USE CASE STUDY REPORT

Group Number: Group 19

Student Names: Aryaman Kakad and Andrew Le

COMPLAINT MANAGEMENT SYSTEM

Executive Summary

The primary objective of this report is to design, implement, and study a database that we created that is ready for application to be used by small and medium sized businesses that work within highly regulated industries. Doing this requires companies to keep track of their products from sourcing raw materials until it reaches the end user.

This system was modeled from our personal experience working within the medical device industry and includes traceability between complaints and the overall supply chain. We first created an EER and UML diagram to help identify important entities and attributes. These diagrams were mapped into a relational model so that we could better understand our primary and foreign keys. After, we implemented this relational model into a mySQL database and noSQL in order to analyze and study our database.

We were able to successfully run queries which gave various insights, shown in this report. This was done through loading our database into a Python application which helped us visualize information, tracking our complaints from the customer and the non-conformities associated with each supplier.

I. Introduction

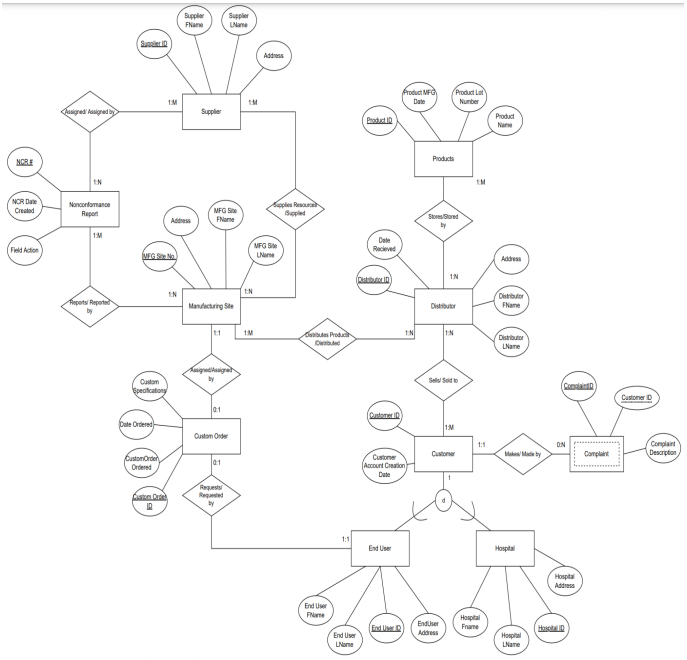
By 2023, companies that are selling in Europe must have a system in place for post-market surveillance, and that includes a system to manage their complaints, as well as track their non-conformities, without using a legacy system. Companies should have been working for the last couple of years to meet domestic requirements and a system like this would help in addressing any potential regulatory issues or product risks that a company may encounter.

Larger companies tend to understand their requirements upon setting up their systems or when they are scaling their companies, hire contractors that are able to set up systems for post-market surveillance and their associated data pipelines. Large companies also tend to have employee's whose job it is to regularly keep track of any changes to regulations in order to stay compliant. Small and medium sized companies however will often continue operation unnoticed, without being compliant. If and when they do get noticed, they may be ordered to completely stop sales and production. After addressing and creating their systems, they must undergo multiple audits to ensure that their systems properly fulfill all theri requirements set by their regulatory bodies.

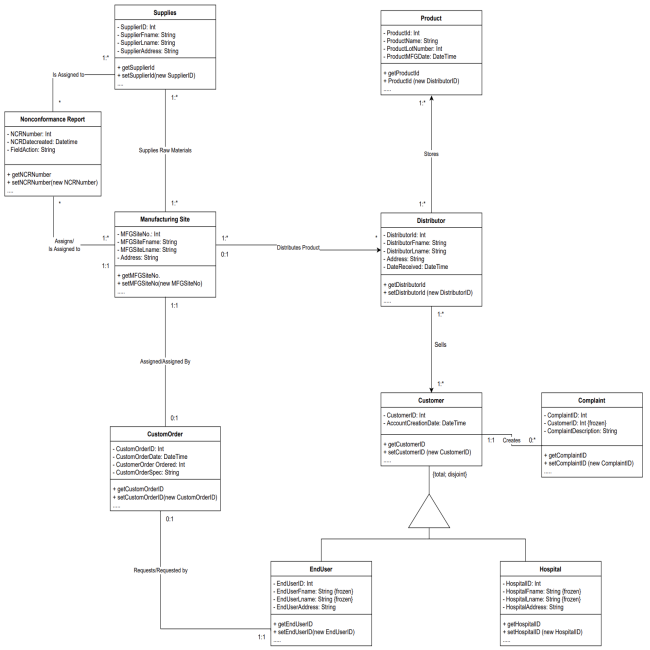
Our system addresses this issue and sets up a foundation for most manufacturing companies to use. It includes aspects of customer support, sales, manufacturing, supply chain, and is useful for overall operations management. We easily identify problematic suppliers or issues with a company's manufacturing process with this traceability system and are able to feed complaints back into assessing risks with our supply chain and manufacturing processs

II. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Primary Key - Underlined Foreign Key - *Italicized*

→ SUPPLIER (SupplierID, SupplierFName, SupplierLname, Supplier_Address)

→ SUPPLIES (*SupplierID, MFGSiteNo)*

- *SupplierID* is a foreign key and it refers to SupplierID in the SUPPLIER relation. It cannot be NULL on update/delete cascade.
- *MFGSiteNo* is a foreign key and it refers to MFGSiteNo in the Manufacturing Site relation. It cannot be NULL on update/delete cascade.

→ ASSIGNED (*SupID*, *NCR#*)

- *SupID* is a foreign key and it refers to SupplierID in the SUPPLIER relation. It cannot be NULL on update/delete cascade.
- *NCR#* is a foreign key and it refers to NCR# in the NON-CONFORMANCE REPORT relation. It cannot be NULL on update/delete cascade.

→ NON-CONFORMANCE REPORT (NCR#, NCR_Date_Created, Field_Action)

→ REPORTS (*MFGSiteNumber, NCRReportNumber)*

- *MFGSiteNumber* is a foreign key and it refers to MFGSiteNo in the Manufacturing Site relation. It cannot be NULL on update/delete cascade.
- *NCR_Report_Number* is a foreign key and it refers to NCR# in the NON-CONFORMANCE REPORT relation. It cannot be NULL on update/delete cascade.

→ MANUFACTURING SITE (MFGSiteNo, MFGSite_Address, MFGSite_FName, MFGSite_LName)

→ DISTRIBUTES (*MFGSite#, DistributorID)*

- *MFGSite#* is a foreign key and it refers to MFGSiteNo in the Manufacturing Site relation. It cannot be NULL on update/delete cascade.
- *DistributorID* is a foreign key and it refers to DistributorID in the DISTRIBUTOR relation. It cannot be NULL on update/delete cascade.

→ DISTRIBUTOR (DistributorID, DistributorFname, DistributorLname, DistributorAddress, DateReceived)

→ STORES (*DistributorIDNumber, ProductID)*

- *DistributorIDNumber* is a foreign key and it refers to DistributorID in the DISTRIBUTOR relation. It cannot be NULL on update/delete cascade.
- *ProductID* is a foreign key and it refers to ProductID in the PRODUCT relation. It cannot be NULL on update/delete cascade.

→ PRODUCTS (ProductID, ProductName, ProductMFGDate, ProductLotNumber)

→ CUSTOMER (CustomerID, CustomerAccountCreationDate)

→ SELLS (*DistributorNumber, CustomerNumber)*

- *DistributorNumber* is a foreign key and it refers to DistributorID in the DISTRIBUTOR relation. It cannot be NULL on update/delete cascade.
- *CustomerID* is a foreign key and it refers to CustomerID in the CUSTOMER relation. It cannot be NULL on update/delete cascade.

→ ENDUSER (EndUserID, EndUserFName, EndUserLName, EndUserAddress, *CustomerIDNumber*)

- *CustomerIDNumber* is a foreign key and it refers to CustomerID in the CUSTOMER relation. It cannot be NULL on update/delete cascade.

→ CUSTOMORDER (CO_ID, CO_Ordered, CO_Specs, CO_DeliveredDate, *EndUserID, MFGSiteIDNumber*)

- *EndUserID* is a foreign key and it refers to the EndUserID in the End-User relation. It cannot be NULL on update/delete cascade.
- *MFGSiteNo* is a foreign key and it refers to MFGSiteNo in the Manufacturing Site relation. It cannot be NULL on update/delete cascade.

→ HOSPITAL (HospitalID, HospitalFname, HospitalLname, HospitalAddress, *CustomerID)*

- *CustomerID* is a foreign key and it refers to CustomerID in the CUSTOMER relation. It cannot be NULL on update/delete cascade.

→ COMPLAINT (ComplaintID, Complaint Description, *Customer#*)

- *Customer#* is a foreign key and refers to CustomerID in the relation CUSTOMER. NULL NOT ALLOWED on Update/Delete cascade.

IV. Implementation of Relational Model via SQL and NoSQL

1. MySQL Implementation

Using NESTED query, Retrieving SupplierID, SupplierFname and SupplierLname where the SupplierID supplier supplies raw products to the Manufacturing Site First Name is Hubert.

```
SELECT SupplierID, SupplierFname, SupplierLname
FROM Supplier
WHERE SupplierID IN (SELECT SupplierID
                     FROM SUPPLIES
                     WHERE MFGSiteNo =(
                                       SELECT MFGSiteNo
                                       FROM `MANUFACTURING SITE`
                                       WHERE MFGSite_Fname = 'Hubert'
                                                                  ))
```

SupplierID	SupplierFname	SupplierLname
6	Davis-Mills	LLC
29	Kassulke-Cartwright	Inc
3	McKenzie-Ritchie	and Sons
12	Gottlieb LLC	Ltd
26	Casper Group	Inc
27	Frami, Sipes and Satterfield	Group
15	Becker Inc	Inc
21	Zieme-Zemlak	PLC
4	Lind, Tremblay and Ziemann	Ltd
NULL	NULL	NULL

Using Correlated Query, Retrieving the Custom Order IDs who had more products ordered in their Custom Order than the average number of products in the custom order manufactured by their respective manufacturing site.

```
SELECT CO_ID, EndUserID, MFGSiteIDNumber, CO_DeliveredDate
FROM CUSTOMORDER c1
WHERE CO_Ordered > (SELECT AVG(CO_Ordered)
                    FROM CUSTOMORDER c2
                    WHERE c2.MFGSiteIDNumber = c1.MFGSiteIDNumber);
```

CO_ID	EndUserID	MFGSiteIDNumber	CO_DeliveredDate
1	1996	103	1986-07-25 00:00:00
3	1270	101	1996-10-05 00:00:00
4	1562	101	2016-09-08 00:00:00
6	1167	101	2007-09-05 00:00:00
7	1869	102	1998-10-17 00:00:00
10	1338	101	1979-09-29 00:00:00

Using CORRELATED query, retrieving Supplier ID who have more than one Non-Conformance Reports assigned to them.

```
SELECT DISTINCT a1.SupID, a1.`NCR#`
FROM assigned a1
WHERE (     SELECT COUNT(*)
            FROM assigned a2
            WHERE a1.SupID = a2.SupID
            AND a1.`NCR#` = a2.`NCR#`) > 1;
```

SupID	NCR#
7	12800
4	11086
26	14672

Retrieves the Names of the Customers who do not have the least number of custom orders from the Manufacturing Site 101 and 102 and the Custom Order Delivery Date is after 1995-01-01.

```
SELECT EndUserID, EndUserFname, EndUserLname
FROM ENDUSER
WHERE EndUserID IN (
                    SELECT EndUserID
                    FROM CUSTOMORDER
                    WHERE CO_DeliveredDate > '1995-01-01'
                    AND CO_ORDERED > ANY
                                        (SELECT CO_ORDERED
                                        FROM CUSTOMORDER
                                        WHERE MFGSiteIDNumber != '103'))
ORDER BY ENDUserFname ASC;
```

EndUserID	EndUserFname	EndUserLname
1270	Baron	Breitenberg
1046	Flossie	McClure
1869	Mekhi	Larkin
1167	Rylee	Wehner
1562	Vance	Ward
NULL	NULL	NULL

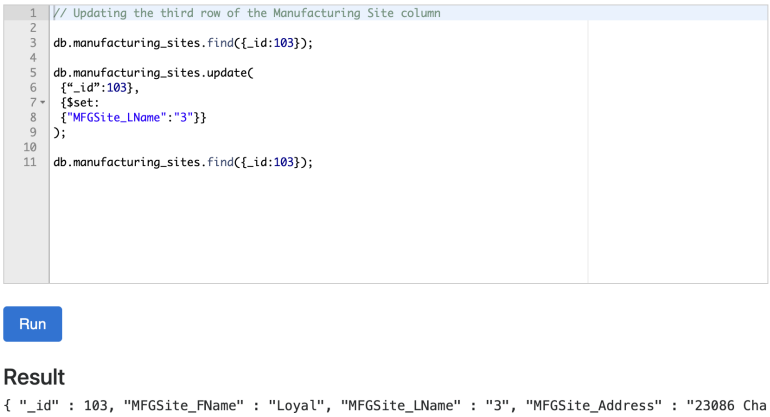
2. NoSQL Implementation

In our project, we have 16 tables out of which we have created and implemented which are as follows.

a. Manufacturing Site
b. Supplier
c. Supplies
d. Non-Conformance Report
e. Assigned
f. Report
g. Distributor
h. Stores
i. Distributes
j. Product

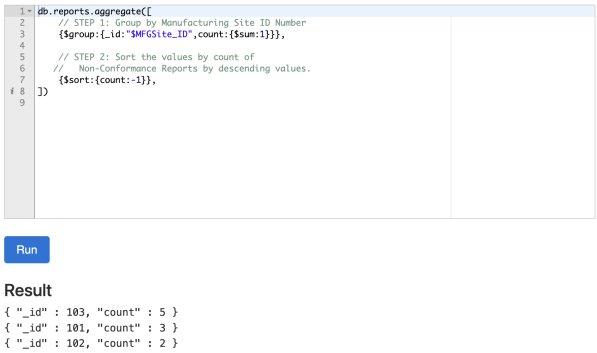
Updating the third row of the Manufacturing Site column

```
db.manufacturing_sites.find({_id:103});

db.manufacturing_sites.update(
 {“_id”:103},
 {$set:
 {"MFGSite_LName":"3"}}
);

db.manufacturing_sites.find({_id:103});
```



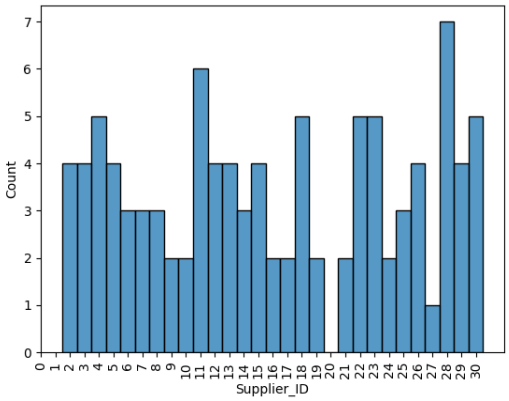
// Count of Non-Conformance Reports per Manufacturing Site.

```
db.reports.aggregate([
   // STEP 1: Group by Manufacturing Site ID Number
   {$group:{_id:"$MFGSite_ID",count:{$sum:1}}},

   // STEP 2: Sort the values by count of
  //   Non-Conformance Reports by descending values.
   {$sort:{count:-1}},
])
```

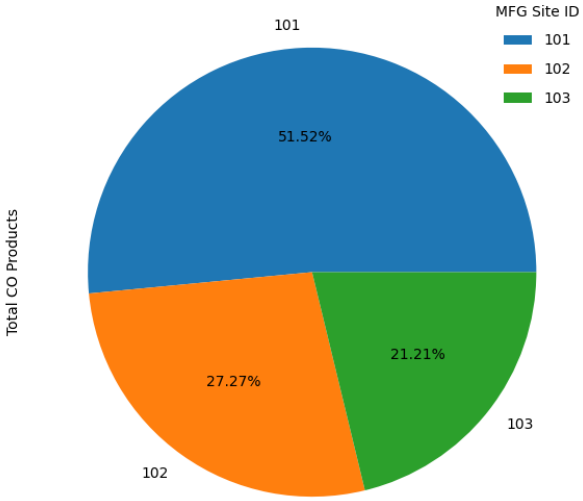


V. Database Access via Python

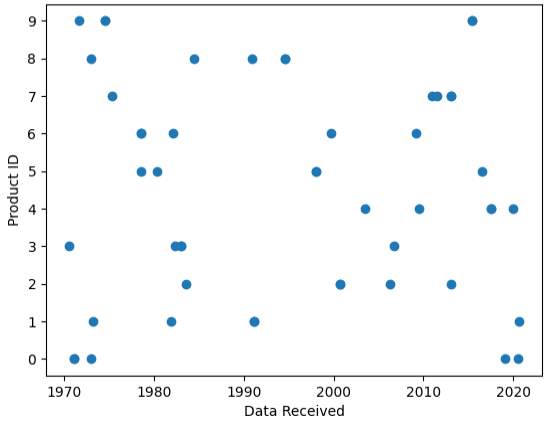
The database is accessed using Python and visualization of analyzed data is shown below. The connection of MySQL to Python is done using mysql.connector, followed by cursor.execute to run and fetchall from query, followed by converting the list into a dataframe using pandas library and using matplotlib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



Graph 1. Non-conformances per Supplier



Graph 3. Customer Orders per MFG Site



Graph 2. Product Distribution vs Date Received

V. Summary and Recommendations

Our Complaints Management Systems database, designed on mySQL, is intended for smaller and medium sized companies that require a foundation for creating pipelines regarding their overall Enterprise Resource Management with a focus on complaints information that they receive from their customers, as well as a focus on tracking non-conformities that occur and tracing it back to their root cause. This will result in that company being able to eventually customize this system and achieve conformance to the appropriate regulatory requirements.

Improvements on this system would come from adding more information regarding the device and design history files, including all the root cause analysis information and risk assessments.

Potential shortcomings are mainly in our noSQL, where we do not have all of out tables and relationships implemented. This would not give an accurate scope of our model. If we were to add more tables and queries to the noSQL database, we could likely build the entire database there.