**SYSTEM REQUIREMENTS SPECIFICATION DOCUMENT**

Document Number: BBT 3104-SyRS-2020/04

Version: 3.7

Review Date:

Submission Date: 7/14/2020

Case Study: Wanji’s Food Industries

Group Number: 36

Team Members:

|  |  |  |
| --- | --- | --- |
|  | **Student Details** | **Mark Awarded** |
|  | Lilian Thairu – 111878. – Group A – SU Email: Lilian.Thairu@strathmore.edu |  |
|  | Lydia Kahuria – 110875. – Group A– SU Email:Lydia.Kahuria@strathmore.edu |  |
|  | Fredah Kioko– 103428. – Group – SU Email:Fredah.Kioko@strathmore.edu |  |

|  |  |  |
| --- | --- | --- |
| **Version Change** | **Date of Change** | **Changes Made** |
| Version 1.0 to 1.2 | 25th April 2020 | Added the business process and objectives |
| Version 1.2 to 1.6 | 9th May 2020 | Added the system capabilities, Added the system constraints, Added the conditions |
| Version 1.6 to 2.1 | 22nd May 2020 | Added ERD and the Uml, database Schema and the system Architecture |
| Version 2.1 to 2.7 | 19th June 2020 | Added the Dependencies |
| Version 2.7 to 2.9 | 26th June 2020 | Added 1st Normal form and he 2nd Normal form |
| Version 2.9 to 3.1 | 3rd July 2020 | Added 3rd ,BOYCE and 5th Normal form |
| Version 3.1 to 3.6 | 10th July 2020 | Added The search engine , the procedures,functions,views |
| Version 3.6 to 3.7 | 16th July 2020 | Added all the DDL statements,updated the list of figures,the list of tables and wrote the abstract. |

# 

# 

# Executive Summary

***<This will be filled in at the end. It is like an abstract>***

Wanji’s food production is a new food processing industry that provides its customers with healthy and nutritious foods and affordable prices.This is done by mixing a variety of cereals and healthy foods and coming up with snacks and flour that fulfils all the dietary needs to a targeted group of people with special needs for example the sick,lactating mothers,children and everyone seeking to lead a healthy lifestyle.Most of the production processes are automated using state of the art machines.The goal is to develop a computerised way that will enable the organisation keep,record and retrieve records that can be further analysed to generate reports that will help the organisation in making decisions.

By analysing the business process a suitable database will be generated to improve the strategies and operations of this industry . It will also be structured in such a way that it is effective and easy to understand , flexible enough to capture and store all the necessary data of the organisation.

**Table of Contents**

[Executive Summary 1](#_gjdgxs)

[List of Figures 3](#_3znysh7)

[List of Tables 4](#_2et92p0)

[List of Models 5](#_2zbgiuw)

[Chapter 1. Introduction 6](#_tyjcwt)

[1.1 The Mission of the Business 6](#_3dy6vkm)

[1.2 The Day to Day Operations of the Business 6](#_1t3h5sf)

[1.3 Statement of Objectives: Aim of Designing and Implementing the Database 6](#_26in1rg)

[Chapter 2. System Capabilities 7](#_lnxbz9)

[A. Example1 Data (E.g. Patient Data or Client Data or Bicycle Data) 7](#_35nkun2)

[B. Example2 Data 7](#_1ksv4uv)

[C. Example3 Data 7](#_44sinio)

[Chapter 3. System Conditions 8](#_1y810tw)

[A. Example1 Data (E.g. Patient Data or Client Data or Bicycle Data) 8](#_4i7ojhp)

[B. Example2 Data 8](#_1ci93xb)

[C. Example3 Data 8](#_3whwml4)

[Chapter 4. System Constraints 9](#_1pxezwc)

[A. Example1 Data (E.g. Patient Data or Client Data or Bicycle Data) 9](#_49x2ik5)

[B. Example2 Data 9](#_2p2csry)

[C. Example3 Data 9](#_147n2zr)

[Chapter 5. Assumptions 10](#_1hmsyys)

[Chapter 6. Database Design 11](#_41mghml)

[a. Entity Relationship Diagram Based on the Chen Notation 11](#_1egqt2p)

[b. Database Schema Based on the UML Notation 12](#_vx1227)

[Chapter 7. Normalization 13](#_4f1mdlm)

[a. List of Functional Dependencies 13](#_2u6wntf)

[b. Normal Forms 13](#_19c6y18)

[Chapter 8.](#_111kx3o)

1. Storage Engines
2. Indexes
3. Normal Triggers
4. Temporal Triggers
5. Procedures
6. Functions
7. Views

Chapter 9.DDL Statements

# List of Figures

[Figure 1.1 FLOWCHART 7](#_4d34og8)

[Figure 1.2 BUSINESS PROCESS 9](#_2s8eyo1)

[Figure 1.3 CONTEXT DIAGRAM(LEVEL 0) 9](#_17dp8vu)

[Figure 1.4 LEVEL DIAGRAM 11](#_3rdcrjn)

[Figure 6.1 ERD USING CHEN NOTATION 26](#_2grqrue)

[Figure 6.2 UML DIAGRAM 27](#_3fwokq0)

[Figure 6.3 SYSTEM ARCHITECTURE 28](#_1v1yuxt)

# List of Tables

[Table 7‑1Employee Details Table 43](#_Toc46087602)

[Table 7‑2 Salary Details Table 43](#_Toc46087603)

[Table 7‑3 Supplier details table 44](#_Toc46087604)

[Table 7‑4 Raw Materials order details 45](#_Toc46087605)

[Table 7‑5 Supplier Payment Details 45](#_Toc46087606)

[Table 7‑6 Raw material Invoice table 45](#_Toc46087607)

[Table 7‑7 Machine Table 46](#_Toc46087608)

[Table 7‑8 Product details 46](#_Toc46087609)

[Table 7‑9 Inventory Details 47](#_Toc46087610)

[Table 7‑10 Test Market Details 47](#_Toc46087611)

[Table 7‑11 Retailer Details Table 47](#_Toc46087612)

[Table 7‑12 Retail Order Table 47](#_Toc46087613)

[Table 7‑13 Retail Order Invoice Table 48](#_Toc46087614)

[Table 7‑14 Retailer Payment Table 49](#_Toc46087615)

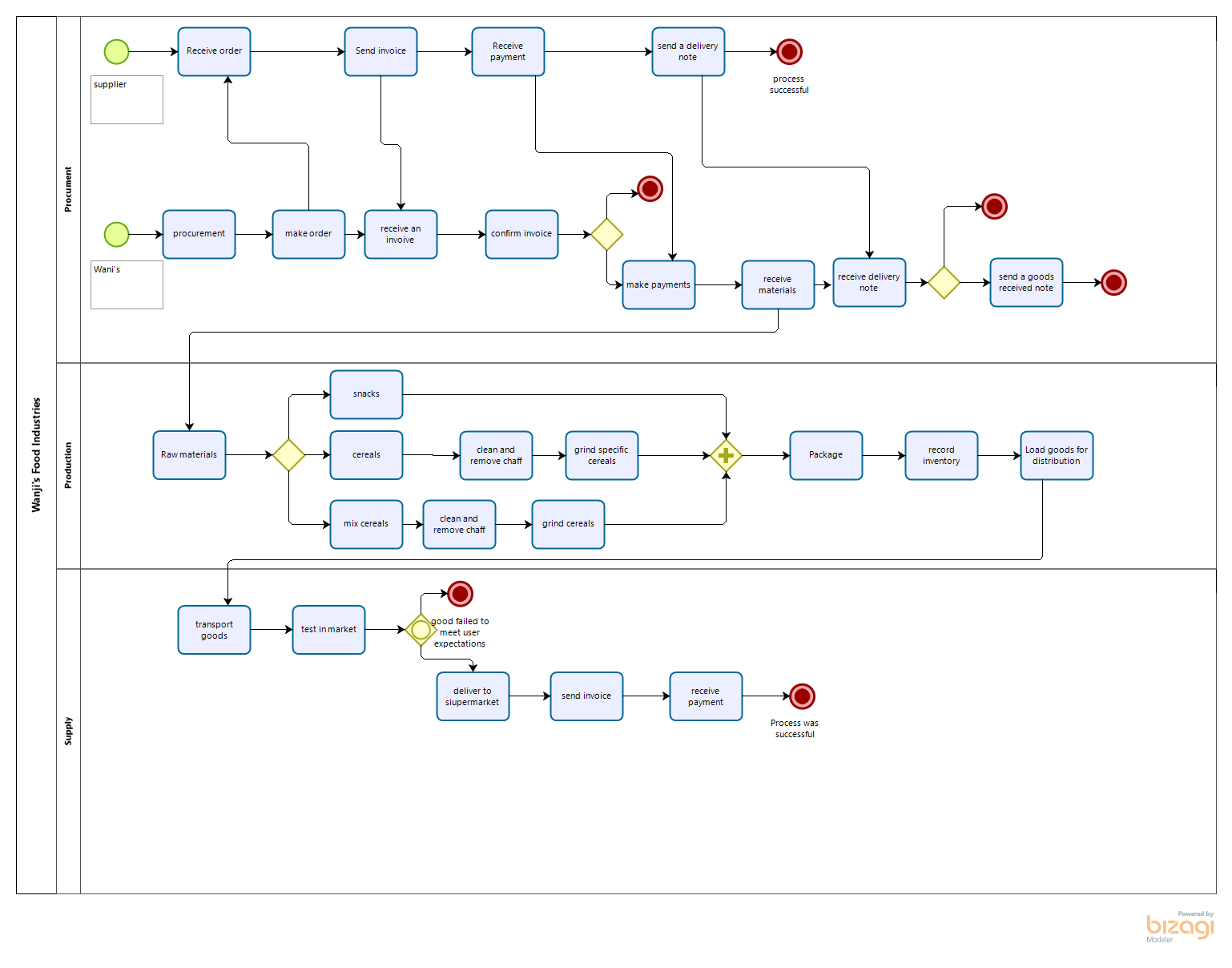
# Introduction

## The Mission of the Business

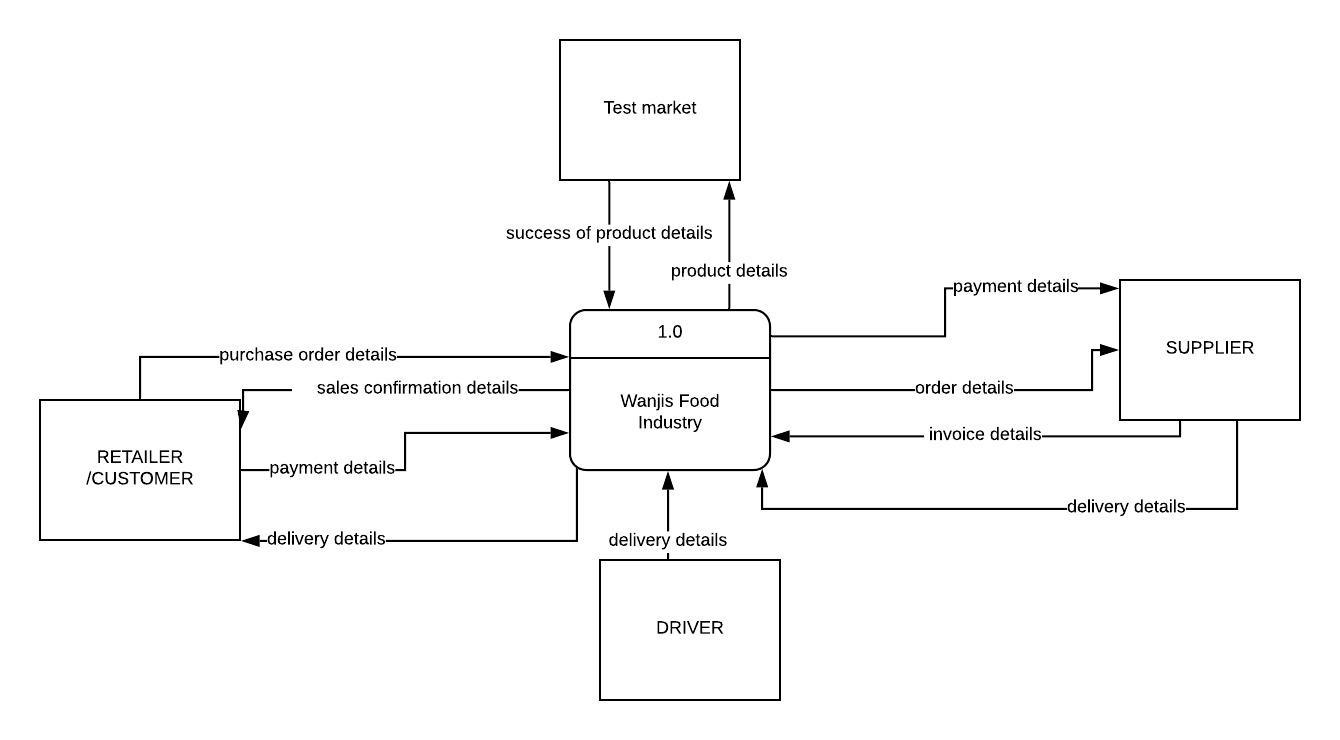
The mission of Wanji’s Food Industries is to process healthy and nutritious food products targeted to a specific group of people with specific needs hence solving the societal problem of malnutrition by making them affordable to the consumers.

## The Day to Day Operations of the Business*<Create a flowchart here that represents the business processes involved in producing the business' main product/service>*

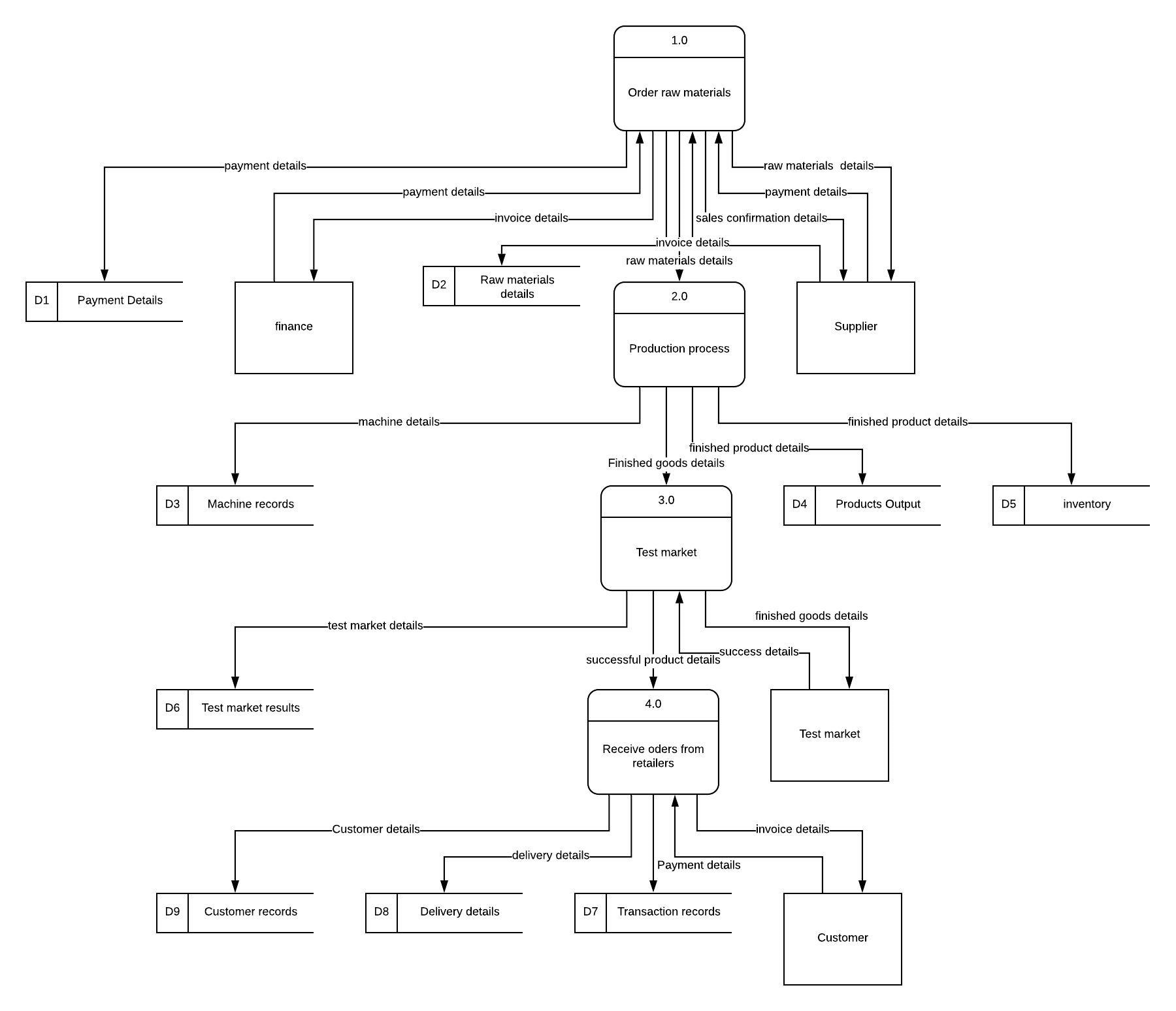
*Figure 1.1 FLOWCHART*



*Figure 1.2 BUSINESS PROCESS*

**<Create a Data Flow Diagram (DFD) that represents the flow of information through the business organization in the case study>****

*Figure 1.3 CONTEXT DIAGRAM(LEVEL 0)*



*Figure 1.4 LEVEL DIAGRAM*

## Statement of Objectives

***<Hint: Think in terms of an IT-based solution to address the negative effects associated with the non-value adding activities in the organization’s business processes>***

1. To record the inventory of the organization in a database in order to generate reports that may be used to make decisions in the future.
2. To store customer orders in order to know the demand of the products and the success rate of each product in a database management system
3. To analyze the stored Purchase details of raw materials against the output of products processes by the industry using business intelligence dashboard
4. To manage staff details and know the activeness of each member and any bonuses if any by creating relationships and automating the calculations of salaries in the database
5. To compare sales of different products to know the performance of the food processing industry using a graph and displayed in the business intelligence dashboard
6. To track payments made to the supplier and those received from retailers
7. To track the quality of raw materials given by the supplier by recording the supplier details.

# System Capabilities

## Employee Details Data

***<Guiding example: The database should be capable of storing data about the clients and for each client, the following data should be stored:>***

|  |  |
| --- | --- |
| **1.** | Employee\_ ID |
| **2.** | Employee\_ First name |
| **3.** | Employee \_Last Name |
| **4.** | Employee\_Contacts |
| **5.** | Employee\_ Department |
| **6.** | Employee\_Salary |
| **7.** | Employee\_Bonus/Allowances |
| **8.** | Status |

## Raw Materials Data

## *<Guiding example: The database should be capable of storing data about the rental properties and for each rental property, the following data should be stored:>*

|  |  |
| --- | --- |
| **1.** | Raw Material ID |
| **2.** | Raw Material Name |
| **3.** | Product Output |

## Supplier Payment Details(SP)

|  |  |
| --- | --- |
| **1.** | SP Transaction Number |
| **2.** | SP Method of payment |
| **3.** | SP Invoice received |
| **4.** | SP Name of supplier |
| **5.** | SP Amount paid |
| **6.** | SP Day of payment |

## D. Retailer Payment Details Data

|  |  |
| --- | --- |
| **1.** | Retailer Transaction Number |
| **2.** | Retailer Method of payment |
| **3.** | Retailer Name of supplier |
| **4.** | SP Retailer Amount paid |
| **5.** | Retailer Day of payment |

## E. Supplier Details Data

|  |
| --- |
| 1. Supplier Id |
| 2. Supplier Company Name |
| 1. Supplier Invoice |

## Sales Details Data

|  |  |
| --- | --- |
| **1.** | Sales Product ID |
| **2.** | Sales Product Name |
| **3.** | Number of products sold |
| **4.** | Date of Sales |

## Product Details Data

|  |  |
| --- | --- |
| **1.** | Product ID |
| **2.** | Product Name |
| **3.** | Manufacture\_Date |
| **4.** | Raw\_Materials ID |

## Machine Details Data

|  |  |
| --- | --- |
| **1.** | Machine Number |
| **2.** | Machine Type |
| **3.** | Machine Active Years |

## 4. Machine Output

## I. Order Details Data

|  |  |
| --- | --- |
| **1.** | Order Id |
| **2.** | Order details |
| **3.** | Order Invoice |
| **4.** | Delivery Note |
| **5.** | Date of order |

## I. Test Market Details Data

|  |  |
| --- | --- |
| **1.** | Test Market ID |
| **2.** | Test Market location |
| **3.** | Test Retailer Name |
| **4.** | Test product Name |
|  | Test Duration |
| **6.** | Test Success status |

# System Conditions

## Employee Details Data *<Client Data>*

***<Guiding example: The database should contain the following measurable characteristics for each capability specified in Chapter 2.>***

|  |  |
| --- | --- |
| **1.** | The employee Identification is their National ID. Each National ID is unique to a person. |
| **2.** | The employees should fill in their first name which is either one or two. |
| **3.** | The employees should fill out their last name as well which is either one or two as well. |
| **4.** | Telephone numbers will be required so that the employee can be contacted easily in case of an unexpected outcome. So employees may have more than one phone number but only two most reachable contacts will be necessary. |
| **5.** | Every employee should fill in the department they are featured in, that is, where they mainly do their work for example the production department as well as the accounts department. One employee belongs to one department. |
| **6.** | The amount of salary received is also information required in the database. The salary can be written in thousands or even millions, but it can only be once. |
| **7.** | When employees receive bonuses, in essence, an extra amount of cash from the normal salary, it needs to be recorded in the database. This can be written in thousands only, and can only be filled in once. |
| **8.** | Status implies to the position one holds in the company hence, this is information crucial to the company that must be recorded in the database. One employee holds one position. |

## Raw Material Data

|  |  |
| --- | --- |
| **1.** | The product ID is the number that is assigned to the product (Serial number) as received from the supplier.Here it will act as the raw materials needed in production This number is unique to each product. This is also the number used in scanning for sales. |
| **2.** | The raw materials can have products of one or more names. |
| **3.** | This is the number of hours and the amount of that were used in the production of the product, it can be measured using the number of hours. |

## Supplier Payment Details Data

|  |  |
| --- | --- |
| **1.** | The transaction number is unique to a specific supplier. (This is the foreign key for the supplier table) |
| **2.** | Many suppliers can be paid using the same method of payment. |
| **3.** | A supplier delivers their own invoice hence 1:1 |
| **4.** | Two or more suppliers may share the same name hence, m:1 |
| **5.** | One payment is specific to the supplier who is being paid. |
| **6.** | Many suppliers can pay on the same day, hence m:1. |

# Retailer Payment Details Data

|  |  |
| --- | --- |
| **1.** | The transaction number is unique to a specific retailer. |
| **2.** | Many retailers can pay using the same method of payment. |
| **3.** | A retailer is given their own invoice hence 1:1 |
| **4.** | Two or more retailers may share the same name hence, m:1 |
| **5.** | Many retailers can pay on the same day, hence m:1 |

# Supplier details Data

|  |  |
| --- | --- |
| **1.** | The Company’s ID is unique to each supplier, hence which includes the Companies identification number, 1:1 |
| **2.** | Order details may share a common list of requirements, m:1 |
| **3.** | Order Invoice Each supplier delivers their own unique type of invoice, 1:1 |
| **4.** | Order Date Wanji’s Food industries may opt to send orders on the same day |
| **5.** | Many goods may come from the same supplier m:1 |

# Sales Details Data

|  |  |
| --- | --- |
| **1.** | The item ID is unique to every item. |
| **2.** | An item may have a common name, hence, 1:1 |
| **3.** | The number of an item that has been sold may be the same as another item, hence m:1 |

# Inventory Details Data

|  |  |
| --- | --- |
| **1.** | The item number is specific to an item, 1:1 |
| **2.** | Many goods from different suppliers may be received on the same day, hence, m:1 |
| **3.** | The quantity of the number of orders sent may be the same for different products hence, m:1 |

# Machine Details Data

|  |  |
| --- | --- |
| **1.** | The machine number is specific to a machine,in this case it is the build number of the machine 1:1 |
| **2.** | Each machine performs a different functionality, hence, 1:1 |
| **3.** | The years active for machines may be the same hence, m:1 |

# 4. The machine output is the amount of products the machine can process in this the output

# depends on the amount of raw materials used and the amount of minimise wastages hence m:1

## I Order Details Data

|  |  |
| --- | --- |
| **1.** | Order ID is the number provided by the procurement document number It should be unique to every order |
| **2.** | Order Details should be the list of raw materials to the supplier |
| **3.** | Order Invoice this is the invoice received from the supplier |
| **4.** | Delivery Note this is the document with the list of goods delivered to the organization from the supplier |
| **5.** | Date of order this is the date that the raw materials were orderd from the supplier |

## I. Test Market Details Data

|  |  |
| --- | --- |
| **1.** | Test Market ID is the unique Identifier for the testing process.It will be named based on the sample and the number of times it has been tested |
| **2.** | Test Market location this is the geographical area in which the retailers used to test the product are based on . |
| **3.** | Test Retailer Name this is the name of the retailer in which the experiment is taking place |
| **4.** | Test product Name this is the product produced by wanji that is examined before its official release to the broader market |
| **5.** | Test Duration this the period in wich the duration takes place it shall consist of the first date and the last date of the test |
| **6.** | Test Success status this is the result of the experiments and the few notes attached to it. |

# System Constraints

***<E.g. interrelational, intarelational, static, dynamic, semantic, primary key, & foreign key>***

*E.g. The database must satisfy the following constraints:*

## Employee Details Data

|  |  |
| --- | --- |
| **1.** | The Employee Identification number, first name and Last name shall have a key constraint. It will be intrarelational, composite, static key constraint to form the primary key. The Employee ID domain constraint is VARCHAR |
| **2.** | First name shall have a key constraint,that is dynamic and is intrarelational with the last name and Employee ID. The domain constraint is STRING |
| **3.** | The last name shall have a dynamic key constraint as well and is intrarelational with the first name and Employee ID. The domain constraint is STRING |
| **4.** | Contacts shall have a static constraint that is intrarelational with the First Name and Last Name. The domain constraint is BOOLEAN |
| **5.** | Department is intrarelational with the salary and bonuses but is interrelational with the status. The domain constraint is STRING |
| **6.** | Salary is intrarelational with the Department and the status but is interrelational with the bonus. It is also a semantic key constraint as the employee cannot earn more than the employer. The domain constraint is BOOLEAN |
| **7.** | The Bonus is semantic,static,intrarelational with the Department and status but interrelational with the Salary. The Domain constraint is BOOLEAN |
| **8.** | The Status is intrarelational with the salary and the bonus but interelational with the Department. The domain constraint is STRING |

## Raw Materials Data

|  |  |
| --- | --- |
| **1.** | The Raw material ID shall have a static,interrelational,composite constraint to the Product name and intrarelational to the output of the product. It is the primary key for the products data and also the Foreign key for the Sales Details Data and the Inventory Details Data. The domain constraint is BOOLEAN |
| **2.** | The Raw Material Name is dynamic,intrarelational to the output of the product and interrelational to the Product ID. The domain name is STRING |
| **3.** | The output of the product shall have a dynamic constraint that is Intrarelational to the Product ID and Product Name and is interrelational to the number of products sold. The domain constraint is BOOLEAN |

## Supplier Payment Details Data

|  |  |
| --- | --- |
| **1.** | The transaction number is interrelational,composite to the Company ID of the supplier, the Name of the supplier and the method of payment used. The transaction Number forms the Primary key for the supplier Payment Details Data. The Domain constraint is BOOLEAN |
| **2.** | The method of payment shall have a static constraint that is interelational to the transaction number, the Company ID and the Company Name. The domain constraint is BOOLEAN |
| **3.** | The invoice received is intrarelational to the amount paid and the day of payment but it is interrelational to the name of the supplier. The domain constraint is BOOLEAN |
| **4.** | The name of the supplier is intrarelational to the amount paid, the invoice received and Day of Payment. The domain constraint is STRING |
| **5.** | The amount paid is intrarelational to the name of supplier but is interrelational to the transaction number and the invoice received. The domain constraint is BOOLEAN |
| **6.** | The Day of payment is interrelational to the amount paid and the invoice received. The domain constraint is DATE |

# Retailer Details Data

|  |  |
| --- | --- |
| **1.** | The transaction number will have a composite, interrelational to the Name of the retailer and the method of payment. The Domain constraint is BOOLEAN |
| **2** | The |
| **2.** | The method of payment is interrelational to the transaction number and the Retailers’ Name. The domain constraint is BOOLEAN |
| **3.** | The name of the retailer is intrarelational to the amount paid and Day of Payment. The domain constraint is STRING |
| **4.** | The amount paid is intrarelational to the name of the retailer but is interrelational to the transaction number. The domain constraint is BOOLEAN |
| **5.** | The Day of payment is interrelational to the amount paid and the Delivery note. The domain constraint is DATE. |

# Supplier Details Data

|  |  |
| --- | --- |
| **1.** | The Supplier ID is interrelational to the Transaction number for the payment as well as the Company Name. The Company ID forms the primary key for the supplier Details Data and is the Foreign key for the supplier Details Data. The domain constraint is BOOLEAN |
| **2.** | The Order Details is interrelational to the Company ID. The domain constraint is STRING |
| **3.** | The Invoice is intrarelational to the Company ID and Company Name. The Domain constraint is VARCHAR |
| **4.** | The supplier\_Company Name is the attribute key to this particular table. The Domain Constraint is STring |
| **5.** | The Date of order is intrarelational to the Company ID and the Company Name. The domain constraint is DATE |
| **6.** | The delivery note is interelational to the company ID and the raw materials. The domain constraint is VARCHAR |

## Product Details Data

|  |  |
| --- | --- |
| **1.** | The Product ID will have a composite semantic, interrelational to the Product name and Product ID while intrarelational to the output of the product and the Number of products sold. The domain constraint is BOOLEAN |
| **2.** | The Product Name received is intrarelational to the Deliver Note, Invoice Received, Product ID, Product Name and Amount Paid. The domain name is DATE |
| **3.** | The manufacture Date is Intrarelational to the Product ID, Product Name and Amount Paid. The domain constraint is BOOLEAN |

## 

## Machine Details Data

|  |  |
| --- | --- |
| **1.** | The Machine Number is intrarrelational to the build number when the machine was purchased. The domain constraint is VARCHAR |
| **2.** | The Machine Type is intrarelational to the Machine Number The domain name is STRING |
| **3.** | The Years active have a domain constraint that is BOOLEAN |

4. The Machine output shall have an intrarelation constraint.

# Order Details Data

|  |  |
| --- | --- |
| **1.** | The Order ID will have a compositeDomain constraint is BOOLEAN |
| **2.** | The order details are interrelational to the transaction number and the Retailers’ Name. The domain constraint is BOOLEAN |
| **3.** | The Order invoice is intrarelational to the amount paid and Day of Payment. The domain constraint is STRING |
| **4.** | The delivery note is intrarelational to the name of the retailer but is interrelational to the transaction number. The domain constraint is BOOLEAN |
| **5.** | The Day of order Delivery note. The domain constraint is DATE. |

## I. Test Market Details Data

|  |  |
| --- | --- |
| **1.** | Test Market ID is unique to the relation VARCHAR |
| **2.** | Test Market location is the location in which the retail is located and the area surrounding it STRING |
| **3.** | Test Retailer Name is the retail shop in which the product was being tested VARCHAR |
| **4.** | Test product Name the name of the product that was being tested in the market VARCHAR |
|  | Test Duration is the date to start and end the market experiment current TIMESTAMP |
| **6.** | Test Success status is weather or not the product is successful . VARCHAR |

# Assumptions

It is assumed that wanjis food industries has a stable relationship with all its suppliers

It is assumed that all The payment transactions occur via the bank or mpesa.

It is assumed that all the food processing machines are of the same brand

It is also assumed that the suppliers are paid using the same method.

It is assumed that all orders are delivered on time with minimal or no delays.

It is assumed that the machines used by Wanjis Food industry are of good quality and in good working condition.

It is assumed that employees may receive bonuses or allowances.

It is assumed that all the temporary employees come back when needed and do not change.

It is assumed that each employee working in the manufacturing stage has a specific experty hence only operates a specific machine in which they are qualified.

It is assumed that all Suppliers have company ID’S that are unique to them

It is assumed that new products are tested for six months in the market.

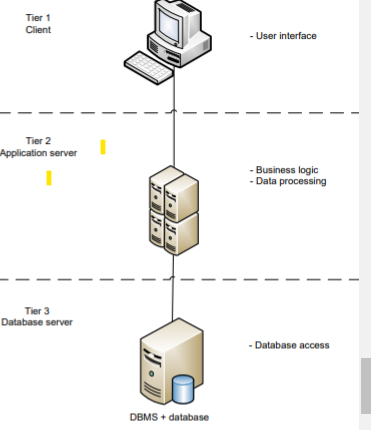
# DatabaseDesign EntityRelationshipDiagramBasedontheChenNotation

*Figure 6.1 ERD USING CHEN NOTATION*

## Database Schema Based on the UML Notation

*Figure 6.2 UML DIAGRAM*

## System Architecture



*Figure 6.3 SYSTEM ARCHITECTURE*

These are the advantages of a 3 tier system architecture

1) Less expensive hardware for the client in tier 1

2) Eliminates the concerns of software distribution because the core of the application is installed on the centralized application server

3) Modularity enables one tier to be replaced/modified without affecting the other tiers

4) Load balancing is easier with the separation of the core business logic (coded as algorithms in a software: the algorithms process the data based on predefined business logic documented in SOPs, e.g. an algorithm to calculate the Net Present Value) from the database functions (storage and retrieval of data) I saw it best to use a three-tier architecture because of the advantages that comes with it. First, this type of architecture is very scalable whereby the application servers can be used in many machines and the database does not require direct connection to every client. This allows for business growth over time. The company also needs computers that are going to act as clients, each computer costs about $800 with good specifications and capability to be able to handle all the system - User interface - Main business logic and data processing logic Tier 1 C

# Normalization

## List of Functional Dependencies

***E.g. Full, Partial, Transitive, Multi-Valued Dependencies (MVD), Join Dependencies (JD), etc.***

Full Function Dependencies

**Product**

The *product \_number*(INT) is fully dependent on the *product \_Id* (INT)and the *product \_type(VARCHAR)*.

**Supplier**

The *supplier \_delivery \_note(VARCHAR)* is fully dependent on the *supplier\_ Id* (INT)and the *Supplier \_Company \_name*.(VARCHAR)

**Order**

The *delivery \_note(VARCHAR)* is fully dependent on the key constraints that is The *order ID(INT)*, the *order detail(VARCHAR)s*, the *invoice of the order(VARCHAR)* and the *date of the order(DATE TIMEDTAMP).*

The *Order \_invoice(VARCHAR)* is fully dependent on the key constraints that is *The order \_ID(INT)*, the *order details* (VARCHAR)and the *date \_of \_ order(DATE TIMESTAMP)*.

**Retailer**

*Retailer \_order* ()is fully functionally dependent on the retailer id the retailer name and the retailer location.

Partial Dependency

**Supplier**

The *supplier\_ transaction\_ number* (VARCHAR)partially depends on the  *supplier \_ID* (INT)and the *date \_of\_ transaction*.(DATE TIMESTAMP )

The *machine \_output* (INT)partially depends on the *raw \_material\_ input* (VARCHAR)and the *Employee\_ id(INT).*

Transitive Dependency

**Employee**

The *employee\_contacts(INT)* transitively depend on the *Employee ID* (INT)

The *employee \_last name* (STRING)and *employee\_ first name(STRING)* transitively depends on the *Employee \_id(INT)*

**Supplier**

The *supplier \_company \_name(STRING)* transitively depends on the *supplier\_ Id(INT)*

Multi-Valued Dependency

**Machine**

*Machine\_Id* (VARCHAR)is a multi- determinant of the *Employee\_Id(INT)*  and the machine years active.(INT)

**Retailer**

The *Retailer \_Id* (INT)multi-determines the *retailer\_ location* (VARCHAR)and the *retailer \_order* (DATE TIMESTAMP)

**Supplier**

The *Supplier\_Id*(INT) multi-determines the Supplier\_payment of method and the *date\_of \_transaction(DATE TIMESTAMP)*

**Employee**

The *Employee Id(INT)* multi-determines the Employee\_Department(VARCHAR) *Employee\_Firstname(VARCHAR),Employee\_lastname(VARCHAR)*, salary and the *date\_of \_registration(Date TIMESTAMP)*

Join Dependency

**Employee**

(*Employee\_id(INT),Employee\_Firstname(VARCHAR),Employee\_lastname(VARCHAR)*, *Employee\_depertment(VARCHAR),employee\_contacts(INT)*)

Can be decomposed into multiple table

<**employee contacts**>

(*employee\_ID,employee\_firstname,employee\_last\_name*)

<**employee department**>

## Normal Forms

***i.e. 1NF, 2NF, 3NF, BCNF, 4NF, and 5NF***

1. **Employee**

1NF

This table shall be split into two **tables** in which the first has the Employee details and Salary, where the Employee ID is the primary key and

2NF

3NF

BOYCE

4NF

5N

**B.       Raw Material Data**

INF

Two tables shall be obtained from the main table. The first table contains the raw material ID  as the primary key and the raw material Name as the repeating group and the second table shall contain the raw material ID as the foreign key and the Number of hours taken to produce the item as the repeating group.

2NF

There are no partial dependencies in the tables hence the tables are already in the 2NF.

3NF

There are no transitive dependencies hence the tables are already in the 3NF.

BCNF

The determinant of the two tables are Candidate keys hence the relation is already in BCNF.

4NF

There are no multi-valued dependencies in the tables hence the tables are already in the 4NF.

5NF

The original relation is similar to the new relation, in conclusion, relation **Raw Material** Data has a non-additive lossless join dependency.

**C. Supplier Payment Details(SP)**

1NF

This table shall be split into three **tables** in which the first has the Transaction number, Invoice number and Payment details, where the Transaction number and the Invoice Number are the primary key and the Payment Details is the repeating group. The second table will contain the Transaction number as the foreign key and the Method of payment as the repeating group. The third table shall contain the Invoice Number as the foreign key and Date of payment as the repeating group.

2NF

There are no partial dependencies in each of the tables hence the tables are already in the 2NF.

3NF

There are no transitive dependencies hence the tables are already in the 3NF.

BCNF

The determinant of the three tables are Candidate keys hence the relation is already in BCNF.

4NF

There are no multi-valued dependencies in the tables hence the tables are already in the 4NF.

5NF

The original relation is similar to the new relation, in conclusion, relation **Supplier Payment Details** Data has a non-additive lossless join dependency.

**D.        Retailer  Details Data**

1NF

Four tables shall be obtained from the main table. The first table contains the Retailer ID and the Transaction number as the primary key and the Invoice number as the attribute. The second table contains the Transaction number as the foreign key and the Method of Payment as the repeating group. The third table contains the Retailer ID as the foreign key and the Retailer name and the Retailer Location as the repeating groups. Finally the last table contains the Invoice Number as the foreign key and the Date of payment as the repeating group.

2NF

There are no partial dependencies in any of the tables hence the tables are already in the 2NF.

3NF

There are no transitive dependencies hence the tables are already in the 3NF.

BCNF

The determinant of the four tables are Candidate keys hence the relation is already in BCNF.

4NF

The first table will be split into two, hence having **five** tables. The multi-determinant is the Transaction Number, therefore, the first table shall have the Transaction Number and the Retailer ID and the second table shall have the Transaction Number and the Invoice Number.

The third table will be split into two, hence having a total of **six** tables. The multi-determinant is the Retailer ID, henceforth, the tables obtained are: fourth table will have Retailer ID and Retailer Name and the fifth table shall have the Retailer ID and the Retailer Location.

5NF

The original relation is similar to the new relation, in conclusion, relation **Retailers Details** Data has a non-additive lossless join dependency.

**E.         Supplier Details Data**

1NF

Two tables shall be obtained from the main table. The first table contains the Company ID and the Order Invoice Number as the primary key and the Order Details as the repeating group. The final table will contain the Order Invoice Number as the foreign key and the Types of goods as the repeating group.

2NF

There are no partial dependencies in each of the tables hence the tables are already in the 2NF.

3NF

There are no transitive dependencies hence the tables are already in the 3NF.

BCNF

The determinant of the two tables are Candidate keys hence the relation is already in BCNF.

4NF

The first table will be split into two tables , adding up to **three** tables in total. The multi-determinant is the Company ID, therefore, the first table shall contain the Company ID and the Order Invoice Number and the second table shall have the Company ID and the Order Details.

5NF

The original relation is similar to the new relation, in conclusion, relation **Supplier Details** Data has a non-additive lossless join dependency.

**G.        Inventory Details Data**

1NF

Two tables will be obtained from the main table. The first table contains the Item ID as the primary key and date of delivery as the repeating group and the second table contains the item ID as the foreign key and the number of orders as the repeating group.

2NF

There are no partial dependencies in each of the tables hence the tables are already in the 2NF.

3NF

There are no transitive dependencies hence the tables are already in the 3NF.

BCNF

The determinant of the two tables are Candidate keys hence the relation is already in BCNF.

4NF

There are no multi-valued dependencies in the tables hence the tables are already in the 4NF.

5NF

The original relation is similar to the new relation, in conclusion, relation **Inventory Details** Data has a non-additive lossless join dependency.

**H.      Machine Details Data**

1NF

**Two** tables shall be obtained from the main table. The first table will contain the Machine number and the Functionality as the primary key and the number of years active as the repeating group. The second table will have the Functionality as the foreign key and the Machine output as the repeating group.

2NF

There are no partial dependencies in each of the tables hence the tables are already in the 2NF.

3NF

There are no transitive dependencies hence the tables are already in the 3NF.

BCNF

The determinant of the two tables are Candidate keys hence the relation is already in BCNF.

4NF

The first table will be split into two hence having a total of **three** tables. The multi-determinant is the Machine Number, therefore, the first table shall have the Machine Number and the Functionality and the second table shall have the Machine Number and the Number of Years active.

5NF

The original relation is similar to the new relation, in conclusion, relation **Machine Details** Data has a non-additive lossless join dependency.

1. **Order Details Data**

1NF

Two tables will be obtained from the main table. The first table will contain the Order ID and the Order invoice as the primary key, the Order Details and the delivery note Number as the respective attributes. The second table will contain the Order ID  as the foreign key and the Date of Order as the repeating group.

2NF

There are no partial dependencies in each of the tables hence the tables are already in the 2NF

3NF

There are no transitive dependencies hence the tables are already in the 3NF.

BCNF

The determinant of the two tables are Candidate keys hence the relation is already in BCNF.

4NF

The first table will be split into two hence having a total of **three** tables. The multi-determinant is the Order Invoice, therefore, the first table shall have the Order Invoice and the Order ID and the second table shall have the Order Invoice and the Order Details.

5NF

The original relation is similar to the new relation, in conclusion, relation **Order Details** Data has a non-additive lossless join dependency.

Below is the outcome of each table after Normalization

**Employee Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Employee\_ID | Employee\_FirstName | Employee\_LastName | Employee\_Department | Employee\_year of registration | Employee\_position | Employee\_status |
| 12834577 | Samuel | pembe | Operations | 12-04-20 | Operations Manager | Permanent |
| 18902948 | Sarah | Nyota | Finance | 18-04-20 | Analyst | Permanent |
| 23457890 | Dominic | Mawingu | Operations | 20-05-20 | Machine master | Temporary |
| 28901849 | pauline | Jua | Marketing | 21-06-20 | Director | Permanent |

Table 7‑1Employee Details Table

**Salary Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Salary ID | Employee\_ID | Employee\_First Name | Monthly\_salary | Bonus | Net\_Salary |
| H67D45L9 | 12834577 | Samuel | 100,000 | 5% per year | 100,5000 |
| J76B23U5 | 23457890 | Dominic | 10,000 | N/A | N/A |

Table 7‑2 Salary Details Table

**Supplier Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Supplier\_ID | Supplier\_Name | Supplier\_Location | Supplier\_Status |
| T90M56788 | Banana\_Manana | Nakuru | Active |
| X34N00126 | Maize \_Meal | Migori | Active |
| M789HM83 | Millete \_Kenya | Nyandarua | Inactive |

Table 7‑3 Supplier details table

**Raw Materials order details**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Raw\_Material\_ID | Rawmaterial\_Name | Raw material\_Breed | Rawmaterial\_Type | Supplier\_ID | Supplier\_Name | Number of\_bags |
| T89M56H0 | Maize | Corn fibre | Combed | M789HM83 | Maize \_Meal | 60 |
| B45V67N2 | Millet | Finger millet | Cereal | K779HM88 | Millet \_Kenya | 55 |
| H78P77K4 | Sorghum | red | Poaceae | W56TH7V8 | Food Service | 50 |
| S45B28K9 | Soyabeans | Glycine max | Protein | J34JU67IF4 | Organic Food | 45 |

Table 7‑4 Raw Materials order details

**Supplier Payment Details**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SP\_ID | SP\_Amount | SP\_Transaction Number | SP\_Method | SP\_Date | Supplier\_ID | Raw\_Material\_ID |
| M789HM83 | 6000 | VADE0B248932 | bank | 19-08-20 | T90M56788 | T89M56H0 |
| W56TH7V8 | 5500 | OAX345267 | M-pesa | 12-06-20 | X34N00126 | B45V67N2 |

Table 7‑5 Supplier Payment Details

**Raw material Invoice**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Raw Material Invoice ID | Raw\_Material\_ID | Raw Material name | Rawmaterial\_quantity | Rawmaterial\_Price | Raw Material  breed |
| 45678934 | T89M56H0 | Maize | 6kg | 6000 | Corn fibre |
| 09786546 | H78PJ7KL4 | Millet | 5.5kg | 5500 | Finger millet |
| 98742301 | B45VA67N2 | Soyabeans | 4.5kg | 4500 | Glycine max |

Table 7‑6 Raw material Invoice table

**Machine**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Machine  \_ID | Machine\_Type | Rawmaterals Input | Machine\_Status | Machine\_Output | Employee\_ID | Operator\_name | Years\_Active |
| G56V92C2 | Oil mill plant | Soyabeans | Fine | Cooking Oil | 23457890 | Samuel | 7 |
| Q11H76N5 | Wheat Mill | Millet | Great | Flour | 89764536 | Dominic | 5 |
| E43F80Z7 | Milling | Sorghum | Good | Sugar | 67560923 | Ann | 9 |

Table 7‑7 Machine Table

**Product Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Product  \_ID | Product\_Name | Product\_Type | Product\_Number |
| H23T67F8 | Flour | Cereal | 40 |
| D67X42M1 | Sugar | Phocaea | 70 |
| B56O84Z3 | Cooking Oil | Protein | 90 |

Table 7‑8 Product details

**Inventory Details**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Inventory\_ID | Products\_Name | Products\_Quantity | Product d\_manufacture date | Product \_Expiry date | Machine\_ID |
| J87V64U2 | Sugar | 7kg | 20/3/20 | 19-8-24 | E43F80Z7 |
| B76H95S5 | Cooking Oil | 10kg | 28/2/20 | 15-8-25 | G56V92C2 |

Table 7‑9 Inventory Details

**Test Market Details**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test\_Market\_ID | Retailer\_ID | Retailer\_Name | Product\_ID | Product\_Name | Target \_Market | Success\_Rate | Comments |
| A56F87O5 | J87K09V5 | Atta | H23T67F8 | Flour | convenience stores | 56.26% | Its performance in the market is average |
| R45L68F0 | C45X68S6 | Tuskys | D67X42M1 | Sugar | Supermarkets | 67.08% | Has high sales and profits |

Table 7‑10 Test Market Details

**Retailer**

|  |  |  |  |
| --- | --- | --- | --- |
| Retailer\_ID | Retailer\_Name | Retailer\_Location | Retailer\_Status |
| J87K09V5 | Atta | Nyanza | Active |
| C45X68S6 | Tuskys | Nairobi | Permanent |

Table 7‑11 Retailer Details Table

**Retailer Order Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Retailer order\_id | Retailer order\_product | Retailer order\_Quantity | Retailer order\_date | Retailer order date of delivery | Retailer\_ID |
| Y67J09M6 | Cooking Oil | 40kg | 11-01-20 | 18-01-20 | J87K09V5 |
| S45P09B5 | Flour | 60kg | 17-12-20 | 22-12-20 | C45X68S6 |

Table 7‑12 Retail Order Table

**Retailer Order invoice table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Retailer order invoice ID | Retailer order\_id | Retailer Order  Product name | Retailer order Product breed | Retailer Order product quantity | Retailer order product price | Retailer order date |
| L89Y56V5 | H78X56T6 | Sugar | Sucrose | 20kg | 4000 | 28-02-20 |
| B67E45M7 | Y67J09M6 | Cooking Oil | Olive Breed | 40kg | 5500 | 11-01-20 |
| D45E22T5 | S45P09B5 | Flour | Bread flour | 60kg | 6500 | 17-12-20 |

Table 7‑13 Retail Order Invoice Table

**Retailer Payment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Retailer Payment ID | Retailer Payment Transaction Number | Retailer Payment Method | Retailer Payment Date | Retailer Payment Amount | Retailer\_ID |
| F67N34L0 | VADE0B248932 | Bank | 15-01-20 | 6000 | J87K09V5 |
| D78K98P7 | JXR847014 | Mpesa | 28-04-20 | 7500 | H78F56I3 |
| X20C67U5 | ACRAF23DB3C4 | Bank | 19-12-20 | 2500 | C45X68S6 |

Table 7‑14 Retailer Payment Table

# Advanced Database Objects

## a. Storage Engines

# *List the storage engine that each relation will use*

CREATE TABLE **Employee** (i INT) ENGINE = INNODB;

CREATE TABLE **Retailer** (i INT) ENGINE = INNODB; ‘

CREATE TABLE **Test\_Market**(i INT) ENGINE = INNODB;

CREATE TABLE **Supplier**(i INT) ENGINE = INNODB;

CREATE TABLE **Employee\_Salary** (i INT) ENGINE = INNODB;

CREATE TABLE **Supplier\_Payment method** (i INT) ENGINE = INNODB; ‘

CREATE TABLE **Order\_details**(i INT) ENGINE =INNODB;

CREATE TABLE **Product** (i INT) ENGINE = INNODB;

CREATE TABLE **Raw materials** (i INT) ENGINE = INNODB;

For all the INNODB tables the size of an InnoDB table can be up to 64TB ,it balances high reliability and high performance (write performance) and it delivers maximum performance when processing large data volumes hence it has an advantage

## b. Indexes

# *List all the attributes that will be indexed as well as the type of index that will be used*

# Primary key,super key,candidate,composite foreign…hash index-used in memory and NDB,cluster b+ tree index used in a join,r tree, ndb uses hash

i.)Employees

(employee\_Id is the primary key

Employee\_status is B+Tree

Employee\_department is B+Tree

Employee\_firstname is B+Tree

Employee\_lastname is B+Tree

)

ii.)Retailer

( Retailer ID Primary Key

Retailer\_ order is B+Tree

Retailer\_ location is B+Tree

Retailer\_ name is B+Tree

)

iii.)Supplier

(Supplier ID

Supplier\_ name is B+Tree

)

iv.)Order details

(Order ID is Primary Key

Order \_details is B+Tree

Order\_invoice is B+Tree

Order\_ name is B+Tree

)

v)Test Market

(Test market ID is primary Key)

## c. Normal Triggers

# *List all the normal triggers and how they will be used to enforce the database constraints*

# AFTER UPDATE: The trigger is fired after an existing tuple in the relation has been updated to back up data .This applies to when an order is made and when payments are made.

BEFORE DELETE: The trigger is fired before an existing tuple in the relation is saved so that in the case the user undoes the action all records are retained.

AFTER INSERT: The trigger is fired after a new tuple has been inserted into the relation.This applies to when results from the test market are made a view should be made of the analysis of the results and when an employee is registered or a supplier.

## d. Temporal Triggers

# *List all the temporal triggers and how they will be used to enforce the database constraints*

On completion preserve the orders made and the payment transactions and the information in the database for future references on the dashboard.

At the end of six months the experiment the data should be updated and the report of the products displayed on the dashboard .

Every month a reminder to check and add inventory of products should be executed.

At the end of one month a paycheck should be printed.

## e. Procedures

# *List all the procedures and how they will be used to retrieve data required for reports*

# 

A procedure called “proc\_salaries”-This procedure will be used in calculating the net salaries of the employees after bonuses are added if it applies and then retrieving them for printing.

A procedure called “proc\_profits”-This procedure will be used in calculating the profits made out of the sales the company acquired.

A procedure called “proc\_outputs”-This procedure will be used in calculating the efficiency of the production by taking the machine inputs and the outputs and comparing them.

A procedure called “proc\_testing”-This procedure will be used to retrieve the test results of a product in the new market and the related comments.

## f. Functions

# *List all the functions and how they will be used to retrieve data required for reports*

A “FUNC\_salary”-this will be the easiest way for getting the net salaries for each employee using one command .The func\_salary is a **deterministic FUNCTION.**

A “FUNC\_ptofiting”-this function will use if statements to aid in the ease of calculating profits made in the organization.

A “FUNC\_output”-this function will use if statements to aid in the automatic calculation on the machine outputs to know the efficiency.

A “FUNC\_test”-this function will use if statements to retrieve the information collected during the test market of a product.

# 

## g. Views

# *List all the views and how they can be used to provide various displays of the data for different users*

**Sales\_made view**

This is an important view for the people in the finance and Accounting department to calculate profits and perform budgeting for future productions and also draw a conclusion of the products that are selling more than others hence the marketing department might strategize on how to advertise unpopular products .

**Products produced view**

This is an important view because it will show the products made and the inputs used in the production of products and also provide a reference of the ingredients used in making of the products especially when the marketing department are creating labels for each product. It is also important for the operation department to keep track of the quantity and quality of the products.

**Purchase details view**

This view is important because it gives the procurement department a refrence point in that the purchased items will be compared with the invoiced items so that the finance department might approve the payment of the goods orderd .The delivery note will also be seen in this view

**Test Market Results and comments**

This view is important because before wanji launches a product in the market it is tested and based on the success rates it is now purchased by other retailers and customers.The comments given act as feedback since Wanji aims to provide quality and nutritious foods it will aid in achieving this promise.

**Employee payment details view**

The human resource department will require this view in order to calculate the salaries and bonuses of each employee in the organization and also know the working status of the employee since Wanji employs both temporary and permanent staff.

**Retailer Orders View**

It is important to see the products ordered by the Retailers in order to make the delivery to the right location and on time to ensure customer satisfaction and also calculate the cost of goods and the cost of making deliveries in the case of delivery.

# DDL Statements

***Provide the DDL statements that were used to create the database***

**DDL For creating the table ‘Employee Details Data’ and its respective index**

CREATE TABLE `Employee Table` (

`Employee\_ID` `Employee\_ID` INT (8) NOT NULL;

`Employee\_First Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Employee\_Last Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Employee\_Department` VARCHAR (20) NOT NULL,

`Employee\_year of registration` DATE NOT NULL,

`Employee\_position` VARCHAR (30) NOT NULL,

`Employee\_status` VARCHAR (30) NOT NULL,

PRIMARY KEY (`Employee\_ID`),

UNIQUE `UNIQUE` (`Employee\_position`, `Employee\_status`)

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Salary Table’ and its respective index**

CREATE TABLE `Salary Table` (

`Salary ID` CHAR (8) NOT NULL FIRST, ADD PRIMARY KEY (`Salary ID`);

`Employee\_ID` CHAR (8) NOT NULL,

`Employee\_First Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Monthly\_salary` INT NOT NULL,

`Bonus` INT NOT NULL,

`Net\_Salary` INT NOT NULL,

PRIMARY KEY (`Employee\_ID`)

ALTER TABLE `salary table` ADD CONSTRAINT `Constraint fk` FOREIGN KEY (`Employee\_ID`) REFERENCES `employee table`(`Employee\_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Supplier Table’ and its respective index**

CREATE TABLE `Supplier Table` (

`Supplier\_ID` CHAR (8) NOT NULL,

`Supplier\_Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Supplier\_Location` VARCHAR (30) DEFAULT 'NOT NULL',

`Supplier\_Status` VARCHAR (20) NOT NULL,

PRIMARY KEY (`Supplier\_ID`),

UNIQUE ` UNIQUE ` (`Supplier\_Location`)

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Raw Materials order details’ and its respective index**

CREATE TABLE `Raw Materials order details` (

` Raw\_Material\_ID ` CHAR (8) NOT NULL,

`Raw-material\_Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Raw material\_Breed` VARCHAR (20) DEFAULT 'NOT NULL',

`Raw-material\_Type` VARCHAR (30) NOT NULL,

`Supplier\_ID` CHAR (8) NOT NULL,

`Supplier\_Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Number of\_bags` INT (10) NOT NULL,

PRIMARY KEY (`Raw\_Material ID`),

UNIQUE `UNIQUE` (`Supplier\_ID`),

ALTER TABLE `raw materials order details` ADD CONSTRAINT `Foreign key` FOREIGN KEY (`Supplier\_ID`) REFERENCES `supplier table`(`Supplier\_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Supplier Payment Details’ and its respective index**

CREATE TABLE `Supplier Payment Details` (

`SP\_ID` CHAR (8) NOT NULL,

`SP\_Amount` INT (30) NOT NULL,

`SP\_Transaction Number` CHAR (20) NOT NULL,

`SP\_Method` VARCHAR (10) NOT NULL,

`SP\_Date` DATE NOT NULL,

`Supplier\_ID` CHAR (8) NOT NULL,

`Raw\_Material\_ID` CHAR (8) NOT NULL,

PRIMARY KEY (`SP\_ID`),

UNIQUE `UNIQUE` (`SP\_Amount`, `SP\_Date`),

ALTER TABLE `supplier payment details` ADD CONSTRAINT `Foreign key Supplier\_ID` FOREIGN KEY (`Supplier\_ID`) REFERENCES `supplier table`(`Supplier\_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE `supplier payment details` ADD CONSTRAINT `Foreign key Raw\_Material\_ID` FOREIGN KEY (`Raw\_Material\_ID`) REFERENCES `raw materials order details`(`Raw\_Material\_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Raw material Invoice’ and its respective index**

CREATE TABLE `Raw material Invoice` (

`Raw Material Invoice ID` CHAR (8) NOT NULL

` Raw\_Material\_ID ` CHAR (8) NOT NULL,

`Raw Material name` VARCHAR (20) DEFAULT 'NOT NULL',

`Raw-material\_quantity` CHAR (10) NOT NULL,

`Raw-material\_Price` INT (30) NOT NULL,

`Raw Material breed` VARCHAR (30) NOT NULL,

PRIMARY KEY (` Raw\_Material\_ID `),

ALTER TABLE `raw material invoice` ADD CONSTRAINT `Constraint foreign key` FOREIGN KEY (`Raw\_Material\_ID`) REFERENCES `raw materials order details`(`Raw\_Material\_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Machine’ and its respective index**

CREATE TABLE `Machine` (

`Machine \_ID` CHAR (8) NOT NULL,

`Machine\_Type` VARCHAR (20) DEFAULT 'NOT NULL',

`Rawmaterals Input` VARCHAR (20) NOT NULL,

`Machine\_Status` VARCHAR (10) DEFAULT 'NOT NULL',

`Machine\_Output` VARCHAR (20) NOT NULL,

`Employee\_ID` INT (8) NOT NULL,

`Operator\_name` VARCHAR (10) NOT NULL,

`Years\_Active` INT (10) NOT NULL,

PRIMARY KEY (`Machine \_ID`),

UNIQUE `UNIQUE` (`Employee\_Id`, `Years\_Active`),

ALTER TABLE `machine` ADD CONSTRAINT `FK Constraint` FOREIGN KEY (`Employee\_ID`) REFERENCES `employee table`(`Employee\_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Product Details’ and its respective index**

CREATE TABLE `Product Details` (

`Product \_ID` CHAR (8) NOT NULL,

`Product\_Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Product\_Type` VARCHAR (30) NOT NULL,

`Product\_Number` INT (10) NOT NULL,

PRIMARY KEY (`Product \_ID`)

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Inventory Details’ and its respective index**

CREATE TABLE `Inventory Details` (

`Inventory\_ID` CHAR (8) NOT NULL,

`Products\_Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Products\_Quantity` CHAR (10) NOT NULL,

`Product d\_manufacture date` DATE NOT NULL,

`Product \_Expiry date` DATE NOT NULL,

`Machine\_ID` INT (8) NOT NULL,

PRIMARY KEY (`Inventory\_ID`),

UNIQUE `UNIQUE` (`Product d\_manufacture date`, `Product \_Expiry date`)

ALTER TABLE `inventory details` ADD CONSTRAINT `foreign key constraint` FOREIGN KEY (`Machine\_ID`) REFERENCES `machine` (`Machine \_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Test Market Details’ and its respective index**

CREATE TABLE `Test Market Details` (

`Test\_Market\_ID` CHAR (8) NOT NULL,

`Retailer\_ID` CHAR (8) NOT NULL,

`Retailer\_Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Product\_ID` CHAR (8) NOT NULL,

`Product\_Name` VARCHAR (10) DEFAULT 'NOT NULL',

`Target \_Market` VARCHAR (20) NOT NULL,

`Success\_Rate` DECIMAL NOT NULL,

`Comments` TEXT (200) NOT NULL,

PRIMARY KEY (`Test\_Market\_ID`)

ALTER TABLE `inventory details` ADD CONSTRAINT `foreign key constraint` FOREIGN KEY (`Machine\_ID`) REFERENCES `machine`(`Machine \_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Retailer’ and its respective index**

CREATE TABLE `Retailer` (

`Retailer\_ID` CHAR (8) NOT NULL,

`Retailer\_Name` VARCHAR (20) DEFAULT 'NOT NULL',

`Retailer\_Location` VARCHAR (20) NOT NULL,

`Retailer\_Status` VARCHAR (30) NOT NULL,

PRIMARY KEY (`Retailer\_ID`)

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Retailer Order Table’ and its respective index**

CREATE TABLE `Retailer Order Table` (

`Retailer order\_id` CHAR (8) NOT NULL,

`Retailer order\_product` VARCHAR (30) DEFAULT 'NOT NULL',

`Retailer order\_Quantity` CHAR (10) NOT NULL,

`Retailer order\_date` DATE NOT NULL,

`Retailer order date of delivery` DATE NOT NULL,

`Retailer\_ID` CHAR (8) NOT NULL,

PRIMARY KEY (`Retailer order\_id`),

ALTER TABLE `retailer order table` ADD CONSTRAINT `Foreign key Retailer ID` FOREIGN KEY (`Retailer\_ID`) REFERENCES `retailer`(`Retailer\_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Retailer Order invoice table’ and its respective index**

CREATE TABLE `Retailer Order invoice table` (

`Retailer order\_id` CHAR (8) NOT NULL,

`Retailer Order Product name` VARCHAR (20) DEFAULT 'NOT NULL',

`Retailer order Product breed` VARCHAR (20) DEFAULT 'NOT NULL',

`Retailer Order product quantity` CHAR (10) NOT NULL,

`Retailer order product price` INT NOT NULL, `

Retailer order date` DATE NOT NULL,

PRIMARY KEY (`Retailer order\_id`),

ALTER TABLE `retailer order invoice table` ADD CONSTRAINT `FK Retailer order invoice ID` FOREIGN KEY (`Retailer order\_id`) REFERENCES `retailer order table` (`Retailer order\_id`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the table ‘Retailer Payment’ and its respective index**

CREATE TABLE `Retailer Payment` (

`Retailer Payment ID` CHAR (8) NOT NULL,

`Retailer Payment Transaction Number` CHAR (20) NOT NULL,

`Retailer Payment Method` VARCHAR (10) DEFAULT 'NOT NULL',

`Retailer Payment Date` DATE NOT NULL,

`Retailer Payment Amount` INT NOT NULL,

`Retailer\_ID` CHAR (8) NOT NULL,

PRIMARY KEY (`Retailer Payment ID`),

UNIQUE `UNIQUE` (`Retailer Payment Transaction Number`),

ALTER TABLE `retailer payment` ADD CONSTRAINT `Foreign Key Retailer Payment` FOREIGN KEY (`Retailer\_ID`) REFERENCES `retailer`(`Retailer\_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT;

) ENGINE = InnoDB DEFAULT CHARSET=utf8mb4;

**DDL For creating the procedure**

CREATE PROCEDURE `Resolve Issue` (IN input INT)

BEGIN SELECT \* FROM Support inner join ` Raw material Invoice ` on ` Raw Material Invoice ID ` = ` Raw Material Invoice No.` WHERE `support.Issue Resolved?` = @input;

END

**DDL For creating the function**

CREATE FUNCTION get\_total\_products (in \_year PLS\_INT)

RETURN NUMBER

IS

First\_total\_product NUMBER: = 0;

BEGIN

--get total sales

SELECT SUM (Product\_Number \* Products\_Quantity)

INTO Inventory Details

FROM Product Details

INNER JOIN Inventory Details Using (Products\_Quantity)

GROUP BY EXTRACT (YEAR FROM Product d\_manufacture date)

HAVING EXTRACT (YEAR FROM Product d\_manufacture date) = IN YEAR;

RETURN total\_products

END;

**DDL For creating the Temporal Trigger**

CREATE EVENT inactivate\_Key

ON SCHEDULE

EVERY 1 DAY

STARTS '2020-02-28 00:00:00'

DO

UPDATE `Operation Key`SET `Operation Key.Key Status` = '0' where DATE (`Expiry Date`) <= CURRENT\_DATE ();

END;

**DDL For creating the Normal Trigger**

CREATE TRIGGER hr.salary\_check

BEFORE INSERT OR UPDATE OF salary, Employee\_ID ON Salary Table

FOR EACH ROW

WHEN (new. Employee\_ID <> 'AD\_VP')

pl/sql\_block