**USE CASE STUDY REPORT: EFFECTIVE WAREHOUSE MANAGEMENT**

**Exclusive Summary:**

The primary objective of this study was to design and implement a sophisticated relational database that can be used by the warehouses that supply goods to distribution centers and to stores from there where the end user can buy online or offline. Sorting out all the data that one gets in this industry is humongous for which the old-fashioned invoicing cannot do the job of maintaining all the data under one roof. As this is a big industry that has a lot of scope for expansion, as the industry is this big, profit margins are smaller on single items but on whole it has good margins in profits, because of these low profits on items, the margin of error is very minimal. While looking at this huge industry people are still using old-fashioned invoicing outward and inward data, which is not as efficient as the relational database.

This database can be connected to a Python application platform. Get all the data from the database then work on the analysis part to get the insights from the data also the warehouse management can find out the trend of the consumers at what time of the year. As the preferences of people change concerning the seasons. And maintain the stock according to the trend along with a bit of buffer stock.

The database was modeled taking requirements of data fields that are required for the warehouse to the end user which includes distribution centers, stores, consumers, online orders also about goods. The UML and EER diagrams were designed keeping the business model and structure in mind followed by mapping the conceptual model to a relational model with the required primary and foreign keys. This database was then fully implemented fully MySQL with a well-structured 16 tables and several relationships were implemented between tables after that these tables were read in MongoDB with Studio3T GUI to study all the tables in a NoSQL environment

The created Database is a great hit, and by connecting it to Python, the analytics competencies are enormous, some of which have been shown in this study with the graphical representation of the intuitions. These queries can be very helpful in tracking inward, outward, distribution, percentages of distribution, and a lot more like these.

**Introduction:**

A warehouse is a place where raw materials or manufactured goods may be stored before their export or distribution to centers, from centers, the goods are then distributed to stores where they met the end user either online or offline. Maintaining such huge volumes of data in an old-fashioned way is not practical.

To extract maximum profits from warehouses they should be stocked up with the optimal number of goods, both overstocking and understocking can become a hole to the profits that they are making. Understock can ruin the chances of making profits and overstock can create more chaos if the goods expire, also the investment on overstocking becomes a dead investment and might also become a burden to maintain a lot of goods, especially food related. Keeping the details of food products which has less life span, keeping that in mind all the time and maintaining them requires a lot of ground men on work. Which is an additional burden on the warehouse organizer. To cut out all these extra expenses and make more profits relational database comes into play

The sales of goods like food, clothing, and electronics are very unstable, the inflow and outflow of such huge volumes and finding out the trends through invoicing method is not feasible, the trend of food, clothing, and electronics is very volatile because the preferences of the people change according to seasons which gives a huge advantage to make profits by creating a sophisticated database which can accommodate all the data that is required to find out the trend. To be precise about the trend we got to consider the goods sold in the stores by knowing from where one good is going to where in one system and the magnitude of that good. We can calculate the percentage of the goods that have been supplied to each state and district knowing this information, we can distribute the goods to the distribution center based on this information. Once the database is connected to python, numerous insights can be drawn and changes in the business cycle can be made to get profits.

**II. Conceptual Data Modeling**

*1) Enhanced Entity Relationship***Diagram

Description automatically generated**

*2) Unified Modeling Language*

Diagram, schematic

Description automatically generated

**III. Mapping Conceptual Model to Relational Model**

Primary Key- Underlined Foreign Key- *Italicized*

Warehouse (warehouse\_name, warehouse\_phonenumber, warehouse\_state, warehouse\_city, warehouse\_pincode)

Primary Key: warehouse\_name

Foreign Key: NA

Distribution\_center (center\_name, center\_phonenumber, center\_address, *warehouse\_name*)

Primary Key: center\_name

Foreign Key: warehouse\_name [NOT NULL]

Store (store\_ID, store\_name, store\_location)

Primary Key: store\_ID *SSN, warehouse\_name*, *center\_name*, *store\_ID*

Foreign Key: NA

Online\_order (order\_ID, ordered\_date, delivery\_date, order\_qty, *customer\_ID*)

Primary Key: order\_ID

Foreign Key: customer\_ID [NOT NULL]

Customer (customer\_ID, customer\_name, customer\_number, customer\_address)

Primary Key: customer\_ID

Foreign Key: NA

Goods (product\_code, product\_name, price, stock, manufactured\_date, manufactured\_name)

Primary Key: product\_code

Foreign Key: NA

Food (*product\_code*, weight, expiry\_date, category, nutrition\_facts, ingredients)

Primary Key: product\_code

Foreign Key: product\_code [NOT NULL]

Clothing (*product\_code*, colour, size, fitting, material\_type)

Primary Key: product\_code

Foreign Key: product\_code [NOT NULL]

Electronics (*product\_code*, specifications, model\_no, weight)

Primary Key: product\_code

Foreign Key: product\_code [NOT NULL]

Manager (manager\_Id, SSN)

Primary Key: manager\_Id

Employee (*SSN*, *manager\_id*, emp\_name, emp\_address, emp\_phonenumber)

Primary Key: SSN

Foreign Key: manager\_id [NULL IS ALLOWED]

Employee\_Email (SSN, emp\_email)

Primary Key: SSN, emp\_email

Foreign Key: SSN [NOT NULL]

Works\_At (*SSN, warehouse\_name*, *center\_name*, *store\_ID*)

Primary Key: SSN, warehouse\_name, center\_name, store\_ID

Foreign Key: SSN, warehouse\_name, center\_name, store\_ID

Supplies\_To (*center\_name*, *store\_ID*)

Primary Key: center\_name, store\_ID

Foreign Key: center\_name, store\_ID [NOT NULL]

Sells (*store\_ID*, *product\_code*)

Primary Key: store\_ID, product\_code

Foreign Key: store\_ID, product\_code [NOT NULL]

Payment (payment\_ID, payment\_method, amount)

Primary Key: payment\_ID

Foreign Key: NA

Receives (*order\_ID*, *payment\_ID*, *store\_ID*, amount)

Foreign Key: order\_ID, payment\_ID, store\_ID

**IV. Implementation of Relation Model via MySQL and NoSQL**

**SQL Implementation:** The Database was implemented in MySQL Workbench.

**Creating a schema named ‘Warehouse’ and creating all the tables in Relational Model.**

A screenshot of a computer

Description automatically generated with medium confidence

**The following Queries are executed after creating the database and loading the data.**

use warehouse;

**Query 1: Displaying the required tables**

select \* from Warehouse; Graphical user interface

Description automatically generated

**Query 2: Find the top 5 goods with highest stock available and show their price**

select product\_name, sum(stock) as stock, round(avg(price)

Table

Description automatically generated

from Goods

group by product\_name

order by stock DESC

limit 5;

**Query 3: Total amount of money through credit and debit for online orders**

Graphical user interface, application

Description automatically generatedselect payment\_method, sum(amount)

from Payment

group by payment\_method;

Text

Description automatically generated**Query 4: Find the warehouse name if the distribution center which supplies to more than 20 stores.**

select warehouse\_name from Distribution\_center

where center\_name in

(

select center\_name from Supplies\_To

group by center\_name

having count(\*) > 20

);

Table

Description automatically generated with medium confidence**Query 5: Displaying the products that have max price higher than the products from manufacturer "Mia".**

select product\_name from goods

group by product\_name

having max(price) >=

all(

select price from goods

where manufacturer\_name = "Mia"

);

**Query 6: Find the address of customers who ordered top 5 quantity in online order.**

Table

Description automatically generatedselect c.customer\_name,c.customer\_address, o.order\_qty

from Online\_order as o

INNER JOIN Customer as c

on o.customer\_ID=c.customer\_ID

order by o.order\_qty DESC limit 5;

**Query 7: Select product names and codes from goods of a manufacturer named "Mason" with price <= 1000 using "EXISTS".**

Graphical user interface

Description automatically generated with low confidenceselect product\_code, product\_name

from goods as g

where exists (

select \* from goods g1

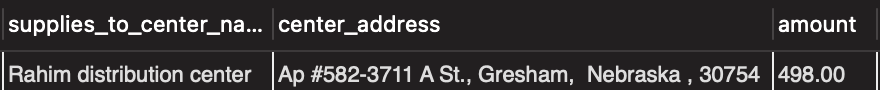
where g.manufacturer\_name = "Mason"

and g.price <= "1000"

) ;

**Query 8: Find the distribution center name, address which supplies to the highest revieved payment happened in a store.**

select s.supplies\_to\_center\_name,d.center\_address, r.amount

from Receives as r

INNER JOIN Supplies\_To as s

on r.receives\_store\_ID=s.supplies\_to\_store\_ID

INNER JOIN Distribution\_center as d

on s.supplies\_to\_center\_name=d.center\_name

order by r.amount desc limit 1;

**Query 9: Find the stock availability of all the cloths with respect to each type.**

Table

Description automatically generatedselect c.material\_type,

SUM(g.stock) as Total\_Stock

from Clothing as c

inner join Goods as g

on c.clothing\_product\_code=g.product\_code

group by c.material\_type;

**Query 10: Find the stock availability of all the food types with respect to each type.**

Table

Description automatically generatedselect f.category as Food\_Category,

SUM(g.price\*g.stock) as Total\_Cost

from Food as f inner join Goods as g

on f.food\_product\_code=g.product\_code

group by f.category;

**NoSQL Implementation:** MongoDB was used as a query language for NoSQL Database. Studio 3T was used as a GUI.

**Created a Database names “warehouse” and created all the collections in MongoDB**

**Graphical user interface, text

Description automatically generated**

**The following Queries are executed after creating the database and loading the data.**

use warehouse;

**Query 1: Displaying the required collections**

db.Customer.find();



**Query 2: Joining Distribution table and warehouse table on warehouse\_name** [Adding 2 screenshots because it’s a wide table].

db.Distribution\_center.aggregate([

{

$lookup: {

from: "Warehouse",

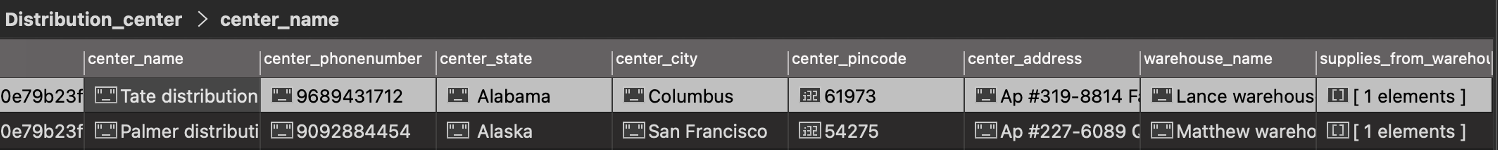
localField: "warehouse\_name",

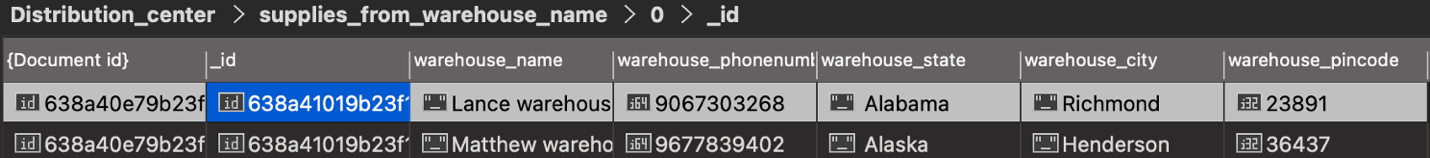
foreignField: "warehouse\_name",

as: "supplies\_from\_warehouse\_name",

},

}

]) 



**V. Database Access via R or Python**

The ‘Warehouse’ data in MySQL is connected in Python using ‘mysql-connector’. For better data analysis all the tables are loaded as python dataframes. Few data visualizations are performed in the application. Such as,

**Graph 1: Percentage contribution of every state in total orders placed.**

Chart, pie chart

Description automatically generated

**Graph 2: Top 10 customer who purchased more products**

**Chart, bar chart

Description automatically generated**

**Graph 3: Food category Vs Stock Availability**

**Chart, bar chart

Description automatically generated**

**VI. Summary and Recommendation**

Effective warehouse management is ready to use a database that can be used in the warehouse management in place of their invoicing methods which gives them an edge over others as they can keep track of where the good is being shipped and where is it reaching the end user that gives a better understanding to the warehouse manager about how much of the goods have to be stored in the warehouse, which helps them to not overbuy or under buy the goods that have to be distributed.

With a few modifications, UI part, and data security, this database will be industry ready database and an application can be created with the right set of UI designs and applying data science techniques we can assess the trends of the goods from manufacturing unit to end customer.