USE CASE STUDY REPORT

Group No.: Group 16

Students: Nethra Narayanan & Pratyusha Yellapragada

Executive Summary:

The project involved designing an inventory management database system for small to mid-size grocery chains that focuses on shelf restocking. We developed an automated system to aid employees of each grocery store to spike their overall labor productivity while also improving the sales efficiency. The EER and UML models were created, followed by a relational model is derived from the conceptual model by following the entity and referential integrity constraints and is normalized up to 3.5(BCNF) normalization to ensure the database system contains no redundancies and no insert deletion and update anomalies. This database was then implemented in MySQL by using DDL and importing the required data. DML was then used for querying and obtaining necessary results. The NoSQL implementation of this model was done in MongoDB. The SQL database was connected to python in order to generate insightful visualizations about product delivery timeline, labor demographics and shelf restocking guidance. Here, we plotted three types of charts which included bar charts, histogram and a pie chart depicting the difference in time between the order and delivery periods of a product and employee age tracking histograms.

I. Introduction

On our weekly grocery run, we hardly stop to think about how a store is able to consistently maintain fresh produce and groceries. We are left unaware of the organisation and exertion that takes place behind the scenes, especially all the manual labor that goes on overnight, so that we wake up to fresh produce every day. However, in this increasingly constrained labour market, several grocery retail chains face the challenge of managing the products on the shelves and keeping their customers satisfied. Legacy based paper processes and limited line of sight into critical data proves difficult for operators to make real-time decisions based on priority. Any improvements made to this scenario can positively improve productivity.

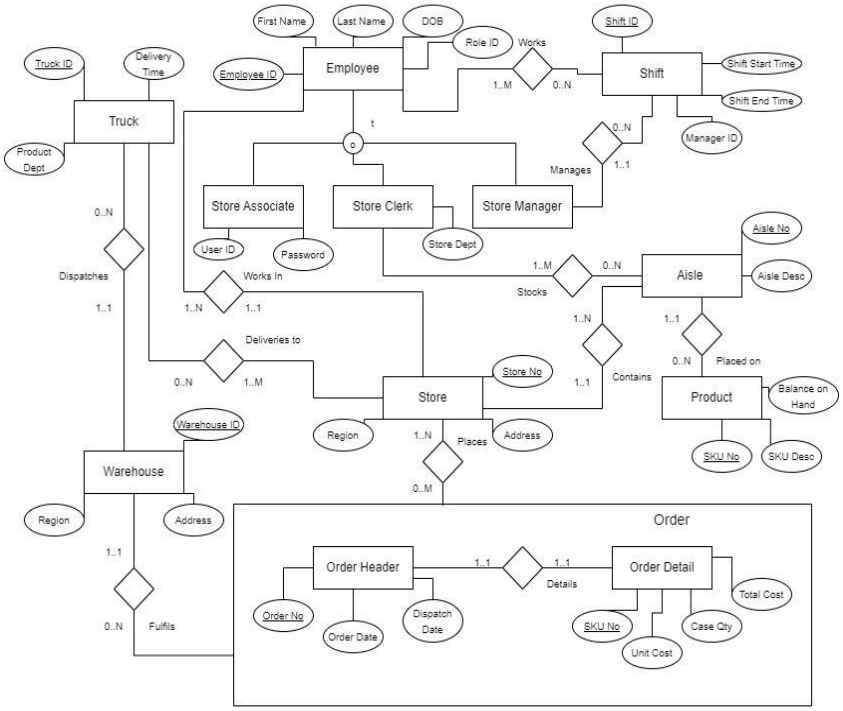
Our project aims to create a holistic system for product order placement, labor tracking, truck and delivery insights and shelf restocking guidance. The primary goal of this system is the integration of the database for all mentioned actions. We aim at providing an automated system to aid employees of each grocery store to spike their overall labor while also improving the sales efficiency

Requirements:

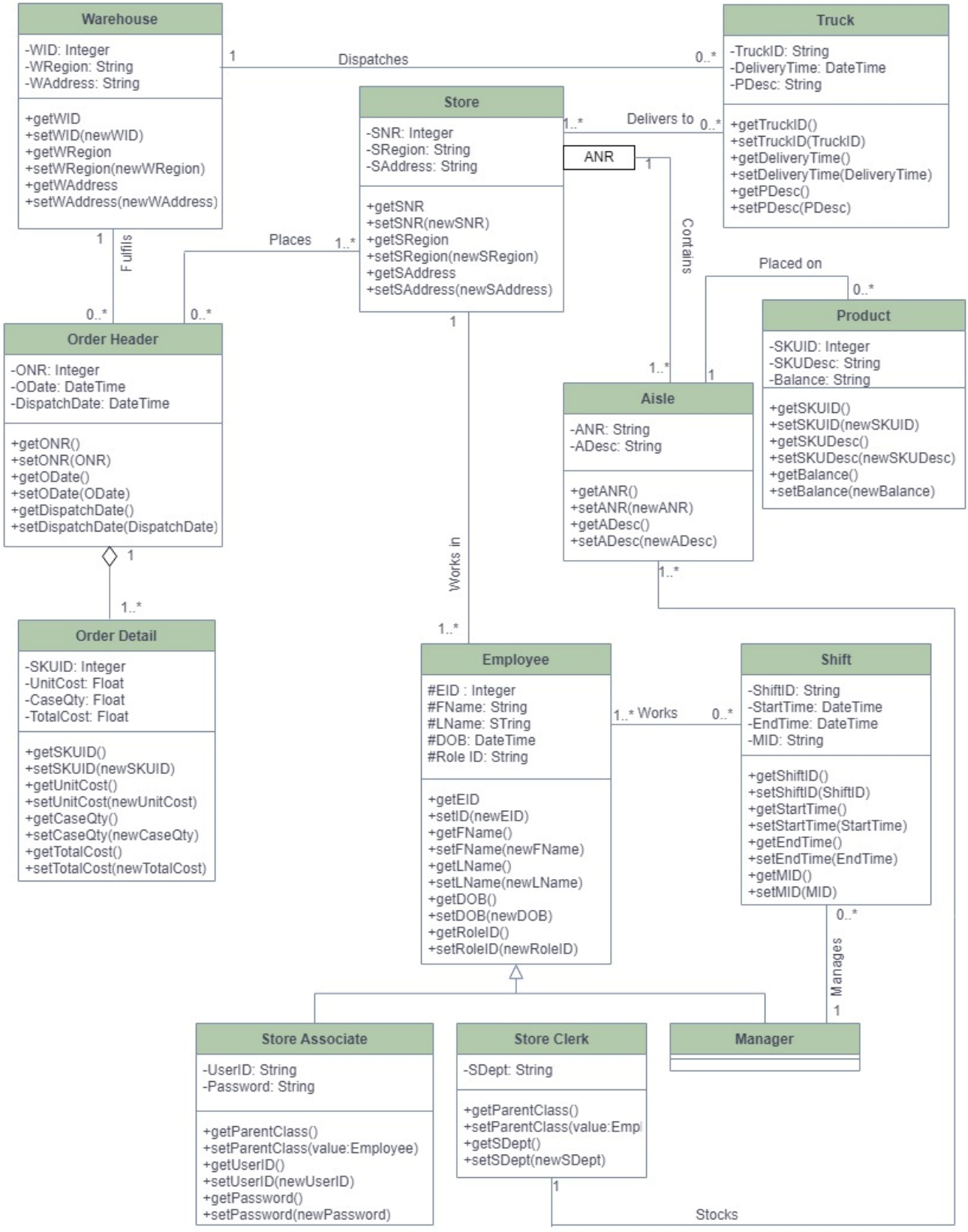
• Stores are responsible for making orders through the order placement system based on current balance on hand of items.
• Employee can log in to the store using their unique Employee ID to place an order.
• The system will track the balance for both the store and the warehouse and thus reserve details are always available to view.
• Once the orders are placed, it is recorded in Order Header and Order Detail tables.
• The Warehouse that the order is placed to will fulfil the order based on their reserves and will dispatch the order
• The schedule information of the truck that is delivering the product and the truck content information at a product SKU level is stored in the Database.
• The Truck information is recorded and can be seen/tracked by the employees at the store.
• A static table contains store planogram information of aisle numbers and descriptions. There is also a mapping of the product SKUs to different aisles to aid the employees in restocking the aisles.
• Employee details and shift schedules will also be stored.

II. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key- Underlined **Foreign Key- Bold**

Relational Model:

Truck (truck_id, delivery_time, product_dept, **warehouse_id**),
TRUCK_ID is the primary key for Truck relation
WAREHOUSE_ID is the foreign key for the Warehouse Dispatches Truck relationship type: NULL values NOT ALLOWED

Store (store_no, region, address,**order_id**)
STORE_NO is the primary key for Store relation
ORDER_ID is the foreign key for the Store places order relationship type: NULL values NOT ALLOWED

Delivers_To(**truck_id, store_no**)
TRUCK_ID and STORE_NO are the foreign keys for the Truck Delivers to the store relationship type: NULL values NOT ALLOWED

Warehouse (warehouse_id, region, address)

Employee (employee_id, role_id, first_name, last_name, date_of_birth,**store_id**)
EMPLOYEE_ID is the primary key for the Employee relation
STORE_ID is the foreign key for the Employee Works in Store relationship type: NULL values NOT ALLOWED

Store_Associate(employee_id, role_id, first_name, last_name, date_of_birth, user_id, password),
EMPLOYEE_ID is the inherited primary key and ROLE_ID is the primary key for this relation

Store_Clerk (employee_id, role_id, first_name, last_name, date_of_birth, store_dept),
EMPLOYEE_ID is the inherited primary key and ROLE_ID is the primary key for this relation

Store_Manager(employee_id, role_id, first_name, last_name, date_of_birth),
EMPLOYEE_ID is the inherited primary key and ROLE_ID is the primary key for this relation

Shift(shift_id,shift_start_time,shift_end_time,**role_id**)
SHIFT_ID is the primary key for the Shift relation
ROLE_ID is the foreign key for the Employee manages Shift relationship Type: NULL values NOT ALLOWED

Works(**employee_id,role_id,shift_id**)
EMPLOYEE_ID and SHIFT_ID are the foreign keys for the Employee works shift relationship type: NULL values NOT ALLOWED

Aisle (aisle_no, aisle_desc, **store_no**),
AISLE_NO is the primary key for the Aisle relation
STORE_NO is the foreign key for the Store contains aisle relation: NULL values NOT ALLOWED

Product (sku_no , balance_on_hand, sku_desc, **aisle_no**)
SKU_NO is the primary key for the product relation
AISLE_NO is the foreign key for the product placed on relation: NULL values NOT ALLOWED

Order (order_no, sku_no, **warehouse_id**)
WAREHOUSE_ID is the foreign key for Warehouse fulfils order relationship type: NULL values NOT ALLOWED

Order_Detail (sku_no, unit_cost, case_qty, total_cost)

Order_Header (order_no, order_date, dispatch_date)

Order_Placed (**store_no, order_no, sku_no**)
STORE_NO,ORDER_NO and SKU_NO are the foreign keys for the Store places Order relationship Type: NULL values NOT ALLOWED

IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

The database was created in MySQL and some of the queries that we performed are:

Query 1: Query to retrieve truck schedule information for trucks after 3rd Sept using inner join

```
select * from truck t inner join delivery d on t.truckid = d.truckid
where delivery_time  > '2022-09-03 10:06:00';
```

truckid	delivery_time	warehouseid	truckid	storeid
TR01	2022-09-21 13:32:00	1	TR01	4007
TR02	2022-09-16 14:37:00	2	TR02	3010
TR03	2022-09-22 14:21:00	3	TR03	4001
TR04	2022-09-15 13:55:00	4	TR04	2010
TR05	2022-09-14 16:15:00	5	TR05	3005
TR06	2022-09-10 15:32:00	6	TR06	2002

Query 2: Query to retrieve data from aisles having balance on hand less than average

```
select * from product
group by store_aisle_no having balance_on_hand < avg(balance_on_hand);
```

sku_no	balance_on_hand	shelf_capacity	sku_desc	store_aisle_no
1924	16	26	brandy	1001-10
1311	22	32	candy	1001-11
2193	11	21	beef	1001-12
1273	6	16	artif. sweetener	1001-14
1450	8	18	bags	1001-15
1923	13	23	beverages	1001-5

product 2 ×

Query 3: Query to retrieve data from 4 order tables using inner join for specific orders

```
select distinct o.orderid, o.sku_no, o.warehouseid, od.unit_cost, od.case_qty, od.total_cost,
oh.order_date, oh.dispatch_date, op.storeid
from orders o inner join order_header oh on o.orderid = oh.orderid
inner join order_detail od on o.sku_no = od.sku_no
inner join order_placed op on o.orderid = op.orderid
where o.orderid  in ('49288-0110', '50076-100','50436-6924','57344-154','63824-203');
```

orderid	sku_no	warehouseid	unit_cost	case_qty	total_cost	order_date	dispatch_date	storeid
49288-0110	2070	3	4	1	4	2022-09-21 07:38:00	2022-09-22 14:18:00	4001
50076-100	1273	2	3	1	3	2022-09-16 01:45:00	2022-09-16 12:32:00	3010
50436-6924	2525	5	6	1	6	2022-09-12 20:33:00	2022-09-13 12:27:00	3005
57344-154	1450	4	5	1	5	2022-09-10 20:30:00	2022-09-11 17:30:00	2010
63824-203	2421	1	2	1	2	2022-09-18 12:56:00	2022-09-18 15:35:00	4007

Query 4: Double nested query to retrieve employee information of all managers

```
select first_name, last_name, date_of_birth from employee where employeeid in (select
employeeid from works where shiftid in (select shiftid from shift where shiftid = "fullshift"));
```

first_name	last_name	date_of_birth
Renee	Billie	1978-07-09
Beilul	Tal	1987-03-28
Netty	Gettins	1983-04-08
Penny	Ogg	1977-01-29
Sammie	Mc Ilwrick	1991-03-10
Glori	Dongate	1996-02-24

employee 4 ×

Query 5: Correlated Query to get products with 3 lowest balance on hand percentages

```
SELECT P1.SKU_NO, P1.SKU_DESC, (P1.BALANCE_ON_HAND/P1.SHELF_CAPACITY*100),
P1.STORE_AISLE_NO
FROM PRODUCT P1
```

```
WHERE 3 >
(SELECT COUNT(*)
FROM PRODUCT P2
WHERE (P1.BALANCE_ON_HAND/P1.SHELF_CAPACITY*100) >
(P2.BALANCE_ON_HAND/P2.SHELF_CAPACITY*100));
```

	SKU_NO	SKU_DESC	(P1.BALANCE_ON_HAND/P1.SHELF_CAPACITY*10	STORE_AISLE_NO
▶	1273	artif. sweetener	37.5	1001-14
	2070	baby cosmetics	41.17647058823529	1001-7
	2421	abrasive cleaner	33.33333333333333	1001-7

Query 6: Case query to check the stock of items

```
select
Case
WHEN balance_on_hand>shelf_capacity*0.50 THEN "Well Stocked"
WHEN balance_on_hand between shelf_capacity*0.40 and shelf_capacity*0.50
THEN "Medium Stock"
ELSE "To be restocked"
END as balance_on_hand, sku_no,sku_desc
from product
group by store_aisle_no;
```

	balance_on_hand	sku_no	sku_desc
▶	Well Stocked	1340	citrus fruit
	Well Stocked	1924	brandy
	Well Stocked	1311	candy
	Well Stocked	2193	beef
	Well Stocked	1780	canned fish
	To be restocked	1273	artif. sweetener

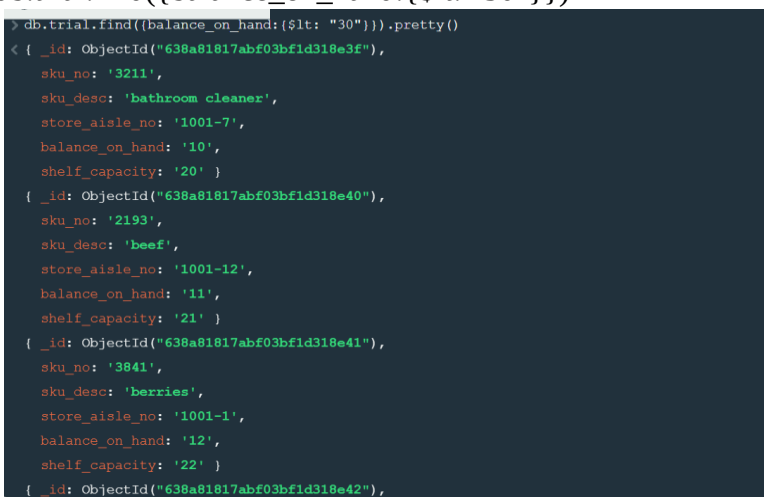
Result 9 ×

NoSQL Implementation:

The following MongoDB queries were executed:

Query 1: Find balance on hand lesser than 30:

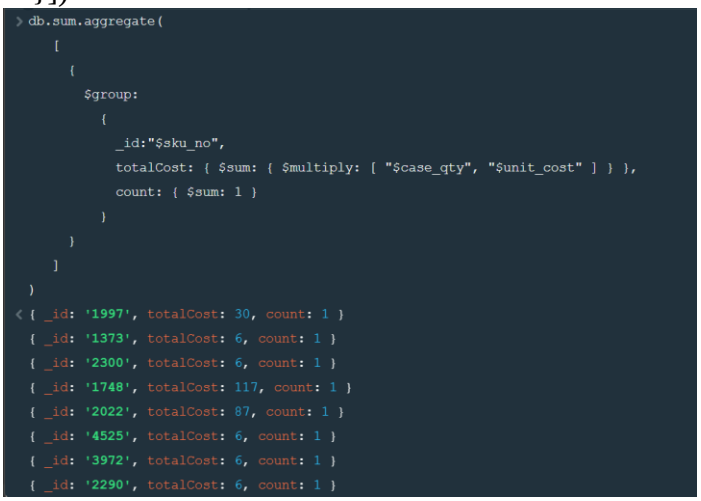
```
db.trial.find({balance_on_hand:{$lt: "30"}})
```



```
> db.trial.find({balance_on_hand:{$lt: "30"}}).pretty()
< { _id: ObjectId("638a81817abf03bf1d318e3f"),
    sku_no: '3211',
    sku_desc: 'bathroom cleaner',
    store_aisle_no: '1001-7',
    balance_on_hand: '10',
    shelf_capacity: '20' }
  { _id: ObjectId("638a81817abf03bf1d318e40"),
    sku_no: '2193',
    sku_desc: 'beef',
    store_aisle_no: '1001-12',
    balance_on_hand: '11',
    shelf_capacity: '21' }
  { _id: ObjectId("638a81817abf03bf1d318e41"),
    sku_no: '3841',
    sku_desc: 'berries',
    store_aisle_no: '1001-1',
    balance_on_hand: '12',
    shelf_capacity: '22' }
  { _id: ObjectId("638a81817abf03bf1d318e42"),
```

Query 2: Find the total cost by multiplying the case quantity and unit cost.

```
db.sum.aggregate([{
    $group:{
      _id: "$sku_no",totalCostt: { $sum: { $multiply: [ "$unit_cost", "$case_qty" ] } },
      count: { $sum: 1 }}
  }])
```



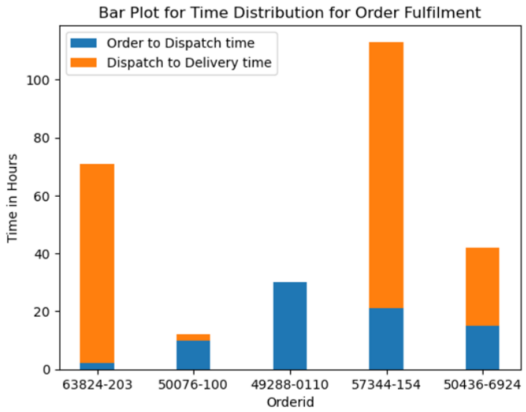
Query 3: Using aggregate function to match role id as Store Associate(i.e., ASS-01)

```
db.trial.aggregate([{$match:{roleid:"ASS-01"}}])
```

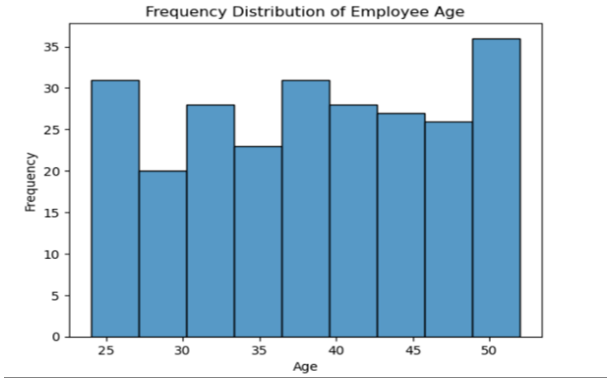
V. Database Access via Python

The database is connected toPython and visualization of the analyzed data is shown below. The connection of MySQL to Python is done using mysql.connector, followed by cursor.excecute to run and fetchall from query, followed by converting the list into a dataframe using pandas library and using matplotlib, seaborn to plot the graphs for the analytics.

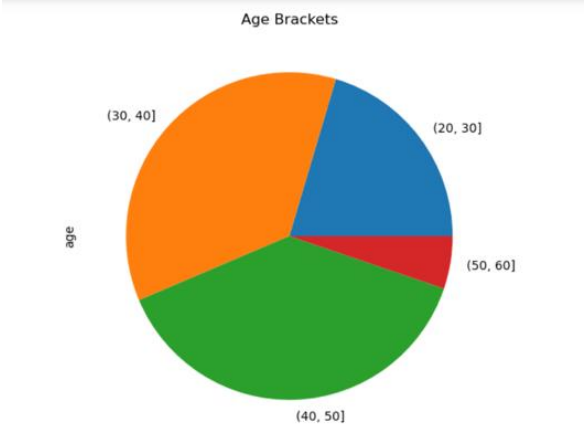
Graph 1: Bar Plot for Time Distribution for Order Fulfilment



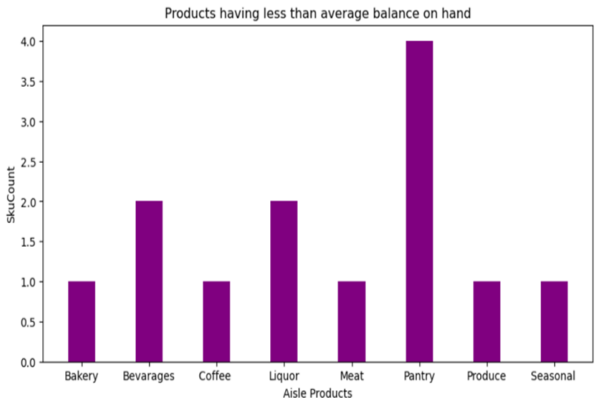
Graph 2: Frequency Distribution of Employee age



Graph 3: Age Buckets of Employees



Graph 4: Products have less than average Balance



VII. Summary and Recommendation

The Grocery Chain Inventory Management System database designed incorporates delivery and truck insights, labor management and in-store product management for small to mid-size grocery store chains. As a result of this model, labor productivity could be improved and overtime expenses can be reduced, as the employees will have more insight into their individual working hours based on requirements and can thus plan schedules accordingly. The stock and reserve inventory view will help the stores prioritize the best-selling items and thus help the retailers stay clearer and more focused on things that can profit them as well as the people.

Improvement on the database would be to include labor standards and stocking time to the data model to measure how we can optimize the shelf restocking. We can also include sales related to data to understand which items are selling out fastest. There is also scope for expansion at the warehouse end, as we are mainly focusing from the perspective of the store. Machine Learning models can be implemented on the data to optimize the restocking time and additionally front-end application with the user id and password can be created for order placement.