddhartha Setty

Problem Statement



- The business problem that is being implemented here, is for a database system of a retail and e-commerce store.
- Information about the Warehouse, Retail Store, the employees that are part of the entire operation.
- This database will also contain information about the e-commerce website, the products that are available at the retail and the e-commerce website, the final order and if it is an online order or an in person pick up of their order.
- This database also has information regarding the products in stock, the customer orders, the registered customer and also the employee information

EER Diagram of the Database



SQL Query-1

In a recent news it was said that there's swine flu in Colorado hence all meat products especially chicken have been contaminated, hence the supermarket has decided to recall all the chicken products sold to customers that came from Warehouse in Colorado. Retrieve the list of all the customers the supermarket should contact.

```
5
6 select user_ID,user_name,user_PhoneNo,user_email
7 from DMA_Project_1.Registered_User2
8 where user_ID in(
9     select user_ID from DMA_Project_1.Warehouse_Order as WO inner join DMA_Project_1.Final_Order as FO
10    where WO.finorder_ID = FO.finorder_ID and
11    WO.WH_ID in (select WH_ID from DMA_Project_1.Warehouse where WH_Name like "%colorado%") and
12    FO.finorder_ID in ( select finorder_ID from DMA_Project_1.Order_Product as OP inner join DMA_Project_1.Product as P
13    where OP.prod_ID = P.prod_ID and P.prod_name like "%chicken%"));
14
```

100% 1:113

Result Grid Filter Rows: Search Export:

	user_ID	user_name	user_PhoneNo	user_email
▶	28-2297938	Ashika Kalmadi	617-206-8766	ashika@ezinearticles.com
	18-2217938	Ishita Kalmadi	617-226-8766	ishita@ezinearticles.com
	28-2297123	Luna Potter	617-216-8766	luna@ezinearticles.com
	28-2297121	Harry Potter	617-006-8766	harry@ezinearticles.com

Result Grid

Form Editor

SQL Query-2

Registered user Samantha received her order however, there was one item missing in the order. She informed the customer care about the same. Retrieve the contact details of the warehouse the customer care should contact in order to check about Samantha's order.

```
20 select * from DMA_Project_1.Warehouse
21 where WH_ID in(
22     select WH_ID
23     from DMA_Project_1.Delivery_Associate
24     where finorder_ID in (
25         select finorder_ID
26         from DMA_Project_1.Final_Order
27         where user_ID in (
28             select user_ID
29             from DMA_Project_1.Registered_User2
30             where user_name like '%samantha%')));
31
```

100% 53:30

Result Grid Filter Rows: Search

Export:

	WH_ID	WH_Name	Wh_Phone	WH_address
▶	33-7488679	Rio Grande	461-656-7003	Nevada

Result Grid



SQL Query-3

Generation wise spending habits of registered customers

```
79 • Select
80   case
81     when year(user_DoB) between 1945 and 1964 then "Baby Boom Generation"
82     when year(user_DoB) between 1965 and 1980 then "Gen X"
83     when year(user_DoB) between 1981 and 1996 then "Millennial"
84     when year(user_DoB) between 1997 and 2010 then "Gen Z"
85     else "Silent Generation"
86   end as user_DoB , Avg(F0.finorder_totalprice) as Average_Money_Spent_Generation_wise
87   From DMA_Project_1.Registered_User2 as RU inner join DMA_Project_1.Final_Order_Data as F0
88   on F0.user_ID = RU.user_ID
89   Group by
90   case
91     when year(user_DoB) between 1945 and 1964 then "Baby Boom Generation"
92     when year(user_DoB) between 1965 and 1980 then "Gen X"
93     when year(user_DoB) between 1981 and 1996 then "Millennial"
94     when year(user_DoB) between 1997 and 2010 then "Gen Z"
95     else "Silent Generation"
96   end
97   ;
98
```

100% 2:98

Result Grid Filter Rows: Search Export:

	user_DoB	Average_Money_Spent_Generation_w...	
▶	Gen X	333.8181818181818	
	Gen Z	323.2857142857143	
	Millennial	304	

SQL Query-4

```
120 select prod_type as PRODUCT ,YEAR(prod_expiry_date)as YEAR,sum(prod_quantity) as SUM_OF_PRODTYPE
121 from Project_Data.Product
122 where prod_type like "%vegan%" or prod_type like "%meat%" group by YEAR(prod_Expiry_date),prod_type
123 Order by YEAR(prod_Expiry_date);
124
125
126
```

100% 1:125

Result Grid



Filter Rows:



Search

Export:



PRODUCT	YEAR	SUM_OF_PRODTYPE
Delii(meat)	2013	6900
Vegan(meat)	2013	80
Delii(meat)	2014	5100
Vegan(meat)	2014	100
Delii(meat)	2015	5000
Vegan(meat)	2015	500
Delii(meat)	2016	4500
Vegan(meat)	2016	3800
Delii(meat)	2017	4000
Vegan(meat)	2017	200
Delii(meat)	2018	3800
Vegan(meat)	2018	4000
Delii(meat)	2019	3500
Vegan(meat)	2019	4800
Delii(meat)	2020	1000
Vegan(meat)	2020	5100
Delii(meat)	2021	500
Vegan(meat)	2021	5500
Delii(meat)	2022	3106
Vegan(meat)	2022	6000

Result 8

Action Output

	Time	Action	Response
38	16:45:36	select prod_type as PRODUCT ,YEAR(prod_expiry_date)as YEAR,sum(prod_quantity) as SUM_...	20 row(s) returned

Year wise trend of vegan products vs meat products

NoSQL Query-1

Retrieve the list of all Customer names who use a VISA credit card.

```

yground Untitled-3  DMA_Project_MongoDB_Queries  ▶  □  ...
shru > Downloads >  DMA_Project_MongoDB_Queries

{first_name: "Leonard",
last_name: "Hofstadter",
payment_info: [
  {
    Credit_card_no: 123999001234,
    CVV: 897,
    expiry_date: "2032-11-23",
    company_name: "MASTERCARD",
  }
]},
});
// The $elemMatch operator matches documents that
// contain an array field with at least one element
// that matches all the specified query criteria.

db.mycustomers.find({
  payment_info:{
    $elemMatch: {
      company_name: "VISA"
    }
  }
})

Playground Result  ×
1  [
2  {
3    "_id": {
4      "$oid": "638fd3ccb8f267eadf2d93d6"
5    },
6    "first_name": "Lavina",
7    "last_name": "Pinto",
8    "payment_info": [
9      {
10       "Credit_card_no": 123456781234,
11       "CVV": 909,
12       "expiry_date": "2025-10-23",
13       "company_name": "VISA"
14     }
15   ],
16 },
17 {
18   "_id": {
19     "$oid": "638fd420d40b49d1ac8ba583"
20   },
21   "first_name": "Abu",
22   "last_name": "Salim",
23   "payment_info": [
24     {
25       "Credit_card_no": 123456781212,
26       "CVV": 900,
27       "expiry_date": "2025-01-23",
28       "company_name": "VISA"
29     }
30   ]
31 },
32 {
33   "_id": {
```


NoSQL Query-2

```
211 //iterate through customers
212
213 db.mycustomers.find().forEach(function(doc)
214 {print("Customer Name: "+doc.first_name)});
```

► Run Selected Lines From Playground

215

216

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Customer Name: John
Customer Name: Donald
Customer Name: Vijay
Customer Name: Susan
Customer Name: Miranda
Customer Name: Troy
Customer Name: Mike
Customer Name: Ashley
Customer Name: Lily
Customer Name: Amber
Customer Name: Troy

Iterate the list of all customer first names
using forEach

NoSQL Query-3

Retrieve the second highest customer spender

```
// MongoDB Playground Untitled-3 • DMA_Project_MongoDB_Queries • ▶ □ ... {} Playground Result ×
> shru > Downloads > DMA_Project_MongoDB_Queries
{ "item": "Dairy Products", "quantity": 350, "tags": [ "Mozarell",
{ "item": "Women's Fashion", "quantity": 15, "tags": [ "Skirts",
{ "item": "Furniture", "tags": [ "office", "home" ] },
{ "item": "Children's Toys", "quantity": 5, "tags": [ "Bikes", "(
] );

db.inventory.find( { quantity: { $nin: [ 5, 15 ] } }, { _id: 0 } )
//exists
db.inventory.find( { quantity: { $exists: true, $nin: [ 5, 15 ] } }

//regex
// SELECT * FROM products
// WHERE sku like "%789";

db.inventory.find( { tags: { $regex: "^cheese*" } } );

db.mycustomers.find().sort({totalbillamount: -1}).skip(1).limit(1);
```

```
1  [
2  {
3    "_id": {
4      "$oid": "6390dd5b7adca91f03f5e837"
5    },
6    "first_name": "Jeff",
7    "last_name": "Bezo",
8    "totalbillamount": 890
9  }
10 ]
```

NoSQL Query-4

```
Users > shru > Downloads > DMA_Project_MongoDB_Queries
392 //      _id: $first_name,
393 //      avg_totalbillamount: {$avg :"$totalbillamount"}
394 //    },
395 //  },
396 //  { $skip: 2}
397 // });
398
399 db.mycustomers.find().sort({totalbillamount: - 1}).skip(1);
400
401
402 use inventory;
403
404 db.inventory.insertMany( [
405   { "item": "Dairy Products", "quantity": 350, "tags": [ "Mozarella c
406   { "item": "Women's Fashion", "quantity": 15, "tags": [ "Skirts", "S
407   { "item": "Furniture", "tags": [ "office", "home" ] },
408   { "item": "Children's Toys", "quantity": 5, "tags": [ "Bikes", "Cri
409 ] );
410
411 db.inventory.find( { quantity: { $nin: [ 5, 15 ] } }, { _id: 0 } )
412
▶ Run Selected Lines From Playground
```

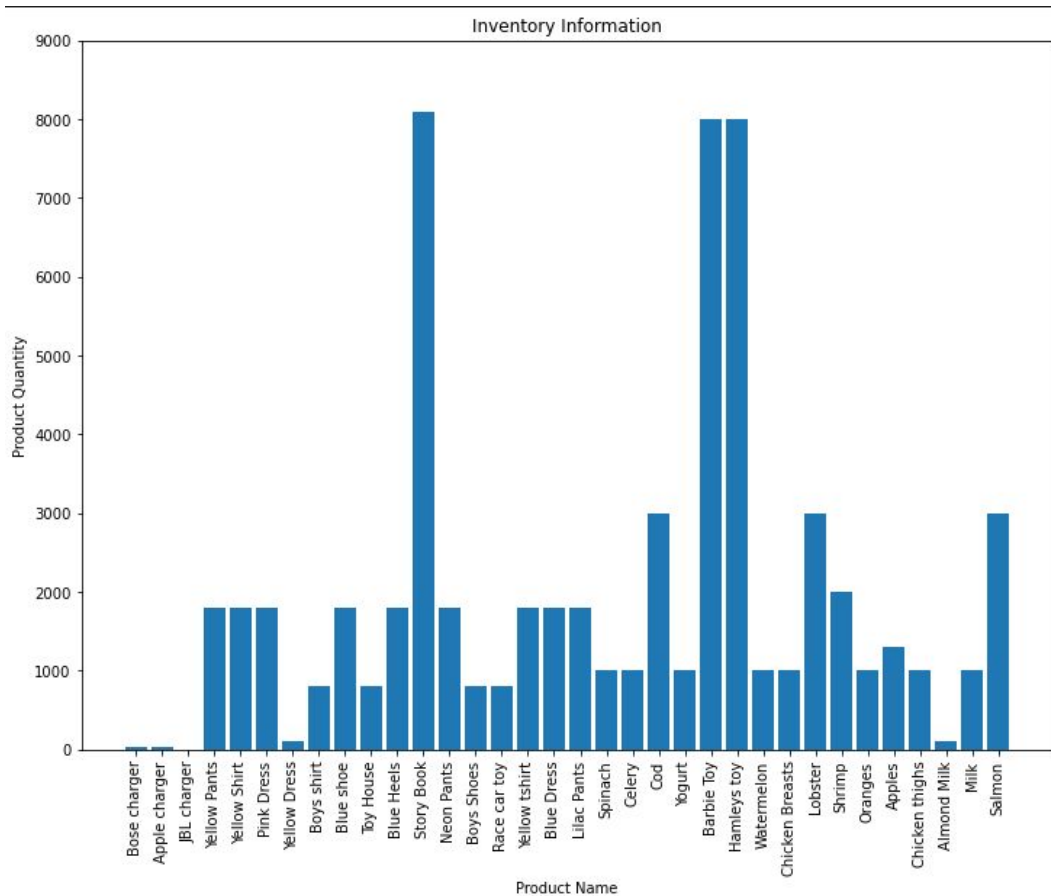
```
{ } Playground Result x
1 [
2   {
3     "item": "Dairy Products",
4     "quantity": 350,
5     "tags": [
6       "Mozarella cheese",
7       "cheddar cheese"
8     ]
9   },
10  {
11    "item": "Furniture",
12    "tags": [
13      "office",
14      "home"
15    ]
16  }
17 ]
```

Retrieve the product details of products that are not below the threshold quantity.

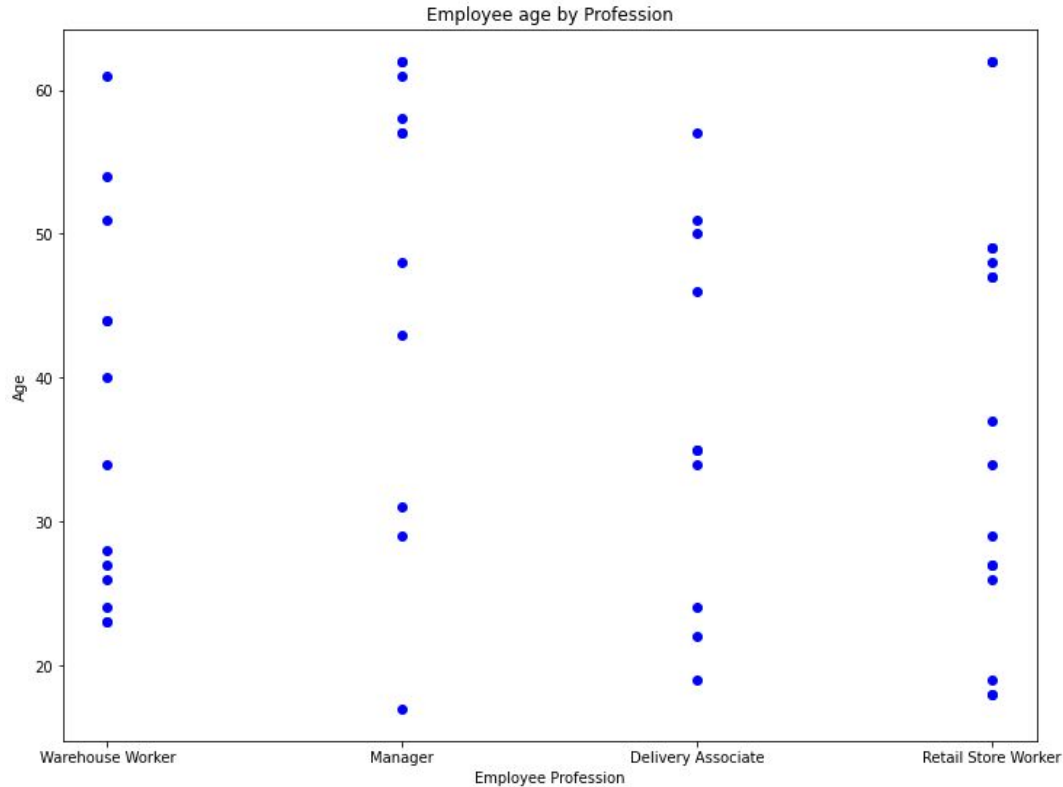
Visualization-1



The quantity of the each product stored in the warehouse inventory



Visualization-2

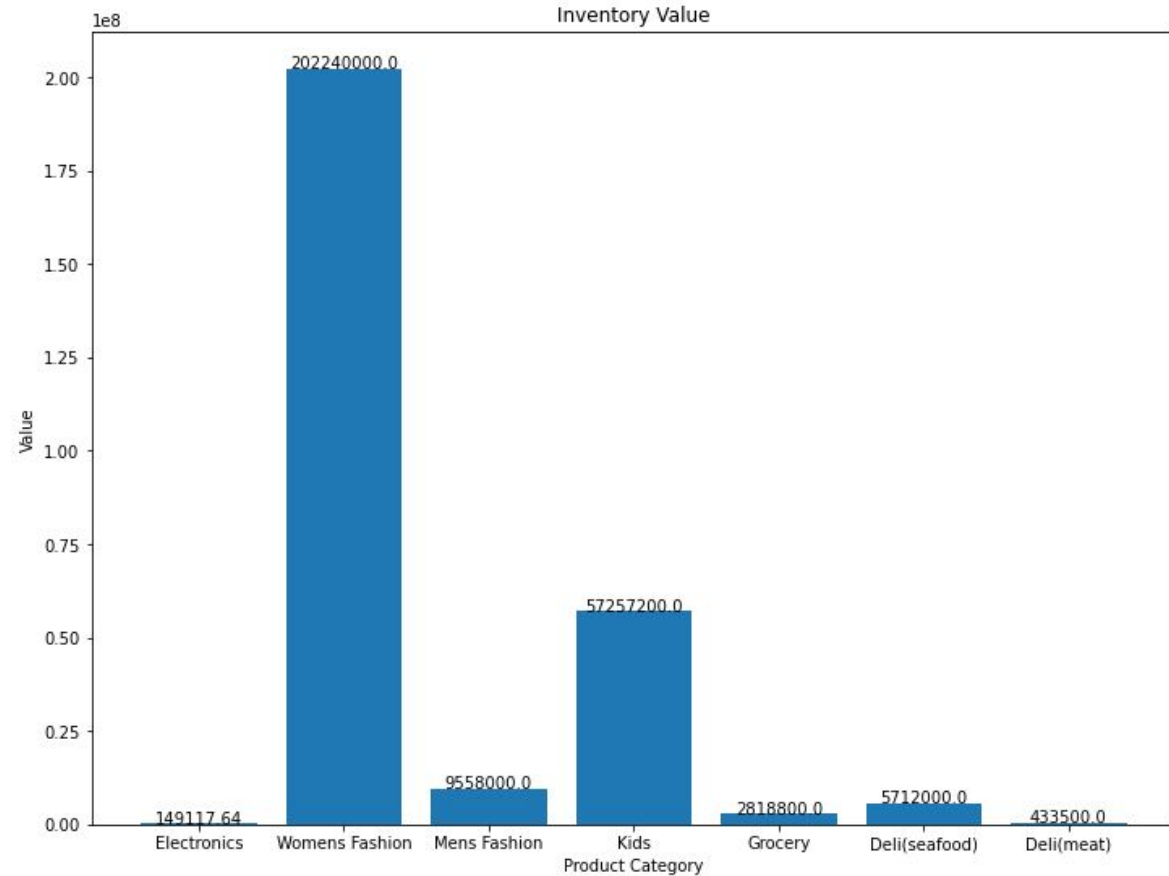


Scatter plot of the employee age distribution based on their profession/role

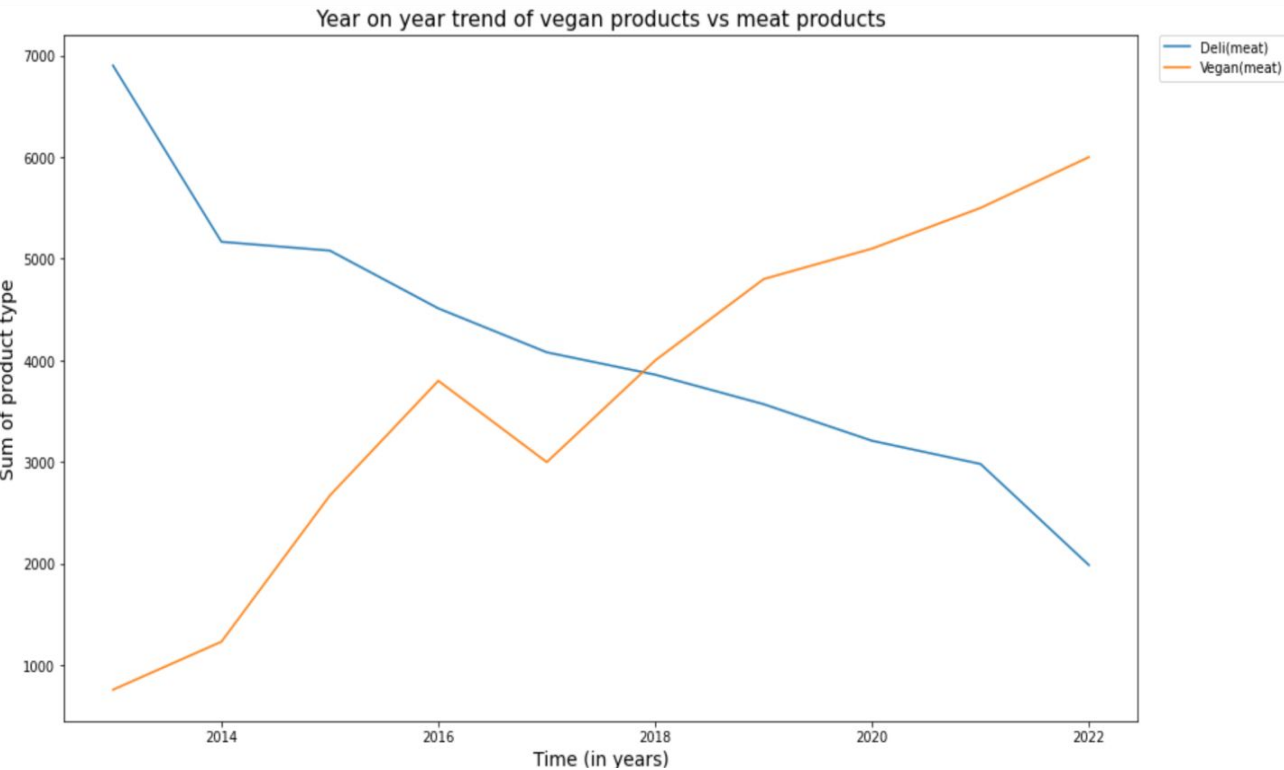
Visualization-3



The Inventory value for all categories of products in USD



Visualization-4



Year on year trend of vegan vs meat products



THANK YOU.